Name: Tanaz Pathan
Student ID: 202318056

Real-Time E-commerce Order Processing System Using Kafka

Introduction:

This report outlines the development of a Kafka-based system for managing e-commerce orders in real time. The system handles inventory management anddelivery processing by implementing Kafka producers and consumers.

Kafka Installation:

The setup involved installing the necessary libraries and configuring Kafka producers tosend messages to Kafka topics.

- Installed `kafka-python` and `confluent-kafka` libraries using pip.
- Configured Kafka producers for inventory orders and delivery orders.
- Defined a function to acknowledge message delivery and flush messages toensure they are sent.

Producer Implementation:

Two Kafka producers were implemented to send messages for inventory orders anddelivery orders. Each producer sends messages with a specific type to the respectiveKafka topics.

- Implemented a producer for inventory orders to send messages to the 'inventory_topic'.
- Implemented a producer for delivery orders to send messages to the 'delivery_topic'.

BDP_Kafka_assgn.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[3]

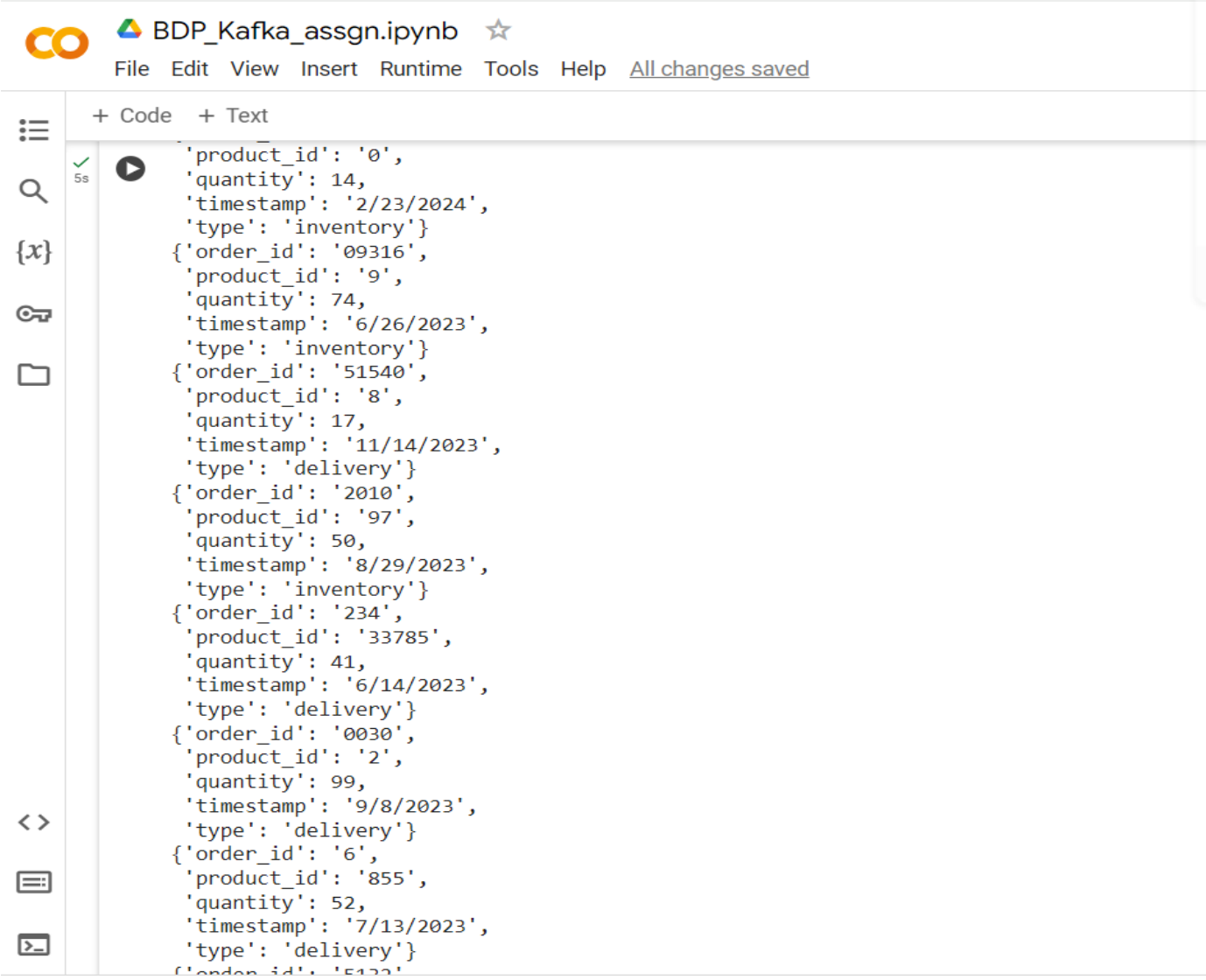
```python
conf = {
    'bootstrap.servers': "localhost:9092",
    'client.id': socket.gethostname()
}

# Function to acknowledge message delivery
def acked(err, msg):
    if err is not None:
        print("Failed to deliver message: %s: %s" % (str(msg), str(err)))
    else:
        print("Message produced: %s" % (str(msg)))

# Inventory Orders Producer
producer_inventory = Producer(conf)
def send_inventory_message(data):
    producer_inventory.produce('inventory_topic', key=str(data['order_id']), value=str(data), callback=acked)
    producer_inventory.poll(0)

# Delivery Orders Producer
producer_delivery = Producer(conf)
def send_delivery_message(data):
    producer_delivery.produce('delivery_topic', key=str(data['order_id']), value=str(data), callback=acked)
    producer_delivery.poll(0)

# Ensure all messages are sent
producer_inventory.flush()
producer_delivery.flush()
```

```
0
```

Message Sending:

Messages were sent to Kafka topics using the implemented producers. Each messagecontains order details such as order ID, product ID, quantity, and timestamp.

- Inventory messages were sent to the 'inventory_topic'.
- Delivery messages were sent to the 'delivery_topic'.



Consumer Implementation:

Consumers were implemented to consume messages from Kafka topics for inventorydata and delivery data. Each consumer listens to its respective Kafka topic and processes incoming messages.

- Implemented consumers for inventory data and delivery data.
- Subscribed consumers to the 'inventory_topic' and 'delivery_topic', respectively.
- Defined functions to consume and process incoming messages.

BDP_Kafka_assgn.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5]
```
consumer_delivery.subscribe(['delivery_topic'])

def consume_delivery():
    while True:
        msg = consumer_delivery.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
            else:
                print(msg.error())
                break
        print('Received delivery message: {}'.format(msg.value()))

# Start consuming (typically in separate threads or processes)
inventory_thread = threading.Thread(target=consume_inventory)
delivery_thread = threading.Thread(target=consume_delivery)

inventory_thread.start()
delivery_thread.start()
```

[6]
```
# Inventory data consumer
consumer_inventory = Consumer(c_conf)
consumer_inventory.subscribe(['inventory_topic'])

# Delivery data consumer
consumer_delivery = Consumer(c_conf)
consumer_delivery.subscribe(['delivery_topic'])
```

Message Processing:

Incoming messages were processed by the consumers to update inventory databases,schedule deliveries, update delivery status, and notify customers as per the message type.

- Inventory data consumers processed messages from the 'inventory_topic'.
- Delivery data consumers processed messages from the 'delivery_topic'.

BDP_Kafka_assgn.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5]

```
            msg = consumer_inventory.poll(1.0)
            if msg is None:
                continue
            if msg.error():
                if msg.error().code() == KafkaError._PARTITION_EOF:
                    continue
                else:
                    print(msg.error())
                    break
            print('Received inventory message: {}'.format(msg.value()))

    # Delivery Data Consumer
    consumer_delivery = Consumer(c_conf)
    consumer_delivery.subscribe(['delivery_topic'])

    def consume_delivery():
        while True:
            msg = consumer_delivery.poll(1.0)
            if msg is None:
                continue
            if msg.error():
                if msg.error().code() == KafkaError._PARTITION_EOF:
                    continue
                else:
                    print(msg.error())
                    break
            print('Received delivery message: {}'.format(msg.value()))

    # Start consuming (typically in separate threads or processes)
    inventory_thread = threading.Thread(target=consume_inventory)
    delivery_thread = threading.Thread(target=consume_delivery)
```

✓ 0s completed at 10:53 PM

Conclusion:

The real-time e-commerce order processing system using Kafka was successfully implemented. Kafka producers and consumers were configured to handle inventory anddelivery orders, ensuring efficient management of e-commerce operations.