# Week 6 - LAQ's

## Instructions

## Make a case study for trigger on update and insert operation.

Case Study: Triggers on Update and Insert Operations

Background

In a retail management system, we have two tables: orders and order_history. The orders table stores current order details, while the order_history table maintains a log of all changes made to orders for auditing and tracking purposes. We want to implement triggers to automatically update the order_history table whenever an order is inserted or updated in the orders table.

Objectives

1. Automatically log new orders into the order_history table when they are created.
2. Track changes to existing orders by logging updates in the order_history table.

Implementation

1. Creating the Tables

First, we create the necessary tables:

```
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   customer_name VARCHAR(100),
   order_date DATETIME,
   status VARCHAR(50)
);

CREATE TABLE order_history (
   history_id INT PRIMARY KEY AUTO_INCREMENT,
   order_id INT,
   status VARCHAR(50),
   change_timestamp DATETIME,
    operation VARCHAR(10)
);
```

2. Creating the Insert Trigger

We create a trigger that logs information into the order_history table after a new order is inserted into the orders table.

```
CREATE TRIGGER trg_after_insert_order
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
    INSERT INTO order_history (order_id, status, change_timestamp, operation)
    VALUES (NEW.order_id, NEW.status, NOW(), 'INSERT');
END;
```

3. Creating the Update Trigger

Next, we create a trigger that logs changes into the order_history table whenever an existing order is updated.

```
CREATE TRIGGER trg_after_update_order
AFTER UPDATE ON orders
FOR EACH ROW
BEGIN
    INSERT INTO order_history (order_id, status, change_timestamp, operation)
    VALUES (NEW.order_id, NEW.status, NOW(), 'UPDATE');
END;
```

Testing the Triggers

1. Inserting Data

When a new order is inserted:

```
INSERT INTO orders (order_id, customer_name, order_date, status)
VALUES (1, 'John Doe', NOW(), 'Pending');
```

This action will automatically create a corresponding entry in the order_history table:

| history_id | order_id | status | change_timestamp | operation |
|------------|----------|---------|----------------------|-----------|
| 1 | 1 | Pending | 2024-11-11 12:00:00 | INSERT |

2. Updating Data

When an existing order is updated:

```
UPDATE orders
SET status = 'Shipped'
WHERE order_id = 1;
```

This action will also create an entry in the order_history table:

| history_id | order_id | status | change_timestamp | operation |
|------------|----------|---------|----------------------|-----------|
| 2 | 1 | Shipped | 2024-11-11 12:05:00 | UPDATE |