How to Calculate Working Days Between Dates in Power BI

Authored by

stats writer

January 26, 2026

Calculating the number of working days between two dates in Power BI is a common requirement for project management, financial analysis, and operational reporting. While simple date arithmetic using the DATEDIFF function can determine the total elapsed time, it fails to account for non-working days such as weekends and holidays. To accurately reflect true operational time, we must employ specialized functions within the DAX (Data Analysis Expressions) language.

The most efficient tool for this purpose is the NETWORKDAYS function, which is specifically engineered to exclude specified non-working periods. By mastering this function, analysts can provide precise operational metrics crucial for business intelligence dashboards and reports, ensuring that metrics like cycle time and project duration are based strictly on available work hours, not calendar time.

Power BI: Calculate Working Days Between Two Dates

Introduction: Understanding Date Calculations in Power BI

In Power BI, accurate time intelligence is paramount for measuring performance indicators like cycle time, delivery speed, or project duration. Simply subtracting a start date from an end date provides the total calendar days, which often misrepresents the actual time spent working. For example, a project spanning 10 calendar days might only involve 7 working days due to weekends. The DAX engine offers several functions to handle date calculations, but when the requirement is strictly focused on operational days, the standard arithmetic methods fall short.

The core challenge lies in defining what constitutes a **working day**. Most organizational metrics rely on a standard Monday-to-Friday schedule, excluding Saturday and Sunday. However, some businesses operate on different schedules (e.g., retail, healthcare), or must account for nationally recognized holidays. Therefore, any robust calculation solution must be flexible enough to handle these variations, allowing for custom weekend definitions and the exclusion of specific holiday dates. This necessity leads us directly to the powerful, yet often underutilized, NETWORKDAYS function in DAX.

The Essential NETWORKDAYS Function in DAX

The NETWORKDAYS function is the primary tool for solving this specific time intelligence problem. It calculates the number of whole working days between two dates, automatically excluding Saturday and Sunday by default, mirroring standard business operations. Unlike the original Excel function from which it draws inspiration, the DAX version offers superior integration within the data model environment, allowing it to leverage established date relationships and calculated tables.

To utilize this function, you must provide, at minimum, two arguments: the **start date** and the **end**

date. These dates typically reference columns within your existing tables. The function returns an integer representing the count of working days, inclusive of the start and end dates if they fall on a working day. Understanding the syntax is critical for successful implementation in a Calculated Column or a Measure.

You can use the following standard syntax in DAX to calculate the number of working days between two dates in Power BI:

Working Days Between = NETWORKDAYS(my_data, my_data)

This particular example demonstrates the creation of a new calculated column named **Working Days Between**. This column efficiently determines the operational duration by calculating the difference between the date found in the **Start Date** column and the date in the **End Date** column within the specified table, here referred to as **my_data**.

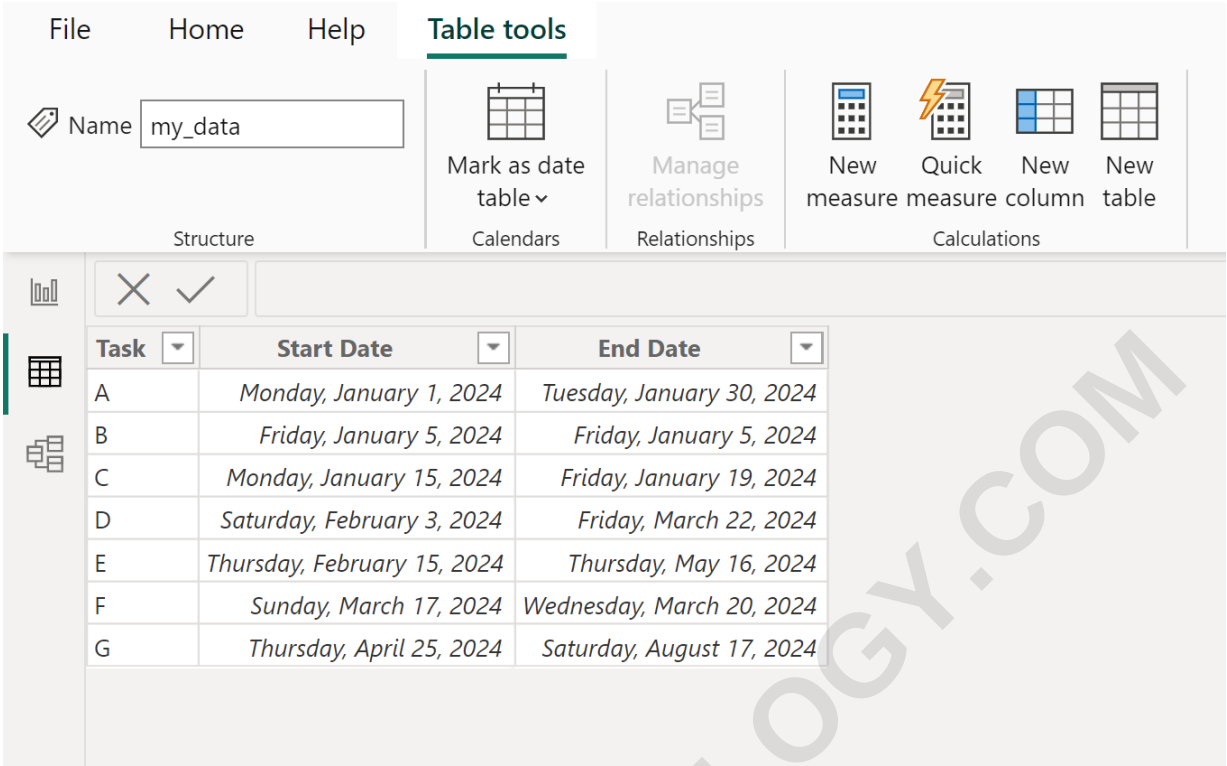
It is important to note that by default, the simple application of the **NETWORKDAYS** function assumes that only **Monday through Friday** are considered operational working days. Weekends, defined as Saturday and Sunday, are automatically excluded from the count. While this default setting covers the majority of business scenarios, we will later explore how to override this behavior using optional arguments to accommodate non-standard work weeks.

Setting Up Your Data Model for Success

Before deploying the DAX formula, ensuring your underlying data model is structured correctly is the foundational step. The calculation relies on having clean, date-formatted columns representing the beginning and end points of the measured period. In project management, this might be a task's Scheduled Start Date and Actual Completion Date. If your data contains null values or improperly formatted text strings in these date columns, the NETWORKDAYS function will return errors or inaccurate results.

The example scenario below uses a simple fact table called **my_data**. This table contains detailed records for various company tasks, including the date when the task began (**Start Date**) and the date when the task concluded (**End Date**). Both of these columns must be correctly designated as Date or Date/Time data types within the Power BI Query Editor before any calculations are performed. A robust data quality check at this stage prevents downstream calculation failures.

Suppose we have the following table in Power BI named **my_data** that contains information on the start date and end date for various tasks at some company. This represents the source data we will use for our calculation:



Task	Start Date	End Date
A	Monday, January 1, 2024	Tuesday, January 30, 2024
B	Friday, January 5, 2024	Friday, January 5, 2024
C	Monday, January 15, 2024	Friday, January 19, 2024
D	Saturday, February 3, 2024	Friday, March 22, 2024
E	Thursday, February 15, 2024	Thursday, May 16, 2024
F	Sunday, March 17, 2024	Wednesday, March 20, 2024
G	Thursday, April 25, 2024	Saturday, August 17, 2024

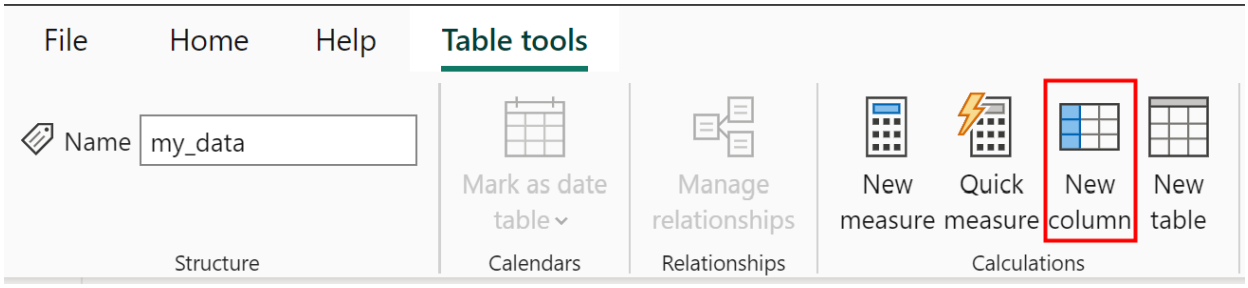
Our objective is to create a new derived column that precisely captures the number of operational days separating the two dates shown. This new column will enhance the table's utility by providing an immediate, actionable metric without needing to resort to iterative calculation methods or complex calendar tables (unless custom holidays are required).

Step-by-Step Implementation: Creating the Calculated Column

To perform this calculation, we must introduce a new attribute into the **my_data** table using a Calculated Column. Unlike a Measure, which aggregates results dynamically based on context, a Calculated Column computes a row-level result and stores it within the Data Model. This is the ideal approach when the required value (the number of working days) is static for each unique row of data.

The process begins by accessing the modeling tools within Power BI Desktop. In the table view, locate the necessary functions to add new data elements.

To do so, click the **Table tools** tab in the ribbon and then click the **New column** icon:



Once the formula bar appears, it is ready to receive the DAX expression. Carefully type the formula, ensuring correct referencing of the table and column names. Case sensitivity is generally not an issue for function names, but best practice dictates precise naming conventions for readability and maintenance.

Then type the following formula into the formula bar:

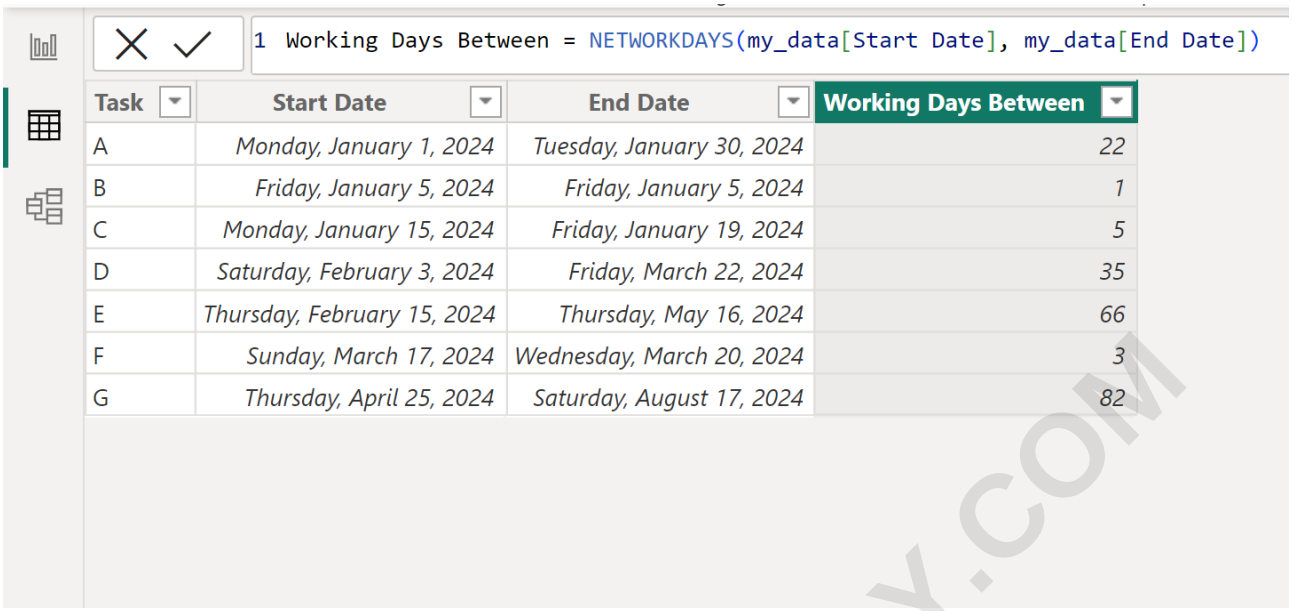
Working Days Between = NETWORKDAYS(my_data, my_data)

Upon committing the formula, the Power BI engine calculates the result for every row in the **my_data** table. This calculation utilizes the default weekend setting (Saturday/Sunday exclusion). The newly created column immediately appears in your table view and fields list, ready for use in visuals and further calculations.

Interpreting and Validating the Results

The successful application of the formula yields a new column named **Working Days Between**. This column provides an immediate, quantifiable measure of the time elapsed strictly according to the working week definition. Reviewing the results against the calendar confirms the function's accuracy, particularly how it skips over weekend days.

This will create a new column named **Working Days Between** that contains the calculated number of working days between the dates in the **Start Date** and **End Date** columns:



```
1 Working Days Between = NETWORKDAYS(my_data[Start Date], my_data[End Date])
```

Task	Start Date	End Date	Working Days Between
A	Monday, January 1, 2024	Tuesday, January 30, 2024	22
B	Friday, January 5, 2024	Friday, January 5, 2024	1
C	Monday, January 15, 2024	Friday, January 19, 2024	5
D	Saturday, February 3, 2024	Friday, March 22, 2024	35
E	Thursday, February 15, 2024	Thursday, May 16, 2024	66
F	Sunday, March 17, 2024	Wednesday, March 20, 2024	3
G	Thursday, April 25, 2024	Saturday, August 17, 2024	82

Examining the output confirms specific date range behaviors. For instance, a duration covering January 1st to January 30th, 2024, includes four full weekends (8 non-working days), resulting in 22 working days. When the start and end dates are identical and fall on a working day, the count is correctly 1. If the same start and end dates fall on a weekend, the result is 0.

The resulting values demonstrate the effectiveness of the calculation based on the default Monday-Friday work week:

There are **22** days between 1/1/2024 and 1/30/2024.
There is **1** working day between 1/5/2024 and 1/5/2024.
There are **5** working days between 1/15/2024 and 1/19/2024.

This precise calculation is vital for performance reporting, enabling businesses to report on **operational efficiency** rather than misleading calendar duration.

Advanced Usage: Customizing Weekends and Holidays

While the basic syntax of NETWORKDAYS is useful, real-world business requirements often necessitate customization. The DAX function provides optional arguments that allow users to define non-standard weekends and incorporate specific corporate or public holidays into the exclusion criteria. This capability ensures that the calculation accurately reflects diverse global or industry-specific work schedules.

The optional third argument, designated as the **Weekend** parameter, uses a numerical code to specify which days are considered non-working. For example, a value of 1 (the default) indicates Saturday and Sunday are excluded. However, a value of 11 indicates only Sunday is excluded,

suitable for industries operating six days a week, or a value of 7 indicates Friday and Saturday are excluded, common in certain Middle Eastern countries. Utilizing this parameter allows the calculation to be tailored precisely to the company's operating rhythm.

Furthermore, to exclude specific public holidays--a critical factor in accurate project timelines--the optional fourth argument, **Holidays**, must be used. This argument requires a separate, dedicated table containing a single column of holiday dates. This holiday table must be defined and loaded into the Data Model, and the reference must be the entire column containing the dates. For instance, if your holiday table is named **Holiday_Dates** and the column is , the extended formula would look like this:

Working Days Custom = NETWORKDAYS(my_data, my_data, 1, Holiday_Dates)

This enhanced syntax ensures that not only are weekends (defined by the third argument, here defaulted to 1 for Sat/Sun) excluded, but any date listed in the **Holiday_Dates** table is also subtracted from the total working day count. This level of detail elevates the reliability of time intelligence reporting in Power BI significantly.

Ensuring Data Quality and Reliability

The reliability of the NETWORKDAYS function output is directly dependent on the quality of the input data. Analysts must vigilantly check for potential data anomalies that could skew results. Common issues include non-date values in the start or end columns, dates that have been entered in reverse order (End Date before Start Date), or incomplete holiday lists.

If the End Date is chronologically earlier than the Start Date, the NETWORKDAYS function will return a **negative number**, indicating that the duration flows backward in time. While mathematically correct, this usually signifies an error in the source data entry. Robust dashboard design should include checks or conditional formatting to flag negative results, alerting users to data quality issues that require resolution in the source system.

For long-term analysis, maintaining the separate Holiday Table is crucial. This table should be dynamic and updated annually or whenever new public holidays are announced. Since the holiday dates are passed as a column reference, any changes made to the source data of the holiday table will automatically be reflected in the Calculated Column upon data refresh, ensuring that historical and future calculations remain accurate without manual DAX modification.

Conclusion and Further Resources

Mastering the calculation of working days between two dates using the NETWORKDAYS function is an essential skill for any serious Power BI practitioner. It provides an immediate, clean, and

flexible method to move beyond simple calendar calculations and deliver meaningful operational metrics. Whether using the default weekend settings or implementing customized holiday calendars, this function ensures that your time intelligence reports are highly accurate and actionable.

By focusing on clean data inputs, understanding the optional parameters for weekends and holidays, and utilizing the formula as a Calculated Column, you can efficiently enrich your Data Model. We encourage readers to explore the broader applications of DAX date functions to further refine complex time-based analysis.

The following tutorials explain how to perform other common tasks in Power BI: