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

Authored by

mohammed looti

January 11, 2026

Determining the maximum value across several columns within a dataset is a common requirement in data analysis, particularly when working with Power BI. The standard process involves leveraging either the **MAX** function or the more versatile **MAXX** function, typically implemented within a calculated column or a measure. These Data Analysis Expressions (DAX) functions provide the necessary structure to iterate through rows or evaluate expressions to locate the highest numerical entry across the specified fields. By calculating this maximum value, analysts can derive crucial insights, such as identifying peak performance indicators or extreme data points. The result can then be seamlessly integrated into various visuals for comparative analysis, often refined using built-in features like filters and slicers to target specific subsets of the data.

While DAX provides powerful iterative capabilities, the most straightforward and efficient approach for finding the maximum value across multiple static columns in the data source is often achieved directly within the Power Query Editor, which handles the ETL (Extract, Transform, Load) operations before the data even reaches the data model. This method, which utilizes the built-in **Maximum** function under the **Add column** tab, is highly recommended for transformation tasks that require column-level aggregations and should persist in the imported table structure.

Understanding Maximum Value Calculation in Power BI

In data modeling, especially within Power BI, the need to compare data points horizontally across a single row--such as comparing three different scores for a single entity--is frequent. Unlike vertical aggregations (e.g., summing a column), horizontal row-level calculations require specific engine capabilities. When dealing with numerical data, identifying the highest value among these horizontal fields provides a critical metric for performance tracking or identifying a peak event.

This capability is critical for scenarios where we need a row-by-row comparison across attributes rather than a summary aggregation for the entire table. For instance, if a table tracks sales performance across various regions (Region A, Region B, Region C) for a single product, finding the maximum value for each row would instantly identify the best-performing region for that product's specific observation period. While powerful, the choice of function (MAX or MAXX) and implementation method (DAX or Power Query) depends heavily on the specific data requirements and performance goals of the overall report.

Power Query vs. DAX: Choosing the Right Tool

The decision to use Power Query (M language) or DAX is fundamental in Power BI development. Power Query handles data preparation, cleaning, and structural changes during the initial data load, focusing on column-level and table-level transformations. DAX, conversely, is the language of the data model, focusing on calculations, aggregations, and filtering logic that occur after the data has been loaded into the engine. When the goal is to create a new, persistent column whose value

is derived directly from a simple comparison of existing columns within the same row--as is the case when finding the maximum of three game scores--Power Query often provides the cleanest, fastest, and most easily maintainable solution.

Using Power Query means the calculation is executed only once during the data refresh process, and the resulting column is stored permanently in the data model. This approach minimizes the strain on the DAX engine during report interaction, resulting in faster visual load times and improved user experience. If, however, the maximum value needed to be dynamically calculated based on context filtering (e.g., finding the maximum score only for players belonging to a certain team, regardless of the row structure), DAX would be the appropriate tool, utilizing iterative functions like **MAXX** within a measure.

The Easiest Method: Utilizing the Power Query Editor

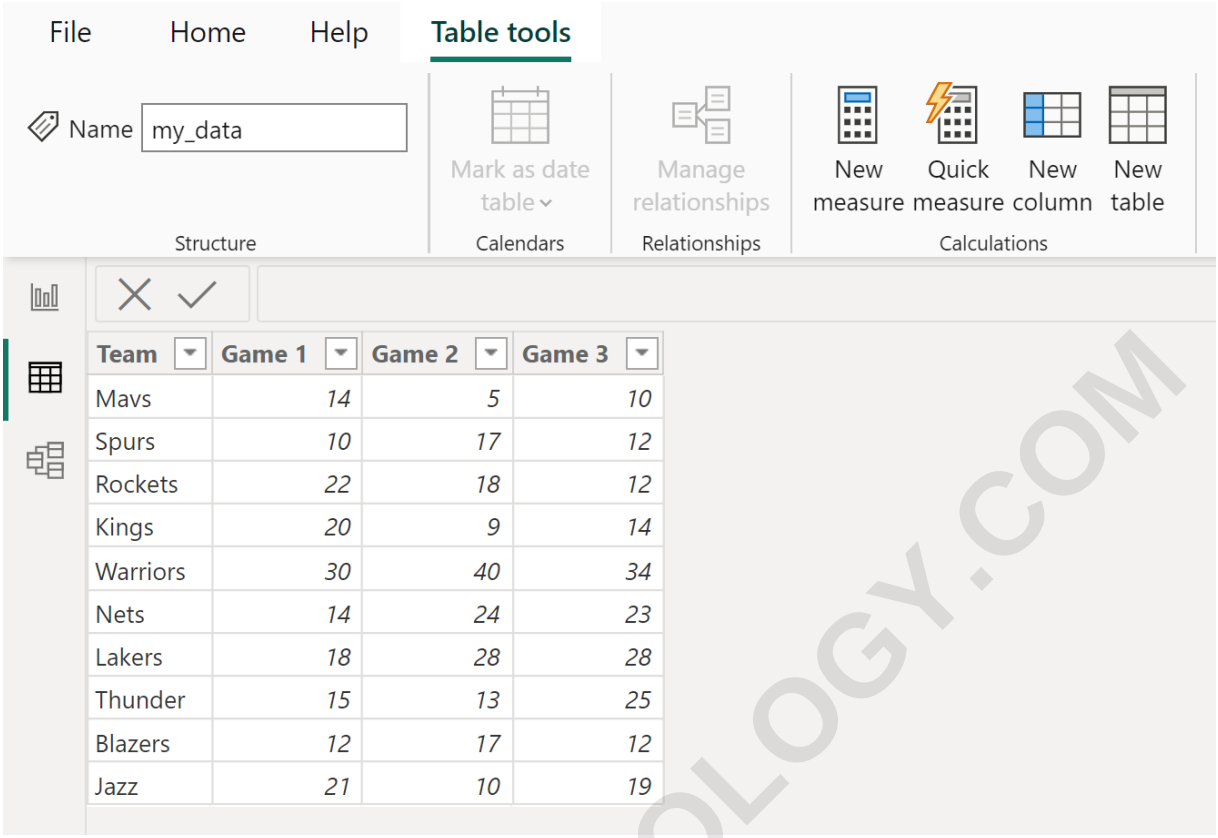
As noted, the most straightforward way to find the maximum value across multiple columns for every row is by utilizing the built-in **Maximum** function available within the **Add column** tab of the Power Query Editor. This method is highly intuitive and requires no manual coding in the M language, making it accessible even for novice Power BI users. It generates a step in the Power Query process that systematically compares the values in the selected columns for each row and outputs the highest found value into a new column, effectively automating a complex conditional logic step.

This approach is powerful because it handles potential non-numeric entries or nulls robustly, adhering to the data transformation standards enforced by Power Query. By selecting the target columns and applying the simple statistical function, the editor translates this action into a precise M formula that is applied consistently across the entire table. We will now proceed with a practical demonstration using this exact technique, showing how to transform a source table to derive the maximum observed value across three distinct observation periods.

Step-by-Step Example: Finding the Max Across Games

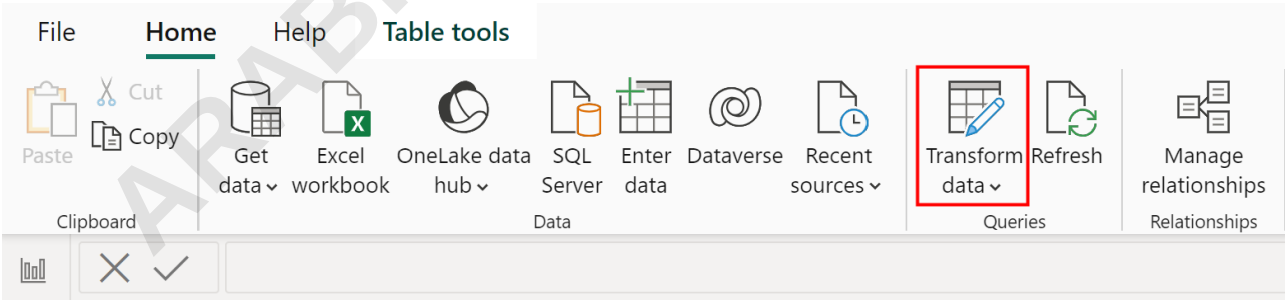
We begin our practical illustration by assuming we have imported a dataset into Power BI. This dataset, named **my_data**, tracks the individual points scored by various basketball players across three separate games. Our objective is to calculate the single highest score achieved by each player across these three games and display it in a new column.

The initial structure of the data table looks like this:



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 begin the transformation process, navigate to the **Home** tab of the Power BI Desktop interface and click the **Transform data** icon. This action launches the dedicated Power Query Editor window, where all pre-load data preparation takes place. This editor is essential for modifying the structure and content of your data before it is committed to the final data model.

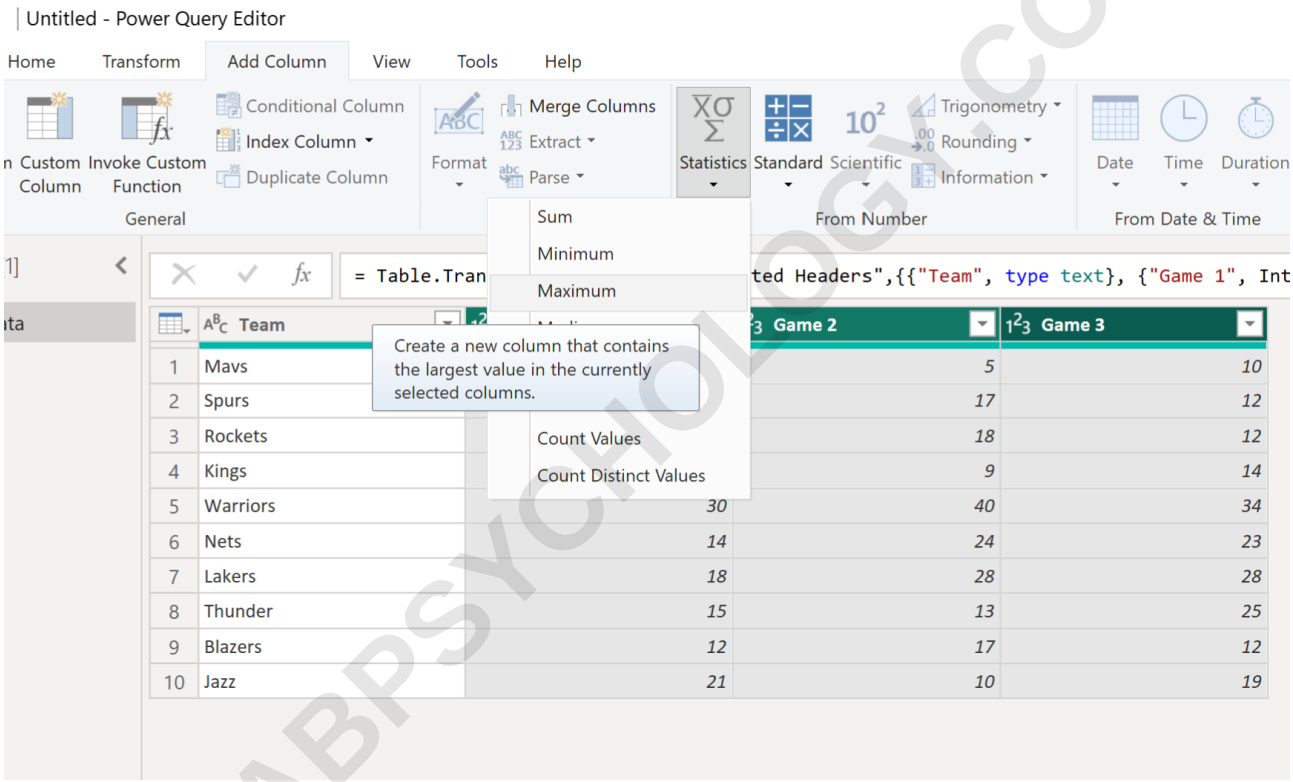


Applying the Maximum Function in Power Query

Once inside the Power Query Editor, locate and select the **Add column** tab along the top ribbon interface. This section contains various tools dedicated to creating new columns based on calculations, conditions, or transformations of existing data fields. Before applying the function, we must first identify the columns that are subject to the comparison. Hold the **Ctrl** key and click on

the column headers for **Game 1**, **Game 2**, and **Game 3** to select them simultaneously. The order of selection does not affect the final result for a maximum calculation, but sequential selection is generally good practice.

With the required columns highlighted, navigate to the **From Selection** group within the **Add column** tab. Click on the **Statistics** icon. This dropdown menu presents a variety of aggregate functions, such as Sum, Average, Minimum, and Maximum. Select **Maximum** from this menu. This step instructs Power Query to perform a row-level comparison across the three selected columns and determine the highest value observed for that specific player's row.

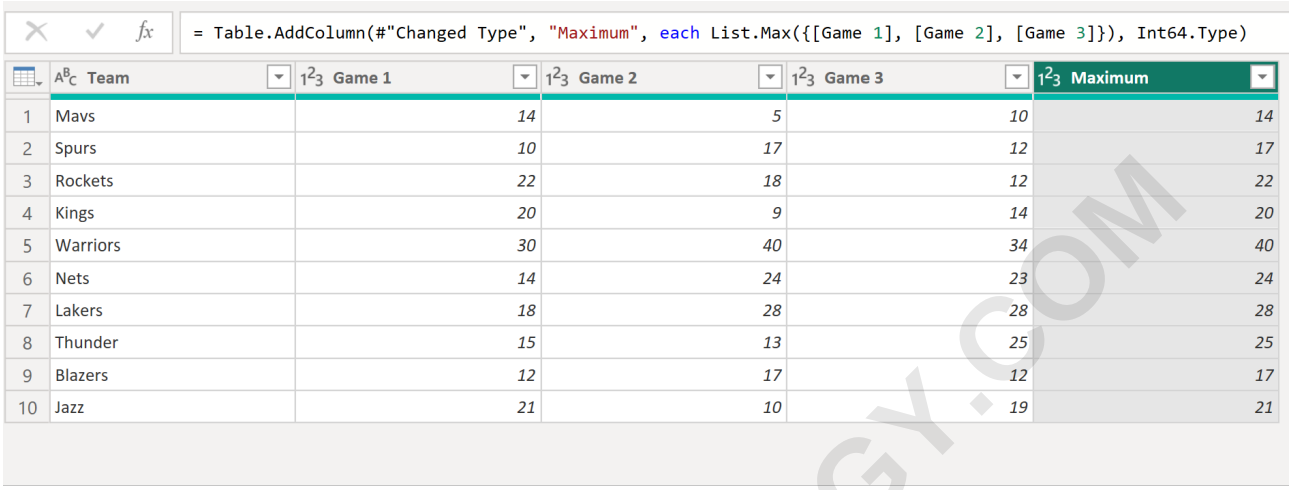


Upon execution, the Power Query Editor automatically creates a new column, typically titled **Maximum** (though it is best practice to rename it later for clarity, such as "Max Score"). This column now houses the highest score achieved by each corresponding player across the three games, fulfilling the analytical objective. After confirming the results and ensuring the data type is correctly set to Whole Number, close and apply the changes to load the transformed data back into the main Power BI data model.

Reviewing the Transformation Results

The newly created **Maximum** column provides immediate insight into peak performance. The transformed table, now visible back in the Power BI Desktop environment, displays the derived

maximum values alongside the original game scores, allowing for easy verification and subsequent analysis. This transformation is now a permanent part of the data model structure until the Power Query steps are altered.



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

	Team	Game 1	Game 2	Game 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 yields clear results regarding the players' highest individual performances:

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**.

This process demonstrates how effortlessly the maximum value calculation can be managed within Power Query, providing a robust, pre-calculated metric ready for reporting.

Alternative Approach: Implementing DAX Functions (MAX and MAXX)

While the Power Query method is ideal for transforming existing columns, scenarios often arise where the calculation must be dynamic or based on complex expressions that are better handled by DAX. If we were forced to perform this inter-column maximum calculation using DAX, we would typically rely on creating a calculated column, as this operation requires row context--meaning the calculation must execute for every row independently.

Since the basic **MAX** function in DAX only operates on a single column (or two scalar values), finding the maximum of multiple columns requires a slightly indirect approach. One common DAX pattern for finding the maximum across multiple columns in a calculated column is using a nested sequence of **MAX** functions, which is effective for a small, known number of columns:

```
Max Score DAX = MAX ( , MAX ( , ) )
```

This formula ensures that the calculation is performed within the row context established by the calculated column, comparing the values of Game 1 against the maximum of Game 2 and Game 3, thus determining the overall maximum value across all three columns. For a much larger number of columns, a more advanced technique involving the **ROW** function combined with **MAXX** might be used, but this adds significant complexity and is typically less performant than the Power Query approach for static column comparisons.

Summary of Key Techniques

In summary, choosing the optimal method for finding the maximum value across multiple columns in Power BI depends entirely on whether the requirement is a static data transformation or a dynamic aggregation based on filter context. For simple, row-level comparisons of existing columns, the **Power Query Editor** provides the most efficient and user-friendly solution, leveraging the built-in **Maximum** statistic function under the Add Column tab.

Conversely, for more complex analytical tasks involving dynamic filtering or iterating over virtual tables, DAX functions like **MAXX** or nested **MAX** functions must be employed, typically within a calculated column or a measure, depending on the desired outcome. Mastering both techniques ensures flexibility and efficiency when developing robust Power BI data models and adhering to best practices for data transformation.

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