**What is SQL?**

SQL (Structured Query Language) is a standard Database language designed for managing and manipulating relational databases. It allows users to create, retrieve, update, and delete data within a database. SQL is highly versatile and used across different database systems, including MySQL, PostgreSQL, Oracle, SQL Server, and SQLite.

SQL works with tables, where data is stored in rows and columns. Its syntax is simple and English-like, making it easy to understand and use.

### What is a Database?

A **database** is an organized collection of structured information or data, typically stored electronically in a computer system. It enables efficient data management, retrieval, and manipulation. Databases are designed to handle large amounts of data while ensuring data integrity and security.

**Purpose of SQL**

The primary purpose of SQL is to interact with databases to perform various operations like:

1. **Data Querying**: Retrieve specific data using SELECT statements.
2. **Data Manipulation**: Insert, update, delete, and alter data in a database.
3. **Data Definition**: Create or modify the structure of database objects such as tables, indexes, and views.
4. **Access Control**: Manage access permissions and ensure data security using SQL commands.
5. **Transaction Control**: Manage transactions and ensure the integrity of data during multiple operations.

**Who Developed SQL and When Was It Developed?**

SQL (Structured Query Language) was developed in the early 1970s by **IBM** researchers **Donald D. Chamberlin** and **Raymond F. Boyce**. They were part of the team working on IBM’s pioneering database project known as **System R**, which was designed to implement a relational database management system (RDBMS).

SQL was initially called **SEQUEL** (Structured English Query Language), which was inspired by the **relational model of data** proposed by **Edgar F. Codd** in 1970. Codd’s work on the relational model revolutionized the way databases were designed and managed, enabling the structured organization of data using tables and relationships. SEQUEL was later renamed SQL due to trademark issues.

**Timeline of SQL Development:**

* **1970**: Edgar F. Codd publishes a paper introducing the relational model of data.
* **1973–1974**: IBM researchers Chamberlin and Boyce develop SEQUEL, which was later renamed SQL.
* **1979**: Oracle Corporation (then known as Relational Software Inc.) releases the first commercial SQL-based RDBMS, Oracle V2.
* **1986**: SQL becomes a standard when the **American National Standards Institute (ANSI)** adopts SQL as the standard relational database query language.
* **1987**: SQL is adopted as an international standard by the **International Organization for Standardization (ISO)**.

**Who Should Learn SQL?**

* **Developers**: Both backend and full-stack developers need SQL to interact with databases in web applications.
* **Data Analysts**: SQL is essential for querying data and extracting insights for business intelligence.
* **Database Administrators**: They need SQL to manage databases, ensure data integrity, and optimize performance.
* **Data Scientists**: SQL is often used to clean and prepare large datasets for analysis.
* **System Administrators**: For managing database servers and tuning performance.

In general, anyone who works with data or systems involving relational databases will benefit from learning SQL.

### ****What is DBMS (Database Management System)?****

A **Database Management System (DBMS)** is software that allows users to **create, manage, and interact with databases**. It provides a systematic and organized way to store, retrieve, update, and manage data. A DBMS ensures that the data is consistently organized and remains easily accessible.

The DBMS acts as an intermediary between the user and the database, ensuring that the data is stored safely, can be retrieved efficiently, and can be manipulated as needed by various applications. It also provides security, data integrity, and backup/recovery features.

#### **Key Functions of a DBMS:**

1. **Data Definition**: Helps define the structure of the data (schema) and the relationships between different data entities.
2. **Data Manipulation**: Allows users to query, update, and delete data from the database.
3. **Data Security**: Ensures that only authorized users can access or modify the database.
4. **Backup and Recovery**: Ensures data is safe from accidental loss or system failures, and provides a way to restore it.
5. **Concurrency Control**: Manages simultaneous data access to ensure that multiple users can interact with the database without conflicts.
6. **Data Integrity**: Ensures that the data remains accurate and consistent throughout its lifecycle.

**Basic SQL Commands used till now:**

### 1. Show Databases

* **Syntax**:

SHOW DATABASES;

* **Usage**: Displays all databases on the server.
* **Explanation**: Use this command to see which databases are available for you to work with.

### 2. Use Database

* **Syntax**:

USE database\_name;

* **Usage**: Selects a specific database to work with.
* **Explanation**: Before performing operations, specify the database context. Replace database\_name with the actual name of your database.

### 3. Describe Structure of the Table

* **Syntax**:

DESC table\_name;

* **Example**:

DESC student;

* **Usage**: Describes the structure of the specified table.
* **Explanation**: This command shows details about the columns in the table, including data types and constraints.

### 4. Show Tables

* **Syntax**:

SHOW TABLES;

* **Usage**: Lists all tables in the currently selected database.
* **Explanation**: This helps you understand the structure of your database and the available tables.

### 5. Create Table

* **Syntax**:

CREATE TABLE table\_name (

column1 datatype constraints,

column2 datatype constraints,

...

);

* **Example**:

CREATE TABLE student (

id INT PRIMARY KEY,

name VARCHAR(20),

city VARCHAR(20),

college VARCHAR(20)

);

* **Usage**: Creates a new table named student.
* **Explanation**: Defines the structure for storing data. The id column is specified as the primary key, ensuring unique records.

### 6. Insert Into Table

* **Syntax**:

INSERT INTO table\_name (column1, column2, column3) VALUES (value1, value2, value3);

* **Example**:

INSERT INTO student (id, name, city, college) VALUES (101, 'Amarjeet Kumar Singh', 'Bihar', 'IES College Bhopal');

* **Usage**: Adds a new record to the student table.
* **Explanation**: Populate your table with data. Use single quotes for string values. Ensure that the primary key (id) is unique.

### 7. Select From Table

* **Syntax**:

SELECT column1, column2, column3 FROM table\_name;

* **Example**:

SELECT name FROM student; -- Selecting a specific column

Or to select all columns:

SELECT \* FROM student;

* **Usage**: Retrieves records from the student table.
* **Explanation**:
  + SELECT: Specifies which columns to retrieve. Use \* to select all columns.
  + FROM: Indicates the table from which to retrieve the data.

### SQL Comments

**Definition:** Comments in SQL are used to annotate or explain sections of SQL code, making it easier to understand. They are ignored by the SQL engine during execution.

### Why Use Comments?

1. **Documentation**: Helps in explaining the purpose and functionality of SQL queries.
2. **Readability**: Makes complex queries easier to read and understand for others (or yourself in the future).
3. **Debugging**: Allows developers to temporarily disable parts of the code without deleting them.

### Types of Comments in SQL

#### 1. Single-line Comments

* **Syntax**: Use -- (double dash).
* **Example**:

-- This is a single-line comment

SELECT \* FROM Employees; -- This retrieves all records from Employees table

* **When to Use**:
  + For brief explanations or notes.
  + When annotating a specific line of code.

#### 2. Multi-line Comments

* **Syntax**: Use /\* ... \*/ (slash and asterisk).
* **Example**:

/\* This is a multi-line comment

It can span multiple lines \*/

SELECT \* FROM Employees;

* **When to Use**:
  + For detailed explanations or documentation.
  + When comments span multiple lines.
  + To describe complex queries or logic.
  + To temporarily disable blocks of code for debugging.

### Summary

* **Single-line Comments**: Quick notes for single lines.
* **Multi-line Comments**: Detailed explanations for longer comments or multiple lines.