A Project Report on Store Ledger Tracker System

CAP776(Programming in Python)

Submitted by: Chetan Prakash Sharma
Reg No: 12326255
Roll No: 65
Lovely Professional University, Punjab

CA-1

Project Name: Store Ledger Tracker.

Introduction

The Shopkeeper Ledger Tracker is a user-friendly Python application designed to streamline the bookkeeping process for small business owners. By automating manual tasks and providing accurate record-keeping, this application significantly reduces the time and effort required to manage sales transactions and customer accounts.

Previously, shopkeepers faced the challenge of maintaining detailed ledgers manually, which was a time-consuming and error-prone process. The Shopkeeper Ledger Tracker addresses these issues by providing a digital solution that simplifies the bookkeeping process.

With this application, shopkeepers can easily track sales data, customer information, and outstanding dues. It eliminates the need for manual calculations and ensures accurate record-keeping, leading to improved financial management and decision-making.

Key Features

- **Product Database:** Predefined list of products with unique IDs, names, and prices.
- **Customer Management:** Easy entry and tracking of customer information.
- **Sales Transaction Recording:** Automated calculation of totals based on selected products and quantities.
- **Payment Tracking:** Records payment status for each transaction.
- **Ledger Maintenance:** Stores transaction details in a structured format for easy retrieval.
- **Customer Account Summary:** Provides a comprehensive view of each customer's transactions and outstanding dues.

How It Works

1. **Product Selection:** The shopkeeper chooses products from the predefined list.
2. **Customer Information:** The shopkeeper enters the customer's name.
3. **Transaction Details:** The shopkeeper specifies the quantity of each selected product.
4. **Payment Information:** The shopkeeper indicates whether the payment has been made or if there's an outstanding amount.
5. **Ledger Update:** The application automatically records the transaction in the ledger.

Advantages

- **Efficiency:** Automates manual tasks, saving time and effort.
- **Accuracy:** Eliminates human errors in calculations and record-keeping.
- **Organization:** Maintains a clear and structured ledger for easy reference.
- **Insights:** Provides valuable data for analyzing sales trends and customer behavior.
- **Scalability:** Can be easily adapted to accommodate growing businesses.

Disadvantages

- **Limited Reporting:** While the application provides basic transaction details, it might lack advanced reporting features for in-depth analysis.
- **Manual Data Entry:** While the system automates calculations, manual data entry for products and customer information is still required.
- **Lack of Integration:** The application might not integrate with other business systems like accounting software or point-of-sale systems.
- **Dependency on Data Accuracy:** The accuracy of the ledger depends on the correctness of the data entered by the shopkeeper.

Actual Code of the project

```
product_data = [

    {"index": 1, "product_name": "Garlic Oil - Vegetarian Capsule 500 mg",
"brand": "Sri Sri Ayurveda", "price": 220, "rating": 4.1, "stock": 10,
"sold": 5},

    {"index": 2, "product_name": "Water Bottle - Orange", "brand":
"Mastercook", "price": 180, "rating": 2.3, "stock": 20, "sold": 15},

    {"index": 3, "product_name": "Brass Angle Deep - Plain, No.2",
"brand": "Trm", "price": 119, "rating": 3.4, "stock": 5, "sold": 1},

    {"index": 4, "product_name": "Cereal Flip Lid Container/Storage Jar -
Assorted Colour", "brand": "Nakoda", "price": 149, "rating": 3.7},

    {"index": 5, "product_name": "Creme Soft Soap - For Hands & Body",
"brand": "Nivea", "price": 162, "rating": 4.4},

    {"index": 6, "product_name": "Germ - Removal Multipurpose Wipes",
"brand": "Nature Protect", "price": 169, "rating": 3.3},

    {"index": 7, "product_name": "Multani Mati", "brand": "Satinance",
"price": 58, "rating": 3.6},

    {"index": 8, "product_name": "Hand Sanitizer - 70% Alcohol Base",
"brand": "Bionova", "price": 250, "rating": 4},

    {"index": 9, "product_name": "Biotin & Collagen Volumizing Hair
Shampoo + Biotin & Collagen Hair Conditioner", "brand": "StBotanica",
"price": 1098, "rating": 3.5},

    {"index": 10, "product_name": "Scrub Pad - Anti- Bacterial, Regular",
"brand": "Scotch brite", "price": 20, "rating": 4.3},

    {"index": 11, "product_name": "Wheat Grass Powder - Raw", "brand":
"NUTRASHIL", "price": 261, "rating": 4},

    {"index": 12, "product_name": "Butter Cookies Gold Collection",
"brand": "Sapphire", "price": 600, "rating": 2.2},

    {"index": 13, "product_name": "Face Wash - Oil Control, Active",
"brand": "Oxy", "price": 110, "rating": 5},
    {"index": 14, "product_name": "Mold & Mildew Remover with Bleach",
"brand": "Clorox", "price": 350, "rating": 3.8},

    {"index": 15, "product_name": "Just Spray - Mosquito Repellent Room
Spray", "brand": "Herbal Strategi", "price": 200, "rating": 4.2},

    {"index": 16, "product_name": "Dove Plastic Soap Case - Assorted
Colour", "brand": "Nakoda", "price": 49, "rating": 4},

    {"index": 17, "product_name": "Smooth Skin Oil - For Dry Skin",
"brand": "Aroma Treasures", "price": 324, "rating": 5},

    {"index": 18, "product_name": "Salted Pumpkin", "brand": "Graminway",
"price": 180, "rating": 4.9},

    {"index": 19, "product_name": "Flax Seeds - Roasted", "brand": "True
Elements", "price": 120, "rating": 4.3},

    {"index": 20, "product_name": "Organic Tofu - Soy Paneer", "brand":
"Murginns", "price": 85.14, "rating": 3.9},

    {"index": 21, "product_name": "Ceramic Barrel Brush - Colour May
Vary", "brand": "Bronson Professional", "price": 525, "rating": 4.2},

    {"index": 22, "product_name": "Instant Noodles - Chicken Satay
Flavor", "brand": "Koka", "price": 45, "rating": 4.1},

    {"index": 23, "product_name": "Chia Seeds", "brand": "NaturoBell",
"price": 120, "rating": 3.9},

    {"index": 24, "product_name": "Cleanse Green Tea - Whole Leaf Loose
Tea", "brand": "Cambridge Tea Party", "price": 75, "rating": 3.9},

    {"index": 25, "product_name": "Veggie Cutter", "brand": "IRICH",
"price": 195, "rating": 5},

    {"index": 26, "product_name": "Insulated Hot Fresh Casserole For
Roti/Chapati - White", "brand": "Cello", "price": 659, "rating": 3.3},

    {"index": 27, "product_name": "Granola - Happy Berries", "brand": "Fit
& Flex", "price": 245, "rating": 3.5},

    {"index": 28, "product_name": "Flaxseed - Pesticide Free", "brand":
```

```
"Safe Harvest", "price": 53.9, "rating": 4},

    {"index": 29, "product_name": "Paratha Puff", "brand": "Switz",
"price": 90, "rating": 4.3},

    {"index": 30, "product_name": "Lip Butter - Rose", "brand": "Organic
Harvest", "price": 169.15, "rating": 1.5},

    {"index": 31, "product_name": "Fruit Power - Masala Sugarcane",
"brand": "Real", "price": 19, "rating": 2.9},

    {"index": 32, "product_name": "Chocobakes Choc Filled Cookies",
"brand": "Cadbury", "price": 102, "rating": 4.2},

    {"index": 33, "product_name": "Amber - Deodorant Body Spray", "brand":
"Old Spice", "price": 211.65, "rating": 3.4},

    {"index": 34, "product_name": "Green Tea - Tulsi Loose Leaf", "brand":
"Octavius", "price": 225, "rating": 3.7},

    {"index": 35, "product_name": "Pet Solitaire Container Set - Silver",
"brand": "Steelo", "price": 499, "rating": 3.9},

    {"index": 36, "product_name": "Dhania - Dal", "brand": "bb Royal",
"price": 98, "rating": 4.1},

    {"index": 37, "product_name": "Pudina Chutney Masala", "brand":
"Catch", "price": 46.75, "rating": 3.9},

    {"index": 38, "product_name": "Bodylicious Deodorant Spray - Mate (For
Men)", "brand": "Engage", "price": 136.5, "rating": 4.1},

    {"index": 39, "product_name": "Sport Deo Spray - Fresh, for Men",
"brand": "Engage", "price": 112.75, "rating": 3.7},

    {"index": 40, "product_name": "Choco Deck - Mini Delights", "brand":
"Fabelle", "price": 160, "rating": 4.4},

    {"index": 41, "product_name": "Eau De Toilette - Homme Green",
"brand": "Colour Me", "price": 427.5, "rating": 4.3},

    {"index": 42, "product_name": "Lemon & Tea Tree Oil Soap", "brand":
"Liril", "price": 360, "rating": 4.2},
    {"index": 43, "product_name": "Flavoured Cream Wafer Roll -
Strawberry", "brand": "Twister", "price": 275, "rating": 4.3},

    {"index": 44, "product_name": "Storage/Lunch Steel Container with PP
Lid - Red", "brand": "Classic Essentials", "price": 109, "rating": 2.6},

    {"index": 45, "product_name": "Plain Green Olives", "brand": "Figaro",
"price": 179, "rating": 5},

    {"index": 46, "product_name": "Quinoa - Organic", "brand": "Organic
Tattva", "price": 250, "rating": 3.9},

    {"index": 47, "product_name": "After Shave Splash - Arctic Ice",
"brand": "Gillette", "price": 459.62, "rating": 3.9},

    {"index": 48, "product_name": "Colour Catcher Sheets", "brand": "Dr.
Beckmann", "price": 160, "rating": 4.6},

    {"index": 49, "product_name": "Premium Walnut Kernels - Snow White",
"brand": "Vedaka", "price": 799, "rating": 4.3},

    {"index": 50, "product_name": "Green Coffee Beans", "brand":
"Neuherbs", "price": 189, "rating": 3.5},

    {"index": 51, "product_name": "Cold Pressed Extra Virgin Coconut Oil",
"brand": "Neuherbs", "price": 329, "rating": 4.4},

    {"index": 52, "product_name": "Karela Juice - Neem", "brand":
"Neuherbs", "price": 199, "rating": 4.1},

    {"index": 53, "product_name": "Sunflower Oil", "brand": "Fortune",
"price": 105, "rating": 4.2},

    {"index": 54, "product_name": "Lemongrass Body Wash", "brand":
"Cinthol", "price": 200, "rating": 3.8},

    {"index": 55, "product_name": "Neem Soap", "brand": "Medimix",
"price": 37, "rating": 4},

    {"index": 56, "product_name": "Calcium Tablets", "brand": "Himalaya",
"price": 150, "rating": 3.8},

    {"index": 57, "product_name": "Face Mask - Aloe Vera", "brand":
```

```
"Patanjali", "price": 60, "rating": 4.2},

    {"index": 58, "product_name": "Shampoo - Anti Dandruff", "brand":
"Dove", "price": 150, "rating": 4},

    {"index": 59, "product_name": "Aloe Vera Gel", "brand": "Khadi",
"price": 120, "rating": 4.5},

    {"index": 60, "product_name": "Toothpaste - Mint", "brand": "Colgate",
"price": 50, "rating": 4.3},

    {"index": 61, "product_name": "Shaving Foam", "brand": "Gillette",
"price": 199, "rating": 3.7},

    {"index": 62, "product_name": "Deodorant - Musk", "brand": "Park
Avenue", "price": 150, "rating": 4.1},

    {"index": 63, "product_name": "Lip Balm - Strawberry", "brand":
"Nivea", "price": 100, "rating": 4.3},

    {"index": 64, "product_name": "Soap - Lavender", "brand": "Dove",
"price": 75, "rating": 3.9},

    {"index": 65, "product_name": "Body Lotion - Almond", "brand":
"Vaseline", "price": 120, "rating": 4},

    {"index": 66, "product_name": "Coconut Oil", "brand": "Parachute",
"price": 60, "rating": 4.2},

    {"index": 67, "product_name": "Toothbrush", "brand": "Oral-B",
"price": 30, "rating": 3.8},

    {"index": 68, "product_name": "Mouthwash - Mint", "brand":
"Listerine", "price": 120, "rating": 4.1},

    {"index": 69, "product_name": "Sanitizer", "brand": "Dettol", "price":
50, "rating": 4.3},

    {"index": 70, "product_name": "Face Cream - Anti Aging", "brand":
"Olay", "price": 299, "rating": 3.7}

]

customers = {
    "Alice": {"purchases": [1, 3, 5], "dues": 0},

    "Bob": {"purchases": [2, 4], "dues": 50},

    "Charlie": {"purchases": [6, 7, 8], "dues": 100},

    "David": {"purchases": [3, 5], "dues": 0},

    "Eve": {"purchases": [1, 6], "dues": 150},

    "Frank": {"purchases": [4, 7], "dues": 0},

    "Grace": {"purchases": [2, 9], "dues": 20},

    "Hannah": {"purchases": [5, 8], "dues": 70},

    "Ian": {"purchases": [1, 3, 9], "dues": 0},

    "Jack": {"purchases": [2, 4, 6], "dues": 50},

    "Kathy": {"purchases": [7, 8], "dues": 10},

    "Liam": {"purchases": [1, 2, 3], "dues": 0},

    "Mona": {"purchases": [4, 5], "dues": 60},

    "Nina": {"purchases": [7, 9], "dues": 0},

    "Oscar": {"purchases": [6, 8], "dues": 30},

    "Paul": {"purchases": [1, 4, 5], "dues": 0},

    "Quincy": {"purchases": [2, 6], "dues": 90},

    "Rachel": {"purchases": [3, 8], "dues": 0},

    "Steve": {"purchases": [5, 7, 9], "dues": 120},

    "Tina": {"purchases": [1, 2, 8], "dues": 0},

    "Uma": {"purchases": [4, 6], "dues": 40},

    "Vince": {"purchases": [7, 9], "dues": 0},

    "Wendy": {"purchases": [2, 5], "dues": 30},
    "Xander": {"purchases": [1, 6, 7], "dues": 0},

    "Yara": {"purchases": [3, 4, 9], "dues": 50},

    "Zane": {"purchases": [2, 8], "dues": 0},

    "Adam": {"purchases": [1, 5], "dues": 0},

    "Bella": {"purchases": [2, 7], "dues": 100},

    "Cindy": {"purchases": [3, 6, 9], "dues": 0},

    "Derek": {"purchases": [4, 5], "dues": 60},

    "Ella": {"purchases": [1, 7, 8], "dues": 0},

    "Felix": {"purchases": [2, 3], "dues": 30},

    "Gina": {"purchases": [4, 6, 9], "dues": 0},



}



products = {item['index']: item for item in product_data}



def display_product(product):

    stock_info = f"Stock: {product.get('stock', 'Not Available')}"

    sold_info = product.get('sold', 'Not Available')

    print(f"Index: {product['index']}, Name: {product['product_name']},
Brand: {product['brand']}, Price: {product['price']}, Rating:
{product['rating']}, {stock_info}, Sold: {sold_info}")



def display_products(products):

    for product in products.values():
        display_product(product)


def total_products_sold():

    total_sold = sum(product.get('sold', 0) for product in
products.values())

    print(f"Total products sold: {total_sold}")


def most_expensive_product_sold():

    most_expensive = max(products.values(), key=lambda x: x.get('price',
0))

    print("Most expensive product sold:")

    display_product(most_expensive)


def least_expensive_product_sold():

    least_expensive = min(products.values(), key=lambda x: x.get('price',
float('inf')))

    least_expensive_products = [product for product in products.values()
if product['price'] == least_expensive['price']]

    print("Least expensive products sold:")

    for product in least_expensive_products:

        display_product(product)


def products_left_in_stock():

    stock_left = sum(product.get('stock', 0) for product in
products.values())
```

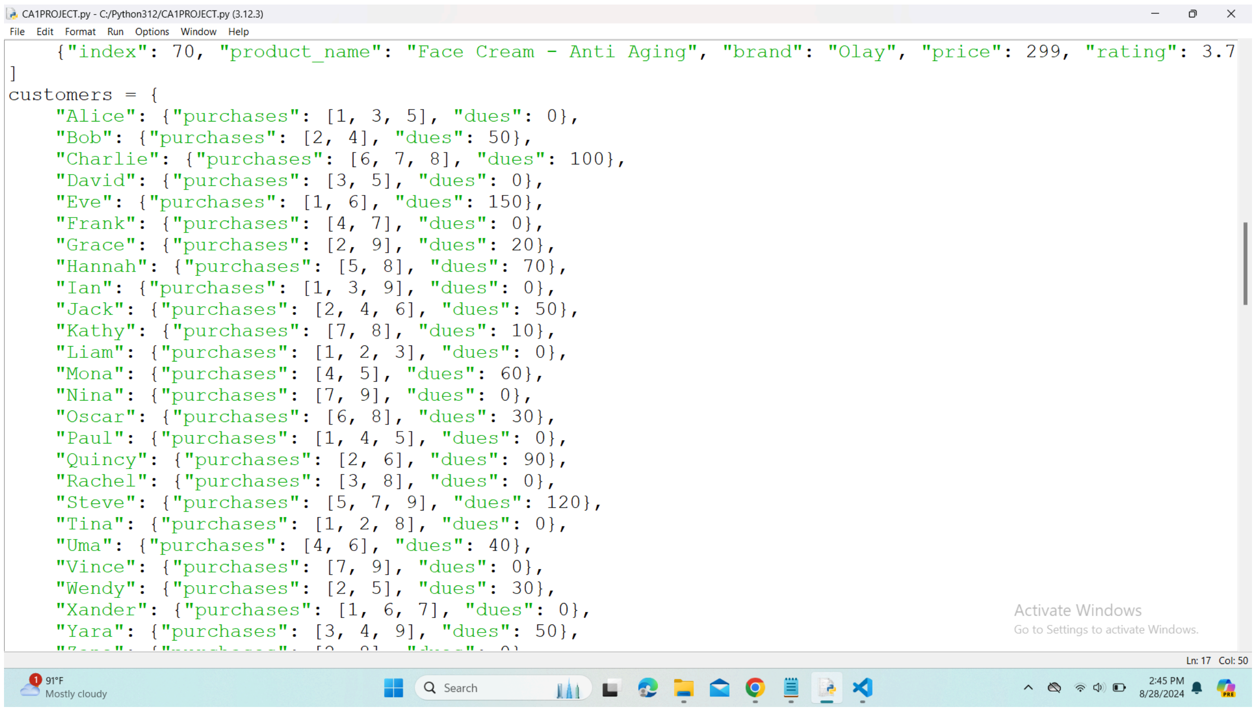
```python
    print(f"Products left in stock: {stock_left}")


def total_sales_amount():
    total_sales = sum(product.get('price', 0) * product.get('sold', 0) for product in products.values())
    print(f"Total sales amount: {total_sales}")


def highest_rated_product():
    highest_rated = max(products.values(), key=lambda x: x.get('rating', 0))
    highest_rated_products = [product for product in products.values() if product['rating'] == highest_rated['rating']]
    print("Highest rated products:")
    for product in highest_rated_products:
        display_product(product)


def lowest_rated_product():
    lowest_rated = min(products.values(), key=lambda x: x.get('rating', float('inf')))
    lowest_rated_products = [product for product in products.values() if product['rating'] == lowest_rated['rating']]
    print("Lowest rated products:")
    for product in lowest_rated_products:
        display_product(product)
```

```python
def total_customers_with_pending_dues():

    total_customers_with_pending_dues = sum(1 for customer in
customers.values() if customer['dues'] > 0)

    print(f"Total customers with pending dues:
{total_customers_with_pending_dues}")


def products_purchased_by_customer(customer_name):

    if customer_name in customers:

        print(f"Products purchased by {customer_name}:")

        for index in customers[customer_name]['purchases']:

            if index in products:

                display_product(products[index])

            else:

                print(f"Product with index {index} not found.")

    else:

        print("Customer not found.")


def display_least_rated_product():

    lowest_rated = min(products.values(), key=lambda x: x.get('rating',
float('inf')))

    print("Least rated products:")

    for product in products.values():

        if product['rating'] == lowest_rated['rating']:

            display_product(product)
```

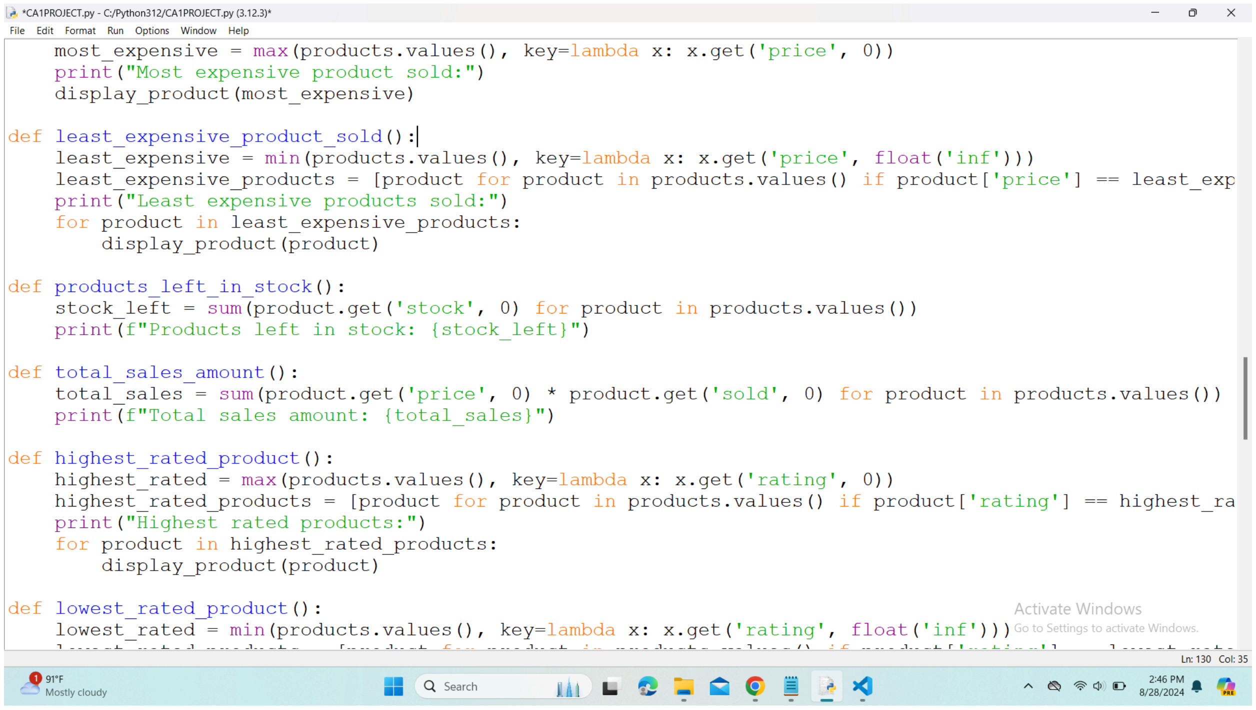
```python
def display_most_rated_product():
    most_rated = max(products.values(), key=lambda x: x.get('rating', 0))
    print("Most rated products:")
    for product in products.values():
        if product['rating'] == most_rated['rating']:
            display_product(product)


while True:
    print("\n1. Total products sold")
    print("2. Most expensive product sold")
    print("3. Least expensive product sold")
    print("4. Products left in stock")
    print("5. Total sales amount")
    print("6. Total customers with pending dues")
    print("7. Products purchased by a specific customer")
    print("8. Customers with dues greater than a specified amount")
    print("9. Display best rated products")
    print("10. Exit")

    try:
        choice = int(input("Enter your choice: "))
    except ValueError:
        print("Please enter a valid number.")

        continue


    if choice == 1:

        total_products_sold()

    elif choice == 2:

        most_expensive_product_sold()

    elif choice == 3:

        least_expensive_product_sold()

    elif choice == 4:

        products_left_in_stock()

    elif choice == 5:

        total_sales_amount()

    elif choice == 6:

        total_customers_with_pending_dues()

    elif choice == 7:

        customer_name = input("Enter the customer's name: ")

        products_purchased_by_customer(customer_name)

    elif choice == 8:

        dues_threshold = int(input("Enter the dues threshold: "))

        customers_with_high_dues = [customer for customer in
customers.values() if customer['dues'] > dues_threshold]

        print("Customers with dues greater than", dues_threshold, ":")

        for customer in customers_with_high_dues:
```

```python
            print(customer)

    elif choice == 9:

        highest_rated_product()

    elif choice == 10:

        print("Exiting the program. Goodbye!")

        break

    else:

        print("Invalid choice, please try again.")
```

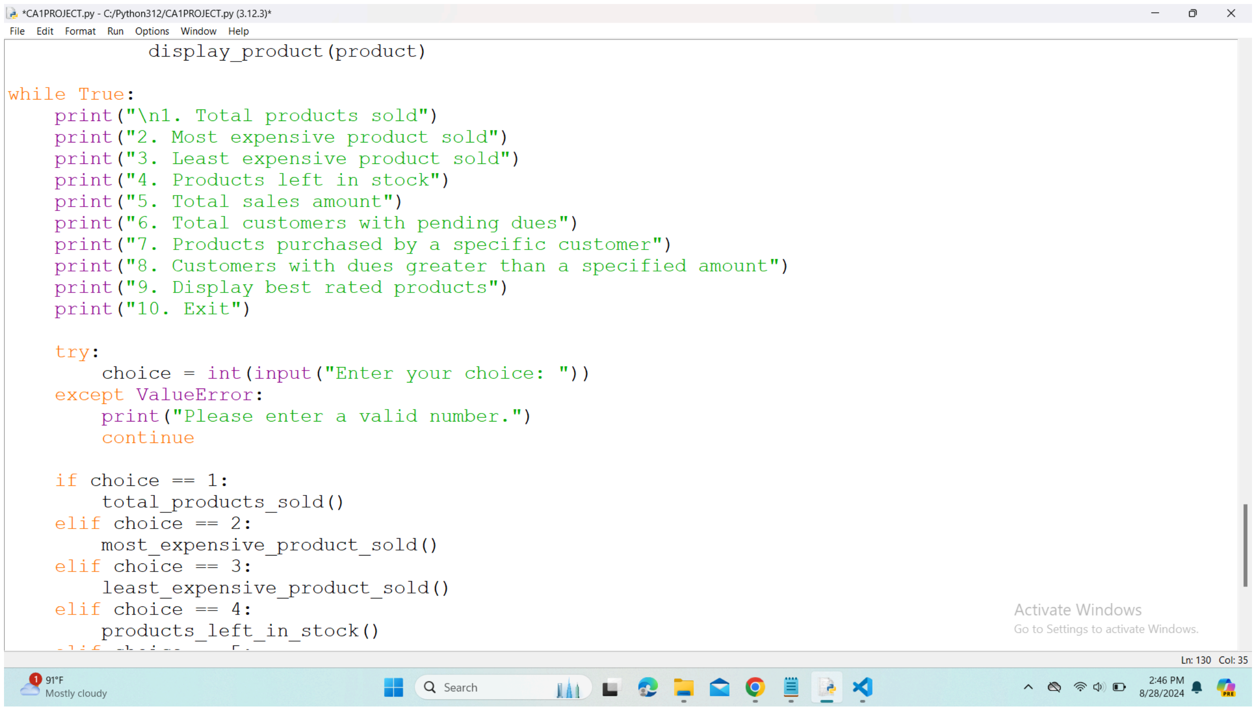
Screenshot of the output:


CA1PROJECT.py - C:/Python312/CA1PROJECT.py (3.12.3)
File Edit Format Run Options Window Help
Ln: 17 Col: 50
Activate Windows
Go to Settings to activate Windows.

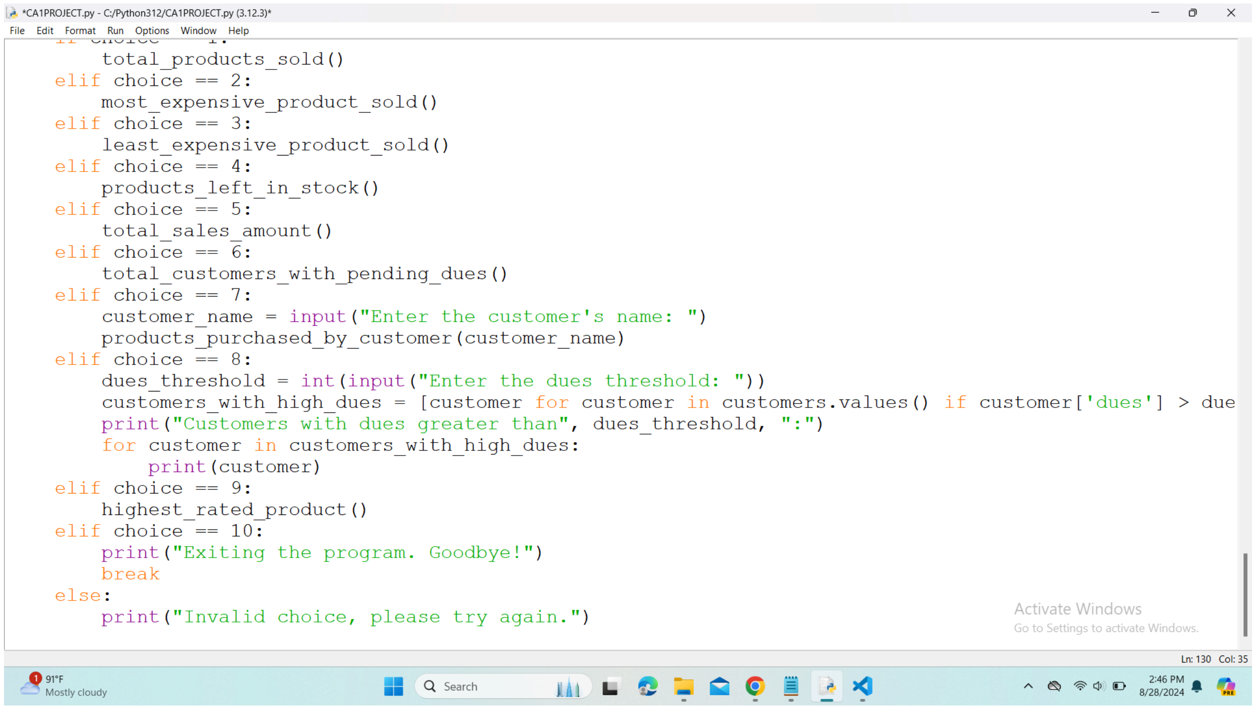
```python
    {"index": 70, "product_name": "Face Cream - Anti Aging", "brand": "Olay", "price": 299, "rating": 3.7
]
customers = {
    "Alice": {"purchases": [1, 3, 5], "dues": 0},
    "Bob": {"purchases": [2, 4], "dues": 50},
    "Charlie": {"purchases": [6, 7, 8], "dues": 100},
    "David": {"purchases": [3, 5], "dues": 0},
    "Eve": {"purchases": [1, 6], "dues": 150},
    "Frank": {"purchases": [4, 7], "dues": 0},
    "Grace": {"purchases": [2, 9], "dues": 20},
    "Hannah": {"purchases": [5, 8], "dues": 70},
    "Ian": {"purchases": [1, 3, 9], "dues": 0},
    "Jack": {"purchases": [2, 4, 6], "dues": 50},
    "Kathy": {"purchases": [7, 8], "dues": 10},
    "Liam": {"purchases": [1, 2, 3], "dues": 0},
    "Mona": {"purchases": [4, 5], "dues": 60},
    "Nina": {"purchases": [7, 9], "dues": 0},
    "Oscar": {"purchases": [6, 8], "dues": 30},
    "Paul": {"purchases": [1, 4, 5], "dues": 0},
    "Quincy": {"purchases": [2, 6], "dues": 90},
    "Rachel": {"purchases": [3, 8], "dues": 0},
    "Steve": {"purchases": [5, 7, 9], "dues": 120},
    "Tina": {"purchases": [1, 2, 8], "dues": 0},
    "Uma": {"purchases": [4, 6], "dues": 40},
    "Vince": {"purchases": [7, 9], "dues": 0},
    "Wendy": {"purchases": [2, 5], "dues": 30},
    "Xander": {"purchases": [1, 6, 7], "dues": 0},
    "Yara": {"purchases": [3, 4, 9], "dues": 50},
    "Zara": {"purchases": [2, 8], "dues": 0},
```


CA1PROJECT.py - C:/Python312/CA1PROJECT.py (3.12.3)
File Edit Format Run Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 130 Col: 35
91°F
Mostly cloudy
Search
2:46 PM
8/28/2024

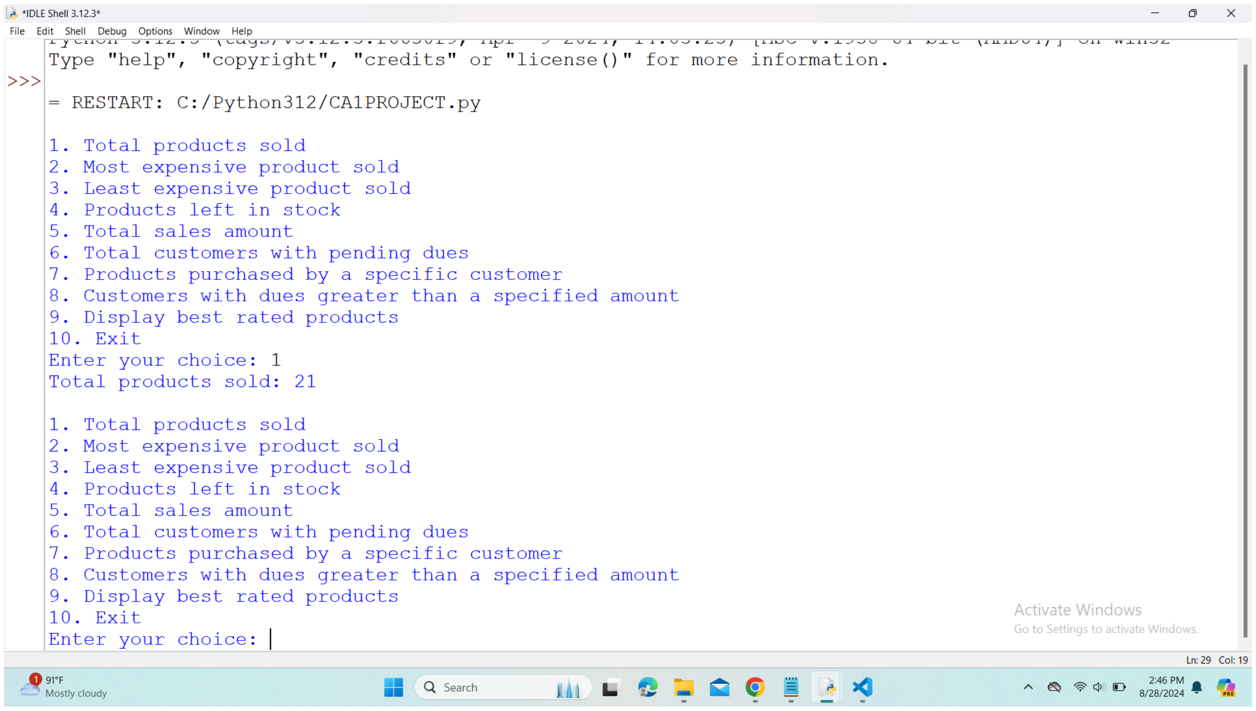
```
    most_expensive = max(products.values(), key=lambda x: x.get('price', 0))
    print("Most expensive product sold:")
    display_product(most_expensive)

def least_expensive_product_sold():
    least_expensive = min(products.values(), key=lambda x: x.get('price', float('inf')))
    least_expensive_products = [product for product in products.values() if product['price'] == least_exp
    print("Least expensive products sold:")
    for product in least_expensive_products:
        display_product(product)

def products_left_in_stock():
    stock_left = sum(product.get('stock', 0) for product in products.values())
    print(f"Products left in stock: {stock_left}")

def total_sales_amount():
    total_sales = sum(product.get('price', 0) * product.get('sold', 0) for product in products.values())
    print(f"Total sales amount: {total_sales}")

def highest_rated_product():
    highest_rated = max(products.values(), key=lambda x: x.get('rating', 0))
    highest_rated_products = [product for product in products.values() if product['rating'] == highest_ra
    print("Highest rated products:")
    for product in highest_rated_products:
        display_product(product)

def lowest_rated_product():
    lowest_rated = min(products.values(), key=lambda x: x.get('rating', float('inf')))
```


CA1PROJECT.py - C:/Python312/CA1PROJECT.py (3.12.3)
File Edit Format Run Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 130 Col: 35
91°F
Mostly cloudy
Search
2:46 PM
8/28/2024

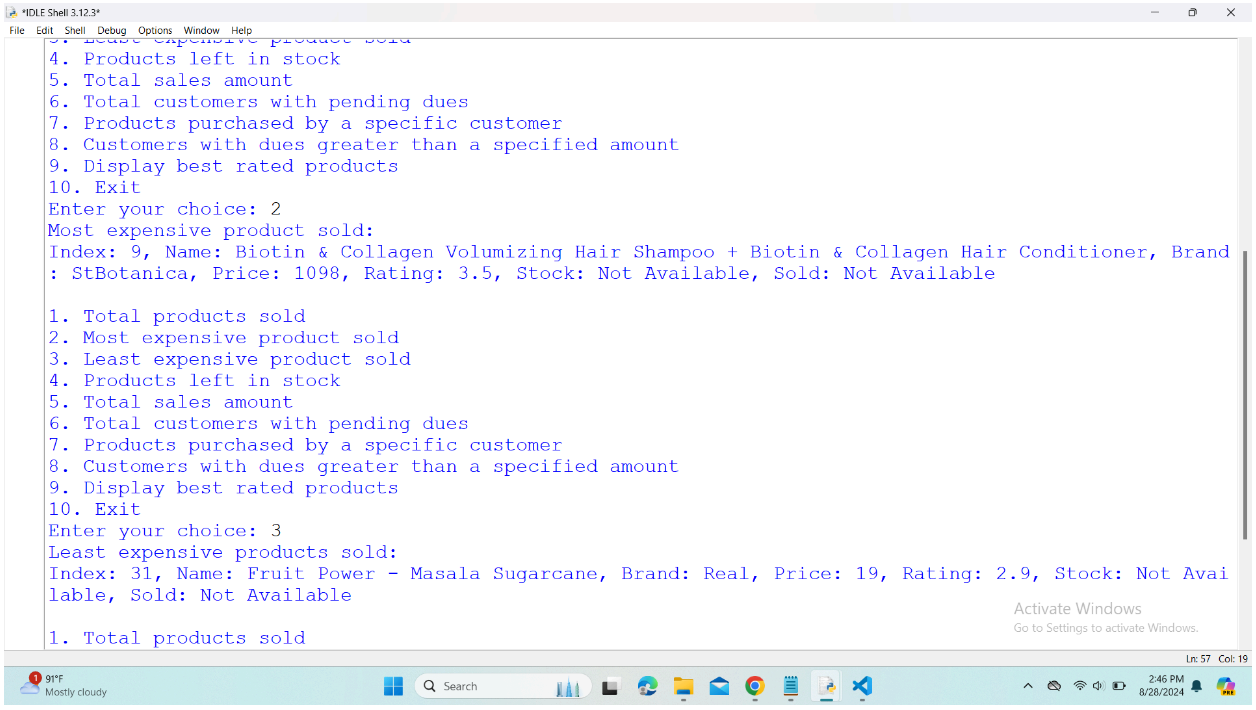
```
            display_product(product)

while True:
    print("\n1. Total products sold")
    print("2. Most expensive product sold")
    print("3. Least expensive product sold")
    print("4. Products left in stock")
    print("5. Total sales amount")
    print("6. Total customers with pending dues")
    print("7. Products purchased by a specific customer")
    print("8. Customers with dues greater than a specified amount")
    print("9. Display best rated products")
    print("10. Exit")

    try:
        choice = int(input("Enter your choice: "))
    except ValueError:
        print("Please enter a valid number.")
        continue

    if choice == 1:
        total_products_sold()
    elif choice == 2:
        most_expensive_product_sold()
    elif choice == 3:
        least_expensive_product_sold()
    elif choice == 4:
        products_left_in_stock()
```

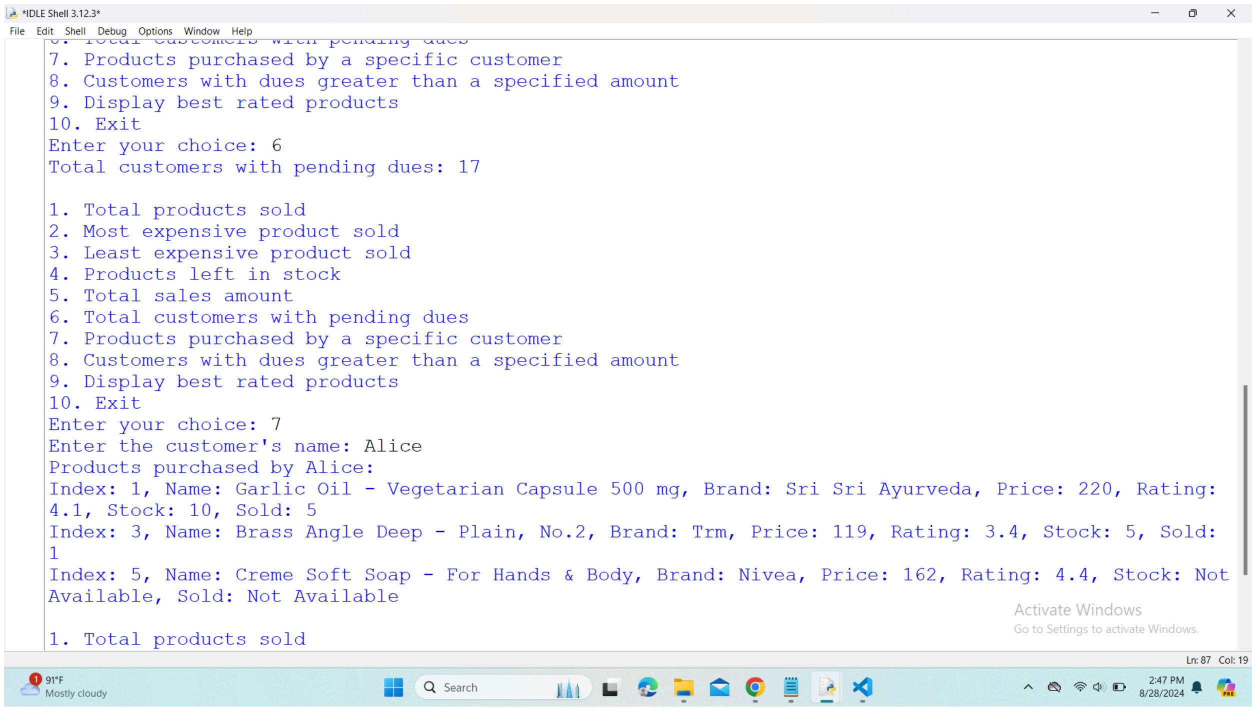

CA1PROJECT.py - C:/Python312/CA1PROJECT.py (3.12.3)
File Edit Format Run Options Window Help
Ln: 130 Col: 35
Activate Windows
Go to Settings to activate Windows.
91°F
Mostly cloudy
Search
2:46 PM
8/28/2024

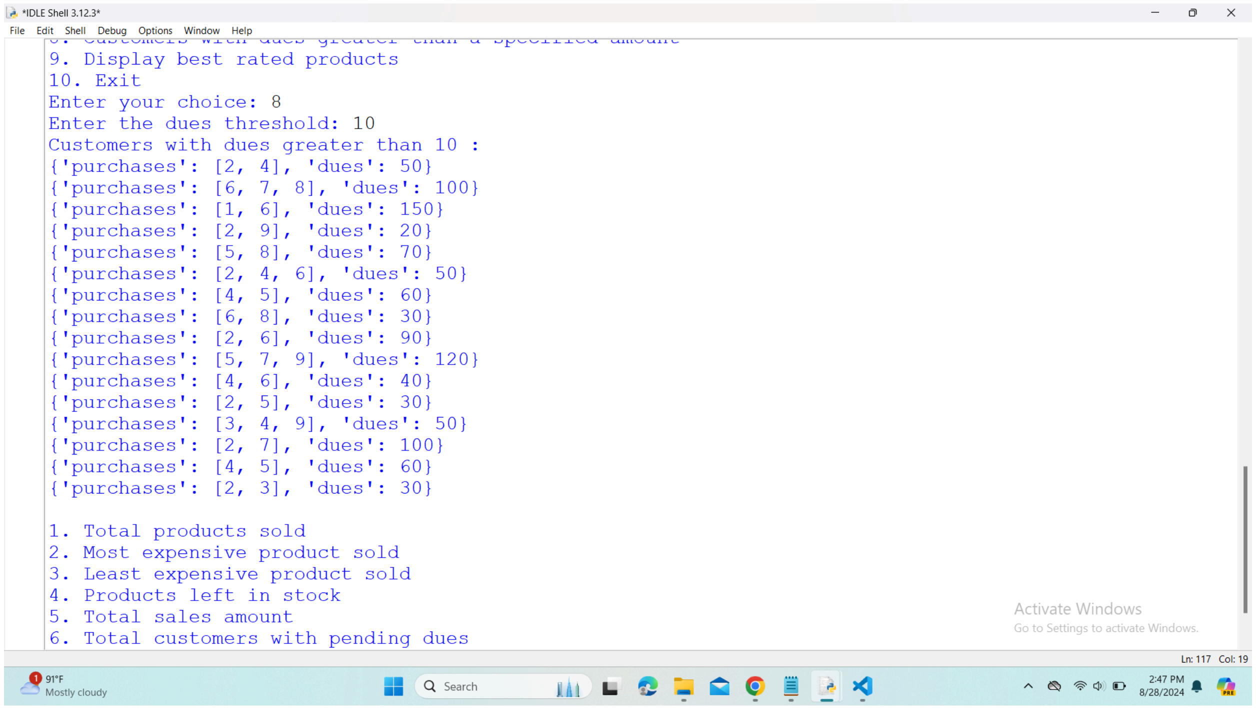
```
        total_products_sold()
    elif choice == 2:
        most_expensive_product_sold()
    elif choice == 3:
        least_expensive_product_sold()
    elif choice == 4:
        products_left_in_stock()
    elif choice == 5:
        total_sales_amount()
    elif choice == 6:
        total_customers_with_pending_dues()
    elif choice == 7:
        customer_name = input("Enter the customer's name: ")
        products_purchased_by_customer(customer_name)
    elif choice == 8:
        dues_threshold = int(input("Enter the dues threshold: "))
        customers_with_high_dues = [customer for customer in customers.values() if customer['dues'] > due
        print("Customers with dues greater than", dues_threshold, ":")
        for customer in customers_with_high_dues:
            print(customer)
    elif choice == 9:
        highest_rated_product()
    elif choice == 10:
        print("Exiting the program. Goodbye!")
        break
    else:
        print("Invalid choice, please try again.")
```



```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Python312/CA1PROJECT.py

1. Total products sold
2. Most expensive product sold
3. Least expensive product sold
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 1
Total products sold: 21

1. Total products sold
2. Most expensive product sold
3. Least expensive product sold
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 
```


```
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 2
Most expensive product sold:
Index: 9, Name: Biotin & Collagen Volumizing Hair Shampoo + Biotin & Collagen Hair Conditioner, Brand
: StBotanica, Price: 1098, Rating: 3.5, Stock: Not Available, Sold: Not Available

1. Total products sold
2. Most expensive product sold
3. Least expensive product sold
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 3
Least expensive products sold:
Index: 31, Name: Fruit Power - Masala Sugarcane, Brand: Real, Price: 19, Rating: 2.9, Stock: Not Avai
lable, Sold: Not Available

1. Total products sold
```


```
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 6
Total customers with pending dues: 17

1. Total products sold
2. Most expensive product sold
3. Least expensive product sold
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
7. Products purchased by a specific customer
8. Customers with dues greater than a specified amount
9. Display best rated products
10. Exit
Enter your choice: 7
Enter the customer's name: Alice
Products purchased by Alice:
Index: 1, Name: Garlic Oil - Vegetarian Capsule 500 mg, Brand: Sri Sri Ayurveda, Price: 220, Rating:
4.1, Stock: 10, Sold: 5
Index: 3, Name: Brass Angle Deep - Plain, No.2, Brand: Trm, Price: 119, Rating: 3.4, Stock: 5, Sold:
1
Index: 5, Name: Creme Soft Soap - For Hands & Body, Brand: Nivea, Price: 162, Rating: 4.4, Stock: Not
Available, Sold: Not Available

1. Total products sold
```



```
9. Display best rated products
10. Exit
Enter your choice: 8
Enter the dues threshold: 10
Customers with dues greater than 10 :
{'purchases': [2, 4], 'dues': 50}
{'purchases': [6, 7, 8], 'dues': 100}
{'purchases': [1, 6], 'dues': 150}
{'purchases': [2, 9], 'dues': 20}
{'purchases': [5, 8], 'dues': 70}
{'purchases': [2, 4, 6], 'dues': 50}
{'purchases': [4, 5], 'dues': 60}
{'purchases': [6, 8], 'dues': 30}
{'purchases': [2, 6], 'dues': 90}
{'purchases': [5, 7, 9], 'dues': 120}
{'purchases': [4, 6], 'dues': 40}
{'purchases': [2, 5], 'dues': 30}
{'purchases': [3, 4, 9], 'dues': 50}
{'purchases': [2, 7], 'dues': 100}
{'purchases': [4, 5], 'dues': 60}
{'purchases': [2, 3], 'dues': 30}

1. Total products sold
2. Most expensive product sold
3. Least expensive product sold
4. Products left in stock
5. Total sales amount
6. Total customers with pending dues
```

Future Works

File Handling:

To enhance the system's effectiveness and durability, file handling can be integrated to store and retrieve ledger data. By saving transaction records as files, shopkeepers can maintain a historical record of their business activities, preventing data loss due to unexpected system behavior. This feature also enables backup and recovery capabilities, ensuring data security and continuity.

Data Analysis with Pandas and NumPy:

Leveraging powerful data analysis libraries like Pandas and NumPy can significantly enhance the system's capabilities. Shopkeepers can utilize these tools to perform various analytical tasks, such as:

- **Tracking trends in customer purchases:** Identify patterns in customer behavior and preferences.
- **Calculating average spend per customer:** Analyze customer spending habits and target high-value customers.
- **Generating sales reports:** Create detailed reports on product performance, sales revenue, and customer demographics.

By incorporating data analysis capabilities, shopkeepers can gain valuable insights into their business operations, leading to improved inventory management, customer retention strategies, and overall business growth.

Data Visualization with Matplotlib and Seaborn:

To effectively communicate insights derived from data analysis, incorporating data visualization libraries like Matplotlib and Seaborn can be highly beneficial. These libraries enable the creation of informative and visually appealing charts and graphs, such as:

- **Sales trend analysis:** Visualize changes in sales over time.
- **Customer segmentation:** Analyze customer demographics and spending patterns.
- **Product performance:** Compare the popularity and profitability of different products.

By visualizing data, shopkeepers can gain a deeper understanding of their business performance and make data-driven decisions.