Submitted by: Anurag Choudhury
ID - (202318059)

Assignment Report on
Real-Time E-commerce Order Processing System Using Kafka

To develop a Kafka-based system for managing e-commerce orders in real-time, you'll need to set up producers, consumers, and implement message filtering logic. Below are the steps you can follow to achieve this:

Step 1: Set Up Kafka

1. Install Kafka: I ensure that Kafka is installed and running on my system or a server.
2. Create Kafka Topics: I create Kafka topics named inventory_orders and delivery_orders for each producer to send messages to.

Step 2: Implement Kafka Producers

1. Inventory Orders Producer (inventory_orders_producer):
    - I develop a producer that filters messages where the type field is inventory.
    - I implement a Kafka producer that reads inventory-related events from a data source and sends messages with type set to inventory to the inventory_orders topic.
2. Delivery Orders Producer (delivery_orders_producer):
    - I develop a producer that filters messages where the type field is delivery.
    - I create a Kafka producer that reads delivery-related events and sends messages with type set to delivery to the delivery_orders topic.

Step 3: Implement Kafka Consumers

1. Inventory Data Consumer (inventory_data_consumer):
    - I configure a Kafka consumer that subscribes to the inventory_orders topic.
    - I implement logic to process inventory messages received by updating inventory databases or systems accordingly.
2. Delivery Data Consumer (delivery_data_consumer):
    - I set up a Kafka consumer for the delivery_orders topic.
    - I develop logic to handle delivery-related messages such as scheduling deliveries, updating delivery status, and notifying customers.

Step 4: Develop Message Filtering Logic

1. Producer Message Filtering:
    - I implement logic within each producer (inventory_orders_producer and

delivery_orders_producer) to filter messages based on the type field from the incoming data source.

- I ensure that only messages matching the desired type (i.e., inventory or delivery) are sent to Kafka.

Additional Considerations

- Error Handling: I implement error handling within producers and consumers to manage exceptions or failed operations gracefully.
- Scalability: I design my system to handle increasing loads by considering Kafka partitioning, consumer groups, and scaling strategies.
- Monitoring and Logging: I utilize Kafka monitoring tools and logging frameworks to monitor system performance and troubleshoot issues effectively.

By following these steps and best practices, I'll be able to develop a robust Kafka-based e-commerce order management system capable of real-time inventory management and delivery processing.