```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.*;
import java.util.*;
import java.util.ArrayList;

class DBConnection {
   public static Connection getConnection() {
       try {
           String url = "jdbc:mysql://localhost:3306/labib";
           String user = "root";
           String password = "labib557"; // set your MySQL password here
           Class.forName("com.mysql.cj.jdbc.Driver");
           return DriverManager.getConnection(url, user, password);
       } catch (Exception e) {
           e.printStackTrace();
           return null;
       }
   }
}
class Product {
   private int productId;
   private String name;
   private double price;
   private int quantity;


   public Product(int productId, String name, double price, int quantity) {
       this.productId = productId;
       this.name = name;
       this.price = price;
       this.quantity = quantity;
   }

   public int getProductId() { return productId; }
   public String getName() { return name; }
   public double getPrice() { return price; }
   public int getQuantity() { return quantity; }

   public void setQuantity(int quantity) {
       this.quantity = quantity;
   }


   public void display() {
       System.out.println( "ID: " + productId + ", Name: " + name + ", Price:
$" + price + ", Quantity: " + quantity);
   }
}

class ProductDAO {

   public void addProduct(Product p) {
       try (Connection con = DBConnection.getConnection()) {
           String query = "INSERT INTO products (name, price, quantity)
VALUES (?, ?, ?)";
           PreparedStatement stmt = con.prepareStatement(query);
           stmt.setString(1, p.getName());
           stmt.setDouble(2, p.getPrice());
           stmt.setInt(3, p.getQuantity());
           stmt.executeUpdate();
           System.out.println("✅ Product Added!");
       } catch (Exception e) {
           e.printStackTrace();
       }
   }

   public List<Product> getAllProducts() {
       List<Product> list = new ArrayList<>();
       try (Connection con = DBConnection.getConnection()) {
           String query = "SELECT * FROM products";
           Statement stmt = con.createStatement();
           ResultSet rs = stmt.executeQuery(query);
           while (rs.next()) {
               Product p = new Product(
                       rs.getInt("product_id"),
                       rs.getString("name"),
                       rs.getDouble("price"),
                       rs.getInt("quantity")
               );
               list.add(p);
           }
       } catch (Exception e) {
           e.printStackTrace();
       }
       return list;
   }

   public void deleteProduct(int productId) {
       try (Connection con = DBConnection.getConnection()) {
           String query = "DELETE FROM products WHERE product_id = ?";
           PreparedStatement stmt = con.prepareStatement(query);
           stmt.setInt(1, productId);
           int result = stmt.executeUpdate();
           if (result > 0) {
               System.out.println(" Product Deleted!");
           } else {
               System.out.println(" Product not found.");
           }
       } catch (Exception e) {
           e.printStackTrace();
       }
   }
}
```

```java
class Customer {
   int customerId;
   String name;
   String email;

   // Constructor
   public Customer(int id, String name, String email) {
       this.customerId = id;
       this.name = name;
       this.email = email;
   }
   public int getCustomerId() {
       return customerId; }
   public String getName() {
       return name; }
   public String getEmail() {
       return email; }

}


class CustomerDAO {
   public void addCustomer(Customer c) {
       try (Connection con = DBConnection.getConnection()) {
           String sql = "INSERT INTO customers (name, email) VALUES (?, ?)";
           PreparedStatement stmt = con.prepareStatement(sql);
           stmt.setString(1, c.getName());
           stmt.setString(2, c.getEmail());
           stmt.executeUpdate();
           System.out.println("✅ Customer added.");
       } catch (Exception e) {
           e.printStackTrace();
       }
   }
   public void deleteCustomer(int customerId) {
       try (Connection con = DBConnection.getConnection()) {
           String sql = "DELETE FROM customers WHERE customer_id = ?";
           PreparedStatement stmt = con.prepareStatement(sql);
           stmt.setInt(1, customerId);
           int affected = stmt.executeUpdate();
           if (affected > 0) {
               System.out.println("🗑️ Customer deleted.");
           } else {
               System.out.println("❌ Customer ID not found.");
           }
       } catch (Exception e) {
           e.printStackTrace();
       }
   }

   public List<Customer> getAllCustomers() {
       List<Customer> list = new ArrayList<>();
       try (Connection con = DBConnection.getConnection()) {
           Statement stmt = con.createStatement();
           ResultSet rs = stmt.executeQuery("SELECT * FROM customers");
           while (rs.next()){
               Customer c = new Customer(
                       rs.getInt("customer_id"),
                       rs.getString("name"),
                       rs.getString("email")
               );
               list.add(c);
           }
       } catch (Exception e) {
           e.printStackTrace();
       }
       return list;
   }
}
class OrderResult {
   private int orderId;
   private long totalPrice;

   public OrderResult(int orderId, long totalPrice) {
       this.orderId = orderId;
       this.totalPrice = totalPrice;
   }

   public int getOrderId() {
       return orderId;
   }

   public long getTotalPrice() {
       return totalPrice;
   }
}
class OrderDAO {
   public OrderResult placeOrder(int customerId, int productId, int quantity)
{
       long totalPrice = 0;
       int orderId = -1;

       try (Connection con = DBConnection.getConnection()) {
           con.setAutoCommit(false);

           // 1. Get product price and stock
           String getProduct = "SELECT price, quantity FROM products WHERE
product_id = ?";
           PreparedStatement ps = con.prepareStatement(getProduct);
           ps.setInt(1, productId);
           ResultSet rs = ps.executeQuery();

           if (!rs.next()) {
               System.out.println("❌ Product not found.");
               return null;
           }
           double price = rs.getDouble("price");
           int available = rs.getInt("quantity");

           if (available < quantity) {
               System.out.println("❌ Not enough stock.");
               return null;
           }

           totalPrice = Math.round(price * quantity);  // ✅ Correct
totalPrice

           // 2. Insert order (only customer_id, let DB generate order_id)
           String insertOrder = "INSERT INTO orders (customer_id) VALUES
(?)";
           PreparedStatement orderStmt = con.prepareStatement(insertOrder,
Statement.RETURN_GENERATED_KEYS);
           orderStmt.setInt(1, customerId);
           orderStmt.executeUpdate();

           ResultSet orderKeys = orderStmt.getGeneratedKeys();
           if (orderKeys.next()) {
               orderId = orderKeys.getInt(1);  // ✅ Get auto-generated
orderId
           }

           // 3. Insert order detail (fixed comma!)
           String insertDetail = "INSERT INTO order_details (order_id,
customer_id, product_id, quantity, total_price) VALUES (?, ?, ?, ?, ?)";
           PreparedStatement detailStmt = con.prepareStatement(insertDetail);
           detailStmt.setInt(1, orderId);
           detailStmt.setInt(2, customerId);
           detailStmt.setInt(3, productId);
           detailStmt.setInt(4, quantity);
           detailStmt.setLong(5, totalPrice);
           detailStmt.executeUpdate();

           // 4. Update product stock
           String updateStock = "UPDATE products SET quantity = ? WHERE
product_id = ?";
           PreparedStatement updateStockStmt =
con.prepareStatement(updateStock);
           updateStockStmt.setInt(1, available - quantity);
           updateStockStmt.setInt(2, productId);
           updateStockStmt.executeUpdate();

           con.commit();
           System.out.println("✅ Order placed successfully!");

       } catch (Exception e) {
           e.printStackTrace();
       }
```

```java
        return new OrderResult(orderId, totalPrice);
    }
}


public class Main {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       ProductDAO dao = new ProductDAO();
       CustomerDAO customerDAO = new CustomerDAO();
       OrderDAO orderDAO = new OrderDAO();
       List<Product> productList = new ArrayList<>();


       while (true) {
           System.out.println("\n===== STORE MANAGEMENT SYSTEM =====");
           System.out.println("1. Add Product");
           System.out.println("2. View All Products");
           System.out.println("3. Delete Product");
           System.out.println("4. Add Customer");
           System.out.println("5. View Customers");
           System.out.println("6. Place Order");

           System.out.println("0. Exit");
           System.out.print("Enter choice: ");
           int choice = sc.nextInt();

           switch (choice) {
               case 1:
                   sc.nextLine(); // consume newline
                   System.out.print("ProductId: ");
                   int ProductId = sc.nextInt();

                   sc.nextLine(); // consume newline
                   System.out.print("Product Name: ");
                   String name = sc.nextLine();
                   System.out.print("Price: ");
                   double price = sc.nextDouble();
                   System.out.print("Quantity: ");
                   int quantity = sc.nextInt();
                   Product p = new Product(ProductId,name, price, quantity);
                   dao.addProduct(p);
                   break;

               case 2:
                   List<Product> products = dao.getAllProducts();
                   System.out.println("\nID\tName\t\tPrice\tQuantity");
                   for (Product prod : products) {
                       System.out.printf("%d\t%-10s\t%.2f\t%d\n",
                               prod.getProductId(),
                               prod.getName(),
                               prod.getPrice(),
                               prod.getQuantity()
```

```java
                            );
                        }
                        break;


                case 3:
                        System.out.print("Enter Product ID to delete: ");
                        int pid = sc.nextInt();
                        dao.deleteProduct(pid);
                        break;
                case 4:
                        System.out.print("Customer ID: ");
                        int cid = sc.nextInt();
                        sc.nextLine();
                        System.out.print("Customer Name: ");
                        String cname = sc.nextLine();
                        System.out.print("Email: ");
                        String email = sc.nextLine();
                        customerDAO.addCustomer(new Customer(cid,cname, email));
                        break;

                case 5:
                        List<Customer> customers = customerDAO.getAllCustomers();
                        System.out.println("\nID\tName\t\tEmail");
                        for (Customer cust : customers) {
                            System.out.printf("%d\t%-10s\t%s\n",
cust.getCustomerId(), cust.getName(), cust.getEmail());
                        }
                        break;

                case 6:
                        System.out.print("Enter Customer ID to delete: ");
                        int deleteId = sc.nextInt();
                        customerDAO.deleteCustomer(deleteId);
                        break;

                case 7:
                        System.out.print("Customer ID: ");
                        int custId = sc.nextInt();
                        System.out.print("Product ID: ");
                        int prodId = sc.nextInt();
                        System.out.print("Quantity: ");
                        int qty = sc.nextInt();
                        OrderResult result = orderDAO.placeOrder( custId,prodId,
qty);
                        if (result != null && result.getOrderId() != -1) {
                            System.out.printf(" Order ID: %d\n",
result.getOrderId());
                            System.out.printf(" Total Price: %d\n",
result.getTotalPrice());
                        }
                        break;
                case 0:
                        System.out.println("Exiting... Thank you!");
                        sc.close();
                        System.exit(0);
                        break;

                default:
                        System.out.println(" Invalid choice. Try again.");
                }
            }
        }
    }
```