Quiz 7: Data Streams [125 points]

In this quiz, we will demonstrate how to handle live streaming data, where information is received in real-time. Here, analytical operations need to be performed dynamically as the data arrives, without having a central repository of all the data to refer to. We will be using the twelveData API for stock price data.

An unusual feature of stock price APIs is that they are typically metered. For example, for the "free" tier, twelveData gives 800 API credits per day. And some "heavy-weight" requests cost 100 credits!

Setup:

1. Create your account using your Tufts email ID on twelveData
2. Login and Save your API Key (You will use your own unique API key in the following exercise). Store this key in a file keys.py your Google Drive with the content
   `twelveDataKey = 'use_your_own_key_blablawhatever'`
3. We'll use a GCP Dataproc Cluster soon, but first we need to exercise the API in a Jupyter notebook. Here is the code:

```
try:
    import twelvedata
except ModuleNotFoundError:
    !pip install twelvedata[pandas,matplotlib,plotly,websocket-client]
    import twelvedata
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
import sys
need_in_path = '/a/location/in/your/google/drive/that/contains/your/keys.py'
if need_in_path not in sys.path:
    sys.path.append(need_in_path)
```

```
from keys import twelveDataKey as api_key
```

   Verify that the api_key variable contains the key before proceeding. Storing your key in a secure location rather than embedding it in your code is required. An automatic 10% penalty will result if your submitted code includes your key[1]!
4. [20 points for setting up data collection] We'll be working with just 3 stocks: AAPL, MSFT and IBM.
5. [25 points for collecting data] Pull the stock data of each stock at 15-minute intervals once for each day, at the end of the day and gather up at least 4 years of stock price data.
6. IMPORTANT NOTE: The data should be pulled after the market closes at 4 pm between Monday – Friday since that is the only time you will get real time stock data from this API, when the markets are operational. Plan your work accordingly. You will most probably not be able to pull data outside these timings accurately and on weekends.

[1] If we need to run the code to test it, we'll use our own key.

Algorithmic Stock Trading

Automated trading relies on good quality data feeds and their timely delivery is what accounts for the cost. Watch @How algorithms shape our world | Kevin Slavin and answer the following questions:

7. [12 points] If you had to buy 1,000,000 shares of a stock without letting the market know, list some strategies you might use. Be as specific as possible: "I would break up the order into manageable chunks" wouldn't get you much credit. Search, research, and then answer.
8. [12 points] If you had to figure out if someone was "dumping" a large quantity of stocks, how would you do it? What parameters, over what period of time?

The best approach to this problem is to pair up with another classmate, one of you in charge of "dumping," the other in charge of "detecting."

Technical Analysis of Stock Trading

Calculation of moving stock price averages are part of many trading strategies (reference).

We will be using the two moving averages strategy, with the shorter-term MA being 10-day and the longer average being 40-day. When the shorter-term MA crosses above the longer-term MA, it's a buy signal, as it indicates that the trend is shifting up. This is known as a **golden cross.** Meanwhile, when the shorter-term MA crosses below the longer-term MA, it's a sell signal, as it indicates that the trend is shifting down. This is known as a **dead/death cross.** Both types of crosses are shown in the figure below:

To simulate a data stream, you are given the feeds above, which produce stock prices for Apple (Symbol: `aapl`), Microsoft (Symbol: `msft`) IBM (Symbol: `ibm`).

9. The program at https://github.com/singhj/big-data-repo/blob/main/spark-examples/stock-price-feeder.py was designed to feed stock prices of Microsoft and Google. However, it relies on an almost-obsolete API (finance-datareader).
10. [16 points] Create a similar program `new-stock-price-feeder.py` that uses a more modern API (e.g., twelveData) instead.

The line `new-stock-price-feeder.py | nc -lk 9999` can then be run used for feeding price data into spark.

11. [8 pts] Set up the stream to feed data into pyspark streams aaplPrice and msftPrice.
12. [8 pts] From aaplPrice produce two other streams aapl10Day, aapl40Day
13. [8 pts] From msftPrice produce two more streams msft10Day and msft40Day .
14. [16 pts]. Compare the two moving averages (10-day MA and the 40-day MA) to indicate buy and sell signals . Your output should be of the form `[( <date> buy <symbol>), ( <date> sell <symbol>), etc]`.

For each case, submit your spark code and the buy/sell recommendations for each stock. [Actual trades have to specify the number of shares. Assume that each time you need to buy/sell, you are trading 1000 shares of Apple[2].]

[2] Correspondingly, the number of Microsoft shares traded should be 1000÷***k***.