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).

```python
from Exception import InvalidDataException

class Customers:
    def __init__(self,CustomerId,FirstName,LastName,Email,Phone,Address):
        self._CustomerId=CustomerId
        self._FirstName=FirstName
        self._LastName=LastName
        self._Email=Email
        self._Phone=Phone
        self._Address=Address

    def __init__(self,db_connector):
        self._db_connector = db_connector
    @property
    def CustomerId(self):
        return self._CustomerId
    @CustomerId.setter
    def CustomerId(self, new_customer_id):
        self._CustomerId = new_customer_id
    @property
    def FirstName(self):
        return self._FirstName
    @FirstName.setter
    def FirstName(self,new_FirstName):
        self._FirstName=new_FirstName
    @property
    def LastName(self):
        return self._LastName

```

```python
    @Email.setter
    def Email(self,new_Email):
        if "@" in new_Email and "." in new_Email:
            self._Email =new_Email
        else:
            raise InvalidDataException("Invalid email format.")

    @property
    def Phone(self):
        return self._Phone
    @Phone.setter
    def Phone(self, new_Phone):
        if len(new_Phone) == 10 and new_Phone.isdigit():
            self._Phone = new_Phone
        else:
            raise InvalidDataException("Invalid phone number format.")

    @property
    def Address(self):
        return self._Address
    @Address.setter
    def Address(self, new_Address):
        if isinstance(new_Address, str):
            self._Address = new_Address
        else:
            raise InvalidDataException("Invalid address format")

    def create_customer(self,CustomerId,FirstName,LastName,Email, Phone,Address):
        try:
            self._db_connector.open_connection()
```

```python
    def create_customer(self,CustomerId,FirstName,LastName,Email, Phone,Address):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM customers WHERE Email = %s", (Email,))
            existing_customer = cursor.fetchone()

            if existing_customer:
                print("Error: Email address is already in use.")
            else:
                cursor.execute("INSERT INTO customers (CustomerId,FirstName,LastName, Email, Phone,Address) VALUES (%s, %s, %s,%s,%s,%s)"
                self._db_connector.connection.commit()
                print("Customer created successfully")

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

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()
    def calculate_Total_Orders(self,Customerid):
        try:
            self._db_connector.open_connection()
            query = """ SELECT COUNT(OrderID) AS TotalOrders FROM Orders
                        WHERE Customerid = %s"""
            values = (Customerid,)
            self._db_connector.cursor.execute(query, values)
            result = self._db_connector.cursor.fetchone()
```

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

            customer_details=self._db_connector.cursor.fetchone()

            if customer_details:
                print("Customer Details:")
                print(f"Customer ID:{customer_details[0]}")
                print(f"First Name: {customer_details[1]}")
                print(f"Last Name: {customer_details[2]}")
                print(f"Email: {customer_details[3]}")
                print(f"Phone: {customer_details[4]}")
                print(f"Address: {customer_details[5]}")
            else:
                 print("Customer not found.")

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

        finally:
            self._db_connector.close_connection()


    def update_customer_Info(self,Customerid,new_Email,new_Phone,new_Address):
        try:
            self._db_connector.open_connection()

            query="UPDATE customers SET Email=%s,Phone=%s, Address=%s WHERE Customerid=%s"
            values=(new_Email,new_Phone,new_Address,Customerid)

```

```python
    def calculate_Total_Orders(self,Customerid):
        try:
            self._db_connector.open_connection()
            query = """ SELECT COUNT(OrderID) AS TotalOrders FROM Orders
                        WHERE Customerid = %s"""
            values = (Customerid,)
            self._db_connector.cursor.execute(query, values)
            result = self._db_connector.cursor.fetchone()

            if result:
                total_orders = result[0]
                print(f"Total Orders for Customer {Customerid}: {total_orders}")
            else:
                print("No orders found for this customer")

        except Exception as e:
            print(f"Error calculating total orders: {e}")
        finally:
            self._db_connector.close_connection()


    def get_Customer_Details(self,Customerid):
        try:
            self._db_connector.open_connection()

            query ="SELECT * FROM customers WHERE CustomerId=%s"
            values =(Customerid,)

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

```python
from DatabaseConnector import DatabaseConnector
from Customers import Customers
from Products import Products
from Orders import Orders
from OrderDetails import OrderDetails
from Inventory import Inventory

db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
db_connector.open_connection()
customers_manager= Customers(db_connector)


#customers_manager.create_customer("1001","Ajay","Chaudhary","ajaychaudhary123@gmail.com", "123458759","Lucknow")

customers_manager.update_customer_Info("101", "chauhan@gmail.com", "123458907", "Noida")
```

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

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 sucessfully
Connection closed

Process finished with exit code 0
```

```python
from DatabaseConnector import DatabaseConnector
from Customers import Customers
from Products import Products
from Orders import Orders
from OrderDetails import OrderDetails
from Inventory import Inventory

db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
db_connector.open_connection()
customers_manager= Customers(db_connector)


customers_manager.create_customer("1001","Ajay","Chaudhary", "ajaychaudhary123@gmail.com", "123458759","Lucknow")
```

```
/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
```

```python
from DatabaseConnector import DatabaseConnector
from Customers import Customers
from Products import Products
from Orders import Orders
from OrderDetails import OrderDetails
from Inventory import Inventory

db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
db_connector.open_connection()
customers_manager= Customers(db_connector)


#customers_manager.create_customer("1001","Ajay","Chaudhary", "ajaychaudhary123@gmail.com", "123458759","Lucknow")

#customers_manager.update_customer_Info("101", "chauhan@gmail.com", "123458907", "Noida")

customers_manager.get_Customer_Details(7)
```

```
/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
```

```python
from Products import Products
from Orders import Orders
from OrderDetails import OrderDetails
from Inventory import Inventory

db_connector = DatabaseConnector(host="localhost", database="TS", user="root", password="qwerty123")
db_connector.open_connection()
customers_manager= Customers(db_connector)


#customers_manager.create_customer("1001","Ajay","Chaudhary", "ajaychaudhary123@gmail.com", "123458759","Lucknow")

#customers_manager.update_customer_Info("101", "chauhan@gmail.com", "123458907", "Noida")

#customers_manager.get_Customer_Details(7)

customers_manager.calculate_Total_Orders(5)
```

```
/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
```

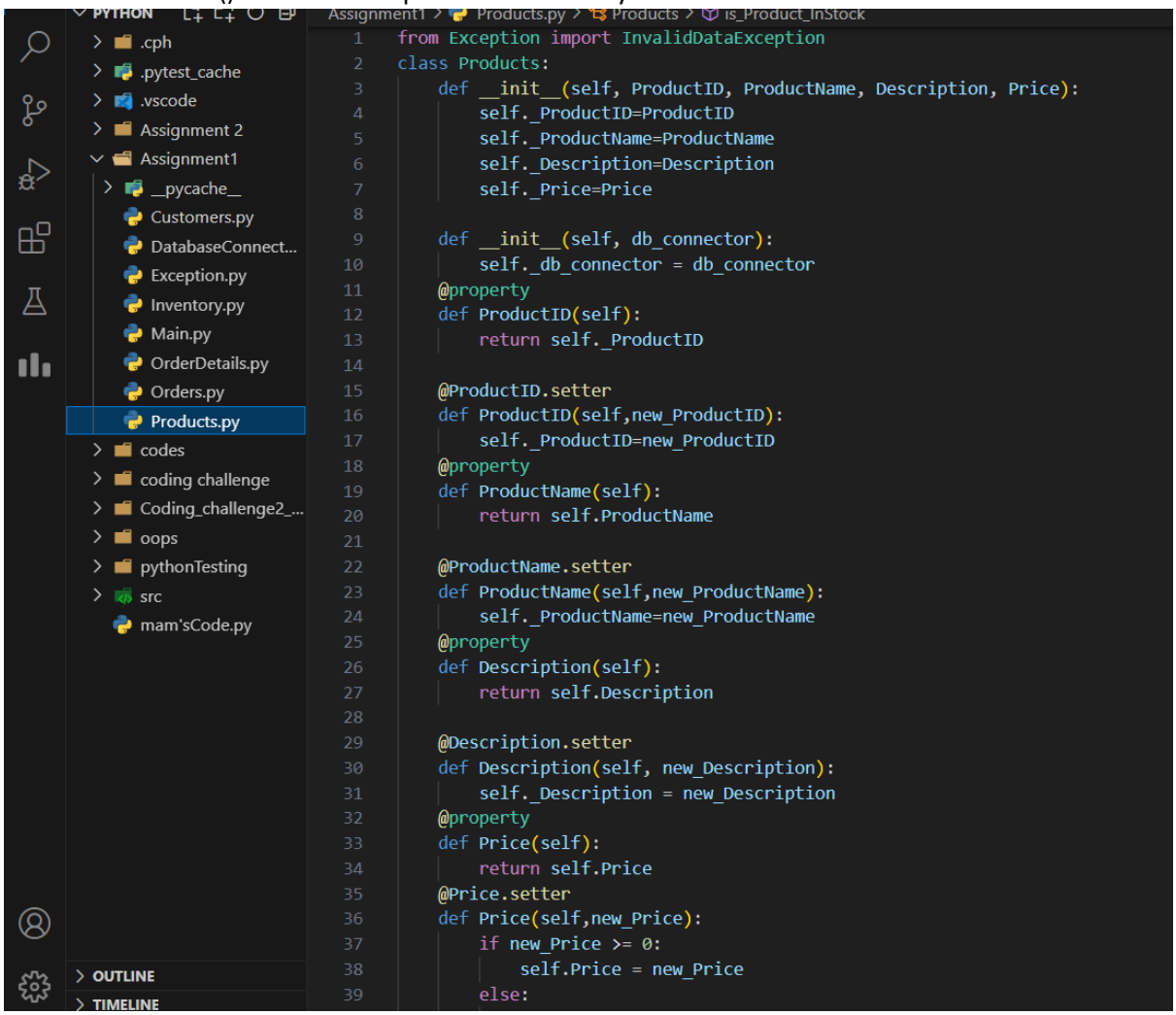
Products Class:

Attributes:

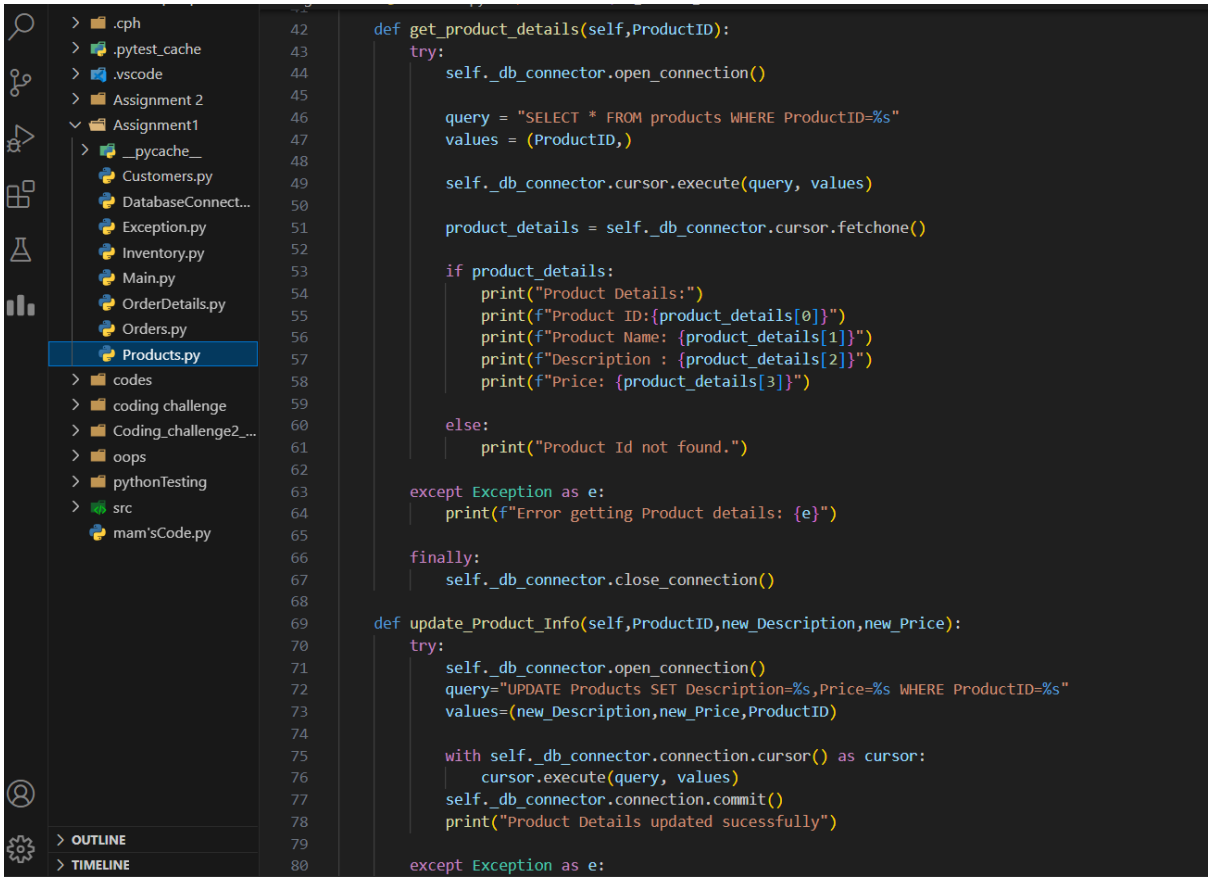
- ProductID (int)
- ProductName (string)
- Description (string)
- Price (decimal)

Methods:

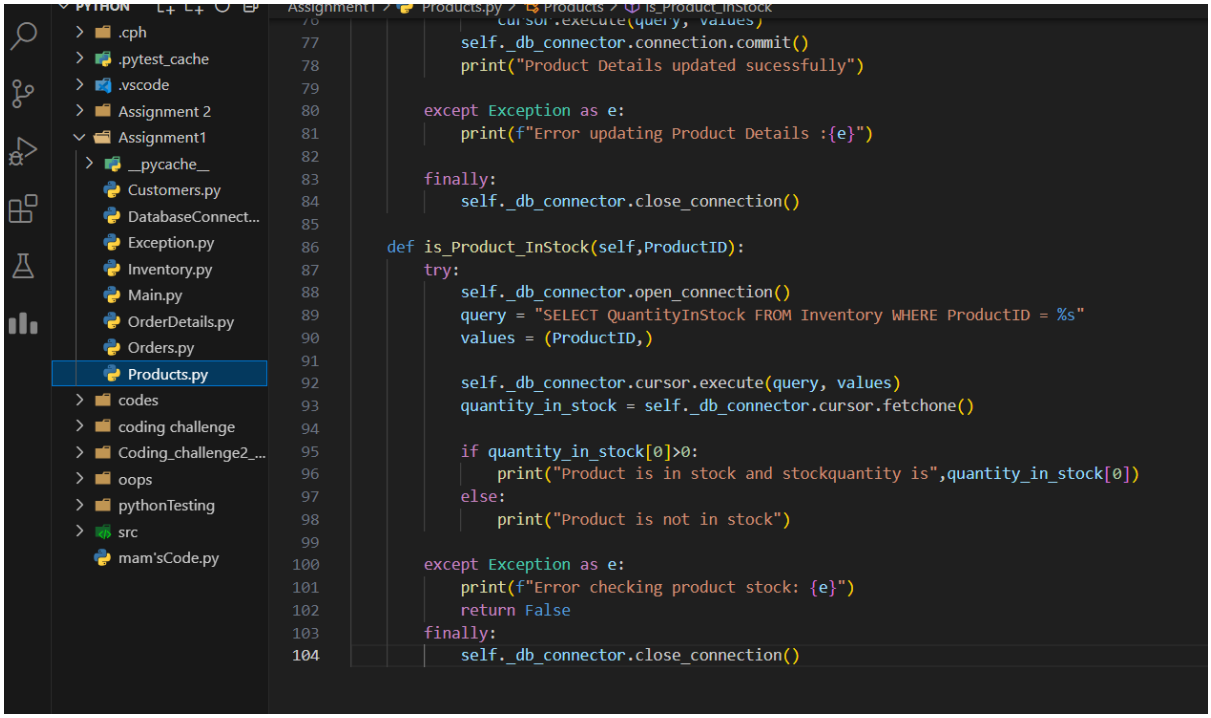
- GetProductDetails(): Retrieves and displays detailed information about the product.
- UpdateProductInfo(): Allows updates to product details (e.g., price, description).
- IsProductInStock(): Checks if the product is currently in stock.




```python
from Exception import InvalidDataException
class Products:
    def __init__(self, ProductID, ProductName, Description, Price):
        self._ProductID=ProductID
        self._ProductName=ProductName
        self._Description=Description
        self._Price=Price

    def __init__(self, db_connector):
        self._db_connector = db_connector
    @property
    def ProductID(self):
        return self._ProductID

    @ProductID.setter
    def ProductID(self,new_ProductID):
        self._ProductID=new_ProductID
    @property
    def ProductName(self):
        return self.ProductName

    @ProductName.setter
    def ProductName(self,new_ProductName):
        self._ProductName=new_ProductName
    @property
    def Description(self):
        return self.Description

    @Description.setter
    def Description(self, new_Description):
        self._Description = new_Description
    @property
    def Price(self):
        return self.Price
    @Price.setter
    def Price(self,new_Price):
        if new_Price >= 0:
            self.Price = new_Price
        else:
```




```python
    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:
```




```python
            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()
```

```
products_manager1=Products(db_connector)

products_manager1.get_product_details(1)
```

```
Run: Main
/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
```

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

```
Run: Main
/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
```

```
products_manager1.is_Product_InStock(1)
```

```
Run: Main
/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)

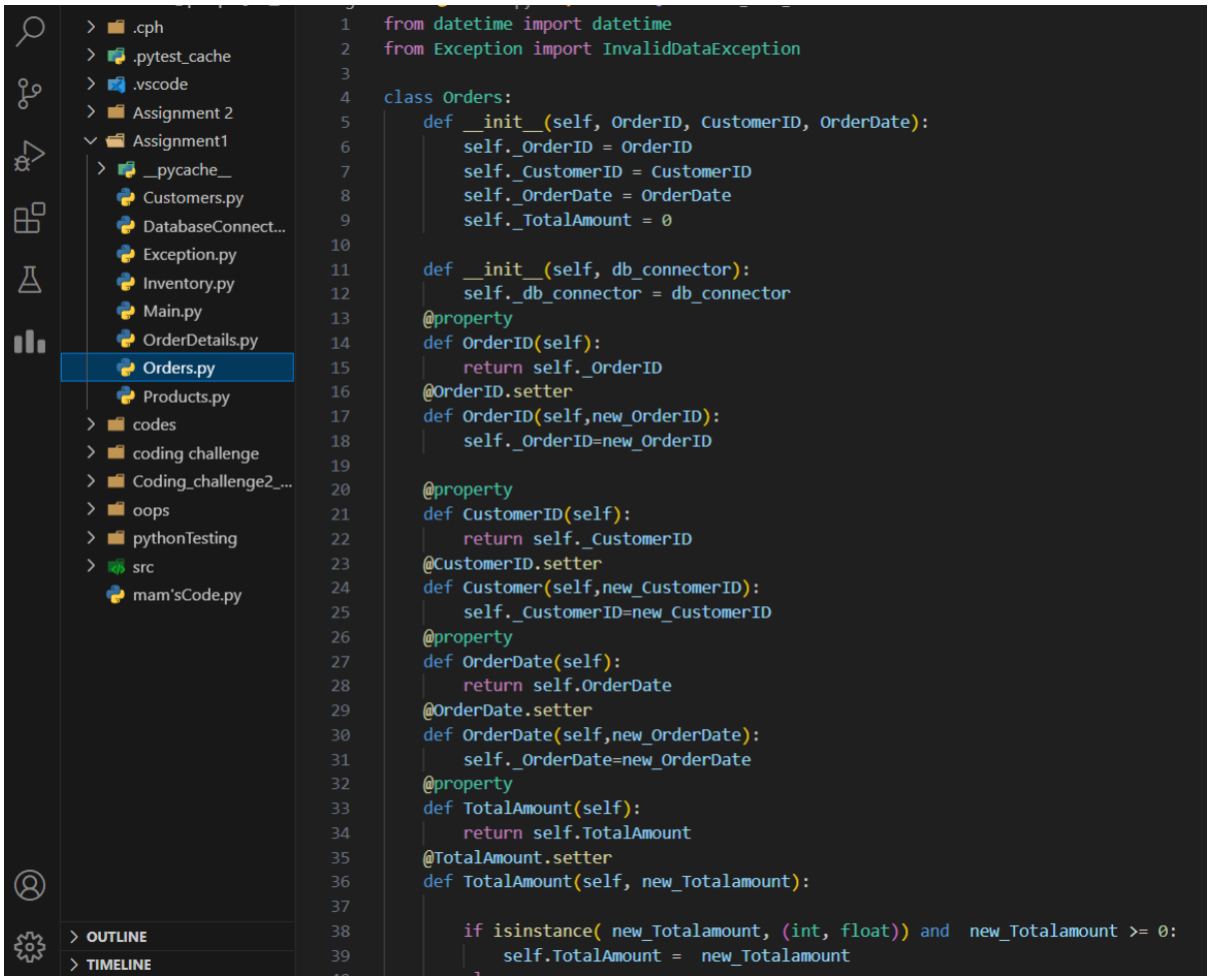
Methods:

• CalculateTotalAmount() - Calculate the total amount of the order.

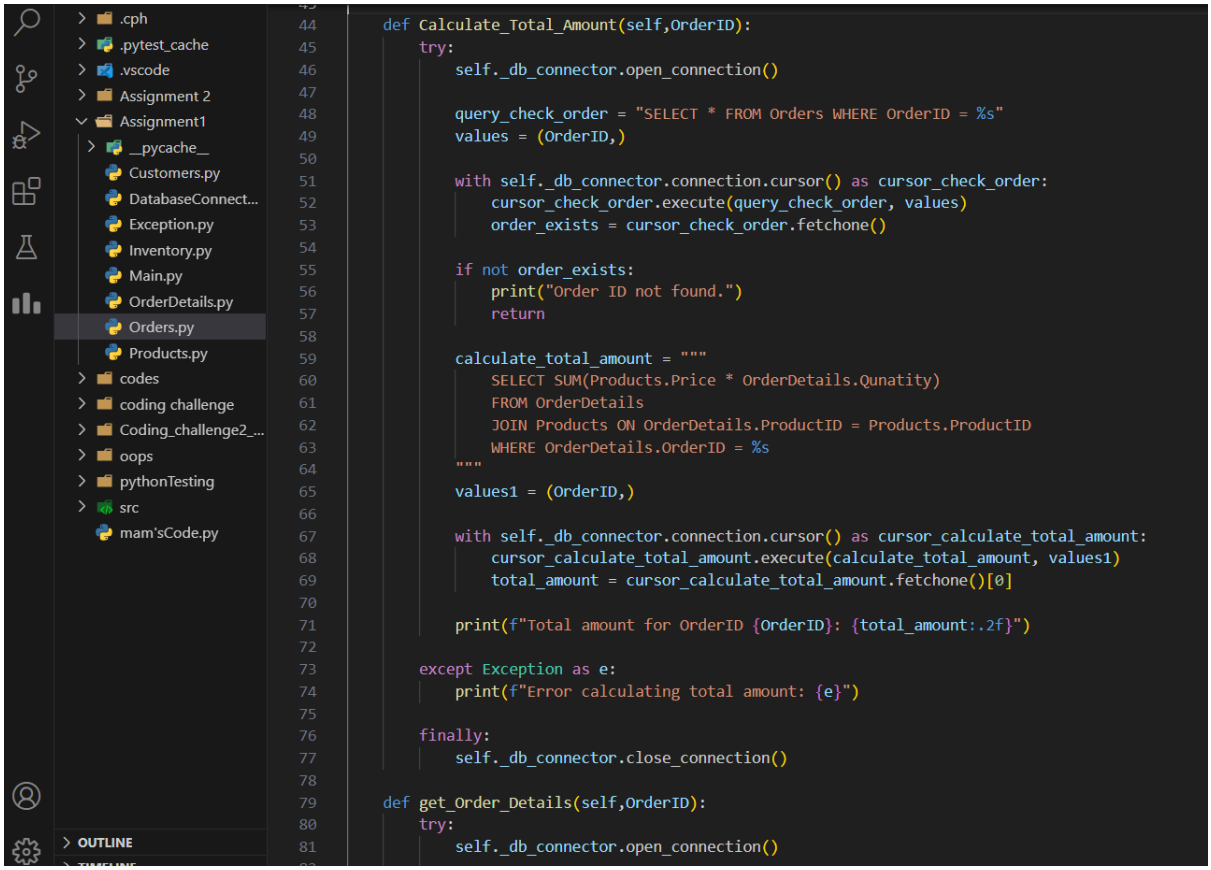
• GetOrderDetails(): Retrieves and displays the details of the order (e.g., product list and quantities).

• UpdateOrderStatus(): Allows updating the status of the order (e.g., processing, shipped).

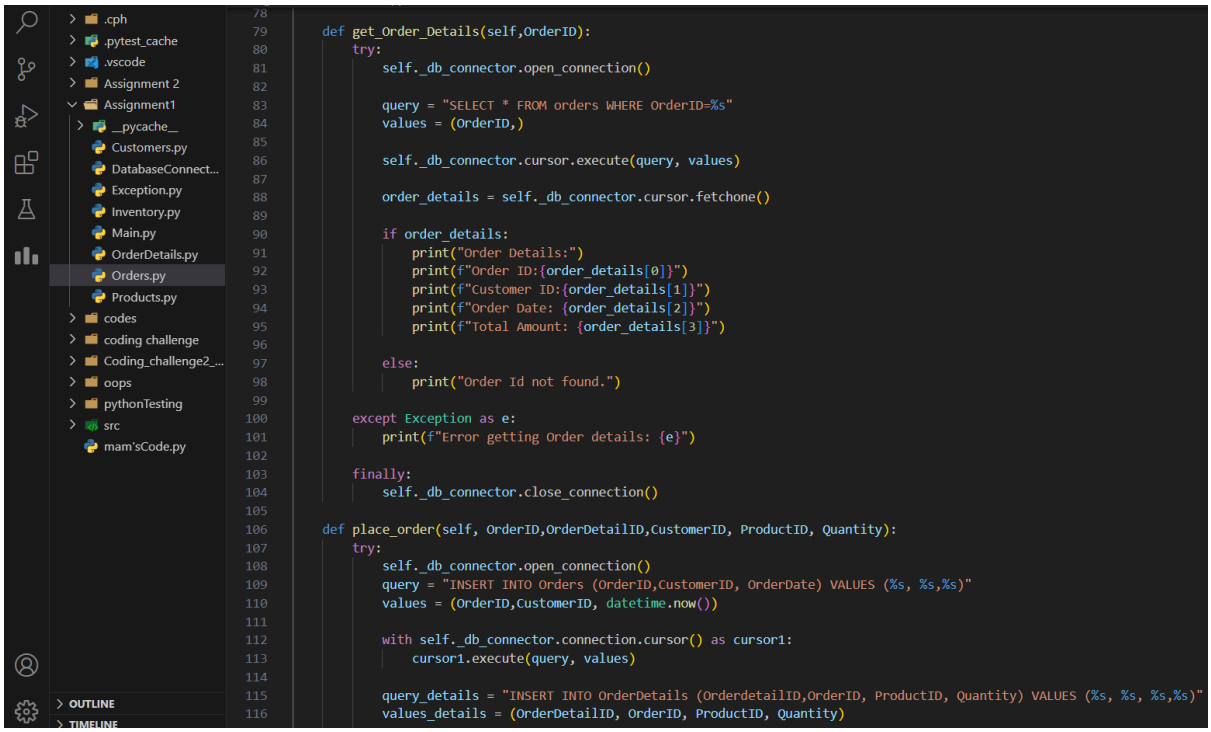
• CancelOrder(): Cancels the order and adjusts stock levels for products.




```python
from datetime import datetime
from Exception import InvalidDataException

class Orders:
    def __init__(self, OrderID, CustomerID, OrderDate):
        self._OrderID = OrderID
        self._CustomerID = CustomerID
        self._OrderDate = OrderDate
        self._TotalAmount = 0

    def __init__(self, db_connector):
        self._db_connector = db_connector
    @property
    def OrderID(self):
        return self._OrderID
    @OrderID.setter
    def OrderID(self,new_OrderID):
        self._OrderID=new_OrderID

    @property
    def CustomerID(self):
        return self._CustomerID
    @CustomerID.setter
    def Customer(self,new_CustomerID):
        self._CustomerID=new_CustomerID
    @property
    def OrderDate(self):
        return self.OrderDate
    @OrderDate.setter
    def OrderDate(self,new_OrderDate):
        self._OrderDate=new_OrderDate
    @property
    def TotalAmount(self):
        return self.TotalAmount
    @TotalAmount.setter
    def TotalAmount(self, new_Totalamount):

        if isinstance( new_Totalamount, (int, float)) and  new_Totalamount >= 0:
            self.TotalAmount =  new_Totalamount
```




```python
    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()
```




```python
    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)
```

```python
            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 avaialble 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
```

```python
orders_manager2=Orders(db_connector)

orders_manager2.get_Order_Details(1)
# orders_manager2.Calculate_Total_Amount(2)
# orders_manager2.place_order(1015,2015,102,2,2)
```

```
Run: Main
/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
```

```python
orders_manager2=Orders(db_connector)

#orders_manager2.get_Order_Details(1)
orders_manager2.Calculate_Total_Amount(2)
```

```
Main
/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)

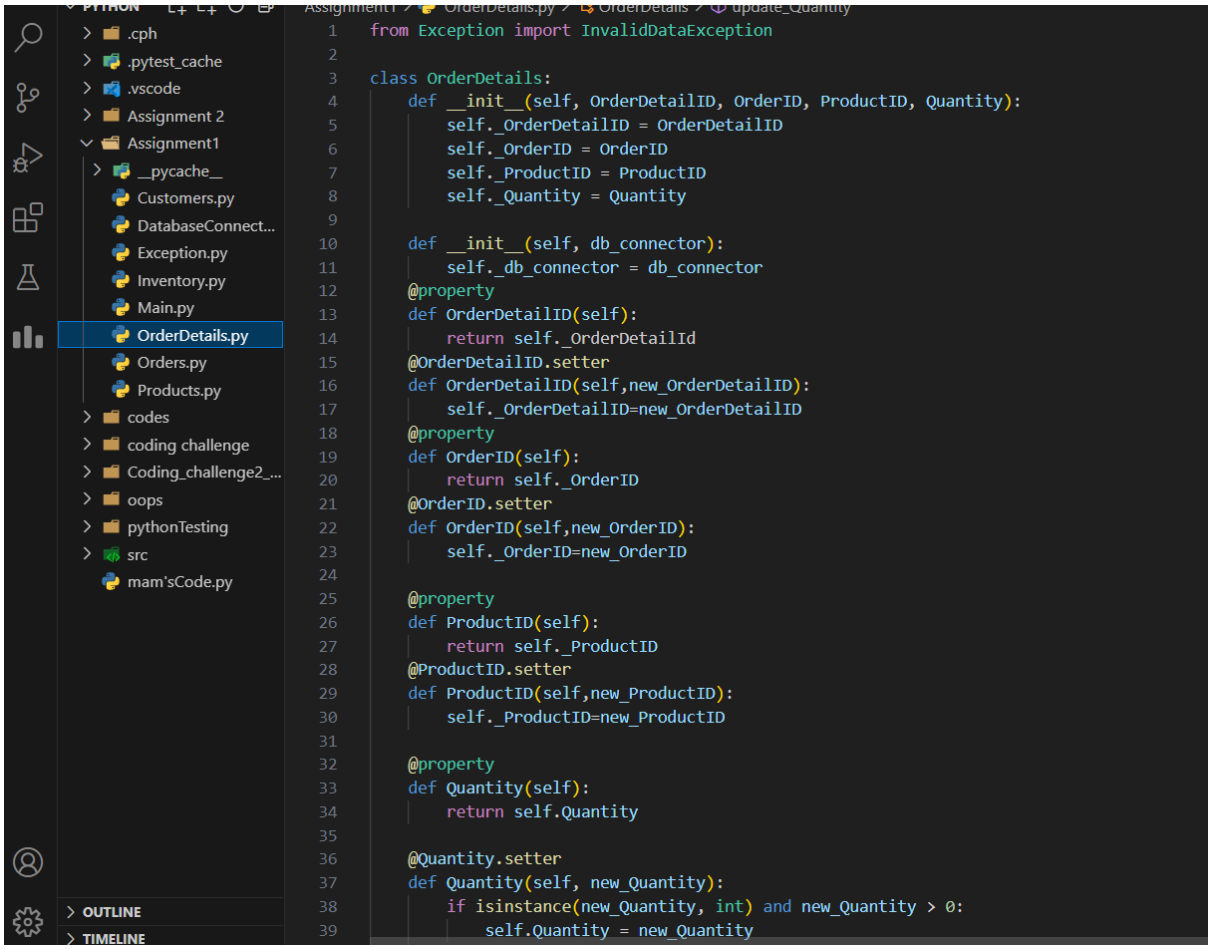
Methods:

• CalculateSubtotal() - Calculate the subtotal for this order detail.

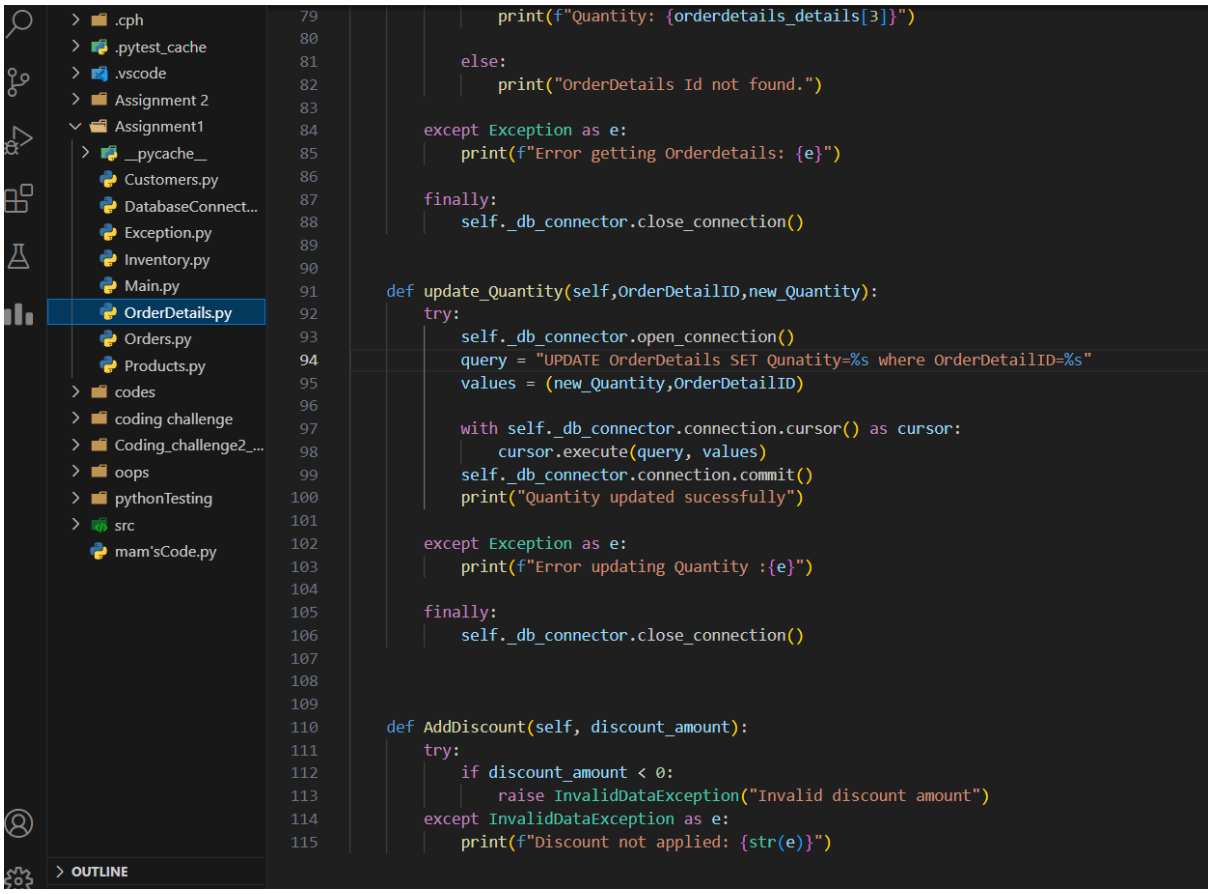
• GetOrderDetailInfo(): Retrieves and displays information about this order detail.

• UpdateQuantity(): Allows updating the quantity of the product in this order detail.

• AddDiscount(): Applies a discount to this order detail.




```python
    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]}")
```




```python
                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)}")
```

```
Scratches and Consoles
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
```

```
Run:  Main
/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.

```python
from Exception import InvalidDataException,InsufficientStockException
from datetime import datetime
class Inventory:
    def __init__(self, InventoryID, ProductID, QuantityInStock, LastStockUpdate):
        self._InventoryID = InventoryID
        self._ProductID = ProductID
        self._QuantityInStock = QuantityInStock
        self._LastStockUpdate = LastStockUpdate

    def __init__(self,db_connector):
        self._db_connector = db_connector

    @property
    def InventoryID(self):
        return self._InventoryID
    @InventoryID.setter
    def InventoryID(self,new_InventoryID):
        self._InventoryID=new_InventoryID
    @property
    def ProductID(self):
        return self._ProductID
    @ProductID.setter
    def ProductID(self,new_ProductID):
        self._ProductID=new_ProductID
    @property
    def QuantityInStock(self):
        return self._QuantityInStock
    @QuantityInStock.setter
    def QuantityInStock(self,new_QuantityInStock):
        if new_QuantityInStock >= 0:
```

```python
    def QuantityInStock(self,new_QuantityInStock):
        if new_QuantityInStock >= 0:
            self._QuantityInStock = new_QuantityInStock
        else:
            raise InvalidDataException("Quantity in stock must be non-negative")
    @property
    def LastStockUpdate(self):
        return self._LastStockUpdate
    @LastStockUpdate.setter
    def LastStockUpdate(self,new_LastStockUpdate):
        self._LastStockUpdate=new_LastStockUpdate

    def Get_Product(self,InventoryID):
        try:
            self._db_connector.open_connection()
            query = ("SELECT P.ProductID, P.ProductName, P.Description, P.Price FROM Products P"
                     " INNER JOIN Inventory I ON P.ProductID = I.ProductID WHERE I.InventoryID = %s")
            values = (InventoryID,)

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

            if product_data:
                product_id, product_name, description, price = product_data
                print(f"Product ID: {product_id}, Product Name: {product_name}, Description: {description}, Price: rs{price}")
            else:
                print("Product not found in the inventory.")

        except Exception as e:
```

```python
    def Is_Product_Available(self, quantity_to_check, InventoryID):
        try:
            self._db_connector.open_connection()
            query = "SELECT QuantityInStock FROM Inventory WHERE InventoryID = %s"
            values = (InventoryID,)
            with self._db_connector.cursor as cursor:
                cursor.execute(query, values)
                current_quantity = cursor.fetchone()
            if current_quantity is not None and current_quantity[0] >= quantity_to_check:
                print(f"Product is available in sufficient quantity: {current_quantity[0]} units.")
                return True
            else:
                print("Product is not available in sufficient quantity.")
                return False
        except Exception as e:
            print(f"Error checking product availability: {e}")
            return False

        finally:
            self._db_connector.close_connection()

    def Get_InventoryValue(self):
        try:
            self._db_connector.open_connection()
            query = "SELECT P.ProductID, P.Price, I.QuantityInStock FROM Products P INNER JOIN Inventory I ON P.ProductID = I.ProductID"
            with self._db_connector.cursor as cursor:
                cursor.execute(query)
                products_data = cursor.fetchall()
            if products_data:
                total_value = 0
```

```python
    def List_LowStock_Products(self,threshold):
        try:
            self._db_connector.open_connection()
            query = "SELECT ProductID, QuantityInStock FROM Inventory WHERE QuantityInStock < %s"
            values = (threshold,)
            with self._db_connector.cursor as cursor:
                cursor.execute(query, values)
                low_stock_products = cursor.fetchall()
            if low_stock_products:
                print("Low stock products:")
                for product in low_stock_products:
                    print(f"ProductID: {product[0]}, QuantityInStock: {product[1]}")
            else:
                print("No products with quantities below the specified threshold.")

        except Exception as e:
            print(f"Error listing low stock products: {e}")

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

```python
inventory_manager=Inventory(db_connector)
inventory_manager.Get_Product(8)
```

Run: Main

```
/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
```

```python
inventory_manager.remove_From_Inventory(2,1)

# inventory_manager.update_Stock_Quantity(2,301)
```

```
Run: Main
/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
```

```python
inventory_manager.Is_Product_Available(2,3)
```

```
Run: Main
/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
```

```python
#inventory_manager.get_QuantityInStock(1)

inventory_manager.add_To_Inventory(2,301)
```

```
Run: Main
/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

```python
inventory_manager.update_Stock_Quantity(2,301)

# inventory_manager.Is_Product_Available(2,3)
```

```
Run: Main
/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