How to Use CARDS Statement in SAS?

Authored by

stats writer

November 19, 2025

Introduction to the CARDS Statement in SAS

The CARDS statement is a fundamental component within the DATA step of SAS, the powerful statistical software suite. It serves a crucial function: facilitating the direct incorporation of data values into a new dataset immediately following the programming logic. When used, the CARDS statement signals to the SAS compiler that the subsequent lines of the program are not executable code but raw data intended for processing. This method is particularly useful when dealing with small to moderate amounts of data that are easier to manage and verify when embedded directly within the program script, rather than stored in a separate external file.

The primary purpose of the CARDS statement is to define the boundaries of the internal data block. Within the DATA step, the flow of execution typically involves defining the dataset name, specifying variables using the INPUT statement, and then using CARDS to introduce the actual records. Each line of data following the CARDS statement is treated as a single record, and the values within that line are read sequentially according to the specifications provided in the preceding INPUT statement. Understanding this sequence is vital for accurate data reading and processing within SAS programming environments.

This statement enables rapid prototyping and testing of SAS code, as it eliminates the need to manage separate file paths or ensure external file accessibility. While modern SAS development often relies on importing data from large, structured external sources (like databases or dedicated text files), the CARDS method remains an essential tool for creating test datasets, examples, or handling legacy code. It is intrinsically tied to how SAS processes data input buffer records, ensuring a clean, line-by-line interpretation of the provided values.

The Role of CARDS in the SAS DATA Step

The DATA step is the foundation of data management in SAS, responsible for reading data, creating new datasets, and manipulating existing ones. The CARDS statement is inextricably linked to the functionality of the INPUT statement within this step. The INPUT statement dictates the names of the variables, their order, and their type (character or numeric), while the CARDS statement provides the raw material--the actual data values--that SAS attempts to map onto those variables.

When SAS encounters the CARDS statement, it immediately switches its interpretation mode. It stops processing executable SAS commands and begins treating the subsequent lines as literal data until it encounters a semicolon (**;**) on a line by itself, which acts as the termination signal for the data block. This seamless integration allows the programmer to define both the structure (metadata) and the content (raw values) of the dataset within a single, cohesive block of code.

It is crucial to recognize that the data provided following CARDS resides within the input buffer.

SAS reads one record (one line of data) from this buffer at a time and applies the rules defined by the preceding INPUT statement. If the data types or lengths defined in INPUT do not match the structure of the data provided after CARDS, SAS will generate errors or assign missing values, highlighting the necessity for perfect alignment between the data definition and the data values.

Understanding the Basic Syntax of CARDS

Using the CARDS statement requires a specific structure within the DATA step to ensure successful dataset creation. The syntax is straightforward but must be positioned correctly relative to other statements like DATA, INPUT, and the closing RUN statement. The general structure begins with the specification of the new dataset name, followed by the definition of variables via the INPUT statement. The CARDS statement is then placed on its own line, followed immediately by the data values starting on the next line.

The entire data block is terminated by a single semicolon on a subsequent line. This delimiter is non-negotiable for the standard CARDS statement and must appear immediately following the last line of data to signal the return to executable code processing. This structure ensures that SAS clearly understands where the raw data ends and where the procedural code (like the RUN statement) resumes.

```
data my_data;
input var1 $ var2;
cards;
A 12
B 19
C 23
D 40
;
run;
```

In this foundational example, the **DATA** statement specifies the creation of a dataset named **my_data**. The INPUT statement defines **var1** as a character variable (indicated by the trailing dollar sign **$**) and **var2** as a numeric variable. The CARDS statement then signals the start of the data block, where 'A', 'B', 'C', 'D' are records for **var1**, and '12', '19', '23', '40' are records for **var2**. The mandatory semicolon terminates the data block and allows the final **RUN** statement to execute the DATA step.

Detailed Breakdown of the Data Input Process

To fully appreciate the power of the CARDS statement, one must understand how SAS utilizes the

information provided in conjunction with the INPUT statement. The coordination between these two statements is the essence of in-stream data reading in SAS.

data: This statement initiates the DATA step and assigns the desired name to the resulting SAS dataset.
input: This critical component defines the structure of the data records. It specifies the variable names and, if necessary, their types. A trailing dollar sign (**$**) after a variable name explicitly declares that variable as a character variable, instructing SAS to read non-numeric data for that field. Variables without the dollar sign are assumed to be numeric.
cards: This statement marks the exact point where the SAS compiler ceases interpreting lines as code and begins interpreting them as raw data records. Every line that follows, up until the terminating semicolon, is treated as data.

Once SAS encounters the **CARDS** statement, it understands that the immediate subsequent lines contain the actual values corresponding to the structure defined by the INPUT variables. The data is read using list input by default (if column specifications are not used), meaning values are separated by one or more spaces, and SAS moves sequentially through the input variables, assigning values from the record line by line. This process continues until all records provided after the CARDS statement have been read and processed, resulting in a complete, new dataset stored in memory or on disk.

Note on Character Variables: The use of the dollar sign "**$**" is essential for handling descriptive or textual data. If you omit the dollar sign for a character variable, SAS will attempt to read the text as a number, leading to conversion errors and potential missing values in the resulting dataset. This specific marker ensures that SAS correctly processes text strings of varying lengths, rather than treating the input as numerical values, which enhances data integrity during the reading phase.

Historical Context: Why is it Called CARDS?

The name "CARDS" is a direct and fascinating nod to the early history of computing and data processing. Before interactive computing environments became standard, data and programs were often physically entered into computers using punch cards (also known as Hollerith cards). These cards were the standard method for inputting data into mainframe computers throughout the mid-20th century, representing data through patterns of punched holes.

Programmers would manually punch holes into these stiff paper cards, where the presence or absence of a hole in a specific column represented a character or data value. These decks of cards were then fed sequentially into card readers, which served as the primary input mechanism for the computer system. When writing a SAS program in the earliest versions of the software, if a user wanted to embed data directly within the program script, they were essentially mimicking the process of feeding a physical stack of data cards after the program instruction cards.

The **CARDS** statement thus served as a conceptual separator: "Here is the code," followed by, "Now, here are the data cards." Although the physical technology of punch cards has been obsolete for decades, the term has been maintained in SAS--alongside its synonym DATALINES--as a piece of linguistic legacy, connecting contemporary data analysis techniques back to the foundational methods of computing. This history underscores the longevity and robustness of the SAS language syntax and provides context for understanding this unusual naming convention.

Practical Example: Creating a Dataset with Numeric and Character Variables

Let us examine a practical application of the CARDS statement to build a simple dataset containing NBA team statistics. This example clearly demonstrates how to mix character (team names) and numeric (points and assists) variables seamlessly within the in-stream data block.

The following code block uses CARDS to create a dataset named **my_data** with three variables: **team**, **points**, and **assists**. Notice the dollar sign used immediately after **team** in the INPUT statement, correctly designating it as a character field, while the subsequent variables are read as default numerics.

```
/*create dataset*/
data my_data;
input team $ points assists;
cards;
Mavs 14 9
Spurs 23 10
Rockets 38 6
Suns 19 4
Kings 30 4
Blazers 19 6
Lakers 22 14
Heat 19 5
Magic 14 8
Nets 27 8
;
run;
proc print data=original_data;
```

Once this code is executed, SAS reads the 10 data lines following the CARDS statement. For the first record, "Mavs" is assigned to the character variable **team**, "14" is assigned to the numeric variable **points**, and "9" is assigned to the numeric variable **assists**. This process repeats for all subsequent lines until the semicolon is encountered. The result is a perfectly structured dataset,

ready for statistical analysis or further manipulation, as shown in the output image below.

Obs	team	points	assists
1	Mavs	14	9
2	Spurs	23	10
3	Rockets	38	6
4	Suns	19	4
5	Kings	30	4
6	Blazers	19	6
7	Lakers	22	14
8	Heat	19	5
9	Magic	14	8
10	Nets	27	8

The DATALINES Alternative: Equivalence and Usage

It is important for any SAS user to know that the CARDS statement has a functional synonym: the **DATALINES** statement. Both statements perform the exact same action--signaling the start of an in-stream data block immediately followed by the raw data records, terminated by a semicolon. They are 100% interchangeable within the SAS DATA step environment.

Historically, the preference between CARDS and DATALINES varied, but in contemporary SAS programming, **DATALINES** is often encountered more frequently in modern documentation and code examples, largely because its name is more descriptive of its function (inputting data lines) than the historical reference implied by CARDS. However, from a compiler perspective, they are fully equivalent, meaning any code written using one can be instantly switched to the other without altering the logic or output.

If we replace the CARDS statement in the previous example with **DATALINES**, the output dataset is identical in every way, demonstrating their complete equivalence in function and syntax execution:

```
/*create dataset*/
data my_data;
input team $ points assists;
datalines;
Mavs 14 9
```

```
Spurs 23 10
Rockets 38 6
Suns 19 4
Kings 30 4
Blazers 19 6
Lakers 22 14
Heat 19 5
Magic 14 8
Nets 27 8
;
run;
proc print data=original_data;
```

The resulting dataset, shown in the image below, confirms that using **DATALINES** produces the exact same structure and content as using the **CARDS** statement. Therefore, both statements are equally valid for implementing in-stream data input in SAS programming.

Obs	team	points	assists
1	Mavs	14	9
2	Spurs	23	10
3	Rockets	38	6
4	Suns	19	4
5	Kings	30	4
6	Blazers	19	6
7	Lakers	22	14
8	Heat	19	5
9	Magic	14	8
10	Nets	27	8

Advanced Usage: CARDS4 and Handling Embedded Semicolons

A notable limitation of the standard CARDS (or DATALINES) statement is its requirement for the data block to be terminated by a solitary semicolon on its own line. This structure poses a problem if the raw data itself happens to contain semicolons, as SAS would prematurely stop reading the data block, leading to truncation of records and syntax errors in subsequent code.

To overcome this limitation, SAS provides the **CARDS4** (or **DATALINES4**) statement. The addition of the '4' indicates that the termination signal is altered. Instead of a single semicolon, **CARDS4** requires the data block to be terminated by a line containing four consecutive semicolons (**;;;;**). This allows the inclusion of single semicolons within the actual data records without confusing the SAS compiler.

Using **CARDS4** follows the same syntax principles as CARDS, replacing the keyword and adjusting the terminator. This feature demonstrates the flexibility of the DATA step in accommodating various data complexities, ensuring that even in-stream data loading can handle specialized characters without issue. When data integrity relies on preserving semicolons within the records, **CARDS4** is the preferred method for safe data input.

When to Opt for CARDS vs. External File Input

While the CARDS statement offers immediate convenience, deciding between embedded data (CARDS/DATALINES) and reading data from an external file is a crucial design choice in SAS programming. This decision typically hinges on the volume, complexity, and source of the data, influencing code efficiency and maintainability.

Small Datasets and Examples: CARDS excels for small, static datasets (e.g., fewer than 100 observations) used for testing logic, providing reproducible examples in documentation, or quickly demonstrating a procedure. Embedding the data makes the code self-contained and highly portable, simplifying sharing and debugging.
Large Datasets: For production environments dealing with thousands or millions of records, using CARDS is impractical and inefficient. Embedding large volumes of data bloats the program file and slows compilation. In these cases, the data should reside in dedicated external files (e.g., CSV, TXT, Excel, or database tables). SAS provides robust mechanisms, such as the **INFILE** statement, to handle large, external data streams efficiently.
Security and Separation of Concerns: In large-scale applications, it is considered best practice to separate the program logic from the data itself. Using external files ensures that data can be updated, secured, and managed independently of the SAS code used to process it. Data governance and regulatory compliance often necessitate this separation, making external data sources the standard for enterprise analytical projects.

In summary, the CARDS statement should be viewed as a powerful utility for rapid development and pedagogical purposes, rather than a method for handling massive operational datasets. Its simplicity and integration within the DATA step make it indispensable for certain programming tasks, but it does not replace the necessity of robust external file handling techniques for enterprise-level data processing.

Summary and Best Practices

The CARDS statement remains a valuable, if historically rooted, feature of the SAS language. It provides a direct, elegant way to embed data records immediately following the procedural code that defines how those records should be read. Its utility is highest in scenarios requiring quick testing or the creation of self-contained examples.

Best practices suggest programmers should prioritize clarity and maintainability. When utilizing this technique, ensure the variable definitions in the INPUT statement precisely match the format of the data provided after CARDS, paying special attention to character (**$**) versus numeric variables. Misalignment here is the single most common source of error when using in-stream data.

Furthermore, while CARDS is fully supported, many organizations adopt **DATALINES** for modern code clarity due to its more intuitive name. Regardless of the term used, the mandatory semicolon (**;**) termination must be included immediately following the last line of data to signal the end of the data block, allowing the SAS program flow to resume processing subsequent executable statements, such as the RUN command. Mastering the CARDS statement ensures proficiency in handling all forms of data input within the SAS environment.