Learning to Calculate Averages Using Multiple OR Criteria with Excel’s AVERAGEIFS Function

Authored by

Mohammed looti

November 10, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning to Calculate Averages Using Multiple OR Criteria with Excel’s AVERAGEIFS Function*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=15961

Mastering Conditional Averaging: Overcoming the AVERAGEIFS Limitation

The AVERAGEIFS function in Excel is an indispensable tool for performing sophisticated data aggregation. Its core utility lies in calculating the average of a range of cells that meet a set of specific criteria. Crucially, the function is fundamentally built upon **AND logic**, meaning that every single criterion specified must be satisfied simultaneously for a data point to be included in the calculation. This structure ensures precision when filtering datasets based on the strict intersection of conditions, suchable as determining the average sales performance for employees who are both located in the 'North Region' **AND** whose sales volume exceeds $50,000. This inherent design serves the majority of multi-criteria averaging needs efficiently and reliably.

While the native **AND logic** of AVERAGEIFS is highly effective for intersectional filtering, data analysis frequently demands a more flexible approach: conditional averaging based on **OR logic**. This requirement arises when we need to include records that satisfy condition A, condition B, or both. For example, calculating the average transaction value for customers categorized as 'Gold' **OR** 'Silver' members cannot be achieved using the standard arguments of **AVERAGEIFS** alone. Addressing this gap requires analysts to move beyond the function's default constraints, employing advanced Excel techniques to simulate disjunctive logic within the calculation framework.

Simulating OR Logic Using Array Formulas

The primary challenge in conditional averaging is the inability of standard functions like AVERAGEIFS to natively process disjunctions, or complex **OR logic**, across criteria ranges. If we attempt to average values corresponding to entries that must match one of several possible predefined criteria within a single column, the traditional syntax breaks down. To overcome this systemic limitation, we must harness the computational power of **array formulas**, specifically by combining the **AVERAGE** function with the powerful IF function, compelling Excel to evaluate logical conditions simultaneously across entire ranges.

The technical foundation for simulating **OR logic** within this array context hinges on the application of basic arithmetic, specifically the addition operator (+). In Excel's array processing environment, logical tests--which inherently return **TRUE** or **FALSE**--are coerced into numerical equivalents: **TRUE** translates to 1, and **FALSE** translates to 0. When multiple logical tests are added together, the resulting sum indicates how many conditions were met for that specific row. Therefore, if the sum of these tests is greater than zero (i.e., 1 or more), it successfully signifies that at least one of the specified conditions was fulfilled, thereby implementing the desired logical OR operation. This mathematical transformation is the key mechanism that allows complex filtering within a single formula structure.

Understanding this boolean transformation is fundamental to mastering array formulas. The

addition of logical arrays fundamentally performs the job of a disjunctive test, enabling the formula to selectively include data points where only one of the stipulated conditions is met. This technique moves beyond simple filtering towards true conditional aggregation, offering unparalleled control over which data subset contributes to the final average calculation.

Constructing the AVERAGE and IF Combination

The most robust and widely accepted method for integrating **OR logic** into conditional averaging involves the precise nesting of the IF function inside the **AVERAGE** function. This arrangement creates a dynamic array that selectively returns only the values from the target range that correspond to the rows where the established criteria have been successfully satisfied. The resulting formula structure is highly efficient because the **IF** function acts as a powerful filter, passing only the relevant numerical data points directly to the **AVERAGE** function for final computation, effectively ignoring all rows that do not meet the disjunctive criteria.

The general syntax required to calculate the average of a target range (TargetRange) where a criteria range (CriteriaRange) meets Condition 1 **OR** Condition 2 is structured as follows. Note how the logical tests are grouped and separated by the addition operator, which performs the crucial **OR** evaluation:

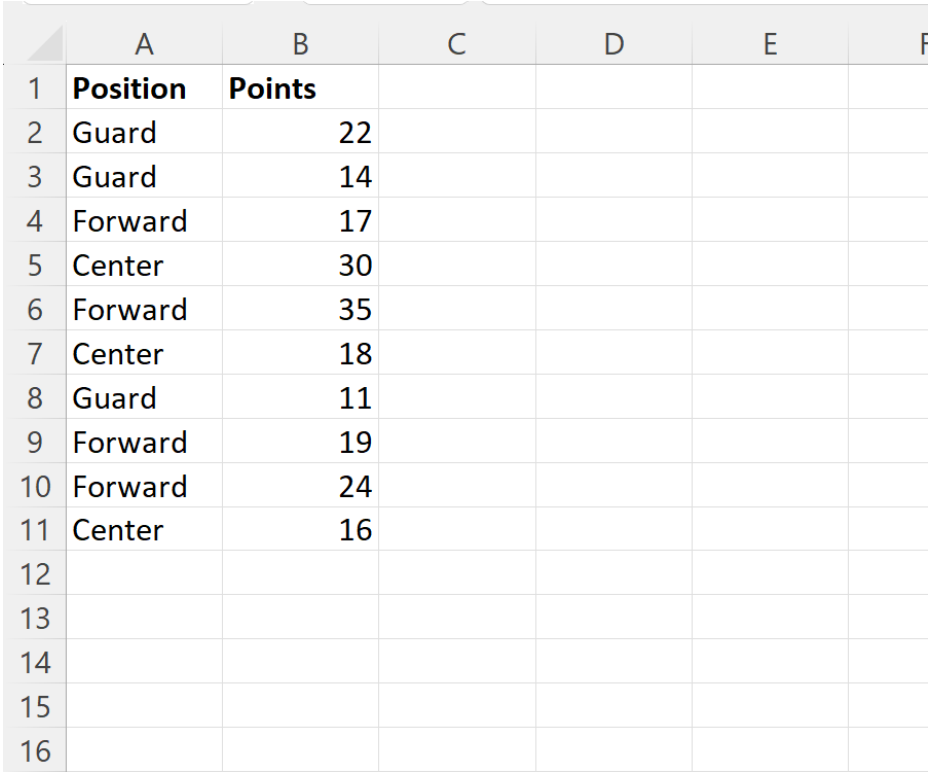
=AVERAGE(IF((A2:A11="Guard")+(A2:A11="Forward"),B2:B11))

Within this precise formula, the logical test segments `(A2:A11="Guard")` and `(A2:A11="Forward")` are assessed row by row across the specified range. If a corresponding cell in `A2:A11` matches 'Guard', the first test yields 1 (TRUE); if it matches 'Forward', the second test yields 1 (TRUE). The addition operator ensures that if one or both tests return 1, the total is greater than zero. Consequently, the **IF** function is triggered, returning the corresponding value from the target range (`B2:B11`) to the outer **AVERAGE** function. This method successfully filters the dataset to average only the values in range **B2:B11** where the positional data in **A2:A11** is equal to 'Guard' *or* 'Forward', providing a flexible solution that standard conditional functions cannot replicate.

Practical Application: Conditional Averaging in a Dataset

To demonstrate the practical utility of the **AVERAGE(IF(...))** array structure, consider a scenario involving sports analytics, specifically tracking basketball player performance. Our dataset includes detailed records showing the position of each player and the total points they accumulated over the course of a season. The goal is to perform a targeted analysis, calculating the average points scored exclusively by players whose position is designated as either "Guard" **OR** "Forward," while deliberately excluding the scores of players categorized as "Center." This requirement necessitates the precise implementation of the array formula technique to embed the crucial disjunctive criteria.

The sample data, organized neatly within an Excel worksheet, provides the raw material for our analysis. We are tasked with instructing Excel to systematically review the **Position** column (A2:A11) and, contingent upon satisfying the logical **OR** test, extract the corresponding data from the **Points** column (B2:B11). The image below illustrates the layout of this data, highlighting the relationship between the categorical criteria and the numerical values targeted for averaging:



	A	B	C	D	E	F
1	**Position**	**Points**				
2	Guard	22				
3	Guard	14				
4	Forward	17				
5	Center	30				
6	Forward	35				
7	Center	18				
8	Guard	11				
9	Forward	19				
10	Forward	24				
11	Center	16				
12						
13						
14						
15						
16						

The execution of this advanced calculation requires inputting the derived array formula directly into a designated results cell. The structure ensures that the data is meticulously filtered before the averaging process begins, guaranteeing that the resulting average accurately reflects only the players who fulfill the defined **OR condition**. This focused approach provides analysts with the capability to perform highly specific segment analysis that is otherwise unattainable using simple aggregated functions.

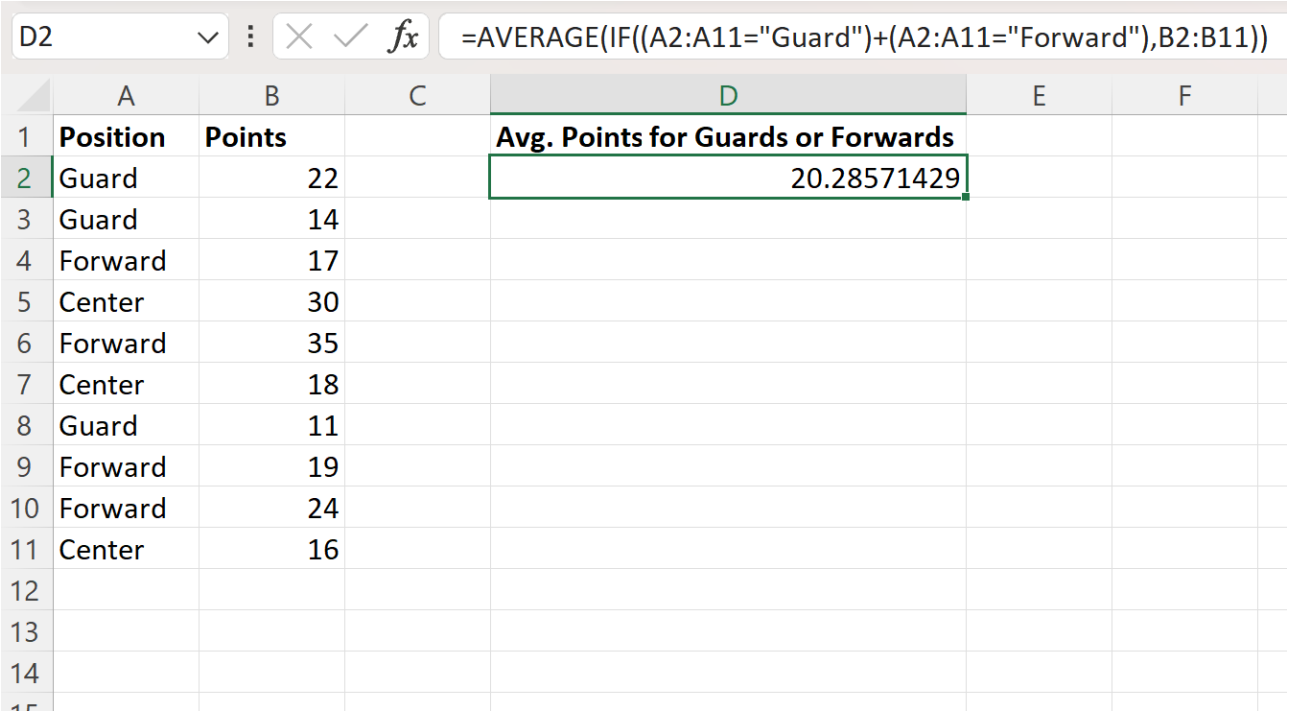
Implementation Steps and Calculation Verification

To successfully calculate the average score for players categorized as "Guard" or "Forward," the previously defined array structure must be correctly entered into the worksheet. The final formula, which encapsulates the entire filtering and averaging process, is:

=AVERAGE(IF((A2:A11="Guard")+(A2:A11="Forward"),B2:B11))

It is essential to note the method of confirmation required for **array formulas**. In legacy versions of Excel (prior to Excel 365 or Excel 2021), this formula must be finalized by pressing **Ctrl + Shift + Enter** simultaneously, which signals to Excel that it must process the formula as an array, visually denoted by the appearance of curly braces (`{}`) around the formula in the formula bar. Modern versions of Excel often handle this automatically via dynamic array functionality, but knowing the manual confirmation method remains critical for compatibility and understanding the underlying mechanism of array processing.

The result of applying this sophisticated filtering technique is displayed in the following screenshot, demonstrating the formula's execution within the worksheet environment:



	A	B	C	D	E	F
1	**Position**	**Points**		**Avg. Points for Guards or Forwards**		
2	Guard	22		20.28571429		
3	Guard	14				
4	Forward	17				
5	Center	30				
6	Forward	35				
7	Center	18				
8	Guard	11				
9	Forward	19				
10	Forward	24				
11	Center	16				
12						
13						
14						

As clearly indicated by the output, the calculated average points scored by players designated as Guard *or* Forward is precisely **20.2857**. This powerful application of nested functions allows analysts to conduct complex conditional analyses that extend far beyond the intrinsic limitations of standard conditional functions, ensuring accurate and targeted data insights.

To ensure the absolute accuracy and reliability of the calculated result derived from the array formula, a manual verification process is highly recommended. This involves systematically isolating and summing the data points that satisfy the established OR logic criteria. We must identify all rows where the position is either "Guard" or "Forward" and list their corresponding point values to confirm the calculation:

Guard: 22

Forward: 14
Guard: 17
Forward: 35
Guard: 11
Forward: 19
Guard: 24

Seven distinct data points satisfy the combined condition. The manual calculation confirms the process: Average Points = (22 + 14 + 17 + 35 + 11 + 19 + 24) / 7, which equals **142 / 7**. Performing this division confirms the result of **20.2857**. This verification step validates that the nested **AVERAGE** and **IF** array formula successfully executed the required **OR condition**, accurately filtering the source data and yielding the correct conditional average.

Integrating AND and OR Logic for Advanced Calculations

While the **AVERAGE(IF(...))** array structure provides the direct solution for conditional averaging with OR logic, it is beneficial to understand how this technique relates to other advanced calculations in Excel. For scenarios involving conditional summing or counting, the **SUMPRODUCT** function often serves as a robust, non-array alternative, especially appealing in older versions of Excel as it bypasses the need for the tedious Ctrl+Shift+Enter confirmation. However, for conditional averaging, the array method remains the gold standard for its directness and flexibility.

A significant strength of the array formula approach is its ability to seamlessly combine both **AND logic** and **OR logic** within a single construct. As established, **OR conditions** are implemented using the addition operator (+). Conversely, **AND conditions** are implemented using the multiplication operator (*). To achieve complex filtering--for example, averaging players who are either 'Guard' OR 'Forward' (OR condition) AND whose total points are greater than 20 (AND condition)--you simply group the OR conditions within parentheses and multiply the result by the AND condition's logical test. Mastery of these arithmetic operators in the array context is indispensable for professional data analysts seeking maximum flexibility in spreadsheet aggregation.

To further enhance your expertise in sophisticated data manipulation using conditional functions and array formulas in Excel, we recommend exploring resources focused on the following advanced topics:

Understanding the nuances of array processing and how Excel handles logical coercion (TRUE=1, FALSE=0).
Effective utilization of the IF function for constructing highly nested and complex logical tests.
Comparative analysis and optimal use cases for the AVERAGEIFS, SUMIFS, and COUNTIFS functions in combination with array methods.