Learning to Use the IF Function with Months in Google Sheets

Authored by

Mohammed looti

November 11, 2025

Introduction to Conditional Logic in Google Sheets

Analyzing time-series data often requires filtering information based on specific criteria, such as the month in which a transaction occurred. Google Sheets provides powerful tools for this type of conditional evaluation, primarily through the combination of the IF function and the MONTH function. This approach allows users to automate complex decision-making processes directly within their spreadsheets, ensuring that calculations or values are returned only when specific monthly conditions are met. Mastering this technique is essential for anyone performing rigorous financial analysis, sales tracking, or any form of calendar-based data segmentation.

The fundamental challenge when working with dates in a spreadsheet environment is that a date is typically stored as a serial number, not as a simple text string. For instance, the date 10/15/2023 is not immediately recognizable as "October" by simple formulas. We must therefore extract the numerical component representing the month before applying any conditional checks. The MONTH function serves this precise purpose, transforming a full date serial number into an integer ranging from 1 (January) to 12 (December), making it suitable for direct comparison within a logical test.

Once the month has been successfully isolated as a numerical value, the IF function takes over. The IF function structure is simple yet robust: it evaluates a logical expression and returns one value if the expression is **TRUE** and a different value if the expression is **FALSE**. By nesting the MONTH function inside the logical test of the IF function, we can create precise filters that activate based solely on the month of the year. This powerful combination is the cornerstone of dynamic monthly data analysis.

Deconstructing the Core Formula: IF and MONTH Functions

The formula provided below represents the most efficient way to use conditional logic based on a month in Google Sheets. It is designed to perform a specific action--returning the value from cell **B2**--if the date stored in cell **A2** falls within the tenth month of the year (October). Otherwise, it defaults to returning a zero value.

=IF(MONTH(A2)=10, B2, 0)

To fully appreciate its functionality, we must break down the three primary components of this formula. First, the **logical_expression** is defined by `MONTH(A2)=10`. The MONTH function takes the date from **A2** and extracts its corresponding month number. This resulting number is then compared to **10**. If the month is indeed October, this entire expression evaluates to **TRUE**. Second, the **value_if_true** argument is set to **B2**. If the logical test is **TRUE**, the formula immediately returns the existing value found in cell **B2**, typically representing the relevant metric for that date. Finally, the **value_if_false** argument, which is **0** in this case, dictates the output if the month is

anything other than October. This structure ensures a clean, binary output based purely on the monthly criterion.

Understanding the role of the numerical month designation is crucial. In spreadsheet software, months are universally represented by integers: 1 for January, 2 for February, 3 for March, and so on, up to 12 for December. When we specify **10** in the formula, we are explicitly targeting the month of October, regardless of the year. This standardization simplifies the conditional logic significantly, preventing the need to rely on language-specific text comparisons (e.g., comparing "Oct" or "October"), which can often lead to errors related to formatting or localization.

Practical Application: Filtering Data Based on a Single Month

To illustrate the utility of this conditional filtering, consider a common business scenario involving sales performance tracking. Suppose a company maintains a comprehensive dataset in Google Sheets that records daily sales figures. The management team requires a report that isolates sales made specifically during October, perhaps for end-of-quarter analysis or budgeting purposes. The raw data includes the transaction date in Column A and the corresponding total sales value in Column B, as shown in the initial setup below:

	A	B	C	D
1	**Date**	**Sales**		
2	10/12/2023	5		
3	10/15/2023	6		
4	10/19/2023	19		
5	11/5/2023	14		
6	11/30/2023	18		
7	12/7/2023	13		
8	12/22/2023	10		
9	12/23/2023	12		
10	1/4/2024	12		
11	1/20/2024	7		
12				
13				
14				
15				

The objective is clear: we need to create a third column (Column C) that only displays the sales amount if the date listed in Column A is within October. If the date falls outside of October (e.g.,

September or November), the formula must return a value of zero. This filtering process allows analysts to quickly sum or analyze the October sales figures without manually sifting through data points from other months. This efficiency is paramount when dealing with large volumes of transactional data.

Implementing this solution requires careful placement of the formula. We will begin by typing the conditional formula into cell **C2**, which corresponds to the first row of data. Subsequently, this formula can be rapidly applied to all subsequent rows using the fill handle feature of Google Sheets. This scalability ensures that the analysis remains consistent across the entire dataset. The resulting output in Column C will either mirror the sales value from Column B or display a zero, acting as a perfect binary filter.

Step-by-Step Implementation of the Monthly Filter

To execute the filtering task described above, we insert the combined IF function and MONTH function into cell **C2**. The formula specifies that if the month extracted from the date in **A2** is equal to 10 (October), the cell should return the value in **B2**; otherwise, it returns 0. This precise syntax ensures that the condition is met only by the targeted month.

=IF(MONTH(A2)=10, B2, 0)

After entering the formula into **C2** and dragging it down to apply it to the remaining rows, the results clearly demonstrate the effectiveness of the conditional logic. The following screenshot illustrates the outcome. Notice how the values in Column C only reflect the sales figures when the corresponding date in Column A falls within October. For instance, the sales recorded on 9/15/2023 are correctly excluded and replaced with 0, while sales recorded on 10/1/2023 are preserved.

C2 =IF(MONTH(A2)=10, B2, 0)

	A	B	C	D
1	**Date**	**Sales**	**Sales if Date is October**	
2	10/12/2023	5	5	
3	10/15/2023	6	6	
4	10/19/2023	19	19	
5	11/5/2023	14	0	
6	11/30/2023	18	0	
7	12/7/2023	13	0	
8	12/22/2023	10	0	
9	12/23/2023	12	0	
10	1/4/2024	12	0	
11	1/20/2024	7	0	
12				
13				
14				
15				

The resulting column, in effect, serves as a conditional flag. If the date in Column A is recognized as being in October, the formula successfully returns the associated sales value from Column B. In all other scenarios, where the date belongs to any of the eleven non-October months, the formula executes the **value_if_false** condition, returning a value of zero. This methodology provides a clean, automated way to segment data for monthly reporting, allowing for simple aggregation functions (like `SUM(C:C)`) to calculate the total sales exclusively for October.

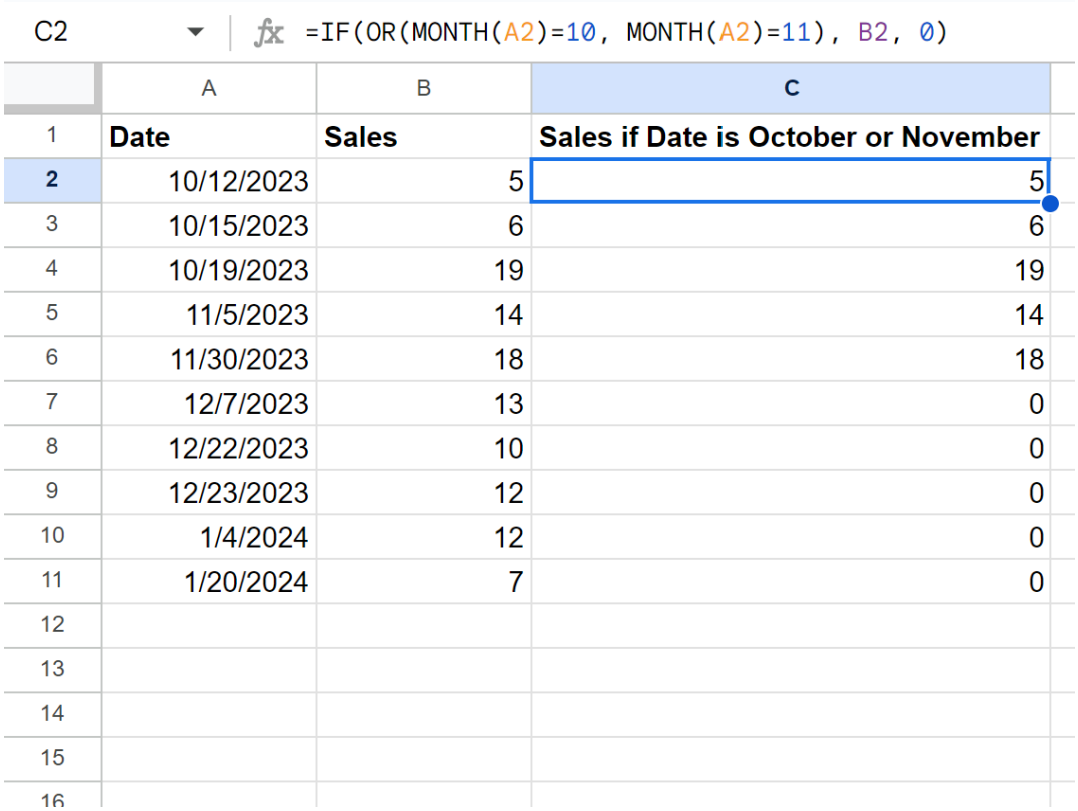
Handling Multiple Conditions Using the OR Function

In many analytical scenarios, the required condition is not limited to a single month but may encompass several months. For example, a business might need to analyze sales data across an entire fiscal quarter, such as Q4, which typically spans October, November, and December. The standard IF function can only handle one single logical test effectively. To incorporate multiple monthly criteria, we must introduce the OR function, which is a fundamental component of Boolean logic.

The OR function allows us to check if **any** of the supplied logical arguments are **TRUE**. If even one condition is met, the OR function returns **TRUE** overall, which then satisfies the logical test of the outer IF function. In the context of monthly filtering, this means we can specify multiple month numbers within the OR function. The following revised formula demonstrates how to check for either October (10) or November (11):

=IF(OR(MONTH(A2)=10, MONTH(A2)=11), B2, 0)

When this extended formula is applied, the output in the results column adjusts to include sales data from both October and November, while still returning zero for all other months. This method is highly flexible; one can easily expand the arguments within the OR function to include December (12) or any other combination of months necessary for the analysis. The screenshot below confirms the successful filtering using the multi-conditional approach, demonstrating that sales from both October and November are now preserved:



	A	B	C
1	**Date**	**Sales**	**Sales if Date is October or November**
2	10/12/2023	5	5
3	10/15/2023	6	6
4	10/19/2023	19	19
5	11/5/2023	14	14
6	11/30/2023	18	18
7	12/7/2023	13	0
8	12/22/2023	10	0
9	12/23/2023	12	0
10	1/4/2024	12	0
11	1/20/2024	7	0
12			
13			
14			
15			
16			

If the date in Column A corresponds to either October or November, the OR function returns **TRUE**, and the resulting cell returns the sales value from Column B. If the date is outside this two-month range, the OR function returns **FALSE**, and the formula defaults to a zero value. This technique is invaluable for constructing reports that rely on evaluating data across non-contiguous or custom timeframes.

Advanced Considerations and Best Practices

While the combination of IF function and MONTH function is robust, analysts should be aware of several best practices and advanced considerations to ensure accuracy and efficiency in their spreadsheets. The primary concern revolves around date formatting. Google Sheets must

recognize the input in Column A as a valid date serial number for the MONTH function to work correctly. If the input is formatted as plain text (e.g., manually typing "Oct 15, 2023" without proper sheet formatting), the MONTH function may return an error or an incorrect value, highlighting the importance of consistent data entry standards.

Another crucial consideration, especially when dealing with multi-year datasets, is the fact that the MONTH function ignores the year entirely. If your dataset spans 2022, 2023, and 2024, the formula `MONTH(A2)=10` will flag October sales from all three years. If the requirement is to filter sales for October **2023 only**, the formula must be extended to include the `YEAR()` function and the `AND()` function. For instance, the logical test would become `AND(MONTH(A2)=10, YEAR(A2)=2023)`, ensuring that both the month and the year conditions are met simultaneously.

Furthermore, for exceptionally large datasets or when performing more complex filtering, an alternative method involves using the **QUERY** function. The **QUERY** function, which utilizes SQL-like syntax, can often be more efficient and readable for extensive conditional filtering. For example, a query equivalent to our simple October filter might look like `=QUERY(A2:B, "SELECT B WHERE MONTH(A) = 9", 0)`. Note that in the **QUERY** function, months are zero-indexed (0 for January, 9 for October), which is a key difference from the standard MONTH function (1 for January, 10 for October). Choosing between the nested IF function approach and the **QUERY** function depends entirely on the size of the data and the complexity of the required conditional filtering.

Additional Resources for Date Management

The ability to conditionally analyze data based on time metrics is fundamental to spreadsheet expertise. For users looking to expand their knowledge beyond the simple month extraction demonstrated here, exploring other time-related functions is highly recommended. Functions such as `DAY()`, `YEAR()`, `WEEKDAY()`, and `EOMONTH()` provide the building blocks necessary for constructing sophisticated date-based reports and dashboards.

The following tutorials explain how to perform other common tasks in Google Sheets: