Experiment No: 8

Name: Prashant Rajendra Dhond **Roll No. : 23**
Batch: A **Performance Date : 24-3-25**

Topic:	Perform Sub Queries, Nested Queries and Joins.
Prerequisite:	Knowledge of concepts sub query, nested query, Joins and SQL syntax.
Mapping With COs:	CSL402.2, CSL402.3
Objective:	- To implement Subqueries, Nested Queries and Joins. - Write different types of problems that can solve by: - Sub queries - Nested queries - Combine data across tables according to their system. (Implement JOIN)

```
mysql> create database factory;
Query OK, 1 row affected (0.04 sec)

mysql> use factory;
Database changed
mysql> CREATE TABLE Employees (
    ->     emp_id INT PRIMARY KEY,
    ->     emp_name VARCHAR(50),
    ->     salary INT,
    ->     dept_id INT
    -> );
Query OK, 0 rows affected (0.16 sec)

mysql>
mysql> CREATE TABLE Departments (
    ->     dept_id INT PRIMARY KEY,
    ->     dept_name VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> CREATE TABLE Orders (
    ->     order_id INT PRIMARY KEY,
    ->     customer_id INT,
    ->     amount INT
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> CREATE TABLE Customers (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql>
```

```
mysql> INSERT INTO Employees (emp_id, emp_name, salary, dept_id) VALUES
    -> (101, 'Alice', 60000, 1),
    -> (102, 'John', 70000, 2),
    -> (103, 'Mike', 40000, NULL);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM Customers;
Empty set (0.00 sec)

mysql> SELECT * FROM Orders;
Empty set (0.00 sec)

mysql> INSERT INTO Customers (customer_id, customer_name) VALUES
    -> (1, 'Alice'),
    -> (2, 'Bob'),
    -> (3, 'Charlie'),
    -> (4, 'David');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Orders (order_id, customer_id, amount) VALUES
    -> (101, 1, 7000),
    -> (102, 2, 4000),
    -> (103, 3, 9000),
    -> (104, 4, 3000);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

1. Subqueries

A **subquery** is a query inside another query.

Problem 1: Find employees earning more than the average salary.

```
mysql> SELECT emp_name, salary FROM Employees
    -> WHERE salary > (SELECT AVG(salary) FROM Employees);
+----------+--------+
| emp_name | salary |
+----------+--------+
| Alice    |  60000 |
| John     |  70000 |
+----------+--------+
2 rows in set (0.00 sec)

mysql>
```

Problem 2: Find the second highest salary.

```
mysql> SELECT MAX(salary) AS second_highest_salary FROM Employees
    -> WHERE salary < (SELECT MAX(salary) FROM Employees);
+-----------------------+
| second_highest_salary |
+-----------------------+
|                 60000 |
+-----------------------+
1 row in set (0.00 sec)
```

2. Nested Queries

A **nested query** is where the inner query provides results used by the outer query.

Problem 3: Find customers who have placed an order greater than ₹5000.

```
mysql> SELECT customer_name FROM Customers
    -> WHERE customer_id IN (SELECT customer_id FROM Orders WHERE amount > 5000);
+---------------+
| customer_name |
+---------------+
| Alice         |
| Charlie       |
+---------------+
2 rows in set (0.00 sec)
```

Problem 4: Fetch employee details who work in the 'IT' department

```
mysql> SELECT * FROM Employees
    -> WHERE dept_id = (SELECT dept_id FROM Departments WHERE dept_name = 'IT');
+--------+----------+--------+---------+
| emp_id | emp_name | salary | dept_id |
+--------+----------+--------+---------+
|    101 | Alice    |  60000 |       1 |
+--------+----------+--------+---------+
1 row in set (0.00 sec)
```

3. Joins

Joins are used to combine records from two or more tables.

Problem 5: Fetch employees along with their department names (INNER JOIN).

```
mysql> SELECT e.emp_name, d.dept_name FROM Employees e
    -> INNER JOIN Departments d ON e.dept_id = d.dept_id;
+----------+-----------+
| emp_name | dept_name |
+----------+-----------+
| Alice    | IT        |
| John     | HR        |
+----------+-----------+
2 rows in set (0.00 sec)
```

Problem 6: List all employees with department names, including employees without departments (LEFT JOIN).

```
mysql> SELECT e.emp_name, d.dept_name FROM Employees e
    -> LEFT JOIN Departments d ON e.dept_id = d.dept_id;
+----------+-----------+
| emp_name | dept_name |
+----------+-----------+
| Alice    | IT        |
| John     | HR        |
| Mike     | NULL      |
+----------+-----------+
3 rows in set (0.00 sec)
```

Problem 7: Fetch customers and their order details (RIGHT JOIN).

```
mysql> SELECT c.customer_name, o.order_id, o.amount FROM Customers c
    -> RIGHT JOIN Orders o ON c.customer_id = o.customer_id;
+---------------+----------+--------+
| customer_name | order_id | amount |
+---------------+----------+--------+
| Alice         |      101 |   7000 |
| Bob           |      102 |   4000 |
| Charlie       |      103 |   9000 |
| David         |      104 |   3000 |
+---------------+----------+--------+
4 rows in set (0.00 sec)
```

Outcome:	After completion of this lab, the students will understand and be able to do the following: - Describe the types of problems that subqueries can solve. - Sub queries are nested within a SELECT, INSERT, UPDATE, or DELETE statement. - A subquery can be used inside the WHERE or HAVING clauses of the outer SELECT, INSERT, UPDATE, or DELETE statements. - Build and execute sub query. - Define and execute various types of joins.
Instructions:	1. This experiment is a compulsory experiment. All the students are required to perform this experiment individually. 2. Implement Subqueries, Nested Queries and all the types of Joins for the assigned system.
Deliverables:	**For Submissions:** 1. Implement Subqueries, Nested Queries and all the types of Joins for the assigned system. (All implemented queries with output snapshots) 2. Viva based on Subqueries, Nested Queries and all the types of Joins.
Conclusion:	In this experiment, students will understand and be able to do the following: ● Describe the types of problems that subqueries can solve. ● Sub queries are nested within a SELECT, INSERT, UPDATE, or DELETE statement. ● A subquery can be used inside the WHERE or HAVING clauses of the outer SELECT, INSERT, UPDATE, or DELETE statements. ● Build and execute sub query. ● Define and execute various types of joins.

References:	/ ELEARN MOODLE

Don Bosco Institute of Technology
Department of Computer Engineering

Assessment Rubric for Experiment No. 8

Title of Experiment : Perform Sub Queries, Nested Queries and Joins.
Year and Semester : 2nd Year and IVth Semester

Sr. No.	**Criteria**	**1 Marks**	**2 Marks**	**3 Marks**	**4 Marks**	**5 Marks**
1	Execution	Executed 10-30% queries based on following: -Sub query - nested querying - All Types of Joins	Executed 31-50% queries based on following: -Sub query - nested querying - All Types of Joins	Executed 51-70% queries based on following: -Sub query - nested querying - All Types of Joins	Executed 71-89% queries based on following: -Sub query - nested querying - All Types of Joins	Executed 90-100% queries based on following: -Sub query - nested querying - All Types of Joins
2	Documentation	20-39% of solutions are documented properly.	40-59% of solutions are documented properly.	60-79% of solutions are documented properly.	80-100% of the solution is documented properly.	
3	Viva	Students hardly answered.	Students have problems while answering.	Questions are answered fairly well.	Questions are answered completely and correctly.	
4	Submission on Time	Submitted after the given deadline	Submitted before the given deadline			