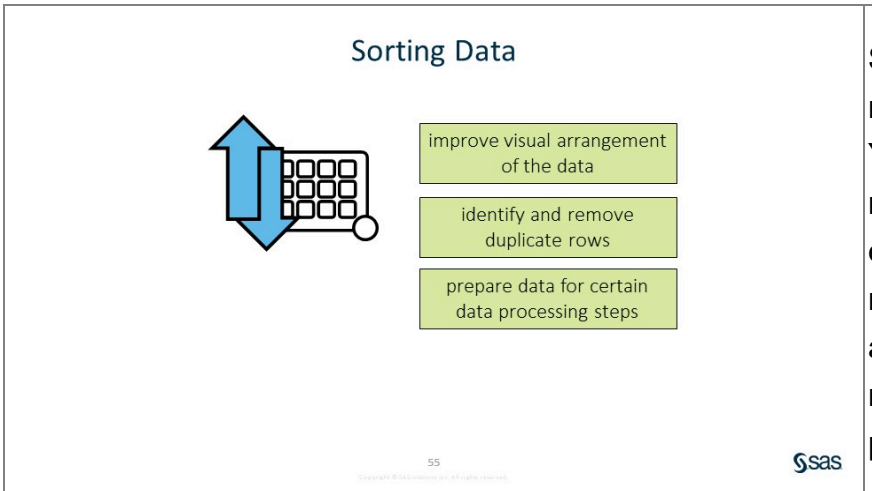
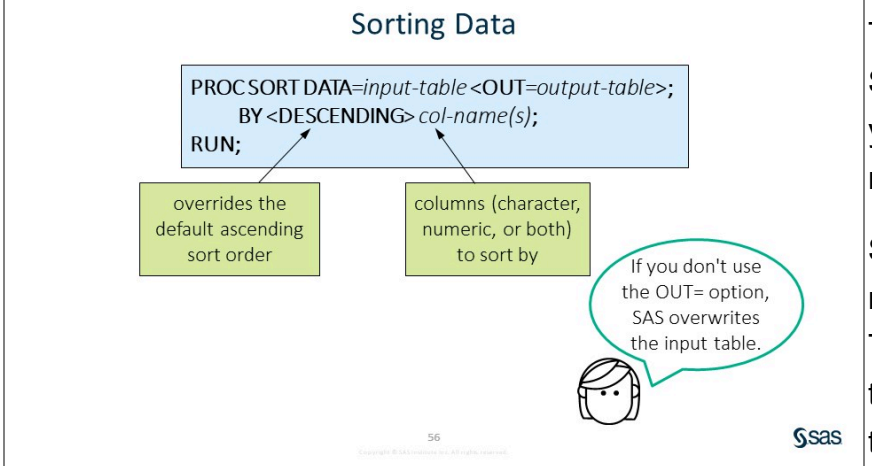
B3.4 - Sorting Data and Removing Duplicates



Sorting data can be a helpful or necessary step in exploring your data. You might want to sort on groups or measures so that you can visually examine the high or low values. You might use sorting as a way to identify and remove duplicate rows. Also, sorting might be required for certain data processing steps.



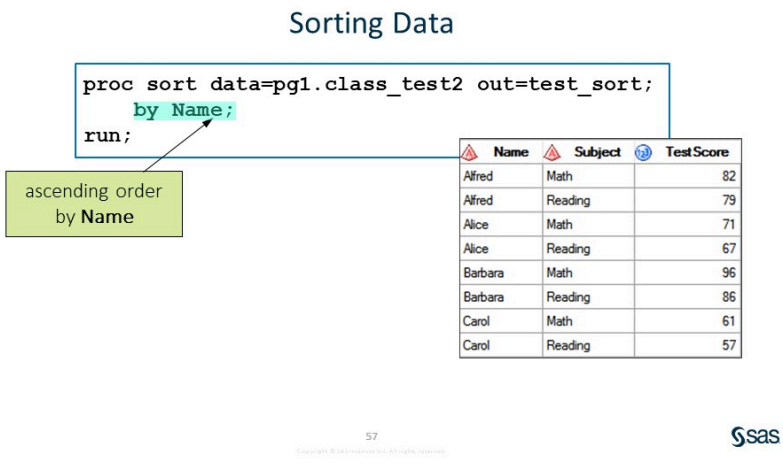
To sort the rows in a table, you use the SORT procedure. Using PROC SORT, you can sort on one or more character or numeric columns.

So how does it work? First, SAS rearranges the rows in the input table. Then SAS creates a table that contains the rearranged rows either by replacing the original table or by creating a new table. By default, SAS replaces the original SAS table unless you use the OUT= option to specify an output table. Keep in mind that PROC SORT does not generate printed output, so you have to open or print the sorted table if you want to look at it.

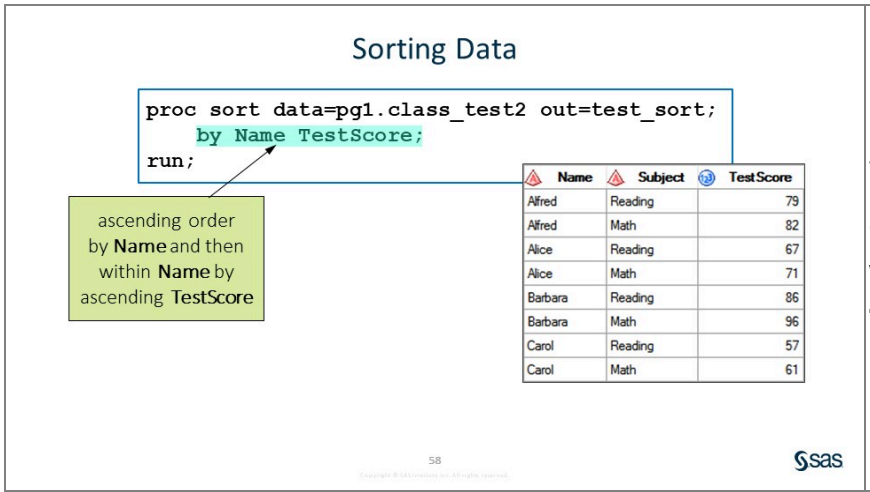
Let's check out the basic syntax. Similar to PROC PRINT and other procedures, you use the DATA= option to specify the

input table. Next, we often use the OUT= option in the PROC SORT statement so we don't permanently sort the input table. If you don't include the OUT= option, PROC SORT changes the sort order of the input table.

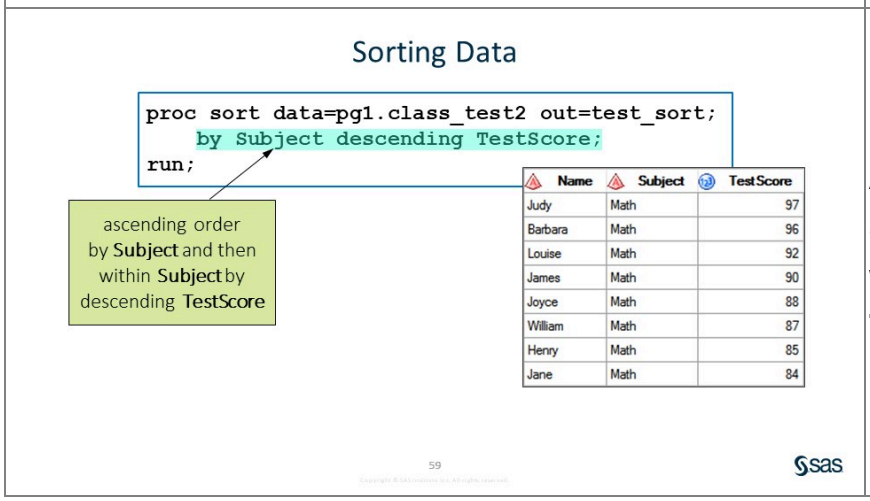
Every PROC SORT step must include a BY statement. The BY statement specifies one or more columns in the input table whose values are used to sort the rows. The BY statement also indicates whether you want to sort in ascending or descending order. By default, SAS sorts in ascending order, but if you want to sort by a column in descending order, you must specify the DESCENDING keyword immediately before each column that you want in descending order.



In this BY statement, we sorting the data by ascending values of **Name**.



This BY statement sorts first by ascending values of **Name**, and then within **Name** by ascending values of **TestScore**.



And finally this BY statement sorts by ascending values of **Subject**, and then within **Subject** by descending values of **TestScore**.

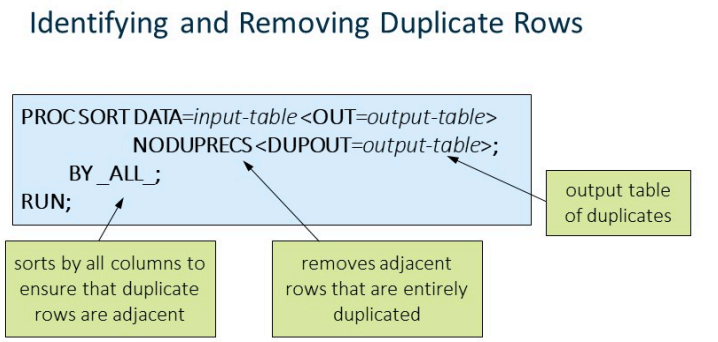
Activity 3.07

Open **p103a07.sas** from the **activities** folder and perform the following tasks:

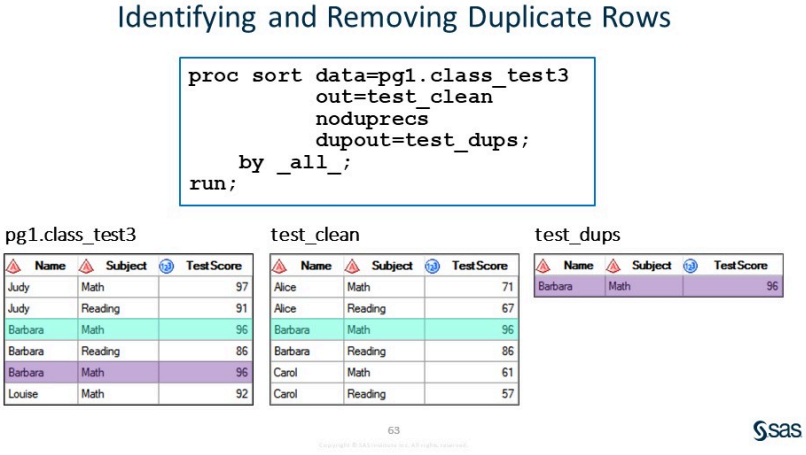
1. Modify the OUT= option in the PROC SORT statement to create a temporary table named **storm_sort**.
2. Complete the WHERE and BY statements to answer the following question: Which storm in the North Atlantic basin (*NA* or *na*) had the strongest **MaxWindMPH**?

Click here for Solution.

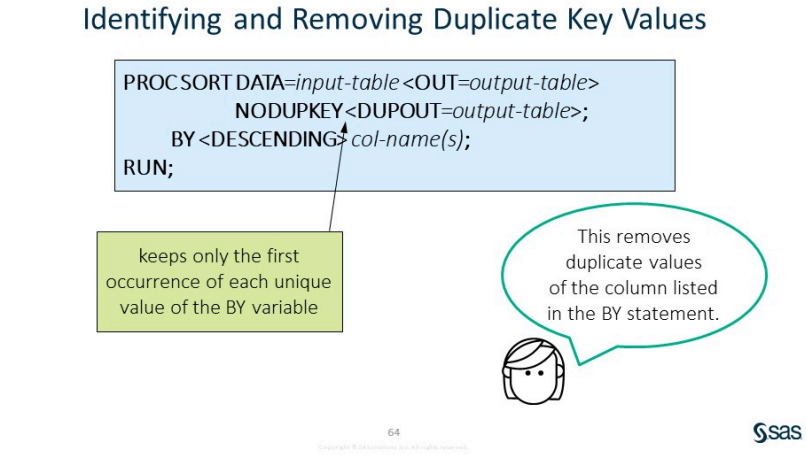
Identifying and Removing Duplicate Rows



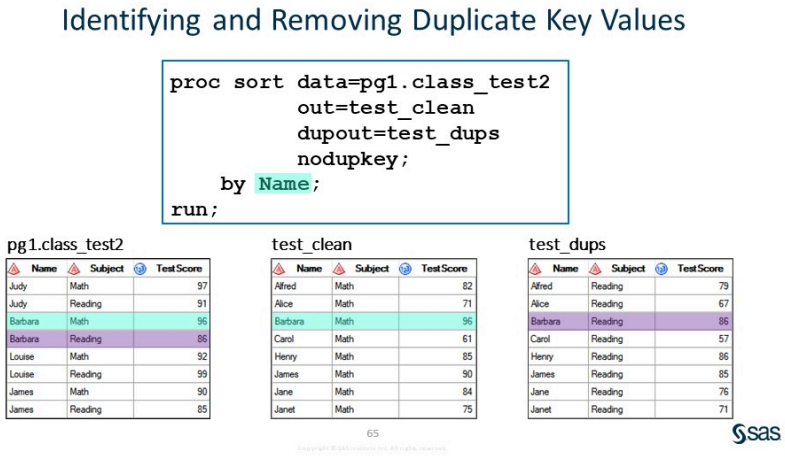
The SORT procedure provides an easy way to identify and remove duplicates in your data. In the PROC SORT statement, you can use the NODUPRECS option to remove adjacent rows that are entirely duplicated. In other words, remove rows that are next to each other in the data where the values for every column match. You can use the keyword _ALL_ in the BY statement instead of a column name. This sorts the data by all columns so that entirely duplicated rows are adjacent, and then the NODUPRECS option can do its job. The table listed in the OUT= option will have the duplicates removed. It's also helpful for validation to specify the DUPOUT= option and generate a table of the duplicate rows that were removed by NODUPRECS.



Here's an example of using the NODUPRECS option. You can see that there are two rows for Barbara that contain identical information.



Another option that's available is NODUPKEY. This option is helpful when you want to identify whether you have duplicated values for select columns. When you add the NODUPKEY option to the PROC SORT statement, only the first occurrence for each unique value of the column listed in the BY statement is kept

	in the output table. Let's see how this works in an example.
	In this example, we keep only the first row for each unique value of **Name** in **class_test2**. Because the math test was listed first for each student in the data, the output table **test_clean** has all the math test information, and the **test_dups** table has the reading test information.

Demo: Identifying and Removing Duplicate Values

3_4 - Demo - Identifying and Removing Duplicate Values.pdf
(https://clemson.instructure.com/courses/237270/files/23074695/download?wrap=1)
(https://clemson.instructure.com/courses/237270/files/23074695/download?download_frd=1)