Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

(SCOPE)

Fall Semester 2023-24

CSI1001 - PRINCIPLES OF DATABASE SYSTEMS

SLOT-L47+L48

Database Application development

Inventory Management System

By

1. Rayan Banerjee (22MIC0062)

2. Sujal Ravindra Dixit (22MIC0115)

ABSTRACT

- For companies of all sizes, inventory management is an essential procedure that can lower expenses, boost efficiency, and enhance customer satisfaction with a well-designed system. This project report details the planning, development, and deployment of an inventory management system for a small company, emphasizing the system's usability, cost, and scalability.

- A few of the general benefits about IMS's are:

1. Reduce costs: By avoiding stockouts and overstocking, businesses can reduce their inventory carrying costs.

2. Improve customer satisfaction: By ensuring that products are available when customers want them, businesses can improve customer satisfaction and loyalty.

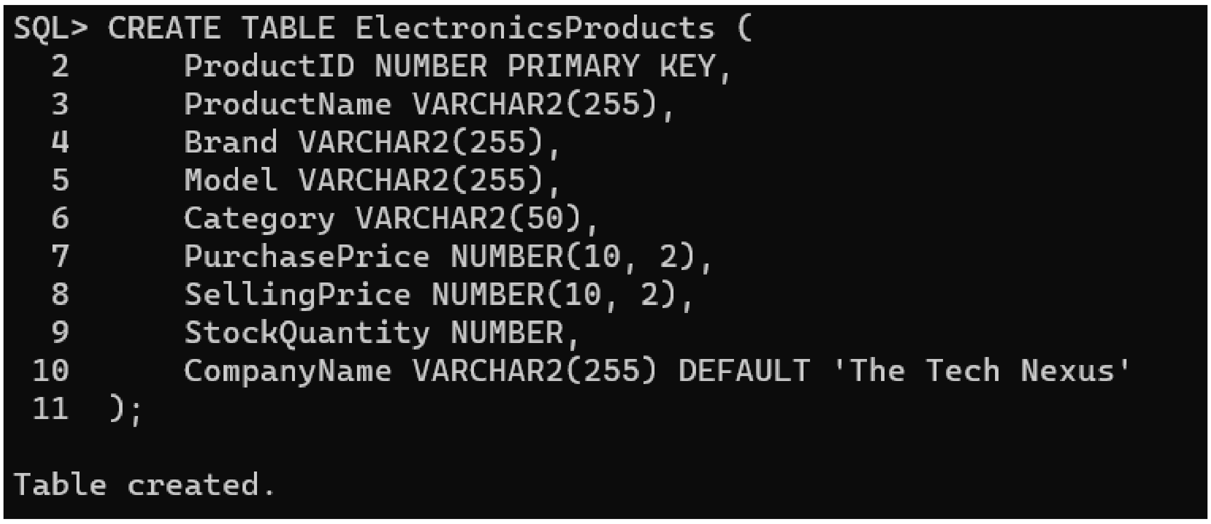
3. Increase efficiency: By automating inventory tracking and reporting tasks, IMS's can save businesses time and resources.

- The system stores data on inventory items, sales transactions, and purchase orders in a relational database. It offers automatic reorder point generation, purchase order management, real-time visibility into inventory levels, and a variety of inventory reports.

- All things considered, the inventory management system covered in this project is a complete and approachable solution that can assist small businesses in more efficiently managing their inventories.

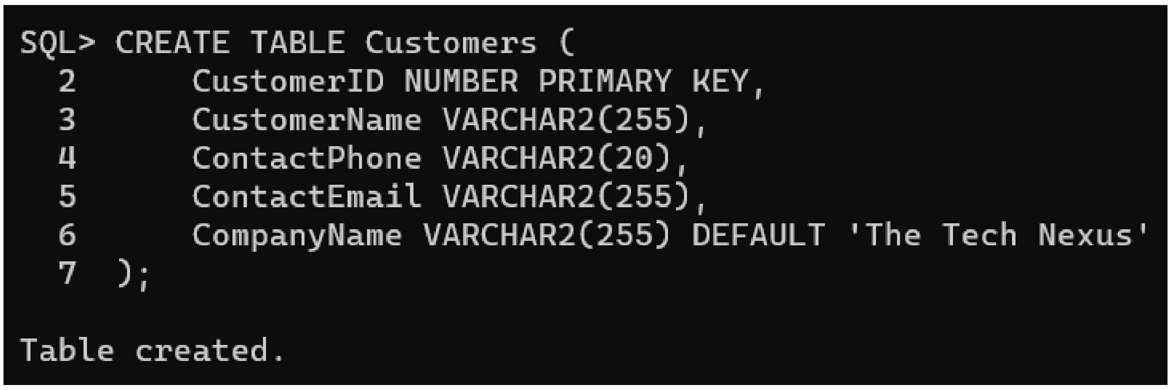
Database Design

Product Table:



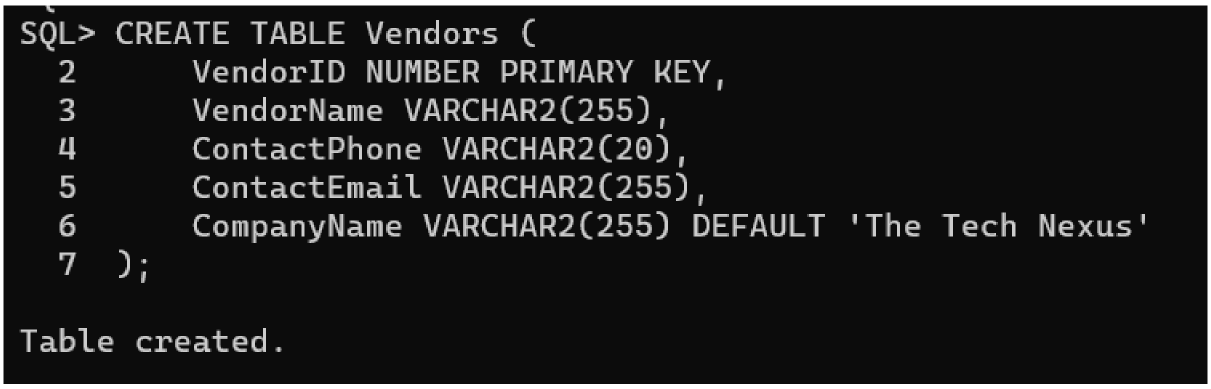
```
SQL> CREATE TABLE ElectronicsProducts (
         ProductID NUMBER PRIMARY KEY,
         ProductName VARCHAR2(255),
         Brand VARCHAR2(255),
         Model VARCHAR2(255),
         Category VARCHAR2(50),
         PurchasePrice NUMBER(10, 2),
         SellingPrice NUMBER(10, 2),
         StockQuantity NUMBER,
         CompanyName VARCHAR2(255) DEFAULT 'The Tech Nexus'
     );

Table created.
```

Customer Table:



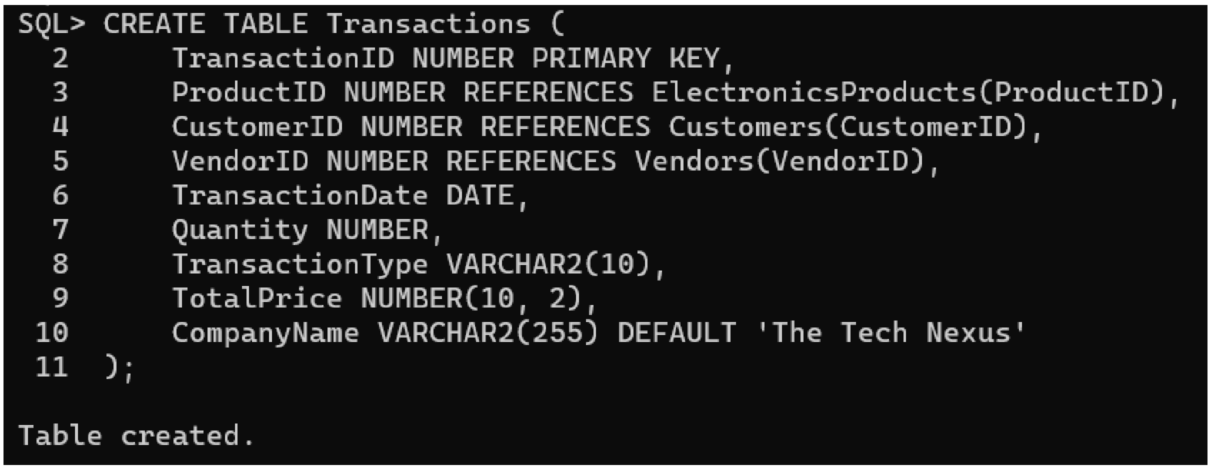
```
SQL> CREATE TABLE Customers (
         CustomerID NUMBER PRIMARY KEY,
         CustomerName VARCHAR2(255),
         ContactPhone VARCHAR2(20),
         ContactEmail VARCHAR2(255),
         CompanyName VARCHAR2(255) DEFAULT 'The Tech Nexus'
     );

Table created.
```

Vendors Table:



```
SQL> CREATE TABLE Vendors (
         VendorID NUMBER PRIMARY KEY,
         VendorName VARCHAR2(255),
         ContactPhone VARCHAR2(20),
         ContactEmail VARCHAR2(255),
         CompanyName VARCHAR2(255) DEFAULT 'The Tech Nexus'
     );

Table created.
```

Transaction Table:



```
SQL> CREATE TABLE Transactions (
         TransactionID NUMBER PRIMARY KEY,
         ProductID NUMBER REFERENCES ElectronicsProducts(ProductID),
         CustomerID NUMBER REFERENCES Customers(CustomerID),
         VendorID NUMBER REFERENCES Vendors(VendorID),
         TransactionDate DATE,
         Quantity NUMBER,
         TransactionType VARCHAR2(10),
         TotalPrice NUMBER(10, 2),
         CompanyName VARCHAR2(255) DEFAULT 'The Tech Nexus'
     );

Table created.
```

Sample Program

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import javafx.scene.Node;


public class App extends Application {
```

```
private static final String DB_URL = "jdbc:oracle:thin:@LAPTOP-GENOS:1522:XE";
private static final String DB_USER = "SYSTEM";
private static final String DB_PASSWORD = "rayan62";

private Connection connection;

TextField productIdField = new TextField();
TextField productNameField = new TextField();
TextField brandField = new TextField();
TextField modelField = new TextField();
TextField categoryField = new TextField();
TextField purchasePriceField = new TextField();
TextField sellingPriceField = new TextField();
TextField stockQuantityField = new TextField();

TextField customerIdField = new TextField();
TextField customerNameField = new TextField();
TextField contactPhoneField = new TextField();
TextField contactEmailField = new TextField();

TextField vendorIdField = new TextField();
TextField vendorNameField = new TextField();
TextField vendorContactPhoneField = new TextField();
TextField vendorContactEmailField = new TextField();

TextField transactionIdField = new TextField();
TextField transactionProductIdField = new TextField();
TextField transactionCustomerIdField = new TextField();
TextField transactionVendorIdField = new TextField();
TextField transactionDateField = new TextField();
TextField transactionQuantityField = new TextField();
TextField transactionTypeField = new TextField();
TextField transactionTotalPriceField = new TextField();
TextField companyNameField = new TextField();
TextField quantityField = new TextField();
TextField totalPriceField = new TextField();







public static void main(String[] args) {
   launch(args);
}

@Override
public void start(Stage stage) {
   initDatabase();

   // Create a JavaFX UI
   VBox root = new VBox(10);
   root.setPadding(new Insets(10));
   Scene scene = new Scene(root, 1280, 720);

// Create a new ScrollPane
ScrollPane scrollPane = new ScrollPane();

// Set its content to your VBox layout
scrollPane.setContent(buttonLayout);
```

```
    // Add the ScrollPane to your scene or another layout
    Scene scene1 = new Scene(scrollPane, 1920, 1080); // Replace 800 and 600 with your
desired scene width and height

    Stage.setScene(scene); // Set the scene on the primary stage

    primaryStage.show(); // Display the stage
}

        // Create buttons for fetching data
Button fetchProductsButton = new Button("Fetch Products");
Button fetchCustomersButton = new Button("Fetch Customers");
Button fetchVendorsButton = new Button("Fetch Vendors");
Button fetchTransactionsButton = new Button("Fetch Transactions");

// Create buttons for Product CRUD operations
Button insertProductButton = new Button("Insert Product");
Button updateProductButton = new Button("Update Product");
Button deleteProductButton = new Button("Delete Product");

// Create buttons for Customer CRUD operations
Button insertCustomerButton = new Button("Insert Customer");
Button updateCustomerButton = new Button("Update Customer");
Button deleteCustomerButton = new Button("Delete Customer");

// Create buttons for Vendor CRUD operations
```

```
Button insertVendorButton = new Button("Insert Vendor");
Button updateVendorButton = new Button("Update Vendor");
Button deleteVendorButton = new Button("Delete Vendor");

// Create buttons for Transaction CRUD operations
Button insertTransactionButton = new Button("Insert Transaction");
Button updateTransactionButton = new Button("Update Transaction");
Button deleteTransactionButton = new Button("Delete Transaction");

// Add all buttons to the layout




// Style the buttons (you can adjust the styling as needed)
insertProductButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
updateProductButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
deleteProductButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
insertCustomerButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
updateCustomerButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
deleteCustomerButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
insertVendorButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
updateVendorButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
deleteVendorButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
insertTransactionButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
updateTransactionButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
deleteTransactionButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
fetchProductsButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
fetchCustomersButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
fetchVendorsButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
```

```
fetchTransactionsButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");


        Label titleLabel = new Label("The Tech Nexus");
        titleLabel.setStyle("-fx-font-size: 24px; -fx-text-fill: black;");
        titleLabel.setPadding(new Insets(0, 0, 10, 0));

        VBox buttonLayout = new VBox(10);

        // Product Management Section
        VBox productBox = createSection("Product Management", productIdField,
productNameField, brandField, modelField, categoryField, purchasePriceField,
sellingPriceField, stockQuantityField);
        productBox.getChildren().addAll(insertProductButton, updateProductButton,
deleteProductButton);
        buttonLayout.getChildren().add(productBox);

        // Customer Management Section
        VBox customerBox = createSection("Customer Management", customerIdField,
customerNameField, contactPhoneField, contactEmailField);
        customerBox.getChildren().addAll(insertCustomerButton, updateCustomerButton,
deleteCustomerButton);
        buttonLayout.getChildren().add(customerBox);

        // Vendor Management Section
        VBox vendorBox = createSection("Vendor Management", vendorIdField,
vendorNameField, vendorContactPhoneField, vendorContactEmailField);
        vendorBox.getChildren().addAll(insertVendorButton, updateVendorButton,
deleteVendorButton);
        buttonLayout.getChildren().add(vendorBox);

        // Transaction Management Section
        VBox transactionBox = createSection("Transaction Management", transactionIdField,
transactionProductIdField, transactionCustomerIdField, transactionVendorIdField,
transactionDateField, transactionQuantityField, transactionTypeField,
transactionTotalPriceField, companyNameField);

        transactionBox.getChildren().addAll(insertTransactionButton, updateTransactionButton,
deleteTransactionButton);

        buttonLayout.getChildren().add(transactionBox);





root.getChildren().add(buttonLayout);

        // Text area for displaying data

        TextArea resultTextArea = new TextArea();

        resultTextArea.setWrapText(true);

        resultTextArea.setEditable(true);


        // Create HBox for fetch buttons

        HBox fetchButtonsBox = new HBox(10);

        fetchButtonsBox.getChildren().addAll(fetchProductsButton, fetchCustomersButton,
fetchVendorsButton, fetchTransactionsButton);


        // Layout setup

        HBox resultBox = new HBox(10);

        resultBox.getChildren().add(resultTextArea);


        // Add all sections to the root VBox

        root.getChildren().addAll(titleLabel, productBox, customerBox, vendorBox,
transactionBox, fetchButtonsBox, resultBox);


        // Create a new ScrollPane

ScrollPane scrollPane = new ScrollPane();
```

```
// Set its content to your VBox layout
scrollPane.setContent(buttonLayout);


        stage.setTitle("The Tech Nexus: Inventory Manager");
        stage.setScene(scene);
        stage.show();
    }



    private void initDatabase() {
        try {
            // Load the Oracle JDBC driver
            Class.forName("oracle.jdbc.driver.OracleDriver");


            connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }

    // Implement your button styling method
    private Button styledButton(String text) {
        Button button = new Button(text);
        // Apply your button styling here
        return button;
    }

    // Implement your data manipulation methods (insert, update, delete, fetch) here
```

```
private void updateProduct(String productId, String productName, String brand, String
model, String category,
String purchasePrice, String sellingPrice, String stockQuantity) {
if (productId.isEmpty() || productName.isEmpty()) return;

try {
String query = "UPDATE ElectronicsProducts " +
"SET ProductName = ?, Brand = ?, Model = ?, Category = ?, PurchasePrice = ?, SellingPrice
= ?, StockQuantity = ? " +
"WHERE ProductID = ?";

PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setString(1, productName);
preparedStatement.setString(2, brand);
preparedStatement.setString(3, model);
preparedStatement.setString(4, category);
preparedStatement.setString(5, purchasePrice);
preparedStatement.setString(6, sellingPrice);
preparedStatement.setString(7, stockQuantity);
preparedStatement.setString(8, productId);

preparedStatement.executeUpdate();
} catch (SQLException e) {
e.printStackTrace();
}
}

// Delete a product
private void deleteProduct(String productId) {
if (productId.isEmpty()) return;
```

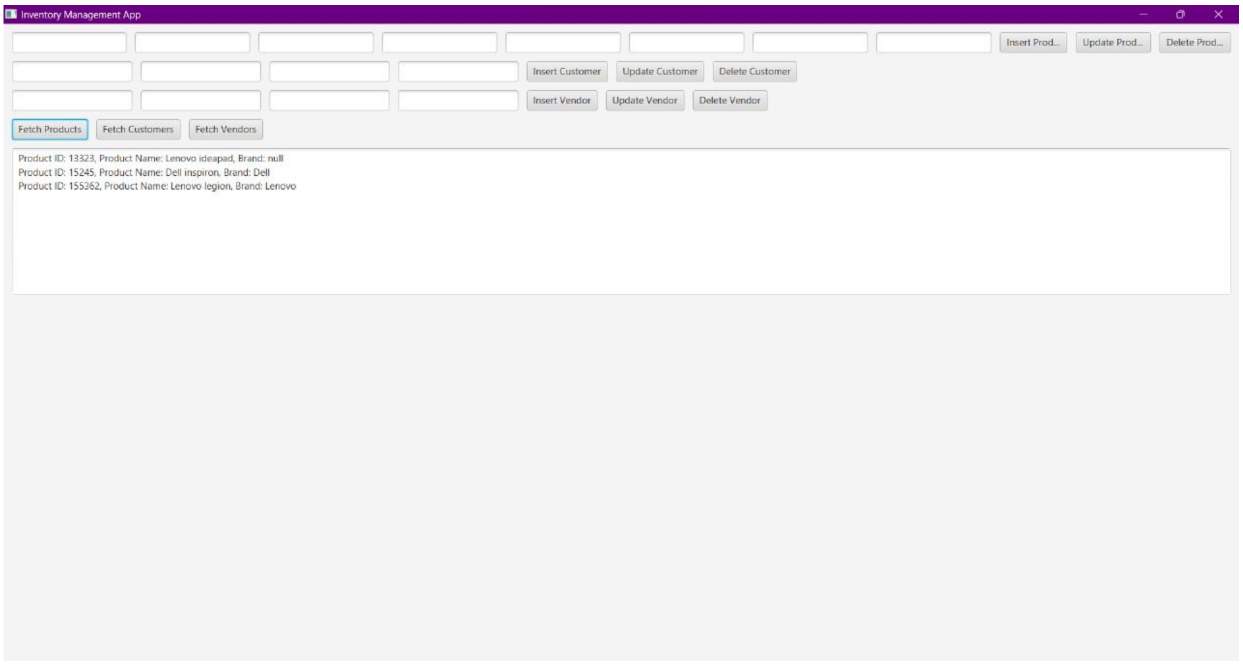
```
try {
String query = "DELETE FROM ElectronicsProducts WHERE ProductID = ?";

PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setString(1, productId);

preparedStatement.executeUpdate();
} catch (SQLException e) {
e.printStackTrace();
}
}



   private void insertProduct(String productId, String productName, String brand, String
model, String category,
                        String purchasePrice, String sellingPrice, String stockQuantity) {
        if (productId.isEmpty() || productName.isEmpty()) return;

        try {
            String query = "INSERT INTO ElectronicsProducts (ProductID, ProductName,
Brand, Model, Category, PurchasePrice, SellingPrice, StockQuantity) " +
                "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, productId);
            preparedStatement.setString(2, productName);
            preparedStatement.setString(3, brand);
            preparedStatement.setString(4, model);
            preparedStatement.setString(5, category);
            preparedStatement.setString(6, purchasePrice);
            preparedStatement.setString(7, sellingPrice);
            preparedStatement.setString(8, stockQuantity);

            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


private void insertCustomer(String customerId, String customerName, String contactPhone,
String contactEmail) {
    if (customerId.isEmpty() || customerName.isEmpty()) return;

    try {
        String query = "INSERT INTO Customers (CustomerID, CustomerName,
ContactPhone, ContactEmail) " +
            "VALUES (?, ?, ?, ?)";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, customerId);
        preparedStatement.setString(2, customerName);
        preparedStatement.setString(3, contactPhone);
        preparedStatement.setString(4, contactEmail);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
private void updateCustomer(String customerId, String customerName, String contactPhone,
String contactEmail) {
    if (customerId.isEmpty() || customerName.isEmpty()) return;

    try {
        String query = "UPDATE Customers " +
            "SET CustomerName = ?, ContactPhone = ?, ContactEmail = ? " +
            "WHERE CustomerID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, customerName);
        preparedStatement.setString(2, contactPhone);
        preparedStatement.setString(3, contactEmail);
        preparedStatement.setString(4, customerId);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}


private void deleteCustomer(String customerId) {
    if (customerId.isEmpty()) return;

    try {
        String query = "DELETE FROM Customers WHERE CustomerID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, customerId);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void insertVendor(String vendorId, String vendorName, String vendorContactPhone,
String vendorContactEmail) {
    if (vendorId.isEmpty() || vendorName.isEmpty()) return;


    try {
        String query = "INSERT INTO Vendors (VendorID, VendorName, ContactPhone,
ContactEmail) " +
            "VALUES (?, ?, ?, ?)";


        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, vendorId);
        preparedStatement.setString(2, vendorName);
        preparedStatement.setString(3, vendorContactPhone);
        preparedStatement.setString(4, vendorContactEmail);


        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}



private void updateVendor(String vendorId, String vendorName, String vendorContactPhone,
String vendorContactEmail) {
    if (vendorId.isEmpty() || vendorName.isEmpty()) return;

    try {
        String query = "UPDATE Vendors " +
            "SET VendorName = ?, ContactPhone = ?, ContactEmail = ? " +
            "WHERE VendorID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, vendorName);
        preparedStatement.setString(2, vendorContactPhone);
        preparedStatement.setString(3, vendorContactEmail);
        preparedStatement.setString(4, vendorId);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}


private void deleteVendor(String vendorId) {
    if (vendorId.isEmpty()) return;

    try {
        String query = "DELETE FROM Vendors WHERE VendorID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, vendorId);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}



    private void insertTransaction(String transactionId, String productId, String customerId,
String vendorId,
        String transactionDate, String quantity, String transactionType, String totalPrice, String
companyName) {
    if (transactionId.isEmpty() || productId.isEmpty() || customerId.isEmpty() ||
vendorId.isEmpty()) {
        return;
    }

    try {
        String query = "INSERT INTO Transactions (TransactionID, ProductID, CustomerID,
VendorID, TransactionDate, Quantity, TransactionType, TotalPrice, CompanyName) " +
            "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, transactionId);
        preparedStatement.setString(2, productId);
        preparedStatement.setString(3, customerId);
        preparedStatement.setString(4, vendorId);
        preparedStatement.setString(5, transactionDate);
        preparedStatement.setString(6, quantity);
        preparedStatement.setString(7, transactionType);
        preparedStatement.setString(8, totalPrice);
        preparedStatement.setString(9, companyName);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void updateTransaction(
    String transactionId, String productId, String customerId, String vendorId,
    String transactionDate, String quantity, String transactionType,
    String totalPrice, String companyName) {
    if (transactionId.isEmpty()) return;

    try {
        String query = "UPDATE YourTransactionTable " +
            "SET ProductID = ?, CustomerID = ?, VendorID = ?, TransactionDate = ?, Quantity =
?, " +
            "TransactionType = ?, TotalPrice = ?, CompanyName = ? " +
            "WHERE TransactionID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, productId);
        preparedStatement.setString(2, customerId);
        preparedStatement.setString(3, vendorId);
        preparedStatement.setString(4, transactionDate);
        preparedStatement.setString(5, quantity);
        preparedStatement.setString(6, transactionType);
        preparedStatement.setString(7, totalPrice);
        preparedStatement.setString(8, companyName);
        preparedStatement.setString(9, transactionId);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void deleteTransaction(String transactionId) {
    if (transactionId.isEmpty()) return;

    try {
        String query = "DELETE FROM YourTransactionTable WHERE TransactionID = ?";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, transactionId);

        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}




    private void fetchProducts(TextArea resultTextArea) {
        try {
            String query = "SELECT * FROM ElectronicsProducts";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            ResultSet resultSet = preparedStatement.executeQuery();

            StringBuilder result = new StringBuilder();
            while (resultSet.next()) {
                int productId = resultSet.getInt("ProductID");
                String productName = resultSet.getString("ProductName");
                String brand = resultSet.getString("Brand");
                result.append("Product ID: ").append(productId).append(", Product Name: ").append(productName).append(", Brand: ").append(brand).append("\n");
            }

            resultTextArea.setText(result.toString());
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }



private void fetchCustomers(TextArea resultTextArea) {
    try {
        String query = "SELECT * FROM Customers";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();

        StringBuilder result = new StringBuilder();
        while (resultSet.next()) {
            int customerId = resultSet.getInt("CustomerID");
            String customerName = resultSet.getString("CustomerName");
            String contactPhone = resultSet.getString("ContactPhone");
            result.append("Customer ID: ").append(customerId).append(", Customer Name: ").append(customerName).append(", Contact Phone: ").append(contactPhone).append("\n");
        }

        resultTextArea.setText(result.toString());
    } catch (SQLException e) {
        e.printStackTrace();
    }
}


private void fetchVendors(TextArea resultTextArea) {
    try {
        String query = "SELECT * FROM Vendors";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();

        StringBuilder result = new StringBuilder();
        while (resultSet.next()) {
            int vendorId = resultSet.getInt("VendorID");
            String vendorName = resultSet.getString("VendorName");
            String contactPhone = resultSet.getString("ContactPhone");
            result.append("Vendor ID: ").append(vendorId).append(", Vendor Name: 
").append(vendorName).append(", Contact Phone: ").append(contactPhone).append("\n");
        }

        resultTextArea.setText(result.toString());
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void fetchTransactions(TextArea resultTextArea) {
    try {
        String query = "SELECT * FROM Transactions";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();

        StringBuilder result = new StringBuilder();
        while (resultSet.next()) {
            int transactionId = resultSet.getInt("TransactionID");
            int productId = resultSet.getInt("ProductID");
            int customerId = resultSet.getInt("CustomerID");
            int vendorId = resultSet.getInt("VendorID");
            String transactionDate = resultSet.getString("TransactionDate");
            int quantity = resultSet.getInt("Quantity");
            String transactionType = resultSet.getString("TransactionType");
            double totalPrice = resultSet.getDouble("TotalPrice");
            String companyName = resultSet.getString("CompanyName");

            result.append("Transaction ID: ").append(transactionId).append(", Product ID: 
").append(productId)
                .append(", Customer ID: ").append(customerId).append(", Vendor ID: 
").append(vendorId)
                .append(", Transaction Date: ").append(transactionDate).append(", Quantity: 
").append(quantity)
                .append(", Transaction Type: ").append(transactionType).append(", Total Price: 
").append(totalPrice)
                .append(", Company Name: ").append(companyName).append("\n");
        }

        resultTextArea.setText(result.toString());
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

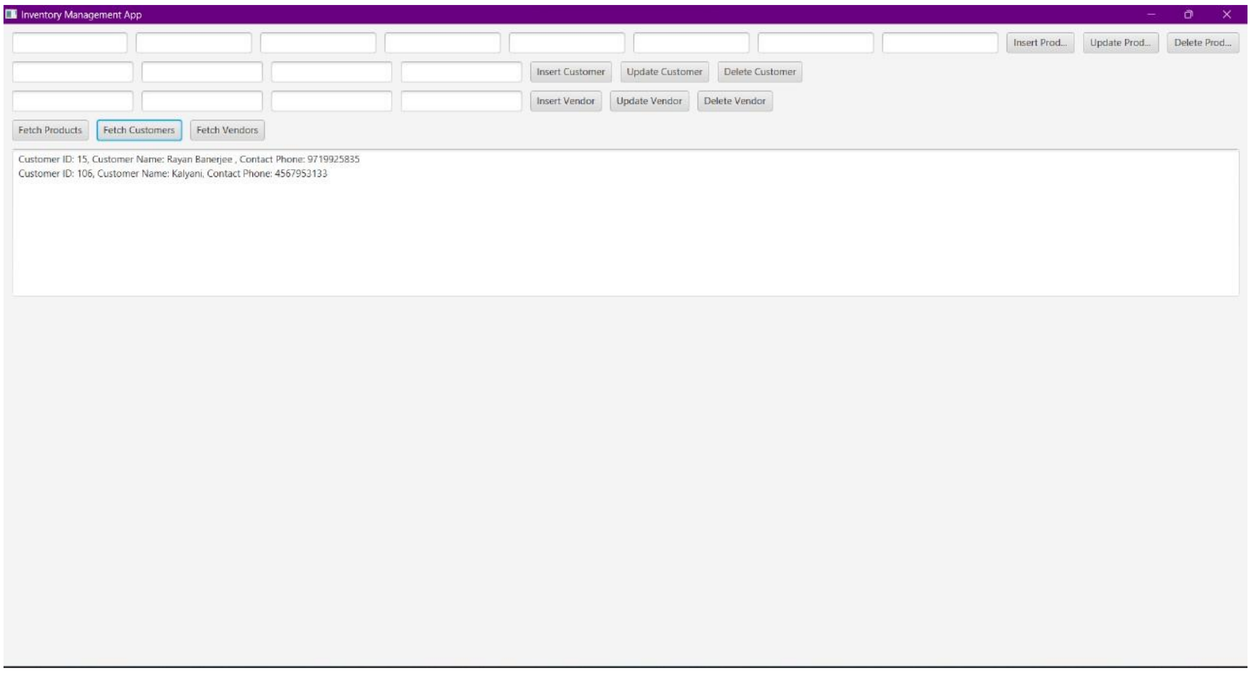
```
private void clearTransactionFields(
        TextField transactionIdField, TextField productIdField, TextField customerIdField,
        TextField vendorIdField, TextField transactionDateField, TextField quantityField,
        TextField transactionTypeField, TextField totalPriceField, TextField
companyNameField) {
    transactionIdField.clear();
    productIdField.clear();
    customerIdField.clear();
    vendorIdField.clear();
    transactionDateField.clear();
    quantityField.clear();
    transactionTypeField.clear();
    totalPriceField.clear();
    companyNameField.clear();
}


    private void clearProductFields(TextField productId, TextField productName, TextField
brand, TextField model,
                                    TextField category, TextField purchasePrice, TextField sellingPrice,
TextField stockQuantity) {
        productId.clear();
        productName.clear();
        brand.clear();
        model.clear();
        category.clear();
        purchasePrice.clear();
        sellingPrice.clear();
        stockQuantity.clear();
    }
```

```
private void clearCustomerFields(TextField customerId, TextField customerName, TextField
contactPhone, TextField contactEmail) {
    customerId.clear();
    customerName.clear();
    contactPhone.clear();
    contactEmail.clear();
}
public  start2(Stage primaryStage) {
    // Your setup code here...

    // Create a new ScrollPane
    ScrollPane scrollPane = new ScrollPane();

    // Set its content to your VBox layout
    scrollPane.setContent(buttonLayout);

    // Add the ScrollPane to your scene or another layout
    Scene scene = new Scene(scrollPane, 800, 600); // Replace 800 and 600 with your desired
scene width and height

    primaryStage.setScene(scene); // Set the scene on the primary stage

    primaryStage.show(); // Display the stage
}


private void clearVendorFields(TextField vendorId, TextField vendorName, TextField
vendorContactPhone, TextField vendorContactEmail) {
    vendorId.clear();
    vendorName.clear();
    vendorContactPhone.clear();
    vendorContactEmail.clear();}

    public void clearTransactionFields(TextField... fields) {
        for (TextField field : fields) {
            field.clear();
        }
    }

    // Create a VBox for a section with a title and fields
    private VBox createSection(String title, Node... fields) {
        VBox section = new VBox(10);
        section.setPadding(new Insets(10));
        section.getChildren().addAll(new Label(title), new HBox(10, fields));
        return section;
    }
}
```

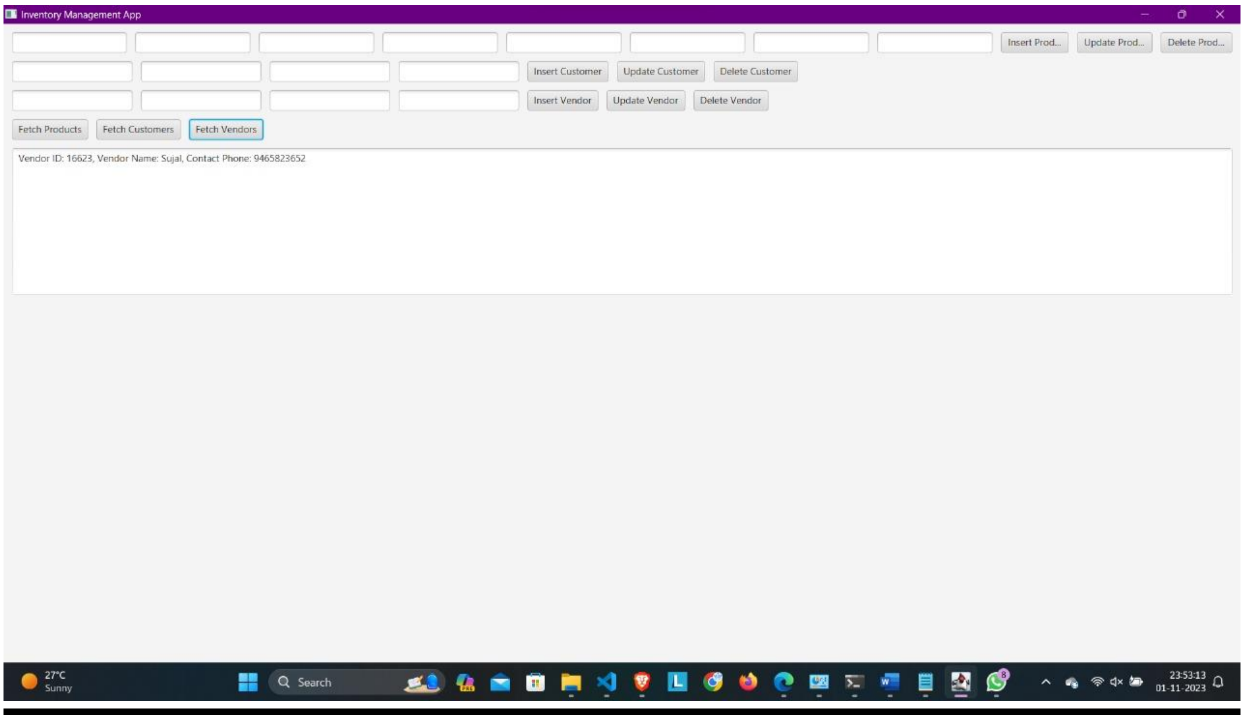
Program Output



```
Product ID: 13323, Product Name: Lenovo ideapad, Brand: null
Product ID: 15245, Product Name: Dell inspiron, Brand: Dell
Product ID: 155362, Product Name: Lenovo legion, Brand: Lenovo
```



```
Customer ID: 15, Customer Name: Rayan Banerjee , Contact Phone: 9719925835
Customer ID: 106, Customer Name: Kalyani, Contact Phone: 4567953133
```



```
Vendor ID: 16623, Vendor Name: Sujal, Contact Phone: 9465823652
```