

Assignment 1

Task 1: Classes and Their Attributes:

Task 2: Class Creation:

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

Customers Class:

Attributes:

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

Methods:

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

The screenshot shows a Python code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Run.
- Explorer:** Shows a tree view of files and folders. The current file is `Customers.py`, which is selected in the center panel.
- Code Editor:** Displays the `Customers` class definition. The code includes methods for initializing customer data, setting properties for CustomerId, FirstName, LastName, Email, and Phone, and validating email format. It also includes a `create_customer` function that inserts new customer data into a database.
- Status Bar:** Shows the current line (Ln 150), column (Col 50), spaces (Spaces: 4), encoding (UTF-8), and Python version (3.11.5 64-bit). It also indicates that Go Live and Prettier are available.

```
37     @Email.setter
38     def Email(self,new_Email):
39         if "@" in new_Email and "." in new_Email:
40             self._Email =new_Email
41         else:
42             raise InvalidDataException("Invalid email format.")
43
44     @property
45     def Phone(self):
46         return self._Phone
47     @Phone.setter
48     def Phone(self, new_Phone):
49         if len(new_Phone) == 10 and new_Phone.isdigit():
50             self._Phone = new_Phone
51         else:
52             raise InvalidDataException("Invalid phone number format.")
53
54     @property
55     def Address(self):
56         return self._Address
57     @Address.setter
58     def Address(self, new_Address):
59         if isinstance(new_Address, str):
60             self._Address = new_Address
61         else:
62             raise InvalidDataException("Invalid address format")
63
64     def create_customer(self,CustomerId,FirstName,LastName,Email, Phone,Address):
65         try:
66             self._db_connector.open_connection()
67             cursor = self._db_connector.connection.cursor()
68
69             cursor.execute("SELECT * FROM customers WHERE Email = %s", (Email,))
70             existing_customer = cursor.fetchone()
71
72             if existing_customer:
73                 print("Error: Email address is already in use.")
74             else:
75                 cursor.execute("INSERT INTO customers (CustomerId,FirstName,LastName, Email, Phone,Address) VALUES (%s, %s, %s,%s,%s)", (CustomerId,FirstName,LastName, Email, Phone,Address))
```

```

73     |         print("Error: Email address is already in use.")
74     |     else:
75     |         cursor.execute("INSERT INTO customers (CustomerId,FirstName,LastName, Email, Phone,Address) VALUES (%s, %s, %s,%s,%s)", (Customerid,First_name,Last_name,Email,Phone,Address))
76     |         self._db_connector.connection.commit()
77     |         print("Customer created successfully")
78
79 except Exception as e:
80     |     print(f"Error: {e}")
81
82 finally:
83     |     if cursor:
84     |         cursor.close()
85     |     self._db_connector.close_connection()
86
87 def calculate_Total_Orders(self,Customerid):
88     try:
89         self._db_connector.open_connection()
90         query = """ SELECT COUNT(OrderID) AS TotalOrders FROM Orders
91             WHERE CustomerId = %s"""
92         values = (Customerid,)
93         self._db_connector.cursor.execute(query, values)
94         result = self._db_connector.cursor.fetchone()
95
96         if result:
97             total_orders = result[0]
98             print(f"Total Orders for customer {Customerid}: {total_orders}")
99         else:
100             print("No orders found for this customer")
101
102 except Exception as e:
103     |     print(f"Error calculating total orders: {e}")
104 finally:
105     |     self._db_connector.close_connection()
106
107
108 def get_Customer_Details(self,customerid):
109     try:
110         self._db_connector.open_connection()
111
112         query ="SELECT * FROM customers WHERE CustomerId=%s"
113
114         self._db_connector.cursor.execute(query,values)
115
116         customer_details=self._db_connector.cursor.fetchone()
117
118         if customer_details:
119             print("Customer Details:")
120             print(f"Customer ID:{customer_details[0]}")
121             print(f"First Name: {customer_details[1]}")
122             print(f"Last Name: {customer_details[2]}")
123             print(f"Email: {customer_details[3]}")
124             print(f"Phone: {customer_details[4]}")
125             print(f"Address: {customer_details[5]}")
126         else:
127             print("Customer not found.")
128
129     except Exception as e:
130         |     print(f"Error getting customer details: {e}")
131
132     finally:
133         |     self._db_connector.close_connection()
134
135
136 def update_customer_Info(self,Customerid,new_Email,new_Phone,new_Address):
137     try:
138         self._db_connector.open_connection()
139
140         query="UPDATE customers SET Email=%s,Phone=%s, Address=%s WHERE CustomerId=%s"
141         values=(new_Email,new_Phone,new_Address,Customerid)
142
143         with self._db_connector.connection.cursor() as cursor:
144             cursor.execute(query,values)
145             self._db_connector.connection.commit()
146             print("Customer account updated sucessfully")
147     except Exception as e:
148         |     print(f"Error updating customer account :{e}")
149     finally:
150         |     self._db_connector.close_connection()

```

The screenshot shows the Visual Studio Code interface with the Python extension installed. The Explorer sidebar on the left lists files and folders related to a project named 'Assignment1'. The 'TERMINAL' tab is active, displaying a command-line session:

```
PS C:\Users\krish\OneDrive\Documents\Python> & C:/Python311/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Total Orders for Customer 5: 0
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

The status bar at the bottom indicates the file is 3.11.5 64-bit and the current line and column are Ln 2, Col 33.

This screenshot is nearly identical to the one above, but the terminal output has changed to show specific customer details:

```
PS C:\Users\krish\OneDrive\Documents\Python> & C:/Python311/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Customer Details:
Customer ID:100
First Name: Krishna
Last Name: Patle
Email: krishyappa@gmail.com
Phone: 123456789
Address: Mankapur
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

The status bar at the bottom indicates the file is 3.11.5 64-bit and the current line and column are Ln 17, Col 44.

The screenshot shows the Visual Studio Code interface with the Python extension installed. The Explorer sidebar on the left lists a project structure under 'PYTHON'. The 'TERMINAL' tab is active, displaying a terminal session where the user has connected to a MySQL database. The session output includes:

```
Phone: 1234567890
Address: Mankapur
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python & C:/Python311/python.exe c:/users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Customer account updated successfully
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

The terminal also shows the current working directory as 'C:\Users\krish\OneDrive\Documents\Python'.

The screenshot shows a developer's environment with multiple windows open:

- Browser Tab:** Shows a search bar with "Python" and several tabs related to "Assignment 2".
- Code Editor:** An Explorer sidebar lists Python files: Assignment2.py, Main.py, Inventory.py, Exception.py, and DatabaseConnector.py. The Main.py file is open, displaying code for connecting to MySQL and interacting with Customer and OrderDetails managers.
- Code Editor:** A second code editor window titled "Query 1" contains SQL queries for creating a database and selecting from tables.
- Data Grid:** A "Result Grid" displays a table of customer data with columns: CustomerID, FirstName, LastName, Email, Phone, and Address.
- Output Window:** Shows the command "python311/python.exe c:/Users/krish/OneDrive/Documents/Assignment2/Main.py" and its execution results, including connection details and a "Connection closed" message.
- Action Output:** Shows the execution of two SQL queries: one for "inventory" and one for "customers".

Products Class:

Attributes:

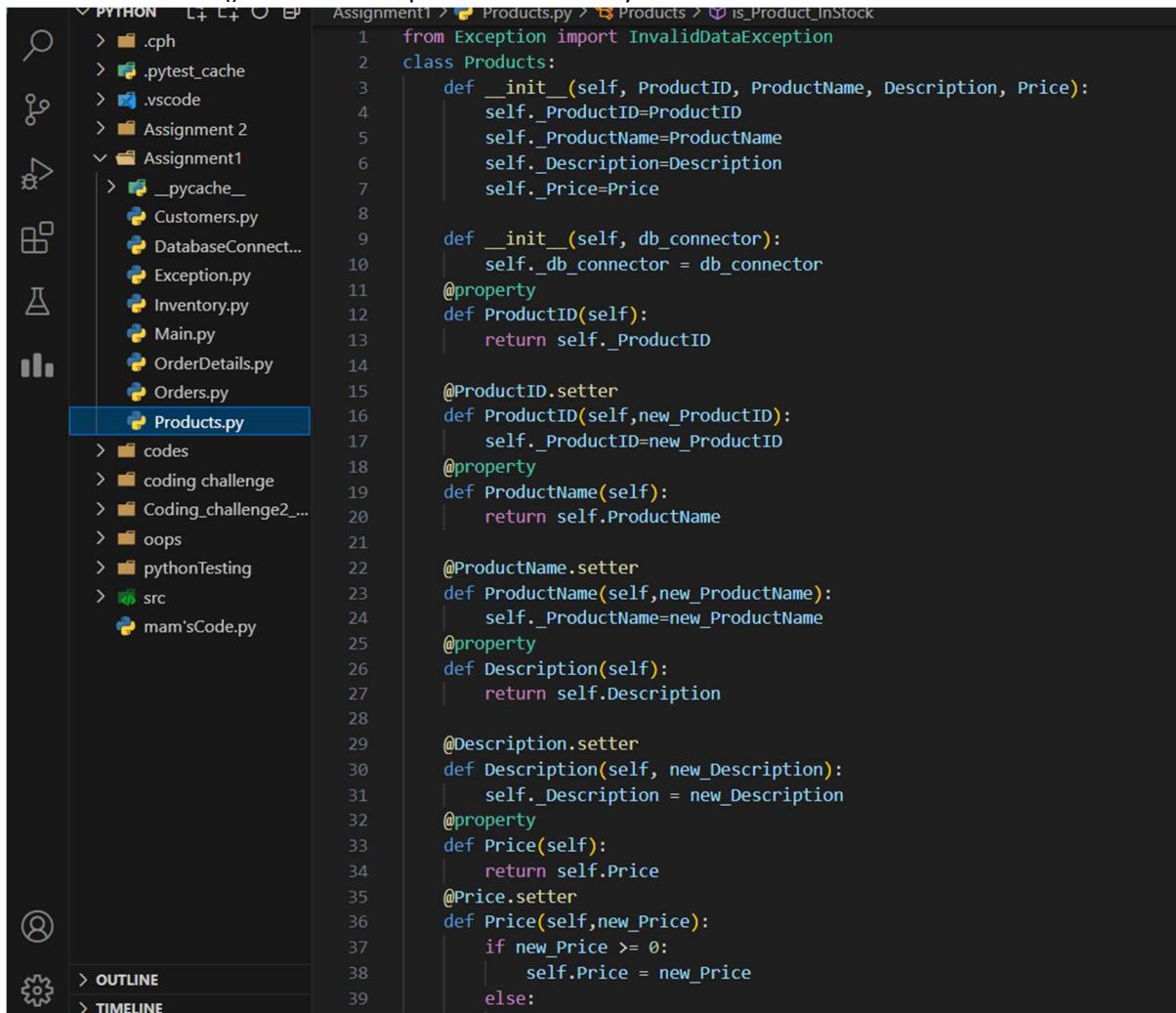
- ProductID (int)
 - ProductName (string)

- Description (string)

- Price (decimal)

Methods:

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



The screenshot shows the VS Code interface with the following details:

- File Explorer:** On the left, it shows a tree view of files. The 'Assignment1' folder is expanded, containing 'Products.py' which is selected and highlighted with a blue border. Other files like 'Customers.py', 'DatabaseConnect...', 'Exception.py', 'Inventory.py', 'Main.py', 'OrderDetails.py', and 'Orders.py' are also listed.
- Code Editor:** The main area displays the Python code for 'Products.py'. The code defines a class 'Products' with methods for initializing product details, setting and getting product ID, name, description, and price, and validating the price.
- Status Bar:** At the bottom, it shows 'Assignment1 > Products.py > Products > is_Product_InStock'.
- Sidebar:** On the far left, there are icons for search, find, replace, and other development tools.

```

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

```

    42     def get_product_details(self, ProductID):
    43         try:
    44             self._db_connector.open_connection()
    45
    46             query = "SELECT * FROM products WHERE ProductID=%s"
    47             values = (ProductID,)
    48
    49             self._db_connector.cursor.execute(query, values)
    50
    51             product_details = self._db_connector.cursor.fetchone()
    52
    53             if product_details:
    54                 print("Product Details:")
    55                 print(f"Product ID: {product_details[0]}")
    56                 print(f"Product Name: {product_details[1]}")
    57                 print(f"Description : {product_details[2]}")
    58                 print(f"Price: {product_details[3]}")
    59
    60             else:
    61                 print("Product Id not found.")
    62
    63         except Exception as e:
    64             print(f"Error getting Product details: {e}")
    65
    66         finally:
    67             self._db_connector.close_connection()
    68
    69     def update_Product_Info(self, ProductID, new_Description, new_Price):
    70         try:
    71             self._db_connector.open_connection()
    72             query = "UPDATE Products SET Description=%s, Price=%s WHERE ProductID=%s"
    73             values = (new_Description, new_Price, ProductID)
    74
    75             with self._db_connector.connection.cursor() as cursor:
    76                 cursor.execute(query, values)
    77                 self._db_connector.connection.commit()
    78                 print("Product Details updated sucessfully")
    79
    80         except Exception as e:

```

```

    70             self._db_connector.cursor.execute(query, values)
    71             self._db_connector.connection.commit()
    72             print("Product Details updated sucessfully")
    73
    74         except Exception as e:
    75             print(f"Error updating Product Details :{e}")
    76
    77         finally:
    78             self._db_connector.close_connection()
    79
    80     def is_Product_InStock(self, ProductID):
    81         try:
    82             self._db_connector.open_connection()
    83             query = "SELECT QuantityInStock FROM Inventory WHERE ProductID = %s"
    84             values = (ProductID,)
    85
    86             self._db_connector.cursor.execute(query, values)
    87             quantity_in_stock = self._db_connector.cursor.fetchone()
    88
    89             if quantity_in_stock[0]>0:
    90                 print("Product is in stock and stockquantity is", quantity_in_stock[0])
    91             else:
    92                 print("Product is not in stock")
    93
    94         except Exception as e:
    95             print(f"Error checking product stock: {e}")
    96             return False
    97
    98         finally:
    99             self._db_connector.close_connection()

```

Screenshot of the Visual Studio Code interface. The terminal window shows the output of a MySQL connection command:

```
Connected to MySQL database
Product Details:
Product ID:1
Product Name: Laptop
Description : MI Notebook 14 is 10th Gen
Price: 44000.00
Connection Closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

Screenshot of the Visual Studio Code interface. The terminal window shows the output of a MySQL connection command, which fails due to a syntax error:

```
Description : MI Notebook 14 is 10th Gen
Price: 44000.00
Connection Closed
ps C:\Users\krish\OneDrive\Documents\Python> & C:/Python311/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product Details updated successfully
Connection Closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwareware
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop
- Tables
- Views
- Stored Procedures
- Functions
- ticketbookingystem
- world

Administration Schemas Information

No object selected

Query 1

```

1423 • SELECT DISTINCT s.SuspectID, s.Name
1424   FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434

```

Result Grid

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2023-10-12
2	2	15	2023-01-02
3	3	9	2023-09-09
4	4	8	2023-10-03
5	5	12	2022-11-20
6	6	23	2023-03-09
7	7	30	2023-03-10
8	8	25	2022-10-19
9	9	5	2023-05-04
10	10	10	2023-10-10

Action Output

#	Time	Action	Message
1	5 22:02:22	select * from products LIMIT 0, 1000	11 rows returned
2	6 22:02:26	select * from inventory LIMIT 0, 1000	10 rows returned

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwareware
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop
- Tables
- Views
- Stored Procedures
- Functions
- ticketbookingystem
- world

Administration Schemas Information

No object selected

Query 1

```

1422
1423 • SELECT DISTINCT s.SuspectID, s.Name
1424   FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434

```

Result Grid

ProductID	ProductName	Description	Price
1	Laptop	MSI Notebook i5 10th Gen	44000.00
2	Laptop	MSI Notebook i4 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	33000.00
5	IQ Z5 Lite	TQ Z5	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	Iphone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	55000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00

Action Output

#	Time	Action	Message
1	6 22:02:26	select * from inventory LIMIT 0, 1000	10 rows returned
2	7 22:04:31	select * from products LIMIT 0, 1000	11 rows returned

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwareware
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop
- Tables
- Views
- Stored Procedures
- Functions
- ticketbookingystem
- world

Administration Schemas Information

No object selected

Query 1

```

1422
1423 • SELECT DISTINCT s.SuspectID, s.Name
1424   FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434

```

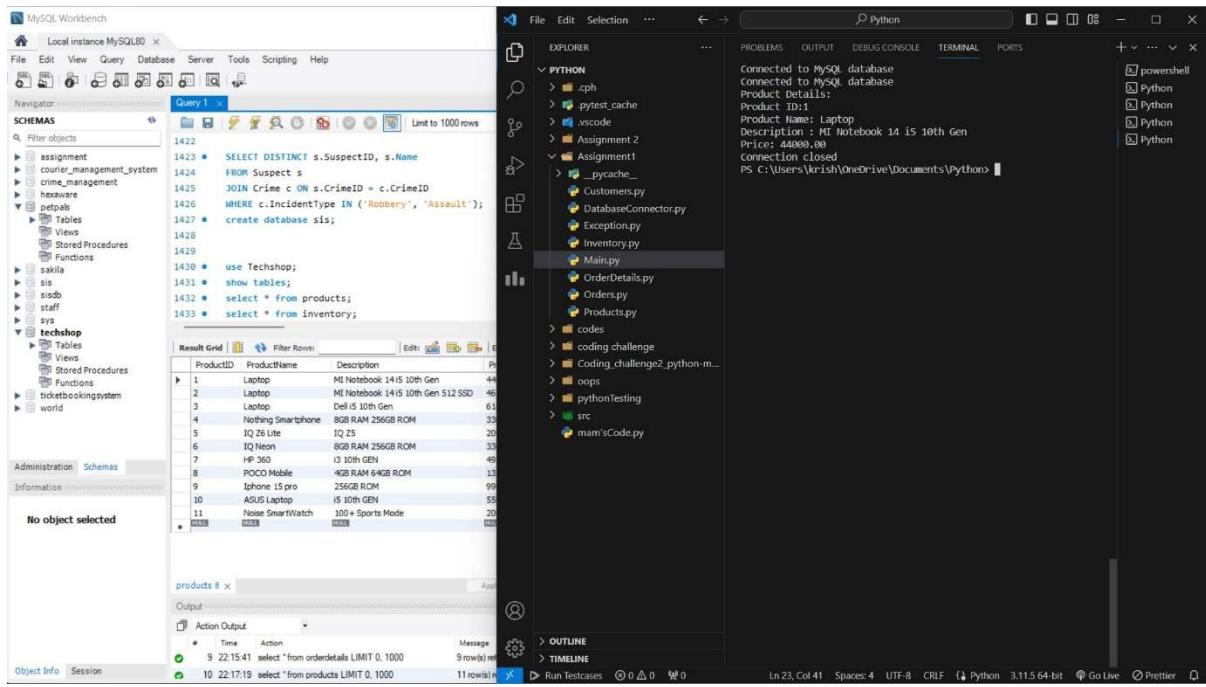
Result Grid

ProductID	ProductName	Description	Price
1	Laptop	MSI Notebook i5 10th Gen	44000.00
2	Laptop	MSI Notebook i4 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	33000.00
5	IQ Z5 Lite	TQ Z5	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	Iphone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	55000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00

Action Output

#	Time	Action	Message
1	6 22:02:26	select * from inventory LIMIT 0, 1000	10 rows returned
2	7 22:04:31	select * from products LIMIT 0, 1000	11 rows returned

Object Info Session



Orders Class:

Attributes:

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

Methods:

- CalculateTotalAmount() - Calculate the total amount of the order.
- GetOrderDetails(): Retrieves and displays the details of the order (e.g., product list and quantities).
- UpdateOrderStatus(): Allows updating the status of the order (e.g., processing, shipped).
- CancelOrder(): Cancels the order and adjusts stock levels for products.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for search, file management, and project navigation. Below the sidebar, there are two tabs: 'OUTLINE' and 'TIMELINE'. The main area displays the following Python code:

```
from datetime import datetime
from Exception import InvalidDataException

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

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

    @property
    def CustomerID(self):
        return self._CustomerID
    @CustomerID.setter
    def CustomerID(self, new_CustomerID):
        self._CustomerID = new_CustomerID

    @property
    def OrderDate(self):
        return self._OrderDate
    @OrderDate.setter
    def OrderDate(self, new_OrderDate):
        self._OrderDate = new_OrderDate

    @property
    def TotalAmount(self):
        return self._TotalAmount
    @TotalAmount.setter
    def TotalAmount(self, new_Totalamount):

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

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for search, file management, and project navigation. Below the sidebar, there are two tabs: 'OUTLINE' and 'TIMELINE'. The main area displays the following Python code:

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

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

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

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

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

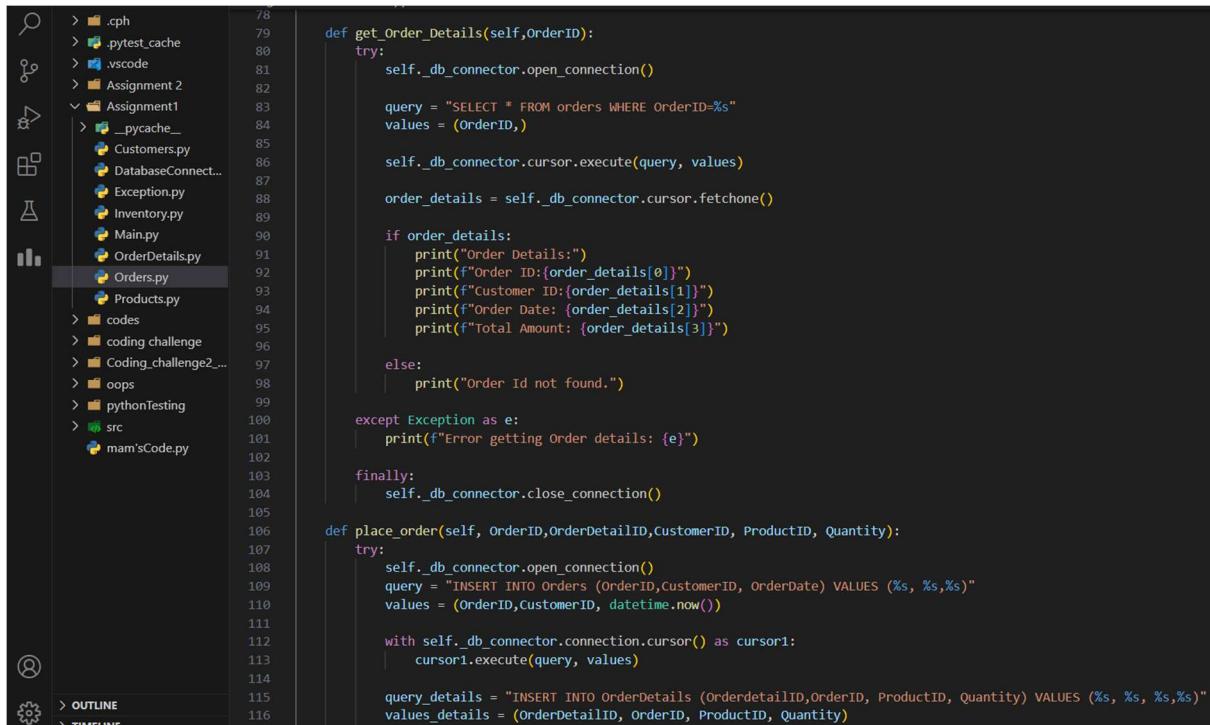
        with self._db_connector.connection.cursor() as cursor_calculate_total_amount:
            cursor_calculate_total_amount.execute(calculate_total_amount, values1)
            total_amount = cursor_calculate_total_amount.fetchone()[0]

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

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

    finally:
        self._db_connector.close_connection()

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



```

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

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

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

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

        order_details = self._db_connector.cursor.fetchone()

        if order_details:
            print("Order Details:")
            print(f"Order ID: {order_details[0]}")
            print(f"Customer ID: {order_details[1]}")
            print(f"Order Date: {order_details[2]}")
            print(f"Total Amount: {order_details[3]}")

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

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

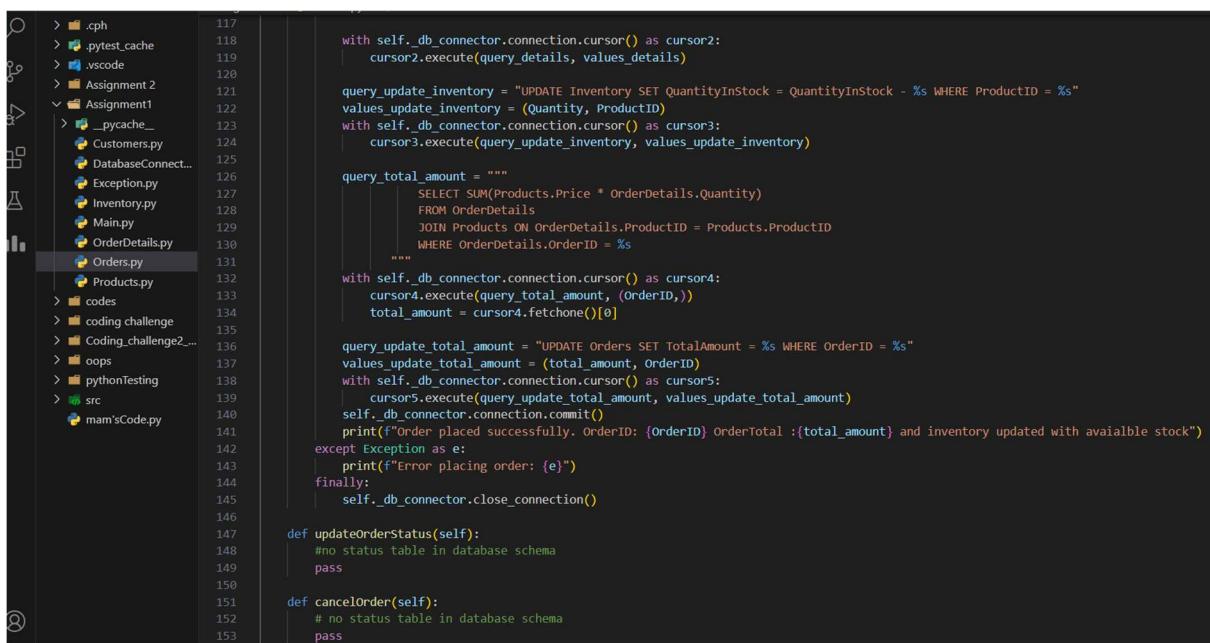
    finally:
        self._db_connector.close_connection()

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

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

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

```



```

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

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

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

query_total_amount = """
    SELECT SUM(Products.Price * OrderDetails.Quantity)
    FROM OrderDetails
    JOIN Products ON OrderDetails.ProductID = Products.ProductID
    WHERE OrderDetails.OrderID = %s
"""
with self._db_connector.connection.cursor() as cursor4:
    cursor4.execute(query_total_amount, (OrderID,))
    total_amount = cursor4.fetchone()[0]

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

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

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

```

```

Connected to MySQL database
Error calculating total amount: 1054 (42S22): Unknown column 'OrderDetails.Quantity' in 'Field list'
PS C:\Users\krish\OneDrive\Documents\Python> & C:/Python311/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Total amount for OrderID 2: 466400.00
Connection Closed
PS C:\Users\krish\OneDrive\Documents\Python>

```

OrderDetailID	OrderID	ProductID	Quantity
1	1	1	1
2	1	3	1
3	4	1	1
4	2	1	5
5	7	9	1
6	10	7	1
8	10	9	1
9	2	3	4
10	7	10	1

orderdetails6 x

Action	Time	Action	Message
7 22:04:31	select * from products LIMIT 0, 1000		11 row(s) returned
8 22:13:04	select * from orderdetails LIMIT 0, 1000		9 row(s) returned

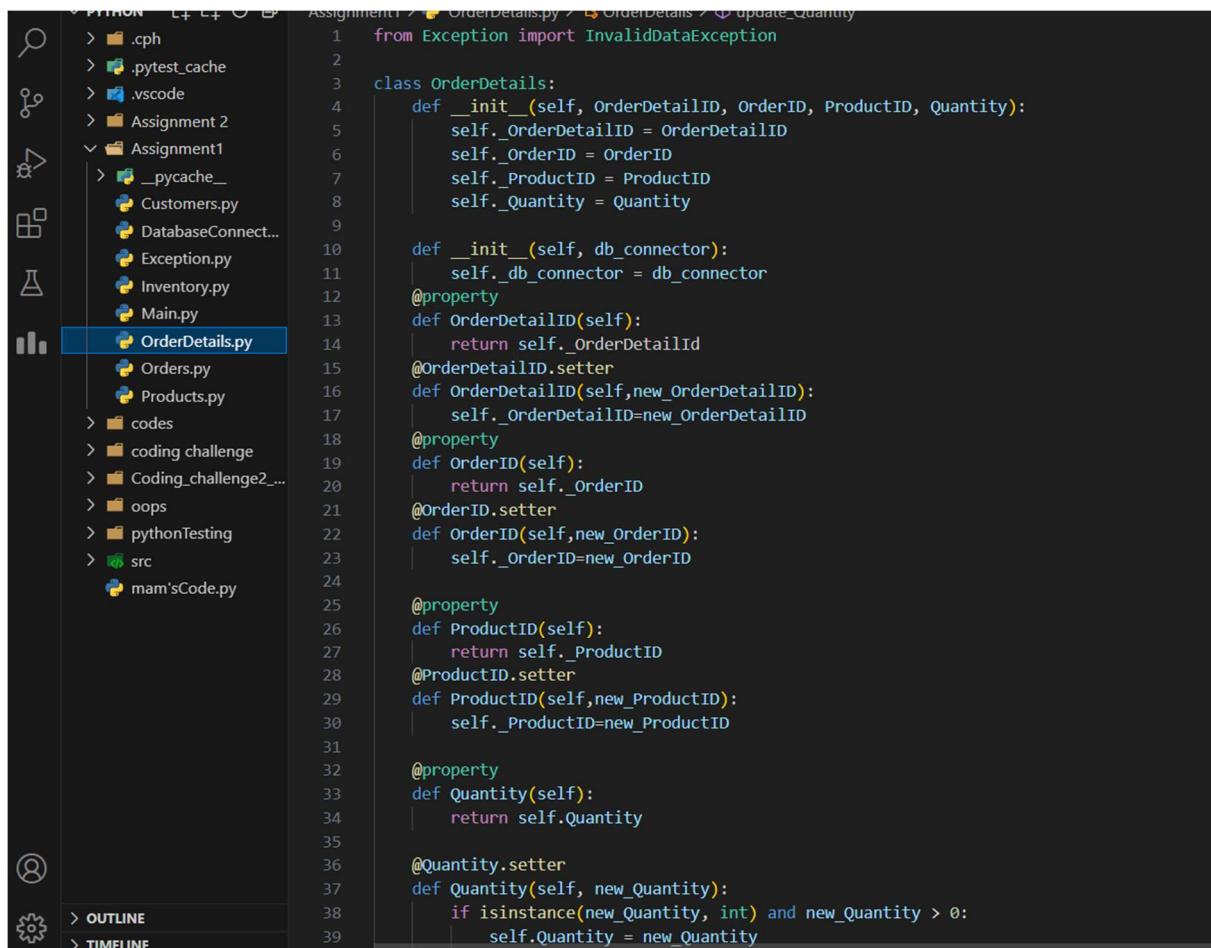
OrderDetails Class:

Attributes:

- OrderDetailID (int)
- Order (Order) - Use composition to reference the Order to which this detail belongs.
- Product (Product) - Use composition to reference the Product included in the order detail.
- Quantity (int)

Methods:

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



The screenshot shows the VS Code interface with the following details:

- File Explorer:** On the left, it shows a tree view of files and folders. The current file, `OrderDetails.py`, is highlighted with a blue selection bar.
- Code Editor:** The main area contains the Python code for the `OrderDetails` class. The code includes several properties with both `getter` and `setter` methods, and a validation check for the `Quantity` setter.
- Status Bar:** At the bottom, it shows navigation links: `> OUTLINE` and `> TIMELINE`.

```
Assignment1 > OrderDetails.py > OrderDetails > update_Quantity
1  from Exception import InvalidDataException
2
3  class OrderDetails:
4      def __init__(self, OrderDetailID, OrderID, ProductID, Quantity):
5          self._OrderDetailID = OrderDetailID
6          self._OrderID = OrderID
7          self._ProductID = ProductID
8          self._Quantity = Quantity
9
10     def __init__(self, db_connector):
11         self._db_connector = db_connector
12     @property
13     def OrderDetailID(self):
14         return self._OrderDetailID
15     @OrderDetailID.setter
16     def OrderDetailID(self,new_OrderDetailID):
17         self._OrderDetailID=new_OrderDetailID
18     @property
19     def OrderID(self):
20         return self._OrderID
21     @OrderID.setter
22     def OrderID(self,new_OrderID):
23         self._OrderID=new_OrderID
24
25     @property
26     def ProductID(self):
27         return self._ProductID
28     @ProductID.setter
29     def ProductID(self,new_ProductID):
30         self._ProductID=new_ProductID
31
32     @property
33     def Quantity(self):
34         return self.Quantity
35
36     @Quantity.setter
37     def Quantity(self, new_Quantity):
38         if isinstance(new_Quantity, int) and new_Quantity > 0:
39             self.Quantity = new_Quantity
```

```
def calculate_subtotal(self, OrderDetailID):
    try:
        self._db_connector.open_connection()
        query = "SELECT OD.Qunatity, P.Price FROM Orderdetails OD INNER JOIN Products P ON OD.ProductID = P.ProductID WHERE OrderDetailID = %s"
        values = (OrderDetailID,)

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

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

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

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

        orderdetails_details = self._db_connector.cursor.fetchone()

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

```

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

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

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

    finally:
        self._db_connector.close_connection()

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

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

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

    finally:
        self._db_connector.close_connection()

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

```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigators

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwavare
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop
- Tables
- Views
- Stored Procedures
- Functions
- ticketbookingystem
- world

Query 1

```

1424 FROM Suspect s
1425 JOIN Crime c ON s.CrimeID = c.CrimeID
1426 WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434 • select * from orderdetails;
1435

```

Result Grid

OrderDetailID	OrderID	ProductID	Quantity
1	1	1	1
2	1	3	1
3	4	1	1
4	2	1	5
5	7	9	1
6	10	7	1
8	10	9	1
9	2	3	4
10	7	10	1
11			

No object selected

orderdetails6 X

Action Output

Time	Action	Message
7 22:04:31	select * from products LIMIT 0, 1000	11 rows(s) returned
8 22:13:04	select * from orderdetails LIMIT 0, 1000	9 rows(s) returned

Object Info Session

Ln 37, Col 43 Spaces: 4 UFT-8 CRLF ¶ Go Live ▾

MySQL Workbench

Local instance MySQL80 X

File Edit Selection ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Connected to MySQL database

Order detail not found.

Connection closed

PS C:\Users\krish\OneDrive\Documents\Python> & C:\Python311\pyt hon.exe c:/users/krish/onedrive/documents/python/Assignment1/Main.py

Connected to MySQL database

Connected for OrderDetailID: 2

Subtotal for OrderDetailID: 2: 61600.00

Connection closed

PS C:\Users\krish\OneDrive\Documents\Python>

EXPLORER

- PYTHON
 - .cph
 - pytest_cache
 - vscode
 - Assignment 2
 - Assignment1
 - __pycache__
 - Customers.py
 - DatabaseConnector.py
 - Exception.py
 - Inventory.py
 - Main.py
 - OrderDetails.py
 - Orders.py
 - Products.py
 - codes
 - coding challenge
 - Coding_challenge2_python-m...
 - oops
 - pythonTesting
 - src
 - main'sCode.py

OUTLINE

TIMELINE

Ln 35, Col 46 Spaces: 4 UFT-8 CRLF ¶ Go Live ▾

The screenshot shows two integrated development environments side-by-side. On the left is MySQL Workbench, displaying a query editor with several SQL statements related to a 'Suspect' table and its relationships with other tables like 'Crime' and 'OrderDetails'. A result grid shows 10 rows of data from the 'orderdetails' table. On the right is PyCharm, showing a terminal window where a Python script named 'Main.py' is being run. The terminal output indicates a connection to a MySQL database and a successful update of a 'Quantity' field. Below the terminal are file navigation panels for both MySQL Workbench and PyCharm.

Inventory class:

Attributes:

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

Methods:

- GetProduct(): A method to retrieve the product associated with this inventory item.
- GetQuantityInStock(): A method to get the current quantity of the product in stock.
- AddToInventory(int quantity): A method to add a specified quantity of the product to the inventory.
- RemoveFromInventory(int quantity): A method to remove a specified quantity of the product from the inventory.
- UpdateStockQuantity(int newQuantity): A method to update the stock quantity to a new value.
- IsProductAvailable(int quantityToCheck): A method to check if a specified quantity of the product is available in the inventory.
- GetInventoryValue(): A method to calculate the total value of the products in the inventory based on their prices and quantities.

- **ListLowStockProducts(int threshold):** A method to list products with quantities below a specified threshold, indicating low stock.
- **ListOutOfStockProducts():** A method to list products that are out of stock
- **ListAllProducts():** A method to list all products in the inventory, along with their quantities.

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

1422
1423 • SELECT DISTINCT s.SuspectID, s.Name
1424 FROM Suspect s
1425 JOIN Crime c ON s.CrimeID = c.CrimeID
1426 WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;

```

The results grid shows the following data:

ProductID	ProductName	Description	Price
1	Laptop	MI Notebook 14 i5 10th Gen	44000.00
2	Laptop	MI Notebook 14 i5 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	33000.00
5	IQ Z6 Lite	TZ Z5	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	iPhone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	59000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00

The Python terminal shows the following output:

```

PS C:\Users\krish\OneDrive\Documents\Python> & c:/Python311/python.exe c:/users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product details updated successfully
Connection closed
> & c:/Python311/python.exe c:/users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Product is in stock and stock quantity is 10
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python> 

```

The screenshot shows the MySQL Workbench interface with a query editor containing the same SQL code as the previous screenshot:

```

1422
1423 • SELECT DISTINCT s.SuspectID, s.Name
1424 FROM Suspect s
1425 JOIN Crime c ON s.CrimeID = c.CrimeID
1426 WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;

```

The results grid shows the same data as the previous screenshot:

ProductID	ProductName	Description	Price
1	Laptop	MI Notebook 14 i5 10th Gen	44000.00
2	Laptop	MI Notebook 14 i5 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	33000.00
5	IQ Z6 Lite	TZ Z5	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	iPhone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	59000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00

The Python terminal shows the following output:

```

Connected to MySQL database
Connected to MySQL database
Product ID:1
Product Name: Laptop
Description : MI Notebook 14 i5 10th Gen
Price: 44000.00
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python> 

```

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwavare
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop

Query 1

```

1422 • SELECT DISTINCT s.SuspectID, s.Name
1423   FROM Suspect s
1424   JOIN Crime c ON s.CrimeID = c.CrimeID
1425   WHERE c.IncidentType IN ('Robbery', 'Assault');
1426
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;

```

Result Grid

ProductID	ProductName	Description	Price
1	Laptop	MI Notebook 14i5 10th Gen	44000.00
2	Laptop	MI Notebook 14i5 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	38000.00
4	Mobile SmartPhone	8GB RAM 256GB ROM	33000.00
5	IQ26 Lite	IQ Z5	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49000.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	Iphone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	55000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00

Action Output

#	Time	Action	Message
6	22:02:26	select * from inventory LIMIT 0, 1000	10 rows(s) returned
7	22:04:31	select * from products LIMIT 0, 1000	11 rows(s) returned

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwavare
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop

Query 1

```

1424 • FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434 • select * from orderdetails;
1435

```

Result Grid

OrderDetailID	OrderID	ProductID	Quantity
1	1	1	1
2	1	3	3
3	4	1	1
4	2	1	5
5	7	9	1
6	10	7	1
7	8	10	9
8	2	3	4
9	7	10	1

Action Output

#	Time	Action	Message
8	22:13:04	select * from orderdetails LIMIT 0, 1000	9 row(s) returned
9	22:15:41	select * from orderdetails LIMIT 0, 1000	9 row(s) returned

Object Info Session

MySQL Workbench

Local instance MySQL80 X

File Edit Selection ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

EXPLORER

PYTHON

- .cph
- pytest_cache
- vscode
- Assignment 2
- Assignment1
- __pycache__
- Customers.py
- DatabaseConnector.py
- Exception.py
- Inventory.py
- Main.py
- OrderDetails.py
- Orders.py
- Products.py
- codes
- coding challenge
- Coding_challenge2_python-m...
- oops
- pythonTesting
- src
- main'sCode.py

Connected to MySQL database
Error updating Quantity :1054 (42S22): Unknown column 'Quantity' in 'field list'
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python> & C:/Python311/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py
Connected to MySQL database
Connected to MySQL database
Quantity updated successfully
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- assignment
- courier_management_system
- crime_management
- hwavare
- petpals
- sakila
- sis
- sisdb
- staff
- sys
- techshop

Query 1

```

1424 • FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434 • select * from orderdetails;
1435

```

Result Grid

OrderDetailID	OrderID	ProductID	Quantity
1	1	1	1
2	1	3	3
3	4	1	1
4	2	1	5
5	7	9	1
6	10	7	1
7	8	10	9
8	2	3	4
9	7	10	1

Action Output

#	Time	Action	Message
8	22:13:04	select * from orderdetails LIMIT 0, 1000	9 row(s) returned
9	22:15:41	select * from orderdetails LIMIT 0, 1000	9 row(s) returned

Object Info Session

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows a folder structure under PYTHON containing .cph, .pytest_cache, .vscode, Assignment 2, Assignment1, and Main.py.
- TERMINAL:** Displays a MySQL connection status and a command-line session:

```
Connected to MySQL database
Connected to MySQL database
Product Details:
Product ID:1
Product Name: Laptop
Description : MI Notebook 14 i5 10th Gen
Price: 44000.00
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

- OUTPUT:** Shows the results of the MySQL query executed in the terminal.
- SQL ADDITIONS:** A floating window displays the query and its results in a grid format.

ProductID	ProductName	Description	Price
1	Laptop	MI Notebook 14 i5 10th Gen	44000.00
2	Laptop	MI Notebook 14 i5 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	32000.00
5	IQ 26 Lite	iQ 26	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	Iphone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	55000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00
12	None	None	None

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows a folder structure under PYTHON containing .cph, .pytest_cache, .vscode, Assignment 2, Assignment1, and Main.py.
- TERMINAL:** Displays a MySQL connection status and a command-line session:

```
Connected to MySQL database
Connected to MySQL database
Product ID: 8, Product Name: POCO Mobile, Description: 4GB RAM 64GB ROM, Price: rs13200.00
Connection closed
PS C:\Users\krish\OneDrive\Documents\Python>
```

- OUTPUT:** Shows the results of the MySQL query executed in the terminal.
- SQL ADDITIONS:** A floating window displays the query and its results in a grid format.

ProductID	ProductName	Description	Price
1	Laptop	MI Notebook 14 i5 10th Gen	44000.00
2	Laptop	MI Notebook 14 i5 10th Gen 512 SSD	46200.00
3	Laptop	Dell i5 10th Gen	61600.00
4	Nothing Smartphone	8GB RAM 256GB ROM	32000.00
5	IQ 26 Lite	iQ 26	20000.00
6	IQ Neon	8GB RAM 256GB ROM	33000.00
7	HP 360	i3 10th GEN	49500.00
8	POCO Mobile	4GB RAM 64GB ROM	13200.00
9	Iphone 15 pro	256GB ROM	99000.00
10	ASUS Laptop	i5 10th GEN	55000.00
11	Noise SmartWatch	100+ Sports Mode	2000.00
12	None	None	None

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows a folder named "PYTHON" containing files like .cph, pytest.cache, .vscode, Assignment1, Assignment2, Main.py, OrderDetails.py, Orders.py, Products.py, codes, coding challenge, Coding_challenge2, oops, pythonTesting, and src/mam'sCode.py.
- PROBLEMS:** Displays a terminal output window showing a successful connection to a MySQL database and the creation of a database named "sis".
- DEBUG CONSOLE:** Shows a powershell session connected to the MySQL database, running queries to select data from Suspect, Crime, and Orderdetails tables.
- TERMINAL:** Shows the command "python11/python.exe c:/Users/krish/OneDrive/Documents/Python/Assignment1/Main.py" being run.
- OUTPUT:** Shows the results of the database queries in a grid format.
- SQL ADDITIONS:** A floating window provides context help for the current caret position.

This screenshot is nearly identical to the first one, showing the same VS Code environment and database interactions. The main difference is in the "ACTION OUTPUT" section of the SQL ADDITIONS panel, which now includes two additional log entries:

Time	Action	Message	Duration / Fetch
12 22:19:40	select * from products LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
13 22:21:14	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows the VS Code interface on the left and MySQL Workbench on the right. In the VS Code Explorer, there is a folder named 'PYTHON' containing several files like Main.py, OrderDetails.py, Orders.py, and Products.py. The terminal shows a connection to a MySQL database and some sample queries.

MySQL Workbench Query Grid:

```

SELECT DISTINCT s.SuspectID, s.Name
FROM Suspect s
JOIN Crime c ON s.CrimeID = c.CrimeID
WHERE c.IncidentType IN ('Robbery', 'Assault');
CREATE DATABASE sis;
USE Techshop;
SHOW TABLES;
SELECT * FROM products;
SELECT * FROM inventory;
SELECT * FROM orderdetails;
    
```

MySQL Workbench Result Grid:

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2023-10-12
2	2	15	2023-01-02
3	3	9	2023-12-09
4	4	8	2023-10-05
5	5	12	2022-11-20
6	6	23	2023-02-09
7	7	30	2023-03-10
8	8	25	2022-10-19
9	9	5	2023-05-04
10	10	10	2023-10-10

MySQL Workbench Action Output:

Time	Action	Message	Duration / Fetch
15:22:23.24	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
16:22:24.17	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows the VS Code interface on the left and MySQL Workbench on the right. In the VS Code Explorer, there is a folder named 'PYTHON' containing several files like Main.py, OrderDetails.py, Orders.py, and Products.py. The terminal shows a connection to a MySQL database and some sample queries.

MySQL Workbench Query Grid:

```

SELECT DISTINCT s.SuspectID, s.Name
FROM Suspect s
JOIN Crime c ON s.CrimeID = c.CrimeID
WHERE c.IncidentType IN ('Robbery', 'Assault');
CREATE DATABASE sis;
USE Techshop;
SHOW TABLES;
SELECT * FROM products;
SELECT * FROM inventory;
SELECT * FROM orderdetails;
    
```

MySQL Workbench Result Grid:

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2023-10-12
2	2	15	2023-01-02
3	3	9	2023-12-09
4	4	8	2023-10-05
5	5	12	2022-11-20
6	6	23	2023-02-09
7	7	30	2023-03-10
8	8	25	2022-10-19
9	9	5	2023-05-04
10	10	10	2023-10-10

MySQL Workbench Action Output:

Time	Action	Message	Duration / Fetch
16:22:24.17	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
17:22:24.54	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows a dual-monitor setup. On the left monitor, the Visual Studio Code interface is visible, displaying a Python project structure with files like Main.py, OrderDetails.py, Orders.py, and Products.py. The terminal tab shows a MySQL connection and some sample queries. On the right monitor, the phpMyAdmin interface is open, showing a query editor with the following SQL code:

```

1423 • SELECT DISTINCT s.SuspectID, s.Name
1424   FROM Suspect s
1425   JOIN Crime c ON s.CrimeID = c.CrimeID
1426   WHERE c.IncidentType IN ('Robbery', 'Assault');
1427 • create database sis;
1428
1429
1430 • use Techshop;
1431 • show tables;
1432 • select * from products;
1433 • select * from inventory;
1434 • select * from orderdetails;

```

The result grid displays the following data:

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2023-10-12
2	2	15	2023-01-02
3	3	9	2023-09-09
4	4	8	2023-10-05
5	5	12	2022-11-20
6	6	23	2023-02-09
7	7	30	2023-03-10
8	8	25	2022-10-19
9	9	5	2023-05-04
10	10	10	2023-10-10
11	11	10	2023-10-10
12	12	10	2023-10-10
13	13	10	2023-10-10
14	14	10	2023-10-10
15	15	10	2023-10-10
16	16	10	2023-10-10
17	17	10	2023-10-10

The output panel shows the results of the last two queries:

Action	Time	Action	Message	Duration / Fetch
16	22:24:17	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
17	22:24:54	select * from inventory LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

This screenshot is nearly identical to the one above, showing the same Python project in VS Code and the same MySQL database query in phpMyAdmin. The result grid and output panel also show the same data and results.

The screenshot displays a development environment with multiple windows open:

- EXPLORER**: Shows the project structure under **PYTHON**, including files like .cph, pytest.cache, .vscode, Assignment1, Assignment2, Main.py, Inventory.py, OrderDetails.py, Orders.py, Products.py, and various challenge files.
- TERMINAL**: Shows the command-line output of running `python.exe` on `Main.py`. It connects to a MySQL database named `Techshop` and lists products with their IDs, names, and quantities.
- OUTPUT**: Shows the **Action Output** pane with two log entries related to the MySQL queries.
- Query Editor**: A MySQL query editor window titled **Query 1** containing several SELECT statements.
- Result Grid**: A table showing the results of a query, listing **InventoryID**, **ProductID**, **QuantityInStock**, and **LastStockUpdate**.

InventoryID	ProductID	QuantityInStock	LastStockUpdate
1	1	10	2023-10-12
2	2	15	2023-01-02
3	3	9	2023-09-09
4	4	8	2023-10-03
5	5	12	2023-11-20
6	6	23	2023-03-09
7	7	30	2023-03-10
8	8	25	2023-10-19
9	9	5	2023-05-04
10	10	10	2023-10-10

Action Output pane:

Action	Time	Message	Duration / Fetch
select * from inventory LIMIT 0, 1000	16 22:24:17	10 row(s) returned	0.000 sec / 0.000 sec
select * from inventory LIMIT 0, 1000	17 22:24:54	10 row(s) returned	0.000 sec / 0.000 sec