How to Retrieve Multiple Results with INDEX MATCH in Google Sheets

Authored by

stats writer

January 24, 2026

Google Sheets: Use INDEX MATCH to Return Multiple Results

The standard lookup capabilities in spreadsheet applications, such as the basic VLOOKUP or a simple implementation of INDEX MATCH, are fundamentally designed to halt execution and return the very first corresponding value they find. While this behavior is suitable for unique identifiers, it presents a significant obstacle when managing data where a single criterion--such as a team name or a category--is linked to multiple records. This common scenario demands a more robust solution that can dynamically retrieve an entire list of matches rather than just the first one.

To overcome the limitations of standard lookups in Google Sheets, we must integrate the core lookup functions (INDEX and MATCH) with array processing functions like SMALL and conditional logic. This combination generates a complex array formula capable of generating a list of all row numbers that meet the search criteria, ensuring that every result is systematically extracted. This method is crucial for advanced data analysis and accurate reporting, especially when working with large, unsummarized datasets.

The Challenge of Retrieving Multiple Results in Data Management

In many business and analytical contexts, data often contains one-to-many relationships. For instance, an employee ID is unique (one-to-one), but a department name is tied to multiple employees (one-to-many). Traditional lookup functions like VLOOKUP or standard INDEX MATCH are single-result tools; once the condition is met for the first time, the process stops. Implementing a solution that handles multiple returns requires constructing a formula that transforms the standard lookup logic into an iterative array operation.

This array approach forces the formula to evaluate every row individually against the specified criteria, collecting the relative position of every match. By dynamically generating these positions, the formula can then sequentially feed them back into the INDEX function, which retrieves the corresponding output value. This level of control is fundamental for creating dynamic dashboards or reports where the output list must automatically update based on a single search key.

Understanding the Complex INDEX MATCH Array Formula

The formula required to perform this multi-result lookup is significantly more intricate than a standard lookup, leveraging the power of conditional arrays to handle the iteration process. This method requires specific function combinations to calculate the exact row number for each matching instance. Understanding this syntax is key to customizing it for various datasets and lookup requirements.

You can use the following syntax with INDEX MATCH in Google Sheets to return multiple results:

=IFERROR(INDEX($B$2:$B$11,SMALL(IF($D$2=$A$2:$A$11,ROW($A$2:$A$11)-ROW($A$2)+1),ROW(1:1))),"")

This particular formula is an elegant solution. It is designed to return all of the values located within the result range (designated as **B2:B11**) only when the corresponding value in the search range (**A2:A11**) is precisely equal to the defined criterion found in cell **D2**. Crucially, the function is wrapped in IFERROR to suppress calculation errors once all matches have been found, returning a clean blank space ("") instead of a disruptive `#NUM!` or `#N/A` error.

Deconstructing the Core Components of the Formula

To truly master this technique, one must analyze the individual roles of the nested functions, particularly how the SMALL and ROW functions collaborate to generate sequential indices. This complex segment is responsible for calculating and ordering the relative positions of all matching entries.

`IF($D$2=$A$2:$A$11, ROW($A$2:$A$11)-ROW($A$2)+1)`: This is the conditional array generator. It checks every cell in the lookup range (`$A$2:$A$11`) against the lookup value (`$D$2`). If a match is found, it calculates the relative row number within the array starting from 1 (`ROW($A$2:$A$11)-ROW($A$2)+1`). If no match is found, it returns `FALSE`. The result is an array containing only the relative row numbers of the matches (e.g., `{1; FALSE; 3; 4; FALSE; ...}`).
`SMALL(..., ROW(1:1))`: The SMALL function takes the array generated by the `IF` statement. The `ROW(1:1)` component dynamically serves as the 'k' argument for SMALL, telling it to retrieve the k-th smallest value from the array. When this formula is dragged down one row, `ROW(1:1)` becomes `ROW(2:2)`, which returns 2, thus asking for the second smallest match's row index. This sequential retrieval is the engine that drives the multiple results functionality.
`INDEX($B$2:$B$11, ...)`: Finally, the INDEX function uses the sequential relative row number provided by SMALL to pull the actual player name from the results range (`$B$2:$B$11`).

Step-by-Step Example Implementation

The following example shows how to utilize this powerful array formula structure in a practical scenario involving athlete data. We will use a simple dataset showing teams and corresponding player names to demonstrate the extraction of multiple player names associated with a single team.

Suppose we have the following dataset in Google Sheets that displays the team affiliation and player name for various basketball players:

	A	B	C
1	**Team**	**Player**	
2	Mavs	Dan	
3	Mavs	Mike	
4	Warriors	Steven	
5	Heat	Carl	
6	Heat	Jake	
7	Mavs	Brad	
8	Warriors	Tyler	
9	Kings	AJ	
10	Mavs	Spencer	
11	Heat	Dawkins	
12			
13			
14			
15			

Our objective is to dynamically retrieve the names of every player associated with a specific team, starting with the "Mavs" team, and list them in a separate output area. This requires setting up the criteria cell and then implementing the formula designed for array processing.

To initiate the lookup, we must first specify our search criterion. We will type the team name, "Mavs", into cell **D2**. Next, we will enter the multi-result formula into the first output cell, **E2**. All range references must be absolute (using the `$` sign) to ensure they do not shift when the formula is dragged down, while the `ROW(1:1)` component must remain relative to increment correctly.

Type the following formula precisely into cell **E2**:

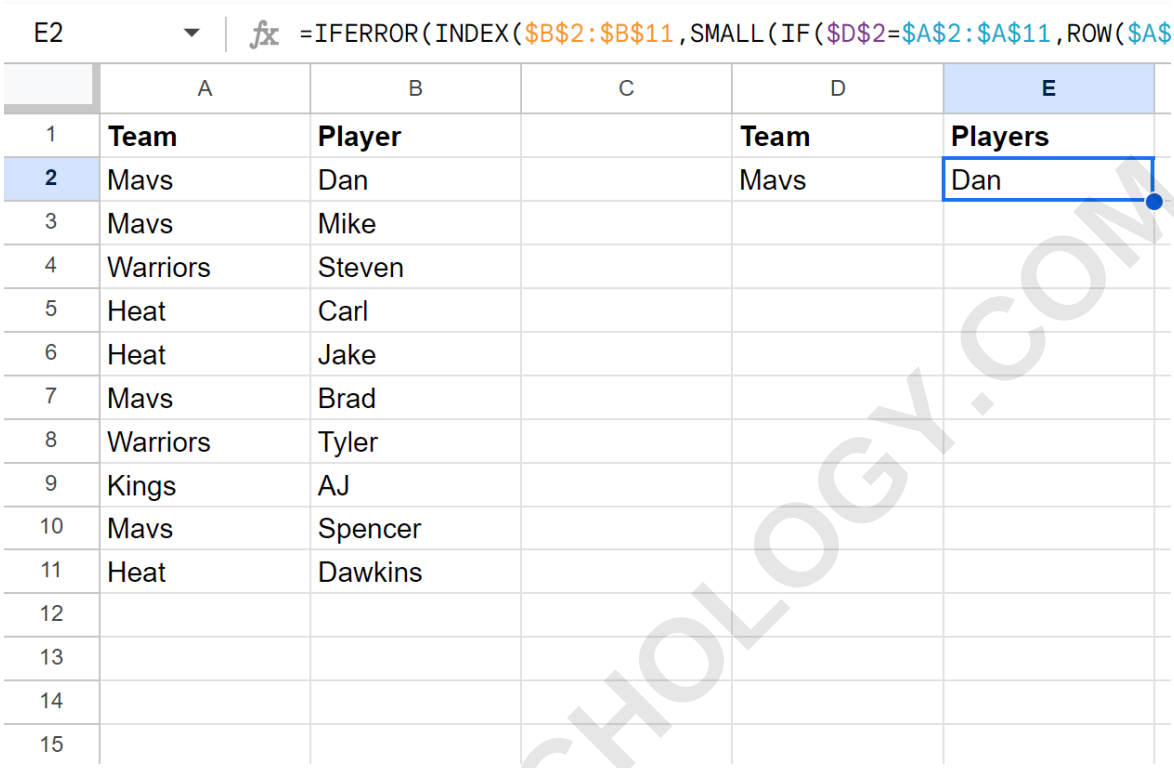
=IFERROR(INDEX($B$2:$B$11,SMALL(IF($D$2=$A$2:$A$11,ROW($A$2:$A$11)-ROW($A$2)+1),ROW(1:1))),"")

It is essential to understand that this formula often needs to be entered as an array formula in older spreadsheet software (using `Ctrl+Shift+Enter`). However, modern versions of Google Sheets typically handle this implicit array calculation automatically, meaning a simple **Enter** key press is usually sufficient.

Analyzing the Results and Dynamic Behavior

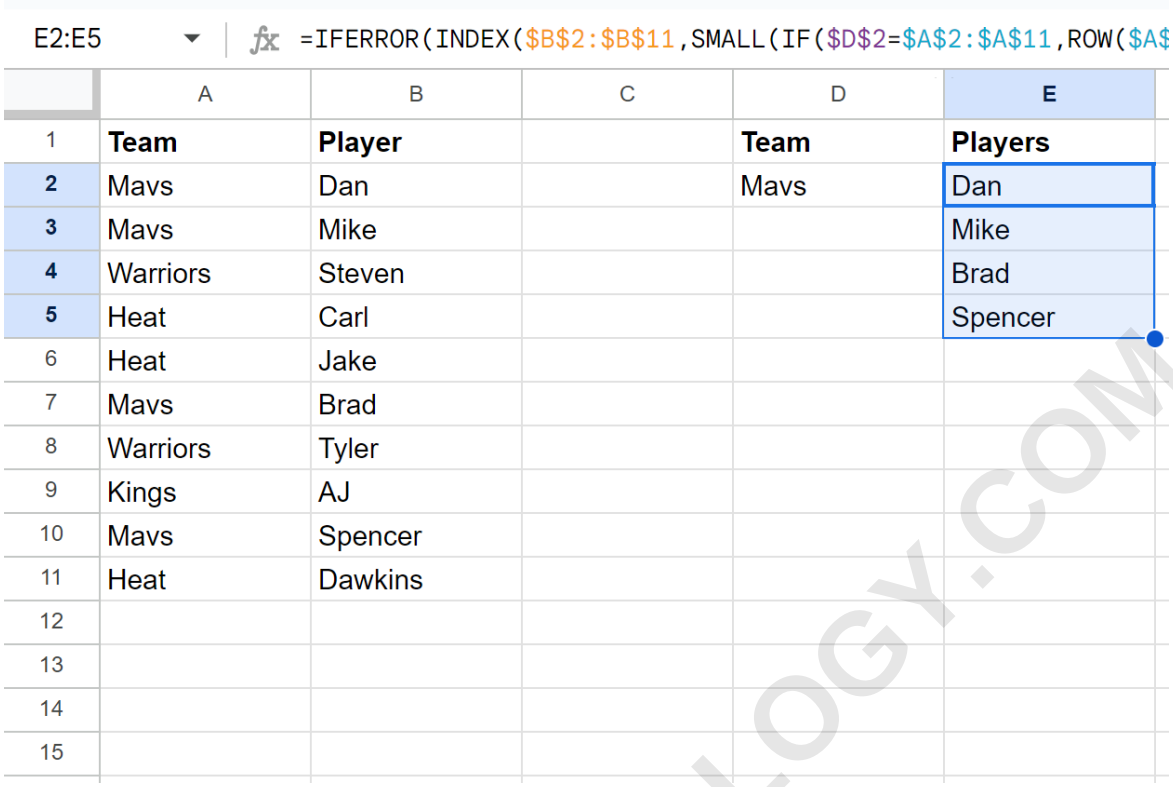
Once we press **Enter** in cell E2, the formula executes its array calculation. It identifies the first

instance where the team in Column A matches "Mavs", calculates its relative row position (which is 1, corresponding to A2), and the INDEX function retrieves the value from that position in Column B. The name of the first player on the Mavs team will be returned:



This single result is only the beginning. The real power of the formula is revealed when it is extended. The critical step is to apply the formula to the remaining cells in the output column (Column E). By clicking and dragging the fill handle (the small square at the bottom-right of cell E2) down to cell E11, the formula is copied, and the dynamic reference `ROW(1:1)` is systematically updated.

As the formula moves down, `ROW(1:1)` increments to `ROW(2:2)`, `ROW(3:3)`, and so on. This forces the SMALL function to retrieve the second smallest matching row index, then the third, and so forth. This sequential indexing results in the display of all players on the Mavs team, one by one:



Notice that the names of each of the four players on the Mavs team are now shown sequentially. The formula continues to search until all matching indices are exhausted. Once the SMALL function attempts to find a 5th, 6th, or subsequent smallest index (when only four exist), it will return an error (a `#NUM!` error). This is where the outer IFERROR wrapper comes into play, catching that error and returning the specified blank string (`""`), ensuring the list terminates cleanly.

Creating a Dynamic Multi-Result Dashboard

One of the most valuable aspects of using this complex INDEX MATCH array structure is the ability to create truly dynamic reports. Because the lookup criterion is isolated in cell **D2**, changing the value in that single cell instantly recalculates the entire output column (Column E), providing a real-time update of the results.

For example, if you change the team name in cell **D2** from "Mavs" to "Rockets", the entire list in column E will instantly adjust, retrieving only the names of players associated with the Rockets team. The use of absolute references ($) for the data ranges ensures that the structure remains stable regardless of the lookup criteria change.

	A	B	C	D	E
1	**Team**	**Player**		**Team**	**Players**
2	Mavs	Dan		Heat	Carl
3	Mavs	Mike			Jake
4	Warriors	Steven			Dawkins
5	Heat	Carl			
6	Heat	Jake			
7	Mavs	Brad			
8	Warriors	Tyler			
9	Kings	AJ			
10	Mavs	Spencer			
11	Heat	Dawkins			
12					
13					
14					
15					

This dynamic linkage makes the formula highly efficient for building user-friendly interfaces within Google Sheets, such as drop-down menus connected to cell D2, allowing users to select a criterion and immediately view all associated records. This is a massive improvement over manual filtering or running multiple single lookups.

Exploring Alternative Solutions for Multiple Results

While the array-based INDEX MATCH formula is a robust, cross-platform solution (often necessary for compatibility with other spreadsheet software), Google Sheets offers more modern, streamlined functions specifically designed to handle dynamic array outputs, such as the `FILTER` function.

VLOOKUP vs. INDEX MATCH vs. FILTER

When selecting a method, practitioners should weigh the trade-offs between compatibility, complexity, and ease of use:

Advanced INDEX MATCH (The discussed method): Highly compatible with Excel and powerful for complex conditional logic. However, the syntax is long and prone to errors if not constructed perfectly. It relies on sequential filling (dragging down) and the IFERROR wrapper to manage empty results.

`FILTER` Function: Unique to Google Sheets (and modern Excel). The syntax is significantly simpler: `=FILTER(Result_Range, Criteria_Range = Criteria_Value)`. This function

automatically spills the results into the adjacent cells, requiring only a single formula entry. It is the preferred modern method for simple multi-column, multi-row lookups in Google Sheets.

Although the `FILTER` function simplifies the task significantly, mastering the advanced INDEX MATCH technique is invaluable for scenarios where you need granular control over the output, or when operating in environments that do not support the newer `FILTER` function's dynamic array output. It represents a deeper understanding of how spreadsheet functions process arrays and conditional logic.

Summary of Key Benefits

The structured implementation of INDEX MATCH for multiple results transforms data retrieval capabilities in Google Sheets. This approach ensures high reliability and flexibility when dealing with real-world datasets.

Key advantages include:

Efficiency for Large Datasets: By utilizing array processing, the formula quickly identifies all matching records without requiring manual sorting or multiple filters.
Dynamic Reporting: The output list adjusts instantly when the lookup criterion is modified, perfect for interactive dashboards.
Robust Error Handling: The inclusion of IFERROR prevents unsightly error messages from cluttering the spreadsheet once all available results have been retrieved.
Sequential Output Control: The SMALL and ROW functions ensure that the results are pulled in the order they appear in the source data, maintaining data integrity.

Mastering this technique empowers users to move beyond simple lookups and perform sophisticated conditional data extraction, thereby greatly enhancing data management capabilities within the Google Sheets environment.

Further Learning and Related Tutorials

For those interested in extending their knowledge of advanced data manipulation, we recommend exploring tutorials on related concepts:

How to implement two-way lookups using INDEX MATCH.
Leveraging the `QUERY` function for SQL-like data retrieval in Google Sheets.
Understanding the array behavior of the `ARRAYFORMULA` function.

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