How to Easily Perform a Left Join in Google Sheets

Authored by

stats writer

November 20, 2025

Mastering data integration is essential for effective spreadsheet analysis, and the concept of a Left Join is fundamental in this process. When working within Google Sheets, a left join is a powerful method used to merge two distinct datasets, ensuring that all records from the primary (left) table are retained, regardless of whether a corresponding match exists in the secondary (right) table. Only the rows from the right table that successfully match the specified key column in the left table are included in the final output.

While standard database environments often utilize the dedicated `LEFT JOIN` keyword within SQL queries, Google Sheets requires alternative approaches to achieve the same result. Data professionals commonly rely on a combination of built-in array functions and lookup tools to efficiently execute a true left join operation. This comprehensive guide details a robust, non-SQL method using the powerful VLOOKUP function alongside ArrayFormula to seamlessly combine your datasets.

Defining the Left Join Concept

The core principle of a Left Join revolves around preservation. Specifically, it ensures that every single record from the primary table--designated as the "left" table--is included in the resulting output. The magic happens when attempting to append data from the secondary table (the "right" table). Only those records in the right table that have a perfect match based on a designated key column (often called the join key) are successfully merged into the corresponding rows of the left table. If no match is found, the resulting columns are populated with blank or error values, ensuring that no data from the primary dataset is lost.

Understanding this concept is vital, as it differs significantly from an inner join, which would only return records where matches exist in both tables. Because we are using Google Sheets instead of a relational database, we must adapt our strategy, utilizing functions like VLOOKUP to iteratively search and return data based on the join key. This method allows spreadsheet users to bypass the limitations of the platform's native querying capabilities for complex merging tasks. The following step-by-step example shows how to use the VLOOKUP function in tandem with array processing to perform a left join in Google Sheets.

Why Use VLOOKUP and ArrayFormula for Left Joins?

While the `QUERY` function in Google Sheets can execute simple data retrieval and filtering, its ability to perform robust, multi-column joins akin to SQL's `LEFT OUTER JOIN` is often restricted or requires overly complicated nested formulas. Using the combination of VLOOKUP and ArrayFormula provides a more transparent and manageable solution for spreadsheet environments. The VLOOKUP function is designed specifically for searching for a value in the first column of a range and returning a value from a corresponding column in the same row, making it the perfect tool for

matching data based on a key identifier.

The addition of ArrayFormula transforms a standard lookup into a mass operation. Instead of manually writing and dragging the formula down thousands of rows--a process prone to errors and slow performance--ArrayFormula allows a single cell formula to spill the results across an entire range of cells simultaneously. This efficiency is critical when dealing with large datasets, ensuring that the join operation is executed quickly and correctly for every row in the left table without manual intervention. This approach is highly favored by advanced Google Sheets users for its power and scalability.

Setting Up the Data: Preparing the Left and Right Tables

The initial requirement for performing a successful Left Join is structuring your data into two distinct, well-defined tables. In this example, our primary dataset, or the **Left Table**, contains basic information such as Player Names and their corresponding Teams. This table will serve as the foundation, meaning every player listed here must be present in the final merged output. First, let's enter the following values for two tables in Google Sheets:

	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Assists**	**Rebounds**
2	Mavs	99		Mavs	30	29
3	Warriors	103		Thunder	22	40
4	Lakers	104		Magic	28	42
5	Nets	98		Kings	25	28
6	Heat	99		Grizzlies	25	34
7	Thunder	104		Pelicans	28	36
8	Magic	110		Hornets	32	37
9	Kings	90		Nets	30	33
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						

Our secondary dataset, the **Right Table**, contains supplemental statistics, such as Assists and Rebounds, indexed by the Team name. The critical element connecting these two tables is the **Team** column, which acts as the unique identifier or the join key. We will specifically focus on using the **Team** column from the Left Table to locate and retrieve the corresponding statistical data from the Right Table (Columns E through G). We will perform a left join in which we keep all rows from the left table and only join in the rows from the right table that have matching values in the **Team** column.

Ensuring Data Integrity: Preparing the Output Range

Before applying the merging formula, it is best practice to define a dedicated output area for the joined dataset. This separation prevents overwriting the source data and provides a clean canvas for the results of the left join operation. Begin by creating a copy of the columns from the Left Table (Player and Team) and paste them into a new, clearly demarcated range, such as starting from row 12. This copied data, which includes all the records we wish to preserve, serves as the base structure for our final joined table. Next, let's copy and paste the values of the first table into a new cell range:

	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Assists**	**Rebounds**
2	Mavs	99		Mavs	30	29
3	Warriors	103		Thunder	22	40
4	Lakers	104		Magic	28	42
5	Nets	98		Kings	25	28
6	Heat	99		Grizzlies	25	34
7	Thunder	104		Pelicans	28	36
8	Magic	110		Hornets	32	37
9	Kings	90		Nets	30	33
10						
11						
12	**Team**	**Points**				
13	Mavs	99				
14	Warriors	103				
15	Lakers	104				
16	Nets	98				
17	Heat	99				
18	Thunder	104				
19	Magic	110				
20	Kings	90				
21						
22						
23						
24						

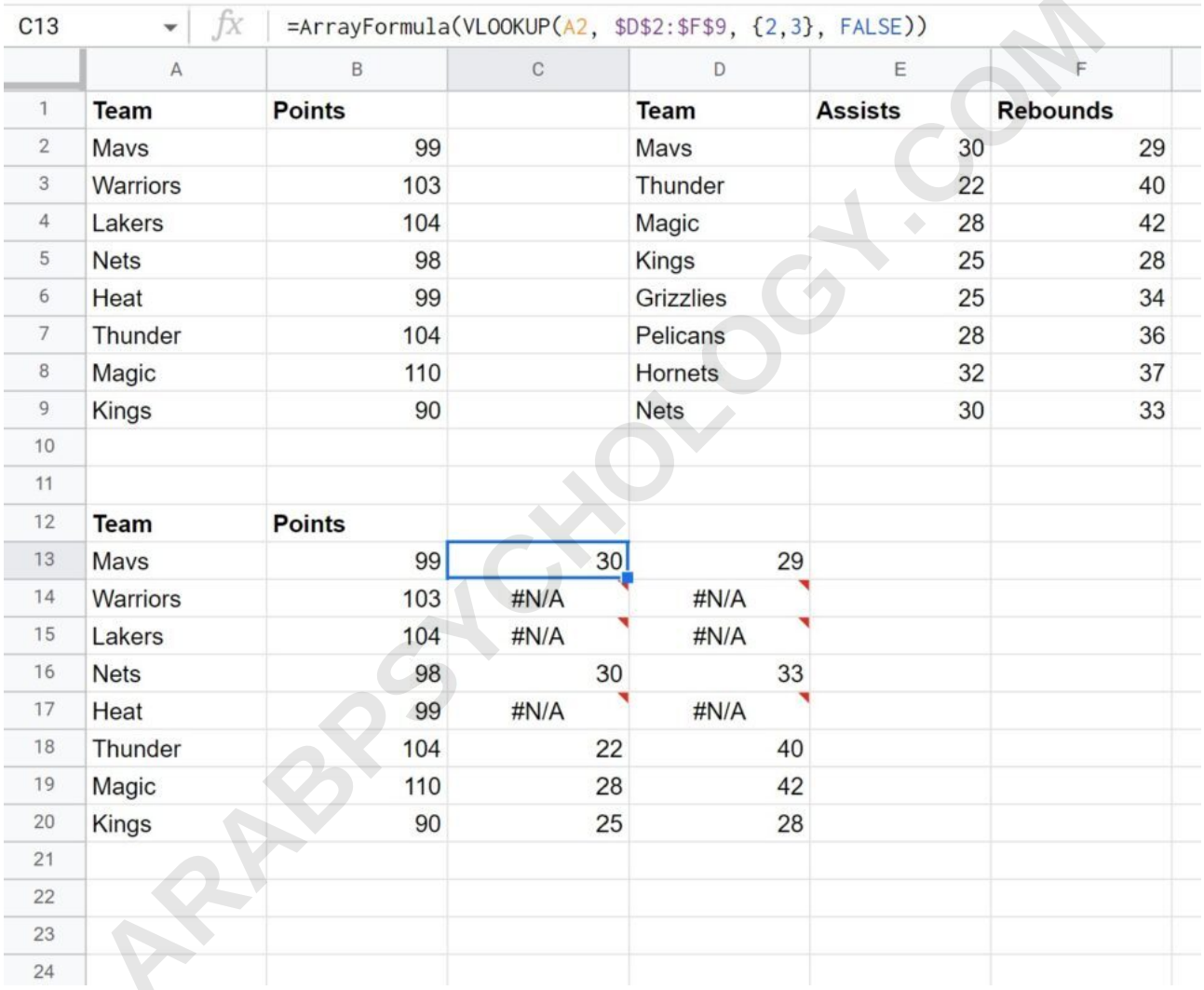
Executing the Left Join Formula using ArrayFormula and VLOOKUP

The crucial step involves deploying a single, powerful formula that executes the lookup operation across the entire range of the left table. We utilize the combined strength of ArrayFormula and VLOOKUP to achieve this efficient merge. Navigate to the first cell in the designated output column, which in our case is cell **C13**, located immediately adjacent to the team names we copied. Input the formula structure below, ensuring that the lookup key references the appropriate column in your copied data that contains the team names. This setup enables the array formula to process all rows efficiently.

The formula instructs VLOOKUP to search the entire Right Table range (`$E$2:$G$9`). By using absolute references (the dollar signs), we ensure that this lookup range remains fixed as the formula calculates results for all rows. Furthermore, we leverage VLOOKUP's ability to return multiple columns by enclosing the column index numbers (2 and 3, corresponding to Assists and Rebounds) within curly braces: `{2,3}`. Enter the following formula into cell **C13**:

=ArrayFormula(VLOOKUP(A2, $E$2:$G$9, {2,3}, FALSE))

Once entered, the array capabilities ensure that the results automatically populate columns C and D for all rows corresponding to the copied data, completing the data merge. We'll then drag and fill this formula down to each remaining cell in column C if the array result does not automatically spill over the entire range, although typically ArrayFormula handles this automatically.



	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Assists**	**Rebounds**
2	Mavs	99		Mavs	30	29
3	Warriors	103		Thunder	22	40
4	Lakers	104		Magic	28	42
5	Nets	98		Kings	25	28
6	Heat	99		Grizzlies	25	34
7	Thunder	104		Pelicans	28	36
8	Magic	110		Hornets	32	37
9	Kings	90		Nets	30	33
10						
11						
12	**Team**	**Points**				
13	Mavs	99	30	29		
14	Warriors	103	#N/A	#N/A		
15	Lakers	104	#N/A	#N/A		
16	Nets	98	30	33		
17	Heat	99	#N/A	#N/A		
18	Thunder	104	22	40		
19	Magic	110	28	42		
20	Kings	90	25	28		
21						
22						
23						
24						

To ensure maximum readability, it is highly recommended to add the appropriate column headers, **Assists** and **Rebounds**, to the cells above the newly merged data (C12 and D12). This finalizes the data presentation and confirms that the left join is now complete. Feel free to add the column headers **Assists** and **Rebounds** as well:

	A	B	C	D	E	F
1	**Team**	**Points**		**Team**	**Assists**	**Rebounds**
2	Mavs	99		Mavs	30	29
3	Warriors	103		Thunder	22	40
4	Lakers	104		Magic	28	42
5	Nets	98		Kings	25	28
6	Heat	99		Grizzlies	25	34
7	Thunder	104		Pelicans	28	36
8	Magic	110		Hornets	32	37
9	Kings	90		Nets	30	33
10						
11						
12	**Team**	**Points**	**Assists**	**Rebounds**		
13	Mavs	99	30	29		
14	Warriors	103	#N/A	#N/A		
15	Lakers	104	#N/A	#N/A		
16	Nets	98	30	33		
17	Heat	99	#N/A	#N/A		
18	Thunder	104	22	40		
19	Magic	110	28	42		
20	Kings	90	25	28		
21						
22						
23						
24						

The left join is now complete.

Interpreting the Results: Handling #N/A Values

A successful Left Join guarantees that all rows from the original data are present in the final output. The most crucial aspect of interpreting the results is understanding how unmatched data is handled. If a given team didn't have a match in the right table, then a `#N/A` value is shown for the Assists and Rebounds columns.

This error is not a sign of failure in the join process; rather, it is the expected behavior for missing data in a left join operation, clearly indicating that the specific record in the left table had no counterpart in the right table. For example, the **Mavs** team successfully matched an entry in the Right Table, so they have corresponding values for Assists and Rebounds. However, the rows associated with the **Warriors** team show the `#N/A` error across the statistical columns, signifying that the Warriors team was present in the Left Table but was entirely absent from the Right Table.

To enhance the readability and further clean the data, advanced users often nest this entire formula structure within an `IFERROR` function. This practice allows you to replace the standard `#N/A` error with a more user-friendly indicator, such as a zero (0), a hyphen (-), or an empty string (""), depending on whether missing data should be treated as zero or simply displayed as blank. Utilizing `IFERROR` ensures the output remains clean and ready for immediate reporting or further calculation.

Summary and Further Refinements

By mastering the combination of VLOOKUP and ArrayFormula, you gain a versatile tool for advanced data manipulation in Google Sheets, enabling you to integrate data effectively and derive deeper insights from your spreadsheets. This technique is especially powerful because it is non-destructive, preserving the full set of your original data while selectively enriching it with supplemental information. Always ensure data consistency between your key columns and utilize absolute references for fixed ranges to guarantee the reliability of your joins.