1. Basic SELECT Statement

The **SELECT** statement is used to retrieve data from a database. You can select one or more columns from one or more tables.

Syntax:

```
sql
Copy
SELECT column1, column2, ...
FROM table_name;
```

Example 1: Selecting Specific Columns

To select only specific columns (e.g., `Name` and `Salary`) from the `Employees` table:

```
sql
Copy
SELECT Name, Salary
FROM Employees;
```

Example 2: Selecting All Columns

You can use the `*` wildcard to select all columns from the table:

```
sql
Copy
SELECT *
FROM Employees;
```

This will retrieve all columns in the `Employees` table.

---

2. DISTINCT

The **DISTINCT** keyword is used to remove duplicate values from the result set. It ensures that only unique rows are returned for the selected columns.

Syntax:

```
sql
Copy
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Example:

To get a list of unique job titles from the `Employees` table:

```
sql
Copy
SELECT DISTINCT JobTitle
```

```
FROM Employees;
```

This query will return all unique job titles without duplicates.

---

3. ORDER BY

The **ORDER BY** clause is used to sort the result set. You can sort by one or more columns in ascending or descending order.

Syntax:

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

- **ASC**: Sorts in ascending order (default).
- **DESC**: Sorts in descending order.

Example 1: Sorting by One Column (Ascending)

To get all employees sorted by their `Salary` in ascending order:

```
sql
Copy
SELECT Name, Salary
FROM Employees
ORDER BY Salary;
```

Example 2: Sorting by One Column (Descending)

To get all employees sorted by their `Salary` in descending order:

```
sql
Copy
SELECT Name, Salary
FROM Employees
ORDER BY Salary DESC;
```

Example 3: Sorting by Multiple Columns

To sort employees by their `JobTitle` (alphabetically) and then by `Salary` (descending):

```
sql
Copy
SELECT Name, JobTitle, Salary
FROM Employees
ORDER BY JobTitle ASC, Salary DESC;
```

---

4. LIMIT

The **LIMIT** clause is used to specify the number of records to return. It is especially useful when you want to retrieve a subset of rows, like the top 10 records.

Syntax:

```
sql
Copy
SELECT column1, column2, ...
FROM table_name
LIMIT number_of_rows;
```

Example:

To retrieve the first 5 employees from the `Employees` table:

```
sql
Copy
SELECT Name, Salary
FROM Employees
LIMIT 5;
```

This query returns only the first 5 rows of the result set.

---

5. OFFSET

The **OFFSET** clause is used to specify how many rows to skip before starting to return rows. It is commonly used in combination with **LIMIT** to implement pagination in queries.

Syntax:

```
sql
Copy
SELECT column1, column2, ...
FROM table_name
LIMIT number_of_rows OFFSET number_of_rows_to_skip;
```

Example:

To retrieve employees 6 through 10 from the `Employees` table (skipping the first 5 records):

```
sql
Copy
SELECT Name, Salary
FROM Employees
LIMIT 5 OFFSET 5;
```

This query skips the first 5 rows and returns the next 5 rows, which will be employees 6 through 10.

Practical Use Cases for These Concepts:

1. **Basic SELECT Statement**: Retrieving data from a table.

   ```
   sql
   Copy
   SELECT Name, Department FROM Employees;
   ```

2. **DISTINCT**: Removing duplicates in the result set, e.g., getting unique job titles:

   ```
   sql
   Copy
   SELECT DISTINCT JobTitle FROM Employees;
   ```

3. **ORDER BY**: Sorting the result set, e.g., sorting employees by age:

   ```
   sql
   Copy
   SELECT Name, Age FROM Employees ORDER BY Age DESC;
   ```

4. **LIMIT**: Limiting the number of results returned, e.g., showing only the first 3 employees:

   ```
   sql
   Copy
   SELECT Name, Salary FROM Employees LIMIT 3;
   ```

5. **OFFSET**: Skipping a set number of rows, e.g., pagination:

   ```
   sql
   Copy
   SELECT Name, Salary FROM Employees LIMIT 5 OFFSET 5;
   ```

Summary

- **SELECT**: Retrieves data from a database table.
- **DISTINCT**: Removes duplicate values from the result set.
- **ORDER BY**: Sorts the result set by one or more columns in ascending or descending order.
- **LIMIT**: Restricts the number of rows returned in the result set.
- **OFFSET**: Skips a specified number of rows before starting to return rows, useful for pagination.

DML (Data Manipulation Language) Commands

DML commands are used for manipulating the data within database tables. Unlike DDL commands that define the structure of the database, DML commands are used to interact with the data. The key DML commands are:

1. **INSERT**
2. **SELECT**
3. **UPDATE**
4. **DELETE**

Below is a detailed explanation of each DML command, along with examples.

---

1. INSERT

The **INSERT** statement is used to add new rows (records) to a table.

Syntax:

```
sql
Copy
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- You must specify the table name and the values that should be inserted into each column.
- You can insert values into specific columns or all columns.

Example 1: Insert Data into Specific Columns

If you want to insert only specific columns into the `Employees` table:

```
sql
Copy
INSERT INTO Employees (Name, Age, Department)
VALUES ('John Doe', 30, 'Marketing');
```

In this example, a new employee named **John Doe**, aged **30**, and working in the **Marketing** department is added to the `Employees` table.

Example 2: Insert Data into All Columns

If you want to insert data into all columns:

```
sql
Copy
INSERT INTO Employees
VALUES (101, 'Jane Smith', 25, 'Finance', 60000);
```

Here, data is inserted into all columns of the `Employees` table in the order the columns are defined.

---

2. SELECT

The **SELECT** statement is used to retrieve data from a table. It is the most commonly used DML command to query and view data in a database.

Syntax:

```
sql
Copy
SELECT column1, column2, ...
FROM table_name;
```

- **SELECT** allows you to fetch specific columns or all columns using `*`.
- **WHERE** is used to filter the results.
- **ORDER BY** is used to sort the result.

Example 1: Select All Columns

To retrieve all data from the `Employees` table:

```
sql
Copy
SELECT * FROM Employees;
```

Example 2: Select Specific Columns

To retrieve only specific columns:

```
sql
Copy
SELECT Name, Department FROM Employees;
```

Example 3: Select with Conditions

You can filter the results with the **WHERE** clause:

```
sql
Copy
SELECT * FROM Employees WHERE Age > 30;
```

This retrieves all employees whose age is greater than 30.

---

3. UPDATE

The **UPDATE** statement is used to modify the existing records in a table.

Syntax:

```
sql
Copy
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- **SET** specifies the column(s) and the new values.
- **WHERE** condition limits the rows to update. If you omit the **WHERE** clause, all rows in the table will be updated.

Example 1: Update a Single Column

To increase the salary of an employee with a specific `EmployeeID`:

```
sql
Copy
UPDATE Employees
SET Salary = 75000
WHERE EmployeeID = 101;
```

This will update the **Salary** of the employee with `EmployeeID` 101 to **75000**.

Example 2: Update Multiple Columns

You can also update multiple columns in one statement:

```
sql
Copy
UPDATE Employees
SET Salary = 80000, Department = 'HR'
WHERE EmployeeID = 102;
```

This updates the **Salary** to **80000** and changes the **Department** to **HR** for the employee with `EmployeeID` 102.

---

4. DELETE

The **DELETE** statement is used to remove one or more rows from a table.

Syntax:

```
sql
Copy
DELETE FROM table_name
WHERE condition;
```

- **WHERE** specifies which rows should be deleted. If you omit the **WHERE** clause, all rows in the table will be deleted.

Example 1: Delete a Single Record

To delete an employee with a specific `EmployeeID`:

```
sql
Copy
DELETE FROM Employees
WHERE EmployeeID = 103;
```

This deletes the record of the employee with `EmployeeID` 103 from the `Employees` table.

Example 2: Delete All Records

To delete all records from a table, without deleting the table itself, omit the **WHERE** clause:

```
sql
Copy
DELETE FROM Employees;
```

This will remove all rows from the `Employees` table.

Summary of DML Commands:

1. **INSERT**: Adds new records to a table.
   - Example: `INSERT INTO Employees (Name, Age, Department) VALUES ('John Doe', 30, 'Marketing');`
2. **SELECT**: Retrieves data from a table.
   - Example: `SELECT Name, Department FROM Employees WHERE Age > 30;`
3. **UPDATE**: Modifies existing records in a table.
   - Example: `UPDATE Employees SET Salary = 75000 WHERE EmployeeID = 101;`
4. **DELETE**: Removes records from a table.
   - Example: `DELETE FROM Employees WHERE EmployeeID = 103;`
