

Assignment 1

TechShop, an electronic gadgets shop

Task 1: Classes and Their Attributes:

Task 2: Class Creation:

- Create the classes (Customers, Products, Orders, OrderDetails and Inventory) with the specified attributes.
- Implement the constructor for each class to initialize its attributes.
- Implement methods as specified.

Customers Class:

Attributes:

- CustomerID (int)
- FirstName (string)
- LastName (string)
- Email (string)
- Phone (string)
- Address (string)

Methods:

- CalculateTotalOrders(): Calculates the total number of orders placed by this customer.
- GetCustomerDetails(): Retrieves and displays detailed information about the customer.
- UpdateCustomerInfo(): Allows the customer to update their information (e.g., email, phone, or address).

```

1  from Exception import InvalidDataException
2
3  class Customers:
4      def __init__(self, CustomerId, FirstName, LastName, Email, Phone, Address):
5          self._CustomerId=CustomerId
6          self._FirstName=FirstName
7          self._LastName=LastName
8          self._Email=Email
9          self._Phone=Phone
10         self._Address=Address
11
12     def __init__(self, db_connector):
13         self._db_connector = db_connector
14
15     @property
16     def CustomerId(self):
17         return self._CustomerId
18
19     @CustomerId.setter
20     def CustomerId(self, new_customer_id):
21         self._CustomerId = new_customer_id
22
23     @property
24     def FirstName(self):
25         return self._FirstName
26
27     @FirstName.setter
28     def FirstName(self, new_FirstName):
29         self._FirstName=new_FirstName
30
31     @property
32     def LastName(self):
33         return self._LastName

```

```

34
35     @Email.setter
36     def Email(self, new_Email):
37         if "@" in new_Email and "." in new_Email:
38             self._Email =new_Email
39         else:
40             raise InvalidDataException("Invalid email format.")
41
42     @property
43     def Phone(self):
44         return self._Phone
45
46     @Phone.setter
47     def Phone(self, new_Phone):
48         if len(new_Phone) == 10 and new_Phone.isdigit():
49             self._Phone = new_Phone
50         else:
51             raise InvalidDataException("Invalid phone number format.")
52
53     @property
54     def Address(self):
55         return self._Address
56
57     @Address.setter
58     def Address(self, new_Address):
59         if isinstance(new_Address, str):
60             self._Address = new_Address
61         else:
62             raise InvalidDataException("Invalid address format")
63
64     def create_customer(self, CustomerId, FirstName, LastName, Email, Phone, Address):
65         try:
66             self._db_connector.open_connection()

```

```

64 def create_customer(self, CustomerId, FirstName, LastName, Email, Phone, Address):
65     try:
66         self._db_connector.open_connection()
67         cursor = self._db_connector.connection.cursor()
68
69         cursor.execute("SELECT * FROM customers WHERE Email = %s", (Email,))
70         existing_customer = cursor.fetchone()
71
72         if existing_customer:
73             print("Error: Email address is already in use.")
74         else:
75             cursor.execute("INSERT INTO customers (CustomerId, FirstName, LastName, Email, Phone, Address) VALUES (%s, %s, %s, %s, %s, %s)"
76                             % (CustomerId, FirstName, LastName, Email, Phone, Address))
77             self._db_connector.connection.commit()
78             print("Customer created successfully")
79
80     except Exception as e:
81         print(f"Error: {e}")
82
83     finally:
84         if cursor:
85             cursor.close()
86         self._db_connector.close_connection()
87
88 def calculate_Total_Orders(self, Customerid):
89     try:
90         self._db_connector.open_connection()
91         query = """ SELECT COUNT(OrderID) AS TotalOrders FROM Orders
92                     WHERE Customerid = %s"""
93         values = (Customerid,)
94         self._db_connector.cursor.execute(query, values)
95         result = self._db_connector.cursor.fetchone()

```

```

114         self._db_connector.cursor.execute(query, values)
115
116         customer_details = self._db_connector.cursor.fetchone()
117
118         if customer_details:
119             print("Customer Details:")
120             print(f"Customer ID: {customer_details[0]}")
121             print(f"First Name: {customer_details[1]}")
122             print(f>Last Name: {customer_details[2]}")
123             print(f>Email: {customer_details[3]}")
124             print(f>Phone: {customer_details[4]}")
125             print(f>Address: {customer_details[5]}")
126         else:
127             print("Customer not found.")
128
129     except Exception as e:
130         print(f"Error getting customer details: {e}")
131
132     finally:
133         self._db_connector.close_connection()
134
135
136 def update_customer_Info(self, Customerid, new_Email, new_Phone, new_Address):
137     try:
138         self._db_connector.open_connection()
139
140         query = "UPDATE customers SET Email=%s, Phone=%s, Address=%s WHERE Customerid=%s"
141         values = (new_Email, new_Phone, new_Address, Customerid)
142

```

```

86     def calculate_Total_Orders(self, Customerid):
87         try:
88             self._db_connector.open_connection()
89             query = """ SELECT COUNT(OrderID) AS TotalOrders FROM Orders
90                         WHERE Customerid = %s"""
91             values = (Customerid,)
92             self._db_connector.cursor.execute(query, values)
93             result = self._db_connector.cursor.fetchone()
94
95             if result:
96                 total_orders = result[0]
97                 print(f"Total Orders for Customer {Customerid}: {total_orders}")
98             else:
99                 print("No orders found for this customer")
100
101         except Exception as e:
102             print(f"Error calculating total orders: {e}")
103         finally:
104             self._db_connector.close_connection()
105
106
107     def get_Customer_Details(self, Customerid):
108         try:
109             self._db_connector.open_connection()
110
111             query = "SELECT * FROM customers WHERE CustomerId=%s"
112             values = (Customerid,)
113
114             self._db_connector.cursor.execute(query, values)

```

```

1  from DatabaseConnector import DatabaseConnector
2  from _Customers import Customers
3  from Products import Products
4  from Orders import Orders
5  from OrderDetails import OrderDetails
6  from Inventory import Inventory
7
8  db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
9  db_connector.open_connection()
10 customers_manager = Customers(db_connector)
11
12
13 #customers_manager.create_customer("1001", "Ajay", "Chaudhary", "ajaychaudhary123@gmail.com", "123456789", "Lucknow")
14
15 customers_manager.update_customer_Info("101", "chauhan@gmail.com", "123456789", "Noida")
16
17
18

```

Run: Main

```

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Customer account updated successfully
Connection closed
Process finished with exit code 0

```

```
assignment1 ~/PycharmProjects/assignment1
venv
Customers.py
DatabaseConnector.py
Exception.py
Inventory.py
Main.py
OrderDetails.py
Orders.py
Products.py
External Libraries
Scratches and Consoles

1 from DatabaseConnector import DatabaseConnector
2 from Customers import Customers
3 from Products import Products
4 from Orders import Orders
5 from OrderDetails import OrderDetails
6 from Inventory import Inventory
7
8 db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
9 db_connector.open_connection()
10 customers_manager = Customers(db_connector)
11
12
13 customers_manager.create_customer("1001", "Ajay", "Chaudhary", "ajaychaudhary123@gmail.com", "123458759", "Lucknow")
14
15
16
17
18
```

Run: Main

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py

Connected to MySQL database
Connected to MySQL database
Customer created successfully
Connection closed

```
assignment1 ~/PycharmPre
venv
Customers.py
DatabaseConnector.py
Exception.py
Inventory.py
Main.py
OrderDetails.py
Orders.py
Products.py
External Libraries
Scratches and Consoles

1 from DatabaseConnector import DatabaseConnector
2 from Customers import Customers
3 from Products import Products
4 from Orders import Orders
5 from OrderDetails import OrderDetails
6 from Inventory import Inventory
7
8 db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
9 db_connector.open_connection()
10 customers_manager = Customers(db_connector)
11
12
13 #customers_manager.create_customer("1001", "Ajay", "Chaudhary", "ajaychaudhary123@gmail.com", "123458759", "Lucknow")
14
15 #customers_manager.update_customer_info("101", "chauhan@gmail.com", "123458987", "Noida")
16
17 customers_manager.get_Customer_Details(7)
18
```

Run: Main

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py

Connected to MySQL database
Connected to MySQL database
Customer Details:
Customer ID:7
First Name: Aniket
Last Name: Chaubey
Email: aniket.chaubey@gmail.com
Phone: 8837472973
Address: Indira nagar
Connection closed

```
assignment1 ~/PycharmPre
venv
Customers.py
DatabaseConnector.py
Exception.py
Inventory.py
Main.py
OrderDetails.py
Orders.py
Products.py
External Libraries
Scratches and Consoles

3 from Products import Products
4 from Orders import Orders
5 from OrderDetails import OrderDetails
6 from Inventory import Inventory
7
8 db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
9 db_connector.open_connection()
10 customers_manager = Customers(db_connector)
11
12
13 #customers_manager.create_customer("1001", "Ajay", "Chaudhary", "ajaychaudhary123@gmail.com", "123458759", "Lucknow")
14
15 #customers_manager.update_customer_info("101", "chauhan@gmail.com", "123458987", "Noida")
16
17 #customers_manager.get_Customer_Details(7)
18
19 customers_manager.calculate_Total_Orders(5)
20
```

Run: Main

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py

Connected to MySQL database
Connected to MySQL database
Total Orders for Customer 5: 1
Connection closed
Process finished with exit code 0

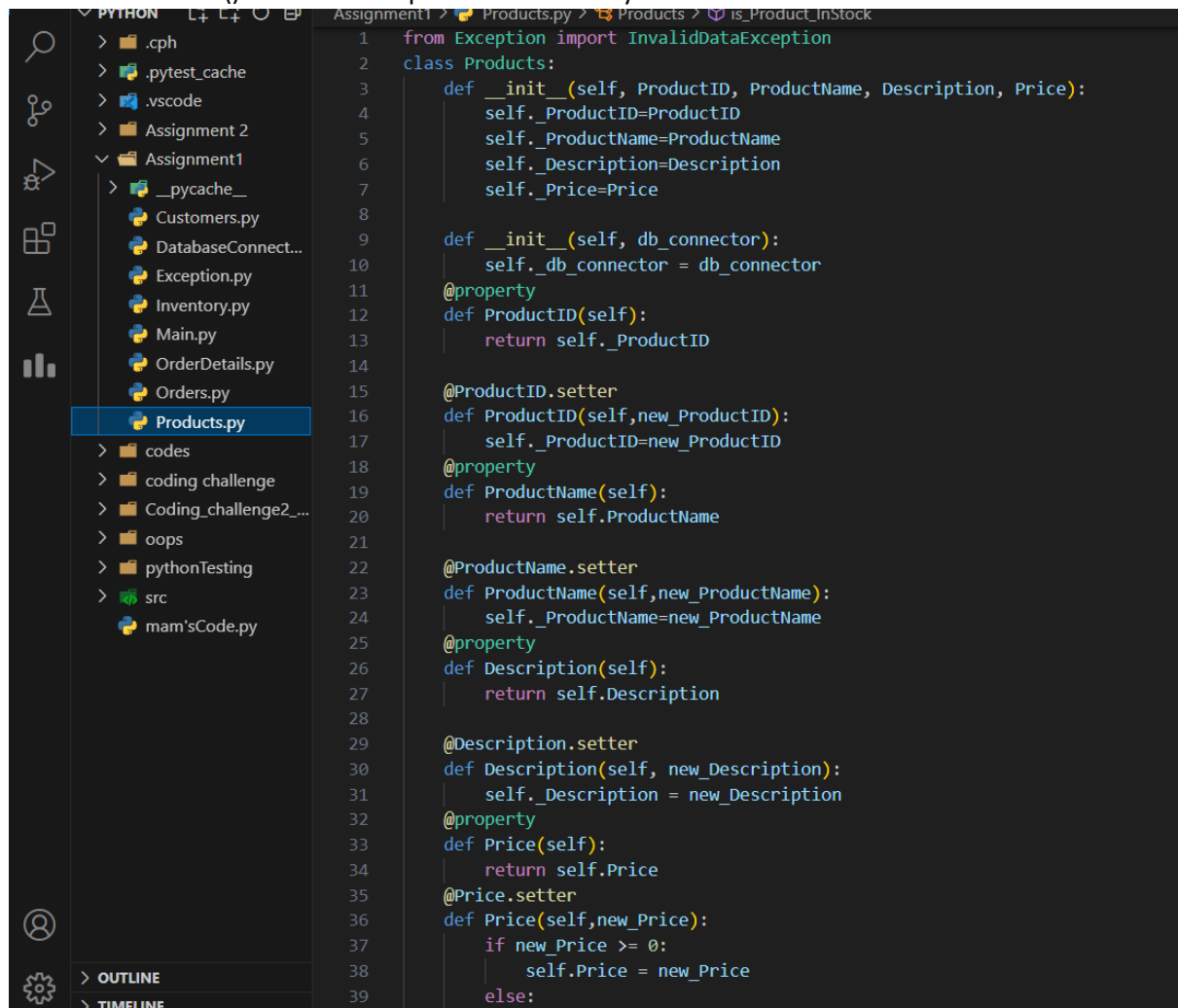
Products Class:

Attributes:

- ProductID (int)
- ProductName (string)
- Description (string)
- Price (decimal)

Methods:

- GetProductDetails(): Retrieves and displays detailed information about the product.
- UpdateProductInfo(): Allows updates to product details (e.g., price, description).
- IsProductInStock(): Checks if the product is currently in stock.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'PYTHON' folder containing various files, including 'Products.py' which is selected. The code editor displays the following Python code:

```
1 from Exception import InvalidDataException
2 class Products:
3     def __init__(self, ProductID, ProductName, Description, Price):
4         self._ProductID=ProductID
5         self._ProductName=ProductName
6         self._Description=Description
7         self._Price=Price
8
9     def __init__(self, db_connector):
10         self._db_connector = db_connector
11
12     @property
13     def ProductID(self):
14         return self._ProductID
15
16     @ProductID.setter
17     def ProductID(self,new_ProductID):
18         self._ProductID=new_ProductID
19
20     @property
21     def ProductName(self):
22         return self._ProductName
23
24     @ProductName.setter
25     def ProductName(self,new_ProductName):
26         self._ProductName=new_ProductName
27
28     @property
29     def Description(self):
30         return self._Description
31
32     @Description.setter
33     def Description(self, new_Description):
34         self._Description = new_Description
35
36     @property
37     def Price(self):
38         return self._Price
39
40     @Price.setter
41     def Price(self,new_Price):
42         if new_Price >= 0:
43             self._Price = new_Price
44         else:
```

```
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

def get_product_details(self,ProductID):
    try:
        self._db_connector.open_connection()

        query = "SELECT * FROM products WHERE ProductID=%s"
        values = (ProductID,)

        self._db_connector.cursor.execute(query, values)

        product_details = self._db_connector.cursor.fetchone()

        if product_details:
            print("Product Details:")
            print(f"Product ID:{product_details[0]}")
            print(f"Product Name: {product_details[1]}")
            print(f"Description : {product_details[2]}")
            print(f"Price: {product_details[3]}")

        else:
            print("Product Id not found.")

    except Exception as e:
        print(f"Error getting Product details: {e}")

    finally:
        self._db_connector.close_connection()

def update_Product_Info(self,ProductID,new_Description,new_Price):
    try:
        self._db_connector.open_connection()
        query="UPDATE Products SET Description=%s,Price=%s WHERE ProductID=%s"
        values=(new_Description,new_Price,ProductID)

        with self._db_connector.connection.cursor() as cursor:
            cursor.execute(query, values)
        self._db_connector.connection.commit()
        print("Product Details updated sucessfully")

    except Exception as e:
```

```
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104

        self._db_connector.cursor.execute(query, values)
        self._db_connector.connection.commit()
        print("Product Details updated sucessfully")

    except Exception as e:
        print(f"Error updating Product Details :{e}")

    finally:
        self._db_connector.close_connection()

def is_Product_InStock(self,ProductID):
    try:
        self._db_connector.open_connection()
        query = "SELECT QuantityInStock FROM Inventory WHERE ProductID = %s"
        values = (ProductID,)

        self._db_connector.cursor.execute(query, values)
        quantity_in_stock = self._db_connector.cursor.fetchone()

        if quantity_in_stock[0]>0:
            print("Product is in stock and stockquantity is",quantity_in_stock[0])
        else:
            print("Product is not in stock")

    except Exception as e:
        print(f"Error checking product stock: {e}")
        return False

    finally:
        self._db_connector.close_connection()
```

```
21
22
23 products_manager1=Products(db_connector)
24
25 products_manager1.get_product_details(1)
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product Details:
Product ID:1
Product Name: nothing earphone
Description : bluetooth earphone
Price: 5000
Connection closed

Process finished with exit code 0
```

```
26 products_manager1.update_Product_Info("5","IQ Z5 ",20000)
27
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product Details updated sucessfully
Connection closed

Process finished with exit code 0
```

```
27
28 products_manager1.is_Product_InStock(1)
29
30
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product is in stock and stockquantity is 15
Connection closed

Process finished with exit code 0
```

Orders Class:

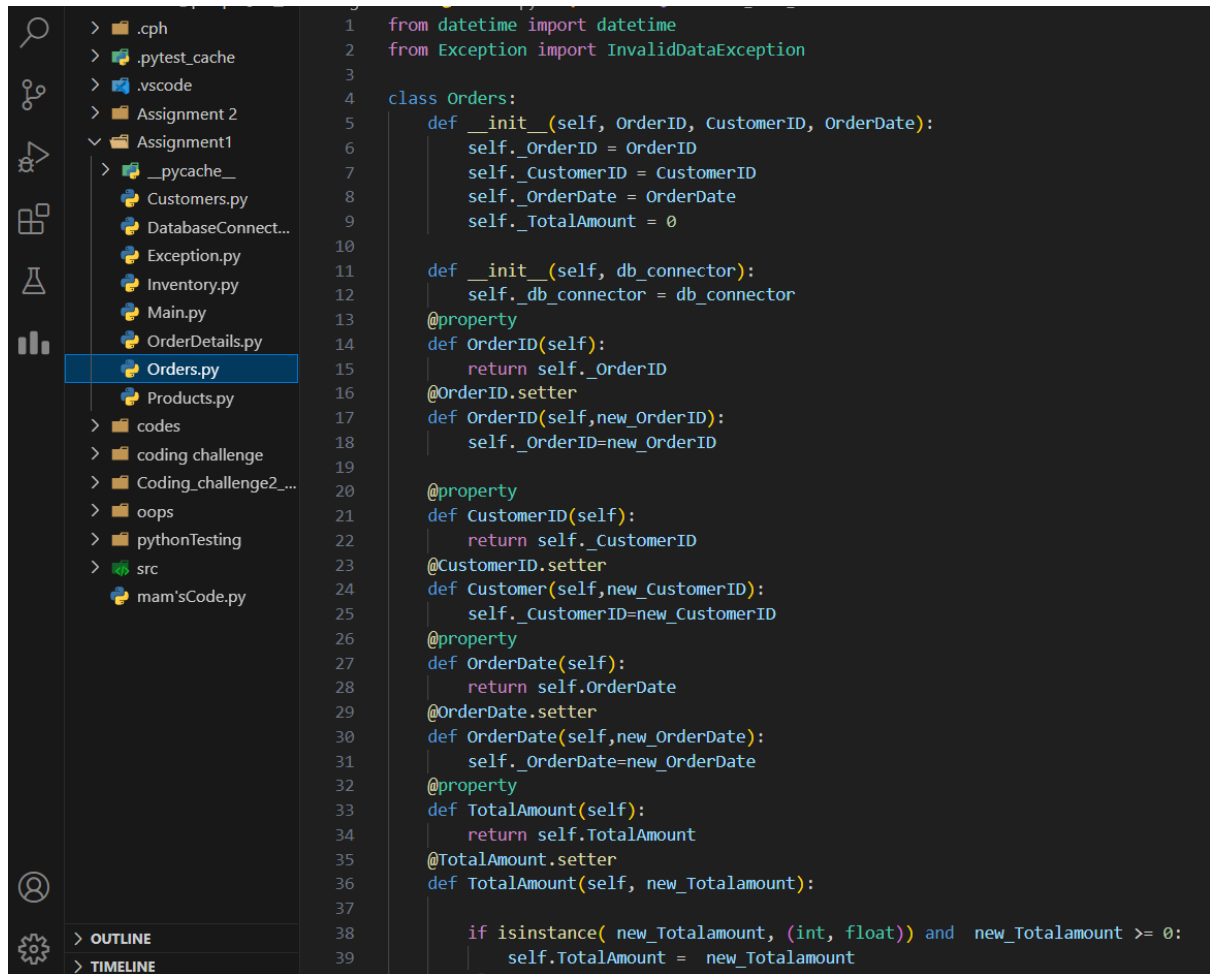
Attributes:

- OrderID (int)
- Customer (Customer) - Use composition to reference the Customer who placed the order.
- OrderDate (DateTime)
- TotalAmount (decimal)

Methods:

- CalculateTotalAmount() - Calculate the total amount of the order.
- GetOrderDetails(): Retrieves and displays the details of the order (e.g., product list and quantities).

- UpdateOrderStatus(): Allows updating the status of the order (e.g., processing, shipped).
- CancelOrder(): Cancels the order and adjusts stock levels for products.



```

1  from datetime import datetime
2  from Exception import InvalidDataException
3
4  class Orders:
5      def __init__(self, OrderID, CustomerID, OrderDate):
6          self._OrderID = OrderID
7          self._CustomerID = CustomerID
8          self._OrderDate = OrderDate
9          self._TotalAmount = 0
10
11     def __init__(self, db_connector):
12         self._db_connector = db_connector
13
14     @property
15     def OrderID(self):
16         return self._OrderID
17     @OrderID.setter
18     def OrderID(self, new_OrderID):
19         self._OrderID = new_OrderID
20
21     @property
22     def CustomerID(self):
23         return self._CustomerID
24     @CustomerID.setter
25     def CustomerID(self, new_CustomerID):
26         self._CustomerID = new_CustomerID
27
28     @property
29     def OrderDate(self):
30         return self._OrderDate
31     @OrderDate.setter
32     def OrderDate(self, new_OrderDate):
33         self._OrderDate = new_OrderDate
34
35     @property
36     def TotalAmount(self):
37         return self._TotalAmount
38     @TotalAmount.setter
39     def TotalAmount(self, new_TotalAmount):
40
41         if isinstance(new_TotalAmount, (int, float)) and new_TotalAmount >= 0:
42             self._TotalAmount = new_TotalAmount

```

```
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82

def Calculate_Total_Amount(self, OrderID):
    try:
        self._db_connector.open_connection()

        query_check_order = "SELECT * FROM Orders WHERE OrderID = %s"
        values = (OrderID,)

        with self._db_connector.connection.cursor() as cursor_check_order:
            cursor_check_order.execute(query_check_order, values)
            order_exists = cursor_check_order.fetchone()

        if not order_exists:
            print("Order ID not found.")
            return

        calculate_total_amount = """
        SELECT SUM(Products.Price * OrderDetails.Qunatity)
        FROM OrderDetails
        JOIN Products ON OrderDetails.ProductID = Products.ProductID
        WHERE OrderDetails.OrderID = %s
        """
        values1 = (OrderID,)

        with self._db_connector.connection.cursor() as cursor_calculate_total_amount:
            cursor_calculate_total_amount.execute(calculate_total_amount, values1)
            total_amount = cursor_calculate_total_amount.fetchone()[0]

        print(f"Total amount for OrderID {OrderID}: {total_amount:.2f}")

    except Exception as e:
        print(f"Error calculating total amount: {e}")

    finally:
        self._db_connector.close_connection()

def get_Order_Details(self, OrderID):
    try:
        self._db_connector.open_connection()
```

```
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116

def get_Order_Details(self, OrderID):
    try:
        self._db_connector.open_connection()

        query = "SELECT * FROM orders WHERE OrderID=%s"
        values = (OrderID,)

        self._db_connector.cursor.execute(query, values)

        order_details = self._db_connector.cursor.fetchone()

        if order_details:
            print("Order Details:")
            print(f"Order ID:{order_details[0]}")
            print(f"Customer ID:{order_details[1]}")
            print(f"Order Date: {order_details[2]}")
            print(f"Total Amount: {order_details[3]}")
        else:
            print("Order Id not found.")

    except Exception as e:
        print(f"Error getting Order details: {e}")

    finally:
        self._db_connector.close_connection()

def place_order(self, OrderID, OrderDetailID, CustomerID, ProductID, Quantity):
    try:
        self._db_connector.open_connection()
        query = "INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES (%s, %s, %s)"
        values = (OrderID, CustomerID, datetime.now())

        with self._db_connector.connection.cursor() as cursor1:
            cursor1.execute(query, values)

        query_details = "INSERT INTO OrderDetails (OrderdetailID, OrderID, ProductID, Quantity) VALUES (%s, %s, %s, %s)"
        values_details = (OrderDetailID, OrderID, ProductID, Quantity)
```

```

117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153

with self.db_connector.connection.cursor() as cursor2:
    cursor2.execute(query_details, values_details)

query_update_inventory = "UPDATE Inventory SET QuantityInStock = QuantityInStock - %s WHERE ProductID = %s"
values_update_inventory = (Quantity, ProductID)
with self.db_connector.connection.cursor() as cursor3:
    cursor3.execute(query_update_inventory, values_update_inventory)

query_total_amount = """
    SELECT SUM(Products.Price * OrderDetails.Quantity)
    FROM OrderDetails
    JOIN Products ON OrderDetails.ProductID = Products.ProductID
    WHERE OrderDetails.OrderID = %s
"""

with self.db_connector.connection.cursor() as cursor4:
    cursor4.execute(query_total_amount, (OrderID,))
    total_amount = cursor4.fetchone()[0]

query_update_total_amount = "UPDATE Orders SET TotalAmount = %s WHERE OrderID = %s"
values_update_total_amount = (total_amount, OrderID)
with self.db_connector.connection.cursor() as cursor5:
    cursor5.execute(query_update_total_amount, values_update_total_amount)
self.db_connector.connection.commit()
print(f"Order placed successfully. OrderID: {OrderID} orderTotal :{total_amount} and inventory updated with available stock")
except Exception as e:
    print(f"Error placing order: {e}")
finally:
    self.db_connector.close_connection()

def updateOrderStatus(self):
    #no status table in database schema
    pass

def cancelOrder(self):
    # no status table in database schema
    pass

```

```

30
31 orders_manager2=Orders(db_connector)
32
33 orders_manager2.get_Order_Details(1)
34 # orders_manager2.Calculate_Total_Amount(2)
35 # orders_manager2.place_order(1015,2015,102,2,2)
36

```

Run: Main x

```

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Order Details:
Order ID:1
Customer ID:1
Order Date: 2023-08-23
Total Amount: 48000
Connection closed

Process finished with exit code 0

```

```

30
31 orders_manager2=Orders(db_connector)
32
33 #orders_manager2.get_Order_Details(1)
34 orders_manager2.Calculate_Total_Amount(2)
35

```

Run: Main x

```

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Error calculating total amount: 1054 (42S22): Unknown column 'OrderDetails.Qunatity' in 'field list'
Connection closed

Process finished with exit code 0

```

OrderDetails Class:

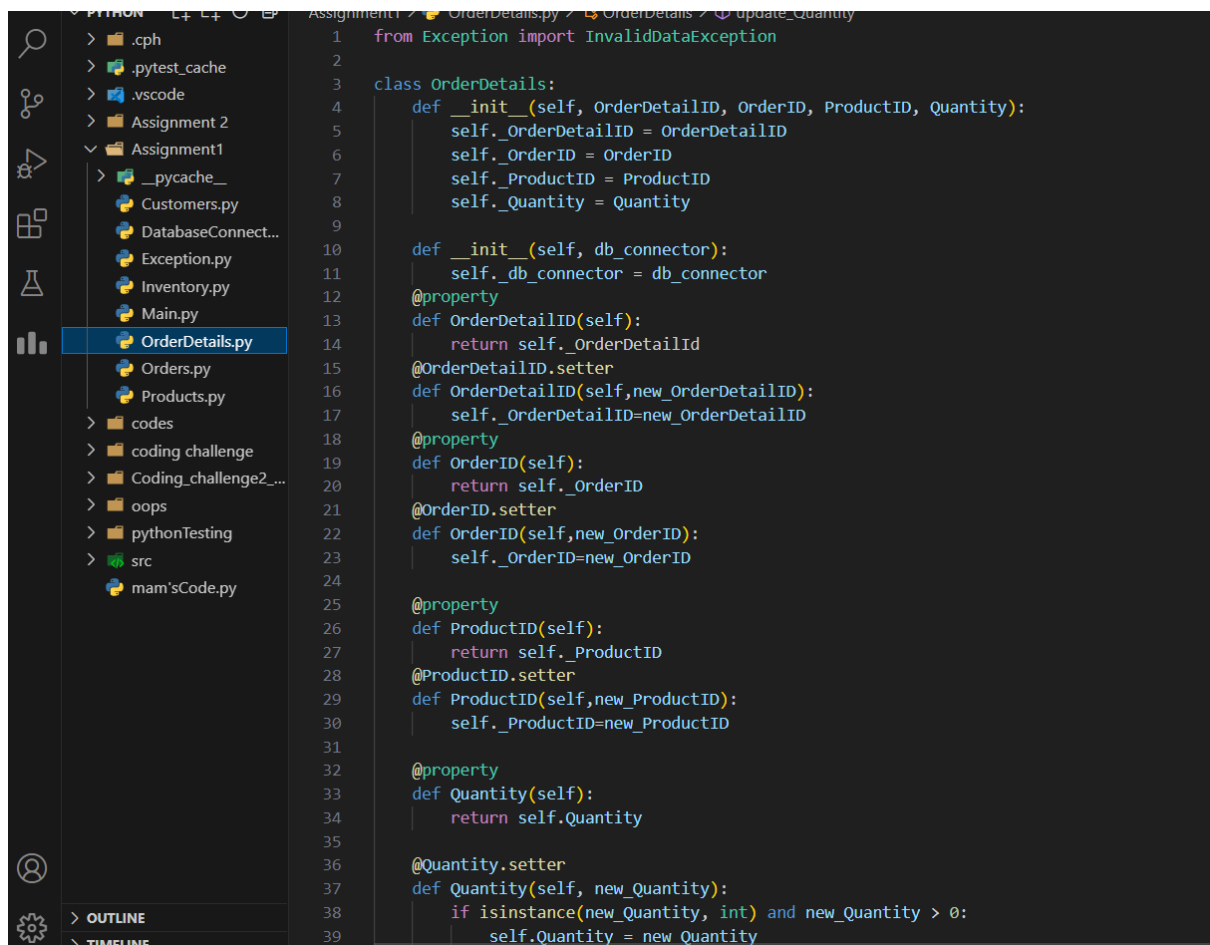
Attributes:

- OrderDetailID (int)
- Order (Order) - Use composition to reference the Order to which this detail belongs.

- Product (Product) - Use composition to reference the Product included in the order detail.
- Quantity (int)

Methods:

- CalculateSubtotal() - Calculate the subtotal for this order detail.
- GetOrderDetailInfo(): Retrieves and displays information about this order detail.
- UpdateQuantity(): Allows updating the quantity of the product in this order detail.
- AddDiscount(): Applies a discount to this order detail.



```

1  from Exception import InvalidDataException
2
3  class OrderDetails:
4      def __init__(self, OrderDetailID, OrderID, ProductID, Quantity):
5          self._OrderDetailID = OrderDetailID
6          self._OrderID = OrderID
7          self._ProductID = ProductID
8          self._Quantity = Quantity
9
10     def __init__(self, db_connector):
11         self._db_connector = db_connector
12
13     @property
14     def OrderDetailID(self):
15         return self._OrderDetailID
16
17     @OrderDetailID.setter
18     def OrderDetailID(self, new_OrderDetailID):
19         self._OrderDetailID = new_OrderDetailID
20
21     @property
22     def OrderID(self):
23         return self._OrderID
24
25     @OrderID.setter
26     def OrderID(self, new_OrderID):
27         self._OrderID = new_OrderID
28
29     @property
30     def ProductID(self):
31         return self._ProductID
32
33     @ProductID.setter
34     def ProductID(self, new_ProductID):
35         self._ProductID = new_ProductID
36
37     @property
38     def Quantity(self):
39         return self._Quantity
40
41     @Quantity.setter
42     def Quantity(self, new_Quantity):
43         if isinstance(new_Quantity, int) and new_Quantity > 0:
44             self._Quantity = new_Quantity
  
```

```
> .pytest_cache 43
> .vscode 44
> Assignment 2 45
> Assignment1 46
  > __pycache__ 47
    Customers.py 48
    DatabaseConnect... 49
    Exception.py 50
    Inventory.py 51
    Main.py 52
    OrderDetails.py 53
    Orders.py 54
    Products.py 55
  > codes 56
  > coding challenge 57
  > Coding_challenge2_... 58
  > oops 59
  > pythonTesting 60
  > src 61
    mam'sCode.py 62

> OUTLINE 63

def Calculate_Subtotal(self, OrderDetailID):
    try:
        self._db_connector.open_connection()
        query = "SELECT OD.Qunatity, P.Price FROM Orderdetails OD INNER JOIN Products P ON OD.ProductID = P.ProductID WHE
        values = (OrderDetailID,)"

        with self._db_connector.cursor as cursor:
            cursor.execute(query, values)
            order_data = cursor.fetchone()

        if order_data:
            quantity, price = order_data
            subtotal = quantity * price
            print(f"Subtotal for OrderDetailID {OrderDetailID}: {subtotal}")
        else:
            print("Order detail not found.")
    except Exception as e:
        print(f"Error calculating subtotal: {e}")
    finally:
        self._db_connector.close_connection()

def get_OrderDetail_Info(self, OrderDetailID):
    try:
        self._db_connector.open_connection()

        query = "SELECT * FROM orderdetails WHERE OrderDetailID=%s"
        values = (OrderDetailID,)"

        self._db_connector.cursor.execute(query, values)

        orderdetails_details = self._db_connector.cursor.fetchone()

        if orderdetails_details:
            print("OrderDetails Details:")
            print(f"OrderDetailID:{orderdetails_details[0]}")
            print(f"OrderID:{orderdetails_details[1]}")
            print(f"ProductID: {orderdetails_details[2]}")
            print(f"Quantity: {orderdetails_details[3]}")
```

```
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115

        print(f"Quantity: {orderdetails_details[3]}")

        else:
            print("OrderDetails Id not found.")

    except Exception as e:
        print(f"Error getting Orderdetails: {e}")

    finally:
        self._db_connector.close_connection()

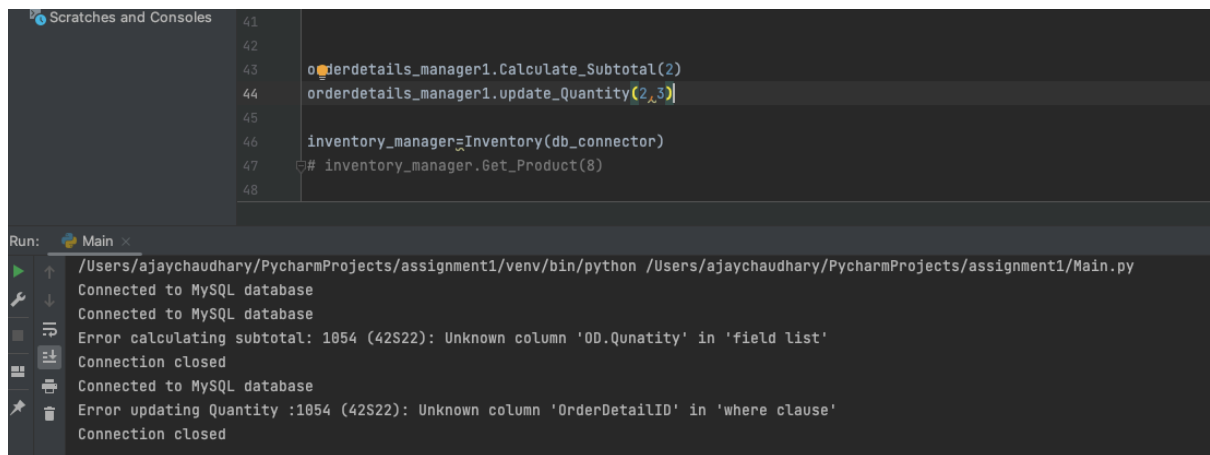
def update_Quantity(self, OrderDetailID, new_Quantity):
    try:
        self._db_connector.open_connection()
        query = "UPDATE OrderDetails SET Qunatity=%s where OrderDetailID=%s"
        values = (new_Quantity, OrderDetailID)

        with self._db_connector.connection.cursor() as cursor:
            cursor.execute(query, values)
            self._db_connector.connection.commit()
            print("Quantity updated sucessfully")

    except Exception as e:
        print(f"Error updating Quantity :{e}")

    finally:
        self._db_connector.close_connection()

def AddDiscount(self, discount_amount):
    try:
        if discount_amount < 0:
            raise InvalidDataException("Invalid discount amount")
    except InvalidDataException as e:
        print(f"Discount not applied: {str(e)}")
```



The screenshot shows a PyCharm IDE with a code editor and a console window. The code editor displays the following Python code:

```
41  
42  
43 orderdetails_manager1.Calculate_Subtotal(2)  
44 orderdetails_manager1.update_Quantity(2,3)  
45  
46 inventory_manager=Inventory(db_connector)  
47 # inventory_manager.Get_Product(8)  
48
```

The console window shows the output of the program, including connection messages and error messages:

```
Run: Main x  
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py  
Connected to MySQL database  
Connected to MySQL database  
Error calculating subtotal: 1054 (42S22): Unknown column 'OD.Qunatity' in 'field list'  
Connection closed  
Connected to MySQL database  
Error updating Quantity :1054 (42S22): Unknown column 'OrderDetailID' in 'where clause'  
Connection closed
```

Inventory class:

Attributes:

- InventoryID(int)
- Product (Composition): The product associated with the inventory item.
- QuantityInStock: The quantity of the product currently in stock.
- LastStockUpdate

Methods:

- GetProduct(): A method to retrieve the product associated with this inventory item.
- GetQuantityInStock(): A method to get the current quantity of the product in stock.
- AddToInventory(int quantity): A method to add a specified quantity of the product to the inventory.
- RemoveFromInventory(int quantity): A method to remove a specified quantity of the product from the inventory.
- UpdateStockQuantity(int newQuantity): A method to update the stock quantity to a new value.
- IsProductAvailable(int quantityToCheck): A method to check if a specified quantity of the product is available in the inventory.
- GetInventoryValue(): A method to calculate the total value of the products in the inventory based on their prices and quantities.
- ListLowStockProducts(int threshold): A method to list products with quantities below a specified threshold, indicating low stock.
- ListOutOfStockProducts(): A method to list products that are out of stock
- ListAllProducts(): A method to list all products in the inventory, along with their quantities.

```

1  from Exception import InvalidDataException, InsufficientStockException
2  from datetime import datetime
3  class Inventory:
4      def __init__(self, InventoryID, ProductID, QuantityInStock, LastStockUpdate):
5          self._InventoryID = InventoryID
6          self._ProductID = ProductID
7          self._QuantityInStock = QuantityInStock
8          self._LastStockUpdate = LastStockUpdate
9
10     def __init__(self, db_connector):
11         self._db_connector = db_connector
12
13     @property
14     def InventoryID(self):
15         return self._InventoryID
16     @InventoryID.setter
17     def InventoryID(self, new_InventoryID):
18         self._InventoryID = new_InventoryID
19     @property
20     def ProductID(self):
21         return self._ProductID
22     @ProductID.setter
23     def ProductID(self, new_ProductID):
24         self._ProductID = new_ProductID
25     @property
26     def QuantityInStock(self):
27         return self._QuantityInStock
28     @QuantityInStock.setter
29     def QuantityInStock(self, new_QuantityInStock):
30         if new_QuantityInStock >= 0:

```

```

29     def QuantityInStock(self, new_QuantityInStock):
30         if new_QuantityInStock >= 0:
31             self._QuantityInStock = new_QuantityInStock
32         else:
33             raise InvalidDataException("Quantity in stock must be non-negative")
34     @property
35     def LastStockUpdate(self):
36         return self._LastStockUpdate
37     @LastStockUpdate.setter
38     def LastStockUpdate(self, new_LastStockUpdate):
39         self._LastStockUpdate = new_LastStockUpdate
40
41     def Get_Product(self, InventoryID):
42         try:
43             self._db_connector.open_connection()
44             query = ("SELECT P.ProductID, P.ProductName, P.Description, P.Price FROM Products P"
45                     " INNER JOIN Inventory I ON P.ProductID = I.ProductID WHERE I.InventoryID = %s")
46             values = (InventoryID,)
47
48             with self._db_connector.cursor as cursor:
49                 cursor.execute(query, values)
50                 product_data = cursor.fetchone()
51
52                 if product_data:
53                     product_id, product_name, description, price = product_data
54                     print(f"Product ID: {product_id}, Product Name: {product_name}, Description: {description}, Price: rs{price}")
55                 else:
56                     print("Product not found in the inventory.")
57
58         except Exception as e:

```

```

47 def Is_Product_Available(self, quantity_to_check, InventoryID):
48     try:
49         self._db_connector.open_connection()
50         query = "SELECT QuantityInStock FROM Inventory WHERE InventoryID = %s"
51         values = (InventoryID,)
52         with self._db_connector.cursor as cursor:
53             cursor.execute(query, values)
54             current_quantity = cursor.fetchone()
55             if current_quantity is not None and current_quantity[0] >= quantity_to_check:
56                 print(f"Product is available in sufficient quantity: {current_quantity[0]} units.")
57                 return True
58             else:
59                 print("Product is not available in sufficient quantity.")
60                 return False
61     except Exception as e:
62         print(f"Error checking product availability: {e}")
63         return False
64
65     finally:
66         self._db_connector.close_connection()
67
68 def Get_InventoryValue(self):
69     try:
70         self._db_connector.open_connection()
71         query = "SELECT P.ProductID, P.Price, I.QuantityInStock FROM Products P INNER JOIN Inventory I ON P.ProductID = I.ProductID"
72         with self._db_connector.cursor as cursor:
73             cursor.execute(query)
74             products_data = cursor.fetchall()
75             if products_data:
76                 total_value = 0

```

```

188 def List_LowStock_Products(self, threshold):
189     try:
190         self._db_connector.open_connection()
191         query = "SELECT ProductID, QuantityInStock FROM Inventory WHERE QuantityInStock < %s"
192         values = (threshold,)
193         with self._db_connector.cursor as cursor:
194             cursor.execute(query, values)
195             low_stock_products = cursor.fetchall()
196             if low_stock_products:
197                 print("Low stock products:")
198                 for product in low_stock_products:
199                     print(f"ProductID: {product[0]}, QuantityInStock: {product[1]}")
200             else:
201                 print("No products with quantities below the specified threshold.")
202
203     except Exception as e:
204         print(f"Error listing low stock products: {e}")
205
206     finally:
207         self._db_connector.close_connection()
208

```

```

46 inventory_manager=Inventory(db_connector)
47 inventory_manager.Get_Product(8)

```

Run: Main x

/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py

Connected to MySQL database

Connected to MySQL database

Product ID: 8, Product Name: OnePlus 10R, Description: android phone, Price: rs25000

Connection closed

Process finished with exit code 0


```
53     inventory_manager.remove_From_Inventory(2,1)
54
55     # inventory_manager.update_Stock_Quantity(2,301)
56
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Connected to MySQL database
2 units removed from inventory successfully.
Connection closed

Process finished with exit code 0
```

```
56
57     inventory_manager.Is_Product_Available(2,3)
58
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product is available in sufficient quantity: 25 units.
Connection closed

Process finished with exit code 0
```

```
49     #inventory_manager.get_QuantityInStock(1)
50
51     inventory_manager.add_To_Inventory(2,301)
52
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Connected to MySQL database
2 units added to inventory successfully.
Connection closed

Process finished with exit code 0
```

Version Control Run Python Packages TODO Python Console Problems Terminal Services

```
54
55     inventory_manager.update_Stock_Quantity(2,301)
56
57     # inventory_manager.Is_Product_Available(2,3)
58
```

Run: Main x

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Stock quantity updated to 2 successfully.
Connection closed

Process finished with exit code 0
```

```
59     inventory_manager.Get_InventoryValue()
60
61     # inventory_manager.List_LowStock_Products(8)
```

Run: Main ×

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Total value of the inventory: rs4034000
Connection closed
Process finished with exit code 0
```

```
63     inventory_manager.List_OutOf_StockProducts()
64
65     # inventory_manager.List_All_Products()
66
```

Run: Main ×

```
/Users/ajaychaudhary/PycharmProjects/assignment1/venv/bin/python /Users/ajaychaudhary/PycharmProjects/assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
No products are currently out of stock.
Connection closed
Process finished with exit code 0
```

```
65     inventory_manager.List_All_Products()
66
```

Run: Main ×

```
All products in the inventory:
ProductID: 1, QuantityInStock: 13
ProductID: 2, QuantityInStock: 10
ProductID: 3, QuantityInStock: 25
ProductID: 4, QuantityInStock: 15
ProductID: 5, QuantityInStock: 30
ProductID: 6, QuantityInStock: 28
ProductID: 7, QuantityInStock: 40
ProductID: 8, QuantityInStock: 10
ProductID: 9, QuantityInStock: 22
ProductID: 10, QuantityInStock: 12
```

Version Control Run Python Packages TODO Python Console Problems Terminal Services