Excel: Combine Duplicate Rows and Sum

Authored by

stats writer

November 17, 2025

RECOMMENDED CITATION

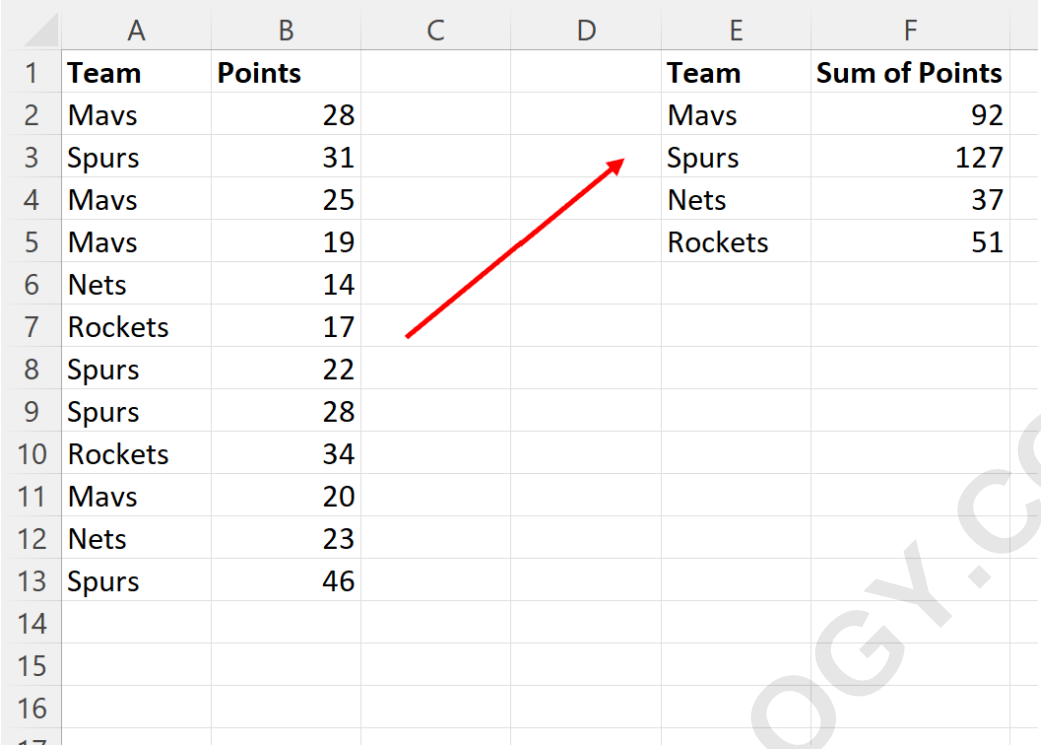
stats writer (2025). *Excel: Combine Duplicate Rows and Sum*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=94609

The Challenge of Data Aggregation in Excel

Handling large datasets often presents a common challenge in data management: consolidating redundant entries while preserving and summarizing associated numerical values. In business intelligence, finance, and scientific research, users frequently encounter spreadsheets where certain identifier columns--such as product names, employee IDs, or geographic regions--contain duplicate data. The objective is not simply to remove these duplicates, but rather to combine them into single, unique records and perform essential data aggregation, typically by summing related values from another column. This process is crucial for generating accurate summary reports and preparing data for further analysis.

The need to combine rows with duplicate identifiers and simultaneously sum their corresponding values is a fundamental task that, if approached inefficiently, can lead to complex and time-consuming manual manipulations. Imagine tracking sales records where a single product appears multiple times due to transactions spanning different dates. While the original detail is valuable, calculating the total revenue generated by that specific product requires aggregating the individual sales figures. Manually sorting, filtering, and then summing these values is error-prone and impractical for datasets containing thousands of rows.

Fortunately, modern versions of Excel provide powerful, streamlined solutions to this problem. By leveraging a combination of Dynamic Array Functions and specialized conditional calculations, we can achieve this aggregation quickly and reliably. Specifically, the combination of the **UNIQUE** function and the **SUMIF** function offers an elegant, two-step method to transform raw transactional data into a clean, summarized report, fulfilling the requirement to combine duplicate entries and sum their associated quantitative measures.



	A	B	C	D	E	F
1	**Team**	**Points**			**Team**	**Sum of Points**
2	Mavs	28			Mavs	92
3	Spurs	31			Spurs	127
4	Mavs	25			Nets	37
5	Mavs	19			Rockets	51
6	Nets	14				
7	Rockets	17				
8	Spurs	22				
9	Spurs	28				
10	Rockets	34				
11	Mavs	20				
12	Nets	23				
13	Spurs	46				
14						
15						
16						

Prerequisites: Understanding the Core Functions

Before diving into the practical example, it is essential to establish a strong understanding of the two principal functions we will be utilizing. These functions are the backbone of the aggregation process. The first function, **UNIQUE**, is a relatively recent addition to the Excel toolkit, introduced as part of the suite of Dynamic Array Functions. Its primary role is to extract distinct values from a specified range, automatically spilling the results into adjacent cells--a feature that significantly simplifies generating the list of unique identifiers needed for our summary table.

The second essential function is **SUMIF**, a long-standing and highly versatile function used for conditional summation. Unlike a simple **SUM** function, **SUMIF** allows the user to define a specific criterion (or condition) that must be met before a value is included in the total sum. In our context, **SUMIF** will reference the list of unique identifiers generated by the **UNIQUE** function and ensure that only the corresponding points or values for that specific identifier are aggregated, thereby achieving the desired combination and summation in a single step.

The synergy between **UNIQUE** and **SUMIF** is what makes this approach superior to older methods involving PivotTables or complex combinations of array formulas. The **UNIQUE** function dynamically creates the summary key, eliminating the need for manual copy-pasting and removal of duplicates. Subsequently, the **SUMIF** function efficiently cross-references this dynamically generated key against the original dataset to calculate the totals. This integration ensures that the resulting summary table is fully automated and will update instantaneously if the source data is

modified, providing a robust solution for ongoing data analysis.

Setting Up the Scenario: The Sample Dataset

To illustrate this powerful technique, we will utilize a practical scenario involving sports statistics. Suppose we are analyzing a dataset containing performance metrics for several basketball players across various teams. This raw data, located in columns A and B, records the team name alongside the total points scored by players associated with that team in specific games or periods. Crucially, the dataset shows multiple entries for the same team, reflecting the need for aggregation.

Our primary goal is to transform this transactional list into a summary table that clearly shows the total combined points scored by each unique team. This requires identifying the unique team names first, and then conditionally summing all point entries corresponding to that specific team name. The data structure is straightforward, consisting of two main fields: the categorical identifier (Team Name) and the quantitative measure (Points Scored).

The initial dataset looks like this, where we can clearly observe that entries such as the **Team** column are repeated multiple times across the rows. This redundancy in the 'Team' column is exactly what we need to address using our combined function approach. The visual representation below confirms the structure of our starting data:

	A	B	C	D	E	F
1	**Team**	**Points**				
2	Mavs	28				
3	Spurs	31				
4	Mavs	25				
5	Mavs	19				
6	Nets	14				
7	Rockets	17				
8	Spurs	22				
9	Spurs	28				
10	Rockets	34				
11	Mavs	20				
12	Nets	23				
13	Spurs	46				
14						
15						
16						
17						

Step 1: Extracting Unique Identifiers using the UNIQUE Function

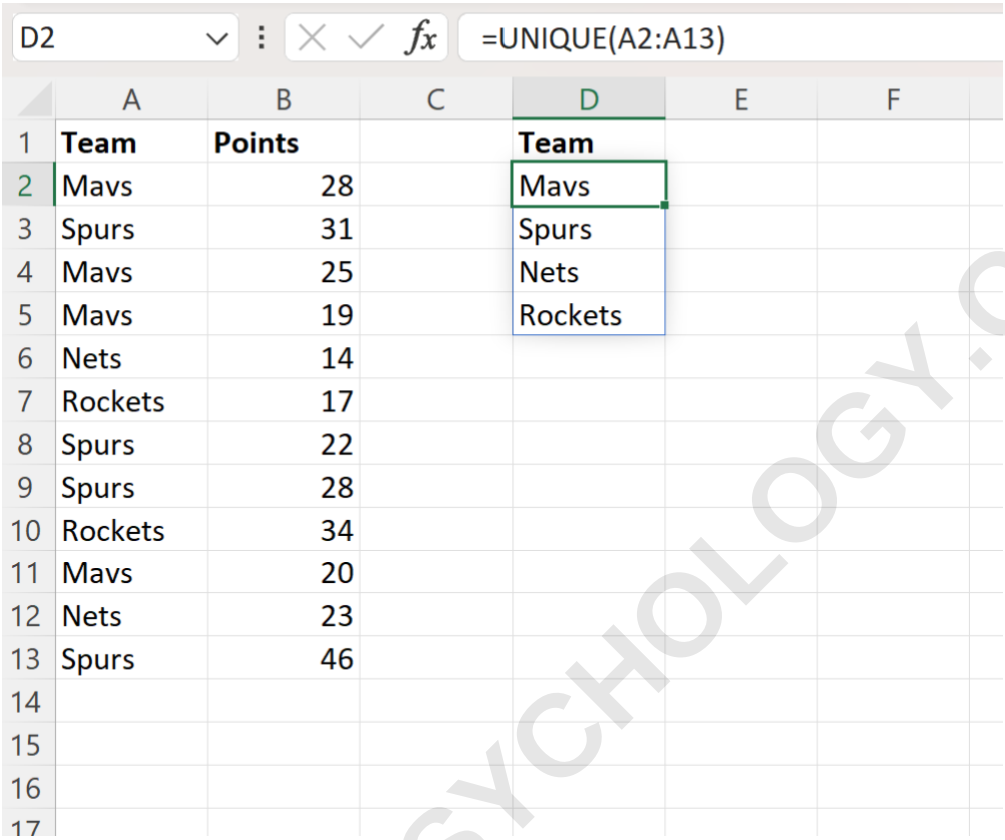
The first critical step in achieving our aggregated summary is to generate a list of all unique team names present in the dataset. This list will serve as the foundation for our summary report. Instead of manually copying the 'Team' column and using the 'Remove Duplicates' feature--a static process that requires repetition whenever the source data changes--we employ the **UNIQUE** function, which is designed specifically for this purpose and provides dynamic updating.

To implement this, we navigate to an unused cell, typically adjacent to the source data, such as cell **D2**. In this cell, we input the formula that references the entire range containing the team names (A2 through A13 in our example). The **UNIQUE** function requires only one primary argument: the array or range from which to extract the unique values.

The syntax for this implementation is remarkably simple. We instruct Excel to look at the team column and return only the distinct entries. By typing the following formula into cell **D2**, Excel automatically identifies the unique values ("Mavs," "Knicks," "Lakers," etc.) and populates the cells below D2 dynamically.

=UNIQUE(A2:A13)

The dynamic array capability means that the output will "spill" down into cells D3, D4, and D5 without needing to drag the formula down. This instantly provides us with the consolidated list of teams, which acts as our unique criterion range for the subsequent summation step. The following visual confirmation demonstrates the immediate result of applying the **UNIQUE** function:



	A	B	C	D	E	F
1	**Team**	**Points**		**Team**		
2	Mavs	28		Mavs		
3	Spurs	31		Spurs		
4	Mavs	25		Nets		
5	Mavs	19		Rockets		
6	Nets	14				
7	Rockets	17				
8	Spurs	22				
9	Spurs	28				
10	Rockets	34				
11	Mavs	20				
12	Nets	23				
13	Spurs	46				
14						
15						
16						
17						

Deep Dive into the UNIQUE Function Syntax and Best Practices

While the basic use of the **UNIQUE** function is straightforward, understanding its optional arguments allows for greater flexibility when handling complex datasets. The full syntax of the **UNIQUE** function is: `UNIQUE(array, , )`. In our basic example, we only used the required `array` argument (A2:A13), relying on the default behavior to return unique rows (or values) and all values, not just those appearing exactly once.

The optional argument is a logical value (TRUE or FALSE) that dictates whether to compare by row (FALSE, the default) or by column (TRUE). Since our data identifiers were organized vertically in a column, the default setting sufficed. The second optional argument, , is also a logical value. If set to TRUE, the function only returns values that appear in the range only once. If set to FALSE (the default, which we used), it returns every unique value, even those that appear multiple times in the source data, which is necessary for our aggregation goal.

A significant benefit of using the **UNIQUE** function is its integration with Dynamic Array Functions technology. If a new team name is added to the source data (e.g., in cell A14), the list generated by **UNIQUE** in column D will automatically expand to include the new entry. However, to ensure this feature works seamlessly, it is often best practice to use structured table references or dynamic named ranges rather than fixed cell ranges (like A2:A13). For simplicity in this introductory example, we used the fixed range, but advanced users should consider using references like `A2:A1000` or, ideally, `Table1`.

Step 2: Applying Conditional Summation with SUMIF

With our list of unique identifiers now established in column D, the next step is to calculate the total points corresponding to each team. This is where the **SUMIF** function becomes invaluable. We need a calculation that checks the team name in the original data range, compares it against the unique team names we generated, and if they match, includes the corresponding points in the sum.

We will place this conditional summation formula in the cell adjacent to the first unique team name, specifically cell **E2**. The **SUMIF function** requires three primary arguments: the range to check the condition against, the criterion (the condition itself), and the range of values to sum.

For the "range to check" argument, we must reference the original team column (A2:A13), ensuring we use **absolute references** (e.g., $A$2:$A$13) so the range remains fixed when the formula is copied down. The "criterion" argument refers to the specific unique team name we are currently calculating, which is the value in cell **D2**. Finally, the "sum range" is the column containing the values we wish to aggregate--the 'Points' column (B2:B13), also secured with absolute references ($B$2:$B$13).

The resulting formula typed into cell **E2** looks like this:

=SUMIF($A$2:$A$13, D2, $B$2:$B$13)

Once the formula is entered into E2, we can simply drag the fill handle down from E2 to E5 (or rely on the spill behavior if using a spilled range reference in the criterion). This action applies the **SUMIF** calculation for every unique team listed in column D, successfully combining the duplicate rows and summing their associated points.

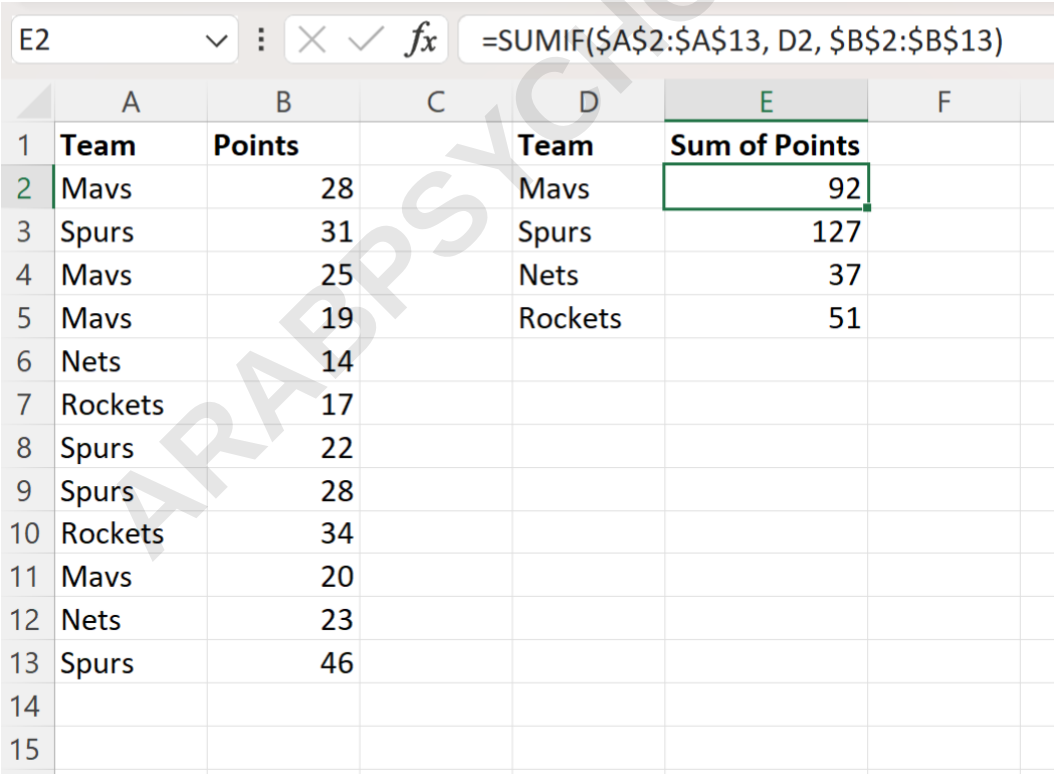
Detailed Breakdown of the SUMIF Arguments

Understanding the precise role of each argument in the **SUMIF function** is vital for troubleshooting and modifying the formula for different datasets. The structure `SUMIF(range, criteria, sum_range)` is deliberate and dictates the flow of the conditional check.

The first argument, `range` (`$A$2:$A$13`), specifies the set of cells where the condition will be evaluated. This range must contain the potential duplicate identifiers (the Team names). Using the absolute references ($$) here is a critical practice. If we failed to fix the row and column references, dragging the formula down would shift the range (e.g., from A2:A13 to A3:A14), causing calculation errors or omitting data.

The second argument, `criteria` (`D2`), is the condition itself. This cell contains the specific unique team name (e.g., "Mavs") that the **SUMIF** function must match within the `range` (A2:A13). Crucially, this argument is intentionally left as a **relative reference** (D2, not $D$2). This allows the reference to update automatically to D3, D4, and D5 as the formula is copied down, ensuring that each calculated sum corresponds to the correct unique team name.

Finally, the third argument, `sum_range` (`$B$2:$B$13`), identifies the cells containing the numerical values that should be added together when the corresponding entry in the `range` meets the `criteria`. Like the first argument, this range must also be secured with **absolute references** to maintain a fixed, correct reference to the 'Points' column throughout all calculations. This conditional logic effectively filters the entire 'Points' column based on the unique team name, providing the final aggregated score.



	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Sum of Points**	
2	Mavs	28		Mavs	92	
3	Spurs	31		Spurs	127	
4	Mavs	25		Nets	37	
5	Mavs	19		Rockets	51	
6	Nets	14				
7	Rockets	17				
8	Spurs	22				
9	Spurs	28				
10	Rockets	34				
11	Mavs	20				
12	Nets	23				
13	Spurs	46				
14						
15						

Interpreting the Results and Data Validation

The final output, spanning columns D and E, represents the complete, combined, and summed dataset. Column D provides the clean, unique list of teams, and Column E presents the total aggregated points scored by all players associated with that team across the entire original dataset. This transformation effectively converts a transactional ledger into a summary report.

For instance, examining the result for "Mavs" shows a total sum of 92 points. We can validate this by manually checking the original source data (A2:B13) and confirming all rows labeled "Mavs" sum up to 92 (e.g., 20 + 35 + 37 = 92). Similarly, the total points for "Spurs" are aggregated to 127. This result confirms that the **SUMIF** function correctly identified the conditional matches and accumulated the corresponding values, successfully resolving the initial data aggregation challenge.

This dynamic method offers superior data integrity compared to manual methods. If any player's points or team name in the original A2:B13 range is updated, the **UNIQUE** list in column D and the corresponding sums in column E will instantly recalculate, ensuring the summary report remains accurate without any user intervention. This level of automation is essential for maintaining consistent and reliable reports, particularly in fast-paced data environments.

The sum of points values scored for all players on the **Mavs** team is **92**, calculated by combining all individual Mavs entries.
The sum of points values scored for all players on the **Spurs** team is **127**, demonstrating the effectiveness of the conditional summation.
The sums for all other unique teams are calculated automatically using the same **SUMIF function** structure.

Conclusion: Mastering Dynamic Array Functions for Data Management

The technique of combining the **UNIQUE** and **SUMIF** function provides an efficient and scalable methodology for handling data aggregation challenges in Excel. This method bypasses the complexity of older array formulas and the manual steps associated with sorting and filtering, delivering a clean, dynamically updated summary table.

By first utilizing the **UNIQUE** function, we establish the necessary framework of distinct criteria. Following this, the precise application of the **SUMIF** function, particularly the careful use of absolute and relative references, ensures that the summation accurately maps the individual transactional entries back to their respective aggregated category. This two-step process is a powerful illustration of modern Excel capabilities, especially the integration of Dynamic Array Functions.

Mastering this combination is fundamental for any serious data analyst working with spreadsheets. It enables rapid conversion of raw, detailed transactional data into actionable summary reports,

ensuring both accuracy and efficiency. This method is applicable across virtually any domain where data requires consolidation based on categorical identifiers and summation of associated numerical values.