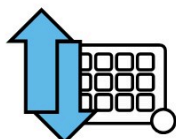


# B3.4 - Sorting Data and Removing Duplicates

## Sorting Data



improve visual arrangement of the data

identify and remove duplicate rows

prepare data for certain data processing steps

55

Copyright © SAS Institute Inc. All rights reserved.

sas

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.

## Sorting Data

```
PROC SORT DATA=input-table <OUT=output-table>;
  BY <DESCENDING> col-name(s);
RUN;
```

overrides the default ascending sort order

columns (character, numeric, or both) to sort by

If you don't use the OUT= option, SAS overwrites the input table.



56

Copyright © SAS Institute Inc. All rights reserved.

sas

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.

### Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;  
  by Name;  
run;
```

ascending order  
by **Name**

Name	Subject	Test Score
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67
Barbara	Math	96
Barbara	Reading	86
Carol	Math	61
Carol	Reading	57

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

## Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;
  by Name TestScore;
run;
```

ascending order  
by **Name** and then  
within **Name** by  
ascending **TestScore**

Name	Subject	TestScore
Alfred	Reading	79
Alfred	Math	82
Alice	Reading	67
Alice	Math	71
Barbara	Reading	86
Barbara	Math	96
Carol	Reading	57
Carol	Math	61

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

58

Copyright © SAS Institute Inc. All rights reserved.



## Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;
  by Subject descending TestScore;
run;
```

ascending order  
by **Subject** and then  
within **Subject** by  
descending **TestScore**

Name	Subject	TestScore
Judy	Math	97
Barbara	Math	96
Louise	Math	92
James	Math	90
Joyce	Math	88
William	Math	87
Henry	Math	85
Jane	Math	84

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

59

Copyright © SAS Institute Inc. All rights reserved.



## Activity 3.07

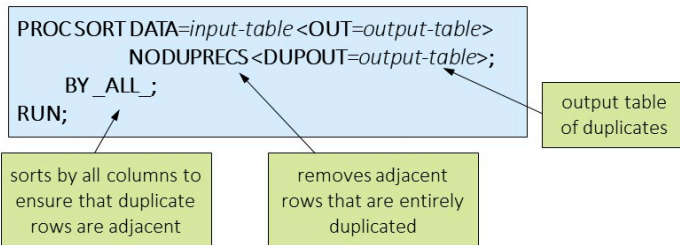
Open **p103a07.sas** from the **activities** folder and perform the following tasks:

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

## Identifying and Removing Duplicate Rows



62

Copyright 2014 SAS Institute Inc. All rights reserved.

sas

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.

## Identifying and Removing Duplicate Rows

```

proc sort data=pg1.class_test3
  out=test_clean
  noduprecs
  dupout=test_dups;
  by _all_;
run;

```

pg1.class_test3				test_clean				test_dups			
Name	Subject	Test Score		Name	Subject	Test Score		Name	Subject	Test Score	
Judy	Math	97		Alice	Math	71		Barbara	Math	96	
Judy	Reading	91		Alice	Reading	67					
Barbara	Math	96		Barbara	Math	96					
Barbara	Reading	86		Barbara	Reading	86					
Barbara	Math	96		Carol	Math	61					
Louise	Math	92		Carol	Reading	57					

63

Copyright 2014 SAS Institute Inc. All rights reserved.

sas

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

## Identifying and Removing Duplicate Key Values

```

PROC SORT DATA=input-table <OUT=output-table>
  NODUPKEY <DUPOUT=output-table>;
  BY <DESCENDING> col-name(s);
RUN;

```

keeps only the first occurrence of each unique value of the BY variable

This removes duplicate values of the column listed in the BY statement.



64

Copyright 2014 SAS Institute Inc. All rights reserved.

sas

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.

### Identifying and Removing Duplicate Key Values

```
proc sort data=pg1.class_test2
  out=test_clean
  dupout=test_dups
  nodupkey;
  by Name;
run;
```

pg1.class\_test2

Name	Subject	Test Score
Judy	Math	97
Judy	Reading	91
Barbara	Math	96
Barbara	Reading	86
Louise	Math	92
Louise	Reading	99
James	Math	90
James	Reading	85

test\_clean

Name	Subject	Test Score
Afred	Math	82
Alice	Math	71
Barbara	Math	96
Carol	Math	61
Henry	Math	85
James	Math	90
Jane	Math	84
Janet	Math	75

test\_dups

Name	Subject	Test Score
Afred	Reading	79
Alice	Reading	67
Barbara	Reading	86
Carol	Reading	57
Henry	Reading	86
James	Reading	85
Jane	Reading	76
Janet	Reading	71

65

Copyright © SAS Institute Inc. All rights reserved.

sas

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://clemons.instructure.com/courses/237270/files/23074695/download?wrap=1> ↓

[https://clemons.instructure.com/courses/237270/files/23074695/download?download\\_frd=1](https://clemons.instructure.com/courses/237270/files/23074695/download?download_frd=1) ⓘ

