Order Management System

Tarun Tamang

Entity

Product Class:- Sub Class Electronics and Clothing

```
from util.DBconnection import DBConnection


class Product(DBConnection):
    def __init__(self):
        super().__init__()
        self.productId = 0
        self.productName = ''
        self.description = ''
        self.price = 0.0
        self.quantityInStock = 0
        self.type = ''

    def get_product_id(self):
        return self.productId

    def set_product_id(self,product_id):
        self.productId=product_id

    def get_product_name(self):
        return self.productName

    def set_product_name(self, productName):
        self.productName = productName

    def get_description(self):
        return self.description

    def set_description(self, description):
        self.description = description

    def get_price(self):
        return self.price

    def set_price(self, price):
        self.price = price

    def get_quantityInStock(self):
        return self.quantityInStock

    def set_quantityInStock(self, quantityInStock):
        self.quantityInstock = quantityInStock

    def get_type(self):
        return self.type

    def set_type(self, type):
        self.type = type


class Electronics(Product):
    def __init__(self):
        super().__init__()
        self.brand = ''
        self.warrantyPeriod = 0


    def get_brand(self):
        return self.brand

    def set_brand(self,brand):
        self.brand=brand

    def get_warrantyPeriod(self):
        return self.warrantyPeriod

    def set_warrantyPeriod(self,warranty):
        self.warrantyPeriod=warranty


class Clothing(Product):
    def __init__(self):
        super().__init__()
        self.size = ''
        self.color = ''


    def get_size(self):
        return self.size

    def set_size(self,size):
        self.size=size

    def get_color(self):
        return self.color

    def set_color(self,color):
        self.color=color
```

User Class

```
from util.DBconnection import DBConnection

class User(DBConnection):
    def __init__(self):
        super().__init__()
        self.userId = 0
        self.username = ''
        self.password = ''
        self.role = ''

    def get_userId(self):
        return self.userId

    def set_userId(self,userId):
        self.userId=userId

    def get_username(self):
        return self.username

    def set_username(self, username):
        self.username = username

    def get_password(self):
        return self.password

    def set_password(self, password):
        self.password = password

    def get_role(self):
        return self.role

    def set_role(self, role):
        self.role = role
```

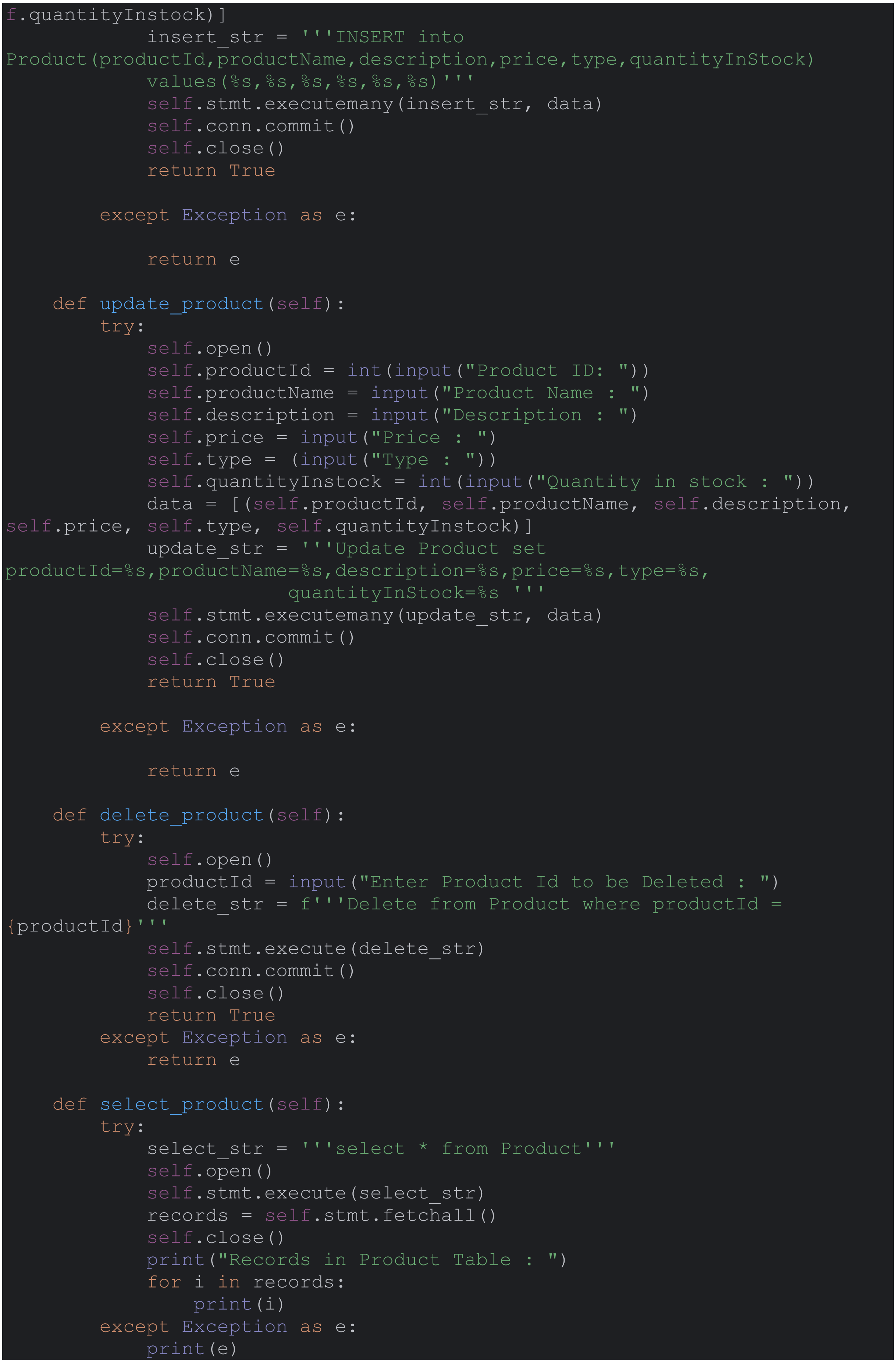
Orders Class

```
from entity.Product import Product
from entity.User import User


class Orders(Product,User):
    def __init__(self):
        super().__init__()

        self.orderId = 0

    def get_order_id(self):
        return self.orderId

    def set_order_id(self,orderId):
        self.orderId=orderId
```

Dao

ProductDao

```
from entity.Product import Product

class ProductDao( Product):
    def __init__(self):
        super().__init__()


    def perform_product_actions(self):
        while True:
            print("(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_product_table()
            elif ch==2:
                print(self.add_product())
            elif ch==3:
                print(self.update_product())
            elif ch==4:
                print(self.delete_product())
            elif ch==5:
                self.select_product()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_product_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Product(
            productId int primary key,
            productName varchar(50) not null,
            description varchar(50),
            price double(10,2) not null ,
            type enum('Electronics','Clothing'),
            quantityInStock int not null
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Product Table created Successfully")
        except Exception as e:
            print(e)


    def add_product(self):
        try:
            self.open()
            self.productId=int(input("Product ID: "))
            self.productName=input("Product Name : ")
            self.description = input("Description : ")
            self.price = input("Price : ")
            self.type = input("Type : ")
            self.quantityInstock = int(input("Quantity in stock : "))
            data =
[(self.productId,self.productName,self.description,self.price,self.type,sel
```



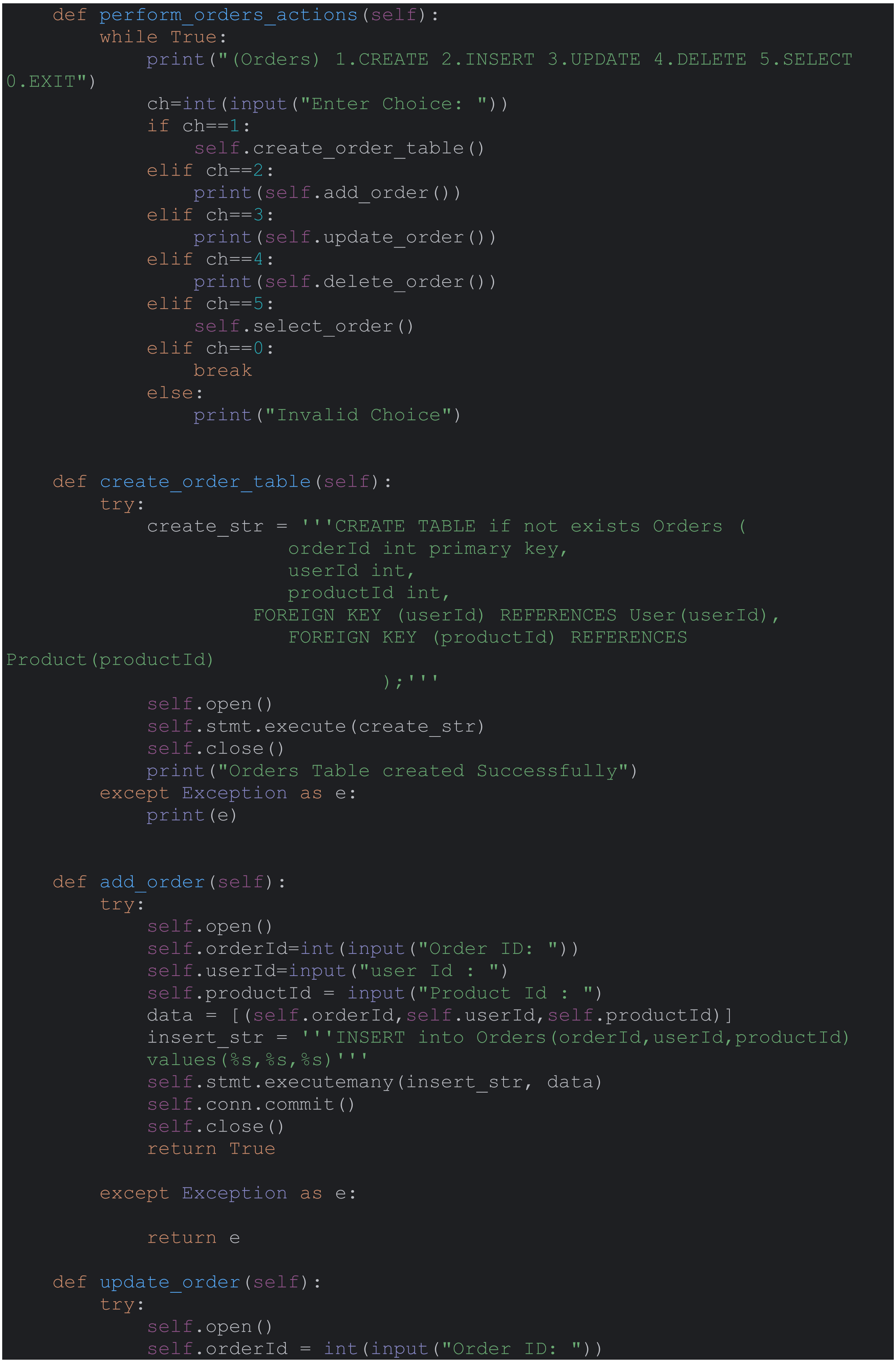
```
f.quantityInstock)]
            insert_str = '''INSERT into 
Product(productId,productName,description,price,type,quantityInStock) 
            values(%s,%s,%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_product(self):
        try:
            self.open()
            self.productId = int(input("Product ID: "))
            self.productName = input("Product Name : ")
            self.description = input("Description : ")
            self.price = input("Price : ")
            self.type = (input("Type : "))
            self.quantityInstock = int(input("Quantity in stock : "))
            data = [(self.productId, self.productName, self.description, 
self.price, self.type, self.quantityInstock)]
            update_str = '''Update Product set 
productId=%s,productName=%s,description=%s,price=%s,type=%s,
                        quantityInStock=%s '''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_product(self):
        try:
            self.open()
            productId = input("Enter Product Id to be Deleted : ")
            delete_str = f'''Delete from Product where productId = 
{productId}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_product(self):
        try:
            select_str = '''select * from Product'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Product Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

UserDao

```
from entity.User import User

class UserDao( User):
    def __init__(self):
        super().__init__()


    def perform_user_actions(self):
        while True:
            print("(Users) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_user_table()
            elif ch==2:
                print(self.add_user())
            elif ch==3:
                print(self.update_user())
            elif ch==4:
                print(self.delete_user())
            elif ch==5:
                self.select_user()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_user_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS User(
            userId int primary key,
            username varchar(50) not null,
            password varchar(50) not null
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("User Table created Successfully")
        except Exception as e:
            print(e)


    def add_user(self):
        try:
            self.open()
            self.userId=int(input("User ID: "))
            self.username=input("User Name : ")
            self.password = input("Password : ")
            data = [(self.userId,self.username,self.password)]
            insert_str = '''INSERT into User(userId,username,password)
            values(%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:

            return e

    def update_user(self):
        try:
            self.open()
            self.userId = int(input("User ID: "))
            self.username = input("User Name : ")
            self.password = input("Password : ")
            data = [(self.userId, self.username, self.password)]
            update_str = '''Update User set userId=%s,
username=%s,password=%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_user(self):
        try:
            self.open()
            userId = input("Enter User Id to be Deleted : ")
            delete_str = f'''Delete from user where userId = {userId}'''
            (self.stmt.execute
             (delete_str))
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_user(self):
        try:
            select_str = '''select * from User'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in User Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrdersDao

```
from entity.Orders import Orders

class OrdersDao( Orders):
    def __init__(self):
        super().__init__()
```



```
    def perform_orders_actions(self):
        while True:
            print("(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_order_table()
            elif ch==2:
                print(self.add_order())
            elif ch==3:
                print(self.update_order())
            elif ch==4:
                print(self.delete_order())
            elif ch==5:
                self.select_order()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_order_table(self):
        try:
            create_str = '''CREATE TABLE if not exists Orders (
                        orderId int primary key,
                        userId int,
                        productId int,
                     FOREIGN KEY (userId) REFERENCES User(userId),
                        FOREIGN KEY (productId) REFERENCES
Product(productId)
                                );'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Orders Table created Successfully")
        except Exception as e:
            print(e)


    def add_order(self):
        try:
            self.open()
            self.orderId=int(input("Order ID: "))
            self.userId=input("user Id : ")
            self.productId = input("Product Id : ")
            data = [(self.orderId,self.userId,self.productId)]
            insert_str = '''INSERT into Orders(orderId,userId,productId)
            values(%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_order(self):
        try:
            self.open()
            self.orderId = int(input("Order ID: "))
            self.userId = input("user Id : ")
            self.productId = input("Product Id : ")
            data = [(self.orderId, self.userId, self.productId)]
            update_str = '''Update Orders set
orderId=%s,userId=%s,productId=%%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_order(self):
        try:
            self.open()
            order_id = input("Enter Order Id to be Deleted : ")
            delete_str = f'''Delete from Orders where orderId =
{order_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_order(self):
        try:
            select_str = '''select * from Orders'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Orders Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrderProcessor

```
from dao.IOrderManagementRepository import IOrderManagementRepository
from dao.OrdersDao import OrdersDao
from dao.ProductDao import ProductDao
from dao.UserDao import UserDao
from exception.OrderNotFoundException import OrderNotFoundException


class OrderProcessor(IOrderManagementRepository):
    def createOrder(self):
        o = OrdersDao()
        o.add_order()

        pass
    def createProduct(self):
        p=ProductDao()
        p.add_product()
        pass

    def createUser(self):
        u=UserDao()
        u.add_user()
        pass

    def getAllProducts(self):
        p=ProductDao()
        p.select_product()
        pass

    def getOrderByUser(self,orderId):
        try:
            self.open()
            self.stmt.execute(f'''SELECT * FROM Orders WHERE orderId =
{orderId}''')
            records = self.stmt.fetchall()
            self.close()
            return records
        except OrderNotFoundException as e:
            return e


        pass
```

IOrderManagementRepository

```
from abc import ABC,abstractmethod
from util.DBconnection import DBConnection
from entity.Orders import Orders


class IOrderManagementRepository(ABC,DBConnection):
    @abstractmethod
    def createOrder(self):
        pass

    @abstractmethod
    def createProduct(self):
        pass

    @abstractmethod
    def createUser(self):
        pass

    @abstractmethod
    def getAllProducts(self):
        pass

    @abstractmethod
    def getOrderByUser(self,orderId):
        pass
```

ElectronicsDao

```
from entity.Product import Electronics

class ElectronicsDao( Electronics):
    def __init__(self):
        super().__init__()


    def perform_electronic_actions(self):
        while True:
            print("(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_electronic_table()
            elif ch==2:
                print(self.add_product())
            elif ch==3:
                print(self.update_product())
            elif ch==4:
                print(self.delete_product())
            elif ch==5:
                self.select_product()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_electronic_table(self):
        try:
            create_str = '''CREATE TABLE if not exists Electronics (
                        productId INT PRIMARY KEY,
                        brand VARCHAR(255),
                        warrantyPeriod INT,
                        FOREIGN KEY (productId) REFERENCES 
Product(productId)
                        );'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Electronics Table created Successfully")
        except Exception as e:
            print(e)


    def add_product(self):
        try:
            self.open()
            self.productId=int(input("Product ID: "))
            self.brand=input("brand : ")
            self.warrantyPeriod = int(input(" Warranty Period: "))
            data = [(self.productId,self.brand,self.warrantyPeriod)]
            insert_str = '''INSERT into 
Electronics(productId,brand,warrantyPeriod)
            values(%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:

            return e

    def update_product(self):
        try:
            self.open()
            self.productId = int(input("Product ID: "))
            self.brand = input("Brand: ")
            self.warrantyPeriod = int(input("WarrantyPeriod : "))
            data = [(self.productId, self.brand, self.warrantyPeriod)]
            update_str = '''Update Electronics set
productId=%s,brand=%s,warrantyPeriod=%%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_product(self):
        try:
            self.open()
            product_id = input("Enter Product Id to be Deleted : ")
            delete_str = f'''Delete from Electronics where order_id =
{product_id}'''
            self.stmt.executemany(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_product(self):
        try:
            select_str = '''select * from Electronics'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Electronics Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

ClothingDao

```
from entity.Product import Clothing

class ClothingDao( Clothing):
    def __init__(self):
        super().__init__()


    def perform_clothing_actions(self):
        while True:
            print("(Clothing) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_clothing_table()
            elif ch==2:
                print(self.add_product())
            elif ch==3:
                print(self.update_product())
            elif ch==4:
                print(self.delete_product())
            elif ch==5:
                self.select_product()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_clothing_table(self):
        try:
            create_str = '''CREATE TABLE if not exists Clothing (
                        productId INT PRIMARY KEY,
                        color VARCHAR(255),
                        size varchar(255),
                        FOREIGN KEY (productId) REFERENCES
Product(productId)
                        );'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Electronics Table created Successfully")
        except Exception as e:
            print(e)


    def add_product(self):
        try:
            self.open()
            self.productId=int(input("Product ID: "))
            self.color=input("color : ")
            self.size = int(input(" size: "))
            data = [(self.productId,self.color,self.size)]
            insert_str = '''INSERT into Clothing(productId,color,size)
            values(%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_product(self):
        try:
            self.open()
            self.productId = int(input("Product ID: "))
            self.color = input("color: ")
            self.size = int(input("size : "))
            data = [(self.productId, self.color, self.size)]
            update_str = '''Update Clothing set 
productId=%s,color=%s,size=%%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_product(self):
        try:
            self.open()
            product_id = input("Enter Product Id to be Deleted : ")
            delete_str = f'''Delete from Clothing where order_id = 
{product_id}'''
            self.stmt.executemany(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_product(self):
        try:
            select_str = '''select * from Clothing'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Clothing Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

Exceptions

```
class OrderNotFoundException(Exception):
    def __init__(self,orderId):
        super().__init__(f"Order ID : {orderId} not found in the system..")
```

```
class UserNotFoundException(Exception):
    def __init__(self,userId):
        super().__init__(f"user ID : {userId} not found in the system..")
```

Util

DBConnection

```
import sys
import mysql.connector as sql
from util.DButil import DBUtil


class DBConnection:
    def open(self):
        try:
            connection_properties=DBUtil.getDBConn()
            self.conn=sql.connect(**connection_properties)
            self.stmt=self.conn.cursor()
        except Exception as e:
            print(str(e) + 'Database not Connected:')
            sys.exit(1)

    def close(self):
        self.conn.close()
```

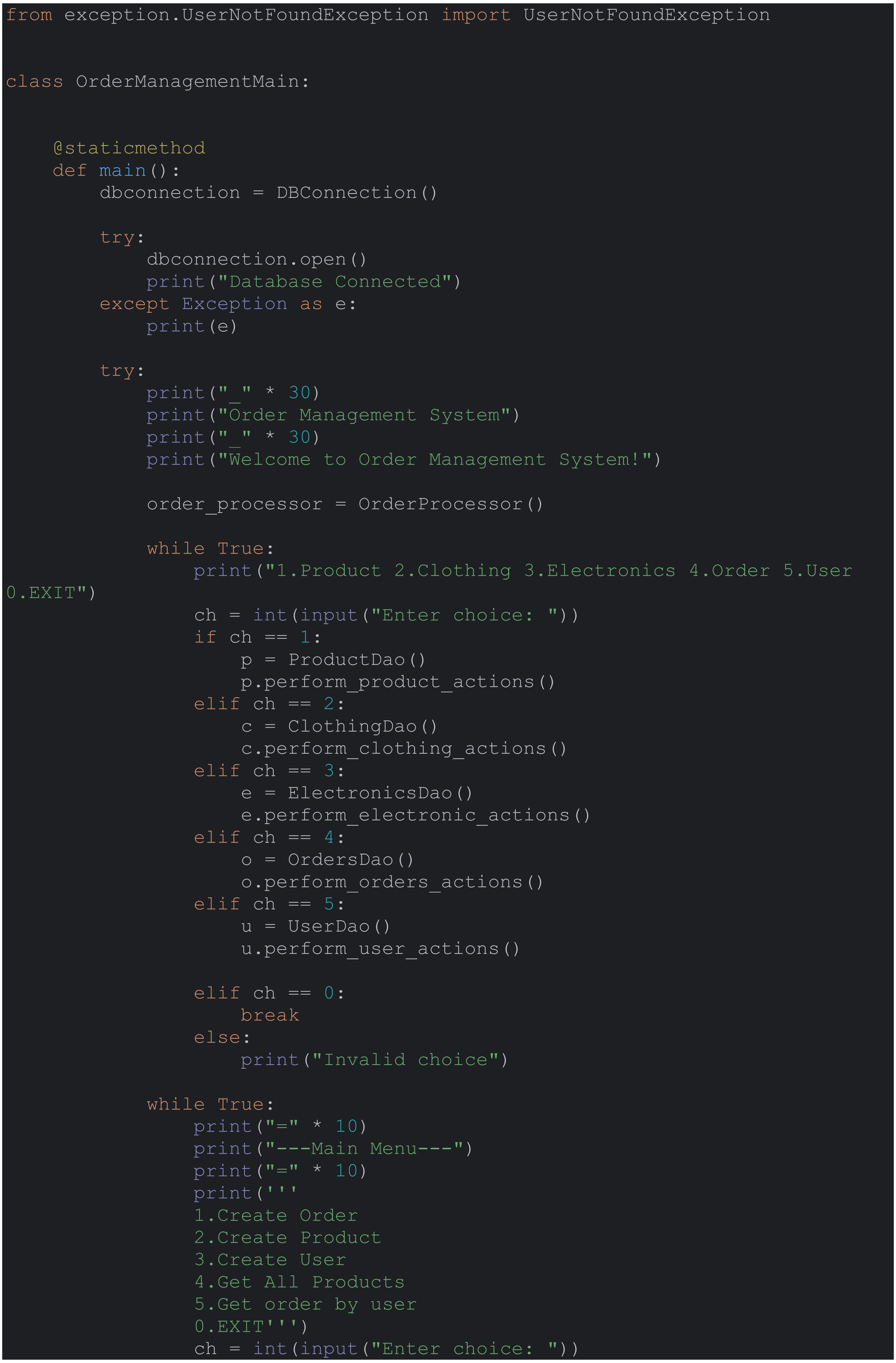
DButil

```
class DBUtil:
    connection_properties = None

    @staticmethod
    def getDBConn():
        if DBUtil.connection_properties is None:
            host = 'localhost'
            database = 'OrderManagementSystem'
            port = '3306'
            user = 'root'
            password = 'root'
            DBUtil.connection_properties = {'host': host,'database':
database,'port':port,'user': user,'password': password}
        return DBUtil.connection_properties
```

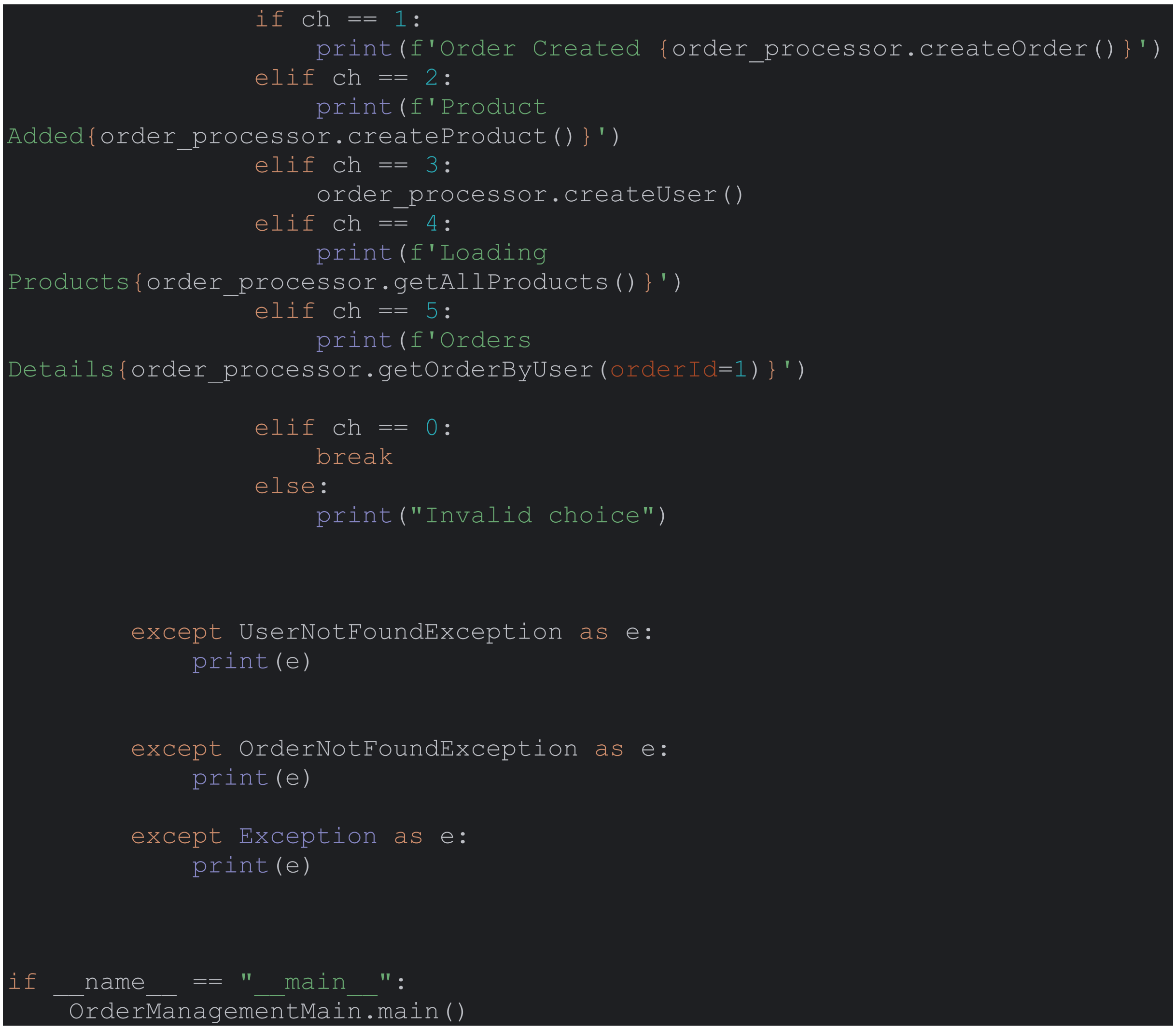
Main

OrderManagementMain

```
from util.DBconnection import DBConnection
from dao.OrderProcessor import OrderProcessor
from dao.UserDao import UserDao
from dao.OrdersDao import OrdersDao
from dao.ProductDao import ProductDao
from dao.ClothingDao import ClothingDao
from dao.ElectronicsDao import ElectronicsDao
from exception.OrderNotFoundException import OrderNotFoundException
```

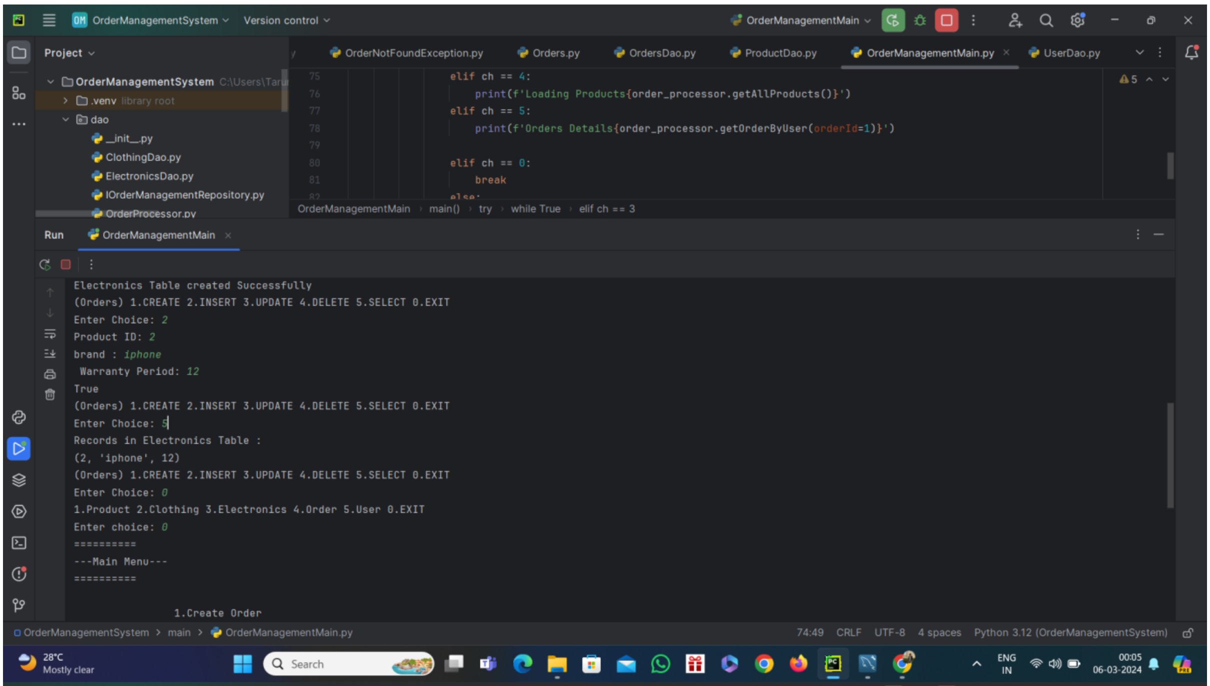


```
from exception.UserNotFoundException import UserNotFoundException


class OrderManagementMain:


    @staticmethod
    def main():
        dbconnection = DBConnection()

        try:
            dbconnection.open()
            print("Database Connected")
        except Exception as e:
            print(e)

        try:
            print("_" * 30)
            print("Order Management System")
            print("_" * 30)
            print("Welcome to Order Management System!")

            order_processor = OrderProcessor()

            while True:
                print("1.Product 2.Clothing 3.Electronics 4.Order 5.User
0.EXIT")
                ch = int(input("Enter choice: "))
                if ch == 1:
                    p = ProductDao()
                    p.perform_product_actions()
                elif ch == 2:
                    c = ClothingDao()
                    c.perform_clothing_actions()
                elif ch == 3:
                    e = ElectronicsDao()
                    e.perform_electronic_actions()
                elif ch == 4:
                    o = OrdersDao()
                    o.perform_orders_actions()
                elif ch == 5:
                    u = UserDao()
                    u.perform_user_actions()

                elif ch == 0:
                    break
                else:
                    print("Invalid choice")

            while True:
                print("=" * 10)
                print("---Main Menu---")
                print("=" * 10)
                print('''
                1.Create Order
                2.Create Product
                3.Create User
                4.Get All Products
                5.Get order by user
                0.EXIT''')
                ch = int(input("Enter choice: "))
```



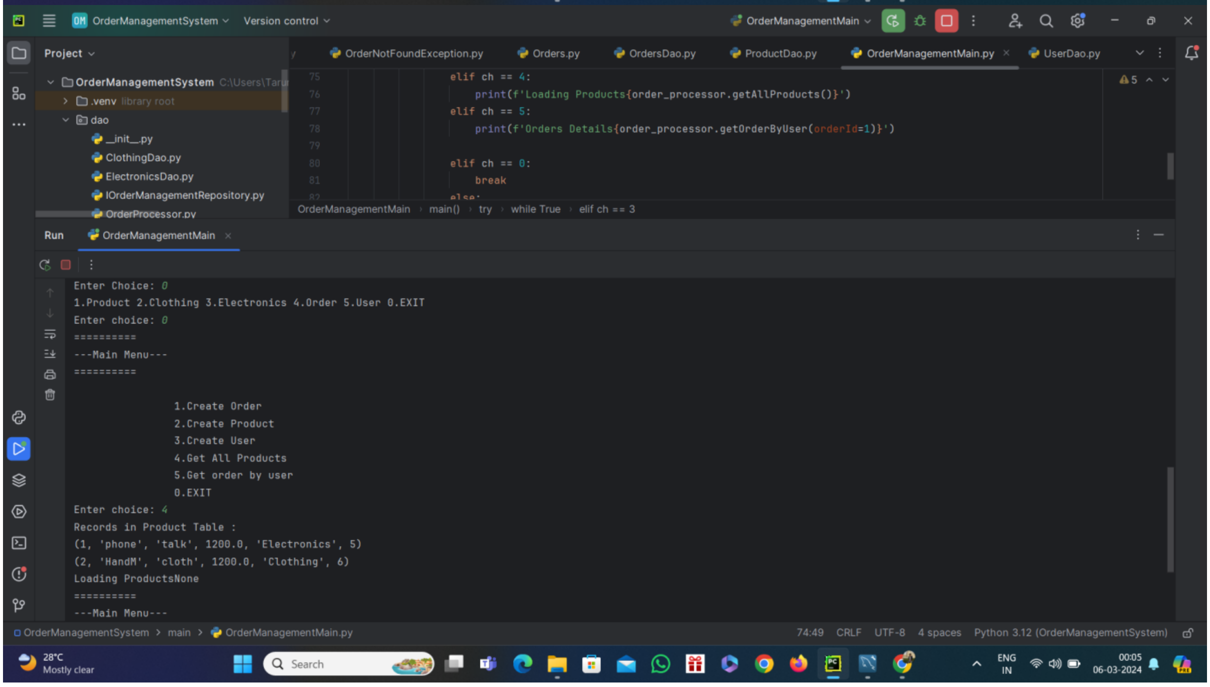
```
                if ch == 1:
                    print(f'Order Created {order_processor.createOrder()}')
                elif ch == 2:
                    print(f'Product 
Added{order_processor.createProduct()}')
                elif ch == 3:
                    order_processor.createUser()
                elif ch == 4:
                    print(f'Loading 
Products{order_processor.getAllProducts()}')
                elif ch == 5:
                    print(f'Orders 
Details{order_processor.getOrderByUser(orderId=1)}')

                elif ch == 0:
                    break
                else:
                    print("Invalid choice")



        except UserNotFoundException as e:
            print(e)


        except OrderNotFoundException as e:
            print(e)

        except Exception as e:
            print(e)



if __name__ == "__main__":
    OrderManagementMain.main()
```

SS



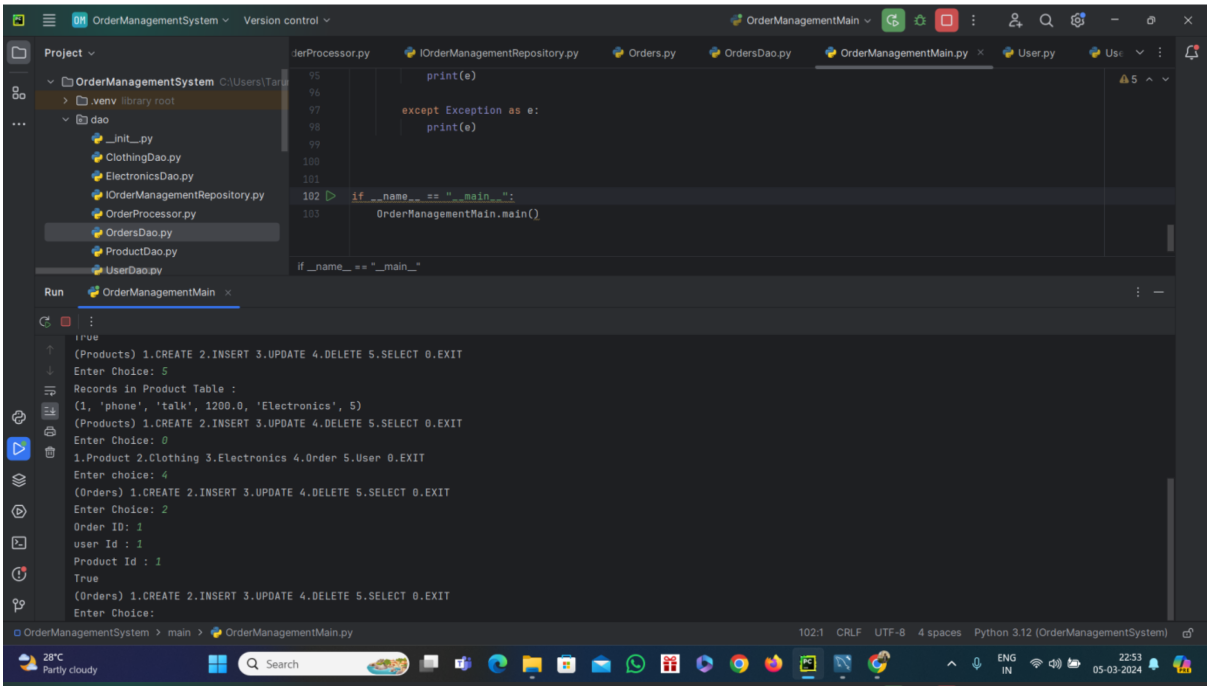
```
            elif ch == 4:
                print(f'Loading Products{order_processor.getAllProducts()}')
            elif ch == 5:
                print(f'Orders Details{order_processor.getOrderByUser(orderId=1)}')

            elif ch == 0:
                break
            else:
```

```
Electronics Table created Successfully
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Product ID: 2
brand : iphone
 Warranty Period: 12
True
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Electronics Table :
(2, 'iphone', 12)
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 0
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 0
==========
---Main Menu---
==========

                1.Create Order
```



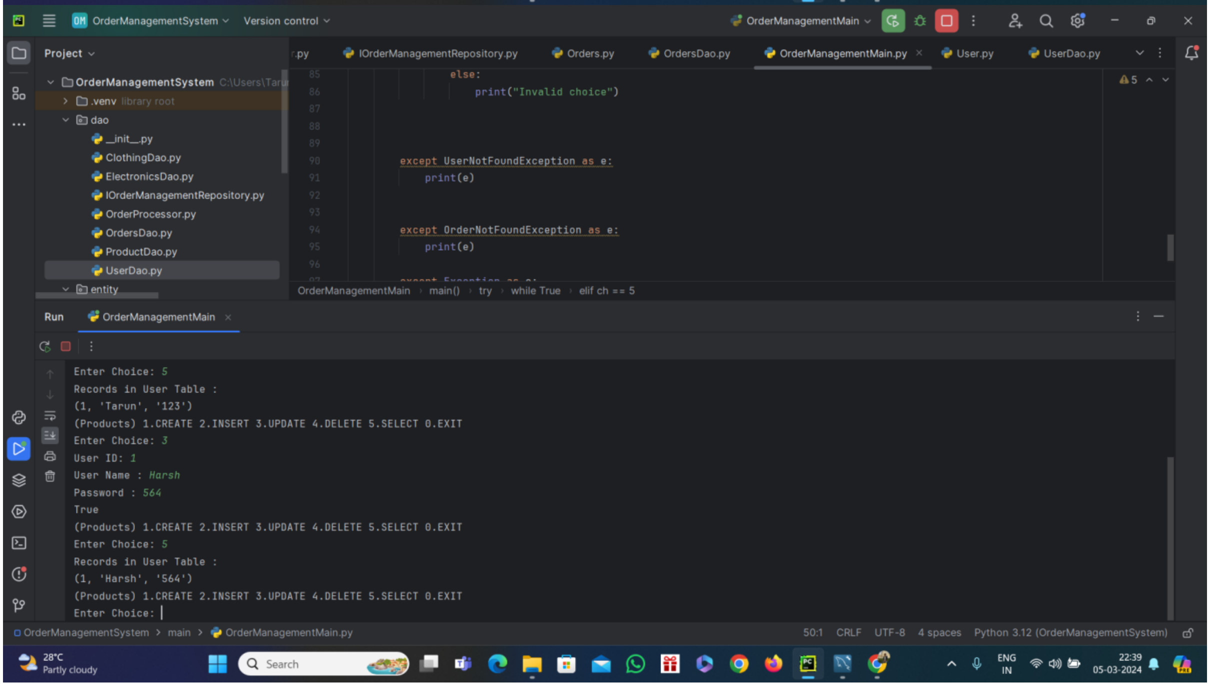
```
            elif ch == 4:
                print(f'Loading Products{order_processor.getAllProducts()}')
            elif ch == 5:
                print(f'Orders Details{order_processor.getOrderByUser(orderId=1)}')

            elif ch == 0:
                break
            else:
```

```
Enter Choice: 0
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 0
==========
---Main Menu---
==========

                1.Create Order
                2.Create Product
                3.Create User
                4.Get All Products
                5.Get order by user
                0.EXIT
Enter choice: 4
Records in Product Table :
(1, 'phone', 'talk', 1200.0, 'Electronics', 5)
(2, 'HandM', 'cloth', 1200.0, 'Clothing', 6)
Loading ProductsNone
==========
---Main Menu---
```



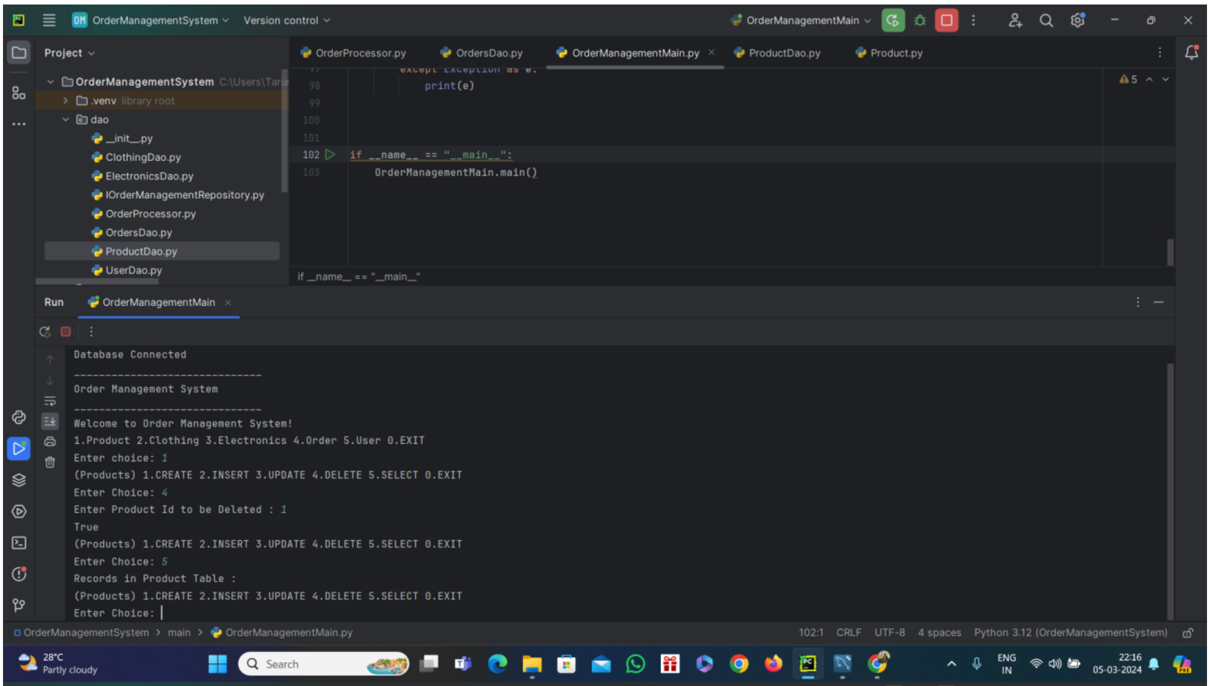
```
        print(e)

    except Exception as e:
        print(e)



if __name__ == "__main__":
    OrderManagementMain.main()
```

```
True
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Product Table :
(1, 'phone', 'talk', 1200.0, 'Electronics', 5)
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 0
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 4
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Order ID: 1
user Id : 1
Product Id : 1
True
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```



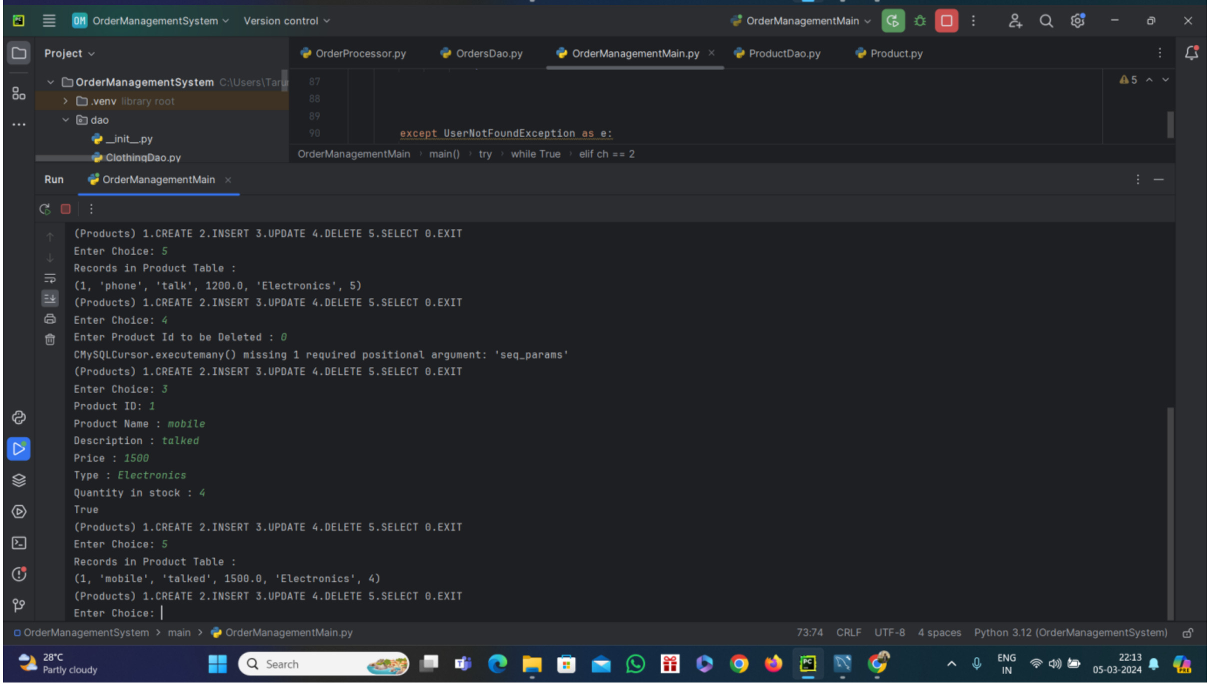
```
            else:
                print("Invalid choice")




    except UserNotFoundException as e:
        print(e)


    except OrderNotFoundException as e:
        print(e)

    except Exception as e:
```

```
Enter Choice: 5
Records in User Table :
(1, 'Tarun', '123')
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 3
User ID: 1
User Name : Harsh
Password : 564
True
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in User Table :
(1, 'Harsh', '564')
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```



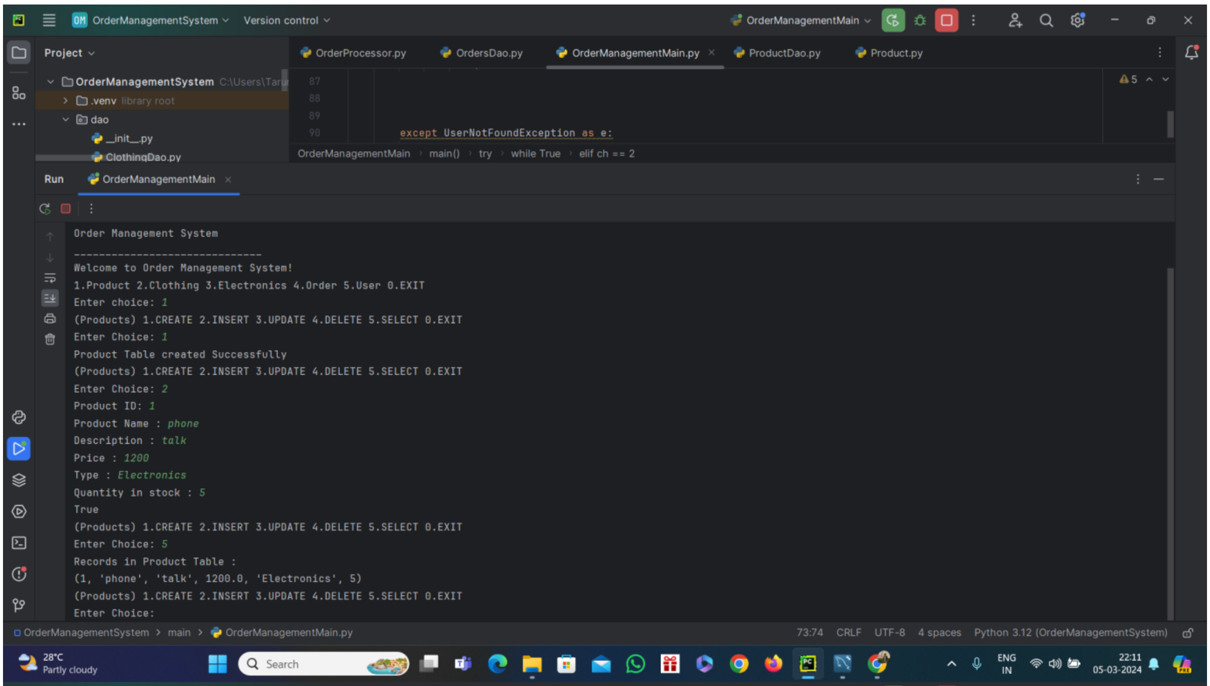
```
        except Exception as e:
            print(e)



if __name__ == "__main__":
    OrderManagementMain.main()
```

```
Database Connected
-----------------------------
Order Management System
-----------------------------
Welcome to Order Management System!
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 1
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 4
Enter Product Id to be Deleted : 1
True
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Product Table :
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 
```



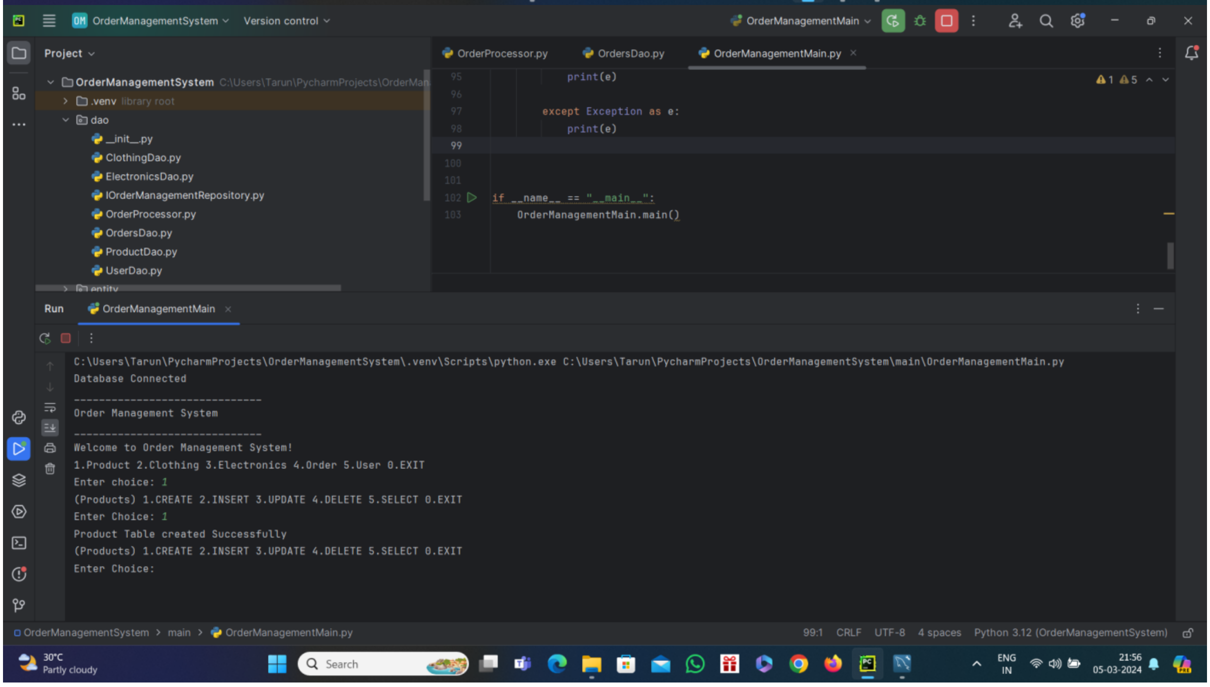
```



        except UserNotFoundException as e:
```

```
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Product Table :
(1, 'phone', 'talk', 1200.0, 'Electronics', 5)
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 4
Enter Product Id to be Deleted : 0
CMySQLCursor.executemany() missing 1 required positional argument: 'seq_params'
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 3
Product ID: 1
Product Name : mobile
Description : talked
Price : 1500
Type : Electronics
Quantity in stock : 4
True
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Product Table :
(1, 'mobile', 'talked', 1500.0, 'Electronics', 4)
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 
```



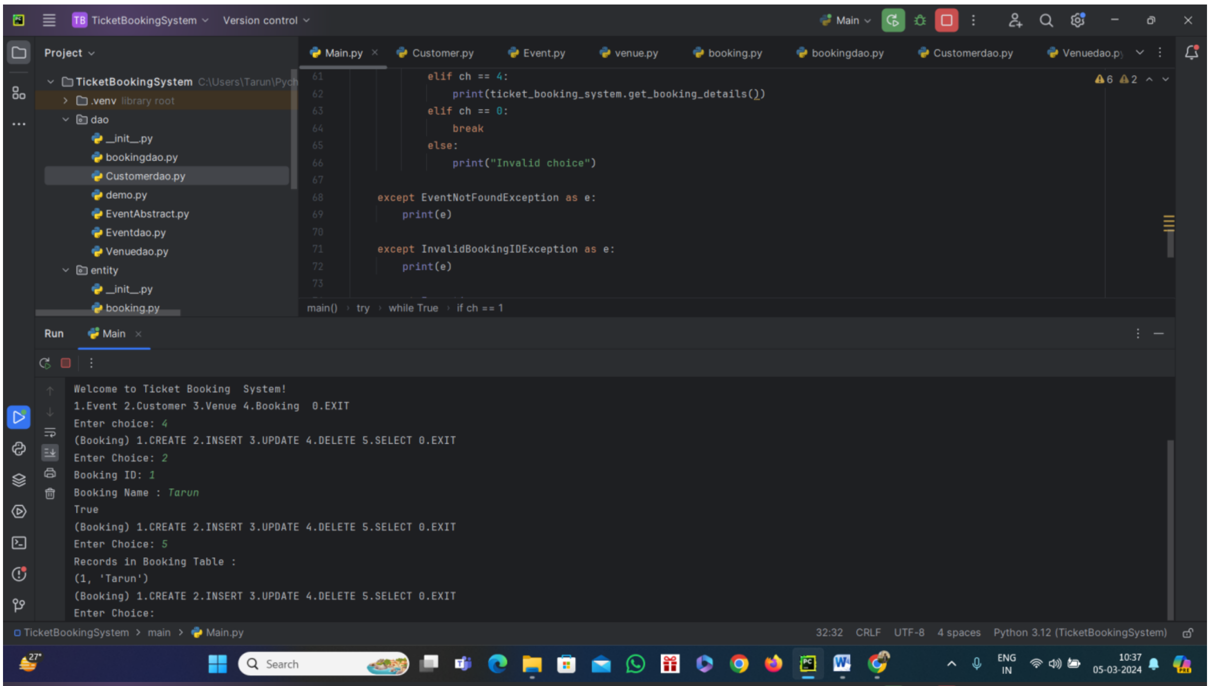
```
except UserNotFoundException as e:
```

```
Order Management System

------------------------------
Welcome to Order Management System!
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 1
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 1
Product Table created Successfully
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Product ID: 1
Product Name : phone
Description : talk
Price : 1200
Type : Electronics
Quantity in stock : 5
True
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Product Table :
(1, 'phone', 'talk', 1200.0, 'Electronics', 5)
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```



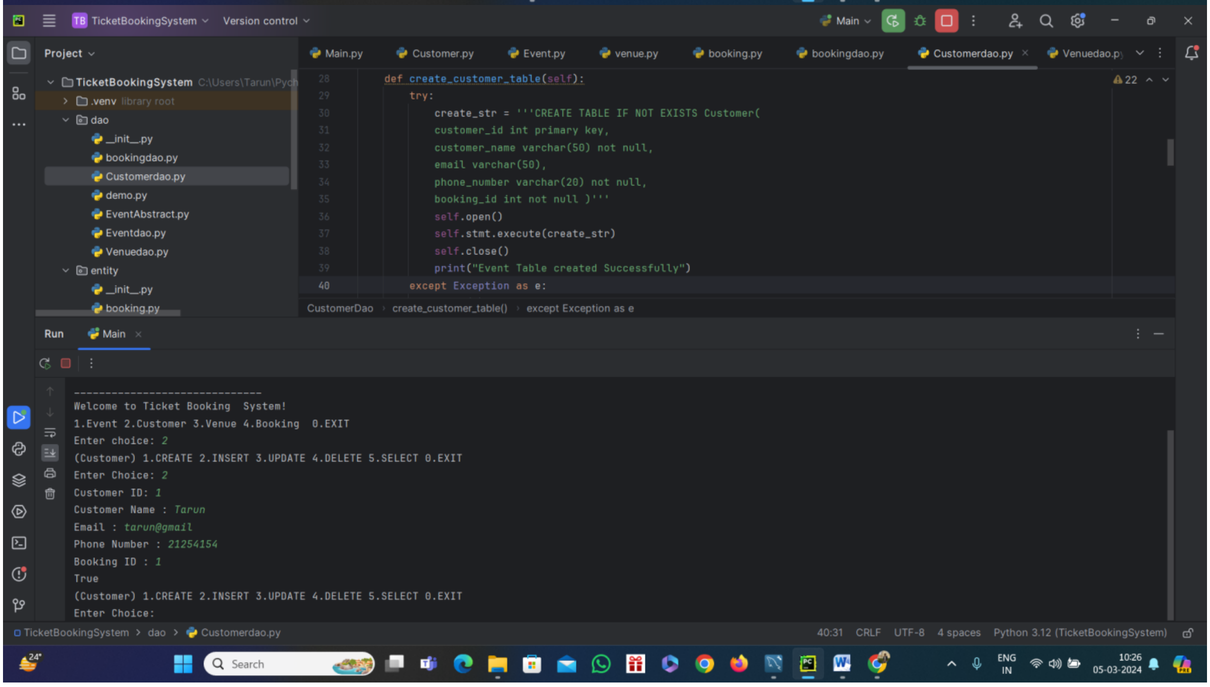
```
            print(e)

        except Exception as e:
            print(e)



if __name__ == "__main__":
    OrderManagementMain.main()
```

```
C:\Users\Tarun\PycharmProjects\OrderManagementSystem\.venv\Scripts\python.exe C:\Users\Tarun\PycharmProjects\OrderManagementSystem\main\OrderManagementMain.py
Database Connected

------------------------------
Order Management System

------------------------------
Welcome to Order Management System!
1.Product 2.Clothing 3.Electronics 4.Order 5.User 0.EXIT
Enter choice: 1
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 1
Product Table created Successfully
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```



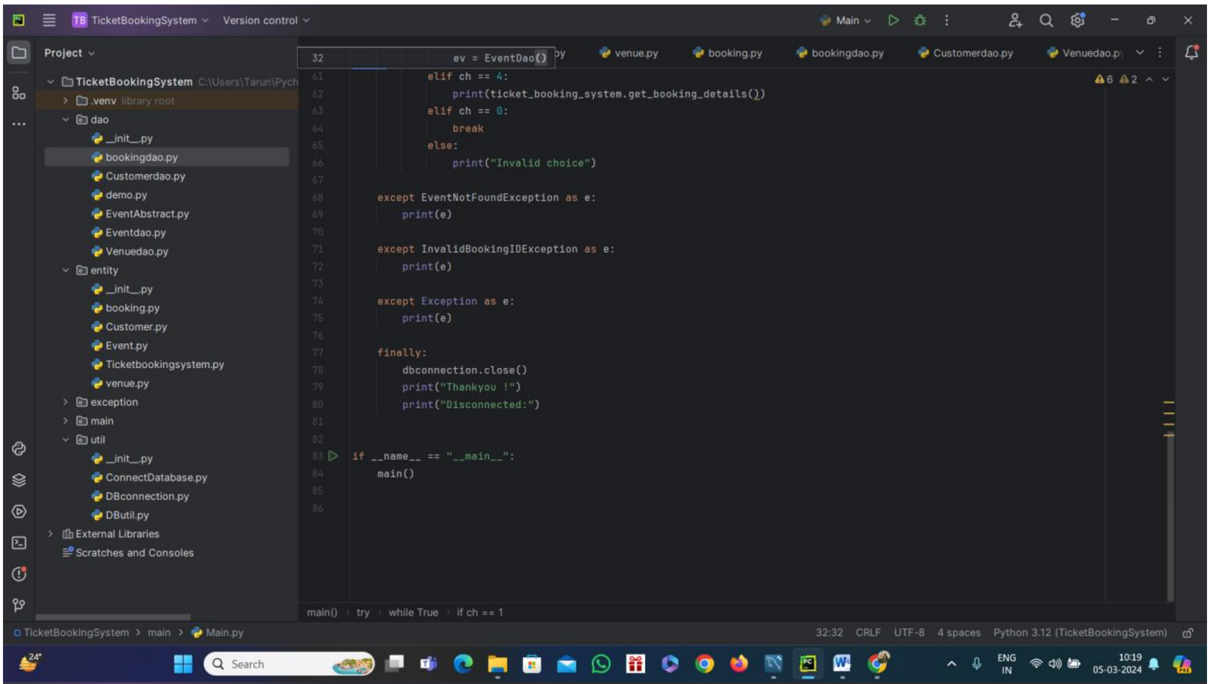
```
                elif ch == 4:
                    print(ticket_booking_system.get_booking_details())
                elif ch == 0:
                    break
                else:
                    print("Invalid choice")

        except EventNotFoundException as e:
            print(e)

        except InvalidBookingIDException as e:
            print(e)

```

```
Welcome to Ticket Booking  System!
1.Event 2.Customer 3.Venue 4.Booking  0.EXIT
Enter choice: 4
(Booking) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Booking ID: 1
Booking Name : Tarun
True
(Booking) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Booking Table :
(1, 'Tarun')
(Booking) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```

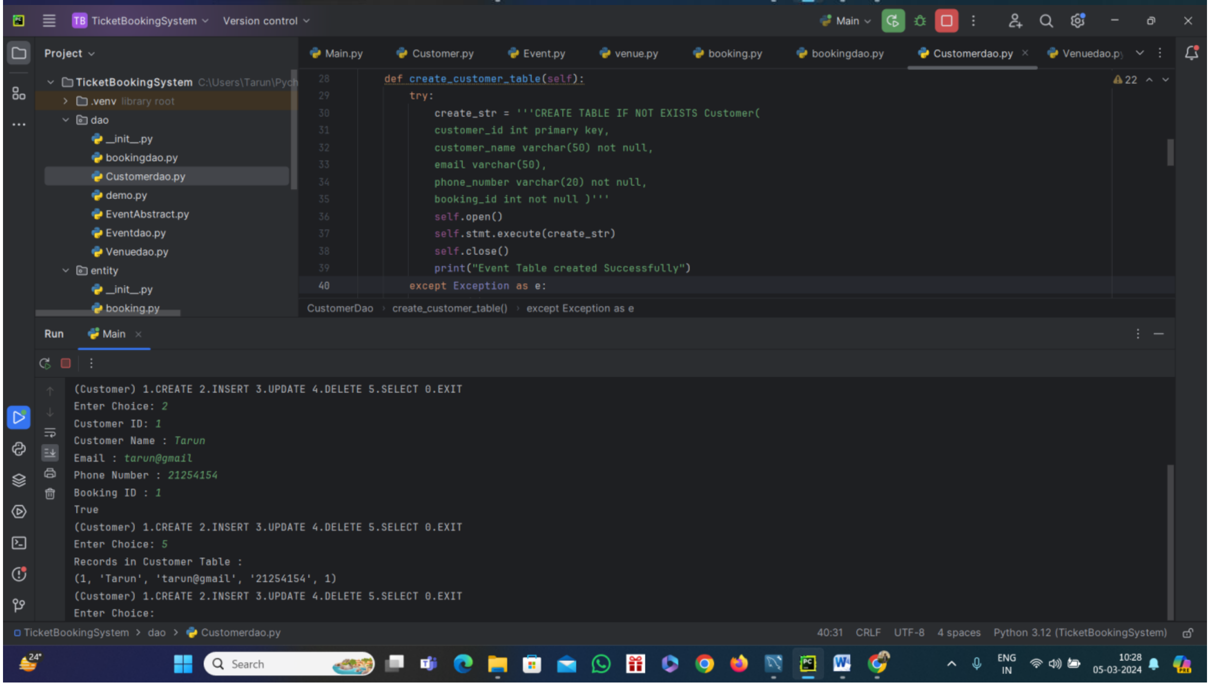


```
    def create_customer_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Customer(
            customer_id int primary key,
            customer_name varchar(50) not null,
            email varchar(50),
            phone_number varchar(20) not null,
            booking_id int not null )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Event Table created Successfully")
        except Exception as e:
```

```
------------------------------
Welcome to Ticket Booking  System!
1.Event 2.Customer 3.Venue 4.Booking  0.EXIT
Enter choice: 2
(Customer) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Customer ID: 1
Customer Name : Tarun
Email : tarun@gmail
Phone Number : 21254154
Booking ID : 1
True
(Customer) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```

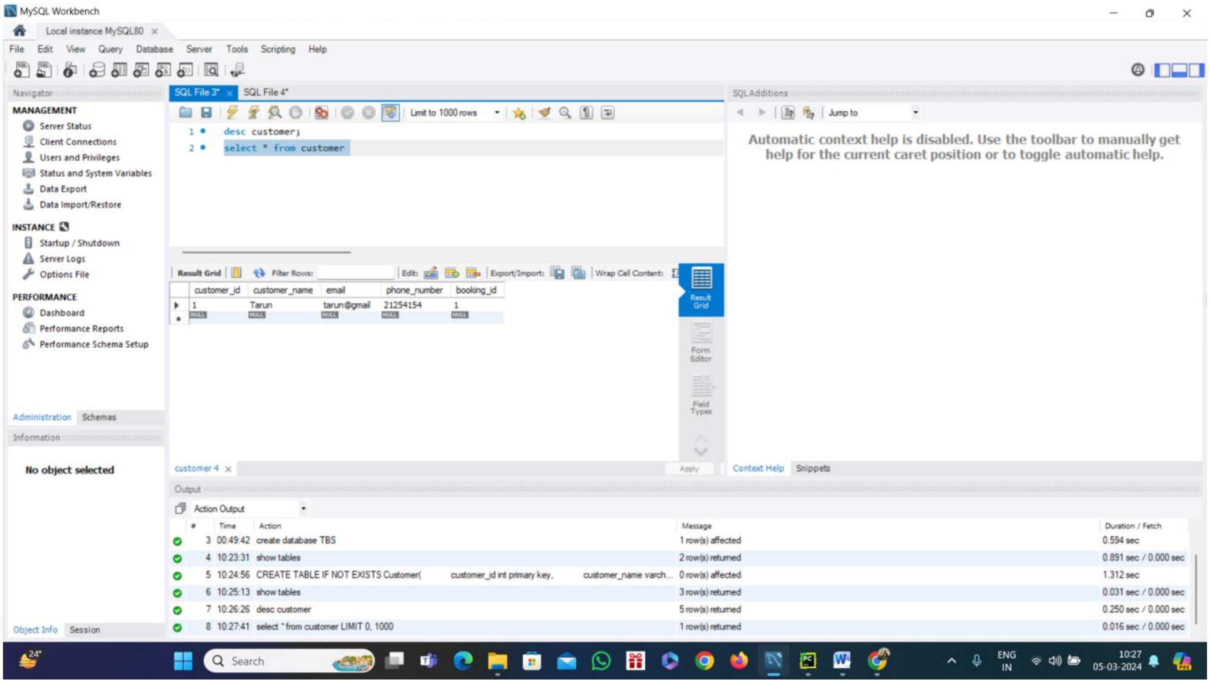


```
            ev = EventDao()
            elif ch == 4:
                print(ticket_booking_system.get_booking_details())
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    except EventNotFoundException as e:
        print(e)

    except InvalidBookingIDException as e:
        print(e)

    except Exception as e:
        print(e)

    finally:
        dbconnection.close()
        print("Thankyou !")
        print("Disconnected:")


if __name__ == "__main__":
    main()
```



```
    def create_customer_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Customer(
            customer_id int primary key,
            customer_name varchar(50) not null,
            email varchar(50),
            phone_number varchar(20) not null,
            booking_id int not null )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Event Table created Successfully")
        except Exception as e:
```

```
(Customer) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 2
Customer ID: 1
Customer Name : Tarun
Email : tarun@gmail
Phone Number : 21254154
Booking ID : 1
True
(Customer) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Customer Table :
(1, 'Tarun', 'tarun@gmail', '21254154', 1)
(Customer) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
```



```
desc customer;
select * from customer
```

customer_id	customer_name	email	phone_number	booking_id
1	Tarun	tarun@gmail	21254154	1
NULL	NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

No object selected

#	Time	Action	Message	Duration / Fetch
3	00:49:42	create database TBS	1 row(s) affected	0.594 sec
4	10:23:31	show tables	2 row(s) returned	0.891 sec / 0.000 sec
5	10:24:56	CREATE TABLE IF NOT EXISTS Customer(customer_id int primary key, customer_name varch...	0 row(s) affected	1.312 sec
6	10:25:13	show tables	3 row(s) returned	0.031 sec / 0.000 sec
7	10:26:26	desc customer	5 row(s) returned	0.250 sec / 0.000 sec
8	10:27:41	select * from customer LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec