

# CODING CHALLENGE:

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

- Project:** OMS
- File:** db\_util.py
- Code Content:**

```
1 | import mysql.connector
2 |
3 |
4 | class DBUtil:
5 |     @staticmethod
6 |     def getDBConn():
7 |         connection = mysql.connector.connect(
8 |             host="localhost",
9 |             user="root",
10 |             password="root",
11 |             database="oms"
12 |         )
13 |         return connection
14 |
15 | class Product:
16 |     def __init__(self, productId, productName, description, price, quantityInStock, type):
17 |         self.productId = productId
18 |         self.productName = productName
19 |         self.description = description
20 |         self.price = price
21 |         self.quantityInStock = quantityInStock
22 |         self.type = type
23 |
24 | class Electronics(Product):
25 |
26 |
27 |
28 |
29 |
30 | class Clothing(Product):
31 |     def __init__(self, productId, productName, description, price, quantityInStock, type):
32 |         super().__init__(productId, productName, description, price, quantityInStock, type)
33 |         self.size = size
34 |         self.color = color
35 |
36 | class User:
37 |     def __init__(self, userId, username, password, role):
38 |         self.userId = userId
39 |         self.username = username
40 |         self.password = password
41 |         self.role = role
42 |
43 | class UserNotFoundException(Exception):
44 |     pass
45 |
46 | class OrderNotFoundException(Exception):
47 |     pass
```

- Toolbars and Status Bar:** Includes file navigation, search, and system status (CPU, RAM, battery, network, date/time).

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

- Project:** OMS
- File:** db\_util.py
- Code Content:** The same as the first screenshot, but with additional class definitions at the bottom of the file.

```
24 | class Electronics(Product):
25 |     def __init__(self, productId, productName, description, price, quantityInStock, type):
26 |         super().__init__(productId, productName, description, price, quantityInStock, type)
27 |         self.brand = brand
28 |         self.warrantyPeriod = warrantyPeriod
29 |
30 | class Clothing(Product):
31 |     def __init__(self, productId, productName, description, price, quantityInStock, type):
32 |         super().__init__(productId, productName, description, price, quantityInStock, type)
33 |         self.size = size
34 |         self.color = color
35 |
36 | class User:
37 |     def __init__(self, userId, username, password, role):
38 |         self.userId = userId
39 |         self.username = username
40 |         self.password = password
41 |         self.role = role
42 |
43 | class UserNotFoundException(Exception):
44 |     pass
45 |
46 | class OrderNotFoundException(Exception):
47 |     pass
```

- Toolbars and Status Bar:** Includes file navigation, search, and system status (CPU, RAM, battery, network, date/time).

The screenshot shows the PyCharm IDE interface with the project 'OMS' open. The left sidebar displays the project structure with files like db\_util.py, main.py, and order\_management\_repository.py selected. The right pane shows the code for db\_util.py:

```
45     3 usages
46     class OrderNotFound(Exception):
47         pass
48
49     6 usages
50     class IOrderManagementRepository:
51         @staticmethod
52         def createOrder(user, products):
53             try:
54                 # Checking here if the user exists in the database or not
55                 connection = DBUtil.getDBConn()
56                 cursor = connection.cursor()
57                 cursor.execute(operation="SELECT * FROM User WHERE userId = %s", params=(user.userId,))
58                 existing_user = cursor.fetchone()
59
60                 if existing_user is None:
61                     # and if the User is not found, creating the user
62                     cursor.execute(operation="INSERT INTO User (userId, username, password, r", params=(user.userId, user.username, user.password, user.r))
63                     connection.commit()
64
65                 # Creating the order
66                 for product in products:
67                     cursor.execute(operation="INSERT INTO ProductOrder (userId, productId)", params=(user.userId, product.productId))
68                     connection.commit()
69
70             finally:
71                 connection.close()
```

The status bar at the bottom indicates Python 3.12 (OMS) and the date/time 2023-12-22.

The screenshot shows the PyCharm IDE interface with the project 'OMS' open. The left sidebar displays the project structure with files like db\_util.py, main.py, and order\_management\_repository.py selected. The right pane shows the code for db\_util.py:

```
70
71
72     connection.close()
73 except Exception as e:
74     print(f"Error creating order: {e}")
75     raise
76
77     1 usage
78     @staticmethod
79     def cancelOrder(userId, orderId):
80         try:
81             # Checking here if the user exists in the database or not
82             connection = DBUtil.getDBConn()
83             cursor = connection.cursor()
84
85             cursor.execute(operation="SELECT * FROM User WHERE userId = %s", params=(user.userId,))
86             existing_user = cursor.fetchone()
87
88             if existing_user is None:
89                 raise UserNotFound("User not found")
90
91             cursor.execute(operation="SELECT * FROM ProductOrder WHERE userId = %s AND orderId = %s", params=(user.userId, orderId))
92             existing_order = cursor.fetchone()
93
94             if existing_order is None:
95                 raise OrderNotFound("Order not found")
96
97             # Cancelling the order
```

The status bar at the bottom indicates Python 3.12 (OMS) and the date/time 2023-12-22.

The screenshot shows the PyCharm IDE interface with the project 'OMS' open. The left sidebar displays the project structure, including files like main.py, db\_util.py, and order\_management\_repository.py. The right pane shows the code for db\_util.py, specifically the 'cancel\_order' function. The code uses a cursor to execute a DELETE query to remove an order from the database. It handles exceptions for UserNotFound and OrderNotFound, and prints error messages for other exceptions.

```
# Cancelling the order
cursor.execute(operation="DELETE FROM ProductOrder WHERE userId = %s AND order_id = %s", params=(userId, orderId))
connection.commit()

connection.close()
except UserNotFound as e:
    print(f"UserNotFound: {e}")
    raise
except OrderNotFound as e:
    print(f"OrderNotFound: {e}")
    raise
except Exception as e:
    print(f"Error cancelling order: {e}")
    raise

1 usage
@staticmethod
def createProduct(admin_user, product):
    try:
        # Checking here if the user exists in the database or not
        connection = DBUtil.getDBConn()
        cursor = connection.cursor()
        cursor.execute(operation="SELECT * FROM User WHERE userId = %s AND role = 'Admin'")
        existing_admin = cursor.fetchone()

        if existing_admin is None:
            raise UserNotFound("Admin user not found")
```

The screenshot shows the PyCharm IDE interface with the project 'OMS' open. The left sidebar displays the project structure, including files like main.py, db\_util.py, and order\_management\_repository.py. The right pane shows the code for db\_util.py, specifically the 'create\_product' function. The code uses a cursor to execute an INSERT INTO query for a product. It checks if the product type is Electronics or Clothing and performs the appropriate insertion. It handles exceptions for UserNotFound and inserts a record for the product.

```
raise UserNotFound("Admin user not found")  # Creating the product
cursor.execute(operation="INSERT INTO Product (productId, productName, description, price, quantityInStock, type)", "VALUES (%s, %s, %s, %s, %s, %s)", params=(product.productId, product.productName, product.description, product.price, product.quantityInStock, product.type))
connection.commit()

# Checking if it is a electronics product or a clothing product
if product.type == 'Electronics':
    cursor.execute(operation="INSERT INTO Electronics (productId, brand, warranty)", "VALUES (%s, %s, %s)", params=(product.productId, product.brand, product.warranty))
elif product.type == 'Clothing':
    cursor.execute(operation="INSERT INTO Clothing (productId, size, color)", "VALUES (%s, %s, %s)", params=(product.productId, product.size, product.color))

connection.commit()
connection.close()
except UserNotFound as e:
    print(f"UserNotFound: {e}")
    raise
except Exception as e:
    print(f"Error creating product: {e}")
    raise
```

The screenshot shows the PyCharm IDE interface with the project 'OMS' open. The left sidebar displays the project structure, including files like .venv, db\_util.py, main.py, and order\_management\_repository.py. The right pane shows the code editor for db\_util.py. The code defines two static methods: createUser and getAllProducts. The createUser method inserts a user into the User table with parameters (userId, username, password, role). The getAllProducts method retrieves all products from the Product table.

```
148
149     1 usage
150     @staticmethod
151     def createUser(user):
152         try:
153             # Creating the user
154             conn = DBUtil.getDBConn()
155             cursor = conn.cursor()
156             cursor.execute(operation: "INSERT INTO User (userId, username, password, role)", params: (user.userId, user.username, user.password, user.role))
157             conn.commit()
158             conn.close()
159         except Exception as e:
160             print(f"Error creating user: {e}")
161             raise
162
163     1 usage
164     @staticmethod
165     def getAllProducts():
166         try:
167             # getting all products from the database
168             conn = DBUtil.getDBConn()
169             cursor = conn.cursor(dictionary=True)
170             cursor.execute("SELECT * FROM Product")
171             products = cursor.fetchall()
172             conn.close()
173             return products
```

This screenshot shows the same PyCharm session after modifications. The code editor now includes logic for getting ordered products by user. The getOrderByUser method performs a JOIN between the Product and ProductOrder tables to filter results by user ID. The code also includes a class definition for OrderProcessor and a main function.

```
173     return products
174     except Exception as e:
175         print(f"Error getting all products: {e}")
176         raise
177
178     1 usage
179     @staticmethod
180     def getOrderByUser(user):
181         try:
182             # getting all products ordered by a specific user
183             conn = DBUtil.getDBConn()
184             cursor = conn.cursor(dictionary=True)
185             cursor.execute(operation: "SELECT p.* FROM Product p "
186                           "JOIN ProductOrder po ON p.productId = po.productId "
187                           "WHERE po.userId = %s",
188                           params: (user.userId,))
189             ordered_products = cursor.fetchall()
190             conn.close()
191             return ordered_products
192         except Exception as e:
193             print(f"Error getting ordered products: {e}")
194             raise
195
196     class OrderProcessor(IOrderManagementRepository):
197         pass
198
199     1 usage
200     def main():
```

PyCharm Project Structure:

- OMS
- .venv library root
- Lib
- Scripts
- .gitignore
- pyenv.cfg
- db\_util.py
- main.py
- order\_management\_repository.py

Current File: db\_util.py

```
def main():  
    while True:  
        print("Menu:")  
        print("1: createUser")  
        print("2: createProduct")  
        print("3: cancelOrder")  
        print("4: getAllProducts")  
        print("5: getOrderByUser")  
        print("6: exit/end")  
  
        choice = input("Enter your choice: ")  
  
        if choice == "1":  
            userId = int(input("Enter userId: "))  
            username = input("Enter username: ")  
            password = input("Enter password: ")  
            role = input("Enter role: ")  
            user = User(userId, username, password, role)  
            IOrderManagementRepository.createUser(user)  
  
        elif choice == "2":  
            # Assuming admin_user is already created and available  
            admin_user = User(userid=1, username="rohan", password="123", role="Admin")  
  
            productId = int(input("Enter productId: "))  
            productName = input("Enter productName: ")  
            description = input("Enter description: ")  
            price = float(input("Enter price: "))
```

File Status: 1:1 CRLF UTF-8 4 spaces Python 3.12 (OMS) 6:11 PM 2023-12-22

PyCharm Project Structure:

- OMS
- .venv library root
- Lib
- Scripts
- .gitignore
- pyenv.cfg
- db\_util.py
- main.py
- order\_management\_repository.py

Current File: db\_util.py

```
price = float(input("Enter price: "))  
quantityInStock = int(input("Enter quantityInStock: "))  
productType = input("Enter product type (Electronics/Clothing): ")  
  
if productType.lower() == "electronics":  
    brand = input("Enter brand: ")  
    warrantyPeriod = int(input("Enter warrantyPeriod: "))  
    product = Electronics(productId, productName, description, price, quantityInStock, brand, warrantyPeriod)  
elif productType.lower() == "clothing":  
    size = input("Enter size: ")  
    color = input("Enter color: ")  
    product = Clothing(productId, productName, description, price, quantityInStock, size, color)  
else:  
    print("Invalid product type.")  
    continue  
  
IOrderManagementRepository.createProduct(admin_user, product)  
  
elif choice == "3":  
    userId = int(input("Enter userId: "))  
    orderId = int(input("Enter orderId: "))  
    try:  
        IOrderManagementRepository.cancelOrder(userId, orderId)  
        print("Order cancelled successfully.")  
    except UserNotFound as e:  
        print(f"UserNotFound: {e}")  
    except OrderNotFound as e:  
        print(f"OrderNotFound: {e}")
```

File Status: 1:1 CRLF UTF-8 4 spaces Python 3.12 (OMS) 6:11 PM 2023-12-22

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

- Project View:** Shows the project structure under "OMS". The file "db\_util.py" is currently selected.
- Code Editor:** Displays the content of "db\_util.py". The code handles user input for choices 1 through 6, interacting with an Order Management Repository. It includes exception handling for UserNotFound and OrderNotFound.
- Status Bar:** Shows the file path "OMS > db\_util.py", the current file name "db\_util.py", and other settings like encoding (CRLF), line width (1:1), and Python version (Python 3.12 (OMS)).
- System Tray:** Shows the date and time (2023-12-22, 6:11 PM), battery level (Haze), and system icons.

```
249     except UserNotFound as e:
250         print(f"UserNotFound: {e}")
251     except OrderNotFound as e:
252         print(f"OrderNotFound: {e}")
253
254     elif choice == "4":
255         products = IOrderManagementRepository.getAllProducts()
256         for product in products:
257             print(product)
258
259     elif choice == "5":
260         userId = int(input("Enter userId: "))
261         user = User(userId, username="", password="", role="") #given empty because
262         ordered_products = IOrderManagementRepository.getOrderByUser(user)
263         for product in ordered_products:
264             print(product)
265
266     elif choice == "6":
267         break
268
269     else:
270         print("Invalid choice.Enter correct number")
271
272 if __name__ == "__main__":
273     main()
```

The screenshot shows the PyCharm IDE interface with the 'OMS' project open. The 'creatingtables.py' file is the active editor, displaying SQL code for creating three tables: Product, Electronics, and Clothing. The code uses Python's DB API to execute the SQL statements.

```
1 create_database oms;
2 use oms;
3
4
5 CREATE TABLE Product (
6     productId INT PRIMARY KEY,
7     productName VARCHAR(255),
8     description VARCHAR(255),
9     price DOUBLE,
10    quantityInStock INT,
11    type VARCHAR(255)
12 );
13
14 CREATE TABLE Electronics (
15     productId INT PRIMARY KEY,
16     brand VARCHAR(255),
17     warrantyPeriod INT,
18     FOREIGN KEY (productId) REFERENCES Product(productId)
19 );
20
21
22 CREATE TABLE Clothing (
23     productId INT PRIMARY KEY,
24     size VARCHAR(255),
25     color VARCHAR(255),
26     FOREIGN KEY (productId) REFERENCES Product(productId)
27 );
28
```

The screenshot shows the PyCharm IDE interface with the 'OMS' project open. The 'creatingtables.py' file is the active editor, displaying SQL code for creating four tables: Clothing, User, ProductOrder, and a modified version of the Product table. The code uses Python's DB API to execute the SQL statements.

```
21
22 CREATE TABLE Clothing (
23     productId INT PRIMARY KEY,
24     size VARCHAR(255),
25     color VARCHAR(255),
26     FOREIGN KEY (productId) REFERENCES Product(productId)
27 );
28
29
30 CREATE TABLE User (
31     userId INT PRIMARY KEY,
32     username VARCHAR(255),
33     password VARCHAR(255),
34     role VARCHAR(255)
35 );
36
37 CREATE TABLE ProductOrder(
38     orderId int primary key,
39     userId INT,
40     productId INT);
41 /*created this to add order details */
42 ALTER TABLE productorder
43 MODIFY COLUMN orderId INT AUTO_INCREMENT;
```