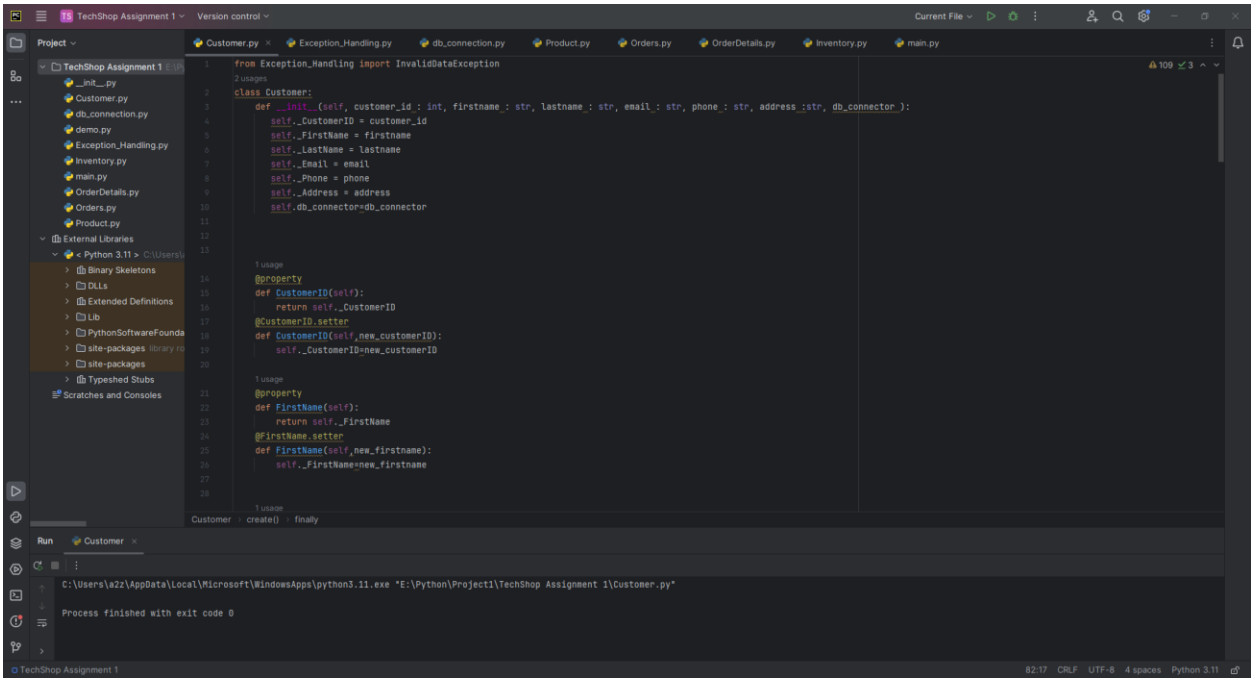
Name: Manas Rustagi

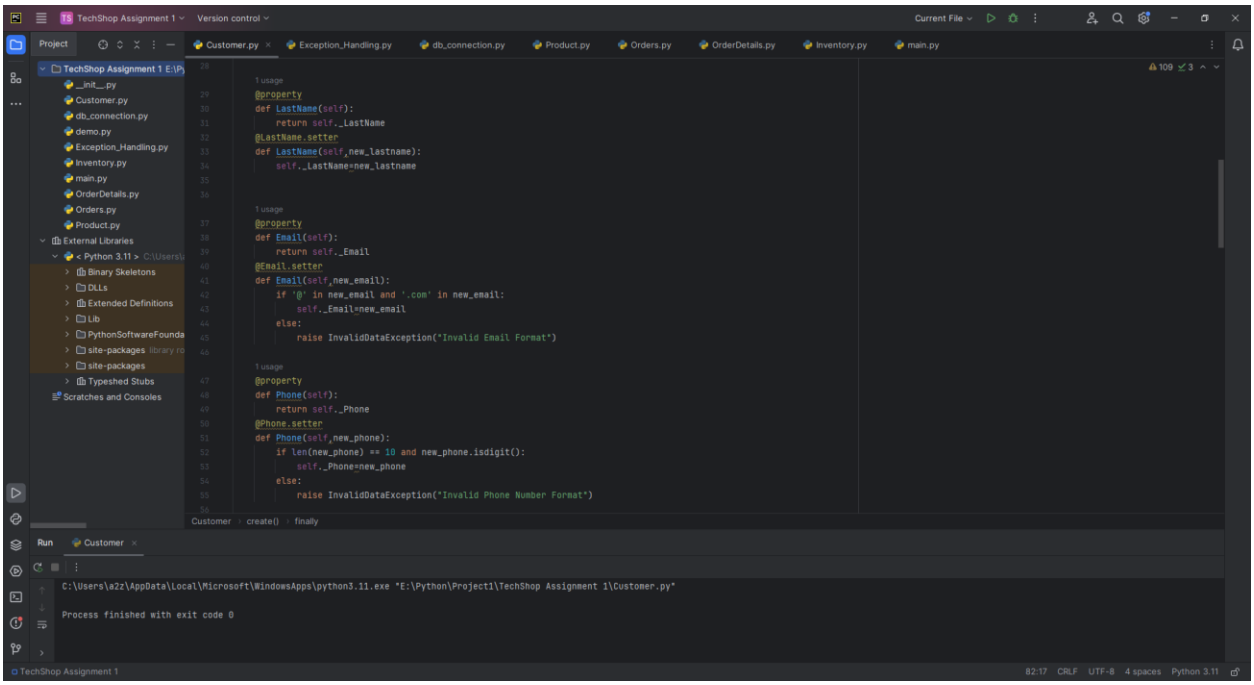
Python Assignment 1 TechShop

Customer Class :



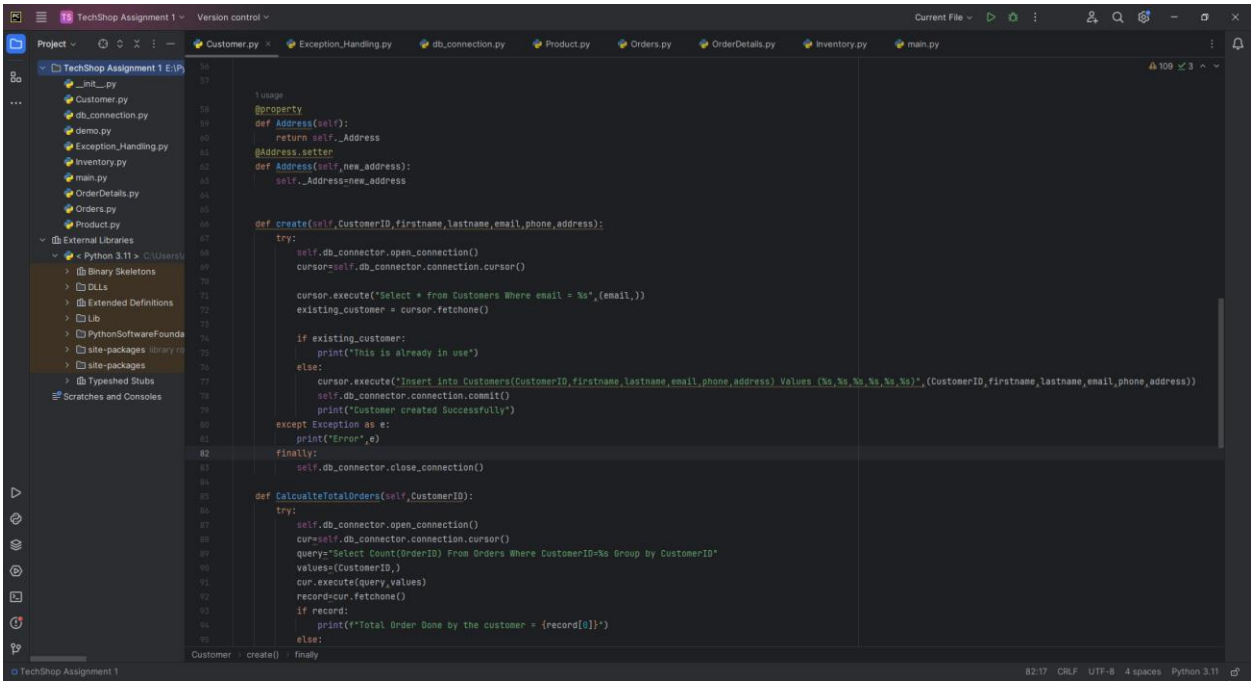
```
from Exception_Handling import InvalidDataException
2 usages
class Customer:
    def __init__(self, customer_id_: int, firstname_: str, lastname_: str, email_: str, phone_: str, address_:str, db_connector_):
        self._CustomerID = customer_id
        self._FirstName = firstname
        self._LastName = lastname
        self._Email = email
        self._Phone = phone
        self._Address = address
        self.db_connector=db_connector



    1 usage
    @property
    def CustomerID(self):
        return self._CustomerID
    @CustomerID.setter
    def CustomerID(self,new_customerID):
        self._CustomerID=new_customerID

    1 usage
    @property
    def FirstName(self):
        return self._FirstName
    @FirstName.setter
    def FirstName(self,new_firstname):
        self._FirstName=new_firstname


    1 usage
```

```
C:\Users\a2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\Customer.py"

Process finished with exit code 0
```

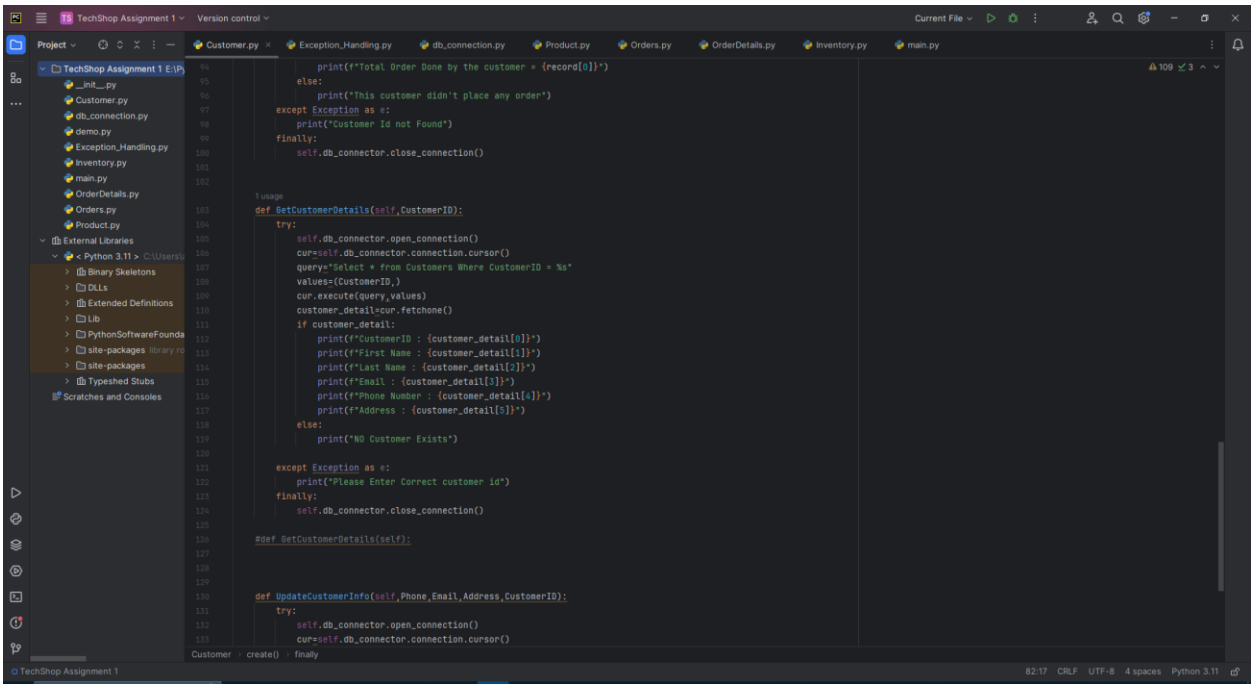


```
    1 usage
    @property
    def LastName(self):
        return self._LastName
    @LastName.setter
    def LastName(self,new_lastname):
        self._LastName=new_lastname



    1 usage
    @property
    def Email(self):
        return self._Email
    @Email.setter
    def Email(self,new_email):
        if '@' in new_email and '.com' in new_email:
            self._Email=new_email
        else:
            raise InvalidDataException("Invalid Email Format")

    1 usage
    @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")
```

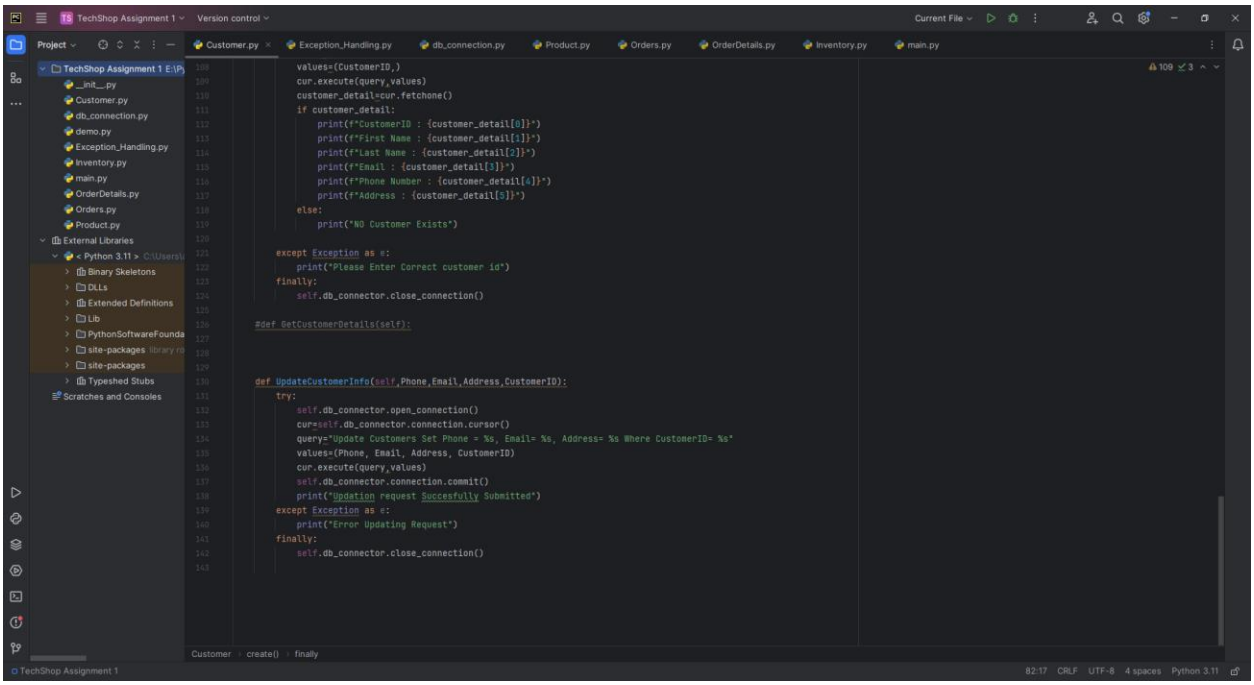
```
C:\Users\a2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\Customer.py"

Process finished with exit code 0
```



```
    @property
    def Address(self):
        return self._Address
    @Address.setter
    def Address(self,new_address):
        self._Address=new_address


    def create(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("This is already in use")
            else:
                cursor.execute("Insert into Customers(CustomerID,firstname,lastname,email,phone,address) Values (%s,%s,%s,%s,%s,%s)",(CustomerID,firstname,lastname,email,phone,address))
                self.db_connector.connection.commit()
                print("Customer created Successfully")
        except Exception as e:
            print("Error",e)
        finally:
            self.db_connector.close_connection()

    def CalcualteTotalOrders(self,CustomerID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select Count(OrderID) From Orders Where CustomerID=%s Group by CustomerID"
            values=(CustomerID,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"Total Order Done by the customer = {record[0]}")
            else:
```

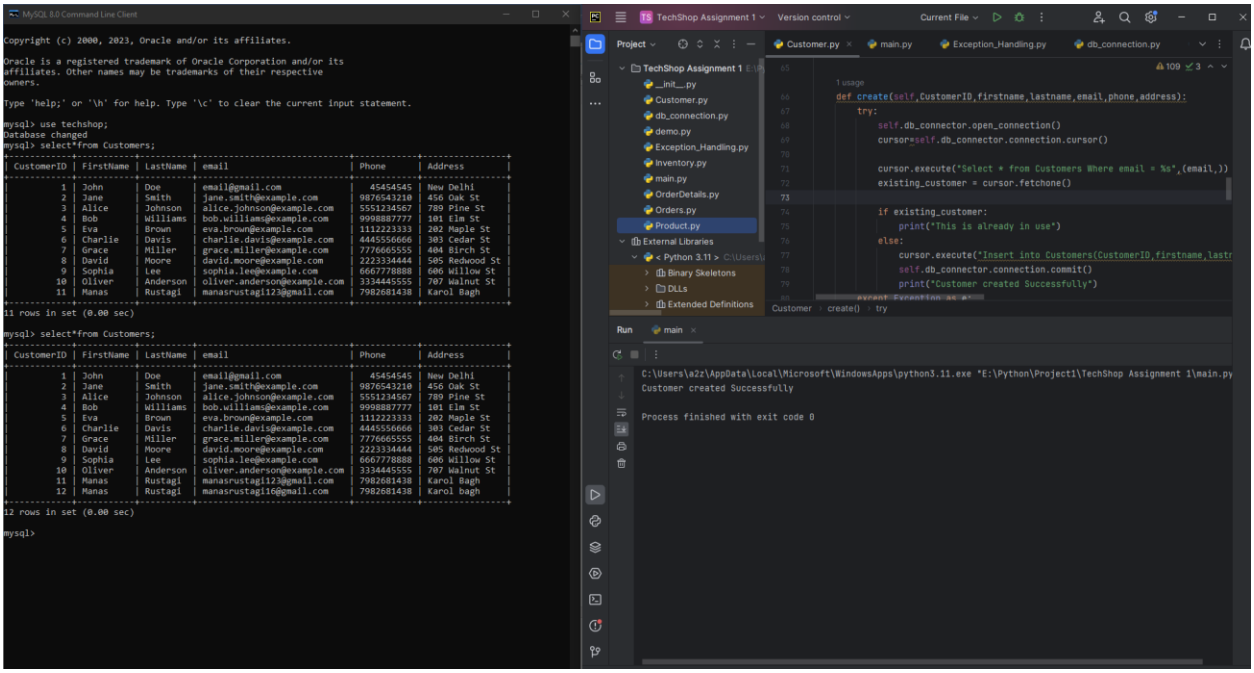


```
                print(f"Total Order Done by the customer = {record[0]}")
            else:
                print("This customer didn't place any order")
        except Exception as e:
            print("Customer Id not Found")
        finally:
            self.db_connector.close_connection()


    1 usage
    def GetCustomerDetails(self,CustomerID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select * from Customers Where CustomerID = %s"
            values=(CustomerID,)
            cur.execute(query,values)
            customer_detail=cur.fetchone()
            if customer_detail:
                print(f"CustomerID : {customer_detail[0]}")
                print(f"First Name : {customer_detail[1]}")
                print(f"Last Name : {customer_detail[2]}")
                print(f"Email : {customer_detail[3]}")
                print(f"Phone Number : {customer_detail[4]}")
                print(f"Address : {customer_detail[5]}")
            else:
                print("NO Customer Exists")

        except Exception as e:
            print("Please Enter Correct customer id")
        finally:
            self.db_connector.close_connection()

    #def GetCustomerDetails(self):



    def UpdateCustomerInfo(self,Phone,Email,Address,CustomerID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
```



```
            values=(CustomerID,)
            cur.execute(query,values)
            customer_detail=cur.fetchone()
            if customer_detail:
                print(f"CustomerID : {customer_detail[0]}")
                print(f"First Name : {customer_detail[1]}")
                print(f"Last Name : {customer_detail[2]}")
                print(f"Email : {customer_detail[3]}")
                print(f"Phone Number : {customer_detail[4]}")
                print(f"Address : {customer_detail[5]}")
            else:
                print("NO Customer Exists")

        except Exception as e:
            print("Please Enter Correct customer id")
        finally:
            self.db_connector.close_connection()

    #def GetCustomerDetails(self):


    def UpdateCustomerInfo(self,Phone,Email,Address,CustomerID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Update Customers Set Phone = %s, Email= %s, Address= %s Where CustomerID= %s"
            values=(Phone, Email, Address, CustomerID)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print("Updation request Succesfully Submitted")
        except Exception as e:
            print("Error Updating Request")
        finally:
            self.db_connector.close_connection()
```

Customer created Successfully(CustomerID=12)

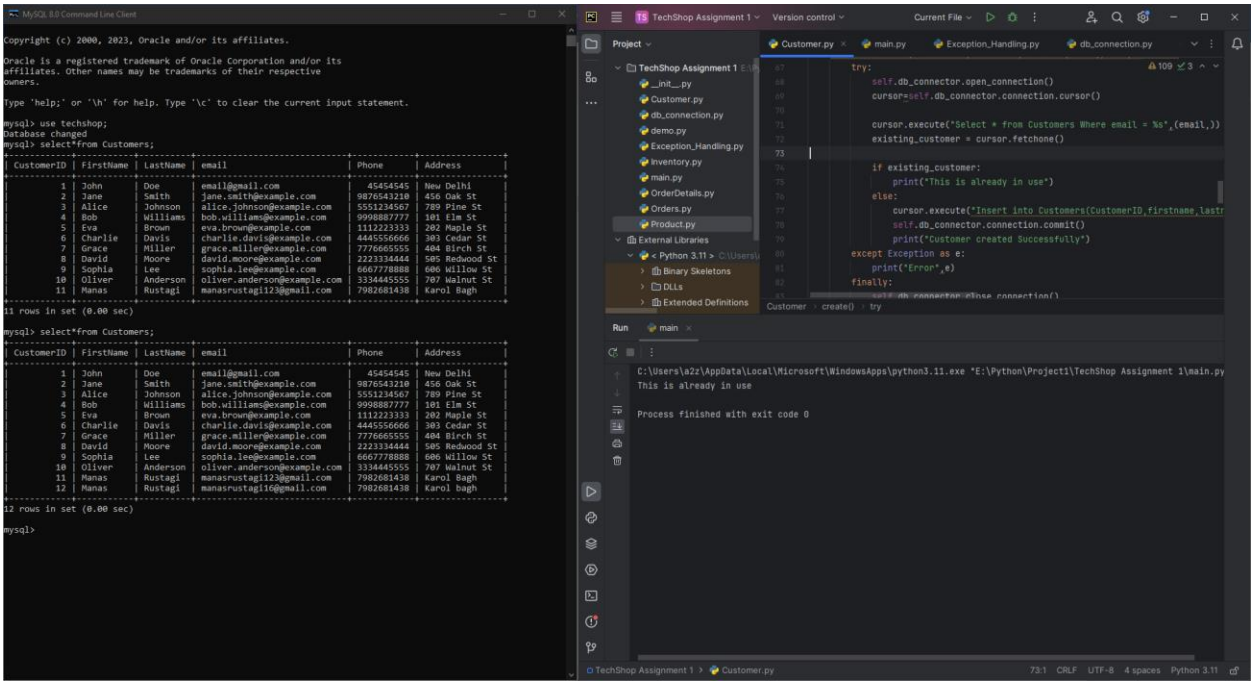


```
Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use techshop;
Database changed
mysql> select*from Customers;
+------------+-----------+----------+--------------------------+------------+---------------+
| CustomerID | FirstName | LastName | email                    | Phone      | Address       |
+------------+-----------+----------+--------------------------+------------+---------------+
|          1 | John      | Doe      | email@gmail.com          |   45454545 | New Delhi     |
|          2 | Jane      | Smith    | jane.smith@example.com   | 9876543210 | 456 Oak St    |
|          3 | Alice     | Johnson  | alice.johnson@example.com| 5551234567 | 789 Pine St   |
|          4 | Bob       | Williams | bob.williams@example.com | 9998887777 | 101 Elm St    |
|          5 | Eva       | Brown    | eva.brown@example.com    | 1112223333 | 202 Maple St  |
|          6 | Charlie   | Davis    | charlie.davis@example.com| 4445556666 | 303 Cedar St  |
|          7 | Grace     | Miller   | grace.miller@example.com | 7776665555 | 404 Birch St  |
|          8 | David     | Moore    | david.moore@example.com  | 2223334444 | 505 Redwood St|
|          9 | Sophia    | Lee      | sophia.lee@example.com   | 6667778888 | 606 Willow St |
|         10 | Oliver    | Anderson | oliver.anderson@example.com | 3334445555 | 707 Walnut St |
|         11 | Manas     | Rustagi  | manasrustagi123@gmail.com| 7982681438 | Karol Bagh    |
+------------+-----------+----------+--------------------------+------------+---------------+
11 rows in set (0.00 sec)

mysql> select*from Customers;
+------------+-----------+----------+--------------------------+------------+---------------+
| CustomerID | FirstName | LastName | email                    | Phone      | Address       |
+------------+-----------+----------+--------------------------+------------+---------------+
|          1 | John      | Doe      | email@gmail.com          |   45454545 | New Delhi     |
|          2 | Jane      | Smith    | jane.smith@example.com   | 9876543210 | 456 Oak St    |
|          3 | Alice     | Johnson  | alice.johnson@example.com| 5551234567 | 789 Pine St   |
|          4 | Bob       | Williams | bob.williams@example.com | 9998887777 | 101 Elm St    |
|          5 | Eva       | Brown    | eva.brown@example.com    | 1112223333 | 202 Maple St  |
|          6 | Charlie   | Davis    | charlie.davis@example.com| 4445556666 | 303 Cedar St  |
|          7 | Grace     | Miller   | grace.miller@example.com | 7776665555 | 404 Birch St  |
|          8 | David     | Moore    | david.moore@example.com  | 2223334444 | 505 Redwood St|
|          9 | Sophia    | Lee      | sophia.lee@example.com   | 6667778888 | 606 Willow St |
|         10 | Oliver    | Anderson | oliver.anderson@example.com | 3334445555 | 707 Walnut St |
|         11 | Manas     | Rustagi  | manasrustagi123@gmail.com| 7982681438 | Karol Bagh    |
|         12 | Manas     | Rustagi  | manasrustagi16@gmail.com | 7982681438 | Karol bagh    |
+------------+-----------+----------+--------------------------+------------+---------------+
12 rows in set (0.00 sec)

mysql>
```

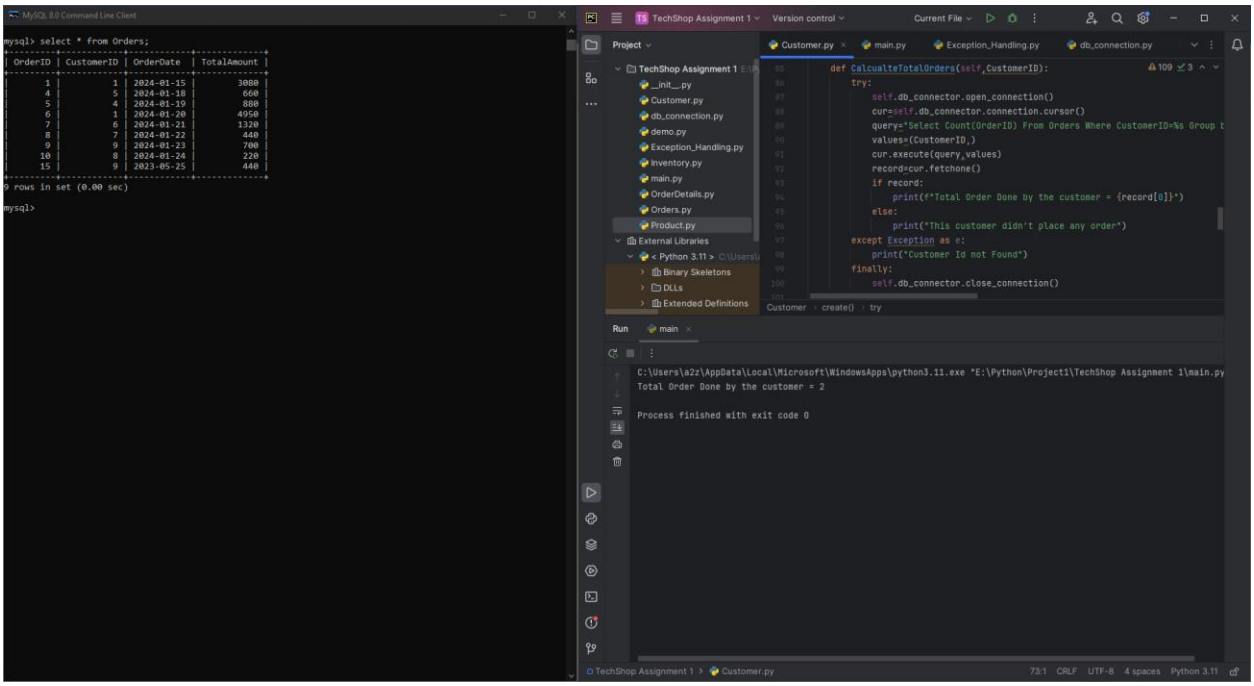
```
    def create(self,CustomerID,firstname,lastname,email,phone,address):
        try:
            self.db_connector.open_connection()
            cursor=self.db_connector.connection.cursor()

            cursor.execute("Select * from Customers Where email = %s",(email,))
            existing_customer = cursor.fetchone()

            if existing_customer:
                print("This is already in use")
            else:
                cursor.execute("Insert into Customers(CustomerID,firstname,lastr
                self.db_connector.connection.commit()
                print("Customer created Successfully")
```

```
C:\Users\a2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py
Customer created Successfully

Process finished with exit code 0
```

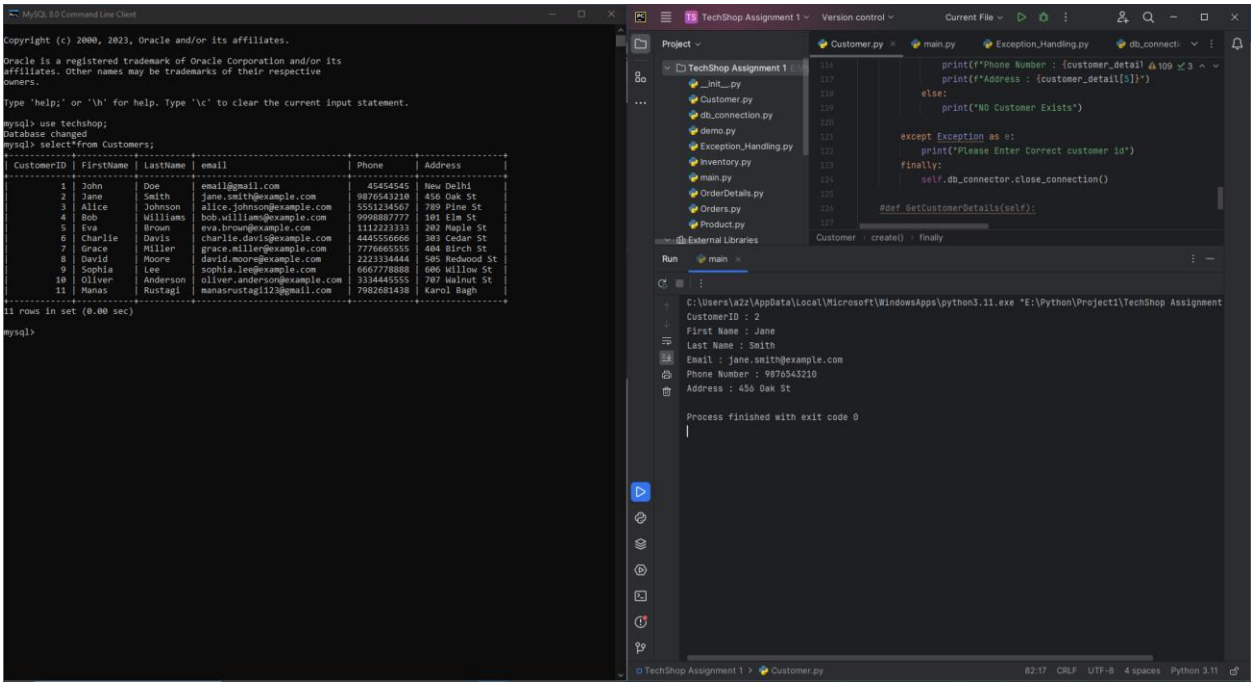
Email Id already exists (Cannot create New user)



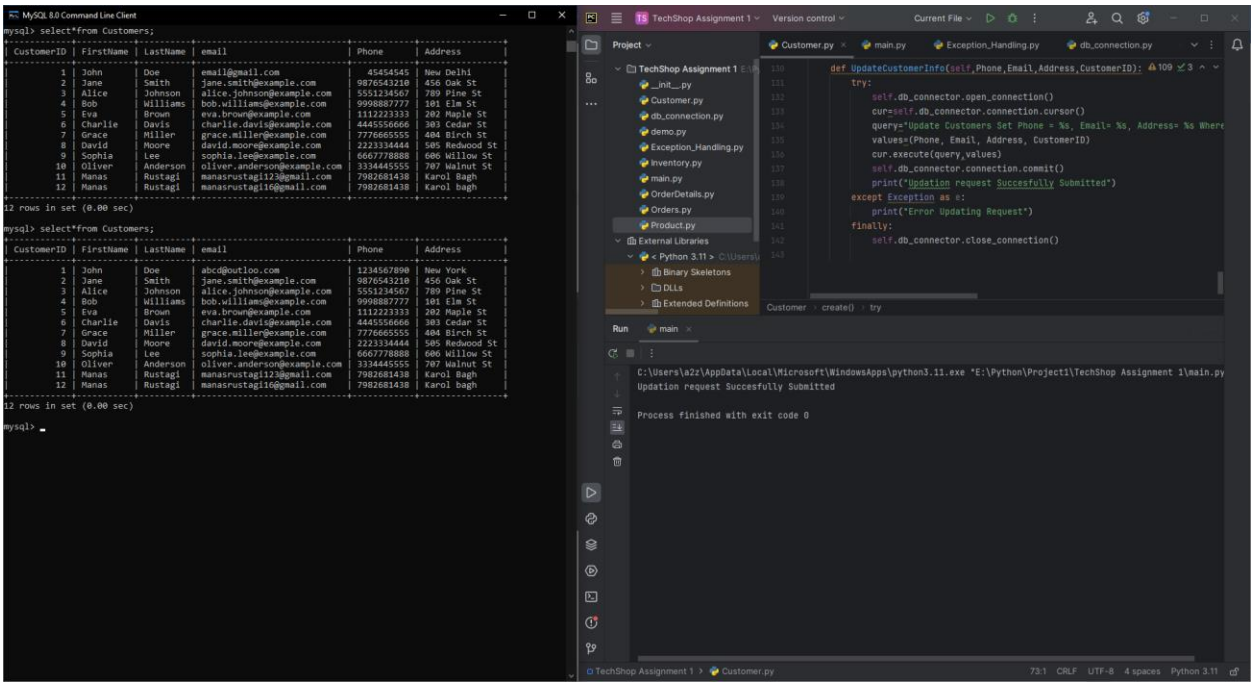
Total Orders Done by a Customer(CustomerID=1)



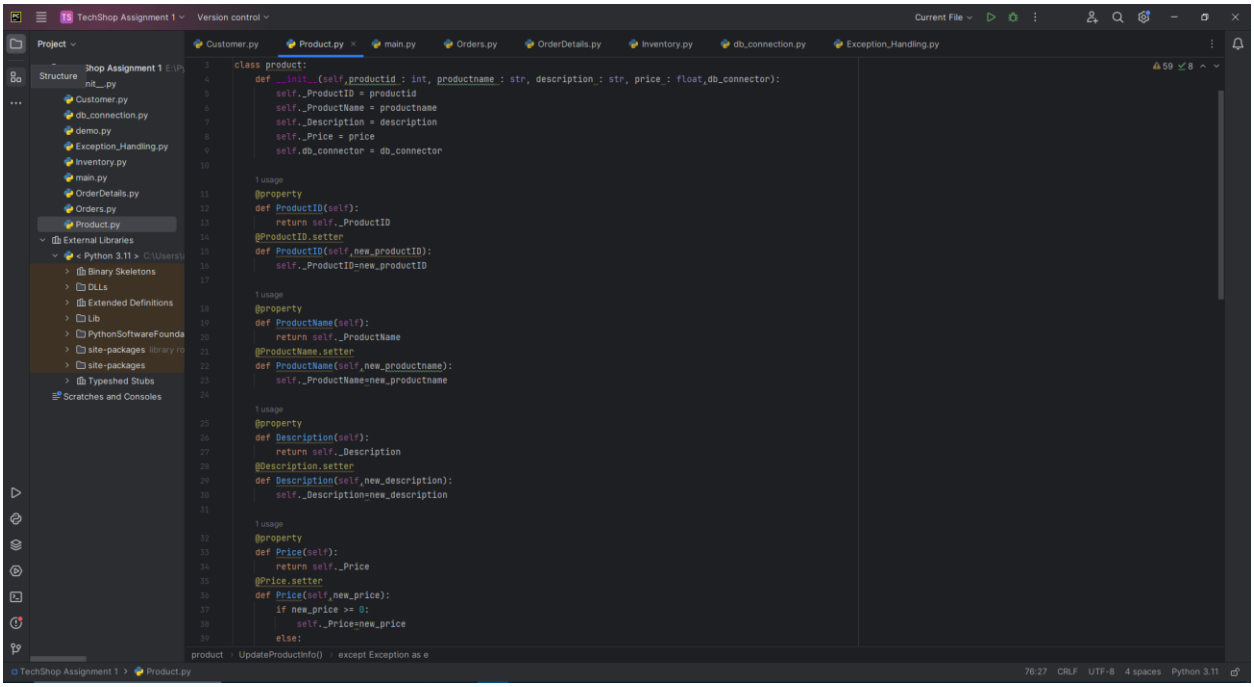
Getting Info about a Customer(CustomerID=2)



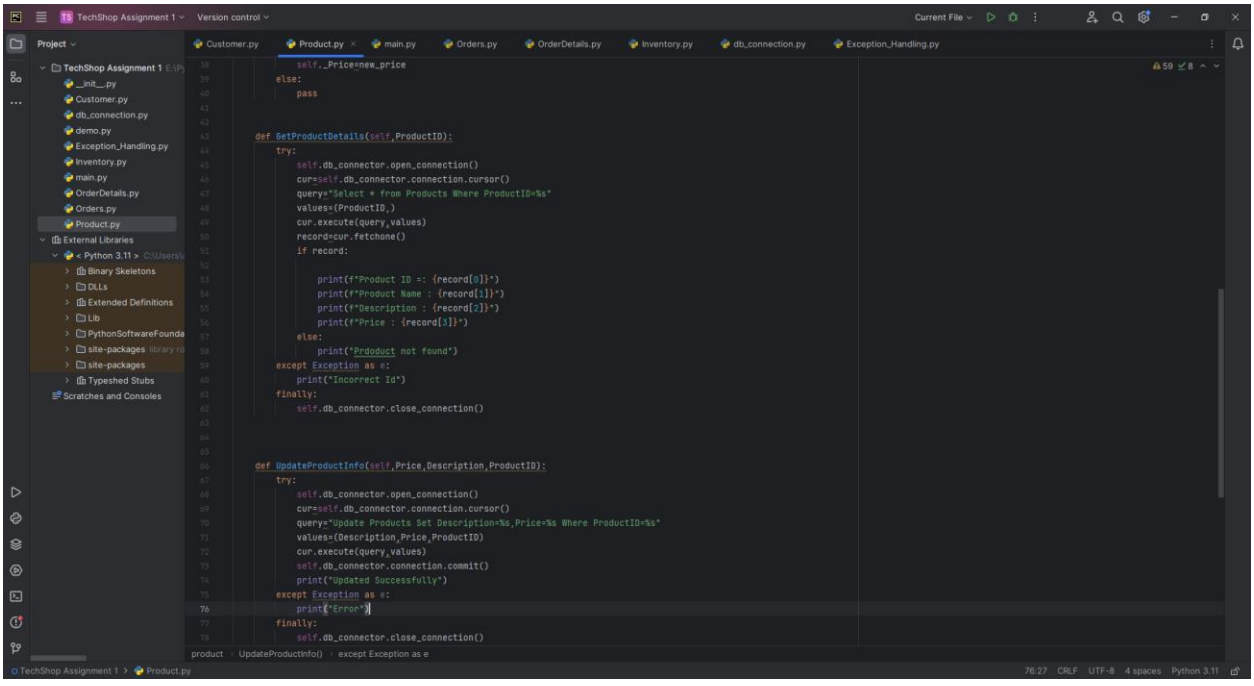
Updating Customer Info(CustomerId=2)



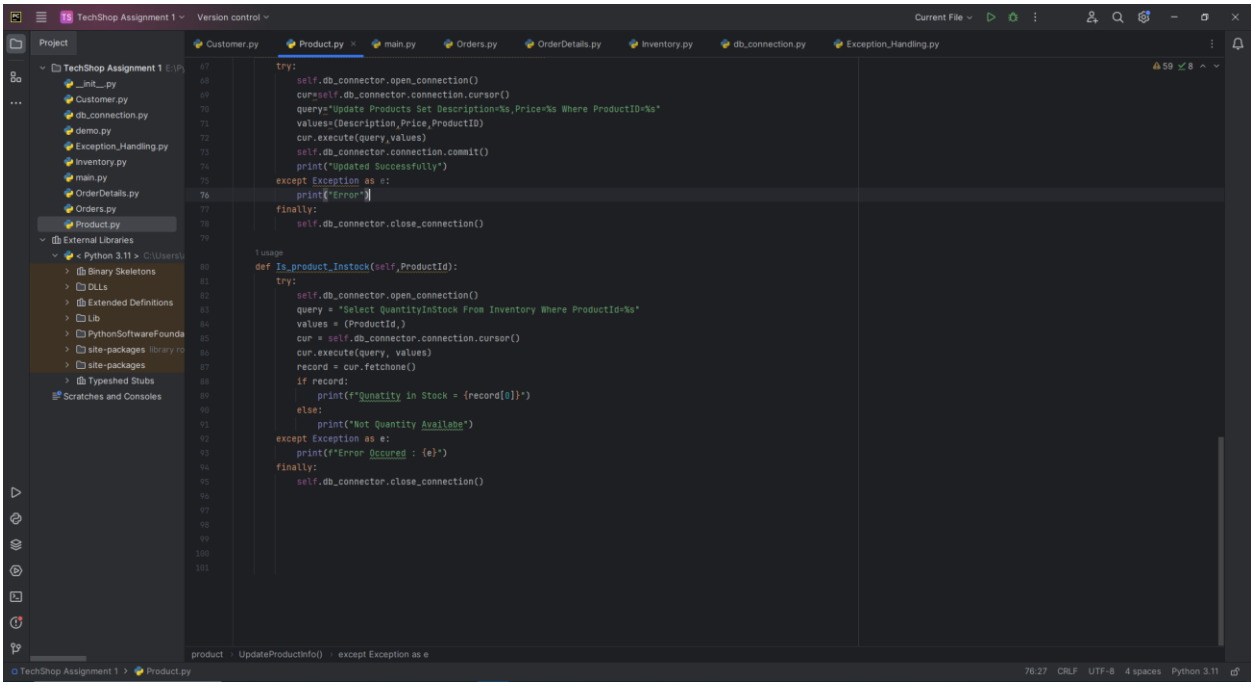
Products Class:



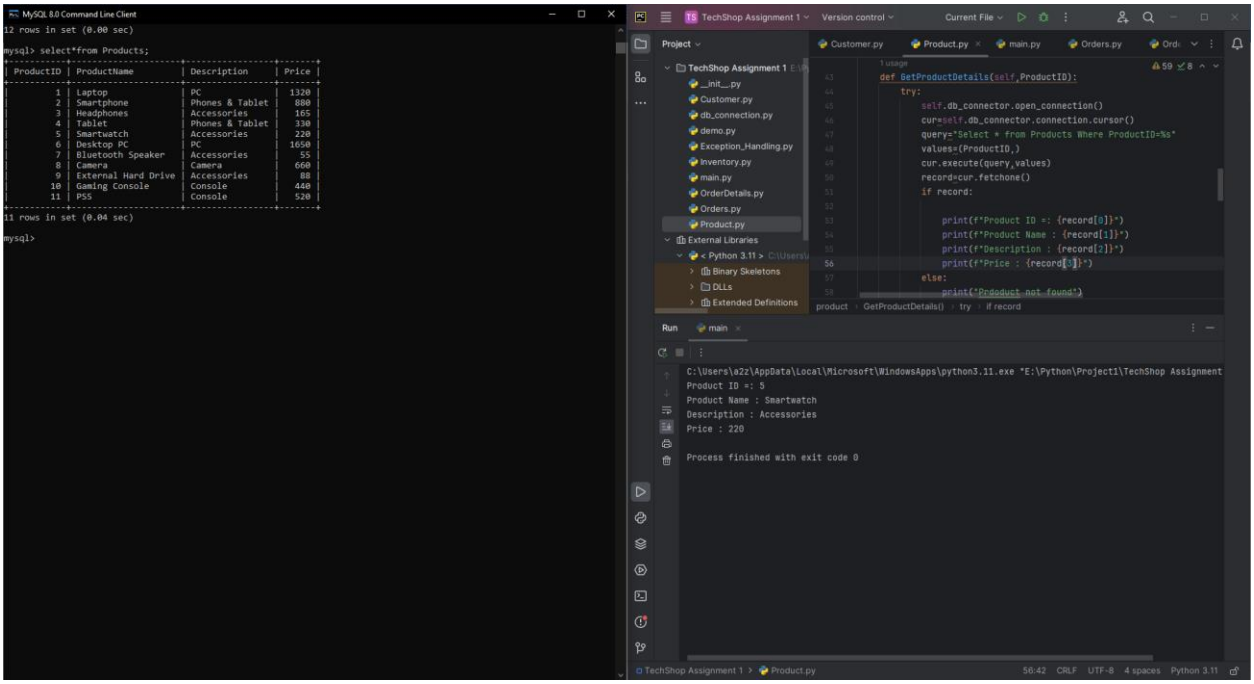
```
class product:
    def __init__(self,productid : int, productname : str, description : str, price : float,db_connector):
        self._ProductID = productid
        self._ProductName = productname
        self._Description = description
        self._Price = price
        self.db_connector = db_connector

    1 usage
    @property
    def ProductID(self):
        return self._ProductID
    @ProductID.setter
    def ProductID(self,new_productID):
        self._ProductID=new_productID

    1 usage
    @property
    def ProductName(self):
        return self._ProductName
    @ProductName.setter
    def ProductName(self,new_productname):
        self._ProductName=new_productname

    1 usage
    @property
    def Description(self):
        return self._Description
    @Description.setter
    def Description(self,new_description):
        self._Description=new_description

    1 usage
    @property
    def Price(self):
        return self._Price
    @Price.setter
    def Price(self,new_price):
        if new_price >= 0:
            self._Price=new_price
        else:
```



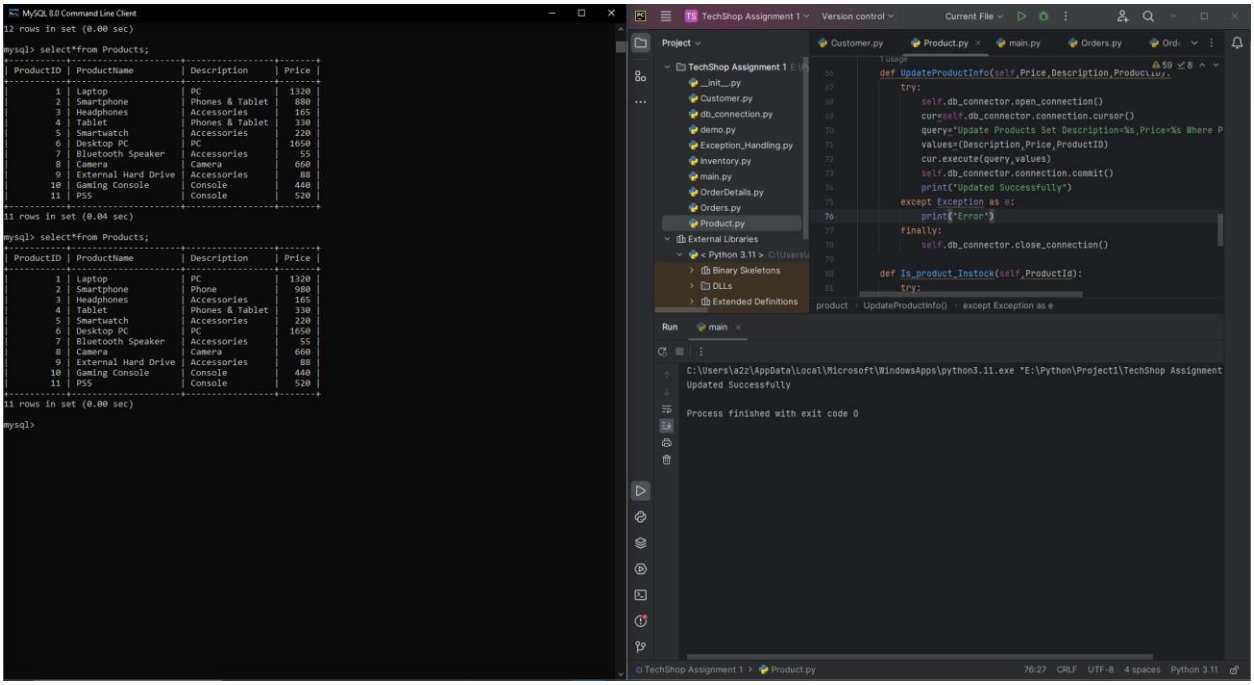
```
            self._Price=new_price
        else:
            pass


    def GetProductDetails(self,ProductID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select * from Products Where ProductID=%s"
            values=(ProductID,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:

                print(f"Product ID =: {record[0]}")
                print(f"Product Name : {record[1]}")
                print(f"Description : {record[2]}")
                print(f"Price : {record[3]}")
            else:
                print("Prdoduct not found")
        except Exception as e:
            print("Incorrect Id")
        finally:
            self.db_connector.close_connection()



    def UpdateProductInfo(self,Price,Description,ProductID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Update Products Set Description=%s,Price=%s Where ProductID=%s"
            values=(Description,Price,ProductID)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print("Updated Successfully")
        except Exception as e:
            print("Error")
        finally:
            self.db_connector.close_connection()
```



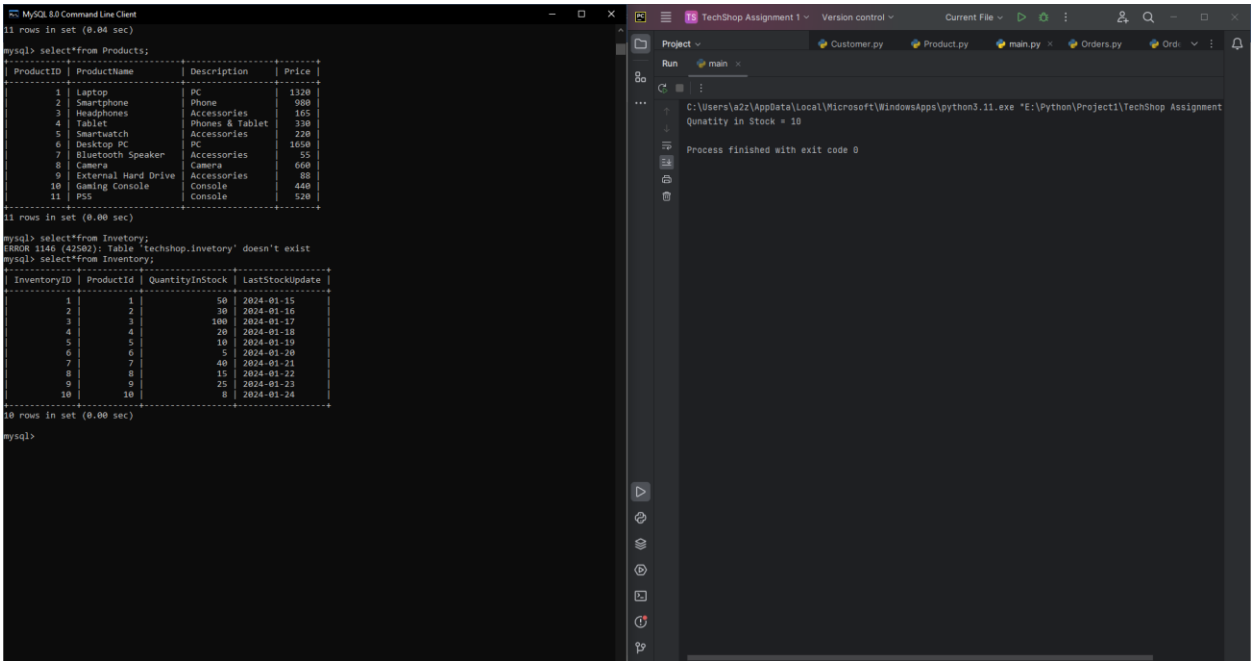
Getting Product Info(ProductID=5)



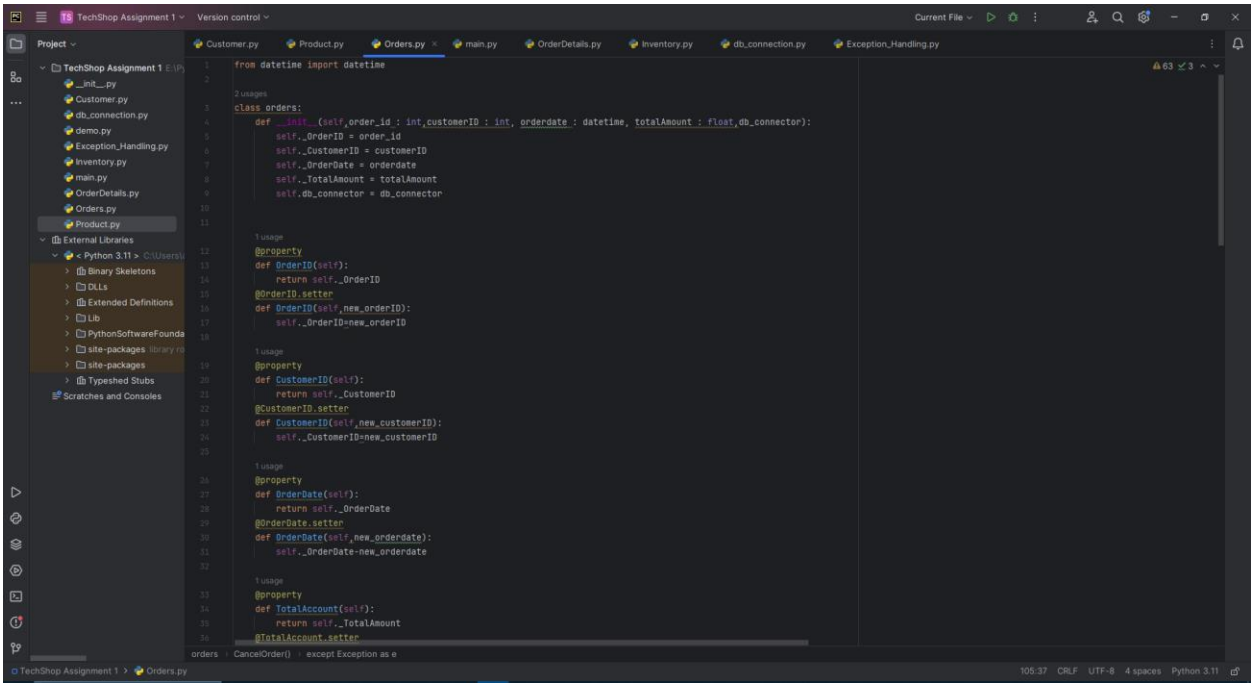
Updating Product Info(ProductId=2)



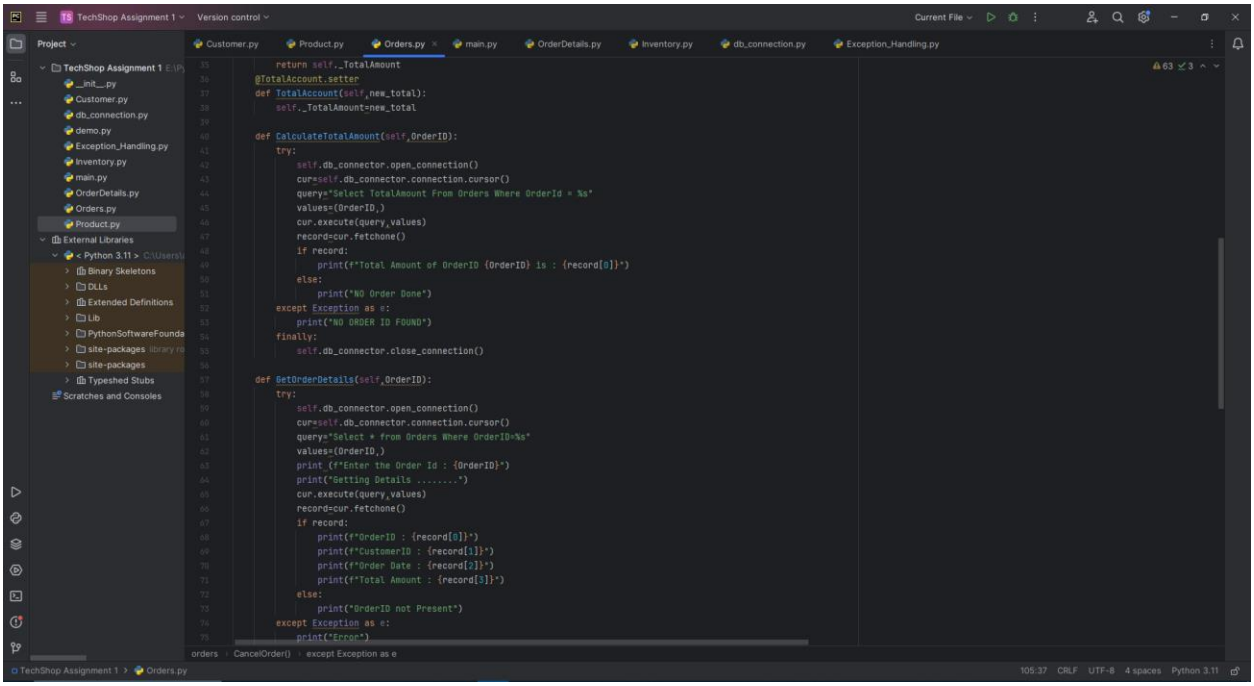
Checking if Product is available or not in the Inventory(Product=5)



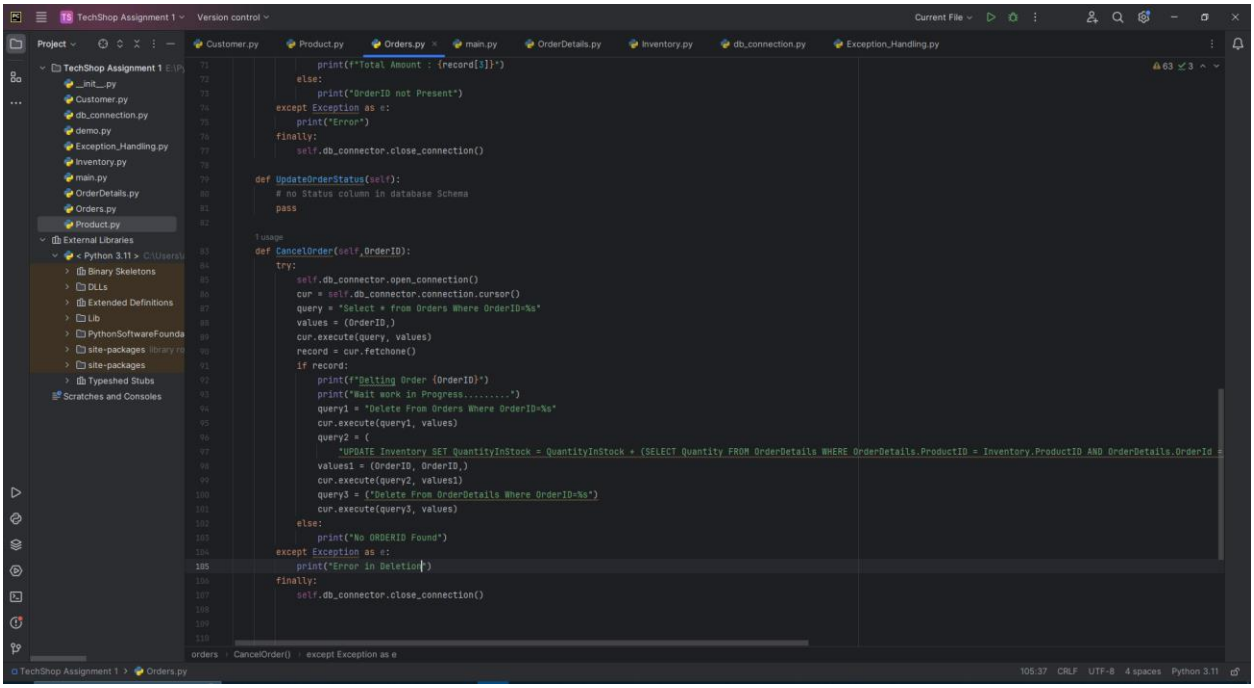
Order Class:



```python
from datetime import datetime

2 usages
class orders:
    def __init__(self,order_id : int,customerID : int, orderdate : datetime, totalAmount : float,db_connector):
        self._OrderID = order_id
        self._CustomerID = customerID
        self._OrderDate = orderdate
        self._TotalAmount = totalAmount
        self.db_connector = db_connector


    1 usage
    @property
    def OrderID(self):
        return self._OrderID
    @OrderID.setter
    def OrderID(self,new_orderID):
        self._OrderID=new_orderID

    1 usage
    @property
    def CustomerID(self):
        return self._CustomerID
    @CustomerID.setter
    def CustomerID(self,new_customerID):
        self._CustomerID=new_customerID

    1 usage
    @property
    def OrderDate(self):
        return self._OrderDate
    @OrderDate.setter
    def OrderDate(self,new_orderdate):
        self._OrderDate=new_orderdate

    1 usage
    @property
    def TotalAccount(self):
        return self._TotalAmount
    @TotalAccount.setter
```

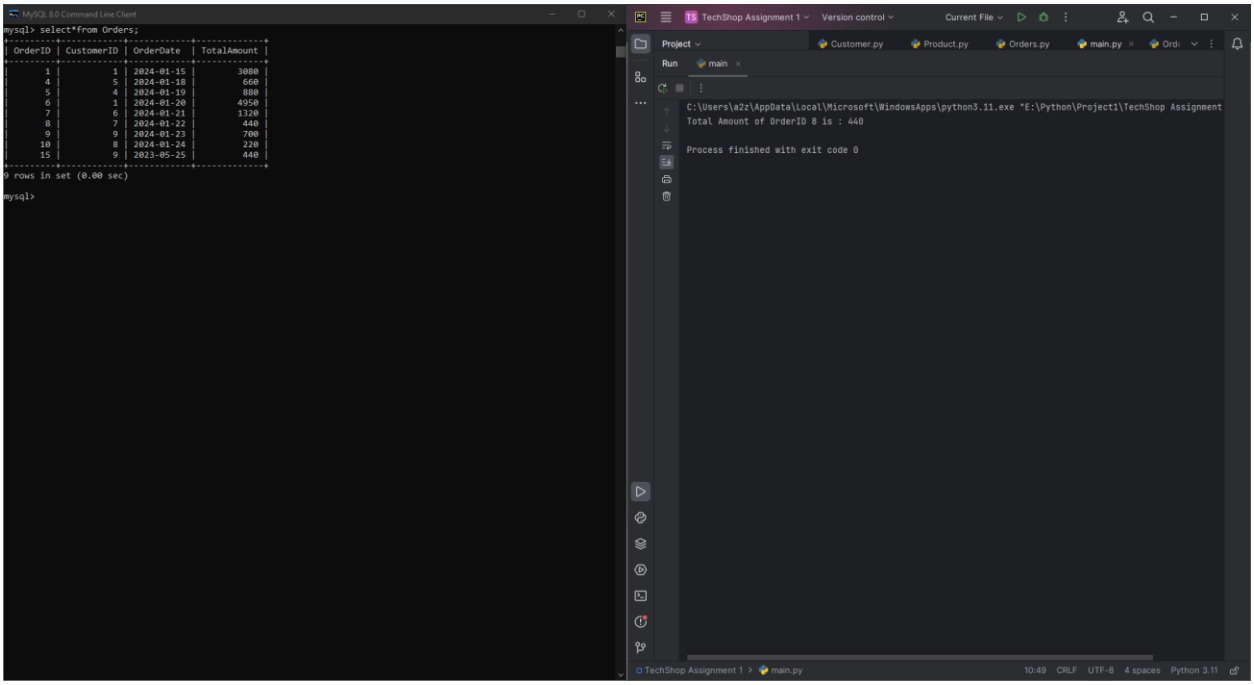


```python
        return self._TotalAmount
    @TotalAccount.setter
    def TotalAccount(self,new_total):
        self._TotalAmount=new_total

    def CalculateTotalAmount(self,OrderID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select TotalAmount From Orders Where OrderId = %s"
            values=(OrderID,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"Total Amount of OrderID {OrderID} is : {record[0]}")
            else:
                print("NO Order Done")
        except Exception as e:
            print("NO ORDER ID FOUND")
        finally:
            self.db_connector.close_connection()

    def GetOrderDetails(self,OrderID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select * from Orders Where OrderID=%s"
            values=(OrderID,)
            print (f"Enter the Order Id : {OrderID}")
            print("Getting Details .....")
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"OrderID : {record[0]}")
                print(f"CustomerID : {record[1]}")
                print(f"Order Date : {record[2]}")
                print(f"Total Amount : {record[3]}")
            else:
                print("OrderID not Present")
        except Exception as e:
            print("Error")
```



```
                print(f"Total Amount : {record[3]}")
            else:
                print("OrderID not Present")
        except Exception as e:
            print("Error")
        finally:
            self.db_connector.close_connection()

    def UpdateOrderStatus(self):
        # no Status column in database Schema
        pass

    1 usage
    def CancelOrder(self,OrderID):
        try:
            self.db_connector.open_connection()
            cur = self.db_connector.connection.cursor()
            query = "Select * from Orders Where OrderID=%s"
            values = (OrderID,)
            cur.execute(query, values)
            record = cur.fetchone()
            if record:
                print(f"Delting Order {OrderID}")
                print("Wait work in Progress.........")
                query1 = "Delete From Orders Where OrderID=%s"
                cur.execute(query1, values)
                query2 = (
                    "UPDATE Inventory SET QuantityInStock = QuantityInStock + (SELECT Quantity FROM OrderDetails WHERE OrderDetails.ProductID = Inventory.ProductID AND OrderDetails.OrderId =
                values1 = (OrderID, OrderID,)
                cur.execute(query2, values1)
                query3 = ("Delete From OrderDetails Where OrderID=%s")
                cur.execute(query3, values)
            else:
                print("No ORDERID Found")
        except Exception as e:
            print("Error in Deletion")
        finally:
            self.db_connector.close_connection()
```

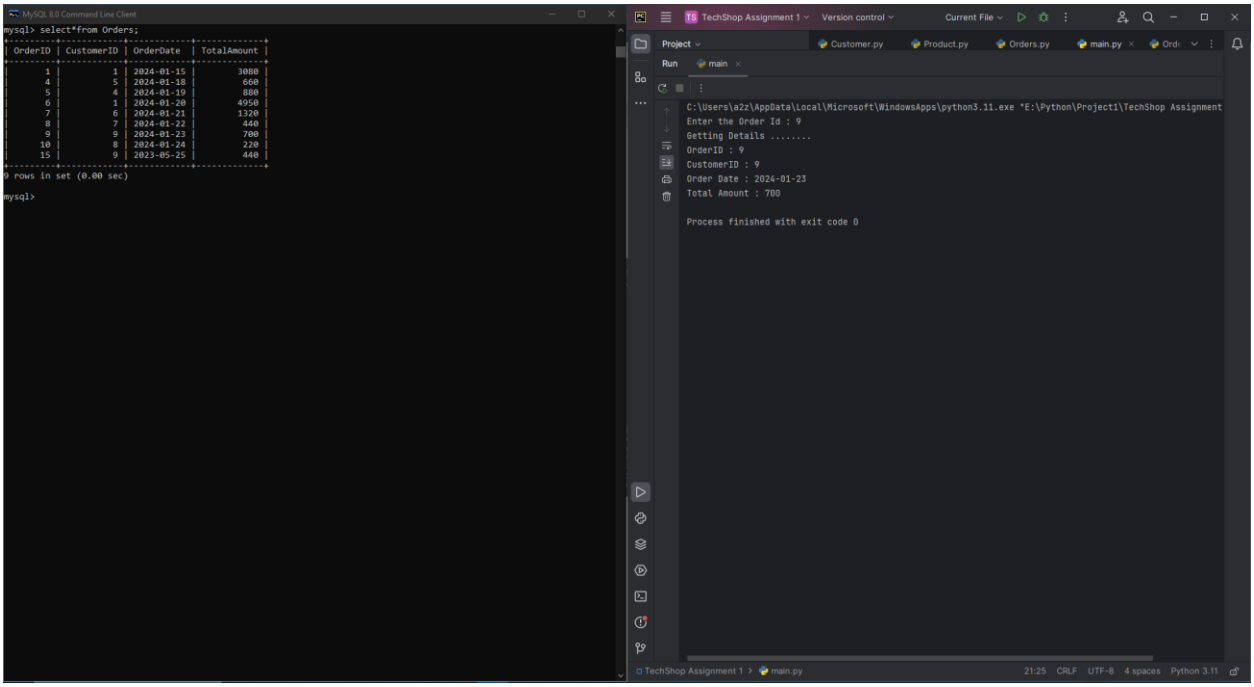
Calculating The Total Amount of a Particular Order:



```
mysql> select*from Orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|       1 |          1 | 2024-01-15 |        3080 |
|       4 |          5 | 2024-01-18 |         660 |
|       5 |          4 | 2024-01-19 |         880 |
|       6 |          1 | 2024-01-20 |        4950 |
|       7 |          6 | 2024-01-21 |        1320 |
|       8 |          7 | 2024-01-22 |         440 |
|       9 |          9 | 2024-01-23 |         700 |
|      10 |          8 | 2024-01-24 |         220 |
|      15 |          9 | 2023-05-25 |         440 |
+---------+------------+------------+-------------+
9 rows in set (0.00 sec)

mysql>
```

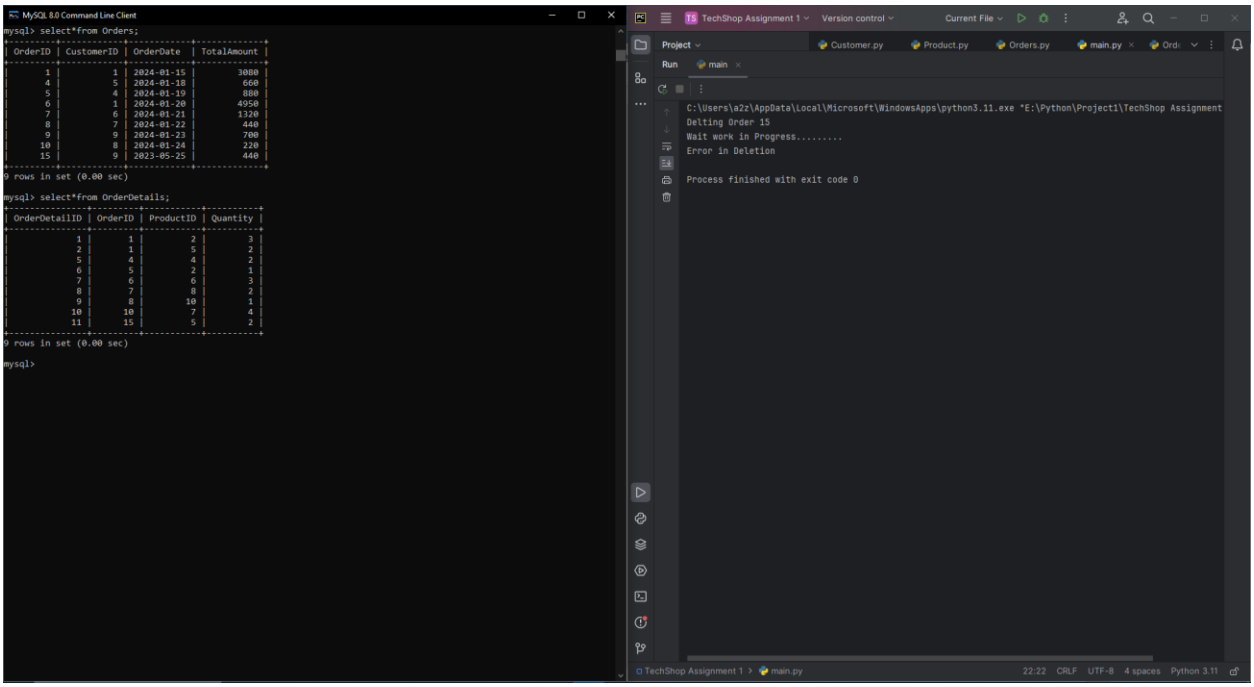
```
C:\Users\a2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment
Total Amount of OrderID 8 is : 440

Process finished with exit code 0
```

Getting info about a Particular Order

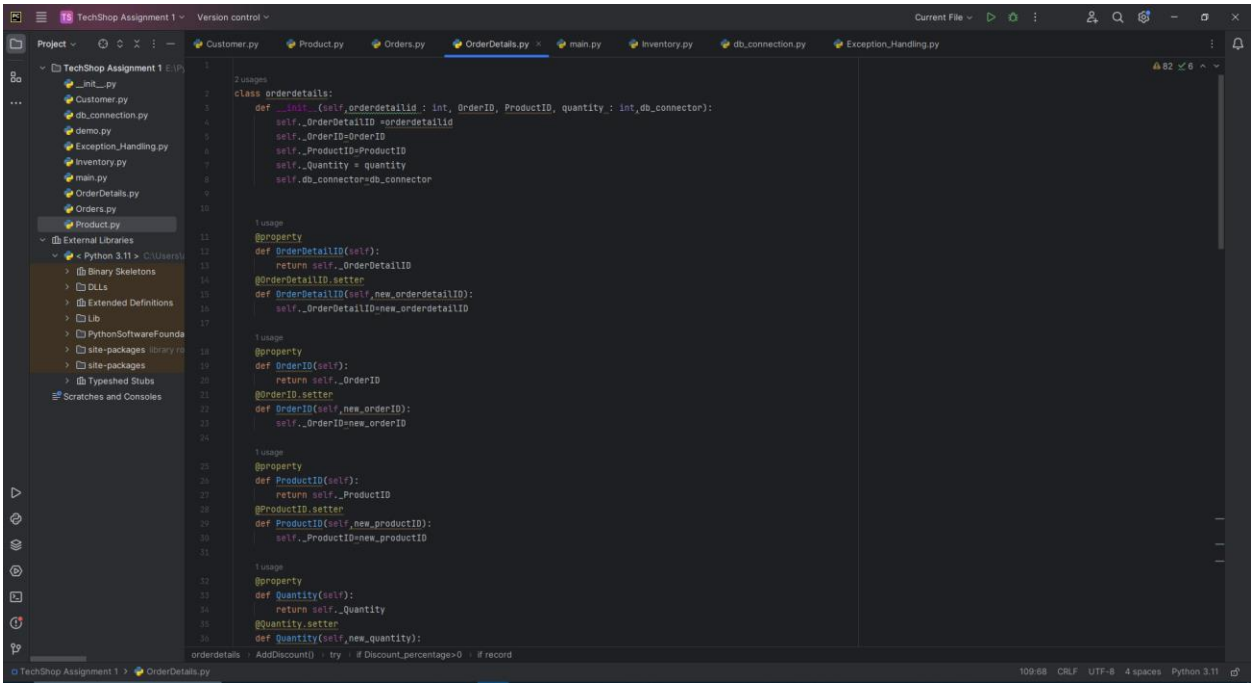


Cancelling Order

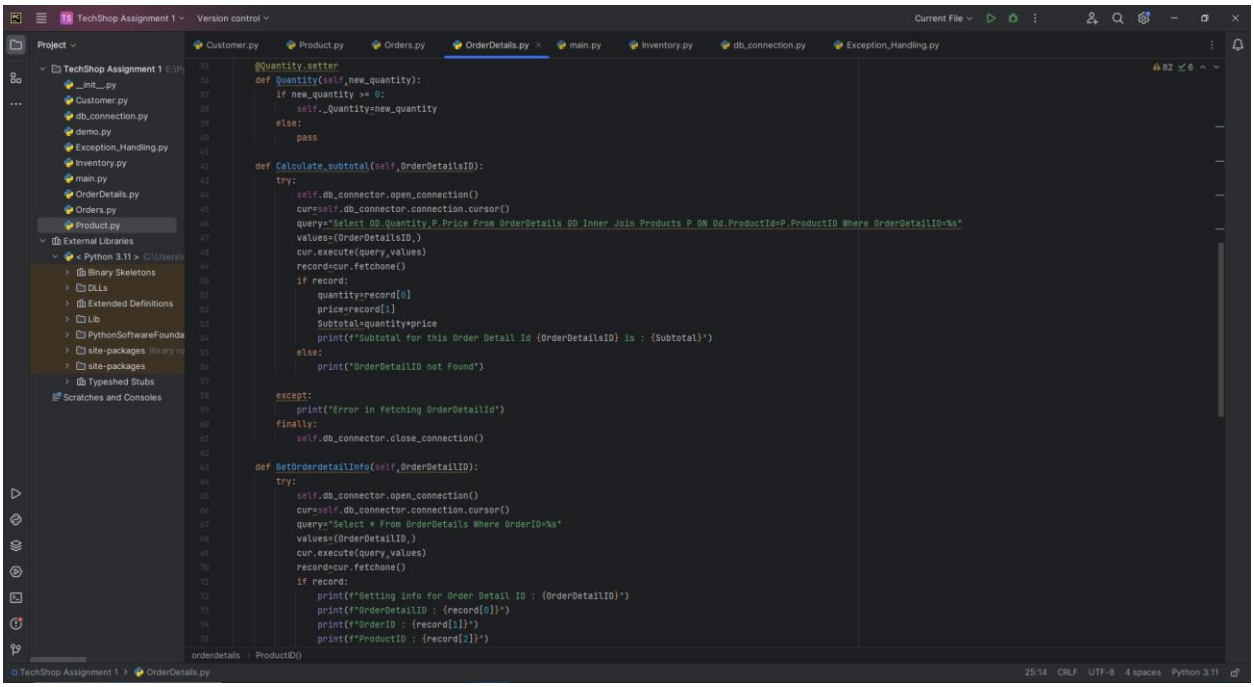
Cannot Delete Order Because we have Order Detail corresponding to that Order ID



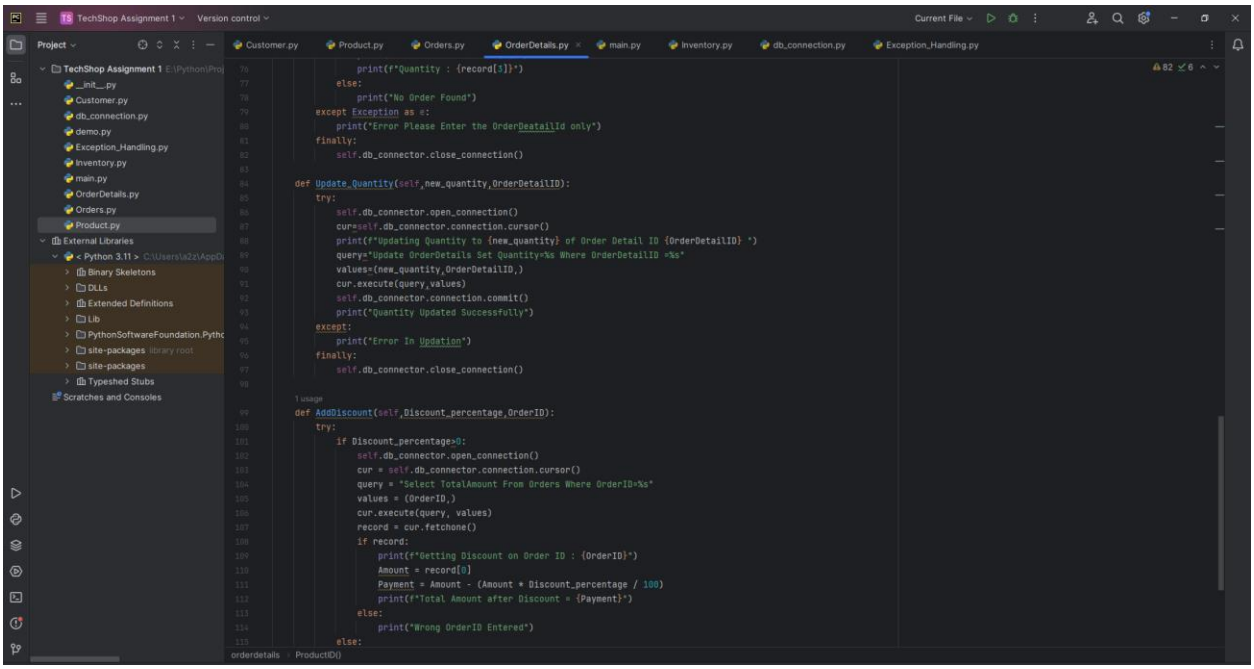
OrderDetails Class:



```
2 usages
class orderdetails:
    def __init__(self,orderdetailid : int, OrderID, ProductID, quantity : int,db_connector):
        self._OrderDetailID =orderdetailid
        self._OrderID=OrderID
        self._ProductID=ProductID
        self._Quantity = quantity
        self.db_connector=db_connector


    1 usage
    @property
    def OrderDetailID(self):
        return self._OrderDetailID
    @OrderDetailID.setter
    def OrderDetailID(self,new_orderdetailID):
        self._OrderDetailID=new_orderdetailID

    1 usage
    @property
    def OrderID(self):
        return self._OrderID
    @OrderID.setter
    def OrderID(self,new_orderID):
        self._OrderID=new_orderID

    1 usage
    @property
    def ProductID(self):
        return self._ProductID
    @ProductID.setter
    def ProductID(self,new_productID):
        self._ProductID=new_productID

    1 usage
    @property
    def Quantity(self):
        return self._Quantity
    @Quantity.setter
    def Quantity(self,new_quantity):
```



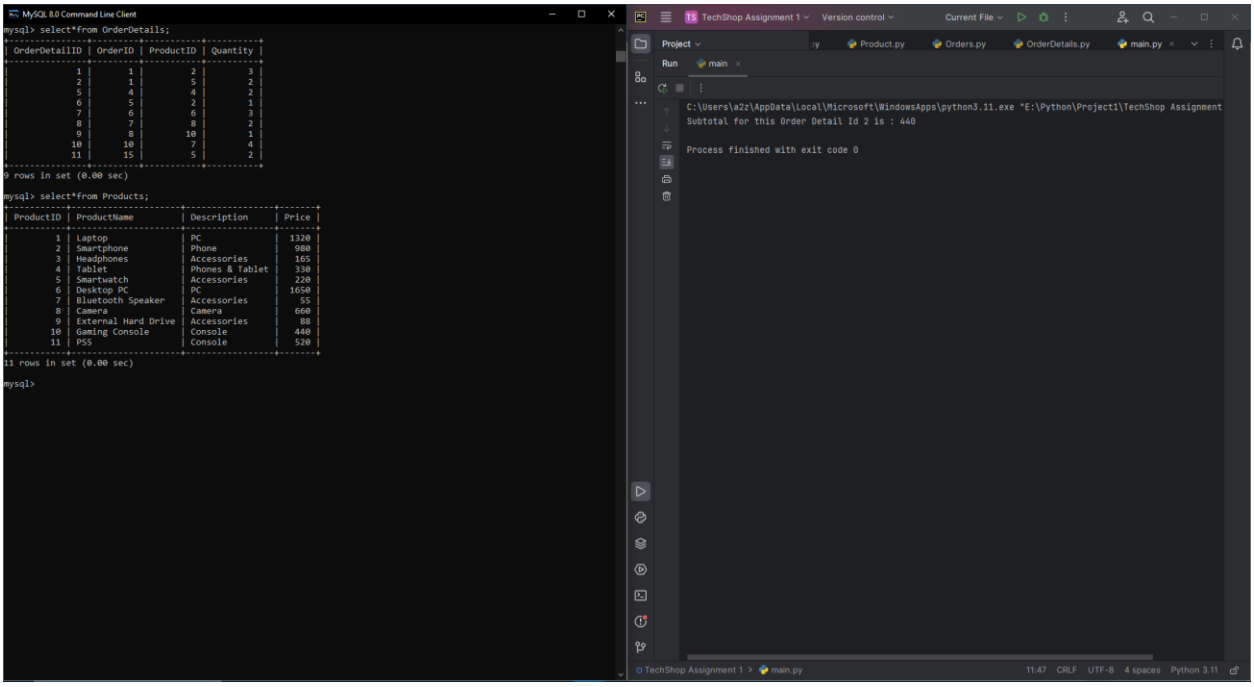
```
    @Quantity.setter
    def Quantity(self,new_quantity):
        if new_quantity >= 0:
            self._Quantity=new_quantity
        else:
            pass

    def Calculate_subtotal(self,OrderDetailsID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select OD.Quantity,P.Price From OrderDetails OD Inner Join Products P ON Od.ProductId=P.ProductID Where OrderDetailID=%s"
            values=(OrderDetailsID,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                quantity=record[0]
                price=record[1]
                Subtotal=quantity*price
                print(f"Subtotal for this Order Detail Id {OrderDetailsID} is : {Subtotal}")
            else:
                print("OrderDetailID not Found")

        except:
            print("Error in fetching OrderDetailId")
        finally:
            self.db_connector.close_connection()

    def GetOrderdetailInfo(self,OrderDetailID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select * From OrderDetails Where OrderID=%s"
            values=(OrderDetailID,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"Getting info for Order Detail ID : {OrderDetailID}")
                print(f"OrderDetailID : {record[0]}")
                print(f"OrderID : {record[1]}")
                print(f"ProductID : {record[2]}")
```



```
                print(f"Quantity : {record[5]}")
            else:
                print("No Order Found")
        except Exception as e:
            print("Error Please Enter the OrderDeatailId only")
        finally:
            self.db_connector.close_connection()

    def Update_Quantity(self,new_quantity,OrderDetailID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            print(f"Updating Quantity to {new_quantity} of Order Detail ID {OrderDetailID} ")
            query="Update OrderDetails Set Quantity=%s Where OrderDetailID =%s"
            values=(new_quantity,OrderDetailID,)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print("Quantity Updated Successfully")
        except:
            print("Error In Updation")
        finally:
            self.db_connector.close_connection()

    1 usage
    def AddDiscount(self,Discount_percentage,OrderID):
        try:
            if Discount_percentage>0:
                self.db_connector.open_connection()
                cur = self.db_connector.connection.cursor()
                query = "Select TotalAmount From Orders Where OrderID=%s"
                values = (OrderID,)
                cur.execute(query, values)
                record = cur.fetchone()
                if record:
                    print(f"Getting Discount on Order ID : {OrderID}")
                    Amount = record[0]
                    Payment = Amount - (Amount * Discount_percentage / 100)
                    print(f"Total Amount after Discount = {Payment}")
                else:
                    print("Wrong OrderID Entered")
                else:
```

```
                    Payment = Amount - (Amount * Discount_percentage / 100)
                    print(f"Total Amount after Discount = {Payment}")
                else:
                    print("Wrong OrderID Entered")
            else:
                print("Discount Percentage Cannot be in Negative")
        except:
            print("Error in Payment")
        finally:
            self.db_connector.close_connection()
```

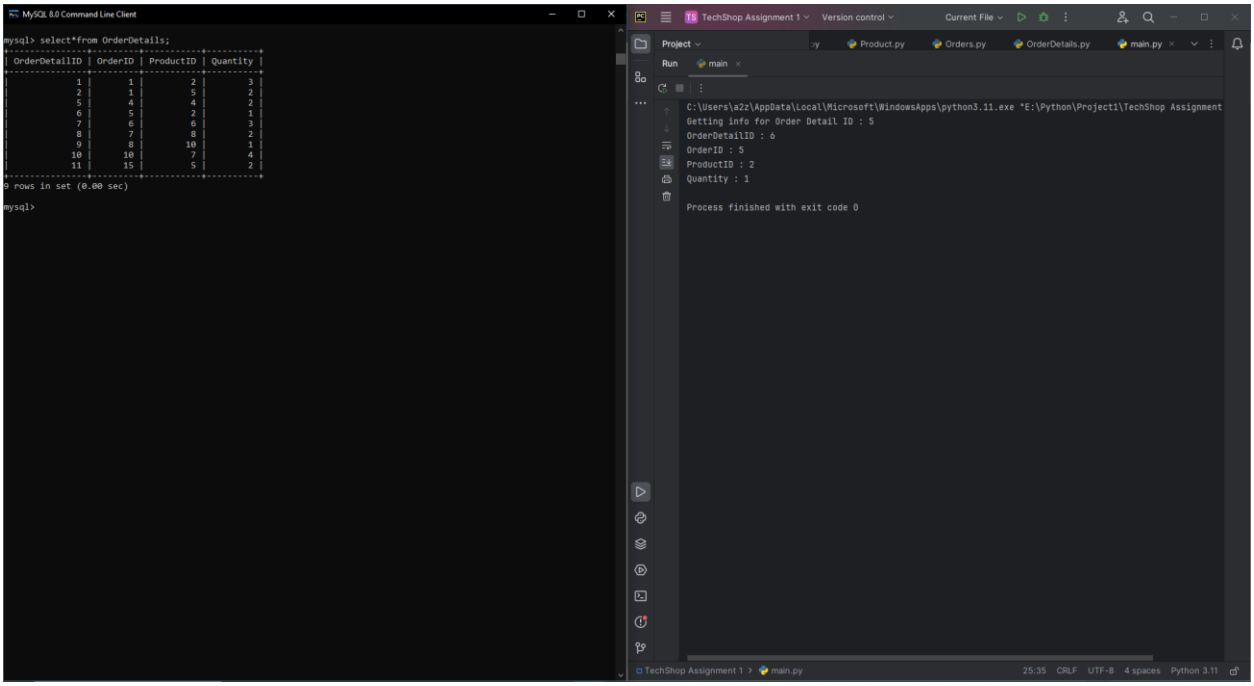
Calculating Subtotal of a Particular Order Detail



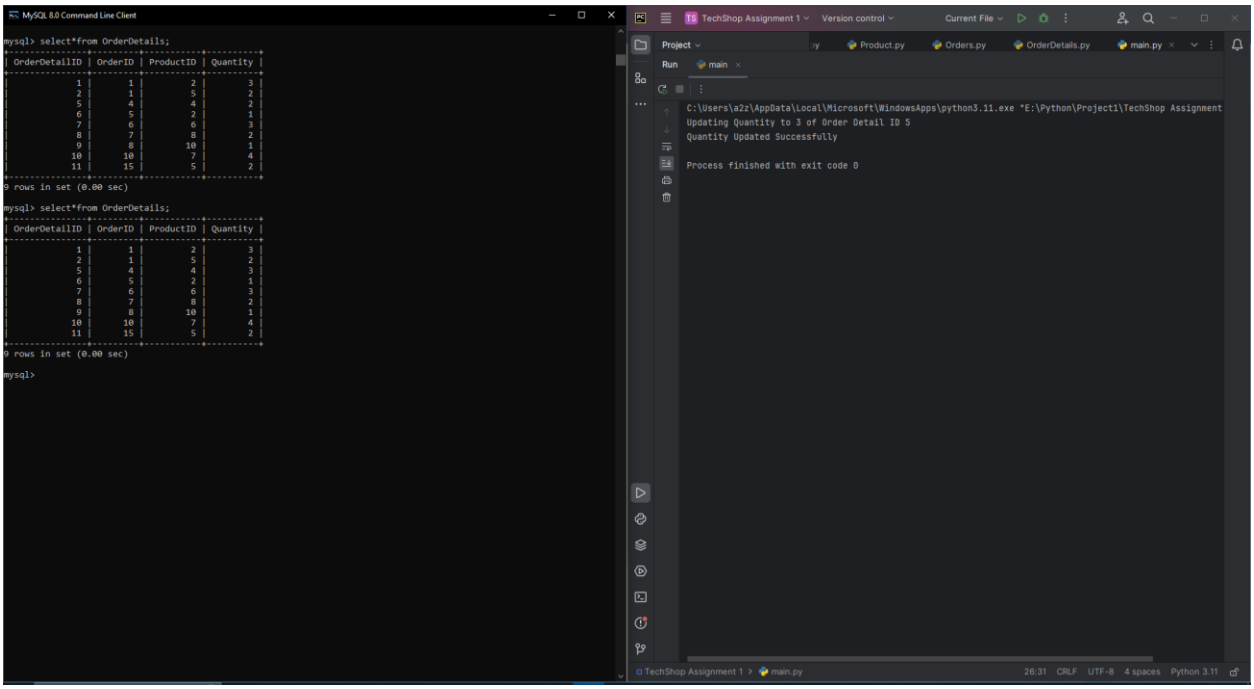
```
mysql> select*from OrderDetails;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|             1 |       1 |         2 |        3 |
|             2 |       1 |         5 |        2 |
|             5 |       4 |         4 |        2 |
|             6 |       5 |         2 |        1 |
|             7 |       6 |         6 |        3 |
|             8 |       7 |         8 |        2 |
|             9 |       8 |        10 |        1 |
|            10 |      10 |         7 |        4 |
|            11 |      15 |         5 |        2 |
+---------------+---------+-----------+----------+
9 rows in set (0.00 sec)

mysql> select*from Products;
+-----------+--------------------+----------------+-------+
| ProductID | ProductName        | Description    | Price |
+-----------+--------------------+----------------+-------+
|         1 | Laptop             | PC             |  1320 |
|         2 | Smartphone         | Phone          |   980 |
|         3 | Headphones         | Accessories    |   165 |
|         4 | Tablet             | Phones & Tablet|   330 |
|         5 | Smartwatch         | Accessories    |   220 |
|         6 | Desktop PC         | PC             |  1650 |
|         7 | Bluetooth Speaker  | Accessories    |    55 |
|         8 | Camera             | Camera         |   660 |
|         9 | External Hard Drive| Accessories    |    88 |
|        10 | Gaming Console     | Console        |   440 |
|        11 | PS5                | Console        |   520 |
+-----------+--------------------+----------------+-------+
11 rows in set (0.00 sec)

mysql>
```

```
C:\Users\a2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment
Subtotal for this Order Detail Id 2 is : 440

Process finished with exit code 0
```

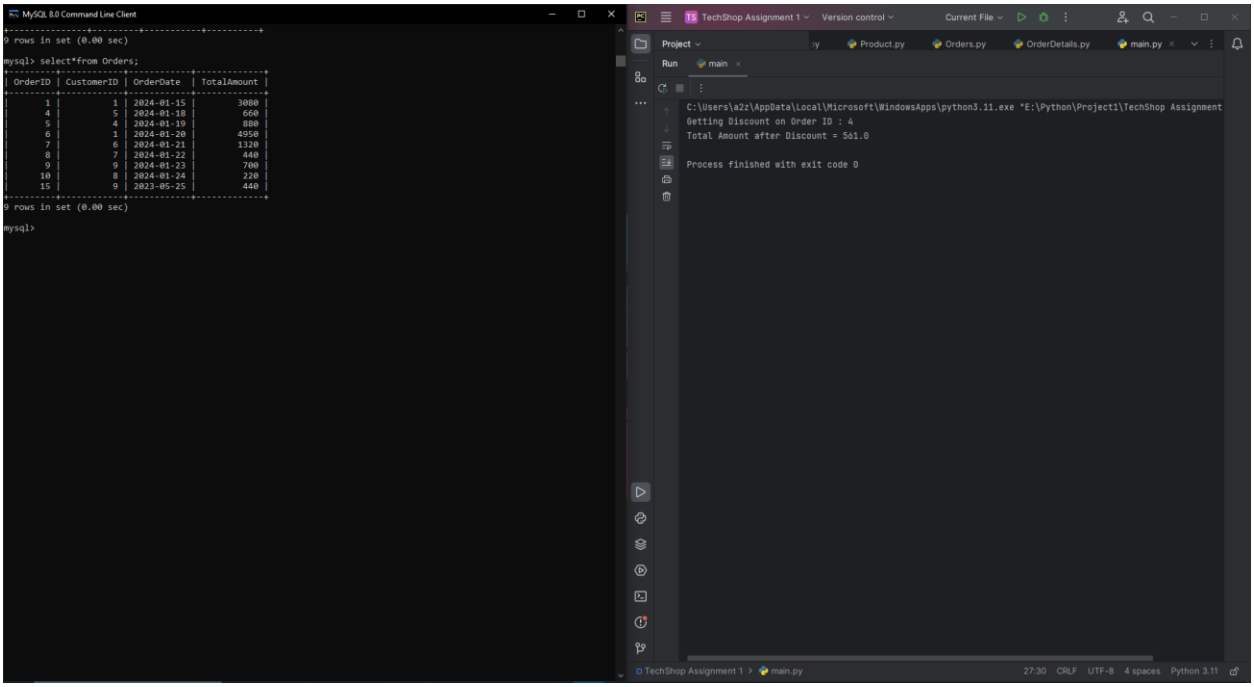
Getting all the info for a particular Order Detail



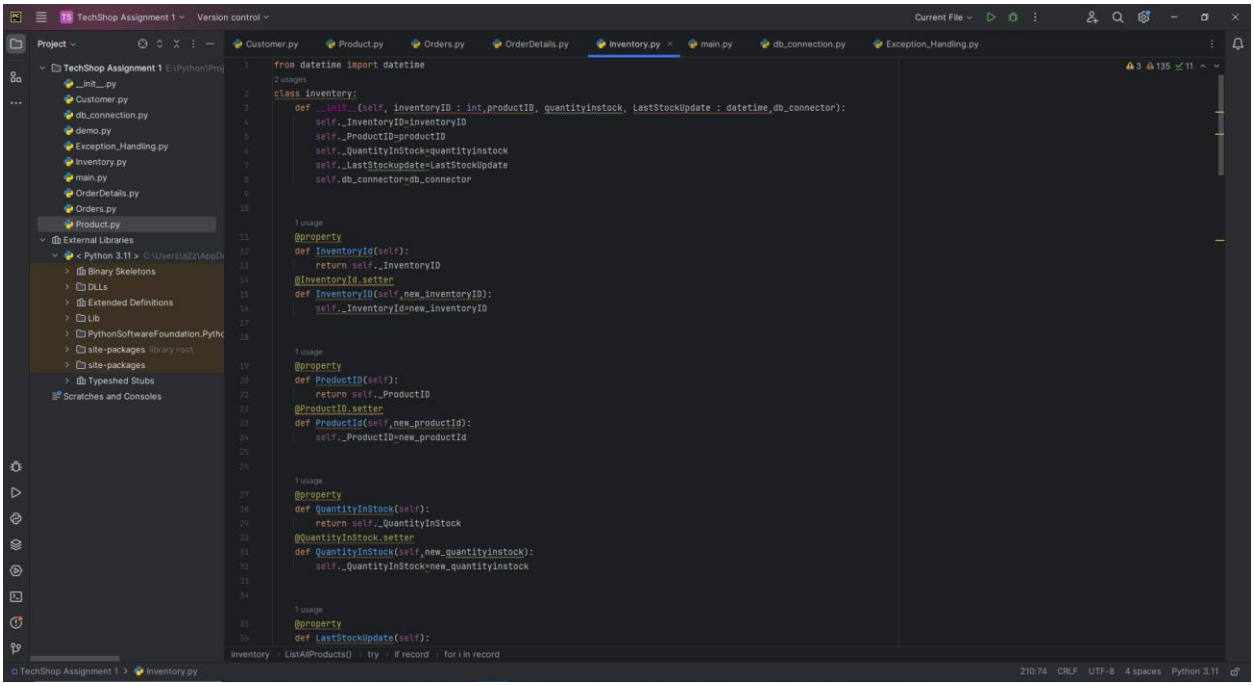
Updating the Quantity in the Order Detail



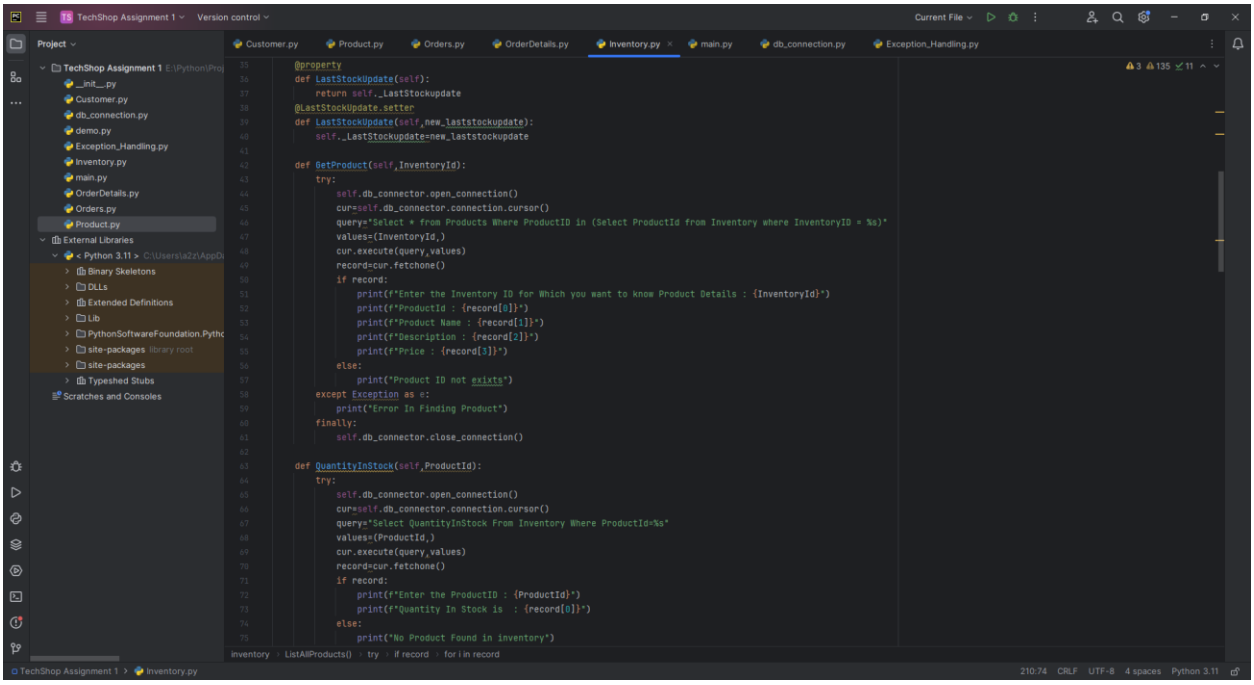
Adding Discount in Total Bill



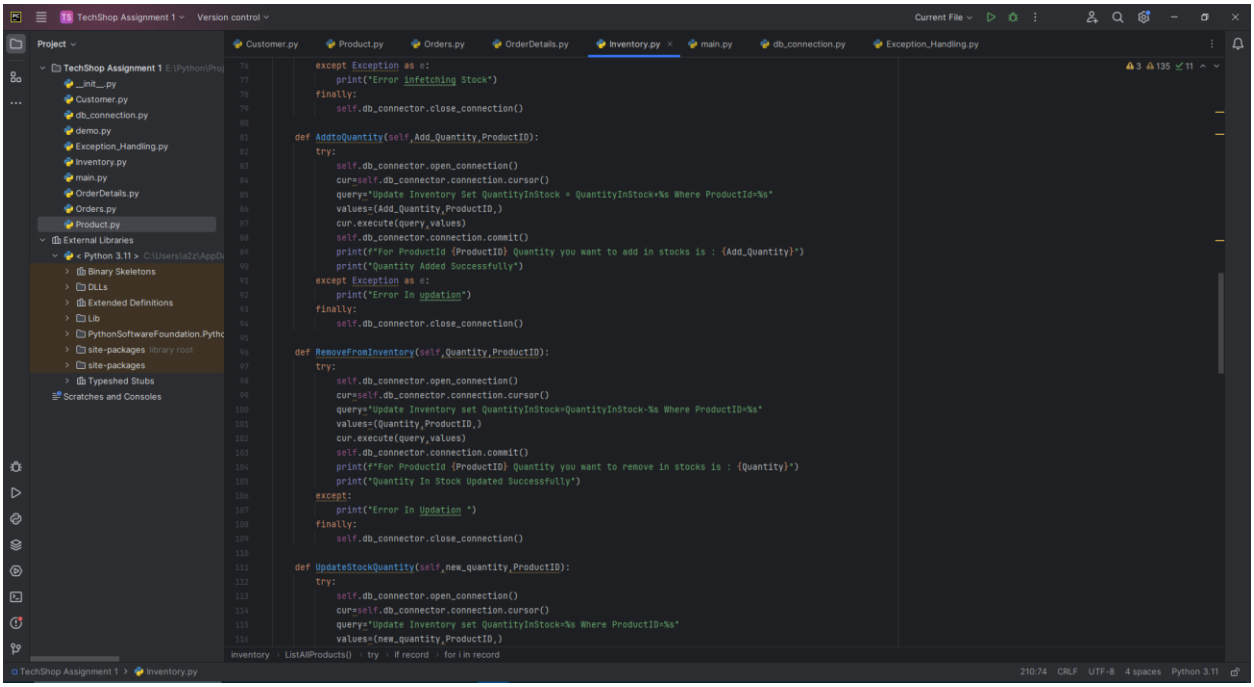
Inventory Class:



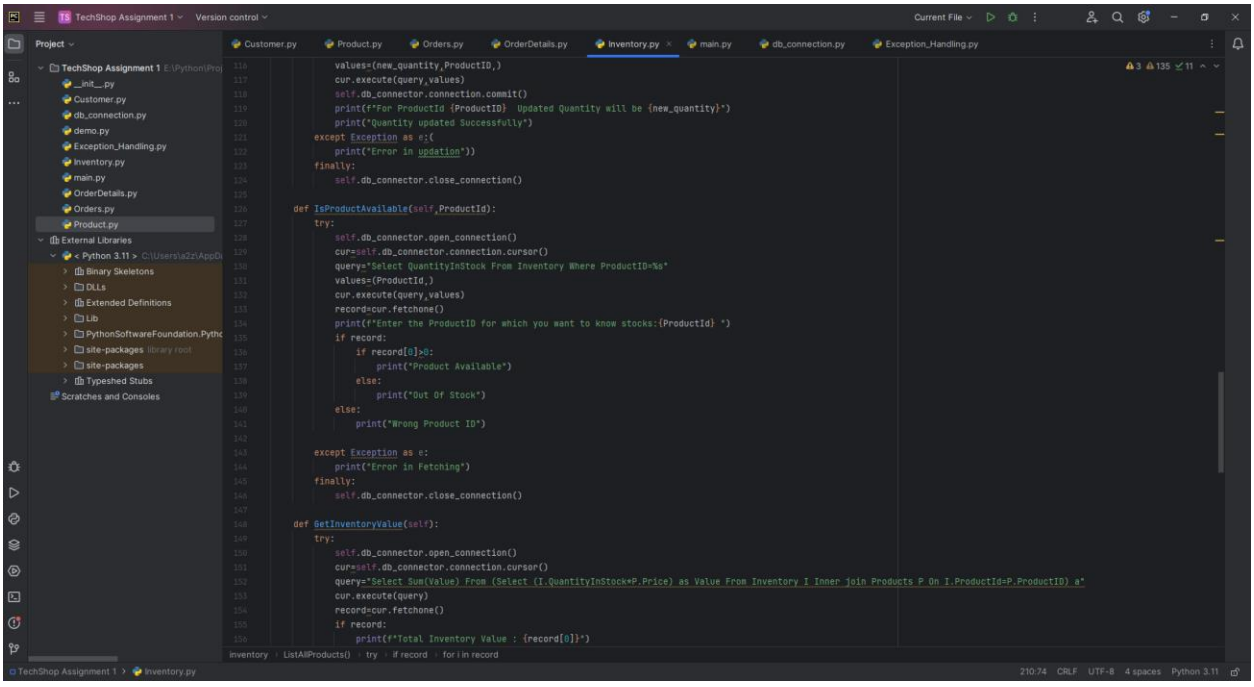
```python
from datetime import datetime
class inventory:
    def __init__(self, inventoryID : int,productID, quantityinstock, LastStockUpdate : datetime,db_connector):
        self._InventoryID=inventoryID
        self._ProductID=productID
        self._QuantityInStock=quantityinstock
        self._LastStockupdate=LastStockUpdate
        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):
        self._QuantityInStock=new_quantityinstock


    @property
    def LastStockUpdate(self):
```



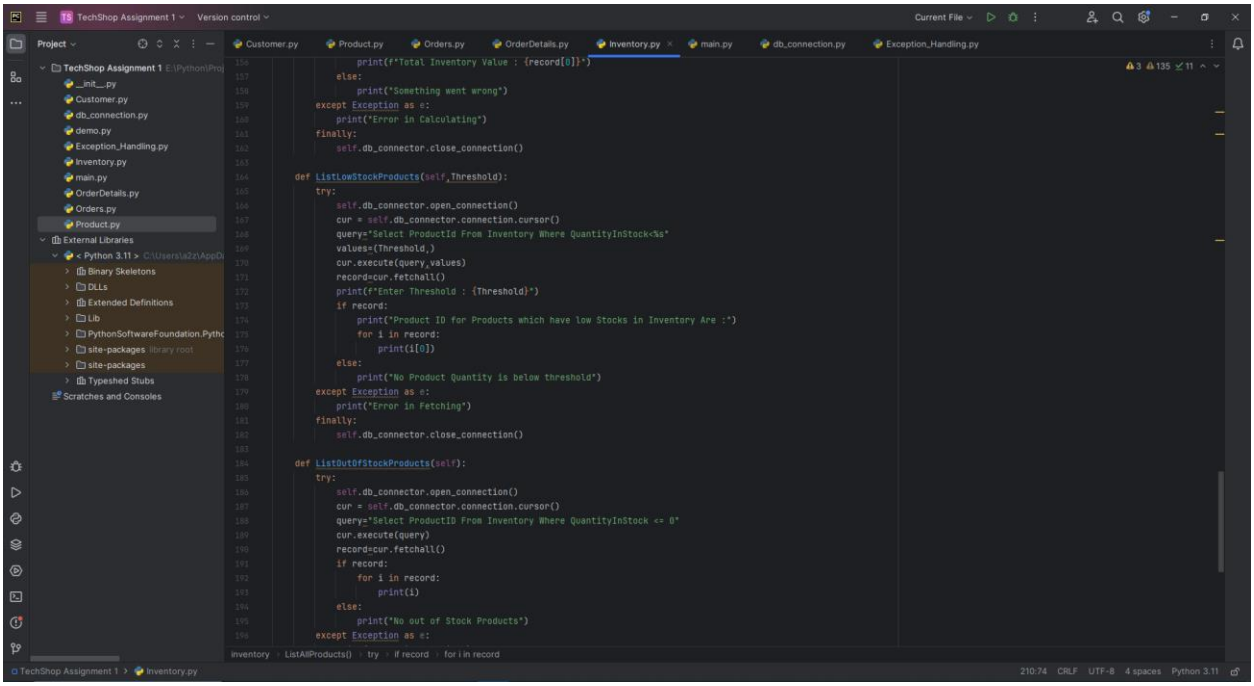
```python
    @property
    def LastStockUpdate(self):
        return self._LastStockupdate
    @LastStockUpdate.setter
    def LastStockUpdate(self,new_laststockupdate):
        self._LastStockupdate=new_laststockupdate

    def GetProduct(self,InventoryId):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select * from Products Where ProductID in (Select ProductId from Inventory where InventoryID = %s)"
            values=(InventoryId,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"Enter the Inventory ID for Which you want to know Product Details : {InventoryId}")
                print(f"ProductId : {record[0]}")
                print(f"Product Name : {record[1]}")
                print(f"Description : {record[2]}")
                print(f"Price : {record[3]}")
            else:
                print("Product ID not exixts")
        except Exception as e:
            print("Error In Finding Product")
        finally:
            self.db_connector.close_connection()

    def QuantityInStock(self,ProductId):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select QuantityInStock From Inventory Where ProductId=%s"
            values=(ProductId,)
            cur.execute(query,values)
            record=cur.fetchone()
            if record:
                print(f"Enter the ProductID : {ProductId}")
                print(f"Quantity In Stock is  : {record[0]}")
            else:
                print("No Product Found in inventory")
```

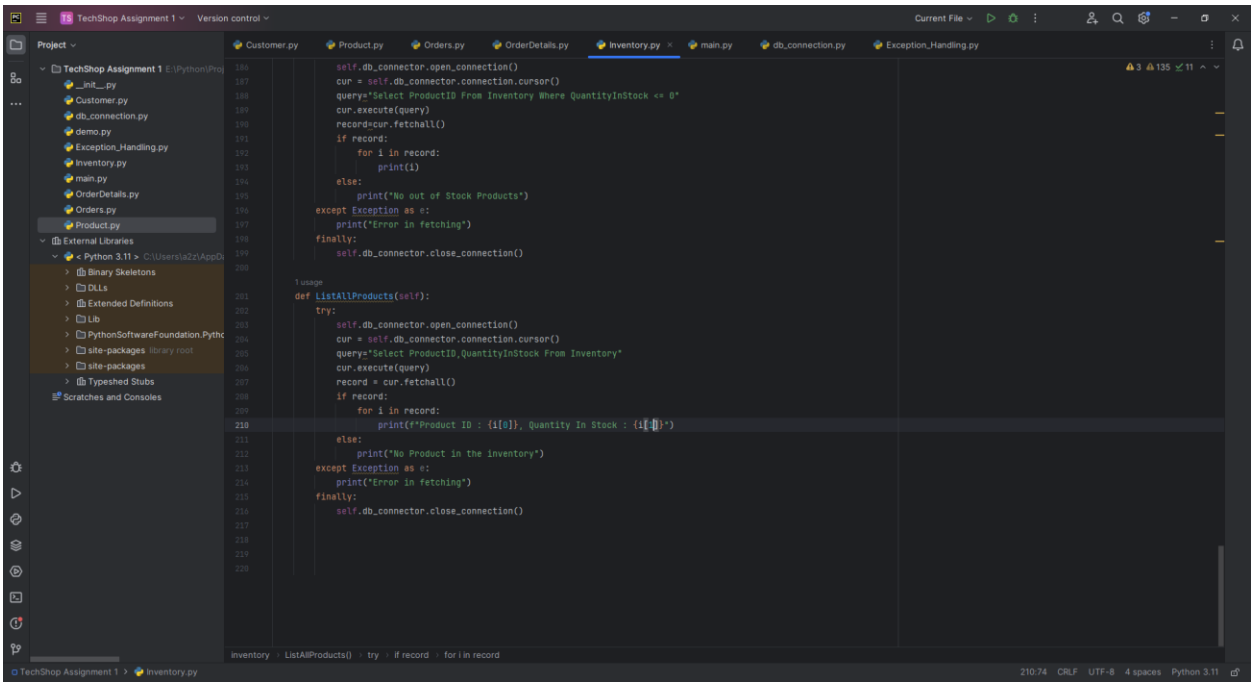


```
        except Exception as e:
            print("Error infetching Stock")
        finally:
            self.db_connector.close_connection()

    def AddtoQuantity(self,Add_Quantity,ProductID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Update Inventory Set QuantityInStock = QuantityInStock+%s Where ProductId=%s"
            values=(Add_Quantity,ProductID,)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print(f"For ProductId {ProductID} Quantity you want to add in stocks is : {Add_Quantity}")
            print("Quantity Added Successfully")
        except Exception as e:
            print("Error In updation")
        finally:
            self.db_connector.close_connection()

    def RemoveFromInventory(self,Quantity,ProductID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Update Inventory set QuantityInStock=QuantityInStock-%s Where ProductID=%s"
            values=(Quantity,ProductID,)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print(f"For ProductId {ProductID} Quantity you want to remove in stocks is : {Quantity}")
            print("Quantity In Stock Updated Successfully")
        except:
            print("Error In Updation ")
        finally:
            self.db_connector.close_connection()

    def UpdateStockQuantity(self,new_quantity,ProductID):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Update Inventory set QuantityInStock=%s Where ProductID=%s"
            values=(new_quantity,ProductID,)
```

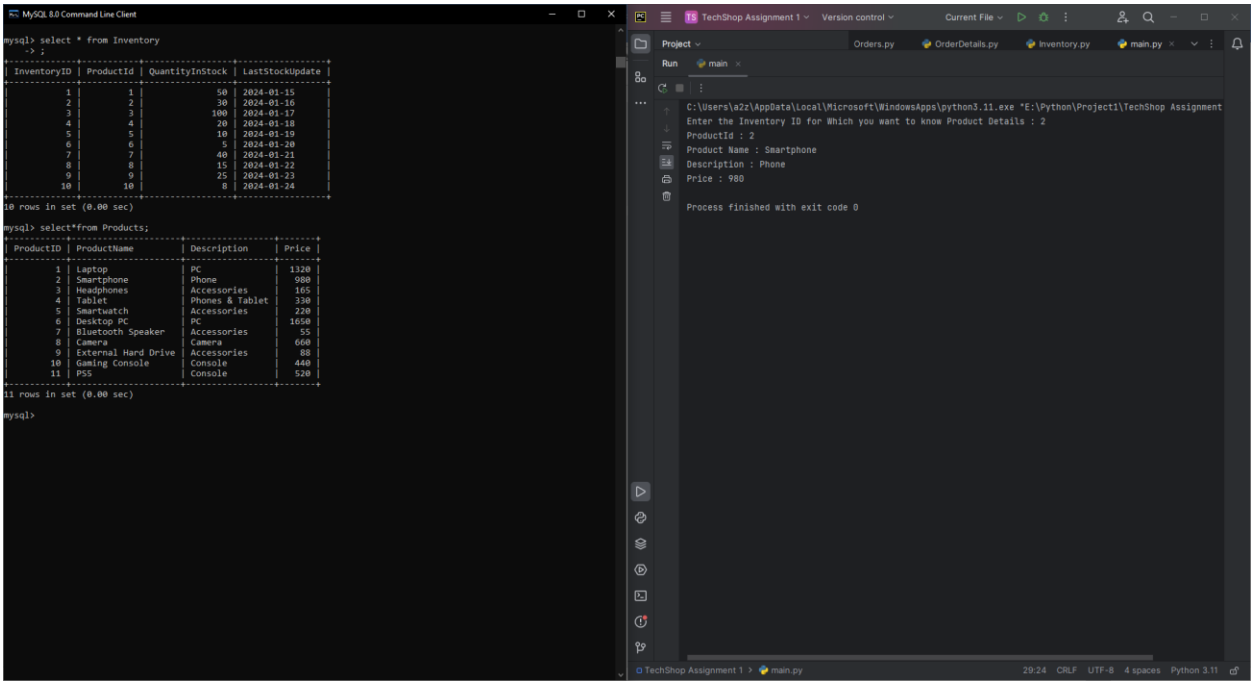


```
            values=(new_quantity,ProductID,)
            cur.execute(query,values)
            self.db_connector.connection.commit()
            print(f"For ProductId {ProductID}  Updated Quantity will be {new_quantity}")
            print("Quantity updated Successfully")
        except Exception as e:{
            print("Error in updation"))
        finally:
            self.db_connector.close_connection()

    def IsProductAvailable(self,ProductId):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select QuantityInStock From Inventory Where ProductID=%s"
            values=(ProductId,)
            cur.execute(query,values)
            record=cur.fetchone()
            print(f"Enter the ProductID for which you want to know stocks:{ProductId} ")
            if record:
                if record[0]>0:
                    print("Product Available")
                else:
                    print("Out Of Stock")
            else:
                print("Wrong Product ID")

        except Exception as e:
            print("Error in Fetching")
        finally:
            self.db_connector.close_connection()

    def GetInventoryValue(self):
        try:
            self.db_connector.open_connection()
            cur=self.db_connector.connection.cursor()
            query="Select Sum(Value) From (Select (I.QuantityInStock*P.Price) as Value From Inventory I Inner join Products P On I.ProductId=P.ProductID) a"
            cur.execute(query)
            record=cur.fetchone()
            if record:
                print(f"Total Inventory Value : {record[0]}")
```

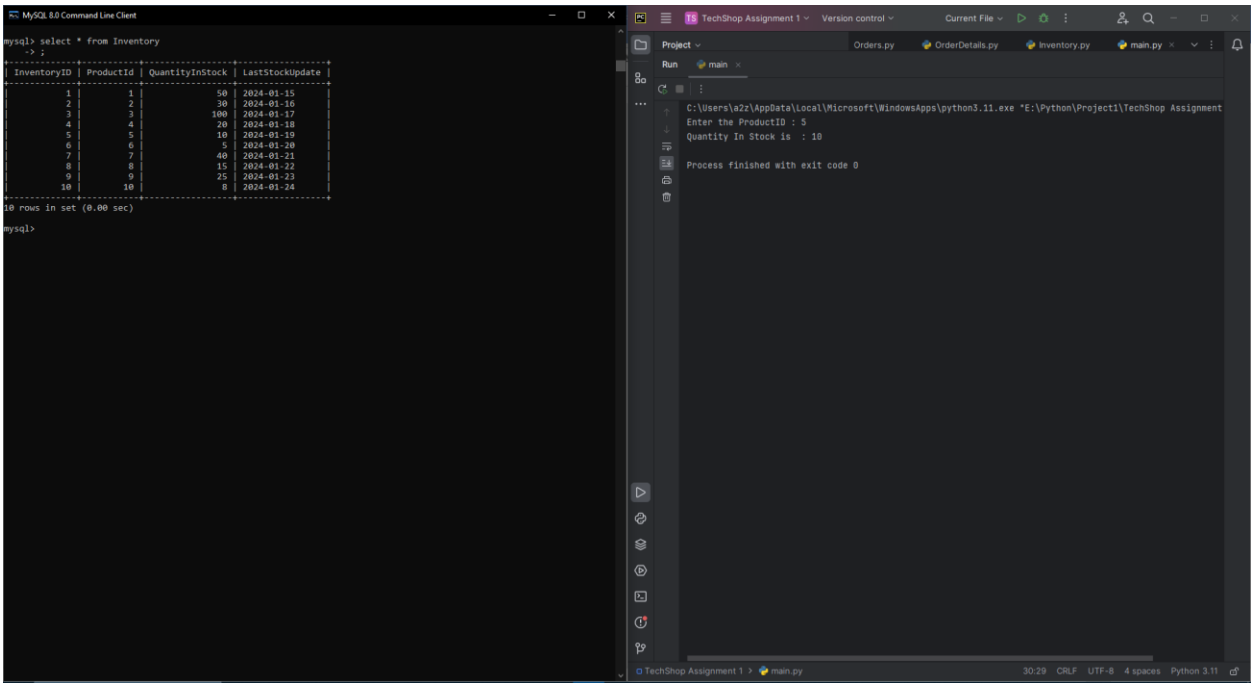

TechShop Assignment 1
Version control
Current File
Project
Customer.py
Product.py
Orders.py
OrderDetails.py
Inventory.py
main.py
db_connection.py
Exception_Handling.py
print(f"Total Inventory Value : {record[0]}")
else:
print("Something went wrong")
except Exception as e:
print("Error in Calculating")
finally:
self.db_connector.close_connection()
def ListLowStockProducts(self,Threshold):
try:
self.db_connector.open_connection()
cur = self.db_connector.connection.cursor()
query="Select ProductId From Inventory Where QuantityInStock<%s"
values=(Threshold,)
cur.execute(query,values)
record=cur.fetchall()
print(f"Enter Threshold : {Threshold}")
if record:
print("Product ID for Products which have low Stocks in Inventory Are :")
for i in record:
print(i[0])
else:
print("No Product Quantity is below threshold")
except Exception as e:
print("Error in Fetching")
finally:
self.db_connector.close_connection()
def ListOutOfStockProducts(self):
try:
self.db_connector.open_connection()
cur = self.db_connector.connection.cursor()
query="Select ProductID From Inventory Where QuantityInStock <= 0"
cur.execute(query)
record=cur.fetchall()
if record:
for i in record:
print(i)
else:
print("No out of Stock Products")
except Exception as e:
inventory › ListAllProducts() › try › if record › for i in record
TechShop Assignment 1 › Inventory.py
210:74 CRLF UTF-8 4 spaces Python 3.11


TechShop Assignment 1
Version control
Current File
Project
Customer.py
Product.py
Orders.py
OrderDetails.py
Inventory.py
main.py
db_connection.py
Exception_Handling.py
self.db_connector.open_connection()
cur = self.db_connector.connection.cursor()
query="Select ProductID From Inventory Where QuantityInStock <= 0"
cur.execute(query)
record=cur.fetchall()
if record:
for i in record:
print(i)
else:
print("No out of Stock Products")
except Exception as e:
print("Error in fetching")
finally:
self.db_connector.close_connection()
1 usage
def ListAllProducts(self):
try:
self.db_connector.open_connection()
cur = self.db_connector.connection.cursor()
query="Select ProductID,QuantityInStock From Inventory"
cur.execute(query)
record = cur.fetchall()
if record:
for i in record:
print(f"Product ID : {i[0]}, Quantity In Stock : {i[1]}")
else:
print("No Product in the inventory")
except Exception as e:
print("Error in fetching")
finally:
self.db_connector.close_connection()
inventory › ListAllProducts() › try › if record › for i in record
TechShop Assignment 1 › Inventory.py
210:74 CRLF UTF-8 4 spaces Python 3.11

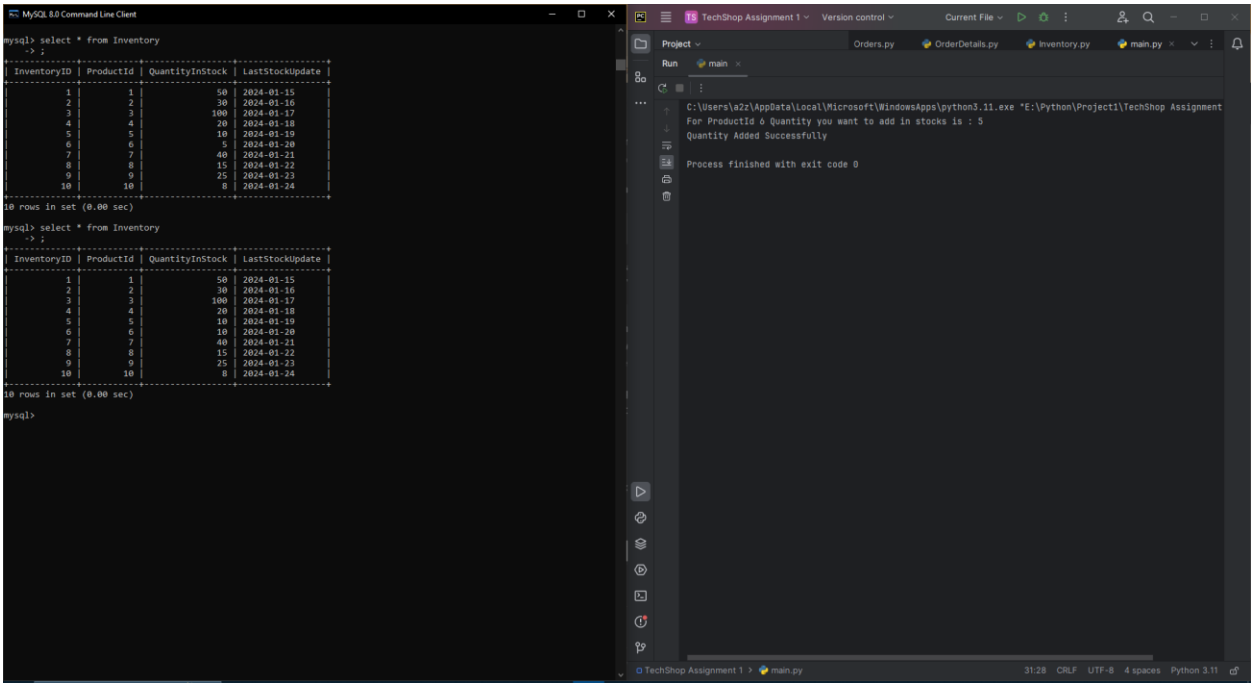
GetProduct(): A method to retrieve the product associated with this inventory item.



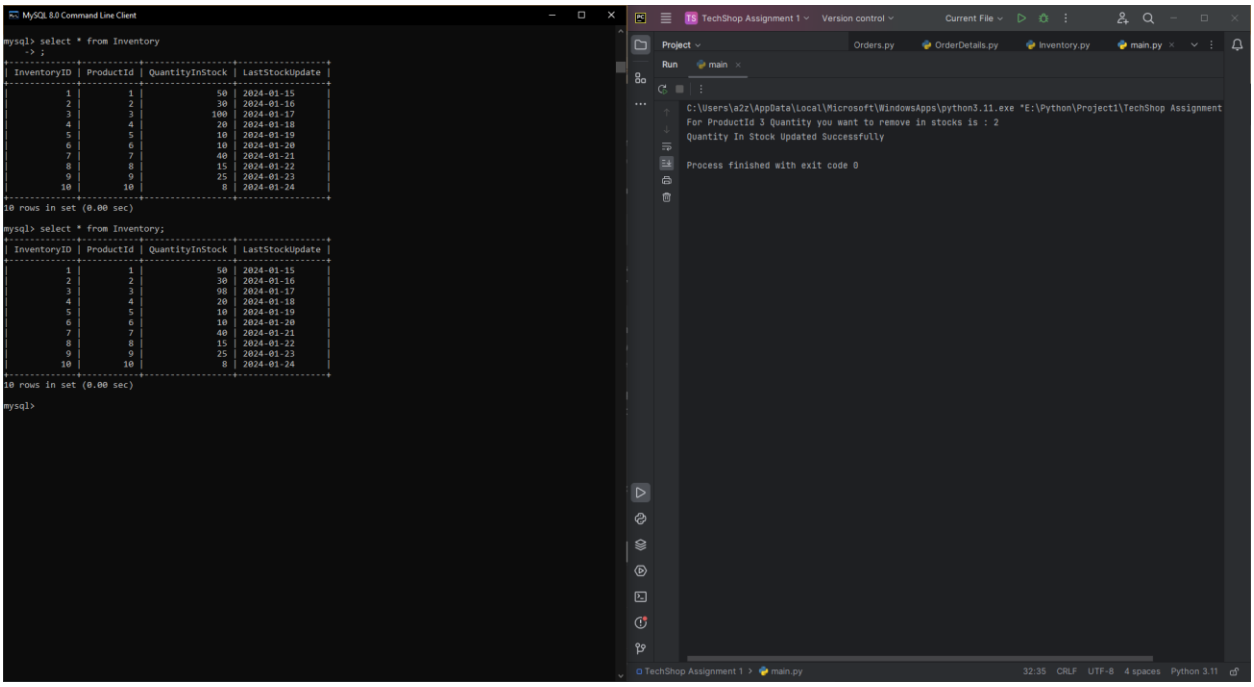
GetQuantityInStock(): A method to get the current quantity of the product in stock.



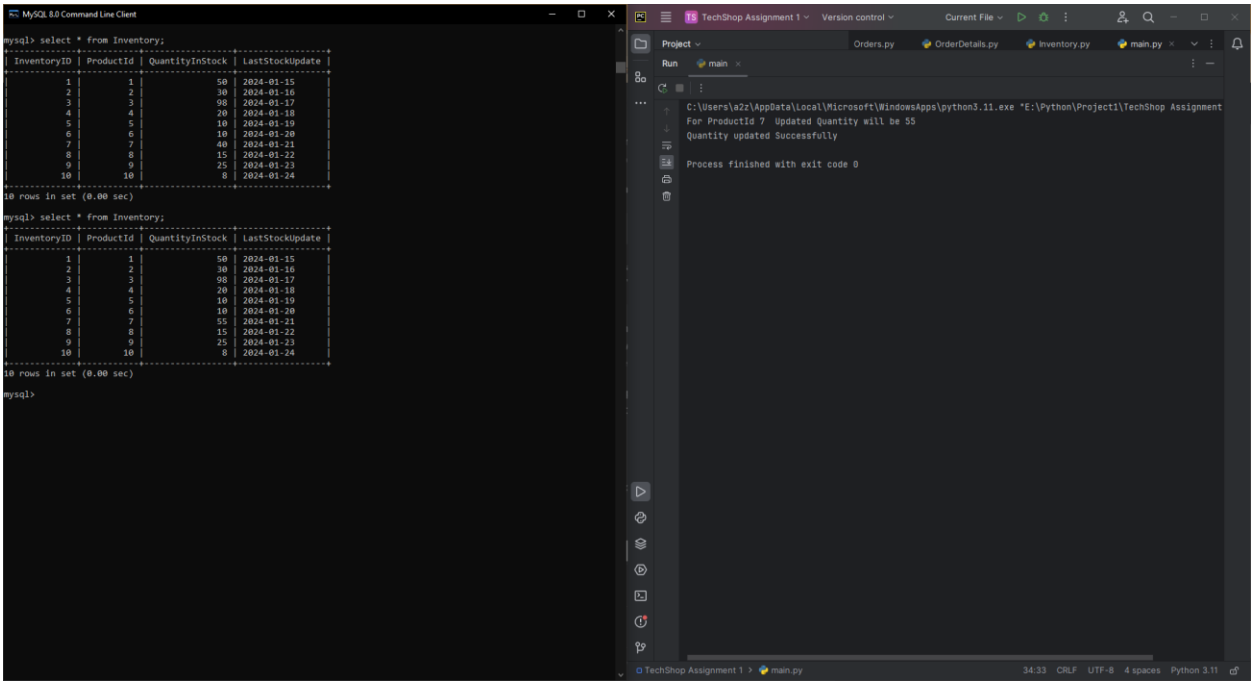
AddToInventory(int quantity): A method to add a specified quantity of the product to the inventory.



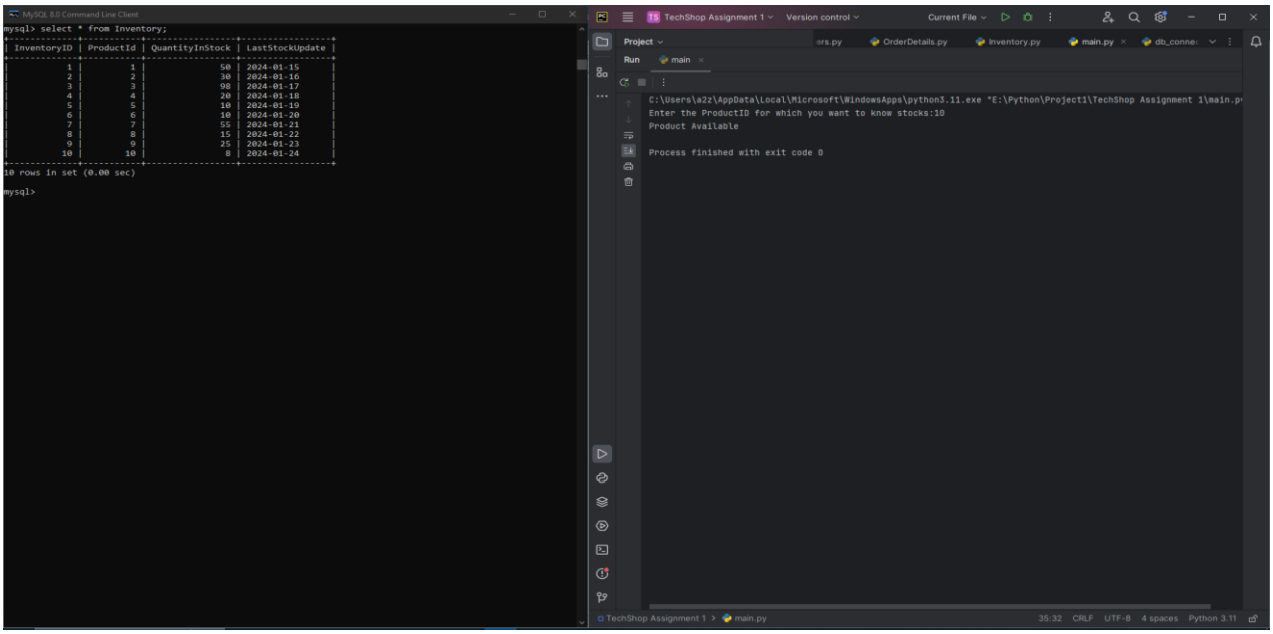
RemoveFromInventory(int quantity): A method to remove a specified quantity of the product from the inventory.

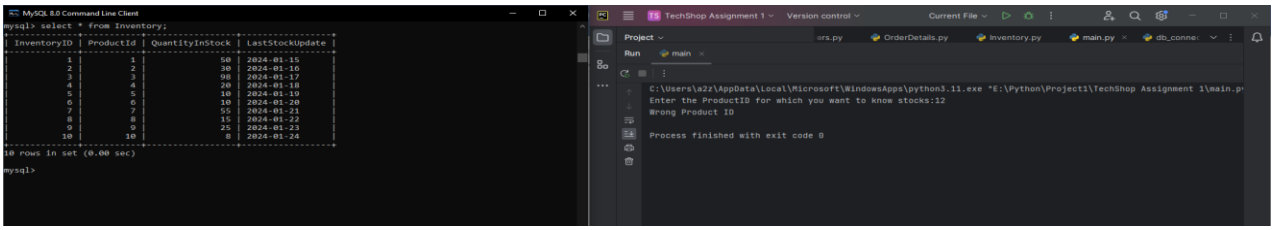


UpdateStockQuantity(int newQuantity): A method to update the stock quantity to a new value.

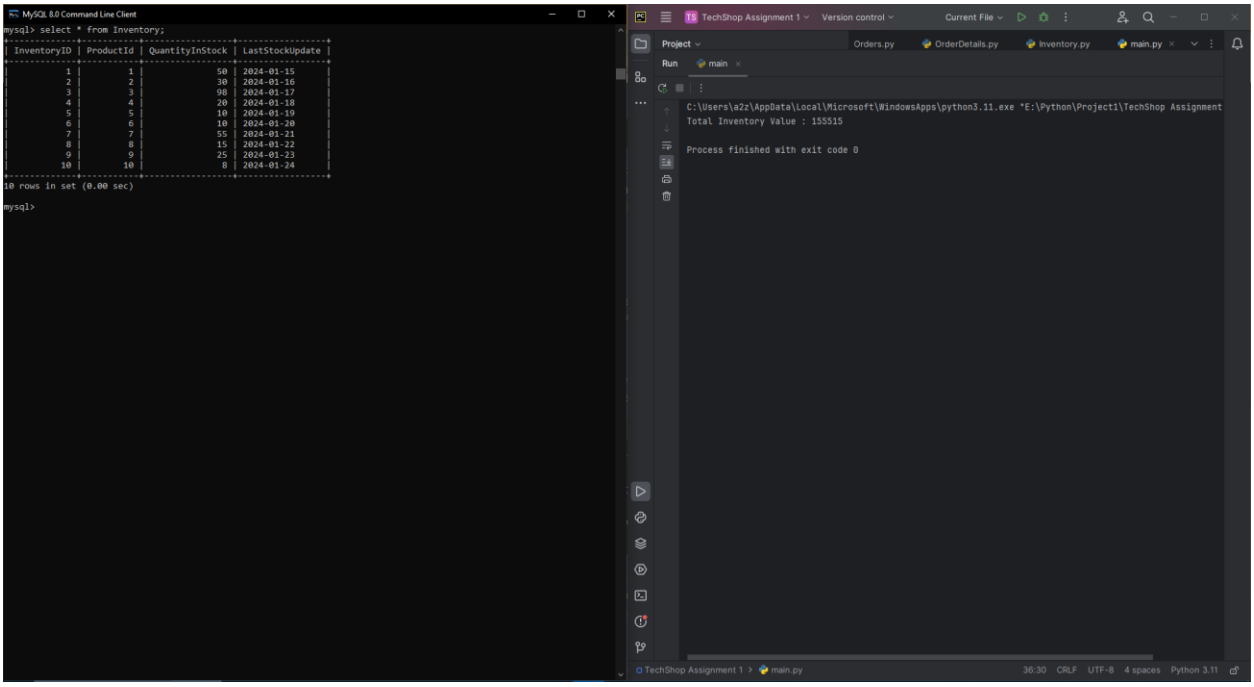


IsProductAvailable(int quantityToCheck): A method to check if a specified quantity of the product is available in the inventory.

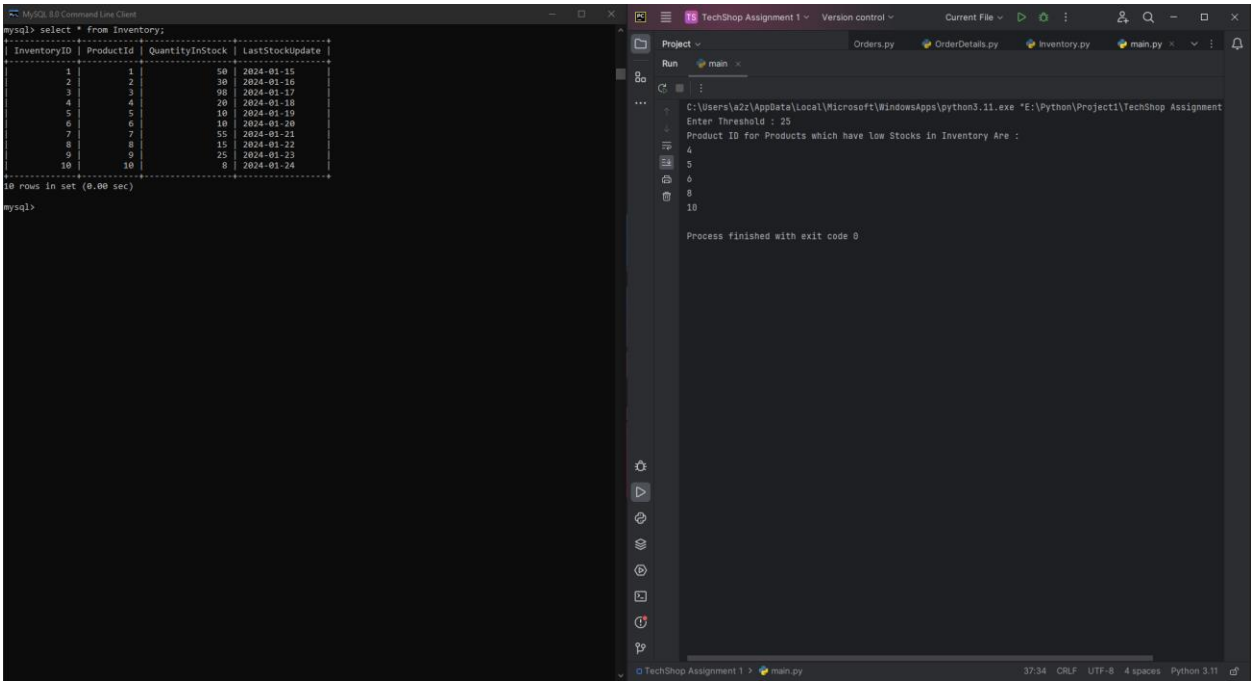




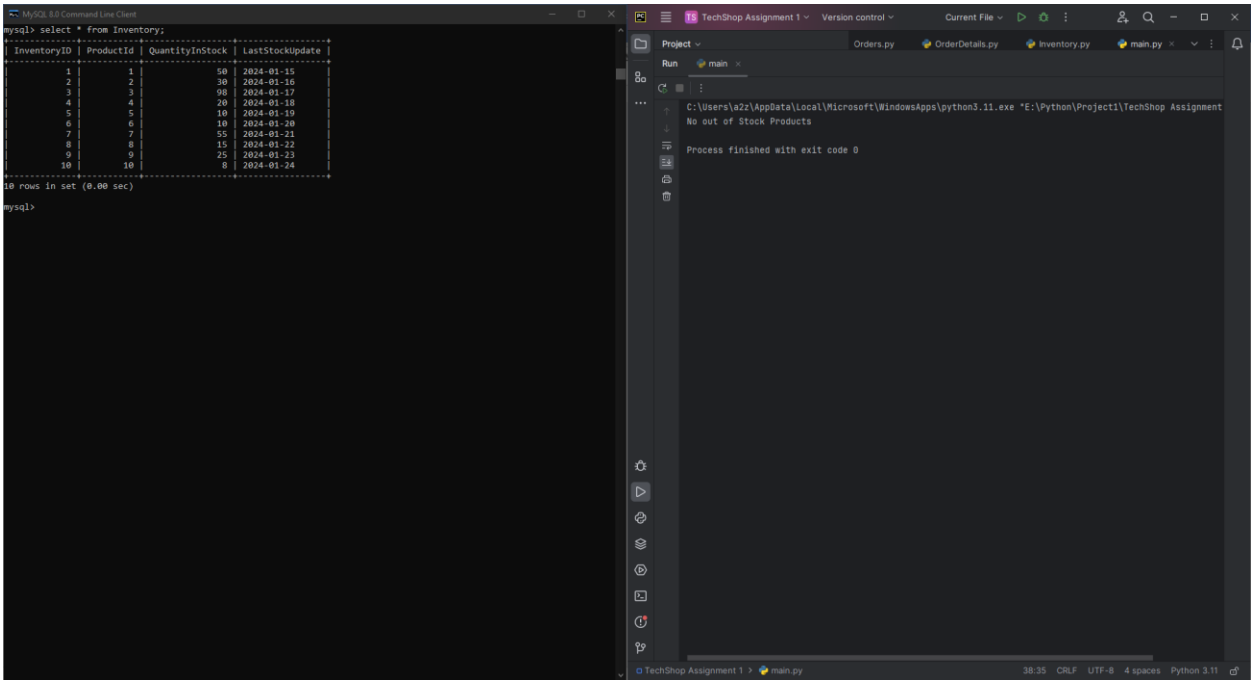
GetInventoryValue(): A method to calculate the total value of the products in the inventory based on their prices and quantities.



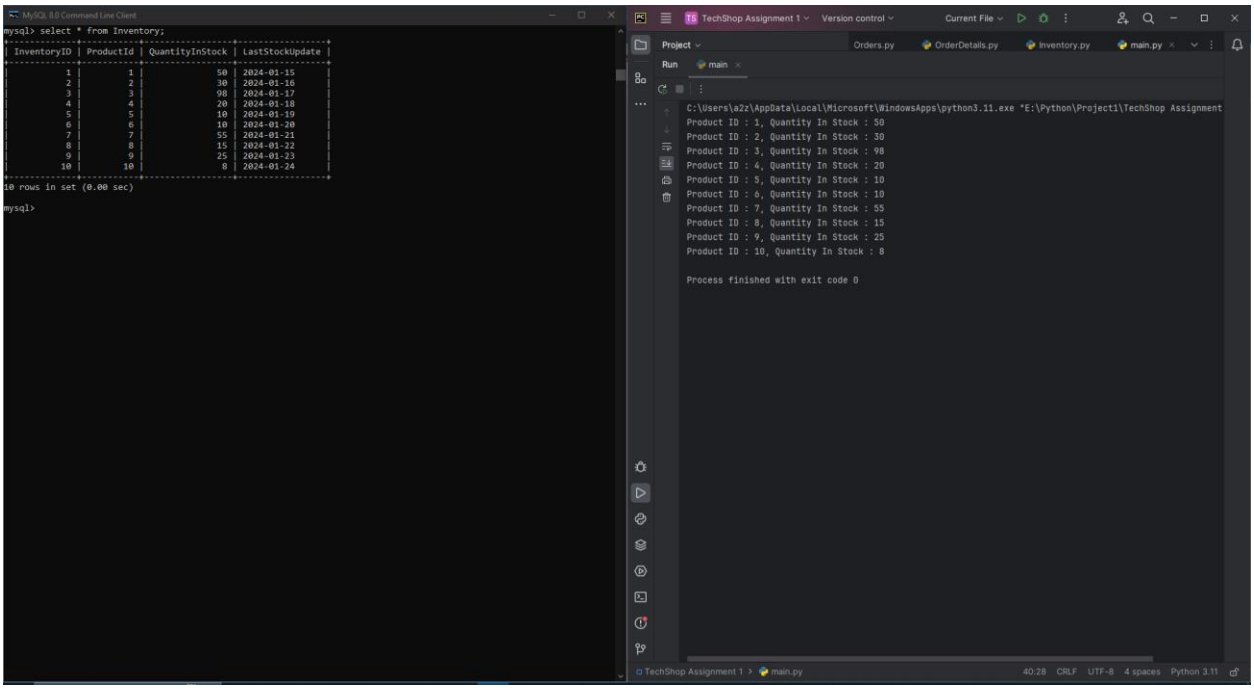
ListLowStockProducts(int threshold): A method to list products with quantities below a specified threshold, indicating low stock.



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.



Database Connectivity