

How to Filter Data by Excluding Values in Power BI Using “NOT IN

Authored by
stats writer

January 25, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Filter Data by Excluding Values in Power BI Using “NOT IN.* PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=127608>

The ability to precisely filter datasets is absolutely fundamental to effective business intelligence, especially within the sophisticated environment of Power BI. While standard SQL environments often rely on the explicit Logical Operator "NOT IN," the Data Analysis Expressions (DAX) language achieves this essential exclusionary filtering through a combination of the NOT function and the IN operator. This powerful construct allows analysts to define a specific subset of values--such as product categories, geographic regions, or team names--and then ensure that any data records matching those defined values are comprehensively excluded from the resulting calculation or output table.

This capability is indispensable for scenarios requiring the isolation of specific data segments, such as focusing on all products except the top seller, analyzing regional performance excluding headquarters, or removing known outliers that could skew overall metrics. By effectively implementing the "NOT IN" logic in DAX, users gain enhanced control over their Filter Context, resulting in cleaner, more focused datasets that directly support streamlined visualization and actionable insights. Understanding how to structure this combined operator within functions like CALCULATETABLE is critical for advanced data manipulation and modeling within Power BI.

Implementing the "NOT IN" Operator in Power BI (Detailed Guide and Examples)

Understanding Exclusionary Filtering in DAX

Data Analysis Expressions (DAX) operates on contexts--specifically, the Filter Context and the Row Context. When we wish to exclude data, we are fundamentally manipulating the Filter Context to prevent certain rows from being evaluated or returned. Unlike programming languages that might offer a built-in NOT IN keyword, DAX requires the use of the NOT function applied around a set of values defined using the IN operator. This structure explicitly tells the DAX engine to evaluate membership within a specified set and then negate that result, effectively filtering for non-membership.

The core concept revolves around the evaluation of a Boolean condition. The IN operator checks if a column's value matches any item in the provided list (a table expression or a literal list enclosed in curly brackets). When the NOT function wraps this IN operation, the result is inverted: rows where the value **is** present in the list return FALSE, and rows where the value **is not** present return TRUE. This TRUE/FALSE evaluation then dictates which rows are retained by the surrounding filtering function, such as CALCULATETABLE or CALCULATE, ensuring that only the desired records--those excluded from the list--are included in the final output.

It is paramount to remember that the list of values used with the IN operator must be defined using

curly brackets { }. This syntax convention signifies a specific set of literal values or a single-column table used for comparison. For complex data modeling tasks in Power BI, leveraging this exclusionary logic is far more efficient than writing multiple, convoluted AND statements with inequality operators (e.g., `<> "A" AND <> "C"`), especially when the exclusion list is extensive or dynamically generated.

The Core DAX Syntax for "NOT IN"

The standard methodology for executing an exclusionary filter involves wrapping the membership check within the NOT function, often coupled with a table manipulation function like CALCULATETABLE, which is specifically designed to return a filtered set of rows based on the filter expression provided. This technique is especially useful when the goal is not merely to filter a measure, but to create a physical or virtual table reflecting the desired exclusion.

The following syntax demonstrates the standard structure used in Power BI to implement the "NOT IN" operation, yielding a new filtered table based on exclusion criteria applied to a specific column:

```
filtered_data =  
CALCULATETABLE('my_data', NOT('my_data' IN {"A", "C"}))
```

This particular DAX expression creates a new table, conventionally named **filtered_data**. The function operates on the source table, **'my_data'**, applying the specified filter expression. The filter expression, `NOT('my_data' IN {"A", "C"})`, mandates that only rows from the source table where the value in the column does not match either "A" or "C" will be included in the resultant table. This is a highly efficient way to handle set exclusion in the DAX environment.

It is important to acknowledge the crucial role played by the curly brackets { } when specifying the values that constitute the exclusion list. These brackets define the single-column table or set of literal values against which the IN operator performs its membership test. In this specific scenario, the literal text values "A" and "C" are treated as the set of teams that must be actively filtered out, providing clear and concise instruction to the CALCULATETABLE function.

CALCULATETABLE: The Engine Behind Table Filtering

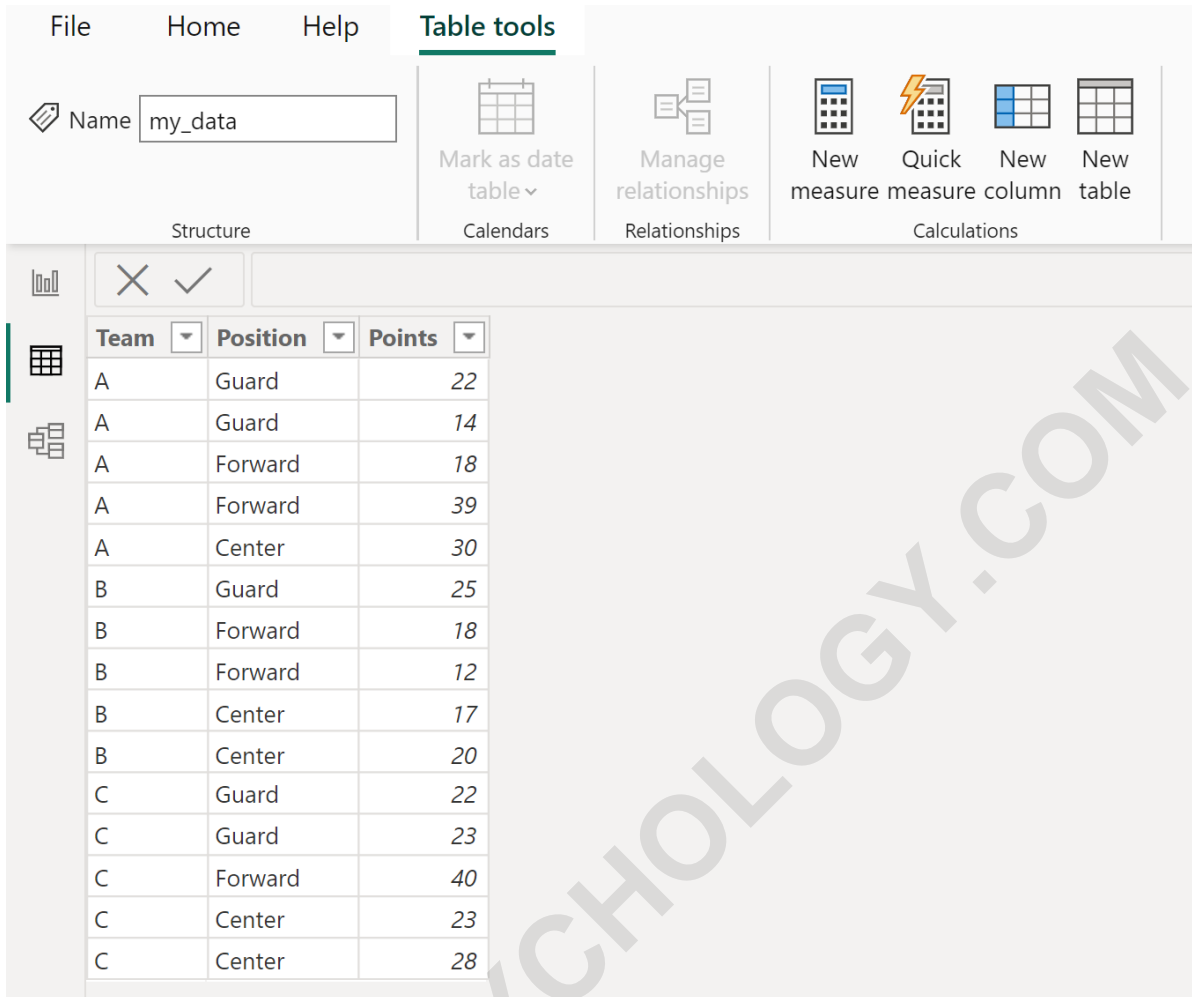
The CALCULATETABLE function is central to advanced table manipulation in Power BI. Unlike CALCULATE, which modifies the Filter Context for measure evaluation, CALCULATETABLE returns a new table derived from an existing table, applying the provided filters. This is precisely why it is the appropriate function when the analytical requirement is to generate a dedicated subset of data that excludes certain values. The function takes the base table as its first argument and subsequent arguments define the filter expressions that shape the output.

When `CALCULATETABLE` encounters the `NOT (IN)` filter, it systematically iterates through the rows of the base table. For each row, the filter expression is evaluated. If the current row's value for the column being tested (e.g., 'my_data') is found within the exclusion list ({"A", "C"}), the `IN` operation returns `TRUE`, which the outer `NOT` function inverts to `FALSE`. Consequently, that row is discarded. Conversely, if the value is not in the exclusion list, the `IN` operation returns `FALSE`, the `NOT` function inverts it to `TRUE`, and the row is included in the new **filtered_data** table. This detailed process ensures precise control over the resulting data structure.

Setting Up the Scenario in Power BI

To illustrate this functionality, let us consider a practical dataset within Power BI. Suppose we have a table named **my_data** that captures vital statistics regarding basketball players, including their associated team and position. This hypothetical dataset is critical for performance metrics, but for a specific analysis, we are instructed to exclude all data related to teams "A" and "C".

The initial dataset, **my_data**, is structured as follows, containing information that spans various teams and positions, presenting the starting point for our exclusionary filtering task:



The screenshot displays the Power BI Desktop interface. At the top, the 'Table tools' ribbon is active, showing options like 'Name' (set to 'my_data'), 'Mark as date table', 'Manage relationships', and 'Calculations' (with sub-options: 'New measure', 'Quick measure', 'New column', 'New table'). Below the ribbon, a data table is visible with the following columns: Team, Position, and Points. The table contains 15 rows of data.

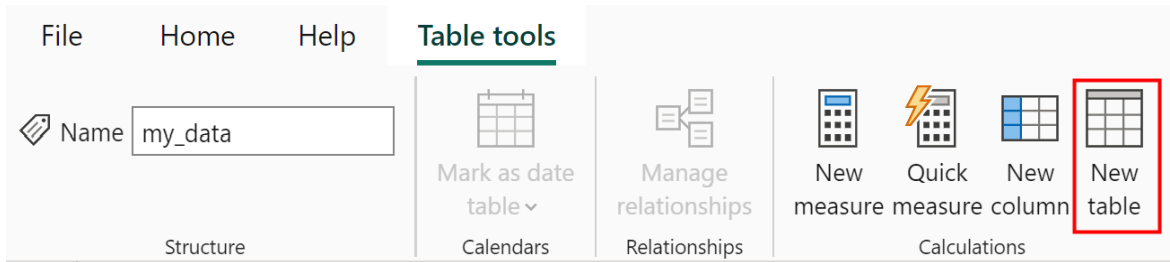
Team	Position	Points
A	Guard	22
A	Guard	14
A	Forward	18
A	Forward	39
A	Center	30
B	Guard	25
B	Forward	18
B	Forward	12
B	Center	17
B	Center	20
C	Guard	22
C	Guard	23
C	Forward	40
C	Center	23
C	Center	28

Our objective is clearly defined: we must produce a derivative table that contains only players who are definitively **not** members of team A or team C. This requires creating a new table object in the data model rather than just calculating a summarized measure, necessitating the use of the CALCULATE function with the appropriate DAX syntax we have previously discussed.

Step-by-Step Table Creation Using Exclusion

To initiate the process of creating the filtered table in Power BI Desktop, navigation to the modeling tools is necessary. Within the interface, the analyst must access the **Table tools** tab, which contains the options for defining new, calculated tables within the existing data model. Subsequently, the **New table** icon must be clicked to open the formula bar and begin writing the requisite DAX definition.

This action prepares the environment for inputting the DAX code that defines the filtering operation. The specific location of the **New table** button, typically found within the Data or Modeling tabs, is essential for launching the calculation editor:



Once the formula bar is active, the analyst must input the exclusionary DAX formula precisely as defined earlier. The formula explicitly names the new table, **filtered_data**, and utilizes the CALCULATETABLE function to apply the set exclusion logic to the base table, **'my_data'**. The list {"A", "C"} ensures the precise identification of teams to be removed from the resulting dataset.

filtered_data =

CALCULATETABLE('my_data', NOT('my_data' IN {"A", "C"}))

Upon committing this formula, Power BI executes the calculation, resulting in the creation of the new table named **filtered_data**. Examination of this new table confirms that it successfully incorporates only the players whose team affiliation is neither A nor C, fulfilling the primary filtering requirement. This resultant table is now ready for use in subsequent reports, visuals, or further analytical operations, having been successfully cleansed of the specified exclusion criteria.

The screenshot shows the Power BI interface with the DAX formula bar active. The formula is:


```
1 filtered_data =
2 CALCULATETABLE('my_data', NOT('my_data'[Team] IN {"A", "C"}))
```

 Below the formula bar, a table view is displayed with the following data:

Team	Points	Position
B	25	Guard
B	18	Forward
B	12	Forward
B	17	Center
B	20	Center

Extending Filtering with Multiple Exclusion Criteria

While filtering based on a single column is common, real-world data analysis frequently demands

the application of multiple, independent exclusion criteria simultaneously. For instance, an analyst might need to exclude specific teams **AND** exclude players occupying certain positions, creating a compounded exclusionary Filter Context. DAX handles this easily by combining multiple `NOT (IN)` expressions using the Logical Operator AND, represented by `&&` (double ampersand).

The objective in this advanced scenario is to generate a table where records must satisfy two distinct negative conditions. Specifically, we want to exclude records where:

The value in the **Team** column is either A or C.

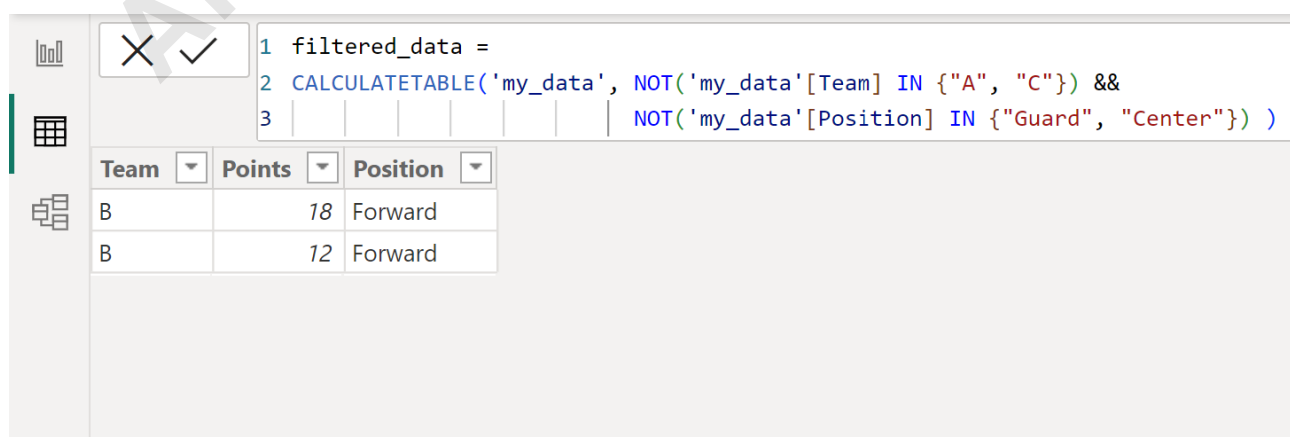
The value in the **Position** column is either Guard or Center.

Both conditions must be independently filtered by exclusion, and the results of both filters must be true for a row to be included in the final table. This necessitates linking the two distinct `NOT (IN)` filter expressions using the `&&` operator within the CALCULATETABLE function. This approach ensures that the row must pass both the team exclusion filter and the position exclusion filter simultaneously.

The following DAX syntax achieves this complex, compound exclusion, demonstrating the robust capability of DAX to handle multi-faceted filtering requirements:

```
filtered_data =  
CALCULATETABLE('my_data', NOT('my_data' IN {"A", "C"}) &&  
NOT('my_data' IN {"Guard", "Center"}))
```

Executing this formula within Power BI creates a new table, again named **filtered_data**, which precisely adheres to the dual constraints. The resulting table only contains players who belong neither to team A nor C, and simultaneously do not hold the position of Guard or Center. This sophisticated filtering minimizes the dataset to only those players meeting the stringent dual exclusionary criteria, providing a narrow and highly specific focus for analysis.



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

```
1 filtered_data =  
2 CALCULATETABLE('my_data', NOT('my_data'[Team] IN {"A", "C"}) &&  
3 | NOT('my_data'[Position] IN {"Guard", "Center"})) )
```

Below the formula bar, a table view is displayed with the following data:

Team	Points	Position
B	18	Forward
B	12	Forward

Performance Considerations and Best Practices

While the `NOT (IN)` syntax is highly readable and effective, implementing complex exclusionary filters requires consideration of performance, especially when dealing with very large datasets in Power BI. DAX is optimized for set operations, making the `IN` operator generally efficient. However, when the list of excluded values is extremely long, the engine must spend more time evaluating membership for every row.

A recommended best practice is to always define the exclusion list using literal values (as shown with curly brackets) if the list is static and relatively small. If the exclusion list is dynamic or derived from another table, it is essential to ensure that the referenced table is as small and optimized as possible, ideally containing only unique values. Furthermore, while we used `CALCULATETABLE` here to demonstrate table generation, if the ultimate goal is only to filter a measure, using the same filter logic within the `CALCULATE` function often results in superior performance, as it avoids creating an entirely new physical table in the model. Always strive to minimize the impact on the Filter Context manipulation when working with complex expressions.

Note: A comprehensive understanding of all available Logical Operator and filtering functions in DAX is essential for optimizing query performance. Consulting the official Microsoft DAX documentation is highly recommended for mastering various filtering techniques and understanding their implications on the data model.

Conclusion

The ability to implement exclusionary logic efficiently using the `NOT (IN)` combination is a fundamental skill for any advanced Power BI analyst. By wrapping the `IN` operator, which defines the set of values to be matched, within the `NOT` function, we achieve the functionality equivalent to "NOT IN" in other query languages. This method ensures that the resulting data structure--whether a newly calculated table or a filtered measure result--is precisely tailored to exclude the specified data points, driving highly targeted and effective data visualization and reporting.

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