Python Fundamentals Exercise

Case Study Title: "BookMart: A Mini Bookstore Management System"

Scenario

You are hired by a small bookstore, *BookMart*, to develop a Python-based application that helps them manage their inventory, customers, and sales records. The application should be simple, menu-driven, and demonstrate the use of Python fundamentals, modularity, reusable functions, object-oriented programming (OOP), and proper error handling.

Problem Statement

The application should perform the following tasks:

1. **Book Management**
    - Add new books to the inventory (title, author, price, and quantity).
    - View the list of all available books.
    - Search for a book by its title or author.
2. **Customer Management**
    - Add customer details (name, email, and phone number).
    - View the list of all customers.
3. **Sales Management**
    - Sell a book to a customer (reduce the book quantity after a sale and log the transaction).
    - View all sales records.
4. **Error Handling**
    - Handle cases where the book is out of stock.
    - Ensure proper data validation for customer details and book details (e.g., price and quantity must be positive numbers).
5. **Modularity**

    - Split the code into multiple Python modules for better organization (e.g., book_management.py, customer_management.py, sales_management.py, and main.py).

Requirements:

Python Fundamentals

- Use lists, dictionaries, and strings to manage data.
- Implement basic input and output for user interaction.

Control and Looping Constructs

- Use loops for menu navigation and data iteration.
- Use conditional statements for validating inputs and implementing logic.

Functions

- Use reusable functions for tasks like adding books, searching books, and selling books.

Object-Oriented Programming (OOP)

- Implement Book and Customer classes with appropriate attributes and methods.
- Use inheritance to create a Transaction class that extends the functionality of the Customer class for managing sales records.

Exception Handling

- Handle exceptions such as invalid user input (e.g., entering text when a number is expected).
- Handle cases where a book is not available in sufficient quantity for a sale.

Exercise Tasks

Task 1: Create the Data Models (Classes)

- Create a Book class with attributes: title, author, price, and quantity. Add methods to display book details.
- Create a Customer class with attributes: name, email, and phone. Add methods to display customer details.

- Create a Transaction class that inherits from Customer and adds attributes for the book_title and quantity_sold.

Task 2: Create the Modules

- **book_management.py:**
  - Functions to add a book, view all books, and search for a book.
- **customer_management.py:**
  - Functions to add a customer and view all customers.
- **sales_management.py:**
  - Functions to sell a book and view all sales records.

Task 3: Implement Exception Handling

- Handle invalid inputs (e.g., non-numeric input for price/quantity).
- Handle the case when trying to sell a book that is out of stock or doesn't exist.

Task 4: Create the Main Program

- Create a main.py file that imports the above modules.
- Implement a menu-driven program that allows the user to choose operations like managing books, customers, and sales.

Expected Output

Sample Menu:

```
Welcome to BookMart!

1. Book Management

2. Customer Management

3. Sales Management

4. Exit


Enter your choice:
```

Example Input and Output Scenarios

1. **Add Book**
   Input:

   ```
   Title: Python 101

   Author: John Doe

   Price: 500

   Quantity: 10
   ```

   Output:

   ```
   Book added successfully!
   ```

2. **Sell Book**
   Input:

   ```
   Customer Name: Alice

   Book Title: Python 101

   Quantity: 2
   ```

   Output:

   ```
   Sale successful! Remaining quantity: 8
   ```

3. **Handle Invalid Output**
   Input:

   ```
   Price: -500
   ```

   Output:

   ```
   Invalid input! Price must be a positive number.
   ```

4. **Out of Stock**
   Input:

   ```
   Book Title: Python 101
   ```

```
Quantity: 20
```

Output:

```
Error: Only 10 copies available. Sale cannot be
completed.
```