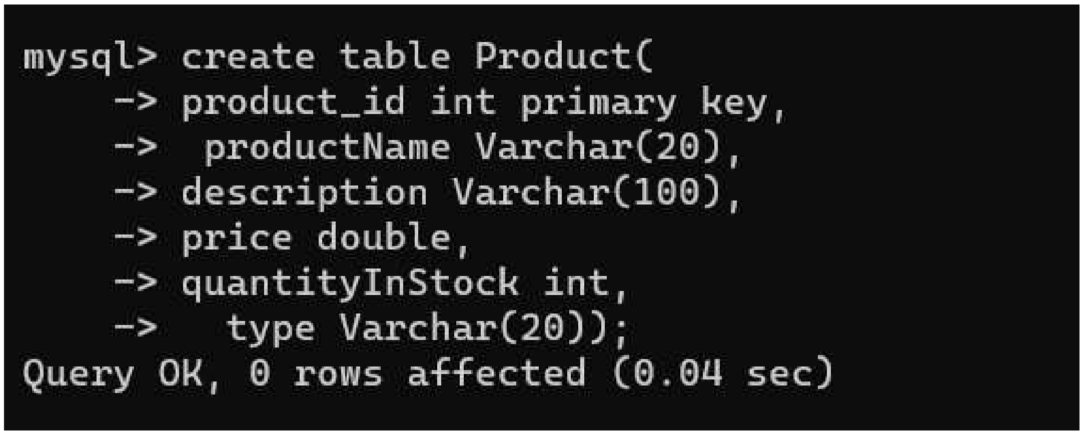
# *Python Coding Challenge*

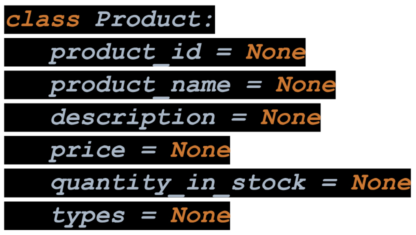
***Create SQL Schema from the product and user class, use the class attributes for table column names.***

***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]***

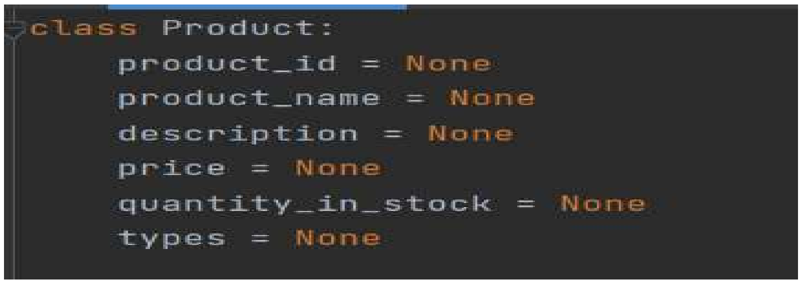
*create table Product(*
*-> product_id int primary key,*
*->  productName Varchar(20),*
*-> description Varchar(100),*
*-> price double,*
*-> quantityInStock int,*
*->   type Varchar(20));*



```
mysql> create table Product(
    -> product_id int primary key,
    ->  productName Varchar(20),
    -> description Varchar(100),
    -> price double,
    -> quantityInStock int,
    ->   type Varchar(20));
Query OK, 0 rows affected (0.04 sec)
```



```
class Product:
    product_id = None
    product_name = None
    description = None
    price = None
    quantity_in_stock = None
    types = None
```

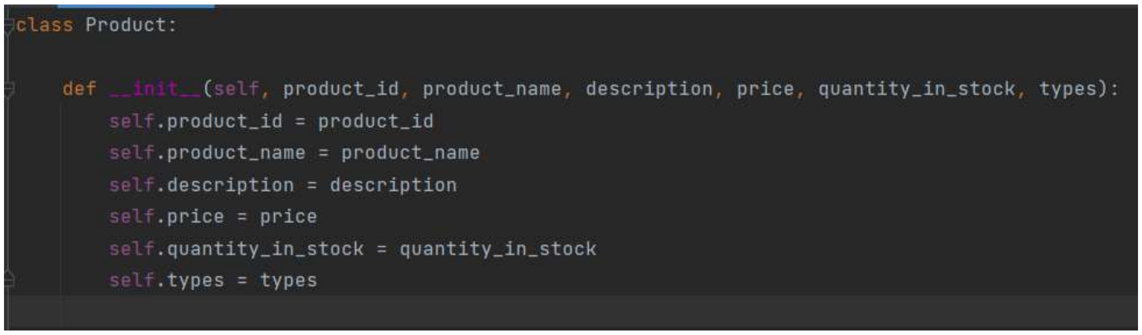


```
class Product:
    product_id = None
    product_name = None
    description = None
    price = None
    quantity_in_stock = None
    types = None
```

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

*Implementing Constructor:*

```
class Product:

    def __init__(self, product_id, product_name, description, price,
quantity_in_stock, types):
        self.product_id = product_id
        self.product_name = product_name
        self.description = description
        self.price = price
        self.quantity_in_stock = quantity_in_stock
        self.types = types
```

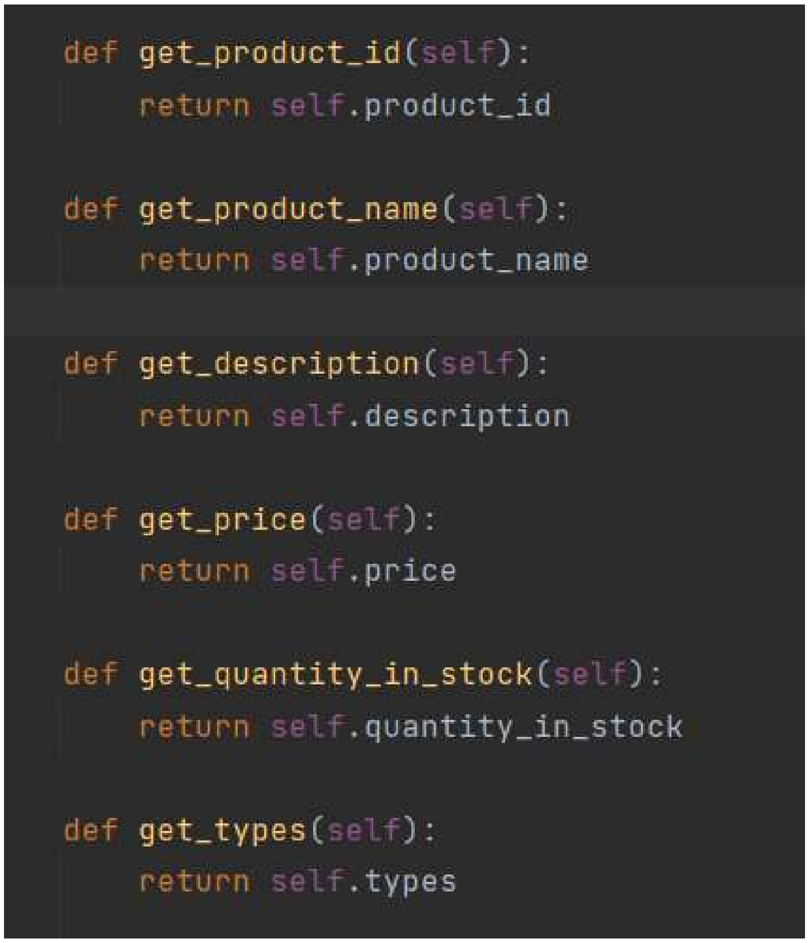


```
class Product:

    def __init__(self, product_id, product_name, description, price, quantity_in_stock, types):
        self.product_id = product_id
        self.product_name = product_name
        self.description = description
        self.price = price
        self.quantity_in_stock = quantity_in_stock
        self.types = types
```

*Implementing Getter and Setter Methods:*

***Getter Methods:***

```
def get_product_id(self):
    return self.product_id

def get_product_name(self):
    return self.product_name

def get_description(self):
    return self.description

def get_price(self):
    return self.price

def get_quantity_in_stock(self):
    return self.quantity_in_stock

def get_types(self):
    return self.types
```

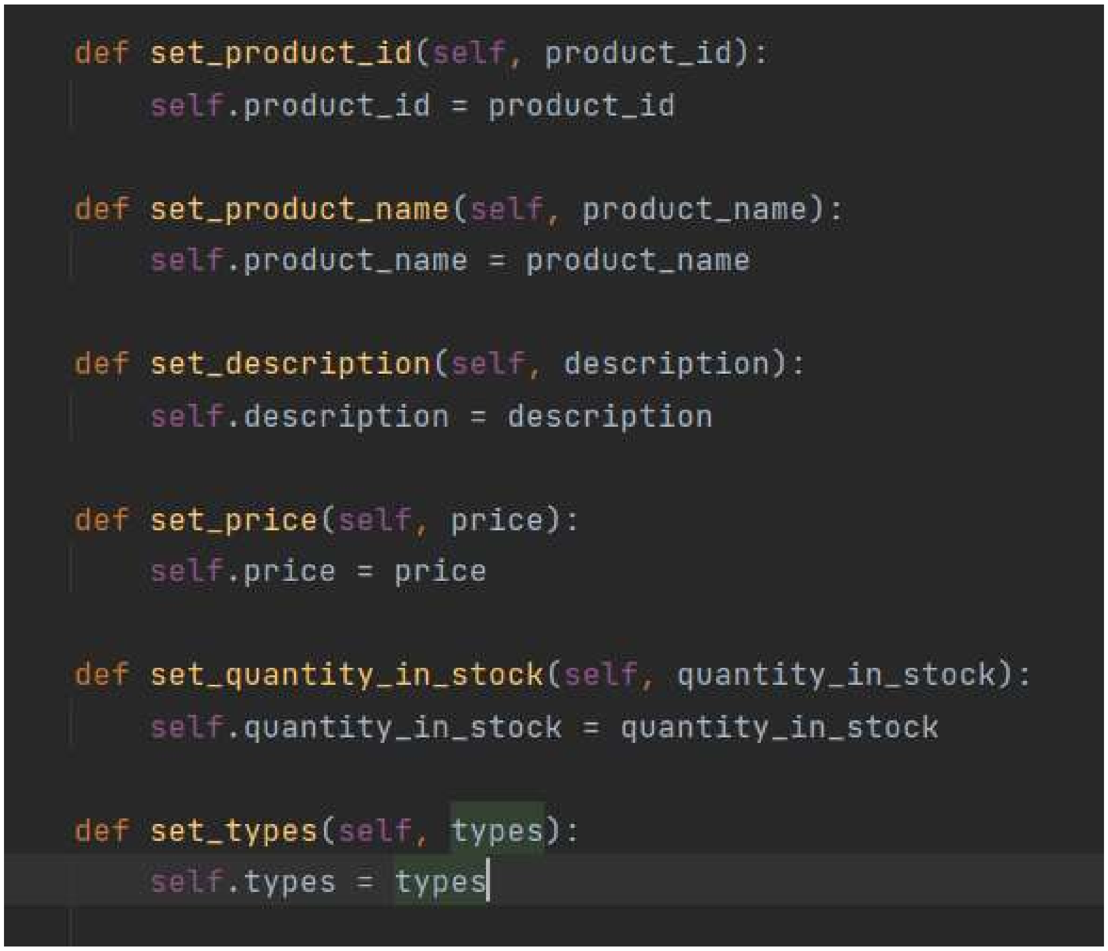


```
def get_product_id(self):
    return self.product_id

def get_product_name(self):
    return self.product_name

def get_description(self):
    return self.description

def get_price(self):
    return self.price

def get_quantity_in_stock(self):
    return self.quantity_in_stock

def get_types(self):
    return self.types
```

### *Setter Methods:*

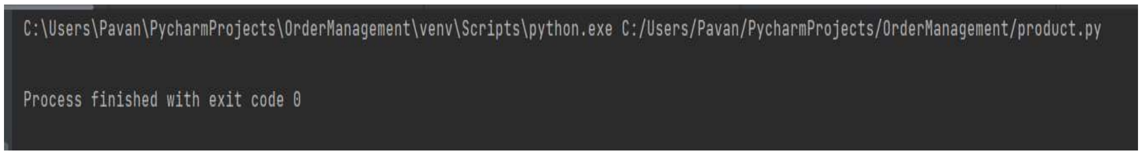
```
def set_product_id(self, product_id):
    self.product_id = product_id

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

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

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

```
def set_quantity_in_stock(self, quantity_in_stock):
    self.quantity_in_stock = quantity_in_stock

def set_types(self, types):
    self.types = types
```



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

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

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

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

def set_quantity_in_stock(self, quantity_in_stock):
    self.quantity_in_stock = quantity_in_stock

def set_types(self, types):
    self.types = types
```

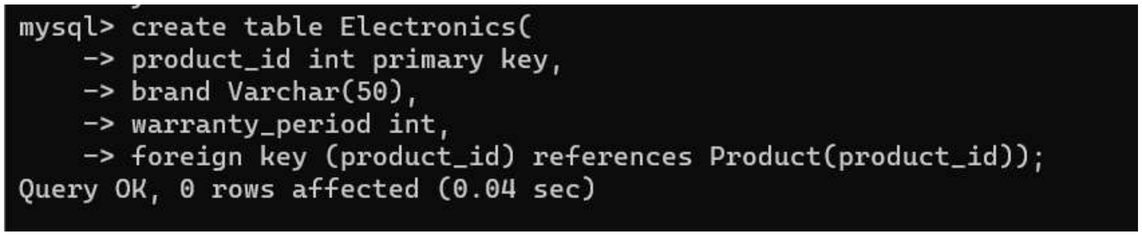
*Output:*



```
C:\Users\Pavan\PycharmProjects\OrderManagement\venv\Scripts\python.exe C:/Users/Pavan/PycharmProjects/OrderManagement/product.py

Process finished with exit code 0
```

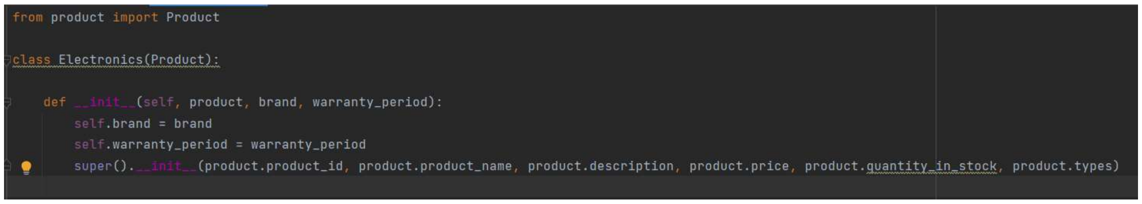
***3. Create a subclass Electronics that inherits from Product. Add attributes specific to electronics products, such as:***
***• brand (String)***
***• warrantyPeriod (int)***

*create table Electronics(*
*-> product_id int primary key,*
*-> brand Varchar(50),*
*-> warranty_period int,*
*-> foreign key (product_id) references Product(product_id));*



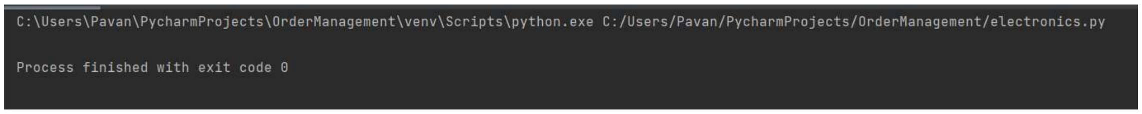
```
mysql> create table Electronics(
    -> product_id int primary key,
    -> brand Varchar(50),
    -> warranty_period int,
    -> foreign key (product_id) references Product(product_id));
Query OK, 0 rows affected (0.04 sec)
```

```
from product import Product

class Electronics(Product):

   def __init__(self, product, brand, warranty_period):
       self.brand = brand
       self.warranty_period = warranty_period
       super().__init__(product.product_id, product.product_name,
product.description, product.price, product.quantity_in_stock,
product.types)
```



```
from product import Product

class Electronics(Product):

    def __init__(self, product, brand, warranty_period):
        self.brand = brand
        self.warranty_period = warranty_period
        super().__init__(product.product_id, product.product_name, product.description, product.price, product.quantity_in_stock, product.types)
```

*Output:*

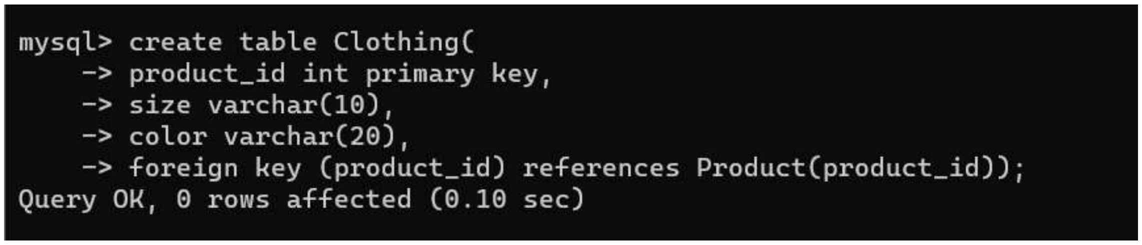


```
C:\Users\Pavan\PycharmProjects\OrderManagement\venv\Scripts\python.exe C:/Users/Pavan/PycharmProjects/OrderManagement/electronics.py

Process finished with exit code 0
```

***4. Create a subclass Clothing that also inherits from Product. Add attributes specific to clothing products, such as:***
- ***size (String)***
- ***color (String)***

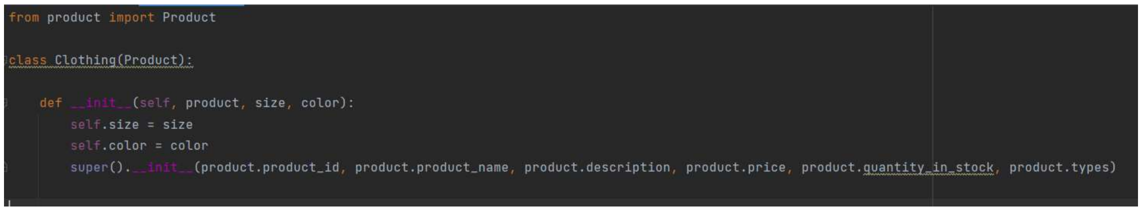
*create table Clothing(*
*-> product_id int primary key,*

```
    -> size varchar(10),
    -> color varchar(20),
    -> foreign key (product_id) references Product(product_id));
```



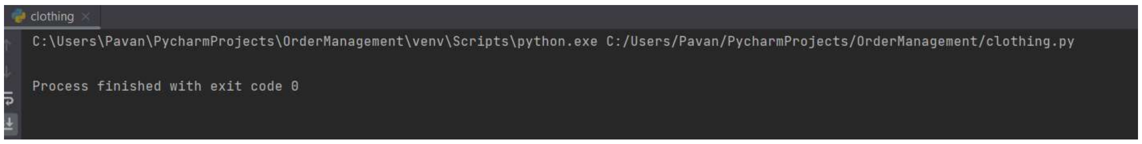
```
mysql> create table Clothing(
    -> product_id int primary key,
    -> size varchar(10),
    -> color varchar(20),
    -> foreign key (product_id) references Product(product_id));
Query OK, 0 rows affected (0.10 sec)
```

```python
from product import Product

class Clothing(Product):

    def __init__(self, product, size, color):
        self.size = size
        self.color = color
        super().__init__(product.product_id, product.product_name,
product.description, product.price, product.quantity_in_stock,
product.types)
```



```python
from product import Product

class Clothing(Product):

    def __init__(self, product, size, color):
        self.size = size
        self.color = color
        super().__init__(product.product_id, product.product_name, product.description, product.price, product.quantity_in_stock, product.types)
```

*Output:*

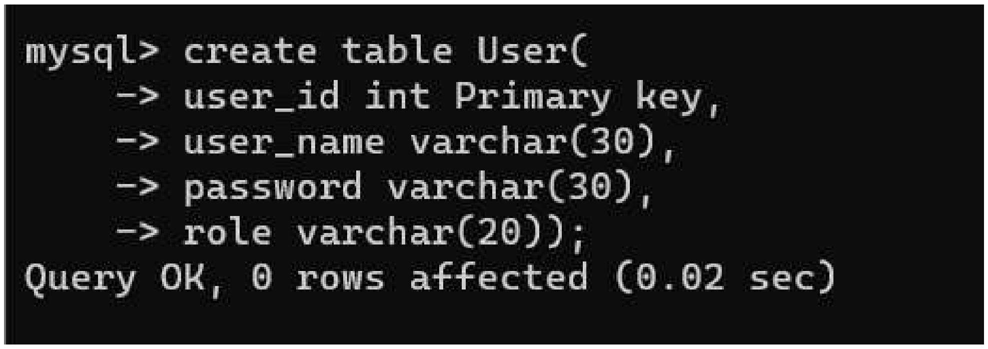


```
C:\Users\Pavan\PycharmProjects\OrderManagement\venv\Scripts\python.exe C:/Users/Pavan/PycharmProjects/OrderManagement/clothing.py

Process finished with exit code 0
```

***5. Create a User class with attributes:***
***• userId (int)***
***• username (String)***
***• password (String)***
***• role (String) // "Admin" or "User"***

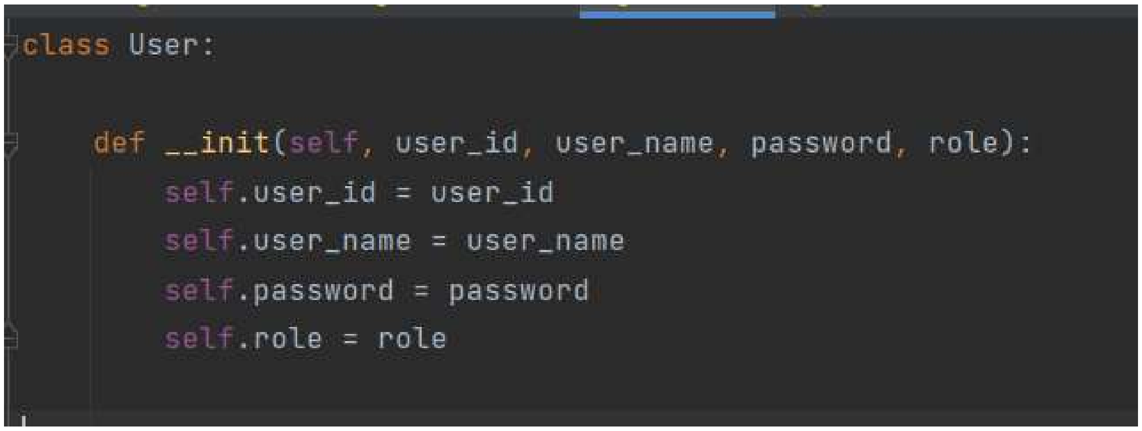
```
mysql> create table User(
    -> user_id int Primary key,
```

*-> user_name varchar(30),*
*-> password varchar(30),*
*-> role varchar(20));*



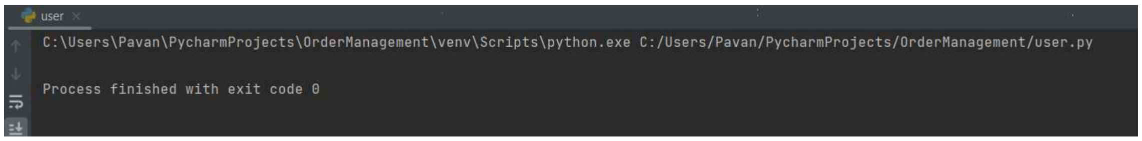
```
mysql> create table User(
    -> user_id int Primary key,
    -> user_name varchar(30),
    -> password varchar(30),
    -> role varchar(20));
Query OK, 0 rows affected (0.02 sec)
```

```
class User:

    def __init(self, user_id, user_name, password, role):
        self.user_id = user_id
        self.user_name = user_name
        self.password = password
        self.role = role
```



```
class User:

    def __init(self, user_id, user_name, password, role):
        self.user_id = user_id
        self.user_name = user_name
        self.password = password
        self.role = role
```

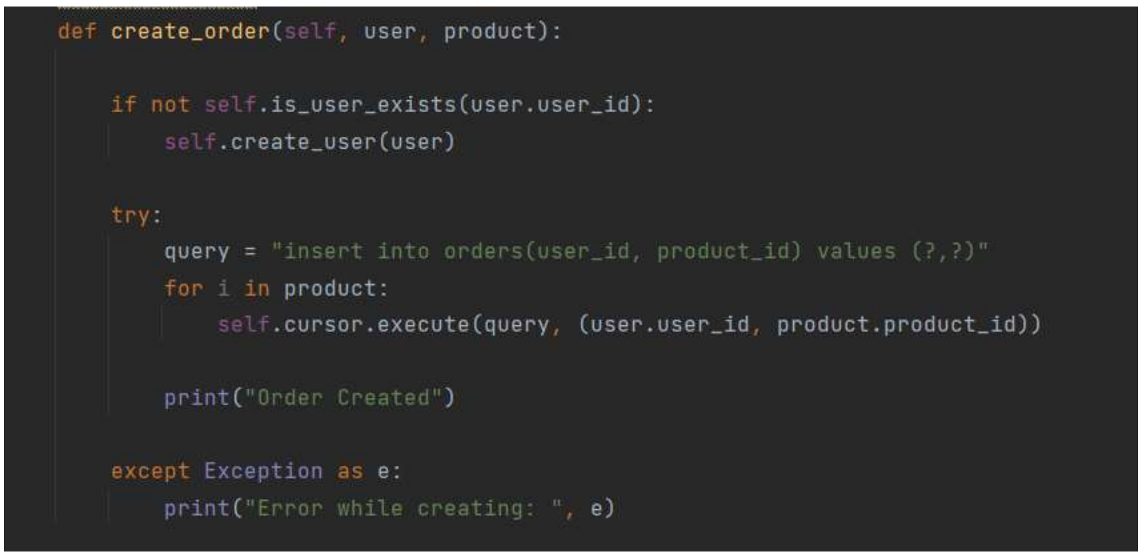
*Output:*



```
user
C:\Users\Pavan\PycharmProjects\OrderManagement\venv\Scripts\python.exe C:/Users/Pavan/PycharmProjects/OrderManagement/user.py

Process finished with exit code 0
```

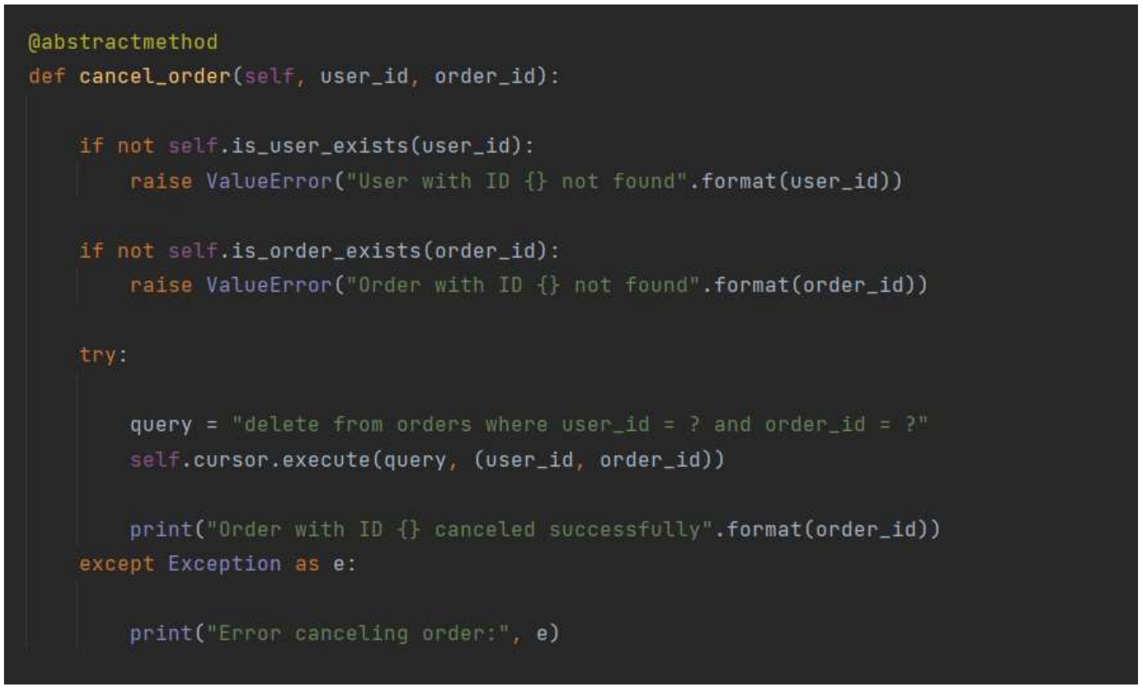
***6. Define an interface/abstract class named IOrderManagementRepository with methods for:***

*• 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.*



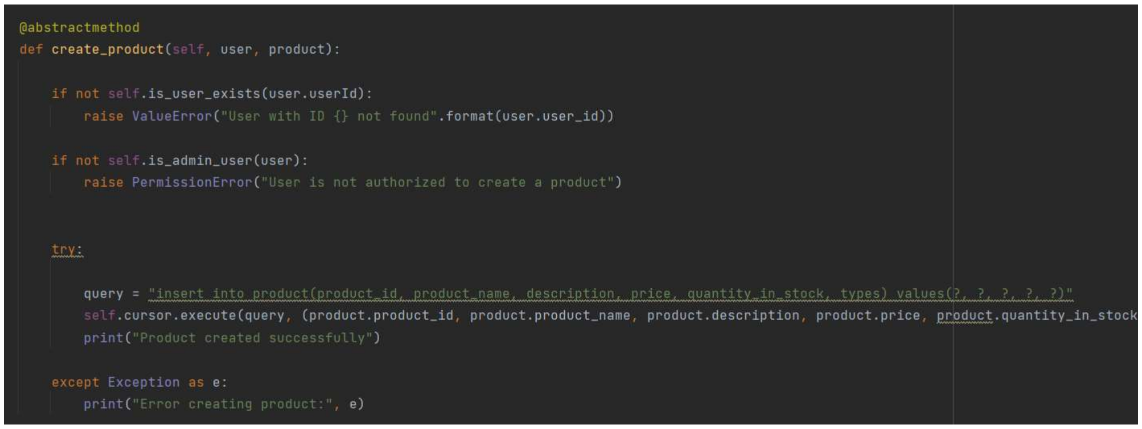
```python
    def create_order(self, user, product):

        if not self.is_user_exists(user.user_id):
            self.create_user(user)

        try:
            query = "insert into orders(user_id, product_id) values (?,?)"
            for i in product:
                self.cursor.execute(query, (user.user_id, product.product_id))

            print("Order Created")

        except Exception as e:
            print("Error while creating: ", e)
```

*• 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*



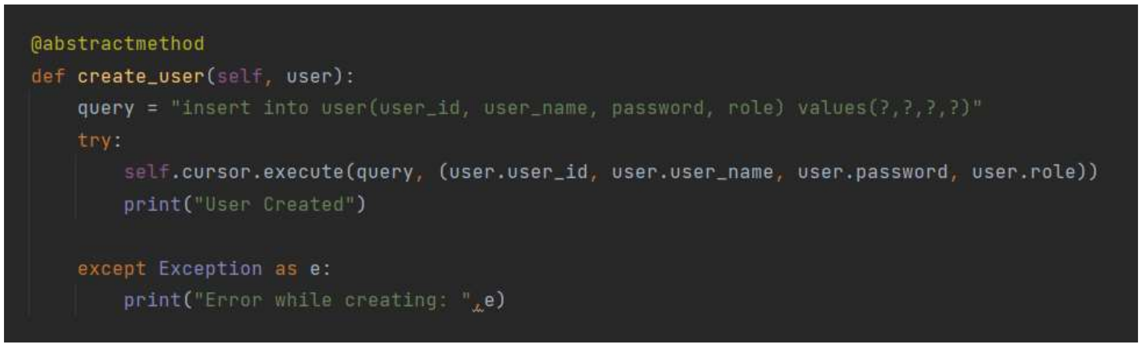
```python
@abstractmethod
def cancel_order(self, user_id, order_id):

    if not self.is_user_exists(user_id):
        raise ValueError("User with ID {} not found".format(user_id))

    if not self.is_order_exists(order_id):
        raise ValueError("Order with ID {} not found".format(order_id))

    try:

        query = "delete from orders where user_id = ? and order_id = ?"
        self.cursor.execute(query, (user_id, order_id))

        print("Order with ID {} canceled successfully".format(order_id))
    except Exception as e:

        print("Error canceling order:", e)
```

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



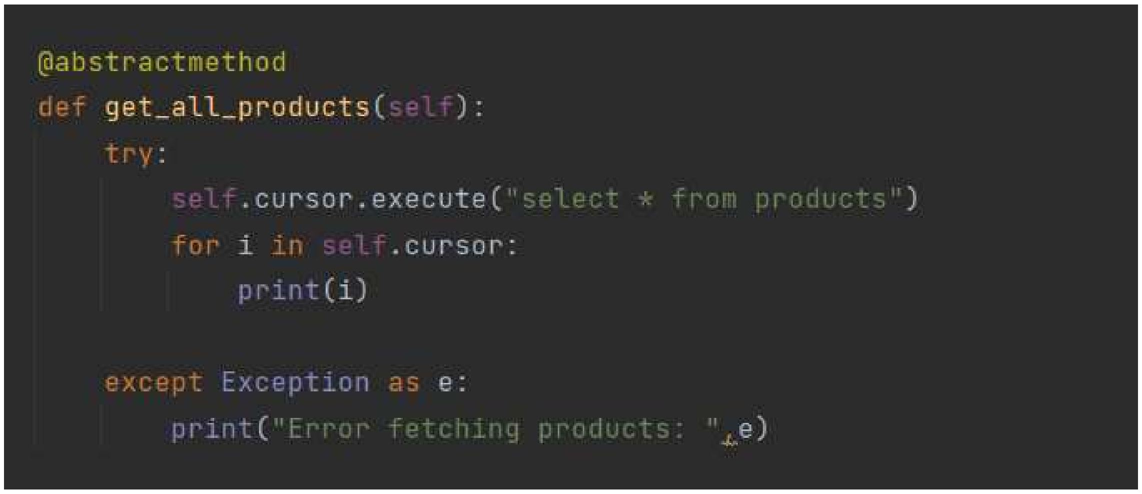
```
@abstractmethod
def create_product(self, user, product):

    if not self.is_user_exists(user.userId):
        raise ValueError("User with ID {} not found".format(user.user_id))

    if not self.is_admin_user(user):
        raise PermissionError("User is not authorized to create a product")


    try:

        query = "insert into product(product_id, product_name, description, price, quantity_in_stock, types) values(?, ?, ?, ?, ?)"
        self.cursor.execute(query, (product.product_id, product.product_name, product.description, product.price, product.quantity_in_stock
        print("Product created successfully")

    except Exception as e:
        print("Error creating product:", e)
```

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



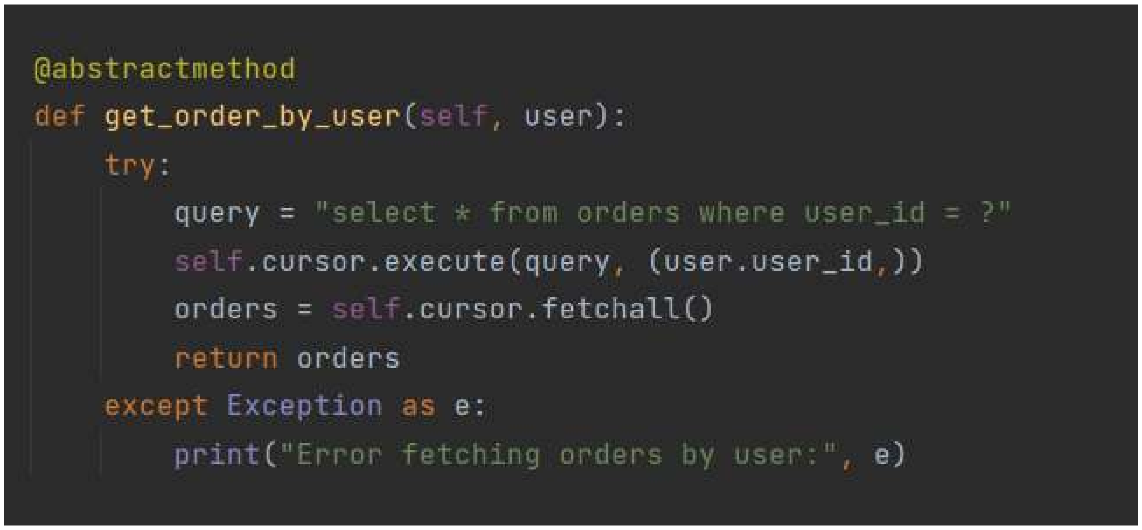
```
@abstractmethod
def create_user(self, user):
    query = "insert into user(user_id, user_name, password, role) values(?,?,?,?)"
    try:
        self.cursor.execute(query, (user.user_id, user.user_name, user.password, user.role))
        print("User Created")

    except Exception as e:
        print("Error while creating: ",e)
```

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



```
@abstractmethod
def get_all_products(self):
    try:
        self.cursor.execute("select * from products")
        for i in self.cursor:
            print(i)

    except Exception as e:
        print("Error fetching products: ",e)
```

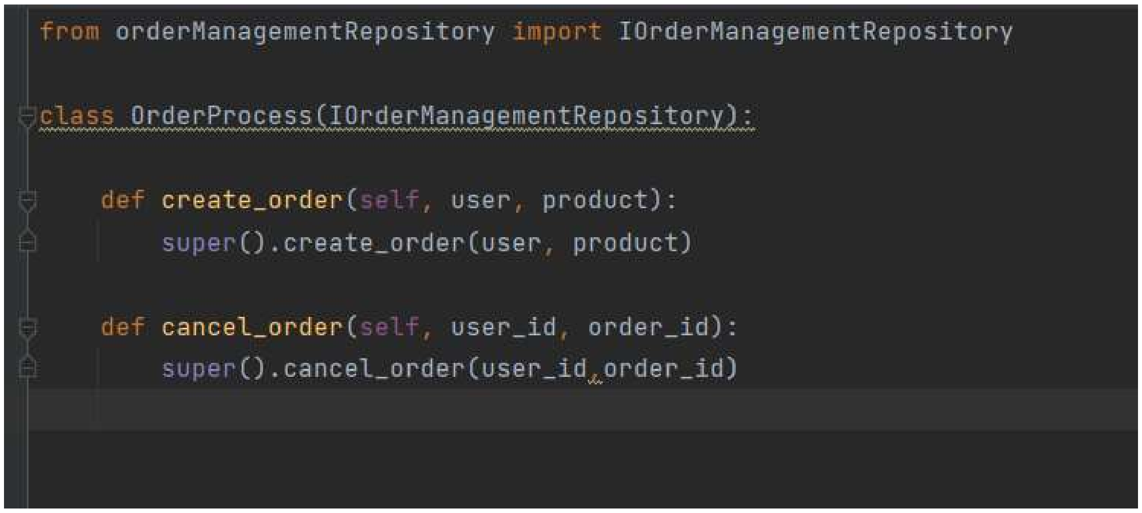
• *getOrderByUser(User user): return all product ordered by specific user from database.*



```
@abstractmethod
def get_order_by_user(self, user):
    try:
        query = "select * from orders where user_id = ?"
        self.cursor.execute(query, (user.user_id,))
        orders = self.cursor.fetchall()
        return orders
    except Exception as e:
        print("Error fetching orders by user:", e)
```

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

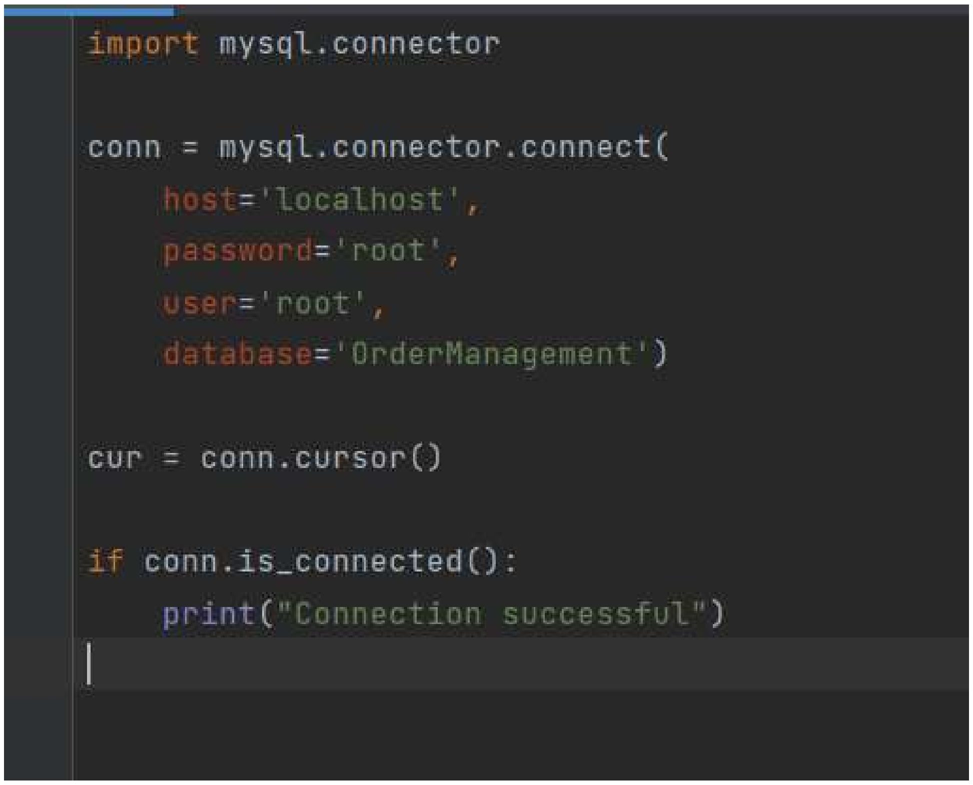
```
from orderManagementRepository import IOrderManagementRepository

class OrderProcess(IOrderManagementRepository):

   def create_order(self, user, product):
       super().create_order(user, product)

   def cancel_order(self, user_id, order_id):
       super().cancel_order(user_id,order_id)
```



```
from orderManagementRepository import IOrderManagementRepository

class OrderProcess(IOrderManagementRepository):

    def create_order(self, user, product):
        super().create_order(user, product)

    def cancel_order(self, user_id, order_id):
        super().cancel_order(user_id,order_id)
```

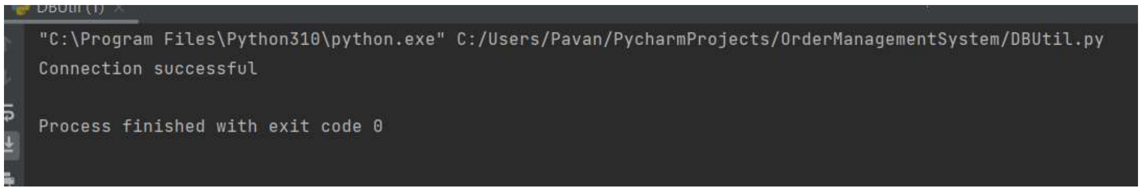
***8. Create DBUtil class and add the following method. • static getDBConn():Connection Establish a connection to the database and return database Connection***

```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    password='root',
    user='root',
    database='OrderManagement')

cur = conn.cursor()

if conn.is_connected():
    print("Connection successful")
```



```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    password='root',
    user='root',
    database='OrderManagement')

cur = conn.cursor()

if conn.is_connected():
    print("Connection successful")
```

*Output:*



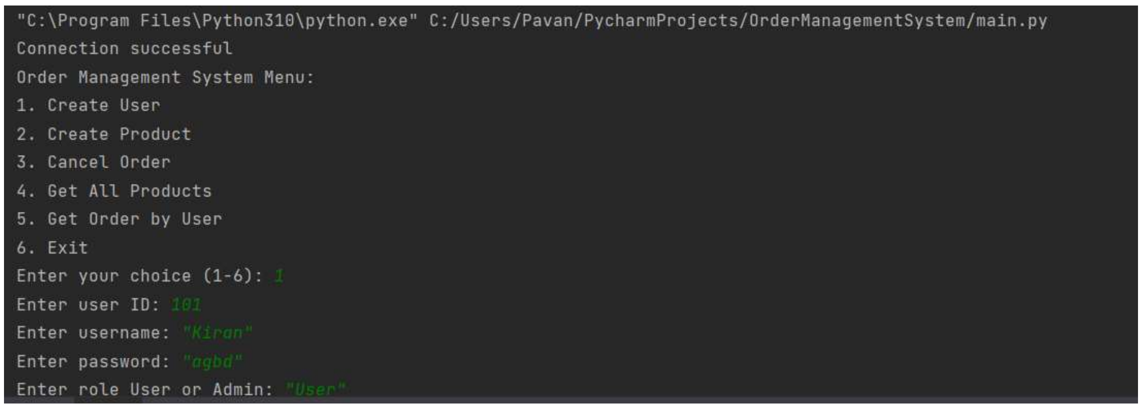
```
"C:\Program Files\Python310\python.exe" C:/Users/Pavan/PycharmProjects/OrderManagementSystem/DBUtil.py
Connection successful

Process finished with exit code 0
```

***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".***

```
from orderProcess import OrderProcess
from user import User
from product import Product

class OrderManagementMain:
    def __init__(self):
        pass

    def display_menu(self):
        print("Order Management System Menu:")
        print("1. Create User")
        print("2. Create Product")
        print("3. Cancel Order")
        print("4. Get All Products")
        print("5. Get Order by User")
        print("6. Exit")

    def run(self):
        while True:
            self.display_menu()
            choice = input("Enter your choice (1-6): ")
            if choice == '1':
                self.create_user()
            elif choice == '2':
                self.create_product()
            elif choice == '3':
                self.cancel_order()
            elif choice == '4':
                self.get_all_products()
            elif choice == '5':
                self.get_order_by_user()
            elif choice == '6':
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please enter a number from 1 to 6.")

    def create_user(self):

        user_id = input("Enter user ID: ")
        user_name = input("Enter username: ")
        password = input("Enter password: ")
        role = input("Enter role User or Admin: ")
        user = User(user_id, user_name, password, role)
        OrderProcess.create_user(user)

    def create_product(self):
```

```python
        user_id = input("Enter user ID: ")
        user_name = input("Enter username: ")
        password = input("Enter password: ")
        role = input("Enter role User or Admin: ")
        user = User(user_id, user_name, password, role)

        product_id = input("Enter product ID: ")
        product_name = input("Enter product_name: ")
        description = input("About the product: ")
        price = input("Price of the product: ")
        quantity = input("Number of items in stock: ")
        types = input("Electronic or Clothing: ")

        product = Product(product_id, product_name, description, price,
quantity, types)

        OrderProcess.create_product(user, product)

    def cancel_order(self):

        user_id = input("Enter User_id: ")
        order_id = input("Enter Order ID: ")
        OrderProcess.cancel_order(user_id, order_id)

    def get_all_products(self):
        OrderProcess.get_all_products()

    def get_order_by_user(self):
        user_id = input("Enter User ID: ")
        OrderProcess.get_order_by_user(user_id)


order_management_main = OrderManagementMain()

order_management_main.run()
```

*Output:*



```
"C:\Program Files\Python310\python.exe" C:/Users/Pavan/PycharmProjects/OrderManagementSystem/main.py
Connection successful
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit
Enter your choice (1-6): 1
Enter user ID: 101
Enter username: "Kiran"
Enter password: "agbd"
Enter role User or Admin: "User"
```