### Basic Queries

**PROC SQL** <options>;<br>
**SELECT** column-1, ..., column-n<br>
**FROM** input-table<br>
*WHERE* expression<br>
*GROUP BY* col-name<br>
*HAVING* expression<br>
*ORDER BY* col-name <DESC> <..., col-name>;

**SQL Query Order of Execution:**

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Retrieve data from a table</td>
</tr>
<tr>
<td>FROM</td>
<td>Choose and join tables</td>
</tr>
<tr>
<td>WHERE</td>
<td>Filter the data</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Aggregate the data</td>
</tr>
<tr>
<td>HAVING</td>
<td>Filter the aggregate data</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Sort the final data</td>
</tr>
</tbody>
</table>

### Managing Tables

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE</td>
<td><code>CREATE TABLE</code> table-name (column-specification-1,&lt;,...column-specification-n&gt;);</td>
</tr>
<tr>
<td>DESCRIBE TABLE</td>
<td><code>DESCRIBE TABLE</code> table-name-1 &lt;,...,table-name-n&gt;;</td>
</tr>
<tr>
<td>DROP TABLE</td>
<td><code>DROP TABLE</code> table-name-1 &lt;,...,table-name-n&gt;;</td>
</tr>
</tbody>
</table>

### Managing Views

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE VIEW</td>
<td><code>CREATE VIEW</code> table-name AS query;</td>
</tr>
<tr>
<td>DESCRIBE VIEW</td>
<td><code>DESCRIBE VIEW</code> view-name-1 &lt;,...,view-name-n&gt;;</td>
</tr>
<tr>
<td>DROP VIEW</td>
<td><code>DROP VIEW</code> view-name-1 &lt;,...,view-name-n&gt;;</td>
</tr>
</tbody>
</table>

### Modifying Columns

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABEL=</td>
<td>Select col-name&lt;br&gt;Label='column label'</td>
</tr>
<tr>
<td>FORMAT=</td>
<td>Select col-name&lt;br&gt;Format=format.</td>
</tr>
</tbody>
</table>

Creating a new column<br>
Filtering new columns<br>

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>col-name AS new-col-name</td>
</tr>
<tr>
<td>WHERE</td>
<td>CALCULATED new-col-name</td>
</tr>
</tbody>
</table>

### Modifying Rows

Inserting rows into tables<br>
Eliminating duplicate rows<br>
Filtering rows<br>

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT INTO</td>
<td>table&lt;br&gt;SET column-name=value&lt;br&gt;&lt;,...column-name=value&gt;;</td>
</tr>
<tr>
<td>VALUES</td>
<td>(value1, value2, ...)</td>
</tr>
<tr>
<td>SELECT</td>
<td>column-1, ..., column-n FROM input-table;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE</td>
<td>col-name IN (value1, value2, ...)</td>
</tr>
<tr>
<td>WHERE</td>
<td>col-name LIKE &quot;_string%&quot;</td>
</tr>
<tr>
<td>WHERE</td>
<td>col-name BETWEEN value AND value</td>
</tr>
<tr>
<td>WHERE</td>
<td>col-name IS NULL</td>
</tr>
<tr>
<td>WHERE</td>
<td>date-value &quot;&lt;01JAN2019&quot;&gt;&quot;d</td>
</tr>
<tr>
<td>WHERE</td>
<td>time-value &quot;&lt;14:45:35&quot;&gt;&quot;t</td>
</tr>
<tr>
<td>WHERE</td>
<td>datetime-value &quot;&lt;01JAN201914:45:35&quot;&gt;&quot;dt</td>
</tr>
</tbody>
</table>

### Remerging Summary Statistics

**SELECT** col-name, summary function(argument)<br>**FROM** input table;
Joins Summary

**Inner Join**

```sql
SELECT <list>
FROM table-A INNER JOIN table-B
ON A.Key = B.Key;
```

**Full Join**

```sql
SELECT <list>
FROM table-A FULL JOIN table-B
ON A.Key = B.Key;
```

**Right Join**

```sql
SELECT <list>
FROM table-A RIGHT JOIN table-B
ON A.Key = B.Key;
```

**Left Join**

```sql
SELECT <list>
FROM table-A LEFT JOIN table-B
ON A.Key = B.Key;
```

---

**Set Operators**

The **INTERSECT** operator selects unique rows that are common to both tables.

```sql
SELECT <list>
FROM table-A INTERSECT
SELECT <list>
FROM table-B;
```

The **EXCEPT** operator selects unique rows from table A that are not found in table B.

```sql
SELECT <list>
FROM table-A EXCEPT
SELECT <list>
FROM table-B;
```

The **UNION** operator selects unique rows from both tables.

```sql
SELECT <list>
FROM table-A UNION
SELECT <list>
FROM table-B;
```

The **OUTER UNION** operator selects all rows from both tables.

```sql
SELECT <list>
FROM table-A OUTER UNION
SELECT <list>
FROM table-B;
```

---

**Creating Macro Variables**

**Storing a value in a macro variable using SQL:**

```sql
SELECT col-name-1 <,...,col-name-n>
INTO:macrvar_1 <,...,macvar-n>
FROM input-table;
```

**Storing a list of values in a macro variable using SQL:**

```sql
SELECT col-name-1 <,...,col-name-n>
INTO:macrvar_1 SEPARATED BY 'delimiter'
FROM input-table;
```

**Viewing the value of the macro variable in the SAS Log:**

```
%PUT &=macvar;
```

---

**Subqueries**

```sql
SELECT col-name,
(SELECT function(argument)
FROM input-table)
FROM input-table;
```

```sql
SELECT col-name, <,...,col-name>
FROM input-table
WHERE col-name
(SELECT function(argument)
FROM input-table)
```

---

**Accessing DBMS Data**

The SQL pass-through facility enables you to code in the native DBMS SQL syntax and pass the query to the database.

```
PROC SQL;
CONNECT TO DBMS-name <AS alias>
( DBMS-connection-options);
SELECT col-name
FROM CONNECTION TO DBMS-name\|alias (dbms-query);
DISCONNECT FROM DBMS-name\|alias;
QUIT;
```