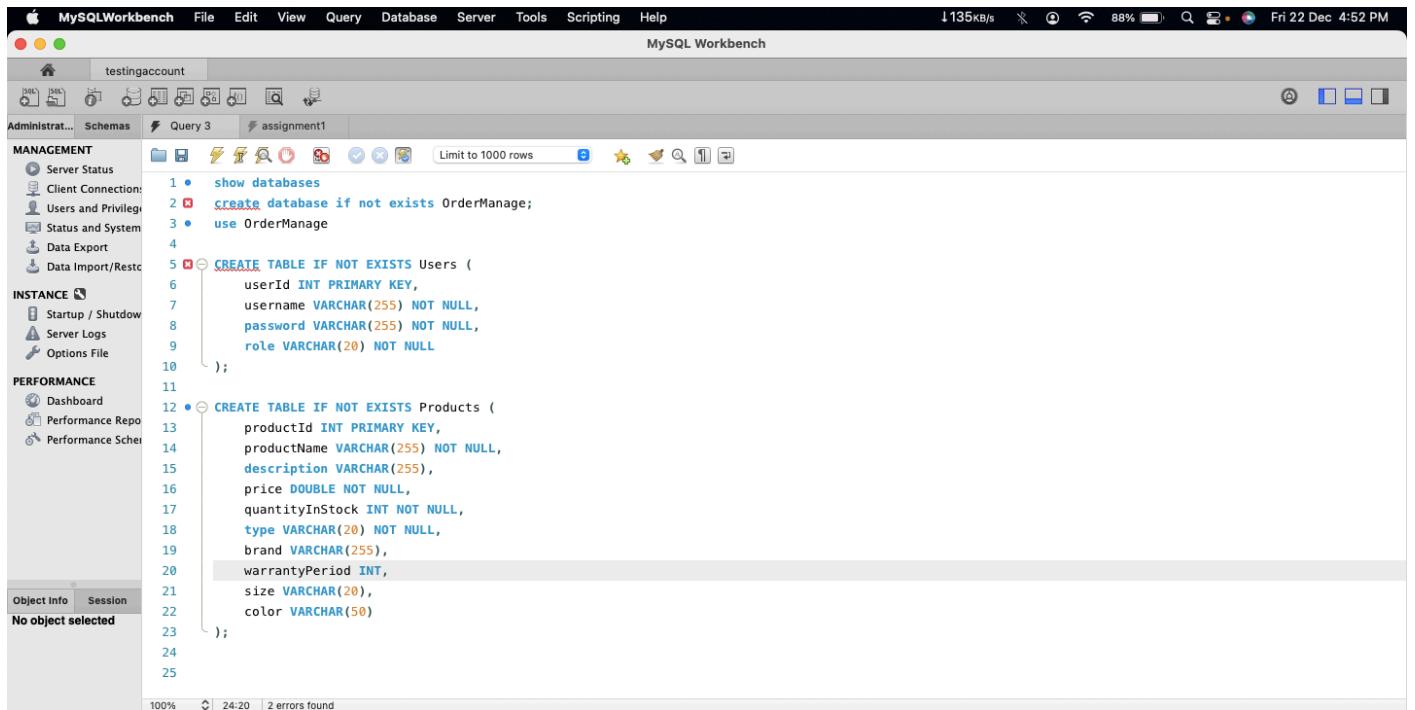


Coding Challenge - 6

Order Management System

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



The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a query editor window titled 'Query 3'. The code in the editor is as follows:

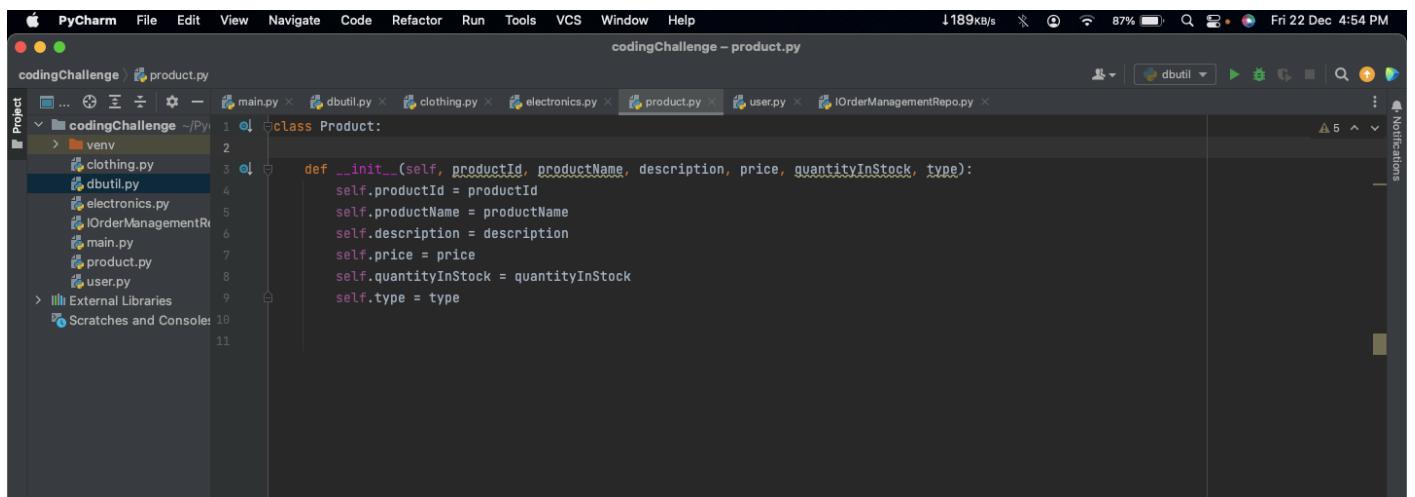
```
1 • show databases
2 • create database if not exists OrderManage;
3 • use OrderManage
4
5 • CREATE TABLE IF NOT EXISTS Users (
6     userId INT PRIMARY KEY,
7     username VARCHAR(255) NOT NULL,
8     password VARCHAR(255) NOT NULL,
9     role VARCHAR(20) NOT NULL
10 );
11
12 • CREATE TABLE IF NOT EXISTS Products (
13     productId INT PRIMARY KEY,
14     productName VARCHAR(255) NOT NULL,
15     description VARCHAR(255),
16     price DOUBLE NOT NULL,
17     quantityInStock INT NOT NULL,
18     type VARCHAR(20) NOT NULL,
19     brand VARCHAR(255),
20     warrantyPeriod INT,
21     size VARCHAR(20),
22     color VARCHAR(50)
23 );
24
25
```

The code creates a database named 'OrderManage' and two tables: 'Users' and 'Products'. The 'Users' table has columns for userId (primary key), username, password, and role. The 'Products' table has columns for productId (primary key), productName, description, price, quantityInStock, type, brand, warrantyPeriod, size, and color.

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]

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



The screenshot shows the PyCharm IDE with a project named 'codingChallenge'. The 'product.py' file is open in the editor. The code defines a class 'Product' with an __init__ method:

```
1 class Product:
2
3     def __init__(self, productId, productName, description, price, quantityInStock, type):
4         self.productId = productId
5         self.productName = productName
6         self.description = description
7         self.price = price
8         self.quantityInStock = quantityInStock
9         self.type = type
```

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

- **brand** (String)
- **warrantyPeriod** (int)

The screenshot shows the PyCharm interface with the project 'codingChallenge' open. The 'electronics.py' file is the active editor, displaying the following code:

```
from product import Product

class Electronics(Product):
    def __init__(self, productId, productName, description, price, quantityInStock, brand, warrantyPeriod):
        super().__init__(productId, productName, description, price, quantityInStock, type="electronics")
        self.brand = brand
        self.warrantyPeriod = warrantyPeriod
```

The 'Run' tab at the bottom shows the command run: `/Users/ajaychaudhary/PycharmProjects/codingChallenge/venv/bin/python /Users/ajaychaudhary/PycharmProjects/codingChallenge/electronics.py`, resulting in the message: `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)

The screenshot shows the PyCharm interface with the project 'codingChallenge' open. The 'clothing.py' file is the active editor, displaying the following code:

```
from product import Product

class Clothing(Product):
    def __init__(self, productId, productName, description, price, quantityInStock, size, color):
        super().__init__(productId, productName, description, price, quantityInStock, type="Clothing")
        self.size = size
        self.color = color
```

The 'Run' tab at the bottom shows the command run: `/Users/ajaychaudhary/PycharmProjects/codingChallenge/venv/bin/python /Users/ajaychaudhary/PycharmProjects/codingChallenge/clothing.py`, resulting in the message: `Process finished with exit code 0`.

5. Create a **User** class with attributes:

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

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "codingChallenge". Inside the "venv" directory, there are files: venv, clothing.py, dbutil.py, electronics.py, IOrderManagementRepo.py, main.py, product.py, and user.py.
- Edit Tab:** The file "user.py" is open, showing the following code:

```
class User:  
    def __init__(self, userId, username, password, role):  
        self.userId = userId  
        self.username = username  
        self.password = password  
        self.role = role
```
- Run Tab:** A run configuration for "clothing" is selected, showing the command: /Users/ajaychaudhary/PycharmProjects/codingChallenge/venv/bin/python /Users/ajaychaudhary/PycharmProjects/codingChallenge/clothing.py. The output shows: Process finished with exit code 0.
- Bottom Bar:** Shows the system tray with various application icons.

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

- **createOrder(User user, list of products):** check the user as already present in database to create order or create user (store in database) and create order.
- **cancelOrder(int userId, int orderId):** check the userid and orderId already present in database and cancel the order. if any userId or orderId not present in database throw exception corresponding **UserNotFoundException** or **OrderNotFoundException** exception
- **createProduct(User user, Product product):** check the admin user as already present in database and create product and store in database.
- **createUser(User user):** create user and store in database for further development.
- **getAllProducts():** return all product list from the database.
- **getOrderByUser(User user):** return all product ordered by specific user from database.

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help ↓200KB/s 77% Fri 22 Dec 5:06 PM

codingChallenge ~Py/ codingChallenge - IOrderManagementRepo.py

Project

codingChallenge

- venv
- clothing.py
- dbutil.py
- electronics.py
- IOrderManagementRe...
- main.py
- product.py
- user.py

External Libraries

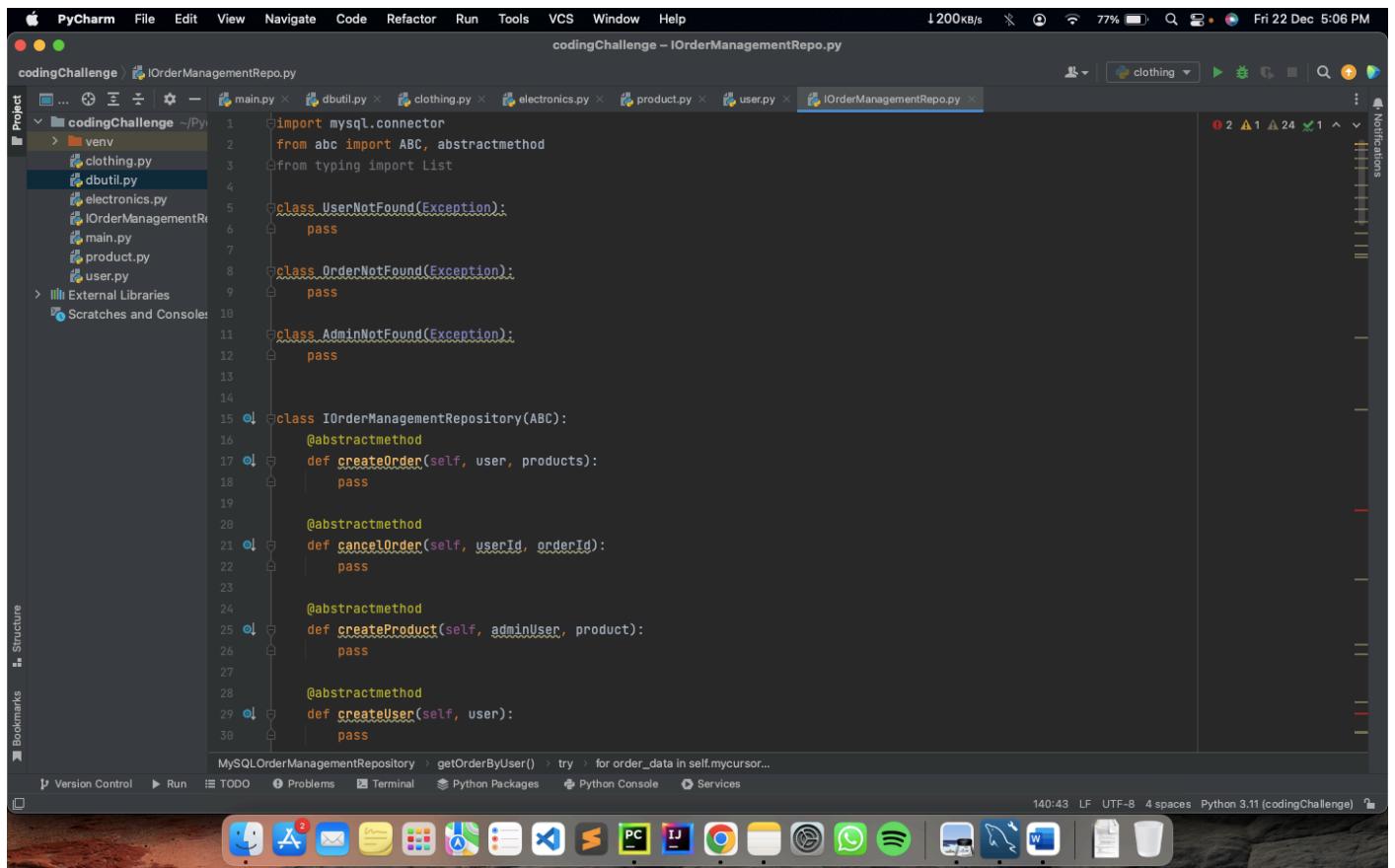
Scratches and Consoles

```
1 import mysql.connector
2 from abc import ABC, abstractmethod
3 from typing import List
4
5 class UserNotFoundException(Exception):
6     pass
7
8 class OrderNotFoundException(Exception):
9     pass
10
11 class AdminNotFoundException(Exception):
12     pass
13
14
15 class IOrderManagementRepository(ABC):
16     @abstractmethod
17     def createOrder(self, user, products):
18         pass
19
20     @abstractmethod
21     def cancelOrder(self, userId, orderId):
22         pass
23
24     @abstractmethod
25     def createProduct(self, adminUser, product):
26         pass
27
28     @abstractmethod
29     def createUser(self, user):
30         pass
31
32     @abstractmethod
33     def getAllProducts(self):
34         pass
35
36     @abstractmethod
37     def getOrderByUser(self, user):
38         pass
39
```

MySQLOrderManagementRepository > getOrderByUser() > try > for order_data in self.mycursor...

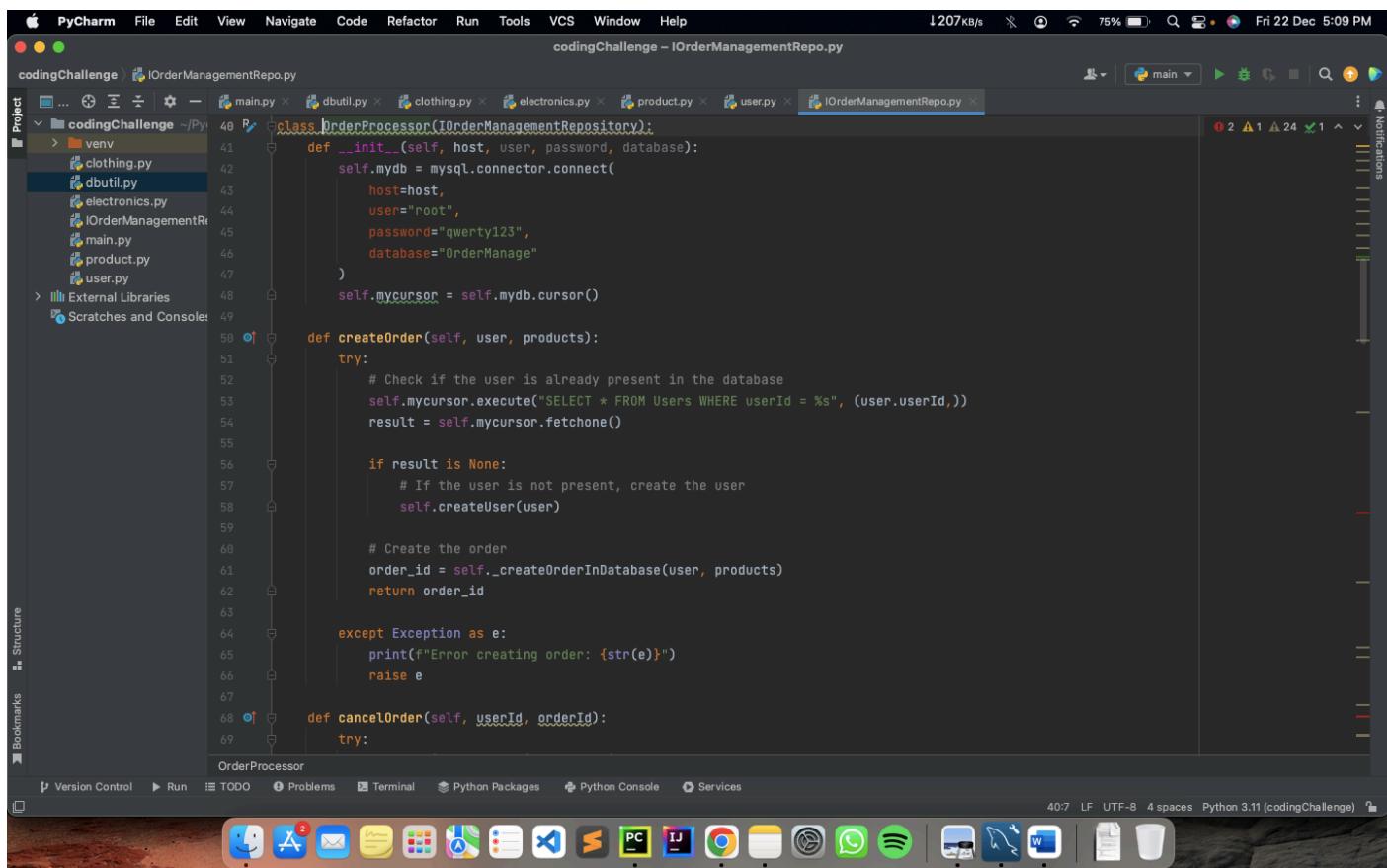
Version Control Run TODO Problems Terminal Python Packages Python Console Services

140:43 LF UTF-8 4 spaces Python 3.11 (codingChallenge)



```
28     @abstractmethod
29     def createUser(self, user):
30         pass
31
32     @abstractmethod
33     def getAllProducts(self):
34         pass
35
36     @abstractmethod
37     def getOrderByUser(self, user):
38         pass
39
```

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



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** codingChallenge
- File:** IOrderManagementRepo.py
- Code:** Implementation of the OrderProcessor class, which implements the IOrderManagementRepository interface. The code handles creating and canceling orders.

```
class OrderProcessor(IOrderManagementRepository):
    def __init__(self, host, user, password, database):
        self.mydb = mysql.connector.connect(
            host=host,
            user="root",
            password="qwerty123",
            database="OrderManage"
        )
        self.mycursor = self.mydb.cursor()

    def createOrder(self, user, products):
        try:
            # Check if the user is already present in the database
            self.mycursor.execute("SELECT * FROM Users WHERE userId = %s", (user.userId,))
            result = self.mycursor.fetchone()

            if result is None:
                # If the user is not present, create the user
                self.createUser(user)

            # Create the order
            order_id = self._createOrderInDatabase(user, products)
            return order_id

        except Exception as e:
            print(f"Error creating order: {str(e)}")
            raise e

    def cancelOrder(self, userId, orderId):
        try:
            # Check if the user is present in the database
            self.mycursor.execute("SELECT * FROM Users WHERE userId = %s", (userId,))
            user_result = self.mycursor.fetchone()

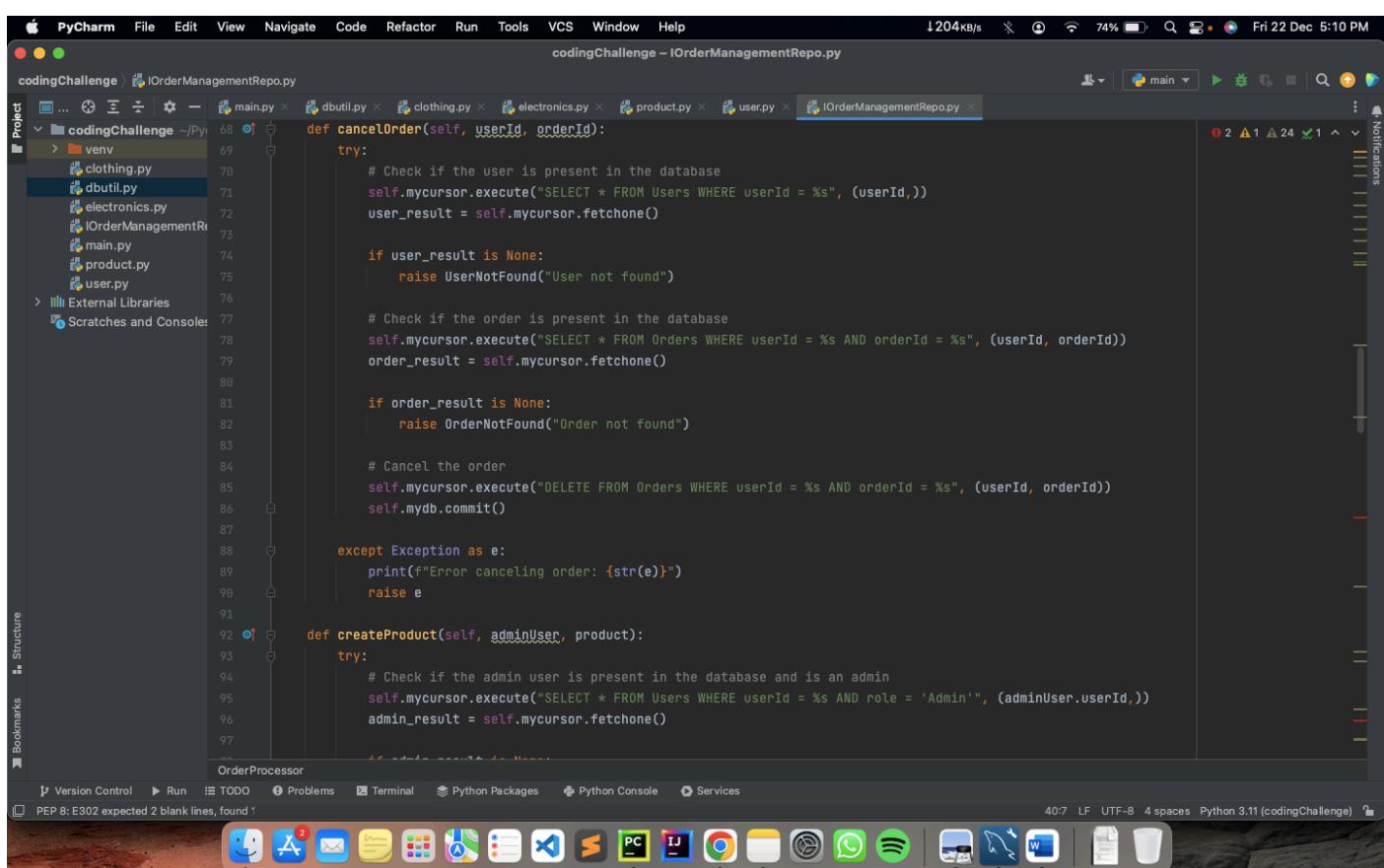
            if user_result is None:
                raise UserNotFound("User not found")

            # Check if the order is present in the database
            self.mycursor.execute("SELECT * FROM Orders WHERE userId = %s AND orderId = %s", (userId, orderId))
            order_result = self.mycursor.fetchone()

            if order_result is None:
                raise OrderNotFound("Order not found")

            # Cancel the order
            self.mycursor.execute("DELETE FROM Orders WHERE userId = %s AND orderId = %s", (userId, orderId))
            self.mydb.commit()

        except Exception as e:
            print(f"Error canceling order: {str(e)}")
            raise e
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** codingChallenge
- File:** IOrderManagementRepo.py
- Code:** Implementation of the OrderProcessor class, continuing from the previous screenshot. It includes a method for creating products.

```
def cancelOrder(self, userId, orderId):
    try:
        # Check if the user is present in the database
        self.mycursor.execute("SELECT * FROM Users WHERE userId = %s", (userId,))
        user_result = self.mycursor.fetchone()

        if user_result is None:
            raise UserNotFound("User not found")

        # Check if the order is present in the database
        self.mycursor.execute("SELECT * FROM Orders WHERE userId = %s AND orderId = %s", (userId, orderId))
        order_result = self.mycursor.fetchone()

        if order_result is None:
            raise OrderNotFound("Order not found")

        # Cancel the order
        self.mycursor.execute("DELETE FROM Orders WHERE userId = %s AND orderId = %s", (userId, orderId))
        self.mydb.commit()

    except Exception as e:
        print(f"Error canceling order: {str(e)}")
        raise e

    def createProduct(self, adminUser, product):
        try:
            # Check if the admin user is present in the database and is an admin
            self.mycursor.execute("SELECT * FROM Users WHERE userId = %s AND role = 'Admin'", (adminUser.userId,))
            admin_result = self.mycursor.fetchone()
```

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge - IOrderManagementRepo.py

```
def createProduct(self, adminUser, product):
    try:
        # Check if the admin user is present in the database and is an admin
        self.mycursor.execute("SELECT * FROM Users WHERE userId = %s AND role = 'Admin'", (adminUser.userId,))
        admin_result = self.mycursor.fetchone()

        if admin_result is None:
            raise AdminNotFound("Admin not found")

        # Create the product
        product_id = self._createProductInDatabase(product)
        return product_id

    except Exception as e:
        print(f"Error creating product: {str(e)}")
        raise e

def createUser(self, user):
    try:
        # Create the user
        self.mycursor.execute("INSERT INTO Users (userId, username, password, role) VALUES (%s, %s, %s, %s)",
                             (user.userId, user.username, user.password, user.role))
        self.mydb.commit()

    except Exception as e:
        print(f"Error creating user: {str(e)}")
        raise e

def getAllProducts(self):
    try:
```

Version Control Run TODO Problems Terminal Python Packages Python Console Services

PEP 8: E302 expected 2 blank lines, found 1

40:7 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge - IOrderManagementRepo.py

```
def getAllProducts(self):
    try:
        # Retrieve all products from the database
        self.mycursor.execute("SELECT * FROM Products")
        products = []
        for product_data in self.mycursor.fetchall():
            product = Product(product_data)
            products.append(product)
        return products

    except Exception as e:
        print(f"Error getting all products: {str(e)}")
        raise e

def getOrderByUser(self, user):
    try:
        # Retrieve all products ordered by the specific user from the database
        self.mycursor.execute("SELECT * FROM Orders WHERE userId = %s", (user.userId,))
        order_products = []
        for order_data in self.mycursor.fetchall():
            product_id = order_data[2]
            product = self._getProductById(product_id)
            order_products.append(product)
        return order_products

    except Exception as e:
        print(f"Error getting order by user: {str(e)}")
        raise e

def _createOrderInDatabase(self, user, products):
```

Version Control Run TODO Problems Terminal Python Packages Python Console Services

PEP 8: E302 expected 2 blank lines, found 1

40:7 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge - IOrderManagementRepo.py

```
codingChallenge > IOrderManagementRepo.py
Project  main.py dbutil.py clothing.py electronics.py product.py user.py IOrderManagementRepo.py
  codingChallenge ~/Py... 149
    venv
    > venv
    dbutil.py
    clothing.py
    electronics.py
    OrderManagementRe...
    main.py
    product.py
    user.py
  External Libraries
  Scratches and Consoles

def _createOrderInDatabase(self, user, products):
    try:
        # Create the order
        self.mycursor.execute("INSERT INTO Orders (userId) VALUES (%s)", (user.userId,))
        self.mydb.commit()

        # Retrieve the order ID
        self.mycursor.execute("SELECT LAST_INSERT_ID()")
        order_id = self.mycursor.fetchone()[0]

        # Add products to the order
        for product in products:
            self.mycursor.execute("INSERT INTO OrderProducts (orderId, productId) VALUES (%s, %s)",
                                  (order_id, product.productId))
            self.mydb.commit()

    return order_id

except Exception as e:
    print(f"Error creating order in the database: {str(e)}")
    raise e

def _createProductInDatabase(self, product):
    try:
        # Create the product
        self.mycursor.execute("INSERT INTO Products (productName, description, price, quantityInStock, type, brand, warrantyPeriod, size, color) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)", (product.productName, product.description, product.price, product.quantityInStock, product.type, product.brand, product.warrantyPeriod, product.size, product.color))
        self.mydb.commit()

    except Exception as e:
        print(f"Error creating product in the database: {str(e)}")
        raise e

    return product_id

def _getProductById(self, product_id):
    try:
        self.mycursor.execute("SELECT * FROM Products WHERE productId = %s", (product_id,))
        product_data = self.mycursor.fetchone()
        product = Product(*product_data)
        return product

    except Exception as e:
        print(f"Error getting product by ID: {str(e)}")
        raise e
```

OrderProcessor > _createOrderInDatabase() > try > for product in products

Version Control Run TODO Problems Terminal Python Packages Python Console Services

163:35 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge - IOrderManagementRepo.py

```
codingChallenge > IOrderManagementRepo.py
Project  main.py dbutil.py clothing.py electronics.py product.py user.py IOrderManagementRepo.py
  codingChallenge ~/Py... 177
    venv
    > venv
    dbutil.py
    clothing.py
    electronics.py
    OrderManagementRe...
    main.py
    product.py
    user.py
  External Libraries
  Scratches and Consoles

    self.mydb.commit()

    # Retrieve the product ID
    self.mycursor.execute("SELECT LAST_INSERT_ID()")
    product_id = self.mycursor.fetchone()[0]

    return product_id

except Exception as e:
    print(f"Error creating product in the database: {str(e)}")
    raise e

def _getProductById(self, product_id):
    try:
        self.mycursor.execute("SELECT * FROM Products WHERE productId = %s", (product_id,))
        product_data = self.mycursor.fetchone()
        product = Product(*product_data)
        return product

    except Exception as e:
        print(f"Error getting product by ID: {str(e)}")
        raise e
```

OrderProcessor > _createOrderInDatabase() > try > for product in products

Version Control Run TODO Problems Terminal Python Packages Python Console Services

163:35 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

8. Create **DBUtil** class and add the following method.

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

The screenshot shows the PyCharm IDE interface with the following details:

- File Path:** codingChallenge - dbutil.py
- Code Content:** The code defines a static method `getDBConn` that connects to a MySQL database using the `mysql.connector` module. It handles exceptions and prints a success message if the connection is established.
- Project Structure:** The project structure shows files like main.py, dbutil.py, clothing.py, electronics.py, product.py, user.py, and IOrderManagementRepo.py.
- Toolbars and Status Bar:** The bottom of the screen shows various toolbars and a status bar indicating the file has 1209KBs, is at 70% battery, and was recorded on Fri 22 Dec 5:15 PM.

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".

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help Fri 22 Dec 5:17 PM 1201kB/s 68% 🔍 Fri 22 Dec 5:17 PM

codingChallenge - main.py

Project codingChallenge ~/Py... main.py dbutil.py clothing.py electronics.py product.py user.py IOrderManagementRepo.py

```
from product import Product
from electronics import Electronics
from clothing import Clothing
from user import User
from IOrderManagementRepo import IOrderManagementRepository
from IOrderManagementRepo import OrderProcessor

class OrderManagement:
    def __init__(self):
        self.order_management_repository = OrderProcessor(
            host="localhost",
            user="root",
            password="qwertyst123",
            database="OrderManage"
        )

    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 Orders by User")
        print("6. Exit")

    def run(self):
        while True:
            self.display_menu()
            choice = input("Enter your choice: ")

            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_orders_by_user()
            elif choice == "6":
                print("Exiting Order Management System. Goodbye!")
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    def create_user(self):
        user_id = int(input("Enter User ID: "))
        username = input("Enter Username: ")
        password = input("Enter Password: ")
        role = input("Enter Role (Admin/User): ")

        user = User(userId=user_id, username=username, password=password, role=role)
        self.order_management_repository.createUser(user)
        print("User created successfully.")
```

External Libraries Scratches and Consoles

Structure Bookmarks

OrderManagement > run() > while True

Version Control Run TODO Problems Terminal Python Packages Python Console Services

30:1 LF UTF-8 4 spaces Python 3.11 (codingChallenge) 🔍

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help Fri 22 Dec 5:17 PM 1205kB/s 68% 🔍 Fri 22 Dec 5:17 PM

codingChallenge - main.py

Project codingChallenge ~/Py... main.py dbutil.py clothing.py electronics.py product.py user.py IOrderManagementRepo.py

```
def run(self):
    while True:
        self.display_menu()
        choice = input("Enter your choice: ")

        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_orders_by_user()
        elif choice == "6":
            print("Exiting Order Management System. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a valid option.")

    def create_user(self):
        user_id = int(input("Enter User ID: "))
        username = input("Enter Username: ")
        password = input("Enter Password: ")
        role = input("Enter Role (Admin/User): ")

        user = User(userId=user_id, username=username, password=password, role=role)
        self.order_management_repository.createUser(user)
        print("User created successfully.")
```

External Libraries Scratches and Consoles

Structure Bookmarks

OrderManagement > run() > while True

Version Control Run TODO Problems Terminal Python Packages Python Console Services

30:1 LF UTF-8 4 spaces Python 3.11 (codingChallenge) 🔍

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge – main.py

Project codingChallenge ~/Py codingChallenge venv dbutil.py clothing.py electronics.py OrderManagementRe main.py product.py user.py External Libraries Scratches and Consoles

```
def create_product(self):
    product_id = int(input("Enter Product ID: "))
    product_name = input("Enter Product Name: ")
    description = input("Enter Product Description: ")
    price = float(input("Enter Product Price: "))
    quantity_in_stock = int(input("Enter Quantity in Stock: "))
    product_type = input("Enter Product Type (Electronics/Clothing): ")

    if product_type == "Electronics":
        brand = input("Enter Brand: ")
        warranty_period = int(input("Enter Warranty Period (in months): "))
        product = Product(productId=product_id, productName=product_name, description=description,
                           price=price, quantityInStock=quantity_in_stock, product_type=product_type,
                           brand=brand, warrantyPeriod=warranty_period)
    elif product_type == "Clothing":
        size = input("Enter Size: ")
        color = input("Enter Color: ")
        product = Product(productId=product_id, productName=product_name, description=description,
                           price=price, quantityInStock=quantity_in_stock, product_type=product_type,
                           size=size, color=color)
    else:
        print("Invalid product type. Please enter 'Electronics' or 'Clothing'.")
        return

    self.order_management_repository.createProduct(user, product)
    print("Product created successfully.")

def cancel_order(self):
    user_id = int(input("Enter User ID: "))
    order_id = int(input("Enter Order ID: "))

    try:
        self.order_management_repository.cancelOrder(user_id, order_id)
        print("Order canceled successfully.")
    except UserNotFound:
        print("User not found.")
    except OrderNotFound:
        print("Order not found.")

def get_all_products(self):
    products = self.order_management_repository.getAllProducts()
    print("All Products:")
    for product in products:
        print(f"Product ID: {product.productId}, Product Name: {product.productName}")

def get_orders_by_user(self):
    user_id = int(input("Enter User ID: "))
    user = User(userId=user_id, username="", password="", role="")
    orders = self.order_management_repository.getOrderByUser(user)
    print("Orders by User:")
    for order in orders:
        print(f"Product ID: {order.productId}, Product Name: {order.productName}")

if __name__ == "__main__":
    order_management_system = OrderManagement()
    order_management_system.run()
```

Version Control Run TODO Problems Terminal Python Packages Python Console Services

30:1 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help

codingChallenge – main.py

Project codingChallenge ~/Py codingChallenge venv dbutil.py clothing.py electronics.py OrderManagementRe main.py product.py user.py External Libraries Scratches and Consoles

```
def cancel_order(self):
    user_id = int(input("Enter User ID: "))
    order_id = int(input("Enter Order ID: "))

    try:
        self.order_management_repository.cancelOrder(user_id, order_id)
        print("Order canceled successfully.")
    except UserNotFound:
        print("User not found.")
    except OrderNotFound:
        print("Order not found.")

def get_all_products(self):
    products = self.order_management_repository.getAllProducts()
    print("All Products:")
    for product in products:
        print(f"Product ID: {product.productId}, Product Name: {product.productName}")

def get_orders_by_user(self):
    user_id = int(input("Enter User ID: "))
    user = User(userId=user_id, username="", password="", role="")
    orders = self.order_management_repository.getOrderByUser(user)
    print("Orders by User:")
    for order in orders:
        print(f"Product ID: {order.productId}, Product Name: {order.productName}")

if __name__ == "__main__":
    order_management_system = OrderManagement()
    order_management_system.run()
```

Version Control Run TODO Problems Terminal Python Packages Python Console Services

30:1 LF UTF-8 4 spaces Python 3.11 (codingChallenge)

Outputs:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes files like `main.py`, `dbutil.py`, `clothing.py`, `electronics.py`, `product.py`, and `user.py`. The code editor displays `main.py` with the following content:

```
from product import Product
from electronics import Electronics
from clothing import Clothing
from user import User
from IOrderManagementRepo import IOrderManagementRepository
from IOrderManagementRepo import OrderProcessor

class OrderManagement:
    def __init__(self):
        self.order_management_repository = OrderProcessor(
            host="localhost",
            user="root",
            password="qwerty123",
            database="OrderManage"
        )

    def display_menu(self):
        print("Order Management System Menu:")

OrderManagement
```

The run output window shows the program's execution:

```
Run: main x
/Users/ajaychaudhary/PycharmProjects/codingChallenge/venv/bin/python /Users/ajaychaudhary/PycharmProjects/codingChallenge/main.py
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice:
```

The status bar at the bottom indicates the file is `main.py` and the Python version is `Python 3.11 (codingChallenge)`.

The screenshot shows the PyCharm IDE interface. The run output window shows the program's execution for creating a new user:

```
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice: 1
Enter User ID: 01
Enter Username: user_01
Enter Password: user_pass@123
Enter Role (Admin/User): admin
User created successfully.
```

The status bar at the bottom indicates the file is `main.py` and the Python version is `Python 3.11 (codingChallenge)`.

The screenshot shows the PyCharm IDE interface. The run output window shows the program's execution for creating a new product:

```
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice: 1
Enter User ID: 02
Enter Username: user_02
Enter Password: user_pass002
Enter Role (Admin/User): user
User created successfully.
```

The screenshot shows the PyCharm IDE interface. The run output window shows the program's execution for creating a new product, continuing from the previous step:

```
Enter your choice: 2
Enter Product ID: 001
Enter Product Name: laptop
Enter Product Description: m1 macbook air
Enter Product Price: 149.99
Enter Quantity in Stock: 50
Enter Product Type (Electronics/Clothing): Electronics
Enter Brand: Apple
Enter Warranty Period (in months): 12
```

The screenshot shows a PyCharm IDE interface with a terminal window open. The terminal displays a menu system for an Order Management System:

```
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice: 6
Exiting Order Management System. Goodbye!
```

Below the terminal, the status bar shows "Process finished with exit code 0".

The screenshot shows a terminal window with the following command and output:

```
/Users/ajaychaudhary/PycharmProjects/codingChallenge/venv/bin/python /Users/ajaychaudhary/PycharmProjects/codingChallenge/main.py
```

The output of the script is:

```
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Orders by User
6. Exit
Enter your choice: 5
Enter User ID: 01
Orders by User:
```