Google Sheets: Create a List Based on Criteria

Authored by

stats writer

November 17, 2025

Google Sheets stands as an indispensable tool in the modern digital workspace, offering powerful capabilities for data manipulation, analysis, and reporting. Its versatility extends far beyond basic spreadsheet functions, enabling users to construct intricate data models and automate complex processes. Among its most advanced features is the ability to dynamically generate a distinct list based on specified criteria. This functionality is critical for efficiency, allowing users to precisely sort and filter large datasets, extracting only the relevant records instantly. When dealing with thousands of rows of information or when mandated to produce rapid, specialized reports, mastering criterion-based list generation becomes paramount. This technique not only streamlines data extraction but ensures that the resulting reports are clean, structured, and derived directly from the source data, significantly reducing manual effort and bolstering data integrity.

While basic filtering functions in Google Sheets can hide irrelevant data, they do not automatically pull qualifying records into a separate, clean output range. This limitation often necessitates the use of more complex array formulas that combine several powerful functions. The methodology detailed in this guide utilizes a robust combination of INDEX, SMALL, and ROW, wrapped within an IFERROR wrapper, to create an array formula that spills the results dynamically. Understanding this formula is key to automating advanced data queries within your spreadsheets.

The Core Formula for Dynamic List Generation

To effectively create a clean, criterion-based list that automatically updates as your source data changes, we employ a sophisticated array formula. This formula leverages the power of conditional logic to identify the row numbers corresponding to the records that meet the specified requirements, and then uses those row numbers to extract the actual values. This approach is significantly more robust than simple filtering methods because the resulting list exists independently of the source data range.

The following is the fundamental structure used to achieve this dynamic extraction in Google Sheets. Note that this type of formula typically requires input as an array formula, often achieved by pressing **Ctrl+Shift+Enter** (or **Cmd+Shift+Enter**) in older systems, although modern versions of Sheets often handle this implicitly when using array-aware functions like IF combined with INDEX.

You can use the following basic formula to create a list based on criteria in Google Sheets:

=IFERROR(INDEX($A$2:$A$12,SMALL(IF($B$2:$B$12=$B$2,ROW($B$2:$B$12)),ROW(1:1))-1,1),"")

This particular formula is designed to perform a conditional extraction. Specifically, it creates a list of values found within the source column range **A2:A12**. The condition dictating which values are

extracted is based on whether the corresponding value in the criterion range **B2:B12** is precisely equal to the reference value located in cell **B2**. The use of absolute references (the dollar signs, e.g., **$A$2**) is essential to ensure that when the formula is dragged down to subsequent rows, the reference ranges remain fixed and unchanging.

Deconstructing the Formula Components for Clarity

To master this technique, it is vital to understand the role of each function within this complex nested structure. The formula works from the inside out, first identifying the qualifying positions, and then retrieving the data sequentially. This methodology is often termed array manipulation and is a key skill for advanced spreadsheet users.

Conditional Row Identification (`IF($B$2:$B$12=$B$2, ROW($B$2:$B$12))`): This is the core logical engine. It evaluates every cell in the criterion range (**$B$2:$B$12**) against the target criteria (**$B$2**). If the condition is **TRUE**, it returns the absolute row number of that cell using the ROW function. If **FALSE**, it returns the boolean value **FALSE**, which is conveniently ignored by the subsequent SMALL function. This crucial step results in an array containing only the row numbers of matching records and a collection of FALSE values.

Sequencing the Results (`SMALL(...,ROW(1:1))`): The SMALL function is used to extract the K-th smallest value from the array generated above. By dynamically setting the K argument using `ROW(1:1)`, we instruct the function to pull the 1st smallest valid row number in the current cell. When the formula is copied down one row, `ROW(1:1)` automatically adjusts to `ROW(2:2)` (returning 2), thereby retrieving the 2nd smallest row number, and so on. This mechanism ensures that the results are pulled in their original order without generating unwanted gaps.

Offset Correction (`...-1`): Since the ROW function returns the absolute row number (e.g., 2, 3, 4...), and the INDEX function requires an offset relative to the start of the defined data range (which begins at row 2, hence an offset of 1), we must subtract 1 from the resulting row number. This calculates the correct positional index within the scope of the **$A$2:$A$12** range.

Retrieving the Value (`INDEX($A$2:$A$12, ...)`): The INDEX function then uses the corrected relative row index to locate and return the corresponding value from the data column (**$A$2:$A$12**), effectively completing the data retrieval for that specific row in the output list.

Clean Error Handling (`IFERROR(...,"")`): Finally, the entire formula is wrapped in the IFERROR function. Once all valid matching records have been extracted, the SMALL function will inevitably attempt to find a non-existent K-th smallest value, resulting in a standard error (usually #NUM!). The IFERROR wrapper catches this error and returns an empty string (`""`), ensuring the list generation stops cleanly without displaying distracting error messages.

Setting Up the Dataset for Practical Application

To illustrate the application of this powerful formula, we will use a sample dataset tracking player names, their teams, and their positions. This dataset provides a clear structure for demonstrating both single and multiple criterion filtering scenarios effectively. The data is structured with headers in row 1, and records starting in row 2.

The following examples show how to use this formula in practice with the following dataset in Google Sheets:

	A	B	C	D
1	**Player**	**Team**	**Position**	
2	Andy	Mavs	Guard	
3	Bob	Mavs	Forward	
4	Charles	Nets	Guard	
5	Doug	Spurs	Center	
6	Eric	Heat	Forward	
7	Frank	Mavs	Guard	
8	George	Warriors	Center	
9	Harry	Spurs	Forward	
10	Isaiah	Heat	Guard	
11	Jake	Nets	Guard	
12	Ken	Spurs	Forward	
13				
14				
15				
16				

In this sample data range, Column A contains the **Player Name** (the specific values we wish to extract), Column B lists the **Team** (our primary criterion field), and Column C details the **Position** (our secondary criterion field). Our objective throughout the following examples is to extract the player names into a separate column (Column E) based on specific conditional logic applied to Columns B and C.

Example 1: Filtering Data Based on a Single Criterion

In our first scenario, the requirement is straightforward: we need to extract a list of all players who are exclusively members of the **Mavs** team. This task necessitates setting the criterion range (Column B, cells B2 through B12) to match a specific static value, which is assumed here to be the

value found in cell **B2** for simple demonstration, although defining the criterion in a separate cell (e.g., F1) is recommended for production environments.

The goal is to populate Column E dynamically with the names of players whose corresponding Team entry in Column B is "Mavs". We utilize the standard array formula structure defined above, focusing the conditional check solely on the Team column.

We can use the following formula to create a list of players who are on the **Mavs** team:

=IFERROR(INDEX($A$2:$A$12,SMALL(IF($B$2:$B$12=$B$2,ROW($B$2:$B$12)),ROW(1:1))-1,1),"")

To implement this, you should enter the formula into the initial cell of your output range, which we designate as cell **E2**. After formula entry, it is essential to then drag the formula handle downwards to cover the entire potential range of results in column E (e.g., down to E12). Since the formula utilizes the relative sequencing reference `ROW(1:1)`, each subsequent cell will correctly adjust its K-value, requesting the next qualifying player name in the correct order.

We can type this formula into cell **E2** and then drag it down to the remaining cells in column E to create a list of players who are on the Mavs team:

E2 fx =IFERROR(INDEX($A$2:$A$12,SMALL(IF($B$2:$B$12=$B$2,ROW($B$2:$B$12

	A	B	C	D	E
1	**Player**	**Team**	**Position**		**Players on Mavs Team**
2	Andy	Mavs	Guard		Andy
3	Bob	Mavs	Forward		Bob
4	Charles	Nets	Guard		Frank
5	Doug	Spurs	Center		
6	Eric	Heat	Forward		
7	Frank	Mavs	Guard		
8	George	Warriors	Center		
9	Harry	Spurs	Forward		
10	Isaiah	Heat	Guard		
11	Jake	Nets	Guard		
12	Ken	Spurs	Forward		
13					
14					
15					

Upon execution, the formula successfully identifies and extracts the names corresponding to the "Mavs" team, automatically leaving blank cells once all matches have been found, thanks to the robust IFERROR function wrapper.

The result is a list of three players:

Andy
Bob
Frank

A verification against the original dataset confirms the accuracy of this extraction. This method demonstrates the power of combining INDEX and SMALL functions in handling sophisticated single-criterion extraction tasks, providing a dynamic list that eliminates manual copy-pasting and ensures real-time accuracy.

Example 2: Implementing Logic for Multiple Criteria

Often, data filtering requires matching several conditions simultaneously, increasing the precision of the output. For example, we might need a list of players who meet two strict criteria: belonging to the **Mavs** team AND holding the **Guard** position. Adapting the core array formula for multiple conditions requires incorporating logical multiplication (the asterisk symbol, `*`) within the IF statement.

In array formulas, multiplying two conditional checks acts as an **AND** operator. If the first condition returns **TRUE** (which Sheets treats arithmetically as 1) and the second condition returns **TRUE** (also 1), the product is 1 (TRUE), allowing the ROW number to be returned. Conversely, if either or both conditions are FALSE (treated as 0), the product is 0 (FALSE), which is ignored by the SMALL function, thereby excluding that row from the final list.

We can use the following formula to create a list of players who are on the **Mavs** team and have a position of **Guard**:

=IFERROR(INDEX($A$2:$A$12,SMALL(IF(($B$2:$B$12=$B$2)*($C$2:$C$12=$C$2),ROW($B$2:$B$12)),ROW(1:1))-1,1),"")

Notice the critical logical addition: `($B$2:$B$12=$B$2)*($C$2:$C$12=$C$2)`. We are checking if the Team range (B2:B12) equals the criterion in B2 AND if the Position range (C2:C12) equals the criterion in C2. For this demonstration, we assume B2 contains "Mavs" and C2 contains "Guard" as our reference criteria.

We can type this formula into cell **E2** and then drag it down to the remaining cells in column E to

create a list of players who are on the Mavs team and have a position of Guard:

E2 =IFERROR(INDEX($A$2:$A$12,SMALL(IF(($B$2:$B$12=$B$2)*($C$2:$C$12=$

	A	B	C	D	E
1	**Player**	**Team**	**Position**		**Guard on Mavs Team**
2	Andy	Mavs	Guard		Andy
3	Bob	Mavs	Forward		Frank
4	Charles	Nets	Guard		
5	Doug	Spurs	Center		
6	Eric	Heat	Forward		
7	Frank	Mavs	Guard		
8	George	Warriors	Center		
9	Harry	Spurs	Forward		
10	Isaiah	Heat	Guard		
11	Jake	Nets	Guard		
12	Ken	Spurs	Forward		
13					
14					
15					
16					

The resulting list is significantly smaller than the previous example, accurately reflecting only those entries that satisfy both required conditions. This powerful logical multiplication technique allows for complex and customized combinations of criteria to be applied to the data extraction process.

The result is a list of two players:

Andy
Frank

We can confidently look at the original dataset to confirm that both of these players are on the **Mavs** team and simultaneously hold the **Guard** position. Should you require an **OR** condition (e.g., Team is Mavs OR Position is Guard), you would simply replace the multiplication operator (*) with the addition operator (+) between the conditional checks.

Advanced Techniques: Alternative Methods for Filtering

While the INDEX-SMALL array formula is highly educational and robust, particularly in environments requiring formula compatibility, Google Sheets offers two specialized functions that simplify dynamic list generation: **FILTER** and **QUERY**.

For most modern Sheets users, the **FILTER** function provides a simpler, less error-prone alternative, especially when dealing with basic single-column output needs. The function automatically handles the sequencing and error suppression, spilling the results dynamically without manual row-by-row input.

The syntax for the FILTER function is straightforward: `=FILTER(range_to_return, condition_1, , ...)`.

For Example 2 (Mavs team AND Guard position) using FILTER: `=FILTER(A2:A12, B2:B12="Mavs", C2:C12="Guard")`.

Alternatively, the **QUERY** function, which uses SQL-like syntax, offers maximum flexibility and performance for very large datasets: `=QUERY(A2:C12, "SELECT A WHERE B = 'Mavs' AND C = 'Guard'", 0)`. While these alternatives are often preferred for simplicity, understanding the INDEX-SMALL approach remains valuable for scenarios demanding specific array manipulation or backward compatibility.

Summary of Key Best Practices

Maintaining clean and accurate dynamic lists requires adhering to several best practices when designing your spreadsheet structure and implementing these advanced formulas:

Use Absolute References: Always utilize absolute references (e.g., **$A$2:$A$12**) for your data range and criterion reference cells within the array formula. This prevents unintended range shifting when the formula is copied down the output column.

Dedicated Criterion Cells: To maximize user accessibility and formula flexibility, establish dedicated input cells outside of the primary dataset (e.g., F1 and G1) for holding the search criteria. This allows users to easily change the extraction criteria without ever modifying the complex array formula itself.

Robust Error Handling: The inclusion of the IFERROR wrapper is essential for clean list generation. Without it, the list would terminate with several unsightly **#NUM!** or **#REF!** errors after the final matching record is displayed, compromising the professional appearance of the report.

Validate Data Types: Ensure that your criteria reference cells contain data matching the type in the source columns (e.g., searching for a number should reference a numeric cell, not text). Mismatched data types are a frequent cause of errors in array formulas.

In conclusion, mastering the ability to generate dynamic lists based on conditional criteria is a cornerstone of advanced data management within Google Sheets. This tutorial meticulously demonstrated the construction and application of the powerful nested formula utilizing IFERROR,

INDEX, SMALL, and ROW functions. We provided comprehensive examples showing how to isolate data based on a single condition and how to enforce strict filtering using multiple, simultaneous conditions via logical multiplication. By employing these techniques, users can significantly enhance their data reporting capabilities, ensuring that extracted data is always accurate, current, and perfectly organized, thereby maximizing productivity and data reliability across all their spreadsheet operations.