One problem –> Multiple solutions; various ways of removing duplicates from dataset using SAS®

Jaya Dhillon, Louisiana State University

ABSTRACT

In real world, analysts seldom come across data which is in ready to use format. Data cleaning is a critical aspect and one of the major problems faced is duplicate records within a dataset. For instance, we may want to have all unique Employee ID's in our dataset or we may want unique transactions from a list of customer transactions. This paper will cover various options available for treating duplicates records. Usage of options such as NODUPKEY in combination with DUPOUT, how NOUNIQUEKEY, NODUPRECS and NODUPKEY are different from each other, will be some of the points discussed here.

INTRODUCTION

Analysts spend the majority of time in the data exploration and preparation phase of a data mining project. A critical aspect of this phase is the detection and removal of duplicate records. This paper discusses how to resolve the problem of duplicate records. There are various ways in SAS to tackle the problem of duplication. A few options are mentioned in this paper and include NODUPRECS (alias NODUP), NODUPKEY. NOUNIQUEKEY is new to SAS version 9.3 and will be discussed as well.

DATASET

For the purpose of explanation, I have created a dataset "Emp_Details" containing details of employees working for different departments for a company XYZ. This dataset has 3 columns – Emp_ID, Emp_Name and Dept. Table 1 contains the code for creating the dataset; whereas, Table 2 displays the dataset.

Table 1.

```
data Emp_Details;
Input Emp_ID 6. Emp_Name $ Dept $;
datalines;
110011 Jack Sales
110012 Jonny Finance
110016 Jill Sales
110025 Harry Admin
110018 Harry Admin
110030 Mary Finance
110031 Potter Finance
110030 Mary Finance
110041 Tom Admin
;
run;
```

In Table 2, observation number 6 and 8 are exact duplicates, something which is definitely a case for deletion. However, observation number 4 and 5 have the same Employee Name (Harry) and Dept (Admin), but different employee IDs. This could be a possibility in real life where two people with same name work for the same department.

Table 2.

Obs	Emp_ID	Emp_Name	Dept
1	110011	Jack	Sales
2	110012	Jonny	Finance
3	110016	Jill	Sales
4	110025	Harry	Admin
5	110018	Harry	Admin
6	110030	Mary	Finance
7	110031	Potter	Finance
8	110030	Mary	Finance
9	110041	Tom	Admin

VARIOUS WAYS OF HANDLING DUPLICATE RECORDS

NODUPRECS

One of the most commonly used option is NODUPRECS. It deletes all those records which have the same values across all the variables. Example1 shows the code used to delete duplicate record using NODUPRECS and partial log displays "1 duplicate observations were deleted".

Example 1

```
Proc Sort data=Emp_Details noduprecs;
by Emp_ID;
run;
```

Partial SAS Log

```
NOTE: There were 9 observations read from the data set WORK.EMP_DETAILS.
NOTE: 1 duplicate observations were deleted.
NOTE: The data set WORK.EMP_DETAILS has 8 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Output Dataset

Obs	Emp_ID	Emp_Name	Dept
1	110011	Jack	Sales
2	110012	Jonny	Finance
3	110016	Jill	Sales
4	110018	Harry	Admin
5	110025	Harry	Admin
6	110030	Mary	Finance
7	110031	Potter	Finance
8	110041	Tom	Admin

Let's consider another example using the same dataset, after changing the BY variable. In example 2, the BY variable has been changed to "Dept" and we see in the SAS log that "0 duplicate observations were deleted". The reason is that SAS sorts the data on BY variable and compares one observation with the previous observation, if it encounters an exact match (same values across all variables) the record is deleted. So in example 2, after sorting by Dept., SAS couldn't find observation number 6 and 8 one after the other. Therefore, it is critical to use the correct BY variable to get the desired result.

Example 2

```
Proc Sort data=Emp_Details noduprecs;
by Dept;
run;
```

Partial SAS Log

```
NOTE: There were 9 observations read from the data set
WORK.EMP_DETAILS.
NOTE: 0 duplicate observations were deleted.
NOTE: The data set WORK.EMP_DETAILS has 9 observations and 3
variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

NODUPKEY

NODUPKEY is another option available in SAS to handle duplicate records. SAS looks for exact matches for BY variables specified. The difference between NODUP and NODUPKEY is that NODUP looks for exact matches for all the variables, whereas NODUPKEY looks for exact matches for BY variables specified in the code. Example 3 contains the code for removing duplicates based on three BY variables: EMP_ID, Emp_Name and Dept. The result is similar to

the one in example 1. However we would get different results by not specifying all the variables in BY variable.

Example 3

```
Proc Sort data=Emp_Details nodupkey dupout=dup_rec;
by Emp_ID Emp_Name Dept;
run;
```

Partial SAS Log

```
NOTE: There were 9 observations read from the data set WORK.EMP_DETAILS.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.DUP_REC has 1 observations and 3 variables.
NOTE: The data set WORK.EMP_DETAILS has 8 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
       cpu time            0.01 seconds
```

Output Dataset

Obs	Emp_ID	Emp_Name	Dept
1	110011	Jack	Sales
2	110012	Jonny	Finance
3	110016	Jill	Sales
4	110018	Harry	Admin
5	110025	Harry	Admin
6	110030	Mary	Finance
7	110031	Potter	Finance
8	110041	Tom	Admin

NOUNIQUEKEY

NOUNIQUEKEY is a new option added in SAS version 9.3. This option eliminates unique records based on BY variable. Duplicate records deleted from the original dataset can be saved in another dataset by using the option OUT. If option OUT is not used, the original dataset is overwritten and contains only the duplicate cases. UNIQUEOUT option can be used to store all the unique cases from the original dataset. Example 4 contains the code for eliminating unique records from duplicates by using NOUNIQUEKEY and UNIQUEOUT option.

Example 4

```
Proc Sort data=Emp_Details out=dup_rec nouniquekey uniqueout=unique_rec;
by Emp_ID Emp_Name Dept ;
run;
```

Partial SAS Log

```
NOTE: There were 9 observations read from the data set WORK.EMP_DETAILS.
NOTE: 7 observations with unique key values were deleted.
NOTE: The data set WORK.DUP_REC has 2 observations and 3 variables.
NOTE: The data set WORK.UNIQUE_REC has 7 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
       cpu time            0.01 seconds
```

Output Dataset: DUP_REC

Obs	Emp_ID	Emp_Name	Dept
1	110030	Mary	Finance
2	110030	Mary	Finance

Output Dataset: UNIQUE_REC

Obs	Emp_ID	Emp_Name	Dept
1	110011	Jack	Sales
2	110012	Jonny	Finance
3	110016	Jill	Sales
4	110018	Harry	Admin
5	110025	Harry	Admin
6	110031	Potter	Finance
7	110041	Tom	Admin

CONCLUSION

This was a brief discussion on removing duplicates using the readily available options in SAS in combination with PROC SORT. There are various other ways in which we can remove duplicate records. For instance using PROC SQL, data step, Macros etc.

REFERENCES

SAS Institute Inc. (2012), “Base SAS(R) 9.3 Procedures Guide, Second Edition” Cary, NC.

ACKNOWLEDGEMENTS

I would like to thank Dr. Joni Shreve for her valuable guidance and support while I was writing this paper.

CONTACT INFORMATION

Your comments and questions are valued and welcomed. You can contact the author at:

Jaya Dhillon, MSA Student
Louisiana State University
Phone: 919-324-445
E-Mail: jdhill1@lsu.edu