How to Return Multiple Columns with VLOOKUP Alternatives in Google Sheets (Easy Guide)

Authored by

stats writer

November 28, 2025

The standard functionality of the **VLOOKUP** function is inherently designed to return a single corresponding value based on a specified lookup key. In collaborative spreadsheet environments like **Google Sheets**, a direct **VLOOKUP** operation is strictly limited to extracting data from only one column index specified in its arguments. This single-column constraint is a fundamental characteristic of the function's design, focusing on vertical data retrieval efficiency.

However, modern spreadsheet platforms offer sophisticated mechanisms to circumvent this limitation. While you cannot execute a vanilla **VLOOKUP** to fetch multiple columns, advanced users routinely employ powerful workarounds. These solutions typically involve harnessing the function in conjunction with the ArrayFormula function, which effectively enables the simultaneous retrieval of data across multiple columns by treating the column index parameter as an array itself. This combination transforms the typical scalar output into an array output.

Furthermore, if this combined approach proves overly complex or if enhanced flexibility and dynamic data manipulation are primary requirements, alternative, often more robust functions are readily available. These include the highly flexible pairing of INDEX and MATCH or the comprehensive QUERY function, both of which excel at performing complex lookups and data aggregation that inherently support multi-column outputs without needing functional nesting.

The VLOOKUP and ArrayFormula Workaround

To overcome the inherent restriction of **VLOOKUP** returning only one column, we introduce the **ArrayFormula** wrapper. This wrapper instructs **Google Sheets** to process the formula output as a collection of values rather than a single cell result. When applying this technique, the crucial modification occurs in the `index` argument of the **VLOOKUP** function, traditionally reserved for a single column number.

Instead of supplying a single integer (e.g., `2`), we provide a comma-separated array of integers enclosed in curly braces (e.g., `{2, 4}`). This array specifies all the column indices within the defined range that the **VLOOKUP** should return. The **ArrayFormula** then processes the **VLOOKUP** result for each index in the array and outputs the results horizontally, spanning multiple columns starting from the cell where the formula is entered. This powerful method allows the user to leverage the simplicity of **VLOOKUP** while achieving the desired multi-column output, making it an essential technique for data analysts working within **Google Sheets**.

The following syntax demonstrates how to effectively utilize the **VLOOKUP** function within an ArrayFormula in **Google Sheets** to look up a value in a range and return the corresponding values from multiple specified columns:

=ArrayFormula(VLOOKUP(A14, A2:D11, {2, 4}, FALSE))

In this specific formula structure, the operation is designed to identify the lookup value present in cell **A14** across the defined search range, **A2:D11**. Upon finding an exact match, it is programmed to return the corresponding values found in column index **2** and column index **4** of that defined range. The key mechanism enabling this multi-column retrieval is the use of the array constant `{2, 4}` within the column index parameter, facilitated by the outer **ArrayFormula** wrapper that ensures the output is expanded horizontally across adjacent cells.

Decoding the Formula Arguments

Understanding the components of the combined formula is crucial for successful implementation and debugging. Each parameter plays a specific role in ensuring accurate data retrieval. The `search_key`, represented here by `A14`, is the value we are attempting to locate within the first column of the data range. The `range`, denoted as `A2:D11`, establishes the entire dataset where both the lookup column and the result columns reside. It is paramount that the `search_key` exists in the very first column of this defined `range` for **VLOOKUP** to function correctly.

The most critical element for this multi-column application is the column indices parameter, `{2, 4}`. These numerical values refer to the relative position of the columns within the `A2:D11` range, where `A` is column 1, `B` is column 2, `C` is column 3, and so forth. By supplying this array, we instruct the function to return the data from columns B and D (indices 2 and 4, respectively). If we wanted to include column C, the array would be modified to `{2, 3, 4}`. This flexibility allows users to select any combination of columns in any desired order.

Finally, the `is_sorted` argument is set to `FALSE`. This Boolean value is extremely important as it compels VLOOKUP to search exclusively for **exact matches**. If `TRUE` or omitted (default behavior), the function assumes the lookup column is sorted and may return the closest approximate match, which is rarely desirable when dealing with structured identifier data like names or unique IDs. For reliable, precise data retrieval, especially in cases involving multiple returned values, setting this argument to `FALSE` is standard operating procedure.

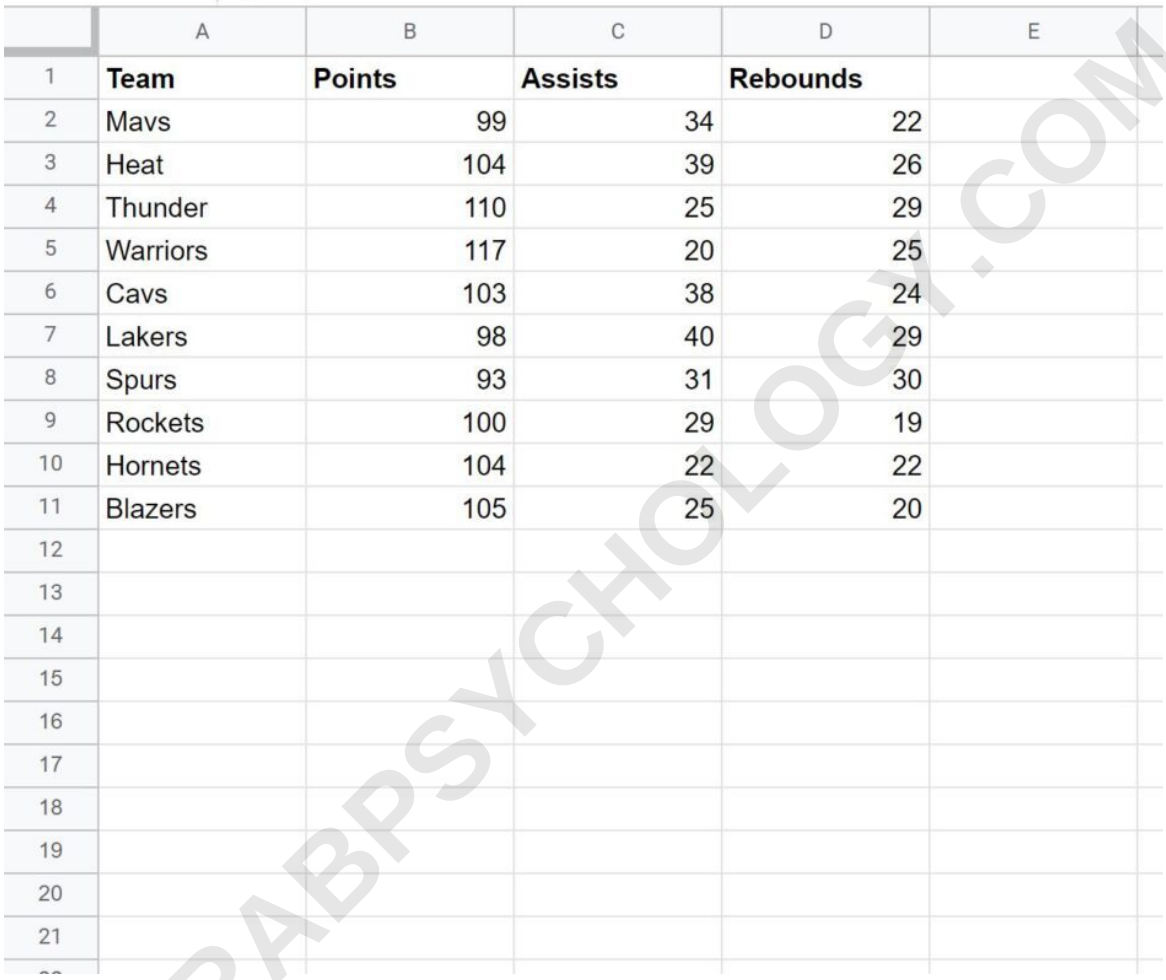
Step-by-Step Example: Implementing the Multi-Column VLOOKUP

To illustrate the practical application of this complex lookup technique, consider a typical scenario involving sports statistics. Suppose we maintain a dataset in **Google Sheets** that compiles essential performance metrics for various basketball teams over a season. Our goal is to quickly retrieve the points and rebounds for a specific team name without executing two separate **VLOOKUP** formulas.

We begin with the foundational dataset structure. This structure must be clear, with the unique identifier (Team Name) located in the leftmost column of the range we intend to search. This setup

is crucial because the **VLOOKUP** function is fundamentally restricted to searching only the first column of its input array.

Suppose we have the following dataset in **Google Sheets** that shows information about various basketball teams. The dataset spans columns A through D, covering Team, Points, Assists, and Rebounds, respectively:



	A	B	C	D	E
1	**Team**	**Points**	**Assists**	**Rebounds**	
2	Mavs	99	34	22	
3	Heat	104	39	26	
4	Thunder	110	25	29	
5	Warriors	117	20	25	
6	Cavs	103	38	24	
7	Lakers	98	40	29	
8	Spurs	93	31	30	
9	Rockets	100	29	19	
10	Hornets	104	22	22	
11	Blazers	105	25	20	
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					

If we want to search for the team "Blazers" (whose name is entered into cell A14) and return their corresponding Points (Column B, index 2) and Rebounds (Column D, index 4), we apply the combined formula. The range is `A2:D11`, and the indices are `{2, 4}`. We place this entire formula into a single output cell, such as B14, and the results will automatically spill into C14.

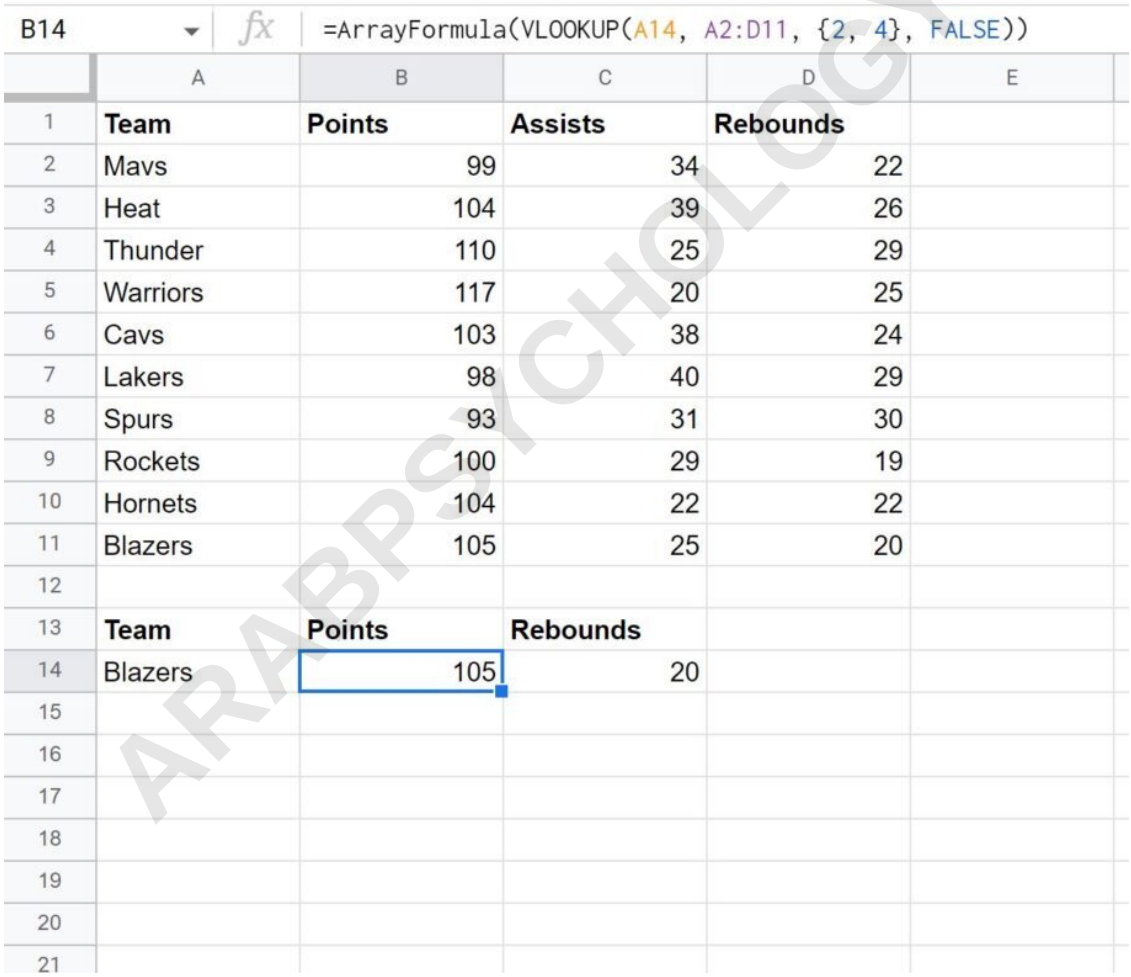
We can use the following formula structure combined with ArrayFormula to look up the team in column A and return the corresponding values for points and rebounds:

=ArrayFormula(VLOOKUP(A14, A2:D11, {2, 4}, FALSE))

Analyzing the Example Results and Data Structure

Once the formula is correctly entered into cell B14 (remembering to use Ctrl+Shift+Enter or simply typing `=ArrayFormula(...)` if using newer Google Sheets features that auto-detect array outputs), the output will be displayed immediately. The values are returned horizontally, occupying B14 and C14, corresponding to the Points and Rebounds of the "Blazers" team, respectively. This confirms that the combined **VLOOKUP** and **ArrayFormula** successfully executed a lookup operation that resulted in a multi-column output from a single formula entry point.

The following screenshot demonstrates the practical application of this formula. Notice that the formula is only physically present in the first cell of the output range (B14), but the resulting values occupy both B14 and C14:



	A	B	C	D	E
1	**Team**	**Points**	**Assists**	**Rebounds**	
2	Mavs	99	34	22	
3	Heat	104	39	26	
4	Thunder	110	25	29	
5	Warriors	117	20	25	
6	Cavs	103	38	24	
7	Lakers	98	40	29	
8	Spurs	93	31	30	
9	Rockets	100	29	19	
10	Hornets	104	22	22	
11	Blazers	105	25	20	
12					
13	**Team**	**Points**	**Rebounds**		
14	Blazers	105	20		
15					
16					
17					
18					
19					
20					
21					

The successful execution of the **VLOOKUP** formula returns the exact values found in the points and rebounds columns for the specific row where the team name precisely matches "Blazers." This dynamic linkage is the cornerstone of efficient data reporting. Moreover, one of the primary advantages of utilizing this formula structure is its inherent dynamism. If the team name input in

cell **A14** is modified, the formula automatically recalculates and pulls the corresponding points and rebounds values for the newly specified team, without requiring any changes to the formula structure itself.

Important Considerations: Exact Matches and Dynamic Range Updates

A critical feature embedded in the formula is the `FALSE` argument, which enforces an exact match lookup. Without this argument, **VLOOKUP** defaults to `TRUE`, which is designed for looking up values within sorted ranges and finding the nearest match if an exact one is not present. In data reporting where precision is necessary--such as matching unique team names--using `FALSE` ensures that errors are returned if the team name does not exist, preventing incorrect data association.

Furthermore, maintaining the dynamic nature of this lookup requires attention to the input cell **A14**. For instance, if we update the team name in cell **A14** from "Blazers" to "Warriors," the entire lookup calculation instantaneously updates, reflecting the new statistics. This immediate recalculation provides users with a powerful, dynamic lookup tool for querying large datasets rapidly.

For example, suppose we change the team name in cell **A14** to "Warriors" to query their statistics:

	A	B	C	D	E
1	**Team**	**Points**	**Assists**	**Rebounds**	
2	Mavs	99	34	22	
3	Heat	104	39	26	
4	Thunder	110	25	29	
5	Warriors	117	20	25	
6	Cavs	103	38	24	
7	Lakers	98	40	29	
8	Spurs	93	31	30	
9	Rockets	100	29	19	
10	Hornets	104	22	22	
11	Blazers	105	25	20	
12					
13	**Team**	**Points**	**Rebounds**		
14	Warriors	117	25		
15					
16					
17					
18					
19					
20					
21					

Upon this modification, the **VLOOKUP** formula immediately returns the values from the points and rebounds columns corresponding to the row where the team name contains the exact text "Warriors." This demonstrates the formula's utility as a dynamic data extraction tool, allowing for rapid querying of the source data simply by changing a single input cell.

Alternative Advanced Lookup Methods

While the **VLOOKUP** and ArrayFormula combination is effective, it is essential to recognize that it remains a workaround built around a function inherently limited to vertical searching. For situations demanding greater flexibility, particularly when the lookup column is not the leftmost column, or when complex criteria filtering is necessary, two other methods stand out: the **INDEX and MATCH** pair, and the **QUERY function**.

The **INDEX and MATCH** method is often preferred by spreadsheet experts because it eliminates the requirement for the lookup column to be the first in the range. **MATCH** finds the position of the lookup value, and **INDEX** returns the value at that position in a specified column. To return multiple columns, one would simply nest the **MATCH** function inside multiple **INDEX** functions, or use the

ArrayFormula wrapper alongside **INDEX and MATCH**, similar to the **VLOOKUP** technique, to define an array of desired result columns.

The QUERY function, unique to **Google Sheets**, offers the most flexible and SQL-like approach to data retrieval. Users can write simple queries (e.g., `SELECT B, D WHERE A = 'Blazers'`) to pull specific columns based on specific criteria. The **QUERY function** inherently supports multi-column outputs and conditional filtering, often resulting in cleaner, more readable formulas than complex nested lookups. While the **VLOOKUP** array method is quick for simple lookups, **QUERY** is the superior choice for high-volume, criteria-based, multi-column data extraction.