How to Extract the Date from a Datetime in Power BI

Authored by

stats writer

January 13, 2026

In the realm of business intelligence, managing time-series data is paramount. Often, raw datasets contain comprehensive **datetime** stamps, combining both date and precise time information. While this detail is valuable for granular analysis, many reporting requirements necessitate isolating just the date component for aggregation, grouping, or relationship building. **Power BI**, Microsoft's leading analytic tool, offers powerful capabilities for handling these transformations, primarily through the use of Data Analysis Expressions (**DAX**).

The simplest and most robust way to extract the date from a composite **datetime** value in **Power BI** involves defining a calculated column using specific **DAX** functions. While the `DATE` function exists, the `FORMAT` function provides greater flexibility and control over the output structure, making it the preferred choice for ensuring presentation consistency across reports and dashboards. Understanding this process is fundamental for anyone working extensively with transactional or logged data, as it allows for cleaner data models and more insightful visual analysis.

This comprehensive guide will detail the exact syntax and step-by-step implementation required to extract the date component, ensuring that your data adheres perfectly to your analytical needs. We will focus specifically on defining a new column within your data model using the highly versatile **FORMAT function**, demonstrating how this method ensures both computational accuracy and presentation elegance.

Understanding DAX Syntax for Date Extraction

To perform calculations and transformations directly within the **DAX** data model, we utilize calculated columns. A calculated column allows you to define an expression that evaluates row by row, adding new derived information to your existing table structure. When dealing with **datetime** values, the goal is often not just to extract the date numerically, but to display it in a user-friendly, consistent format that simplifies subsequent filtering and visualization tasks within **Power BI** reports.

The standard syntax in **DAX** for extracting the date from a **datetime** column relies on the powerful `FORMAT` function. This function takes two main arguments: the value or column reference you wish to format, and the specific format string you want applied. The output is always a text string, which is an important consideration for subsequent data aggregation, though it is highly effective for visual presentation.

date = FORMAT('my_data', "M/D/YYYY")

This particular expression constructs a new column named **date**. It meticulously extracts the date portion from the source column named **Datetime**, which resides within the table called **my_data**.

The format string `"M/D/YYYY"` dictates the exact appearance of the resulting date, ensuring a standardized representation across all rows. Using the **FORMAT function** in this manner ensures precise control over the output presentation, which is often essential for regulatory reporting or specific user interface requirements.

Detailed Review of the FORMAT Function

While simpler functions might extract a date component, the **FORMAT function** offers unparalleled flexibility by allowing developers to define exactly how the output should appear. This is crucial because date formatting is highly dependent on regional standards and analytical requirements. For instance, some users may require the Month name spelled out (e.g., 'January'), while others need a short numeric representation (e.g., '01'). The `FORMAT` function handles all these variations by interpreting the provided format string.

The format string, placed within quotation marks, acts as a template for the resulting text output. Common date formatting elements include: `M` (Month number), `MM` (Month number with leading zero), `MMM` (Abbreviated Month name), `MMMM` (Full Month name), `D` (Day number), `DD` (Day number with leading zero), `YY` or `YYYY` (Two or four-digit year). The structure `"M/D/YYYY"`, as used in our example, is a common US-centric standard, but this can be easily modified to suit ISO (`"YYYY-MM-DD"`) or European (`"DD/MM/YYYY"`) conventions simply by adjusting the format string parameter.

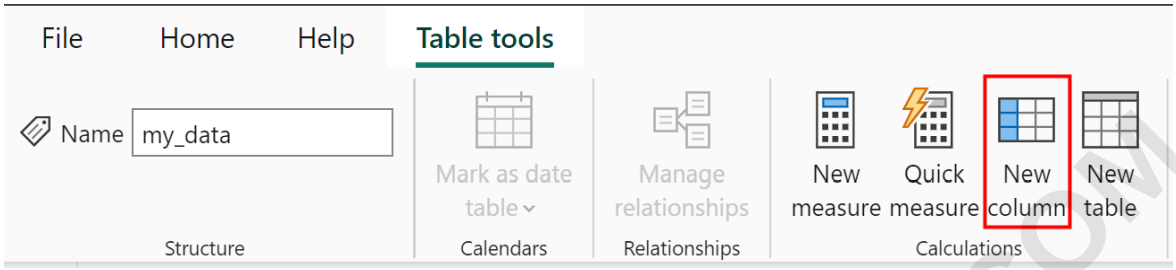
It is important to remember that when the `FORMAT` function is utilized for date extraction, the resulting column's data type is **Text**, not Date. While this is excellent for display purposes, it may necessitate additional steps if the derived date column needs to be used in complex date hierarchy calculations or if time intelligence functions are required. For visualization, however, converting the **datetime** value to a clean, readable text format often resolves many common presentation headaches.

Step-by-Step Practical Application in Power BI Desktop

Let us now walk through the precise steps necessary to implement this solution within the **Power BI** Desktop environment. This process assumes you have already loaded your data table (named **my_data** in this scenario) which contains the raw **Datetime** column requiring transformation. The overall objective is to enrich the data model by permanently adding a column that isolates the date for streamlined reporting.

The process begins by navigating to the modeling tools within **Power BI** Desktop. You must ensure that the table containing the target **Datetime** column is selected in the Data view. To initiate the creation of a new calculated column, you must locate the ribbon interface. The following visual guides the initial interaction:

To initiate the creation of the derived date column, click the **Table tools** tab located at the top of the interface. Once the appropriate tab is active, locate and click the icon specifically labeled **New column**. This action opens the formula bar, allowing you to input the **DAX** expression that will define the new column's contents.

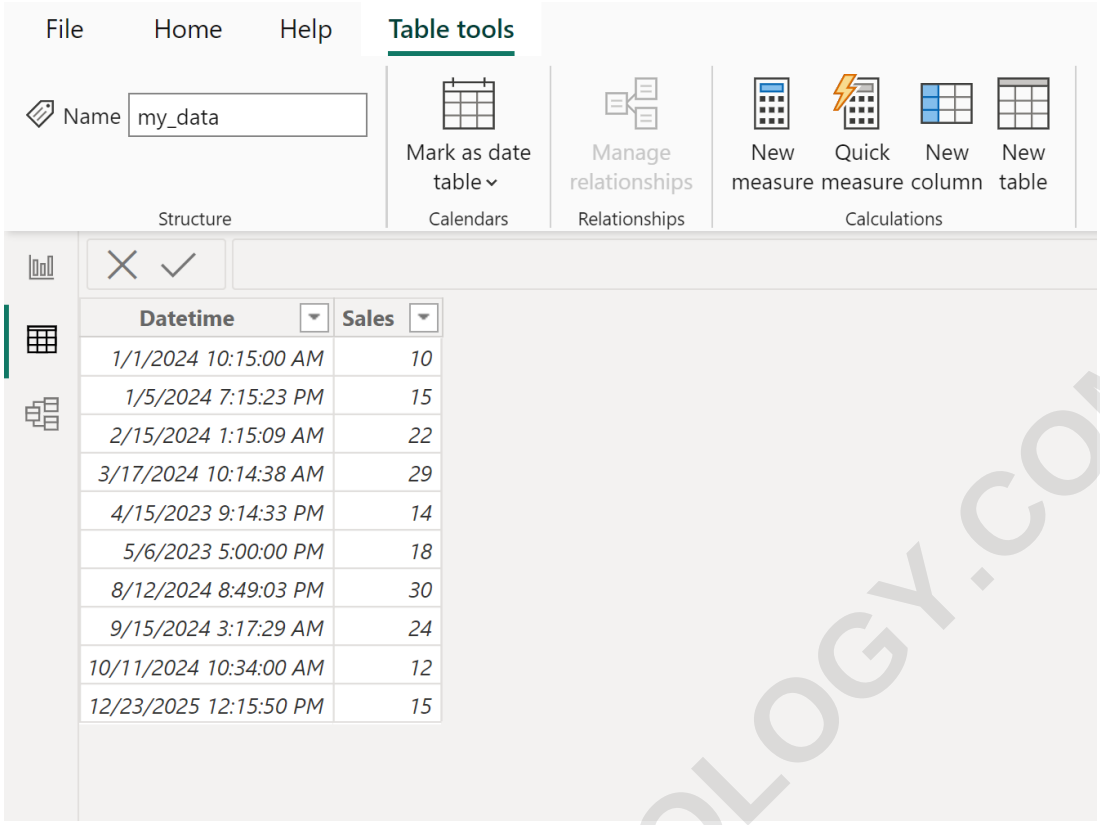


Once the formula bar appears, it is ready to accept the **DAX** logic. This is where you will leverage the `FORMAT` function to specify the extraction and presentation rules. Carefully typing the formula ensures that the calculation engine correctly processes the transformation row by row across your entire dataset.

Example: Transforming the Sample Sales Data

Consider a scenario where we are analyzing sales transactions. Our sample table, imaginatively named **my_data**, contains essential metrics, including the sales amount and a precise **Datetime** stamp capturing when the sale occurred. This detailed **datetime** information is excellent for tracking activity down to the second, but for daily or weekly aggregation, the time component is extraneous and can complicate group-by operations.

Suppose we have the following table named **my_data** in **Power BI** that contains information about total sales made by some company at specific **datetime** stamps:



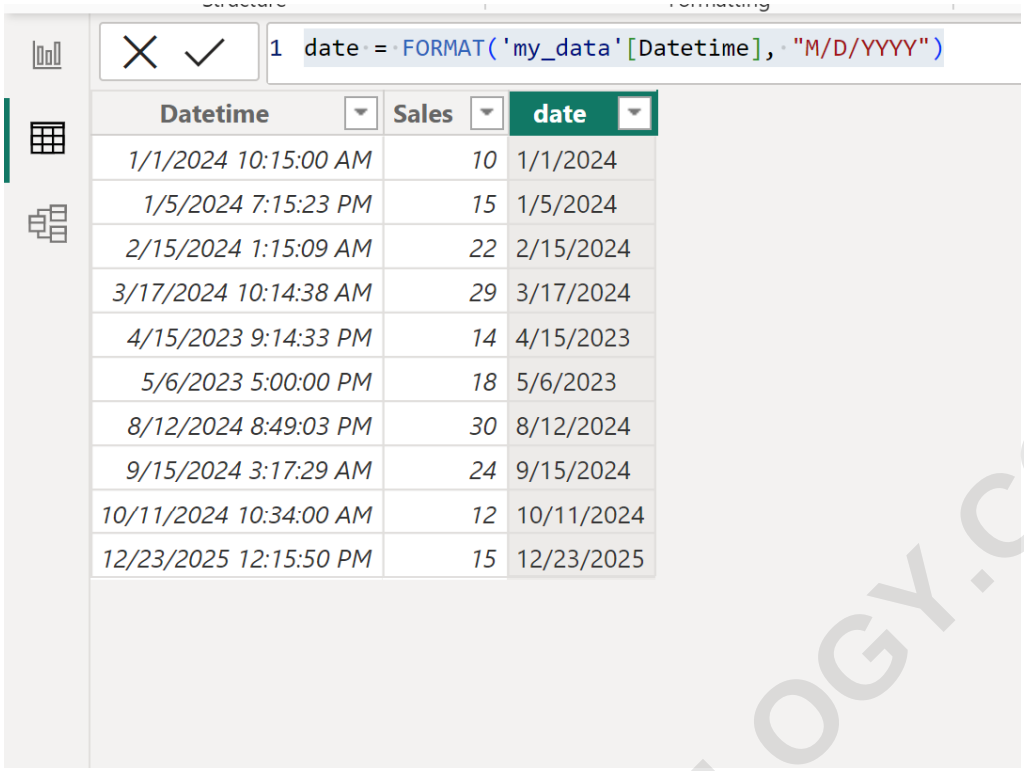
Datetime	Sales
1/1/2024 10:15:00 AM	10
1/5/2024 7:15:23 PM	15
2/15/2024 1:15:09 AM	22
3/17/2024 10:14:38 AM	29
4/15/2023 9:14:33 PM	14
5/6/2023 5:00:00 PM	18
8/12/2024 8:49:03 PM	30
9/15/2024 3:17:29 AM	24
10/11/2024 10:34:00 AM	12
12/23/2025 12:15:50 PM	15

Our specific objective is to extract only the date portion from every entry within the **Datetime** column. This derived date column will then serve as a clean, standardized dimension for subsequent reporting requirements, such as calculating total sales per day or creating date hierarchies.

Following the steps outlined in the previous section (clicking **Table tools** and then **New column**), we proceed by entering the calculated column definition into the formula bar. This definition utilizes the **FORMAT function** combined with the desired date mask:

date = FORMAT('my_data', "M/D/YYYY")

Upon committing this formula, the **Power BI** engine calculates the output for every row, resulting in a new column named **date**. This new column now exclusively contains the date component, stripped of its time stamp, as demonstrated below. Notice how the calculation isolates the date regardless of the time recorded (AM or PM, different seconds):



Reviewing specific transformations confirms the efficacy of the **DAX** formula:

The formula successfully extracts **1/1/2024** from the full **datetime** value 1/1/2024 10:15:00 AM.
The extraction yields **1/5/2024** from the **datetime** 1/5/2024 7:15:23 PM.
The result is **2/15/2024**, derived from the input **datetime** 2/15/2024 1:15:09 AM.

This process ensures every row now has a clean date field available for aggregated analysis, simplifying the subsequent development of reports and dashboards.

Alternative Methods Using Power Query (M-Language)

While **DAX** is indispensable for calculations within the data model, transformations related to data cleaning and type manipulation are often best handled upstream using the Power Query Editor, which utilizes the M-language. Performing these extractions in Power Query minimizes the load on the **DAX** engine and is generally considered a best practice for data preparation, especially when dealing with large datasets.

In the Power Query Editor, extracting the date from a **datetime** column is straightforward and does not require complex function writing. The graphical interface provides built-in transformation tools. The typical steps involve selecting the target column (e.g., **Datetime**), navigating to the Transform tab, clicking on the Date button, and selecting the 'Date Only' or 'Convert to Date' option. This operation automatically generates M-code, such as `Table.TransformColumnTypes`, or uses

functions like `Date.From()`.

The primary advantage of using M-language for this extraction is that the resulting column is correctly classified as a true **Date** type, unlike the **Text** output generated by the **DAX FORMAT function**. A native Date type enables seamless integration with Power BI's automatic date hierarchy features and guarantees compatibility with advanced time intelligence functions, leading to a more efficient and robust data model.

Deciding between **DAX** (calculated column) and M-Query (transformation step) depends on the use case. If the transformation must happen dynamically based on other measures or requires filtering within the visual layer, **DAX** is necessary. However, if the goal is simple, static data cleaning and type conversion--as is typically the case for date extraction--M-Query is the superior and recommended approach for performance reasons.

Best Practices for Date Handling and Performance

Effective date handling is critical for maintaining high performance in large **Power BI** data models. Choosing the right method for date extraction can significantly impact refresh times and the responsiveness of reports. The general principle is to perform as many static transformations as possible during the ETL (Extract, Transform, Load) phase, which means prioritizing Power Query (M-language) over **DAX** calculated columns where feasible.

Using **DAX** calculated columns, especially those involving string operations like the **FORMAT function**, adds overhead to the data model. Calculated columns consume RAM and increase the size of the PBIX file because the results are stored row by row, permanently within the VertiPaq engine. While the formula demonstrated (`FORMAT`) is simple, relying heavily on calculated columns for basic data preparation should be avoided when Power Query offers a more efficient alternative.

Furthermore, when dealing with international deployments, careful consideration of the date format string is essential. The `M/D/YYYY` format is ambiguous outside of specific regions. Best practice dictates using an unambiguous standard, such as the ISO format (`YYYY-MM-DD`), or using locale-aware formats provided by the Power Query Editor to ensure consistency across different user settings. Finally, whenever performance is a critical factor, ensure that any extracted date column is of the native **Date** data type (achieved via Power Query) to optimize storage and enable efficient use of built-in calendar tables and time intelligence features provided by **Power BI**.

Following these guidelines ensures that your data preparation is not only accurate but also optimized for speed and scalability, leading to a more professional and reliable business intelligence solution.