Learning the SAS TODAY Function: A Tutorial with Examples

Authored by

Mohammed looti

November 14, 2025

The Indispensable Role of the TODAY Function in SAS Date Management

In the expansive and rigorous discipline of statistical programming and data analysis, particularly within the **SAS** environment, the accurate handling of date and time variables is a foundational requirement. Dates are far more than mere numerical labels; they serve as critical anchors for executing time-series analysis, ensuring chronological data integrity, and automating timely report generation. Among the most fundamental tools available for date manipulation in **SAS** is the **TODAY function**. This powerful, zero-argument function provides an instantaneous and reliable mechanism for retrieving the operating system's current date, making it absolutely indispensable across a vast spectrum of analytical and reporting applications.

The primary function of the **TODAY function** is to return the current date as a **SAS date value**. To leverage this utility effectively, analysts must first comprehend the internal structure of date storage within the software. **SAS** inherently stores all dates as a running count of the number of days that have elapsed since a specific reference point: January 1, 1960. This reference date, universally known as the **SAS epoch**, is vital because it facilitates extremely efficient arithmetic calculations and robust comparisons between time points. However, because this internal representation is a simple, non-human-readable integer, it necessitates the consistent application of specific formatting techniques before data can be presented to end-users.

This comprehensive guide is structured to transition you seamlessly through the practical integration of the **TODAY function**. We will begin by observing its default, raw numeric output and subsequently delve into the application of various **SAS date formats**. Whether your task involves stamping reports with the precise current time, calculating the age of records relative to the present date, or simply tagging data processing steps, the **TODAY function** offers a precise and fundamental starting point. Upon completion of this tutorial, you will possess a complete understanding of how to weave reliable date handling into all your **SAS programming** workflows.

Understanding the Raw Output: Numeric SAS Date Values

When the **TODAY function** is executed within **SAS** without the subsequent application of any explicit display formatting, it exposes the current date in its most fundamental numeric form. This numeric integer represents the exact **SAS date value**, meticulously counting the number of days that have transpired since the established SAS epoch of January 1, 1960. This specific internal representation is absolutely central to how **SAS** efficiently manages and performs all date-related computational tasks, ensuring mathematical accuracy in comparisons, calculations, and overall storage.

To clearly illustrate this core behavior, we will utilize a basic **DATA step**. This procedural step involves the creation of a temporary dataset, named `my_data`, where the result of the **TODAY**

function is assigned to a new variable called `today_date`. After the dataset is successfully generated, we employ the **PROC PRINT** procedure to display the contents. This allows us to directly observe the large, raw numeric integer produced by the function before any user-friendly display modifications are applied, highlighting the distinction between internal storage and external presentation.

```
/* Create a dataset that contains the current date */
data my_data;
today_date=today();
run;

/* View the created dataset to see the raw SAS date value */
proc print data=my_data;
```

When unformatted, the output generated by the **TODAY function** will invariably appear as a substantial integer. For instance, if the program were executed on a hypothetical date such as May 5, 2023, the variable `today_date` would store the numeric value **23135**. This figure is the precise quantification of days elapsed since the SAS epoch. While this numeric format is flawless for internal computational speed and accuracy, it lacks immediate clarity for human interpretation. This initial observation powerfully underscores the fundamental requirement for employing **SAS date formats** to bridge the critical gap between the internal data structure and external reporting readability.

Obs	today_date
1	23135

The Crucial Separation: Format vs. Value in SAS

As clearly demonstrated in the previous example, raw **SAS date values** are numeric integers that offer zero intuitive meaning to end-users. This inherent challenge is systematically addressed and overcome through the rigorous application of **SAS date formats**, which are essential utilities for effective data presentation. A format functions as a specific instruction set, directing the SAS system on how to visually translate an internal numeric value into a recognizable and customary date string, such as "DD-MON-YY" or the widely used "MM/DD/YYYY" structure.

It is absolutely critical for all analysts to internalize the distinction that applying a date format modifies only the visual display of the variable, having zero effect on the underlying numeric value stored in the system's memory. This separation is paramount for maintaining data integrity: all

subsequent arithmetic operations, date comparisons, and time calculations continue to operate correctly on the accurate numeric count, while reports and outputs simultaneously display dates in a concise and highly user-friendly manner. **SAS provides an extensive library of built-in date formats**, enabling analysts to easily comply with diverse regional standards, regulatory requirements, and highly specific display mandates. The appropriate format selection must always be driven by the specific needs of the target audience and the overall context of the data presentation.

The standard methodology for implementing a format involves utilizing the **FORMAT statement**. This statement can be strategically placed within a **DATA step** to permanently associate the chosen format with the variable within the saved dataset, or it can be placed in a **PROC step** for temporary display modifications specific to that output or report. By associating a format with a variable, you effectively control how those numeric dates are rendered throughout your session or within a designated output file. The following practical examples demonstrate how to pair the instantaneous current date retrieval from the **TODAY** function with the versatility of the **FORMAT statement** to achieve perfectly clear date representations.

Practical Application: Formatting with DDMMYY10.

The **DDMMYY10. format** is frequently implemented across global **SAS** installations, particularly gaining importance in geographical regions that strictly adhere to the day-month-year convention. This format structures the date output as `DD/MM/YYYY`, consistently employing forward slashes as clean visual separators. In this structure, `DD` unambiguously represents the day, `MM` the month, and `YYYY` the complete four-digit year. The numerical suffix "10" in the **DDMMYY10.** format is critical, as it specifically denotes the required overall width of the formatted field, ensuring adequate space for all eight date characters plus the two necessary separators to display correctly without truncation.

To successfully integrate this format, we must insert the **FORMAT statement** into our existing **SAS programming** structure immediately following the point where the `today_date` variable is created within the **DATA step**. The code provided below illustrates this precise implementation. We also strategically include a `PUT` statement, which writes the newly formatted date value directly to the SAS log, serving as an efficient real-time verification check. The final, neatly structured output table is then generated using the **PROC PRINT** procedure.

```
/* Create a dataset that contains the current date */
data my_data;
today_date=today();
format today_date ddmmyy10.;
put today_date;
```

```
run;

/* View the created dataset with the DDMMYY10. format applied */
proc print data=my_data;
```

Upon successful execution, the **TODAY function** continues its designed behavior of generating the underlying numeric **SAS date value** internally. However, the accompanying instruction `format today_date ddmmyy10.;` overrides the default numeric display, yielding a conventional and highly readable date string. For instance, if the execution date is May 5, 2023, the output for the `today_date` column will be clearly rendered as **05/05/2023**. This transformation is vital, as it drastically improves the interpretability and accessibility of the data for any user reviewing the resulting dataset, report, or log output.

Obs	today_date
1	05/05/2023

Enhancing Readability with Textual Date Formats (DATE9.)

While purely numeric formats like **DDMMYY10.** excel in brevity and space efficiency, the complete **SAS date formats** library also offers powerful options tailored for enhanced textual clarity. A highly favored option among these is the **DATE9. format**, which presents dates using the structure `DDMMMYYYY`, producing outputs such as "05MAY2023". This format is widely adopted because it explicitly spells out the month using a standardized three-letter abbreviation, effectively eliminating the potential for ambiguity that frequently arises when relying solely on numeric date representations (e.g., confusion between US MM/DD/YY and European DD/MM/YY standards). The "9" in **DATE9.** is specified to ensure the field width is perfectly sufficient to accommodate the day, the three-letter month, and the complete four-digit year.

To successfully implement the **DATE9. format**, we adhere to the same precise procedural steps utilized in the previous example. We adjust the DATA step by updating the FORMAT statement to specifically invoke `date9.`. This change serves as a direct instruction to the **SAS** system to display the `today_date` variable in this new, clearer format, while critically ensuring that the underlying numeric **SAS date value** remains untouched for all computational purposes.

```
/* Create a dataset that contains the current date */
data my_data;
today_date=today();
format today_date date9.;
```

```
put today_date;
run;

/* View the created dataset with the DATE9. format applied */
proc print data=my_data;
```

Execution of this structured code yields an output where the current date is rendered using the highly textual **DATE9.** format. If the script were run on May 5, 2023, the `today_date` variable will be displayed clearly as **05MAY2023**. This format is exceptionally valuable and often preferred in generating formal reports, executive summaries, or summary tables where maximizing clarity and the explicit identification of the month are prioritized over purely numerical representation. The inherent flexibility and depth offered by SAS formats empower developers to meticulously tailor their outputs to precisely meet any user requirement, consistently solidifying the SAS system's reputation for robust and adaptable data management and reporting.

Obs	today_date
1	05MAY2023

Exploring Related Functions and Date Manipulation Best Practices

While the preceding examples provide a strong foundational understanding of the **TODAY** function, its utility represents only the entry point to the extensive capabilities provided by the SAS date and time function library. SAS offers a comprehensive suite designed to address virtually every advanced date display need, including specialized formats that incorporate the day of the week, full month names, or entirely custom separators. Other commonly used formats include **MMDDYY10.** (rendering as 05/05/2023), **MONYY7.** (e.g., MAY23), and the highly descriptive **WEEKDATE.** (e.g., Friday, May 5, 2023). Each distinct format allows for highly customized and contextually appropriate date rendering, ensuring that data presentation always aligns with analytical goals.

When systematically integrating date functions into your **SAS** programs, strict adherence to established best practices is paramount for maintaining absolute data integrity and computational accuracy. Firstly, analysts must continuously reinforce the core principle that the **TODAY** function, like all SAS date functions, outputs a purely numeric **SAS date value**, and that formatting serves only as a visual overlay. The underlying numeric count must remain stable and untouched to guarantee the accuracy of all mathematical operations, such as calculating precise differences in days between two distinct recorded events. Secondly, establishing and consistently applying a standardized set of formats across all reports and datasets will dramatically enhance readability, reduce ambiguity, and mitigate potential confusion for end-users interacting with the data.

It is important to note that while **TODAY** efficiently retrieves the current date at midnight, SAS provides several closely related functions for more intricate date and time manipulation. For example, `DATE()` is recognized as an official alias for the **TODAY function**, performing the identical operation, whereas the `DATETIME()` function retrieves the current date and time expressed as a comprehensive SAS datetime value, which includes hours, minutes, and seconds. Furthermore, functions like `MDY()` (Month, Day, Year) allow users to construct a SAS date value accurately from individual numeric components. Exploring these advanced tools can substantially expand your capacity to handle complex, real-world date challenges within your **SAS programming** projects. For the most complete and authoritative reference on all these utilities, always consult the official **SAS documentation**.

Conclusion: Mastering Current Date Retrieval in SAS

The **TODAY function** remains a foundational and critically important utility in **SAS** for dynamically and reliably incorporating the current date into data processing routines and automated reporting mechanisms. By achieving a deep and practical understanding of its core behavior--which is to return a numeric **SAS date value** based on the 1960 epoch--and subsequently mastering the skillful application of the immense variety of **SAS date formats**, users gain the crucial ability to manage date data that is simultaneously computationally reliable for backend operations and effortlessly interpretable for frontend reporting.

Effective and precise date management constitutes a vital cornerstone of robust statistical data analysis and efficient business intelligence reporting. The simplicity and immediate utility offered by the **TODAY** function, when synergistically combined with the versatility of the SAS formatting options, establishes an exceptionally strong framework for tackling even the most advanced time-series analysis and date manipulations. We strongly recommend that all users make continuous reference to the official SAS documentation, as it provides the most comprehensive and up-to-date technical details, ensuring that your date handling capabilities remain current, accurate, and highly precise within all your SAS programs.

Additional Resources for Advanced SAS Date and Time Functions

To further advance your proficiency in **SAS** programming and complex date management scenarios, we highly recommend exploring these authoritative resources and tutorials. These materials will help you move beyond simple current date retrieval into more sophisticated time manipulation techniques, ensuring you can tackle any analytical challenge:

Official **SAS Functions and CALL Routines: Reference** for a comprehensive, encyclopedic list of all date, time, and datetime functions available in the system.

Official **SAS Formats and Informats: Reference** for an exhaustive, detailed guide to all available

date, time, and datetime formats and their usage specifications.

Tutorials focused on accurately calculating the difference between two dates and generating complex time intervals in SAS.

Guides detailing the rigorous process of converting date and time values between different formats or various regional system standards.

Articles discussing advanced interval calculation functions such as `INTNX` (Interval Next) and `INTCK` (Interval Check) for forecasting and period counting.