Example of Popular Quick Results with PROC SQL E-Guide

(Purchase the full version now at SASSavvy.com)

The single most concise productivity e-Guide available!

Sunil@SASSavvy.com

Download all PROC SQL Examples
E-mail Sunil Your PROC SQL Questions

Basic Usage of Proc SQL

Common examples for:
- Creating Tables or Views
- Changing Table Structure
- Updating, Adding, or Deleting Data

I. Table Access and Retrieval

*Column Definition, *Join Tables,
*Subset Table, *Sort Table

II. Creating Macro Variables

III. Table Structure Operations

IV. Table Content Operations

V. Connecting to Relational Databases

The SAS e-Guide assumes you have general knowledge of SAS Programming. Common features are illustrated through task-based examples. Web browser may need to be open for links.

PROC SQL Basic Usage:

```
1. proc sql;                                    /* required */
2.   < proc sql statement 1 > ;
3.   < proc sql statement 2 > ;
4. quit;                                        /* required */
```

Lines 2, 3: all of the examples in the e-Guide are expected to be embedded between lines 1 and 4. SAS accepts one or more PROC SQL statements within each PROC SQL block of code.

Eight common benefits for using PROC SQL:

```
1. proc sql;                                    /* required */
2.   create table mytable as select ...;       /* new table */
3.   create view myview as select ...;         /* new view */
4.   select ...;                                /* powerful query tool */
5.   alter table mytable ...;                   /* new table structure */
6.   update table mytable ...;                  /* update table content */
7.   insert table mytable ...;                  /* update table content */
8    delete from table mytable ...;             /* delete table content */
9.   drop table mytable;                        /* delete table */
10. quit;                                       /* required */
```

Line 1: six options exist to help debug – NOEXEC (to check syntax without executing the code), FEEDBACK (displays code executed), INOBS=# of rows read, OUTOBS=# of rows written, _METHOD (displays PROC SQL execution options), _TREE (display visual structure of logic). See SAS On-Line reference for complete set of features and options. See SAS paper on undocumented features. Remember to set OPTIONS MSGLEVEL=I for _METHOD and _TREE options since default is MSGLEVEL=N. After the syntax is error-free, you may need to investigate your data and logic. Note that once debug options are applied, the RESET NOFEEDBACK is required, for example, to reset to default. NUMBER option displays the row number which can be also saved as a new column with the MONOTONIC() keyword. See SAS paper on the MONOTONIC() keyword. Can add NOWARN option to prevent displaying warning messages. Finally, the FLOW option is useful to wrap text within each cell.

Line 2: create mytable table with SELECT statement. If a table already exists, then LIKE, instead of AS, <table> option can be used to copy the table structure with zero rows.

Line 3: create myview view with SELECT statement. Views are similar to tables except that views do not require space and can not be indexed. Once created, it is useful to DESCRIBE view to display the table structure, similar to PROC CONTENTS, to the SAS log.

Line 4: query table with SELECT statement. Add VALIDATE before SELECT to check syntax without executing the code. Note that while generally PROC SQL is used for data management tasks, the results of SELECT statements may be directed to Excel, RTF, PDF or HTML files through ODS. See SAS site for overview of SELECT statement. Also note that PROC SQL stores the # of rows returned in the &SQLOBS automatic macro variable which can be used for further macro processing. See automatic macro variable section. See SAS site for more info on automatic macro variables.

Line 5: change MYTABLE table structure by copying another table, creating columns with attributes, dropping columns, modifying column attributes or adding integrity constraints. See III Table Structure Operations.

Line 6: update MYTABLE table content by multiplying by number or string modification. Changes can be conditionally applied with the WHERE clause. See IV Table Content Operations.

Line 7: add data content to MYTABLE table as series of values for all columns, one column or data from another table. See IV Table Content Operations.

Line 8: delete table content from MYTABLE table based matching condition from WHERE clause.
Line 9: delete table MYTABLE. See IV Table Content Operations.
Line 10: QUIT is required.

I. Table Access and Retrieval

Creating SAS tables enable quick access to the data. Options for data access and retrieval include selecting columns, joining tables, subsetting table, and sorting tables.

Basic Lines of Code for Table Access and Retrieval – Six Components:

```
1. create table mytable as
2. select name, sex                /* select columns */
3. from sashelp.class              /* source table */
4. where sex = 'F'                 /* option to subset table */
5. group by sex                    /* option to group by */
6. having weight > avg(weight)     /* option to include having */
7. order by name                   /* option to sort table */
8. ;                               /* required */
```

Line 1: create table MYTABLE. As in the DATA Step, you can also specify dataset options such as DROP= or KEEP=. Note that all six PROC SQL components are contained within one statement and must be specified in this order: 1) selecting, 2) source table, 3) row subset condition, 4) group by, 5) group subset condition, 6) sort order. Four additional dimensions of PROC SQL over the DATA Step include: 1) descriptive statistics using summary functions along with COALESCE(), 2) group by columns, 3) conditional processing, and 4) row and group level subsetting along with subqueries.
Line 2: select columns. **See A. Column Definition.**
Line 3: from source table. **See B. Join Tables.**
Line 4: apply condition for subsetting table. **See C. Subset Table.**
Line 5: can group by column.
Line 6: can group select records based on summary function.
Line 7: sort result table. **See D. Sort Table.**
Line 8: ';' required to end PROC SQL statement.

A. Column Definition

When defining columns, you can select columns that already exist in the table or create new columns. When creating new columns, remember to specify column attributes such as label, format and length. Finally, when selecting or creating columns to be saved as a table, the column order defines the order stored in the table. See SAS's web site for more info on SELECT statements.

Four Options for Selecting Columns: Select Clause

```
1. select name, sex
2. select name label = "My Label" format = $10. length = 10
3. select *
4. select distinct sex              /* alternative to use unique */
```

Line 1: list selected columns, separate each column by comma ',' and in order of output.
Line 2: after the column name, to define column attributes such as label, format, and length. Note that LENGTH option for character columns is still without '$' and that CHAR is required in the CREATE TABLE statement. LENGTH is always equal to a number. **See III. Table Structure Operations.**
Line 3: use wildcard '*' to select all columns from the table.

Comparing PROC SQL with DATA Step

DATA Step	PROC SQL
Data set, observations, variables	Tables, rows, columns
SAS Functions, Data set options	SAS Functions, COALESCE(), Data set options
If-Then Statements	Case-Select Clause
Do Loop, Output	Joins can simulate Do Loop
Space to separate variables	Comma ‘,’ to separate variables
New variable = valid expression;	Valid expression AS new variable
IF/Where Statements	Where for details/Having for summaries
Multiple SAS Statements	One SAS Statement
By default, includes all variables	By default, excludes all variables
Many-to-many merge	Cartesian Product is better
By default, If A or B; Full Outer	By default, If A and B; Inner Join
If A; If B; If A and B; If A or B; If A not B;	Joins: Left, Right, Inner, Full, Except
Set A B;	Sets: Outer Union Corr
Can recycle data set and variable names	Requires new data set and column names
NA	Unique PROC SQL keywords

For more information see SAS tips paper. See SAS Blog on topic. See SAS paper for DATA Step Die-hards. See SAS paper on top 10 reasons for using PROC SQL. See SAS paper on comparing program efficiency. See SAS paper on why PROC SQL is a must know skill.

Comparing PROC SQL with SAS Procedures

PROC FREQ, PROC MEANS, PROC PRINT, PROC SORT, PROC DATASETS	PROC SQL
Proc Freq data=X; tables sex/list; run;	Proc sql; Select distinct sex from X; quit;
Proc Means data=X mean min; var weight; run;	Proc sql; Select mean(weight), min(weight) from X; quit;
Proc Print data=X; var name sex; run;	Proc sql; Select name, sex from X; quit;
Proc Sort data=X; by name; run;	Proc sql; Select * from X order by name; quit;
Proc Datasets; delete X; run;	Proc sql; Drop X; quit;

See SAS paper on comparing PROC SQL with other SAS Procedures.

Efficiency Gains with PROC SQL

Task	Efficiency Gains
Efficiency Categories:	CPU Time, Memory, I/O, Data Storage, Programming Time
Sort large unsorted datasets	Combination of Proc Sort and Data Step
Using presorted large datasets when joining datasets by uncommon variables	(SORTEDBY=) option takes advantage of presorted datasets when joining datasets. Instead of remaining the variables, PROC SQL can join by different variable names.
Simple Index	Logical reference without physically sorting – best if

	WHERE results in 25% fewer records

General PROC SQL Usage

```
Proc sql;

create table
mylib.newclass as

select name, sex label = 'Sex'

from
sashelp.class


where sex = 'F'



order by name;

quit;
```

When defining columns, you can select columns that already exist in the tables or create new columns. Expressions and summary functions could also be used. Column attributes can also be defined.

Alias can be used to reference tables. Joining tables is easy to accomplish with PROC SQL. When joining tables, options include inner or outer joins. Inner joins return a table containing rows that match both tables. Outer joins return a table containing rows that match both tables plus all nonmatching rows form the left, the right, or both tables.

When subsetting tables using existing or new columns, you can subset by rows or by groups. The difference is in the selection condition. Subsetting by row compares rows to an expression and subsetting by group compares rows to a group expression which generally contains a summary function.

Subsetting the table using subqueries offers greater power and flexibility in pre-processing a table to dynamically specify the subset condition.

When sorting tables, you can sort by rows or by groups. Sorting by group requires the group by clause.

Key SAS books and references for more information

PROC SQL: beyond the basics using SAS
The Essential PROC SQL Handbook for SAS Users
PROC SQL by Example: Using SQL within SAS

Webcast: Proc SQL Tips and Techniques, Dictionary Tables

Sunil's top 15 recommended Proc SQL Papers
Get involved, e-mail me to consider your favorite SAS paper or link.