

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

Authored by
stats writer

January 25, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Return Multiple Values in Google Sheets Based on One Criteria*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=127549>

Google Sheets stands out as an exceptionally powerful and flexible application, providing users with robust capabilities to efficiently **organize and analyze data** within a highly accessible spreadsheet format. One of the most frequently encountered challenges in spreadsheet manipulation involves extracting specific information when a single entry may correspond to several associated results. Addressing this need, Google Sheets offers sophisticated methods to successfully return multiple values based on a designated single criteria. This functionality is essential for advanced filtering and precision retrieval of data from extensive datasets, allowing users to apply a specific condition to isolate relevant records swiftly.

The ability to retrieve associated records based on a common field far surpasses the limitations of simple lookup functions that typically only return the first match. Utilizing combined functions such as FILTER, VLOOKUP (when enhanced for array output), or the versatile QUERY function, spreadsheet users can customize their data extraction workflows. While these built-in functions offer streamlined solutions, mastering complex array formulas--specifically the combination of **INDEX** and **SMALL**--provides unmatched flexibility, particularly in scenarios where native array support is limited or specific ordering is required.

This capability is not merely a convenience; it forms the backbone of effective data analysis and informed decision-making. By accurately isolating all relevant data points corresponding to a singular condition, analysts can ensure comprehensive coverage and avoid overlooking crucial information. Whether you are tracking sales records linked to a specific region, managing inventory items tied to a single supplier, or compiling historical sports statistics, Google Sheets provides an efficient and reliable solution for retrieving complete sets of multiple values predicated upon a clear, single criteria.

Google Sheets: Return Multiple Values Based on Single Criteria

The Advanced Array Formula for Multiple Returns

While Google Sheets offers several straightforward methods for data retrieval, the most robust and widely applicable technique for extracting multiple matching results is through a specialized array formula combining the power of **INDEX**, **SMALL**, and **IF** functions. This combination allows the formula to systematically search the dataset, identify all row numbers that meet the specified criteria, and return the corresponding values one by one, effectively creating a dynamic list of results. This approach is particularly valuable because it ensures that even if the source data is rearranged, the integrity of the lookup mechanism remains sound, providing a reliable output structure regardless of the input order.

To implement this sophisticated lookup mechanism, we utilize the following structure. This specific formula is designed to function as an array formula, meaning it processes multiple values simultaneously rather than just a single cell reference. It isolates the required indices (row numbers) for every row where the criteria matches, sorts those indices, and then uses them sequentially to pull the corresponding data points from the desired result column. Understanding the components is crucial for successful deployment and modification across various datasets and business requirements.

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

```
=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)))
```

Breaking down this complex structure reveals its efficiency. The **IF** function creates an array of row numbers only where the condition ($D2=B1:B14$) is met. The **MATCH(ROW(...), ROW(...))** component generates a sequential index array for those matching rows based on their relative position. The **SMALL** function then iterates through this list of indices, picking the 1st smallest, 2nd smallest, and so on, using the dynamically increasing count provided by the **ROWS(\$A\$1:A1)** function as its iteration number. Finally, the **INDEX** function uses this sequential row number to pull the corresponding data point from the designated result range ($A1:A14$).

Deconstructing the Key Formula Components

To master this technique, it is beneficial to understand the role of each component within the nested function chain. The formula essentially operates by generating a filtered list of row positions that satisfy the criteria specified in cell **D2** against the lookup range **B1:B14**, and then retrieving the corresponding values from the return range **A1:A14** sequentially. This specific formula returns all of the values in the range **A1:A14** where the corresponding value in the range **B1:B14** is equal to the value in cell **D2**, thereby providing a complete list of results that satisfy the criteria.

The **IF(D2=B1:B14, ...)** section is the heart of the criteria match. It tests every cell in the criteria range (B1:B14) against the single criteria cell (D2). If the condition is true, it returns the row number's position in the dataset using the **MATCH(ROW())** sequence; otherwise, it returns a blank string (""). This conditional execution is vital, as it prevents non-matching rows from interfering with the sequential retrieval process. This operation generates a mixed array containing valid positions and ignored values, which is subsequently processed by the outer functions.

The **ROWS(\$A\$1:A1)** component is arguably the most clever part of this non-native array solution. When you drag the formula down from cell E2 to E3, E4, and so on, this range dynamically

expands (e.g., $\$A\$1:A2$, $\$A\$1:A3$). Since the first absolute reference ($\$A\1) is fixed and the second relative reference ($A1$) changes, the **ROWS()** function returns 1, 2, 3, and so forth. This count provides the 'k' value for the **SMALL** function, ensuring that the formula requests the 1st smallest match index, then the 2nd smallest, and continues until all multiple values have been retrieved. Without this dynamic incrementing counter, the formula would simply return the same result repeatedly.

Setting Up the Practical Example Dataset

To illustrate the application of this powerful array formula, we will use a practical example involving sports statistics. Suppose we have a historical dataset in Google Sheets detailing the winners of the NBA finals over various years. Our goal is to query this dataset to return a list of all years associated with a specific winning team, demonstrating how a single criteria (the team name) can yield multiple corresponding results (the years). This setup mimics many real-world scenarios where one field serves as a lookup key for historical or repetitive events.

This dataset spans cells A1 through B14, with Column A representing the "Year" (the values we wish to retrieve) and Column B representing the "Winning Team" (the criteria range). We will designate cell **D2** as the input cell where the user specifies the team name they are interested in. The results will be dynamically displayed starting in cell **E2** and filling downwards, creating a clean extracted list adjacent to the input criteria.

The following visual representation shows the initial structure of our sample data, which catalogs the champions for the NBA finals across different seasons. Note the clear distinction between the key fields used for the lookup operation: the return column (A) contains the year, and the criteria column (B) contains the team name.

	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				

For our initial test case, suppose we specifically want to isolate every year that the Golden State Warriors won the title. This requires the formula to search Column B for every instance of "Warriors" and return the corresponding year from Column A, regardless of the order in which they appear in the original data, thus proving the non-sequential retrieval capability of the formula.

Step-by-Step Implementation for Data Extraction

The successful implementation of this array formula requires precise entry and appropriate handling of its array nature. It is crucial to ensure that all range references within the formula are correctly set as absolute (using the dollar sign \$) where necessary, especially for the lookup ranges (A1:A14 and B1:B14) and the criteria cell (D2), while maintaining the necessary relative reference for the iteration counter (A1 within the ROWS function).

First, we must define our criteria. To isolate the years the Warriors won, we must accurately type the criteria "Warriors" into cell **D2**. This cell now acts as the central reference point (the single criteria) for our entire lookup operation, ensuring that the formula only pulls records that exactly match this text string. Case sensitivity may apply depending on Google Sheets settings, so precision is key here.

Next, we input the complete array formula into the first result cell, **E2**. The formula, as defined

earlier, is structured specifically to search the range **B1:B14** for the content of **D2**, and return corresponding values from **A1:A14**. The formula ensures that the calculation begins by identifying the smallest index corresponding to a match, which is achieved by setting the initial iteration count to 1.

=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)))

Once we press **Enter**, the formula executes. Because we are in cell E2, the **ROWS(\$A\$1:A1)** component resolves to 1, causing the **SMALL** function to pull the first (smallest) row index where the Warriors are listed. This yields the first year the Warriors won, confirming the successful initial setup of the lookup mechanism and providing the first matched entry in the output column.

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

	A	B	C	D	E	F
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						
18						

Completing the List and Handling Errors

After successfully retrieving the first match in cell E2, the subsequent step involves extending this formula vertically to retrieve all remaining multiple values. This is achieved by using the drag-and-fill handle located at the bottom right corner of cell E2 and dragging the formula down through column E. As the formula is dragged, the crucial **ROWS(\$A\$1:A1)** component increments its count

(to 2, 3, 4, etc.), sequentially instructing the **INDEX** function to fetch the second, third, fourth, and subsequent matching row indices derived from the filtered array.

This iterative process continues until the formula has exhausted all available matches in the dataset corresponding to the criteria in D2. Once all entries satisfying the criteria have been returned, the **SMALL** function attempts to retrieve a k-th value that does not exist in the filtered array of indices (because k exceeds the total number of matches). When this occurs, the formula returns a **#NUM!** error value. This error indicator serves a practical purpose, signifying that the complete list of results for the specified criteria is finished, marking the functional end of the returned data list.

The final output clearly lists all the years corresponding to the Warriors winning the **NBA finals**. We can observe the complete list of years extracted from the original data, followed immediately by the error marker. This visual arrangement confirms that the extraction process was successful and complete for the chosen criteria, providing accurate and segregated results.

E2:E6 fx =INDEX(\$A\$1:\$A\$14, SMALL(IF(D\$2=\$B\$1:\$B\$14, MATCH(ROW(\$B\$1:\$B\$14), \$B\$1:\$B\$14), 0)), ROW()-1)

	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 execution of the array formula, we have successfully filtered the dataset and determined that the Warriors won the finals during the following years:

2015

2017
2018
2022

Dynamically Updating Results with New Criteria

A significant advantage of using this formula structure is its complete dynamism and reliance on external cell referencing. The entire list of results is dependent solely on the content of the criteria cell, **D2**. If we change the team name in cell **D2**, the array formula recalculates instantly across all cells in column E, automatically updating the list of corresponding years. This instantaneous feedback loop makes the setup ideal for interactive dashboards or reporting tools where users frequently switch criteria without needing to adjust the underlying formulas.

For instance, let us update cell **D2** from "Warriors" to "Lakers". As soon as this change is registered, the **IF** component within the formula immediately evaluates the entire range **B1:B14** against the new criteria. The index array is regenerated to include only the rows where "Lakers" appears, and the **SMALL** and **INDEX** functions subsequently pull the newly identified years into column E, replacing the previous results entirely and ensuring data integrity.

The resulting data provides the complete list of championship years for the Lakers within the analyzed period. This illustrates the flexibility and power of using complex nested functions to create highly reactive and powerful data visualization tools within Google Sheets, requiring maintenance only at the source data level.

	A	B	C	D	E
D2				Lakers	
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					

We can clearly see that the Lakers won the finals during the following years based on the new criteria:

2010

2020

This dynamic capability ensures that the data retrieval process is scalable and reusable. Users are free to change the team name in cell **D2** to any team name listed in the dataset, and the list of associated winning years will automatically reflect the change without requiring any modification to the formula itself in column E. Feel free to explore other entries in your dataset by simply modifying the criteria cell.

Alternative Methods and Further Learning

While the INDEX/SMALL/IF array formula is highly reliable and offers granular control, it is important to acknowledge that Google Sheets provides other powerful methods that can achieve the same goal of returning multiple values based on a single criteria, often with simpler syntax for those familiar with database querying. Specifically, the QUERY function offers a SQL-like interface that can filter and return entire columns based on conditions, typically yielding the cleanest solution when error handling is not complex, using commands like `=QUERY(A1:B14, "SELECT A WHERE B`

```
= ' "&D2&" ' ) ).
```

The [FILTER](#) function is another excellent built-in tool, especially if you are working with modern versions of Google Sheets that natively support array output. Using `=FILTER(A1:A14, B1:B14=D2)` is often the most straightforward solution, returning all matching years instantly without the need for the complex INDEX/SMALL iteration. However, the INDEX/SMALL method remains essential when precise formatting, zero suppression, or compatibility with older spreadsheet standards is required, offering flexibility in controlling the output structure.

Mastery of these advanced techniques allows users to move beyond basic data manipulation and create highly efficient, automated reporting tools. The following tutorials explain how to perform other common and advanced data tasks in Google Sheets, enabling deeper expertise in spreadsheet manipulation:

[Exploring Advanced Conditional Formatting Rules](#)

[Implementing Dynamic Dropdown Lists with Data Validation](#)

[Utilizing Google Apps Script for Custom Functions](#)

[Generating Pivot Tables for Summarizing Large Datasets](#)