## Unit- 4

**SQL – need for SQL, types of SQL statements, data types in SQL, SELECT statement with various operators, single row and multi row functions, group by and having clauses.**

### ❖ Need for SQL

SQL (Structured Query Language) is essential for managing and manipulating relational databases. It allows users to:

- **Create, read, update, and delete (CRUD)** data.
- Define the structure of the database (tables, views, indexes).
- Control access to the data (permissions, roles).
- Retrieve and analyze data efficiently through queries.

SQL is the standard language for relational database management systems (RDBMS) like **MySQL, PostgreSQL, Oracle, and SQL Server.**

### ❖ Types of SQL Statements

SQL statements are categorized based on their functionality.

**SQL is broadly categorized into five main types:**

**1. DDL – Data Definition Language**

Used to define and modify database structures:

- CREATE – Create a new table/database.
- ALTER – Modify an existing table.
- DROP – Delete a table/database.
- TRUNCATE – Remove all rows from a table.

| Statement | SQL Example | Explanation |
|---|---|---|
| CREATE | CREATE TABLE Students ( ID INT, Name VARCHAR(50), Age INT ); | Creates the Students table |
| ALTER | ALTER TABLE Students ADD Email VARCHAR(100); | Adds a new column to the table |

| Statement | SQL Example | Explanation |
|---|---|---|
| DROP | DROP TABLE Students; | Deletes the Students table |
| TRUNCATE | TRUNCATE TABLE Students; | Removes all rows but keeps structure |

**2. DML – Data Manipulation Language**

Used to manage data within tables:

- INSERT – Add new data.
- UPDATE – Modify existing data.
- DELETE – Remove data.

| Statement | SQL Example | Explanation |
|---|---|---|
| INSERT | INSERT INTO Students (ID, Name, Age) VALUES (1, 'Alice', 20); | Adds a new student record |
| UPDATE | UPDATE Students SET Age = 21 WHERE ID = 1; | Modifies the age of Alice |
| DELETE | DELETE FROM Students WHERE ID = 1; | Deletes Alice's record |

**3. DQL – Data Query Language**

Used to fetch/query data:

- SELECT – Retrieve data from one or more tables.

| Statement | SQL Example | Explanation |
|---|---|---|
| SELECT | SELECT * FROM Students WHERE Age > 18; | Retrieves students older than 18 |

**4. TCL – Transaction Control Language**

Used to manage transactions:

- ❖ COMMIT – Save changes, Saves changes made by a transaction.

- ROLLBACK – Undo changes, Undoes changes made by a transaction.
    - SAVEPOINT – Sets a point within a transaction to roll back to.

| Statement | SQL Example | Explanation |
|---|---|---|
| COMMIT | COMMIT; | Saves the transaction permanently |
| ROLLBACK | ROLLBACK; | Undoes changes since last commit |
| SAVEPOINT | SAVEPOINT sp1; | Sets a point to roll back to |

**5. DCL – Data Control Language**

Used to control access:

- GRANT – Provide access privileges, Gives permissions to users.
- REVOKE – Withdraw access privileges, Removes permissions from users.

| Statement | SQL Example | Explanation |
|---|---|---|
| GRANT | GRANT SELECT ON Students TO user1; | Gives read access to user1 |
| REVOKE | REVOKE SELECT ON Students FROM user1; | Removes read access from user1 |

**DDL (Data Definition Language)**: These statements define the structure of the database.

- **CREATE**: Creates a new database, table, index, or view.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Department VARCHAR(50),
    Salary DECIMAL(10, 2),
    HireDate DATE
);
```

- EmployeeID: Unique ID for each employee (**Primary Key**).
- FirstName / LastName: Names (up to 50 characters).
- Department: Department name.
- Salary: Employee salary with up to 10 digits and 2 decimals.
- HireDate: The date the employee was hired.

📋 Employees Table

| EmployeeID | FirstName | LastName | Department | Salary | HireDate |
|---|---|---|---|---|---|
| 101 | Alice | Johnson | HR | 50000.00 | 2021-06-15 |
| 102 | Bob | Smith | IT | 65000.00 | 2020-09-01 |
| 103 | Clara | Lee | Finance | 72000.00 | 2019-11-20 |
| 104 | David | Kumar | IT | 68000.00 | 2022-01-10 |

- **ALTER**: Modifies an existing database object.

**Add a new coloumn**

```sql
ALTER TABLE Students
ADD Gender CHAR(1);
```

Now your table becomes:

| StudentID | FirstName | LastName | Age | Gender |
|---|---|---|---|---|
| 1 | John | Doe | 20 | |
| 2 | Alice | Smith | 22 | |

- **DROP**: Deletes an existing database object.

```sql
ALTER TABLE Students
DROP COLUMN Gender;
```

```sql
-- DDL
CREATE TABLE Students (ID INT, Name VARCHAR(50), Age INT);

-- DML
INSERT INTO Students VALUES (1, 'Alice', 20);
INSERT INTO Students VALUES (2, 'Bob', 19);

-- DQL
SELECT * FROM Students WHERE Age > 18;

-- TCL
SAVEPOINT before_update;
UPDATE Students SET Age = 18 WHERE ID = 2;
ROLLBACK TO before_update;

-- DCL
GRANT SELECT ON Students TO user1;
```

**Data Types in SQL**

SQL supports a wide range of data types, which can be categorized as follows:

- **Numeric Data Types**:
  - **INT**: Integer (whole numbers).
  - **DECIMAL(p,s)**: Fixed-point numbers (precision p, scale s).

- **FLOAT**: Floating-point numbers (approximate precision).
- **DOUBLE**: Double precision floating-point number.

- **Character/String Data Types**:
  - **CHAR(n)**: Fixed-length string of length n.
  - **VARCHAR(n)**: Variable-length string with maximum length n.
  - **TEXT**: Large text strings.

- **Date/Time Data Types**:
  - **DATE**: Date (year, month, day).
  - **TIME**: Time of day (hours, minutes, seconds).
  - **DATETIME**: Date and time.

- **Boolean**:
  - **BOOLEAN**: Represents TRUE or FALSE values.

## SELECT Statement with Various Operators

### 📌 Basic Syntax

```sql
SELECT column1, column2 FROM table_name WHERE condition;
```

The SELECT statement is used to retrieve data from a database. You can use different operators to filter and manipulate the data:

- **Comparison Operators**:
  - =: Equal to
  - != or <>: Not equal to
  - >: Greater than
  - <: Less than
  - >=: Greater than or equal to
  - <=: Less than or equal to

- ❖ BETWEEN: Within a range.
- ❖ IN: Matches any of the values in a list.
- ❖ LIKE: Searches for a specified pattern.
- ❖ IS NULL: Checks for null values.

### 📌 Examples

```sql
SELECT * FROM employees WHERE salary > 50000;
SELECT name FROM students WHERE grade IN ('A', 'B');
SELECT * FROM products WHERE name LIKE 'Sam%';
```

### 📌 Syntax:

```sql
SELECT column1, column2
FROM table_name
WHERE condition;
```

- **Logical Operators**:
  - ❖ AND: Combines multiple conditions (all must be true).
  - ❖ OR: Combines multiple conditions (at least one must be true).
  - ❖ NOT: Negates a condition.
- **Ordering and Limiting Results**:
  - ❖ **ORDER BY: Sorts the results by one or more columns.**

### 📌 Operators:

| Operator | Example | Use |
|---|---|---|
| `=` | `age = 25` | Equals |
| `!=` or `<>` | `name != 'John'` | Not equal |
| `<, >, <=, >=` | `salary > 50000` | Comparisons |
| `BETWEEN` | `age BETWEEN 20 AND 30` | Range |
| `IN` | `department IN ('HR', 'Sales')` | Set of values |
| `LIKE` | `name LIKE 'A%'` | Pattern matching |
| `IS NULL` | `email IS NULL` | Null check |
| `AND`, `OR`, `NOT` | Combines conditions | |

### 📌 Example Table: `Employees`

| ID | Name | Age | Salary | Dept |
|---|---|---|---|---|
| 1 | Alice | 30 | 60000 | HR |
| 2 | Bob | 25 | 50000 | Sales |
| 3 | Charlie | 28 | 70000 | IT |

### 📌 Example Query:

sql Copy Edit

```sql
SELECT name, salary FROM Employees WHERE salary > 55000;
```

**5. Single Row and Multi-Row Functions**

- **Single Row Functions**: Operate on **a single row at a time**, returning a single result.
  - ❖ **String Functions**: UPPER(), LOWER(), CONCAT(), LENGTH().
  - ❖ **Numeric Functions**: ROUND(), CEIL(), FLOOR(), ABS().
  - ❖ **Date Functions**: CURRENT_DATE(), DATEADD(), DATEDIFF().
- **1. `CURRENT_DATE()` (or `GETDATE()` in SQL Server)**

- ✅ **What it does:**
- Returns the **current system date** (without time).

## 📘 Syntax:

```sql
SELECT CURRENT_DATE;        -- MySQL / PostgreSQL
SELECT GETDATE();           -- SQL Server (returns date + time)
```

## 🧪 Example Output:

| CURRENT_DATE |
| --- |
| 2025-04-08 |

### 📌 Operates on one row and returns one result per row.

| Type | Function | Example |
| --- | --- | --- |
| String | `UPPER()`, `LOWER()`, `LENGTH()`, `SUBSTR()` | `UPPER('hello')` → `HELLO` |
| Numeric | `ROUND()`, `TRUNC()`, `MOD()` | `ROUND(123.456, 2)` → `123.46` |
| Date | `SYSDATE`, `NOW()`, `DATEDIFF()` | `DATEDIFF(NOW(), '2024-01-01')` |

## 📘 Example Table: Employees

| EmpID | Name | Salary | JoinDate |
| --- | --- | --- | --- |
| 1 | alice | 25000 | 2021-01-15 |
| 2 | bob | 30000 | 2020-06-10 |
| 3 | charlie | 27000 | 2022-03-01 |

### 💡 Example Query 1 (String Function):

```sql
SELECT UPPER(Name) AS Upper_Name FROM Employees;
```

### 🔽 Output:

| Upper_Name |
| --- |
| ALICE |
| BOB |
| CHARLIE |

### 💡 Example Query 2 (Numeric Function):

```sql
SELECT Salary, ROUND(Salary * 0.1, 0) AS Bonus FROM Employees;
```

### 🔽 Output:

| Salary | Bonus |
| --- | --- |
| 25000 | 2500 |
| 30000 | 3000 |
| 27000 | 2700 |

- **Multi-Row Functions**: Operate on **multiple rows and return a single** result, often used with grouping.
    - ❖ **Aggregate Functions**: COUNT(), SUM(), AVG(), MIN(), MAX().

| Function | Use |
| --- | --- |
| `COUNT()` | Number of rows |
| `SUM()` | Total value |
| `AVG()` | Average value |
| `MIN()` / `MAX()` | Minimum / Maximum value |

### 💡 Example Query:

sql Copy

```sql
SELECT COUNT(*) AS Total_Employees, AVG(Salary) AS Average_Salary FROM Employees;
```

🔽 **Output:**

| Total_Employees | Average_Salary |
| --- | --- |
| 3 | 27333.33 |

### 💡 With GROUP BY:

```sql
SELECT Department, COUNT(*) AS Emp_Count
FROM Employees
GROUP BY Department;
```

(Assume `Department` column exists)

🔽 **Output Example:**

| Department | Emp_Count |
|---|---|
| HR | 2 |
| IT | 3 |

## 🆚 Single-row vs Multi-row Functions

| Feature | Single-Row Function | Multi-Row Function (Group) |
|---|---|---|
| Works on | One row at a time | Group of rows |
| Returns | One result per row | One result per group |
| Used with | SELECT, WHERE | SELECT + GROUP BY / HAVING |
| Examples | UPPER(), ROUND(), LENGTH() | COUNT(), AVG(), MAX() |

### 6. GROUP BY and HAVING Clauses

- **GROUP BY Clause**: Used to group rows with the same values in specified columns, often used with aggregate functions. Used to group rows that have the same values in specified columns.

## 📘 Syntax:

```sql
SELECT column1, aggregate_function(column2)
FROM table
GROUP BY column1;
```

**Example:**

Assume we have a table called Sales:

| Product | Category | Amount |
|---|---|---|
| Pen | Stationery | 10 |
| Pencil | Stationery | 5 |
| Eraser | Stationery | 2 |
| Soap | Toiletries | 15 |
| Shampoo | Toiletries | 20 |

### Query:

```sql
SELECT Category, SUM(Amount) AS Total_Sales
FROM Sales
GROUP BY Category;
```

## 📊 Result:

| Category | Total_Sales |
|---|---|
| Stationery | 17 |
| Toiletries | 35 |

👉 This groups the sales by Category and calculates the total Amount per category.

- **HAVING Clause**: Used to filter groups after the GROUP BY clause has been applied (acts like a WHERE clause for groups). Used to filter groups based on aggregate results. Works **after** GROUP BY.
- The GROUP BY clause groups data by one or more columns, and the HAVING clause filters the result of the GROUP BY based on a condition.

### 📘 Syntax:

```sql
SELECT column1, aggregate_function(column2)
FROM table
GROUP BY column1
HAVING aggregate_function(column2) condition;
```

### 🧪 Example:

```sql
SELECT Category, SUM(Amount) AS Total_Sales
FROM Sales
GROUP BY Category
HAVING SUM(Amount) > 20;
```

### 📊 Result:

| Category | Total_Sales |
|---|---|
| Toiletries | 35 |

👉 The HAVING clause filters out the Stationery group because its total sales (17) is less than 20.

**Key Differences: WHERE vs HAVING**

| Feature | WHERE Clause | HAVING Clause |
|---|---|---|
| Used on | Individual rows | Grouped rows |
| Used with | SELECT, UPDATE, DELETE | GROUP BY (mostly) |
| Applies | Before grouping | After grouping |

| Topic | Key Points |
|---|---|
| Need for SQL | Essential for managing relational databases |
| SQL Types | DDL, DML, DQL, DCL, TCL |
| Data Types | Numeric, String, Date/Time, Others |
| SELECT + Operators | Used to filter and retrieve specific data |
| Single Row Functions | Operate on individual rows |
| Multi Row Functions | Operate on groups of rows |
| GROUP BY + HAVING | Used for aggregation and filtering groups |

**Offline Platforms**

**These are software or applications you install on your computer.**

| Platform | Description |
| --- | --- |
| **MySQL Workbench** | GUI tool for MySQL databases (free, popular) |
| **Oracle SQL Developer** | For working with Oracle DB (supports PL/SQL) |
| **Microsoft SQL Server Management Studio (SSMS)** | For Microsoft SQL Server |
| **pgAdmin** | GUI for PostgreSQL |
| **SQLite Studio** | Lightweight, good for learning and small projects |
| **DBeaver** | Universal SQL client supporting MySQL, PostgreSQL, Oracle, etc. |
| **XAMPP** | Includes MySQL + phpMyAdmin for local web development |

## 🌐 2. Online Platforms

These are websites where you can practice SQL directly in your browser — no installation needed!

| Platform | Highlights |
| --- | --- |
| SQL Fiddle | Test small SQL queries with sample schema |
| W3Schools SQL Tryit | Beginner-friendly |
| LeetCode SQL | Practice with real problems |
| HackerRank SQL | Challenges + certificates |
| Mode SQL Tutorial | Interactive learning |
| DB Fiddle | Supports PostgreSQL, MySQL, SQLite |
| Replit | You can create PostgreSQL projects online |