

How to Easily Use XLOOKUP in Google Sheets: A Step-by-Step Guide

Authored by
stats writer

November 30, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Use XLOOKUP in Google Sheets: A Step-by-Step Guide*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=102660>

The XLOOKUP function represents a significant advancement in data querying capabilities within modern spreadsheet applications, including Google Sheets. This versatile tool is designed to search for a specific value--the **search key**--within a designated lookup range and subsequently retrieve a corresponding value from a separate result range. It addresses many of the inherent limitations found in its predecessors, the legacy VLOOKUP and HLOOKUP functions, establishing itself as the preferred method for complex data lookups.

Unlike VLOOKUP, which is restricted to vertical searches starting from the leftmost column, XLOOKUP offers genuine flexibility. It can perform searches both vertically and horizontally across data sets, eliminating the need for separate horizontal and vertical lookup functions. Furthermore, it inherently supports returning values from columns to the left of the lookup column, a feature that required complex workarounds using INDEX and MATCH functions previously. This enhancement greatly simplifies formula construction and improves overall sheet efficiency.

To leverage the power of XLOOKUP, the user provides three mandatory arguments: the value being sought (the **search key**), the array or range where the search takes place (the **lookup range**), and the array or range containing the desired output (the **result range**). Additionally, XLOOKUP includes powerful optional arguments that allow users to define whether the search requires an exact or approximate match, specify a value to return if no match is found, and even dictate the search direction (starting from the first item or the last). Understanding these features is essential for becoming proficient in data analysis using Google Sheets.

Understanding the Full XLOOKUP Syntax

The primary purpose of the **XLOOKUP** function in Google Sheets is to efficiently find a specified piece of data within a column or row and return associated data from another location. While incredibly powerful, its basic implementation is straightforward, requiring only three key parameters.

The complete syntax for the **XLOOKUP** function, including its optional components, is defined as follows:

=XLOOKUP(search_key, lookup_range, result_range, , ,)

The formula requires three mandatory arguments that dictate the fundamental operation, followed by three optional arguments that control handling errors, specifying match types, and determining search direction.

search_key: This is the critical value or cell reference that the function must locate. It represents the item you are attempting to look up in the dataset.

lookup_range: This defines the single row or column where the **search_key** is expected to reside.

It is crucial that this range contains the precise data you are searching for.

result_range: This is the range (of equal size to the lookup range) from which the corresponding result will be returned. The function uses the position of the found **search_key** to determine which value to pull from this range.

We will first explore the practical application of the basic three-argument function before delving into the optional parameters.

Example 1: Performing a Basic Exact Match Lookup

To illustrate the efficiency of XLOOKUP, let us consider a practical scenario involving a dataset of basketball players. This dataset, displayed below, contains player names, their affiliated teams, and the total points they scored. Our goal is to quickly retrieve the point total for a specific team listed in our search field.

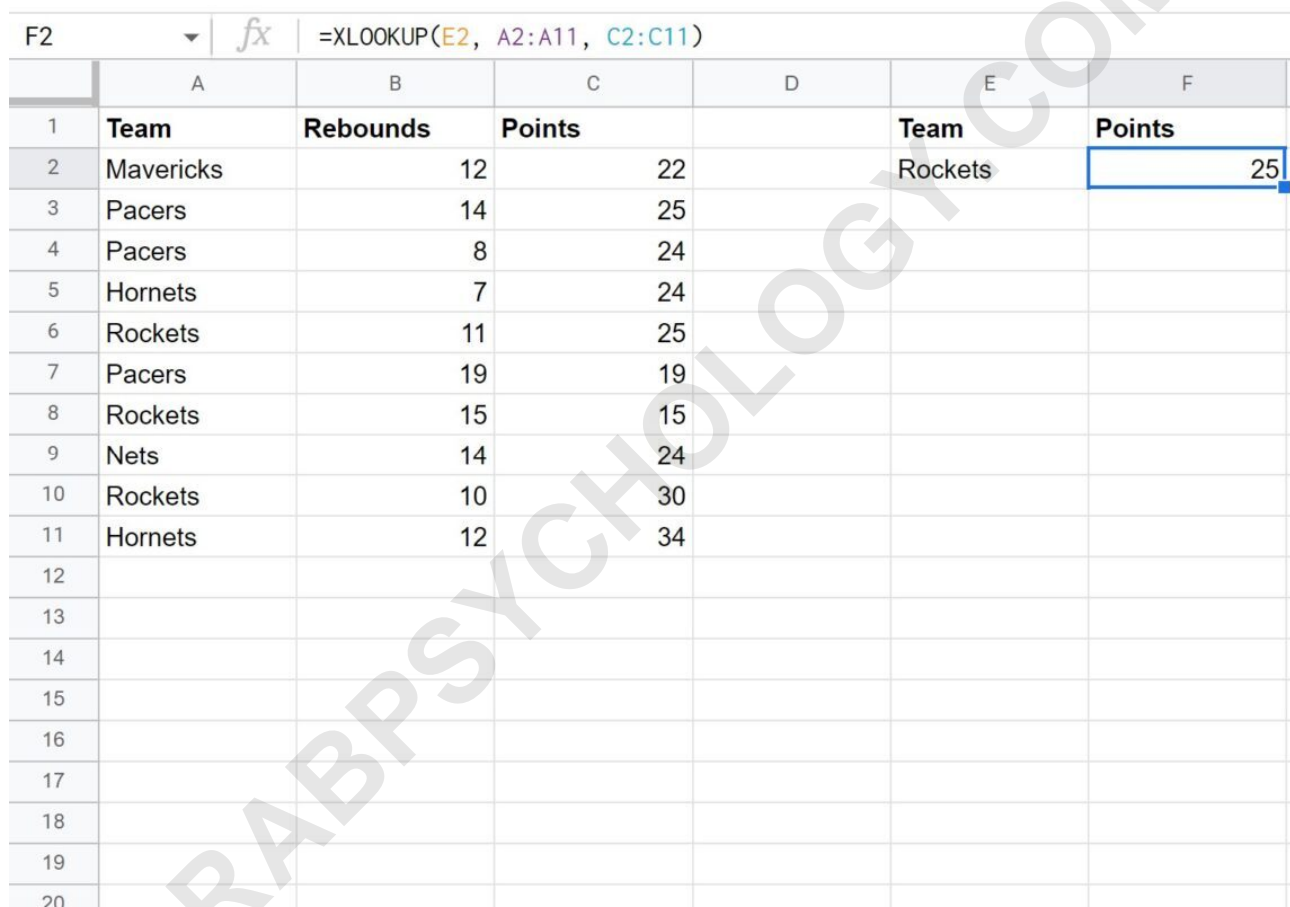
	A	B	C	D	
1	Team	Rebounds	Points		
2	Mavericks	12	22		
3	Pacers	14	25		
4	Pacers	8	24		
5	Hornets	7	24		
6	Rockets	11	25		
7	Pacers	19	19		
8	Rockets	15	15		
9	Nets	14	24		
10	Rockets	10	30		
11	Hornets	12	34		
12					
13					
14					
15					
16					
17					
18					
19					

In this example, the raw data spans columns A through C (A2:C11). We wish to look up the team name provided in cell **E2** (our **search_key**) within the Team column (**A2:A11**, the **lookup_range**), and retrieve the corresponding value from the Points column (**C2:C11**, the **result_range**). Since we are omitting the optional arguments, XLOOKUP defaults to an exact match.

The formula used in cell F2 to perform this operation is structured as follows:

=XLOOKUP(E2, A2:A11, C2:C11)

Executing this formula successfully finds the team name "Rockets" in column A. It then navigates to the corresponding row in column C and extracts the associated point total, which is the desired outcome of the lookup operation. The result is immediately visible in the cell where the formula is entered, as demonstrated in the following visual representation:



	A	B	C	D	E	F
1	Team	Rebounds	Points		Team	Points
2	Mavericks	12	22		Rockets	25
3	Pacers	14	25			
4	Pacers	8	24			
5	Hornets	7	24			
6	Rockets	11	25			
7	Pacers	19	19			
8	Rockets	15	15			
9	Nets	14	24			
10	Rockets	10	30			
11	Hornets	12	34			
12						
13						
14						
15						
16						
17						
18						
19						
20						

The power of this simple usage lies in its ability to connect data across disparate columns without relying on sorted data or worrying about the return column's position relative to the lookup column, which were significant drawbacks of older lookup methods.

Handling Missing Values with the `if_not_found` Argument

A significant enhancement provided by `XLOOKUP` is the built-in mechanism for handling lookup failures. Traditionally, if a lookup function like `VLOOKUP` could not locate the **search_key**, it would return the standard error message `#N/A`. While functional, these error messages can disrupt data

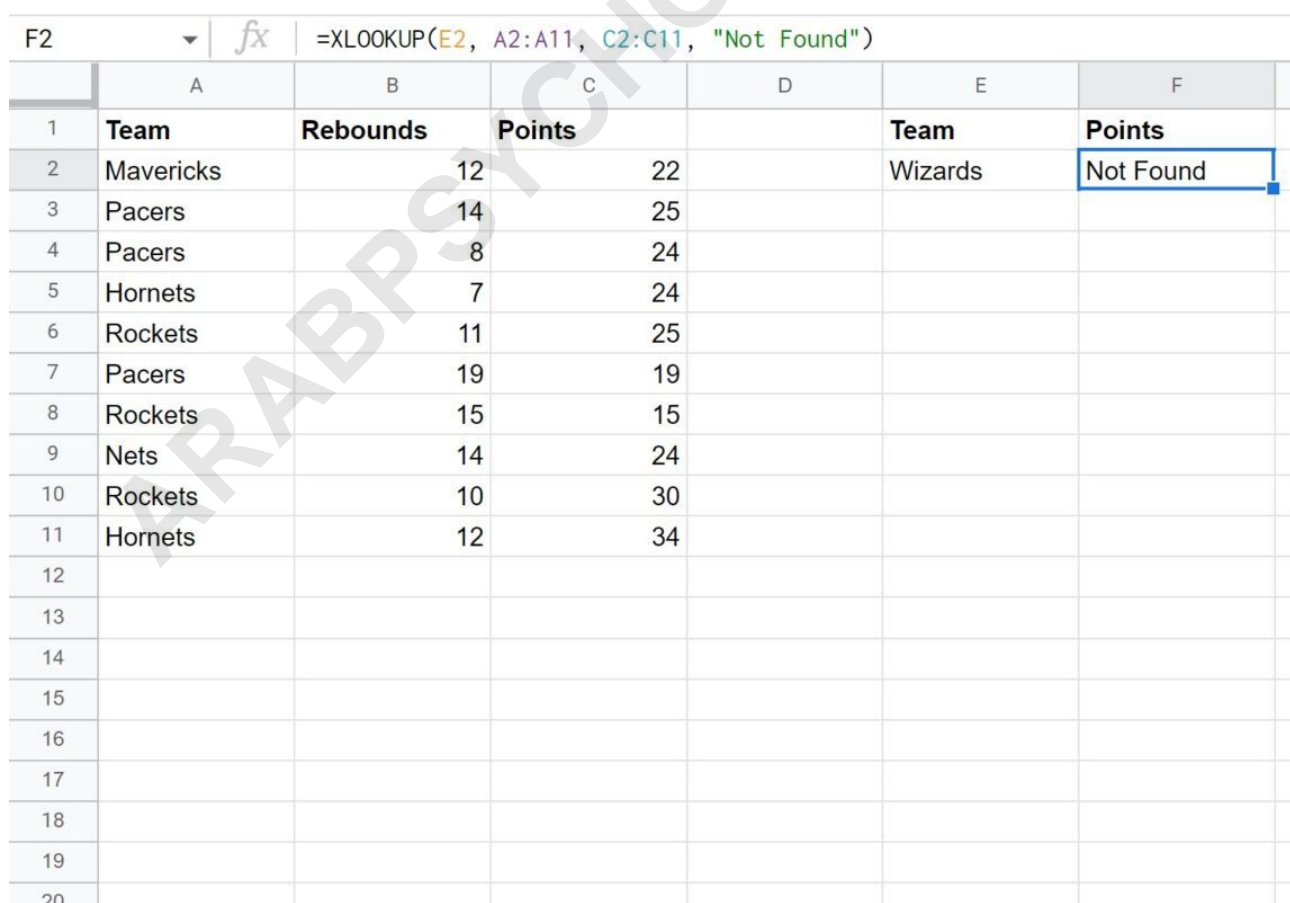
aggregation and look unprofessional in reports.

The XLOOKUP function includes a fourth optional argument, , which allows the user to specify a custom message or value to be returned instead of the default error. This eliminates the common need to wrap the lookup function inside an IFERROR function, streamlining the formula structure considerably. If this fourth argument is omitted, the function defaults back to returning the #N/A error when a match is not found.

Consider the scenario where we search for a team name, "Wizards," that does not exist within our dataset. To provide a clearer result than a simple error code, we can modify our previous formula by adding the text string "Not Found" as the fourth argument. Text strings must be enclosed in quotation marks, as shown below:

=XLOOKUP(E2, A2:A11, C2:C11, "Not Found")

The application of this improved formula is shown in the following screenshot, where the input in cell E2 is now "Wizards". Since "Wizards" does not exist in column A, the custom message "Not Found" is seamlessly returned, improving user experience and data hygiene:



	A	B	C	D	E	F
1	Team	Rebounds	Points		Team	Points
2	Mavericks	12	22		Wizards	Not Found
3	Pacers	14	25			
4	Pacers	8	24			
5	Hornets	7	24			
6	Rockets	11	25			
7	Pacers	19	19			
8	Rockets	15	15			
9	Nets	14	24			
10	Rockets	10	30			
11	Hornets	12	34			
12						
13						
14						
15						
16						
17						
18						
19						
20						

Controlling Match Behavior with the `match_mode` Argument

The fifth optional argument, `match_mode`, is critical for defining how the `XLOOKUP` function should handle comparisons between the `search_key` and the values in the `lookup_range`. While the default setting performs an exact match, the flexibility of this argument allows for approximate matches, which are often necessary when dealing with numerical thresholds or ranges.

The argument accepts one of four integer values, each defining a specific type of comparison:

0 (Default): Exact match. This requires the `search_key` to be identical to a value in the `lookup_range`. If no exact match is found, the result depends on the argument (or defaults to `#N/A`).

-1: Exact match or next smallest item. If an exact match is not found, the function returns the value corresponding to the next item that is smaller than the `search_key`. This is highly useful for tiered pricing tables or progressive tax calculations.

1: Exact match or next largest item. If an exact match is not found, the function returns the value corresponding to the next item that is larger than the `search_key`.

2: Wildcard match. This enables the use of special characters (`*`, `?`, `~`) within the `search_key` to find partial matches. For instance, using `"Rock*"` would match "Rockets," "Rockies," or "Rocky Mountains."

When using the approximate match options (-1 or 1), it is highly recommended, although not strictly enforced by [Google Sheets](#), that the `lookup_range` be sorted ascendingly for option -1 and descendingly for option 1, to ensure predictable and accurate results, mimicking the behavior of classic lookup functions.

The ability to perform native wildcard searches (mode 2) represents another substantial improvement over older functions, which often required complex array formulas or text functions to achieve the same result. This feature makes finding patterns or partial data extremely efficient.

Defining Search Direction with the `search_mode` Argument

The final optional argument, `search_mode`, controls the order in which the `lookup_range` is traversed. By default, `XLOOKUP` begins its search at the top of the range (or the left side, if horizontal) and moves downward (or rightward). However, there are scenarios, particularly when dealing with recent transactions or the last entry in a ledger, where starting the search from the bottom is necessary.

The argument provides four options for controlling the search process:

1 (Default): Search first-to-last. This is the standard vertical search, starting from the first cell of

the range (A2) and moving towards the last (A11). This is suitable for most chronological or standard lookups.

-1: Search last-to-first. This allows the user to quickly find the most recent occurrence of a duplicated value. For instance, if a player name appears multiple times, mode -1 will return the data corresponding to their last appearance in the list.

2: Binary search (Ascending order required). This mode is optimized for speed when dealing with extremely large, sorted datasets. If the data is not correctly sorted in ascending order, the function will likely return an incorrect result.

-2: Binary search (Descending order required). Similar to mode 2, this is a highly optimized search method that requires the data in the **lookup_range** to be sorted in descending order for accurate and rapid execution.

The flexibility provided by the search direction modes drastically simplifies tasks that previously required complex combinations of functions like `LARGE` or `ARRAYFORMULA` to find the nth occurrence or the last occurrence of a value. By integrating this functionality directly into the core lookup function, [Google Sheets](#) users can write far cleaner and more maintainable formulas.

Comparison: XLOOKUP vs. VLOOKUP and HLOOKUP

While `VLOOKUP` and `HLOOKUP` remain available in [Google Sheets](#) for backward compatibility, understanding why `XLOOKUP` is superior is key to modern spreadsheet mastery. `VLOOKUP` requires the result column to be specified by an index number (e.g., column 3), which makes the formula vulnerable to errors if columns are inserted or deleted. `XLOOKUP`, however, uses direct range references (e.g., C2:C11), meaning the formula adjusts automatically if the sheet structure changes.

Furthermore, `VLOOKUP` is inherently limited to searching left-to-right. If the value you need to retrieve is located in a column to the left of your lookup column, `VLOOKUP` cannot handle it without creating complex array workarounds. `XLOOKUP` completely eliminates this limitation, allowing two independent range selections for the lookup and result columns, regardless of their relative position.

The error handling is also significantly improved. Where `VLOOKUP` required nesting within `IFERROR` to handle missing values, `XLOOKUP` integrates this functionality directly via the argument. Similarly, the ability to search reverse (last-to-first) using the parameter is a native feature in `XLOOKUP` that would be challenging to replicate efficiently using legacy functions.

Conclusion: Adopting XLOOKUP for Modern Data Workflows

The adoption of `XLOOKUP` marks a new standard for lookup operations in [Google Sheets](#). Its comprehensive feature set—including bidirectional search, robust error handling, flexible match

modes, and control over search direction--solidifies its role as a vastly superior tool compared to the traditional VLOOKUP and HLOOKUP functions.

By mastering the mandatory arguments (**search_key**, **lookup_range**, **result_range**) and leveraging the optional arguments (**if_not_found**, **match_mode**, **search_mode**), users can create dynamic, reliable, and easily maintainable formulas. This functionality is essential for anyone performing serious data analysis or complex reporting within the Google Sheets environment. We encourage users transitioning from older systems to integrate **XLOOKUP** immediately to benefit from its improved design and reduced error probability.

For a complete and authoritative reference on all arguments and potential usage scenarios, always consult the official documentation for the **XLOOKUP** function, which provides the most current information regarding its implementation and capabilities.

ARABPSYCHOLOGY.COM