# How to Easily Add Months to Dates in Google Sheets

Authored by

**stats writer**

November 30, 2025

Google Sheets is an indispensable tool for data analysis and management, offering a wide array of functions to handle complex calculations. One common requirement in financial modeling, project scheduling, or tracking subscriptions is the need to adjust a specific date by a fixed number of months. While simply adding 30 days might approximate a month, this method fails to account for the varying lengths of months and the complications of leap years, leading to inaccurate results. Fortunately, Google Sheets provides a dedicated and robust function, the **EDATE function**, specifically designed to solve this precise challenge with absolute precision. This function is essential for ensuring that calculations involving monthly intervals, such as finding the maturity date of a loan or the renewal date of a contract, remain accurate regardless of the starting day or month length.

The **EDATE function** serves a straightforward yet powerful purpose: calculating a future or past date that is exactly a specified number of months away from a starting date. It neatly handles all the complex logic related to calendar structure, such as automatically adjusting for month-end issues. For instance, if you start on January 31st and add one month, the function correctly calculates February 28th (or February 29th in a leap year), not March 3rd (which 30 days might yield). This automatic handling of date boundaries makes **EDATE** a cornerstone for accurate time-series analysis within any complex spreadsheet environment. Understanding its syntax and application is crucial for anyone relying on Google Sheets for serious date calculations.

To utilize this functionality effectively, users must understand that the **EDATE function** requires two primary components: the initial starting date and the integer value representing the number of months to be added or subtracted. The returned value is a new date, presented as a serial number in the underlying Google Sheets system, which is then typically formatted by the cell settings to display as a recognizable date format (e.g., MM/DD/YYYY). We will explore the precise structure of this function, examine practical examples demonstrating how to both advance and reverse time using positive and negative month counts, and provide detailed visual aids to ensure complete comprehension of this highly useful tool for managing time-based data.

## Understanding the EDATE Function Syntax and Purpose

The core mechanism for manipulating dates by specific monthly increments in Google Sheets is the **EDATE()** function. This function belongs to the family of date and time functions and is specifically engineered to maintain consistency when calculating future or past dates. It eliminates the need for complex nested formulas involving `IF` statements or manual calculation of day counts, thereby significantly simplifying the process of date arithmetic within the spreadsheet environment. Its design promotes accuracy, particularly when dealing with long-term forecasts or recurring deadlines that span multiple years.

The formula adheres to a very clean and explicit syntax, making it easy to implement even for

novice users. The function signature clearly defines the necessary inputs, ensuring that the calculation is executed properly and returns a predictable result. The output of the function is always a valid date serial number, which guarantees compatibility with other date-based calculations, such as determining the difference between two dates using subtraction, or utilizing other time-based functions like `NETWORKDAYS` or `EOMONTH`. Mastering this basic structure is the first step toward advanced date management in your spreadsheets.

You can use the **EDATE()** function in Google Sheets to quickly add a certain number of months to a date.

This formula uses the following basic syntax:

**EDATE(start_date, months)**

where:

**start_date**: This is the mandatory initial date from which the calculation begins. It must be entered as a valid date format, either directly within quotation marks (e.g., "1/1/2023") or, more commonly, as a cell reference containing a date value.
**months**: This mandatory argument represents the number of months to be added or subtracted. It must be an integer (whole number). Positive values advance the date, while negative values move the date backward in time.

## Detailed Breakdown of EDATE Arguments

Understanding the nuances of the two required arguments, `start_date` and `months`, is essential for accurate formula construction. The `start_date` argument must be recognized by Google Sheets as a date serial value. While we see dates displayed in familiar formats like 'MM/DD/YYYY', the underlying system treats every date as a sequential number counting the days since a baseline date (January 1, 1900, in most spreadsheet applications). If you input a date that the spreadsheet cannot interpret, the **EDATE** function will likely return an error, such as `#VALUE!`, indicating a failure to convert the text input into a valid serial number. Therefore, always reference a cell that already contains a confirmed date value or use the `DATE()` function nested within **EDATE** if constructing the date from scratch.

The second argument, `months`, introduces the critical element of time manipulation. It is crucial that this value is supplied as an integer. Although **EDATE** will often attempt to truncate or round non-integer inputs, relying on whole numbers ensures predictable and consistent results. For instance, if you input 1.5 months, the function will likely treat it as 1 month, leading to a potential miscalculation if the half-month was intended. Furthermore, the sign of this integer dictates the direction of time travel: a value of 12 moves the date forward one year, while a value of -12 moves

the date backward one year. This flexibility allows a single function structure to handle both forecasting and historical analysis.

Consider a scenario where you need to calculate a payment date that is precisely ten months after the start date located in cell A1. The input for `start_date` would simply be `A1`, and the input for `months` would be `10`. This leads to a concise and highly readable formula structure. The following demonstrates the typical implementation when referencing a starting date in a cell:

For example, we can use the following syntax to add 10 months to the date in cell A1:

**=EDATE(A1, 10)**

## Practical Example 1: Adding Positive Months

To demonstrate the functionality of **EDATE** in a live spreadsheet environment, we will walk through an example of calculating future milestones based on a set of initial project dates. This method is exceptionally useful for project managers or HR specialists who must schedule recurring events or track employee anniversary dates. The principle is simple: apply the **EDATE** formula once and then efficiently drag it down the column to calculate the corresponding end dates for all entries in the dataset.

The following example shows how to use this function in practice.

# Example: Add Months to Date in Google Sheets

Suppose we have the following list of dates in Google Sheets, representing various contract initiation dates:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Date** | | | |
| 2 | 1/2/2022 | | | |
| 3 | 1/15/2022 | | | |
| 4 | 2/17/2022 | | | |
| 5 | 2/24/2022 | | | |
| 6 | 2/25/2022 | | | |
| 7 | 3/4/2022 | | | |
| 8 | 3/19/2022 | | | |
| 9 | 4/15/2022 | | | |
| 10 | 5/1/2022 | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |



In this setup, Column A contains our starting dates, often referred to as the `start_date` argument for the **EDATE** function. We aim to calculate three separate projections: the date one month out, the date six months out, and the date fifteen months out. Notice the varying magnitudes of the month count; using 15 months, which crosses a year boundary, highlights the function's ability to seamlessly handle year transitions without manual intervention. The use of cell references (e.g., `A2`) allows the formula to be dynamic, ensuring that if the original start date changes, all projected dates update instantly.

We can use the following formulas in columns B, C, and D to add a specific number of months to the values in the Date column (Column A):

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Date** | **Date + 1 Month** | **Date + 6 Months** | **Date + 15 months** |
| 2 | 1/2/2022 | 2/2/2022 | 07/02/2022 | 04/02/2023 |
| 3 | 1/15/2022 | 2/15/2022 | 07/15/2022 | 04/15/2023 |
| 4 | 2/17/2022 | 3/17/2022 | 08/17/2022 | 05/17/2023 |
| 5 | 2/24/2022 | 3/24/2022 | 08/24/2022 | 05/24/2023 |
| 6 | 2/25/2022 | 3/25/2022 | 08/25/2022 | 05/25/2023 |
| 7 | 3/4/2022 | 4/4/2022 | 09/04/2022 | 06/04/2023 |
| 8 | 3/19/2022 | 4/19/2022 | 09/19/2022 | 06/19/2023 |
| 9 | 4/15/2022 | 5/15/2022 | 10/15/2022 | 07/15/2023 |
| 10 | 5/1/2022 | 6/1/2022 | 11/01/2022 | 08/01/2023 |
| 11 | | =EDATE(A2, 1) | =EDATE(A2, 6) | =EDATE(A2, 15) |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |
| 19 | | | | |
| 20 | | | | |



By examining the output, we can observe precisely how **EDATE** handles calendar complexities. In Column B, which uses the formula `=EDATE(A2, 1)`, the values show the result of adding exactly one month. For instance, 1/31/2023 moves to 2/28/2023, correctly stopping at the last day of February, demonstrating the function's intelligent month-end handling. Similarly, Column C, using `=EDATE(A2, 6)`, provides the six-month projection. Finally, Column D, utilizing `=EDATE(A2, 15)`, clearly shows the results of projecting the date 15 months into the future, crossing into the next calendar year and confirming that **EDATE** performs accurate calculations across year boundaries.

The values in column B show the value of the original date plus one month, using the formula `EDATE(A2, 1)`.

The values in column C show the value of the original date plus six months, using the formula `EDATE(A2, 6)`.

The values in column D show the value of the original date plus 15 months, using the formula `EDATE(A2, 15)`.

## Interpreting Date Formats and Results

A common point of confusion for new users of date functions in spreadsheet software relates to how the results are displayed. When **EDATE** successfully executes, it returns a serial number. If the target cell (where the formula is entered) is formatted as 'General' or 'Number', the output will appear as a large integer (e.g., 44957), which is the number of days elapsed since the spreadsheet epoch date. This raw number is technically the correct result, but it is not user-friendly. It is imperative that the cell containing the **EDATE** formula be formatted specifically as a date to display the result in a conventional calendar format (e.g., 03/15/2024).

Formatting is controlled via the 'Format' menu in Google Sheets, under the 'Number' or 'Date' options. Users have the flexibility to choose various date displays, such as short date (MM/DD/YY), long date (Month Day, Year), or specific ISO standards (YYYY-MM-DD). The calculation itself remains unchanged regardless of the display format, but selecting an appropriate format is essential for clear communication of the results. If you encounter a large number instead of a date after using **EDATE**, simply check and adjust the cell formatting.

Furthermore, the integrity of the **EDATE** calculation lies in its adherence to calendar logic, particularly concerning the day of the month. If the starting day is, say, the 31st, and the resulting month does not have 31 days (e.g., April or September), **EDATE** automatically adjusts the resulting date to the last day of that shorter month. This ensures that the result is a valid date. For example, applying **EDATE(3/31/2024, 1)** yields 4/30/2024, maintaining the intent of moving forward one month while respecting the physical limitations of the calendar. This feature is a key differentiator from simpler date addition methods.

## Practical Example 2: Subtracting Months Using Negative Values

The versatility of the **EDATE function** extends beyond simple future projections; it is equally adept at calculating past dates. This is achieved by utilizing a negative integer for the `months` argument. Calculating past dates is frequently required for historical analysis, determining eligibility periods, or identifying the original start date of a contract given a known renewal date. The mechanical process remains identical to adding months, with the sign of the integer being the only modification required.

Note that you can also use negative numbers to subtract months from a date. This allows for efficient historical tracking within your dataset.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Date** | **Date - 1 Month** | **Date - 6 Months** | **Date - 15 months** |
| 2 | 1/2/2022 | 12/2/2021 | 07/02/2021 | 10/02/2020 |
| 3 | 1/15/2022 | 12/15/2021 | 07/15/2021 | 10/15/2020 |
| 4 | 2/17/2022 | 1/17/2022 | 08/17/2021 | 11/17/2020 |
| 5 | 2/24/2022 | 1/24/2022 | 08/24/2021 | 11/24/2020 |
| 6 | 2/25/2022 | 1/25/2022 | 08/25/2021 | 11/25/2020 |
| 7 | 3/4/2022 | 2/4/2022 | 09/04/2021 | 12/04/2020 |
| 8 | 3/19/2022 | 2/19/2022 | 09/19/2021 | 12/19/2020 |
| 9 | 4/15/2022 | 3/15/2022 | 10/15/2021 | 01/15/2021 |
| 10 | 5/1/2022 | 4/1/2022 | 11/01/2021 | 02/01/2021 |
| 11 | | =EDATE(A2, -1) | =EDATE(A2, -6) | =EDATE(A2, -15) |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |



In the visual example above, we take the same list of starting dates and apply negative month counts: -1, -6, and -15. Column B now uses the formula `=EDATE(A2, -1)`, effectively finding the date one month prior. Column C uses `-6` to look six months into the past, and Column D uses `-15` to rewind the calendar by 15 months, crossing into the previous year. This consistent behavior, governed entirely by the sign of the second argument, underscores the functional elegance of **EDATE**.

The values in column B show the value of the original date **minus** one month, using the formula `EDATE(A2, -1)`.

The values in column C show the value of the original date **minus** six months, using the formula `EDATE(A2, -6)`.

The values in column D show the value of the original date **minus** 15 months, using the formula `EDATE(A2, -15)`.

When subtracting months, users must be mindful of how date boundary conditions are handled. If the starting date is, for instance, March 31st, and you subtract one month, the resulting date will be February 28th (or 29th), as there is no 31st of February. This rule, known as boundary preservation, ensures that the resulting date is logically consistent within the calendar month. If a different behavior is required (e.g., always returning the 31st of the previous month if available),

then a more complex combination of `DATE`, `YEAR`, and `MONTH` functions would be needed, but for standard monthly calculations, **EDATE** provides the most reliable result.

## Common Errors and Troubleshooting EDATE

While the **EDATE function** is straightforward, users may occasionally encounter errors. The most frequent error is the `#VALUE!` error. This error almost always signifies that Google Sheets failed to interpret the `start_date` argument as a valid date serial number. This typically happens if the date was entered manually as text in a format that the sheet's locale settings do not recognize (e.g., trying to input a US-style date format (MM/DD/YYYY) into a locale set for European date formats (DD/MM/YYYY)), or if the referenced cell contains non-date text or an empty string. To resolve this, ensure the starting date cell is correctly formatted as a date, or use the `DATE(year, month, day)` function to explicitly construct the date within the **EDATE** formula.

Another potential issue involves the `months` argument. Although **EDATE** is quite robust, if the month input is extremely large, the resulting date might exceed the maximum date supported by the spreadsheet software, though this is rare in practical applications. More commonly, confusion arises if a user inputs a fractional value (e.g., 3.5). While **EDATE** might process this by truncating the decimal, reliance on non-integer inputs violates the intended use of the function and should be avoided. Always round or truncate your month count externally before feeding it into the formula to ensure clarity and consistency in the calculation.

Finally, be aware of regional settings. Although **EDATE** itself is standardized, the way Google Sheets interprets dates entered directly into cells is heavily dependent on the spreadsheet settings. If you share a sheet across different regions, verify that the date format used for entry is consistent or switch to an explicit formatting method using cell referencing. By systematically checking the data type of the `start_date` and ensuring the `months` argument is a signed integer, troubleshooting the **EDATE** function becomes a simple matter of checking the two inputs.

## Alternatives and Related Date Functions

While **EDATE** is the ideal tool for adding months while preserving the day number (or adjusting to the month end), sometimes different date calculation logic is required. The most closely related function is **EOMONTH()** (End of Month). **EOMONTH** functions identically to **EDATE** in terms of syntax, requiring a `start_date` and a `months` offset. However, **EOMONTH** always returns the last day of the resulting month, regardless of the day of the month provided in the `start_date`. This is invaluable for calculations such as invoicing cycles or period-end closing dates, where the exact day within the month is less important than hitting the final calendar day of the intended period.

For highly customized date addition that involves adding days, months, and years separately, or

when strict day-of-month preservation is necessary even across month-end boundaries (which **EDATE** intentionally violates for calendar logic), users might need to employ a combination of the `DATE`, `YEAR`, `MONTH`, and `DAY` functions. For example, to add 15 months and 10 days, one would need to use `DATE(YEAR(A1), MONTH(A1)+15, DAY(A1)+10)`. While this offers greater control, it requires significantly more complex formula construction and runs the risk of generating invalid dates if the resulting day count exceeds the number of days in the calculated month. Therefore, **EDATE** remains the most efficient solution for purely monthly increments.

## Conclusion and Documentation Reference

The **EDATE function** is a powerful and necessary utility in the Google Sheets arsenal for anyone managing time-sensitive data. Its ability to accurately calculate future or past dates based on monthly offsets, while correctly handling variable month lengths and year transitions, ensures the integrity of financial models, project schedules, and contractual timelines. By adhering to the simple syntax of `EDATE(start_date, months)` and understanding how to utilize positive and negative integers, users can perform complex date arithmetic with minimal effort and maximum reliability.

To ensure ongoing proficiency and to explore advanced functionalities related to date manipulation, users are encouraged to consult the authoritative resources provided by Google. Comprehensive documentation details all edge cases and provides official examples, serving as the ultimate reference for resolving any ambiguities regarding the function's behavior in specific scenarios.

**Note**: You can find the complete documentation for the **EDATE()** function directly on the Google Sheets help center.