How to use the IN= Option in SAS?

Authored by

stats writer

November 19, 2025

1. Introduction to the IN= Option in SAS

The SAS system is a powerful tool for data manipulation and statistical analysis. When dealing with complex analytical pipelines, users frequently encounter scenarios requiring the merging or concatenation of multiple input datasets. A critical challenge in this process is tracking the origin of each observation in the resulting combined file. This necessity is precisely addressed by the versatile **IN=** option, a powerful feature within the DATA step environment.

The primary function of the **IN=** option is not merely to facilitate the combination of data, but rather to automatically generate a temporary indicator variable. This variable acts as a flag, signaling whether a record currently being processed originated from a specific input source referenced in the **SET**, **MERGE**, or **UPDATE** statements. This capability is indispensable for ensuring data integrity, performing conditional processing, or simply documenting the lineage of every observation in the consolidated output dataset.

By effectively utilizing **IN=**, analysts gain granular control over their data flow. For example, if you are merging customer lists from two different regions, the **IN=** option immediately tells you which region contributed each record, allowing for subsequent filtering, classification, or the addition of descriptive variables directly into the new output file. Understanding and implementing this option is fundamental for robust and efficient data management in SAS programming.

2. Technical Definition and Purpose of IN=

The **IN=** option is specified directly following the name of an input dataset within the parentheses of a SET statement or **MERGE** statement, followed by an equals sign and a chosen variable name. This syntax instructs SAS to create a temporary, binary indicator variable—a Boolean variable—that exists only during the execution of that specific DATA step.

The variable created by **IN=** holds a numeric value of either 1 or 0. Specifically, if the current observation being processed originated from the data specified alongside the **IN=** option, the variable is assigned a value of 1 (representing true). Conversely, if the observation came from any other dataset listed in the SET statement or **MERGE** statement, the variable receives a value of 0 (representing false). This mechanism provides a simple yet powerful logical check for data origin.

It is crucial to note that the indicator variable created by **IN=** is temporary by default; it is not automatically written to the output dataset. To retain this information, the programmer must explicitly utilize conditional logic (such as **IF-THEN/ELSE** statements) within the DATA step to create a new, permanent variable based on the value of the temporary **IN=** variable. This permanent variable can then be included in the final output dataset, ensuring the source information is preserved for future analysis.

3. Practical Application: Appending and Identifying Sources

One of the most common applications of the **IN=** option occurs when appending (concatenating) two or more datasets using the SET statement. When concatenating, SAS processes all observations from the first listed dataset, followed sequentially by all observations from the second, and so on. Without the **IN=** option, the resulting consolidated dataset appears as a single unit, making it impossible to determine the original source of any specific row without prior knowledge or manual tracking. This is often necessary for maintaining accountability in large-scale data systems.

The ability to track the source becomes critically important in quality control, auditing, and heterogeneous data integration tasks. For instance, if you merge historical data with current transactional data, using **IN=** allows you to flag all current transactions immediately. This flag can then be used downstream to apply different business rules, perform differential statistical analysis, or subset the data based on the observation vintage.

Furthermore, while the **IN=** option is most intuitive when used with the SET statement for concatenation, it is equally vital in merging operations using the **MERGE** statement. When joining datasets, **IN=** helps identify whether a match was found for a given observation in each input dataset, a capability often leveraged to simulate complex join types (e.g., left, right, or inner joins) by filtering the resulting merged dataset based on the resulting **IN=** flags.

4. Setting Up the Example Datasets

To demonstrate the utility of the **IN=** option, we will use a straightforward example involving two distinct datasets containing performance statistics for basketball teams. Imagine we have records for teams in the Eastern and Western conferences of a league, and we need to combine them into a single file while maintaining a clear indicator of their original conference affiliation.

The first step involves creating the two input datasets, `east_data` and `west_data`, using the DATA step and **DATALINES** statement. Both datasets share the same structure: a character variable for the team name (`team`) and a numeric variable for their recorded points (`points`). This homogeneous structure ensures that concatenation proceeds smoothly, focusing the demonstration purely on the source identification problem.

The code block below illustrates the creation and initial viewing of these two datasets. Note the use of the **PROC PRINT** statement, which is essential for verifying that the input data structures are correct before proceeding with any manipulation. This verification step is a fundamental practice in SAS programming to prevent errors in subsequent steps.

```
/*create East dataset*/
data east_data;
```

input team $ points;
datalines;
Celtics 22
Pistons 14
Nets 35
Hornets 19
Magic 22
;
run;

/*create West dataset*/
data west_data;
input team $ points;
datalines;
Mavs 40
Rockets 39
Warriors 23
Lakers 19
Clippers 25
;
run;

/*view datasets*/
proc print data=east_data;
proc print data=west_data;

Obs	team	points
1	Celtics	22
2	Pistons	14
3	Nets	35
4	Hornets	19
5	Magic	22

Obs	team	points
1	Mavs	40
2	Rockets	39
3	Warriors	23
4	Lakers	19
5	Clippers	25

5. Step 1: Merging Data Without IN=

Before introducing the **IN=** option, let us first perform a simple concatenation of the two datasets using the SET statement. This process generates a single output dataset, `all_data`, which contains all rows from `east_data` followed sequentially by all rows from `west_data`. This operation is efficient for combining homogenous data structures but inherently obscures the origin of the data.

The syntax for this basic concatenation is minimalist, requiring only the SET statement to list the datasets in the desired order. While this successfully achieves the objective of combining the information, the resulting dataset lacks the crucial metadata needed to distinguish between observations that originated from the East versus the West. Every row appears identical in terms of its source, which restricts potential downstream analyses based on conference affiliation.

The following DATA step demonstrates the creation of the combined dataset, followed by a PROC PRINT to display the output. Observe that although the data is merged, there is no column identifying the original conference, highlighting the deficiency we aim to resolve.

```
/*create new dataset*/
data all_data;
set east_data west_data;
```

```
run;

/*view new dataset*/
proc print data=all_data;
```

Obs	team	points
1	Celtics	22
2	Pistons	14
3	Nets	35
4	Hornets	19
5	Magic	22
6	Mavs	40
7	Rockets	39
8	Warriors	23
9	Lakers	19
10	Clippers	25

As confirmed by the output, all observations from both input datasets have been successfully combined into `all_data`. However, subsequent analysis might require knowing the geographical conference, which necessitates the introduction of the **IN=** option for source tracking.

6. Step 2: Leveraging IN= for Source Identification (The Core Technique)

To resolve the ambiguity regarding data origin, we introduce the **IN=** option during the concatenation process. We will apply the **IN=** option to the `west_data` dataset, creating a temporary indicator variable, which we will arbitrarily name `i`. This variable name is chosen by the programmer and need not be declared beforehand.

The crucial modification occurs within the parentheses of the `west_data` reference in the SET statement: `west_data(in=i)`. During the execution of the DATA step, the variable `i` will be set to 1 if the current observation originated from `west_data`, and 0 otherwise (meaning it came from `east_data`, as it is the only other dataset listed).

We then use standard **IF-THEN/ELSE** logic to create a permanent, character variable called `conf` (for conference). If `i` is true (i.e., `i = 1`), which SAS evaluates concisely as `IF i THEN`, we assign the value 'West' to `conf`. If `i` is false (i.e., `i = 0`), we execute the **ELSE** clause and assign the value 'East'. This resulting `conf` variable is retained in the final output dataset, successfully tagging

every observation with its source conference.

```
/*create new dataset*/
data all_data;
set east_data west_data(in=i);
if i then conf='West';
else conf='East';
run;

/*view new dataset*/
proc print data=all_data;
```

Obs	team	points	conf
1	Celtics	22	East
2	Pistons	14	East
3	Nets	35	East
4	Hornets	19	East
5	Magic	22	East
6	Mavs	40	West
7	Rockets	39	West
8	Warriors	23	West
9	Lakers	19	West
10	Clippers	25	West

The resulting output clearly demonstrates the efficacy of the **IN=** option. We now have a new column, `conf`, which accurately indicates whether the observation came from the 'East' or 'West' dataset, thereby preserving the critical source metadata throughout the data integration process.

7. Detailed Breakdown of the IN= Logic

To fully appreciate the mechanism, it is important to understand how SAS iterates through the DATA step when using the SET statement with **IN=**. SAS reads the datasets sequentially. In our example, it first reads all observations from `east_data`. During this phase, since the **IN=i** option was applied specifically to `west_data`, the indicator variable `i` is automatically initialized and set to 0 for every record read from `east_data`.

When SAS finishes processing `east_data` and begins reading `west_data`, the variable `i` is

immediately flipped to 1. For every subsequent observation read from `west_data`, `i` remains 1. The conditional logic within the DATA step interprets these values:

If `i` is 0 (False, coming from `east_data`), the **ELSE** condition executes, assigning `conf='East'`.
If `i` is 1 (True, coming from `west_data`), the **IF i THEN** condition executes, assigning `conf='West'`.

This simple Boolean variable mechanism is the core strength of the **IN=** option, allowing programmers to easily differentiate between data sources without manual intervention or complex pre-processing of the input datasets. It ensures that data provenance is captured instantly as the data is streamed into the new output file.

8. Advanced Usage: Conditional Flagging

While the previous example utilized the **IF-THEN/ELSE** structure to assign descriptive character values ('East'/'West'), the **IN=** option can also be used in a more streamlined manner for simple flagging, especially when only one dataset source needs explicit identification. This is achieved by relying solely on the **IF-THEN** structure, without an **ELSE** clause.

Consider a scenario where we only need to flag observations originating from the Eastern Conference. We apply the **IN=** option to `east_data`, creating a variable (say, `i`) that is 1 for Eastern teams and 0 for Western teams. We then use an **IF-THEN** statement to create a new character variable, `east_conf`, which only receives a marker (e.g., an asterisk `*`) when the condition is met.

If the condition `IF i THEN` is true (meaning the record came from `east_data`), the variable `east_conf` is assigned the value `*`. If the condition is false (the record came from `west_data`), the variable `east_conf` retains its default missing value (a blank space for character variables), effectively creating a sparse flag column that minimizes output file size while still retaining necessary information.

```
/*create new dataset*/
data all_data;
set east_data(in=i) west_data;
if i then east_conf='*';
run;

/*view new dataset*/
proc print data=all_data;
```

Obs	team	points	east_conf
1	Celtics	22	*
2	Pistons	14	*
3	Nets	35	*
4	Hornets	19	*
5	Magic	22	*
6	Mavs	40	
7	Rockets	39	
8	Warriors	23	
9	Lakers	19	
10	Clippers	25	

The resulting output clearly shows the creation of the `east_conf` column. Observations originating from `east_data` are marked with an `*`, while observations from `west_data` have a blank value. This demonstrates the flexibility of using the **IN=** option to generate simple binary flags for filtering, verification, or subsequent processing steps based on highly specific criteria.

9. Conclusion and Further Resources

The **IN=** option is an indispensable tool in the SAS programmer's arsenal, particularly when dealing with complex data integration and data lineage challenges. By automatically generating a temporary Boolean variable during the DATA step, it allows for immediate, programmatic identification of the source dataset for every observation read. This functionality is essential for maintaining data quality and auditability in professional environments.

Whether used for simple concatenation flagging or within intricate merging logic, mastering the **IN=** option streamlines data preparation and enhances the traceability of analytic results. Always remember that the indicator variable itself is temporary; its value must be transferred to a permanent variable using conditional logic (**IF-THEN/ELSE**) if it is to be retained in the final output file.

For those looking to expand their expertise in data manipulation within the SAS environment, exploring related options and procedures is highly recommended. The following concepts often complement the use of **IN=** and offer alternative methods for data management:

The **FIRST.** and **LAST.** variables for boundary checking in grouped data processing, often used similarly to **IN=** but based on grouped variables rather than input source.

The use of the **MERGE** statement with the **BY** clause for matching records across datasets, where **IN=** flags can determine match success.
Advanced filtering techniques using the **WHERE** statement in conjunction with source flags created using **IN=**.

Further tutorials are available to deepen your understanding of these and other common tasks in SAS programming.