E-COMMERCE APPLICATION – CASE STUDY

ENTITY:

Customers.py

```
from util.DBConnUtil import DBConnection

class Customers(DBConnection):
    def __init__(self):
        super().__init__()
        self.customerId = 0
        self.name = ' '
        self.email = ' '
        self.password = ' '

    #Setters

    def set_customerId(self,value):
        self.customerId = value

    def set_name(self,value):
        self.name = value

    def set_email(self,value):
        self.email = value

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

    #Getters

    def get_customerId(self):
        return self.customerId

    def get_name(self):
        return self.name

    def get_email(self):
        return self.email

    def get_password(self):
        return self.password

    def __str__(self):
        return f'customerId : {self.customerId} name: {self.name} email: {self.email}
password: {self.password}'
```

Products.py

```
from util.DBConnUtil import DBConnection

class Products(DBConnection):
    def __init__(self):
        super().__init__()
        self.productId = 0
        self.name = ' '
        self.price = 0.0
        self.description = ' '
        self.stockQuantity = 0
```

```
    #Setters

    def set_productId(self,value):
        self.productId = value

    def set_name(self,value):
        self.name = value

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

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

    def set_stockQuantity(self,value):
        self.stockQuantity = value

    #Getters

    def get_productId(self):
        return self.productId

    def get_name(self):
        return self.name

    def get_price(self):
        return self.price

    def get_description(self):
        return self.description

    def get_stockQuantity(self):
        return self.stockQuantity

    def __str__(self):
        return (f'productId : {self.productId} name: {self.name} price: {self.price}\n'
\
                f'description : {self.description} Stock: {self.stockQuantity}')
```

Cart.py

```
from entity.Customers import Customers
from entity.Products import Products

class Cart(Customers,Products):
    def __init__(self):
        super().__init__()
        self.cartId = 0
        self.customerId = 0
        self.productId = 0
        self.quantity = 0

    #Setters

    def set_cartId(self,value):
        self.cartId = value

    def set_customerId(self,value):
        self.customerId = value

    def set_productId(self,value):
        self.productId = value
```

```
    def set_quantity(self,value):
        self.quantity = value

    #Getters

    def get_cartId(self):
        return self.cartId

    def get_customerId(self):
        return self.customerId

    def get_productId(self):
        return self.productId

    def get_quantity(self):
        return self.quantity

    def __str__(self):
        return (f'cartId : {self.cartId} customerId: {self.customerId}\n' \
                f'productId : {self.productId} quantity: {self.quantity}')
```

Orders.py

```
from entity.Customers import Customers

class Orders(Customers):
    def __init__(self):
        super().__init__()
        self.orderId = 0
        self.customerId = 0
        self.orderDate = ' '
        self.totalPrice = 0.0
        self.shippingAddress = ' '

    #Setters

    def set_orderId(self,value):
        self.orderId = value

    def set_customerId(self,value):
        self.customerId = value

    def set_orderDate(self,value):
        self.orderDate = value

    def set_totalPrice(self,value):
        self.totalPrice = value

    def set_shippingAddress(self,value):
        self.shippingAddress = value


    #Getters

    def get_orderId(self):
        return self.orderId

    def get_customerId(self):
        return self.customerId

    def get_orderDate(self):
        return self.orderDate

    def get_totalPrice(self):
        return self.totalPrice
```

```
    def get_shippingAddress(self):
        return self.shippingAddress


    def __str__(self):
        return (f'orderId : {self.orderId} customerId: {self.customerId}\n' \
                f'orderDate : {self.orderDate} totalPrice: {self.totalPrice}
shippingAddress : {self.shippingAddress}')
```

OrderItems.py

```
from entity.Orders import Orders
from entity.Products import Products

class OrderItems(Orders,Products):
    def __init__(self):
        super().__init__()
        self.orderitemId = 0
        self.orderId = 0
        self.productId = 0
        self.quantity = 0

    #Setters

    def set_orderitemId(self,value):
        self.orderitemId = value

    def set_orderId(self,value):
        self.orderId = value

    def set_productId(self,value):
        self.productId = value

    def set_quantity(self,value):
        self.quantity = value

    #Getters

    def get_orderitemId(self):
        return self.orderitemId

    def get_orderId(self):
        return self.orderId

    def get_productId(self):
        return self.productId

    def get_quantity(self):
        return self.quantity

    def __str__(self):
        return (f'orderitemId : {self.orderitemId} orderId: {self.orderId}\n' \
                f'productId : {self.productId} quantity: {self.quantity}')
```

DAO:

CustomersDAO.py

```
from entity.Customers import Customers
from exception.CustomerNotFoundException import CustomerNotFoundException

class CustomerDAO(Customers):
    def __init__(self):
        super().__init__()

    def perform_customer_actions(self):
        while True:
            print("(Customers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter your choice: "))
            if ch == 1:
                self.create_customers_table()
            elif ch == 2:
                print(self.add_customers())
            elif ch == 3:
                print(self.update_customers())
            elif ch == 4:
                print(self.delete_customers())
            elif ch == 5:
                print(self.select_customers())
            elif ch == 0:
                break
            else:
                print("Invalid input")

    def create_customers_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Customers(
                            customerId INT PRIMARY KEY,
                            name VARCHAR(50) NOT NULL,
                            email VARCHAR(50),
                            password VARCHAR(50)
                            )
                            '''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Customers Table created successfully")
        except Exception as e:
            print(e)

    def add_customers(self):
        try:
            self.open()
            self.customerId = int(input("Enter Customer ID: "))
            self.name = input("Enter Name: ")
            self.email = input("Enter Email: ")
            self.password = input("Enter password: ")
            data = [(self.customerId,self.name,self.email,self.password)]
            insert_str = '''INSERT INTO Customers(customerId,name,email,password)
                            VALUES(%s,%s,%s,%s)'''

            self.stmt.executemany(insert_str,data)
            self.conn.commit()
            self.close()
            print("Inserted Successfully")
            return True
        except Exception as e:
            return e
```

```
    def update_customers(self):
        try:
            self.open()
            customer_id = int(input("Enter Customer ID to be updated: "))
            self.name = input("Enter name: ")
            self.email = input("Enter Email: ")
            self.password = input("Enter password: ")
            data = [(self.name,self.email,self.password,customer_id)]
            update_str = f'''UPDATE Customers SET name = %s, email = %s, password = %s
                            WHERE customerId = %s'''
            self.stmt.executemany(update_str,data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
        except CustomerNotFoundException as e:
            return e

    def delete_customers(self):
        try:
            self.open()
            customer_id = int(input("Enter customerId to be deleted: "))
            delete_str = f'''DELETE FROM Customers WHERE CustomerId = {customer_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            print("Deleted Successfully")
            return True
        except CustomerNotFoundException as e:
            return e

    def select_customers(self):
        try:
            self.open()
            select_str = '''SELECT * FROM Customers'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

ProductsDAO.py

```
from entity.Products import Products
from exception.ProductNotFoundException import ProductNotFoundException

class ProductsDAO(Products):
    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 your choice: "))
            if ch == 1:
                self.create_products_table()
            elif ch == 2:
                print(self.add_products())
            elif ch == 3:
                print(self.update_products())
```

```
            elif ch == 4:
                print(self.delete_products())
            elif ch == 5:
                print(self.select_products())
            elif ch == 0:
                break
            else:
                print("Invalid input")

    def create_products_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Products(
                            productId INT PRIMARY KEY,
                            name VARCHAR(50) NOT NULL,
                            price FLOAT,
                            description TEXT,
                            stockQuantity INT NOT NULL
                            )
                            '''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Products Table created successfully")
        except Exception as e:
            print(e)

    def add_products(self):
        try:
            self.open()
            self.productId = int(input("Enter Product ID: "))
            self.name = input("Enter Name: ")
            self.price = float(input("Enter price: "))
            self.description = input("Enter description about product: ")
            self.stockQuantity = int(input("Enter stock Quantity: "))
            data =
[(self.productId,self.name,self.price,self.description,self.stockQuantity)]
            insert_str = '''INSERT INTO
Products(productId,name,price,description,stockQuantity)
                            VALUES(%s,%s,%s,%s,%s)'''

            self.stmt.executemany(insert_str,data)
            self.conn.commit()
            self.close()
            print("Inserted Successfully")
            return True
        except Exception as e:
            print(e)

    def update_products(self):
        try:
            self.open()
            product_id = int(input("Enter Product ID to be updated: "))
            self.name = input("Enter name: ")
            self.price = float(input("Enter price: "))
            self.description = input("Enter description: ")
            self.stockQuantity = int(input("Enter stock quantity: "))
            data =
[(self.name,self.price,self.description,self.stockQuantity,product_id)]
            update_str = f'''UPDATE Products SET name = %s, price = %s, description =
%s, stockQuantity = %s
                            WHERE productId = %s'''
            self.stmt.executemany(update_str,data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
```

```
        except ProductNotFoundException as e:
            return e

    def delete_products(self):
        try:
            self.open()
            product_id = int(input("Enter productId to be deleted: "))
            delete_str = f'''DELETE FROM Products WHERE productId = {product_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            print("Deleted Successfully")
            return True
        except ProductNotFoundException as e:
            return e

    def select_products(self):
        try:
            self.open()
            select_str = '''SELECT * FROM Products'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

CartDAO:

```
from entity.Customers import Customers
from entity.Products import Products

class CartDAO(Customers,Products):
    def __init__(self):
        super().__init__()

    def perform_cart_actions(self):
        while True:
            print("(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter your choice: "))
            if ch == 1:
                self.create_cart_table()
            elif ch == 2:
                print(self.add_cart())
            elif ch == 3:
                print(self.update_cart())
            elif ch == 4:
                print(self.delete_cart())
            elif ch == 5:
                print(self.select_cart())
            elif ch == 0:
                break
            else:
                print("Invalid input")

    def create_cart_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Cart(
                            cartId INT PRIMARY KEY,
                            customerId INT,
                            productId INT,
                            quantity INT,
                            FOREIGN KEY(customerId) REFERENCES Customers(customerId) ON
DELETE CASCADE,
                            FOREIGN KEY(productId) REFERENCES Products(productId) ON
```

```
DELETE CASCADE
                            )
                            '''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Cart Table created successfully")
        except Exception as e:
            print(e)

    def add_cart(self):
        try:
            self.open()
            self.cartId = int(input("Enter Cart ID: "))
            self.customerId = int(input("Enter Customer ID: "))
            self.productId = int(input("Enter Product ID: "))
            self.quantity = int(input("Enter quantity: "))
            data = [(self.cartId,self.customerId,self.productId,self.quantity)]
            insert_str = '''INSERT INTO Cart(cartId,customerId,productId,quantity)
                            VALUES(%s,%s,%s,%s)'''

            self.stmt.executemany(insert_str,data)
            self.conn.commit()
            self.close()
            print("Inserted Successfully")
            return True
        except Exception as e:
            print(e)

    def update_cart(self):
        try:
            self.open()
            cart_id = int(input("Enter cart ID to be updated: "))
            self.customerId = int(input("Enter Customer ID: "))
            self.productId = int(input("Enter Product ID: "))
            self.quantity = int(input("Enter quantity: "))
            data = [(self.customerId,self.productId,self.quantity,cart_id)]
            update_str = f'''UPDATE Cart SET customerId = %s, productId = %s, quantity=
%s WHERE cartId = %s'''
            self.stmt.executemany(update_str,data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
        except Exception as e:
            print(e)

    def delete_cart(self):
        try:
            self.open()
            cart_id = int(input("Enter Cart ID to be deleted: "))
            delete_str = f'''DELETE FROM Cart WHERE cartId = {cart_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            print("Deleted Successfully")
            return True
        except Exception as e:
            print(e)

    def select_cart(self):
        try:
            self.open()
            select_str = '''SELECT * FROM Cart'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            for i in records:
```

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

OrdersDAO:

```
from entity.Orders import Orders
from exception.OrderNotFoundException import OrderNotFoundException
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 your choice: "))
            if ch == 1:
                self.create_Orders_table()
            elif ch == 2:
                print(self.add_Orders())
            elif ch == 3:
                print(self.update_Orders())
            elif ch == 4:
                print(self.delete_Orders())
            elif ch == 5:
                print(self.select_Orders())
            elif ch == 0:
                break
            else:
                print("Invalid input")

    def create_Orders_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Orders(
                            orderId INT PRIMARY KEY,
                            customerId INT,
                            orderDate DATE,
                            totalPrice FLOAT,
                            shippingAddress VARCHAR(50),
                            FOREIGN KEY(customerId) REFERENCES Customers(CustomerId) ON
DELETE CASCADE
                            )
                            '''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Orders Table created successfully")
        except Exception as e:
            print(e)

    def add_Orders(self):
        try:
            self.open()
            self.orderId = int(input("Enter Order ID: "))
            self.customerId = int(input("Enter Customer ID: "))
            self.orderDate = input("Enter Order date: ")
            self.totalPrice = float(input("Enter Total Price: "))
            self.shippingAddress = input("Enter Shipping Address: ")
            data =
[(self.orderId,self.customerId,self.orderDate,self.totalPrice,self.shippingAddress)]
            insert_str = '''INSERT INTO
Orders(orderId,customerId,orderDate,totalPrice,shippingAddress)
                            VALUES(%s,%s,%s,%s,%s)'''

            self.stmt.executemany(insert_str,data)
```

```
            self.conn.commit()
            self.close()
            print("Inserted Successfully")
            return True
        except Exception as e:
            print(e)

    def update_Orders(self):
        try:
            self.open()
            order_id = int(input("Enter Order ID to be updated: "))
            self.customerId = int(input("Enter Customer ID: "))
            self.orderDate = input("Enter Order Date: ")
            self.totalPrice = float(input("Enter Total Price: "))
            self.shippingAddress = input("Enter Shipping Address: ")
            data =
[(self.customerId,self.orderDate,self.totalPrice,self.shippingAddress,order_id)]
            update_str = f'''UPDATE Orders SET customerId = %s, orderDate = %s,
totalPrice = %s, shippingAddress = %s
                            WHERE orderId = %s'''
            self.stmt.executemany(update_str,data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
        except OrderNotFoundException as e:
            return 3

    def delete_Orders(self):
        try:
            self.open()
            order_id = int(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()
            print("Deleted Successfully")
            return True
        except OrderNotFoundException as e:
            return e

    def select_Orders(self):
        try:
            self.open()
            select_str = '''SELECT * FROM Orders'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrderItemsDAO:

```
from entity.Orders import Orders
from entity.Products import Products

class OrderItemsDAO(Orders,Products):
    def __init__(self):
        super().__init__()

    def perform_OrderItems_actions(self):
        while True:
            print("(Order Items) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
```

```
            ch = int(input("Enter your choice: "))
            if ch == 1:
                self.create_OrderItems_table()
            elif ch == 2:
                print(self.add_OrderItems())
            elif ch == 3:
                print(self.update_OrderItems())
            elif ch == 4:
                print(self.delete_OrderItems())
            elif ch == 5:
                print(self.select_OrderItems())
            elif ch == 0:
                break
            else:
                print("Invalid input")

    def create_OrderItems_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS OrderItems(
                            OrderItemId INT PRIMARY KEY,
                            orderId INT,
                            productId INT,
                            quantity INT,
                            FOREIGN KEY(orderId) REFERENCES Orders(orderId) ON DELETE
CASCADE,
                            FOREIGN KEY(productId) REFERENCES Products(productId) ON
DELETE CASCADE
                            )
                            '''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("OrderItems Table created successfully")
        except Exception as e:
            print(e)

    def add_OrderItems(self):
        try:
            self.open()
            self.OrderItemId = int(input("Enter OrderItem ID: "))
            self.orderId = int(input("Enter order ID: "))
            self.productId = int(input("Enter product ID: "))
            self.quantity = int(input("Enter quantity: "))
            data = [(self.OrderItemId,self.orderId,self.productId,self.quantity)]
            insert_str = '''INSERT INTO
OrderItems(OrderItemId,orderId,productId,quantity)
                            VALUES(%s,%s,%s,%s)'''

            self.stmt.executemany(insert_str,data)
            self.conn.commit()
            self.close()
            print("Inserted Successfully")
            return True
        except Exception as e:
            print(e)

    def update_OrderItems(self):
        try:
            self.open()
            OrderItem_id = int(input("Enter OrderItem ID to be updated: "))
            self.orderId = int(input("Enter order ID: "))
            self.productId = int(input("Enter product ID: "))
            self.quantity = int(input("Enter quantity: "))
            data = [(self.orderId,self.productId,self.quantity,OrderItem_id)]
            update_str = f'''UPDATE OrderItems SET orderId = %s, productId = %s,
quantity = %s
                            WHERE OrderItemId = %s'''
```

```
            self.stmt.executemany(update_str,data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
        except Exception as e:
            print(e)

    def delete_OrderItems(self):
        try:
            self.open()
            OrderItem_id = int(input("EnterOrderItem ID to be deleted: "))
            delete_str = f'''DELETE FROM OrderItems WHERE OrderItemId =
{OrderItem_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            print("Deleted Successfully")
            return True
        except Exception as e:
            print(e)

    def select_OrderItems(self):
        try:
            self.open()
            select_str = '''SELECT * FROM OrderItems'''
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrderProcessor:

```
from dao.CartDAO import CartDAO
from dao.ProductsDAO import ProductsDAO
from dao.CustomersDAO import CustomerDAO
from dao.OrdersDAO import OrdersDAO
from exception.CustomerNotFoundException import CustomerNotFoundException

class OrderProcessor(CartDAO,OrdersDAO):
    def __init__(self):
        super().__init__()

    def createProduct(self):
        p = ProductsDAO()
        p.add_products()

    def createCustomer(self):
        c = CustomerDAO()
        c.add_customers()

    def deleteProduct(self):
        p = ProductsDAO()
        p.select_products()
        print()
        p = ProductsDAO()
        p.delete_products()
        print("After deleting:")
        p = ProductsDAO()
        p.select_products()

    def deleteCustomer(self):
        c = CustomerDAO()
        c.select_customers()
```

```
        print()
        c = CustomerDAO()
        c.delete_customers()
        print("After deleting:")
        c = CustomerDAO()
        c.select_customers()

    def addToCart(self):
        c1 = CartDAO()
        c1.add_cart()

    def removefromCart(self):
        c1 = CartDAO()
        c1.select_cart()
        print()
        c1 = CartDAO()
        c1.delete_cart()
        print()

    def viewCart(self):
        print('='*22)
        print("Products for reference")
        print('='*22)
        p1 = ProductsDAO()
        p1.select_products()
        print('-'*13)
        print("items in cart")
        print('-'*13)
        c1 = CartDAO()
        c1.select_cart()
        print()

    def placeOrder(self):
        o = OrdersDAO()
        o.add_Orders()

    def viewCustomerOrder(self,customerId):
        try:
            self.open()
            #customer_id = int(input("Enter customer ID: "))
            self.stmt.execute(f'''SELECT COUNT(*) FROM Orders WHERE customerId =
{customerId}''')
            count = self.stmt.fetchone()[0]
            if count == 0:
                return CustomerNotFoundException(customerId)
            else:
                print("Total Count or Orders: ", count)
                self.stmt.execute(f'''SELECT * FROM Orders WHERE customerId =
{customerId} ''')
                records = self.stmt.fetchall()
                self.close()
                return records

        except CustomerNotFoundException as e:
            return e
        except Exception as e:
            return e
```

EXCEPTIONS:

CustomerNotFoundException:

```
class CustomerNotFoundException(Exception):
    def __init__(self,customerId):
        super().__init__(f"Customer Id : {customerId} not found in the database")
```

OrderNotFoundException:

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

ProductNotFoundException:

```
class ProductNotFoundException(Exception):
    def __init__(self,productId):
        super().__init__(f'product ID : {productId} not found in the system..')
```

UTIL:

DBConnUtil:

```
#import sys
import mysql.connector as sql
from util.DBPropertyUtil import PropertyUtil


class DBConnection:
    def open(self):
        try:
            # print('--Database Is Connected:--')
            connection_properties = PropertyUtil.getConnectionString()
            self.conn = sql.connect(**connection_properties)
            self.stmt = self.conn.cursor()
        except Exception as e:
            print(str(e) + ' --Database Is Not Connected:--')
            sys.exit(1)

    def close(self):
        self.conn.close()
        # print('--Connection Is Closed:--')
```

DBPropertyUtil:

```
class PropertyUtil:
    connection_properties = None

    @staticmethod
    def getConnectionString():
        if PropertyUtil.connection_properties is None:
            host = 'localhost'
            database = 'ecomdb'
            user = 'root'
            password = 'Mahitharsha@1'
            PropertyUtil.connection_properties = {'host': host, 'database': database,
'user': user, 'password': password}
        return PropertyUtil.connection_properties
```

MAIN:

Main.py (Executable file)

```
from dao.ProductsDAO import ProductsDAO
from dao.OrdersDAO import OrdersDAO
from dao.OrderItemsDAO import OrderItemsDAO
from dao.CustomersDAO import CustomerDAO
from dao.CartDAO import CartDAO
from exception.OrderNotFoundException import OrderNotFoundException
from exception.ProductNotFoundException import ProductNotFoundException
from exception.CustomerNotFoundException import CustomerNotFoundException
from util.DBConnUtil import DBConnection
from dao.OrderProcessor import OrderProcessor

def main():
    dbconnection = DBConnection()

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

    try:
        print("=" * 22)
        print("E-Commerce Application")
        print("=" * 22)
        print("Welcome to E commerce Online Shopping!")

        while True:
            print("1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit")
            ch = int(input("Enter choice: "))
            if ch == 1:
                c = CustomerDAO()
                c.perform_customer_actions()
            elif ch == 2:
                p = ProductsDAO()
                p.perform_Product_actions()
            elif ch == 3:
                c1 = CartDAO()
                c1.perform_cart_actions()
            elif ch == 4:
                o = OrdersDAO()
                o.perform_Orders_actions()
            elif ch == 5:
                o1 = OrderItemsDAO()
                o1.perform_OrderItems_actions()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

        ecommerce = OrderProcessor()

        while True:
            print("*"*10)
            print("---Menu---")
            print("*"*10)
            print("1.Register Customer\n2.Create Product\n3.Delete Product\n4.Delete
Customer\n5.Add To Cart\n6.Remove From Cart\n7.View Cart\n8.Place Order\n9.View
Customer Order\n0.Exit")
            ch = int(input("Enter choice: "))
            if ch == 1:
                print(f'Customer Created {ecommerce.createCustomer()}')
            elif ch == 2:
                ecommerce.createProduct()
```

```
            elif ch == 3:
                print(f'Product deleted: {ecommerce.deleteProduct()}')
            elif ch == 4:
                ecommerce.deleteCustomer()
            elif ch == 5:
                ecommerce.addToCart()
            elif ch == 6:
                ecommerce.removefromCart()
            elif ch == 7:
                ecommerce.viewCart()
            elif ch == 8:
                print(f'Order placed : {ecommerce.placeOrder()}')
            elif ch == 9:
                print(f'Customer Orders: {ecommerce.viewCustomerOrder(int(input("Enter
customer ID:")))}')
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    except CustomerNotFoundException as e:
        return e
    except ProductNotFoundException as e:
        return e
    except OrderNotFoundException as e:
        return e

    finally:
        dbconnection.close()
        print("Thank you for visiting!!")
        print("--Connection is closed__")

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

UNIT TESTING:

Test.py

```
import unittest
from unittest.mock import Mock
from dao.OrderProcessor import OrderProcessor
from dao.ProductsDAO import ProductsDAO
from dao.CustomersDAO import CustomerDAO
from dao.OrdersDAO import OrdersDAO
from dao.CartDAO import CartDAO



class TestOrderProcessor(unittest.TestCase):
    def setUp(self):
        # Set up any necessary objects or mocks
        self.order_processor_repo = OrderProcessor

    def test_create_product(self):
        # Mocking a product
        product = ProductsDAO()

        # Mocking the repository's interaction with the database
        self.order_processor_repo.createProduct = Mock(return_value=True)
```

```python
        # Testing the create_product method
        result = self.order_processor_repo.createProduct(product)

        self.assertTrue(result)

    # Similarly, you can write tests for other methods in OrderProcessorRepository

    def test_create_customer(self):
        # Mocking a product
        customer = CustomerDAO()

        # Mocking the repository's interaction with the database
        self.order_processor_repo.createCustomer = Mock(return_value=True)

        # Testing the create_product method
        result = self.order_processor_repo.createCustomer(customer)

        self.assertTrue(result)

    def test_order_product(self):
        # Mocking a product
        order = OrdersDAO()

        # Mocking the repository's interaction with the database
        self.order_processor_repo.add_Orders = Mock(return_value=True)

        # Testing the order_product method
        result = self.order_processor_repo.add_Orders(order)

        self.assertTrue(result)

    def test_add_to_cart(self):
        customer = CustomerDAO()
        product = ProductsDAO()
        quantity = 3

        self.order_processor_repo.add_to_cart = Mock(return_value=True)
        result = self.order_processor_repo.add_to_cart(customer, product, quantity)

        self.assertTrue(True)


if __name__ == '_main_':
    unittest.main()
```

OUTPUTS:

#Running the Main file

```
main ×    OrderProcessor ×
C:\Users\shyam\PycharmProjects\E-COMMERCE\venv\Scripts\python.exe C:\Users\shyam\PycharmProjects\E-COMMERCE\main\main.py
--Database Is Connected:--
=====================
E-Commerce Application
=====================
Welcome to E commerce Online Shopping!
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice:
```

#Selecting the options

```
Run:  main  OrderProcessor
Welcome to E commerce Online Shopping!
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 1
(Customers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter your choice: 0
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 2
(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter your choice: 0
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 3
(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter your choice: 0
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 4
(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter your choice: 0
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 5
(Order Items) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter your choice: 0
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
```

➔ By selecting respective options from above we can perform CREATE,INSERT,UPDATE,DELETE and SELECT operations that automatically reflects in our database.

#Directly exit from the loop to enter into the menu

```
C:\Users\shyam\PycharmProjects\E-COMMERCE\venv\Scripts\python.exe C:\Users\shyam\PycharmProjects\E-COMMERCE\main\main.py
--Database Is Connected:--
=====================
E-Commerce Application
=====================
Welcome to E commerce Online Shopping!
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 0
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice:
```

1. Register Customer

```
=======================
Welcome to E commerce Online Shopping!
1.Customer 2.Products 3.Cart 4.Orders 5.Order Items 0.exit
Enter choice: 0
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 1
Enter Customer ID: 7
Enter Name: Ayan
Enter Email: Ayan@gmail.com
Enter password: Ayan@1
Inserted Successfully
```

2. Create Product

```
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 2
Enter Product ID: 7
Enter Name: Charger
Enter price: 599
Enter description about product: Type c charger
Enter stock Quantity: 20
Inserted Successfully
```

3. Delete Product

```
Enter choice: 3
(1, 'realme 11 pro', 23000.0, 'Mobile', 22)
(2, 'redmi note 11', 13000.0, 'Mobile', 20)
(3, 'Hp pavilion', 55000.0, 'Laptop', 12)
(4, 'Hp mouse', 1999.0, 'Wireless mouse', 23)
(5, 'Earbuds', 1999.0, 'TWS', 12)
(7, 'Charger', 599.0, 'Type c charger', 20)

Enter productId to be deleted: 7
Deleted Successfully
After deleting:
(1, 'realme 11 pro', 23000.0, 'Mobile', 22)
(2, 'redmi note 11', 13000.0, 'Mobile', 20)
(3, 'Hp pavilion', 55000.0, 'Laptop', 12)
(4, 'Hp mouse', 1999.0, 'Wireless mouse', 23)
(5, 'Earbuds', 1999.0, 'TWS', 12)
Product deleted: None
```

4. Delete Customer

```
Enter choice: 4
(1, 'HARSHA', 'HARSHA@GJMAIL.CO,', 'PASSWORD@1')
(2, 'Dheeraj', 'Dheeraj@gmail.com', 'Dheerak@1')
(3, 'mahii', 'mahi@gmail.com', 'mahi@1')
(4, 'dddd', 'ffff', 'ffff')
(5, 'qqqq', 'qqq@gmail.com', 'qqq@1')
(7, 'Ayan', 'Ayan@gmail.com', 'Ayan@1')

Enter customerId to be deleted: 7
Deleted Successfully
After deleting:
(1, 'HARSHA', 'HARSHA@GJMAIL.CO,', 'PASSWORD@1')
(2, 'Dheeraj', 'Dheeraj@gmail.com', 'Dheerak@1')
(3, 'mahii', 'mahi@gmail.com', 'mahi@1')
(4, 'dddd', 'ffff', 'ffff')
(5, 'qqqq', 'qqq@gmail.com', 'qqq@1')
**********
```

5. Add to Cart

```
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 5
Enter Cart ID: 7
Enter Customer ID: 2
Enter Product ID: 3
Enter quantity: 1
Inserted Successfully
```

6. Remove From Cart

```
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 6
(1, 1, 1, 1)
(2, 2, 1, 1)
(3, 3, 1, 1)
(5, 1, 2, 2)
(7, 2, 3, 1)

Enter Cart ID to be deleted: 7
Deleted Successfully
```

7. View Cart

```
Enter choice: 7
======================
Products for reference
======================
(1, 'realme 11 pro', 23000.0, 'Mobile', 22)
(2, 'redmi note 11', 13000.0, 'Mobile', 20)
(3, 'Hp pavilion', 55000.0, 'Laptop', 12)
(4, 'Hp mouse', 1999.0, 'Wireless mouse', 23)
(5, 'Earbuds', 1999.0, 'TWS', 12)
-------------
items in cart
-------------
(1, 1, 1, 1)
(2, 2, 1, 1)
(3, 3, 1, 1)
(5, 1, 2, 2)
```

8. Place Order

```
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 8
Enter Order ID: 8
Enter Customer ID: 3
Enter Order date: 2024-03-08
Enter Total Price: 13000
Enter Shipping Address: ght street
Inserted Successfully
```

9. View Customer Order

```
Enter choice: 9
Enter customer ID:2
Total Count or Orders:  2
Customer Orders: [(2, 2, datetime.date(2024, 2, 28), 12000.0, 'town street'), (5, 2, datetime.date(2024, 3, 1), 1000.0, 'mjcvh')]
```

0. Exit

```
**********
---Menu---
**********
1.Register Customer
2.Create Product
3.Delete Product
4.Delete Customer
5.Add To Cart
6.Remove From Cart
7.View Cart
8.Place Order
9.View Customer Order
0.Exit
Enter choice: 0
Thank you for visiting!!
--Connection is closed__
```