Paper 041-2012

Intelligent PROC SORT NODUPKEY

Andrea Zimmerman, Capital One, Richmond, VA

ABSTRACT

Have you ever had a data set that you were updating records in and needed to eliminate the old records and replace them with new records? The SORT procedure with the NODUPKEY option eliminates rows that duplicate your key fields, choosing which record to keep based on its own logic, which might not be the one you want. This paper shows how to combine a PROC SORT and a DATA step to get the sorted data set with the exact records that you want.

INTRODUCTION

When updating a SAS® data set, you may end up with duplicate rows of data. If you record the date and time when you add the row, it will be obvious which row you'd want to keep when deleting the redundant records. The NODUPKEY option in PROC SORT allows you to indicate that you would like records that repeat the key to be reduced down to one single observation. But SAS will randomly select one of the rows to keep. By following a PROC SORT with a DATA step, you can achieve a sorted data set, eliminate the duplicate records, and specifically keep the records you want.

DATA SETUP

First, just a few preliminaries. Let's start with the SASHELP.CLASS data set. We'll assume that is our base data set, but all of the boys have had a growth spurt and are now 6 inches taller and weigh 5 pounds more. We'll create an update data set with the rows of data we want to update in our data set.

```
data class;
   set sashelp.class;
   format update_date mmddyy8.;
   update_date=today()-1;/*set to yesterday's date*/
run;

data updates;
   set sashelp.class;
   format update_date mmddyy8.;
   if sex='M';
   /*boys had growth spurts*/
   height=height+6;
   weight=weight+5;

   update_date=today();/*set to today's date*/
run;

proc append
   base=class
   data=updates;
run;
```

PROC SORT

First we run a PROC SORT **without** the NODUPKEY option. The BY statement should have the fields you want to sort by, followed by the field that tells you which row you'd want to keep, such as an UPDATE_DT var. Leave out any fields that you would want to update (such as age, height, and weight)

```
proc sort data=class;
   by name sex update_date;
run;
```

Name	Sex	Age	Height	Weigh t	update_date
Alfred	M	14	69	112.5	7/7/2011
Alfred	M	14	75	117.5	7/8/2011
Alice	F	13	56.5	84	7/7/2011
Barbara	F	13	65.3	98	7/7/2011
Carol	F	14	62.8	102.5	7/7/2011
Henry	M	14	63.5	102.5	7/7/2011
Henry	M	14	69.5	107.5	7/8/2011
James	M	12	57.3	83	7/7/2011
James	M	12	63.3	88	7/8/2011
Jane	F	12	59.8	84.5	7/7/2011
Janet	F	15	62.5	112.5	7/7/2011
Jeffrey	M	13	62.5	84	7/7/2011
Jeffrey	M	13	68.5	89	7/8/2011
John	M	12	59	99.5	7/7/2011
John	M	12	65	104.5	7/8/2011
Joyce	F	11	51.3	50.5	7/7/2011
Judy	F	14	64.3	90	7/7/2011
Louise	F	12	56.3	77	7/7/2011
Mary	F	15	66.5	112	7/7/2011
Philip	M	16	72	150	7/7/2011
Philip	M	16	78	155	7/8/2011
Robert	M	12	64.8	128	7/7/2011
Robert	M	12	70.8	133	7/8/2011
Ronald	M	15	67	133	7/7/2011
Ronald	M	15	73	138	7/8/2011
Thomas	M	11	57.5	85	7/7/2011
Thomas	M	11	63.5	90	7/8/2011
William	M	15	66.5	112	7/7/2011
William	M	15	72.5	117	7/8/2011

Fig. 1 Results after the proc sort. Note how when there is a duplicate, we want the second row.

DATA STEP

Next, create a new data set by SETting the previously sorted data set. Use the BY statement with the exact same variable list as the PROC SORT. SAS will create FIRST. and LAST. variables. KEEP either the first or last row depending on which you want and whether you sorted ascending or descending.

```
data class outdated;
   set class;
   by name sex update_date;
   if last.sex then output class;
   else output outdated;
run;
```

It is not necessary to write the duplicates to a file such as OUTDATED. It makes for a nice way to check yourself, and if you discover an error, you have not lost the original data. It is also helpful in debugging or if an audit trail for the data is required.

CLASS DATA SET					
Name	**Sex**	**Age**	**Height**	**Weight**	**update_date**
Alfred	M	14	75	117.5	7/8/2011
Alice	F	13	56.5	84	7/7/2011
Barbara	F	13	65.3	98	7/7/2011
Carol	F	14	62.8	102.5	7/7/2011
Henry	M	14	69.5	107.5	7/8/2011
James	M	12	63.3	88	7/8/2011
Jane	F	12	59.8	84.5	7/7/2011
Janet	F	15	62.5	112.5	7/7/2011
Jeffrey	M	13	68.5	89	7/8/2011
John	M	12	65	104.5	7/8/2011
Joyce	F	11	51.3	50.5	7/7/2011
Judy	F	14	64.3	90	7/7/2011
Louise	F	12	56.3	77	7/7/2011
Mary	F	15	66.5	112	7/7/2011
Philip	M	16	78	155	7/8/2011
Robert	M	12	70.8	133	7/8/2011
Ronald	M	15	73	138	7/8/2011
Thomas	M	11	63.5	90	7/8/2011
William	M	15	72.5	117	7/8/2011

Fig. 2 Note that we have one row per student, and we have the most recent row for each.

OUTDATED DATA SET					
Name	**Sex**	**Age**	**Height**	**Weight**	**update_date**
Alfred	M	14	69	112.5	7/7/2011
Henry	M	14	63.5	102.5	7/7/2011
James	M	12	57.3	83	7/7/2011
Jeffrey	M	13	62.5	84	7/7/2011
John	M	12	59	99.5	7/7/2011
Philip	M	16	72	150	7/7/2011
Robert	M	12	64.8	128	7/7/2011
Ronald	M	15	67	133	7/7/2011
Thomas	M	11	57.5	85	7/7/2011
William	M	15	66.5	112	7/7/2011

Fig. 3 Note that the rows we wanted to remove ended up in this data set.

CONCLUSIONS

By joining the power of the DATA step and the use of LAST., we can control which duplicate rows are kept when we use PROC SORT. The informative variable can be any field where you either want the row with minimum or the maximum value.

ACKNOWLEDGMENTS

I'd like to thank the VASUG members and officers for their support as well as my team and manager for allowing me the opportunity to share my knowledge.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Andrea Zimmerman
Capital One, Inc.
15000 Capital One Drive
Richmond, VA 23228
Work Phone: 804-284-7681
Email: andrea.zimmerman@capitalone.com
Twitter: ANWZimmerman

APPENDIX

COMPLETE CODE

For simplicity with trying on your own.

```
data class;
   set sashelp.class;
   format update_date mmddyy8.;
   update_date=today()-1;/*set to yesterday's date*/
run;

data updates;
   set sashelp.class;
   format update_date mmddyy8.;
   if sex='M';
   /*boys had growth spurts*/
   height=height+6;
   weight=weight+5;

   update_date=today();/*set to today's date*/
run;

proc append
   base=class
   data=updates;
run;

proc sort data=class;
   by name sex update_date;
run;

data class outdated;
   set class;
   by name sex update_date;
   if last.sex then output class;
   else output outdated;
run;
```