How to Create IF Statements with Multiple Conditions in Power BI

Authored by

stats writer

January 14, 2026

An IF statement with multiple conditions in Power BI allows analysts to define sophisticated logical expressions that evaluate various data fields concurrently. These expressions must ultimately resolve to a single boolean value--either **TRUE** or **FALSE**. By combining multiple tests using logical operators like OR and AND, you gain the power to create highly customized classifications and dynamic data transformations within your data model. This capability is fundamental for generating nuanced reports and accurate business intelligence dashboards in Power BI.

The flexibility of combining conditions is crucial when business rules dictate that a specific outcome depends on meeting criteria across several metrics simultaneously, or when satisfying just one of several criteria is sufficient for categorization. Mastering the syntax of these combined logical functions within DAX (Data Analysis Expressions) is essential for moving beyond simple, single-condition evaluations and building robust data models.

In the sections that follow, we will explore the precise DAX syntax required to structure these multi-conditional IF statements. We will focus specifically on how the OR function and the AND function are nested within the primary conditional test to achieve different analytical outcomes, providing clear, practical examples for implementation in your calculated columns.

Understanding Conditional Logic Structure in DAX

The foundation of all conditional logic in DAX is the IF function, which follows the standardized format: `IF(logical_test, value_if_true, value_if_false)`. The complexity is managed entirely within the `logical_test` parameter. This parameter must resolve to a single **TRUE** or **FALSE** boolean result, even if it incorporates dozens of individual checks.

To combine multiple checks into one cohesive `logical_test`, we rely on boolean operator functions such as OR and AND. These functions are designed to accept two or more conditional arguments (e.g., `> 100`) and evaluate them based on specific rules of inclusion or exclusion. Unlike some scripting languages, DAX requires that these operators be used in a functional syntax, meaning they are explicitly wrapped around their respective conditions.

When implementing these conditional statements as a calculated column, it is important to remember that Power BI evaluates the expression row by row. This is known as **row context**. For example, if you define a **Rating** column using combined conditions, the formula checks the specific Points and Assists values of the current row before assigning "Good" or "Bad." Understanding this context is vital for debugging and ensuring that the logical results align perfectly with the data at the granular level.

You can use the following syntax in DAX to write an IF statement with multiple conditions in Power BI:

Method 1: Writing an IF Statement with OR Condition

The OR function is essential when defining classification rules where meeting any single criterion is sufficient to trigger the TRUE result. This method provides flexibility in grouping data, allowing records that satisfy different pathways to be included in the same category. For example, a customer might be classified as **High Priority** if they have high sales volume OR if they have a critical service contract.

In this structure, we nest the OR function inside the logical test parameter of the primary IF statement. The primary goal is to return a positive result if condition 1 is TRUE, condition 2 is TRUE, or if both are TRUE. The only scenario that returns the `value_if_false` result is when all included conditions are FALSE. The following code illustrates how to generate a **Rating** column based on whether a player exceeds a threshold in either **Points** or **Assists**.

```
Rating =
IF(
OR(
'my_data' > 20,
'my_data' > 4
),
"Good",
"Bad"
)
```

This specific calculation creates a new column named **Rating**. It returns the value "Good" if the value in the **Points** column is greater than 20 **or** if the value in the **Assists** column is greater than 4. If neither of these conditions are met--meaning both Points are 20 or less AND Assists are 4 or less--then the function defaults to returning the value "Bad" instead. This structure is highly useful for inclusive categorization where multiple paths lead to a successful outcome.

Method 2: Writing an IF Statement with AND Condition

The AND function is utilized when classification demands that **all** criteria must be satisfied concurrently. This is the opposite of the OR function, enforcing a strict conjunction of requirements, providing a precise filtering mechanism for complex datasets in Power BI. For example, a transaction might only be flagged as **Suspicious** if the dollar amount exceeds $10,000 AND the customer's account is less than 30 days old.

Similar to the OR method, we nest the AND function within the logical test parameter of the primary IF statement. The structure forces the logical check to confirm that every condition listed within the

`AND` function returns TRUE before the primary IF statement resolves to TRUE. If even a single condition within the `AND` statement is FALSE, the entire logical test fails, and the `value_if_false` result is returned.

```
Rating =
IF(
AND(
'my_data' > 20,
'my_data' > 4
),
"Good",
"Bad"
)
```

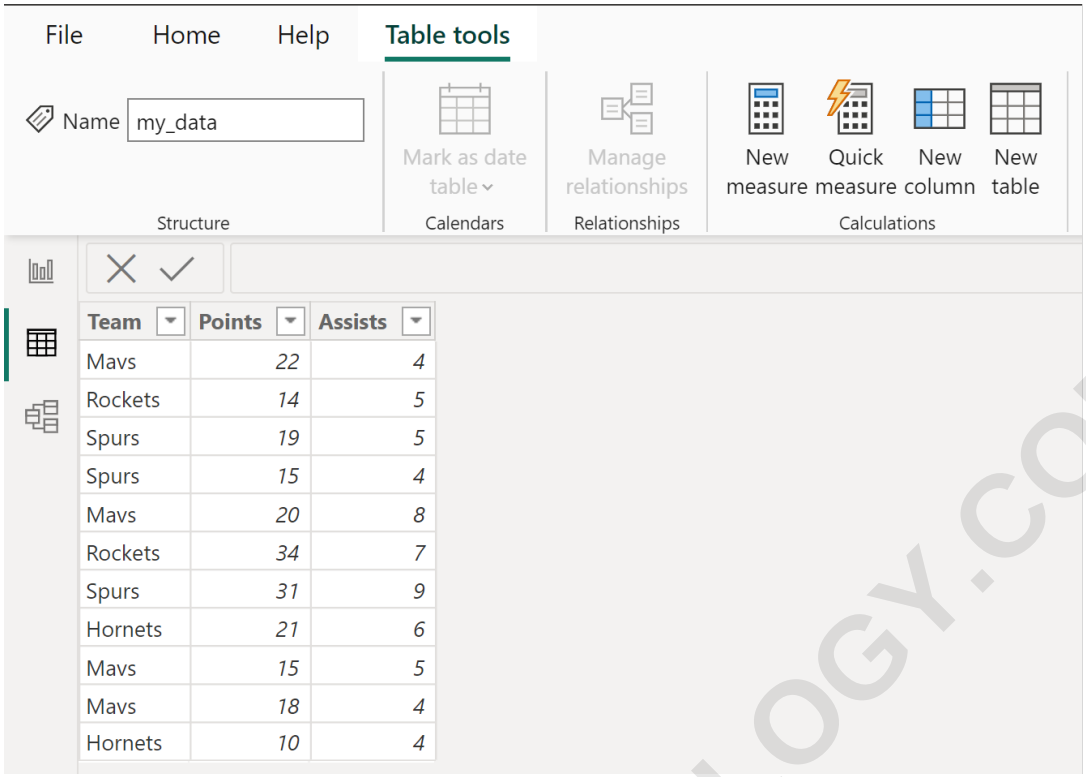
This implementation creates the **Rating** calculated column. It returns "Good" only if the value in the **Points** column is greater than 20 **and** the value in the **Assists** column is simultaneously greater than 4. If both of these conditions are not met, then the function returns "Bad" instead. This method is preferred when data classification requires the simultaneous fulfillment of multiple, non-negotiable criteria.

Practical Demonstration: Setting Up the Scenario

To illustrate the practical differences between the OR and AND conditional operators, we will utilize a small dataset. This hypothetical table, named **my_data** in Power BI, tracks athlete performance metrics. The goal is to categorize each player's performance using calculated columns based on their individual scores in **Points** and **Assists**.

The sample data below provides a clear baseline for testing our conditional logic. Observe the variation in performance across the rows: some players excel in Points, some in Assists, and a few in both. Our subsequent formulas will assign a **Rating** to each row based on how these individual values interact with the logical thresholds we set (Points > 20 and Assists > 4).

The following examples demonstrate how to utilize each conditional method in practice using the sample table in Power BI named **my_data**:

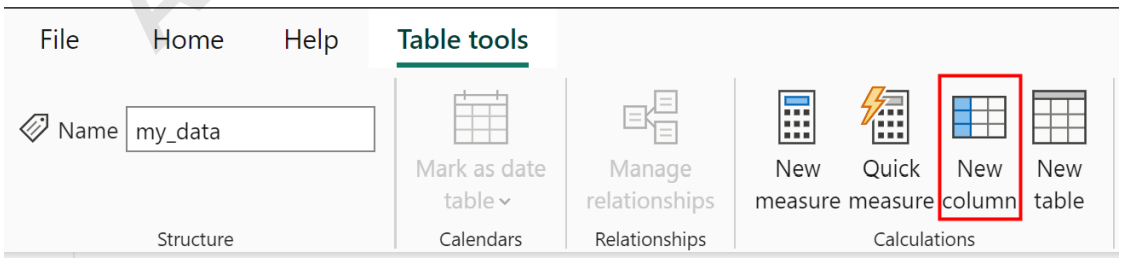


Team	Points	Assists
Mavs	22	4
Rockets	14	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	34	7
Spurs	31	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

Example 1: Implementing the Inclusive OR Condition

Suppose our objective is to add a new column that classifies players as "Good" if the value in the **Points** column is greater than 20 **or** if the value in the **Assists** column is greater than 4. This inclusive logic ensures that success in either key metric results in a positive rating, rewarding specialized performance.

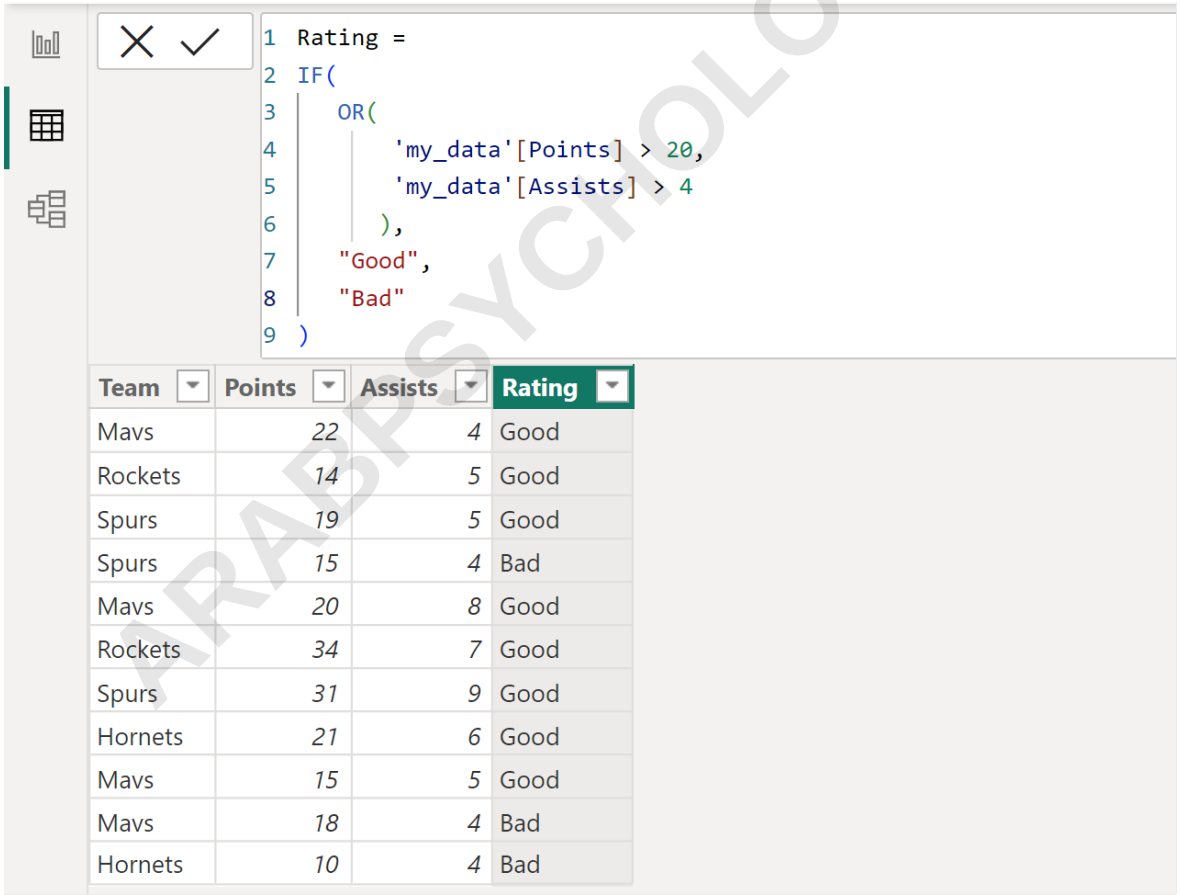
To begin this process, navigate to the table containing your data in Power BI Desktop and click the **New column** icon, typically found in the Table tools ribbon. This action initiates the creation of a calculated column and opens the formula bar where the DAX expression must be entered.



Subsequently, input the following DAX formula into the formula bar. This expression utilizes the IF statement combined with the OR function to implement the inclusive multiple condition logic:

```
Rating =
IF(
OR(
'my_data' > 20,
'my_data' > 4
),
"Good",
"Bad"
)
```

Upon execution, the resulting table will show the new **Rating** column populated according to the OR logic. For instance, players with moderate Points but high Assists (like Row 5, 15 Points and 5 Assists) are correctly classified as "Good" because the second condition was met, highlighting the power of the inclusive OR operation.



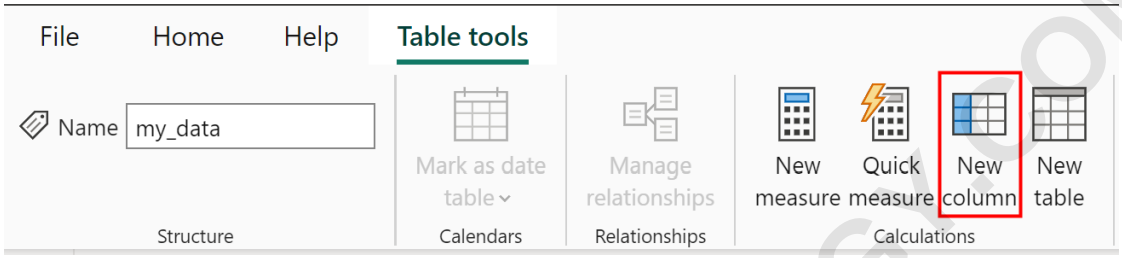
Team	Points	Assists	Rating
Mavs	22	4	Good
Rockets	14	5	Good
Spurs	19	5	Good
Spurs	15	4	Bad
Mavs	20	8	Good
Rockets	34	7	Good
Spurs	31	9	Good
Hornets	21	6	Good
Mavs	15	5	Good
Mavs	18	4	Bad
Hornets	10	4	Bad

Example 2: Implementing the Exclusive AND Condition

Now, let us define a stricter categorization. We require the player to be rated "Good" only if the

value in the **Points** column is greater than 20 **and** the value in the **Assists** column is simultaneously greater than 4. This ensures that the high rating is reserved only for elite players who satisfy both performance benchmarks, rewarding comprehensive excellence.

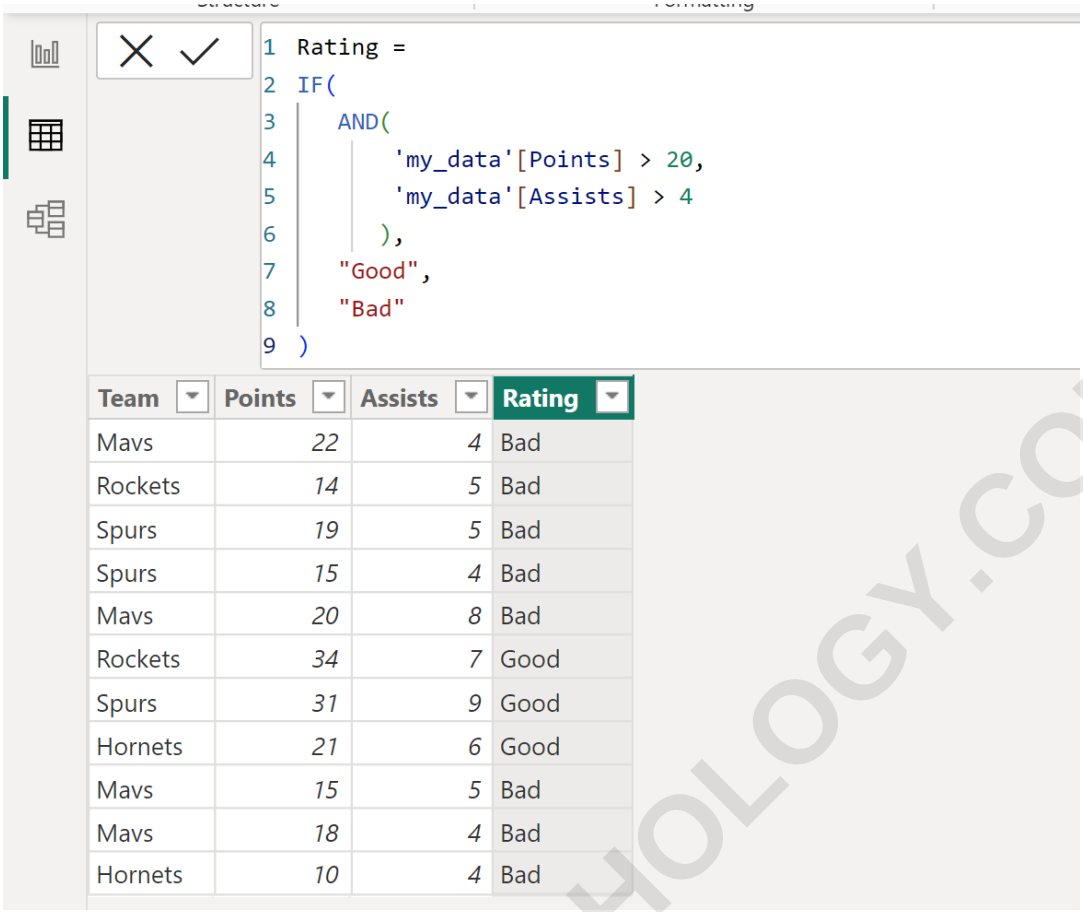
To implement this restrictive logic, we must once again initiate the creation of a calculated column. Click the **New column** icon in the Power BI interface. While the procedural step remains the same as in Example 1, the change in the internal DAX logic will drastically affect the final categorization outcome.



Next, input the following DAX formula into the formula bar. This expression uses the IF statement combined with the AND function to enforce simultaneous fulfillment of the conditions:

```
Rating =
IF(
AND(
'my_data' > 20,
'my_data' > 4
),
"Good",
"Bad"
)
```

Executing this formula will create a new column named **Rating**. You will observe that fewer rows qualify as "Good" compared to the OR condition. For example, the player in Row 5 (15 Points, 5 Assists) is now rated "Bad" because they failed the Points condition, illustrating the stringent nature of the AND function.



```
Rating =
IF(
    AND(
        'my_data'[Points] > 20,
        'my_data'[Assists] > 4
       ),
    "Good",
    "Bad"
)
```

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

Handling More Complex Logic: Beyond Binary Outcomes

While combining IF statements with AND or OR is perfect for binary TRUE/FALSE outcomes, real-world data often requires categorization into three or more levels (e.g., Low, Medium, High). In these scenarios, nesting multiple IF statements can quickly become confusing and difficult to maintain:

```
IF(condition1, result1, IF(condition2, result2, IF(condition3, result3, default_result)))
```

For complex, multi-tiered categorization requirements, the **SWITCH(TRUE(), ...)** function in DAX is the recommended professional solution. By setting the expression parameter to `TRUE()`, the function evaluates a series of logical tests sequentially until it finds the first one that returns TRUE, executing the corresponding result. This significantly improves readability and maintainability over deep nesting.

When selecting your conditional strategy in Power BI, remember the distinction: use IF(AND/OR) for simple binary classification based on complex inputs, and use **SWITCH(TRUE())** for managing multiple distinct outcomes or hierarchical categorization.

Further Learning Resources

To deepen your understanding of Data Analysis Expressions and conditional logic within Power BI, consulting the official documentation is highly recommended. The following resources provide detailed technical specifications for the functions discussed:

The complete documentation for the **IF** function in DAX.
The official reference for the **OR** function, including usage with multiple arguments.
Detailed examples of the **AND** function used in various calculation contexts.

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