**7 Ways to Create a Running Total in Power BI**

Do you need to create a running total in Power BI?

Whenever you are working with the data, calculating Running Totals is just a common task in the day-to-day life of a data analyst. Using Running Total, you can easily show and visualize the trends and patterns in your data.

If you use Power BI frequently, you might have faced issues creating Running Totals in the tool. Throughout this article, I will introduce different ways of creating it in Power BI. Using the Power Query editor, DAX Measure, and based on some conditions as well.

What is a Running Total?

The Running Total (also called Cumulative Total) is a sum of all previous and current values in a column until the last working row of that column. They are defined and viewed in the context of the total of a column containing a numeric value as soon as a new value is added to it. You also sum up all previous and current cell values to check this. This is the general concept of Running Total.

This post explores different ways of creating Running Total in Power BI.

The Example Dataset

A screenshot of a spreadsheet

Description automatically generated

I will use dummy sales data for this example, which holds the daily sales record for each product category for the year 2022. The dataset looks like the one below in Excel.

I have this data loaded inside Power BI and now want to create a Running Total for the Sales Amount column so that I can see every day how much sales are happening, plus total sales at the end of any given day starting from January 1st, 2022.

Running Total in Power Query

A screenshot of a computer

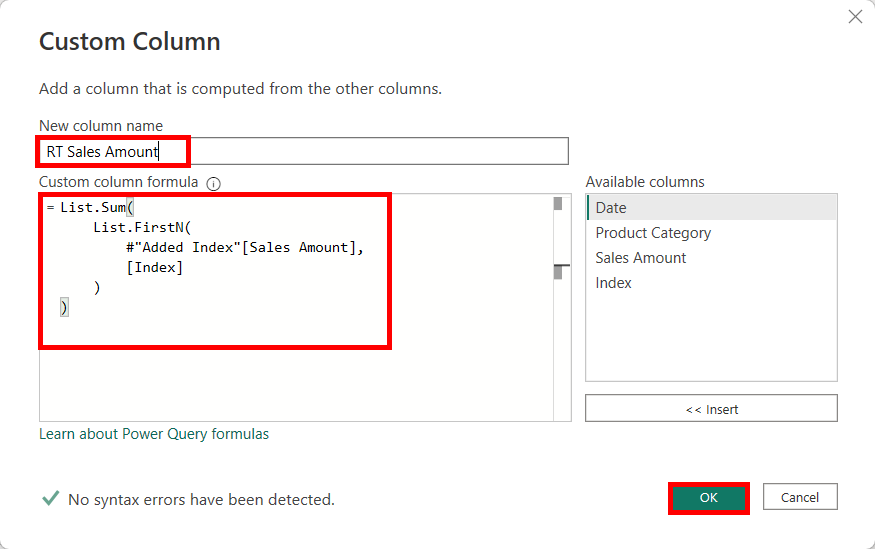
Description automatically generated

Click the **Transform Data** option to launch the **Power Query Editor** from Power BI Desktop. This option is placed under the **Queries** section inside the **Home** tab.

A screenshot of a computer

Description automatically generated

Since you are working with the Power Query, which doesn’t work on the row/cell references like Excel, you must add an index column for calculating the running total in Power Query.



Go to the **Add Column** tab > **Index Column** dropdown > select **From 1**. This will create the index column, which starts with 1 and increments by 1 unit at every iteration.

Now, you wish to add a custom column again, which can sum up the **Sales Amount** column and generate a running total.

Click on the **Add Column** > **Custom Column**. This will open up the **Custom Column** window.

Put the following formula in the **Custom column formula** section to create a column named **RT Sales Amount**. Column names should be entered in the **New column name** section.

List.Sum(

List.FirstN(

#"Added Index"[Sales Amount],

[Index]

)

)

Once done, click on the **OK** button to create a custom column for running total.

Formula Explanation

* The **List.FirstN()** function takes two arguments. The first argument is a list I provided as a Sales Amount column. Since I am making this calculation on the table, which is modified by the previous step named **“Added Index**” in Power Query, I have to use this step as a reference and then add the column name in squared brackets.
* The second parameter is the **countOrCondition**. This argument informs the formula about how many elements are then captured for the calculation. For example, if you are on row three, it says to select the first three elements and provide them as a list. In this case, a column named Index holds the count of elements this function should consider at each iteration. Hence it is provided as input.
* This entire formula is then wrapped under **List.Sum()** function generates the total for the selected column in the first parameter based on the number of elements mentioned in the second parameter.
* A final word of warning here. The function/formula you use makes things slow at the execution level. The entire formula checks the index value and list value during every iteration. It doesn’t hold the list values in buffer memory and forgets it after the evaluation of the formula after the evaluation and execution of each iteration. To tackle this, use of the **table.Buffer()** formula is recommended when working with data that holds millions or trillions of rows for better and faster performance.

A screenshot of a computer

Description automatically generated

If you check now an additional column in your table sums up the running total for the Sales Amount for each row.

Categorywise Running Total in Power Query

The running total captured in the previous section is standard stuff. But things get quirky when you have the category-wise running totals to be prepared. For example, you have four different product categories Bicycles, Clothes, Toffies, and Toys. What if you wanted a running total based on these four categories? Well, it is also possible, and I will show you how through this section.

Load the data to Power BI and access the Power Query editor. You first need to sort the data based on the product categories so that you can add an index to the sorted data and then generate the running total in the Power Query.

A screenshot of a computer

Description automatically generated

Click the OK button on the **dropdown arrow** from the **Product Category** column > **Sort Ascending** >. This will sort the data based on Product Category in ascending order. All Bicycles will appear first, Clothes will appear after the bicycles, and so on.

Then add an index column so that you can add the running total.

In this step, I am directly going to write an M query rather than creating a custom column.

A white screen with yellow border

Description automatically generated

To write a new query, click the **Add Step** option (**fx** symbol) and copy-paste the following query into the formula bar.

= Table.AddColumn(

#"Added Index",

"Product Wise Running Total",

(OutTable) =>

List.Sum(

Table.SelectRows(

#"Added Index",

(InTable) => InTable[Index] <= OutTable[Index]

and

InTable[Product Category] = OutTable[Product Category])[Sales Amount]

)

)

A screenshot of a spreadsheet

Description automatically generated

This will create a custom **“Product Wise Running Total”** column and generate a cumulative sum for the Sales Amount data based on the product categories.

Formula Explanation

* The **Table.SelectRows()** function generates a tabular layout for each row of the original table, which is created in the previous step named **“Added Index.”** the **InTable** and **OutTable** are temporarily designed tables. The **InTable** works as an input table, and **OutTable** works as an output table.
* Since both tables are basically single-liners, you are compiling a collection that will have an index value higher than the previous index value of the input table and less than the index value of the output table. The result is stored in a list, obviously.
* The other condition based on which the data is being filtered is product category of both the dummy tables should match.
* To sum up the values for the selected rows, **List.Sum()** function is used. It basically sums up all the values where the current index value is greater than the previous index value and the product category is the same. The output result is again a list of tables (remember, each row is considered a table here).
* But since I wanted a column rather than a list, I will use the **table.AddColumn()** formula and convert this list of tables into a column that provides the running total based on the product category values.

**Note:** The **InTable** and **OutTable** are temporarily designed tables for this case, and their presence will never be found anywhere outside of this formula. They are in buffer memory and are not being created physically.

Running Total Using a DAX Measure

Now, enough of Power Query stuff! Now I will teach you about creating the running total using a DAX Measure.

A screenshot of a computer

Description automatically generated

Suppose you have sales data as shown above with three columns, namely Date, Product Category, and Sales Amount in Power BI.

Create a Matrix visual based on this data in Report View, which consists of two columns Date and Sales Amount.

A screenshot of a computer

Description automatically generated

In the **Visualization** pane, select **Matrix** > drag the **Date column** from Sales\_Table through the Data pane and drop in the **Rows** > drag **Sales Amount** and drop in **Values**. The visual should look like the one above.

We want a DAX formula that can return the running total of Sales Amount in a way that for each day, the values sum up, and finally, at the end of the year, it would be the total sales starting from January 1st Until December 31st. You can quickly get this done by creating a measure. Let’s see how.

A screenshot of a computer

Description automatically generated

Use the mouse to right-click on the **Sales\_Table** from the **Data** section and choose the first option **New Measure**. This will allow me to create a newly formulated measure in Power BI.

A screenshot of a computer code

Description automatically generated

Write down the following formula in the **formula bar** at the top of the canvas. Feel free to give an appropriate name to the measure so that it then be identified with that name inside the table.

Sales Running Total =

VAR maxdate = MAX(Sales\_Table[Date])

RETURN

CALCULATE(

SUM(Sales\_Table[Sales Amount]),

Sales\_Table[Date] <= maxdate,

ALL(Sales\_Table[Date])

)

Once you are done writing (or copying) the formula in the formula bar, click on the Commit button that looks like a right-tick at the left. This will evaluate this formula and potentially execute it if there are no errors. If there are any errors, it will throw warnings in yellow colored warning boxes at the end of the formula bar.

Formula Explanation

* The **VAR** keyword allows you to create a variable with the name maxdate. It captures the maximum date available inside the date column (in this case, December 31st, 2022) using the **MAX** function. Every **VAR** keyword comes with a **RETURN** keyword which consumes the variable defined under **VAR** in subsequent formula expressions.
* The **SUM** function takes the total of the Sales Amount.
* The **CALCULATE** function then applies a date filter to all rows until the rows where each date is before or equal to the value stored in the maxdate variable (December 31st, 2023).
* Finally, the **ALL** function returns all the values from this table, ignoring other filters, if any, apart from the one applied in the **CALCULATE** function.
* This execution happens row by row until the last date of the table. Returning the running total of the Sales Amount.

A screenshot of a computer

Description automatically generated

Drag the newly created measure **Sales Running Total** inside the **Values** section (below **Sum of Sales Amount**) of the visual and see the day-wise running total as above.

Categorywise Running Total using a DAX Measure

Once you know how to work with DAX, it is pretty simple to extend the logic here. Suppose you wish the running total category-wise in Power BI. You can extend the very same logic described above. The only thing that will change is you will be working with the **Product Category** column instead of the **Date** column in this scenario.

Use the following formula to create a category-wise running total in Power BI.

Sales Running Total Categorywise =

VAR prodcat = SELECTEDVALUE(Sales\_Table[Product Category])

RETURN

CALCULATE(

SUM(Sales\_Table[Sales Amount]),

Sales\_Table[Product Category] <= prodcat,

ALL(Sales\_Table[Product Category])

)

The formula above works precisely the same apart from the **SELECTEDVALUE** function. The function is introduced here to return the first value from a column reference (Sales\_Table[Product Category]) if it is the only value returned in the CALCULATE function’s filter context. Otherwise, it returns the blank value.

Therefore, if you see, when a variable named prodcat is used as a filtering criterion inside the **CALCULATE** function, it runs the **SUM** for each **Product Category** because that is what the **SELECTEDVALUE** is choosing for each iteration.

In Short, this function creates four unique lists of distinct product categories at each iteration, which are then used as a filtering criterion.

A screenshot of a computer

Description automatically generated

Then, please create a new Matrix/Table visual and drag the **Product Category**, **Sales Amount**, and **Sales RT Categorywise** in it to see how product wise running total is created.

Running Total Using Quick Measure (No Need to Write DAX Formula)

Now, a few people might be lazy like me (High Five if you are in the same league! 😉) and might be bamboozled because they might have to write formulas created above. Power BI takes care of such lazy people and allows you to create a Quick Measure where you drop the columns you want the running total for and then the field based on which you want the running total. This method creates a DAX Measure without writing a DAX Code.

That might sound weird, but it is true. Let’s check it out.

A screenshot of a computer

Description automatically generated

Right Click on the **Sales\_Table** to see the options and click on the **New Quick Measure** option (third from top) to create a quick measure without writing a DAX formula.

A screenshot of a calculator

Description automatically generated

This will open up the **Quick Measure** pane, as shown in the screenshot above. There are two tabs, **Calculations** and **Suggestions**.

In the first tab, you can select your own calculation from the dropdown list and provide the required fields. The Power BI will create a DAX Formula for your measure based on the values and fields provided by you in the backend.

As the name reflects, the **Suggestions** tab will suggest what measures you can create based on the value and fields from your table. However, you need to have a work account to sign in to Power BI to use this feature quickly.

For this demo, I will stick to the Calculations tab.

A screenshot of a calculator

Description automatically generated

Select the **Running Total** through the **Totals** group from the **Select a calculation** dropdown under the **Calculations** tab.

There will be three sections.

* **Base value:** represents a value whose running total or cumulative total you want. In this example column, Sales Amount.
* **Field:** is a column based on which the running total will be calculated. In this example, it will either be the Date or Product Category column.
* **Direction:** Sorting direction, either ascending or descending, when you calculate the running total.

A screenshot of a computer

Description automatically generated

Drag and drop the **Sales Amount** column in the **Base Value** section. The **Date** should be in the **Field** section. Click on the **Add** button to create a quick measure out of the selection. That’s it!

A computer screen shot of a computer code

Description automatically generated

Power BI creates a measure with a suitable name based on the data provided. In this case, the column name is Sales Amount running total in Date, and I will rename it to **Sales RT Quick Measure**.

The exciting thing is Power BI automatically writes a DAX formula that calculates the datewise running total for Sales Amount. Check it out in the screenshot above.

A screenshot of a computer

Description automatically generated

Now you can check this measure across the date values in the Matrix visual, and it works the same way the Sales Running Total measure works that I created in a couple of sections above.

Creating MTD, QTD, and YTD Running Totals in Power Query

In the past, I have shared various methods of creating daily running totals and category-based running totals in Power BI. These techniques involve using Power Query and DAX measures. Nonetheless, when dealing with running totals, generating month-to-date, quarter-to-date, and year-to-date values is crucial.

In this section, I will demonstrate how to create date-based running totals using Power Query.

A screenshot of a computer

Description automatically generated

To access the **Power Query Editor**, simply go to the **Home** tab and click on the **Transform data** option located under the **Queries** section.

A screenshot of a computer

Description automatically generated

To create an index column starting with index value 1, go to the **Power Query Editor** and navigate to the **Add Column** section. Then, click on the **Index Column** dropdown and select **“From 1.”**

After successfully setting up the index column, click on the Add step (fx) button in the Formula Bar. Then, simply copy and paste the formula below to generate a new column that accurately captures the month-to-date running total.

= Table.AddColumn(

#"Added Index",

"Monthly Running Total",

(OutTable) =>

List.Sum(

Table.SelectRows(

#"Added Index",

(InTable) => InTable[Index] <= OutTable[Index]

and

Date.EndOfMonth(InTable[Date]) = Date.EndOfMonth(OutTable[Date]))[Sales Amount]

)

)

The explanation for this formula can be found in the “Category Wise Running Total in Power Query” section. The only difference is that you have included an additional Power Query function called **“Date.EndOfMonth()”** and replaced the **Product Category** column with the **Date** column in both the Input and Output tables.

A screenshot of a spreadsheet

Description automatically generated

The formula generates a running total for each month, as displayed in the screenshot. The running total stops at the end of January 2022 and starts over with the Sales Amount for February. This process repeats for each month until the last date value is reached.

Once you know the recipe, extending this logic and creating the quarterly and yearly running total is pretty straightforward. Use the following formulas for them respectively in Power Query.

//Formula for Quarterly Running Total

= Table.AddColumn(

#"Monthly Running Total",

"Quarterly Running Total",

(OutTable) =>

List.Sum(

Table.SelectRows(

#"Monthly Running Total",

(InTable) => InTable[Index] <= OutTable[Index]

and

Date.EndOfQuarter(InTable[Date]) = Date.EndOfQuarter(OutTable[Date]))[Sales Amount]

)

)

//Formula for Yearly Running Total

= Table.AddColumn(

#"Quarterly Running Total",

"Yearly Running Total",

(OutTable) =>

List.Sum(

Table.SelectRows(

#"Quarterly Running Total",

(InTable) => InTable[Index] <= OutTable[Index]

and

Date.EndOfYear(InTable[Date]) = Date.EndOfYear(OutTable[Date]))[Sales Amount]

)

)

The second formula (yearly running total) will produce more beneficial outcomes when multi-year data is present in your source table.

Creating MTD, QTD, and YTD Running Totals Using DAX Measure

It’s pretty simple to apply the same concept of calculating running totals monthly, quarterly, or yearly in Power BI by utilizing DAX Measures.

A screenshot of a computer

Description automatically generated

To create a DAX measure from scratch, simply right-click on the **Sales\_Table** and select the **New Measure** option.

A screenshot of a computer program

Description automatically generated

In the formula bar, please type the following formula and click on the Commit button to execute it.

Sales RT MTD =

CALCULATE(

SUM(Sales\_Table[Sales Amount]),

DATESMTD(Sales\_Table[Date])

)

* The **CALCULATE** function here takes the first argument as the **SUM** of the **Sales Amount** column.
* This total of the **Sales Amount** is then filtered based on the **DATESMTD** function, which returns a set of dates until the last date of a particular month visible in filtering criteria.

A screenshot of a computer

Description automatically generated

Then drag this newly created measure into the **Matrix** visual alongside the **Date** and **Sales Amount** columns and see the difference. At the start of the new month, the cumulative sum starts from scratch, which makes it a monthly running total.

Using the same logic, you can create the quarterly and yearly running totals for the Sales Amount column. The formula for both is as shown below.

Sales RT QTD =

CALCULATE(

SUM(Sales\_Table[Sales Amount]),

DATESQTD(Sales\_Table[Date])

)

Sales RT YTD =

CALCULATE(

SUM(Sales\_Table[Sales Amount]),

DATESYTD(Sales\_Table[Date])

)

Conclusions

* There are various ways of creating a running total in Power BI using the Power Query formulas (that can be tricky sometimes) or using the DAX Measures.
* If you are creating a running total in Power Query, it will be a part of your data model, and you must add the Index column so that the formulas understand that they need to work with it to generate the running total.
* In DAX Measures, you don’t need to create any index as it can run row by row without having the index.
* You can use the Quick Measure option and save yourself some time that it takes to write the DAX formula to create running totals. However, it would be best if you were 100% sure about the structure in your head so that you can drag the right columns in the right Value and Field sections. Otherwise, it would be a mess.
* You can create the running totals based on categorical variables such as product Category and Numerical variables such as Dates.
* When you are working with dates, you can create monthly, quarterly, and yearly running totals as well using both Power Query and DAX Measures.

Phew! These were a few ways of creating Running Total in Power BI using either DAX Measures or the Power Query formulas. If you loved those, please give me a heads up using the comment section below, and in case you need any other method to create the same in Power BI, I am all ears up! Until the next time, ciao! Keep following our blog post for such awesome Power BI tips and tricks in the future.