# How to Easily Delete Rows in SAS: 3 Step-by-Step Examples

Authored by

**stats writer**

December 1, 2025

The ability to efficiently manage and manipulate data sets is fundamental to effective statistical programming in SAS. One of the most common requirements data analysts face is the selective removal of unwanted observations (rows) based on specific criteria. While some programming languages use explicit SQL commands like `DELETE FROM`, SAS typically utilizes the powerful DATA Step in conjunction with the dedicated DELETE Statement.

This comprehensive guide explores three distinct, yet essential, techniques for removing observations from a data set using SAS. We will cover scenarios ranging from deleting rows based on a single condition to implementing complex compound conditional logic involving multiple variables. Each method is illustrated with practical code examples and clear explanations of the underlying logic, ensuring you can apply these techniques immediately to your own data preparation workflows.

## Understanding Row Deletion in the SAS DATA Step

In the SAS environment, removing rows is typically handled during the creation of a new data set using the DATA Step. Unlike transactional databases where rows are deleted in place, SAS processes the input data set observation by observation, and if a condition is met, the observation is explicitly told not to write to the output data set. This process uses the `SET` statement to read the original data and the DELETE Statement within an `IF-THEN` structure to implement the filtering rule.

The core concept revolves around specifying which rows should be excluded from the new output table. When the DELETE Statement is executed within the DATA Step, it immediately stops processing the current observation and prevents it from being written to any output data sets defined in that step. The step then proceeds to the next observation. This is a highly efficient way to filter data, as it minimizes resource usage by only writing the required observations.

Alternatively, some users choose to use a DELETE Statement implicitly by using the `WHERE` statement within the `SET` statement or a `PROC` (like `PROC SQL`). However, for complex conditional logic that might involve temporary variables or calculations, the `IF-THEN DELETE` approach within the DATA Step remains the most versatile and robust technique for controlling data flow.

## Core Methods for Conditional Deletion

The approach you choose for row deletion depends entirely on the complexity of the filtering criteria. Below are the three primary methods, demonstrating how conditional logic is applied to select observations for deletion.

Here are the three most common ways to delete rows in SAS:

**Method 1: Delete Rows Based on One Condition**

```
data new_data;
set original_data;
if var1 = "string" then delete;
run;
```

This method employs a simple `IF-THEN DELETE` structure where the DELETE Statement is triggered only when a single condition involving a variable (`var1`) is met. This is ideal for straightforward filtering needs, such as removing all records belonging to a specific category or exceeding a certain threshold value.

**Method 2: Delete Rows Based on Several Conditions (AND Logic)**

```
data new_data;
set original_data;
if var1 = "string" and var2 < 10 then delete;
run;
```

When deletion must be conditional on the simultaneous truth of multiple criteria, we utilize the logical operator `AND`. This ensures that an observation is only removed if **all** specified conditions (e.g., `var1` equals "string" AND `var2` is less than 10) are true for that row. This method is essential for narrow, specific filtering tasks.

**Method 3: Delete Rows Based on One of Several Conditions (OR Logic)**

```
data new_data;
set original_data;
if var1 = "string" or var2 < 10 then delete;
run;
```

The `OR` logical operator allows for a broader deletion scope. An observation will be deleted if at least one of the specified conditions is true (e.g., `var1` equals "string" OR `var2` is less than 10). This is useful when you need to clean up data by removing observations that fall into any one of several unacceptable categories.

## Setting Up the Sample Data Set

To demonstrate these three conditional deletion methods, we will use a small sample data set named `original_data`. This data set simulates player statistics, including their team, position, and

points scored. Reviewing the initial structure is crucial before applying any deletion criteria.

The data set contains three variables: `team` (character), `position` (character), and `points` (numeric). We use the `DATALINES` statement within the DATA Step to quickly input these observations. Subsequently, the `PROC PRINT` procedure is used to display the contents of the newly created table.

The following examples show how to use each method with the following dataset in SAS:

```
/*create dataset*/
data original_data;
input team $ position $ points;
datalines;
A Guard 15
A Guard 19
A Guard 22
A Forward 25
A Forward 27
B Guard 11
B Guard 13
B Forward 19
B Forward 22
B Forward 26
;
run;

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

| Obs | team | position | points |
|---|---|---|---|
| 1 | A | Guard | 15 |
| 2 | A | Guard | 19 |
| 3 | A | Guard | 22 |
| 4 | A | Forward | 25 |
| 5 | A | Forward | 27 |
| 6 | B | Guard | 11 |
| 7 | B | Guard | 13 |
| 8 | B | Forward | 19 |
| 9 | B | Forward | 22 |
| 10 | B | Forward | 26 |

## Example 1: Deleting Rows Based on a Single Criterion

The first practical scenario involves applying a filter based on a single variable's value. In this example, our objective is to remove all observations associated with Team "A" from the `original_data` set, effectively retaining only records for Team "B". This is achieved by checking the value of the `team` variable against the specific criterion "A" and invoking the DELETE Statement if the condition is true.

The use of the single condition is the simplest form of row filtering in SAS. The structure `IF team = "A" THEN DELETE;` clearly instructs the DATA Step to discard any row where the `team` variable matches the string literal "A". This demonstrates the power of explicit conditional control over the data writing process.

The following code shows how to delete all rows from the dataset where **team** is equal to "A."

```
/*create new dataset*/
data new_data;
set original_data;
if team = "A" then delete;
run;

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

| Obs | team | position | points |
|---|---|---|---|
| 1 | B | Guard | 11 |
| 2 | B | Guard | 13 |
| 3 | B | Forward | 19 |
| 4 | B | Forward | 22 |
| 5 | B | Forward | 26 |

Notice that all rows where **team** was equal to "A" have been deleted, leaving only the five observations pertaining to Team B.

## Example 2: Deleting Rows Based on Conjunctive (AND) Criteria

More intricate data cleaning often requires combining multiple conditions using the logical operator `AND`. This technique ensures that an observation is targeted for deletion only if *all* specified conditions are simultaneously satisfied. This rigorous form of conditional logic allows for highly targeted data removal.

In this second example, we aim to delete observations that meet two specific criteria: the team must be "A" AND the points scored must be less than 20. If an observation belongs to Team "A" but has points of 20 or higher, it will be retained. Conversely, if it scores less than 20 points but belongs to Team "B", it is also retained. The `AND` operator dictates a strict intersection of these conditions.

This method is essential for cleaning up specific subsets of data without impacting other records that may share one characteristic but not the other. This ensures precision when modifying a data set. The following code shows how to delete all rows from the dataset where **team** is equal to "A" *and* **points** is less than 20:

```
/*create new dataset*/
data new_data;
set original_data;
if team = "A" and points < 20 then delete;
run;

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

| Obs | team | position | points |
|---|---|---|---|
| 1 | A | Guard | 22 |
| 2 | A | Forward | 25 |
| 3 | A | Forward | 27 |
| 4 | B | Guard | 11 |
| 5 | B | Guard | 13 |
| 6 | B | Forward | 19 |
| 7 | B | Forward | 22 |
| 8 | B | Forward | 26 |

Notice that the two rows where **team** was equal to "A" *and* **points** was less than 20 have been deleted. All other observations, including the remaining Team A players and all Team B players, were retained because they failed the conjunction test.

## Example 3: Deleting Rows Based on Disjunctive (OR) Criteria

In contrast to the rigorous requirements of the `AND` operator, the `OR` operator broadens the scope of deletion. When employing the `OR` operator in your conditional logic, an observation is deleted if it satisfies *any* one of the specified conditions, resulting in a much larger reduction of the data set.

For this final example, we will delete all rows from the dataset where **team** is equal to "A" or **points** is less than 20. This means that any row belonging to Team A will be removed, regardless of points, and any row scoring less than 20 points will be removed, regardless of team. This is particularly effective for removing outliers or entire categories of data in one step.

Using the `OR` statement provides a high degree of flexibility but requires careful consideration, as it can inadvertently remove desirable data if the criteria are too broad. Understanding the difference between `AND` (intersection) and `OR` (union) is paramount for effective data manipulation in SAS. The following code shows how to delete all rows from the dataset where **team** is equal to "A" or **points** is less than 20:

```
/*create new dataset*/
data new_data;
set original_data;
if team = "A" or points < 20 then delete;
run;

/*view new dataset*/
```

```
proc print data=new_data;
```

| Obs | team | position | points |
|---|---|---|---|
| 1 | B | Forward | 22 |
| 2 | B | Forward | 26 |

Notice that the eight rows where **team** was equal to "A" or **points** was less than 20 have been deleted. Only observations that belonged to Team B *and* had points of 20 or higher remain. This demonstrates the sweeping effect of the `OR` operator in filtering data.

## Summary of SAS Deletion Techniques

Mastering the DELETE Statement within the DATA Step is essential for effective data preparation in SAS. These three examples cover the foundational techniques required to handle nearly any conditional row deletion requirement, whether simple or complex. Remember that the choice between `AND` and `OR` determines the strictness of your filter: `AND` narrows the selection, while `OR` broadens it.

When working with large datasets, always verify your deletion logic using the `PROC PRINT` procedure or comparable data viewing techniques to ensure that only the intended observations were removed. Furthermore, it is a best practice to always create a new data set (e.g., `new_data`) rather than attempting to modify the source data set in place, thus preserving the original raw data for auditing and verification purposes.

For those looking to expand their knowledge of data manipulation in SAS, similar conditional logic principles apply to other data management tasks, such as subsetting observations without using the DELETE Statement (e.g., using explicit `OUTPUT` statements or the `WHERE` clause). The techniques demonstrated here form the basis for much of advanced data processing.

The following tutorials explain how to perform other common tasks in SAS: