Google Sheets Query: Insert Blank Columns in Output

Authored by

Mohammed looti

October 27, 2025

In the realm of spreadsheet management, particularly when utilizing Google Sheets, the presentation of data is often as critical as the data itself. A well-organized output significantly enhances readability and aesthetic appeal, making complex information accessible to end-users. While standard formatting options exist, advanced users frequently need precise control over the layout generated by computational functions. This article addresses a sophisticated technique for data manipulation: the direct insertion of blank, visually separating columns within the result set of a QUERY function.

This guide provides a comprehensive breakdown of the necessary syntax and logical mechanisms required to seamlessly integrate empty columns into your **Google Sheets Query** results. We will explore how leveraging specific components of the query language allows you to create highly customized, visually optimized data views. This method ensures that the source data remains untouched, offering a non-destructive way to refine the output layout, which is invaluable for dynamic reporting and dashboard creation.

By mastering this subtle but powerful technique, you will elevate your spreadsheet proficiency, enabling you to deliver polished, professional data presentations that are structured exactly to your specifications.

Deconstructing the Power of the Google Sheets QUERY Function

The QUERY function is widely recognized as the most versatile and potent tool within **Google Sheets**, offering capabilities analogous to the structured query language, SQL. This function is essential for users who need to perform complex aggregation, filtering, and transformation operations, allowing data to be dynamically selected and rearranged from one range into another.

At its operational core, the **QUERY function** requires two primary components: the **data range** (specifying the source data) and the **query string**. The query string, written in a specific syntax, dictates the precise logic for data extraction--including which columns to display, which rows to include (via `WHERE` clauses), and how to sort the output (using `ORDER BY`). For the specific goal of inserting blank columns, our focus must narrow to two critical clauses: the SELECT clause, which determines the output columns, and the crucial LABEL clause, which manages headers.

A deep understanding of the **QUERY function** is foundational for advanced data analysis. It empowers users to move beyond simple arithmetic and static formulas, enabling the creation of dynamic reports that update automatically as the underlying source data changes, thereby streamlining complex data workflows in **Google Sheets**.

Implementing a Single Blank Column using Empty String Literals

To effectively introduce a blank column into the output of your query, we utilize a clever

implementation within the SELECT clause. This technique involves treating an empty string as a selectable column element. By including a literal empty string where the blank column should appear, the query interprets this string as a column containing only null values, thus creating the visual separation required.

Observe the fundamental syntax required to select Column A, insert a blank column, and subsequently select Column B from a specified dataset:

=QUERY(A1:C12, "SELECT A, ' ', B LABEL ' ' """)

In this formula, the segment `SELECT A, ' ', B` instructs the query to retrieve data from column A, followed by the representation of an empty column (denoted by the literal string `**' '**`), and finally the data from column B. If left alone, this empty column would inherit a default system header (e.g., "Col1"). To prevent this, we must ensure the column is completely blank, including the header row.

This is where the LABEL clause becomes indispensable. The subsequent segment, `**LABEL ' ' ""**`, explicitly assigns an empty string `""` as the header for the column represented by the literal string `' '`. This meticulous step ensures a truly clean, blank column--devoid of both content and a header--resulting in a highly professional and segmented presentation of your data output.

Practical Demonstration and Syntax Application

To solidify this concept, let us apply it using a concrete example. Consider a typical dataset that tracks player statistics, where Column A contains the Player Name, Column B contains Points, and Column C contains Rebounds.

	A	B	C	D
1	**Team**	**Points**	**Assists**	
2	Mavericks	22	4	
3	Hawks	24	10	
4	Kings	30	12	
5	Hawks	18	8	
6	Lakers	14	8	
7	Lakers	15	7	
8	Spurs	22	12	
9	Hawks	20	7	
10	Warriors	15	5	
11	Spurs	17	10	
12	Cavs	24	11	
13				
14				
15				
16				
17				

If the requirement is to display the Player Name (Column A) and their Points (Column B), but with a prominent visual gap between them for enhanced readability, the syntax introduced earlier is the perfect solution. You would input the following QUERY function into an output cell, such as E1:

=QUERY(A1:C12, "SELECT A, ' ', B LABEL ' ' """)

Upon execution, Google Sheets processes the query, extracting the specified data columns while simultaneously generating the blank space instructed by the empty string literal. The resultant table effectively separates the player names and scores, offering a clear visual distinction, as demonstrated in the output screenshot below:

A15 | fx | =QUERY(A1:C12, "SELECT A, ' ', B LABEL ' ' ''")

	A	B	C	D	E
1	**Team**	**Points**	**Assists**		
2	Mavericks	22	4		
3	Hawks	24	10		
4	Kings	30	12		
5	Hawks	18	8		
6	Lakers	14	8		
7	Lakers	15	7		
8	Spurs	22	12		
9	Hawks	20	7		
10	Warriors	15	5		
11	Spurs	17	10		
12	Cavs	24	11		
13					
14					
15	Team		Points		
16	Mavericks		22		
17	Hawks		24		
18	Kings		30		
19	Hawks		18		
20	Lakers		14		
21	Lakers		15		
22	Spurs		22		
23	Hawks		20		
24	Warriors		15		
25	Spurs		17		
26	Cavs		24		
27					
28					

This successful outcome confirms that the values from Column A are returned, followed by a completely empty column, and then the values from Column B. This visual partitioning significantly enhances the data layout, making the information far easier to scan and interpret, all without modifying the integrity or structure of the original data source.

Scaling Up: Inserting Multiple Blank Columns Simultaneously

The methodology for inserting a single blank column is easily scalable to incorporate multiple blank columns within your query output. The core principle remains consistent: for every desired blank

column, a separate empty string literal must be added to the SELECT clause, and each must be paired with a corresponding empty entry in the LABEL clause to suppress the header.

If the requirement is to insert two distinct blank columns between, for example, column A and column B, the query string must be carefully adjusted to account for two separate placeholders:

=QUERY(A1:C12, "SELECT A, ' ', ' ', B LABEL ' ' '', ' ' '''')

Note the introduction of a second placeholder, `' '` (using two spaces inside the quotes), within the `SELECT` statement, along with its matching entry in the `LABEL` clause. It is absolutely crucial for the string literals used in the `SELECT` clause to be unique (e.g., `' '` and `' '`) so that the `LABEL` clause can reference and suppress the headers for each blank column individually. Although the resulting header is always the empty string `''`, the placeholder itself must be unique when dealing with multiple blank columns.

Applying this revised formula to your Google Sheets dataset will yield an output clearly showing two distinct blank columns separating the selected data, as illustrated below:

A15 | =QUERY(A1:C12, "SELECT A, ' ', ' ', B LABEL ' ' '', ' ' ''")

	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavericks	22	4			
3	Hawks	24	10			
4	Kings	30	12			
5	Hawks	18	8			
6	Lakers	14	8			
7	Lakers	15	7			
8	Spurs	22	12			
9	Hawks	20	7			
10	Warriors	15	5			
11	Spurs	17	10			
12	Cavs	24	11			
13						
14						
15	Team			Points		
16	Mavericks			22		
17	Hawks			24		
18	Kings			30		
19	Hawks			18		
20	Lakers			14		
21	Lakers			15		
22	Spurs			22		
23	Hawks			20		
24	Warriors			15		
25	Spurs			17		
26	Cavs			24		
27						

This advanced technique grants granular and precise control over the visual spacing and organization of your query outputs, allowing for highly customized, presentation-ready data arrangements tailored precisely to complex reporting or analytical demands.

Key Considerations and Best Practices for Clean Output

When deploying blank columns using the **Google Sheets** SELECT clause, the role of the LABEL clause cannot be overstated. Without its implementation, the blank columns would inherit default headers such as 'Col1', 'Col2', or even the literal string used (e.g., ' ' or ' '). Such unintended headers fundamentally undermine the objective of creating a visually clean, blank separation space.

It is imperative to ensure that every blank column placeholder (such as `' '`, `' '`, or any other unique string literal used for spacing) in your `SELECT` statement has a direct, corresponding entry in the `LABEL` clause, even though the mapping is always to an empty string `''`. This disciplined approach guarantees that the resulting query output is professional, tidy, and completely free from any unwanted header information.

While this method is exceptionally powerful for presentation and visual separation within query results, it is important to remember that it is purely a formatting technique. The underlying data source remains structurally unchanged. For instances requiring more comprehensive data manipulation or permanent structural alterations to the source, alternative functions or App Scripting may be necessary. However, for dynamic, non-destructive adjustments to the visual layout of queried data, this approach remains the most efficient and robust solution.

Conclusion: Mastering Data Layout in Google Sheets

The capability to strategically insert blank columns directly within the output of a Google Sheets query provides a sophisticated mechanism for controlling the visual presentation of your dataset. By carefully applying the combination of empty string literals within the SELECT clause and the precise header suppression managed by the LABEL clause, users can achieve highly readable and aesthetically superior data arrangements.

This technique proves invaluable for generating reports, constructing dashboards, or creating any data display environment where clear separation and meticulous organization are paramount. It grants the user the ability to dynamically refine data presentation without resorting to manual column insertions or complex post-query formatting rules.

We encourage experimentation with this advanced feature. By mastering such sophisticated aspects of the SQL-like query language, you can unlock significant gains in efficiency and professionalism across all your data manipulation tasks within **Google Sheets**.

Additional Resources for Google Sheets Mastery

To further refine your proficiency in **Google Sheets** and its powerful query capabilities, consider exploring supplementary tutorials focused on related advanced operations.

Advanced filtering and conditional clause usage (e.g., `WHERE` and `GROUP BY`).

Techniques for joining multiple data ranges using array literals within the `QUERY` input.

Comprehensive guides on data type handling and formatting within the query language.

Continuously learning and applying new functions and strategies will dramatically boost your

productivity and capability in managing, analyzing, and presenting complex spreadsheet data effectively.