Learning Conditional Lookups in Google Sheets: Combining IF, INDEX, and MATCH Functions

Authored by

Mohammed looti

November 11, 2025

Mastering Conditional Lookups: The IF INDEX MATCH Synergy

The strategic combination of the IF function with the robust lookup power of the INDEX function and MATCH function creates an exceptionally powerful and flexible system for data retrieval within Google Sheets. This advanced technique moves far beyond the limitations of standard lookup mechanisms, enabling users to perform complex, conditional queries where the target data array is not fixed but is dynamically determined by a preceding logical condition. By meticulously nesting these core functions, we achieve a level of dynamism that is essential for effectively managing large, segmented datasets where the precise location of the required information depends entirely on a specific criterion defined elsewhere in the spreadsheet.

When dealing with multiple distinct data tables--perhaps representing different departments, geographical regions, or, as demonstrated in our example, various sports teams--it is highly inefficient and prone to error to write separate, manually updated formulas for every potential scenario. The IF function serves as the critical decision-maker, acting as a gatekeeper that routes the subsequent lookup process. It evaluates a logical test and, based on whether that test returns **TRUE** or **FALSE**, it executes one of two entirely different lookup routines. This ensures that the appropriate data table is queried automatically every time the condition changes, a fundamental practice for creating highly adaptive and maintenance-friendly spreadsheets.

Why Standard Lookups Fall Short

Fixed-array functions, such as VLOOKUP, inherently require the user to specify a static, unchanging range for the lookup operation. While excellent for simple tables, this rigidity becomes a major limitation when the data is segregated into multiple independent tables on the same sheet. A standard VLOOKUP cannot conditionally switch its lookup array from, say, the "Mavs" table to the "Pacers" table simply because a user changed a selection cell. This inability to handle dynamic array selection is precisely where the **IF INDEX MATCH** combination proves its superior value.

The requirement for a single output cell to dynamically switch between retrieving data from multiple entities is a common challenge in advanced data modeling. The solution lies in using the **IF** function not just for simple value substitution, but for determining the structure of the formula itself. The **IF** statement determines which range--which is defined by the subsequent **INDEX MATCH** pairing--should be activated. This methodology allows for the construction of a single, powerful formula that can successfully navigate and extract data from multiple source tables, based solely on the input provided in a control cell.

Analyzing the Nested IF INDEX MATCH Structure

The core of this technique is the strategic nesting of the IF function to establish a cascading

conditional logic. This structure dictates which specific dataset the formula will search before the actual high-precision lookup is performed.

The following specific formula structure demonstrates how to integrate this conditional logic seamlessly with the precision lookup capabilities of **INDEX** and **MATCH** in Google Sheets:

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

This particular formula initiates by checking if the value in cell **B1** is equal to "Mavs." If this condition is met (**TRUE**), the formula executes the first **INDEX MATCH** sequence, looking for "Guard" in the designated Mavs position column (**A7:A9**) and returning the value from the third column of the Mavs data range (**A7:D9**). However, if the initial check is **FALSE**, the formula immediately proceeds to the nested IF function, which then checks if the value in cell **B1** is equal to "Pacers." If this second condition is **TRUE**, the formula executes the second **INDEX MATCH** sequence, retrieving data from the Pacers' range (**A13:D15**).

Step-by-Step Implementation Example

To solidify the understanding of this conditional lookup method, we use a practical example involving two basketball team statistics tables. Imagine a spreadsheet containing two distinct datasets that track various statistics for players on the Mavs and the Pacers, respectively:

	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				

Our objective is to return the rebounds value for the "Guard" position, where the source team (Mavs or Pacers) is determined dynamically by the team name typed into cell **B1**. This requirement to switch the lookup table based on user input perfectly illustrates the necessity of the nested **IF INDEX MATCH** approach. A fixed VLOOKUP would be incapable of handling this conditional array selection.

To accomplish this dynamic retrieval task, we must input the complete nested formula into the designated output cell, typically cell **B2**, which will house the final result (the Guard's rebound count based on the selected team). Implementing this requires careful attention to ensuring that each conditional branch references the correct, segregated data range.

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

Deconstructing the Precision Lookup

Within each branch of the conditional logic, the **INDEX MATCH** pairing is responsible for locating the precise data point. The process starts with the MATCH function , which is tasked with locating the row number of the specified lookup criterion--in this case, the text string "Guard"--within the

designated Player Position column (either **A7:A9** or **A13:A15**, depending on the **IF** condition). Using an exact match criterion (0), the **MATCH** function reliably returns the relative row position of the Guard within that specific data array.

The row number calculated by the **MATCH** function is then seamlessly fed as an argument into the INDEX function . The **INDEX** function requires three pieces of information: the overall array (the table range selected by the **IF** statement), the row number (provided by **MATCH**), and the column number. In our example, the column number is hardcoded as **3**, corresponding to the "Rebounds" column within the selected range. This mechanism ensures that the **INDEX** function returns the exact cell value located at the intersection of the correct table, the correct position (row), and the desired statistic (column).

Achieving Dynamic Data Retrieval

The true utility of this structure is immediately evident when the control cell, **B1**, is modified. Initially, if **B1** is set to "Mavs," the formula executes the first conditional block:

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					

Since **B1** is "Mavs," the formula accurately returns **2**, which represents the rebounds value for the

Guard from the Mavs dataset (range **A7:D9**). The initial logical test returns **TRUE**, and the formula halts execution after the first result is returned.

Now, observe the dynamic switching capability. If we change the value in cell **B1** to "Pacers," the initial logical test ("B1='Mavs'") immediately returns **FALSE**. This action prompts the formula to evaluate the nested condition ("B1='Pacers'"). Since this second test now returns **TRUE**, the formula executes the second **INDEX MATCH** block, targeting the Pacers' data range (**A13:D15**). This instantaneous and automatic switch ensures the formula maintains data integrity across conditional changes.

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 formula updates instantaneously, returning a value of **4**, which is the accurate rebound count for the Guard from the Pacers dataset. This powerful technique is highly scalable; while this example uses two conditional checks, dozens of additional IF function statements can be nested, turning a single formula cell into a sophisticated, multi-range lookup tool capable of handling complex data segmentation across an entire Google Sheets project.

Scaling Conditional Logic and Advanced Resources

Mastering the conditional lookup using **IF INDEX MATCH** is a significant milestone in becoming

proficient in advanced spreadsheet management. This technique provides the structural framework necessary to handle complex data environments where the target array is inherently variable. Understanding how to correctly nest the **IF** statements and define the corresponding lookup ranges ensures that your spreadsheets are both robust and flexible. This approach is superior to manual data organization or reliance on static formulas when dealing with frequently changing or segmented data sources.

To further enhance your skills in complex data retrieval and manipulation tasks within Google Sheets, we recommend exploring tutorials that delve into related functions and techniques. Building upon your knowledge of conditional lookups will allow you to construct even more sophisticated and fault-tolerant sheets.

The following tutorials explain how to perform other common tasks in Google Sheets:

Understanding Array Formulas and their interaction with lookup functions.

Implementing conditional formatting based on the results of dynamic lookups.

Using error handling functions such as IFERROR to gracefully manage scenarios where the lookup value is not found.

Advanced data validation techniques for controlling the input values in selector cells like **B1**.