How to Import Data from Multiple Google Sheets Using IMPORTRANGE

Authored by

stats writer

November 30, 2025

In the realm of data management, consolidating information from disparate sources into a single, cohesive view is often essential. For users of Google Sheets, the primary tool for achieving this powerful integration is the IMPORTRANGE function. This versatile function allows a spreadsheet to securely pull data from another spreadsheet, bridging organizational silos and centralizing reporting. While a single invocation handles one source, achieving consolidation across several sheets requires combining multiple IMPORTRANGE function calls using an array literal structure. Understanding this composite technique is crucial for managing complex datasets across your organizational ecosystem. Furthermore, advanced manipulation, often involving the QUERY function, allows for cleaning, sorting, and filtering the amalgamated data set precisely to meet specific reporting needs, ensuring the output is not only complete but also highly structured.

The Necessity of Data Consolidation using IMPORTRANGE

Modern data workflows often necessitate drawing information from various departmental reports, project trackers, or legacy systems, all stored as separate Google Sheets files. Manually copying and pasting this data is highly inefficient, error-prone, and unsustainable, especially as the source sheets are updated frequently. The IMPORTRANGE function overcomes these challenges by creating a live, dynamic link between spreadsheets. However, when attempting to stack data from multiple sources--for instance, aggregating quarterly sales reports stored in three separate documents--a simple, single use of the function is insufficient. We must employ the technique of array literals to vertically combine the results of multiple import operations.

The core mechanism for combining these disparate data streams involves enclosing multiple IMPORTRANGE function calls within curly braces (`{}`). This structure tells Google Sheets to treat the individual function outputs as elements of a single array. By separating these function calls with a semicolon (`;`), we instruct the system to vertically stack the results, appending the data from the second sheet below the data from the first, and so forth. This method is exceptionally powerful because it maintains the dynamic connection to all original source documents, ensuring that any changes made upstream are immediately reflected in the consolidated destination sheet upon refresh.

Mastering the Syntax for Multi-Source Imports

To successfully import data from several sources simultaneously, you must first ensure that the structure of the data you are importing is consistent across all source sheets. All ranges specified in the imports must have the same number of columns, even if some columns are empty in one source. Failure to match the column structure will result in a `#REF!` error, as Google Sheets cannot reconcile the dimensions of the arrays being stacked. Once structural consistency is confirmed, the combined formula uses the powerful QUERY function to wrap the array literal, allowing for

immediate cleaning and manipulation of the combined dataset.

The standard syntax involves nesting the imports within curly braces, which themselves are wrapped by the QUERY function. This allows the combined data set to be treated as a virtual temporary range that the QUERY function can operate on. For horizontal stacking (if needed, though less common for consolidation), commas would be used instead of semicolons, but for standard appending of rows, the semicolon is essential. The syntax below demonstrates how to achieve vertical consolidation from three distinct spreadsheet sources, retrieving specific ranges from named sheets within each source document:

The following basic syntax outlines the method for combining results from multiple spreadsheets at once:

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

This construction generates a consolidated table by stacking the results of the three individual IMPORTRANGE function calls. Each function requires two primary parameters: the spreadsheet key (or full URL) and the string specifying the sheet name and range in A1 notation. Note that the sheet name must be enclosed in single quotes if it contains spaces or special characters. Furthermore, the use of the QUERY function, even without a specific manipulation clause, is often preferred for stability and future extensibility when dealing with combined arrays in Google Sheets.

Prerequisites and Authorization Handshake

Before any data can be successfully imported, the destination sheet must be granted permission to access the source sheet. This process, often referred to as the "Authorization Handshake," is a critical security measure within the Google ecosystem. When you first execute an **IMPORTRANGE** function involving a new source spreadsheet, the cell where the formula resides will typically display a `#REF!` error along with a prompt to "Allow Access." Clicking this prompt establishes the secure link required for data transfer. It is paramount that the user performing this operation has at least view access to the source spreadsheet; otherwise, the handshake will fail.

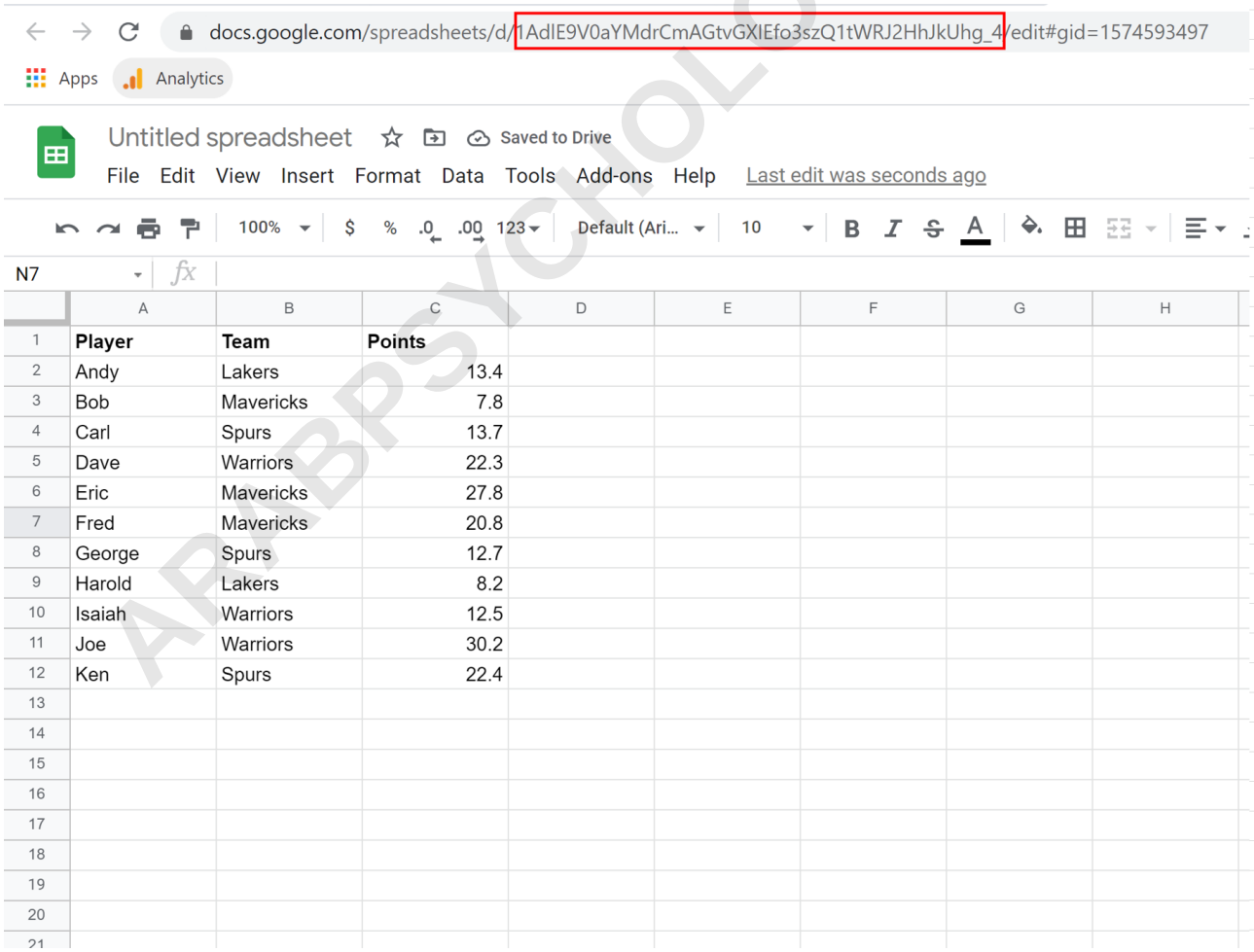
When working with multiple **IMPORTRANGE** calls within a single array, you must perform this authorization handshake for every unique source spreadsheet ID listed in the formula. If you have three source spreadsheets, you will need to initiate three separate authorization requests, one for each unique spreadsheet key. A reliable method for performing this is to temporarily place each

individual **IMPORTRANGE** function in its own cell, grant permission, and confirm the data loads correctly. Once all permissions are secured, you can combine them into the final, complex array formula with confidence that the data access paths are open. This systematic approach saves significant time in troubleshooting array consolidation issues later on.

Example: Combining Data from Two Separate Spreadsheets

Let us walk through a practical scenario where we need to merge player statistics housed in two distinct Google Sheets documents. Assume the goal is to create one master list for comprehensive analysis. We will use the spreadsheet IDs--the long string of characters in the URL--to specify the sources, ensuring maximum efficiency and readability within the formula.

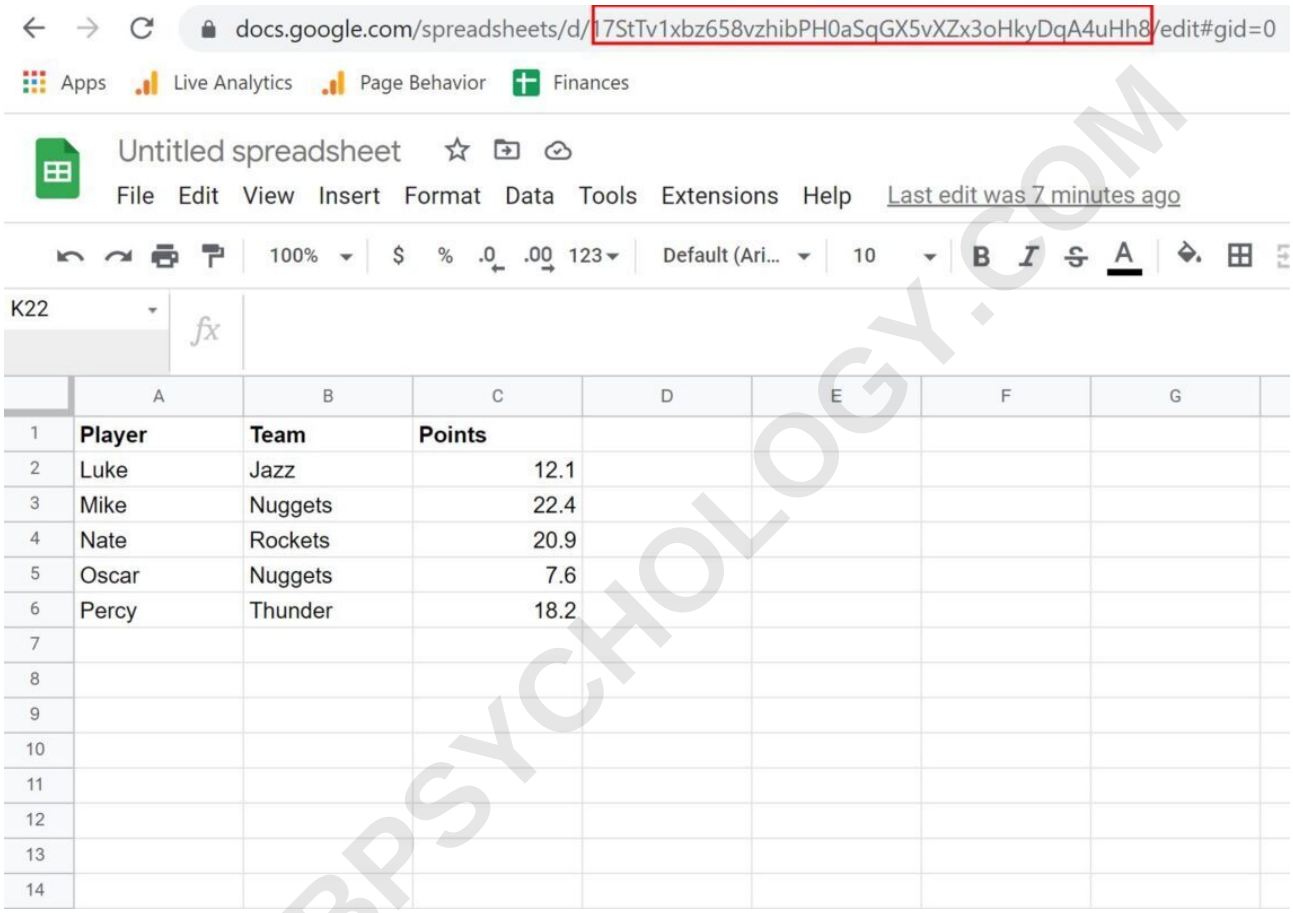
Our first source sheet, which contains historical statistics, is located in a tab named **stats**. This sheet holds the primary dataset we wish to begin our consolidation with. The structure of the data, including player names and their corresponding numerical statistics, is critical for alignment:



	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								

The second source sheet, containing more recent or supplementary data, resides in a tab called

stats2. Critically, we must ensure the column structure (data types and order) of the imported range from this second sheet perfectly mirrors that of the first sheet to prevent array errors. If the columns do not align, the array structure will fail to stack the data correctly, leading to corrupted output or errors. This second sheet provides the additional rows we intend to append to the master list:



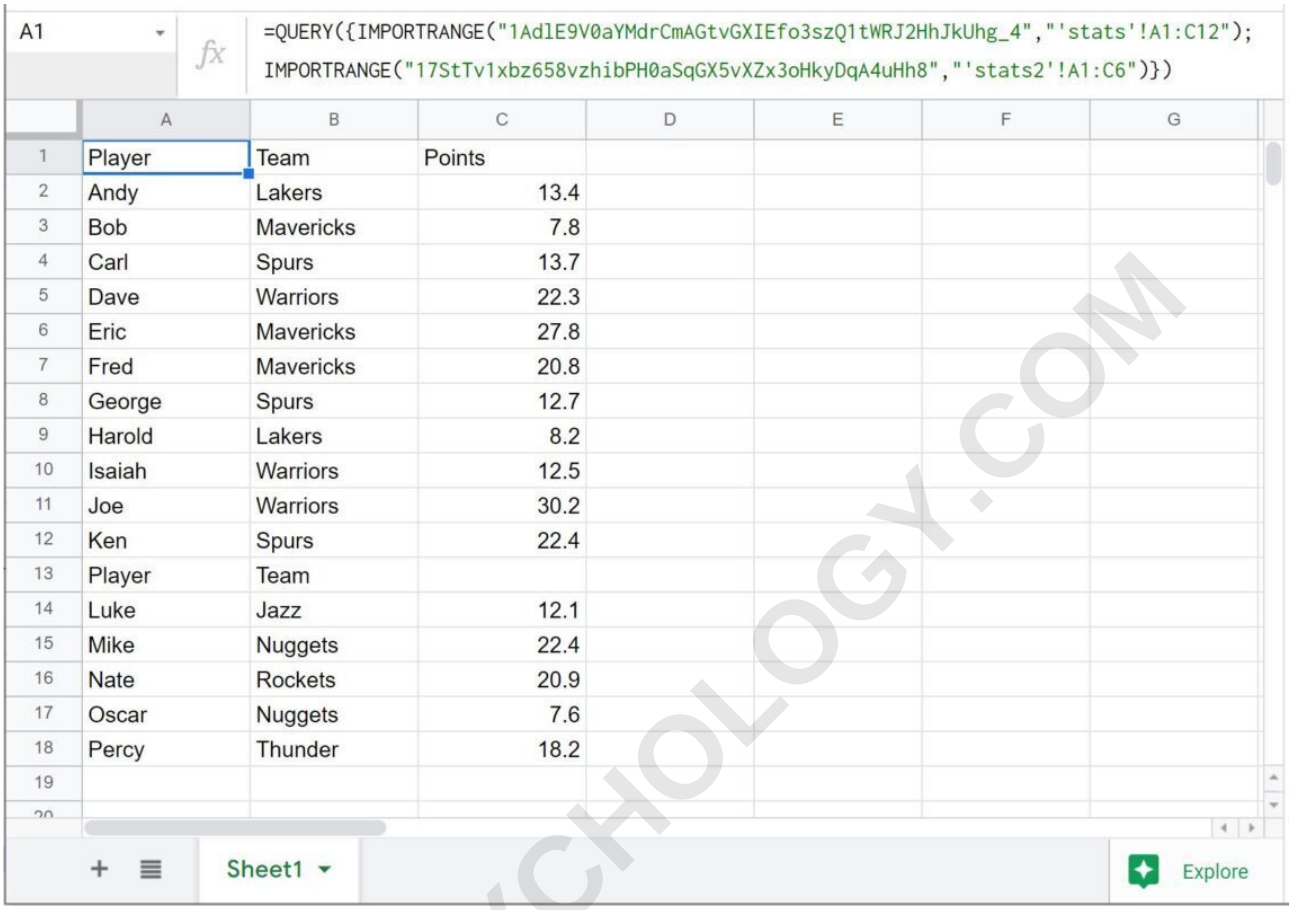
	A	B	C	D	E	F	G
1	**Player**	**Team**	**Points**				
2	Luke	Jazz	12.1				
3	Mike	Nuggets	22.4				
4	Nate	Rockets	20.9				
5	Oscar	Nuggets	7.6				
6	Percy	Thunder	18.2				
7							
8							
9							
10							
11							
12							
13							
14							

By combining the spreadsheet IDs and the desired ranges, we construct the following array literal, wrapped in the QUERY function, to import and stack the data from both sources into our new destination sheet:

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

When executed, this formula results in a single, unified data table. The screenshot below illustrates the result of this operation, showing the seamless concatenation of rows. Notice how the rows from the second sheet are appended directly following the last row imported from the first sheet. This successful consolidation confirms that both the authorization and the array structure were correctly

implemented, yielding a powerful merged dataset.



	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							

Addressing the Issue of Duplicate Headers

A common operational challenge arises when performing vertical stacking: if the imported range from the second and subsequent source sheets includes a header row (e.g., A1:C6 in the second example), that header row will appear mid-table in the consolidated output. This duplication introduces redundant metadata and corrupts the integrity of the unified dataset, making it unsuitable for direct analysis, sorting, or pivot table creation. For instance, in the example above, the header column from the second sheet is returned in row 13 of the combined output, which is undesirable.

To produce a clean, ready-to-use consolidated table, we must selectively exclude these duplicate headers. The most robust method for achieving this data cleaning is leveraging the second parameter of the QUERY function, which enables SQL-like manipulation of the combined array. By adding a `where` clause, we can instruct the query engine to filter out any row where the value in the header column matches the specific header text, thus removing the redundant titles.

The key to effective filtering is referencing the columns correctly within the `where` clause. When

using QUERY on an array literal, the columns are referenced numerically as `Col1`, `Col2`, `Col3`, and so on, regardless of the column letters (A, B, C) used in the source sheets. If the Player name is in the first column of the imported range, it corresponds to `Col1`. We then set a condition to exclude any row where `Col1` is equal to the string value of the header, which is 'Player' in this specific scenario.

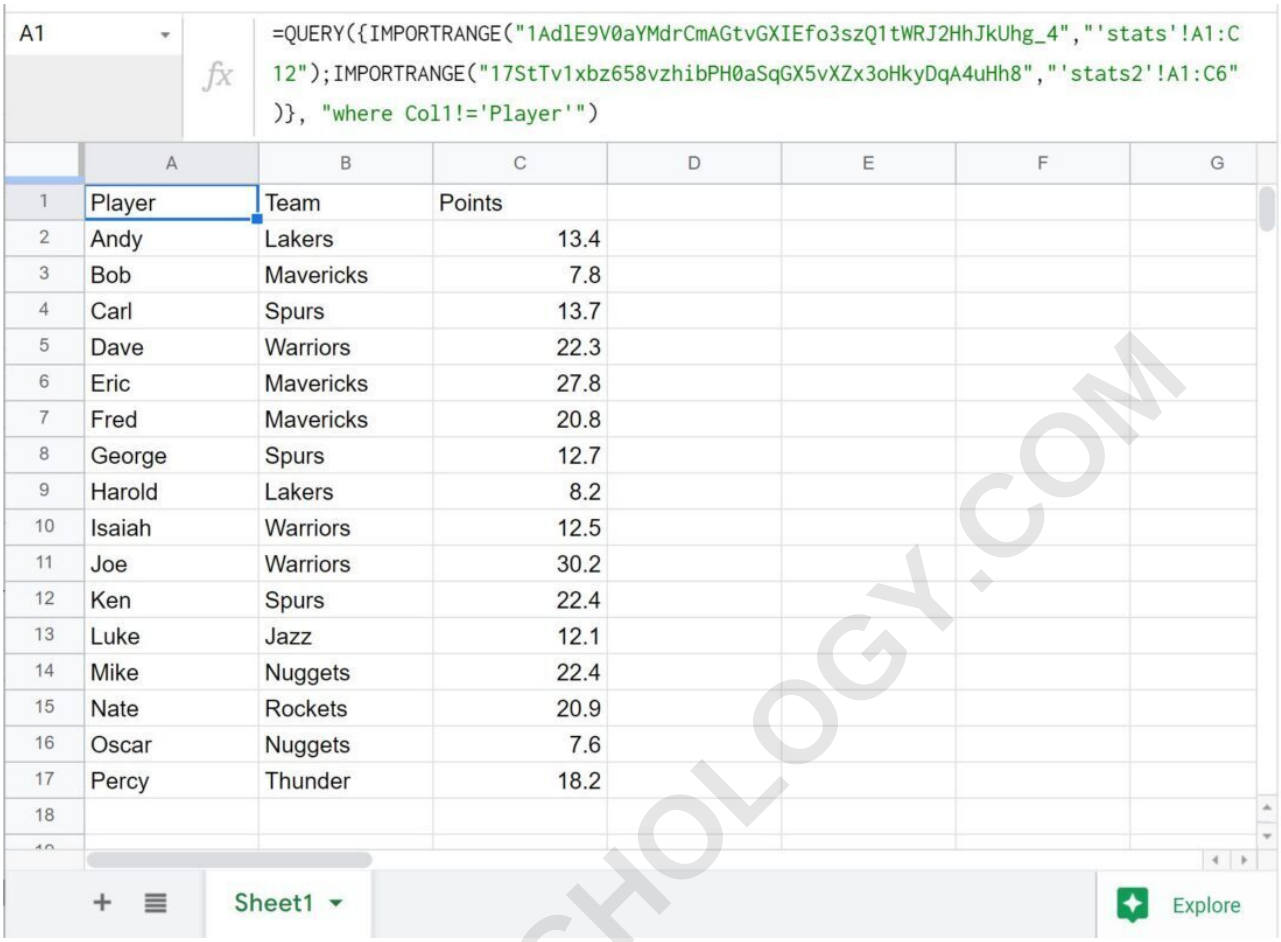
Advanced Filtering and Exclusion using the WHERE Clause

By inserting a specific `where` statement into our combined formula, we can instruct the QUERY function to dynamically clean the data, preserving only the content rows and discarding any duplicate headers. This technique not only removes redundant text but also maintains the live connection, ensuring that if the source data changes, the filtering logic is reapplied automatically upon refresh. The syntax below demonstrates the inclusion of the `where Col1!='Player'` clause, which efficiently prunes the unwanted header row from the consolidated output:

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

This revised formula performs the following sequence of operations: first, it fetches and stacks all data using the array literal; second, the QUERY function receives this array; and finally, it executes the `where` clause, which translates to: "select all rows where the value in the first column is not equal to the text 'Player'." This successfully removes the extraneous header row, which typically appears only once among the data rows.

The visual result of applying this sophisticated filtering mechanism is a dataset that is structurally clean and ready for analysis. As illustrated in the following screenshot, the data from both sources is now seamlessly merged, starting with the original header row from the first sheet, followed immediately by the subsequent data rows from both sources, devoid of the second header. This demonstrates the power of combining array structures with the filtering capabilities of the QUERY function for high-quality data preparation.



	A	B	C
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			

Troubleshooting Common Multi-IMPORTRANGE Errors

While powerful, combining multiple **IMPORTRANGE** calls can lead to specific errors that require careful troubleshooting. The most common issues stem from structural incompatibilities or unauthorized access.

The primary error, as mentioned earlier, is the `#REF!` error, which often indicates a failure in authorization. Always ensure that the handshake has been completed for every unique source spreadsheet ID. Another frequent cause of the `#REF!` error in array formulas is a mismatch in the number of columns being imported. For example, if the first **IMPORTRANGE** pulls A:C (3 columns) and the second pulls A:D (4 columns), the vertical stacking will fail. All ranges specified within the curly braces must have identical column widths. To fix this, adjust the smaller ranges to include empty columns so the column counts match.

A secondary issue involves data type inconsistencies. The QUERY function is strict about column data types. When merging data, if one sheet has numerical values in a column but another sheet has mixed text and numerical values in the same column position, QUERY may interpret the column as containing text, potentially resulting in numerical values being returned as blank or zero.

It is crucial to clean source data beforehand or use array manipulation functions (like `VALUE()` or `TEXT()`) outside of the main consolidation formula to force consistent typing across the array before the query is executed.

Best Practices for Maintaining Consolidated Spreadsheets

Maintaining a consolidated sheet built upon multiple dynamic imports requires adherence to certain best practices to ensure longevity and performance. Firstly, always use the secure spreadsheet ID within the **IMPORTRANGE** function instead of the full URL, as this slightly improves performance and reduces formula length. Secondly, limit the scope of the imported ranges; never use open-ended ranges like A:Z across all imports unless absolutely necessary, as this significantly slows down the refresh rate and burdens the Google Sheets server.

Furthermore, if you intend to perform extensive calculations or visualizations on the consolidated data, consider separating the import process from the analysis process. Use one sheet strictly for the complex multi-IMPORTRANGE array (the Raw Data sheet), and then use simpler reference formulas (like `=Sheet1!A:C`) on subsequent sheets for calculation and reporting. This modular approach isolates potential calculation errors from the core data import mechanism and makes troubleshooting much simpler. Finally, document the source spreadsheets clearly, perhaps in a hidden cell next to the formula, detailing which URL corresponds to which data set, aiding future maintenance efforts.