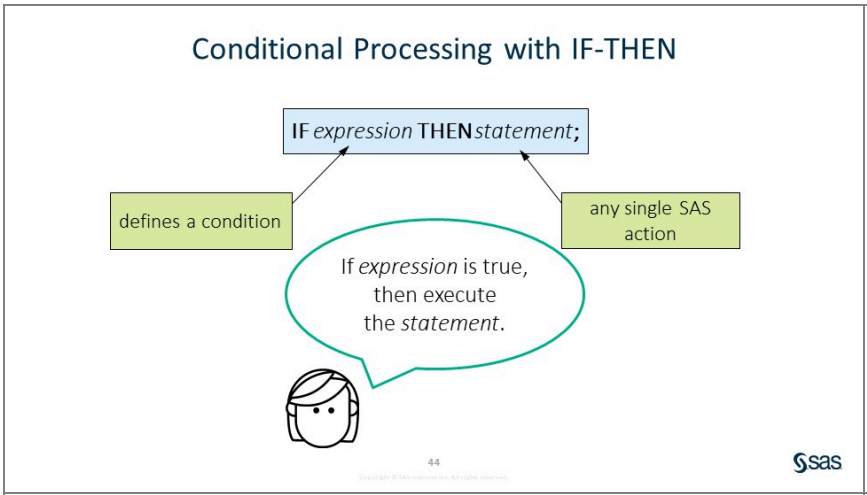
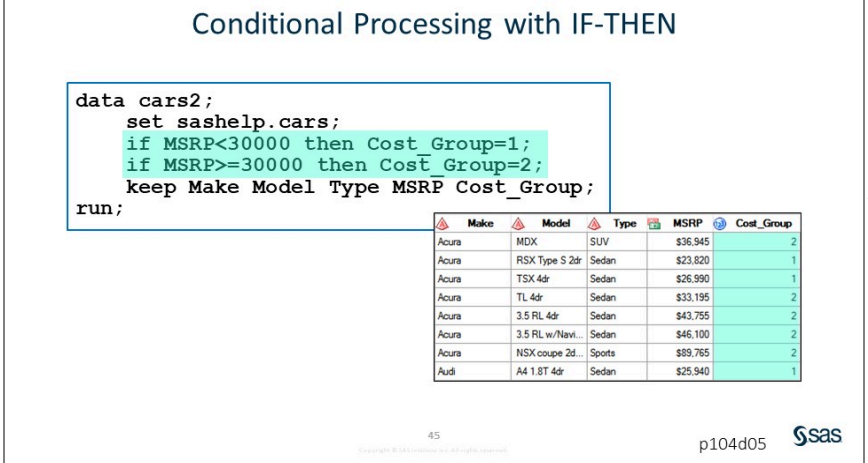
B4.3 - Conditional Processing

Conditional Processing with If-Then/Else



Often in the DATA step, we need to process data conditionally. In other words, if some condition is met, then execute one statement. If a different condition is met, then execute another statement. We can accomplish this using IF-THEN logic.

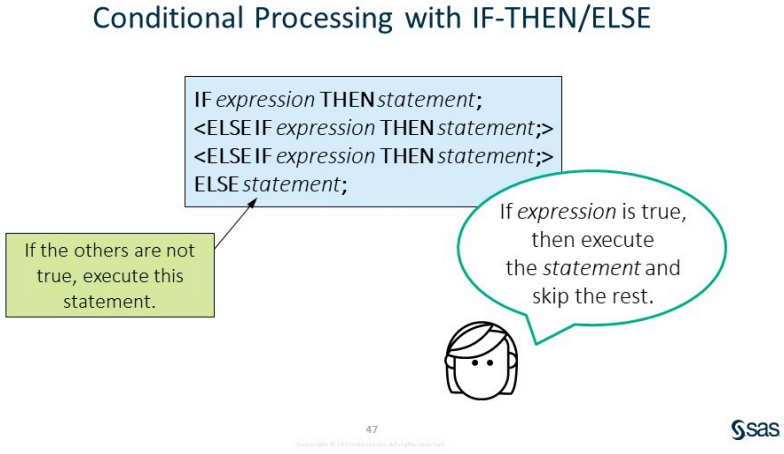


Let's look at this example code. Here I'm reading the **sashelp.cars** table. In the **cars2** table, I want to create a new column named **Cost_Group** and assign a number to the column based on the value of the manufacturer's suggested retail price, or MSRP.

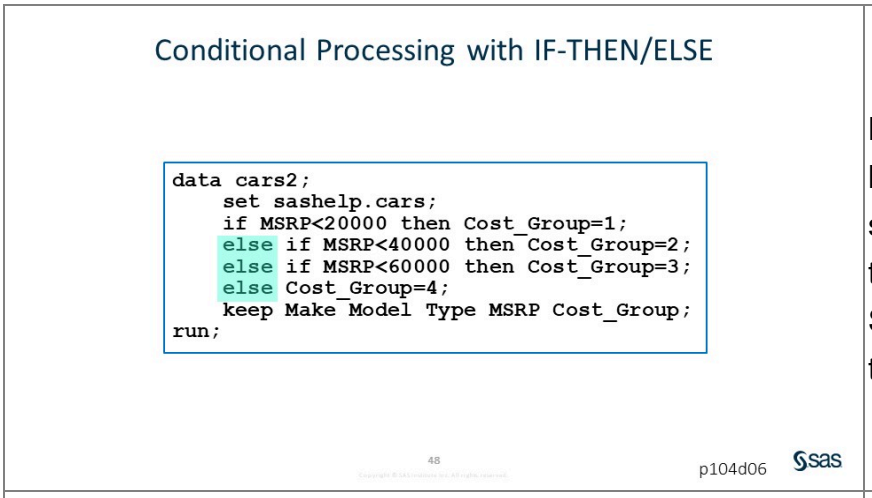
I need to use conditional logic to do this. After the keyword IF, we provide a condition: **MSRP** less than 30,000 or **MSRP** greater than or equal to 30,000. If the condition is true, then execute a single DATA step statement after the keyword THEN. The executable statements assign a value of 1 or 2 to the new column **Cost_Group**. The IF-THEN statements conclude with a semicolon. I've always liked this syntax because it is really intuitive.

Demo: Conditional Processing with IF-THEN

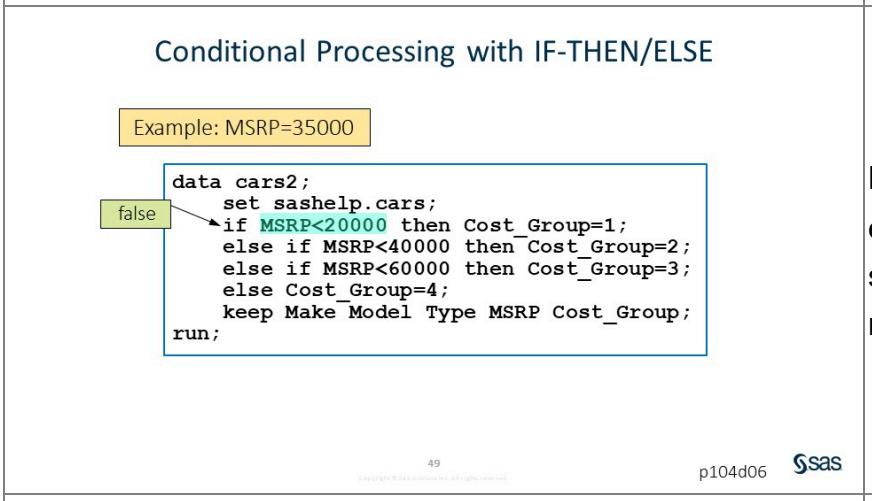
4_3 - Demo - Conditional Processing with IF-THEN.pdf
(https://clemson.instructure.com/courses/237270/files/23073937/download?wrap=1)
(https://clemson.instructure.com/courses/237270/files/23073937/download?download_frd=1)



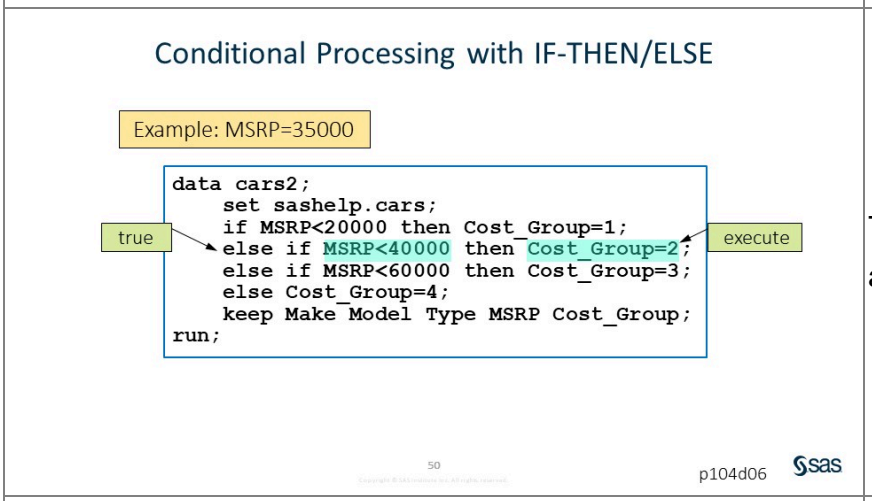
When you have multiple IF-THEN statements, SAS tests all conditions in sequence for every row of the data. The last true condition executes the statement that determines the value in the output table. Suppose you want to treat these conditions as a hierarchy so that when a true condition is found, SAS simply executes the statement following THEN and skips the subsequent IF statements. If you want to enforce this type of sequential testing, then the magic word you can add is ELSE.



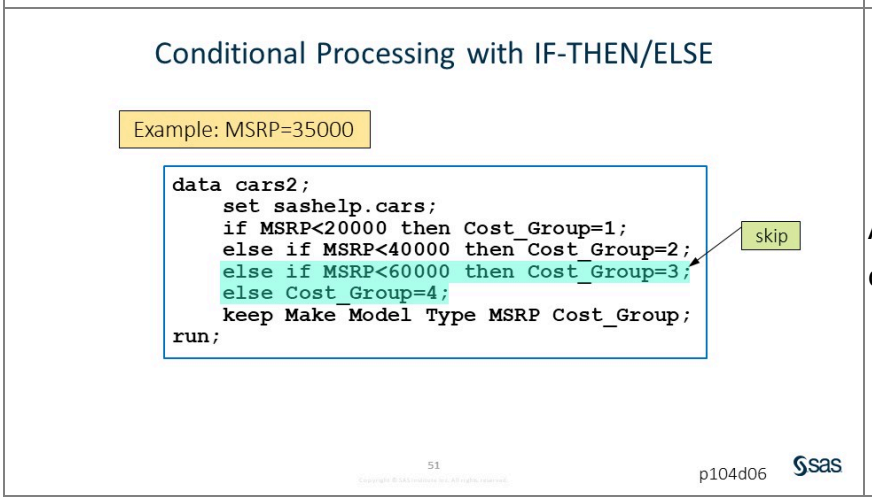
Looking at this code example, the keyword ELSE is not in the first statement, but it has been added in the three statements that follow. This tells SAS to test the conditions only until a true expression is found.



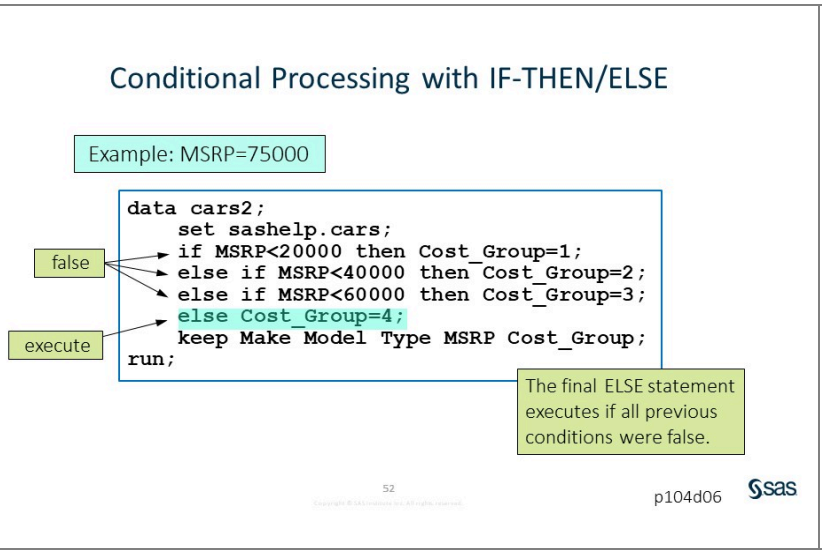
Let's look at an example where **MSRP** is equal to 35000. The first IF-THEN statement is false, so SAS moves to the next statement.



The second condition is true, so SAS assigns the value 2 to **Cost_Group**.



And then SAS skips the rest of the conditional processing statements.



For a row with **MSRP** equal to 75000, none of the stated **MSRP** conditions are true, so the last assignment statement is executed. Notice in this final ELSE statement that there is no condition, just an assignment statement. There is no reason to test that final condition because if the preceding conditions are all false, we know **Cost_Group** should be 4.

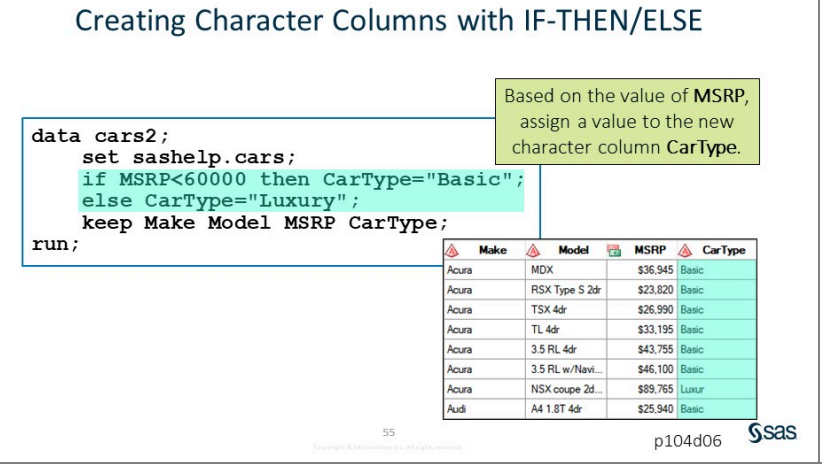
Activity 4.07

Open **p104a07.sas** from the **activities** folder and perform the following tasks:

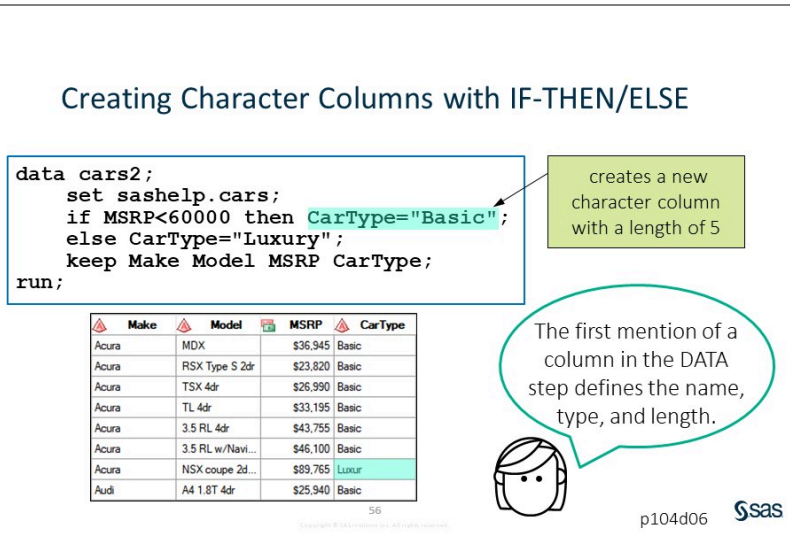
1. Add the **ELSE** keyword to test conditions sequentially until a true condition is met.
2. Change the final IF-THEN statement to an ELSE statement.
3. How many storms are in **PressureGroup** 1?

Click here for Solution.

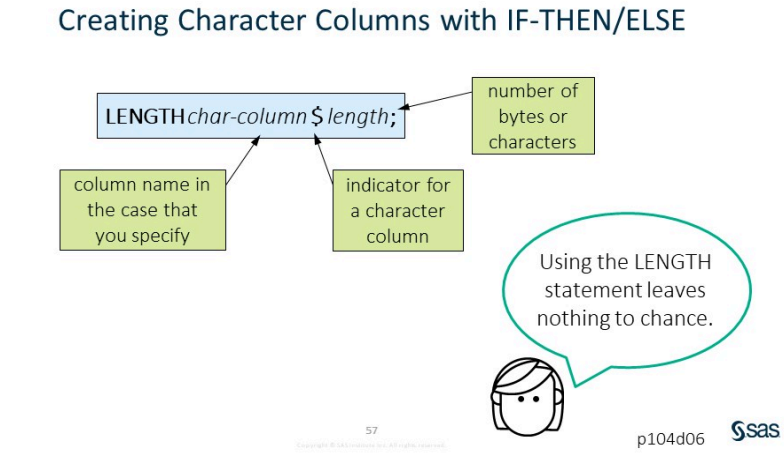
Creating Character Columns with If-then/Else



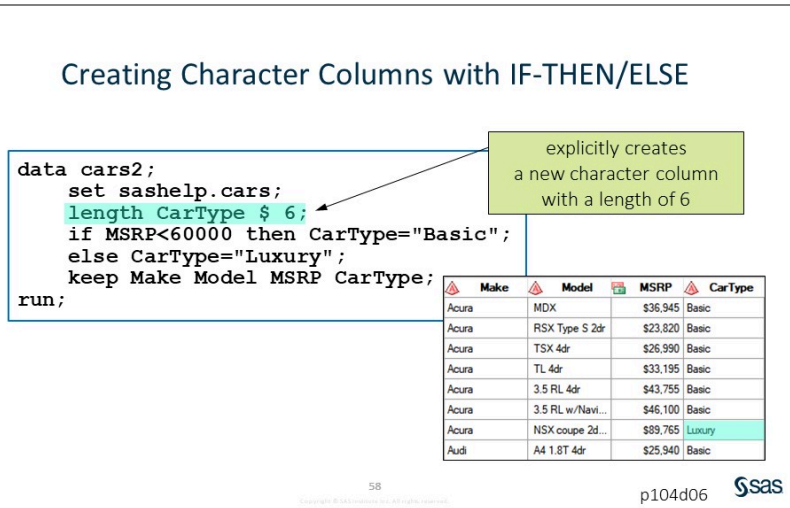
We can also use conditional processing to create character columns.



It's important to know that the first occurrence of a column in the DATA step defines the name, type, and length of the column. So, if you have an assignment statement that defines a character column and assigns the value *Basic*, the column is created with a length of 5, the number of characters in the word *Basic*. You can see from the output that *LUXURY* is truncated because it has six characters.



One way to avoid this problem is to explicitly define a character column in the DATA step with a LENGTH statement. The syntax for this statement is the keyword LENGTH followed by the name of the column, a dollar sign to indicate a character column, and the length you want to assign.



In this example code, the LENGTH statement defines the **CarType** column with a length of 6 before the conditional processing statements.

Activity 4.08

Open **p104a08.sas** from the **activities** folder and perform the following tasks:

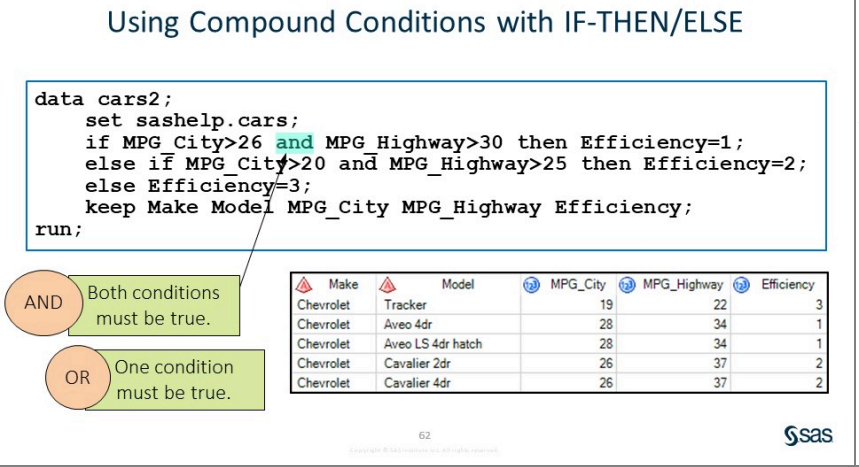
1. Run the program and examine the results. Why is **Ocean** truncated?
   What value is assigned when Basin='*na*'?
2. Modify the program to add a LENGTH statement to declare the name, type, and length of **Ocean** before the column is created.

   LENGTH *char-column* $ *length*;

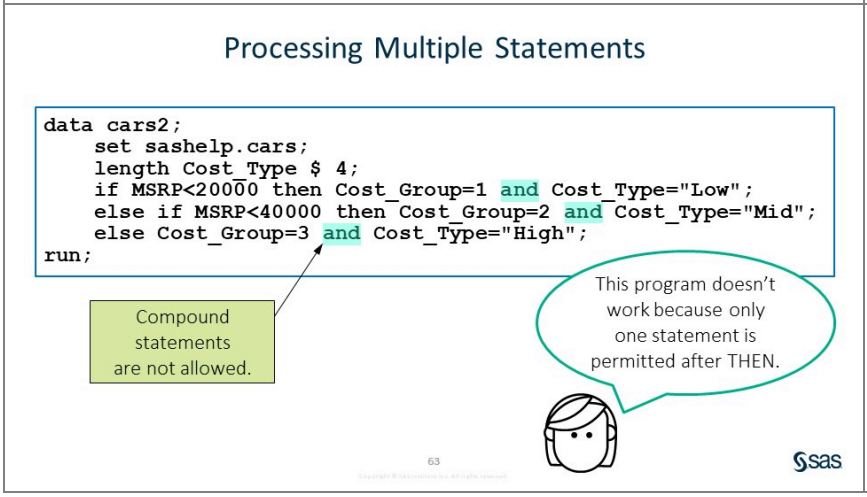
3. Add an assignment statement after the KEEP statement to convert **Basin** to uppercase. Run the program.
4. Move the LENGTH statement to the end of the DATA step. Run the program. Does it matter where the LENGTH statement is in the DATA step?

Click here for Solution.

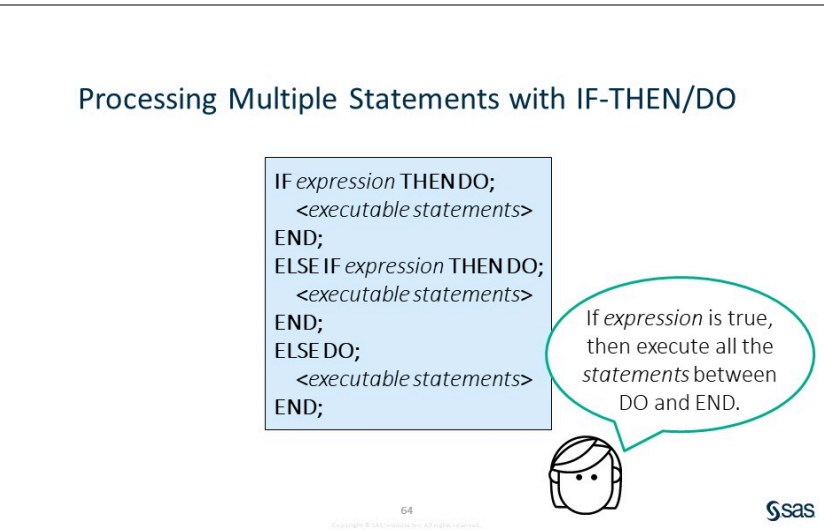
Using Compound Conditions with If-Then/Else



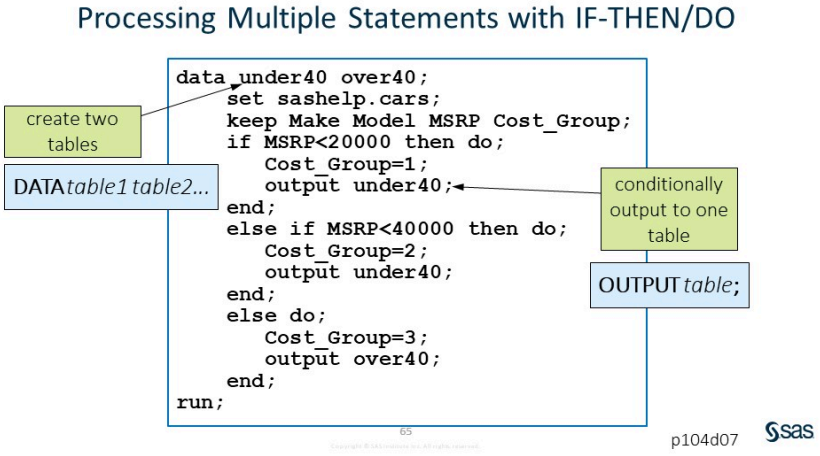
Let's look at another scenario. What if I want to evaluate compound conditions? That's perfectly fine! You can create compound conditions with AND or OR. If you use AND, both conditions must be true, and if you use OR, one condition must be true.



So if you can specify a compound condition to evaluate, can you do the same after the keyword THEN to execute multiple statements? If I attempt to use AND between two statements, the program fails with a syntax error because I am allowed only one executable statement following THEN.



Don't worry, SAS offers alternate syntax that you can use when you want to execute multiple statements for a given condition. We call this syntax IF-THEN/DO. After a condition, you type **THEN DO** and a semicolon. After that statement, you can list as many statements as you need to process, and then close the block with an END statement. This is repeated for each of the ELSE IF or ELSE DO blocks.



In this example, we use the DATA step to create not one, but two tables. In the DATA statement we can list more than one output table. In the first condition, if **MSRP** is less than 20000, we assign **Cost_Group** a value of 1, and then use the explicit OUTPUT statement to tell SAS which of the two tables to write that row to. Just remember that because these statements execute in sequence, we must first assign a value to **Cost_Group**, and then output the row to a particular table. The remaining conditions also include statements to assign a different value to **Cost_Group** and output to either the **under40** or **over40** tables.

Activity: 4.09

Open **p104a09.sas** from the **activities** folder. Run the program. Why does the program fail?

Click here for Solution.

Demo: Processing Multiple Statements with If-Then/DO

4_3 Demo - Processing Multiple Statements with If-THEN_DO.pdf
(https://clemson.instructure.com/courses/237270/files/23074517/download?wrap=1)
(https://clemson.instructure.com/courses/237270/files/23074517/download?download_frd=1)