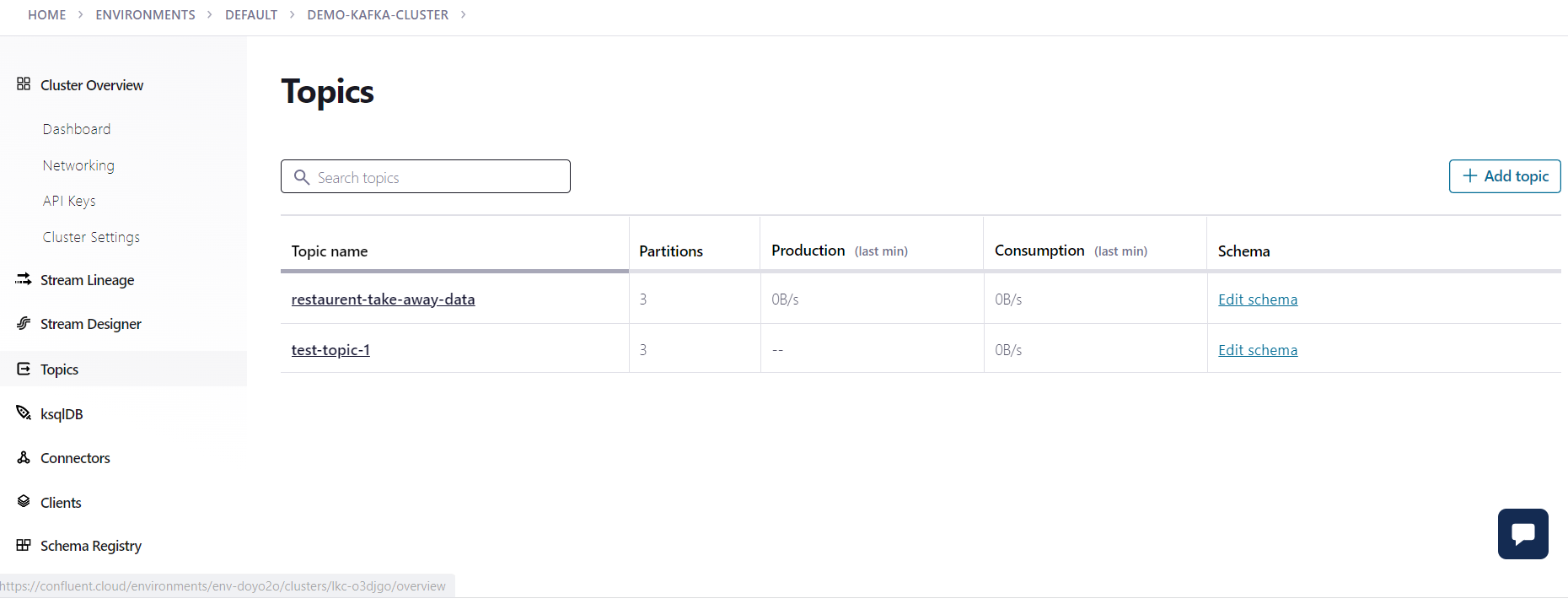
**Kafka Assignment**

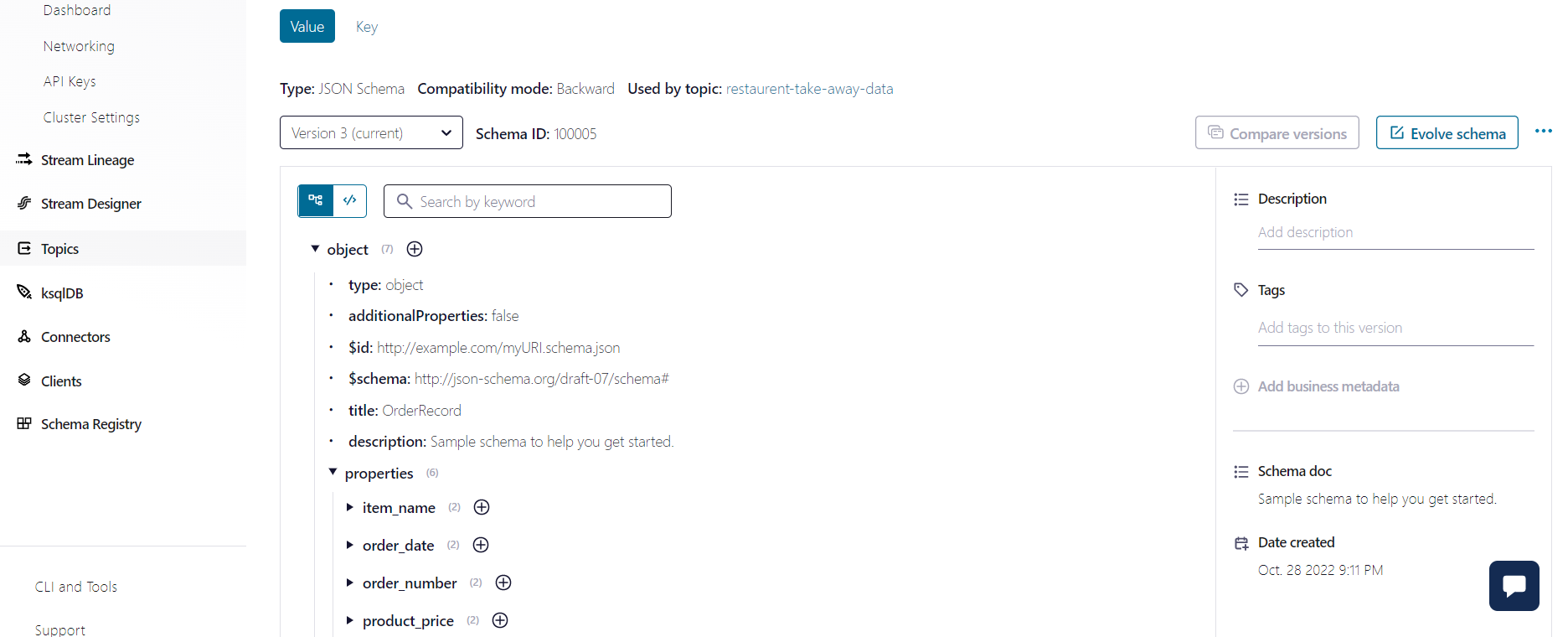
Download restaurant data from below mentioned link.

Download Data Link -> <https://github.com/shashank-mishra219/Confluent-Kafka-Setup/blob/main/restaurant_orders.csv>

1. Create one kafka topic named as "restaurent-take-away-data" with 3 partitions



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



Json format for 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"

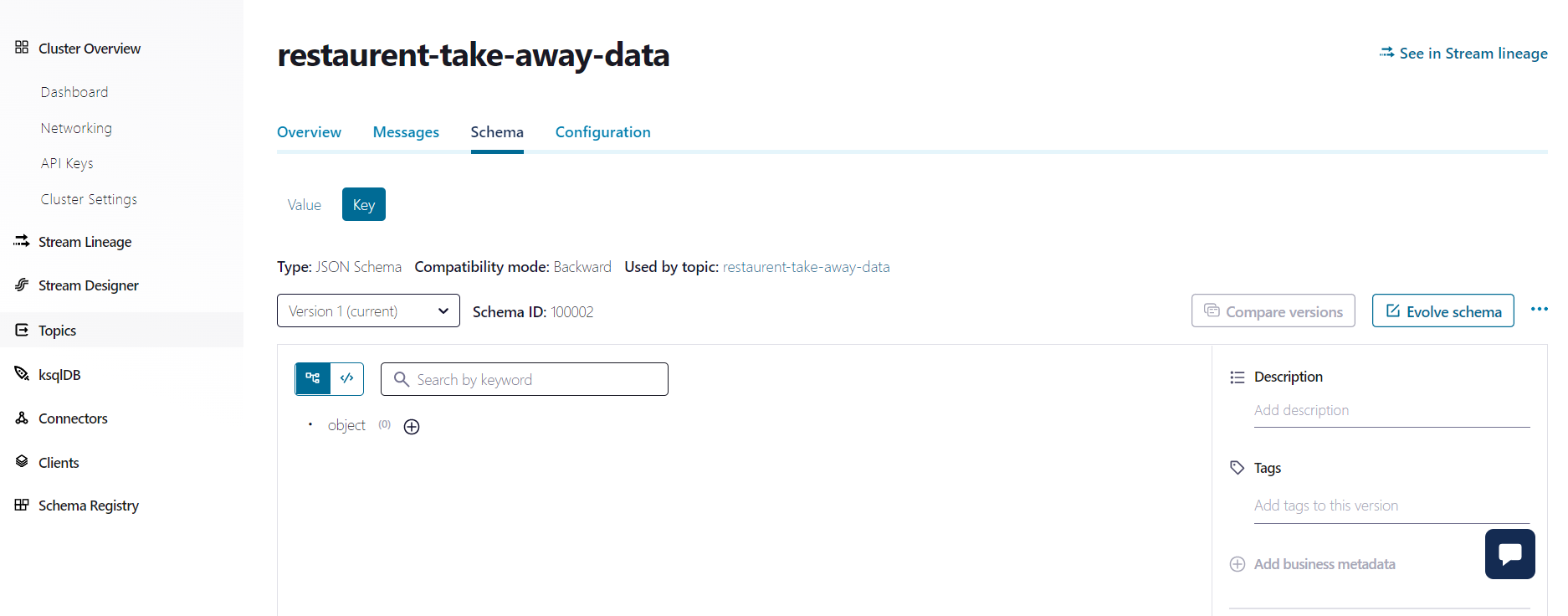
}

},

"title": "OrderRecord",

"type": "object"

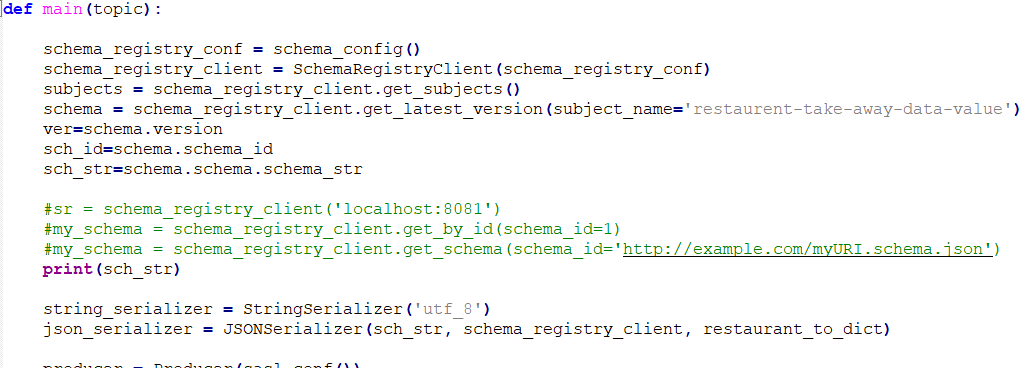
}



Json format for key:

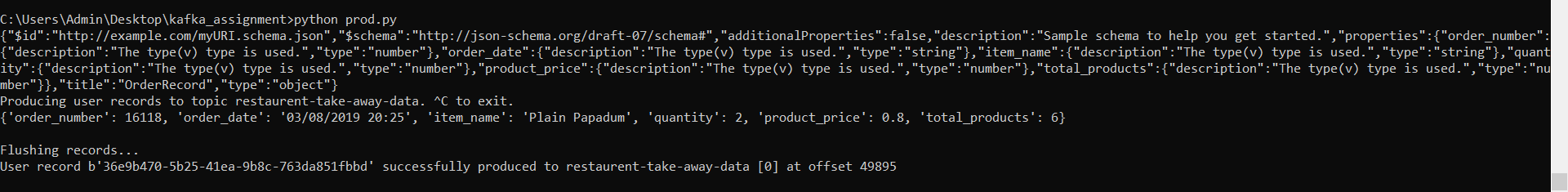
“string”

3. 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.



Use the get\_latest\_version function of SchemaRegistryClient and pass the subject\_name which will be the value parameter of the topic in our case “restaurant-take-away-data-value”.

Fetch the schema into sch\_str and use it in JSONSerializer.

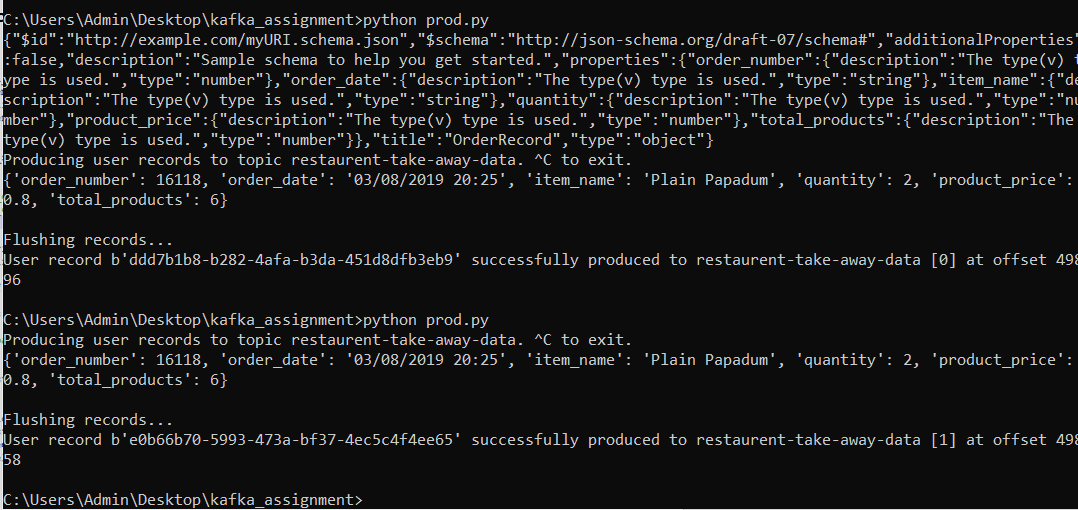


Above is the value of sch\_str which is printed, this gets the latest version of the schema from schema registry

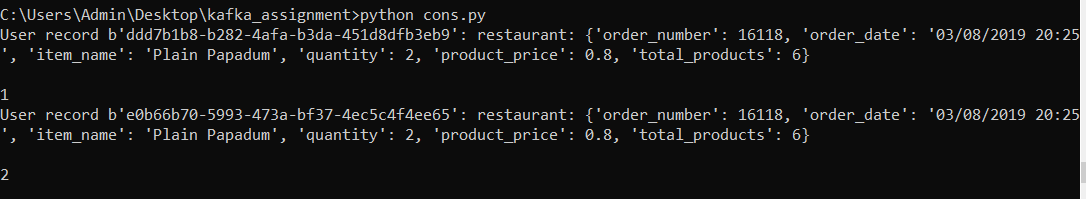
4. From producer code, publish data in Kafka Topic one by one and use dynamic key while publishing the records into the Kafka Topic.



A break statement is given where the producer juts publishes 1 record and the flushes it and the consumer consumes 1 record each time the producer produces.



Here as you can see I ran producer 2 times and it published 2 times the record.



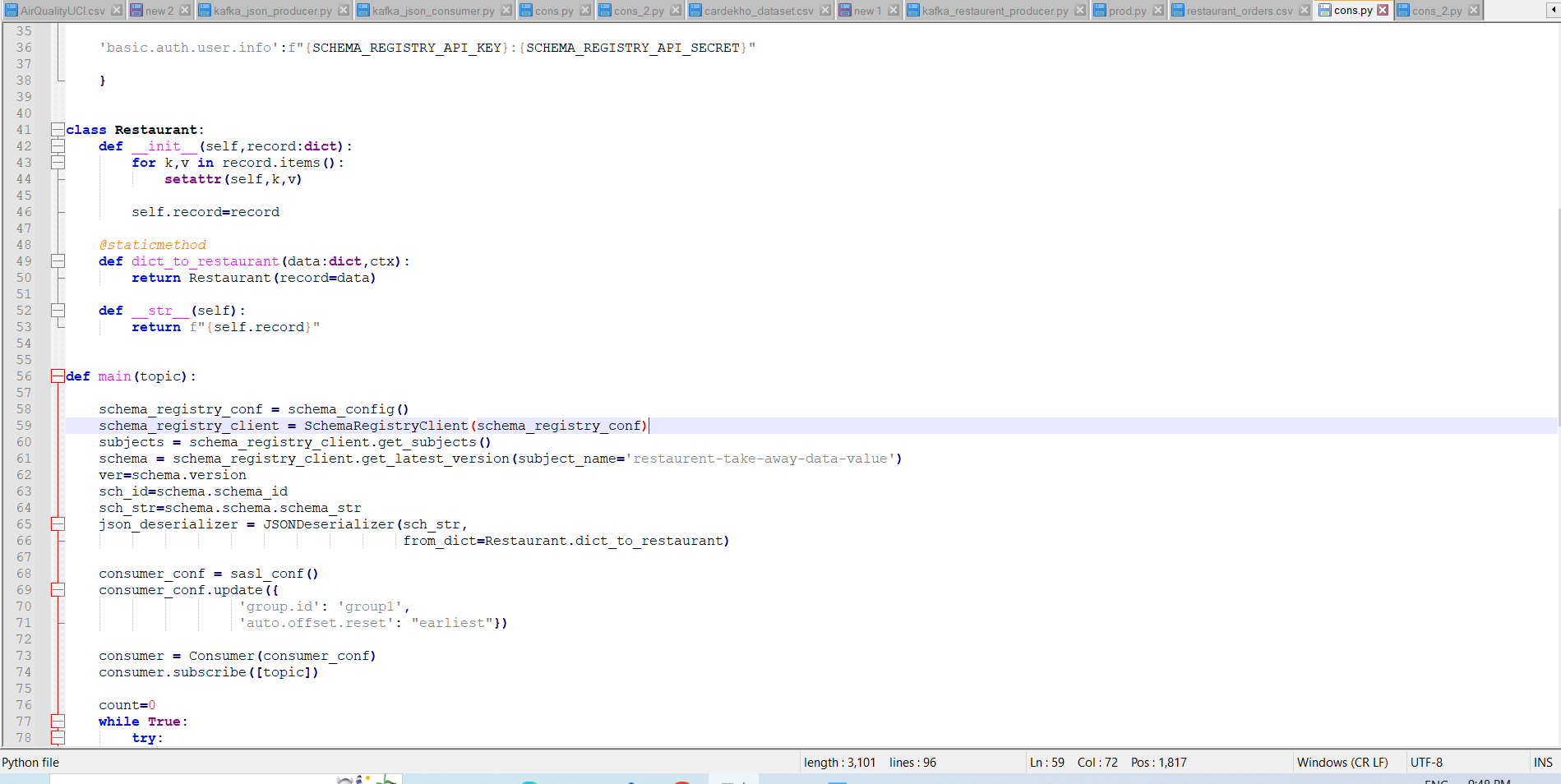
The consumer consumed the records each time one by one.

5. 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 deserialize the data.

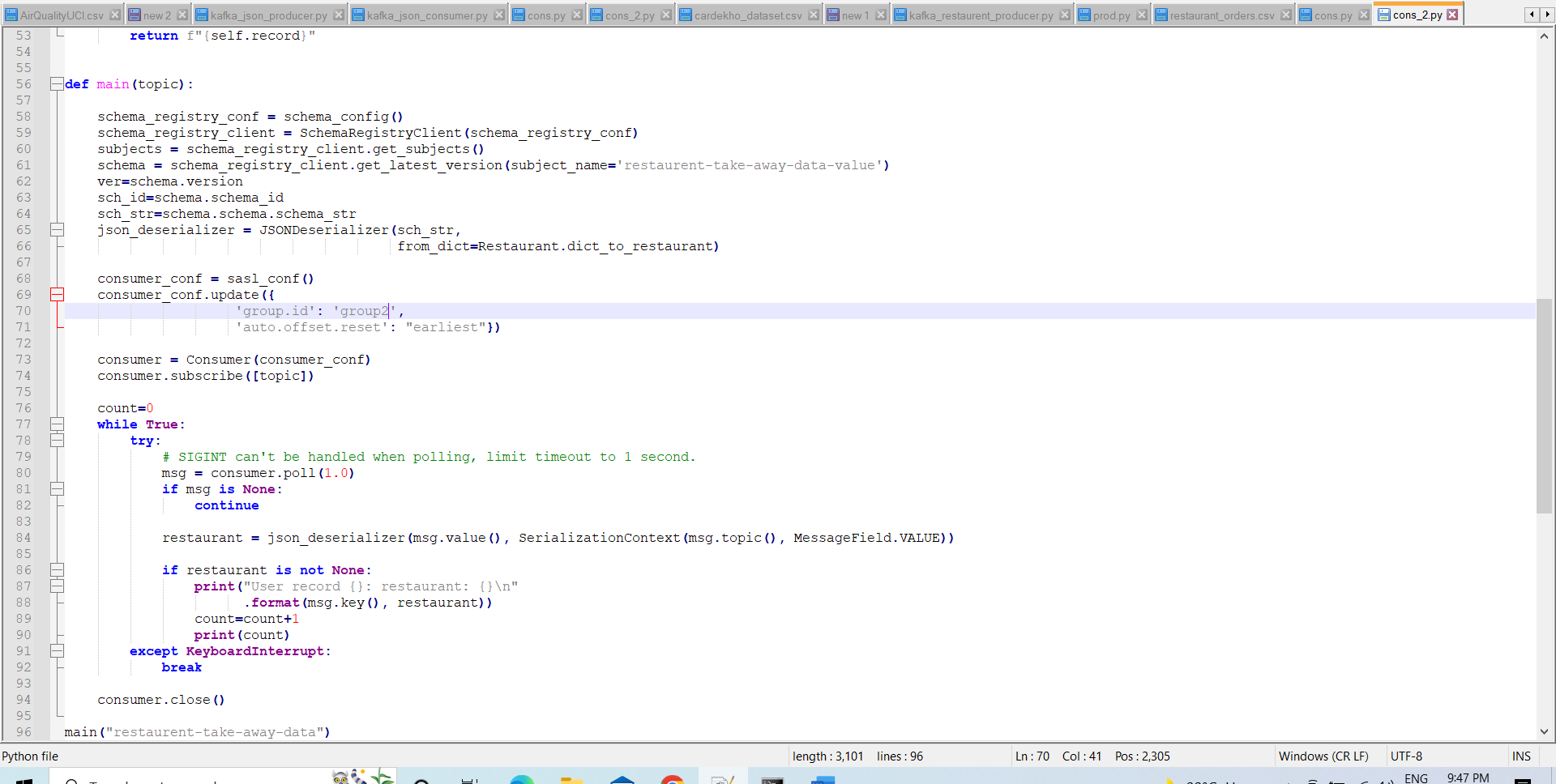
Now test two scenarios with your consumer code:

1. 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 consumer consumed and printed on the terminal.

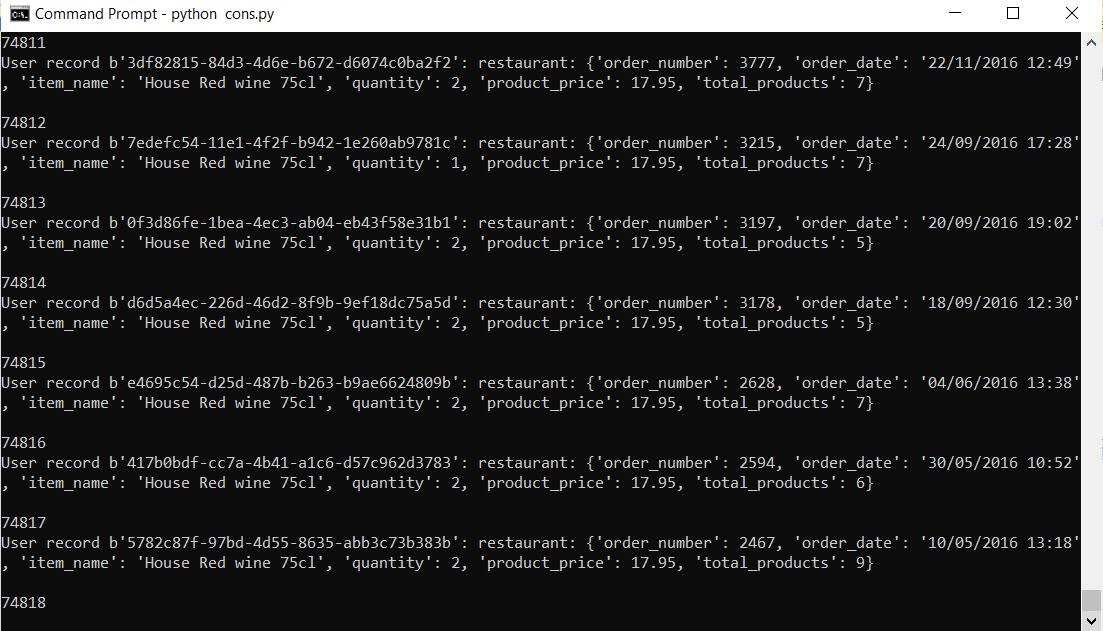
cons.py

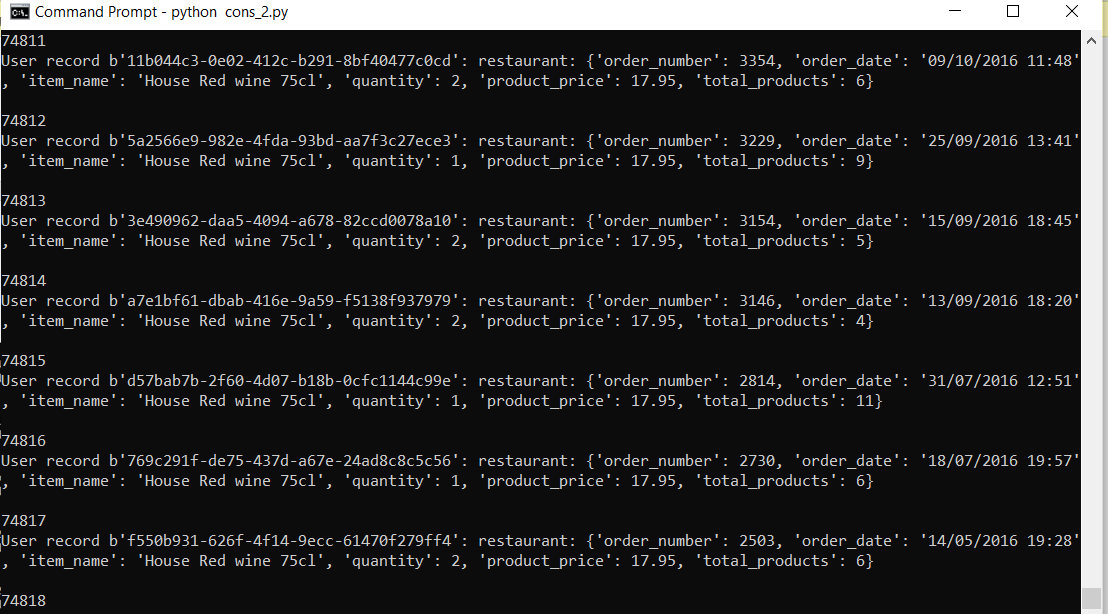


cons\_2.py



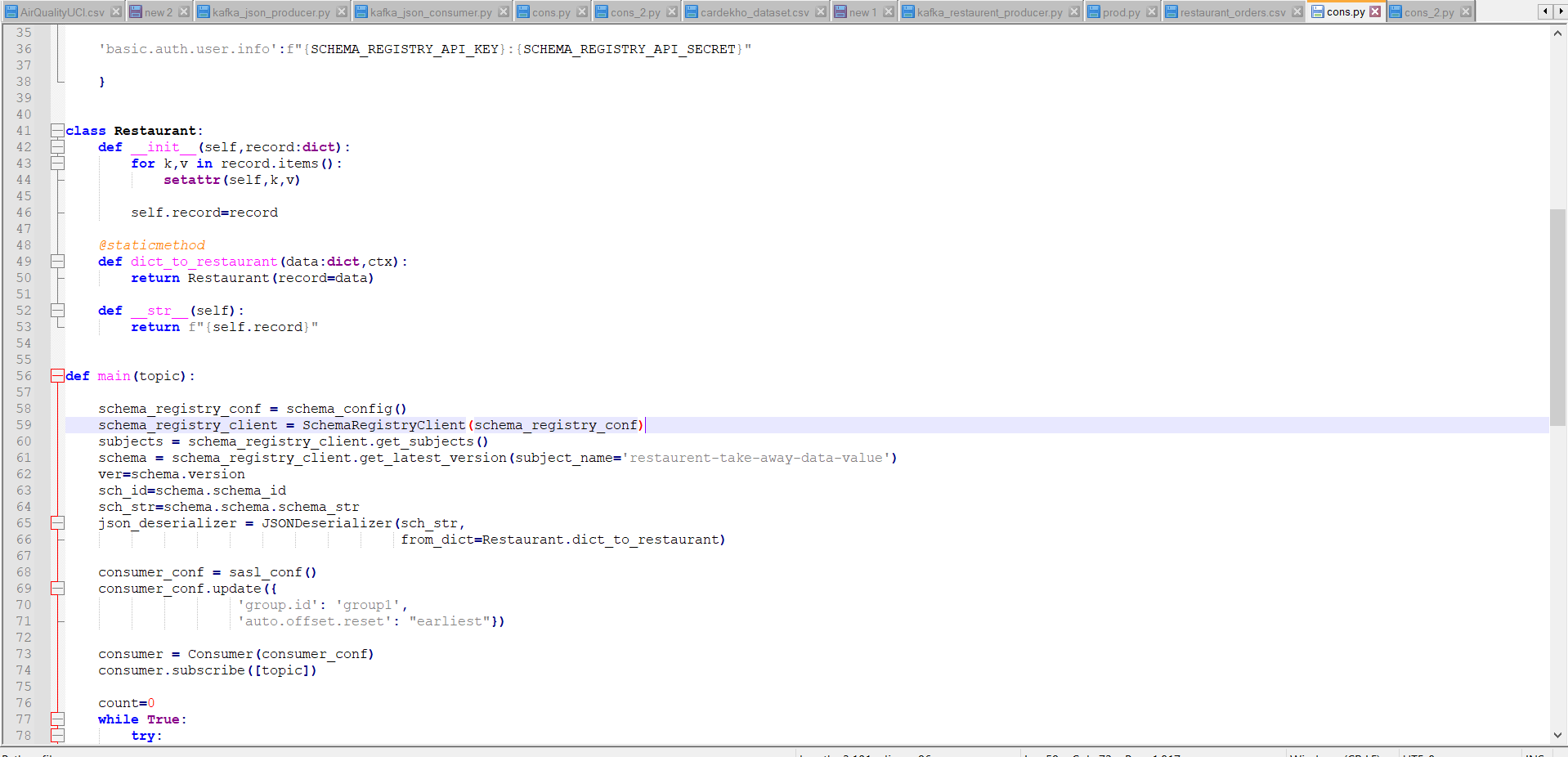
Here both consumer cons.py and cons\_2.py have different group ids 1 and 2 respectively and hence belong to two separate consumer groups and as a result both of them separately consume the whole data and as a result we get the same count i.e. 74818



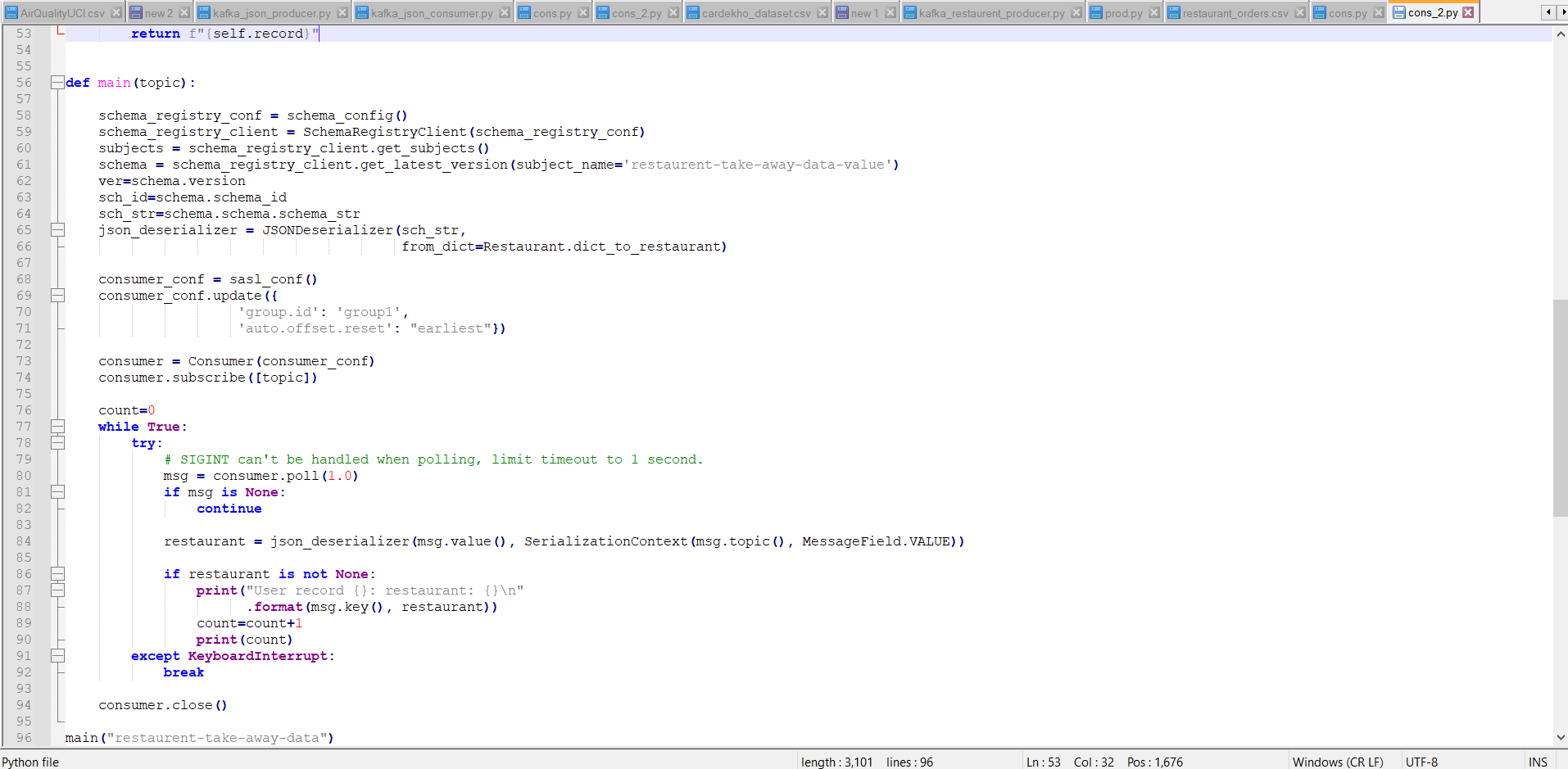


1. Use "group.id" property in consumer config for both consumers and mention same group\_ids in kafka\_consumer\_1.py & kafka\_consumer\_2.py, apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed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

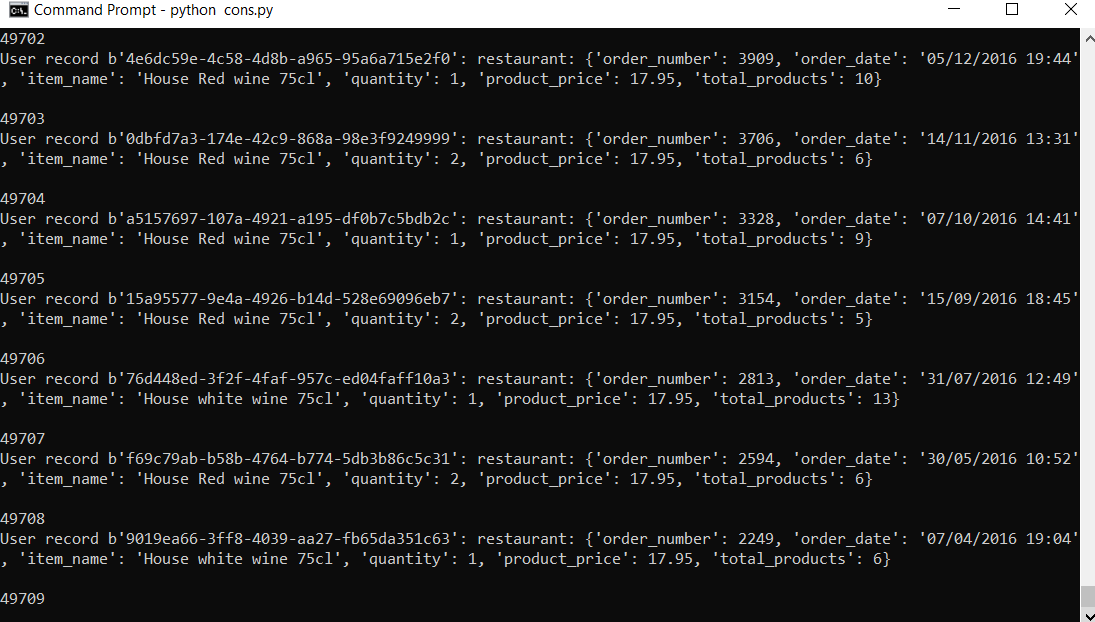
cons.py

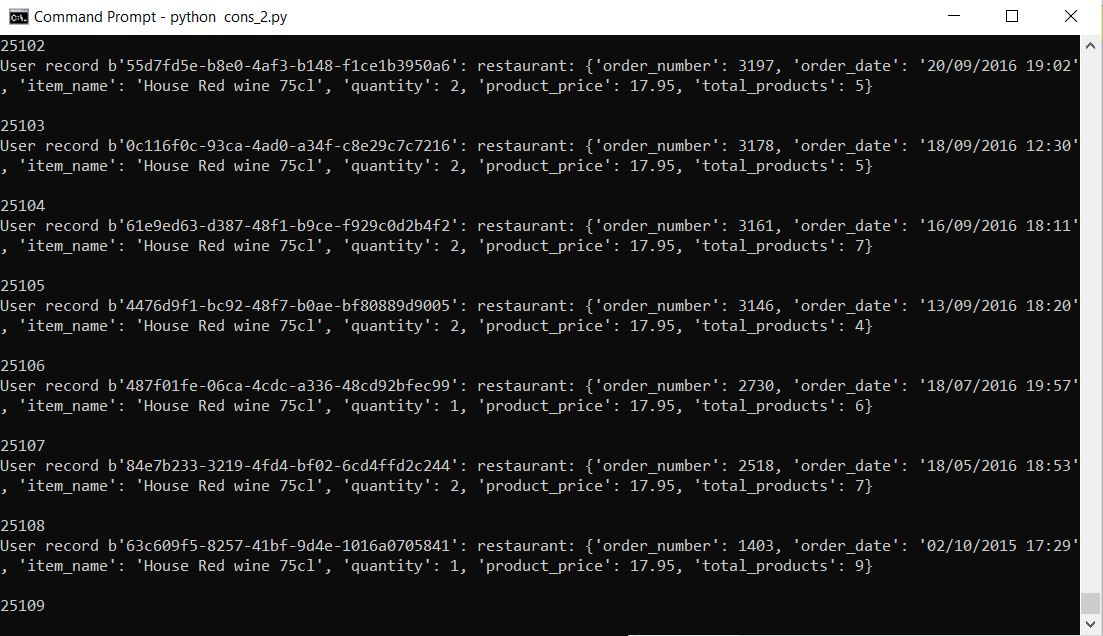


cons\_2.py



Here as you can see both the consumer belong to same consumer group and have same group id which is 1 and hence they parallelly consume records which producer produces and the sum of both he counts will be equal to total records.



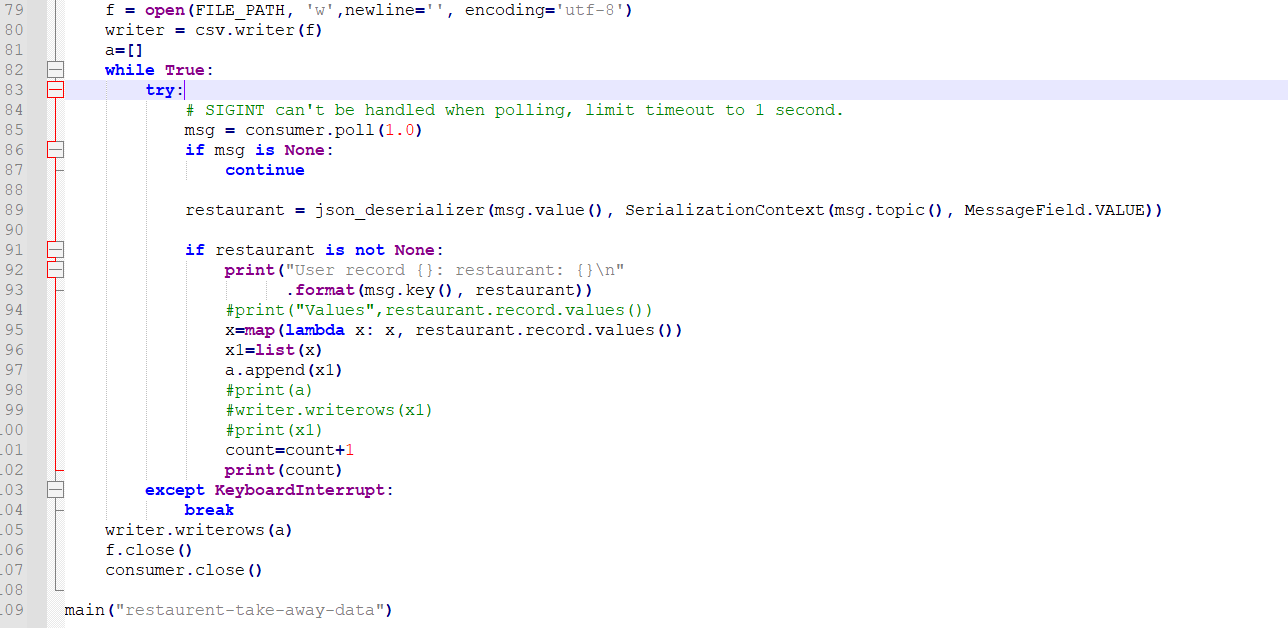


If you see the cons.py consumed 49709 while cons\_2.py consumed 25109 if we add them both it’s resulting in 74818 which is the count each consumers generated when they were having different group id’s and hence belonged to different consumer groups.

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.



Import csv module and assign path on which you want to create a file



Open the file in write mode with parameters such as newline encoding etc.

Then do a map operation and convert the result to a list and append to an empty list

Finally write the appended list to the file

