### 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>

### Modifying Columns

- **LABEL=**
  ```sql
  SELECT col-name LABEL='column label'
  ```

- **FORMAT=**
  ```sql
  SELECT col-name FORMAT=format.
  ```

- Creating a new column
  ```sql
  SELECT col-name AS new-col-name
  ```

- Filtering new columns
  ```sql
  WHERE CALCULATED new-col-name
  ```

### Modifying Rows

- **Inserting rows into tables**
  ```sql
  INSERT INTO table 
  SET column-name=value 
  <,...column-name=value>
  ```

- Eliminating duplicate rows
  ```sql
  SELECT DISTINCT col-name<,...col-name
  ```

### Managing Tables

- **CREATE TABLE**
  ```sql
  CREATE TABLE table-name 
  (column-specification-1<,...column-specification-n>);
  ```

- **DESCRIBE TABLE**
  ```sql
  DESCRIBE TABLE table-name-1 <,...table-name-n>;
  ```

- **DROP TABLE**
  ```sql
  DROP TABLE table-name-1 <,...table-name-n>;
  ```

### Managing Views

- **CREATE VIEW**
  ```sql
  CREATE VIEW table-name AS query;
  ```

- **DESCRIBE VIEW**
  ```sql
  DESCRIBE VIEW view-name-1 <,...view-name-n>;
  ```

- **DROP VIEW**
  ```sql
  DROP VIEW view-name-1 <,...view-name-n>;
  ```

### Remerging Summary Statistics

```sql
SELECT col-name, summary function(argument) 
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;
```

## 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:

```sas
%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)
```

## 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;
```

## 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.

```sas
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;
```