DBMS LAB-11

106122022

B.Aruteswar Reddy

Q1.

EmployeeDetails.xml

```
<EmployeeDetails>
   <Employee>
      <EmpNo>101</EmpNo>
      <EName>John Smith</EName>
      <Job>Manager</Job>
      <WorkingHours>9</WorkingHours>
      <Dept>Sales</Dept>
      <DeptNo>1</DeptNo>
      <Salary>45000</Salary>
   </Employee>
   <Employee>
      <EmpNo>102</EmpNo>
      <EName>Jane Doe</EName>
      <Job>Researcher</Job>
      <WorkingHours>8</WorkingHours>
      <Dept>Research</Dept>
      <DeptNo>2</DeptNo>
      <Salary>32000</Salary>
   </Employee>
   <Employee>
      <EmpNo>103</EmpNo>
      <EName>Mark Spencer</EName>
```

```
        <Job>Analyst</Job>
        <WorkingHours>10</WorkingHours>
        <Dept>Finance</Dept>
        <DeptNo>3</DeptNo>
        <Salary>50000</Salary>
    </Employee>
    <Employee>
        <EmpNo>104</EmpNo>
        <EName>Sarah Johnson</EName>
        <Job>Developer</Job>
        <WorkingHours>7</WorkingHours>
     <Dept>IT</Dept>
        <DeptNo>4</DeptNo>
        <Salary>40000</Salary>
    </Employee>
    <Employee>
        <EmpNo>105</EmpNo>
        <EName>Linda Smith</EName>
        <Job>Tester</Job>
        <WorkingHours>9</WorkingHours>
        <Dept>Research</Dept>
        <DeptNo>2</DeptNo>
        <Salary>35000</Salary>
    </Employee>
</EmployeeDetails>
```

FlightDetails.xml

```
<FlightDetails>
```

```
<Flight>
    <FlNo>FL101</FlNo>
    <FlName>Flight A</FlName>
    <PilotName>Captain America</PilotName>
    <From>New York</From>
    <To>Los Angeles</To>
    <Date>2024-12-01</Date>
    <DepartsTime>08:00</DepartsTime>
    <ArrivesTime>11:00</ArrivesTime>
    <Price>4500</Price>
</Flight>
<Flight>
    <FlNo>FL102</FlNo>
    <FlName>Flight B</FlName>
    <PilotName>Captain Marvel</PilotName>
    <From>Chicago</From>
    <To>Miami</To>
    <Date>2024-12-01</Date>
    <DepartsTime>09:00</DepartsTime>
    <ArrivesTime>12:00</ArrivesTime>
    <Price>5500</Price>
</Flight>
<Flight>
    <FlNo>FL103</FlNo>
    <FlName>Flight C</FlName>
    <PilotName>Iron Man</PilotName>
    <From>San Francisco</From>
    <To>New York</To>
```

```
        <Date>2024-12-02</Date>
        <DepartsTime>10:00</DepartsTime>
        <ArrivesTime>17:00</ArrivesTime>
        <Price>3000</Price>
    </Flight>
    <Flight>
        <FlNo>FL104</FlNo>
        <FlName>Flight D</FlName>
        <PilotName>Black Widow</PilotName>
        <From>Houston</From>
        <To>Seattle</To>
        <Date>2024-12-02</Date>
        <DepartsTime>11:00</DepartsTime>
        <ArrivesTime>14:00</ArrivesTime>
        <Price>7000</Price>
    </Flight>
    <Flight>
        <FlNo>FL105</FlNo>
        <FlName>Flight E</FlName>
        <PilotName>Thor</PilotName>
        <From>Miami</From>
        <To>Atlanta</To>
        <Date>2024-12-03</Date>
        <DepartsTime>12:00</DepartsTime>
        <ArrivesTime>14:30</ArrivesTime>
        <Price>4800</Price>
    </Flight>
</FlightDetails>
```

run_queries.sh

```
#!/bin/bash

# Define XML files

EMPLOYEE_XML="EmployeeDetails.xml"

FLIGHT_XML="FlightDetails.xml"

echo "=== Employee Queries ==="

# 1. List salaries > 30000

echo "Salaries > 30000:"

xmlstarlet sel -t -m "/EmployeeDetails/Employee[Salary > 30000]" -v "Salary" -n "$EMPLOYEE_XML"

# 2. Get employee numbers with last name starting with "S"

echo "Employee numbers with last name starting with 'S':"

xmlstarlet sel -t -m "/EmployeeDetails/Employee[starts-with(EName, 'S')]" -v "EmpNo" -n "$EMPLOYEE_XML"

# 3. Get names of employees in the "Research" department

echo "Names of employees in 'Research' department:"xmlstarlet sel -t -m "/EmployeeDetails/Employee[Dept = 'Research']" -v "EName" -n "$EMPLOYEE_XML"

# 4. Get employees who work for more than 8 hours

echo "Employees working for more than 8 hours:"

xmlstarlet sel -t -m "/EmployeeDetails/Employee[WorkingHours > 8]" -v "." -n "$EMPLOYEE_XML"

# 5. Display salary from highest to lowest

echo "Salaries from highest to lowest:"

xmlstarlet sel -t -m "/EmployeeDetails/Employee" -v "Salary" -n "$EMPLOYEE_XML" | sort -nr

# 6. Display employee names in alphabetical order

echo "Employee names in alphabetical order:"
```

```
xmlstarlet sel -t -m "/EmployeeDetails/Employee" -v "EName" -n "$EMPLOYEE_XML" | sort

echo ""

echo "=== Flight Queries ==="

# 1. List price of journeys < 5000

echo "Prices of journeys < 5000:"

xmlstarlet sel -t -m "/FlightDetails/Flight[Price < 5000]" -v "Price" -n "$FLIGHT_XML"

# 2. Find the departing time of a particular flight on a particular date

echo "Departing time of FL101 on 2024-12-01 from New York:"

xmlstarlet sel -t -m "/FlightDetails/Flight[FlNo = 'FL101' and Date = '2024-12-01' and From = 'New York']" -v "DepartsTime" -n "$FLIGHT_XML"

# 3. Find the flight names handled by a particular pilot

echo "Flight names handled by 'Iron Man':"

xmlstarlet sel -t -m "/FlightDetails/Flight[PilotName = 'Iron Man']" -v "FlName" -n "$FLIGHT_XML"

# 4. Count the number of flights for a particular flight on a particular date

echo "Count of FL102 flights on 2024-12-01:"

xmlstarlet sel -t -m "/FlightDetails/Flight[FlNo = 'FL102' and Date = '2024-12-01']" -v "count(.)" -n "$FLIGHT_XML"

# 5. Find the arrival time of a flight

echo "Arrival time of FL103 on 2024-12-02 from San Francisco:"

xmlstarlet sel -t -m "/FlightDetails/Flight[FlNo = 'FL103' and Date = '2024-12-02' and From = 'San Francisco']" -v "ArrivesTime" -n "$FLIGHT_XML"
```

```
rana@DESKTOP-4TF54EU:/mnt/c/Users/Sadhguna/OneDrive/Desktop/hi$ ./run_queries.sh
=== Employee Queries ===
Salaries > 30000:
45000
32000
50000
40000
35000
Employee numbers with last name starting with 'S':
104
Names of employees in 'Research' department:
Jane Doe
Linda Smith
```

```
Employees working for more than 8 hours:

        101
        John Smith
        Manager
        9
        Sales
        1
        45000


        103
        Mark Spencer
        Analyst
        10
        Finance
        3
        50000


        105
        Linda Smith
        Tester
        9
        Research
        2
        35000
```

```
Salaries from highest to lowest:
50000
45000
40000
35000
32000
Employee names in alphabetical order:
Jane Doe
John Smith
Linda Smith
Mark Spencer
Sarah Johnson

=== Flight Queries ===
Prices of journeys < 5000:
4500
3000
4800
Departing time of FL101 on 2024-12-01 from New York:
08:00
Flight names handled by 'Iron Man':
Flight C
Count of FL102 flights on 2024-12-01:
1
Arrival time of FL103 on 2024-12-02 from San Francisco:
17:00
```

Q2.

a. Display Details of an Employee by Employee ID

```
CREATE PROCEDURE DisplayEmployeeDetails(IN emp_id INT)
BEGIN
    SELECT * FROM Employees WHERE SSN = emp_id;
END;
```

b. Add Details of a New Employee

```
CREATE PROCEDURE AddEmployee(
    IN emp_id INT, IN fname VARCHAR(50), IN lname VARCHAR(50), IN bdate DATE,
    IN address VARCHAR(100), IN sex CHAR(1), IN salary DECIMAL(10,2),
    IN super_ssn INT, IN dept_no INT
)
BEGIN
    INSERT INTO Employees (SSN, FName, LName, BDate, Address, Sex, Salary, SuperSSN, DNo)
    VALUES (emp_id, fname, lname, bdate, address, sex, salary, super_ssn, dept_no);
END;
```

c. Increase Employee Salary

```
CREATE PROCEDURE RaiseSal(IN emp_id INT, IN hike_amount DECIMAL(10,2))
BEGIN
    UPDATE Employees
    SET Salary = Salary + hike_amount
    WHERE SSN = emp_id;
END;
```

d. Delete Employee by Name

```
CREATE PROCEDURE DeleteEmployeeByName(IN emp_name VARCHAR(50))
BEGIN
    DELETE FROM Employees WHERE CONCAT(FName, ' ', LName) = emp_name;
END;
```

e. List Employees by Department Number

```
CREATE PROCEDURE ListEmployeesByDept(IN dept_no INT)
BEGIN
    SELECT FName, LName FROM Employees WHERE DNo = dept_no;
END;
```

f. Highest Salary in Each Department

Step 1: Procedure dept_highest

```
CREATE PROCEDURE dept_highest(IN dept_no INT, OUT max_salary DECIMAL(10,2))
BEGIN
    SELECT MAX(Salary) INTO max_salary FROM Employees WHERE DNo = dept_no;
END;
```

Step 2: Procedure to List Highest Salary in Each Department

```
CREATE PROCEDURE ListDeptHighestSalaries()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE dept_no INT;
    DECLARE max_salary DECIMAL(10,2);
    DECLARE dept_cursor CURSOR FOR SELECT DISTINCT DNo FROM Employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN dept_cursor;

    read_loop: LOOP
        FETCH dept_cursor INTO dept_no;
        IF done THEN
            LEAVE read_loop;
        END IF;

        CALL dept_highest(dept_no, max_salary);
        SELECT dept_no AS Department, max_salary AS HighestSalary;
    END LOOP;

    CLOSE dept_cursor;
END;
```

g. Minimum Salary of Employees

```
CREATE FUNCTION MinSalary() RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE min_salary DECIMAL(10,2);
    SELECT MIN(Salary) INTO min_salary FROM Employees;
    RETURN min_salary;
END;
```

h. Number of Employees in the Organization

```
CREATE FUNCTION EmployeeCount() RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE emp_count INT;
    SELECT COUNT(*) INTO emp_count FROM Employees;
    RETURN emp_count;
END;
```

i. Display Salary of an Employee by Employee ID

```
CREATE FUNCTION GetEmployeeSalary(emp_id INT) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE salary DECIMAL(10,2);
    SELECT Salary INTO salary FROM Employees WHERE SSN = emp_id;
    RETURN salary;
END;
```

j. Average Salary by Department

```
CREATE FUNCTION AvgSalaryByDept(dept_no INT) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE avg_salary DECIMAL(10,2);
    SELECT AVG(Salary) INTO avg_salary FROM Employees WHERE DNo = dept_no;
    RETURN avg_salary;
END;
```

k. Count of Employees with Salary More Than 30,000

```
CREATE FUNCTION CountHighSalaryEmployees() RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE count_high_salary INT;
    SELECT COUNT(*) INTO count_high_salary FROM Employees WHERE Salary > 30000;
    RETURN count_high_salary;
END;
```

l. Count of Employees in Tiruchirappalli

```sql
CREATE FUNCTION CountEmployeesInTiruchirappalli() RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE count_trichy INT;
    SELECT COUNT(*) INTO count_trichy FROM Employees WHERE Address LIKE '%Tiruchirappalli%';
    RETURN count_trichy;
END;
```

Q3.

Disable Autocommit

By default, MySQL commits changes automatically. To prevent this, set autocommit to 0 (off) for the current session:

```sql
SET autocommit = 0;
```

Start a Transaction

Begin a new transaction, allowing for multiple SQL statements to be executed as a single unit:

```sql
START TRANSACTION;
```

Update an Employee's Salary

Update the Salary of an employee (with SSN 101 for example) in the Employees table:

```sql
UPDATE Employees
SET Salary = Salary + 5000
WHERE SSN = 101;
```

Create a Savepoint

A SAVEPOINT allows you to mark a point within a transaction to which you can later roll back if needed:

```sql
SAVEPOINT update_salary;
```

Make Another Update

Make another update, perhaps changing the Address of the same employee, to demonstrate rolling back to a savepoint:

```
UPDATE Employees
SET Address = 'New Address, Tiruchirappalli'
WHERE SSN = 101;
```

Rollback to Savepoint

If you decide that only the first update is needed (the Salary change) but not the address change, you can roll back to the update_salary savepoint:

```
ROLLBACK TO update_salary;
```

Commit the Transaction

After finalizing all desired changes, use COMMIT to make the transaction's changes permanent:

```
COMMIT;
```

Enable Autocommit Again

To return to the default behavior where each SQL statement is committed automatically, re-enable autocommit:

```
SET autocommit = 1;
```