How to Check if a Date is on a Weekend in Excel

Authored by

stats writer

February 7, 2026

The ability to accurately determine whether a specific date falls on a weekend is a critical requirement for various data analysis and reporting tasks within Excel. Whether managing intricate work schedules, calculating business days for project timelines, or automating payroll processes, identifying Saturdays and Sundays efficiently is paramount. This specialized technique allows users to analyze data that might be intrinsically affected by the concept of non-working days.

This article provides a comprehensive guide to implementing a powerful formula that leverages several core Excel functions to check for weekend status. The central goal is to return a clear indicator, typically a Boolean value or a custom text output, which simplifies the filtering, sorting, and conditional formatting of large date ranges. Understanding this logic is fundamental for anyone seeking to master date manipulation in spreadsheet software.

Checking for Weekend Dates in Excel using Conditional Logic

The Core Function: Identifying Weekend Dates Using Conditional Logic

To reliably determine if a date falls on a Saturday or Sunday, we employ a highly effective combination of three essential logical functions: the WEEKDAY function, the OR function, and the IF function. The central component is the WEEKDAY function, which is responsible for converting a date serial number into a numeric value corresponding to its position within the week.

The standard configuration for the WEEKDAY function in Excel (using the default return type) dictates that the number 1 represents Sunday and the number 7 represents Saturday. Consequently, any date that returns either a 1 or a 7 is definitively classified as a weekend date. This fundamental numbering convention provides the necessary logical input for our conditional evaluation.

You can use the following comprehensive formula in Excel to check if a given date, specifically located in cell **A2**, is on a weekend. This formula is easily adaptable by changing the reference cell to match your dataset:

=IF(OR(WEEKDAY(A2)={1,7}), "Weekend", "Not on Weekend")

Deconstructing the Weekend Detection Formula

This particular formula provides an elegant and concise solution by nesting the key logical checks within the primary conditional structure provided by the IF function. To fully appreciate its implementation, we must break down the operational role of each component, beginning with the

innermost calculation that establishes the day of the week.

First, the expression `WEEKDAY(A2)` executes, returning a number from 1 to 7 based on the date present in cell **A2**. In the default system used here, 1 signifies Sunday and 7 signifies Saturday. Second, the array comparison `={1,7}` checks if the numeric result of the `WEEKDAY` function equals 1 **or** if it equals 7. This array structure simplifies what would otherwise require multiple nested logical checks.

Third, the OR function evaluates the results of this array comparison. If either condition (the day being 1 or the day being 7) is met, the OR function returns a logical TRUE, thereby confirming that the date is a weekend. The outermost IF function then reads this TRUE/FALSE result and assigns the corresponding custom text outputs.

This structure ensures sequential and reliable processing. Based on the logic presented, the formula applied to cell **A2** will return one of the following descriptive texts:

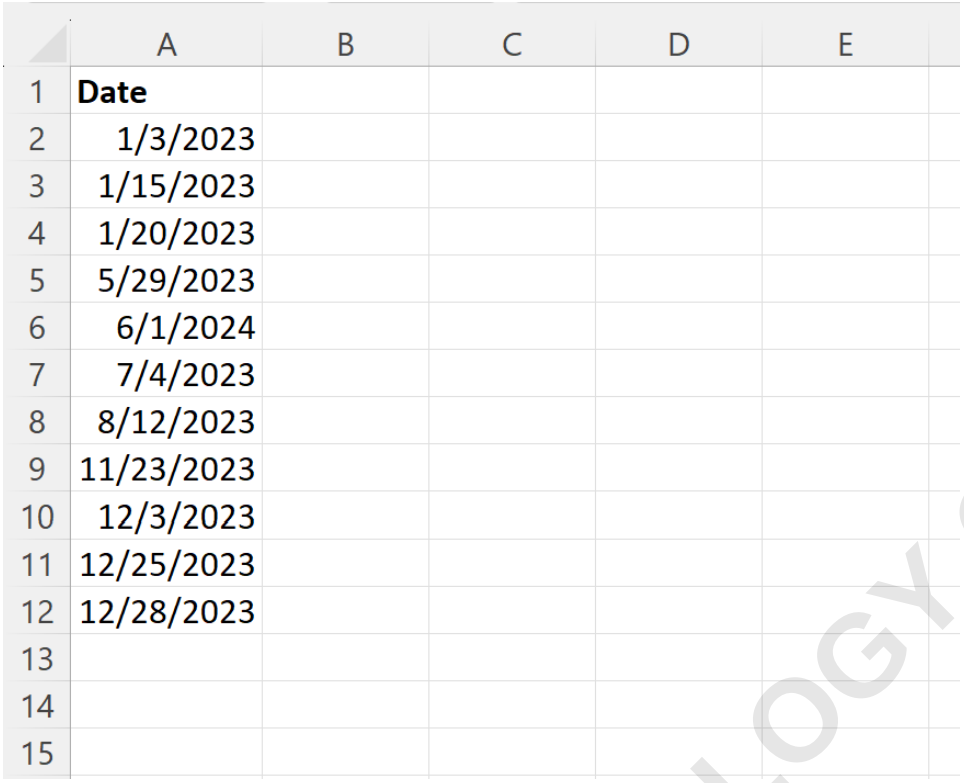
"Weekend"
"Not on Weekend"

Practical Implementation: Step-by-Step Example

Demonstrating the application of this formula within a real-world dataset helps solidify the understanding of the underlying principles. The following example illustrates how to quickly classify an entire list of dates in a spreadsheet using the conditional weekend detection mechanism.

Suppose we are working with a data sheet containing a chronological list of dates in column A, representing scheduled events or attendance records. Our objective is to generate a parallel column, column B, that clearly labels each date as either a "Weekend" or "Not on Weekend."

Consider the following initial list of dates present in Excel, starting from cell A2:



	A	B	C	D	E
1	**Date**				
2	1/3/2023				
3	1/15/2023				
4	1/20/2023				
5	5/29/2023				
6	6/1/2024				
7	7/4/2023				
8	8/12/2023				
9	11/23/2023				
10	12/3/2023				
11	12/25/2023				
12	12/28/2023				
13					
14					
15					

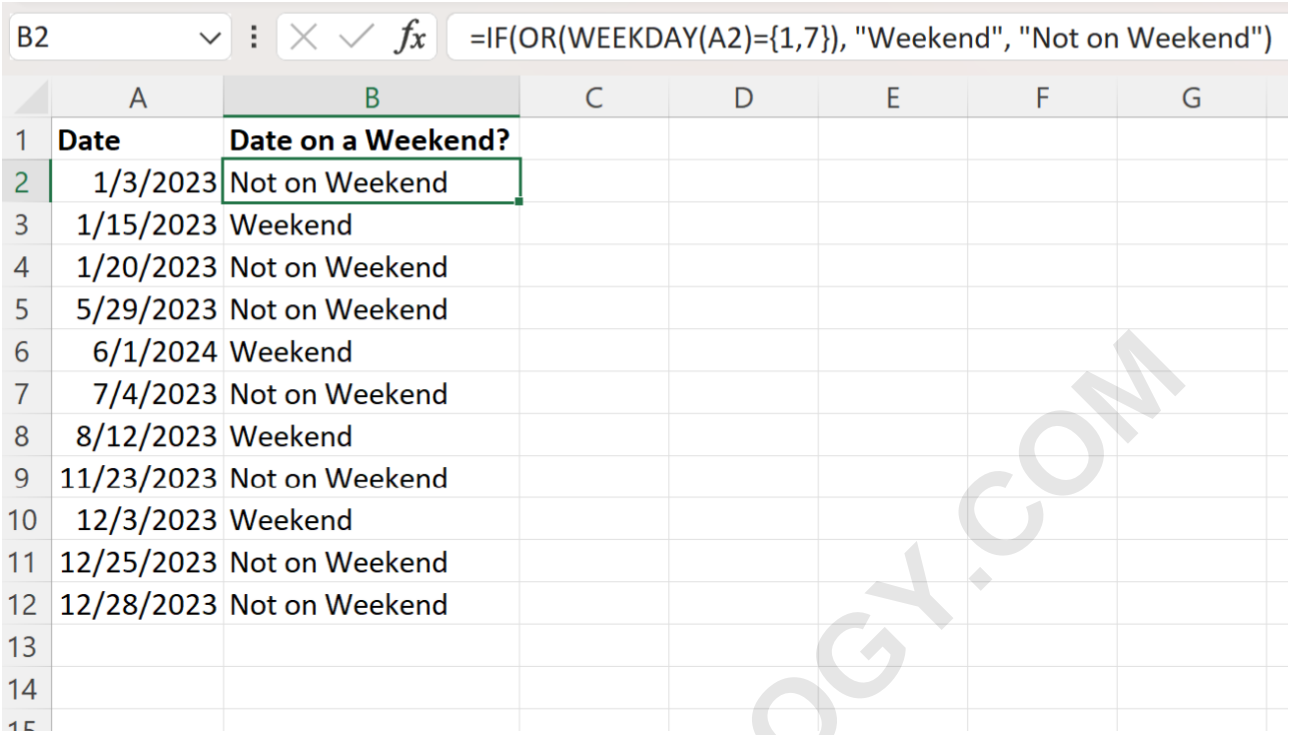
Our first task is to analyze the first date entry in cell A2. We initiate this process by entering the formula into the corresponding cell in the adjacent column, specifically cell **B2**. This formula contains the reference to A2, ensuring that the correct date is evaluated.

Type the following formula precisely into cell **B2**:

=IF(OR(WEEKDAY(A2)={1,7}), "Weekend", "Not on Weekend")

Once the formula is successfully entered into **B2**, the final step involves applying this identical logic across the entire dataset. By utilizing the fill handle (the small square at the bottom-right corner of the selected cell), we click and drag the formula down to encompass every remaining cell in column B, dynamically updating the cell reference for each subsequent row.

After successfully applying the formula across the range, the spreadsheet will appear as shown below, with column B providing the accurate calculated weekend status for every date in column A:



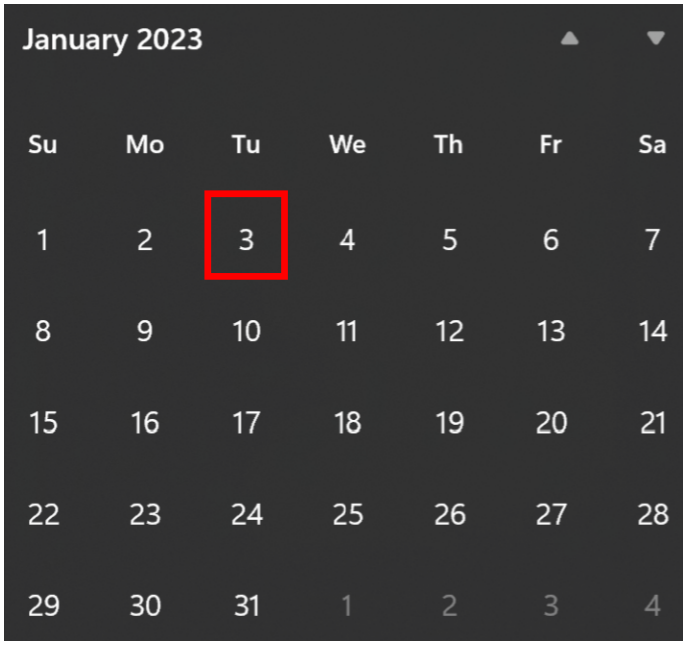
	A	B	C	D	E	F	G
1	**Date**	**Date on a Weekend?**					
2	1/3/2023	Not on Weekend					
3	1/15/2023	Weekend					
4	1/20/2023	Not on Weekend					
5	5/29/2023	Not on Weekend					
6	6/1/2024	Weekend					
7	7/4/2023	Not on Weekend					
8	8/12/2023	Weekend					
9	11/23/2023	Not on Weekend					
10	12/3/2023	Weekend					
11	12/25/2023	Not on Weekend					
12	12/28/2023	Not on Weekend					
13							
14							

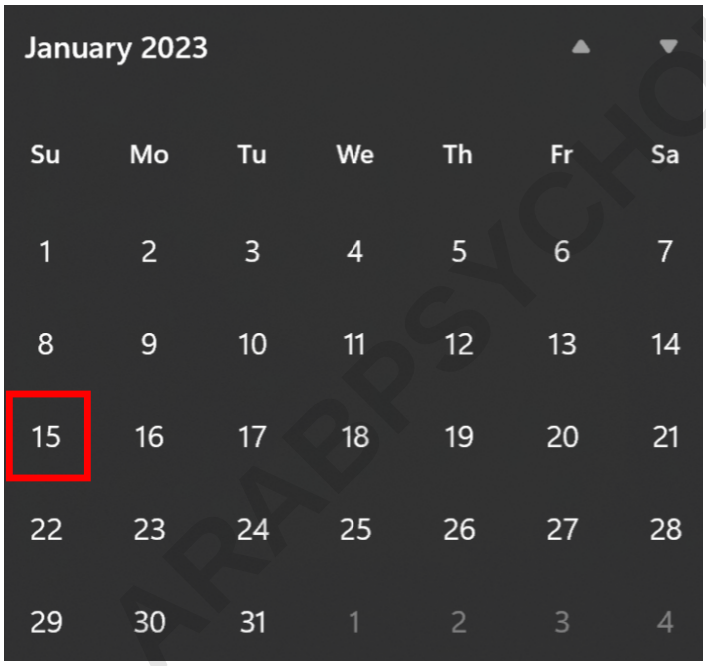
Interpreting the Results and Customizing Output

The calculated data in Column B offers immediate clarity regarding the nature of each corresponding date. This helper column is invaluable for subsequent downstream analyses, such as calculating total working hours, excluding non-operational days from scheduling, or generating filtered reports based strictly on operational timelines.

To ensure the integrity and accuracy of the formula, it is always a best practice to perform a manual verification on a sample of dates against a reliable calendar source. For example, if we examine the date **1/3/2023**, the formula should correctly return "Not on Weekend." We can confirm this finding by consulting a standard calendar reference, which validates its status as a weekday.

The verification process confirms that the logical structure of `WEEKDAY(date) = 1` (Sunday) or `WEEKDAY(date) = 7` (Saturday) is functioning precisely as intended based on the default system configuration:





A significant advantage of using the IF function lies in the customizable nature of the output values. While we utilized the descriptive text strings "Weekend" and "Not on Weekend," users can easily substitute these values based on organizational reporting needs. For instance, returning numerical values (e.g., 1 for weekend, 0 for weekday) can facilitate direct summation or averaging in further calculations. If a raw Boolean value (TRUE or FALSE) is preferred, the user can simplify the formula by removing the IF function wrapper and using only the logical test: `=OR(WEEKDAY(A2)={1,7})`.

Understanding WEEKDAY Return Types and System Flexibility

The precision of our weekend detection formula is entirely dependent on the correct configuration of the WEEKDAY function. This function accepts two key arguments: the mandatory **date serial number** and an optional **return_type** argument. The **return_type** parameter is crucial, as it determines which day of the week is assigned the starting numeric value and, consequently, which numbers represent the weekend.

In the standard formula `=IF(OR(WEEKDAY(A2)={1,7}), ...)`, we relied on the default behavior, where the **return_type** is implicitly 1. Under this widely used setting, the numeric assignments designate Sunday as 1 and Saturday as 7. Thus, checking for 1 or 7 correctly identifies the standard weekend.

However, if your regional or organizational standards dictate a different start to the week--for example, treating Monday as the first day of the week--you must adjust the **return_type**. Using **return_type** 2 sets Monday as 1 and Sunday as 7. If this type is used, the weekend values shift to 6 (Saturday) and 7 (Sunday), necessitating a modification of the formula to `=IF(OR(WEEKDAY(A2, 2)={6,7}), "Weekend", "Not on Weekend")`. Selecting the correct **return_type** parameter is non-negotiable for maintaining the computational accuracy of the weekend check across different locales.

Alternative Methods for Weekend Detection

Although the combined use of WEEKDAY function, OR function, and IF function provides a flexible and direct solution, Excel offers alternative functions that are useful, particularly when the calculation involves counting or scheduling. The `NETWORKDAYS.INTL` function, while primarily designed to calculate the number of working days between two dates, can be skillfully adapted for basic weekend identification.

The advantage of `NETWORKDAYS.INTL` is its ability to handle non-standard weekends (e.g., Friday/Saturday or only Sunday). By checking the number of working days between a date and itself, and specifying the standard Saturday/Sunday weekend rule (code 1), the function returns 1 if it is a weekday and 0 if it is a weekend. This provides a numerical basis for the check: `=IF(NETWORKDAYS.INTL(A2, A2, 1)=0, "Weekend", "Not on Weekend")`.

The application of `NETWORKDAYS.INTL` is generally recommended in situations where the organizational definition of a weekend needs to be flexible or compliant with international standards, as it features a robust custom weekend code system. However, for straightforward verification of standard Saturday and Sunday dates, the traditional `WEEKDAY` formula remains the most common and easily understood approach.

Troubleshooting and Best Practices

Successful implementation of date validation formulas requires adherence to several best practices to mitigate common errors. Firstly, it is critical to ensure that the cells containing the dates are correctly formatted as actual Excel dates. If the dates are inadvertently stored as text strings, the WEEKDAY function will fail, resulting in a `#VALUE!` error. Utilizing diagnostic functions like `ISDATE` or simple format checks can confirm data integrity before complex formulae are applied.

Secondly, if you utilize a non-default **return_type** setting (e.g., using option 2 or 3), strict consistency is paramount. Mixing systems--for instance, using `WEEKDAY(date, 1)` in one calculation and `WEEKDAY(date, 2)` in a dependent calculation--will lead to discrepancies in your weekend criteria (the numeric array) and generate inaccurate results across your linked spreadsheets. Always standardize the definition of the week's start day across all relevant documents.

Finally, it must be acknowledged that this specific formula only accounts for Saturday and Sunday based on their day-of-week index. It does **not** account for public or statutory holidays. If your definition of a non-working day must include holidays, you will need to escalate to more complex functions such as `WORKDAY.INTL` or `NETWORKDAYS.INTL`, which incorporate an optional third argument for specifying a range containing a list of holiday dates. This integration provides a significantly more robust, albeit more involved, method for operational scheduling.

The following tutorials explain how to perform other common operations in Excel: