Name: Srivartshan M

Date: 23/11/2024

Basic SQL Queries

1. **Display all rows and columns from the employees table:**

```
SELECT * FROM employees;
```

2. **Retrieve only the name and salary of all employees from the employees table:**

```
SELECT name, salary FROM employees;
```

3. **Find all employees whose salary is greater than 50,000:**

```
SELECT * FROM employees WHERE salary > 50000;
```

4. **List all employees who joined the company in the year 2020:**

```
SELECT * FROM employees WHERE YEAR(joined_date) = 2020;
```

5. **Retrieve the details of employees whose names start with the letter 'A':**

```
SELECT * FROM employees WHERE name LIKE 'A%';
```

Aggregate Functions

1. **Calculate the average salary of all employees:**

```
SELECT AVG(salary) AS average_salary FROM employees;
```

2. **Find the total number of employees in the company:**

```
SELECT COUNT(*) AS total_employees FROM employees;
```

3. **Find the highest salary in the employees table:**

```
SELECT MAX(salary) AS highest_salary FROM employees;
```

4. **Calculate the total salary paid by the company for all employees:**

```
SELECT SUM(salary) AS total_salary FROM employees;
```

5. **Find the count of employees in each department:**

```
SELECT department_id, COUNT(*) AS num_employees FROM employees GROUP BY
department_id;
```

Joins

1. **Retrieve employee names along with their department names:**

```
SELECT e.name AS employee_name, d.name AS department_name
FROM employees e
JOIN departments d ON e.department_id = d.id;
```

2. **List all employees who have a manager (self-join on employees table):**

```
SELECT e.name AS employee_name, m.name AS manager_name
FROM employees e
JOIN employees m ON e.manager_id = m.id;
```

3. **Find the names of employees who are working on multiple projects:**

```
SELECT e.name
FROM employees e
JOIN employee_projects ep ON e.id = ep.employee_id
JOIN projects p ON ep.project_id = p.id
GROUP BY e.name
HAVING COUNT(DISTINCT p.id) > 1;
```

4. **Display all projects and the employees assigned to them:**

```
SELECT p.project_name, e.name AS employee_name
FROM projects p
JOIN employee_projects ep ON p.id = ep.project_id
JOIN employees e ON ep.employee_id = e.id;
```

5. **Retrieve the names of employees who do not belong to any department:**

```
SELECT name FROM employees WHERE department_id IS NULL;
```

Subqueries

1. **Find the employees with the second-highest salary:**

```
SELECT name FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees WHERE salary < (SELECT
MAX(salary) FROM employees));
```

2. **Retrieve the names of employees whose salary is above the department average salary:**

```
SELECT e.name
FROM employees e
WHERE e.salary > (SELECT AVG(salary) FROM employees WHERE department_id =
e.department_id);
```

3. **Find employees who earn more than the average salary of the entire company:**

```
SELECT name FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

4. **Find the department with the highest number of employees:**

```
SELECT department_id
FROM employees
GROUP BY department_id
ORDER BY COUNT(*) DESC
LIMIT 1;
```

5. **List all employees who work in a department located in 'New York':**

```
SELECT e.name
FROM employees e
JOIN departments d ON e.department_id = d.id
WHERE d.location = 'New York';
```

Set Operators

1. **Find employees who work in either the 'HR' or 'Finance' department:**

```
SELECT name FROM employees WHERE department_id IN (SELECT id FROM departments
WHERE name = 'HR')
UNION
SELECT name FROM employees WHERE department_id IN (SELECT id FROM departments
WHERE name = 'Finance');
```

2. **Retrieve the names of employees who are working on both Project A and Project B:**

```
SELECT e.name
FROM employees e
JOIN employee_projects ep ON e.id = ep.employee_id
JOIN projects p ON ep.project_id = p.id
WHERE p.project_name IN ('Project A', 'Project B')
GROUP BY e.name
HAVING COUNT(DISTINCT p.project_name) = 2;
```

3. **Find employees who are not assigned to any project:**

```
SELECT e.name
FROM employees e
LEFT JOIN employee_projects ep ON e.id = ep.employee_id
WHERE ep.project_id IS NULL;
```

4. **Get all unique job titles across all departments:**

```
SELECT DISTINCT job_title FROM employees;
```

5. **Combine two tables (employees and former_employees) and remove duplicates:**

```
SELECT name FROM employees
UNION
SELECT name FROM former_employees;
```

DML and DDL

1. **Add a new employee to the employees table:**

```
INSERT INTO employees (name, salary, department_id, joined_date)
VALUES ('John Doe', 60000, 1, '2024-01-01');
```

2. **Update the salary of all employees in the 'IT' department by 10%:**

```
UPDATE employees
SET salary = salary * 1.10
WHERE department_id = (SELECT id FROM departments WHERE name = 'IT');
```

3. **Delete all employees who have not worked for more than 5 years:**

```
DELETE FROM employees WHERE DATEDIFF(CURRENT_DATE, joined_date) > 1825;
```

4. **Create a new table departments_backup with the same structure as the departments table:**

```
CREATE TABLE departments_backup AS SELECT * FROM departments WHERE 1 = 0;
```

5. **Drop the temporary_data table from the database:**

```
DROP TABLE temporary_data;
```

Constraints

1. **Add a primary key to the employees table:**

```
ALTER TABLE employees
ADD CONSTRAINT pk_employee_id PRIMARY KEY (id);
```

2. **Create a foreign key between employees and departments tables:**

```
ALTER TABLE employees
ADD CONSTRAINT fk_department_id FOREIGN KEY (department_id) REFERENCES
departments(id);
```

3. **Add a unique constraint to the email column in the employees table:**

```
ALTER TABLE employees
ADD CONSTRAINT unique_email UNIQUE (email);
```

4. **Check all constraints applied on the employees table:**

```
SHOW CREATE TABLE employees;
```

5. **Remove the NOT NULL constraint from the phone_number column in the employees table:**

```
ALTER TABLE employees
MODIFY phone_number VARCHAR(15) NULL;
```