

# How to Extract Data From Another Sheet Based on Specific Criteria

Authored by  
**stats writer**

November 21, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Extract Data From Another Sheet Based on Specific Criteria*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=98721>

Efficiently managing and analyzing large datasets often requires the ability to selectively extract information from various sources within a single workbook. When working within spreadsheet environments like [Google Sheets](#), one of the most common and powerful tasks is to pull data from one sheet to another based on specific conditions or criteria. This process of conditional data retrieval is fundamental to creating dynamic dashboards, summarized reports, and streamlined workflows. While many users initially turn to basic lookup functions, mastering more advanced techniques ensures greater flexibility and scalability, especially when dealing with complex filtering requirements or varying data structures.

The challenge typically arises when a user needs to return not just a single value, but an entire row or range of rows that satisfy a specified set of requirements. For instance, you might need to extract all sales transactions recorded by a specific employee, or list all inventory items below a certain stock level. Achieving this accurately and dynamically requires functions that can interpret database-style language, allowing for powerful filtering and selection operations. This article explores the optimal methods for performing this crucial data operation in Google Sheets, focusing on the specialized tools available beyond traditional lookup methods.

## Understanding the Limitations of VLOOKUP

Many users attempting conditional data retrieval initially consider using the VLOOKUP function. This function is indeed a staple for lookup operations, designed to search for a specified value in the first column of a given range and return a corresponding value from a designated column in the same row. The standard syntax for VLOOKUP requires four arguments: the search key, the range to search within, the column index of the value you wish to return, and a boolean value indicating whether you desire an exact or approximate match.

While VLOOKUP is highly effective for single-cell lookups, it possesses significant limitations when the goal is to extract multiple records based on a criterion. Specifically, VLOOKUP is inherently designed to stop after finding the first match. If your dataset contains ten rows matching your criteria--say, ten transactions by 'Employee A'--VLOOKUP will only return the data associated with the very first instance it encounters, ignoring the remaining nine. Furthermore, it struggles when the lookup column is not the leftmost column of the range, often requiring complex workarounds using array formulas or other indexing techniques, which quickly become cumbersome and difficult to maintain.

Given these limitations, especially the inability to return multiple matching records, spreadsheet experts typically recommend bypassing VLOOKUP entirely for large-scale conditional data extraction. Instead, the focus shifts to functions that are built for array output and database-style filtering, such as the powerful QUERY function, which provides a declarative and SQL-like approach to data manipulation.

## Mastering the Google Sheets QUERY Function

The most powerful and flexible tool available in [Google Sheets](#) for extracting data based on specific conditions is the [QUERY function](#). This function is essentially an embedded database language interpreter, allowing users to write sophisticated SQL-like statements directly within a cell formula. Unlike traditional spreadsheet formulas that operate on single cells or defined ranges, the [QUERY function](#) processes entire datasets, enabling complex operations like filtering, sorting, aggregation, and conditional selection in a single, concise command.

By utilizing the [QUERY function](#), you gain the ability to specify exactly which rows you wish to select based on any number of conditions, and crucially, it automatically returns all rows that meet those conditions. This capability directly addresses the primary deficiency of VLOOKUP when performing conditional [data retrieval](#) across sheets. When you need to pull data from another sheet that satisfies certain [criteria](#), the [QUERY function](#) provides unparalleled clarity and efficiency.

The QUERY function operates on three primary arguments, structured as follows: =QUERY(data, query, ). The first argument, data, specifies the source range (e.g., Sheet1!A1:C11). The second argument, query, is the core SQL-like command that dictates the selection logic. The third argument, , is optional but crucial, specifying the number of header rows included in the source data, typically set to 1 to ensure correct interpretation.

### Syntax Breakdown of the QUERY Function

The core filtering logic resides within the query string. For a comprehensive extraction, we typically use the SELECT \* WHERE pattern. The SELECT \* clause instructs the function to return all columns from the defined data range. The WHERE clause then introduces the conditional logic that must be satisfied for a row to be included in the output. When referencing columns within the query string itself, it is mandatory to use the column letter notation (A, B, C, etc.), regardless of the header names in your spreadsheet.

For instance, if you want to pull all data from Sheet1 where the value in the first column (Column A) is equal to 'Mavs', the necessary formula structure is demonstrated below. This specific example pulls data from the range A1:C11 located in Sheet1:

**=query(Sheet1!A1:C11, "select \* where A='Mavs'", 1)**

This formula is highly descriptive: it instructs [Google Sheets](#) to select \* (all columns) from the specified source range, but only where the value in Column A is exactly equal to the text string Mavs. The inclusion of the final argument, 1, confirms that the range includes one header row, which is beneficial for clarity in the resulting output table. This concise syntax provides a robust foundation for conditional data extraction.

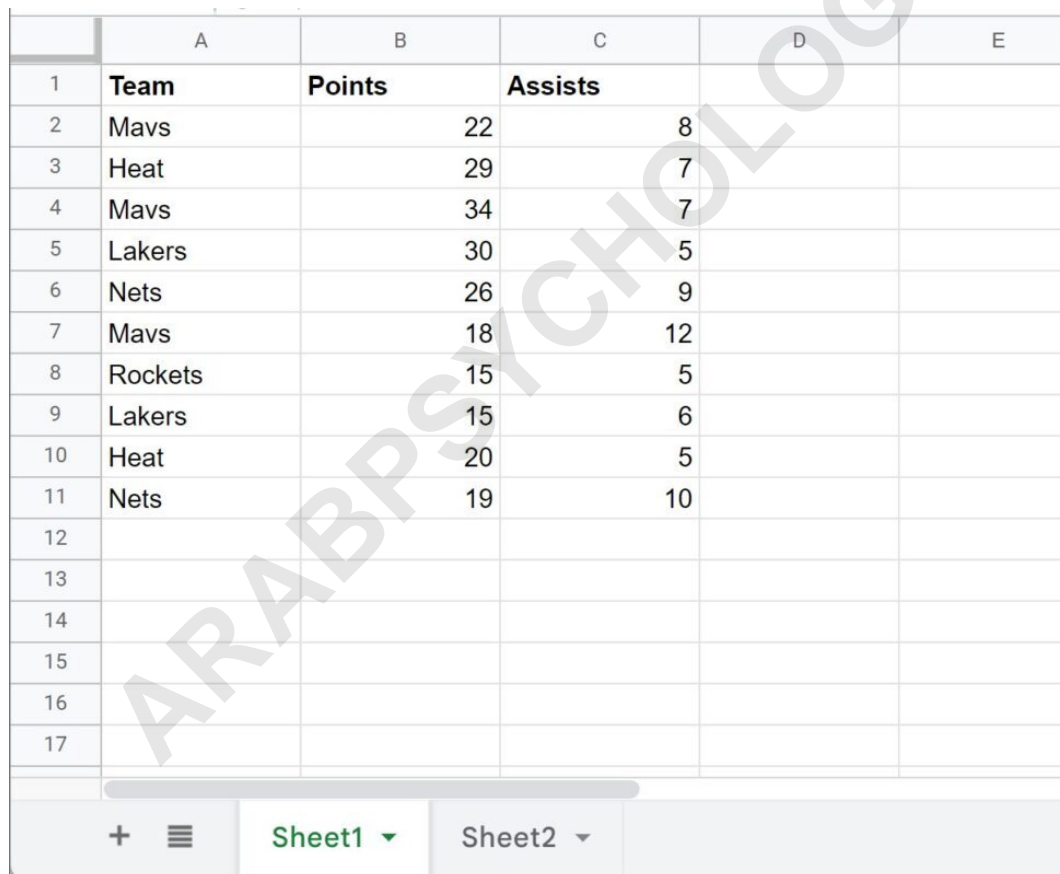
## Practical Example: Pulling Data Based on Single Criteria

To solidify the understanding of the QUERY function, we will follow a practical, step-by-step example using a sample dataset. Our goal is to dynamically filter raw data residing in Sheet1 and transplant only the relevant subset into Sheet2, ensuring the output remains current with the source data. This procedure is common in reporting where source data is constantly updated, but analysts only need to view subsets relevant to specific teams or categories.

Our first requirement is to set up the source data. We utilize a small dataset tracking basketball team statistics in Sheet1, spanning columns A through C: Team, Points, and Assists. We must ensure the column header names are clear, as these will guide our filtering logic, even though the QUERY function uses letter references internally.

The sample data in Sheet1 looks like this:

	A	B	C	D	E
1	<b>Team</b>	<b>Points</b>	<b>Assists</b>		
2	Mavs	22	8		
3	Heat	29	7		
4	Mavs	34	7		
5	Lakers	30	5		
6	Nets	26	9		
7	Mavs	18	12		
8	Rockets	15	5		
9	Lakers	15	6		
10	Heat	20	5		
11	Nets	19	10		
12					
13					
14					
15					
16					
17					



Our specific objective is to execute a selective data retrieval operation that isolates all rows pertaining to the "Mavs" team, transferring this filtered data to cell A1 of Sheet2.

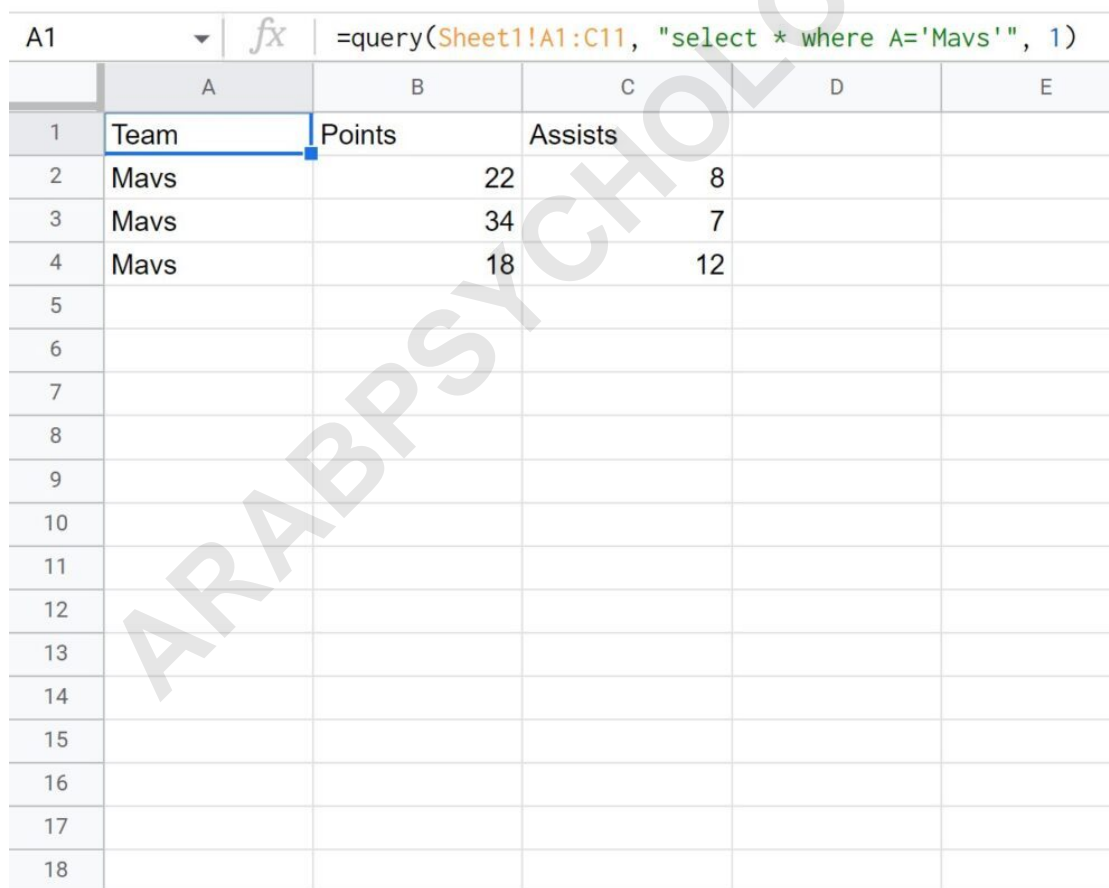
## Implementing the Single-Criterion Query Formula

To perform the conditional filtering, navigate to Sheet2 and select cell A1. This cell will house the array formula, acting as the anchor point for the entire filtered table output. The formula must meticulously reference the source sheet, the full data range, and the required filtering condition.

Since the 'Team' column is Column A in our range, the criterion is specified as where A='Mavs'. We input the complete formula into cell A1 of Sheet2:

**=query(Sheet1!A1:C11, "select \* where A='Mavs'", 1)**

Upon execution, the QUERY function processes the source data and populates Sheet2 exclusively with the matching records. All rows where the Team column (Column A) is equal to Mavs are dynamically pulled, resulting in a clean, filtered view. The result generated by the formula is visualized below:



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	Team	Points	Assists		
2	Mavs	22	8		
3	Mavs	34	7		
4	Mavs	18	12		
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

Notice that all three rows matching the condition from the original data in Sheet1 have been accurately transferred to Sheet2. This capability to return multiple records based on a single criterion is the key advantage of using QUERY over traditional lookup functions.

## Advanced Data Pulling: Using Multiple Criteria

The true utility of the QUERY function is unleashed when you need to impose multiple, simultaneous filtering criteria. For instance, a common request might be to filter data based on both a categorical label (like a team name) and a quantitative threshold (like points scored). The SQL-like structure of the QUERY language supports logical operators, most notably AND and OR, allowing for sophisticated conditional filtering.

To illustrate, let us refine our data extraction to pull data only if two specific conditions are met concurrently: (1) the Team column (Column A) must equal Mavs, AND (2) the Points column (Column B) must be greater than 20. By chaining these conditions using the AND operator within the WHERE clause, we narrow the result set significantly, demanding that both requirements be satisfied for inclusion.

The refined syntax incorporates the numeric comparison and `B>20` directly after the initial text comparison. Remember that numerical values are not enclosed in quotes when used in criteria. The resulting formula to be placed in Sheet2 is:

```
=query(Sheet1!A1:C11, "select * where A='Mavs' and B>20", 1)
```

Executing this advanced formula provides a highly targeted result set, as seen in the following visual representation:

	A	B	C	D	E
A1	<code>=query(Sheet1!A1:C11, "select * where A='Mavs' and B&gt;20", 1)</code>				
1	Team	Points	Assists		
2	Mavs	22	8		
3	Mavs	34	7		
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

The output correctly shows only the two rows from Sheet1 that met both the team requirement (Mavs) and the points requirement (Points greater than 20). This precise control over conditional extraction is why the QUERY function is essential for complex data retrieval tasks in Google Sheets.

## Handling Different Data Types in Query Criteria

When defining filtering logic within the QUERY function, successful execution relies heavily on correctly handling various data types--specifically text, numbers, and dates. Incorrect delimiter usage is the most frequent error encountered by new users of this function.

For filtering based on text data, the criterion value must be enclosed strictly in single quotes, such as `where A='Bulls'`. This syntax is crucial for the query language to recognize the value as a literal string to be matched. Text comparisons are generally case-sensitive, so consistency between the criterion and the source data is paramount.

In contrast, numeric data values must never be enclosed in quotes. They are used directly with mathematical operators like `=`, `>`, `<`, and `<=`. For example, filtering for records where assists (Column C) are less than 15 would be written as `where C<15`. If you need to include a decimal or

scientific notation, it follows the same unquoted rule.

For filtering by date data, Google Visualization API Query Language requires a specialized format. Dates must be formatted as YYYY-MM-DD and preceded by the date keyword. To filter records occurring exactly on December 31, 2023, the criterion would be where D = date '2023-12-31'. Adhering to these type-specific rules ensures accurate and error-free execution of your conditional queries.

## Comparison to Other Conditional Functions

While the QUERY function stands out for its flexibility and power, [Google Sheets](#) offers other conditional functions, such as FILTER and DGET, which are also capable of pulling data based on [criteria](#). Understanding the trade-offs between these functions helps in choosing the most appropriate tool for a given task.

The FILTER function is highly intuitive and easy to use, requiring only the range to filter and one or more conditional arrays (e.g., =FILTER(Sheet1!A:C, Sheet1!A:A="Mavs")). It excels in simple filtering scenarios and is often faster for basic filtering operations than QUERY. However, FILTER lacks the advanced database capabilities of QUERY, such as sorting, grouping, aggregation (SUM, AVG, COUNT), and the ability to select only specific columns (e.g., SELECT A, C instead of SELECT \*).

In contrast, DGET (Database Get) is designed to retrieve a single value from a database-like table based on a set of [criteria](#) defined in a separate range, similar to [VLOOKUP](#), but allowing multiple criteria. The critical limitation of DGET is that it will return an error if it finds multiple matching records, meaning it cannot perform the multi-row data extraction necessary for building dynamic reports. Therefore, for robust, multi-row, and highly customized data extraction across sheets, the QUERY function remains the industry standard in [Google Sheets](#).

## Summary and Best Practices for Data Extraction

The process of selectively extracting data from one sheet to another based on specified conditions is a cornerstone of advanced spreadsheet management. While initial attempts might involve simple functions like [VLOOKUP](#), their inherent inability to return multiple matches necessitates the use of more powerful array-based tools.

The QUERY function in [Google Sheets](#) provides the most versatile and efficient solution for this challenge. By leveraging its SQL-like language structure, users can define precise [criteria](#) for filtering, allowing for both single and multiple conditional checks using logical operators like AND. This capability ensures that you can always perform complex, dynamic [data retrieval](#) operations with confidence.

Best practices for utilizing the QUERY function include:

**Explicit Sheet References:** Always include the sheet name when referencing data from another sheet (e.g., Sheet1!A1:C11).

**Use Column Letters:** Within the query string itself, refer to columns using their letter (A, B, C) and not header names or cell references (A1, B5).

**Mind Data Types:** Ensure text strings are enclosed in single quotes, numbers are not quoted, and dates use the date 'YYYY-MM-DD' format.

**Set the Header Count:** Always specify the correct number of header rows in the third argument (typically 1) to ensure accurate output formatting.

By adhering to these guidelines, you can ensure your conditional data pulls are accurate, maintainable, and highly responsive to changes in your source data.