1. **What is RDBMS**

**Ans:** An **RDBMS (Relational Database Management System)** is a type of database management system that stores and manages data in the form of **tables** (also called relations).

Each table consists of **rows** (records/tuples) and **columns** (fields/attributes). The key idea of an RDBMS is that data can be related to other data using **keys** (primary key, foreign key).

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

1. **What is SQL**

**Ans: SQL (Structured Query Language)** is a **standard programming language** used to communicate with and manage data in a **relational database (RDBMS)**.

It is used to **store, retrieve, update, and delete** data, as well as to define and control the structure of databases.

1. **Write SQL Commands**

**1.DDL (Data Definition Language)**

Used to define and manage database structure.

* **CREATE** → Create database objects (table, database, etc.)

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50),

Age INT

);

* **ALTER** → Modify an existing table

ALTER TABLE Students ADD Email VARCHAR(50);

* **DROP** → Delete a table or database

DROP TABLE Students;

* **TRUNCATE** → Remove all rows from a table (structure remains)

TRUNCATE TABLE Students;

**2.DML (Data Manipulation Language)**

Used to modify data inside tables.

* **INSERT** → Add new records

INSERT INTO Students VALUES (1, 'Nisha', 21, 'nisha@mail.com');

* **UPDATE** → Modify existing records

UPDATE Students SET Age = 22 WHERE StudentID = 1;

* **DELETE** → Remove records

DELETE FROM Students WHERE StudentID = 1;

1. **What is join?**

**Ans:** A **JOIN** in SQL is used to **combine data from two or more tables** based on a **related column** between them (usually a **primary key** in one table and a **foreign key** in another).

### Why use JOIN?

Because data in RDBMS is usually stored in **multiple related tables**, JOIN helps fetch meaningful combined information.

1. **Write type of joins.**

## Ans: ****Types of Joins in SQL****

1. **INNER JOIN**
   * Returns only rows with **matching values** in both tables.  
     Example: Students enrolled in courses.
2. **LEFT JOIN (LEFT OUTER JOIN)**
   * Returns **all rows from the left table**, and matched rows from the right table.
   * If no match → shows NULL for right table columns.
3. **RIGHT JOIN (RIGHT OUTER JOIN)**
   * Returns **all rows from the right table**, and matched rows from the left table.
   * If no match → shows NULL for left table columns.
4. **FULL JOIN (FULL OUTER JOIN)**
   * Returns **all rows from both tables**, matched or not.
   * Unmatched rows show NULL on one side.
5. **CROSS JOIN**
   * Returns the **Cartesian product** of two tables (every row of one with every row of the other).
6. **SELF JOIN**
   * A table joins **with itself** (useful for hierarchical data like employees & managers).
7. **How Many constraint and describes it self**

**Ans:** A **constraint** is a rule applied to table columns to ensure **data integrity, accuracy, and reliability** in a database.

**Types of Constraints in SQL**

**NOT NULL**

Ensures a column **cannot store NULL values**.

CREATE TABLE Students (

StudentID INT NOT NULL,

Name VARCHAR(50) NOT NULL

);

**UNIQUE**

Ensures **all values in a column are different** (no duplicates allowed).

CREATE TABLE Employees (

Email VARCHAR(100) UNIQUE

);

**PRIMARY KEY**

**Combination of NOT NULL + UNIQUE**.

Identifies each row in a table **uniquely**.

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50)

);

**FOREIGN KEY**

Links two tables together.

Ensures the value in one table matches the **primary key** in another.

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

StudentID INT,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);

**CHECK**

Ensures that values meet a **specific condition**.

CREATE TABLE Students (

Age INT CHECK (Age >= 18)

);

**DEFAULT**

Assigns a **default value** if no value is provided.

CREATE TABLE Students (

City VARCHAR(50) DEFAULT 'Delhi'

);

1. **Difference between RDBMS vs DBMS**

**Ans:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **DBMS (Database Management System)** | **RDBMS (Relational Database Management System)** |
| **Data Storage** | Stores data as **files** | Stores data in **tables (rows & columns)** |
| **Relationship** | No relation between data | Data is related using **keys (PK, FK)** |
| **Normalization** | Not supported | Supports **normalization** to remove redundancy |
| **Data Integrity** | Less focus on constraints | Maintains integrity using **constraints** |
| **Examples** | File System, XML Database | MySQL, Oracle, SQL Server, PostgreSQL |
| **Security** | Provides basic security | Provides **advanced security & access control** |
| **Data Handling** | Suitable for **small data** | Suitable for **large, complex data** |
| **Redundancy** | High (same data may repeat) | Low (redundancy controlled by relations) |

1. **What is an SQL alias?**

## Ans: ****SQL Alias****

An **alias** in SQL is a **temporary name** given to a **table or a column** to make queries easier to read and write.

* It is **not permanent** (only exists while the query runs).
* Created using the keyword **AS** (though AS is optional).

**Alias for a Column**

SELECT Name AS StudentName, Age AS StudentAge

FROM Students;

Here, Name will be shown as **StudentName** in the output.

**Alias for a Table**

SELECT s.Name, c.CourseName

FROM Students AS s

JOIN Courses AS c

ON s.CourseID = c.CourseID;

1. **Write a query to create the table in Structured Query Language.**

**Ans:** CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Age INT CHECK (Age >= 18),

Email VARCHAR(100) UNIQUE,

City VARCHAR(50) DEFAULT 'Delhi'

);

1. **Write a query to insert data into table.**

### **Ans:** Example: Insert Data into Students Table

#### 1. **Insert a single row**

INSERT INTO Students (StudentID, Name, Age, Email, City)

VALUES (1, 'Nisha', 21, 'nisha@mail.com', 'Mumbai');

#### 2. **Insert multiple rows at once**

INSERT INTO Students (StudentID, Name, Age, Email, City)

VALUES

(2, 'Amit', 22, 'amit@mail.com', 'Delhi'),

(3, 'Priya', 20, 'priya@mail.com', 'Bangalore'),

(4, 'Rahul', 23, 'rahul@mail.com', 'Chennai');

1. **Write a query to update data into table with validations.**

**Ans:** UPDATE Students

SET

Age = 22,

City = 'Pune'

WHERE

StudentID = 2

AND Age >= 18;

1. **Write a query to delete data from table with validations.**

**Ans:**

DELETE FROM Students

WHERE StudentID = 3 -- Delete specific student

AND Age < 18; -- Validation: only delete if age is less than 18

1. **Write a query to insert new column in existing table.**

**Ans:**

ALTER TABLE Students

ADD PhoneNumber VARCHAR(15);

1. **Write a query to drop table and database.**

### **Ans:**

### 1. **Drop a Table**

DROP TABLE Students;

### 2. **Drop a Database**

DROP DATABASE SchoolDB;

1. **Write a query to find max and min value from table.**

#### **Ans:**

#### 1. **Find Maximum Age**

SELECT MAX(Age) AS MaxAge

FROM Students;

#### 2. **Find Minimum Age**

SELECT MIN(Age) AS MinAge

FROM Students;

1. **Create two tables named Seller and Product apply foreign key in product table**

### **Ans:** Step 1: Create Seller Table

CREATE TABLE Seller (

SellerID INT PRIMARY KEY, -- Unique ID for each seller

SellerName VARCHAR(50) NOT NULL,

Contact VARCHAR(15)

);

Step 2: Create Product Table with Foreign Key

CREATE TABLE Product (

ProductID INT PRIMARY KEY, -- Unique ID for each product

ProductName VARCHAR(50) NOT NULL,

Price DECIMAL(10,2),

SellerID INT, -- Foreign key column

FOREIGN KEY (SellerID) REFERENCES Seller(SellerID)

);

1. **Fetch data from both table using different joins.**

**Ans:**

### **INNER JOIN**

* Returns only the rows **that have matching values** in both tables.

SELECT s.SellerName, p.ProductName, p.Price

FROM Seller s

INNER JOIN Product p

ON s.SellerID = p.SellerID;

**Result:** Only products that have a valid seller.

### 2️ **LEFT JOIN (LEFT OUTER JOIN)**

* Returns **all rows from the left table (Seller)** and matching rows from Product.
* If no match → Product columns show NULL.

SELECT s.SellerName, p.ProductName, p.Price

FROM Seller s

LEFT JOIN Product p

ON s.SellerID = p.SellerID;

**Result:** All sellers will be listed, even if they have no products.

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

* Returns **all rows from the right table (Product)** and matching rows from Seller.
* If no match → Seller columns show NULL.

SELECT s.SellerName, p.ProductName, p.Price

FROM Seller s

RIGHT JOIN Product p

ON s.SellerID = p.SellerID;

**Result:** All products will be listed, even if they are not linked to a seller.

### 4️ **FULL JOIN (FULL OUTER JOIN)**

* Returns **all rows from both tables**, whether matched or not.

SELECT s.SellerName, p.ProductName, p.Price

FROM Seller s

FULL OUTER JOIN Product p

ON s.SellerID = p.SellerID;

**Result:** Shows all sellers and all products; unmatched rows have NULL values.

### 5️ **CROSS JOIN**

* Returns **Cartesian product** (every seller combined with every product).

SELECT s.SellerName, p.ProductName, p.Price

FROM Seller s

CROSS JOIN Product p;

**Result:** All possible combinations of sellers and products.

1. **What is API Testing**

## Ans:

**API Testing** is a type of **software testing** that focuses on **testing Application Programming Interfaces (APIs)** directly.

* The goal is to verify that the API **works as expected, meets functionality requirements, performs well, and is secure**.
* Unlike UI testing, API testing **does not involve the user interface**; it directly tests the business logic, data, and responses.

1. **Types of API Testing**

## Ans: ****Types of API Testing****

1. **Functional Testing**
   * Checks whether the API **performs its intended function**.
   * Example: For a GET /users API, it should return a list of users correctly.
2. **Load/Performance Testing**
   * Checks the **performance of the API under heavy load**.
   * Example: Testing how POST /orders API behaves when 1000 requests are sent simultaneously.
3. **Security Testing**
   * Ensures that the API is **secure from unauthorized access or attacks**.
   * Example: Checking authentication, authorization, data encryption, and SQL injection.
4. **Reliability/Regression Testing**
   * Ensures the API **continues to work correctly after updates or changes**.
   * Example: Testing old endpoints after adding new features.
5. **Validation Testing**
   * Ensures that the API **returns correct data** according to business rules and requirements.
   * Example: Checking if GET /user/1 returns age, name, and email in the correct format.
6. **Error Handling Testing**
   * Verifies that the API **handles invalid requests gracefully** and returns appropriate error codes.
   * Example: Sending a wrong user ID should return 404 Not Found.
7. **Integration Testing**
   * Tests how the API **interacts with other APIs, databases, or services**.
   * Example: Checking if placing an order through API updates both the orders database and inventory system correctly.
8. **What is Responsive Testing?**

## Ans:

**Responsive Testing** is a type of **software testing** that checks whether a **website or web application** is displayed correctly and functions properly across **different devices, screen sizes, and resolutions**.

* The goal is to ensure that the **UI/UX adapts smoothly** whether the user is on a **desktop, tablet, or mobile device**.
* It is an important part of **cross-browser and cross-device testing**.

1. **Which types of tools are available for Responsive Testing**

**Ans:**

## ****Types of Tools for Responsive Testing****

Responsive testing tools are mainly used to **check websites across different devices, screen sizes, and browsers**. They can be categorized as:

### 1️ **Browser Developer Tools (Built-in Tools)**

* Most modern browsers have built-in **responsive testing features**.
* Example: **Google Chrome DevTools**, **Firefox Developer Tools**.
* **Features:**
  + Simulate different screen sizes and resolutions.
  + Test orientation (portrait/landscape).
  + Inspect and debug HTML/CSS for responsiveness.

### 2 **Online Responsive Testing Tools**

* These are **web-based platforms** where you can test your site on multiple devices and resolutions.
* **Examples:**
  + **Responsinator** – shows website on popular device frames.
  + **Screenfly** – test website on desktops, tablets, smartphones, and TVs.
  + **Am I Responsive?** – quick visual simulation for multiple devices.

### 3️ **Cross-Browser Testing Platforms**

* Allow testing **both responsiveness and browser compatibility**.
* **Examples:**
  + **BrowserStack** – real devices, real browsers.
  + **LambdaTest** – cloud platform for automated and manual testing.
  + **Sauce Labs** – test on multiple OS, browsers, and devices.

### 4️ **Automated Testing Tools (with Responsive Capabilities)**

* Allow **automated testing of responsiveness** in addition to functional tests.
* **Examples:**
  + **Selenium WebDriver** – can resize browser windows and test layouts programmatically.
  + **Cypress** – can test UI responsiveness in automated scripts.

1. **What is the full form of .ipa, .apk**

### **Ans: .apk**

**Full Form:** **Android Package**

**.ipa Full Form:** **iOS App Store Package** (or sometimes called **iOS App Archive**)

1. **How to create step for to open the developer option mode ON?**

**Ans:**

## ****A. For Android Devices****

1. **Open Settings** → Go to your device **Settings** app.
2. **About Phone** → Scroll down and tap **About Phone** (or About Device).
3. **Find Build Number** → Look for **“Build Number”** or **“Software Version”**.
4. **Tap Multiple Times** → Tap **Build Number 7 times** consecutively.
   * You may need to enter your **PIN or password**.
5. **Confirmation Message** → You will see “**You are now a developer!**”
6. **Open Developer Options** → Go back to **Settings → System → Developer Options** (location may vary by device).
7. **Enable Features** → Turn ON features like **USB Debugging, OEM Unlocking, or Stay Awake**.

## ****B. For iOS Devices****

* **iOS does not have traditional Developer Options like Android**, but you can enable **Developer Mode** for app testing:

1. **Install Xcode** → Connect your iPhone/iPad to a Mac with **Xcode installed**.
2. **Trust Device** → On iPhone: **Settings → General → Device Management / Profiles** and trust your developer account.
3. **Enable Developer Mode** → Go to **Settings → Privacy & Security → Developer Mode → Turn ON**.
4. **Restart Device** → Your iOS device may ask to restart to apply Developer Mode.