Learning to Use VLOOKUP in Excel to Return Data from Multiple Columns

Authored by

Mohammed looti

October 31, 2025

The **VLOOKUP function** in **Microsoft Excel** is widely recognized as a foundational tool for efficient data retrieval. Its primary utility involves searching for a specific value in the leftmost column of a data table and returning a corresponding piece of information from a single, designated column. While indispensable for basic lookups, its traditional single-column output presents a significant constraint when users need to extract multiple associated data points based on one search criterion.

Fortunately, advanced users of **Excel** have developed a powerful technique to transcend this limitation. This method leverages the structural flexibility of **VLOOKUP** by incorporating an **array constant** directly into the column index number argument. This simple modification transforms the function into a highly efficient multi-column data extraction engine. This comprehensive guide will walk you through the necessary syntax, explain the underlying logic, and provide a detailed practical example to help you integrate this advanced skill into your data analysis toolkit.

Understanding the Standard VLOOKUP Mechanics

To fully appreciate the efficiency of the multi-column approach, it is necessary to first review the standard operational mechanics of the **VLOOKUP function**. The name itself--Vertical Lookup--describes its action: scanning vertically down the first column of a specified table **range** to locate the desired piece of data.

The basic structure of the function dictates the parameters necessary for a successful lookup. Understanding the role of each argument is crucial before attempting the advanced technique. The core syntax of **VLOOKUP** is as follows: `=VLOOKUP(lookup_value, table_array, col_index_num, )`.

Each argument serves a specific purpose in directing the lookup process:

`lookup_value`: This required argument specifies the value that the function searches for in the first column of the specified table.
`table_array`: This defines the **range** of cells containing the data. It is mandatory that the first column of this range holds the `lookup_value`.
`col_index_num`: This argument historically accepts a single integer indicating the column number within the `table_array` from which the corresponding data should be retrieved. This single input is the source of the standard VLOOKUP's limitation.
: This optional argument determines the type of match. Setting it to `FALSE` (or `0`) ensures an **exact match**, which is highly recommended for accurate data retrieval. Setting it to `TRUE` (or omitting it) provides an approximate match, suitable only if the lookup column is sorted ascendingly.

Mastering Multi-Column Data Retrieval with Array Constants

The technique for enabling **VLOOKUP** to return multiple values simultaneously centers entirely on manipulating the `col_index_num` argument. Instead of providing a single column index number, we introduce an **array constant** containing all the desired column indices. This informs **Excel** that the function should return not one, but several corresponding values. The structure for this advanced operation is:

=VLOOKUP(lookup_value,table_array,{col1,col2,col3,...},FALSE)

This formulation transforms the standard lookup into a powerful **array formula**, which requires a specific input method detailed in the next section. Key changes in the argument definitions include:

`table_array`: This **range** must be carefully defined to encompass all columns whose data you intend to return, starting with the lookup column.
`{col1,col2,col3,...}`: This is the critical **array constant**, enclosed in curly braces `{}`. The numbers listed within must correspond to the relative column indices within the `table_array`. For example, if your `table_array` is A:E, and you want data from columns C, D, and E, your array constant would be `{3,4,5}`. The comma separator tells **Excel** to output the results horizontally across adjacent cells.
`FALSE`: Maintaining `FALSE` is essential to ensure an **exact match**, preventing unreliable data returns when working with unsorted or non-numeric data.

Practical Application: Step-by-Step Multi-Column Lookup

To demonstrate the utility of this method, let us apply it to a practical scenario involving a dataset of basketball team statistics in **Excel**. Our objective is to search for a team name and simultaneously retrieve its corresponding points, assists, and steals, which reside in separate columns.

Examine the sample dataset below, where team names are in Column A, Wins in Column B, Points in Column C (Column 3), Assists in Column D (Column 4), and Steals in Column E (Column 5).

	A	B	C	D	E	F
1	**Team**	**Rebounds**	**Points**	**Assists**	**Steals**	
2	Mavericks	12	22	6	4	
3	Rockets	14	28	7	4	
4	Pacers	8	24	7	6	
5	Hornets	7	24	8	5	
6	Rockets	11	25	7	7	
7	Celtics	19	19	6	8	
8	Knicks	15	15	9	3	
9	Nets	14	24	12	5	
10	Raptors	10	30	10	5	
11	Hornets	12	34	8	4	
12						
13						
14						
15						
16						
17						

Suppose we want to look up the team "Pacers" (our `lookup_value`, located in cell G2) and extract the data from columns 3, 4, and 5. We need the results to populate three adjacent cells (H2, I2, and J2). The structure of our array-based **VLOOKUP** formula will be:

=VLOOKUP(G2,A2:E11,{3,4,5},FALSE)

Critical Execution: Entering the Array Formula (Ctrl+Shift+Enter)

The most critical step when implementing this multi-column retrieval method is the formula entry process, as it involves creating an **array formula**. Failure to follow these steps precisely will result in incorrect output or an error message:

Select the Output Range: Before typing the formula, you must first select all the adjacent cells where you expect the results to appear. In this example, select cells H2, I2, and J2.

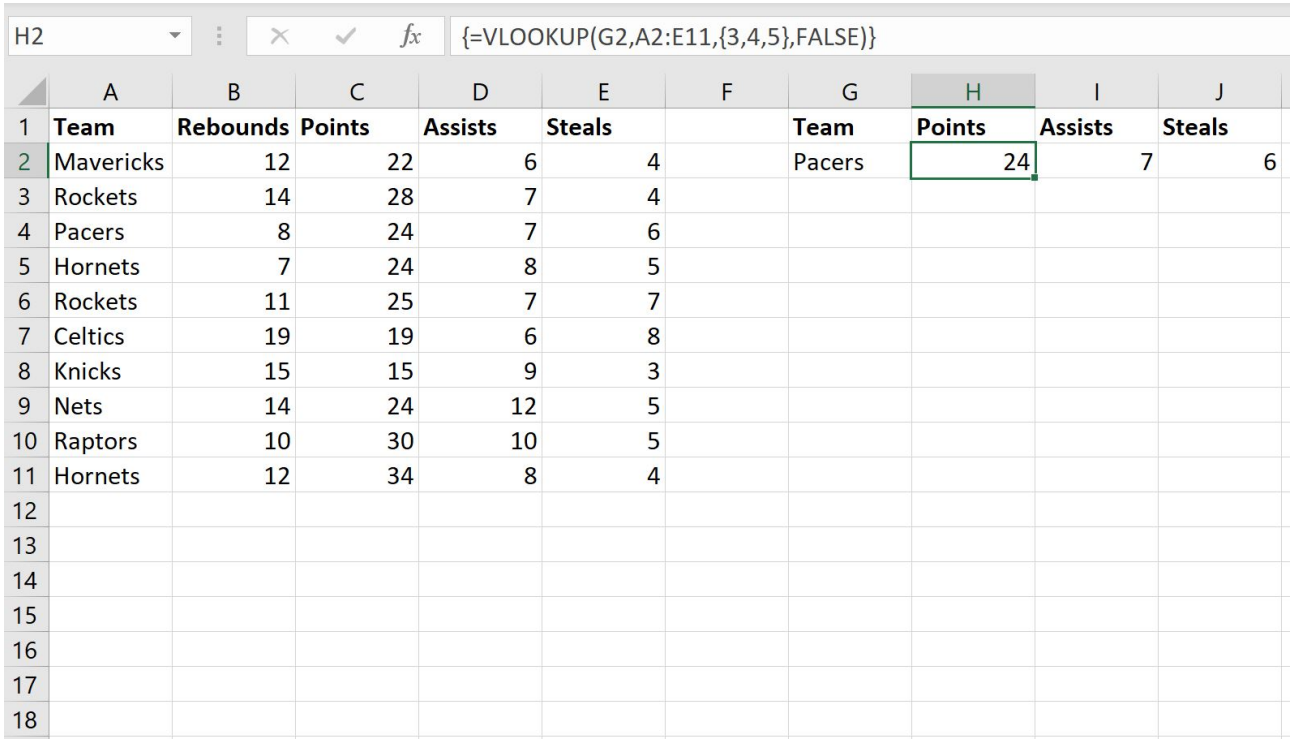
Type the Formula: With the entire range selected, type the formula exactly as shown above into the formula bar: `=VLOOKUP(G2,A2:E11,{3,4,5},FALSE)`.

Commit as an Array Formula: Instead of pressing the standard `Enter` key, you must press **`Ctrl + Shift + Enter`** simultaneously (often abbreviated as CSE). This command signals to **Excel** that the function should be treated as an array operation capable of outputting results across

multiple cells.

Upon successful CSE entry, **Excel** automatically encloses the formula in curly braces `{}` in the formula bar (e.g., `{=VLOOKUP(G2,A2:E11,{3,4,5},FALSE)}`). Note that you should never type these braces manually.

The following screenshot illustrates the formula entry process after selecting the target output cells:



	A	B	C	D	E	F	G	H	I	J
1	**Team**	**Rebounds**	**Points**	**Assists**	**Steals**		**Team**	**Points**	**Assists**	**Steals**
2	Mavericks	12	22	6	4		Pacers	24	7	6
3	Rockets	14	28	7	4					
4	Pacers	8	24	7	6					
5	Hornets	7	24	8	5					
6	Rockets	11	25	7	7					
7	Celtics	19	19	6	8					
8	Knicks	15	15	9	3					
9	Nets	14	24	12	5					
10	Raptors	10	30	10	5					
11	Hornets	12	34	8	4					
12										
13										
14										
15										
16										
17										
18										

Once entered, the **array formula** instantly populates the selected output range with the correct values corresponding to the "Pacers" lookup, demonstrating the power of the array constant method:

	A	B	C	D	E	F	G	H	I	J
1	**Team**	**Rebounds**	**Points**	**Assists**	**Steals**		**Team**	**Points**	**Assists**	**Steals**
2	Mavericks	12	22	6	4		Pacers	24	7	6
3	Rockets	14	28	7	4					
4	Pacers	8	24	7	6					
5	Hornets	7	24	8	5					
6	Rockets	11	25	7	7					
7	Celtics	19	19	6	8					
8	Knicks	15	15	9	3					
9	Nets	14	24	12	5					
10	Raptors	10	30	10	5					
11	Hornets	12	34	8	4					
12										
13										
14										
15										
16										
17										

Important Considerations and Best Practices

While utilizing an array constant within **VLOOKUP** is highly effective, adhering to certain best practices is essential for reliability, maintenance, and performance:

The CSE Requirement: Reiterate the necessity of the `Ctrl + Shift + Enter` input. If you need to edit the formula later, you must select the entire output range and re-enter the formula using CSE. If you simply press `Enter` after editing, the formula will revert to standard behavior, often resulting in errors or only the first value of the array being returned.

Indexing Accuracy: Always verify that the column numbers specified in your **array constant** `{}` accurately reflect the relative position of the desired columns within the defined `table_array`. A miscount is a common source of error.

Use Absolute References: To ensure the formula works correctly when copied or dragged across a spreadsheet, use absolute references (e.g., `$A$2:$E$11`) for the `table_array`. This prevents the lookup range from shifting incorrectly.

Performance Overhead: **Array formulas** can be computationally demanding, particularly when applied repeatedly across very large spreadsheets. For massive datasets, consider alternative, more efficient methods such as the combination of **INDEX and MATCH**, which offers greater flexibility, or the modern **XLOOKUP** function available in newer versions of **Excel**.

Error Handling with IFERROR: To gracefully manage scenarios where the `lookup_value` does not exist in the table, wrap the entire formula within an `IFERROR` function. A robust structure would look like: `=IFERROR(VLOOKUP(G2,A2:E11,{3,4,5},FALSE),"Data Missing")`.

Conclusion: Expanding VLOOKUP's Utility

The mastery of utilizing an **array constant** within the column index argument significantly expands the functional scope of the traditional **VLOOKUP function**. This technique allows for the instantaneous retrieval of multiple related data points with a single, elegant formula, saving substantial time and reducing the complexity associated with chaining multiple individual lookups.

While modern **Excel** versions offer functions like **XLOOKUP**, which inherently support multi-column returns, understanding the array constant method for **VLOOKUP** remains invaluable. It ensures broad compatibility across older **Excel** installations and provides a foundational understanding of how **array formulas** operate. By integrating this advanced **VLOOKUP** trick into your skill set, you gain powerful efficiency in managing and manipulating data.

Additional Resources

For more in-depth information and further examples of the **VLOOKUP function**, including its full documentation and various applications, please refer to the official Microsoft Office Support pages:

VLOOKUP function - Microsoft Support
Guidelines and examples of array formulas - Microsoft Support