Paper 048-2010

# The Mystery of the PROC SORT Options NODUPRECS and NODUPKEY Revealed

Britta KelseyBassett, Schwan's Home Service, Inc., Marshall MN

## Abstract

The NODUPRECS (or NODUP) and NODUPKEY options can be useful with the SORT procedure but they can be dangerous if you do not understand them completely. They work similarly in that they both can eliminate unwanted observations, but NODUPRECS compares all the variables in your data set while NODUPKEY compares just the BY variables. Also, you must be aware of how your data set is currently sorted to eliminate the observations that you want because these options compare adjacent observations. In this paper, I will describe this in greater detail and show examples of how to use the NODUPRECS and NODUPKEY options. All examples shown were done in the SAS® system for PCs, version 9.1.3. The intended audience for this paper is beginner level SAS programmers.

## Introduction

There seems to be confusion among SAS users about the NODUPRECS (or NODUP) and NODUPKEY options. These can be very useful but dangerous if you are not clear on how they work. I will bring some clarification to these PROC SORT options and will show some examples of how they work so you can put them to good use. Throughout the rest of this paper I will be using the alias NODUP for the NODUPRECS option.

## Defining NODUP and NODUPKEY Options

The NODUP option checks for and eliminates duplicate observations. If you specify this option, PROC SORT compares all variable values for each observation to those for the previous observation that was written to the output data set. If an exact match is found, the observation is not written to the output data set.

The NODUPKEY option checks for and eliminates observations with duplicate BY variable values. If you specify this option, PROC SORT compares all BY variable values for each observation to those for the previous observation written to the output data set. If an exact match using the BY variable values is found, the observation is not written to the output data set.

Notice that with the NODUPKEY option, PROC SORT is comparing all *BY* variable values while the NODUP option compares *all* the variables in the data set that is being sorted. An easy way to remember the difference between these options is to keep in mind the word "key" in NODUPKEY. It evaluates the "key" or BY variable values that you specify. One thing to beware of with both options is that they both compare the previous observation written to the output data set. So, if the observations that you want eliminated are not adjacent in the data set after the sort, they will not be eliminated.

## NODUP Examples

Below is the code for the data set, called CUST_INFO that I will be using throughout this paper. This data set is similar to one used in food industry that contains customer information. It contains customer numbers, the catalog group they are assigned to, their region of the country, and their order preference which is a numeric code. Notice that customer 01 (highlighted in red) has two observations that are exactly the same and customer 03 (highlighted in blue) has two observations with only the variable REGION being different.

```
data cust_info;
  input customer 1-2 group $ 4-5 region $ 6-7 ordpref 9-10;
  datalines;
01 A GL 0
02 A NE 1
03 B SE 1
04 B GL 2
05 C MT 1
06 C MT 3
07 C NE 2
01 A GL 0
03 B NE 1
  ;
  run;
```

Example 1:

In this example, the SORT procedure is used with the NODUP option. By using a PROC SORT, I want to order the data by the variable CUSTOMER and eliminate any observations that have the exact same information for all variables.

```
proc sort data=cust_info nodup out=ex1;
      by customer;
run;
```

The output data set EX1 looks like:

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 03 | B | SE | 1 |
| 03 | B | NE | 1 |
| 04 | B | GL | 2 |
| 05 | C | MT | 1 |
| 06 | C | MT | 3 |
| 07 | C | NE | 2 |

Notice that customer 03 (highlighted in blue) still has two observations because all the variable values are not the same for these two observations.

Example 2:

In this example, I want to order the data by GROUP and eliminate any observations that have the exact same information for all variables. Using the PROC SORT statement, I first use the BY variable GROUP to see if I get the result I want.

```
proc sort data=cust_info nodup out=ex2_1;
      by group;
run;
```

The output data set EX2_1 looks like:

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 01 | A | GL | 0 |
| 03 | B | SE | 1 |
| 04 | B | GL | 2 |
| 03 | B | NE | 1 |
| 05 | C | MT | 1 |
| 06 | C | MT | 3 |
| 07 | C | NE | 2 |

This sort did not do the elimination that I wanted. When the procedure sorted by GROUP, the second observation for customer 01 just moved up under customer 02. The two observations for customer 01 are not adjacent; therefore, the second one did not get eliminated. To get the results that I want, I will have to sort by the GROUP and CUSTOMER variables.

```
proc sort data=cust_info nodup out=ex2_2;
      by group customer;
run;
```

The output data set EX2_2 looks like:

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 03 | B | SE | 1 |
| 03 | B | NE | 1 |
| 04 | B | GL | 2 |
| 05 | C | MT | 1 |

| | | | |
|---|---|---|---|
| 06 | C | MT | 3 |
| 07 | C | NE | 2 |

Now the second observation for customer 01 is eliminated and the data set is ordered by group which is the desired result.

## NODUPKEY Examples

For these examples, I will be using the data set CUST_INFO from above.

Example 3:

In this example the SORT procedure is used with the NODUPKEY option. This is similar to example 1 above but I am going to use the NODUPKEY option instead of the NODUP option and compare the difference in results.

```
proc sort data=cust_info nodupkey out=ex3;
      by customer;
run;
```

The output data set EX3 looks like:

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 03 | B | SE | 1 |
| 04 | B | GL | 2 |
| 05 | C | MT | 1 |
| 06 | C | MT | 3 |
| 07 | C | NE | 2 |

Notice that the PROC SORT eliminated one of the observations for customer 01 and one for customer 03. Because I sorted by the CUSTOMER variable and used the NODUPKEY option, the SORT procedure is checking for adjacent observations with only the same customer number and eliminating any duplicates after the first occurrence of that number. In contrast, in the output data set for example 1, customer 03 still had 2 observations.

Here it is important to note that the first observation for customer 03 has a SE for the variable REGION and the second observation for customer 03 has a NE for that variable. The NODUPKEY option is only looking at the BY variable of CUSTOMER so it ignores the difference in all other variables and it eliminates the second observation with a REGION of NE. This is where this option can be dangerous because you may eliminate observations that you actually wanted to keep since they may have different values for a non-BY variable.

Example 4:

In this example I will sort by the variable GROUP as I did in example 2 above but instead of using the NODUP option, will use the NODUPKEY option to see the difference in results.

```
proc sort data=cust_info nodupkey out=ex4;
      by group;
run;
```

The output data set EX4 looks like:

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 03 | B | SE | 1 |
| 05 | C | MT | 1 |

Since I sorted by the variable GROUP and used the NODUPKEY option in this example, PROC SORT only kept the first observation it encountered for each group and eliminated the duplicates after that. By making the one change in options from NODUP to NODUPKEY, the output data set EX4 (with only one observation per group) looks quite different from the output data set EX2_1 generated in example 2. Again, this is why you should be cautious when using these options. The difference in output data sets is obvious with this small input data set but it may not be as obvious when using a very large data set.

Example 5:

In this example I will sort by the variables GROUP and REGION with the NODUPKEY option to see what will happen to my data set.

```
proc sort data=cust_info nodupkey out=ex5;
      by group region;
run;
```

To help better understand what is being eliminated, I will once again show the data set CUST_INFO and compare it to the output data set EX5:

Data set CUST_INFO -

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 03 | B | SE | 1 |
| 04 | B | GL | 2 |
| 05 | C | MT | 1 |
| 06 | C | MT | 3 |
| 07 | C | NE | 2 |
| 01 | A | GL | 0 |
| 03 | B | NE | 1 |

Data set EX5 –

| CUSTOMER | GROUP | REGION | ORDPREF |
|---|---|---|---|
| 01 | A | GL | 0 |
| 02 | A | NE | 1 |
| 04 | B | GL | 2 |
| 03 | B | NE | 1 |
| 03 | B | SE | 1 |
| 05 | C | MT | 1 |
| 07 | C | NE | 2 |

Here the second observation for customer 01 is eliminated because it has the same BY variable values as the first observation for customer 01 (highlighted in red). Also, customer 06 is eliminated because it has the same BY variable values as customer 05 and they were adjacent (highlighted in green). Customer 03 (highlighted in blue) still has two observations because each observation has a different region.

## Conclusion

The NODUP option in the SORT procedure eliminates observations that are exactly the same across *all* variables. The NODUPKEY option eliminates observations that are exactly the same across the *BY* or "key" variables. Keep in mind that both of these options compare adjacent observations in the output data set. Now that you know how the NODUP and NODUPKEY options work, you can use them in confidence to get the data set you want!

## References

SAS Institute Inc. (1990), *SAS Procedures Guide, Version 6, Third Edition,* Cary, NC: SAS Institute Inc.

## Contact Information

Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Britta KelseyBassett
Schwan's Home Service, Inc.
115 West College Drive
Marshall, MN 56258
E-mail: britta.kelseybassett@schwans.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.