Using the SQL Procedure

Kirk Paul Lafler
Software Intelligence Corporation

Abstract

The SQL procedure follows most of the guidelines established by the American National Standards Institute (ANSI). In its current form, PROC SQL provides many features available in the DATA step and the MEANS, PRINT, and SORT procedures. An advantage of using PROC SQL is that it can often result in fewer and shorter statements than using existing DATA step and procedure methods. This hands-on workshop illustrates numerous examples of how PROC SQL and its many statements can be used. In particular, participants will learn how to create and modify tables and views, retrieve data using the SELECT statement, and perform efficient queries against SAS System data sets.

Prerequisites

This workshop introduces the syntax and uses of the SQL Procedure. No prior knowledge of SQL is required since this material is designed to acquaint the user and/or programmer to the many features found within the SQL Procedure. You should have a working knowledge of the following concepts before enrolling in this workshop:

- How to use the SAS System on your host computer system.
- How to interact with the SAS System in your environment.
- How to create and process SAS System data sets under Version 6.06.
- How to use the SAS System Display Manager System windows and text editor.
- How to assign and reference Librefs and Filerefs in your environment.
- What the rows and columns represent in a SAS System data set.

Workshop Objectives

After completion of this workshop, you should have a working knowledge of the following concepts:

- Basic SQL concepts.
- Specific SQL syntax requirements.
- How to construct queries to retrieve, sum, group, and sort data.
- How to create and modify simple tables and views.
- How to Rearrange data using the ORDER BY clause.

Brief History of Structured Query Language (SQL)

Structured Query Language (SQL) has an interesting history: It is a universal language that was developed to easily access data that is stored in relational databases or tables. A relation is represented as a two-dimensional table cousisting of rows and columns. A series of guidelines were developed by Dr. E. F. Codd, an IBM mathematician, through the use of relational mathematics. SQL evolved over time to access data regardless of the application software (e.g., COBOL, PL/1, etc.) being used.

Basic SQL Concepts

SQL boasts the ability to define, manipulate, and control relational databases or tables as well as providing easy user access. The concept behind SQL is that the user does not have to specify physical attributes about the data such as data structure, location, and/or data type. The user will concentrate on what data should be selected, but not how to select them. It also provides methods of making changes to tables without the need to change application programs. Consequently, data independence is one of the design goals of the relational model.

The SQL Procedure in the Base SAS System

PROC SQL has the following features:

- Can run interactively as well as batch (noninteractive).

- Uses the Structured Query Language to create, modify, and retrieve data from tables.
- Can be augmented through the use of Global statements such as TITLE and FOOTNOTE.
- Accesses tables via a two-level name where the first level is the Libref and the second level name is the name of the table.

Terminology

The following terminology is provided to help relate SQL and SAS System terms and concepts.

Column -- the same as a variable in the SAS System.

Libref -- acts as the alias or nickname. Points to a SAS data library.

Relational Database Management System -- a database system that forms relationships between data items.

Row -- the same as an observation in the SAS System.

Structured Query Language (SQL) -- a highly standardized high-level language used in relational database management systems to create and alter objects within a database.

Table -- the same as a SAS System data set.

View -- contains a definition or description of data stored in another location.

To retrieve and display data, the SELECT statement is used. To display data in a specific order, list the columns (variables) in the order desired on the SELECT statement.

To sum and display data in groups, SQL uses the GROUP BY clause. The GROUP BY clause is used when a summary function is specified in the query. The resulting output is grouped by the column specified in the GROUP BY clause.

To arrange results in a particular order (Ascending or Descending), SQL uses the ORDER BY clause. One or more columns can be selected for sorting. The default sort order is ascending (lowest to highest). To order data from highest to lowest specify the DESC (Descending) option following the column-name in the ORDER BY clause.

Retrieve and Display Data

To extract and retrieve data, we will use the SQL Procedure's SELECT statement. You will see in the following examples that the desired column(s) or variable(s) are displayed in the order indicated in the SELECT statement.

Exercise 1:

```
PROC SQL;
  SELECT  SSN, SEX
      FROM  libref.PATIENTS;
QUIT;
```

In Exercise 1, the columns SSN (Social Security Number) and SEX from the PATIENTS data set are selected and displayed.

Output:

SSN	SEX
123456789	M
043667543	F
153675389	M
932456132	M
...	...

Exercise 2:

```
PROC  SQL;
  SELECT  SSN,  LASTNAME
      FROM  libref.PATIENTS;
QUIT;
```

In exercise 2, the columns SSN and Patient's Last Name from the PATIENTS data set are selected and displayed.

Output:

SSN	LASTNAME
123456789	Smith
043667543	Jones
153675389	Cranberry
932456132	Henderson
...	...

Sum and Display Data in Groups

To sum and display data in groups, we will use the SQL statement GROUP BY. In the next example the GROUP BY clause is used when a summary function is used in a query. The column being summed is WEIGHT (Patient's Weight) with the results of the query being grouped by SEX (Patient's Gender).

Exercise 3:

```
PROC SQL;
  SELECT SEX, SUM(WEIGHT) AS
      TOTWEIGH
      FROM libref.PATIENTS
/* Shortness of Breath */
      WHERE SYMPTOM='10'
      GROUP BY SEX;
QUIT;
```

We are requesting the SQL procedure to perform several operations in exercise 3. Let's first examine the significance of the WHERE clause. It tells the SQL processor to extract only those rows (records) that contain a value of '10' (Shortness of Breath) in the SYMPTOM column. Rows not meeting this criteria are automatically excluded from the query. Then, the columns SEX (gender) and WEIGHT are selected from the PATIENTS data set. Next, the SQL processor groups rows (in ascending order) by the value found in the column SEX and totals each patient's weight storing the results in TOTWEIGH. Finally, the columns SEX and TOTWEIGH from the PATIENTS data set are selected and displayed.

Output:

SEX	TOTWEIGH
F	1800
M	3155

Arrange Results in Ascending Order

To arrange results in ascending order, we will direct SQL by using the ORDER BY clause. One or more columns can be selected for sorting. One or more columns can be ordered in either ascending and/or descending order. The default sort order is ascending (lowest to highest). If you want to override the default order (arrange in descending order), you need to specify DESC following the column-name that is specified.

Exercise 4:

```
PROC SQL;
  SELECT LASTNAME, EDUC
      FROM libref.PATIENTS
      ORDER BY EDUC;
QUIT;
```

In exercise 4, all rows are first arranged in ascending order by the column EDUC (patient's years of education). Then the columns LASTNAME and EDUC from the PATIENTS data set are selected and displayed.

Output:

LASTNAME	EDUC
Candle	10
Robertson	12
Cranberry	13
...	...

Syntax Requirements

The Structured Query Language (SQL) is directed by the statements, options, and components within the SQL procedure to create, retrieve, and modify data from tables and views. The syntax requirements for using the SQL procedure within the SAS System follow.

```
PROC SQL  < option(s) > ;
  CREATE  create-statement;
  DELETE  delete-statement;
  DROP  drop-statement;
  SELECT  select-statement;
  UPDATE  update-statement;
QUIT;
```

SQL Procedure Statements

SQL procedure statements are presented on the following pages.

SELECT Statement

The SELECT statement tells the SQL processor what columns of data to use in the query, formats desired information, and displays it as output.

General Format - SELECT:

```
PROC SQL;
  SELECT query-expression;
QUIT;
```

Exercise 5:

```
PROC SQL;
  SELECT LASTNAME, DOB
      FROM libref.PATIENTS;

  SELECT CLINIC, SYMPTOM, DIAGNOSE
      LABEL='Patient"s Diagnosis'
      FROM libref.PATIENTS;
QUIT;
```

CREATE Statement

The CREATE statement provides a way to create tables, views, and indexes for a data set (table). There are multiple ways of creating tables, views, and indexes. This workshop will illustrate the statement that creates a new table and the column definitions within the table.

General Format - CREATE TABLE:

```
PROC SQL;
  CREATE TABLE table-name
    (column-definition(s));
QUIT;
```

Data Types and Widths for Columns:

Data Types: CHARACTER, INTEGER, DECIMAL

Widths: Character columns default to 8 characters, Numeric columns are created using the maximum precision possible by the SAS System. A LENGTH statement can be specified in the DATA step.

Exercise 6:

```
PROC SQL;
  CREATE TABLE CLINICS(CLIN_NO
      char(6), DIRECTOR char(20));
QUIT;
PROC CONTENTS DATA=CLINICS;
RUN;
```

In exercise 6, a table called CLINICS is defined with two columns, CLIN_NO and DIRECTOR. The CONTENTS procedure is used to display data set (table) information.

General Format - CREATE VIEW:

```
PROC SQL;
  CREATE VIEW view-name AS
    query-expression;
QUIT;
```

where:

the view-name should be kept to a maximum of seven characters due to constraints of some host systems.

Note:

A view is not the same thing as a table. A view represents a stored query-expression (sometimes known as a virtual table) where a table represents stored data.

Exercise 7:

```
PROC SQL;
  CREATE VIEW libref.CONTACTS AS
      SELECT CLIN_NO, LASTNAME
    FROM libref.PATIENTS;


  SELECT *
      FROM libref.CONTACTS;
QUIT;
```

In exercise 7, a view is created called CONTACTS using the data derived from the PATIENTS data set. A query is then performed on the new view.

DELETE Statement

The DELETE statement removes one or more rows from a table as indicated in the WHERE clause. It is typically used with a WHERE clause in order to inform the SQL processor what row(s) to exclude.

General Format - DELETE FROM:

```
PROC SQL;
  DELETE FROM table-name
      WHERE sql-expression;
QUIT;
```

Exercise 8:

```
PROC SQL;
  DELETE FROM CLINICS
    WHERE CLIN_NO='011234';


/*  Displays all fields in CLINICS  */
      SELECT  CLIN_NO, REGION
            FROM  CLINICS;
QUIT;
```

In exercise 8, rows containing the value of '011234' in the CLIN_NO column are removed from the CLINICS data set. A query is then performed to select and display the columns CLIN_NO and REGION from CLINICS.

DROP Statement

The DROP statement deletes a table, view, or index. You must specify the libref when tables, views, or indexes are stored permanently.

General Format - DROP:

```
PROC SQL;
  DROP  TABLE  table-name;
  DROP VIEW view-name;
  DROP INDEX index-name
      FROM table-name;
QUIT;
```

Exercise 9:

```
PROC SQL;
  DROP TABLE libref.CLINICS;

  DROP VIEW libref.CONTACTS;
QUIT;
```

In exercise 9, the table (data set) CLINICS is first deleted. Then the view CONTACTS is deleted.

UPDATE Statement

The UPDATE statement allows for columns within existing rows (observations) of a table or view to be changed. Caution should be exercised while updating tables. Make sure you back-up your tables prior to changing data since accidents can happen.

General Format - UPDATE:

```
PROC SQL;
  UPDATE table-name  | libref.view-name
      SET set-clause
        < WHERE  where-expression >;
QUIT;
```

Exercise 10:

```
PROC SQL;
  UPDATE libref.PATCOPY /* Backup Copy */
      SET  WEIGHT=WEIGHT + 1;


      SELECT LASTNAME,  WEIGHT
            FROM libref.PATCOPY;
QUIT;
```

In exercise 10, an UPDATE statement is used to increment the WEIGHT column because of improper calibration of the scale. A query is then performed to select and display the columns LASTNAME and WEIGHT from the PATCOPY data set.

SQL Procedure Statement Components

The SQL procedure statement components provide a way to further enhance the selection and/or update criteria. The following components are available.

BETWEEN -- searches for data lying within certain parameters.

Column-Definition -- defines data types and widths.

Column-Modifier -- establishes column attributes.

From-List -- indicates what table or view to use in a FROM clause.

Group-by-Item -- indicates the groups of variables values processed in a GROUP BY clause.

Order-by-Item -- indicates the order in which observations are displayed in an ORDER BY clause.

SQL-Expression -- identifies functions, expressions, and operators that are used to connect them.

Conclusion

Structured Query Language (SQL) is a universal language that is available within the base SAS System product. It was developed to easily access data that is stored in relational databases or tables. Relations can be thought of as two-dimensional tables consisting of rows and columns (like a SAS System data set). Through the use of relational mathematics a series of guidelines for defining, manipulating, and controlling tables were developed by Dr. E. F. Codd.

The concept behind SQL is to free the user from having to specify physical attributes about the data such as data structure, location, and/or data type. The user concentrates on what data should be selected, but not how to select it.

The SQL Procedure provides a standardized way to retrieve and display data, sum and display data in groups, and arrange results in ascending or descending order by columns.

PROC SQL can run in both interactive and batch modes. It can be used to create, modify, and retrieve data from tables. Global statements such as TITLE and OPTIONS can be used with PROC SQL. Tables are accessed via two-level names where the first is the Libref and the second level is the name of the table.

Acknowledgments

My sincere thanks are extended to everyone involved with the SUGI Conference especially Gary Katsanis and David Woo, Section Chairs of the Hands-on Workshops.

Author Contact

The author will be happy to answer questions and accept suggestions at the following address:

Kirk Paul Lafler
Software Intelligence Corporation
P.O. Box 1390
Spring Valley, CA 91979-1390
Tel: (619) 670-SOFT or (619) 670-7638