How to Find Partial Match in Two Columns in Excel

Authored by

stats writer

November 18, 2025

Introduction: Understanding Partial Matching in Data Analysis

In modern data analysis, a common requirement is to reconcile data sets where identifiers do not match exactly. This scenario often arises when attempting to link two tables where one uses full descriptors and the other uses abbreviations, nicknames, or slightly misspelled entries. Identifying a partial match is essential for tasks like data cleaning, merging data from disparate sources, or performing complex lookups where standardization is lacking. While exact matches are handled simply by functions like VLOOKUP, partial matching requires a strategic combination of functions and special characters known as wildcards.

The standard lookup functions in Excel are primarily designed to search for exact correspondences between values. However, by incorporating specific techniques, we can adapt these tools to perform flexible lookups. The most reliable and widely used method for achieving partial string matching involves augmenting the lookup value with the asterisk (*****) wildcard, which represents any sequence of characters, effectively turning an exact search into a contains-search.

This guide provides an expert walkthrough on how to leverage the power of the VLOOKUP function, enhanced with wildcards and error handling, to efficiently find and return partial matches between two distinct columns in your spreadsheet. Mastering this technique is fundamental for anyone working with real-world, messy data sets where complete standardization cannot be guaranteed.

The Core Excel Function: VLOOKUP and Wildcards

To successfully find a partial match, we rely on the fundamental structure of the VLOOKUP function, which is traditionally structured as `=VLOOKUP(lookup_value, table_array, col_index_num, range_lookup)`. The key innovation here lies in manipulating the `lookup_value` argument. Instead of searching for the exact content of cell B2, for example, we construct a search string that incorporates wildcards using concatenation operators.

The asterisk wildcard (*****) serves as a placeholder for zero or more characters. By placing this wildcard before and after the value we are searching for, we instruct Excel to look for any cell in the lookup range that **contains** the specified string, regardless of what precedes or follows it. For instance, if we are looking up "ABC," constructing the lookup value as `"*"&"ABC"&"*"` tells the function to accept "123ABC456," "ABC Company," or simply "ABC" as valid matches.

This powerful construction transforms VLOOKUP from a strict comparison tool into a flexible pattern-matching utility. When applied to finding partial matches between two columns, Column B contains the partial search term (e.g., "Rockets"), and Column A holds the complete list (e.g., "Houston Rockets"). By formatting the search term correctly, the function can successfully identify the connection and return the corresponding data, proving invaluable in large-scale reconciliation

projects.

Step-by-Step Formula Breakdown

The standardized formula structure used to execute a robust partial match lookup is shown below. This syntax is highly versatile and forms the basis for complex data comparisons across different sheets or workbooks in Excel. Understanding each component is vital for proper application and troubleshooting.

=IFERROR(VLOOKUP("*"&B2&"*", $A$2:$A$9, 1, 0), "")

Let us dissect the core lookup mechanism: `VLOOKUP("*"&B2&"*", $A$2:$A$9, 1, 0)`. The first argument, `"*"&B2&"*"`, is the dynamically generated **lookup value**. We use the ampersand (**&**) operator to concatenate the wildcard asterisk (*****), the content of the target cell (**B2**), and another asterisk. This ensures that the search term from B2 is treated as a substring that must exist somewhere within the cells of the lookup range.

The second argument, `$A$2:$A$9`, defines the **table array**. This is the range where the function will search for the partial match. It is crucial to use absolute references (the dollar signs) for this range. Using absolute references ensures that when the formula is dragged down to apply to subsequent rows, the lookup range remains fixed and consistent, preventing errors and incorrect results. The third argument, `1`, indicates that we want to return the value from the **first column** of the table array, which is necessary since we are attempting to return the value that matched partially.

Finally, the fourth argument, `0` (or **FALSE**), specifies that we require an **exact match** for the wildcard pattern. Although we are performing a partial match search, setting this argument to `0` or `FALSE` is mandatory when using VLOOKUP with wildcards to ensure the function executes the pattern recognition correctly. If a match is found using this pattern, the value from the specified column (Column 1 in this case) is returned.

Handling Errors with IFERROR

While the VLOOKUP function is effective for finding matches, it returns the standard `#N/A` error if no match is found for the specified lookup value or pattern. In a data reconciliation task, an `#N/A` error can be visually distracting and complicated to work with if you intend to perform further calculations or reporting on the resulting column.

To manage this outcome gracefully, we wrap the entire VLOOKUP formula within the IFERROR function. The IFERROR function takes two arguments: the value to check for an error, and the

value to return if an error is detected. In our case, the `value` is the VLOOKUP formula itself.

The completed formula, `=IFERROR(VLOOKUP(...), "")`, instructs Excel to first execute the partial match lookup. If the VLOOKUP successfully returns a match, that result is displayed. However, if VLOOKUP encounters an error--meaning no partial match was found--the IFERROR function intercepts the `#N/A` and returns the specified alternative, which is an empty string (`""`). This leaves the resulting cell blank, making the output cleaner and easier to analyze.

Practical Example: Matching Team Names

To illustrate the practical application of this technique, consider a scenario involving sports data. We have two separate lists of basketball team names. Column A contains the full, official names of the teams, while Column B contains abbreviated or nickname versions of the same teams. Our goal is to use the abbreviated name in Column B to find and retrieve the corresponding full name from Column A, demonstrating a successful partial match lookup.

Suppose we have the following initial data structure in our Excel sheet:

	A	B	C	D	E
1	**Full Team Name**	**Team Abbreviation**			
2	Dallas Mavericks	Rockets			
3	Cleveland Cavaliers	Spurs			
4	Houston Rockets	Lakers			
5	San Antonio Spurs	Miami			
6	Orlando Magic	Pacers			
7	Miami Heat	Lakers			
8	Indiana Pacers	Orlando			
9	Denver Nuggets	Jazz			
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					

The task is to populate Column C (labeled "Full Team Match") by checking if the abbreviated team name in B2 ("Rockets") is contained within any of the full team names listed in the range A2:A9.

This is a classic partial matching problem, ideal for the VLOOKUP with wildcards approach. We are effectively querying whether B2 is a **substring** of any entry in A2:A9.

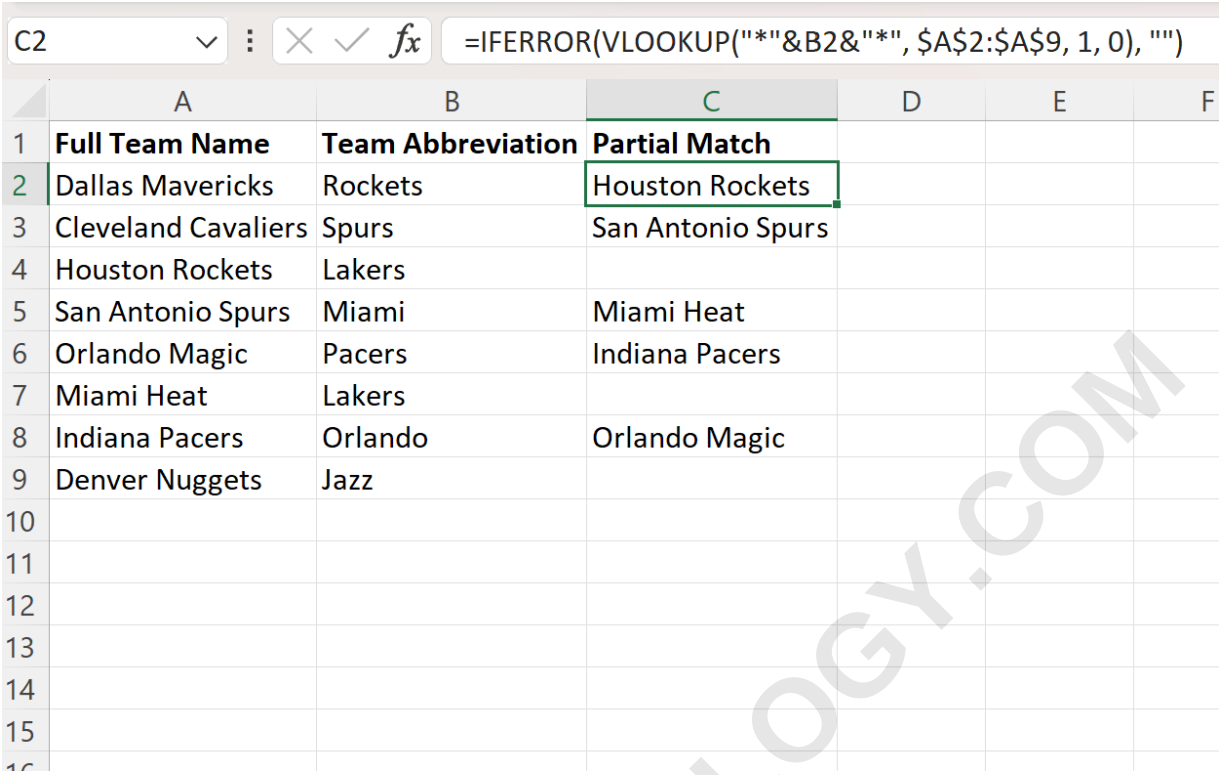
Applying the Formula: A Detailed Walkthrough

We begin the process by navigating to cell **C2**, which will house the formula responsible for checking the first abbreviated team name against the full list. Our objective is to search for the contents of cell B2 ("Rockets") within the range A2:A9. We must construct the lookup value to encapsulate B2 with wildcards, ensuring that if "Rockets" is found within "Houston Rockets," it returns the full name "Houston Rockets."

The formula typed into cell **C2** must utilize absolute references for the lookup array to facilitate easy copying. The structure is implemented exactly as planned, combining error handling with the pattern matching:

=IFERROR(VLOOKUP("*"&B2&"*", $A$2:$A$9, 1, 0), "")

After entering this formula into **C2**, the result immediately displays "Houston Rockets." This confirms that the search term "Rockets" successfully located its corresponding full name in Column A. The next crucial step is propagation: we must copy this formula down to the remaining cells in Column C (C3 through C9). This is efficiently done by clicking the fill handle (the small square at the bottom-right corner of cell C2) and dragging it downwards.



	A	B	C	D	E	F
1	**Full Team Name**	**Team Abbreviation**	**Partial Match**			
2	Dallas Mavericks	Rockets	Houston Rockets			
3	Cleveland Cavaliers	Spurs	San Antonio Spurs			
4	Houston Rockets	Lakers				
5	San Antonio Spurs	Miami	Miami Heat			
6	Orlando Magic	Pacers	Indiana Pacers			
7	Miami Heat	Lakers				
8	Indiana Pacers	Orlando	Orlando Magic			
9	Denver Nuggets	Jazz				
10						
11						
12						
13						
14						
15						

When dragging the formula, the relative reference **B2** automatically adjusts (to B3, B4, etc.), ensuring each row checks its corresponding abbreviated team name. Simultaneously, the absolute reference **$A$2:$A$9** remains constant, ensuring every lookup is performed against the complete list of full team names, thereby producing accurate and consistent results across the entire output column.

Interpreting the Results

The completed Column C now provides a clear overview of which abbreviated team names from Column B successfully found a partial match in the list of full names in Column A. If a partial match is established, the corresponding full name from Column A is returned in the result column. This provides immediate data linkage, confirming the relationship between the two pieces of information.

Conversely, if the VLOOKUP function fails to find the abbreviated term as a substring within any of the entries in the lookup array, it generates an error. Because the formula is wrapped in IFERROR, this `#N/A` error is suppressed, and a blank cell is returned, indicating that no connection could be made for that particular row.

Reviewing the key outcomes in the resulting column reveals specific insights:

The "Rockets" term had a partial match with "Houston Rockets," successfully linking the data.

The "Spurs" term, although abbreviated, had a partial match with "San Antonio Spurs," demonstrating the power of the wildcards.

The "Lakers" term did not find any partial match within the specified list (A2:A9), leading the IFERROR function to return a blank value, confirming its absence.

Alternative Methods for Partial Matching

While the VLOOKUP and wildcard combination is robust and standard, modern versions of Excel offer alternative functions, such as **XLOOKUP**, which can also be adapted for partial matching. However, XLOOKUP still requires the use of wildcards and an optional fourth argument set to `2` (which enables wildcard character match) to achieve the same result. The fundamental principle of concatenating the search term with asterisks remains essential, regardless of the lookup function chosen.

For more advanced or complex partial matching scenarios, especially those involving case sensitivity or positional constraints, users often turn to array formulas combining **ISNUMBER**, **SEARCH** (or **FIND** for case sensitivity), and **INDEX/MATCH**. For instance, a formula like `=INDEX(A:A, MATCH(TRUE, ISNUMBER(SEARCH(B2, A:A)), 0))` can perform a similar partial match lookup. This combination checks if the search value (B2) is found within any cell in the lookup array (A:A) and returns the corresponding row index if the check is true.

However, these alternatives often require entering the formula using **Ctrl+Shift+Enter** (making it an array formula) or understanding more intricate function nesting. The **VLOOKUP** method utilizing wildcards is generally favored for its simplicity, backward compatibility, and the ease with which it handles the concatenation required for a simple "contains" search across the lookup array, making it the preferred method for most users needing a quick and clean partial match solution.

Advanced Considerations and Limitations

When utilizing wildcards for partial matching, it is critical to understand the limitations related to duplicate matches. If the search term (e.g., "San") matches multiple entries in the lookup column (e.g., "San Antonio Spurs" and "San Francisco 49ers"), VLOOKUP will always return the **first match** it encounters within the specified range. It does not prioritize or list subsequent matches. This limitation underscores the need for clear data hierarchy and potentially the use of helper columns or more complex array formulas if all possible matches must be retrieved.

Another important consideration involves special characters. If the data you are searching for actually contains an asterisk (*****) or question mark (**?**), which are the standard wildcards, you must use a tilde (**~**) as an escape character before the wildcard. For example, to search for "A*B", the lookup value must be constructed as `"A~*B"`. Failure to escape actual wildcards will result in Excel

treating them as placeholders instead of literal characters, leading to incorrect matches.

Finally, performance should be considered when applying this formula across extremely large datasets (tens of thousands of rows). VLOOKUP with wildcards performs string searching, which is computationally intensive compared to simple numerical or exact string matching. While effective for typical business datasets, extreme caution should be taken when using this method on massive arrays, as it may cause significant recalculation delays.

Conclusion

Finding partial matches between two columns is a fundamental skill in advanced data analysis using Excel. By strategically combining the robust lookup capabilities of VLOOKUP with the flexibility of wildcards, you can construct a formula that efficiently reconciles disparate data points, such as abbreviated names and full descriptions.

The standard formula, `=IFERROR(VLOOKUP("*"&B2&"*", $A$2:$A$9, 1, 0), "")`, is an elegant and powerful solution. It allows the identification of contained strings and manages potential errors cleanly by utilizing the IFERROR function, resulting in a professional and highly functional output column.

This technique is applicable across numerous professional contexts, from inventory management where product codes vary slightly, to human resources databases where employee names might have prefixes or suffixes that need to be ignored during the matching process. Mastering the use of wildcards within lookup functions ensures your data manipulation skills are equipped to handle the complexities of real-world data preparation.