How to Return Multiple Matches with VLOOKUP and TEXTJOIN in Excel: A Comprehensive Guide

Authored by

Mohammed looti

November 10, 2025

When analysts manage extensive and detailed datasets within Microsoft **Excel**, a recurring requirement is the need to efficiently retrieve data points corresponding to a specific lookup criterion. The traditional go-to solution is the **VLOOKUP** function, a cornerstone of data analysis. However, a significant inherent limitation of **VLOOKUP** is that it is fundamentally designed to retrieve only the very first matching entry it encounters, completely ignoring any subsequent records that meet the same criteria. This restriction poses a critical challenge when the goal is to consolidate multiple associated values that match a single criterion into one cell, necessitating the adoption of more dynamic and powerful functions capable of handling conditional array processing and text aggregation.

To successfully overcome this limitation and accurately aggregate multiple matching results, we must implement a sophisticated combination of functions. This methodology centers on pairing the modern **TEXTJOIN** function with the powerful conditional processing capabilities of the **IF** function. By structuring these elements into an array formula, we first identify every instance where the lookup criteria are met and then seamlessly join all the corresponding results using a specified separator. This advanced technique is absolutely indispensable for generating streamlined reports or performing complex data analysis tasks where a consolidated, single-cell output is mandatory for clarity and efficiency.

Deconstructing the Solution: TEXTJOIN, IF, and Array Logic

Before implementing the complex formula, it is essential to establish a clear understanding of the core functions driving this solution. The key to aggregation lies within the **TEXTJOIN** function, which became available in Excel 2016 and is standard in Microsoft 365 environments. This function is specifically engineered to combine text from multiple ranges or strings, providing the crucial ability to specify a delimiter that is inserted between each combined text item. This capability is precisely what allows us to transform disparate lookup results into a cohesive, single-cell string.

The standard syntax for **TEXTJOIN** requires three main arguments: the delimiter (the separation character, such as a comma, pipe, or space), a boolean value determining whether empty cells should be ignored (which we typically set to TRUE), and the list of text items to be joined. In this advanced implementation, the list of text items is not a static range but is dynamically generated by an embedded **IF** statement operating across an array of cells. This means the formula evaluates the condition against every row in the specified range simultaneously, a fundamental characteristic of **array formulas** in Excel.

When the **IF** function is executed within this array context, it performs a logical test across the entire lookup column. If the condition is met (for example, if the team name in the lookup column matches the specified lookup value), the **IF** function returns the corresponding value from the

designated results column. Conversely, if the condition is not met, it returns an empty string (""). This resulting array, which is a mixture of actual results and placeholder empty strings, is then seamlessly passed to the **TEXTJOIN** function, which handles the final concatenation and automatically filters out the empty strings, resulting in the desired aggregated output.

Implementing the Advanced Formula Syntax

To successfully execute this multi-match lookup, we must structure the functions into a single, cohesive formula that leverages Excel's implicit array handling capabilities, thereby circumventing the inherent single-return limitation found in functions like **VLOOKUP**. The structure is designed to look up a value in one column, identify all matching instances, and return all corresponding values from a secondary column, combining them into a singular cell.

The precise formula used to achieve this powerful aggregation is:

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

Let us thoroughly analyze the components of this critical formula. The outermost function, **TEXTJOIN(", ",...)**, initiates the aggregation process. The first argument, **", "**, defines the **delimiter**--a comma followed by a space--that will separate the returned values. The subsequent comma (representing the second argument, which is left blank or implicitly TRUE) instructs **TEXTJOIN** to ignore all empty cells generated internally. The nested **IF** function then executes the core conditional logic across the array.

The logical test, **$A$2:$A$12**=**D2**, is the engine of the operation. It rigorously compares every cell within the designated lookup range (A2 through A12) against the specific lookup value residing in cell **D2**. If this comparison yields **TRUE**, the formula returns the corresponding value from the result range, **$B$2:$B$12**. If the comparison is **FALSE**, it returns an empty string, represented by the double quotes (""). Thanks to the modern architecture of Microsoft **Excel**, this intermediate array of results is intelligently processed and seamlessly passed to **TEXTJOIN**, which performs the final concatenation, resulting in a single string of all matching values.

Practical Demonstration: Consolidating Sports Data

To fully appreciate the utility of this combined function approach, we will apply it to a practical, real-world data scenario. Imagine a spreadsheet within Microsoft **Excel** that tracks the individual points scored by various basketball players, categorized meticulously by their respective teams. Our primary objective is to look up a specific team name and retrieve all the corresponding individual scores, consolidating them into one clear, easily interpretable cell.

The illustrative structure of our sample dataset is presented below, detailing player points

alongside their team affiliations:

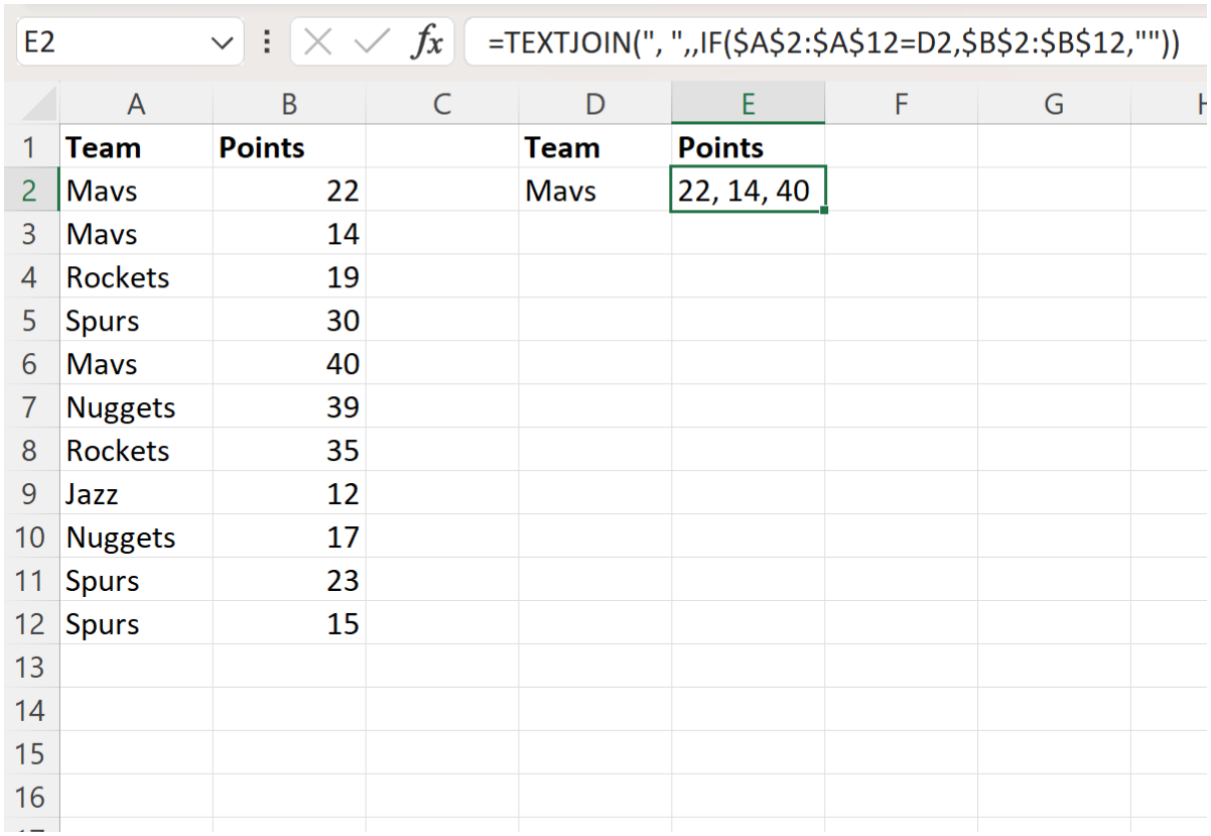
	A	B	C	D	E
1	**Team**	**Points**			
2	Mavs	22			
3	Mavs	14			
4	Rockets	19			
5	Spurs	30			
6	Mavs	40			
7	Nuggets	39			
8	Rockets	35			
9	Jazz	12			
10	Nuggets	17			
11	Spurs	23			
12	Spurs	15			
13					
14					
15					
16					
17					

In this example, we aim to perform a lookup specifically for the team designated as "Mavs" located within the **Team** column (Column A) and extract all associated point values from the **Points** column (Column B). We will define the team we are searching for, "Mavs," in cell **D2**. The result of the consolidated scores will be placed in cell **E2**.

We navigate to cell **E2** and input the complete **array formula**:

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

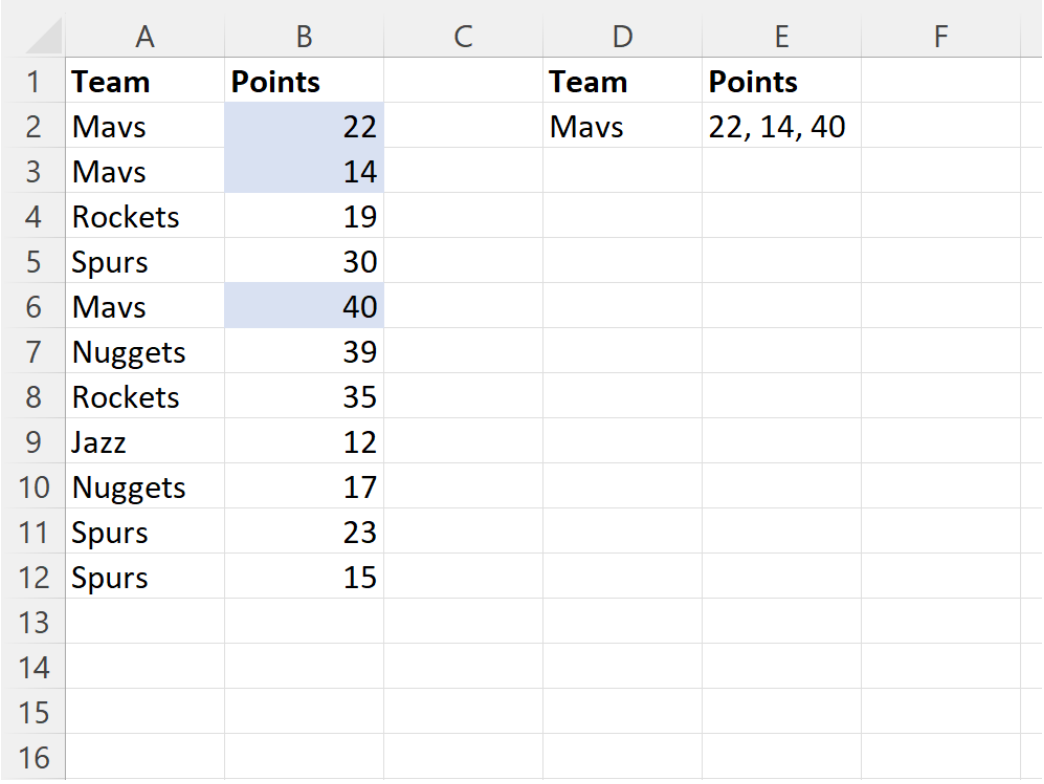
Once the formula is entered and confirmed, the spreadsheet recalculates, and the output cell **E2** instantly displays the consolidated data. The following image clearly illustrates the implementation, showing the lookup value in D2 and the applied formula in E2:



	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							

Upon execution, the formula meticulously scans the data range **A2:A12**. For every row where "Mavs" is identified, it extracts the corresponding point value from Column B. Given that "Mavs" appears three times in the dataset (with scores 15, 22, and 19), the formula successfully aggregates these three numerical values. The final output presented in cell **E2** is the single string: "15, 22, 19". This provides a concise and comprehensive summary of all relevant data points for the selected team, validating the power of this array method.

The subsequent screenshot confirms the final, aggregated result, demonstrating how the formula successfully returned all matching values from the **Points** column for every row where the team criteria equaled "Mavs," completing the successful data consolidation:



	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 Output Presentation by Modifying the Delimiter

A significant advantage derived from utilizing the **TEXTJOIN** function is the superior flexibility it offers in controlling the final output format. While the preceding example correctly used a comma and a space (", ") as the separator, the first argument of the function--the **delimiter**--can be effortlessly customized to meet any specific reporting or presentation standards required. This essential flexibility allows data professionals to adhere to distinct requirements, whether they necessitate pipe symbols (|), dashes (-), or simple spaces to cleanly separate the aggregated data points.

For instance, if the reporting requirement specifies using only a single space to separate the scores, perhaps to visually represent the output as a sequence of numbers rather than a formal list, we must simply modify the first argument of the **TEXTJOIN** function. We replace the string ", " with a single space character enclosed within quotation marks (" ").

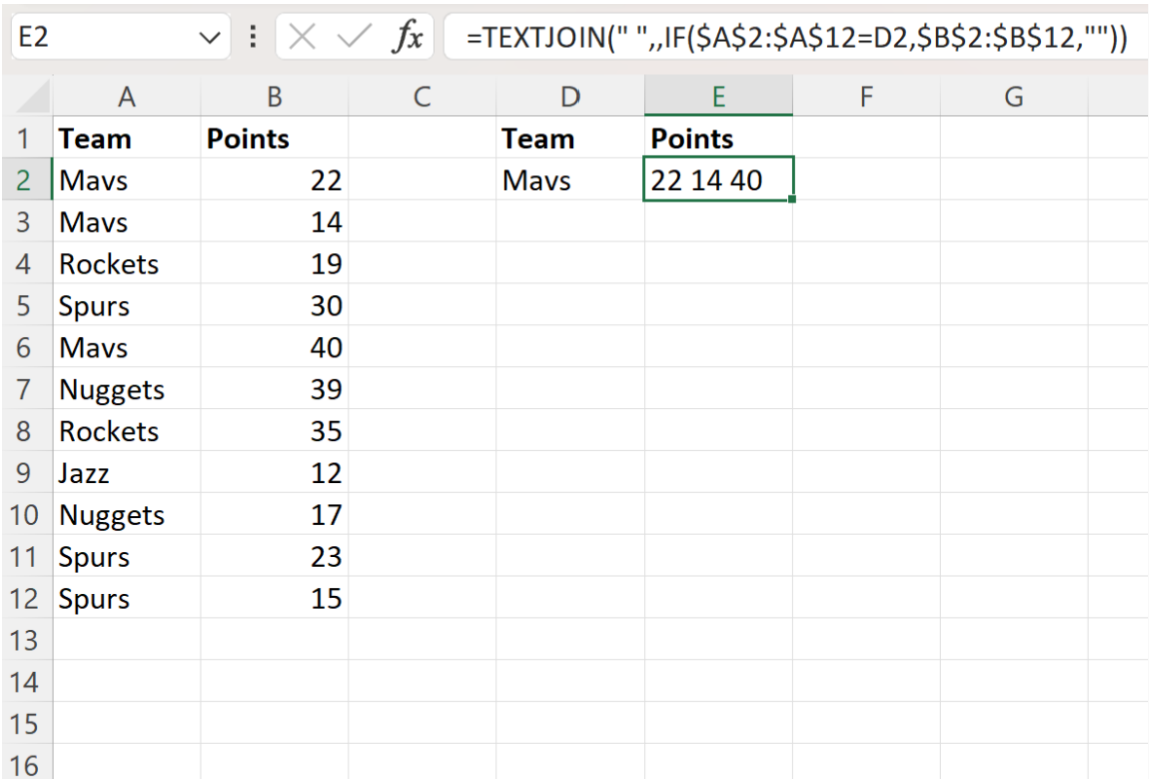
The revised formula syntax necessary to implement a single space as the **delimiter** is:

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

This minor alteration fundamentally changes the presentation of the output string while maintaining the core functionality of the lookup and aggregation process. The inner mechanics--the **IF** function

generating the conditional array--remain absolutely identical; only the final presentation layer is affected by the modification of the delimiter argument. This customization capability ensures that the final data output is not only accurate but also formatted optimally for any subsequent use, whether for external reporting or further analysis.

The following screenshot clearly illustrates the result when this modified formula is applied. Notice how the values are now separated exclusively by a space, providing a cleaner, more concise sequence of results:



	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							

As clearly demonstrated, this formula successfully returns all matching values from the **Points** column for each row where the **Team** column is equal to "Mavs," with each value now separated solely by a space. This highlights the high degree of customizability essential for professional-grade data handling and reporting.

Conclusion: Mastering Multi-Value Lookups

While the standard **VLOOKUP** function serves as a foundational tool for single-value lookups in Microsoft **Excel**, it is clearly insufficient for scenarios demanding the aggregation of multiple matching results. By skillfully combining the text-joining capabilities of **TEXTJOIN** with the powerful conditional logic of the **IF** function within an **array formula** structure, data analysts can efficiently bypass this critical limitation. This robust technique enables the consolidation of complex, multi-row

results into streamlined, single-cell outputs, significantly boosting report clarity and overall analytical efficiency.

Achieving mastery over this specific formula structure is crucial for anyone seeking advanced proficiency in spreadsheet management. It is important to recognize that this elegant approach is far superior to attempting complex, multi-step workarounds involving manual concatenation or intricate pivoting, especially when dealing with large, dynamic datasets where both efficiency and accuracy are paramount. We strongly encourage users to practice customizing the logical test within the **IF** function and experimenting with different **delimiters** in the **TEXTJOIN** function to fully harness the versatility and power of this robust method.

Additional Resources for Advanced Excel Operations

To further enhance your proficiency in advanced data manipulation and complex lookups within Excel, we recommend exploring the following tutorials which detail how to perform other common, sophisticated operations:

A comprehensive understanding of the operational differences between **VLOOKUP** and the **INDEX/MATCH** combination.

Effective use of **delimiters** in text functions to parse complex string data into separate columns.

Implementing dynamic sorting and filtering using modern array functions such as **SORT** and **FILTER**.

Applying advanced conditional formatting rules based on dynamic lookup results.