

Google Sheets: How can I filter a column by multiple values?

Authored by
stats writer

November 23, 2025

RECOMMENDED CITATION

stats writer (2025). *Google Sheets: How can I filter a column by multiple values?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=100127>

The Challenge of Multi-Criteria Filtering in Google Sheets

One of the most frequent challenges users face when managing large datasets in Google Sheets is the need to filter records based on multiple, specific values within a single column. While standard filter views are excellent for simple filtering (e.g., matching a single criterion), they become cumbersome when attempting to isolate data points that match any one of several defined criteria simultaneously. Manually selecting dozens of specific entries often leads to errors and is highly inefficient. Fortunately, Google Sheets offers powerful function combinations that automate this complex process, providing dynamic, scalable solutions for data analysis.

The most robust and streamlined approach involves integrating the **FILTER function** with the **REGEXMATCH function**. This combination harnesses the power of Regular Expressions to define complex search patterns, allowing you to specify a list of acceptable values for a column in a concise manner. This method avoids the necessity of linking multiple logical `OR` conditions, which can make formulas extremely long and difficult to maintain.

Below is the foundational formula structure used to efficiently filter a dataset based on whether a column contains any of the specified values. This formula is highly adaptable, requiring only changes to the data range and the list of desired strings within the regular expression pattern. Understanding this structure is key to mastering complex data manipulation tasks within your spreadsheets.

Core Formula: Combining FILTER and REGEXMATCH

To successfully filter your data based on multiple input strings, you must first understand the role of the two primary functions involved. The **FILTER function** is designed to return a filtered version of the source range, contingent upon one or more conditional tests. The range is the entire dataset you wish to analyze, while the conditions are typically boolean arrays (TRUE/FALSE) derived from tests applied to individual columns.

The conditional test in this specific scenario is generated by the **REGEXMATCH function**. This function checks if a piece of text matches a provided Regular Expression. Critically, when given a range (like `A1:A11`), it returns an array of TRUE or FALSE values--a perfect input for the **FILTER function**. The beauty of Regular Expressions is the vertical bar symbol (|), which acts as the logical 'OR' operator, allowing us to list all desired values in a single string.

The following basic formula illustrates how this powerful combination is structured. It instructs Google Sheets to display the rows from the target range (A1:C11) only where the corresponding cell in the condition column (A1:A11) matches any one of the three specified strings.

```
=FILTER(A1:C11, REGEXMATCH(A1:A11, "string1|string2|string3"))
```

This particular formula will filter the rows in the range **A1:C11** to only show the rows where the value in the range **A1:A11** is equal to either string1, string2, or string3. This ensures that the output dataset is highly targeted and relevant to the specific criteria defined by the user.

Practical Application: Filtering a Basketball Dataset

To fully appreciate the utility of this filtering technique, let us apply it to a practical dataset. Imagine you are analyzing sports statistics and have a table detailing various attributes of basketball players, including their names, positions, and the teams they play for. Our objective is to isolate only those players belonging to a select group of teams, perhaps for comparative analysis or specific reporting.

Suppose we have the following dataset that contains information about various basketball players. The data range covers columns A, B, and C, starting from row 1.

	A	B	C	D
1	Team	Position	Points	
2	Mavs	Guard	22	
3	Mavs	Forward	29	
4	Celtics	Guard	20	
5	Heat	Guard	34	
6	Celtics	Center	36	
7	Mavs	Forward	19	
8	Warriors	Forward	15	
9	Heat	Center	29	
10	Celtics	Forward	25	
11	Nuggets	Guard	20	
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

This dataset, ranging from A1 to C11, represents the full population of data we are working with. Our task is to perform a surgical extraction, focusing specifically on rows where the 'Team' column (Column A) meets our multi-value criteria.

Now suppose we'd like to filter the dataset to only show rows where the Team column contains 'Heat' or 'Celtics'. This represents a common use case where data analysts need to swiftly extract information pertaining to specific organizational units or categories from a much larger master list.

Executing the Multi-Value Filter Formula

The implementation of the formula is straightforward once the ranges and target values are correctly identified. Our data range is **A1:C11** (the entire table we want returned). Our condition column is **A1:A11** (the column containing the team names). Our criteria are 'Heat' and 'Celtics'. We combine these criteria into a single regular expression string separated by the pipe symbol (|).

To perform this task, we input the following formula into a blank cell outside the original data range (e.g., Cell E1). The resulting filtered table will dynamically update whenever the source data changes, offering a powerful, automated view of the data.

=FILTER(A1:C11, REGEXMATCH(A1:A11, "Heat|Celtics"))

The result of executing this formula showcases the efficiency of using REGEXMATCH function for logical OR operations. Instead of writing `(A1:A11="Heat")+(A1:A11="Celtics")`, which can become unwieldy with many conditions, the regular expression handles all conditions simultaneously within a clean, delimited string.

The following screenshot demonstrates the output of this formula when applied to the basketball dataset. Notice how the resulting table perfectly isolates the relevant rows, excluding teams like 'Rockets' or 'Lakers'.

	A	B	C	D	E
A14	=FILTER(A1:C11, REGEXMATCH(A1:A11, "Heat Celtics"))				
1	Team	Position	Points		
2	Mavs	Guard	22		
3	Mavs	Forward	29		
4	Celtics	Guard	20		
5	Heat	Guard	34		
6	Celtics	Center	36		
7	Mavs	Forward	19		
8	Warriors	Forward	15		
9	Heat	Center	29		
10	Celtics	Forward	25		
11	Nuggets	Guard	20		
12					
13					
14	Celtics	Guard	20		
15	Heat	Guard	34		
16	Celtics	Center	36		
17	Heat	Center	29		
18	Celtics	Forward	25		
19					
20					
21					
22					
23					

Observe that the filtered dataset only contains the rows where the team is equal to 'Heat' or 'Celtics'. This confirms the successful implementation of the multi-value inclusion filter using the combined power of **FILTER** and **REGEXMATCH**.

Advanced Filtering: Excluding Multiple Values using NOT

While filtering to include specific values is highly useful, data analysis often requires the opposite: filtering to exclude a set of unwanted values. For example, if you want to analyze all teams **except** the 'Heat' and the 'Celtics', you can easily adapt the existing structure by incorporating the **NOT function**.

The **NOT function** simply reverses a logical value. If the **REGEXMATCH function** returns TRUE (meaning the cell contains 'Heat' or 'Celtics'), wrapping it in NOT() makes it FALSE, thereby

excluding that row from the final FILTER function output. Conversely, if the cell does not match the criteria (e.g., it contains 'Rockets'), REGEXMATCH returns FALSE, which NOT() converts to TRUE, ensuring the row is included.

The formula structure remains nearly identical, with the key difference being the simple placement of **NOT()** immediately surrounding the **REGEXMATCH** clause. This small modification transforms the formula from an inclusion filter into a powerful exclusion filter.

Also note that you can filter for rows where the team name is **not equal** to 'Heat' or 'Celtics' by using the NOT function outside of the **REGEXMATCH** function as follows:

```
=FILTER(A1:C11, NOT(REGEXMATCH(A1:A11, "Heat|Celtics")))
```

The following screenshot shows how the implementation of the **NOT** function alters the output, displaying all rows that fall outside the specified criteria.

	A	B	C	D	E
1	Team	Position	Points		
2	Mavs	Guard	22		
3	Mavs	Forward	29		
4	Celtics	Guard	20		
5	Heat	Guard	34		
6	Celtics	Center	36		
7	Mavs	Forward	19		
8	Warriors	Forward	15		
9	Heat	Center	29		
10	Celtics	Forward	25		
11	Nuggets	Guard	20		
12					
13					
14	Team	Position	Points		
15	Mavs	Guard	22		
16	Mavs	Forward	29		
17	Mavs	Forward	19		
18	Warriors	Forward	15		
19	Nuggets	Guard	20		
20					
21					
22					
23					

Deep Dive into Regular Expressions for Filtering

The efficiency of the [FILTER function](#) using **REGEXMATCH** hinges entirely on the syntax of [Regular Expressions](#). While the use of the vertical bar (|) for simple OR matching is fundamental, understanding additional syntax elements allows for far more granular control over the filtering process. For instance, if you need to ensure the column value matches the specified string exactly and is not just a substring, you should anchor the expression.

To ensure an exact match (e.g., matching "Celtics" but not "Celtics Jr."), you must use the start-of-string anchor (^) and the end-of-string anchor (\$). The formula used in the previous example implicitly assumes exact matching because the data contains single, distinct team names, but for user-generated text fields, explicit anchoring is crucial. An exactly matched expression would look like: `"^Heat|Celtics$"`. Note that in this context, the anchors must surround the entire pattern if you want the whole column value to match **one** of the strings exactly, or they can surround each string individually depending on the exact requirement. For simple whole-string matching across multiple values, the expression `"^(Heat|Celtics)$"` is generally safest.

Furthermore, [Regular Expressions](#) allow for case-insensitive filtering. By default, **REGEXMATCH** in [Google Sheets](#) is case-sensitive. If you wish to filter for 'heat' regardless of capitalization, you would typically need to convert the column to a consistent case (e.g., using `LOWER(A1:A11)`) before applying the match, or utilize more advanced regex syntax if available in the specific environment. However, for most basic matching tasks involving clean, consistent category data, the simple pipe operator (|) provides the necessary power.

Alternative Filtering Method: Leveraging the QUERY Function

While the FILTER/REGEXMATCH combination is highly effective, especially for dynamic or complex criteria, an alternative method exists for multi-criteria filtering: the **QUERY function**. The QUERY function is perhaps the most powerful tool in [Google Sheets](#), as it allows users to write SQL-like queries directly within the spreadsheet environment.

For filtering a single column by multiple values, the QUERY function uses the `WHERE` clause combined with the `OR` keyword or the `MATCHES` keyword. If we assume Column A is the team column (represented as 'Col1' within the query language), the inclusion formula looks like this:

```
=QUERY(A1:C11, "SELECT * WHERE Col1 = 'Heat' OR Col1 = 'Celtics'", 1)
```

The exclusion logic is equally clear, utilizing the `NOT` operator within the `WHERE` clause:

```
=QUERY(A1:C11, "SELECT * WHERE NOT (Col1 = 'Heat' OR Col1 = 'Celtics')", 1)
```

While the **QUERY function** syntax might be more familiar to users with database experience, the **FILTER/REGEXMATCH** method often proves faster for simple OR-based multi-criteria filtering because it avoids the overhead of parsing the SQL-like string, making it a preferred choice when only a single condition needs to be tested across multiple potential matches.

Best Practices for High-Performance Filtering

When working with extensive datasets in Google Sheets, optimization is crucial. Using dynamic formulas like **FILTER function** and **REGEXMATCH** efficiently requires adherence to several best practices to maintain spreadsheet speed and accuracy.

Define Explicit Ranges: Avoid using open-ended ranges like `A:C` unless absolutely necessary. Filtering on `A1:C1000` is significantly faster than filtering on `A:C`, as the latter forces Google Sheets to process thousands of empty rows.

Minimize Regex Complexity: While Regular Expressions are powerful, complex patterns consume more computational resources. For simple OR conditions, keep the string concise (e.g., `"Value1|Value2|Value3"`). If you have dozens of criteria, consider listing them in a helper column and using `JOIN("|", range)` to dynamically create the regex string, which improves readability and maintenance.

Isolate Formulas: Place complex output formulas like **FILTER** in dedicated summary tabs or cells that are not frequently recalculating other parts of the sheet. This isolates the computation load and prevents cascading recalculations that can slow down overall sheet performance.

By following these best practices, you can ensure that your multi-value filtering solutions remain responsive and accurate, even as your underlying data grows in size and complexity. The combined power of **FILTER** and **REGEXMATCH** offers a superior alternative to manual filtering or overly complex nested logical functions.

Summary of Multi-Value Filtering Techniques

Filtering a column by multiple values in Google Sheets is a fundamental skill for advanced data management. We have demonstrated that the most efficient method for handling inclusion (OR logic) is the combination of the **FILTER function** and the **REGEXMATCH function**, utilizing the pipe delimiter (`|`) within the regular expression pattern to list all desired matches.

Furthermore, we explored how to invert this logic using the **NOT function**, allowing users to quickly exclude a list of unwanted values. Finally, the **QUERY function** was presented as a viable alternative, particularly for those familiar with standard database query syntax. Mastery of these techniques ensures dynamic and accurate data extraction, making your Google Sheets analysis more scalable and professional.