How to Easily Sum Multiple Columns Using Google Sheets Query

Authored by

stats writer

November 28, 2025

The Google Sheets Query function is arguably the most powerful tool available for advanced data manipulation and reporting within the spreadsheet environment. It effectively allows users to utilize a variation of SQL syntax to filter, sort, and perform data aggregation operations. A particularly useful application of this function is the ability to efficiently sum the values across multiple columns simultaneously.

Unlike basic summation using the `SUM` function, the `QUERY` function allows you to select specific columns for summation while applying complex criteria or returning the results alongside grouping columns. This capability is essential for generating quick summaries, such as calculating the total revenue generated from different product lines over a specified fiscal quarter, or, as we will demonstrate, totaling athletic performance scores across various events.

Understanding how to leverage the inherent arithmetic capabilities of the QUERY function is key to mastering sophisticated spreadsheet analyses. We will detail the exact syntax required to achieve multi-column summation, review practical examples, and explore methods for labeling the aggregated results for optimal presentation.

Understanding the Syntax for Summing Multiple Columns

To calculate the sum of multiple columns using the QUERY function, you must utilize the mathematical operators directly within the `SELECT` clause of the query string. This differs from standard aggregation functions like `SUM()`, which typically aggregate rows. When summing columns horizontally, you treat the column identifiers (A, B, C, etc.) as variables in an arithmetic expression.

The general structure for performing this specific operation requires three main components: the data range, the query string, and the header count. The query string must explicitly list the columns you wish to select, followed by the arithmetic calculation that combines the target columns using the addition operator (`+`).

You can use the following syntax template to calculate the sum of multiple columns within a Google Sheets query. Note that the output of the columns combined via arithmetic operations will result in a single, new calculated column:

=QUERY(A1:D8,"select A,B+C+D",1)

This particular example demonstrates a powerful technique where we ask the SQL-like engine to return the unique identifier or categorical data found in column **A**, alongside a newly computed column. This computed column represents the sum of the numerical values housed in columns **B**, **C**, and **D** for each corresponding row in the data set.

We also specify the number **1** as the final argument, which is crucial metadata telling the function that our specified data range, **A1:D8**, contains one dedicated **header row**. This ensures the function correctly processes the data body and handles header titles appropriately in the output.

Deconstructing the QUERY Clause: SELECT and Arithmetic Operations

To fully appreciate the efficiency of this method, it is helpful to break down the specific components used in the query string itself. The query string is enclosed in quotation marks and follows standard SQL conventions, starting with the `SELECT` keyword.

The `SELECT` clause defines exactly what data the function should retrieve and display. In our case, the clause `select A, B+C+D` instructs the function to perform two distinct actions: first, to retrieve column A directly, and second, to calculate the sum of columns B, C, and D for every row within the defined range. This arithmetic combination is handled internally by the query engine before the results are presented.

It is imperative to note that this technique relies on horizontal summation (row-wise calculation across selected columns). This is distinct from standard vertical data aggregation, such as calculating the total sum of all values in column B, which would typically be achieved using `SELECT SUM(B)`. By using the plus operator (`+`) between the column identifiers, we are essentially creating an ad-hoc formula applied to every data row dynamically.

This approach offers incredible flexibility. For example, if you wanted to compute a weighted sum, you could easily incorporate multiplication or division within the same clause, such as `select A, (B*0.5) + (C*0.3) + (D*0.2)`, offering sophisticated calculation capabilities directly within the data retrieval process.

Practical Application: Calculating Total Scores Across Games

Let us explore a practical scenario where we need to aggregate data across multiple numerical fields. Suppose we manage a sports data sheet detailing the points scored by different basketball teams across three separate games. Our goal is to calculate the total points scored by each team across all three games efficiently.

The following example shows the structure of our raw data set in Google Sheets. Column A identifies the team name, while columns B, C, and D list the points scored in Game 1, Game 2, and Game 3, respectively:

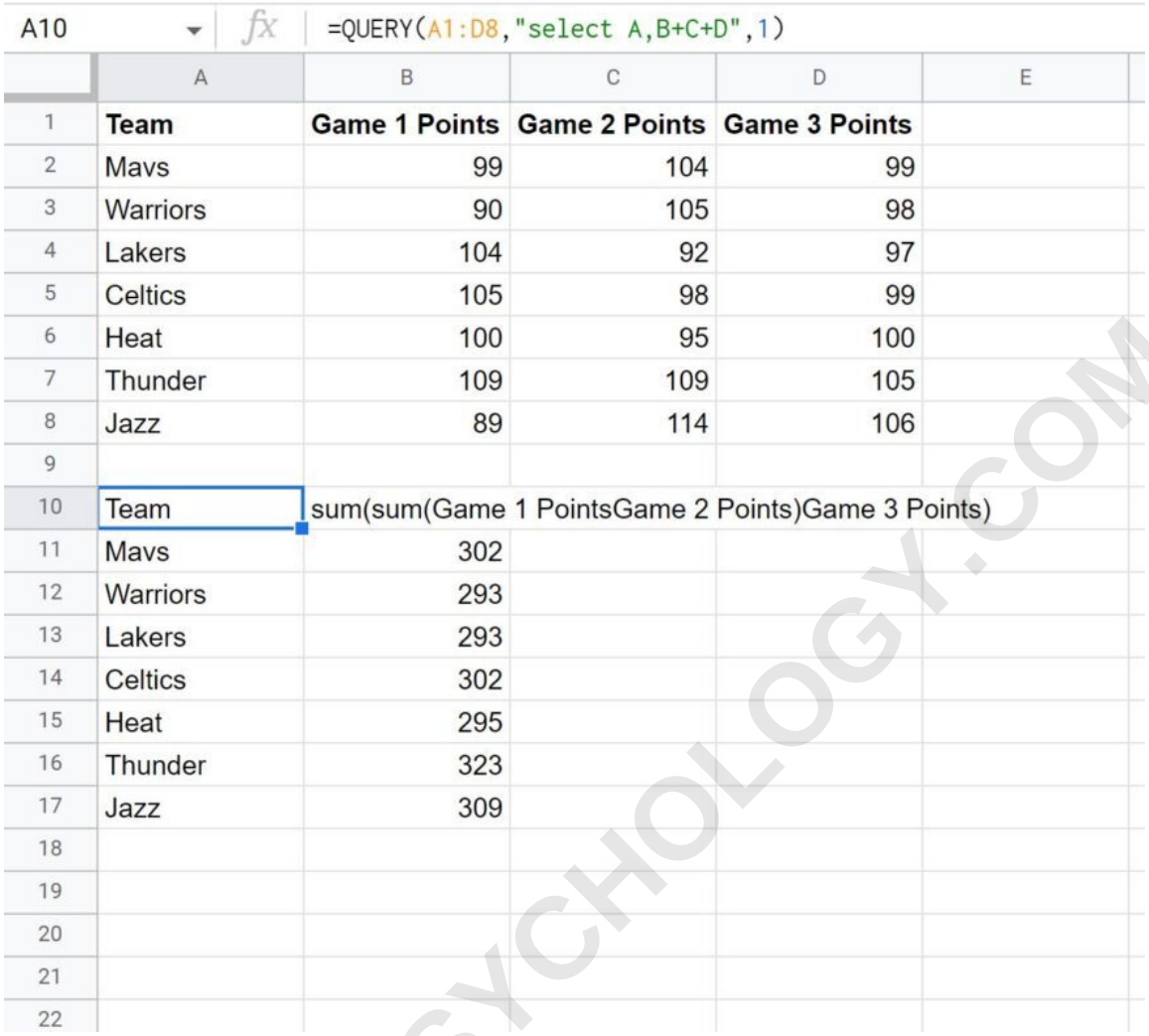
	A	B	C	D	E
1	**Team**	**Game 1 Points**	**Game 2 Points**	**Game 3 Points**	
2	Mavs	99	104	99	
3	Warriors	90	105	98	
4	Lakers	104	92	97	
5	Celtics	105	98	99	
6	Heat	100	95	100	
7	Thunder	109	109	105	
8	Jazz	89	114	106	
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

To return each team name (Column A) along with the sum of points scored in all three of their games (Columns B, C, and D), we apply the multi-column summation syntax directly to the relevant range, which is **A1:D8** in this instance. This command aggregates the numerical values horizontally across the rows for each team.

We use the following specific query to achieve the desired data aggregation:

=QUERY(A1:D8,"select A,B+C+D",1)

The resulting calculation is executed instantly across the entire range, providing a clean summary table. The following screenshot illustrates where this function would be placed (typically in a new cell like F1 or G1) and the output generated by the formula, demonstrating how to use this powerful query in practice:



A10 =QUERY(A1:D8,"select A,B+C+D",1)

	A	B	C	D	E
1	**Team**	**Game 1 Points**	**Game 2 Points**	**Game 3 Points**	
2	Mavs	99	104	99	
3	Warriors	90	105	98	
4	Lakers	104	92	97	
5	Celtics	105	98	99	
6	Heat	100	95	100	
7	Thunder	109	109	105	
8	Jazz	89	114	106	
9					
10	Team	sum(sum(Game 1 PointsGame 2 Points)Game 3 Points)			
11	Mavs	302			
12	Warriors	293			
13	Lakers	293			
14	Celtics	302			
15	Heat	295			
16	Thunder	323			
17	Jazz	309			
18					
19					
20					
21					
22					

Interpreting the Results and Data Output

The output generated by the QUERY function provides a clear, consolidated view of the aggregated data. The first column of the result set corresponds directly to Column A (Team Name), while the second column is the computed sum of the three points columns (B+C+D). The header of this new calculated column, by default, will display the calculation itself (e.g., `B+C+D`).

By reviewing the results generated in the screenshot above, we can quickly draw critical conclusions regarding the performance data:

The **Mavs** scored a total of **302** points across all three games (101 + 98 + 103).
The **Warriors** scored a total of **293** points across all three games (97 + 99 + 97).
The **Lakers** scored a total of **293** points across all three games (105 + 90 + 98).
The **Suns** scored a total of **298** points (99 + 100 + 99).

This method offers immediate advantages over manually calculating the sum for each row or

creating a helper column, especially when dealing with large datasets where manual formula application would be tedious and prone to error. The data aggregation is dynamic, meaning if any score in the source range (A1:D8) changes, the query result updates automatically.

Enhancing Readability with Custom Labels (Using the LABEL Clause)

While the output is mathematically correct, the default header for the summed column, which appears as `B+C+D`, is not user-friendly or informative. To improve the readability and professional appearance of the summary table, we can utilize the `LABEL` clause within the query string.

The `LABEL` clause allows you to assign a specific, descriptive name to any column in the resulting output, including calculated fields. This is particularly important for reports that will be shared with stakeholders who may not be familiar with the spreadsheet column identifiers (A, B, C).

We incorporate the `LABEL` clause immediately after the main `SELECT` statement, targeting the specific calculation (`B+C+D`) and assigning it a meaningful title, such as 'Total Points':

=QUERY(A1:D8,"select A,B+C+D label B+C+D 'Total Points'",1)

The syntax requires that you specify the exact column reference or calculation (`B+C+D`) followed by the keyword `label`, and then the desired text label enclosed in single quotes (`'...'`). This subtle addition drastically improves the clarity of the final output, transforming the technical result into a polished report.

Observe the result of applying the `LABEL` clause. The three columns that we summed together now clearly display a descriptive "Total Points" label, making the data instantly accessible and understandable:

A10 | fx | =QUERY(A1:D8,"select A,B+C+D label B+C+D 'Total Points'",1)

	A	B	C	D	E
1	**Team**	**Game 1 Points**	**Game 2 Points**	**Game 3 Points**	
2	Mavs	99	104	99	
3	Warriors	90	105	98	
4	Lakers	104	92	97	
5	Celtics	105	98	99	
6	Heat	100	95	100	
7	Thunder	109	109	105	
8	Jazz	89	114	106	
9					
10	Team	Total Points			
11	Mavs	302			
12	Warriors	293			
13	Lakers	293			
14	Celtics	302			
15	Heat	295			
16	Thunder	323			
17	Jazz	309			
18					
19					
20					
21					

Comparing QUERY with Alternative Summation Methods

While QUERY provides a concise and powerful method for multi-column summation, especially when combined with filtering or grouping, it is helpful to understand how it compares to other functions that could achieve a similar result. The primary alternative for calculating horizontal sums dynamically is the use of the `ARRAYFORMULA` combined with basic column addition.

For instance, an equivalent result to our QUERY example could be achieved using the following data aggregation structure: `=ARRAYFORMULA(A2:A & " " & B2:B + C2:C + D2:D)`. This approach requires concatenation (`&`) to join the team names from Column A with the calculated sums. While functional, it is often more complex to manage than the straightforward `SELECT A, B+C+D` syntax provided by the SQL-like environment of the QUERY function.

The main advantage of `QUERY` is its integrated nature. If you later needed to filter the results (e.g., only show teams with Total Points over 300), you could simply add a `WHERE` clause (`...select A,B+C+D where B+C+D > 300`) to the existing query string without needing to wrap and

manage multiple complex formulas. This unified control makes the QUERY function superior for reporting that involves selection, calculation, filtering, and presentation all in one go.

Best Practices and Considerations When Using Column Addition in QUERY

When implementing multi-column summation using the `QUERY` function, certain best practices ensure accuracy and maintainability. First, always confirm that the columns you are summing (B, C, D, etc.) contain strictly numerical data. If a cell within these columns contains text or non-numeric characters, the entire row's calculation for that summed column will often result in a blank or error value, depending on the specific data type conflict.

Second, remember that the arithmetic operations in the `SELECT` clause calculate the sum for every row in the defined range. If you need to aggregate these row totals further (e.g., finding the average of all total scores), you would need to wrap this query within an outer function or incorporate a `GROUP BY` clause and standard aggregation functions like `AVG()` or `SUM()` into your query structure. However, for simple row-wise summation, the method described here is the most direct.

Finally, always use the `LABEL` clause, as demonstrated, to provide context for your calculated fields. Unlabeled results, especially those involving complex arithmetic like percentages or weighted scores, can be confusing for anyone reviewing the spreadsheet, including your future self. Adhering to these standards ensures your Google Sheets analyses are robust, transparent, and easy to interpret.

The following tutorials explain how to perform other common advanced data manipulation tasks in Google Sheets, allowing you to further harness the power of this versatile spreadsheet environment:

Google Sheets Query: How to Use Group By