

# Kafka Assignment Report

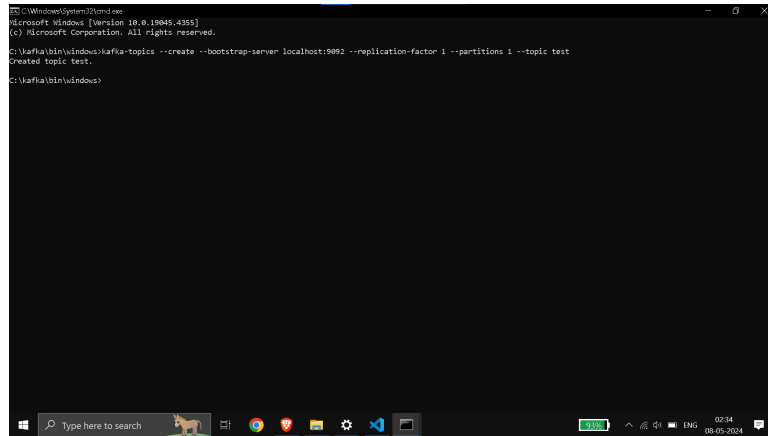
by Sobhan Behuria  
Student ID: 202318040

# Contents

0.1	Creating Topic . . . . .	3
0.2	Producer 1: Inventory Orders Producer . . . . .	3
0.3	Producer 2: Delivery Orders Producer . . . . .	3
0.4	Consumer 1: Inventory Data Consumer . . . . .	4
0.5	Consumer 2: Delivery Data Consumer . . . . .	4
0.6	Filter Logic . . . . .	5

## 0.1 Creating Topic

While establishing a Kafka environment to manage e-commerce order data, a pivotal stage was the initiation of a Kafka topic titled "test". This topic functions as a conduit for structuring and analyzing incoming order-related messages, thereby optimizing data management and analysis within the e-commerce ecosystem.

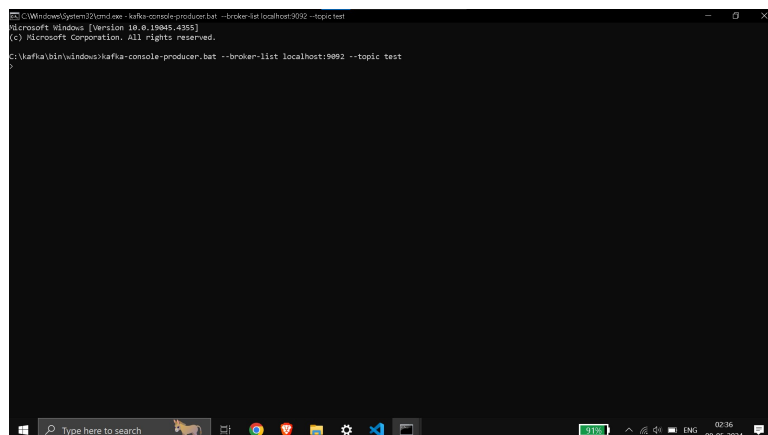


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.

C:\kafka\bin\windows>kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
Created topic test.

C:\kafka\bin\windows>
```

**Figure 1:** Initialization of the "test" topic



```
C:\Windows\system32\cmd.exe - kafka-console-producer.bat --broker-list localhost:9092 --topic test
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.

C:\kafka\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic test
```

**Figure 2:** Consumer Console

## 0.2 Producer 1: Inventory Orders Producer

To handle inventory orders, a Python script titled producer1.py was developed. It loaded JSON data from data.json and applied a custom filtering function. Subsequently, a Kafka producer was instantiated to dispatch the filtered inventory messages to the specified Kafka topic, streamlining inventory management processes within the system.

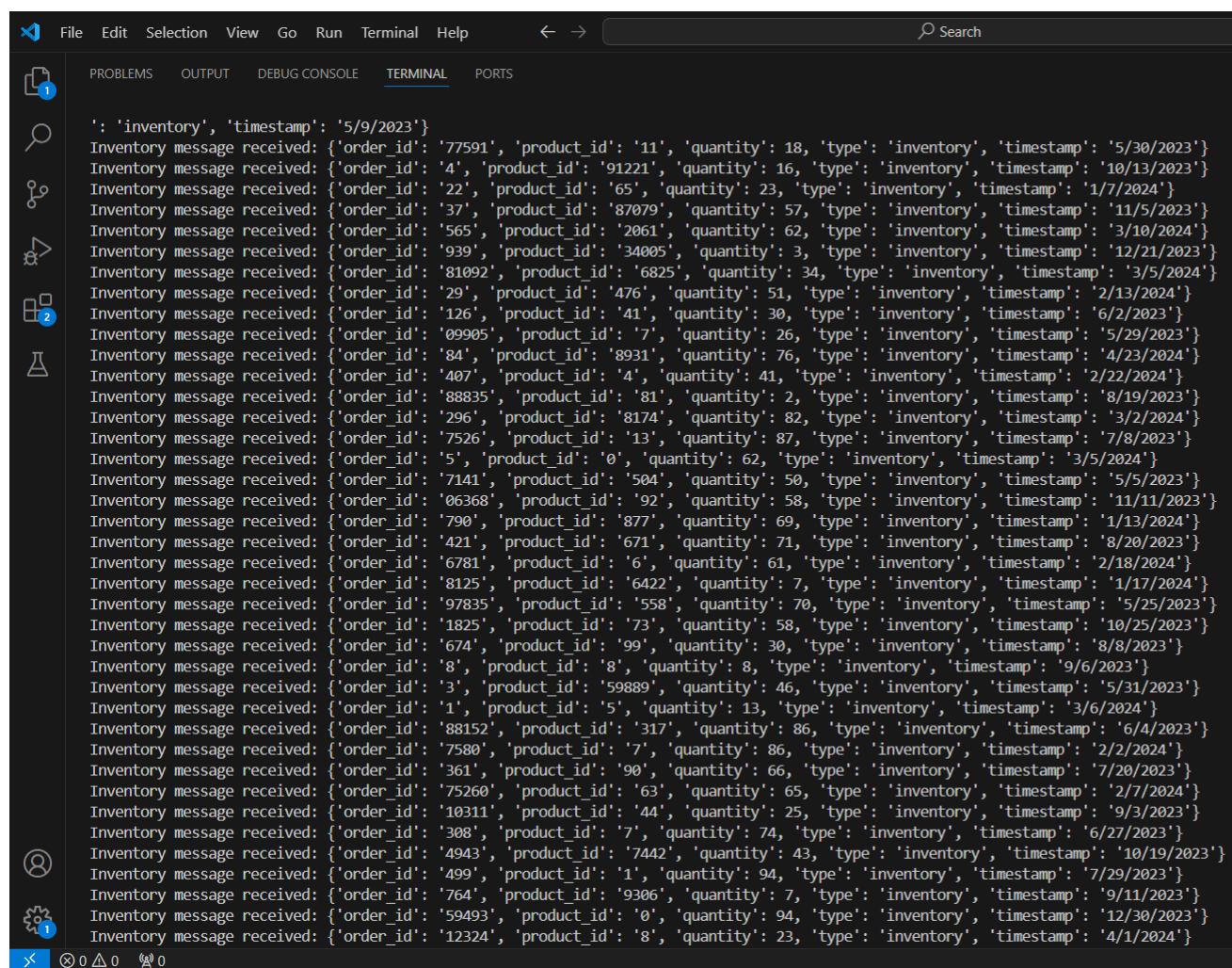
## 0.3 Producer 2: Delivery Orders Producer

In managing delivery orders, a Python script dubbed producer2.py was crafted. It parsed JSON data from data.json and underwent filtration via a designated function. Following this, a Kafka producer was initiated to transmit the refined delivery messages to the

allocated Kafka topic, thus enhancing the efficiency of order delivery management within the system.

## 0.4 Consumer 1: Inventory Data Consumer

To oversee inventory data, a Python script titled `consumer1.py` was devised. It commenced by initializing a Kafka consumer to subscribe to the assigned Kafka topic. The script then undertook the task of processing incoming messages, specifically filtering for inventory data, and subsequently executed actions to synchronize and update inventory databases or systems as needed.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

': 'inventory', 'timestamp': '5/9/2023'}
Inventory message received: {'order_id': '77591', 'product_id': '11', 'quantity': 18, 'type': 'inventory', 'timestamp': '5/30/2023'}
Inventory message received: {'order_id': '4', 'product_id': '91221', 'quantity': 16, 'type': 'inventory', 'timestamp': '10/13/2023'}
Inventory message received: {'order_id': '22', 'product_id': '65', 'quantity': 23, 'type': 'inventory', 'timestamp': '1/7/2024'}
Inventory message received: {'order_id': '37', 'product_id': '87079', 'quantity': 57, 'type': 'inventory', 'timestamp': '11/5/2023'}
Inventory message received: {'order_id': '565', 'product_id': '2061', 'quantity': 62, 'type': 'inventory', 'timestamp': '3/10/2024'}
Inventory message received: {'order_id': '939', 'product_id': '34005', 'quantity': 3, 'type': 'inventory', 'timestamp': '12/21/2023'}
Inventory message received: {'order_id': '81092', 'product_id': '6825', 'quantity': 34, 'type': 'inventory', 'timestamp': '3/5/2024'}
Inventory message received: {'order_id': '29', 'product_id': '476', 'quantity': 51, 'type': 'inventory', 'timestamp': '2/13/2024'}
Inventory message received: {'order_id': '126', 'product_id': '41', 'quantity': 30, 'type': 'inventory', 'timestamp': '6/2/2023'}
Inventory message received: {'order_id': '09905', 'product_id': '7', 'quantity': 26, 'type': 'inventory', 'timestamp': '5/29/2023'}
Inventory message received: {'order_id': '84', 'product_id': '8931', 'quantity': 76, 'type': 'inventory', 'timestamp': '4/23/2024'}
Inventory message received: {'order_id': '407', 'product_id': '4', 'quantity': 41, 'type': 'inventory', 'timestamp': '2/22/2024'}
Inventory message received: {'order_id': '88835', 'product_id': '81', 'quantity': 2, 'type': 'inventory', 'timestamp': '8/19/2023'}
Inventory message received: {'order_id': '296', 'product_id': '8174', 'quantity': 82, 'type': 'inventory', 'timestamp': '3/2/2024'}
Inventory message received: {'order_id': '7526', 'product_id': '13', 'quantity': 87, 'type': 'inventory', 'timestamp': '7/8/2023'}
Inventory message received: {'order_id': '5', 'product_id': '0', 'quantity': 62, 'type': 'inventory', 'timestamp': '3/5/2024'}
Inventory message received: {'order_id': '7141', 'product_id': '504', 'quantity': 50, 'type': 'inventory', 'timestamp': '5/5/2023'}
Inventory message received: {'order_id': '06368', 'product_id': '92', 'quantity': 58, 'type': 'inventory', 'timestamp': '11/11/2023'}
Inventory message received: {'order_id': '790', 'product_id': '877', 'quantity': 69, 'type': 'inventory', 'timestamp': '1/13/2024'}
Inventory message received: {'order_id': '421', 'product_id': '671', 'quantity': 71, 'type': 'inventory', 'timestamp': '8/20/2023'}
Inventory message received: {'order_id': '6781', 'product_id': '6', 'quantity': 61, 'type': 'inventory', 'timestamp': '2/18/2024'}
Inventory message received: {'order_id': '8125', 'product_id': '6422', 'quantity': 7, 'type': 'inventory', 'timestamp': '1/17/2024'}
Inventory message received: {'order_id': '97835', 'product_id': '558', 'quantity': 70, 'type': 'inventory', 'timestamp': '5/25/2023'}
Inventory message received: {'order_id': '1825', 'product_id': '73', 'quantity': 58, 'type': 'inventory', 'timestamp': '10/25/2023'}
Inventory message received: {'order_id': '674', 'product_id': '99', 'quantity': 30, 'type': 'inventory', 'timestamp': '8/8/2023'}
Inventory message received: {'order_id': '8', 'product_id': '8', 'quantity': 8, 'type': 'inventory', 'timestamp': '9/6/2023'}
Inventory message received: {'order_id': '3', 'product_id': '59889', 'quantity': 46, 'type': 'inventory', 'timestamp': '5/31/2023'}
Inventory message received: {'order_id': '1', 'product_id': '5', 'quantity': 13, 'type': 'inventory', 'timestamp': '3/6/2024'}
Inventory message received: {'order_id': '88152', 'product_id': '317', 'quantity': 86, 'type': 'inventory', 'timestamp': '6/4/2023'}
Inventory message received: {'order_id': '7580', 'product_id': '7', 'quantity': 86, 'type': 'inventory', 'timestamp': '2/2/2024'}
Inventory message received: {'order_id': '361', 'product_id': '90', 'quantity': 66, 'type': 'inventory', 'timestamp': '7/20/2023'}
Inventory message received: {'order_id': '75260', 'product_id': '63', 'quantity': 65, 'type': 'inventory', 'timestamp': '2/7/2024'}
Inventory message received: {'order_id': '10311', 'product_id': '44', 'quantity': 25, 'type': 'inventory', 'timestamp': '9/3/2023'}
Inventory message received: {'order_id': '308', 'product_id': '7', 'quantity': 74, 'type': 'inventory', 'timestamp': '6/27/2023'}
Inventory message received: {'order_id': '4943', 'product_id': '7442', 'quantity': 43, 'type': 'inventory', 'timestamp': '10/19/2023'}
Inventory message received: {'order_id': '499', 'product_id': '1', 'quantity': 94, 'type': 'inventory', 'timestamp': '7/29/2023'}
Inventory message received: {'order_id': '764', 'product_id': '9306', 'quantity': 7, 'type': 'inventory', 'timestamp': '9/11/2023'}
Inventory message received: {'order_id': '59493', 'product_id': '0', 'quantity': 94, 'type': 'inventory', 'timestamp': '12/30/2023'}
Inventory message received: {'order_id': '12324', 'product_id': '8', 'quantity': 23, 'type': 'inventory', 'timestamp': '4/1/2024'}
```

Figure 3: Result of Consumer1

## 0.5 Consumer 2: Delivery Data Consumer

For managing delivery tasks, a Python script dubbed `consumer2.py` was engineered. It kicked off by initializing a Kafka consumer to subscribe to the designated Kafka topic. Upon receiving incoming messages, the script diligently filtered for delivery data, thereafter executing a series of actions including scheduling deliveries, updating statuses, and promptly notifying customers as required.

```
Delivery message received: {'order_id': '819', 'product_id': '50', 'quantity': 60, 'type': 'delivery', 'timestamp': '2/14/2024'}
Delivery message received: {'order_id': '18', 'product_id': '01888', 'quantity': 67, 'type': 'delivery', 'timestamp': '3/30/2024'}
Delivery message received: {'order_id': '4', 'product_id': '264', 'quantity': 16, 'type': 'delivery', 'timestamp': '5/20/2023'}
Delivery message received: {'order_id': '189', 'product_id': '92886', 'quantity': 58, 'type': 'delivery', 'timestamp': '12/2/2023'}
Delivery message received: {'order_id': '4', 'product_id': '794', 'quantity': 9, 'type': 'delivery', 'timestamp': '5/8/2023'}
Delivery message received: {'order_id': '65250', 'product_id': '64', 'quantity': 17, 'type': 'delivery', 'timestamp': '12/7/2023'}
Delivery message received: {'order_id': '96', 'product_id': '9065', 'quantity': 85, 'type': 'delivery', 'timestamp': '2/27/2024'}
Delivery message received: {'order_id': '41', 'product_id': '45247', 'quantity': 7, 'type': 'delivery', 'timestamp': '3/6/2024'}
Delivery message received: {'order_id': '605', 'product_id': '61607', 'quantity': 72, 'type': 'delivery', 'timestamp': '3/4/2024'}
Delivery message received: {'order_id': '2', 'product_id': '45', 'quantity': 28, 'type': 'delivery', 'timestamp': '6/13/2023'}
Delivery message received: {'order_id': '4973', 'product_id': '8', 'quantity': 51, 'type': 'delivery', 'timestamp': '2/2/2024'}
Delivery message received: {'order_id': '84', 'product_id': '1354', 'quantity': 37, 'type': 'delivery', 'timestamp': '11/10/2023'}
Delivery message received: {'order_id': '16', 'product_id': '08', 'quantity': 26, 'type': 'delivery', 'timestamp': '5/1/2023'}
Delivery message received: {'order_id': '651', 'product_id': '25', 'quantity': 45, 'type': 'delivery', 'timestamp': '10/11/2023'}
Delivery message received: {'order_id': '89649', 'product_id': '5618', 'quantity': 63, 'type': 'delivery', 'timestamp': '8/31/2023'}
Delivery message received: {'order_id': '2272', 'product_id': '38', 'quantity': 94, 'type': 'delivery', 'timestamp': '7/19/2023'}
Delivery message received: {'order_id': '157', 'product_id': '564', 'quantity': 55, 'type': 'delivery', 'timestamp': '1/30/2024'}
Delivery message received: {'order_id': '73345', 'product_id': '7', 'quantity': 80, 'type': 'delivery', 'timestamp': '11/16/2023'}
Delivery message received: {'order_id': '34', 'product_id': '44827', 'quantity': 51, 'type': 'delivery', 'timestamp': '4/18/2024'}
Delivery message received: {'order_id': '766', 'product_id': '69', 'quantity': 86, 'type': 'delivery', 'timestamp': '6/16/2023'}
Delivery message received: {'order_id': '27116', 'product_id': '91', 'quantity': 75, 'type': 'delivery', 'timestamp': '1/19/2024'}
Delivery message received: {'order_id': '29', 'product_id': '2', 'quantity': 88, 'type': 'delivery', 'timestamp': '3/12/2024'}
Delivery message received: {'order_id': '7', 'product_id': '13', 'quantity': 7, 'type': 'delivery', 'timestamp': '5/18/2023'}
Delivery message received: {'order_id': '401', 'product_id': '7', 'quantity': 19, 'type': 'delivery', 'timestamp': '5/1/2023'}
Delivery message received: {'order_id': '864', 'product_id': '6', 'quantity': 17, 'type': 'delivery', 'timestamp': '11/16/2023'}
Delivery message received: {'order_id': '910', 'product_id': '5378', 'quantity': 18, 'type': 'delivery', 'timestamp': '3/7/2024'}
Delivery message received: {'order_id': '73', 'product_id': '125', 'quantity': 39, 'type': 'delivery', 'timestamp': '12/21/2023'}
Delivery message received: {'order_id': '49', 'product_id': '44783', 'quantity': 79, 'type': 'delivery', 'timestamp': '6/14/2023'}
Delivery message received: {'order_id': '67', 'product_id': '36', 'quantity': 33, 'type': 'delivery', 'timestamp': '6/18/2023'}
Delivery message received: {'order_id': '80757', 'product_id': '769', 'quantity': 19, 'type': 'delivery', 'timestamp': '9/7/2023'}
Delivery message received: {'order_id': '5', 'product_id': '242', 'quantity': 13, 'type': 'delivery', 'timestamp': '8/28/2023'}
Delivery message received: {'order_id': '7716', 'product_id': '585', 'quantity': 13, 'type': 'delivery', 'timestamp': '4/8/2024'}
Delivery message received: {'order_id': '8514', 'product_id': '41', 'quantity': 36, 'type': 'delivery', 'timestamp': '1/18/2024'}
Delivery message received: {'order_id': '41562', 'product_id': '38', 'quantity': 50, 'type': 'delivery', 'timestamp': '10/3/2023'}
Delivery message received: {'order_id': '76', 'product_id': '26', 'quantity': 65, 'type': 'delivery', 'timestamp': '5/30/2023'}
Delivery message received: {'order_id': '7125', 'product_id': '737', 'quantity': 100, 'type': 'delivery', 'timestamp': '6/18/2023'}
Delivery message received: {'order_id': '8', 'product_id': '64345', 'quantity': 55, 'type': 'delivery', 'timestamp': '10/19/2023'}
Delivery message received: {'order_id': '2', 'product_id': '74377', 'quantity': 54, 'type': 'delivery', 'timestamp': '12/11/2023'}
Delivery message received: {'order_id': '769', 'product_id': '5528', 'quantity': 66, 'type': 'delivery', 'timestamp': '3/5/2024'}
```

Figure 4: Result of Consumer2

## 0.6 Filter Logic

A Python script named filterlogic.py was created to showcase message filtering logic using the supplied JSON data. The script loaded JSON data from the provided file data.json and defined separate functions to filter inventory and delivery messages. It then printed the filtered inventory and delivery messages for verification purposes.

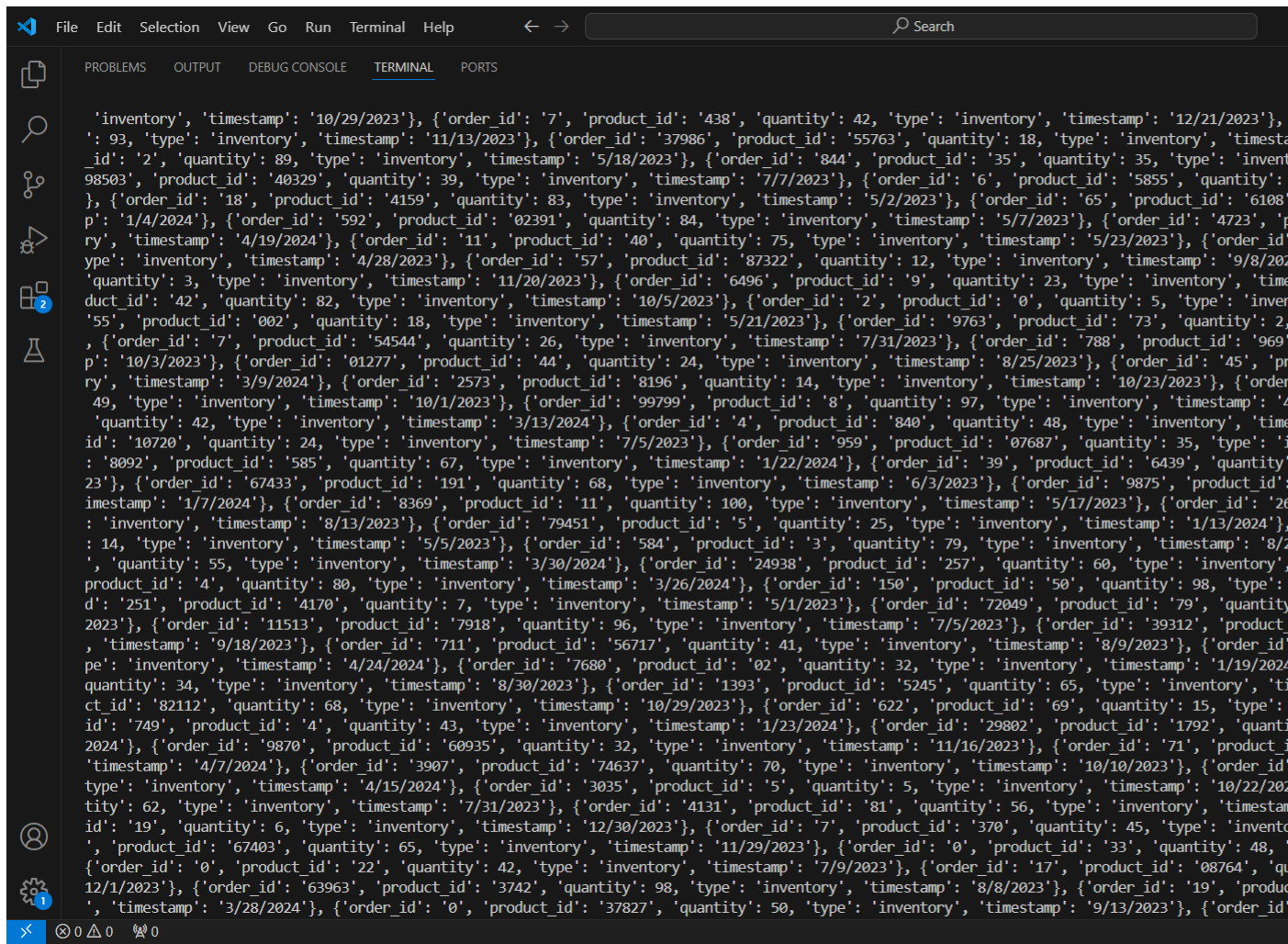


Figure 5: Filter Logic