How to Use the “Not Equal” Operator in Power BI for Data Filtering

Authored by

stats writer

January 26, 2026

The "Not Equal" operator in Power BI is a foundational logical operator that empowers users to perform essential comparative analyses. This operator is crucial for determining if two specific values are distinct, returning a Boolean result (TRUE if they are not equal, FALSE if they are). Its primary utility lies in data transformation and filtering, allowing analysts to quickly segment datasets based on exclusionary criteria.

In the context of Power BI, the ability to isolate data points that do not match a given criteria is vital for efficient data analysis. While the "Equals" operator identifies matching records, the "Not Equal" counterpart handles the complementary task: identifying outliers, non-conforming entries, or simply all records outside a predefined group. This enhances the precision of reports and visualizations, ensuring focus is placed on the specific data subsets required for decision-making.

Implementing the "Not Equal" operator is straightforward, typically involving its integration into DAX expressions or utilizing it directly within visual filters. Mastering this operator allows users to build robust data models. It is a necessary component when constructing complex conditional logic, especially when dealing with categorical data where exclusion criteria define the required dataset.

The Role of DAX in Power BI Filtering

All sophisticated data manipulation and calculation within Power BI rely heavily on DAX, or Data Analysis Expressions. DAX is the formula language used throughout the Power BI ecosystem, providing the engine for custom columns, measures, and calculated tables. Understanding how comparative operators function within the DAX syntax is the first step toward leveraging their power effectively.

When applying filtering logic, DAX interprets the "Not Equal" condition as a crucial element in determining row context. Whether creating a dynamic classification column using the IF function or defining a wholly new subset of data using table manipulation functions, the **<>** operator ensures that only rows meeting the inequality condition are processed. This deterministic approach is fundamental to creating accurate data summaries and visualizations.

Beyond simple filtering, the **<>** operator plays a vital role in data governance and anomaly detection. For instance, an analyst might use "Not Equal" to exclude known errors, test cases, or specific product lines from aggregated metrics, ensuring that core business reporting remains unbiased. This level of granular control over data inclusion and exclusion highlights the necessity of correctly implementing logical operators within the DAX framework.

Syntax and Implementation: Using the <> Operator

The universal representation for the "Not Equal" operator within DAX is the combination of the less than and greater than symbols: **<>**. Unlike some programming languages that might use `!=`, DAX

maintains this specific syntax for consistency across all comparison operations. Recognizing and correctly applying this symbol is mandatory for writing clean and valid DAX formulas, whether defining calculated columns or creating filtered tables.

The application of **<>** typically falls into two major operational categories, both of which are central to data modeling in Power BI. The first involves classifying individual rows based on a comparison, resulting in a new calculated column that annotates the data. The second involves manipulating the entire data model or a specific table, applying the inequality as a filter context to derive a smaller, targeted table.

These two primary methods are illustrated below using a hypothetical dataset. The critical takeaway is that while the goal--excluding a specific value--is the same, the method of implementation (column versus table creation) depends entirely on the required output and the impact on the overall data model structure. Regardless of the method chosen, the **<>** operator serves as the definitive logical gatekeeper.

Power BI: Use "Not Equal" Operator

Implementing the Not Equal Operator in DAX

As confirmed, the syntax for the "not equal" operator in DAX is **<>**. Below are the two standard approaches for utilizing this operator in your data modeling efforts:

Method 1: Use <> to Create New Column

This method leverages the inequality operator within a conditional statement to classify each row based on whether a specific criterion is met or not met. This is ideal for adding descriptive labels to your data model.

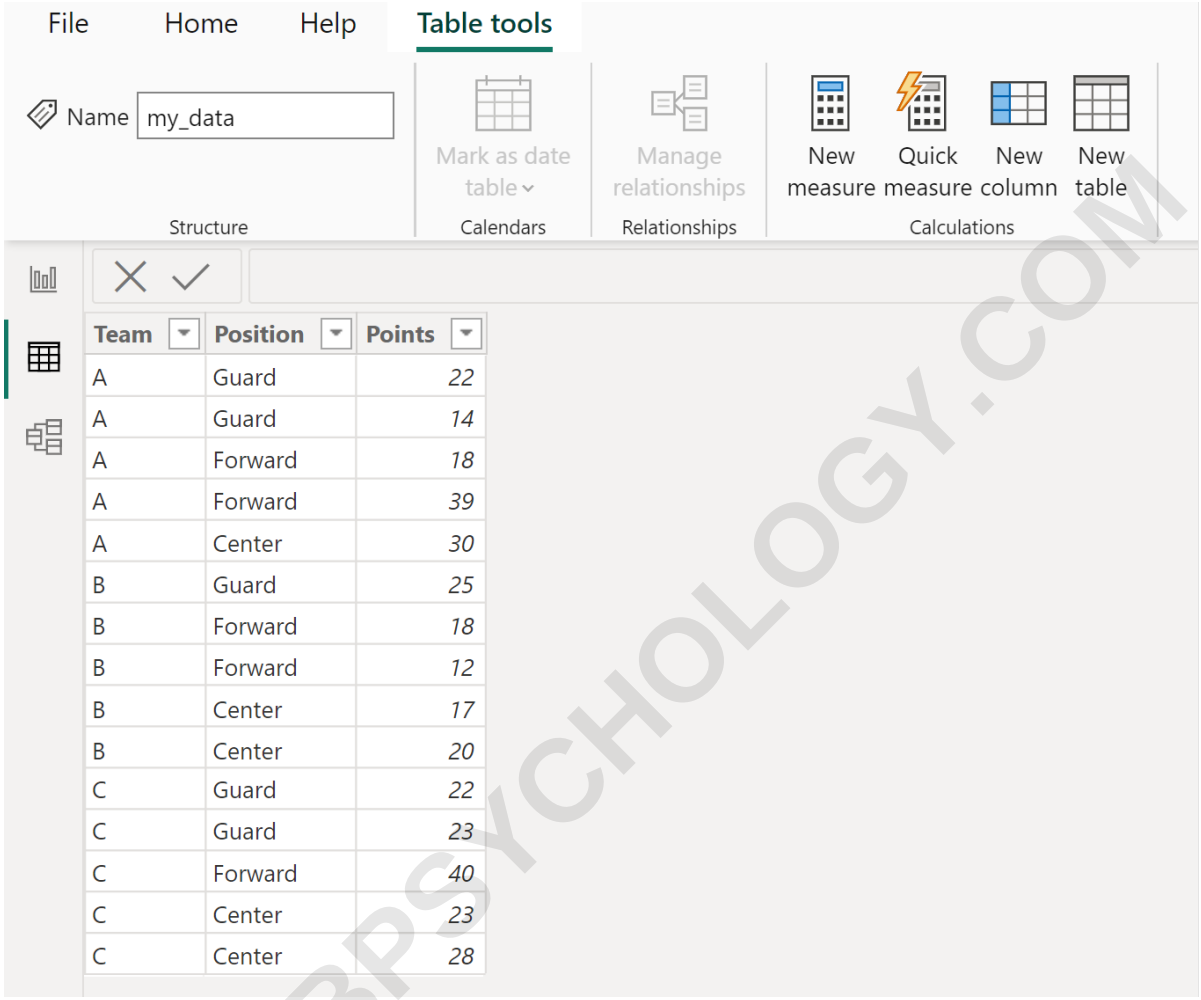
Team Classification =
IF('my_data' <> "B", "Not on Team B", "On Team B")

Method 2: Use <> to Create New Table

This method employs the inequality operator as a filter argument within a table function, resulting in a new table that contains only the rows that satisfy the "not equal" condition. This is essential for creating subsets of data for focused analysis.

filtered_data =
CALCULATETABLE('my_data', 'my_data' <> "B")

The following detailed examples demonstrate the practical implementation of both methods using a sample table named **my_data** in Power BI, which contains information concerning various basketball players:



Team	Position	Points
A	Guard	22
A	Guard	14
A	Forward	18
A	Forward	39
A	Center	30
B	Guard	25
B	Forward	18
B	Forward	12
B	Center	17
B	Center	20
C	Guard	22
C	Guard	23
C	Forward	40
C	Center	23
C	Center	28

Practical Application 1: Defining a New Column Based on Inequality

One of the most frequent uses of the "Not Equal" operator is to generate a new, descriptive column that categorizes data based on a specific exclusionary criterion. In this scenario, we aim to distinguish all players who are **not** on Team B, providing immediate clarity within the dataset itself. This is achieved by employing the IF function in conjunction with the **<>** operator.

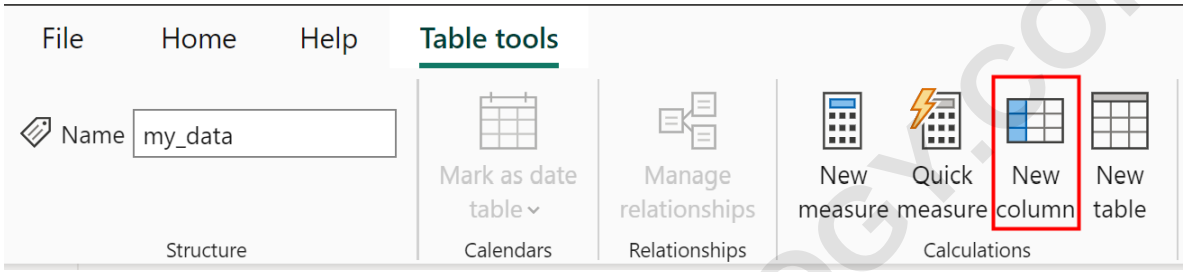
The objective is straightforward: we want the new column to return "Not on Team B" if the value in the **Team** column does not equal "B," and conversely, return "On Team B" if the value equals "B." This binary classification is invaluable for quick filtering and visual grouping within reports. The calculated column, unlike a measure, stores its result row by row within the table structure.

To begin this process in Power BI Desktop, navigate to the **Table tools** ribbon. This tab provides

all the necessary functionality for modifying the structure and content of your active data model. Selecting the "New Column" option initiates the DAX formula editor, where the logical comparison will be defined.

Follow these steps to implement the calculated column:

First, ensure you have selected the appropriate table (**my_data** in this case) in the Data view.
Click the **Table tools** tab in the ribbon.
Then, click the **New column** icon to open the formula bar.



Step-by-Step: Implementing the New Column Formula

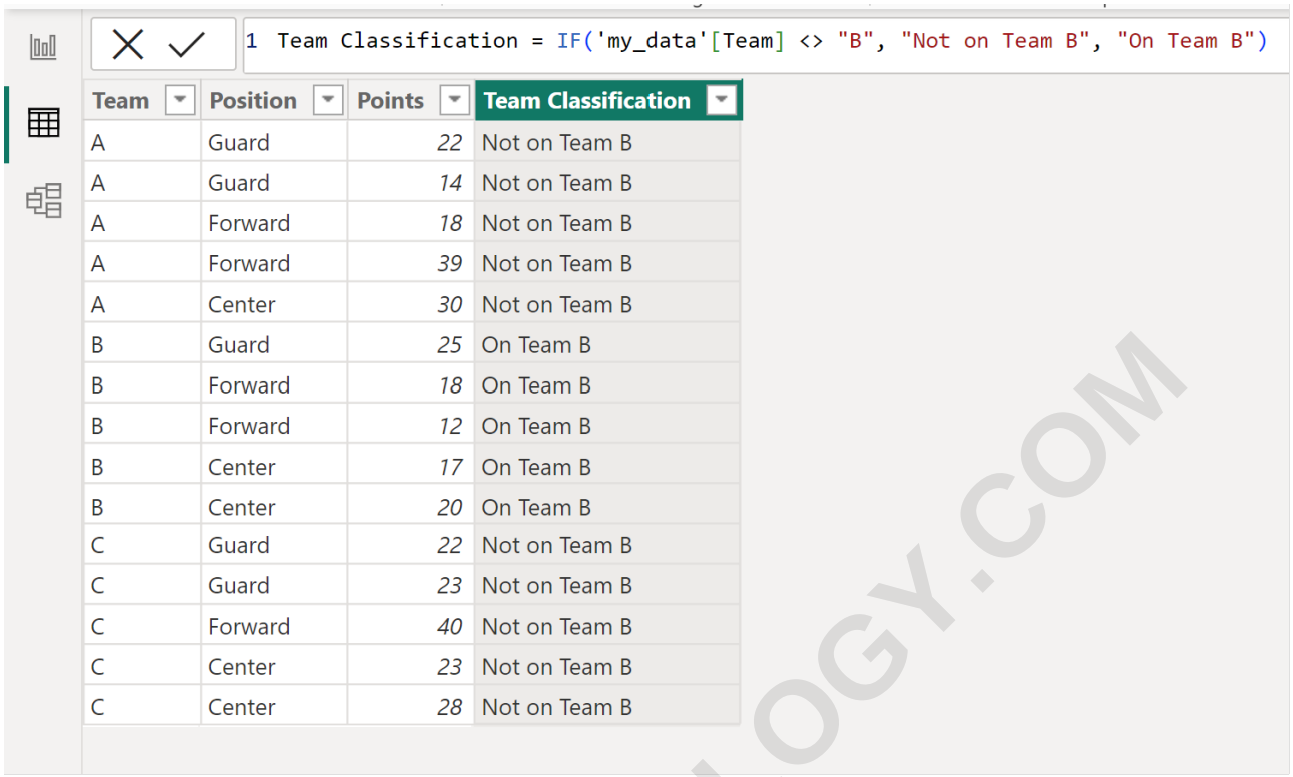
Once the formula bar is active, the critical step is inputting the precise DAX syntax that incorporates the **<>** operator. The formula utilizes the IF function, which requires three arguments: the condition (the logical test), the value if TRUE, and the value if FALSE.

Type the following complete formula into the formula bar. Notice how the logical test checks for inequality first:

Team Classification =
IF('my_data' <> "B", "Not on Team B", "On Team B")

Upon committing the formula (by pressing Enter or clicking the checkmark), Power BI evaluates the expression for every row in the **my_data** table. If the value in the `` column is anything other than "B," the condition is TRUE, and "Not on Team B" is assigned. This efficient use of the **<>** operator allows for precise data labeling without requiring multiple nested conditional checks.

The resultant data table immediately reflects the addition of the new column named **Team Classification**, providing the specified classification strings based on the inequality logic applied:



Team	Position	Points	Team Classification
A	Guard	22	Not on Team B
A	Guard	14	Not on Team B
A	Forward	18	Not on Team B
A	Forward	39	Not on Team B
A	Center	30	Not on Team B
B	Guard	25	On Team B
B	Forward	18	On Team B
B	Forward	12	On Team B
B	Center	17	On Team B
B	Center	20	On Team B
C	Guard	22	Not on Team B
C	Guard	23	Not on Team B
C	Forward	40	Not on Team B
C	Center	23	Not on Team B
C	Center	28	Not on Team B

Practical Application 2: Filtering Data to Create a New Table

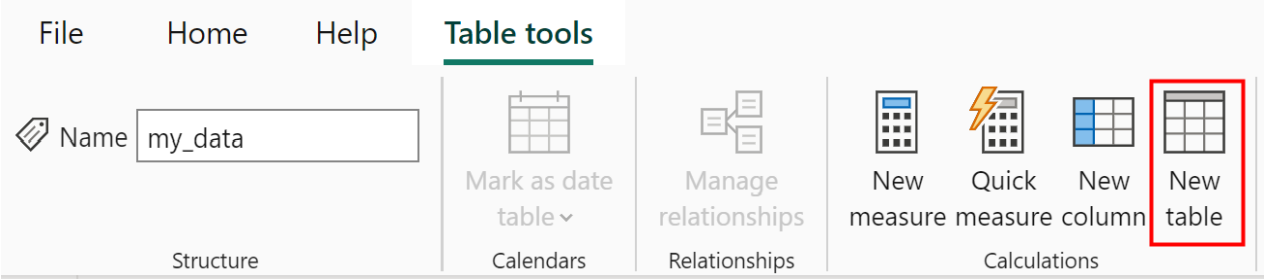
The second major application of the "Not Equal" operator is creating entirely new data tables that serve as filtered subsets of an existing table. This approach is highly beneficial when specific reporting views or complex calculations require a dedicated table containing only records that satisfy an exclusionary rule. For example, if we need to perform calculations only on players who are **not** part of Team B, creating a filtered table simplifies subsequent steps.

In this example, the goal is to generate a new table containing only the rows from **my_data** where the value in the **Team** column is not equal to "B." To achieve this, we employ the powerful CALCULATETABLE function, which applies a filter context to an existing table expression.

Initiating the creation of a new table follows a process similar to creating a new column but targets a different output type. Navigate to the **Table tools** tab within Power BI Desktop and select the appropriate icon for table creation.

Follow these steps to generate the filtered table:

Ensure you are in the Data view or Model view.
Click the **Table tools** tab in the ribbon.
Click the **New table** icon to activate the formula bar specifically for table definitions.



Step-by-Step: Implementing the New Table Formula

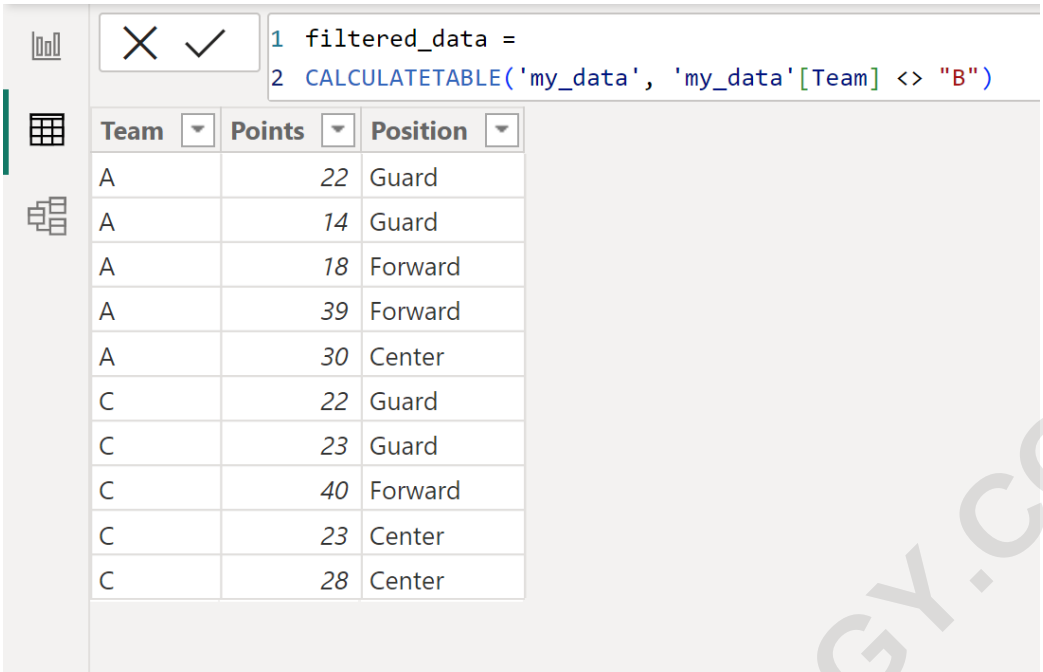
Defining a new table requires a function that returns a table object, which is precisely what CALCULATETABLE does. This function takes the base table as its first argument and one or more filter expressions as subsequent arguments. Our filter expression will utilize the **<>** operator to exclude records where the `` column equals "B."

Input the following formula into the table formula bar:

filtered_data =
CALCULATETABLE('my_data', 'my_data' <> "B")

When this formula is executed, Power BI constructs a new table named **filtered_data**. This table is derived from **my_data**, but only rows where the team designation is not "B" are included. This demonstrates the efficiency of using the "Not Equal" operator within a powerful table function like CALCULATETABLE.

The resulting table, **filtered_data**, contains only the player records where the Team column value satisfies the inequality condition, confirming the accurate application of the filter context:



```
1 filtered_data =
2 CALCULATETABLE('my_data', 'my_data'[Team] <> "B")
```

Team	Points	Position
A	22	Guard
A	14	Guard
A	18	Forward
A	39	Forward
A	30	Center
C	22	Guard
C	23	Guard
C	40	Forward
C	23	Center
C	28	Center

Advanced Uses and Considerations for <>

While the examples focused on filtering categorical text data, the **<>** operator can be applied equally effectively to numerical values, dates, and other data types, allowing for flexible filtering logic. For instance, one could filter transactions where the `` **<>** 0, or exclude records where the `` **<>** a specific date. Understanding data types is crucial, as comparing a string field to a numeric value using **<>** will often lead to unexpected results or errors.

It is important to remember that DAX provides a comprehensive set of operators. While **<>** handles inequality, combinations with logical operators like **AND** (`&&`) and **OR** (`||`) allow for highly complex conditional checks. For example, you might look for rows where ` <> "B"` **AND** ` > 10`. Proper use of parentheses is recommended to ensure the correct order of evaluation, especially when mixing multiple logical operators.

Note: For a full mastery of comparative operations, users should consult the official documentation detailing all available operators you can use in DAX.

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