Week 10

Creating Stored Procedures & Triggers

AIM: Implement Sub Programs (Stored Procedures) in PL/SQL.

1. Create a Procedure, which receives employee number and displays employee name, Designation and salary.

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE GetEmployeeDetails(IN EmpNoParam INT)
    -> BEGIN
    -> SELECT Ename, Job, Sal
    -> FROM Employee
    -> WHERE Empno = EmpNoParam;
    -> END $$
Query OK, 0 rows affected (0.02 sec)

mysql> call GetEmployeeDetails(7900);
    -> $$
+-------+-------+-------+
| Ename | Job   | Sal   |
+-------+-------+-------+
| JAMES | CLERK | 25000 |
+-------+-------+-------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

2. Create a procedure, which accepts employee number and returns employee name and department name in an out parameter.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetEmployeeAndDept(IN EmpNo INT, OUT EmpName VARCHAR(50), OUT DeptName VARCHAR(50))
    -> BEGIN
    -> SELECT Ename, Dname
    -> INTO EmpName, DeptName
    -> FROM Employee E
    -> JOIN Department D ON E.Deptno = D.Deptno
    -> WHERE E.Empno = EmpNo;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> call GetEmployeeAndDept(7900, @x,@y);
Query OK, 1 row affected (0.01 sec)

mysql> select @x, @y;
+-------+-------+
| @x    | @y    |
+-------+-------+
| JAMES | SALES |
+-------+-------+
1 row in set (0.00 sec)
```

3. Create a procedure, which takes the department number and gets the total salary of that department.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetTotalSalaryByDept(IN DeptNo INT)
    -> BEGIN
    -> SELECT SUM(Sal) AS TotalSalary
    -> FROM Employee
    -> WHERE Deptno = DeptNo;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> CALL GetTotalSalaryByDept(20);
+-------------+
| TotalSalary |
+-------------+
|      411000 |
+-------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

4. Create a procedure to accept Department number and display Name, Designation, Sal and Deptno of each employee belonging to such Department.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE getEmployeesByDept(dept_no INT)
    -> BEGIN
    -> SELECT Ename, Job, Sal, Deptno
    -> FROM Employee
    -> WHERE Deptno = dept_no;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> call getEmployeesByDept(10);
+--------+-----------+-------+--------+
| Ename  | Job       | Sal   | Deptno |
+--------+-----------+-------+--------+
| CLARK  | MANAGER   | 45000 |     10 |
| KING   | PRESIDENT | 50000 |     10 |
| MILLER | CLERK     | 23000 |     10 |
+--------+-----------+-------+--------+
3 rows in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)
```

5. Create a procedure to get a cube of a passed number.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE getCube(num INT, OUT result INT)
    -> BEGIN
    ->     SET result = num * num * num;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> call getCube(4);
ERROR 1318 (42000): Incorrect number of arguments for PROCEDURE week10.getCube; expected 2, got 1
mysql> call getCube(4, @cube);
Query OK, 0 rows affected (0.01 sec)

mysql> select @cube;
+-------+
| @cube |
+-------+
|    64 |
+-------+
1 row in set (0.01 sec)
```

6. Create a procedure to find out the maximum salary for the passed designation.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE getMaxSalaryByJob(job_title VARCHAR(50), OUT max_salary FLOAT)
    -> BEGIN
    ->     SELECT MAX(Sal) INTO max_salary
    ->     FROM Employee
    ->     WHERE Job = job_title;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> call getMaxSalaryByJob(CLERK, @sal);
ERROR 1054 (42S22): Unknown column 'CLERK' in 'field list'
mysql> call getMaxSalaryByJob('SALESMAN', @sal);
Query OK, 1 row affected (0.01 sec)

mysql> select @sal;
+-------+
| @sal  |
+-------+
| 25000 |
+-------+
1 row in set (0.00 sec)
```

7. Create a procedure to find out Total salary for the passed department Name

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE getTotalSalaryByDeptName(dept_name VARCHAR(50), OUT total_salary FLOAT)
    -> BEGIN
    ->     SELECT SUM(Sal) INTO total_salary
    ->     FROM Employee
    ->     WHERE Deptno = (SELECT Deptno FROM Department WHERE Dname = dept_name);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> call getTotalSalaryByDeptName('RESEARCH', @tsal);
Query OK, 1 row affected (0.01 sec)

mysql> select @tsal;
+--------+
| @tsal  |
+--------+
| 133000 |
+--------+
1 row in set (0.00 sec)
```

8. Create a procedure to check whether the passed number is Odd or Even.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE checkOddEven(num INT, OUT result VARCHAR(10))
    -> BEGIN
    ->     IF num % 2 = 0 THEN
    ->         SET result = 'Even';
    ->     ELSE
    ->         SET result = 'Odd';
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> call checkOddEven(177, @result);
Query OK, 0 rows affected (0.00 sec)

mysql> select @result;
+---------+
| @result |
+---------+
| Odd     |
+---------+
1 row in set (0.00 sec)
```

9. Create a procedure to find out total annual income for the employee, who's number we passed as argument.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE getAnnualIncome(emp_no INT, OUT annual_income FLOAT)
    -> BEGIN
    ->     SELECT Sal * 12 INTO annual_income
    ->     FROM Employee
    ->     WHERE Empno = emp_no;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> call getAnnualIncome(7698, @asal);
Query OK, 1 row affected (0.00 sec)

mysql> select @asal;
+---------+
| @asal   |
+---------+
| 600000 |
+---------+
1 row in set (0.00 sec)
```

10. Create a procedure, accept 2 numbers as input, and operator also input and the result as output.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE calculator(num1 FLOAT, num2 FLOAT, operator CHAR(1), OUT result FLOAT)
    -> BEGIN
    ->     IF operator = '+' THEN
    ->         SET result = num1 + num2;
    ->     ELSEIF operator = '-' THEN
    ->         SET result = num1 - num2;
    ->     ELSEIF operator = '*' THEN
    ->         SET result = num1 * num2;
    ->     ELSEIF operator = '/' THEN
    ->         SET result = num1 / num2;
    ->     ELSE
    ->         SET result = NULL;
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> call calculator(4,23,'*',@res);
Query OK, 0 rows affected (0.00 sec)

mysql> select @res;
+------+
| @res |
+------+
|   92 |
+------+
1 row in set (0.00 sec)
```

Problem 1: Log Deleted Employee Records

Objective: Create a trigger that logs any deletion of employee records into a separate log table to keep track of deletions for auditing purposes.

Context: To maintain a history of deleted employees for audit and analysis purposes, the company wants to store a record every time an employee is deleted. This record should include the employee's number, name, and the timestamp of when the deletion occurred.

Task:

1. **Create a table named EmpLog that will store deleted employee information.**
    - **The table should have the following columns:**
        - **ActionType (VARCHAR): This should store a string indicating the action performed ('DELETE' in this case).**
        - **Empno (INT): Employee number.**
        - **Ename (VARCHAR): Employee name.**
        - **ActionDate (DATETIME): Timestamp when the action was performed.**
2. **Create a trigger named EMP_DELETED_DATA that automatically inserts a record into the EmpLog table every time an employee is deleted from the Employee table.**
    - **The ActionType should be set to 'DELETE'.**
    - **The Empno and Ename should be retrieved from the deleted record.**
    - **The ActionDate should be the current timestamp.**

Testing:

1. **Delete an employee record from the Employee table.**
2. **Check the EmpLog table to verify that the deleted employee's information is stored correctly.**

```
mysql> CREATE TABLE EmpLog (
    ->     ActionType VARCHAR(50),
    ->     Empno INT,
    ->     Ename VARCHAR(50),
    ->     ActionDate DATETIME
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DELIMITER //
mysql> CREATE TRIGGER EMP_DELETED_DATA
    -> AFTER DELETE ON Employee
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO EmpLog (ActionType, Empno, Ename, ActionDate)
    ->     VALUES ('DELETE', OLD.Empno, OLD.Ename, NOW());
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> DELETE FROM Employee WHERE Empno = 7654; -- Delete an employee
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM EmpLog; -- Check the log table
+------------+-------+--------+---------------------+
| ActionType | Empno | Ename  | ActionDate          |
+------------+-------+--------+---------------------+
| DELETE     |  7654 | MARTIN | 2024-12-03 16:08:37 |
+------------+-------+--------+---------------------+
1 row in set (0.00 sec)
```

Problem 2: Track Employee Salary Updates

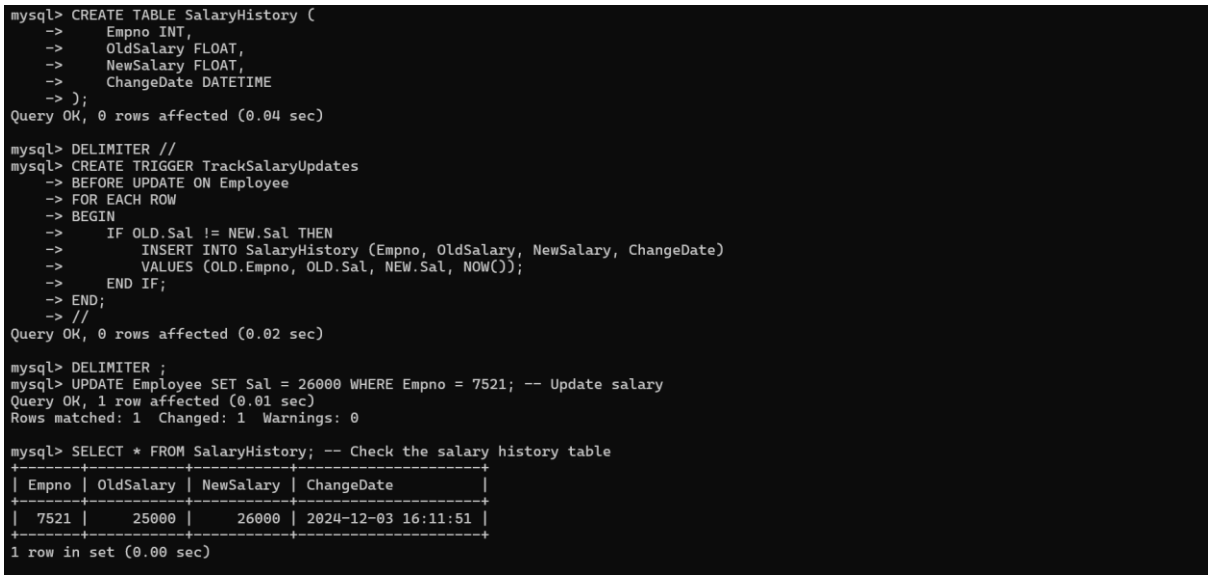
Objective: Create a trigger that logs any changes to an employee's salary.

Task:

1. Create a table named SalaryHistory with the following columns:
    - Empno (INT)
    - OldSalary (FLOAT)
    - NewSalary (FLOAT)
    - ChangeDate (DATETIME)
2. Create a trigger named TrackSalaryUpdates that will insert a record into SalaryHistory every time the Sal column in the Employee table is updated.

Testing:

1. Update the salary of an employee to see the changes tracked in SalaryHistory.
2. View the contents of the SalaryHistory table.



```
mysql> CREATE TABLE SalaryHistory (
    ->     Empno INT,
    ->     OldSalary FLOAT,
    ->     NewSalary FLOAT,
    ->     ChangeDate DATETIME
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER //
mysql> CREATE TRIGGER TrackSalaryUpdates
    -> BEFORE UPDATE ON Employee
    -> FOR EACH ROW
    -> BEGIN
    ->     IF OLD.Sal != NEW.Sal THEN
    ->         INSERT INTO SalaryHistory (Empno, OldSalary, NewSalary, ChangeDate)
    ->         VALUES (OLD.Empno, OLD.Sal, NEW.Sal, NOW());
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql> UPDATE Employee SET Sal = 26000 WHERE Empno = 7521; -- Update salary
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM SalaryHistory; -- Check the salary history table
+-------+-----------+-----------+---------------------+
| Empno | OldSalary | NewSalary | ChangeDate          |
+-------+-----------+-----------+---------------------+
|  7521 |     25000 |     26000 | 2024-12-03 16:11:51 |
+-------+-----------+-----------+---------------------+
1 row in set (0.00 sec)
```

Problem 3: Automatically Update Department Count

Objective: Keep track of the number of employees in each department automatically whenever employees are added or removed.

Task:

1. Create a table named DepartmentCount with the following columns:
    - Deptno (INT)
    - EmployeeCount (INT)
2. Write triggers to automatically update the EmployeeCount whenever an employee is inserted or deleted in the Employee table.

```
mysql> CREATE TABLE DepartmentCount (
    ->     Deptno INT PRIMARY KEY,
    ->     EmployeeCount INT
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> INSERT INTO DepartmentCount (Deptno, EmployeeCount)
    -> SELECT Deptno, COUNT(*)
    -> FROM Employee
    -> GROUP BY Deptno;
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> DELIMITER //
mysql> CREATE TRIGGER UpdateDeptCountAfterInsert
    -> AFTER INSERT ON Employee
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE DepartmentCount
    ->     SET EmployeeCount = EmployeeCount + 1
    ->     WHERE Deptno = NEW.Deptno;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TRIGGER UpdateDeptCountAfterDelete
    -> AFTER DELETE ON Employee
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE DepartmentCount
    ->     SET EmployeeCount = EmployeeCount - 1
    ->     WHERE Deptno = OLD.Deptno;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> DELIMITER ;
mysql>
mysql> INSERT INTO Employee VALUES (8000, 'TEST', 'CLERK', '2024-12-01', 15000, 30);
Query OK, 1 row affected (0.02 sec)

mysql> DELETE FROM Employee WHERE Empno = 8000;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM DepartmentCount;
+--------+---------------+
| Deptno | EmployeeCount |
+--------+---------------+
|     10 |             3 |
|     20 |             5 |
|     30 |             5 |
+--------+---------------+
3 rows in set (0.00 sec)
```