SQL Query Clauses

SELECT Clause

The SELECT clause is used to retrieve data from a database. It specifies the columns to be displayed.

Syntax:
```
SELECT column1, column2, ...
FROM table_name;
```

Example:
```
SELECT FirstName, LastName
FROM Employees;
```

FROM Clause

The FROM clause specifies the table from which to retrieve the data.

Syntax:
```
SELECT column1, column2, ...
FROM table_name;
```

Example:
```
SELECT *
FROM Employees;
```

WHERE Clause

The WHERE clause is used to filter records that meet a certain condition.

Syntax:
```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example:
```
SELECT *
FROM Employees
WHERE Position = 'Manager';
```

GROUP BY Clause

The GROUP BY clause groups rows that have the same values into summary rows. It is often used with aggregate functions like COUNT, MAX, MIN, SUM, and AVG.

Syntax:

```
SELECT column1, aggregate_function(column2)
FROM table_name
WHERE condition
GROUP BY column1;
```

Example:

```
SELECT Position, COUNT(*)
FROM Employees
GROUP BY Position;
```

HAVING Clause

The HAVING clause is used to filter groups based on a condition. It is similar to the WHERE clause, but it applies to groups of rows rather than individual rows.

Syntax:

```
SELECT column1, aggregate_function(column2)
FROM table_name
WHERE condition
GROUP BY column1
HAVING condition;
```

Example:

```
SELECT Position, COUNT(*)
FROM Employees
GROUP BY Position
HAVING COUNT(*) > 1;
```

ORDER BY Clause

The ORDER BY clause is used to sort the result set in ascending or descending order.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC];
```

Example:

```
SELECT FirstName, LastName
FROM Employees
ORDER BY LastName ASC;
```

LIMIT Clause

The LIMIT clause is used to specify the number of records to return.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
LIMIT number;
```

Example:

```
SELECT *
FROM Employees
LIMIT 10;
```

JOIN Clause

The JOIN clause is used to combine rows from two or more tables based on a related column between them.

- INNER JOIN: Returns records that have matching values in both tables.

  Syntax:

```
 SELECT column1, column2, ...
 FROM table1
 INNER JOIN table2
 ON table1.common_column = table2.common_column;
```

  Example:

```
 SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName
 FROM Employees
 INNER JOIN Departments
 ON Employees.DepartmentID = Departments.DepartmentID;
```

- LEFT JOIN (or LEFT OUTER JOIN): Returns all records from the left table and the matched records from the right table. Returns NULL for non-matching rows in the right table.

  Syntax:

```
 SELECT column1, column2, ...
 FROM table1
```

LEFT JOIN table2
ON table1.common_column = table2.common_column;

Example:
SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName
FROM Employees
LEFT JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;

- RIGHT JOIN (or RIGHT OUTER JOIN): Returns all records from the right table and the matched records from the left table. Returns NULL for non-matching rows in the left table.

Syntax:
SELECT column1, column2, ...
FROM table1
RIGHT JOIN table2
ON table1.common_column = table2.common_column;

Example:
SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName
FROM Employees
RIGHT JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;

- FULL JOIN (or FULL OUTER JOIN): Returns all records when there is a match in either left or right table. Returns NULL for non-matching rows in both tables.

Syntax:
SELECT column1, column2, ...
FROM table1
FULL JOIN table2
ON table1.common_column = table2.common_column;

Example:
SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName
FROM Employees
FULL JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;

UNION Clause

The UNION clause is used to combine the result sets of two or more SELECT statements. Each SELECT statement within the UNION must have the same number of columns in the result sets with similar data types.

Syntax:
```
SELECT column1, column2, ...
FROM table1
UNION
SELECT column1, column2, ...
FROM table2;
```

Example:
```
SELECT FirstName, LastName
FROM Employees_USA
UNION
SELECT FirstName, LastName
FROM Employees_Canada;
```

INSERT INTO Clause

The INSERT INTO clause is used to insert new records into a table.

Syntax:
```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

Example:
```
INSERT INTO Employees (FirstName, LastName, BirthDate, Position)
VALUES ('John', 'Doe', '1980-01-01', 'Manager');
```

UPDATE Clause

The UPDATE clause is used to modify existing records in a table.

Syntax:
```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example:
```
UPDATE Employees
SET Position = 'Senior Manager'
WHERE EmployeeID = 1;
```

DELETE Clause

The DELETE clause is used to delete existing records from a table.

Syntax:

```
DELETE FROM table_name
WHERE condition;
```

Example:

```
DELETE FROM Employees
WHERE EmployeeID = 1;
```

Summary

These clauses are fundamental to constructing SQL queries, enabling users to retrieve, insert, update, delete, and manage data effectively within a relational database. By combining these clauses, complex and powerful queries can be crafted to meet various data manipulation and retrieval requirements.