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

Authored by

Mohammed looti

October 28, 2025

The Challenge of Multiple Returns in Data Lookups

In the realm of Microsoft Excel, the ability to efficiently retrieve data based on specific criteria is paramount for effective analysis and reporting. Standard lookup functions, such as the ubiquitous VLOOKUP or the more flexible INDEX/MATCH combination, are powerful tools designed primarily to return the **first** corresponding value they encounter. However, data often contains duplicate entries matching the condition, leading to a critical and frequent challenge: how do you extract **all multiple values** that satisfy a single input condition? Addressing this limitation requires moving beyond simple lookups and implementing a highly advanced, nested array formula.

This comprehensive guide introduces a robust and reliable formula capable of handling this specific data retrieval challenge. The methodology presented here is designed to sequentially pull all relevant entries from a designated data range whenever they match the user-defined criterion. Crucially, this technique is highly versatile and maintains compatibility across virtually all versions of Excel, making it an indispensable asset for detailed data analysis and complex reporting requirements, especially in environments where newer dynamic array functions are unavailable.

The foundation of this solution rests upon a meticulously crafted array formula structure, which strategically combines several core Excel functions to perform a conditional lookup and systematically retrieve every single matching result. Specifically, this formula is tailored to return values from a specified results column (e.g., **A1:A14**) based on a corresponding value in a lookup column (e.g., **B1:B14**) matching the user's input, which is housed in a single designated cell (e.g., **E1**). Understanding the syntax and logic of this formula is key to unlocking advanced data manipulation capabilities in Excel.

=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)))

Deconstructing the Multi-Conditional Array Formula

To fully grasp the sophisticated mechanics of this data retrieval method, it is essential to dissect the individual components of the formula and analyze their collaborative interaction. This complex formula operates as a miniature program within Excel, orchestrating several key functions to systematically identify and return multiple matching values. We will explore each nested element below, starting from the outermost wrapper and moving inward to the conditional core.

The overall structure relies on the principle of generating an array of row numbers corresponding only to the matching data points, and then using an iterative function to pull those row numbers sequentially. This approach bypasses the limitations of traditional lookup methods by treating the entire range simultaneously, which is the defining characteristic of an array formula.

INDEX($A$1:$A$14, ...): The Retrieval Wrapper. The INDEX function serves as the final output mechanism. It retrieves a value from the designated range, **$A$1:$A$14**, based on a calculated row number. In our example, this is the array containing the desired results (e.g., the Years). The use of absolute references here is critical, ensuring the range remains constant when the formula is copied down the column. The remainder of the formula's structure is dedicated solely to calculating the necessary row position for each successful match.

SMALL(...): The Iterative Mechanism. The SMALL function is indispensable for sequential retrieval. It functions by returning the k-th smallest value from a given set. In this context, it extracts the relative row numbers of the matching data points one by one. As the formula is dragged vertically, the 'k' argument (supplied by the ROWS function) increments (1st, 2nd, 3rd, etc.), allowing SMALL to pull the next matching row index in the sequence.

The Conditional Core: IF, MATCH, and ROW Functions

The true intelligence of the array formula resides within its conditional core, the nested combination of the IF, MATCH, and ROW functions. This structure is responsible for filtering the data and generating the list of row numbers that correspond only to successful matches.

IF(E$1=$B$1:$B$14, MATCH(ROW(...)), ""): The Filter. This segment conducts a logical test across the entire dataset. When the condition is TRUE (a match is found), it returns the relative row number; otherwise, it returns an empty string (`""`).

E$1=$B$1:$B$14: This performs the logical comparison, generating an array of TRUE/FALSE values by comparing the lookup column **$B$1:$B$14** against the single criterion cell **E$1**. The mixed reference **E$1** ensures the row index of the criterion cell remains constant.

MATCH(ROW($B$1:$B$14), ROW($B$1:$B$14)): This intricate combination is a standard array formula technique used to generate a sequential numerical index corresponding to the rows within the range. The ROW($B$1:$B$14) function first returns the absolute row numbers (e.g., {1; 2; ...; 14}). When MATCH searches for these numbers within themselves, the result is the relative index {1; 2; ...; 14}. This index is returned by the IF function only if the condition is TRUE.

"": If the conditional test fails (FALSE), the IF function returns an empty string. This is crucial because the SMALL function ignores text values, effectively preventing errors from non-matching rows.

ROWS($A$1:A1): The 'k' Index Generator. This dynamic component provides the required argument for the SMALL function. When initially entered into cell E2, **ROWS($A$1:A1)** evaluates to 1. When the formula is dragged down, the range expands (e.g., $A$1:A2, $A$1:A3), causing the output to increment (2, 3, etc.). This ensures that we sequentially request the 1st, 2nd, 3rd, and

subsequent row numbers generated by the filtered array of matches.

Practical Application: Extracting Sequential Data

To demonstrate the practical utility of this powerful formula, we will apply it to a real-world scenario: filtering historical sports data. Imagine a dataset tracking the winners of the NBA Finals across several seasons. Our objective is to dynamically retrieve all the years associated with a specific winning team name.

The sample dataset is structured across two primary columns: Column A contains the "Year" (the data we wish to retrieve), and Column B lists the "Winner" (the lookup column). The user will interact with the worksheet by inputting their desired team name into a designated criteria cell, which we will set as **E1**. For our initial walkthrough, we will search for the years the "Warriors" secured the championship title.

The visualization below provides a clear representation of our data layout, showing the source data in columns A and B, and the starting point for our dynamic results in column E.

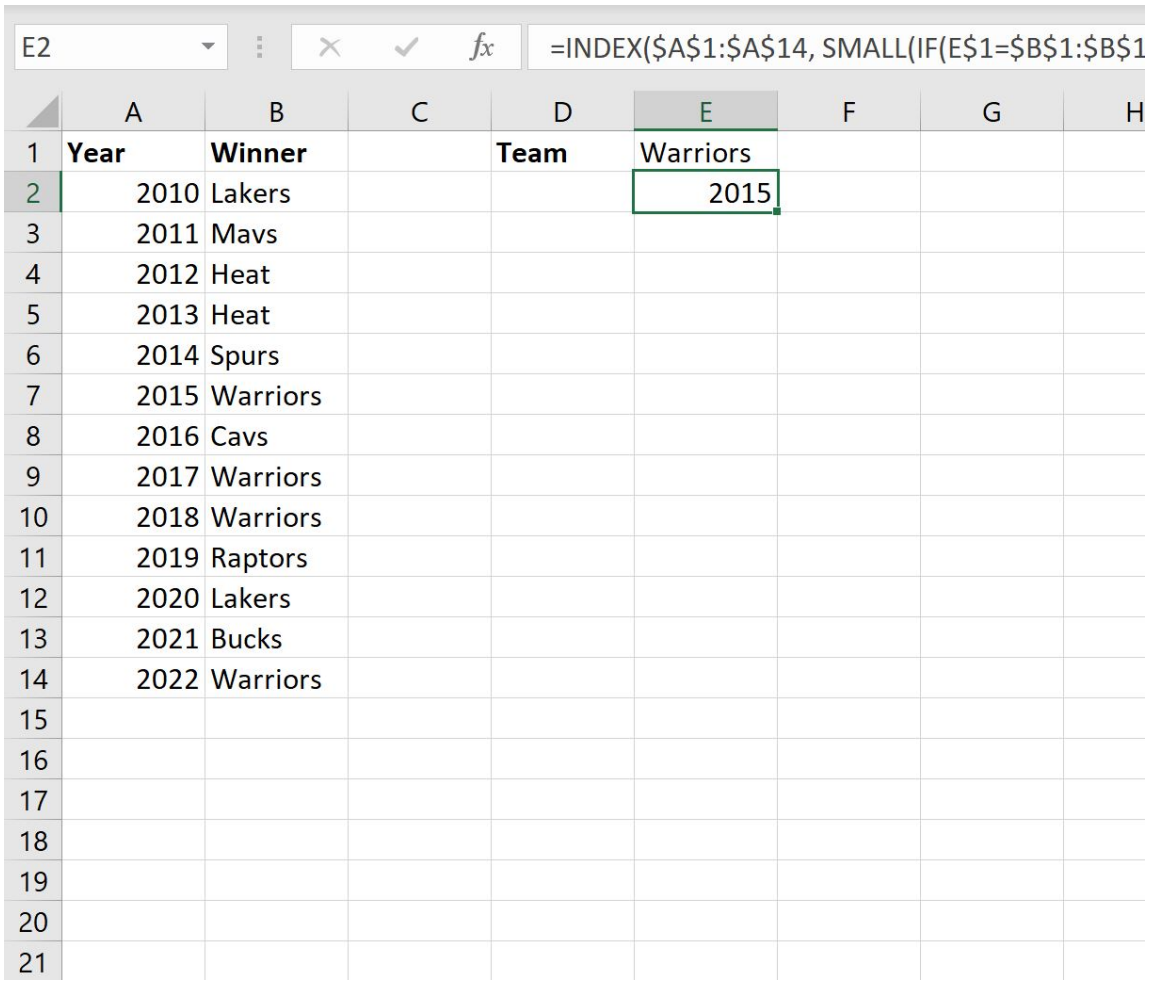
	A	B	C	D	E	F
1	**Year**	**Winner**				
2	2010	Lakers				
3	2011	Mavs				
4	2012	Heat				
5	2013	Heat				
6	2014	Spurs				
7	2015	Warriors				
8	2016	Cavs				
9	2017	Warriors				
10	2018	Warriors				
11	2019	Raptors				
12	2020	Lakers				
13	2021	Bucks				
14	2022	Warriors				
15						
16						
17						
18						
19						
20						
21						

Implementing and Managing the Array Formula Output

Once the data is prepared and the criterion is entered into cell **E1**, the implementation phase begins. The full array formula must be meticulously entered into the first result cell, **E2**.

=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)))

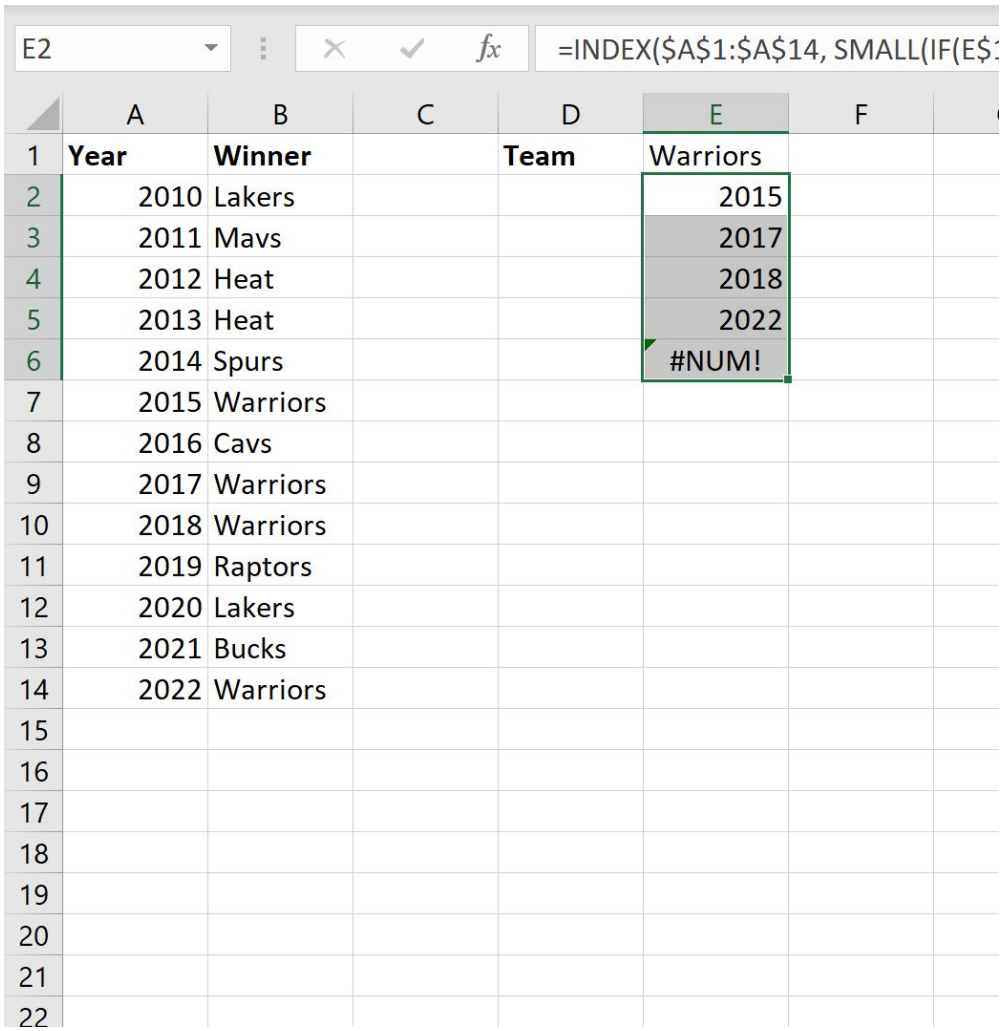
It is crucial to remember that this is an array formula. Depending on your version of Excel (pre-Excel 365), you must confirm the entry by pressing **Ctrl + Shift + Enter** simultaneously, which tells Excel to treat it as an array operation and wraps the formula in curly braces ({}). If you are using a modern version of Excel (365), simply pressing **Enter** is usually sufficient, as it handles dynamic arrays natively. After successful entry, cell **E2** will immediately display the first year in which the specified team won the championship.



	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								

To retrieve the complete list of matching years, the next step is to extend the formula. Use the fill handle--the small square located at the bottom-right corner of cell **E2**--and drag it down the

column. Continue this process until the formula runs out of matches, at which point you will encounter the #NUM! error. This error is expected and serves as a natural terminator for the results, signaling that the SMALL function can no longer find a corresponding 'k-th' smallest row number.



Based on the results for the "Warriors," the extracted winning years are clearly displayed:

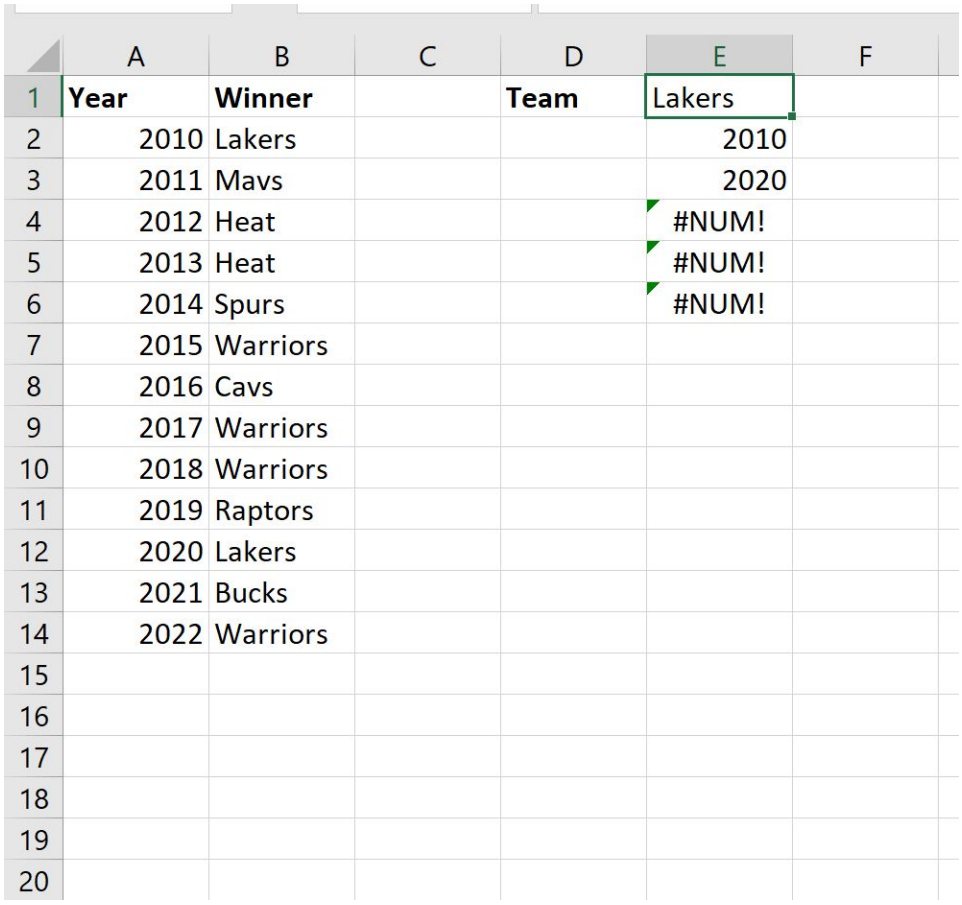
2015
2017
2018
2022

Enhancing User Experience Through Dynamic Filtering

A significant advantage of employing this complex formula structure is its inherent dynamic nature. The list of results automatically recalculates and updates instantaneously whenever the input

criterion in cell **E1** is modified. This feature transforms a static spreadsheet into a powerful, interactive analytical tool, eliminating the need for manual filtering or formula adjustments every time a new search is required.

For example, if we decide to shift our focus from the "Warriors" to the "Lakers," we simply change the text in cell **E1**. Upon this modification, the entire output range in column E will automatically refresh, displaying only the winning years relevant to the newly specified team. This flexibility greatly streamlines the process of exploring different subsets of data within a single worksheet environment.



	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						

As evidenced by the refreshed results shown above, the Lakers won the NBA Finals during these years:

2010
2020

While the array formula works effectively, users may wish to suppress the visible #NUM! error that appears when no more matches are found. This can be achieved by wrapping the entire structure within an IFERROR function: `=IFERROR( , "")`. This refined approach substitutes the error

message with a clean, empty cell, improving the professional appearance of the worksheet.

Modern Alternatives and Performance Considerations

Although the INDEX/SMALL/IF array formula provides a highly compatible and robust solution, it is important to acknowledge certain performance considerations. For workbooks containing extremely large datasets, traditional array formulas can be resource-intensive, potentially leading to slower recalculation times. Users working with newer versions of Excel, particularly those with an Office 365 subscription, have access to more efficient, modern dynamic array functions.

The most straightforward modern alternative is the FILTER function, which streamlines the multi-criteria lookup process dramatically. For users seeking robust, repeatable data transformation workflows, tools like Power Query (Get & Transform Data) offer superior performance and flexibility for importing, shaping, and filtering large volumes of data. Furthermore, the built-in Advanced Filter feature in Excel remains a viable, non-formula-based method for data extraction.

Nonetheless, mastering the classic INDEX/SMALL/IF technique remains an essential skill. Its high compatibility ensures that solutions built using this method will function reliably across diverse organizational environments, regardless of the Excel version being used. It provides a timeless and versatile solution for complex data retrieval challenges where dynamic array functions are not available.

Further Resources for Excel Mastery

Expand your Excel proficiency with additional tutorials that explain how to perform other common tasks and advanced data manipulations:

How to Use VLOOKUP with Multiple Criteria in Excel
How to Use Dynamic Array Formulas for Sorting in Excel
How to SUM if Multiple Conditions are Met in Excel
How to Remove Duplicate Values in Excel