Use AVERAGEIFS in Google Sheets (3 Examples)

Authored by

Mohammed looti

October 31, 2025

Introduction to the Power of AVERAGEIFS in Google Sheets

The **AVERAGEIFS** function in **Google Sheets** represents a significant leap forward from basic statistical methods, providing a robust framework for complex conditional **data analysis**. While the simpler `AVERAGEIF` function is limited to calculating an average based on a single condition, `AVERAGEIFS` expands this capability exponentially, allowing users to define and test multiple **criteria** simultaneously across various columns or **range**s. This advanced functionality is crucial for extracting highly specific and accurate insights from large or intricate spreadsheets.

The necessity for precise, multi-conditional averaging arises frequently in professional environments. Consider a scenario where you must determine the average performance of sales representatives who exceeded a certain quota AND operate within a specific geographical territory. Or perhaps you need the **average value** of inventory items that are marked as "In Stock" AND were procured before a specific date. Performing these calculations manually, involving complex filtering and subsetting, is not only time-intensive but also significantly increases the probability of human error, compromising the integrity of your statistical results.

By leveraging **AVERAGEIFS**, this tedious manual process is fully automated. The function applies an "AND" logic to all specified conditions, ensuring that only data points meeting every single requirement are included in the final calculation. This streamlining capability enhances efficiency, guarantees accuracy in statistical computations, and empowers data analysts to move beyond generalized aggregates, pinpointing specific trends and performance indicators within their complex data structures with unparalleled precision.

Mastering the AVERAGEIFS Syntax and Structure

To unlock the full potential of this powerful tool, a comprehensive understanding of its **syntax** is essential. The structure of the **formula** is deliberately designed to clearly segment the data to be averaged from the conditions that must be met. This logical organization allows for highly flexible conditional requirements, accommodating almost any analytical need within your spreadsheet environment.

The basic structure of the **AVERAGEIFS** function is defined as follows, with optional components enclosed in square brackets:

AVERAGEIFS(average_range, criteria_range1, criterion1,)

Understanding the distinct role of each argument is key to formulating accurate calculations. The arrangement is critical: the function first requires the numerical data to be averaged, followed by pairs of ranges and corresponding conditions that dictate which rows are eligible for inclusion. The core logic relies on these pairs being evaluated sequentially, yet simultaneously, ensuring that only

rows satisfying all parameters are considered.

average_range: This is the mandatory initial **range** containing the numeric values that the function will average. If you are calculating the average of "Sales Revenue," this range must consist solely of the revenue figures. It is vital that this range contains numerical data, as non-numeric cells will be ignored, and if the range is empty, an error may occur.
criteria_range1: This is the first **range** of cells that will be checked against the specified condition. For the function to correctly map the criteria to the average values, this range must have the exact same number of rows and columns as the `average_range`.
criterion1: This argument defines the condition that must be met within the `criteria_range1`. Criteria can be static numbers (e.g., `100`), text strings (e.g., `"East Region"`), logical expressions (e.g., `">50"`), or references to cells containing the required value. A critical rule is that all text criteria and logical expressions containing comparison operators must be enclosed in double quotation marks.
: These optional pairs allow for the implementation of additional, more granular filters. You can add as many range-criterion pairs as needed to define complex, multi-layered conditions. The function processes these pairs using the logical "AND" operator: a row's value is included in the average only if the associated cell in `criteria_range1` meets `criterion1` AND the cell in `criteria_range2` meets `criterion2`, and so on.

Preparing Your Data for Conditional Averaging

Before diving into practical examples, it is imperative to ensure your data is structured optimally for conditional analysis in **Google Sheets**. Effective data preparation is the bedrock of accurate **data analysis**, guaranteeing that the ranges referenced in your formulas are consistent and reliable. For demonstration purposes, we will utilize a sample **dataset** detailing hypothetical player statistics within a sports league, which records performance across different teams and roles.

The following table serves as our foundational dataset, meticulously organized into distinct columns that we will use as both averaging and criteria ranges throughout our examples:

	A	B	C	D
1	**Team**	**Position**	**Points**	
2	Mavs	Guard	22	
3	Mavs	Guard	28	
4	Mavs	Forward	25	
5	Mavs	Forward	30	
6	Mavs	Center	18	
7	Spurs	Guard	13	
8	Spurs	Guard	19	
9	Spurs	Forward	22	
10	Spurs	Forward	30	
11	Spurs	Center	11	
12				
13				
14				
15				
16				

This table contains three primary columns essential for our analysis: **Team** (Column A), which provides categorical data for the first set of criteria; **Position** (Column B), offering a second layer of categorical filtering; and **Points** (Column C), which holds the core numerical data that we intend to average. By maintaining clearly labeled columns and consistent data types within each column, we ensure maximum efficiency and accuracy when applying the `AVERAGEIFS` function to answer highly specific analytical questions about this raw data.

Example 1: Averaging with a Single Text Condition

The most straightforward application of **AVERAGEIFS** involves calculating an average based on a single condition, typically a text match. This is particularly useful when you need to isolate the performance or metric of a specific group, category, or entity within your large **dataset**, providing focused statistical intelligence.

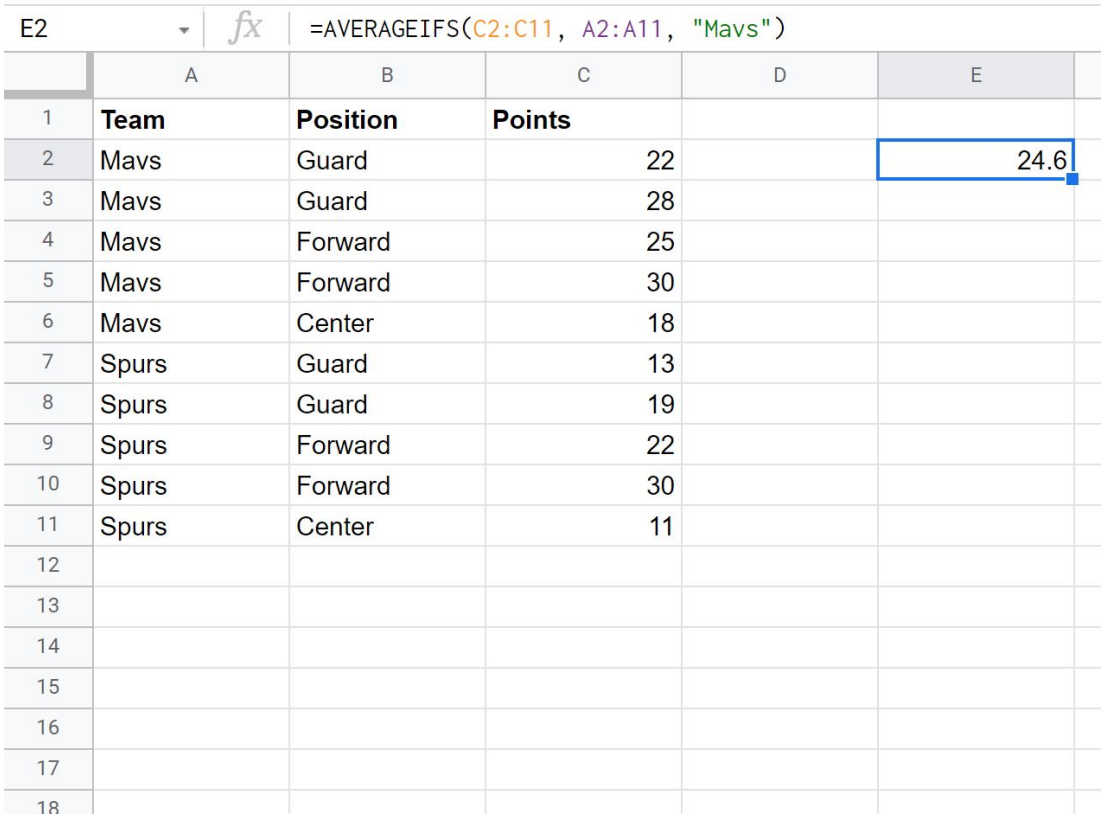
Our objective here is to determine the average points scored exclusively by players belonging to the "Mavs" team. To achieve this, we must configure the **formula** to examine the **Team** column (our criteria range) for all instances of the text string "Mavs" and then calculate the average of the corresponding values in the **Points** column (our average range).

The precise formula used to execute this single-condition calculation is structured as follows:

=AVERAGEIFS(C2:C11, A2:A11, "Mavs")

In this setup, the range `C2:C11` is explicitly defined as the numerical `average_range` (the **Points** column). The range `A2:A11` is specified as the `criteria_range1` (the **Team** column), and `"Mavs"` functions as the `criterion1`. The use of quotation marks around "Mavs" signals to **Google Sheets** that this is a required text string match. Only rows where the team name matches this exact string will have their points included in the calculation.

The visual application of this formula within the spreadsheet clearly highlights the data being processed and the final outcome:



	A	B	C	D	E
1	**Team**	**Position**	**Points**		
2	Mavs	Guard	22		24.6
3	Mavs	Guard	28		
4	Mavs	Forward	25		
5	Mavs	Forward	30		
6	Mavs	Center	18		
7	Spurs	Guard	13		
8	Spurs	Guard	19		
9	Spurs	Forward	22		
10	Spurs	Forward	30		
11	Spurs	Center	11		
12					
13					
14					
15					
16					
17					
18					

Upon execution, the result returned by **Google Sheets** is the figure **24.6**. This accurately represents the **average value** of points for all players identified as belonging to the "Mavs" team within the defined scope. To ensure complete confidence in the function, we can manually verify this: the Mavs players scored 22, 28, 25, 30, and 18 points. The sum is 123, and dividing by the count (5) yields precisely 24.6, confirming the function's accuracy.

Example 2: Averaging with Multiple Text Conditions

The true utility and efficiency of **AVERAGEIFS** become apparent when applying multiple conditions simultaneously. This scenario allows for far more refined and targeted **data analysis**, effectively narrowing down the focus to an extremely specific subset of the overall **dataset**, which is essential

for detailed performance metrics.

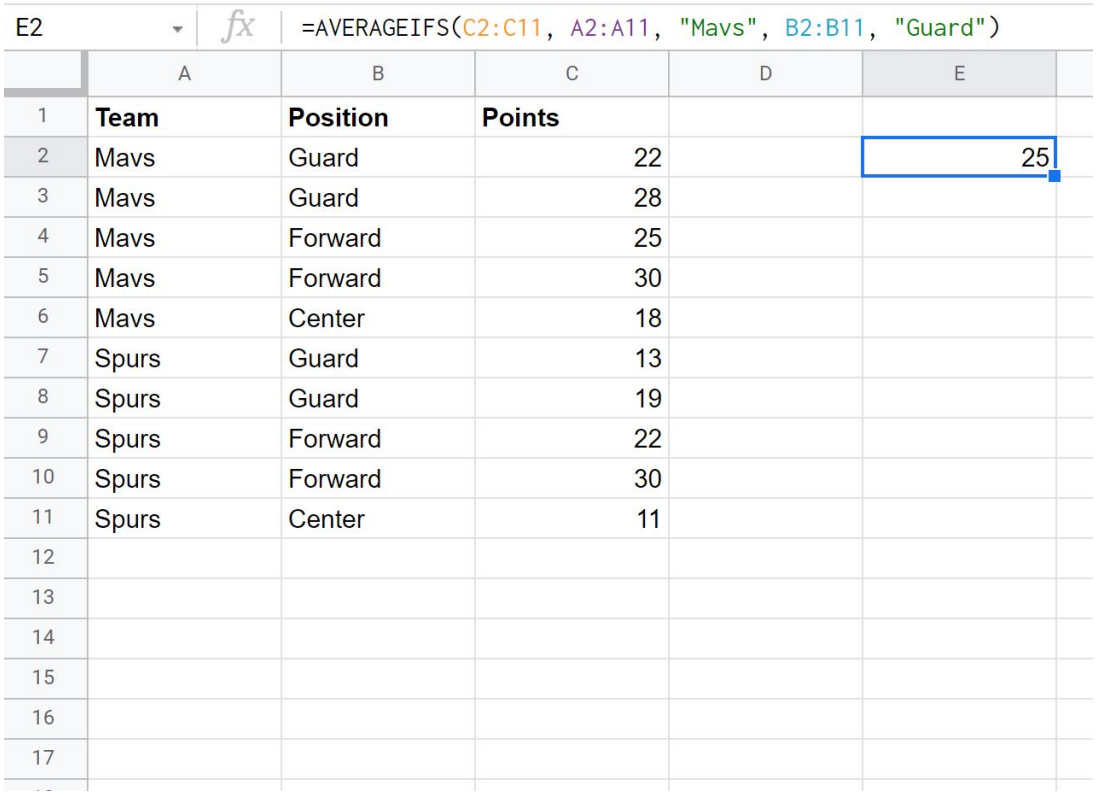
Let us now refine our previous query. Instead of simply finding the average points for all "Mavs" players, we want to know the average points scored only by "Mavs" players who specifically play the "Guard" position. This necessitates two distinct, mandatory conditions: the team must equal "Mavs" AND the position must equal "Guard." If a player meets one condition but not the other, their points will be excluded from the calculation, adhering strictly to the function's "AND" logic.

The **formula** must therefore incorporate a second criteria range and criterion pair into its **syntax** to handle this level of segmentation:

=AVERAGEIFS(C2:C11, A2:A11, "Mavs", B2:B11, "Guard")

In this expanded structure, `C2:C11` remains the `average_range` (Points). The first condition checks `A2:A11` for the "Mavs" **criterion**. Crucially, the second condition is added: `B2:B11` (the **Position** column) must match the "Guard" **criterion**. This dual filtering process ensures that the calculated average is highly specific, reflecting only the players who fit both categorical descriptions.

The visual output of this formula in **Google Sheets** confirms the result:



The calculated **average value** is exactly **25**. This figure represents the average points for the

specific cohort of players who are members of the "Mavs" team AND occupy the "Guard" position. Manual verification confirms this outcome: the qualifying players scored 22 and 28 points. The sum (50) divided by the count (2) equals 25, validating the function's ability to handle multiple textual conditions accurately.

Example 3: Combining Text and Numeric Conditions

The full flexibility of **AVERAGEIFS** is demonstrated when it is used to manage a blend of categorical (text) and quantitative (numeric or logical) **criteria**. This capability allows analysts to define a very narrow data window, such as calculating an average within a specific group but only for values that meet a minimum or maximum threshold.

For this scenario, we aim to find the average points for all "Spurs" players, but only those who have scored points greater than 15. This requires two distinct checks: first, a text match on the **Team** column, and second, a numeric comparison on the **Points** column. A key architectural point here is that the column we are averaging (Points, `C2:C11`) will also serve as one of our criteria ranges. This dual use of a **range** is a common and powerful technique in conditional aggregation.

The resulting **formula** structure must reflect this sophisticated mix of conditions:

=AVERAGEIFS(C2:C11, A2:A11, "Spurs", C2:C11, ">15")

The formula starts with the `average_range` (`C2:C11`). The first condition checks `A2:A11` for "Spurs." The second condition then checks the points themselves (`C2:C11` again) against the logical expression `">15"`. It is absolutely essential that comparison operators (such as `>`, `<`, `>=`, `<=`, and `<>`) be enclosed in quotation marks when defined as criteria within the `AVERAGEIFS` function, as they are treated as text strings defining a logical rule.

Observing the result of this combined conditional formula in the spreadsheet:

E2 =AVERAGEIFS(C2:C11, A2:A11, "Spurs", C2:C11, ">15")

	A	B	C	D	E
1	**Team**	**Position**	**Points**		
2	Mavs	Guard	22		23.66666667
3	Mavs	Guard	28		
4	Mavs	Forward	25		
5	Mavs	Forward	30		
6	Mavs	Center	18		
7	Spurs	Guard	13		
8	Spurs	Guard	19		
9	Spurs	Forward	22		
10	Spurs	Forward	30		
11	Spurs	Center	11		
12					
13					
14					
15					
16					
17					

The resultant **average value** is approximately **23.67**. This high-precision statistic reflects the average score of "Spurs" players who satisfy the quantitative requirement of scoring more than 15 points. To verify this result, we identify the qualifying Spurs players: Player 7 (19 points), Player 8 (22 points), and Player 10 (30 points). The sum (71) divided by the count (3) results in 23.666..., or 23.67 rounded, confirming the accurate application of both text and numeric filtering by the function.

Advanced Considerations and Best Practices

Moving beyond the basic examples, maximizing the efficiency and reliability of **AVERAGEIFS** requires implementing several key best practices. These techniques ensure your conditional averages are dynamic, user-friendly, and robust, particularly when dealing with changing criteria or large **dataset**s.

Utilizing Cell References for Dynamic Criteria: Hardcoding values directly into a **formula** (e.g., `"Mavs"` or `">15"`) makes your analysis rigid. A far superior approach is referencing a cell containing the desired **criterion**. For text matches, if "Mavs" is in cell Z1, the criterion is simply `Z1`. For numeric comparisons, you must use concatenation: if 15 is in cell Z2, the criterion becomes `">"&Z2`. This method significantly improves the flexibility of your spreadsheet models, allowing users to update criteria outside the complex formula structure.

Employing Wildcards for Flexible Text Matches: Google Sheets supports the use of wildcards within text criteria. The asterisk (`*`) represents any sequence of zero or more characters, while the question mark (`?`) represents any single character. For instance, using `"Mav*"` as a criterion would match "Mavs," "Maverick," or "Mavtown," enabling broader data analysis and pattern recognition without requiring exact text matches.
Implementing IFERROR for Graceful Error Handling: If no rows in your data meet all the specified criteria, **AVERAGEIFS** will attempt to divide by zero, resulting in a disruptive `#DIV/0!` error. To prevent this, wrap the function in an `IFERROR` statement. A robust implementation would look like: `=IFERROR(AVERAGEIFS(...), "No qualifying records found")`. This practice maintains a clean, professional appearance for your spreadsheets and provides clear feedback to the user.
Maintaining Consistent Range Sizes: A critical requirement of the `AVERAGEIFS` **syntax** is that all criteria ranges (`criteria_range1`, `criteria_range2`, etc.) must match the dimensions (number of rows and columns) of the `average_range`. Failing to adhere to this rule will often result in incorrect calculations or a `#VALUE!` error, as the function cannot correctly align the conditions with the values to be averaged on a row-by-row basis.

Conclusion and Further Learning

The **AVERAGEIFS** function is undoubtedly a cornerstone of sophisticated **data analysis** within **Google Sheets**. Its capacity to handle simultaneous, multi-criteria filtering allows users to move far beyond rudimentary statistical aggregates, enabling the extraction of highly specific **average value**s that are essential for granular reporting and informed, strategic decision-making across all sectors.

By mastering the function's logical structure and **syntax**, and by diligently applying the best practices discussed--particularly the use of cell references and careful error handling--you can significantly enhance the reliability and flexibility of your spreadsheets. This proficiency transforms raw data into actionable intelligence, revealing nuanced patterns and trends that would remain obscured by simpler aggregation methods.

We strongly encourage continued experimentation with `AVERAGEIFS` using diverse datasets and complex condition combinations. The process of applying these concepts in real-world scenarios is the most effective way to internalize the function's mechanics and maximize its analytical utility. To further deepen your expertise in conditional calculations and advanced spreadsheet techniques, we recommend exploring the following related resources:

Google Sheets Documentation for AVERAGEIFS: Consult the official help center for the most current information, compatibility notes, and detailed technical specifications of the function.
Understanding AVERAGEIF vs. AVERAGEIFS: Review the specific distinctions between the

single-condition and multi-condition averaging functions to ensure optimal selection for future tasks.

Using SUMIFS and COUNTIFS: Study the parallel aggregate functions, SUMIFS and COUNTIFS, which share the same powerful conditional structure and are indispensable for conditional summation and counting tasks.

Advanced Filtering Techniques in Google Sheets: Explore other complementary methods, such as FILTER function and Pivot Tables, which offer alternative ways to segment and analyze complex data structures.

These resources will guide your continuous development, helping you become an increasingly proficient and effective data analyst within the Google Sheets ecosystem.