Module 4 - Database

## **.1) What is RDBMS?**

**RDBMS (Relational Database Management System)** is a type of database management system based on the **relational model**. Data is stored in **tables (relations)** made up of **rows and columns**, where:

* **Rows** represent records.
* **Columns** represent attributes (fields).
* Each table has a **primary key** to uniquely identify rows.
* Tables can be linked using **foreign keys** to establish relationships.

**Examples**: MySQL, Oracle, PostgreSQL, SQL Server.

**Key Features:**

* Data integrity and consistency.
* ACID compliance (Atomicity, Consistency, Isolation, Durability).
* Supports **SQL** for managing data.

## 

**2) What is SQL?**

**SQL (Structured Query Language)** is a standard programming language used to **manage and manipulate relational databases**.

### **SQL can be divided into:**

* **DDL (Data Definition Language)** – CREATE, ALTER, DROP
* **DML (Data Manipulation Language)** – SELECT, INSERT, UPDATE, DELETE
* **DCL (Data Control Language)** – GRANT, REVOKE
* **TCL (Transaction Control Language)** – COMMIT, ROLLBACK, SAVEPOINT

### **3. Write SQL Commands**

SQL is made up of various types of commands, each serving a specific purpose in interacting with a relational database. These commands can be broadly classified into **five main categories**:

#### **DDL (Data Definition Language)**

Used to define and modify database structures such as tables and schemas.

* CREATE: Creates a new table or database.
* ALTER: Modifies an existing table (e.g., add/remove a column).
* DROP: Deletes a table or database.
* TRUNCATE: Deletes all data from a table but keeps the structure.

#### **DML (Data Manipulation Language)**

Used to manipulate data stored in the database.

* INSERT: Adds new records.
* UPDATE: Modifies existing records.
* DELETE: Removes specific records.

#### 

#### **DQL (Data Query Language)**

Used to retrieve data from tables.

* SELECT: Fetches data from one or more tables.

### 

### **4. What is JOIN?**

A **JOIN** in SQL is used to retrieve data from **two or more related tables**. When data is normalized in a relational database, it's split into multiple tables to reduce redundancy and improve consistency. JOIN operations help bring related data back together based on **common columns** (usually keys).

For example, you might have:

#### **Types of JOINs:**

* **INNER JOIN**: Returns records that have matching values in both tables.
* **LEFT JOIN (LEFT OUTER JOIN)**: Returns all records from the left table, and matched records from the right.
* **RIGHT JOIN (RIGHT OUTER JOIN)**: Returns all records from the right table, and matched ones from the left.
* **FULL JOIN (FULL OUTER JOIN)**: Returns all records when there's a match in either table.
* **CROSS JOIN**: Returns the Cartesian product (every combination of rows).

5. **Types of JOINs in SQL**

#### **1. INNER JOIN**

**Returns only rows that have matching values in both tables. It excludes non-matching rows.**

**2. LEFT JOIN (or LEFT OUTER JOIN)**

Returns all rows from the left table, even if there are no matches in the right table. If no match exists, NULLs appear for the right table’s columns.

#### **3. RIGHT JOIN (or RIGHT OUTER JOIN)**

Returns all rows from the right table, and matching rows from the left. If no match exists, NULLs appear for the left table’s columns.

#### 

#### **4. FULL OUTER JOIN**

Combines results of LEFT and RIGHT joins. Returns all records when there is a match in one of the tables.  
 Note: Not supported in some databases like MySQL unless using a UNION workaround.

#### **5. CROSS JOIN**

Returns the Cartesian product of both tables. Every row in the first table is matched with every row in the second. Used rarely and only in specific cases.

### **6. SQL Constraints**

SQL constraints define rules applied to columns or tables to maintain **data integrity**, **accuracy**, and **consistency**. They restrict the type of data that can be inserted, preventing invalid or duplicate entries.

#### **Types of SQL Constraints:**

1. **NOT NULL** – Ensures that a column cannot have a NULL value.

2. **UNIQUE** – Ensures that all values in a column are different.

3. **PRIMARY KEY** – Uniquely identifies each row; combines NOT NULL and UNIQUE.

4. **FOREIGN KEY** – Creates a relationship between two tables by referencing the primary key of another table.

5. **CHECK** – Ensures that a value satisfies a specific condition.

6. **DEFAULT** – Assigns a default value if none is provided during data insertion.

### **7. Difference Between RDBMS and DBMS**

| **Feature** | **DBMS** | **RDBMS** |
| --- | --- | --- |
| **Definition** | Software for managing data | Extension of DBMS using relational model |
| **Storage Format** | Files, records | Tables (rows and columns) |
| **Relationships** | Not supported | Supports relationships via foreign keys |
| **Normalization** | Not supported | Supported to eliminate redundancy |
| **Data Integrity** | Low | High via constraints and keys |
| **Multi-user** | Often single-user | Designed for multi-user |
| **Examples** | XML, CSV, MS Access | MySQL, Oracle, PostgreSQL, SQL Server |

In summary, **RDBMS** is more powerful, structured, and reliable than a basic **DBMS**, and it's used in enterprise applications.

### **✅ 8. Full Form of .ipa and .apk**

* **.ipa** – *iOS App Store Package*. It is the file format used to install applications on Apple devices such as iPhones and iPads. .ipa files are compiled and signed with an Apple developer certificate and contain binaries, resources, and metadata.
* **.apk** – *Android Package Kit*. This is the package file format used by Android OS to distribute and install apps. .apk files include compiled Java/Kotlin code, XML files, resources, and a manifest file.

These files represent the final form of mobile apps ready to be installed on real devices or emulators.

### **✅ 9. How to Enable Developer Options on Android**

Developer options provide advanced settings for developers to test, debug, and optimize Android applications.

**Steps to enable Developer Options:**

1. Go to **Settings > About Phone**.
2. Locate the **Build Number** (under “Software Information” or “Version”).
3. Tap **Build Number** **7 times** quickly.
4. You’ll get a message: *“You are now a developer!”*
5. Go back to **Settings**, and you'll see **Developer Options** enabled.

**Features available:**

* **USB Debugging**: Required for Android Studio or ADB access.
* **Show layout bounds**: Helps visualize UI layout.
* **Force GPU Rendering**: Improves performance in certain apps.
* **Animation scale**: Speeds up or slows down transitions.

It’s useful for testing and debugging apps during development.

### **✅ 10. What is an SQL Alias?**

An **alias** in SQL is a temporary name assigned to a table or column for the duration of a query. It improves readability, especially in complex joins or calculations.

Aliases are created using the AS keyword (optional):

Benefits:

* Shortens long or complex table/column names.
* Makes output easier to understand.
* Required in some aggregate or subquery operations.

### **11. Write a Query to Create a Table in SQL**

Creating a table is one of the first steps in designing a relational database. A table consists of columns (fields) and rows (records). In SQL, the CREATE TABLE statement defines the structure of the table, including column names, data types, and constraints.

**Example:**

sql

CopyEdit

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

age INT CHECK (age >= 0),

email VARCHAR(100) UNIQUE,

city VARCHAR(100),

enrollment\_date DATE DEFAULT CURRENT\_DATE

);

**Explanation of key parts:**

* student\_id: An integer used as the primary key to uniquely identify each student.
* name: A string of up to 100 characters that cannot be NULL.
* age: A positive integer (validated with a CHECK constraint).
* email: Must be unique for each student.
* enrollment\_date: Automatically defaults to the current date.

Tables form the backbone of a relational database, and proper table creation ensures structured, efficient, and reliable data storage.

### **✅ 12. Write a Query to Insert Data into a Table**

The INSERT INTO statement is used to add new records (rows) into a table. You must match the values provided with the table’s structure.

**Syntax:**

sql

CopyEdit

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

**Example:**

sql

CopyEdit

INSERT INTO Students (student\_id, name, age, email, city)

VALUES (1, 'Alice Johnson', 20, 'alice@example.com', 'New York');

You can also insert multiple records at once:

sql

CopyEdit

INSERT INTO Students (student\_id, name, age, email, city)

VALUES

(2, 'Bob Smith', 22, 'bob@example.com', 'Chicago'),

(3, 'Carol White', 19, 'carol@example.com', 'Boston');

Always ensure data integrity by respecting constraints like NOT NULL, UNIQUE, and FOREIGN KEY.

### **✅ 13. Write a Query to Update Data with Validations**

The UPDATE statement modifies existing records in a table. It is essential to use the WHERE clause to avoid changing all records unintentionally.

**Example:**

sql

CopyEdit

UPDATE Students

SET city = 'Los Angeles'

WHERE student\_id = 2 AND age >= 20;

**Explanation:**

* The SET clause updates the city to "Los Angeles".
* The WHERE clause ensures the change only affects students who meet both conditions: student\_id = 2 and age >= 20.

**Without a WHERE clause**, all records in the table would be updated, which could lead to data loss.

### **✅ 14. Write a Query to Delete Data with Validations**

The DELETE statement removes records from a table. Like UPDATE, it must be used with a WHERE clause to target specific rows.

**Example:**

sql

CopyEdit

DELETE FROM Students

WHERE age < 18;

This command deletes all students who are younger than 18.

To delete all data from a table:

sql

CopyEdit

DELETE FROM Students;

But **this is dangerous** unless you’re sure. An alternative to remove all records while resetting the identity (ID) column is:

sql

CopyEdit

TRUNCATE TABLE Students;

(Note: TRUNCATE is faster but cannot be rolled back in many databases.)

### **✅ 15. Write a Query to Insert a New Column in an Existing Table**

To modify the structure of a table and add a new column, use the ALTER TABLE command.

**Example:**

sql

CopyEdit

ALTER TABLE Students

ADD phone\_number VARCHAR(15);

This adds a new column called phone\_number that can store up to 15 characters. You can also add constraints to the new column:

sql

CopyEdit

ALTER TABLE Students

ADD status VARCHAR(10) DEFAULT 'active' NOT NULL;

After adding, you may update or insert values in this new column using UPDATE or INSERT.

### **✅ 16. Write a Query to Drop a Table and Database**

To permanently delete a table or an entire database, use the DROP command.

**Drop a table:**

sql

CopyEdit

DROP TABLE Students;

**Drop a database:**

sql

CopyEdit

DROP DATABASE SchoolDB;

**Caution:** DROP is irreversible and removes all associated data and structure. Use it carefully, typically in development or testing environments.

### **✅ 17. Write a Query to Find MAX and MIN Values from a Table**

To analyze data and get the highest or lowest value in a column, use SQL aggregate functions MAX() and MIN().

**Example:**

sql

CopyEdit

SELECT MAX(age) AS Oldest, MIN(age) AS Youngest FROM Students;

**Explanation:**

* MAX(age): Returns the oldest student's age.
* MIN(age): Returns the youngest student's age.

You can also apply conditions:

sql

CopyEdit

SELECT MAX(age) FROM Students WHERE city = 'New York';

These queries are useful in reports, statistics, and dashboards.

### **✅ 18. Create Seller and Product Tables and Apply Foreign Key**

To define relationships between tables, you can use **foreign keys**. This is common when you have one-to-many relationships, like one seller can have many products.

**Create Seller table:**

sql

CopyEdit

CREATE TABLE Seller (

seller\_id INT PRIMARY KEY,

seller\_name VARCHAR(100) NOT NULL

);

**Create Product table with foreign key:**

sql

CopyEdit

CREATE TABLE Product (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2),

seller\_id INT,

FOREIGN KEY (seller\_id) REFERENCES Seller(seller\_id)

);

**INNER JOIN to fetch data from both:**

sql

CopyEdit

SELECT p.product\_name, p.price, s.seller\_name

FROM Product p

INNER JOIN Seller s ON p.seller\_id = s.seller\_id;

This query combines product and seller data based on matching seller IDs.

### **✅ 19. What is API Testing?**

**API Testing** focuses on validating **Application Programming Interfaces (APIs)** directly. Unlike UI testing, which tests graphical interfaces, API testing checks endpoints that connect different software systems or layers.

APIs process requests and return data. API testing checks:

* Correct **response codes** (200 OK, 404 Not Found, etc.)
* Accurate **data** returned
* Proper **authentication and authorization**
* **Performance** under load
* **Error handling**

Popular tools include **Postman**, **SoapUI**, **REST Assured**, and **JMeter**.

API testing is critical in microservices, mobile apps, and any system where front-end and back-end are separated. It's faster, more stable, and can be automated easily.

### **✅ 20. Types of API Testing**

API Testing can be divided into several types:

1. **Functional Testing** – Ensures API performs its intended functions correctly.
2. **Load Testing** – Checks API performance under high traffic.
3. **Security Testing** – Validates authorization, access control, and encryption.
4. **Validation Testing** – Ensures data format, values, and logic are correct.
5. **Error Handling Testing** – Confirms meaningful error responses (e.g., 400 Bad Request).
6. **Integration Testing** – Verifies how different APIs or modules work together.
7. **Unit Testing** – Performed by developers to test individual functions or methods.

Thorough API testing ensures reliable and secure communication between software systems.

### **✅ 21. What is Responsive Testing?**

**Responsive Testing** ensures that a website or application **adjusts its layout and elements** based on different screen sizes and device types. The goal is to maintain usability and aesthetics across **smartphones, tablets, laptops, and desktops**.

Modern web design uses:

* **CSS media queries**
* **Fluid grids and flexible images**
* **Viewport meta tags**

Responsive Testing checks:

* Whether the layout breaks on small screens.
* Button size, font readability, scroll behavior.
* Cross-browser and cross-platform compatibility.

Tools like Chrome DevTools, BrowserStack, and Responsinator are widely used to simulate various screen resolutions and devices.

It’s crucial for improving user experience and SEO, as mobile usability is now a ranking factor.

### **✅ 22. Tools Available for Responsive Testing**

Responsive Testing tools help developers and testers verify how websites and apps perform on various screen sizes and devices.

#### **Popular Tools:**

1. **Chrome DevTools** – Built-in browser tool to simulate phones, tablets, and custom resolutions.
2. **BrowserStack** – Cloud-based testing on real mobile and desktop devices.
3. **Responsinator** – Simple tool to preview your site on different devices.
4. **Sizzy** – A browser made for responsive design testing.
5. **Percy** – Visual testing tool that highlights layout differences across screens.
6. **LambdaTest** – Live and automated testing across browsers and resolutions.