How to Return Multiple Values in Google Sheets Based on One Criteria

Authored by

mohammed looti

January 9, 2026

The ability of Google Sheets to handle complex data retrieval tasks is fundamental for advanced data analysis. Unlike simple functions that return a single result, many scenarios require the extraction of **multiple values** that satisfy a specific **single criteria**. This capability transforms a standard spreadsheet application into a powerful data manipulation tool, essential for tasks ranging from inventory management to detailed financial reporting. While functions like VLOOKUP are limited to returning only the first match, advanced users often leverage combinations of functions--or powerful alternatives like FILTER or QUERY--to efficiently retrieve entire subsets of data.

Mastering the technique for returning multiple corresponding records is crucial for anyone working with large or dynamic datasets. When dealing with criteria-based extraction, the primary challenge is to create a dynamic structure that adjusts automatically as the criteria change or as new matching data is added. This requires moving beyond basic lookup functions and employing powerful tools, particularly complex nested formulas or specific array functions designed for this purpose. This guide will focus on a robust, traditional method utilizing a nested array formula involving **INDEX**, **SMALL**, **IF**, and **ROW**, providing detailed clarity on how these components work together to deliver comprehensive results.

The Power of Advanced Array Formulas for Data Retrieval

While modern Google Sheets features like the dedicated FILTER function offer a straightforward syntax for this task, understanding the underlying principles of array logic is invaluable. Historically, and still often necessary in specific environments, a complex nested formula involving **INDEX** and **SMALL** has been the standard workaround for achieving multi-criteria lookups. This method forces the spreadsheet to evaluate conditions across an entire range, rather than just a single cell, effectively generating a list of row numbers where the criteria are met.

This particular approach relies on an Array Formula structure. An Array Formula calculates results on an array of values instead of a single value, making it extremely powerful for conditional processing. By embedding logical checks within the formula, we can instruct the sheet to identify all instances where a specified criterion matches a value within the designated criteria column. The output of this complex nesting is a sequence of row numbers that the **INDEX** function can then iterate through to return the associated values.

The following is the precise structure required to return multiple values in Google Sheets based on a single criteria. It is important to note that this formula must be entered as an Array Formula in many spreadsheet environments, though Sheets often handles this implicitly when using functions like **IF** within this nesting.

You can use the following formula to return multiple values in Google Sheets based on a single criteria:

=INDEX($A$1:$A$14, SMALL(IF(D$2=$B$1:$B$14, MATCH(ROW($B$1:$B$14), ROW($B$1:$B$14)), ""), ROWS($A$1:A1)))

This particular formula is designed to return all of the values located within the data range **A1:A14**, specifically where the corresponding value in the criteria range **B1:B14** is exactly equal to the single criteria value specified in cell **D2**. Understanding how each nested function contributes to this result is key to successful implementation and debugging.

Deconstructing the Complex INDEX/SMALL/IF Formula

To effectively utilize this solution, we must break down the purpose of each function within the nested structure. This intricate formula works by first identifying the row numbers of matches, then sorting those matches, and finally retrieving the data associated with those sorted row numbers. This sequential execution ensures that the returned values are listed consecutively without gaps, even if the matching data in the source table is scattered.

The IF Statement (`IF(D$2=$B$1:$B$14, ...)`): This is the core logical test. It checks every cell in the criteria range (`$B$1:$B$14`) against the specified criteria (`D$2`). If the condition is **TRUE** (a match is found), it proceeds to the next step. If it is **FALSE**, it returns an empty string (`""`).

The MATCH/ROW Combination (`MATCH(ROW($B$1:$B$14), ROW($B$1:$B$14))`): This is a sophisticated way to generate a sequential list of row numbers corresponding to the data set (in this case, 1 through 14) for only the rows where the condition was true. When combined with the **IF** function, it effectively creates an array containing only the relative row indices of the matching data, interspersed with empty strings where no match occurred.

The SMALL Function (`SMALL(..., ROWS($A$1:A1))`): The **SMALL** function is critical for iteration. It takes the array of row indices generated by the **IF** statement and finds the K-th smallest value. The `ROWS($A$1:A1)` part determines K. When you drag the formula down one cell (e.g., to A2), `ROWS($A$1:A2)` becomes 2, retrieving the second smallest row number. This mechanism ensures that the results are pulled sequentially, one match per cell, until all matches are retrieved.

The INDEX Function (`INDEX($A$1:$A$14, ...)`): Finally, the **INDEX** function uses the row number outputted by **SMALL** to look up and return the actual corresponding value from the designated return range (`$A$1:$A$14`). Since **SMALL** systematically returns the 1st, 2nd, 3rd, etc., matching row index, **INDEX** retrieves the corresponding data points in order.

The careful use of absolute and relative referencing (dollar signs, e.g., `$A$1` vs `A1`) is paramount here. The criteria range, return range, and the overall row reference arrays must be absolute (e.g., `$A$1:$A$14`) so they do not shift when the formula is dragged down. Conversely, the iteration

counter (`ROWS($A$1:A1)`) must be partially relative to allow the lower bound (A1) to remain fixed while the upper bound (A1, A2, A3...) expands, increasing the K value for the **SMALL** function.

Case Study: Retrieving NBA Finals Winners Data

To demonstrate the practical application of this powerful Array Formula, we will use a common scenario involving historical sports data. Suppose we have a dataset tracking the NBA finals winners over several years and we want to isolate all years associated with a specific team. This scenario perfectly illustrates the need for a multi-value return based on a single team name criteria.

Consider the following hypothetical dataset established in Google Sheets, spanning cells A1 through B14. Column A contains the Year (the value we want to return), and Column B contains the Winning Team (the criteria range).

The structure of our data is as follows:

	A	B	C	D
1	**Year**	**Winner**		
2	2010	Lakers		
3	2011	Mavs		
4	2012	Heat		
5	2013	Heat		
6	2014	Spurs		
7	2015	Warriors		
8	2016	Cavs		
9	2017	Warriors		
10	2018	Warriors		
11	2019	Raptors		
12	2020	Lakers		
13	2021	Bucks		
14	2022	Warriors		
15				
16				
17				

Our objective is to create a dynamic lookup that, when given a team name in a separate cell, returns a sequential list of all corresponding years in a new column. For this example, we will begin by attempting to return every year that the **Golden State Warriors** won the NBA championship, using "Warriors" as our initial criteria.

Step-by-Step Implementation and Initial Result

The implementation requires careful placement of both the criteria and the formula. We must designate a cell to hold our lookup criteria, allowing easy modification later, and then input the array formula into the first cell of the results column.

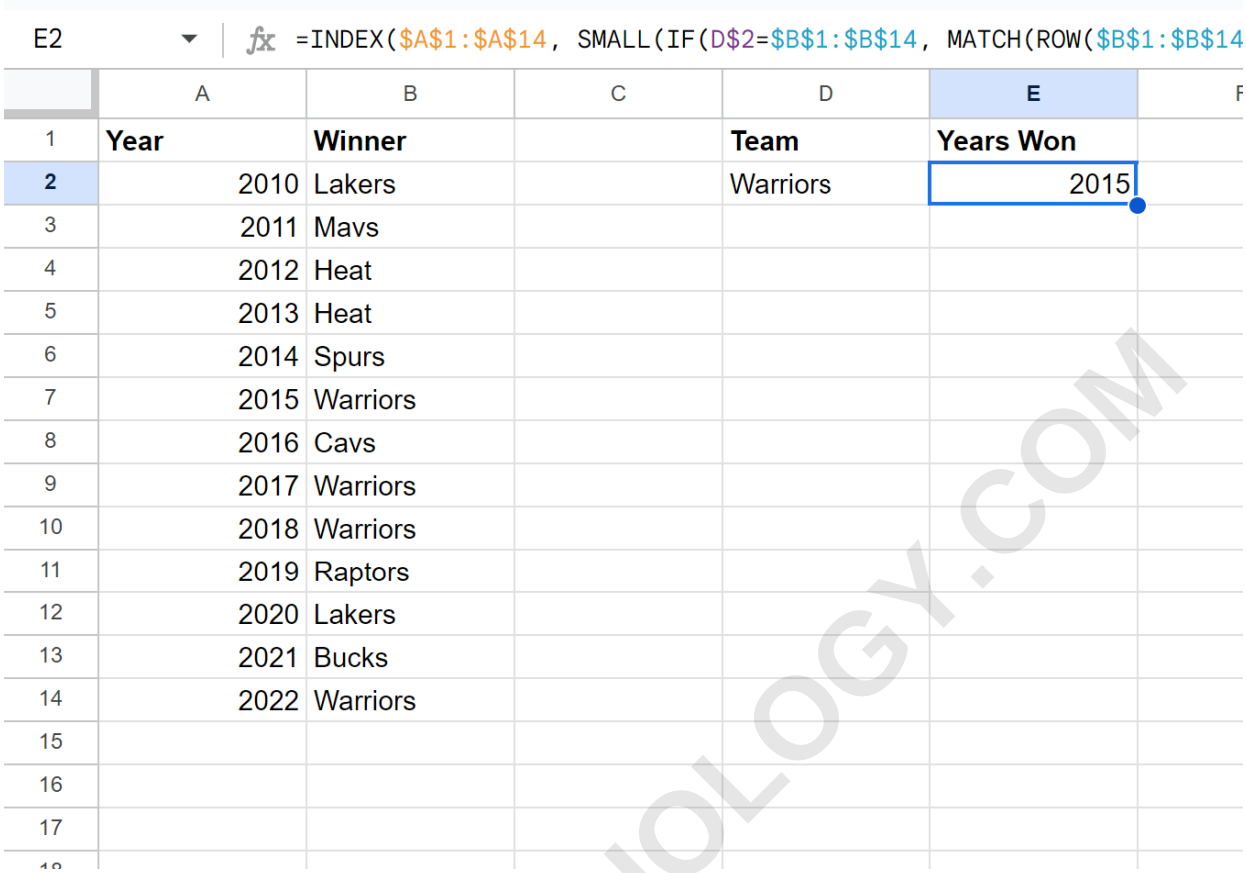
First, define the criteria: Type the team name, **Warriors**, into cell **D2**. This cell will serve as the reference point (`D$2`) for the conditional check within our formula.

Next, enter the complete formula into cell **E2**. Since the criteria range is `B1:B14` and the return range is `A1:A14`, the formula, when adapted to these specific ranges, looks exactly like the generalized formula introduced earlier:

=INDEX($A$1:$A$14, SMALL(IF(D$2=$B$1:$B$14, MATCH(ROW($B$1:$B$14), ROW($B$1:$B$14)), ""), ROWS($A$1:A1)))

Press **Enter**. Because the `ROWS($A$1:A1)` component is currently equal to 1, the formula executes the **SMALL** function to find the first (K=1) smallest row number where the criteria "Warriors" is found. The result will display the earliest year corresponding to a Warriors victory.

Upon pressing Enter, the first matching value, 2015, will be correctly displayed in cell E2, confirming that the formula structure is sound and has successfully identified the first match within the dataset.



	A	B	C	D	E
1	**Year**	**Winner**		**Team**	**Years Won**
2	2010	Lakers		Warriors	2015
3	2011	Mavs			
4	2012	Heat			
5	2013	Heat			
6	2014	Spurs			
7	2015	Warriors			
8	2016	Cavs			
9	2017	Warriors			
10	2018	Warriors			
11	2019	Raptors			
12	2020	Lakers			
13	2021	Bucks			
14	2022	Warriors			
15					
16					
17					

Iterating Results and Handling the Error Condition

The true utility of this array method emerges when we populate the formula across multiple cells. Since the formula is constructed to dynamically iterate through the sorted list of matching row numbers, dragging the formula down column E will sequentially reveal all subsequent years the Warriors won the championship.

To retrieve all matching values, simply select cell **E2** and drag the fill handle (the small square at the bottom right corner of the cell) downwards into the cells below (E3, E4, E5, etc.). As the formula moves down, the `ROWS($A$1:A1)` reference expands, increasing K, allowing **SMALL** to look for the second, third, and fourth smallest row indices, respectively.

The iteration continues until the **SMALL** function attempts to retrieve a K-th value that does not exist because all available matches have already been returned. At this point, the function will return the standard error value for a lack of numerical result: **#NUM!**. This error serves as a clear indicator that the list of matching results is complete.

E2:E6 =INDEX($A$1:$A$14, SMALL(IF(D$2=$B$1:$B$14, MATCH(ROW($B$1:$B$

	A	B	C	D	E
1	**Year**	**Winner**		**Team**	**Years Won**
2	2010	Lakers		Warriors	2015
3	2011	Mavs			2017
4	2012	Heat			2018
5	2013	Heat			2022
6	2014	Spurs			#NUM!
7	2015	Warriors			
8	2016	Cavs			
9	2017	Warriors			
10	2018	Warriors			
11	2019	Raptors			
12	2020	Lakers			
13	2021	Bucks			
14	2022	Warriors			
15					
16					
17					
18					

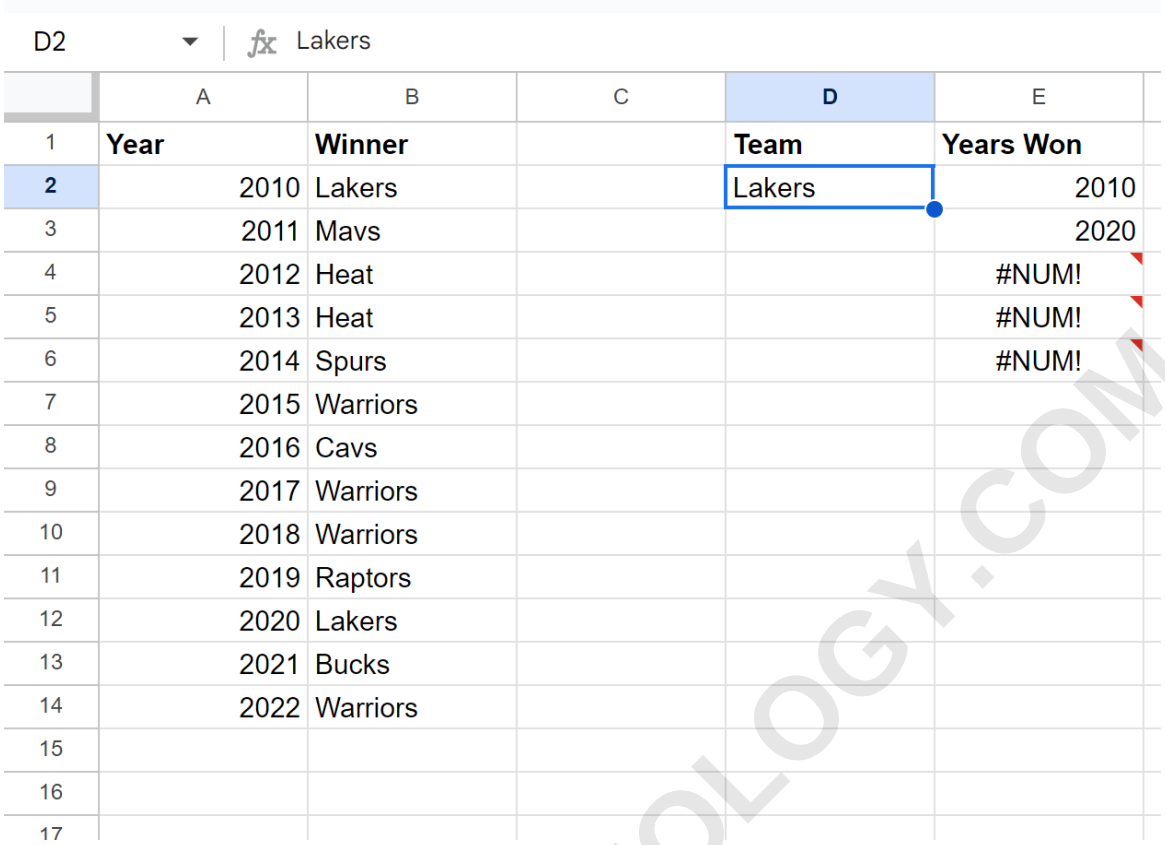
Based on the results generated by the array formula in column E, we can confirm that the Warriors won the finals during the following years:

2015
2017
2018
2022

Creating a Dynamic Lookup Criteria

One of the most powerful aspects of defining the criteria in a separate cell (D2) is the inherent dynamic nature of the entire structure. If the team name in cell **D2** is changed, the entire list of results in column E automatically recalculates and updates instantly to reflect the new criteria, without needing to modify the complex formula itself.

For instance, if we update cell **D2** from "Warriors" to "Lakers," the array formula immediately recalculates the row indices, the **SMALL** function retrieves the relevant rows for the Lakers, and the **INDEX** function pulls the correct years. The list of results in column E will instantly reflect this change, demonstrating the efficiency and versatility of this method for dynamic reporting and analysis.



D2 | Lakers

	A	B	C	D	E
1	**Year**	**Winner**		**Team**	**Years Won**
2	2010	Lakers		Lakers	2010
3	2011	Mavs			2020
4	2012	Heat			#NUM!
5	2013	Heat			#NUM!
6	2014	Spurs			#NUM!
7	2015	Warriors			
8	2016	Cavs			
9	2017	Warriors			
10	2018	Warriors			
11	2019	Raptors			
12	2020	Lakers			
13	2021	Bucks			
14	2022	Warriors			
15					
16					
17					

The updated list accurately reflects the years the Los Angeles Lakers won the championships according to our dataset:

2010
2020

Users are encouraged to experiment by changing the team name in cell **D2** to any team present in the criteria column (B1:B14) to observe the instant, dynamic updating of the results. This dynamic mechanism is particularly useful in dashboards or summary sheets where the criteria often changes based on user input or external dependencies.

Alternative and Modern Solutions: FILTER and QUERY Functions

While the nested **INDEX/SMALL/IF** formula provides a robust, universally compatible method for multi-value lookup, modern Google Sheets users often prefer simpler, dedicated functions. The two primary alternatives that achieve the exact same result--returning multiple values based on a single criterion--are the FILTER function and the QUERY function. These functions handle the array processing implicitly, leading to much cleaner and more readable syntax.

The FILTER function is perhaps the most direct replacement for the complex array formula. It takes

the range you want to return (Column A) and a condition (Column B equals D2). Unlike the INDEX/SMALL method which requires copying the formula down and generates errors when matches run out, the FILTER function is a spill-over function, meaning a single formula in cell E2 would automatically populate all results below it. For our NBA example, the equivalent formula would be: `=FILTER(A1:A14, B1:B14 = D2)`. This simplicity drastically reduces complexity and potential error points.

Similarly, the QUERY function offers unparalleled flexibility, especially when dealing with multiple criteria, sorting, or aggregation requirements. The QUERY function uses the Google Visualization API Query Language, which resembles SQL, allowing users to define exactly what data to select and what conditions to apply. For the same NBA example, the formula would be: `=QUERY(A1:B14, "SELECT A WHERE B = '" & D2 & "'", 0)`. While slightly more complex than FILTER, QUERY is the ultimate tool for highly sophisticated data extraction and transformation within Google Sheets.

Summary of Data Extraction Techniques

Retrieving multiple values based on a single criterion is a fundamental skill in advanced spreadsheet management. Whether utilizing the traditional, robust **INDEX/SMALL/IF Array Formula** or embracing modern solutions like **FILTER** and **QUERY**, the goal remains the same: efficient, dynamic, and accurate data extraction. The INDEX/SMALL approach is valuable for understanding the mechanics of array processing and for compatibility in environments where spill-over functions might be restricted. However, for most modern Google Sheets applications, **FILTER** or **QUERY** provide superior readability and ease of maintenance.

The choice of method depends heavily on the complexity of the criteria and the user's proficiency. For simple single-criteria lookups, **FILTER** is recommended due to its concise syntax. For situations requiring sorting, grouping, or highly complex conditional logic, the QUERY function is the industry standard. Regardless of the tool chosen, mastering the ability to retrieve entire sets of data based on dynamic criteria significantly enhances productivity and analytic capability within the spreadsheet environment.

Further Learning and Related Tutorials

To continue developing expertise in complex data manipulation within Google Sheets, consider exploring related tutorials that build upon the concepts introduced here. These advanced functions are often combined to solve intricate business problems, such as handling unique entries or performing conditional aggregation.

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