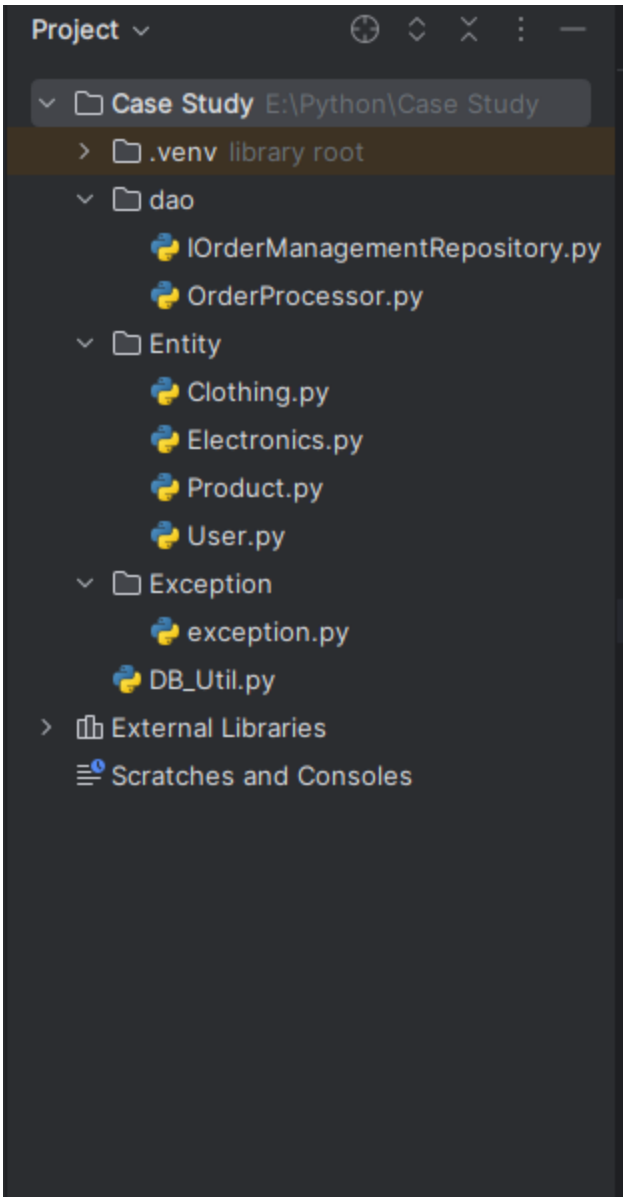


Create SQL Schema from the product and user class, use the class attributes for table column names.

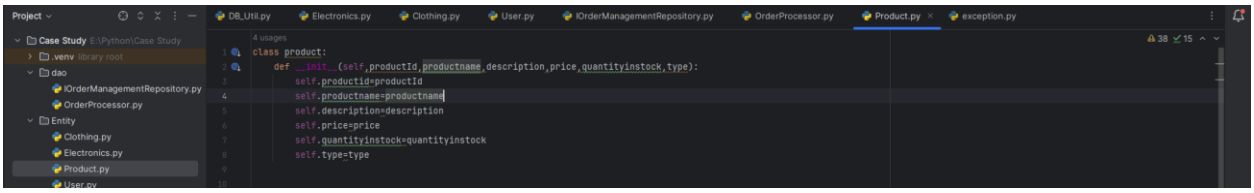
1. Create a base class called Product with the following attributes:

- productId (int)
- productName (String)
- description (String)
- price (double)
- quantityInStock (int)
- type (String) [Electronics/Clothing]

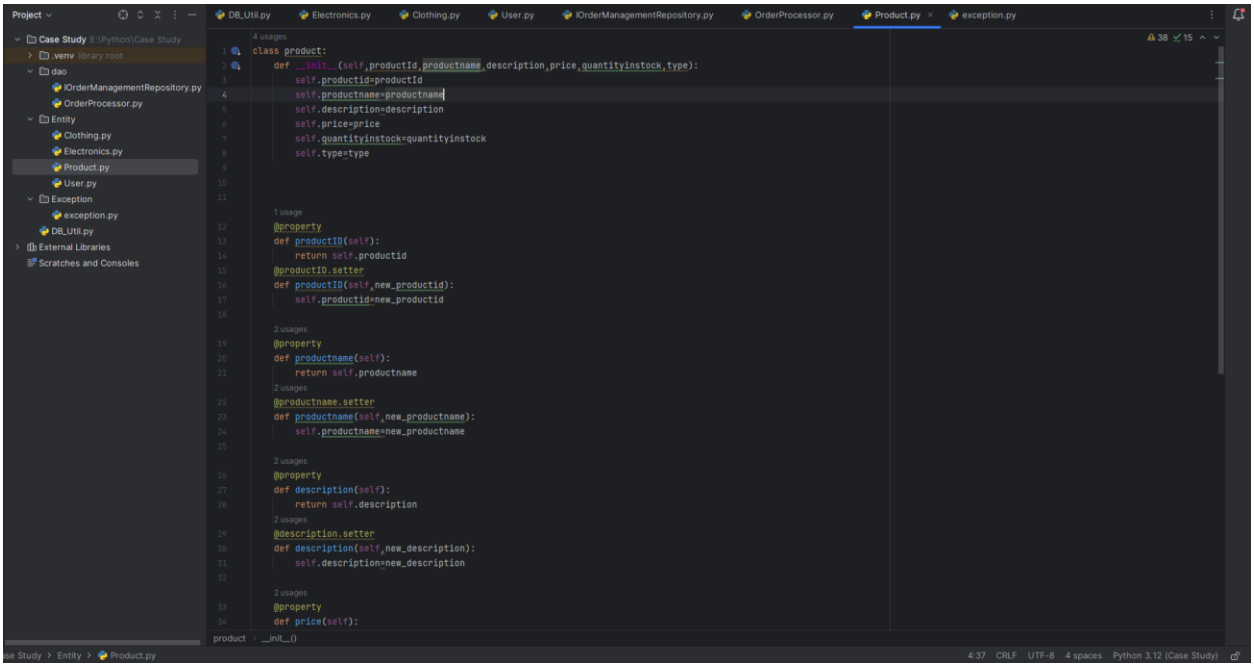
```
mysql> create table Product(
    -> productID int primary key,
    -> productName text,
    -> description text,
    -> price int,
    -> QuantityInStock int,
    -> type enum("Electronics","Clothing"));
Query OK, 0 rows affected (0.04 sec)

mysql> create table user(
    -> userID int primary key,
    -> userName text,
    -> password text,
    -> role enum("Admin","User"));
Query OK, 0 rows affected (0.03 sec)
```

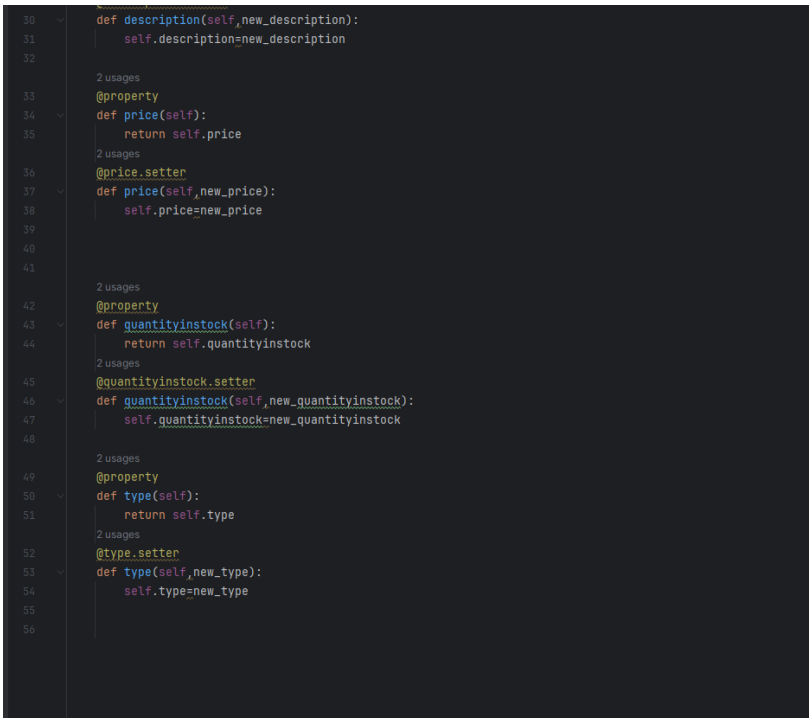
```
mysql> create table Orders(
    -> OrderID int primary key,
    -> userID int,
    -> foreign key(userID) references user(userID),
    -> productID int,
    -> foreign key(productID) references Product(productID),
    -> Quantity int,
    -> TotalAmount int);
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> create table Orders(
    -> OrderID int primary key,
    -> userID int,
    -> foreign key(userID) references user(userID),
    -> productID int,
    -> foreign key(productID) references Product(productID),
    -> Quantity int,
    -> TotalAmount int);
Query OK, 0 rows affected (0.04 sec)
```



2. Implement constructors, getters, and setters for the Product class.





3. Create a subclass Electronics that inherits from Product. Add attributes specific to electronics products, such as:

- brand (String)
- warrantyPeriod (int)

```python
from Product import product

class electronics(product):
    def __init__(self,productId,brand,warrantyperiod):
        super().__init__(productId)
        self.brand=brand
        self.warrantyperiod=warrantyperiod


    2 usages
    @property
    def brand(self):
        return self.brand
    2 usages
    @brand.setter
    def brand(self,new_brand):
        self.brand=new_brand

    2 usages
    @property
    def warrantyperiod(self):
        return self.warrantyperiod
    2 usages
    @warrantyperiod.setter
    def warrantyperiod(self,new_warrantyperiod):
        self.warrantyperiod=new_warrantyperiod
```

4. Create a subclass Clothing that also inherits from Product. Add attributes specific to clothing products, such as:

• size (String)

• color (String)

```python
from Product import product


class clothing(product):
    def __init__(self, productId, size, color):
        super().__init__(productId)
        self.size = size
        self.color = color

    2 usages
    @property
    def size(self):
        return self.size

    2 usages
    @size.setter
    def size(self, new_size):
        self.size = new_size

    2 usages
    @property
    def color(self):
        return self.color

    2 usages
    @color.setter
    def color(self, new_color):
        self.color = new_color

```

5. Create a User class with attributes:

- userId (int)
- username (String)
- password (String)
- role (String) // "Admin" or "User"

```python

class user:
    def __init__(self,userid,username,password,role):
        self.userid=userid
        self.username=username
        self.password=password
        self.role=role

    2 usages
    @property
    def userid(self):
        return self.userid

    2 usages
    @userid.setter
    def userid(self, new_userid):
        self.userid = new_userid

    2 usages
    @property
    def username(self):
        return self.username

    2 usages
    @username.setter
    def username(self, new_username):
        self.username = new_username

    2 usages
    @property
    def password(self):
        return self.password

    2 usages
    @password.setter
    def password(self, new_password):
        self.password = new_password

    2 usages
    @property
    def role(self):
```

```python

    2 usages
    @property
    def role(self):
        return self.role

    2 usages
    @role.setter
    def role(self, new_role):
        self.role = new_role
```

6. Define an interface/abstract class named IOrderManagementRepository with methods for:

• createOrder(User user, list of products): check the user as already present in database to create order or create user (store in database) and create order.

• cancelOrder(int userId, int orderId): check the userid and orderId already present in database and cancel the order. if any userId or orderId not present in database throw exception corresponding UserNotFound or OrderNotFound exception

• createProduct(User user, Product product): check the admin user as already present in database and create product and store in database.

• createUser(User user): create user and store in database for further development.

• getAllProducts(): return all product list from the database.

• getOrderByUser(User user): return all product ordered by specific user from database.

```python
from OrderProcessor import orderprocessor
from DB_Util import DBConnector
from Exception.exception import InvalidDataException
db_connector = DBConnector(host="localhost", user="root", password="root", port="3306", database="OrderManagementSystem")


1 usage
class IOrderManagementRepository():
    def __init__(self,db_connector):
        self.db_connector=db_connector


    1 usage
    def create_order(self):
        userID=input("Enter the User ID : ")
        productID=input("Enter the Product Id of order you want to purchase : ")
        Quantity=input("Quantity you want to purchase : ")
        existing_record=self.get_user_id(userID)
        if existing_record is None:
            self.create_user()

        existing_product=self.get_productID(productID)
        if existing_product is None:
            self.create_product()
        orderprocess.create_order(userID,productID,Quantity)

    1 usage
    def cancel_order(self):
        userID=input("Please Enter the user ID : ")
        orderId=input("Enter the Order ID you want to delete")
        existing_order=self.get_order_id(userID,orderId)
        if existing_order:
            self.delete_order(orderId)
        else:
            raise InvalidDataException("This Order ID not associated with this user ID")

    2 usages
    def create_product(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        productId=self.get_unique_productId()
```

DB_Util.py | Electronics.py | Clothing.py | User.py | IOrderManagementRepository.py | OrderProcessor.py | Product.py | exception.py

```python
        cur = self.db_connector.connection.cursor()
        productId=self.get_unique_productId()
        productname=input("Enter the Product Name : ")
        description=input("Enter the discription of Product : ")
        price=input("Enter the price of Product : ")
        Quantity=int(input("Enter the Quantity in Stock : "))
        type=input("Enter the type of the Product 1. Electronics  2. Clothing : ")
        query="Insert into Product values(%s,%s,%s,%s,%s,%s)"
        values=(productId,productname,description,price,Quantity,type,)
        cur.execute(query,values)
        self.db_connector.connection.commit()

    2 usages
    def create_user(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        userID=self.get_unique_userId()
        username=input("Enter the username : ")
        password=input("Enter the password : ")
        role=input("Enter the Role 1.Admin 2.User")
        query="Insert into User values(%s,%s,%s,%s)"
        values=(userID,username,password,role,)
        cur.execute(query,values)



    1 usage
    def get_all_Products(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("select Productname from Product")
        record = cur.fetchall()
        for i in record:
            print(i[0])

    1 usage
    def get_allProducts(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("select Productname from Product")
        record = cur.fetchall()
```

DB_Util.py | Electronics.py | Clothing.py | User.py | IOrderManagementRepository.py | OrderProcessor.py | Product.py | exception.py

```python
        record = cur.fetchall()
        return record[0]

    1 usage
    def getOrderByUser(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        UserID=input("Enter the user Id For which you want to fetch orders: ")
        query="Select OrderId from Orders Where UserID=%s"
        values=(UserID)
        cur.execute(query,values)
        record=cur.fetchall()
        for i in record:
            print(i[0])


    1 usage
    def get_user_id(self,userID):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select UserID from user Where UserID=%s",(userID,))
        return cur.fetchone()


    1 usage
    def get_productID(self,productID):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select ProductID from Product Where ProductID=%s",(productID,))
        return cur.fetchone()

    def get_quantity(self,productId):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select QuantityInStock From Product Where ProductId =%s",(productId,))
        record=cur.fetchone()
        return record[0]

    1 usage
    def get_order_id(self,UserID,orderID):
        self.db_connector.open_connection()
```

```python
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select orderId from Orders Where userId=%s and orderID=%s",(UserID,orderID,))
        return cur.fetchone()

    1 usage
    def delete_order(self,orderID):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Delete From Orders Where OrderId=%s",(orderID))
        self.db_connector.connection.commit()

    1 usage
    def get_unique_productId(self):
        return len(self.get_allProducts())+1



    1 usage
    def get_unique_userId(self):
        return len(self.get_alluser())+1

    1 usage
    def get_alluser(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("select userName from User")
        record=cur.fetchall()
        return record[0]
```

7. Implement the IOrderManagementRepository interface/abstractclass in a class called OrderProcessor. This class will be responsible for managing orders.

```python
from abc import ABC, abstractmethod

class IOrderManagemntRespository(ABC):
    @abstractmethod
    def create_order(self,user,products):
        pass

    @abstractmethod
    def cancel_order(self,userID,orderId):
        pass

    @abstractmethod
    def create_product(self,admin_user,products):
        pass

    @abstractmethod
    def create_user(self,user):
        pass

    @abstractmethod
    def get_all_products(self):
        pass

    @abstractmethod
    def getOrderByUser(self,user):
        pass


2 usages
class orderprocessor():
    def __init__(self,db_connector):
        self.db_connector=db_connector


    1 usage
    def create_order(self,userID,productID,Quantity):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        orderID=self.get_unique_orderId()
        userID=userID
        productID=productID
```

```python
2 usages
class orderprocessor():
    def __init__(self,db_connector):
        self.db_connector=db_connector


    1 usage
    def create_order(self,userID,productID,Quantity):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        orderID=self.get_unique_orderId()
        userID=userID
        productID=productID
        Quantity=Quantity
        cur.execute("Select Price from Product Where ProductId=%s",(productID,))
        record=cur.fetchone()
        TotalAmount=Quantity*record[0]
        Query="INSERT INTO Orders (orderID, userID, ProductId, TotalAmount,Quantity,OrderDate) VALUES (%s, %s, %s, %s,%s,CURDATE())"
        values=(orderID,userID,productID,TotalAmount,Quantity,)
        cur.execute(Query,values)
    1 usage
    def get_unique_orderId(self):
        return len(self.get_allorders()) + 1

    1 usage
    def get_allorders(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select orderID from Orders")
        record = cur.fetchall()
        return record
```

8. Create DBUtil class and add the following method.

• static getDBConn():Connection Establish a connection to the database and return database Connection

DB_Util.py × Electronics.py Clothing.py User.py IOrderManagementRepository.py OrderProcessor.py Product.py exception.py

```python
import mysql.connector
2 usages
class DBConnector:
    def __init__(self, host, user, password, port, database):
        self.host = host
        self.user = user
        self.password = password
        self.port = port
        self.database = database
        self.connection = None

    13 usages (13 dynamic)
    def open_connection(self):
        if self.connection is None or not self.connection.is_connected():
            self.connection = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                port=self.port,
                database=self.database
            )

    def close_connection(self):
        if self.connection is not None and self.connection.is_connected():
            self.connection.close()
```

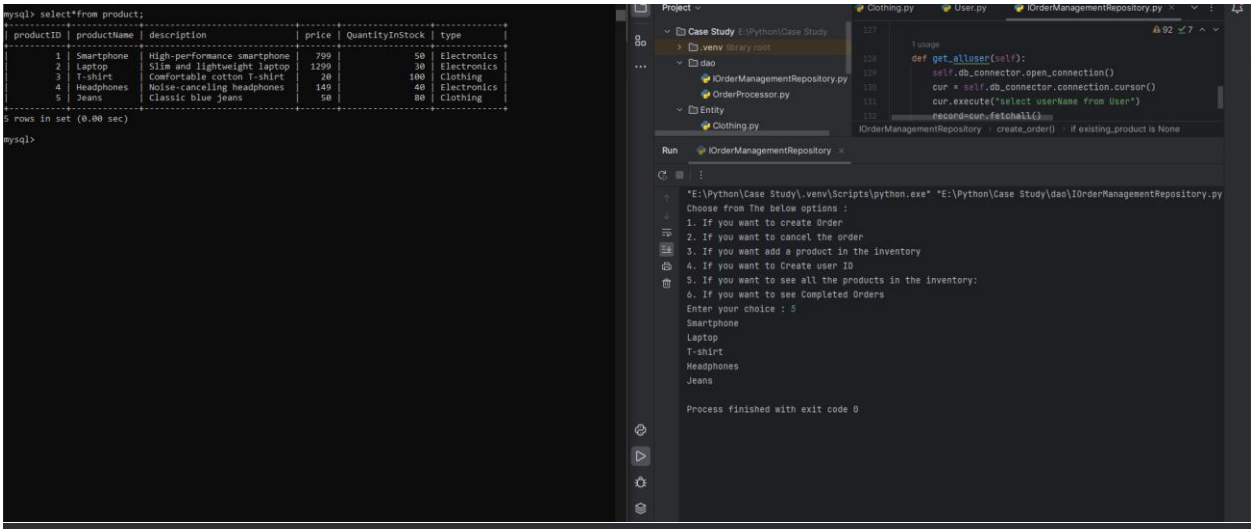
9. Create OrderManagement main class and perform following operation:

• main method to simulate the loan management system. Allow the user to interact with the system by entering choice from menu such as "createUser", "createProduct", "cancelOrder", "getAllProducts", "getOrderbyUser", "exit"

```
1 usage
def main():
    print("Choose from The below options : ")
    print("1. If you want to create Order")
    print("2. If you want to cancel the order")
    print("3. If you want add a product in the inventory")
    print("4. If you want to Create user ID")
    print("5. If you want to see all the products in the inventory:")
    print("6. If you want to see Completed Orders")
    choice = input("Enter your choice : ")
    if choice == '1':
        order_respositoy.create_order()
    elif choice == '2':
        order_respositoy.cancel_order()
    elif choice == '3':
        order_respositoy.create_product()
    elif choice == '4':
        order_respositoy.create_user()
    elif choice == '5':
        order_respositoy.get_all_Products()
    elif choice == '6':
        order_respositoy.getOrderByUser()
    else:
        print("Enter correct choice")
order_respositoy=IOrderManagementRepository(db_connector)
orderprocess = orderprocessor(db_connector)

main()
```

OutPut:

```
"E:\Python\Case Study\.venv\Scripts\python.exe" "E:\Python\Case Study\dao\IOrderManagementRepository.py"
Choose from The below options :
1. If you want to create Order
2. If you want to cancel the order
3. If you want add a product in the inventory
4. If you want to Create user ID
5. If you want to see all the products in the inventory:
6. If you want to see Completed Orders
Enter your choice :
```




```
"E:\Python\Case Study\.venv\Scripts\python.exe" "E:\Python\Case Study\dao\IOrderManagementRepository.py"
Choose from The below options :
1. If you want to create Order
2. If you want to cancel the order
3. If you want add a product in the inventory
4. If you want to Create user ID
5. If you want to see all the products in the inventory:
6. If you want to see Completed Orders
Enter your choice : 3
Enter the Product Name : SmartWatch
Enter the discription of Product : Accessories
Enter the price of Product : 950
Enter the Quantity in Stock : 50
Enter the type of the Product 1. Electronics  2. Clothing : Electronics

Process finished with exit code 0
```

```
mysql> select*from product;
+-----------+-------------+----------------------------+-------+-----------------+-------------+
| productID | productName | description                | price | QuantityInStock | type        |
+-----------+-------------+----------------------------+-------+-----------------+-------------+
|         1 | Smartphone  | High-performance smartphone |   799 |              50 | Electronics |
|         2 | Laptop      | Slim and lightweight laptop |  1299 |              30 | Electronics |
|         3 | T-shirt     | Comfortable cotton T-shirt  |    20 |             100 | Clothing    |
|         4 | Headphones  | Noise-canceling headphones  |   149 |              40 | Electronics |
|         5 | Jeans       | Classic blue jeans          |    50 |              80 | Clothing    |
+-----------+-------------+----------------------------+-------+-----------------+-------------+
5 rows in set (0.00 sec)

mysql> select*from product;
+-----------+-------------+----------------------------+-------+-----------------+-------------+
| productID | productName | description                | price | QuantityInStock | type        |
+-----------+-------------+----------------------------+-------+-----------------+-------------+
|         1 | Smartphone  | High-performance smartphone |   799 |              50 | Electronics |
|         2 | Laptop      | Slim and lightweight laptop |  1299 |              30 | Electronics |
|         3 | T-shirt     | Comfortable cotton T-shirt  |    20 |             100 | Clothing    |
|         4 | Headphones  | Noise-canceling headphones  |   149 |              40 | Electronics |
|         5 | Jeans       | Classic blue jeans          |    50 |              80 | Clothing    |
|         6 | SmartWatch  | Accessories                 |   950 |              50 | Electronics |
+-----------+-------------+----------------------------+-------+-----------------+-------------+
6 rows in set (0.00 sec)
```

```
"E:\Python\Case Study\.venv\Scripts\python.exe" "E:\Python\Case Study\dao\IOrderManagementRepository.py"
Choose from The below options :
1. If you want to create Order
2. If you want to cancel the order
3. If you want add a product in the inventory
4. If you want to Create user ID
5. If you want to see all the products in the inventory:
6. If you want to see Completed Orders
Enter your choice : 4
Enter the username : manas
Enter the password : manas
Enter the Role 1.Admin 2.UserAdmin

Process finished with exit code 0
```

```
mysql> select*from user;
+--------+----------+---------------+-------+
| userID | userName | password      | role  |
+--------+----------+---------------+-------+
|      1 | admin1   | admin_password | Admin |
|      2 | user1    | user_password1 | User  |
|      3 | user2    | user_password2 | User  |
|      4 | user3    | user_password3 | User  |
|      5 | user4    | user_password4 | User  |
+--------+----------+---------------+-------+
5 rows in set (0.00 sec)

mysql> select*from user;
+--------+----------+---------------+-------+
| userID | userName | password      | role  |
+--------+----------+---------------+-------+
|      1 | admin1   | admin_password | Admin |
|      2 | user1    | user_password1 | User  |
|      3 | user2    | user_password2 | User  |
|      4 | user3    | user_password3 | User  |
|      5 | user4    | user_password4 | User  |
|      6 | manas    | manas          | User  |
+--------+----------+---------------+-------+
6 rows in set (0.00 sec)

mysql>
```

```
Project
Case Study E:\Python\Case Study
.venv library root
dao
IOrderManagementRepository.py
OrderProcessor.py
Entity
Clothing.py

Clothing.py   User.py   IOrderManagementRepository.py

56        values=(userID,username,password,role,)
57        cur.execute(query,values)
58        self.db_connector.connection.commit()

IOrderManagementRepository > create_user()

Run   IOrderManagementRepository

"E:\Python\Case Study\.venv\Scripts\python.exe" "E:\Python\Case Study\dao\IOrderManagementRepository.py"
Choose from The below options :
1. If you want to create Order
2. If you want to cancel the order
3. If you want add a product in the inventory
4. If you want to Create user ID
5. If you want to see all the products in the inventory:
6. If you want to see Completed Orders
Enter your choice : 4
Enter the username : manas
Enter the password : manas
Enter the Role 1.Admin 2.User : User

Process finished with exit code 0
```

```
"E:\Python\Case Study\.venv\Scripts\python.exe" "E:\Python\Case Study\dao\IOrderManagementRepository.py"
Choose from The below options :
1. If you want to create Order
2. If you want to cancel the order
3. If you want add a product in the inventory
4. If you want to Create user ID
5. If you want to see all the products in the inventory:
6. If you want to see Completed Orders
Enter your choice : 1
Enter the User ID : 2
Enter the Product Id of order you want to purchase : 2
Quantity you want to purchase : 2
```

```
mysql> select* from orders
    -> ;
+---------+--------+-----------+----------+-------------+------------+
| OrderID | userID | productID | Quantity | TotalAmount | OrderDate  |
+---------+--------+-----------+----------+-------------+------------+
|       1 |      2 |         3 |        4 |          80 | 2024-02-06 |
+---------+--------+-----------+----------+-------------+------------+
1 row in set (0.00 sec)
```