Google Sheets Query: Use the Label Clause

Authored by

Mohammed looti

November 2, 2025

The world of spreadsheet analysis relies heavily on efficient data extraction and presentation. Within Google Sheets, this capability is primarily driven by the immensely versatile QUERY function. This function allows users to execute complex data manipulation tasks using a language remarkably close to standard Structured Query Language (SQL). While filtering and aggregation are core uses, the presentation of the final output is equally crucial. Generating professional, clear, and instantly understandable reports hinges on utilizing the **label** clause.

The **label** clause serves as the bridge between raw data output and polished reporting. Its primary mechanism is providing precise control over the column headers returned by the query. Instead of relying on cryptic default identifiers--such as column letters (A, B, C) or vague source headers--the **label** clause permits the assignment of specific, user-friendly titles. This customization is not merely cosmetic; it is fundamental to ensuring that the query output is self-explanatory, reducing ambiguity, and significantly enhancing the overall accessibility of the data to any stakeholder, regardless of their familiarity with the underlying data structure.

The Power of the Google Sheets QUERY Function

The QUERY function is arguably the most sophisticated tool available in Google Sheets for handling large datasets. It is typically employed for complex operations, including filtering records based on multiple criteria, aggregating data using statistical methods, and precise sorting. When a query is executed, the function processes the data from the specified range and returns a new, structured dataset. By default, this resulting dataset retains the column headers from the original source range, whether those headers are explicit names or generic column letters.

While this default behavior is often acceptable for basic, internal data review, it creates immediate friction when preparing data for external communication or executive dashboards. Imagine a large dataset where columns are simply named "Col1," "Col2," or where a calculation results in a generic header like "sum B." These unhelpful identifiers force the viewer to constantly reference the original sheet or an accompanying data dictionary to understand the meaning of the figures. Effective data presentation demands immediate clarity, a requirement that the default QUERY output frequently fails to meet, particularly when dealing with complex joins or calculated fields.

This necessity for clear presentation is amplified when the query involves internal calculations. When using aggregate functions--such as `SUM()`, `AVG()`, or `COUNT()`--the QUERY language automatically generates a header that describes the operation (e.g., `sum Revenue`). Although technically accurate, these headers are often too verbose or too technical for a polished report. The only mechanism within the QUERY language designed to override these default or cryptic headers and assign meaningful, descriptive titles is the powerful **label** clause.

Why Header Customization Matters: Introducing the LABEL Clause

The primary and indispensable purpose of the **label** clause is to dramatically enhance the readability and professional quality of the query output. It transforms raw, technical results into polished summaries by allowing the user to dictate the exact terminology used for every column header. Instead of being constrained by cryptic column identifiers or auto-generated aggregate names, the user gains the freedom to assign descriptive names that align perfectly with the business context of the analysis. This level of granular customization is essential for complex reporting where metrics might be derived from multiple calculations.

Structuring the QUERY statement correctly is critical when implementing the **label** clause. According to the internal rules of the QUERY language, the **label** clause must always be placed at the very end of the query string. It follows all other operational clauses, including `SELECT`, `WHERE`, `GROUP BY`, `PIVOT`, and `ORDER BY`. This specific placement ensures that the function first determines the final set of columns to be displayed and then applies the designated custom names to those columns just before the results are finalized and displayed in the output cell.

The syntax directs the QUERY function to assign new, customized names to the selected columns in the resulting output table without altering the underlying structure or data of the source spreadsheet. This separation of presentation logic from data logic is a core principle of effective data manipulation and ensures data integrity while maximizing report clarity. Mastery of this clause is foundational for anyone producing professional data summaries within Google Sheets.

Mastering the Syntax: Labeling Individual Columns

The most straightforward application of the **label** clause involves renaming a single column. This highly efficient approach is perfect for scenarios where only one header requires clarification or modification. The syntax is concise and requires two key elements: the column identifier and the desired new label. The column identifier can be the original column letter (A, B, C) or the column name if the query is selecting data by header name. Crucially, the new label itself must be enclosed in single quotes (`' '`) to distinguish it from other keywords in the query string.

When constructing the query string, the general format is `LABEL 'new_label'`. This instruction tells the QUERY engine, "Take the output column corresponding to this identifier, and in the final presentation, replace its existing header with this new string." It's important to remember that if the column being labeled is not included in the `SELECT` statement, the label command will have no effect, reinforcing the requirement that the **label** clause acts exclusively upon the selected output fields defined earlier in the query.

Consider the following syntax for renaming a single column within a comprehensive query. This example demonstrates how to select all data from a range while specifically targeting Column A for

a custom header:

=QUERY(A1:C13, "select * label A 'Column A Data Identifier'")

In this specific formula, we instruct the QUERY function to select all columns (`select *`) from the range A1:C13. The instruction `label A 'Column A Data Identifier'` then takes precedence for the header of the first column. Columns B and C, which are also selected, automatically retain their original headers because they were not explicitly referenced within the **label** clause. This targeted application highlights the precision and control offered by this clause, allowing for necessary renaming without affecting other selected fields.

Advanced Labeling: Handling Multiple Columns and Calculated Fields

For creating detailed and comprehensive reports, it is almost always necessary to customize the headers of several columns simultaneously. Fortunately, the syntax for labeling multiple columns is an intuitive and scalable extension of the single-column method. To label more than one field, you simply append subsequent column-label pairs to the **label** clause, ensuring they are separated by a comma. This structure allows vast customization of the report headers within a single, streamlined query statement.

The generalized multi-column format is `LABEL 'label1', 'label2', 'label3'`, and so on. This ability to execute multiple renames at once significantly streamlines the process of preparing complex data structures for professional presentation, making the overall data management workflow more efficient and less prone to manual header adjustments post-query execution.

Furthermore, the **label** clause is indispensable when dealing with calculated fields, especially those generated by aggregate functions. Unlike simple column selections (A, B, C), aggregate functions like `SUM(B)` or `AVG(D)` return a header that reflects the operation (e.g., `sum B`). To rename this calculated field, you must reference the calculated field itself in the label clause. For example, if your `SELECT` statement includes `SUM(B)`, the corresponding label clause would be `label sum(B) 'Total Sales Revenue'`, providing immediate and contextual clarity.

Observe the following structure used to rename two foundational columns within a query:

=QUERY(A1:C13, "select A, B, C label A 'Primary Identifier', B 'Secondary Metric'")

In this formula, we select three columns but apply two distinct labels: Column A is renamed to 'Primary Identifier', and Column B is renamed to 'Secondary Metric'. Column C retains its original header. This example confirms that the comma serves as the necessary separator between each renaming operation within the **label** clause. Utilizing this technique allows spreadsheet analysts to

perform sophisticated data manipulation and present the results with maximum clarity, a feature often lacking in raw SQL-like query outputs.

A Practical Demonstration: Enhancing Data Readability

To fully appreciate the practical impact of the **label** clause, let us examine a real-world scenario involving a sports statistics dataset. Assume we have three critical fields: Team Name, Points Scored, and Rebounds Achieved. Our objective is to extract these three columns using the QUERY function but ensure that the resulting headers are optimized for a public-facing report, moving beyond the default, often vague source names.

In our first step, we aim to refine only the 'Team' column header, renaming it to the more formal 'Team Name' to improve descriptive accuracy and formal tone. The other columns, 'Points' and 'Rebounds,' will temporarily retain their existing names, demonstrating the clause's ability to selectively target specific fields for modification.

The following formula selects the necessary columns (Team, Points, and Rebounds) and then specifically instructs the QUERY engine to apply the custom label 'Team Name' only to the designated column, leaving the others untouched:

E1 fx =QUERY(A1:C13, "select * label A 'Team Name'")

	A	B	C	D	E	F	G
1	**Team**	**Points**	**Rebounds**		Team Name	Points	Rebounds
2	Mavs	96	30		Mavs	96	30
3	Nets	93	22		Nets	93	22
4	Hawks	94	28		Hawks	94	28
5	Heat	94	25		Heat	94	25
6	Magic	99	25		Magic	99	25
7	Spurs	105	26		Spurs	105	26
8	Rockets	103	28		Rockets	103	28
9	Hornets	95	33		Hornets	95	33
10	Suns	93	31		Suns	93	31
11	Bucks	90	30		Bucks	90	30
12	Warriors	88	36		Warriors	88	36
13	Lakers	91	24		Lakers	91	24
14							
15							
16							
17							
18							

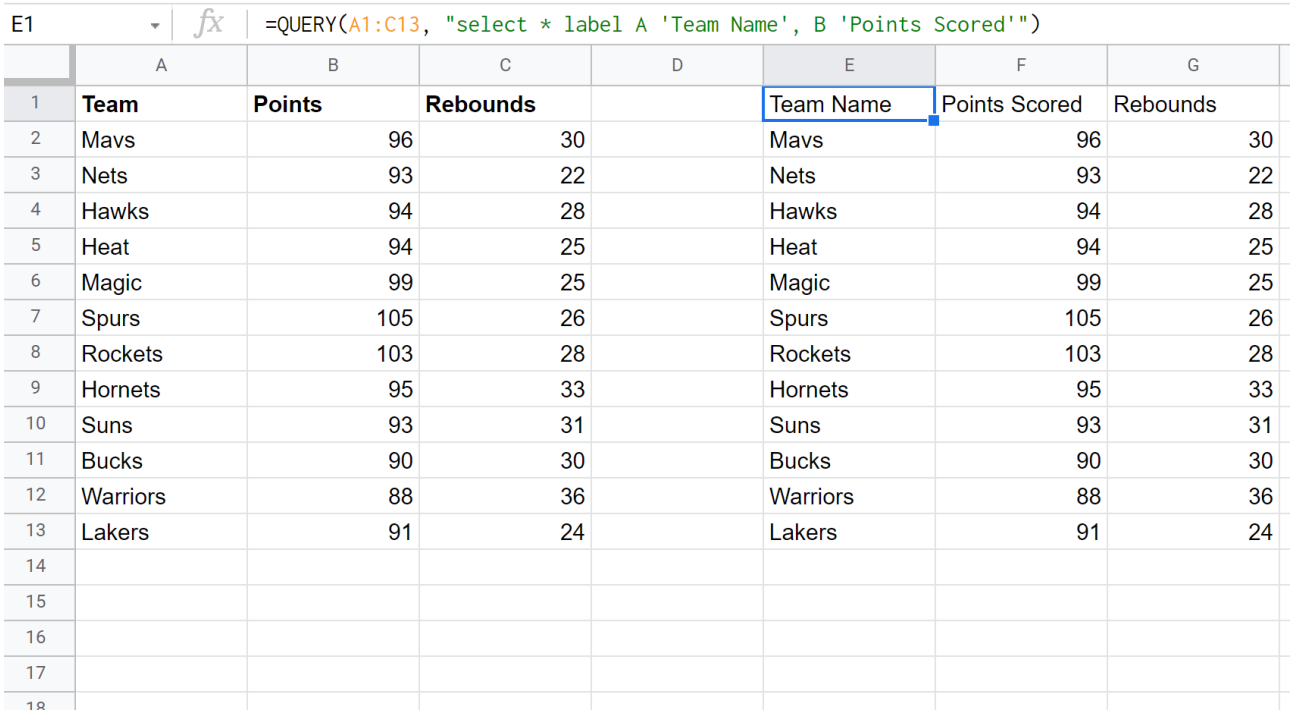
The immediate output demonstrates a clear improvement in data clarity. The original header for the team column has been successfully replaced by 'Team Name,' which is significantly more

descriptive for any user reviewing the results. This targeted modification confirms that the **label** operation is precise and only affects those columns explicitly referenced within the clause, allowing for granular control over the final report schema.

Enhancing Output Clarity: Multi-Column Refinement

Building upon the foundation of the single-column example, professional reporting often necessitates the renaming of several metrics to standardize nomenclature or provide greater contextual detail. In this subsequent step, we will apply the multi-column labeling technique to rename both the 'Team' column and the 'Points' column simultaneously, further enhancing the report's professional quality and ensuring all key metrics are clearly defined.

Our objective is to label the 'Team' column as 'Team Name' and the 'Points' column as 'Points Scored'. We achieve this by listing both renaming instructions within the single **label** clause, separated by a comma, as demonstrated in the previous syntax overview:



	A	B	C	D	E	F	G
1	**Team**	**Points**	**Rebounds**		Team Name	Points Scored	Rebounds
2	Mavs	96	30		Mavs	96	30
3	Nets	93	22		Nets	93	22
4	Hawks	94	28		Hawks	94	28
5	Heat	94	25		Heat	94	25
6	Magic	99	25		Magic	99	25
7	Spurs	105	26		Spurs	105	26
8	Rockets	103	28		Rockets	103	28
9	Hornets	95	33		Hornets	95	33
10	Suns	93	31		Suns	93	31
11	Bucks	90	30		Bucks	90	30
12	Warriors	88	36		Warriors	88	36
13	Lakers	91	24		Lakers	91	24
14							
15							
16							
17							

As the visual output confirms, we successfully customized two column headers using a single, efficient **label** clause. The syntax remains highly manageable: list the column identifier, followed by the new label in single quotes, and separate each pair with a comma. This methodology is completely scalable, allowing users in Google Sheets to rename any necessary number of columns, ensuring optimal report readability and strict adherence to organizational reporting standards. This flexibility makes the **label** clause an indispensable component of advanced Google Sheets query writing.

Best Practices for Professional Query Output

While the **label** clause is syntactically simple, maximizing its effectiveness requires adherence to several best practices focused on clarity, consistency, and handling complex calculations. Integrating these principles into your data reporting workflow will ensure that your query results are not only accurate but also immediately accessible and professional.

To maximize the effectiveness of the **label** clause in your Google Sheets queries, consider the following guidelines:

Prioritize Clarity Over Brevity: While analysts often favor short names, the chosen label must be descriptive enough to stand alone without requiring external context or explanation. A label like "Rev YTD" is better than "R," but "Revenue Year-to-Date" is best for external audiences or those unfamiliar with internal abbreviations.
Ensure Naming Consistency: If generating a series of related reports or dashboards, maintain a rigorous standard for naming conventions across all of them. Consistency ensures that users familiar with one report can instantly understand the terminology used in another (e.g., always use "Customer ID" instead of alternating between "Customer ID" and "Client Reference").
Mandatory Use for Calculated Fields: Always utilize the **label** clause when your `SELECT` statement includes aggregate functions (like `SUM`, `AVG`, or `COUNT`). Relying on default headers (e.g., `sum ColB`) leads to confusion and undermines the report's professional credibility, particularly in complex data manipulation scenarios.
Use Single Quotes Carefully: Ensure that the custom label text is strictly enclosed in single quotes. Using double quotes or no quotes will result in a parsing error within the query string, as the double quotes delineate the entire query string itself.

Mastering the **label** clause is a simple technique that yields disproportionately large benefits in terms of report quality and accessibility. By controlling precisely how your output headers appear, you fundamentally improve the robustness of your data analysis and enhance the professional presentation of your findings.

The following tutorials explain how to perform other common operations with Google Sheets queries:

Google Sheets Query: How to Use Group By