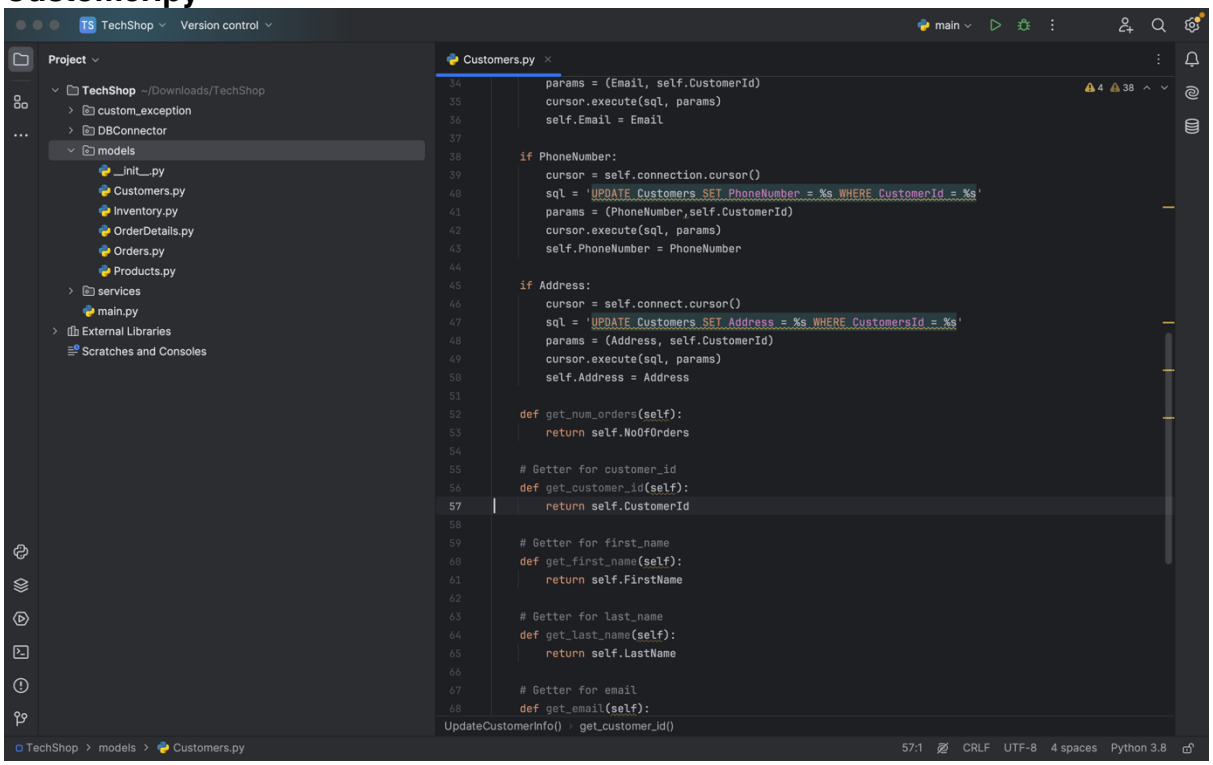
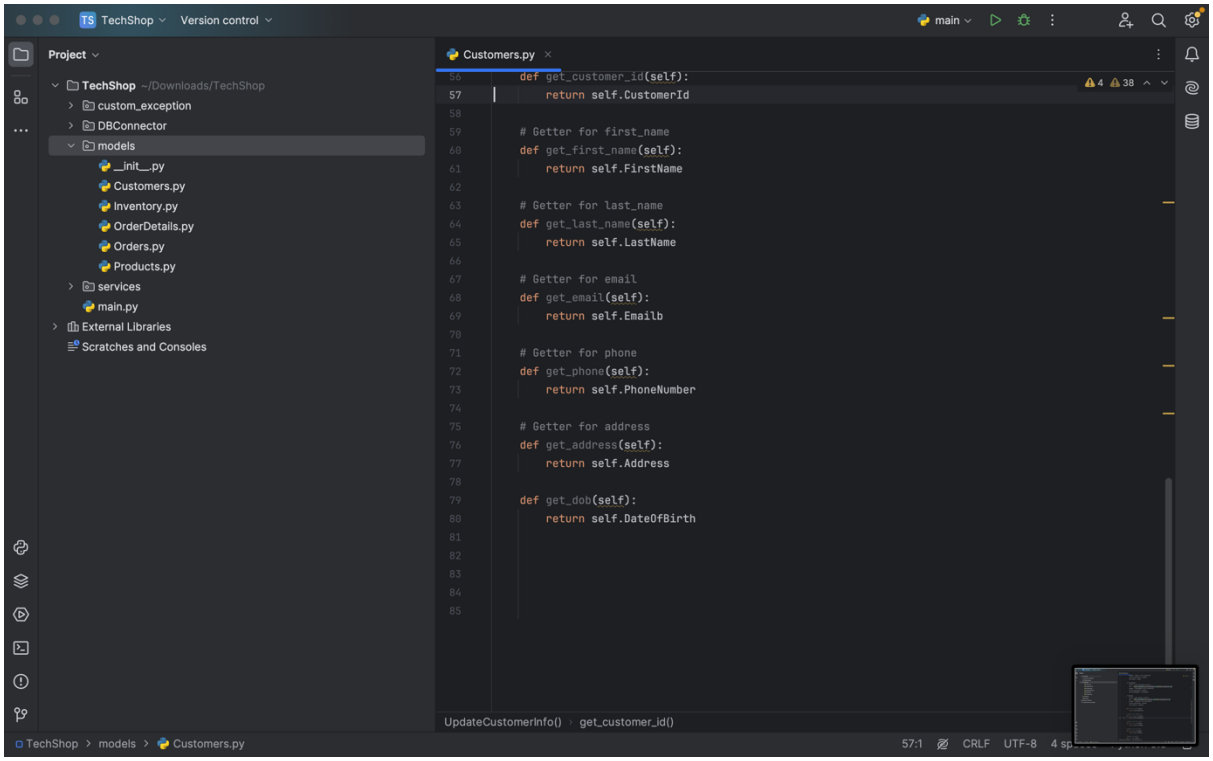
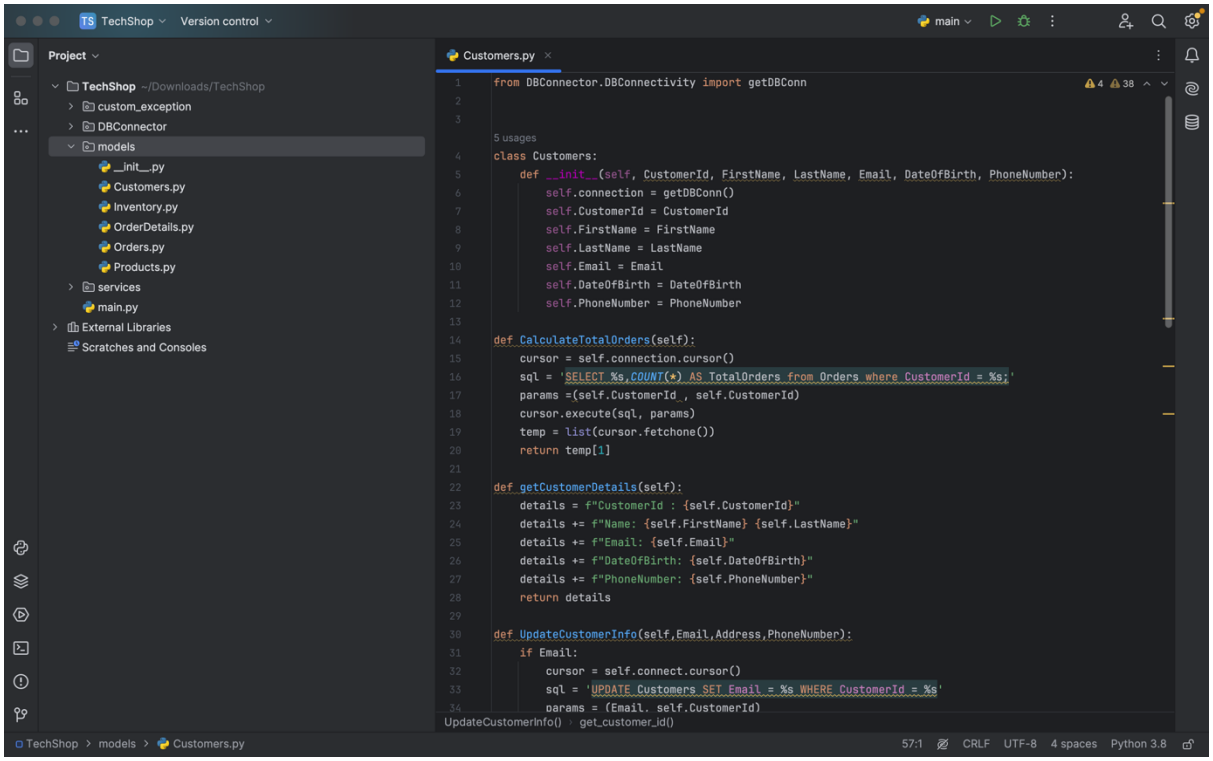
TechShop, an electronic gadgets shop
By
Sridhar S

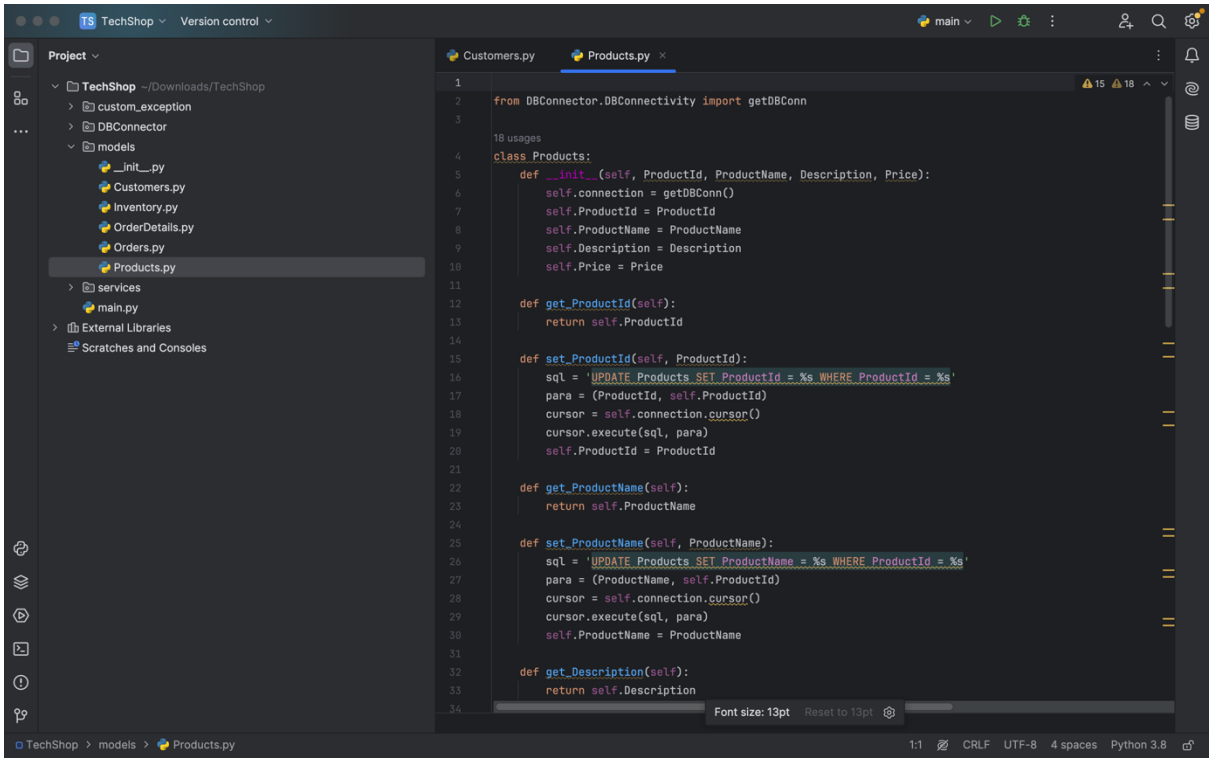
Customer.py

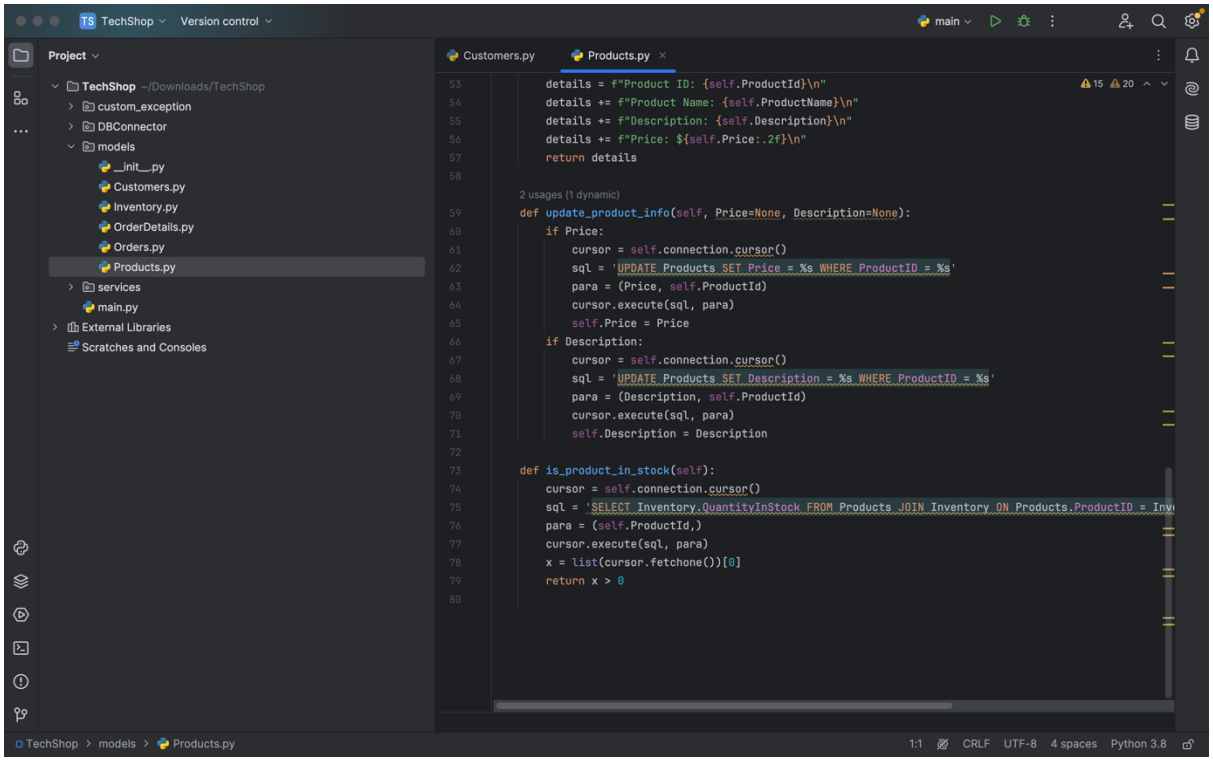


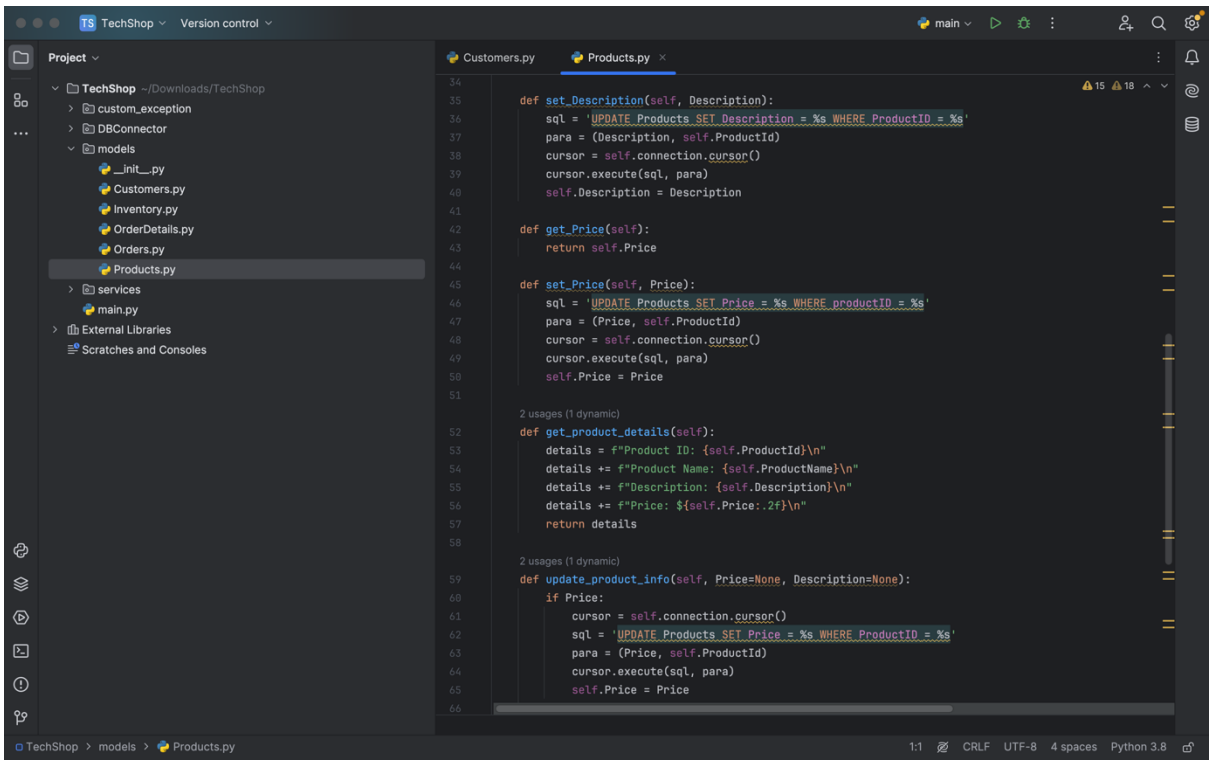



TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
from DBConnector.DBConnectivity import getDBConn
5 usages
class Customers:
def __init__(self, CustomerId, FirstName, LastName, Email, DateOfBirth, PhoneNumber):
self.connection = getDBConn()
self.CustomerId = CustomerId
self.FirstName = FirstName
self.LastName = LastName
self.Email = Email
self.DateOfBirth = DateOfBirth
self.PhoneNumber = PhoneNumber
def CalculateTotalOrders(self):
cursor = self.connection.cursor()
sql = 'SELECT %s,COUNT(*) AS TotalOrders from Orders where CustomerId = %s;'
params =(self.CustomerId , self.CustomerId)
cursor.execute(sql, params)
temp = list(cursor.fetchone())
return temp[1]
def getCustomerDetails(self):
details = f"CustomerId : {self.CustomerId}"
details += f"Name: {self.FirstName} {self.LastName}"
details += f"Email: {self.Email}"
details += f"DateOfBirth: {self.DateOfBirth}"
details += f"PhoneNumber: {self.PhoneNumber}"
return details
def UpdateCustomerInfo(self,Email,Address,PhoneNumber):
if Email:
cursor = self.connect.cursor()
sql = 'UPDATE Customers SET Email = %s WHERE CustomerId = %s'
params = (Email, self.CustomerId)
UpdateCustomerInfo() › get_customer_id()
TechShop › models › Customers.py
57:1 CRLF UTF-8 4 spaces Python 3.8

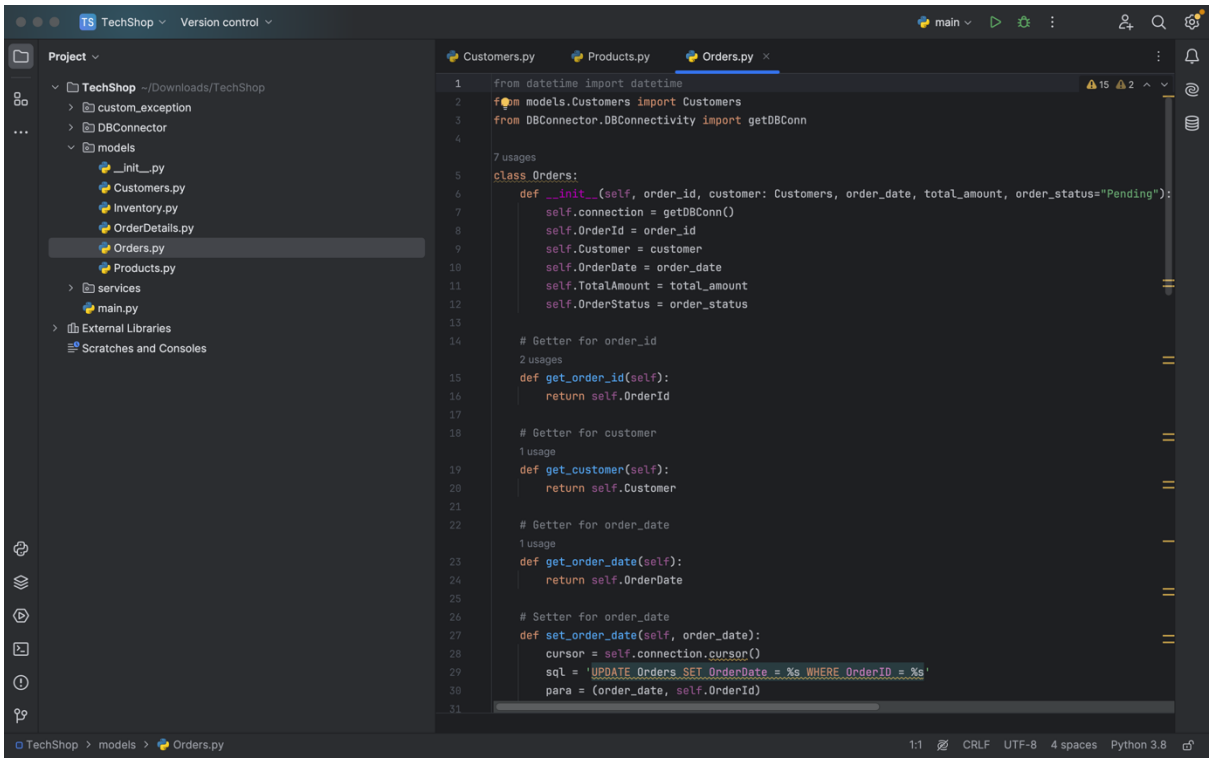
Products.py

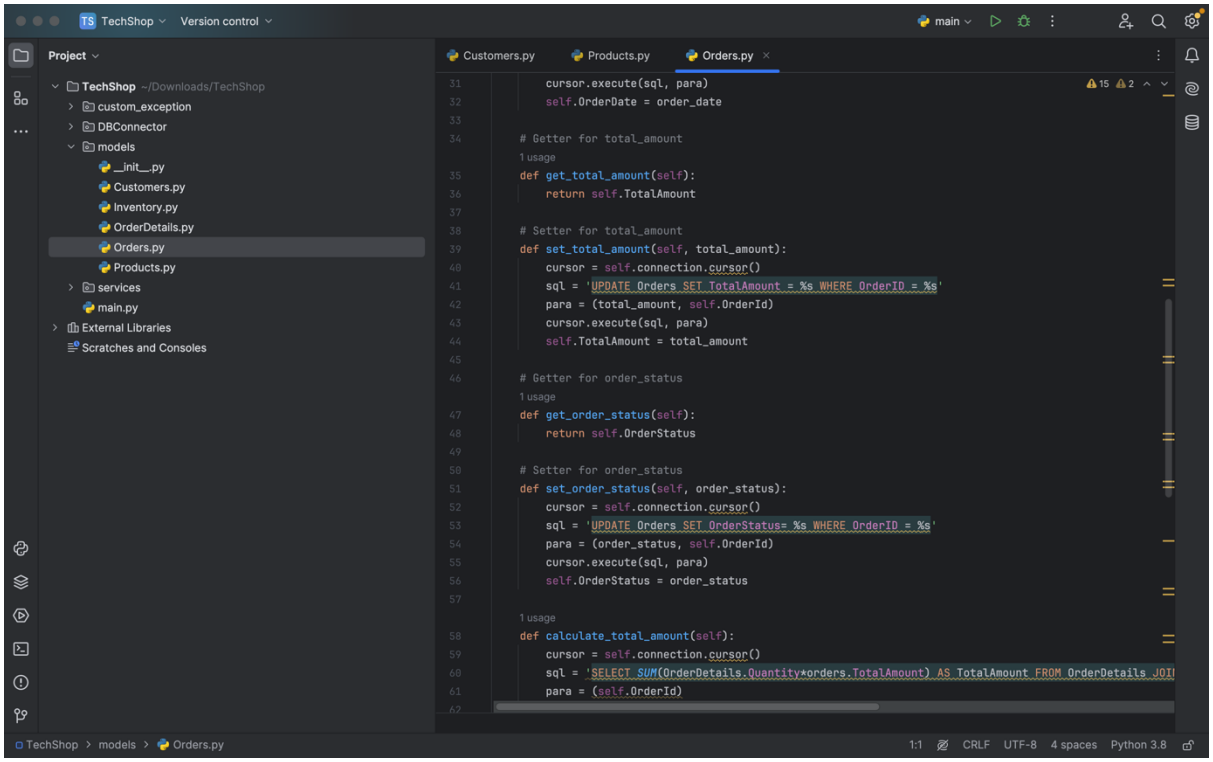

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
from DBConnector.DBConnectivity import getDBConn
18 usages
class Products:
def __init__(self, ProductId, ProductName, Description, Price):
self.connection = getDBConn()
self.ProductId = ProductId
self.ProductName = ProductName
self.Description = Description
self.Price = Price
def get_ProductId(self):
return self.ProductId
def set_ProductId(self, ProductId):
sql = 'UPDATE Products SET ProductId = %s WHERE ProductId = %s'
para = (ProductId, self.ProductId)
cursor = self.connection.cursor()
cursor.execute(sql, para)
self.ProductId = ProductId
def get_ProductName(self):
return self.ProductName
def set_ProductName(self, ProductName):
sql = 'UPDATE Products SET ProductName = %s WHERE ProductId = %s'
para = (ProductName, self.ProductId)
cursor = self.connection.cursor()
cursor.execute(sql, para)
self.ProductName = ProductName
def get_Description(self):
return self.Description
Font size: 13pt
Reset to 13pt
TechShop › models › Products.py
1:1 CRLF UTF-8 4 spaces Python 3.8

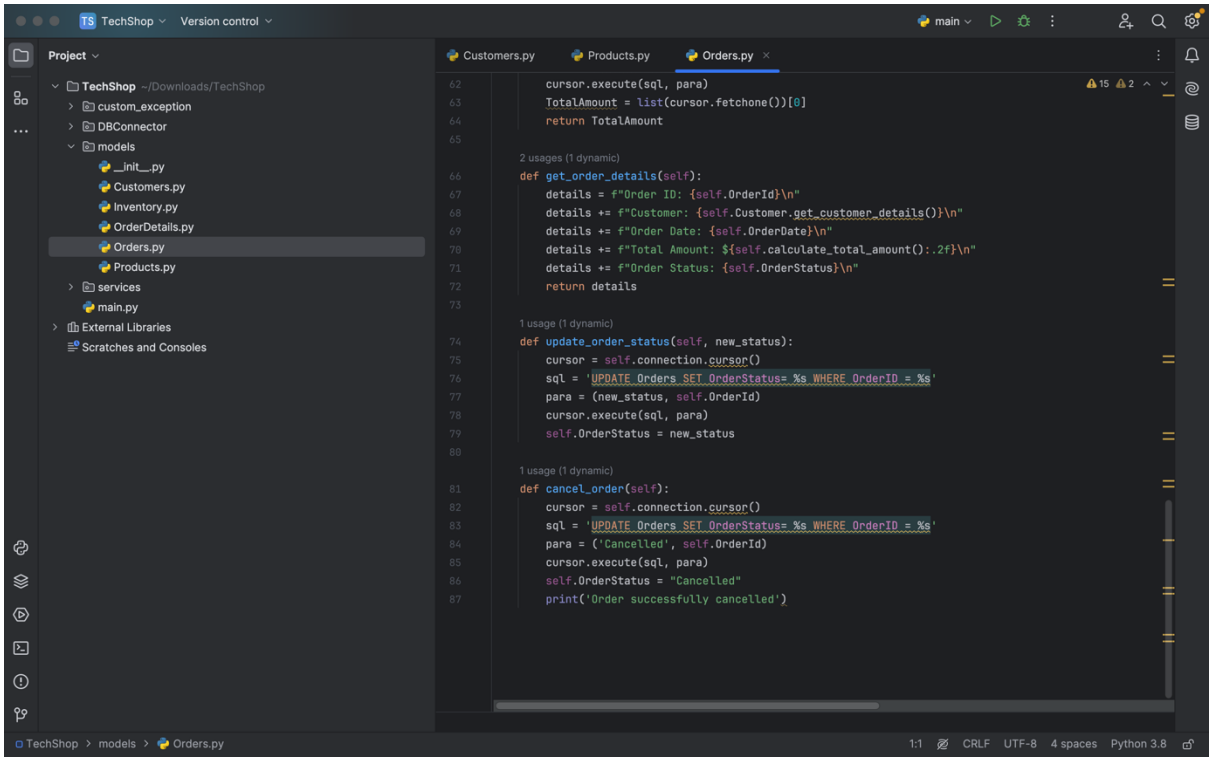

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
details = f"Product ID: {self.ProductId}\n"
details += f"Product Name: {self.ProductName}\n"
details += f"Description: {self.Description}\n"
details += f"Price: ${self.Price:.2f}\n"
return details
2 usages (1 dynamic)
def update_product_info(self, Price=None, Description=None):
if Price:
cursor = self.connection.cursor()
sql = 'UPDATE Products SET Price = %s WHERE ProductID = %s'
para = (Price, self.ProductId)
cursor.execute(sql, para)
self.Price = Price
if Description:
cursor = self.connection.cursor()
sql = 'UPDATE Products SET Description = %s WHERE ProductID = %s'
para = (Description, self.ProductId)
cursor.execute(sql, para)
self.Description = Description
def is_product_in_stock(self):
cursor = self.connection.cursor()
sql = 'SELECT Inventory.QuantityInStock FROM Products JOIN Inventory ON Products.ProductID = Inv
para = (self.ProductId,)
cursor.execute(sql, para)
x = list(cursor.fetchone())[0]
return x > 0
TechShop > models > Products.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
def set_Description(self, Description):
sql = 'UPDATE Products SET Description = %s WHERE ProductID = %s'
para = (Description, self.ProductId)
cursor = self.connection.cursor()
cursor.execute(sql, para)
self.Description = Description
def get_Price(self):
return self.Price
def set_Price(self, Price):
sql = 'UPDATE Products SET Price = %s WHERE productID = %s'
para = (Price, self.ProductId)
cursor = self.connection.cursor()
cursor.execute(sql, para)
self.Price = Price
2 usages (1 dynamic)
def get_product_details(self):
details = f"Product ID: {self.ProductId}\n"
details += f"Product Name: {self.ProductName}\n"
details += f"Description: {self.Description}\n"
details += f"Price: ${self.Price:.2f}\n"
return details
2 usages (1 dynamic)
def update_product_info(self, Price=None, Description=None):
if Price:
cursor = self.connection.cursor()
sql = 'UPDATE Products SET Price = %s WHERE ProductID = %s'
para = (Price, self.ProductId)
cursor.execute(sql, para)
self.Price = Price
TechShop > models > Products.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8

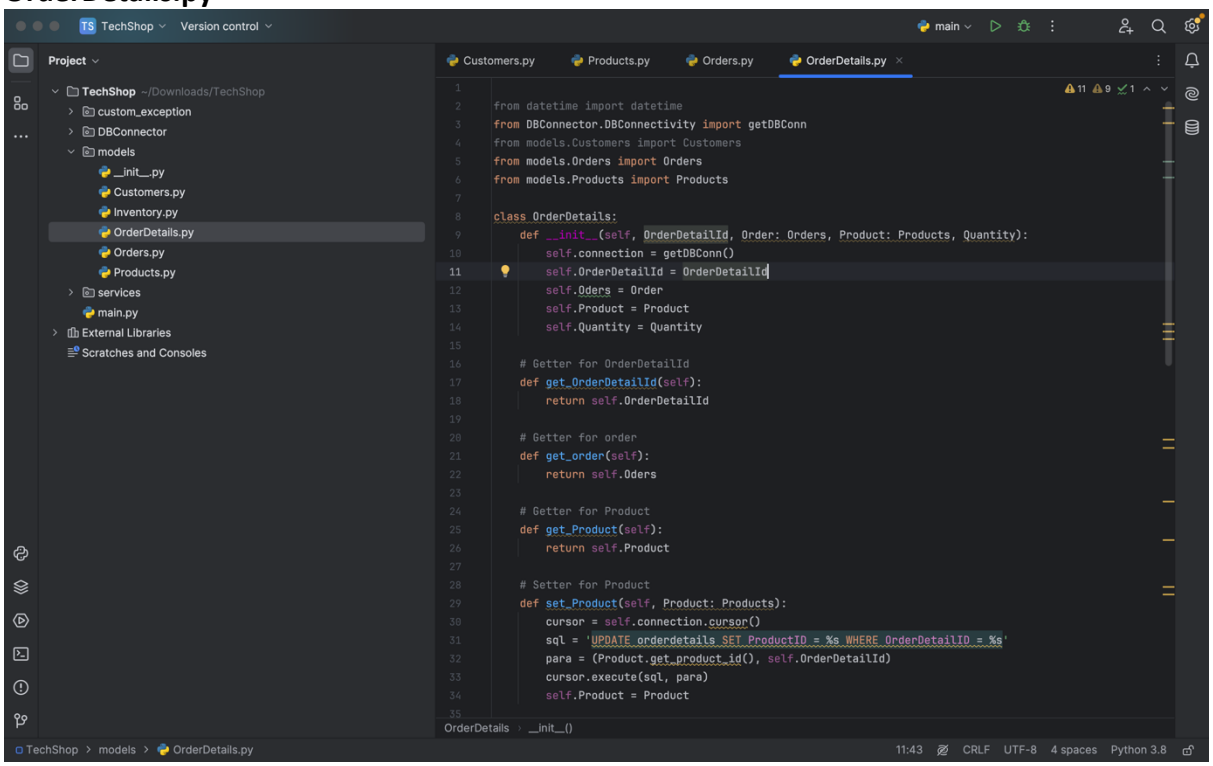
Orders.py:

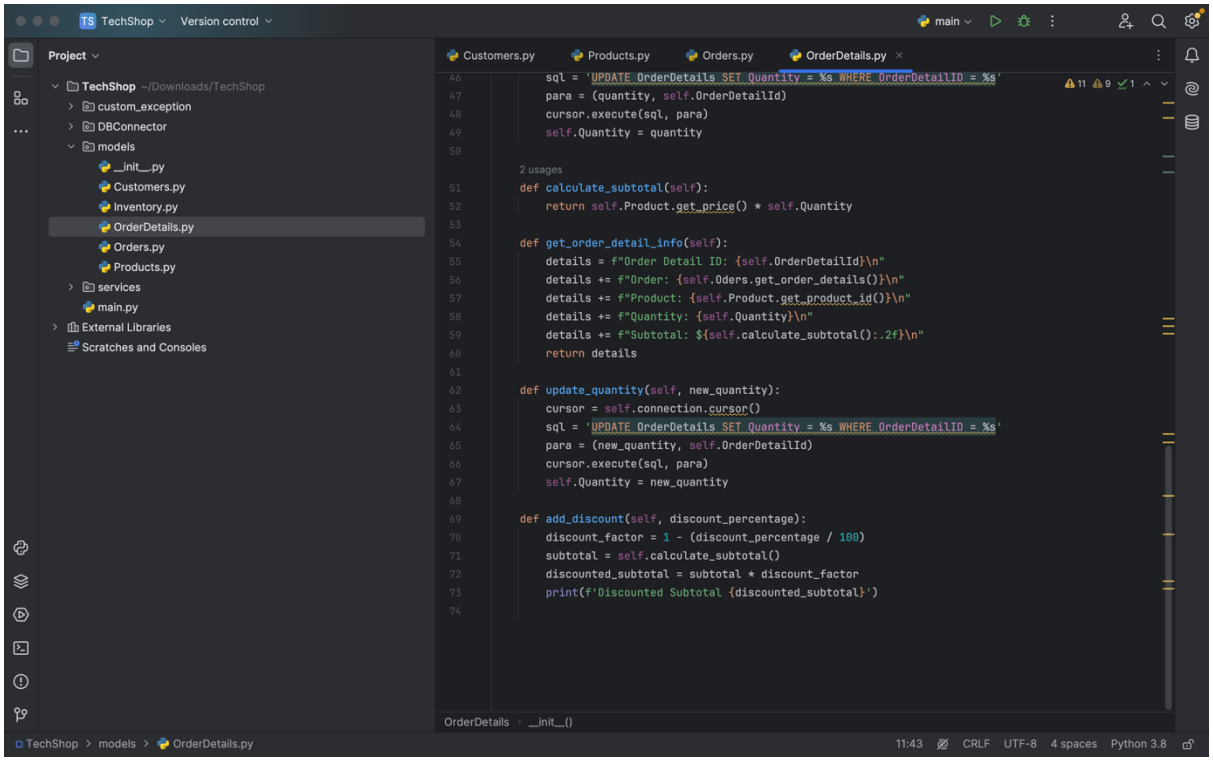

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
Orders.py
from datetime import datetime
from models.Customers import Customers
from DBConnector.DBConnectivity import getDBConn
7 usages
class Orders:
def __init__(self, order_id, customer: Customers, order_date, total_amount, order_status="Pending"):
self.connection = getDBConn()
self.OrderId = order_id
self.Customer = customer
self.OrderDate = order_date
self.TotalAmount = total_amount
self.OrderStatus = order_status
# Getter for order_id
2 usages
def get_order_id(self):
return self.OrderId
Getter for customer
1 usage
def get_customer(self):
return self.Customer
Getter for order_date
1 usage
def get_order_date(self):
return self.OrderDate
# Setter for order_date
def set_order_date(self, order_date):
cursor = self.connection.cursor()
sql = 'UPDATE Orders SET OrderDate = %s WHERE OrderID = %s'
para = (order_date, self.OrderId)
TechShop > models > Orders.py
1:1 CRLF UTF-8 4 spaces Python 3.8

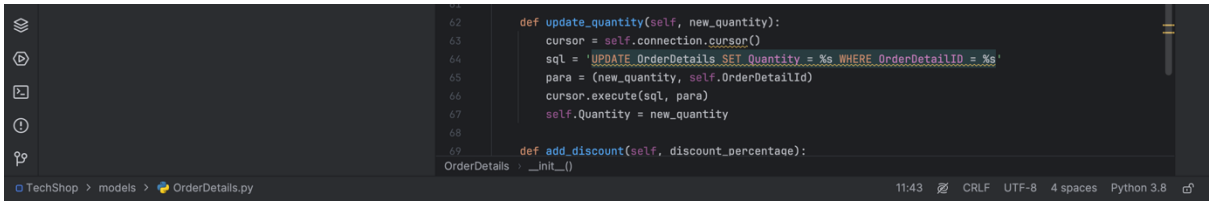

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
Orders.py
cursor.execute(sql, para)
self.OrderDate = order_date
# Getter for total_amount
1 usage
def get_total_amount(self):
return self.TotalAmount
# Setter for total_amount
def set_total_amount(self, total_amount):
cursor = self.connection.cursor()
sql = 'UPDATE Orders SET TotalAmount = %s WHERE OrderID = %s'
para = (total_amount, self.OrderId)
cursor.execute(sql, para)
self.TotalAmount = total_amount
# Getter for order_status
1 usage
def get_order_status(self):
return self.OrderStatus
# Setter for order_status
def set_order_status(self, order_status):
cursor = self.connection.cursor()
sql = 'UPDATE Orders SET OrderStatus= %s WHERE OrderID = %s'
para = (order_status, self.OrderId)
cursor.execute(sql, para)
self.OrderStatus = order_status
1 usage
def calculate_total_amount(self):
cursor = self.connection.cursor()
sql = 'SELECT SUM(OrderDetails.Quantity*orders.TotalAmount) AS TotalAmount FROM OrderDetails JOI
para = (self.OrderId)
TechShop > models > Orders.py
1:1 CRLF UTF-8 4 spaces Python 3.8



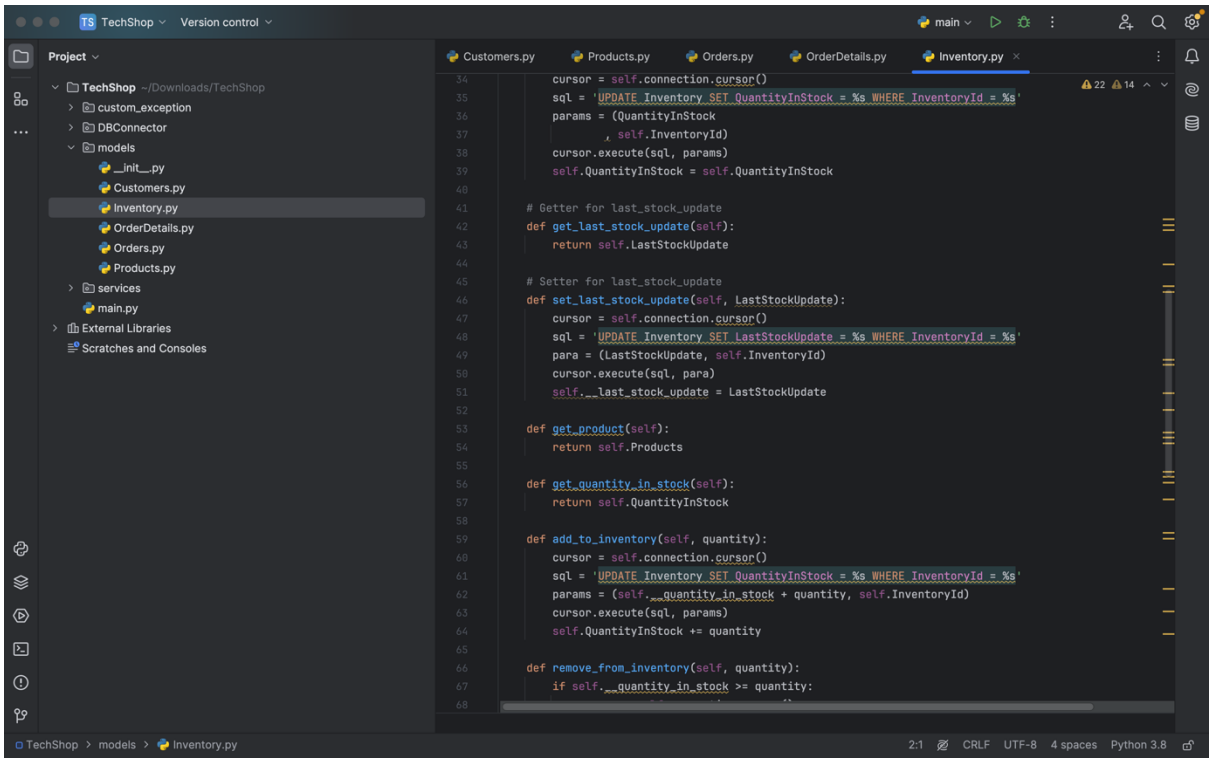
OrderDetails.py

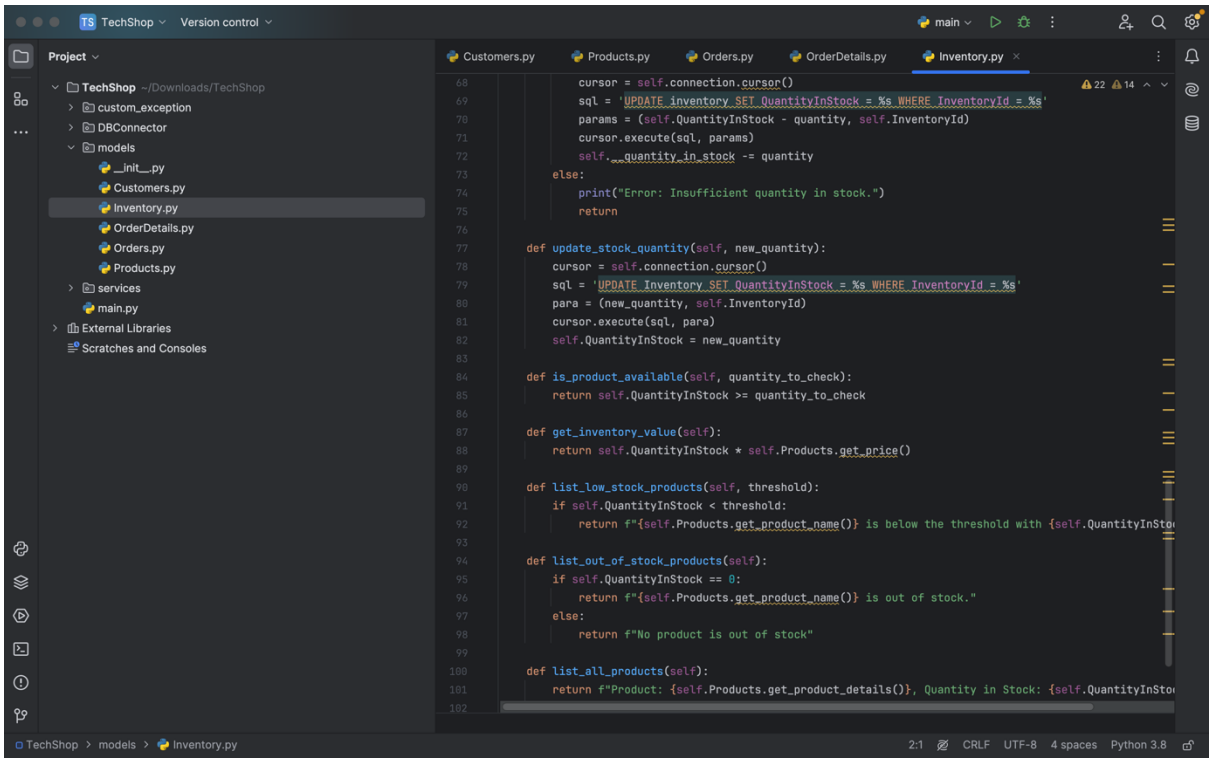


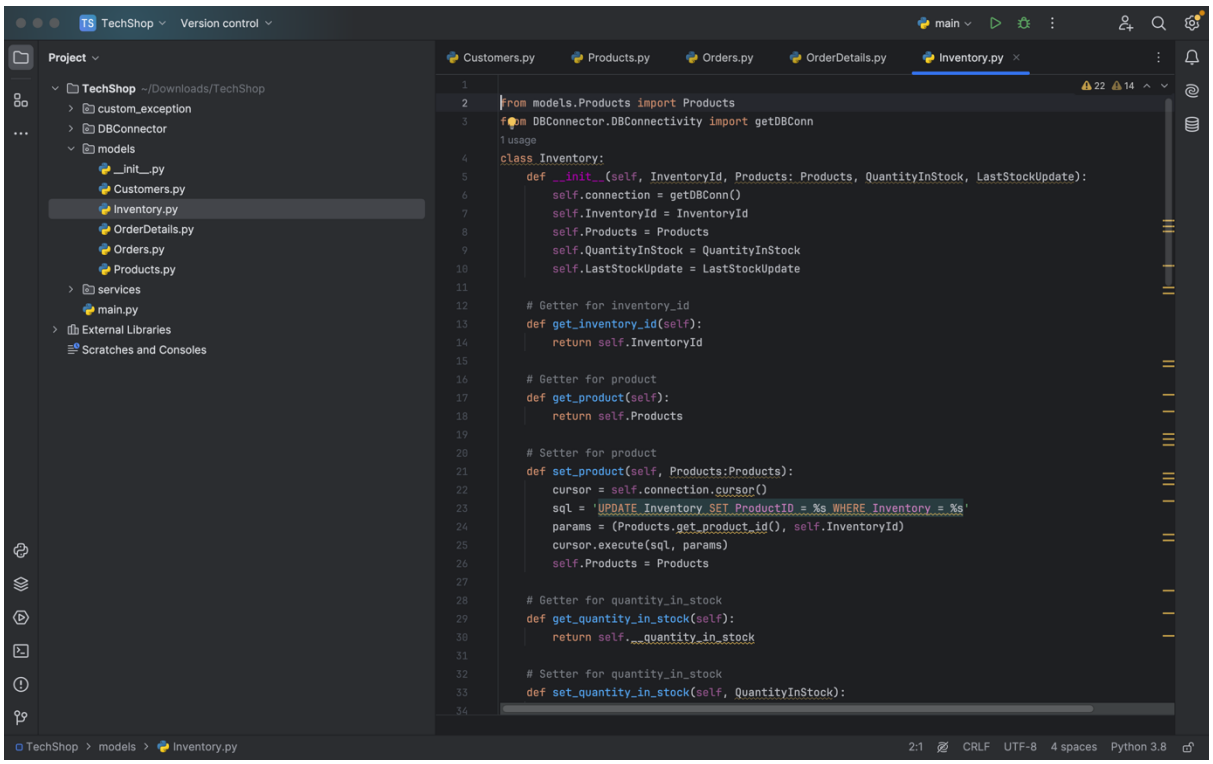




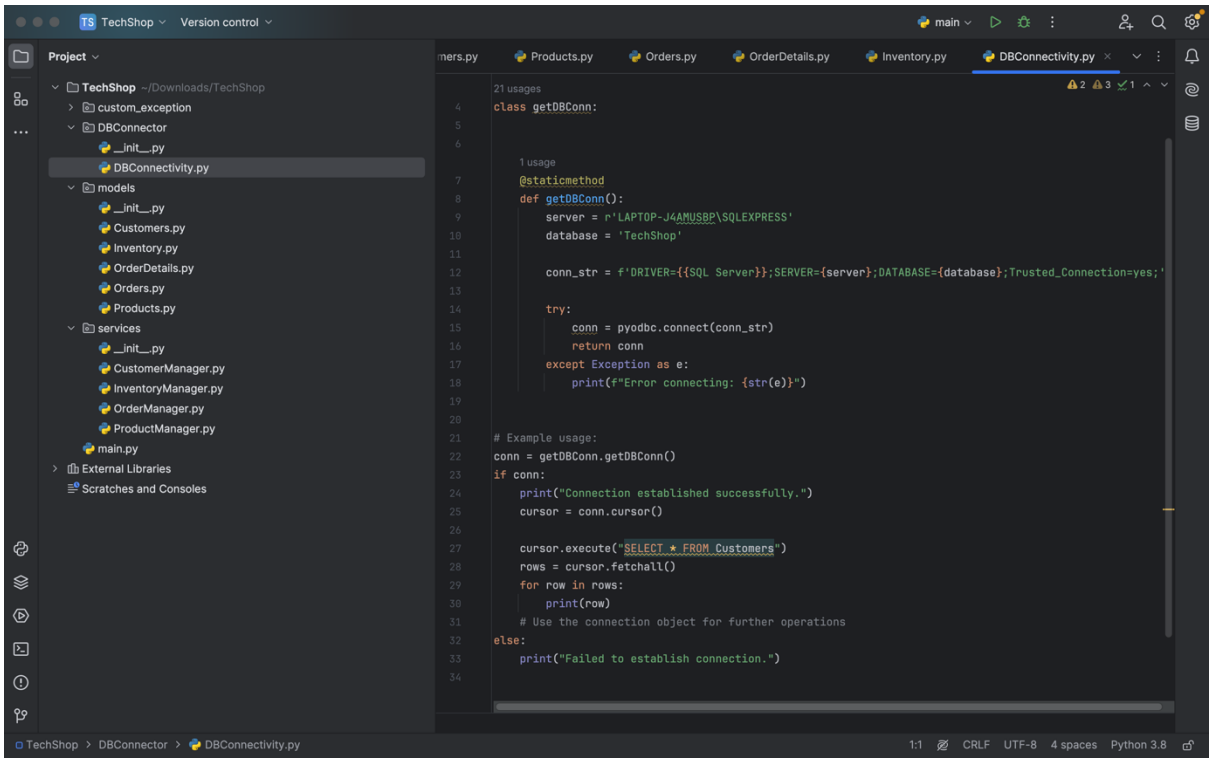
Inventory.py



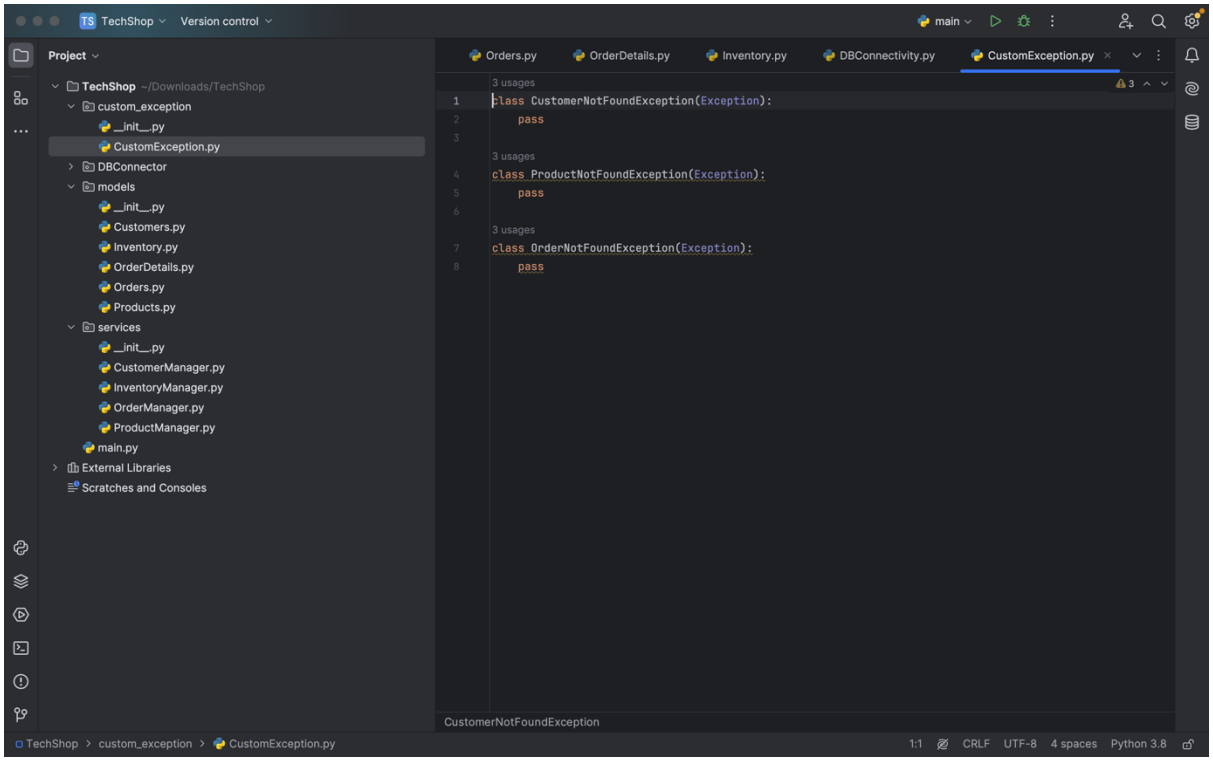

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
Orders.py
OrderDetails.py
Inventory.py
cursor = self.connection.cursor()
sql = 'UPDATE inventory SET QuantityInStock = %s WHERE InventoryId = %s'
params = (self.QuantityInStock - quantity, self.InventoryId)
cursor.execute(sql, params)
self.__quantity_in_stock -= quantity
else:
print("Error: Insufficient quantity in stock.")
return
def update_stock_quantity(self, new_quantity):
cursor = self.connection.cursor()
sql = 'UPDATE Inventory SET QuantityInStock = %s WHERE InventoryId = %s'
para = (new_quantity, self.InventoryId)
cursor.execute(sql, para)
self.QuantityInStock = new_quantity
def is_product_available(self, quantity_to_check):
return self.QuantityInStock >= quantity_to_check
def get_inventory_value(self):
return self.QuantityInStock * self.Products.get_price()
def list_low_stock_products(self, threshold):
if self.QuantityInStock < threshold:
return f"{self.Products.get_product_name()} is below the threshold with {self.QuantityInSto
def list_out_of_stock_products(self):
if self.QuantityInStock == 0:
return f"{self.Products.get_product_name()} is out of stock."
else:
return f"No product is out of stock"
def list_all_products(self):
return f"Product: {self.Products.get_product_details()}, Quantity in Stock: {self.QuantityInSto
TechShop > models > Inventory.py
2:1 CRLF UTF-8 4 spaces Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
main.py
External Libraries
Scratches and Consoles
Customers.py
Products.py
Orders.py
OrderDetails.py
Inventory.py
from models.Products import Products
from DBConnector.DBConnectivity import getDBConn
1 usage
class Inventory:
def __init__(self, InventoryId, Products: Products, QuantityInStock, LastStockUpdate):
self.connection = getDBConn()
self.InventoryId = InventoryId
self.Products = Products
self.QuantityInStock = QuantityInStock
self.LastStockUpdate = LastStockUpdate
# Getter for inventory_id
def get_inventory_id(self):
return self.InventoryId
# Getter for product
def get_product(self):
return self.Products
# Setter for product
def set_product(self, Products:Products):
cursor = self.connection.cursor()
sql = 'UPDATE Inventory SET ProductID = %s WHERE Inventory = %s'
params = (Products.get_product_id(), self.InventoryId)
cursor.execute(sql, params)
self.Products = Products
# Getter for quantity_in_stock
def get_quantity_in_stock(self):
return self.__quantity_in_stock
# Setter for quantity_in_stock
def set_quantity_in_stock(self, QuantityInStock):
TechShop > models > Inventory.py
2:1 CRLF UTF-8 4 spaces Python 3.8

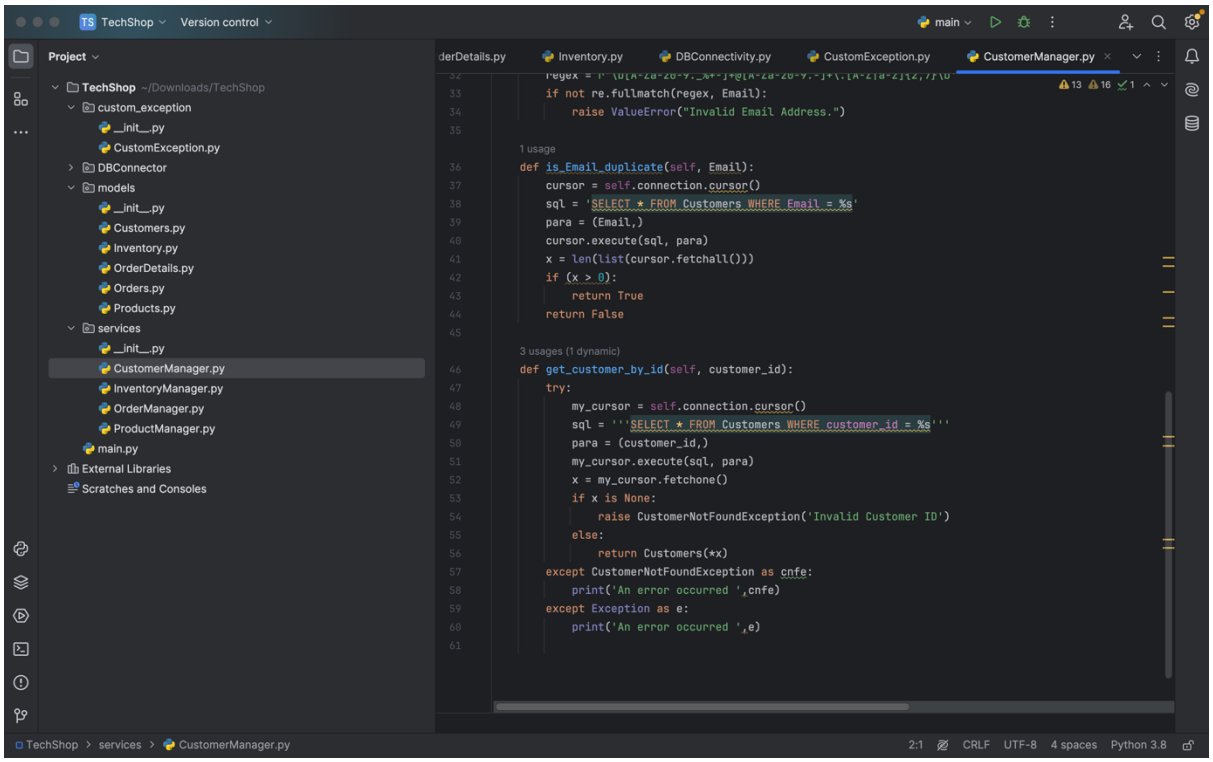
DbConnectivity.py

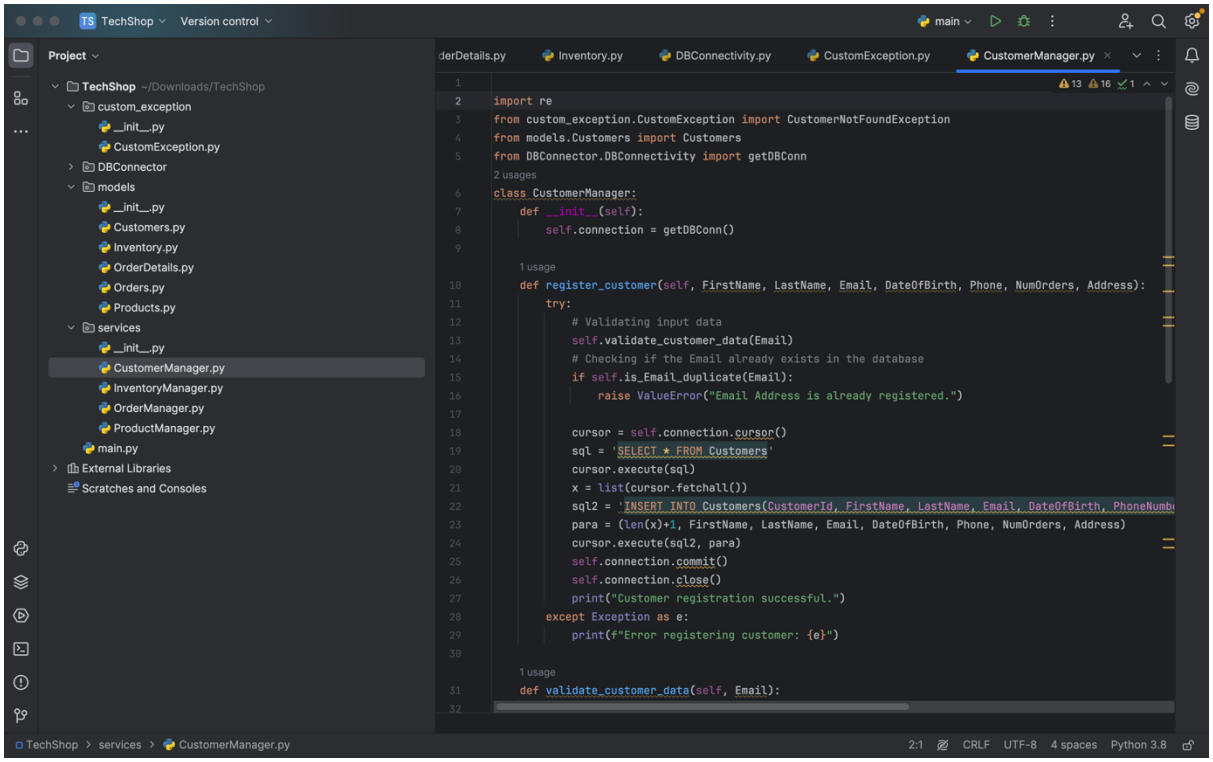


CustomException.py

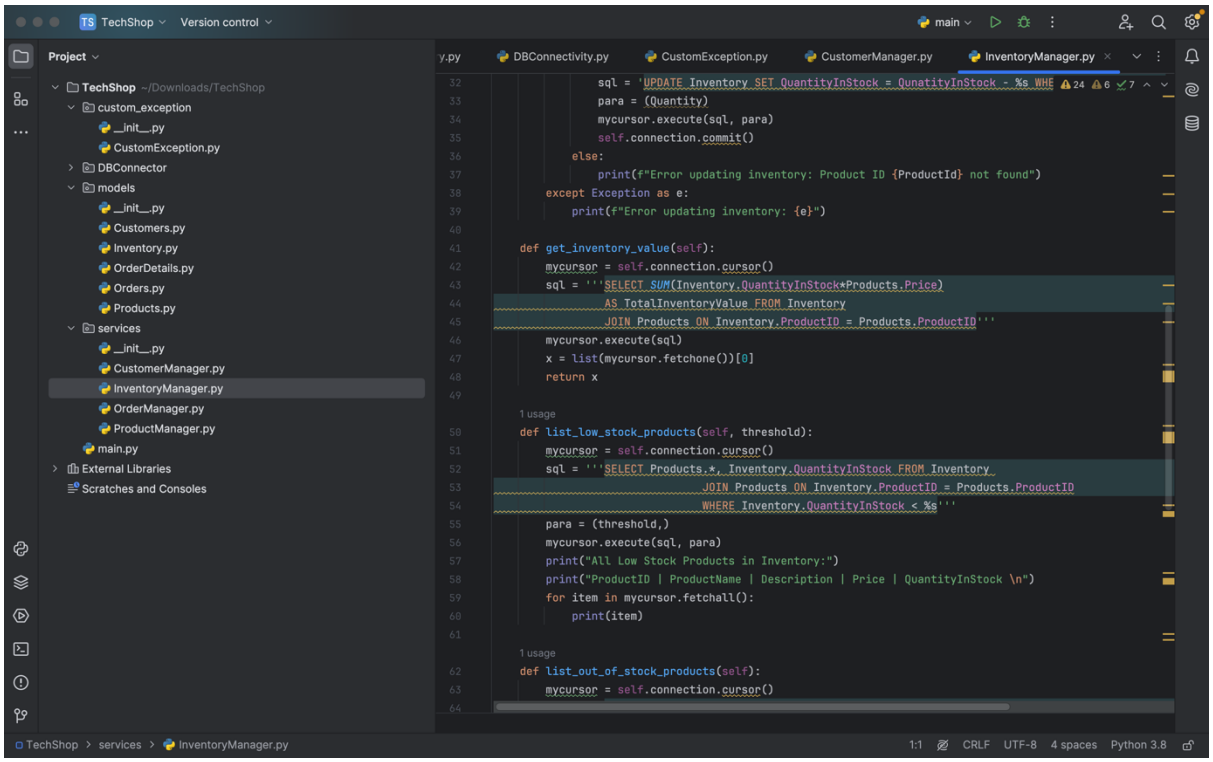


CustomerManager.py

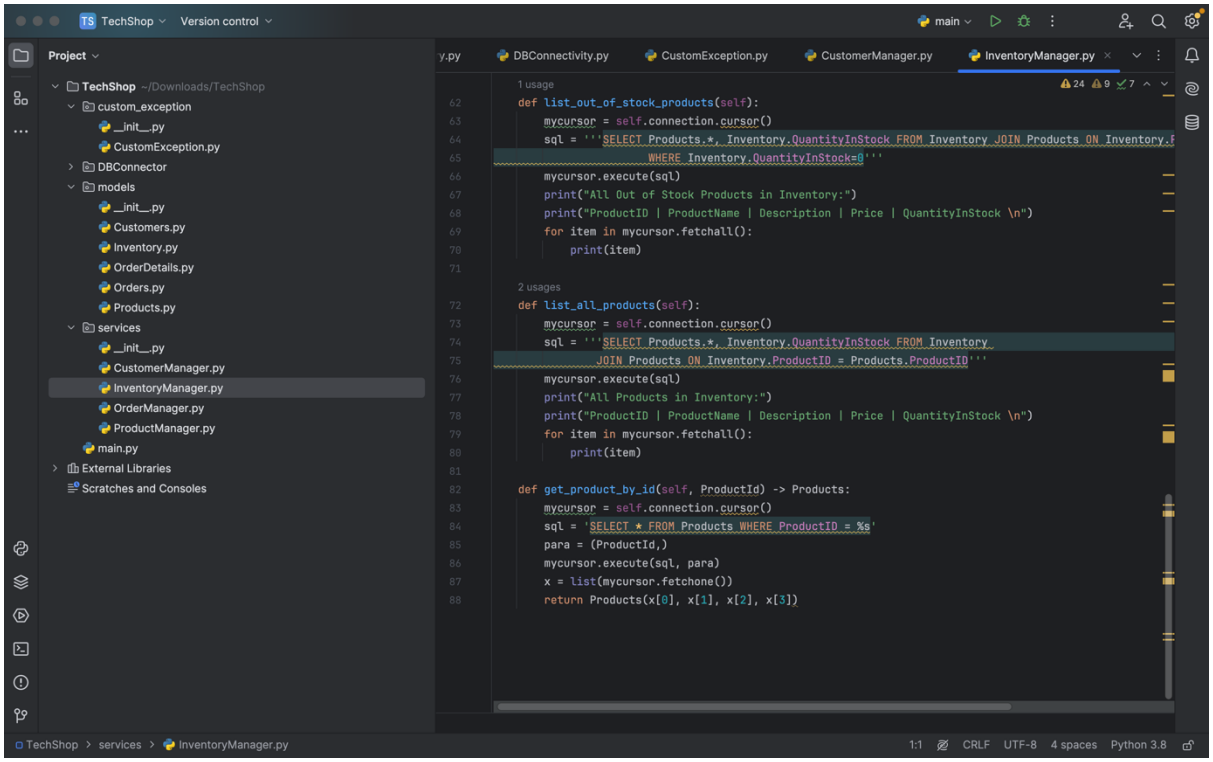




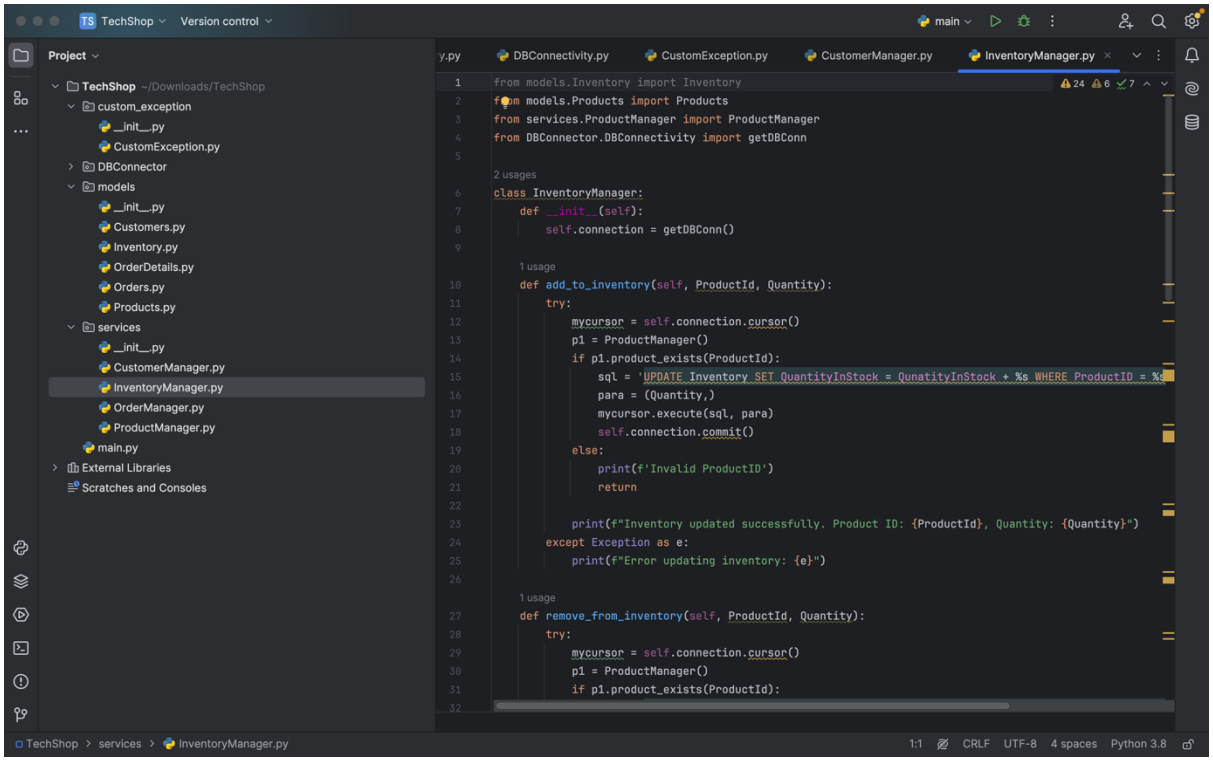
InvenotryManager.py



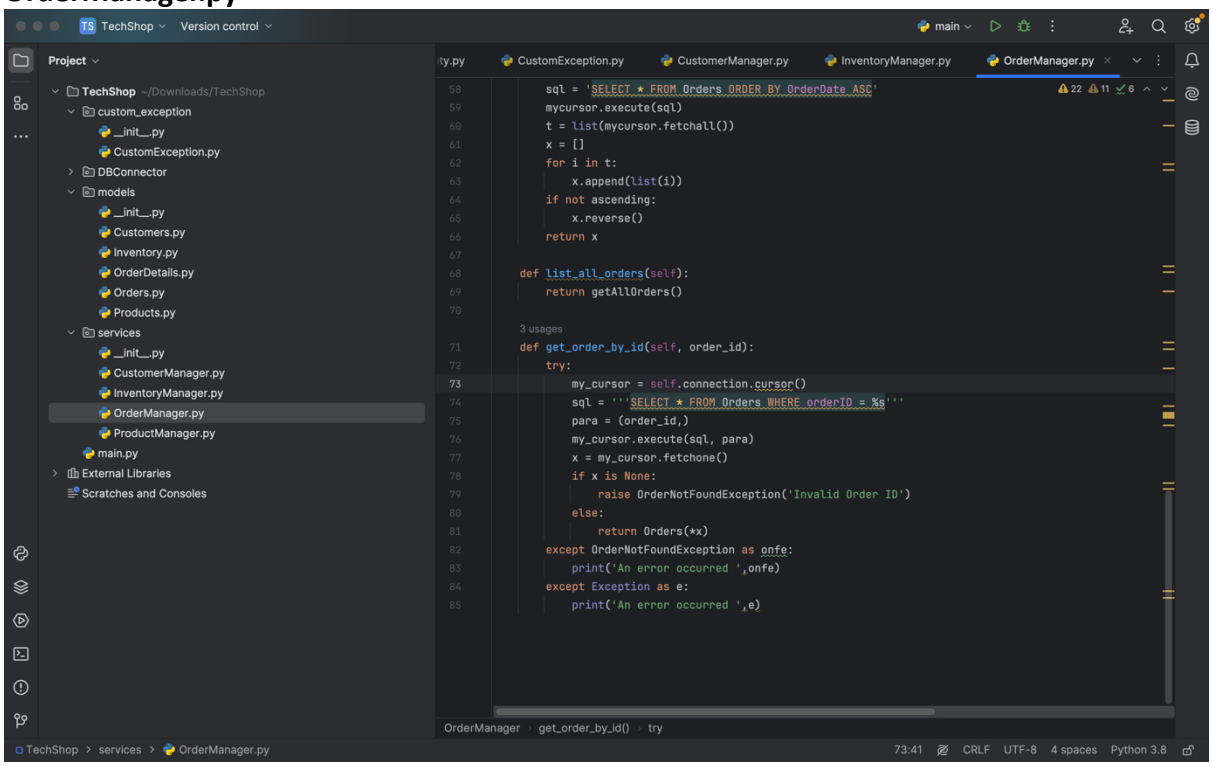
```
                sql = 'UPDATE Inventory SET QuantityInStock = QunatityInStock - %s WHE
                para = (Quantity)
                mycursor.execute(sql, para)
                self.connection.commit()
            else:
                print(f"Error updating inventory: Product ID {ProductId} not found")
        except Exception as e:
            print(f"Error updating inventory: {e}")

    def get_inventory_value(self):
        mycursor = self.connection.cursor()
        sql = '''SELECT SUM(Inventory.QuantityInStock*Products.Price)
                 AS TotalInventoryValue FROM Inventory
                 JOIN Products ON Inventory.ProductID = Products.ProductID'''
        mycursor.execute(sql)
        x = list(mycursor.fetchone())[0]
        return x

    1 usage
    def list_low_stock_products(self, threshold):
        mycursor = self.connection.cursor()
        sql = '''SELECT Products.*, Inventory.QuantityInStock FROM Inventory
                                JOIN Products ON Inventory.ProductID = Products.ProductID
                                WHERE Inventory.QuantityInStock < %s'''
        para = (threshold,)
        mycursor.execute(sql, para)
        print("All Low Stock Products in Inventory:")
        print("ProductID | ProductName | Description | Price | QuantityInStock \n")
        for item in mycursor.fetchall():
            print(item)

    1 usage
    def list_out_of_stock_products(self):
        mycursor = self.connection.cursor()
```

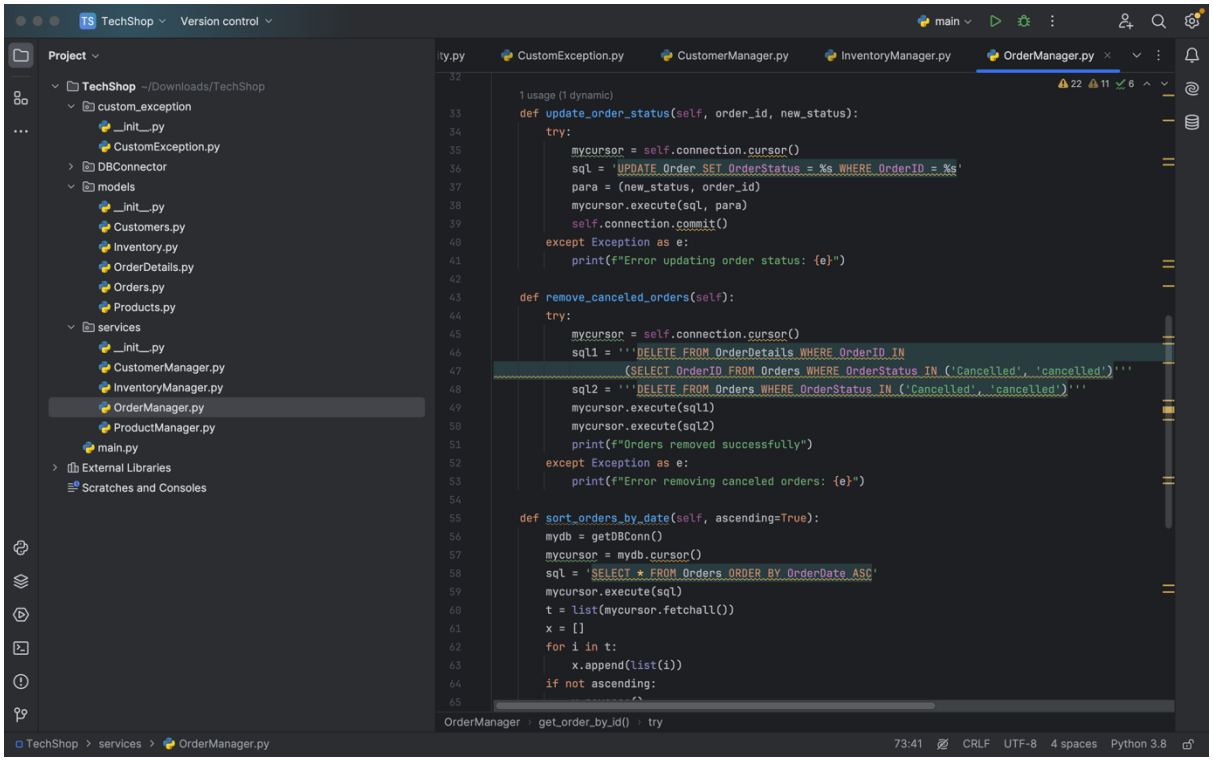


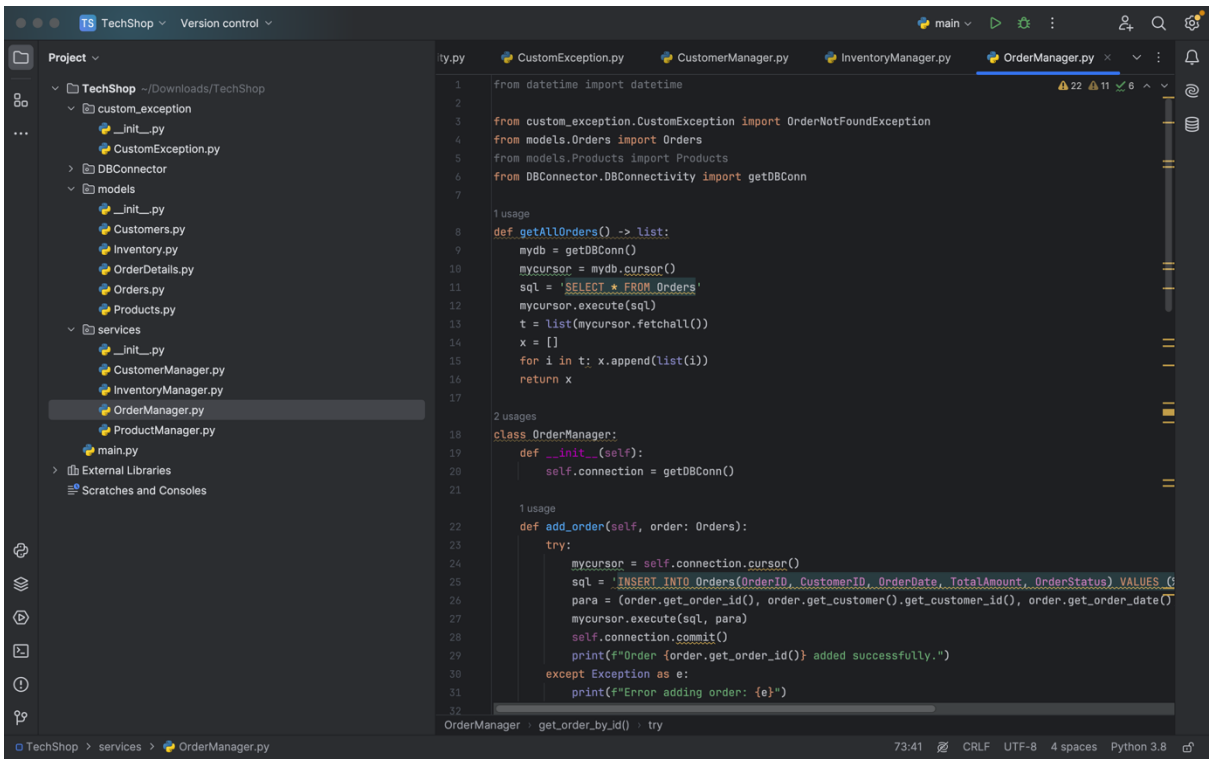
```
    1 usage
    def list_out_of_stock_products(self):
        mycursor = self.connection.cursor()
        sql = '''SELECT Products.*, Inventory.QuantityInStock FROM Inventory JOIN Products ON Inventory.P
                        WHERE Inventory.QuantityInStock=0'''
        mycursor.execute(sql)
        print("All Out of Stock Products in Inventory:")
        print("ProductID | ProductName | Description | Price | QuantityInStock \n")
        for item in mycursor.fetchall():
            print(item)

    2 usages
    def list_all_products(self):
        mycursor = self.connection.cursor()
        sql = '''SELECT Products.*, Inventory.QuantityInStock FROM Inventory
                    JOIN Products ON Inventory.ProductID = Products.ProductID'''
        mycursor.execute(sql)
        print("All Products in Inventory:")
        print("ProductID | ProductName | Description | Price | QuantityInStock \n")
        for item in mycursor.fetchall():
            print(item)

    def get_product_by_id(self, ProductId) -> Products:
        mycursor = self.connection.cursor()
        sql = 'SELECT * FROM Products WHERE ProductID = %s'
        para = (ProductId,)
        mycursor.execute(sql, para)
        x = list(mycursor.fetchone())
        return Products(x[0], x[1], x[2], x[3])
```



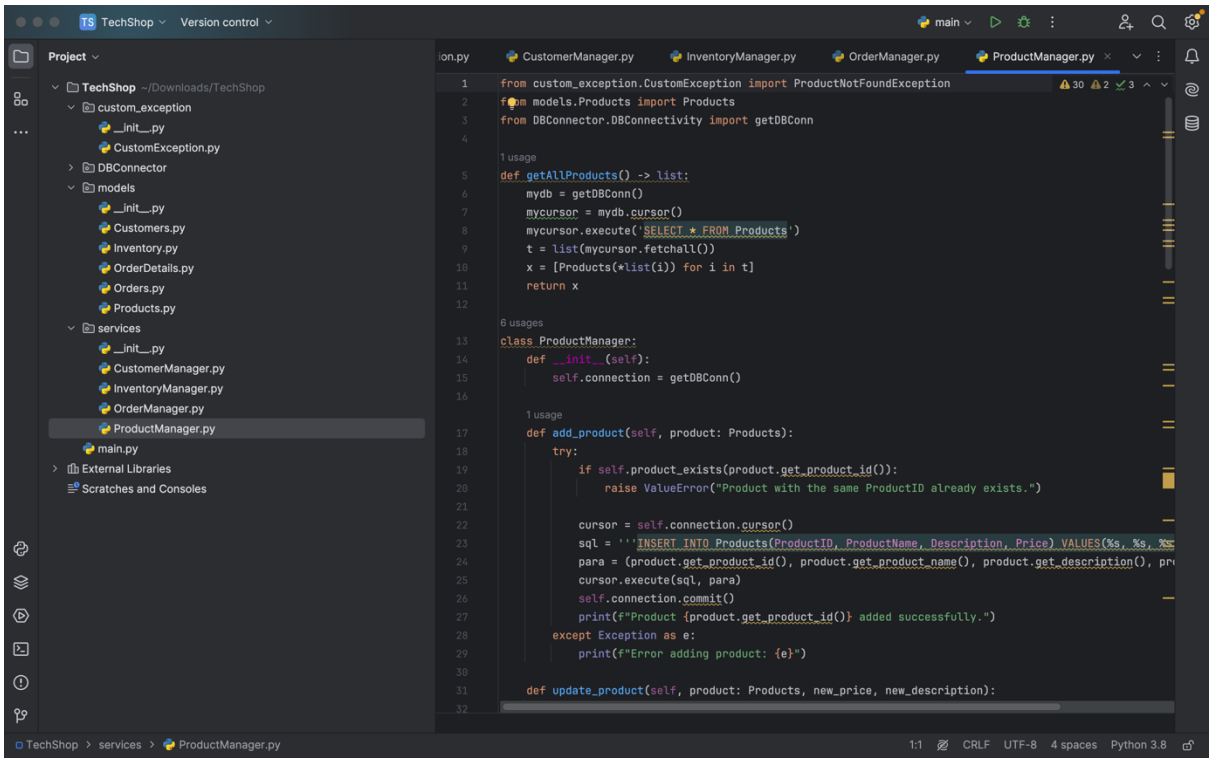
OrderManager.py



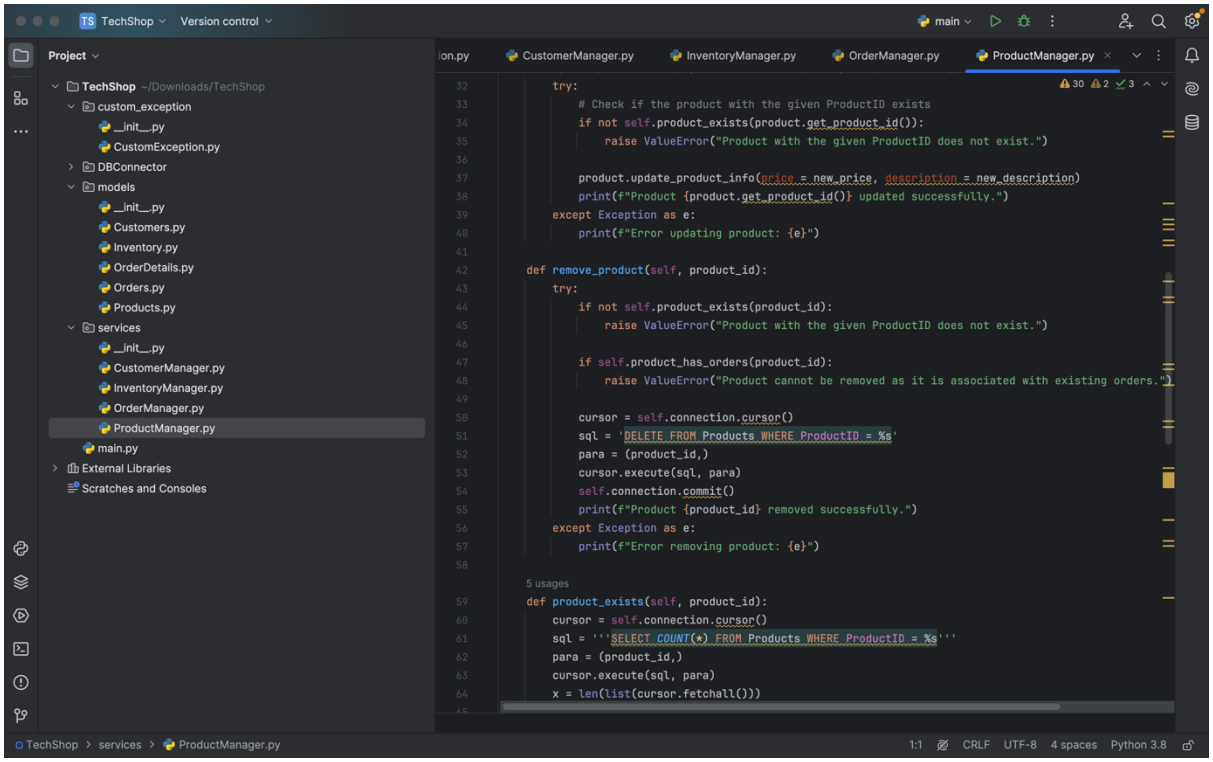

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ity.py
CustomException.py
CustomerManager.py
InventoryManager.py
OrderManager.py
1 usage (1 dynamic)
def update_order_status(self, order_id, new_status):
try:
mycursor = self.connection.cursor()
sql = 'UPDATE Order SET OrderStatus = %s WHERE OrderID = %s'
para = (new_status, order_id)
mycursor.execute(sql, para)
self.connection.commit()
except Exception as e:
print(f"Error updating order status: {e}")
def remove_canceled_orders(self):
try:
mycursor = self.connection.cursor()
sql1 = '''DELETE FROM OrderDetails WHERE OrderID IN
(SELECT OrderID FROM Orders WHERE OrderStatus IN ('Cancelled', 'cancelled')'''
sql2 = '''DELETE FROM Orders WHERE OrderStatus IN ('Cancelled', 'cancelled')'''
mycursor.execute(sql1)
mycursor.execute(sql2)
print(f"Orders removed successfully")
except Exception as e:
print(f"Error removing canceled orders: {e}")
def sort_orders_by_date(self, ascending=True):
mydb = getDBConn()
mycursor = mydb.cursor()
sql = 'SELECT * FROM Orders ORDER BY OrderDate ASC'
mycursor.execute(sql)
t = list(mycursor.fetchall())
x = []
for i in t:
x.append(list(i))
if not ascending:
OrderManager › get_order_by_id() › try
TechShop › services › OrderManager.py
73:41 CRLF UTF-8 4 spaces Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ity.py
CustomException.py
CustomerManager.py
InventoryManager.py
OrderManager.py
from datetime import datetime
from custom_exception.CustomException import OrderNotFoundException
from models.Orders import Orders
from models.Products import Products
from DBConnector.DBConnectivity import getDBConn
1 usage
def getAllOrders() -> list:
mydb = getDBConn()
mycursor = mydb.cursor()
sql = 'SELECT * FROM Orders'
mycursor.execute(sql)
t = list(mycursor.fetchall())
x = []
for i in t: x.append(list(i))
return x
2 usages
class OrderManager:
def __init__(self):
self.connection = getDBConn()
1 usage
def add_order(self, order: Orders):
try:
mycursor = self.connection.cursor()
sql = 'INSERT INTO Orders(OrderID, CustomerID, OrderDate, TotalAmount, OrderStatus) VALUES (%
para = (order.get_order_id(), order.get_customer().get_customer_id(), order.get_order_date()
mycursor.execute(sql, para)
self.connection.commit()
print(f"Order {order.get_order_id()} added successfully.")
except Exception as e:
print(f"Error adding order: {e}")
OrderManager › get_order_by_id() › try
TechShop › services › OrderManager.py
73:41 CRLF UTF-8 4 spaces Python 3.8

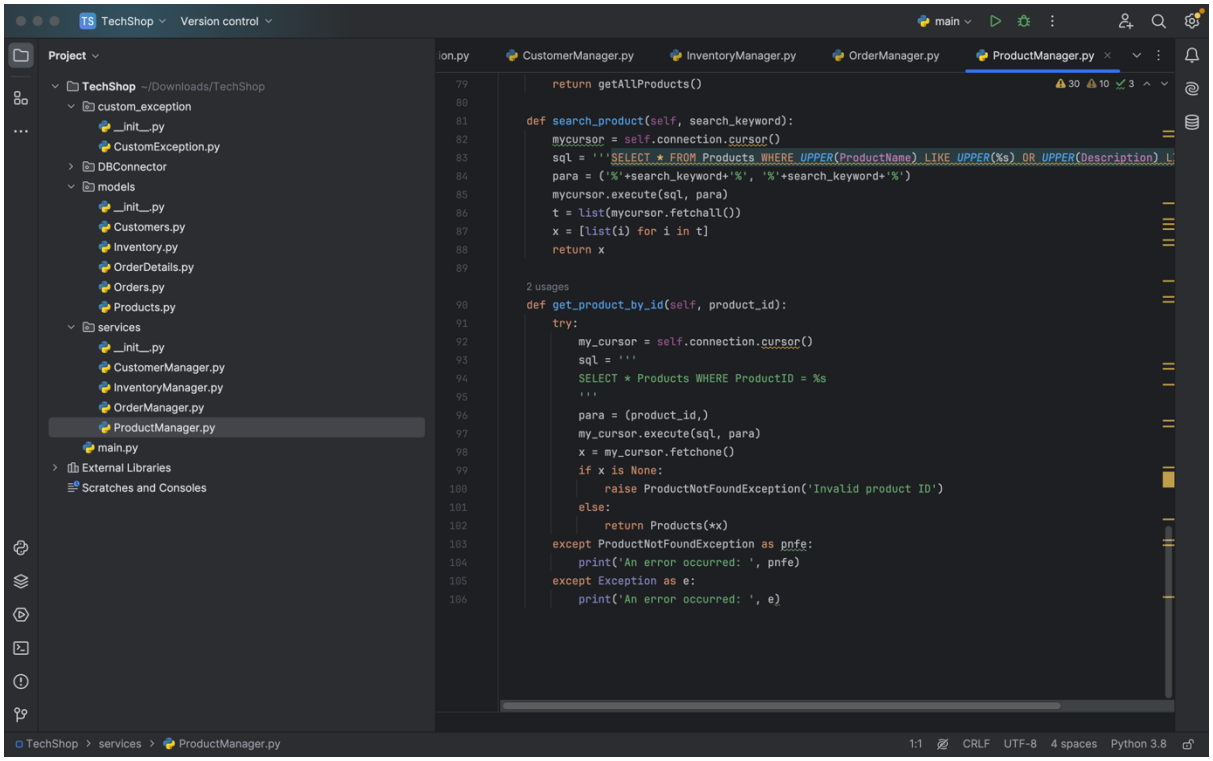
ProductManager.py

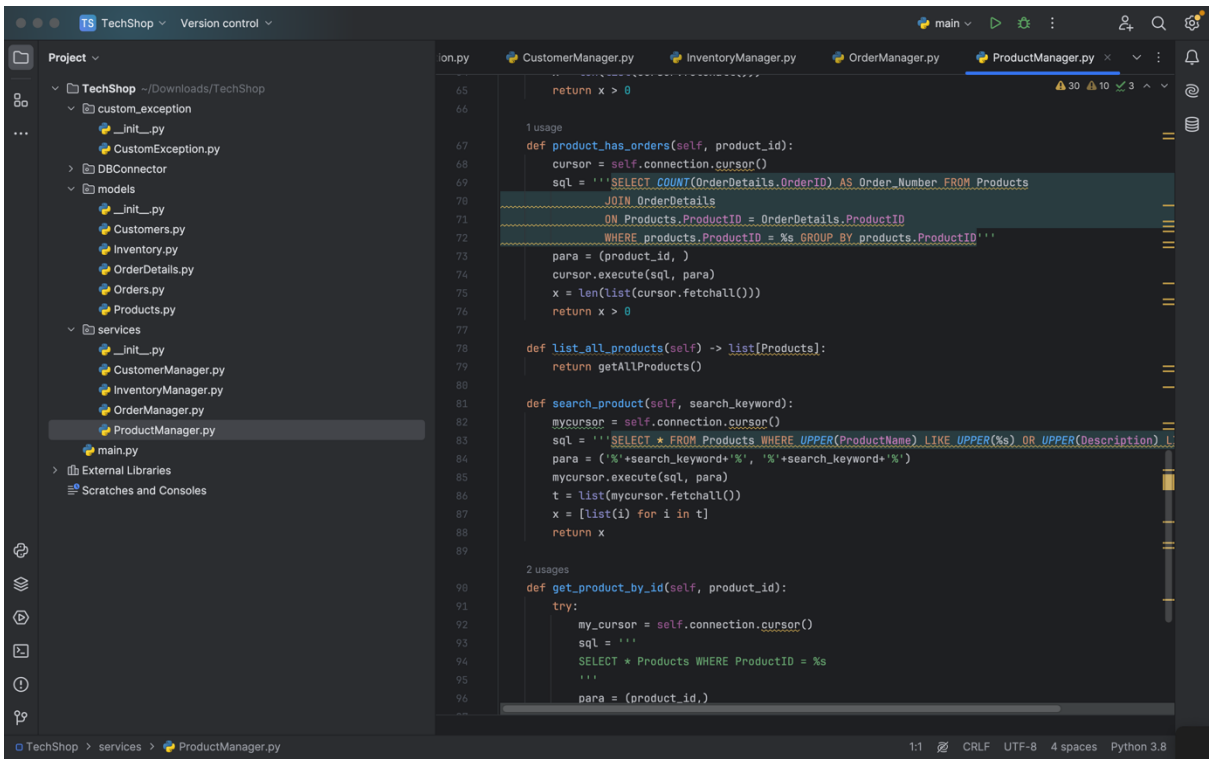


```python
from custom_exception.CustomException import ProductNotFoundException
from models.Products import Products
from DBConnector.DBConnectivity import getDBConn

1 usage
def getAllProducts() -> list:
    mydb = getDBConn()
    mycursor = mydb.cursor()
    mycursor.execute('SELECT * FROM Products')
    t = list(mycursor.fetchall())
    x = [Products(*list(i)) for i in t]
    return x

6 usages
class ProductManager:
    def __init__(self):
        self.connection = getDBConn()

    1 usage
    def add_product(self, product: Products):
        try:
            if self.product_exists(product.get_product_id()):
                raise ValueError("Product with the same ProductID already exists.")

            cursor = self.connection.cursor()
            sql = '''INSERT INTO Products(ProductID, ProductName, Description, Price) VALUES(%s, %s, %s
            para = (product.get_product_id(), product.get_product_name(), product.get_description(), pr
            cursor.execute(sql, para)
            self.connection.commit()
            print(f"Product {product.get_product_id()} added successfully.")
        except Exception as e:
            print(f"Error adding product: {e}")

    def update_product(self, product: Products, new_price, new_description):
```

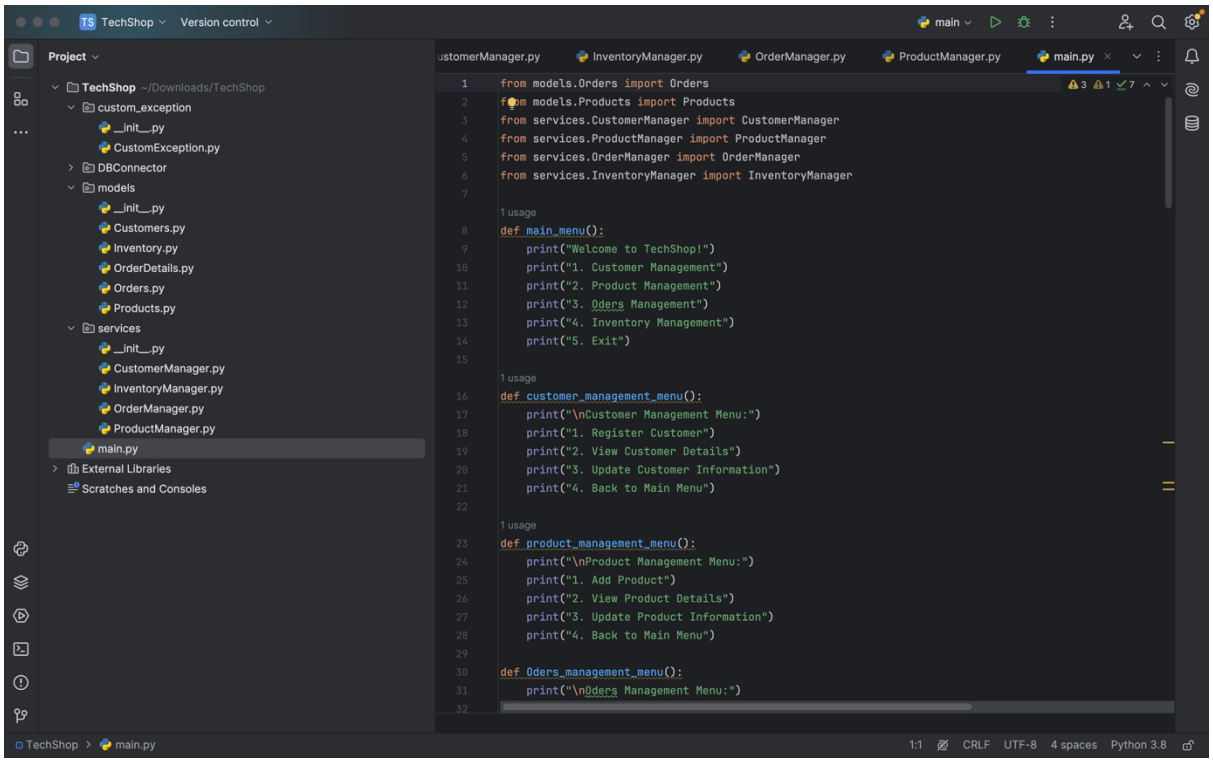


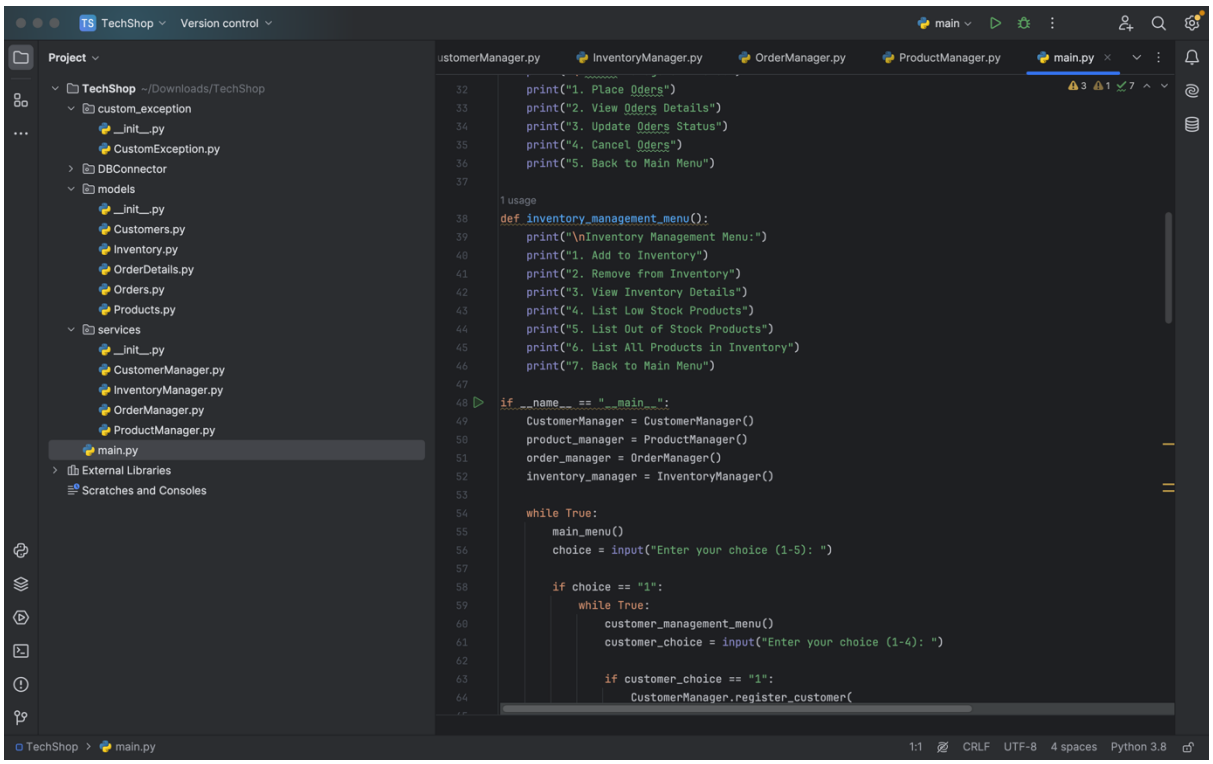
```python
        try:
            # Check if the product with the given ProductID exists
            if not self.product_exists(product.get_product_id()):
                raise ValueError("Product with the given ProductID does not exist.")

            product.update_product_info(price = new_price, description = new_description)
            print(f"Product {product.get_product_id()} updated successfully.")
        except Exception as e:
            print(f"Error updating product: {e}")

    def remove_product(self, product_id):
        try:
            if not self.product_exists(product_id):
                raise ValueError("Product with the given ProductID does not exist.")

            if self.product_has_orders(product_id):
                raise ValueError("Product cannot be removed as it is associated with existing orders.")

            cursor = self.connection.cursor()
            sql = 'DELETE FROM Products WHERE ProductID = %s'
            para = (product_id,)
            cursor.execute(sql, para)
            self.connection.commit()
            print(f"Product {product_id} removed successfully.")
        except Exception as e:
            print(f"Error removing product: {e}")

    5 usages
    def product_exists(self, product_id):
        cursor = self.connection.cursor()
        sql = '''SELECT COUNT(*) FROM Products WHERE ProductID = %s'''
        para = (product_id,)
        cursor.execute(sql, para)
        x = len(list(cursor.fetchall()))
```

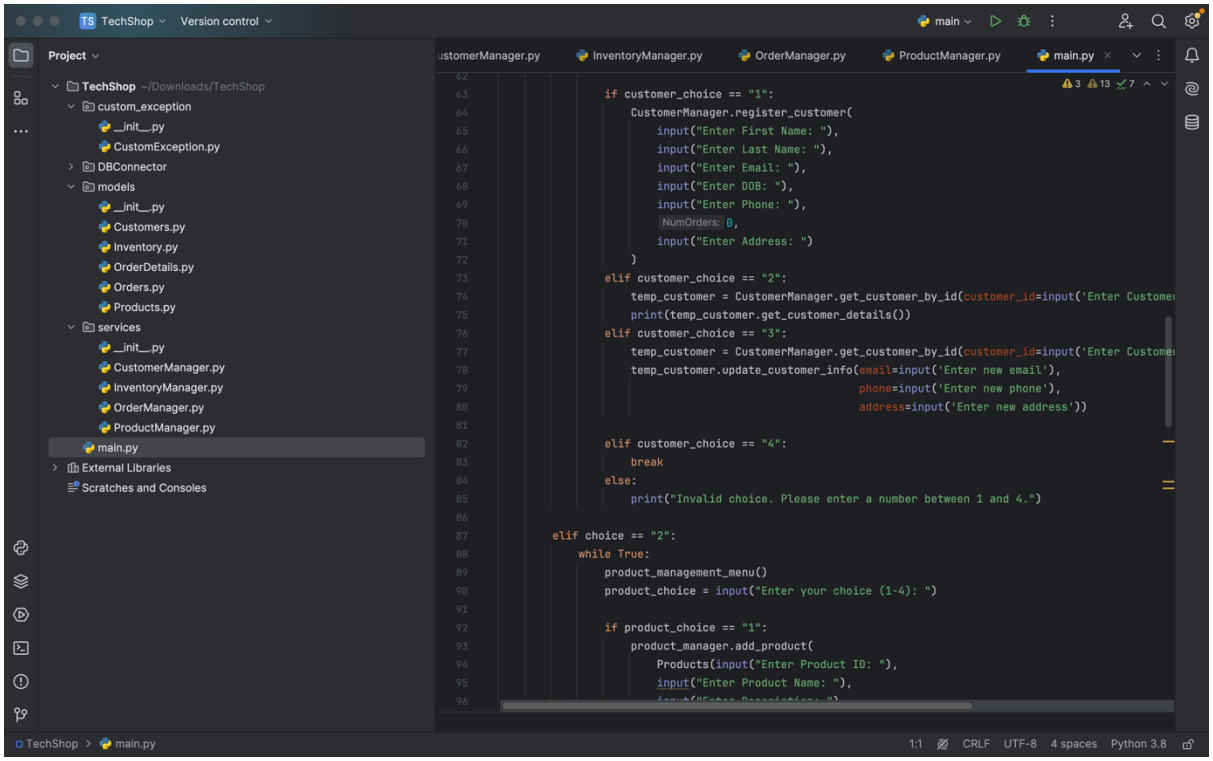

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ion.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
return getAllProducts()
def search_product(self, search_keyword):
mycursor = self.connection.cursor()
sql = '''SELECT * FROM Products WHERE UPPER(ProductName) LIKE UPPER(%s) OR UPPER(Description) L
para = ('%'+search_keyword+'%', '%'+search_keyword+'%')
mycursor.execute(sql, para)
t = list(mycursor.fetchall())
x = [list(i) for i in t]
return x
2 usages
def get_product_by_id(self, product_id):
try:
my_cursor = self.connection.cursor()
sql = '''
SELECT * Products WHERE ProductID = %s
'''
para = (product_id,)
my_cursor.execute(sql, para)
x = my_cursor.fetchone()
if x is None:
raise ProductNotFoundException('Invalid product ID')
else:
return Products(*x)
except ProductNotFoundException as pnfe:
print('An error occurred: ', pnfe)
except Exception as e:
print('An error occurred: ', e)
TechShop > services > ProductManager.py
1:1 CRLF UTF-8 4 spaces Python 3.8

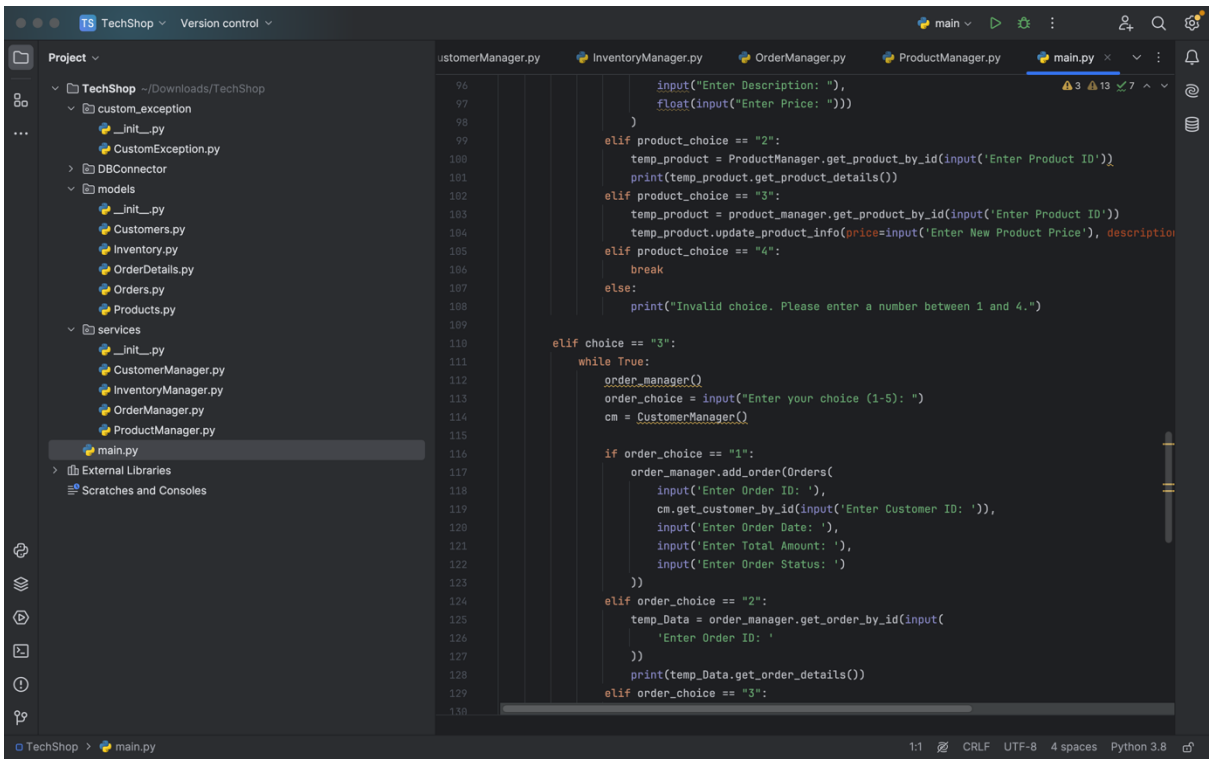

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ion.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
return x > 0
1 usage
def product_has_orders(self, product_id):
cursor = self.connection.cursor()
sql = '''SELECT COUNT(OrderDetails.OrderID) AS Order_Number FROM Products
JOIN OrderDetails
ON Products.ProductID = OrderDetails.ProductID
WHERE products.ProductID = %s GROUP BY products.ProductID'''
para = (product_id,)
cursor.execute(sql, para)
x = len(list(cursor.fetchall()))
return x > 0
def list_all_products(self) -> list[Products]:
return getAllProducts()
def search_product(self, search_keyword):
mycursor = self.connection.cursor()
sql = '''SELECT * FROM Products WHERE UPPER(ProductName) LIKE UPPER(%s) OR UPPER(Description) L
para = ('%'+search_keyword+'%', '%'+search_keyword+'%')
mycursor.execute(sql, para)
t = list(mycursor.fetchall())
x = [list(i) for i in t]
return x
2 usages
def get_product_by_id(self, product_id):
try:
my_cursor = self.connection.cursor()
sql = '''
SELECT * Products WHERE ProductID = %s
'''
para = (product_id,)
TechShop > services > ProductManager.py
1:1 CRLF UTF-8 4 spaces Python 3.8

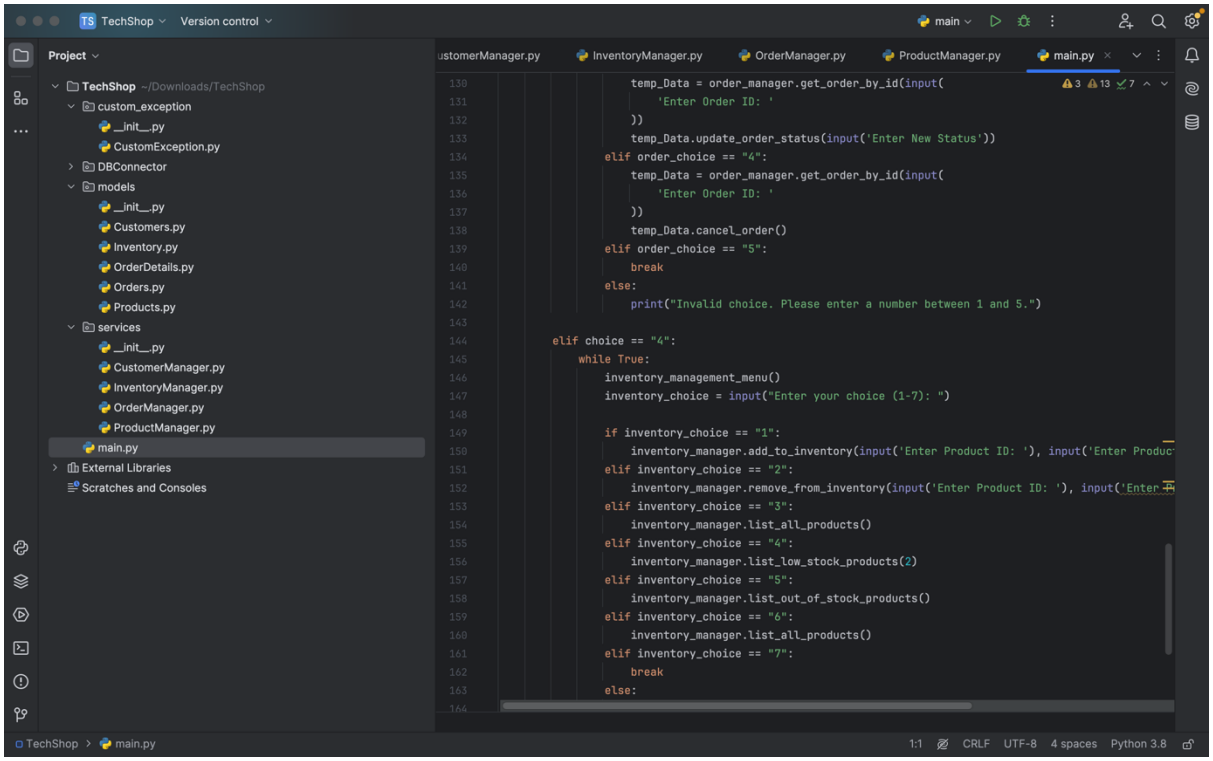
Main.py

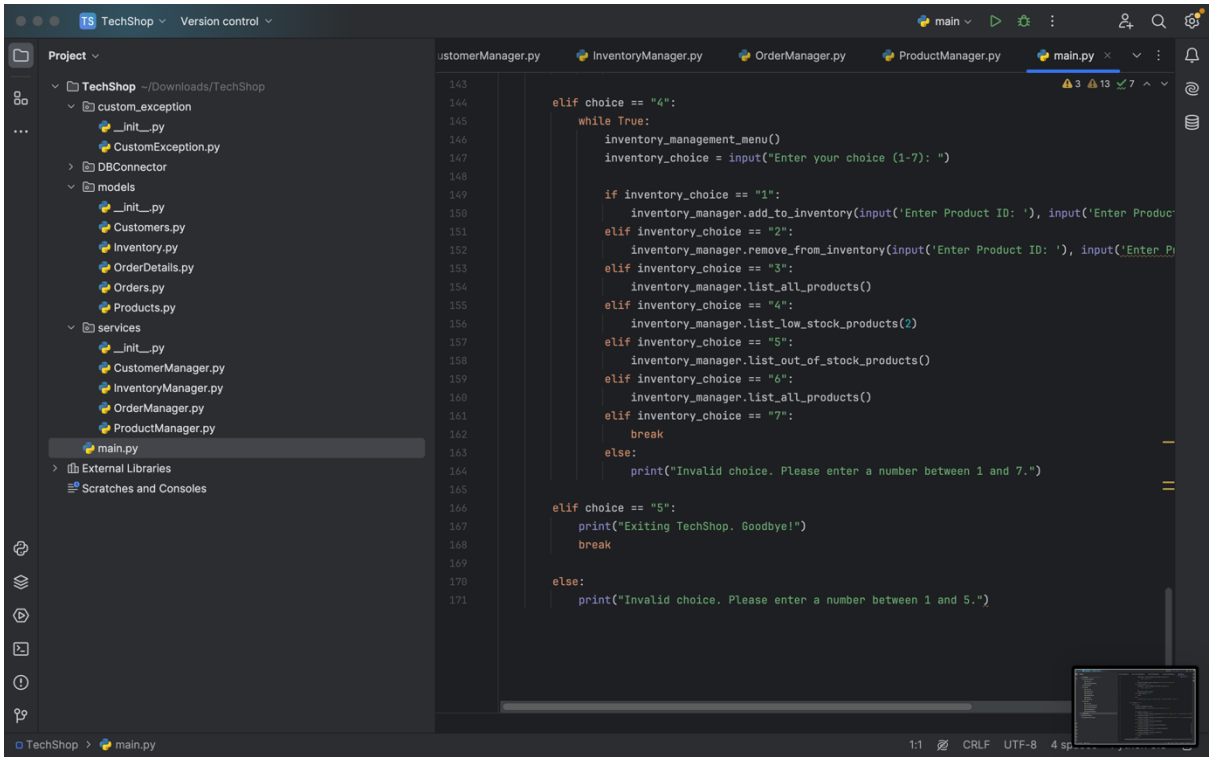

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
from models.Orders import Orders
from models.Products import Products
from services.CustomerManager import CustomerManager
from services.ProductManager import ProductManager
from services.OrderManager import OrderManager
from services.InventoryManager import InventoryManager
1 usage
def main_menu():
print("Welcome to TechShop!")
print("1. Customer Management")
print("2. Product Management")
print("3. Oders Management")
print("4. Inventory Management")
print("5. Exit")
1 usage
def customer_management_menu():
print("\nCustomer Management Menu:")
print("1. Register Customer")
print("2. View Customer Details")
print("3. Update Customer Information")
print("4. Back to Main Menu")
1 usage
def product_management_menu():
print("\nProduct Management Menu:")
print("1. Add Product")
print("2. View Product Details")
print("3. Update Product Information")
print("4. Back to Main Menu")
def Oders_management_menu():
print("\nOders Management Menu:")
TechShop
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
print("1. Place Oders")
print("2. View Oders Details")
print("3. Update Oders Status")
print("4. Cancel Oders")
print("5. Back to Main Menu")
1 usage
def inventory_management_menu():
print("\nInventory Management Menu:")
print("1. Add to Inventory")
print("2. Remove from Inventory")
print("3. View Inventory Details")
print("4. List Low Stock Products")
print("5. List Out of Stock Products")
print("6. List All Products in Inventory")
print("7. Back to Main Menu")
if __name__ == "__main__":
CustomerManager = CustomerManager()
product_manager = ProductManager()
order_manager = OrderManager()
inventory_manager = InventoryManager()
while True:
main_menu()
choice = input("Enter your choice (1-5): ")
if choice == "1":
while True:
customer_management_menu()
customer_choice = input("Enter your choice (1-4): ")
if customer_choice == "1":
CustomerManager.register_customer(
TechShop
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8

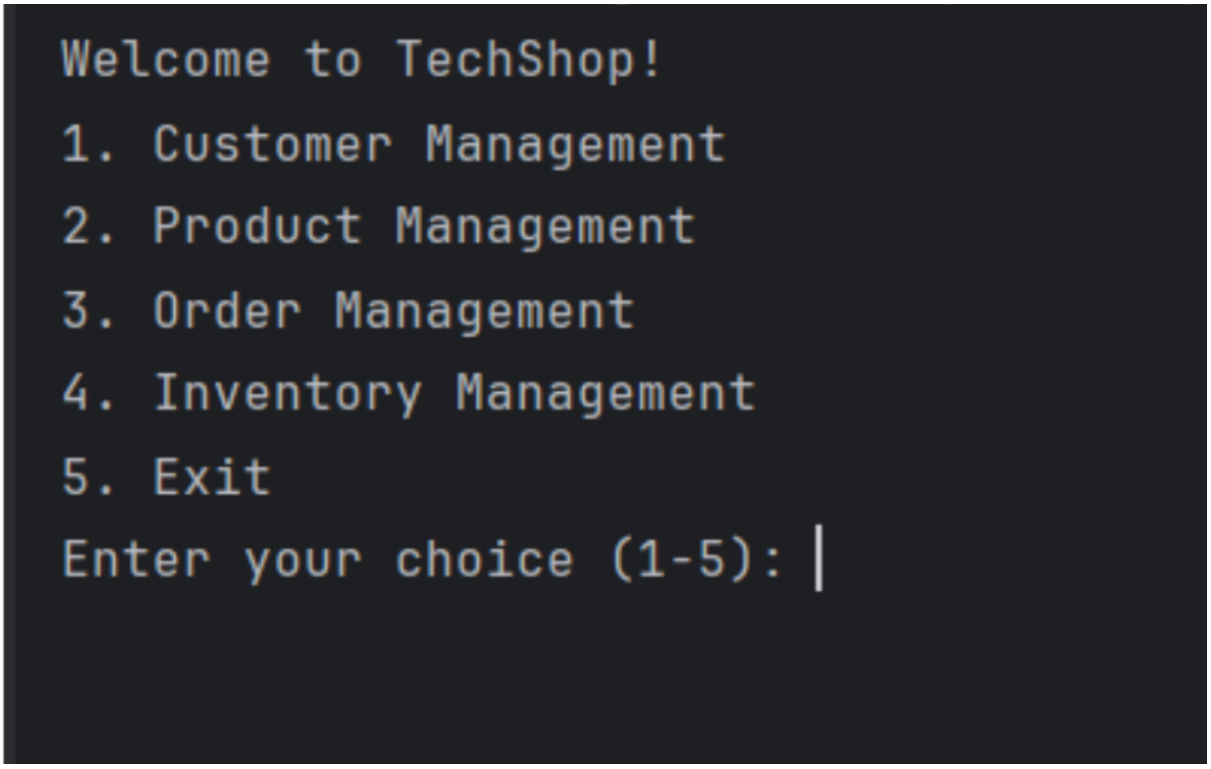

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
if customer_choice == "1":
CustomerManager.register_customer(
input("Enter First Name: "),
input("Enter Last Name: "),
input("Enter Email: "),
input("Enter DOB: "),
input("Enter Phone: "),
NumOrders: 0,
input("Enter Address: ")
)
elif customer_choice == "2":
temp_customer = CustomerManager.get_customer_by_id(customer_id=input('Enter Custome
print(temp_customer.get_customer_details())
elif customer_choice == "3":
temp_customer = CustomerManager.get_customer_by_id(customer_id=input('Enter Custome
temp_customer.update_customer_info(email=input('Enter new email'),
phone=input('Enter new phone'),
address=input('Enter new address'))
elif customer_choice == "4":
break
else:
print("Invalid choice. Please enter a number between 1 and 4.")
elif choice == "2":
while True:
product_management_menu()
product_choice = input("Enter your choice (1-4): ")
if product_choice == "1":
product_manager.add_product(
Products(input("Enter Product ID: "),
input("Enter Product Name: "),
TechShop
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8

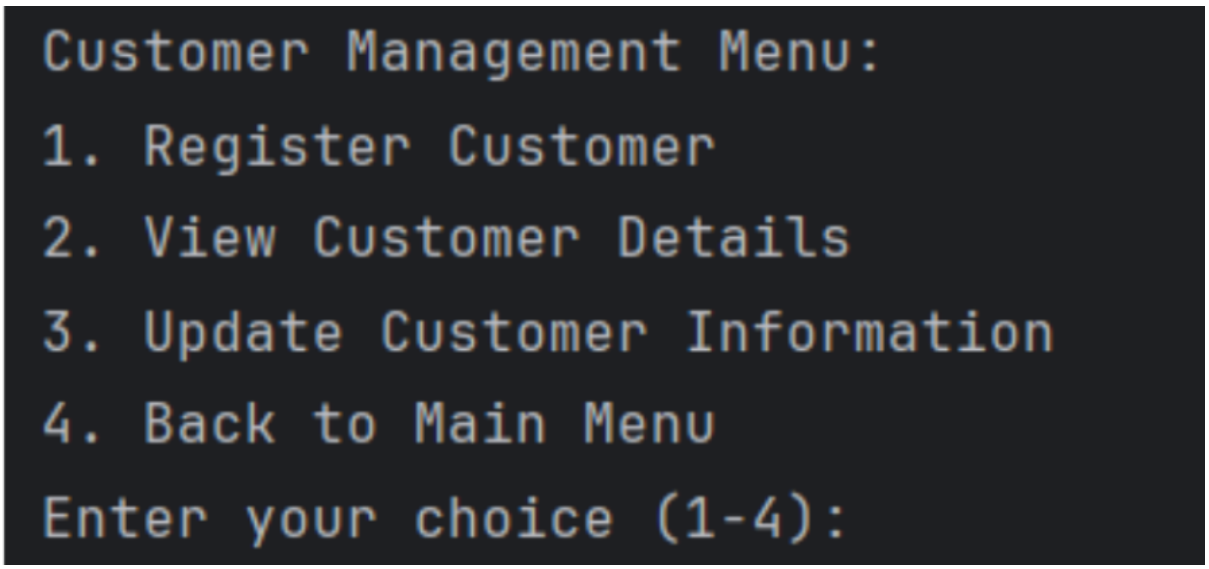

TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
input("Enter Description: "),
float(input("Enter Price: ")))
)
elif product_choice == "2":
temp_product = ProductManager.get_product_by_id(input('Enter Product ID'))
print(temp_product.get_product_details())
elif product_choice == "3":
temp_product = product_manager.get_product_by_id(input('Enter Product ID'))
temp_product.update_product_info(price=input('Enter New Product Price'), descriptio
elif product_choice == "4":
break
else:
print("Invalid choice. Please enter a number between 1 and 4.")
elif choice == "3":
while True:
order_manager()
order_choice = input("Enter your choice (1-5): ")
cm = CustomerManager()
if order_choice == "1":
order_manager.add_order(Orders(
input('Enter Order ID: '),
cm.get_customer_by_id(input('Enter Customer ID: ')),
input('Enter Order Date: '),
input('Enter Total Amount: '),
input('Enter Order Status: ')
))
elif order_choice == "2":
temp_Data = order_manager.get_order_by_id(input(
'Enter Order ID: '
))
print(temp_Data.get_order_details())
elif order_choice == "3":
TechShop
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
temp_Data = order_manager.get_order_by_id(input(
'Enter Order ID: '
))
temp_Data.update_order_status(input('Enter New Status'))
elif order_choice == "4":
temp_Data = order_manager.get_order_by_id(input(
'Enter Order ID: '
))
temp_Data.cancel_order()
elif order_choice == "5":
break
else:
print("Invalid choice. Please enter a number between 1 and 5.")
elif choice == "4":
while True:
inventory_management_menu()
inventory_choice = input("Enter your choice (1-7): ")
if inventory_choice == "1":
inventory_manager.add_to_inventory(input('Enter Product ID: '), input('Enter Produc
elif inventory_choice == "2":
inventory_manager.remove_from_inventory(input('Enter Product ID: '), input('Enter
elif inventory_choice == "3":
inventory_manager.list_all_products()
elif inventory_choice == "4":
inventory_manager.list_low_stock_products(2)
elif inventory_choice == "5":
inventory_manager.list_out_of_stock_products()
elif inventory_choice == "6":
inventory_manager.list_all_products()
elif inventory_choice == "7":
break
else:
TechShop
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.8


TechShop
Version control
main
Project
TechShop ~/Downloads/TechShop
custom_exception
__init__.py
CustomException.py
DBConnector
models
__init__.py
Customers.py
Inventory.py
OrderDetails.py
Orders.py
Products.py
services
__init__.py
CustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
External Libraries
Scratches and Consoles
ustomerManager.py
InventoryManager.py
OrderManager.py
ProductManager.py
main.py
elif choice == "4":
while True:
inventory_management_menu()
inventory_choice = input("Enter your choice (1-7): ")
if inventory_choice == "1":
inventory_manager.add_to_inventory(input('Enter Product ID: '), input('Enter Produc
elif inventory_choice == "2":
inventory_manager.remove_from_inventory(input('Enter Product ID: '), input('Enter P
elif inventory_choice == "3":
inventory_manager.list_all_products()
elif inventory_choice == "4":
inventory_manager.list_low_stock_products(2)
elif inventory_choice == "5":
inventory_manager.list_out_of_stock_products()
elif inventory_choice == "6":
inventory_manager.list_all_products()
elif inventory_choice == "7":
break
else:
print("Invalid choice. Please enter a number between 1 and 7.")
elif choice == "5":
print("Exiting TechShop. Goodbye!")
break
else:
print("Invalid choice. Please enter a number between 1 and 5.")
TechShop
main.py
1:1
CRLF
UTF-8

Outputs:


Welcome to TechShop!
1. Customer Management
2. Product Management
3. Order Management
4. Inventory Management
5. Exit
Enter your choice (1-5):


Customer Management Menu:
1. Register Customer
2. View Customer Details
3. Update Customer Information
4. Back to Main Menu
Enter your choice (1-4):

```
Enter your choice (1-4): 2
Enter CustomerID: 1
Customer ID: 1
Name: John Doe
DOB: 1990-01-15
Email: doe.john@email.com
Phone: 1234567890
Address: Avenue Street 1, Los Angeles, USA
```

```
Welcome to TechShop!
1. Customer Management
2. Product Management
3. Order Management
4. Inventory Management
5. Exit
Enter your choice (1-5): 5
Exiting TechShop. Goodbye!
```

```
Product Management Menu:
1. Add Product
2. View Product Details
3. Update Product Information
4. Back to Main Menu
Enter your choice (1-4):
```

```
Enter your choice (1-5): 3

Order Management Menu:
1. Place Order
2. View Order Details
3. Update Order Status
4. Cancel Order
5. Back to Main Menu
Enter your choice (1-5):
```