How to Easily Use Wildcard Characters in Google Sheets Query

Authored by

stats writer

December 3, 2025

Introduction to Wildcard Functionality in Google Sheets

Wildcard characters are indispensable tools in data manipulation and search operations, allowing users to substitute one or more characters in a search pattern. While many spreadsheet functions rely on specific matching, wildcards introduce a powerful layer of flexibility, enabling partial matches and pattern recognition. In the context of Google Sheets Query, which uses a language similar to standard SQL, the designated wildcard character for matching sequences of text is typically the percent sign (**%**). This functionality is vital for users dealing with large datasets where they need to retrieve data based on uncertain, partial, or variable inputs, moving beyond simple equality checks.

Traditional spreadsheet functions sometimes utilize the asterisk (*****) or the question mark (**?**) for wildcard matching, but the **Google Sheets Query** function specifically adopts the SQL-standard approach, which uses **%** to represent zero or more characters. Understanding this distinction is crucial for constructing effective queries. The power of the wildcard lies in its ability to handle unstructured or semi-structured data inputs efficiently, ensuring that you can locate all relevant records even if you only know a small portion of the target text string. This capability transforms basic data filtering into sophisticated pattern extraction.

When preparing to use wildcards within the **QUERY** function, it is essential to remember that these characters are always paired with the **LIKE operator**, which we will discuss in detail shortly. This combination tells the system that it should not look for an exact match, but rather evaluate whether the data fits the specified pattern. Whether you are trying to find product names that start with a specific prefix, identify email addresses ending in a certain domain, or locate notes containing a specific keyword anywhere in the text, mastering the **%** wildcard is the key to unlocking the full potential of your Google Sheets data analysis.

Understanding the Google Sheets Query Function Syntax

The **QUERY** function is perhaps the most robust data processing tool available in Google Sheets, offering complex filtering, sorting, and aggregation capabilities that surpass standard functions like **FILTER** or **SORT**. Before diving into wildcards, a firm grasp of the basic **QUERY** structure is necessary. The function requires two mandatory arguments: the **data range** and the **query string**. The basic syntax looks like this: `=QUERY(data, query, )`. The **data** argument specifies the range of cells you want to analyze, while the **query** argument is where the SQL-like command resides, defining what to select and how to filter it.

The **query string** itself must be enclosed in double quotes and typically follows the structure: `"SELECT WHERE ORDER BY "`. For instance, `"SELECT A, B WHERE C > 10"`. When incorporating text filtering, especially involving wildcards, we primarily focus on the **WHERE clause**. This clause

defines the criteria that each row must meet to be included in the results. If the data in column A must contain a certain phrase, we write the condition within the **WHERE** clause, employing the specific tools designed for pattern matching--the **LIKE operator** and the **%** wildcard.

A common pitfall for newcomers is confusing the use of wildcards in **QUERY** with their usage in standard functions like **COUNTIF** or **VLOOKUP**. In those contexts, the asterisk (*****) is generally used. However, because the **QUERY** function is designed to mimic standard database interaction languages (SQL), it demands the **%** symbol for partial text matching. Furthermore, within the query string, any text criteria being searched must be enclosed in single quotes (e.g., `'search term'`), distinguishing them from column identifiers (e.g., `A`, `B`). Adhering to these syntactic rules ensures your wildcard queries execute correctly and return the intended results.

The Role of the LIKE operator and the Percent Wildcard (%)

The **LIKE operator** is the specific component within the **QUERY** function's WHERE clause that enables pattern matching rather than simple equality testing. It instructs the database engine to compare the specified column's data against a pattern template. This is where the power of the **%** wildcard comes into play. The percent sign (**%**) acts as a flexible placeholder, signifying that any sequence of zero or more characters can occupy that position in the search string. This flexibility is fundamental to performing the three primary types of partial searches: prefix, suffix, and substring matching.

Consider a scenario where you are searching column A for the word "apple." If you use `WHERE A = 'apple'`, only cells containing that exact text will be returned. If, however, you use the **LIKE operator** combined with the wildcard, such as `WHERE A like 'apple%'`, you are now asking the system to find anything that starts with "apple" (e.g., "applesauce," "appletree," or "apple"). The placement of the **%** determines the nature of the search. Placing it at the end signifies a prefix search, placing it at the beginning signifies a suffix search, and placing it on both sides enables a substring search.

It is important to note that the **LIKE operator** performs case-sensitive matching by default in the standard Google Sheets environment. If you need a case-insensitive search, advanced techniques or preparatory data cleaning might be required (such as using functions to convert the column to all lowercase before querying, although this complicates the standard **QUERY** syntax). For most standard data retrieval tasks, however, the combination of **LIKE** and **%** provides the necessary precision to isolate data based on complex textual patterns within your dataset, maximizing the utility of the Google Sheets Query function.

To utilize the power of pattern matching, you will incorporate the **%** sign as the primary wildcard character in your **Google Sheets queries**, always placing it within the single quotes of the pattern string:

Method 1: Searching for Entries That Start With Specific Characters (Prefix Match)

The first common application of the wildcard is performing a prefix search, often necessary when you know the initial letters of a text entry but need to capture all variations that follow. This method uses the specified text followed immediately by the **%** wildcard. The placement of the wildcard at the end ensures that the query seeks strings that begin exactly with the given prefix, while accepting any subsequent characters. This is extremely useful for categorizing items based on naming conventions or isolating data entries based on a specific departmental or project code found at the beginning of the text.

To implement a prefix search, the formula structure must specify the column you are querying and use the **LIKE operator**. For instance, if you are working with a list of tasks in column A and want all tasks starting with the prefix "hello," the pattern you define is `'hello%'`. The system checks the beginning of every string in column A to see if it matches "h-e-l-l-o," followed by anything (or nothing, as **%** matches zero or more characters).

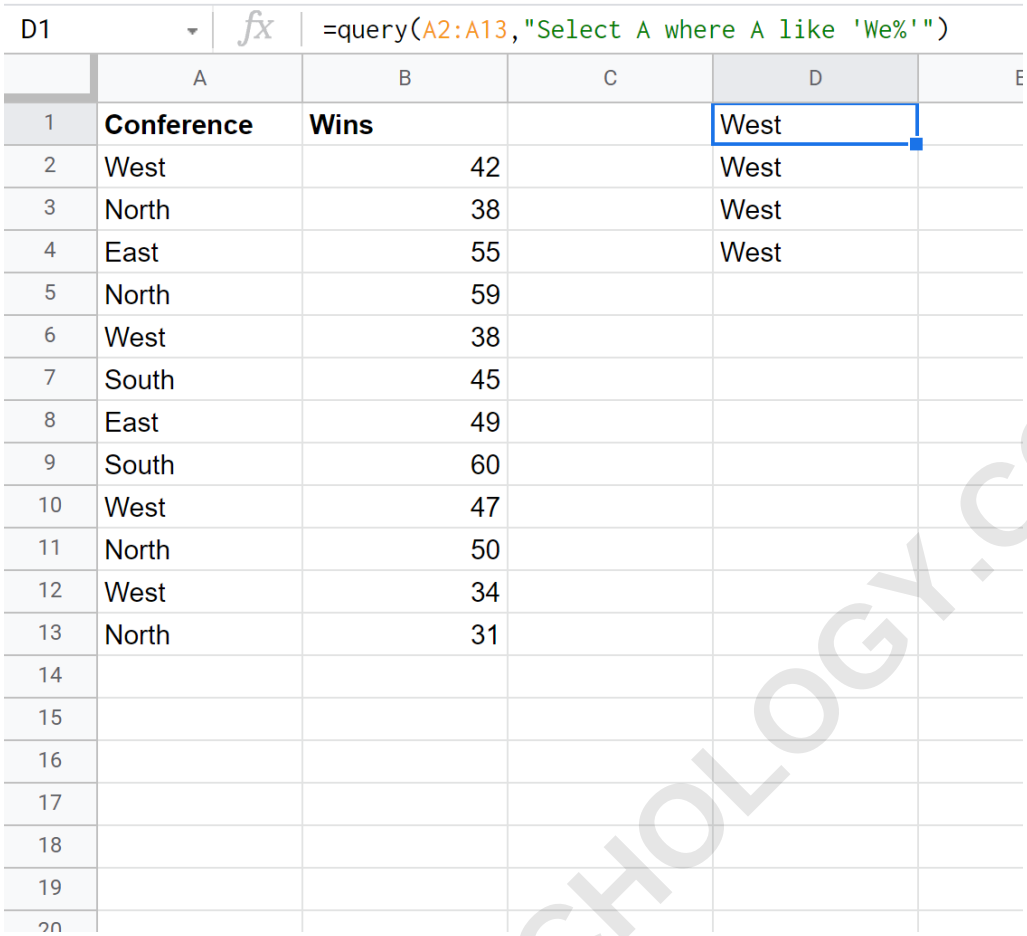
Below is the standard syntax for returning cells that start with certain characters. Notice how the prefix "hello" is anchored at the front, followed by the open-ended wildcard:

=QUERY(A1:A10, "Select A where A like 'hello%'")

Let us consider a practical example. We can use the following formula to efficiently return every cell in column A that starts with the specific prefix **'We'**, perhaps identifying all entries related to a project starting with that abbreviation:

=QUERY(A1:A10, "Select A where A like 'We%'")

The accompanying screenshot provides a visual confirmation of how this formula successfully filters the dataset, returning only those values that meet the criteria defined by the prefix match. This method ensures high precision when the initial characters of the target data are known and fixed:



	A	B	C	D
1	**Conference**	**Wins**		West
2	West	42		West
3	North	38		West
4	East	55		West
5	North	59		
6	West	38		
7	South	45		
8	East	49		
9	South	60		
10	West	47		
11	North	50		
12	West	34		
13	North	31		
14				
15				
16				
17				
18				
19				

Method 2: Searching for Entries That End With Specific Characters (Suffix Match)

The second essential application of the wildcard is conducting a suffix search, which is crucial when you need to identify data based on the ending portion of the text, such as file extensions, domain names, or specific standardized suffixes in codes. In this method, the **%** wildcard is placed at the beginning of the pattern, allowing any characters to precede the required suffix. This effectively anchors the search pattern to the end of the text string.

For instance, if you are searching a list of product descriptions in column A and need to find all products that end with "Pro," you would define the pattern as `'%Pro'`. The wildcard at the beginning ensures that the search accepts any text leading up to the sequence "P-r-o," but the entry must terminate with that exact suffix. This technique is invaluable for data validation or when aggregating items based on a common closing identifier.

The structure for returning cells that end with certain characters requires placing the **%** sign before the desired suffix, telling the **QUERY** function to look for any sequence of characters followed by

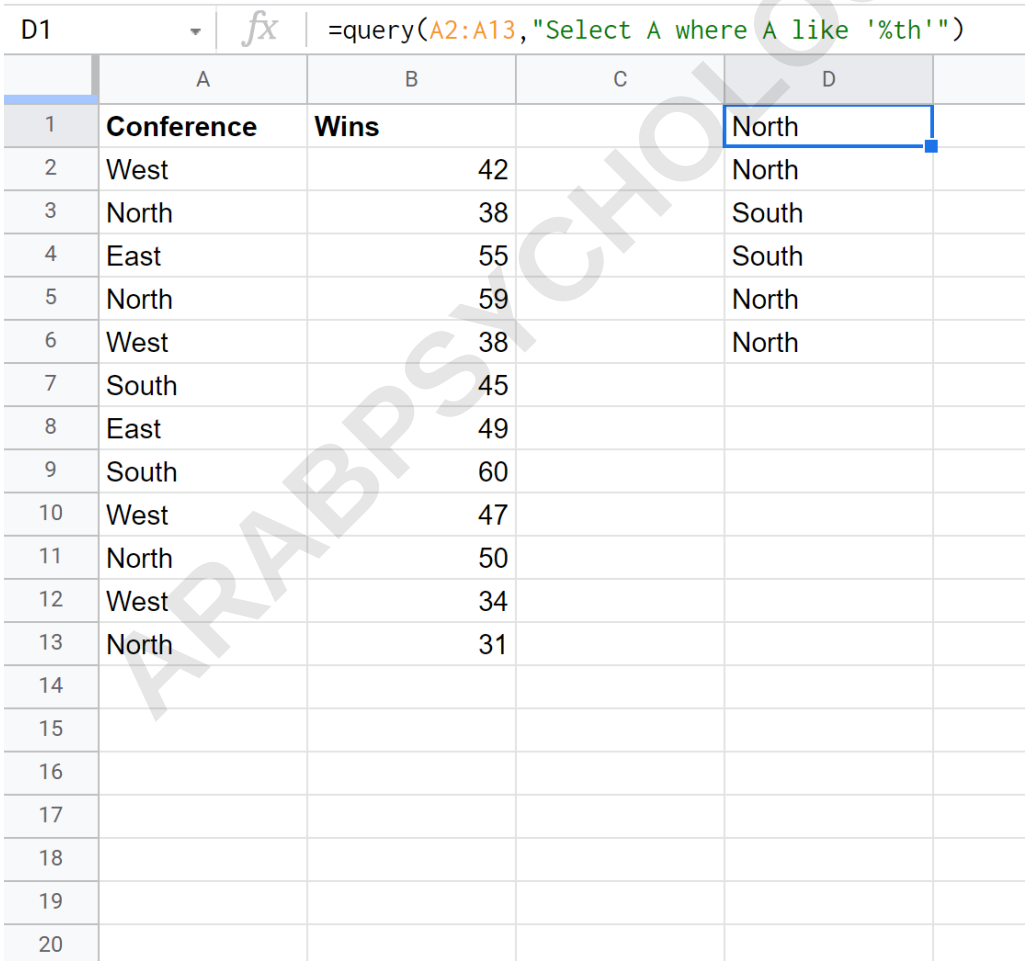
the specified pattern:

=QUERY(A1:A10, "Select A where A like '%hello'")

In a concrete scenario, suppose we want to retrieve all entries in column A that end with the suffix **'th'**. This could be used to find words like "width," "length," or "fourth." We structure the formula by preceding the suffix with the **%** wildcard:

=QUERY(A1:A10, "Select A where A like '%th'")

The resulting data, as shown in the visual aid below, demonstrates how this formula effectively isolates only those records that conclude with the specified sequence. Mastering the suffix match ensures you can precisely target data based on trailing identifiers, a frequent requirement in sophisticated data filtering tasks within Google Sheets Query:



	A	B	C	D
1	**Conference**	**Wins**		North
2	West	42		North
3	North	38		South
4	East	55		South
5	North	59		North
6	West	38		North
7	South	45		
8	East	49		
9	South	60		
10	West	47		
11	North	50		
12	West	34		
13	North	31		
14				
15				
16				
17				
18				
19				
20				

Method 3: Searching for Entries That Contain Certain Characters (Substring Match)

The most flexible use of the **%** wildcard is the substring match, which allows you to find a specific sequence of characters anywhere within the larger text string--at the beginning, end, or middle. This method is the workhorse for general keyword searching and content analysis. To achieve a substring match, the required text is enclosed by wildcards on both sides. This signals to the **LIKE operator** that zero or more characters may precede the target word, and zero or more characters may follow it.

This approach is particularly powerful for unstructured text data, such as comments, notes, or long descriptions, where the exact location of the keyword is unpredictable. For example, if you are analyzing customer feedback in column A and want to find every comment that mentions the word "delivery," you would use the pattern `'%delivery%'`. This ensures that records like "Fast delivery was great," "The delivery truck was late," or "Need better delivery times" are all successfully captured by the query.

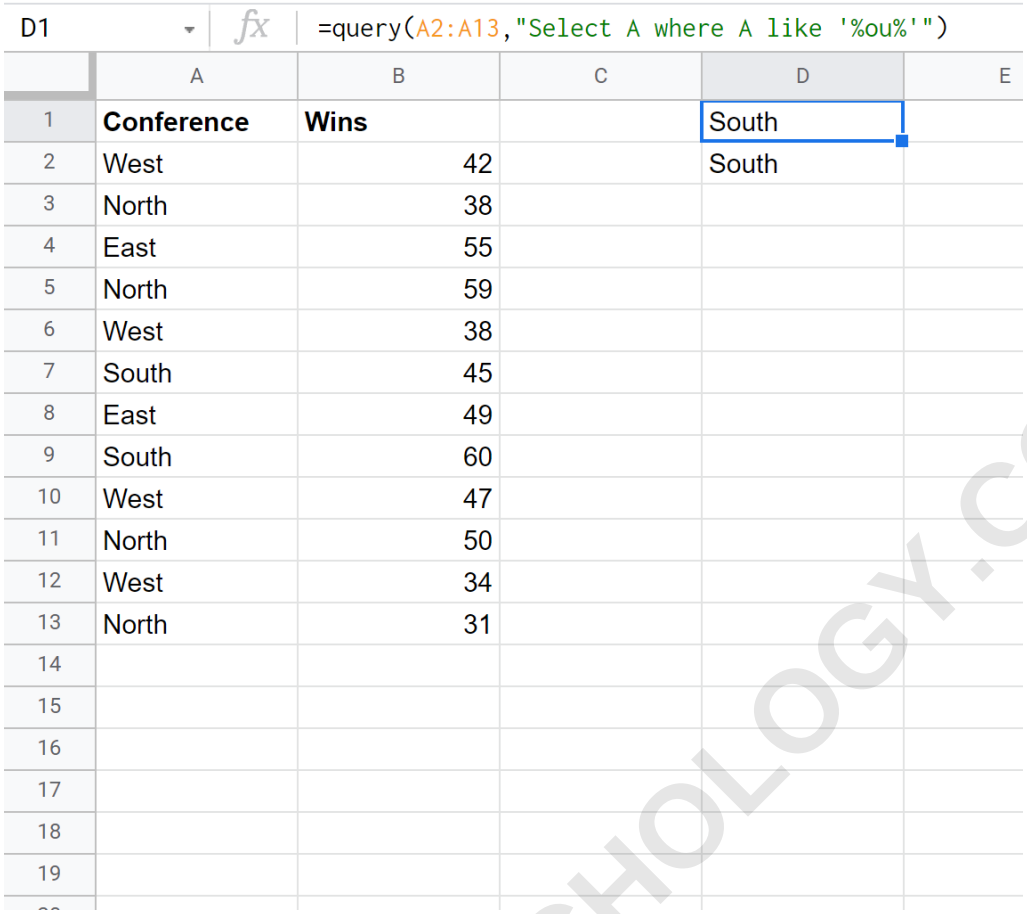
The syntax for returning cells that contain certain characters involves sandwiching the desired keyword or phrase between two **%** signs. This establishes the pattern as fully flexible on both ends, maximizing the search scope across the entire length of the text entry:

=QUERY(A1:A10, "Select A where A like '%hello%'")

Let's apply this to a specific requirement: returning every cell in column A that contains the sequence **'ou'** somewhere within the string. This could pull words like "you," "bought," or "house." The formula leverages the bidirectional wildcard approach to ensure comprehensive coverage:

=QUERY(A1:A10, "Select A where A like '%ou%'")

As illustrated in the final screenshot, the result set includes all entries regardless of where the target substring appears. This substring matching capability, facilitated by the **%** wildcard, is an indispensable technique for deep-diving into textual data using the powerful SQL-like capabilities of the **QUERY** function:



	A	B	C	D	E
1	**Conference**	**Wins**		South	
2	West	42		South	
3	North	38			
4	East	55			
5	North	59			
6	West	38			
7	South	45			
8	East	49			
9	South	60			
10	West	47			
11	North	50			
12	West	34			
13	North	31			
14					
15					
16					
17					
18					
19					

Advanced Considerations and Best Practices for Wildcard Queries

While the **%** wildcard paired with the **LIKE operator** is highly effective, there are several advanced considerations and best practices that can improve the performance and accuracy of your queries. One key aspect is understanding performance implications. Although pattern matching is versatile, it is generally computationally more expensive than exact matches (using `WHERE A = 'term'`). For extremely large datasets (hundreds of thousands of rows), excessive use of bidirectional wildcards (`'%term%'`) might lead to slower query execution times, as the system has to scan the full length of every string.

Another important consideration is handling special characters within the search term. If your required search term itself contains a literal percent sign (**%**) or an underscore (_, the single-character wildcard in SQL, though less commonly used in the Sheets dialect for partial string matching), you would typically need an escape character to treat it literally rather than as a wildcard. While Google Sheets handles the **QUERY** language slightly differently from traditional SQL dialects, generally, if you must search for a literal **%**, you may need to preprocess the data or use more complex regular expressions instead of the simple **LIKE** clause, as **QUERY** interprets **%** aggressively as a wildcard.

Finally, it is crucial to ensure data consistency before running wildcard queries. Since **QUERY** is case-sensitive by default for text matching, the pattern `'apple%'` will not match "Apple Pie." If case insensitivity is required, a recommended best practice is to modify the column contents within the **QUERY** itself before applying the filter, although this requires expertise with transformation functions like `lower()` within the LIKE operator context. For instance, `"SELECT A WHERE lower(A) like 'apple%'"` converts all values in column A to lowercase temporarily for the comparison, ensuring both "Apple" and "apple" are matched equally.

Alternative Wildcard Techniques: REGEXMATCH

While the **LIKE operator** combined with the **%** wildcard is the standard SQL approach within the **QUERY** function, users who require even more complex pattern matching capabilities should be aware of the **MATCHES** clause, which allows the use of regular expressions (Regex) directly within the query string. Regex offers exponential power compared to simple wildcards, enabling searches based on character classes, repetitions, specific positions, and negative lookups.

For example, if you needed to find all strings that contain a number followed immediately by the letter 'A', standard **%** wildcards cannot achieve this precise matching. However, using the **MATCHES** operator and an appropriate Regex pattern (e.g., `'%d+A%'`) allows for this complexity. The syntax would look like `"SELECT A WHERE A MATCHES '.*A.*'"`. Although Regex has a steep learning curve, it is the ultimate tool for highly specific textual pattern recognition within your data.

For most basic prefix, suffix, or substring searches, the **LIKE** and **%** approach remains superior due to its simplicity and readability. However, keeping the **MATCHES** option in reserve is essential for when data analysis requires advanced, intricate pattern validation, providing a complete toolkit for managing and filtering complex data in the Google Sheets Query environment.

Summary of Wildcard Query Methods

To ensure clarity and easy reference, the following list summarizes the three fundamental methods for using the **%** wildcard within the **QUERY** function, highlighting how the placement of the wildcard dictates the nature of the text search:

Prefix Match (Starts With): Place the wildcard **after** the search term (e.g., `'Data%'`). This retrieves all records that begin with "Data" but may have any characters following it. This is highly effective for filtering based on initial codes or common prefixes.

Suffix Match (Ends With): Place the wildcard **before** the search term (e.g., `'%Base'`). This retrieves all records that end exactly with "Base," accepting any preceding characters. Useful for finding files or entries based on their conclusion.

Substring Match (Contains): Place the wildcard **before and after** the search term (e.g., `'%Key%'`). This performs the broadest search, capturing the term "Key" if it appears anywhere within the text string. This is the standard method for general keyword searching.

By systematically applying these methods, users can move beyond rigid exact-match filtering and leverage the true power of text manipulation offered by the SQL-derived language of the **Google Sheets Query** function.