Appendix F: PROC SQL in SAS

- SQL stands for Structured Query Language, a language suited for database management and manipulation.
- `PROC SQL` is a SAS procedure that is based on SQL statements.
- Some of the syntax is similar to the SAS data step, but there are key differences, e.g., `CREATE TABLE` (rather than `DATA`) creates a data set.
- `PROC SQL` does not need a `RUN;` statement to execute. `PROC SQL` is typically ended with a `QUIT;` statement.

- PROC SQL performs many of the same tasks as the DATA step, but PROC SQL has some advantages:
  1. Faster execution speed.
  2. Joining tables with PROC SQL considered by some more convenient than MERGE in a DATA step.
  3. SQL code can easily access external databases (e.g., Oracle, DB2).

- One of the simplest tasks in PROC SQL is to select and print a data set that is already created. An easy way to do this:

```
PROC SQL;
SELECT *
FROM tablename;
QUIT;
```

(The * says to select *all* variables (columns) in the table.)

- By default this code prints the data set to the output window.
- We can also select only a few variables by specifying the variable names (separated by commas) in the SELECT statement.

- We may wish to create a new data set from part of a previous one. We use the `CREATE TABLE ... AS` statement:

```
CREATE TABLE newtablename AS
SELECT var1, var3, var4
FROM oldtablename;
```

- Some `DATA` step keywords work in `PROC SQL` as well (`DROP, KEEP, RENAME, WHERE`).

One way to create a data set from scratch is to use to `CREATE TABLE` keywords without `AS`:

- After the `CREATE TABLE` line, you specify the names and types of the variables.
- The raw data is entered into the table with an `INSERT INTO` statement:

```
CREATE TABLE tablename
(var1 var1type
var2 var2type
var3 var3type);
```

Other tasks using SQL keywords:

- `DISTINCT:` selects unique values of variables that have duplicate values.
- `ORDER BY:` sorts a table by the values of one or more variables.

- Subsetting in `PROC SQL` is typically done with a `WHERE` statement.
- Various calculations can be done to create new variables. Calculations may be done on the whole table, or on groups of observations identified by some grouping variable.
- If a calculation involves a variable not in the original data set, but which has been calculated, use keyword `CALCULATED` with that variable.
- To "subset" based on "calculated" variables, do not use `WHERE`, but rather use the `HAVING` keyword.

  The `PROC SQL` equivalent of an `IF-THEN` statement is a `CASE` statement:

```
  CASE expression
WHEN expvalue1 THEN resvalueA
WHEN expvalue2 THEN resvalueB
...
ELSE resvalueZ END AS resultcolumn
```

Joining Tables in PROC SQL

- Compared to merging data sets in the DATA step, joining tables in PROC SQL is executed faster.
- In PROC SQL, the key columns (BY variables) do not need to be sorted first.
- "Many to many" merges are possible using PROC SQL.
- There are four main methods of joining tables using PROC SQL: the *inner join*, the *left join*, the *right join*, and the *full join*.

Important statements:

- FROM statement specifies source tables and "aliases" for those source tables, and also specifies the method of joining.
- ON statement specifies "key columns" (like BY variables in a DATA step merge) and possibly logical operators.
- The SELECT statement contains the table aliases as well as the variables to be selected.

Ways of Joining Tables

- *Inner join*: result lists only observations for which the values of the "key columns" match.
- *Left join*: result lists all observations in the "left" table (listed first in the `FROM` statement) and only the matching observations in the "right" table.
- *Right join*: result lists all observations in the "right" table (listed second in the `FROM` statement) and only the matching observations in the "left" table.
- *Full join*: a combination of the left and right joins.

Editing/Managing Data Tables with PROC SQL

- `INSERT INTO` is a typical way to add new observations to a table.
- `VALUES` statement specifies the values to be added (in parentheses, separated by spaces)
- Another way: Use a `SET` keyword with column names and newly assigned values:

```
SET col1 = 7, col2 = 'charstring', col3 = 44;
```

- To delete observations from a table, use `DELETE FROM` statement.
- To change the values of one or more columns in a table, use `UPDATE` statement along with `SET` statement.
- `ALTER TABLE` can be used to change column formats or to delete columns from a table. (Typically done with the `MODIFY` and `DROP` keywords, respectively.)
- `DROP TABLE` can also be used to delete an entire table.

Other Things

- `NOPRINT` option suppresses printing to the OUTPUT window:

```
PROC SQL NOPRINT;
```

- Note: When a `CREATE` statement is used, `NOPRINT` is the default.
- Within `PROC SQL` there is a system table `DICTIONARY.COLUMNS` with contents information about the columns of a table.
- We can view this with a `WHERE` statement specifying the Libname and Memname of our table.