

How to Find the Maximum Value Across Multiple Columns in Power BI

Authored by
stats writer

January 28, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Find the Maximum Value Across Multiple Columns in Power BI*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128183>

Determining the maximum value across several columns is a common requirement in data analysis, particularly when working with tools like Power BI. This capability is crucial for identifying peak performance, highest recorded metrics, or critical outliers within a dataset. While multiple approaches exist, the choice of method often depends on where you prefer to execute the transformation: within the data preparation stage using the Power Query Editor or later in the data model using Data Analysis Expressions (DAX).

The immediate goal is usually to create a new derived column that summarizes the row-wise maximum from a specified set of numerical fields. Both the graphical user interface (GUI) approach in Power Query and the formulaic approach using **DAX** offer reliable solutions. For raw data preparation and cleaning, the Power Query method is generally preferred for its efficiency and ease of implementation, as it performs the calculation during the data load phase. However, if the calculation needs to respond dynamically to user interactions or filters applied in reports, the DAX method becomes indispensable.

We will explore two primary methodologies for achieving this aggregation: the simple statistical functions available in the Power Query Editor's ribbon, which uses the **M language** behind the scenes, and the more flexible, calculation-intensive functions like **MAX** and **MAXX** within the DAX environment. Understanding both provides data analysts with the versatility required to handle complex modeling scenarios effectively.

Find Max of Multiple Columns in Power BI

Method 1: Utilizing the Power Query Editor Interface

The most straightforward and often the easiest way to compute the maximum value across several columns in Power BI is by utilizing the built-in statistical functions available within the **Power Query Editor**. This method is graphical, requiring no manual coding, making it highly accessible for analysts who are less familiar with the underlying M language syntax. It involves selecting the target columns and applying a standard aggregation operation directly from the ribbon interface.

This technique is superior when the maximum calculation needs to be static and integrated directly into the foundational dataset structure. Since Power Query handles the data transformation before it enters the main data model, the resulting column is treated as a standard column. This approach significantly optimizes performance, as the calculation is executed once during the data refresh process rather than being recalculated dynamically every time a report element interacts with it.

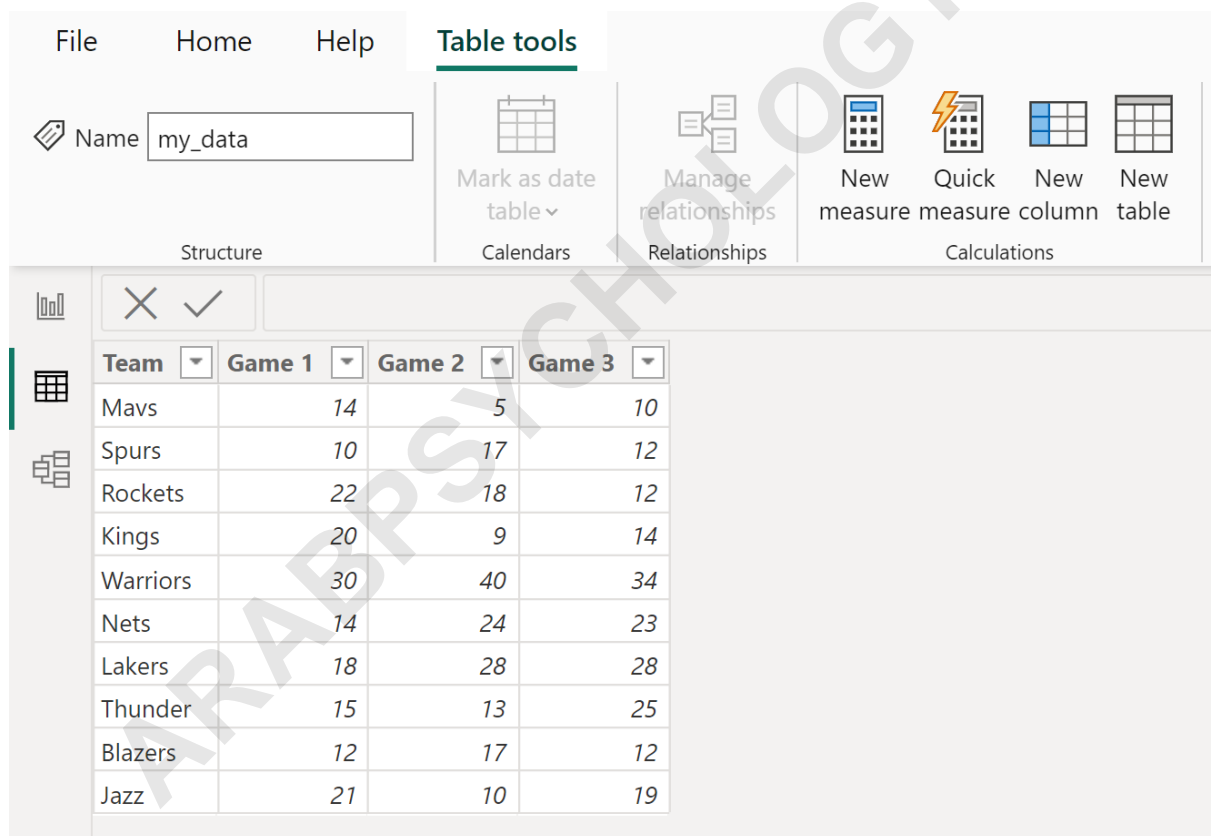
To implement this method, users must first access the Power Query environment by clicking the **Transform data** button located on the Home tab of Power BI Desktop. Once the editor opens, the relevant table must be selected. The transformation itself relies on the **Add column** tab, where

specialized functions like Maximum, Minimum, Average, and Sum are grouped under the **Statistics** command. This streamlined process ensures accuracy and consistency in data preparation.

Step-by-Step Example: Finding the Maximum via Power Query

Let us walk through a practical demonstration of finding the row-wise maximum using the Power Query Editor. Suppose we have a dataset named **my_data** detailing the points scored by various basketball players across three distinct games. Our objective is to determine the highest score achieved by each player across the scope of these three games.

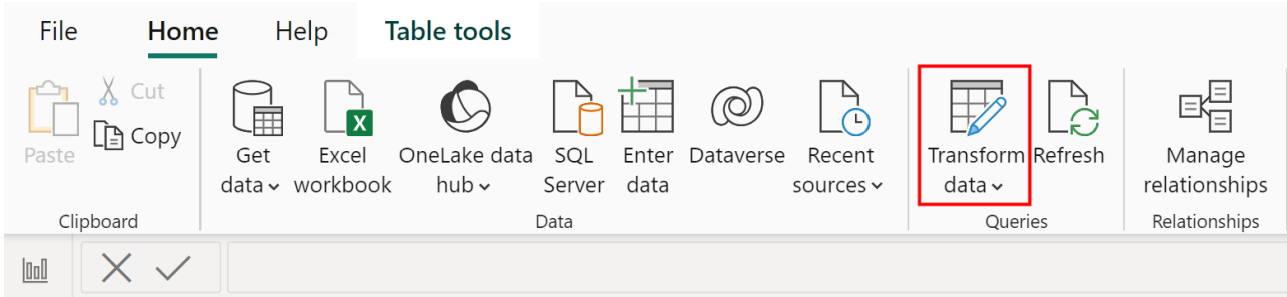
The initial structure of the data table resembles the following visualization, showing distinct columns for each game's performance metrics:



The screenshot shows the Power BI interface with the 'Table tools' ribbon active. The ribbon includes options like 'Mark as date table', 'Manage relationships', and 'Calculations'. Below the ribbon, a table is displayed with the following data:

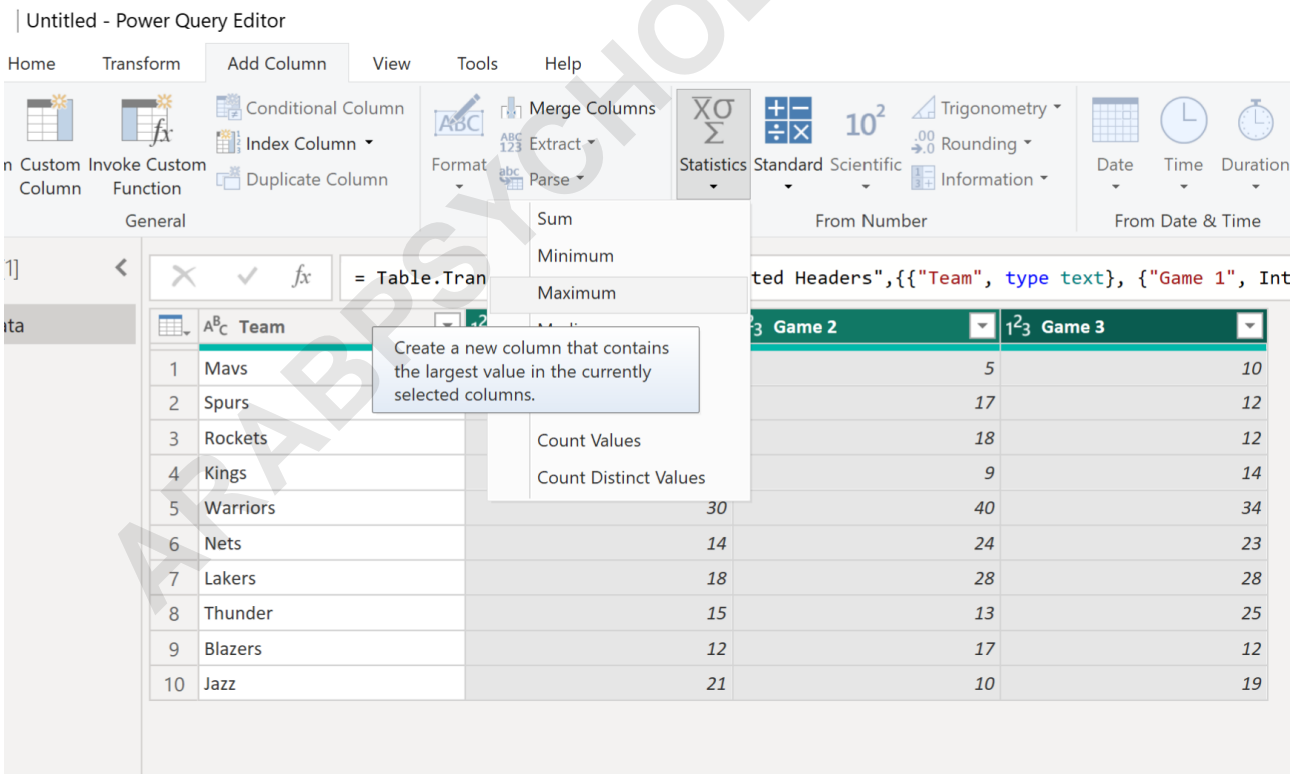
Team	Game 1	Game 2	Game 3
Mavs	14	5	10
Spurs	10	17	12
Rockets	22	18	12
Kings	20	9	14
Warriors	30	40	34
Nets	14	24	23
Lakers	18	28	28
Thunder	15	13	25
Blazers	12	17	12
Jazz	21	10	19

To initiate the transformation process, navigate back to the main **Power BI** Desktop view and click the **Transform data** icon found within the **Home** tab. This action launches the dedicated environment for data manipulation:



Within the **Power Query Editor (Power Query)**, the procedure involves first selecting the fields we wish to compare. Switch to the **Add column** tab along the top ribbon. Then, utilizing the **Ctrl** key, select the column headers corresponding to **Game 1**, **Game 2**, and **Game 3** simultaneously. Once the columns are highlighted, locate and click the **Statistics** drop-down menu and choose the **Maximum** option.

This sequence of steps triggers the creation of a new column based on the selected aggregation logic, as illustrated below:



The resulting table incorporates a new column automatically labeled **Maximum**, which successfully calculates the highest score achieved for each row across the three specified games:

fx = Table.AddColumn(#"Changed Type", "Maximum", each List.Max({[Game 1], [Game 2], [Game 3]}), Int64.Type)

	A ^B C Team	1 ² 3 Game 1	1 ² 3 Game 2	1 ² 3 Game 3	1 ² 3 Maximum
1	Mavs	14	5	10	14
2	Spurs	10	17	12	17
3	Rockets	22	18	12	22
4	Kings	20	9	14	20
5	Warriors	30	40	34	40
6	Nets	14	24	23	24
7	Lakers	18	28	28	28
8	Thunder	15	13	25	25
9	Blazers	12	17	12	17
10	Jazz	21	10	19	21

Analyzing the output demonstrates the successful calculation:

The max points scored by the **Mavs** player across the three games was **14**.

The max points scored by the **Spurs** player across the three games was **17**.

The max points scored by the **Rockets** player across the three games was **22**.

Method 2: Leveraging DAX Functions for Dynamic Calculations

While Power Query is excellent for static data preparation, situations often arise where the maximum calculation must be performed dynamically within the data model itself, especially when interacting with visual filters or slicers. For these scenarios, DAX, or Data Analysis Expressions, offers powerful functions suitable for creating calculated columns and measures. When calculating the maximum across multiple columns within the same row, the standard **MAX** function is the appropriate tool.

It is critical to distinguish between row context and filter context when using DAX. When creating a calculated column, DAX inherently operates in a **row context**, meaning the formula evaluates row by row. This allows the **MAX** function to easily compare the values of specified columns on that particular row and return the largest one. The syntax is intuitive: simply list the column references within the MAX function, separated by commas.

For example, to replicate the result achieved in Power Query but using DAX, you would create a new calculated column in the modeling view using a formula similar to `Max Score = MAX('my_data', 'my_data', 'my_data')`. This approach requires that the columns being compared reside within the same table, leveraging the inherent row context provided during the creation of a calculated column.

DAX Implementation: Understanding MAX vs. MAXX

When dealing with maximum values in DAX, analysts often encounter both **MAX** and **MAXX** functions, and understanding their distinct purposes is vital for correct implementation. The standard **MAX** function is designed to take two or more scalar expressions (like column names) as arguments and return the greatest value among them. As demonstrated, this is the ideal choice for row-wise comparison across static columns.

In contrast, the **MAXX** function is an iterator function. It takes a table and an expression as arguments, iterating row by row over the specified table and evaluating the expression for each row, ultimately returning the maximum result of that expression. This function is typically reserved for more complex calculations, such as finding the maximum value of a measure across a filtered subset of data, or finding the maximum of a computed column that does not yet exist.

Although **MAXX** can technically achieve the same result as **MAX** for comparing fixed columns (e.g., `MAXX('my_data', MAX('my_data', 'my_data', 'my_data'))`), this usage is redundant and inefficient compared to the simple **MAX** function. The efficiency gains provided by sticking to the simpler **MAX** function when comparing column values within the row context make it the preferable DAX solution for this specific task.

Code Example: Implementing DAX MAX

To provide a clear reference for the DAX approach, here is the structure required to create a calculated column that finds the maximum score across the three numerical fields. This formula is entered directly into the formula bar when creating a new column in Power BI Desktop Modeling View.

```
Max_Score_DAX = MAX (
'my_data',
'my_data',
'my_data'
)
```

This concise expression efficiently leverages the internal calculation engine of DAX to process each row. If you need to include more columns in the maximum calculation, you simply extend the list of arguments within the parentheses. Note that unlike the Power Query method, which required selecting columns via the GUI, the DAX method necessitates explicit referencing of the table and column names using the standard `'Table'` syntax.

Choosing the Right Approach: Power Query vs. DAX

The determination of the "best" method hinges entirely on the analytical requirement and the desired location of the transformation within the data pipeline. If the data source requires consistent, pre-aggregated values that are fundamental to the dataset regardless of user interaction, the Power Query (or M language) approach is highly recommended. It results in a cleaner data model by pushing complex transformations back to the source ingestion phase, thereby minimizing the computational load during report viewing and interaction.

Conversely, if the maximum value must be responsive to filtering, or if the columns being compared are themselves results of complex measures or calculations that only exist in the data model, then Power Query is insufficient. In such dynamic scenarios, a DAX calculated column or measure using **MAX** (or possibly **MAXX** for iterative scenarios) is the mandatory choice, as DAX operates over the fully loaded data model and respects the current filter context.

A good rule of thumb for efficient Power Query development is to perform all possible data shaping and column manipulations in the Editor, reserving DAX primarily for calculations that rely on complex relationships, time intelligence, or dynamic filtering. Finding the maximum across existing numerical columns is a prime example of a task ideally suited for the powerful, yet simple, graphical tools available in the Power Query environment.

Summary and Conclusion

In summary, the most efficient and recommended technique for finding the maximum value among multiple columns in Power BI depends on the context. For static, pre-loaded data transformation, the **Maximum statistical function** within the **Power Query Editor** is fast, reliable, and requires no formula writing. For dynamic calculations within the data model that must respond to report interactions, the **MAX function** in DAX provides the necessary flexibility and is the appropriate tool for comparing column values row-by-row.

Both methods are efficient and accurate, but the Power Query approach excels in ease of implementation and performance optimization for baseline data structure definitions. Utilizing the GUI tool avoids manual syntax errors and streamlines the process of adding basic statistical aggregations, making it the preferred starting point for most analysts tackling this requirement.

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