

How to Easily Return Multiple Values Based on Single Criteria in Excel

Authored by
stats writer

November 28, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Return Multiple Values Based on Single Criteria in Excel*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=100899>

The Challenge of Retrieving Multiple Values in Excel

Microsoft Excel is an immensely powerful tool for data analysis and management, but users often face complexity when attempting to retrieve multiple corresponding values based on a single lookup criterion. Standard lookup functions like VLOOKUP or the basic INDEX/MATCH combination are fundamentally designed to return only the first match they encounter. When dealing with large datasets containing repeated categories--such as sales records, inventory lists, or historical sports scores--a more advanced solution is required to extract every instance that satisfies the condition. This limitation necessitates the use of complex array logic, often combining functions like INDEX, SMALL, IF, and MATCH, to process the entire range conditionally and generate a sequential list of all matches.

This sophisticated method transforms the traditional one-to-one lookup into a one-to-many extraction process. By leveraging the power of an Array Formula, we can instruct Excel to evaluate every cell in the lookup range against our specified criterion. Instead of stopping at the first match, the formula internally generates an array of row numbers corresponding to all qualifying entries. Subsequently, these row numbers are sequentially fed into the main retrieval function, allowing us to populate a new column with all relevant results. Understanding the underlying mechanism of this technique is crucial for mastering dynamic data extraction in complex spreadsheets.

The Advanced Array Formula for Conditional Retrieval

To successfully return multiple values based on a single criterion in Excel, we employ a specific, powerful formula structure. This structure must be entered as an Array Formula (by pressing **CTRL+SHIFT+ENTER** upon entry in older versions of Excel, or simply **ENTER** in newer dynamic array versions). The core concept relies on isolating the row numbers of matches and then using the SMALL function to pull these row numbers in ascending order.

You can utilize the following foundational formula structure to achieve this dynamic extraction. This template assumes the data you wish to retrieve resides in the first range (the return column), and the criteria range is specified in the second (the lookup column):

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

This formula is designed to scan the lookup range, which in this instance is specified as **\$B\$1:\$B\$14**, and compare every value against the criterion located in cell **E\$1**. For every match found, it returns the corresponding value from the retrieval range, **\$A\$1:\$A\$14**. The critical element is the dynamic component provided by the ROWS function, which ensures that as the formula is dragged down, it iteratively searches for the first, second, third, and subsequent

matches, providing a complete list.

Deconstructing the Core Formula: INDEX and SMALL

To truly appreciate the power of this technique, it is essential to break down the roles played by the individual functions. The formula is fundamentally built around the INDEX function, which serves as the final delivery mechanism. The syntax of INDEX is simple: `INDEX(array, row_num,)`. In our case, the array is **\$A\$1:\$A\$14** (the column containing the values we want to return), and the crucial task is calculating the correct `row_num` for each sequential match.

This calculation is handled entirely by the inner nested functions. The SMALL function, `SMALL(array, k)`, is used to return the k-th smallest value from a given array. Within our complex array, SMALL is instructed to iterate through the list of row numbers that the IF function generates. When the formula is first entered into the target area, it looks for the 1st smallest row number (where k=1). When copied down one cell, the internal counter adjusts, and it seeks the 2nd smallest row number (k=2), and so on. This sequential pulling of row indices is precisely what allows us to extract all matches one by one, ensuring no valid match is skipped.

Generating Row Position Indices with IF and MATCH

The core conditional logic resides in the IF statement: `IF(E$1=$B$1:$B$14, MATCH(ROW($B$1:$B$14), ROW($B$1:$B$14)), "")`. This segment performs the necessary comparison and index generation crucial for the array's operation. The condition `E$1=$B$1:$B$14` evaluates the entire lookup range **\$B\$1:\$B\$14** against the criterion in **E\$1**, resulting in a temporary internal array composed entirely of TRUE and FALSE values, marking which rows satisfy the condition.

Crucially, where the condition is TRUE (meaning a match is found), the formula executes the TRUE argument: `MATCH(ROW(B1:B14), ROW(B1:B14))`. This seemingly redundant MATCH structure is a highly effective way within an Array Formula to generate a sequence of relative row numbers (1, 2, 3, 4, etc.) corresponding to the position of the match within the original range. For instance, if the criterion matches rows 3 and 7, the IF statement returns an array containing 3, 7, and empty strings "" for the non-matching rows. If the condition is FALSE, it explicitly returns an empty string "". This array of position indices (the relative row numbers of the matches) becomes the input for the SMALL function, which then handles the ordering and sequential selection process.

Controlling Iteration: The Role of the ROWS Function

To ensure the SMALL function retrieves the generated row indices sequentially (1st, 2nd, 3rd,

etc.), we utilize the dynamic range calculation provided by the `ROWS(A1:A1)` argument. This segment is highly critical for iterative extraction and is the key to expanding the results down a column.

The ROWS function calculates the number of rows in a specified range. Notice the mixed referencing used: **\$A\$1** is absolutely fixed, while **A1** is relative. When the formula is placed in the first output cell (E2) and copied down, this range expands:

In cell E2, `ROWS(A1:A1)` returns **1**. SMALL looks for the 1st smallest index.

When dragged down to cell E3, the range expands to `ROWS(A1:A2)`, returning **2**. SMALL looks for the 2nd smallest index.

When dragged down to cell E4, the range expands to `ROWS(A1:A3)`, returning **3**. SMALL looks for the 3rd smallest index.

This dynamic range calculation provides the necessary counter, *k*, for the SMALL function, guaranteeing that every matched value is retrieved exactly once and in the correct order, row by row. This ensures the output is a clean, consecutive list of results corresponding to the lookup criteria in cell **E1**.

Step-by-Step Implementation Example

Let us walk through a practical application using a sample dataset detailing the NBA finals winners over several years. Our goal is to dynamically extract all the years a specific team won, based on a single criterion cell.

Suppose we have the following data organized where Column A lists the Year (our return range) and Column B lists the Winner (our lookup range). We want to input the team name in cell **E1** and have the winning years populate automatically starting in cell **E2**.

	A	B	C	D	E	F
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						
18						
19						
20						
21						

We begin by setting the criterion in cell **E1**--for this initial example, we will search for "Warriors." Next, we type the comprehensive array formula into cell **E2**. It is essential to ensure that absolute and relative references are correctly maintained, particularly the fixed references for the data arrays and the mixed reference for the ROWS function, as demonstrated below:

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

Once the formula is entered, remember the necessary steps for array functions in older versions of Excel: pressing **CTRL+SHIFT+ENTER**. If entered correctly, Excel will automatically enclose the formula in curly braces {}. Upon successful entry, the first year the Warriors won will be displayed in cell E2:

	A	B	C	D	E	F	G	H
1	Year	Winner		Team	Warriors			
2	2010	Lakers			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								
19								
20								
21								

Finalizing the Extraction and Error Management

With the formula successfully calculating the first match, the next step is to extend this extraction down the column to reveal all remaining matches. This is accomplished by using the fill handle to drag the formula down through subsequent cells in column E. Because the internal ROWS counter increases with each row, the formula sequentially retrieves the 2nd, 3rd, 4th, and subsequent matching row indices until all matches are exhausted.

When the formula runs out of available matches for the specified criterion (e.g., if the Warriors only won four times within the dataset), the SMALL function will attempt to find a 5th smallest index from an array that contains only four valid row numbers and many empty strings "". Since SMALL cannot rank an empty string numerically, it returns the standard **#NUM!** error, which signals that the extraction process is complete.

As shown below, dragging the formula down displays the years until the list is exhausted:

	A	B	C	D	E	F
1	Year	Winner		Team	Warriors	
2	2010	Lakers			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						
19						
20						
21						
22						

By implementing this technique, we can clearly see that the Warriors secured the finals during the following years within our data range:

2015
2017
2018
2022

Dynamic Updating Based on Criteria Changes

One of the most valuable features of this Array Formula solution is its dynamic nature. Since the criterion is linked to cell **E1**, changing the team name in that cell automatically triggers a full recalculation of the entire array, instantly updating the list of winning years in column E.

For instance, suppose we update cell **E1** from "Warriors" to "Lakers." The underlying IF function immediately re-evaluates the **\$B\$1:\$B\$14** range, generating a new array of row indices

corresponding only to the "Lakers" entries. The formula then pulls the years associated with those new indices.

	A	B	C	D	E	F
1	Year	Winner		Team	Lakers	
2	2010	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						
18						
19						
20						

The list in column E instantly reflects the new criteria, showcasing the Lakers' victories:

2010

2020

Conclusion and Modern Alternatives

While the INDEX/SMALL/IF/MATCH array formula remains a robust and universally applicable solution for retrieving multiple matches in all versions of Excel, it is admittedly complex to construct and debug. It requires precise attention to absolute and relative referencing, and the mandatory use of **CTRL+SHIFT+ENTER** in older spreadsheet environments.

For users operating on modern versions of Office 365 or Excel 2021 and later, a significantly simpler alternative exists: the **FILTER** function. This is a native dynamic array function that can perform this exact task with far less complexity and greater efficiency. The equivalent formula using FILTER would be concise, such as `=FILTER(A1:A14, B1:B14=E1)`. This function

automatically spills the results into the adjacent cells, eliminating the need for iteration via the ROWS function and manual error handling (which would typically involve wrapping the complex array in an IFERROR statement). Regardless of the version of Excel you use, mastering conditional data extraction is a foundational skill for advanced spreadsheet management, allowing you to build dynamic reports that instantly adapt to changing criteria.

ARABPSYCHOLOGY.COM