IBM Macros for IBM EMM
Version 9 Release 0
January 15, 2013

User's Guide

IBM

Contents

Chapter 1. Using Macros in IBM Campaign

This chapter provides usage information about the macros available for use in IBM® Campaign. All IBM Campaign users should read this chapter before attempting to use the remainder of this guide.

Macro Function Summaries

The tables in this section summarize the macro functions by category.

Important: Not all macros listed in this guide are available in IBM Campaign.

Macros available only in IBM PredictiveInsight are denoted by this icon:

Macros available only in IBM Interact are denoted by this icon:

Detailed reference pages for each macro function are provided in alphabetical order starting in Chapter 4, "IBM Macros Reference," on page 31. "Macro Function Parameters" on page 26 provides information about the macro function input parameters.

Related information:

"Statistical Functions"

"Math and Trigonometric Functions" on page 2

"String Functions" on page 5

"Date and Time Functions" on page 6

"Grouping Functions" on page 7

"Miscellaneous Functions" on page 8

Statistical Functions

Macro Name	Returns	Description
AVG	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
COUNT	Single value in a new column.	Counts the number of values in a specified data range
MAX	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the maximum of a range of cells

Macro Name	Returns	Description
MEAN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
MIN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the minimum of a range of cells
STDV or STDEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the standard deviation of a range of cells
VARIANCE	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the variance of a range of cells

Math and Trigonometric Functions

Macro Name	Returns	Description
ABS	One column for each input column	Computes the absolute value of the contents of the specified data range
ACOS	One column for each input column	Computes the arc cosine of the contents of the specified data range
ACOT	One column for each input column	Computes the arc cotangent of the contents of the specified data range
ASIN	One column for each input column	Computes the arc sine of the contents of the specified data range
ATAN	One column for each input column	Computes the arc tangent of the contents of the specified data range
AVG	One column for each input column	Calculates the arithmetic mean or average of the cells in the specified data range

Macro Name	Returns	Description
BETWEEN	One column for each input column	Compares two values to determine whether the provided value is between two other values
CEILING	One column for each input column	Computes the ceiling of each value in the specified data range
COLUMN	One column for each input column	Creates new columns, vertically concatenating the input values in each column
COS	One column for each input column	Computes the cosine of the contents of the specified data range
COSH	One column for each input column	Computes the hyperbolic cosine of the contents of the specified data range
COT	One column for each input column	Computes the cotangent of the contents of the specified data range
COUNT	One column containing a single value	Counts the number of cells containing values in the specified data range
EXP	One column for each input column	Computes the natural number (e) raised to the contents of each cell in the specified data range
FACTORIAL	One column for each input column	Computes the factorial of each value in the specified data range
FLOOR	One column for each input column	Computes the floor of each value in the specified data range
FRACTION	One column for each input column	Returns the fractional part of each value in the specified data range
INT	One column for each input column	Computes the integer value (rounded down) of the contents of the specified data range
INVERSE	One column for each input column	Computes the negative of the contents of the specified data range
LN	One column for each input column	Computes the natural log of the contents of the specified data range
LOG	One column for each input column	Computes the natural log of the contents of the specified data range
LOG2	One column for each input column	Computes the log base2 of the contents of the specified data range

Macro Name	Returns	Description
LOG10	One column for each input column	Computes the log base10 of the contents of the specified data range
MAX	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the maximum of a range of cells
MEAN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
MIN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the minimum of a range of cells
RANDOM	One column with the specified number of values	Returns the specified number of random numbers
RANDOM_GAUSS	One column with the specified number of values	Returns the specified number of random values from a Gaussian distribution
ROUND	One column for each input column	Computes the rounded value of the contents of the specified data range
SIGN	One column for each input column	Computes the sign (positive or negative) of the values in the specified data range
SIN	One column for each input column	Computes the sine of the contents of the specified data range
SINH	One column for each input column	Computes the hyperbolic sine of the contents of the specified data range
SQRT	One column for each input column	Computes the square root of the contents of the specified data range
STDV or STDEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the standard deviation of a range of cells

Macro Name	Returns	Description
SUM	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the sum of a range of cells
TAN	One column for each input column	Computes the tangent of the contents of the specified data range
TANH	One column for each input column	Computes the hyperbolic tangent of the contents of the specified data range
TOTAL	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the sum of a range of cells
TRUNCATE	One column for each input column	Returns the non-fractional part of each value in the specified data range
VARIANCE	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the variance of a range of cells

String Functions

Macro Name	Returns	Description
FORMAT	One column for each input column	Provides output formatting control for both numbers and strings (such as output width, alignment, numeric precision, decimal point symbol, grouping symbol, and so on). Returns the formatted output string.
LIKE	One column for each input column	Determines whether a text string matches a specified pattern
LOWER	One column for each input column	Converts string value to lowercase
LTRIM	One column for each input column	Removes leading space characters from each string value
NUMBER	One column for each input column	Converts ASCII text strings for times and dates to numeric values

Macro Name	Returns	Description
POSITION	One column for each input column	Returns the starting position of a pattern in a text string
RTRIM	One column for each input column	Removes trailing space characters from each string value
STRING_CONCAT	One column with a value for each row of the shortest input column	Concatenates text strings from the specified data ranges
STRING_HEAD	One column for each input column	Returns the first *n* characters of each string in the specified data range
STRING_LENGTH	One column for each input column	Returns the length of each string in the specified data range
STRING_PROPER	One column for each input column	Converts each string value by changing the first letter or any letter that follows a white space character or symbol (other than underscore) into uppercase, and all other characters into lowercase
STRING_SEG	One column for each input column	Returns the string segment between two specified indexes
STRING_TAIL	One column for each input column	Returns the last *n* characters of each string in the specified data range
SUBSTR or SUBSTRING	One column for each input column	Returns characters from a string from a starting position
UPPER	One column for each input column	Converts string value to uppercase

Date and Time Functions

Macro Name	Returns	Description
ADD_MONTHS	One column for each input column	Returns a date after adding the specified number of months
CURRENT_DATE	One column for each input column	Returns the current date in `format`
CURRENT_DAY	One column for each input column	Returns the current day of the month as a number from 1-31
CURRENT_JULIAN	One column for each input column	Returns the Julian number for the current date
CURRENT_MONTH	One column for each input column	Returns the current month of the year as a number from 1-12

Macro Name	Returns	Description
CURRENT_TIME	One column for each input column	Returns the current time as a string
CURRENT_WEEKDAY	One column for each input column	Returns the current weekday of the month as a number from 0-6
CURRENT_YEAR	One column for each input column	Returns the current year as a number
DATE	One column for each input column	Converts a date string into a Julian date
DATE_FORMAT	One column for each input column	Transforms date formats
DATE_JULIAN	One column for each input column	Returns the Julian date
DATE_STRING	One column for each input column	Returns the date string of the Julian date
DAY_BETWEEN	One column for each input column	Returns the number of days between two dates
DAY_FROMNOW	One column for each input column	Returns the number of days from the current date to the specified date
DAY_INTERVAL	One column for each input column	Returns the number of days between two dates
DAYOF	One column for each input column	Returns the day of the month as a number
MONTHOF	One column for each input column	Returns the month of the year as a number
WEEKDAY	One column for each input column	Converts ASCII text date strings to the day of the week
WEEKDAYOF	One column for each input column	Returns the weekday of the week as a number
YEAROF	One column for each input column	Returns the year as a number

Grouping Functions

Macro Name	Returns	Description
GROUPBY	One new column with a value for each row	Summarizes across multiple rows of data within a group
GROUPBY_WHERE	One new column with a value for each row	Summarizes across multiple rows of data that meet a specified condition and are within a group

Miscellaneous Functions

Macro Name	Returns	Description
IF	One column with a value for each row of the shortest input column	Begins a conditional if-then-else statement
ISERROR	One column with a value for each row of the shortest input column	Returns a one if any value in the input row contains an error (???) cell, else zero
ISEVEN	One column for each input column	Tests if input values are even (that is, divisible by two)
ISODD	One column for each input column	Tests if input values are odd (that is, not divisible by two)
ROWNUM	One column for each input column	Generates sequential numbers from one to the number of records

Macro Function Parameters

This section describes the parameters and usage for macro functions in IBM Campaign.

Format Specifications

This section describes the format for some commonly used parameters. It applies to all references to these parameters by macro function specifications in this chapter.

data

The data parameter represents a data column for a macro function to act upon. It can be a constant or a field. See the specific macro function for details.

Note: IBM Campaign does not support calculations on multiple fields at the same time, or on a subset of rows, as can be done in IBM PredictiveInsight.

Some other parameter names also use the same format as data. The descriptions of these parameters reference this section and format.

keyword

The keyword parameter controls the behavior of the macro function. It indicates that a keyword can be specified (if it is omitted, the default is used). The keyword choices are listed for each individual macro function in the following form:

{choice1 | choice2 | choice3}

Select the keyword choice providing the wanted behavior. The default choice is shown in bold. For example, given the following options:

{**RADIANS** | DEGREES}

The following macro functions are both valid:

COS(V1, RADIANS) COS(V1, DEGREES)

Note: Many macro functions take the keyword parameters {ALL | COL | ROW}. These keywords do not apply in IBM Campaign because the input data is always a single column or field. The macro always behaves as if the COL keyword were specified. Therefore, you do not need to specify these keywords when using IBM Campaign.

Use of Constants

Most of the macro function parameters can take numeric constants or expressions evaluating to a numeric constant (macro functions operating on strings can take string constants).

In macro functions performing record-by-record operations (for example, adding two numeric columns), using a constant is equivalent to specifying a column containing that constant value in each row. Essentially, when a constant is provided as an input parameter, the constant is expanded to same length as the input.

Some macro functions can take ASCII text strings and numeric constants. Parameters that can accept both numeric constants and ASCII text strings are noted in the "Parameters" section of each macro function.

Examples are provided in the following table.

Function Definition	How the Constant Is Interpreted
PERCENT_UTILIZ = (CURR_BAL*100)/ CREDIT_LIM	The constant 100 is interpreted as a column containing the same number of rows as the column CURR_BAL, with each row containing the constant 100. The derived field PERCENT_UTILIZ will contain each value of CURR_BAL multiplied by 100 and divided by each value of CREDIT_LIM.
NAME = STRING_CONCAT ("Mr. ", LAST_NAME)	The constant "Mr." is interpreted as a column containing the same number of rows as the column LAST_NAME, with each row containing the constant "Mr.". The derived field NAME will contain each of the text strings in LAST_NAME prefaced by "Mr.".

Chapter 2. Using Macros in IBM Interact

This section provides usage information about the macros available for use in IBM Interact. All IBM Interact users should read this section before attempting to use the remainder of this guide.

Macro Function Summaries

The tables in this section summarize the macro functions by these categories:

Important: Not all macros listed in this guide are available in IBM Interact. Macros available only in IBM PredictiveInsight are denoted by this icon:

Detailed reference pages for each macro function are provided in alphabetical order starting in Chapter 4, "IBM Macros Reference," on page 31. "Macro Function Parameters" on page 26 provides information about the macro function input parameters.

Related information:

"Statistical Functions"

"Math and Trigonometric Functions" on page 12

"String Functions" on page 13

"Date and Time Functions" on page 13

"Miscellaneous Functions" on page 14

Statistical Functions

Macro Name	Returns	Description
AVG	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
MAX	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the maximum of a range of cells
MEAN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells

Macro Name	Returns	Description
MIN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the minimum of a range of cells
STDV or STDEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the standard deviation of a range of cells

Math and Trigonometric Functions

Macro Name	Returns	Description
AVG	One column for each input column	Calculates the arithmetic mean or average of the cells in the specified data range
MAX	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the maximum of a range of cells
MEAN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
MIN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the minimum of a range of cells
STDV or STDEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the standard deviation of a range of cells

Macro Name	Returns	Description
SUM	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the sum of a range of cells
TOTAL	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the sum of a range of cells

String Functions

Macro Name	Returns	Description
LIKE	One column for each input column	Determines whether a text string matches a specified pattern
LOWER	One column for each input column	Converts string value to lowercase
LTRIM	One column for each input column	Removes leading space characters from each string value
NUMBER	One column for each input column	Converts ASCII text strings for times and dates to numeric values
RTRIM	One column for each input column	Removes trailing space characters from each string value
STRING_CONCAT	One column with a value for each row of the shortest input column	Concatenates strings from the specified data ranges
SUBSTR or SUBSTRING	One column for each input column	Returns characters from a string from a starting position
UPPER	One column for each input column	Converts string value to uppercase

Date and Time Functions

Macro Name	Returns	Description
ADD_MONTHS	One column for each input column	Returns a date after adding the specified number of months
CURRENT_DATE	One column for each input column	Returns the current date in `format`

Macro Name	Returns	Description
CURRENT_DAY	One column for each input column	Returns the current day of the month as a number from 1-31
CURRENT_MONTH	One column for each input column	Returns the current month of the year as a number from 1-12
CURRENT_WEEKDAY	One column for each input column	Returns the current weekday of the month as a number from 0-6
CURRENT_YEAR	One column for each input column	Returns the current year as a number
DATE	One column for each input column	Converts a date string into a Julian date
DATE_FORMAT	One column for each input column	Transforms date formats

Miscellaneous Functions

Macro Name	Returns	Description
EXTERNALCALLOUT	Values as defined by the custom application written with the ExternalCallout API	Calls a custom application written with the ExternalCallout API. For more information, see the *IBM InteractDeveloper's Guide*.
IF	One column with a value for each row of the shortest input column	Begins a conditional if-then-else statement

Macro Function Parameters

This section describes the parameters and usage for macro functions in IBM Interact.

Format Specifications

This section describes the format for some commonly used parameters. It applies to all references to these parameters by macro function specifications in this section.

data

The data parameter represents a data column for a macro function to act upon. It can be a constant or a field. See the specific macro function for details.

Note: IBM Interact does not support calculations on multiple fields at the same time, or on a subset of rows, as can be done in IBM PredictiveInsight.

Some other parameter names also use the same format as data. The description of these parameters refers to this section and format.

keyword

The keyword parameter controls the behavior of the macro function. It indicates that a keyword can be specified (if it is omitted, the default is used). The keyword choices are listed for each individual macro function in the following form:

{**choice1** | choice2 | choice3}

Select the keyword choice providing the wanted behavior. The default choice is shown in bold. For example, given the following options:

{**RADIANS** | DEGREES}

The following macro functions are both valid:

COS(V1, RADIANS) COS(V1, DEGREES)

Note: Many macro functions take the keyword parameters {ALL | COL | ROW}. These keywords do not apply in IBM Interact because the input data is always a single column or field. The macro always behaves as if the COL keyword were specified. Therefore, you do not need to specify these keywords when using IBM Interact.

Use of Constants

Most of the macro function parameters can take numeric constants or expressions evaluating to a numeric constant (macro functions operating on strings can take string constants).

In macro functions performing record-by-record operations (for example, adding two numeric columns), using a constant is equivalent to specifying a column containing that constant value in each row. Essentially, when a constant is provided as an input parameter, the constant is expanded to same length as the input.

Some macro functions can take ASCII text strings as well as numerical constants. Parameters that can accept both numeric constants and ASCII text strings are noted in the "Parameters" section of each macro function.

Examples are provided in the following table.

Function Definition	How the Constant Is Interpreted
PERCENT_UTILIZ = (CURR_BAL*100)/ CREDIT_LIM	The constant 100 is interpreted as a column containing the same number of rows as the column CURR_BAL, with each row containing the constant 100. The derived field PERCENT_UTILIZ will contain each value of CURR_BAL multiplied by 100 and divided by each value of CREDIT_LIM.
NAME = STRING_CONCAT ("Mr.", LAST_NAME)	The constant "Mr." is interpreted as a column containing the same number of rows as the column LAST_NAME, with each row containing the constant "Mr.". The derived field NAME will contain each of the text strings in LAST_NAME prefaced by "Mr.".

Note: Constants, such as DT_DELIM_M_D_Y, require single quotation marks.

Chapter 3. Using Macros in IBM PredictiveInsight

This chapter provides usage information about the macros available for use in IBM PredictiveInsight. All IBM PredictiveInsight users should read this chapter before attempting to use the remainder of this guide.

Macro Function Summaries

The tables in this section summarize the macro functions by these categories:

Important: Not all macros listed in this guide are available in IBM PredictiveInsight. Macros available only in IBM Interact are denoted by this icon:

Detailed reference pages for each macro function are provided in alphabetical order starting in Chapter 4, "IBM Macros Reference," on page 31. "Macro Function Parameters" on page 26 provides information about the macro function input parameters.

Related information:

"Statistical Functions"

"Math and Trigonometric Functions" on page 18

"Engineering Functions" on page 22

"String Functions" on page 23

"Date and Time Functions" on page 24

"Grouping Functions" on page 7

Statistical Functions

Macro Name	Returns	Description
AVG	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
AVG_DEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the average deviation of a range of cells
HISTOGRAM	Single value in a new column	Computes the histogram of a specified data range using provided bin boundaries

Macro Name	Returns	Description
KURTOSIS	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the kurtosis of a range of cells
MEAN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the arithmetic mean or average of a range of cells
SKEW	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the skew of the distribution of a range of cells
STAT	Variable number of columns (see macro)	Computes the first through fourth moments of the specified data range
STDV or STDEV	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the standard deviation of a range of cells
VARIANCE	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the variance of a range of cells
XTAB	One column for each distinct value in the second input parameter with a number of rows equal to the number of distinct values in the first input parameter	Calculates distinct values in two columns and returns the COUNTOF each combination

Math and Trigonometric Functions

Macro Name	Returns	Description
ABS	One column for each input column	Computes the absolute value of the contents of the specified data range
ACOS	One column for each input column	Computes the arc cosine of the contents of the specified data range

Macro Name	Returns	Description
ACOT	One column for each input column	Computes the arc cotangent of the contents of the specified data range
ASIN	One column for each input column	Computes the arc sine of the contents of the specified data range
ATAN	One column for each input column	Computes the arc tangent of the contents of the specified data range
CEILING	One column for each input column	Computes the ceiling of each value in the specified data range
COS	One column for each input column	Computes the cosine of the contents of the specified data range
COSH	One column for each input column	Computes the hyperbolic cosine of the contents of the specified data range
COT	One column for each input column	Computes the cotangent of the contents of the specified data range
COUNT	One column containing a single value	Counts the number of cells containing values in the specified data range
COV	Single value in one or more columns	Computes the covariance of two input ranges
DERIVATIVE	One column for each input column	Computes the derivative of the values in the specified data range
DIV	One column for each input column	Divides one specified data range by another
EQ	One column for each input column	Returns TRUE if one data range is equal to another
EXP	One column for each input column	Computes the natural number (e) raised to the contents of each cell in the specified data range
FACTORIAL	One column for each input column	Computes the factorial of each value in the specified data range
FLOOR	One column for each input column	Computes the floor of each value in the specified data range
FRACTION	One column for each input column	Returns the fractional part of each value in the specified data range
GAUSS	One column for each input column	Calculates the Gaussian of the values in the specified data range

Macro Name	Returns	Description
GAUSS_AREA	One column for each input column	Calculates the area under the Gaussian of the values in the specified data range
GE	One column for each input column	Returns TRUE if one data range is greater than or equal to another
GT	One column for each input column	Returns TRUE if one data range is greater than another
INT	One column for each input column	Computes the integer value (rounded down) of the contents of the specified data range
INTEGRAL	One column for each input column	Computes the integral of the values in the specified data range
INVERSE	One column for each input column	Computes the negative of the contents of the specified data range
ISEVEN	One column for each input column	Tests if input values are even (that is, divisible by two)
ISODD	One column for each input column	Tests if input values are odd (that is, not divisible by two)
LE	One column for each input column	Returns TRUE if one data range is less than or equal to another
LN	One column for each input column	Computes the natural log of the contents of the specified data range
LOG	One column for each input column	Computes the natural log of the contents of the specified data range
LOG2	One column for each input column	Computes the log base2 of the contents of the specified data range
LOG10	One column for each input column	Computes the log base10 of the contents of the specified data range
LT	One column for each input column	Returns TRUE if one data range is less than another
MAX	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the maximum of a range of cells

Macro Name	Returns	Description
MIN	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the minimum of a range of cells
MINUS	One column for each input column	Subtracts one specified data range from another
MOD	One column for each input column	Computes the modulo of the contents of the specified data range
MULT	One column for each input column	Multiplies the contents of two data ranges
NE	One column for each input column	Returns TRUE if one data range is not equal to another
PLUS	One column for each input column	Adds the contents of two data ranges
POW	One column for each input column	Computes a base value raised to the specified exponential powers
RANDOM	One column with the specified number of values	Returns the specified number of random numbers
RANDOM_GAUSS	One column with the specified number of values	Returns the specified number of random values from a Gaussian distribution
ROUND	One column for each input column	Computes the rounded value of the contents of the specified data range
SIGN	One column for each input column	Computes the sign (positive or negative) of the values in the specified data range
SIN	One column for each input column	Computes the sine of the contents of the specified data range
SINH	One column for each input column	Computes the hyperbolic sine of the contents of the specified data range
SQRT	One column for each input column	Computes the square root of the contents of the specified data range
SUM	Single value in a new column for the ALL keyword; one column with a single value for each input column for the COL keyword; one column with a value for each row for the ROW keyword.	Computes the sum of a range of cells
TAN	One column for each input column	Computes the tangent of the contents of the specified data range

Macro Name	Returns	Description
TANH	One column for each input column	Computes the hyperbolic tangent of the contents of the specified data range
TOTAL	Single value in a new column for the `ALL` keyword; one column with a single value for each input column for the `COL` keyword; one column with a value for each row for the `ROW` keyword.	Computes the sum of a range of cells
TRUNCATE	One column for each input column	Returns the non-fractional part of each value in the specified data range

Engineering Functions

Macro Name	Returns	Function
DELAY	One column for each input column	Returns the input columns' values delayed by a specified number of time steps
GRID	One column for each input column	Returns a grid of all possible value combinations (one per row)
LAG	One column for each input column	Returns the input columns' values lagging by a specified number of time steps
NORM_MAXMIN	One column for each input column	Computes the min/max normalization of a data range
NORM_SIGMOID	One column for each input column	Computes the sigmoidal normalization of a data range
NORM_ZSCORE	One column for each input column	Computes the z-score normalization of a data range
PCA	One column for each input column, plus one	Computes the eigenvectors for principal components of the specified data range
PCA_FEATURES	One column for each input column, plus one	Extracts *n*features from the specified data range
SAMPLE_RANDOM	One column for each input column	Returns columns of n cells, each containing a random sample from the specified data range
SLIDE_WINDOW	Number of input columns times the `width` parameter	Creates a pattern from a specified window and slides it down to create the next pattern

Macro Name	Returns	Function
SORT	One column for each input column	Sorts the values in the specified data range in ascending or descending order
SUBSAMPLE	One column for each input column	Reduces data by returning every n-th row value

String Functions

Macro Name	Returns	Function
DISTINCT	One column for each input column	
FORMAT	One column for each input column	Provides output formatting control for both numbers and strings (for example, output width, alignment, numeric precision, decimal point symbol, grouping symbol, and so on). Returns the formatted output string.
LIKE	One column for each input column	Determines whether a text string matches a specified pattern
NUMBER	One column for each input column	Converts ASCII text strings for times and dates to numerical values
POSITION	One column for each input column	Returns the starting position of a pattern in a text string
STRING_CONCAT	One column with a value for each row of the shortest input column	Concatenates text strings from the specified data ranges
STRING_HEAD	One column for each input column	Returns the first n characters of each string in the specified data range
STRING_LENGTH	One column for each input column	Returns the length of each string in the specified data range
STRING_SEG	One column for each input column	Returns the string segment between two specified indexes
STRING_TAIL	One column for each input column	Returns the last n characters of each string in the specified data range
SUBSTR	One column for each input column	Returns characters from a string from a starting position
SUBSTRING	One column for each input column	Returns characters from a string from a starting position

Date and Time Functions

Macro Name	Returns	Description
ADD_MONTHS	One column for each input column	Returns a date after adding the specified number of months
CURRENT_DATE	One column for each input column	Returns the current date in `format`
CURRENT_DAY	One column for each input column	Returns the current day of the month as a number from 1-31
CURRENT_JULIAN	One column for each input column	Returns the Julian number for the current date
CURRENT_MONTH	One column for each input column	Returns the current month of the year as a number from 1-12
CURRENT_TIME	One column for each input column	Returns the current time as a string
CURRENT_WEEKDAY	One column for each input column	Returns the current weekday of the month as a number from 0-6
CURRENT_YEAR	One column for each input column	Returns the current year as a number
DATE	One column for each input column	Converts a date string into a Julian date
DATE_FORMAT	One column for each input column	Transforms date formats
MONTHOF	One column for each input column	Returns the month of the year as a number
WEEKDAY	One column for each input column	Converts ASCII text date strings to the day of the week
WEEKDAYOF	One column for each input column	Returns the weekday of the week as a number
YEAROF	One column for each input column	Returns the year as a number

Grouping Functions

Macro Name	Returns	Description
GROUPBY	One new column with a value for each row	Summarizes across multiple rows of data within a group
GROUPBY_WHERE	One new column with a value for each row	Summarizes across multiple rows of data that meet a specified condition and are within a group

Miscellaneous Functions

Macro Name	Returns	Function
BUFFER	One column for each input column	Copies the input data range, and updates dynamically
COLUMN	One column for each input column	Creates new columns, vertically concatenating the input values in each column
CONSTANT	One column for each input column	Copies the input data range one time (no dynamic update)
COUNT_DIFF	Two columns	Returns each unique value in the input with a count of the number of times that value appeared
CV_FOLDS	One column with a value for each row of the shortest input column	Divides input patterns into n folds of data for cross-validation
DATALINK	One column for each column of linked data	Creates an internal link to data in a IBM PredictiveInsight spreadsheet
DDELINK	One column for each column of linked data	Creates an external link to data from another Windows application
DECIMATE	MAX_VALUE columns (one column for each output value)	Decimates a column of numbers into multiple columns where a one indicates the index value
EXTRACT	One column for each input column	Extracts rows given the values in a predicate column
IF	One column with a value for each row of the shortest input column	Begins a conditional statement if-then-else statement
INIT	One column	Initializes previous time step values for a recursive function
ISERROR	One column with a value for each row of the shortest input column	Returns a one if any value in the input row contains an error (???) cell, else zero
ISMEMBER	One column for each input column	Tests an input range against a "table" of values, returning one if a value is contained in the table, else zero
MAXINDEX	One column with a value for each row of the shortest input column	Returns the column index of the nth (first, second, third, and so on) maximum value for each row of the specified column

Macro Name	Returns	Function
RANK	One column for each input column	Divides data into nbins (default 10) groups, each with approximately an equal number of distinct values, and returns the group into which each row of data falls
SORT	One column for each input column	Sorts the specified data range in ascending or descending order

Notes on Macro Reference Pages

Each of the available macro functions is described in the *Macro Function Summaries* section for each product. Before starting work with a macro, review the syntax conventions in the preface.

Simple examples for each macro function are provided. In these examples, any additional columns created are named VX, VY, VZ, and so on. The actual names used in the spreadsheet depend on your specific situation.

Macro Function Parameters

This section describes the parameters and usage for macro functions in IBM PredictiveInsight.

Format Specifications

This section describes the format for some commonly used parameters. It applies to all references to these parameters by macro function specifications in this chapter.

data

The data parameter represents a data range for a macro function to act upon. It can typically be a constant, a column, or a cell range (see the specific macro function for details). The format for the data parameter is as follows:

begin_data [: end_data]

Where begin_data can be a constant (for example, 10.2), the name of a column (for example, V1), or a cell range (for example, V1[1:100]). The end_data parameter is optional. If it is provided, begin_data is used as a starting point and must be a column or a cell range. The ending point is specified by end_data.

Note: Some other parameter names also use the same format as data. The description of these parameters refer to this section and format.

keyword

The keyword parameter controls the behavior of the macro function. It indicates that a keyword can be specified (if it is omitted, the default is used). The keyword choices are listed for each individual macro function in the following form:

{   **choice1**| choice2 | choice3}

Select the keyword choice providing the wanted behavior. The default choice is shown in bold. For example, given the following format:

```
{          ALL| COL | ROW}
```

The following macro functions are all valid:

```
AVG(V1:V5) AVG(V1:V5, ALL) AVG(V1:V5, COL) AVG(V1:V5, ROW)
```

Use of Cell and Column Ranges

Cell and column ranges can be provided as inputs for most parameters of macro functions. They must abide by the following rules:

- Number of columns must match
- Returned values begin in the first cell
- Cell ranges automatically filled with zeros

Number of Columns Must Match

When two or more data ranges are provided as input and column-wise computations are performed, the two data ranges must contain the same number of columns. Otherwise, only the dimensions of the smaller data range are used (some macro functions will signal an error). If the data ranges contain a different number of rows, most macro functions perform computations up to and including the last row of the shortest column.

- For example, with column ranges, the macro definition V6 = V1:V3 AND V4:V6 generates three output columns (both data ranges contain three columns). Column V1 is AND-ed with column V4; column V2 is AND-ed with V5 and column V3 is AND-ed with V6. However, V6 = V1:V3 AND V4:V5 returns only two output columns (the first column range contains three columns, the second contains only two, and the lesser of the two is used). In this case, column V1 is AND-ed with column V4 and column V2 is AND-ed with V5. Column V3 is not used.

- With cell ranges, the macro definition V7 = V1[1:5]:V2 AND V4[10:50]:V5 generates two output columns (both input ranges have two columns). The output columns V7 and V8 will contain five values (cells 1-5 AND-ed with cells 10-14). The macro definition V7 = V1[1:5]:V2 AND V4 generates only one output column, because the second data range contains only one column.

Important: Specifying a column without a cell range is equivalent to specifying the entire column (that is, a cell range of one through the length of the column).

Returned Values Begin in the First Cell

Any values returned by a macro function are placed in consecutive cells, beginning with the first cell (for example, TEMP[1]). For example, computing V2=SIN(V1[100:200]) would place 101 values in cells 1-100 of column V2.

Important: If you need to perform a row-by-row operation on a cell range and you want to keep the results in the corresponding rows (that is, if you operate on cells [10:20] and you want the results to be placed in cells 10-20 of the resulting column), compute specify the cell range [1:20] instead. This will compute some unnecessary values, but it will place the results in the wanted rows.

Cell Ranges Automatically Filled With Zeros

If you specify a cell range, any blank (empty) cells in the cell range are automatically filled with zeros. For example, V3 = V1[1:3]*V2 produces:

```
V1 V2 V3 1 2 2 3 4 12 [] 6 0
```

where [] represents a blank cell (that is, column V1 contains only two cell values). However, specifying V3 = V1*V2 would produce the two values 2 and 12 (calculations are performed up to the shorter of the two columns).

Use of Constants

Most of the macro function parameters can take numeric constants or expressions evaluating to a numeric constant (macro functions operating on strings can take string constants). In macro functions performing record-by-record operations, using a constant is equivalent to specifying a column containing that constant value in each row. Essentially, when a constant and a cell or column range are provided as input parameters, the constant is expanded to same dimensions as the cell or column range. Any column containing a single cell used as input in a macro function is considered a constant.

Some macro functions can take ASCII text strings as well as numerical constants. Parameters that can accept both numeric constants and ASCII text strings are noted in the "Parameters" section of each macro function.

Examples are provided in the following table.

Function Definition	How the Constant Is Interpreted
V1=3+5	Each of the constants is interpreted as a single column containing the single value. The column V1 will contain the single value 8.
V2=2*V1	The constant 2 is interpreted as a column containing the same number of rows as the column V1, with each row containing the constant 2. The column V2 will contain each value of V1 multiplied by 2.
V2 = STRING_CONCAT(V1, "ing")	The constant "ing" is interpreted as a column containing the same number of rows as the column V1, with each row containing the constant "ing". The column V2 will contain each of the text strings in V1 concatenated with "ing".
V4=V1:V3/ AVG(V1:V3)	The expression AVG(V1:V3) evaluates to a constant value, say x. The constant x is interpreted as three columns containing as many rows as the shortest column of V1, V2, or V3. Each cell contains the constant x. The output columns, V4-V6, will contain the values from columns V1-V3 divided by x.
V3=V1[10:20]^2	The constant 2 is interpreted as a cell range with 11 rows, each containing the value 2.

Computational Precision

All calculations in **IBM PredictiveInsight** spreadsheets are limited to a maximum precision of 32 bits.

Integer Calculations

Macro functions performing integer calculations (BIT_AND, BIT_NOT, BIT_OR, BIT_XOR, and T0) do not handle negative numbers. Values must be between 0 - $(2^{24} - 1)$; otherwise, an error is returned.

Invalid Cell Results and Blank Cells

Cells Containing ???

If any spreadsheet operation produces invalid results, the cell will contain ??? instead of a computed value. For example, taking the square root of a negative number using the SQRT macro function produces ??? for each negative input value.

Computations Based on ???

After a cell contains a ??? value, most calculations using that cell will propagate the ??? result. For example, summing a column containing one or more cells with ??? produces ???.

If the ??? result appears in one or more cells, click one of the cells containing ??? and read the error message displayed in the **Function Definition** text box. See the macro function reference in this guide to see why the error is occurring. Then, modify the function definition so that all input values are valid. For example, with the square root example, you might first take the absolute value of the inputs:

```
V2 = SQRT(V1) --> V2 = SQRT(ABS(V1))
```

Note: You may need to backtrace ??? values in any dependent columns to rectify the problem.

Important: Any cells containing ??? passed to the experiment manager as part of a training or test pattern are passed as zeros.

Blank Cells and ??? Cells

Blank cells are simply empty cells. They appear only at the end of a column or in empty columns. Blank cells and ??? cells are treated differently by different macro functions. Most macro functions treat blank cells as zeros and signal and propagate errors in ??? cells with the following exceptions:

- The MAXINDEX macro function ignores both blank cells and ??? cells.
- The OFFSET macro function ignores blank cells and ??? cells.
- The DDELINK macro function passes on blanks and ??? cells.
- For macro functions that skip blank cells, see "Use of Constants" on page 28. Errors in ??? cells are still signaled and propagated.
- For macro functions that skip blank cells, but use ??? cells as described, see "Macro Functions That Skip Blank and Use ??? Calls" on page 30.
- For macro functions that will return ??? for all cells if any of the cells in the listed arguments contain ???, see "Macros Functions That Cannot Process ??? Cells" on page 30.

Macro Functions That Skip Blank and Propagate ??? Cells

AVG	NORM_MINMAX
DERIVATIVE	NORM_ZSCORE
KURTOSIS	PCA
MAX	PCA_FEATURES
MEAN	SKEW
MIN	SUM
MOMENTS	VARIANCE

Macro Functions That Skip Blank and Use ??? Calls

Macro Function	How ??? Cells Are Used
COUNT	??? cells are counted.
COLUMN	??? cells are copied.
CV_FOLDS	Question marks (???) are treated as a separate class.
DELAY	??? cells are copied.
FREQ	Question marks (???) in a string column are counted as a separate class. In a numerical column, ??? is counted as a zero.
EXTRACT	Question marks (???) in the `predicate_col` are treated as zeros.
LAG	??? cells are copied.
MERGE	??? cells are copied.
SORT	All ??? cells are sorted at the end.
SAMPLE_RANDOM	??? cells can be sampled.
SELECT	??? cells are copied.
SUBSAMPLE	??? cells can be sampled.

Macros Functions That Cannot Process ??? Cells

Macro Function	Argument That Cannot Contain Any ???'s
GRID	`col1, col2`
HISTOGRAM	`data, bin_col`
INTEGRAL	`data, multiplier`
ISERROR	`data`

Chapter 4. IBM Macros Reference

This section provides reference information for the available spreadsheet macro functions in the IBM Marketing Platform suite.

The available macros are listed on the following pages, in alphabetical order. Each macro is provided with its syntax, possible parameters, and some examples.

Important: Do not use function names or keywords from the IBM Macro Language for column headings on user tables in IBM Campaign, whether mapping from a database or a flat file. These reserved words can cause errors if used in column headings on mapped tables.

ABS

Syntax

```
ABS(data)
```

Parameters

`data`

The numerical values for which to compute the absolute value. This parameter can be a constant value, a column, a cell range, or an expression that evaluates to any of these types. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

ABS calculates the absolute value of the numbers in the specified data range. The absolute value of a number is its value without its sign (that is, positive numbers are unchanged; negative numbers are returned as positive numbers). ABS returns one new column for each input column, each containing the absolute value of numbers in the corresponding input column.

Examples

`TEMP = ABS(-3) or TEMP = ABS(3)` Creates a column named TEMP containing the value 3.
`TEMP = ABS(V1)` Creates a column named TEMP, where each value is the absolute value of the contents of column V1.
`TEMP = ABS(V1:V3)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the absolute values of the contents of column V1, the values of the VX column are the absolute values of the contents of column V2, and the values of the VY column are the absolute values of the contents of column V3.
`TEMP = ABS(V1[10:20])` Creates a column named TEMP, where the first 11 cells contain the absolute values of the values in rows 10-20 of column V1. The other cells in TEMP are empty.

```
TEMP = ABS(V1[1:5]:V2)
```

Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the absolute values of the corresponding rows of column V1, and the values in column VX are the absolute values of the corresponding rows of column V2.

Related Functions

Function	Description
SIGN	Computes the sign (positive or negative) of the values in the specified data range

ACOS

Syntax

ACOS(data [, units_keyword])

Parameters

data

The numerical values to compute the arc cosine value of. This parameter can be a constant value, a column, a cell range, or an expression that evaluates to any of these types. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

units_keyword

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following values:

RADIAN - Performs the calculations in radians (default)

DEGREE - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by PI and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

ACOS calculates the arccosine of the values in the specified data range. The arccosine is the angle whose cosine is the contents of each cell. ACOS returns one new column for each input column, each containing the arccosine of numbers in the corresponding input column.

If the keyword RADIAN is used, ACOS returns values in the range 0 to π. If the keyword DEGREE is used, ACOS returns values in the range 0 - 180.

Note: The cell contents of each specified column must have values between -1.0 and 1.0 inclusive. Otherwise, a blank cell is returned for each invalid input.

Examples

`TEMP = ACOS(0) or TEMP = ACOS(0, 0) or TEMP = ACOS(0, RADIAN)` Creates a column named TEMP containing the value 1.571 ($\pi/2$ radians).
`TEMP = ACOS(0, 1) or TEMP = ACOS(0, DEGREE)` Creates a column named TEMP containing the value 90 (degrees).
`TEMP = ACOS(V1)` Creates a column named TEMP, where each value is the arccosine (in radians) of the contents of column V1.
`TEMP = ACOS(V1:V3, 1)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the arccosines of the contents of column V1, the values of the VXcolumn are the arccosines of the contents of column V2, and the values of the VY column are the arccosines of the contents of column V3. All values are in degrees.
`TEMP = ACOS(V1[10:20])` Creates a new column named TEMP, where the first 11 cells contain the arccosines of the values in rows 10-20 of column V1 (in radians). The other cells in TEMP are empty.
`TEMP = ACOS(V1[1:5]:V2)` Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the arccosines of the corresponding rows of column V1, and the values in column VX are the arccosines of the corresponding rows of column V2. All values are in radians.

Related Functions

Function	Description
ACOT	Computes the arc cotangent of the contents of the specified data range
ASIN	Computes the arc sine of the contents of the specified data range
ATAN	Computes the arc tangent of the contents of the specified data range
COS	Computes the cosine of the contents of the specified data range

ACOT

Syntax

`ACOT(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the arc cotangent of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`ACOT` returns the angle whose cotangent is the contents of each cell. `ACOT` returns one new column for each input column, each containing the arccotangent of numbers in the corresponding input column. 64-bit floating-point numbers are used.

Examples

`TEMP = ACOT(0.5)` or `TEMP = ACOT(0.5, 0)` or `TEMP = ACOT(0.5, RADIAN)`   Creates a column named `TEMP` containing the value `2.157` (radians).
`TEMP = ACOT(1, 1)` or `TEMP = ACOT(1, DEGREE)`   Creates a column named `TEMP` containing the value `0.022` (1/45) degrees.
`TEMP = ACOT(0)`   Creates a column named `TEMP` containing the value `MAX32_Float` in radians.
`TEMP = ACOT(V1)`   Creates a new column named `TEMP`, where each value is the arccotangent (in radians) of the contents of column `V1`.
`TEMP = ACOT(V1:V3, 1)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the arccotangents of the contents of column `V1`, the values of the `VX` column are the arccotangents of the contents of column `V2`, and the values of the `VY` column are the arccotangents of the contents of column `V3`. All values are in degrees.

TEMP = ACOT(V1[10:20])
Creates a new column named TEMP, where the first 11 cells contain the arccotangents of the values in rows 10-20 of column V1 (in radians). The other cells in TEMP are empty.
TEMP = ACOT(V1[1:5]:V2)
Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the arccotangents of the corresponding rows of column V1, and the values in column VX are the arccotangents of the corresponding rows of column V2. All values are in radians.

Related Functions

Function	Description
ACOS	Computes the arc cosine of the contents of the specified data range
ASIN	Computes the arc sine of the contents of the specified data range
ATAN	Computes the arc tangent of the contents of the specified data range
COT	Computes the cotangent of the contents of the specified data range

ADD_MONTHS

Syntax

ADD_MONTHS(months, date_string [, input_format])

Parameters

months

An integer that represents a number of months to add to the date_string.

date_string

A text string representing a valid date, in the format DELIM_M_D_Y, or in the format specified by the optional input_format argument.

input_format

The format that will be used for the calculated date. For a list of supported date formats, see the DATE_FORMAT function. Note that the input_format determines both the format of the input string and also the format of the output string.

Description

ADD_MONTHS returns a date after adding the specified number of months to the specified date_string. The date will be returned in the default format (DELIM_M_D_Y) or the format specified by the optional input_format argument. If you want a different format as the output, use DATE_FORMAT.

If increasing the month by the specified number of months produces an invalid date, then the result is calculated to be the last day of the month, as shown in the last example below. When necessary, leap years are taken into account. For example, adding one month to 31-Jan-2012 will result in 29-Feb-2012.

Examples

`ADD_MONTHS(12, '06-25-11')` adds one year (12 months) to the specified date and returns the date 06-25-12.

`ADD_MONTHS(3, '2011-06-25', DT_DELIM_Y_M_D)` adds three months to the specified date and returns the date 2011-09-25.

`ADD_MONTHS(1, '02-28-2011')` returns the date 03-28-2011.

`ADD_MONTHS(1, '03-31-2012')` returns the date 04-30-2012.

Related Functions

Function	Description
`DATE`	Converts a date string into a Julian date.
`DATE_FORMAT`	Transforms a date of `input_format` to `output_format`.

ALIGN

Syntax

`ALIGN(ref_series, series, range)`

Parameters

`ref_series`

`series`

`range`

Description

Align a range with position identified by its series to the reference series each row in <range> has a corresponding number, as identified in <series>. The row is aligned to the correct row offset by matching <series> to the offset in <reference_series>. If <series> does not contain the number in <reference_series>, 0 is padded. If <reference_series> does not contain the number in <series>, the row is ignored. If more than 1 row align with the reference series, the first one is used. this is used to align dates, as expressed by numbers, associated with time series starting from different dates. This way, we can generate a block where each row are data from the same day.

AND

Syntax

```
data1 AND data2 data1 && data2
```

Parameters

data1

The numbers to logical AND with the values in `data2`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The number(s) to logical AND with the values in `data1`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

AND calculates the logical AND between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in `data1` logically AND-ed to the corresponding column of `data2` (that is, the first column of `data1` is logically AND-ed to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data1` is logically AND-ed by that value. If `data2` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data2` and one column from `data2`. The first row of `data1` is logically AND-ed to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The AND operator can be abbreviated with a double-ampersand (&&). Use the double-ampersand to separate the two arguments (for example, to specify V1 AND 3, you can simply type V1&&3).

This macro is available in IBM Interact.

Examples

```
TEMP = 1 AND 8 or TEMP = 1 && 8
```

Creates a new column named TEMP containing the value one (any non-zero number is treated as a one).

```
TEMP = V1 && 1
```

Creates a new column named TEMP with the value one for each value of column V1.

`TEMP = V1 && V1` Creates a new column named `TEMP` with the value one for each non-zero value in column `V1` and the value zero for each zero in column `V1`.
`TEMP = V1 && V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` logically AND-ed with the corresponding row value of column `V2`.
`TEMP = V1:V3 && V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` logically AND-ed with the corresponding row values of column `V4`. The column `VX` contains the logically AND-ed values from columns `V2` and `V5`. The column `VY` contains the logically AND-ed values from columns `V3` and `V6`.
`TEMP = V1[10:20] && V2` or `TEMP = V1[10:20] && V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the logically AND-ed result of the values in rows 10-20 of column `V1` by the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`NOT`	Computes the logical NOT of the contents of the specified data range
`OR`	Computes the logical OR between two specified data ranges

ASIN

Syntax

`ASIN(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the arc sine of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

ASIN calculates the arcsine of the values in the specified data range. The arcsine is the angle whose sine is the contents of each cell. ASIN returns one new column for each input column, each containing the arcsine of numbers in the corresponding input column.

If the keyword RADIAN is used, ASIN returns values in the range $-\pi/2$ to $\pi/2$. If the keyword DEGREE is used, ASIN returns values in the range -90 to 90.

Note: The cell contents of each specified column must have values between -1.0 and 1.0 inclusive. Otherwise, ??? is returned for each invalid input.

Examples

TEMP = ASIN(0.5) or TEMP = ASIN(0.5, 0) or TEMP = ASIN(0.5, RADIAN) Creates a new column named TEMP containing the value 0.524 ($\pi/6$ radians).
TEMP = ASIN(0.5, 1) or TEMP = ASIN(0.5, DEGREE) Creates a new column named TEMP containing the value 30 (degrees).
TEMP = ASIN(V1) Creates a new column named TEMP, where each value is the arcsine (in radians) of the contents of column V1.
TEMP = ASIN(V1:V3, 1) Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the arcsines of the contents of column V1, the values of the VX column are the arcsines of the contents of column V2, and the values of the VY column are the arcsines of the contents of column V3. All values are in degrees.
TEMP = ASIN(V1[10:20]) Creates a new column named TEMP, where the first 11 cells contain the arcsines of the values in rows 10-20 of column V1 (in radians). The other cells in TEMP are empty.
TEMP = ASIN(V1[1:5]:V2) Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the arcsines of the corresponding rows of column V1, and the values in column VX are the arcsines of the corresponding rows of column V2. All values are in radians.

Related Functions

Function	Description
ACOS	Computes the arc cosine of the contents of the specified data range
ACOT	Computes the arc cotangent of the contents of the specified data range

Function	Description
ATAN	Computes the arctangent of the contents of the specified data range
SIN	Computes the sine of the contents of the specified data range

ATAN

Syntax

ATAN(data [, units_keyword])

Parameters

data

The numerical values to compute the arc tangent of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

units_keyword

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

RADIAN - Performs the calculations in radians (default)

DEGREE - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by PI and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

ATAN calculates the arctangent of the values in the specified data range. The arctangent is the angle whose tangent is the contents of each cell. ATAN returns one new column for each input column, each containing the arctangent of numbers in the corresponding input column.

If the keyword RADIAN is used, ATAN returns values in the range $-\pi/2$ to $\pi/2$. If the keyword DEGREE is used, ATAN returns values in the range -90 to 90.

Examples

TEMP = ATAN(1) or TEMP = ATAN(1, 0) or TEMP = ATAN(1, RADIAN)

Creates a new column named TEMP containing the value 0.785 ($\pi/4$ radians).

`TEMP = ATAN(1, 1)` or `TEMP = ATAN(1, DEGREE)` Creates a new column named `TEMP` containing the value 45 (degrees).
`TEMP = ATAN(V1)` Creates a new column named `TEMP`, where each value is the arctangent (in radians) of the contents of column `V1`.
`TEMP = ATAN(V1:V3, 1)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the arctangents of the contents of column `V1`, the values of the `VX` column are the arctangents of the contents of column `V2`, and the values of the `VY` column are the arctangents of the contents of column `V3`. All values are in degrees.
`TEMP = ATAN(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the arctangents of the values in rows 10-20 of column `V1` (in radians). The other cells in `TEMP` are empty.
`TEMP = ATAN(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the arctangents of the corresponding rows of column `V1`, and the values in column `VX` are the arctangents of the corresponding rows of column `V2`. All values are in radians.

Related Functions

Function	Description
`ACOS`	Computes the arccosine of the contents of the specified data range
`ASIN`	Computes the arcsine of the contents of the specified data range
`ATAN`	Computes the arctangent of the contents of the specified data range
`TAN`	Computes the tangent of the contents of the specified data range

AVG

Syntax

`AVG(data [, keyword])`

Parameters

`data`

The numerical values to compute the arithmetic mean of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`AVG` calculates the arithmetic mean or average of the cells in the specified data range. The arithmetic mean is calculated by summing the contents of all cells divided by the number of cells. The number of columns returned by `AVG` depends on `keyword`.

- If `keyword` is `ALL`, `AVG` returns one new column, containing a single value (the average of all cells in `data`).
- If `keyword` is `COL`, `AVG` returns a new column for each input column. Each new column contains one value (the average of all cells in the corresponding input column).
- If `keyword` is `ROW`, `AVG` returns one new column containing the average across each row of `data`.

Note: Blank cells are ignored in the calculation.

Note: `AVG` is the same as the `MEAN` macro function.

This macro is available in IBM Interact.

Examples

`TEMP = AVG(MERGE(3, 4, 5))` or `TEMP = AVG(MERGE(3, 4, 5), ALL)` Creates a new column named TEMP containing the value 4.
`TEMP = AVG(MERGE(-10, 3, 10))`Creates a new column named TEMP containing the value 1.
`TEMP = AVG(V1)` Creates a new column named TEMP containing a single value which is the arithmetic mean of the contents of column V1.
`TEMP = AVG(V1:V3)` Creates a new column named TEMP containing a single value which is the arithmetic mean of the contents of columns V1, V2, and V3.
`TEMP = AVG(V1[10:20])` Creates a new column named TEMP containing a single value which is the arithmetic mean of the cells in rows 10-20 of column V1.

TEMP = AVG(V1[1:5]:V4) Creates a new column named TEMP containing a single value which is the arithmetic mean of the cells in rows 1-5 of columns V1 through V4.
TEMP = AVG(V1:V3, COL) Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the arithmetic   mean of the contents of column V1, the single value in the VX column is the arithmetic mean of the contents of column V2, and the single value in the VY column is the arithmetic mean of the contents of column V3.
TEMP = AVG(MERGE(1,4), COL) Creates two new columns named TEMP and VX. TEMP contains a single value of one; VX contains a single value of four.
TEMP = AVG(V1[1:5]:V3, COL) Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the arithmetic mean of the cells in rows 1-5 of column V1, the value in column VX is the arithmetic mean of the cells in rows 1-5 of column V2, and the value in column VY is the arithmetic mean of the cells in rows 1-5 of column V3.
TEMP = AVG(V1, ROW) Creates a new column named TEMP, containing the same values as column V1 (the arithmetic mean of any number is itself).
TEMP = AVG(V1:V3, ROW) Creates a new column named TEMP where each cell entry is the arithmetic mean of the corresponding row across columns   V1, V2, and V3.
TEMP = AVG(V1[1:5]:V3, ROW) Creates a new column named TEMP, where the cells in rows 1-5 contain the arithmetic mean of the corresponding row across columns V1 through V3. The other cells   in TEMP are empty.

Related Functions

Function	Description
AVG_DEV	Computes the average deviation of a range of cells
SUM or TOTAL	Computes the sum of a range of cells

AVG_DEV

Syntax

AVG_DEV(data [, keyword])

Parameters

data

The values to compute the average deviation of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of `data`

ROW - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

AVG_DEV calculates the average deviation of the values in the specified data range from the mean. The arithmetic mean is calculated by summing the contents of all cells divided by the number of cells. The full equation for AVG_DEV is as follows:

$$\sum_{i}^{n} |x_i - mean|$$

The number of columns returned by AVG depends on `keyword`.
- If `keyword` is ALL, AVG returns one new column, containing a single value (the average deviation of all cells in `data`).
- If `keyword` is COL, AVG returns a new column for each input column. Each new column contains one value (the average deviation of all cells in the corresponding input column).
- If `keyword` is ROW, AVG returns one new column containing the average deviation across each row of `data`.

Note: Blank cells are ignored in the average.

Examples

TEMP = AVG_DEV(MERGE(3, 4, 5)) or TEMP = AVG_DEV(MERGE(3, 4, 5), ALL)   Creates a new column named TEMP containing the value 0.67.
TEMP = AVG_DEV(COLUMN(-4, 0))   Creates a new column named TEMP containing the value 2.
TEMP = AVG_DEV(V1)   Creates a new column named TEMP containing a single value which is the average deviation of the contents of column V1.
TEMP = AVG_DEV(V1:V3)   Creates a new column named TEMP containing a single value which is the average deviation of the contents of columns V1, V2, and V3.
TEMP = AVG_DEV(V1[10: 20])   Creates a new column named TEMP containing a single value which is the average deviation of the cells in rows 10-20 of column V1.

`TEMP = AVG_DEV(V1[1:5]:V4)`
Creates a new column named TEMP containing a single value which is the average deviation of the cells in rows 1-5 of columns V1 through V4.
`TEMP = AVG_DEV(V1:V3, COL)`
Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the average deviation of the contents of column V1, the single value in the VX column is the average deviation of the contents of column V2, and the single value in the VY column is the average deviation of the contents of column V3.
`TEMP = AVG_DEV(MERGE(1,4), COL)`
Creates two new columns named TEMP and VX, both containing the single value 0.
`TEMP = AVG_DEV(V1[1:5]:V3, COL)`
Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the average deviation of the cells in rows 1-5 of column V1, the value in column VX is the average deviation of the cells in rows 1-5 of column V2, and the value in column VY is the average deviation of the cells in rows 1-5 of column V3.
`TEMP = AVG_DEV(V1, ROW)`
Creates a new column named TEMP, containing a zero for each value of column V1 (the average deviation of any number is zero).
`TEMP = AVG_DEV(V1:V3, ROW)`
Creates a new column named TEMP where each cell entry is the average deviation of the corresponding row across columns V1, V2, and V3.
`TEMP = AVG_DEV(V1[1:5]:V3, ROW)`
Creates a new column named TEMP, where the cells in rows 1-5 contain the average deviation of the corresponding row across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
AVG or MEAN	Computes the arithmetic mean or average of a range of cells
SUM or TOTAL	Computes the sum of a range of cells

BETWEEN

Syntax

`value1 BETWEEN value2 AND value3`

Parameters

`Equivalent to value1 >= value2 AND < value3`

Description

BETWEEN is a special variant of the comparison predicate. The details of this predicate are important and the order of the operands has some unexpected implications. See the examples section.

Note: FROM and FOR use identical syntax.

Examples

`SELECT * FROM movie_titles WHERE our_cost BETWEEN 11.00 and 27.50 ;`   returns a list of movies that can be purchased for no less than $11.00, but cost no more than 27.50.
`10 BETWEEN 5 AND 15 Is true, but: 10 BETWEEN 15 AND 5 Is false:`   because the equivalent way of expressing BETWEEN (using AND) has a specific order that does not matter when you are using literals, but might matter a good deal if you provide `value2` and `value3` by using host variables, parameters, or even subqueries.

BIT_AND

Syntax

`data1 BIT_AND data2 data1 & data2`

Parameters

`data1`

The non-negative integers to bitwise AND with the values in `data2`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The non-negative integer(s) to bitwise AND with the values in `data1`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`BIT_AND` performs a bitwise AND between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in `data1` bitwise AND-ed to the corresponding column of `data2` (that is, the first column of `data1` is bitwise AND-ed to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data1` is bitwise AND-ed by that value. If `data2` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data2` and one column from `data2`. The first row of `data1` is bitwise AND-ed to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Precision for this macro function is limited to integer values less than 2^{24}. No negative values are allowed.

Note: The `BIT_AND` operator can be abbreviated with an ampersand (&). Use the ampersand to separate the two arguments (for example, to specify `BIT_AND(V1, 3)`, you can simply type V1&3.

Examples

`TEMP = 3 BIT_AND 7 or TEMP = 3 & 7` Creates a new column named `TEMP` containing the value three (bitwise AND of 011 and 111 equals 011).
`TEMP = V1 & 8` Creates a new column named `TEMP`, where each value is the contents of column `V1` bitwise AND-ed with the binary value 1000.
`TEMP = V1 & V1` Creates a new column named `TEMP` containing the same contents as the column `V1` (every value AND-ed with itself produces itself).
`TEMP = V1 & V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` bitwise AND-ed with the corresponding row value of column `V2`.
`TEMP = V1:V3 & V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` bitwise AND-ed with the corresponding row values of column `V4`. The column `VX` contains the bitwise AND-ed values from columns `V2` and `V5`. The column `VY` contains the bitwise AND-ed values from columns `V3` and `V6`.
`TEMP = V1[10:20] & V2 or TEMP = V1[10:20] & V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the bitwise AND-ed result of the values in rows 10-20 of column `V1` by the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`BIT_NOT`	Computes the bitwise NOT of the contents of the specified data range
`BIT_OR`	Computes the bitwise OR between two specified data ranges
`BIT_XOR or XOR`	Computes the bitwise XOR between two specified data ranges

BIT_NOT

Syntax

`BIT_NOT data ~ data`

Parameters

`data`

The non-negative integers to bitwise NOT. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`BIT_NOT` calculates the bitwise NOT of the values in the specified data range. It returns one new column for each input column, each containing the bitwise NOT of the values in the corresponding columns of `data`.

Note: Precision for this macro function is limited to integer values less than 2^{24}. No negative values are allowed.

Note: Using a column containing the same number x in each row as `data` is the same as using the constant x as `data`.

Note: The `BIT_NOT` operator can be abbreviated with a tilda (~). Use the tilda before the data value (for example, to specify `BIT_NOT(V1)`, you can simply type `~V1`.

Examples

`TEMP = BIT_NOT 3 or TEMP = ~3` Creates a new column named `TEMP` containing the value four (bitwise NOT of 011 equals 100).
`TEMP = ~V1` Creates a new column named `TEMP`, where each value is the bitwise NOT of the contents of column `V1`.
`TEMP = ~V1:V3` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the bitwise NOT of the contents of column `V1`, the values of the `VX` column are the bitwise NOT of the contents of column `V2`, and the values of the `VY` column are the bitwise NOT of the contents of column `V3`.
`TEMP = ~V1[100:200]` Creates a new column named `TEMP`, where the first 101 cells contain the bitwise NOT of the values in rows 1-50 of column `V1`.

Related Functions

Function	Description
`BIT_AND`	Computes the bitwise AND between two specified data ranges
`BIT_OR`	Computes the bitwise OR between two specified data ranges
`BIT_XOR or XOR`	Computes the bitwise XOR between two specified data ranges

BIT_OR

Syntax

```
data1 BIT_OR data2 data1 OR data2 data1 | data2
```

Parameters

`data1`

The non-negative integers to bitwise OR with the values in `data2`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The non-negative integer(s) to bitwise OR with the values in `data1`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`BIT_OR` performs a bitwise OR between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in `data1` bitwise OR-ed to the corresponding column of `data2` (that is, the first column of `data1` is bitwise OR-ed to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data1` is bitwise OR-ed by that value. If `data2` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data2` and one column from `data2`. The first row of `data1` is bitwise OR-ed to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Precision for this macro function is limited to integer values less than 2^{24}. No negative values are allowed.

Note: The `BIT_OR` operator can be abbreviated with a vertical bar (|). Use the vertical bar to separate the two columns (for example, to specify `BIT_OR(V1, 3)`, you can simply type `V1|3`. You also can use `OR`.

Examples

`TEMP = 3 BIT_OR 7 or TEMP = 3 OR 7 or TEMP = 3	7`
Creates a new column named `TEMP` containing the value seven (bitwise OR of 011 and 111 equals 111).
`TEMP = V1	8`
Creates a new column named `TEMP`, where each value is the contents of column `V1` bitwise OR-ed with the binary value 1000.

`TEMP = V1	V1`
Creates a new column named `TEMP` containing the same contents as the column `V1` (every value OR-ed with itself produces itself).
`TEMP = V1	V2`
Creates a new column named `TEMP`, where each value is the row value of column `V1` bitwise OR-ed with the corresponding row value of column `V2`.
`TEMP = V1:V3	V4:V6`
Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` logically OR-ed with the corresponding row values of column `V4`. The column `VX` contains the logically OR-ed values from columns `V2` and `V5`. The column `VY` contains the logically OR-ed values from columns `V3` and `V6`.
`TEMP = V1[10:20]	V2` or `TEMP = V1[10:20]	V2[1:11]`
Creates a new column named `TEMP`, where the first 11 cells contain the bitwise OR-ed result of the values in rows 10-20 of column `V1` by the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`BIT_AND`	Computes the bitwise AND between two specified data ranges
`BIT_NOT`	Computes the bitwise NOT of the contents of the specified data range
`BIT_XOR` or `XOR`	Computes the bitwise XOR between two specified data ranges

BIT_XOR

Syntax

`data1 BIT_XOR data2`

Parameters

`data1`

The non-negative integers to bitwise XOR with the values in `data2`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The non-negative integer(s) to bitwise XOR with the values in `data1`. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

BIT_XOR performs a bitwise XOR between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in data1 bitwise XOR-ed to the corresponding column of data2 (that is, the first column of data1 is bitwise XOR-ed to the first column of data, the second column with the second column, and so on).

If data2 is a constant, each value in data1 is bitwise XOR-ed by that value. If data2 contains one or more columns, the calculations are performed on a row-by-row basis between one column from data2 and one column from data2. The first row of data1 is bitwise XOR-ed to the first row value of data2, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Precision for this macro function is limited to integer values less than 2^{24}. No negative values are allowed.

Examples

`TEMP = 3 BIT_XOR 7` Creates a new column named TEMP containing the value four (bitwise XOR of 011 and 111 equals 100).
`TEMP = V1 BIT_XOR 8` Creates a new column named TEMP, where each value is the contents of column V1, bitwise XOR-ed with the binary value 1000.
`TEMP = V1 BIT_XOR V1` Creates a new column named TEMP containing all zeros (every value XOR-ed with itself produces zero).
`TEMP = V1 BIT_XOR V2` Creates a new column named TEMP, where each value is the row value of column V1 bitwise XOR-ed with the corresponding row value of column V2.
`TEMP = V1:V3 BIT_XOR V4:V6` Creates three new columns named TEMP, VX, and VY. The column TEMP contains the values in V1 bitwise XOR-ed with the corresponding row values of column V4. The column VX contains the bitwise XOR-ed values from columns V2 and V5. The column VY contains the bitwise XOR-ed values from columns V3 and V6.
`TEMP = V1[10:20] BIT_XOR V2` or `TEMP = V1[10:20] BIT_XOR V2[1:11]` Creates a new column named TEMP, where the first 11 cells contain the bitwise XOR-ed result of the values in rows 10-20 of column V1 by the values in rows 1-11 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
BIT_AND	Computes the bitwise AND between two specified data ranges
BIT_NOT	Computes the bitwise NOT of the contents of the specified data range

Function	Description
BIT_OR	Computes the bitwise OR between two specified data ranges

BUFFER

Syntax

```
BUFFER(data)
```

Parameters

```
data
```

The values to copy as constants. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

BUFFER makes a dynamic copy of the values in specified data range. These copied values change if the corresponding values in the input column(s) change. BUFFER returns one new column for each input column, each containing a copy of the values in the corresponding input column.

Note: Applying the BUFFER macro function can result in significant performance improvements in running experiments. If the input or output columns for an experiment are based on complex (slow) calculations in the spreadsheet, add the BUFFER macro function to each of the columns. This will calculate the values and store the calculated values. Otherwise, every time the experiment accesses the spreadsheet for pattern data, the values must be recomputed. If any of the input values change, the data in BUFFER will dynamically update like other macro functions in the spreadsheet.

Note: If the data values will not ever change, use the CONSTANT macro function instead. This creates a static copy of the data range.

Note: If a user function is built from a function definition using the BUFFER macro function, the portion of the function definition enclosed in the BUFFER macro function is considered a constant. Any input variables will not be required to apply the user function.

Examples

`TEMP = BUFFER(4.3)`
Creates a new column named TEMP containing the value 4.3.
`TEMP = BUFFER(V1)`
Creates a new column named TEMP, where each value is a copy of the contents of column V1.

`TEMP = BUFFER(V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are copies of the contents of column V1, the values of the `VX` column are copies of the contents of column V2, and the values of the `VY` column are copies of the contents of column V3.
`TEMP = BUFFER(V1[10:20])`
Creates a new column named `TEMP`, where the first 11 cells contain copies of the values in rows 10-20 of column V1. The other cells in `TEMP` are empty.
`TEMP = BUFFER(V1[50:99]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-50 (the other cells are empty). The values in column `TEMP` are copies of the rows of column V1, and the values in column `VX` are copies of the values in column V2.
`TEMP = BUFFER(EXTRACT(!ISERROR(V1:V3), V1:V3))`
Creates three new columns named `TEMP`, `VX`, and `VY`. The three columns contain the valid rows (that is, rows which do not contain any ??? cells) of columns V1 - V3. Wrapping the `EXTRACT` macro function inside `CONSTANT` prevents columns `TEMP`, `VX`, and `VY` from being recomputed if columns V1 - V3 change, thereby avoiding the computationally intensive `EXTRACT` macro function.

Related Functions

Function	Description
`CONSTANT`	Copies the input data range once (no dynamic update)

CEILING

Syntax

`CEILING(data)`

Parameters

`data`

The numerical values to compute the ceiling of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`CEILING` calculates the ceiling of the values in the specified data range. The ceiling of a number is the smallest integer *not* less than the number. `CEILING` returns one new column for each input column, each containing the ceiling of numbers in the corresponding input column.

Examples

`TEMP = CEILING(4.3)`
Creates a new column named `TEMP` containing the value 5.

`TEMP = CEILING(2.9)`
Creates a new column named TEMP containing the value -2.
`TEMP = CEILING(V1)`
Creates a new column named TEMP, where each value is the ceiling of the contents of column V1.
`TEMP = CEILING(V1:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the ceilings of the contents of column V1, the values of the VX column are the ceilings of the contents of column V2, and the values of the VY column are the ceilings of the contents of column V3.
`TEMP = CEILING(V1[10:20])`
Creates a new column named TEMP, where the first 11 cells contain the ceilings of the values in rows 10-20 of column V1. The other cells in TEMP are empty.
`TEMP = CEILING(V1[50:99]:V2)`
Creates two new columns named TEMP and VX, each with values in rows 1-50 (the other cells are empty). The values in column TEMP are the ceilings of the rows of column V1, and the values in column VX are the ceilings of the values in column V2.

Related Functions

Function	Description
`FLOOR or INT`	Computes the floor of each value in the specified data range
`FRACTION`	Returns the fractional part of each value in the specified data range
`TRUNCATE`	Returns the non-fractional part of each value in the specified data range

COLUMN

Syntax

`COLUMN(data [, data]...) or (data [, data]...)`

Parameters

`data`

A value to use in creating a column. This can be a constant value (numeric or ASCII text in quotes), a column, a cell range, or an expression evaluating to any of the above. This parameter can be repeated multiple times, but subsequent parameters must have the same dimensionality (that is, column width) as the first parameter. All values in all `data` parameters must be either numeric or ASCII text (that is, you cannot mix numeric and text values). If multiple `data` parameters are provided, they all must have the same number of columns. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

COLUMN vertically concatenates its inputs into columns of a function group. It returns the same number of new columns as provided in each input parameter. An unlimited number of arguments can be provided. All arguments must be either numeric or ASCII text strings (that is, you cannot mix numeric and text values).

Note: The COLUMN macro function can be abbreviated by listing the data arguments separated by commas inside parentheses (for example, (for example, TEMP = MEAN((1,2,3,4), ALL). If not used inside another macro function, the pair of parentheses is not necessary (for example, V1=1,2,3 is equivalent to V1=COLUMN(1,2,3).

Examples

TEMP = COLUMN(3, 4, 5) or TEMP = (3,4,5) or TEMP = 3,4,5 Creates a new column named TEMP with the first three cells containing the values 3, 4, and 5.
TEMP = COLUMN("one","two", "three") Creates a new column named TEMP with the first three cells containing the values "one ", "two ", and " three ".
TEMP = AVG(V1), STDV(V1) Creates a new column named TEMP with the average of column V1 in the first cell and the standard deviation of column V1 in the second cell.
TEMP = V1:V2, V3:V4 Creates two new columns named TEMP and VX where the column TEMP contains the values from column V1 followed by the values from column V3. The column VX contains the values from column V2 followed by the values from column V4.
TEMP = V1:V2, V3:V4 Creates two new columns named TEMP and VX where the column TEMP contains the values from cells 1-10 of column V1 followed by all the values from column V3. The column VX contains the values from cells 1-10 of column V2 followed by all the values from column V4.
TEMP = V1:V2, V3:V4 Creates two new columns named TEMP and VX, each containing a single value. The column TEMP contains the average of columns V1 and V2. The column VX contains the average of columns V3 and V4.

Related Functions

Function	Description
MERGE	Creates a data group by horizontally concatenating the input values
TO	Generate range operator
TRANSPOSE	Transposes a specified data range

CONSTANT

[P]

Syntax

CONSTANT(data)

Parameters

data

The values to copy as constants. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

CONSTANT makes a static copy of the values in specified data range. These copied values do *not* change if the corresponding values in the input column(s) change. The data values are copied once at the time the macro function is applied. CONSTANT returns one new column for each input column, each containing a static copy of the values in the corresponding input column.

Note: Applying the CONSTANT macro function can result in significant performance improvements in running experiments. If the input or output columns for an experiment are based on complex (slow) calculations in the spreadsheet, add the CONSTANT macro function to each of the columns. This will calculate the values once and store the calculated values. Otherwise, every time the experiment accesses the spreadsheet for pattern data, the values must be recomputed.

Note: If the data values may change, use the BUFFER macro function instead. This creates a dynamic copy of the data range, where the copied values will change if the corresponding input values change.

Note: If a user function is built from a function definition using the CONSTANT macro function, the portion of the function definition enclosed in the CONSTANT macro function is considered a constant. Any input variables will not be required to apply the user function.

Examples

TEMP = CONSTANT(4.3)
Creates a new column named TEMP containing the value 4.3.
TEMP = CONSTANT(V1)
Creates a new column named TEMP, where each value is a static copy of the contents of column V1.
TEMP = CONSTANT (V1:V3)
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are copies of the contents of column V1, the values of the VX column are copies of the contents of column V2, and the values of the VY column are copies of the contents of column V3.

`TEMP = CONSTANT(V1[10:20])`
Creates a new column named `TEMP`, where the first 11 cells contain copies of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = CONSTANT(V1[50:99]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-50 (the other cells are empty). The values in column `TEMP` are copies of the rows of column `V1`, and the values in column `VX` are copies of the values in column `V2`.
`TEMP = CONSTANT(EXTRACT(!ISERROR(V1:V3), V1:V3))`
Creates three new columns named `TEMP`, `VX`, and `VY`. The three columns contain the valid rows (that is, rows which do not contain any ??? cells) of columns `V1` - `V3`. Wrapping the `EXTRACT` macro function inside `CONSTANT` prevents columns `TEMP`, `VX`, and `VY` from being recomputed if columns `V1` - `V3` change, thereby avoiding the computationally intensive `EXTRACT` macro function.

Related Functions

Function	Description
`BUFFER`	Copies the input data range, and updates dynamically

COS

Syntax

`COS(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the cosine of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

COS calculates the cosine of the values in the specified data range. It returns one new column for each input column, each containing the cosine of numbers in the corresponding input column.

Examples

`TEMP = COS(PI) or TEMP = COS(PI, 0) or TEMP = COS(PI, RADIAN)` Returns a new column named TEMP containing a single value of -1.
`TEMP = COS(90, 1) or TEMP = COS(90, DEGREE)` Returns a new column named TEMP containing a single value of zero.
`TEMP = COS(V1) or TEMP = COS(V1, 0) or TEMP = COS(V1, RADIAN)` Creates a new column named TEMP, where each value is the cosine (in radians) of the contents of column V1.
`TEMP = COS(V1:V3, 1)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the cosines of the contents of column V1, the values of the VX column are the cosines of the contents of column V2, and the values of the VY column are the cosines of the contents of column V3. All values are in degrees.
`TEMP = COS(V1[10:20])` Creates a new column named TEMP, where the first 11 cells contain the cosines of the values in rows 10-20 of column V1 (in radians). The other cells in TEMP are empty.
`TEMP = COS(V1[1:5]:V2)` Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the cosines of the corresponding rows of column V1, and the values in column VX are the cosines of the corresponding rows of column V2. All values are in radians.

Related Functions

Function	Description
ACOS	Computes the arccosine of the contents of the specified data range
COSH	Computes the hyperbolic cosine of the contents of the specified data range
SIN	Computes the sine of the contents of the specified data range
TAN	Computes the tangent of the contents of the specified data range

COSH

Syntax

`COSH(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the hyperbolic cosine of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`COSH` calculates the hyperbolic cosine of the values in the specified data range. For x in radians, the hyperbolic cosine of a number is:

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2}$$

where e is the natural number, 2.7182818. `COSH` returns one new column for each input column, each containing the hyperbolic cosine of numbers in the corresponding input column.

Note: If the value x is too large, an overflow error is returned. This occurs if *cosh* (*x*) exceeds the maximum 32-bit floating-point value.

Examples

`TEMP = COSH(0)` or `TEMP = COSH(0, 0)` or `TEMP = COSH(0, RADIAN)`
Returns a new column named `TEMP` containing the value one.
`TEMP = COSH(V1)`
Creates a new column named `TEMP`, where each value is the hyperbolic cosine (in radians) of the contents of column `V1`.

`TEMP = COSH(V1:V3, 1)` or `TEMP = COSH(V1:V3, DEGREE)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the hyperbolic cosines of the contents of column `V1`, the values of the `VX` column are the hyperbolic cosines of the contents of column `V2`, and the values of the `VY` column are the hyperbolic cosines of the contents of column `V3`. All values are in degrees.
`TEMP = COSH(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the hyperbolic cosines of the values in rows 10-20 of column `V1` (in radians). The other cells in `TEMP` are empty.
`TEMP = COSH(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the hyperbolic cosines of the corresponding rows of column `V1`, and the values in column `VX` are the hyperbolic cosines of the corresponding rows of column `V2`. All values are in radians.

Related Functions

Function	Description
ACOS	Computes the arccosine of the contents of the specified data range
COS	Computes the cosine of the contents of the specified data range
SINH	Computes the hyperbolic sine of the contents of the specified data range
TANH	Computes the hyperbolic tangent of the contents of the specified data range

COT

Syntax

`COT(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the cotangent of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`COT` calculates the cotangent of values in the specified data range. The cotangent is the reciprocal of the tangent. `COT` returns one new column for each input column, each containing the cotangent of numbers in the corresponding input column.

Note: If a cell contains a value whose tangent is zero, then the arccotangent is infinity. In this case, `COT` returns the largest 32-bit floating-point number.

Examples

`TEMP = COT(90)` or `TEMP = COT(90, 0)` or `TEMP = COT(90, RADIAN)`   Returns a new column named `TEMP` containing the value `-0.5`.
`TEMP = COT(0)`   Returns a new column named `TEMP` containing the value `MAX_FLOAT_32`.
`TEMP = COT(V1, 1)` or `TEMP = COT(V1, DEGREE)`   Creates a new column named `TEMP`, where each value is the cotangent of the contents (in degrees) of the column `V1`.
`TEMP = COT(V1:V3, 1)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the cotangents of the contents of column `V1`, the values of the `VX` column are the cotangents of the contents of column `V2`, and the values of the `VY` column are the cotangents of the contents of column `V3`. All values are in degrees.
`TEMP = COT(V1[10:20])`   Creates a new column named `TEMP`, where the first 11 cells contain the cotangents of the values in rows 10-20 of column `V1` (in radians). The other cells in `TEMP` are empty.
`TEMP = COT(V1[1:5]:V2)`   Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the cotangents of the corresponding rows of column `V1`, and the values in column `VX` are the cotangents of the corresponding rows of column `V2`. All values are in radians.

Related Functions

Function	Description
`ACOT`	Computes the arccotangent of the contents of the specified data range
`COS`	Computes the cosine of the contents of the specified data range

Function	Description
SIN	Computes the sine of the contents of the specified data range
TAN	Computes the tangent of the contents of the specified data range

COUNT

Syntax

COUNT(data)

Parameters

data

The cell range to count the number of cells in. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

COUNT counts the number of values in the specified data range. It returns a new column containing a single value representing the number of cells containing values within the specified data range.

Note: Counting a blank column returns zero.

Examples

TEMP = COUNT(AVG(V1:V5))   Creates a new column named TEMP containing a single value of one (the function AVG returns a single cell in the default mode).
TEMP = COUNT(V1)   Creates a new column named TEMP containing a single value indicating the number of cells containing values in column V1.
TEMP = COUNT(V1:V3)   Creates a new column named TEMP containing a single value indicating the number of cells containing values in columns V1, V2, and V3.
TEMP = COUNT(V1[10:20])   Creates a new column named TEMP containing the value 11 (ranges are inclusive), given that the cells all contain values.
TEMP = COUNT(V1[1:5]:V4)   Creates a new column named TEMP containing the value 20 (5 cells in each column times 4 columns = 20 cells), given that all the cells contain values.
TEMP = COUNT(V1[1:10])   Creates a new column named TEMP containing the value 3, given that rows 1-3 of column V1 contain values and rows 4-10 are empty.

Related Functions

Function	Description
COUNT_DIFF	Returns each unique value in the input with a count of the number of times that value appeared
OFFSET	Returns the offset of each value in the input column from the top of the stream
SUM or TOTAL	Computes the sum of a range of cells

COUNT_DIFF

Syntax

COUNT_DIFF(data)

Parameters

data

The cell range to count the unique values and the frequency of their occurrence. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

COUNT_DIFF finds the number of different values in the specified data range and counts the number of times each value appears. It returns two new columns. The first column contains each unique value. The second column contains the number of times the corresponding value appeared in the data range. COUNT_DIFF ignores case when comparing text strings. For example, "Aaa" and "aAa" and "AAA" are all counted as the same class.

Note: The COUNT_DIFF macro function may take a long time to compute when data is large. A "Computing..." progress bar will be displayed until the computation is complete. If you decide to cancel the computation, click on the "X" in the progress bar and delete the function definition containing the COUNT_DIFF macro function.

Examples

TEMP = COUNT_DIFF(COLUMN(1,2,3,1))Creates two new columns named TEMP and VX. TEMP contains the values 1, 2, and 3. VX contains the counts 2, 1, and 1.
TEMP = COUNT_DIFF(COLUMN("x", "a", "a", "b")) Creates two new columns named TEMP and VX. TEMP contains the values x, a, and b. VX contains the counts 1, 2, and 1.

`TEMP = COUNT_DIFF(V1)`
Creates two new columns named TEMP and VX, where TEMP contains all of the unique values in column V1, and VX contains a count for each corresponding row of TEMP.
`TEMP = COUNT_DIFF(V1:V3)`
Creates two new columns named TEMP and VX, where TEMP contains all of the unique values in columns V1 - V3, and VX contains a count for each corresponding row of TEMP.
`TEMP = COUNT_DIFF(V1[10:20])`
Creates two new columns named TEMP and VX, where TEMP contains all of the unique values in rows 10-20 of column V1, and VX contains a count for each corresponding row of TEMP.
`TEMP = COUNT_DIFF(V1[1:5]:V4)`
Creates two new columns named TEMP and VX, where TEMP contains all of the unique values in rows 1-5 of columns V1 - V4, and VX contains a count for each corresponding row of TEMP.

Related Functions

Function	Description
COUNT	Counts the number of cells containing values in the specified data range
HISTOGRAM	Computes the histogram of a specified data range using provided bin boundaries

COV

Syntax

`COV(data1, data2)`

Parameters

`data1`

The first data set. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The second data set. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

COV counts the covariance of the two specified data ranges. [1] It outputs as many columns as the minimum dimension of the inputs (both width and height). COV is

calculated as follows:

$$COV(x, y) = \sum_{i=1}^{n} \frac{(x_i - \mu_x)(y_i - \mu_y)}{N}$$

where x and y are columns containing the same number of values N, μ_x is the mean of x, and μ_y is the mean of y.

Examples

`TEMP = COV(COLUMN(1,2), COLUMN(1,3))`   Creates a new column named `TEMP` containing the value `0.5`.
`TEMP = COV(V1, V2)`   Creates a new column named `TEMP` containing the covariance between data in column `V1` and data in column `V2`.
`TEMP = COV(V1:V2, V3:V4)`   Creates two new column named `TEMP` and `VX`. The column `TEMP` will contain the covariance between columns `V1` and `V3`. The column `VX` will contain the covariance between columns `V2` and `V4`.
`TEMP = COV(V1[1:50]:V2,V3[100:200]:V4)`   Creates two new column named `TEMP` and `VX`. The column `TEMP` will contain the covariance between rows 1-50 of column `V1` and rows 100-200 of column `V3`. The column `VX` will contain the covariance between rows 1-50 of column `V2` and rows 100-200 of column `V4`.

CURRENT_DATE

Syntax

`CURRENT_DATE([format])`

Parameters

`format`

One of the keywords in the following table specifying the date format of `date_string`.

Description

`CURRENT_DATE` returns the current date in `format`. The date is determined by the clock on the IBM server. If no `format` keyword is supplied, the default of `DELIM_M_D_Y` is used.

For all recommended databases, IBM Campaign attempts to run the CURRENT_DATE macro in the database using a database-supported current time SQL call (e.g., SYSDATE, GETDATE, DATE, or TODAY). In these cases, all parameters (including the format of the date) of this macro function are ignored

and the output includes whatever is returned by the database (e.g., a time component may be included in the output). If this occurs and you want to return just the date or the date in a different format, you can write your own custom macro using raw SQL or use other IBM macros. For example:

```
DATE_STRING(CURRENT_JULIAN( ), ...)
```

In some cases, the CURRENT_DATE() macro is run on the IBM Campaign server (e.g., if running against a flat file, against a non-recommended database with no equivalent SQL support, or if the Campaign macro expression cannot be resolved in the database). In these cases, all parameters are recognized and the output is returned in the selected format.

Note: See "Valid Format Keywords" for more information on available date formats.

This macro is available in IBM Interact.

Examples

If the date today is the 7th of September, 2000, `CURRENT_DATE()` returns "09/07/00".

Related Functions

Function	Description
DATE_FORMAT	Converts dates from one format to another.
DATE_JULIAN	Returns the Julian date of the input.
DATE_STRING	Returns the date string of the Julian date.
DATE	Converts a date string to Julian date.

CURRENT_DAY

Syntax

CURRENT_DAY()

Description

CURRENT_DAY returns the current day of the month as a number between 1-31. The date is determined by the system clock on the IBM Server.

This macro is available in IBM Interact.

Examples

If the date today is the 19th of June, `CURRENT_DAY()` will return the number 19.

Related Functions

Function	Description
CURRENT_JULIAN	Returns the Julian number for the current date.
CURRENT_MONTH	Returns the current month as a number.

Function	Description
CURRENT_TIME	Returns the current time as a string.
CURRENT_WEEKDAY	Returns the current weekday as a number.
CURRENT_YEAR	Returns the current year as a number.

CURRENT_JULIAN

Syntax

CURRENT_JULIAN()

Description

CURRENT_JULIAN() returns the Julian number for the current date (the number of days elapsed since January, 1, 0000). This is equivalent to the macro DATE(CURRENT_DATE()).

Examples

If the date today is the 31st of August, 2000, CURRENT_JULIAN() returns the number 730729.

Related Functions

Function	Description
CURRENT_DAY	Returns the current day as a number.
CURRENT_MONTH	Returns the current month as a number.
CURRENT_TIME	Returns the current time as a string.
CURRENT_WEEKDAY	Returns the current weekday as a number.
CURRENT_YEAR	Returns the current year as a number.

CURRENT_MONTH

Syntax

CURRENT_MONTH()

Description

CURRENT_MONTH returns the current month of the year as a number between 1-12.

This macro is available in IBM Interact.

Examples

If the date today is the 19th of June, CURRENT_MONTH() will return the number 6.

Related Functions

Function	Description
CURRENT_DAY	Returns the current day as a number.

Function	Description
CURRENT_JULIAN	Returns the current Julian number.
CURRENT_TIME	Returns the current time as a string.
CURRENT_WEEKDAY	Returns the current weekday as a number.
CURRENT_YEAR	Returns the current year as a number.

CURRENT_TIME

Syntax

CURRENT_TIME()

Description

CURRENT_TIME returns the current time as a string. The time is determined by the system clock on the IBM server.

Date setting on your web application

To correctly display dates on your web application within current versions of IBM Campaign, your backend server's configuration file must first be correctly configured. This is especially important for the dDateFormat and DateOutputFormatString parameters for the database containing the system tables. If these are not configured correctly, dates will also display incorrectly in Campaign. You configure these properties using IBM Marketing Platform.

To set dates for a specific language on your web application

Note: All referenced files are installed by the web application installer unless specifically noted.

Important: webapphome refers to the directory where the Campaign web application was installed. language_code refers to the language setting(s) you choose for your system.

1. Edit the webapphome/conf/campaign_config.xml file to ensure that language_code is present in the comma-separated list in the <supportedLocales> tag, as shown below:

   <supportedLocales>en_US, language_code</supportedLocales>

2. In the webapphome/webapp directory, copy the entire directory tree en_US to language_code (case sensitive).

3. In webapphome/webapp/WEB-INF/classes/resources, copy StaticMessages_en_US.properties to StaticMessages_ language_code.properties. Also copy ErrorMessages_en_US.properties to ErrorMessages_ language_code.properties.

4. Edit the StaticMessages_ language_code.properties: search for DatePattern and change it to read DatePattern=dd/MM/yyyy (case sensitive).

   Note: This format is defined by Java™. Complete detail about the format can be found in Java documentation for java.text.SimpleDateFormat at http://java.sun.com. The StaticMessages.properties file does not need to be modified.

5. For WebSphere®: Re-jar the web application.

6. For WebLogic: Remove the current web application module.
   a. Add the new module.
   b. Redeploy the web application.
   c. Restarting the Campaign listener is not necessary.
7. Ensure that the web browser's language setting has `language_codeset` to the first priority. For more details, see the sections below, To set your web browser for the correct language and To set your computer to display a specific language.

   Note: Be sure to use a hyphen, as opposed to an underscore, in `language_code`. The web application configuration is the only place where a hyphen is used instead of an underscore.
8. Log in to Campaign. Dates should be displayed in Campaign in the format specified in `StaticMessages_language_code.properties`.

For information on how to configure the time for IBM Campaign, see the *IBM Campaign* documentation.

Examples

If the time is 10:54 a.m., `CURRENT_TIME()` will return the string "10:54:00 AM".

Related Functions

Function	Description
CURRENT_DAY	Returns the current day as a number.
CURRENT_JULIAN	Returns the current Julian number.
CURRENT_WEEKDAY	Returns the current weekday as a number.
CURRENT_YEAR	Returns the current year as a number.

CURRENT_WEEKDAY

Syntax

CURRENT_WEEKDAY()

Description

CURRENT_WEEKDAY returns the current day of the week as a number between 0-6. Sunday is represented as 0, Monday as 1, and so on.

This macro is available in IBM Interact.

Examples

If today is Friday, `CURRENT_WEEKDAY()` will return the number 5.

Related Functions

Function	Description
CURRENT_DAY	Returns the current day as a number.
CURRENT_JULIAN	Returns the current Julian number.

Function	Description
CURRENT_MONTH	Returns the current month as a number.
CURRENT_TIME	Returns the current time as a string.
CURRENT_YEAR	Returns the current year as a number.

CURRENT_YEAR

Syntax

CURRENT_YEAR()

Description

CURRENT_YEAR returns the current year as a number.

This macro is available in IBM Interact.

Examples

If the current year is 2000, CURRENT_YEAR() will return the number: 2000.

Related Functions

Function	Description
CURRENT_DAY	Returns the current day as a number.
CURRENT_JULIAN	Returns the current Julian number.
CURRENT_MONTH	Returns the current month as a number.
CURRENT_TIME	Returns the current time as a string.
CURRENT_WEEKDAY	Returns the current weekday as a number.
MONTHOF	Returns the month of the year as a number.
WEEKDAYOF	Returns the weekday of the week as a number.
YEAROF	Returns the year as a number.

CV_FOLDS

Syntax

CV_FOLDS(num_folds, data [, class_data] [seed])

Parameters

num_folds

The number of folds to create for cross-validation. This value must be a positive integer greater than 1. This value must be less than 65,536 or the number of rows in data, which ever is less.

```
data
```

The input variables. This can be a column, a cell range, or an expression evaluating to either of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

```
class_data
```

If this optional data range is provided, the `CV_FOLDS` macro function will create folds while maintaining even class probabilities. The contents of `class_data` are used as the outputs for each corresponding input pattern.

If `class_data` is a single column, `CV_FOLDS` assumes that the specified column contains values for multiple output classes (that is, each distinct value is considered a separate class). If `class_data` is a data range, each output column is considered a different class. (With a data range, the values in each column would be one if a pattern belongs to that class, or zero if the pattern does not belong to that class.)

For the format definition of `class_data` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

```
seed
```

A seed value to use for the random-number generator. This must be an integer.

Description

`CV_FOLDS` evenly divides the input data into the specified number of folds. Each fold will contain the same number of input patterns. [2] It places each row of the input data range into a fold by returning a new column containing fold numbers ranging in value between one and `num_folds`.

If the optional parameter `class_data` is provided, the output class information is used to create cross-validation folds such that output class probabilities are maintained. That is, within each fold, the probability of each output class will be the same. [3]

Examples

`TEMP = CV_FOLDS(3, V1, 0)` Creates a new column named TEMP containing a value for each row of column V1. The column TEMP will contain the values 1, 2, and 3 for the three different folds. No class probabilities are maintained. The value zero is used as the seed for the random number generator.
`TEMP = CV_FOLDS(100, V1:V15)` Creates a new column named TEMP containing a value for each row of the shortest column in V1-V15. The column TEMP will contain the values 1-100 for the 100 different folds. No class probabilities are maintained. A random seed is selected.
`TEMP = CV_FOLDS(50, V1:V10, V11)` Creates a new column named TEMP containing a value for each row of the shortest column in V1-V10. The column TEMP will contain the values 1-50 for the 50 different folds. The column V11 contains the output classes. Each fold will have the same output class probabilities. A random seed is selected.

```
TEMP = CV_FOLDS(10, V1:V10, V11:V15, 96)
```

Creates a new column named TEMP containing a value for each row of the shortest column in V1-V10. The column TEMP will contain the values 1-10 for the 10 different folds. Each of the output columns V11-V15 represents an output class. Each fold will have the same output class probabilities. The value 96 is used as a seed for the random number generator.

DATALINK

Syntax

```
DATALINK([spreadsheet,] cells)
```

Parameters

spreadsheet

The name of the spreadsheet in the current IBM PredictiveInsight work session to link with (for example, Sheet1). If no value is specified, the current spreadsheet is used.

cells

The specific cells within spreadsheet to link with. The cells parameter specifies the column(s) and row(s) to link with and can be specified as any of the following:

```
Cn Cn:Cm RnCm | CmRn RnCm:RNCM | CmRn:CMRN
```

The characters C and R specify column and row, respectively. The variables n, m, N, and M are the row and column numbers.

Description

DATALINK creates an internal link with data in the current IBM PredictiveInsight work session. It returns the number of columns specified in the cells parameter. Changes in the source data columns will automatically be reflected.

Examples

```
TEMP = DATALINK(C1)
```

Creates a new column named TEMP containing the values in column one of the current spreadsheet.

```
TEMP = DATALINK(Sheet2, C1:C3)
```

Creates three new columns named TEMP, VX, and VY, containing the values in columns 1-3 of the **Sheet2** spreadsheet.

```
TEMP = DATALINK(Sheet4, C5R10) or TEMP = DATALINK(Sheet4, R10C5)
```

Creates a new column named TEMP containing the cell value in the 5th column, 10th row of the **Sheet4** spreadsheet.

```
TEMP = DATALINK(Sheet1, C1R1:C3R500) or TEMP = DATALINK(Sheet1, R1C1:R500C3)
```

Creates three new columns named TEMP, VX, and VY, containing the values in rows 1-500 of columns 1-3 of the **Sheet2** spreadsheet.

Related Functions

Function	Description
DDELINK	Creates an external link to data from another Windows application

DATE

Syntax

DATE(date_string [, format])

Parameters

date_string

A text representing a valid date.

format

One of the keywords in the table under "Valid Format Keywords," specifying the date format of date_string.

Description

DATE converts a date string into a Julian date (the number of days elapsed since January 1, 0000). Virtually any date format is supported via the optional format keyword, which specifies how the date is represented. If no format keyword is supplied, the default of DELIM_M_D_Y is used.

Date formats are either fixed-width (for example, the date February 28, 1970 is represented as 02281970 in MMDDYYYY format), or delimited (for example, February 28, 1970, 2-28-1970, or 02/28/1970). All of the previous examples are variants of DELIM_M_D_YY format.

In delimited formats, delimiters are slash (/), dash(-), space ( ), comma (,), or colon (:); years can be represented by either 2 or 4 digits; and months can be fully spelled out (for example, February), abbreviated (for example, Feb), or numeric (for example, 2 or 02).

For all years specified as two-digits:
* Two-digit years less than the Year 2000 threshold (default is 20, but can be set by the user) are considered to be in the 2000's.
* Two-digit years greater than or equal to the threshold are considered to be in the 1900's.

This macro is available in IBM Interact.

Examples

`DATE("8/31/2000")` returns the number 730729.

Valid Format Keywords

Keyword	Description	Example(s)
MM	2-digit month	01, 02, 03, ..., 12
MMDD	2-digit month and 2-digit day	March 31 is 0331
MMDDYY	2-digit month, 2-digit day, and 2-digit year	March 31, 1970 is 033170
MMDDYYYY	2-digit month, 2-digit day, and 4-digit year	March 31, 1970 is 03311970
DELIM_M_D	Any delimited month followed by day	March 31, 3/31, or 03-31
DELIM_M_D_Y	Any delimited month, day, and year	March 31, 1970 or 3/31/70
DELIM_Y_M	Any delimited year followed by month	March, 70; 3-70; or 3/1970
DELIM_Y_M_D	Any delimited year, month, and day	1970 Mar 31 or 70/3/31
YYMMM	2-digit year and 3-letter month	70MAR
YYMMMDD	2-digit year, 3-letter month, and 2-digit day	70MAR31
YY	2-digit year	70
YYMM	2-digit year and 2-digit month	7003
YYMMDD	2-digit year, 2-digit month, and 2-digit day	700331
YYYYMMM	4-digit year and 3-letter month	1970MAR
YYYYMMMDD	4-digit year, 3-letter month, and 2-digit day	1970MAR31
YYYY	4-digit year	1970
YYYYMM	4-digit year and 2-digit month	197003
YYYYMMDD	4-digit year, 2-digit month, and 2-digit day	19700331
DELIM_M_Y	Any delimited month followed by year	3-70, 3/70, Mar 70, March 1970
DELIM_D_M	Any delimited day followed by month	31-3, 31/3, 31 March
DELIM_D_M_Y	Any delimited day, month, and year	31-MAR-70, 31/3/1970, 31 03 70
DD	2-digit day	31
DDMMM	2-digit day and 3-letter month	31MAR

DDMMMYY	2-digit day, 3-letter month, and 2-digit year	31MAR70
DDMMMYYYY	2-digit day, 3-letter month, and 4-digit year	31MAR1970
DDMM	2-digit day and 2-digit month	3103
DDMMYY	2-digit day, 2-digit month, and 2-digit year	310370
DDMMYYYY	2-digit day, 2-digit month, and 4-digit year	31031970
MMYY	2-digit month and 2-digit year	0370
MMYYYY	2-digit month and 4-digit year	031970
MMM	3-letter month	MAR
MMMDD	3-letter month and 2-digit day	MAR31
MMMDDYY	3-letter month, 2-digit day, and 2-digit year	MAR3170
MMMDDYYYY	3-letter month, 2-digit day, and 4-digit year	MAR311970
MMMYY	3-letter month and 2-digit year	MAR70
MMMYYYY	3-letter month and 4-digit year	MAR1970
MONTH	Month of the year	January, February, March, and so on or Jan, Feb, Mar, and so on
WEEKDAY	Day of the week	Sunday, Monday, Tuesday, and so on (Sunday = 0)
WKD	Abbreviated day of the week	Sun, Mon, Tues, and so on (Sun = 0)

Related Functions

Function	Description
DATE_FORMAT	Converts dates from one format to another.
DATE_JULIAN	Returns the Julian date of the input.
DATE_STRING	Returns the date string of the Julian date.
CURRENT_DATE	Returns the current date in a specified format.

DATE_FORMAT

Syntax

```
DATE_FORMAT(date_string, input_format, output_format)
```

Parameters

`date_string`

A text representing a valid date.

`input_format`

One of the keywords in the table below specifying the date format of `date_string`.

`output_format`

One of the keywords in the table below specifying the wanted output date format.

Description

`DATE_FORMAT()` transforms a date of `input_format` to another format `output_format`.

If the date is fixed-width, it must be set to one of the following values:
- DDMMYY[YY]
- DDMMMYY[YY]
- MMDDYY[YY]
- MMMDDYY[YY]
- YY[YY]MMDD
- YY[YY]MMMDD

MM is a 2-digit month and MMM is the 3-character month abbreviation.

If the date is delimited (any delimiter can be used including SPACE, DASH, SLASH), it must be set to one of these values:
- DELIM_D_M_Y
- DELIM_M_D_Y
- DELIM_Y_M_D

This macro is available in IBM Interact.

Examples

`DATE_FORMAT("012171", MMDDYY, MMDDYYYY)` returns the string "01211971".

Note: See "DATE" on page 73 for additional information on valid date formats.

Related Functions

Function	Description
DATE	Converts a date string to a Julian date.
DATE_JULIAN	Returns the Julian date of the input.
DATE_STRING	Returns the date string of the Julian date.

DATE_JULIAN

Syntax

```
DATE_JULIAN(year, month, day)
```

Parameters

`year`

Valid 2-digit or 4-digit year number.

`month`

Valid month number between 1-12.

`day`

Valid day number between 1-31.

Description

DATE_JULIAN returns the Julian date of the specified input. The Julian date is the number of days elapsed since January 1, 0000.

Examples

DATE_JULIAN (2000,08,31) returns the number 730729.

Related Functions

Function	Description
DATE	Converts a date string to a Julian date.
DATE_FORMAT	Converts dates from one format to another.
DATE_STRING	Returns the date string of the Julian date.

DATE_STRING

Syntax

```
DATE_STRING(julian_date [, 'output_format'[, max_length]])
DATE_STRING(julian_date [, 'format_string'[, max_length]]
```

Parameters

`julian_date`

A number representing a Julian date, the number of days elapsed since January 1, 0000.

`output_format`

String, valid date format.

```
max_length
```

```
format_string
```

A format string optionally including any combination of the following format codes:

Code	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month (01 - 31)
%H	Hour in 24-hour format (00 - 23)
%I	Hour in 12-hour format (01 - 12)
%j	Day of year (001 - 366)
%m	Month (01 - 12)
%M	Minute (00 - 59)
%p	Current locale's AM/PM indicator for 12-hour clock
%S	Second (00 - 59)
%U	Week of year, with Sunday as first day of week (00 - 51)
%w	Weekday (0 - 6; Sunday is 0)
%W	Week of year, with Monday as first day of week (00 - 51)
%x	Date representation for current locale
%X	Time representation for current locale
%y	2-digit year (00 - 99)
%Y	4-digit year
%z, %Z	Time zone name or abbreviation; no output if time zone is unknown
%%	Percent sign

Description

DATE_STRING returns the date string of the Julian date. If output_format is not provided, the default keyword DELIM_M_D_Y will be used.

Examples

DATE_STRING(730729) returns the string "08/31/00".

Note: See "DATE" on page 73 for additional information on valid date formats.

Related Functions

Function	Description
DATE	Converts a date string to a Julian date.
DATE_JULIAN	Returns the Julian date of the input.
DATE_FORMAT	Converts dates from one format to another.

DAY_BETWEEN

Syntax

DAY_BETWEEN(from_date_string, to_date_string [, input_format])

Parameters

from_date_string

A text representing a valid date from which to count the number of days elapsed.

to_date_string

A text representing a valid date to which the number of days is counted. This date must be in the same format as from_date_string.

input_format

One of the keywords in the table below, specifying the date format of from_date_string and to_date_string.

Description

DAY_BETWEEN returns the number of days between from_date_string and to_date_string. If input_format is not provided, the default keyword DELIM_M_D_Y will be used.

Examples

DAY_BETWEEN("08/25/00","08/31/00") returns the number 6.

Note: See "DATE" on page 73 for additional information on valid date formats.

Related Functions

Function	Description
DAY_FROMNOW	Returns the number of days between the current day and a specified date.
DAY_INTERVAL	Returns the number of days between two specified dates.

DAY_FROMNOW

Syntax

DAY_FROMNOW(to_year, to_month, to_day)

Parameters

to_year

Valid 2-digit or 4-digit year number.

to_month

Valid month number between 1-12.

to_day

Valid day number between 1-31.

Description

DAY_FROMNOW returns the number of days between the current day and the date specified by to_year/to_month/to_day.

Note: If the specified date is in the past, the returned value will be negative.

Examples

If today's date is the 31st of August, 2000, DAY_FROMNOW(2000,12,31) returns the number 122.

Related Functions

Function	Description
DAY_BETWEEN	Returns the number of days between two specified date strings.
DAY_INTERVAL	Returns the number of days between two specified dates.

DAY_INTERVAL

Syntax

DAY_INTERVAL(from_year, from_month, from_day, to_year, to_month, to_day)

Parameters

from_year

Valid 2-digit or 4-digit year number.

from_month

Valid month number between 1-12.

from_day

Valid day number between 1-31.

to_year

Valid 2-digit or 4-digit year number.

to_month

Valid month number between 1-12.

to_day

Valid day number between 1-31.

Description

DAY_INTERVAL returns the number of days between the specified from date (from_year/from_month/from_day) and the specified to date (to_year/to_month/ to_day).

Examples

DAY_INTERVAL(2000,8,31,2000,12,31) returns the number 122.

Related Functions

Function	Description
DAY_BETWEEN	Returns the number of days between two specified date strings.
DAY_FROMNOW	Returns the number of days between the current day and a specified date.

DAYOF

Syntax

DAYOF(date_string [, input_format])

Parameters

date_string

A text representing a valid date.

input_format

One of the keywords in the table below, specifying the date format of date_string.

Description

DAYOF returns the day of the month as a number for the date represented by the date_string. If input_format is not provided, the default keyword DELIM_M_D_Y will be used.

Examples

DAYOF("08/31/00") returns the number 31.

Note: See "DATE" on page 73 for additional information on valid date formats.

DDELINK

Syntax

DDELINK(service, topic, items)

Parameters

service

The service name (for example, excel) to create a DDE link with.

topic

The topic within service with which to link. For most applications, the topic is a filename. Type the full path and filename of the wanted topic (for example, c:\stock\prices\05jan.xls).

items

The items in topic to link with. The syntax of the items depends on the selected service. For example, in Excel, R1C1:R10C20 selects rows 1-10 and columns 1-20.

Description

DDELINK creates a Dynamic Data Exchange (DDE) link with data in an external Windows application. It returns the number of columns specified in the items parameter. Changes in the source application will automatically be reflected in the IBM PredictiveInsight spreadsheet.

Examples

TEMP = DDELINK(Excel, c:\excel\data.xls, C1:C2)   Creates two new columns named TEMP and VX, containing the values in columns 1-2 of the c:\excel\data.xls spreadsheet.
TEMP = DDELINK(Excel, c:\excel\data.xls, R1:R10)   Creates as many new columns as needed to accommodate rows 1-10 of the c:\excel\data.xls spreadsheet.
TEMP = DDELINK(Excel, c:\excel\data.xls, R1C1:R100C3)   Creates three new columns named TEMP, VX and VY containing rows 1-100 of the c:\excel\data.xls spreadsheet.
TEMP = DDELINK(123W, c:\lotsuite\sample.wk4, A:A1..A:C8)   Creates three new columns named TEMP, VX and VY containing rows 1-8 of columns A-C of spreadsheet A in the Lotus® file sample.wk4.

Related Functions

Function	Description
DATALINK	Creates an internal link to data in a IBM PredictiveInsight spreadsheet

---

DECIMATE

Syntax

DECIMATE(column, max_value)

Parameters

column

The column of values to decimate. All values within this column must be positive integers less than max_value.

max_value

The number of columns to return. This must be a positive integer greater than or equal to the maximum value in column.

Description

DECIMATE converts a positive integer value into a binary pattern of max_value columns in length. If the value is n, the$n$-th column contains a one; all other columns contain zeros. This macro function returns max_value columns.

Note: DECIMATE is the opposite of the MAXINDEX macro function.

Examples

TEMP = DECIMATE(COLUMN(1,2,3), 3) Creates new columns named TEMP, VX, and VY with a row for each corresponding row of input. The first row contains 1 0 0, the second contains 0 1 0, and the third contains 0 0 1.
TEMP = DECIMATE(COLUMN(1,1,2), 3) Creates new columns named TEMP, VX, and VY with a row for each corresponding row of input. The first row contains 1 0 0, the second contains 1 0 0, and the third contains 0 1 0.
TEMP = DECIMATE(V1, 10) Creates ten new columns with a row for each corresponding row of input. Each row contains a single one in the column representing the corresponding input value. All other columns contain zeros.

Related Functions

Function	Description
MAX	Computes the maximum of a range of cells

Function	Description
MAXINDEX	Returns the input columns values lagging by a specified number of time steps
MIN	Computes the minimum of a range of cells

DELAY

Syntax

DELAY(delay, data)

Parameters

delay

The number of time steps to delay. This value must be a positive integer.

data

The values to delay. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

DELAY returns values in the input data range, delayed by the specified number of time steps. It views each input column as a data series in time. and returns one new column for each input column. Each new column contains the time step-delayed values (delayed by delay number of time steps) of the numbers in the corresponding input column.

Note: The DELAY macro function returns a column with values such that the cell VY[x] = data[x + delay].

Note: This function is useful in creating patterns from time-series data. To create multiple delays, use the SLIDE_WINDOW macro function.

Examples

TEMP = DELAY(1, COLUMN(1,2,3,4))
Creates a new column named TEMP containing the values 2, 3, and 4 in cells 1-3, respectively.
TEMP = DELAY(2, V1)
Creates a new column named TEMP, where each value is the contents of column V1 delayed by two time steps.

`TEMP = DELAY(10, V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the contents of column V1 delayed by ten time steps, the values of the VX column are the contents of column V2 delayed by ten time steps, and the values of the VY column are the contents of column V3 delayed by ten time steps.
`TEMP = DELAY(1, V1[10:20])`
Creates a new column named `TEMP`, where the first 10 cells contain the values in column V1 delayed by one time step (that is, rows 11-20 of column V1). The other cells in `TEMP` are empty.
`TEMP = DELAY(2, V1[1:5]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-3 (the other cells are empty). The values in column `TEMP` are the corresponding rows of column V1 delayed by two time steps, and the values in column `VX` are the corresponding rows of column V2 delayed by two time steps (that is, rows 3-5 of columns V1 and V2).

Related Functions

Function	Description
`LAG`	Returns the input columns values lagging by a specified number of time steps
`SLIDE_WINDOW`	Creates a pattern from a specified window and slides it down to create the next pattern

DERIVATIVE

Syntax

`DERIVATIVE(data [, divisor])`

Parameters

`data`

The numerical values to compute the derivative of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`divisor`

A value to divide each value in `data` by. This can be a constant value or an expression evaluating to a constant.

Description

`DERIVATIVE` calculates the derivative of the values in a time-series. Each value is the difference between the current value and the value from the next time-step. If a value is provided for `divisor`, each value is divided by the specified value.

`DERIVATIVE` returns one new column for each input column, each containing derivative of the values in the corresponding input column.

Note: The length of the returned column will be one less than the length of the source data column (`data`).

Examples

`TEMP = DERIVATIVE(5)`   Creates a new column named `TEMP` containing all blank cells (at least two cell values are required to generate a result).
`TEMP = DERIVATIVE(COLUMN(1, 2, 5))`   Creates a new column named `TEMP`, containing the values 1 and 3.
`TEMP = DERIVATIVE(V1)`   Creates a new column named `TEMP`, where each value is the derivative of the contents of column `V1`.
`TEMP = DERIVATIVE(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the derivatives of the contents of column `V1`, the values of the `VX` column are the derivatives of the contents of column `V2`, and the values of the `VY` column are the derivatives of the contents of column `V3`.
`TEMP = DERIVATIVE(V1[10:20])`   Creates a new column named `TEMP`, where the cells in rows 10-20 contain the derivatives of the corresponding rows of column `V1`. The other cells in `TEMP` are empty.
`TEMP = DERIVATIVE(V1[1:5]:V2)`   Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the derivatives of the corresponding rows of column `V1`, and the values in column `VX` are the derivatives of the corresponding rows of column `V2`.

Related Functions

Function	Description
`INTEGRAL`	Computes the integral of the values in the specified data range

DISTANCE

Syntax

`DISTANCE(lat1,long1,lat2,long2[,UNIT_OF_MEASURE][,PRECISION])`

Parameters

`lat1`

The latitude of the first point, as a decimal value.

`long1`

The longitude of the first point, as a decimal value.

`lat2`

The latitude of the second point, as a decimal value.

`long2`

The longitude of the second point, as a decimal value.

`UNIT_OF_MEASURE`

An optional parameter indicating the unit of measure for the returned distance. Values are MILES or KILOMETERS. If you omit this parameter, MILES is the default.

`PRECISION`

An optional parameter indicating the level of precision following the decimal point for the returned distance. If you specify a precision value, the returned distance is truncated to the number of decimal places you specify. The maximum value is 5. If you omit this value, the number of decimal places is not truncated.

Description

`DISTANCE` calculates the distance between two points. Latitude and longitude are expected to be in decimal units.

Examples

`DISTANCE (18.529747,73.839798,18.533511,73.8777995,MILES,2)` returns the value 2.50 Miles.

DISTINCT

Syntax

`DISTINCT(data)`

Parameters

`data`

The data range.

Description

`DISTINCT` finds the unique values in the specified data range. It returns this list of values in a single column. `DISTINCT` ignores case when comparing text strings. For example, \042Aaa\042 and \042aAa\042 and \042AAA\042 are all counted as the same value.

The `DISTINCT` macro function may take a long time to compute when data is large. A \042Computing...\042 progress bar will be displayed until the computation is

complete. If you decide to cancel the computation, click on the \042X\042 in the progress bar and delete the function definition containing the `DISTINCT` macro function.

DIV

Syntax

```
data DIV divisor data / divisor
```

Parameters

`data`

The numerical values to divide into. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`divisor`

The value(s) to divide the values in the specified data range by. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `divisor` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`DIV` divides the specified data range by the divisor value. It returns a new column for each input column, each containing the corresponding column in `data1` divided by the corresponding column of `data2` (that is, the first column of `data1` is divided by to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data1` is divided by that value. If `data2` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data2` and one column from `data2`. The first row of `data1` is divided by the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Using a column containing the same number x in each row as `divisor` is the same as using the constant x as `divisor`.

Note: The `DIV` operator can be abbreviated with a slash (/).

This macro is available in IBM Interact.

Examples

`TEMP = 8 DIV 4 or TEMP = 8/4`
Creates a new column named `TEMP` containing the value two.

`TEMP = V1/8`   Creates a new column named `TEMP`, where each value is the contents of column `V1` divided by eight.
`TEMP =V1:V3/2`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the contents of column `V1` divided by two, the values of the `VX` column are the contents of column `V2` divided by two, and the values of the `VY` column are the contents of column `V3` divided by two.
`TEMP = V1/V1`   Creates a new column named `TEMP` containing all ones (since any number divided by itself is one).
`TEMP = V1/V2`   Creates a new column named `TEMP`, where each value is the row value of column `V1` divided by the corresponding row value of column `V2`.
`TEMP = V1:V3/V4:V6`   Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` divided by the corresponding row values of column `V4`. The column `VX` contains the division of column `V2` by `V5`. The column `VY` contains the division of column `V3` by `V6`.
`TEMP = V1[10:20] / V2` or `TEMP = V1[10:20] / V2[1:11]`   Creates a new column named `TEMP`, where the first 11 cells contain the result of dividing the values in rows 10-20 of column `V1` by the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`MOD`	Computes the modulo of the contents of the specified data range
`MULT`	Multiplies the contents of two data ranges
`POW`	Computes a base value raised to the specified exponential powers

EQ

Syntax

`data1 EQ data2 data1 == data2 (data1 = data2)`

Parameters

`data1`

The cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The number(s) to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

EQ compares the two specified data ranges, returning a one if the values are equal or a zero if they are not equal. It returns a new column for each input column, each containing the corresponding column in `data1` compared to the corresponding column of `data2` (that is, the first column of `data1` is compared to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data` is compared to that value. If `data2` is a column, the calculations are performed on a row-by-row basis. The values in `data1` are compared to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

When comparing strings, case does not matter (that is, "Yes", "YES", "yes", and "yeS" are all considered equal).

Note: The EQ operator can be abbreviated with a double equal sign (==). Inside parentheses, a single equal sign (=) also can be used for the EQ macro function (outside parentheses, the equal sign is interpreted as the assignment operator).

This macro is available in IBM Interact.

Examples

`TEMP = 3 EQ 4 or TEMP = 3==4 or TEMP = (3=4)`   Creates a new column named TEMP containing the value zero (since three is not equal to four).
`TEMP = "No" == "NO"`   Creates a new column named TEMP containing the value one (string compares are case insensitive).
`TEMP = V1 == 8`   Creates a new column named TEMP, where each value is one if the corresponding row value of the column V1 is equal to the number eight, otherwise zero.
`TEMP = V1==V1`   Creates a new column named TEMP containing all ones (since every number is equal to itself).
`TEMP = V1==V2`   Creates a new column named TEMP, where each value is the row value of column V1 compared to the corresponding row value of column V2.
`TEMP = V1:V3 == V4:V6`   Creates three new columns named TEMP, VX, and VY. The column TEMP contains the values in V1 compared to the corresponding row values of column V4. The column VX compares columns V2 and V5. The column VY compares columns V3 and V6.

```
TEMP = V1[10:20] == V2  or  TEMP = V1[10:20] == V2[1:11]
```

Creates a new column named TEMP, where the first 11 cells contain the results of comparing the values in rows 10-20 of column V1 to rows 1-11 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
EQ	Returns TRUE if one data range is equal to another
GE	Returns TRUE if one data range is greater than or equal to another
GT	Returns TRUE if one data range is greater than another
LE	Returns TRUE if one data range is less than or equal to another
LT	Returns TRUE if one data range is less than another
NE	Returns TRUE if one data range is not equal to another

EXP

Syntax

```
EXP(data)
```

Parameters

data

The numerical values used as an exponent to the natural number, e. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

EXP raises the natural number, e, by each of the values in the specified data range (that is, calculates e^x). The constant e equals 2.7182818. EXP returns one new column for each input column, each containing the result e^x for each value x in the corresponding input column(s). EXP is the inverse of the LN macro function.

Note: If the value x is too large or too small, an overflow error is returned. This occurs if e^x exceeds the maximum or minimum 32-bit floating-point value.

Examples

```
TEMP = EXP(2)Creates a new column named TEMP containing the value 7.39.
```

`TEMP = EXP(V1)` Creates a new column named TEMP, where each value is result of raising *e* to the contents of column V1.
`TEMP = EXP(V1:V3)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the results of raising *e* to the column V1, the values of the VX column are the results of raising *e* to the contents of column V2, and the values of the VY column are the results of raising *e* to the contents of column V3.
`TEMP = EXP(V1[10:20])` Creates a new column named TEMP, where the first 11 cells contain the results of raising *e* to the values in rows 10-20 of column V1. The other cells in TEMP are empty.
`TEMP = EXP(V1[1:5]:V2)` Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the results of raising *e* to the corresponding row values of column V1, and the values in column VX are the results of raising *e* to the corresponding row values of column V2.

Related Functions

Function	Description
`LN` or `LOG`	Computes the natural log of the contents of the specified data range
`LOG2`	Computes the log base2 of the contents of the specified data range
`LOG10`	Computes the log base10 of the contents of the specified data range
`POW`	Exponential power

EXTERNALCALLOUT

Syntax

`EXTERNALCALLOUT( calloutName, arg1, ...)`

Parameters

`calloutName`

The name of the callout you created using the ExternalCallout API. This name must match the name of the External Callout category you created in IBM Marketing Platform.

`arg1`

An argument required by your callout, if required.

Description

EXTERNALCALLOUT enables you to call an external application to add data to your interactive flowchart. EXTERNALCALLOUT can return whatever you have created the callout to do. You must write this callout in Java using the ExternalCallout API. For more details, see the *IBM Interact Developer's Guide*.

Examples

EXTERNALCALLOUT(getStockPrice, UNCA)

Calls the callout getStockPrice passing the name of the stock, UNCA, as the argument. This user defined callout returns the stock price as defined by the callout.

EXTRACT

Syntax

EXTRACT(predicate_col, data)

Parameters

predicate_col

A column of boolean values or an expression evaluating to a single column of boolean values. Boolean values are interpreted as zero or non-zero. This column should contain at least as many rows as the data range from which data is being extracted. Otherwise, predicate_col will be a limit to the number of rows processed by the EXTRACT macro function (see "Description" below).

data

The values to extract. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

EXTRACT returns the rows in the specified data range that have a value of one in the corresponding row of the predicate column. This macro function reduces data by "throwing out" all rows where the corresponding value in predicate_col is zero. EXTRACT returns one new column for each input column, each containing the values in the corresponding input column for which the corresponding value in predicate_col is one. The extracted rows of data will occupy the first n cells of the output columns where n is the number of ones in predicate_col.

Since EXTRACT operates on a row-by-row basis, it produces a result for each row up to the last value of the shortest column (that is, the shortest column out of predicate_col and the columns in the data range data). All non-zero values in predicate_col are evaluated as one.

Note: Generally, you will want to create a predicate column using one of the comparison macro functions (for example, ==, >, <, ISEVEN, ISODD, etc.). You can then extract the rows of interest from a specified data range using the EXTRACT macro function. This can be useful to "weed out" bad data points (for example, when a particular value exceeds the maximum or minimum value for a data variable). It also can be used to consolidate all examples of a particular class (for example, if the column V3 contains ones and zeros for one of the output classes, use V4=EXTRACT(V3, V1:V2) to extract the inputs V1 and V2). Since EXTRACT condenses all the extracted rows as a block of data (that is, it fills the cell range VX[1:n}:VY), where n is the number of extracted rows, it is a useful function for copying a range of cells from their current row locations to rows 1- *n* of the spreadsheet.

Examples

TEMP = EXTRACT(1, V1)Creates a new column named TEMP containing a copy of column V1.
TEMP = EXTRACT(1, V1[50:100]:V2) Creates two new columns named TEMP and VX with values in the first 51 cells. The values in the TEMP column are the cells 50-100 of column V1, and the values in the VX column are the cells 50-100 of column V2.
TEMP = EXTRACT(V3, V1:V2) Creates two new columns named TEMP and VX. For each row where the value in column V3 is one, the corresponding row across columns V1 and V2 are extracted into columns TEMP and VX, respectively.
TEMP = EXTRACT(V1>V2, V1) Creates a new column named TEMP containing all the values in column V1 that were greater than the corresponding values in column V2.
TEMP = EXTRACT(V3[10:20], V1[10:20]:V2) Creates two new columns named TEMP and VX. For rows 10-20 where the value in column V3 is one, the corresponding row across columns V1 and V2 is extracted into columns TEMP and VX, respectively.

Related Functions

Function	Description
IF	Begins a conditional statement if-then-else statement
SELECT	Returns the specified column(s) from a data range
SUBSAMPLE	Reduces data by returning every n-th row value
SUBSTITUTE	*Substitutes values in a column with a value specified in a conversion table*

FACTORIAL

Syntax

FACTORIAL(data)

Parameters

`data`

The integer values to compute the factorial for. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above, but must be greater than or equal to zero. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`FACTORIAL` calculates the factorial of the values in the specified data range. All inputs must be integers greater than or equal to zero. The factorial of an integer less than or equal to one is one. For integers $X \geq 2$, the factorial $x! = x\,(x\text{-}1)(x\text{-}2)...(x - (x\text{-}1))$. `FACTORIAL` returns one new column for each input column, each containing the factorial of numbers in the corresponding input column.

Note: Any value greater than 34 will produce ??? (floating-point overflow error).

Examples

`TEMP = FACTORIAL(3)` Creates a new column named `TEMP` containing the value 6.
`TEMP = FACTORIAL(-2)` Generates an error 333, indicating that the argument must be greater than or equal to 0.
`TEMP = FACTORIAL(V1)` Creates a new column named `TEMP`, where each value is the factorial of the contents of column `V1`.
`TEMP = FACTORIAL(V1:V3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the factorials of the contents of column `V1`, the values of the `VX` column are the factorials of the contents of column `V2`, and the values of the `VY` column are the factorials of the contents of column `V3`.
`TEMP = FACTORIAL(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the factorials of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = FACTORIAL(V1[50:99]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-50 (the other cells are empty). The values in column `TEMP` are the factorials of the rows of column `V1`, and the values in column `VX` are the factorials of the values in column `V2`.

FLOOR

Syntax

`FLOOR(data)`

Parameters

`data`

The numerical values to compute the floor of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

FLOOR calculates the floor of the values in the specified data range. The floor of a number is the greatest integer less than the number. FLOOR returns one new column for each input column, each containing the floor of numbers in the corresponding input column.

Note: This is the same as the INT macro function.

Examples

`TEMP = FLOOR(4.3)`
Creates a new column named `TEMP` containing the value 4.
`TEMP = FLOOR(2.9)`
Creates a new column named `TEMP` containing the value -3.
`TEMP = FLOOR(V1)`
Creates a new column named `TEMP`, where each value is the floor of the contents of column `V1`.
`TEMP = FLOOR(V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the floors of the contents of column `V1`, the values of the `VX` column are the floors of the contents of column `V2`, and the values of the `VY` column are the floors of the contents of column `V3`.
`TEMP = FLOOR(V1[10:20])`
Creates a new column named `TEMP`, where the first 11 cells contain the floors of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = FLOOR(V1[50:99]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-50 (the other cells are empty). The values in column `TEMP` are the floors of the rows of column `V1`, and the values in column `VX` are the floors of the values in column `V2`.

Related Functions

Function	Description
`CEILING`	Computes the ceiling of each value in the specified data range
`FRACTION`	Returns the fractional part of each value in the specified data range
`TRUNCATE`	Returns the non-fractional part of each value in the specified data range

FORMAT

Syntax

`Format` has two forms, one for numeric datatypes and one for text/character datatypes.

For numeric datatypes:

`FORMAT(colName, width [, precision [, format_type [, alignment [, padding]]]])`

For text/character datatypes:

`FORMAT(colName, width [, alignment])`

Parameters

`colName`

The macro examines `colName` and determines its datatype, then imposes the appropriate rules for subsequent parameters accordingly.

`width`

Width should be large enough to hold the complete result, otherwise the result will be truncated. Acceptable values are from 1 to 29 if `colName` is numeric, otherwise from 1 to 255.

`precision`

Precision is number of digits after the decimal point. Acceptable values are from 0 to 15. If it's zero, then the result is integer. Default precision value is 2.

`format_type`

Valid keywords for `format_type` are:

PERIOD	Period(.) is used as decimal symbol. No digit grouping symbol is used. This is the default value.
COMMA	Comma(,) is used as decimal symbol. No digit grouping symbol is used.
PERIOD_COMMA	Period as decimal symbol and comma as digit grouping symbol.
COMMA_PERIOD	Comma as decimal symbol and period as digit grouping symbol.

`alignment`

Valid keywords for alignment are LEFT and RIGHT. Default value is RIGHT for numeric datatypes and LEFT for text/character datatypes.

`padding`

Valid keywords for padding are SPACE and ZERO. Default value is SPACE. ZERO is ignored (and instead SPACE is used) if alignment is LEFT.

Note that numeric strings held within a text/character datatype are treated as text/character. Also note that the numeric form takes multiple optional keywords, each with a default value. However, to override the default of second or subsequent optional keywords you MUST code the defaults for the preceding optional keywords (in effect they become required). For example: to override alignment to be LEFT you must code: FORMAT(myNumCol, 10, 2, PERIOD, LEFT).

Description

FORMAT converts numeric data to a string form with various formatting options to control and define the output string. This will be especially useful for creating Snapshot files with specific formats for mailing file purposes.

Examples

The following example defines a derived field using FORMAT.
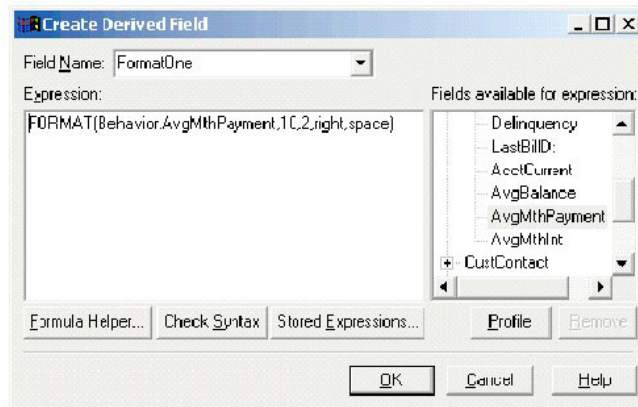
The following examples show the same field, AvgMthPayment, in three formats.

Unformatted:
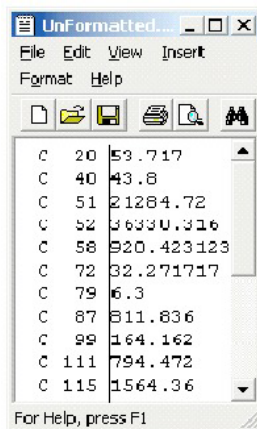
Formatted using FORMAT(Behavior.AvgMthPayment,10,2,right,space):

Formatted using FORMAT(Behavior.AvgMthPayment,10,4):

FRACTION

Syntax

FRACTION(data)

Parameters

data

The numerical values to compute the fraction of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

FRACTION calculates the fractional part of the values in the specified data range. It returns one new column for each input column, each containing the fractional part of the numbers in the corresponding input column.

Note: The `FRACTION` macro function and the `TRUNCATE` macro function are complementary in that they sum to the original values.

Examples

`TEMP = FRACTION(4.3)`   Creates a new column named TEMP containing the value `0.3`.
`TEMP = FRACTION(2.9)`   Creates a new column named TEMP containing the value `-0.9`.
`TEMP = FRACTION(V1)`   Creates a new column named TEMP, where each value is the fractional part of the contents of column `V1`.
`TEMP = FRACTION(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the fractional parts of the contents of column `V1`, the values of the `VX` column are the fractional parts of the contents of column `V2`, and the values of the `VY` column are the fractional parts of the contents of column `V3`.
`TEMP = FRACTION(V1[10:20])`   Creates a new column named TEMP, where the first 11 cells contain the fractional parts of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
TEMP = FRACTION(V1[50:99]:V2)   Creates two new columns named `TEMP` and `VX`, each with values in rows 1-50 (the other cells are empty). The values in column `TEMP` are the fractional parts of the rows of column `V1`, and the values in column `VX` are the fractional parts of the values in column `V2`.

Related Functions

Function	Description
`CEILING`	Computes the ceiling of each value in the specified data range
`FLOOR`	Computes the floor of each value in the specified data range
`TRUNCATE`	Returns the non-fractional part of each value in the specified data range

GAUSS

Syntax

`GAUSS(data [, mean, std])`

Parameters

`data1`

The cell range to compute the Gaussian of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

mean

The mean of the Gaussian. If this parameter is not provided, the default is zero. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in mean must equal the number of columns in data, unless mean is a constant or a single column. For the format definition of mean, (same as the definition of data) see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

std

The standard deviation of the Gaussian. If this parameter is not provided, the default is one. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in std must equal the number of columns in data, unless std is a constant or a single column. For the format definition of std, (same as the definition of data) see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Syntax

GAUSS computes the Gaussian value of the numbers in the specified data range. It returns one new column for each input column, each containing the Gaussian value of the corresponding input. GAUSS is computed as follows:

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The mean and standard deviation parameters are treated as follows:
- If no mean and standard deviation are provided, zero and one are used, respectively.
- If mean and std are constants, these values are used to specify the Gaussian for all values in data.
- If mean and std are single columns, the corresponding row values are used for each row of data.
- If mean and std are column ranges (both must be the same number of columns as data), each cell in data uses its individual pair of corresponding cells in mean and std.

Examples

```
TEMP = GAUSS(0) or TEMP = GAUSS(0, 0, 1)
```

Creates a new column named TEMP containing the value 0.4.

`TEMP = GAUSS(V1)`
Creates a new column named TEMP, where each value is the Gaussian of the corresponding row of column V1, using a zero-mean, unit-variant Gaussian.
`TEMP = GAUSS(V1:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the Gaussians of column V1, the values of the VX column are the Gaussians of column V2, and the values of the VY column are the Gaussians of column V3. The Gaussian is zero-mean and unit-variant.
`TEMP = GAUSS(V1[1:50]:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the Gaussians of rows 1-50 column V1, the values of the VX column are the Gaussians of rows 1-50 column V2, and the values of the rows of the VY column are the Gaussians of column V3. The Gaussian is zero-mean and unit-variant.
`TEMP = GAUSS(V1, 0, 3.5)`
Creates a new column named TEMP, where each value is the Gaussian of the corresponding row of column V1. The Gaussian has a mean of 0 and a standard deviation of 3.5.
`TEMP = GAUSS(V1, V2, V3)`
Creates a new column named TEMP, where each value is the Gaussian of column V1, using the corresponding row value of column V2 as the mean and the corresponding row value of column V3 as the standard deviation.
`TEMP = GAUSS(V1:V2, V3:V4, V5:V6)`
Creates two new columns named TEMP and VX. The column TEMP contains the Gaussians of the values in V1 using the corresponding rows of column V3 as the mean and the corresponding rows of column V5 as the standard deviation. The column VX contains the Gaussians of the values in V2 using the corresponding rows of column V4 as the mean and the corresponding rows of column V6 as the standard deviation.

Related Functions

`RANDOM_GAUSS`	Returns the specified number of random values from a Gaussian distribution

GAUSS_AREA

Syntax

`GAUSS_AREA(data [, mean, std])`

Parameters

`data1`

The cell range to compute the area under the Gaussian for. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `mean`, (same as the definition of `data`) see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`mean`

The mean of the Gaussian. If this parameter is not provided, the default is zero. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `mean` must equal the number of columns in `data`, unless `mean` is a constant or a single column. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`std`

The standard deviation of the Gaussian. If this parameter is not provided, the default is one. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `std` must equal the number of columns in `data`, unless `std` is a constant or a single column. For the format definition of `std`, (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`GAUSS_AREA` computes the area under the Gaussian (from $-\infty$ to the specified data value X) of the numbers in the specified data range (see the following figure). It returns one new column for each input column, each containing the area under the Gaussian of the corresponding input.

Area Under a Gaussian Computed by GAUSS_AREA
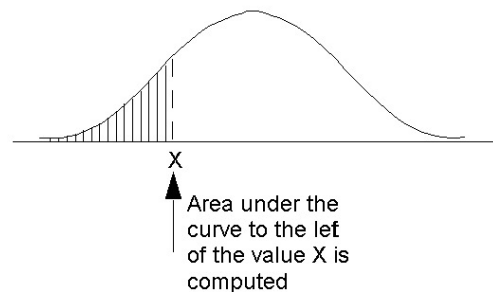
X

Area under the
curve to the left
of the value X is
computed

The mean and standard deviation parameters are treated as follows:
- If no mean and standard deviation are provided, zero and one are used, respectively.
- If `mean` and `std` are constants, these values are used to specify the Gaussian for all values in `data`.
- If `mean` and `std` are single columns, the corresponding row values are used for each row of `data`.
- If `mean` and `std` are column ranges (both must be the same number of columns as `data`), each cell in `data` uses its individual pair of corresponding cells in `mean` and `std`.

Note: In the last case above, when `mean` and `std` are column ranges, the length of each column determines how many rows will be present in the corresponding output column. If a column of `mean` or `std` is a single cell, that value will be used

for all row values of data. If mean or std contain multiple rows, the corresponding rows are calculated. Rows of data for which there are no corresponding values in mean and std are not computed.

Examples

TEMP = GAUSS_AREA(0) or TEMP = GAUSS_AREA(0, 0, 1)
Creates a new column named TEMP containing the value 0.5.
TEMP = GAUSS_AREA(V1)
Creates a new column named TEMP, where each value is the area under the Gaussian of the corresponding row of column V1, using a zero-mean, unit-variant Gaussian.
TEMP = GAUSS_AREA(V1:V3)
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the areas under the Gaussians of column V1, the values of the VX column are the areas under the Gaussians of column V2, and the values of the VY column are the areas under the Gaussians of column V3. The Gaussian is zero-mean and unit-variant.
TEMP = GAUSS_AREA(V1[1:50]:V3)
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the areas under the Gaussians of rows 1-50 column V1, the values of the VX column are the areas under the Gaussians of rows 1-50 column V2, and the values of the rows of the VY column are the areas under the Gaussians of column V3. The Gaussian is zero-mean and unit-variant.
TEMP = GAUSS_AREA(V1, 0, 3.5)
Creates a new column named TEMP, where each value is the area under the Gaussian of the corresponding row of column V1. The Gaussian has a mean of 0 and a standard deviation of 3.5.
TEMP = GAUSS_AREA(V1, V2, V3)
Creates a new column named TEMP, where each value is the area under the Gaussian of column V1, using the corresponding row value of column V2 as the mean and the corresponding row value of column V3 as the standard deviation.
TEMP = GAUSS_AREA(V1:V2, V3:V4, V5:V6)
Creates two new columns named TEMP and VX. The column TEMP contains the areas under the Gaussians of the values in V1 using the corresponding rows of column V3 as the mean and the corresponding rows of column V5 as the standard deviation. The column VX contains the areas under the Gaussians of the values in V2 using the corresponding rows of column V4 as the mean and the corresponding rows of column V6 as the standard deviation.

GE

Syntax

data1 GE data2 data1 >= data2

Parameters

data1

The numerical cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The number(s) to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`GE` compares the two specified data ranges, returning a one if the values in the first data set are greater than or equal to the values in the second data set or a zero otherwise. It returns a new column for each input column, each containing the corresponding column in `data1` compared to the corresponding column of `data2` (that is, the first column of `data1` is compared to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data` is compared to that value. If `data2` is a column, the calculations are performed on a row-by-row basis. The values in `data1` are compared to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The `GE` operator can be abbreviated with a greater than sign followed by an equal sign (>=).

This macro is available in IBM Interact.

Examples

`TEMP = 9 GE 4 or TEMP = 9 >= 4` Creates a new column named `TEMP` containing the value one (since nine is greater than four).
`TEMP = V1 >= 8` Creates a new column named `TEMP`, where each value is one if the corresponding row value of the column `V1` is greater than or equal to the number eight, otherwise zero.
`TEMP = V1:V3 >= 2` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the contents of column `V1` compared to the value two, the values of the `VX` column are the contents of column `V2` compared to the value two, and the values of the `VY` column are the contents of column `V3` compared to the value two.
`TEMP = V1 >= V1` Creates a new column named `TEMP` containing all ones (since every number is equal to itself).
`TEMP = V1 >= V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` compared to the corresponding row value of column `V2`.

```
TEMP = V1:V3 >= V4:V6
```

Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` compared to the corresponding row values of column `V4`. The column `VX` compares columns `V2` and `V5`. The column `VY` compares columns `V3` and `V6`.

```
TEMP = V1[10:20] >= V2 or TEMP = V1[10:20] >= V2[1:11]
```

Creates a new column named `TEMP`, where the first 11 cells contain the results of comparing the values in rows 10-20 of column `V1` to the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

NE	Returns TRUE if one data range is not equal to another

GRID

Syntax

GRID(col1 [, col2]...)

Parameters

col1

The first column to produce the grid from. This can be a constant, a column, a single-column cell range, or any expression evaluating to one of the above.

col2

One or more additional columns to use in producing the grid. This can be a constant, a column, a single-column cell range, or any expression evaluating to one of the above.

Description

GRID generates a grid of all possible combinations of values using cells in each of the input columns. It returns a new column for each input column. If any of the inputs are constants, each value in the other input columns are paired with the that constant value.

Using the notation V1[1] for the first cell in column V1, V1[2] for the second cell, and so on, the expression GRID(V1, V2) would produce:

```
V1[1] V2[1] V1[1] V2[2] V1[1] V2[3] : : V1[2] V2[1] V1[2] V2[2] V1[2] V2[3]
: : : :
```

All cells in column V1 are paired against each cell in column V2. GRID behaves similarly for more than two input columns. The number of rows generated is equal to the product of the length of the input columns.

Note: The concatenated length of all of the input arguments cannot exceed $(2^{16}-1)/16$.

Examples

`TEMP = GRID(1,2)` Creates two new columns named TEMP and VX, containing the values 1 and 2, respectively.
`TEMP = GRID(COLUMN(1, 2), COLUMN(3, 4)) Creates two new columns named TEMP and VX. The rows across these two columns are: 1 3 1 4 2 3 2 4`
`TEMP = GRID(V1)` Creates a new column named TEMP containing a copy of the values in column V1.
`TEMP = GRID(V1, 3)` Creates two new columns named TEMP and VX. The column TEMP is a copy of column V1, and column VX contains the value 3 for each row of column V1.
`TEMP = GRID(V1, V2)` Creates two new columns named TEMP and VX, where each row represents one of the possible cell combinations (see description).
`TEMP = GRID(V1, V3, V7)` Creates three new columns named TEMP, VX, and VY, where each row represents one of the possible cell combinations (see description).

Related Functions

Function	Description
`SLIDE_WINDOW`	Creates a pattern from a specified window and slides it down to create the next pattern

GROUPBY

Syntax

`GROUPBY(group_field, keyword, rolled_field [,output_field])`

Parameters

- `group_field`

  Specifies the variable over which records are grouped (that is, all the same values of the specified variable are grouped together).

- `keyword`

  Specifies the summary roll-up function to perform on the rolled-field.

- `rolled_field`

  Specifies the variable to be summarized or rolled up.

- `output_field`

  Identifies an alternate variable to be returned for a single row of a group and can be used only with the keywords `MinOf`, `MaxOf`, and `MedianOf`.

Description

GROUPBY summarizes across multiple rows of data within a group. The output of this function is a single column. The output is the result of the operation specified by `keyword` on the `rolled_field` over the homogeneous group specified by the `group_field`. If there is more than one answer satisfying a specified condition, the first one encountered is returned.

If the optional `output_field` is not supplied, then the output is the result of the operation on `rolled_field`. If `output_field` is supplied, then the result is the `output_field` of the row within the group.

If there are multiple rows within a group that satisfy the specified condition (for example, there are ties for the max value), the `output-field` associated with the first row satisfying the condition is returned.

Note: To work with grouping over multiple columns, you can enclose a list of field names, separated by commas, within a set of "curly" brackets "{ }" and using this as the first parameter in the `GROUPBY` macro call.

Supported keywords are as follows (case insensitive):

Keyword	String? Yes/No	Description
CountOf	Yes	Returns the number of records in each group (`rolled_field` can be numeric or string; the returned value is the same regardless of the value of `rolled_field`).
MinOf	Yes	Returns the minimum value of `rolled_field` in each group (`rolled_field` can be numeric or string; if `rolled_field` is a string, the value closest to the beginning of the alphabet where alphabetically sorted is returned).
MaxOf	Yes	Returns the maximum value of `rolled_field` in each group (`rolled_field` can be numeric or string; if `rolled_field` is a string, the value closest to the end of the alphabet when alphabetically sorted is returned).
DiffOf	Yes	Returns the number of distinct values of `rolled_field` in each group (`rolled_field` can be numeric or string).

AvgOf	No	Returns the average value of `rolled_field` in each group (`rolled_field` must be numeric).
ModeOf	Yes	Returns the modal value (that is, the most commonly occurring value) of `rolled_field` in each group (`rolled_field` can be numeric or string).
MedianOf	Yes	Returns the median value (that is, the middle value when sorted by `rolled_field`) of `rolled_field` in each group (`rolled_field` can be numeric or string; if `rolled_field` is a string, the values are sorted alphabetically).
OrderOf	Yes	Returns the order of `rolled_field` in each group (`rolled_field` must be numeric). If multiple records have the same value, they all receive the same value.
SumOf	No	Returns the sum of `rolled_field` in each group (`rolled_field` must be numeric).
StdevOf	No	Returns the standard deviation of `rolled_field` in each group (`rolled_field` must be numeric).
IndexOf	Yes	Returns the 1-based index (ordered by rolled_field) of each record (rolled_field can be numeric or string). The sort order is ascending. Note: For numeric fields, the sort order of RankOf and IndexOf can be made descending by putting a minus sign (-) in front of the sort field.

RankOf	Yes	
		Returns the 1-based category (ordered by rolled_field) in which each record lies (rolled_field can be numeric or string). The sort order is ascending. Note: For numeric fields, the sort order of RankOf and IndexOf can be made descending by putting a minus sign (-) in front of the sort field.

Examples

`GROUPBY (Household_ID, SumOf, Account_Balance)`
Computes the sum of all account balances by household.
`GROUPBY (Cust_ID, MinOf, Date(Account_Open_Date), Acc_Num)`
Returns the account number of first account opened by customer.

GROUPBY_WHERE

Syntax

`GROUPBY_WHERE(group_field, keyword, rolled_field, where_value [,output_field])`

Parameters

- group_field

  Specifies the variable over which records are grouped (that is, all the same values of the specified variable are grouped together).

- keyword

  Specifies the summary roll-up function to perform.

- rolled_field

  Specifies the variable to be summarized or rolled up.

- where_value

  An expression that evaluate to a one or zero value that specifies which rows are to be included in the roll-up operation.

- output_field

  Identifies an alternate variable to be returned for a single row of a group and can be used only with the keywords `MinOf`, `MaxOf`, and `MedianOf`

Description

`GROUPBY_WHERE` summarizes across specific rows of data within a group. The output of this function is a single column. The output is the result of the operation specified by `keyword` on the `rolled_field` over the homogeneous group specified by the `group_field`, filtered by the `where_value`. Only rows with a `where_value` of one are included in the calculation.

If the optional `output_field` is not supplied, then the result is the result of the operation on `rolled_field`. If `output_field` is supplied, then the result is the `output_field` of the row within the group.

Note: See "GROUPBY" on page 107 for more information on valid values for `keyword`.

Examples

`GROUPBY_WHERE (Household_ID, SumOf, Account_Balance, Account_Balance>0)`   Computes the sum of all accounts with positive balances for each household.

`GROUPBY_WHERE (Cust_ID, AvgOf,  Purchase_Amt,Date(Current_Date) -`
 `Date(Purchase_Date)<90)`
 	
 Computes the average purchase amount for each customer for purchases in the last 90 days.

GT

Syntax

`data1 GT data2 data1 > data2`

Parameters

`data1`

The numerical cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`data2`

The numbers to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

GT compares the two specified data ranges, returning a one if the values in the first data set are greater than the values in the second data set or a zero otherwise. It returns a new column for each input column, each containing the corresponding column in `data1` compared to the corresponding column of `data2` (that is, the first column of `data1` is compared to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data` is compared to that value. If `data2` is a column, the calculations are performed on a row-by-row basis. The values in `data1` are compared to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The `GT` operator can be abbreviated with a greater than sign (>).

This macro is available in IBM Interact.

Examples

`TEMP = 3 GT 4 or TEMP = 3 > 4` Creates a new column named TEMP containing the value zero (since three is not greater than four).
`TEMP = V1 > 8` Creates a new column named TEMP, where each value is one if the corresponding row value of the column V1 is greater than the number eight, otherwise zero.
`TEMP = V1:V3 > 2` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the contents of column V1 compared to the value two, the values of the VX column are the contents of column V2 compared to the value two, and the values of the VY column are the contents of column V3 compared to the value two.
`TEMP = V1 > V1` Creates a new column named TEMP containing all zeros (since no number is greater than itself).
`TEMP = V1 > V2` Creates a new column named TEMP, where each value is the row value of column V1 compared to the corresponding row value of column V2.
`TEMP = V1:V3 > V4:V6` Creates three new columns named TEMP, VX, and VY. The column TEMP contains the values in V1 compared to the corresponding row values of column V4. The column VX compares columns V2 and V5. The column VY compares columns V3 and V6.
`TEMP = V1[10:20] > V2 or TEMP = V1[10:20] > V2[1:11]` Creates a new column named TEMP, where the first 11 cells contain the results of comparing the values in rows 10-20 of column V1 to the values in rows 1-11 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
EQ	Returns TRUE if one data range is equal to another
GE	Returns TRUE if one data range is greater than or equal to another
LE	Returns TRUE if one data range is less than or equal to another
LT	Returns TRUE if one data range is less than another
NE	Returns TRUE if one data range is not equal to another

HISTOGRAM

Syntax

```
HISTOGRAM(data, bin_col)
```

Parameters

`data`

The cell range to compute the histogram of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. All columns in `data` must be the same data type (that is, numeric or text string). For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`bin_col`

The values for the bin boundaries. This can be a constant value, a column, a single-column cell range, or an expression evaluating to any of the above. The data type of `bin_col` must be the same as `data`. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`HISTOGRAM` computes the histogram (that is, frequency of occurrence of data values in various bins) of the values in the specified data range. It returns a single column with the number of data values in `data` that fall within the corresponding bin range specified by `bin_col`.

For numerical values, each two adjacent rows of `bin_col` form a "bin". Any value in `data` that falls within a bin is accumulated for that bin. The output column contains the final count of the number of data values within each bin. The first boundary value is *included* in the bin; the second boundary value is *excluded*. For example, the pair of boundary values 1 and 2 will contain a count of all values in `data` are greater than or equal to 1 and less than 2. The length of the output column is one less than the length of `bin_col`.

For text strings, only exact matches of the text string in `bin_col` is counted in that bin. The length of the output column is the length of `bin_col`. For numerical data, if `bin_col` is scalar (that is, contains a single cell value), than the number of items in `data` is counted.

Note: The `HISTOGRAM` macro function places data points into bins differently than **IBM PredictiveInsight** 's histogram graph. The histogram graph exclude the minimum (except for the leftmost bin) and include the maximum of each bin boundary.

Examples

```
TEMP = HISTOGRAM(1...10, COLUMN(1, 3, 10)
```

Creates a new column named `TEMP` containing the values 2 and 7.

`TEMP = HISTOGRAM("a","b","a"),COLUMN("a","b","c"))`   Creates a new column named TEMP containing the values 2, 1, and 0.
`TEMP = HISTOGRAM(V1, COLUMN(1, 25, 50, 75, 101)`   Creates a new column named TEMP containing four values. The first value is the number of values in column V1 greater than or equal to 1 and less than 25. The second value is the number of values in column V1 greater than or equal to 25 and less than 50. The third and fourth values contain the counts in the third and fourth quartiles, respectively.
`TEMP = HISTOGRAM(V1:V3, V4)`   Creates a new column named TEMP, where each value a count of the number of values in columns V1 - V3 that fall within the bin boundaries specified by column V4.
`TEMP = HISTOGRAM(V1[50:100]:V5, V6[1:10])`   Creates a new column named TEMP containing 10 values. Each value is the number of data values in rows 50-100 of columns V1 - V5 that fall within the bin boundaries specified by rows 1-10 of column V6.

Related Functions

Function	Description
COUNT	Counts the number of cells containing values in the specified data range

IF

Syntax

`IF(predicate_col, then_value) IF(predicate_col, then_value, else_value)`

Parameters

`predicate_col`

A column of boolean values or an expression evaluating to a single column of boolean values. Boolean values are interpreted as zero or non-zero. This column should contain at least as many rows as the data range from which data is being extracted. Otherwise, `predicate_col` will be a limit to the number of rows processed by the EXTRACT macro function (see "Description" below).

`then_value`

The value(s) to return if the corresponding row of `predicate_col` contains a non-zero value. This can be a constant value, a column, or an expression evaluating to any of the above. See "Macro Function Parameters" on page 8 for the format definition of `then_value` (same as `data`).

`else_value`

If this optional parameter is provided, it is returned if the corresponding row of `predicate_col` contains a zero. This can be a constant value, a column, or an expression evaluating to any of the above. If `else_value` is not provided, a zero is returned whenever `predicate_col` evaluates to false. See "Macro Function Parameters" on page 8 for the format definition of `else_value` (same as `data`).

Description

IF evaluates the expression in `predicate_col` and returns `then_value` if the expression is true, or `else_value` if the expression is false. It returns the same number of columns in `then_value` and `else_value`. The new column(s) will contain the corresponding row value(s) from `then_value` if the value of `predicate_col` is non-zero. If `else_value` is provided, it is returned when the value of `predicate_col` is zero. If `else_value` is not provided, zero is returned.

Since IF operates on a row-by-row basis, it produces a result for each row up to the last value of the shortest column (that is, the shortest column out of `predicate_col`, `then_value`, and `else_value`).

Note: Generally, you will want to create a predicate column using one of the comparison macro functions (for example, ==, >, <, ISEVEN, ISODD, and so on).

This macro is available in IBM Interact.

Examples

`TEMP = IF(1, V1)`Creates a new column named TEMP containing a copy of column V1.
`TEMP = IF(V1, 1, 0)` Creates a new column named TEMP, where each value is one if the corresponding value of column V1 is non-zero, otherwise zero.
`TEMP = IF(V3, V1, V2)` Creates a new column named TEMP, where each value is copied from column V1 if the corresponding value of column V3 is non-zero; otherwise the value is copied from column V2.
`TEMP = IF(ABS(V1-AVG(V1)) < STDV(V1), V1)` Creates a new column named TEMP containing each value in column V1 that is less than one standard deviation away from the mean.
`TEMP = IF(V3[20:30], V1[30:40], V2)` Creates a new column named TEMP containing values for rows 10-20. Each value is copied from column V1 (cells 10-20) if the corresponding value of column V3 (cells 30-40) is non-zero; otherwise the value is copied from column V2 (cells 1-11).

Related Functions

Function	Description
EXTRACT	Extracts rows given the values in a predicate column
SELECT	Returns the specified column(s) from a data range

IN

Syntax

`valuet IN (value1 AND value2 . . . .) or valuet IN subquery`

Parameters

The first form permits using a list of values instead of a subquery.

The second form uses a subquery that is evaluated to produce an intermediate result, against which further processing can be performed.

Description

The IN predicate lets you use a list of values instead of a subquery, or will introduce a subquery.

Note: IN differs from ISMEMBER because IN is performed on the database (if possible), while ISMEMBER is computed on the server.

Note: The IN predicate has a negative version, NOT IN. The format for this is identical to IN. NOT IN is true only if the provided value is not found in the values returned by the subquery.

This macro is available in IBM Interact.

Important: When using IN in IBM Interact, you can only use the `value IN (value1 AND value2 . . . .)` syntax.

Examples

`TEMP = IN(25, COLUMN(1...10))`
Returns the specified column(s) from a data range
`TEMP = IN("cat", COLUMN("cat", "dog", "bird"))`
Creates a new column named `TEMP` containing the value one.
`TEMP = IN(V1, V1)`
Creates a new column named `TEMP` containing all ones.
`TEMP = IN(V1, V2)`
Creates a new column named `TEMP`, where each value is a one if the corresponding row of column `V1` contains a value in column `V2`, else a zero.

Examples

`SELECT title, current_rental_price FROM movie_titles WHERE our_cost IN (14.95, 24.95, 29.95 ) ;`
means that all movies that cost either $14.95, $24.95, or $29.95 will evaluate as TRUE to the IN predicate and will, in turn, produce a list of titles and accompanying current rental prices for those movies.
`UPDATE movie_titles SET current_rental_price = (regular_rental_price * .9) WHERE title IN ( SELECT movie_title FROM movie_stars WHERE actor_last_name = 'Stewart' AND actor_first_name = 'James' ) ;`
After the subquery produces its list of movie titles, the list is processed against the MOVIE_TITLES table and the appropriate rental prices are discounted for *Philadelphia Story*, *It's a Wonderful Life*, and so on.

INIT

Syntax

`INIT(val1 [, val2]...) INIT(column)`

Parameters

`val1`

The numerical value of the recursive function at time (t-1).

`val2`

The numerical value of the recursive function at time (t-n), where n is the parameter number. This parameter can be repeated multiple times to provide initial values for an indefinite number of previous time steps.

`column`

A column of numerical values. The first cell will be assigned to the time step (t-1), the second cell value to (t-2), and so on.

Description

INIT specifies the initial values for a recursive function definition. The first provided value is assigned to the time step (t-1), the second to (t-2), and so on. If a time step is not initialized using INIT, its value is assumed to be zero. For example, given the statement

`V1 = INIT(1, 2, 3)`

the value for time step (t-4) is zero (as are all other time steps further back in time). An INIT statement is required before defining any recursive function.

Note: To initialize all values to zero, you can simply specify `INIT()` without any arguments.

The INIT macro function does not return any values. If used alone in a function definition, it returns a blank column.

Examples

`TEMP = INIT() t = 1 to 10 TEMP = 1 + TEMP[t-1]`
Creates a new column named TEMP containing the values 1-10.
`TEMP = INIT(1) t = 1 TO 100 TEMP = TEMP[t-1]+TEMP[t-1]`
Creates a new column named TEMP containing the values 2, 4, 8, 16, 32, and so on. The first 100 cells of TEMP will contain values.
`TEMP = INIT(1, 2, 3, 4, 5) t = 1 to 500 TEMP = TEMP[t-5]`
Creates a column named TEMP containing the sequence of values 5, 4, 3, 2, 1 repeated 100 times.

```
TEMP = INIT(1, 2, 3) t = 1 to 1000 TEMP = 2*TEMP[t-1] + 4*TEMP[t-2]^2 - TEMP[t-3]
```

Creates a new column named TEMP containing values for the cursive function:

$$ \mathrm{TEMP} \;=\; 2\mathrm{*TEMP}(t-1) + 4\mathrm{*TEMP}(t-2)^2 - \mathrm{TEMP}(t-2) $$

1000 cell values are computed.

Related Functions

Function	Description
TO	Generate range operator

INT

Syntax

```
INT(data)
```

Parameters

```
data
```

The numerical values to round down to an integer value. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

INT calculates the greatest integer less than the values (also known as the floor) in the specified data range. INT returns one new column for each input column, each containing the floor of numbers in the corresponding input column.

Note: This is the same as the FLOOR macro function.

Examples

`TEMP = INT(4.7)`
Creates a new column named TEMP containing the value 4.
`TEMP = INT(-1.5)`
Creates a new column named TEMP containing the value -2.
`TEMP = INT(V1)`
Creates a new column named TEMP, where each value is the largest integer less than or equal to the contents of column V1.
`TEMP = V1 - INT(V1)`
Creates a new column named TEMP containing the decimal portion of each value in column V1.

`TEMP = INT(V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the largest integers less than or equal to the contents of column `V1`, the values of the `VX` column are the largest integers less than or equal to the contents of column `V2`, and the values of the `VY` column are the largest integers less than or equal to the contents of column `V3`.
`TEMP = INT(V1[10:20])`
Creates a new column named `TEMP`, where the first 11 cells contain the largest integers less than or equal to the corresponding values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = INT(V1[1:5]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the largest integers less than or equal to the corresponding row values of column `V1`, and the values in column `VX` are the largest integers less than or equal to the corresponding row values of column `V2`.

Related Functions

Function	Description
ROUND	Computes the rounded value of the contents of the specified data range
TRUNCATE	Returns the non-fractional part of each value in the specified data range

---

INTEGRAL

Syntax

`INTEGRAL(data [, multiplier])`

Parameters

`data`

The numerical values to compute the integral of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`multiplier`

A value to multiply each value in `data` by. This can be a constant value or an expression evaluating to a constant.

Description

`INTEGRAL` calculates the integral of the values in a time-series. Each value is the sum of all previous values in time. If a value is provided for `multiplier`, each value is multiplied by the specified value. `INTEGRAL` returns one new column for

each input column, each containing integral of the values in the corresponding input column.

Examples

`TEMP = INTEGRAL(5)`   Creates a new column named TEMP containing the value 5.
`TEMP = INTEGRAL(COLUMN(1,2,3))`   Creates a new column named TEMP containing the values 1, 3, and 6.
`TEMP = INTEGRAL(COLUMN(1,2,3), 2)`   Creates a new column named TEMP containing the values 2, 6, and 12.
`TEMP = INTEGRAL(V1)`   Creates a new column named TEMP, where each value is the sum of all previous cells in column V1.
`TEMP = INTEGRAL(V1, 10)`   Creates a new column named TEMP, where each value is the sum of all previous cells in column V1 times 10.
`TEMP = INTEGRAL(V1:V3)`   Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the integrals of column V1, the values of the VX column are the integrals of column V2, and the values of the VY column are the integrals of column V3.
`TEMP = INTEGRAL(V1:V3)`   Creates a new column named TEMP, where the first 11 cells contain the integrals of the corresponding values in rows 10-20 of column V1. The other cells in TEMP are empty.
`TEMP = INTEGRAL(V1:V3)`   Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the integrals of the corresponding row values of column V1, and the values in column VX are the integrals of the corresponding row values of column V2.

Related Functions

Function	Description
`DTEMP = INTEGRAL(V1:V3)`	Computes the derivative of the values in the specified data range
`SUM` or `TOTAL`	Computes the sum of a range of cells

INVERSE

Syntax

`INVERSE(data)`

Parameters

`data`

The numerical values to compute the inverse of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`INVERSE` calculates the negative of the values in the specified data range. It returns *-x* (that is, negative values are returned as positive values, and positive values are returned as negative values). `INVERSE` returns one new column for each input column, each containing the inverse of the values in the corresponding input column.

Note: To invert a value or a column, precede it with a minus sign (`-`). For example, `V2 = -V1` is the same as `V2 = INVERSE(V1)`.

Examples

`TEMP = INVERSE(3.2)` Creates a new column named `TEMP` containing the value `-3.2`.
`TEMP = INVERSE(V1)` Creates a new column named `TEMP`, where each value is the negative of the values in column `V1`.
`TEMP = INVERSE(V1:V3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the negatives of values in column `V1`, the values of the `VX` column are the negatives of the values in column `V2`, and the values of the `VY` column are the negatives of the values in column `V3`.
`TEMP = INVERSE(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the negatives of the values of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = INVERSE(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the negatives of the values of the corresponding rows of column `V1`, and the values in column `VX` are the negatives of the values of the corresponding rows of column `V2`.

Related Functions

Function	Description
`ABS`	Computes the absolute value of the contents of the specified data range
`NOT`	Computes the logical NOT of the contents of the specified data range
`SIGN`	Computes the sign (positive or negative) of the values in the specified data range

IS

Syntax

IS <keyword>

Parameters

keyword

Search condition, typically "NULL," "TRUE," "UNKNOWN," and "FALSE."

Description

IS is used in complex search conditions. The more complex the search, the more useful the IS condition can be. These Boolean search conditions provide an alternative way of expressing the basic search conditions.

This macro is available in IBM Interact.

IS returns different results in IBM Interact from IBM Campaign. NULL returns 1 if there is at least one NULL value for an audience id. UNKNOWN returns 1 for an audience id if it doesn't have any value.

Examples

SELECT customer FROM customer_table1 WHERE (last_name = "Smith" AND first_name = "John") IS TRUE ;
will produce a list of all customers having the name John Smith.
SELECT customer FROM customer_table1 WHERE (last_name = "X" AND first_name = "X") IS UNKNOWN ;
asks for any non-null value.
SELECT cost FROM cost_table1 WHERE (current_cost = "200" IS FALSE ;
lists all values from the cost table that are not $200.

ISERROR

Syntax

ISERROR(data)

Parameters

data

The values to test if any of the rows contain an error (that is, a ??? cell). This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`ISERROR` checks if any cell of each row of the specified data range contains an error (that is, a `???` cell). It returns one new column, each row containing a one if the corresponding row of `data` contains an error. Otherwise, it contains a zero. This row-by-row calculation produces a result for each row up to the last value of the longest column.

Note: This function is useful for detecting errors in a column and then using the `EXTRACT` macro function to pull out the good rows of data.

Examples

`TEMP = ISERROR(-3)`   Creates a new column named `TEMP` containing the value zero.
`TEMP = ISERROR(V1)`   Creates one new columns named `TEMP`, where each value is a one if the corresponding row of column `V1` contains `???`, otherwise, a zero.
`TEMP = ISERROR(V1:V3)`   Creates one new columns named `TEMP`, where each value is a one if any of the cells in the corresponding rows of column `V1` - `V3` contains `???`, otherwise, a zero.
`TEMP = ISERROR(V1[50:100]:V10)`   Creates one new columns named `TEMP`, with values in rows 1-50. Each value is a one if any of the cells in rows 50-100 of columns `V1` - `V10` contains `???`, otherwise, a zero.

Related Functions

Function	Description
`EXTRACT`	Extracts rows given the values in a predicate column

ISEVEN

Syntax

`ISEVEN(data)`

Parameters

`data`

The numerical values to test if they are even. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

ISEVEN tests each value in the specified data set for evenness. It returns one new column for each input column, each containing a one for all even values (that is, the value modulo two is zero) or a zero for all non-even values (that is, odd values).

Note: For non-integer values, the macro function INT is applied first. For example, ISEVEN(2.5) = 1, since 2 is even.

Examples

`TEMP = ISEVEN(-3)` Creates a new column named TEMP containing the value zero.
`TEMP = ISEVEN(MERGE(3, 2, 0))` Creates three new columns named TEMP, VX, and VY. TEMP contains the value 0, VX contains the value 1, and VY contains the value 1.
`TEMP = ISEVEN(V1)` Creates a new column named TEMP, where each value is the result of testing the contents of column V1 for evenness.
`TEMP = ISEVEN(V1:V3)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the results of testing the contents of column V1 for evenness, the values of the VX column are the results of testing the contents of column V2 for evenness, and the values of the VY column are the results of testing the contents of column V3 for evenness.
`TEMP = ISEVEN(V1[10:20])` Creates a new column named TEMP, where the first 11 cells contain the results of testing the values in rows 10-20 of column V1 for evenness. The other cells in TEMP are empty.
`TEMP = ISEVEN(V1[1:5]:V2)` Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the results of testing the corresponding rows of column V1 for evenness, and the values in column VX are the results of testing the corresponding rows of column V2 for evenness.

Related Functions

Function	Description
ISODD	Tests if input values are odd (that is, not divisible by two)
ISMEMBER	Tests an input range against a "table" of values, returning one if a value is contained in the table, else zero

ISMEMBER

Syntax

```
ISMEMBER(data, table)
```

Parameters

```
data
```

The values to test if they members of a table. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. If `data` contains more than one column, all columns must be the same data type (either numeric or text strings). For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

```
table
```

The table values to compare against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. If `table` contains more than one column, all columns must be the same data type as `data` (either numeric or text strings). The number of data values in `table` cannot exceed 16 million. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Note: ISMEMBER differs from IN because ISMEMBER is computed on the server while IN is performed on the database whenever possible.

Description

`ISMEMBER` compares the data values in the specified data range to a table of data values. It returns one new column for each input column in `data`, each containing a one if the corresponding input value is a member of `table`, else a zero.

Examples

`TEMP = ISMEMBER(25, COLUMN(1...10))`
Creates a new column named TEMP containing the value zero.
`TEMP = ISMEMBER("cat", COLUMN("cat", "dog", "bird"))`
Creates a new column named TEMP containing the value one.
`TEMP = ISMEMBER(V1, V1)`
Creates a new column named TEMP containing all ones.
`TEMP = ISMEMBER(V1, V2)`
Creates a new column named TEMP, where each value is a one if the corresponding row of column V1 contains a value in column V2, else a zero.
`TEMP = ISMEMBER(V1:V2, V5:V10)`
Creates two new columns named TEMP and VX. The column TEMP contains a one if the corresponding row of column V1 is a member of columns V5 - V10, else a zero. The column VX P contains a one if the corresponding row of column V2 is a member of columns V5 - V10, else a zero.

```
TEMP = ISMEMBER(V1[10:15]:V2, V3[1:100]:V6)
```

Creates two new columns named TEMP and VX, each with values in rows 1-6 (the other cells are empty). The values in column TEMP are one if the contents of rows 10-15 of column V1 are members of rows 1-100 of columns V3 - V6. The values in column VX are one if the contents of rows 10-15 of column V2 are members of rows 1-100 of columns V3 - V6.

Related Functions

Function	Description
ISEVEN	Tests if input values are even (that is, divisible by two)
ISODD	Tests if input values are odd (that is, not divisible by two).

ISODD

Syntax

ISODD(data)

Parameters

data

The numerical values to test if they are odd. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

ISODD tests each value in the specified data set for oddness. It returns one new column for each input column, each containing a one for all odd values (that is, the value modulo two is one) or a zero for all non-odd values (that is, even values).

Note: For non-integer values, the macro function INT is applied first. For example, ISODD(2.5) = 0, since 2 is not odd.

Examples

```
TEMP = ISODD(-3)
```

Creates a new column named TEMP containing the value one.

```
TEMP = ISODD(MERGE(1, 4, 0))
```

Creates three new columns named TEMP, VX, and VY. TEMP contains the value 1, VX contains the value 0, and VY contains the value 0.

```
TEMP = ISODD(V1)
```

Creates a new column named TEMP, where each value is the result of testing the contents of column V1 for oddness.

`TEMP = ISODD(V1:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the results of testing the contents of column V1 for oddness, the values of the VX column are the results of testing the contents of column V2 for oddness, and the values of the VY column are the results of testing the contents of column V3 for oddness.
`TEMP = ISODD(V1[10:20])`
Creates a new column named TEMP, where the first 11 cells contain the results of testing the values in rows 10-20 of column V1 for oddness. The other cells in TEMP are empty.
`TEMP = ISODD(V1[1:5]:V2)`
Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the results of testing the corresponding rows of column V1 for oddness, and the values in column VX are the results of testing the corresponding rows of column V2 for oddness.

Related Functions

Function	Description
`ISEVEN`	Tests if input values are even (that is, divisible by two)
`ISMEMBER`	Tests an input range against a "table" of values, returning one if a value is contained in the table, else zero

KURTOSIS

Syntax

`KURTOSIS(data [, keyword])`

Parameters

`data`

The numerical values to compute the kurtosis of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product. There must be at least four values in `data`.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL`

Performs the computation on all cells in `data` (default)

`COL`

Performs the computation separately for each column of `data`

`ROW`

Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`KURTOSIS` calculates the kurtosis of the values in the specified data range. Kurtosis is a measurement of the relative peakedness or flatness of a distribution compared to a normal distribution. The more negative the kurtosis is, the flatter the distribution. The more positive the kurtosis is, the sharper the peak of the distribution.

The kurtosis is calculated as follows:

$$\left\{ \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{j=1}^{n} \left(\frac{x_j - mean}{\sigma} \right) \right\} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

where *n* is the number of samples in the distribution, *mean* is the average, and ? is the standard deviation of the distribution. A minimum of three data values must be provided to compute kurtosis.

Examples

`TEMP = KURTOSIS(MERGE(3, 4, 5))` or `TEMP = KURTOSIS(MERGE(3, 4, 5), ALL)`   Creates a new column named TEMP containing the value -1.5.
`TEMP = KURTOSIS(V1)`   Creates a new column named TEMP containing a single value which is the kurtosis of the contents of column V1.
`TEMP = KURTOSIS(V1:V3)`   Creates a new column named TEMP containing a single value which is the kurtosis of the contents of columns V1, V2, and V3.
`TEMP = KURTOSIS(V1[10:20])`   Creates a new column named TEMP containing a single value which is the kurtosis of the cells in rows 10-20 of column V1.
`TEMP = KURTOSIS(V1[1:5]:V4)`   Creates a new column named TEMP containing a single value which is the kurtosis of the cells in rows 1-5 of columns V1 through V4.

TEMP = KURTOSIS(V1:V3, COL)
Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the kurtosis of the contents of column V1, the single value in the VX column is the kurtosis of the contents of column V2, and the single value in the VY column is the kurtosis of the contents of column V3.
TEMP = KURTOSIS(MERGE(1,4), COL)
Creates two new columns named TEMP and VX, each containing the value -3.
TEMP = KURTOSIS(V1[1:5]:V3, COL) or TEMP = KURTOSIS(V1[1:5]:V3[1:5], COL)
Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the kurtosis of the cells in rows 1-5 of column V1, the value in column VX is the kurtosis of the cells in rows 1-5 of column V2, and the value in column VY is the kurtosis of the cells in rows 1-5 of column V3.
TEMP = KURTOSIS(V1:V3, ROW)
Creates a new column named TEMP where each cell entry is the kurtosis of the corresponding row across columns V1, V2, and V3.
TEMP = KURTOSIS(V1[1:5]:V3],ROW) or TEMP = KURTOSIS(V1[1:5]:V3[1:5], ROW)
Creates a new column named TEMP, where the cells in rows 1-5 contain the kurtosis of the corresponding row across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
STAT	Computes the first through fourth moments of the specified data range
SKEW	Computes the skew of the distribution of a range of cells
STDV or STDEV	Computes the standard deviation of a range of cells
VARIANCE	Computes the variance of a range of cells

LAG

Syntax

LAG(lag, data)

Parameters

lag

The number of time steps to lag. This value must be a positive integer.

data

The values to lag. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see

the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

LAG returns values in the input data range, lagging by the specified number of time steps. It views each input column as a data series in time. and returns one new column for each input column. Each new column contains the time step-lagged values (lagging by lag number of time steps) of the numbers in the corresponding input column. The first lag values in the returned column(s) are zero. The length of the returned column(s) are the lengths of in the corresponding input column(s) + lag.

Note: The LAG macro function returns a column with values such that the cell VY[x] = data[x - lag] for x ? lag, else 0.

Examples

TEMP = LAG(1, COLUMN(1,2,3,4)) Creates a new column named TEMP containing the values 0, 1, 2, 3, and 4 in cells 1-5, respectively.
TEMP = LAG(2, V1) Creates a new column named TEMP, where each value is the contents of column V1 lagging by two time steps.
TEMP = LAG(10, V1:V3) Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the contents of column V1 lagging by ten time steps, the values of the VX column are the contents of column V2 lagging by ten time steps, and the values of the VY column are the contents of column V3 lagging by ten time steps.
TEMP = LAG(5, V1[10:20]) Creates a new column named TEMP, where the first 5 cells contain 0, followed by rows 10-20 of column V1. The other cells in TEMP are empty.
TEMP = LAG(2, V1[1:5]:V2) Creates two new columns named TEMP and VX, each with values in rows 1-7 (the other cells are empty). The values in rows 1-2 of each column are zeros. The remaining values in TEMP are the values in rows 1-5 of column V1. The remaining values in column VX are rows 1-5 of column V2.

Related Functions

Function	Description
DELAY	Returns the input columns values delayed by a specified number of time steps
SLIDE_WINDOW	Creates a pattern from a specified window and slides it down to create the next pattern

LE

Syntax

```
data1 LE data2 data1 <= data2
```

Parameters

data1

The numerical cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The number(s) to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

LE compares the two specified data ranges, returning a one if the values in the first data set are less than or equal to the values in the second data set or a zero otherwise. It returns a new column for each input column, each containing the corresponding column in data1 compared to the corresponding column of data2 (that is, the first column of data1 is compared to the first column of data, the second column with the second column, and so on).

If data2 is a constant, each value in data is compared to that value. If data2 is a column, the calculations are performed on a row-by-row basis. The values in data1 are compared to the first row value of data2, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The LE operator can be abbreviated with a less than sign followed by an equal sign (<=).

This macro is available in IBM Interact.

Examples

`TEMP = 4 LE 4 or TEMP = 4 <= 4`
Creates a new column named TEMP containing the value one (since four is equal to itself).
`TEMP = V1 <= 8`
Creates a new column named TEMP, where each value is one if the corresponding row value of the column V1 is less than or equal to the number eight, otherwise zero.

```
TEMP = V1:V3 <= 2
```

Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the contents of column V1 compared to the value two, the values of the VX column are the contents of column V2 compared to the value two, and the values of the VY column are the contents of column V3 compared to the value two.

```
TEMP = V1 <= V1
```

Creates a new column named TEMP containing all ones (since every number is equal to itself).

```
TEMP = V1 <= V2
```

Creates a new column named TEMP, where each value is the row value of column V1 compared to the corresponding row value of column V2.

```
TEMP = V1[10:20] <= V2 or TEMP = V1[10:20] <= V2[1:11]
```

Creates a new column named TEMP, where the first 11 cells contain the results of comparing the values in rows 10-20 of column V1 with the values in rows 1-11 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
EQ	Returns TRUE if one data range is equal to another
GE	Returns TRUE if one data range is greater than or equal to another
GT	Returns TRUE if one data range is greater than another
LT	Returns TRUE if one data range is less than another
NE	Returns TRUE if one data range is not equal to another

LIKE

Syntax

```
data1 [NOT] LIKE data2
```

Parameters

data1

The cell range to compare. This can be a text string or an expression evaluating to a text string. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The text pattern to compare all values in the specified column against. This can be a text string or an expression evaluating to a text string. The number of columns in data2 must equal the number of columns in data1, unless data2 is a constant. For

the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

An underscore (_) in `data2` represents a wildcard character that will match any single character in `data1`. A percent sign (%) will match zero or more characters in `data1`.

Description

`LIKE` compares the two specified data ranges, returning a one if the strings match or a zero if they do not. It returns a new column for each input column, each containing the corresponding column in `data1` compared to the corresponding column of `data2` (that is, the first column of `data1` is compared to the first column of `data2`, the second column with the second column, and so on).

If `data2` is a string constant, each string in `data1` is compared to that string. If `data2` is a column, the calculations are performed on a row-by-row basis. The first row string in `data1` is compared to the first row string of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last string in the shortest column.

When comparing strings, case does not matter (that is, "Yes", "YES", "yes", and "yeS" are all considered equal).

Note: The LIKE macro has a negative version, NOT LIKE. The format for this is identical to LIKE. NOT LIKE returns a one if the string in `data1` does not match the template defined by `data2`.

This macro is available in IBM Interact.

Examples

`TEMP = "gold" LIKE "gold"` Creates a new column named `TEMP` containing the value one (since the two strings match).
`TEMP = "No" LIKE "NO"` Creates a new column named `TEMP` containing the value one (string compares are case insensitive).
`TEMP = V1 LIKE "gold%"` Creates a new column named `TEMP`, where each value is one if the corresponding row value of the column `V1` is equal to the string "gold" followed by any number of characters. Otherwise, each value is zero.
`TEMP = V1 LIKE "g_ld"` Creates a new column named `TEMP`, where each value is one if the corresponding row value of the column `V1` is equal to the string "g" followed by any character, followed by "ld". Otherwise, each value is zero.
`TEMP = V1 LIKE V1` Creates a new column named `TEMP` containing all ones (since every number is equal to itself).
`TEMP = V1 LIKE V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` compared to the corresponding row value of column `V2`.

```
TEMP = V1:V3 LIKE V4:V6
```

Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the strings in V1 compared to the corresponding row strings of column V4. The column VX compares columns V2 and V5. The column VY compares columns V3 and V6.

```
TEMP = V1[10:20] LIKE V2 or TEMP = V1[10:20] LIKE V2[1:11]
```

Creates a new column named `TEMP`, where the first 11 cells contain the results of comparing the strings in rows 10-20 of column V1 to rows 1-11 of column V2. The other cells in `TEMP` are empty.

Related Functions

Function	Description
EQ	Returns TRUE if one data range is equal to another
GE	Returns TRUE if one data range is greater than or equal to another
GT	Returns TRUE if one data range is greater than another
LE	Returns TRUE if one data range is less than or equal to another
LT	Returns TRUE if one data range is less than another
NE	Returns TRUE if one data range is not equal to another

LN or LOG

Syntax

`LN(data) or LOG(data)`

Parameters

`data`

The numerical values to compute the natural logarithm of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

LN or `LOG` calculates the natural log of each value in the specified data range. It returns one new column for each input column, each containing the natural logarithm of numbers in the corresponding input column. Natural logarithms are based on the constant e = 2.7182818. LN is the inverse of the EXP macro function.

Note: All values in the specified data range must be greater than zero. Otherwise, a blank cell is returned for each invalid input.

Examples

`TEMP = LN(3)` or `TEMP = LOG(3)` Creates a new column named `TEMP` containing the value `1.099`.
`TEMP = LN(V1)` Creates a new column named `TEMP`, where each value is the natural log of the contents of column `V1`.
`TEMP = LN(V1:V3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the natural logs of the contents of column `V1`, the values in the `VX` column are the natural logs of the contents of column `V2`, and the values in the `VY` column are the natural logs of the contents of column `V3`.
`TEMP = LN(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the natural logs of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = LN(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the natural logs of the corresponding rows of column `V1`, and the values in column `VX` are the natural logs of the corresponding rows of column `V2`.

Related Functions

Function	Description
`EXP`	Computes the natural number (e) raised to the contents of each cell in the specified data range
`LOG2`	Computes the log base2 of the contents of the specified data range
`LOG10`	Computes the log base10 of the contents of the specified data range
`POW`	Computes a base value raised to the specified exponential power(s)

LOG2

Syntax

`LOG2(data)`

Parameters

`data`

The numerical values to compute the base2 logarithm of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

LOG2 calculates the base-2 logarithm of values in the specified data range. It returns one new column for each input column, each containing the base2 logarithm of numbers in the corresponding input column.

Note: All values in the specified data range must be greater than zero. Otherwise, a blank cell is returned for each invalid input.

Examples

`TEMP = LOG2(8)` Creates a new column named `TEMP` containing the value three.
`TEMP = LOG2(V1)` Creates a new column named `TEMP`, where each value is the base2 log of the contents of column `V1`.
`TEMP = LOG2(V1:V3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the base2 logs of the contents of column `V1`, the values of the `VX` column are the base2 logs of the contents of column `V2`, and the values of the `VY` column are the base2 logs of the contents of column `V3`.
`TEMP = LOG2(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the base-2 logs of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = LOG2(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the base-2 logs of the corresponding rows of column `V1`, and the values in column `VX` are the base-2 logs of the corresponding rows of column `V2`.

Related Functions

Function	Description
`LN` or `LOG`	Computes the natural log of the contents of the specified data range
`LOG10`	Computes the log base10 of the contents of the specified data range
`POW`	Exponential power

LOG10

Syntax

`LOG10(data)`

Parameters

`data`

The numerical values to compute the base10 logarithm of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`LOG10` calculates the base-10 logarithm of the values in the specified data range. It returns one new column for each input column, each containing the base10 logarithm of numbers in the corresponding input column.

Note: All values in the specified data range must be greater than zero. Otherwise, a blank cell is returned for each invalid input.

Examples

`TEMP = LOG10(100)` Creates a new column named `TEMP` containing the value two.
`TEMP = LOG10(V1)` Creates a new column named `TEMP`, where each value is the base10 log of the contents of column `V1`.
`TEMP = LOG10(V1:V3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the base10 logs of the contents of column `V1`, the values of the `VX` column are the base10 logs of the contents of he column `V2`, and the values of the `VY` column are the base10 logs of the contents of column `V3`.
`TEMP = LOG10(V1[10:20])` Creates a new column named `TEMP`, where the first 11 cells contain the base-10 logs of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = LOG10(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the base-10 logs of the corresponding rows of column `V1`, and the values in column `VX` are the base-10 logs of the corresponding rows of column `V2`.

Related Functions

Function	Description
`LN` or `LOG`	Computes the natural log of the contents of the specified data range
`LOG2`	Computes the log base2 of the contents of the specified data range
`POW`	Exponential power

LOWER

Syntax

`LOWER(data)`

Parameters

data

The string value to be converted to lowercase.

Description

LOWER converts each string value in the specified data range to lowercase. It returns a new column with each cell containg the lowercased string of the corresponding input cell.

This macro is available in IBM Interact.

Examples

`Temp = LOWER "GOLD"`
Creates a new column named Temp containing "gold".
`TEMP = LOWER( "JAN 15, 1997")`
Creates a new column named TEMP, which contains the ASCII text string " `jan 15, 1997` ".
`TEMP = LOWER( "Pressure")`
Creates a new column named TEMP, which contains the ASCII text string "`pressure`".
`TEMP = LOWER(V1)`
Creates a new column named TEMP containing lowercase characters of each string in column `V1`.

LT

Syntax

`data1 LT data2 data1 < data2`

Parameters

data1

The numerical cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The number(s) to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

LT compares the two specified data ranges, returning a one if the values in the first data set are less than the values in the second data set or a zero otherwise. It

returns a new column for each input column, each containing the corresponding column in `data1` compared to the corresponding column of `data2` (that is, the first column of `data1` is compared to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data` is compared to that value. If `data2` is a column, the calculations are performed on a row-by-row basis. The values in `data1` are compared to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The `LT` operator can be abbreviated with a less than sign (<).

This macro is available in IBM Interact.

Examples

`TEMP = 3 LT 4 or TEMP = 3 < 4` Creates a new column named `TEMP` containing the value one (since three is less than four).
`TEMP = V1 < 8` Creates a new column named `TEMP`, where each value is one if the corresponding row value of the column `V1` is less than the number eight, otherwise zero.
`TEMP = V1:V3 < 2` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the contents of column `V1` compared to the value two, the values of the `VX` column are the contents of column `V2` compared to the value two, and the values of the `VY` column are the contents of column `V3` compared to the value two.
`TEMP = V1 < V1` Creates a new column named `TEMP` containing all zeros (since no number is less than itself).
`TEMP = V1 < V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` compared to the corresponding row value of column `V2`.
`TEMP = V1[10:20] < V2 or TEMP = V1[10:20] < V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the results of comparing the values in rows 10-20 of column `V1` to rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
EQ	Returns TRUE if one data range is equal to another
GE	Returns TRUE if one data range is greater than or equal to another
GT	Returns TRUE if one data range is greater than another
LE	Returns TRUE if one data range is less than or equal to another

LTRIM

Syntax

LTRIM(data)

Parameters

data

The string from which the leading space will be removed.

Description

LTRIM removes leading space characters from each string value in the specified data range, returning the converted string. It returns one new column for each input column.

This macro is available in IBM Interact.

Examples

Temp = LTRIM " gold"

Creates a new string named Temp which contains "gold".

MAX

Syntax

MAX(data [, keyword])

Parameters

data

The numerical values to compute the maximum of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {ALL | COL | ROW}. These keywords do not apply in IBM Campaign because the input data is always a single column or field. The macro will always behave as if the COL keyword were specified. Therefore, you do not need to specify these keywords when using IBM Campaign.

Description

MAX calculates the maximum of the values in the specified data range. It returns a single new column containing the maximum value.

This macro is available in IBM Interact.

Examples

TEMP = MAX(3) or TEMP = MAX(3, ALL) Creates a new column named TEMP containing the value three.
TEMP = MAX(SELECT(COLUMN(1,3,5), V1:V5)) Creates a new column named TEMP containing a single value which is the maximum of columns V1, V3, and V5.
TEMP = MAX(V1) Creates a new column named TEMP containing a single value which is the maximum value of the contents of column V1.
TEMP = MAX(V1:V3) Creates a new column named TEMP containing a single value which is the maximum of the columns V1, V2, and V3.
TEMP = MAX(V1[10:20]) Creates a new column named TEMP containing a single value which is the maximum of the cells in rows 10-20 of column V1.
TEMP = MAX(V1[1:5]:V4) Creates a new column named TEMP containing a single value which is the maximum of the cells in rows 1-5 of columns V1 through V4.
TEMP = MAX(V1:V3, COL) Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the maximum of the contents of column V1, the single value in the VX column is the maximum of the contents of column V2, and the single value in the VY column is the maximum of the contents of column V3.
TEMP = MAX(V1[1:5]:V3, COL) Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the maximum of the cells in rows 1-5 of column V1, the value in column VX is the maximum of the cells in rows 1-5 of column V2, and the value in column VY is the maximum of the cells in rows 1-5 of column V3.
TEMP = MAX(V1:V3, ROW) Creates a new column named TEMP where each cell entry is the maximum of the corresponding row across columns V1, V2, and V3.

```
TEMP = MAX(V1[10:20]:V3, ROW)
```

Creates a new column named TEMP, where the first 11 cells contain the maximum of the values in rows 10-20 across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
DECIMATE	Decimates a column of numbers into multiple columns where a "one" indicates the index value.
MAXINDEX	Returns the column index of the nth (first, second, third, etc.) maximum value for each row of the specified column
MIN	Computes the minimum of a range of cells

MAXINDEX

Syntax

```
MAXINDEX(data [, n])
```

Parameters

data

The start of a data range for which to compute the index of the nth maximum value for each row. This can be a column, or an expression evaluating to a column. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

n

Number, greater than zero. The default value is one, which returns the index of the highest value for each row. (Two returns the index for the second highest value, three the index for the third highest value, etc.)

Description

MAXINDEX finds the nth maximum value in each row of the specified data range and returns a column index for its location. It returns a new column containing a single value representing the index of the nth maximum value for each row. A value of one indicates the first cell in the first column. If there is more than one maximum value, the first column containing the nth maximum is returned.

Note: If you have multiple columns, each one representing a separate output class, you can use MAXINDEX to select the "winning class." You can then train with a single output column rather than multiple output columns. For example, V4 = MAXINDEX(V1:V3) would do the following:

```
V1 V2 V3 V4 0 1 0 2 1 0 0 1 0 0 1 3
```

Examples

`TEMP = MAXINDEX(MERGE(3,5,-2))` Creates a new column named `TEMP` containing the value two (since the maximum value is five, which occurs in the second column).
`TEMP = MAXINDEX(V1)` Creates a new column named `TEMP` containing a one for each row of column `V1`.
`TEMP=MAXINDEX (V6:V8,3)` Creates a new column named TEMP with each value representing the index of the minimum value (in this case 3rd highest of 3) of the corresponding row across the columns V6, V7 and V8. A one is retuned if the minimum is in column V6, a two if the minimum is in column V7, and a three is the minimum is in column V8
`TEMP = MAXINDEX(V6:V8)` Creates a new column named `TEMP` with each value representing the index of the maximum value of the corresponding row across the columns `V6`, `V7`, and `V8`. A one is returned if the maximum is in column `V6`, a two if the maximum is in column `V7`, and a three if the maximum is in column `V8`.
`TEMP = MAXINDEX(V1[1:5]:V3]` Creates a new column named `TEMP`, where the cells in rows 1-5 contain a value representing the index of the maximum value of the corresponding row across columns `V1` through `V3`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`DECIMATE`	Decimates a column of numbers into multiple columns where a one indicates the index value
`MAX`	Computes the maximum of a range of cells
`MIN`	Computes the minimum of a range of cells

MEAN

Syntax

`MEAN(data [, keyword])`

Parameters

`data`

The numerical values to compute the arithmetic mean of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in `data` (default)

COL - Performs the computation separately for each column of `data`

ROW - Performs the computation separately for each row of `data`

See "DATE" on page 73 for more details on using keywords.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`MEAN` calculates the arithmetic mean or average of the cells in the specified data range. The arithmetic mean is calculated by summing the contents of all cells divided by the number of cells. The number of columns returned by `MEAN` depends on `keyword`.

- If `keyword` is `ALL`, `MEAN` returns one new column, containing a single value (the average of all cells in `data`).
- If `keyword` is `COL`, `MEAN` returns a new column for each input column. Each new column contains one value (the average of all cells in the corresponding input column).
- If `keyword` is `ROW`, `MEAN` returns one new column containing the average across each row of `data`.

Note: Blank cells are ignored in the mean.

Note: `MEAN` is the same as the `AVG` macro function.

This macro is available in IBM Interact.

Examples

`TEMP = MEAN(MERGE(3, 4, 5))` or `TEMP = MEAN(MERGE(3, 4, 5), ALL)`   Creates a new column named `TEMP` containing the value 4.
`TEMP = MEAN(MERGE(-10, 6, 10))`   Creates a new column named `TEMP` containing the value 2.
`TEMP = MEAN(V1)`   Creates a new column named `TEMP` containing a single value which is the arithmetic mean of the contents of column `V1`.
`TEMP = MEAN(V1:V3)`   Creates a new column named `TEMP` containing a single value which is the arithmetic mean of the contents of columns `V1`, `V2`, and `V3`.
`TEMP = MEAN(V1[10:20])`   Creates a new column named `TEMP` containing a single value which is the arithmetic mean of the cells in rows 10-20 of column `V1`.

```
TEMP = MEAN(V1[1:5]:V4)
```

Creates a new column named TEMP containing a single value which is the arithmetic mean of the cells in rows 1-5 of columns V1 through V4.

```
TEMP = MEAN(V1:V3, COL)
```

Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the arithmetic mean of the contents of column V1, the single value in the VX column is the arithmetic mean of the contents of column V2, and the single value in the VY column is the arithmetic mean of the contents of column V3.

```
TEMP = MEAN(MERGE(1,4),COL)
```

Creates two new columns named TEMP and VX. TEMP contains a single value of one; VX contains a single value of four.

```
TEMP = MEAN(V1[10:20]:V3, COL)
```

Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the arithmetic mean of the cells in rows 10-20 of column V1, the value in column VX is the arithmetic mean of the cells in rows 10-20 of column V2, and the value in column VY is the arithmetic mean of the cells in rows 10-20 of column V3.

```
TEMP = MEAN(V1:V3, ROW)
```

Creates a new column named TEMP where each cell entry is the arithmetic mean of the corresponding row across columns V1, V2, and V3.

```
TEMP = MEAN(V1[1:5]:V3,ROW)
```

Creates a new column named TEMP, where the cells in rows 1-5 contain the arithmetic mean of the corresponding row across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
AVG_DEV	Computes the average deviation of a range of cells
SUM or TOTAL	Computes the sum of a range of cells

MERGE

Syntax

```
MERGE(data [, data]...) {data [, data]...}
```

Parameters

```
data
```

The name of a column to combine into a data range. This can be a constant value (numeric or ASCII text in quotes), a column, a cell range, or an expression evaluating to any of the above. This parameter can be repeated multiple times. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

MERGE horizontally concatenates its input into a new group. It returns as many new columns as provided as input. An unlimited number of arguments can be provided.

Note: The MERGE macro function can be specified using braces ({}). Simply insert the arguments within braces separated by commas (for example, TEMP = {1,2,3} is equivalent to TEMP = MERGE(1,2,3)).

Examples

`TEMP = MERGE(3, 4, "five")` or `TEMP = {3, 4, "five"}`   Creates three new columns named TEMP, VX, and VY, containing the values 3, 4, and "five " respectively.
`TEMP = V1:V3` or `TEMP = MERGE(V1:V3)`   Creates three new columns named TEMP, VX, and VY, where TEMP is a copy of the column V1, VX is a copy of the column V2, and VY is a copy of the column V3.
`TEMP = MERGE(V1, V3, V5:V7)`   Creates five new columns named TEMP, VW, VX, VY, and VZ. TEMP is a copy of column V1; VW is a copy of column V3; VX through VZ are copies of columns V5 through V7.
`TEMP = AVG(MERGE(V1,V3,V5), ROW)`   Creates one new column named TEMP, where each row is the average of the corresponding row across columns V1, V3, and V5. An average is only computed up to the last row of the shortest input column.
`TEMP = MERGE(V1[10:50],V3, V5:V7[1:30])`   Creates five new columns named TEMP, VW, VX, VY, and VZ. TEMP is a copy of the values in rows 10-50 of column V1; VW is a copy of the values column V3; VX through VZ are copies of the values in rows 1-30 of columns V5 through V7.
`TEMP = AVG(MERGE(V1, V5:V6))` or `TEMP = AVG({V1, V5:V6})`   Creates one new column named TEMP containing the average of all cells in columns V1, V5, and V6.

Related Functions

Function	Description
COLUMN	Creates new column(s), vertically concatenating the input values in each column
SELECT	Returns the specified column(s) from a data range
TRANSPOSE	Transposes a specified data range

MIN

Syntax

`MIN(data [, keyword])`

Parameters

`data`

The numerical values to compute the minimum of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

See "DATE" on page 73 for more details on using keywords.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`MIN` calculates the minimum of all the cells in the specified data range. It returns a single column containing the minimum value.

This macro is available in IBM Interact.

Examples

`TEMP = MIN(MERGE(1,10,-2))`Creates a new column named TEMP containing the value `-2`.
`TEMP = MIN(V1)` Creates a new column named TEMP containing a single value which is the minimum value of column `V1`.
`TEMP = MIN(V1:V3)` Creates a new column named TEMP containing a single value which is the minimum of columns `V1`, `V2`, and `V3`.
`TEMP = MIN(V1[10:20])` Creates a new column named TEMP containing a single value which is the minimum of the cells in rows 10-20 of column `V1`.
`TEMP = MIN(V1[1:5]:V4)` Creates a new column named TEMP containing a single value which is the minimum of the cells in rows 1-5 of columns `V1` through `V4`.

`TEMP = MIN(V1:V3, COL)`
Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the minimum of column V1, the single value in the VX column is the minimum of column V2, and the single value in the VY column is the minimum of column V3.
`TEMP = MIN(V1[1:5]:V3, COL)`
Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the minimum of the cells in rows 1-5 of column V1, the value in column VX is the minimum of the cells in rows 1-5 of column V2, and the value in column VY is the minimum of the cells in rows 1-5 of column V3.
`TEMP = MIN(V1:V3, ROW)`
Creates a new columns named TEMP where each cell entry is the minimum of the corresponding row across columns V1, V2, and V3.
`TEMP = MIN(V1[10:20]:V3, ROW)`
Creates a new column named TEMP, where the first 11 cells contain the minimum of the values in rows 1-5 across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
DECIMATE	Decimates a column of numbers into multiple columns where a one indicates the index value
MAX	Computes the maximum of a range of cells
MAX_TO_INDEX	Returns the column index of the maximum value for each row of the specified column

MINUS

Syntax

`data MINUS subtrahend data - subtrahend`

Parameters

`data`

The cell range containing numbers to subtract from. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`subtrahend`

The number(s) to subtract from all values in the specified column. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `subtrahend` must equal the number of columns in `data`, unless `subtrahend` is a constant. For the format definition of `subtrahend` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

MINUS subtracts `subtrahend` from the specified data range `data`. It returns a new column for each input column, each containing the corresponding column in `data` minus the corresponding column of `subtrahend` (that is, the first column of `data` subtracts the first column of `subtrahend`, the second column with the second column, and so on).

If `subtrahend` is a constant, each value in `data` is subtracts that value. If `subtrahend` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data` and one column from `subtrahend`. The first row of `data` subtracts the first row value of `subtrahend`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The MINUS operator can be abbreviated with a minus sign or hyphen (`-`).

This macro is available in IBM Interact.

Examples

`TEMP = 7 MINUS 4 or TEMP = 7 - 4` Creates a new column named `TEMP` containing the value three.
`TEMP = V1 - 8` Creates a new column named `TEMP`, where each value is the contents of column `V1` minus eight.
`TEMP = V1:V3 - 2` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the contents of column `V1` minus two, the values of the `VX` column are the contents of column `V2` minus two, and the values of the `VY` column are the contents of column `V3` minus two.
`TEMP = V1 - V1` Creates a new column named `TEMP` containing all zeros (since any number minus itself is zero).
`TEMP = V1 - V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` minus the corresponding row value of column `V2`.
`TEMP = V1:V3 -V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` minus the corresponding row values of column `V4`. The column `VX` subtracts column `V5` from `V2`. The column `VY` subtracts column `V6` from `V3`.
`TEMP = V1[10:20] - V2 or TEMP = V1[10:20] - V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the values in rows 10-20 of column `V1` minus the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
PLUS	Adds the contents of two data ranges

Function	Description
SUM or TOTAL	Computes the sum of a range of cells

MOD

Syntax

`data MOD divisor data % divisor`

Parameters

`data`

The integer values to compute the modulo of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`divisor`

The non-zero base integer to compute the modulo in respect to. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `divisor` must equal the number of columns in `data`, unless `divisor` is a constant. For the format definition of `divisor` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`MOD` calculates the remainder of dividing the specified data range by a specified value. This is computed by dividing `divisor` into each value and returning the remainder. It returns one new column for each input column, each containing the numbers in `data` modulo `divisor`. The remainder will have the same sign (positive or negative) as `data`.

If `divisor` is a constant, each value in the specified column is calculated modulo that value. If `divisor` is a column, the calculations are performed on a row-by-row basis. The values in `data` are calculated modulo the first row value of `divisor`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: If `divisor` is zero, a divide by zero error is returned.

Note: The `MOD` operator can be abbreviated with the percent sign (`%`). For example, `TEMP = 5 % 3` is equivalent to `TEMP = 5 MOD 3`.

This macro is available in IBM Interact.

Examples

`TEMP = 10 MOD 8 or TEMP = 10 % 8`

Creates a new column named `TEMP` containing the value 2.

`TEMP = -10 % 8`
Creates a new column named `TEMP` containing the value `-2`.
`TEMP = V1 % 8`
Creates a new column named `TEMP`, where each value is the contents of column `V1`, modulo eight.
`TEMP = V1:V3 % 2`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the values modulo two of the contents of column `V1`, the values of the `VX` column are the values modulo two of the contents of column `V2`, and the values of the `VY` column are the values modulo two of the contents of column `V3`.
`TEMP = V1 % V1`
Creates a new column named `TEMP`, containing a zero for each entry in the column `V1`. This is because every number modulo itself is zero.
`TEMP = V1 % V2`
Creates a new column named `TEMP`, where each value is the row value of column `V1` modulo the corresponding row value of column `V2`. Note that if V2=V1, then all zeros are returned, as in the previous example.
`TEMP = V1:V3 % V4:V6`
Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in V1 modulo the corresponding row values of column V4. The column `VX` contains the results of column V2 modulo V5. The column `VY` contains the results of column V3 modulo V6.
`TEMP = V1[10:20] % V2 or TEMP = V1[10:20] % V2[1:11]`
Creates a new column named `TEMP`, where the first 11 cells are the values of the values in rows 10-20 of column V1 modulo the values in rows 1-11 of column V2. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`DIV`	Divides one specified data range by another
`MOD`	Computes the modulo of the contents of the specified data range

---

MONTHOF

Syntax

`MONTHOF(date_string [, input_format])`

Parameters

`date_string`

A text representing a valid date.

`input_format`

One of the keywords in the table below, specifying the date format of `date_string`.

Description

`MONTHOF` returns the month as a number for the date specified by the `date_string`. If `input_format` is not provided, the default keyword `DELIM_M_D_Y` will be used.

Examples

`MONTHOF("012171",MMDDYY)` returns the number 1.

Note: See "DATE" on page 73 for additional information on valid date formats.

Related Functions

Function	Description
DAYOF	Returns the day of the week as a number.
WEEKDAYOF	Returns the weekday of the week as a number.
YEAROF	Returns the year as a number.

MULT

Syntax

`data MULT multiplier data * multiplier`

Parameters

`data`

The numerical values to multiply. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`multiplier`

The number to multiply all values in the specified column by. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `multiplier` must equal the number of columns in `data`, unless `multiplier` is a constant. For the format definition of `multiplier` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`MULT` multiplies the values in the two specified data ranges. It returns one new column for each input column, each containing the numbers in `data` multiplied by `multiplier`. If `multiplier` is a constant, each value in `data` is multiplied by that value. If `multiplier` is a column, the calculations are performed on a row-by-row basis. The values in `data` are multiplied by the first row value of `multiplier`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Using a column containing the same number x in each row as `multiplier` is the same as using the constant x as `multiplier`.

Note: The `MULT` operator can be abbreviated with an asterisk (`*`).

This macro is available in IBM Interact.

Examples

`TEMP = 8 MULT 4 or TEMP = 8 * 4` Creates a new column named TEMP containing the value 32.
`TEMP = V1 * 8` Creates a new column named TEMP, where each value is the contents of column V1 multiplied by eight.
`TEMP = V1:V3 * 2` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are two times the contents of column V1, the values of the VX column are two times the contents of column V2, and the values of the VY column are two times the contents of column V3.
`TEMP = V1 * V1` Creates a new column named TEMP containing the square of each value in column V1.
`TEMP = V1 * V2` Creates a new column named TEMP, where each value is the row value of column V1 multiplied by the corresponding row value of column V2.
`TEMP = V1:V3 * V4:V6` Creates three new columns named TEMP, VX, and VY. The column TEMP contains the values in V1 times the corresponding row values of column V4. The column VX multiplies column V2 by V5. The column VY multiplies column V3 by V6.
`TEMP = V1[10:20] * V2 or TEMP = V1[10:20] * V2[1:11]` Creates a new column named TEMP, where the first 11 cells contain the values in rows 10-20 of column V1 times the values in rows 1-11 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
`DIV`	Divides one specified data range by another
`EXP`	Computes the natural number (e) raised to the contents of each cell in the specified data range
`POW`	Computes a base value raised to the specified exponential power(s)

NE

Syntax

`data1 NE data2 data1 != data2 data1 <> data2`

Parameters

data1

The cell range to compare. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The number(s) to compare all values in the specified column against. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in data2 must equal the number of columns in data1, unless data 2 is a constant. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

NE compares the two specified data ranges, returning a one if the values are not equal or a zero if they are equal. It returns a new column for each input column, each containing the corresponding column in data1 compared to the corresponding column of data2 (that is, the first column of data1 is compared to the first column of data, the second column with the second column, and so on).

If data2 is a constant, each value in data1 is compared to that value. If data2 is a column, the calculations are performed on a row-by-row basis. The values in the first row of data1 are compared to the first row value of data2, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Using a column containing the same number x in each row as data2 is the same as using the constant x as data2.

Note: The NE operator can be abbreviated with an exclamation point followed by an equal sign (!=) or by a less-than sign followed by a greater-than sign (<>).

This macro is available in IBM Interact.

Examples

`TEMP = 3 NE 4 or TEMP = 3 != 4 TEMP = 3 <> 4`   Creates a new column named TEMP containing the value one (since three is not equal to four).
`TEMP = V1 != 8`   Creates a new column named TEMP, where each value is one if the corresponding row value of the column V1 is not equal to the number eight, otherwise zero.
`TEMP = V1:V3 != 2`   Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the contents of column V1 compared to the value two, the values of the VX column are the contents of column V2 compared to the value two, and the values of the VY column are the contents of column V3 compared to the value two.

`TEMP = V1 != V1`
Creates a new column named `TEMP` containing all zeros (since every number is equal to itself).
`TEMP = V1 != V2`
Creates a new column named `TEMP`, where each value is the row value of column `V1` compared to the corresponding row value of column `V2`.
`TEMP = V1:V3 != V4:V6`
Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` compared to the corresponding row values of column `V4`. The column `VX` compares columns `V2` and `V5`. The column `VY` compares columns `V3` and `V6`.
`TEMP = V1[10:20] != V2` or `TEMP = V1[10:20] != V2[1:11]`
Creates a new column named `TEMP`, where the first 11 cells contain the results of comparing the values in rows 10-20 of column `V1` and rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`EQ`	Returns TRUE if one data range is equal to another
`GE`	Returns TRUE if one data range is greater than or equal to another
`GT`	Returns TRUE if one data range is greater than another
`LE`	Returns TRUE if one data range is less than or equal to another
`LT`	Returns TRUE if one data range is less than another

NORM_MINMAX

Syntax

```
NORM_MINMAX(data [, keyword]) NORM_MINMAX(data, min, max [, keyword])
NORM_MINMAX(data, base_data [, keyword])
```

Parameters

`data`

The numerical values to normalize. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`min, max`

These two parameters provide a minimum and a maximum value to use for normalization. They must be constants, except with the ROW keyword, where they can be constants or columns.

base_data

This parameter specifies a data range to use for computing the maximum and minimum to use for normalization. The number of columns provided in base_data must be the same number of columns as provided for data. For the format definition of base_data (same as data), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

NORM_MINMAX calculates the normalized values of the specified data range. All returned values will be between zero and one, inclusive. The min/max normalization is performed as follows:

$$VX[y] = \frac{VX[y] - min}{max - min}$$

where *min* and *max* are determined as follows:
• If min and max are provided, these values are used for the minimum and maximum, respectively. If these parameters are provided with the ROW keyword, min and max can be columns, specifying a min and max value for each row of data. If min and max are columns, the columns must be either the same length as data or scalar (that is, contain a single value which is used as a constant applied to all values in the corresponding column of data).
• If base_data is provided, the minimum and maximum of this data range are calculated and used to normalize data. The columns in base_data must contain two or more cell values.
• If neither of the above mutually exclusive options are provided, the minimum and maximum values are automatically computed from data.

Note: Since all returned values are between 0.0 and 1.0, any values computed using Equation (above) less than 0.0 are returned as 0.0. Similarly, any values greater than 1.0 are returned as 1.0. `NORM_MINMAX` always returns a data range with the same dimensions as the input data range. The `ALL` keyword specifies to compute the mean and standard deviation over the entire input data range. The `COL` keyword specifies to compute a mean and standard deviation for each input column and to use those values for normalizing that column. The `ROW` keyword specifies to compute a mean and standard deviation for each row in the specified data range and to use those values for normalizing that row.

Note: If the minimum and maximum are equal, then all zeros are returned.

Examples

`TEMP = NORM_MINMAX(3)`   Creates a new column named `TEMP` containing the value zero.
`TEMP = NORM_MINMAX(COLUMN(3, 4, 5))`   Creates a new column named `TEMP` containing the values `0`, `0.5`, and `1`. (The minimum and maximum [3 and 5] are calculated from the data range automatically.)
`TEMP = NORM_MINMAX(COLUMN(3, 4, 5), 0, 10)`   Creates a new column named `TEMP` containing the values `0.3`, `0.4`, and `0.5`. (This time the minimum and maximum [0 and 10] are provided as arguments.)
`TEMP = NORM_MINMAX(V1)`   Creates a new column named `TEMP` containing normalized values of the contents of column `V1`. The minimum and maximum values used for normalization are calculated over the column `V1`.
`TEMP = NORM_MINMAX(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated over columns `V1`, `V2`, and `V3`.
`TEMP = NORM_MINMAX(V1[1:5]:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`, each with values in rows 1-5. The contents of column `TEMP` are the normalized values of the corresponding rows in column `V1`, the contents of column `VX` are the normalized values of the corresponding rows of column `V2`, and the contents of column `VY` are the normalized values of the corresponding rows of column `V3`. The minimum and maximum for normalization purposes are calculated over rows 1-5 of columns `V1-V3`.
`TEMP = NORM_MINMAX(V1:V3, V4:V6)`   Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated over columns `V4`, `V5`, and `V6`.
`TEMP = NORM_MINMAX(V1:V3, COL)`   Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated for each column independently (that is, a min/max is calculated for column `V1`, a separate min/max is calculated for column `V2`, etc.).

`TEMP = NORM_MINMAX(V1[10:50]:V3, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`, each with values in the first 41 rows. The contents of column `TEMP` are the normalized values of rows 10-50 of column `V1`, the contents of column `VX` are the normalized values of rows 10-50 of column `V2`, and the contents of column `VY` are the normalized values of rows 10-50 of column `V2`. The minimum and maximum for normalization purposes are calculated independently across rows 10-50 for each column.
`TEMP = NORM_MINMAX(V1:V3, V4:V6, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated for each column independently using columns `V4-V6` (that is, a min/max is calculated over column `V4` for normalizing column `V1`, a separate min/max is calculated over column `V5` for normalizing column `V2`, etc.).
`TEMP = NORM_MINMAX(V1:V3, ROW)` Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated over independently over each row across columns `V1`, `V2`, and `V3`.
`TEMP = NORM_MINMAX(V1[10:20]:V3, ROW)` Creates three new columns named `TEMP`, `VX`, and `VY`, each with values in the first 11 rows. The contents of column `TEMP` are the normalized values of rows 10-20 in column `V1`, the contents of column `VX` are the normalized values of rows 10-20 of column `V2`, and the contents of column `VY` are the normalized values of rows 10-20 of column `V3`. The minimum and maximum for normalization purposes are calculated over each of the rows 10-20 of columns `V1-V3`.
`TEMP = NORM_MINMAX(V1:V3, V8:V10, ROW)` Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The minimum and maximum values used for normalization are calculated independently for each row across columns `V8-V10`.

Related Functions

Function	Description
`NORM_SIGMOID`	Computes the sigmoidal normalization of a data range
`NORM_ZSCORE`	Computes the z-score normalization of a data range

NORM_SIGMOID

Syntax

`NORM_SIGMOID(data [, keyword]) NORM_SIGMOID(data, mean, std [, keyword])`
`NORM_SIGMOID(data, base_data [, keyword])`

Parameters

`data`

The values to normalize. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`mean, std`

These two parameters provide the mean and the standard deviation to use for normalization. They must be constants, except with the `ROW` keyword, where they can be constants or columns.

`base_data`

This parameter specifies a data range to use for computing the mean and standard deviation to use for normalizing `data`.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`NORM_SIGMOID` calculates the normalized values of the specified data range. A sigmoidal normalization redistributes data along a sigmoid curve, returning values between `-1.0` and `+1.0`, inclusive. Essentially, all data within a standard deviation of the mean is linearly distributed in the middle range of the sigmoid. Outliers are represented along the tails of the sigmoid. This allows you to keep very large outlier data points without sacrificing discrimination ability among points near the mean.

The sigmoidal normalization is performed as follows:

$$VX[y] = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}$$

where

$$\alpha = \frac{VX[y] - mean}{std}$$

and *mean* and *std* are determined as follows:

- If `mean` and `std` are provided, these values are used for the mean and standard deviation, respectively. If these parameters are provided with the `ROW` keyword, `mean` and `std` can be columns, specifying a mean and standard deviation for each row of `data`. If `min` and `max` are columns, the columns must be either the same length as `data` or scalar (that is, contain a single value which is used as a constant applied to all values in the corresponding column of `data`).
- If `base_data` is provided, the mean and standard deviation of this data range are calculated and used to normalize `data`. The columns in `base_data` must contain two or more cell values.
- If neither of the above mutually exclusive options are provided, the mean and standard deviation are automatically computed from `data`.

`NORM_SIGMOID` always returns a data range with the same dimensions as the input data range. The `ALL` keyword specifies to compute the mean and standard deviation over the entire input data range. The `COL` keyword specifies to compute a mean and standard deviation for each input column and to use those values for normalizing that column. The `ROW` keyword specifies to compute a mean and standard deviation for each row in the specified data range and to use those values for normalizing that row.

Note: If the standard deviation is zero, all zeros are returned.

Note: To normalize data using the same `base_data` range (for example, in wrapped user functions), make `mean` and `std` constants (this can be done using the `CONSTANT` macro function).

Examples

`TEMP = NORM_SIGMOID(COLUMN(3, 4, 5))`   Creates a new column named `TEMP` containing the values `-0.55`, `0`, and `0.55`. (The mean and standard deviation [4 and 0.816] are calculated from the data range automatically.)
`TEMP = NORM_SIGMOID(COLUMN(3, 4, 5), 3.5, 1.2)`   Creates a new column named `TEMP` containing the values `-0.21`, `0.21`, and `0.55`. (This time the mean and standard deviation [3.5 and 1.2] are provided as arguments.)
`TEMP = NORM_SIGMOID(V1)` or `TEMP = NORM_SIGMOID(V1,ALL)`   Creates a new column named `TEMP` containing normalized values of the contents of column `V1`. The mean and standard deviation used for normalization are calculated over column `V1`.
`TEMP = NORM_SIGMOID(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated over columns `V1`, `V2`, and `V3`.

```
TEMP = NORM_SIGMOID(V1[10:50]:V3)
```

Creates three new columns named TEMP, VX, and VY, each with values in rows 1-41. The contents of column TEMP are the normalized values of rows 10-50 of column V1, the contents of column VX are the normalized values of rows 10-50 of column V2, and the contents of column VY are the normalized values of rows 10-50 of column V3. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns V1-V3.

```
TEMP = NORM_SIGMOID(V1:V3, V4)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated over column V4.

```
TEMP = NORM_SIGMOID(V1:V3, V4:V8)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated over columns V4-V8.

```
TEMP = NORM_SIGMOID(V1:V3, COL)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated for each column independently (that is, a mean and standard deviation is calculated for column V1, a separate mean and standard deviation is calculated for column V2, etc.).

```
TEMP = NORM_SIGMOID(V1[10:50]:V3, COL)
```

Creates three new columns named TEMP, VX, and VY, each with values in rows 1-41. The contents of column TEMP are the normalized values of rows 10-50 of column V1, the contents of column VX are the normalized values of rows 10-50 of column V2, and the contents of column VY are the normalized values of rows 10-50 of column V3. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns V1-V3. The mean and standard deviation for normalization purposes are calculated independently for each column.

```
TEMP = NORM_SIGMOID(V1:V3, V4:V6, COL)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated for each column independently using columns V4-V6 (that is, a mean and standard deviation is calculated over column V4 for normalizing column V1, a separate mean and standard deviation is calculated over column V5 for normalizing column V2, etc.).

```
TEMP = NORM_SIGMOID(V1:V3, ROW)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated over independently over each row across columns V1, V2, and V3.

```
TEMP = NORM_SIGMOID(V1[10:50]:V3, ROW)
```

Creates three new columns named TEMP, VX, and VY, each with values in rows 1-41. The contents of column TEMP are the normalized values of rows 10-50 of column V1, the contents of column VX are the normalized values of rows 10-50 of column V2, and the contents of column VY are the normalized values of rows 10-50 of column V3. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns V1-V3. The mean and standard deviation for normalization purposes are calculated over each row of columns V1-V3.

```
TEMP = NORM_SIGMOID(V1:V3, V4:V10, ROW)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated independently for each row across columns V4-V10.

Related Functions

Function	Description
NORM_MINMAX	Computes the min/max normalization of a data range
NORM_ZSCORE	Computes the z-score normalization of a data range

NORM_ZSCORE

Syntax

```
NORM_ZSCORE(data [, keyword]) NORM_ZSCORE(data, mean, std [, keyword])
NORM_ZSCORE(data, base_data [, keyword])
```

Parameters

data

The numerical values to normalize. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

mean, std

These two parameters provide the mean and the standard deviation to use for normalization. They must be constants, except with the ROW keyword, where they can be constants or columns.

base_data

This parameter specifies a data range to use for computing the mean and standard deviation to use for normalizing data.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

NORM_ZSCORE calculates the normalized values of the specified data range. The z-score normalization is performed as follows:

$$VX[y] = \frac{VX[y] - mean}{std}$$

where *mean* and *std* are determined as follows:

- If mean and std are provided, these values are used for the mean and standard deviation, respectively. If these parameters are provided with the ROW keyword, mean and std can be columns, specifying a mean and standard deviation for each row of data. If min and max are columns, the columns must be either the same length as data or scalar (that is, contain a single value which is used as a constant applied to all values in the corresponding column of data).
- If base_data is provided, the mean and standard deviation of this data range are calculated and used to normalize data. The columns in base_data must contain two or more cell values.
- If neither of the above mutually exclusive options are provided, the mean and standard deviation are automatically computed from data.

NORM_ZSCORE always returns a data range with the same dimensions as the input data range. It computes a mean and standard deviation for each input column and uses those values for normalizing that column.

Note: If the standard deviation is zero, all zeros are returned.

Examples

TEMP = NORM_ZSCORE(COLUMN(3, 4, 5))   Creates a new column named TEMP containing the values -1.22, 0, and 1.22. (The mean and standard deviation [4 and 0.816] are calculated from the data range automatically.)
TEMP = NORM_ZSCORE(COLUMN(3, 4, 5), 3.5, 1.2)   Creates a new column named TEMP containing the values -0.42, 0.42, and 1.25. (This time the mean and standard deviation [3.5 and 1.2] are provided as arguments.)
TEMP = NORM_ZSCORE(V1)   Creates a new column named TEMP containing normalized values of the contents of column V1. The mean and standard deviation used for normalization are calculated over the column V1.

`TEMP = NORM_ZSCORE(V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated for each column independently (that is, a mean and standard deviation is calculated for column `V1`, a separate mean and standard deviation is calculated for column `V2`, etc.).
`TEMP = NORM_ZSCORE(V1[10:50]:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`, each with values in rows 1-41. The contents of column `TEMP` are the normalized values of rows 10-50 of column `V1`, the contents of column `VX` are the normalized values of rows 10-50 of column `V2`, and the contents of column `VY` are the normalized values of rows 10-50 of column `V3`. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns `V1-V3`. The mean and standard deviation for normalization purposes are calculated independently for each column.
`TEMP = NORM_ZSCORE(V1:V3, V4:V6)`
Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated for each column independently using columns `V4-V6` (that is, a mean and standard deviation is calculated over column `V4` for normalizing column `V1`, a separate mean and standard deviation is calculated over column `V5` for normalizing column `V2`, etc.).
`TEMP = NORM_ZSCORE(V1:V3, COL)`
Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated for each column independently (that is, a mean and standard deviation is calculated for column `V1`, a separate mean and standard deviation is calculated for column `V2`, etc.).
`TEMP = NORM_ZSCORE(V1[10:50]:V3, COL)`
Creates three new columns named `TEMP`, `VX`, and `VY`, each with values in rows 1-41. The contents of column `TEMP` are the normalized values of rows 10-50 of column `V1`, the contents of column `VX` are the normalized values of rows 10-50 of column `V2`, and the contents of column `VY` are the normalized values of rows 10-50 of column `V3`. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns `V1-V3`. The mean and standard deviation for normalization purposes are calculated independently for each column.
`TEMP = NORM_ZSCORE(V1[10:50]:V3, COL)`
Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated for each column independently using columns `V4-V6` (that is, a mean and standard deviation is calculated over column `V4` for normalizing column `V1`, a separate mean and standard deviation is calculated over column `V5` for normalizing column `V2`, etc.).
`TEMP = NORM_ZSCORE (V1:V3, ROW)`
Creates three new columns named `TEMP`, `VX`, and `VY`. Each contains the normalized values of the contents of columns `V1`, `V2`, and `V3`, respectively. The mean and standard deviation used for normalization are calculated over independently over each row across columns `V1`, `V2`, and `V3`.

```
TEMP = NORM_ZSCORE(V1[10:50]:V3, ROW)
```

Creates three new columns named TEMP, VX, and VY, each with values in rows 1-41. The contents of column TEMP are the normalized values of rows 10-50 of column V1, the contents of column VX are the normalized values of rows 10-50 of column V2, and the contents of column VY are the normalized values of rows 10-50 of column V3. The mean and standard deviation for normalization purposes are calculated over rows 10-50 of columns V1-V3. The mean and standard deviation for normalization purposes are calculated over each row of columns V1-V3.

```
TEMP = NORM_ZSCORE(V1:V3, V4:V10, ROW)
```

Creates three new columns named TEMP, VX, and VY. Each contains the normalized values of the contents of columns V1, V2, and V3, respectively. The mean and standard deviation used for normalization are calculated independently for each row across columns V4-V10.

Related Functions

Function	Description
NORM_MINMAX	Computes the min/max normalization of a data range
NORM_SIGMOID	Computes the sigmoidal normalization of a data range

NOT

Syntax

```
NOT(data) ! data
```

Parameters

```
data
```

The numerical values to compute the logical NOT of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

NOT returns the logical NOT of the values in the specified data range. It returns one new column for each input column, each containing the logical NOT of the values in the corresponding input column. This function returns zero for non-zero values and one for zero values.

Note: The NOT operator can be abbreviated with an exclamation mark (!). Use the exclamation mark before the data value (for example, to specify NOT(V1), you can simply type !V1).

This macro is available in IBM Interact.

Examples

`TEMP = NOT(3.2) or TEMP = !1` Creates a new column named `TEMP` containing the value zero.
`TEMP = !0 or TEMP = !(2+2=3)` Creates a new column named `TEMP` containing the value one.
`TEMP = !V1` Creates a new column named `TEMP`, where each value is the logical NOT of the values in column `V1`.
`TEMP = !V1:V3` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the logical NOTs of values in column `V1`, the values of the `VX` column are the logical NOTs of the values in column `V2`, and the values of the `VY` column are the logical NOTs of the values in column `V3`.
`TEMP = !V1[10:20]` Creates a new column named `TEMP`, where the first 11 cells contain the logical NOTs of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = !V1[1:5]:V2` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the logical NOTs of the values of the corresponding rows of column `V1`, and the values in column `VX` are the logical NOTs of the values of the corresponding rows of column `V2`.

Related Functions

Function	Description
`AND`	Computes the logical AND between two specified data ranges
`INVERSE`	Computes the negative of the contents of the specified data range
`OR`	Computes the logical OR between two specified data ranges
`SIGN`	Computes the sign (positive or negative) of the values in the specified data range

NPV

Syntax

`NPV(data, rate [, keyword])`

Parameters

`data`

The numerical values representing the expected net cash flow used to compute the net present value. This can be a row, a column, a cell range, or an expression evaluating to any of the above.

rate

The numerical value representing the rate of discount over the length of one period.

keyword

This optional keyword determines how the computation is performed over the input data range. If no keyword is provided, ROW is used as the default. Select one of the following:

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

Description

NPV calculates the net present value of an investment based on a series of periodic cash flows and a discount rate. The net present value of an investment is today's value of a series of future payments (negative values) and income (positive values). NPV is calculated by the formula:

$$NPV = \sum_{t=1}^{n} \frac{data_t}{(1 + rate)^t}$$

where n is the number of cash flows (data values).

The order of the data values is used to interpret the order of the cash flows. The NPV investment begins one period before the date of the first cash flow value and ends with the last cash flow in the list.

Note: If your first cash flow occurs at the beginning of the first period, the first value must be added to the NPV result, not included in the data values.

The number of columns returned by NPV depends on keyword.
- If keyword is COL, NPV returns a new column for each input column. Each new column contains one value (the net present value of all cells in the corresponding input column).
- If keyword is ROW, NPV returns one new column containing the net present value across each row of data.

Note: Blank cells are ignored in the NPV.

Examples

`TEMP = NPV(V1:V3,.10))` or `TEMP = NPV(V1:V3,.10, ROW)` Creates a new column named TEMP containing a single value which is the NPV of the values of columns V1, V2, and V3. The discount rate used is 10%.
`TEMP = NPV(V1, .10, COL)` Creates a new column named TEMP containing a single value which is the NPV of the contents of column V1 using a discount rate of 10%.
`TEMP = NPV(V1:V3, .10) - 1000` Creates a new column named TEMP containing a single value which is the NPV of the contents of columns V1, V2, and V3 with an `initial payment` of 1000. The discount rate is 10%.
`TEMP = NPV(V1[10:20], .10L, COL)` Creates a new column named TEMP containing a single value which is the NPV of the cells in rows 10-20 of column V1. The discount rate is 10%.
`TEMP = NPV(V1[1:5]:V4, .10)` Creates a new column named TEMP containing a single value which is the NPV of the cells in rows 1-5 of columns V1 through V4.

NUMBER

Syntax

`NUMBER(data [, conversion_keyword])`

Parameters

`data`

The ASCII text data to convert to numerical values. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`conversion_keyword`

This optional keyword specifies how to interpret text formats for dates and times. Select one of the keywords in the following table.

Note: If this parameter is not specified, the default is 1.

Conversion Keyword	Format	Description
0	#####	Converts the first 5 characters of each text string into a unique number
1	$ (default)	Converts dollar values to numerics (for example, "$123.45" to 123.45)

2	%	Converts a percentage value to numerics (for example, "50%" to `0.5`)
3	`mm/dd/yy hh:mm`	Converts a date and time to the number of days elapsed since January 1, 0000 (1900 is automatically added to the yy year)
4	`dd-mmm-yy`	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the yy year)
5	`mm/dd/yy`	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the yy year)
6	`mmm-yy`	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (1900 is automatically added to the yy year)
7	`dd-mmm`	Converts a date to the number of days since the beginning of the year (for example, "01-FEB" to `32`)
8	`mmm`	Converts a 3-letter month abbreviation to a value between 1-12 (for example, "DEC" to `12`)
9	`{January | February | March ... }`	Converts a fully spelled-out month name to a value between 1-12 (for example, "March" to `3`)
10	`{Sun | Mon | Tue ... }`	Converts a 3-day weekday abbreviation to a value between 0-6, where Sunday marks the beginning of the week (for example, "Sun" to `0`)
11	`{Sunday | Monday | Tuesday ... }`	Converts a fully spelled-out weekday name to a value between 0-6, where Sunday marks the beginning of the week (for example, "Monday" to `1`)
12	`hh:mm:ss {AM | PM}`	Converts the time to the number of seconds elapsed since 00:00:00 AM (midnight) (for example, "01:00:00 AM" to `3600`)

13	`hh:mm:ss`	Converts the time to the number of seconds elapsed since 00:00:00 AM (midnight) (for example, "01:00:00" to 3600)
14	`hh:mm {AM │ PM}`	Converts the time to the number of minutes elapsed since 00:00:00 AM (midnight) (for example, "01:00 AM" to 60)
15	`hh:mm`	Converts the time to the number of minutes elapsed since 00:00:00 AM (midnight) (for example, "01:00" to 60)
16	`mm:ss`	Converts the time to the number of seconds elapsed since 00:00:00 AM (midnight) (for example, "30:00" to 1800)
17	`ddmm`	Converts a date to the number of days since the beginning of the year (for example, "3101" to 31)
18	`ddmmm`	Converts a date to the number of days since the beginning of the year (for example, "31JAN" to 31)
19	`ddmmmyy`	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
20	`ddmmmyyyy`	Converts a date to the number of days elapsed since January 1, 0000 (for example, "31JAN0000" to 31)
21	`ddmmyy`	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
22	`ddmmyyyy`	Converts a date to the number of days elapsed since January 1, 0000 (for example, "31010000" to 31)
23	`mmdd`	Converts a date to the number of days since the beginning of the year (for example, "0131" to 31)

24	mmddyy	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
25	mmddyyyy	Converts a date to the number of days elapsed since January 1, 0000 (for example, "01010001" to 366)
26	mmm	Converts a 3-letter month abbreviation to a value between 1-12 (for example, "MAR" to 3) [Note this is the same as conversion keyword 8]
27	mmmdd	Converts a date to the number of days since the beginning of the year (for example, "JAN31" to 31)
28	mmmddyy	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
29	mmmddyyyy	Converts a date to the number of days elapsed since January 1, 0000 (for example, "FEB010001" to 32)
30	mmmyy	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
31	mmmyyyy	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (for example, "FEB0001" to 32)
32	mmyy	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)

33	mmyyyy	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (for example, "020001" to 32)
34	yymm	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
35	yymmdd	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
36	yymmm	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
37	yymmmdd	Converts a date to the number of days elapsed since January 1, 0000 (1900 is automatically added to the year if yy is less than or equal to 20; otherwise 2000 is added)
38	yyyy	Converts the year the number of years elapsed since the year 0000 (for example, "1998" to 1998)
39	yyyymm	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (for example, "000102" to 32)
40	yyyymmdd	Converts a date to the number of days elapsed since January 1, 0000 (for example, "00010201" to 32)
41	yyyymmm	Converts a date to the number of days elapsed between the first of the specified month and January 1, 0000 (for example, "000102" to 32)

42	yyyymmmdd	Converts a date to the number of days elapsed since January 1, 0000 (for example, "0001FEB01" to 32)
43	<day>* <month>	Converts any delimited date with day followed by month to the number of days elapsed since the beginning of the year (for example, "15-JAN" to 15)
44	<day>* <month>* <year>	Converts any delimited date with day appearing before month followed by year to the number of days elapsed since January 1, 0000 (for example, "1/1/0001" to 366)
45	<month>* <day>	Converts any delimited date with month followed by day to the number of days since the beginning of the year (for example, "JAN 31" to 31)
46	<month>* <day>* <year>	Converts any delimited date with month followed by day followed by year to the number of days elapsed since January 1, 0000 (for example, "JAN 1, 0001" to 366)
47	<month>* <year>	Converts any delimited date with month followed by year to the number of days elapsed between the first of the specified month and January 1, 0000
48	<year>* <month>	Converts any delimited date with year followed by month to the number of days elapsed between the first of the specified month and January 1, 0000
49	<year>* <month>* <day>	Converts any delimited date with month followed by day followed by year to the number of days elapsed since January 1, 0000 (for example, "0001/01/01" to 366)
50	yy	Converts the year to the number of years elapsed since the year 0000 (for example, "97" to 97)
51	mm	Converts the month to a value between 1-12 (for example, "SEP" to 9)

52	dd	Converts the day to a value between 1-31 (for example, "28" to 28)
53	{January | February | March ... }	Converts a fully spelled-out month name to a value between 1-12 (for example, "March" to 3) [Note this is the same as conversion keyword 9]
54	{Sunday | Monday | Tuesday ... }	Converts a fully spelled-out weekday name to a value between 1-7, where Sunday marks the beginning of the week (for example, "Sunday" to 1)
55	{Sun | Mon | Tue ... }	Converts a 3-day weekday abbreviation to a value between 1-7, where Sunday marks the beginning of the week (for example, "Sun" to 1)

Description

NUMBER converts text values in the specified data range into numerical values using the specified format for converting dates and times. If a text string cannot be parsed using the specified conversion_keyword, NUMBER will generate an error. Format 0 converts the first five characters of each text string into different number for each unique text string. This is an easy way to change a column of text into unique classes for outputs to a classifier.

The delimited formats (conversion keywords 43-49) support any of the following as delimiters:
- / (slash)
- - (dash)
- , (comma)
- " " (space)
- : (colon)

Months can be represented as mm or mmm; days can be represented as d or dd; years can be represented as yy or yyyy.

Note: In support of year 2000 compliance, all years in dates may be designated as yyyy instead of yy. For backwards compatibility, conversion keywords 1-16, yy (2-digit years) automatically have 1900 added. For conversion keywords 17-55, yy < threshold automatically have 2000 added; yy ≥ threshold automatically have 1900 added.

Note: The year 2000 threshold value is set in the **Data Cleaning** tab of the **Advanced Settings** window (invoke using **Options > Settings > Advanced Settings**).

Note: If you change the value year 2000 threshold value, you must update all macro functions using the NUMBER macro function to manipulate date values with

2-digit years. To force an update of a macro function, you can make any edit (for example, adding a space and deleting it) and clicking the check mark icon to accept the change.

Note: When using format 0, only the first five characters of each text string are used to generate a unique number. All strings with the same first five characters will be translated to the same numeric value. The same text string will produce the same numerical value every time, even across different spreadsheets. If required, use string macros to manipulate strings so that the first five characters uniquely define a class. Note that the resulting numerical values may be very small. Use the **Display Formats** window to either increase the number of decimal places displayed, or change the format to exponential mode (00E+00).

This macro is available in IBM Interact.

Examples

`TEMP = NUMBER("$1.23")` or `TEMP = NUMBER("123%", 2)`   Creates a new column named `TEMP` containing the number `1.23`.
`TEMP = NUMBER(column("Jan", "Mar", "Dec", 8)`   Creates a new column named `TEMP` containing the numbers 1, 3, and 12.
`TEMP = NUMBER("1:52 PM", 14)`   Creates a new column named `TEMP` containing the number 832.
`TEMP = NUMBER("1/1/95", 5)`   Creates a new column named `TEMP` containing the number 728660.
`TEMP = NUMBER(V1)`   Creates a new column named `TEMP` containing the numeric values of the text strings in column `V1`. Any dollar values are correctly converted into numerical values. ??? 's returned for text strings that cannot be parsed using the $ format.
`TEMP = NUMBER(V1:V3, 4)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the numerical values of text strings in column `V1`. The column `VX` contains the numerical values of text strings in column `V2`. The column `VY` contains the numerical values of text strings in column `V3`. Any dates in the format `dd-mmm-yy` are converted into the number of days offset from January 1, 0000. ??? 's are returned for text strings that cannot be parsed using the $ format.
`TEMP = NUMBER(V1[10:20]:V2, 10)`   Creates two new columns named `TEMP` and `VX`. The column `TEMP` contains the numerical values of text strings in rows 10-20 of column `V1`. The column `VX` contains the numerical values of text strings in rows 10-20 column `V2`. All standard three character representations of days of the week are converted into the numbers 0-6 (0 = Sunday, 6= Saturday). If there is no match for a weekday name, ??? is returned.
`TEMP = NUMBER(V1, 0)`   Assuming that column `V1` contains all 5-digit text strings, creates one new column named `TEMP` containing a different numerical value for each unique string.

Related Functions

Function	Description
WEEKDAY	Converts ASCII text date strings to the day of the week

OFFSET

Syntax

OFFSET(data)

Parameters

data

The values to compute the offset of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

OFFSET returns the values of the specified data range offset from the first value. It returns one new column containing the offset values, beginning with one and continuing to the length of the longest column in the data range.

Examples

TEMP = OFFSET(5)
Creates a new column named TEMP containing the value one.
TEMP = OFFSET(V1)
Creates a new column named TEMP, where each value is the offset of the column V1, starting with one and ending with the length of the column V1.
TEMP = OFFSET(V1:V3)
Creates a new column named TEMP, where each value is the offset, starting with one and ending with the longest column of V1, V2, or V3.

Related Functions

Function	Description
COUNT	Counts the number of cells containing values in the specified data range
DELAY	Returns the input column(s) values delayed by a specified number of time steps

OR

Syntax

```
data1 OR data2 data1 || data2
```

Parameters

data1

The numbers to logical OR with the values in data2. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The number(s) to logical OR with the values in data1. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in data2 must equal the number of columns in data1, unless data2 is a constant. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

OR calculates the logical OR between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in data1 logically OR-ed to the corresponding column of data2 (that is, the first column of data1 is logically OR-ed to the first column of data, the second column with the second column, and so on).

If data2 is a constant, each value in data1 is logically OR-ed by that value. If data2 contains one or more columns, the calculations are performed on a row-by-row basis between one column from data2 and one column from data2. The first row of data1 is logically OR-ed to the first row value of data2, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Using a column containing the same number x in each row as data2 is the same as using the constant x as data2.

Note: The OR operator can be abbreviated with a double-vertical bar (||). Use the double-vertical bar to separate the two arguments (for example, to specify V1 OR 3, you can simply type V1||3).

This macro is available in IBM Interact.

Examples

```
TEMP = 1 OR 8 or TEMP = 1 || 8
```

Creates a new column named TEMP containing the value one (any non-zero number is treated as a one).

`TEMP = V1 || 1` Creates a new column named `TEMP` containing all ones (every value OR-ed with the number one produces one).
`TEMP = V1 || V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` logically OR-ed with the corresponding row value of column `V2`.
`TEMP = V1:V3 || V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` logically OR-ed with the corresponding row values of column `V4`. The column `VX` contains the logically OR-ed values from columns `V2` and `V5`. The column `VY` contains the logically OR-ed values from columns `V3` and `V6`.
`TEMP = V1[10:20] || V2` Creates a new column named `TEMP`, where the first 11 cells contain the logical OR-ed result of the values in rows 10-20 of columns `V1` and `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`AND`	Computes the logical AND between two specified data ranges
`NOT`	Computes the logical NOT of the contents of the specified data range

---

PCA

Syntax

`PCA(data)`

Parameters

`data`

The numerical values for which to compute the principal components. This can be a constant, a column, a cell range, or an expression evaluating to any of the above.

Description

PCA performs principal component analysis on the specified data range. It finds the orthogonal eigenvectors to the data range specified by `data` using singular value decomposition. It returns one new column for each of the n columns specified as input, plus one additional column. The first n columns contain the eigenvectors (each eigenvector is read as a row across the n columns). The last returned column contains the corresponding magnitudes of eigenvalues. The eigenvectors are ordered according to their corresponding eigenvalues.

Note: Missing values (for example, empty cells and ??? 's) are counted as zeros. Any short columns in `data` are padded with zeros to the length of the longest column.

Here are the details for how the PCA is computed:

- Each of the k rows of `data` is an n-dimensional vector v_i (n is the number of columns in `data`). These are used to compute the correlation matrix A as follows:

$$A = \sum_{i=1}^{k} v_i v_i^{T}$$

- The n-by-n correlation matrix A is decomposed using singular value decomposition into three matrices:

$$A = U \Sigma U^{T}$$

The rows of U are the eigenvectors of A and Σ is a diagonal matrix where each diagonal element is the magnitude of the eigenvalues for A.

The `PCA` macro function returns U in the first n columns and the diagonal elements of Σ in the last column.

Examples

`TEMP = PCA(5)`Creates two new columns named `TEMP` and `VX`, containing the values `-1` and `0`, respectively.
`TEMP = PCA(V1)` Creates two new columns named `TEMP` and `VX`. The column `TEMP` contains the value one, and the column `VX` contains the corresponding eigenvalue.
`TEMP = PCA(V1:V3)` Creates four new columns named `TEMP`, `VX`, `VY`, and `VZ`. The values in the three columns contain one eigenvector per row for the data in columns `V1-V3`. The value in column `VZ` contains the corresponding eigenvalues.

Related Functions

Function	Description
`PCA_FEATURES`	Extracts n features from the specified data range

PCA_FEATURES

P

Syntax

PCA_FEATURES(num_features, data [, PCA(base_data)])

Parameters

num_features

The number of features to extract from the specified data range using principal component analysis (PCA). This value must be a positive integer between one and the number of columns in the data range specified by data.

data

The numerical values to extract features from. This can be a column, a cell range, or an expression evaluating to either of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

PCA(base_data)

If this optional parameter is provided, PCA is performed on this base_data data range and the resulting eigenvectors are used to extract features from the data data range. For the format definition of base_data (same as data), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product. The number of columns in base_data must be the same as the number of columns in data.

Description

PCA_FEATURES extracts the top num_features features from the specified data range. It returns num_features columns using the eigenvectors generated by principal component analysis on the data range base_data, if provided. If not provided, it uses data to generate the eigenvectors. In this case, data is automatically normalized using the zero-mean unit-variant method prior to the principal component analysis.

The features are computed as follows:
• The data range data is automatically normalized using the zero-mean unit-variant method. In other words,
  PCA_FEATURES(num_features, data)

is equivalent to

PCA_FEATURES(num_features, data,PCA(data, COL))

No normalization of data is provided automatically. To normalize data using NORM_ZSCORE, you can specify the following:

PCA_FEATURES(num_features, data, PCA(NORM_ZSCORE(data, COL)))

- Principal component analysis is performed on the normalized data range to generate its eigenvectors (see details described for the PCA macro function). This occurs automatically for `data` if `base_data` is not provided. It is performed by the explicit call to the PCA macro function if `base_data` is provided.

- Each row (v_i) of the data range (`data`) is transformed into a new coordinate system (u_i) based on the top `num_features` (m) ranked eigenvectors which compose E_m:

$$u_i = E_m v_i = \begin{bmatrix} E_{11} & \cdots & E_{1n} \\ \cdots & \cdots & \cdots \\ E_{m1} & \cdots & E_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ \cdots \\ v_n \end{bmatrix}$$

- The k rows of the transformed data (u_1 to u_k) are returned (n columns).

If the `base_data` data range is provided, it must have the same number of columns as the `data` data range, otherwise an error is returned.

Note: Because calculating PCA on a data range can be compute intensive, using the BUFFER macro function on the PCA calculation is much more efficient. For example: `PCA_FEATURES(num_features, range, BUFFER(PCA(base_data)))`

Examples

`TEMP = PCA_FEATUES(5, V1:V7)`Creates five new columns named TEMP, VW, VX, VY, and VZ, containing the top five features of the data range V1:V7. The data range V1:V7 is used as the basis for the transformation.
`TEMP = PCA_FEATURES(3, V1:V4, PCA(V10:V13))` Creates three new columns named TEMP, VX, and VY, containing the top three features of the data range V1:V4. The data range V10:V13 is used as the basis for the transformation.
`TEMP = PCA_FEATURES(3, V1:V4, BUFFER(PCA(V10:V13)))` Creates three new columns named TEMP, VX, and VY, containing the top three features of the data range V1:V4. The data range V10:V13 is used as the basis for the transformation. Once the principal components of the data range V10:V13 are calculated, those values are stored as constants. If the data values in columns V10 - V13 change, they will not effect this function definition.

Related Functions

Function	Description
PCA	Computes the eigenvectors for principal components of the specified data range

POSITION

Syntax

POSITION(colName, pattern [, start [, occurrence]])

Parameters

colName

The value of a column (must be string type).

pattern

The pattern, or string, for which you are searching.

start

The byte with which to begin the search.

occurrence

Specify a value for n, where you are searching for the n th occurrence of the pattern to return.

Description

POSITION returns the starting byte position of a pattern, or string, within the value of a column (colName) which must be string type. If start is specified, it begins to search from there. Occurrence is the nth occurrence of pattern to return.

Note: The search is not case sensitive.

Examples

In the example below, we are searching for the pattern or string, 'A ', within the value of the column, dbo_BaseInfo.BranchCd, and assigning the returned value to a derived filed POStest.
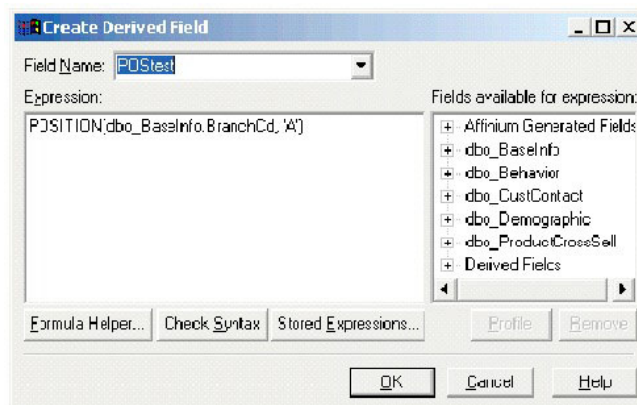
The following example shows a few rows from the table with the values from dbo_BaseInfo.BranchCd and POStest shown side-by-side.

A more complex example:

```
STRING_SEG(POSITION(CellCode,"X",1,2)+1,
```

```
STRING_LENGTH(CellCode),CellCode) = "AAA"
```

This returns rows where the values of `CellCode` have "AAA " at the end following the second occurrence of "X ".

PLUS

Syntax

```
data PLUS addend data + addend
```

Parameters

`data`

The cell range containing numbers to add. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`addend`

The number(s) to add to all values in the specified column. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `data2` must equal the number of columns in `data1`, unless `data2` is a constant. For the format definition of `addend` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

PLUS adds the values in the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in `data1` summed with the corresponding column of `data2` (that is, the first column of `data1` is added to the first column of `data`, the second column with the second column, and so on).

If `data2` is a constant, each value in `data1` increased by that value. If `data2` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `data2` and one column from `data2`. The first row of `data1` is added to the first row value of `data2`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The `PLUS` operator can be abbreviated with a plus sign (+).

This macro is available in IBM Interact.

Examples

`TEMP = 3 PLUS 4 or TEMP = 3 + 4` Creates a new column named `TEMP` containing the value seven.
`TEMP = V1 + 8` Creates a new column named `TEMP`, where each value is the contents of column `V1` plus eight.
`TEMP = V1 + V1` Creates a new column named `TEMP` containing two times the contents of column `V1`.
`TEMP = V1 + V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` plus the corresponding row value of column `V2`.
`TEMP = V1:V3 + V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` plus the corresponding row values of column `V4`. The column `VX` sums columns `V2` and `V5`. The column `VY` sums column `V5` and `V6`.
`TEMP = V1[10:20] + V2 or TEMP = V1[10:20] + V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the sums of the values in rows 10-20 of column `v1` and the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`MINUS`	Subtracts one specified data range from another
`SUM or TOTAL`	Computes the sum of a range of cells

POW

Syntax

`base POW exponent base ^ exponent`

Parameters

`base`

The numerical values to raise to an exponential power. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `base` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`exponent`

The exponential number(s) to raise the values in `data` by. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in `exponent` must equal the number of columns in `base`, unless `base` is a constant. For the format definition of `exponent` (same as `data`), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`POW` raises the values in the first data range to the power specified in the second data range (that is, calculates $base^{exponent}$). It returns one new column for each input column, each containing the result of raising the `base` to the `exponent` power (that is, the first column of `data1` is raised to the first column of `data`, the second column with the second column, and so on).

If `exponent` is a constant, each value in `base` is raised by that value. If `exponent` contains one or more columns, the calculations are performed on a row-by-row basis between one column from `base` and one column from `exponent`. The first row of `base` is raised to the first row value of `exponent`, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The `POW` operator can be abbreviated with a circumflex (^). For example, `TEMP = 2^8` is equivalent to `TEMP = 2 POW 8`.

Note: If the value x is too large or too small, an overflow is returned. This occurs if `base^exponent` exceeds the maximum or minimum 32-bit floating-point value.

Examples

`TEMP = 2 POW 3 or TEMP = 2^3`   Creates a new column named `TEMP` containing the value eight.
`TEMP = V1 ^ 0.5`   Creates a new column named `TEMP`, where each value is the square root of the contents of column `V1` (this is equivalent to `SQRT(V1)`).
`TEMP = V1 ^ V3`   Creates a new column named `TEMP`, where each value is the row value of column `V1` raised to the corresponding row value of column `V2`.
`TEMP = V1:V3 ^ V4:V6`   Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` raised to the corresponding row values of column `V4`. The column `VX` contains the result of column `V2` raised to the corresponding values in column `V5`. The column `VY` contains the the result of column `V3` raised to the corresponding values of `V6`.

```
TEMP = V1[10:20] POW V2 or TEMP = V1[10:20] POW V2[1:11]
```

Creates a new column named TEMP, where the first 11 cells contain the result of raising the values in rows 10-20 of column V1 by the values in rows 1-10 of column V2. The other cells in TEMP are empty.

Related Functions

Function	Description
EXP	Computes the natural number (e) raised to the contents of each cell in the specified data range
LN or LOG	Computes the natural log of the contents of the specified data range
LN2	Computes the log base2 of the contents of the specified data range
LN10	Computes the log base10 of the contents of the specified data range

RANDOM

Syntax

RANDOM(num [, seed]) RANDOM(num, value1, value2 [, seed])

Parameters

num

The number of random numbers to generate. This value must be a positive integer greater than zero.

value1

A bound on the random numbers to generate. This can be any constant value or an expression evaluating to a constant. If this parameter is not provided, the default is zero.

value2

The other bound on the random numbers to generate. This can be any constant value or an expression evaluating to a constant. If this parameter is not provided, the default is one.

seed

An optional seed to use for random number generation. This must be an integer.

Description

RANDOM generates a column of random numbers. It returns one new column containing num random numbers. If value1 and value2 are specified, the random numbers will be generated between (and including) those bounds. If they are not

specified, the default is to generate values between zero and one. If seed is
provided, it will be used as a seed to the random number generator.

Note: If seed is greater than or equal to 2^{32} , the value is replaced with 2^{32}- 1.
Values of seed above 2^{24} will be rounded (that is, precision is lost). Therefore,
multiple values may result in the same value of seed.

Examples

`TEMP = RANDOM()`
Creates one new column named TEMP containing random numbers of unlimited length.
`TEMP = RANDOM(100)`
Creates one new column named TEMP containing 100 random numbers between 0.0 and 1.0.
`TEMP = RANDOM(100, 5943049)`
Creates one new column named TEMP containing 100 random generated from the seed number 5943049.
`TEMP = RANDOM(100, 0, 100)`
Creates one new column named TEMP containing 100 random numbers between 0 and 100.0.
`TEMP = RANDOM(100, 0, 100, 5943049)`
Creates one new column named TEMP containing 100 random numbers between -0 and 100 generated from the seed number 5943049.

Related Functions

Function	Description
`RANDOM_GAUSS`	Returns the specified number of random values from a Gaussian distribution
`SAMPLE_RANDOM`	Returns column(s) of n cells, each containing a random sample from the specified data range

RANDOM_GAUSS

Syntax

`RANDOM_GAUSS(num [, seed]) RANDOM_GAUSS(num, mean, std [, seed])`

Parameters

`num`

The number of random numbers to generate. This value must be a positive integer
greater than zero.

`mean`

The mean of the Gaussian. This can be any constant value or an expression
evaluating to a constant. If this parameter is not provided, the default is zero.

```
std
```

The standard deviation of the Gaussian. This can be any constant value or an expression evaluating to a constant. If this parameter is not provided, the default is one.

```
seed
```

An optional seed to use for random number generation. This must be an integer. (If a non-integer value is supplied, the floor of the value is automatically used instead.)

Description

RANDOM_GAUSS generates a column of random numbers based on a Gaussian distribution. It returns one new column containing num random numbers. If mean and std are specified, the random numbers will be generated using a Gaussian distribution with the specified mean and standard deviation. If they are not specified, the default Gaussian has a mean of zero and standard deviation of one. If seed is provided, it will be used as a seed to the random number generator.

Examples

`TEMP = RANDOM_GAUSS(100)`   Creates one new column named TEMP containing 100 values randomly sampled from a zero-mean, unit-standard deviation Gaussian.
`TEMP = RANDOM_GAUSS(500, 3)`   Creates one new column named TEMP containing 100 values randomly sampled from a zero-mean, unit-standard deviation Gaussian. The number 3 is used as a seed for the random number generator.
`TEMP = RANDOM_GAUSS(5000, 100, 32)`   Creates one new column named TEMP containing 5000 values randomly sampled from a Gaussian with a mean of 100 and standard deviation of 32.
`TEMP = RANDOM_GAUSS(500, -1, 2, 3)`   Creates one new column named TEMP containing 500 values randomly sampled from a Gaussian with a mean of -1 and a standard deviation of 2. The number 3 is used as a seed for the random number generator.

Related Functions

Function	Description
RANDOM	Returns the specified number of random numbers
SAMPLE_RANDOM	Returns column(s) of n cells, each containing a random sample from the specified data range

RANK

Syntax

```
RANK(data [, nbins])
```

Parameters

data

This can be a constant value, a column, a cell range, or an expression evaluating to any of the above.

nbins

The number of bins which the data will be decided into. The default value is ten.

Description

RANK divides data into nbins (default 10) groups, each with approximately an equal number of distinct values, and returns the group into which each row of data falls. The output will be between 1 and nbins; if the number of distinct values of data is less than nbins, the output will be between 1 and the number of distinct values of data.

RANK silently imposes an upper bound of 1024*1024 on the value of nbins. That same number is also used as a maximum number of distinct values to track; subsequent distinct values will be included in the highest bin.

Examples

TEMP=RANK(V6) Creates a new column named TEMP with each value, a one through ten, representing the bin of data that the row falls into. In this case, the default number of bins, 10, is applied.
TEMP=RANK(V6,15) Creates a new column named TEMP with each value, a one through fifteen, representing the bin of data that the row falls into.
TEMP = REPEAT(3, V1:V3) Creates three new columns named TEMP, VX, and VY. TEMP contains three copies of column V1, VX contains three copies of column V2, and VY contains three copies of column V3. Any uneven length columns are padded to the length of the longest column of V1 - V3.
TEMP = REPEAT(10, V1[10:50]:V2) Creates two new columns named TEMP and VX. The values in column TEMP are 10 copies of rows 10-50 of column V1, and the values in column VX are 10 copies of rows 10-50 of column V2.
TEMP = REPEAT((1,2,3), (10, 20, 30)) Creates one new column named TEMP, containing the cell values 10, 20, 20, 30, 30, 30.
TEMP = REPEAT(V1, V2) Creates one new column named TEMP. The value in cell V2[1] is repeated V1[1] times, the value in cell V2[2] is repeated V1[2] times, and so on until the end of column V1.

`TEMP = REPEAT(V1, V2:V3)`
Creates two new columns named TEMP and VX. The TEMP column contains copies of cells in V2; the VX column contains copies of cells in V3. There are V1[1] copies of V2[1] and V3[1], V1[2] copies of V2[2] and V3[2]. This continues until the end of column V1 or the end of the longest column in `data`, which ever is shorter. Shorter columns in `data` are padded with zeros.
`TEMP = REPEAT(3, V1, COL)`
Creates three new columns named TEMP, VX, and VY, each a copy of column V1.
`TEMP = REPEAT(2, V1:V2, COL)`
Creates four new columns named TEMP, VX, VY, and VZ. Column TEMP is a copy of column V1; column VX is a copy of column V2, column VY is a copy of column V1; and column VZ is a copy of column V2.

REPEAT

P

Syntax

`REPEAT(num_times, data [, keyword])`

Parameters

num_times

The number of times to repeat the specified data range. This can be a constant, a column, or an expression evaluating to any of the above. All values must be positive integers.

data

The numerical values to repeat. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

keyword

This optional keyword determines how the specified data range is replicated. Select one of the following:

ROW - Repeats rows of `data` vertically (default)

COL - Repeats columns of `data` horizontally

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

REPEAT repeats the specified data range a number of times, either vertically (COL) or horizontally (ROW).

With the COL keyword, it returns one new column for each input column containing num_times copies of data, vertically concatenated. If the input columns of data have different lengths, the shorter columns are automatically padded. Numeric columns are padded with zero and text string columns are padded with empty strings ("").

With the ROW keyword, the number of columns returned depends on whether num_times is a constant or a column. If num_times is a constant, it returns num_times times the number of columns in data. If num_times is a column, it returns the specified number of copies of each column of data, where the first row value of num_times is the number of times to repeat the first column of data; the second row value is the number of times to repeat the second of column of data, etc. Any rows greater than the number of columns in data ignored.

Examples

`TEMP = REPEAT(10, 1) or TEMP = REPEAT(10, 1, ROW)` Creates a new column named TEMP containing ten 1 's.
`TEMP = REPEAT(2, V1)` Creates a new column named TEMP containing two copies of the column V1 concatenated vertically.
`TEMP = REPEAT(3, V1:V3)` Creates three new columns named TEMP, VX, and VY. TEMP contains three copies of column V1, VX contains three copies of column V2, and VY contains three copies of column V3. Any uneven length columns are padded to the length of the longest column of V1 - V3.
`TEMP = REPEAT(10, V1[10:50]:V2)` Creates two new columns named TEMP and VX. The values in column TEMP are 10 copies of rows 10-50 of column V1, and the values in column VX are 10 copies of rows 10-50 of column V2.
`TEMP = REPEAT((1,2,3), (10, 20, 30))` Creates one new column named TEMP, containing the cell values 10, 20, 20, 30, 30, 30.
`TEMP = REPEAT(V1, V2)` Creates one new column named TEMP. The value in cell V2[1] is repeated V1[1] times, the value in cell V2[2] is repeated V1[2] times, and so on until the end of column V1.
`TEMP = REPEAT(V1, V2:V3)` Creates two new columns named TEMP and VX. The TEMP column contains copies of cells in V2; the VX column contains copies of cells in V3. There are V1[1] copies of V2[1] and V3[1], V1[2] copies of V2[2] and V3[2]. This continues until the end of column V1 or the end of the longest column in data, which ever is shorter. Shorter columns in data are padded with zeros.
`TEMP = REPEAT(3, V1, COL)` Creates three new columns named TEMP, VX, and VY, each a copy of column V1.

```
TEMP = REPEAT(2, V1:V2, COL)
```

Creates four new columns named TEMP, VX, VY, and VZ. Column TEMP is a copy of column V1; column VX is a copy of column V2, column VY is a copy of column V1; and column VZ is a copy of column V2.

ROTATE_LEFT

Syntax

```
ROTATE_LEFT(num_cols, data)
```

Parameters

num_cols

The number of columns to rotate to the left. This value must be a non-negative integer. The value of zero copies the rows without any rotation.

data

The numerical values to rotate to the left. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

ROTATE_LEFT rotates the specified data range left by a number of columns. It returns one new column for each input column, each containing a copy of the corresponding input column rotated to the left by num_cols positions. Columns that are rotated off the left are wrapped around on the right.

Note: ROTATE_LEFT only works with numerical data. None of the data provided in the data parameter can be ASCII text.

Examples

```
TEMP = ROTATE_LEFT(1, MERGE(1, 2, 3)
```

Creates three new columns named TEMP, VX, and VY. TEMP contains the value two, VX contains the value three, and VY contains the value one.

```
TEMP = ROTATE_LEFT(0, V1:V3)
```

Creates three new columns named TEMP, VX, and VY. TEMP is a copy of column V1, VX is a copy of column V2, and VY s a copy of column V3.

```
TEMP = ROTATE_LEFT(4, V1:V3)
```

Creates three new columns named TEMP, VX, and VY. TEMP is a copy of column V2, VX is a copy of column V3, and VY is a copy of column V1.

```
TEMP = ROTATE_LEFT(1, V1[10:50]:V2)
```

Creates two new columns named TEMP and VX, each with values in the first 41 rows (the other cells are empty). The values in column TEMP are the values in rows 10-50 of column V2, and the values in column VX are the values in rows 10-50 of column V1.

Related Functions

Function	Description
ROTATE_RIGHT	Rotates the columns in the specified data range to the right

ROTATE_RIGHT

Syntax

```
ROTATE_RIGHT(num_cols, data)
```

Parameters

num_cols

The number of columns to rotate to the right. This value must be a non-negative integer. The value of zero copies the rows without any rotation.

data

The numerical values to rotate to the right. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

ROTATE_RIGHT rotates the specified data range right by a number of columns. It returns one new column for each input column, each containing a copy of the corresponding input column rotated to the right by num_cols positions. Columns that are rotated off the right are wrapped around on the left.

Note: ROTATE_RIGHT only works with numerical data. None of the data provided in the data parameter can be ASCII text.

Examples

```
TEMP = ROTATE_RIGHT(1, MERGE(1, 2, 3)
```

Creates three new columns named TEMP, VX, and VY. TEMP contains the value three, VX contains the value one, and VY contains the value two.

```
TEMP = ROTATE_RIGHT(0, V1:V3)
```

Creates three new columns named TEMP, VX, and VY. TEMP is a copy of column V1, VX is a copy of column V2, and VY is a copy of column V3.

`TEMP = ROTATE_RIGHT(4, V1:V3)`
Creates three new columns named TEMP, VX, and VY. TEMP is a copy of column V3, VX is a copy of column V1, and VY is a copy of column V2.
`TEMP = ROTATE_RIGHT(1, V1[10:50]:V2)`
Creates two new columns named TEMP and VX, each with values in the first 41 rows (the other cells are empty). The values in column TEMP are the values in rows 10-50 of column V2, and the values in column VX are the values in rows 10-50 of column V1.

Related Functions

Function	Description
`ROTATE_LEFT`	Rotates the columns in the specified data range to the left

ROUND

Syntax

`ROUND(data)`

Parameters

`data`

The numerical values to round. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

ROUND rounds the values in the specified data range to the nearest integer. It returns one new column for each input column, each containing the rounded value of numbers in the corresponding input column. Numbers exactly halfway in-between are rounded up (for example, `2.5` is rounded to `3.0` and `2.5` is rounded to `-2.0`).

Examples

`TEMP = ROUND(3.2)`
Creates a new column named TEMP containing the value three.
`TEMP = ROUND(V1)`
Creates a new column named TEMP, where each value is the rounded value of the contents of column V1.
`TEMP = ROUND(V1:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the rounded values of the contents of column V1, the values of the VX column are the rounded values of the contents of column V2, and the values of the VY column are the rounded values of the contents of column V3.

`TEMP = ROUND(V1[10:20])`
Creates a new column named `TEMP`, where the first 11 cells contain the rounded values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = ROUND(V1[1:5]:V2)`
Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the rounded values of the corresponding rows of column `V1`, and the values in column `VX` are the rounded values of the corresponding rows of column `V2`.

Related Functions

Function	Description
`INT`	Computes the integer value (rounded down) of the contents of the specified data range
`MOD`	Computes the modulo of the contents of the specified data range
`TRUNCATE`	Returns the non-fractional part of each value in the specified data range

ROWNUM

Syntax

`ROWNUM()`

Description

ROWNUM generates sequential numbers from one to the number of records. The number for the first record is one, two for the second record, and so on

Note: The maximum number of records that `ROWNUM` can handle is two billion.

RTRIM

Syntax

`RTRIM(data)`

Parameters

`data`

Description

RTRIM removes trailing space characters from each string value in the specified data range, returning the converted string. It returns one new column for each input column.

This macro is available in IBM Interact.

Examples

```
Temp = RTRIM "gold "
```

Creates a new string named Temp which contains "gold".

SAMPLE_RANDOM

Syntax

SAMPLE_RANDOM(num_samples, data [, seed])

Parameters

num_samples

The number of samples to take from each column in the specified data range.

data

The values to randomly sample. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

seed

An optional seed to use for random number generation. This must be an integer. (If a non-integer value is supplied, the floor of the value is automatically used instead.)

Description

SAMPLE_RANDOM randomly samples the specified data range. It returns one new column for each input column, each containing num_samples numbers randomly sampled from the corresponding input column of data. Samples are taken in the order they appear in each column (that is, data values will remain in the same relative order to each other). If seed is provided, it will be used as a seed to the random number generator.

Examples

```
TEMP = SAMPLE_RANDOM(100, 3)
```

Creates a new column named TEMP with 100 cells all containing the value 3.

```
TEMP = SAMPLE_RANDOM(100, V1)
```

Creates a new column named TEMP containing 100 values, where each value is a random sample of the contents of column V1.

TEMP = SAMPLE_RANDOM(50, V1:V3)
Creates three new columns named TEMP, VX, and VY, each containing 50 values. The values in the TEMP column are random samples from column V1, the values of the VX column are random samples from column V2, and the values of the VY column are random samples from column V3.
TEMP = SAMPLE_RANDOM(100, V1[10:50]:V3)
Creates three new columns named TEMP, VX, and VY, each containing 41 values. The values in the TEMP column are random samples from cells 10-50 of column V1, the values of the VX column are random samples from cells 10-50 of column V2, and the values of the VY column are random samples from cells 10-50 of column V3. The other cells in TEMP are empty.

Related Functions

Function	Description
RANDOM	Returns the specified number of random numbers
RANDOM_GAUSS	Returns the specified number of random values from a Gaussian distribution
SUBSAMPLE	Reduces data by returning every n-th row value

SELECT

Syntax

SELECT(col_nums, data) SELECT(from_col, data) SELECT(from_col, to_col, data)

Parameters

col_nums

The name of a column containing the column numbers to extract from the specified data range (for example, a column containing the numbers 1, 3, 4, and 7 would extract the first, third, fourth, and seventh column from the specified data range). This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of col_nums (same as data), see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

from_col

The numerical position of the column to extract from the specified data range. A value of one extracts the first column of a specified data range.

to_col

If this parameter is provided, `from_col` is used as a starting point and must be a column or a cell range. The ending point is specified by `to_col`. This value must be larger than `from_col`.

`data`

The cell range containing the column(s) to extract. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`SELECT` returns the specified columns from a data range. The columns to select can be specified in different ways. The `col_nums` parameter contains column numbers to extract from the specified data range. to extract a single column, use `from_col`. To extract a consecutive range of columns, specify `from_col` as a starting point (it must be a column or a cell range) and `to_col` as an ending column.

If `data` is scalar (that is, a constant or a variable containing a single value), selecting the first column returns a new column containing that constant. Selecting any other column from a constant returns a column containing ???.

This macro is often embedded into more complex functions.

Note: To extract multiple non-adjacent columns, use the `COLUMN` macro function to create a column containing the column numbers to select. See the examples below.

Examples

`TEMP = SELECT(1, 3)` Creates a new column named `TEMP` containing the number three.
`TEMP = SELECT(1, V1)` or `TEMP = SELECT(1, V1:V3)`   Creates a new column named `TEMP`, which is a copy of column V1.
`TEMP = SELECT(2, 4, V1:V5)`   Creates three new columns named `TEMP`, `VX`, and `VY`. `TEMP` is a copy of column V2, `VX` is a copy of column V3, and `VY` is a copy of column V4.
`TEMP = SELECT(COLUMN(1,4), V6:V10)`   Creates two new columns named `TEMP` and `VX`. `TEMP` is a copy of column V6, and `VX` is a copy of column V9.
`TEMP = SELECT(COLUMN(1,4), V6[25:74]:V10)`   Creates two new columns named `TEMP` and `VX`, each containing 50 values. `TEMP` is a copy of cells 25-74 of column V6, and `VX` is a copy of cells 25-74 of column V9. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`EXTRACT`	Extracts rows given the values in a predicate column

Function	Description
MERGE	Creates a data group by horizontally concatenating the input values

SIGN

Syntax

```
SIGN(data)
```

Parameters

```
data
```

The numerical values to compute the sign of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`SIGN` tests the sign of the values in the specified data range. It returns one new column for each input column, each containing the sign of numbers in the corresponding input column. Positive one is returned for all values greater than zero; negative one is returned for all values less than zero; zero is returned for values of zero.

Examples

`TEMP = SIGN(-3)`   Creates a new column named `TEMP` containing the value -1.
`TEMP = SIGN(MERGE(3, 2, 0))`   Creates three new columns named `TEMP`, `VX`, and `VY`. `TEMP` contains the value 1, `VX` contains the value -1, and `VY` contains the value 0.
`TEMP = SIGN(V1)`   Creates a new column named `TEMP`, where each value is the sign of the contents of column `V1`.
`TEMP = SIGN(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the signs of the contents of column `V1`, the values of the `VX` column are the signs of the contents of column `V2`, and the values of the `VY` column are the signs of the contents of column `V3`.
`TEMP = SIGN(V1[10:20])`   Creates a new column named `TEMP`, where the first 11 cells contain the signs of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.

```
TEMP = SIGN(V1[10:50]:V2)
```

Creates two new columns named `TEMP` and `VX`, each with values in rows 1-41 (the other cells are empty). The values in column `TEMP` are the signs of the values in rows 10-50 of column `V1`, and the values in column `VX` are the signs of the values in rows 10-50 of column `V2`.

Related Functions

Function	Description
ABS	Computes the absolute value of the contents of the specified data range
INVERSE	Computes the negative of the contents of the specified data range

SIN

Syntax

SIN(data [, units_keyword])

Parameters

data

The numerical values to compute the sine of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

units_keyword

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

RADIAN - Performs the calculations in radians (default)

DEGREE - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

SIN calculates the sine of values in the specified data range. It returns one new column for each input column, each containing the sine of numbers in the corresponding input column.

Examples

`TEMP = SIN(PI/2) or TEMP = SIN(PI/2, 0) or TEMP = SIGN(PI/2, RADIAN)` Creates a new column named `TEMP` containing the value one.
`TEMP = SIN(V1)` Creates a new column named `TEMP`, where each value is the sine (in radians) of the contents of column `V1`.
`TEMP = SIN(V1:V3, 1) or TEMP = SIN(V1:V3, DEGREE)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the sines of the contents of column `V1`, the values of the `VX` column are the sines of the contents of column `V2`, and the values of the `VY` column are the sines of the contents of column `V3`. All values are in degrees.
`TEMP = SIN(V1[10:50]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-41 (the other cells are empty). The values in column `TEMP` are the sines of the values in rows 10-50 of column `V1`, and the values in column `VX` are the sines of the values in rows 10-50 of column `V2`. All values are in radians.

Related Functions

Function	Description
`ASIN`	Computes the arcsine of the contents of the specified data range
`COS`	Computes the cosine of the contents of the specified data range
`SINH`	Computes the hyperbolic sine of the contents of the specified data range
`TAN`	Computes the tangent of the contents of the specified data range

SINH

Syntax

`SINH(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the hyperbolic sine of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`units_keyword`

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`SINH` calculates the hyperbolic sine of the values in the specified data range. It returns one new column for each input column, each containing the hyperbolic sine of numbers in the corresponding input column. For *x* in radians, the hyperbolic sine of a number is:

$$\sinh(x) = \frac{e^x - e^{-x}}{2}$$

where *e* is the natural number, 2.7182818.

Note: If the value *x* is too large, an overflow error is returned. This occurs if *sinh* (*x*) exceeds the maximum 32-bit floating-point value.

Examples

`TEMP = SINH(1)` or `TEMP = SINH(1, 0)` or `TEMP = SINH(1, RADIAN)` Creates a new column named `TEMP` containing the value `1.18`.
`TEMP = SINH(V1)` Creates a new column named `TEMP`, where each value is the hyperbolic sine (in radians) of the contents of column `V1`.
`TEMP = SINH(V1:V3, 1)` or `TEMP = SINH(V1:V3, DEGREE)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the hyperbolic sines of the contents of column `V1`, the values of the `VX` column are the hyperbolic sines of the contents of column `V2`, and the values of the `VY` column are the hyperbolic sines of the contents of column `V3`. All values are in degrees.
`TEMP = SINH(V1[10:50]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-41 (the other cells are empty). The values in column `TEMP` are the hyperbolic sines of the values in rows 10-50 of column `V1`, and the values in column `VX` are the hyperbolic sines of the values in rows 10-50 of column `V2`. All values are in radians.

Related Functions

Function	Description
COSH	Computes the hyperbolic cosine of the contents of the specified data range
SIN	Computes the sine of the contents of the specified data range
TANH	Computes the hyperbolic tangent of the contents of the specified data range

SKEW

Syntax

SKEW(data [, keyword])

Parameters

data

The numerical values to compute the skew of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product. There must be at least three values in data.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

SKEW calculates the skew of the distribution of all the cells in the specified data range. Skewness measures the degree of asymmetry of a distribution about its mean. A positive skew value indicates a distribution with an asymmetric tail leaning towards more positive values; a negative skew indicates a distribution with an asymmetric tail leaning towards more negative values. A zero skew value indicates that the distribution is symmetrical about its mean.

Skew is calculated as follows:

$$\frac{n}{(n-1)(n-2)}\sum_{1}^{n}\left(\frac{x_i - \text{mean}}{\sigma}\right)^3$$

where n is the number of samples in the distribution, *mean* is the average, and σ is the standard deviation of the distribution. A minimum of three data values must be provided to compute skew.

Note: If the standard deviation σ =0, SKEW returns zero.

Examples

`TEMP = SKEW(3) or TEMP = SKEW(3, ALL)` Creates a new column named `TEMP` containing the value zero.
`TEMP = SKEW(MERGE(3, 7, 2))` Creates a new column named `TEMP` containing the value `0.14`.
`TEMP = SKEW(V1)` Creates a new column named `TEMP` containing a single value which is the skew of the contents of column `V1`.
`TEMP = SKEW(V1:V3)` Creates a new column named `TEMP` containing a single value which is the skew of the contents of columns `V1`, `V2`, and `V3`.
`TEMP = SKEW(V1[10:20])` Creates a new column named `TEMP` containing a single value which is the skew of the cells in rows 10-20 of column `V1`.
`TEMP = SKEW(V1[1:5]:V4)` Creates a new column named `TEMP` containing a single value which is the skew of the cells in rows 1-5 of columns `V1` through `V4`.
`TEMP = SKEW(V1:V3, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`. The single value in the `TEMP` column is the skew of the contents of column `V1`, the single value in the `VX` column is the skew of the contents of column `V2`, and the single value in the `VY` column is the skew of the contents of column `V3`.
`TEMP = SKEW(V1[1:5]:V3,COL)` Creates three new columns named `TEMP`, `VX`, and `VY`, each containing a single value. The value in column `TEMP` is the skew of the cells in rows 1-5 of column `V1`, the value in column `VX` is the skew of the cells in rows 1-5 of column `V2`, and the value in column `VY` is the skew of the cells in rows 1-5 of column `V3`.
`TEMP = SKEW(V1:V3, ROW)` Creates a new columns named `TEMP` where each cell entry is the skew of the corresponding row across columns `V1`, `V2`, and `V3`.

```
TEMP = SKEW(V1[10:50]:V3,ROW)
```

Creates a new column named TEMP, where the first 41 cells contain the skew of the values in rows 10-50 across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
AVG or MEAN	Computes the arithmetic mean or average of a range of cells
KURTOSIS	Computes the kurtosis of a range of cells
STAT	Computes the first through fourth moments of the specified data range
VARIANCE	Computes the variance of a range of cells

SLIDE_WINDOW

Syntax

SLIDE_WINDOW(width, data [, increment])

Parameters

width

The size (vertical number of rows) of the sliding window.

data

The cell range to use to slide a window over to generate data. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

increment

The number of rows to increment each time; the default is one. This must be an integer greater than zero (and less than the length of data).

Description

SLIDE_WINDOW uses a specified window size and slides it over the specified data range, creating patterns from the window data values. It returns (the number of input columns x width) columns. The sliding window begins at the top of data and covers width rows. The data values in this window (read left to right, top to bottom) are concatenated together to form a single row in the output data range. The sliding window moves down by increment rows each time.

For example, assume the columns V1 and V2 contain the following data:

10 2 20 3 30 4 40 5 50 ...

Then the expression V3=SLIDE_WINDOW(2, V1:V2) produces the following output in columns V3:V6:

`10 2 20 2 20 3 30 3 30 4 40 ...`

The first two rows create the first row of output. Then the window slides down one row to create the next pattern, and so on.

Note: This function is useful in creating patterns from time-series data.

Examples

`TEMP = SLIDE_WINDOW(1, V1)` Creates a new column named `TEMP` containing a copy of the values in column `V1`.
`TEMP = SLIDE_WINDOW(3, V1:V3)` Creates nine new columns with each row containing a three-by-three window of data from columns `V1:V3`. Rows 1-3 of the input form the first row of output, rows 2-4 form the second, etc.
`TEMP = SLIDE_WINDOW(2, V1:V3[10:20])` Creates six new columns with each row containing a three-by-two window of data from rows 10-20 of columns `V1:V3`. Rows 10-11 of the input form the first row of output, rows 11-12 form the second, etc.
`TEMP = SLIDE_WINDOW(2, MERGE(V1, V3, V5))` Creates six new columns with each row containing a three-by-two window of data from columns `V1`, `V3`, and `V5`. Rows 1-2 of the input form the first row of the output, rows 2-3 form the second, etc.
`TEMP = SLIDE_WINDOW(1, V1:V3, 2)` Creates three new columns, where the first row contains data from `V1[1]:V3`, the second row contains data from `V1[3]:V3`, the third row contains data from `V1[5]:V3`, and so on (every other row is skipped).
`TEMP = SLIDE_WINDOW(10, V1, 10)` Creates ten new columns, where the first row contains data from `V1[1:10]`, the second row contains data from `V1[11:20]`, the third row contains data from `V1[21:30]`, and so on.
`TEMP = SLIDE_WINDOW(3, V1:V2, 5)` Creates six new columns, where each row contains a two-by-three window of data from columns `V1:V2`. The first row contains data from `V1[1:3]:V2`; the second row contains data from `V1[6:8]:V2`, the third row contains data from `V1[11:13]:V2`, and so on.

Related Functions

Function	Description
GRID	Returns a grid of all possible value combinations (one per row)

SORT

Syntax

```
SORT(column [, keyword]) SORT(column, data [, keyword])
```

Parameters

`column`

In the first format (no `data` provided), this is the column of data to sort (numerical or text). This can be a constant, a column, or single-column cell range, or an expression evaluating to one of the above. This data range cannot contain more than 2^{29} values.

`data`

When this parameter is provided, it is the data to sort using `column` as the sort criteria (`data` can contain columns of numerical data and text). The `data` parameter can be a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product. This data range cannot contain more than 2^{29} rows.

`keyword`

This optional keyword determines whether the values are sorted in increasing (minimum to maximum) or decreasing (maximum to minimum) order. Select one of the following:

`ASCEND` - Sort `data` in ascending (increasing) order (default)

`DESCEND` - Sort `data` in descending (decreasing) order

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

SORT sorts the values in the specified data range (either `column` or `data` based on `column`) based on the `keyword` value (`ASCEND` or `DESCEND`). It returns one new column for each input column to be sorted. If only `column` is provided, the values in `column` will be sorted in ascending or descending order as specified by the `keyword` parameter. For a column of text, ascending order is alphabetical order (a-z) and descending is the reverse order (z-a). If `data` also is provided, it will be sorted using `column` as the sort criteria.

Note: If a single-column cell range is provided for `column`, to sort the *corresponding* rows of `data`, you must specify the same cell range for `data`. Otherwise, the default is to sort the first *n* rows of `data`. For example, to sort corresponding rows, specify: TEMP = SORT(V1[100:200], V2[100:200]:V5) Otherwise, TEMP = SORT(V1[100:200], V2:V5) is equivalent to: TEMP = SORT(V1[100:200], V2[1:101]:V5)

Examples

`TEMP = SORT(COLUMN(5, 3, 2, 4, 1))` or `TEMP = SORT(COLUMN(5, 3, 2, 4, 1), ASCEND)` Creates a new column named `TEMP` containing the values 1, 2, 3, 4, and 5.
`TEMP = SORT(COLUMN("b", "c", "a"))` Creates a new column named `TEMP` containing the strings a, b, and c.
`TEMP = SORT(10...15, DESCEND)` Creates a new column named `TEMP` containing the values 15, 14, 13, 12, 11, and 10.
`TEMP = SORT(V1)` Creates a new column named `TEMP` containing the values in column V1 sorted in ascending order.
`TEMP = SORT(V1, V1:V3)` Creates three new columns named `TEMP`, VX, and VY. The values in the TEMP column are the sorted contents of column V1 in ascending order. The values of the VX column are the corresponding contents of column V2, and the values of the VY column are the corresponding contents of column V3.
`TEMP = SORT(V1[10:20], DESCEND)` Creates a new column named `TEMP`, where the first 11 cells contain the sorted values of the cells in rows 10-20 of column V1, in descending order. The other cells in `TEMP` are empty.
`TEMP = SORT(V1[5:10], V2)` or `TEMP = SORT(V1[5:10], V2[1:6])` Creates a new column named `TEMP`, where the first 6 cells contain the values from rows 1-6 of column V2, sorted in descending order of cells 5-10 of column V1. The other cells in `TEMP` are empty.
`TEMP = SORT(V1[5:10], V2[5:10])` Creates a new column named `TEMP`, where the first 6 cells contain the sorted values of the cells in rows 5-10 of column V2 according to descending order of cells 5-10 of column V1. The other cells in `TEMP` are empty.
`TEMP = SORT(V1[10:50], V2:V3)` or `TEMP = SORT(V1[10:50], V2[1:41]:V3)` Creates two new columns named `TEMP` and VX, each with values in rows 1-41 (the other cells are empty). The values in column `TEMP` are the values from rows 1-41 of column V2, sorted according to rows 10-50 of column V1. Similarly, the values in column VX are the values from rows 1-41 of column V3, sorted according to rows 10-50 of column V1. Column V1 is sorted in ascending order.

SQRT

Syntax

SQRT(data)

Parameters

data

The numerical values to compute the square root of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

SQRT calculates the square root of the values in the specified data range. It returns one new column for each input column, each containing the positive square root of numbers in the corresponding input column.

Note: If a value in the defined data range is negative, a ??? is returned for that cell.

Examples

`TEMP = SQRT(2)`   Creates a new column named `TEMP` containing the value `1.41`.
`TEMP = SQRT(V1)`   Creates a new column named `TEMP`, where each value is the square root of the contents of column `V1`.
`TEMP = SQRT(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the square roots of the contents of column `V1`, the values of the `VX` column are the square roots of the contents of column `V2`, and the values of the `VY` column are the square roots of the contents of column `V3`.
`TEMP = SQRT(V1[10:20])`   Creates a new column named `TEMP`, where the first 11 cells contain the square roots of the values in rows 10-20 of column `V1`. The other cells in `TEMP` are empty.
`TEMP = SQRT(V1[10:50]:V2)`   Creates two new columns named `TEMP` and `VX`, each with values in rows 1-41 (the other cells are empty). The values in column `TEMP` are the square roots of the values in rows 10-50 of column `V1`, and the values in column `VX` are the square roots of the values in rows 10-50 of column `V2`.

Related Functions

Function	Description
`DIV`	Divides one specified data range by another
`MULT`	Multiplies the contents of two data ranges
`POW`	Computes a base value raised to the specified exponential power(s)

STAT

Syntax

`STAT(data [, keyword])`

Parameters

`data`

The numerical values to compute the moments for (that is, mean, standard deviation, skew, and kurtosis). This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product. There must be at least three values in `data`.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

`STAT` calculates the first four moments of the values in the specified data range. The first moment is the average. The second moment is the standard deviation. The third moment is skew, and the fourth moment is kurtosis.

The number of columns returned by the `STAT` macro function depends on `keyword` and the number of columns in `data`.

- If the keyword `ALL` is used (the default), the moments are computed over all values in `data`. One column is returned containing four values.
- If the keyword `COL` is used, moments are calculated separately for each input column. One column is returned for each input column, each containing four values.
- If the keyword `ROW` is used, the moments are calculated across each row of `data`. `STAT` returns four columns. The moments are listed across each row for each row of the input data range.

Examples

`TEMP = STAT(MERGE(1, 2, 3, 4, 5))` or `TEMP = STAT(MERGE(1, 2, 3, 4, 5), ALL)`   Creates a new column named TEMP containing the values 3, 1.58, 0, and -1.2.
`TEMP = STAT(V1)`   Creates a new column named TEMP containing the first four moments of column V1.
`TEMP = STAT(V1:V3)`   Creates a new column named TEMP containing the first four moments of columns V1, V2, and V3.
`TEMP = STAT(V1[10:20])`   Creates a new column named TEMP containing the first four moments of the cells in rows 10-20 of column V1.

`TEMP = STAT(V1[1:5]:V4)`
Creates a new column named `TEMP` containing the first four moments of the cells in rows 1-5 of columns V1 through V4.
`TEMP = STAT(V1:V3, COL)`
Creates three new columns named TEMP, VX, and VY. The four values in the TEMP column are the moments of column V1, the four values in the VX column are the moments of column V2, and the four values in the column VY are the moments of column V3.
`TEMP = STAT(V1[1:5]:V3, COL)` or `TEMP = STAT(V1[1:5]:V3[1:5], COL)`
Creates three new columns named TEMP, VX, and VY, each containing four values. The values in column TEMP are the moments of the cells in rows 1-5 of column V1, the values in column VX are the moments of the cells in rows 1-5 of column V2, and the values in column VY are the moments of the cells in rows 1-5 of column V3.
`TEMP = STAT(V1:V3, ROW)`
Creates four new columns named TEMP, VX, VY, and VZ. TEMP contains the average of each row across columns V1, V2, and V3, VX contains the standard deviation, VY contains the skew, and VZ contains the kurtosis.
`TEMP = STAT(V1[50:100]:V3],ROW)` or `TEMP = STAT(V1[50:100]:V3[50:100], ROW)`
Creates four new columns named TEMP, VX, VY, and VZ, each containing 51 rows. TEMP contains the average, VX contains the standard deviation, VY contains the skew, and VZ contains the kurtosis. The first row corresponds to row 50 across columns V1, V2, and V3. The second row corresponds to row 51, and so on.

Related Functions

Function	Description
`AVG` or `MEAN`	Computes the arithmetic mean or average of a range of cells
`KURTOSIS`	Computes the kurtosis of a range of cells
`SKEW`	Computes the skew of the distribution of a range of cells
`STDV` or `STDEV`	Computes the standard deviation of a range of cells
`VARIANCE`	Computes the variance of a range of cells

STDV or STDEV

Syntax

`STDV(data [, keyword]) STDEV(data [, keyword])`

Parameters

`data`

The numerical values to compute the standard deviation of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

```
keyword
```

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`STDV` calculates the standard deviation of all the cells in the specified data range. The standard deviation of a distribution is the square root of the variance. The standard deviation is calculated as follows:

$$\sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_j - mean)^2}$$

where the *x*'s are the samples, *n* is the number of samples, and *mean* is the average of the distribution.

Note: If the number of samples *n* = 1, `STDV` returns an error.

This macro is available in IBM Interact.

Examples

`TEMP = STDV(MERGE(1, 2, 1, 0))` or `TEMP = STDEV(MERGE(1, 2, 1, 0))` Creates a new column named `TEMP` containing the value `0.71`.
`TEMP = STDV(V1)` Creates a new column named `TEMP` containing a single value which is the standard deviation of the contents of column `V1`.

`TEMP = STDV(V1:V3)` Creates a new column named `TEMP` containing a single value which is the standard deviation of the contents of columns `V1`, `V2`, and `V3`.
`TEMP = STDV(V1[1:5]:V4)` Creates a new column named `TEMP` containing a single value which is the standard deviation of the cells in rows 1-5 of columns `V1` through `V4`.
`TEMP = STDV(V1:V3, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`. The single value in the `TEMP` column is the standard deviation of the contents of column `V1`, the single value in the `VX` column is the standard deviation of the contents of column `V2`, and the single value in the `VY` column is the standard deviation of the contents of column `V3`.
`TEMP = STDV(V1[10:50]:V3,COL)` Creates three new columns named `TEMP`, `VX`, and `VY`, each containing a single value. The value in column `TEMP` is the standard deviation of the cells in rows 10-50 of column `V1`, the value in column `VX` is the standard deviation of the cells in rows 10-50 of column `V2`, and the value in column `VY` is the standard deviation of the cells in rows 10-50 of column `V3`.
`TEMP = STDV(V1:V3, ROW)` Creates a new columns named `TEMP` where each cell entry is the standard deviation of the corresponding row across columns `V1`, `V2`, and `V3`.
`TEMP = STDV(V1[1:5]:V3,ROW)` Creates a new column named `TEMP`, where the cells in rows 1-5 contain the standard deviations of the corresponding row across columns `V1` through `V3`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`KURTOSIS`	Computes the kurtosis of a range of cells
`SKEW`	Computes the skew of the distribution of a range of cells
`STAT`	Computes the first through fourth moments of the specified data range
`VAR`	Computes the variance of a range of cells

STRING_CONCAT

Syntax

`STRING_CONCAT(string1, string2, ... stringN)`

Parameters

`string`

An ASCII text string to concatenate. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. See the *Macro Function Parameters* section of the *Using Macros* chapter for your product for the format definition of `string` (same as `data`).

Description

STRING_CONCAT concatenates the ASCII text values in the specified data ranges. It returns one new column for each input column, each containing the concatenated strings from the corresponding rows of `strings`. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: The total width of each resulting string cannot exceed 255 characters.

This macro is available in IBM Interact.

IBM Interact also supports the following syntax:

STRING_CONCAT(*string1* , *string2* , ... *stringN*)

For example, STRING_CONCAT('a', 'b', 'c', 'd') is valid.

Examples

`TEMP = STRING_CONCAT("house", "boat")`   Creates a new column named `TEMP`, which contains the ASCII text string `"houseboat "`.
`TEMP = STRING_CONCAT(V1, ".")`   Creates a new column named `TEMP`, each row containing the ASCII text string in the corresponding row of column `V1` with an appended period.
`TEMP = STRING_CONCAT(V1, V2)`   Creates a new column named `TEMP`, each row containing the containing the ASCII text string in column `V1` concatenated with the text string in column `V2`.
`TEMP = STRING_CONCAT(V1:V3, V4:V6)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the concatenated strings from the corresponding rows of column `V1` and `V4`, the values of the `VX` column are the concatenated strings from the corresponding rows of column `V2` and `V5`, and the values of the `VY` column are the concatenated strings from the corresponding rows of column `V3` and `V6`.
`TEMP = STRING_CONCAT(V1[5:10]:V2, V3:V4)`   Creates two new columns named `TEMP` and `VX`. The values in the `TEMP` column are strings from rows 5-10 of column `V1` concatenated with the rows 1-6 of column `V3`. The values in `VX` are the strings from rows 5-10 of column `V2` concatenated with the rows 1-6 of column `V4`.
`TEMP = STRING_CONCAT('a', 'b', 'c', 'd')`   Creates a new column named `TEMP`, which contains the ASCII text string `"abcd "`.

Related Functions

Function	Description
`STRING_HEAD`	Returns the first n characters of each string in the specified data range
`STRING_LENGTH`	Returns the length of each string in the specified data range

Function	Description
STRING_SEG	Returns the string segment between two specified indexes
STRING_TAIL	Returns the last n characters of each string in the specified data range

STRING_HEAD

Syntax

STRING_HEAD(num_chars, data)

Parameters

num_chars

The number of characters to returns from the beginning of each string in `data`. This must be a positive integer greater than zero.

data

ASCII text string values. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

STRING_HEAD returns the first `num_chars` characters from each string value in the specified data range. If `num_chars` is greater than the number of characters in a text string, the remaining characters are padded with the null character " \0 ".

Examples

TEMP = STRING_HEAD(3, "JAN 15, 1997")
Creates a new column named TEMP, which contains the ASCII text string " JAN ".
TEMP = STRING_HEAD(10, "Pressure")
Creates a new column named TEMP, which contains the ASCII text string " Pressure ".
TEMP = STRING_HEAD(5, V1)
Creates a new column named TEMP containing the first five characters of each string in column V1.
TEMP = STRING_HEAD(1, V1:V3)
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the first characters of the strings in the corresponding rows of column V1, the values of the VX column are the first characters of strings in corresponding rows of column V2, and the values of the VY column are the first characters of the strings in corresponding rows of column V3.

```
TEMP = STRING_HEAD(12, V4[1:50]:V6]
```

Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the first 12 characters of the strings in rows 1-50 of column V1, the values of the VX column are the first 12 characters of strings in rows 1-50 of column V2, and the values of the VY column are the first 12 characters of the strings in rows 1-50 of column V3.

Related Functions

Function	Description
STRING_CONCAT	Concatenates two text strings from the specified data ranges
STRING_LENGTH	Returns the length of each string in the specified data range
STRING_SEG	Returns the string segment between two specified indexes
STRING_TAIL	Returns the last n characters of each string in the specified data range

STRING_LENGTH

Syntax

STRING_LENGTH(data)

Parameters

data

ASCII text string values to compute the length of. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

STRING_LENGTH returns the length of each string value in the specified data range. It returns one new column for each input column, each containing the length of the corresponding text string.

Note: If STRING_LENGTH is applied to columns containing numerical data, it returns zeros.

Examples

```
TEMP = STRING_LENGTH("four")
```

Creates a new column named TEMP containing the value 4.

```
TEMP = STRING_LENGTH(4)
```

Creates a new column named TEMP containing the value 0.

`TEMP = STRING_LENGTH(V1)`   Creates a new column named `TEMP`, where each value is the length of the string in the corresponding row of column `V1`.
`TEMP = STRING_LENGTH(V1:V3)`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the lengths of the strings in the corresponding rows of column `V1`, the values of the `VX` column are the lengths of strings in corresponding rows of column `V2`, and the values of the `VY` column are the lengths of the strings in corresponding rows of column `V3`.
`TEMP = STRING_LENGTH(V4[1:50]:V6]`   Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the lengths of the strings in rows 1-50 of column `V1`, the values of the `VX` column are the lengths of strings in rows 1-50 of column `V2`, and the values of the `VY` column are the lengths of the strings in rows 1-50 of column `V3`.

Related Functions

Function	Description
`STRING_CONCAT`	Concatenates two text strings from the specified data ranges
`STRING_HEAD`	Returns the first n characters of each string in the specified data range
`STRING_SEG`	Returns the string segment between two specified indexes
`STRING_TAIL`	Returns the last n characters of each string in the specified data range

STRING_PROPER

Syntax

`STRING_PROPER(data)`

Parameters

`data`

The string value to convert.

Description

`STRING_PROPER` converts each string value in the specified data range by changing the first letter or any letter that follows a white space character or symbol (other than underscore) into uppercase, and all other characters to lowercase. It returns one new column for each input column, each containg the converted string in the corresponding input column.

Examples

`Temp = STRING_PROPER`

STRING_SEG

Syntax

```
STRING_SEG(from, to, data)
```

Parameters

`from`

The number of characters offset from the beginning of the string to start extracting the string segment from. This must be a positive integer greater than zero and less than `to`, or `STRING_SEG` returns an empty string.

`to`

The number of characters offset from the beginning of the string to stop extracting the string segment from. This must be a positive integer greater than or equal to `from`. If `to` equals `from` (and `to` is less than or equal to the length of the string), one character is returned.

`data`

ASCII text string values. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`STRING_SEG` returns the string segment between two indexes from each string value in the specified data range. If `from` is greater than the length of a string, nothing is returned. If `to` is greater than the length of a string, all characters from `from` are returned.

Examples

`TEMP = STRING_SEG(1, 6, "JAN 15, 1997")`
Creates a new column named `TEMP`, which contains the ASCII text string " Jan 15 ".
`TEMP = STRING_SEG(5, 20, "Pressure")`
Creates a new column named `TEMP`, which contains the ASCII text string " sure ".
`TEMP = STRING_SEG(5, 6, V1)`
Creates a new column named `TEMP` containing the fifth and sixth characters of each string in column V1.
`TEMP = STRING_SEG(10, 20, V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are characters 10-20 of the strings in the corresponding rows of column V1, the values of the `VX` column are the characters 10-20 of strings in corresponding rows of column V2, and the values of the `VY` column are the characters 10-20 of the strings in corresponding rows of column V3.

```
TEMP = STRING_SEG(5, 10, V4[1:50]:V6]
```

Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are characters 5-10 of the strings in rows 1-50 of column V1, the values of the VX column are characters 5-10 of strings in rows 1-50 of column V2, and the values of the VY column are characters 5-10 of the strings in rows 1-50 of column V3.

Related Functions

Function	Description
STRING_CONCAT	Concatenates two text strings from the specified data ranges
STRING_HEAD	Returns the first n characters of each string in the specified data range
STRING_LENGTH	Returns the length of each string in the specified data range
STRING_TAIL	Returns the last n characters of each string in the specified data range

STRING_TAIL

Syntax

STRING_TAIL(num_chars, data)

Parameters

num_chars

The number of characters to returns from the end of each string in data. This must be a positive integer greater than zero.

data

ASCII text string values. This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

STRING_TAIL returns the last num_chars characters from each string value in the specified data range. All string values are padded to the length of the longest string with null characters " \0 ". Then the last num_chars are returned from each string. If num_chars is greater than the number of characters in a text string, the entire text string is returned.

Examples

```
TEMP = STRING_TAIL(3, "JAN 15, 1997")
```

Creates a new column named TEMP, which contains the ASCII text string " 997 ".

`TEMP = STRING_TAIL(10, "Pressure")`
Creates a new column named `TEMP`, which contains the ASCII text string " `Pressure` ".
`TEMP = STRING_TAIL(5, V1)`
Creates a new column named `TEMP` containing the last five characters of each string in column `V1`.
`TEMP = STRING_TAIL(1, V1:V3)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the last characters of the strings in the corresponding rows of column `V1`, the values of the `VX` column are the last characters of strings in corresponding rows of column `V2`, and the values of the `VY` column are the last characters of the strings in corresponding rows of column `V3`.
`TEMP = STRING_TAIL(12, V4[1:50]:V6]`
Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the last 12 characters of the strings in rows 1-50 of column `V1`, the values of the `VX` column are the last 12 characters of strings in rows 1-50 of column `V2`, and the values of the `VY` column are the last 12 characters of the strings in rows 1-50 of column `V3`.

Related Functions

Function	Description
`STRING_CONCAT`	Concatenates two text strings from the specified data ranges
`STRING_HEAD`	Returns the first n characters of each string in the specified data range
`STRING_LENGTH`	Returns the length of each string in the specified data range
`STRING_SEG`	Returns the string segment between two specified indexes

SUBSAMPLE

Syntax

`SUBSAMPLE(num_samples, data)`

Parameters

`num_samples`

The number of samples to extract. This must be a positive integer less than the number of cells in the specified data range (that is, the `SUBSAMPLE` macro function cannot be used to *increase* the number of data points by replication).

`data`

The values to sample from. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see

the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

SUBSAMPLE evenly samples the requested number of data points from the specified data range. It returns one new column for each input column, each containing num_samples rows of the numbers uniformly extracted from the corresponding input column. The first row value and each *n*-th row value thereafter is returned, so that a total of num_samples are extracted.

Note: This macro function can be used both to increase or decrease the number of samples.

Examples

`TEMP = SUBSAMPLE(100, V1)` Creates a new column named TEMP, which contains 100 values evenly sampled from column V1.
`TEMP = SUBSAMPLE(50, V1:V3)` Creates three new columns named TEMP, VX, and VY, each containing 50 values. The values in the TEMP column are the sample values of the contents of column V1, the values of the VX column are the samples values of the contents of column V2, and the values of the VY column are the sample values of the contents of column V3.
`TEMP = SUBSAMPLE(5, V1[0:100])` Creates a new column named TEMP with values in the first five rows. Data is evenly sampled from rows 0-100 of column V1.
`TEMP = SUBSAMPLE(250, V1[1:10]:V2)` Creates two new columns named TEMP and VX, each with values in the first 250 rows. The values in column TEMP are evenly sampled from rows 1-400 of column V1, and the values in column VX are evenly sampled from rows 1-400 of column V2.

Related Functions

Function	Description
EXTRACT	Extracts rows given the values in a predicate column
SAMPLE_RANDOM	Returns column(s) of n cells, each containing a random sample from the specified data range

SUBSTITUTE

Syntax

SUBSTITUTE(data, from_table, to_table)

Parameters

data

The numerical or string values to convert. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

from_table

A column containing values you want to convert. The length of the from_table column must be the same as to_table.

to_table

A column containing the values to which to convert. The length of the to_table column must be the same as from_table.

Description

SUBSTITUTE converts values in data using the substitution pairs specified in from_table and to_table. Each value in data found in from_table is substituted with the value in the corresponding row of to_table.

SUBSTITUTE can be used to change both numerical and string values. It always returns a data range with the same dimensions as data.

Note: If you use SUBSTITUTE to convert strings to numbers or vice versa, *all* values in data must be converted. That is, every value in data must appear in from_table. Otherwise, the resulting row contains ???.

Examples

`TEMP = SUBSTITUTE(COLUMN(1,5,10),COLUMN(1), COLUMN(7))`   Creates a new column named TEMP, which contains the values 7, 5, 10 (the value 1 is replaced with 7).
`TEMP = SUBSTITUTE(COLUMN("blue", "red"), COLUMN("blue", "red"), COLUMN(0, 1))`   Creates a new column named TEMP, which contains the values 0 and 1 (the string "blue" is replaced with 0 and "red" is replaced with 1).
`TEMP = SUBSTITUTE(V1, V2, V3)`   Creates a new column named TEMP containing the values of column V1, where any values found in column V2 are replaced with the value in the corresponding row of column V3.
`TEMP = SUBSTITUTE(V1:V2, V4, V5)`   Creates two new columns named TEMP and VX, each with values from columns V1 and V2 respectively, where any values found in column V4 are replaced with the value in the corresponding row of column V5.
`TEMP = SUBSTITUTE(V1[10:20]:V2, V4, V5)`   Creates two new columns named TEMP and VX, each with values from rows 10-20 of columns V1 and V2 respectively, where any values found in column V4 are replaced with the value in the corresponding row of column V5.

Related Functions

Function	Description
EXTRACT	Extracts rows given the values in a predicate column
ISMEMBER	Tests an input range against a "table" of values, returning one if a value is contained in the table, else zero

SUBSTR or SUBSTRING

Syntax

SUBSTR(string_value, start_pos[, nchars]) or SUBSTR(string_value FROM start_pos[FOR nchars]) SUBSTRING(string_value, start_pos[, nchars]) or SUBSTRING(string_value FROM start_pos[FOR nchars])

Parameters

string_value

The string from which a substring will be taken.

start_pos

The starting character from each the substring will be extracted.

nchars

The number of characters to be extracted (must be greater than or equal to 0). If this value is not provided, all remaining characters in string_value are extracted.

Description

SUBSTR or SUBSTRING extracts nchars characters from the string, starting at start_pos. If nchars is omitted, SUBSTR and SUBSTRING extracts characters from start_pos through the end of the string. Trailing spaces are automatically truncated.

This macro is available in IBM Interact.

Important: IBM Interact supports the following formats only: SUBSTR(string_value, start_pos[, nchars]) or SUBSTRING(string_value, start_pos[, nchars])

Examples

SUBSTR SUBSTR Returns	("abcdef" FROM 1 FOR 2) ("abcdef",1,2) 'ab'
SUBSTR SUBSTR Returns	("abcdef" FROM -2 FOR 4) ("abcdef",-2,4) 'a'
SUBSTR SUBSTR Returns	("abcdef" FROM 3) ("abcdef",3) 'cdef'

SUM

Syntax

SUM(data [, keyword])

Parameters

data

The numerical values to compute the sum of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

keyword

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

ALL - Performs the computation on all cells in data (default)

COL - Performs the computation separately for each column of data

ROW - Performs the computation separately for each row of data

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {ALL | COL | ROW}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the COL keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

SUM calculates the sum of all the cells in the specified data range. It returns a single column.

Note: SUM is the same as the TOTAL macro function.

This macro is available in IBM Interact.

Examples

TEMP = SUM(3)Creates a new column named TEMP containing the value three.
TEMP = SUM((COLUMN(3, 5, 1)) Creates a new column named TEMP containing the value nine.

`TEMP = SUM(V1)`
Creates a new column named `TEMP` containing a single value which is the sum of the contents of column V1.
`TEMP = SUM(V1:V3)`
Creates a new column named `TEMP` containing a single value which is the sum of the contents of columns V1, V2, and V3.
`TEMP = SUM(V1[1:5]:V4)`
Creates a new column named `TEMP` containing a single value which is the sum of the cells in rows 10-20 of columns V1 through V4.
`TEMP = SUM(V1:V3, COL)`
Creates three new columns named `TEMP`, `VX`, and `VY`. The single value in the `TEMP` column is the sum of the contents of column V1, the single value in the VX column is the sum of the contents of column V2, and the single value in the VY column is the sum of the contents of column V3.
`TEMP = SUM(V1[1:5]:V3, COL)`
Creates three new columns named `TEMP`, `VX`, and `VY`, each containing a single value. The value in column `TEMP` is the sum of the cells in rows 1-5 of column V1, the value in column VX is the sum of the cells in rows 1-5 of column V2, and the value in column VY is the sum of the cells in rows 1-5 of column V3.
`TEMP = SUM(V1:V3, ROW)`
Creates a new columns named `TEMP`, where each cell entry is the sum of the corresponding row across columns V1, V2, and V3.
`TEMP = SUM(V1[1:5]:V3, ROW)`
Creates a new column named `TEMP`, where the cells in rows 1-5 contain the sum of the corresponding row across columns V1 through V3. The other cells in `TEMP` are empty.

Related Functions

Function	Description
AVG or MEAN	Computes the arithmetic mean or average of a range of cells
AVG_DEV	Computes the average deviation of a range of cells

TAN

Syntax

`TAN(data [, units_keyword])`

Parameters

`data`

The numerical values to compute the tangent of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

```
units_keyword
```

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

`RADIAN` - Performs the calculations in radians (default)

`DEGREE` - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by `PI` and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

TAN calculates the tangent of the values in the specified data range. It returns one new column for each input column each containing the tangent of numbers in the corresponding input column.

Examples

`TEMP = TAN(PI/4)` or `TEMP = TAN(PI/4, 0)` or `TEMP = TAN(PI/4, RADIAN)` Creates a new column named `TEMP` containing the value one.
`TEMP = TAN(V1)` Creates a new column named `TEMP`, where each value is the tangent (in radians) of the contents of column `V1`.
`TEMP = TAN(V1:V3, 1)` or `TEMP = TAN(V1:V3, DEGREE)` Creates three new columns named `TEMP`, `VX`, and `VY`. The values in the `TEMP` column are the tangents of the contents of column `V1`, the values of the `VX` column are the tangents of the contents of column `V2`, and the values of the `VY` column are the tangents of the contents of column `V3`. All values are in degrees.
`TEMP = TAN(V1[1:5]:V2)` Creates two new columns named `TEMP` and `VX`, each with values in rows 1-5 (the other cells are empty). The values in column `TEMP` are the tangents of the corresponding rows of column `V1`, and the values in column `VX` are the tangents of the corresponding rows of column `V2`. All values are in radians.

Related Functions

Function	Description
`ATAN`	Computes the arctangent of the contents of the specified data range
`COS`	Computes the cosine of the contents of the specified data range
`COT`	Computes the cotangent of the contents of the specified data range

Function	Description
SIN	Computes the sine of the contents of the specified data range
TANH	Computes the hyperbolic tangent of the contents of the specified data range

TANH

Syntax

TANH(data [, units_keyword])

Parameters

data

The numerical values to compute the hyperbolic tangent of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

units_keyword

This optional keyword determines whether the input values and results are interpreted as degrees or radians. Select one of the following:

RADIAN - Performs the calculations in radians (default)

DEGREE - Performs the calculations in degrees

If this parameter is not specified, the default is radians. (To convert from radians to degrees, divide by PI and multiply by 180.)

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Description

TANH calculates the hyperbolic tangent of the values in the specified data range. It returns one new column for each input column, each containing the hyperbolic tangent of numbers in the corresponding input column. The hyperbolic tangent of a number is calculated as follows:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

Note: If the value x is too large, an overflow error is returned. This occurs if tanh (x) exceeds the maximum 32-bit floating-point value. If cosh (x) is zero, TANH returns the maximum 32-bit floating point value.

Examples

`TEMP = TANH(PI) or TEMP = TANH(PI, 0) or TEMP = TANH(PI, RADIAN)` Creates a new column named TEMP containing the value one.
`TEMP = TANH(V1)` Creates a new column named TEMP, where each value is the hyperbolic tangent (in radians) of the contents of column V1.
`TEMP = TANH(V1:V3, 1) or TEMP = TANH(V1:V3, DEGREE)` Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the hyperbolic tangents of the contents of column V1, the values of the VX column are the hyperbolic tangents of the contents of column V2, and the values of the VY column are the hyperbolic tangents of the contents of column V3. All values are in degrees.
`TEMP = TANH(V1[1:5]:V2)` Creates two new columns named TEMP and VX, each with values in rows 1-5 (the other cells are empty). The values in column TEMP are the hyperbolic tangents of the corresponding rows of column V1, and the values in column VX are the hyperbolic tangents of the corresponding rows of column V2. All values are in radians.

Related Functions

Function	Description
ATAN	Computes the arctangent of the contents of the specified data range
COSH	Computes the hyperbolic cosine of the contents of the specified data range
COT	Computes the cotangent of the contents of the specified data range
SINH	Computes the hyperbolic sine of the contents of the specified data range
TAN	Computes the tangent of the contents of the specified data range

TO

Syntax

`begin TO end begin...end`

Parameters

`begin`

The beginning number in the range to create. This can be an integer constant value or an expression evaluating to an integer constant.

end

The end number in the range to create. This can be an integer constant value or an expression evaluating to an integer constant.

Description

`TO` creates a single column containing the integer values beginning with `begin` and ending with `end`. This macro function is used to define the time variable in recursive functions (see the `INIT` macro function).

Note: The `TO` operator can be abbreviated with three periods (`...`).

Examples

`TEMP = 1 TO 10 or TEMP = 1...10`
Creates a new column named `TEMP` containing the values 1-10.
`TEMP = 0 to -10`
Creates a new column named `TEMP` containing the values 0 to 10.

Related Functions

Function	Description
`COLUMN`	Creates new column(s), vertically concatenating the input values in each column
`MERGE`	Creates a data group by horizontally concatenating the input values

TOTAL

Syntax

`TOTAL(data [, keyword])`

Parameters

`data`

The numerical values to compute the sum of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

ROW - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`TOTAL` calculates the sum of all the cells in the specified data range.

Note: `TOTAL` is the same as the `SUM` macro function.

Examples

`TEMP = TOTAL(3)`Creates a new column named `TEMP` containing the value three.
`TEMP = TOTAL((COLUMN(3, 5, 1))` Creates a new column named `TEMP` containing the value nine.
`TEMP = TOTAL(V1)` Creates a new column named `TEMP` containing a single value which is the sum of the contents of column `V1`.
`TEMP = TOTAL(V1:V3)` Creates a new column named `TEMP` containing a single value which is the sum of the contents of columns `V1`, `V2`, and `V3`.
`TEMP = TOTAL(V1[1:5]:V4)` Creates a new column named `TEMP` containing a single value which is the sum of the cells in rows 10-20 of columns `V1` through `V4`.
`TEMP = TOTAL(V1:V3, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`. The single value in the `TEMP` column is the sum of the contents of column `V1`, the single value in the `VX` column is the sum of the contents of column `V2`, and the single value in the `VY` column is the sum of the contents of column `V3`.
`TEMP = TOTAL(V1[1:5]:V3, COL)` Creates three new columns named `TEMP`, `VX`, and `VY`, each containing a single value. The value in column `TEMP` is the sum of the cells in rows 1-5 of column `V1`, the value in column `VX` is the sum of the cells in rows 1-5 of column `V2`, and the value in column `VY` is the sum of the cells in rows 1-5 of column `V3`.
`TEMP = TOTAL(V1:V3, ROW)` Creates a new columns named `TEMP` where each cell entry is the sum of the corresponding row across columns `V1`, `V2`, and `V3`.

```
TEMP = TOTAL(V1[1:5]:V3, ROW)
```

Creates a new column named TEMP, where the cells in rows 1-5 contain the sum of the corresponding row across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
AVG or MEAN	Computes the arithmetic mean or average of a range of cells
AVG_DEV	Computes the average deviation of a range of cells

TRANSPOSE

Syntax

TRANSPOSE(data)

Parameters

data

The numerical or string values to transpose. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

TRANSPOSE transposes the specified data range. It changes the horizontal and vertical orientation of the data range (that is, the first row of data becomes the first column, the second row becomes the second column, and so on).

Note: The transposed data range must be rectangular. Any empty cells in a numerical column are replaced with zeros. Empty cells in a string column are replaced with the empty string ("").

Examples

```
TEMP = TRANSPOSE(COLUMN(1,2,3))
```

Creates three new columns named TEMP, VX, and VY, each containing a single value 1, 2, and 3.

```
TEMP = TRANSPOSE(MERGE("a","b"))
```

Creates a new column named TEMP containing a and b.

```
TEMP = TRANSPOSE(V1)
```

Creates a new column for each row in column V1. Each column contains one value, the corresponding row value of column V1.

`TEMP = TRANSPOSE(V1:V3)`
Creates one new column for each row of the longest column, V1, V2, or V3. Each column has three rows containing the transposed values of `V1:V3..`
`TEMP = TRANSPOSE(V1[10:15])`
Creates six new columns, each containing one row. The first column contains the value from `V1[10]`, the second column contains `V1[11]`, and so on.
`TEMP = TRANSPOSE(V1[50:99]:V2)`
Creates 100 new columns. Each column has two rows containing the transposed values of rows 50-99 of columns V1 and V2.

Related Functions

Function	Description
`COLUMN`	Creates new column(s), vertically concatenating the input values in each column
`MERGE`	Creates a data group by horizontally concatenating the input values

TRUNCATE

Syntax

`TRUNCATE(data)`

Parameters

`data`

The numerical values to truncate. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

`TRUNCATE` calculates the whole part of each value in the specified data range. It returns one new column for each input column, each containing the whole number (non-fractional) part of the numbers in the corresponding input column.

Note: The `FRACTION` macro function and the `TRUNCATE` macro function are complementary in that they sum to the original values.

Examples

`TEMP = TRUNCATE(4.3)`
Creates a new column named `TEMP` containing the value 4.
`TEMP = TRUNCATE(2.9)`
Creates a new column named `TEMP` containing the value -2.

`TEMP = TRUNCATE(V1)`
Creates a new column named TEMP, where each value is the fractional part of the contents of column V1.
`TEMP = TRUNCATE(V1:V3)`
Creates three new columns named TEMP, VX, and VY. The values in the TEMP column are the truncated parts of column V1, the values of the VX column are the truncated parts of column V2, and the values of the VY column are the truncated parts of column V3.
`TEMP = TRUNCATE(V1[10:20])`
Creates a new column named TEMP, where the first 11 cells contain the truncated parts of the values in rows 10-20 of column V1. The other cells in TEMP are empty.
`TEMP = TRUNCATE(V1[50:99]:V2)`
Creates two new columns named TEMP and VX, each with values in rows 1-50 (the other cells are empty). The values in column TEMP are the truncated parts of the rows of column V1, and the values in column VX are the truncated parts of the values in column V2.

Related Functions

Function	Description
`CEILING`	Computes the ceiling of each value in the specified data range
`FLOOR`	Computes the floor of each value in the specified data range
`FRACTION`	Returns the fractional part of each value in the specified data range

UPPER

Syntax

`UPPER(data)`

Parameters

`data`

The string value to be converted to uppercase.

Description

UPPER converts each string value in the specified data range to uppercase. It returns one new column for each input column, each containing the uppercase string in the corresponding input column.

This macro is available in IBM Interact.

Examples

`Temp = UPPER "gold"`
Creates a new column named Temp containing "GOLD".

`TEMP = UPPER( "jan 15, 1997")`
Creates a new column named TEMP, which contains the ASCII text string " JAN 15, 1997 ".
`TEMP = UPPER( "Pressure")`
Creates a new column named TEMP, which contains the ASCII text string "PRESSURE".
`TEMP = UPPER(V1)`
Creates a new column named TEMP containing uppercase characters of each string in column V1.

VARIANCE

Syntax

`VARIANCE(data [, keyword])`

Parameters

`data`

The numerical values to compute the variance of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`keyword`

This optional keyword determines how the computation is performed over the input data range. Select one of the following:

`ALL` - Performs the computation on all cells in `data` (default)

`COL` - Performs the computation separately for each column of `data`

`ROW` - Performs the computation separately for each row of `data`

For more details on using keywords in IBM Campaign, see "Format Specifications" on page 8.

For more details on using keywords in IBM PredictiveInsight, see "Format Specifications" on page 26.

Note: Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **IBM Campaign** because the input data is always a single column or field. The macro will always behave as if the `COL` keyword were specified. Therefore, you do not need to specify these keywords when using **IBM Campaign** .

Description

`VARIANCE` calculates the variance of all the values in the specified data range. Variance is the standard deviation squared. The variance is calculated as follows:

$$\frac{1}{n-1}\sum_{j=1}^{n}(x_j - mean)^2$$

where the *x*'s are the samples, *n* is the number of samples, and *mean* is the average of the distribution.

Note: If the number of samples $n = 1$, VARIANCE returns an error.

Examples

TEMP = VARIANCE(MERGE(3, 4, 5)) or TEMP = VARIANCE(MERGE(3, 4, 5), ALL) Creates a new column named TEMP containing the value 0.67.
TEMP = VARIANCE(MERGE(-10, 5, 10)) Creates a new column named TEMP containing the value 72.2.
TEMP = VARIANCE(V1) Creates a new column named TEMP containing a single value which is the variance of the contents of column V1.
TEMP = VARIANCE(V1:V3) Creates a new column named TEMP containing a single value which is the variance of the contents of columns V1, V2, and V3.
TEMP = VARIANCE(V1[10:20]) Creates a new column named TEMP containing a single value which is the variance of the cells in rows 10-20 of column V1.
TEMP = VARIANCE(V1[1:5]:V4) Creates a new column named TEMP containing a single value which is the variance of the cells in rows 1-5 of columns V1 through V4.
TEMP = VARIANCE(V1:V3, COL) Creates three new columns named TEMP, VX, and VY. The single value in the TEMP column is the variance of the contents of column V1, the single value in the VX column is the variance of the contents of column V2, and the single value in the VY column is the variance of the contents of column V3.
TEMP = VARIANCE(MERGE(1,4), COL) Creates two new columns named TEMP and VX, each containing the value zero.
TEMP = VARIANCE_(V1[1:5]:V3, COL) or TEMP = VARIANCE(V1[1:5]:V3[1:5], COL) Creates three new columns named TEMP, VX, and VY, each containing a single value. The value in column TEMP is the variance of the cells in rows 1-5 of column V1, the value in column VX is the variance of the cells in rows 1-5 of column V2, and the value in column VY is the variance of the cells in rows 1-5 of column V3.

`TEMP = VARIANCE(V1:V3, ROW)`
Creates a new column named TEMP where each cell entry is the variance of the corresponding row across columns V1, V2, and V3.
`TEMP = VARIANCE(V1[1:5]:V3,ROW)` or `TEMP = VARIANCE(V1[1:5]:V3[1:5], ROW)`
Creates a new column named TEMP, where the cells in rows 1-5 contain the variance of the corresponding row across columns V1 through V3. The other cells in TEMP are empty.

Related Functions

Function	Description
`KURTOSIS`	Computes the kurtosis of a range of cells
`SKEW`	Computes the skew of the distribution of a range of cells

WEEKDAY

Syntax

`WEEKDAY(data [, conversion_keyword])`

Parameters

`data`

The ASCII text dates to convert to numerical values representing days of the week (1-7). This can be ASCII text in quotes, a column of text, a cell range containing text, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

`conversion_keyword`

This optional keyword specifies how to interpret text formats for dates and times. Select one of the following:

1 – mm/dd/yy (default)

2 – dd-mmm-yy

3 – mm/dd/yy hh:mm

If this parameter is not specified, the default is 1.

Description

`WEEKDAY` converts text values in the specified data range into numerical values representing days of the week using the specified format for converting dates and times. The number 0 for Sunday, a 1 for Monday, and so on up to 6 for Saturday. If a text string cannot be parsed using the specified `conversion_keyword`, `WEEKDAY` will return an error.

Examples

`TEMP = WEEKDAY("1/1/95")` Creates a new column named `TEMP` containing the number 0 (January 1, 1995 is a Sunday).
`TEMP = WEEKDAY(V1, 2)` Creates a new column named `TEMP` containing numbers for the days of the week for the text strings in column `V1`. All text strings in column `V1` are expected to be of the form `dd-mmm-yy` (otherwise ??? 's are returned).
`TEMP = WEEKDAY(V1:V3, 3)` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains numbers representing the days of the week of text strings in column `V1`. The column `VX` contains numbers representing the days of the week of text strings in column `V2`. The column `VY` contains numbers representing the days of the week of text strings in column `V3`. All text strings in columns `V1` - `V3` are expected to be of the form `mm/dd/yy hh:mm` (otherwise ??? 's are returned).
`TEMP = WEEKDAY(V1[10:20]:V2, 10)` Creates two new columns named `TEMP` and `VX`. The column `TEMP` contains the numbers representing the days of the week of text strings in rows 10-20 of column `V1`. The column `VX` contains the numbers representing the days of the week of text strings in rows 10-20 column `V2`. All text strings are expected to be of the form `mm/dd/yy` (otherwise ??? 's are returned).

Related Functions

Function	Description
`NUMBER`	Converts ASCII text strings for times and dates to numerical values

WEEKDAYOF

Syntax

`WEEKDAYOF(date_string [, input_format])`

Parameters

`date_string`

A text representing a valid date.

`input_format`

One of the keywords in the table below, specifying the date format of `date_string`.

Description

`WEEKDAYOF` returns the day of the week as a number between 0-6 (Sunday 0, Monday 1, and so on) for the date specified by the `date_string`. If `input_format` is not provided, the default keyword `DELIM_M_D_Y` will be used.

Examples

WEEKDAYOF("08312000", MMDDYYYY) returns the number 4, since Thursday is the 4th day of the week.

Note: See "DATE" on page 73 for additional information on valid date formats.

Related Functions

Function	Description
DAYOF	Returns the day of the month as a number.
MONTHOF	Returns the month of the year as a number.
YEAROF	Returns the year as a number.

XOR

Syntax

data1 XOR data2

Parameters

data1

The non-negative integers to bitwise XOR with the values in data2. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

data2

The non-negative integer(s) to bitwise XOR with the values in data1. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. The number of columns in data2 must equal the number of columns in data1, unless data2 is a constant. For the format definition of data, see the "Macro Function Parameters" section in the chapter in this guide for your IBM product.

Description

XOR performs a bitwise XOR between the two specified data ranges. It returns a new column for each input column, each containing the corresponding column in data1 bitwise XOR-ed to the corresponding column of data2 (that is, the first column of data1 is bitwise XOR-ed to the first column of data, the second column with the second column, and so on).

If data2 is a constant, each value in data1 is bitwise XOR-ed by that value. If data2 contains one or more columns, the calculations are performed on a row-by-row basis between one column from data2 and one column from data2. The first row of data1 is bitwise XOR-ed to the first row value of data2, the second row with the second row, and so on. This row-by-row calculation produces a result for each row up to the last value of the shortest column.

Note: Precision for this macro function is limited to integer values less than 2^{24}. No negative values are allowed.

Examples

`TEMP = 3 XOR 7` Creates a new column named `TEMP` containing the value four (bitwise XOR of 011 and 111 equals 100).
`TEMP = V1 XOR 8` Creates a new column named `TEMP`, where each value is the contents of column `V1`, bitwise XOR-ed with the binary value 1000.
`TEMP = V1 XOR V1` Creates a new column named `TEMP` containing all zeros (every value XOR-ed with itself produces zero).
`TEMP = V1 XOR V2` Creates a new column named `TEMP`, where each value is the row value of column `V1` bitwise XOR-ed with the corresponding row value of column `V2`.
`TEMP = V1:V3 XOR V4:V6` Creates three new columns named `TEMP`, `VX`, and `VY`. The column `TEMP` contains the values in `V1` bitwise XOR-ed with the corresponding row values of column `V4`. The column `VX` contains the bitwise XOR-ed values from columns `V2` and `V5`. The column `VY` contains the bitwise XOR-ed values from columns `V3` and `V6`.
`TEMP = V1[10:20] XOR V2` or `TEMP = V1[10:20] XOR V2[1:11]` Creates a new column named `TEMP`, where the first 11 cells contain the bitwise XOR-ed result of the values in rows 10-20 of column `V1` by the values in rows 1-11 of column `V2`. The other cells in `TEMP` are empty.

Related Functions

Function	Description
`BIT_AND`	Computes the bitwise AND between two specified data ranges
`BIT_NOT`	Computes the bitwise NOT of the contents of the specified data range
`BIT_OR`	Computes the bitwise OR between two specified data ranges

XTAB

Syntax

`XTAB(col1, col2 [, operator_keyword, numeric_col3])`

Parameters

`col1`

The first column to produce the `xtab` from. This can be a constant, a column, a single-column cell range, or any expression evaluating to one of the above.

col2

The second column to produce the xtab from. This can be a constant, a column, a single-column cell range, or any expression evaluating to one of the above.

operator_keyword

One of the valid operator keywords (see below).

numeric_col3

The third column to produce the xtab from. This can be a constant, a column, a single-column cell range, or any expression evaluating to one of the above containing a numeric value.

Description

XTAB calculates distinct values in col1 and col2. Then it computes operator_keyword of numeric_col3 at the intersection of each col1 value with each col2 value.

The operator_keyword defaults to COUNTOF, in which case numeric_col3 is not used.

Possible operator_keywords include:

COUNTOF - returns the number of records at each intersection.

COUNTZERO - returns the number of records at each intersection for which numeric_col3 is 0.

COUNTNONZERO - returns the number of records at each intersection for which numeric_col3 is not 0.

COUNTNULL - returns the number of records at each intersection for which numeric_col3 is NULL.

MINOF - returns the smallest value of numeric_col3 at each intersection; returns missing value if there are no values at the intersection.

MAXOF - returns the largest value of numeric_col3 at each intersection; returns missing value if there are no values at the intersection.

SUMOF - returns the sum of all numeric_col3 values at each intersection.

AVGOF - returns the average of all non- NULL numeric_col3 values at each intersection.

STDEVOF - returns the standard deviation of all non- NULL numeric_col3 values at each intersection.

Note: The XTAB macro function may take a long time to compute when data is large. A "Computing..." progress bar will be displayed until the computation is complete. If you decide to cancel the computation, click on the "X" in the progress bar and delete the function definition containing the XTAB macro function.

Examples

`TEMP=XTAB(V1,V2)`
Creates a series of rows and columns that compute the count of the distinct values in the intersections of columns V1 and V2.
`TEMP=XTAB(V4,V5, SUMOF V6)`
Creates a series of rows and columns that represent the intersection of distinct values of columns V4 and V5. The measure at each intersection is the sum of the values in column V6 for the rows corresponding to that intersection.

YEAROF

Syntax

`YEAROF(date_string [, input_format])`

Parameters

`date_string`

A text representing a valid date.

`input_format`

One of the keywords in the table below, specifying the date format of `date_string`.

Description

YEAROF returns the year as a number for the date specified by the `date_string`. If `input_format` is not provided, the default keyword `DELIM_M_D_Y` will be used.

Examples

`YEAROF("31082000", DDMMYYYY)` returns the number 2000.

For additional information on valid date formats, see "DATE" on page 73.

Related Functions

Function	Description
`DAYOF`	Returns the day of the month as a number.
`MONTHOF`	Returns the month of the year as a number.
`WEEKDAYOF`	Returns the day of the week as a number.

Contacting IBM technical support

If you encounter a problem that you cannot resolve by consulting the documentation, your company's designated support contact can log a call with IBM technical support. To ensure that your problem is resolved efficiently and successfully, you collect information before you log your call.

If you are not a designated support contact at your company, contact your IBM administrator for information.

Information to gather

Before you contact IBM technical support, gather the following information:
- A brief description of the nature of your issue.
- Detailed error messages that you see when the issue occurs.
- Detailed steps to reproduce the issue.
- Related log files, session files, configuration files, and data files.
- Information about your product and system environment, which you can obtain as described in "System information."

System information

When you call IBM technical support, you might be asked to provide information about your environment.

If your problem does not prevent you from logging in, much of this information is available on the About page, which provides information about your IBM applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, you can obtain the version number of any IBM application by viewing the `version.txt` file that is located under the installation directory for each application.

Contact information for IBM technical support

For ways to contact IBM technical support, see the IBM Product Technical Support website: (http://www.ibm.com/support/entry/portal/open_service_request).

Note: To enter a support request, you must log in with an IBM account. If possible, this account must be linked to your IBM customer number. To learn more about associating your account with your IBM customer number, see **Support Resources > Entitled Software Support** on the Support Portal.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
170 Tracer Lane
Waltham, MA 02451
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Privacy Policy and Terms of Use Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer's website terms of use (e.g. privacy policy) which includes a link to IBM's and Client's data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor's computer by IBM on the Client's behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client's behalf on website visitor's devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Online Privacy Statement at: http://www.ibm.com/privacy/details/us/en section entitled "Cookies, Web Beacons and Other Technologies."

IBM ®

Printed in USA