

# How to Filter a Column in Google Sheets Based on Another Column's Values

Authored by  
**stats writer**

February 1, 2026

## RECOMMENDED CITATION

stats writer (2026). *How to Filter a Column in Google Sheets Based on Another Column's Values*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128934>

In the realm of data management, efficiently isolating specific subsets of information is paramount for effective analysis and reporting. When utilizing Google Sheets, a common requirement is filtering the contents of one column based on specific criteria derived from the values present in another, distinct column. While Google Sheets offers standard graphical filtering tools, these methods often fall short when dynamic, criteria-based filtering using a list of values is necessary. This powerful technique involves employing a combination of specialized functions to dynamically generate a filtered output range, ensuring that only records meeting the precise conditions are displayed. Mastering this methodology significantly enhances your capability to organize, manipulate, and analyze complex datasets within the spreadsheet environment, moving beyond static data views toward dynamic reporting.

The conventional approach of manually applying filters through the toolbar is often cumbersome, especially when the list of desired criteria is extensive or frequently changing. Furthermore, standard filtering only hides rows; it does not produce a separate, clean output table suitable for further calculation or visualization. To achieve a truly dynamic, formula-driven filter--where the results automatically update as the source data or criteria list changes--we must leverage the robust array manipulation capabilities native to Google Sheets. This article focuses on the definitive method for cross-column filtering, specifically utilizing the powerful combination of the FILTER function and the MATCH function, a technique favored by advanced spreadsheet users for its efficiency and adaptability across diverse analytical tasks.

## Advanced Data Extraction in Google Sheets: Filtering Columns Dynamically

### The Foundational Methodology: FILTER and MATCH Functions

The most effective and dynamic way to filter data in Google Sheets based on criteria defined in a separate list is by combining two fundamental functions: the FILTER function and the MATCH function. This powerful pairing allows the user to specify a range of data they wish to return (the output column or columns) and then provide a boolean array (True/False list) generated by the MATCH function, which dictates which rows meet the comparison conditions. This technique is vastly superior to manual filtering for reproducible reporting and sophisticated data validation tasks, as the filtered results update instantaneously whenever source data or the reference criteria list is modified.

Specifically, the FILTER function requires two primary arguments: first, the range of data you want returned, and second, the condition (or set of conditions) that must be met for each row in that range. This condition must resolve to an array of boolean values (TRUE or FALSE). The role of the MATCH function in this scenario is critical; it is used to check for the existence of each value in the

main dataset column against the values listed in the separate criteria column. If a match is found, `MATCH` returns the position of the match (a number), and if no match is found, it returns the error `#N/A`. By strategically processing these outputs, we can transform the results into the required boolean array for the `FILTER` function.

To illustrate the necessary syntax, consider a scenario where we want to filter the data range **A2:B12**. We want to apply a conditional check against the values in column A (specifically **A2:A12**) to ensure they match any of the values listed in our criteria range, **D2:D4**. The combined formula structure achieves this comparison elegantly, resulting in a filtered table containing only the relevant rows. This technique ensures that complex, multi-criteria filtering operations remain clean, readable, and highly functional within any extensive [Google Sheets](#) workbook.

## Understanding the FILTER Function Syntax

The `FILTER` function is designed to return a filtered version of the source range, based on specific conditions applied to corresponding rows. Its standard syntax is straightforward: `=FILTER(range_to_filter, condition1, )`. The `range_to_filter` is the entire block of data you wish to output, including all columns you want to see in the final result. In our example, this is **A2:B12**, encompassing both the criteria column and the associated data column(s).

The true complexity lies in defining the `condition` argument. When filtering based on a list of items, we cannot simply use comparison operators like `=` or `>`, as these only work for single value comparisons. Instead, the condition must be an array that has the same number of rows as the `range_to_filter`. Each cell in this condition array must contain either `TRUE` (to keep the row) or `FALSE` (to discard the row). Because the `FILTER` function is an array function, it automatically expands the results into the adjacent cells, making it a perfect tool for creating dynamic report tables.

When used independently, the `FILTER` function is often used for simple tasks, such as filtering a list of sales based only on values greater than \$1000. However, its true power is unleashed when it is nested with other array-processing functions like `MATCH`, which can generate the necessary complex conditional array. By wrapping the `MATCH` function output--which contains numerical positions or error codes--within the `ISNUMBER` function (or relying on Google Sheets' implicit error handling), we successfully convert the row-by-row match results into the required boolean format (`TRUE` or `FALSE`), preparing it for the external `FILTER` function wrapper.

## The Role of the MATCH Function in Conditional Filtering

The `MATCH` function is typically used to locate the relative position of an item in a range. Its syntax is `=MATCH(search_key, range, search_type)`. When used for filtering, the `search_key` is the

column in your main dataset that you wish to compare (e.g., **A2:A12**), and the `range` is the separate column containing your specific filtering criteria (e.g., **D2:D4**).

Crucially, when we feed an entire range (like **A2:A12**) into the `search_key` argument of `MATCH`, it automatically operates as an array formula, checking each cell in **A2:A12** against the list in **D2:D4**. If a value in **A2:A12** is found in **D2:D4**, `MATCH` returns a number representing its position within the criteria range (**D2:D4**). If the value is not found, `MATCH` returns the error value `#N/A`. We specify a `search_type` of 0, which dictates an exact match, essential for accurate criteria matching.

The resulting array from the `MATCH` function operation will be a mixture of numbers and `#N/A` errors. Since `FILTER` accepts boolean values, the numerical results (positions) are interpreted as `TRUE` (keep the row), and the `#N/A` errors are interpreted as `FALSE` (discard the row). Although using `ISNUMBER(MATCH(...))` makes the logic explicit, the concise formula works because Google Sheets manages this conversion internally when the `MATCH` output is used directly as the filter condition, as demonstrated in the example provided.

## Constructing the Combined Formula

To execute this advanced filtering, we must nest the functions correctly to ensure the logical flow from comparison to output. The structure is designed to be highly readable and immediately functional. The core formula links the data range (what you want to see) with the conditional check (the `MATCH` result).

The definitive formula structure used to filter the range **A2:B12** for rows where the value in **A2:A12** matches any value from the criteria list in **D2:D4** is as follows:

```
=FILTER(A2:B12,MATCH(A2:A12,D2:D4,0))
```

This single array formula, when entered into the output cell, instructs Google Sheets to evaluate every row in the primary range. For each row, the `MATCH` function attempts to find the corresponding cell value from column A within the specified criteria list in column D. The success or failure of this match determines whether the row is included in the final filtered output, thus achieving highly efficient and dynamic data extraction.

## Practical Example: Filtering Basketball Data

The following example demonstrates the practical application of this dynamic filtering technique using a sample dataset related to basketball statistics. We aim to extract specific players based solely on a predefined list of teams.

Suppose we have the following initial dataset, spanning columns A and B, which contains

information about various basketball players, including their team affiliations and points scored:

	A	B	C	D
1	<b>Team</b>	<b>Points</b>		
2	Mavs	22		
3	Warriors	14		
4	Mavs	29		
5	Lakers	34		
6	Kings	28		
7	Lakers	14		
8	Nets	17		
9	Celtics	24		
10	Warriors	30		
11	Rockets	31		
12	Wizards	18		
13				
14				
15				

Next, we define our criteria list. This list, placed in a separate column (Column D in this scenario), dictates exactly which teams we are interested in isolating from the main dataset. This external location for the criteria list makes future updates simple and clean, as you only need to edit the teams listed in **D2:D4** to instantly change the filtered results. Our goal is to filter the original table to only include rows where the team name in column A is present in the specified criteria list:

	A	B	C	D
1	<b>Team</b>	<b>Points</b>		<b>Team</b>
2	Mavs	22		Mavs
3	Warriors	14		Lakers
4	Mavs	29		Celtics
5	Lakers	34		
6	Kings	28		
7	Lakers	14		
8	Nets	17		
9	Celtics	24		
10	Warriors	30		
11	Rockets	31		
12	Wizards	18		
13				
14				
15				
16				

## Executing the Formula and Analyzing Results

To perform this precise filtering operation, we implement the combined [FILTER function](#) and [MATCH](#) formula. We will place the entire formula into an empty cell outside the existing data ranges, such as cell **A14**. This location ensures the resulting filtered table does not overwrite or interfere with the source data or the criteria list.

The formula entered into cell **A14** is:

**=FILTER(A2:B12,MATCH(A2:A12,D2:D4,0))**

Upon execution, the `MATCH(A2:A12, D2:D4, 0)` segment generates an array where values corresponding to the "Hawks," "Rockets," and "Blazers" return a number (position 1, 2, or 3), and all other teams return #N/A. The `FILTER` function then interprets these numerical matches as `TRUE` and the errors as `FALSE`, thereby only returning the rows associated with the matched teams from the `A2:B12` range.

The following screenshot displays the resulting output, demonstrating how the dataset is dynamically filtered to include only the basketball players associated with the teams listed in the criteria range (D2:D4):

A14     $\text{fx}$  =FILTER(A2:B12, MATCH(A2:A12, D2:D4, 0))

	A	B	C	D
1	<b>Team</b>	<b>Points</b>		<b>Team</b>
2	Mavs	22		Mavs
3	Warriors	14		Lakers
4	Mavs	29		Celtics
5	Lakers	34		
6	Kings	28		
7	Lakers	14		
8	Nets	17		
9	Celtics	24		
10	Warriors	30		
11	Rockets	31		
12	Wizards	18		
13				
14	Mavs	22		
15	Mavs	29		
16	Lakers	34		
17	Lakers	14		
18	Celtics	24		
19				
20				

Notice that the filtered dataset successfully isolates rows where the team name in column A is an exact match for one of the team names specified in column D. This result confirms the efficiency of the `FILTER(MATCH)` array formula combination for handling complex, list-based filtering requirements in Google Sheets.

### Alternative Filtering Techniques: QUERY vs. Native Filters

While the `FILTER(MATCH)` combination is highly effective for list-based criteria, it is important to acknowledge other filtering methods available within Google Sheets, each suited for different analytical contexts.

The Query function is Google Sheets' most powerful data manipulation tool, utilizing a syntax similar to SQL. If you are dealing with very large datasets or require complex aggregations alongside filtering, `QUERY` is often the preferred choice. To replicate the filtering exercise above using `QUERY`, the syntax would involve combining the data selection with a `WHERE` clause and the `matches` or `contains` operators, often using explicit `OR` statements to include multiple criteria. For

**example:** `=QUERY(A2:B12, "SELECT A, B WHERE A = 'Hawks' OR A = 'Rockets' OR A = 'Blazers'", 0)`. While incredibly versatile, `QUERY` requires the criteria list to be manually embedded into the formula string, making it less dynamic than the `FILTER(MATCH)` approach where the criteria list (Column D) can be changed independently of the formula logic.

For immediate, non-permanent filtering needs, Google Sheets provides built-in Filter Views. These are especially useful when multiple users need to view the same data filtered differently without affecting what others see. To filter by a list using the native UI, one would select the filter icon on the header row, choose 'Filter by values,' and then manually select or paste the required list items. However, this is a static, manual process that does not automatically update with changes in the source data or criteria list, limiting its utility for automated reporting or complex data analysis pipelines. The `FILTER(MATCH)` method, conversely, is dynamic and ideal for building permanent, self-updating data subsets.

The `FILTER` function in Google Sheets remains the backbone of dynamic array manipulation, and when combined with the positional power of `MATCH`, it provides a highly reliable method for advanced conditional data management, allowing users to precisely control which data subsets are extracted and presented.

**Note:** You can find the complete documentation for the `FILTER` function in Google Sheets [here](#).

## Further Learning and Related Tutorials

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

How to use the `ARRAYFORMULA` function to apply calculations across entire columns efficiently.

Techniques for using `VLOOKUP` or `INDEX/MATCH` for multi-criteria lookups within large datasets.

Advanced applications of the `QUERY` function for complex grouping and aggregation tasks.

Methods for integrating spreadsheet data with external sources using `IMPORTDATA` or similar functions.