Learning to Sum Multiple Columns with the Google Sheets QUERY Function

Authored by

Mohammed looti

October 28, 2025

Harnessing the Power of the Google Sheets QUERY Function

The QUERY function in Google Sheets stands as one of the most sophisticated and powerful tools available for data manipulation and analysis within the spreadsheet environment. It grants users the ability to process data using a syntax highly analogous to **Structured Query Language** (SQL), moving far beyond simple cell referencing and basic arithmetic formulas. By mastering this function, analysts can perform complex filtering, sorting, aggregation, and restructuring of large **datasets** directly within the familiar Google Sheets interface, significantly improving reporting efficiency.

The primary utility of the QUERY function lies in its capability to dynamically select specific columns and rows based on defined criteria. However, real-world data often requires more than simple selection; frequently, numerical data is distributed across several columns, and a consolidated total is required for meaningful insight. For instance, tracking monthly sales figures or, as we will demonstrate, game scores, necessitates combining these values into a single summary column.

This guide specifically focuses on solving the common requirement of summing values from multiple distinct columns into one calculated output using the arithmetic capabilities inherent to the QUERY function. We will explore the exact syntax required to achieve this aggregation, ensuring your resulting report is both accurate and easily digestible, transforming fragmented numerical data into a single, cohesive metric.

Defining the Core Syntax for Multi-Column Summation

Calculating the sum of several columns within a QUERY function relies on exploiting the arithmetic addition operator (+) directly within the SELECT clause. Unlike traditional spreadsheet formulas where you might sum the range after extracting the data, the QUERY function allows you to define this computation as a new, virtual column during the extraction phase itself. This method is highly efficient as it processes the calculation internally before rendering the final output.

The fundamental structure for performing this calculation involves listing the columns you wish to include in your output, placing the addition operation between the columns intended for summation. The syntax below illustrates the simplest form, where we retrieve the value of Column A and simultaneously calculate the sum of Columns B, C, and D:

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

In this crucial example, the formula instructs Google Sheets to process the data range `A1:D8`. The SELECT clause then specifies two outputs: the values from **Column A** (likely an identifier or

category) and a dynamically computed column representing the sum of the corresponding values in **Columns B**, **C**, and **D** for every row. Furthermore, the final argument, `1`, is vital; it specifies that the range includes one header row, ensuring that the first row is correctly interpreted as column labels rather than numerical data to be summed. Misstating or omitting the header argument can introduce errors or lead to incorrect calculations, especially if the first row contains text.

Step-by-Step Practical Application: Aggregating Game Scores

To provide a clear demonstration of this methodology, let us apply the formula to a realistic scenario involving sports statistics. Imagine you are tracking the performance of several basketball teams, where each game's score is recorded in a separate column. Our objective is to generate a concise report that shows each team's name alongside their total combined score across all games.

Our source **dataset**, residing in the range A1:D8 of your Google Sheets, contains the following structure:

	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 obtain each team's name (Column A) and the sum of points scored in Game 1 (Column B), Game 2 (Column C), and Game 3 (Column D), we construct the necessary QUERY function. This

approach bypasses the need for auxiliary columns or complex array formulas, instead relying on the query engine to perform the required aggregation row-by-row:

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

By entering this formula into any unoccupied cell, such as F1, the function immediately processes the data within the specified range. The result is a dynamically generated table starting at that cell, displaying the team identifier and the consolidated total score. This streamlined process demonstrates the efficiency of using arithmetic operators within the query language itself for rapid data summation and reporting.

A10 fx =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					

Analyzing and Interpreting Aggregated Query Results

Upon successful execution of the query, the resulting output table will present two clear columns: the first containing the descriptive identifiers (Team Names from **Column A**) and the second

containing the calculated aggregate scores (the sum of **Columns B, C, and D**). This consolidated format allows for immediate, high-level analysis of the data without requiring the user to manually trace or calculate individual sums across rows.

Observing the output generated by our example, we can draw immediate and concrete conclusions regarding team performance. The aggregation task provides a clear performance metric, which is crucial for ranking or comparison. For instance, the resulting totals highlight the performance differences:

The **Mavs** lead with a grand total of **302** points scored across the three games.
The **Warriors** and the **Lakers** are tied, both accumulating a total of **293** points over the combined games.

This succinct presentation not only validates the accuracy of the summation syntax but also underscores how the QUERY function transforms raw, multi-column numerical data into actionable insights. By minimizing the visual noise of unnecessary columns, the report focuses the user's attention squarely on the key performance indicator (the total score), making data interpretation faster and less prone to error.

Enhancing Readability using the LABEL Clause

While the basic summation formula provides mathematically accurate results, the automatic header assigned to the calculated column by the QUERY function is typically the mathematical expression itself (e.g., "B+C+D"). This default label is often technical, cumbersome, and lacks the professional clarity expected in formal reports or shared Google Sheets. To significantly improve the usability and interpretation of your output, it is highly recommended to utilize the LABEL clause.

The LABEL clause is appended to the main query string and allows you to assign a descriptive, user-friendly name to any output column, including those that are dynamically computed. This ensures that anyone viewing the spreadsheet can instantly understand the context of the calculated values without needing to decipher the underlying arithmetic expression. By defining a custom label, we ensure the report maintains a high degree of readability and professionalism.

To implement a custom label for our summed points column, we modify the previous query by introducing the `label` keyword, specifying the original calculated column (`B+C+D`) and its desired new name ('Total Points'):

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

The addition of `label B+C+D 'Total Points'` ensures that the output column header is now explicitly clear, replacing the technical notation with a meaningful description. As demonstrated in

the following visual, this small syntactic change yields a substantial improvement in the overall presentation of the data, making the report immediately understandable to all stakeholders.

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					

Essential Considerations and Best Practices for Data Quality

While the `QUERY` function handles arithmetic operations proficiently, the success of summing multiple columns is fundamentally dependent on the quality and structure of the underlying data. Adhering to specific best practices ensures that your aggregated results are accurate and that potential runtime errors are mitigated before they occur.

A primary consideration involves **Data Types**. The addition operator (`+`) expects to perform operations on numerical values. If any cells within the columns designated for summation (B, C, or D in our example) contain non-numeric data types, such as text strings or dates formatted incorrectly, the query may return errors (e.g., `#VALUE!`) or, in some cases, silently ignore the non-numeric data, leading to an inaccurate total. It is therefore crucial to perform a data integrity check on your source **dataset** prior to running complex queries, ensuring all intended numerical fields are

uniformly formatted as numbers.

Handling **Empty Cells and Null Values** is another important point. In the context of the QUERY function's arithmetic operations, empty cells are conventionally treated as zero (0). This behavior is generally beneficial for summation tasks, as it means the absence of a value does not halt the calculation but simply contributes zero to the total. However, if your business logic requires explicit differentiation between a recorded zero and a missing data point, you must be aware that the `QUERY` function's summation treats them identically, potentially requiring source data preprocessing if strict differentiation is needed.

Finally, meticulous **Range Specification and Scalability** must be considered. Always ensure that the data range specified (e.g., `A1:D8`) precisely covers the rows and columns intended for processing. Incorrect range definition is a common source of errors, either by inadvertently excluding data rows or by including blank rows or irrelevant text, which can disrupt the calculation, particularly the crucial header row count. For advanced scenarios where summing dozens of adjacent columns is necessary, while the `B+C+D+...` syntax works, it becomes tedious. In such rare cases, alternative methods using `ARRAYFORMULA` combined with other functions might offer a more scalable solution, though direct addition remains the cleanest method for typical datasets.

Further Resources for Advanced Google Sheets Mastery

Mastery of the `QUERY` function extends far beyond simple summation. Its comprehensive structure allows for highly sophisticated data analysis, including grouping, complex filtering using the `WHERE` clause, and ordering of results. To fully leverage the analytical potential of Google Sheets, users are encouraged to explore the broader capabilities of the query language.

To deepen your understanding and tackle more complex data challenges, the following resources provide essential guidance on advanced operations within Google Sheets:

Google Sheets Query: How to Use Group By: This resource teaches how to aggregate rows based on shared criteria, allowing you to calculate sums, averages, or counts across categories, rather than just row-by-row.
Google Visualization API Query Language Reference: The official, definitive documentation for the syntax used by the QUERY function, essential for understanding every clause and function parameter.
Google Sheets Help Center: The central hub for official tutorials, troubleshooting steps, and comprehensive documentation covering all features and functionalities of the Google Sheets platform.

By integrating the methods demonstrated here with these advanced concepts, you can transform

your raw data into sophisticated, high-quality, and reliable reports efficiently and effectively.