Paper 072-2007

Calculating Statistics Using PROC MEANS versus PROC SQL

Jyotheeswara Naidu Yellanki, Newark, DE, USA.

ABSTRACT:

Base SAS provided the PROC MEANS, which was very powerful/flexible procedure used to perform descriptive statistical analysis. This lead to a widespread use of MEANS procedure, as a result it became very popular amongst the older generation analysts. The proliferation of Relational Data base Management System (RDBMS) in information technology world lead to the introduction of PROC SQL in SAS v6.0. Being as English like language, Structure Query Language (SQL) become very popular amongst the newer generation analysts. Current SAS developer community has some analysts who use PROC MEANS to do all of the statistical work and some who use PROC SQL to do their part of the statistical work. But PROC SQL can perform or cater majority of the functionality that a PROC MEANS can deliver. Hence there is no real need for either community to learn the other's PROCs. However analysts who do maintenance or enhancement projects need to be familiar or proficient with both PROCs. This presentation is intended to explain and emphasize the similarities and differences between SQL and MEANS procedures with examples. It will also act as a good reference document with ideas for MEANS users on how to code equivalent SQL and vice versa.

INTRODUCTION:

PROC SQL is a widely used language for retrieving and updating data in tables/views. Mainly it is used to retrieve data from RDBMS, calculate the descriptive statistics or summarize the data. The MEANS procedure provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations. Both PROC MEANS and PROC SQL are now part of the base SAS product. This Paper will attempt to compare and contrast the data analysis of MEANS Procedure with equivalent methods in SQL procedure (with SQL Procedure we can do many things, but this paper will discuss about only SELECT statement). This paper will not access the efficiency or other system issues.

SIMILARITIES AND DIFFERENCES:

SYNTAX:

THE GENERAL SYNTAX OF MEANS PROCEDURES IS:

```
PROC MEANS DATA=sas-dataset-name <option(s)> <statistic-keyword(s)>;
BY <DESCENDING> variable-1 <...  <DESCENDING> variable-n><NOTSORTED>;
 CLASS variable(s) </ option(s)>;
FREQ variable;
ID variable(s);
OUTPUT <OUT=SAS-data-set> <output-statistic-specification(s)>
<id-group-specification(s)> <maximum-id-specification(s)>
<minimum-id-specification(s)> </ option(s)> ;
TYPES request(s);
VAR variable(s) < / WEIGHT=weight-variable>;
WAYS list;
WEIGHT variable;
```

THE GENERAL SYNTAX OF SQL PROCEDURES IS:

```
SELECT <DISTINCT> object-item <,object-item>...
 <INTO :macro-variable-specification
<, :macro-variable-specification>...>
 FROM from-list
<WHERE sql-expression>
<GROUP BY group-by-item
<,group-by-item>...>
<HAVING sql-expression>
<ORDER BY order-by-item
<,order-by-item>...>;
```

Note: The order of the statements after PROC MEANS statement is not important. In PROC SQL procedure the order of the **clauses** are very important.

NAMING CONVENTIONS:

In SQL we will use RDBMS words and in MEANS we will use SAS words. The correlation between RDBMS words and SAS words are shown in the below table.

SAS Words	RDBMS Words
Data Set	Table
Observations	Rows
Variables	Columns

DESCRIPTIVE STATISTICS IN EACH PROCEDURE:

Here is the sort list of main statistical functions that wither of PROCs can or can't perform.

Descriptive Statistics Keyword	MEANS	SQL
CLM	Yes	No
CSS	Yes	Yes
CV	Yes	Yes
KURTOSIS | KURT	Yes	No
LCLM	Yes	No
MAX	Yes	Yes
MEAN | AVG	Yes	Yes
MIN	Yes	Yes
N	Yes	Yes
NMISS	Yes	Yes
PRT	No	Yes
RANGE	Yes	Yes
SKEWNESS | SKEW	Yes	No
STDDEV | STD	Yes	Yes
STDERR	Yes	Yes
SUM	Yes	Yes
SUMWGT	Yes	Yes
UCLM	Yes	No
USS	Yes	Yes
VAR	Yes	Yes

INPUT DATA SOURCE:

In MEANS procedure the input data is supplied through DATA=sas-datset-name. If this is omitted then it will take the most recently created SAS data set in that session. In SQL procedure we must have to give the table name in the FROM Clause.

OUTPUT FROM PROCEDURE :

Both the procedures will print the output in OUTPUT window or we can create SAS dataset/table. But in MEANS procedure the format of the data display in the output window is not same as the data created in the output SAS dataset. Whereas in SQL it is same.

DEFAULT STATISTICS:

By default the MEANS procedure will produce N (counts), mean, standard deviation, max and min on all numeric variables in the dataset. The MEANS procedure will not perform any statistics on character variables. If there are no numeric variable in the dataset then PROC MEANS procedure will only give the number of observations(N) in that dataset. Here is the simple code.

```
PROC MEANS DATA=sas-dataset-name; run;
```

SQL procedure will not calculate any statistics by default. But the SQL procedure will calculate some statistics on character variable. For Example sex is a char variable in SASHELP.CLASS dataset, we can perform max, min, n and nmiss values.

```
Proc SQL;
Select max(sex),Min(sex),n(sex),nmiss(sex)
From sashelp.class;
```

CODE COMPARISON:

Let us take couple of examples to compare the code between SQL and MEANS

Calculating the simple statistics:

First we will consider how the results are displayed in the output window without creating into dataset/table.

By default the MEANS procedure will produces N, mean, standard deviation, max and min on all numeric variables in the dataset. We select the required statistics by specifying the statistics keywords in the MEANS statement. Similarly we can restrict the variable on which you want perform the statistics by specifying the variable names in VAR statement. The following MEANS procedure will display the results in output window and calculate the SUM, MEAN and STD on Age and HEIGHT variable.

```
        PROC MEANS DATA=SASHELP.CLASS NONOBS MAXDEC=2 SUM MEAN STD ;
        VAR AGE HEIGHT;
        RUN;
```

```
Variable             Sum            Mean          Std Dev
----------------------------------------------------------
Age               253.00           13.32             1.49
Height           1184.40           62.34             5.13
```

NOTE: The option MXDEC= is used to limit the decimal places in the result.
NONOBS is used to suppress reporting the total number of observations for each unique combination of the class variables

To produce the same result as above, we have use the following SQL:

```
        PROC SQL;
        select 'Age' as Variable,
                sum(age) as Sum format 10.2,
                avg(age) as Mean format 10.2,
               std(age) as std  format 10.2 label 'Std Dev'
        from sASHELP.CLASS
        union
        select 'Height' as Variable,
                sum(height) as Sum format 10.2,
                avg(height) as Mean format 10.2,
                 std(height) as std  format 10.2 label 'Std Dev'
        from sASHELP.CLASS;
        QUIT;
```

GROUP PROCESSING USING CLASS/GROUP BY:

Often we want statistics for grouped observations instead of for whole observations. Use the CLASS statement to calculate the statistics based on category.

```
        PROC MEANS DATA=SASHELP.CLASS NONOBS MAXDEC=2 SUM MEAN STD ;
        CLASS sex;
        VAR AGE HEIGHT;
        RUN;
```

```
Sex    Variable             Sum             Mean         Std Dev
----------------------------------------------------------------
F      Age               119.00            13.22            1.39
       Height            545.30            60.59            5.02

M      Age               134.00            13.40            1.65
       Height            639.10            63.91            4.94
-----------------------------------------------------------------
```

To produce the similar result as above we have use the following SQL:

```
        PROC SQL;
        select sex as Sex,
               'Age' as Variable,
                sum(age) as Sum format 10.2,
                avg(age) as Mean format 10.2,
                std(age) as std  format 10.2 label 'Std Dev'
        from SASHELP.CLASS
        Group by
        select  sex as Sex,
                'Height' as Variable,
                 sum(height) as Sum format 10.2,
                avg(height) as Mean format 10.2,
                std(height) as std  format 10.2 label 'Std Dev'
        from sASHELP.CLASS
        group by sex;
        quit;
```

But the SQL output looks slightly different.

```
Sex  Variable         Sum        Mean     Std Dev
-------------------------------------------------
F    Age           119.00       13.22        1.39
F    Height        545.30       60.59        5.02
M    Age           134.00       13.40        1.65
M    Height        639.10       63.91        4.94
```

The CLASS or GROUP BY variable can be numeric or char, but they should contain a limited number of discrete values that represents meaningful groupings.

GROUP PROCESSING USING CLASS WITH TYPES STATEMENT IN PROC MEANS:

By default Grouping will takes place all combinations of CLASS variables. By using TYPES statement we can select overall or individual CLASS variable. For example.

```
       data grade;
       input Name $ Gender $ Status $  Year $ Section $ Score  FinalGrade @@;
       datalines;
       Abbott    F 2 97 A 90 87 Branford  M 1 97 A 92 97
       David     M 3 99 c 87 96 Crandell  M 2 98 B 81 71
       Dennison  M 1 97 A 85 72 Edgar     F 1 98 B 89 80
       Nancy     F 3 99 B 79 88 Faust     M 1 97 B 78 73
       Greeley   F 2 97 A 82 91 Hart      F 1 98 B 84 80
       Mick      M 3 98 c 77 91 Isley     M 2 99 A 88 86
       Jasper    M 1 97 B 91 93 Ray       M 3 97 B 76 90
       Mary      F 3 97 C 75 79 Nick      M 2 98 B 85 89
       Billy     M 2 98 C 77 83 Taylor    F 3 98 A 86 81
       Roy       M 3 98 B 92 84 Mandy     F 3 97 C 88 87
       ;
       run;
       proc means data=grade nonobs n mean sum maxdec=2;
       class Status Year;
       var Score;
       types () status*year;
       run;
```

It will produce overall counts, means and sum on entire data as one output and by grouping status and year as separate output. The results looks like this.

Output due to :Types ()

```
   N     Mean          Std Dev
-----------------------------------
  20     84.10          1682.00
-----------------------------------
```

Output due to: types status*year

```
Status      Year         N            Mean             Sum
-----------------------------------------------------------
1           97           4           86.50          346.00
            98           2           86.50          173.00
2           97           2           86.00          172.00
            98           3           81.00          243.00
```

```
          99          1           88.00          88.00
3         97          3           79.67         239.00
          98          3           85.00         255.00
          99          2           83.00         166.00
-----------------------------------------------------------
```

The SQL code to produce similar output as below.

```
       proc sql;
       select count(*)   as N,
              avg(score) as Mean format 10.2 ,
              std(score)  as Sum  format 10.2
       from grade;
       quit;

       proc sql;
       select status,year,
               count(*) as N,
               avg(score) as Mean format 10.2 ,
               sum(score)  as Sum  format 10.2
       from grade
       group by status, year;
       quit;
```

Exclude analysis variable values that are not in CLASSDATA.

In PROC MEANS we can specify a secondary data set that contains the combinations of class variables to analyze and exclude from the analysis for all combinations of class variable values that are not in the CLASSDATA= data set For example we want select only the class variables combination listed below SAS dataset statyesr then use the SAS dataset name in CLASSDATA= option in PROC MEANS statement.

```
       data statyear;
       input Status $  Year $ @@;
       datalines;
       1 97 1 98 2 97 2 98 3 97 3 99
       ;
       run;

       proc means data=grade nonobs nway n mean sum maxdec=0
       classdata=statyear exclusive;
       class status year;
       var score;
```

The excluded output looks like

```
Status      Year        N             Mean            Sum
-----------------------------------------------------------
1           97          4           86.50         346.00
            98          2           86.50         173.00
2           97          2           86.00         172.00
            98          3           81.00         243.00
3           97          3           79.67         239.00
            99          2           83.00         166.00
------------------------------------------------------------
```

We can produce the same result using SQL with sub quires

```
       proc sql;
       select status,year,
               count(*) as N,
              mean(score) as Mean format 10.2 ,
              sum(score)  as Sum  format 10.2
       from grade
```

```
      where status||year  in (select status||year from statyear)
      group by status, year;
      quit;
```

Creating output SAS dataset from PROC MEANS / PROC SQL:

All the examples listed above are producing the output in the printed report form. Now we discuss how to create the results in SAS datasets using PROC MEANS and PROC SQL.

In PROC MEANS using OUTPUT statement we can create the SAS dataset. The syntax is
OUTPUT <OUT=SAS-data-set> <output-statistic-specification(s)>
The output-statistic-specification(s) may be one or more of the following forms. In each form, stat is a statistics request keyword.
Form 1: stat=name-list. The name list specifies variables containing stat. The variables in name list have a one-to-one correspondence with the variable listed in the VAR statement.

Form 2: stat (varlist) =name-list. The name list identifies variables containing statistic stat for analysis variables varlist. The variables in name list have a one-to-one correspondence with the variable listed in the VAR statement.

Form 3: stat=. This form requests an output dataset that contains the same variables as in the VAR statement list.

Form 4: stat(varlist)=. This form specifies that VAR statement variables in varlist represent statistic stat in the output dataset. This form and form 3 may naot be used in the same OUTPUT statement and vice versa.

For example

```
      Proc means data=grade nonobs maxdec=2 noprint;
      Class status year;
      Var  score finalgrade;
      Output out=sumgrade  max=scr_max grade_max  mean(score finalgrade) = scr_mean
      grade_mean  sum=;
      format scr_mean grade_mean 10.1;
      Run;
      proc print data=sumgrade;run;
```

This will create two automatic variables _type_ and _FREQ_.
TYPE identifies the level of summary.
Value of _TYPE_ = 0 indicates the summary record for the entire dataset.
Value of _TYPE_ = 1 identifies summary data for each level of year across all status(status is ignored).
Value of _TYPE_ = 2 identifies summary data for each level of status across all year (year is ignored).
Value of _TYPE_ = 3 identifies summary data contain statistics for each level of status within each level of year.

Note: The option NWAY in the PROC MEANS statement will select the maximum value of _TYPE_.

The _FREQ_ is the number of observations in each level of summary.

The output from the above code will look like:

```
                                        grade_                                Final
Status  Year  _TYPE_  _FREQ_  scr_max    max     scr_mean  grade_mean  Score  Grade

                  0      20       92      97        84.1        84.9   1682   1698
          97      1       9       92      97        84.1        85.4    757    769
          98      1       8       92      91        83.9        82.4    671    659
          99      1       3       88      96        84.7        90.0    254    270
   1              2       6       92      97        86.5        82.5    519    495
   2              2       6       90      91        83.8        84.5    503    507
   3              2       8       92      96        82.5        87.0    660    696
   1      97      3       4       92      97        86.5        83.8    346    335
   1      98      3       2       89      80        86.5        80.0    173    160
   2      97      3       2       90      91        86.0        89.0    172    178
   2      98      3       3       85      89        81.0        81.0    243    243
   2      99      3       1       88      86        88.0        86.0     88     86
   3      97      3       3       88      90        79.7        85.3    239    256
```

```
3     98     3     3     92     91         85.0       85.3    255    256
3     99     3     2     87     96         83.0       92.0    166    184
```

To get the same results from PROC SQL, use the following code

```
PROC SQL;
Create table sumgrade as
Select  ' ' as Status, ' ' as Year ,0 as _TYPE_ , count(*) as _FREQ_,
           Max(score) as scr_max, max(finalgrade) as grade_mean ,
          avg(score) as scr_max format 10.1,
        avg(finalgrade) as grade_mean format 10.1,
        sum(score) as score, sum(finalgrade) as finalgrade
from grade
union
Select  ' ' as Status, Year ,1 as _TYPE_ , count(*) as _FREQ_,
        Max(score) as scr_max, max(finalgrade) as grade_mean ,
        avg(score) as scr_max format 10.1 ,
        avg(finalgrade) as grade_mean format 10.1,
       sum(score) as score, sum(finalgrade) as finalgrade
from grade
group by year
union
Select   Status, ' ' as Year ,2 as _TYPE_ , count(*) as _FREQ_,
       Max(score) as scr_max, max(finalgrade) as grade_mean ,
       avg(score) as scr_max format 10.1 ,
       avg(finalgrade) as grade_mean format 10.1,
       sum(score) as score, sum(finalgrade) as finalgrade
from grade
group by status
union
Select   Status,  Year ,3 as _TYPE_ , count(*) as _FREQ_,
       Max(score) as scr_max, max(finalgrade) as grade_mean ,
       avg(score) as scr_max format 10.1 ,
       avg(finalgrade) as grade_mean format 10.1,
       sum(score) as score, sum(finalgrade) as finalgrade
from grade
group by status,year
order by _TYPE_ ;
quit;
proc print data=sumgrade; run;
```

Here is table with the summaries of our findings.

Description	MEANS	SQL
Input data	DATA=dataset-name	FROM table-name
Output dataset	OUTPUT OUT=temp	CREATE TABLE temp AS
Analysis variables	VAR var1 var2	SELECT *stat*(var1),*stat*(var2)
Grouping Variables	CLASS var1 var2	GROUP BY var1,var2
Statistics to be performed	Specify in PROC MEANS statement	Specify in SELECT statement

Note: in the above table the *stat* is any statistical function.

CONCLUSION:

This paper covered frequently used options and statements in PROC MEANS and equivalent code using PROC SQL.. This paper will be a good reference document those who are proficient in PROC MEANS and need work with PROC SQL and vice versa. There are certain functionality that are provided by PROC MEANS which can't be replicated by using PROC SQL and vice versa.

REFERENCES:

SAS Institute Inc., SAS OnlineDoc ® 9, http://v9doc.sas.com/sasdoc/
SAS Institute Inc., Base SAS 9.1 ® Procedures Guide.

CONTACT INFORMATION:

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

Jyotheeswara Naidu Yellanki
Email: yjnaidu@hotmail.com