Unica Macros for Unica Campaign and Interact V12.1.5 User's Guide

Contents

Chapter 1. Using macros in Unica Campaign

This chapter provides usage information about Unica Campaign macros. Be sure to read this chapter before you attempt to use macros in Unica Campaign.

Macro function summaries for Unica Campaign

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

Detailed reference pages for each macro function are provided in alphabetical order in Macros Reference (on page 26).

> ⚠️ **Important:** Macros can apply to both Unica Campaign and Unica Interact or to only one of these products. The macro descriptions identify the products in which they are available.

See Macro function parameters for Unica Campaign (on page 13) for information about the macro function input parameters.

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

Macro Name	Returns	Description
`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
`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 co-sine of the contents of the specified data range
ACOT	One column for each input column	Computes the arc cotan-gent 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 tan-gent of the contents of the specified data range
AVG	One column for each input column	Calculates the arith-metic mean or average of the cells in the speci-fied data range
BETWEEN	One column for each input column	Compares two values to determine whether the provided value is be-tween two other values

Macro Name	Returns	Description
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

Macro Name	Returns	Description
FAC-TORIAL	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
LOG10	One column for each input column	Computes the log base10 of the contents

Macro Name	Returns	Description
		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

Macro Name	Returns	Description
		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
`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

Macro Name	Returns	Description
`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

Macro Name	Returns	Description
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
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

Macro Name	Returns	Description
`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
`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_FOR-MAT`	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_BE-TWEEN`	One column for each input column	Returns the number of days between two dates
`DAY_FROM-NOW`	One column for each input column	Returns the number of days from the current date to the specified date
`DAY_IN-TERVAL`	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

Macro Name	Returns	Description
`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
`GROUP-BY`	One new column with a value for each row	Summarizes across multiple rows of data within a group
`GROUP-BY_-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
`ISER-ROR`	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

Macro Name	Returns	Description
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 for Unica Campaign

This section describes the parameters and usage for macro functions in Unica 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:** Unica Campaign does not support calculations on multiple fields at the same time or on a subset of row.

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 Unica 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 Unica 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."`.

Using Custom DB Function

Calling custom DB functions is supported only for ORACLE, DB2, Oracle ODBC and MSSQL. User Defined Function (UDF) is a programming construct that accepts parameters, does actions and returns the response of that action.

Steps to define DB function and create custom DB macros.

1. Define DB function The function name should not use campaign reserved keywords/ function names like below,
   - ADD_MONTHS
   - DATABASE_FUNCTION
   - DB2_DAYS
   - SUBSTR
   - SUBSTRING
   - LCLNUM
   - DAYOF
   - MONTHOF
   - YEAROF

- CURRENT_DATE
- LTRIM
- RTRIM

2. Define custom macro that uses custom DB function as defined by user

3. Macro would be visible in formula editor.

4. Use the macro.

> **Note:** Campaign will not do any syntax check. If arguments not passed as expected, will be runtime error.

Custom DB functions can be used in custom macro, derived field, expressions as part of select, extract, segement process box. For example, If you have defined custom DB function as MyCustomFunction, then to invoke the function user can write **$**MyCustomFunction (Arg1, Arg2, ...).

Execution of these function are on database only. Hence, it is faster in execution as compared to other macros.

To create and use custom macro, refere Campaign User guide.

Configure CustomMacroSchema to specify the DB schema on which the custom DB function is invoked.

> **Note:** CustomMacroSchema could be different schema than the one specified in OwnerForTableDisplay. Multiple datasources could use same CustomMacroSchema, so as to use custom db function from a common schema.

CustomMacroSchema

Configuration category

`Campaign|partitions|partition[n]| dataSources|dataSourcename`

Description

Specifies the schema used for calling user defined database functions.

The default value is blank.

For all data sources, set this property to the user of the database to which they are trying to call user defined database function.

Default value - No default value defined

Limitations:

Campaign supports UDF which returns a single value.

Campaign processes return value of custom db function as string upto 255 characters.

Chapter 2. Using macros in Unica Interact

This chapter provides usage information about Unica Interact macros. Be sure to read this chapter before you attempt to use macros in Unica Interact.

Formula Helper and Macro Expression Syntax Checking

⚠️ **Important:** The **Formula Helper** dialog box, including its Syntax Checking function, currently validates macro expressions according to what Unica Campaign supports. However, Unica Interact supports only a subset of Unica Campaign macro functionality. Therefore, you must ensure that the macros and keywords (such as date format keywords) used for Unica Interact are supported. Look for notes related to Unica Interact in the Macros Reference chapter of this guide.

Macro function summaries for Unica Interact

The tables in the following sections provide detailed descriptions of the macros that are specific to Unica Interact.

Detailed reference pages for each macro function are provided in alphabetical order in Macros Reference (on page 26).

⚠️ **Important:** Macros can apply to both Unica Campaign and Unica Interact or to only one of these products. The macro descriptions identify the products in which they are available.

Macro function parameters for Unica Interact (on page 23) provides information about the macro function input parameters for Unica Interact.

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
`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 `ST-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 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
`ST-DV 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
`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 col-	Computes the sum of a range of cells

Macro Name	Returns	Description
	umn for the `COL` keyword; one column with a value for each row for the `ROW` keyword.	

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
`INCLUDE`	One column for each input column	Determines whether a value is in-cluded in a specified list.

Date and Time Functions

Macro Name	Returns	Description
`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 in the range 1 - 31
`CURRENT_-MONTH`	One column for each input column	Returns the current month of the year as a number in the range 1 - 12
`CURRENT_-WEEKDAY`	One column for each input column	Returns the current weekday of the month as a number in the range 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_FOR-MAT`	One column for each input column	Transforms date formats

Miscellaneous Functions

Macro Name	Returns	Description
`EX-TER-NAL-CALL-OUT`	Values as defined by the custom application written with the External-Callout API	Calls a custom application written with the External-Callout API. For more information, see the Unica Interact Administrator's Guide.

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
`IN-DEXOF`	List of values for given condition on a column	Designed as a predefined macro, which can be used by passing as a parameter inside EXTERNALCALL-OUT macro. It returns a list of indexes, which satisfy the given expression.

Macro function parameters for Unica Interact

This section describes the parameters and usage for macro functions in Unica 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:** Unica Interact does not support calculations on multiple fields at the same time, or on a subset of rows.

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 Unica 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 Unica 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. Macros Reference

This section describes each available macro that is available for use in Campaign, Interact, or both. Macros are listed in alphabetical order.

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

Valid Date Format Keywords

The following table shows the keywords for valid formats, with a description and example of each.

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_M_D_YYYY`	Any delimited month, day, and 4-digit year	March 31, 1970 or 3/31/1970

Keyword	Description	Example(s)
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

Keyword	Description	Example(s)
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

Keyword	Description	Example(s)
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)

ABS macro

The ABS macro is available only in Unica Campaign.

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 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 macro

The `ACOS` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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 `VX` column are the arccosines of the contents of column `V2`, and the values of the `VY` column are the arc-cosines 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

Func- tion	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 macro

The ACOT macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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 macro

The `ADD_MONTHS` macro is available in Unica Campaign. For ORACLE, DB2 and MSSQLServer databases execution is preferred on the database instead of Campaign Server.

Syntax

```
ADD_MONTHS(months, date_string [, input_format[,DB]])
```

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.

DB

It is an optional parameter. The macro execution is preferred on the database for ORACLE/ DB2/MSSQLServer, even if DB parameter is not specified. For rest of the database types, the behaviour remains the same i.e. execution on campaign server.

Execution will implicitly happen on database if the expression contains database column.

Execution will happen on campaign server if the expression contains all non database columns. e.g. UCGF or date strings etc. To force the execution on database, include DB parameter. Please note in order to be able to specify DB parameter it's a must to use input_format too.

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.

Sample expressions and where its executed. DATE1, DATE2 are DB columns.

S.No	Expression	Execution On
1	DATE1 < ADD_MONTHS(1,DATE2)	Database
2	DATE1 < ADD_MONTHS(1,DATE2,DELIM_M_D_-Y,DB)	Database
3	ADD_MONTHS(1,'02-29-2016',DELIM_M_D_Y,DB) > DATE1	Database
4	ADD_MONTHS(24,'2012-02-29',DT_DELIM_Y_M_D) > DATE2	Campaign Server
5	ADD_MONTHS(24,'2012-02-29',DT_DELIM_Y_M_-D,DB) < DATE2	Database
6	DATE2 < ADD_MONTHS(1, DATE2)	Campaign Server
7	DATE2 < ADD_MONTHS(1, DATE2,DELIM_M_D_-Y,DB)	Database
8	ADD_MONTHS(24,'2012-02-29', DELIM_Y_M_D, DB) > DATE2	Database
9	ADD_MONTHS(24,'02-29-2020') > DATE2	Campaign Server
10	DATE1 = ADD_MONTHS(1,DATE2)	Database
11	DATE1 = ADD_MONTHS(1,DATE2,DELIM_M_D_-Y,DB)	Database
12	DATE1 != ADD_MONTHS(1,DATE2,DELIM_M_D_-Y,DB)	Database

S.No	Expression	Execution On
13	DATE1 != ADD_MONTHS(1,DATE2)	Database
14	ADD_MONTHS(3,'11NOV',DDMMM,DB) >DATE_- FORMAT(DATE1,DT_DELIM_Y_M_D,DDMMM)	Campaign Server
15	ADD_MONTHS(0,'2012-02-29',DT_DELIM_Y_M_D) < DATE1	Database
16	ADD_MONTHS(-1, DATE1, DT_DELIM_Y_M_D, DB) < DATE2	Database

Related functions

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

AND macro

The `AND` macro is available in Unica Campaign and Unica Interact.

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 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 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`).

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 macro

The `ASIN` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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 -π/2 to π/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
`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 macro

The `ATAN` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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 -π/2 to π/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

Func-tion	Description
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 macro

The `AVG` macro is available in Unica Campaign and Unica Interact.

Syntax

```
AVG(data [, keyword])
```

Parameters

`data`

The numerical values for which to compute the arithmetic mean. These values 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 product.

`keyword`

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

`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 Unica Campaign, see Format Specifications (on page 13).

> **Note:** Many macro functions take the keyword parameters `{ALL | COL | ROW}`.
> These keywords do not apply in **Unica 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 you use **Unica
> 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, then dividing the
result by the number of cells. The number of columns that are 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 that contains 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.

Examples

```
TEMP = AVG(V1)
```

Creates a column named TEMP containing a single value that is the arithmetic mean
of the contents of column V1.

```
TEMP = AVG(V1:V3)
```

Creates a column named TEMP containing a single value that is the arithmetic mean of the contents of columns V1, V2, and V3.

```
TEMP = AVG(V1[10:20])
```

Creates a column named TEMP containing a single value that is the arithmetic mean of the cells in rows 10-20 of column V1.

```
TEMP = AVG(V1[1:5]:V4)
```

Creates a column named TEMP containing a single value that 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. The single value in the VY column is the arithmetic mean of the contents of column V3.

```
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. 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 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 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 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
`SUM` or `TOTAL`	Computes the sum of a range of cells

BETWEEN macro

The `BETWEEN` macro is available only in Unica Campaign.

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

`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 macro

The `BIT_AND` macro is available in Unica Campaign and Unica Interact.

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 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 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 col-
> umn `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 macro

The `BIT_NOT` macro is available in Unica Campaign and Unica Interact.

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 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 macro

The `BIT_OR` macro is available in Unica Campaign and Unica Interact.

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 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 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 macro

The `BIT_XOR` macro is available in Unica Campaign and Unica Interact.

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 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 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
BIT_OR	Computes the bitwise OR between two specified data ranges

CEILING macro

The `CEILING` macro is available only in Unica Campaign.

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 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 ceil-

ings 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 macro

The `COLUMN` macro is available only in Unica Campaign.

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 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`.

COS macro

The `COS` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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

Func-tion	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 macro

The `COSH` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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 macro

The `COT` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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
`SIN`	Computes the sine of the contents of the specified data range

Func- tion	Description
`TAN`	Computes the tangent of the contents of the specified data range

COUNT macro

The `COUNT` macro is available only in Unica Campaign.

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 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
`SUM` or `TOTAL`	Computes the sum of a range of cells

CURRENT_DATE macro

The `CURRENT_DATE` macro is available in Unica Campaign and Unica Interact.

Syntax

`CURRENT_DATE([format])`

Parameters

`format`

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

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

Description

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

For all recommended databases, Unica 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 macros. For example:

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

In some cases, the CURRENT_DATE() macro is run on the Unica 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:** Not all formats available in Unica Campaign are supported by Unica Interact.

Note that you might have to use the DATE_FORMAT macro to adjust the CURRENT_DATE for your database type. For example, the following macro works with DB2:

```
table_name = CURRENT_DATE()-1
```

However, for Oracle, you must use the DATE_FORMAT macro as follows:

```
table_name = DATE_FORMAT(CURRENT_DATE()-1, DELIM_M_D_YYYY,'%Y-%m-%d')
```

Examples

If the date today is the 13th of September, 2015, `CURRENT_DATE()` returns "09/13/15".

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 macro

The `CURRENT_DAY` macro is available in Unica Campaign and Unica Interact.

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 Server.

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.

Function	Description
`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_JULIAN macro

The `CURRENT_JULIAN` macro is available only in Unica Campaign.

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 macro

The `CURRENT_MONTH` macro is available in Unica Campaign and Unica Interact.

Syntax

`CURRENT_MONTH()`

Description

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

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.
`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 macro

The `CURRENT_TIME` macro is available only in Unica Campaign.

Syntax

`CURRENT_TIME()`

Description

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

Date setting on your web application

To correctly display dates on your web application within current versions of Unica 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 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 Unica Campaign listener is not necessary.
7. Ensure that the web browser's language setting has `language_code`set 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 Unica Campaign, see the Unica 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.

Function	Description
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 macro

The CURRENT_WEEKDAY macro is available in Unica Campaign and Unica Interact.

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.

Examples

If today is Friday, CURRENT_WEEKDAY() returns the number 5.

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_YEAR	Returns the current year as a number.

CURRENT_YEAR macro

The `CURRENT_YEAR` macro is available in Unica Campaign and Unica Interact.

Syntax

`CURRENT_YEAR()`

Description

`CURRENT_YEAR` returns the current year as a number.

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.

DATE

Syntax

`DATE(input_date, [input_date format])`

Parameters

`date_string`

A string of text representing a valid date.

`format`

Optional, one of the keywords in the table under the "Valid Date Format Keywords," specifying the date format of `date_string`.

Description

The Unica Interact `DATE` macro converts an input date into a format-neutral integer value.

The `DATE` macro is computed as follows: DATE (X) = **365 +** the number of whole days elapsed from noon on January 1, 0001 A.D. To the DATE (X), an optional input DATE format keyword can be provided to specify how to parse the input `DATE`. If no `format` keyword is supplied, the default of `DELIM_M_D_Y` is used. For more information, see Valid Date Format keywords (on page 26) for additional information on valid date formats.

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 in the `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:

- By default, Unica Interact assumes that delimited two-digit dates are between the years of 1920-2020
- Two-digit years less than the millennium cutoff (default is 20, but can be set by the JVM parameter) 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.

Note:

> ✎   • Not all `DATE Formats` available in Unica Campaign are supported by Unica
>   Interact.
>   • For more information on two-digit years, see the "Valid Date Format keywords"
>   topic in this guide.
>   • For more information on configuring the millennium cutoff for two-digit years,
>   see the JVM Arguments section in the Unica Interact Tuning Guide.

This macro is available in Unica Interact.

Many business systems use Julian Date offsets. The result of Unica Interact DATE() macro is related to the Julian Date as follows:

Julian Date = DATE(...) + 1,721,059 + fraction of day elapsed since previous noon.

Useful DATE() values for the A.D. period include:

- January `1, 2050 A.D` returns `748,749`.
- January `1, 2000 A.D` returns `730,486`
- January `1, 1990 A.D` returns `726,834`
- January `1, 1900 A.D` returns `693,962`
- January `1, 0001 A.D` returns `365`

> ✎   **Note:** In accordance with the ISO 8601 standard and XML schema definition for
> Date and the DateTime objects, the proleptic Gregorian calendar is used to compute
> the count of days elapsed. In this calendaring system, the hypothetical year 0000
> A.D. is synonymous with 0001 B.C.

Examples

`DATE("8/31/2000")` returns the number `730,729`.

`DATE("8/31/2000",DELIM_MM_DD_YYYY)` returns the number `730,729`.

`DATE("2015-01-01",DELIM_Y_M_D)` returns the number `735,965`.

`DATE("01",DD)`, `DATE("0101",MMDDD)` and `DATE("1970-01-01",DELIM_Y_M_D)` return the number `719,529`.

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 macro

The `DATE_FORMAT` macro is available in Unica Campaign and Unica Interact.

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

✎ **Note:** Not all formats available in Unica Campaign are supported by Unica Interact.

Examples

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

✎ **Note:** See DATE *(on page 77)* 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 macro

The `DATE_JULIAN` macro is available only in Unica Campaign.

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 macro

The `DATE_STRING` macro is available only in Unica Campaign.

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)

Code	Description
`%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. The preceding zeros in the year are not truncated. For example, the year 0201 is displayed as 0201, and the year 0001 is displayed as 0001.
`%4Y`	4-digit year. The preceding zeros in the year are not truncated. For example, the year 0201 is displayed as 0201, and the year 0001 is displayed as 0001.
`%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 77)* 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 macro

The `DAY_BETWEEN` macro is available only in Unica Campaign.

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 77)* 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_INTER-VAL`	Returns the number of days between two specified dates.

DAY_FROMNOW macro

The `DAY_FROMNOW` macro is available only in Unica Campaign.

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 macro

The `DAY_INTERVAL` macro is available only in Unica Campaign.

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_BE-` `TWEEN`	Returns the number of days between two specified date strings.
`DAY_FROM-` `NOW`	Returns the number of days between the current day and a specified date.

DAYOF macro

The `DAYOF` macro is available only in Unica Campaign.

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 77)* for additional information on valid date formats.

DISTANCE macro

The DISTANCE macro is available in Unica Campaign and Unica Interact.

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. Always use a comma and a space to separate numeric values. This is necessary to accommodate languages that use a comma as a decimal separator, as shown in the second example below. It is supported to calculate the distance between multiple points. If (lat1, long1) is a list with multiple values and (lat2, long2) is also a list with multiple values, the distance of the first point in list 1 and first point in list 2 is calculated as returned as the first element in the result list, the distance of the second point in list 1 and second point in list 2 is calculated as returned as the second element in the result list, and so on, until all the elements in either list 1 and list 2 are calculated. If list 1 has only one element, while the list 2 has multiple elements, the distances between the element in list 1 and all elements in list 2 are calculated.

Examples

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

DISTANCE (18,529747, 73,839798, 18,533511, 73,8777995, KILOMETERS, 1) returns the value 4,0 kilometers.

DIV macro

The `DIV` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`/`).

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 macro

The `EQ` macro is available in Unica Campaign and Unica Interact.

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 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 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).

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 macro

The `EXP` macro is available only in Unica Campaign.

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 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 re-

sults 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 macro

The `EXTERNALCALLOUT` macro is available only in Unica Interact.

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 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 Unica Interact Administrator'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.

INDEXOF macro

The INDEXOF macro is an internal macro available only in Unica Interact. This macro is passed as a parameter in the EXTERNALCALLOUT macro. The macro gets added to EXTERNALCALLOUT with the start of server. No external configuration is required to use this macro.

Syntax

```
EXTERNALCALLOUT('indexOf',dimension field expression)
```

Parameters

'indexOf'

indexOf is passed as a predefined callout name in the EXTERNALCALLOUT macro. This parameter is mandatory and case insensitive.

Dimension field expression

An argument required by the 'indexof' callout. The users are required to pass a condition, which can involve multiple dimension table fields.

Description

'indexOf' macro provides the capability to query multiple dimension table fields. This macro returns the list of indexes satisfying the given condition for each customer. While creating an interactive flowchart, users can get records based on a given expression. The macro generates an error, if incorrect number of arguments are passed to it. In the event of any syntax errors, the error message is displayed while running the flowchart.

A syntax check does not validate these errors.

Examples

For the following dimension table **Account_details**, `EXTERNAL_CALLOUT('IndexOf',`
`Account_details.AccountId>1000 AND account_details.Status=='G')`

AccountId	Balance	Status
101	1100	G
102	800	G
103	1600	G
104	2100	G

The above expression using 'INDEXOF' returns a list containing indexes 1 and 4.

FACTORIAL macro

The `FACTORIAL` macro is available only in Unica Campaign.

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 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-1)(x-2)…(x - (x-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 macro

The `FLOOR` macro is available only in Unica Campaign.

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 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 macro

The `FORMAT` macro is available in both Unica Campaign and Unica Interact (from version 12.1.1).

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])
```

> **Note:** In case of Unica Interact, width is an optional parameter.

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.

> **Note:** In case of Unica Interact, in order to override the maximum width limit for TEXT type (255), a JVM parameter '`-Dcom.unicacorp.interact.maxStringLengthInFormatMacro`' with default value as 255 is introduced.

`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`.
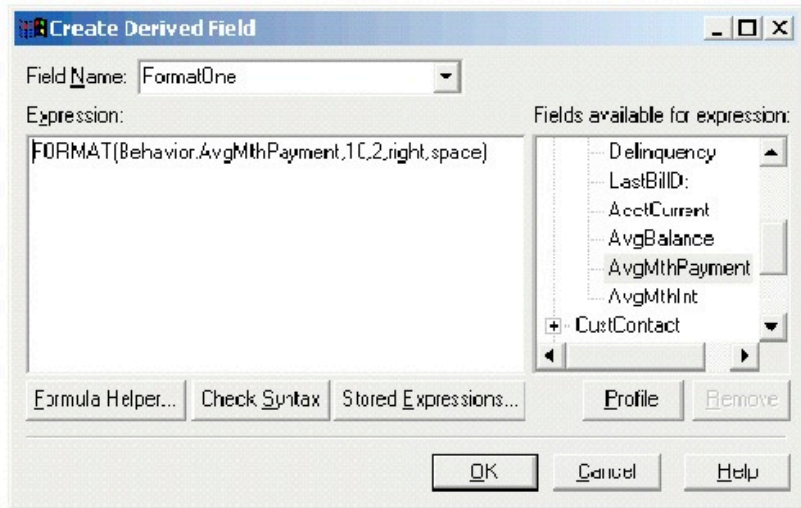
The following examples show the same field, `AvgMthPayment`, in three formats.

Unformatted:

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

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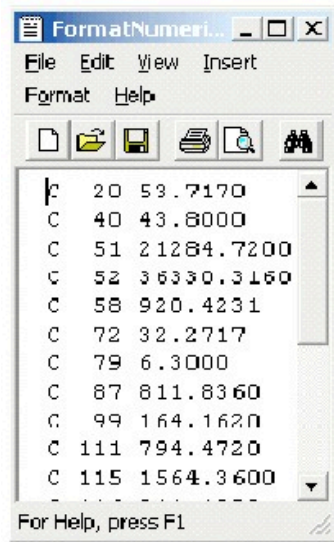
FRACTION macro

The `FRACTION` macro is available only in Unica Campaign.

Syntax

```
FRACTION(data)
```

Parameters

`data`

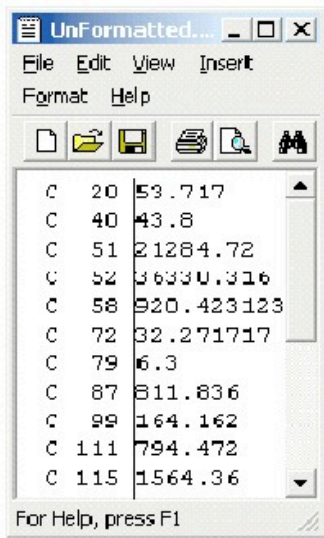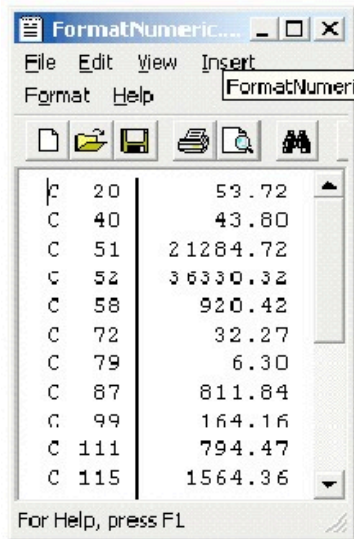The numerical values to compute the fraction of. This can be a constant value, a column, a cell range, or an expression evaluating to any of the above. For the format definition of `data`, see the "Macro Function Parameters" section in the chapter in this guide for your product.

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

GE macro

The GE macro is available in Unica Campaign and Unica Interact.

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 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 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 (`>=`).

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

GET macro

The GET macro is available in Unica Interact.

Syntax

```
GET(<dim field>, index)
```

Parameters

`index`

GET the element at the specified index. When used in Interact, it is allowed to have multiple indexes as a result of another expression. In this case, the elements in <dim field> specified in "index" are retrieved. Invalid index(es) will be skipped.

`dim field`

You get the element at the specified index from the dimensional field.

Description

This macro is listed under All built-in macros. You can make use of this macro while designing an interactive flowchart. If index is out of range, an error is returned with the expected index range.

Note: This macro supports 1-based indexing.

Example

```
GET(inttest183_interact_pftbl_null.rank,3)
```

GROUPBY macro

The `GROUPBY` macro is available only in Unica Campaign.

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):

Key-word	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).
`Median-Of`	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.

Key-word	String? Yes/ No	Description
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 macro

The `GROUPBY_WHERE` macro is available only in Unica Campaign.

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 macro *(on page 111)* 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 macro

The `GT` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`>`).

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

IF macro

The `IF` macro is available in Unica Campaign and Unica Interact.

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.

`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 for Unica Campaign (on page 13) 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 for Unica Campaign (on page 13) 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).

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).

IN macro

The `IN` macro is available in Unica Campaign and Unica Interact.

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:** 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.

> **Important:** When using `IN` in Unica 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.

INCLUDE macro

The `[NOT] INCLUDE` macro is available in Unica Interact.

Syntax

```
data1 INCLUDE data2
```

Parameters

`data1`

This can be an expression or a list of values

`data2`

This can be a constant or a list of values.

Description

`INCLUDE` compares both specified data. It returns 1, if any value from `data2` is in `data1`, else it returns 0. If `data2` is a string or numeric constant, then it compares with the values in `data1`. It returns 1 if it is present, else it returns 0.

> **Note:** The `INCLUDE` macro has a negative version, `NOT INCLUDE`. The format for this is identical to `INCLUDE`. `NOT INCLUDE` returns 1, if the values from data2 are not included in data1

Examples

```
TEMP = {'A', 'B', 'C'} INCLUDE 'A'
```

Creates a new column named TEMP containing the value 1 (since A is in the list).

```
TEMP = {'A', 'B', 'C'} INCLUDE 'D'
```

Creates a new column named TEMP containing the value 0 (since D is not in the list).

```
TEMP = {1,2,3} INCLUDE 1
```

Creates a new column named TEMP containing the value 1 (since 1 is in the list).

```
TEMP = {1,2,3} INCLUDE 4
```

Creates a new column named TEMP containing the value 0 (since 4 is not in the list).

```
TEMP = {'A', 'B', 'C'} NOT INCLUDE 'A'
```

Creates a new column named TEMP containing the value 0 (since A is in the list).

`TEMP = {1,2,3} NOT INCLUDE 4`

Creates a new column named TEMP containing the value 1 (since 4 is not in the list).

`TEMP = EligibleSegments INCLUDE "Segment1"`

Creates a new column named TEMP containing the value 1 (if Segment1 is in the `Eli-gibleSegments` list) or 0 (if it is not).

INT macro

The `INT` macro is available only in Unica Campaign.

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 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

INVERSE macro

The `INVERSE` macro is available only in Unica Campaign.

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 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 macro

The `IS` macro is available in Unica Campaign and Unica Interact.

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.

IS returns different results in Unica Interact from Unica 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.

ISERROR macro

The `ISERROR` macro is available only in Unica Campaign.

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 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.

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.

ISODD macro

The `ISODD` macro is available only in Unica Campaign.

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 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(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)

ISEVEN macro

The `ISEVEN` macro is available only in Unica Campaign.

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 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(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

Func- tion	Description
`ISODD`	Tests if input values are odd (that is, not divisible by two)

ISODD macro

The `ISODD` macro is available only in Unica Campaign.

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 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(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)

LE macro

The `LE` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`<=`).

Examples

```
TEMP = 4 LE 4 or TEMP = 4 <= 4
```

Creates a new column named `TEMP` containing the value one (since four is equal to it-self).

```
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 macro

The `LIKE` macro is available in Unica Campaign and Unica Interact.

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 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 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`.

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 macro

The `LN` or `LOG` macro is available only in Unica Campaign.

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 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

Func-tion	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 macro

The `LOG2` macro is available only in Unica Campaign.

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 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 macro

The `LOG10` macro is available only in Unica Campaign.

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 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 macro

The `LOWER` macro is available in Unica Campaign and Unica Interact.

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.

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 macro

The `LT` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`<`).

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

Func-tion	Description
LE	Returns TRUE if one data range is less than or equal to another

LTRIM macro

The LTRIM macro is available in Unica Campaign and Unica Interact.

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.

Examples

```
Temp = LTRIM " gold"
```

Creates a new string named Temp which contains

"gold".

MAX macro

The MAX macro is available in Unica Campaign and Unica Interact.

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 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 Unica Campaign, see Format Specifications (on page 13).

> ✎ **Note:** Many macro functions take the keyword parameters $\{$`ALL` $|$ `COL` $|$ `ROW`$\}$. These keywords do not apply in Unica 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 Unica Campaign.

Description

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

Examples

```
TEMP = MAX(3) or TEMP = MAX(3, ALL)
```

Creates a new column named `TEMP` containing the value three.

```
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
`MIN`	Computes the minimum of a range of cells

MEAN macro

The `MEAN` macro is available in Unica Campaign and Unica Interact.

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 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 77) for more details on using keywords.

> ✏️ **Note:** Many macro functions take the keyword parameters `{ALL | COL | ROW}`. These keywords do not apply in Unica 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 Unica 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.

Examples

```
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(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
`SUM` or `TOTAL`	Computes the sum of a range of cells

MIN macro

The MIN macro is available in Unica Campaign and Unica Interact.

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 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 77) for more details on using keywords.

📝 **Note:** Many macro functions take the keyword parameters `{ALL | COL | ROW}`. These keywords do not apply in **Unica 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 **Unica Campaign** .

Description

`MIN` calculates the minimum of all the cells in the specified data range. It returns a single column containing the minimum value.

Examples

`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
`MAX`	Computes the maximum of a range of cells
`MAX_TO_IN-DEX`	Returns the column index of the maximum value for each row of the specified column

MINUS macro

The `MINUS` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`-`).

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
SUM or TOTAL	Computes the sum of a range of cells

MOD macro

The `MOD` macro is available in Unica Campaign and Unica Interact.

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 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 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`.

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 macro

The `MONTHOF` macro is available only in Unica Campaign.

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 77)* 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 macro

The MULT macro is available in Unica Campaign and Unica Interact.

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 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 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 (`*`).

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

Func-tion	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 macro

The `NE` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`<>`).

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

Func- tion	Description
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

NOT macro

The NOT macro is available in Unica Campaign and Unica Interact.

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 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).

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

Func-tion	Description
`AND`	Computes the logical AND between two specified data ranges

Func-tion	Description
`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

NUMBER macro

The NUMBER macro is available in Unica Campaign and Unica Interact.

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 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`)

Conversion Keyword	Format	Description
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`)

Conversion Keyword	Format	Description
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

Conversion Keyword	Format	Description
		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)

Conversion Keyword	Format	Description
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

Conversion Keyword	Format	Description
		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

Conversion Keyword	Format	Description
		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`)

Conversion Keyword	Format	Description
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; $yy \geq$ `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`).

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

OR macro

The `OR` macro is available in Unica Campaign and Unica Interact.

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 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 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`).

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

Func-tion	Description
`AND`	Computes the logical AND between two specified data ranges
`NOT`	Computes the logical NOT of the contents of the specified data range

POSITION macro

The `POSITION` macro is available only in Unica Campaign.

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.

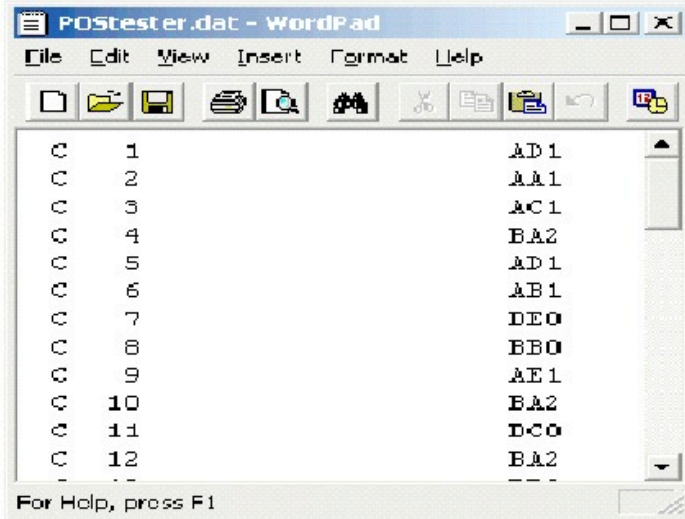> **Note:** The search is not case sensitive.

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`.
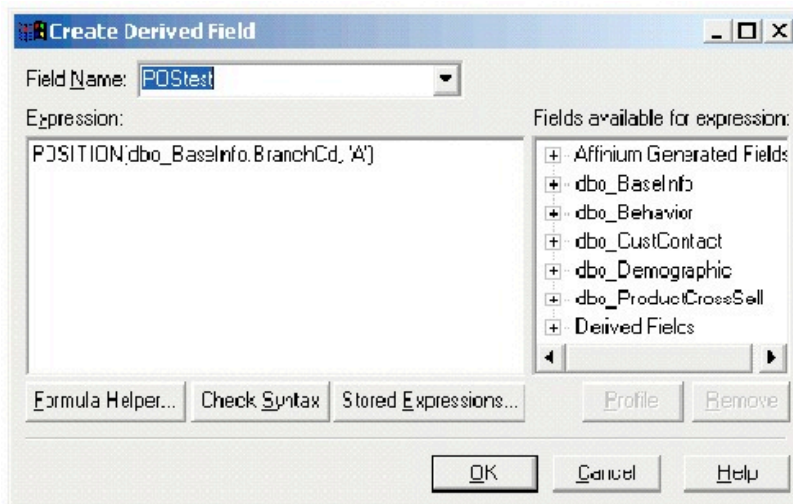
The following example shows a few rows from the table with the values from `dbo_BaseInfo.BranchCd` and `POStest` shown side-by-side.

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 macro

The `PLUS` macro is available in Unica Campaign and Unica Interact.

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 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 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 (`+`).

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

POWER macro

The `POWER` macro is available in Unica Campaign and Unica Interact.

Syntax

```
base POWER 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 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 product.

Description

`POWER` 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 `POWER` operator can be abbreviated with a circumflex (`^`). For example, `TEMP = 2^8` is equivalent to `TEMP = 2 POWER 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 POWER 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] POWER V2` or `TEMP = V1[10:20] POWER 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

Func-tion	Description
`LN10`	Computes the log base10 of the contents of the specified data range

RANDOM macro

The `RANDOM` macro is available only in Unica Campaign.

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

RANDOM_GAUSS macro

The `RANDOM_GAUSS` macro is available only in Unica Campaign.

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

ROUND macro

The `ROUND` macro is available only in Unica Campaign.

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 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

Func- tion	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
`TRUN- CATE`	Returns the non-fractional part of each value in the specified data range

ROWNUM macro

The `ROWNUM` macro is available only in Unica Campaign.

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 macro

The RTRIM macro is available in Unica Campaign and Unica Interact.

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.

Examples

```
Temp = RTRIM "gold "
```

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

SIGN macro

The `SIGN` macro is available only in Unica Campaign.

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 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(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

Func-tion	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 macro

The `SIN` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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

Func-tion	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 macro

The `SINH` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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

COUNT_DIM macro

The COUNT_DIM macro is available in Unica Interact.

Syntax

```
COUNT_DIM(<dim field>)
```

Parameter

```
dim field
```

Dimensional field.

Description

This macro is listed under All built-in macros. Use this macro while creating an interactive flowchart to get the size of the dimensional field.It can be used under the following Interactive Strategy options:

- Consider this rule eligible if the following expression is true
- Use the following expression as the marketing score

Other macros can similarly be used under Interactive Strategy.

Example

```
COUNT_DIM(inttest183_interact_pftbl_null.creditScore)
```

SORT macro

The SORT macro is available in Unica Interact.

Syntax

```
SORT(<dim field>[, <sort order>])
```

Parameter

```
dim field
```

Dimensional field.

`sort order`

Whether to sort by ascending or descending order. Valid values are ASC and DESC.

Description

This macro is listed under All built-in macros. Use this macro to sort the dimension field data in natural order while creating an interactive flowchart.

Example

```
SORT(inttest183_interact_pftbl_null.rank)
```

SQRT macro

The `SQRT` macro is available only in Unica Campaign.

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 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)

STDV or STDEV macro

The `STDV` or `STDEV` macro is available in Unica Campaign and Unica Interact.

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 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 Unica Campaign, see Format Specifications (on page 13).

> 📝 **Note:** Many macro functions take the keyword parameters `{ALL | COL | ROW}`. These keywords do not apply in **Unica 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 **Unica 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.

Examples

```
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
`VAR`	Computes the variance of a range of cells

STRING_CONCAT macro

The `STRING_CONCAT` macro is available in Unica Campaign and Unica Interact.

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. If this parameter

is a numeric or a date time value, it is converted to string using the default format in the Interact run time server. See the "Macro Function Parameters" section r for your product for the format definition of string (same as data).

> **Note:** Numeric and datetime values can be directly passed to this macro in all Interact areas. However, validation fails if used in an interactive flowchart.

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.

Unica 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
`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 macro

The `STRING_HEAD` macro is available only in Unica Campaign.

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 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 col-

umn `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_CON-CAT`	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 macro

The `STRING_LENGTH` macro is available only in Unica Campaign.

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 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_CON-CAT`	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 macro

The `STRING_PROPER` macro is available only in Unica Campaign.

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 contaning the converted string in the corresponding input column.

Examples

```
Temp = STRING_PROPER
```

STRING_SEG macro

The `STRING_SEG` macro is available only in Unica Campaign.

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 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_CON-CAT`	Concatenates two text strings from the specified data ranges
`STRING_HEAD`	Returns the first n characters of each string in the specified data range

Function	Description
`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 macro

The `STRING_TAIL` macro is available only in Unica Campaign.

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 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_CON-CAT	Concatenates two text strings from the specified data ranges
STRING_HEAD	Returns the first n characters of each string in the specified data range

Function	Description
`STRING_-LENGTH`	Returns the length of each string in the specified data range
`STRING_SEG`	Returns the string segment between two specified indexes

SUBSTR or SUBSTRING macro

The `SUBSTR` or `SUBSTRING` macro is available in Unica Campaign and Unica Interact.

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 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. To avoid syntax errors, be sure to separate numeric values with a comma and a space, as shown in the examples.

> ⚠️ **Important:** Unica 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 macro

The `SUM` macro is available in Unica Campaign and Unica Interact.

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 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 Unica Campaign, see Format Specifications (on page 13).

> ✏️ **Note:** Many macro functions take the keyword parameters {ALL | COL | ROW}. These keywords do not apply in Unica 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 Unica 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.

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

TAN macro

The `TAN` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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
`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 macro

The TANH macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

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

TOTAL macro

The `TOTAL` macro is available in Unica Campaign and Unica Interact.

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 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 Unica Campaign, see Format Specifications (on page 13).

> ✏️ **Note:** Many macro functions take the keyword parameters `{ALL | COL | ROW}`. These keywords do not apply in Unica 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 Unica 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

TRUNCATE macro

The `TRUNCATE` macro is available only in Unica Campaign.

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 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 macro

The `UPPER` macro is available in Unica Campaign and Unica Interact.

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.

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 macro

The `VARIANCE` macro is available only in Unica Campaign.

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 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 Unica Campaign, see Format Specifications (on page 13).

> 📝 **Note:** Many macro functions take the keyword parameters {`ALL` | `COL` | `ROW`}. These keywords do not apply in **Unica 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 **Unica 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:

$$\frac{1}{n-1}\sum_{j=1}^{n}(x_j - \text{mean})^2$$

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(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_(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.

WEEKDAY macro

The WEEKDAY macro is available only in Unica Campaign.

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 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 col-

umn `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 macro

The WEEKDAYOF macro is available only in Unica Campaign.

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 77)* 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 macro

The XOR macro is available in Unica Campaign and Unica Interact.

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 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 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

YEAROF macro

The YEAROF macro is available only in Unica Campaign.

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 77).

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.