1. Introduction to SQL

What is SQL?

SQL (Structured Query Language) is a standard programming language designed for managing and manipulating databases. It allows users to perform various operations such as retrieving, inserting, updating, and deleting data in a database.

Importance and Applications of SQL

- **Data Management**: SQL helps in managing large volumes of data efficiently.
- **Querying Data**: SQL allows you to retrieve specific information using simple commands.
- **Data Analysis**: Widely used for data analysis in business intelligence tools.
- **Application Development**: Backend of many web and mobile applications.
- **Standardized Language**: SQL is used across various database management systems.

Database vs. DBMS

- **Database**: A collection of organized data stored electronically.
- **DBMS (Database Management System)**: Software used to manage and interact with databases (e.g., MySQL, PostgreSQL).

Overview of RDBMS

RDBMS (Relational Database Management System) stores data in tables with rows and columns. Relationships between tables are defined using keys.

Popular RDBMS Software

- **MySQL**: Open-source and widely used.
- **PostgreSQL**: Known for advanced features and extensibility.
- **Oracle**: Used in enterprise environments.
- **SQL Server**: Microsoft's relational database solution.

---

2. SQL Basics

SQL Syntax and Structure

- Commands are written in English-like statements.
- End each command with a semicolon `;`.
- Keywords (e.g., SELECT, INSERT) are case-insensitive but typically written in uppercase.

Data Types in SQL

- **INT**: Integer numbers.
- **VARCHAR**: Variable-length text.
- **DATE**: Stores dates.
- **FLOAT**: Decimal numbers.

Understanding Tables, Rows, and Columns

- **Table**: A collection of related data, organized in rows and columns.
- **Row**: A single record in the table.
- **Column**: A field in the table, representing one attribute of the data.

Creating and Managing Databases

- **`CREATE DATABASE`**: Creates a new database.

  ```
  CREATE DATABASE SchoolDB;
  ```

- **`USE DATABASE`**: Selects the database to work with.

  ```
  USE SchoolDB;
  ```

---

3. Data Definition Language (DDL)

Creating Tables: `CREATE TABLE`

Used to define the structure of a table.

```
CREATE TABLE Students (
  StudentID INT PRIMARY KEY,
  Name VARCHAR(100),
  Age INT,
  Grade VARCHAR(10)
);
```

Altering Tables: `ALTER TABLE`

Used to modify an existing table's structure.

- Add a column:

  ```
  ALTER TABLE Students ADD Email VARCHAR(100);
  ```

- Drop a column:

  ```
  ALTER TABLE Students DROP COLUMN Email;
  ```

Dropping Tables: `DROP TABLE`

Used to delete an entire table and its data.

```
DROP TABLE Students;
```

Constraints

Constraints enforce rules on data in tables.

- **Primary Key**: Uniquely identifies each row.

  ```
  PRIMARY KEY (StudentID)
  ```

- **Foreign Key**: Ensures relationships between tables.

```
FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
```

- **Unique**: Ensures values in a column are unique.

  ```
  UNIQUE (Email)
  ```

- **Check**: Restricts the values in a column.

  ```
  CHECK (Age >= 18)
  ```

- **Default**: Assigns a default value.

  ```
  DEFAULT 'Not Assigned'
  ```

- **Not Null**: Ensures a column cannot have NULL values.

  ```
  NOT NULL
  ```

---

4. Data Manipulation Language (DML)

Inserting Data: `INSERT INTO`

Adds new records to a table.

```
INSERT INTO Students (StudentID, Name, Age, Grade) VALUES (1, 'Alice', 20, 'A');
```

Updating Data: `UPDATE`

Modifies existing data in a table.

```
UPDATE Students SET Grade = 'B' WHERE StudentID = 1;
```

Deleting Data: `DELETE`

Removes records from a table.

```
DELETE FROM Students WHERE StudentID = 1;
```

Retrieving Data: `SELECT`

Used to fetch data from a table.

- **Basic `SELECT` Query**:

  ```
  SELECT * FROM Students;
  ```

- **Filtering with `WHERE`**:

  ```
  SELECT * FROM Students WHERE Age > 18;
  ```

- **Comparison Operators**:
  - `=`: Equal to.
  - `!=`: Not equal to.
  - `>` / `<`: Greater than / Less than.
  - `LIKE`: Pattern matching.

- `BETWEEN`: Range values.
- `IN`: Matches any value in a list.

```
SELECT * FROM Students WHERE Grade IN ('A', 'B');
```

Sorting Results: `ORDER BY`

Sorts the result set in ascending (default) or descending order.

```
SELECT * FROM Students ORDER BY Name ASC;
```