

How to Sum Values in Google Sheets Using the QUERY Function

Authored by
stats writer

December 3, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Sum Values in Google Sheets Using the QUERY Function*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=104063>

Introduction to the SUM Function within Google Sheets QUERY

The QUERY function in Google Sheets is arguably the most powerful tool for data analysis and manipulation within the spreadsheet environment. It allows users to execute operations using a specialized version of the Structured Query Language (SQL), transforming raw data into meaningful insights. One of the most frequently employed operations within this framework is the use of the **SUM** function, which falls under the category of aggregate functions.

The primary purpose of the **SUM** function is straightforward: to calculate the total numerical value from a specified range or column. However, when paired with the robust capabilities of the **QUERY** function, its utility is amplified significantly. Instead of merely summing up an entire column, we can dynamically calculate totals based on complex criteria, filtering the data before aggregation occurs. This capability is essential for generating concise reports, calculating departmental expenditures, or summarizing sales metrics where only specific conditions must be met.

Understanding how to integrate **SUM** into your **QUERY** statements is key to moving beyond basic spreadsheet functionality. Unlike simple functions like `SUMIF` or `SUMIFS`, the **QUERY** function offers unparalleled flexibility in combining aggregation with powerful data selection and filtering logic. This article will serve as an expert guide, detailing the precise syntax and methodology required to deploy the **SUM** function effectively across various conditional scenarios in your Google Sheet documents. We will explore methods ranging from simple column aggregation to complex, multi-criteria summation.

Understanding the QUERY Function Syntax

Before diving into specific examples of summing data, it is crucial to establish a foundational understanding of the overall QUERY function structure. The standard syntax requires two primary arguments: the data range and the query string itself. The query string is where the SQL-like commands reside, defining what data to select, how to filter it, and which aggregate functions, such as **SUM**, to apply.

When using an aggregate function like **SUM()**, the structure of the SQL clause changes slightly compared to a simple data selection query. Typically, a query that uses **SUM()** does not include a standard `SELECT *` command, but rather specifies the aggregate operation directly, followed by the column identifier. If the intent is to sum an entire column, the syntax remains relatively simple. However, if the intent is to calculate sums per group (e.g., total sales per region), a `GROUP BY` clause must also be introduced, though we will focus primarily on filtering before summation in this guide.

The core concept relies on specifying the target column within the **SUM()** parentheses inside the

query string. For instance, if you want to sum column C, the command within the query string would begin: `select sum(C)`. This instructs the function to iterate over the data in the specified range and calculate the total of all numeric values found within the designated column. This powerful structure allows us to find the sum of values in rows that meet certain criteria, providing immediate, actionable data summaries.

You can use the **SUM()** function in a [Google Sheets](#) query to find the sum of values in rows that meet certain conditions, as demonstrated in the following three foundational methods.

Method 1: Aggregating All Data (Simple SUM)

The simplest application of the **SUM** function within a [QUERY function](#) involves calculating the total of a numeric column across the entirety of the specified data range. This method is analogous to using the basic `SUM()` function in Google Sheets, but it provides the result dynamically through the query interface, often serving as a preliminary step before more complex filtering is introduced.

To implement this, the query string only requires the `SELECT` clause followed immediately by the **SUM()** function applied to the target column. No `WHERE` clause is necessary, as all rows are implicitly included in the aggregation. This approach ensures that every numeric cell within the designated column contributes to the final calculated total.

The syntax below illustrates how to execute a simple summation over a range defined as A1:C13, targeting the values in Column C for aggregation. Note that the entire query string is enclosed in double quotes, and the data range is separated from the query string by a comma.

Syntax Example: Sum All Rows

```
=QUERY(A1:C13, "select sum(C)")
```

Method 2: Conditional Summation (Single Criterion)

The true power of integrating **SUM** with the [QUERY function](#) becomes apparent when introducing the `WHERE` clause. This allows us to filter the dataset based on a specific condition before the summation is performed. Instead of calculating the total of all values in Column C, we might only want to sum the values in Column C where Column B meets a certain criterion, such as being greater than 100.

This conditional aggregation is vital for data segmentation. For example, if Column B represents the number of items sold and Column C represents the revenue per transaction, using a `WHERE` clause allows us to calculate the total revenue only for transactions where the quantity sold

exceeded a particular threshold. This precision ensures that the resulting aggregate value is highly targeted and relevant to the analytical question being posed.

When constructing the query string, the `WHERE` clause must follow the `SELECT sum(C)` command. The condition specifies the filtering logic, including the column identifier, the comparison operator (e.g., `>`, `<`, `=`, `!=`), and the value being tested against. The example below demonstrates summing column C only for those rows where the corresponding value in column B is greater than 100.

Syntax Example: Sum Rows that Meet One Criteria

```
=QUERY(A1:C13, "select sum(C) where B>100")
```

Method 3: Advanced Conditional Summation (Multiple Criteria)

For more complex analytical needs, the **SUM** function can be combined with multiple conditions using Boolean logic operators: **AND** and **OR**. This allows the user to define highly specific subsets of data for aggregation, greatly increasing the granularity of the analysis performed within Google Sheets.

The **OR** operator calculates the sum if a row satisfies at least one of the specified conditions. For instance, you might want to sum the total sales (Column C) if the region is 'East' (Column A) **OR** if the transaction size is large (Column B > 500). This broadens the scope of inclusion, ensuring that data meeting either criterion contributes to the final total. When using text criteria, such as 'Mavs' in the example below, the text must be enclosed in single quotes within the double-quoted query string.

In contrast, the **AND** operator is restrictive, requiring that a row satisfies both (or all) specified conditions simultaneously before the value in the aggregation column (Column C) is included in the sum. Using **AND** allows for pinpoint accuracy in filtering. If you need the sum of sales (Column C) only for those transactions that occurred in the 'North' region (Column A) **AND** were recorded in the year 2023 (Column D), the **AND** operator ensures both criteria are met. This is critical for generating precise, intersectional reports.

Syntax Example: Sum Rows that Meet Multiple Criteria

```
=QUERY(A1:C13, "select sum(C) where A='Mavs' OR B>100")
```

```
=QUERY(A1:C13, "select sum(C) where A='Mavs' AND B>100")
```

The following expanded examples utilize sample data to clearly illustrate how each of these powerful summation methods is used in practice, providing visual context for the results generated by the respective **QUERY** formulas.

Practical Example 1: Summing All Rows

In this first practical scenario, imagine we have a dataset tracking team performance, where Column C represents the total points scored across a season. Our goal is to quickly determine the grand total of all points scored by every team recorded in the sheet. This is the most basic application of the **SUM** aggregate function within the QUERY function.

We utilize the formula `=QUERY(A1:C13, "select sum(C)")`. The range A1:C13 encompasses the dataset, and the query string specifies the single action: aggregate the values in Column C. The simplicity of this command belies its efficiency, as it processes the entire range instantly to return a single, summary value.

The result generated provides an immediate, unfiltered summary metric. In this specific case, the formula processes the data for all teams listed, resulting in a single figure that represents the collective scoring effort.

E1	fx	=QUERY(A1:C13, "select sum(B)")			
	A	B	C	D	E
1	Team	Points	Rebounds		sum Points
2	Mavs	96	30		1141
3	Nets	93	22		
4	Hawks	94	28		
5	Heat	94	25		
6	Magic	99	25		
7	Spurs	105	26		
8	Rockets	103	28		
9	Hornets	95	33		
10	Suns	93	31		
11	Bucks	90	30		
12	Warriors	88	36		
13	Lakers	91	24		
14					
15					
16					
17					
18					

After executing the query, we can clearly see that the teams scored a total of **1,141** points, illustrating the effectiveness of this straightforward aggregation method.

Practical Example 2: Summing Based on One Criterion

Moving to a more detailed analysis, we now introduce a single filtering condition using the `WHERE` clause. Suppose Column B tracks the number of rebounds achieved by each team, and we are specifically interested in the total points scored (Column C) only by those teams that exhibited a lower rebounding performance--specifically, teams with less than 30 rebounds.

The formula used for this targeted summation is `=QUERY(A1:C13, "select sum(C) where B < 30")`. Here, the condition `B < 30` acts as a mandatory filter. The **SUM** function only includes the points from Column C for rows where the corresponding value in Column B satisfies the inequality. This exclusion of high-rebounding teams ensures the final total is highly specific.

This method provides crucial insight into subgroups within the data. By isolating teams based on a performance metric (rebounds), we can analyze the scoring output of that specific demographic. This is a common requirement in data reporting, where only segments of the population or dataset need to be aggregated.

E1					
=QUERY(A1:C13, "select sum(B) where C<30")					
	A	B	C	D	E
1	Team	Points	Rebounds		sum Points
2	Mavs	96	30		679
3	Nets	93	22		
4	Hawks	94	28		
5	Heat	94	25		
6	Magic	99	25		
7	Spurs	105	26		
8	Rockets	103	28		
9	Hornets	95	33		
10	Suns	93	31		
11	Bucks	90	30		
12	Warriors	88	36		
13	Lakers	91	24		
14					
15					
16					
17					
18					
19					

Upon execution, the total points scored solely by teams who had less than 30 rebounds was calculated to be **679**. This figure is significantly lower than the overall total, highlighting the impact of the applied filter.

Practical Example 3: Summing Based on Multiple Criteria (OR Logic)

The final level of complexity involves combining two or more filtering criteria using the **OR** operator. In this example, we aim to calculate the total rebounds (Column B) for any team that meets one of two conditions: either the team name (Column A) is 'Mavs' OR the total points scored (Column C) is greater than 100. Note that here we are summing Column B instead of Column C, demonstrating the flexibility of the aggregate function application.

The formula employed is `=QUERY(A1:C13, "select sum(B) where A='Mavs' OR C>100")`. Because we use **OR**, a row's rebound value is included in the sum if the team name is 'Mavs', regardless of their points, OR if their points exceed 100, regardless of their team name. This inclusive logic ensures that a broad set of qualifying data contributes to the final total.

This technique is invaluable when running analyses that require aggregating data based on disjunctive conditions. For instance, calculating the total inventory value for items that are either located in Warehouse X OR have a unit price above \$50. The use of **OR** ensures maximum inclusion based on the defined criteria, making it suitable for high-level preliminary analysis or broad reporting requirements.

	A	B	C	D	E
E1					=QUERY(A1:C13, "select sum(C) where A='Mavs' or B>100")
1	Team	Points	Rebounds		sum Rebounds
2	Mavs	96	30		84
3	Nets	93	22		
4	Hawks	94	28		
5	Heat	94	25		
6	Magic	99	25		
7	Spurs	105	26		
8	Rockets	103	28		
9	Hornets	95	33		
10	Suns	93	31		
11	Bucks	90	30		
12	Warriors	88	36		
13	Lakers	91	24		
14					
15					
16					
17					
18					
19					

The sum of the rebounds among teams that meet either of these criteria (Team A is 'Mavs' OR Points C > 100) is determined to be **84**.

Conclusion and Related Google Sheets Operations

Mastering the use of the **SUM** function within the QUERY function provides Google Sheets users with an unparalleled ability to perform advanced, conditional aggregation. By leveraging the SQL-like syntax, particularly the `WHERE` clause combined with Boolean logic (AND/OR), users can move beyond simple spreadsheet totaling into sophisticated data mining and reporting.

The ability to dynamically sum data based on one or multiple criteria streamlines the reporting process, eliminating the need for complex, nested `IF` statements or relying on pivot tables for every aggregation requirement. Whether calculating the total budget spent across specific projects or summarizing performance metrics for filtered demographic groups, the **QUERY(SUM)** combination is a fundamental skill for advanced spreadsheet professionals.

While **SUM** is a powerful tool, it is just one of many aggregate functions available within the QUERY framework. Functions like `AVG` (average), `COUNT`, `MAX`, and `MIN` follow similar

implementation rules, allowing for a full suite of analytical operations. We encourage further exploration into these functions to maximize your data processing capabilities.

The following tutorials explain how to perform other common operations with Google Sheets queries, expanding your command over aggregate and relational functions:

[Google Sheets Query: How to Use Group By](#)

[Google Sheets Query: How to Use the COUNT Function](#)

ARABPSYCHOLOGY.COM