How to use SAS’s WHERE option with SET

Authored by

stats writer

November 19, 2025

The ability to efficiently manage and filter data is central to effective data analysis using SAS (Statistical Analysis System). One of the most fundamental techniques for achieving this is leveraging the WHERE option specifically within the SET statement. This powerful mechanism allows users to perform precise data subsetting, ensuring that only observations meeting specified criteria are carried forward into a newly created dataset. Understanding how to use the WHERE option effectively is essential for optimizing memory usage and streamlining subsequent data processing steps in SAS.

The core function of the WHERE option is to apply a filtering condition--or a logical expression--directly to the input dataset specified in the SET statement, before the observations are even read into the current data step buffer. This positioning makes it highly efficient compared to using a separate WHERE statement outside the data step boundaries. By utilizing combinations of variables, comparison operators (like greater than or equal to), and Boolean operators (such as AND or OR), analysts gain complete control over which records are included in the final output.

The WHERE option is attached directly to the input dataset name within the SET statement. This syntax allows SAS to perform the subsetting operation extremely early in the data processing cycle, significantly improving performance, especially when dealing with large datasets. When you use the WHERE option, you instruct the system to create a new dataset that includes only the rows from the specified input dataset where your defined logical conditions are successfully met.

We will explore two fundamental approaches for applying this technique: filtering based on a single variable criterion, and filtering based on complex conditions involving multiple variables and operators. These two methods form the basis of most data preparation tasks in SAS.

Method 1: Subsetting with a Single Condition

The simplest application of the WHERE option involves applying a single logical test to one variable. This is perfect when you need to isolate records based on a straightforward threshold, such as scores above a certain level or dates within a specific range. The syntax is clean and highly readable, making it easy to implement rapid filters.

Here is the standard structure for implementing a single condition filter:

```
data new_data;
set my_data (where = (points>20));
run;
```

In this example, a new dataset named new_data is created. Crucially, the SET statement instructs SAS to read observations from my_data only if the value in the points column is strictly greater than 20. All other observations are discarded during the read process, optimizing the creation of

the subsetted dataset.

Method 2: Subsetting with Multiple Conditions

For more complex data preparation tasks, analysts often need to combine multiple conditions using Boolean operators like OR or AND. This allows for highly specific data subsetting. Using the OR operator includes an observation if at least one of the defined conditions is true, while the AND operator requires all specified conditions to be true simultaneously.

The following structure demonstrates how to use the OR operator within the WHERE option to satisfy multiple criteria:

```
data new_data;
set my_data (where = (points>20 or team="Rockets"));
run;
```

This code generates new_data by drawing records from my_data. An observation is included if it meets one of two criteria: either the player's points score is greater than 20, or the team name matches "Rockets". This inclusive filtering approach is useful for capturing diverse groups of observations that share at least one defined characteristic.

It is important to note the versatility of the WHERE option here: simply replacing the OR operator with AND would drastically change the output, limiting the results only to observations that satisfy both criteria simultaneously. This careful selection of logical operators (part of the logical expression) is key to precise data manipulation.

Creating the Sample Dataset for Demonstration

To illustrate these methods practically, we will first create a sample dataset named my_data. This dataset represents basketball statistics, containing variables for team name, points scored, and assists made. This initial step is necessary to demonstrate the filtering capabilities of the WHERE option effectively.

```
/*create dataset*/
data my_data;
input team $ points assists;
datalines;
Mavs 22 10
Rockets 12 14
Spurs 29 8
Kings 13 10
```

Warriors 44 10
Heat 18 8
Magic 11 5
Pelicans 19 3
Blazers 12 8
;
run;

/*view dataset*/
proc print data=my_data;

Executing the code above creates the base dataset. We can inspect the resulting table using PROC PRINT to visualize all nine original observations before applying any subsetting logic.

Obs	team	points	assists
1	Mavs	22	10
2	Rockets	12	14
3	Spurs	29	8
4	Kings	13	10
5	Warriors	44	10
6	Heat	18	8
7	Magic	11	5
8	Pelicans	19	3
9	Blazers	12	8

Practical Application 1: Filtering by a Single Condition (Points > 20)

Our first practical example demonstrates Method 1, where we use the WHERE option to perform simple subsetting based on a numerical threshold. We aim to isolate all observations where the value of the points variable exceeds 20. This is a common operation when analyzing performance metrics or sales data.

The following syntax creates a new dataset, new_data, by applying the condition directly within the SET statement:

/*create new dataset*/
data new_data;

```
set my_data (where = (points>20));
run;

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

Upon reviewing the output, we confirm that only observations satisfying the condition points > 20 were retained. This included Mavs (22 points), Spurs (29 points), and Warriors (44 points). All other records were filtered out efficiently at the source.

Obs	team	points	assists
1	Mavs	22	10
2	Spurs	29	8
3	Warriors	44	10

This result clearly demonstrates how the WHERE option restricts the flow of observations, ensuring that the new dataset contains only the relevant subset of records.

Practical Application 2: Filtering by Multiple Conditions (Points > 20 OR Team = "Rockets")

Our second demonstration uses the complexity of Method 2, applying an OR condition to select observations that satisfy either of two distinct criteria. This scenario is useful when combining groups based on divergent qualifications.

We will create new_data containing all teams that scored more than 20 points OR the specific team "Rockets," regardless of their score. The syntax below executes this dual-condition filter:

```
/*create new dataset*/
data new_data;
set my_data (where = (points>20 or team="Rockets"));
run;

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

The resulting dataset includes not only the high-scoring teams (Mavs, Spurs, Warriors) but also the

Rockets, even though the Rockets only scored 12 points. This outcome confirms that the OR operator successfully included any observation that satisfied at least one part of the complex logical expression.

Obs	team	points	assists
1	Mavs	22	10
2	Rockets	12	14
3	Spurs	29	8
4	Warriors	44	10

Note that if we had used the AND operator instead (e.g., where = (points > 20 and team="Rockets")), the output dataset would be empty, as no team in the original data satisfies both conditions simultaneously.

Understanding Syntax and Efficiency

The key advantage of using the WHERE option within the SET statement, rather than a standalone WHERE statement later in the data step, is efficiency. When the option is placed on the SET statement, the filter is applied directly to the input file, which reduces the amount of data transferred and processed by SAS. This method is generally preferred for performance optimization, especially when working with vast amounts of data stored on disk.

Furthermore, the WHERE option supports a wide array of comparison and special operators, including IN, LIKE, and functions that can return boolean outcomes. This allows for extremely flexible and robust filtering criteria, making it an indispensable tool for data quality checks and preparation work.

Related Data Manipulation Techniques in SAS

While the WHERE option on the SET statement is excellent for early, efficient subsetting, SAS offers several other methods for data manipulation, depending on the required complexity and stage of processing. For instance, using a standard IF-THEN-DELETE or IF-THEN OUTPUT structure within the main data step allows for more complex conditional processing, often involving newly created variables or temporary calculations, which the WHERE option cannot access.

However, when the goal is simply to filter existing observations based on existing variable values, the syntax demonstrated here remains the quickest and most efficient standard practice in SAS programming. Mastering this specific technique ensures data analysts can swiftly prepare clean, targeted datasets for further statistical modeling and reporting.

For more advanced data preparation and analysis tasks in SAS, consider exploring topics such as:

How to handle missing data using various data step functions.
Techniques for merging and appending multiple datasets using the MERGE and SET statements.
Advanced uses of PROC SQL for complex querying and aggregation.