

## Coding Challenge :-6

### Order Mangement System

SQL tables:

```
Database_Query_OMS
Limit to 1000 rows

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
```

```
Database_Query_OMS
Limit to 1000 rows

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;
44
45
```

## Code:

```
1 import mysql.connector
2
3 class DBUtil:
4     @staticmethod
5     def getDBConn():
6         connection = mysql.connector.connect(
7             host="localhost",
8             user="root",
9             password="root",
10            database="lms"
11        )
12        return connection
13
14 class Product:
15     def __init__(self, productId, productName, description, price, quantityInStock, type):
16         self.productId = productId
17         self.productName = productName
18         self.description = description
19         self.price = price
20         self.quantityInStock = quantityInStock
21         self.type = type
22
23 class Electronics(Product):
24     def __init__(self, productId, productName, description, price, quantityInStock, type, brand, warrantyPeriod):
25         super().__init__(productId, productName, description, price, quantityInStock, type)
26         self.brand = brand
27         self.warrantyPeriod = warrantyPeriod
28
29 class Clothing(Product):
30     def __init__(self, productId, productName, description, price, quantityInStock, type, size, color):
31         super().__init__(productId, productName, description, price, quantityInStock, type)
32         self.size = size
33         self.color = color
34
```

```
35 class User:
36     def __init__(self, userId, username, password, role):
37         self.userId = userId
38         self.username = username
39         self.password = password
40         self.role = role
41
42 class UserNotFound(Exception):
43     pass
44
45 class OrderNotFound(Exception):
46     pass
47
```

```

48 class JOrderManagementRepository:
49     @staticmethod
50     def createOrder(user, products):
51         try:
52             # Checking here if the user exists in the database or not
53             connection = DBUtil.getDBConn()
54             cursor = connection.cursor()
55             cursor.execute("SELECT * FROM User WHERE userid = %s", (user.userId,))
56             existing_user = cursor.fetchone()
57
58             if existing_user is None:
59                 # and if the user is not found, creating the user
60                 cursor.execute("INSERT INTO User (userid, username, password, role) VALUES (%s, %s, %s, %s)",
61                               (user.userId, user.username, user.password, user.role))
62                 connection.commit()
63
64             # Creating the order
65             for product in products:
66                 cursor.execute("INSERT INTO ProductOrder (userid, productId) VALUES (%s, %s)",
67                               (user.userId, product.productId))
68                 connection.commit()
69
70             connection.close()
71         except Exception as e:
72             print(f"Error creating order: {e}")
73             raise
74

```

```

76 @staticmethod
77 def cancelOrder(userId, orderId):
78     try:
79         # Checking here if the user exists in the database or not
80         connection = DBUtil.getDBConn()
81         cursor = connection.cursor()
82
83         cursor.execute("SELECT * FROM User WHERE userid = %s", (userId,))
84         existing_user = cursor.fetchone()
85
86         if existing_user is None:
87             raise UserNotFound("User not found")
88
89         cursor.execute("SELECT * FROM ProductOrder WHERE userid = %s AND orderId = %s", (userId, orderId))
90         existing_order = cursor.fetchone()
91
92         if existing_order is None:
93             raise OrderNotFound("Order not found")
94
95         # Cancelling the order
96         cursor.execute("DELETE FROM ProductOrder WHERE userid = %s AND orderId = %s", (userId, orderId))
97         connection.commit()
98
99         connection.close()
100     except UserNotFound as e:
101         print(f"UserNotFound: {e}")
102         raise
103     except OrderNotFound as e:
104         print(f"OrderNotFound: {e}")
105         raise
106     except Exception as e:
107         print(f"Error cancelling order: {e}")
108         raise

```

```

110 @staticmethod
111 def createProduct(admin_user, product):
112     try:
113         # Checking here if the user exists in the database or not
114         connection = DBUtil.getDBConn()
115         cursor = connection.cursor()
116         cursor.execute("SELECT * FROM User WHERE userId = %s AND role = 'Admin'", (admin_user.userId,))
117         existing_admin = cursor.fetchone()
118
119         if existing_admin is None:
120             raise UserNotFound("Admin user not found")
121
122         # Creating the product
123         cursor.execute("INSERT INTO Product (productId, productName, description, price, quantityInStock, type) "
124             "VALUES (%s, %s, %s, %s, %s, %s)",
125             (product.productId, product.productName, product.description, product.price,
126             product.quantityInStock, product.type))
127         connection.commit()
128
129         # Checking if it is a electronics product or a clothing product
130         if product.type == 'Electronics':
131             cursor.execute("INSERT INTO Electronics (productId, brand, warrantyPeriod) "
132                 "VALUES (%s, %s, %s)",
133                 (product.productId, product.brand, product.warrantyPeriod))
134         elif product.type == 'Clothing':
135             cursor.execute("INSERT INTO Clothing (productId, size, color) "
136                 "VALUES (%s, %s, %s)",
137                 (product.productId, product.size, product.color))
138
139         connection.commit()
140         connection.close()
141     except UserNotFound as e:
142         print(f"UserNotFound: {e}")
143         raise
144     except Exception as e:
145         print(f"Error creating product: {e}")
146         raise

```

```

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

```

```

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

```

## Inputs to codes:

```

197 def main():
198     while True:
199         print("Menu:")
200         print("1: createUser")
201         print("2: createProduct")
202         print("3: cancelOrder")
203         print("4: getAllProducts")
204         print("5: getOrderByUser")
205         print("6: exit/end")
206
207         choice = input("Enter your choice: ")
208
209         if choice == "1":
210             userId = int(input("Enter userId: "))
211             username = input("Enter username: ")
212             password = input("Enter password: ")
213             role = input("Enter role: ")
214             user = User(userId, username, password, role)
215             OrderManagementRepository.createUser(user)
216
217         elif choice == "2":
218             # Assuming admin_user is already created and available
219             admin_user = User(1, "rohan", "123", "Admin")
220
221             productId = int(input("Enter productId: "))
222             productName = input("Enter productName: ")
223             description = input("Enter description: ")
224             price = float(input("Enter price: "))
225             quantityInStock = int(input("Enter quantityInStock: "))
226             productType = input("Enter product type (Electronics/Clothing): ")
227

```

```

228     if productType.lower() == "electronics":
229         brand = input("Enter brand: ")
230         warrantyPeriod = int(input("Enter warrantyPeriod: "))
231         product = Electronics(productId, productName, description, price, quantityInStock, productType, brand, warrantyPeriod)
232     elif productType.lower() == "clothing":
233         size = input("Enter size: ")
234         color = input("Enter color: ")
235         product = Clothing(productId, productName, description, price, quantityInStock, productType, size, color)
236     else:
237         print("Invalid product type.")
238         continue
239
240     OrderManagementRepository.createProduct(admin_user, product)
241
242 elif choice == "3":
243     userId = int(input("Enter userId: "))
244     orderId = int(input("Enter orderId: "))
245     try:
246         OrderManagementRepository.cancelOrder(userId, orderId)
247         print("Order cancelled successfully.")
248     except UserNotFound as e:
249         print(f"UserNotFound: {e}")
250     except OrderNotFound as e:
251         print(f"OrderNotFound: {e}")
252
253 elif choice == "4":
254     products = OrderManagementRepository.getAllProducts()
255     for product in products:
256         print(product)
257
258 elif choice == "5":
259     userId = int(input("Enter userId: "))
260     user = User(userId, "", "", "") #given empty because we don't require other attributes in this step
261     ordered_products = OrderManagementRepository.getOrderByUser(user)
262     for product in ordered_products:
263         print(product)

```

```

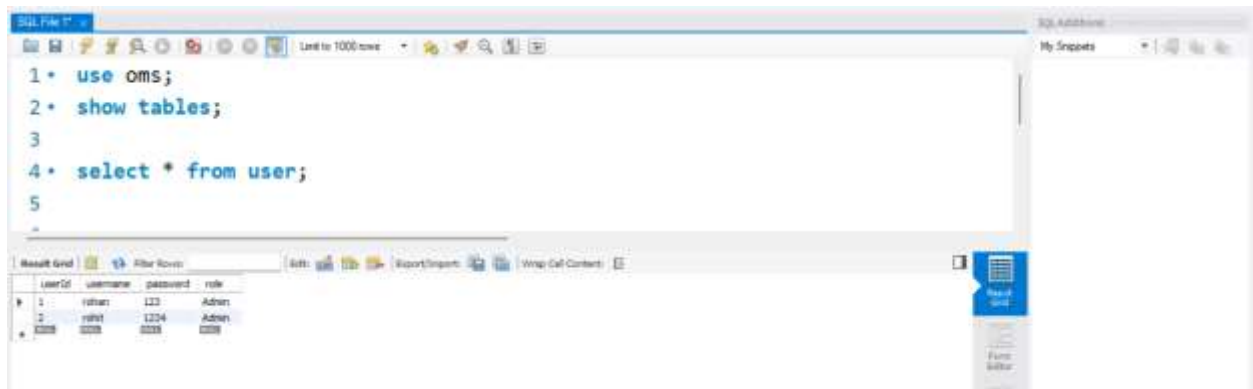
258     elif choice == "5":
259         userId = int(input("Enter userId: "))
260         user = User(userId, "", "", "") #given empty because we don't require other attributes in this step
261         ordered_products = OrderManagementRepository.getOrderByUser(user)
262         for product in ordered_products:
263             print(product)
264
265     elif choice == "6":
266         break
267
268     else:
269         print("Invalid choice.Enter correct number")
270
271 if __name__ == "__main__":
272     main()
273

```

## Operations and outputs:

```
PS D:\Hexaware\Coding_Challenge\Python> & "C:\Program Files\Python312\python.exe" d:\Hexaware\Coding_Challenge\Python\oms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderById
6: exit/end
Enter your choice: 1
Enter userid: 2
Enter username: rohit
Enter password: 1234
Enter role: Admin
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderById
6: exit/end
Enter your choice: 6
PS D:\Hexaware\Coding_Challenge\Python>
```

## Changes in Db:



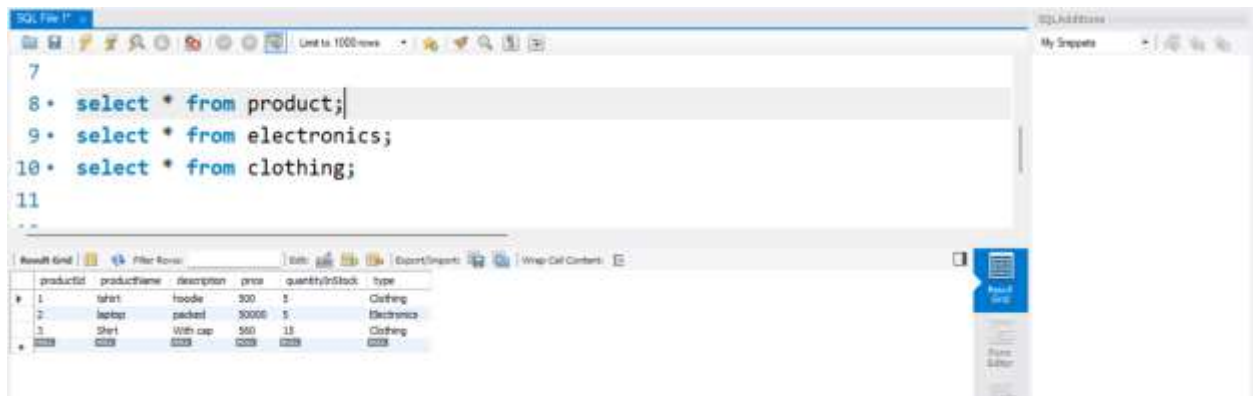


2.

```
PS 0:\Hexaware\Coding_Challenge\Python> & "C:\Program Files\Python312\python.exe" d:\Hexaware\Coding_Challenge\Python\oms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderlyUser
6: exit/end
Enter your choice: 2
Enter productId: 3
Enter productName: shirt
Enter description: with cap
Enter price: 500
Enter quantityInStock: 15
Enter product type (Electronics/Clothing): Clothing
Enter size: L
Enter color: Blue
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderlyUser
6: exit/end
Enter your choice: 6
PS 0:\Hexaware\Coding_Challenge\Python>
```

Changes in DB:

Product table:



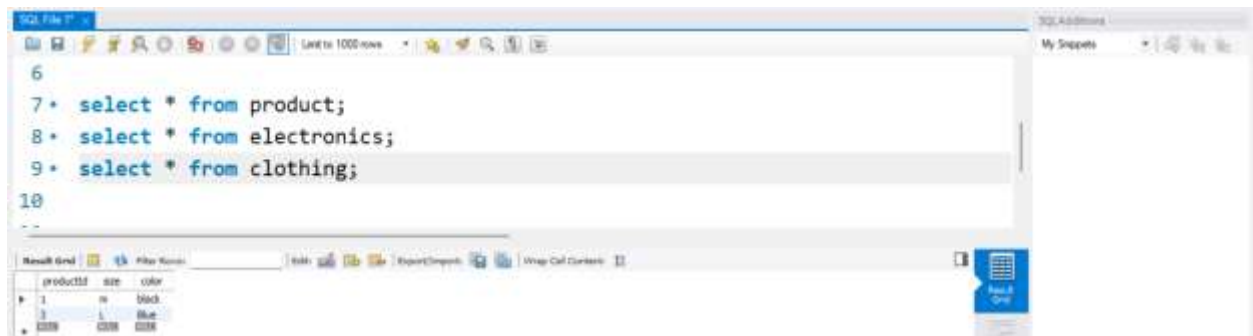
The screenshot shows the SQL Server Enterprise Manager interface. The 'SQL: File 1' window displays the following SQL queries:

```
7
8 * select * from product;
9 * select * from electronics;
10 * select * from clothing;
11
```

The 'Results' window shows the data for the 'product' table:

productId	productName	description	price	quantityInStock	type
1	shirt	hoodie	300	5	Clothing
2	laptop	packed	30000	5	Electronics
3	shirt	With cap	500	15	Clothing

Clothing table:



The screenshot shows the SQL Server Enterprise Manager interface. The 'SQL: File 1' window displays the following SQL queries:

```
6
7 * select * from product;
8 * select * from electronics;
9 * select * from clothing;
10
```

The 'Results' window shows the data for the 'clothing' table:

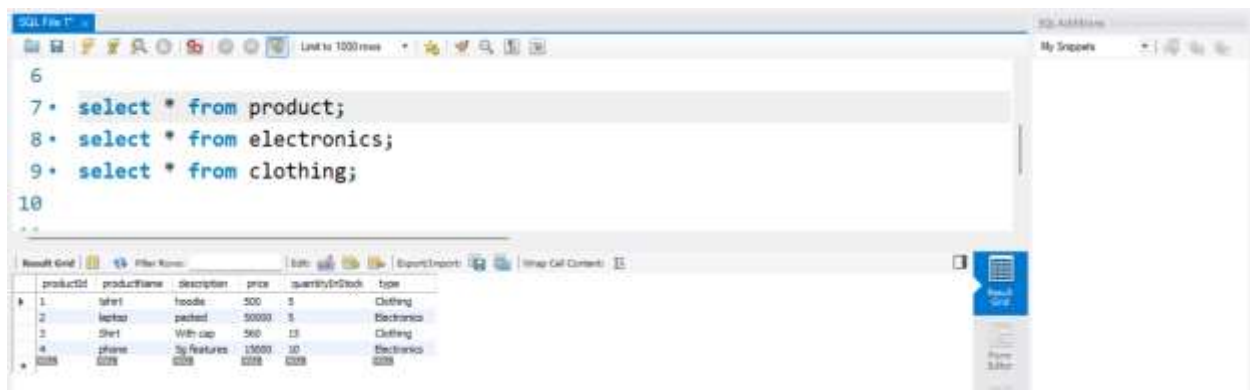
productId	size	color
1	M	black
3	L	Blue



## Inserting Electronic Item:

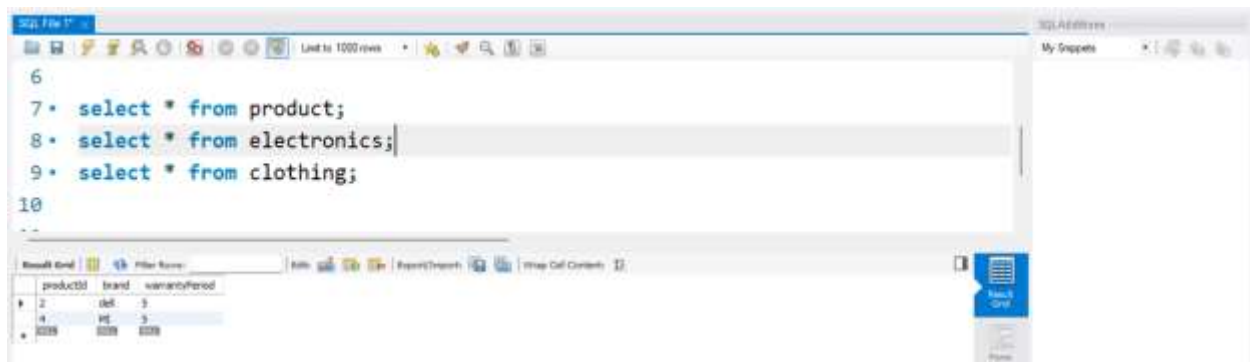
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Vexware\Coding_Challenge\Python> & "C:\Program Files\Python312\python.exe" d:\Vexware\Coding_Challenge\Python\cms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderById
6: exit/end
Enter your choice: 2
Enter productId: 4
Enter productName: phone
Enter description: 5g features
Enter price: 15000
Enter quantityInStock: 10
Enter product type (Electronics/Clothing): Electronics
Enter brand: M
Enter warrantyPeriod: 5
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderById
6: exit/end
Enter your choice: 6
PS D:\Vexware\Coding_Challenge\Python>
```

## Product table:



productId	productName	description	price	quantityInStock	type
1	shirt	hoodie	500	5	Clothing
2	leptop	packed	50000	5	Electronics
3	shirt	With cap	560	15	Clothing
4	phone	5g features	15000	10	Electronics

## Electronic table:



productId	brand	warrantyPeriod
2	del	5
4	MS	5
5		

3.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Hexaware\Coding_Challenge\Python> & "C:/Program Files/Python312/python.exe" d:\Hexaware\Coding_Challenge\Python\oms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByIdUser
6: exit/end
Enter your choice: 3
Enter userId: 1
Enter orderId: 2
Order cancelled successfully.
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByIdUser
6: exit/end
Enter your choice: 6
PS D:\Hexaware\Coding_Challenge\Python>
```

Changes in DB:

```
12
13 • select * from productorder;
```

orderId	userId	productId
1	1	1

As there was only one order it got deleted.

4.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Hexaware\Coding_Challenge\Python> & "C:/Program Files/Python312/python.exe" d:\Hexaware\Coding_Challenge\Python\oms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByIdUser
6: exit/end
Enter your choice: 4
[{'productId': 1, 'productName': 'tshirt', 'description': 'hoodie', 'price': 500.0, 'quantityInStock': 5, 'type': 'Clothing'},
{'productId': 2, 'productName': 'laptop', 'description': 'packed', 'price': 50000.0, 'quantityInStock': 5, 'type': 'Electronics'},
{'productId': 3, 'productName': 'shirt', 'description': 'with cap', 'price': 500.0, 'quantityInStock': 15, 'type': 'Clothing'},
{'productId': 4, 'productName': 'phone', 'description': '5g features', 'price': 15000.0, 'quantityInStock': 10, 'type': 'Electronics'}]
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByIdUser
6: exit/end
Enter your choice: 6
PS D:\Hexaware\Coding_Challenge\Python>
```

**5.**

```

PS D:\Hexaware\Coding_Challenge\Python & "C:\Program Files\Python112\python.exe" d:\Hexaware\Coding_Challenge\Python\cms.py
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByUser
6: exit/end
Enter your choice: 5
Enter userID: 1
[{"productId": 1, "productName": "tshirt", "description": "hoodie", "price": 500.0, "quantityInStock": 5, "type": "Clothing"},
{"productId": 2, "productName": "laptop", "description": "packed", "price": 50000.0, "quantityInStock": 5, "type": "Electronics"},
{"productId": 3, "productName": "Shirt", "description": "with cap", "price": 500.0, "quantityInStock": 15, "type": "Clothing"}]
Menu:
1: createUser
2: createProduct
3: cancelOrder
4: getAllProducts
5: getOrderByUser
6: exit/end
Enter your choice: 6
PS D:\Hexaware\Coding_Challenge\Python
```

**Table where this data got retrieved:**

The screenshot shows the SAP Fiori 'Manage Stock' app interface. The top bar contains the following elements: 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The main area displays a table with the following data:

order	user	productId
1	1	1
2	1	2
3	1	3
4	2	3

The sidebar on the right contains a 'Result Grid' icon and a 'Filter Rows' icon.