Filtering Data in Google Sheets: A Guide to Using Multiple Conditions

Authored by

Mohammed looti

October 31, 2025

The Power of Conditional Filtering in Data Analysis

The ability to efficiently analyze and segment large datasets is fundamental to effective data management. While standard sorting and filtering tools provided by **Google Sheets** offer basic functionality, extracting data that satisfies multiple, complex criteria requires a more powerful solution. This is where the dedicated **FILTER function** becomes indispensable. By mastering the `FILTER` function, users can dynamically pull specific subsets of rows that meet precise conditional requirements, significantly streamlining reporting and analysis processes.

Understanding how to structure these conditional statements is crucial. Unlike native interface filters, the `FILTER` function allows you to embed sophisticated **Boolean logic** directly into the formula, enabling dynamic results that automatically update as the source data changes. We will explore the two primary methods for applying multiple conditions: the implicit **AND** operator (requiring all conditions to be true) and the explicit **OR** operator (requiring at least one condition to be true).

These techniques move beyond simple data selection, allowing analysts to construct powerful, reusable extraction queries. Whether you need to find records that match two criteria simultaneously or those that satisfy one of several possible outcomes, the methods outlined below provide the control necessary for advanced spreadsheet manipulation within **Google Sheets**.

Method 1: Applying the AND Condition for Precise Filtering

The most common requirement in data filtering is to find rows that satisfy every criterion simultaneously. In the context of the **FILTER function**, this logical requirement, known as the **AND** condition, is implemented by separating each conditional array with a simple comma. Each comma acts as a logical gate; if any condition evaluates to `FALSE` for a given row, that row is excluded from the final filtered output.

The syntax for the `FILTER` function requires two main components: the **data range** you wish to return, followed by one or more conditional arrays that must have the same number of rows as the data range. For example, if we want to filter the data within the range `A1:C10` to only include rows where column A equals "A" **AND** column C is less than 20, the formula is structured as follows:

=FILTER(A1:C10, A1:A10="A", C1:C10<20)

This formula executes a sequence of checks on the source data. The first argument, `A1:C10`, specifies the output data block. The second argument, `A1:A10="A"`, generates an array of `TRUE` or `FALSE` values indicating whether the cell in column A equals "A". The third argument, `C1:C10<20`, generates a second array indicating whether the value in column C is less than 20. Because these

arrays are separated by commas, the **FILTER function** only returns a row if the corresponding position in **both** the second and third arrays is `TRUE`. This method is the foundational approach for creating highly specific data subsets.

Method 2: Utilizing the OR Condition with Boolean Addition

Implementing the **OR** logical requirement--where a row is returned if it satisfies Condition 1 **OR** Condition 2 (or both)--requires a different approach in **Google Sheets**. Unlike the **AND** condition, which uses commas, the **OR** condition is simulated using arithmetic addition (`+`) on the conditional arrays. This relies on the fundamental principles of **Boolean logic** within array operations: when treated numerically, `TRUE` is interpreted as the number 1, and `FALSE` is interpreted as the number 0.

When you add two conditional arrays together, the result is an array of sums. If a row satisfies the first condition (1) and fails the second (0), the sum is 1. If it fails the first (0) and satisfies the second (1), the sum is 1. If it satisfies both (1+1), the sum is 2. Critically, if it fails both (0+0), the sum is 0. Since the `FILTER` function considers any non-zero value (1 or 2) as `TRUE`, adding the conditional arrays effectively implements the **OR** logic. Parentheses are mandatory to ensure that each condition is evaluated as a separate array before the addition operation occurs.

To filter the range `A1:C10`, returning rows where column A equals "A" **OR** column C is less than 20, the formula uses the addition operator:

=FILTER(A1:C10, (A1:A10="A")+(C1:C10<20))

This filter returns the rows in the specified **data range** where the value in column A is equal to "A" **or** the value in column C is less than 20. The key takeaway is the implementation of arithmetic operators to represent logical operations in array formulas, a powerful technique exclusive to spreadsheet functions like `FILTER`.

Visualizing the Dataset and Context

To demonstrate the practical application of both the **AND** and **OR** filtering methods, we will use a common dataset structure. Imagine a simple league table tracking team performance, including columns for Team, Player Name, and Points scored. This standard structure allows us to easily isolate specific records based on criteria applied to different columns simultaneously.

The following dataset, spanning the range `A1:C10`, will serve as the reference for all subsequent examples. We will aim to filter this table to highlight specific teams or players based on their performance metrics.

	A	B	C	D	E
1	**Team**	**Position**	**Points**		
2	A	Guard	15		
3	A	Guard	19		
4	A	Forward	22		
5	A	Forward	29		
6	A	Forward	25		
7	B	Guard	21		
8	B	Guard	30		
9	B	Forward	14		
10	B	Forward	12		
11					
12					
13					
14					
15					
16					
17					
18					

This visual representation helps anchor the forthcoming formulaic explanations. Note that the primary criteria for our examples will involve matching the "Team" column (Column A) against the string "A" and checking the "Points" column (Column C) against the numerical value 20. By applying the two different logical operators--**AND** (comma separation) and **OR** (addition)--we can observe how the returned results differ dramatically, illustrating the necessity of choosing the correct Boolean implementation for your data analysis needs.

Example 1: Filtering Data Requiring ALL Conditions (AND)

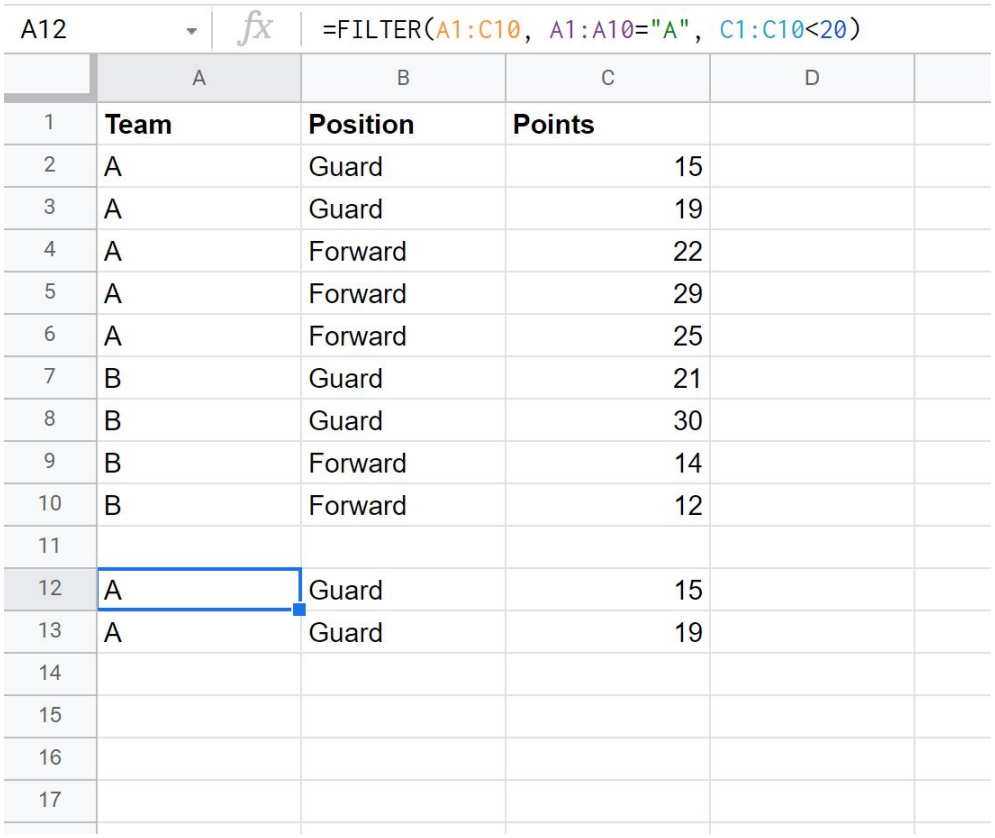
For our first practical exercise, we aim to isolate only those records that satisfy a strict, dual requirement: the team must be "A" **AND** the points scored must be less than 20. This scenario is typical when performing highly targeted analysis, such as identifying low-performing members of a specific group. Since we require both conditions to be met simultaneously, we must utilize the comma separation technique discussed in Method 1.

We deploy the following formula, specifying the complete data output range (`A1:C10`) and then separating the two conditional arrays with a comma:

=FILTER(A1:C10, A1:A10="A", C1:C10<20)

This formula instructs **Google Sheets** to review each row. For Row 2 (Team A, 30 Points), the first condition is `TRUE`, but the second condition (30 < 20) is `FALSE`; therefore, the row is excluded. For Row 5 (Team B, 15 Points), the first condition is `FALSE`, even though the second condition is `TRUE`; therefore, it is also excluded. Only rows where both conditions evaluate to `TRUE` will pass the filter and appear in the resulting output table.

The following screenshot illustrates the precise results generated by applying this **AND** filter. Notice that the output is highly restrictive, ensuring that every returned record adheres to the specified criteria of being Team "A" **and** scoring fewer than 20 points. This strict intersection of data points confirms the successful implementation of the implicit **AND** operator using the comma delimiter within the **FILTER function**.



	A	B	C	D
1	**Team**	**Position**	**Points**	
2	A	Guard	15	
3	A	Guard	19	
4	A	Forward	22	
5	A	Forward	29	
6	A	Forward	25	
7	B	Guard	21	
8	B	Guard	30	
9	B	Forward	14	
10	B	Forward	12	
11				
12	A	Guard	15	
13	A	Guard	19	
14				
15				
16				
17				

The only rows returned are the ones where the team is equal to "A" **and** the points is less than 20.

Example 2: Filtering Data Requiring ANY Condition (OR)

In contrast to the previous example, sometimes the analytical goal is to capture a broader scope of data, including records that satisfy either one condition or another. This flexible requirement utilizes the **OR** logical operator, allowing for a union of datasets based on the criteria. Following the principles of Method 2, we must use the arithmetic addition operator (`+`) to simulate the **OR** logic,

wrapping each conditional statement in parentheses.

We want to filter for all rows where the team is equal to "A" **or** the points is less than 20. This allows us to capture all members of Team A, regardless of their score, while also capturing any player, regardless of team, who scored fewer than 20 points. This comprehensive approach is useful when identifying all relevant data points across different categories.

The formula structure explicitly groups the conditions using parentheses and sums their **Boolean logic** outputs:

=FILTER(A1:C10, (A1:A10="A")+(C1:C10<20))

When this formula is applied, **data range** records that satisfy either condition will be returned. For instance, Row 2 (Team A, 30 Points) satisfies the first condition (Team A) and is included. Row 5 (Team B, 15 Points) fails the first condition but satisfies the second condition (Points < 20) and is therefore also included. This method results in a larger, more inclusive dataset compared to the restrictive **AND** operator.

The following visual confirmation shows the expanded results from the **OR** filter. Observe that rows belonging to teams other than "A" are included, provided their point total is under 20. Similarly, all rows belonging to Team A are included, regardless of their point total. This confirms the successful use of the addition operator to implement the disjunctive **OR** criteria within the `FILTER` function.

A12 | fx | =FILTER(A1:C10, (A1:A10="A")+(C1:C10<20))

	A	B	C	D
1	**Team**	**Position**	**Points**	
2	A	Guard	15	
3	A	Guard	19	
4	A	Forward	22	
5	A	Forward	29	
6	A	Forward	25	
7	B	Guard	21	
8	B	Guard	30	
9	B	Forward	14	
10	B	Forward	12	
11				
12	A	Guard	15	
13	A	Guard	19	
14	A	Forward	22	
15	A	Forward	29	
16	A	Forward	25	
17	B	Forward	14	
18	B	Forward	12	
19				
20				
21				

The only rows returned are the ones where the team is equal to "A" **or** the points is less than 20.

Advanced Filtering Concepts and Resources

Mastering the **FILTER function** with multiple conditions opens up possibilities for highly customized data retrieval in **Google Sheets**. Analysts can combine these methods to create intricate nested conditions--for example, filtering for records that meet (Condition 1 AND Condition 2) OR (Condition 3 AND Condition 4). This is achieved by combining the comma separator (for AND groups) and the addition operator (for OR groups), ensuring meticulous use of parentheses to define the order of logical evaluation.

While the `FILTER` function is powerful, complex filtering involving many conditions or different logical groups sometimes makes the `QUERY` function a viable alternative, as it uses SQL-like syntax that can be easier to read for very long formulas. However, for array-based manipulations and direct comparisons, `FILTER` remains the standard and often more performant choice. Successfully

implementing these two methods for **AND** and **OR** conditions provides the fundamental knowledge required for nearly all dynamic data extraction tasks.

For those seeking to expand their knowledge of data manipulation within the spreadsheet environment, exploring related functions and tutorials is highly recommended. Understanding how to integrate conditional filtering with functions like `SORT`, `UNIQUE`, and `ARRAYFORMULA` can lead to the construction of sophisticated, fully automated dashboards and reports.

Additional Resources

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