

# How to Easily Find the Maximum Value with Criteria Using MAXIFS in Google Sheets

Authored by  
**stats writer**

December 1, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Easily Find the Maximum Value with Criteria Using MAXIFS in Google Sheets*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=103003>

The **MAXIFS** function is an indispensable tool in [Google Sheets](#), designed specifically for advanced conditional data aggregation. Unlike the standard **MAX** function, which simply identifies the highest numerical value in a specified [range](#), **MAXIFS** introduces robust filtering capabilities. It empowers users to pinpoint the maximum value only among cells that satisfy one or multiple defined conditions, or [criteria](#).

This functionality is crucial when dealing with complex datasets where segregation and specific metric analysis are required. For instance, instead of finding the highest sales figure across an entire quarter, you might need to find the highest sales figure specifically attributed to the East region or a particular product line. **MAXIFS** handles these precise requirements efficiently, saving considerable time compared to manual filtering or using array formulas. Mastering this function significantly enhances your capability to perform sophisticated [data analysis](#) and reporting within the spreadsheet environment.

Throughout this comprehensive guide, we will explore the precise structure and arguments required by the **MAXIFS** function, demonstrate its application using practical, real-world examples, and discuss best practices for optimizing its performance. Whether you are analyzing sales data, managing inventory records, or tracking sports statistics, understanding conditional aggregation is fundamental to unlocking the full potential of your spreadsheets.

You can use the **MAXIFS** function in [Google Sheets](#) to find the maximum value in a specified range, contingent upon meeting a predefined set of criteria.

## Understanding the MAXIFS Syntax and Arguments

To effectively utilize the **MAXIFS** function, it is essential to first grasp its precise structure, or [syntax](#). Unlike simpler functions, **MAXIFS** requires an ordered sequence of arguments that define both the values to be evaluated and the conditions under which those values are selected. The function is designed to handle pairs of criteria arguments, allowing for scalability from one condition up to many simultaneous restrictions.

The generalized format of the **MAXIFS** function is as follows:

**=MAXIFS(range, criteria\_range1, criteria1, )**

Understanding the role of each argument is critical for successful implementation. The function requires at least three primary components, followed by optional subsequent pairs of criteria ranges and criteria values. We can break down the required arguments into the following list:

**range** (Required): This is the actual [range](#) of cells from which the maximum value will be returned.

It must contain numerical data. This is often referred to as the **Maximum Range**.

**criteria\_range1** (Required): This is the first range of cells that will be evaluated against the first condition. It must be the same size as the Maximum Range, although it contains the criteria values, not necessarily numerical results.

**criteria1** (Required): This is the first condition or pattern that must be met in `criteria_range1`. This can be a number, a text string enclosed in quotes (e.g., "Team A"), a logical expression (e.g., ">50"), or a cell reference.

**criteria\_range2, criteria2, ...** (Optional): These subsequent pairs allow you to introduce additional filtering conditions. **MAXIFS** requires that all specified criteria are met simultaneously (an **AND** logic) for a value to be considered for the maximum calculation.

### Example 1: Finding the Maximum Value Based on a Single Criterion

A common scenario in data analysis involves segmenting data based on a single qualifying factor. For instance, consider a detailed dataset tracking basketball player performance, specifically recording points scored and team affiliation. We want to identify the single highest score, but only within a specific team. This demonstrates the power of using MAXIFS with just one condition.

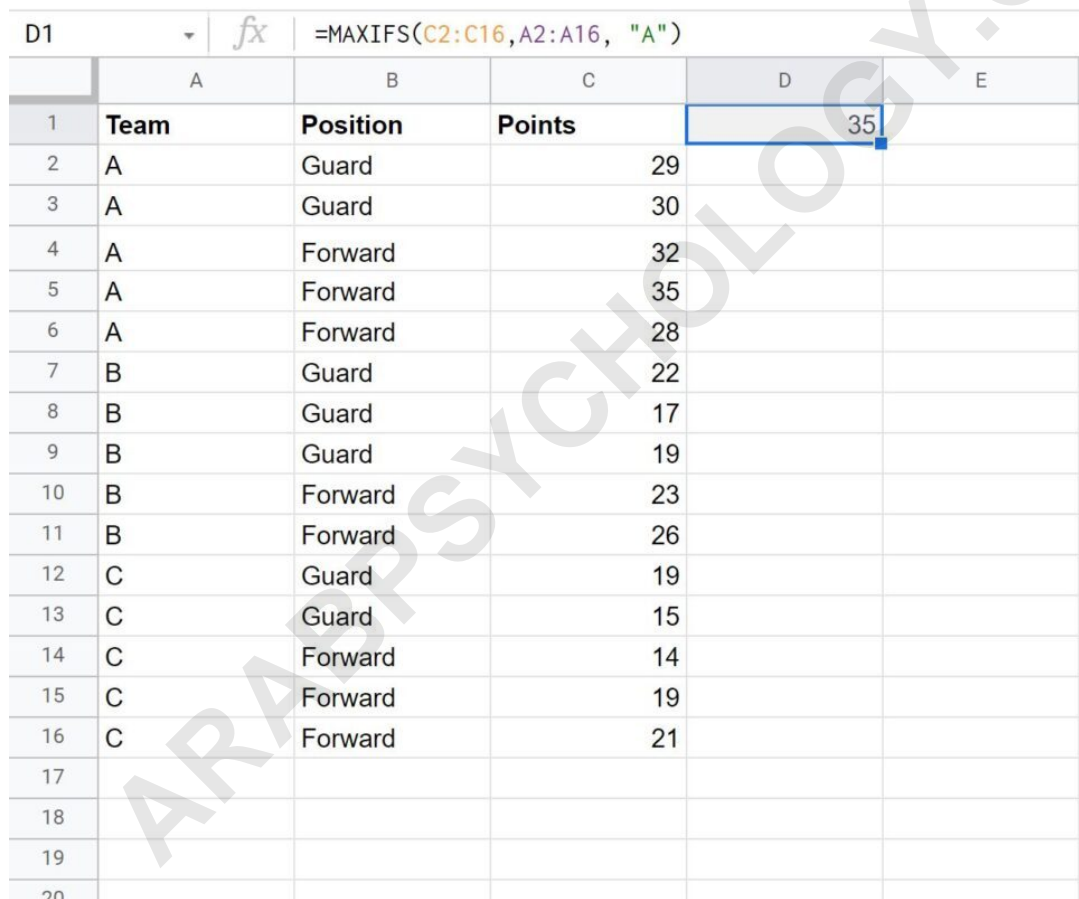
Assume we have the following organized dataset, mapping 15 different basketball players' points against their respective teams:

	A	B	C	D	
1	<b>Team</b>	<b>Position</b>	<b>Points</b>		
2	A	Guard	29		
3	A	Guard	30		
4	A	Forward	32		
5	A	Forward	35		
6	A	Forward	28		
7	B	Guard	22		
8	B	Guard	17		
9	B	Guard	19		
10	B	Forward	23		
11	B	Forward	26		
12	C	Guard	19		
13	C	Guard	15		
14	C	Forward	14		
15	C	Forward	19		
16	C	Forward	21		
17					
18					

Our objective is simple: calculate the maximum points scored **if and only if** the player belongs to Team A. To achieve this, we define the range containing the values we wish to maximize (Points) and then specify the criterion range (Team) and the criterion itself ("A"). The resulting formula is concise and highly effective:

**=MAXIFS(C2:C16, A2:A16, "A")**

In this formula, C2:C16 is the **maximum range** (Points), A2:A16 is the **criteria range** (Team), and "A" is the specific **criterion**. Once applied, Google Sheets evaluates only those rows where Column A equals "A". The screenshot below illustrates the deployment of this formula, clearly showing the resultant maximum value:



	A	B	C	D	E
1	<b>Team</b>	<b>Position</b>	<b>Points</b>		
2	A	Guard	29		
3	A	Guard	30		
4	A	Forward	32		
5	A	Forward	35		
6	A	Forward	28		
7	B	Guard	22		
8	B	Guard	17		
9	B	Guard	19		
10	B	Forward	23		
11	B	Forward	26		
12	C	Guard	19		
13	C	Guard	15		
14	C	Forward	14		
15	C	Forward	19		
16	C	Forward	21		
17					
18					
19					
20					

Upon execution, the function confirms that the highest score achieved by any player on Team A is **35**. This foundational example serves as a gateway to utilizing more complex conditional logic within your datasets.

## Example 2: Applying MAXIFS with Multiple Logical Criteria (AND Logic)

The true power of the MAXIFS function emerges when you need to filter data based on several independent conditions simultaneously. Unlike some conditional functions that rely on array processing to achieve multiple filters, **MAXIFS** handles multi-criteria evaluation natively and efficiently. This enables highly specific data segmentation, ensuring that only records meeting every defined condition are considered for the maximum calculation.

Using the same basketball player dataset shown previously, we now introduce a second layer of complexity. We want to determine the highest score achieved by a player who is specifically a **Guard** on Team A. This requires satisfying two criteria at once: the player must belong to Team A (Criterion 1) **AND** the player must hold the Guard position (Criterion 2).

	A	B	C	D
1	<b>Team</b>	<b>Position</b>	<b>Points</b>	
2	A	Guard	29	
3	A	Guard	30	
4	A	Forward	32	
5	A	Forward	35	
6	A	Forward	28	
7	B	Guard	22	
8	B	Guard	17	
9	B	Guard	19	
10	B	Forward	23	
11	B	Forward	26	
12	C	Guard	19	
13	C	Guard	15	
14	C	Forward	14	
15	C	Forward	19	
16	C	Forward	21	
17				
18				

To implement this dual condition, we simply append the second criteria pair to the formula established in Example 1. The formula structure clearly demonstrates how the pairs of `criteria_range` and `criteria` arguments are chained together:

**=MAXIFS(C2:C16, A2:A16, "A", B2:B16, "Guard")**

In this structure, `C2:C16` remains the maximum range. The first criteria pair filters for Team A (`A2:A16` and `"A"`). The second criteria pair, `B2:B16` and `"Guard"`, filters the remaining records for players in the Guard position. The resulting output, as shown in the practical application below,

identifies the specific maximum value that satisfies this stringent condition:

	A	B	C	D	E
1	<b>Team</b>	<b>Position</b>	<b>Points</b>	30	
2	A	Guard	29		
3	A	Guard	30		
4	A	Forward	32		
5	A	Forward	35		
6	A	Forward	28		
7	B	Guard	22		
8	B	Guard	17		
9	B	Guard	19		
10	B	Forward	23		
11	B	Forward	26		
12	C	Guard	19		
13	C	Guard	15		
14	C	Forward	14		
15	C	Forward	19		
16	C	Forward	21		
17					
18					
19					

The calculation confirms that the highest score among players on Team A who are Guards is **30**. It is important to remember that **MAXIFS** supports an extensive number of criteria pairs; you can use the same syntax to introduce as many criterion as your analysis requires.

### Advanced Criteria Types: Using Operators and Wildcards

While the previous examples demonstrated filtering based on exact text matches, the **MAXIFS** function is far more flexible. It fully supports the use of comparison operators and wildcards within the criteria arguments, enabling dynamic and sophisticated conditional logic. This is essential when searching for maximum values that fall above, below, or within a specified threshold, rather than matching a static value.

Comparison operators are typically used with numerical data but can also be applied to dates. These operators must be enclosed in quotation marks and often paired with a number or a cell reference. Examples include: ">100" (greater than 100), "<=50" (less than or equal to 50), or "<>A1" (not equal to the value in cell A1). Using operators allows us to ask complex questions like: "What is the maximum salary earned by an employee who processed more than 50 orders in a

quarter?" This opens up conditional calculations to a far broader range of analytical queries, moving beyond simple categorization.

Furthermore, [Google Sheets](#) supports the use of wildcards for filtering text strings, making partial matching straightforward. The asterisk (\*) matches any sequence of zero or more characters, while the question mark (?) matches any single character. For example, using the [criterion](#) "A\*" would match any team name starting with the letter 'A', such as 'Aces', 'Angels', or 'A'. Conversely, "?????" would match any four-character entry. This flexibility is invaluable when data entry is inconsistent or when searching across broad categories without explicit exact match requirements.

## Common Errors and Troubleshooting MAXIFS

Although the **MAXIFS** function is powerful, users frequently encounter specific errors that stem primarily from mismatched [range](#) sizes or incorrect criteria formatting. Understanding these common pitfalls ensures smoother implementation and quicker troubleshooting when formulas do not yield the expected results, helping to maintain data integrity and speed of [data analysis](#).

The most frequent error occurs when the size of the **maximum range** (the range containing values to maximize) does not align exactly with the size of the **criteria ranges**. If C2:C16 is the maximum range (15 rows), then all subsequent criteria ranges (e.g., A2:A16, B2:B16) must also span 15 rows. If there is a mismatch, the function will often return an error value, such as #VALUE!, indicating a structural inconsistency in the formula's arguments. Always double-check that your ranges have identical dimensions and starting/ending row numbers, as this is a strict requirement for all conditional aggregate functions in Google Sheets.

Another area prone to error involves the handling of text and numerical criteria. When specifying text criteria, it must always be enclosed in double quotation marks (e.g., "Guard"). However, if you are referencing a cell that already contains the text (e.g., cell E1), you should use the cell reference without quotes (e.g., E1). Similarly, when combining operators with numbers or references, ensure correct concatenation is used if the operator is dynamic. For instance, to reference a cell F1 containing a dynamic limit, the criterion should be formatted as ">"&F1. Neglecting these formatting rules often leads to incorrect results or errors, especially when mixing static strings with dynamic cell references within the [syntax](#).

## MAXIFS vs. Array Formulas: Performance Considerations

Before the introduction of specialized conditional functions like [MAXIFS](#) and [MINIFS](#), users relied on complex array formulas, often involving the [MAX function](#) nested within an [ARRAYFORMULA](#) construct or utilizing the [FILTER](#) function. While these methods are still viable, **MAXIFS** generally offers significant performance and clarity advantages, particularly in large [Google Sheets](#)

environments.

The primary benefit of using **MAXIFS** is its optimized internal calculation structure. Array formulas, especially those involving complex logical tests on large datasets, can be resource-intensive, leading to noticeable spreadsheet lag and slower recalculation times as the sheet grows. **MAXIFS** is designed specifically for this conditional maximum task and is typically processed more efficiently by the spreadsheet engine because it avoids the overhead of creating and processing a virtual array of filtered results. This optimization makes it the recommended choice for mission-critical data analysis involving thousands of rows of data.

Furthermore, **MAXIFS** formulas are inherently more readable and easier to debug than nested array formulas. Its structure is explicit: Maximum Range first, followed by easily identifiable pairs of Criteria Range and Criteria. This transparent structure reduces the likelihood of errors during formula creation and maintenance, making it more accessible for users who are not experts in advanced spreadsheet logic. While array formulas offer greater flexibility in certain niche scenarios (like OR logic which requires specific structuring), **MAXIFS** should be the default choice for conditional maximum calculation due to its native efficiency and improved clarity.