

How to Easily Filter Google Sheets Data by Month Number Using Query

Authored by
stats writer

November 30, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Filter Google Sheets Data by Month Number Using Query*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=102621>

The ability to efficiently filter and analyze data is fundamental to powerful spreadsheet usage. In Google Sheets, the primary tool for advanced data manipulation is the QUERY function. While querying numerical or text fields is straightforward, filtering based on date components, such as the month number, introduces a specific complexity that must be addressed using built-in date functions. This guide details how to leverage the specialized MONTH function within your **QUERY** statement to accurately isolate records corresponding to a particular month.

To successfully execute a query based on a month number, you must understand a critical characteristic of date handling in Google Sheets. The `MONTH()` function, when used within the query language, is designed to return the month number of a given date or timestamp, but it utilizes a **zero-based index**. This means January is represented by 0, February by 1, and December by 11. Consequently, when constructing a query that filters data for, say, February (the second calendar month), we must look for the value 1 and then adjust the syntax accordingly, typically by adding 1 to the result of the `MONTH()` function to align with the standard 1-12 calendar representation.

For instance, if your goal is to extract all entries from a range where the date in column A falls within January, the statement requires careful formulation. Instead of simply searching for month 1, you must compensate for the zero-indexing. The structure generally follows the pattern: `SELECT * WHERE MONTH(DateColumn) + 1 = TargetMonthNumber`. This ensures that the QUERY function correctly interprets the filter criteria, enabling powerful and dynamic data reporting based on specific temporal dimensions.

Understanding the Google Sheets QUERY Function

The QUERY function is perhaps the most versatile and powerful function within the Google Sheets ecosystem. It allows users to manipulate, filter, and aggregate data using a syntax reminiscent of the Structured Query Language (SQL), making complex data operations manageable through a single formula. Mastering this function is key to generating reports, summaries, and dynamic dashboards directly within your spreadsheet environment, bypassing the need for extensive manual filtering or complex array formulas.

The structure of the function is defined by three main arguments: the data range, the query string, and the header count. The query string itself is where the filtering logic resides, encapsulated in double quotes. This string must follow SQL-like syntax, using clauses like `SELECT`, `WHERE`, `GROUP BY`, and `ORDER BY`. When dealing with dates, the **QUERY** function provides specialized capabilities, allowing internal calculations on date columns, such as extracting the month, year, or day components for filtering purposes.

However, it is vital to remember that all non-standard functions used within the query string must

be recognized by the Sheets query parser. This means we cannot use standard Google Sheets functions directly inside the quote marks unless they are specific functions supported by the internal query language, such as `YEAR()`, `MONTH()`, and `DAY()`. This distinction is why filtering by month requires the special handling detailed in the following sections, particularly concerning the zero-indexed output of the `MONTH()` function within the query environment.

Decoding the MONTH() Function and Zero-Indexing

As briefly mentioned, the most common pitfall when filtering by month in [Google Sheets](#) is the interpretation of the MONTH function output within the **QUERY** context. Unlike many conventional date systems where January is assigned the value 1, the internal Sheets **QUERY** function's implementation of `MONTH(date_column)` returns a value ranging from 0 to 11. This zero-based indexing system is a crucial detail that requires a mandatory adjustment for accurate month filtering based on the standard calendar view.

To convert this zero-indexed result into the human-readable, one-indexed calendar month (where January = 1 and December = 12), we must simply add **1** to the result of the `MONTH()` function call within the query string. The correct syntax for checking a month number in the `WHERE` clause is therefore always `MONTH(A) + 1 = N`, where 'A' is the column containing the date and 'N' is the desired calendar month number (1 through 12). Failing to include this `+ 1` adjustment will result in the query returning data for the preceding month, leading to inaccurate results and reporting errors.

The following example demonstrates the fundamental syntax used to query for rows that contain a specific month in a designated date column. This formula targets the month February, which corresponds to the calendar month number 2, thus requiring the adjusted index value. This robust approach ensures that your dynamic reporting aligns perfectly with standard chronological expectations.

You can use the following formula to query for rows in [Google Sheets](#) that contain a specific month in a date column:

```
=QUERY(A1:C13, "select A,B,C where month(A)+1=2", 1)
```

This particular query instructs the system to return the values present in columns **A**, **B**, and **C** from the specified range (**A1:C13**). Crucially, the filter condition mandates that the date found in column A must correspond to the second calendar month, achieved by setting `month(A)+1=2` in the `WHERE` clause. This ensures high precision when isolating month-specific data points for analysis.

It is important to emphasize that the integer **2** in the formula represents the target calendar month number, i.e., February. If you wished to filter for data from January, you would substitute 2 with **1**.

Similarly, to query for data belonging to December, the target number would be **12**. This straightforward numerical substitution is all that is required to adjust the filter across the entire year.

Example 1: Query Where Date Contains Specific Month

Let us delve into a practical application by isolating all rows corresponding specifically to February. This is a common requirement in monthly reporting and sales analysis. By applying the zero-indexing compensation discussed earlier, we can construct a precise QUERY function that scans the entire dataset and extracts only the relevant records, streamlining the data review process significantly.

We utilize the range **A1:C13** and specify the columns we wish to return (A, B, and C). The core logic resides in the `WHERE` clause, which ensures that the month extracted from column **A**, after adding the necessary offset of 1, equals 2. This structure is both robust and highly readable, allowing other users to quickly grasp the filtering intent and methodology.

We can use the following query to return all rows where the month in column **A** contains February:

```
=QUERY(A1:C13, "select A,B,C where month(A)+1=2", 1)
```

The following screenshot visually confirms the result of applying this query to a sample dataset:

	A	B	C	D	E
A15	=QUERY(A1:C13, "select A,B,C where month(A)+1=2", 1)				
1	Date	Sales	Refunds		
2	1/1/2022	14	3		
3	1/3/2022	15	2		
4	1/8/2022	19	2		
5	2/1/2022	20	3		
6	2/15/2022	6	6		
7	2/19/2022	9	5		
8	2/24/2022	12	3		
9	3/6/2022	11	3		
10	3/10/2022	15	1		
11	3/29/2022	10	5		
12	4/10/2022	19	0		
13	4/15/2022	8	4		
14					
15	Date	Sales	Refunds		
16	2/1/2022	20	3		
17	2/15/2022	6	6		
18	2/19/2022	9	5		
19	2/24/2022	12	3		
20					
21					
22					

Upon reviewing the output, notice that the four rows returned by the query all contain February in column **A**. This confirms the accuracy and effectiveness of the `month(A)+1=2` condition when filtering date ranges precisely.

Example 2: Query Where Date Contains One of Several Specific Months

Often, data analysis requires filtering based on multiple criteria simultaneously. For instance, you might need to view sales data for specific promotional months like February and April. The **QUERY** language accommodates this requirement elegantly through the use of logical operators, specifically the `OR` operator, allowing for the inclusion of multiple month criteria within a single statement.

To include records from both February and April, we must chain two distinct month criteria using the `OR` operator. The first condition remains `month(A)+1=2` (February), and the second condition is

`month(A)+1=4` (April). The use of `OR` ensures that a row is returned if it satisfies either the first condition or the second condition, providing a comprehensive view of the selected periods without requiring multiple individual queries.

We can use the following query to return all rows where the month in column **A** contains February or April:

=QUERY(A1:C13, "select A,B,C where month(A)+1=2 or month(A)+1=4", 1)

The following screenshot shows how to use this query in practice, demonstrating the combined output of records spanning both targeted months:

A15 fx =QUERY(A1:C13, "select A,B,C where month(A)+1=2 or month(A)+1=4", 1)						
	A	B	C	D	E	F
1	Date	Sales	Refunds			
2	1/1/2022	14	3			
3	1/3/2022	15	2			
4	1/8/2022	19	2			
5	2/1/2022	20	3			
6	2/15/2022	6	6			
7	2/19/2022	9	5			
8	2/24/2022	12	3			
9	3/6/2022	11	3			
10	3/10/2022	15	1			
11	3/29/2022	10	5			
12	4/10/2022	19	0			
13	4/15/2022	8	4			
14						
15	Date	Sales	Refunds			
16	2/1/2022	20	3			
17	2/15/2022	6	6			
18	2/19/2022	9	5			
19	2/24/2022	12	3			
20	4/10/2022	19	0			
21	4/15/2022	8	4			
22						
23						
24						
25						

This method can be extended indefinitely by adding more `OR month(A)+1=N` clauses for every additional month you wish to include in your filtered output, providing maximum flexibility for

periodic reporting and deep dive analysis.

Advanced Filtering: Using Comparison Operators

While equality filters (=) are useful for specific months, the true power of the **QUERY** language lies in its ability to handle comparison operators such as greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=). When combined with the adjusted `MONTH()` function, these operators allow for filtering data spanning ranges of months, rather than just isolated months or tedious chaining of `OR` statements.

For instance, if you needed to retrieve all data entries occurring in the first half of the year (January through June), you would formulate a condition that checks if the adjusted month number is less than or equal to 6. The required query fragment would look like `where month(A)+1 <= 6`. This powerful shorthand negates the need for chaining six separate `OR` clauses, significantly simplifying the formula structure and improving computational efficiency.

Conversely, to filter specifically for the second quarter (April, May, June), you would use a combination of the `AND` and comparison operators: `where month(A)+1 >= 4 and month(A)+1 <= 6`. This technique of defining complex temporal boundaries provides unparalleled control over time-series data filtering, enabling analysts to define boundaries with ease, which is crucial for detailed financial or operational reviews.

Troubleshooting Common Errors: Date Format Importance

A frequent source of frustration when working with date-based queries is encountering errors or receiving blank results when the formula appears syntactically correct. Almost invariably, this issue stems from the underlying data type or the date format of the column being queried. The **QUERY** function requires the data in the specified column (e.g., column A) to be explicitly recognized by Google Sheets as a valid date value, not merely a text string that resembles a date.

If you receive any errors, such as `#VALUE!`, or if the formula returns an empty set despite knowing that data for the specified month exists, the very first step in debugging must be confirming the cell formatting. Text strings, even those formatted by the user to look like dates (e.g., manually typing "2/1/2024"), will not be properly processed by the internal `MONTH()` function, which relies on the numerical serial value of the date for calculation.

To convert the values in your date column to the correct Date format, simply highlight the entire column containing the dates and then click **Format** along the top ribbon. Next, click **Number**, and finally, click **Date**. This action converts the cell contents into the correct numerical serial format that the **QUERY** function is designed to handle, resolving most filtering issues immediately.

Summary and Best Practices for Month Filtering

Successfully filtering data by month number in Google Sheets using the **QUERY** function relies entirely on understanding and correctly applying the zero-indexed output of the internal `MONTH()` operator. By consistently adding `+1` to the month extraction function, you bridge the gap between the system's internal dating mechanism (0-11) and standard calendar numbering (1-12).

To optimize your data processing and avoid common pitfalls, always adhere to these best practices: 1) Verify that your date column is formatted specifically as a Date format before running the query; 2) Remember the `+1` adjustment is non-negotiable for accurate month matching; and 3) Utilize logical operators (`OR`, `AND`) and comparison operators (`>`, `<`) for complex, multi-month filtering requirements. Implementing these strategies will maximize the utility and precision of your time-series analysis within Google Sheets.

The **QUERY** function, particularly when integrated with chronological filters, provides powerful capabilities for analytical tasks. Whether you are generating a quick report for a single month or building a complex dashboard covering multiple quarters, the syntactical method outlined here ensures reliable and reproducible results for all your date-based filtering needs.