Paper 1561-2014

Top 10 SQL Tricks in SAS®

Chao Huang, Oklahoma State University

ABSTRACT

One great thing separating SAS from other statistical languages is that SAS has native SQL (Structured Query Language) capacity by its SQL procedure. In addition to its CURD (Create, Read, Update and Delete) features, SQL also significantly enhances SAS's power in descriptive statistics and data management. In this paper, we introduce ten useful tips by the SQL procedure for everyday practice of SAS.

INTRODUCTION

PROC SQL is the implementation of the SQL syntax in SAS. It first appeared in SAS 6.0, and since then has been very popular for SAS users. SAS ships with a few sample data sets in its HELP library, and SASHELP.CLASS is one of them. This dataset contains 5 variables including name, weight, height, sex and age for 19 simulated teenagers, and in this paper I primarily use it for the demonstration purpose. Here I summarize the 10 interesting tricks and tips using PROC SQL. At the beginning, I first make a copy of SASHELP.CLASS at the WORK library and transform the row number of the data set to a new variable *obs*.

```
data class;
   set sashelp.class;
   /* Give an index for each child*/
   obs = _n_;
run;
```

Tricks FOR THE SQL PROCEDURE IN SAS

1. Calculate the median of a variable

With the aggregating HAVING clause and some self-join techniques, PROC SQL can easily calculate the median for a variable.

```
proc sql;
   select avg(weight) as Median
   from (select e.weight
   from class e, class d
   group by e.weight
   having sum(case when e.weight = d.weight then 1 else 0 end)
      >= abs(sum(sign(e.weight - d.weight))));
quit;
```

2. Draw a horizontal histogram

A histogram visualizes the distribution pattern of a variable. PROC SQL can draw a horizontal histogram by showing the frequency bars with a few asterisks for each level of the variable *age*.

```
proc sql;
   select age, repeat('*',count(*)*4) as Frequency
   from class
   group by age
   order by age;
quit;
```

3. Return the running total for a variable

A running total is the summation of a sequence of numbers which is updated each time with the increase of the observations. In the example below, I calculate the running total and save them as a new variable *Running_total* by the SUM function and a conditional statement, which logically is similar to an example in SAS/IML[1].

```
proc sql;
   select name, weight,
      (select sum(a.weight) from class as
      a where a.obs <= b.obs) as Running_total
   from class as b;
quit;
```

4. Report the total number for a variable

PROC SQL is a flexible way to find the total number for any variable by its set operator UNION and the SUM function. In the example, the total number of the variable *weight* is reported at the bottom of the output table.

```
proc sql;
   select name, weight
   from class
   union all
   select 'Total', sum(weight)
   from class;
quit;
```

5. Retrieve the metadata for a data set

SAS stores the metadata at its DICTIONARY data sets. PROC SQL can visit the directory, retrieve the column detail, and return the information to the users.

```
proc sql;
   select name, type, varnum
   from sashelp.vcolumn
   where libname = 'WORK' and memname = 'CLASS';
quit;
```

6. Rank a variable

Besides the designated ranking procedure PROC RANK in SAS, PROC SQL can also do some simple ranking as well.

```
proc sql;
   select name, a.weight, (select count(distinct b.weight)
   from class b
   /* Rank by the ascending order for the weight variable*/
   where b.weight <= a.weight) as rank
   from class a;
quit;
```

7. Simple random sampling

PROC SQL is widely used in simple random sampling. For example, I randomly choose 8 observations by the OUTOBS option at the PROC statement. The randomization process is realized by the RANUNI function at the ORDER BY statement with a seed 1234.

```
proc sql outobs = 8;
   select *
   from class
   order by ranuni(1234);
quit;
```

8. Replicate a data set without data

In PROC SQL, it is a fairly straightforward one-line statement to create a new empty data set while keeps all the structure of the original data set.

```
proc sql;
   create table class2 like class;
quit;
```

9. Transpose data

Usually DATA step ARRAY and PROC TRANSPOSE allow SAS users to restructure the data set, while PROC SQL sometimes is an alternative solution. For instance, if we need a wide-to-long operation to list the names of the children by their gender in the CLASS date set, then PROC SQL can fulfill the functionality through the combinations of some queries and subqueries.

```
proc sql;
   select max(case when sex='F'
      then name else ' ' end) as Female,
      max(case when sex='M'
      then name else ' ' end) as Male
   from (select e.sex,
      e.name,
      (select count(*) from class d
      where e.sex=d.sex and e.obs < d.obs) as level
      from class e)
   group by level;
quit;
```

10. Count the missing values

Another advantage of PROC SQL is that its NMISS function works for both numeric and character variables [2], which makes PROC SQL an ideal tool for missing value detection.

```
proc sql;
   select count(*) 'Total', nmiss(weight)
      'Number of missing values for weight'
   from class;
quit;
```

CONCLUSION

The combination of SAS's powerful functions and the SQL procedure will benefit SAS users in data management and descriptive statistics.

REFERENCES

1. Rick Wicklin. ‘Compute a running total for a "window" of time’. The DO Loop. June 22, 2012. http://blogs.sas.com/content/iml/2012/06/22/compute-a-running-total/
2. Chao Huang, Yu Fu. ‘Top 10 Most Powerful Functions for PROC SQL’. SAS Global Forum Proceeding 2013. http://support.sas.com/resources/papers/proceedings13/257-2013.pdf

CONTACT INFORMATION

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

Chao Huang
Oklahoma State University
221 PIO Building
Stillwater, OK. 74075
Email: hchao8@gmail.com
Web: www.sasanalysis.com