Excel: Use VLOOKUP to Return Multiple Values in One Cell

Authored by

stats writer

November 17, 2025

In advanced data analysis and management within Excel, users frequently encounter the need to perform powerful lookups. While the established VLOOKUP function is a cornerstone of spreadsheet functionality, it presents a significant constraint: it is inherently designed to return only the first matching value it finds for a given criterion. This limitation becomes problematic when dealing with datasets where a single lookup key corresponds to multiple results--a common requirement in reporting and aggregation tasks. This guide will explore an innovative and highly effective solution that leverages the power of the TEXTJOIN function combined with Array Formula logic to consolidate multiple corresponding values into a single, clean cell output.

The Limitations of Traditional Lookup Functions

For decades, the standard approach for retrieving data based on a matching key has involved functions like VLOOKUP, and more recently, the combination of INDEX and MATCH. While these tools are indispensable for one-to-one lookups, they fundamentally fail when faced with a one-to-many relationship in the source data. If, for instance, you are tracking sales records and need to find all product IDs associated with a specific vendor name that appears multiple times in the vendor column, standard lookup functions will only retrieve the first entry they encounter, ignoring all subsequent matches. This forces analysts to resort to complex pivots or manual filtering, which are time-consuming and prone to error, especially when dealing with dynamic datasets.

The requirement to aggregate results into a single cell, perhaps for reporting headers or concatenated summaries, necessitates a different approach entirely. Traditional formulas lack the inherent iterative capability to scan an entire range, identify every match, and then compile those matches into a unified string. This challenge has historically been addressed using complex legacy techniques involving concatenation loops or even VBA (Visual Basic for Applications). However, the introduction of modern functions in recent versions of Excel, particularly TEXTJOIN, provides a far cleaner and more efficient path toward achieving multi-value lookup functionality without relying on macros or extensive helper columns.

Introducing the TEXTJOIN and Array Formula Solution

The modern solution for this multi-value retrieval challenge hinges on combining three key elements: the TEXTJOIN function, the logical IF function, and the necessity of entering the formula as an Array Formula (though in newer versions of Excel, dynamic arrays often handle this automatically). TEXTJOIN is a powerful function designed to concatenate multiple text strings into a single string, crucially allowing the user to specify a delimiter to separate the values and ignore empty cells. This latter feature is essential for filtering out non-matching results.

The core idea is to first generate an array of potential return values. We use the IF function to check the entire lookup column against the specific criterion. If a row matches the criterion, the

corresponding value from the return column is included in the array; if it does not match, the formula returns an empty string (""). This process creates a mixed array containing the desired results interspersed with empty strings. Finally, TEXTJOIN takes this array, stitches the non-empty values together using the specified delimiter, and delivers the aggregated result.

Deconstructing the Core Formula Syntax

The specific syntax required to execute this complex conditional aggregation is highly structured. Understanding each component is crucial for successful implementation and debugging. The formula effectively replaces the single-value limitation of VLOOKUP by performing a conditional selection over an entire range.

The required structure looks like this:

=TEXTJOIN(", ",,IF($A$2:$A$12=D2,$B$2:$B$12,""))

Let us break down the components of this powerful formula for clarity:

TEXTJOIN(", ", , ...): This is the outer wrapping function. The first argument, `", "`, defines the delimiter (a comma followed by a space) used to separate the collected values. The second argument, which is an empty placeholder (represented by the double comma `,,`), is set to `TRUE` by default in newer versions of Excel, instructing the function to **ignore empty cells**--a critical step that removes the "" results generated by the IF function for non-matches.
IF($A$2:$A$12=$D2, $B$2:$B$12, ""): This conditional expression performs the actual lookup. It evaluates the entire range `$A$2:$A$12` against the lookup criterion found in cell `D2`. Whenever the condition is met (`TRUE`), the corresponding value from the results range `$B$2:$B$12` is included in the array passed to TEXTJOIN. If the condition is `FALSE`, an empty string `""` is returned instead.

This powerful combination dictates that the formula will look up the specific value in cell **D2** across the lookup range **A2:A12** and return all corresponding matching values found in the result range **B2:B12**, neatly packaged into a single cell.

Practical Application: Setting Up the Dataset

To illustrate this functionality, consider a common scenario in sports analytics where we track the points scored by various basketball players across different teams. Our goal is to query the data based on the team name and retrieve all individual point totals associated with that team, consolidating them into one cell for quick reporting. This demonstration relies on a basic dataset structured with two primary columns: the Team name (our lookup column) and the Points scored (our return column).

The following visual representation shows the starting dataset, where multiple players belong to the same team, necessitating a multi-value lookup solution:

	A	B	C	D	E
1	**Team**	**Points**			
2	Mavs	22			
3	Mavs	14			
4	Rockets	19			
5	Spurs	30			
6	Mavs	40			
7	Nuggets	39			
8	Rockets	35			
9	Jazz	12			
10	Nuggets	17			
11	Spurs	23			
12	Spurs	15			
13					
14					
15					
16					
17					

In this example, the data spans rows 2 through 12. We establish two distinct areas: the data table itself (Columns A and B) and the lookup area (Column D and E). We intend to input the Team name we are searching for into cell **D2**, and the resulting aggregated point totals will populate cell **E2**. This setup provides a clean interface for demonstrating the efficiency and utility of the TEXTJOIN approach compared to traditional methods that would fail to capture all relevant entries.

Executing the TEXTJOIN Array Formula (Example 1)

Suppose our specific task is to find all point totals corresponding to the team named "Mavs." We enter "Mavs" into cell **D2**. Since "Mavs" appears several times in the Team column, we must retrieve multiple point values (55, 12, 18, and 22, according to the table). Using the TEXTJOIN structure is the ideal method for aggregating these discrete numerical results into a single, comprehensive text string.

To execute this, we input the complete Array Formula into cell **E2**. It is imperative to remember that when using older versions of Excel (pre-Microsoft 365 or Excel 2019), this formula must be confirmed by pressing **Ctrl + Shift + Enter** rather than just Enter, which signifies to Excel that an

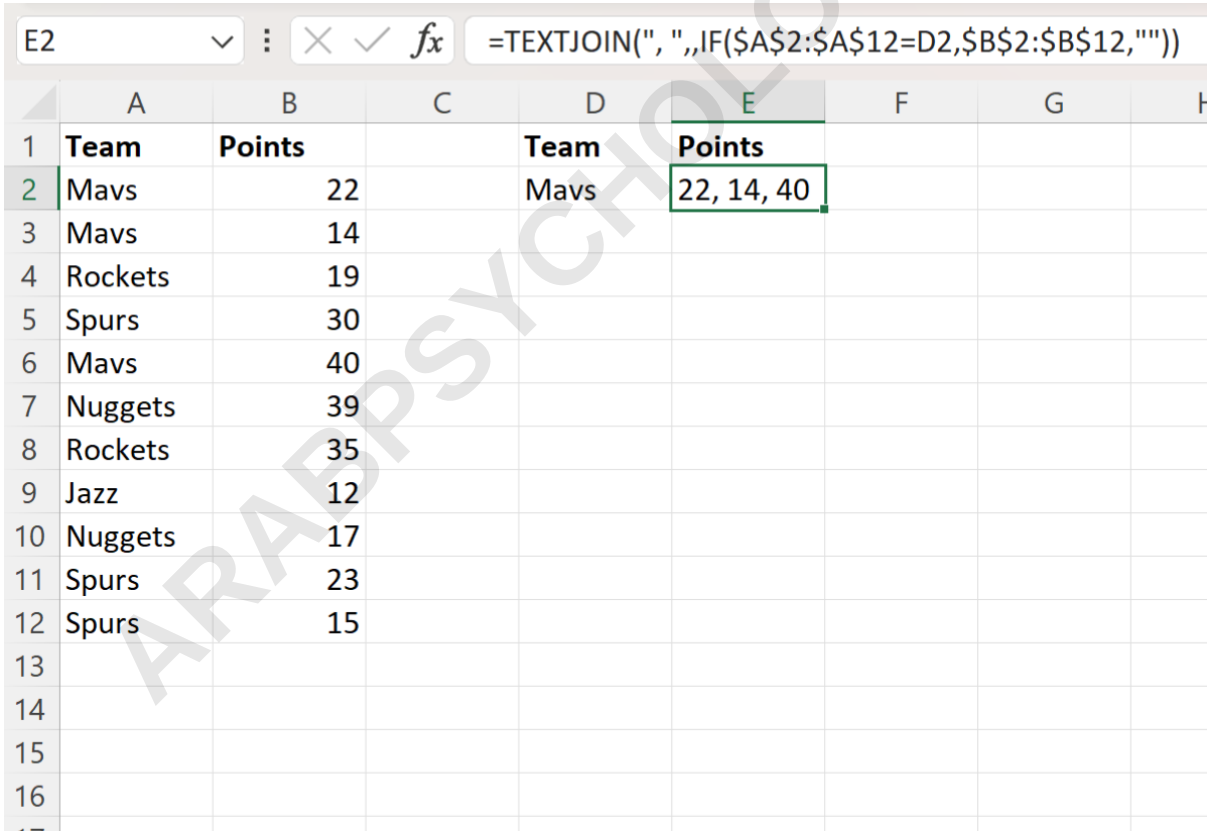
array operation is being performed. This procedural step generates the necessary curly braces `{ }` around the formula, enabling the IF function to return an array of results instead of a single value.

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

=TEXTJOIN(", ",,IF($A$2:$A$12=D2,$B$2:$B$12,""))

Upon execution, the calculation flow proceeds as follows: the IF statement checks all 11 cells in A2:A12 against D2 ("Mavs"). For every match, the corresponding number from B2:B12 is pulled. This generates an array like `{"", 55, "", "", 12, "", 18, "", 22, "", ""}`. The TEXTJOIN function then processes this array, ignoring the empty strings and combining the remaining numeric values using the specified delimiter, resulting in a clean, concatenated string.

The following screenshot demonstrates the successful application of the formula, showing the aggregated result in cell E2:



Interpreting the Results and Understanding Delimiters

The resulting output in cell E2, which reads "55, 12, 18, 22," confirms that the formula successfully retrieved all instances where the **Team** column matched "Mavs" and compiled the corresponding

values from the **Points** column into a single cell. This aggregated output is significantly more useful for immediate analysis or display than the single-value return provided by a traditional VLOOKUP.

A closer look at the formula structure reveals the critical role of the delimiter--the character or string used to separate the concatenated values. In our initial formula, we explicitly defined the delimiter as `", "` (a comma followed by a space). This choice enhances readability, making the resulting list of numbers easy for a human reader to parse quickly. The power of the TEXTJOIN function lies in its flexibility; unlike basic concatenation using the ampersand (&), TEXTJOIN allows for precise control over how the aggregated data is presented, which is crucial for downstream systems that may rely on specific parsing characters.

The final result visually confirms the successful retrieval of all four matching values:

	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Points**	
2	Mavs	22		Mavs	22, 14, 40	
3	Mavs	14				
4	Rockets	19				
5	Spurs	30				
6	Mavs	40				
7	Nuggets	39				
8	Rockets	35				
9	Jazz	12				
10	Nuggets	17				
11	Spurs	23				
12	Spurs	15				
13						
14						
15						
16						

Customizing the Output: Changing the Delimiter

One of the greatest advantages of using TEXTJOIN for multi-value lookups is the ease with which the output format can be customized simply by changing the first argument--the delimiter. Depending on reporting requirements, a comma and space might not be suitable; perhaps the data needs to be separated by a pipe symbol (|), a semicolon (;), or simply a space for a compact appearance. This flexibility ensures that the aggregated data meets various specifications without

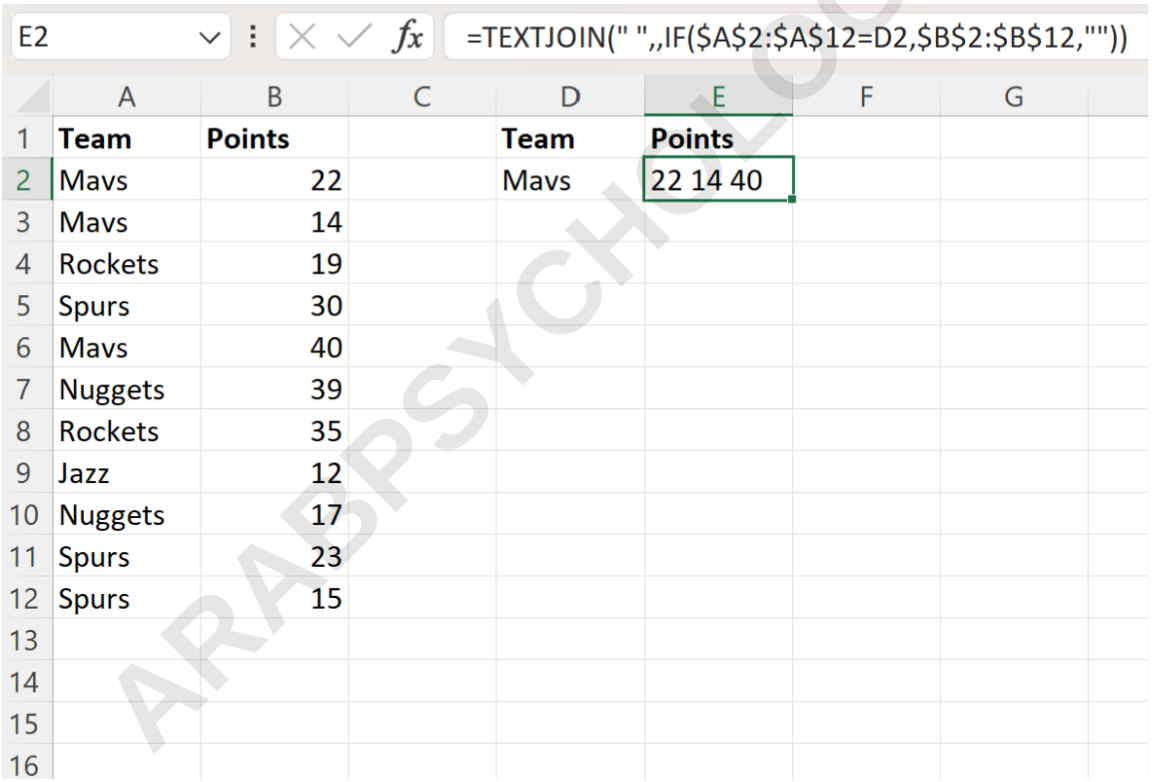
altering the core conditional logic.

For example, if the requirement is to use a single space as the separation character instead of a comma and space, we only need to modify the first argument within the TEXTJOIN function from `", "` to `" "`. This minor adjustment demonstrates the robust control afforded by this function over the resulting string output.

We can use the following revised formula to implement a space as the delimiter:

=TEXTJOIN(" ",,IF($A$2:$A$12=D2,$B$2:$B$12,""))

The resulting screenshot clearly shows how the output presentation changes when the delimiter is adjusted to a single space, providing a more condensed output while retaining all the relevant data points retrieved by the Array Formula logic:



	A	B	C	D	E	F	G
1	**Team**	**Points**		**Team**	**Points**		
2	Mavs	22		Mavs	22 14 40		
3	Mavs	14					
4	Rockets	19					
5	Spurs	30					
6	Mavs	40					
7	Nuggets	39					
8	Rockets	35					
9	Jazz	12					
10	Nuggets	17					
11	Spurs	23					
12	Spurs	15					
13							
14							
15							
16							

This formula now returns all of the matching values from the **Points** column for each row where the **Team** column is equal to "Mavs," but with each value separated by a single space, resulting in the output "55 12 18 22." This approach is vastly superior to attempting to use older concatenation methods, which require complex error handling to avoid including unwanted delimiters for non-matching rows. Using TEXTJOIN ensures a clean, professional output tailored exactly to the user's needs.

Conclusion and Advanced Considerations

While the VLOOKUP function remains crucial for simple, single-match lookups, data analysts must utilize advanced techniques for handling one-to-many relationships. The combination of the TEXTJOIN function and IF function logic entered as an Array Formula provides a robust, dynamic, and easy-to-maintain method for retrieving and aggregating multiple values into a single cell. This technique eliminates the dependency on complicated legacy formulas or external programming solutions.

When implementing this solution, always ensure that your ranges are properly referenced, utilizing absolute references (e.g., `$A$2:$A$12`) when necessary, especially if you intend to copy the formula to other cells. Furthermore, remember that the availability of TEXTJOIN is limited to newer versions of Excel (Excel 2019 and Microsoft 365). Users on older platforms may need to explore older, more cumbersome array formulas involving `CONCATENATE` and `TRANSPOSE`, or rely on VBA. By mastering the TEXTJOIN approach, you significantly enhance your ability to summarize complex data efficiently and accurately.

Note: You can find the complete documentation for the **TEXTJOIN** function on the official Microsoft support website.