Assignment - 8

Database Management Systems (DBMS)

Name: Kamithkar Vinod
Course: PG DAC August 2025
PRN: 250850320040
Form No: 250500480
Date: 30-10-2025

Problem 1: Before_Insert

Task: Create a trigger to automatically store employee names in uppercase before inserting into the table.

Code: —

```
DELIMITER $$
CREATE TRIGGER before_emp_insert_upper
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    SET NEW.emp_name = UPPER(NEW.emp_name);
END$$
DELIMITER ;

INSERT INTO employees(emp_name, salary) VALUES ('john doe',
   40000);
SELECT * FROM employees;
```

Output: —

```
mysql> INSERT INTO employees(emp_name, salary) VALUES
('john doe', 40000);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employees;
+--------+----------+----------+
| emp_id | emp_name | salary   |
+--------+----------+----------+
|      1 | john doe | 40000.00 |
|      2 | JOHN DOE | 40000.00 |
+--------+----------+----------+
```

Problem 2: Before_Insert

Task: Create a trigger that ensures no employee salary is less than 10000 when inserting data.

Code: —

```sql
DELIMITER $$
CREATE TRIGGER before_emp_insert_min_salary
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
 IF NEW.salary < 10000 THEN
   SET NEW.salary = 10000;
 END IF;
END$$
DELIMITER ;

-- Example to try
INSERT INTO employees(emp_name, salary) VALUES ('mary', 5000);
SELECT * FROM employees;
```

Output: —

```
mysql> SELECT * FROM employees;
+--------+----------+----------+
| emp_id | emp_name | salary   |
+--------+----------+----------+
|      1 | john doe | 40000.00 |
|      2 | JOHN DOE | 40000.00 |
|      3 | DAVID    | 35000.00 |
|      4 | MARY     | 10000.00 |
+--------+----------+----------+
```

Problem 3: After_Insert

Task: Create a trigger that records every new employee addition in the audit_log table.

Code:

```
DELIMITER $$
CREATE TRIGGER after_emp_insert_log
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
 INSERT INTO audit_log(action, emp_id, new_salary)
 VALUES ('INSERT', NEW.emp_id, NEW.salary);
END$$
DELIMITER ;

-- Example to try
INSERT INTO employees(emp_name, salary) VALUES ('david', 35000);
SELECT * FROM audit_log;
```

Output:

```
mysql> SELECT * FROM audit_log;
+--------+--------+--------+------------+------------+---------------------+
| log_id | action | emp_id | old_salary | new_salary | log_time            |
+--------+--------+--------+------------+------------+---------------------+
|      1 | INSERT |      3 |       NULL |   35000.00 | 2025-10-31 12:31:17 |
+--------+--------+--------+------------+------------+---------------------+
```

Problem 4: After_Insert

Task: Create a trigger that displays a welcome message when a new employee is inserted.

Code:

```
-- Write your SQL code here
```

Output:

Problem 5: Before_Update

Task: Create a trigger that prevents salary reduction for any employee.

Code:

```
DELIMITER $$
CREATE TRIGGER before_emp_update_no_decrease
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
 IF NEW.salary < OLD.salary THEN
```

```
   SET NEW.salary = OLD.salary;
 END IF;
END$$
DELIMITER ;

-- Example to try
UPDATE employees SET salary = 10000 WHERE emp_id = 1;
SELECT * FROM employees;
```

Output: —

```
mysql> UPDATE employees SET salary = 10000 WHERE emp_id = 1;
Query OK, 0 rows affected (0.01 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> SELECT * FROM employees;
+--------+----------+----------+
| emp_id | emp_name | salary   |
+--------+----------+----------+
|      1 | john doe | 40000.00 |
|      2 | JOHN DOE | 40000.00 |
|      3 | DAVID    | 35000.00 |
|      4 | MARY     | 10000.00 |
|      5 | SOPHIA   | 28000.00 |
+--------+----------+----------+
```

Problem 6: Before_Update

Task: Create a trigger that rounds off the updated salary to the nearest integer.

Code: —

```
DELIMITER $$
CREATE TRIGGER before_emp_update_round
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
 SET NEW.salary = ROUND(NEW.salary, 0);
END$$
DELIMITER ;

-- Example to try
UPDATE employees SET salary = 45789.65 WHERE emp_id = 2;
SELECT * FROM employees;
```

Output: —

```
mysql> UPDATE employees SET salary = 45789.65 WHERE emp_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM employees;
+--------+----------+----------+
| emp_id | emp_name | salary   |
+--------+----------+----------+
|      1 | john doe | 40000.00 |
|      2 | JOHN DOE | 45790.00 |
|      3 | DAVID    | 35000.00 |
|      4 | MARY     | 10000.00 |
|      5 | SOPHIA   | 28000.00 |
+--------+----------+----------+
```

Problem 7: After_Update

Task: Create a trigger that logs old and new salaries whenever an employee's salary is updated.

Code: —

```
DELIMITER $$
CREATE TRIGGER after_emp_update_log
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
 INSERT INTO audit_log(action, emp_id, old_salary, new_salary)
 VALUES ('UPDATE', NEW.emp_id, OLD.salary, NEW.salary);
END$$
DELIMITER ;

-- Example to try
UPDATE employees SET salary = 48000 WHERE emp_id = 1;
SELECT * FROM audit_log;
```

Output: —

```
mysql> UPDATE employees SET salary = 48000 WHERE emp_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM audit_log;
+--------+--------+--------+------------+------------+---------------------+
| log_id | action | emp_id | old_salary | new_salary | log_time            |
+--------+--------+--------+------------+------------+---------------------+
|      1 | INSERT |      3 |       NULL |   35000.00 | 2025-10-31 12:31:17 |
|      2 | INSERT |      4 |       NULL |   10000.00 | 2025-10-31 12:31:54 |
|      3 | INSERT |      5 |       NULL |   28000.00 | 2025-10-31 12:34:27 |
|      4 | UPDATE |      1 |   40000.00 |   48000.00 | 2025-11-04 13:26:56 |
+--------+--------+--------+------------+------------+---------------------+
```

Problem 8: After_Update

Task: Create a trigger that displays a message whenever an employee's salary is increased.

Code: —

```
-- Write your SQL code here
```

Output: —

Problem 9: Before_Delete

Task: Create a trigger that prevents deletion of a record if the employee name is "CEO".

Code: —

```
DELIMITER $$
CREATE TRIGGER before_emp_delete_ceo
BEFORE DELETE ON employees
FOR EACH ROW
BEGIN
 IF OLD.emp_name = 'CEO' THEN
   SIGNAL SQLSTATE '45000'
   SET MESSAGE_TEXT = 'Cannot delete CEO record!';
 END IF;
END$$
DELIMITER ;

-- Example to try
INSERT INTO employees(emp_name, salary) VALUES ('CEO', 90000);
DELETE FROM employees WHERE emp_name = 'CEO'; -- should raise
   error
```

Output: —

```
mysql> INSERT INTO employees(emp_name, salary) VALUES ('CEO', 90000);
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM employees WHERE emp_name = 'CEO';
ERROR 1644 (45000): Cannot delete CEO record!
```

Problem 10: Before_Delete

Task: Create a trigger that logs any attempt to delete an employee record in the audit_log table.

Code: —

```
DELIMITER $$
CREATE TRIGGER before_emp_delete_log
BEFORE DELETE ON employees
FOR EACH ROW
BEGIN
 INSERT INTO audit_log(action, emp_id, old_salary)
```

```
 VALUES ('DELETE_ATTEMPT', OLD.emp_id, OLD.salary);
END$$
DELIMITER ;

-- Example to try
DELETE FROM employees WHERE emp_id = 2;
SELECT * FROM audit_log;
```

Output: —

```
mysql> DELETE FROM employees WHERE emp_id = 2;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM audit_log;
+--------+----------------+--------+------------+------------+---------------------+
| log_id | action         | emp_id | old_salary | new_salary | log_time            |
+--------+----------------+--------+------------+------------+---------------------+
|      1 | INSERT         |      3 |       NULL |   35000.00 | 2025-10-31 12:31:17 |
|      2 | INSERT         |      4 |       NULL |   10000.00 | 2025-10-31 12:31:54 |
|      3 | INSERT         |      5 |       NULL |   28000.00 | 2025-10-31 12:34:27 |
|      4 | UPDATE         |      1 |   40000.00 |   48000.00 | 2025-11-04 13:26:56 |
|      5 | INSERT         |      6 |       NULL |   90000.00 | 2025-11-04 13:28:51 |
|      6 | DELETE_ATTEMPT |      2 |   45790.00 |       NULL | 2025-11-04 11:39:17 |
+--------+----------------+--------+------------+------------+---------------------+
```

Problem 11: After_Delete

Task: Create a trigger that copies deleted employee records into deleted_employees.

Code: —

```
DELIMITER $$
CREATE TRIGGER after_emp_delete_copy
AFTER DELETE ON employees
FOR EACH ROW
BEGIN
 INSERT INTO deleted_employees(emp_id, emp_name)
 VALUES (OLD.emp_id, OLD.emp_name);
END$$
DELIMITER ;

-- Example to try
DELETE FROM employees WHERE emp_id = 3;
SELECT * FROM deleted_employees;
```

Output: —

```
mysql> DELETE FROM employees WHERE emp_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM deleted_employees;
+--------+----------+---------------------+
| emp_id | emp_name | deleted_on          |
+--------+----------+---------------------+
|      3 | DAVID    | 2025-11-04 11:40:15 |
+--------+----------+---------------------+
```

Problem 12: After_Delete

Task: Create a trigger that logs every deletion in the audit_log table.

Code: —

```
DELIMITER $$
CREATE TRIGGER after_emp_delete_log
AFTER DELETE ON employees
FOR EACH ROW
BEGIN
 INSERT INTO audit_log(action, emp_id, old_salary)
 VALUES ('DELETE', OLD.emp_id, OLD.salary);
END$$
DELIMITER ;

-- Example to try
DELETE FROM employees WHERE emp_id = 4;
SELECT * FROM audit_log;
```

Output: —

```
mysql> DELETE FROM employees WHERE emp_id = 4;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM audit_log;
+--------+---------------+--------+------------+------------+---------------------+
| log_id | action        | emp_id | old_salary | new_salary | log_time            |
+--------+---------------+--------+------------+------------+---------------------+
|      1 | INSERT        |      3 |       NULL |   35000.00 | 2025-10-31 12:31:17 |
|      2 | INSERT        |      4 |       NULL |   10000.00 | 2025-10-31 12:31:54 |
|      3 | INSERT        |      5 |       NULL |   28000.00 | 2025-10-31 12:34:27 |
|      4 | UPDATE        |      1 |   40000.00 |   48000.00 | 2025-11-04 13:26:56 |
|      5 | INSERT        |      6 |       NULL |   90000.00 | 2025-11-04 13:28:51 |
|      6 | DELETE_ATTEMPT |      2 |   45790.00 |       NULL | 2025-11-04 11:39:17 |
|      7 | DELETE_ATTEMPT |      3 |   35000.00 |       NULL | 2025-11-04 11:40:15 |
|      8 | DELETE_ATTEMPT |      4 |   10000.00 |       NULL | 2025-11-04 11:41:37 |
|      9 | DELETE        |      4 |   10000.00 |       NULL | 2025-11-04 11:41:37 |
+--------+---------------+--------+------------+------------+---------------------+
```