How to Calculate Days Between a Date and Today in Power BI

Authored by

mohammed looti

January 10, 2026

Calculating the elapsed time between a specific historical point and the current moment is a fundamental requirement in business intelligence. In Power BI (Business Intelligence), this calculation is efficiently handled using powerful time intelligence functions built into the DAX (Data Analysis Expressions) language. Specifically, determining the number of days that have passed since an event--such as a sales transaction, project start date, or maintenance activity--until today requires the precise integration of two key functions: the DATEDIFF function and the TODAY() function.

The primary mechanism relies on the DATEDIFF function, which is engineered to compute the quantitative difference between two distinct dates, returning the result in a specified interval unit, such as years, months, or, critically for this scenario, days. When structuring the measure or calculated column to address the user's requirement, the specific date column from the dataset serves as the designated start date. Conversely, the dynamic end date is always provided by the TODAY() function, which perpetually retrieves the system's current date at the moment of data refresh or query execution. This pairing ensures that the calculated metric--the number of days--is constantly up-to-date, thereby facilitating dynamic, real-time analysis and visualization of time-based metrics within the Power BI environment.

The Core Tool: Understanding the DATEDIFF Function in DAX

The DATEDIFF function is arguably one of the most essential components of time intelligence operations within DAX. Its structure is straightforward yet incredibly versatile, requiring three mandatory arguments: the starting date, the ending date, and the interval unit requested for the calculation. When calculating the difference in days, we specify the interval as **DAY**. This function provides a robust alternative to complex manual date arithmetic, ensuring accuracy even when dealing with leap years or varying month lengths, which can often complicate calculations if done using simple subtraction of date serial numbers.

To effectively calculate the time elapsed until the current moment, the specific date column containing the historical data must be designated as the **start_date** parameter. For instance, if tracking inventory age, the 'Inventory Date' column would be the starting point. The absolute key to dynamic calculation lies in correctly defining the **end_date**. By using the TODAY() function as the end date, we instruct Power BI to dynamically substitute the current system date, making the resulting calculated column or measure automatically adjust every time the report is viewed or refreshed.

The resulting calculation provides vital context for metrics such as lead time, aging reports, or measuring the duration since a customer's last interaction. Utilizing the DATEDIFF function correctly ensures that your reports display accurate, real-time insights, transforming static date differences into powerful operational metrics.

The Dynamic Element: Integrating TODAY() for Real-Time Metrics

While DATEDIFF handles the arithmetic of subtraction, the TODAY() function provides the essential dynamic component necessary for calculating the days elapsed "until today." Unlike using a fixed date value (which would become outdated immediately), TODAY() is a volatile function; it does not require any input parameters and reliably returns the current date based on the operating system's clock at the time of data processing or report interaction.

It is crucial to differentiate TODAY() from the **NOW()** function in DAX. While **NOW()** returns both the current date and the precise time, TODAY() returns only the date component (time is set to 12:00:00 AM). Since we are interested in calculating the difference in whole days, using TODAY() simplifies the calculation and avoids potential fractional results or complexities related to time zones if the underlying date column lacks a time component.

By embedding TODAY() directly into the DATEDIFF formula, we create a truly adaptive calculation. This method is the cornerstone of generating effective aging analyses, where the time elapsed must reflect the most current state of the business data. The automatic update feature eliminates the need for manual adjustment, making the Power BI report sustainable and reliable over long periods.

DAX Syntax for Calculating the Day Difference

The following syntax demonstrates the standard and efficient way to calculate the number of days between a date recorded in a data table and the current day within Power BI using a calculated column definition in DAX. This formula is designed to execute row by row, providing a specific difference for every entry in your dataset.

You can use the following syntax in DAX to calculate the number of days between a date and today in Power BI:

Difference = DATEDIFF('my_data', TODAY(), DAY)

This particular example creates a new column named **Difference** that contains the number of days between the date found in the column of a table explicitly named **my_data** and the current date derived from the TODAY() function. It is imperative that the table and column names used in the formula accurately reflect your data model structure. Misspelling either the table name (e.g., **'my_data'**) or the column name (e.g.,) will result in a validation error within the Power BI formula bar, preventing the column creation.

The result of this calculation, the **Difference** column, will be a whole number, representing the count of full days elapsed. If the date in the column is later than today's date, the result of the

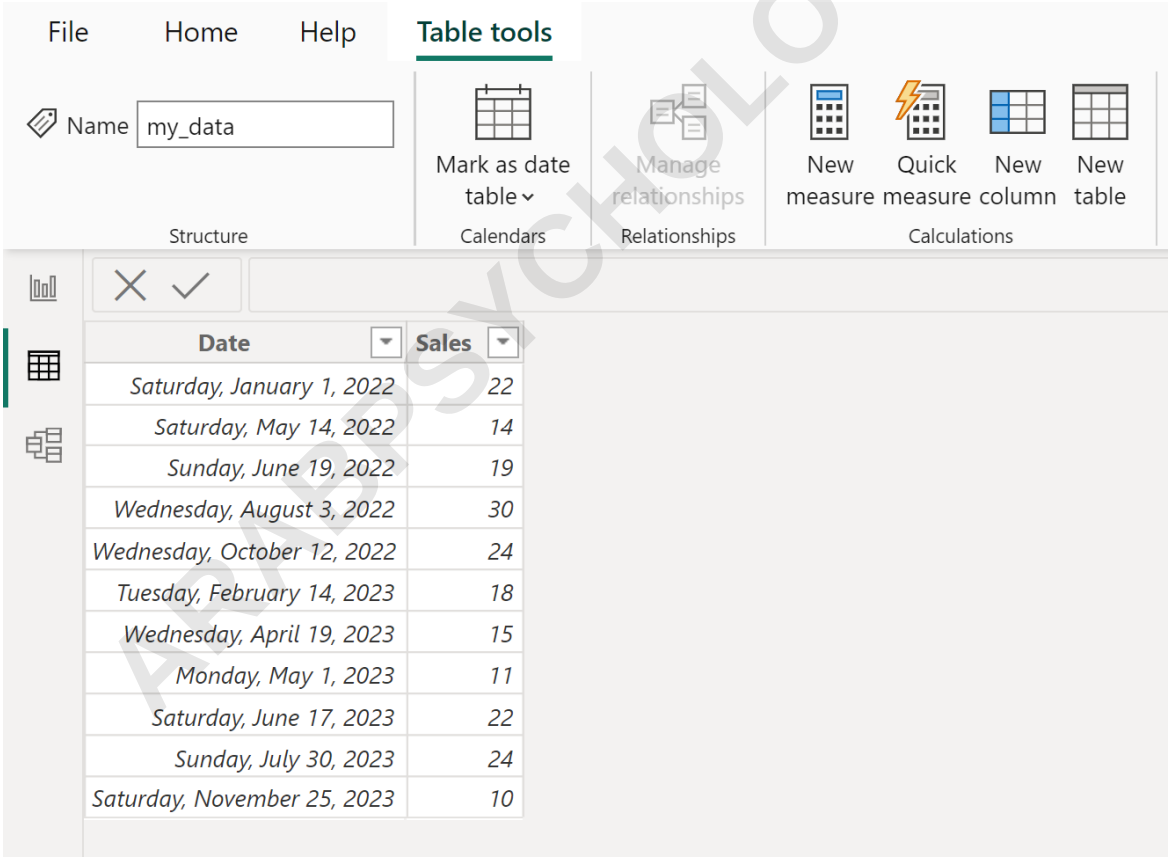
calculation will be a negative number, indicating how many days into the future that date lies. This inherent capability allows for advanced scenario planning and tracking future milestones.

Practical Walkthrough: Calculating Days Since Sales Date

To illustrate this concept clearly, we will walk through a common scenario in business intelligence: calculating the aging of sales records. Suppose we are working with a data table in Power BI named **my_data**. This table documents total sales figures recorded on various historical dates. Our objective is to determine exactly how many days have passed since each recorded sale.

The following table snippet represents the raw data we start with in our **my_data** table. Notice the **Date** column, which contains the specific historical dates we need to measure against the present.

Suppose we have the following table in Power BI named **my_data** that contains information about the total sales made on various dates at some company:



Date	Sales
Saturday, January 1, 2022	22
Saturday, May 14, 2022	14
Sunday, June 19, 2022	19
Wednesday, August 3, 2022	30
Wednesday, October 12, 2022	24
Tuesday, February 14, 2023	18
Wednesday, April 19, 2023	15
Monday, May 1, 2023	11
Saturday, June 17, 2023	22
Sunday, July 30, 2023	24
Saturday, November 25, 2023	10

For the purposes of this specific demonstration and example, let us assume that the date this article is being processed and the corresponding calculation is executed is **1/8/2024**. All subsequent calculated differences will be referenced against this specific date anchor provided by the TODAY() function.

This article is being written on **1/8/2024**.

The next step involves leveraging Power BI's data modeling tools to introduce the new calculated column that will hold our time intelligence result.

Step-by-Step Implementation: Creating the Calculated Column

Implementing the DAX formula requires navigation within the Power BI Desktop interface, specifically within the Data View or the Report View, ensuring that the target table (**my_data**) is selected. The process begins by explicitly telling Power BI that we intend to add a new, row-context-specific column.

Suppose that we would like to create a new column that contains the number of days between the dates in the **Date** column and today's date.

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

Once the **New column** option is selected, the DAX formula bar will become active. This is where the calculation logic is defined. It is crucial to enter the formula precisely as intended, ensuring correct capitalization and proper referencing of table and column names, as DAX is sensitive to syntax errors, although not strictly case-sensitive for function names like DATEDIFF.

Then type the following formula into the formula bar:

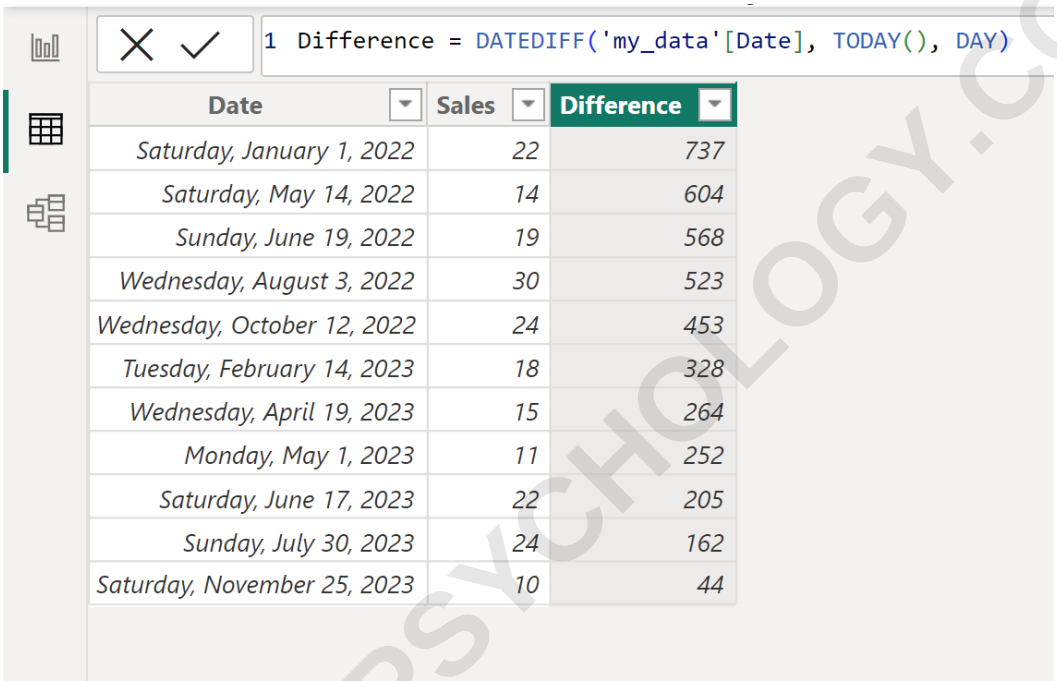
Difference = DATEDIFF('my_data', TODAY(), DAY)

After confirming the formula by pressing Enter, Power BI executes the calculation across all rows in the **my_data** table. For each row, it retrieves the specific date, compares it to the current date (1/8/2024 in this example), and returns the integer difference. The result is a new, populated column named **Difference**, ready for analysis.

Interpreting the Results and Practical Applications

Upon successful execution of the DATEDIFF formula, the **my_data** table is augmented with the **Difference** column, visually confirming the time elapsed in days for each transaction. This immediate numerical representation transforms raw dates into actionable metrics, providing an instant view of data aging relative to the current moment.

This will create a new column named **Difference** that contains the number of days between the dates in the **Date** column and today's date of 1/8/2024:



Date	Sales	Difference
Saturday, January 1, 2022	22	737
Saturday, May 14, 2022	14	604
Sunday, June 19, 2022	19	568
Wednesday, August 3, 2022	30	523
Wednesday, October 12, 2022	24	453
Tuesday, February 14, 2023	18	328
Wednesday, April 19, 2023	15	264
Monday, May 1, 2023	11	252
Saturday, June 17, 2023	22	205
Sunday, July 30, 2023	24	162
Saturday, November 25, 2023	10	44

From the resulting output table, we can draw the following specific conclusions based on the reference date of 1/8/2024:

The sale recorded on **1/1/2022** occurred **737** days prior to today's date of 1/8/2024. This long duration might signify aging sales that require specific attention or categorization in historical analysis.
The transaction dated **5/14/2022** is exactly **604** days old as of 1/8/2024. Such metrics are vital for calculating time-based thresholds, perhaps flagging accounts that haven't been active in over 600 days.
The more recent sale on **6/19/2022** has an age of **568** days relative to today's date. Analyzing the pattern of these differences provides immediate insight into the frequency and recency of activities captured in the dataset.

The practical applications of this calculated column are vast. It enables the creation of dynamic

time filters (e.g., "Sales older than 365 days"), the calculation of moving averages based on recency, and robust segmentation of data into aging buckets (e.g., 0-30 days, 31-90 days, 90+ days). This simple DATEDIFF calculation forms the basis for sophisticated time intelligence dashboards.

Common Pitfalls and Troubleshooting Date Functions

While the DATEDIFF and TODAY() combination is powerful, users often encounter specific issues, primarily stemming from data type conflicts or context transitions in DAX. The most critical requirement is that the column referenced (e.g., **my_data**) must be explicitly defined as a **Date** data type within the Power BI data model. If the column is incorrectly formatted as text or general number, DAX will fail to recognize it as a valid date parameter, resulting in an error, regardless of how clean the data appears visually.

Another common issue arises when attempting to use this calculated column in a measure context where aggregation is required. If you want to calculate the average age of all sales, you must wrap the column calculation inside an iterator function like **AVERAGEX** to handle the row context correctly. Simply using **AVERAGE('my_data')** works only if the **Difference** is a calculated column; if attempting to create a measure from scratch, the full DATEDIFF formula must be utilized within the iterating function.

Finally, be mindful of report refresh cycles. The TODAY() function updates its value only when the underlying data model is refreshed. If the report viewer is seeing cached data, the calculation will reflect the date of the last successful refresh, not the literal moment the report is being viewed. For truly real-time updates (which often require dynamic query mechanisms or specific gateway setups), this limitation of the TODAY() function must be acknowledged and managed.

Conclusion: Enhancing Time Intelligence in Power BI

Mastering the calculation of elapsed time using the DATEDIFF and TODAY() functions is a foundational skill for any Power BI developer. This simple yet critical DAX pattern provides the engine for dynamic aging analysis, project management tracking, and effective time-series reporting across virtually any dataset. By ensuring your date columns are properly typed and the formula adheres to the strict DAX syntax, you unlock significant analytical potential within your dashboards.

Note: You can find the complete documentation for the **DATEDIFF** function in DAX.

Related Time Intelligence Tutorials

Building upon the foundation of calculating simple date differences, Power BI offers numerous

advanced techniques for time intelligence. These techniques allow for complex comparisons, period-over-period calculations, and fiscal calendar management, enabling deeper insights into trends and performance fluctuations.

The following tutorials explain how to perform other common tasks in Power BI, providing further avenues to enhance your data analysis capabilities: