PharmaSUG 2012 - Paper HW04

Ready To Become Really Productive Using PROC SQL?

Sunil K. Gupta, Gupta Programming, Simi Valley, CA

ABSTRACT

Using PROC SQL, can you identify at least four ways to: select and create variables, create macro variables, create or modify table structure, and change table content? Learn how to apply multiple PROC SQL programming options through task-based examples. This hands-on workshop reviews topics in table access, retrieval, structure and content, as well as creating macro variables. References are provided for key PROC SQL books, relevant webinars, podcasts as well as key SAS® technical papers.

INTRODUCTION

```
PROC SQL;             /* Anatomy of PROC SQL */

CREATE table mytable as

/* Nine Benefits: Validate/Create/Drop Views or Tables,
                  Create/Alter/Update/Insert/Delete Variables */

/* Four Main Components: SELECT, FROM, WHERE, ORDER */

1. SELECT name, sex

 /* Four selection options: ‘,’, label, ‘*’, distinct */
 /* Eight creation options: functions, summary function, constant,
    character expression, select-case when, select-case <var_name> when, summary
function with subset condition */
 /* Five macro variable creation options: into :, into : separated by,
    into : - :, summary function into:, select-case into: */

2. FROM sashelp.class as class,
   Mylib.students as students

 /* Four join options: inner matching/outer LEFT/FULL/RIGHT JOIN */
 /* FROM <DS1> <FULL JOIN> <DS2>  ON <DS1.VAR1> = <DS2.VAR2> */

3. WHERE class.name = students.name and class.sex = ‘F’

 /* Four subsetting options: direct/calculated variable, function, summary function */
 /* Two main subquery options: one/multiple values with HAVING <Variable> <Operator>
  (SELECT <Variable> FROM <Table> WHERE <Condition Expression>) */

4. ORDER by name
/* Two sorting options: order/group by calculated, desc */

; QUIT;
```

PROC SQL Examples

1. Basic example - Essential building block components (Columns, Joins, Condition, Sort)

2. Selecting Column Definitions
a. Basic Structure
b. Column Attributes
c. All Columns
d. Distinct Columns
e. Distinct Columns without Order By

3. Creating Column Definitions
a. Functions such as int()
b. Functions such as max()
c. Summary Functions such as sum()
d. Constant
e. Character String Expression
f. Select-Case When Condition
g. Select-Case <Var_Name> When Condition
h. Summary function with subset condition

4. Subsetting Tables
a. Age Calculation
b. Function such as index()

5. Subqueries
a. Resulting in One row
b. Resulting in Multiple rows

6. Creating Macro Variables
a. One macro variable storing one value
b. One macro variable storing multiple values

SAMPLE DATA SET

Below is the sample data set that will be used in the examples. The data set sashelp.class is included with SAS/BASE.

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Barbara	F	13	65.3	98.0
3	Carol	F	14	62.8	102.5
4	Jane	F	12	59.8	84.5
5	Janet	F	15	62.5	112.5
6	Joyce	F	11	51.3	50.5
7	Judy	F	14	64.3	90.0
8	Louise	F	12	56.3	77.0
9	Mary	F	15	66.5	112.0
10	Alfred	M	14	69.0	112.5
11	Henry	M	14	63.5	102.5
12	James	M	12	57.3	83.0
13	Jeffrey	M	13	62.5	84.0

Obs	Name	Sex	Age	Height	Weight
14	John	M	12	59.0	99.5
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Figure 1. Sample Dataset

Example 1. Basic example - Essential building block components (Columns, Joins, Condition, Sort)

Between the required PROC SQL and quit statements, in general, there is only one PROC SQL statement. Each of the four components is included in this statement: columns, joins, conditions and sorts. Note that when creating a table, it is recommended to create a new table to prevent SAS warning message about writing to the same table.

1. Select name for all females.

```
title1 "HOW Example 1: Basic Example with all four components";
title2 "Four components: A. Columns (), B. Joins (), C. Condition (), D. Sort ()";
title3 "Which components are required?";
proc sql;
 select name
 from sashelp.class
 where sex = 'F'
 order by name;
quit;
```

Name
Alice
Barbara
Carol
Jane
Janet
Joyce
Judy
Louise
Mary

Figure 2. All Female NAMES

Example 2. Selecting Column Definitions
a. Basic Structure
b. Column Attributes
c. All Columns
d. Distinct Columns
e. Distinct Columns without Order By

2a. Select name and sex for all females.

Multiple columns are separated by ‘,’.

```
title1 "HOW Example 2a: Select Columns (name, sex)";
proc sql;
 select name, sex
 from sashelp.class
 where sex = 'F'
 order by name;
quit;
```

Name	Sex
Alice	F
Barbara	F
Carol	F
Jane	F
Janet	F
Joyce	F
Judy	F
Louise	F
Mary	F

Figure 3. All Female NAMES and SEX

2b. Define attributes for name: label, format and length.

Note that although name is a character variable, the length does not include ‘$’ as in the DATA Step.

```
title1 "HOW Example 2b: Select Columns (Add Attributes – label, format and length)";
proc sql;
 select name label = 'My label' format = $10. length = 10
 from sashelp.class
 where sex = 'F'
 order by name;
quit;
```

My label
Alice
Barbara
Carol
Jane

My label
Janet
Joyce
Judy
Louise
Mary

Figure 4. Attributes for NAME

2c. Select all columns in table for all females.

```
title1 "HOW Example 2c: Select All Columns (*)";
proc sql;
 select *
 from sashelp.class
 where sex = 'F'
 order by name;
quit;
```

Name	Sex	Age	Height	Weight
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112

Figure 5. All columns for all Females

2d. Select distinct sex for all females.

```
title1 "HOW Example 2d: Select Columns (distinct)";
proc sql;
 select distinct sex
 from sashelp.class
 where sex = 'F'
 order by name;
quit;
```

Sex
F
F
F

Sex
F
F
F
F
F
F

Figure 6. Distinct SEX for all Females

2e. Select distinct sex for all females without repeats.

```
title1 "HOW Example 2e: Selecting Columns (distinct) without order by clause";
proc sql;
 select distinct sex
 from sashelp.class
 where sex = 'F';
quit;
```

Sex
F

Figure 7. Distinct SEX for all Females without ORDER BY NAME

Example 3. Creating Column Definitions
a. Functions such as int()
b. Functions such as max()
c. Summary Functions such as sum()
d. Constant
e. Character String Expression
f. Select-Case When Condition
g. Select-Case <Var_Name> When Condition
h. Summary function with subset condition

Note that new variable names are specified towards the end as compared to the beginning in the DATA Step. In general for all new columns, remember to specify a length, especially since character columns can be truncated.

3a. Create age and calculated age using functions.

```
title1 "HOW Example 3a: Creating Column using Functions int((age+150)/10) as myage";
proc sql;
 select age, int((age+150)/10) as myage length = 8 format = 3.
 from sashelp.class;
quit;
```

Age	myage
14	16
13	16
13	16
14	16

Age	myage
14	16
12	16
12	16
15	16
13	16
12	16
11	16
14	16
12	16
15	16
16	16
12	16
15	16
11	16
15	16

Figure 8. AGE and MYAGE columns

3b. Create height, weight and max of height and weight.

```
title1 "HOW Example 3b: Creating Column using Functions max(height, weight) as
maxval";
proc sql;
 select height, weight,
  max(height, weight) as maxval length = 8 format = 3.
 from sashelp.class;
quit;
```

Height	Weight	maxval
69	112.5	113
56.5	84	84
65.3	98	98
62.8	102.5	103
63.5	102.5	103
57.3	83	83
59.8	84.5	85
62.5	112.5	113
62.5	84	84
59	99.5	100

Height	Weight	maxval
51.3	50.5	51
64.3	90	90
56.3	77	77
66.5	112	112
72	150	150
64.8	128	128
67	133	133
57.5	85	85
66.5	112	112

Figure 9. HEIGHT, WEIGHT and MAXVAL columns

3c1. Create weight and percent of total weight using summary functions.

The new summary variable is added back to the data set. Note that this would generally require multiple DATA Steps or often SAS procedures such as PROC MEANS.

```
title1 "HOW Example 3c1: Creating Column using Summary Functions(
(weight/sum(weight))*100) as wpercnt";
proc sql;
 select weight,
  ((weight/sum(weight))*100) as wpercnt length = 8 format = 4.1
 from sashelp.class;
quit;
```

Weight	wpercnt
112.5	5.9
84	4.4
98	5.2
102.5	5.4
102.5	5.4
83	4.4
84.5	4.4
112.5	5.9
84	4.4
99.5	5.2
50.5	2.7
90	4.7
77	4.1
112	5.9
150	7.9

Weight	wpercnt
128	6.7
133	7.0
85	4.5
112	5.9

Figure 10. WEIGHT and WPERCNT columns

3c2. Create sex, weight and percent of total weight by sex.

The previous PROC SQL code can be grouped by sex to get weight percents by sex instead of by overall weight. Once sum_weight is created, it can be used on the wpercnt calculation with the CALCULATED keyword before sum_weight.

```
title1 "HOW Example 3c2: Creating Column Group by Sex (sum(weight)
(weight/sum(weight))*100) as wpercnt";
proc sql;
 select sex, weight, sum(weight) as sum_weight,
  ((weight/sum(weight))*100) as wpercnt length = 8 format = 4.1
 from sashelp.class
 group by sex;
quit;
```

Sex	Weight	sum_weight	wpercnt
F	90	811	11.1
F	84.5	811	10.4
F	50.5	811	6.2
F	98	811	12.1
F	102.5	811	12.6
F	112	811	13.8
F	77	811	9.5
F	84	811	10.4
F	112.5	811	13.9
M	150	1090	13.8
M	83	1090	7.6
M	102.5	1090	9.4
M	99.5	1090	9.1
M	112	1090	10.3
M	112.5	1090	10.3
M	84	1090	7.7
M	85	1090	7.8

Sex	Weight	sum_weight	wpercnt
M	133	1090	12.2
M	128	1090	11.7

Figure 11. SEX, WEIGHT, SUM_WEIGHT and WPERCNT columns

3c3. Create sex, name and total sex by sex.

Another example of adding count by sex back to the data set.

```
title1 "HOW Example 3c3: Creating Column Group by Sex for each name count(sex) as
gender_cnt";
proc sql;
 select sex, name, count(sex) as gender_cnt length=8 format=4.0
 from sashelp.class
 group by sex
 order by sex;
quit;
```

Sex	Name	gender_cnt
F	Judy	9
F	Jane	9
F	Joyce	9
F	Barbara	9
F	Carol	9
F	Mary	9
F	Louise	9
F	Alice	9
F	Janet	9
M	Philip	10
M	James	10
M	Henry	10
M	John	10
M	William	10
M	Alfred	10
M	Jeffrey	10
M	Thomas	10
M	Ronald	10
M	Robert	10

Figure 12. SEX, NAME and GENDER_CNT columns

3d. Create constant for all records.

```
title1 "HOW Example 3d: Creating Column using 'my constant' as myname";
proc sql;
 select
  'my constant' as myname length = 15
 from sashelp.class;
quit;
```

myname
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant
my constant

Figure 13. MYNAME column

3e. Create character expression name and sex.

```
title1 "HOW Example 3e: Creating Column using character expression '  name || ',' ||
sex as namesex";
proc sql;
 select
  name || "," || sex as namesex length = 35
 from sashelp.class;
quit;
```

namesex
Alfred ,M
Alice ,F
Barbara ,F
Carol ,F
Henry ,M
James ,M
Jane ,F
Janet ,F
Jeffrey ,M
John ,M
Joyce ,F
Judy ,F
Louise ,F
Mary ,F
Philip ,M
Robert ,M
Ronald ,M
Thomas ,M
William ,M

Figure 14. NAMESEX column

3f. Create age and new column agegrp based on age values.

For any type of conditional logic within PROC SQL, you will need to apply the select-case clause.

```
title1 "HOW Example 3f: Creating Column using select-case when to create agegrp";
proc sql;
 select  age,
  case
       when age > 0 and age < 13  then 1
       when age between 13 and 15 then 2
       when age > 15              then 3
     else                           .
  end as agegrp length = 8
 from sashelp.class;
quit;
```

Age	agegrp
14	2
13	2
13	2

Age	agegrp
14	2
14	2
12	1
12	1
15	2
13	2
12	1
11	1
14	2
12	1
15	2
16	3
12	1
15	2
11	1
15	2

Figure 15. AGE and AGEGRP columns

3g. Alternative to create sex and new column sexgrp based on sex values.

For any type of conditional logic within PROC SQL, you will need to apply the select-case clause. You have the option to specify sex only once after CASE. Each when clause will automatically insert sex as part of the evaluation.

```
title1 "HOW Example 3f: Creating Column using select-case when to create sexgrp";
proc sql;
 select  sex,
  case sex
       when 'M'  then 1     /* similar to when sex = 'M' */
       when 'F'  then 2     /* similar to when sex = 'F' */
     else        .
  end as sexgrp length = 4
 from sashelp.class;
quit;
```

3h. Create count of males and females.

This is an alternative to applying the WHERE clause.

```
title1 "HOW Example 3h: Creating Columns using where condition in summary function";
proc sql;
 select
  sum(sex='M') as nmale length = 4, sum(sex='F') as nfemale length = 4
 from sashelp.class;
quit;
```

nmale	nfemale
10	9

Figure 16. NMALE and NFEMALE columns

Example 4. Subsetting Tables
a. Age Calculation
b. Function such as index()

4a. Select age and new column agegrp based on age values for agegrp = 3.

```
title1 "HOW Example 4a: Subsetting tables using calculated agegrp column";
proc sql;
 select  age,
  case
       when age > 0 and age < 13  then 1
       when age between 13 and 15 then 2
       when age > 15              then 3
     else                           .
  end as agegrp length = 4
 from sashelp.class
 where calculated agegrp = 3;
quit;
```

Age	agegrp
16	3

Figure 17. AGE and AGEGRP columns

4b. Select name, sex where name contains 'J'.

```
title1 "HOW Example 4b: Subsetting tables using Function index()";
proc sql;
 select
  name, sex
 from sashelp.class
 where index(name, 'J') > 0;
quit;
```

Name	Sex
James	M
Jane	F
Janet	F
Jeffrey	M
John	M
Joyce	F
Judy	F

Figure 18. NAME and SEX columns

Example 5. Subqueries
a. Resulting in One row
b. Resulting in Multiple rows

5a. Select sex, weight where weight is greater than the average weight.

Notice that with subqueries, you can select records from one table based on a conditions in another table.

```
title1 "HOW Example 5a: Using Subquery Conditions resulting in one row";
title2 'Select by sex, sex and weight, weight greater than the overall average
weight';
title3 'Three part approach: subquery results, population, confirm subset';

proc sql;
 create table mean_wgt as
 select avg(weight) as m_wgt from sashelp.class;

 select m_wgt from mean_wgt;
quit;

proc sql;
 Select sex, weight
 from sashelp.class
 order by sex, weight;
quit;

proc sql;
 Select sex, weight
 from sashelp.class
  having weight >
  (select m_wgt from mean_wgt);
quit;
```

100.0263

Figure 19. AVG(MEAN)

Sex	Weight
F	50.5
F	77
F	84
F	84.5
F	90
F	98
F	102.5
F	112
F	112.5
M	83
M	84
M	85

Sex	Weight
M	99.5
M	102.5
M	112
M	112.5
M	128
M	133
M	150

Figure 20. All SEX and WEIGHT records

Sex	Weight
M	112.5
F	102.5
M	102.5
F	112.5
F	112
M	150
M	128
M	133
M	112

Figure 21. SEX and WEIGHT records greater than AVG(WEIGHT)

5b. Select age where age does not equal any female ages.

```
title1 "HOW Example 5b: Using Subquery Conditions resulting in multiple rows";
title2 'Select by sex, sex and age, for non-matching female ages';
title3 'Three part approach: subquery results, population, confirm subset';

proc sql;
 select distinct age from sashelp.class where sex = "F" order by age;
quit;

proc sql;
 Select sex, age
 from sashelp.class
 order by sex;
quit;

proc sql;
 Select sex, age
 from sashelp.class
  having age ~in
 (select distinct age from sashelp.class where sex = "F") ;
quit;
```

Age
11
12
13
14
15

Figure 22. Distinct AGE column

Sex	Age
F	14
F	12
F	11
F	13
F	14
F	15
F	12
F	13
F	15
M	16
M	12
M	14
M	12
M	15
M	14
M	13
M	11
M	15
M	12

Figure 23. All SEX and AGE records

Sex	Age
M	16

Figure 24. Only Male AGE record

Example 6. Creating Macro Variables
a. One macro variable storing one value
b. One macro variable storing multiple values

6a. Create macro variable storing total male count.

Best to assure selection criteria displays desired single result before saving to macro variable.

```
title1 "HOW Example 6a: Creating macro variable resulting in one row";
title2 'Count of males';
title3 'Two part approach: value to save, macro variable name';

proc sql;
 select count(sex) as gender_cnt
 from sashelp.class
 where sex = 'M';
quit;

proc sql;
 select count(sex) as gender_cnt into :male_cnt
 from sashelp.class
 where sex = 'M';
quit;
%put 'Number of Males = ' &male_cnt;
```

gender_cnt
10

Number of Males = 10

Figure 25. GENERDER_CNT column

6b. Create macro variable storing male names.

Best to assure selection criteria displays desired multiple results before saving to macro variable.

```
title1 "HOW Example 6b: Creating macro variables resulting in multiple rows";
title2 'Count of males';
title3 'Two part approach: value to save, macro variable names';

proc sql;
 select name as male
 from sashelp.class
 where sex = 'M';
quit;

proc sql;
 select name into :male_name separated by ', '
 from sashelp.class
 where sex = 'M';
quit;
%put 'Names of Males = ' &male_name;
```

male
Alfred
Henry
James

male
Jeffrey
John
Philip
Robert
Ronald
Thomas
William

```
Names of Males = Alfred, Henry, James, Jeffrey, John, Philip, Robert, Ronald, Thomas,
William
```

Figure 26. Male NAMES column

SUMMARY

By understanding the subtle differences in the various combinations for selecting, joining, subsetting,and sorting using PROC SQL, you are more empowered to apply and remember the syntax. With the added bonus of creating one or more macro variables storing one or more values, SAS programmers can do it all with one powerful and unique procedure.

As presented, there are four main components to PROC SQL: select, joins, where, order. There are four column selection options, six column creation options and five macro variable creation options. For joining tables, there are four outer join options. For subsetting tables, there are three options and two subquery options. For sorting tables, there are two options.

Some examples of PROC SQL's flexibility include merging back summary-level information in one step as compared to multiple data steps or using PROC MEANS. Other useful table structure and content operators that are beyond the scope of this paper include: alter, drop, update, insert and delete. Please visit my SASCommunity.org page for updates to my list of top 10 PROC SQL papers.

REFERENCES

Bhat,Gajanan, "Merging Tables in DATA Step vs. PROC SQL: Convenience and Efficiency Issues", SUGI 26, Coder's Corner

DeFoor, Jimmy, "Proc SQL – A Primer for SAS® Programmers", SUGI 31, Tutorials

Gupta, Sunil K., Quick Results with PROC SQL, http://www.sascommunity.org/wiki/Quick_Results_with_Proc_SQL

Lafler, Kirk Paul, "A Hands-On Tour Inside the World of PROC SQL", SUGI 31 Hands-On Workshop

Lafler, Kirk Paul, "Frame Your View of Data with the SQL Procedure"

Lafler, Kirk Paul, "Querying the Data Warehouse with the SQL Procedure SELECT Statement", SUGI 23

Lafler, Kirk Paul, "Ten Great Reasons to Learn SAS Software's SQL Procedure", SUGI 23, Hands-On Workshop

Lafler, Kirk Paul, "Undocumented and Hard-to-find SQL Features", SUGI 28, Advanced Tutorials

Lafler, Kirk Paul, PROC SQL Tips and Techniques Webcast:
http://support.sas.com/publishing/bbu/webinar/Lafler_junewebinar.wmv

Whitlock, Ian, "PROC SQL - Is it a Required Tool for Good SAS Programming?", SUGI 26, Beginning Tutorials

Williams, Christianna, "PROC SQL for DATA Step Die-hards", SAS Global Forum 2008

Winn, Thomas, “Introduction to Using Proc SQL”, SUGI 22, Beginning Tutorials

CONTACT INFORMATION

The author welcomes your comments and suggestions.

Sunil K. Gupta
Senior SAS Consultant
Gupta Programming
213 Goldenwood Circle
Simi Valley, CA 93065
Phone: (805)-577-8877 E-mail: Sunil@GuptaProgramming.com
www.GuptaProgramming.com
www.SASSavvy.com

Sunil is a best selling SAS author and global corporate trainer. Currently, he is a Senior SAS Consultant at Gupta Programming. Most recently, he launched www.SASSavvy.com and released two popular e-guides on Quick Results with PROC SQL and Anatomy of SAS Macros. He has been using SAS® software for over 18 years and is a SAS Base Certified Professional. He is also the author of *Quick Results with the Output Delivery System*, *Data Management and Reporting Made Easy with SAS Learning Edition 2.0,* and *Sharpening Your SAS Skills.* Most recently, he is teaching his latest popular courses, Maximizing Productivity and Efficiency using PROC SQL and Best Practices in SAS Statistical Programming in Regulatory Submission.