Assignment: SQL Tasks for SpecialForce Private Limited - Database Operations (MySQL)

Scenario:

SpecialForce Private Limited is expanding its workforce and needs help with managing its employee records, departments, and ongoing projects. As a fresh database consultant, your task is to create and manipulate the database to manage their growing employee, department, and project data.

Tasks:

Task 1: Create Tables

1. **Create three tables**: Employees, Departments, and Projects to track employees, departments, and projects, respectively.
    - Ensure each table has a **Primary Key** for uniquely identifying records.
    - Set up **Foreign Key** constraints to link employees to departments and projects.
    - Use appropriate **constraints** (e.g., NOT NULL, UNIQUE, etc.) to maintain data integrity.

```
CREATE TABLE Employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT,
    project_id INT,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id),
    FOREIGN KEY (project_id) REFERENCES Projects(project_id)
);

CREATE TABLE Departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE Projects (
    project_id INT AUTO_INCREMENT PRIMARY KEY,
    project_name VARCHAR(100) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE
);
```

Task 2: Insert Data (Given in excel sheet)

Once you have created the tables, insert the provided data into the respective tables. The data contains details about employees, departments, and projects.

```
INSERT INTO Departments (department_id, department_name)
VALUES
(1, 'IT'),
(2, 'HR'),
(3, 'Sales'),
(4, 'Finance'),
(5, 'Marketing');

INSERT INTO Projects (project_id, project_name, start_date, end_date, department_id)
VALUES
(201, 'Project Phoenix', '2021-01-15', '2022-07-30', 1),
(202, 'Client Onboarding', '2020-06-20', NULL, 3),
(203, 'Financial Overhaul', '2019-03-10', '2021-12-15', 4),
(204, 'Marketing Revamp', '2022-03-01', NULL, 5),
(205, 'Internal System Audit', '2023-02-15', NULL, 2);

INSERT INTO Employees (employee_id, first_name, last_name, email, hire_date, salary,
department_id)
VALUES
(101, 'Ravi', 'Sharma', 'ravi.sharma@specialforce.com', '2017-05-15', 55000, 1),
(102, 'Neha', 'Kapoor', 'neha.kapoor@specialforce.com', '2019-03-23', 48000, 2),
(103, 'Jyoti', 'Verma', 'jyoti.verma@specialforce.com', '2020-11-02', 60000, 1),
(104, 'Anil', 'Patil', 'anil.patil@specialforce.com', '2018-09-18', 70000, 3),
(105, 'Pooja', 'Singh', 'pooja.singh@specialforce.com', '2021-06-10', 40000, 4),
(106, 'Sanjay', 'Iyer', 'sanjay.iyer@specialforce.com', '2018-01-22', 75000, 3),
(107, 'Jatin', 'Reddy', 'jatin.reddy@specialforce.com', '2021-12-12', 85000, 2),
(108, 'Shreya', 'Mehta', 'shreya.mehta@specialforce.com', '2022-04-19', 30000, 5),
(109, 'Rajesh', 'Gupta', 'rajesh.gupta@specialforce.com', '2020-08-11', 90000, 1),
(110, 'Kavita', 'Nair', 'kavita.nair@specialforce.com', '2021-02-07', 50000, 2);
```

Queries to Perform:

Query 1: Write a query to retrieve the first name, last name, and department name of all employees. If an employee does not belong to any department, the department name should be NULL.

```
SELECT e.first_name, e.last_name, d.department_name
FROM Employees e
LEFT JOIN Departments d ON e.department_id = d.department_id;
```

Query 2: Write a query to find all employees in the IT department who earn more than ₹50,000.

```
SELECT first_name, last_name, salary
FROM Employees
WHERE department_id = 1 AND salary > 50000;
```

Query 3: Write a query to list the first name, last name, and email of all employees whose first name starts with 'J' and whose email contains specialforce.com.

```
SELECT first_name, last_name, email
FROM Employees
WHERE first_name LIKE 'J%'
AND email LIKE '%specialforce.com%';
```

Query 4: Write a query to find all the distinct department names in the Departments table.

```
SELECT DISTINCT department_name
FROM Departments;
```

Query 5: Write a query to calculate the total salary expenditure of each department.

```
SELECT d.department_name, SUM(e.salary) FROM Employees e
JOIN Departments d ON e.department_id = d.department_id
GROUP BY d.department_name;
```

Query 6: Write a query to find the average salary of employees in the Finance department.

```
SELECT AVG(salary) FROM Employees
WHERE department_id = (SELECT department_id from Departments WHERE department_name
= ‘FINANCE’);
```

Query 7: Write a query to find the minimum and maximum salaries of employees in the Sales department.

```
SELECT MIN(salary), MAX(salary) FROM Employees
WHERE department_id = (SELECT department_id from Departments WHERE department_name
= ‘SALES’);
```

Query 8: Write a query to count the number of employees in each department.

```
SELECT d.department_name, COUNT(e.employee_id) FROM Departments d
LEFT JOIN Employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

Query 9: Write a query to find all employees who were hired between January 1, 2018, and December 31, 2020. Sort the result by hire date in ascending order.

```
SELECT first_name, last_name, hire_date FROM Employees
WHERE hire_date BETWEEN '2018-01-01' AND '2020-12-31'
ORDER BY hire_date ASC;
```

Query 10: Write a query to list all employees who do not have an email address.

```
SELECT first_name, last_name FROM Employees
WHERE email IS NULL;
```

Query 11: Write a query to find all employees who work in HR, Finance, or IT departments.

```
SELECT first_name, last_name , department_name FROM Employees e, Departments d
WHERE e.department_id = d.department_id and department_name IN ('HR', 'FINANCE', 'IT');
```

Query 12: Write a query to list the first name, last name, and salary of employees earning between ₹30,000 and ₹70,000. Sort the results by salary in descending order.

```
SELECT first_name, last_name, salary FROM Employees
WHERE salary BETWEEN 30000 AND 70000
ORDER BY salary DESC;
```

Transaction Management Tasks:

Use transaction control statements to manage the salary updates as follows:

Task 1: Increase HR Salaries:
Write a query to increase the salaries of all employees in the HR department by 5%. Start a transaction before applying the changes.

```
START TRANSACTION;

UPDATE Employees
SET salary = salary + salary * 0.05
WHERE department_id = (SELECT department_id from Departments WHERE
department_name = 'HR');
```

Task 2: Savepoint Before Sales Increase:
Set a savepoint before increasing the salaries of employees in the Sales department by 3%.

SAVEPOINT sales_increase_savepoint;

Task 3: Rollback Sales Salary Increase:
Rollback to the savepoint created before the Sales salary increase.

ROLLBACK TO SAVEPOINT sales_increase_savepoint;

Task 4: Commit the Transaction:
After rolling back the Sales increase, commit the changes made to the HR department salaries.

Commit;

Query 13: Write a query to join the Employees and Departments tables to list employees and their department names. Make sure all employees are included, even if they don't belong to any department.

```
SELECT e.first_name, e.last_name, d.department_name FROM Employees e
LEFT JOIN Departments d ON e.department_id = d.department_id;
```

Query 14: Write a query to list employees who are working on projects that started after January1, 2023.

```
SELECT e.first_name, e.last_name, p.project_name FROM Employees e
JOIN Projects p ON e.department_id = p.department_id
WHERE p.start_date > '2023-01-01';
```

Query 15: Write a query to list all departments, even those without any employees assigned

```
SELECT d.department_name, COUNT(e.employee_id)  FROM Departments d
LEFT JOIN Employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

Query 16: Write a query to find the employee with the highest salary in each department.

```
SELECT d.department_name, e.first_name, e.last_name, e.salary FROM Employees e
JOIN Departments d ON e.department_id = d.department_id
WHERE (e.salary, e.department_id) IN (
   SELECT MAX(salary), department_id
   FROM Employees
   GROUP BY department_id
);
```

Query 17: Write a query to remove all data from the Employees table but keep the structure intact.

```
TRUNCATE TABLE Employees;
```

Query 18: Write a query to drop the Projects table from the database.

```
DROP TABLE Projects;
```

Query 19: SpecialForce Private Limited realized they need to store the phone numbers of employees. Write a query to add a new column phone_number (VARCHAR(15)) to the Employees table using the ALTER statement.

```
ALTER TABLE Employees
ADD COLUMN phone_number VARCHAR(15);
```

Query 20: The company also decided to track the budget for each project. Write a query to add a column budget (DECIMAL(10,2)) to the Projects table.

```
ALTER TABLE Projects
ADD COLUMN budget DECIMAL(10, 2);
```

Query 21: Write a query to find the 2nd largest salary from the Employees table using:

- A **subquery**.
- The LIMIT clause.

```
SELECT MAX(salary) FROM Employees
WHERE salary < (SELECT MAX(salary) FROM Employees);
```

Query 22: Write a query to find the 3rd largest salary from the Employees table using:

- A **subquery**.
- The LIMIT clause.

```
SELECT salary FROM Employees
ORDER BY salary DESC
LIMIT 1 OFFSET 2;
```

Query 23: Write a query to drop the Projects table.

```
DROP TABLE Projects;
```

Query 24: Write a query to truncate the Employees table.

```
TRUNCATE TABLE Employees;
```