Learning Guide: Extracting Multiple Matches Using INDEX and MATCH in Google Sheets

Authored by

Mohammed looti

November 12, 2025

Addressing the Need for Multi-Result Lookups in Google Sheets

While fundamental lookup functions, such as the widely used VLOOKUP or the standard INDEX MATCH combination, are highly effective for retrieving a single, unique data point, they are insufficient when dealing with complex datasets that require the extraction of multiple corresponding values. In numerous real-world data analysis scenarios, a single criterion--such as a specific product code or a team name--is often associated with several entries, establishing a critical one-to-many relationship. To manage this dynamic extraction process effectively within a spreadsheet environment like Google Sheets, analysts must deploy a sophisticated, array-based formula structure.

This powerful technique requires the precise nesting of several core functions: specifically, INDEX, SMALL, IF, and ROW. The primary goal of this combination is to dynamically generate an array containing all relevant row numbers that satisfy the specified criteria. Furthermore, by wrapping the entire construction with the IFERROR function, we ensure the output is refined and clean, gracefully handling any errors that occur once all matching results have been successfully extracted. Mastering the logic and exact syntax of this formula is essential for performing advanced data extraction without relying solely on the simpler, but less versatile, FILTER function.

The detailed syntax provided below is the cornerstone for executing a multi-result lookup successfully in Google Sheets. It represents the optimal method for dynamic data extraction when it is anticipated that a single lookup key will correspond to multiple records. It is important to note that this functionality fundamentally operates as an Array Formula, although modern Google Sheets often handles the implicit array output automatically, simplifying the execution process compared to older spreadsheet software versions that required explicit array entry (Ctrl+Shift+Enter).

=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))),"")

In this specific configuration, the formula is meticulously engineered to systematically iterate through the dataset. It is designed to return all corresponding values found within the designated results range, **B2:B11**, contingent upon the associated criteria within the lookup range **A2:A11** being an exact match for the key value situated in cell **D2**. Our subsequent analysis will provide a meticulous breakdown of how each constituent function contributes synergistically to achieve this sophisticated mechanism.

Deconstructing the Advanced Iterative Lookup Mechanism

To truly gain mastery over this advanced data retrieval technique, one must first acquire a clear

comprehension of the specific role played by each nested function. This intricate formula effectively serves as a highly sophisticated extension of the standard INDEX MATCH concept, specifically engineered and adapted to manage multiple indices rather than just one. The fundamental core logic hinges on the successful generation of a sequence of relative row numbers that precisely correspond to every match found, which subsequently allows the formula to be seamlessly copied down the results column to sequentially retrieve all subsequent entries.

The process initiates with the innermost function grouping, `IF($D$2=$A$2:$A$11, ...)`, which is dedicated to performing the essential criteria check. In this step, the entirety of the lookup range, **A2:A11**, is systematically compared against the single, static lookup value residing in cell **D2**. Whenever a match is positively identified, the IF function executes its true argument, which involves calculating the relative row number. This row number calculation, specified as `ROW($A$2:$A$11)-ROW($A$2)+1`, is absolutely vital because it transforms the absolute row numbers of the spreadsheet (e.g., 2, 3, 4, etc.) into relative positional indices (e.g., 1, 2, 3, etc.) within the scope of the defined data array. Conversely, if no match is identified, the IF function automatically returns a value of `FALSE`.

The resultant Array Formula, which is now composed of a mix of relative row numbers and `FALSE` values, is then input directly into the powerful SMALL function. The defining role of the SMALL function is to control the iteration process. The argument `ROW(1:1)` is crucial here, as it dynamically dictates which smallest row number should be retrieved in the current iteration. When the formula is initially placed in cell E2, `ROW(1:1)` evaluates to 1, thus retrieving the first smallest matching row number. As the formula is subsequently dragged down to cells E3, E4, and beyond, the argument changes to `ROW(2:2)`, then `ROW(3:3)`, evaluating to 2, 3, and so on, thereby retrieving the second, third, and subsequent smallest row numbers sequentially. This iterative mechanism is precisely what permits the sequential retrieval of multiple distinct results.

Finally, the outermost function, the INDEX function, utilizes the sequence of row numbers meticulously generated by the SMALL function to extract and pull the corresponding value from the designated results range, **B2:B11**. This entire robust construction is strategically encased within the IFERROR wrapper. This wrapper serves a critical role by substituting an empty string (`""`) whenever the SMALL function attempts to locate a row number that simply does not exist (i.e., when all available matches have already been returned). This crucial step ensures that the results column maintains a professional and clean appearance, preventing disruptive error messages like `#NUM!` from displaying beneath the final valid match.

Practical Application: Utilizing a Basketball Dataset

To clearly illustrate the practical utility of this powerful multi-match lookup formula, let us examine a typical data scenario involving a sports database. Assume we are working with a dataset within

Google Sheets that meticulously lists various professional basketball players, pairing each individual player with the name of their respective team. This setup perfectly exemplifies a classic one-to-many relationship, as it is evident that a single team name will correspond to multiple distinct players.

Our primary objective is to implement dynamic data filtering: we aim to input a single team name once and have the spreadsheet automatically generate a complete list of all associated player names. Successfully achieving this task requires the complex INDEX MATCH array structure that we have thoroughly analyzed in the preceding sections. The configuration of our sample dataset is presented visually below, where Column A contains the team names and Column B lists the player names that we intend to retrieve:

	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			

For the purposes of this demonstration, we will focus on retrieving the names of every player affiliated with the team designated as the **Mavs**. We must first designate a specific cell to hold our dynamic lookup criterion--in this case, cell **D2**--where we will input the text "Mavs." It is this central lookup cell that will serve as the engine, driving the entire dynamic filtering process for the resulting column of player names.

The next critical step in the implementation process involves initiating the lookup. The complete Array Formula must be entered into the very first cell of the results column, which, based on our setup, is cell **E2**. This single, correctly constructed formula, when properly implemented and subsequently copied down, will autonomously manage the entire sequence of data extraction

based on the crucial iterative nature provided by the nested SMALL and IF functions.

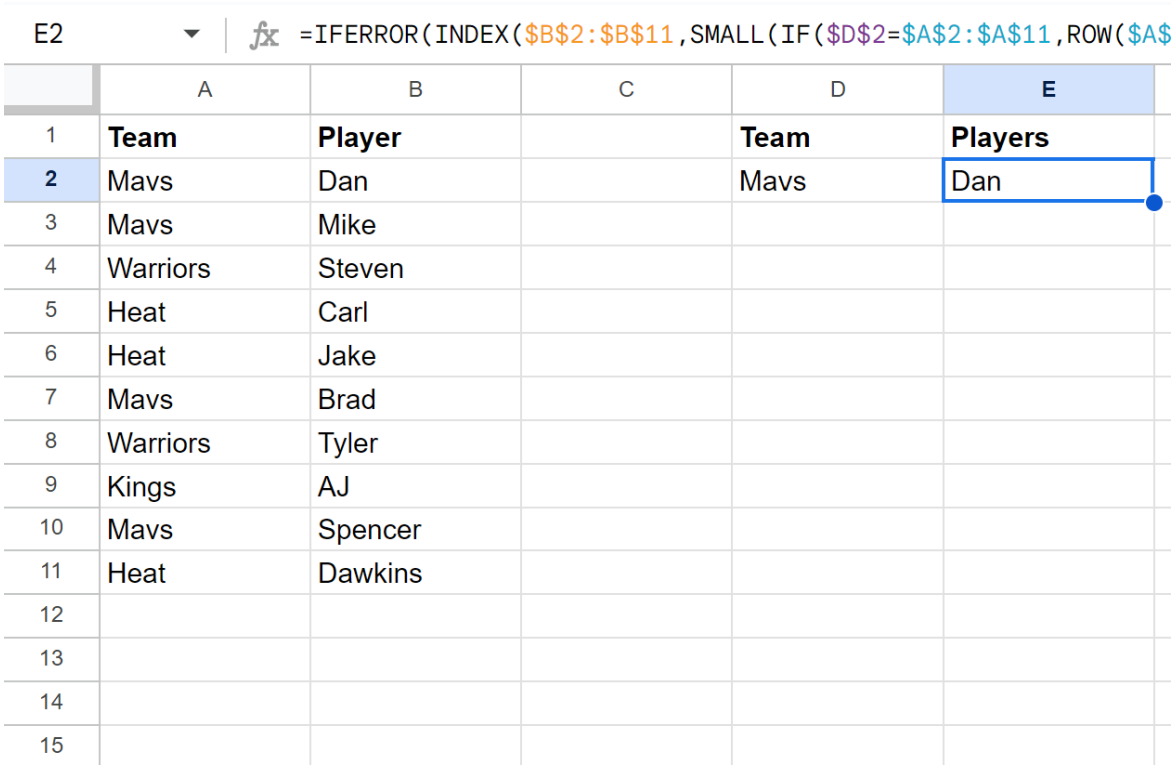
Execution and Interpretation of the Iterative Lookup

Following the setup phase, we proceed by accurately entering the complete formula into cell **E2**. It is absolutely mandatory that all cell references within the formula are correctly structured, rigorously utilizing **absolute references** (e.g., `$A$2`) for all ranges that must remain constant when the formula is copied or dragged, while simultaneously employing the necessary relative referencing for the iterative component, specifically `ROW(1:1)`. This balance ensures both stability and sequential progression.

The exact formula entered into cell **E2** is:

=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))),"")

Immediately upon pressing **Enter**, the spreadsheet engine executes the Array Formula. Because the initial `ROW(1:1)` evaluates to 1, the formula successfully isolates the very first instance where the team listed in column A corresponds exactly to the lookup value "Mavs" (found in cell D2), and subsequently returns the corresponding player name from column B. This crucial initial result serves to validate the entire setup and confirm that the complex core logic is functioning precisely as intended.



The true advantage and power of utilizing this iterative method become apparent when we replicate the formula across subsequent cells. By simply clicking and dragging the formula handle downwards from cell E2 to the remaining cells in column E, we activate the iterative component. This involves the sequential change of the `ROW(1:1)` argument to `ROW(2:2)`, `ROW(3:3)`, and so on, which automatically instructs the formula to find the remaining matches. Each cell in column E now rigorously searches for the Nth match (where N is determined by the current row offset), efficiently retrieving the names of all players associated with the Mavs team one by one.

E2:E5 =IFERROR(INDEX($B$2:$B$11,SMALL(IF($D$2=$A$2:$A$11,ROW(A

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

As clearly demonstrated within the resulting table, the names of all four players associated with the designated Mavs team are successfully retrieved and displayed in a clean, sequential list within Column E. This process provides a truly dynamic and easily reproducible solution for efficiently extracting multiple records based solely on a single lookup key, offering a substantial advantage over simpler, non-array-based lookup functions.

Achieving Dynamic Filtering and Unparalleled Flexibility

One of the most compelling and valuable attributes of this advanced INDEX MATCH technique is its intrinsic flexibility and remarkable dynamism. Because the core lookup criterion is isolated to a single dedicated cell, **D2**, and because all references to the underlying data ranges utilize absolute references (enforced by the use of dollar signs), the entire result set can be instantly and effortlessly updated simply by modifying the content of that single lookup cell. This design promotes robust and scalable spreadsheet models.

For instance, should the user decide to instantly switch the analysis to view the players for a completely different team, such as the "Lakers," they only need to accurately type "Lakers" into cell **D2**. The precise moment the value in **D2** is altered, the underlying Array Formula spread throughout column E recalculates automatically and instantaneously. The internal IF condition immediately generates a brand new set of matching relative row numbers, specifically tailored to the Lakers team, and the INDEX function then retrieves those corresponding player names based

on the updated indices.

This powerful dynamic capability positions the formula as an outstanding choice for constructing interactive dashboards or sophisticated reports within Google Sheets. Rather than requiring the user to rewrite formulas or physically manipulate data structures, they can efficiently change one single input cell to perform complex filtering operations across the entire dataset. When the available matching results are exhausted (for example, if the Lakers only have three players listed in the data), the trailing cells automatically display empty strings due to the safety wrapper provided by the reliable IFERROR function, thus ensuring the visual output remains impeccably clean and professional.

	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					

As clearly depicted above, instantaneously changing the criterion in **D2** immediately updates column E to accurately display the players associated with the "Lakers." This successfully demonstrates the inherent robustness achieved by meticulously combining the SMALL and ROW functions within the advanced INDEX MATCH framework, thereby achieving flexible and sophisticated dynamic data extraction that significantly surpasses the limitations of static lookup methodologies.

Comparative Advantage Over VLOOKUP and FILTER

Although the FILTER function is commonly suggested as the most straightforward solution for retrieving multiple results in contemporary spreadsheet applications, the complex INDEX MATCH

array technique retains substantial value. This is particularly true when considering crucial factors such as formula compatibility, granular control over the output format, and integration with legacy systems or non-array dependent functions.

Traditional lookup functions, most notably VLOOKUP, are fundamentally engineered to terminate the search process immediately after successfully locating the very first match. They are simply not designed with the capability to continue scanning the data range to locate subsequent matching entries. The advanced iterative INDEX MATCH structure masterfully overcomes this intrinsic limitation by compelling the calculation to return a comprehensive array of row positions for all matches, effectively transforming the standard lookup into a sequential, iterative process that is driven by the dragging action.

Furthermore, in environments where specific cell-by-cell formatting or seamless integration with other non-array-specific formulas is a necessary requirement, the iterative INDEX MATCH method provides far superior control. In contrast to the FILTER function, which automatically "spills" all results into contiguous cells and must be contained entirely within a single formula call, this robust method places each individual result into its own separate cell. This crucial distinction enables independent cell manipulation, the application of conditional formatting to specific entries, or integration into other complex formulas that rely exclusively on single-cell inputs, all without the disruptive "spill" effects often associated with a traditional Array Formula.

Continuing Your Google Sheets Mastery

Achieving proficiency in the dynamic retrieval of multiple results is a significant milestone toward advanced data proficiency within Google Sheets. The successful implementation of these complex array operations requires a firm and comprehensive understanding of how individual functions interact and behave when placed within deeply nested structures. To further solidify and enhance your spreadsheet skills, we strongly recommend exploring educational resources that delve into related advanced topics.

These specialized resources are designed to help you fully leverage the immense analytical power inherent in your spreadsheet software, allowing you to move decisively beyond basic data entry tasks and towards sophisticated, high-level data processing and robust reporting capabilities.

The following tutorials and concepts will help you execute other common advanced data tasks in Google Sheets:

Gaining a deeper understanding of the fundamental differences between VLOOKUP and INDEX MATCH for standard, single-result lookups.

Properly utilizing the IFERROR function to ensure robust and reliable error handling across all your

complex formulas.

Exploring advanced techniques for combining the FILTER and INDEX functions in complex data queries and reports.