# Learning to Filter Data in Google Sheets: Using SEARCH with Multiple Criteria

Authored by

**Mohammed looti**

January 28, 2026

Mastering data manipulation in Google Sheets often requires advanced techniques for isolating information based on complex criteria. A frequent challenge data analysts face is identifying rows where a specific text field or string contains not just one, but multiple distinct values simultaneously. While this type of filtering might seem complicated at first glance, it can be executed efficiently and elegantly by combining the power of the SEARCH function with the robust extraction capabilities of the FILTER function. This synergy allows users to perform highly targeted searches across large datasets.

The core methodology for applying the **SEARCH** function to find multiple required values within a specific data range in Google Sheets centers around leveraging the implicit "AND" logic of the FILTER function. The foundational syntax for this powerful combination is demonstrated below:

**=FILTER(A2:A10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))**

This specific formula is constructed to return the contents of all cells within the specified range, **A2:A10**, that satisfy two simultaneous conditions: the presence of the text string "Backup" AND the presence of the text string "Guard." The following sections will meticulously break down the mechanics of both the SEARCH and FILTER functions and provide practical, detailed examples to illustrate their combined application.

## Deconstructing the SEARCH Function

The SEARCH function is a vital component of text manipulation within Google Sheets, designed primarily to locate a smaller substring within a larger body of text. A key characteristic that sets **SEARCH** apart from the related FIND function is its inherent case-insensitivity; it treats variations like "backup," "Backup," and "BACKUP" as identical matches. When **SEARCH** successfully locates the specified substring, it returns the numerical starting position of that substring. Conversely, if the substring is not found anywhere in the text, **SEARCH** returns an error value, typically `#VALUE!`. This behavior--returning a number on success and an error on failure--is what makes it uniquely suited for Boolean logic within conditional functions.

The basic syntax required for implementing the **SEARCH** function is straightforward: `SEARCH(search_for, text_to_search, )`. Understanding the role of each argument is essential for effective use:

**search_for:** This mandatory argument defines the precise substring of text that the function is instructed to locate.
**text_to_search:** This mandatory argument specifies the text string or the reference to the cell containing the text where the search should be conducted.
**:** This is an optional argument that allows the user to specify the character position within

**text_to_search** from which the search should commence. If this argument is omitted, the function defaults to beginning the search at the first character (position 1).

When **SEARCH** is embedded as a condition within other formulas, such as the FILTER function, its numerical output (any number greater than zero, indicating a match) is automatically interpreted by the receiving function as a **TRUE** logical value. Similarly, the error returned upon failure to find a match is interpreted as **FALSE**. This seamless conversion from positional data (numbers) to logical data (TRUE/FALSE) provides the foundation for creating sophisticated and reliable conditional logic within complex spreadsheet formulas.

## Employing the FILTER Function for Conditional Extraction

The FILTER function stands out as one of the most essential and heavily utilized functions within Google Sheets, offering unparalleled ability to dynamically extract and display a specific subset of data. Its primary role is to sift through a designated data range and return only those rows or columns that strictly adhere to one or more defined criteria. This functionality is crucial for dynamic reporting, ensuring that analytical results automatically update whenever the underlying source data is modified.

The fundamental syntax for the powerful **FILTER** function is structured as follows: `FILTER(range, condition1, )`. Each parameter serves a distinct and important purpose in defining the data extraction process:

**range:** This argument specifies the exact data range that the function will analyze and whose rows or columns will constitute the final output.
**condition1, :** These are the filtering criteria. For a row to be successfully included in the filtered output, it must satisfy all specified conditions. Each condition must resolve to a Boolean expression (TRUE or FALSE) or an array of TRUE/FALSE values that must align perfectly in size (number of rows or columns) with the initial **range** argument.

A critical feature of the **FILTER** function, particularly relevant to multi-value searching, is its handling of multiple conditions. When several conditions are passed to the function as separate arguments, **FILTER** inherently connects them using an implicit AND operator. Consequently, if you supply multiple conditions, a row must meet ALL conditions to be included in the final results. This built-in "AND" logic makes **FILTER** the perfect container for aggregating multiple **SEARCH** conditions to ensure all required textual values are present.

## Implementing AND Logic with Combined SEARCH and FILTER Functions

The true advantage of working within Google Sheets becomes apparent when functions are combined to solve complex data requirements. When the need arises to isolate data entries that

contain multiple, distinct substrings--meaning all strings must be present--the pairing of the SEARCH function and the FILTER function offers a highly efficient and structurally sound solution. Every separate instance of **SEARCH** acts as one independent condition for **FILTER**, and since **FILTER** mandates that all conditions must be TRUE (due to the implicit AND operator), this structure guarantees a multi-value match.

Revisiting the primary formula that was introduced at the beginning of this guide, we can break down its execution step-by-step to fully grasp its operational flow:

**=FILTER(A2:A10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))**

The formula operates on the following principles: First, `FILTER(A2:A10, ...)` dictates that the output data should be sourced from the data range **A2:A10**. Second, the first condition, `SEARCH("Backup", A2:A10)`, is evaluated. For every cell in the range, **SEARCH** checks for "Backup." If found, it returns a number (TRUE); otherwise, an error (FALSE). Third, the second condition, `SEARCH("Guard", A2:A10)`, executes exactly the same process, checking for the presence of "Guard." Since both **SEARCH** conditions are presented as distinct arguments to **FILTER**, a cell from **A2:A10** will only be included in the final output if the result of the first **SEARCH** is TRUE AND the result of the second **SEARCH** is TRUE. This elegantly implements the desired AND logic for finding text that satisfies multiple criteria.

## Practical Example: Filtering a Dataset for Multiple Roles

To demonstrate the practical effectiveness of this technique, we will use a hypothetical dataset containing basketball player names and their specific positions. Imagine we need to quickly extract all players who are designated with a position that includes both the term "Backup" and the term "Guard." This scenario is highly typical in analyzing complex inventory, role descriptions, or product specifications where multiple attributes reside in a single text field.

We begin with the source data, which is organized within Google Sheets (Note: the positions are located in column A):

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Position** | **Points** | | |
| 2 | Starting Shooting Guard | 22 | | |
| 3 | Starting Center | 12 | | |
| 4 | Starting Point Guard | 15 | | |
| 5 | Backup Shooting Guard | 5 | | |
| 6 | Backup Center | 4 | | |
| 7 | Backup Point Guard | 9 | | |
| 8 | Backup Power Forward | 6 | | |
| 9 | Starting Power Forward | 28 | | |
| 10 | Backup Shooting Guard | 7 | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |
| 19 | | | | |

To meet our objective, we input the precise multi-value formula into an empty cell (e.g., D1). This formula explicitly instructs the application to look only at the position column (A2:A10) for both required strings:

**=FILTER(A2:A10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))**

Upon execution, Google Sheets evaluates the conditions for each row in the specified range. The output accurately displays only those position descriptions that contain both "Backup" and "Guard," demonstrating the formula's ability to precisely target data based on complex textual criteria:

D1 =FILTER(A2:A10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Position** | **Points** | | Backup Shooting Guard | |
| 2 | Starting Shooting Guard | 22 | | Backup Point Guard | |
| 3 | Starting Center | 12 | | Backup Shooting Guard | |
| 4 | Starting Point Guard | 15 | | | |
| 5 | Backup Shooting Guard | 5 | | | |
| 6 | Backup Center | 4 | | | |
| 7 | Backup Point Guard | 9 | | | |
| 8 | Backup Power Forward | 6 | | | |
| 9 | Starting Power Forward | 28 | | | |
| 10 | Backup Shooting Guard | 7 | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | | | | | |

## Expanding Results: Including Associated Data Columns

In real-world data analysis, filtering rarely ends with just the matching column. It is almost always necessary to retrieve accompanying data points associated with the filtered entries. In our basketball example, after identifying the players with the "Backup Guard" role, we likely want to see their corresponding points scored. Expanding the output to include additional columns is straightforward, requiring only a slight modification to the initial argument of the FILTER function.

To include the player names (Column B) along with their positions (Column A) in the filtered result, we simply adjust the first argument--the output data range--from **A2:A10** to **A2:B10**. Critically, the subsequent SEARCH function conditions must still reference the column containing the text being filtered (in this case, **A2:A10**), as the criteria are based solely on the position description.

The updated formula is structured as follows:

**=FILTER(A2:B10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))**

The initial range **A2:B10** now specifies the breadth of the output columns, while the two **SEARCH** conditions ensure that the filtering logic remains focused on the appropriate data column. The resulting output demonstrates the comprehensive nature of the FILTER function, providing both the position and the associated player name for all matching rows:

D1 fx =FILTER(A2:B10, SEARCH("Backup", A2:A10), SEARCH("Guard", A2:A10))

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Position** | **Points** | | Backup Shooting Guard | 5 |
| 2 | Starting Shooting Guard | 22 | | Backup Point Guard | 9 |
| 3 | Starting Center | 12 | | Backup Shooting Guard | 7 |
| 4 | Starting Point Guard | 15 | | | |
| 5 | Backup Shooting Guard | 5 | | | |
| 6 | Backup Center | 4 | | | |
| 7 | Backup Point Guard | 9 | | | |
| 8 | Backup Power Forward | 6 | | | |
| 9 | Starting Power Forward | 28 | | | |
| 10 | Backup Shooting Guard | 7 | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | | | | | |

## Advanced Logic: Handling OR Searches and Case Sensitivity

While the combined SEARCH function and FILTER function method perfectly handles "AND" logic, complex data analysis often requires implementing "OR" logic or stricter search parameters. Google Sheets offers powerful alternatives to manage these nuanced requirements, significantly enhancing data manipulation capabilities.

**Implementing "OR" Logic for Multiple Values:** When the requirement is to find cells containing "Backup" OR "Guard" (or both), simply using separate **SEARCH** arguments in **FILTER** will not work, as that enforces the implicit AND operator. To achieve "OR" logic, we must combine the conditions in a way that allows either match to return TRUE. This is typically done by summing the results of the **SEARCH** functions. Since **SEARCH** returns a number (TRUE) on success and an error (FALSE) on failure, we can use the addition operator (+) to force Boolean arithmetic. In

Google Sheets, a non-zero sum is interpreted as TRUE. The formula structure becomes: `=FILTER(A2:A10, (SEARCH("Backup", A2:A10) + SEARCH("Guard", A2:A10)))`. If either **SEARCH** returns a number, the sum will be greater than zero, satisfying the FILTER condition.

**Enforcing Case-Sensitive Searches:** As previously noted, the SEARCH function is intentionally case-insensitive. If your analysis demands strict differentiation between, for example, "ID" and "id," you must substitute **SEARCH** with the FIND function. **FIND** behaves identically to **SEARCH** in terms of returning a position number or an error, but it is strictly case-sensitive. The syntax remains consistent: `FIND(search_for, text_to_search, )`.

**Leveraging Regular Expressions:** For searches involving highly complex patterns, variable structures, or intricate textual boundaries, regular expressions provide the most advanced pattern-matching capabilities. Google Sheets supports this through the REGEXMATCH function. While requiring a steeper learning curve, **REGEXMATCH** can define intricate search criteria that are impossible to achieve with simple string functions, thereby offering the ultimate level of flexibility in text analysis.

## Conclusion: Mastering Multi-Value Data Filtering

The ability to efficiently filter data based on multiple textual criteria is an indispensable skill for effective data management in Google Sheets. By strategically combining the SEARCH function and the FILTER function, users can establish robust formulas that enforce an implicit "AND" operation, ensuring that all specified substrings must be present in the target cell for inclusion.

We have demonstrated not only the core mechanics of this AND logic but also the simple process of modifying the initial data range argument to expand the filtered output, providing comprehensive results that include associated columns. Furthermore, understanding the advanced considerations--such as using addition for "OR" logic, employing FIND for strict case sensitivity, and utilizing REGEXMATCH for complex pattern matching--empowers you to tackle virtually any text-based filtering challenge. Mastering these functions transforms raw datasets into actionable, precisely filtered insights.

## Additional Resources

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