Table Calculation Functions

This article introduces table calculation functions and their uses in Tableau. It also demonstrates how to create a table calculation using the calculation editor.

Why use table calculation functions

Table calculation functions allow you to perform computations on values in a table.

For example, you can calculate the percent of total an individual sale is for the year, or for several years.

Quick Table Calculations

Quick table calculations allow you to quickly apply a common table calculation to your visualization using the most typical settings for that calculation type. This article demonstrates how to apply a quick table calculation to a visualization using an example.

The following quick table calculations are available in Tableau for you to use:

* Running total (cum. Total)
* Difference
* Percent difference
* Percent of total
* Rank
* Percentile (35%,42%,55%,67%,92% etc.)
* Quartile (25%, 50%, 75%, 100%)
* Moving average
* YTD total
* Compound growth rate
* Year of year growth
* YTD growth

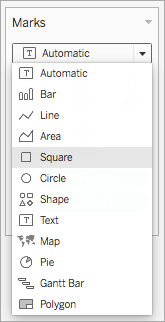
For more information, see Table Calculation Types.

Apply a quick table calculation

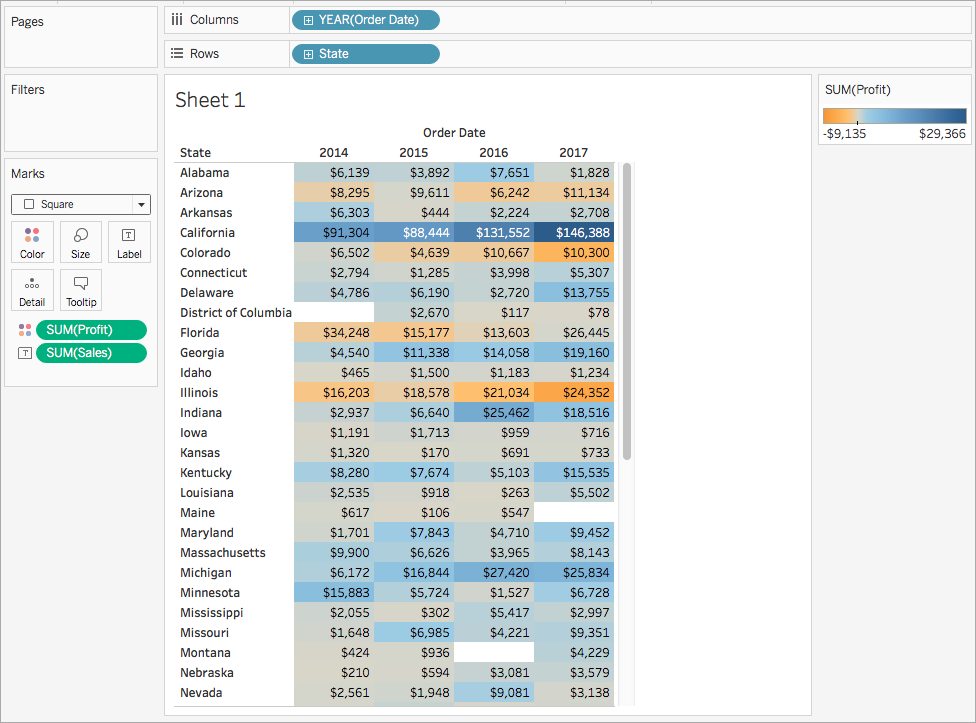
Follow the steps below to learn how to apply a table calculation to a visualization.

Step 1 : Set up the visualization

1. Open Tableau Desktop and connect to the **Sample-Superstore** data source, which comes with Tableau.
2. Navigate to a new worksheet.
3. From the **Data** pane, under Dimensions, drag **Order Date** to the **Columns**shelf.
4. From the **Data** pane, under Dimensions, drag **State**to the **Rows**shelf.
5. From the **Data** pane, under Measures, drag **Sales**to **Text**on the Marks Card.
6. From the **Data** pane, under Measures, drag **Profit**to **Color**on the Marks Card.
7. On the Marks card, click the Mark Type drop-down and select **Square**.



The visualization updates to look like this:

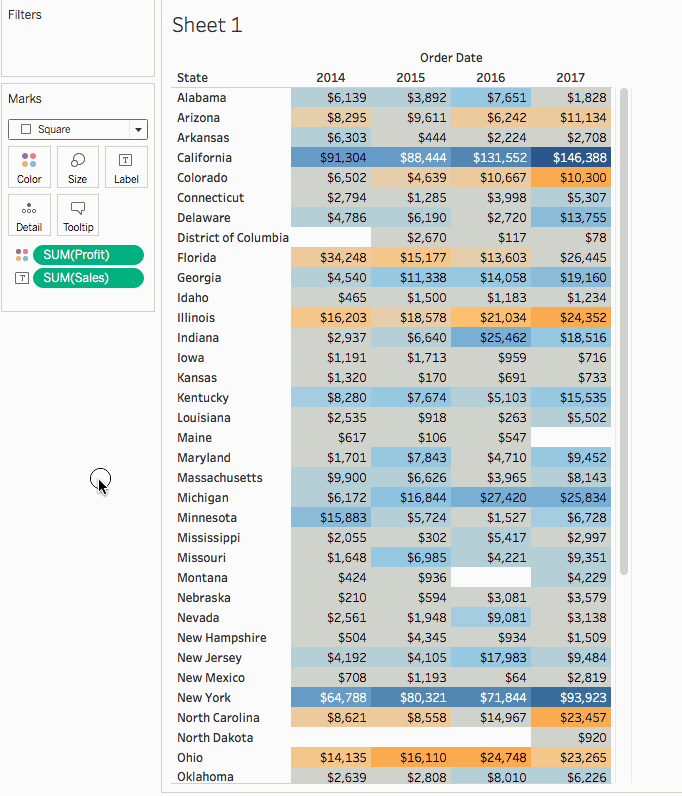


Step 2 : Apply the quick table calculation

1. On the Marks card, right-click **SUM(Profit)** and select **Quick Table Calculation** > **Moving Average**.

**Note**: You can only perform quick table calculations on measures in the view.

A delta symbol appears on the field to indicate that a quick table calculation is being applied to the field. The colors in the visualization update to show the moving average of profit across the years.

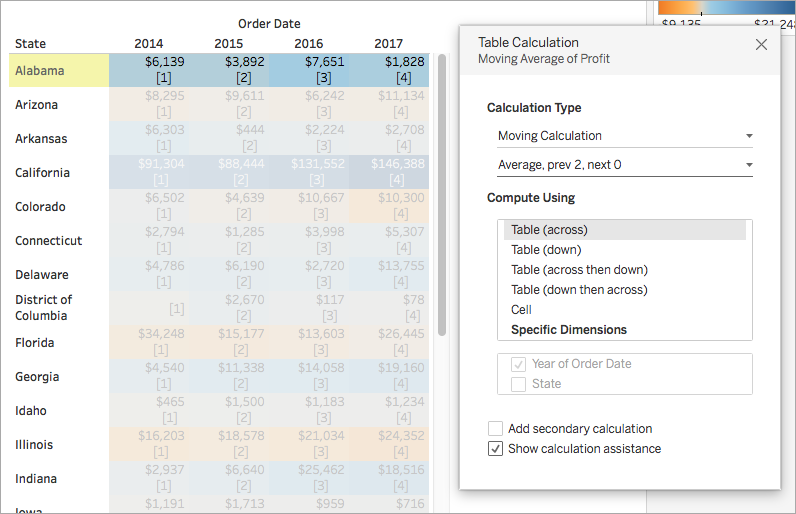


Step 3 : (Optional) Customize the quick table calculation

1. On the Marks card, right-click **Sum(Profit)** and select **Edit Table Calculation**.
2. In the dialog box that opens, you can configure the following options:
   * The calculation type
   * How to summarize the values
   * How to compute the calculation

For more information about the settings in this dialog box, see Table Calculations: Use the Visual Structure of Your View and Table Calculations: Reference Specific Dimensions in Your View .

The visualization updates as you make changes to the calculation. Highlighting and numbering are used to demonstrate how the calculation is being computed. For example, in the following image, the calculation is being computed across the table, for each row.



If this setting is changed, the visualization updates to indicate the change.

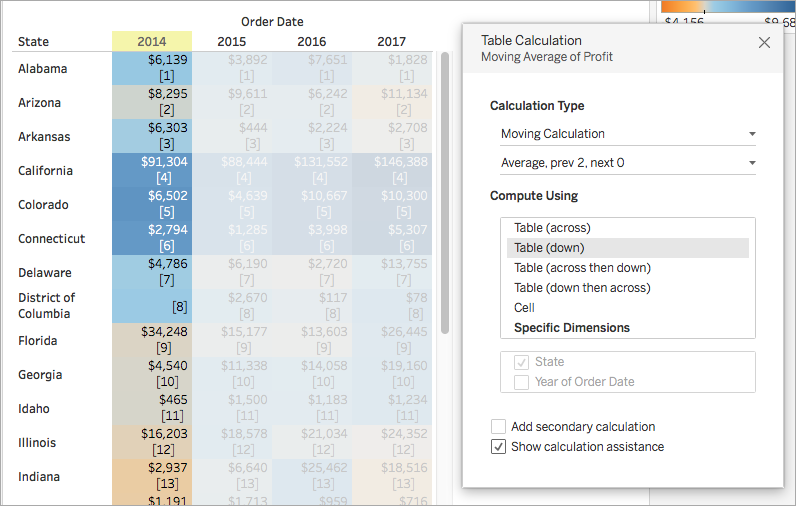
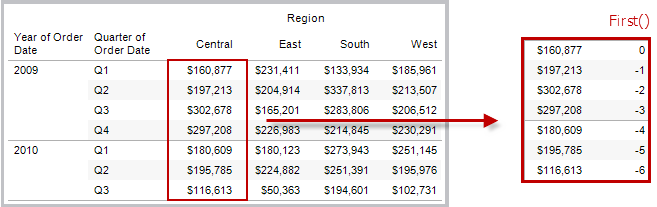


Table calculation functions available in Tableau

FIRST( )

Returns the number of rows from the current row to the first row in the partition. For example, the view below shows quarterly sales. When FIRST() is computed within the Date partition, the offset of the first row from the second row is -1.

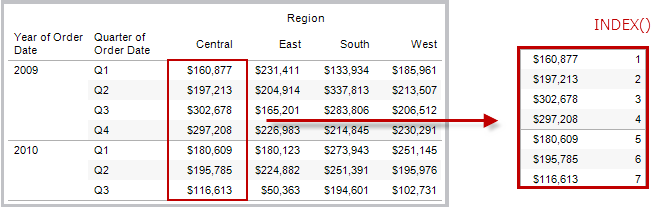


Example

When the current row index is 3, FIRST() = -2.

INDEX( )

Returns the index of the current row in the partition, without any sorting with regard to value. The first row index starts at 1. For example, the table below shows quarterly sales. When INDEX() is computed within the Date partition, the index of each row is 1, 2, 3, 4..., etc.

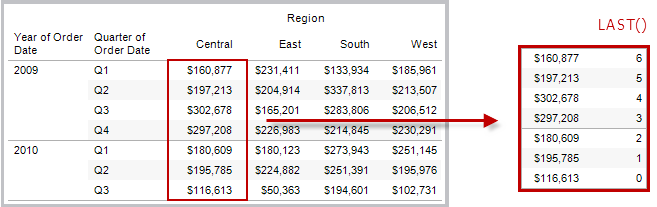


Example

For the third row in the partition, INDEX() = 3.

LAST( )

Returns the number of rows from the current row to the last row in the partition. For example, the table below shows quarterly sales. When LAST() is computed within the Date partition, the offset of the last row from the second row is 5.



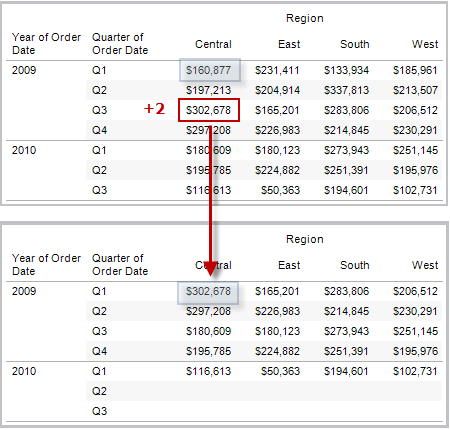
Example

When the current row index is 3 of 7, LAST() = 4.

LOOKUP(expression, [offset])

Returns the value of the expression in a target row, specified as a relative offset from the current row. Use FIRST() + n and LAST() - n as part of your offset definition for a target relative to the first/last rows in the partition. If offset is omitted, the row to compare to can be set on the field menu. This function returns NULL if the target row cannot be determined.

The view below shows quarterly sales. When LOOKUP (SUM(Sales), 2) is computed within the Date partition, each row shows the sales value from 2 quarters into the future.



Example

LOOKUP(SUM([Profit]), FIRST()+2) computes the SUM(Profit) in the third row of the partition.

PREVIOUS\_VALUE(expression)

Returns the value of this calculation in the previous row. Returns the given expression if the current row is the first row of the partition.

Example

SUM([Profit]) \* PREVIOUS\_VALUE(1) computes the running product of SUM(Profit).

RANK(expression, ['asc' | 'desc'])

Returns the standard competition rank for the current row in the partition. Identical values are assigned an identical rank. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

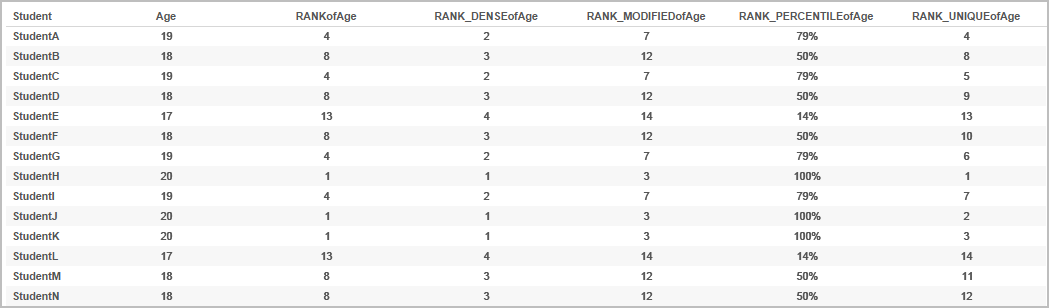
With this function, the set of values (6, 9, 9, 14) would be ranked (4, 2, 2, 1).

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.

For information on different ranking options, see Table Calculation Type: Rank.

Example

The following image shows the effect of the various ranking functions (RANK, RANK\_DENSE, RANK\_MODIFIED, RANK\_PERCENTILE, and RANK\_UNIQUE) on a set of values. The data set contains information on 14 students (StudentA through StudentN); the **Age**column shows the current age of each student (all students are between 17 and 20 years of age). The remaining columns show the effect of each rank function on the set of age values, always assuming the default order (ascending or descending) for the function.



RANK\_DENSE(expression, ['asc' | 'desc'])

Returns the dense rank for the current row in the partition. Identical values are assigned an identical rank, but no gaps are inserted into the number sequence. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (3, 2, 2, 1).

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.

For information on different ranking options, see Table Calculation Type: Rank.

RANK\_MODIFIED(expression, ['asc' | 'desc'])

Returns the modified competition rank for the current row in the partition. Identical values are assigned an identical rank. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (4, 3, 3, 1).

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.

For information on different ranking options, see Table Calculation Type: Rank.

RANK\_PERCENTILE(expression, ['asc' | 'desc'])

Returns the percentile rank for the current row in the partition. Use the optional 'asc' | 'desc' argument to specify ascending or descending order. The default is ascending.

With this function, the set of values (6, 9, 9, 14) would be ranked (0.25, 0.75, 0.75, 1.00).

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.

For information on different ranking options, see Table Calculation Type: Rank.

RANK\_UNIQUE(expression, ['asc' | 'desc'])

Returns the unique rank for the current row in the partition. Identical values are assigned different ranks. Use the optional 'asc' | 'desc'argument to specify ascending or descending order. The default is descending.

With this function, the set of values (6, 9, 9, 14) would be ranked (4, 2, 3, 1).

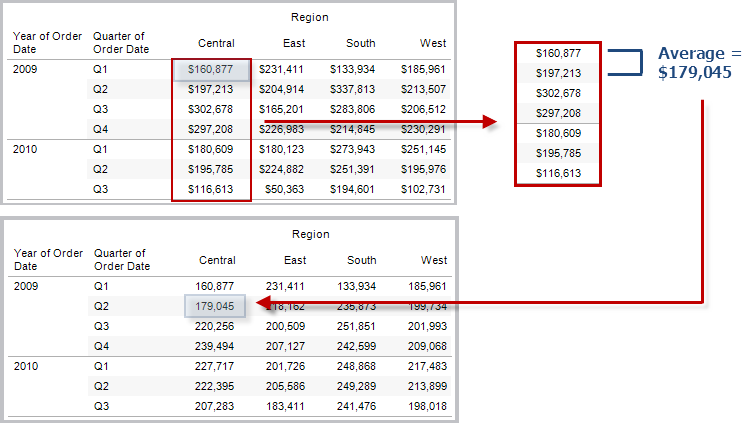
Nulls are ignored in ranking functions. They are not numbered and they do not count against the total number of records in percentile rank calculations.

For information on different ranking options, see Table Calculation Type: Rank.

RUNNING\_AVG(expression)

Returns the running average of the given expression, from the first row in the partition to the current row.

The view below shows quarterly sales. When RUNNING\_AVG(SUM([Sales]) is computed within the Date partition, the result is a running average of the sales values for each quarter.



Example

RUNNING\_AVG(SUM([Profit])) computes the running average of SUM(Profit).

RUNNING\_COUNT(expression)

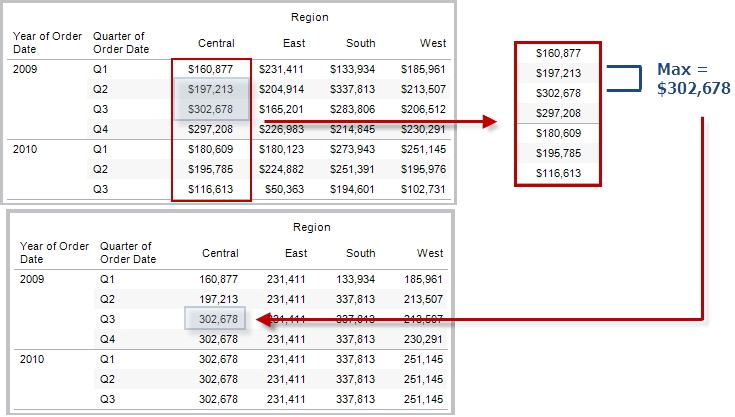
Returns the running count of the given expression, from the first row in the partition to the current row.

Example

RUNNING\_COUNT(SUM([Profit])) computes the running count of SUM(Profit).

RUNNING\_MAX(expression)

Returns the running maximum of the given expression, from the first row in the partition to the current row.

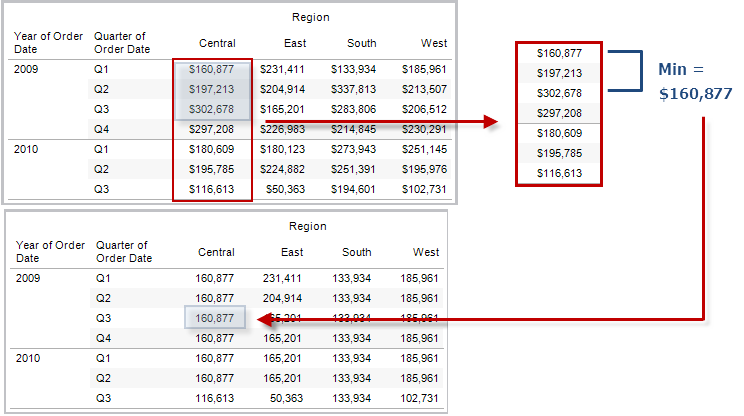


Example

RUNNING\_MAX(SUM([Profit])) computes the running maximum of SUM(Profit).

RUNNING\_MIN(expression)

Returns the running minimum of the given expression, from the first row in the partition to the current row.

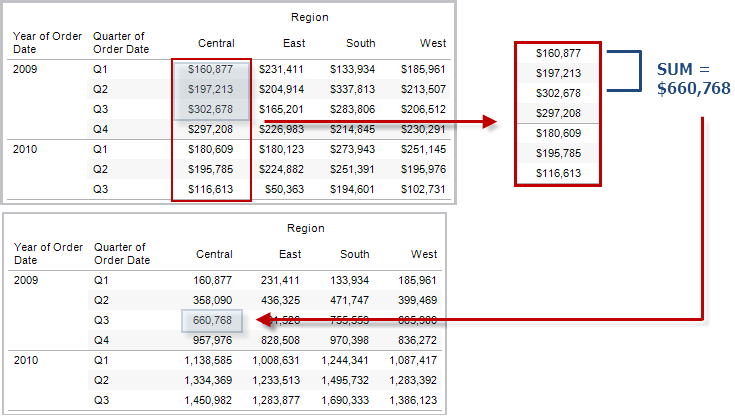


Example

RUNNING\_MIN(SUM([Profit])) computes the running minimum of SUM(Profit).

RUNNING\_SUM(expression)

Returns the running sum of the given expression, from the first row in the partition to the current row.

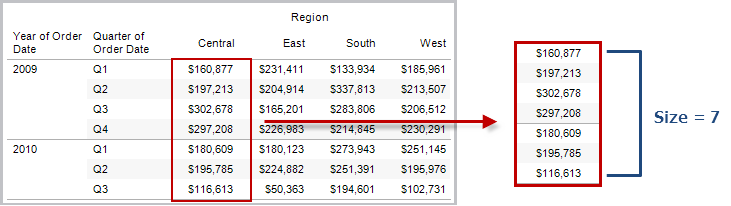


Example

RUNNING\_SUM(SUM([Profit])) computes the running sum of SUM(Profit)

SIZE()

Returns the number of rows in the partition. For example, the view below shows quarterly sales. Within the Date partition, there are seven rows so the Size() of the Date partition is 7.



Example

SIZE() = 5 when the current partition contains five rows.

SCRIPT\_BOOL

Returns a Boolean result from the specified expression. The expression is passed directly to a running external service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.).

In Python expressions, use \_arg*n* (with a leading underscore).

Examples

In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT\_BOOL("is.finite(.arg1)", SUM([Profit]))

The next example returns True for store IDs in Washington state, and False otherwise. This example could be the definition for a calculated field titled IsStoreInWA.

SCRIPT\_BOOL('grepl(".\*\_WA", .arg1, perl=TRUE)',ATTR([Store ID]))

A command for Python would take this form:

SCRIPT\_BOOL("return map(lambda x : x > 0, \_arg1)", SUM([Profit]))

SCRIPT\_INT

Returns an integer result from the specified expression. The expression is passed directly to a running external service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use \_arg*n* (with a leading underscore).

Examples

In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT\_INT("is.finite(.arg1)", SUM([Profit]))

In the next example, k-means clustering is used to create three clusters:

SCRIPT\_INT('result <- kmeans(data.frame(.arg1,.arg2,.arg3,.arg4), 3);result$cluster;', SUM([Petal length]), SUM([Petal width]),SUM([Sepal length]),SUM([Sepal width]))

A command for Python would take this form:

SCRIPT\_INT("return map(lambda x : int(x \* 5), \_arg1)", SUM([Profit]))

SCRIPT\_REAL

Returns a real result from the specified expression. The expression is passed directly to a running external service instance. In

R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use \_arg*n* (with a leading underscore).

Examples

In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT\_REAL("is.finite(.arg1)", SUM([Profit]))

The next example converts temperature values from Celsius to Fahrenheit.

SCRIPT\_REAL('library(udunits2);ud.convert(.arg1, "celsius", "degree\_fahrenheit")',AVG([Temperature]))

A command for Python would take this form:

SCRIPT\_REAL("return map(lambda x : x \* 0.5, \_arg1)", SUM([Profit]))

SCRIPT\_STR

Returns a string result from the specified expression. The expression is passed directly to a running external service instance.

In R expressions, use .arg*n* (with a leading period) to reference parameters (.arg1, .arg2, etc.)

In Python expressions, use \_arg*n* (with a leading underscore).

Examples

In this R example, .arg1 is equal to SUM([Profit]):

SCRIPT\_STR("is.finite(.arg1)", SUM([Profit]))

The next example extracts a state abbreviation from a more complicated string (in the original form 13XSL\_CA, A13\_WA):

SCRIPT\_STR('gsub(".\*\_", "", .arg1)',ATTR([Store ID]))

A command for Python would take this form:

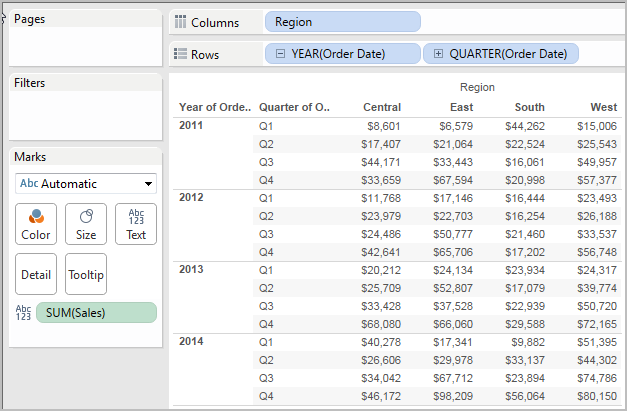
SCRIPT\_STR("return map(lambda x : x[:2], \_arg1)", ATTR([Region]))

TOTAL(expression)

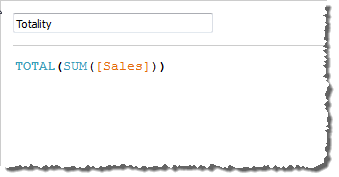
Returns the total for the given expression in a table calculation partition.

Example

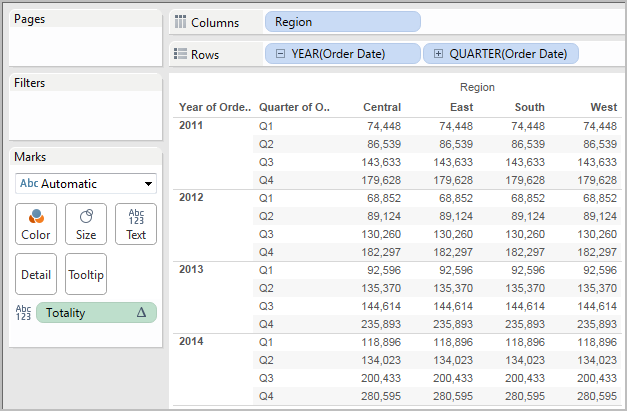
Assume you are starting with this view:



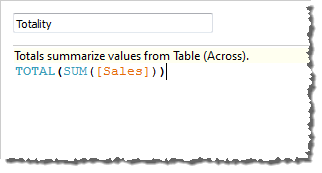
You open the calculation editor and create a new field which you name **Totality**:



You then drop **Totality** on Text, to replace **SUM(Sales)**. Your view changes such that it sums values based on the default **Compute Using**value:



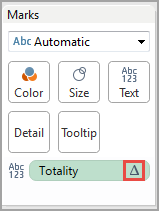
This raises the question, What is the default **Compute Using** value? If you right-click (Control-click on a Mac) **Totality** in the Data pane and choose **Edit**, there is now an additional bit of information available:



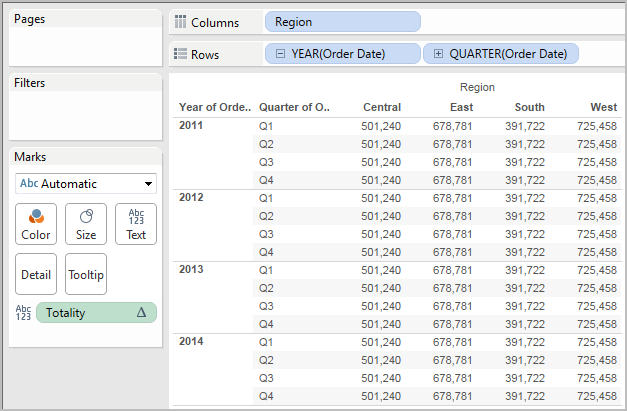
The default **Compute Using** value is **Table (Across)**. The result is that **Totality** is summing the values across each row of your table. Thus, the value that you see across each row is the sum of the values from the original version of the table.

The values in the 2011/Q1 row in the original table were $8601, $6579, $44262, and $15006. The values in the table after **Totality** replaces **SUM(Sales)** are all $74,448, which is the sum of the four original values.

Notice the triangle next to Totality after you drop it on Text:



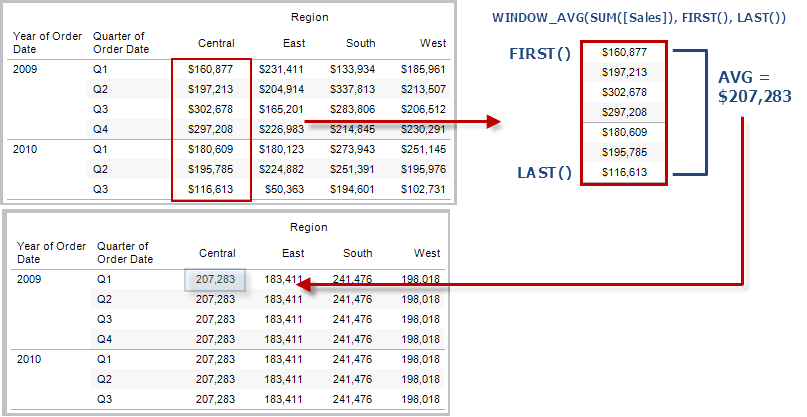
This indicates that this field is using a table calculation. You can right-click the field and choose **Edit Table Calculation** to redirect your function to a different **Compute Using** value. For example, you could set it to **Table (Down)**. In that case, your table would look like this:



WINDOW\_AVG(expression, [start, end])

Returns the average of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

For example, the view below shows quarterly sales. A window average within the Date partition returns the average sales across all dates.



Example

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 two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If start and end are omitted, the entire partition is used.

The Pearson correlation measures the linear relationship between two variables. Results range from -1 to +1 inclusive, where 1 denotes an exact positive linear relationship, as when a positive change in one variable implies a positive change of corresponding magnitude in the other, 0 denotes no linear relationship between the variance, and −1 is an exact negative relationship.

There is an equivalent aggregation fuction: CORR. See Tableau Functions (Alphabetical).

Example

The following formula returns the Pearson correlation of **SUM(Profit)** and **SUM(Sales)** from the five previous rows to the current row.

WINDOW\_CORR(SUM[Profit]), SUM([Sales]), -5, 0)

WINDOW\_COUNT(expression, [start, end])

Returns the count of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

Example

WINDOW\_COUNT(SUM([Profit]), FIRST()+1, 0) computes the count of SUM(Profit) from the second row to the current row

WINDOW\_COVAR(expression1, expression2, [start, end])

Returns the *sample covariance* of two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If the start and end arguments are omitted, the window is the entire partition.

Sample covariance uses the number of non-null data points n - 1 to normalize the covariance calculation, rather than n, which is used by the population covariance (with the WINDOW\_COVARP function). Sample covariance is the appropriate choice when the data is a random sample that is being used to estimate the covariance for a larger population.

There is an equivalent aggregation fuction: COVAR. See Tableau Functions (Alphabetical).

Example

The following formula returns the sample covariance of **SUM(Profit)** and **SUM(Sales)** from the two previous rows to the current row.

WINDOW\_COVAR(SUM([Profit]), SUM([Sales]), -2, 0)

WINDOW\_COVARP(expression1, expression2, [start, end])

Returns the *population covariance* of two expressions within the window. The window is defined as offsets from the current row. Use FIRST()+n and LAST()-n for offsets from the first or last row in the partition. If start and end are omitted, the entire partition is used.

Population covariance is sample covariance multiplied by (n-1)/n, where n is the total number of non-null data points. Population covariance is the appropriate choice when there is data available for all items of interest as opposed to when there is only a random subset of items, in which case sample covariance (with the WINDOW\_COVAR function) is appropriate.

There is an equivalent aggregation fuction: COVARP. Tableau Functions (Alphabetical).

Example

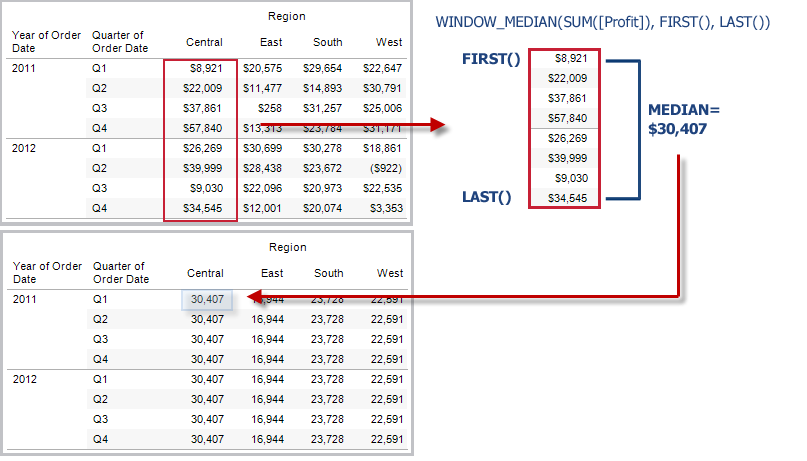
The following formula returns the population covariance of **SUM(Profit)** and **SUM(Sales)** from the two previous rows to the current row.

WINDOW\_COVARP(SUM([Profit]), SUM([Sales]), -2, 0)

WINDOW\_MEDIAN(expression, [start, end])

Returns the median of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

For example, the view below shows quarterly profit. A window median within the Date partition returns the median profit across all dates.



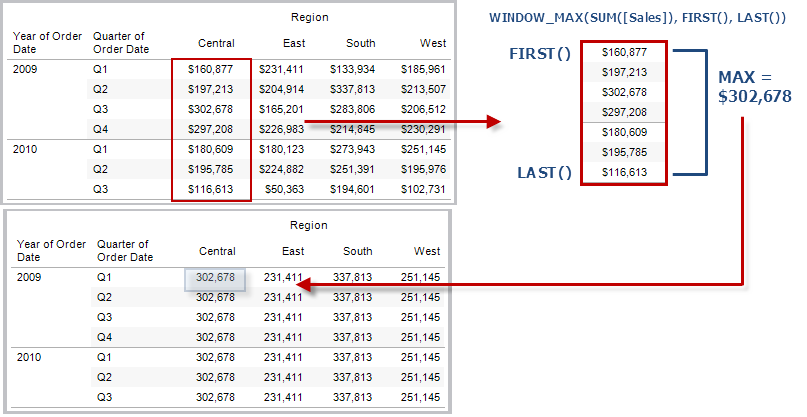
Example

WINDOW\_MEDIAN(SUM([Profit]), FIRST()+1, 0) computes the median of SUM(Profit) from the second row to the current row.

WINDOW\_MAX(expression, [start, end])

Returns the maximum of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

For example, the view below shows quarterly sales. A window maximum within the Date partition returns the maximum sales across all dates.



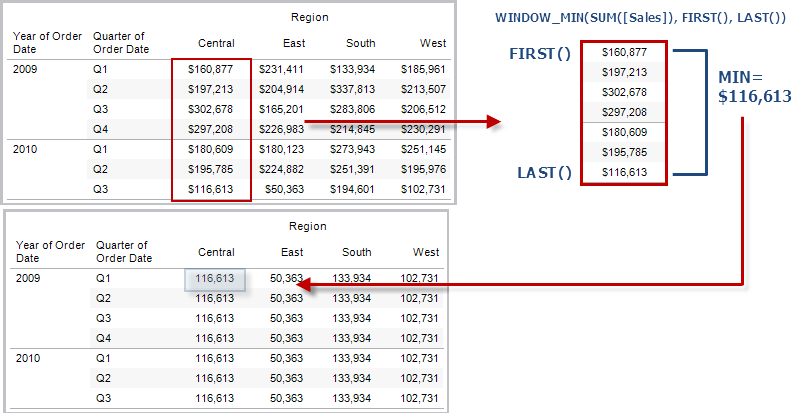
Example

WINDOW\_MAX(SUM([Profit]), FIRST()+1, 0) computes the maximum of SUM(Profit) from the second row to the current row.

WINDOW\_MIN(expression, [start, end])

Returns the minimum of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

For example, the view below shows quarterly sales. A window minimum within the Date partition returns the minimum sales across all dates.



Example

WINDOW\_MIN(SUM([Profit]), FIRST()+1, 0) computes the minimum of SUM(Profit) from the second row to the current row.

WINDOW\_PERCENTILE(expression, number, [start, end])

Returns the value corresponding to the specified percentile within the window. The window is defined by means of offsets from the current row. Use 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.

Example

WINDOW\_PERCENTILE(SUM([Profit]), 0.75, -2, 0) returns the 75th percentile for SUM(Profit) from the two previous rows to the current row.

WINDOW\_STDEV(expression, [start, end])

Returns the sample standard deviation of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

Example

WINDOW\_STDEV(SUM([Profit]), FIRST()+1, 0) computes the standard deviation of SUM(Profit) from the second row to the current row.

WINDOW\_STDEVP(expression, [start, end])

Returns the biased standard deviation of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

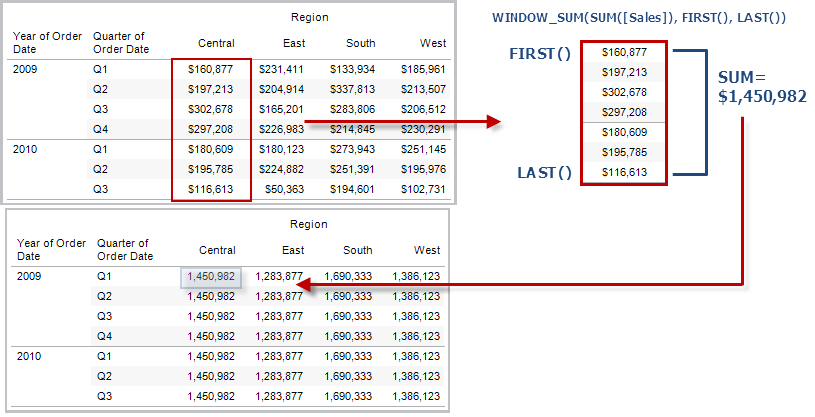
Example

WINDOW\_STDEVP(SUM([Profit]), FIRST()+1, 0) computes the standard deviation of SUM(Profit) from the second row to the current row.

WINDOW\_SUM(expression, [start, end])

Returns the sum of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

For example, the view below shows quarterly sales. A window sum computed within the Date partition returns the summation of sales across all quarters.



Example

WINDOW\_SUM(SUM([Profit]), FIRST()+1, 0) computes the sum of SUM(Profit) from the second row to the current row.

WINDOW\_VAR(expression, [start, end])

Returns the sample variance of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

Example

WINDOW\_VAR((SUM([Profit])), FIRST()+1, 0) computes the variance of SUM(Profit) from the second row to the current row.

WINDOW\_VARP(expression, [start, end])

Returns the biased variance of the expression within the window. The window is defined by means of offsets from the current row. Use 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.

Example

WINDOW\_VARP(SUM([Profit]), FIRST()+1, 0) computes the variance of SUM(Profit) from the second row to the current row.

Create a table calculation using the calculation editor

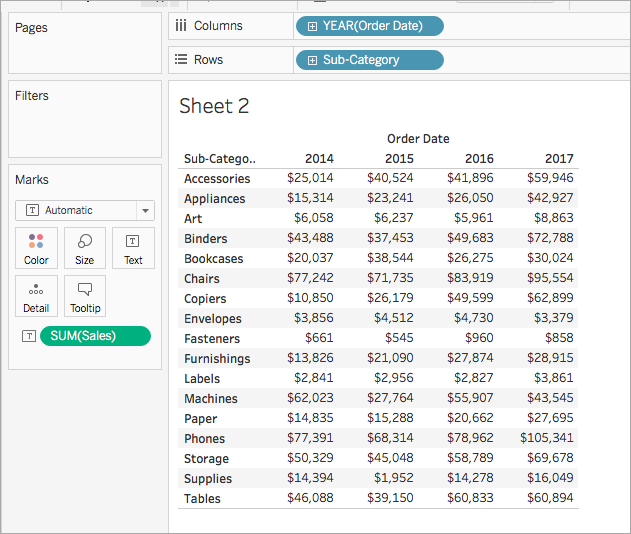
Follow along with the steps below to learn how to create a table calculation using the calculation editor.

**Note**: There are several ways to create table calculations in Tableau. This example demonstrates only one of those ways. For more information, see Transform Values with Table Calculations.

Step 1: Create the visualization

1. In Tableau Desktop, connect to the **Sample-Superstore** saved data source, which comes with Tableau.
2. Navigate to a worksheet.
3. From the **Data** pane, under Dimensions, drag **Order Date** to the **Columns**shelf.
4. From the **Data** pane, under Dimensions, drag **Sub-Category** to the **Rows**shelf.
5. From the **Data** pane, under Measures, drag **Sales**to **Text**on the Marks card.

Your visualization updates to a text table.



Step 2: Create the table calculation

1. Select **Analysis**> **Create Calculated Field**.
2. In the calculation editor that opens, do the following:
   * Name the calculated field, Running Sum of Profit.
   * Enter the following formula:

RUNNING\_SUM(SUM([Profit]))

This formula calculates the running sum of profit sales. It is computed across the entire table.

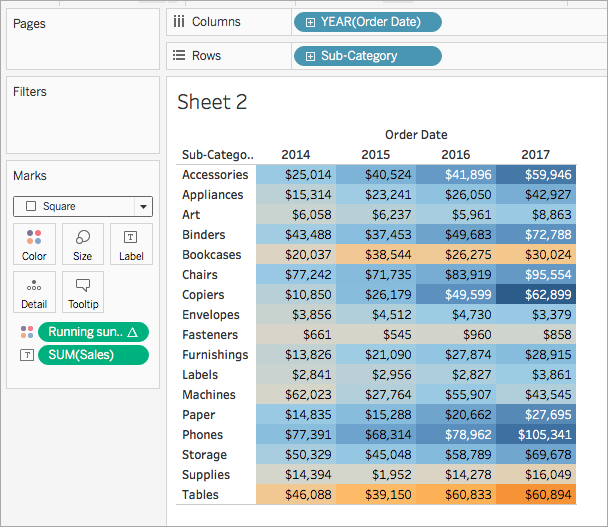
* + When finished, click **OK**.

The new table calculation field appears under Measures in the Data pane. Just like your other fields, you can use it in one or more visualizations.

Step 3: Use the table calculation in the visualization

1. From the Data pane, under Measures, drag **Running Sum of Profit** to **Color**on the Marks card.
2. On the Marks card, click the Mark Type drop-down and select **Square**.

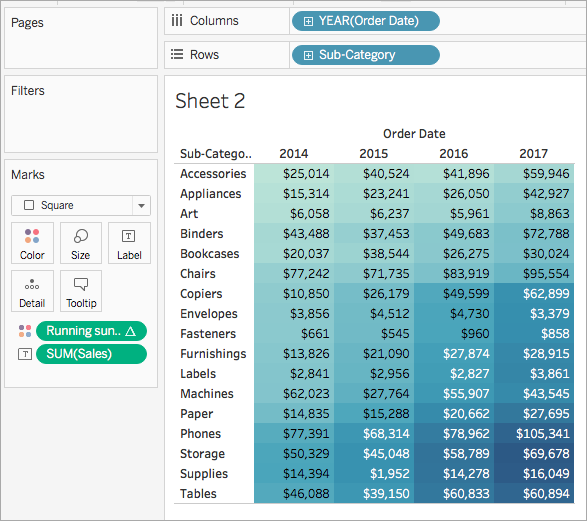
The visualization updates to a highlight table:



Step 4: Edit the table calculation

1. On the Marks card, right-click **Running Sum of Profit**and select **Edit Table Calculation**.
2. In the Table Calculation dialog box that opens, under Compute Using, select **Table (down)**.

The visualization updates to the following:



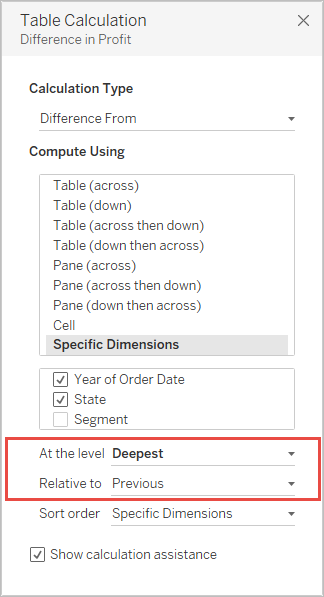
# Create a Table Calculation

You can add a table calculation to a measure in the view. To add a table calculation:

1. Click a measure in the view and choose **Add table calculation** from the field menu.

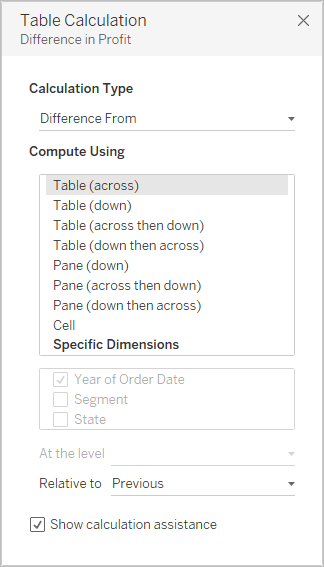
Tableau opens the Table Calculation dialog box and also adds highlighting to the view. As you work in the Table Calculation dialog box, highlighting shows the scope and direction of the calculation.

1. Choose a Calculation Type.
2. For each calculation type there is a specific set of options for you to consider. For example, for a **Difference From** calculation, there are two associated fields:

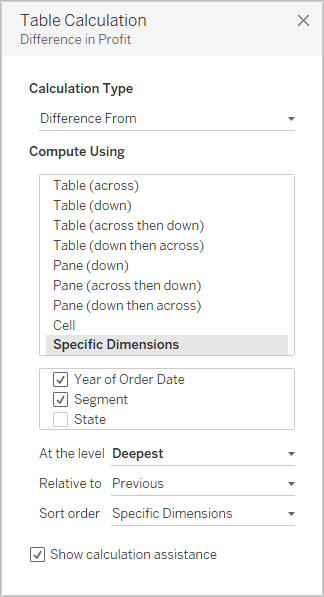


For information on these fields, go to Table Calculation Types, and then click the link for **Difference From Calculation**.

1. To define the table calculation using the visual structure of your view, choose one of the **Compute Using** options.



Alternatively, to define the table calculation by referencing specific dimensions in the view, choose **Specific Dimensions** and then select and order dimensions in the box further down in the Table Calculation dialog box.



# Customize Table Calculations

You can always customize a table calculation by editing it in the Table Calculations dialog box, but there are other, more specialized ways to customize a table calculation.

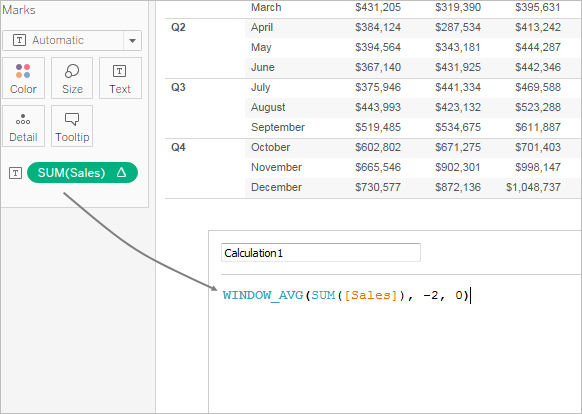
## Customizing a Table Calculation Using Its Context Menu

Click any field in the view to see a context menu listing ways to customize the field. For a field in the view that has a table calculation, you can change the **Compute Using** option—that is, the option that determines the direction and scope of the calculation relative to the visual structure of the view. Do this by clicking the field and then choosing an option from the **Compute Using** list.

For **Difference From**, **Percent Difference From**, and **Percent From** table calculations, you can also specify a different field from which the difference should be computed. Do this by clicking the field and then choosing an option from the **Relative to** list. The options are **Previous**, **Next**, **First**, and **Last**.

## Customizing a Table Calculation Using the Calculation Editor

You can customize a table calculation by dragging it into the calculation editor:



When you edit a table calculation in the calculation editor, you can click **Default Table Calculation** in the lower-right corner of the editor to open the calculation in the Table Calculation dialog box. This will allow you to create a new named calculated field that uses the same table calculation as the one you are basing the calculation on.

## Nested Table Calculations

A nested table calculation can be one of two types of calculated fields:

* A calculated field that includes more than one calculated field with a table calculation (as in the example below), or
* A calculated field that itself has a table calculation and includes at least one calculated field with a table calculation.

With nested table calculations, you can set **Compute Using** configurations for individual calculations independently.

Here is a scenario you can try, using the Sample - Superstore data source that is included with Tableau Desktop, which results in a nested table calculation.

1. Drag **Sub-Category** to Columns and **Region** to Rows.
2. Create a calculated field, *1-nest*, with the definition TOTAL(SUM([Sales])).

TOTAL is a table calculation function, so this calculated field automatically has a table calculation—when you use it in the view, the field will have the tell-tale table triangle, indicating a table calculation:

https://onlinehelp.tableau.com/current/pro/desktop/en-us/Img/table_calculations_indicator.png

For information on table calculation functions, see Table Calculation Functions.

1. Create a second calculated field, *2-nest*, with the definition TOTAL(SUM([Profit])).
2. Create a third calculated field, *3-nest*, with the definition [1-nest] + [2-nest].
3. Drag *3-nest* and drop it to the right of Sub-Category on Columns.
4. Click *3-nest* on Columns and choose **Edit Table Calculation**.

In the Table Calculations dialog box, you can now separately configure the underlying table calculations:

