Learn Partial Match Lookup in Google Sheets: A Step-by-Step Guide

Authored by

Mohammed looti

November 12, 2025

One of the most persistent difficulties encountered when managing datasets in Google Sheets is the requirement to execute lookups based on a **partial match** instead of the default exact match. Standard functions, such as the widely employed VLOOKUP, are inherently designed to retrieve values only when a perfect, character-for-character correspondence is found. However, modern data manipulation often necessitates flexibility due to data entry variations, abbreviations, or naming inconsistencies. By strategically integrating **wildcard characters** into the lookup criteria, we can compel VLOOKUP to successfully search for and identify substrings within a larger text string across two separate columns. This powerful technique is indispensable for critical processes like data cleaning, verifying records, and consolidating disparate datasets.

To effectively identify partial matches between any two specified columns within your spreadsheet environment, you must leverage a powerful combination of functions and specialized syntax. This core formula harnesses the flexibility of the wildcard character--the asterisk (*****)--by concatenating it with the value being searched. This ensures that the search operation scans for the lookup value anywhere within the target cell's content, allowing for prefixes and suffixes to exist around the desired substring.

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

This specific formula is meticulously constructed to verify whether the data held within cell **B2** exists as a partial substring within any cell located in the search range **A2:A9**. The output delivers immediate, actionable feedback: if a partial match is successfully identified, the formula returns the matching value from the specified column (Column A). Conversely, should the search fail to locate the substring within the defined array, the function is designed to return a designated blank space. This crucial error handling mechanism ensures a clean, manageable output dataset devoid of distracting error messages.

Deconstructing the Partial Match Mechanism with Wildcards

The foundation of enabling VLOOKUP to perform a partial match rests entirely upon the precise placement and utilization of **wildcard characters**. Typically, VLOOKUP requires its final argument, the range lookup parameter, to be set to `0` or `FALSE` to mandate an **exact match**. By default, VLOOKUP interprets the search key argument literally. However, by strategically prepending and appending the asterisk wildcard (`*`) to our search term, we fundamentally transform the search criteria from a literal match into an advanced pattern-matching exercise.

In the structure `"*"&B2&"*"`, the ampersand (`&`) functions as the essential concatenation operator. This operator seamlessly joins three distinct elements into a single, cohesive search string: the leading wildcard, the variable content derived from cell **B2**, and the trailing wildcard. The leading asterisk signifies that any sequence of characters (including zero characters) may appear before

the content of **B2**. Analogously, the trailing asterisk indicates that any sequence of characters may follow the content of **B2**. This combined pattern instructs VLOOKUP to efficiently search the lookup range for any cell that contains the exact sequence of characters found in **B2**, irrespective of the surrounding text. This principle is the cornerstone for successfully executing partial match lookups using this versatile function.

Grasping the exact implications of these wildcards is paramount for accurate data retrieval. For example, if cell **B2** contains the text string "Corp", the dynamically generated search string becomes "*Corp*". This pattern will successfully identify and match cells containing "Acme Corporation", "Corp Headquarters", or simply "Corp". The lookup range, provided as the second argument (`$A$2:$A$9`), defines the fixed array where VLOOKUP executes this pattern search. The utilization of **dollar signs** (**$**) establishes an absolute reference, which is crucial because it locks the search range in place, preventing it from shifting when the formula is copied or dragged down to subsequent rows. This ensures the integrity and consistency of the lookup operation across the entire calculation column.

Implementing Robust Error Handling with IFERROR

While the combination of VLOOKUP and wildcards is highly effective for identifying partial matches, any lookup function inherently risks generating the standard `#N/A` error. This error specifically occurs whenever VLOOKUP fails to locate any matching criteria for the provided lookup value within the defined range. In large, complex datasets, having a column polluted with numerous `#N/A` errors significantly complicates subsequent analysis, impairs readability, and demands additional cleaning steps. To proactively mitigate this common issue and ensure a professional output, we encapsulate the entire lookup formula within the powerful IFERROR function.

The **IFERROR** function is a critical component for effective exception handling within Google Sheets. It requires two primary arguments: the calculation to test (which is our comprehensive VLOOKUP formula), and the value to be returned if the first argument evaluates to any type of error. By structuring the formula as `=IFERROR(VLOOKUP(...), "")`, we are instructing the spreadsheet environment to prioritize the VLOOKUP operation. If VLOOKUP successfully returns a match (a retrieved value from Column A), that value is correctly displayed in the cell.

However, if VLOOKUP fails to find a partial match and consequently produces an error (most commonly `#N/A`), the IFERROR function intercepts this error before it is displayed. Instead of presenting the raw error message, the function executes its second argument, which is specified as `""`. This pair of double quotes represents an **empty string**, resulting in a blank cell. This practice is strongly recommended because it maintains a clean result column, ensuring that only positive match results are visible. This significantly improves data readability and streamlines any

subsequent filtering or analysis that depends on the matched data.

The importance of utilizing IFERROR becomes even more apparent when dealing with input data that is expected to contain a high number of non-matching entries. Without this proactive error management, filtering and sorting the data based on partial match success becomes unnecessarily difficult, often requiring manual steps or auxiliary formulas to exclude the error values. By integrating IFERROR, we construct a robust, self-cleaning formula that consistently delivers clear and concise results regardless of whether a partial match is located.

Practical Application: Matching Inconsistent Team Names

To fully appreciate the efficacy of this technique, let us examine a typical real-world scenario involving two columns of related, yet inherently inconsistent, data. Imagine you are tasked with standardizing a database of sports organizations. Column A contains the **Full Team Name** (official, detailed titles), while Column B contains **Abbreviated Team Names** (shorter, potentially inconsistent user entries, or common nicknames). Our precise objective is to use the partial string from Column B to identify and retrieve the corresponding full name listed in Column A.

We begin by visualizing the initial dataset structure below. Column A serves as our master list of authorized full names, and Column B contains the entries we need to cross-reference against the master list for partial content 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			

Our immediate goal is to populate Column C with the relevant full team name from Column A only

when a partial match is successfully confirmed. To commence this process, we input the complete formula directly into cell **C2**, which acts as the starting point for our resulting match column.

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

Upon executing this formula in **C2**, the system immediately attempts to determine if the string "Rockets" (the content of **B2**) is contained anywhere within the locked array `$A$2:$A$9`. Given that the full name "Houston Rockets" exists within that defined range, the formula correctly retrieves and returns "Houston Rockets" to cell **C2**. This successful initial result validates the formula setup for the first row of data. The next critical step involves scaling this logic efficiently to cover every subsequent row in our dataset.

Scaling the Solution and Interpreting the Final Results

Once the formula is accurately placed and confirmed in cell **C2**, scaling the solution across the remainder of the dataset is a rapid and simple operation. By utilizing the fill handle--the small square located at the bottom-right corner of cell **C2**--we can copy and drag the formula down through every remaining cell in Column C. Crucially, because we meticulously used the absolute reference (`$A$2:$A$9`) for the lookup range, that array remains static throughout the process. Concurrently, the relative reference to the search key (**B2**) dynamically updates to **B3**, **B4**, and so on, ensuring each row is checked against the master list.

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				

The resulting data in Column C provides a clear and unambiguous interpretation of the partial matching operation. If the abbreviated team name in Column B successfully locates a partial match within any full team name in Column A, the corresponding full team name is accurately retrieved and displayed in Column C, providing immediate verification. Conversely, if the lookup fails to identify the string from Column B within any cell in Column A, the integrated **IFERROR** function ensures that a blank cell is returned, effectively signaling the absence of a partial match without cluttering the data.

A review of the final output clearly illustrates how the formula capably handles different scenarios encountered during the lookup process:

The abbreviation "Rockets" successfully achieved a partial match with the full name "Houston Rockets," resulting in the accurate return of the full name.
The abbreviated entry "Spurs" matched "San Antonio Spurs," definitively demonstrating that the wildcard effectively handles prefixes and suffixes surrounding the search string.
The abbreviation "Lakers" did not find any partial match in the specified range. Consequently, the IFERROR function correctly returned a blank cell, maintaining data integrity and cleanliness.
The abbreviation "Jazz" matched "Utah Jazz", further confirming the successful and versatile application of wildcards for partial string searches.

This technique offers an exceptionally efficient and highly scalable solution for cross-referencing datasets where absolute textual consistency is impractical or cannot be guaranteed, delivering substantial time savings compared to manual reconciliation efforts.

Limitations and Considerations for Advanced Partial Matching

While the combination of VLOOKUP and wildcards constitutes a fundamental technique for basic partial matching, it is essential to recognize its inherent limitations and consider alternative methods for increasingly complex requirements. The primary constraint of this specific VLOOKUP method is that it ceases its search immediately upon identifying the **first** partial match within the lookup range. If the lookup array is not sorted or if multiple entries contain the same partial string (e.g., "North" matching "North Carolina" and "North Dakota"), VLOOKUP will exclusively return the value corresponding to the entry that appears earliest in the list.

Furthermore, VLOOKUP operates as a generally non-case-sensitive function within Google Sheets, meaning that "rockets" will match "Rockets" without distinction. Although this behavior is frequently beneficial for general data cleaning purposes, users who require true case-sensitive partial matching must utilize more advanced formula constructs. These often involve powerful functions such as **QUERY** or **REGEXMATCH** in conjunction with **FILTER**, as these tools provide granular control over complex search patterns and the manipulation of output arrays.

For professionals managing exceptionally large datasets--those comprising hundreds of thousands of rows--performance and recalculation speed may become significant concerns. While VLOOKUP is generally efficient, iterating such complex formulas across massive ranges can noticeably impede sheet performance. In these high-volume situations, mastering the **QUERY** function provides a robust, SQL-like alternative capable of handling advanced pattern matching and often executing lookups with superior efficiency compared to traditional iterative array formulas. Nonetheless, for the vast majority of routine data analysis tasks involving cross-referencing two columns, the demonstrated VLOOKUP and wildcard approach remains the most straightforward, quickest, and most reliable method to implement effective partial match lookup functionality.

Additional Resources for Spreadsheet Mastery

The following tutorials explain how to perform other common tasks in Google Sheets, building upon the foundational knowledge of lookup functions, VLOOKUP, and error handling with IFERROR: