Real Time Flight Data Streaming - Kafka Producer

Flight Data Overview:

The flight-delays and cancellation data was collected and published by the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics. This data records the flights operated by large air carriers and tracks the on-time performance of domestic flights. This data summarises various flight information such as the number of on-time, delayed, cancelled, and diverted flights published in DOT's monthly in 2015.

In [7]:
```
# import required libraries
from time import sleep
from json import dumps
from kafka import KafkaProducer
import random
import csv
from datetime import timezone
import datetime
```

In [8]:
```
from pathlib import Path

flights_dir = Path('D:/GitHub/DSC680/Project2-Real_Time_Big_Data_Streaming_with_Kafka
```

Step 1

Read in clean csv data for streaming

Write a python program that loads all the data from "flight*.csv". Save the file as Task1_flight_producer.ipynb.

In [9]:
```
# function to read in the csv flight files
def readCSVFile():

    # empty list for all dfs
    flight_dict_list = []

    # load all of the csv files as dfs, 20 flight csv files
    for i in range(1,21):

        # open each flight csv file
        with open(flights_dir.joinpath(f'flight{i}.csv'), 'rt') as f:

            # read in the csv as a dict
            reader = csv.DictReader(f)

            # loop through every single dictionary
            for row in reader:

                # append each dict to the list
                flight_dict_list.append(row)

    # return a list of dictionaries
    return flight_dict_list
```

Step 2

Prepare data for day_of_week usage

The keyFlights are generated from the column ‘DAY_OF_WEEK’ in the dataset which has 7 unique keys. These values 1, 2, 3, 4, 5, 6, and, 7 represents ‘sunday’,‘monday’, ‘tuesday’, ‘wednesday’,’thursday’,’friday’,’saturday’

In [10]:
```
# function that saves each observation in a dict of lists according to her
def prepare_data(all_data):

    # dict of lists containing flight data for each of the 7 unique keys
    dow_dict = {1:[], 2:[], 3:[], 4:[], 5:[], 6:[], 7:[]}

    # loop through all days of the week
    for i in range(0,len(all_data)):

        if all_data[i]['DAY_OF_WEEK'] == '1':

            dow_dict[1].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '2':

            dow_dict[2].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '3':

            dow_dict[3].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '4':

            dow_dict[4].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '5':

            dow_dict[5].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '6':

            dow_dict[6].append(all_data[i])

        elif all_data[i]['DAY_OF_WEEK'] == '7':

            dow_dict[7].append(all_data[i])

    return dow_dict

# function that saves the lenght of a list and saves it as a key value pair
def get_len_data(data):

    len_data = {}

    for key,value in data.items():

        len_data[key] = len(value)

    return len_data
```

Step 3

Create publisher and producer function

Functions are taken from the Week 9 Lab material provided.

In [11]:
```
# function that publishes the message
def publish_message(producer_instance, topic_name, data):
    try:
        producer_instance.send(topic_name, data)
```

```
    except Exception as ex:
        print('Exception in publishing message.')
        print(str(ex))


# function that connects the kafka producer
def connect_kafka_producer():
    _producer = None
    try:
        _producer = KafkaProducer(bootstrap_servers=['192.168.86.48:9092'],
                                  value_serializer=lambda x: dumps(x).encode('ascii')
                                  api_version=(0, 10))
    except Exception as ex:
        print('Exception while connecting Kafka.')
        print(str(ex))
    finally:
        return _producer
```

Step 4

Send desired data batches

Using the above created functions we create the desired flight data batches and send them so the consumers can receive the data.

In [ ]:
```
if __name__ == '__main__':


    ## SET TOPIC AND DATA TO BE SENT

    topic = 'flightTopic'

    all_data = readCSVFile()

    print(all_data)
    ## SET THE PRODUCERS

    producer_X = connect_kafka_producer()
    producer_Y = connect_kafka_producer()


    ## GET DATA AND META DATA FOR EACH KEY

    # dict where each key is a DOW and each value
    # is a list of flights
    key_all_data = prepare_data(all_data)
    print(key_all_data)
    # dict where each key is a DOW and each value
    # is the length of the list of rows
    len_key_all_data = get_len_data(key_all_data)
    print(len_key_all_data)
    # create a dict to count the rows sent so far
    data_counter = {1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0}

    # list to save the y data created
    storage_y_data = []

    iteration_counter = 0


    # start the data publishing process
    print('Publishing records..')

    # set a continous loop to produce and publish data
    while True:
```

```
# loop through the keys from 1 to 7 for each day of the week
for i in range(1,8):


    ## 1. GET DATA AND META DATA FOR KEY

    # get the data for DOW
    key_data = key_all_data[i]

    # get the len of the the data for DOW
    len_key_data = len_key_all_data[i]


    ## 2. GET TIMESTAMP FOR EACH RECORD

    # creates the current date and time in utc timezone
    dt_utc_now = datetime.datetime.utcnow()

    # changes the timestamp into the desired format
    utc_time_round = dt_utc_now.replace(microsecond=0)

    # convert the timestamp to unix-timestamp format
    utc_timestamp = utc_time_round.timestamp()

    # save the unix-timestamp as ts
    ts = {'ts': int(utc_timestamp)}


    ## 3. BATCH FOR X

    # generate a random intiger
    A = random.randint(70,100)

    # ensure there is enough data to be used
    if (data_counter[i] + A) >= len_key_data:

        # if not start from the start again
        data_counter[i] = 0

        # select data
        x_data = key_data[data_counter[i]: data_counter[i] + A]

    else:

        # select data
        x_data = key_data[data_counter[i]: data_counter[i] + A]

    # increment counter
    data_counter[i] += A

    # list that saves the final dicts containing timestamps
    final_x_data = []

    # loop through the list of dicts
    for x in range(0,len(x_data)):

        # append the timestamp into the object to be sent
        final_x_data.append(dict(x_data[x], **ts))


    ## 4. BATCH FOR Y

    # generate a random intiger
    B = random.randint(5,10)

    # ensure there is enough data to be used
    if (data_counter[i] + B) >= len_key_data:

        # if not start from the start again
        data_counter[i] = 0
```

```
            # select data
            y_data = key_data[data_counter[i]: data_counter[i] + B]

        else:

            # select data
            y_data = key_data[data_counter[i]: data_counter[i] + B]

        # increment counter
        data_counter[i] += B

        # list that saves the final dicts containing timestamps
        final_y_data = []

        # loop through the list of dicts
        for x in range(0,len(y_data)):

            # append the timestamp into the object to be sent
            final_y_data.append(dict(y_data[x], **ts))

        # append the final_y_data to the list storage_y_data for later use
        storage_y_data.append(final_y_data)


        ##  5. PUBLISH THE MESSAGE

        if iteration_counter == 0:

            # for producer x publish the message with the new data
            publish_message(producer_X, topic, final_x_data)

            # select the data to send for y
            y_data_send = None

        else:

            # select the data to send for y
            y_data_send = storage_y_data[iteration_counter - 1]

            # for producer x publish the message with the new data
            publish_message(producer_X, topic, final_x_data)

            # for producer y publish the message with the new data
            publish_message(producer_Y, topic, y_data_send)


        ## 6. INCREMENT THE COUNT

        # increment iteration counter
        iteration_counter += 1


        ## 7. PRINT THE DATA SENT

        # print the data being sent
        #"""
        print('X - DATA')
        print(final_x_data)
        print('')
        print('Y - DATA')
        print(y_data_send)
        print('-------------------------')

        #"""

        # send producer to sleep for 5 seconds
        sleep(5)
```

In [ ]:

In [ ]: