# Learning VBA: Creating AVERAGEIF and AVERAGEIFS Functions in Excel

Authored by

**Mohammed looti**

November 15, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning VBA: Creating AVERAGEIF and AVERAGEIFS Functions in Excel*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=2359

## Introduction to Conditional Averaging with VBA

For advanced data analysts and power users aiming to automate sophisticated data aggregation tasks within Microsoft Excel, mastering conditional functions implemented through Visual Basic for Applications (**VBA**) is absolutely essential. This comprehensive guide provides a detailed blueprint for integrating Excel's robust built-in conditional averaging tools--specifically the AVERAGEIF and AVERAGEIFS functions--directly into your custom automation scripts. These functions are critical for calculating precise averages based on specific, predefined criteria, enabling flexible and nuanced data summarization far superior to simple arithmetic means. By embedding these capabilities into your VBA macros, you can significantly enhance the efficiency and analytical depth of your spreadsheet operations.

The programmatic application of conditional logic forms the foundation of advanced spreadsheet development. Whether the objective involves determining the average revenue generated solely by a particular product line or calculating player performance statistics based on multiple simultaneous attributes, combining **VBA** with the conditional averaging functions transforms complex analysis into a streamlined, automated workflow. Throughout this article, we will meticulously explore the precise syntax and practical implementation methodologies required to confidently deploy these powerful averaging tools within your custom **VBA** projects, starting with the single-criterion function.

## Understanding AVERAGEIF and the WorksheetFunction Bridge

The native **AVERAGEIF function** is specifically designed to compute the arithmetic mean of all numeric values within a designated range that successfully satisfy a single, defined criterion. When translating this powerful capability into the **VBA** environment, developers must utilize the WorksheetFunction object. This object serves as a vital bridge, granting your **VBA** code direct access to nearly all of Excel's native calculation engine functions. This seamless integration allows you to combine Excel's robust calculation power with the iterative and control flow structures inherent to **VBA** programming.

The calling convention for AVERAGEIF within a VBA macro is intuitive and follows the exact same parameter order as its native Excel counterpart. You are required to supply three primary arguments: first, the criteria range that holds the values to be tested against the condition; second, the specific criterion itself (which can be a text string, a numerical value, or a logical expression); and third (optional), the actual average range containing the numbers that will be aggregated. This method is ideally suited for straightforward statistical tasks where data must be filtered based on one singular condition prior to calculating the mean.

## Implementing AVERAGEIF Syntax in VBA

To practically demonstrate the mechanics of the **AVERAGEIF function** within an automated context, consider a typical business scenario: calculating the average of numerical metrics in one column contingent upon corresponding text matches in another column. The following VBA macro snippet illustrates the exact syntax required to execute this conditional calculation using the crucial WorksheetFunction object.

```
Sub Averageif_Function()
Range("E2") = WorksheetFunction.AverageIf(Range("A2:A12"), "Mavs", Range("B2:B12"))
End Sub
```

In the code provided above, the automated routine is explicitly tasked with computing the average of values residing within the data range **B2:B12**. Critically, this calculation is only performed for rows where the corresponding entry in the criteria range **A2:A12** precisely matches the specific text criterion, "Mavs". Once the conditional average is successfully calculated, the resulting numerical value is immediately written back to cell **E2** on the currently active worksheet. This highly structured and automated approach provides a clear method for extracting precise statistical insights tailored to a single, defined condition.

## Expanding Analytical Scope with AVERAGEIFS

While the **AVERAGEIF function** is effective for filtering data based on a single condition, real-world data analysis frequently necessitates aggregation based on numerous intersecting constraints. This requirement is fulfilled by the exceptionally powerful AVERAGEIFS function. **AVERAGEIFS** enables the calculation of averages exclusively for data points that satisfy two, three, or potentially dozens of conditions simultaneously. Like its single-criterion predecessor, this multi-conditional function is flawlessly accessed within **VBA** by utilizing the WorksheetFunction object, ensuring robust integration into any complex automation script.

A fundamental structural difference distinguishing AVERAGEIFS from **AVERAGEIF** lies in the ordering of its arguments. In **AVERAGEIFS**, the very first argument provided must be the **Average_Range** (the column containing the values intended for aggregation). This required argument is then followed by sequential pairs of the **Criteria_Range** and the specific Criterion to be applied to that range. This unique ordering significantly enhances flexibility, allowing developers to evaluate filtering conditions across multiple disparate columns before the arithmetic mean is finally computed. Grasping this distinct parameter structure is paramount for unlocking advanced analytical possibilities within automated Excel projects.

## Applying AVERAGEIFS for Complex Multi-Conditional Filtering

When data requirements demand aggregation based on multiple, intersecting conditions--such as averaging inventory values only for "Warehouse A" items that are also "low-stock"--the **AVERAGEIFS function** provides the definitive solution within **VBA**. The following code demonstrates a practical scenario where we simultaneously filter data based on both a text criterion and a numerical comparison criterion.

```
Sub Averageifs_Function()
Range("E2") = WorksheetFunction.AverageIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20")
End Sub
```

This specific VBA macro is meticulously designed to calculate the average of values located in the target range **C2:C12** (e.g., Assists). However, the resulting calculation is highly selective, proceeding only for those rows that successfully fulfill two distinct logical tests: first, the corresponding value in range **A2:A12** must precisely equal the string "Mavs"; **AND** second, the value in range **B2:B12** (e.g., Points) must be numerically greater than 20. The final, aggregated result of this multi-conditional analysis is then automatically outputted directly to cell **E2**. This example clearly showcases how `WorksheetFunction.AverageIfs` facilitates highly granular and specific data computation.

Before proceeding to the practical examples utilizing sample data, it is vital to firmly remember the mandated order of arguments for **AVERAGEIFS**: the range containing the numbers to be averaged must always be specified as the first argument, followed sequentially by the alternating pairs of criteria range and criterion. This strict sequence is necessary because the function requires immediate knowledge of the aggregation column before it can apply the filtering rules across the auxiliary criteria columns.

## Practical Example 1: Single Criterion (AVERAGEIF)

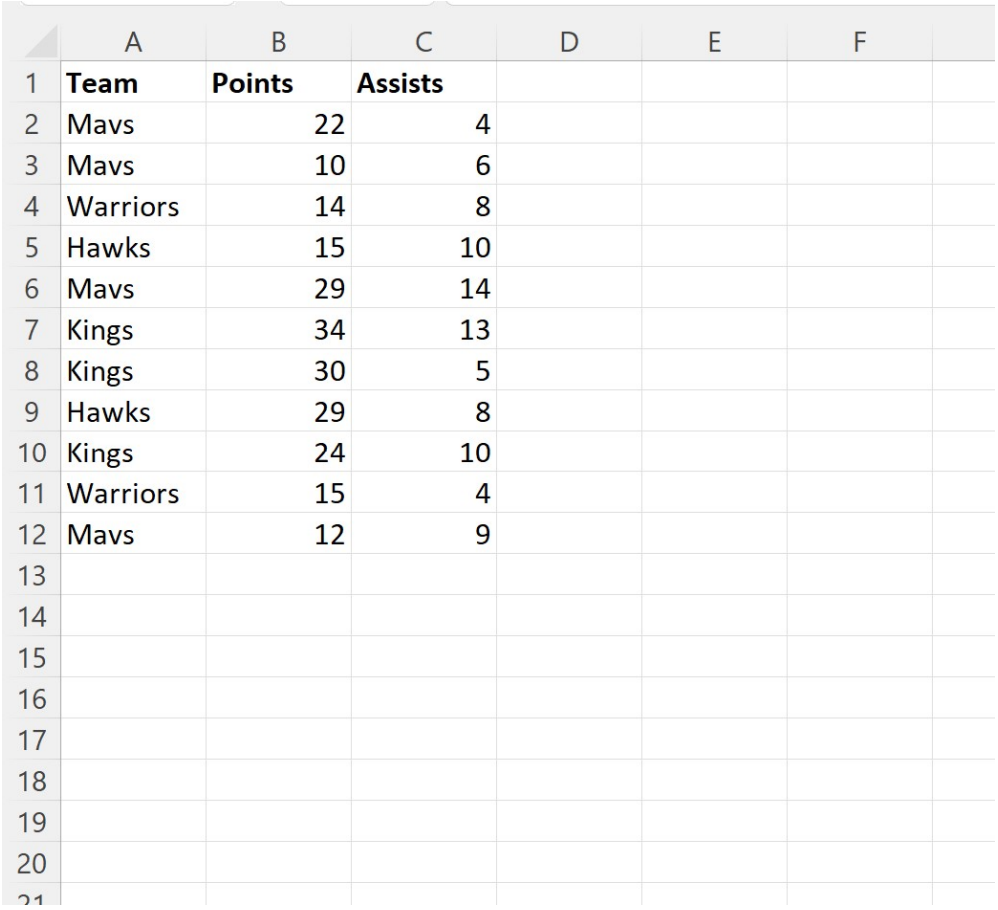
We will now apply the **AVERAGEIF** methodology to a concrete, practical dataset. Our objective in this initial example is straightforward: to accurately determine the average points scored exclusively by players designated as belonging to the "Mavs" team. Since this statistical requirement involves filtering based on only one distinct criterion (Team equals "Mavs"), the **AVERAGEIF function** is the most optimal and efficient tool for achieving this task.

To execute this conditional calculation, we deploy the following simple VBA macro. This routine invokes the WorksheetFunction.AverageIf method, instructing it to scan the team column (A2:A12), filter entries matching "Mavs", and calculate the average of the corresponding values in the points

column (B2:B12).

```
Sub Averageif_Function()
Range("E2") = WorksheetFunction.AverageIf(Range("A2:A12"), "Mavs", Range("B2:B12"))
End Sub
```

Upon successful execution of the code, the active target cell **E2** is populated with the precise result of the conditional average. This automated method effectively eliminates potential manual calculation errors and provides immediate statistical feedback. The image below illustrates the structure of the dataset utilized and indicates the location where the calculated average is displayed.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Team** | **Points** | **Assists** | | | |
| 2 | Mavs | 22 | 4 | | | |
| 3 | Mavs | 10 | 6 | | | |
| 4 | Warriors | 14 | 8 | | | |
| 5 | Hawks | 15 | 10 | | | |
| 6 | Mavs | 29 | 14 | | | |
| 7 | Kings | 34 | 13 | | | |
| 8 | Kings | 30 | 5 | | | |
| 9 | Hawks | 29 | 8 | | | |
| 10 | Kings | 24 | 10 | | | |
| 11 | Warriors | 15 | 4 | | | |
| 12 | Mavs | 12 | 9 | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |

As confirmed in the resulting output visualization below, cell **E2** now prominently holds the value **18.25**. This figure definitively represents the average points scored by all players within the dataset designated under the "Mavs" team affiliation. For manual validation, we can confirm the calculation against the source data: the points scored by Mavs players are 22, 10, 29, and 12. Summing these values (73) and dividing by the count (4) indeed yields **18.25**, thereby confirming the accuracy and reliability of the **VBA** implementation.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Team** | **Points** | **Assists** | | | |
| 2 | Mavs | 22 | 4 | | 18.25 | |
| 3 | Mavs | 10 | 6 | | | |
| 4 | Warriors | 14 | 8 | | | |
| 5 | Hawks | 15 | 10 | | | |
| 6 | Mavs | 29 | 14 | | | |
| 7 | Kings | 34 | 13 | | | |
| 8 | Kings | 30 | 5 | | | |
| 9 | Hawks | 29 | 8 | | | |
| 10 | Kings | 24 | 10 | | | |
| 11 | Warriors | 15 | 4 | | | |
| 12 | Mavs | 12 | 9 | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |

## Practical Example 2: Multiple Criteria (AVERAGEIFS)

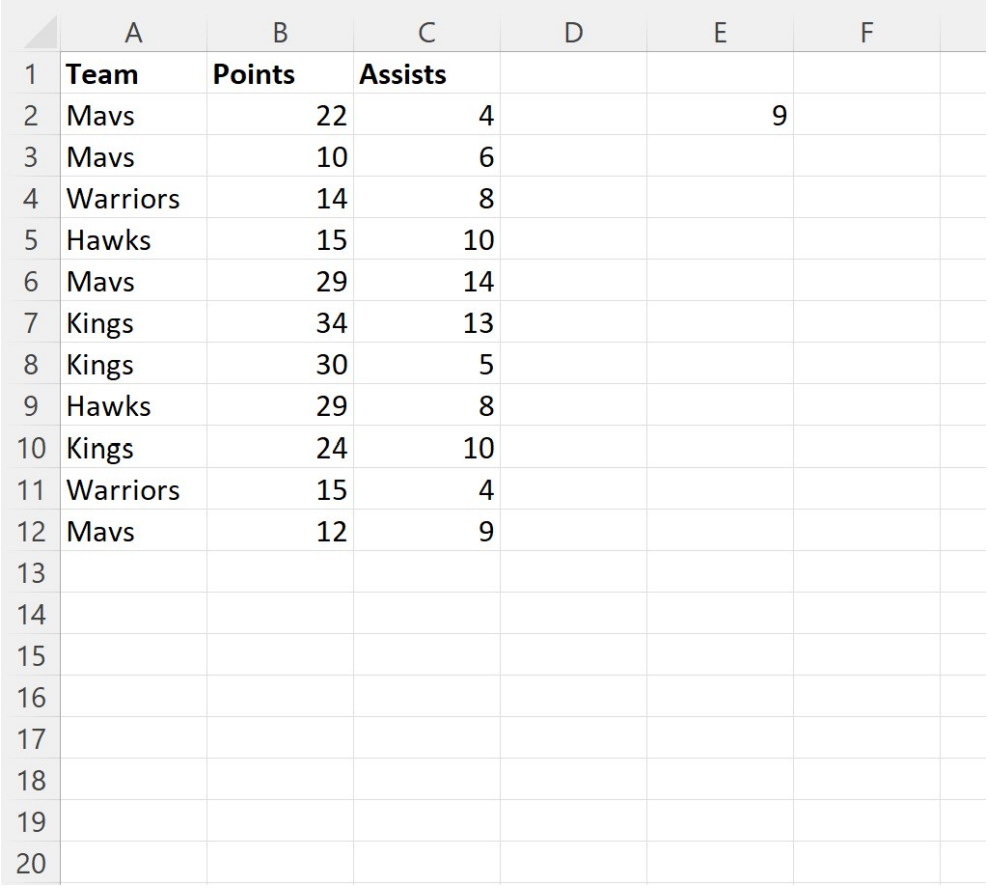
Building upon the foundational knowledge derived from the previous section, we now address a scenario demanding significantly greater analytical precision by introducing a dual filtering requirement. Suppose our organizational objective is to calculate the average number of assists (Column C) exclusively for players who satisfy two stringent conditions simultaneously: they must belong to the "Mavs" team **AND** they must have scored more than 20 points. This necessity for calculating a mean based on simultaneous condition evaluation makes the AVERAGEIFS function the only indispensable tool for this task.

We utilize the same dataset but must adapt our **VBA** routine to accommodate both conditions within the correct **AVERAGEIFS** parameter structure. The code below explicitly defines the range to average first (Assists, C2:C12), followed by the first criteria pair (Team Range A2:A12, with Criterion "Mavs"), and finally the second criteria pair (Points Range B2:B12, with Criterion ">20").

```
Sub Averageifs_Function()
Range("E2") = WorksheetFunction.AverageIfs(Range("C2:C12"), Range("A2:A12"), "Mavs", Range("B2:B12"), ">20")
```

**End Sub**

Once this refined **VBA macro** is executed, Excel diligently applies the filtering logic defined by both criteria simultaneously. The final average of assists for the precise subset of players meeting both conditions is then accurately recorded in cell **E2**. This powerful demonstration highlights the capability of **AVERAGEIFS** to conduct highly specific and targeted data aggregation across multiple columns.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Team** | **Points** | **Assists** | | | |
| 2 | Mavs | 22 | 4 | | 9 | |
| 3 | Mavs | 10 | 6 | | | |
| 4 | Warriors | 14 | 8 | | | |
| 5 | Hawks | 15 | 10 | | | |
| 6 | Mavs | 29 | 14 | | | |
| 7 | Kings | 34 | 13 | | | |
| 8 | Kings | 30 | 5 | | | |
| 9 | Hawks | 29 | 8 | | | |
| 10 | Kings | 24 | 10 | | | |
| 11 | Warriors | 15 | 4 | | | |
| 12 | Mavs | 12 | 9 | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |

Observing the output visualization, **E2** now correctly displays the value **9**. This calculated figure signifies that among the specific group of players who are members of the "Mavs" team **AND** who scored more than 20 points, their average number of assists is 9. It is crucial to **note** that the criterion logic employed here is highly scalable; while this example utilizes only two criteria ranges, the WorksheetFunction.AverageIfs method is designed to efficiently handle a far greater number of paired ranges and criteria as dictated by increasingly complex analytical needs.

## Conclusion: Mastering Conditional Aggregation

The mastery required to seamlessly integrate and deploy the **AVERAGEIF** and **AVERAGEIFS**

**functions** within **VBA** represents a significant professional advancement in automating and streamlining sophisticated conditional data analysis workflows in Excel. These methodologies empower developers and analysts to transcend simple, manual filtering by providing precise, programmatic control over how specific subsets of data are identified, isolated, and averaged, ultimately resulting in statistical reports that are significantly more accurate and nuanced.

Embracing these conditional averaging techniques is invaluable for any user seeking to successfully transition from traditional, time-consuming spreadsheet operations to automated, high-efficiency data processing pipelines. We strongly encourage all readers to actively experiment with diverse criteria types (such as working with dates, applying inequalities, or utilizing wildcard characters) and various datasets to fully grasp the remarkable versatility and robust efficiency that these specialized **VBA** functions introduce to data aggregation and reporting tasks.

## Additional Resources for VBA Development

For those interested in exploring additional **VBA** functionalities and automating other common **Excel** tasks, the following specialized tutorials provide further guidance on similar aggregation and formatting concepts:

VBA: How to Use SUMIF Function
VBA: Conditional Formatting Basics
VBA: Working with Arrays