FIT5202: Data processing for big data

Assignment 2B: Real-time stream processing on big data

Task 1: Kafka Producer

1. Import packages:

In [1]:
```
# import statements
from time import sleep
from json import dumps
from kafka import KafkaProducer
import random
from datetime import datetime,timezone
from pytz import timezone
import csv
import pandas as pd
import glob
```

2. Data Preprocessing:

Check data for null values and prepare data accordingly

In [2]:
```
path = './flight-delays' # path to folder containing data files
all_files = glob.glob(path + "/flight*.csv")
filesdf = []
for filename in all_files: #for each file in the dataset
        df = pd.read_csv(filename, index_col=None, header=0) #read the file
        filesdf.append(df) #append dataframe to the list

flightsDf = pd.concat(filesdf, axis=0, ignore_index=True) # combine the datafra

print("Number of records:", len(flightsDf))

print("Number of null records in each column:")

flightsDf.isnull().sum()
```

```
Number of records: 582184
Number of null records in each column:
```

Out[2]:
```
YEAR                       0
MONTH                      0
DAY                        0
DAY_OF_WEEK                0
AIRLINE                    0
FLIGHT_NUMBER              0
TAIL_NUMBER             1462
ORIGIN_AIRPORT             0
DESTINATION_AIRPORT        0
SCHEDULED_DEPARTURE        0
DEPARTURE_TIME          8633
DEPARTURE_DELAY         8633
TAXI_OUT                8891
WHEELS_OFF              8891
SCHEDULED_TIME             1
ELAPSED_TIME           10455
AIR_TIME               10455
DISTANCE                   0
WHEELS_ON               9257
TAXI_IN                 9257
SCHEDULED_ARRIVAL          0
ARRIVAL_TIME            9257
ARRIVAL_DELAY          10455
DIVERTED                   0
CANCELLED                  0
CANCELLATION_REASON    573213
AIR_SYSTEM_DELAY       475831
SECURITY_DELAY         475831
AIRLINE_DELAY          475831
LATE_AIRCRAFT_DELAY    475831
WEATHER_DELAY          475831
dtype: int64
```

Columns 'CANCELLATION_REASON', 'AIR_SYSTEM_DELAY', 'SECURITY_DELAY' ,'AIRLINE_DELAY' ,'LATE_AIRCRAFT_DELAY', and 'WEATHER_DELAY' have large number of null values so we can avoid publishing these columns.

For the rest of the columns "TAIL_NUMBER", "DEPARTURE_TIME", "DEPARTURE_DELAY", "TAXI_OUT", "WHEELS_OFF", "SCHEDULED_TIME", "ELAPSED_TIME", "AIR_TIME", "DISTANCE", "WHEELS_ON", "TAXI_IN", "SCHEDULED_ARRIVAL", "ARRIVAL_TIME", "ARRIVAL_DELAY" and "DIVERTED" which have null values in some rows, the rows which have either of these columns as null values will not be published.

3. Initialise kafka producer and publish message functions:

In [3]:
```
#function to publish message for the conusmer
def publish_message(producer_instance, topic_name, data):
    try:
        producer_instance.send(topic_name, data)
        print('Message published successfully. Data: ' + str(data))
    except Exception as ex:
        print('Exception in publishing message.')
        print(str(ex))


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

4. Define the 'getFlightRecords' function:

In [4]:
```
def getFlightRecords(fights_dataframe, flight_key, number_of_records, time_stam
    #create empty list
    flightRecords = []

    #sample out the records for the given key and number of records
    df = fights_dataframe.loc[fights_dataframe['DAY_OF_WEEK'] == flight_key].sa

    #add the timestamp column
    df['ts'] = time_stamp

    #add to the list
    flightRecords.append({key: df.to_dict('records')})

    #return the list
    return flightRecords
```

5. Define the main section of the task:

In [19]:
```
if __name__ == '__main__':

    topic = 'flightTopic'

    print('Publishing records..')
    producer = connect_kafka_producer()

    path = './flight-delays' # path to folder containing data files
    all_files = glob.glob(path + "/flight*.csv") # list of all the 'flight' fil

    filesdf = [] # list object to store pandas dataframe of each file

    for filename in all_files: #for each file in the dataset
        df = pd.read_csv(filename, index_col=None, header=0) #read the file
        filesdf.append(df) #append dataframe to the list

    flightsDf = pd.concat(filesdf, axis=0, ignore_index=True) # combine the dat

    #Remove columns with maximum null values
    flightsDf = flightsDf.drop(['CANCELLATION_REASON', 'AIR_SYSTEM_DELAY', 'SEC



    #Remove rows where any column has null values
    flightsDf.dropna(subset = ["TAIL_NUMBER", "DEPARTURE_TIME", "DEPARTURE_DELA

    #list of keys for the dataframe
    keyFlights = list(flightsDf['DAY_OF_WEEK'].unique())

    while True:

        X_data = [] #store data to be send in current batch
        Y_data = [] #store data to be send in next batch but generated in curre
        Y_previous_batch = [] #store pending data for previous batch to be seno


        #for each key generate data
        for key in keyFlights:

            #random number of data points
            #for dataset X
            A = random.randint(70, 100)

            #for dataset Y
            B = random.randint(5, 10)

            #generate timestamp
            ts = int(datetime.now(timezone('UTC')).timestamp())

            #get data for batch X
            flightRecords_A = getFlightRecords(flightsDf, key, A, ts)[0].get(ke

            #get fata for batch Y
            flightRecords_B = getFlightRecords(flightsDf, key, B, ts)[0].get(ke

            for record in flightRecords_A:
                X_data.append(record)

            for record in flightRecords_B:
                Y_data.append(record)

            #to get different timestamp for each key add delay of 1 sec
            sleep(1)

        #combine dataset to be sent (X of current batch, Y of previous batch)
        for record in Y_previous_batch:
            X_data.append(record)
```

In [ ]: