Core Java-Asesessment9

### ****1. E-Commerce Order Management System (JDBC, SQL Queries, ResultSet Handling)****

import java.sql.\*;

public class ECommerceOrderManagement {

public static void main(String[] args) {

// JDBC URL, username, and password

String url = "jdbc:mysql://localhost:3306/ecommerce\_db";

String username = "root";

String password = "password";

// SQL query to fetch orders

String query = "SELECT order\_id, product\_name, quantity FROM orders";

try (Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query)) {

// Iterate over the result set and print order details

while (resultSet.next()) {

int orderId = resultSet.getInt("order\_id");

String productName = resultSet.getString("product\_name");

int quantity = resultSet.getInt("quantity");

System.out.println("Order ID: " + orderId + ", Product: " + productName + ", Quantity: " + quantity);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

### ****2. Secure Employee Database Access (PreparedStatement, Parameterized Queries)****

import java.sql.\*;

public class SecureEmployeeDatabaseAccess {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/employee\_db";

String username = "root";

String password = "password";

String query = "SELECT \* FROM employees WHERE department = ?";

try (Connection connection = DriverManager.getConnection(url, username, password);

PreparedStatement statement = connection.prepareStatement(query)) {

// Set the department parameter dynamically

statement.setString(1, "Sales");

// Execute the query and get the result

try (ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

int empId = resultSet.getInt("employee\_id");

String empName = resultSet.getString("employee\_name");

System.out.println("Employee ID: " + empId + ", Name: " + empName);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

### ****3. Batch Processing for Bulk Data Insertion (JDBC, PreparedStatement, Batch Updates)****

import java.sql.\*;

public class BatchProcessingExample {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/ecommerce\_db";

String username = "root";

String password = "password";

String query = "INSERT INTO orders (order\_id, product\_name, quantity) VALUES (?, ?, ?)";

try (Connection connection = DriverManager.getConnection(url, username, password);

PreparedStatement statement = connection.prepareStatement(query)) {

// Disable auto-commit to manage transactions manually

connection.setAutoCommit(false);

// Add multiple orders to batch

statement.setInt(1, 101);

statement.setString(2, "Laptop");

statement.setInt(3, 2);

statement.addBatch();

statement.setInt(1, 102);

statement.setString(2, "Smartphone");

statement.setInt(3, 5);

statement.addBatch();

statement.setInt(1, 103);

statement.setString(2, "Tablet");

statement.setInt(3, 3);

statement.addBatch();

// Execute the batch

statement.executeBatch();

// Commit the transaction

connection.commit();

System.out.println("Batch insertion successful.");

} catch (SQLException e) {

e.printStackTrace();

}

}

}