Kafka Project Question Paper: eCommerce Scenario

Objective:  
Design and implement a Kafka-based messaging system for an eCommerce platform. The task will involve creating Kafka producers, consumers, topics, and applying partitioning logic to scale and distribute data efficiently.

Scenario:

You are building a Kafka-based messaging system for an eCommerce platform that handles the following key operations:

1. Order Placement: Customers place orders for products.
2. Inventory Updates: Updates on product stock levels after orders are placed.
3. Payment Processing: Payment information is processed after an order is placed.
4. Shipping Information: Shipping updates based on payment status.

Your task is to simulate this eCommerce flow using Kafka.

### **Tasks:**

1. Kafka Producer Creation:

* Implement Kafka producers to publish the following types of events:
  + OrderPlacedEvent: Contains information such as order ID, customer ID, product IDs, quantity, and order total.
  + InventoryUpdateEvent: Contains information about the product ID and updated stock quantity.
  + PaymentProcessingEvent: Contains payment information including order ID, customer ID, payment status, and transaction ID.
  + ShippingUpdateEvent: Contains order ID, shipping status, tracking number, and customer shipping address.

For each event type, create a Kafka producer that will send the data to the appropriate Kafka topic.

2. Kafka Topic Creation and Partitioning Logic:

* Create separate Kafka topics for each of the event types mentioned above.
  + Topics:
    - order-placed
    - inventory-updated
    - payment-processed
    - shipping-updated
* Partitioning Logic:
  + Implement partitioning logic to ensure that related events are processed in parallel but in an ordered manner. For example:
    - Use the order ID as the key for partitioning order-placed and payment-processed topics, ensuring that all events related to a particular order are sent to the same partition.
    - Use the product ID as the key for partitioning the inventory-updated topic, ensuring that updates for the same product go to the same partition.

3. Kafka Consumer Implementation:

* Implement Kafka consumers to subscribe to the following topics:
  + OrderConsumer: Consumes events from the order-placed topic. This consumer will process new orders and pass them to the next step in the system (e.g., payment processing or inventory check).
  + InventoryConsumer: Consumes events from the inventory-updated topic. This consumer will handle inventory updates (e.g., stock replenishment or reductions based on orders).
  + PaymentConsumer: Consumes events from the payment-processed topic. This consumer will handle payment statuses and pass payment success or failure to the shipping system.
  + ShippingConsumer: Consumes events from the shipping-updated topic. This consumer will monitor shipping status and update the system with tracking information.

4. Handling Kafka Consumer Offsets:

* Implement proper offset management for the consumers. You should implement either manual offset commit or auto-commit based on your understanding of Kafka’s offset handling. The system should ensure that no events are lost, and that events can be reprocessed in case of failures.

5. Error Handling and Logging:

* Implement error handling for your consumers and producers. Log all significant events such as:
  + Successfully processed events.
  + Any failed attempts to produce or consume messages.
  + Invalid events or data.

6. System Scaling Consideration (Bonus):

* The system should be scalable. Provide a brief explanation (and if possible, demonstrate) how you would scale this system to handle millions of orders by:
  + Scaling Kafka brokers.
  + Adding more partitions to the topics.
  + Running multiple consumers for parallel processing.

### **Deliverables:**

1. Kafka Producer Code:
   * Implement producers for all four event types (Order, Inventory, Payment, Shipping).
2. Kafka Consumer Code:
   * Implement consumers for each of the event types.
3. Kafka Topic & Partition Design:
   * Provide a description of the Kafka topics you created, their partitioning strategy, and why you chose it.
4. Configuration Files:
   * Provide the configuration files for your Kafka producers and consumers, including Kafka connection details and partitioning logic.
5. Error Handling:
   * Include the error handling and logging mechanisms in your implementation.
6. Documentation:
   * Provide a brief documentation (1–2 pages) explaining your system design, the partitioning logic, and how you managed offsets. Also, describe the scalability plan.

### **Evaluation Criteria:**

* Producer & Consumer Implementation: Correctness and efficiency of the producer-consumer pattern.
* Topic & Partitioning Design: Justification of partitioning logic and how it optimizes performance.
* Error Handling: How errors are managed in both producers and consumers.
* Scalability Considerations: How well the system can be scaled for high traffic and data volumes.
* Code Quality: Cleanliness, readability, and structure of the code.
* Documentation: Clarity and completeness of the explanation provided.