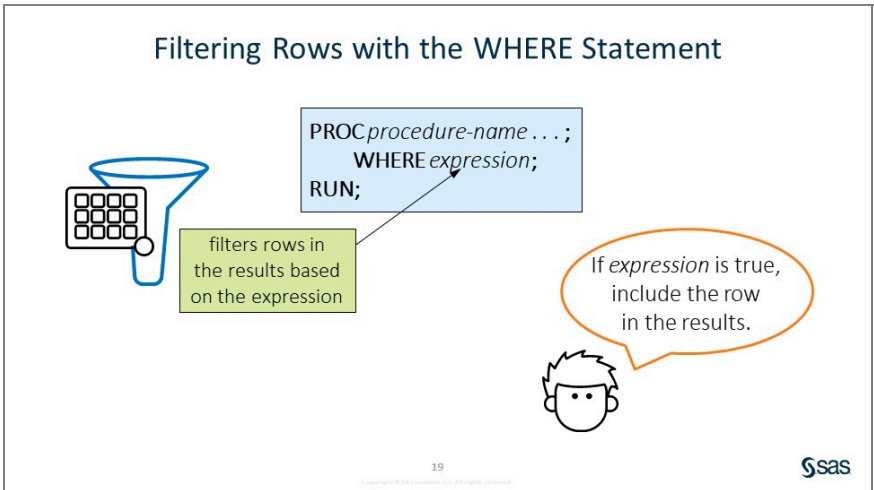
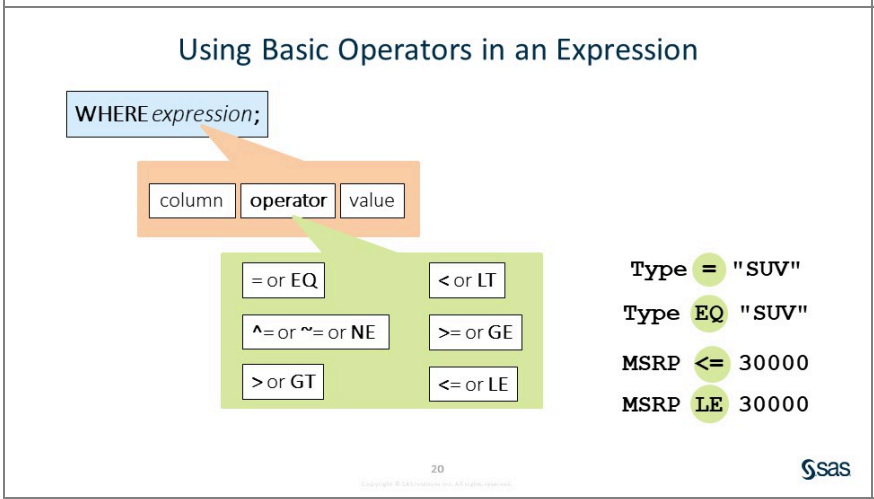
B3.2 - Filtering Rows

Where Statement

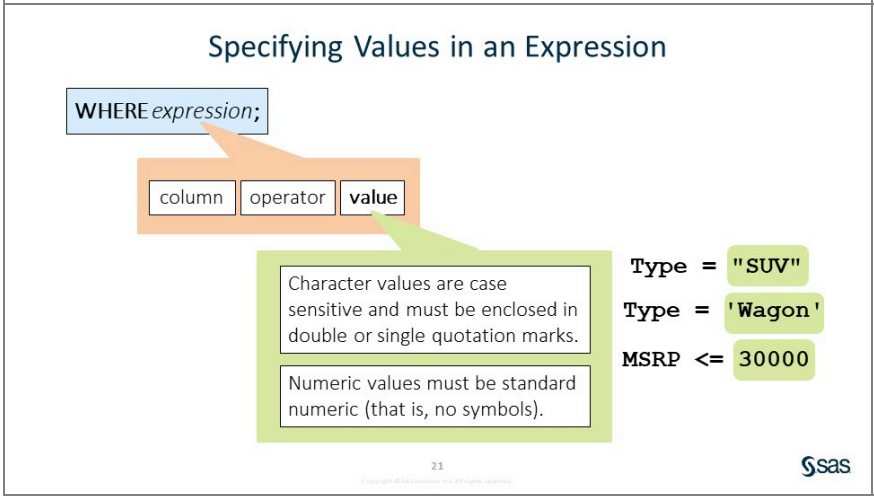


What if you want to filter the rows that appear in a PROC PRINT report? Or what if you only want to calculate summary statistics for a subset of the data based on a condition? You can use the powerful and flexible WHERE statement to subset your data. The WHERE statement can be used in PROC PRINT, MEANS, FREQ, UNIVARIATE and many others.

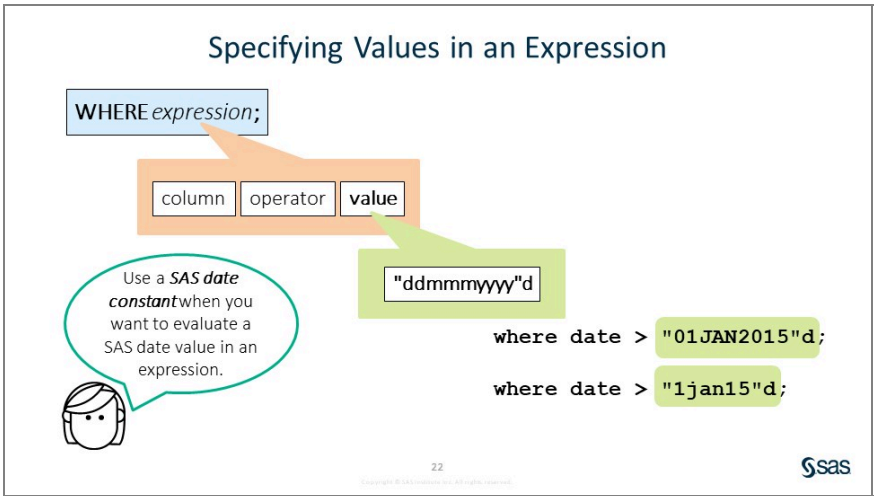


The WHERE statement consists of the keyword WHERE followed by one or more expressions. An expression tests the value of a column against a condition that you specify.

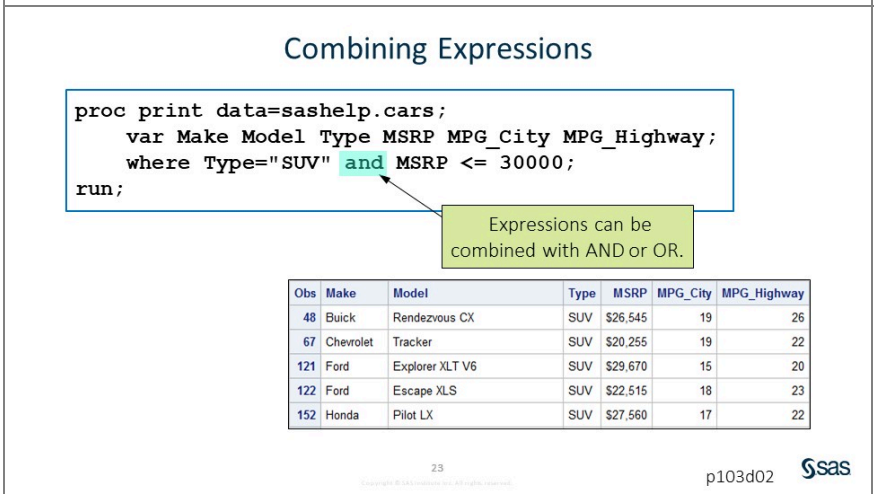
First, let's look at expressions that use basic operators. We can use any of these operators to compare the value of a column to a value that you specify. The expression evaluates as true or false for each row. You can use either the symbol or letters to represent these operators in an expression.



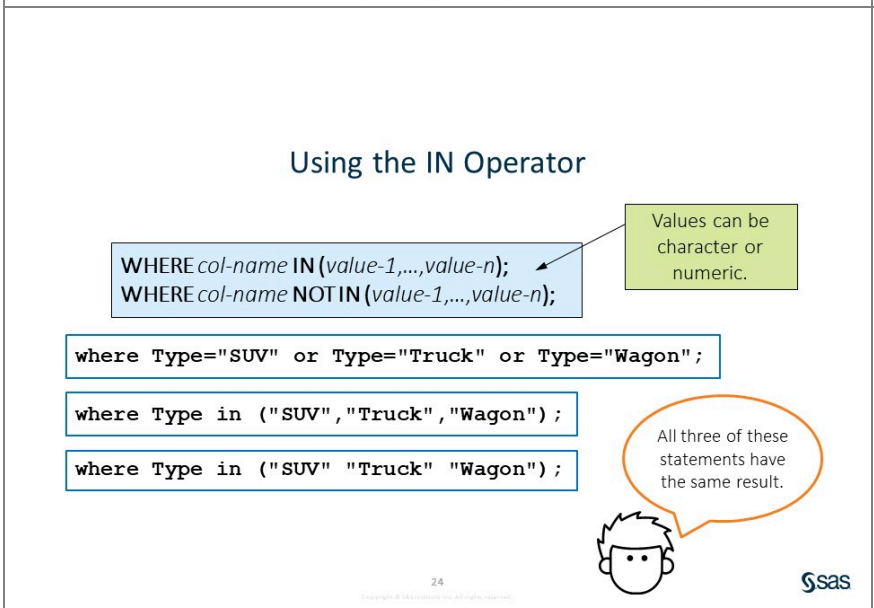
Character values are case sensitive and must be enclosed in double or single quotation marks. Numeric values are not enclosed in quotation marks and must be standard numeric values. In other words, you cannot include special symbols such as commas or dollar signs.



What about comparing the value of a column to a date? Remember that dates are stored as numeric values, so the expression is evaluated based on a numeric comparison. If you want to compare a date column to a fixed date, then you can use the SAS date constant notation. Type the date as a one- or two-digit day, a three-letter month, and two- or four-digit year, enclosed in quotation marks, followed by the letter D. SAS then turns the string date into the numeric equivalent in order to evaluate the expression.



You can also combine multiple expressions with the keywords AND or OR. In this example code, any rows that meet theses two conditions are included in the PROC PRINT results.



We've only begun to learn the ways you can use the WHERE statement to subset rows in a procedure. Let's look at several other examples of operators that enable you to do more complex filtering.

We've seen how the OR keyword can be used to provide multiple values, such as in this example. Notice that each condition has to include TYPE=. This can be tedious if there are several valid values that you would like to include. A more efficient approach in this scenario is to use the IN operator to compare to a list of values.

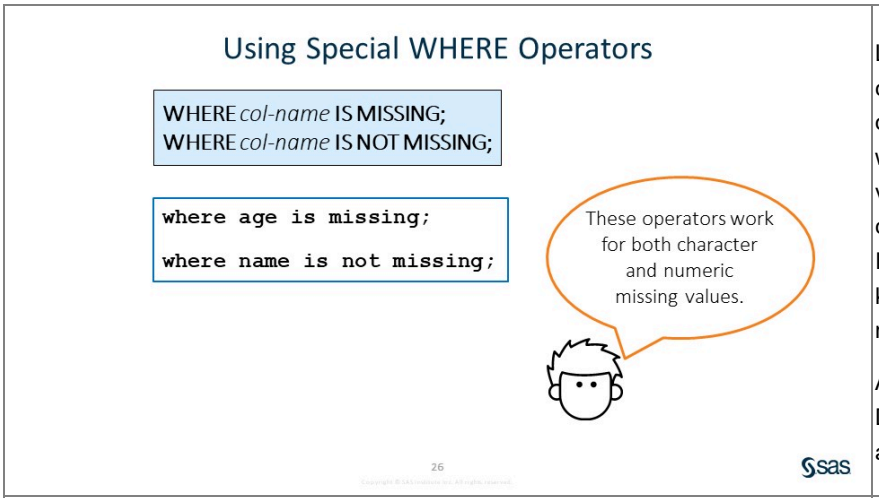
After typing the column name, we use the keyword IN, and in parentheses, list the values separated by commas or spaces. The IN operator works with both numeric and character values. Just keep in mind that character values are case sensitive and must be enclosed in quotation marks. We can also take advantage of the keyword NOT to reverse the logic of the IN operator.

Demo: Filtering Rows with Basic Operators

3d81bc9d5e8d%3FautoPlay%3Dfalse%26displayComments%3Dtrue%26displayTitle%3Dtrue%26displaySharing%3Dfalse%26displayNotes%3Dtrue%2(

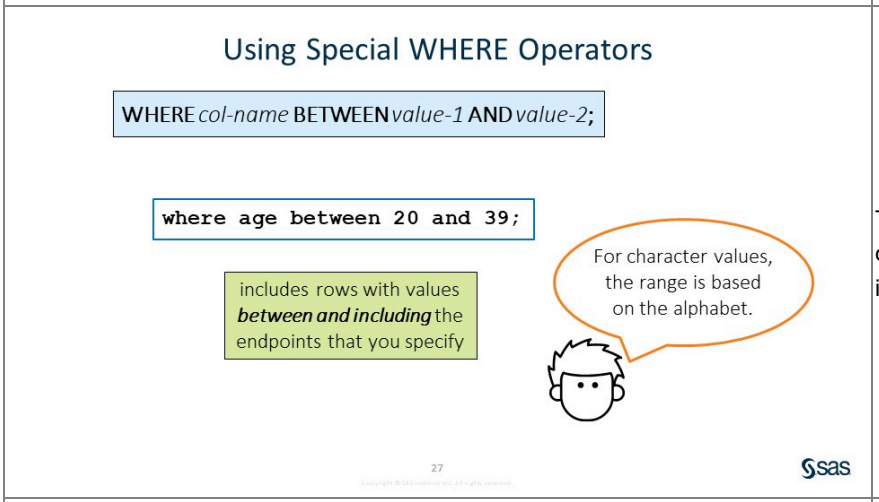
3_2 - Demo - Filtering Rows with Basic Operators.pdf (https://clemson.instructure.com/courses/237270/files/23073940/download?wrap=1)

Special WHERE Operators

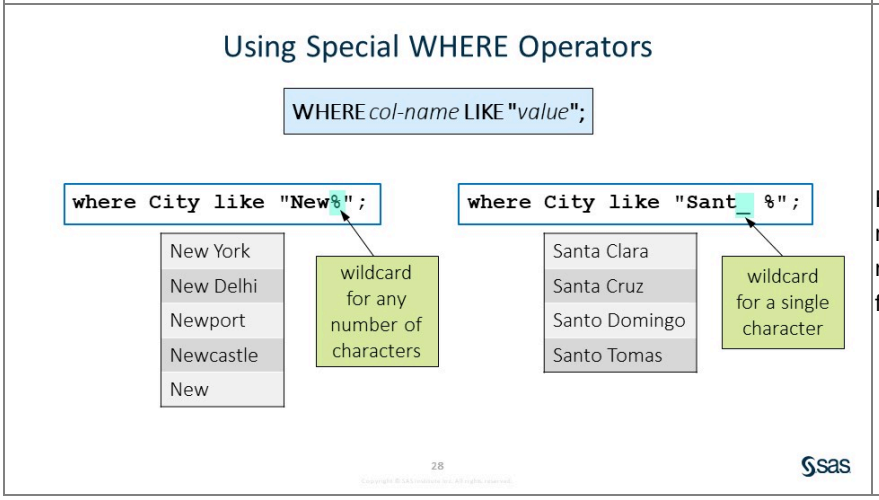


Let's talk about some special WHERE operators that you can use in expressions. Suppose you want to filter your data by missing values. You could write an expression where a column is equal to a period for numeric missing values or a space enclosed in quotation marks for a character missing value, but a simpler option is to use the IS MISSING or IS NOT MISSING special operator. These keywords can be used for either numeric or character missing values.

And if you happen to be working with data coming from a DBMS environment that distinguishes between missing and null values, then there is also an IS NULL operator.



The BETWEEN AND operator is handy for numeric and character ranges. The endpoints of the range are inclusive.



Finally, the LIKE operator enables us to do pattern matching. The percent symbol is a wildcard for any number of characters and the underscore is a wildcard for a single character.

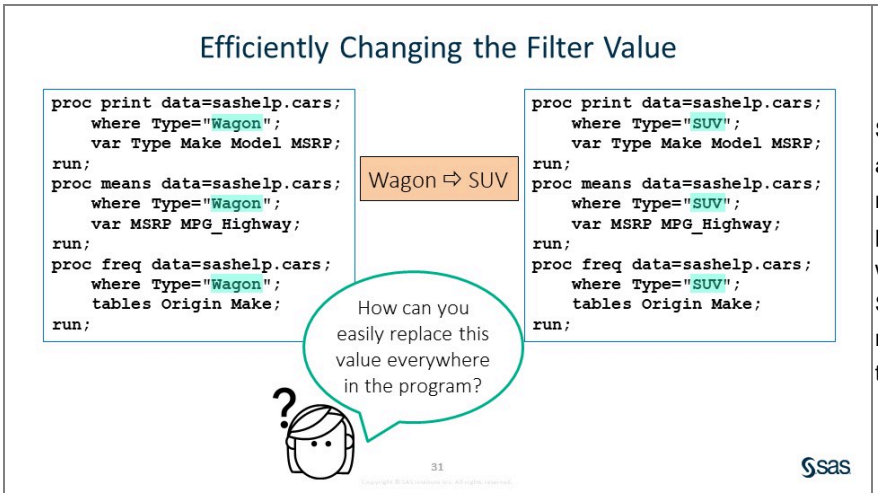
Activity 3.02

Open **p103a02.sas** from the **activities** folder and perform the following tasks:

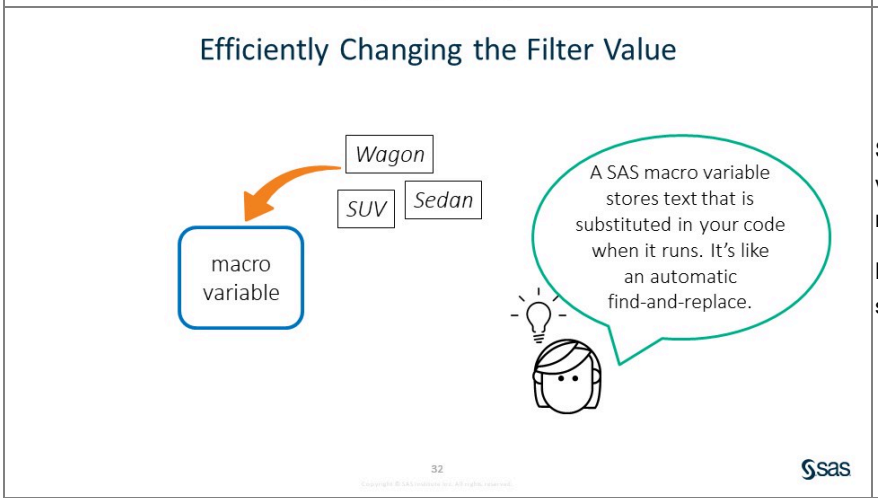
1. Uncomment each WHERE statement one at a time and run the step to observe the rows that are included in the results.
2. Comment all previous WHERE statements. Add a new WHERE statement to print storms that begin with Z. How many storms are included in the results?

Click here for Solution.

Creating SAS Macro Variables

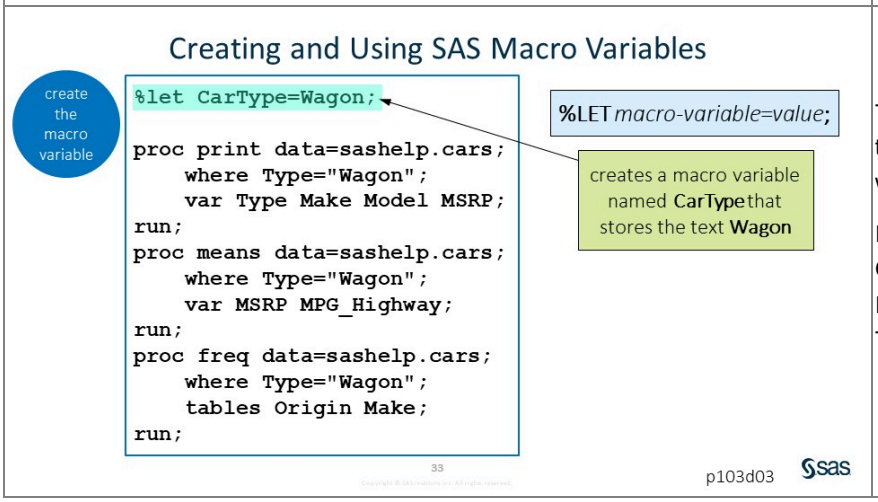


Suppose you have a program with multiple procedures, and you want to filter in each procedure to select only rows where the type of car is a wagon. You run the program, look at the results, and then decide that you want to see similar reports where the type of car is an SUV. Of course, find and replace is an option for modifying the program, but wouldn't it be nice to change that repeating value in one place?



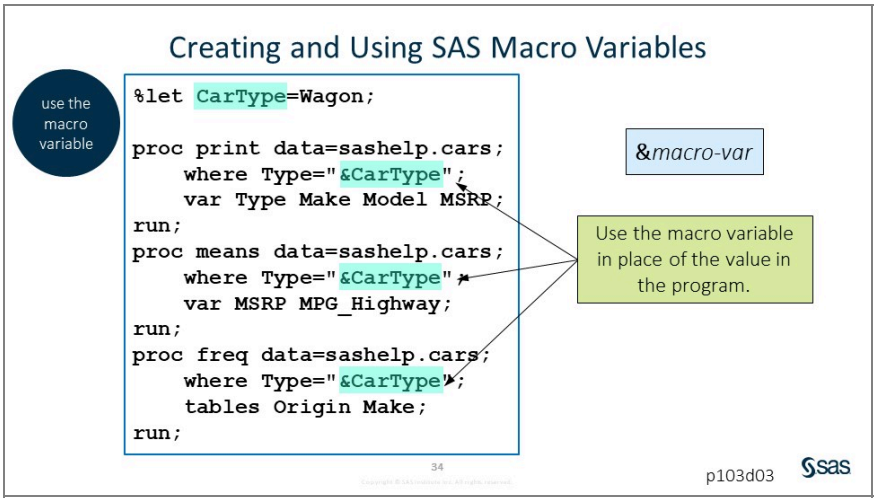
SAS macro variables enable you do just that. A macro variable stores text that is substituted in your code when it runs, like automatic find and replace!

Let's see how we can create and use a macro variable to solve this problem.

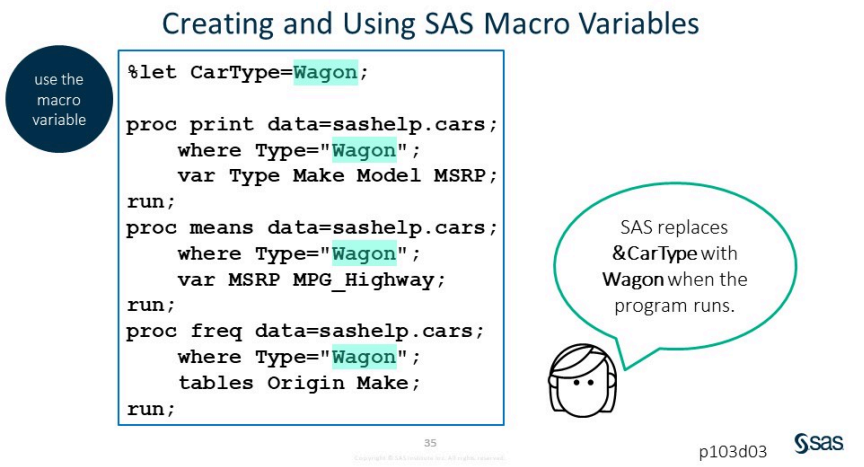


The first step is to create the macro variable, and we do that with the %LET statement. All macro statements begin with a % sign.

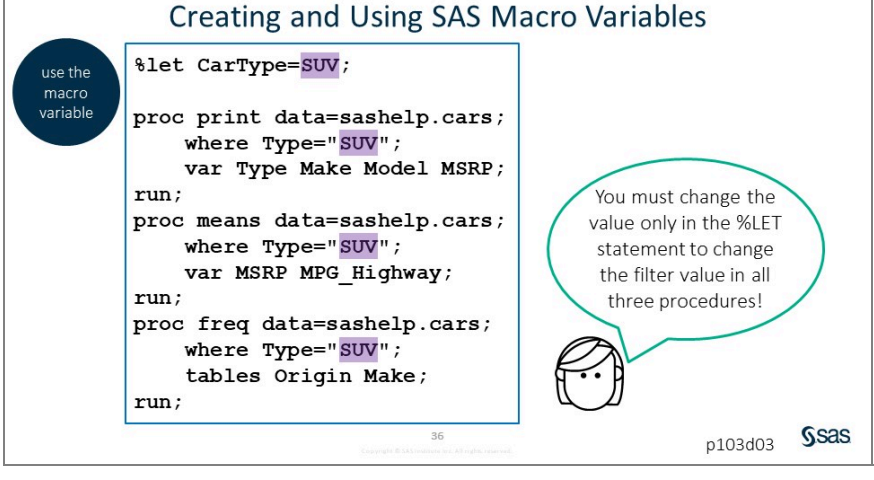
In this example, I'm creating a macro variable named **CarType**. After the equal sign, I provide the text string that I want the macro variable to store. I'll start with *Wagon*. The %LET statement ends with a semicolon.



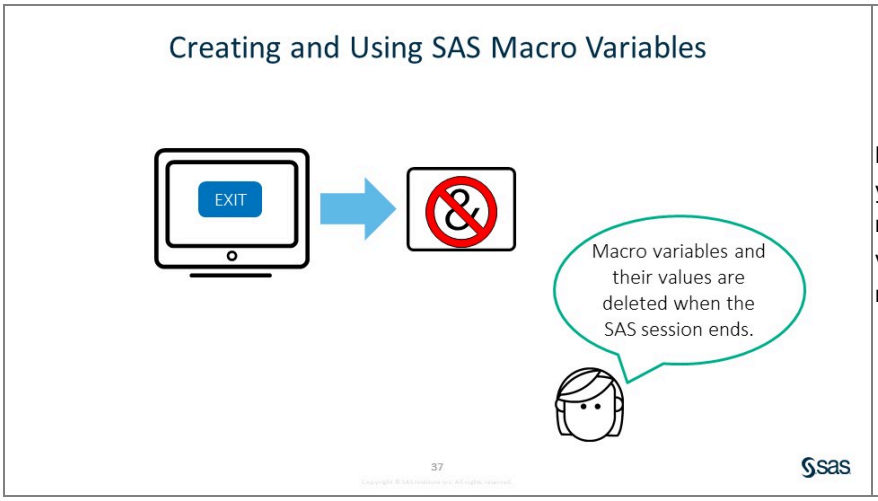
Now that I've created the macro variable, the next step is to use it in the program. In each place where I previously specified the value "*Wagon*", I'll specify the macro variable that holds the value: **CarType**. When you reference a macro variable in your code, you precede the name with an ampersand.



The ampersand triggers SAS to look up the text string stored in the **CarType** macro variable and replace it with *Wagon* before it executes the code.



Now if I want to run these procedures and change the value of the filter, I have to change the value only in the %LET statement!



Like libraries, macro variables are temporary, so when you exit SAS, they are deleted. If you have macro variable references in your program, you must create the macro variable at the beginning of your program before you reference it.

Demo: Filtering Rows Using Macro Variables

3_2 - Demo - Filtering Rows Using Macro Variables.pdf (https://clemson.instructure.com/courses/237270/files/23074693/download?wrap=1)
(https://clemson.instructure.com/courses/237270/files/23074693/download?download_frd=1)

Activity 3.04

Open **p103a04.sas** from the **activities** folder and perform the following tasks:

1. Change the value in the %LET statement from **NA** to **SP**.
2. Run the program and carefully read the log.
   Which procedure did not produce a report?
   What is different about the WHERE statement in that step?

Click here for Solution.