

ORDER MANAGEMENT SYSTEM

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

Order_Management...2VF\SAMEER (66)

```
select * from Products
select * from Users;
select * from Orders
```

100 %

Results Messages

	product_id	product_name	description	price	quantity_in_stock	type	brand	warranty_period	size	color
1	1	Smartphone	Latest model smartphone	699.99	100	Electronics	BrandX	24	NULL	NULL
2	2	Laptop	High performance laptop	1299.99	50	Electronics	BrandY	36	NULL	NULL
3	3	Television	4K Ultra HD TV	899.99	20	Electronics	BrandZ	12	NULL	NULL
4	4	T-Shirt	Cotton T-shirt	19.99	200	Clothing	NULL	NULL	L	Red
5	5	Jeans	Denim jeans	49.99	150	Clothing	NULL	NULL	32	Blue
6	6	Sweater	Woolen sweater	39.99	100	Clothing	NULL	NULL	M	Green
7	7	Headphones	Noise-canceling headphones	199.99	80	Electronics	BrandA	12	NULL	NULL
8	8	Watch	Smartwatch with fitness tracking	299.99	60	Electronics	BrandB	18	NULL	NULL

	user_id	username	password	role
1	1	admin1	password1	Admin
2	2	admin2	password2	Admin
3	3	user1	password3	User
4	4	user2	password4	User
5	5	user3	password5	User
6	6	user4	password6	User
7	7	user5	password7	User
8	8	user6	password8	User

	order_id	user_id	product_id
1	1	1	1
2	2	2	2
3	4	4	4
4	5	5	5
5	6	6	6
6	9	9	9
7	13	13	13
8	14	14	14
9	15	15	15

Query executed successfully.

DESKTOP-804A2VF\SQLEXPRESS0... | DESKTOP-804A2VF\SAMEER... | OrderManagement | 00:00:00 | 65 rows

Tasks

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].

```
# 1. Create a base class called Product with the following attributes:
class Product:
    def __init__(self, productName, description, price, quantityInStock, type, productId=None):
        self.productId = productId
        self.productName = productName
        self.description = description
        self.price = price
        self.quantityInStock = quantityInStock
        self.type = type
```

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

```
11 # 2. Implement constructors, getters, and setters for the Product class.
12
13 # Getters
14
15 def get_product_id(self) -> int:
16     return self.productId
17
18 def get_product_name(self) -> str:
19     return self.productName
20
21 def get_description(self) -> str:
22     return self.description
23
24 def get_price(self) -> float:
25     return self.price
26
27 def get_quantity_in_stock(self) -> int:
28     return self.quantityInStock
29
30 def get_type(self) -> str:
31     return self.type
32
33 # Setters
34 def set_product_name(self, productName: str):
35     self.productName = productName
36
37 def set_description(self, description: str):
38     self.description = description
39
40 def set_price(self, price: float):
41     self.price = price
42
43 def set_quantity_in_stock(self, quantityInStock: int):
44     self.quantityInStock = quantityInStock
45
```

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

```
entity > electronics.py > Electronics
1  # 3. Create a subclass Electronics that inherits from Product
2  from entity.product import Product
3
4
5  class Electronics(Product):
6      def __init__(self, productName, description, price, quantityInStock, brand, model, warranty, power_usage):
7          super().__init__(self, productName, description, price,
8                          quantityInStock, "Electronics") # Call the base class constructor
9          self.brand = brand # Brand of the electronic product
10         self.warranty = warranty # Warranty period (in months or years)
11
12     # Getters
13     def getBrand(self):
14         return self.brand
15
16     def getWarrantyPeriod(self):
17         return self.warrantyPeriod
18
19     # Setters
20     def setBrand(self, brand):
21         self.brand = brand
22
23     def setWarrantyPeriod(self, warrantyPeriod):
24         self.warrantyPeriod = warrantyPeriod
25
```

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

```
entity > clothing.py > Clothing > getSize
1  # 4. Create a subclass Clothing that also inherits from Product.
2
3  from entity.product import Product
4
5
6  class Clothing(Product):
7
8      def __init__(self, productName, description, price, quantityInStock, size, color):
9          super().__init__(self, productName, description, price, quantityInStock, "Clothing")
10         self.size = size # Size of the clothing item (e.g., S, M, L, XL)
11         # Color of the clothing item (e.g., Red, Blue, Black)
12         self.color = color
13
14     # Getters
15     def getSize(self):
16         return self.size
17
18     def getColor(self):
19         return self.color
20
21     # Setters
22     def setSize(self, size):
23         self.size = size
24
25     def setColor(self, color):
26         self.color = color
27
```

5. Create a User class with attributes - `userId (int)`, `username (String)`, `password (String)`, `role (String)` - "Admin" or "User"

```
entity > user.py > User
1 # 5. Create a User class with attributes:
2
3 class User:
4     def __init__(self, username, password, role, userId=None):
5         self.userId = userId
6         self.username = username
7         self.password = password
8         self.role = role
9
10    # Getter for userId
11    @property
12    def userId(self):
13        return self._userId
14
15    # Setter for userId
16    @userId.setter
17    def userId(self, value):
18        self._userId = value
19
20    # Getter for username
21    @property
22    def username(self):
23        return self._username
24
25    # Setter for username
26    @username.setter
27    def username(self, value):
28        self._username = value
29
30    # Getter for password
31    @property
32    def password(self):
33        return self._password
34
35    # Setter for password
36    @password.setter
37    def password(self, value):
38        self._password = value
39
40    # Getter for role
41    @property
42    def role(self):
43        return self._role
44
```

6. Define an interface/abstract class named `IOrderManagementRepository`

```
1 # ABSTRACT CLASS
2 # 6. Define an interface/abstract class named IOrderManagementRepository
3
4 from abc import ABC, abstractmethod
5
6
7 class IOrderManagementRepository(ABC):
8     @abstractmethod
9     def create_order(self, user, products):
10        pass
11
12    @abstractmethod
13    def cancel_order(self, userId, orderId):
14        pass
15
16    @abstractmethod
17    def create_product(self, user, product):
18        pass
19
20    @abstractmethod
21    def create_user(self, user):
22        pass
23
24    @abstractmethod
25    def get_all_products(self):
26        pass
27
28    @abstractmethod
29    def get_order_by_user(self, user):
30        pass
31
```

7. Implement the IOrderManagementRepository interface/abstractclass

Code in : [Order_Process_class \(Implementation\)](#)

8. Create DBUtil class and add the following method (DATABASE CONNECTION)

```
1  import pyodbc
2  # 8. Create DBUtil class
3
4
5  class DBUtil:
6      @staticmethod
7      def getDBConn() -> pyodbc.Connection:
8          """Establish a connection to the database and return the connection object."""
9          connection_string = (
10              r'Driver={SQL Server};'
11              r'Server=DESKTOP-804A2VF\SQLEXPRESS09;'
12              'Database=OrderManagement;'
13              'Trusted_Connection=yes;'
14          )
15          try:
16              connection = pyodbc.connect(connection_string)
17              print("Connected Successfully")
18              return connection
19          except Exception as e:
20              print("Connection failed: {}".format(e))
21              return None
22
23
24  db_connection = DBUtil.getDBConn()
25
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Connected Successfully
Connected Successfully

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:
* History restored

PS C:\Users\SAMEER\OneDrive\Desktop\HEXa\CodingChallenges\OrderManagement> python util/db_connection_util.py
Connected Successfully
PS C:\Users\SAMEER\OneDrive\Desktop\HEXa\CodingChallenges\OrderManagement>

9. Create OrderManagement main class and perform following operation:

CreateUser

Before:

	user_id	username	password	role
29	33	sameer	pal@123	Admin
30	34	sam	eer	Admin
31	35	sartha	sas#234	Admin
32	37	sdf	wer	Admin
33	38	s	f	Admin

User Input:

```
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: 1
Enter username: sarthak
Enter password: sharma
Enter role (Admin/User): Admin
User sarthak created successfully.
User sarthak created with role Admin
```

After:

	user_id	username	password	role
30	34	sam	eer	Admin
31	35	sartha	sas#234	Admin
32	37	sdf	wer	Admin
33	38	s	f	Admin
34	39	sarthak	sharma	Admin



CreateProduct

Before:

	product_id	product_name	description	price	quantity_in_stock	type	brand	warranty_period	size	color
19	19	phone	samsusg A23	34000...	45	Electronics	NULL	NULL	NULL	NULL
20	20	laptop	gaming	12000...	2	Electronics	NULL	NULL	NULL	NULL
21	21	Tv	oled	2300.00	23	Electronics	NULL	NULL	NULL	NULL
22	22	smartphone	vivo curved screen	2300.00	2333	Electronics	NULL	NULL	NULL	NULL
23	23	samsung	tv	33232...	2	Electronics	NULL	NULL	NULL	NULL

Query executed successfully | DESKTOP-804A2VF\SQLEXPRESS0... | DESKTOP-804A2VF\SAMEER... | OrderManagement | 00:00:00 | 23 rows

User Input:

```
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: 2
Enter product name: Sony Xperia
Enter description: SmartPhone
Enter price: 230000
Enter quantity in stock: 23
Enter type (Electronics/Clothing): Electronics
Enter your user ID: 33
Product Sony Xperia created successfully.
```

After:


	product_id	product_name	description	price	quantity_in_stock	type	brand	warranty_period	size	color
20	20	laptop	gaming	12000...	2	Electronics	NULL	NULL	NULL	NULL
21	21	Tv	oled	2300.00	23	Electronics	NULL	NULL	NULL	NULL
22	22	smartphone	vivo curved screen	2300.00	2333	Electronics	NULL	NULL	NULL	NULL
23	23	samsung	tv	33232...	2	Electronics	NULL	NULL	NULL	NULL
24	24	Sony Xperia	SmartPhone	23000...	23	Electronics	NULL	NULL	NULL	NULL

Query executed successfully | DESKTOP-804A2VF\SQLEXPRESS0... | DESKTOP-804A2VF\SAMEER... | OrderManagement | 00:00:00 | 24 rows

CancelOrder:


When Order ID doesn't Exist:

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: 3
Enter user ID: 1
Enter order ID: 9
Order ID 9 not found for User ID 1.
```

Before:



	order_id	user_id	product_id
5	6	6	6
6	9	9	9
7	13	13	13
8	14	14	14
9	15	15	15

User Input:


```
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: 3
Enter user ID: 15
Enter order ID: 15
Order 15 canceled for User ID 15.
```

After:

	order_id	user_id	product_id
4	5	5	5
5	6	6	6
6	9	9	9
7	13	13	13
8	14	14	14

GetAllProducts



```
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: 3
Enter user ID: 15
(6, 'Sweater', 'Woolen sweater', Decimal('39.99'), 100, 'Clothing', None, None, 'M', 'Green')
(7, 'Headphones', 'Noise-canceling headphones', Decimal('199.99'), 80, 'Electronics', 'BrandA', 12, None, None)
(8, 'Watch', 'Smartwatch with fitness tracking', Decimal('299.99'), 60, 'Electronics', 'BrandB', 18, None, None)
(9, 'Shirt', 'Formal shirt', Decimal('29.99'), 100, 'Clothing', None, None, 'M', 'White')
(10, 'Jacket', 'Leather jacket', Decimal('129.99'), 50, 'Clothing', None, None, 'L', 'Black')
(11, 'Microwave', 'Convection microwave', Decimal('99.99'), 40, 'Electronics', 'BrandC', 24, None, None)
(12, 'Blender', 'High-speed blender', Decimal('59.99'), 30, 'Electronics', 'BrandD', 18, None, None)
(13, 'Shoes', 'Running shoes', Decimal('79.99'), 120, 'Clothing', None, None, '9', 'Black')
(14, 'Camera', 'DSLR camera', Decimal('499.99'), 25, 'Electronics', 'BrandE', 24, None, None)
(15, 'Socks', 'Cotton socks', Decimal('9.99'), 300, 'Clothing', None, None, 'One size', 'Gray')
(16, 'asd', 'werwer', Decimal('34235.00'), 34, 'Clothing', None, None, None, None)
(17, 'qwe', 'dfsd', Decimal('234.00'), 234, 'Clothing', None, None, None, None)
(18, 'tV', 'TV OLED', Decimal('230000.00'), 23, 'Electronics', None, None, None, None)
(19, 'phone', 'samsusg A23', Decimal('340000.00'), 45, 'Electronics', None, None, None, None)
(20, 'laptop', 'gaming', Decimal('120000.00'), 2, 'Electronics', None, None, None, None)
(21, 'Tv', 'oled', Decimal('2300.00'), 23, 'Electronics', None, None, None, None)(22, 'smartphone', 'vivo curved screen', Decimal('2300.00'), 2333, 'Electronics', None, None, None, None)
(23, 'samsung', 'tv', Decimal('332323.00'), 2, 'Electronics', None, None, None, None)
(24, 'Sony Xperia', 'SmartPhone', Decimal('230000.00'), 23, 'Electronics', None, None, None, None)
```

GetOrderbyUser:

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: 5

Enter user ID: 4

(4, 'T-Shirt', 'Cotton T-shirt', Decimal('19.99'), 4, 'user2', 'User')

Join - Product, Users, Orders

Results										
	product_id	product_name	description	price	quantity_in_stock	type	brand	warranty_period	size	color
1	1	Smartphone	Latest model smartphone	699.99	100	Electronics	BrandX	24	NULL	NULL
2	2	Laptop	High performance laptop	1299.99	50	Electronics	BrandY	36	NULL	NULL
3	3	Television	4K Ultra HD TV	899.99	20	Electronics	BrandZ	12	NULL	NULL
4	4	T-Shirt	Cotton T-shirt	19.99	200	Clothing	NULL	NULL	L	Red
5	5	Jeans	Denim jeans	49.99	150	Clothing	NULL	NULL	32	Blue
6	6	Sweater	Woolen sweater	39.99	100	Clothing	NULL	NULL	M	Gre...
7	7	Headphones	Noise-canceling headp...	199.99	80	Electronics	BrandA	12	NULL	NULL
8	8	Watch	Smartwatch with fitness ...	299.99	60	Electronics	BrandB	18	NULL	NULL
9	9	Shirt	Formal shirt	29.99	100	Clothing	NULL	NULL	M	White
10	10	Jacket	Leather jacket	129.99	50	Clothing	NULL	NULL	L	Black
11	11	Microwave	Convection microwave	99.99	40	Electronics	BrandC	24	NULL	NULL

	user_id	username	password	role
1	1	admin1	password1	Admin
2	2	admin2	password2	Admin
3	3	user1	password3	User
4	4	user2	password4	User
5	5	user3	password5	User
6	6	user4	password6	User
7	7	user5	password7	User
8	8	user6	password8	User
9	9	user7	password9	User
10	10	user8	passwor...	User
11	11	user9	password...	User

	order_id	user_id	product_id
1	1	1	1
2	2	2	2
3	4	4	4
4	5	5	5
5	6	6	6
6	9	9	9
7	13	13	13
8	14	14	14

Query executed successfully.

