Google Sheets Query: Use “NOT LIKE” in Query

Authored by

Mohammed looti

October 27, 2025

Unlocking Precision: The Indispensable Role of `NOT LIKE` in Google Sheets Query

The QUERY function in Google Sheets (1) stands out as one of the most powerful features available, enabling users to perform sophisticated data manipulation using a language structure closely resembling SQL (1). It allows for seamless extraction, transformation, and summarization of large datasets, moving far beyond basic filtering operations. While inclusion criteria (selecting rows that match a pattern) are frequently utilized, true mastery of data analysis often requires the precise exclusion of specific data points.

This need for precise removal is where the **`NOT LIKE`** operator proves invaluable. This operator is specifically designed to retrieve all rows where the value in a designated column does **not** conform to a specified textual pattern. Unlike simple equality checks, **`NOT LIKE`** handles substring matching, allowing analysts to filter out entries based on partial text matches, regardless of where the pattern appears within the string. This capability is fundamentally crucial for data cleansing, reporting accuracy, and ensuring that only highly relevant information contributes to your final analysis.

Throughout this comprehensive guide, we will dissect the mechanics of **`NOT LIKE`**, moving from its basic syntax to advanced applications involving multiple exclusion criteria. By the end, you will possess a robust understanding of how to leverage this operator for advanced data filtering (1) techniques within your Google Sheets (2) environment, significantly enhancing your control over complex spreadsheet data.

Deconstructing the `NOT LIKE` Syntax and Pattern Matching

Implementing **`NOT LIKE`** within your QUERY function (2) requires integrating it correctly into the `SELECT WHERE` clause. The structure is logical: you specify the column you wish to evaluate, followed by the **`NOT LIKE`** operator, and finally, the pattern you wish to exclude, enclosed in single quotes. This precise construction tells the query processor exactly which rows to discard from the result set based on their textual content.

The true power of **`NOT LIKE`** is unlocked through the strategic use of the wildcard character (1), represented by the percentage sign (`%`). This symbol acts as a placeholder for zero or more characters, enabling flexible pattern definition. For instance, if you define the pattern as **`'text%'`**, the query will exclude strings that start with "text". If you use **`'%text'`**, it excludes strings that end with "text". Most commonly, the pattern **`'%text%'`** is used to exclude any string that contains "text" anywhere within its body, which is essential for general substring exclusion. Mastering the placement of this wildcard character (2) is critical for achieving the exact exclusion behavior you require.

It is vital to remember that the pattern must be treated as a string literal and enclosed in single quotes within the overall query string, which itself is enclosed in double quotes. This nested quoting structure is a common source of error for new users of the QUERY function (3). The following practical example illustrates a basic `NOT LIKE` query designed to process a range and exclude any row where column B contains the specific pattern "uar":

=QUERY(A1:C11, "SELECT * WHERE NOT B LIKE '%uar%'")

In this formula, the query evaluates the data in the `A1:C11` range. It meticulously filters out any record where the corresponding cell in column B contains the subsequence of characters "uar". This demonstrates the fundamental principle of exclusion: identifying and removing data based on a defined textual pattern rather than an exact match.

Practical Demonstration: Excluding Player Positions from a Dataset

To truly appreciate the functionality of `NOT LIKE`, let us apply it to a real-world scenario involving structured data. Consider a sports analyst working with a comprehensive list of basketball players, where the dataset includes their name, their position on the court, and the team they belong to. Our analytical goal is to isolate players who do **not** play specific positional roles, thereby focusing the analysis on other categories.

The foundation of our analysis is the sample dataset shown below. This table represents the raw data upon which the QUERY function (4) will operate. We will specifically target the 'Position' column (Column B) for our exclusion criteria.

	A	B	C	D
1	**Team**	**Position**	**Points**	
2	Mavs	Guard	22	
3	Hawks	Guard	20	
4	Magic	Forward	19	
5	Thunder	Guard	15	
6	Kings	Forward	29	
7	Pacers	Forward	24	
8	Clippers	Forward	28	
9	Nuggets	Guard	20	
10	Lakers	Forward	14	
11	Celtics	Guard	12	
12				
13				
14				
15				
16				
17				
18				
19				
20				

Suppose we need a list that explicitly excludes all players whose position contains the pattern "uar". This pattern is strategically chosen because it appears in common positions like "Guard" and "Shooting Guard." By applying `NOT LIKE '%uar%'` to column B, we instruct Google Sheets (3) to discard all records associated with these roles. The following query executes this precise exclusion:

=QUERY(A1:C11, "SELECT * WHERE NOT B LIKE '%uar%'")

The result of executing this query immediately highlights the effectiveness of the operator. The output table is dramatically reduced, featuring only those players whose position description does not include the target pattern "uar".

A14 =QUERY(A1:C11, "SELECT * WHERE NOT B LIKE '%uar%'")

	A	B	C	D	E
1	**Team**	**Position**	**Points**		
2	Mavs	Guard	22		
3	Hawks	Guard	20		
4	Magic	Forward	19		
5	Thunder	Guard	15		
6	Kings	Forward	29		
7	Pacers	Forward	24		
8	Clippers	Forward	28		
9	Nuggets	Guard	20		
10	Lakers	Forward	14		
11	Celtics	Guard	12		
12					
13					
14	Team	Position	Points		
15	Magic	Forward	19		
16	Kings	Forward	29		
17	Pacers	Forward	24		
18	Clippers	Forward	28		
19	Lakers	Forward	14		
20					
21					
22					
23					

As clearly illustrated in the filtered result set, all rows corresponding to "Guard" and "Shooting Guard" positions have been flawlessly removed. This practical example underscores how **`NOT LIKE`** serves as a robust mechanism for isolating specific data subsets by defining what should be absent, providing a clear, focused view for subsequent steps in the data pipeline.

Implementing Multi-Criteria Exclusion with Logical Operators

The capabilities of **`NOT LIKE`** are not limited to single-column, single-pattern exclusion. For sophisticated data refinement, it is often necessary to apply several simultaneous exclusion conditions across different columns. This is achieved by linking multiple **`NOT LIKE`** statements using logical operators, primarily the `AND` operator (1).

When using the `AND` operator (2) to combine two exclusion criteria, the resulting row must satisfy *both* negative conditions to be included in the final output. For example, if we state `WHERE

NOT Condition1 AND NOT Condition2`, only rows that fail both Condition 1 and Condition 2 will be retained. Conversely, if you were to use the `OR` operator in this context, the filtering would be much stricter, excluding rows that fail Condition 1 *or* Condition 2. Understanding the Boolean logic behind combining exclusions is paramount for accurate results.

Let us extend our basketball example. We now want to exclude not only players whose 'Position' contains "uar" but also players whose 'Team' name contains the pattern "er". This requires two distinct **`NOT LIKE`** clauses, one for column B ('Position') and one for column A ('Team'), connected by the `AND` operator (3).

The resulting combined query structure is as follows:

=QUERY(A1:C11, "SELECT * WHERE NOT B LIKE '%uar%' AND NOT A LIKE '%er%'")

Upon executing this advanced query, the result set is further streamlined. The visual output below confirms that the dataset now only contains players who satisfy both complex, negative conditions simultaneously.

A14 =QUERY(A1:C11, "SELECT * WHERE NOT B LIKE '%uar%' AND NOT A LIKE '%er%'")

	A	B	C	D	E	F
1	**Team**	**Position**	**Points**			
2	Mavs	Guard	22			
3	Hawks	Guard	20			
4	Magic	Forward	19			
5	Thunder	Guard	15			
6	Kings	Forward	29			
7	Pacers	Forward	24			
8	Clippers	Forward	28			
9	Nuggets	Guard	20			
10	Lakers	Forward	14			
11	Celtics	Guard	12			
12						
13						
14	Team	Position	Points			
15	Magic	Forward	19			
16	Kings	Forward	29			
17						
18						
19						
20						
21						
22						

The filtered results clearly demonstrate the combined power of the operators. We have successfully removed players based on position (no "uar") AND removed players based on their team name (no "er"). This level of granular control is essential for targeted reporting and highly focused analysis within Google Sheets (4), allowing users to define complex data subsets by defining what must be absent.

Best Practices and Considerations for `NOT LIKE` Performance

While **`NOT LIKE`** is highly efficient, optimizing its usage through best practices ensures reliable and fast query execution, especially when dealing with increasingly large datasets in Google Sheets (5). A key consideration revolves around the handling of text string comparisons.

A critical aspect of string matching in the QUERY function (5), including **`NOT LIKE`** operations, is case sensitivity (1). By default, "Guard" is treated as distinct from "guard". If your goal is to exclude a pattern regardless of capitalization, you must standardize the case of the column data within the query itself. This is typically achieved by wrapping the column identifier in the `LOWER()` function, as

demonstrated here: `SELECT * WHERE NOT LOWER(B) LIKE '%uar%'`. This technique forces a case-insensitive (2) comparison, significantly improving the robustness of your pattern matching logic.

Furthermore, judicious use of the wildcard character (3) can impact performance. While `'%text%'` is common, if you know the pattern always appears at the start of the string, using `'text%'` is slightly more efficient as it reduces the search space. For extremely large ranges, avoid overly complex or numerous chained conditions unless absolutely necessary, as they increase the processing load. Always test performance on representative samples if speed becomes a bottleneck.

Finally, explore alternative operators that might complement or replace **`NOT LIKE`** depending on the complexity of your exclusion needs. For exclusion based on regular expressions (RegEx), the `MATCHES` operator offers maximal flexibility. If you are dealing with numerical data or exact text matches, standard operators like (not equal to) or `IS NOT NULL` remain the appropriate choice. However, for generalized pattern exclusion, **`NOT LIKE`** remains the most direct and readable method derived from SQL (2) syntax.

Conclusion: Mastering Data Exclusion with `NOT LIKE`

The **`NOT LIKE`** operator is a fundamental tool for any user seeking advanced control over data manipulation within SQL-like queries in Google Sheets. It provides an elegant and highly effective solution for filtering out data based on specific textual patterns rather than requiring exact matches. This capability is paramount for tasks such as data cleaning, isolating anomalies, and preparing refined datasets for advanced reporting.

By thoroughly understanding the interaction between **`NOT LIKE`** and the wildcard character (4) (the percentage sign), users can define precise exclusion boundaries. Furthermore, the ability to combine multiple exclusion rules using logical operators like the `AND` operator (4) elevates the complexity and specificity of achievable filters, allowing for intricate data governance.

We strongly encourage data practitioners to integrate **`NOT LIKE`** into their regular analytical workflow. Its straightforward implementation yields profound results in terms of data accuracy and focus, empowering you to quickly transform raw spreadsheet data into actionable insights by removing the noise and concentrating solely on the signals that matter most.

Further Learning Resources

To deepen your expertise in Google Sheets and advanced data manipulation, we recommend reviewing related documentation on filtering and logical control:

Review the official documentation for the QUERY function for a complete overview of all supported clauses, aggregation functions, and operators.

Explore the use of other Boolean operators, such as `OR`, within the QUERY statement to understand how different logical combinations affect data inclusion and exclusion.

Study techniques for managing case sensitivity (3) using functions like `LOWER()` and `UPPER()` to ensure uniform matching criteria across varied datasets.

Investigate the `MATCHES` operator to implement regular expressions for even more powerful and flexible pattern recognition than standard **`LIKE`** or **`NOT LIKE`** provides.

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