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

Authored by

mohammed looti

January 12, 2026

Calculating a cumulative sum, often referred to as a running total, is a fundamental requirement in business intelligence for analyzing trends over time. This calculation allows analysts to view the total accumulation of a measure up to a specific point, providing critical insight into growth rates, sales performance, or inventory changes across different time periods.

In Power BI, achieving a reliable running total requires mastering the powerful functionality provided by DAX (Data Analysis Expressions). Unlike simple aggregation functions, a running total requires dynamic modification of the filter context for each row evaluated. This advanced technique typically employs a combination of the `CALCULATE`, `SUM`, `ALL`, and `EARLIER` functions to ensure that the calculation correctly sums all preceding values in the defined sequence.

Understanding the Cumulative Sum Concept

A cumulative sum is characterized by its reliance on sequence. It is not just the total of a column; rather, it is the sum of the current value and all previous values, usually ordered by a date or index column. For example, if you track daily sales, the running total for Wednesday is the sum of sales from Monday, Tuesday, and Wednesday combined. This contrasts sharply with standard column totals, which only aggregate all visible rows without regard for ordering.

Successfully calculating a cumulative sum in a tool like Power BI hinges on manipulating the Evaluation Context. When creating a calculated column, DAX evaluates the formula row by row (the row context). To calculate the running total for the current row, the formula must override the existing filters to include all rows that precede the current row's date, while simultaneously excluding all rows that follow it.

This necessity to modify the filter set dynamically is why simple functions like `SUM` or `SUMX` are insufficient on their own. We must utilize `CALCULATE`, the core function in DAX, to transition from the default row context into a modified filter context that encompasses the required historical data points.

The Core DAX Pattern for Running Totals

The standard and most robust method for creating a running total as a Calculated Column in Power BI involves a specific DAX structure. This pattern ensures that the calculation accurately respects the chronological order of your data, preventing incorrect sums or unexpected results when dealing with time series data.

You can use the following syntax in DAX to calculate the cumulative sum of values in a particular column in Power BI, leveraging powerful filter modifications:

The crucial formula uses the `CALCULATE` function to aggregate the data while imposing specific

temporal constraints:

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

This particular example creates a new column named **Cumulative Sum** that contains the cumulative sum of the values in the **Sales** column of the table named **my_data**. This method is highly effective for building running totals directly into your data model.

Breaking Down the DAX Formula Components

To fully appreciate why this pattern works, we must analyze the role of each function within the context of the running total calculation:

`CALCULATE`: This is the engine of the operation. `CALCULATE` is designed to evaluate an expression (in this case, `SUM('my_data')`) within a modified filter context. It takes the filters passed to it and applies them on top of or in place of the existing filters in the evaluation environment.
`SUM('my_data')`: This is the aggregation expression. It tells DAX what value to total up. In our scenario, we are summing the **Sales** column.
`ALL ('my_data')`: This is a powerful filter modifier. When the formula is being evaluated row by row, the current row context automatically filters the table to just that one row. `ALL('my_data')` is necessary to clear all existing filters on the entire **my_data** table. This effectively makes every row in the table visible again before the next filter condition is applied, ensuring we don't accidentally restrict the sum to just the current row.
`'my_data' <= EARLIER ('my_data')`: This is the temporal constraint that defines the cumulative nature of the sum. The `EARLIER` function is critical in a row context: it captures the value of the **Date** column from the outer row context (i.e., the date of the row currently being calculated). The resulting filter condition then says, "Only include rows where the **Date** is less than or equal to the date of the current row." This is what creates the running total effect, ensuring only historical data is included in the sum.

Understanding the interplay between `CALCULATE`, which modifies context, and `ALL` and `EARLIER`, which define the scope of the modification, is essential for mastering running totals in Power BI.

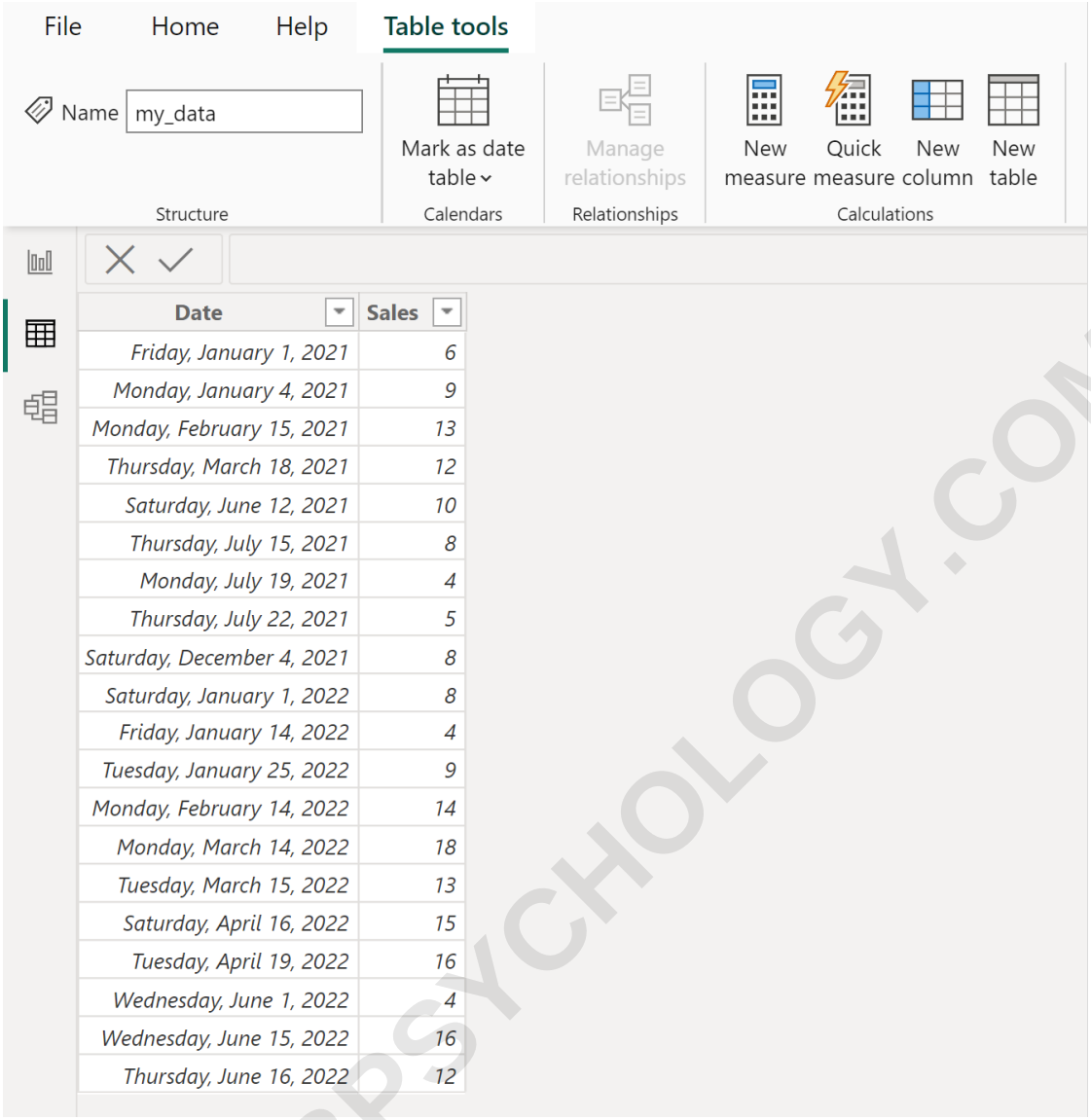
Practical Example Setup: Preparing the Data

To illustrate the implementation, let us consider a typical sales tracking scenario. Suppose we have the following table loaded into our Power BI model, named **my_data**, containing sales figures recorded on various dates. It is crucial that the table includes a proper date column to correctly sequence the running total calculation.

The following example shows how to use this syntax in practice.

Example: How to Calculate a Cumulative Sum in Power BI

Suppose we have the following table in Power BI named **my_data** that contains information about sales made on various dates by some company. Notice the structure, with distinct dates and corresponding sales figures, ready for chronological aggregation:



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 incorporate a new column that meticulously tracks the cumulative sum of the values in the **Sales** column, allowing us to visualize how total sales grow throughout the dates provided in the dataset. This column will be a Calculated Column, meaning it will be computed and stored within the table structure itself.

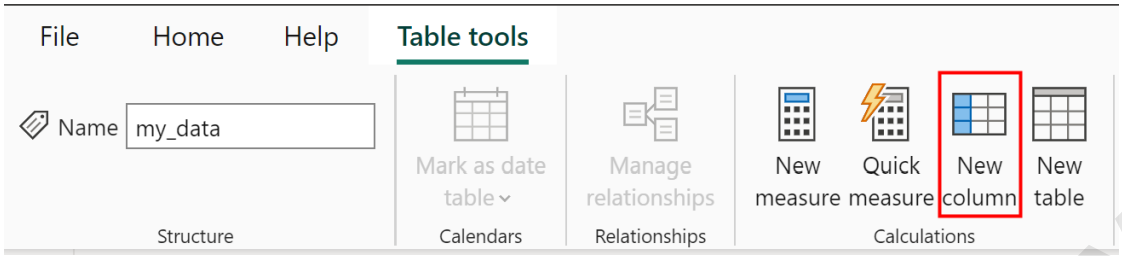
Step-by-Step Implementation of the Cumulative Sum

The implementation requires accessing the data modeling tools within Power BI Desktop and introducing the DAX formula we discussed previously.

Step 1: Create a New Calculated Column

To begin, navigate to the Data View or Model View within Power BI Desktop. Click the **Table tools**

tab along the top ribbon, then click the **New column** icon. This action opens the formula bar, ready for the DAX expression input:



Step 2: Input the DAX Formula

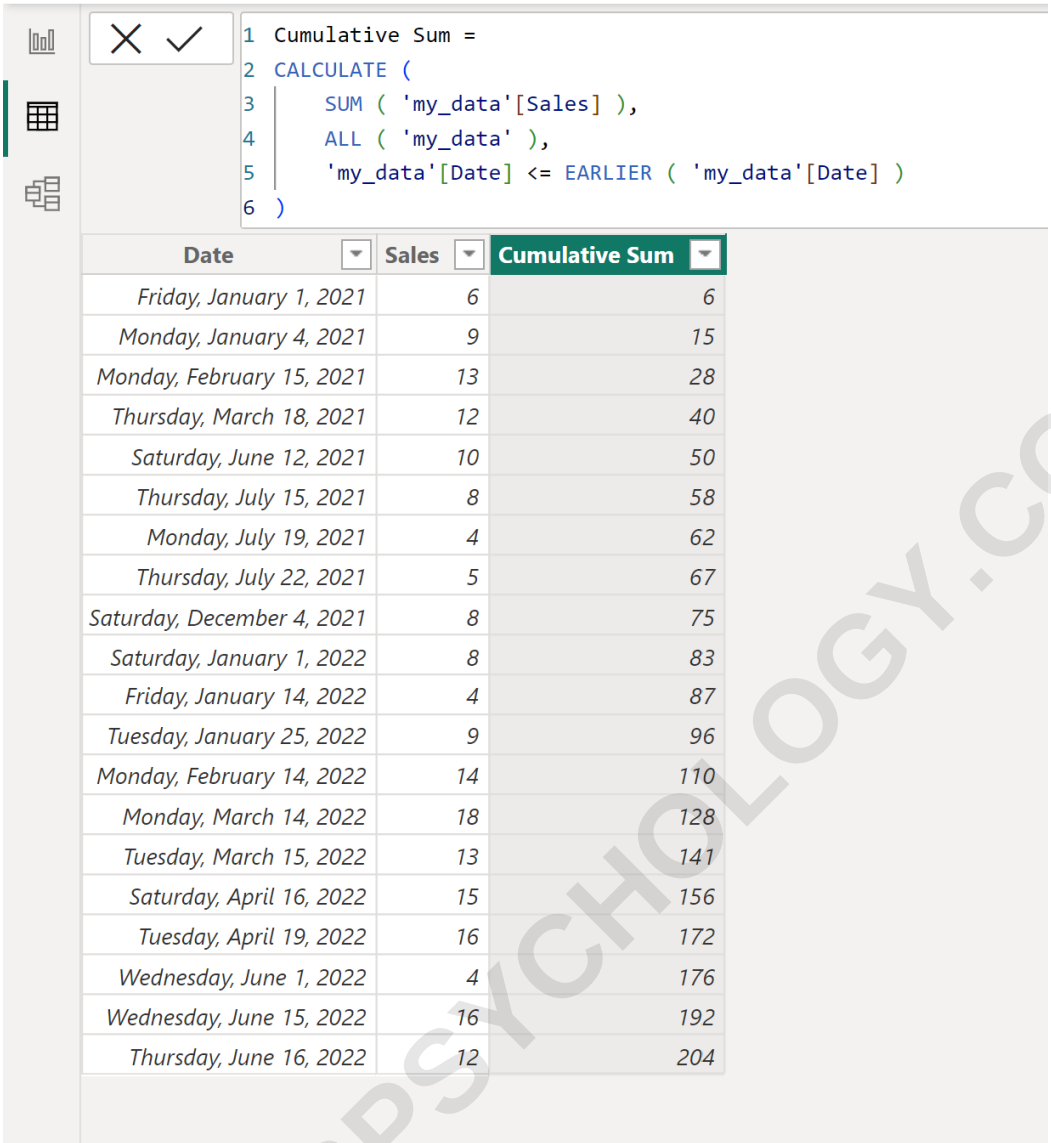
Now, meticulously type or paste the comprehensive running total formula into the formula bar. Ensure that table and column names (e.g., `'my_data'` and) exactly match the names within your data model. The formula is:

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

After pressing Enter, Power BI will process this formula for every single row in the **my_data** table, performing the sophisticated context modification required to generate the running total based on the date sequence.

Analyzing the Results and Context Transition

Upon successful execution of the DAX formula, a new column named **Cumulative Sum** will be appended to the **my_data** table. This column displays the precise running total of sales up to and including the date in that row. The resulting table confirms that the context modification was successful:



```
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

We can validate the results by manually tracing the calculation through the first few rows. The context transition facilitated by `CALCULATE` and the filtering applied by `EARLIER` ensures that each step correctly builds upon the preceding sums:

Cumulative sum of sales for first row (Date 1/1/2024): The filter includes only that row. Result: 6
Cumulative sum of sales for second row (Date 1/2/2024): The filter includes sales up to and including this date. Calculation: 6 + 9 = **15**
Cumulative sum of sales for third row (Date 1/3/2024): The filter includes sales up to and including this date. Calculation: 6 + 9 + 13 = **28**
Cumulative sum of sales for fourth row (Date 1/4/2024): The filter includes sales up to and including this date. Calculation: 6 + 9 + 13 + 12 = **40**

The running total increases monotonically, reflecting the precise accumulation of sales over time.

This resulting column is now ready to be used in visualizations, such as line charts, to easily track growth trends.

Advanced Considerations and Best Practices

While the Calculated Column approach demonstrated here is ideal for static running totals that do not need to respond to external filters (like slicers on a report page), for dynamic analysis, users should consider creating a **Measure** instead. A measure-based cumulative sum typically uses the `FILTER` function combined with `ALL` or `ALLSELECTED` over the date column, allowing the total to dynamically recalculate when visual filters are applied.

Regardless of whether you choose a column or a measure, understanding the concepts of row context, filter context, and how `CALCULATE` transitions between them is the key to generating accurate and efficient running totals in Power BI.

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