

How to Calculate Cumulative Sum by Category in Power BI: A Step-by-Step Guide

Authored by
stats writer

January 28, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Calculate Cumulative Sum by Category in Power BI: A Step-by-Step Guide*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128332>

Power BI is widely recognized as a **powerful data analysis tool** that empowers users to transform raw data into actionable business intelligence. A frequently required and highly valuable calculation in complex data modeling is the ability to generate a **cumulative sum by category**. This calculation is vital because it allows analysts to easily track the running total of a metric (like sales, points, or costs) within the context of a specific grouping (like team, region, or product line) over a defined sequence, typically based on time or an index. Mastering this technique provides a profound view of progression and momentum, helping organizations **monitor trends** and identify inflection points within various segments of their operations. By utilizing specific functions within the DAX language, Power BI allows for the efficient and accurate generation of these cumulative metrics, delivering essential insights that go beyond simple static aggregates.

The core challenge in calculating cumulative totals lies in managing the **evaluation context**. Unlike simple summation, a cumulative total requires the formula to look back at all prior rows within the current category group and sum their values, all while maintaining the integrity of the grouping structure. This rigorous demand necessitates the use of advanced functions within **DAX** (Data Analysis Expressions), the formula language of Power BI. This article will provide a detailed, step-by-step guide on how to architect and implement this precise calculation, ensuring your data models yield accurate and insightful running totals segmented by any desired category.

Understanding the flow of data accumulation is critical for business forecasting and historical analysis. For example, a retail business tracking daily sales categorized by store location might use a cumulative sum to see which location contributed most to the overall running revenue through the quarter. Similarly, in project management, tracking accumulated budgeted costs by project phase provides immediate oversight into potential overruns. The calculation we demonstrate below, utilizing the `CALCULATE` function combined with context modification tools, offers a robust and dynamic solution for generating these essential time-series metrics.

Power BI: Calculate Cumulative Sum by Category

Understanding the Core DAX Formula

To achieve a grouped running total in **DAX**, we must use a formula that iterates through rows, sums the metric, and, crucially, resets the calculation when the grouping category changes. This is accomplished by combining three powerful DAX functions: **CALCULATE**, which modifies the filter context; **ALLEXCEPT**, which manages the grouping context; and **EARLIER**, which manages the row context for sorting and comparison.

The structure below provides the foundational syntax required. This formula is designed to create a new column containing the running total of a measure, constrained by a specific category column. The use of **ALLEXCEPT** is key, as it removes all filters from the table except those applied to the

specified category column (in this case, 'my_data'), ensuring the calculation is performed only within that category's boundaries.

You can use the following syntax in DAX to calculate the cumulative sum of values by category in **Power BI**:

```
Cumulative Sum =  
CALCULATE (  
SUM ( 'my_data' ),  
ALLEXCEPT ( 'my_data', 'my_data' ),  
'my_data' <= EARLIER ( 'my_data' )  
)
```

This particular formula creates a new column named **Cumulative Sum** that contains the running total of values found in the **Points** column. The calculation is explicitly grouped and reset based on the values in the **Team** column. The logic of `'my_data' <= EARLIER ('my_data')` is what enforces the cumulative behavior, comparing the current row's index to all previous indexes within the defined context.

The Critical Role of the Index Column

A crucial prerequisite for the DAX formula above to function correctly is the existence of a sequential **Index** column. The formula relies on the **EARLIER** function to compare the current row's index value against all other index values in the table context. If the data is not inherently sorted by a timestamp or a reliable sequence, an explicit numerical index must be created.

The index column provides the necessary row context for the **CALCULATE** function to iterate and apply the cumulative filter. Without this sequential marker (ranging from 1 to N for the entire table), DAX cannot reliably determine which rows precede the current row, making an accurate running total impossible, especially when combined with the **ALLEXCEPT** function used for category grouping.

This formula assumes that you have created an **Index** column that ranges from 1 to N, applied across the entire data table. If your source data already contains a unique, sequential identifier (like a chronological transaction ID), you may use that column instead, provided it ensures the correct sorting order for the cumulative calculation.

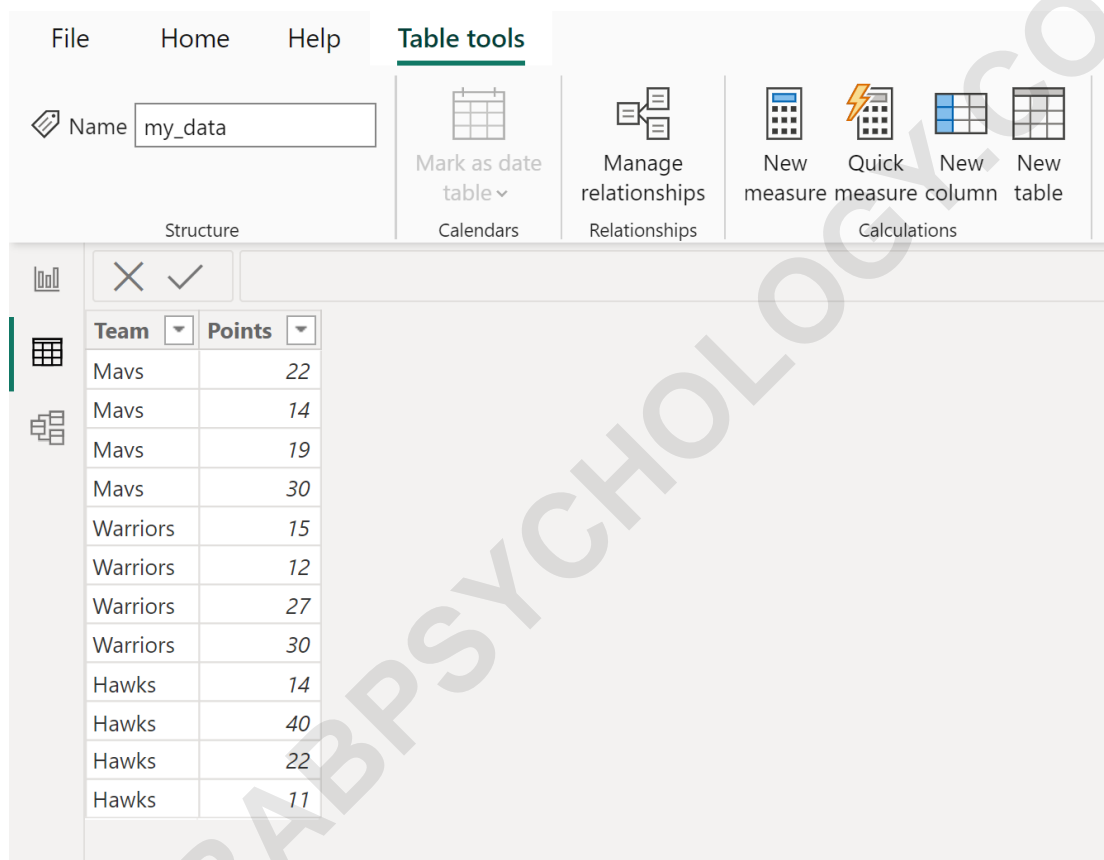
Example Setup: Preparing the Data Table

To illustrate the application of this method, consider a practical scenario involving sports data. Suppose we have a table in **Power BI** named **my_data** that captures points scored by individual

basketball players, identified by their respective teams. Our ultimate objective is to calculate the cumulative points scored for each team as we move down the list of recorded player contributions.

This initial setup is vital for defining the problem and visualizing the goal. The original table contains two key columns: **Team** (the grouping category) and **Points** (the value to be accumulated).

Suppose we have the following table in Power BI named **my_data** that contains information about points scored by basketball players on various teams:



The screenshot shows the Power BI interface with the 'Table tools' ribbon active. The table name is 'my_data'. The ribbon includes options for 'Mark as date table', 'Manage relationships', and 'Calculations' (New measure, Quick measure, New column, New table). Below the ribbon, a table is displayed with the following data:

Team	Points
Mavs	22
Mavs	14
Mavs	19
Mavs	30
Warriors	15
Warriors	12
Warriors	27
Warriors	30
Hawks	14
Hawks	40
Hawks	22
Hawks	11

The requirement is to create a new column that shows the cumulative points scored by players, ensuring that the running total resets every time the **Team** category changes. Before we can apply the complex DAX logic, however, we must ensure the data structure includes the previously mentioned Index column to facilitate the row-by-row comparison necessary for the cumulative function.

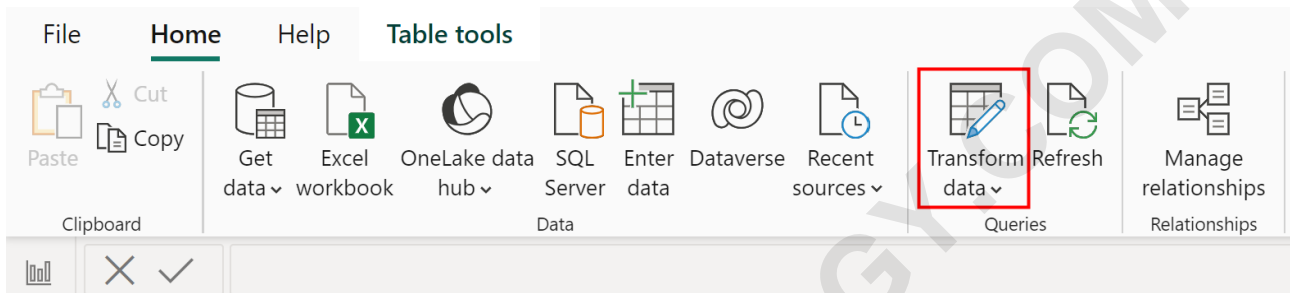
Step-by-Step: Accessing the Power Query Editor

The creation of the required **Index** column is not performed in the DAX environment (Data View) but within the modeling environment, specifically the **Power Query Editor**. Power Query, often

referred to as 'Get & Transform Data', is where data shaping, cleaning, and preliminary transformations are executed before the data is loaded into the data model.

To access this environment, navigate to the main ribbon in **Power BI Desktop**. You will typically find the necessary command on the **Home** tab.

Before we proceed with the DAX calculation, we need to first add an index column to the table. To do so, click the **Home** tab along the top ribbon, then click the **Transform data** icon:

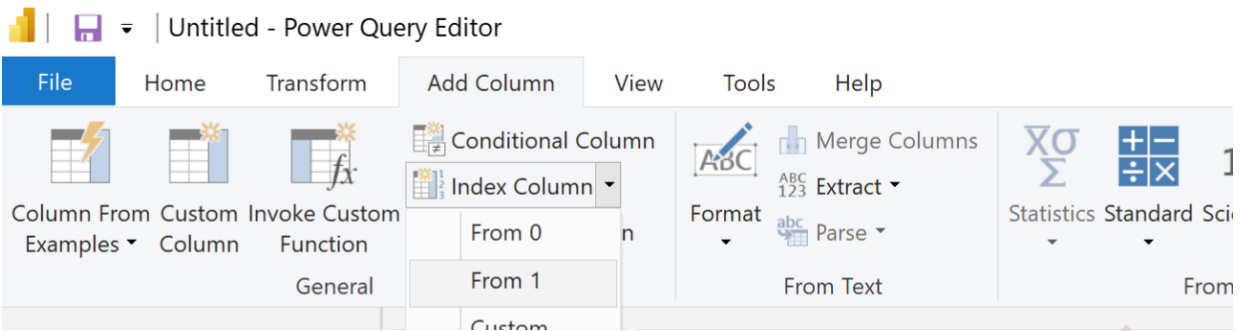


Clicking this icon launches the **Power Query Editor**, opening a new window where all data source transformations are managed. It is within this editor that we can easily add sequential columns without writing complex M language code. This step is non-negotiable for the successful execution of our categorized cumulative sum formula in DAX.

Implementing the Sequential Index Column

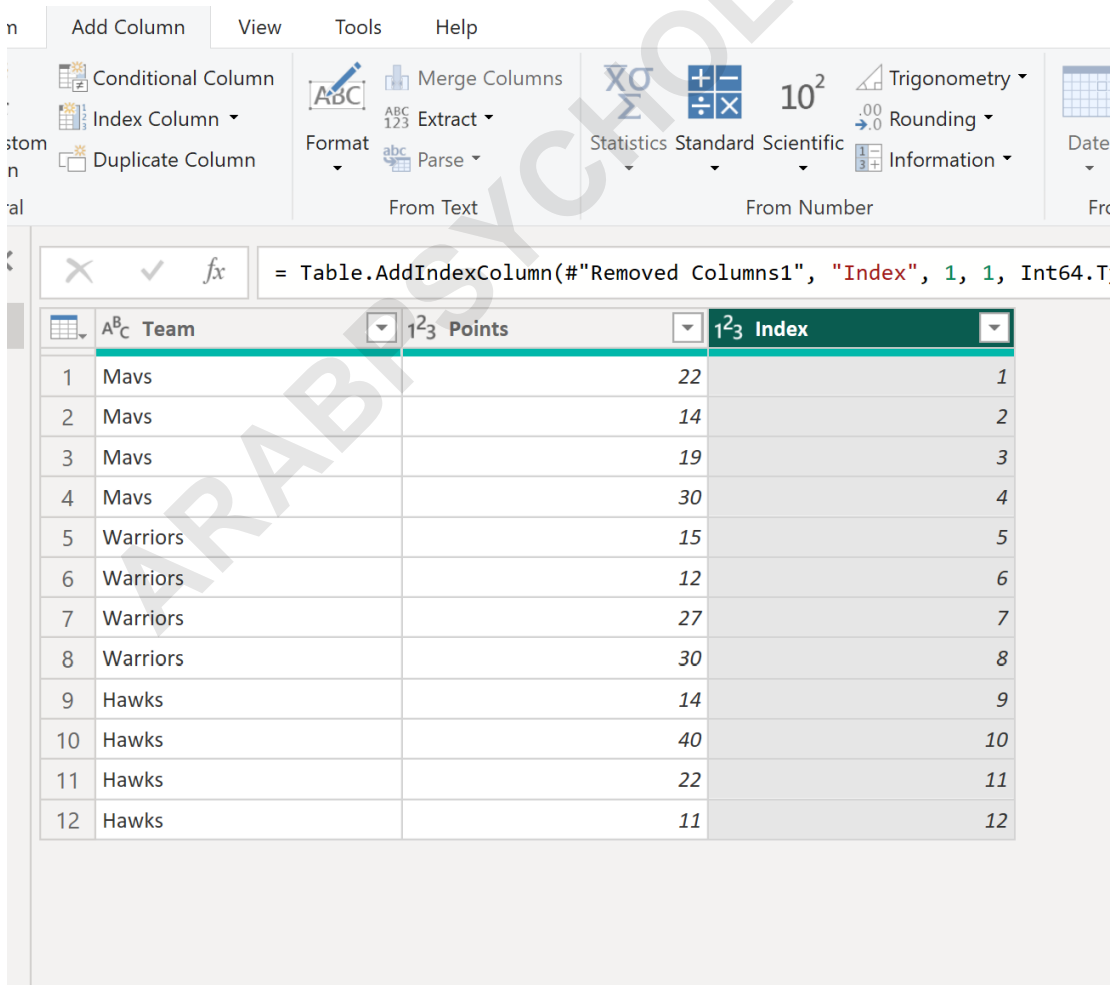
Once inside the **Power Query Editor**, adding the Index column is a straightforward process. The editor provides pre-built tools for common transformation tasks, including the generation of sequential numbering. This column must be added before we exit Power Query and return to the main Power BI Data View.

This will bring up the **Power Query Editor**. Next, click the **Add Column** tab, then click the dropdown arrow next to **Index Column** and choose whether or not you'd like the index values to start at 0 or 1:



It is generally recommended to start the index at 1, especially when dealing with row-based comparisons, as it avoids potential confusion with zero-based indexing in formula writing. However, the choice (starting from 0 or 1) is largely aesthetic as long as the index is consistent and sequential.

We'll click **From 1** so that the following index column is successfully generated and appended to the data table:



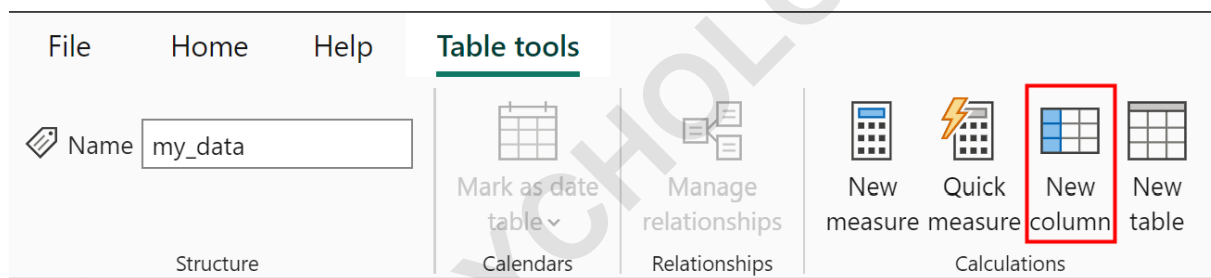
After confirming the creation of the index, you must load the transformed data back into the Power BI Data Model. This is achieved by navigating to the **Home** tab within the Power Query Editor and selecting **Close & Apply**. Once you click **Close & Apply**, the new index column will be added to the table in your Power BI Data View, making it available for DAX calculations.

Applying the Cumulative DAX Calculation

With the necessary Index column now part of the data model, we can proceed to implement the DAX formula that executes the grouped running total. This calculation must be created as a **New Column** (a calculated column) within the Data View or Report View of Power BI Desktop.

To start, navigate to the **Table tools** tab in the Power BI ribbon. This section provides tools specific to modifying table structure and adding calculated measures or columns.

Lastly, to add a new column that shows the cumulative points by team, click the **Table tools** tab, then click the **New column** icon:



Once the formula bar appears, it is time to input the previously defined **DAX** calculation. Remember, this formula uses **EARLIER** to establish the row context and **ALLEXCEPT** to ensure that the calculation only accumulates data within the same team.

Then type the following formula into the formula bar:

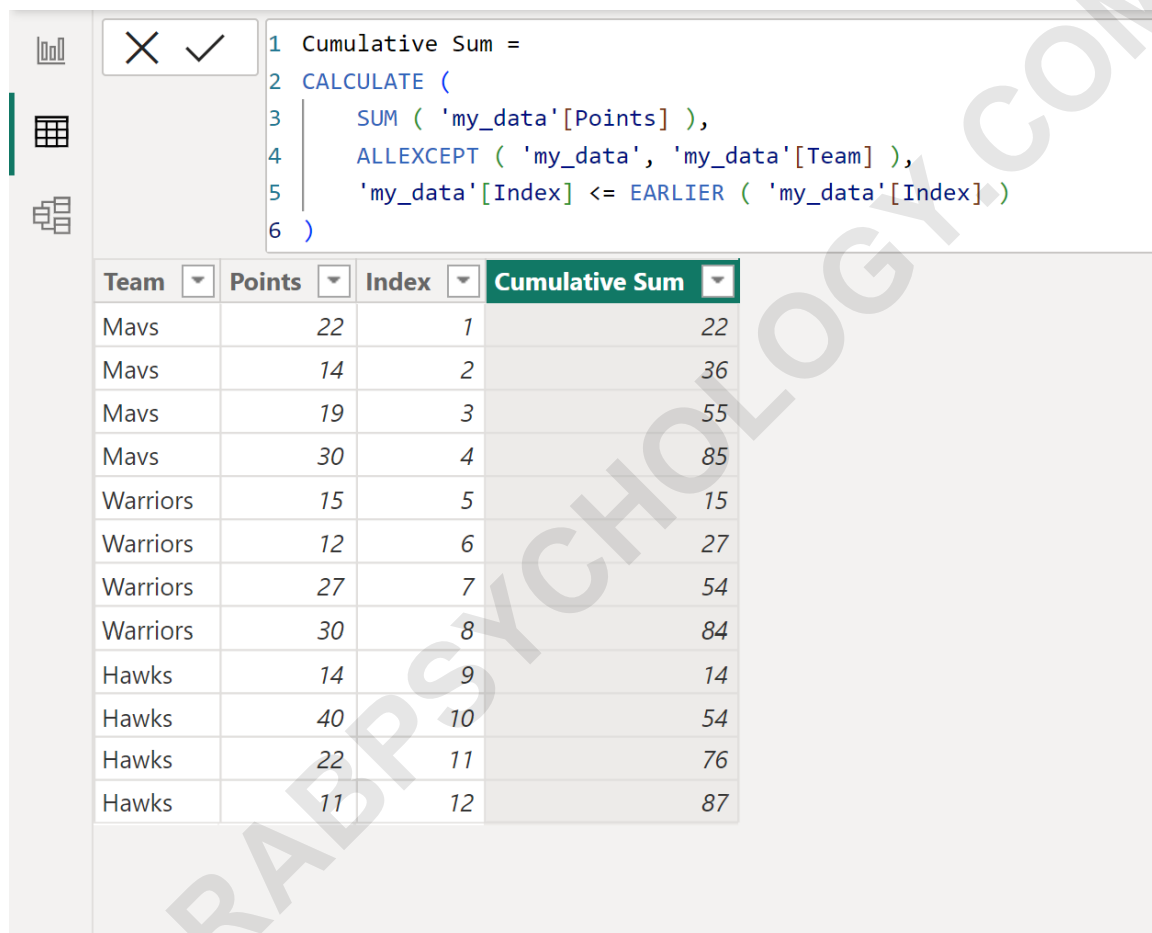
```
Cumulative Sum =  
CALCULATE (  
SUM ( 'my_data' ),  
ALLEXCEPT ( 'my_data', 'my_data' ),  
'my_data' <= EARLIER ( 'my_data' )  
)
```

The calculated column immediately populates across all rows of the table. The result is a highly functional and accurate cumulative sum that respects the team categories defined in your data model, providing immediate visual feedback on the running totals.

Analyzing the Final Cumulative Results

Upon successful execution of the DAX formula, a new column named **Cumulative Sum** is added to the `my_data` table. This column dynamically calculates the accumulated points based on the index order, resetting the count whenever the **Team** value changes.

A new column named **Cumulative Sum** will be created that shows the cumulative sum of values in the **Points** column for each team:



The screenshot displays the DAX formula editor and a data table. The formula calculates the cumulative sum of points for each team, resetting the sum when the team changes. The table below shows the results of this calculation.

```
1 Cumulative Sum =
2 CALCULATE (
3     SUM ( 'my_data'[Points] ),
4     ALLEXCEPT ( 'my_data', 'my_data'[Team] ),
5     'my_data'[Index] <= EARLIER ( 'my_data'[Index] )
6 )
```

Team	Points	Index	Cumulative Sum
Mavs	22	1	22
Mavs	14	2	36
Mavs	19	3	55
Mavs	30	4	85
Warriors	15	5	15
Warriors	12	6	27
Warriors	27	7	54
Warriors	30	8	84
Hawks	14	9	14
Hawks	40	10	54
Hawks	22	11	76
Hawks	11	12	87

Observing the results, notice how the cumulative total increases sequentially for the 'A' Team (10, 25, 40). When the index shifts to the 'B' Team, the cumulative count correctly resets to the first recorded score for that team (8), and then begins accumulating from there (8, 20, 31). This confirms that the combination of **CALCULATE**, **ALLEXCEPT**, and the Index comparison driven by **EARLIER** successfully partitions the running total calculation by the specified category.

The new **Cumulative Sum** column shows the cumulative sum of values in the **Points** column for each team, providing analysts with a powerful tool for sequential performance tracking and comparative analysis within different segments of the data. This metric is invaluable for generating

time-based charts and trend visualizations in your reports.

Conclusion and Further Learning

Calculating a categorized **cumulative sum** is an essential skill in advanced Power BI data modeling. While the DAX formula may initially appear complex due to the interplay of context transition functions, its systematic implementation ensures accurate and granular tracking of running totals across distinct categories. The critical takeaway is the necessity of creating a reliable, sequential index column as the foundation for the row-context comparison.

Mastery of context modification techniques, such as those demonstrated using CALCULATE and ALLEXCEPT, opens the door to solving many complex analytical challenges in Power BI. By applying these concepts, users can move beyond simple aggregations and leverage the full descriptive power of their data.

The following tutorials explain how to perform other common tasks in **Power BI**: