How to Find Partial Matches in Two Columns with Google Sheets

Authored by

mohammed looti

January 9, 2026

Identifying where text strings partially overlap between two distinct columns is a common requirement when working with data analysis or cleaning in Google Sheets. Unlike finding an exact match, which is straightforward, locating a **partial match** requires the strategic use of advanced functions and specific operators. This comprehensive guide will detail the most effective and robust methods for pinpointing these partial overlaps, primarily utilizing a powerful combination of the VLOOKUP function, the IFERROR function, and crucial wildcards. We will also briefly explore how the COUNTIF function can be employed for simple verification checks, ensuring you can manage complex datasets with confidence and precision.

The challenge of finding partial matches often arises when dealing with disparate datasets that need consolidation, such as matching abbreviated product codes to full names, or standardized city names to user-input variations. Simply comparing column A to column B will return only exact matches, ignoring valuable connections where one string is contained within another. By implementing special characters known as wildcards, we can instruct Google Sheets to look for a specific substring anywhere within a larger string, transforming this difficult task into a manageable automated process. If the result of the comparison is greater than zero, it confirms the existence of a partial match. For immediate visual recognition, you can integrate conditional formatting to highlight the cells where a partial match was identified.

Utilizing the VLOOKUP and Wildcard Combination

The most reliable and frequently recommended method for identifying a partial match while simultaneously returning the corresponding value relies on the robust capabilities of the VLOOKUP function. Traditionally used for exact matches, `VLOOKUP` becomes highly flexible when its search criterion is wrapped in wildcards. The core principle involves instructing `VLOOKUP` to search for the contents of a lookup cell, for example, B2, but to allow for any number of characters both before and after that content within the search range A2:A9. This technique allows us to retrieve the actual matching value from the search range directly, rather than just a simple Boolean indicator of presence.

To implement this strategy effectively, we combine the search value with the asterisk (`*`) wildcard using the ampersand (`&`) concatenation operator. The asterisk serves as a placeholder for zero or more characters. Therefore, the search key `"*"&range2&"*"` tells the function: "Find the content of `range2`, regardless of what text precedes it or what text follows it." If `range2` (or B2 in our example) contains the text "Rockets," the formula will successfully locate "Houston Rockets," "Team Rockets East," or any other string within the search column that includes the word "Rockets."

Since `VLOOKUP` is designed to return an error (`#N/A`) if the exact or partial match is not found within the specified range, we wrap the entire expression within the IFERROR function. This crucial step cleans up the output, preventing the sheet from being cluttered with error messages and allowing

us to specify a clean return value, such as a blank cell (""). This makes the resulting data much easier to read and analyze, particularly when applying the formula across hundreds or thousands of rows.

Formula Syntax for Partial Matches

The following syntax represents the standard structure used to find and return partial matches between values in one column and a range in a second column:

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

This particular formula checks if the value in the lookup cell, **B2**, has a partial match with any cell within the defined search range, which is set as **$A$2:$A$9**. Note the absolute referencing (dollar signs) used in the range to ensure that when the formula is dragged down, the search area remains fixed, a critical best practice for range-based comparisons in Google Sheets. The parameters indicate that we are looking for the content of B2 wrapped in wildcards within the lookup range A2:A9, and if found, we return the value from the first column (1) of that range based on an exact match type (0), which is necessary when using wildcards.

If the desired partial match is successfully located, the formula returns the entire text string found in the range **A2:A9**. Conversely, if the search criteria defined by the wildcards fails to locate any containing string, the `VLOOKUP` returns an error, which is then gracefully intercepted by the IFERROR function, resulting in a blank cell being displayed. This controlled output is essential for subsequent data processing or filtering operations.

The careful placement of asterisks (*****) around the reference to cell **B2** is paramount. These wildcards signify that the matching text from **B2** can be preceded by any text or followed by any text within the cells of the search column (A2:A9). This is what enables the partial matching capability, differentiating this technique from standard, exact lookups. The `&` symbols connect the static wildcard characters (in quotes) with the dynamic cell reference, forming a single lookup string.

Step-by-Step Example Implementation

To solidify understanding, let us walk through a practical scenario involving the comparison of full team names against their abbreviated counterparts. Suppose we have one column (Column A) containing the **full team names** of various basketball teams and a second column (Column B) containing **abbreviated names** or short versions of those teams. Our goal is to verify if the abbreviation in Column B exists as a substring within any of the full names in Column A and, if so, retrieve the full name.

Consider the following initial dataset configuration. Column A holds the complete, authoritative list, while Column B contains the potentially messy or abbreviated lookup terms. The use case here is data harmonization--ensuring that the short names can be correctly linked back to their official, long-form counterparts. We will insert the formula into Column C to display the resulting matches.

	A	B	C
1	**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	Thunder	
8	Indiana Pacers	Orlando	
9	Denver Nuggets	Jazz	
10			
11			
12			
13			
14			

To perform this required partial lookup, we must initiate the process in the first result cell, which, in this case, is cell **C2**. We input the formulated expression, ensuring that the lookup value references **B2** and the search range strictly covers **$A$2:$A$9**. This structure ensures that we are looking up the abbreviation "Rockets" within the locked list of full names.

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

After entering the formula into **C2**, we can execute the calculation and then easily apply this logic to the rest of the dataset. This is accomplished by clicking on the small square at the bottom-right corner of cell C2 and dragging the formula down through the remaining cells in column C. Because we used **absolute referencing** (the dollar signs) for the range `$A$2:$A$9`, that reference will remain locked, while the lookup cell reference (`B2`, `B3`, `B4`, etc.) will update dynamically for each row.

Analyzing the Partial Match Results

Once the formula has been successfully applied down column C, the results clearly demonstrate

the effectiveness of the partial matching technique. If the team name in column B successfully finds a partial match within any of the full team names listed in column A, the full team name from column A is returned and displayed in column C. This indicates a successful data linkage based on the partial string similarity.

C2 =IFERROR(VLOOKUP("*"&B2&"*", $A$2:$A$9, 1, 0), "")

	A	B	C	D
1	**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	Thunder		
8	Indiana Pacers	Orlando	Orlando Magic	
9	Denver Nuggets	Jazz		
10				
11				
12				
13				
14				
15				

Conversely, if no part of the abbreviated name in column B is found within any cell in the search range in column A, the `VLOOKUP` returns its default error. As configured, the IFERROR function intercepts this signal and returns a blank cell instead, clearly indicating that no match was established. This output structure provides an immediate visual summary of which abbreviations are represented in the master list and which are not.

The outcome of applying the formula can be summarized by observing specific rows and their corresponding results:

The abbreviated term "Rockets" was successfully linked to the full name "Houston Rockets," as "Rockets" is a partial component of the full name.
Similarly, "Spurs" found a partial match with "San Antonio Spurs," confirming its presence within the longer string.
Crucially, the lookup value "Lakers" had no partial match in the search range, and therefore, the formula returned a blank cell, facilitated by `IFERROR`.

Alternative Method: Using the COUNTIF Function for Verification

While the `VLOOKUP` method is superior for returning the actual matched value, sometimes you only require a simple check: does a partial match exist, yes or no? For this verification purpose, the COUNTIF function provides a simpler, faster alternative. The `COUNTIF` function counts the number of cells within a specified range that meet a given criterion. By combining `COUNTIF` with wildcards, we can count how many times a substring appears in the target column.

The syntax for using COUNTIF function for partial matching is slightly different but follows the same wildcard logic. If we wanted to check how many times the term in cell B2 appears partially in the range A2:A9, the formula would look like this: `=COUNTIF(A2:A9, "*"&B2&"*")`. This formula returns a numerical value representing the count of partial occurrences. If the result is 1 or greater, a partial match exists; if the result is 0, no match was found. This method is often preferred when the output column only needs to serve as a match indicator rather than a data retrieval tool.

This approach is highly useful for validation tasks or preliminary data cleaning where you need to quickly flag rows containing potential matches without needing to pull the corresponding full value. If you need a Boolean output (TRUE/FALSE) instead of a count, you can modify the formula: `=COUNTIF(A2:A9, "*"&B2&"*")>0`. This resulting formula directly tells you whether a partial match was successfully located, returning **TRUE** if found and **FALSE** otherwise. This simplifies the creation of filters or flags within your dataset.

Applying Advanced Formatting with Partial Matches

Once you have successfully identified which rows contain a partial match, utilizing Conditional formatting is an excellent way to visually highlight the successful linkages for immediate identification and analysis. This technique dramatically improves the readability of large datasets, allowing users to instantly spot matches or non-matches without having to scrutinize individual cell values.

If you used the `VLOOKUP` method described previously, highlighting the successful matches in Column C is straightforward: apply a conditional formatting rule that highlights all cells in Column C that are **not empty**. Since only successful matches return the text string and unsuccessful matches return a blank (`""`), this conditional rule effectively flags all confirmed partial matches in the results column.

Alternatively, if you employed the `COUNTIF` method to check for existence, you can use a custom formula within the conditional formatting rules. For example, to highlight the cells in Column B that have a match in Column A, select Column B, choose "Custom formula is," and input the formula: `=COUNTIF($A$2:$A$9, "*"&B2&"*")>0`. This will cause the corresponding cell in Column B to change format whenever its content is found partially within the master list in Column A, providing

powerful visual feedback directly on the lookup column itself.

Troubleshooting Common Partial Match Issues

When working with advanced lookup functions like `VLOOKUP` and COUNTIF function in combination with wildcards, several common issues may arise that require immediate troubleshooting. A frequent problem is **case sensitivity**. By default, Google Sheets functions are case-insensitive, meaning "Rockets" matches "rockets." If your data requires strict case sensitivity, you must transition to more complex array formulas involving functions like `REGEXMATCH` or `FIND`, as the simple `VLOOKUP` and `COUNTIF` functions will not enforce case differentiation.

Another source of error is incorrect range selection or referencing. Ensure that the lookup range (e.g., `$A$2:$A$9`) is absolutely referenced using dollar signs if you plan to drag the formula down multiple rows. If the range is not locked, it will shift, potentially omitting or including incorrect cells in subsequent rows, leading to inaccurate match results or `#REF!` errors. Always confirm that your search range encompasses all necessary data points in the master column.

Finally, double-check your concatenation syntax when embedding the wildcards. The formula must be constructed precisely: `"*"&B2&"*"`. Missing an ampersand (`&`) or forgetting the quotation marks around the asterisk (`*`) will prevent the wildcard from being interpreted correctly. Instead, the lookup will treat the entire string as a literal search term, resulting only in exact matches or errors. Mastering this precise concatenation is key to successfully executing partial lookups in Google Sheets.

The following tutorials explain how to perform other common tasks in Google Sheets: