Module 5 :- Database

Q1. What do you understand By Database ?

* A database is a structured collection of data stored electronically, managed by a Database Management System (DBMS). It organizes data into tables, ensures data integrity, provides secure access, and supports efficient data retrieval and manipulation through query languages like SQL. Databases are essential for handling large volumes of information in various applications.

Q2. What is Normalization?

* Normalization in DBMS is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into tables and defining relationships between them according to specific rules (normal forms). The main goal is to ensure that each piece of data is stored only once, thus minimizing duplication and ensuring consistency.

Q3. What is Difference between DBMS and RDBMS?

* DBMS : Manages data as files without a structured relationship.

Lacks strong integrity constraints.

May not support normalization.

Uses various query methods.

May not fully support ACID properties.

* RDBMS : Stores data in tables with relationships using keys.

Enforces integrity constraints like primary and foreign keys.

Supports normalization to reduce redundancy.

Primarily uses SQL for querying.

Strictly adheres to ACID properties for transaction.

Q4. What is MF Cod Rule of RDBMS Systems?

* Information Rule: All data is stored in tables.
* Guaranteed Access Rule: Each data item can be accessed by table name, primary key, and column name.
* Systematic Treatment of Null Values: Nulls are uniformly treated as missing information.
* View Updating Rule: Views should be updatable.
* High-Level Insert, Update, and Delete: Must support set-based operations for data manipulation.
* Physical Data Independence: Changes in physical storage do not affect data access.
* Logical Data Independence: Changes in logical structure do not affect applications.
* Integrity Independence: Integrity constraints are separate from application programs.
* Distribution Independence: The database should work independently of its distribution across a network.

Q5. What do you understand By Data Redundancy?

* Data redundancy refers to the unnecessary duplication of data within a database or storage system. It means the same piece of data is stored in multiple places, which can lead to increased storage costs, data inconsistencies, and maintenance challenges. Reducing data redundancy is a key goal of database normalization.

Q6. What is DDL Interpreter?

* A DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) that processes and executes DDL commands. These commands define and manage the structure of the database, such as creating, altering, and deleting tables, indexes, and schemas. The DDL interpreter translates these commands into actions that modify the database schema.

Q7. What is DML Compiler in SQL?

* A DML (Data Manipulation Language) compiler in SQL processes and executes DML commands, which are used to query and manipulate data in a database. These commands include operations such as SELECT, INSERT, UPDATE, and DELETE. The DML compiler translates these commands into a form that can be executed by the database engine, ensuring that data is retrieved or modified as requested.

Q8. What is SQL Key Constraints? writing an Example of SQL Key Constraints.

* SQL key constraints are rules that enforce data integrity by defining how data is uniquely identified and related in a table. The main types are:
* Primary Key: Uniquely identifies each record in a table.
* Foreign Key: Ensures referential integrity between tables.
* Unique Key: Ensures all values in a column are unique.
* Not Null: Ensures a column cannot have NULL values.

Example :- CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY , -- Primary Key

FirstName VARCHAR(50) NOT NULL, -- Not Null

Email VARCHAR(100) UNIQUE, -- Unique Key

DepartmentID INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID) -- Foreign Key

);

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY, -- Primary Key

DepartmentName VARCHAR(100) NOT NULL -- Not Null

);

Q9. What is save Point? How to create a save Point write a Query?

* A savepoint in SQL is a marker within a transaction that allows you to roll back part of the transaction to that specific point without affecting the entire transaction. This is useful for managing complex transactions and error recovery.

Create a Savepoint: Use the SAVEPOINT command to define a savepoint in the transaction.

Rollback to Savepoint: Use the ROLLBACK TO SAVEPOINT command to undo changes made after the savepoint.

Release Savepoint: Use the RELEASE SAVEPOINT command to remove the savepoint.

Q10. What is trigger and how to create a Trigger in SQL?

* A trigger in SQL is a special type of stored procedure that automatically executes or fires in response to certain events on a table, such as INSERT, UPDATE, or DELETE operations. Triggers are used to enforce business rules, maintain data integrity, or automatically perform certain actions when specific changes occur in the database.
* Creating a Trigger
* To create a trigger, you use the CREATE TRIGGER statement. You specify the trigger”s name, the event that activates it (INSERT, UPDATE, DELETE), and the action to be performed.

TASK

Task 1 :- CREATE DATABASE myassignment;

CREATE TABLE student (rollno int PRIMARY KEY AUTO\_INCREMENT NOT null, name varchar(20), branch varchar(30));

INSERT INTO student (name, branch) VALUES

(“Jay” , “Computer science”),

(“Suhani” , “electronic and com”),

(“Kirit” , “electronic and com”);

CREATE TABLE Exam ( Rollno INT, S\_code VARCHAR(10), Marks INT, P\_code VARCHAR(10), FOREIGN KEY (Rollno) REFERENCES Student(Rollno) );

INSERT INTO Exam (Rollno, S\_code, Marks, P\_code) VALUES (1, “CS111”, 50, “CS”), (1, “CS112”, 50, “CS”), (2, “EC101”, 66, “EC”), (2, “EC102”, 70, “EC”), (3, “EC101”, 45, “EC”), (3, “EC102”, 50, “EC”);

Task 2:-

CREATE TABLE basicinformation(Firstname varchar(20), Lastname varchar(20), Address varchar(30), City varchar(10), Age int);

INSERT INTO basicinformation(Firstname, Lastname, Address, City, Age) VALUES (“Mickey”, “Mouse”, “123 Fantasy Way”, “Anaheim”, 73), (“Bat”, “Man”, “321 Cavern Ave”, “Gotham”, 54), (“Wonder”, “Women”, “987 Truth Way”,”Paradise”, 39), (“Donald”, “duck”, “555 Quack street”, “Mallard”, 65), (“Bugs”, “Bunny”, “567 Carrot Street”, “Rascal”,58), (“Wiley”, “Coyote”, “999 Acme Way”, “Canyon”, 61), (“Cat”, “Women”, “234 Purrfect street”, “hairball”, 32), (“Tweety”, “bird”, “543”, “Itotitaw”, 28);

Task 3 :-

CREATE TABLE Employee(Employee\_id int PRIMARY KEY AUTO\_INCREMENT not null, First\_name varchar(10), Last\_name varchar(10), Salary int, Joining\_date DATETIME, Department varchar(10));

INSERT INTO Employee (Employee\_id, First\_name, Last\_name, Salary, Joining\_date, Department) VALUES (1, 'John', 'Abraham', 1000000, '2013-01-01 12:00:00', 'Banking'), (2, 'Michael', 'Clarke', 800000, '2013-01-01 12:00:00', 'Insurance'), (3, 'Roy', 'Thomas', 700000, '2013-02-01 12:00:00', 'Banking'), (4, 'Tom', 'Jose', 600000, '2013-02-01 12:00:00', 'Insurance'), (5, 'Jerry', 'Pinto', 650000, '2013-01-15 12:00:00', 'Insurance'), (6, 'Philip', 'Mathew', 750000, '2013-01-15 12:00:00', 'Services'), (7, 'TestName1', '123', 650000, '2013-01-01 12:00:00', 'Services'), (8, 'TestName2', 'Lname%', 600000, '2013-02-01 12:00:00', 'Insurance');

CREATE TABLE Incentive ( Employee\_ref\_id INT, Incentive\_date DATE, Incentive\_amount DECIMAL(15, 2), FOREIGN KEY (Employee\_ref\_id) REFERENCES Employee(Employee\_id) );

INSERT INTO Incentive (Employee\_ref\_id, Incentive\_date, Incentive\_amount)

VALUES

(1, '2013-02-01', 5000),

(2, '2013-02-01', 3000),

(3, '2013-02-01', 4000),

(1, '2013-01-01', 4500),

(2, '2013-01-01', 3500);

B) SELECT First\_name, Joining\_date, Salary FROM Employee;

C) SELECT \* FROM Employee ORDER BY First\_name ASC, Salary DESC;

D) SELECT \* FROM Employee WHERE First\_name LIKE '%J%';

E) SELECT Department, MAX(Salary) AS Max\_Salary FROM Employee GROUP BY Department ORDER BY Max\_Salary ASC;

F) SELECT e.First\_name, i.Incentive\_amount

FROM Employee e

JOIN Incentive i ON e.Employee\_id = i.Employee\_ref\_id

WHERE i.Incentive\_amount > 3000;

G) CREATE TABLE ViewTable ( Employee\_id INT, First\_name VARCHAR(50), Last\_name VARCHAR(50), Salary DECIMAL(15, 2), Joining\_date TIMESTAMP, Department VARCHAR(50) );

CREATE TRIGGER after\_employee\_insert

AFTER INSERT ON Employee

FOR EACH ROW

INSERT INTO ViewTable (Employee\_id, First\_name, Last\_name, Salary, Joining\_date, Department)

VALUES (NEW.Employee\_id, NEW.First\_name, NEW.Last\_name, NEW.Salary, NEW.Joining\_date, NEW.Department);

TASK 4:-

CREATE TABLE Salesperson ( SNo INT PRIMARY KEY, SName VARCHAR(50), City VARCHAR(50), Comm DECIMAL(5, 2) );

INSERT INTO Salesperson (SNo, SName, City, Comm) VALUES (1001, 'Peel', 'London', 0.12), (1002, 'Serres', 'San Jose', 0.13), (1004, 'Motika', 'London', 0.11), (1007, 'Rafkin', 'Barcelona', 0.15), (1003, 'Axelrod', 'New York', 0.10);

CREATE TABLE Customer ( CNM INT PRIMARY KEY, CName VARCHAR(50), City VARCHAR(50), Rating INT, SNo INT, FOREIGN KEY (SNo) REFERENCES Salesperson(SNo) );

INSERT INTO Customer (CNM, CName, City, Rating, SNo) VALUES (201, 'Hoffman', 'London', 100, 1001), (202, 'Giovanne', 'Roe', 200, 1003), (203, 'Liu', 'San Jose', 300, 1002), (204, 'Grass', 'Barcelona', 100, 1007), (206, 'Clemens', 'London', 300, 1007), (207, 'Pereira', 'Roe', 100, 1004);

1. SELECT \* FROM Customer WHERE Rating > 1000;
2. SELECT SName, City FROM Salesperson WHERE City = 'London' AND Comm > 0.12;
3. SELECT \* FROM Salesperson WHERE City IN ('Barcelona', 'London');
4. SELECT \* FROM Salesperson WHERE Comm BETWEEN 0.10 AND 0.12;
5. SELECT \* FROM customer WHERE rating >100 or (rating<=100 AND location=’Roe’)

<The END>