Learn How to Sum Multiple Columns in Power BI Using DAX

Authored by

Mohammed looti

November 12, 2025

When working with data aggregation in Power BI, one common requirement is calculating the sum of values across multiple columns within the same row. This horizontal summation requires a specific approach using DAX (Data Analysis Expressions), as standard aggregation functions typically operate vertically down a single column. The following syntax provides the most efficient method for achieving this sum:

Sum Points = CALCULATE(SUMX('my_data', + +))

This powerful formula, when deployed as a **Calculated Column**, achieves the desired result. It generates a new column named **Sum Points** that meticulously calculates the sum of the corresponding values found in the **Game 1**, **Game 2**, and **Game 3** columns for every distinct row in the specified table, which in this instance is named **my_data**. Understanding the role of the iterating function, SUMX, is crucial for mastering this technique.

The subsequent sections will deconstruct this formula, explain the underlying principles of DAX context, and provide a practical, step-by-step example demonstrating its implementation within the Power BI Desktop environment.

Understanding the Need for Row-Level Aggregation in Power BI

In the realm of business intelligence and data analysis, particularly within Power BI, data manipulation often revolves around two primary axes: aggregating vertically (down columns) or transforming horizontally (across rows). Standard DAX functions like `SUM()` are optimized for vertical aggregation, returning a single scalar value based on the current filter context. However, when the objective is to create a new derived value for each record--such as summing individual scores from multiple attempts--we must shift our focus to row-level operations.

This need arises frequently in financial modeling, project management, and performance tracking where metrics are spread across adjacent columns (e.g., Q1 Revenue, Q2 Revenue, Q3 Revenue). Simply adding the column references together (e.g., `+ `) works perfectly within a Calculated Column, but for complex or iterated calculations, leveraging the SUMX function provides necessary structure and control. The resulting calculated column becomes a permanent fixture of the table, ready to be used in visuals, filters, and subsequent calculations, thereby enriching the overall Data Model.

While simple column addition is straightforward, using an iterator like SUMX is often preferred because it explicitly establishes the necessary Row Context. This makes the formula robust, especially if the underlying logic were to become more complicated, perhaps involving conditional summing or nested calculations. The structure ensures that the calculation engine processes the expression row-by-row before accumulating the final total, making the intent of the calculation

perfectly clear to anyone reviewing the DAX code.

Deconstructing the DAX Syntax: SUMX and CALCULATE

The formula presented above utilizes two of the most fundamental and powerful DAX functions: SUMX and CALCULATE. Understanding their roles is key to advanced data manipulation in Power BI.

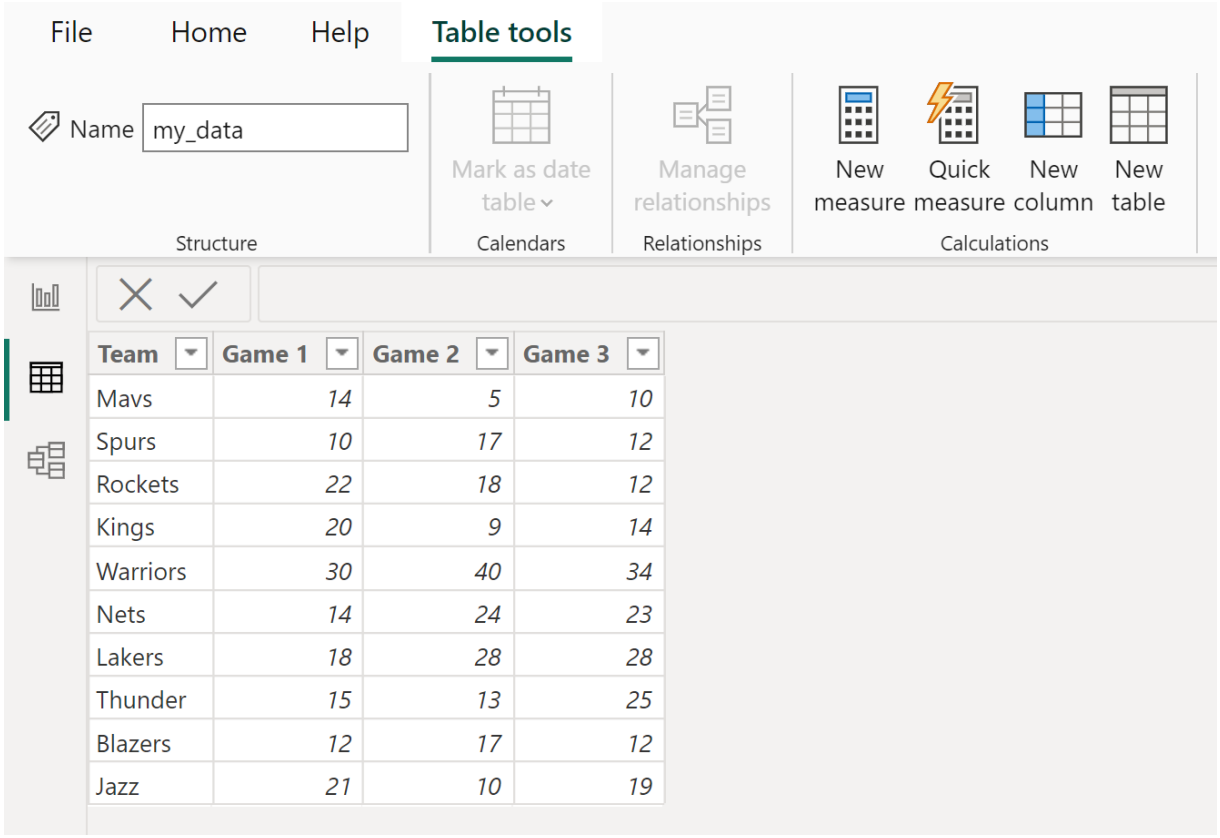
The primary engine for this summation is the SUMX function. As an iterator function, SUMX requires two arguments: the table to iterate over and the expression to evaluate for each row. In our example, `SUMX('my_data', + + )`, the function proceeds as follows:

It selects the table **'my_data'**.
It establishes a temporary Row Context for the first row.
It evaluates the expression **+ +** using the values from that specific row.
It stores the resulting sum.
It repeats this process for every subsequent row in the **'my_data'** table.
Finally, it returns the accumulated sum of all those individual row calculations.

While SUMX handles the iteration, the use of CALCULATE in the overall formula is often a matter of best practice, especially when defining a measure, although here it surrounds a calculated column expression. CALCULATE is the function that modifies the filter context. While the core calculation (the summation itself) establishes its own Row Context via SUMX, wrapping the expression in CALCULATE ensures that any external filter contexts (e.g., filters applied by slicers or report visuals) are properly managed or ignored, depending on the requirements of a more complex scenario. In the simplest case of a calculated column, the expression `SUMX('my_data', + + )` might suffice, but using `CALCULATE` adds robustness and prepares the user for more advanced DAX patterns.

Step-by-Step Example: Implementing the Solution

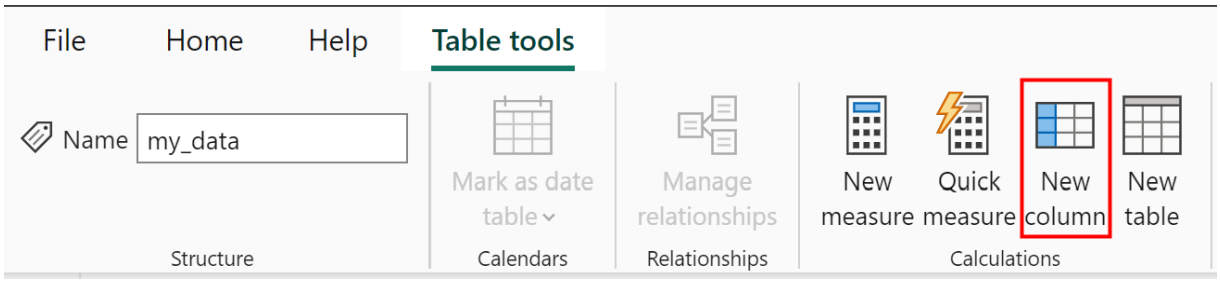
To demonstrate the practical application of this formula, let us consider a typical scenario involving sports performance data. Suppose we have imported the following dataset into Power BI, which is named **my_data**. This table details the points scored by various basketball players across three distinct games:



Team	Game 1	Game 2	Game 3
Mavs	14	5	10
Spurs	10	17	12
Rockets	22	18	12
Kings	20	9	14
Warriors	30	40	34
Nets	14	24	23
Lakers	18	28	28
Thunder	15	13	25
Blazers	12	17	12
Jazz	21	10	19

Our goal is to create a single, consolidated column that clearly displays the total accumulated points for each player across all three competitive games. This calculation will provide immediate insight into overall performance metrics within the Data Model.

The process begins by ensuring you are within the correct data view in Power BI Desktop. To initiate the creation of a new column, navigate to the contextual ribbon menus. First, click on the **Table tools** tab at the top of the interface. Once the tab is active, locate and click the **New column** icon. This action immediately prompts the DAX formula bar to appear, ready for input:



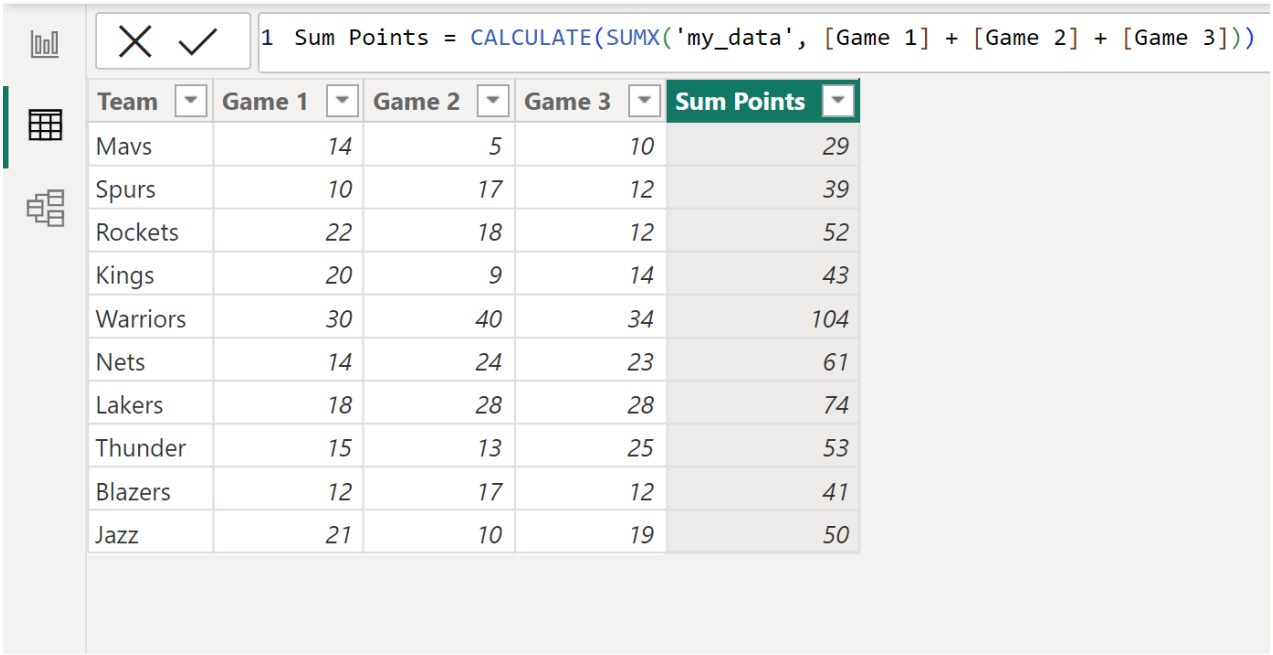
Next, precisely type the previously defined aggregation formula into the formula bar. It is essential to ensure that the table and column names within the formula exactly match those in your Data Model. For this example, we use **'my_data'** as the table name and , , and as the column names:

Sum Points = CALCULATE(SUMX('my_data', + +))

Upon pressing Enter, the DAX engine processes the calculation across all rows. A new column is instantly materialized within the **my_data** table, automatically named **Sum Points** based on the formula definition. This column now dynamically displays the accurate sum of points scored by each player across the entire set of three games, successfully achieving the horizontal summation goal.

Analyzing the Results and Data Context

Following the execution of the DAX formula, the resulting table clearly illustrates the effectiveness of the SUMX iteration. The newly created column, **Sum Points**, is appended to the right of the existing columns, providing the total score for every record:



Team	Game 1	Game 2	Game 3	Sum Points
Mavs	14	5	10	29
Spurs	10	17	12	39
Rockets	22	18	12	52
Kings	20	9	14	43
Warriors	30	40	34	104
Nets	14	24	23	61
Lakers	18	28	28	74
Thunder	15	13	25	53
Blazers	12	17	12	41
Jazz	21	10	19	50

By examining the output, we can verify that the row-level calculation performed by the SUMX function is accurate for each player entry:

The **Mavs** player's total points are calculated as 14 (Game 1) + 5 (Game 2) + 10 (Game 3) = **29** points.
The **Spurs** player's total points are calculated as 10 (Game 1) + 17 (Game 2) + 12 (Game 3) = **39** points.
The **Rockets** player's total points are calculated as 22 (Game 1) + 18 (Game 2) + 12 (Game 3) = **52** points.
The **Lakers** player's total points are calculated as 19 (Game 1) + 10 (Game 2) + 11 (Game 3) = **40**

points.

This newly calculated column is now an integral part of the **my_data** table, functioning just like any other imported column. It can be utilized immediately in visualizations, filtered by various criteria (such as team or location), or used as a base column for further, more complex DAX calculations, such as ranking players or determining overall team averages. The advantage of using a calculated column for this task is that the total value is fixed per row, unlike a measure, which would recalculate based on the filter context of the visualization where it is placed.

Furthermore, the choice of using an iterating function like SUMX over simple column addition is a deliberate decision in sophisticated Data Model design. While the addition operator works for simple numeric columns, SUMX is essential when the aggregation involves measures or calculations that rely on context transition, ensuring maximum flexibility and compatibility should the source data structure or complexity increase in the future.

Conclusion: Mastering Horizontal Aggregation

The ability to correctly perform horizontal aggregation--summing values across multiple columns within the same row--is a fundamental skill for any Power BI developer. By leveraging the specific syntax involving the SUMX iterator, we ensure that the calculation correctly traverses the table, evaluates the necessary expression in the correct Row Context, and delivers accurate results for every record.

Whether you choose to incorporate the optional CALCULATE function for enhanced context management or rely solely on the SUMX iteration, this method remains the authoritative standard for transforming disparate column data into meaningful, row-level totals. Mastering this DAX pattern significantly expands your capability to prepare and shape data for robust reporting and visualization in Power BI.

Additional Resources

The following tutorials explain how to perform other common tasks in Power BI, providing further insight into data transformation and DAX utilization: