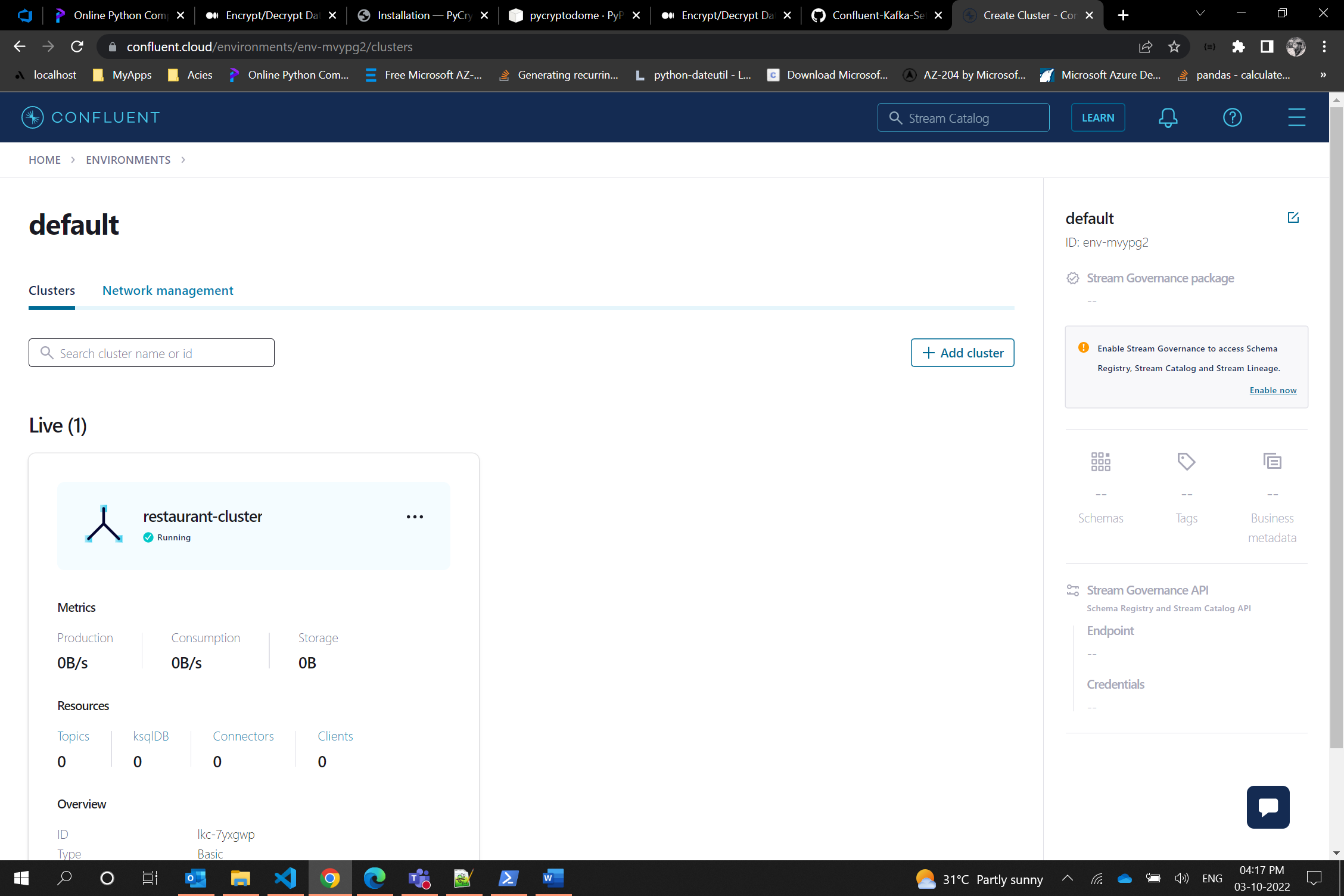
1. Setup Confluent Kafka Account



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



A screenshot of a computer

Description automatically generated

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

{

"$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": {

"Order Number": {

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

"type": "number"

},

"Order Date": {

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

"type": "string"

},

"Item Name": {

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

"type": "string"

},

"Item Name": {

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

"type": "number"

},

"Product Price": {

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

"type": "number"

},

"Total products": {

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

"type": "number"

}

},

"title": "SampleRecord",

"type": "object"

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Write a kafka producer program (python or any other language) to read data records from restaurent 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.

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

Text

Description automatically generated

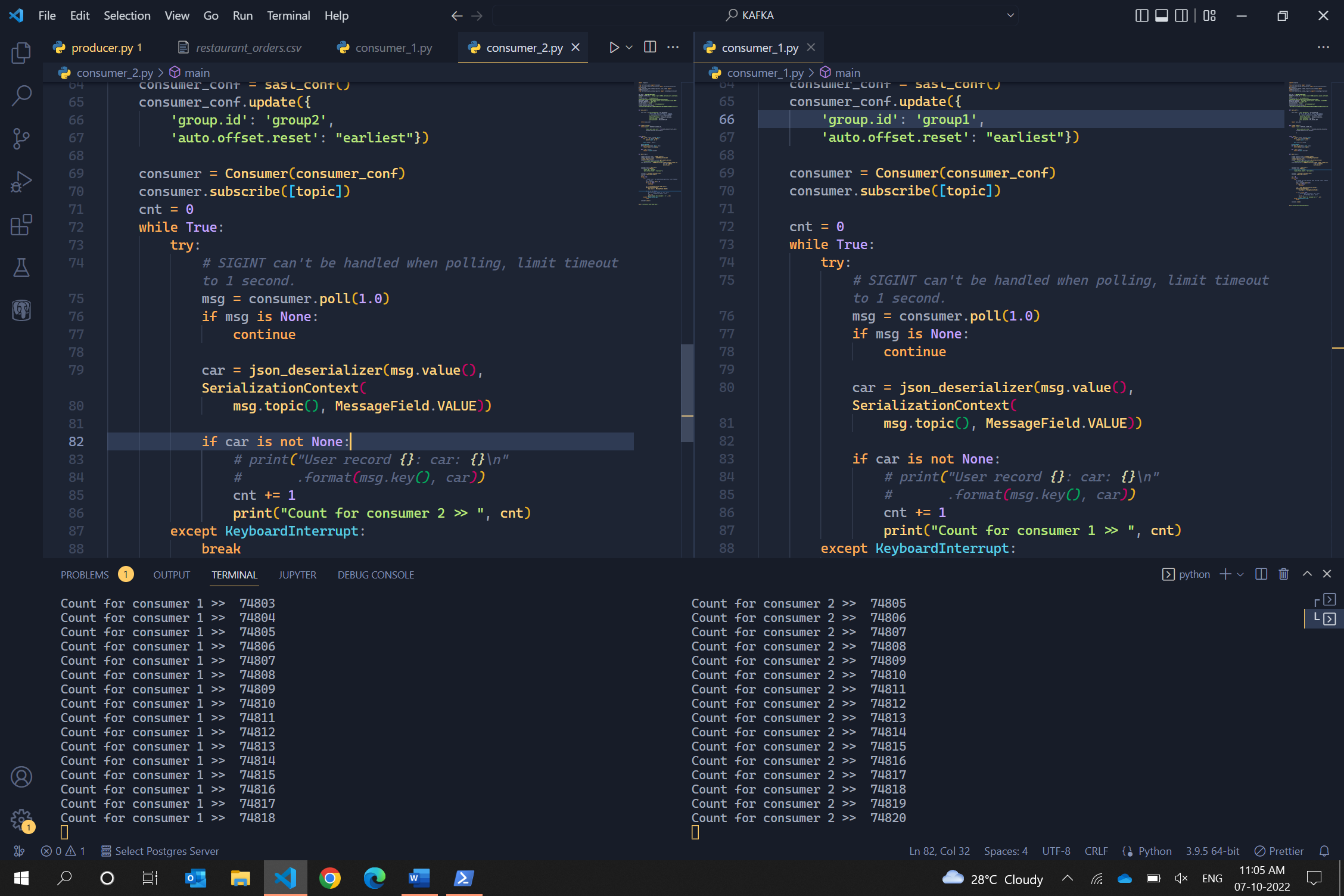
6. Write kafka consumer code and create two copies of same consumer code and save it with different names (kafka\_consumer\_1.py & kafka\_consumer\_2.py), again make sure latest schema version and schema\_str is not hardcoded in the consumer code, read it automatically from the schema registry to deserialize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

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 consumer

consumed and printed on the terminal



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,

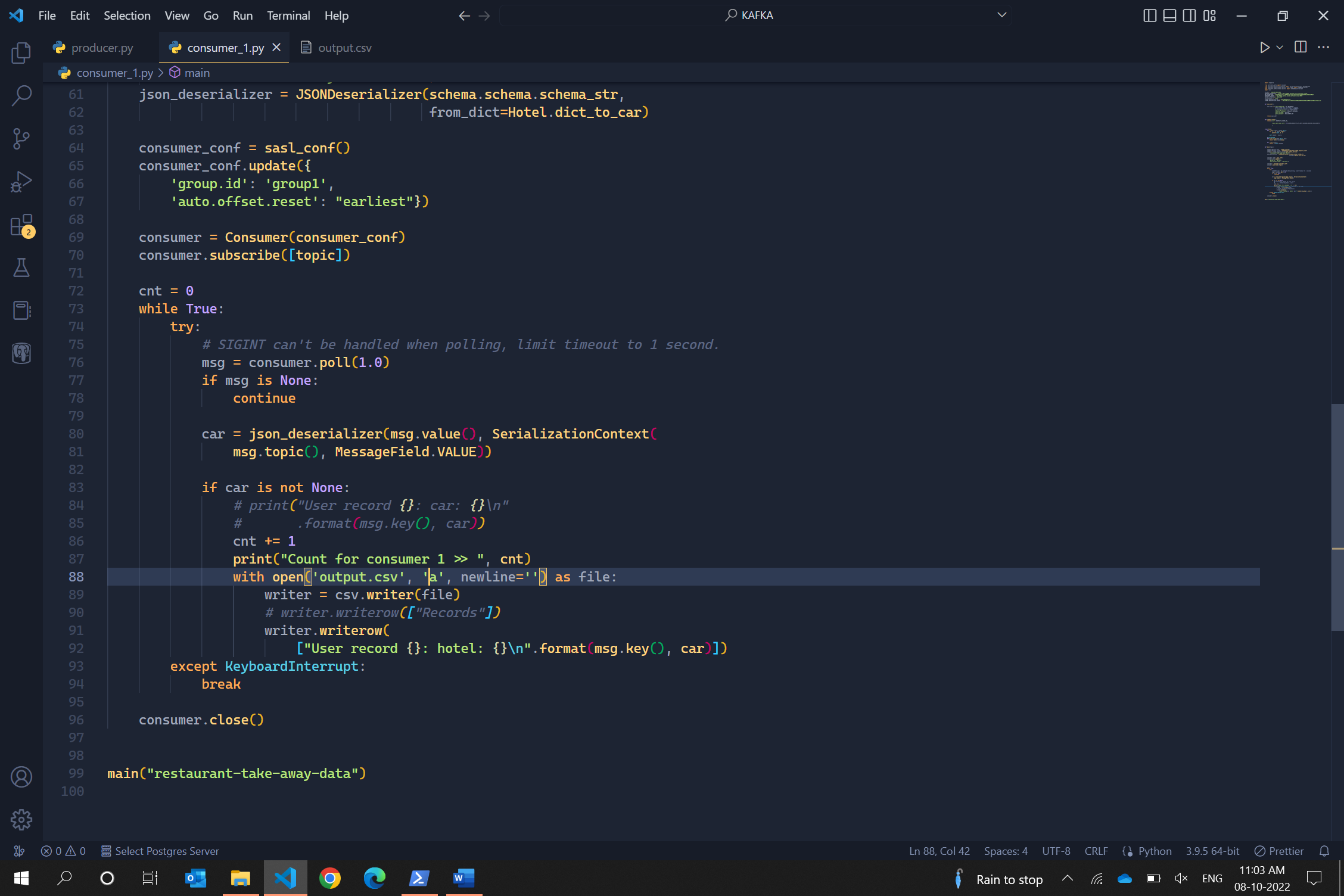
apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer

consumed and printed on the terminal

Text

Description automatically generated

1. 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





Code files:

