Primary Functions

Tableau Reference Guide

The primary functions perform computations on the table.

Note that since the calculations are computed on the aggregated table data, field arguments for these functions must be aggregated as well, for example **SUM**(**[Profit]**).

Function Syntax	Purpose	Example
TOTAL(expression)	Returns the total for the given expression in the current partition.	**TOTAL**(**SUM**(**[Sales]**)) returns the total for the sum of sales based on the current scope and direction.
LOOKUP(expression, [offset])	Returns the value of the expression in a target row, specified as a relative offset from the current row. If the offset is **-1**, then the result will be returned for the previous value in the scope and direction.	**LOOKUP**(**SUM**(**[Profit]**), **FIRST**()+2) computes the **SUM**(**[Profit]**) in the third row of the partition.
MODEL_PERCENTILE (model_specification (optional), target_expression, predictor_expression(s))	Returns the probability (between 0 and 1) of the expected value being less than or equal to the observed mark, defined by the target expression and other predictors. This is the Posterior Predictive Distribution Function, also known as the Cumulative Distribution Function (CDF). This function is the inverse of **MODEL_QUANTILE**.	**MODEL_PERCENTILE**(**SUM**((**[Sales]**), **COUNT**(**[Orders]**)) returns the quantile of the mark for sum of sales, adjusted for count of orders.

Function Syntax	Purpose	Example
MODEL_QUANTILE (model_specification (optional), quantile, target_expression, predictor_expression(s))	Returns a target numeric value within the probable range defined by the target expression and other predictors, at a specified quantile. This is the Posterior Predictive Quantile. This function is the inverse of **MODEL_PERCENTILE**.	**MODEL_QUANTILE**(0.5, **SUM**(**[Sales]**), **COUNT**(**[Orders]**)) returns the median (0.5) predicted sum of sales, adjusted for count of orders.
PREVIOUS_VALUE (expression)	Returns the value of the expression in the previous row.	**SUM**(**[Profit]**) + **PREVIOUS_VALUE**(1) computes the running total of **SUM**(**[Profit]**).
RANK(expression, [order]) **RANK_DENSE**, **RANK_MODIFIED**, **RANK_UNIQUE**, **RANK_PERCENTILE**	Returns the standard competition rank for the current row in the partition.	**RANK**(**AVG**(**[Test Score]**))
RUNNING_SUM(expression), **RUNNING_AVG**, **RUNNING_MAX**, **RUNNING_MIN**, and **RUNNING_COUNT** are similar	Returns the running sum of the given expression, from the first row in the partition to the current row.	**RUNNING_SUM**(**SUM**(**[Profit]**)) computes the running sum of **SUM**(**[Profit]**)

Function Syntax	Purpose	Example
WINDOW_AVG(expression,[start, end]) **WINDOW_SUM**, **WINDOW_MAX**, **WINDOW_MIN**, **WINDOW_MEDIAN**, **WINDOW_COUNT**, **WINDOW_PERCENTILE**, **WINDOW_STDEV**, **WINDOW_STDEVP**, **WINDOW_VAR**, **WINDOW_VARP** are all similar	Returns the average of the expression within the window. If the optional start and end are omitted, the entire partition is used.	**WINDOW_AVG**(**SUM**(**[Profit]**), **FIRST**()+1, 0) computes the average of **SUM**(**[Profit]**) from the second row to the current row.
WINDOW_CORR(expression1, expression2, [start, end])	Returns the Pearson correlation coefficient of the two expressions within the window. If the optional start and end are omitted, the entire partition is used.	**WINDOW_CORR**(**SUM**(**[Sales]**), **SUM**(**[Profit]**)) returns a value from -1 to 1. The result is equal to 1 for an exact positive linear relationship, 0 for no linear relationship, and -1 for an exact negative linear relationship.
WINDOW_COVAR(expression1, expression2, [start, end])	**WINDOW_COVARP** is similar, but for a population, instead of a sample. Returns the sample covariance of two expressions within the window. If the optional start and end are omitted, the entire partition is used. If the two expressions are the same, a value is returned that indicates how widely the variables are distributed.	**WINDOW_COVAR**(**SUM**(**[Sales]**), **SUM**(**[Profit]**)) returns a positive number if the expressions tend to vary together, on average.

In the **RANK** functions you can optionally use 'asc' or 'desc' to specify the ranking order. The default is descending.

Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

The **RANK** functions vary on how they process identical values, such as when the ranks for the set of values (6, 9, 9, 14) are computed:

- **RANK**: Identical values are assigned an identical rank, for example 1, 2, 2, 4.
- **RANK_DENSE**: Identical values are assigned an identical rank, but no gaps are inserted into the number sequence, for example 1, 2, 2, 3.
- **RANK_MODIFIED**: Identical values are assigned an identical rank, however the first value is skipped, for example 1, 3, 3, 4.
- **RANK_UNIQUE**: Identical values are assigned different ranks, for example 1, 2, 3, 4.

In the **WINDOW** functions, the window is defined by means of offsets from the current row. Use the helper functions **FIRST**()+n and **LAST**()-n for offsets from the first or last row in the partition. If the start and end are omitted, the entire partition is used.

To calculate a population and sample standard deviation and variance, use the **WINDOW_STDEV**, **WINDOW_STDEVP**, **WINDOW_VAR**, and **WINDOW_VARP** functions.

To calculate the measure or extent of joint variability of two expressions within a window, use the **WINDOW_CORR**, **WINDOW_COVAR**, and **WINDOW_COVARP** functions.

Script Table Calculation Functions

Additional functions are available to interact directly with external service scripts, including those in R, MATLAB, and Python.

The **SCRIPT** functions vary in the data type returned, Boolean, string, integer, or real number: **SCRIPT_BOOL**, **SCRIPT_STR**, **SCRIPT_INT**, and **SCRIPT_REAL**.