How to Perform Conditional Lookups in Google Sheets Using IF with INDEX MATCH

Authored by

stats writer

January 17, 2026

Example: Use IF Function with INDEX MATCH in Google Sheets

The Power of Conditional Lookups in Google Sheets

The IF Function and the INDEX Function paired with the MATCH Function are two incredibly potent tools within Google Sheets, designed to seamlessly integrate data manipulation and complex logic. When utilized in combination, they enable sophisticated conditional lookups. This powerful synergy allows the user to introduce a dynamic layer of decision-making into their data retrieval process. Specifically, the IF Function first evaluates a specific condition, and only if that condition is met does the sophisticated INDEX Function and MATCH Function combination execute a precise search, retrieving a corresponding value from a designated column within a defined range. This methodology empowers users to highly customize their lookup formulas, making them significantly more responsive and adaptable based on external parameters or logical requirements evaluated in real-time. This combined function is invaluable across numerous data analysis and complex data management scenarios where simple, fixed lookups are insufficient.

The core principle behind this combination is to leverage the IF Function to control the flow of execution. The IF statement acts as a gatekeeper: it tests a condition, and if the condition evaluates to TRUE, it executes one defined action (the first INDEX MATCH lookup); if the condition evaluates to FALSE, it executes a secondary action (which is often a nested IF statement leading to a second INDEX MATCH lookup). This nesting capability is critical for handling multiple datasets or complex scenarios where the desired outcome depends on one of several possible states. Instead of manually updating formulas when the source data changes, this dynamic approach automates the selection process, guaranteeing efficiency and minimizing human error in complex reporting environments.

Mastering this method moves a user beyond basic spreadsheet operations into advanced data modeling, enabling the creation of highly interactive dashboards and reports. The ability to perform conditional lookups means that a single cell can serve as a control panel for various data views. By simply changing a value in a reference cell (the condition tested by the initial IF), the entire subsequent calculation chain--the precise range searched, the row identified, and the column data returned--changes instantly. This level of flexibility is often sought in financial modeling, inventory management, and academic data processing, where results must shift depending on variables like time period, location, or product category.

Understanding the Components: IF, INDEX, and MATCH

To effectively utilize the combined formula, one must first grasp the distinct roles of the three involved functions. The IF Function follows the basic structure: `IF(logical_expression, value_if_true, value_if_false)`. The `logical_expression` is any test that results in TRUE

or FALSE, such as checking if cell B1 equals "Mavs". If TRUE, the function returns the `value_if_true` argument, which in our case, will be an entire INDEX MATCH formula. If FALSE, it returns the `value_if_false` argument, which can be another nested IF statement, allowing us to test for subsequent conditions sequentially. This nested structure forms the backbone of the conditional selection process, guiding the spreadsheet to the correct dataset.

The INDEX Function is responsible for retrieving the final value, based on coordinates provided to it. Its structure is `INDEX(reference, row_number, column_number)`. The `reference` is the entire data range where the final answer lies (e.g., A7:D9). Crucially, the `row_number` is dynamically supplied by the MATCH function, and the `column_number` is a fixed integer corresponding to the column from which we want the output (e.g., column 3 for 'Rebounds'). Unlike VLOOKUP, which is limited to searching the leftmost column, the INDEX function allows the lookup column to be anywhere within the specified range, offering superior flexibility and stability when columns are inserted or deleted.

The MATCH Function acts as the navigational component, locating the exact row number where the desired lookup value resides. Its basic syntax is `MATCH(search_key, range, search_type)`. The `search_key` is the specific item we are looking for (e.g., "Guard"). The `range` is the single column we are searching within (e.g., A7:A9). The `search_type` is almost always set to 0 for an exact match, ensuring data integrity. The output of the MATCH function is not the value itself, but the relative position (row number) of that value within the specified search range. This numerical result is then seamlessly fed directly into the `row_number` argument of the INDEX function, completing the high-precision lookup mechanism.

Why Combine IF and INDEX MATCH?

The necessity of combining these functions arises when dealing with heterogeneous datasets residing within the same spreadsheet environment. Imagine a reporting scenario where metrics for different regional offices are stored in separate tables. A simple VLOOKUP or a standard INDEX MATCH formula can only look up data in one fixed table. If the user needs to switch instantly between the North Region data and the South Region data based on a dropdown selection, a conditional mechanism is required. The IF Function provides this essential routing logic, determining which dataset (which INDEX MATCH array) is active at any given moment. Without the IF function, one would have to manually rewrite the INDEX MATCH range every time the target dataset changed.

Furthermore, nesting the powerful INDEX MATCH structure within an IF framework allows for highly specific error handling or default actions. For instance, if the primary condition (Team = "Mavs") is not met, the IF statement can be programmed not just to test a second condition (Team = "Pacers"), but potentially to return a custom text string like "Team Not Found" or a specific

default value, if neither condition is met. This conditional execution prevents the display of ugly error messages such as `#N/A` or `#REF!`, enhancing the user experience and the overall professionalism of the spreadsheet output. It transforms a rigid lookup tool into a flexible, error-tolerant solution capable of advanced data validation.

The core advantage lies in creating truly dynamic references. If you have five teams, you could potentially nest four IF statements, each containing a complete INDEX MATCH setup targeting a unique data table corresponding to that team. While this can lead to long formulas, it guarantees that the output cell (where the formula resides) always returns the correct, contextually relevant data point, based on the selection made elsewhere in the sheet. This technique is far superior to attempting to use complex array formulas like `FILTER` or `QUERY` when the fundamental need is a simple selection between predefined, separate data blocks based on a single condition.

Deconstructing the Complex Formula Syntax

The provided formula demonstrates a classic nested IF structure used for dynamic table selection. Let's break down the exact syntax used here, which relies on the logic of the IF statement to choose which INDEX MATCH to execute:

=IF(B1="Mavs",(INDEX(A7:D9,MATCH("Guard",A7:A9,0),3)),IF(B1="Pacers",(INDEX(A13:D15,MATCH("Guard",A13:A15,0),3))))

The formula begins with the primary IF Function: `IF(B1="Mavs", , )`. The initial test checks if the contents of cell **B1** exactly match the text string "Mavs". If this condition returns TRUE, the formula executes the first embedded lookup: `(INDEX(A7:D9, MATCH("Guard", A7:A9, 0), 3))`. This lookup specifically targets the Mavs dataset, defined in the range **A7:D9**. Within this range, the MATCH Function finds the row position of "Guard" within the key column **A7:A9**, and the INDEX Function then returns the value from the third column (column 3) of the overall range **A7:D9**.

If the initial test (B1="Mavs") returns FALSE, the formula moves directly to the `Value if False` argument, which is a nested IF statement: `IF(B1="Pacers", , )`. This secondary test checks if cell **B1** matches "Pacers". If this second condition returns TRUE, the formula executes the second embedded lookup: `(INDEX(A13:D15, MATCH("Guard", A13:A15, 0), 3))`. This lookup targets the Pacers dataset, defined in range **A13:D15**. The INDEX Function here retrieves the corresponding data point from column 3 of the Pacers range, based on the row located by the internal MATCH function searching for "Guard" in **A13:A15**. If B1 is neither "Mavs" nor "Pacers", the final value returned by the outermost IF statement's last argument (which is implicitly empty or a default value) would be FALSE or an error, unless a third, final value is explicitly provided.

Practical Application: Setting Up the Dataset

To visualize this powerful application of Google Sheets, consider a practical scenario involving two separate data tables detailing information about professional basketball players. Each table contains the player's position, their height, and specific performance statistics such as rebounds and assists. Our goal is to create a dynamic summary cell that retrieves the 'Rebounds' statistic for the position 'Guard', regardless of which team (Mavs or Pacers) is specified in an input cell.

We assume the following layout for our data structure. The first dataset, corresponding to the "Mavs," is located in the range **A7:D9**. The second dataset, for the "Pacers," is located further down the sheet in range **A13:D15**. Crucially, the structure of both tables is identical: Column 1 holds the Position, Column 2 holds Height, Column 3 holds Rebounds, and Column 4 holds Assists. An input cell, **B1**, is reserved for the user to type in the desired team name ("Mavs" or "Pacers").

	A	B	C	D
1	**Team**	Mavs		
2	**Rebounds**			
3				
4				
5	**Mavs**			
6	**Position**	**Points**	**Rebounds**	**Assists**
7	Guard	22	2	8
8	Forward	20	8	3
9	Center	34	12	3
10				
11	**Pacers**			
12	**Position**	**Points**	**Rebounds**	**Assists**
13	Guard	14	4	10
14	Forward	12	15	7
15	Center	10	10	4
16				
17				
18				

The core requirement is to return the rebounds value for the "Guard" position, selected dynamically based on the team name entered into cell **B1**. This task perfectly illustrates the need for conditional lookups, as the formula must first decide which of the two non-contiguous ranges (A7:D9 or A13:D15) to use before executing the lookup for "Guard." The structure of the data and the placement of the input cell B1 are essential prerequisites for correctly writing and interpreting the

nested IF INDEX MATCH formula.

Step-by-Step Implementation of the Formula

We will now place the complete, nested formula into cell **B2**, which is the designated output cell for our dynamic lookup result. The purpose of this formula is to execute a search within the appropriate dataset selected by the value in **B1**. The lookup criteria is fixed ("Guard"), and the desired return value is also fixed (the value in column 3, Rebounds).

The complete formula structure, carefully accounting for the specific cell ranges and the fixed column index of 3 (Rebounds), is entered as follows into cell **B2**:

=IF(B1="Mavs",(INDEX(A7:D9,MATCH("Guard",A7:A9,0),3)),IF(B1="Pacers",(INDEX(A13:D15,MATCH("Guard",A13:A15,0),3))))

When this complex formula is processed by Google Sheets, the process starts with the outermost IF Function checking cell B1. If B1 equals "Mavs," the first INDEX MATCH executes. This INDEX MATCH searches the Position column of the Mavs data (A7:A9) for "Guard." Assuming "Guard" is in the second row of that range (A8), MATCH returns 2. INDEX then looks into the Mavs data array (A7:D9) and retrieves the value at row 2, column 3 (Rebounds), providing the corresponding rebound count. If B1 does not equal "Mavs," the formula bypasses the entire first lookup and immediately checks for "Pacers" using the nested IF statement, which then executes the analogous lookup in the Pacers data array (A13:D15).

Analyzing the Results: Dynamic Lookups in Action

Let us observe the resulting output when different values are input into the control cell, **B1**. If the user initially sets the value in cell **B1** to "Mavs," the formula successfully passes the first logical test. The first part of the formula is executed, directing the lookup to the Mavs dataset (A7:D9). The internal MATCH Function identifies the row corresponding to "Guard," and the INDEX Function retrieves the value from the third column of that row, which is the Rebounds statistic.

B2 =IF(B1="Mavs",(INDEX(A7:D9,MATCH("Guard",A7:A9,0),3)

	A	B	C	D	E
1	**Team**	Mavs			
2	**Rebounds**	2			
3					
4					
5	**Mavs**				
6	**Position**	**Points**	**Rebounds**	**Assists**	
7	Guard	22	2	8	
8	Forward	20	8	3	
9	Center	34	12	3	
10					
11	**Pacers**				
12	**Position**	**Points**	**Rebounds**	**Assists**	
13	Guard	14	4	10	
14	Forward	12	15	7	
15	Center	10	10	4	
16					
17					
18					

As shown in the screenshot, since **B1** is set to "Mavs," the formula returns the rebounds value for the Guard from the Mavs dataset. In the provided example, the corresponding value for the Guard in the Mavs table is **2**, confirming the successful execution of the conditional lookup against the correct data range. The formula correctly bypasses the Pacers lookup entirely because the initial condition was met and the first resulting value was returned. This instantaneous selection demonstrates the high efficiency of this combined method in dynamically selecting data sources.

If we then modify the input in cell **B1** by changing the value to "Pacers," the entire conditional lookup system automatically updates. The initial IF Function test (B1="Mavs") now fails (returns FALSE). The formula proceeds to the nested IF statement, where the condition (B1="Pacers") is met (returns TRUE). Consequently, the second INDEX MATCH lookup is executed, directing the search to the Pacers dataset (A13:D15).

B2 =IF(B1="Mavs",(INDEX(A7:D9,MATCH("Guard",A7:A9,0),3)),IF

	A	B	C	D	E
1	**Team**	Pacers			
2	**Rebounds**	4			
3					
4					
5	**Mavs**				
6	**Position**	**Points**	**Rebounds**	**Assists**	
7	Guard	22	2	8	
8	Forward	20	8	3	
9	Center	34	12	3	
10					
11	**Pacers**				
12	**Position**	**Points**	**Rebounds**	**Assists**	
13	Guard	14	4	10	
14	Forward	12	15	7	
15	Center	10	10	4	
16					
17					
18					

The result is instantly updated in cell B2. The formula now returns a value of **4**, which accurately reflects the rebounds value for the Guard position found within the Pacers dataset. This powerful, automated response to changes in the control cell highlights the utility of nesting lookups within conditional statements, creating a single, robust formula capable of handling multiple, distinct data sources based on user input. This avoids the necessity of creating multiple lookup cells or relying on complex auxiliary columns for selection.

Advanced Considerations and Alternatives

While nesting IF and INDEX MATCH formulas is highly effective for two or three distinct lookup tables, this technique can become cumbersome if you need to manage five, ten, or even twenty separate datasets. Every new dataset requires another nested IF statement, leading to extremely long and difficult-to-debug formulas. In scenarios involving a large number of possible datasets, alternative approaches are often more manageable and scalable. One such method involves creating a reference table that maps the team name (the lookup key in B1) directly to the starting row or range of its corresponding data table.

For highly complex, multi-criteria lookups across a unified data structure, the `QUERY` function in

Google Sheets offers immense power, using SQL-like syntax to filter and aggregate data conditionally. If the data were consolidated into one large table with an added 'Team' column, a single `QUERY` formula could replace the entire nested IF INDEX MATCH structure, greatly improving readability and maintainability. Another alternative involves using helper columns combined with the `ARRAYFORMULA` wrapper, which can sometimes simplify the conditional logic by pre-filtering data before the final lookup. However, these alternatives require a fundamental restructuring of the data layout, whereas the nested IF INDEX MATCH solution excels precisely because it handles multiple, separate, non-contiguous data blocks without needing consolidation.

Finally, users must be aware of the importance of the absolute vs. relative search range in the MATCH component. In this example, the ranges (A7:A9 and A13:A15) are fixed because they refer to specific, separate tables. However, if the data structure were slightly different and the ranges needed to be copied, appropriate use of dollar signs ($) for absolute referencing would be mandatory. When dealing with conditional logic that selects between defined, disparate ranges, it is generally safer to use absolute references for all range arguments within the INDEX and MATCH components, ensuring that the lookup remains focused on the correct table regardless of where the formula is copied.

Summary and Next Steps

The combination of the IF Function with the INDEX MATCH pairing provides an essential framework for creating dynamic, conditional lookups in Google Sheets. This technique allows a spreadsheet to intelligently select the appropriate data source based on a specific input, significantly enhancing flexibility compared to standard, fixed lookup functions. By understanding how the IF statement controls execution flow and how the INDEX MATCH pair efficiently retrieves positional data, users can build robust solutions for handling heterogeneous datasets within a single worksheet.

To solidify your understanding of this concept, we recommend practicing variations of the syntax. Try modifying the formula to look up a different column (e.g., Column 4 for 'Assists') or change the lookup key (e.g., search for 'Center' instead of 'Guard'). Furthermore, experiment with adding a third dataset and nesting a third IF statement to handle three conditional possibilities. This hands-on approach will ensure mastery of conditional data selection and prepare you for more complex data management tasks.

The following tutorials explain how to perform other common tasks in Google Sheets, building upon the foundational skills demonstrated here:

Understanding the use of absolute references in lookup formulas.
Implementing error trapping with IFERROR in conjunction with complex lookups.
Exploring the capabilities of the QUERY function for unified conditional reporting.