How to Perform Excel Lookups with Wildcards Using XLOOKUP

Authored by

stats writer

January 3, 2026

The **XLOOKUP** function is a formidable tool within **Excel**, designed to supersede older lookup functions by offering greater flexibility and efficiency. One of its most powerful capabilities is the ability to perform lookups using **wildcard** characters. This feature is particularly valuable when the exact spelling or complete identifier of the data you are searching for is unknown, or when you only need to match a specific pattern within a larger text string.

To successfully implement **XLOOKUP** with wildcards, you must utilize specific characters, primarily the asterisk (`*`), which serves as a catch-all for any sequence of characters. By correctly integrating this **wildcard** into your `lookup_value` argument and adjusting a crucial setting within the function's arguments, **XLOOKUP** can search for values that adhere to the specified criteria pattern rather than demanding an **exact match**. This methodology drastically accelerates the process of finding relevant data points, making it an indispensable technique for handling messy or incomplete datasets.

Understanding the Power of XLOOKUP

Before diving into the complexities of partial matching, it is essential to understand why **XLOOKUP** is the preferred function for modern **Excel** users. Unlike its predecessor, **VLOOKUP**, which was limited to searching only in the first column of a range, **XLOOKUP** allows flexible column referencing for both the search array and the return array. Furthermore, **XLOOKUP** natively supports searching in both directions (top-down or bottom-up) and provides integrated error handling, eliminating the need for cumbersome wrapper functions like `IFERROR`.

The core utility of **XLOOKUP** lies in its robust optional arguments, which control how the lookup is performed. These arguments govern actions such as defining the value returned if no match is found (`if_not_found`) and determining the method of searching (`search_mode`). Most importantly for our purpose, the `match_mode` argument controls whether the function requires an exact match, an approximate match, or, crucially, a **wildcard** character match. Mastering these arguments is key to unlocking advanced search capabilities beyond simple one-to-one lookups.

When dealing with large volumes of data where entries might contain varying descriptions or identifiers--such as product names with model numbers, or user inputs with extraneous characters--relying solely on exact matches is inefficient. The ability of **XLOOKUP** to interpret specific character patterns using wildcards transforms it from a simple retrieval tool into a powerful, pattern-recognition engine, ensuring high fidelity results even when data quality is inconsistent.

The Role of Wildcards in Pattern Matching

Pattern matching using **wildcard** characters is a fundamental concept in data search and retrieval. In **Excel**, two primary wildcards are recognized: the asterisk (`*`) and the question mark (`?`). The

asterisk is the most commonly used, representing any sequence of characters of any length (including zero characters). The question mark, conversely, represents any single character.

When incorporating these symbols into a search formula, they act as placeholders. For instance, searching for `"B*k"` would match "Book," "Break," "Bank," or "Balk," as the asterisk substitutes the internal characters. If you were to search for `"B?g"`, it would match "Big," "Bag," or "Bug," but not "Blog" (which requires two intervening characters). Effective use of wildcards depends entirely on understanding which symbol best represents the unknown component of your lookup value.

In the context of **XLOOKUP**, wildcards must be explicitly concatenated with the search string using the ampersand (`&`) operator. This combination creates a dynamic search criterion. If you are searching for a value that simply **contains** a specific text fragment (e.g., finding all entries containing "Sales"), you must enclose the fragment with asterisks (`"*"&"Sales"&"*"`). This ensures the function looks for the fragment regardless of what precedes or follows it in the target cell.

Syntax Breakdown: Integrating Wildcards into XLOOKUP

To successfully execute a **wildcard** lookup using **XLOOKUP**, two critical steps are necessary: first, constructing the `lookup_value` using string concatenation and wildcards; and second, setting the fifth argument, `match_mode`, to the value `2`. This value explicitly tells **Excel** that the search string contains pattern-matching characters.

The standard syntax for a typical **XLOOKUP** with wildcards for partial text matching looks like this:

=XLOOKUP("*"&E1&"*", A2:A11, B2:B11,,2)

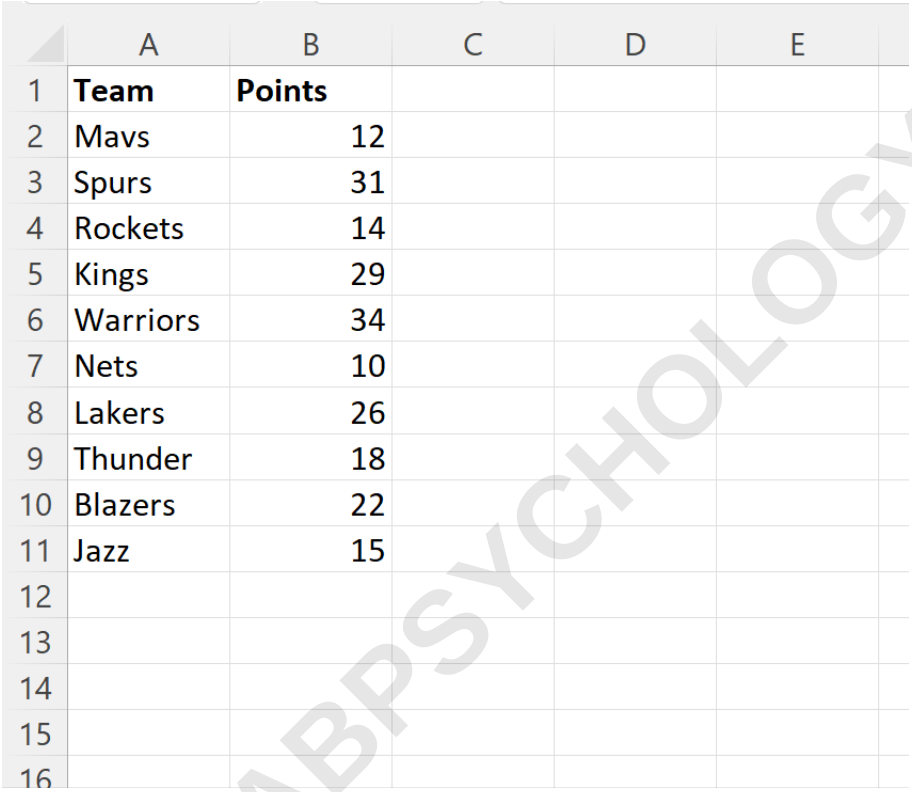
In this construction, the text input in cell `E1` is surrounded by the asterisk wildcard characters. The result is a dynamic search term, such as `"*ock*"` if `E1` contains "ock." This particular formula is designed to find the first value in the range `A2:A11` that **contains** the partial text specified in cell `E1`, subsequently returning the corresponding value from the range `B2:B11`. Note the use of `,,2` at the end: the empty comma represents the optional `if_not_found` argument, which we are skipping, and the `2` specifically sets the **match_mode** to enable **wildcard** matching.

Understanding the placement of the wildcards is essential for controlling the search: if you use `E1&"*"`, the function looks for values that **start** with the text in `E1`. If you use `"*"&E1`, it searches for values that **end** with the text in `E1`. The comprehensive use of surrounding asterisks (as shown in the formula above) provides the greatest flexibility by locating the text fragment anywhere within the cell content.

Practical Example: Implementing Partial Text Match

To illustrate the necessity of using wildcards in **XLOOKUP**, consider a scenario involving a dataset detailing basketball player statistics. Suppose we have records containing the Player's Team and their corresponding Points scored. We want to look up the points for a specific team, but perhaps we only know a small part of the team's name.

Suppose we have the following dataset structure in **Excel**, containing information about points scored by various basketball players:



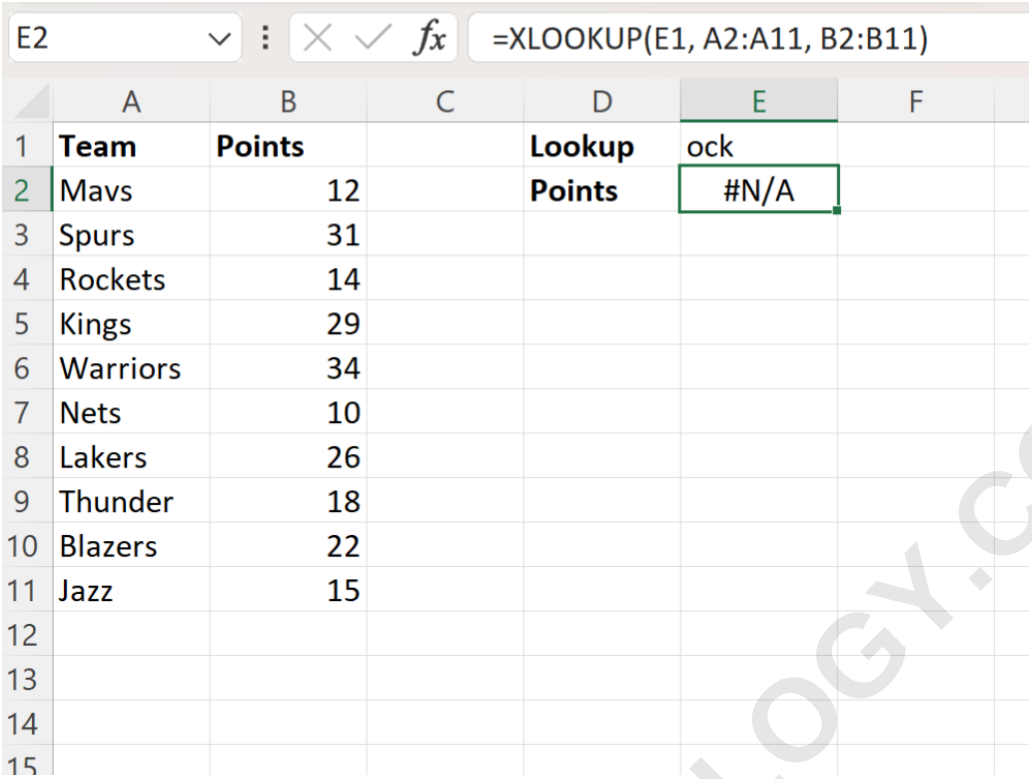
	A	B	C	D	E
1	**Team**	**Points**			
2	Mavs	12			
3	Spurs	31			
4	Rockets	14			
5	Kings	29			
6	Warriors	34			
7	Nets	10			
8	Lakers	26			
9	Thunder	18			
10	Blazers	22			
11	Jazz	15			
12					
13					
14					
15					
16					

If we wish to look up the value in the **Team** column using the partial text "ock" (stored in cell `E1`) and return the corresponding value from the **Points** column, a standard, non-wildcard **XLOOKUP** will fail, as it expects "ock" to be the exact content of a cell in the lookup array.

Let us first attempt to use a basic **XLOOKUP** without the necessary wildcard configuration:

=XLOOKUP(E1, A2:A11, B2:B11)

The following screenshot demonstrates the outcome of this incorrect approach. Because we are asking **XLOOKUP** to find a cell that contains only the text "ock" exactly, and no such cell exists in the Team column, the function cannot find a match.



	A	B	C	D	E	F
1	**Team**	**Points**		**Lookup**	ock	
2	Mavs	12		**Points**	#N/A	
3	Spurs	31				
4	Rockets	14				
5	Kings	29				
6	Warriors	34				
7	Nets	10				
8	Lakers	26				
9	Thunder	18				
10	Blazers	22				
11	Jazz	15				
12						
13						
14						
15						

As illustrated, the formula returns `#N/A`. This error occurs because the default behavior of **XLOOKUP** is to look for an **exact match** (where `match_mode` is `0`), and there is no value in the **Team** column that is strictly equal to "ock." This situation highlights the limitation of standard lookups when dealing with partial text searches.

Analyzing the Formula Arguments for Wildcard Success

To correct the previous error and successfully retrieve the desired points using a partial text match, we must employ the **wildcard** syntax and modify the **match_mode** argument. The successful formula structure requires careful management of the function's five core arguments.

The revised formula utilizes concatenation and sets the `match_mode` to `2`, which enables the **wildcard** character match behavior:

=XLOOKUP("*"&E1&"*", A2:A11, B2:B11,,2)

Let's break down the impact of each argument in this specific context:

lookup_value: `"*"&E1&"*"`. This is the search term, dynamically combining the string in E1 with preceding and succeeding asterisk wildcards. If `E1` holds "ock," the search term becomes `*ock*`, instructing **Excel** to find any cell containing "ock."

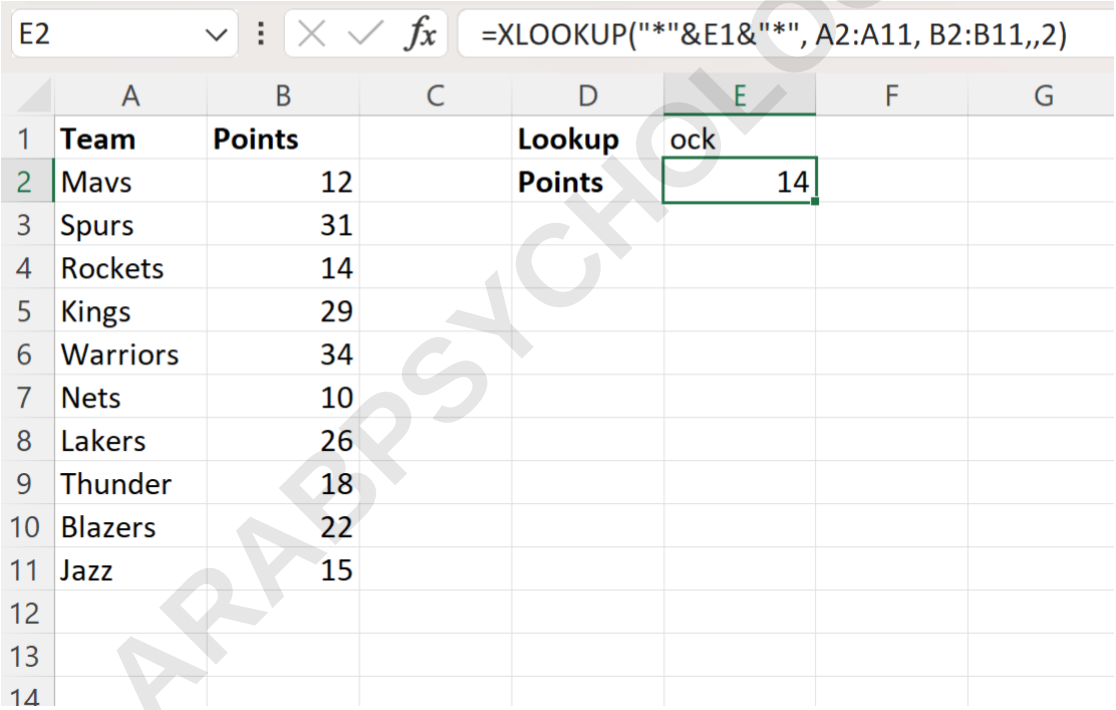
lookup_array: `A2:A11`. This defines the range where the function will search for the pattern. In our example, this is the **Team** column.

return_array: `B2:B11`. This is the range from which the corresponding value will be returned once a match is found in the `lookup_array`.

if_not_found: (Omitted, represented by the first extra comma). We skip this argument, meaning if no match is found, `#N/A` will be returned (as is the default behavior).

match_mode: `2`. This crucial argument activates the **wildcard** matching logic. Without `2`, the formula would still search for the literal string `*ock*` (including the asterisks) as an exact match, leading to an error.

The following screenshot demonstrates the successful execution of the formula using the correct **match_mode**:



The formula successfully returns a value of **14**. This value represents the score in the **Points** column for the first row that contains the text fragment "ock" in the **Team** column (the team "Rockers"). This confirms that **XLOOKUP** correctly interpreted the wildcard pattern and returned the corresponding result from the `return_array`.

Common Pitfalls and Troubleshooting

While using **XLOOKUP** with wildcards is highly effective, users frequently encounter issues that

prevent the formula from working as intended. The most common error is forgetting to set the `match_mode` argument to `2`. If this argument is omitted or set to `0` (exact match, the default), **Excel** will treat the asterisk (`*`) character literally, searching for a cell that actually contains an asterisk, rather than treating it as a placeholder.

Another common mistake involves the misuse of string concatenation. If the input text is hardcoded into the formula, ensure it is properly enclosed in quotation marks, and that the wildcard character is also enclosed in quotes and joined using the ampersand (`&`). For example, `=XLOOKUP("*JONES*", ...)` will work, but `=XLOOKUP("*", "JONES", "*", ...)` is syntactically incorrect. When referencing a cell (e.g., `E1`), the correct format remains `"*"&E1&"*"`.

Finally, remember that **XLOOKUP**, like its predecessors, returns only the **first match** it finds. If multiple teams in the dataset contain the text "ock" (e.g., "Rockers" and "Jockers"), the formula will stop at the first occurrence (A2:A11 is searched top-down by default) and return the associated point value for that first record. If you need to return multiple matches or list all matching values, you would need to combine **XLOOKUP** with more advanced array functions like `FILTER` or `TEXTJOIN`, which lies beyond the scope of a single lookup operation.

Conclusion: Enhancing Data Retrieval Efficiency

The integration of **wildcard** functionality into the **XLOOKUP** function provides a substantial upgrade to data retrieval capabilities in **Excel**. This methodology allows users to perform highly sophisticated pattern-based searches, overcoming the limitations imposed by requirements for perfect data cleanliness or exact matches.

By correctly manipulating the `lookup_value` using string concatenation and wildcards, and crucially, setting the `match_mode` argument to `2`, users gain the power to locate data based on partial identifiers, variable strings, or unknown sequences of characters. This technique is invaluable for professionals working with large, heterogeneous, or frequently updated datasets, significantly streamlining data analysis and reporting workflows.

Mastering this specific syntax for **XLOOKUP** ensures that your search operations are not only precise when an exact match is available, but also resilient and versatile when dealing with real-world data variability. For comprehensive technical details and further advanced applications, users should always consult the official documentation for the **XLOOKUP** function in **Excel**.