Paper 1978-2015

Something for Nothing? Adding Group Descriptive Statistics Using SAS(R) PROC SQL Subqueries II

Sunil K. Gupta, Cytel, Simi Valley, CA

ABSTRACT

Can you actually get something for nothing? With the SAS® PROC SQL subquery and remerging features, yes, you can. When working with categorical variables, you often need to add group descriptive statistics such as group counts and minimum and maximum values for further BY-group processing. Instead of first creating the group count and minimum or maximum values and then merging the summarized data set to the original data set, why not take advantage of PROC SQL to complete two steps in one?

With the PROC SQL subquery and summary functions by the group variable, you can easily remerge the new group descriptive statistics with the original data set. Now with a few DATA step enhancements, you too can include percent calculations.

INTRODUCTION

While Proc SQL can be complex, it is also a powerful and concise programming language. With a few years of experience and applying basic Proc SQL code on a daily basis, SAS programmers may soon realize that combining several benefits of Proc SQL in a single or related process makes a lot of sense to save program development time. One of Proc SQL's strengths is its ability to easily remerge with itself summary level descriptive statistics. The technique presented in this paper is also applicable for adding any new variable to the dataset.

PROC SQL ADVANTAGES

By selecting all variables from the original data set and applying a LEFT JOIN, we are preserving the original dataset. In the subquery, an internal temporary dataset is created based on selecting both the linking variable and the new group descriptive statistics variable. The linking variable can uniquely identifies records or can be the grouping variable. A summary function is applied to the linking variable with the GROUP BY clause.

For example, if there were 10 males and 9 females, then the SEXN value would be 10 for all males and 9 for all females. The value of the linking variable also has a value for SEX, male or female, which is associated with the group count. Conditions can be applied as needed, for example, to exclude missing values. Note that by including NAME in the subquery, one record per NAME value is returned. The alternative is to not include the NAME variable and link by the grouping variable which returns only two records.

The advantages of applying subqueries within remerge techniques is that the original dataset does not change in data content or record sort order. Without subqueries, however, if the GROUP BY clause is directly applied with the wildcard '*', which selects all variables, the original dataset may not be preserved. In general, this approach should be avoided since it may incorrectly cause values to be summarized or records to be rearranged. Directly applying the GROUP BY clause to get group descriptive statistics may be more appropriate for selected limited number of variables in the original dataset instead of applying it to all variables with the wildcard '*'.

SAS TECHNICAL TECHNIQUES OUTLINE

Below is the outline to add group descriptive statistics as shown in Figure 1:

1. Select all master variables to preserve original dataset values and sort order.
2. Include the new group descriptive statistics variables in the outer SELECT.
3. Apply LEFT JOIN to keep all master records.
4. Create internal dataset from the same master outer dataset using a subquery.
5. Select or create grouping variable. Group variables are mostly categorical type variables which are generally character values. To create group descriptive statistics based on a continuous variable, first create a format or IF-THEN logic to group the values into several categories. For example, create a new agegrp variable to group ages.

6. Apply on or more summary functions to get group descriptive statistics on one or more numeric variables.
7. Link internal dataset from subquery to master dataset by grouping or key variable.
8. Create N and Percentages columns

```
proc sql;
  create table class2 as
  select
   a.*,                                  /* 1. Select  all master variables */
   b.sexn                                /* 2. Include the new group descriptive statistics variable */
  from sashelp.class as a
  left join                              /* 3. Apply LEFT JOIN to keep all master records */
  (select                                /* 4. Create internal dataset using subquery */
   sex,                                  /* 5. Select or create grouping variable */
   count(sex) as sexn                    /* 6. Apply one or more summary functions */
  from sashelp.class  where sex > ' '
  group by sex
  ) as b on
 a.sex=b.sex;                            /* 7. Link internal dataset by grouping variable */
quit;
```

Figure 1. PROC SQL LINKED BY GROUP VARIABLE

Because the subquery was linked by the grouping variable, the final dataset class2 is grouped by SEX.

	Name	Sex	Age	Height	Weight	sexn
1	Judy	F	14	64.3	90	9
2	Jane	F	12	59.8	84.5	9
3	Joyce	F	11	51.3	50.5	9
4	Barbara	F	13	65.3	98	9
5	Carol	F	14	62.8	102.5	9
6	Mary	F	15	66.5	112	9
7	Louise	F	12	56.3	77	9
8	Alice	F	13	56.5	84	9
9	Janet	F	15	62.5	112.5	9
10	Philip	M	16	72	150	10
11	James	M	12	57.3	83	10
12	Henry	M	14	63.5	102.5	10
13	John	M	12	59	99.5	10
14	William	M	15	66.5	112	10
15	Alfred	M	14	69	112.5	10
16	Jeffrey	M	13	62.5	84	10
17	Thomas	M	11	57.5	85	10
18	Ronald	M	15	67	133	10
19	Robert	M	12	64.8	128	10

Output 1. PROC SQL LINKED BY GROUP VARIABLE

SAS TECHNICAL TECHNIQUES STEPS

1. Select all master variables, ex. A.*
- Preserve original dataset values and sort order;

2a, 2b. Apply one or more summary functions (COUNT, MIN, MAX, etc.) to get group descriptive stats on one or more numeric variables with the same grouping variable saved as one or more variables, ex. SEXN. Practical application - DUPCNT

3. Left join to keep all master records, ex. LEFT JOIN

4. Subquery creates a temp dataset from the same outer master dataset, ex. ()
- SELECT limited variables
- Any detail level condition such as grouping variable non-missing
- Create new subquery for each new grouping variable

- Apply HAVING for summary level condition, ex. HAVING VISIT=MAX(VISIT)
- Does not have to be all inclusive of the linking variable
- Could be same or different dataset
- No semi-colon at the end

5a, 5b, 7.
Specify one or more grouping character variable (ex. SEX) or formatted numeric variable (ex. AGEGRP)
- Create user defined format for AGEGRP is needed

Two options for linking master and subquery datasets (1-to-M method may be more robust, both options have the same results)
- (One-to-Many) Join by summary variable (ex. SEX) which is many record per master dataset summary variable. Expect one record per unique summary variable.
- (One-to-One) Join by detail key variables (ex. NAME) which is one record per master dataset detail variable. Expect one record by unique detail variable.

Figure 2 is an example of a one-to-one join by key variables to get SEX counts. As you can see, the results are similar to figure 1 which was a one-to-many join by group variable. While in this example, the key variable is only one variable, there may be other cases it may be more than one variable. Linking by the grouping variable, however, offers you an advantage of linking by one variable.

```
proc sql;
  create table class3 as
  select
 1 - a.*,
 2a - b.sexn
  from sashelp.class as a
 3 - left join
 4  (select
 5a - name,
 2b - count(sex) as sexn
  from sashelp.class  where sex > ' '
 7 - group by sex
  ) as b on
 5b - a.name=b.name;
quit;
```

Figure 2. PROC SQL LINKED BY KEY VARIABLE

Because the subquery was linked by the key variable NAME, the final dataset class3 is sorted by NAME. Notice, however, that the group descriptive statistics are still the same as when the subquery was linked by SEX.

	Name	Sex	Age	Height	Weight	sexn
1	Alfred	M	14	69	112.5	10
2	Alice	F	13	56.5	84	9
3	Barbara	F	13	65.3	98	9
4	Carol	F	14	62.8	102.5	9
5	Henry	M	14	63.5	102.5	10
6	James	M	12	57.3	83	10
7	Jane	F	12	59.8	84.5	9
8	Janet	F	15	62.5	112.5	9
9	Jeffrey	M	13	62.5	84	10
10	John	M	12	59	99.5	10
11	Joyce	F	11	51.3	50.5	9
12	Judy	F	14	64.3	90	9
13	Louise	F	12	56.3	77	9
14	Mary	F	15	66.5	112	9
15	Philip	M	16	72	150	10
16	Robert	M	12	64.8	128	10
17	Ronald	M	15	67	133	10
18	Thomas	M	11	57.5	85	10
19	William	M	15	66.5	112	10

Output 2. PROC SQL LINKED BY KEY VARIABLE (Same results as Output 1. PROC SQL LINKED BY GROUP VARIABLE)

Figure 3 is an example of multiple group descriptive statistics joined by grouping variable. SEX counts, minimum and maximum WEIGHT by SEX are added to the dataset. Note that descriptive statistics can be based on variables other than the grouping variable. Note also that with this simple technique, descriptive statistics by other grouping variables can easily be added by repeating lines 3 to 8 and adding the new variable in line 2.

```
proc sql;
  create table class4 as
  select
 1 - a.*,
 2 - b.sexn2, b.minwgt, b.maxwgt
  from sashelp.class as a
 3 - left join
 4  (select
 5 - sex,
 6 - count(sex) as sexn2, min(weight) as minwgt, max(weight) as maxwgt
  from sashelp.class  where sex > ' '
 7 - group by sex
  ) as b on
 8 - a.sex=b.sex;
quit;
```

Figure 3. PROC SQL WITH MULTIPLE GROUP DESCRIPTIVE STATISICS VARIABLES

For each SEX group, the minimum and maximum weight by SEX is added as MINWGT and MAXWGT variables. As expected, the values are the same within each SEX group.

	Name	Sex	Age	Height	Weight	sexn2	minwgt	maxwgt
1	Judy	F	14	64.3	90	9	50.5	112.5
2	Jane	F	12	59.8	84.5	9	50.5	112.5
3	Joyce	F	11	51.3	50.5	9	50.5	112.5
4	Barbara	F	13	65.3	98	9	50.5	112.5
5	Carol	F	14	62.8	102.5	9	50.5	112.5
6	Mary	F	15	66.5	112	9	50.5	112.5
7	Louise	F	12	56.3	77	9	50.5	112.5
8	Alice	F	13	56.5	84	9	50.5	112.5
9	Janet	F	15	62.5	112.5	9	50.5	112.5
10	Philip	M	16	72	150	10	83	150
11	James	M	12	57.3	83	10	83	150
12	Henry	M	14	63.5	102.5	10	83	150
13	John	M	12	59	99.5	10	83	150
14	William	M	15	66.5	112	10	83	150
15	Alfred	M	14	69	112.5	10	83	150
16	Jeffrey	M	13	62.5	84	10	83	150
17	Thomas	M	11	57.5	85	10	83	150
18	Ronald	M	15	67	133	10	83	150
19	Robert	M	12	64.8	128	10	83	150

Output 3. PROC SQL WITH MULTIPLE GROUP DESCRIPTIVE STATISICS VARIABLES

Figure 4 is an example of numeric group descriptive statistics. The data step before Proc SQL creates the numeric grouping variable, AGEGRP based on AGE values. Proc SQL adds the counts for AGEGRP.

```
data class1;
 set sashelp.class;
 if age < 15 then agegrp = 'LT 15';
 else if age >= 15 then agegrp='GE 15';
run;

proc sql;
  create table class5 as
  select
 1 - a.*,
 2 - b.agen
  from class1 as a
 3 - left join
 4  (select
```

```
 5 - agegrp,
 6 - count(age) as agen
   from class1  where age > .
 7 - group by agegrp
   ) as b on
 8 - a.agegrp=b.agegrp;
quit;
```

Figure 4. PROC SQL WITH NUMERIC GROUP DESCRIPTIVE STATISICS VARIABLE

For continuous variables such as AGE, once a user defined format is applied to group into categories of 2 for example, the group counts AGEN can be added. In the example below, we see that there are 5 people with ages greater than or equal to 15 and 14 people with ages less than 15.

	Name	Sex	Age	Height	Weight	agegrp	agen
1	Ronald	M	15	67	133	GE 15	5
2	Philip	M	16	72	150	GE 15	5
3	Mary	F	15	66.5	112	GE 15	5
4	William	M	15	66.5	112	GE 15	5
5	Janet	F	15	62.5	112.5	GE 15	5
6	Jane	F	12	59.8	84.5	LT 15	14
7	James	M	12	57.3	83	LT 15	14
8	Judy	F	14	64.3	90	LT 15	14
9	Barbara	F	13	65.3	98	LT 15	14
10	Louise	F	12	56.3	77	LT 15	14
11	Joyce	F	11	51.3	50.5	LT 15	14
12	Henry	M	14	63.5	102.5	LT 15	14
13	John	M	12	59	99.5	LT 15	14
14	Jeffrey	M	13	62.5	84	LT 15	14
15	Thomas	M	11	57.5	85	LT 15	14
16	Alfred	M	14	69	112.5	LT 15	14
17	Alice	F	13	56.5	84	LT 15	14
18	Carol	F	14	62.8	102.5	LT 15	14
19	Robert	M	12	64.8	128	LT 15	14

Output 4. PROC SQL WITH NUMERIC GROUP DESCRIPTIVE STATISICS VARIABLE

Figure 5 is an example of group descriptive statistics based on another variable. Since grouping and descriptive statistics are related yet independent of each other. Proc SQL enables you to group by one variable such as PRODUCT and calculate descriptive statistics based on another variable such as SALES. In this typical business example, we want SALES descriptive statistics, count, minimum and maximum by each PRODUCT. Notice that the linking variable is the PRODUCT grouping variable.

```
proc sql;
  create table class6 as
  select
 1 - a.*,
 2 - b.salesn, b.salesmn, b.salesmx
   from sashelp.shoes as a
 3 - left join
 4  (select
 5 - product,
 6 - count(sales) as salesn, min(sales) as salesmn, max(sales) as salesmx
   from sashelp.shoes  where sales > .
 7 - group by product
   ) as b on
 8 - a.product=b.product;
quit;
```

Figure 5. PROC SQL WITH GROUP DESCRIPTIVE STATISICS BASED ON ANOTHER VARIABLE

Below, you can see the number of SALES records, SALESN, minimum SALES, SALESMN and maximum SALES, SALESMX for each PRODUCT. As expected, the values are the same, for example, within the Boot PRODUCT. There is a change in SALESN, SALESMN and SALESMX values for Men's Casual PRODUCT. While creating these variables, formats can be applied to add '$' before the SALESMN and SALESMX values.

	Region	Product	Subsidia ry	Stores	Sales	Inventor y	Returns	salesn	salesmn	salesmx
39	Middle East	Boot	Dubai	15	$90,972	$403,259	$4,049	52	1179	286497
40	South America	Boot	Bogota	19	$15,312	$35,805	$1,229	52	1179	286497
41	United States	Boot	Los Angeles	14	$85,932	$347,252	$3,283	52	1179	286497
42	Middle East	Boot	Al-Khobar	10	$15,062	$44,658	$765	52	1179	286497
43	United States	Boot	Seattle	15	$70,790	$226,678	$3,039	52	1179	286497
44	Western Europe	Boot	Geneva	22	$41,341	$171,030	$1,560	52	1179	286497
45	Western Europe	Boot	Heidelberg	14	$65,610	$301,779	$2,829	52	1179	286497
46	Western Europe	Boot	Lisbon	18	$76,349	$341,911	$3,290	52	1179	286497
47	United States	Boot	New York	18	$97,151	$495,479	$3,983	52	1179	286497
48	Western Europe	Boot	London	31	$54,449	$289,527	$2,321	52	1179	286497
49	Western Europe	Boot	Rome	23	$36,244	$209,271	$1,455	52	1179	286497
50	Western Europe	Boot	Paris	15	$19,196	$41,506	$1,006	52	1179	286497
51	Western Europe	Boot	Madrid	1	$1,179	$1,027	$93	52	1179	286497
52	Western Europe	Boot	Copenhagen	2	$1,663	$4,657	$129	52	1179	286497
53	Western Europe	Men's Casual	Lisbon	18	$311,341	$502,959	$12,247	45	9244	1298717
54	Western Europe	Men's Casual	London	13	$151,402	$401,374	$7,159	45	9244	1298717
55	Western Europe	Men's Casual	Copenhagen	1	$38,492	$90,672	$767	45	9244	1298717
56	Western Europe	Men's Casual	Heidelberg	12	$175,694	$595,118	$9,845	45	9244	1298717
57	United States	Men's Casual	New York	20	$456,985	$890,096	$13,453	45	9244	1298717
58	United States	Men's Casual	Seattle	6	$65,842	$112,538	$3,395	45	9244	1298717
59	Western Europe	Men's Casual	Geneva	4	$71,618	$138,897	$2,484	45	9244	1298717
60	Western Europe	Men's Casual	Rome	2	$37,271	$91,670	$1,761	45	9244	1298717
61	Western Europe	Men's Casual	Madrid	4	$71,014	$59,011	$3,784	45	9244	1298717

Output 5. PROC SQL WITH GROUP DESCRIPTIVE STATISICS BASED ON ANOTHER VARIABLE

Figure 6 is an example of adding N and percent columns. This is ideal to qc AE tables.

```
* Checklist of all values, most useful to show zeros for no cases in checklist, do difference with AE terms;
proc sort data=event nodupkey out=event_list (keep=aeterm rename=(aeterm=name));
 by aeterm;
run;


proc sort data=event;
 by pt aeterm;
run;

data master1;
 retain grdtot 0;
 set event;
 by pt aeterm;
 if first.pt then grdtot=grdtot + 1;
 if first.aeterm and aeterm > '' then denom=1;
run;

proc sql;
 select count(distinct(pt)) into :grdtot from pats;

 create table master2 as
 select unique a.name, b.aeterm, c.pd_catn, c.subjectn
 , left(put(c.pd_catn, 4.) || ' (' || strip(put(c.subjectn, 4.)) || ')')  as eventlbl1

 , left(put((c.subjectn/&grdtot*100), 4.2) || '% (' || strip(put(c.subjectn, 4.)) || '/' || strip(put(&grdtot, 4.)) || ')') as eventlbl2
 , '0 (0)' as eventlbl3
 , '0% (0/' || strip(put(&grdtot, 4.)) || ')' as eventlbl4
 from event_list as a
 left join master1 as b on a.name=b.aeterm
 left join
  (select unique pt, aeterm, count(aeterm) as pd_catn, count(denom) as subjectn
   from master1 group by aeterm
  ) as c on b.pt=c.pt and b.aeterm=c.aeterm ;
quit;

data master3;
 set master2;
 varlab=name;
```

```
 if aeterm = '' then do;
  eventlbl1=eventlbl3;
  eventlbl2=eventlbl4;
 end;
 label varlab = 'AE Term'
       eventlbl1 = 'Total Events (Patients)' eventlbl2 = '% (n/N)';
  if varlab > '';
 keep varlab eventlbl1 eventlbl2 pd_catn subjectn;
run;

ods listing;
proc print data=master3;
 var varlab eventlbl1 eventlbl2;
run;
```

Figure 6. PROC SQL WITH N and PERCENT CALCULATIONS

Below, you can see N and % columns,

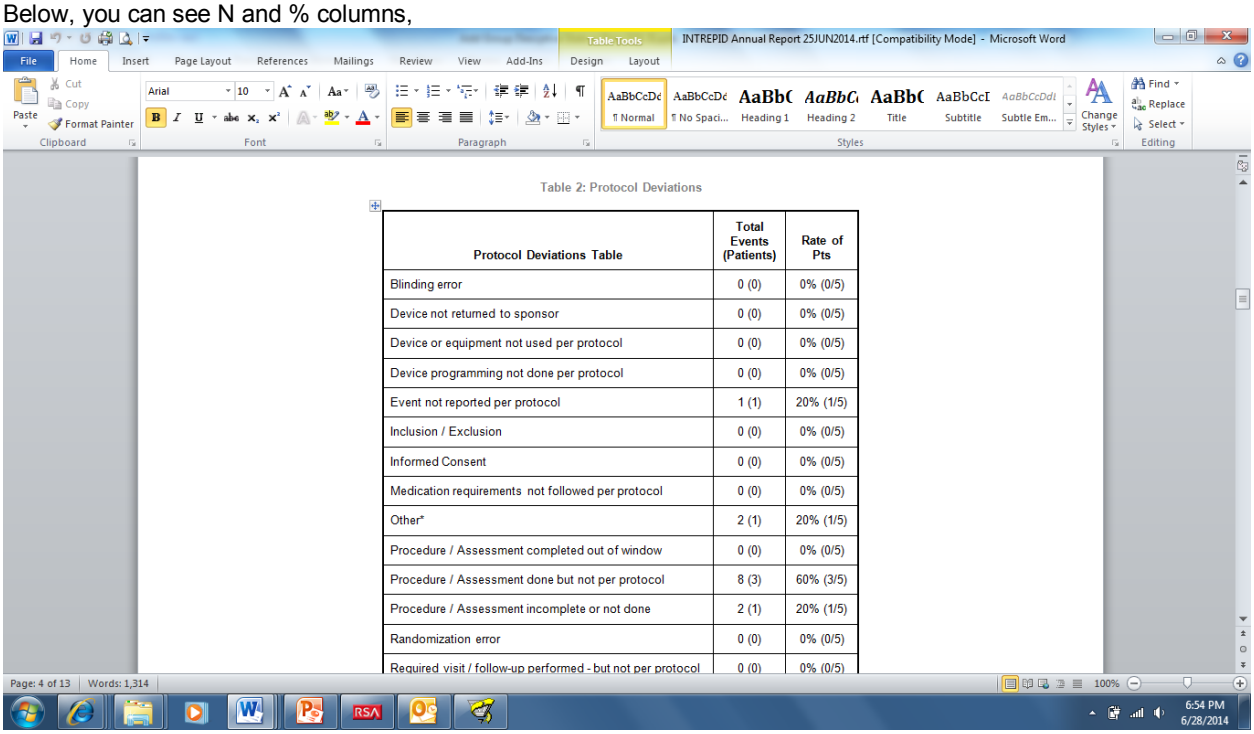
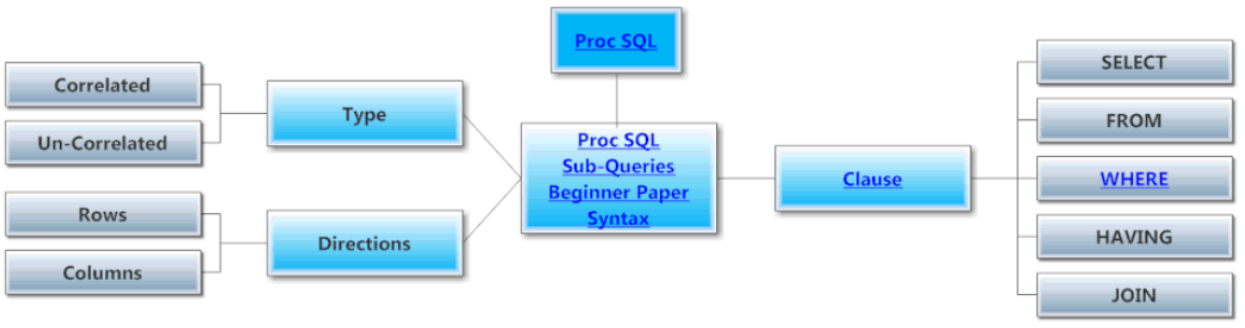


Table 2: Protocol Deviations

Protocol Deviations Table	Total Events (Patients)	Rate of Pts
Blinding error	0 (0)	0% (0/5)
Device not returned to sponsor	0 (0)	0% (0/5)
Device or equipment not used per protocol	0 (0)	0% (0/5)
Device programming not done per protocol	0 (0)	0% (0/5)
Event not reported per protocol	1 (1)	20% (1/5)
Inclusion / Exclusion	0 (0)	0% (0/5)
Informed Consent	0 (0)	0% (0/5)
Medication requirements not followed per protocol	0 (0)	0% (0/5)
Other*	2 (1)	20% (1/5)
Procedure / Assessment completed out of window	0 (0)	0% (0/5)
Procedure / Assessment done but not per protocol	8 (3)	60% (3/5)
Procedure / Assessment incomplete or not done	2 (1)	20% (1/5)
Randomization error	0 (0)	0% (0/5)
Required visit / follow-up performed - but not per protocol	0 (0)	0% (0/5)

Output 6. PROC SQL WITH N and PERCENT CALCULATIONS

For a visual representation of Proc SQL's suq-queries, see the mind map below. This mind map is part of a larger collection of interrelated mind maps. For this and other free SAS mind maps, visit www.SASSavvy.com.



SUMMARY

As you can see, with a few simple techniques combined together, a Proc SQL template can be used on a regular basis to always easily add group descriptive statistics to your dataset. This technique can be duplicated for multiple group descriptive statistics. Without the need for a second DATA step or creating a second dataset, you can now quickly add group descriptive statistics and move forward to the next by-group processing step.

I hope you now have some insights into how and why Proc SQL is one of the best ways to accomplish this task. Based on when I first developed this technique, I realized that I could expand on it to build a more ultimate Proc SQL example with the HAVING, GROUP BY, and the WHERE clauses along with the subquery as shown in figure 6. Note that caution must be applied to prevent the assumption that the HAVING clause impacts the GROUP BY clause when calculating mean values. The HAVING clause is applied to subset the dataset only AFTER calculating the mean.

```
proc sql;
create table base2 as                    /* Add suffix number to base dataset */
select unique a.*, b.aval, b.dtype       /* Keep all A.* & new B.vars */
from base (drop=aval) as a               /* Drop AVAL since replaced */
right join                               /* Save AVERAGE records only */
(                                        /* Subquery start */
select unique usubjid, paramn, ady,      /* Keep linking variables */
 mean(aval) as aval,                     /* Summarize to get one rec/key */
 'AVERAGE' as dtype                      /* Flag data type method */
 from base                               /* FROM same dataset as outer */
 where ady <= 1                          /* Subset only baseline records */
 group by usubjid, paramn                /* GROUP BY key variables for MEAN() */
 having max(ady) = ady                   /* Of the records, keep only max ADY */
)                                        /* Subquery stop */
as b on a.usubjid=b.usubjid and a.paramn=b.paramn and a.ady=b.ady
                                         /* Linking variables with outer dataset */
;
quit;
```

Figure 6. EXAMPLE OF ULTIMATE PROC SQL CODE

Note that with repeated subqueries to update an existing variable already in the master dataset, it is recommended to save results to a new variable such as B.XXX instead of using COALESC() function to keep the non-missing value. After the PROC SQL step, use DATA step to replace the original variable if the B.XXX variable is non-missing. This technique takes advantage of PROC SQL to do the major task and DATA step to adjust the variable.

```
proc sql;
create table base2 as                    /* Add suffix number to base dataset */
select unique a.*, b.b_aval, b.b_dtype   /* Keep all A.* & new B.vars */
from base as a                           /* Drop AVAL since replaced */
left join                                /* Save AVERAGE records only */
```

```
(                                                   /* Subquery start */
select unique usubjid, paramn, ady,                 /* Keep linking variables */
 mean(aval) as b_aval,                              /* Summarize to get one rec/key */
 'AVERAGE' as b_dtype                               /* Flag data type method */
 from base                                          /* FROM same dataset as outer */
 where ady <= 1                                     /* Subset only baseline records */
 group by usubjid, paramn                           /* GROUP BY key variables for MEAN() */
 having max(ady) = ady                              /* Of the records, keep only max ADY */
)                                                   /* Subquery stop */
as b on a.usubjid=b.usubjid and a.paramn=b.paramn and a.ady=b.ady
                                                    /* Linking variables with outer dataset */
;
quit;

data base2;
 set base2;

 if b_aval > . then aval=b_aval;
 if b_dtype >  ' ' then dtype=b_dtype;
run;
```

Figure 7. EXAMPLE OF ULTIMATE PROC SQL CODE SAVING TO NEW VARIABLES

REFERENCES

Gupta, Sunil, "Ready To Become Really Productive Using PROC SQL?", PharmaSUG 2012, SAS Global Forum 2011 and WUSS 2010

Gupta, Sunil, Something for Nothing? Adding Group Descriptive Statistics using Proc SQL's Subqueries, June 2012

http://blogs.sas.com/content/sgf/2012/06/20/adding-group-descriptive-statistics/

CONTACT INFORMATION

The author welcomes your comments and suggestions.
Sunil K. Gupta
Associate Director, Statistical Programing
Cytel
213 Goldenwood Circle
Simi Valley, CA 93065
Phone: (805)-577-8877 E-mail: Sunil.Gupta@Cytel.com
www.SASSavvy.com

Sunil Gupta is a best selling SAS author and global corporate trainer. Sunil is the Associate Director at Cytel. Most recently, Sunil launched www.SASSavvy.com and released five new SAS e-Guides on Quick Results with PROC SQL, Quick Results with PROC REPORT, A to Z Analysis and Validation using PROC TABULATE, Compare and Conquer SAS Programming Techniques and Automating Tasks using SAS Macro Programming. 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*, and *Sharpening Your SAS Skills.*