Real Time Stock Ticker Streaming - Kafka Producer

Stock Data Overview:

Stock data comes from yFinance. Using the API, we can get historical data as well as live data(when market is open), fo one or more stocks.

In [23]:
```
# import required libraries
import pandas as pd
import pandas

from time import sleep
from json import dumps
from kafka import KafkaProducer
import random
import csv
from datetime import timezone
import datetime
```

In [24]:
```
import yfinance as yf
from pandas_datareader import data as pdr
from datetime import datetime
yf.pdr_override()
```

Step 1

Define functions

In [25]:
```
# Get historical data for the given list of stocks
def GetHistoricalData(stock_list,numberOfYears = 0,numberOfMonths = 4):
    #print(numberOfMonths)
    end = datetime.now()
    start = datetime(end.year -  numberOfYears, end.month - numberOfMonths, end.day ,

    historicalData = []
    for stock in stock_list:
        #print(start,end)
        df = yf.download(stock, start, end)
        df = df.reset_index()
        df['Label'] = stock
        df['Date'] = pd.to_datetime(df['Date'])
        historicalData.append(df)
    return(historicalData)
```

In [26]:
```
# Get Live data for the given stock
def GetLiveData(symbol):
    ticker = yf.Ticker(symbol).info
    market_price = ticker['regularMarketPrice']
    market_open = ticker['regularMarketOpen']
    market_High = ticker['regularMarketDayHigh']
    market_Low = ticker['regularMarketDayLow']
    market_volume = ticker['regularMarketVolume']
    market_symbol = ticker['symbol']
    #print('Ticker: AMAT')
    #print('Market Price:', market_price)
    #print('Previous Close Price:', previous_close_price)
    liveData = {'Date':[pd.to_datetime(datetime.now())],'Open': [market_open] , 'High
    return(pd.DataFrame(liveData))
```

In [27]:
```
#Test historical Data of one year
stock_list = ['AMAT','IBM','INTC']
historicalData = GetHistoricalData(stock_list,1,0)
print(historicalData[0].head())
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
        Date        Open        High         Low       Close   Adj Close  \
0 2022-05-09  107.940002  110.790001  105.379997  105.750000  104.722130
1 2022-05-10  109.220001  109.570000  105.180000  107.180000  106.138229
2 2022-05-11  106.660004  109.129997  103.610001  103.919998  102.909904
3 2022-05-12  103.029999  106.879997  102.989998  106.760002  105.722313
4 2022-05-13  108.190002  112.629997  108.029999  111.860001  110.772736

     Volume Label
0  10088400  AMAT
1   8881000  AMAT
2   8311700  AMAT
3   9287100  AMAT
4   8321100  AMAT
```

In [ ]:
```
#test GetLiveData
liveData = GetLiveData('AMAT')
print(liveData.head())
```

Step 2

Create publisher and producer function

In [29]:
```
# function that publishes the message
def publish_message(producer_instance, topic_name, data):
    #print(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 3

Send desired data batches

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


        ## SET TOPIC AND DATA TO BE SENT
```

```
        topic = 'stock_ticker'

        #all_data = df.to_dict(orient='records')

        ## SET THE PRODUCERS

        producer = connect_kafka_producer()

        getLiveData = False

        ## GET DATA AND META DATA FOR EACH KEY


        iteration_counter = 0

        #stock_list = ['AMAT','LRCX','WOLF','KLAC','AAPL', 'GOOG', 'MSFT', 'AMZN']
        stock_list = ['AMAT','GOOG', 'MSFT','AMZN']
        #stock_list = ['AMAT']
        #stock_list = ['AMAT','LRCX','WOLF','KLAC']
        # start the data publishing process
        #print('Publishing records for ', stock_list, '..')

        # set a continous loop to produce and publish data

        #print(len(df_Historical))
        month = 3
        df_Historical = GetHistoricalData(stock_list,2,0)
        while True:
            """
            if month == 0:
                month = 3
            df_Historical = GetHistoricalData(stock_list,1,month)

            month = month - 1
            """
            for stock in range(len(df_Historical)):
                if getLiveData:
                    symbol = df_Historical[stock]['Label'][0]
                    liveData = GetLiveData(symbol)
                    liveData['Date'] = pd.to_datetime(liveData['Date'])
                    df_Historical[stock] = pandas.concat([df_Historical[stock],liveDa
                df_Historical[stock]['Date'] = df_Historical[stock]['Date'].astype(st
                sleep(20)

            all_data = []
            for stock in range(len(df_Historical)):
                all_data.append(df_Historical[stock].to_dict(orient='records'))


            publish_message(producer, topic, all_data)



            if iteration_counter > 1000:
                    break

            iteration_counter += 1

            # send producer to sleep
            sleep(30)
        print("Exited with ", iteration_counter, " iterations")
    except Exception as ex:
        print("exception after ", iteration_counter, " iterations")
        print(str(ex))
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

In [ ]:

In [ ]:

In [ ]: