**Basic Queries**

**PROC SQL**

```sql
PROC SQL <options>;
SELECT column-1 <, ..., column-n>
FROM input-table
WHERE expression
GROUP BY col-name
HAVING expression
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>CREATE TABLE</th>
<th>CREATE TABLE table-name (column-specification-1, ..., column-specification-n);</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIBE TABLE</td>
<td>DESCRIBE TABLE table-name-1 &lt;,...,table-name-n&gt;;</td>
</tr>
<tr>
<td>DROP TABLE</td>
<td>DROP TABLE table-name-1 &lt;,...,table-name-n&gt;;</td>
</tr>
</tbody>
</table>

**Managing Views**

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

**Modifying Columns**

<table>
<thead>
<tr>
<th>LABEL=</th>
<th>SELECT col-name LABEL='column label'</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT=</td>
<td>SELECT col-name FORMAT=format.</td>
</tr>
</tbody>
</table>

Creating a new column

```
SELECT col-name AS new-col-name
```

Filtering new columns

```
WHERE CALCULATED new-col-name
```

**Modifying Rows**

Inserting rows into tables

```
INSERT INTO table
SET column-name=value <, ..., column-name=value>; 
```

```
INSERT INTO table <(column-list)> VALUES (value<,...,value>);
```

```
INSERT INTO table <(column-list)> SELECT column-1<,...,column-n> FROM input-table;
```

Eliminating duplicate rows

```
SELECT DISTINCT col-name<,...,col-name>
```

Filtering rows

```
WHERE col-name IN (value1, value2, ...)
```

```
WHERE col-name LIKE "_string%"
```

```
WHERE col-name BETWEEN value AND value
```

```
WHERE col-name IS NULL
```

```
WHERE date-value "<01JAN2019">d
```

```
WHERE time-value "<14:45:35">t
```

```
WHERE datetime-value "<01JAN201914:45:35">dt
```

**Remerging Summary Statistics**

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

**Inner Join**

\[
\text{SELECT } \text{<list> FROM table-A INNER JOIN table-B ON A.Key=B.Key;}
\]

**Full Join**

\[
\text{SELECT } \text{<list> FROM table-A FULL JOIN table-B ON A.Key=B.Key;}
\]

**Right Join**

\[
\text{SELECT } \text{<list> FROM table-A RIGHT JOIN table-B ON A.Key=B.Key;}
\]

**Left Join**

\[
\text{SELECT } \text{<list> FROM table-A LEFT JOIN table-B ON A.Key=B.Key;}
\]

**Set Operators**

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

\[
\text{SELECT } \text{<list> FROM table-A INTERSECT SELECT } \text{<list> FROM table-B;}
\]

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

\[
\text{SELECT } \text{<list> FROM table-A EXCEPT SELECT } \text{<list> FROM table-B;}
\]

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

\[
\text{SELECT } \text{<list> FROM table-A UNION SELECT } \text{<list> FROM table-B;}
\]

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

\[
\text{SELECT } \text{<list> FROM table-A OUTER UNION SELECT } \text{<list> FROM table-B;}
\]

**Creating Macro Variables**

Storing a value in a macro variable using SQL:

\[
\text{SELECT col-name-1 <,...,col-name-n> INTO:macrvar_1<,...,macvar-n> FROM input-table;}
\]

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

\[
\text{SELECT col-name-1 <,...,col-name-n> INTO:macrvar_1 SEPARATED BY 'delimiter' FROM input-table;}
\]

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

\[
\%PUT &=macvar;
\]

**Subqueries**

\[
\text{SELECT col-name, (SELECT function(argument) FROM input-table) FROM input-table;}
\]

\[
\text{SELECT col-name, <,...,col-name> FROM input-table WHERE col-name (SELECT function(argument) FROM input-table)}
\]

**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.

\[
\text{PROC SQL; CONNECT TO DBMS-name <AS alias> (DBMS-connection-options);}
\]

\[
\text{SELECT col-name FROM CONNECTION TO DBMS-name|alias (dbms-query);}
\]

\[
\text{DISCONNECT FROM DBMS-name|alias; QUIT;}
\]