Kafka Assignment

Real-Time Data Processing with Confluent Kafka, MySQL, and Avro

Objective:
In this assignment, you will build a Kafka producer and a consumer group that work with a MySQL database, Avro serialization, and multi-partition Kafka topics. The producer will fetch incremental data from a MySQL table and write Avro serialized data into a Kafka topic. The consumers will deserialize this data and append it to separate JSON files.

Tools Required:

- Python 3.7 or later
- Confluent Kafka Python client
- MySQL Database
- Apache Avro
- A suitable IDE for coding (e.g., PyCharm, Visual Studio Code)

Background:
You are working in a fictitious e-commerce company called **"BuyOnline"** which has a MySQL database that stores product information such as product ID, name, category, price, and updated timestamp. The database gets updated frequently with new products and changes in product information. The company wants to build a real-time system to stream these updates incrementally to a downstream system for real-time analytics and business intelligence.

Steps:

Step 1: MySQL Table Setup
Create a table named 'product' in MySQL database with the following columns:

➔ id - INT (Primary Key)
➔ name - VARCHAR
➔ category - VARCHAR

➔ price - FLOAT
➔ last_updated - TIMESTAMP

Step 2: Kafka Producer

- Write a Kafka producer in Python that uses a MySQL connector to fetch data from the MySQL table.
- In your producer code, maintain a record of the last read timestamp. Each time you fetch data, use a SQL query to get records where the last_updated timestamp is greater than the last read timestamp.
- Serialize the data into Avro format and publish the data to a Kafka topic named "product_updates". This topic should be configured with 10 partitions.
- Use the product ID as the key when producing messages. This will ensure that all updates for the same product end up in the same partition.
- Update the last read timestamp after each successful fetch.

Step 3: Kafka Consumer Group and Data Transformation

- Write a Kafka consumer in Python and set it up as a consumer group of 5 consumers.
- Each consumer should read data from the "product_updates" topic.
- Deserialize the Avro data back into a Python object.
- Implement data transformation logic. For example:
- Change the category column to uppercase.
- Apply some business logic to update the price column, such as applying a discount if a particular product falls under a specific category.

Step 4: Writing Transformed Data to JSON Files

- Each consumer should convert the transformed Python object into a JSON string and append the JSON string to a separate JSON file. Make sure to open the file in append mode.
- Also, ensure each new record is written in a new line for ease of reading.
- Close the file after writing to make sure all data is saved properly.

Deliverables:

- The source code for the Kafka producer and consumer group in Python.
- SQL queries used for incremental data fetch from MySQL.
- Avro schema used for data serialization.
- Data transformation logic.
- The resulting JSON files with appended records.
- Documentation explaining how to run and test the system.
- Screenshots demonstrating the successful running of your application.

Evaluation Criteria:

Your assignment will be evaluated based on the following:

- Correctness of the Python code.
- Use of Avro for data serialization.
- Successful incremental fetching of data from the MySQL table.
- Successful writing of data to the Kafka topic with proper partitioning.
- Successful setup of a Kafka consumer group.
- Correct transformation and writing of data to separate JSON files.
- Quality of your documentation.

Bonus Points:

For bonus points, you can also provide:

- A Dockerfile to run your application.
- Unit tests for your Python code.
- Any advanced data processing, like filtering or transforming the data before sending it to the downstream system.

Additional Instructions:

Remember to update your database with new product details while testing the producer, so that it can fetch incremental data based on the last_updated timestamp.