Learning to Import Data from Multiple Google Sheets using IMPORTRANGE

Authored by

Mohammed looti

October 31, 2025

Harnessing the full analytical power of Google Sheets frequently necessitates the integration of data streams scattered across various documents. The **IMPORTRANGE** function is the foundational element for pulling datasets from one spreadsheet into another. However, when the requirement shifts from simple imports to consolidating information from **multiple distinct spreadsheets** simultaneously--for comprehensive reporting or unified dashboards--a more powerful and structured technique is necessary. This comprehensive guide details the expert method for combining the efficiency of **IMPORTRANGE** with the flexibility of the **QUERY** function and the structural capability of array literals. By mastering this combination, you can import and unify disparate data sources into a single, cohesive, and dynamically updated view.

The core philosophy behind consolidating data from numerous sheets involves stacking the results vertically. This is achieved by creating an array that holds the output of several independent **IMPORTRANGE** calls. The powerful **QUERY** function then acts as the final processor, treating this stacked data as a single, unified table. This methodology is indispensable for any user who manages complex projects, departmental budget tracking, or comprehensive data aggregation that spans several organizational documents.

```
=QUERY({
IMPORTRANGE("URL1", "'sheetname1'!A1:B10");
IMPORTRANGE("URL2", "'sheetname2'!A1:B10");
IMPORTRANGE("URL3", "'sheetname3'!A1:B10");
})
```

This powerful formula structure leverages the array literal, clearly denoted by the curly braces (`{}`). These braces serve to encapsulate and structure the results returned by each individual **IMPORTRANGE** function. The semicolon (`;`) is the crucial operator within the array, instructing Google Sheets to append the subsequent range directly below the preceding one, resulting in a vertical stack. Once the data is combined into this temporary array, the encompassing **QUERY** function gains the ability to filter, sort, and manipulate the entire consolidated dataset. The following sections will provide a detailed, practical walkthrough, focusing on importing data from two sources and subsequently refining the output to eliminate common structural issues, such as redundant header rows.

Understanding the Core Components

To successfully implement this advanced consolidation technique, a clear understanding of the roles played by each function is paramount. While they work together seamlessly, each component has a specific job within the data pipeline. The process begins with **IMPORTRANGE**, moves through the array literal for structure, and concludes with **QUERY** for manipulation and presentation. Recognizing this sequence allows users to effectively troubleshoot and customize

their formulas.

The **IMPORTRANGE** function acts as the secure connection, serving as the gateway to external spreadsheet data. It requires two mandatory arguments: the unique identifier of the source spreadsheet, which can be the full **spreadsheet URL** or the more reliable **spreadsheet ID**, and the **range string**, which precisely defines the data to be retrieved (e.g., `'Sheet Name'!A1:C10`). It is absolutely essential to remember the initial authentication step: the first time you connect to a new source spreadsheet using **IMPORTRANGE**, you must explicitly grant permissions to allow data transfer. This security measure ensures that external data access is controlled and deliberate.

The **QUERY** function is perhaps the most versatile function in Google Sheets, offering capabilities akin to a simplified version of SQL (Structured Query Language). Its primary role in this context is to manipulate and present the combined dataset according to user specifications. When **QUERY** receives the vertically stacked output from the array literal, it treats the entire collection as a single data table. This allows users to apply sophisticated clauses--such as `SELECT`, `WHERE`, `GROUP BY`, and `ORDER BY`--to filter out unwanted rows, sort the combined data, or aggregate numerical metrics, transforming raw imports into meaningful, refined reports.

The **array literal** (`{}`) is the structural glue that enables vertical consolidation. In Google Sheets syntax, curly braces are used to construct temporary, in-memory arrays. By listing multiple ranges or function outputs separated by a semicolon (`;`), you effectively stack the datasets one on top of the other, forming a continuous stream of data. For example, `{IMPORTRANGE(Range A); IMPORTRANGE(Range B)}` results in a consolidated block where the rows of Range A are immediately followed by the rows of Range B. This mechanism is what enables us to concatenate the results of several **IMPORTRANGE** calls into a single, cohesive range for the **QUERY** function to process.

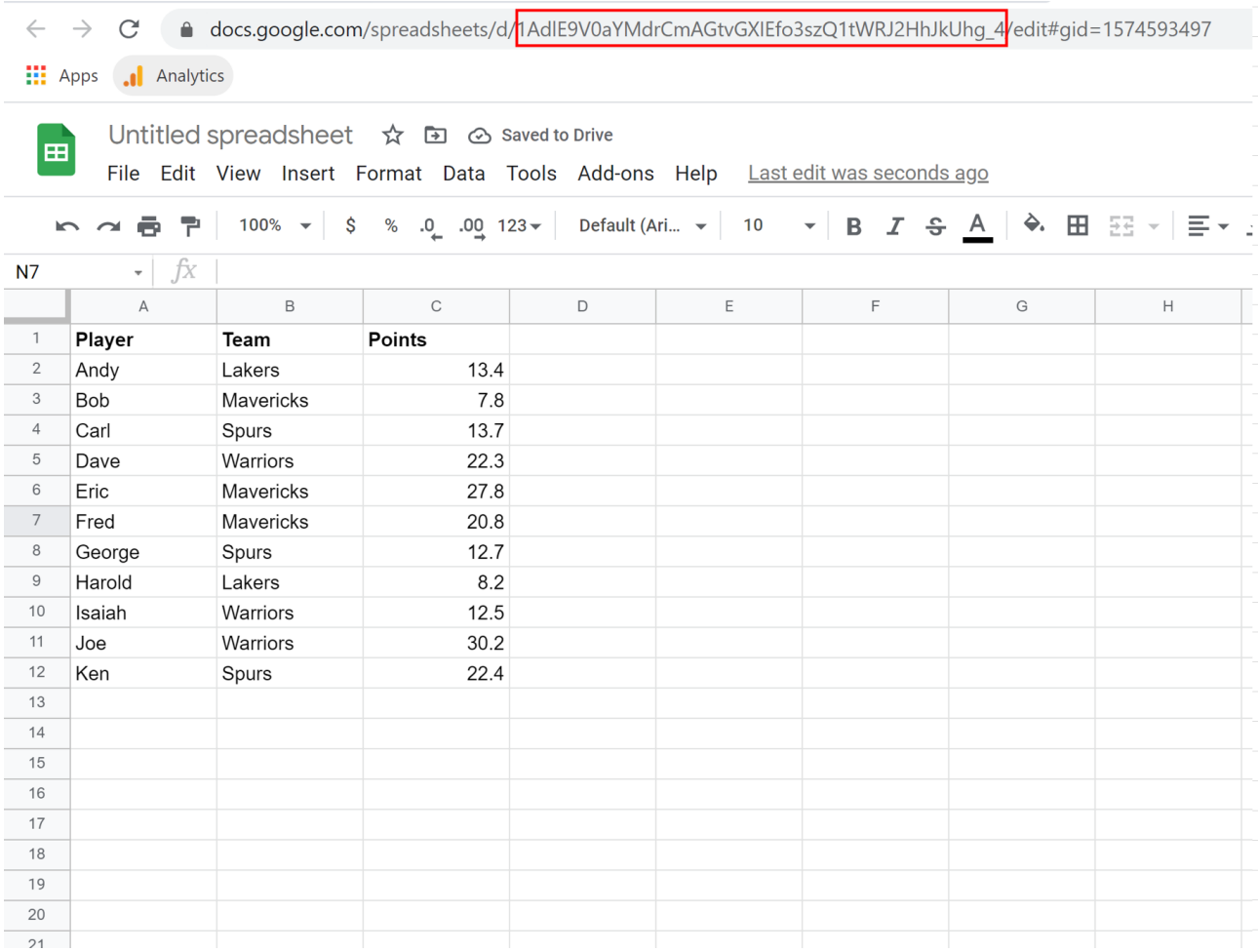
Preparing and Identifying Data Sources

To provide a clear, practical demonstration of this consolidation method, let us establish a typical scenario: aggregating player statistics from two separate Google spreadsheets, perhaps maintained by different project managers or teams. Both spreadsheets are assumed to share an identical structure, containing columns for metrics such as **Player Name**, **Score**, and **Status**. Successful implementation requires precise identification of two key pieces of information for each source sheet: the data range and the unique sheet identifier.

The first source spreadsheet, designated "Stats Sheet 1," holds data within a tab explicitly named **stats**. The most critical component for the **IMPORTRANGE** function is the **spreadsheet ID**. This ID is a long, unique alphanumeric string located in the URL between `/d/` and `/edit`. For instance, if the URL is `https://docs.google.com/spreadsheets/d/1AdlE9V0aYMdrCmAGtvGXIEfo3szQ1tWRJ2HhJkUh`

`g_4/edit`, the ID is `1AdlE9V0aYMdrCmAGtvGXIEfo3szQ1tWRJ2HhJkUhg_4`. Using the ID rather than the full URL is highly recommended as it improves formula stability and readability.

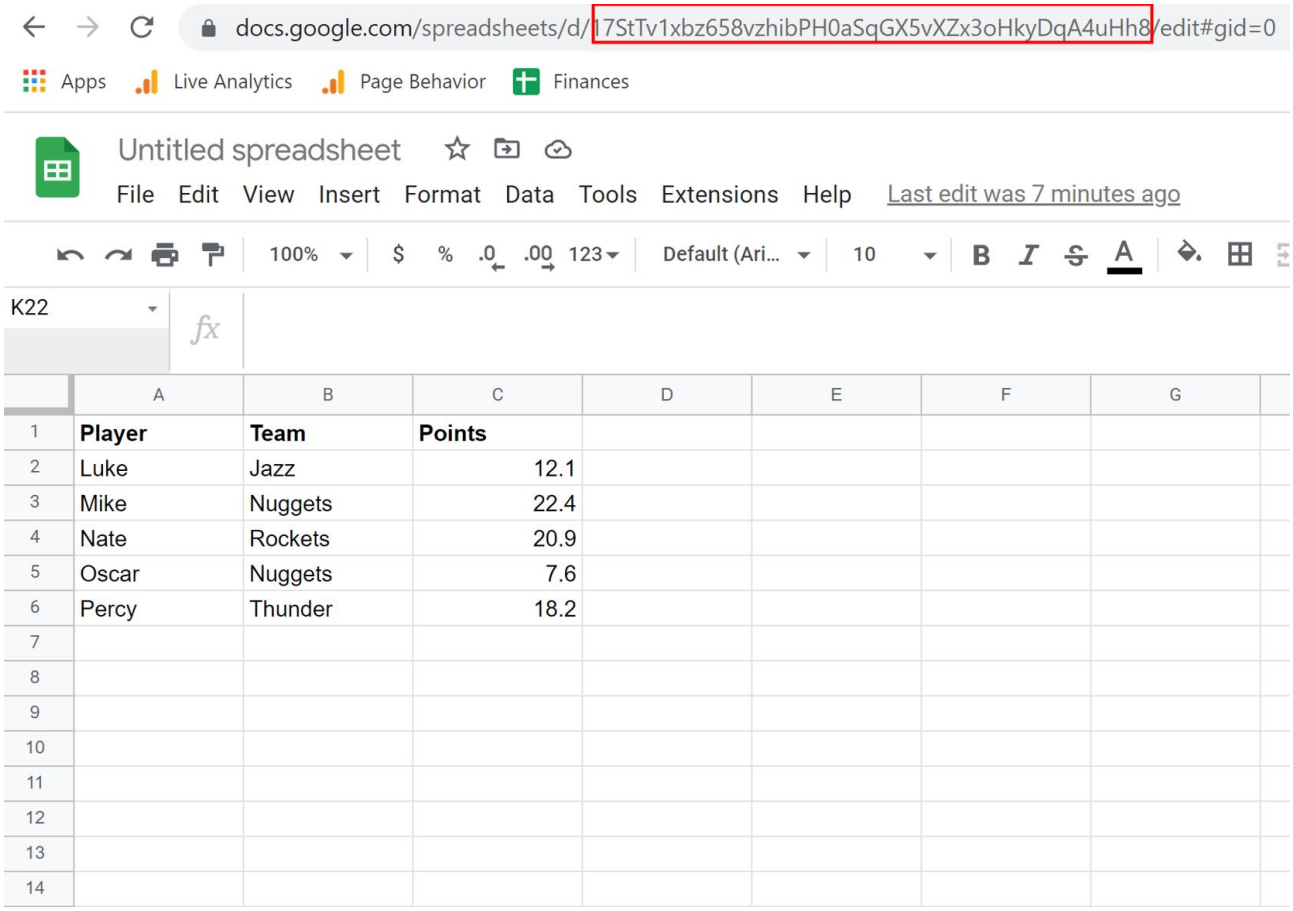
Visualizing the source data confirms the structure we intend to consolidate. "Stats Sheet 1" contains the initial batch of records:



	A	B	C	D	E	F	G	H
1	**Player**	**Team**	**Points**					
2	Andy	Lakers	13.4					
3	Bob	Mavericks	7.8					
4	Carl	Spurs	13.7					
5	Dave	Warriors	22.3					
6	Eric	Mavericks	27.8					
7	Fred	Mavericks	20.8					
8	George	Spurs	12.7					
9	Harold	Lakers	8.2					
10	Isaiah	Warriors	12.5					
11	Joe	Warriors	30.2					
12	Ken	Spurs	22.4					
13								
14								
15								
16								
17								
18								
19								
20								
21								

Similarly, the second source, "Stats Sheet 2," contains supplementary data in a tab named **stats2**. Like the first, this sheet must also have its own distinct **spreadsheet ID**. For the consolidation to work correctly, both imported ranges must contain the same number of columns in the same order, and the data types within corresponding columns must be consistent (e.g., Column C in both sheets should contain scores, not names in one and scores in the other). Inconsistencies in structure or type can lead to frustrating `#VALUE!` or data coercion errors.

Here is a view of the records held in "Stats Sheet 2," ready to be appended to the first dataset:



	A	B	C	D	E	F	G
1	**Player**	**Team**	**Points**				
2	Luke	Jazz	12.1				
3	Mike	Nuggets	22.4				
4	Nate	Rockets	20.9				
5	Oscar	Nuggets	7.6				
6	Percy	Thunder	18.2				
7							
8							
9							
10							
11							
12							
13							
14							

Implementing the Consolidated Import Formula

With our source IDs and ranges identified, we can now construct the core consolidation formula. Open the target Google Sheets document where the aggregated data will reside. In an empty cell, typically `A1`, input the combined formula, ensuring precise syntax for the array literal and the subsequent **QUERY** wrapper. This initial formula focuses solely on retrieving and stacking the data.

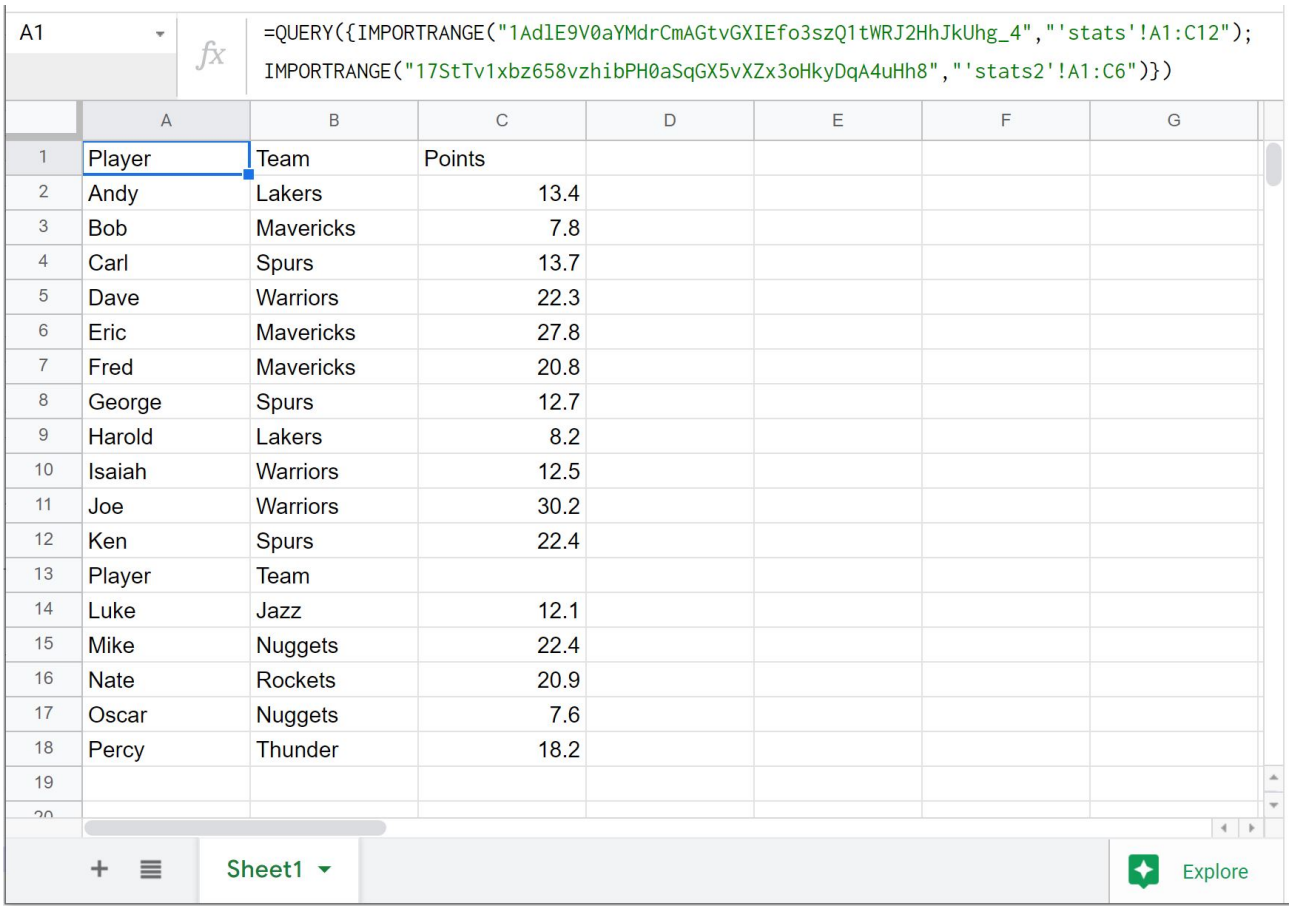
The following syntax demonstrates the chaining of the two **IMPORTRANGE** calls, using the unique IDs and specifying the sheet names and exact data ranges required from each source. Note the critical use of the semicolon (`;`) to separate the two function calls, thereby ensuring vertical stacking within the array:

=QUERY({IMPORTRANGE("1AdlE9V0aYMdrCmAGtvGXIEfo3szQ1tWRJ2HhJkUhg_4","'stats'!A1:C12");
IMPORTRANGE("17StTv1xbz658vzhibPH0aSqGX5vXZx3oHkyDqA4uHh8","'stats2'!A1:C6")})

After entering the formula, if this is the first time you are connecting to either source sheet, you will

observe a `#REF!` error accompanied by a prompt to "Allow access." It is vital to click this prompt and grant the necessary permissions for each external sheet. Once access is successfully authorized, the formula will execute, seamlessly importing and stacking all specified rows from both external spreadsheets into your single target sheet.

The resulting output, as illustrated below, confirms the successful vertical consolidation of the data. However, careful inspection reveals a common structural flaw: because we imported the entire range starting from row 1 in both sheets, the header row from the second sheet (Stats Sheet 2) has been imported along with the data, appearing as a redundant entry (row 13 in this example). Addressing this duplication is the next crucial step in refining the consolidated report.



	A	B	C	D	E	F	G
1	Player	Team	Points				
2	Andy	Lakers	13.4				
3	Bob	Mavericks	7.8				
4	Carl	Spurs	13.7				
5	Dave	Warriors	22.3				
6	Eric	Mavericks	27.8				
7	Fred	Mavericks	20.8				
8	George	Spurs	12.7				
9	Harold	Lakers	8.2				
10	Isaiah	Warriors	12.5				
11	Joe	Warriors	30.2				
12	Ken	Spurs	22.4				
13	Player	Team					
14	Luke	Jazz	12.1				
15	Mike	Nuggets	22.4				
16	Nate	Rockets	20.9				
17	Oscar	Nuggets	7.6				
18	Percy	Thunder	18.2				
19							

Refining Your Output: Excluding Duplicate Headers

A clean, professional dataset requires only one header row, typically placed at the very top. To automatically eliminate the redundant header rows imported from the second and subsequent source sheets, we must utilize the robust filtering capabilities inherent in the **QUERY** function. This is achieved by adding a `WHERE` clause to the overall formula, which specifies a condition that all returned rows must meet, thereby excluding any rows that contain header text.

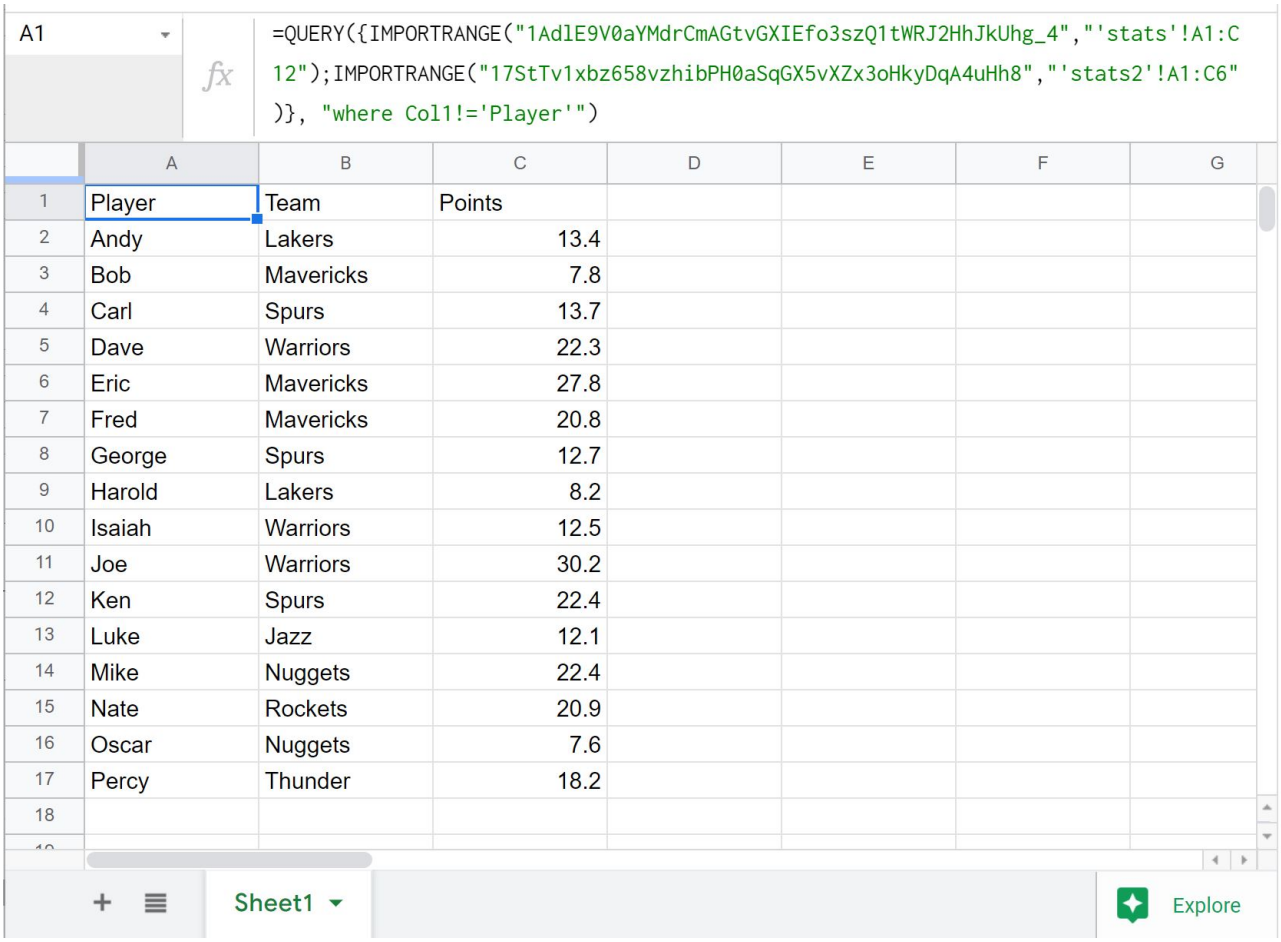
The `WHERE` clause is designed to filter the data source--in this case, the combined array of imported ranges--based on defined criteria. We instruct **QUERY** to look at a specific column and exclude any row where the cell content matches the known header text. Importantly, when referencing columns within the **QUERY** function, you must use generic aliases: `Col1`, `Col2`, `Col3`, and so on, regardless of the column letters (A, B, C) used in the source sheets. Assuming the header text in the first column is 'Player', we will target `Col1`.

The modified formula below incorporates the necessary `WHERE` statement to filter out the unwanted header row. Notice that the entire array literal structure remains the first argument of the **QUERY** function, and the filtering command is passed as the second argument, enclosed in double quotes:

=QUERY({IMPORTRANGE("1AdlE9V0aYMdrCmAGtvGXIEfo3szQ1tWRJ2HhJkUhg_4","'stats'!A1:C12");
IMPORTRANGE("17StTv1xbz658vzhibPH0aSqGX5vXZx3oHkyDqA4uHh8","'stats2'!A1:C6")}, "where Col1!='Player'")

The condition `where Col1!='Player'` explicitly tells the **QUERY** function to exclude any row where the value in the first column is exactly equal to the string 'Player'. This effectively removes the unwanted header row from the second imported sheet while preserving the integrity of all legitimate data entries. If your header is named something different, such as 'Name' or 'Item ID', you must adjust the string within the single quotes accordingly. This technique is easily scalable, ensuring that even if you import ten sheets, only the first sheet's headers are retained.

The final result, demonstrated in the screenshot below, shows a clean, consolidated table where the data from both sources is merged seamlessly, with only the initial header row remaining. This refined output is now ready for further analysis, pivot tables, or dashboard integration.



	A	B	C	D	E	F	G
1	Player	Team	Points				
2	Andy	Lakers	13.4				
3	Bob	Mavericks	7.8				
4	Carl	Spurs	13.7				
5	Dave	Warriors	22.3				
6	Eric	Mavericks	27.8				
7	Fred	Mavericks	20.8				
8	George	Spurs	12.7				
9	Harold	Lakers	8.2				
10	Isaiah	Warriors	12.5				
11	Joe	Warriors	30.2				
12	Ken	Spurs	22.4				
13	Luke	Jazz	12.1				
14	Mike	Nuggets	22.4				
15	Nate	Rockets	20.9				
16	Oscar	Nuggets	7.6				
17	Percy	Thunder	18.2				
18							

Advanced Considerations and Performance Tips

While the combined **IMPORTRANGE** and **QUERY** methodology offers unparalleled data consolidation capabilities, users must be aware of potential pitfalls and implement best practices to ensure formula robustness and spreadsheet performance. The sheer complexity of combining multiple external calls can introduce latency if not managed carefully.

The most frequent challenge encountered is the `#REF!` error, which almost always relates to insufficient permissions. Each time a new external sheet is referenced via **IMPORTRANGE**, the user must explicitly grant access. If the source sheet's sharing settings are subsequently restricted, or if the linking account's access is revoked, the connection will break. Regular auditing of permissions and ensuring that source sheets are shared appropriately with the account running the consolidation formula are critical maintenance steps.

Furthermore, maintaining strict consistency in data types and column alignment is non-negotiable. When the array literal attempts to stack ranges, it expects the corresponding columns across all imported sheets to contain compatible data. If Column B in Sheet 1 holds numbers, but Column B in Sheet 2 holds text strings, Google Sheets may struggle to determine the correct format for the

combined column, often defaulting to text and potentially rendering numerical operations on that column impossible. Ensure that your source sheets are structurally identical before attempting consolidation.

For large-scale data operations, performance optimization is key. While using the full URL works, opting for the concise **spreadsheet ID** is generally recommended for slightly improved efficiency and enhanced formula stability. Crucially, avoid importing excessively large or unnecessary ranges (e.g., `A:Z`). Define your import ranges precisely (e.g., `A1:C100`) to minimize the amount of data the function needs to fetch. If you are consolidating data from dozens of sheets, consider creating dedicated "helper sheets" where individual **IMPORTRANGE** calls reside, and then consolidate the results of those local cells using the array literal, reducing the number of complex external calls within one master formula.

Conclusion and Further Exploration

The ability to skillfully combine the **IMPORTRANGE** function with array literals and the powerful **QUERY** function represents a fundamental advancement in spreadsheet data management. This technique provides a robust, scalable, and dynamic solution for aggregating data from numerous external Google Sheets into a single, comprehensive view, significantly streamlining reporting and analysis workflows.

The demonstrated method not only automates the tedious process of manual data copying but, through the integration of the **QUERY** function, allows for immediate and precise data cleaning, such as eliminating duplicate header rows. This level of control ensures that your consolidated dataset is ready for immediate use, whether for complex pivot tables, chart generation, or integrating into larger business intelligence dashboards. Mastering this syntax is crucial for anyone relying on Google Sheets for complex data pipelines.

To continue enhancing your data manipulation skills, we strongly encourage exploring the full depth of the **QUERY** function's capabilities. Experimenting with advanced clauses such as `GROUP BY` for aggregation, `ORDER BY` for sorting, and `LIMIT` for managing the size of the returned dataset will provide even greater flexibility and efficiency in handling large consolidated reports. The techniques discussed here form the foundation for building sophisticated, interconnected spreadsheet ecosystems.

Additional Resources

For those looking to deepen their expertise in utilizing Google Sheets for data integration and analysis, consider reviewing additional tutorials that demonstrate how **IMPORTRANGE** interacts with other essential functions:

Google Sheets: How to use VLOOKUP with IMPORTRANGE

Google Sheets: How to use SUMIF with IMPORTRANGE

Google Sheets: How to use COUNTIF with IMPORTRANGE

Google Sheets: How to use FILTER with IMPORTRANGE