Learning SAS: Concatenating Datasets with the SET Statement

Authored by

Mohammed looti

May 16, 2026

Mastering the SAS `SET` Statement for Vertical Data Concatenation

The **SET statement** is one of the most essential and versatile commands available in **SAS** programming, serving as the primary mechanism for reading data from existing **SAS datasets** into a **DATA step**. When utilized with multiple dataset names, its primary function shifts to data concatenation. This process involves vertically stacking the rows--or **observations**--of one dataset directly beneath those of another, creating a single, longer dataset. This capability is absolutely crucial for data preparation when input information is fragmented across several identically structured files or tables that need to be aggregated for comprehensive analysis.

It is vital to distinguish concatenation using the **SET statement** from other data combination techniques, such as the `MERGE` statement. Merging combines datasets horizontally based on a shared key or identifier, aligning records side-by-side. In contrast, the **SET statement** performs a sequential, vertical combination: it reads all records from the first dataset, then immediately follows with all records from the second, and so on. The resulting dataset retains the structure (variables) of the input files while accumulating all their **observations**.

The order in which the input datasets are listed within the **SET statement** is determinative, directly influencing the sequence of **observations** in the final combined dataset. This feature makes it exceptionally powerful for scenarios requiring data aggregation, such as compiling quarterly financial reports, combining experimental results from different cohorts, or compiling demographic surveys collected in distinct phases. By employing this method, analysts can efficiently establish a unified dataset, significantly simplifying subsequent statistical analysis and reporting workflows.

Defining the Syntax for Concatenating Multiple SAS Datasets

Implementing data concatenation in **SAS** follows the standardized architecture of a **DATA step**. Every **DATA step** typically begins with the **DATA statement**, which names the new dataset being created (the output file), and concludes with the **RUN statement**, which commands **SAS** to execute the step.

The core functionality for concatenation resides within the **SET statement**. To combine multiple input datasets, one simply lists their names, separated by spaces, immediately following the `SET` keyword. **SAS** processes this command sequentially: it reads and outputs all **observations** from the first dataset, then moves to the second dataset and reads all its **observations**, continuing this process until every specified dataset has been fully incorporated into the new output file.

The basic syntax for this vertical combination, demonstrated below, clearly illustrates how simple it is to instruct **SAS** to stack data from multiple sources.

data new_data;

```
set data1 data2 data3;
run;
```

In this specific example, a new output dataset named `new_data` is generated. This dataset will first hold every record from the source file `data1`, followed sequentially by all records from `data2`, and finally, all records from `data3`. This sequential reading ensures that the combined dataset perfectly preserves the integrity and original order of the records within each source dataset while achieving efficient vertical appending.

Illustrative Case Study: Consolidating Basketball Team Data

To grasp the practical benefits of using the **SET statement** for combining multiple datasets, let us examine a typical scenario within sports analytics. Suppose a researcher is monitoring the performance statistics of players within a professional basketball league. For administrative reasons, or perhaps due to decentralized data collection, the individual player scores for different teams have been logged into separate **SAS datasets**.

The ultimate objective is to merge these separate team datasets into one single, cohesive, and comprehensive dataset. A consolidated file would immensely facilitate league-wide analysis, enabling tasks such as calculating the overall average points scored by all players, efficiently identifying the league's top scorers without filtering by team, or performing comparative statistical analyses between teams from a unified data repository. This example provides a clear demonstration of how to achieve this crucial data aggregation using the power of **SAS**.

The process begins by establishing two distinct input datasets, each containing scoring data for players from a specific team. Once these datasets are prepared, we will then apply the **SET statement** in a subsequent **DATA step** to seamlessly combine these two sources into a single, analysis-ready unit.

Step 1: Creating the Initial Source Datasets in SAS

We must first generate our necessary source datasets. We will start by creating `data1`, which will contain the points scored by players on "Team A." The **DATA statement** initiates the creation of this new file. The subsequent **INPUT statement** is used to define the **variables**: `team` is defined as a character **variable** (denoted by the `$`), and `points` is defined as a standard numeric **variable**. Finally, the **DATALINES statement** signals that the raw data for these variables will be provided immediately within the code block.

```
/*create first dataset*/
data data1;
```

```
input team $ points;
datalines;
A 12
A 15
A 16
A 21
A 22
;
run;

/*view dataset*/
proc print data=data1;
```

Upon successful execution, **SAS** creates `data1`, populated with five **observations**, each representing a player's points from Team A. The subsequent **PROC PRINT** procedure is used specifically to display the contents of `data1`, serving as a critical verification step to ensure data integrity and successful dataset creation before proceeding to the combination phase.

Obs	team	points
1	A	12
2	A	15
3	A	16
4	A	21
5	A	22

Next, we repeat this process to create the second dataset, `data2`, which holds the statistics for players on "Team B." It is important that `data2` maintains the exact same structure as `data1`, ensuring compatible **variables** (`team` and `points`) with identical data types. While the **SET statement** can gracefully manage structural differences, starting with consistent datasets simplifies the concatenation process significantly.

```
/*create second dataset*/
data data2;
input team $ points;
datalines;
B 16
B 22
```

```
B 25
B 29
B 30
;
run;

/*view dataset*/
proc print data=data2;
```

After this step executes, `data2` is created, containing five new **observations** specific to Team B's scoring data. We use the **PROC PRINT** statement once more to display `data2` and confirm its contents. With both distinct yet structurally uniform datasets established, all prerequisites for demonstrating the powerful vertical combination capability of the **SET statement** have been met.

Obs	team	points
1	B	16
2	B	22
3	B	25
4	B	29
5	B	30

Step 2: Implementing the `SET` Statement to Combine Data Vertically

With the individual datasets, `data1` and `data2`, successfully created and verified, we can now proceed to the core task: combining them into a single, unified dataset. This process leverages the efficiency of the **SET statement**. By listing both `data1` and `data2` in sequence, we explicitly instruct **SAS** to concatenate them, stacking the rows from the second file directly below those of the first.

The following **DATA step** is designed to create a new dataset named `data3`. Due to the sequential nature of the `SET` command, `data3` will first receive all five **observations** from `data1`, immediately followed by all five **observations** from `data2`. This results in a longer dataset that preserves all original columns but contains double the number of rows, effectively aggregating the player statistics across both teams.

```
/*create new dataset that combines two datasets*/
data data3;
```

```
set data1 data2;
run;

/*view new dataset*/
proc print data=data3;
```

After executing this code block, **SAS** successfully generates `data3`. The output from the **PROC PRINT** procedure confirms the concatenation: the dataset contains a total of ten **observations**. The first five rows correspond precisely to the data from Team A (`data1`), and the subsequent five rows correspond to the data from Team B (`data2`). This clearly illustrates the seamless vertical stacking and sequential reading performed by the **SET statement**.

Obs	team	points
1	A	12
2	A	15
3	A	16
4	A	21
5	A	22
6	B	16
7	B	22
8	B	25
9	B	29
10	B	30

The resulting **dataset**, `data3`, now provides a complete and unified view of all player scoring records across both Team A and Team B. This aggregation is fundamental for simplifying subsequent comprehensive league analysis or generating reports that must span multiple source files.

Advanced Considerations: Handling Variable Discrepancies

While basic concatenation using the **SET statement** works best with identically structured datasets, it is important to understand how **SAS** manages situations where input datasets may have structural differences. A common challenge arises when the datasets being combined possess non-identical **column names** or when certain **variables** exist in one input file but are absent in another.

SAS intelligently handles these discrepancies during concatenation. If a **variable** is present in one

source dataset but not in the others, the **SET statement** automatically incorporates that **variable** into the new output dataset. For the **observations** originating from the dataset where that column was missing, **SAS** assigns a missing value. Specifically, character **variables** receive blank spaces, and numeric **variables** are assigned the standard numeric missing value (a period, `.`). This automatic imputation ensures that all records are preserved, regardless of minor structural variations.

Furthermore, when combining datasets, the attributes--such as the data type, length, format, and label--of the **variables** in the new output dataset are determined by the attributes defined in the first input dataset listed in the **SET statement** that contains that particular variable. If a variable possesses differing lengths across the input datasets (e.g., a character field is 10 bytes long in one file and 20 bytes in another), it is highly recommended practice to explicitly define the desired attributes using `LENGTH` or `ATTRIB` statements within the **DATA step** to avoid potential data truncation or unexpected behavior during the concatenation process.

Conclusion and Next Steps in SAS Data Management

The **SET statement** is an indispensable component of data manipulation in **SAS**, particularly celebrated for its capacity to perform seamless vertical concatenation of multiple datasets. By fully understanding its sequential processing nature and how **SAS** manages structural differences between input files, users gain the ability to efficiently consolidate disparate data sources into a single, cohesive unit, which is the foundational step for any complex statistical analysis or reporting task.

The detailed basketball example effectively demonstrated the straightforward mechanics of combining two simple datasets, resulting in the vertical stacking of **observations** to produce a comprehensive data view. This fundamental knowledge is critical for any analyst or programmer working extensively with **SAS** when data is distributed across multiple files. Mastery of the **SET statement** ensures more robust and streamlined data management and analysis workflows.

Additional Resources

The following tutorials explain how to perform other common data management and analysis tasks in **SAS**: