* ***Assignment***

**\*1.** **What is RDBMS. ?**

**RDBMS** stands for **Relational Database Management System**. It is a type of database management system that stores data in a **structured format using tables** (also called relations). Each table contains **rows** and **columns**.

**Common RDBMS Software:**

* Mysql
* Oracle Database
* Microsoft SQL Server
* PostgreSQL
* SQLite

**\*2. What is SQL. ?**

**SQL (Structured Query Language)** is a standard programming language specifically designed for managing and manipulating relational databases. It allows users to perform tasks such as:

### Common SQL Clauses:

* WHERE: Filters records.
* ORDER BY: Sorts the result.
* GROUP BY: Groups data by column(s).
* JOIN: Combines rows from two or more tables.

**\*3. Write SQL Commands. ?**

**SQL Command**

* Select – select\* from student
* Delete- DELETE FROM Students

WHERE StudentID = 1;

* Update- SET Age = 16

WHERE StudentID = 1;

* Insert - INSERT INTO Students (StudentID, Name, Age, Grade)

VALUES (1, 'John Doe', 15, '10th');

* Create table – create table schooldb;
* Drop table - DROP DATABASE SchoolDB;

**\*4.**  **What is join. ?**

In **SQL**, a **JOIN** is used to **combine rows from two or more tables** based on a related column between them (usually a **foreign key** and **a primary key** relationship).

**Why use JOIN ?**

To retrieve data that is spread across multiple tables. For example, you might have.

To get the name of employees along with their department names, you would use a **JOIN**.

**\*5. Write type of joins.?**

### 1. INNER JOIN

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

SELECT \* FROM table1

INNER JOIN table2

ON table1.id = table2.id;

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

* Returns all rows from the left table and matched rows from the right table.
* If no match is found, NULLs are returned for columns from the right table.

SELECT \*FROM table1

LEFT JOIN table2

ON table1.id = table2.id;

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

* Returns all rows from the right table and matched rows from the left table.
* If no match is found, NULLs are returned for columns from the left table.

SELECT \*FROM table1

RIGHT JOIN table2

ON table1.id = table2.id;

**4. FULL JOIN (or FULL OUTER JOIN)**

* Returns all rows from both tables.
* If there is no match, NULLs are returned for missing matches from either side.

**\*6. How Many constraint and describes it self. ?**

**1. NOT NULL**

* **Description**: Ensures that a column cannot have a NULL value.
* **Example**:

CREATE TABLE Employee (

ID INT NOT NULL,

Name VARCHAR(100) NOT NULL

);

**2. UNIQUE**

* **Description**: Ensures all values in a column are different.
* **Example**:

CREATE TABLE Employee (

Email VARCHAR(100) UNIQUE

);

**3. PRIMARY KEY**

* **Description**: A combination of NOTNULL and UNIQUE. Uniquely identifies each row.
* **Example**:

CREATE TABLE Employee (

ID INT PRIMARY KEY

);

**4. FOREIGN KEY**

* **Description**: Ensures the value in one table matches a value in another table (used for referential integrity).
* **Example**:

CREATE TABLE Orders (

OrderID INT,

EmployeeID INT,

FOREIGN KEY (EmployeeID) REFERENCES Employee(ID)

);

**\*7. Difference between RDBMS vs DBMS. ?**

|  |  |
| --- | --- |
| **Dbms** | **Rdbms** |
| |  | | --- | | Stores data as files or in a hierarchical or navigational form |  |  | | --- | |  | | |  | | --- | | Stores data in tabular (rows and columns) format |  |  | | --- | |  | |
| |  | | --- | | No relation between tables or data |  |  | | --- | |  | | |  | | --- | | Tables are related to each other via foreign keys |  |  | | --- | |  | |
| |  | | --- | |  |  |  | | --- | | Not necessary | | |  | | --- | | Supports normalization to reduce redundancy |  |  | | --- | |  | |
| |  | | --- | | Limited support |  |  | | --- | |  | | |  | | --- | | Enforces data integrity using constraints (PK, FK, etc.) |  |  | | --- | |  | |
| |  | | --- | | May not fully support |  |  | | --- | |  | | |  | | --- | | Fully supports ACID properties for transactions |  |  | | --- | |  | |
| |  | | --- | | XML, File System, Microsoft Access (non-relational) |  |  | | --- | |  | | |  | | --- | | Mysql, Oracle, PostgreSQL, SQL Server |  |  | | --- | |  | |
| Limited | |  | | --- | | Supports concurrency and multiple users |  |  | | --- | |  | |

* **DBMS** is a more general system for managing databases, suitable for small-scale apps.
* **RDBMS** is a specific type of DBMS that organizes data in **relations (tables)** and is ideal for large-scale, structured data.

**\*8. What is an SQL alias.?**

An **SQL alias** is a temporary name given to a **table** or **column** in a SQL query, typically used to make the output more readable or to simplify complex expressions.

### Benefits of Using Aliases:

* Makes queries shorter and cleaner
* Improves readability
* Helps in self-joins or joins between multiple tables
* Useful for complex calculations or sub queries.

**\*9. Write a query to create the table in Structured Query Language. ?**

SQL query to **create a table** using the CREATE TABLE statement in Structured Query Language (SQL):

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

First Name VARCHAR(50),

Last Name VARCHAR(50),

Age INT,

Email VARCHAR(100),

Enrollment Date Date

);

**\*10. Write a query to insert data into table. ?**

INSERT INTO table\_ name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

CREATE TABLE Employees (

EmployeeID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Department VARCHAR(50)

);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Department)

VALUES (1, 'John', 'Doe', 'HR');

INSERT INTO Employees (EmployeeID, FirstName, LastName, Department)

VALUES

(2, 'Jane', 'Smith', 'Finance'),

(3, 'Alice', 'Johnson', 'IT');

**\*11. Write a query to update data into table with validations. ?**

employee\_id (INT)

name (VARCHAR)

salary (DECIMAL)

department (VARCHAR)

**Requirement:**

* Increase salary by 10% for employees in the Sales department.
* But only if their current salary is less than 50000.

**SQL Query:**

UPDATE employees

SET salary = salary \* 1.10

WHERE department = 'Sales'

AND salary < 50000;

**Validations Used:**

* Department = 'Sales' – ensures only Sales department employees are affected.
* Salary < 50000 – ensures the rule applies only to lower-paid employees.

**\*12.** **Write a query to delete data from table with validations.?**

DELETE FROM Customers

WHERE Status = 'Inactive'

AND LastOrderDate < CURRENT\_DATE - INTERVAL 2 YEAR;

**Validate Before Deleting**

SELECT \* FROM Customers

WHERE Status = 'Inactive'

AND LastOrderDate < CURRENT\_DATE - INTERVAL 2 YEAR;

**\*13. Write a query to insert new column in existing table. ?**

ALTER TABLE table\_name

ADD column\_name data\_type;

ALTER TABLE employees

ADD email VARCHAR(100);

ALTER TABLE employees

ADD (

department VARCHAR(50),

hire\_date DATE

);

**\*14. Write a query to drop table and database. ?**

* **Drop table:**

DROP TABLE table\_name;

* **Drop database:**

DROP DATABASE database\_name;

**\*15. Write a query to find max and min value from table. ?**

**Table: employees**

**Column: salary**

SELECT

MAX(salary) AS max\_salary,

MIN(salary) AS min\_salary

FROM employees;

**Explanation:**

* MAX(salary): Finds the highest value in the salary column.
* MIN(salary): Finds the lowest value in the salary column.
* AS max\_salary and AS min\_salary: Give readable names to the output columns.

**\*16. Create two tables named Seller and Product apply foreign key in product table Fetch data from both table using different joins.?**

**CREATE TABLE Seller** (

Seller\_id INT PRIMARY KEY,

Seller\_name VARCHAR (100),

Location VARCHAR (100)

);

**Product Table with Foreign Key to Seller**

**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)

);

**Sample Data**

Insert sample sellers

INSERT INTO Seller (seller\_id, seller\_name, location) VALUES

(1, 'Alice', 'Delhi'),

(2, 'Bob', 'Mumbai'),

(3, 'Charlie', 'Bangalore');

**Insert sample products**

INSERT INTO Product (product\_id, product\_name, price, seller\_id) VALUES

(101, 'Laptop', 55000.00, 1),

(102, 'Phone', 25000.00, 1),

(103, 'Tablet', 15000.00, 2),

(104, 'Monitor', 10000.00, NULL); -- No seller assigned

Step 3: JOIN Queries

**INNER JOIN**

Fetch sellers who have products.

SELECT Seller.seller\_name, Product.product\_name, Product.price

FROM Seller

INNER JOIN Product ON Seller.seller\_id = Product.seller\_id;

**LEFT JOIN**

Fetch all sellers and their products (if any).

SELECT Seller.seller\_name, Product.product\_name, Product.price

FROM Seller

LEFT JOIN Product ON Seller.seller\_id = Product.seller\_id;

**\*17.**  **What is API Testing. ?**

**API Testing** is a type of software testing that focuses on verifying that **Application Programming Interfaces (APIs)** work as expected. It involves testing APIs directly and as part of integration testing to determine if they return correct responses, handles errors properly, and maintain performance, security, and reliability.

 **Functionality Testing** – Verifies that the API performs the intended business logic correctly.

 **Validation Testing** – Ensures the API returns the correct data in the correct format.

 **Load Testing** – Tests how the API performs under stress or heavy traffic.

 **Security Testing** – Checks for vulnerabilities like data leaks, unauthorized access, etc.

 **Error Handling** – Confirms that proper error codes and messages are returned for invalid inputs or system failures.

 **Automation** – API testing is usually automated using tools (Postman, SoapUI, REST Assured, etc.).

**\*18. Types of API Testing. ?**

* **REST API (Representational State Transfer)**

REST is an architectural style that uses **HTTP** for communication. It's lightweight, stateless, and widely used in web services.

|  |
| --- |
| Architectural style |
|  |
|  |

|  |
| --- |
|  |
| JSON, XML, others | |

|  |
| --- |
|  |
| Faster (lightweight) | |

|  |
| --- |
|  |
| More flexible | |

|  |
| --- |
|  |
| Basic HTTPS, | |

|  |
| --- |
|  |

* **SOAP API (Simple Object Access Protocol)**

SOAP is a **protocol** for exchanging structured information using **XML**. It's more rigid but very secure and reliable.

|  |
| --- |
| Protocol |

|  |
| --- |
|  |
| XML only | |

|  |
| --- |
|  |
|  |

|  |  |
| --- | --- |
| Slower (heavyweight) | |
| More strict |

|  |
| --- |
|  |
| Built-in (WS-Security) | |

|  |
| --- |
|  |

**\*19. What is Responsive Testing. ?**

* **Responsive Testing** is a type of software testing used to verify that a website or web application behaves and displays correctly across different:
* Devices (desktop, tablet, mobile)
* Screen **sizes and resolutions**
* Orientations (portrait and landscape)
* Browsers **and operating systems**
* Text/content not adjusting or wrapping correctly
* Buttons or menus being misaligned
* Overlapping or hidden elements
* Poor touch response on mobile devices

**\*20. Which types of tools are available for Responsive Testing. ?**

There are several tools available for **Responsive Testing**-they help ensure your web application looks and functions correctly across different devices, screen sizes, and orientations. These tools fall into various categories:

* **Types of Tools**
* **Browser Developer Tool**
* Online Responsive Testing Tools
* Cross-Browser Testing Platforms
* Automation Testing Tools
* Visual Regression Testing Tools

**\*21.** **What is the full form of .ipa, .apk. ?**

**Ipa**  = **iOS App Store Package**

**Apk = Android Package Kit**

**\*22. How to create step for to open the developer option mode on. ?**

* **Open "Settings" app on** your Android device.
* **Scroll down** andtapon **"About phone"** or **"About device"** (varies by brand).
* Find **"Build number"** option.
* On some devices, you may need to go into **"Software information"** first.
* **Tap "Build number" 7 times quickly.**
* You might be asked to enter your **device PIN or pattern.**
* A message will appear:  
  **"You are now a developer!"**

To Access Developer Options:

1. Go back to Settings.
2. Scroll to the bottom and you’ll now see "Developer options".
3. Tap it to open.
4. Toggle "ON" at the top to activate Developer Mode.