FIT5202 Data processing for Big Data

Assignment 2B:

Real-time stream processing on big data

Student Name: PEIYU LIU
Student ID: 31153291

1 Producing the streaming data

implement an Apache Kafka producer to simulate the real-time streaming of the data.generate the event timestamp in UTC timezone convert the timestamp to unix-timestamp format (keeping UTC timezone) “ts” column.

In [ ]:

```
# import libraries  that assignment nended.
from time import sleep
from json import dumps
import random
import datetime as dt
import itertools
import os
import pandas as pd
from kafka import KafkaProducer
```

Read the 20 files of “flight*.csv” flightsRawDf

Reference: list all files:https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory (https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory)
Literate all flight files in flight folder. Add files' names with files' path into list

In [ ]:

```
def load_data(file_path):
    file_path = file_path
    files = os.listdir(file_path)
    file_container = []
    for file in files:
        if file.startswith('flight'):  # need all “flight*.csv” files
            file_container.append('./flight-delays/' + file)
    # read data
    data_container = []
    for flag in file_container:
        df = pd.read_csv(flag,index_col=None, header=0)
        data_container.append(df)
    return  pd.concat(data_container,axis=0,ignore_index= True)
```

In [ ]:

```
df = load_data('./flight-delays/')
```

a. Take the DAY_OF_WEEK column as the key, name a variable KeyFlights which contains the set of keys (7 keys).

In [ ]:

```
KeyFlights = set(df['DAY_OF_WEEK'])# set remove duplicated values
KeyFlights
```

b. Create a function getFlightRecords, which returns a variable named flightRecords, which is a dictionary that contains all flight data with their associated keys (step 3).

a dictionary that contains all flight data with their associated keys
reference: to_dict: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_dict.html (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_dict.html)
itertools:https://github.com/dpkp/kafka-python/blob/master/benchmarks/record_batch_compose.py (https://github.com/dpkp/kafka-python/blob/master/benchmarks/record_batch_compose.py)
cicle: itertools.cycle: https://www.geeksforgeeks.org/python-itertools-cycle/ (https://www.geeksforgeeks.org/python-itertools-cycle/)

In [ ]:

```
#to_dict: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_dict.
html
# ‘records’ : list like [{column -> value}, … , {column -> value}]
#itertools.cycle: https://www.geeksforgeeks.org/python-itertools-cycle/
# itertools:https://github.com/dpkp/kafka-python/blob/master/benchmarks/record_b
atch_compose.py
def getFlightRecords(data):
    # group data by keys[1-7]
    key_flightRecords = data.groupby('DAY_OF_WEEK')
    # use itertools.circle function to format data as  key: list
    #{key: <itertools.cycle at 0x7f69bd3c4740>...}
    # cycle() will repeat the incoming sequence indefinitely
    list_mode = 'records'
    #list like [{column -> value}, … , {column -> value}]
    flightRecords = {k:itertools.cycle(v.to_dict(orient=list_mode)) for k,v in k
ey_flightRecords}
    return flightRecords
```

In [ ]:

```
# https://opendev.org/openstack/deb-python-kafka/commit/b8717b4b79462e83344f49bb
d42312cf521d84aa
#{key: <itertools.cycle at 0x7f69bd3c4740>...}
flightRecords = getFlightRecords(df)
```

flightRecords

c. Create a topic called ‘flightTopic’

In [ ]:

```
topic = 'flightTopic'
```

d. Create an instance variable called ‘flightProducer’

```
According to week10 lab task
```

In [ ]:

```
def publish_message(producer_instance, topic_name, data):
    try:
        producer_instance.send(topic_name, value = data)
    except Exception as ex:
        print('Exception in publishing message.')
        print(str(ex))

def connect_kafka_producer():
    _producer = None
    try:
        _producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
                                  # https://docs.python.org/3/library/json.html
                                  # json encode Serialize obj to str
                                  value_serializer=lambda x: dumps(x).encode('as
cii'),
                                  api_version=(0, 10))
    except Exception as ex:
        print('Exception while connecting Kafka.')
        print(str(ex))
    finally:
        return _producer
```

e

Generate A[‘keyFlight’] and B[‘keyFlight’]

```
keyFlight_A,keyFlight_B =[],[]
```

keyFlight_A.append(dict(next(flightRecords[keyFlight]),**{'ts':utc_format}))**
keyFlight_B.append(dict(next(flightRecords[keyFlight]),{'ts':utc_format}))

f

Send X and Y to the consumer a. At time1: X1 and Y 1 are generated on the producer side, but only X1 is sent.
b. At time2: X2 and Y2 are generated on the producer side, but only X2 and pending data from the previous batch (Y1) are sent to the consumer.

```
According to week10 lab task
```

reference:next() [https://www.javatpoint.com/python-next-function (https://www.javatpoint.com/python-next-function)](https://www.javatpoint.com/python-next-function)

In [ ]:

```
if __name__ == '__main__':
    topic = 'flightTopic'
    print("Publishing records...")
    flightProducer = connect_kafka_producer()
    #batch X and Y
    keyFlight_A,keyFlight_B =[],[]
    flag = True
    while flag:
        for keyFlight in [*flightRecords]:
            utc_format = int(dt.datetime.utcnow().timestamp())
            #Generate A[‘keyFlight’] and B[‘keyFlight’] timestamp
            for a in range(random.randint(70,100)):# between 70~100 (inclusive)
                # give data and timestamp to A
                #next() returns the next item from the iterator
                #https://www.javatpoint.com/python-next-function
                keyFlight_A.append(dict(next(flightRecords[keyFlight]),**{'ts':u
tc_format}))
            for b in range(random.randint(5,10)):#5~10 (inclusive)
                # give data and timestamp to B
                 #next() returns the next item from the iterator
                keyFlight_B.append(dict(next(flightRecords[keyFlight]),**{'ts':u
tc_format}))
        publish_message(flightProducer,topic,keyFlight_A)
        keyFlight_A = keyFlight_B
        keyFlight_B=[]
        sleep(5)
```