How to Perform a Left Join in Power BI: A Step-by-Step Guide

Authored by

mohammed looti

January 9, 2026

1. Understanding the Left Join Operation in Power BI

The process of combining disparate datasets is fundamental to effective data analysis and reporting. In Power BI, this crucial task is handled primarily through the merging of queries. A Left Join, specifically, is a powerful technique for integrating information from two sources--known as the left (first) table and the right (second) table--based on a shared identifier or key column.

The core principle of the Left Join, often referred to as a **Left Outer Join**, is to retain all records from the primary, or left, table regardless of whether a corresponding match exists in the secondary, or right, table. If a match is found using the common field, the relevant data from the right table is appended to the row in the left table. If no match is found for a specific record in the left table, the new columns derived from the right table are populated with blank or **null values**.

This approach is particularly valuable when performing comprehensive data audits or longitudinal studies, such as combining a master list of **customer information** with a transaction log. By utilizing a Left Join, you ensure that every customer is included in the resulting dataset, even those who have not yet generated any sales activity, thus preventing the omission of crucial demographic or profile data required for complete segmentation. This capability ensures a holistic view of the data, extending beyond just the intersecting records.

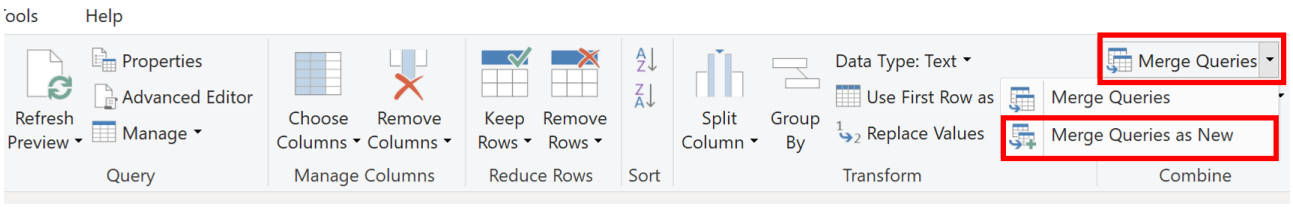
2. The Preferred Method: Merge Queries in Power Query Editor

While there are several complex methods available for data manipulation in Power BI, the most accessible and efficient way to execute a **Left Join** between two tables is by utilizing the built-in **Merge Queries** function. This feature is housed within the Power Query Editor, which acts as the Extract, Transform, Load (ETL) staging area for your data before it is loaded into the Power BI data model.

The Power Query Editor provides a highly intuitive graphical interface that abstracts the underlying complexities of database querying languages. Instead of writing intricate SQL or DAX code, users simply select the tables, define the common key columns, and specify the desired **Join Kind**. For a Left Join, the specific option to select is the **Left Outer** join type.

The easiest way to perform a **left join** between two tables in Power BI is to use the **Merge Queries** function in the **Power Query Editor**.

This function allows for flexible joining operations and is the standard practice for combining data sources prior to visualization. The following image illustrates where the Merge Queries function is located within the interface:

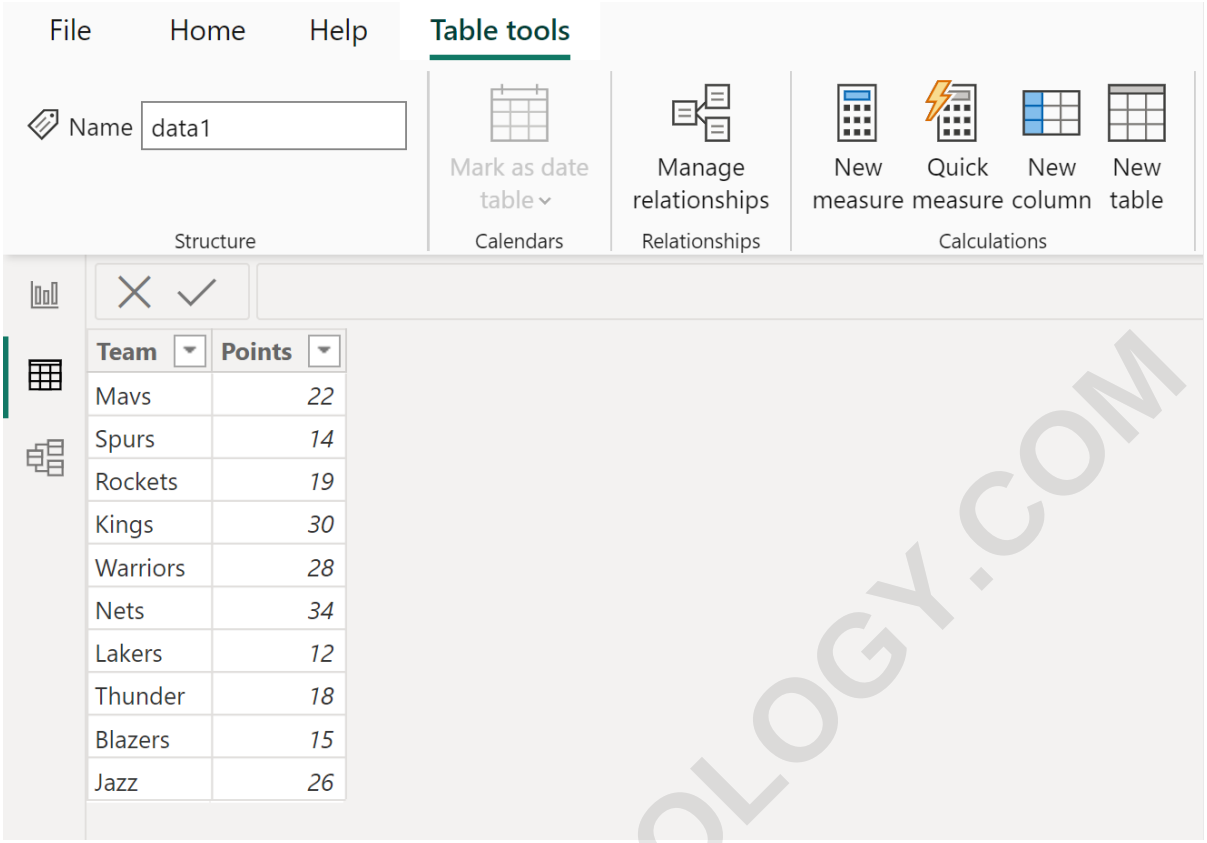


The subsequent steps will walk through a practical example demonstrating the end-to-end process of setting up and finalizing a Left Join operation.

3. Defining the Sample Datasets for the Join Example

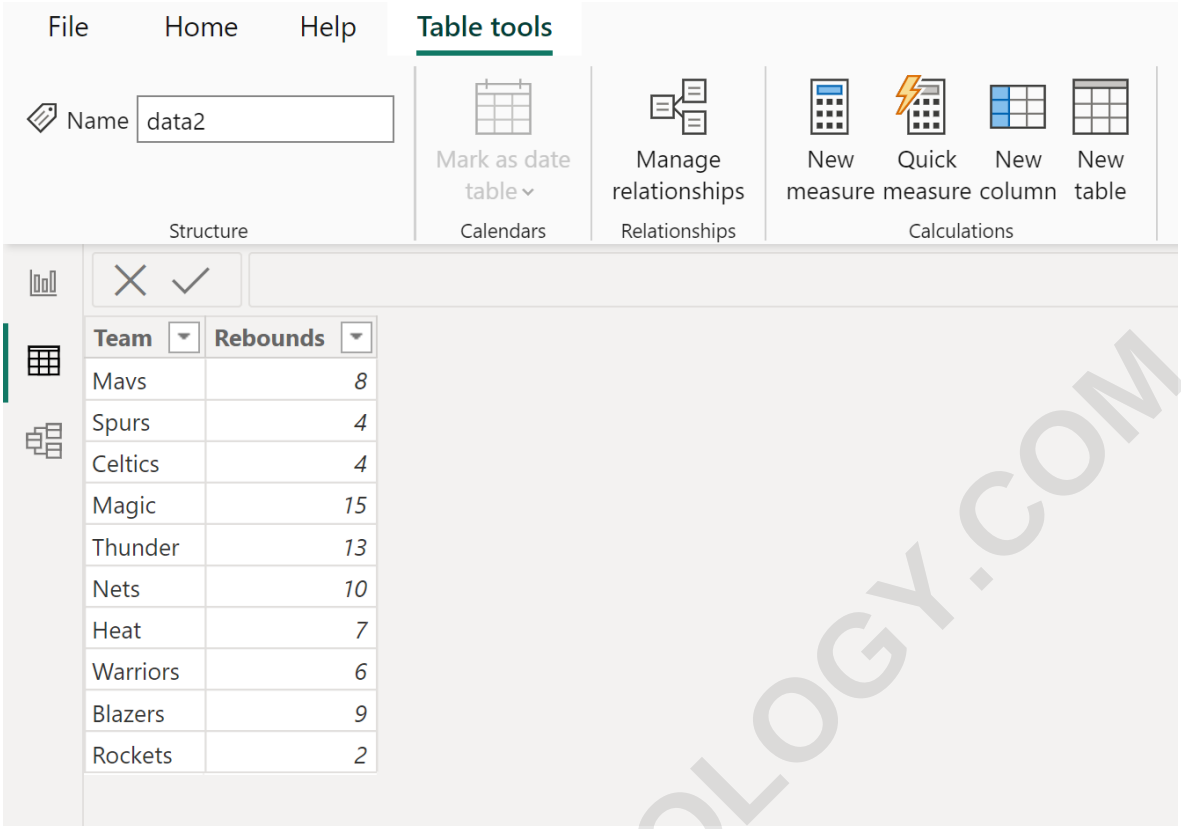
To demonstrate the functionality of the Left Join, we will use two hypothetical tables containing basketball statistics. These tables mimic real-world scenarios where related data is segmented across different sources or reports, requiring consolidation for effective data analysis.

Suppose we have the following initial table in Power BI named **data1**. This table serves as our **primary**, or left, table, and contains core information about basketball players, specifically their team name and the points they have scored. The goal is to ensure that all players listed in **data1** remain in the final merged dataset.



Team	Points
Mavs	22
Spurs	14
Rockets	19
Kings	30
Warriors	28
Nets	34
Lakers	12
Thunder	18
Blazers	15
Jazz	26

Next, consider a second table named **data2**. This table acts as our **secondary**, or right, table, and provides supplementary information--in this case, the number of rebounds associated with various teams. Notice that the players and teams listed in **data2** may not perfectly align with those in **data1**, which is precisely why a Left Join is necessary to handle non-matching data gracefully.



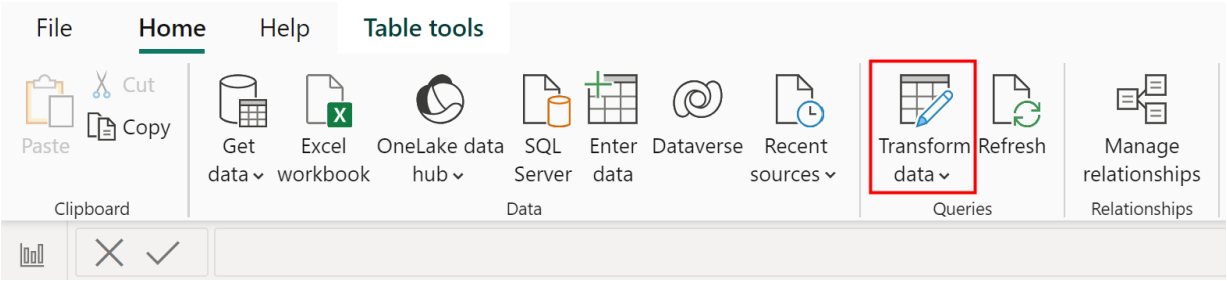
Team	Rebounds
Mavs	8
Spurs	4
Celtics	4
Magic	15
Thunder	13
Nets	10
Heat	7
Warriors	6
Blazers	9
Rockets	2

Our objective is clear: we intend to perform a **left join** operation where we retain every single row from the **data1** table, associating them with the corresponding **Rebounds** values extracted from the **data2** table, matching based on the common **Team** column.

4. Initiating the Power Query Editor for Transformation

Before any merging can occur, we must access the environment where data transformations are managed--the Power Query Editor. This editor is where you define the steps necessary to clean, shape, and combine your data sources.

To begin the process, navigate to the main Power BI Desktop interface. Look for the **Home** tab located along the top ribbon interface. Within this ribbon, locate and click the **Transform data** icon. This action will launch the separate dedicated window for the **Power Query Editor**.



Once the **Power Query Editor** window loads, you will see a list of your existing queries (tables) on the left-hand side. Ensure that both **data1** and **data2** are present and loaded correctly before proceeding to the merge operation.

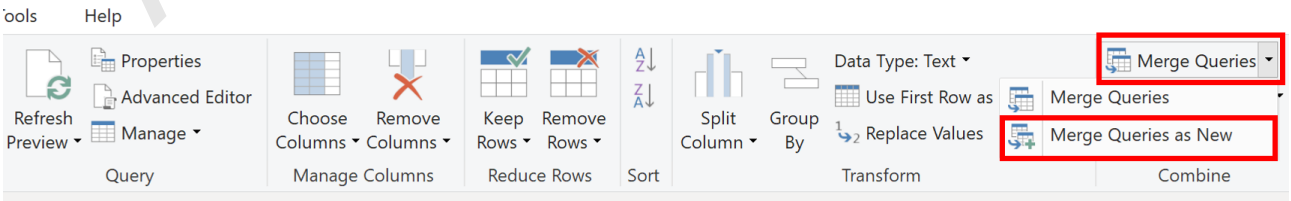
5. Executing the Merge Queries Operation

With the Power Query Editor open, the next step is to initiate the merging process. There are two primary ways to merge queries: merging into the existing table (which modifies the current query) or merging as a new query (which preserves the original tables). For best practice, particularly in complex data analysis scenarios, merging as a new query is recommended to maintain the integrity and traceability of the source data.

Follow these steps to execute the merge:

On the **Home** tab within the Power Query Editor, locate the **Combine** group.
Click the **Merge Queries** icon.
Then click **Merge Queries as New** from the dropdown menu. This action ensures that a new query, typically named "Merge1," is created, containing the result of the Left Join.

This selection prompts a new configuration window where the parameters of the join--the tables and the join type--must be explicitly defined.



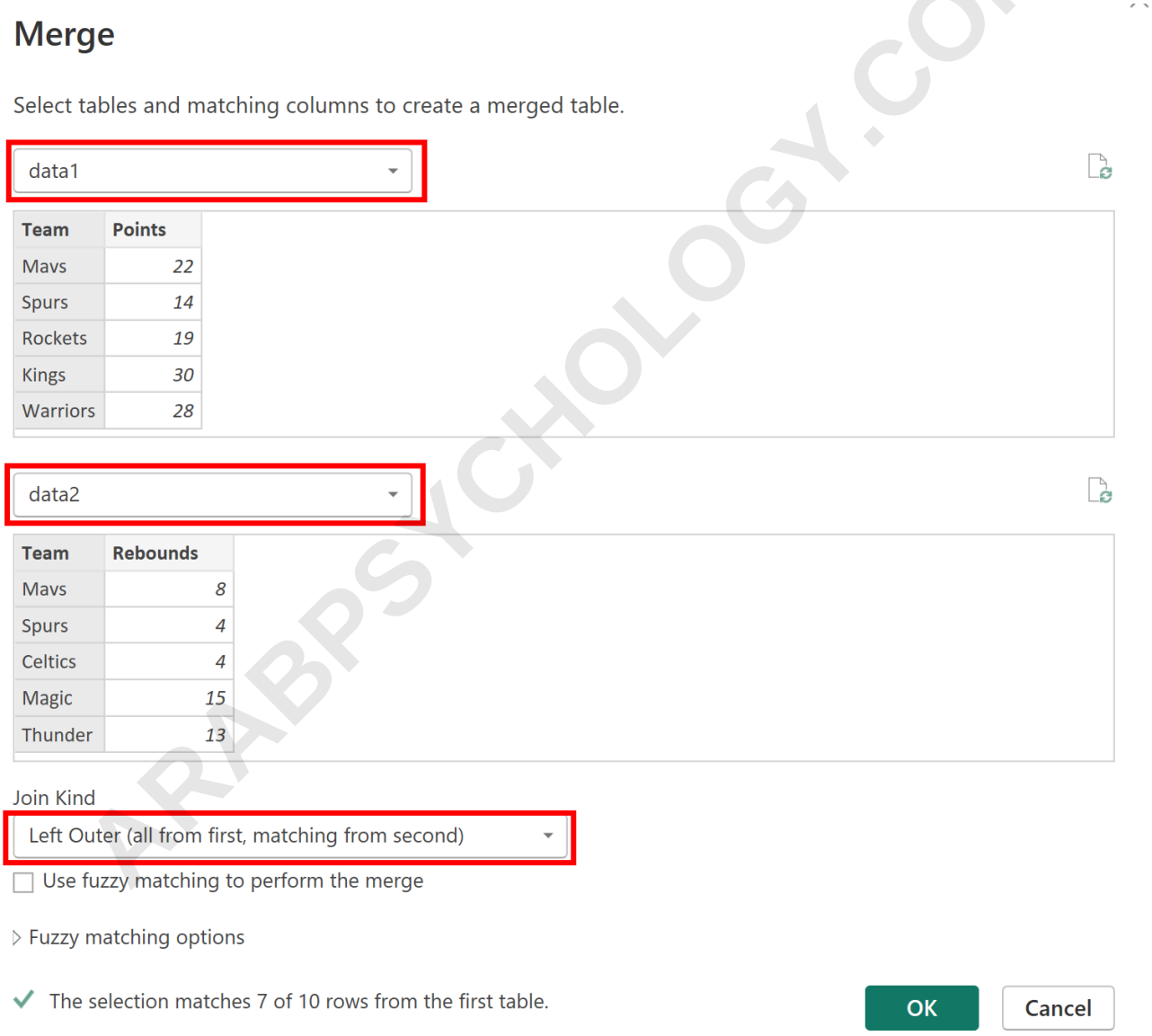
6. Configuring the Left Outer Join Parameters

The Merge dialog box requires three critical pieces of information to successfully execute the join:

the primary table, the secondary table, and the type of join to perform. Since we are performing a **Left Join**, we must define the tables carefully to ensure the intended outcome.

In the new window that appears, choose **data1** as the first table (the left side), choose **data2** as the second table (the right side), and choose **Left Outer** as the **Join Kind**.

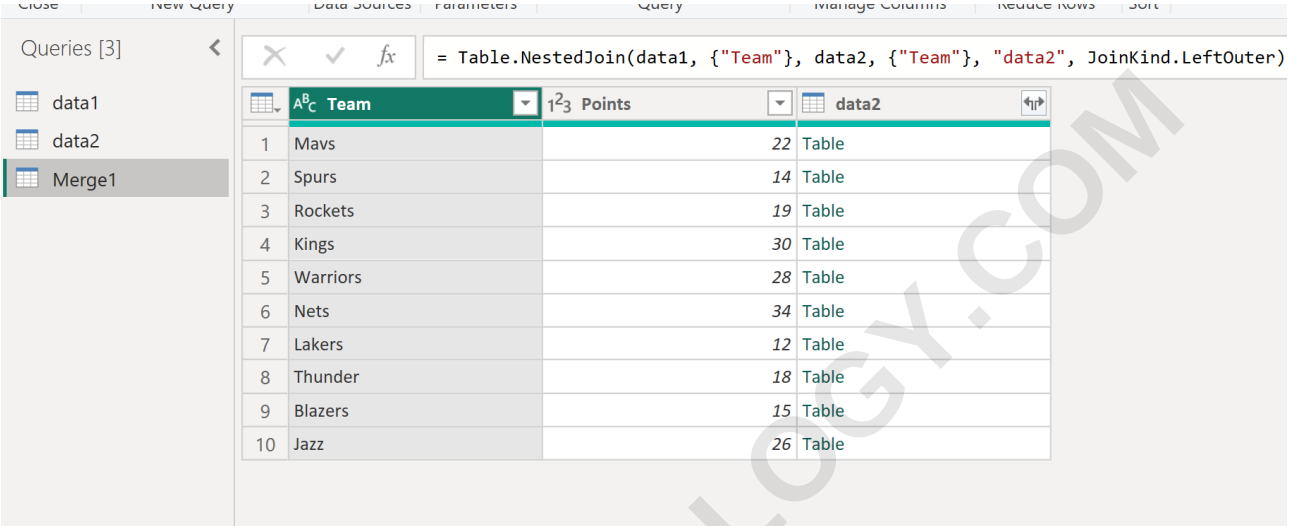
Then click the header for the **Team** column in each table so that Power BI knows to use those columns as the common key for the join. This connection is vital for matching rows between the two data sources.



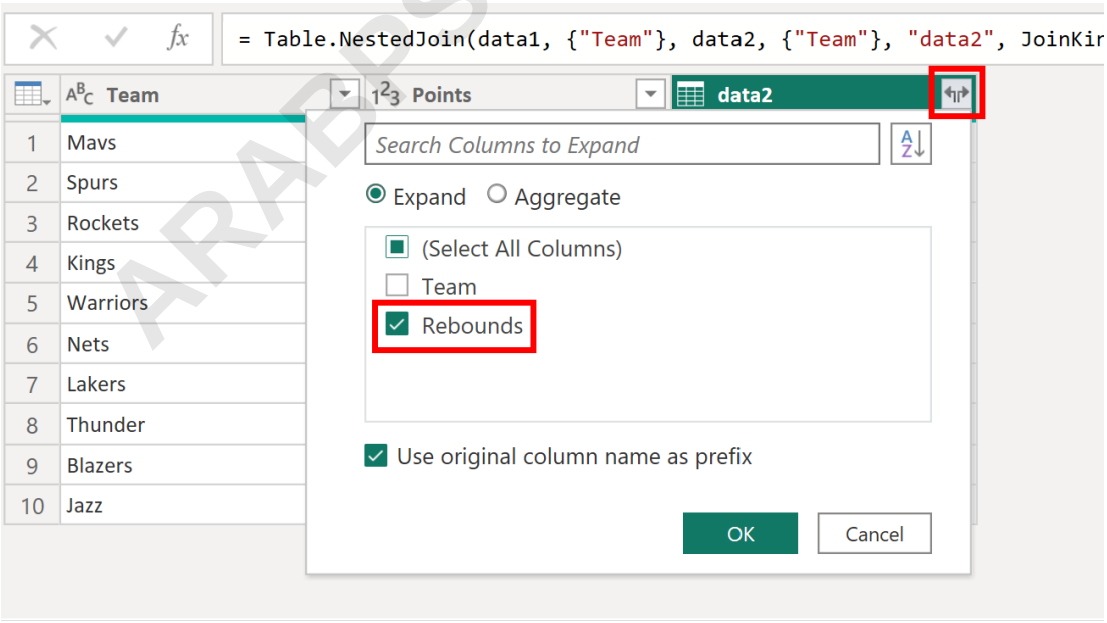
Once you click **OK**, the left join will be performed, resulting in a new query with a structured column named **data2** appended to the original **data1** fields.

7. Expanding the Merged Column and Finalizing the Result

After the initial merge execution, the new column, **data2**, contains nested table objects. We must now specify which columns from **data2** should be extracted and made visible in the final merged table. Since we only need the rebounds data, we will expand only that specific field.

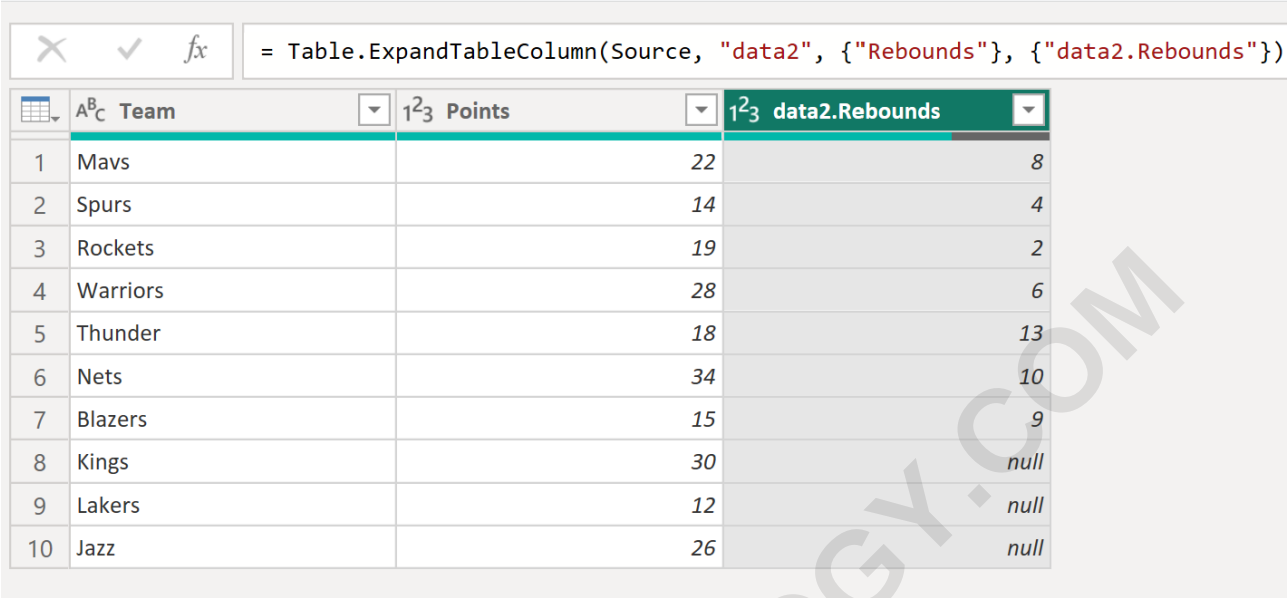


Next, click the left and right arrows on the header of the **data2** column. Then check the box next to **Rebounds** to indicate that this column should be the only one included from **data2** in the final merged table. We are selectively pulling data to keep the resulting table clean and focused.



Once you click **OK**, the **Rebounds** column will be successfully shown from the **data2** table, now

appended row by row to the **data1** information.



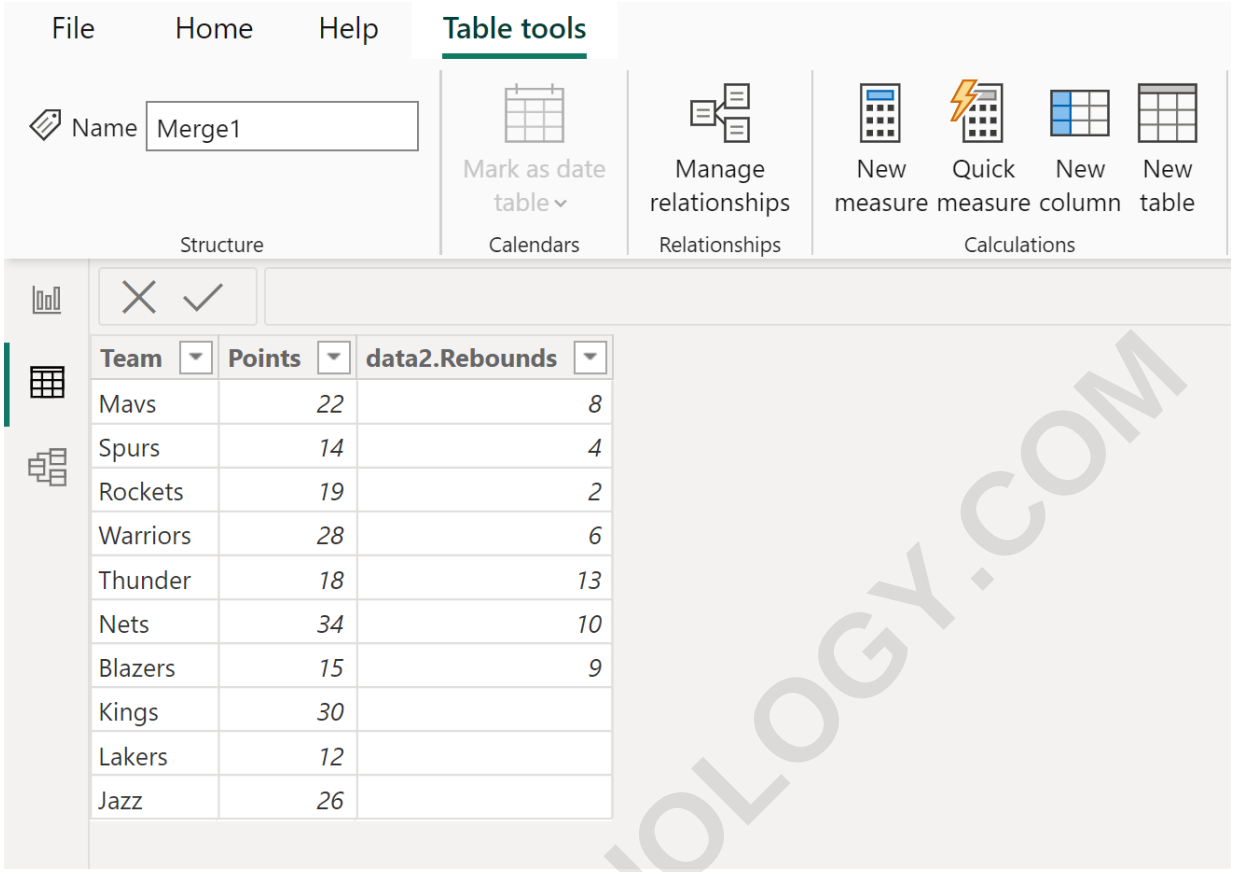
	Team	Points	data2.Rebounds
1	Mavs	22	8
2	Spurs	14	4
3	Rockets	19	2
4	Warriors	28	6
5	Thunder	18	13
6	Nets	34	10
7	Blazers	15	9
8	Kings	30	null
9	Lakers	12	null
10	Jazz	26	null

8. Loading and Analyzing the Final Dataset

The final step is to commit the transformation changes and load the resultant table into the Power BI data model. Once you exit out of the **Power Query Editor** (by clicking Close & Apply), a message box will appear that asks if you'd like to apply your changes.

Click **Yes**.

You will then be able to see the new table named **Merge1** in the Table view within Power BI Desktop, ready for further analysis and report building.



Team	Points	data2.Rebounds
Mavs	22	8
Spurs	14	4
Rockets	19	2
Warriors	28	6
Thunder	18	13
Nets	34	10
Blazers	15	9
Kings	30	
Lakers	12	
Jazz	26	

Notice that **all of the rows from the left table (data1) are kept** in the merged table while only the rows with matching values in the **Team** column from the right table (**data2**) are included. The record for 'Hornets' from **data1**, which had no match in **data2**, is preserved, but its **data2.Rebounds** cell contains a blank value. This validation confirms that the **Left Join** functioned exactly as intended.

Any row in the **Team** column from **data1** that did not have a matching value in the **Team** column from **data2** simply have a blank value in the **data2.Rebounds** column.

Note: If you'd like, you can right click on the header named **data2.Rebounds** and rename the column to just **Rebounds** for cleaner report design.

9. Conclusion and Related Data Integration Tasks

The ability to perform a reliable Left Join using the **Merge Queries** function is a cornerstone of effective data preparation in Power BI. By utilizing the **Left Outer** join type in the **Power Query Editor**, users can ensure that their primary dataset remains intact while enriching it with relevant supplemental data. This process guarantees maximum coverage for subsequent reporting and data analysis tasks.

The following tutorials explain how to perform other common tasks in Power BI, offering different methods for data integration based on specific business requirements:

Performing an Inner Join to find only intersecting records.
Implementing a Full Outer Join to keep all records from both tables.
Utilizing Append Queries to stack data tables vertically.