

# PROC SQL FOR THOSE WHO HAVE BEEN ESCHEWING SQL

Benjy Ben-Baruch  
Mgr, Business Intelligence and Analytics  
Harry and David  
Meford OR

[bben-baruch@harryanddavid.com](mailto:bben-baruch@harryanddavid.com) or [benjbenb@umich.edu](mailto:benjbenb@umich.edu)

MSUG Meeting  
October 20, 2011

# Analysts should add PROC SQL to their tool kit and skill set

- Ian Whitlock<sup>[1]</sup> maintains PROC SQL must be a staple for all SAS programmers.
  - Indeed!
  - It is also very useful for those of us concentrating our efforts on analysis.
- This presentation is meant to convince analysts that it is worth learning SQL to enhance your SAS skills.
- If you are an analyst still procrastinating – or resisting – learning how to use PROC SQL with SAS, then this presentation is for you.

[1] <http://www2.sas.com/proceedings/sugi26/p060-26.pdf>

PROC SQL in SAS has many advantages for analysts as a supplement to our coding skills.

- Data can be joined/merged w/o pre-sorting and w/o indices.
- Variables can be created on the fly,
- Datasets being merged/joined do not have to have the same variable names.
- Using the OVER PARTITION BY syntax, multiple sorts can be done “on the fly”
- Renaming variables is not necessary – but, if desired, is very fast and easy
- Reformatting variables can be done on the fly. SAS format statements can be used -- if not using pass-through SQL.

SQL cannot do everything. (Neither can SAS – or any other tool.)

- SQL has nothing that has the power of PROC FORMAT.
- SQL does not have the analytic power of SAS
- PROC SQL is not an alternative to “regular” SAS – it is an additional programming tool that can often make our coding easier, simpler, and more efficient

# Two kinds of SQL in SAS

- The SAS PROC SQL statement
  - Uses SQL syntax
  - Like all SQL syntax – and like SAS – it has its “quirks”. SQL syntax is very similar across all systems, but each system has its own quirks. (And, of course, these are not always well documented.)
  - You can use PROC SQL on SAS datasets as an alternative to a DATA STEP or some SAS procedures.
  - You can use SAS format statements and SAS functions

# Two kinds of SQL in SAS

- Pass-Through” SQL

- SAS/Access allows us to connect to databases and other systems that use SQL (e.g. Oracle, DB2, SQL Server).
- All tables referenced must be in the SQL database. (Table being created DOES NOT have to be in the SQL database. It can be in any directory/folder/path to which SAS can write datasets.)
- All code and syntax (within the pass-through code) must be in the syntax of the system/software/platform to which code is being passed-through.
- If being passed through to a SQL database, metadata system files can be accessed even w/o a SAS metadata engine. These tables with metadata information are accessed like other tables in the database. (Of course, you have to have permission to read these metadata tables.)

# Basic SQL syntax

- PROC SQL

```
PROC SQL;
```

```
CREATE TABLE tablename AS
```

```
SELECT A.var1, A.var2, B.var1, B.var2
```

```
FROM File1 A , File2 B
```

```
WHERE A.var4=B.var5 ; QUIT;
```

- Pass-Through SQL

```
PROC SQL;
```

```
CONNECT TO SqlSrvr (datasrc=remoteserver schema=schema  
userid=userid pwd=password STRINGDATES=NO);
```

```
CREATE TABLE tablename AS
```

```
SELECT * FROM CONNECTION TO SqlSrvr
```

```
( SELECT A.var1, A.var2, B.var1, B.var2
```

```
FROM File1 A , File2 B
```

```
WHERE A.var4=B.var5
```

```
); DISCONNECT from SqlSrvr;
```

# Notes on Pass-Through SQL

- The CONNECT statement uses system specific syntax similar to that used in the LIBNAME statement.
- Pass-Through SQL can create a SAS dataset or a table in the remote database. If creating a table in the remote database, one may also use the native SQL syntax to create a new table (e.g. INSERT INTO).

The Library name appended to the tablename will determine where the dataset/table is written:

- “CREATE TABLE saslibrary.dataset AS” creates a SAS dataset
- “CREATE TABLE schema.table AS” creates a table on the remote database

Data can be joined/merged w/o pre-sorting and w/o indices  
Datasets being merged/joined do not have to have the  
same variable names.

```
PROC SQL;  
CREATE TABLE newtable AS  
SELECT A.* B.*  
FROM SqlSrvr.table1 A INNER JOIN DATA.dataset1 B  
ON A.personid=B.customerid  
ORDER BY A.zipcode, B.lastname  
; QUIT;
```

- No presorting is necessary! Variable names do not have to be the same!  
(Actually, if they are the same the SQL code becomes more cumbersome because you cannot bring in variables with the same name from more than one table.)
- “A” and “B” are “aliases” for table1 and dataset1.
- All variables from both data sets are brought in to NEWTABLE. “A.\*” and “B.\*” means all variables from the table/dataset designated as “A” and “B” respectively.
- Output dataset can be sorted by a different variable.

Renaming variables is not necessary – but, if desired, is very fast and easy

```
PROC SQL;  
CREATE TABLE newtable AS  
SELECT A.zipcode, A.personid as CustomerID, B.CustomerID as  
    HshldID, B.lastname, B.var1 as gender  
FROM SqlSrvr.table1 A INNER JOIN DATA.dataset1 B  
ON A.personid=B.customerid  
ORDER BY A.zipcode, B.lastname  
; QUIT;
```

Reformatting variables can be done on the fly. SAS format statements can be used -- if not using pass-through SQL.

```
PROC SQL;  
CREATE TABLE newtable AS  
SELECT DATEPART(datetimevariable) as Date,  
       ROI format=DOLLAR6.2,  
       Population format=COMMA12.,  
       StateFIP as ST format=2.  
FROM oldtable;  
QUIT;
```

# Specific statistics can be quickly printed

```
PROC SQL;
```

```
    SELECT COUNT(*) as N, COUNT(DISTINCT custIDs) as N_customers,  
    COUNT(DISTINCT hshlds) as N_hhs, MEAN(spend) as MeanSpend  
    format=dollar8.2, SUM(spend) as TotSpend format=dollar8.2, zipcode  
    FROM newtable  
    GROUP BY zipcode
```

```
; QUIT;
```

- **Output:**
  - N=Total number of observations
  - N\_customers = the number of unique customer IDs
  - N\_hhs = number of unique households
  - MeanSpend is the mean amount spent
  - TotSpend is the total amount spent
- And all of this is aggregated at the zipcode level
- No new dataset or table is create
- The output lacks presentation quality formatting – but it is a very quick way to look at specific statistics while working interactively – especially when one wants to compare total observations to the number of unique values

# NOTES AND COMMENTS

- Regular SAS code cannot be mixed with SQL code in PROC SQL.
- However, in “regular” SAS PROC SQL, SAS functions and certain SAS syntax can be used. This is consistent with the inconsistencies in SQL, or, rather, with the system and engine/software specific nature of SQL
- Care has to be taken working with missing values. Blanks and nulls are different in relational databases and math calculations don't always work the way one expects.
- Care and caution needs to be exercised when dealing with time and date data, and especially datetime variables that cross between systems. In the more recent versions of SAS, the STRINGDATES option is helpful. The SAS DATEPART function is also very useful as is the SQL CAST function.

# NOTES AND COMMENTS

## Some differences between approaches to data

- SQL cannot do everything. (Neither can SAS.) But PROC SQL can be a very useful addition to an analyst's tool-kit.
- SQL does not have the analytic power of SAS. (One indication of this is verbiage. IT and SQL people refer to variables as "columns" in a table – or, sometimes, as fields. To them, these are data to be processed or numbers to be created "on the fly", not variables to be saved in a table and analyzed.
- Analyzing the variation between relationships is a foreign concept. Change is something to be "tracked" and "reported", not used as input data for the beginning of an analysis. Change is to be reported, not analyzed.

# SQL supplements but does not replace traditional SAS syntax

- Analysts have to have a high tolerance for ambiguity and little tolerance for error. IT folk may have a high tolerance for error but they have no tolerance for ambiguity. They live in a binary world. Ambiguity is intolerable. For analysts, ambiguity is normal. Variations of error terms are normal and figuring out how close their variation is to a normal distribution is standard practice. That's why we look at standard errors.
- Anyone maintaining that SQL can do virtually all of the analysis and statistics that SAS can do obviously does not understand statistics or the nature of real analysis. **But this is not a reason to eschew learning how to use SQL in SAS to your advantage.**

# Helpful References

Helen Carey, Carey Consulting, Kaneohe, HI and Ginger Carey, Carey Consulting, Kaneohe, HI *Tips and Techniques for the SAS® Programmer*. SAS Global Forum Paper 272-2011, <http://support.sas.com/resources/papers/proceedings11/272-2011.pdf>

Kirk Paul Lafler, “Undocumented and Hard to Find Proc Sql Features”, [http://www.sas.com/offices/NA/canada/downloads/presentations/Vancouver\\_spring2008/Undocumented.pdf](http://www.sas.com/offices/NA/canada/downloads/presentations/Vancouver_spring2008/Undocumented.pdf)

Ian Whitlock, “PROC SQL - Is it a Required Tool for Good SAS Programming?”, <http://www2.sas.com/proceedings/sugi26/p060-26.pdf>

Gajanan Bhat and Raj Suligavi, “Merging Tables in DATA Step vs. PROC SQL: Convenience and Efficiency Issues”, <http://www2.sas.com/proceedings/sugi26/p104-26.pdf>

Ying Feng, The SQL Procedure: When and How to Use It?, <http://www2.sas.com/proceedings/sugi31/044-31.pdf>