# Google Sheets: Use FILTER Function with OR

Authored by

**Mohammed looti**

November 16, 2025

## Introduction: Mastering Complex Conditional Filtering

Google Sheets remains an indispensable tool for modern data analysis, offering robust, cloud-based functionalities for organizing, processing, and visualizing vast amounts of information. The cornerstone of effective data management lies in the ability to dynamically extract specific subsets of data based on precise criteria. This process--transforming raw data tables into actionable intelligence--is primarily achieved through sophisticated filtering mechanisms. Central to this capability is the **FILTER function**, an essential component for creating responsive and dynamic views of your source data.

While executing basic filtering operations--where all conditions must be met (known as **AND** logic)--is straightforward, many real-world analytical challenges require a broader scope. Frequently, data queries demand retrieving records that satisfy **any one of several criteria**. This inclusionary principle is referred to as **OR logic**. Implementing this specific type of logic within the Google Sheets **FILTER** function requires a unique and elegant technique that diverges significantly from standard spreadsheet methodologies.

This comprehensive guide is engineered to clarify the advanced process of combining the **FILTER** function with **OR** logic. We will thoroughly investigate the technical foundation of this arithmetic operation, provide step-by-step examples, and outline crucial best practices for constructing powerful, multi-conditional filters. By the end of this tutorial, you will possess the expertise necessary to efficiently extract precise data subsets, dramatically enhancing your capacity for complex data manipulation and high-level analysis.

## The Foundation: Understanding the FILTER Function

Before we delve into complex logical operations, it is critical to establish a firm grasp of the **FILTER function** itself. This function is specifically designed to return a range of data derived from a source table, filtered according to one or more criteria. Crucially, unlike manual filtering methods, the results generated by **FILTER** are entirely dynamic; they automatically update in real-time as soon as the underlying source data changes. This inherent dynamism makes it an exceptionally robust tool, perfect for creating automated reporting dashboards and live data analysis feeds.

The fundamental syntax for the **FILTER** function is structured precisely as follows:
`=FILTER(range, condition1, )`.
The `range` argument designates the entire source dataset that you wish to process, while all subsequent arguments, starting with `condition1`, define the filtering criteria. By default, when multiple conditions are supplied as distinct, comma-separated arguments (e.g., `condition1, condition2`), the function operates under strict **AND** logic. This means that every single condition must evaluate to `TRUE` for a specific row to be included in the final output.

During execution, each condition evaluates the specified range against a criterion, producing a vertical array of **Boolean** values (`TRUE` or `FALSE`) corresponding directly to each row in the source data. The **FILTER** function then intelligently selects only those rows where the resulting array aligns with the specified logic. While this design inherently handles **AND** scenarios seamlessly, implementing **OR** logic--where inclusion requires only one condition to be satisfied--necessitates a specific, array-based technique that is unique to Google Sheets array formulas.

## Implementing OR Logic Through Arithmetic Operations

Unlike advanced spreadsheet software platforms that might offer a dedicated `OR()` function usable directly within filtering, Google Sheets employs an efficient arithmetic workaround to simulate **OR logic** when used in conjunction with the **FILTER function**. This powerful technique relies entirely on the strategic application of the **plus sign (+)** operator to mathematically combine multiple conditional arrays into a single, unified criterion.

The core principle of this method is rooted in how Google Sheets processes Boolean operations within an array context. When a conditional statement, such as `(A1:A10="Value")`, is evaluated, it generates an array composed of `TRUE` or `FALSE` results. Crucially, when these **Boolean** results are utilized within arithmetic operations (like addition), `TRUE` is automatically coerced into the numerical value `1`, and `FALSE` is converted to `0`.

Therefore, by combining two or more conditional arrays using the **+** operator, you effectively calculate the sum of their binary results for each corresponding row. If even one condition evaluates to `TRUE` (i.e., yields a `1`), the resulting sum for that row will be 1 or greater. Since the **FILTER** function interprets any non-zero numerical result as the equivalent of `TRUE`, this mathematical summation successfully simulates the intended **OR logic**. Conversely, if the final sum is `0` (meaning all underlying conditions evaluated to `FALSE`), the row is systematically excluded from the results.

The standard syntax for applying a simple two-condition **OR** operation within the function is demonstrated below:

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

It is vital to note that the entire combined condition is treated as the single `condition1` argument within the **FILTER** function. The individual conditions must be wrapped securely in parentheses to ensure precise evaluation before the summation occurs. This strict adherence to parentheses guarantees that the **+** operator performs its role correctly as the array-based **OR** operator.

## Practical Application: Filtering Player Statistics

To solidify this essential concept, let us work through a concrete, practical example utilizing a sports dataset of basketball players. This example data, which we assume begins in cell A1, contains columns detailing the Team, Player Position, and Points scored. Our primary goal is to dynamically filter this data to identify players based on flexible, non-exclusive criteria using **OR** logic.

Visualize your data structured as follows, providing detailed attributes for each athlete:

| | 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 | | | | | |

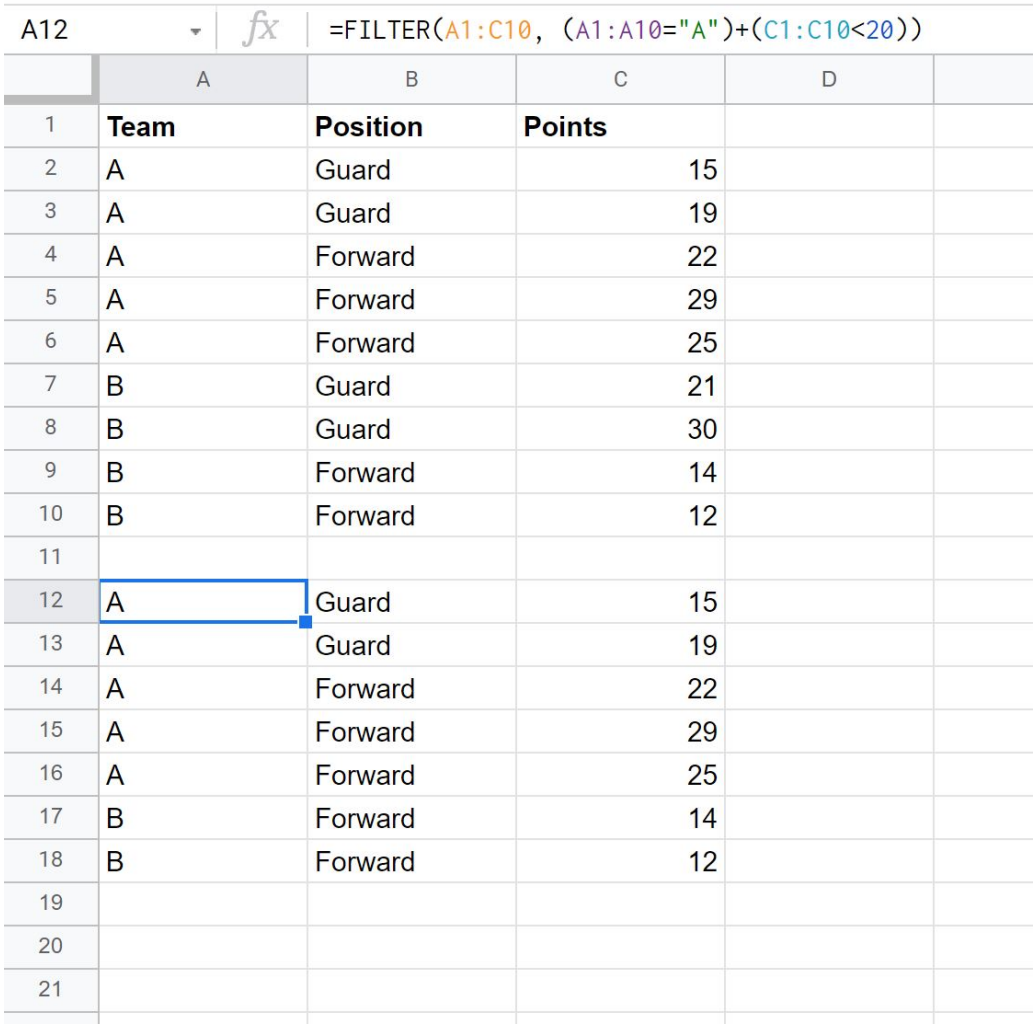
The specific task is to extract all player records where the **Team** designation is "A" **OR** where the total **Points** scored are less than 20. This operation mandates the combination of criteria related to column A and column C using the arithmetic **+** operator. The resulting formula, which should be placed in an empty cell like E1, must be written exactly as follows:

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

In this formula, the `A1:C10` segment defines the complete range of data to be displayed in the output. The first condition, `(A1:A10="A")`, checks the Team column for a match. The second condition, `(C1:C10<20)`, verifies the Points column. The arithmetic **+** operator then ensures that if

a row satisfies the team condition (returning `1`) or the points condition (returning `1`), the resulting sum for that row is greater than zero. A sum greater than zero meets the criteria for inclusion by the **FILTER** function.

Upon execution, the formula generates a dynamically filtered table that strictly adheres to the defined **OR** criteria. The image below clearly showcases the resulting output:




| | 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 | | | | |

As is clearly visible in the filtered results, players are included if they belong to **Team "A"** (such as Player A1, who scored 25 points) or if their **Points** total is below 20 (such as Player B1, who is not on Team A). This result vividly demonstrates the power of **OR logic** in broadening the selection criteria to be maximally inclusive of all relevant data points, achieving a powerful form of data manipulation.

## Scaling Up: Combining Multiple OR Conditions

One of the most significant benefits of leveraging the **plus sign (+)** for implementing **OR logic** is

its inherent infinite scalability. You are not confined to combining only two conditions; this methodology can be effortlessly extended to incorporate three, four, or even more criteria. This is achieved simply by appending additional conditional arrays using the **+** operator. This level of flexibility is absolutely essential for managing complex analytical queries that require matching data against a diverse and extensive set of simultaneous requirements.

This capacity to handle a multitude of conditions ensures that your data processing tasks are highly versatile and responsive to nuanced business logic. For example, in a sales context, you might need to locate all records that originated from the 'East' region, OR records where the total transaction amount exceeds $500, OR those that were processed specifically by Manager Smith. The cumulative arithmetic sum guarantees that any row satisfying even a single one of these criteria will be included in the final, filtered output.

Let us now expand our basketball example by integrating a third condition. Our new objective is to filter for players where the **Team** is "A", **OR** the **Position** is "Guard", **OR** the **Points** total is greater than 15. The necessary formula requires combining three distinct conditional arrays via the summation operator:

**=FILTER(A1:C10, (A1:A10="A")+(B1:B10="Guard")+(C1:C10>15))**

In this expanded syntax, we are evaluating three separate conditions simultaneously:

`(A1:A10="A")` checks the Team column.
`(B1:B10="Guard")` checks the Position column.
`(C1:C10>15)` checks if the player scored more than 15 points.

Each component independently generates an array composed of `1`s and `0`s. The final resulting array, which is formed by summing these three arrays together, will contain non-zero values for any row that successfully satisfies at least one of the criteria. This non-zero result is interpreted as `TRUE` by the **FILTER function**, ensuring the desired maximum inclusivity and specificity for the query.

The execution of this comprehensive formula yields an even larger dataset, successfully including all players who meet any of the three criteria defined.

A12 =FILTER(A1:C10, (A1:A10="A")+(B1:B10="Guard")+(C1:C10>15))

| | 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 | Team | Position | Points | | |
| 13 | A | Guard | 15 | | |
| 14 | A | Guard | 19 | | |
| 15 | A | Forward | 22 | | |
| 16 | A | Forward | 29 | | |
| 17 | A | Forward | 25 | | |
| 18 | B | Guard | 21 | | |
| 19 | B | Guard | 30 | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |

This output decisively validates the efficiency and power of combining multiple **OR logic** conditions, thereby enabling highly sophisticated and targeted data selection capabilities within Google Sheets.

## Best Practices for Robust FILTER Formulas

While mastering the technique of using the **+** operator for **OR logic** provides immense analytical capability, adhering to specific best practices is essential. These guidelines ensure that your formulas remain efficient, easily readable, and minimally prone to errors, particularly when working with complex array formulas in Google Sheets.

**Enforce Clarity with Parentheses:** It is absolutely crucial to always enclose every individual condition within its own distinct set of parentheses (e.g., `(Range=Criterion)`). This is not optional; it is fundamental for correct formula functionality. Parentheses strictly dictate the order of operations, guaranteeing that each condition is properly evaluated into its `TRUE`/`FALSE` array format

before the arithmetic summation (the **+** operator) is applied. This practice is vital for both reliable formula execution and streamlined future debugging efforts.

**Maintain Range Dimensional Consistency:** The ranges utilized in your conditional arrays (e.g., `A1:A10`) must identically match the row count of the main data range specified in the **FILTER function** (e.g., `A1:C10`). If a mismatch occurs in the array dimensions (e.g., trying to compare 10 rows of data against 9 rows of criteria), Google Sheets will inevitably return a calculation error, most commonly `#VALUE!` or `#N/A`, indicating incompatible array sizes.

**Performance Considerations for Large Data:** When handling exceptionally large datasets, the creation of highly complex **FILTER** formulas that involve numerous **OR** conditions can occasionally lead to noticeable delays in processing speed. If you begin to encounter significant slowdowns or latency, you should evaluate whether the built-in `QUERY` function might provide a more optimized solution, as it is often better engineered for executing complex SQL-like queries on massive data tables efficiently.

**Combining AND and OR Logic:** The inherent versatility of the **FILTER** function allows for seamless combination of both logical types. To effectively mix **OR** logic (using the **+** operator) with **AND** logic (using separate, comma-separated arguments or the ***** operator for multiplication), simply wrap your combined **OR** criteria in parentheses and treat the entire expression as a single condition argument. For instance, the structure `=FILTER(range, (condition1 + condition2), condition3)` filters rows that satisfy (condition1 OR condition2) AND condition3. This layered, strategic approach facilitates highly granular and specific data extraction requirements.

## Conclusion: Empowering Your Analytical Workflow

The ability to fluently employ the **FILTER function** alongside robust **OR logic** represents a significant professional advancement in your Google Sheets proficiency. By understanding and strategically utilizing the **plus sign (+)** as the array-based **OR** operator, you unlock the powerful capability to perform flexible, non-restrictive queries across even the most complex structures of your data.

This mastery allows you to move decisively beyond simple, singular filters, empowering you to quickly identify data points that meet any one of a wide variety of criteria simultaneously. Whether your professional focus is on meticulous financial reporting, streamlined inventory management, or detailed customer segmentation, dynamic filtering with **OR** logic ensures that you can swiftly and reliably pinpoint all relevant information required for sound, data-driven decision-making. Its inherent adaptability makes this arithmetic technique an absolutely essential component for any professional engaged in serious data manipulation and analysis.

We highly recommend dedicated practice with diverse data sets and increasingly complex

conditional scenarios. As you build comfort and expertise with the array arithmetic that underpins the **FILTER** function, you will discover that constructing powerful, multi-conditional queries becomes second nature, fundamentally transforming your efficiency and insight within Google Sheets.

## Additional Resources

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