How to Use TODAY() in Google Sheets Query for Easy Date-Based Calculations

Authored by

stats writer

December 1, 2025

The ability to dynamically filter and analyze data based on the current system date is a fundamental requirement for effective spreadsheet management. In Google Sheets Query Language, achieving this dynamic behavior requires integrating the native Sheets function, **TODAY()**, directly into the query string. The TODAY() function itself is straightforward; it returns the current date without the time component. When combined with the powerful **QUERY** function, it allows users to perform sophisticated filtering operations, ensuring reports and dashboards always reflect up-to-the-minute data comparisons against the calendar.

While the **TODAY()** function is simple to use in a standard cell, integrating it into the structured syntax of the **QUERY** function presents a unique challenge. The Google Sheets Query Language treats dates not as simple numbers, but as a specific Date Data Type that must be explicitly formatted and declared using the keyword `date` within the query string. Failing to format the date correctly will lead to errors, as the query engine will interpret the date value as a text string instead of a comparable date object. Mastering this integration is essential for automated, time-sensitive reporting within Google Sheets.

This comprehensive guide will walk you through the precise methodology required to use **TODAY()** effectively within your **QUERY** statements. We will explore the necessary helper functions, such as **TEXT()**, that convert the live date into the exact format required by the query engine, and provide practical examples demonstrating how to retrieve rows based on criteria before, after, or equal to the current date. By the end of this tutorial, you will be equipped to build robust, date-aware data filtering systems.

The Requirement for Date Formatting in Query Language

When working with the Google Sheets Query Language (which is heavily influenced by standard SQL), dates are not handled through standard cell references or calculations. Instead, the query engine demands that any date used for comparison--whether fixed or dynamic--must be supplied as a string formatted strictly as 'YYYY-MM-DD'. This rigid requirement means we cannot simply concatenate **TODAY()** directly into the query string, as the native output format of **TODAY()** (which usually conforms to the locale settings, e.g., M/D/YYYY) is incompatible with the engine's expectation.

To bridge this gap between the native output of the TODAY() function and the required format, we must employ the TEXT() function. The **TEXT()** function allows us to take a numerical value, such as a date or time, and convert it into a designated text string using a specified custom format. By setting this format mask to `"yyyy-mm-dd"`, we guarantee that the current date is presented in the precise structure the **QUERY** engine expects when it encounters the `date` keyword.

The combination of these elements forms the fundamental syntax required for integrating dynamic

date filtering. We wrap the entire query command in quotes, then break out of the quotes using the ampersand operator (`&`) to insert the dynamically formatted date string. This approach ensures that when the query is executed, the resulting string passed to the engine is structurally sound, allowing for successful date comparison operations. The basic structure provided below illustrates the conversion of the current date into the query-compliant format.

=QUERY(A1:C9,"select A, B, C where B < date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'", 1)

Decoding the Dynamic Date Query Structure

Understanding the structure of the dynamic date query is essential for troubleshooting and modification. The formula above is composed of three main parts separated by concatenation operators (`&`). The first part is the static beginning of the query string: `"select A, B, C where B < date '"`. This section defines the columns to select and initiates the `where` clause, specifying that column **B** (our hypothetical date column) must be less than the date that follows. Crucially, it includes the `date` keyword and opens the single quote (`'`) that will contain the formatted date value.

The second part is the dynamic core: `TEXT(TODAY(), "yyyy-mm-dd")`. This segment executes two internal functions. First, the TODAY() function retrieves the current date. Second, the TEXT() function immediately converts that date into the mandatory YYYY-MM-DD string format. Because this result is outside the main query quotation marks, it is calculated by Google Sheets first and then inserted as a literal text string into the query.

Finally, the third part closes the query string: `&"'"`. This is a very simple but critical component. It simply concatenates the closing single quote (`'`) back onto the string. This closing quote is necessary because the query engine requires the date value itself to be enclosed in single quotes, following standard SQL syntax rules for string literals. If this closing quote is omitted, the query string will be incomplete, resulting in a parsing error.

Example 1: Filtering Rows Before the Current Date

To demonstrate the utility of this dynamic filtering, consider a common scenario where we need to review historical transactions or completed tasks--any data point that occurred prior to today. Suppose we are managing a dataset that tracks sales performance, including the date of the sale, the store location, and the total sales amount. We want to quickly isolate only those sales that are now considered closed or past their date threshold.

We will use the following dataset, which shows total sales made by two distinct stores across various dates. For the purpose of this example, let's establish that the system's current date is

1/18/2022. Our goal is to retrieve rows where the date in column **B** is strictly earlier than this current date.

	A	B	C	D
1	**Store**	**Date**	**Sales**	
2	A	1/17/22	34	
3	A	1/18/22	39	
4	A	1/19/22	31	
5	A	1/20/22	40	
6	B	1/17/22	23	
7	B	1/18/22	29	
8	B	1/19/22	28	
9	B	1/20/22	27	
10				
11				
12				
13				
14				
15				

The query must specifically look at column **B** (the Date column) and apply the "less than" operator (<) against the dynamically generated current date. By utilizing the previously defined structure, we construct the formula that ensures only historical data is returned. This method is particularly useful for generating backlog reports or archiving data that no longer needs daily visibility.

=QUERY(A1:C9,"select A, B, C where B < date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'", 1)

Upon execution, the Google Sheets Query Language interprets the dynamically generated date string--which in this fixed example would be '2022-01-18'--and compares it against every entry in column B. The resulting output demonstrates perfect adherence to the criteria: only sales records dated before 1/18/2022 are displayed. This immediate and precise filtering capability highlights the power of combining **QUERY** and dynamic date functions.

A11 =QUERY(A1:C9,"select A, B, C where B < date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'",1)

	A	B	C	D	E	F	G
1	**Store**	**Date**	**Sales**				
2	A	1/17/22	34				
3	A	1/18/22	39				
4	A	1/19/22	31				
5	A	1/20/22	40				
6	B	1/17/22	23				
7	B	1/18/22	29				
8	B	1/19/22	28				
9	B	1/20/22	27				
10							
11	Store	Date	Sales				
12	A	1/17/22	34				
13	B	1/17/22	23				
14							
15							
16							
17							
18							

Example 2: Filtering Rows Equal to or After Current Date

In contrast to retrieving historical data, often the primary business requirement is to look forward--to analyze upcoming events, pending sales, or tasks that are due today or in the future. To achieve this, we simply modify the comparison operator within the `where` clause of our Google Sheets Query Language statement. Instead of using the "less than" operator (<), we utilize the "greater than or equal to" operator (>=).

Returning to our sales dataset, we want to isolate sales that are scheduled for today (1/18/2022) or any date following it. This filtering method is ideal for creating dashboards focused on immediate action items or future planning. Maintaining the same initial dataset for context, we can clearly see which records will be included in our forward-looking report.

	A	B	C	D
1	**Store**	**Date**	**Sales**	
2	A	1/17/22	34	
3	A	1/18/22	39	
4	A	1/19/22	31	
5	A	1/20/22	40	
6	B	1/17/22	23	
7	B	1/18/22	29	
8	B	1/19/22	28	
9	B	1/20/22	27	
10				
11				
12				
13				
14				
15				

The adjustment required in the formula is minimal but impactful. We replace `< date` with `>= date`. This instructs the query engine to include the current date represented by **TODAY()** and all subsequent dates. The necessity of using >= instead of just > ensures that any data entry recorded precisely on the current day is captured, which is often crucial for daily operational reporting.

=QUERY(A1:C9,"select A, B, C where B >= date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'", 1)

Executing this revised query produces a result set that exclusively features future or present sales data. If the current date were to change tomorrow (to 1/19/2022), the query would automatically update, reflecting only records from 1/19/2022 onwards, without any manual intervention required. This automated dynamic capability makes the combination of **QUERY** and TODAY() function highly efficient for ongoing reporting needs.

A11 =QUERY(A1:C9,"select A, B, C where B >= date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'",1)

	A	B	C	D	E	F	G
1	**Store**	**Date**	**Sales**				
2	A	1/17/22	34				
3	A	1/18/22	39				
4	A	1/19/22	31				
5	A	1/20/22	40				
6	B	1/17/22	23				
7	B	1/18/22	29				
8	B	1/19/22	28				
9	B	1/20/22	27				
10							
11	Store	Date	Sales				
12	A	1/18/22	39				
13	A	1/19/22	31				
14	A	1/20/22	40				
15	B	1/18/22	29				
16	B	1/19/22	28				
17	B	1/20/22	27				
18							

Advanced Filtering: Working with Date Ranges

While filtering data before or after a single date is powerful, real-world reporting often requires filtering within specific date ranges, such as "this week," "this month," or "the next 30 days." Using **TODAY()** as the anchor point allows us to define these dynamic ranges by simply adding or subtracting numerical days from the function's output before applying the TEXT() function formatting.

To filter data for the next 7 days, for instance, we would need two conditions linked by the `and` operator in the query's `where` clause. The first condition ensures the date is greater than or equal to today, and the second condition ensures the date is less than or equal to the date seven days from now. This requires calculating a future dynamic date using the structure `TODAY() + 7`.

A formula to retrieve all rows that fall within the next seven days, including today, would look structurally similar to this (assuming column B is the date column):

=QUERY(A1:C9, "select A, B, C where B >= date '"&TEXT(TODAY(), "yyyy-mm-dd")&"' and B <= date '"&TEXT(TODAY()+7, "yyyy-mm-dd")&"'", 1)

Notice that we use two separate `date '...'` clauses, each utilizing the `&TEXT(...)&` concatenation method. The first dynamically determines the start date of the range, and the

second dynamically determines the end date of the range, creating a highly flexible filter that updates daily.

Troubleshooting Common Query Date Errors

When attempting to use dynamic dates in the Google Sheets Query Language, users frequently encounter errors related to date parsing. Understanding these common pitfalls can save significant time during development. The most frequent error message is often related to "Unable to parse query string for function QUERY." This usually indicates that the date format supplied to the query engine is incorrect, or that the required keyword `date` was omitted.

One primary mistake is forgetting the exact formatting string. The query engine requires `yyyy-mm-dd`. If a user mistakenly uses `mm-dd-yyyy` or `yyyy/mm/dd`, the query will fail because it cannot recognize the input as a valid Date Data Type string. Therefore, always double-check the arguments within the TEXT() function: `TEXT(TODAY(), "yyyy-mm-dd")`.

Another common issue involves incorrect quote placement during concatenation. Remember that the date string must be wrapped in single quotes within the final query string, meaning the concatenation logic must look like: `date '` (start quote) & `TEXT(...)` & `'` (end quote). If you miss the final closing single quote, the entire query string is grammatically invalid according to the SQL syntax utilized by Google Sheets.

Finally, ensure that the column you are querying (e.g., column B in our examples) is correctly formatted as a date column within Google Sheets. Although the **QUERY** function is robust, if the underlying data in column B is stored as a text string rather than a recognized date serial number, comparison operations involving the `date` keyword will yield unpredictable or incorrect results, as the engine attempts to compare a date object with a non-date text string.

Summary of Best Practices for Dynamic Date Queries

To maximize efficiency and reliability when integrating the TODAY() function into your **QUERY** statements, adhere to the following best practices:

Strict Format Adherence: Always use the `"yyyy-mm-dd"` format mask within the **TEXT()** function to ensure the dynamic date satisfies the stringent requirements of the query engine.

Use the `date` Keyword: Explicitly declare the date comparison using the `date` keyword immediately followed by the single-quoted date string. This signals to the engine that the following value is a Date Data Type.

Verify Concatenation: Ensure meticulous placement of the double quotes (defining the static

parts of the query), the ampersands (concatenating the dynamic parts), and the single quotes (enclosing the date string within the query).

Keep Data Clean: Before querying, verify that the column containing the dates in your source range is properly formatted as a Date field within Google Sheets. Inconsistent data types are the root cause of many filtering failures.

Utilize Relative Offsets: Leverage arithmetic operations (e.g., `TODAY() + 30` or `TODAY() - 7`) within the **TEXT()** function to easily create dynamic date ranges for tasks like weekly reporting, monthly forecasts, or tracking deadlines.

By following these guidelines and utilizing the powerful combination of **QUERY**, **TODAY()**, and **TEXT()**, you gain unparalleled control over time-sensitive data filtering in Google Sheets, allowing for the creation of dynamic, self-updating reports that require minimal maintenance.