EIU

Order Processing System (MySQL Workbench)

Problem Statement - You are required to design and implement an Order Processing System using MySQL Workbench. The system must ensure data consistency, performance optimization, and error handling while processing customer orders.

Database Requirements -

Create the following tables with appropriate data types, primary keys, and foreign keys:

1. customers
    - customer_id
    - customer_name
    - email
2. products
    - product_id
    - product_name
    - price
    - stock_quantity
3. orders
    - order_id
    - customer_id
    - order_date
    - total_amount
4. order_items
    - order_item_id
    - order_id
    - product_id
    - quantity
    - item_price
5. order_audit
    - audit_id
    - order_id
    - action
    - action_date
6. error_log
    - error_id
    - error_message
    - error_time

Task 1: Query Processing & Optimization

1. Write a query to retrieve:
    - All orders placed by a specific customer
    - Include customer name and total order amount
2. Analyze the query using EXPLAIN.

3. Create suitable indexes to optimize the query and compare execution plans before and after indexing.

Task 2: User-Defined Function

Create a function named calculate_order_total that:

- Accepts an order_id
- Returns the total amount of the order based on order items

Task 3: Trigger Implementation

1. Create a BEFORE INSERT trigger on order_items that:
    - Prevents inserting an order item if sufficient stock is not available
2. Create an AFTER INSERT trigger on orders that:
    - Logs the order creation into the order_audit table

Task 4: Stored Procedure with Transaction & Cursor

Create a stored procedure named process_order that performs the following:

Inputs:

- customer_id
- list of product IDs and quantities (The temporary table is already populated before calling the procedure)

Procedure Requirements:

1. Start a transaction
2. Insert a new record into the orders table
3. Use a cursor to:
    - Iterate through each product in the order
    - Insert records into order_items
    - Update stock in the products table
4. Calculate the total order amount using the function created earlier
5. Update the total amount in the orders table
6. Commit the transaction if all operations succeed

Task 5: Error Handling

1. Implement exception handling using:
    - DECLARE HANDLER
    - SIGNAL or RESIGNAL
2. If any error occurs:
    - Roll back the transaction

- Insert error details into the error_log table
- Display a meaningful error message

Task 6: Transaction Control

1. Demonstrate the use of:
   - START TRANSACTION
   - COMMIT
   - ROLLBACK
2. Explain how ACID properties are maintained in your solution.

Task 7: Testing & Validation

1. Insert sample data into all tables
2. Execute the stored procedure for:
   - A valid order
   - An order with insufficient stock
3. Observe:
   - Trigger execution
   - Transaction rollback
   - Error logging

CRUD Operations Using Embedded SQL (Java + MySQL Workbench)

Problem Statement: Design and implement a Java application that performs CRUD (Create, Read, Update, Delete) operations on a MySQL database using Embedded SQL (JDBC). The application must interact with the database created in MySQL Workbench and ensure proper exception handling, resource management, and security.

Create a database named student_db with the following table:

students

- student_id (Primary Key)
- student_name
- email
- course
- marks

Task 1: Database Setup

1. Create the database and table using MySQL Workbench.
2. Insert at least **5 sample records** into the table.

Task 2: Java Application Requirements

EIU

Develop a menu-driven Java application using JDBC that allows a user to perform the following operations:

1 CREATE (Insert Record)

- Accept student details from the user
- Insert a new record into the students table
- Use PreparedStatement
- Display a success or failure message

2 READ (Retrieve Records)

- Retrieve and display:
  - All student records
  - Student details by student_id
- Use ResultSet to process query results

3 UPDATE (Modify Record)

- Update:
  - Student marks
  - Student course
- Update should be based on student_id
- Display the number of affected rows

4 DELETE (Remove Record)

- Delete a student record using student_id
- Confirm deletion before execution

Task 3: Embedded SQL & Exception Handling

1. Use embedded SQL statements inside Java using JDBC
2. Handle exceptions using:
   - try-catch-finally
   - SQLException
3. Display meaningful error messages

Task 4: Transaction Management

1. Perform multiple updates inside a transaction
2. Use:
   - setAutoCommit(false)
   - commit()
   - rollback() if an error occurs
3. Demonstrate atomicity

EIU

Task 5: Security & Best Practices

1. Prevent SQL Injection using PreparedStatement
2. Close all resources properly:
    - Connection
    - PreparedStatement
    - ResultSet
3. Follow Java naming conventions

Task 6: Testing & Validation

1. Test all CRUD operations
2. Test invalid inputs (e.g., non-existing student ID)
3. Capture and display database errors

Output of the above task -

- Menu displayed on console
- Successful execution of all CRUD operations
- Proper error handling
- Database updated correctly