Proc SQL SELECT Statement

The SELECT statement is the primary tool of PROC SQL. You can use it to retrieve data from a table. You can use several optional clauses within the SELECT statement to place restrictions on a query.

SELECT and FROM Clauses

The SELECT statement must contain a SELECT clause and a FROM clause, both of which are <u>required</u> in a PROC SQL query. For example:

```
select Name
   from score_data;
```

This SELECT statement contains the following:

- a SELECT clause that lists the column ‘Name’
- a FROM clause that lists the table ‘score_data’, which contains the ‘Name’ column

WHERE Clause

The WHERE clause enables you to restrict the data that you retrieve by specifying a condition that each row of the table must satisfy. PROC SQL output includes only those rows that satisfy the condition.

The following SELECT statement contains a WHERE clause that restricts the query output to only those students that have gender = ‘m’ (male):

```
select Name
   from score_data
   where Gender = ‘m’;
```

SCstatisticalprogramming.com

ORDER BY Clause

The ORDER BY clause enables you to sort the output from a table by one or more columns (variables in SAS terms).

That is, you can put character values in either ascending or descending alphabetical order, and you can put numerical values in either ascending or descending numerical order. The default order is ascending.

For example, you can modify the previous example to list the data by descending order of the column ‘score’:

```
select Name
    from score_data
    where Gender = ‘m’
    order by score desc;
```

GROUP BY Clause

The GROUP BY clause enables you to break query results into subsets of rows.

When you use the GROUP BY clause, you use an aggregate function in the SELECT clause or a HAVING clause to instruct PROC SQL how to group the data. PROC SQL calculates the aggregate function separately for each group.

When you do not use an aggregate function, PROC SQL treats the GROUP BY clause as if it were an ORDER BY clause, and any aggregate functions are applied to the entire table.

The following query uses the MEAN function to list the average of score1 of each gender. The GROUP BY clause groups the students by gender, and the ORDER BY clause puts the values of gender in alphabetical order:

```
select *, mean(score3) as score3_ave

from score_data

group by gender

order by gender;
```

HAVING Clause

The HAVING clause works with the GROUP BY clause to restrict the groups in a query's results based on a given condition. PROC SQL applies the HAVING condition after grouping the data and applying aggregate functions.

For example, the following query restricts the groups to include only gender = ‘f’ (female):

```
select *, mean(score3) as score3_ave
from score_data
group by gender
having gender = 'f'
order by gender;
```

SELECT Statement: the order of clauses

When you construct a SELECT statement, you must specify the clauses in the following **order**:

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

Note: **Only the SELECT and FROM clauses are required.**

Example:

```
PROC SQL;
   CREATE TABLE scoredata0 AS
   SELECT stu_id, gender, name
   FROM score_data
   WHERE gender in (‘m’);
QUIT;
```

Note:

- To create a PROC SQL table from a query result, use a CREATE TABLE statement with the AS keyword, and place it before the SELECT statement. scoredata0 in this case is the table (data set) that will be created.
- a SELECT clause is where you list the variables you want. Notice that the variables in this list are separated by commas (spaces do not work).
- The clause FROM names the data set you want to read.
- Finally, a WHERE clause describes the particular subset you want.
- SELECT, FROM, and WHERE form a single query, which you end with a single semicolon.
- the query ends with a QUIT statement. You do not need a RUN statement because PROC SQL executes as soon as a complete query has been specified. If you don’t include a QUIT statement, PROC SQL remains in memory for another query.