Ecommerce Management System (Python Tkinter + SQLite)

Mini Project Report
Submitted By:Rishabh Kumar
Roll Number: 26
Class/Sem: D10B SEM III
Department:INFT

1. Introduction

The Ecommerce Management System is a comprehensive desktop application developed to manage online store operations efficiently. It focuses on product inventory management, customer relationship management, and order processing in a structured manner. The system reduces manual record-keeping, ensures accuracy, and provides a simple and user-friendly interface for store managers and business owners.

2. Objectives

- Maintain product, customer, and order records electronically
- Implement CRUD operations (Create, Read, Update, Delete) for all entities
- Provide an easy-to-use graphical user interface using Python Tkinter
- Connect the application with SQLite database for persistent storage
- Ensure data validation and business logic integrity
- Automate inventory management and order processing

3. Tools & Technologies Used

Component	Technology
Programming Language	Python

GUI Framework	Tkinter
Database	SQLite
Database Interface	sqlite3
IDE	VS Code / PyCharm / Any Python IDE

4. System Design

The Ecommerce Management System follows a client-server architecture:

- Client: Python Tkinter GUI for user interaction
- Server: SQLite database stores product, customer, and order records
- Communication: SQLite3 library for database connectivity

Database Table Structure

Products Table:

- id (INTEGER PRIMARY KEY)
- name (TEXT NOT NULL)
- category (TEXT NOT NULL)
- price (REAL NOT NULL)
- stock (INTEGER NOT NULL)
- description (TEXT)

Customers Table:

- id (INTEGER PRIMARY KEY)
- name (TEXT NOT NULL)
- email (TEXT NOT NULL)
- phone (TEXT)
- address (TEXT)

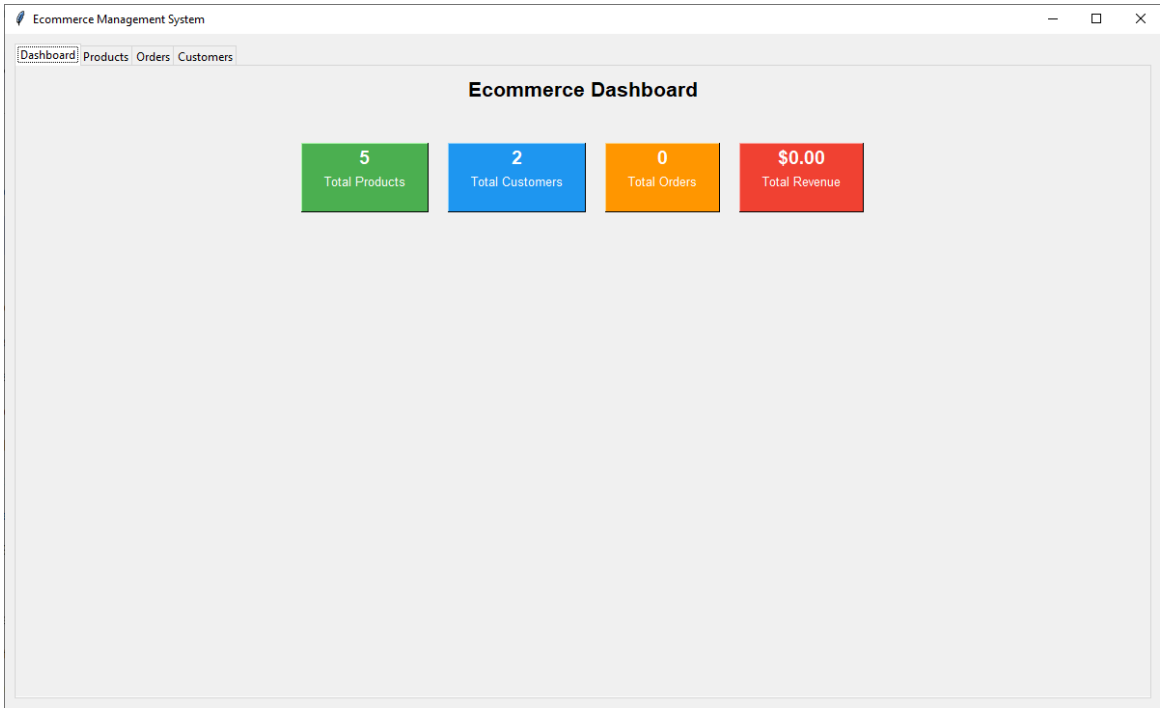
Orders Table:

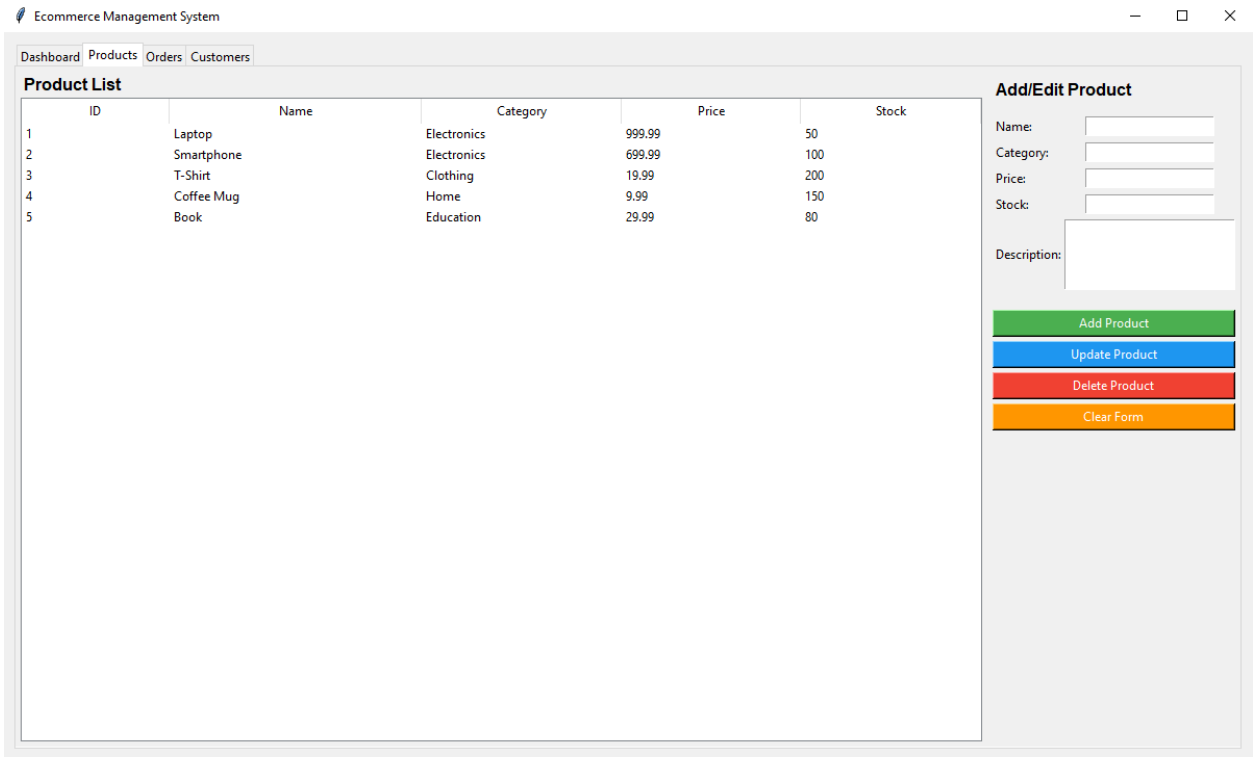
- id (INTEGER PRIMARY KEY)
- customer_id (INTEGER)
- product_id (INTEGER)
- quantity (INTEGER)
- total_price (REAL)
- order_date (TEXT)
- status (TEXT)
- FOREIGN KEY (customer_id) REFERENCES customers(id)
- FOREIGN KEY (product_id) REFERENCES products(id)

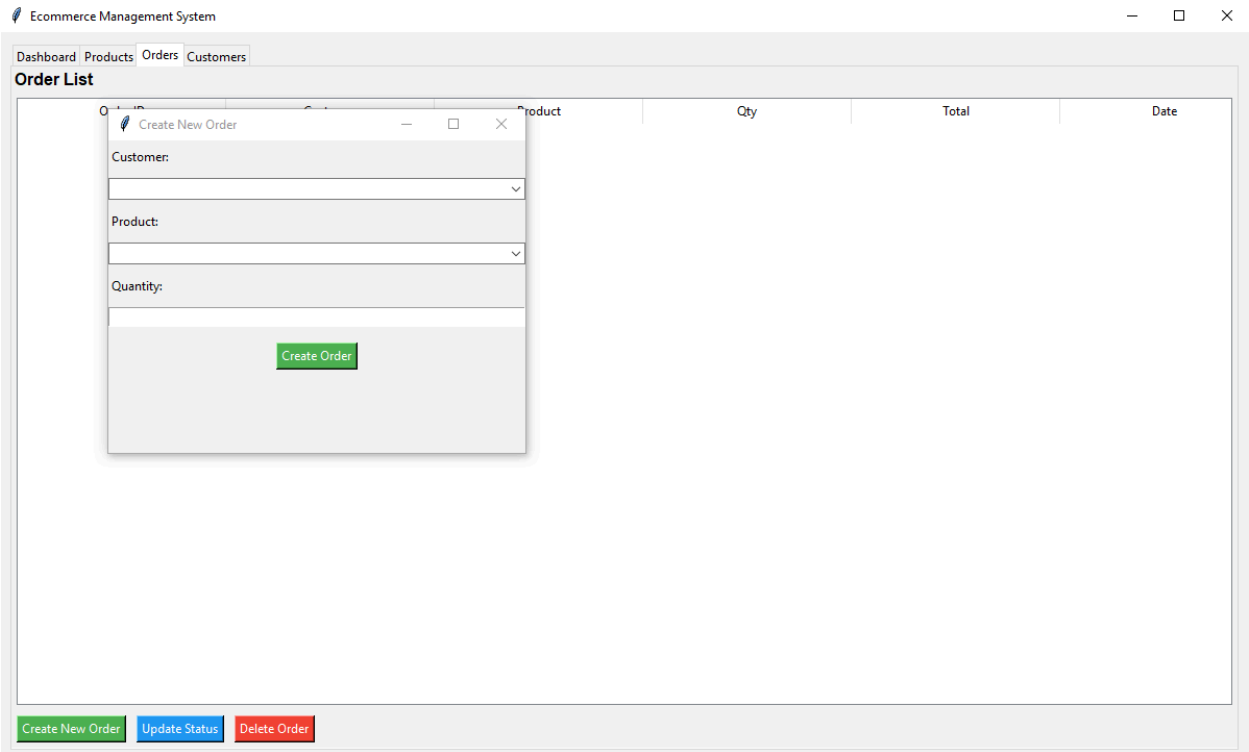
5. Features Implemented

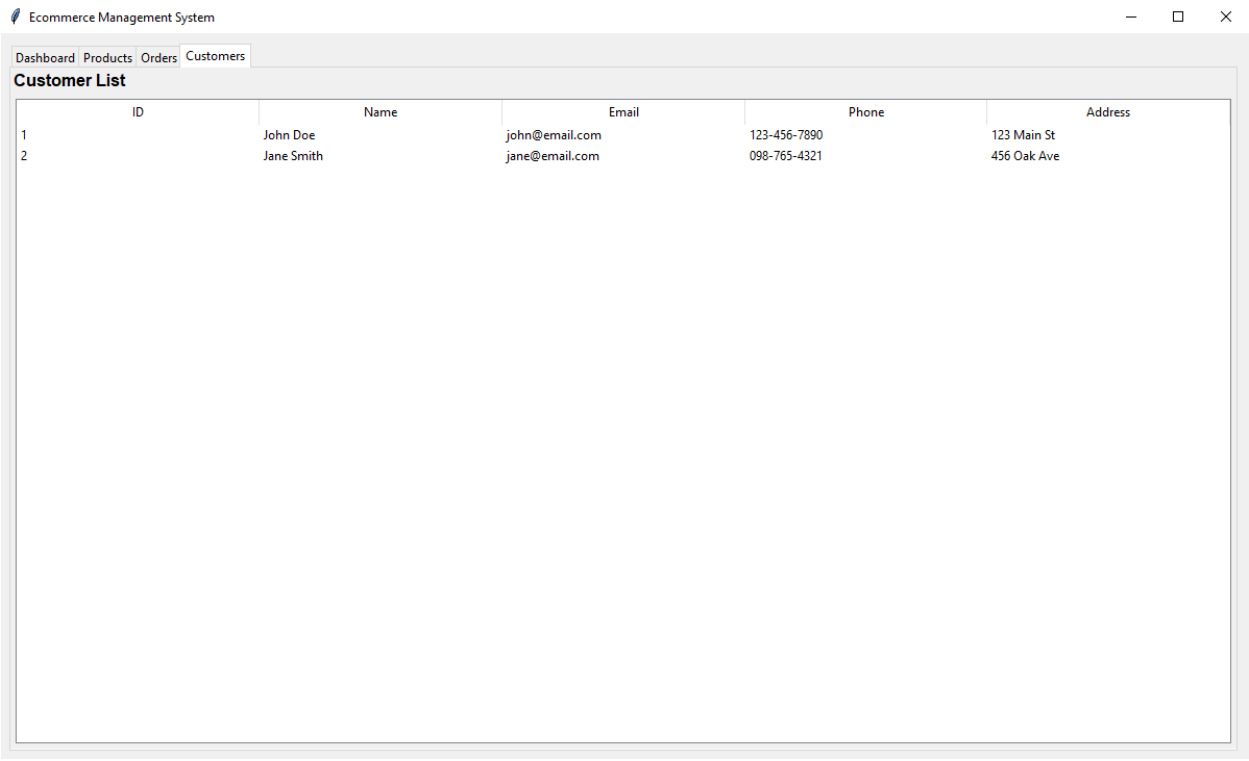
- Dashboard: Real-time business statistics and KPIs display
- Add Product: Add new product records into the database
- View Products: Display all products in a scrollable treeview
- Update Product: Modify existing product details using selection
- Delete Product: Remove product records with confirmation
- Order Management: Create, view, update, and delete orders
- Customer Management: Maintain customer database
- Inventory Automation: Automatic stock updates on order placement
- Input Validation: Ensures price and stock are numeric and required fields are filled
- User-friendly GUI: Designed using Tkinter with proper tabbed layout and labels

6.ScreenShots


Ecommerce Management System
Dashboard
Products
Orders
Customers
Ecommerce Dashboard
5
Total Products
2
Total Customers
0
Total Orders
$0.00
Total Revenue


Ecommerce Management System
Dashboard
Products
Orders
Customers
Product List
ID
Name
Category
Price
Stock
1
Laptop
Electronics
999.99
50
2
Smartphone
Electronics
699.99
100
3
T-Shirt
Clothing
19.99
200
4
Coffee Mug
Home
9.99
150
5
Book
Education
29.99
80
Add/Edit Product
Name:
Category:
Price:
Stock:
Description:
Add Product
Update Product
Delete Product
Clear Form





Code Snippets

Database Connection:

```python
self.conn = sqlite3.connect('ecommerce.db')
self.cursor = self.conn.cursor()
```

Add Product:

```python
def add_product(self):
    name = self.product_name.get()
    category = self.product_category.get()
    price = float(self.product_price.get())
    stock = int(self.product_stock.get())
    description = self.product_description.get("1.0", tk.END).strip()

    self.cursor.execute('''
        INSERT INTO products (name, category, price, stock, description)
        VALUES (?, ?, ?, ?, ?)
    ''', (name, category, price, stock, description))
```

Delete Product:

```python
def delete_product(self):
    product_id = self.products_tree.item(selected[0])['values'][0]
    self.cursor.execute("DELETE FROM products WHERE id=?", (product_id,))
```

Create Order:

```python
def create_order(self):
    customer_id = int(customer_var.get().split(' - ')[0])
    product_id = int(product_var.get().split(' - ')[0])
    quantity = int(quantity_entry.get())

    self.cursor.execute('''
```

```
        INSERT INTO orders (customer_id, product_id, quantity, total_price,
order_date, status)
        VALUES (?, ?, ?, ?, ?, ?)
    ''', (customer_id, product_id, quantity, total_price, datetime.now(),
"Pending"))
```

7. Flow Diagram

```
text
```

```
+----------------------+
|  Start Application   |
+----------------------+
           |
           v
+----------------------+
| Initialize Database |
+----------------------+
           |
           v
+----------------------+
|   Load Sample Data   |
+----------------------+
           |
           v
+----------------------+
|  Display Main GUI    |
+----------------------+
           |
           v
+----------------------+
|  User Selects Tab    |
| Dashboard/Products/ |
|  Orders/Customers    |
+----------------------+
           |
           v
+----------------------+
|  Perform Operations |
|  Add/View/Update/    |
```

```
| Delete/Create Order |
+---------------------+
           |
           v
+----------------------+
|  Update Database &   |
|    Refresh Display   |
+----------------------+
           |
           v
+----------------------+
|      Exit App        |

+----------------------+
```

8. Advantages

- Eliminates manual record-keeping errors in ecommerce operations
- Saves time for store managers with automated inventory management
- Easy to maintain and update product, customer, and order information
- Provides real-time business insights through dashboard
- Automated stock management prevents overselling
- Can be extended for payment processing, shipping, and analytics

9. Limitations

- Basic GUI using Tkinter; can be modernized with custom tkinter or other frameworks
- No advanced security features like user authentication
- Limited to single-store operations; no multi-vendor support
- No integration with payment gateways or shipping providers
- Basic reporting capabilities without advanced analytics

10. Future Enhancements

- Implement user authentication and role-based access control
- Add payment gateway integration (Stripe, PayPal)

- Implement inventory alerts and reordering system
- Add advanced reporting and analytics dashboard
- Integrate with shipping carriers (UPS, FedEx)
- Develop web-based version using Django or Flask
- Add product image support and gallery
- Implement shopping cart and wishlist functionality
- Add customer review and rating system

11. Conclusion

The Ecommerce Management System is a comprehensive yet effective application to manage online store operations electronically. It demonstrates Python Tkinter GUI development, SQLite database connectivity, and complex CRUD operations with business logic. The system successfully handles product management, customer relationships, and order processing with automated inventory tracking. This mini project serves as a solid foundation to build more sophisticated ecommerce management software in the future, with potential for web integration, multi-vendor support, and advanced business intelligence features.