Learning to Query Data Across Multiple Ranges in Google Sheets

Authored by

Mohammed looti

March 7, 2026

Mastering Data Consolidation in Google Sheets with the QUERY Function

Google Sheets is recognized globally as a powerful, cloud-based spreadsheet solution, essential for data management, analysis, and visualization. Its collaborative nature and extensive library of functions make it an indispensable tool for complex projects. Central to its advanced capabilities is the `QUERY` function. This function provides a robust interface for manipulating and extracting structured data using a syntax highly reminiscent of the standard database query language (SQL). The true power of `QUERY` shines when you need to unify information that is scattered across multiple sheets or even different workbooks within your organizational structure.

A frequent challenge for data analysts is dealing with crucial information that isn't contained within a single, neat range but is instead distributed across various segments of a spreadsheet file. Consolidating this fragmented data manually is tedious, inefficient, and highly susceptible to error. Fortunately, the `QUERY` function offers an elegant and scalable solution. It allows users to seamlessly combine, filter, and restructure data from these disparate sources into one comprehensive, cohesive view. This functionality is absolutely critical for generating unified reports, performing advanced analysis, and ensuring high levels of data integrity across large-scale projects.

This guide is designed to provide a deep dive into the syntax and practical applications necessary for using `QUERY` to consolidate information from multiple ranges within Google Sheets. We will meticulously explore the two primary methods: the crucial technique of vertically stacking data to append rows, and the method for horizontally combining data to merge columns. By mastering the use of the **array literal** syntax for these purposes, you will dramatically enhance your efficiency in managing complex datasets, transforming disorganized information into clear, actionable insights.

Vertical Stacking: The Array Literal for Appending Rows

One of the most frequent requirements in data management is the need to combine datasets by placing one directly beneath the other--a process known as vertical concatenation or stacking. This technique is invaluable when you have identical data structures recorded in different locations, such as monthly sales logs, regional performance metrics, or, as we will demonstrate, team statistics stored on separate sheets. The `QUERY` function handles this by utilizing the **array literal** syntax, which allows you to define all the source ranges intended for stacking.

The fundamental structure for vertically stacking multiple ranges involves enclosing all references within curly braces `{}`--the **array literal**--and using a semi-colon `;` as the essential separator between the ranges. The semi-colon is the specific operator that instructs Google Sheets to concatenate the specified data vertically. This means the rows of the second range are placed

immediately below the last row of the first range, and so forth, creating a single, extended column of data.

Here is the core syntax you must employ to query and vertically stack data from multiple ranges within a single `QUERY` command:

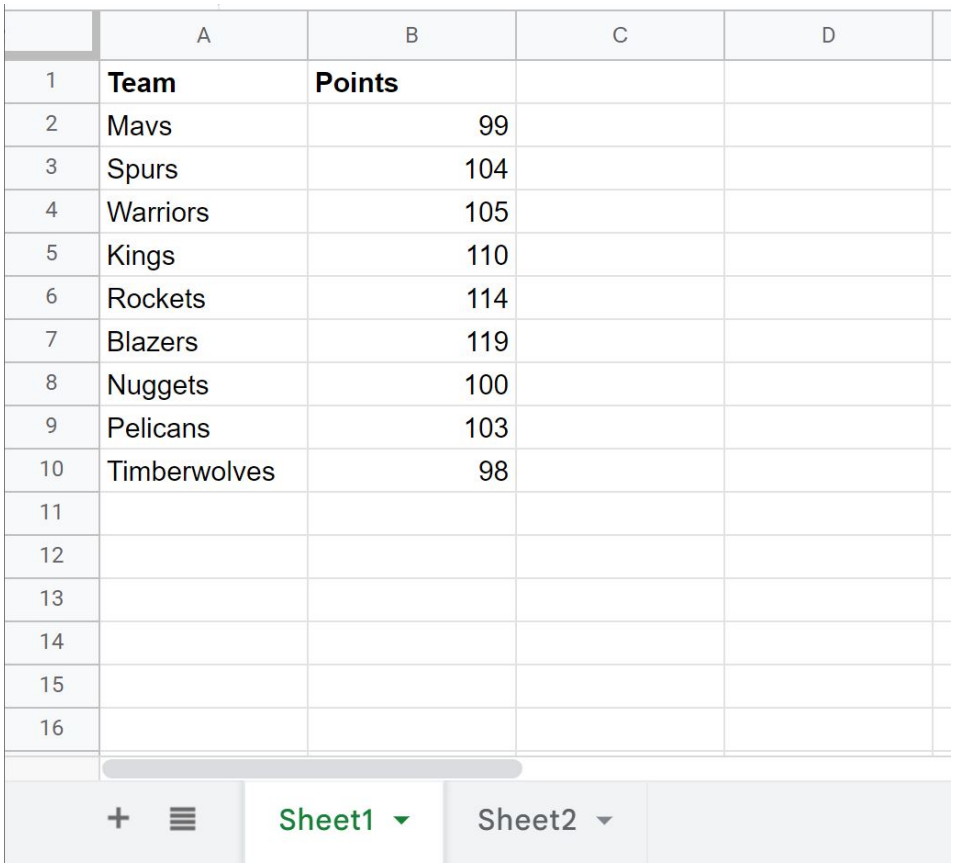
=QUERY({Sheet1!A1:B10; Sheet2!A2:B5})

In this powerful example, the `QUERY` function is set to retrieve data from two distinct locations. It first pulls the cells within the range **A1:B10** from **Sheet1**, and then meticulously stacks the cells from the range **A2:B5** from **Sheet2** directly underneath. For this method to work flawlessly, it is critical that the columns being stacked have compatible data types and maintain a consistent column order across all source ranges. This structure provides the foundation for efficient and reliable data aggregation across distributed datasets.

Practical Demonstration of Vertical Stacking

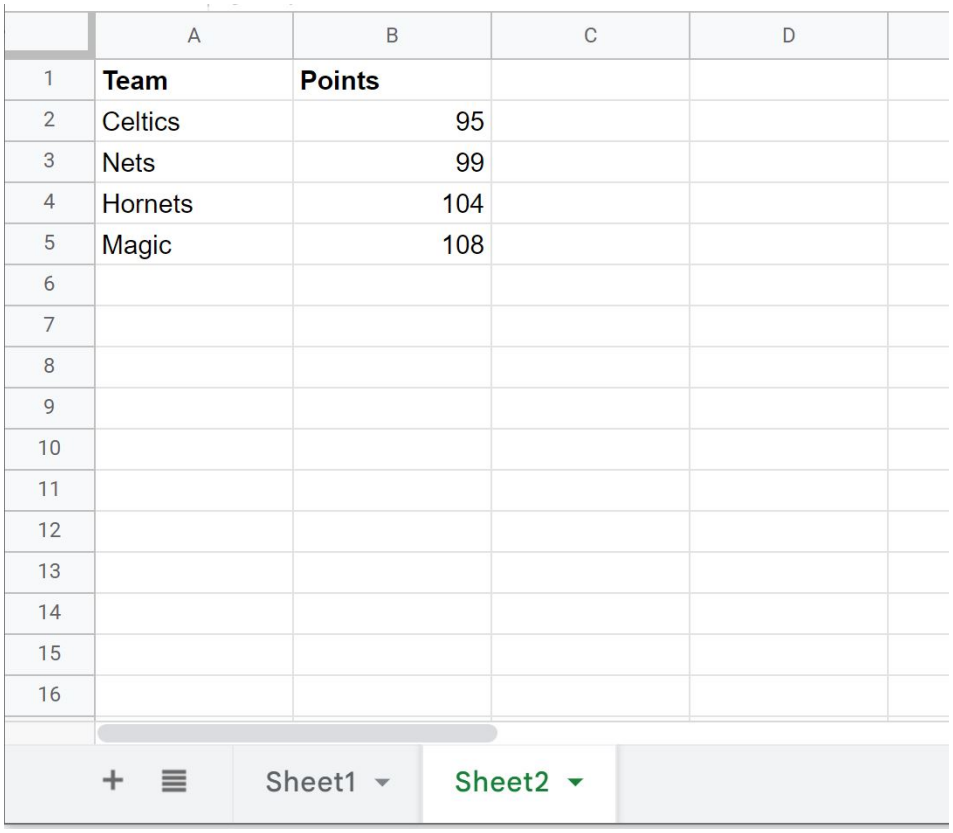
To truly grasp the simplicity and efficiency of using the `QUERY` function for vertical stacking, let us examine a concrete scenario. Imagine you are compiling statistics for various basketball teams, where initial data points are recorded in **Sheet1**, and subsequent or updated data resides in **Sheet2**. Our objective is to consolidate all this information into a single, unified table for comprehensive analysis without manual copy-pasting.

In our Google Sheets workbook, we have the following data entries in our source sheets, detailing team names and their respective scores or points:



	A	B	C	D
1	**Team**	**Points**		
2	Mavs	99		
3	Spurs	104		
4	Warriors	105		
5	Kings	110		
6	Rockets	114		
7	Blazers	119		
8	Nuggets	100		
9	Pelicans	103		
10	Timberwolves	98		
11				
12				
13				
14				
15				
16				

Sheet1 Sheet2



	A	B	C	D
1	**Team**	**Points**		
2	Celtics	95		
3	Nets	99		
4	Hornets	104		
5	Magic	108		
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

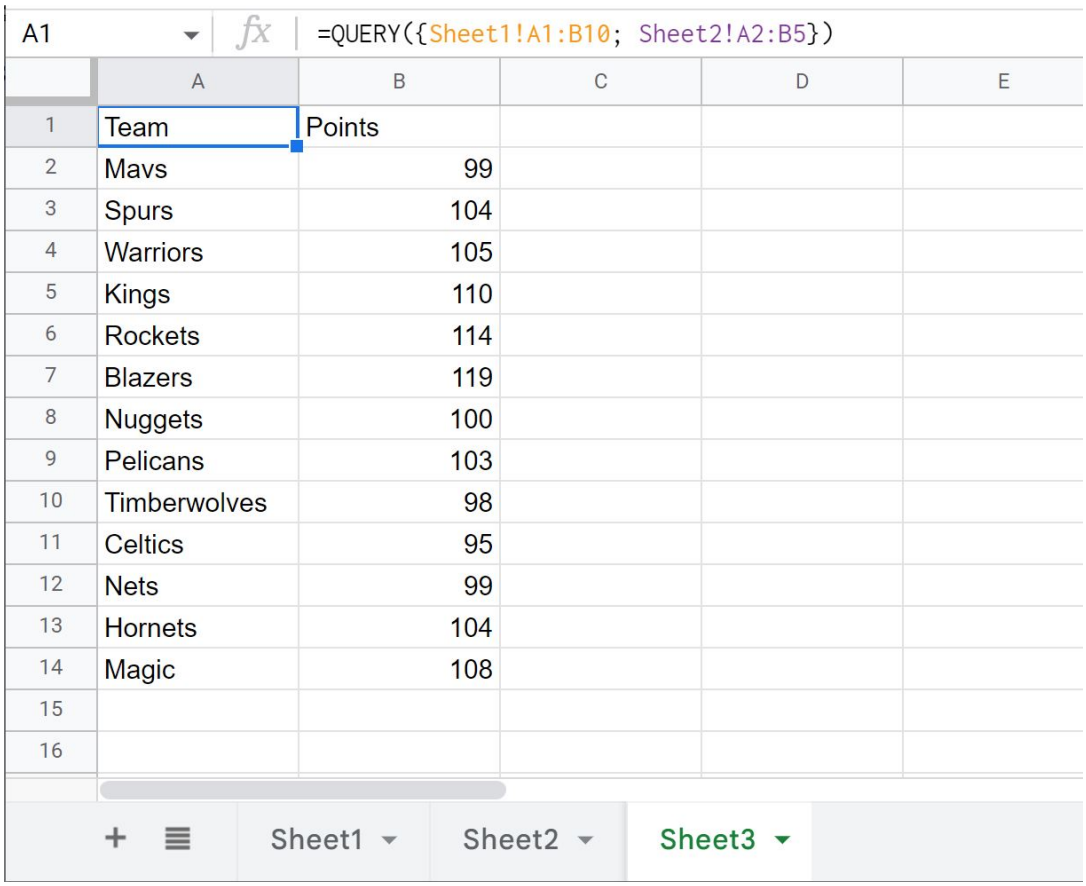
Sheet1 Sheet2

As evident, both sheets contain structurally similar data. To merge these separate datasets, we leverage the QUERY function alongside its **array literal** syntax. The goal is to combine all rows from **Sheet1** and append all valid data rows from **Sheet2** directly below them. We designate a new sheet, **Sheet3**, as the output destination for this consolidated view.

We achieve this consolidation by entering the following formula into the first cell (e.g., A1) of our output sheet, **Sheet3**. Notice that we intentionally exclude the header row (row 1) from **Sheet2** to prevent duplication, as the QUERY function automatically inherits headers from the first range specified:

=QUERY({Sheet1!A1:B10; Sheet2!A2:B5})

The result of this single formula is a continuous dataset that includes all team information from both sources, elegantly presented in **Sheet3**. The output demonstrates how the rows originating from **Sheet1** are seamlessly stacked directly on top of the rows extracted from **Sheet2**. This method is exceptionally powerful for creating unified master lists and reports from fragmented sources, significantly enhancing both readability and data integrity.



	A	B	C	D	E
1	Team	Points			
2	Mavs	99			
3	Spurs	104			
4	Warriors	105			
5	Kings	110			
6	Rockets	114			
7	Blazers	119			
8	Nuggets	100			
9	Pelicans	103			
10	Timberwolves	98			
11	Celtics	95			
12	Nets	99			
13	Hornets	104			
14	Magic	108			
15					
16					

Horizontal Merging: Combining Data Side-by-Side

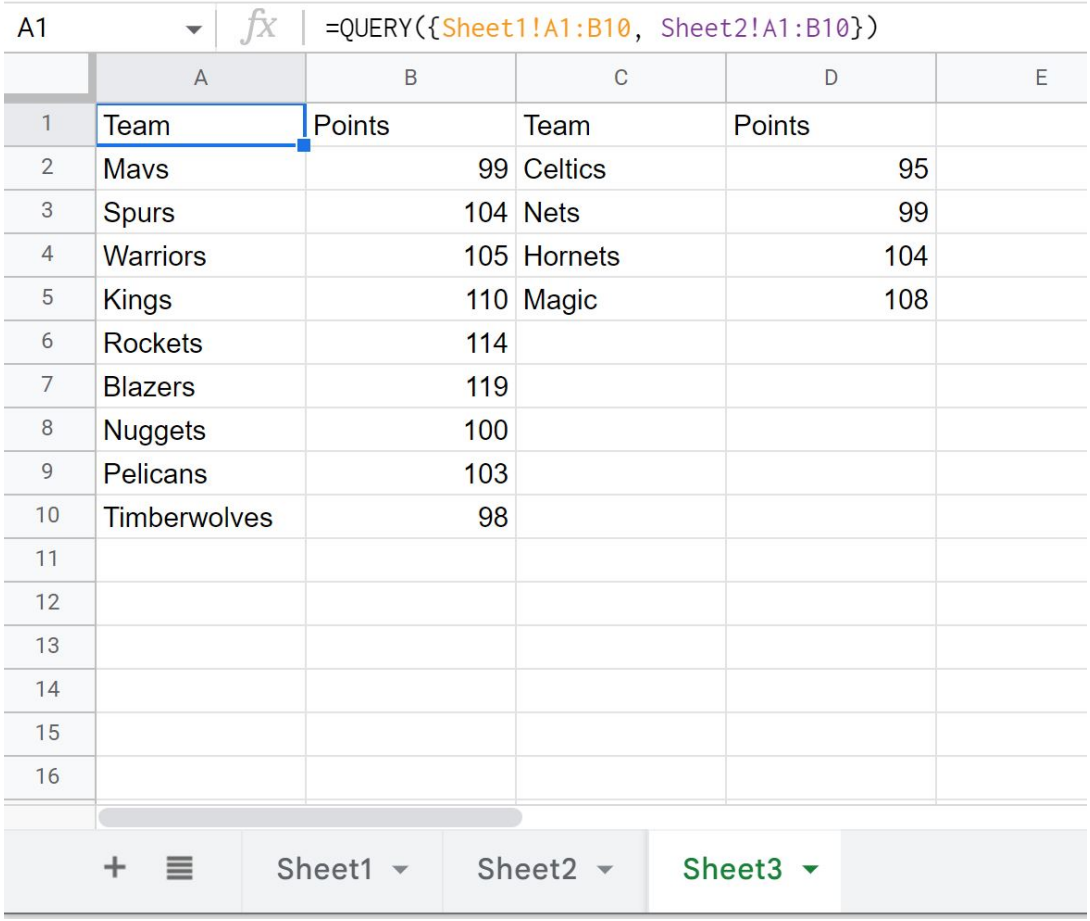
While vertical stacking is essential for appending rows, many analytical scenarios require combining data horizontally--placing ranges side-by-side to merge columns. This approach is highly effective when you have related, but separate, attributes about the same entities (like demographic information in one sheet and performance metrics in another) and you need to view all these attributes on a single output row. Instead of extending the number of rows, horizontal merging extends the number of columns.

To execute this horizontal concatenation within the `QUERY` function's **array literal**, you must substitute the vertical stacking separator, the semi-colon `;`, with a comma `,`. The comma serves as the explicit signal to Google Sheets that the specified ranges should be concatenated horizontally, effectively creating new columns appended to the right of the preceding range.

For instance, if you wish for the columns from **Sheet2** to appear immediately adjacent to the columns from **Sheet1**, expanding the width of your dataset, you would modify the formula to use the comma separator within the **array literal**:

=QUERY({Sheet1!A1:B10, Sheet2!A1:B10})

In this revised formula, the data from **Sheet1!A1:B10** establishes the initial columns of your output. The data from **Sheet2!A1:B10** is then appended directly as subsequent columns to the right. A critical requirement for successful horizontal concatenation is that **both ranges must have the exact same number of rows**. If the row counts differ, the `QUERY` function will automatically pad the shorter range with empty cells to match the longest range, which almost invariably leads to misaligned and unusable data. Therefore, meticulous preparation of your source ranges is paramount for accurate horizontal merging. The visual result of this side-by-side combination is demonstrated below:



	A	B	C	D	E
1	Team	Points	Team	Points	
2	Mavs	99	Celtics	95	
3	Spurs	104	Nets	99	
4	Warriors	105	Hornets	104	
5	Kings	110	Magic	108	
6	Rockets	114			
7	Blazers	119			
8	Nuggets	100			
9	Pelicans	103			
10	Timberwolves	98			
11					
12					
13					
14					
15					
16					

Essential Best Practices for Multi-Range Queries

While combining multiple ranges using the `QUERY` function is an incredibly versatile technique, several best practices must be observed to guarantee the accuracy, efficiency, and long-term robustness of your Google Sheets solutions. Adhering to these guidelines helps prevent common pitfalls associated with complex data consolidation.

Header Management for Vertical Stacking: When using the semi-colon for vertical stacking, the `QUERY` function typically assumes that the column headers are present only in the first range of the **array literal**. Data from subsequent ranges is treated as raw data rows. If your second range also includes headers, these will appear as data in your final consolidated output. To manage this correctly, always adjust the subsequent range references to exclude headers (e.g., use `Sheet2!A2:B5` if row 1 contains headers). You also have the option to use the `label` clause within `QUERY` to explicitly redefine or customize the output headers.

Data Type Strictness: For successful vertical stacking, it is paramount that corresponding columns across all your source ranges maintain strict data type consistency. For example, if a column is interpreted as containing numbers in the first range, it must consistently contain numbers

in all subsequent ranges. Mismatched data types will often lead to `#VALUE!` errors or incorrect interpretations, such as numeric values being incorrectly treated as text strings. The `QUERY` function rigorously enforces a single data type per output column.

Symmetry for Concatenation: Understand the symmetry requirements for each method. When stacking vertically, all ranges within the **array literal** must have the **same number of columns**. Conversely, when combining horizontally using the comma, all ranges must have the **same number of rows**. Failure to meet these symmetry requirements will result in either an error or misaligned data.

Scalability and External Data: For scenarios involving massive datasets or data spread across different Google Sheets workbooks, pure array literal syntax may face performance limitations. In these cases, consider integrating the `IMPORTRANGE` function. You can embed the `IMPORTRANGE` calls within the **array literal** structure, combining its cross-workbook capabilities with the filtering power of `QUERY` for a robust, cross-document solution.

Leveraging Query Clauses: The true utility of this technique is unlocked when you combine the multi-range array with standard SQL-like clauses. Once your data is consolidated into a single array, you can apply powerful clauses such as `SELECT`, `WHERE` (for filtering), `GROUP BY`, `PIVOT`, and `ORDER BY` (for sorting) to dynamically refine your output, enabling highly customized and advanced data manipulation directly within the consolidated view.

By keeping these fundamental guidelines in mind, you can efficiently harness the full potential of `QUERY` with multiple ranges, developing dynamic and reliable data consolidation solutions in Google Sheets.

Note: The examples above used only two cell ranges for instructional clarity. However, the underlying syntax supports querying from as many cell ranges as your project requires. Simply extend the **array literal** with additional ranges separated by semi-colons (for vertical stacking) or commas (for horizontal merging).

Conclusion

The ability to execute sophisticated queries that span multiple ranges using the `QUERY` function is truly a cornerstone of advanced Google Sheets proficiency. By employing the array literal syntax (`{}` with either semi-colons or commas), users gain a flexible and efficient mechanism to overcome the inherent limitations of single-range data operations, whether the goal is to append rows vertically or join columns horizontally. This powerful feature elevates Google Sheets into a dynamic and highly capable data management platform.

By integrating these vertical and horizontal merging techniques into your daily tasks, you can significantly streamline complex workflows, generate comprehensive reports from previously distributed sources, and perform more insightful data manipulation. We strongly encourage you to experiment with these formulas and explore the myriad possibilities they unlock for robust and scalable data analysis within your spreadsheets.

Additional Resources

To further enhance your proficiency with the `QUERY` function and other advanced Google Sheets capabilities, consider exploring the following essential documentation and tutorials:

Google Docs Editors Help: QUERY function
Understanding QUERY Clauses (SELECT, WHERE, GROUP BY, etc.)
Using IMPORTRANGE with QUERY
Advanced QUERY Function Examples and Use Cases
Working with Ranges in Google Sheets