Learning XLOOKUP with Wildcards: An Excel Tutorial

Authored by

Mohammed looti

November 11, 2025

Why Traditional XLOOKUP Falls Short for Fuzzy Matching

The introduction of the **XLOOKUP** function revolutionized data handling within modern versions of **Excel**. As the designated successor to older, less flexible tools like VLOOKUP and HLOOKUP, **XLOOKUP** provides remarkable efficiency by allowing users to search for a specific value within one range and retrieve a corresponding result from any other column, irrespective of location. However, by default, the function operates under the principle of an **exact match**, which requires the lookup value to correspond perfectly, character-for-character, with an entry found in the lookup array. This precision is ideal for perfectly standardized datasets.

Despite the utility of exact matching, real-world data is rarely pristine. Analysts frequently encounter situations where they need to locate records based on incomplete information, misspellings, or simply a portion of a larger text string--a concept known as **partial text** searching. For instance, imagine needing to retrieve sales data for a vendor when only a keyword from their lengthy official name is available, or tracing an inventory item using only a few known characters of its product code. In these scenarios, relying solely on exact matching causes the lookup to fail, inevitably resulting in the frustrating **#N/A** error, signaling that a perfect match could not be located.

To overcome this significant limitation and transform **XLOOKUP** into a robust tool for handling less structured data, we must integrate the use of **wildcard characters**. Activating the designated wildcard match mode allows **XLOOKUP** to perform powerful fuzzy or partial lookups, dramatically increasing the function's adaptability and enhancing the overall analytical capability of the spreadsheet.

Mastering Excel's Wildcard Characters: Asterisk and Question Mark

Flexible searching in **Excel** is fundamentally enabled by two primary **wildcard characters**: the asterisk (`*`) and the question mark (`?`). A solid understanding of these symbols is essential for constructing effective partial-match formulas, not just within **XLOOKUP**, but across many of Excel's search and filter functions.

The **Asterisk (`*`)**: This versatile character acts as a placeholder for any sequence of characters, including a sequence of zero length. It is the key to performing true **partial text** searches. For example, searching for "Trans*" would match "Transport," "Transaction," or simply "Trans." Conversely, placing the asterisk on both sides, such as "*ship*", would locate "Relationship" or "Shipment."

The **Question Mark (`?`)**: This character represents a placeholder for exactly one single character. It is useful when the length of the string is known, but a specific character is variable or unknown. For example, searching for "R?n" would successfully find "Run" or "Ran," but it would intentionally exclude longer words like "Rain."

When preparing the lookup value for a partial match, these wildcards must be dynamically combined with the text fragment provided by the user (often referencing a cell). This combination is achieved through the use of the ampersand operator (`&`), which serves as the standard text concatenation tool in **Excel**. If the search fragment resides in cell F1, the required construction to search for that text anywhere within a target string would be written as `"*"&F1&"*"`. However, it is vital to remember that simply constructing this wildcard pattern is insufficient; the function needs an explicit instruction--a specific argument--to interpret this construction as a pattern rather than a literal string.

The Standard XLOOKUP Structure: A Review of Exact Matching

To fully appreciate the necessity of enabling the wildcard mode, we should first review the default behavior of the **XLOOKUP** function. The standard **syntax** relies on three mandatory arguments, which are typically followed by several optional parameters:

`lookup_value`: The specific item or text string being sought.
`lookup_array`: The single column or row range where the search is conducted.
`return_array`: The corresponding range containing the result to be returned upon finding a match.

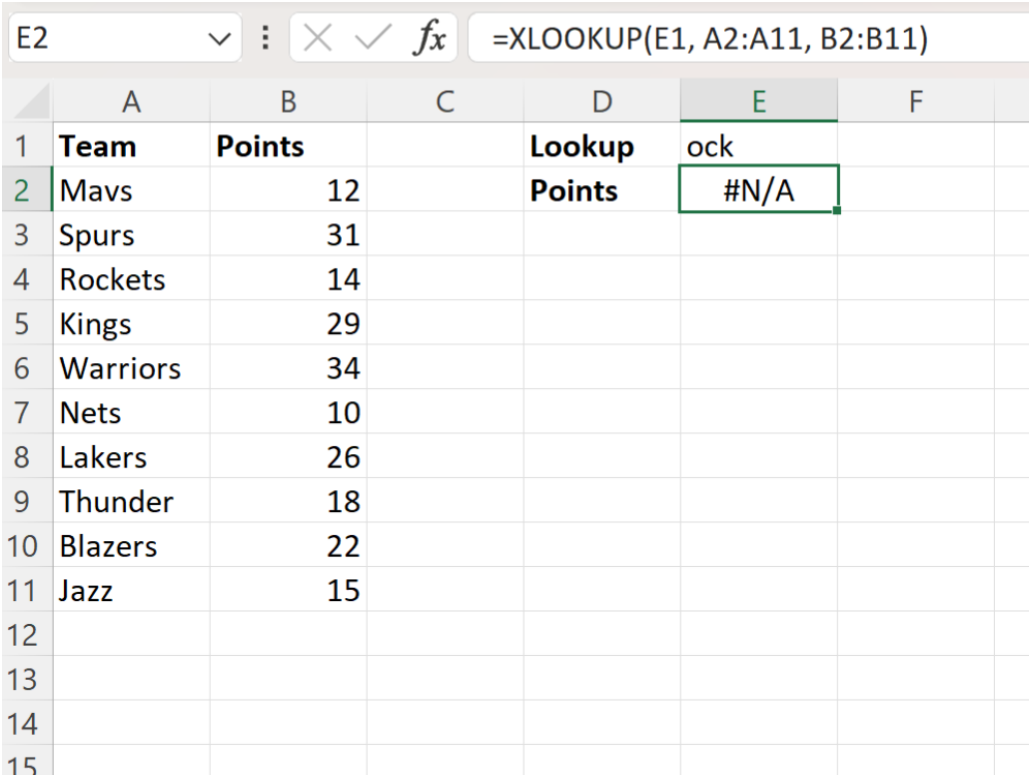
Consider a practical scenario involving a dataset that tracks points scored by various teams, and we are tasked with finding the points scored by a team based on only a few known characters of its name. The data structure is illustrated below, where we attempt to look up points using the partial identifier "ock."

	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 the **partial text** "ock" is entered into cell E1 and we attempt to retrieve the corresponding Points using the default, non-wildcard formula, the lookup fails entirely. This failure occurs because **XLOOKUP** defaults to an exact match (Match Mode 0), which means it searches for a cell that *is* literally "ock." The failing formula structure is shown here:

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

As demonstrated in the subsequent output, the standard formula returns the **#N/A** error. This result clearly highlights the critical need to modify the function's behavior, compelling **XLOOKUP** to search for entries that *contain* the substring "ock," rather than demanding an absolute match against the lookup value.



	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						

Activating Wildcard Search Mode: Modifying the Lookup Value and Match Mode

To successfully transition from an exact match to a robust **partial text** match, two fundamental modifications must be applied to the **XLOOKUP** function's **syntax**: defining the `lookup_value` as a pattern and setting the `match_mode` argument appropriately. These two steps work in concert to unlock the power of **wildcard characters**.

Firstly, the `lookup_value` argument must be restructured using concatenation. Assuming the search fragment is in cell E1 ("ock"), we must enclose this reference with asterisk wildcards (`*`) using the ampersand operator. This critical restructuring yields the search pattern `"*ock*"`, which dynamically instructs **Excel** to locate any string containing "ock" regardless of what characters precede or follow it. This technique makes the lookup value dynamic and flexible.

Secondly, we must address the optional fifth argument: the `match_mode`. By default, this argument is 0 (exact match). To enable the wildcard functionality, this argument must be explicitly set to the integer value 2. This setting tells **XLOOKUP** to interpret the special symbols (`*` and `?`) within the lookup value as patterns rather than literal text. It is essential to understand that without setting `match_mode` to 2, the function will search for the literal string `"*ock*"`, which will also result in an error, demonstrating that both the pattern construction and the mode activation are necessary components of the solution.

Step-by-Step Implementation: The Partial Text Lookup Formula

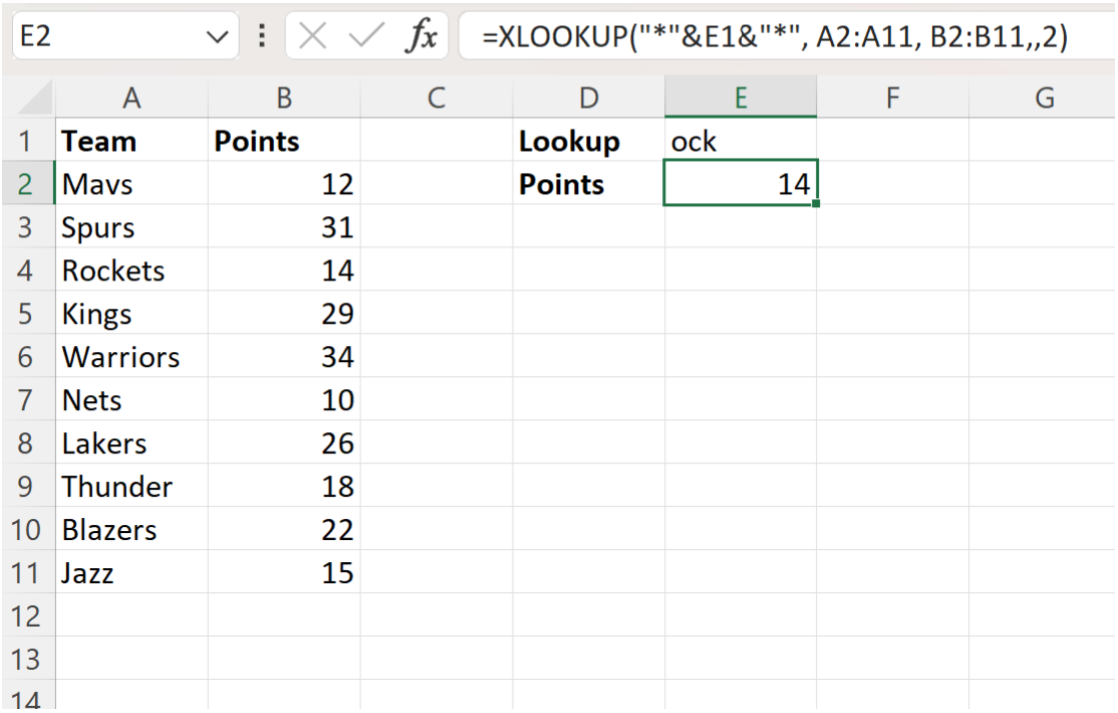
The resulting formula structure provides a comprehensive and robust solution for retrieving data based on substrings. This structure is designed to find the first occurrence within the specified range that matches the defined wildcard pattern and return the corresponding value from the return array.

Let us apply this specialized **XLOOKUP** formula to our existing basketball dataset. Our objective remains to pinpoint the points scored by the first team whose name includes the **partial text** "ock," which is currently stored in cell E1. We need to correctly define the arrays and crucially include the final argument to activate the correct matching behavior.

We input the required formula, ensuring that the lookup array (A2:A11, the Team column) and the return array (B2:B11, the Points column) accurately encompass the data. Notice the inclusion of the concatenated lookup value and the final argument, `2`, which signals the activation of the wildcard matching mode:

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

The following screenshot confirms the successful execution of this powerful partial search:



	A	B	C	D	E	F	G
1	**Team**	**Points**		**Lookup**	ock		
2	Mavs	12		**Points**	14		
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							

Upon calculation, the formula returns a precise value of **14**. This result corresponds to the row containing the team name "Rockets," which is the first entry encountered in the lookup range

(A2:A11) that successfully satisfies the wildcard pattern `"*ock*"`. This practical demonstration clearly establishes the immense benefit of combining **XLOOKUP** with **wildcard characters**, allowing the function to transcend the limitations of strict exact matching and efficiently locate data based on embedded substrings.

Dissecting the Wildcard Syntax and Argument Flow

For complete mastery of this technique, a detailed breakdown of the functional structure is essential, illustrating how each argument contributes specifically to the successful partial match using **wildcard characters**. The formula structure is:

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

`"*"&E1&"*"`: This expression defines the `lookup_value`. By dynamically joining the contents of E1 (e.g., "ock") with leading and trailing asterisks, it resolves into the search pattern `"*ock*"`. This ensures that the search term can be found anywhere within the target text string, facilitating a true **partial text** lookup.
`A2:A11`: This is the `lookup_array`. It explicitly defines the range (the Team names column) that **XLOOKUP** will scan to find the matching pattern.
`B2:B11`: This is the `return_array`. It designates the range (the Points column) from which the corresponding data point is retrieved once a match is identified in the lookup array.
`,,`: We intentionally omit the optional fourth argument, `if_not_found`, and move directly to the final, crucial argument.
`2`: This value sets the `match_mode`. Setting it to 2 is mandatory for activating the **wildcard characters** search functionality, ensuring that **Excel** correctly interprets the asterisk and question mark symbols as pattern matching operators, not literal characters.

This powerful methodology significantly expands the utility of **XLOOKUP**. It is an indispensable technique for analysts dealing with complex or "messy" data, addressing user input errors, or executing database searches where only fragments of the necessary information are readily available. Always remember to set the Match Mode to 2 when utilizing wildcards to ensure the **syntax** is correctly parsed by **Excel**.

Note: Comprehensive documentation for the **XLOOKUP** function and other lookup tools in **Excel** is available on the official Microsoft Support website.

Resources for Advanced Excel Lookup Techniques

For users seeking to deepen their understanding of advanced data manipulation and lookup operations within **Excel**, exploring related functions and techniques is highly recommended. These resources cover methods that extend beyond basic exact matching:

Exploring how to implement and leverage approximate matches within the **XLOOKUP** function.
A detailed comparison outlining the operational differences between VLOOKUP, INDEX/MATCH, and **XLOOKUP**.
Practical techniques for handling complex scenarios involving lookups based on multiple criteria in spreadsheet applications.