

The SQL Procedure – Part 1

Overview/The SELECT Statement

Winnipeg SAS User Group
May 14th, 2008

Presented by: Craig Kasper
Manitoba Health and Healthy Living
Health Information Management Branch

PROC SQL – What is it?

- SQL stands for “Structured Query Language”
 - SQL was originally designed as a standardized language for database queries.
- SQL statements exist for:
 - Selecting and analyzing data
 - Creating, modifying and deleting tables
 - Adding, modifying, and deleting table rows
- This presentation is mainly about selecting and analyzing data.

PROC SQL vs. DATA Statements

- DATA Statement
 - Designed around the concept of rows
 - “Command”-oriented
 - “Do this with every row”
 - More powerful than PROC SQL
- PROC SQL
 - Designed around the concept of columns
 - “Data”-oriented
 - “Use this data to create this output variable”
 - Performs some tasks with less programming than a DATA statement; May be easier to read.

The SELECT statement

- A SELECT statement, in its simplest form, selects listed columns to be output from every row of a source table.¹
 - ```
PROC SQL;
 SELECT TableColumn,
 "Text Value" AS ConstantColumn,
 NumColumn1*NumColumn2+3 AS CalculatedColumn
 FROM example;
```
- SELECT statements can use any functions and expressions that are legal in a DATA statement, anywhere where a value is required.<sup>2</sup>
- Notice that the list of selected columns is separated by commas. In SQL, unlike in ordinary SAS programming, lists of columns are separated by commas, rather than by spaces only.

# The SELECT statement: FROM

- In SAS, the SELECT statement can't select values "out of thin air" – they must come from somewhere, even if you only want to select a single row of values.
- The FROM clause specifies where the SELECT statement will select values from.
  - This may be a single SAS table
  - This may also be from multiple SAS tables that are joined together.<sup>3</sup>

# The SELECT statement: WHERE

- The WHERE clause in SQL, is used to exclude data in certain rows from being selected.
- A WHERE clause consists of one or more criteria, optionally combined using AND and OR.
  - These criteria are expressions that can be evaluated as true or false.
  - In clauses where multiple criteria are combined, only the rows for which the combined criteria is true are retained.
- Any standard SAS function or expression which resolves to a “true” or “false” value can be used in a WHERE clause, including all of SAS’s comparisons, and functions like MISSING().

# WHERE clause examples

- PROC SQL;  
SELECT Neighb, HomeAddress, Agent, SellingPrice  
FROM RealEstateSales  
WHERE SaleYear=2007;
- PROC SQL;  
SELECT Neighb, SellingPrice, CommissionRate  
FROM RealEstateSales  
WHERE SellingPrice >= 200000 AND  
CommissionRate <= .07;
- PROC SQL;  
SELECT Neighb, SellingPrice, CommissionRate  
FROM RealEstateSales  
WHERE SellingPrice <= 100000 OR  
CommissionRate <= .04;

# The SELECT statement: ORDER BY

- A SAS DATA statement, because it is row-oriented, outputs rows in the same order they are read; sorts must be performed outside of the DATA statement.
- An SQL SELECT statement, however, can select and sort in a single statement.
- SQL sort order is specified using an ORDER BY clause.

# The SELECT statement: ORDER BY

- `PROC SQL;`  
    `SELECT Neighb, HomeAddress, Agent, SellingPrice`  
    `FROM RealEstateSales`  
    `WHERE SaleYear=2005`  
    `ORDER BY Agent, Neighb ASC, SellingPrice DESC;`
- An ORDER BY clause sorts using the leftmost column first, then the remaining columns in left to right to break any ties.
- Notice that you can force a column to be sorted in ascending (ASC) or descending (DESC) order, as shown above. If a column's sort order is not specified, it defaults to ascending order.

# Saving SELECT results

- To save the results of a SELECT query, you can use PROC SQL to create a table using CREATE TABLE (name) AS.
- Here's an example:

```
PROC SQL;
```

```
 CREATE TABLE AgentSalesPerformance AS
 SELECT Agent, Neighb, HomeAddress, SellingPrice
 FROM RealEstateSales
 WHERE SaleYear=2007
 ORDER BY Agent, Neighb ASC, SellingPrice DESC;
```

# Review: Parts of a SELECT query

- PROC SQL;  
CREATE TABLE {tablename} AS  
SELECT {columns}  
FROM {table}  
WHERE {row inclusion criteria}  
ORDER BY {sort order};
- The parts of a query must be assembled in this order. If the order is different, SAS will not be able to understand the query and will report an error.

# The SQL Procedure – Part 2

## Summary Functions and Grouping

Winnipeg SAS User Group  
May 14<sup>th</sup>, 2008

Presented by: Craig Kasper  
Manitoba Health and Healthy Living  
Health Information Management Branch

# SQL: Summary functions

- The SQL standard includes functions that can be used to summarize data over multiple rows.
- Here's an example:

- `PROC SQL;`

```
SELECT MAX(SellingPrice) as MaxPrice,
 MIN(SellingPrice) as MinPrice,
 AVERAGE(SellingPrice) as AvgPrice
FROM RealEstateSales
WHERE SaleYear=2005;
```

- The output from this query would look something like:

| <code>MaxPrice</code> | <code>MinPrice</code> | <code>AvgPrice</code> |
|-----------------------|-----------------------|-----------------------|
| <code>1355900</code>  | <code>42000</code>    | <code>189352.9</code> |

# SQL: Summary Functions

- SAS supports the following summary functions (also known as “aggregate functions”) in the SQL procedure:
  - MAX(variable) selects the highest value present in that variable.
  - MIN(variable) selects the lowest value present in that variable.
  - AVG(variable) calculates the average of all non-missing values in that variable.
  - COUNT(variable) counts the number of rows present in which that variable has a non-missing value
  - SUM(variable) calculates the total of all values present in that variable
- Other summary functions also exist, although these are among the most generally useful.<sup>4</sup>

# A Word about Missing Values

- You may have noticed that some of the SQL summary functions handle missing values in a specific way.
- The SQL standard has its own terminology for dealing with missing values; it calls them NULL values.
- SAS's PROC SQL follows this standard.
  - Accordingly, you may see WHERE clauses like “WHERE var1 IS NOT NULL” in other people's SQL code.
  - These are simply checks to determine whether a variable's value is missing or not.

# Grouping

- Remember our example query that produced high, low, and average values for a specific year?
- What if you wanted high, low, and average values by neighbourhood, for each year in the data set? Is it possible to avoid running a separate query for each year-neighbourhood combination? Yes, it is.
- SQL's summary functions return values for specific *groupings* of rows.
- If no groupings have been specified, the entire data set is considered to be a single group, as in our earlier example.

# Grouping: GROUP BY

- For the purposes of these summary functions, grouping is performed by grouping rows with the same values in a particular column or columns into a single group. This is done by adding a GROUP BY clause to the query.

- **PROC SQL;**

```
SELECT Neighb, SaleYear,
 MAX(SellingPrice) as MaxPrice,
 MIN(SellingPrice) as MinPrice,
 AVERAGE(SellingPrice) as AvgPrice
FROM RealestateData
GROUP BY Neighb, SaleYear
ORDER BY Neighb, SaleYear;
```

# Grouping: GROUP BY

- When the query on the previous slide is run, there will be one row for each combination of the grouped variables present in the data set.
  - Each combination will appear only once in the query output, because the variables have been grouped.
  - The maximum, minimum, and average selling prices for each year and neighbourhood will appear in that year-neighbourhood combination's row:

```
– Neighb SaleYear MaxPrice MinPrice AvgPrice
 Maples 2006 195900 97000 151276.25
 Maples 2007 211950 112900 162841.33
```

...

# WHERE and HAVING

- When a SELECT query uses both GROUP BY and WHERE, the WHERE takes effect before the GROUP BY does. This allows for a summarizing query to be applied to only some of the rows of the source table.
- PROC SQL;

```
SELECT Neighb, SaleYear,
 MAX(SellingPrice) as MaxPrice,
 MIN(SellingPrice) as MinPrice,
 AVERAGE(SellingPrice) as AvgPrice
FROM RealEstateData
WHERE AgentName="Kasper"
GROUP BY Neighb, SaleYear
ORDER BY Neighb, SaleYear;
```

# WHERE and HAVING

- A WHERE clause cannot remove rows from the output data after the grouping and summarizing is complete. Instead, a HAVING clause is used to retain or remove rows from the output of a grouped query.<sup>5</sup> Only data rows where the criteria in the HAVING clause are satisfied are retained in the output data.
- **PROC SQL;**  
**SELECT Neighbourhood, SaleYear,**  
**MAX(SellingPrice) as MaxPrice,**  
**MIN(SellingPrice) as MinPrice,**  
**AVERAGE(SellingPrice) as AvgPrice**  
**FROM RealEstateData**  
**GROUP BY Neighbourhood, SaleYear**  
**ORDER BY Neighbourhood, SaleYear**  
**HAVING AvgPrice > 150000;**

# Footnotes

1. While in most cases, a list of columns must be specified explicitly, an asterisk can be used instead of listing all of the columns in a table, as shown below.

```
SELECT * FROM Tablename;
```

The resulting data table contains every column that the table Tablename does.

2. The following functions cannot be used as part of an expression in SQL
  - LAG and DIF
  - SOUND

# Footnotes

3. Joining tables in SQL produces a result similar to interleaving data sets in a DATA step using “SET set1 set2; BY variables;”  
Because SQL queries can join multiple tables on multiple BY values, however, the syntax is a little more complex.  
More information on joining tables in SQL may be found by looking up **SAS Products → Base SAS → SAS SQL Procedure User’s Guide → Retrieving Data from Multiple Tables** in the table of contents of SAS 9.1’s online help.

# Footnotes

4. SAS's PROC SQL understands over a dozen different types of summary functions. A master list of these functions is given below. Where multiple functions yield the same result, they are listed on the same line
- AVG, MEAN           Average or mean of values
  - COUNT, FREQ, N     Aggregate number of non-missing values
  - CSS                 Corrected sum of squares
  - CV                 Coefficient of variation
  - MAX                Largest value
  - MIN                Smallest value
  - NMISS             Number of missing values
  - PRT                Probability of a greater absolute value of Student's t
  - RANGE             Difference between the largest and smallest values
  - STD                Standard deviation
  - STDERR            Standard error of the mean
  - SUM                Sum of values
  - SUMWGT            Sum of the weight variable values which is 1
  - T                  Testing the hypothesis that the population mean is zero
  - USS                Uncorrected sum of squares
  - VAR                Variance
- It is also worth mentioning that while SAS's PROC SQL understands a MEDIAN function, it does not do so in a normal or even useful way. PROC MEANS or PROC UNIVARIATE should be used to calculate medians instead.

# Footnotes

5. If a HAVING clause is used in a query where there is no GROUP BY clause, it is treated like a WHERE clause instead.