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

Authored by

**stats writer**

January 28, 2026

The ability to calculate a cumulative sum, often referred to as a running total, is a fundamental requirement in business intelligence and data analysis. In **Power BI**, this calculation provides essential context by aggregating sequential values over a defined period or category. Instead of viewing isolated data points, the cumulative sum allows analysts to observe the overall performance and trajectory of a metric, such as total sales generated throughout the fiscal year or inventory consumption over several quarters. This methodology is indispensable for identifying significant trends, assessing growth velocity, and forecasting future outcomes with greater precision.

Achieving this powerful calculation in **Power BI** requires leveraging **Data Analysis Expressions (DAX)**. Unlike simple aggregate functions, calculating a running total demands context manipulation. This is because the calculation for any given row must reference the aggregation of all preceding rows based on a specified ordering column, typically a date or index. Understanding how to correctly structure a DAX formula for this purpose is critical for any serious Power BI user looking to build dynamic and insightful reports.

While some might mistakenly seek a simple, dedicated function named "CUMULATE," the actual implementation relies on a sophisticated combination of the **CALCULATE** function, context filtering using **ALL**, and row context transition using the **EARLIER** function. This specific pattern enables the measure or calculated column to iterate through the dataset, selectively summing values up to the current row's context. Mastering this **DAX** pattern unlocks the full potential of temporal analysis within your **Power BI** environment, ensuring that your reports are both accurate and reflect crucial business progression.

## Defining the Core DAX Pattern for Cumulative Sum

To accurately compute a running total in **Power BI**, we must construct a specific **DAX** expression that overrides the standard filter context. This is achieved primarily through the powerful **CALCULATE** function, which is essential for changing the evaluation environment of an expression. The general syntax provided below is the industry standard for calculating a cumulative sum across an ordered dimension, such as a date column.

This formula works by utilizing three key components: first, an aggregation function (like `SUM`) to specify what values are being totaled; second, a context modification function (`ALL`) to temporarily remove existing filters on the relevant table; and third, a critical filter condition that compares the current row's date value with all other dates in the table using the **EARLIER** function. This combination ensures that for every row evaluated, the sum only includes values from dates that are less than or equal to the current row's date, thus building the running total sequentially.

The following **DAX** syntax is designed specifically to calculate the cumulative sum of values contained within a particular column within your Power BI data model:

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

In this specific illustration, we are creating a new calculated column designated as **Cumulative Sum**. This column systematically computes and stores the running total derived from the numerical values present in the **Sales** column, which belongs to the table named **my_data**. This implementation requires that the 'my_data' table contains a reliable date or sequential column upon which the sorting and aggregation can depend.

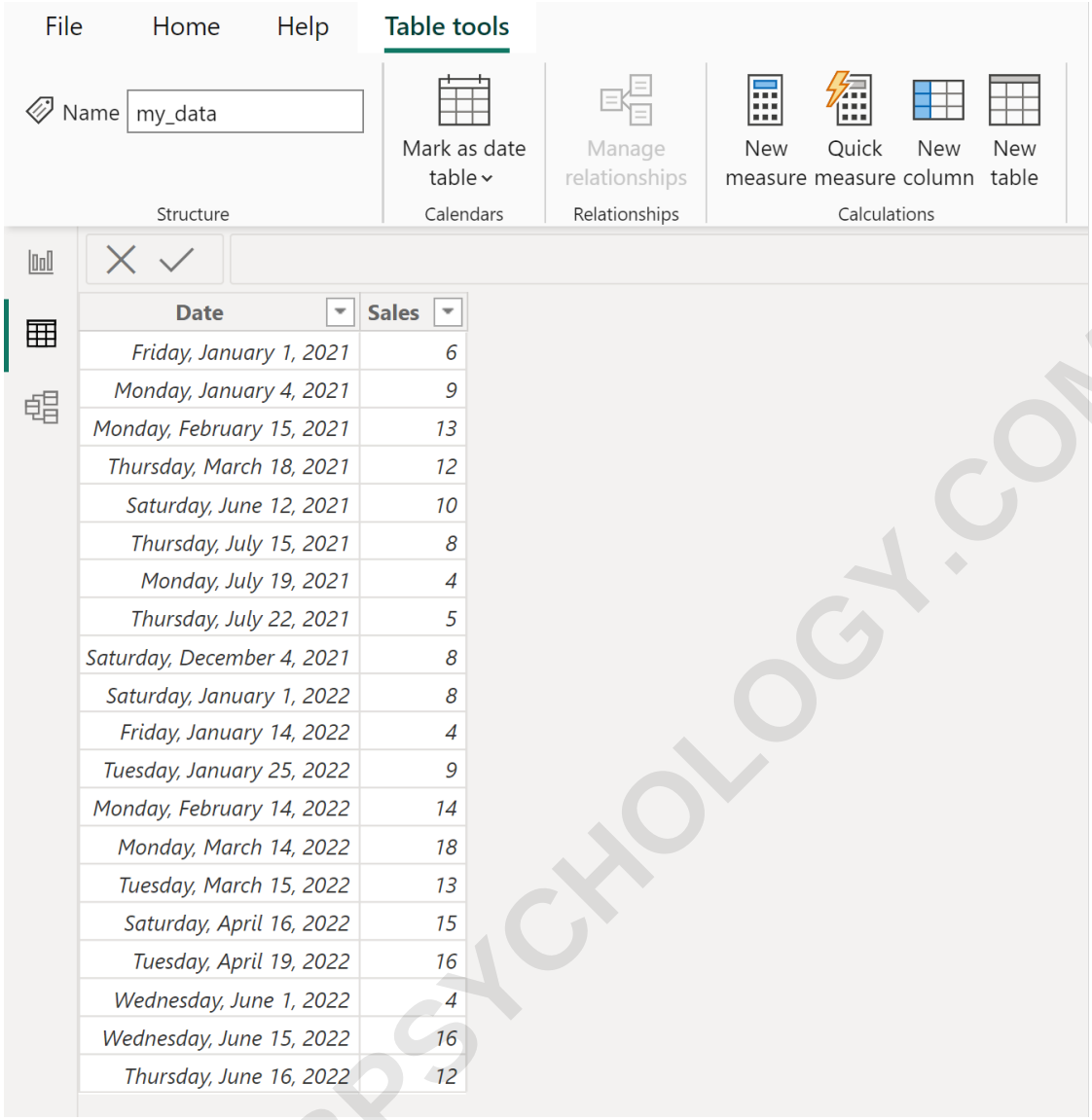
Note that while this example uses a calculated column, often in advanced scenarios, users prefer creating a measure for flexibility, especially when dealing with complex slicers or visual interactions. However, for a simple, pre-calculated running total visible on every row, the calculated column approach demonstrated here is straightforward and highly effective.

## Practical Example: Implementing the Cumulative Sum in Power BI

To fully grasp the application of this essential **DAX** formula, we will walk through a concrete example using sample data. Suppose we are tracking daily sales performance for a fictional company. Our data model includes a table named **my_data**, which meticulously records the daily sales figures alongside the corresponding date. This setup is typical for financial or operational reporting where time-series analysis is paramount.

Before introducing the cumulative sum, the dataset appears as a standard transactional log, showing only the discrete sales volume for each recorded date. Observing the daily fluctuations provides some insight, but it fails to capture the overarching progress toward a monthly or quarterly goal. It is the running total that will bridge this gap, transforming discrete observations into a powerful chronological narrative of performance.

Examine the initial structure of our **my_data** table below. It contains both the date of the transaction and the associated sales amount, establishing the necessary dimensions for our time-based calculation.




| Date | Sales |
|---|---|
| Friday, January 1, 2021 | 6 |
| Monday, January 4, 2021 | 9 |
| Monday, February 15, 2021 | 13 |
| Thursday, March 18, 2021 | 12 |
| Saturday, June 12, 2021 | 10 |
| Thursday, July 15, 2021 | 8 |
| Monday, July 19, 2021 | 4 |
| Thursday, July 22, 2021 | 5 |
| Saturday, December 4, 2021 | 8 |
| Saturday, January 1, 2022 | 8 |
| Friday, January 14, 2022 | 4 |
| Tuesday, January 25, 2022 | 9 |
| Monday, February 14, 2022 | 14 |
| Monday, March 14, 2022 | 18 |
| Tuesday, March 15, 2022 | 13 |
| Saturday, April 16, 2022 | 15 |
| Tuesday, April 19, 2022 | 16 |
| Wednesday, June 1, 2022 | 4 |
| Wednesday, June 15, 2022 | 16 |
| Thursday, June 16, 2022 | 12 |



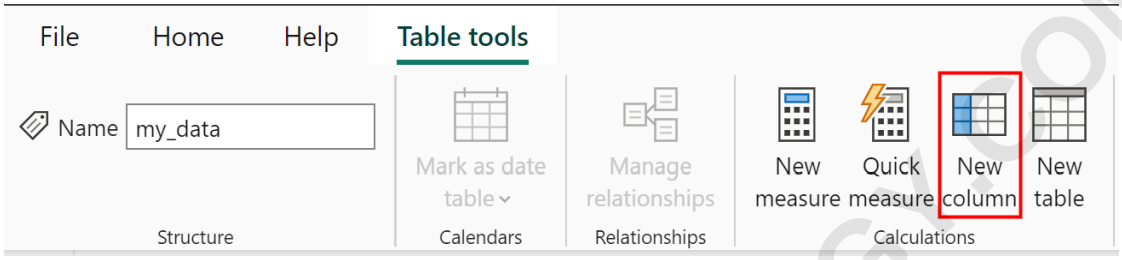
Our objective is to augment this table by incorporating a new column that reflects the aggregation of sales figures sequentially. This new column must demonstrate the total accumulated sales as of that specific date, providing immediate clarity regarding the cumulative performance trajectory. This addition is instrumental for reports aimed at visualizing growth trends or benchmarking progress against targets.

## Procedure for Adding the Calculated Column

The process for integrating this complex calculation into the data model is simple and intuitive within the Power BI Desktop interface. We begin by navigating to the modeling tools designed for data manipulation. Since we are adding a permanent column to the table structure, the "Table tools" context menu is the appropriate starting point.

To initiate the process of creating the running total column, you must first ensure you are viewing the data model or table view where the **my_data** table is visible. Then, locate the ribbon menu at the top of the interface. This will grant access to the feature required for extending the dataset schema.

Follow these precise steps: Click on the **Table tools** tab positioned along the top ribbon menu. Within the options presented, select and click the **New column** icon. This action prepares the environment and opens the formula bar, enabling the entry of our complex **DAX** expression.




## Executing the DAX Formula

Once the formula bar is active, the precise **DAX** expression for calculating the running total must be typed or pasted. The accuracy of this formula is paramount, as any syntax error will prevent the calculation from executing correctly and subsequently populating the new column. Remember that this formula relies on the interaction between row context and filter context.

The core logic resides in how the **CALCULATE** function temporarily removes all filters on the 'my_data' table using `ALL('my_data')`, but then reintroduces a powerful filter: `'my_data' <= EARLIER('my_data')`. The **EARLIER** function is essential here, as it retrieves the value of the 'Date' column from the current row context, allowing the calculation to dynamically sum up all records that occurred on or before that specific date.

Carefully input the following formula into the designated formula bar. Ensure that the table and column names (`'my_data'` and ) precisely match those in your data model, maintaining case sensitivity where required.

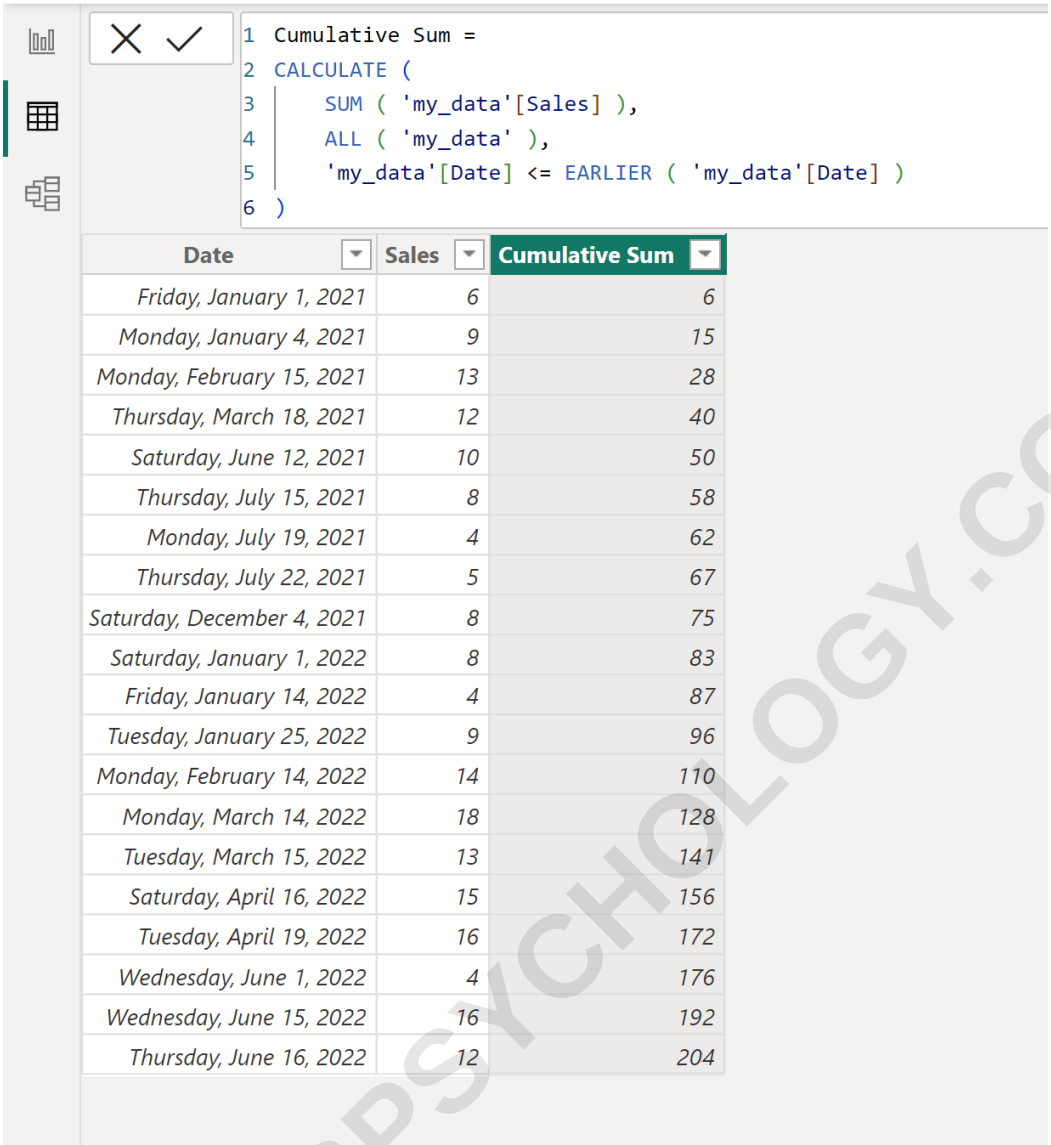
**Cumulative Sum =**
**CALCULATE (**
**SUM ( 'my_data' ),**
**ALL ( 'my_data' ),**
**'my_data' <= EARLIER ( 'my_data' )**
**)**

## Analyzing the Resulting Cumulative Data

Upon successful execution of the **DAX** code, **Power BI** immediately generates the new column, named **Cumulative Sum**. This column integrates seamlessly into the **my_data** table, transforming raw sales figures into meaningful running totals. This transformation is pivotal for visualizing growth over time in subsequent report pages.

The resulting table clearly illustrates how each row's **Cumulative Sum** value is the aggregation of its corresponding **Sales** value plus the cumulative sum calculated in the row immediately preceding it (when ordered by date). This confirms the accurate implementation of the running total logic defined by the cumulative sum concept.

Observe the output generated, confirming that the new column successfully tracks the running total of sales chronologically:



```
1 Cumulative Sum =
2 CALCULATE (
3     SUM ( 'my_data'[Sales] ),
4     ALL ( 'my_data' ),
5     'my_data'[Date] <= EARLIER ( 'my_data'[Date] )
6 )
```

| Date | Sales | Cumulative Sum |
|---|---|---|
| Friday, January 1, 2021 | 6 | 6 |
| Monday, January 4, 2021 | 9 | 15 |
| Monday, February 15, 2021 | 13 | 28 |
| Thursday, March 18, 2021 | 12 | 40 |
| Saturday, June 12, 2021 | 10 | 50 |
| Thursday, July 15, 2021 | 8 | 58 |
| Monday, July 19, 2021 | 4 | 62 |
| Thursday, July 22, 2021 | 5 | 67 |
| Saturday, December 4, 2021 | 8 | 75 |
| Saturday, January 1, 2022 | 8 | 83 |
| Friday, January 14, 2022 | 4 | 87 |
| Tuesday, January 25, 2022 | 9 | 96 |
| Monday, February 14, 2022 | 14 | 110 |
| Monday, March 14, 2022 | 18 | 128 |
| Tuesday, March 15, 2022 | 13 | 141 |
| Saturday, April 16, 2022 | 15 | 156 |
| Tuesday, April 19, 2022 | 16 | 172 |
| Wednesday, June 1, 2022 | 4 | 176 |
| Wednesday, June 15, 2022 | 16 | 192 |
| Thursday, June 16, 2022 | 12 | 204 |



## Verification of Cumulative Sum Calculation

To ensure complete understanding and confirm the formula's effectiveness, let us manually verify the calculation for the initial rows of the dataset. This step validates that the context transition and filtering executed by the **EARLIER** function operate as intended, building the total incrementally based on the chronological order established by the 'Date' column.

Understanding the individual steps of accumulation is crucial for troubleshooting similar calculations or modifying the formula to handle more complex scenarios, such as cumulative sums grouped by product category or region. The sequential nature of the calculation means that the accuracy of later rows depends entirely on the correct summation of all preceding data points.

The following analysis confirms the sequential addition:

For the first row, the cumulative sum of sales starts with the initial value: 6.
For the second row, the cumulative sales value is the sum of the first row and the second row: 6 + 9 = **15**.
For the third row, the cumulative sum aggregates the total from the second row with the third row's sales: 15 + 13 = **28**.
For the fourth row, the running total is calculated by adding the fourth row's sales to the previous cumulative total: 28 + 12 = **40**.

This step-by-step verification confirms that the **CALCULATE** and **EARLIER** combination successfully achieved the desired running total effect across the dataset. The process continues in this manner for all subsequent rows, providing a complete time-series view of accumulated sales.

## Conclusion and Further Power BI Capabilities

Calculating a running total using **DAX** is an essential skill for creating highly impactful time-series reports in **Power BI**. By utilizing the **CALCULATE** function in conjunction with context modification techniques like **ALL** and row context transition using **EARLIER**, analysts gain the power to visualize trends that simple aggregations cannot reveal. This specific DAX pattern is one of the foundational building blocks for complex analytical models.

While this article focused on the calculated column method, remember that similar DAX techniques can be used to create cumulative measures, which respond dynamically to filters and slicers applied in your report view. Depending on whether you need a static table column or a dynamic visual measure, the choice between a calculated column and a calculated measure is crucial for performance and user experience.

Mastering this cumulative sum pattern opens the door to performing other common, yet powerful, analytical tasks within Power BI. These techniques often involve manipulating the filter context to compare periods, calculate year-over-year growth, or determine rolling averages. We encourage further exploration into these related topics to maximize your capabilities within the Power BI ecosystem.

## Related Power BI Tutorials

The following resources detail how to perform other common and sophisticated data manipulation tasks within the Power BI environment: