EXPERIMENT : 6B TRIGGERS

A **trigger** is a special stored program that is automatically executed in response to a specific event on a table.

Triggers are used to:

- Maintain data consistency, Enforce complex business rules
- Automatically update logs or audit tables, Used to maintain integrity of the data.

SYNTAX

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
   -- trigger body
END;
DROP TRIGGER trigger_name;
SHOW TRIGGERS;
```

Types of Triggers

Trigger Type	Description
BEFORE INSERT	Executes before a new record is inserted.
AFTER INSERT	Executes after a record is inserted.
BEFORE UPDATE	Executes before an existing record is updated.
AFTER UPDATE	Executes after a record is updated.
BEFORE DELETE	Executes before a record is deleted.
AFTER DELETE	Executes after a record is deleted.

Create Main Table

```
CREATE TABLE student (
   stud_id INT PRIMARY KEY,
   stud_name VARCHAR(50),
   dept VARCHAR(10),
   marks INT
);
```

Create Log Table

```
CREATE TABLE student_log (
   log_id INT AUTO_INCREMENT PRIMARY KEY,
   stud_id INT,
   action VARCHAR(50),
   log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Task 1 – AFTER INSERT Trigger

```
DELIMITER //
CREATE TRIGGER trg_after_insert_student
AFTER INSERT ON student
FOR EACH ROW
BEGIN
   INSERT INTO student_log (stud_id, action)
   VALUES (NEW.stud_id, 'Record Inserted');
END; //
```

```
DELIMITER ;

INSERT INTO student VALUES (101, 'Arun', 'CSE', 85);
SELECT * FROM student_log;
```

Task 2 – BEFORE INSERT Trigger

```
DELIMITER //
CREATE TRIGGER trg_before_insert_student
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    SET NEW.dept = UPPER(NEW.dept);
END; //
DELIMITER ;

INSERT INTO student VALUES (102, 'Divya', 'ece', 90);
SELECT * FROM student;
```

Task 3 – BEFORE UPDATE Trigger

```
DELIMITER //
CREATE TRIGGER trg_before_update_marks
BEFORE UPDATE ON student
FOR EACH ROW
BEGIN
    IF NEW.marks > 100 THEN
```

```
        SET NEW.marks = 100;
    END IF;
END; //
DELIMITER ;

UPDATE student SET marks = 120 WHERE stud_id = 101;
SELECT * FROM student;
```

Task 4 – AFTER UPDATE Trigger

```
DELIMITER //
CREATE TRIGGER trg_after_update_student
AFTER UPDATE ON student
FOR EACH ROW
BEGIN
    INSERT INTO student_log (stud_id, action)
    VALUES (NEW.stud_id, 'Marks Updated');
END; //
DELIMITER ;

UPDATE student SET marks = 95 WHERE stud_id = 102;
SELECT * FROM student_log;
```

Task 5 – BEFORE DELETE Trigger

```
DELIMITER //
CREATE TRIGGER trg_before_delete_student
BEFORE DELETE ON student
```

```
FOR EACH ROW
BEGIN
   INSERT INTO student_log (stud_id, action)
   VALUES (OLD.stud_id, 'Record to be deleted');
END; //
DELIMITER ;

DELETE FROM student WHERE stud_id = 103;
SELECT * FROM student_log;
```

Task 6 – AFTER DELETE Trigger

```
DELIMITER //
CREATE TRIGGER trg_after_delete_student
AFTER DELETE ON student
FOR EACH ROW
BEGIN
   INSERT INTO student_log (stud_id, action)
   VALUES (OLD.stud_id, 'Record deleted');
END; //
DELIMITER ;

DELETE FROM student WHERE stud_id = 102;
SELECT * FROM student_log;
```

Task 7 – Trigger for Auto Attendance Update

```
CREATE TABLE attendance (
    stud_id INT,
    status VARCHAR(10)
);
DELIMITER //
CREATE TRIGGER trg_after_insert_attendance
AFTER INSERT ON student
FOR EACH ROW
BEGIN
    INSERT INTO attendance (stud_id, status)
    VALUES (NEW.stud_id, 'Present');
END; //
DELIMITER ;

INSERT INTO student VALUES (104, 'Ravi', 'IT', 80);
SELECT * FROM attendance
```

Task 8 – Trigger to Prevent Low Marks

```
DELIMITER //
CREATE TRIGGER trg_before_insert_check_marks
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    IF NEW.marks < 35 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Marks below passing limit!';
```

END IF;

END; //

DELIMITER ;

INSERT INTO student VALUES (105, 'Kiran', 'EEE', 20);

SPOT QUESTIONS:

1. Whenever a new student is added to the student table, an entry should be automatically added to another table student_activity with the message "New record added".

Hint: create TABLE student_activity

Test the trigger

INSERT INTO student VALUES (201, 'Anita', 'CSE', 88);

SELECT * FROM student_activity;

2. Create a trigger so that before inserting data into the student table, the dept value is automatically stored in uppercase (e.g., "cse" → "CSE").

Test the trigger

INSERT INTO student VALUES (202, 'Bala', 'ece', 92);

3. When a student record is deleted from student, the deleted data should automatically move to a table named student_backup.

Hint: CREATE TABLE student_backup with column names stud_id, stud_name,dept, marks

DELETE FROM student WHERE stud_id = 201;

SELECT * FROM student_backup;

4. Each time a new student record is added, a new record should also be added automatically in an attendance table with status “Present”.

Hint : CREATE TABLE attendance with columns stud_id , status .

Test the trigger:

INSERT INTO student VALUES (205, 'Farhan', 'CSE', 78);

SELECT * FROM attendance;