

Selective Output in SAS

By: Zenon Kurjewicz

Zenon_Kurjewicz@umanitoba.ca

Background

- Problem found while examining a very large dataset
 - 100,000+ rows
 - 40+ columns
- Duplicate error
- Needed to examine the duplicate rows
- Created a mock dataset to illustrate the problem.

Mock Dataset

- Mock dataset looks at degree data.
 - Years '03-'05
 - 17 variables (age, gender, degree year, degree, etc.)
- The file was loaded...

```
/* Load the file. */  
  
proc import out = degrees  
  datafile = "d:\SAS_presentation\degrees_present_final.xls"  
  dbms = EXCEL2000 replace;  
  getnames = YES;  
  mixed = YES;  
run;
```

Working with the Dataset

- A copy of the file was made.
- Note: there were 4829 records in the dataset.

```
8  
9 /* Make a copy. */  
10  
11 data copy;  
12 set degrees;  
13 run;  
  
NOTE: There were 4829 observations read from the data set WORK.DEGREES.  
NOTE: The data set WORK.COPY has 4829 observations and 17 variables.  
NOTE: DATA statement used (Total process time):  
      real time          0.21 seconds  
      cpu time           0.03 seconds
```

How Many Unique Students?

- I wondered – how many unique students did I have?
 - 4716 unique student ids.
- Did this indicate a problem?
 - I needed more information, because students could have had multiple degrees.

```
14  /* Make a distinct copy. */
15
16  proc sql;
17
18      create table id_distinct as
19          select distinct id
20          from copy;
NOTE: Table WORK.ID_DISTINCT created, with 4716 rows and 1 columns.
21  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.15 seconds
      cpu time           0.03 seconds
```

Duplicate Students

- I needed to create a file to look at the duplicate students.
 - I used proc sql to create the dataset.
 - The dataset created had all records for students where a student id was found more than once.

```
proc sql;

    create table duplicate_students as
    select *
    from copy
    group by id
    having count(id) > 1;

quit;
```

Examining the Dupes

- There were 224 duplicate rows
- There were some reasonable duplicates:
 - For example a student that received an Arts degree in 2003 and then received an Education degree in 2005.
- There were some questionable duplicates:
 - For example a Law student that appeared to have two identical records.
- How many complete duplicate rows were there?

Complete Duplicates

- I needed to find the complete duplicates ..by first looking at the unique records.
 - There were 4819 unique records.
 - 10 records were duplicates.

```
/* Run the distinct code for all the variables. */  
  
proc sql;  
    create table complete_distinct as  
    select distinct *  
    from copy;  
quit;
```

Remove the Dupes!

- It's hard to remove the dupes because one row is legitimate.
 - You can't eliminate rows simply based on a variable.
- I used **first.** and **output** to remove the duplicates.
 - **first.** is a SAS option that will allow me to identify the unique rows.
 - The **output** option allows me to specify which records to keep.

Remove the Dupes!

- I first sorted the file on all the variables.
- I then used **first.** to identify the records that were going to be unique..
- ..and used **output** to put them into a table.
- Notice the log
 - 4829 records in the original dataset
 - 4819 records in the output file (unique_rows)

```
44 data unique_rows;
45 set copy;
46 by id degree sex age major faculty_code faculty degree_yr
47 city province country university postal type cheated
48 b1_h wpg_other;
49 if first.wpg_other then
50 output;
51 run;
```

NOTE: There were 4829 observations read from the data set WORK.COPY.
NOTE: The data set WORK.UNIQUE_ROWS has 4819 observations and 17 variables.
NOTE: DATA statement used (Total process time):
real time 0.06 seconds
cpu time 0.04 seconds

Who Were the Dupes?

- We may want to examine the dupes, though.
 - See if there was commonality in the records, or some identifiable explanation for the multiple records.
- Add a second table: easy using the **output** statement.
 - Those not in the first table, go in the second table.

```
55 data unique_rows duplicate_rows;
56   set copy;
57   by id degree sex age major faculty_code faculty degree_yr
58     city province country university postal type cheated
59     bl_h wpg_other;
60   if first.wpg_other then
61     output unique_rows;
62   else output duplicate_rows;
63 run;
```

NOTE: There were 4829 observations read from the data set WORK.COPY.
NOTE: The data set WORK.UNIQUE_ROWS has 4819 observations and 17 variables.
NOTE: The data set WORK.DUPLICATE_ROWS has 10 observations and 17 variables.
NOTE: DATA statement used (Total process time):
real time 0.10 seconds
cpu time 0.07 seconds

Review

- Using the **output** statement:
 - Allowed us to remove duplicate rows from our dataset.
 - Allowed us to identify the duplicate rows, so they could be reviewed.
 - Allowed us to do it in one data step.

Other Uses of Output

- Another use for the **output** statement is for bursting.
 - Create multiple tables, based on if statements, using one data step.
 - Efficient way to create group specific data.
 - Example using degree data is to create faculty specific tables.

```
data arts science engineering other;
  set unique_rows;
  if faculty_code = '01' then
    output arts;
  else if faculty_code = '02' then
    output science;
  else if faculty_code = '03' then
    output engineering;
  else output other;
run;
```

Other Uses of Output

- Another use of the output statement is for error checking
 - If bursting, an additional table could be added to log any unexpected data.
 - In the degree dataset example, years 2003, 2004 and 2005 are expected. The “error” table would log any records that did not match one of the expected years.

```
data yr2005 yr2004 yr2003 error;
  set unique_rows;
  if degree_yr = '2005' then
    output yr2005;
  else if degree_yr = '2004' then
    output yr2004;
  else if degree_yr = '2003' then
    output yr2003;
  else output error;
run;
```

Epilogue

- Other ways to identify duplicate rows
 - Using the `proc sort`, an option of `nodup` can be used.
 - Deletes duplicate rows.
 - Using the `proc sort`, an option of `nodupkey` can be used.
 - Uses the by variables as its criteria
 - Selects the first instance of the by variables, ignores other instances.