Learning to Write IF Statements with Multiple Conditions in Power BI

Authored by

Mohammed looti

November 12, 2025

Understanding Conditional Logic in DAX

When harnessing the analytical power of Power BI for sophisticated business intelligence and data modeling tasks, developers frequently encounter the need to classify or flag data based on complex input criteria. Implementing this essential conditional logic requires proficiency in DAX (Data Analysis Expressions). Specifically, we achieve multi-condition testing by seamlessly integrating the foundational IF function with powerful Boolean operators. Mastering this combination is paramount for translating nuanced business requirements directly into derived metrics or calculated columns.

The standard DAX syntax provides an elegant framework for constructing an IF statement capable of evaluating two or more conditions simultaneously. This is executed by nesting the requisite logical tests--typically defined using the OR or AND functions--within the first argument of the IF function itself. If the combined logical test evaluates successfully, the calculated column returns the **TRUE** result; otherwise, it returns the specified **FALSE** alternative.

Before attempting practical implementation, it is vital to grasp the core difference between the two primary logical operators. The choice between using the OR condition and the AND condition fundamentally determines the restrictiveness of the classification rules. The OR operator is inherently inclusive, requiring only that a minimum of one criterion is met for qualification. Conversely, the AND operator is highly exclusive, demanding absolute adherence to all specified rules before returning a positive result.

Method 1: Implementing the OR Condition for Flexible Criteria

The OR function facilitates highly flexible conditional testing within a DAX calculated column. When incorporated into an IF statement, the overall logic immediately evaluates to **TRUE** as soon as any one of the specified conditions is satisfied. This structure is ideally suited for creating broad classifications where a data record qualifies based on excellence or compliance in any one of several defined areas, thereby establishing a lower barrier to entry for the desired category.

To illustrate this inclusive logic, consider the following syntax designed to define a new column named **Rating**. This formula leverages the OR condition to test two different criteria within a table called 'my_data'. This configuration ensures that if a player demonstrates high performance in either **Points** or **Assists**, they are immediately granted the "Good" rating.

```
Rating =
IF(
OR(
'my_data' > 20,
'my_data' > 4
```

```
),
"Good",
"Bad"
)
```

In this specific setup, the resulting column **Rating** will return "Good" if the value in the **Points** column exceeds 20 OR if the value in the **Assists** column exceeds 4. If, and only if, neither of these conditions is met--meaning Points are 20 or less AND Assists are 4 or less--does the OR function return **FALSE**, resulting in the label "Bad." This inclusive method guarantees recognition for performance excellence across alternative paths or specializations.

This versatile conditional structure proves highly effective across various data classification tasks, such as filtering inventory (if Color is Red OR Size is XL), or segmenting customer populations (if total Spend is high OR Purchase Frequency is high). It functions as a broad filter, allowing data to qualify based on multiple non-exclusive paths.

Method 2: Utilizing the AND Condition for Strict Requirements

In stark contrast to the inclusive nature of the OR function, the AND function is used to enforce stringent compliance. For the overall logical test to resolve to **TRUE**, it is an absolute requirement that every single condition nested within the AND operator must also simultaneously evaluate to **TRUE**. This method is indispensable when defining premium categories, identifying records that meet a comprehensive set of mandatory and high-level requirements, or ensuring regulatory compliance.

The following DAX structure demonstrates the precise application of the AND condition. Notice that the "Good" rating is now strictly reserved only for players who successfully meet both the high scoring benchmark (Points > 20) and the high assisting benchmark (Assists > 4) in the same row of data.

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

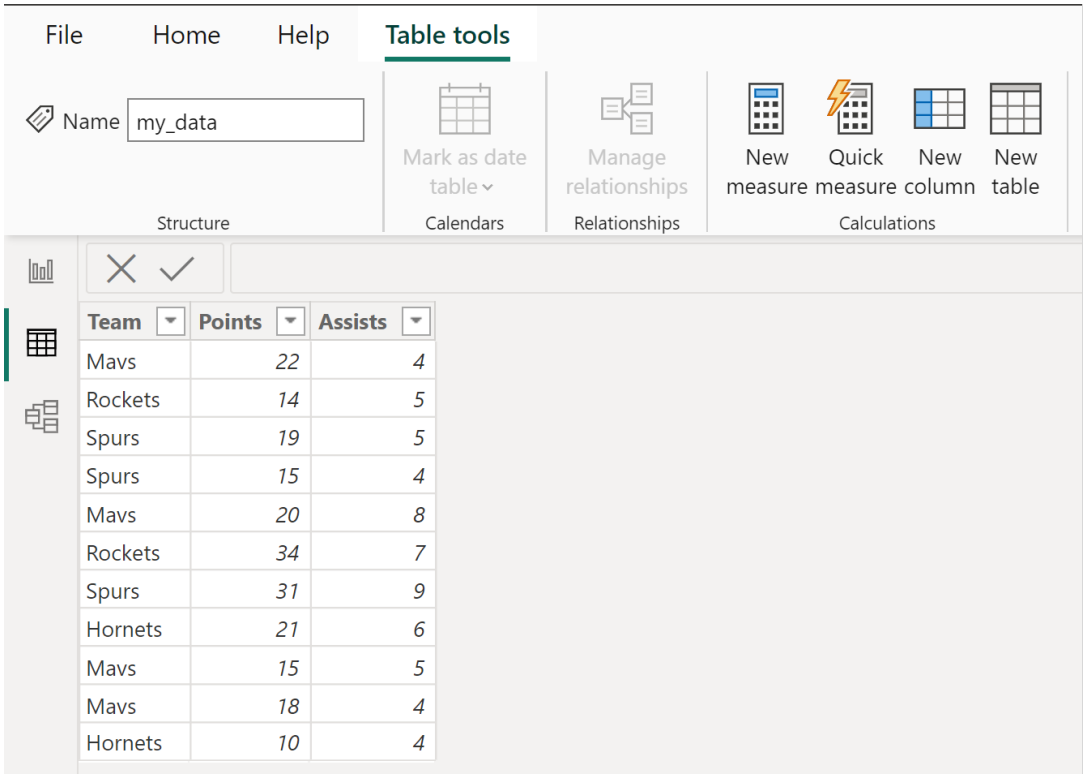
This specific formula generates a new column named **Rating** that returns "Good" only if the value in the **Points** column is greater than 20 AND the value in the **Assists** column is greater than 4. If even one of these essential criteria fails, the entire AND function immediately returns **FALSE**, and the resulting classification is "Bad." This technique is the definitive method for performing data filtering based on multi-faceted criteria where all conditions must be satisfied concurrently.

The AND operator is critical for scenarios requiring rigorous qualification, such as identifying truly successful sales transactions (Revenue greater than X AND Profit Margin greater than Y) or filtering complex compliance records (Transaction Date is within range AND Status is Approved). It mandates a comprehensive and exhaustive evaluation, ensuring that only records meeting the highest, combined standards are correctly classified.

Practical Application: Case Studies in Power BI

To fully internalize the critical distinction between the inclusive OR logic and the restrictive AND logic, we will now apply them within a concrete environment in Power BI. Using a simple performance table named **my_data**, which tracks scores for several individuals, we can clearly observe how these contrasting conditional rules dramatically impact the final classification outcomes for each record.

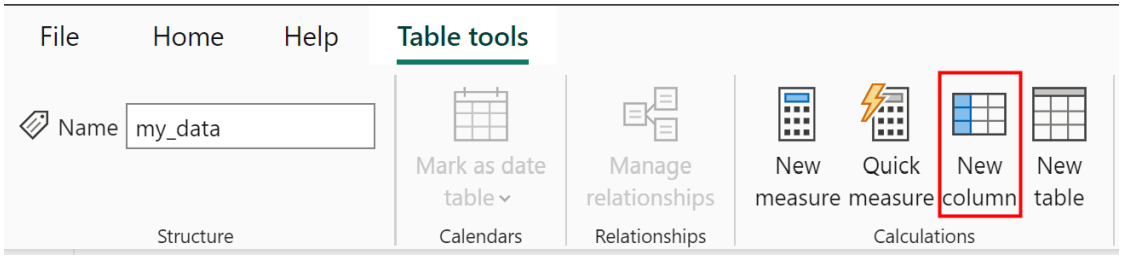
The sample data set we will use for reference throughout the following examples is displayed below. Pay close attention to the varying levels of performance recorded across the key metrics: **Points** and **Assists**. This variance will highlight the specific rows that qualify under each logical rule.



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

Applying the OR Condition Example

Our objective here is to generate a new calculated column that classifies players as "Good" if they have achieved more than 20 **Points** OR more than 4 **Assists**. This process begins in the Power BI interface by clicking the **New column** icon, which prepares the DAX formula bar for input.

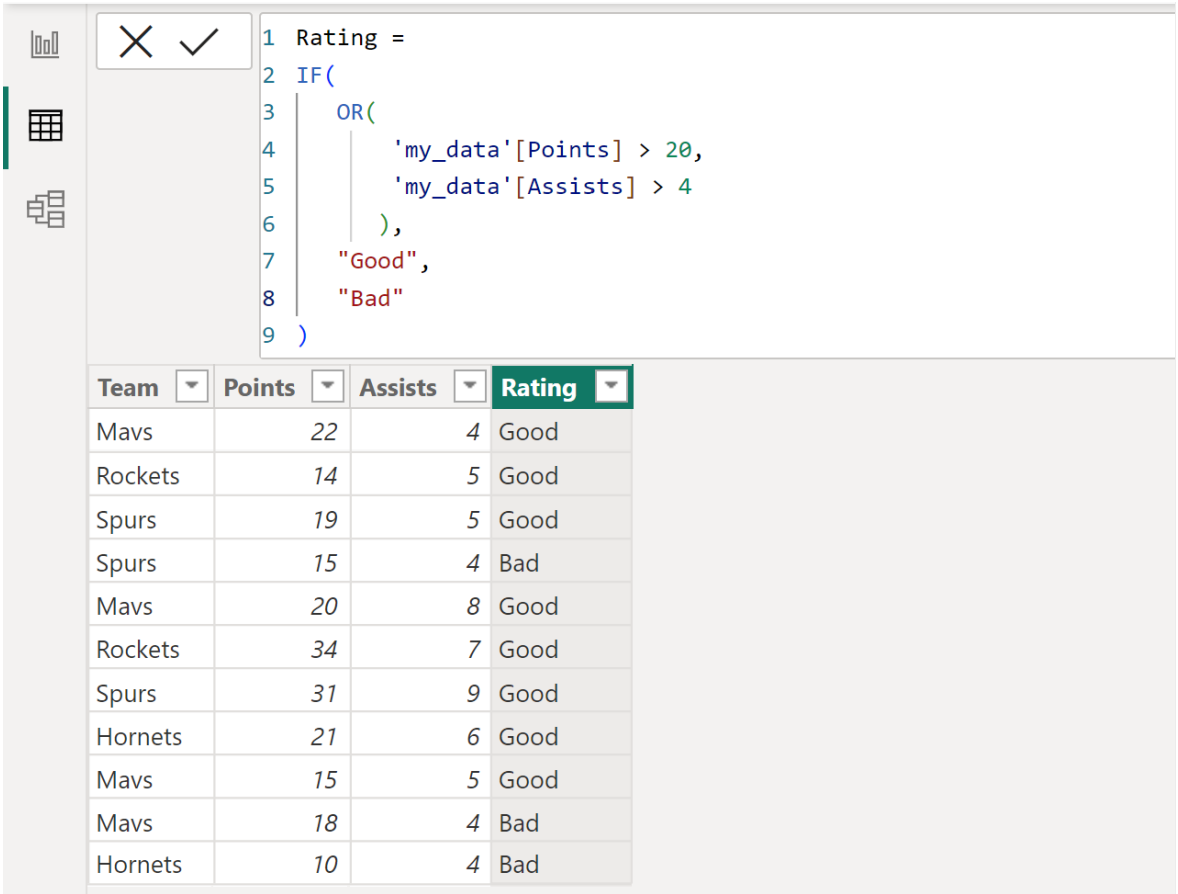


The specific formula utilizing the inclusive OR function is then carefully entered and committed to the model:

```
Rating =
IF(
OR(
'my_data' > 20,
'my_data' > 4
```

),
"Good",
"Bad"
)

The resulting table confirms that the OR logic successfully flags any player meeting at least one of the defined criteria. For instance, notice a player who scored 25 Points but only 3 Assists is still correctly rated "Good." This occurs because the Points condition was met, fulfilling the requirement even though the Assists condition was not. This result powerfully demonstrates how the OR operator accommodates specialization and alternative paths to qualification.

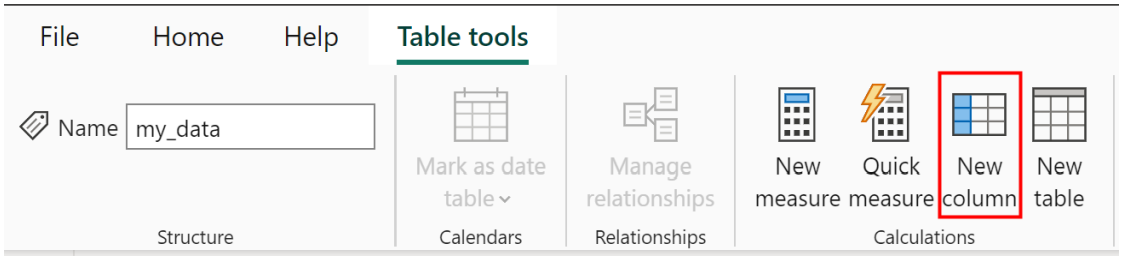


```
Rating =
IF(
    OR(
        'my_data'[Points] > 20,
        'my_data'[Assists] > 4
      ),
    "Good",
    "Bad"
)
```

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

Applying the AND Condition Example

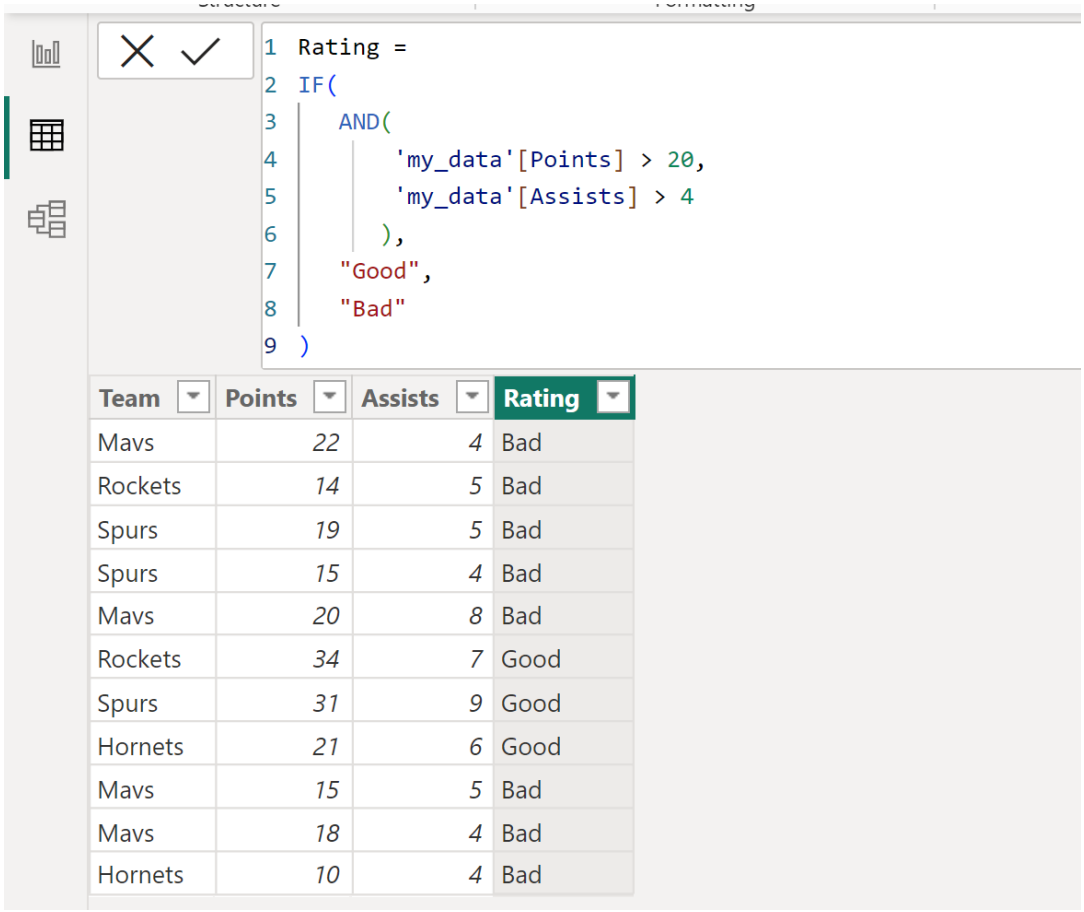
Next, we move to enforce a much stricter qualification rule: a player must be rated "Good" only if they simultaneously possess more than 20 **Points** AND more than 4 **Assists**. We initiate the column creation process again using the **New column** icon in the Power BI interface.



The DAX formula is then updated, replacing the inclusive OR function with the restrictive AND function to mandate dual compliance:

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

This action creates a new column named **Rating** that contains the value "Good" or "Bad" based exclusively on whether the corresponding values in both the **Points** and **Assists** columns meet their thresholds. The resulting table clearly shows that significantly fewer players qualify for the "Good" rating under the AND condition, as both criteria must be satisfied simultaneously for a positive classification. This result effectively illustrates the precision and restrictiveness that the AND operator injects into data classification efforts.



```
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

Advanced Considerations and Additional Resources

While the nested IF statement structure utilizing OR and AND is exceptionally powerful for simple binary classifications (True/False or Good/Bad), DAX offers alternative, more scalable functions for managing complex scenarios. When your requirement involves classifying data into multiple outcomes--such as "High," "Medium," and "Low" categories--relying on extensively nested IF statements can quickly become cumbersome, difficult to debug, and detrimental to code readability. In these situations, the SWITCH function is overwhelmingly the superior choice, providing a cleaner, more modular syntax for evaluating sequential conditions efficiently.

Furthermore, it is critical for performance optimization to remember that all calculated columns, including those created using complex DAX logic, contribute to increasing the size of your Power BI data model and can negatively affect query speed. For conditional logic that does not rely on dynamic filtering (i.e., simple row-by-row categorization based on static values), developers should consider implementing this transformation earlier in the data pipeline using Power Query (M language). This practice helps improve overall model efficiency and reduces the computational strain placed on the DAX calculation engine during report interaction.

For ongoing professional development and to explore the most advanced features, always prioritize referencing the official documentation. The comprehensive resources provided by Microsoft offer detailed explanations for every facet of the IF function, the OR function, and the AND function, ensuring you utilize these foundational tools both effectively and efficiently within your enterprise data models.

Additional Resources

The following curated tutorials explain how to perform other common and advanced data manipulation tasks in Power BI, helping you deepen your technical understanding of conditional structuring and data modeling:

Mastering the DAX SWITCH Function for Multi-Value Logic.
A Guide to Creating and Using Calculated Tables in Power BI.
Performance Tips: Optimizing DAX for Large Data Models.