|  |  |
| --- | --- |
| 01 | (i)Create tables according to the following definition.  CREATE TABLE DEPOSIT (ACTNO VARCHAR2(5), CNAME VARCHAR2(18), BNAME VARCHAR2(18), AMOUNT NUMBER(8,2) , ADATE DATE);  CREATE TABLE BRANCH(BNAME VARCHAR2(18),CITY VARCHAR2(18));  CREATE TABLE CUSTOMERS(CNAME VARCHAR2(19) ,CITY VARCHAR2(18));  CREATE TABLE BORROW(LOANNO VARCHAR2(5), CNAME VARCHAR2(18), BNAME VARCHAR2(18), AMOUNT NUMBER (8,2));  (ii) Insert the data for all tables.  From the above given tables perform the following queries:   1. Describe deposit, branch. 2. List all data from table DEPOSIT 3. List all data from table BORROW. 4. Give account number and amount of depositors.   **Ans:**  CREATE TABLE DEPOSIT (  ACTNO VARCHAR2(5),  CNAME VARCHAR2(18),  BNAME VARCHAR2(18),  AMOUNT NUMBER(8,2),  ADATE DATE  );  CREATE TABLE BRANCH (  BNAME VARCHAR2(18),  CITY VARCHAR2(18)  );  CREATE TABLE CUSTOMERS (  CNAME VARCHAR2(19),  CITY VARCHAR2(18)  );  CREATE TABLE BORROW (  LOANNO VARCHAR2(5),  CNAME VARCHAR2(18),  BNAME VARCHAR2(18),  AMOUNT NUMBER(8,2)  );  -- Inserting into DEPOSIT table  INSERT INTO DEPOSIT VALUES ('A001', 'Customer1', 'Branch1', 500.00, TO\_DATE('2023-11-23', 'YYYY-MM-DD'));  INSERT INTO DEPOSIT VALUES ('A002', 'Customer2', 'Branch2', 1000.00, TO\_DATE('2023-11-24', 'YYYY-MM-DD'));  -- Inserting into BRANCH table  INSERT INTO BRANCH VALUES ('Branch1', 'City1');  INSERT INTO BRANCH VALUES ('Branch2', 'City2');  -- Inserting into CUSTOMERS table  INSERT INTO CUSTOMERS VALUES ('Customer1', 'City1');  INSERT INTO CUSTOMERS VALUES ('Customer2', 'City2');  -- Inserting into BORROW table  INSERT INTO BORROW VALUES ('L001', 'Customer1', 'Branch1', 1500.00);  INSERT INTO BORROW VALUES ('L002', 'Customer2', 'Branch2', 2000.00);  DESCRIBE DEPOSIT;  DESCRIBE BRANCH;  SELECT \* FROM DEPOSIT;  SELECT \* FROM BORROW;  SELECT ACTNO, AMOUNT FROM DEPOSIT; |
| 02 | Create the below given table and insert the data accordingly.  Job (job\_id, job\_title, min\_sal, max\_sal)  Employee (emp\_no, emp\_name, emp\_sal, emp\_comm, dept\_no) deposit(a\_no,cname,bname,amount,a\_date). borrow(loanno,cname,bname,amount).  Insert the data for all tables.  -> Perform following queries:   1. Retrieve all data from employee, jobs and deposit. 2. Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06 3. Display all jobs with minimum salary is greater than 4000 4. Display name and salary of employee whose department no is 20. Give alias name to name of employee.   **Ans:**  CREATE TABLE Job (  job\_id INT,  job\_title VARCHAR(50),  min\_sal DECIMAL(8,2),  max\_sal DECIMAL(8,2)  );  CREATE TABLE Employee (  emp\_no INT,  emp\_name VARCHAR(50),  emp\_sal DECIMAL(8,2),  emp\_comm DECIMAL(8,2),  dept\_no INT  );  CREATE TABLE Deposit (  a\_no VARCHAR(10),  cname VARCHAR(50),  bname VARCHAR(50),  amount DECIMAL(8,2),  a\_date DATE  );  CREATE TABLE Borrow (  loanno VARCHAR(10),  cname VARCHAR(50),  bname VARCHAR(50),  amount DECIMAL(8,2)  );  -- Inserting sample data into Job table  INSERT INTO Job VALUES (1, 'Manager', 5000.00, 10000.00);  INSERT INTO Job VALUES (2, 'Clerk', 2500.00, 5000.00);  -- Inserting sample data into Employee table  INSERT INTO Employee VALUES (101, 'John Doe', 6000.00, 1000.00, 20);  INSERT INTO Employee VALUES (102, 'Jane Smith', 4000.00, NULL, 30);  -- Inserting sample data into Deposit table  INSERT INTO Deposit VALUES ('A001', 'Customer1', 'Branch1', 5000.00, '2006-01-15');  INSERT INTO Deposit VALUES ('A002', 'Customer2', 'Branch2', 7000.00, '2006-07-20');  -- Inserting sample data into Borrow table  INSERT INTO Borrow VALUES ('L001', 'Customer1', 'Branch1', 3000.00);  INSERT INTO Borrow VALUES ('L002', 'Customer2', 'Branch2', 4000.00);  SELECT \* FROM Employee;  SELECT \* FROM Job;  SELECT \* FROM Deposit;  SELECT a\_no, amount  FROM Deposit  WHERE a\_date BETWEEN '2006-01-01' AND '2006-07-25';  SELECT \* FROM Job WHERE min\_sal > 4000;  SELECT emp\_name AS employee\_name, emp\_sal  FROM Employee  WHERE dept\_no = 20; |
| 03 | Create the below given table and insert the data accordingly.  Job (job\_id, job\_title, min\_sal, max\_sal)  Employee (emp\_no, emp\_name, emp\_sal, emp\_comm, dept\_no) deposit(a\_no,cname,bname,amount,a\_date). borrow(loanno,cname,bname,amount).  Insert the data for all tables.   1. List total deposit from deposit. 2. List total loan from karolbagh branch (3) Give maximum loan from branch vrce. 3. Count total number of customers.   **Ans:**  CREATE TABLE Job (  job\_id INT,  job\_title VARCHAR(50),  min\_sal DECIMAL(8,2),  max\_sal DECIMAL(8,2)  );  CREATE TABLE Employee (  emp\_no INT,  emp\_name VARCHAR(50),  emp\_sal DECIMAL(8,2),  emp\_comm DECIMAL(8,2),  dept\_no INT  );  CREATE TABLE Deposit (  a\_no VARCHAR(10),  cname VARCHAR(50),  bname VARCHAR(50),  amount DECIMAL(8,2),  a\_date DATE  );  CREATE TABLE Borrow (  loanno VARCHAR(10),  cname VARCHAR(50),  bname VARCHAR(50),  amount DECIMAL(8,2)  );  -- Inserting sample data into Job table  INSERT INTO Job VALUES (1, 'Manager', 5000.00, 10000.00);  INSERT INTO Job VALUES (2, 'Clerk', 2500.00, 5000.00);  -- Inserting sample data into Employee table  INSERT INTO Employee VALUES (101, 'John Doe', 6000.00, 1000.00, 20);  INSERT INTO Employee VALUES (102, 'Jane Smith', 4000.00, NULL, 30);  -- Inserting sample data into Deposit table  INSERT INTO Deposit VALUES ('A001', 'Customer1', 'Branch1', 5000.00, '2023-01-15');  INSERT INTO Deposit VALUES ('A002', 'Customer2', 'Karolbagh', 7000.00, '2023-07-20');  -- Inserting sample data into Borrow table  INSERT INTO Borrow VALUES ('L001', 'Customer1', 'Branch1', 3000.00);  INSERT INTO Borrow VALUES ('L002', 'Customer2', 'Karolbagh', 4000.00);  INSERT INTO Borrow VALUES ('L003', 'Customer3', 'VRCE', 6000.00);  SELECT SUM(amount) AS total\_deposit FROM Deposit;  SELECT SUM(amount) AS total\_loan\_karolbagh  FROM Borrow  WHERE bname = 'Karolbagh';  SELECT MAX(amount) AS max\_loan\_from\_vrce  FROM Borrow  WHERE bname = 'VRCE';  SELECT COUNT(DISTINCT cname) AS total\_customers  FROM (SELECT cname FROM Deposit UNION SELECT cname FROM Borrow) AS AllCustomers; |

|  |  |
| --- | --- |
| 04 | To implement Single-row functions.   1. Create tables according to the need. 2. Insert the data for all tables. 3. Write a query to display the current date. Label the column Date. 4. For each employee, display the employee number, job, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary 5. Modify your query to add a column that subtracts the old salary from the new salary. Label the column Increase   **Ans:**  CREATE TABLE Employee (  emp\_no INT,  job VARCHAR(50),  salary DECIMAL(8,2)  );  -- Inserting sample data into Employee table  INSERT INTO Employee VALUES (101, 'Manager', 5000.00);  INSERT INTO Employee VALUES (102, 'Clerk', 2500.00);  -- Insert other sample data as needed  SELECT CURRENT\_DATE() AS Date;  SELECT emp\_no, job, salary, ROUND(salary \* 1.15) AS "New Salary"  FROM Employee;  SELECT emp\_no, job, salary, ROUND(salary \* 1.15) AS "New Salary", ROUND(salary \* 0.15) AS Increase  FROM Employee; |
| 05 | Explain and implement joins and use suitable tables  **Ans:**  -- Create tables  CREATE TABLE Employees (  emp\_id INT PRIMARY KEY,  emp\_name VARCHAR(50),  dept\_id INT  );  CREATE TABLE Departments (  dept\_id INT PRIMARY KEY,  dept\_name VARCHAR(50)  );  -- Insert data into tables  INSERT INTO Employees VALUES (1, 'John Doe', 1);  INSERT INTO Employees VALUES (2, 'Jane Smith', 2);  INSERT INTO Employees VALUES (3, 'Bob Johnson', 1);  INSERT INTO Departments VALUES (1, 'HR');  INSERT INTO Departments VALUES (2, 'IT');   1. INNER JOIN:   SELECT Employees.emp\_id, Employees.emp\_name, Departments.dept\_name  FROM Employees  INNER JOIN Departments ON Employees.dept\_id = Departments.dept\_id;   1. LEFT JOIN:   SELECT Employees.emp\_id, Employees.emp\_name, Departments.dept\_name  FROM Employees  LEFT JOIN Departments ON Employees.dept\_id = Departments.dept\_id;   1. RIGHT JOIN:   SELECT Employees.emp\_id, Employees.emp\_name, Departments.dept\_name  FROM Employees  RIGHT JOIN Departments ON Employees.dept\_id = Departments.dept\_id;   1. FULL JOIN:   SELECT Employees.emp\_id, Employees.emp\_name, Departments.dept\_name  FROM Employees  FULL JOIN Departments ON Employees.dept\_id = Departments.dept\_id; |
| 06 | Apply the concept of Aggregating Data using Group functions use suitable tables.  **Ans:**  -- Create tables  CREATE TABLE Employees (  emp\_id INT PRIMARY KEY,  emp\_name VARCHAR(50),  dept\_id INT,  salary INT  );  CREATE TABLE Departments (  dept\_id INT PRIMARY KEY,  dept\_name VARCHAR(50)  );  -- Insert data into tables  INSERT INTO Employees VALUES (1, 'John Doe', 1, 5000);  INSERT INTO Employees VALUES (2, 'Jane Smith', 2, 6000);  INSERT INTO Employees VALUES (3, 'Bob Johnson', 1, 5500);  INSERT INTO Departments VALUES (1, 'HR');  INSERT INTO Departments VALUES (2, 'IT');   1. COUNT: Count the number of employees in each department.   SELECT dept\_id, COUNT(emp\_id) AS employee\_count  FROM Employees  GROUP BY dept\_id;   1. SUM: Calculate the total salary for each department.   SELECT dept\_id, SUM(salary) AS total\_salary  FROM Employees  GROUP BY dept\_id;   1. AVG: Calculate the average salary for each department.   SELECT dept\_id, AVG(salary) AS avg\_salary  FROM Employees  GROUP BY dept\_id;   1. MIN/MAX: Find the minimum and maximum salary in each department.   SELECT dept\_id, MIN(salary) AS min\_salary, MAX(salary) AS max\_salary  FROM Employees  GROUP BY dept\_id; |
| 07 | solve and implement queries using the concept of sub query.  **Ans:**  -- Create tables  CREATE TABLE Employees (  emp\_id INT PRIMARY KEY,  emp\_name VARCHAR(50),  dept\_id INT,  salary INT  );  CREATE TABLE Departments (  dept\_id INT PRIMARY KEY,  dept\_name VARCHAR(50)  );  -- Insert data into tables  INSERT INTO Employees VALUES (1, 'John Doe', 1, 5000);  INSERT INTO Employees VALUES (2, 'Jane Smith', 2, 6000);  INSERT INTO Employees VALUES (3, 'Bob Johnson', 1, 5500);  INSERT INTO Departments VALUES (1, 'HR');  INSERT INTO Departments VALUES (2, 'IT');   1. Subquery in SELECT Clause: Find the average salary of all employees and display it along with each employee's salary.   SELECT emp\_name, salary, (SELECT AVG(salary) FROM Employees) AS avg\_salary  FROM Employees;   1. Subquery in WHERE Clause: List employees whose salary is above the average salary   SELECT emp\_name, salary  FROM Employees  WHERE salary > (SELECT AVG(salary) FROM Employees);   1. Subquery in FROM Clause: Find the total salary for each department using a subquery in the FROM clause.   SELECT dept\_id, total\_salary  FROM (  SELECT dept\_id, SUM(salary) AS total\_salary  FROM Employees  GROUP BY dept\_id  ) AS DepartmentTotals;   1. Correlated Subquery: List departments along with the employees who have the highest salary in each department.   SELECT dept\_name, emp\_name, salary  FROM Employees e  JOIN Departments d ON e.dept\_id = d.dept\_id  WHERE salary = (  SELECT MAX(salary)  FROM Employees  WHERE dept\_id = e.dept\_id  ); |
| 08 | Use Manipulating Data, consider and create tables as per requirements.   1. Give 10% interest to all depositors. 2. Give 10% interest to all depositors having branch pune. 3. Give 10% interest to all depositors living in nagpur and having branch city as ramtek   **Ans:**  -- Create tables  CREATE TABLE Deposit (  act\_no VARCHAR(5) PRIMARY KEY,  cname VARCHAR(18),  bname VARCHAR(18),  amount NUMBER(8,2),  a\_date DATE  );  CREATE TABLE Branch (  bname VARCHAR(18) PRIMARY KEY,  city VARCHAR(18)  );  CREATE TABLE Customers (  cname VARCHAR(19) PRIMARY KEY,  city VARCHAR(18)  );  -- Insert data into tables  INSERT INTO Deposit VALUES ('A001', 'John Doe', 'Pune', 5000.00, TO\_DATE('2023-11-23', 'YYYY-MM-DD'));  INSERT INTO Deposit VALUES ('A002', 'Jane Smith', 'Nagpur', 7000.50, TO\_DATE('2023-11-24', 'YYYY-MM-DD'));  INSERT INTO Branch VALUES ('Pune', 'Pune');  INSERT INTO Branch VALUES ('Ramtek', 'Nagpur');  INSERT INTO Customers VALUES ('John Doe', 'Pune');  INSERT INTO Customers VALUES ('Jane Smith', 'Nagpur');   1. Give 10% interest to all depositors:   UPDATE Deposit  SET amount = amount \* 1.10;   1. Give 10% interest to all depositors having a branch in Pune:   UPDATE Deposit  SET amount = amount \* 1.10  WHERE bname = 'Pune';   1. Give 10% interest to all depositors living in Nagpur and having a branch city as Ramtek:   UPDATE Deposit  SET amount = amount \* 1.10  WHERE cname IN (  SELECT cname  FROM Customers  WHERE city = 'Nagpur'  ) AND bname IN (  SELECT bname  FROM Branch  WHERE city = 'Ramtek'  ); |
| 09 | To Perform Operations Using PL/SQL.   1. Write a PL/SQL Block that will get the salary of employee with employee number ‘105’ and display it on the Screen   **Ans:**  CREATE PROCEDURE GetEmployeeSalary()  BEGIN  DECLARE v\_salary DECIMAL(10,2);  -- Assuming there is an 'Employees' table with columns 'emp\_id' and 'emp\_salary'  SELECT emp\_salary INTO v\_salary  FROM Employees  WHERE emp\_id = 105;  SELECT CONCAT('Employee 105 Salary: ', v\_salary) AS Result;  END   1. Write a PL/SQL block that prints 1 to 5 numbers Using WHILE Loop Statement   **Ans:**  DELIMITER //  CREATE PROCEDURE print\_numbers()  BEGIN  DECLARE counter INT DEFAULT 1;  WHILE counter <= 5 DO  SELECT counter;  SET counter = counter + 1;  END WHILE;  END //  DELIMITER ;  CALL print\_numbers(); |
| 10 | 1. Write a PL/SQL block that implements Implicit Cursor.   +------------+-------------+------+-----+---------+-------+  | Field | Type | Null | Key | Default | Extra |  +------------+-------------+------+-----+---------+-------+  | s\_id | int | NO | PRI | NULL | |  | s\_name | varchar(20) | NO | | NULL | |  | yearofhire | date | YES | | NULL | |  | comm\_per | float(10,2) | YES | | NULL | |  | city | varchar(20) | YES | | NULL | |  **+------------+-------------+------+-----+---------+-------+**  **Ans:**  use sales;  delimiter //  Declare v\_salesperson\_name VARCHAR2(50);  BEGIN  SELECT s\_name INTO v\_salesperson\_name  FROM salesperson  WHERE s\_id = 1;  DBMS\_OUTPUT.PUT\_LINE('Salesperson Name: ' || v\_salesperson\_name);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('No salesperson found for the specified ID.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);  END;  //   1. Write a PL/SQL block that implements Explicit Cursor.   **Ans:**  use sales;  Delimiter //  DECLARE  CURSOR salesperson\_cursor IS  SELECT s\_id, s\_name FROM salesperson;    v\_salesperson\_id salesperson.s\_id%TYPE;  v\_salesperson\_name salesperson.s\_name%TYPE;  BEGIN  FOR rec IN salesperson\_cursor  LOOP    v\_salesperson\_id := rec.s\_id;  v\_salesperson\_name := rec.s\_name;  DBMS\_OUTPUT.PUT\_LINE('Salesperson ID: ' || v\_salesperson\_id);  DBMS\_OUTPUT.PUT\_LINE('Salesperson Name: ' || v\_salesperson\_name);  END LOOP;  END;  // |
| 11 | Write a PL/SQL block that implements Function.  **Ans:**  CREATE FUNCTION generate\_numbers()  RETURNS INT  BEGIN  DECLARE counter INT DEFAULT 1;  WHILE counter <= 5 DO  SELECT counter;  SET counter = counter + 1;  END WHILE;  RETURN 1;  END  Write a PL/SQL block that implements AFTER UPDATE TRIGGER.  **Ans:**  -- Drop tables if they exist  DROP TABLE IF EXISTS reminders;  DROP TABLE IF EXISTS members;  -- Create the members table  CREATE TABLE members (  id INT AUTO\_INCREMENT,  name VARCHAR(100) NOT NULL,  email VARCHAR(255),  birthDate DATE,  PRIMARY KEY (id)  );  -- Create the reminders table  CREATE TABLE reminders (  id INT AUTO\_INCREMENT,  memberId INT,  E\_NAME VARCHAR(255) NOT NULL,  PRIMARY KEY (id , memberId)  );  -- Set delimiter to handle trigger creation  DELIMITER $$  -- Create the after\_members\_insert trigger  CREATE TRIGGER after\_members\_insert AFTER INSERT ON members FOR EACH ROW  BEGIN  IF NEW.birthDate IS NULL THEN  INSERT INTO reminders(memberId, E\_NAME) VALUES (NEW.id, NEW.name);  END IF;  END $$  -- Create the after\_members\_update trigger  CREATE TRIGGER after\_members\_update AFTER UPDATE ON members FOR EACH ROW  BEGIN  IF NEW.birthDate IS NULL AND OLD.birthDate IS NOT NULL THEN  -- If birthDate was updated to NULL, insert a reminder  INSERT INTO reminders(memberId, E\_NAME) VALUES (NEW.id, NEW.name);  END IF;  END $$  -- Reset the delimiter  DELIMITER ;  -- Insert some data  INSERT INTO members(name, email, birthDate) VALUES  ('John Doe', 'john.doe@example.com', NULL),  ('Jane Doe', 'jane.doe@example.com', '2000-01-01');  -- Update a record to trigger the after\_members\_update trigger  UPDATE members SET birthDate = NULL WHERE id = 2;  -- Query to retrieve reminders before delete  SELECT \* FROM reminders WHERE memberId IN (SELECT id FROM members WHERE birthDate IS NULL); |
| 12 | Write a PL/SQL block that implements BEFORE UPDATE TRIGGER  **Ans:**  use employee;  CREATE TABLE employees\_audit (  id INT AUTO\_INCREMENT PRIMARY KEY,  employeeNumber INT NOT NULL,  lastname VARCHAR(50) NOT NULL,  changedat DATETIME DEFAULT NULL,  action VARCHAR(50) DEFAULT NULL  );  CREATE TRIGGER before\_employee\_update BEFORE UPDATE ON emp FOR EACH ROW    INSERT INTO employees\_audit  SET action = 'update',  employeeNumber = OLD.emp\_id,  lastname = OLD.name,  changedat = NOW();    Show triggers;    UPDATE emp  SET Name = 'amit'  WHERE emp\_id = 1; |
| 13 | Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (<some, >=some, <all etc.) and set cardinality (unique, not unique). Assume suitable tables.  **Ans:**  -- Create tables  CREATE TABLE students (  student\_id INT PRIMARY KEY,  student\_name VARCHAR(100) NOT NULL  );  CREATE TABLE courses (  course\_id INT PRIMARY KEY,  course\_name VARCHAR(100) NOT NULL  );  CREATE TABLE enrollments (  enrollment\_id INT PRIMARY KEY,  student\_id INT,  course\_id INT,  grade INT,  FOREIGN KEY (student\_id) REFERENCES students(student\_id),  FOREIGN KEY (course\_id) REFERENCES courses(course\_id)  );  -- Insert sample data  INSERT INTO students VALUES (1, 'Alice');  INSERT INTO students VALUES (2, 'Bob');  INSERT INTO students VALUES (3, 'Charlie');  INSERT INTO courses VALUES (101, 'Math');  INSERT INTO courses VALUES (102, 'English');  INSERT INTO courses VALUES (103, 'History');  INSERT INTO enrollments VALUES (1, 1, 101, 85);  INSERT INTO enrollments VALUES (2, 1, 102, 90);  INSERT INTO enrollments VALUES (3, 2, 101, 75);  INSERT INTO enrollments VALUES (4, 2, 103, 88);  INSERT INTO enrollments VALUES (5, 3, 102, 92);  INSERT INTO enrollments VALUES (6, 3, 103, 78);  **-- Set Membership (IN, NOT IN)**  SELECT student\_name  FROM students  WHERE student\_id IN (SELECT student\_id FROM enrollments WHERE course\_id = 101);  SELECT student\_name  FROM students  WHERE student\_id NOT IN (SELECT student\_id FROM enrollments WHERE course\_id = 102);  **-- Set Comparison (<SOME, >=SOME, <ALL, etc.)**  SELECT student\_name  FROM students  WHERE grade > SOME (SELECT grade FROM enrollments WHERE course\_id = 101);  SELECT student\_name  FROM students  WHERE grade < ALL (SELECT grade FROM enrollments WHERE course\_id = 102);  **-- Set Cardinality (UNIQUE, NOT UNIQUE)**  SELECT student\_name  FROM students  WHERE grade = ALL (SELECT grade FROM enrollments WHERE course\_id = 103)  AND COUNT(\*) = 1;  SELECT student\_name  FROM students  WHERE grade = ALL (SELECT grade FROM enrollments WHERE course\_id = 101)  AND COUNT(\*) > 1; |
| 14 | Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.  **Ans:**  -- Create tables  CREATE TABLE employees (  employee\_id INT PRIMARY KEY,  employee\_name VARCHAR(100) NOT NULL,  department\_id INT  );  CREATE TABLE departments (  department\_id INT PRIMARY KEY,  department\_name VARCHAR(100) NOT NULL  );  -- Insert sample data  INSERT INTO employees VALUES (1, 'Alice', 101);  INSERT INTO employees VALUES (2, 'Bob', 102);  INSERT INTO employees VALUES (3, 'Charlie', 101);  INSERT INTO departments VALUES (101, 'HR');  INSERT INTO departments VALUES (102, 'IT');  **-- Create a view**  CREATE VIEW employee\_view AS  SELECT e.employee\_id, e.employee\_name, e.department\_id, d.department\_name  FROM employees e  JOIN departments d ON e.department\_id = d.department\_id;  -- View the data in the view  SELECT \* FROM employee\_view;  -- Create a view from multiple tables  CREATE VIEW combined\_view AS  SELECT e.employee\_id, e.employee\_name, e.department\_id, d.department\_name  FROM employees e  JOIN departments d ON e.department\_id = d.department\_id;  -- View the data in the combined view  SELECT \* FROM combined\_view; |
| 15 | 1. Write a PL/SQL block that implements Function. 2. Write a PL/SQL block that implements AFTER INSERT TRIGGER.   **Ans:**  DROP TABLE IF EXISTS members;  CREATE TABLE members (  id INT AUTO\_INCREMENT,  name VARCHAR(100) NOT NULL,  email VARCHAR(255),  birthDate DATE,  PRIMARY KEY (id)  );  DROP TABLE IF EXISTS reminders;  CREATE TABLE reminders (  id INT AUTO\_INCREMENT,  memberId INT,  E\_NAME VARCHAR(255) NOT NULL,  PRIMARY KEY (id , memberId)  );  DELIMITER $$  CREATE TRIGGER after\_members\_insert AFTER INSERT ON members FOR EACH ROW  BEGIN  IF NEW.birthDate IS NULL THEN  INSERT INTO reminders(memberId, E\_NAME) VALUES(new.id,new.name);  END IF;  END$$  DELIMITER ;  INSERT INTO members(name, email, birthDate) VALUES('John Doe', 'john.doe@example.com', NULL),  ('Jane Doe', 'jane.doe@example.com','2000-01-01');  SELECT \* FROM members;  SELECT \* FROM reminders; |
| 16 | Write a PL/SQL block that implements BEFORE DELETE TRIGGER  Write a trigger to check the salary is not Zero or Negative.  **Ans:**  -- Create a trigger to check if the salary is not zero or negative  DELIMITER $$  CREATE TRIGGER before\_insert\_update\_salary  BEFORE INSERT OR UPDATE ON employees  FOR EACH ROW  BEGIN  -- Check if the salary is zero or negative  IF NEW.salary <= 0 THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Salary must be greater than zero';  END IF;  END $$  DELIMITER ; |
|  | Write a Trigger that check the employee resides in a city 'Pune' |
| 17 | Create a database with suitable example using MongoDB and implement    Inserting and saving document (batch insert, insert validation)    Removing document    Updating document (document replacement, using modifiers, upserts, updating    documents, returning updated documents) |
| 18 | Execute at least 10 queries on any suitable MongoDB database  Ans:  db.users.insertOne({ name: "Alice", age: 30, city: "New York" });  db.users.find({ age: { $gte: 25 } });  db.users.updateOne({ name: "Alice" }, { $set: { age: 31 } });  db.users.deleteOne({ name: "Alice" });  db.users.find().sort({ age: 1 }); // Sort by age in ascending order  db.users.count();  db.users.find({}, { name: 1, age: 1 }); // Only retrieve name and age fields  db.users.updateMany({ city: "New York" }, { $set: { city: "San Francisco" } });  db.users.find().limit(5); // Limit to the first 5 results  db.users.aggregate([  { $match: { age: { $gte: 25 } } },  { $group: { \_id: "$city", count: { $sum: 1 } } }  ]); |
| 19 | Execute at least 10 queries on any suitable MongoDB database that demonstrates following:    $ where queries    CRUD Database commands  Ans:  db.users.insertOne({ name: "Alice", age: 30, city: "New York" });  db.users.find({ $where: "this.age > 25" });  db.users.updateOne({ name: "Alice" }, { $set: { age: 31 } });  db.users.deleteOne({ name: "Alice" });  db.users.find({}, { name: 1, age: 1 }); // Only retrieve name and age fields  db.users.count();  db.users.updateMany({ city: "New York" }, { $set: { city: "San Francisco" } });  db.users.find().limit(5); // Limit to the first 5 results  db.users.aggregate([  { $match: { age: { $gte: 25 } } },  { $group: { \_id: "$city", count: { $sum: 1 } } }  ]);  db.users.find({ $where: function() { return this.age > 25 && this.city === "New York"; } }); |
| 20 | Implement Mongodb CRUD operations |
| 21 | Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables. |
| 22 | Create a database with suitable example using MongoDB and implement    Inserting and saving document (batch insert, insert validation)    Removing document    Updating document |

|  |  |
| --- | --- |
| 23 | Consider the table and solve the quries:  Sailors( sid, sname, rating, age)  Boats(bid, bname,color) Reserves(sid,bid,day)   1. Find the names of sailors who have reserved boat number 103   **Ans:**  SELECT s.sname  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  WHERE r.bid = 103;   1. Find the names of sailors who have never reserved boat number 103   **Ans**.  SELECT s.sname  FROM Sailors s  WHERE s.sid NOT IN (  SELECT r.sid  FROM Reserves r  WHERE r.bid = 103  );   1. Find the names of sailors who have reserved a red boat   **Ans:**  SELECT DISTINCT s.sname  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  JOIN Boats b ON r.bid = b.bid  WHERE b.color = 'red';   1. Find the colors of boats reserved by Lubber   **Ans.**  SELECT DISTINCT b.color  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  JOIN Boats b ON r.bid = b.bid  WHERE s.sname = 'Lubber'; |
| 24 | Consider the table and solve the quries:  Sailors( sid, sname, rating, age)  Boats(bid, bname,color) Reserves(sid,bid,day)  CREATE TABLE Sailors (  sid INT PRIMARY KEY,  sname VARCHAR(255),  rating INT,  age INT  );  CREATE TABLE Boats (  bid INT PRIMARY KEY,  bname VARCHAR(255),  color VARCHAR(50)  );  CREATE TABLE Reserves (  sid INT,  bid INT,  day DATE,  PRIMARY KEY (sid, bid, day),  FOREIGN KEY (sid) REFERENCES Sailors(sid),  FOREIGN KEY (bid) REFERENCES Boats(bid)  );   1. Find the names of sailors who have reserved a red or a green boat   **ANS**  SELECT DISTINCT s.sname  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  JOIN Boats b ON r.bid = b.bid  WHERE b.color IN ('red', 'green');   1. Find the names of sailors who have reserved both a red and a green boat   **ANS**  SELECT s.sname  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  JOIN Boats b ON r.bid = b.bid  WHERE b.color = 'red' AND s.sid IN (  SELECT s.sid  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  JOIN Boats b ON r.bid = b.bid  WHERE b.color = 'green'  );     1. Find the names of sailors who have reserved at least two different boats   **Ans**  SELECT s.sname  FROM Sailors s  JOIN Reserves r ON s.sid = r.sid  GROUP BY s.sid, s.sname  HAVING COUNT(DISTINCT r.bid) >= 2;   1. Find the sids of silors with age over 20 who have not reserved a red boat   **Ans**  SELECT s.sid  FROM Sailors s  WHERE s.age > 20 AND s.sid NOT IN (  SELECT r.sid  FROM Reserves r  JOIN Boats b ON r.bid = b.bid  WHERE b.color = 'red'  ); |

|  |  |
| --- | --- |
| 25 | Flights(flno, from, to, distance, departs, arrives, price)  Aircraft(aid, aname, cruisingrange)  Certified(eid, aid)  Employees(eid, ename, salary)  create table Flights(  -> flno int Primary key,  -> departure\_location varchar(50),  -> destination varchar(50),  -> distance int,  -> departs time,  -> arrives time,  -> price int);  Query OK, 0 rows affected (0.07 sec)  mysql> create table aircraft(  -> aid int Primary key,  -> aname varchar(50),  -> max\_dist int);  Query OK, 0 rows affected (0.02 sec)  mysql> create table employees(  -> eid int Primary key,  -> ename varchar(50),  -> salary int);  Query OK, 0 rows affected (0.04 sec)  mysql> create table certified(eid int,  -> aid int,  -> foreign key(eid) references employees(eid),  -> foreign key(aid) references aircraft(aid));  Query OK, 0 rows affected (0.03 sec)     1. Compute the **difference between the average salary of a pilot and the average salary of all employees** (including pilots).   **Ans.** select avg(case when c.aid is not null then salary else 0 end)-avg(salary) as sal\_diff  -> from employees e  -> left join certified c on e.eid = c.eid  -> where c.eid is not null;  +-------------+  | sal\_diff |  +-------------+  | -16666.6667 |  +-------------+   1. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.   **Ans.** select ename, salary  -> from employees  -> left join certified on employees.eid = certified.eid  -> where aid is null and salary > (select avg(salary) from employees lef  t join certified on employees.eid = certified.eid  -> where aid is not null);   1. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.   **Ans.** select ename from employees e inner join certified c on e.id = c.id  -> inner join Aircraft a on c.id = a.id  -> where a.max\_dist>10  -> and e.eid not in (select eid from certified where aid in (select aid  from Aircraft where max\_dist<=10)); |