Learn SAS: Extracting the Day of the Week from Date Variables

Authored by

Mohammed looti

November 14, 2025

In the realm of statistical computing and advanced analytics, effectively managing and manipulating dates is a **fundamental skill** for any professional working with data, especially within the powerful environment of SAS. A frequently encountered requirement is the ability to extract specific temporal components from a standard date variable, most notably the day of the week. This specific piece of information is invaluable and often crucial for conducting a wide array of analytical tasks, including intricate scheduling, performing detailed trend analysis, and generating standardized reporting.

Fortunately, SAS provides a suite of **powerful and specialized functions** designed to simplify complex date handling processes. When the objective is to determine the day of the week, experts utilize two primary methods. The first is the **WEEKDAY function**, which yields a precise numerical representation of the day (an integer from 1 to 7). The second, generally preferred for clear reporting, is the combination of the **PUT function** with the specialized **DOWNAME format**, which conveniently outputs the full, textual name of the corresponding day (e.g., "Monday").

This comprehensive guide is designed to delve into the nuances of both of these essential approaches. We will meticulously explain their respective usage, inherent characteristics, and crucial practical applications within the SAS environment. Furthermore, we will walk through a clear, step-by-step example, demonstrating precisely how to implement these functions within your own SAS programs to efficiently derive accurate day-of-week information directly from your existing datasets.

Understanding Internal Date Values and Formats in SAS

Before proceeding with the implementation of specific date functions, it is absolutely essential to establish a firm grasp of how the SAS system handles and stores dates internally. Unlike typical calendar representations, SAS stores date values as a simple, continuous count: the number of days that have elapsed between a fixed reference point, January 1, 1960, and the given date.

Consequently, January 1, 1960, is stored internally as 0, while January 2, 1960, is stored as 1, and this numerical sequence continues indefinitely. This specific numerical representation is universally known as a SAS date value, and it forms the fundamental backbone for the operation of all date functions and calculations within the system.

Although dates are maintained internally using this numerical format, they are displayed to the user in a visually readable format through the application of SAS formats. A format serves as an instruction set that dictates precisely how SAS should present the underlying numerical date value for output purposes. For instance, the internal SAS date value of 22287, which corresponds to January 1, 2021, can be presented to the user as '01JAN2021' by employing the standard DATE9. format, or it could be rendered as 'January 1, 2021' using the WORDDATE. format. Understanding this critical distinction between the **internal numerical storage** and the **external displayed**

presentation is paramount to performing effective and error-free date handling within any SAS program.

Utilizing the WEEKDAY Function for Numerical Extraction

The WEEKDAY function, a core component of SAS's date utility library, stands as a straightforward yet highly effective instrument for extracting the day of the week and representing it as a numerical value. This function requires only a valid SAS date value as its mandatory argument. In return, it yields an integer ranging from 1 to 7, following a fixed convention: the integer 1 consistently represents Sunday, 2 represents Monday, and this progression continues sequentially up to 7, which represents Saturday. This standardized numerical output is highly beneficial for computational tasks.

The numerical output provided by the WEEKDAY function is particularly advantageous when the need arises to execute calculations, apply filtering rules, or implement complex conditional logic based strictly on the day of the week. For practical examples, analysts frequently utilize this function to filter out data recorded during weekend periods (where the function would return 1 or 7), or to systematically categorize observations by specific days for strategic scheduling or resource allocation purposes. Its direct numerical mapping is specifically designed to facilitate seamless programmatic control within data step operations or analytical procedures, making it an indispensable tool for data processing.

Consider a realistic scenario involving a comprehensive dataset of customer transactions where the primary goal is to accurately identify all sales events that occurred specifically on a Tuesday. By incorporating the WEEKDAY function into the data step, an analyst can effortlessly create a new derived variable that represents the numerical day of the week (e.g., 3 for Tuesday). Subsequently, filtering the dataset to select only those records where this new variable precisely equals 3 becomes a quick and efficient operation. This direct, numerical approach significantly streamlines the execution of numerous analytical and reporting tasks.

The Power of PUT and DOWNAME for Textual Output

While the numerical result from the WEEKDAY function is necessary for computational tasks, there are frequent instances--especially in final reporting or output generation--where the actual, descriptive name of the day is required for enhanced clarity and presentation. In these cases, the PUT function, used in specialized conjunction with the DOWNAME. format, becomes an invaluable technique. The core role of the PUT function is to perform a transformation, effectively converting a numeric input value--such as a SAS date value--into a corresponding character string based on a format specified by the user.

The DOWNAME. format is expertly engineered to translate any given SAS date value directly into

the full, textual name of its corresponding day of the week. This results in outputs such as "Sunday", "Monday", or "Tuesday". When this format is correctly applied using the PUT function to a date variable, the output seamlessly provides the textual name of the day. This significantly enhances the immediate accessibility and inherent understandability of your data for stakeholders who may not possess a technical background.

Employing the PUT function combined with DOWNAME. is the ideal methodology for producing highly descriptive reports, generating clear labels, or populating necessary fields in a database where a textual day of the week is explicitly preferred over a potentially ambiguous numerical code. This technique dramatically improves the overall readability of your program output, eliminating the need to manage external or manual lookup tables to translate the numerical day values into their descriptive names.

Practical Implementation: Step-by-Step Derivation in SAS

To solidify our understanding of these functions, let us walk through a highly practical example demonstrating their application in a real-world scenario. Assume the existence of a modest dataset within SAS that contains the birth dates for a select group of individuals. Our defined goal is to systematically augment this existing dataset by introducing two distinct new variables: the first will capture the day of the week as a precise numerical identifier, and the second will provide its full, descriptive name.

We begin the process by constructing our initial dataset, which we name **original_data**. The following SAS code snippet meticulously details the creation of this dataset. Crucially, this setup includes the **birth_date** variable, which is explicitly formatted using DATE9. to ensure that the input dates are both recognized and displayed clearly in a standardized format.

```
/* Creating the initial dataset for date processing */
data original_data;
format birth_date date9.;
input birth_date :date9.;
datalines;
01JAN2021
22FEB2022
14MAR2022
29MAY2022
14OCT2023
01NOV2024
26DEC2025
;
```

run;

/* Displaying the contents of the initial dataset */
proc print data=original_data;

Upon the successful execution of this initial code block, the **original_data** dataset will be accurately populated with all the specified birth dates. The subsequent PROC PRINT statement serves the vital function of allowing us to visually inspect the contents of the newly created dataset, thereby confirming both its intended structure and the integrity of the entered date values.

Obs	birth_date
1	01JAN2021
2	22FEB2022
3	14MAR2022
4	29MAY2022
5	14OCT2023
6	01NOV2024
7	26DEC2025

Next, we proceed to create a new, enhanced dataset, which we will name **new_data**, building directly upon the foundation provided by **original_data**. In this crucial data step, we introduce and define two new variables: **weekday_number**, which is calculated instantaneously using the **WEEKDAY function**, and **weekday_name**, which is generated using the powerful **PUT function** in combination with the **DOWNAME.** format. This transformation effectively enriches our dataset by appending the desired day-of-week information for every single birth date record.

/* Creating the new dataset with derived weekday variables */
data new_data;
set original_data;
weekday_number = WEEKDAY(birth_date);
weekday_name = put(birth_date, dowName.);
run;

/* Viewing the contents of the newly enhanced dataset */
proc print data=new_data;

The execution of this second segment of code successfully generates the fully processed **new_data** dataset. The `SET original_data;` statement ensures that all existing variables are

seamlessly imported from our initial dataset. Following this, the two assignment statements perform the necessary calculations, deriving and assigning the day of the week in both its numerical and named formats to their respective new variables. The concluding PROC PRINT command then displays the final, enhanced dataset, showcasing the results of our date manipulation.

Obs	birth_date	weekday_number	weekday_name
1	01JAN2021	6	Friday
2	22FEB2022	3	Tuesday
3	14MAR2022	2	Monday
4	29MAY2022	1	Sunday
5	14OCT2023	7	Saturday
6	01NOV2024	6	Friday
7	26DEC2025	6	Friday

Analyzing and Interpreting the Derived Results

Upon a careful review of the output displayed by the **new_data** dataset, the immediate presence of the two newly engineered variables--**weekday_number** and **weekday_name**--becomes evident. These derived variables collectively furnish comprehensive day-of-week information corresponding to each recorded individual birth date, serving as a clear demonstration of the effective and complementary application of both the **WEEKDAY** and **PUT** functions within the SAS programming environment.

The **weekday_number** variable provides a clear, quantitative representation, adhering strictly to the SAS standard where the numeral 1 corresponds to Sunday, 2 represents Monday, and this numerical sequence progresses up to 7, which signifies Saturday. Working in tandem, the **weekday_name** variable offers a full, descriptive, and textual name of the day, delivering immediate and intuitive readability for any user interpreting the dataset.

To firmly solidify the understanding of the underlying logic, let us meticulously examine a few specific entries extracted from our illustrative example dataset:

For the recorded date of **January 1st, 2021**, the resulting **weekday_number** is 6, and the corresponding **weekday_name** is **Friday**. This output accurately reflects the fact that January 1, 2021, fell precisely on the sixth day of the defined week structure, which is Friday.
For the date **February 22nd, 2022**, the calculated **weekday_number** is 3, with the associated

weekday_name being **Tuesday**. This result correctly identifies February 22, 2022, as a Tuesday, which is designated as the third day of the week.
In the case of **March 14th, 2022**, the derived **weekday_number** is 2, and the resulting **weekday_name** is **Monday**. This final example clearly illustrates that March 14, 2022, was indeed a Monday, the second day within the standard weekly cycle.

These detailed examples successfully highlight the remarkable ease and precision with which SAS is able to transform inherently complex date information into highly accessible and easily digestible formats, thereby serving the dual needs of rigorous analytical processing and user-friendly, high-quality reporting.

Conclusion and Avenues for Further Exploration

The mastery of robust date manipulation techniques is absolutely indispensable for data professionals operating within the SAS environment. The **WEEKDAY function**, alongside the **PUT function** utilizing the DOWNAME. format, provides highly reliable and straightforward methodologies for extracting crucial day-of-week information from virtually any standard date variable. Regardless of whether your immediate objective demands a numerical code for programmatic calculations or a descriptive, textual name suitable for comprehensive reports, SAS consistently furnishes the requisite tools to achieve your analytical objectives with maximum efficiency.

A thorough understanding of these specific functions represents a vital stepping stone toward unlocking more sophisticated and time-based analyses. For instance, analysts can readily extend this acquired knowledge to perform intricate sales pattern analysis stratified by the day of the week, implement automated report scheduling triggered by specific weekdays, or accurately identify and model trends that are intrinsically tied to predictable weekly cycles. SAS's comprehensive suite of date and time functions empowers users to perform a vast and diverse array of **temporal data transformations**, significantly enhancing data utility.

We strongly encourage all users to actively experiment with both the **WEEKDAY** and **PUT** functions and to further explore other closely related SAS date functions, such as **DATEPART**, **INTNX**, and **INTCK**. Committing to deepening your comfort level and proficiency with SAS date handling will invariably lead to more complex, actionable, and insightful outcomes in your overall data analysis endeavors.

Supplementary Learning Resources

To further deepen your comprehensive understanding of SAS programming principles and advanced data manipulation techniques, we recommend exploring the following tutorials and official documentation. These resources provide detailed explanations on how to effectively

perform other common as well as more complex and advanced tasks within the SAS system.