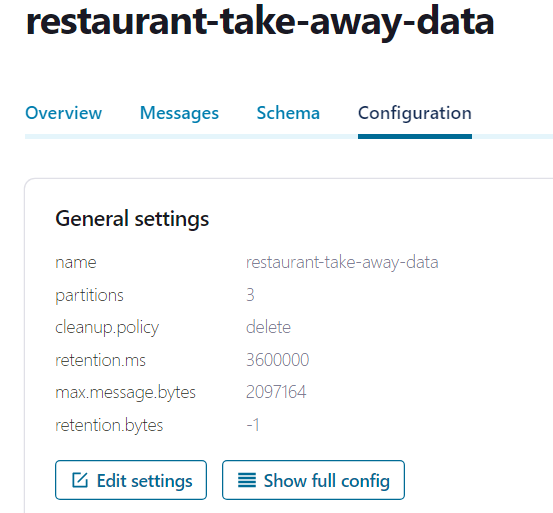
**Assignment-2 (Kafka first assignment)**

**Subham Sahoo (sksplay7@gmail.com)**

**1. Setup Confluent Kafka Account**

Done.

**2. Create one kafka topic named as "restaurant-take-away-data" with 3 partitions**



**3. Setup key (string) & value (json) schema in the confluent schema registry**

**Key :** “string”

**Value :**

{

"$id": "http://example.com/myURI.schema.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"additionalProperties": false,

"description": "Sample schema to help you get started.",

"properties": {

"item\_name": {

"description": "The type(v) type is used.",

"type": "string"

},

"order\_date": {

"description": "The type(v) type is used.",

"type": "string"

},

"order\_number": {

"description": "The type(v) type is used.",

"type": "number"

},

"product\_price": {

"description": "The type(v) type is used.",

"type": "number"

},

"quantity": {

"description": "The type(v) type is used.",

"type": "number"

},

"total\_products": {

"description": "The type(v) type is used.",

"type": "number"

}

},

"title": "SampleRecord",

"type": "object"

}

**4. Write a kafka producer program (python or any other language) to read data records from restaurant data csv file, make sure schema is not hardcoded in the producer code, read the latest version of schema and schema\_str from schema registry and use it for data serialization.**

<https://github.com/sksgit7/Big-data-bootcamp/blob/main/kafka-assignment1/kafka_json_producer.py>

**Used this for getting latest schema :**

topic = 'restaurant-take-away-data'

my\_schema = schema\_registry\_client.get\_latest\_version(topic+'-value').schema.schema\_str

**5. From producer code, publish data in Kafka Topic one by one and use dynamic key while publishing the records into the Kafka Topic.**

**6. Write kafka consumer code and create two copies of same consumer code and save it with different names (kafka\_consumer\_1.py & kafka\_consumer\_2.py), again make sure latest schema version and schema\_str is not hardcoded in the consumer code, read it automatically from the schema registry to desrialize the data.**

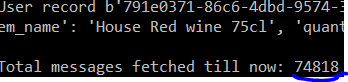
<https://github.com/sksgit7/Big-data-bootcamp/blob/main/kafka-assignment1/kafka_json_consumer1.py>

<https://github.com/sksgit7/Big-data-bootcamp/blob/main/kafka-assignment1/kafka_json_consumer2.py>

**Now test two scenarios with your consumer code:**

**a.) Use "group.id" property in consumer config for both consumers and mention different group\_ids in kafka\_consumer\_1.py & kafka\_consumer\_2.py, apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consume consumed and printed on the terminal**

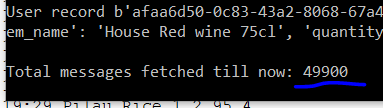
Each 74818 (all records)



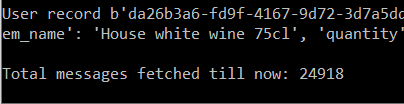
**b.) Use "group.id" property in consumer config for both consumers and mention same group\_ids in kafka\_consumer\_1.py & kafka\_consumer\_2.py, apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed and printed on the terminal**

49900 + 24918 = 74818 (total records in file except header)

consumer-1



Consumer-2



**7. Once above questions are done, write another kafka consumer to read data from kafka topic and from the consumer code create one csv file "output.csv" and append consumed records output.csv file.**

<https://github.com/sksgit7/Big-data-bootcamp/blob/main/kafka-assignment1/kafka_json_consumer_file.py>

Output file : <https://github.com/sksgit7/Big-data-bootcamp/blob/main/kafka-assignment1/output.csv>

