# Google Sheets Query: Use WHERE IN a List

Authored by

**Mohammed looti**

October 27, 2025

## Introduction: Achieving List-Based Filtering in Google Sheets

Efficient data manipulation is paramount in modern analysis. While Google Sheets provides numerous tools for working with tabular information, its powerful QUERY function stands out as the most flexible method for extracting and summarizing specific data. A frequent requirement when dealing with large volumes of data is the need to filter a dataset where a specific column value belongs to a predefined list of criteria. Standard filtering techniques often fall short when addressing this "WHERE IN" scenario, necessitating a more sophisticated approach.

This comprehensive guide is dedicated to mastering the exact method required to implement list-based filtering using the Google Sheets Query Language. We will demonstrate how to elegantly overcome the limitation of not having an explicit `WHERE IN` clause, similar to what is found in standard SQL. The key lies in leveraging the **`WHERE`** clause in conjunction with the highly versatile **`MATCHES`** operator and regular expressions (RegEx). This technique is not only concise but also dramatically enhances the precision and scalability of your data extraction workflows, allowing you to select specific rows based on complex, multiple criteria.

## The QUERY Function, WHERE Clause, and Regular Expressions

The core strength of the QUERY function is its ability to process data using a SQL-like syntax, enabling powerful relational operations directly within your spreadsheet. It requires two main inputs: the source range containing the data you wish to analyze, and the query string itself, which dictates how the data should be selected, filtered, and organized. The query string houses critical components like **`SELECT`**, **`GROUP BY`**, and, most importantly for filtering, the **`WHERE`** clause.

The **`WHERE`** clause is the gatekeeper of your query results, ensuring that only rows that meet predefined logical conditions are returned. While simple comparisons (e.g., equality or inequality) are straightforward, filtering a single column against many possible values--a typical "IN list" requirement--demands the use of advanced pattern matching. This is precisely where the **`MATCHES`** operator comes into play, utilizing the power of regular expressions to define complex search patterns.

To emulate the behavior of a `WHERE IN` clause, we structure the values we are searching for into a single, cohesive regular expression pattern. This pattern uses the pipe symbol (**`|`**), which serves as a logical "OR" operator within the RegEx syntax. This allows the query to match a row if the specified column contains **value1** OR **value2** OR **value3**, and so on. The resulting formula is highly efficient and remarkably clear:

**=QUERY(A1:C11, "SELECT * WHERE A MATCHES '(value1|value2|value3)'")**

In the example above, the query retrieves all entries from the specified range, **A1:C11**, where the value in column A is an exact match for any of the values listed within the parentheses. This single line of code effectively replaces what would otherwise be a long chain of `OR` conditions, providing a scalable solution for filtering criteria.

## Filtering Text Data: The Basketball Example

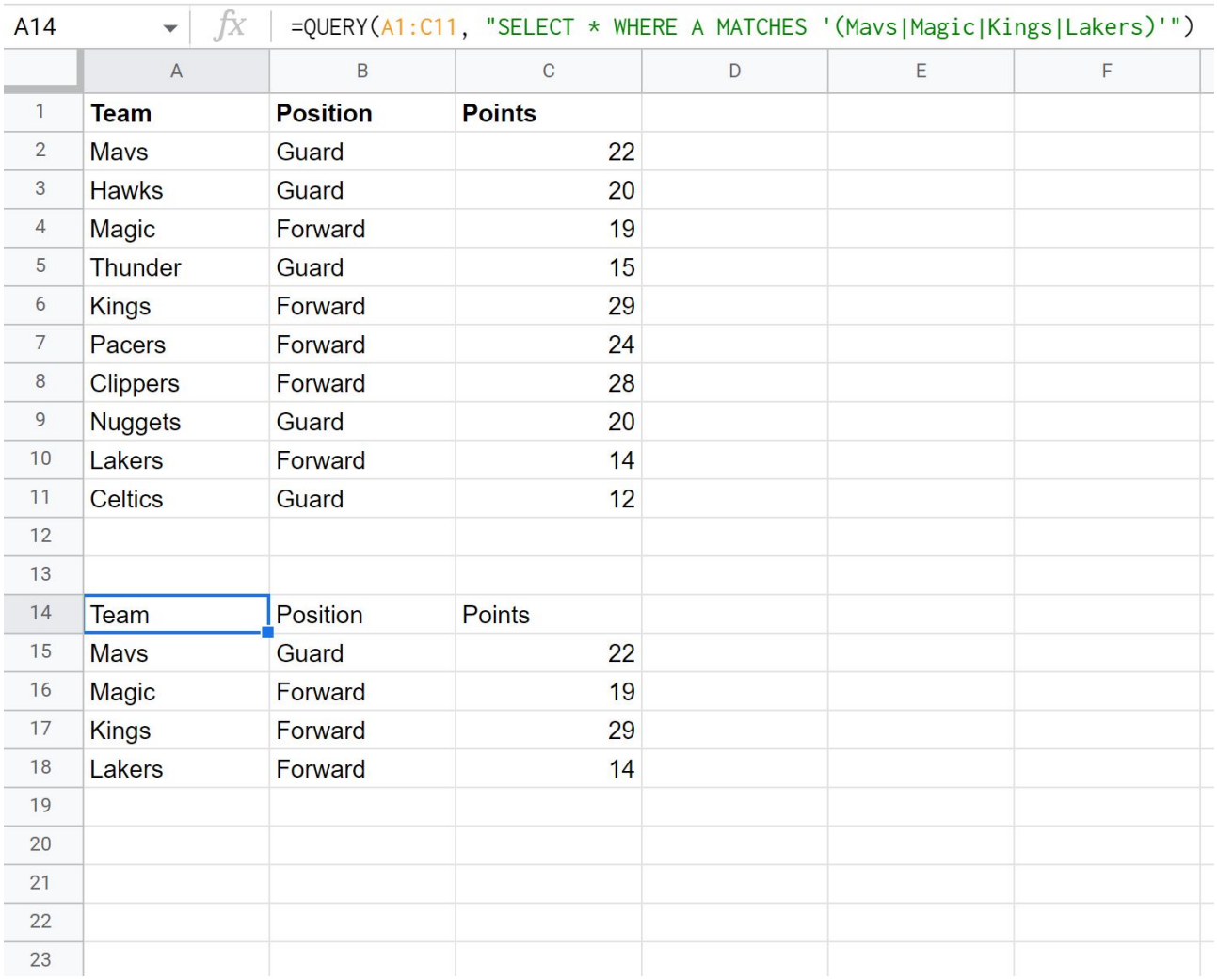
To better understand this powerful technique, let us consider a real-world scenario involving a sports dataset. Suppose we are managing a spreadsheet detailing basketball player statistics, including columns for "Player," "Team," and "Points." Our immediate goal is to generate a filtered view that includes statistics only for a specific collection of teams we are interested in tracking.

| | 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 | | | | |

If our data resides in the range **A1:C11**, and the "Team" information is located in column A, we can construct a precise query to isolate the teams "Mavs," "Magic," "Kings," and "Lakers." The elegance of the regular expression approach allows us to combine these four conditions into one concise filter string.

**=QUERY(A1:C11, "SELECT * WHERE A MATCHES '(Mavs|Magic|Kings|Lakers)'")**

Upon executing this query, Google Sheets processes the request, applying the RegEx pattern to every cell in column A. If the cell content matches any of the listed team names exactly, the entire row is included in the output. The visual result clearly demonstrates how the original dataset is efficiently condensed to include only the desired entries, proving the method's effectiveness for text-based criteria.




| | 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 | Mavs | Guard | 22 | | | |
| 16 | Magic | Forward | 19 | | | |
| 17 | Kings | Forward | 29 | | | |
| 18 | Lakers | Forward | 14 | | | |
| 19 | | | | | | |
| 20 | | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | | | | | |

## Applying `WHERE IN` Logic to Numeric Data

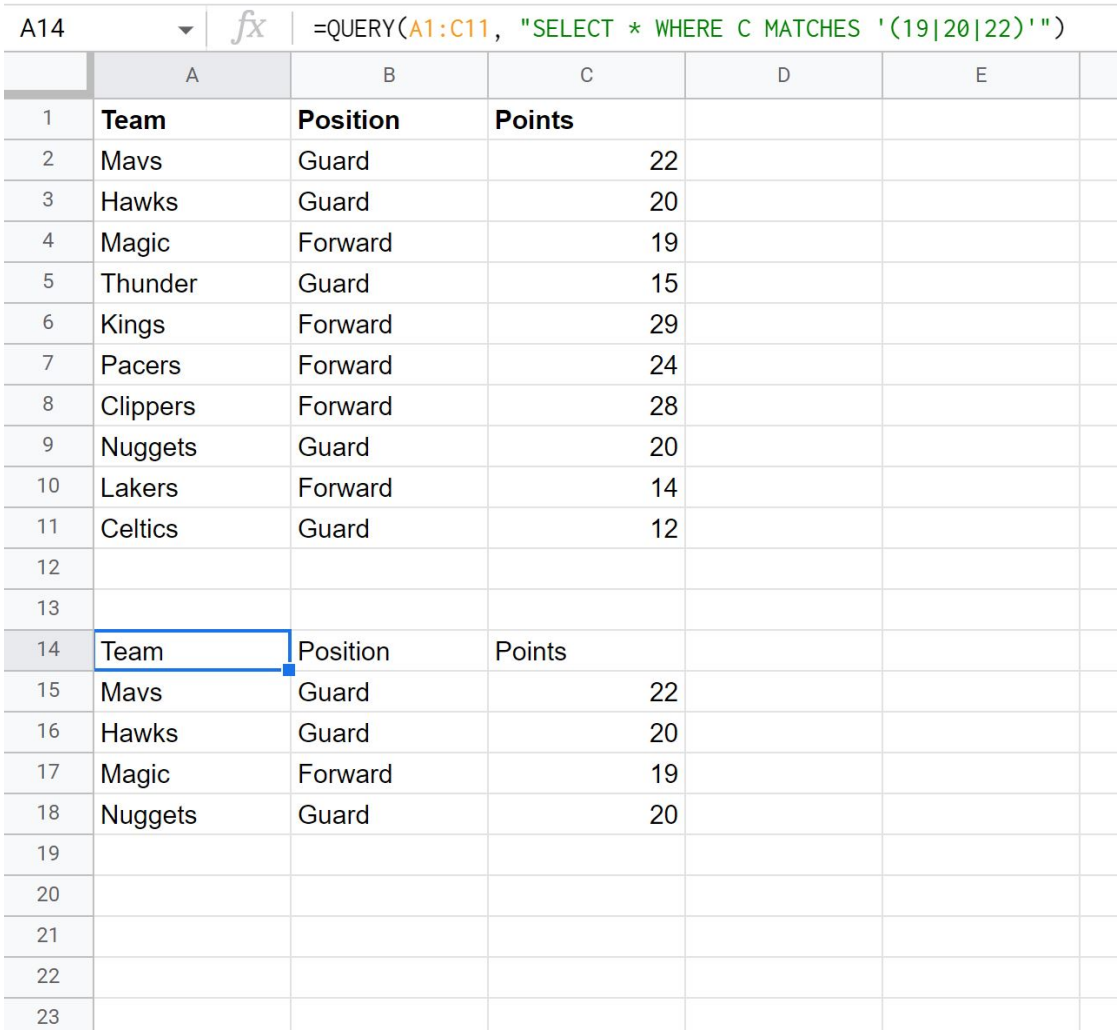
The utility of the **`MATCHES`** operator is not restricted solely to filtering text strings; it functions just as effectively when isolating specific numerical values. This capability is essential for data analysis tasks that require extracting rows based on a defined set of performance metrics, unique identifiers, or other discrete numerical values.

Returning to our basketball dataset, let us now focus on the "Points" column, which corresponds to column C in our data range **A1:C11**. Suppose we need to retrieve all players who scored exactly

**19**, **20**, or **22** points. Since the QUERY function treats all values within the **`MATCHES`** clause as strings for pattern matching purposes, the syntax remains identical to the text-filtering examples.

**=QUERY(A1:C11, "SELECT * WHERE C MATCHES '(19|20|22)'")**

The implementation of this formula provides immediate and accurate results. As shown in the screenshot below, the output table is dynamically generated to display only those entries where the numerical value in column C precisely fits the criteria defined in the regular expression pattern. This demonstrates the seamless adaptability of the RegEx approach for handling both categorical and numerical list filtering within Google Sheets.




| | 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 | Mavs | Guard | 22 | | |
| 16 | Hawks | Guard | 20 | | |
| 17 | Magic | Forward | 19 | | |
| 18 | Nuggets | Guard | 20 | | |
| 19 | | | | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |
| 23 | | | | | |

## Advanced Considerations and Optimization Strategies

While the **`MATCHES`** operator offers a robust solution for list filtering, maximizing its efficiency and reliability requires attention to several advanced details, particularly concerning case sensitivity and

potential performance bottlenecks in large datasets. Applying these best practices ensures your Google Sheets formulas remain clean, fast, and maintainable.

**Handling Case Sensitivity:** It is crucial to remember that the **`MATCHES`** operator in the SQL-like query language is inherently case-sensitive. If your search list contains 'apple' but your data contains 'Apple', no match will occur. To enforce a case-insensitive search, you must normalize the case of the data being queried using functions like **`LOWER()`** or **`UPPER()`** directly within the QUERY string (e.g., `WHERE LOWER(A) MATCHES '(valuea|valueb)'`).

**Performance and Scale:** For data ranges containing thousands of entries, extensive use of complex regular expressions can potentially introduce latency. While the RegEx approach is superior for moderate lists, if you are filtering against hundreds of items, or if performance becomes a concern, alternative methods should be explored. These might include preparing a helper column that flags matching items using functions like **`VLOOKUP`** or **`MATCH`** combined with **`ARRAYFORMULA`**, before querying the helper column.

**Escaping Special Characters:** If any item in your filter list contains characters that are reserved in regular expressions (such as `.`, `*`, `+`, `?`, or parentheses `()`), you must precede them with a backslash () to treat them as literal characters. Failure to escape these characters will cause the query to misinterpret your filter criteria, leading to inaccurate results or formula errors.

**The Alternative OR Chain:** While the **`MATCHES`** operator is highly recommended for its conciseness, especially with long lists, you could technically achieve the same result using explicit **`OR`** operators (e.g., `WHERE A = 'value1' OR A = 'value2'`). However, this method quickly becomes cumbersome, prone to errors, and significantly reduces the readability of the formula as the number of criteria increases, making the RegEx approach the preferred standard for list-based filtering.

## Conclusion: Mastering Efficient Data Extraction

The effective use of the **`MATCHES`** operator in combination with regular expressions is a fundamental skill for anyone seeking to master the Google Sheets Query Language. This powerful technique provides a clean, scalable, and elegant solution for the common requirement of filtering data based on whether a value in a column is present within a predefined list of criteria.

By structuring your multiple criteria using the logical "OR" symbol (**`|`**) within the RegEx pattern, you gain the ability to conduct complex, multi-criteria filtering operations in a single line of code. This dramatically streamlines data analysis workflows, reduces formula complexity, and ensures higher precision in your data extraction tasks. Incorporating this method into your toolkit will unlock new levels of efficiency and help you derive quicker insights from your tabular data within Google Sheets.

## Further Resources for Google Sheets Mastery

To continue developing your proficiency in advanced data management and analysis using Google Sheets, these resources focus on related functions and advanced query operations:

Explore complex aggregation techniques using the SQL `GROUP BY` clause within the QUERY function for summarization and reporting.
Learn how to manipulate and transform raw data using text manipulation functions to ensure data quality before running complex queries.
Discover methods for structuring and optimizing your data models to improve performance when dealing with high volumes of information.