How to Calculate Cumulative Sum by Category in Power BI

Authored by

mohammed looti

January 12, 2026

Mastering Cumulative Calculations in Power BI

The ability to calculate a cumulative sum (or running total) is fundamental for advanced data analysis. When working with analytical tools like Power BI, understanding how to generate these totals specifically by category is essential for monitoring performance trends, inventory movement, or financial accumulation over time. This technique moves beyond simple aggregations, providing deep insight into patterns and growth rates within specific subsets of your data.

In Power BI, achieving this categorized running total requires leveraging the power of DAX (Data Analysis Expressions). Specifically, we combine the robust context modification capabilities of the CALCULATE function with filtering logic that ensures the calculation only considers rows up to the current point within a defined category group. This approach guarantees an accurate, dynamic running total that respects the hierarchical structure of your dataset.

While a simple running total can be achieved, grouping it by a specific category (like 'Team' or 'Product Line') introduces complexity because the calculation must reset every time the category changes. The method outlined below uses a combination of CALCULATE, sorting references (often via an Index column), and context preservation tools like ALLEXCEPT or ALL/FILTER to achieve the desired result--a true running total segmented by category.

Understanding the Core DAX Pattern for Running Totals

To construct a categorized cumulative sum, we need a specific pattern in DAX that manages two critical aspects: first, iterating through the rows in the correct order, and second, ensuring the row context is limited only to the current category. The standard running total formula requires an ordering column, such as a date or, in the absence of a date, a numerical index.

The backbone of this calculation is the CALCULATE function. This function modifies the filter context of an expression. We instruct it to calculate the overall sum of the value column (e.g., 'Points'), but then we layer several filters on top of the default filter context. These filters define which rows are included in the aggregation for the current row.

The two most crucial filtering components are the preservation of the category context and the comparison against the index. We must ensure that the calculation ignores all filters **except** the category column (to calculate the total within that category) and then only includes preceding rows based on the index. The use of functions like ALLEXCEPT or a combination of ALL and FILTER is necessary to achieve this precise context modification, making the calculation both row-specific and category-specific.

The Essential DAX Formula Syntax

The following standard syntax in DAX is used to calculate the cumulative sum of values, grouped by a specific categorical field. Note that this approach requires a unique index column for accurate ordering within the calculation context:

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

This specific formula defines a calculated column named **Cumulative Sum**. It instructs CALCULATE to sum the values in the column. The key to the cumulative calculation lies in the filter arguments:

ALLEXCEPT ('my_data', 'my_data'): This filter removes all existing filters from the table `my_data`, except for any filters applied to the column. This is what preserves the category context, ensuring the running total resets for each new team.

'my_data' <= EARLIER ('my_data'): This crucial filter iterates through the table. For the current row being evaluated, it ensures that only rows where the index is less than or equal to the index of the current row are included in the sum. This creates the running total effect.

It is essential to understand that this formula relies entirely on the existence and proper ranging of the **Index** column, which typically ranges sequentially from 1 to N across your data rows.

Prerequisite: Creating the Sequential Index Column

Before implementing the DAX cumulative sum formula, a necessary structural element must be added to your data model: a sequential index column. This column provides the numerical ordering required by the EARLIER function to accurately determine which rows precede the current row during the calculation of the running total. Without this numerical index, the comparison logic (less than or equal to) within the filter context cannot function correctly.

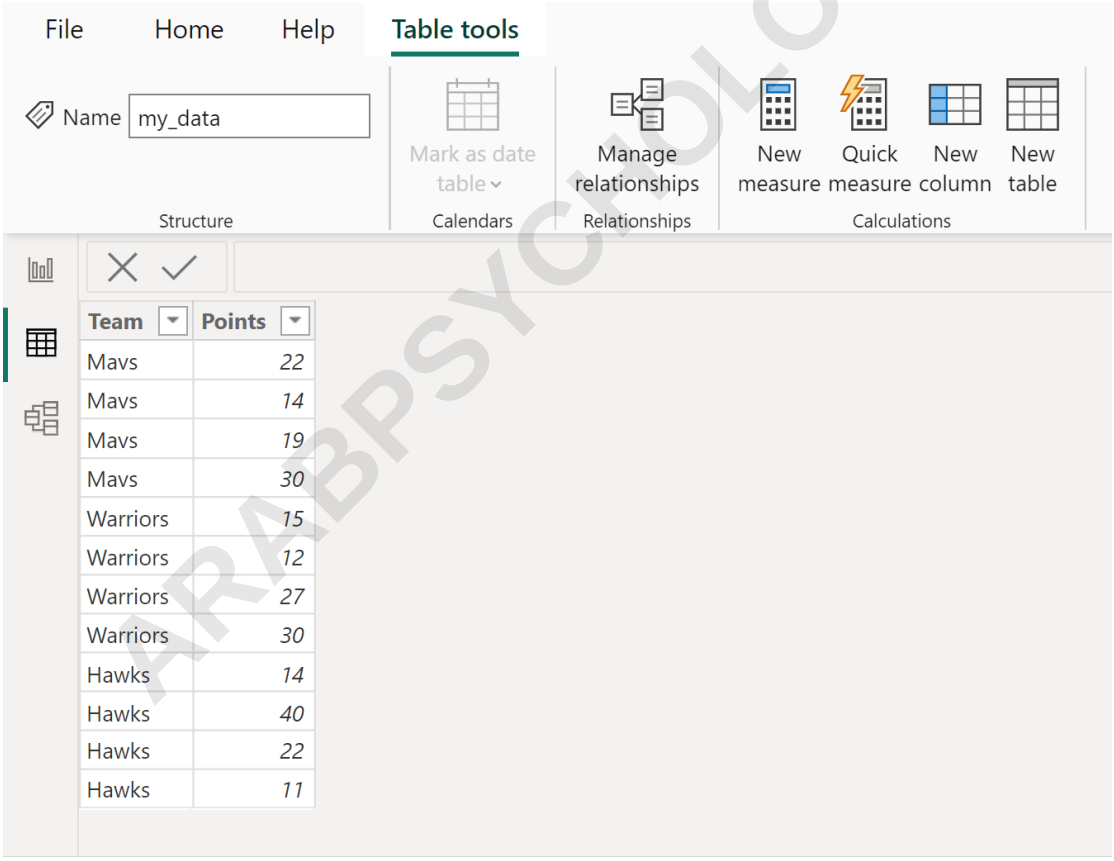
While it might seem simple, the index column is the backbone of time-insensitive running totals in Power BI. It must be created and managed within the data transformation phase, ideally using the Power Query Editor, to ensure it is robust and stable before the DAX engine processes it. Creating the index in Power Query ensures that the values are fixed during data loading, preventing unexpected calculation behavior later on.

In practical terms, the **Index** column serves as a substitute for a naturally occurring chronological field when the requirement is simply to track accumulation based on the physical order of rows within a system or report. For category-based calculations, it is paramount that this index is unique across all rows, facilitating the precise row-by-row aggregation needed for the running total.

Example Scenario: Calculating Team Points

To illustrate this process clearly, consider a practical scenario involving sports data. Suppose we are analyzing a table in Power BI, named **my_data**, which records the points scored by various basketball players across different teams. We aim to determine the total points accumulated by each team up to a specific row entry.

The initial structure of the data table **my_data** contains key fields such as Player, Team (our category), and Points (the value to be summed). Visualizing this initial state helps set the context for the transformation process:



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

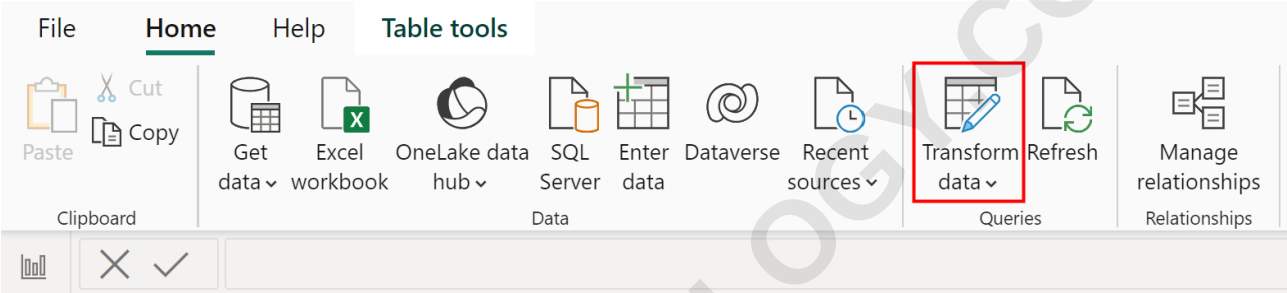
Our objective is to augment this table by creating a new column that meticulously tracks the cumulative sum of the points scored. This calculation must intelligently reset itself whenever the team category changes, providing a running total specific to each team rather than a grand total across all teams.

Step-by-Step Guide: Adding the Index Column via Power Query

As established, the first step is adding the sequential index column. This must be handled in the data transformation phase.

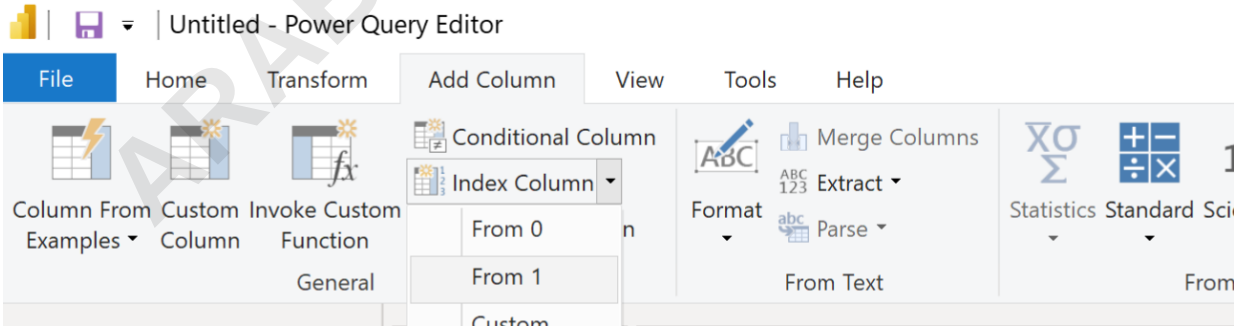
Accessing the Power Query Editor

To begin, navigate to the main Power BI Desktop ribbon. Click the **Home** tab, and then locate and click the **Transform data** icon. This action launches the separate window environment known as the Power Query Editor, which is where data shaping and preparation occurs.

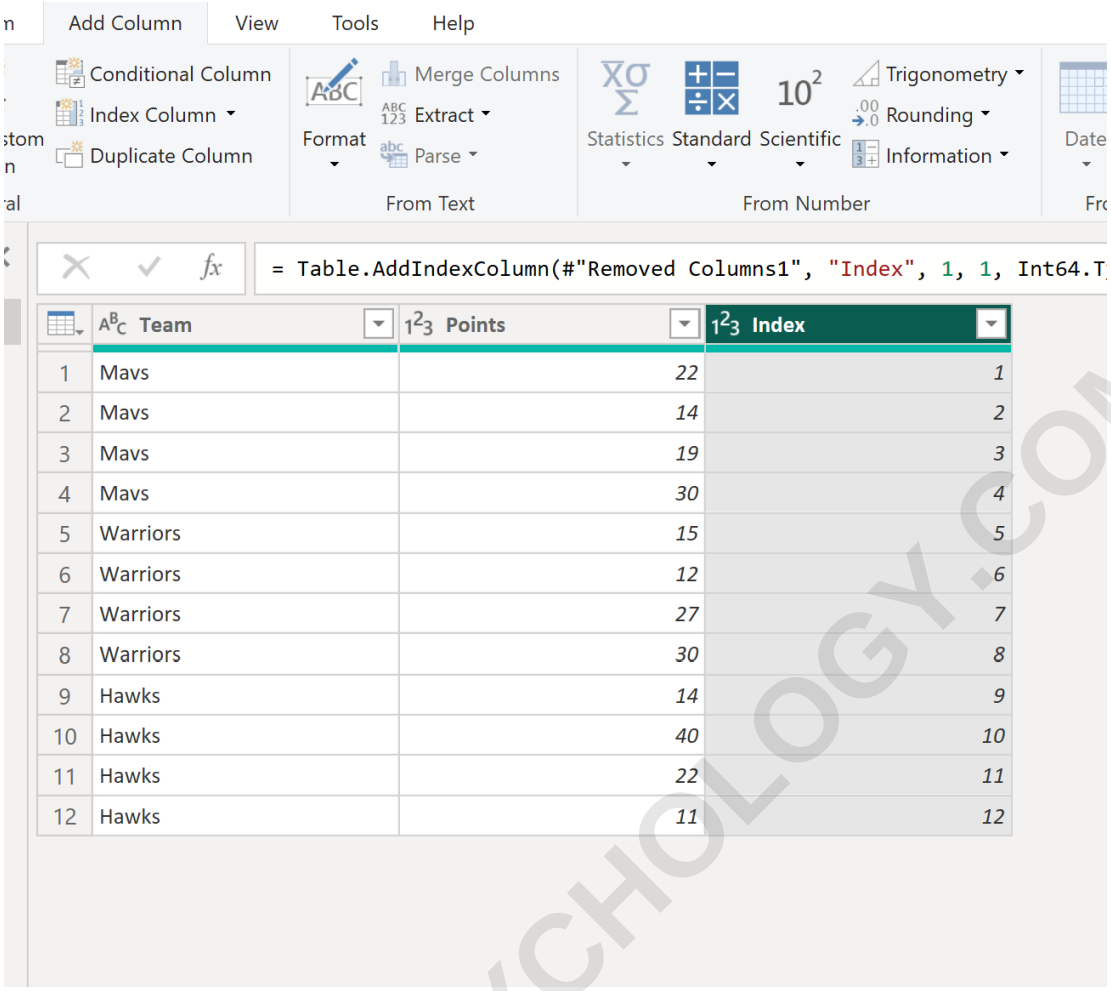


Inserting the Index Column

Once inside the Power Query Editor, select the **Add Column** tab from the top menu ribbon. Within this section, find the **Index Column** dropdown. You will be prompted to choose the starting point for the index values. Although starting from 0 is common in programming, starting **From 1** often provides a clearer, more readable sequence for basic analytical tasks.



We select **From 1** to create a numerical column ranging from 1 up to the total number of records. After applying this step, the Index column is appended to the data table, ensuring every row has a unique identifier for ordering:



	Team	Points	Index
1	Mavs	22	1
2	Mavs	14	2
3	Mavs	19	3
4	Mavs	30	4
5	Warriors	15	5
6	Warriors	12	6
7	Warriors	27	7
8	Warriors	30	8
9	Hawks	14	9
10	Hawks	40	10
11	Hawks	22	11
12	Hawks	11	12

Applying Changes and Returning to Power BI Desktop

After successfully adding the index column, you must apply the changes and load the transformed data back into the Power BI Data Model. In the Power Query Editor, navigate back to the **Home** tab and click **Close & Apply**. This finalizes the data preparation, making the new **Index** column available for use in subsequent DAX calculations.

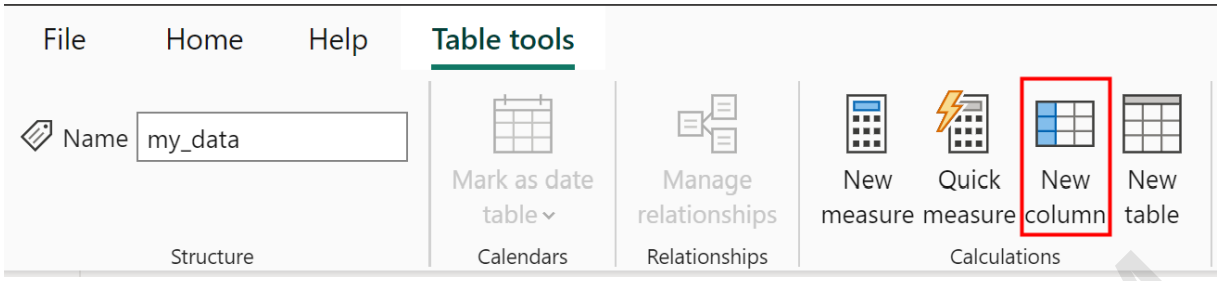
Implementing the Categorized Cumulative Sum in DAX

With the **Index** column now available in the data model, we can proceed to define the actual categorized running total calculation using DAX. Since a running total is a row-by-row calculation that depends on the context of other rows, it is best implemented as a **Calculated Column**.

Creating the New Calculated Column

In the Power BI Desktop interface, ensure you are in the Data view or Model view. Select the table (`my_data`) to which you want to add the column. Then, find the **Table tools** tab in the ribbon (it

may appear after selecting the table) and click the **New column** icon.



Defining the DAX Logic

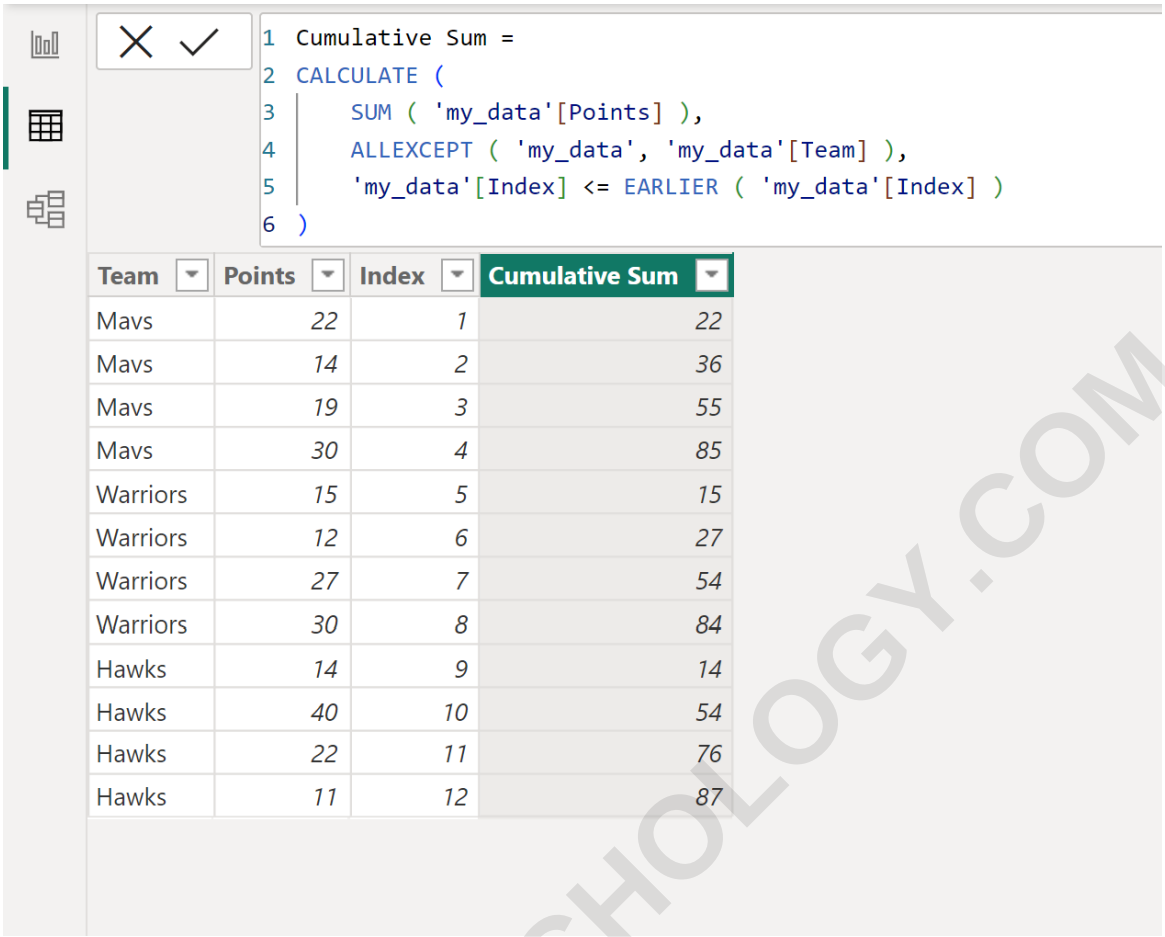
The formula bar will appear, allowing you to input the DAX expression. Carefully input the cumulative sum formula designed to group by the category:

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

Upon pressing Enter, Power BI executes this expression for every row in the `my_data` table. For each row, the cumulative sum is calculated by summing all preceding and current points, strictly filtered by the current team.

Analyzing and Validating the Results

The result of executing the CALCULATE expression is a new column named **Cumulative Sum**. This column demonstrates the desired categorized running total functionality.



```
Cumulative Sum =
CALCULATE (
    SUM ( 'my_data'[Points] ),
    ALLEXCEPT ( 'my_data', 'my_data'[Team] ),
    'my_data'[Index] <= EARLIER ( 'my_data'[Index] )
)
```

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

Observe how the **Cumulative Sum** progresses linearly for Team A (10, 25, 30). Crucially, when the table transitions to Team B, the cumulative sum resets (starts at 5). This confirms that the `ALLEXCEPT ( 'my_data', 'my_data' )` argument successfully preserved the category filter while allowing the row context to iterate correctly based on the Index.

This resulting column is now ready for visualization. You can plot this new column against the index or another sequential field in a line chart to visualize the growth rate or trend of points scored for each team, offering far greater analytical depth than simple aggregated measures.

Further Applications and Related DAX Tasks

While this tutorial focused on calculating the cumulative sum by category using an index, the underlying principles of context transition and filter modification established by the CALCULATE function are applicable to numerous advanced DAX tasks. Understanding how ALLEXCEPT and EARLIER interact is key to solving complex filtering problems where you need to reference a previous state or a specific context.

Common related tasks that use similar DAX logic include calculating year-over-year growth rates,

performing moving averages, or creating rank columns that dynamically adjust based on user-selected filters. Mastering the indexed cumulative sum provides a strong foundation for tackling these more complex modeling challenges.

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