**CS 582: Stock Market Project**

**Deadline: December 2nd (Friday) 11:59pm**

Goal of this project is to learn threading and locking using databases in a real-time environment. For this project, you will download stock market data in real-time

**Step 1:**

Go to <https://polygon.io/docs/stocks/getting-started> and sign up using your NMSU email id.

You MUST use your nmsu email id (@nmsu.edu) to sign up. After you sign up, you need to upgrade your Basic plan to Starter plan. Enter the coupon code NMSUCS582 (just like we did in class).

Please do not share the coupon code. Polygon has agreed to give 1-month free access. Note, YOU DO NOT NEED TO PAY FOR THIS. The coupon code will change the cost of the Starter plan to 0/

Once you sign up, you will find your API Key at: <https://polygon.io/dashboard/api-keys>

This is the link to Python client library for Polygon REST API: <https://github.com/polygon-io/client-python>

Simple example to get started: <https://github.com/polygon-io/client-python/blob/master/examples/rest/simple-get.py>

Note that you need to pass your API Key when you are creating the RestClient...i.e. replace line 3 in the above code

*client = RESTClient()*

with

*client = RESTClient(YOUR\_API\_KEY)*

Replace YOUR\_API\_KEY with your API key that is listed at <https://polygon.io/dashboard/api-keys>

**Step 2:**

Download Stocks.db from Files section on Canvas.

This file is a SQLite (<https://www.sqlite.org/index.html>) database file.

I use DB Browser for SQLite to visualize the database (<https://sqlitebrowser.org/>)

Read the list of stock names from the only table in this database (example: <https://www.sqlitetutorial.net/sqlite-python/sqlite-python-select/>) and store it in a list.

**Step 3:**

Now you need to download 1-minute data in real-time for each of the stock mentioned in Stocks.db. You should run your code during the regular hours of the stock market (9:30am – 4pm Eastern Time). The free Polygon plan that is given to us has a delay of 15 minutes. So, you may need to start your data collection at 9:45am ET.

There are two ways of downloading data: **You only need to implement the second way.**

1. Using a snapshot without mulitthreading (<https://polygon.io/docs/stocks/get_v2_snapshot_locale_us_markets_stocks_tickers>)  
     
   If you run the following command,   
   resp = RESTClient(*YOUR\_API\_KEY*).get\_snapshot\_all("stocks")  
     
   you will get information for all stocks in the past 1 minute. You need to store the “closing” price for each stock. Remember, this information will include stocks that are not listed in Stocks.db. So, you will need to filter accordingly.
2. Using a snapshot with multithreading (<https://polygon.io/docs/stocks/get_v2_snapshot_locale_us_markets_stocks_tickers>)  
     
   If you run the following command,   
   resp = RESTClient(*YOUR\_API\_KEY*).get\_snapshot\_all("stocks" , stockList)  
     
   you will get information for all stocks that are in the list stockList. In order to do multithreading, you should divide the original list of stocks into X sub-lists. You need to create X threads. Write a function that takes a list of stocks as an input and call the above command. This function wil be your input in threading.Thread just like in the example shown below:

Good multithreading example in Python: <https://www.geeksforgeeks.org/multithreading-python-set-1/>

You will need to lock the database. This is a sample code to lock the database. Note, this is not the only way to do it. So feel free to use other ways as well.

lockAcquire = main.lock.acquire(True)  
try:  
 curHighLow.execute("CREATE TABLE IF NOT EXISTS ‘2022-11-02’ (Time FLOAT

PRIMARY KEY, Close FLOAT, SMA20 FLOAT)")  
finally:  
 main.lock.release()  
if not lockAcquire:

print("Houston, we have a problem")

**Step 4:**

Now you need to store the data that you downloaded in a database. The data that you store should be persistent. The above step (step 3) will return lot of information about the stock. You only need to store the time and the “close” price. You do not need to store any other information.

To make it simpler, create an on-disk database. This will be slower than an in-memory database, but it will be persistent.

**Optional:** Create a database in-memory and store the data in it, and regularly copy this database to the disk when the system is idle.

This is a sample code that I have written to do the above:

databaseName = “sma20.db”  
conHLDisk = sqlite3.connect(databaseName, check\_same\_thread=False)  
curHLDisk = conHLDisk.cursor()  
  
with open('dump.sql', 'w') as f:  
 for line in conHighLow.iterdump():  
 f.write('%s\n' % line)  
  
f = open('dump.sql', 'r')  
sqlScript = f.read()  
f.close()  
curHLDisk.executescript(sqlScript)  
  
conHLDisk.commit()  
conHLDisk.close()

**Step 5:**

Find the Simple Moving Average (SMA) of the last 20 1-minute data points (you should the “close” price of each bar) and store it in the database. If there are less than 20 data points at any moment in time, the SMA 20 will be the average of the available data points. If there are >= 20 data points, then the SMA 20 will be the average of last 20 1-minute data points.

In order to find the last 20 data points, find the ROWID of the last tuple (<https://stackoverflow.com/questions/15570096/sqlite-get-rowid>). And then subtract 20 from it to get the timestamp of the row 20 data points back, and then find the average of the prices between these 2 timestamps.

**Project Deliverable:**

Upload your code in a zip file on Canvas. The zip file should also contain a small working video demo showing that it works in real-time. Your code should show how much time is taken to insert all stocks into the database for each 1-minute timestamp. It should also contain the database that has the 20 SMA values for all stocks in stocks.db for any given day that you choose (this day should be mentioned somewhere – one solution would be to name the table with the date in format YYYY-MM-DD).