

Name: Manas Rustagi

Python Assignment 1 TechShop

Customer Class :

The screenshot shows the VS Code interface with the 'Customer' class definition in 'Customer.py'. The code includes properties for CustomerID, FirstName, LastName, Email, and Phone, along with their respective getters and setters. A 'main()' function at the bottom demonstrates the creation of a Customer object.

```
from Exception_Handling import InvalidDataException

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

    @property
    def CustomerID(self):
        return self._CustomerID

    @CustomerID.setter
    def CustomerID(self, new_customerID):
        self._CustomerID = new_customerID

    @property
    def FirstName(self):
        return self._FirstName

    @FirstName.setter
    def FirstName(self, new_firstname):
        self._FirstName = new_firstname

    @property
    def LastName(self):
        return self._LastName

    @LastName.setter
    def LastName(self, new_lastname):
        self._LastName = new_lastname

    @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")

    @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")

Customer.create()
```

The screenshot shows the VS Code interface with the 'Customer' class definition in 'Customer.py'. The code includes properties for CustomerID, FirstName, LastName, Email, and Phone, along with their respective getters and setters. A 'main()' function at the bottom demonstrates the creation of a Customer object.

```
from Exception_Handling import InvalidDataException

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

    @property
    def CustomerID(self):
        return self._CustomerID

    @CustomerID.setter
    def CustomerID(self, new_customerID):
        self._CustomerID = new_customerID

    @property
    def FirstName(self):
        return self._FirstName

    @FirstName.setter
    def FirstName(self, new_firstname):
        self._FirstName = new_firstname

    @property
    def LastName(self):
        return self._LastName

    @LastName.setter
    def LastName(self, new_lastname):
        self._LastName = new_lastname

    @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")

    @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")

Customer.create()
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project Tree:** Shows the project structure under "TechShop Assignment 1 E:\P".
- Code Editor:** Displays the content of `Customer.py`. The code includes methods for creating customers and calculating total orders.
- Status Bar:** Shows file statistics: 109 lines, 3 errors, and 1 warning.
- Bottom Status:** Shows file statistics: B2:17 CRLF UTF-8 4 spaces Python 3.11.

```
Customer.py
usage
@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 OrderID From Orders Where CustomerID=%s Group by CustomerID"
        values=(CustomerID,)
        cur.execute(query,values)
        records=cur.fetchall()
        if record:
            print(f"Total Order Done by the customer = {record[0]}")
        else:
            print("No Order Found")
    except Exception as e:
        print("Error",e)
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project Tree:** The project is named "TechShop Assignment 1 E(P)". It contains several files: __init__.py, Customer.py, db_connection.py, demo.py, Exception_Handling.py, Inventory.py, main.py, OrderDetails.py, Orders.py, Product.py, and db_connection.py.
- Code Editor:** The main window displays the content of `Customer.py`. The code handles customer orders and database interactions. It includes methods for getting customer details, updating customer info, and handling exceptions.
- Status Bar:** The status bar at the bottom right shows the file path as "Customer > create()", the line number as "109", and the current file as "Customer.py".

```
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()

# 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("CustomerID : {customer_detail[0]}")
            print("First Name : {customer_detail[1]}")
            print("Last Name : {customer_detail[2]}")
            print("Email : {customer_detail[3]}")
            print("Phone Number : {customer_detail[4]}")
            print("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()
```

```

Customer.py

values=(CustomerID,)
cur.execute(query,values)
customer_detail=cur.fetchone()
if customer_detail:
    print("CustomerID : {customer_detail[0]}")
    print("First Name : {customer_detail[1]}")
    print("Last Name : {customer_detail[2]}")
    print("Email : {customer_detail[3]}")
    print("Phone Number : {customer_detail[4]}")
    print("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 Successfully Submitted")
except Exception as e:
    print("Error Updating Request")
finally:
    self.db_connector.close_connection()

```

Customer created Successfully(CustomerID=12)

```

MySQL 8.0 Command Line Client

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 'vh' 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 | 123 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 | Olive | Anderson | olive.anderson@example.com | 3334445555 | 707 Walnut St |
| 11 | Manas | Rustagi | manasrustagi@gmail.com | 798261438 | Karol Bagh |
| 12 | Manas | Rustagi | manasrustag123@gmail.com | 798261438 | 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 | 123 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 | olive.anderson@example.com | 3334445555 | 707 Walnut St |
| 11 | Manas | Rustagi | manasrustagi@gmail.com | 798261438 | Karol Bagh |
| 12 | Manas | Rustagi | manasrustag123@gmail.com | 798261438 | Karol Bagh |
12 rows in set (0.00 sec)

mysql>

```

Email Id already exists (Cannot create New user)

MySQL 8.0 Command Line Client

```

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 |
+-----+-----+-----+-----+
| 1 | John | Doe | email@gmail.com |
| 2 | Jane | Smith | jane.smith@example.com |
| 3 | Alice | Johnson | alice.johnson@example.com |
| 4 | Bob | Williams | bob.williams@example.com |
| 5 | Eva | Brown | eva.brown@example.com |
| 6 | Charlie | Davis | charlie.davis@example.com |
| 7 | Grace | Miller | grace.miller@example.com |
| 8 | David | Moore | david.moore@example.com |
| 9 | Sophia | Lee | sophia.lee@example.com |
| 10 | Oliver | Anderson | oliver.anderson@example.com |
| 11 | Manas | Rustagi | manasrustagi12@gmail.com |
| 12 | Manas | Rustagi | manasrustagi16@gmail.com |
+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql> select * from Customers;
+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | email |
+-----+-----+-----+-----+
| 1 | John | Doe | email@gmail.com |
| 2 | Jane | Smith | jane.smith@example.com |
| 3 | Alice | Johnson | alice.johnson@example.com |
| 4 | Bob | Williams | bob.williams@example.com |
| 5 | Eva | Brown | eva.brown@example.com |
| 6 | Charlie | Davis | charlie.davis@example.com |
| 7 | Grace | Miller | grace.miller@example.com |
| 8 | David | Moore | david.moore@example.com |
| 9 | Sophia | Lee | sophia.lee@example.com |
| 10 | Oliver | Anderson | oliver.anderson@example.com |
| 11 | Manas | Rustagi | manasrustagi12@gmail.com |
| 12 | Manas | Rustagi | manasrustagi16@gmail.com |
+-----+-----+-----+-----+
12 rows in set (0.00 sec)

mysql>

```

Customer.py code:

```

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()

```

Total Orders Done by a Customer(CustomerID=1)

MySQL 8.0 Command Line Client

```

mysql> select * from Orders;
+-----+-----+-----+-----+
| OrderID | CustomerID | OrderDate | TotalAmount |
+-----+-----+-----+-----+
| 1 | 1 | 2024-01-15 | 3088 |
| 2 | 1 | 2024-01-16 | 6088 |
| 3 | 1 | 2024-01-17 | 6088 |
| 4 | 1 | 2024-01-19 | 988 |
| 5 | 1 | 2024-01-20 | 4958 |
| 6 | 1 | 2024-01-20 | 4958 |
| 7 | 6 | 2024-01-21 | 1320 |
| 8 | 7 | 2024-01-22 | 420 |
| 9 | 9 | 2024-01-23 | 700 |
| 10 | 8 | 2024-01-24 | 220 |
| 11 | 8 | 2024-01-25 | 440 |
| 12 | 9 | 2023-05-25 | 440 |
+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>

```

Customer.py code:

```

def CalculateTotalOrders(self,CustomerID):
    try:
        self.db_connector.open_connection()
        cursor=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("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()

```

Getting Info about a Customer(CustomerID=2)

MySQL 8.0 Command Line Client

```

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 'vh' for help. Type 'lc' 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 Birch St |
| 4 | Bob | Williams | bob.williams@example.com | 9998887777 | 123 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 | 7778889999 | 404 Birch St |
| 8 | David | Moore | david.moore@example.com | 2223334444 | 505 Reedwood 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 | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
| 12 | Ramas | Rustagi | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql>
```

Customer ID : 2
First Name : Jane
Last Name : Smith
Email : jane.smith@example.com
Phone Number : 9876543210
Address : 456 Oak St

Process finished with exit code 0

Updating Customer Info(CustomerId=2)

MySQL 8.0 Command Line Client

```

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 Birch St |
| 4 | Bob | Williams | bob.williams@example.com | 9998887777 | 123 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 | 7778889999 | 404 Birch St |
| 8 | David | Moore | david.moore@example.com | 2223334444 | 505 Reedwood 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 | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
| 12 | Ramas | Rustagi | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
+-----+-----+-----+-----+-----+
12 rows in set (0.00 sec)

mysql> select * from Customers;
+-----+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | email | Phone | Address |
+-----+-----+-----+-----+-----+
| 1 | John | Doe | abc@outlook.com | 1234567890 | New York |
| 2 | Jane | Smith | jane.smith@example.com | 9876543210 | 456 Oak St |
| 3 | Alice | Johnson | alice.johnson@example.com | 5551234567 | 789 Birch St |
| 4 | Bob | Williams | bob.williams@example.com | 9998887777 | 123 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 | 7778889999 | 404 Birch St |
| 8 | David | Moore | david.moore@example.com | 2223334444 | 505 Reedwood 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 | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
| 12 | Ramas | Rustagi | manansrustagi12@gmail.com | 7980881438 | Karol Bagn |
+-----+-----+-----+-----+-----+
12 rows in set (0.00 sec)

mysql>
```

Customer ID : 2
First Name : Jane
Last Name : Smith
Email : abc@outlook.com
Phone Number : 1234567890
Address : New York

Update request Successfully Submitted

Process finished with exit code 0

Products Class:

```
3     class product:
4         def __init__(self,productid : int, productname : str, description : str, price : float,db_connector):
5             self._ProductID = productid
6             self._ProductName = productname
7             self._Description = description
8             self._Price = price
9             self._db_connector = db_connector
10
11        1 usage
12        @property
13        def ProductID(self):
14            return self._ProductID
15        @ProductID.setter
16        def ProductID(self,new_productID):
17            self._ProductID=new_productID
18
19        1 usage
20        @property
21        def ProductName(self):
22            return self._ProductName
23        @ProductName.setter
24        def ProductName(self,new_productname):
25            self._ProductName=new_productname
26
27        1 usage
28        @property
29        def Description(self):
30            return self._Description
31        @Description.setter
32        def Description(self,new_description):
33            self._Description=new_description
34
35        1 usage
36        @property
37        def Price(self):
38            return self._Price
39        @Price.setter
40        def Price(self,new_price):
41            if new_price >= 0:
42                self._Price=new_price
43            else:
44
45    product - UpdateProductInfo() - except Exception as e
```

```
39           self._Price=new_price
40       else:
41           pass
42
43   def GetProductDetails(self,ProductID):
44       try:
45           self._db_connector.open_connection()
46           cur=self._db_connector.connection.cursor()
47           query='Select * From Products Where ProductID=%s'
48           values=(ProductID,)
49           cur.execute(query,values)
50           record=cur.fetchone()
51           if record:
52
53               print(f'Product ID : {record[0]}')
54               print(f'Product Name : {record[1]}')
55               print(f'Description : {record[2]}')
56               print(f'Price : {record[3]}')
57           else:
58               print('Product not found')
59       except Exception as e:
60           print('Incorrect Id')
61   finally:
62       self._db_connector.close_connection()
63
64
65   def UpdateProductInfo(self,Price,Description,ProductID):
66       try:
67           self._db_connector.open_connection()
68           cur=self._db_connector.connection.cursor()
69           query='Update Products Set Description=%s,Price=%s Where ProductID=%s'
70           values=(Description,Price,ProductID)
71           cur.execute(query,values)
72           self._db_connector.connection.commit()
73           print("Updated Successfully")
74       except Exception as e:
75           print('Error')
76   finally:
77       self._db_connector.close_connection()
78
79 product - UpdateProductInfo() - except Exception as e
```

```

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(f"Error: {e}")
finally:
    self.db_connector.close_connection()

def is_product_Instock(self, ProductID):
    try:
        self.db_connector.open_connection()
        query = "Select QuantityInStock From Inventory Where ProductID=%s"
        values = (ProductID,)
        cur = self.db_connector.connection.cursor()
        cur.execute(query, values)
        record = cur.fetchone()
        if record:
            print(f"Quantity in Stock = {record[0]}")
        else:
            print("Not Quantity Available")
    except Exception as e:
        print(f"Error occurred : {e}")
    finally:
        self.db_connector.close_connection()

```

Getting Product Info(ProductID=5)

MySQL 8.0 Command Line Client

```

mysql> select * from Products;
+-----+-----+-----+-----+
| ProductID | ProductName | Description | Price |
+-----+-----+-----+-----+
| 1 | Laptop | PC | 1320 |
| 2 | Smartphone | Phones & Tablet | 888 |
| 3 | Headphones | Accessories | 165 |
| 4 | Tablet | Phones & Tablet | 88 |
| 5 | Smartwatch | Accessories | 220 |
| 6 | Desktop PC | PC | 1650 |
| 7 | Bluetooth Speaker | Accessories | 55 |
| 8 | Mouse | Electronics | 660 |
| 9 | External Hard Drive | Accessories | 88 |
| 10 | Gaming Console | Console | 448 |
| 11 | PS5 | Console | 520 |
+-----+-----+-----+-----+
12 rows in set (0.00 sec)

mysql>

```

TechShop Assignment 1

```

Project > TechShop Assignment 1 E(P) > Product.py > main.py > Orders.py > OrderDetails.py > Inventory.py > db_connection.py > Exception_Handling.py > _init_.py > Customer.py > db.connector.py > demo.py > Exception_Handling.py > Inventory.py > main.py > OrderDetails.py > Orders.py > Product.py > Usage
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("Product not found")
    except:
        product = GetProductDetails() > try > if record

```

Updating Product Info(ProductId=2)

The screenshot shows two windows side-by-side. On the left is the MySQL Command Line Client showing the output of a query to update a product's price:

```

MySQL 8.0 Command Line Client
12 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 (0.04 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>

```

On the right is the PyCharm IDE showing the Python code for updating a product's information. The code uses a database connector to execute an UPDATE query:

```

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 P"
        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()

def Is_product_InStock(self, ProductId):
    try:
        product = UpdateProductInfo()
    except Exception as e:

```

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

The screenshot shows two windows side-by-side. On the left is the MySQL Command Line Client showing the output of a query to check product availability in the inventory:

```

MySQL 8.0 Command Line Client
11 rows in set (0.04 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> select*from Inventory;
ERROR 1146 (42S02): Table 'techshop.inventory' doesn't exist
mysql> select*from Inventory;
+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 5 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>

```

On the right is the PyCharm IDE showing the Python code for checking product availability. The code prints the quantity in stock for a specific product ID:

```

C:\Users\A2Z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment\main.py"
Quantity in Stock = 10

Process finished with exit code 0

```

Order Class:

```
from datetime import datetime
from typing import List

class Order:
    def __init__(self, order_id: int, customer_id: int, orderdate: datetime, total_amount: float, db_connector):
        self._order_id = order_id
        self._customer_id = customer_id
        self._orderdate = orderdate
        self._total_amount = total_amount
        self._db_connector = db_connector

    @property
    def OrderID(self):
        return self._order_id
    @OrderID.setter
    def OrderID(self, new_order_id):
        self._order_id = new_order_id

    @property
    def CustomerID(self):
        return self._customer_id
    @CustomerID.setter
    def CustomerID(self, new_customer_id):
        self._customer_id = new_customer_id

    @property
    def OrderDate(self):
        return self._orderdate
    @OrderDate.setter
    def OrderDate(self, new_orderdate):
        self._orderdate = new_orderdate

    @property
    def TotalAmount(self):
        return self._total_amount
    @TotalAmount.setter
    def TotalAmount(self, new_total):
        self._total_amount = new_total

    def CancelOrder(self):
        try:
            self._db_connector.open_connection()
            cur = self._db_connector.connection.cursor()
            query = "Delete From Orders Where OrderId = %s"
            values = (self._order_id,)
            cur.execute(query, values)
            record = cur.fetchone()
            if record:
                print(f"Order ID {self._order_id} has been deleted")
            else:
                print("No Order Found")
        except Exception as e:
            print(f"An error occurred: {e}")
        finally:
            self._db_connector.close_connection()

    def GetOrderDetails(self, order_id):
        try:
            self._db_connector.open_connection()
            cur = self._db_connector.connection.cursor()
            query = "Select * from Orders Where OrderId = %s"
            values = (order_id,)
            cur.execute(query, values)
            record = cur.fetchone()
            if record:
                print(f"Order ID : {record[0]}")
                print(f"Customer ID : {record[1]}")
                print(f"Order Date : {record[2]}")
                print(f"Total Amount : {record[3]}")
            else:
                print("Order ID not Present")
        except Exception as e:
            print(f"An error occurred: {e}")

if __name__ == "__main__":
    orders = Order(1, 1, datetime.now(), 100.0, db_connector)
    print(orders)
    orders.CancelOrder()
    print(orders)
```

```
return self._total_amount
    @TotalAmount.setter
    def TotalAmount(self, new_total):
        self._total_amount = 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 Found")
        except Exception as e:
            print(f"An error occurred: {e}")
        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,)
            cur.execute(query, values)
            record = cur.fetchone()
            if record:
                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")
            else:
                print("Order ID not Present")
        except Exception as e:
            print(f"An error occurred: {e}")

if __name__ == "__main__":
    orders = Order(1, 1, datetime.now(), 100.0, db_connector)
    print(orders)
    orders.CancelOrder()
    print(orders)
```

```

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

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

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("Deleting 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 = %s"
            )
            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(f"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 | 3088 |
| 4 | 5 | 2024-01-18 | 668 |
| 5 | 4 | 2024-01-19 | 868 |
| 1 | 3 | 2024-01-20 | 4958 |
| 7 | 6 | 2024-01-21 | 1320 |
| 8 | 7 | 2024-01-22 | 448 |
| 9 | 20 | 2024-01-23 | 768 |
| 18 | 8 | 2024-01-24 | 228 |
| 15 | 9 | 2023-05-25 | 448 |
+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>

```

Project: TechShop Assignment 1

Run: main

C:\Users\az2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"

Total Amount of OrderID 8 is : 448

Process finished with exit code 0

Getting info about a Particular Order

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client showing the results of a query:

```
mysql> select * from Orders;
+-----+-----+-----+-----+
| OrderID | CustomerID | OrderDate | TotalAmount |
+-----+-----+-----+-----+
| 1       | 1          | 2024-01-15 | 3000        |
| 4       | 5          | 2024-01-18 | 600         |
| 5       | 4          | 2024-01-19 | 800         |
| 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         |
+-----+-----+-----+-----+
0 rows in set (0.00 sec)

mysql>
```

On the right is a Python code editor for a project named "TechShop Assignment 1". The main.py file contains:

```
Customer.py    Product.py    Orders.py    main.py    Orders
Project Run main
...
C:\Users\A2Z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Enter the Order Id : 9
Getting Details .....
OrderID : 9
CustomerID : 9
Order Date : 2024-01-23
Total Amount : 700
Process finished with exit code 0
```

21:25 CRLF UTF-8 4 spaces Python 3.11

Cancelling Order

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

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client showing the results of two queries:

```
mysql> select * from Orders;
+-----+-----+-----+-----+
| OrderID | CustomerID | OrderDate | TotalAmount |
+-----+-----+-----+-----+
| 1       | 1          | 2024-01-15 | 3000        |
| 4       | 5          | 2024-01-18 | 600         |
| 5       | 4          | 2024-01-19 | 800         |
| 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         |
+-----+-----+-----+-----+
0 rows in set (0.00 sec)

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

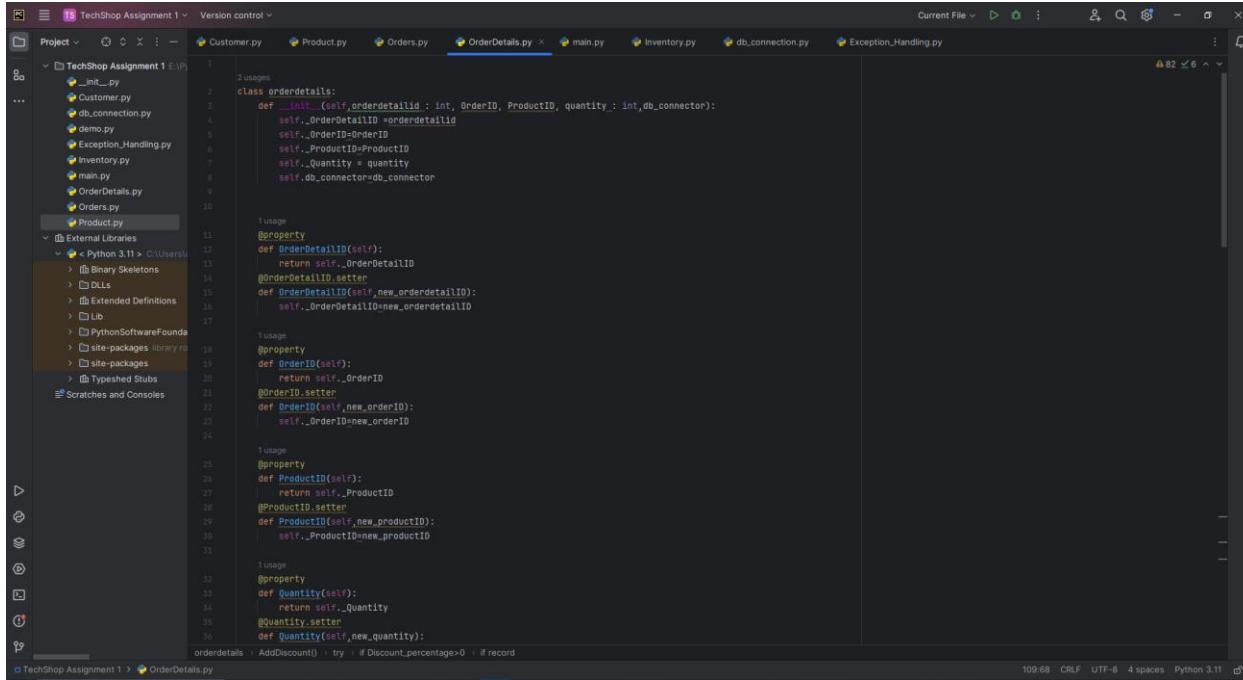
mysql>
```

On the right is a Python code editor for a project named "TechShop Assignment 1". The main.py file contains:

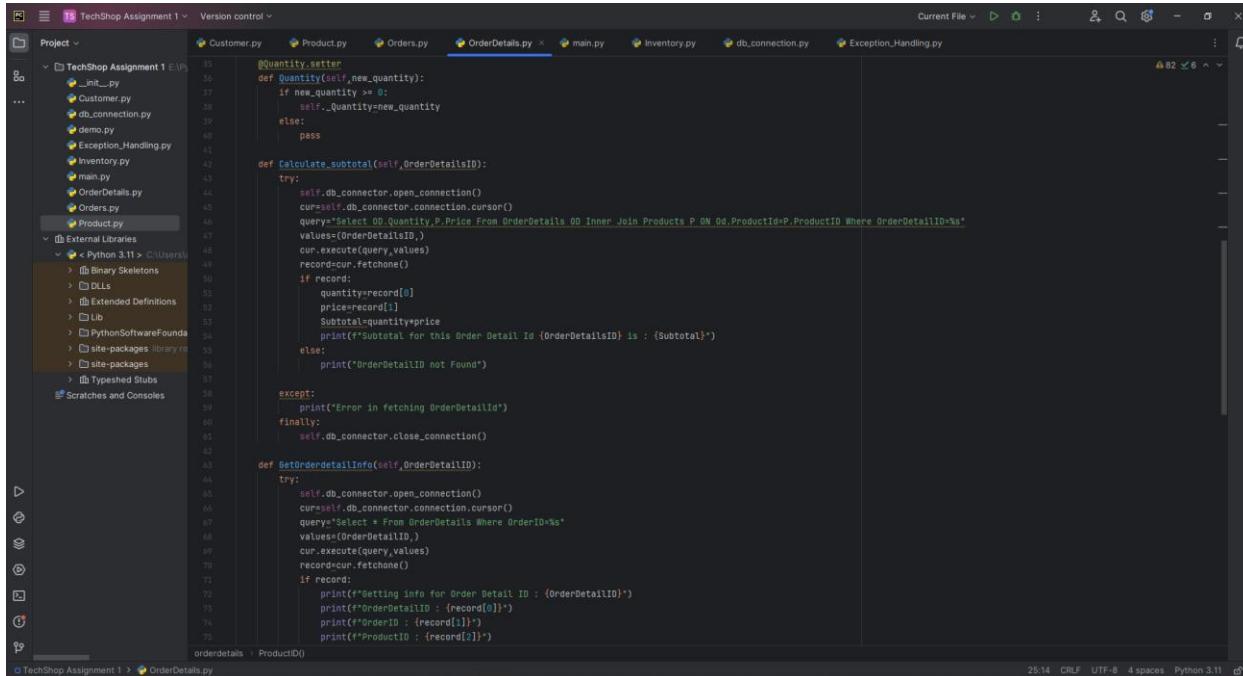
```
Customer.py    Product.py    Orders.py    main.py    Orders
Project Run main
...
C:\Users\A2Z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Deleting Order 15
Wait work in Progress.....
Error in Deletion
Process finished with exit code 0
```

22:22 CRLF UTF-8 4 spaces Python 3.11

OrderDetails Class:



```
Project ▾ TechShop Assignment 1 E(P) Customer.py Product.py Orders.py OrderDetails.py main.py Inventory.py db_connection.py Exception_Handling.py
  1
  2 class OrderDetails:
  3     def __init__(self, orderDetailID: int, OrderID, ProductID, quantity : int, db_connector):
  4         self._orderDetailID = orderDetailID
  5         self._OrderID = OrderID
  6         self._ProductID = ProductID
  7         self._Quantity = quantity
  8         self._db_connector = db_connector
  9
 10
 11     @property
 12     def OrderDetailID(self):
 13         return self._OrderDetailID
 14     @OrderDetailID.setter
 15     def OrderDetailID(self, new_orderDetailID):
 16         self._OrderDetailID = new_orderDetailID
 17
 18
 19     @property
 20     def OrderID(self):
 21         return self._OrderID
 22     @OrderID.setter
 23     def OrderID(self, new_orderID):
 24         self._OrderID = new_orderID
 25
 26
 27     @property
 28     def ProductID(self):
 29         return self._ProductID
 30     @ProductID.setter
 31     def ProductID(self, new_productID):
 32         self._ProductID = new_productID
 33
 34
 35     @property
 36     def Quantity(self):
 37         return self._Quantity
 38     @Quantity.setter
 39     def Quantity(self, new_quantity):
 40         self._Quantity = new_quantity
 41
 42
 43     orderdetails.AddDiscount() -> try : # Discount_percentage>0 -> if record
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
```



```
Project ▾ TechShop Assignment 1 E(P) Customer.py Product.py Orders.py OrderDetails.py main.py Inventory.py db_connection.py Exception_Handling.py
  35
  36     @Quantity.setter
  37     def Quantity(self, new_quantity):
  38         if new_quantity >= 0:
  39             self._Quantity = new_quantity
  40         else:
  41             pass
  42
  43     def Calculate_subtotal(self, OrderDetailsID):
  44         try:
  45             self._db_connector.open_connection()
  46             cur = self._db_connector.connection.cursor()
  47             query = "Select * From OrderDetails OD Inner Join Products P ON OD.ProductID=P.ProductID Where OrderDetailID=%s"
  48             values = (OrderDetailsID,)
  49             cur.execute(query, values)
  50             record = cur.fetchone()
  51             if record:
  52                 quantity = record[0]
  53                 price = record[1]
  54                 Subtotal = quantity * price
  55                 print(f"Subtotal for this Order Detail ID {OrderDetailsID} is : {Subtotal}")
  56             else:
  57                 print("OrderDetailID not Found")
  58
  59         except:
  60             print("Error in fetching OrderDetailID")
  61         finally:
  62             self._db_connector.close_connection()
  63
  64     def GetOrderdetailInfo(self, OrderDetailID):
  65         try:
  66             self._db_connector.open_connection()
  67             cur = self._db_connector.connection.cursor()
  68             query = "Select * From OrderDetails Where OrderID=%s"
  69             values = (OrderDetailID,)
  70             cur.execute(query, values)
  71             record = cur.fetchone()
  72             if record:
  73                 print(f"Getting info for Order Detail ID : {OrderDetailID}")
  74                 print(f"OrderDetailID : {record[0]}")
  75                 print(f"OrderID : {record[1]}")
  76                 print(f"ProductID : {record[2]$")
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** TechShop Assignment 1
- File:** OrderDetails.py
- Code Content:** The code handles order details, including printing record details, updating quantities, adding discounts, and validating inputs. It uses a database connector and exception handling.

```
print("Quantity : {record[1]}")
else:
    print("No Order Found")
except Exception as e:
    print("Error Please Enter the OrderDetailID 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("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 Update")
    finally:
        self.db_connector.close_connection()

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("Getting Discount on Order ID : {OrderID}")
                Amount = record[0]
                Payment = Amount - (Amount * Discount_percentage / 100)
                print("Total Amount after Discount = {Payment}")
            else:
                print("Wrong OrderID Entered")
        else:
            print("Discount Percentage Cannot be 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 |
| 3 | 1 | 4 | 4 |
| 4 | 2 | 1 | 2 |
| 5 | 2 | 4 | 2 |
| 6 | 2 | 5 | 1 |
| 7 | 3 | 6 | 3 |
| 8 | 3 | 7 | 8 |
| 9 | 3 | 8 | 2 |
| 10 | 4 | 10 | 1 |
| 11 | 4 | 10 | 4 |
| 12 | 5 | 7 | 7 |
| 13 | 5 | 5 | 5 |
| 14 | 6 | 5 | 4 |
| 15 | 6 | 15 | 2 |
+-----+-----+-----+
0 rows in set (0.00 sec)

MySQL> select*from Products;
+-----+-----+-----+
| ProductID | ProductName | Description | Price |
+-----+-----+-----+
| 1 | Laptop | PC | 1200 |
| 2 | Smartphone | Phone | 900 |
| 3 | Headphones | Accessories | 165 |
| 4 | Tablet | Phones & Tablet | 350 |
| 5 | Smartwatch | Accessories | 220 |
| 6 | Desktop PC | PC | 1650 |
| 7 | Bluetooth Speaker | Accessories | 55 |
| 8 | Mouse | Electronics | 60 |
| 9 | External Hard Drive | Accessories | 88 |
| 10 | Gaming Console | Console | 440 |
| 11 | PS5 | Console | 520 |
+-----+-----+-----+
11 rows in set (0.00 sec)

MySQL>
```

Getting all the info for a particular Order Detail

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client window, displaying the results of a query:

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

mysql>
```

On the right is the PyCharm IDE window, showing a Python project named "TechShop Assignment 1". The "main.py" file is open, containing the following code:

```
Project: TechShop Assignment 1
Run: main
C:\Users\{User}\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
OrderDetailID : 6
OrderID : 5
ProductID : 2
Quantity : 1
Process finished with exit code 0

25.35 CRLF UTF-8 4 spaces Python 3.11
```

```
from mysql.connector import connect, Error
try:
    connection = connect(
        host='localhost',
        database='techshop',
        user='root',
        password='password'
    )
    if connection.is_connected():
        db_info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_info)
        cursor = connection.cursor()
        cursor.execute("select * from OrderDetails")
        record = cursor.fetchone()
        print(record)
except Error as e:
    print(e)
```

Updating the Quantity in the Order Detail

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client window, displaying the results of a query:

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

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

mysql>
```

On the right is the PyCharm IDE window, showing a Python project named "TechShop Assignment 1". The "main.py" file is open, containing the following code:

```
Project: TechShop Assignment 1
Run: main
C:\Users\{User}\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Updating Quantity to 3 of Order Detail ID 5
Quantity Updated Successfully
Process finished with exit code 0

26.31 CRLF UTF-8 4 spaces Python 3.11
```

```
from mysql.connector import connect, Error
try:
    connection = connect(
        host='localhost',
        database='techshop',
        user='root',
        password='password'
    )
    if connection.is_connected():
        db_info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_info)
        cursor = connection.cursor()
        cursor.execute("select * from OrderDetails")
        record = cursor.fetchone()
        print(record)
        cursor.execute("update OrderDetails set Quantity = 3 where OrderDetailID = 5")
        connection.commit()
except Error as e:
    print(e)
```

Adding Discount in Total Bill

The screenshot shows a dual-pane interface. On the left is a terminal window titled "MySQL 8.0 Command Line Client" displaying a query result from the "Orders" table:

```
0 rows in set (0.00 sec)
mysql> select * from Orders;
+ OrderID | CustomerID | OrderDate | TotalAmount |
| 1       | 1           | 2024-01-15 | 3880
| 4       | 5           | 2024-01-18 | 660
| 5       | 4           | 2024-01-19 | 880
| 6       | 1           | 2024-01-20 | 650
| 6       | 6           | 2024-01-21 | 1120
| 8       | 7           | 2024-01-22 | 440
| 9       | 9           | 2024-01-23 | 700
| 9       | 3           | 2024-01-24 | 240
| 10      | 9           | 2023-05-25 | 440
+-----+
0 rows in set (0.00 sec)

mysql>
```

On the right is a Python IDE window titled "TechShop Assignment 1". It shows a project structure with files: Product.py, Orders.py, OrderDetails.py, and main.py. The main.py file contains the following code:

```
C:\Users\az1\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Getting Discount on Order ID : 4
Total Amount after Discount = 501.0
Process finished with exit code 0
```

The status bar at the bottom indicates the file is "TechShop Assignment 1 > main.py", the time is "27:30", and the encoding is "CRLF - UTF-8 4 spaces Python 3.11".

Inventory Class:

```
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):
        return self._lastStockUpdate

Inventory = ListAllProducts() try : if record : for i in record
```

```
@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()
        cursor=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("Enter the Inventory ID for Which you want to know Product Details : [InventoryId]")
            print("ProductID : {record[0]}")
            print("Product Name : {record[1]}")
            print("Description : {record[2]}")
            print("Price : {record[3]}")
        else:
            print("Product ID not exists")
    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()
        cursor=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("Enter the ProductID : {ProductId}")
            print("Quantity In Stock is : {record[0]}")
        else:
            print("No Product Found in inventory")
    except Exception as e:
        print("Error In Finding Product")
```

```
Project ▾
  TechShop Assignment 1 E:\Python\Proj
    Customer.py
    Product.py
    Orders.py
    OrderDetails.py
    Inventory.py x
    main.py
    db.connector.py
    Exception_Handling.py

  External Libraries
    Python 3.11 > C:\Users\aziz\AppD...
      Binary Skeletons
      DLLs
      Extended Definitions
      Lib
      PythonSoftwareFoundation.Pyth...
      site-packages library root
      site-packages
      TypedStubs
  Scratches and Consoles

Customer.py
Product.py
Orders.py
OrderDetails.py
Inventory.py x
main.py
db.connector.py
Exception_Handling.py

Current File ▾
  3 135 11 ▾

76     except Exception as e:
77         print("Error in fetching Stock")
78     finally:
79         self.db_connector.close_connection()
80
81 def AddtoQuantity(self,Add_Quantity,ProductID):
82     try:
83         self.db_connector.open_connection()
84         cur=self.db_connector.connection.cursor()
85         query="Update Inventory Set QuantityInStock = QuantityInStock+{0} Where ProductID={1}"
86         values=(Add_Quantity,ProductID)
87         cur.execute(query,values)
88         self.db_connector.connection.commit()
89         print("For ProductID ({0}) Quantity you want to add in stocks is : {1}").format(Add_Quantity)
90         print("Quantity Added Successfully")
91     except Exception as e:
92         print("Error In {0}").format(e)
93     finally:
94         self.db_connector.close_connection()
95
96 def RemoveFromInventory(self,Quantity,ProductID):
97     try:
98         self.db_connector.open_connection()
99         cur=self.db_connector.connection.cursor()
100        query="Update Inventory set QuantityInStock=QuantityInStock-{0} Where ProductID={1}"
101        values=(Quantity,ProductID)
102        cur.execute(query,values)
103        self.db_connector.connection.commit()
104        print("For ProductID ({0}) Quantity you want to remove in stocks is : {1}").format(Quantity)
105        print("Quantity In Stock Updated Successfully")
106    except:
107        print("Error In {0}").format(e)
108    finally:
109        self.db_connector.close_connection()
110
111 def UpdateStockQuantity(self,new_quantity,ProductID):
112     try:
113         self.db_connector.open_connection()
114         cur=self.db_connector.connection.cursor()
115         query="Update Inventory set QuantityInStock={0} Where ProductID={1}"
116         values=(new_quantity,ProductID,)

Inventory > ListAllProducts() - try - if record - for i in record
```

```
values=(new_quantity,ProductID)
        cur.execute(query,values)
        self.db_connector.connection.commit()
        print("For ProductID : "+str(ProductID) + " Updated Quantity will be "+str(new_quantity))
        print("Quantity updated successfully")
    except Exception as e:
        print("Error in update")
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("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*I.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("Total Inventory Value : "+str(record[0]))
    except Exception as e:
        print("Error in Fetching")
    finally:
        self.db_connector.close_connection()
```

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** TechShop Assignment 1 - Version control
- Project Tree (Left):** Shows the project structure under "TechShop Assignment 1 E:\Python\Proj".
 - Customer.py
 - Product.py
 - Orders.py
 - OrderDetails.py
 - Inventory.py
 - main.py
 - db_connection.py
 - Exception_Handling.py
 - __init__.py
 - Customer.py
 - db_connection.py
 - demo.py
 - Exception_Handling.py
 - Inventory.py
 - main.py
 - OrderDetails.py
 - Orders.py
 - Product.py
- Code Editor (Right):** The file "Inventory.py" is open, showing Python code for managing inventory. The code includes functions for calculating total inventory value, listing products below a threshold, and listing products with zero stock. It uses a database connection and handles exceptions.
- Status Bar (Bottom):** Shows the file path as "inventory > ListAllProducts() > try > if record > for i in record", along with other status information like "210:74 CRLF UTF-8 4 spaces Python 3.11".

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** TechShop Assignment 1
- Current File:** Inventory.py
- Code Editor:** The code for the `list_all_products` method is displayed. The cursor is at the start of the `for i in record:` loop.
- Toolbars and Status Bar:** Standard PyCharm toolbars and status bar showing file count (3), character count (135), and line number (210).
- Code Structure:** The code uses a cursor navigation feature to highlight the current line of code being edited.

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

The screenshot shows a terminal window on the left and a PyCharm IDE on the right. The terminal window displays MySQL queries and their results:

```
mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 50 | 2024-01-19 |
| 6 | 6 | 5 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 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 | 50 |
| 4 | Tablet | Phones & Tablet | 350 |
| 5 | Smartwatch | Accessories | 220 |
| 6 | Desktop PC | PC | 1650 |
| 7 | Bluetooth Speaker | Accessories | 45 |
| 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>
```

The PyCharm IDE shows a project named "TechShop Assignment 1" with files like Orders.py, OrderDetails.py, Inventory.py, and main.py. A terminal tab in PyCharm shows the command:

```
C:\Users\{User}\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
```

The output of the script is displayed in the terminal:

```
Enter the Inventory ID for Which you want to know Product Details : 2
ProductId : 2
Product Name : Smartphone
Description : Phone
Price : 980
Process finished with exit code 0
```

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

The screenshot shows a terminal window on the left and a PyCharm IDE on the right. The terminal window displays MySQL queries and their results:

```
mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 50 | 2024-01-19 |
| 6 | 6 | 5 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The PyCharm IDE shows a project named "TechShop Assignment 1" with files like Orders.py, OrderDetails.py, Inventory.py, and main.py. A terminal tab in PyCharm shows the command:

```
C:\Users\{User}\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
```

The output of the script is displayed in the terminal:

```
Enter the ProductID : 5
Quantity In Stock is : 10
Process finished with exit code 0
```

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

The screenshot shows a terminal window on the left and a code editor window on the right. The terminal window displays MySQL commands and their results:

```
mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 5 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The code editor window shows a Python script named `main.py` with the following content:

```
C:\Users\aziz\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment"
For ProductId & Quantity you want to add in stocks is : 5
Quantity Added Successfully
Process finished with exit code 0
```

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

The screenshot shows a terminal window on the left and a code editor window on the right. The terminal window displays MySQL commands and their results:

```
mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 100 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> select * from Inventory
->;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 98 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 40 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The code editor window shows a Python script named `main.py` with the following content:

```
C:\Users\aziz\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment"
For ProductId 3 Quantity you want to remove in stocks is : 2
Quantity In Stock Updated Successfully
Process finished with exit code 0
```

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

The screenshot shows a terminal window on the left and a code editor on the right. The terminal window displays MySQL queries and their results. The code editor shows a Python script named `main.py` with some code and output from its execution.

```
MySQL 8.0 Command Line Client
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 98 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 55 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 98 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 55 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The code editor shows the following Python code and its execution output:

```
Project: TechShop Assignment 1
Run: main
C:\Users\az2\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
For ProductID 7 Updated Quantity will be 55
Quantity updated Successfully
Process finished with exit code 0
```

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

The screenshot shows a terminal window on the left and a code editor on the right. The terminal window displays MySQL queries and their results. The code editor shows a Python script named `main.py` with some code and output from its execution.

```
MySQL 8.0 Command Line Client
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 98 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 55 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The code editor shows the following Python code and its execution output:

```
Project: TechShop Assignment 1
Run: main
C:\Users\az2\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Enter the ProductID for which you want to know stocks:10
Product Available
Process finished with exit code 0
```

The screenshot shows a terminal window on the left and a code editor on the right. The terminal window displays MySQL queries and their results. The code editor shows a Python script named `main.py` with some code and output from its execution.

```
MySQL 8.0 Command Line Client
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1 | 1 | 50 | 2024-01-15 |
| 2 | 2 | 30 | 2024-01-16 |
| 3 | 3 | 98 | 2024-01-17 |
| 4 | 4 | 20 | 2024-01-18 |
| 5 | 5 | 10 | 2024-01-19 |
| 6 | 6 | 10 | 2024-01-20 |
| 7 | 7 | 55 | 2024-01-21 |
| 8 | 8 | 15 | 2024-01-22 |
| 9 | 9 | 25 | 2024-01-23 |
| 10 | 10 | 8 | 2024-01-24 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

The code editor shows the following Python code and its execution output:

```
Project: TechShop Assignment 1
Run: main
C:\Users\az2\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Enter the ProductID for which you want to know stocks:12
Wrong Product ID
Process finished with exit code 0
```

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

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client showing the results of a query:

```
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | lastStockUpdate |
+-----+-----+-----+-----+
| 1           | 1          | 50            | 2024-01-15    |
| 2           | 2          | 30            | 2024-01-16    |
| 3           | 3          | 98            | 2024-01-17    |
| 4           | 4          | 20            | 2024-01-18    |
| 5           | 5          | 10            | 2024-01-19    |
| 6           | 6          | 10            | 2024-01-20    |
| 7           | 7          | 55            | 2024-01-21    |
| 8           | 8          | 15            | 2024-01-22    |
| 9           | 9          | 25            | 2024-01-23    |
| 10          | 10         | 8             | 2024-01-24    |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

On the right is a Python code editor for a project named "TechShop Assignment 1". The main.py file contains the following code:

```
C:\Users\A2Z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Total Inventory Value : 155515
Process finished with exit code 0
```

The status bar at the bottom indicates the file is "main.py" and the Python version is "Python 3.11".

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

The screenshot shows a dual-pane interface. On the left is the MySQL 8.0 Command Line Client showing the results of a query:

```
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductId | QuantityInStock | lastStockUpdate |
+-----+-----+-----+-----+
| 1           | 1          | 50            | 2024-01-15    |
| 2           | 2          | 30            | 2024-01-16    |
| 3           | 3          | 98            | 2024-01-17    |
| 4           | 4          | 20            | 2024-01-18    |
| 5           | 5          | 10            | 2024-01-19    |
| 6           | 6          | 10            | 2024-01-20    |
| 7           | 7          | 55            | 2024-01-21    |
| 8           | 8          | 15            | 2024-01-22    |
| 9           | 9          | 25            | 2024-01-23    |
| 10          | 10         | 8             | 2024-01-24    |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

On the right is a Python code editor for a project named "TechShop Assignment 1". The main.py file contains the following code:

```
C:\Users\A2Z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment 1\main.py"
Enter Threshold : 25
Product ID for Products which have low Stocks in Inventory Are :
4
5
6
8
10
Process finished with exit code 0
```

The status bar at the bottom indicates the file is "main.py" and the Python version is "Python 3.11".

ListOutOfStockProducts(): A method to list products that are out of stock.

MySQL 8.0 Command Line Client

```
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductID | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1           | 1          | 50            | 2024-01-15      |
| 2           | 2          | 30            | 2024-01-16      |
| 3           | 3          | 98            | 2024-01-17      |
| 4           | 4          | 20            | 2024-01-18      |
| 5           | 5          | 10            | 2024-01-19      |
| 6           | 6          | 10            | 2024-01-20      |
| 7           | 7          | 55            | 2024-01-21      |
| 8           | 8          | 15            | 2024-01-22      |
| 9           | 9          | 25            | 2024-01-23      |
| 10          | 10         | 8             | 2024-01-24      |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

TechShop Assignment 1

```
Project Run main
C:\Users\az2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment"
No out of Stock Products
Process finished with exit code 0
```

38:35 CRLF UTF-8 4 spaces Python 3.11

ListAllProducts(): A method to list all products in the inventory, along with their quantities.

MySQL 8.0 Command Line Client

```
mysql> select * from Inventory;
+-----+-----+-----+-----+
| InventoryID | ProductID | QuantityInStock | LastStockUpdate |
+-----+-----+-----+-----+
| 1           | 1          | 50            | 2024-01-15      |
| 2           | 2          | 30            | 2024-01-16      |
| 3           | 3          | 98            | 2024-01-17      |
| 4           | 4          | 20            | 2024-01-18      |
| 5           | 5          | 10            | 2024-01-19      |
| 6           | 6          | 10            | 2024-01-20      |
| 7           | 7          | 55            | 2024-01-21      |
| 8           | 8          | 15            | 2024-01-22      |
| 9           | 9          | 25            | 2024-01-23      |
| 10          | 10         | 8             | 2024-01-24      |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

TechShop Assignment 1

```
Project Run main
C:\Users\az2z\AppData\Local\Microsoft\WindowsApps\python3.11.exe "E:\Python\Project1\TechShop Assignment"
Product ID : 1, Quantity In Stock : 50
Product ID : 2, Quantity In Stock : 30
Product ID : 3, Quantity In Stock : 98
Product ID : 4, Quantity In Stock : 20
Product ID : 5, Quantity In Stock : 10
Product ID : 6, Quantity In Stock : 10
Product ID : 7, Quantity In Stock : 55
Product ID : 8, Quantity In Stock : 15
Product ID : 9, Quantity In Stock : 25
Product ID : 10, Quantity In Stock : 8
Process finished with exit code 0
```

40:28 CRLF UTF-8 4 spaces Python 3.11

Database Connectivity

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** TechShop Assignment 1 (Python) is open.
- File:** db_connection.py is the active file, shown in the code editor.
- Code Editor Content:**

```
1 import mysql.connector
2 usages
3 class DBConnector:
4     def __init__(self, host, user, password, port, database):
5         self.host = host
6         self.user = user
7         self.password = password
8         self.port = port
9         self.database = database
10        self.connection = None
11
12    24 usages (24 dynamic)
13    def open_connection(self):
14        if self.connection is None or not self.connection.is_connected():
15            self.connection = mysql.connector.connect(
16                host=self.host,
17                user=self.user,
18                password=self.password,
19                port=self.port,
20                database=self.database
21            )
22
23    24 usages (24 dynamic)
24    def close_connection(self):
25        if self.connection is not None and self.connection.is_connected():
26            self.connection.close()
```
- Toolbars and Menus:** Standard PyCharm toolbars and menus are visible at the top.
- Status Bar:** Shows "201 CRLF UTF-8 4 spaces Python 3.11".