

How to Create a New Table in Power BI Using DAX from an Existing Table

Authored by
stats writer

January 30, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Create a New Table in Power BI Using DAX from an Existing Table*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128571>

Creating a new table within Power BI based on existing data requires leveraging the power of DAX (Data Analysis Expressions). This technique is essential when you need a static subset of data, a dimension table created from fact data, or a temporary structure to simplify complex calculations. The resulting structure is known as a **calculated table**, which exists purely within the data model and does not interact with the original data source connection, unlike standard imported tables.

The initial process of generating a **calculated table** is straightforward within the Power BI Desktop environment. Accessing the capability is done directly through the modeling interface. However, before executing the formula, it is paramount to understand the specific requirements for your new table, such as which columns are necessary and whether any row-level filtering is required. While simple column selection is common, more complex scenarios might necessitate using functions like FILTER in conjunction with table constructors to define precise row subsets.

The foundational steps for initiating this process involve navigating the modeling tools. This approach ensures that the new table formula is correctly registered within the data model. Remember that any table created via DAX is calculated upon data refresh and is stored entirely in memory, making efficiency a crucial consideration when designing the formula. Properly structuring your DAX expression will lead to a clean, well-performing data model optimized for reporting and visualization.

Initial Setup: Defining a Calculated Table using DAX

To successfully generate a new data structure in your model derived from existing tables, follow this precise sequence of actions within the Power BI interface:

Start by navigating to the "Table Tools" or "Data" tab within the Power BI Desktop ribbon, ensuring you are in the Data view or Model view.

Locate and select the "New Table" option, which initiates the formula entry mode, opening the formula bar ready for your DAX expression.

In the designated formula bar, meticulously type the DAX formula required to define the structure and content of the new table.

Utilize a table manipulation function, such as SELECTCOLUMNS or FILTER, to specify the primary source table from which the data will be drawn.

If necessary, incorporate additional filtering conditions or calculated columns within the formula to refine the resulting table structure to meet specific analytical requirements.

Define the name of the new table clearly by setting it as the variable assignment at the beginning of the formula (e.g., `NewTableName = DAX_Expression`).

Click the check mark icon or press Enter to validate the formula syntax and execute the table creation process. The newly generated table will immediately populate the "Fields" pane and be ready for use in reports and visuals.

By effectively employing DAX in this manner, you gain granular control over data manipulation, enabling the creation of focused, organized subsets essential for complex reporting and sophisticated data visualization within your model.

Understanding DAX Functions for Table Construction

When creating a calculated table, the choice of the primary DAX function dictates how the source data is transformed. Two of the most commonly confused functions for this purpose are `SELECTCOLUMNS` and `ADDCOLUMNS`. While both allow you to define the final column set, their execution and purpose are distinct. Understanding these nuances is key to writing efficient and accurate table definitions.

The `SELECTCOLUMNS` function is primarily used for projection--it iterates over an existing table and returns a new table containing only the specified columns. Crucially, it only includes the columns you explicitly define, meaning it discards all original columns unless they are explicitly referenced. This function is ideal for creating thin, focused dimension tables or simply extracting a critical subset of attributes from a much wider fact table, thereby reducing the model's memory footprint for that specific structure.

Conversely, functions like `SUMMARIZE` or combinations involving `ADDCOLUMNS` are often used when aggregation or derivation of new fields is required alongside the retention of existing columns. However, for the simple task outlined here--creating a new table that is an exact projection of certain columns from an existing table--`SELECTCOLUMNS` is the most precise and readable tool. It ensures clarity in your data model by defining exactly what is included and excluded from the new structure.

The Essential Syntax for Selective Column Extraction

The core objective in this scenario is to extract a specific set of attributes from a source table into a new, independent calculated table. This is elegantly achieved using the `SELECTCOLUMNS` function. The syntax requires three main components: the name of the new table, the `SELECTCOLUMNS` function itself, and the list of column pairs defining the output.

The structure begins with the assignment of the new table name, followed by the function call. Inside the function, the first argument is always the name of the existing source table. Subsequent arguments are defined in pairs: the desired name of the column in the new table (enclosed in double quotes) and the expression that defines its value, typically referencing the original column name in brackets. This syntax ensures that you have complete control over both the content and the metadata (column names) of the resulting table structure.

Observe the following syntax structure, which serves as the template for extracting data fields:

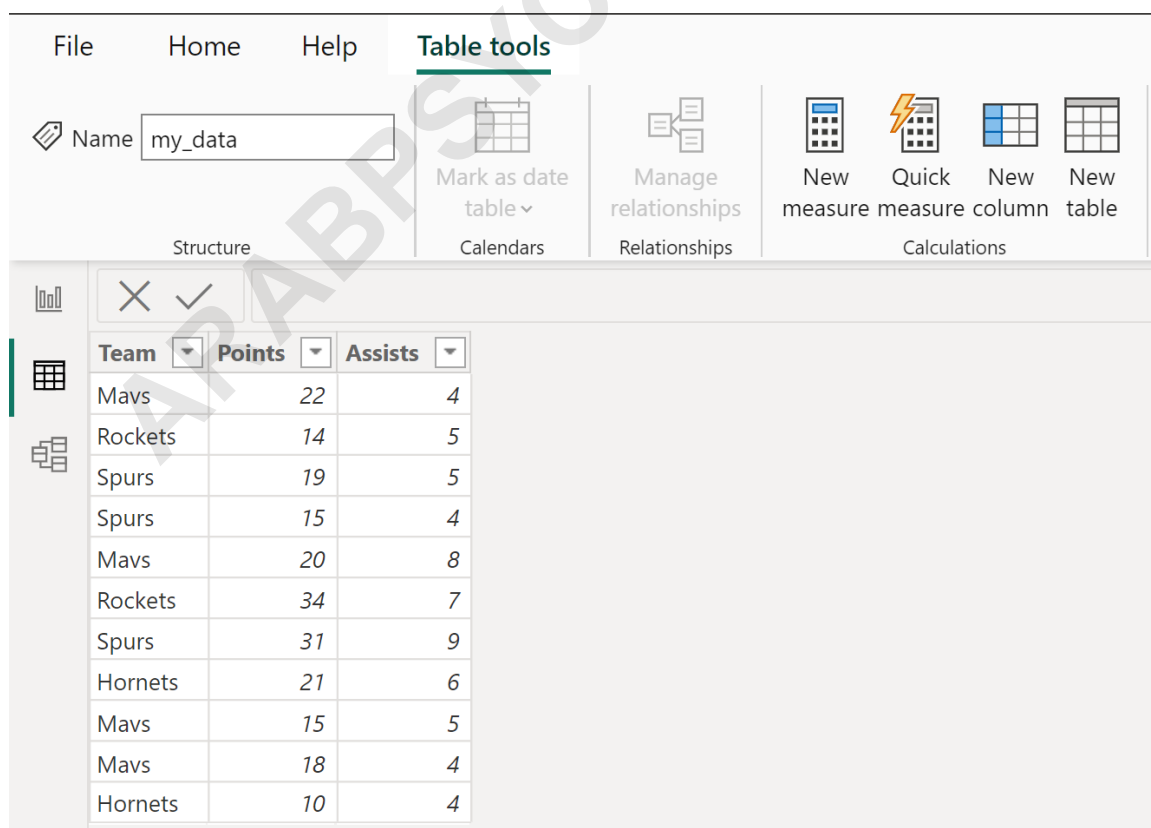
```
New_Data = SELECTCOLUMNS(  
My_Data,  
"Team", ,  
"Points", )
```

This particular command creates a new table named **New_Data**. This table will contain two columns: "Team" and "Points." Crucially, both of these new columns are populated by corresponding fields from the existing source table named **My_Data**. This methodology provides a clean and concise way to generate focused data subsets for targeted reporting tasks or for use in dimensional modeling within the Power BI environment.

Practical Example: Creating a Focused Data Subset

To illustrate the practical application of this DAX technique, consider a common scenario where a large transaction table contains numerous fields, but only a few are relevant for a specific dashboard concerning team performance metrics. We need to isolate just the team identifiers and their accumulated points.

Suppose our existing source data table, named **my_data**, contains extensive information, as depicted in the following snapshot:



The screenshot displays the Power BI interface. The 'Table tools' ribbon is active, showing options like 'Name' (set to 'my_data'), 'Mark as date table', 'Manage relationships', 'New measure', 'Quick measure', 'New column', and 'New table'. Below the ribbon, a data table is visible with the following data:

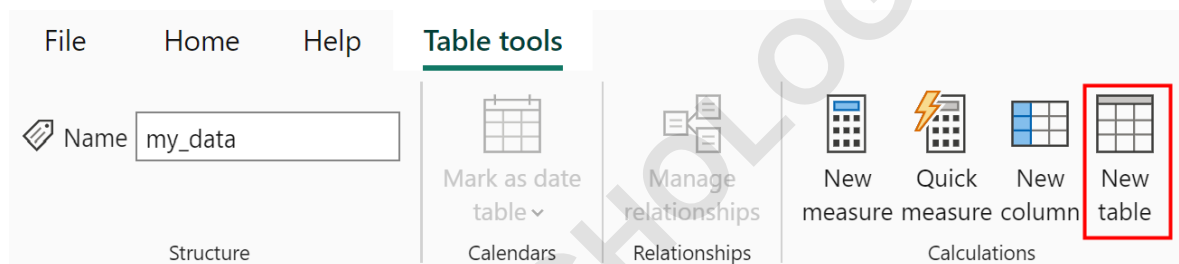
Team	Points	Assists
Mavs	22	4
Rockets	14	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	34	7
Spurs	31	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

Our goal is straightforward: to construct a new, derived table that strictly holds the **Team** identifier and the associated **Points** value, filtering out auxiliary columns like 'Region' and 'Coach'. This calculated structure provides a streamlined view that is easier to manage and faster to process in visuals dedicated solely to scoring metrics. This ability to create data subsets dynamically is one of the key advantages of mastering DAX for data modeling.

Step-by-Step Implementation Guide

The implementation of the SELECTCOLUMNS formula requires precise navigation within the Power BI Desktop interface to ensure the expression is correctly registered as a calculated table.

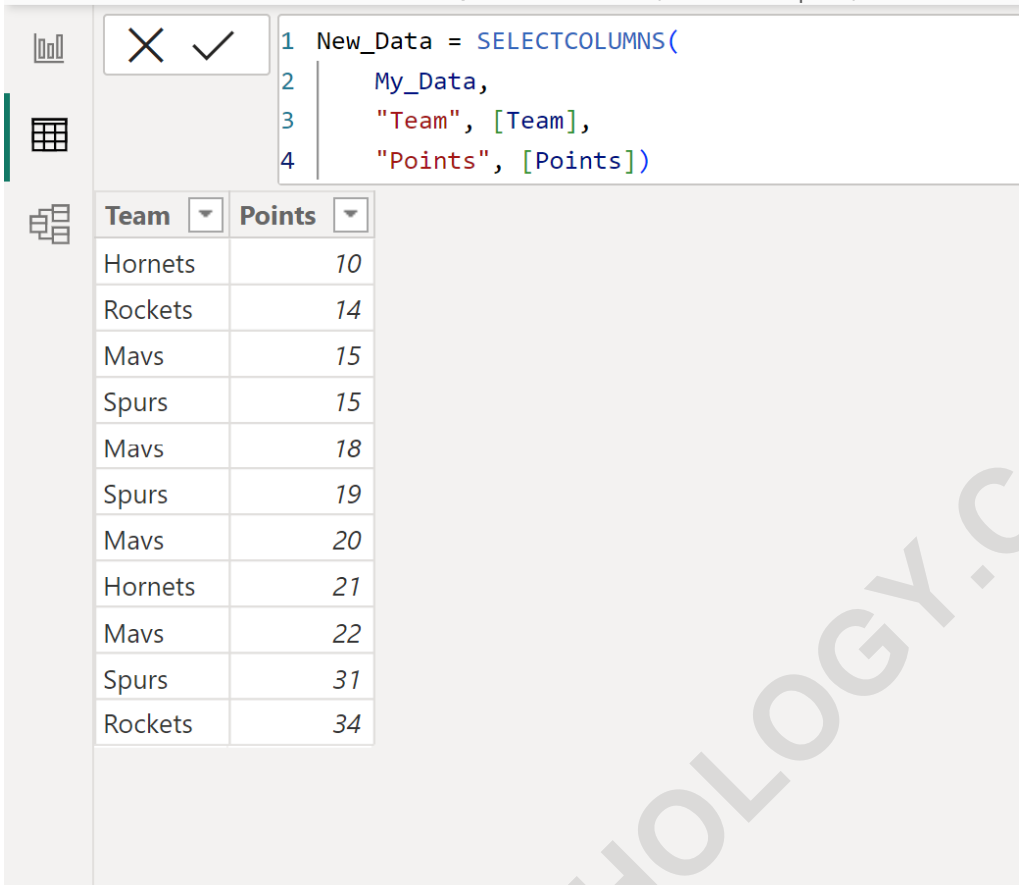
First, navigate to the **Data View** within Power BI Desktop. Once there, locate the **Table Tools** tab on the ribbon. This tab contains the specific commands required for creating new structures within your data model. Click on the **New table** icon, as shown below, to open the formula editor:



Next, input the complete DAX expression into the formula bar. For our example, where we are extracting the Team and Points columns from the **My_Data** source table, the formula is entered exactly as follows:

```
New_Data = SELECTCOLUMNS(  
My_Data,  
"Team", ,  
"Points", )
```

Upon pressing **Enter** or validating the formula, Power BI executes the expression. A new table, designated **New_Data**, is immediately generated and displayed. Observe that this new table faithfully reproduces only the selected columns, **Team** and **Points**, effectively serving as a tailored projection of the original data:



```

1 New_Data = SELECTCOLUMNS(
2     My_Data,
3     "Team", [Team],
4     "Points", [Points])

```

Team	Points
Hornets	10
Rockets	14
Mavs	15
Spurs	15
Mavs	18
Spurs	19
Mavs	20
Hornets	21
Mavs	22
Spurs	31
Rockets	34

Advanced Technique: Customizing Output Column Names

A significant advantage of using the `SELECTCOLUMNS` function is the ability to rename columns simultaneously during the creation of the new table. This feature enhances readability and adheres to specific naming conventions required for downstream reporting, ensuring that the new calculated table is instantly optimized for visualization.

The renaming process occurs within the column pair argument structure of the `SELECTCOLUMNS` function. Instead of repeating the source column name, you specify a new, descriptive name in the first element of the pair (the text string) while maintaining the reference to the original source column in the second element (the expression).

For instance, if the original column names 'Team' and 'Points' are too generic for our report, we can redefine them as 'Team_Name' and 'Points_Scored' respectively. This is achieved using the following modified DAX syntax:

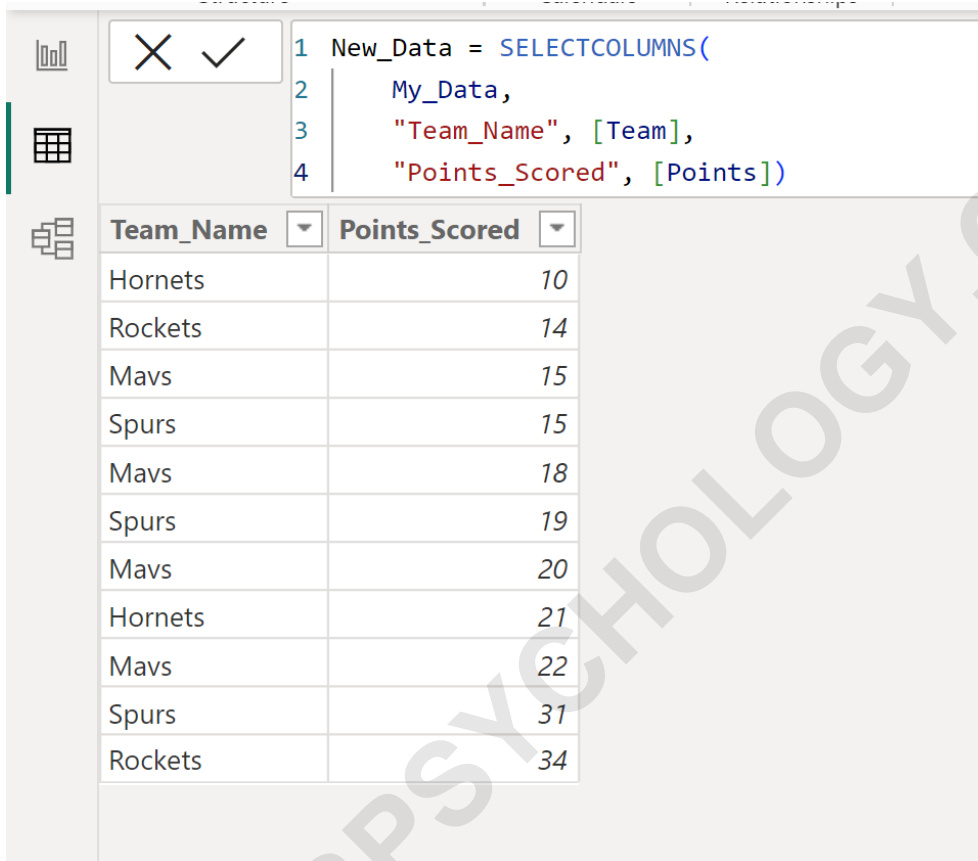
```

New_Data = SELECTCOLUMNS(
My_Data,
"Team_Name", ,

```

"Points_Scored",)

Executing this revised formula generates the same data set but with improved metadata. The resultant **New_Data** table now features the new, more descriptive column headers, which is a crucial step in preparing data for final user consumption:



The screenshot shows the DAX editor in Power BI. The formula bar contains the following DAX code:

```
1 New_Data = SELECTCOLUMNS(  
2     My_Data,  
3     "Team_Name", [Team],  
4     "Points_Scored", [Points])
```

Below the formula bar, a data grid is displayed with two columns: "Team_Name" and "Points_Scored". The data rows are as follows:

Team_Name	Points_Scored
Hornets	10
Rockets	14
Mavs	15
Spurs	15
Mavs	18
Spurs	19
Mavs	20
Hornets	21
Mavs	22
Spurs	31
Rockets	34

Benefits and Strategic Use Cases of Calculated Tables

While the Power Query Editor (M language) is often preferred for initial data transformation and structure creation, using DAX to create a calculated table offers unique advantages, particularly when the table definition relies on measures, complex internal model relationships, or functions that are native only to DAX.

One primary use case is the creation of decoupled dimension tables. If your source data model requires a specific dimension (e.g., a list of unique customers, teams, or products) derived from a large fact table, generating it via calculated table ensures that the dimension is consistently defined within the semantic model, irrespective of external source changes, provided the source data is refreshed. This facilitates easier management of relationships and hierarchies.

Furthermore, calculated tables are instrumental in scenarios requiring complex filtering or iteration. Although the [SELECTCOLUMNS](#) function focuses on projection, combining it with other functions like [FILTER](#) allows for the simultaneous extraction of columns and rows that meet specific criteria, all within a single, powerful formula. This combination creates highly specialized tables necessary for advanced analytic models.

Conclusion and Resources for Further DAX Mastery

Mastering the creation of calculated tables using functions like [SELECTCOLUMNS](#) is a fundamental skill for advanced data modeling in [Power BI](#). This technique provides the necessary flexibility to refine, restructure, and optimize your data model directly within the modeling environment, leading to more efficient processing and clearer reporting outcomes.

By leveraging [DAX](#), data professionals can move beyond simple imports and transformations to truly architect a robust and responsive data solution tailored precisely to their organizational needs. We strongly recommend consulting the official documentation to explore the full potential and nuances of the functions discussed.

Power BI: Create Table from Another Table by Using DAX

Note: You can find the complete documentation for the **SELECTCOLUMNS** function in DAX, which details all arguments and specific usage scenarios.

Related Tutorials for Power BI Data Manipulation

The following tutorials explain how to perform other common tasks in [Power BI](#), complementing your understanding of calculated tables and data modeling best practices:

How to implement row-level security using [DAX](#).

Techniques for using the [FILTER](#) function to refine visual context.

Strategies for creating effective date dimension tables in [Power BI](#).