Embedded SQL Queries and Triggers

Karanjeet Singh (2022235)   Lakshay Trehan(2022267)

March 31, 2024

1 Trigger 1: Reduce MSP Trigger

This trigger is designed to automatically adjust the Manufacturer's Suggested Price (MSP) of a product based on customer feedback. Specifically, it monitors the insertion of new feedback into the database. If a product receives one or more ratings and the average rating is below 2, the MSP of that product is reduced by 10%.

```
DELIMITER //
CREATE TRIGGER reduce_msp_trigger AFTER INSERT ON Feedback
FOR EACH ROW
BEGIN
    DECLARE avg_rating DECIMAL(3, 2);
    DECLARE rating_count INT;

    -- Calculate average rating for the product
    SELECT AVG(Rating), COUNT(*) INTO avg_rating, rating_count
    FROM Feedback
    WHERE ProductID = NEW.ProductID;

    -- Check if the product has been rated 5 times or more and
        average rating is below 2
    IF rating_count >= 1 AND avg_rating < 2 THEN
        -- Reduce MSP by 10%
        UPDATE Product
        SET MSP = MSP * 0.9
        WHERE ProductID = NEW.ProductID;
    END IF;
END //
DELIMITER ;
```

2 Trigger 2: Block After Three Attempts

This trigger monitors login attempts and blocks the user if they have attempted to log in more than three times.

```
DELIMITER //
CREATE TRIGGER BlockAfterThreeAttempts
AFTER INSERT ON LoginAttempts
FOR EACH ROW
```

```
BEGIN
    IF NEW.Attempts >= 3 THEN
        -- Update the user table to set a temporary block
        UPDATE User SET BlockedUntil = NOW() + INTERVAL 1 MINUTE
            WHERE Username = NEW.Username;
    END IF;
END //
DELIMITER ;
```

3 Trigger 3: Update Inventory After Sale

This trigger updates the inventory after a sale has been made.

```
DELIMITER //
CREATE TRIGGER update_inventory_after_sale
AFTER INSERT
ON sales
FOR EACH ROW
BEGIN
    UPDATE inventory
    SET Quantity = Quantity - NEW.QuantitySold
    WHERE ProductID = NEW.ProductID;
END //
DELIMITER ;
```

4 Embedded SQL Queries

1. Signup Route

- **Route:** `/signup`
- **Purpose:** Handles user signup by inserting user data into the User table.
- **SQL Query:**

```
INSERT INTO User (Username, Password, Role, Permissions,
    PersonalInformation)
VALUES (?, ?, ?, ?, ?);
```

2. Login Route

- **Route:** `/login`
- **Purpose:** Handles user login authentication and session management.
- **SQL Query:**

```
SELECT * FROM User WHERE Username = ? AND Password = ?;
```

3. Fetch Cart Items Route

- **Route:** `/cart/:userId`
- **Purpose:** Retrieves items from the shoppingcart table based on the user's ID.
- **SQL Query:**

  ```
  SELECT * FROM shoppingcart WHERE userId = ?;
  ```

4. Delete Cart Item Route

- **Route:** `/delete/:productId`
- **Purpose:** Deletes an item from the shoppingcart table based on the product ID.
- **SQL Query:**

  ```
  DELETE FROM shoppingcart WHERE ProductID = ?;
  ```

5. Checkout Route

- **Route:** `/deleteS/:userId`
- **Purpose:** Handles the checkout process by calculating total price, inserting order data into the orders table, and deleting items from the shopping cart.
- **SQL Queries:**
  - Calculate Total Price:

    ```
    SELECT SUM(TotalPrice) AS totalPrice FROM shoppingcart
        ;
    ```

  - Insert Order Data:

    ```
    INSERT INTO orders (OrderID, UserID, TotalAmount,
        OrderDate) VALUES (?, ?, ?, ?);
    ```

  - Insert Sales Data (for each item in the cart):

    ```
    INSERT INTO sales (ProductID, QuantitySold, SaleDate)
        VALUES (?, ?, ?);
    ```

  - Delete Items from Shopping Cart:

    ```
    DELETE FROM shoppingcart;
    ```

6. Fetch Total Price Route

- **Route:** `/getTotalPrice/:userID`
- **Purpose:** Calculates the total price of items in the shopping cart for a specific user.
- **SQL Query:**

  ```
  SELECT SUM(TotalPrice) AS totalPrice FROM shoppingcart
      WHERE UserID = ?;
  ```

7. **Add To Cart Route**

   - **Route:** `/addToCart`
   - **Purpose:** Adds a product to the shopping cart in the shoppingcart table.
   - **SQL Query:**

   ```
   INSERT INTO shoppingcart (CartID, UserID, ProductID,
       Quantity, TotalPrice, Status) VALUES (?, ?, ?, ?, ?,
       ?);
   ```

8. **Feedback Route**

   - **Route:** `/feedback`
   - **Purpose:** Inserts feedback data into the Feedback table.
   - **SQL Query:**

   ```
   INSERT INTO Feedback (UserID, ProductID, Rating, Comment)
       VALUES (?, ?, ?, ?);
   ```

9. **Fetch Inventory Data Route**

   - **Route:** `/inventory`
   - **Purpose:** Fetches all data from the inventory table.
   - **SQL Query:**

   ```
   SELECT * FROM inventory;
   ```

10. **Insert Inventory Data Route**

    - **Route:** `/inventorys`
    - **Purpose:** Inserts inventory data into the inventory table.
    - **SQL Query:**

    ```
    INSERT INTO inventory (ProductID, Quantity,
        LowInventoryAlert) VALUES (?, ?, ?);
    ```

11. **Update Inventory Data Route**

    - **Route:** `/inventory/:id`
    - **Purpose:** Updates inventory data in the inventory table.
    - **SQL Query:**

    ```
    UPDATE inventory SET Quantity = ?, LowInventoryAlert = ?
        WHERE ProductID = ?;
    ```

12. **Delete Inventory Data Route**

    - **Route:** `/inventory/:id`

- **Purpose:** Deletes inventory data from the inventory table.
- **SQL Query:**

```sql
DELETE FROM inventory WHERE ProductID = ?;
```

13. Fetch Product Data Route

- **Route:** `/products`
- **Purpose:** Fetches all data from the product table.
- **SQL Query:**

```sql
SELECT * FROM product;
```

14. Fetch Product Data for Specific User in Cart Route

- **Route:** `/productsS/:userID`
- **Purpose:** Fetches product data from the SHOPPINGCART table for a specific user.
- **SQL Query:**

```sql
SELECT * FROM SHOPPINGCART WHERE userID = ?;
```

15. Cleanup on Process Exit

- **Purpose:** Deletes shopping cart items from the SHOPPINGCART table where the UserID is not equal to 0.
- **SQL Query:**

```sql
DELETE FROM SHOPPINGCART WHERE UserID != 0;
```