Scraping Stock Performance Data for Companies in Hampton Roads

Table of Contents

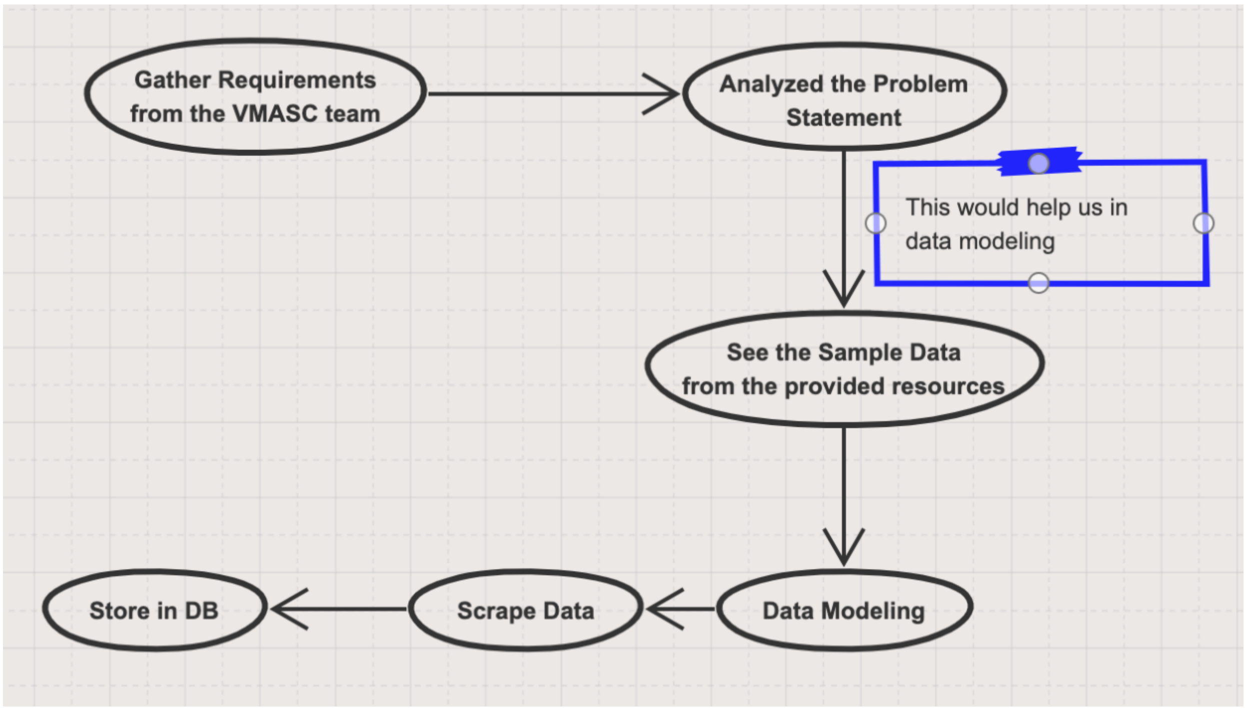
1. Aim
2. Architectural Process and Design
3. Data Modeling
4. Programmatic Design and Explanation
5. Project breakdown for collaborative work
6. Sample Queries

1. AIM

The purpose of the project is to scrape stock performance data from various web resources that would help in tracking companies in Hampton Roads area. This analysis would be used to assess company's overall performance.

2. Architectural Process and Design

This is going to be a vague customized process. Implementation process steps are shown below.


Gather Requirements from the VMASC team
Analyzed the Problem Statement
This would help us in data modeling
See the Sample Data from the provided resources
Data Modeling
Scrape Data
Store in DB

2.1 Gather Requirements:

- Collect companies listed on "https://hamptonroadsalliance.com" with respect to each sector
- Capture all of these companies with their stock symbol in the database.
- Using list of companies captured previously, search web resources "https://seekingalpha.com/page/feeds", "https://finance.google.com" and "https://finance.yahoo.com" to find relevant information on recent company performance.

2.2 Analyze the Problem Statement:

- Took some time to keenly analyze the problem statement. The problem was very specific to stock market, so I have gathered some domain knowledge.

2.3 Sample Data:

- After gaining some knowledge on stock market domain and looking at the provided web resources. I have got some idea on what data to scrape which would help me in assessing company stock performance. Some data that I have taken into consideration is:
    - Historical Stock prices (open, close, etc.)
    - Latest news on stock
    - Stock Market (i.e., NYSE, NasDaq, etc.)
- There are plenty of other resources which can be considered as well. Few of them are:
    - Quarterly Earnings and Financial Data
    - Research Reports
    - Press Release
    - P/E Ratio
    - Market Cap

2.4 Data Modeling:

- Based on the analysis on the sample data above, I have created a data model which I would be explaining in future sections.

2.5 Scrape Data:

- I have written a python script to scrape data from the web resources and have done some data cleaning and modeled it to store it in the database.

2.6 Store in Database:

- I have used the same python script to save the results in the database. I would be explaining them in future sections.

3. Data Modeling

Data modeling is the process of developing data model for the data to be stored in a Database. It has 3 types of data models.

3.1. Conceptual Data Model

This data model defines of **what** the system contains. After seeing the sample data, I have decided to have 4 basic tables and they are:

Company: list of all companies
Sector: list of all sectors
Stock_price_history: Historical data of stock price for each company
Stock_news: Latest stock news data for each company.

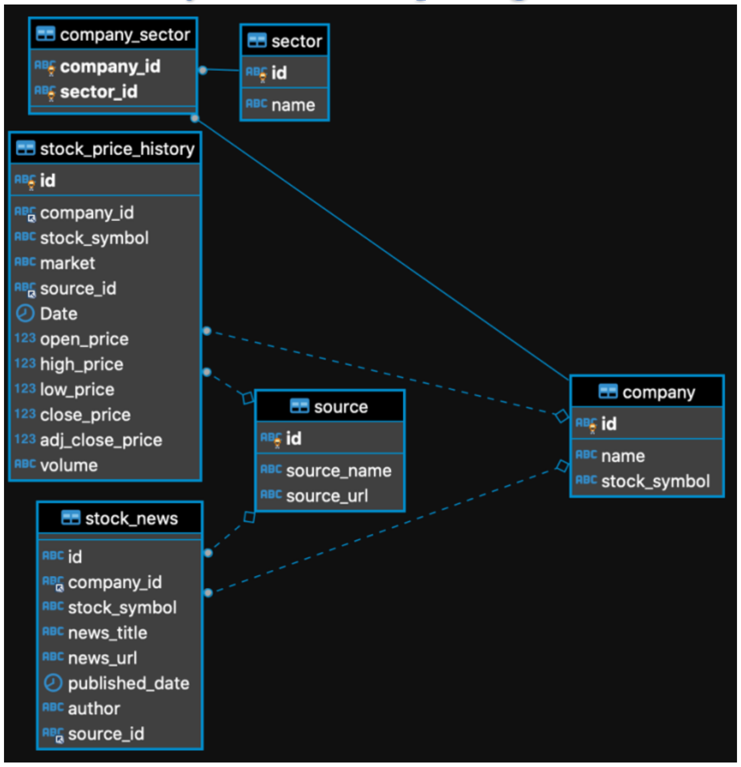
3.2. Logical Data Model

Defines **how** the system should be implemented regardless of the DBMS. The purpose is to develop technical map of rules and data structures. I have decided to add 2 more tables to the data model at this stage. I have also decided on what columns(attributes) go into each table and their data types.

3.3. Physical Data Model

This Data Model describes **HOW** the system will be implemented using a specific DBMS system. Primary and Foreign keys are defined in this stage.
Database Used: **MYSQL, IDE: Dbeaver**

3.3.1. Entity-Relationship Diagram



3.3.2. Data Definitions

company:
This table consists of data related to each company

- id (PK) – varchar (36) | generated as a unique value for each company
- name – varchar (255) | name of the company
- stock_symbol – varchar (36) | associated stock_symbol for each company

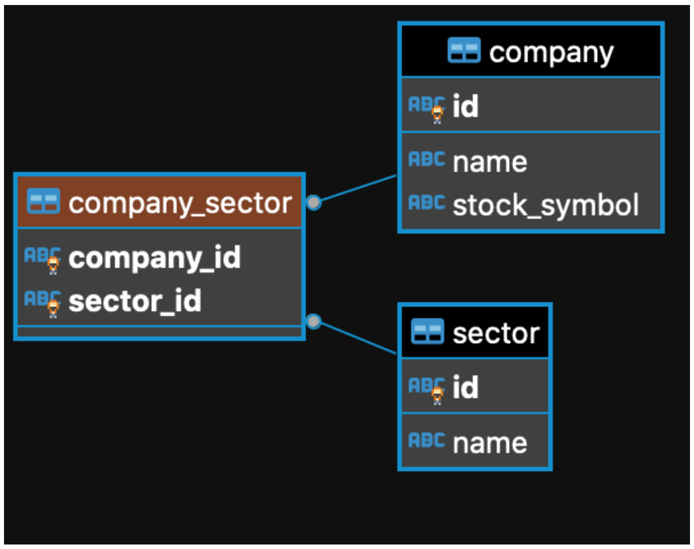
sector:
This table consists of names of each sector.

- id (PK) – varchar (36) | generated as a unique value for each sector
- name – varchar (255) | name of the sector

source:
This table consists data related to all different web sources used for scraping data. This would help us in grouping stock price or stock news data based on source and we could do some analysis over this grouped data and figure out which source contains valid information when compared to another.

- id (PK) – varchar (36) | generated as a unique value for each source
- source_name – varchar (255) | name of each source
- source_url – varchar (36) | link of each source URL that's used to scrape data

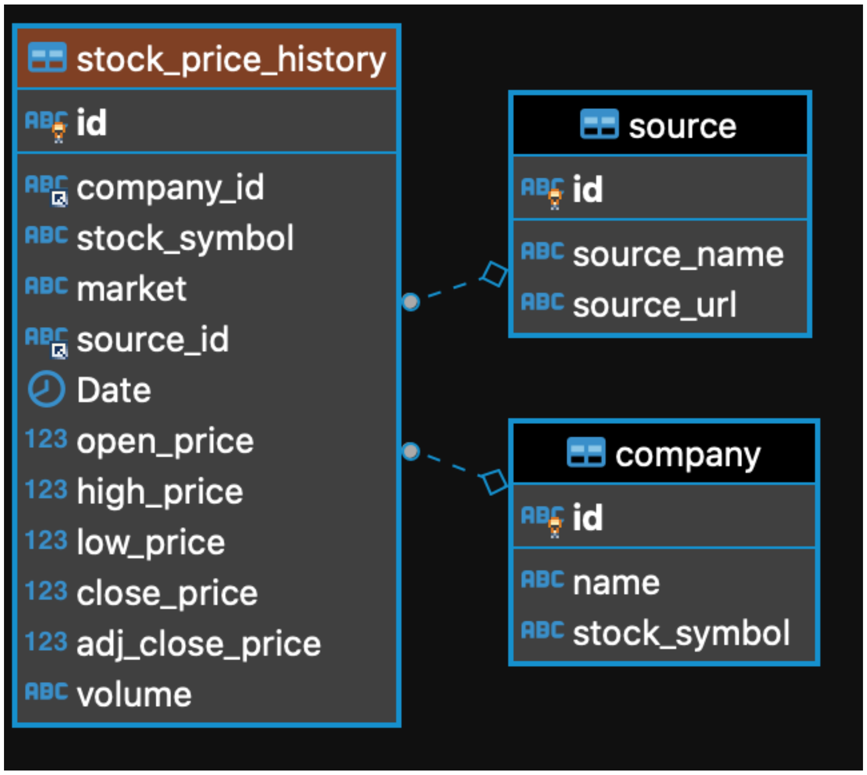
company_sector:



This table stores the relationship between companies and sector. This table could have been eliminated if there had been just one-many relationship between sector and company tables. But based on our data, I have multiple sectors for couple of companies. So, there is a many-many relationship between company and sector table. In this case, it would better to have a separate table for both of them as a standard normalization technique. I have added company_id and sector_id as foreign keys to company and sector table respectively for data consistency.

- company_id (FK) – varchar (36) | company id from company table
- sector_id (FK)- varchar (36) | sector id from sector table

stock_price_history:

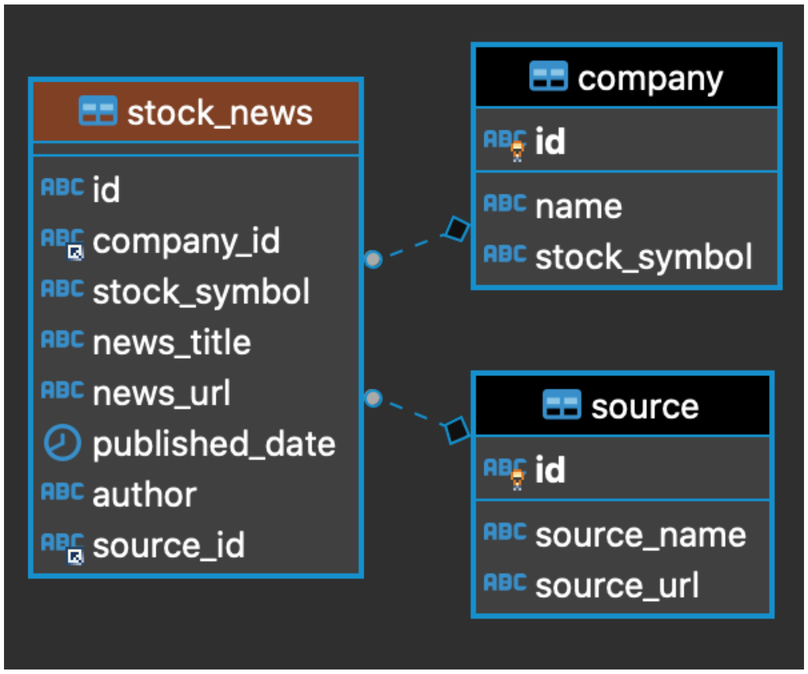


Purpose of this table is to store historical data of stock price for each company. So, company_id is added as a foreign key to company table. I am storing stock_symbol in this table just for the purpose of understanding the data in this table with naked eye. Market data is also stored which would help us to do some analysis if there is any difference in stock price across different markets. All the remaining data in this table would help us understand the performance of the stock over a period of time which would in turn reflect the company's performance. volume data would give us some insights when tied to the date information to see if volume was related to any historical events across the year. I could join this table with company table to get stock_history data for each company.

- id (PK) – varchar (36) | generated as a unique value for each record
- company_id (FK) – varchar (36) | company id from company table
- stock_symbol – varchar (36) | stock symbol of the company
- market – varchar (255) | market name for each stock price in this table

- source_id (FK) – varchar (36) | id of the source which is used to scrape this data
- date – date | date of the stock price
- open_price – double | price of the stock when the market is opened on any day
- high_price – double | highest price the stock had reached during any day
- low_price – double | lowest price the stock had reached during any day
- close_price – double | price of the stock when the market is closed on a any day
- adj_close_price – double | price is adjusted as **dividends** no longer belongs to the company
- volume – double | amount of a stock value that was traded during any day

stock_news:



This table stores latest news data related to each stock. This data would be pretty helpful for doing some sentimental analysis in future to validate or predict stock performance. I could also join this table with source table and group by source table to see what all news articles are scraped from each source. This table can also be joined with company table to get stock news related to each company.

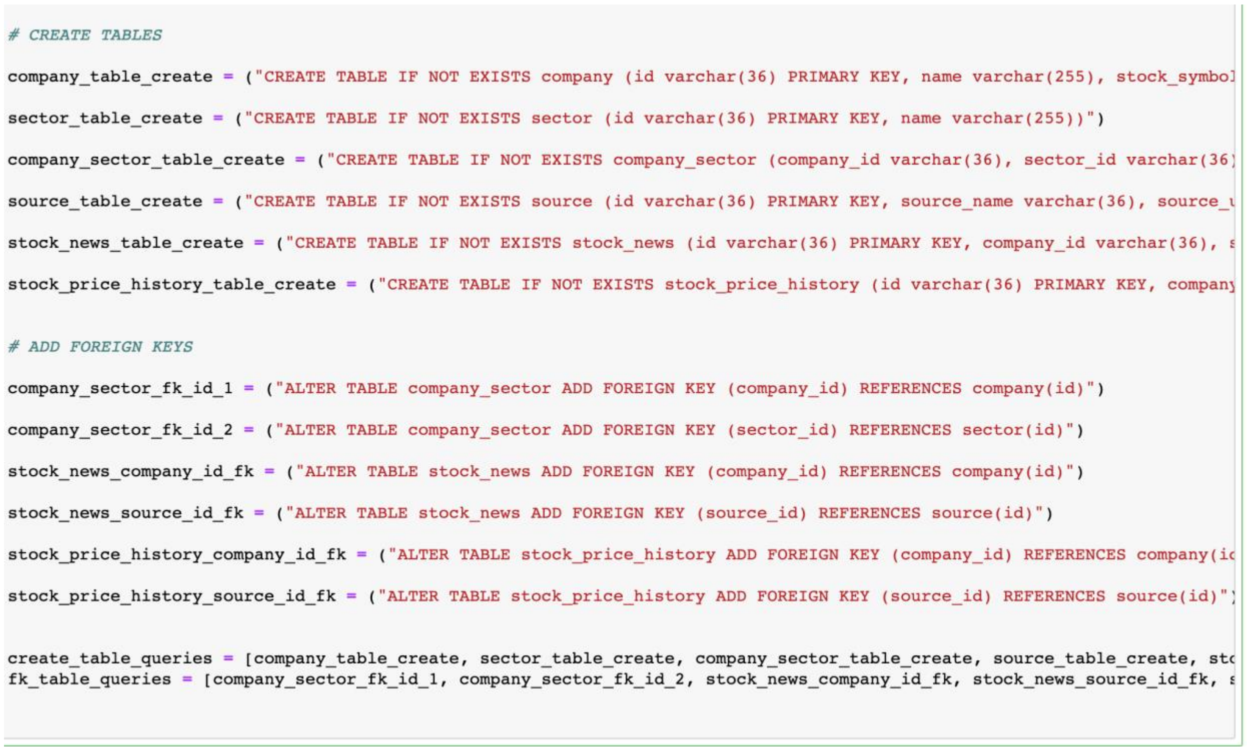
- id (PK) – varchar (36) | generated as a unique value for each record
- company_id (FK) – varchar (36) | company id from company table
- stock_symbol – varchar (36) | stock symbol of the company
- news_title – varchar (255) | title of the news article
- news_url – varchar (255) | link to the news article
- published_date – date | date when news was published
- author – varchar (255) | name of the author
- source_id – varchar (36) | id of the source which is used to scrape this data

4. Programmatic Design and Explanation

I am using python3 and jupyter Notebook to create database and tables and also to scrape data. I have also got the python script in a structured folder with .py scripts.

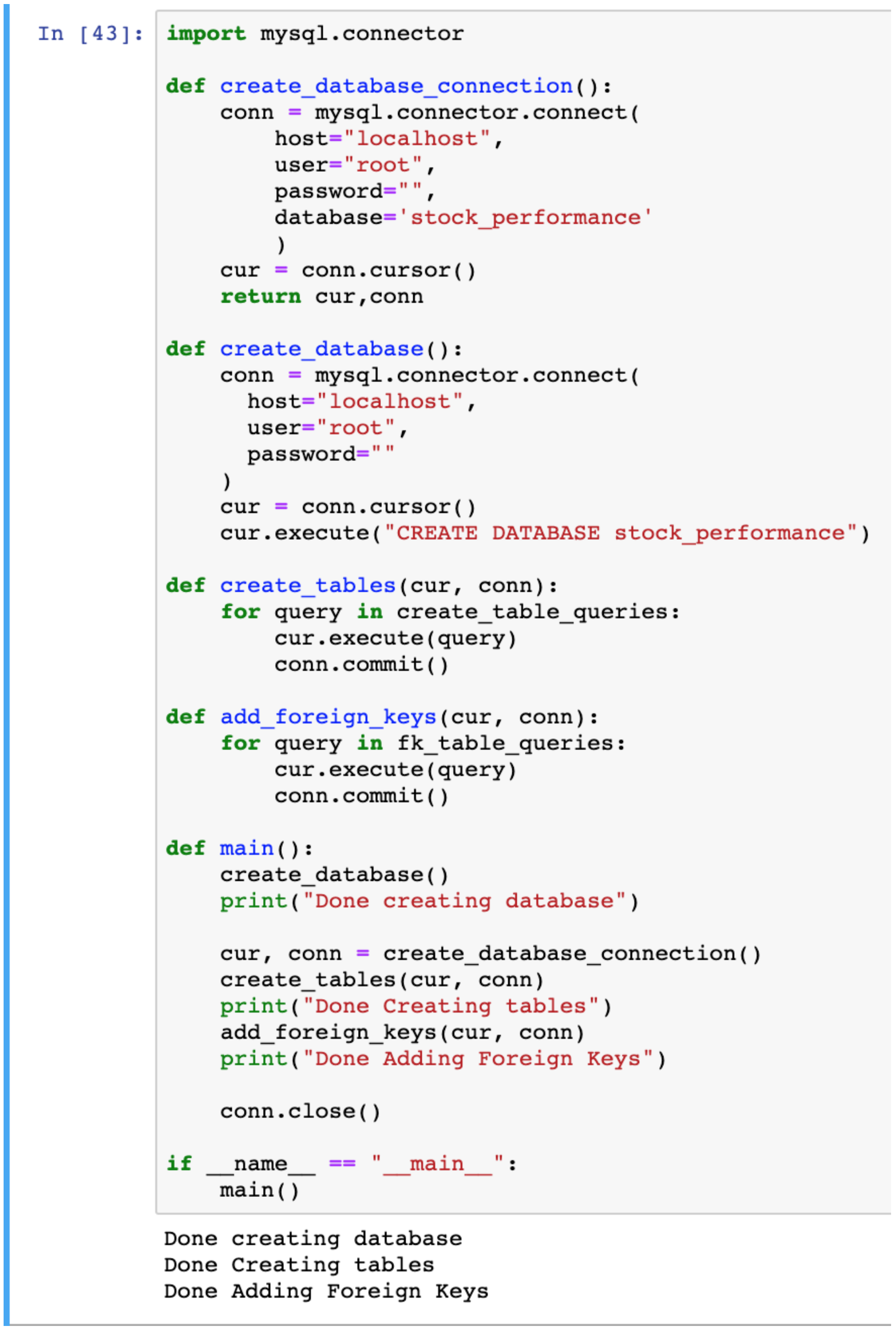
4.1. Create Database and Tables

1. Saving all queries into an array



```
# CREATE TABLES

company_table_create = ("CREATE TABLE IF NOT EXISTS company (id varchar(36) PRIMARY KEY, name varchar(255), stock_symbol

sector_table_create = ("CREATE TABLE IF NOT EXISTS sector (id varchar(36) PRIMARY KEY, name varchar(255))")

company_sector_table_create = ("CREATE TABLE IF NOT EXISTS company_sector (company_id varchar(36), sector_id varchar(36)

source_table_create = ("CREATE TABLE IF NOT EXISTS source (id varchar(36) PRIMARY KEY, source_name varchar(36), source_u

stock_news_table_create = ("CREATE TABLE IF NOT EXISTS stock_news (id varchar(36) PRIMARY KEY, company_id varchar(36), s

stock_price_history_table_create = ("CREATE TABLE IF NOT EXISTS stock_price_history (id varchar(36) PRIMARY KEY, company

# ADD FOREIGN KEYS

company_sector_fk_id_1 = ("ALTER TABLE company_sector ADD FOREIGN KEY (company_id) REFERENCES company(id)")

company_sector_fk_id_2 = ("ALTER TABLE company_sector ADD FOREIGN KEY (sector_id) REFERENCES sector(id)")

stock_news_company_id_fk = ("ALTER TABLE stock_news ADD FOREIGN KEY (company_id) REFERENCES company(id)")

stock_news_source_id_fk = ("ALTER TABLE stock_news ADD FOREIGN KEY (source_id) REFERENCES source(id)")

stock_price_history_company_id_fk = ("ALTER TABLE stock_price_history ADD FOREIGN KEY (company_id) REFERENCES company(id

stock_price_history_source_id_fk = ("ALTER TABLE stock_price_history ADD FOREIGN KEY (source_id) REFERENCES source(id)")


create_table_queries = [company_table_create, sector_table_create, company_sector_table_create, source_table_create, sto
fk_table_queries = [company_sector_fk_id_1, company_sector_fk_id_2, stock_news_company_id_fk, stock_news_source_id_fk, s
```

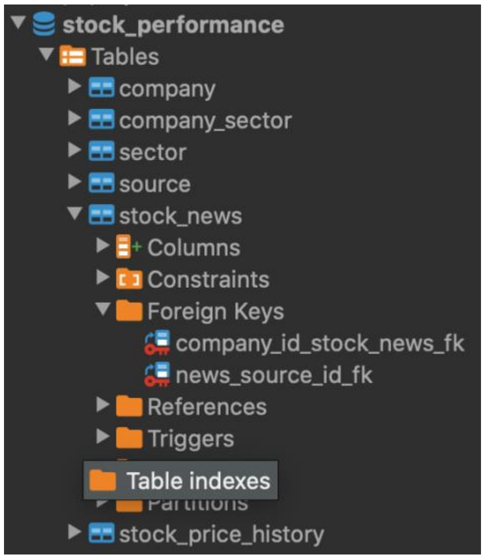
2. This script executes the above queries and creates the database, tables and foreign keys



```
In [43]: import mysql.connector

         def create_database_connection():
             conn = mysql.connector.connect(
                 host="localhost",
                 user="root",
                 password="",
                 database='stock_performance'
                 )
             cur = conn.cursor()
             return cur,conn

         def create_database():
             conn = mysql.connector.connect(
               host="localhost",
               user="root",
               password=""
             )
             cur = conn.cursor()
             cur.execute("CREATE DATABASE stock_performance")

         def create_tables(cur, conn):
             for query in create_table_queries:
                 cur.execute(query)
                 conn.commit()

         def add_foreign_keys(cur, conn):
             for query in fk_table_queries:
                 cur.execute(query)
                 conn.commit()

         def main():
             create_database()
             print("Done creating database")

             cur, conn = create_database_connection()
             create_tables(cur, conn)
             print("Done Creating tables")
             add_foreign_keys(cur, conn)
             print("Done Adding Foreign Keys")

             conn.close()

         if __name__ == "__main__":
             main()
```

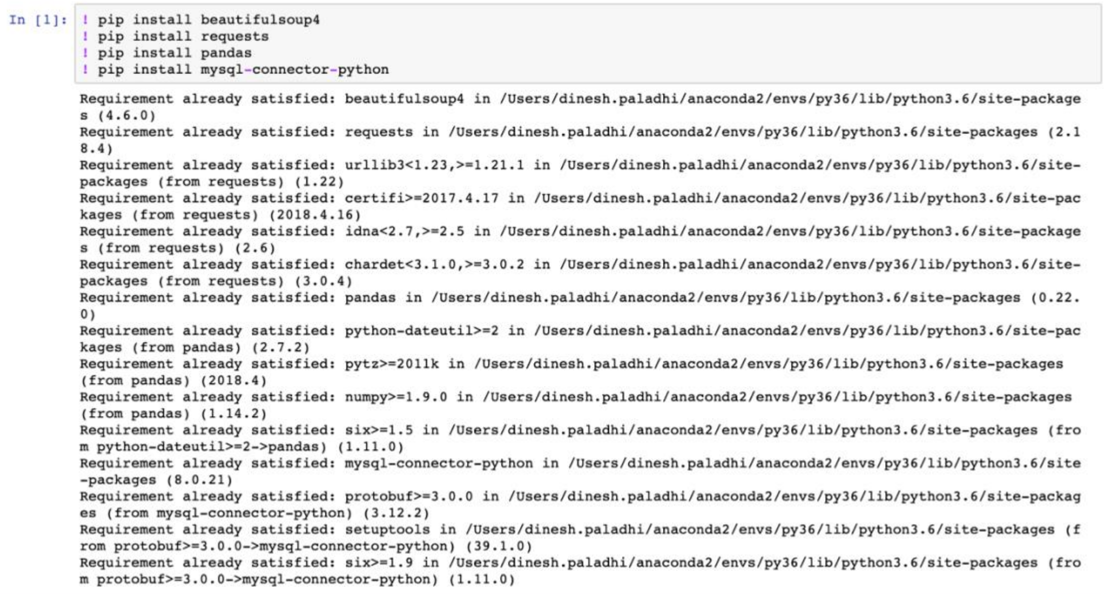
```
Done creating database
Done Creating tables
Done Adding Foreign Keys
```

3. Results:

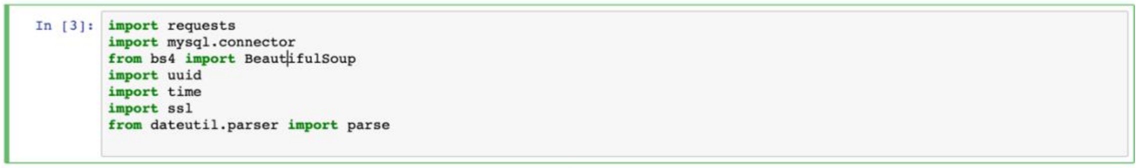


4.2. Create python script for scraping

4.2.1. Installing dependency libraries and including necessary packages



```
In [1]: ! pip install beautifulsoup4
        ! pip install requests
        ! pip install pandas
        ! pip install mysql-connector-python

Requirement already satisfied: beautifulsoup4 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-package
s (4.6.0)
Requirement already satisfied: requests in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages (2.1
8.4)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-
packages (from requests) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-pac
kages (from requests) (2018.4.16)
Requirement already satisfied: idna<2.7,>=2.5 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-package
s (from requests) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-
packages (from requests) (3.0.4)
Requirement already satisfied: pandas in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages (0.22.
0)
Requirement already satisfied: python-dateutil>=2 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-pac
kages (from pandas) (2.7.2)
Requirement already satisfied: pytz>=2011k in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages
(from pandas) (2018.4)
Requirement already satisfied: numpy>=1.9.0 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages
(from pandas) (1.14.2)
Requirement already satisfied: six>=1.5 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages (fro
m python-dateutil>=2->pandas) (1.11.0)
Requirement already satisfied: mysql-connector-python in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site
-packages (8.0.21)
Requirement already satisfied: protobuf>=3.0.0 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packag
es (from mysql-connector-python) (3.12.2)
Requirement already satisfied: setuptools in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages (f
rom protobuf>=3.0.0->mysql-connector-python) (39.1.0)
Requirement already satisfied: six>=1.9 in /Users/dinesh.paladhi/anaconda2/envs/py36/lib/python3.6/site-packages (fro
m protobuf>=3.0.0->mysql-connector-python) (1.11.0)
```



```
In [3]: import requests
        import mysql.connector
        from bs4 import BeautifulSoup
        import uuid
        import time
        import ssl
        from dateutil.parser import parse
```

BeautifulSoup4:

- I have used BeautifulSoup4 web scraping library for scraping data because it provides simple methods and easy to use to scrape data from HTML and XML documents.

- It has good community support to figure out the issues that arise while we are working with this library.

Requests

- is a simple, easy-to-use HTTP library written in Python.

mysql-connector-python

- It is written in pure Python, and it is self-sufficient to execute database queries through python.
- It is an official Oracle-supported driver to work with MySQL and python.
- It is Python 3 compatible, actively maintained.

Other

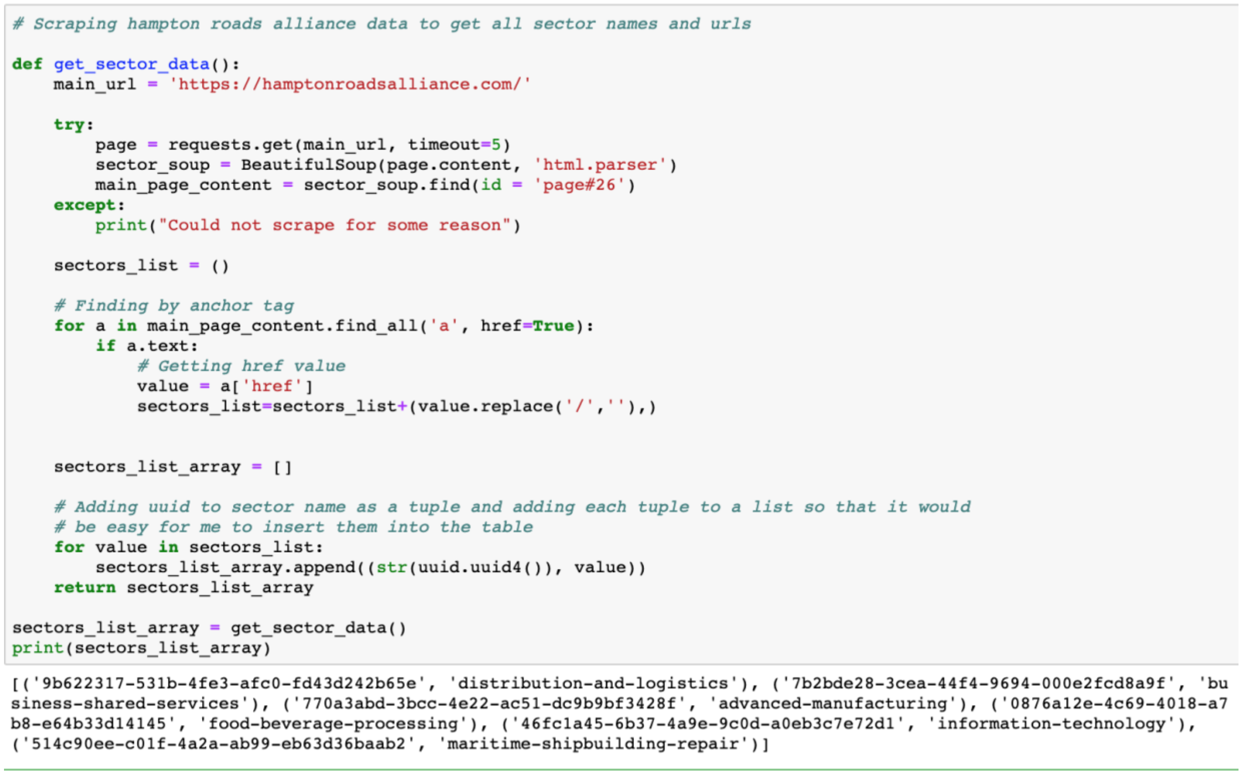
- uuid for generating unique ids for database as it is the standard way to generate ids
- time for adding some time.sleep() functions in between the logic.
- dateutil.parser for parsing date data into standard format

4.2.2. Scrape data

Overview

- I have scraped https://hamptonroadsalliance.com/ to get sector names and company data
- Scraped last 10 days data from https://finance.yahoo.com to get stock price history data
- Scraped top 5 news data from https://finance.yahoo.com to get news data
- Scraped https://seekingalpha.com/page/feeds to get stock news data.

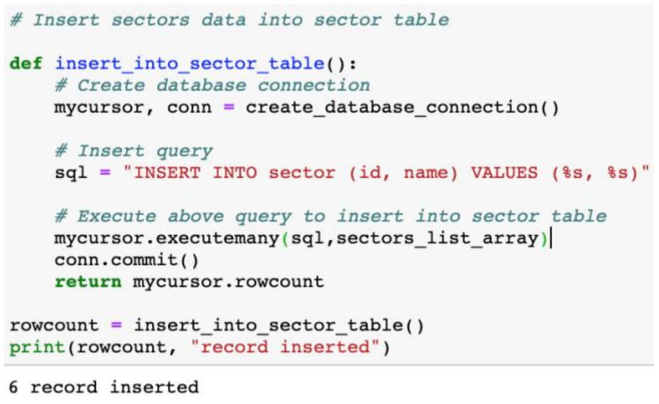
1. I would be scraping web resource #1 (https://hamptonroadsalliance.com/) website to get all sector names in a list



```python
# Scraping hampton roads alliance data to get all sector names and urls

def get_sector_data():
    main_url = 'https://hamptonroadsalliance.com/'

    try:
        page = requests.get(main_url, timeout=5)
        sector_soup = BeautifulSoup(page.content, 'html.parser')
        main_page_content = sector_soup.find(id = 'page#26')
    except:
        print("Could not scrape for some reason")

    sectors_list = ()

    # Finding by anchor tag
    for a in main_page_content.find_all('a', href=True):
        if a.text:
            # Getting href value
            value = a['href']
            sectors_list=sectors_list+(value.replace('/',''),)


    sectors_list_array = []

    # Adding uuid to sector name as a tuple and adding each tuple to a list so that it would
    # be easy for me to insert them into the table
    for value in sectors_list:
        sectors_list_array.append((str(uuid.uuid4()), value))
    return sectors_list_array

sectors_list_array = get_sector_data()
print(sectors_list_array)
```

```
[('9b622317-531b-4fe3-afc0-fd43d242b65e', 'distribution-and-logistics'), ('7b2bde28-3cea-44f4-9694-000e2fcd8a9f', 'bu
siness-shared-services'), ('770a3abd-3bcc-4e22-ac51-dc9b9bf3428f', 'advanced-manufacturing'), ('0876a12e-4c69-4018-a7
b8-e64b33d14145', 'food-beverage-processing'), ('46fc1a45-6b37-4a9e-9c0d-a0eb3c7e72d1', 'information-technology'),
('514c90ee-c01f-4a2a-ab99-eb63d36baab2', 'maritime-shipbuilding-repair')]
```

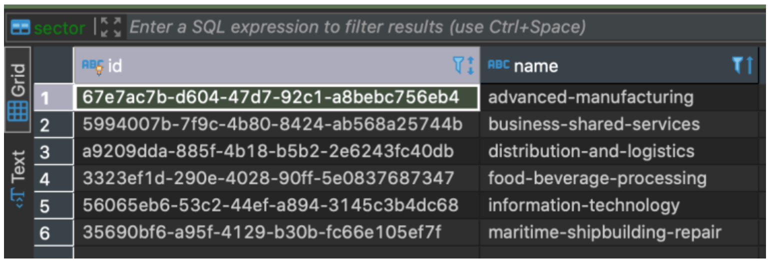
2. Next Step is to save the scraped data into sector table



```
# Insert sectors data into sector table

def insert_into_sector_table():
    # Create database connection
    mycursor, conn = create_database_connection()

    # Insert query
    sql = "INSERT INTO sector (id, name) VALUES (%s, %s)"

    # Execute above query to insert into sector table
    mycursor.executemany(sql,sectors_list_array)
    conn.commit()
    return mycursor.rowcount

rowcount = insert_into_sector_table()
print(rowcount, "record inserted")
```

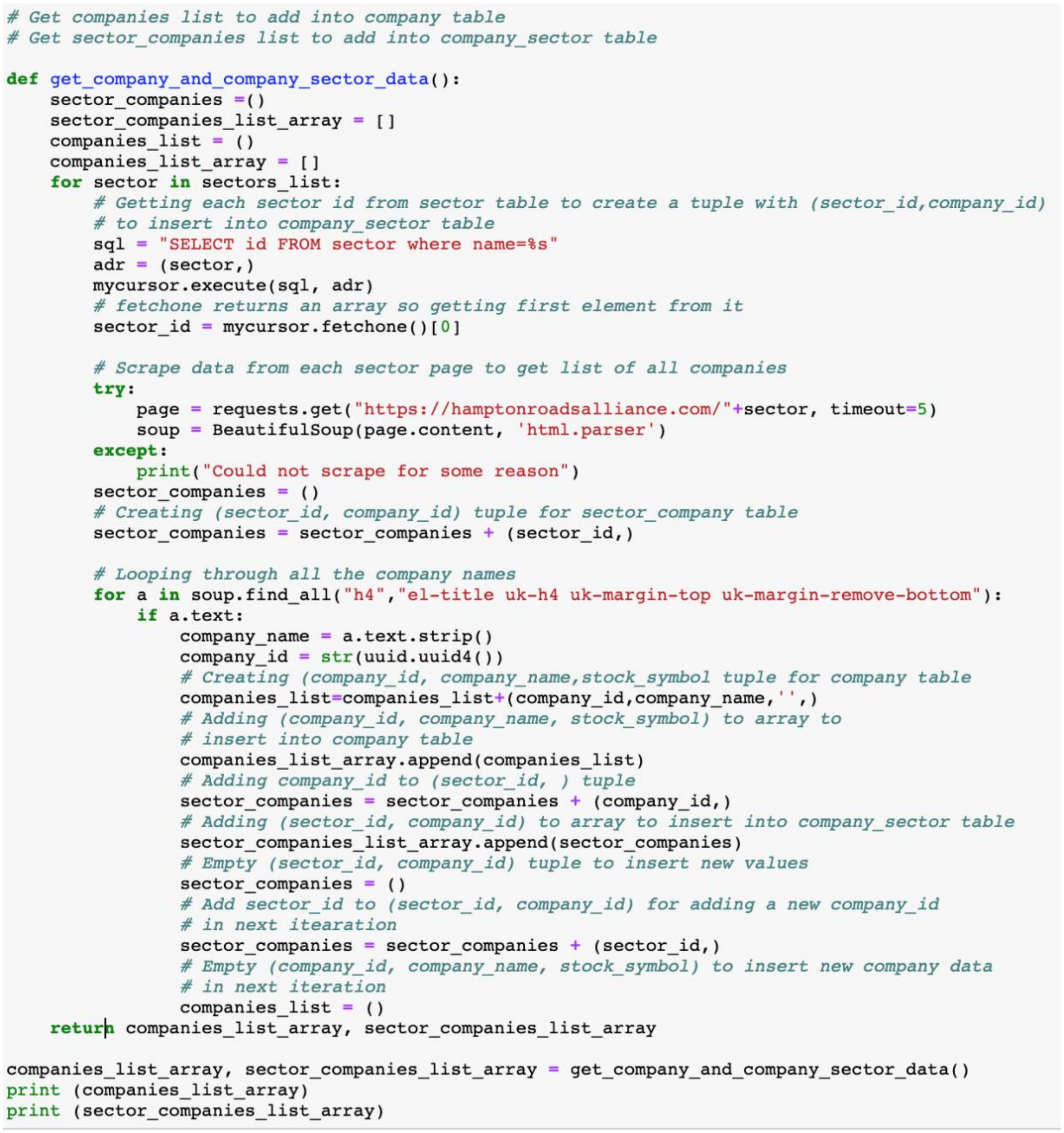
```
6 record inserted
```

Results:



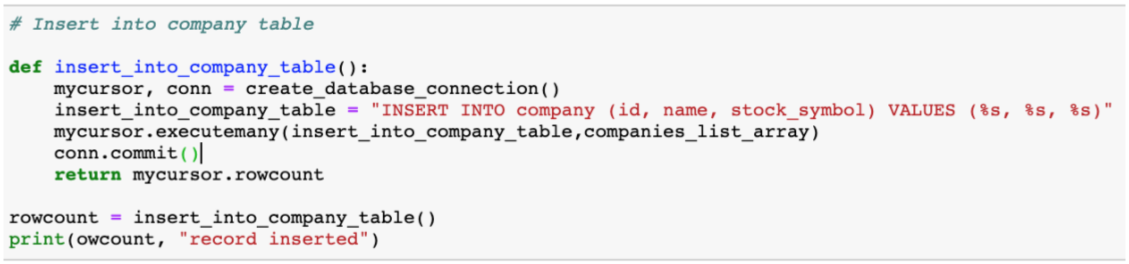
	id	name
1	67e7ac7b-d604-47d7-92c1-a8bebc756eb4	advanced-manufacturing
2	5994007b-7f9c-4b80-8424-ab568a25744b	business-shared-services
3	a9209dda-885f-4b18-b5b2-2e6243fc40db	distribution-and-logistics
4	3323ef1d-290e-4028-90ff-5e0837687347	food-beverage-processing
5	56065eb6-53c2-44ef-a894-3145c3b4dc68	information-technology
6	35690bf6-a95f-4129-b30b-fc66e105ef7f	maritime-shipbuilding-repair

3. Scrape each sector webpage to get list of companies from each webpage and format data to insert it into company table and company_sector table.



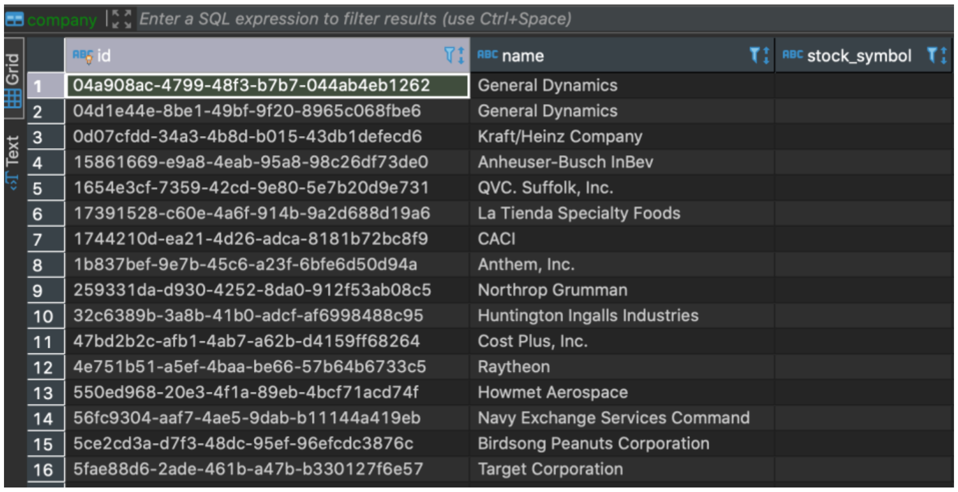
```python
# Get companies list to add into company table
# Get sector_companies list to add into company_sector table

def get_company_and_company_sector_data():
    sector_companies =()
    sector_companies_list_array = []
    companies_list = ()
    companies_list_array = []
    for sector in sectors_list:
        # Getting each sector id from sector table to create a tuple with (sector_id,company_id)
        # to insert into company_sector table
        sql = "SELECT id FROM sector where name=%s"
        adr = (sector,)
        mycursor.execute(sql, adr)
        # fetchone returns an array so getting first element from it
        sector_id = mycursor.fetchone()[0]

        # Scrape data from each sector page to get list of all companies
        try:
            page = requests.get("https://hamptonroadsalliance.com/"+sector, timeout=5)
            soup = BeautifulSoup(page.content, 'html.parser')
        except:
            print("Could not scrape for some reason")
        sector_companies = ()
        # Creating (sector_id, company_id) tuple for sector_company table
        sector_companies = sector_companies + (sector_id,)

        # Looping through all the company names
        for a in soup.find_all("h4","el-title uk-h4 uk-margin-top uk-margin-remove-bottom"):
            if a.text:
                company_name = a.text.strip()
                company_id = str(uuid.uuid4())
                # Creating (company_id, company_name,stock_symbol tuple for company table
                companies_list=companies_list+(company_id,company_name,'',)
                # Adding (company_id, company_name, stock_symbol) to array to
                # insert into company table
                companies_list_array.append(companies_list)
                # Adding company_id to (sector_id, ) tuple
                sector_companies = sector_companies + (company_id,)
                # Adding (sector_id, company_id) to array to insert into company_sector table
                sector_companies_list_array.append(sector_companies)
                # Empty (sector_id, company_id) tuple to insert new values
                sector_companies = ()
                # Add sector_id to (sector_id, company_id) for adding a new company_id
                # in next itearation
                sector_companies = sector_companies + (sector_id,)
                # Empty (company_id, company_name, stock_symbol) to insert new company data
                # in next iteration
                companies_list = ()
    return companies_list_array, sector_companies_list_array

companies_list_array, sector_companies_list_array = get_company_and_company_sector_data()
print (companies_list_array)
print (sector_companies_list_array)
```

4. Insert the above data into company table



```python
# Insert into company table

def insert_into_company_table():
    mycursor, conn = create_database_connection()
    insert_into_company_table = "INSERT INTO company (id, name, stock_symbol) VALUES (%s, %s, %s)"
    mycursor.executemany(insert_into_company_table,companies_list_array)
    conn.commit()
    return mycursor.rowcount

rowcount = insert_into_company_table()
print(owcount, "record inserted")
```

Results:


company | Enter a SQL expression to filter results (use Ctrl+Space)

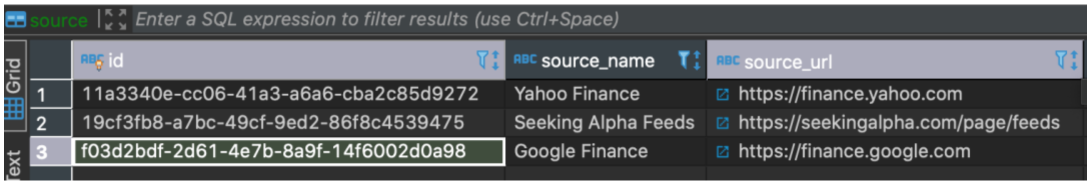
	id	name	stock_symbol
1	04a908ac-4799-48f3-b7b7-044ab4eb1262	General Dynamics	
2	04d1e44e-8be1-49bf-9f20-8965c068fbe6	General Dynamics	
3	0d07cfdd-34a3-4b8d-b015-43db1defecd6	Kraft/Heinz Company	
4	15861669-e9a8-4eab-95a8-98c26df73de0	Anheuser-Busch InBev	
5	1654e3cf-7359-42cd-9e80-5e7b20d9e731	QVC. Suffolk, Inc.	
6	17391528-c60e-4a6f-914b-9a2d688d19a6	La Tienda Specialty Foods	
7	1744210d-ea21-4d26-adca-8181b72bc8f9	CACI	
8	1b837bef-9e7b-45c6-a23f-6bfe6d50d94a	Anthem, Inc.	
9	259331da-d930-4252-8da0-912f53ab08c5	Northrop Grumman	
10	32c6389b-3a8b-41b0-adcf-af6998488c95	Huntington Ingalls Industries	
11	47bd2b2c-afb1-4ab7-a62b-d4159ff68264	Cost Plus, Inc.	
12	4e751b51-a5ef-4baa-be66-57b64b6733c5	Raytheon	
13	550ed968-20e3-4f1a-89eb-4bcf71acd74f	Howmet Aerospace	
14	56fc9304-aaf7-4ae5-9dab-b11144a419eb	Navy Exchange Services Command	
15	5ce2cd3a-d7f3-48dc-95ef-96efcdc3876c	Birdsong Peanuts Corporation	
16	5fae88d6-2ade-461b-a47b-b330127f6e57	Target Corporation	

5. Insert company_sector data into company_sector table

```
# Insert into and company_sector table

def insert_into_company_sector_table():
    mycursor, conn = create_database_connection()
    insert_into_company_sector_table = "INSERT INTO company_sector (sector_id, company_id) VALUES (%s, %s)"
    mycursor.executemany(insert_into_company_sector_table,sector_companies_list_array)
    conn.commit()
    return mycursor.rowcount

rowcount = insert_into_company_sector_table()
print(rowcount, "record inserted")
```

```
48 record inserted
```

6. I have manually inserted stock_symbol data into company table as it was tough to get stock symbol from company name and moreover some of the companies are not public so I did not get any stock symbol for them.
7. I have also added data manually into source table


source | Enter a SQL expression to filter results (use Ctrl+Space)

	id	source_name	source_url
1	11a3340e-cc06-41a3-a6a6-cba2c85d9272	Yahoo Finance	https://finance.yahoo.com
2	19cf3fb8-a7bc-49cf-9ed2-86f8c4539475	Seeking Alpha Feeds	https://seekingalpha.com/page/feeds
3	f03d2bdf-2d61-4e7b-8a9f-14f6002d0a98	Google Finance	https://finance.google.com

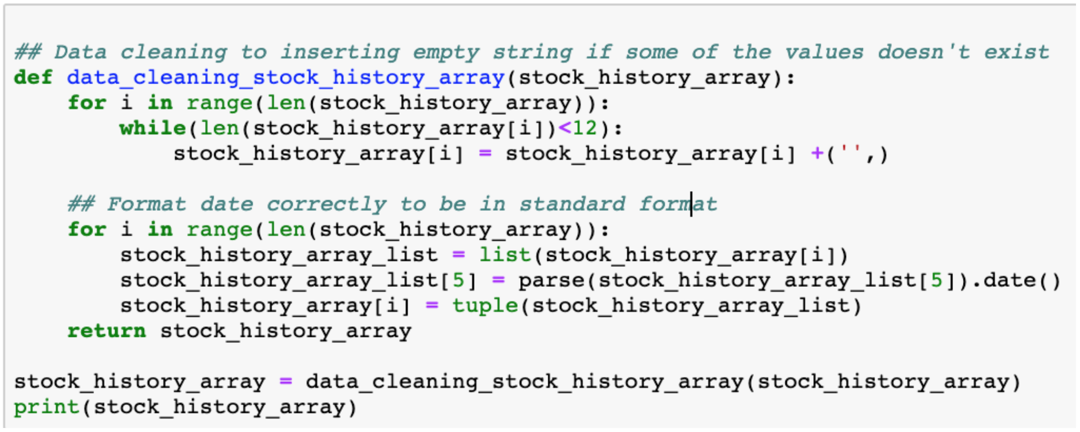
8. Get company data to use for scraping news and stock history data

```python
# Get company data from database so that it can be used it later
# functions to get news and stock history data
def get_company_data():
    mycursor, conn = create_database_connection()
    # Get companies with not null stock_symbol
    sql = "SELECT id,stock_symbol FROM company where stock_symbol is not null"
    mycursor.execute(sql)
    company_data = mycursor.fetchall()
    return company_data

company_data = get_company_data()
print(company_data)
```

9. Scrape last 10 days **stock price history** data from **Yahoo finance (**https://finance.yahoo.com)

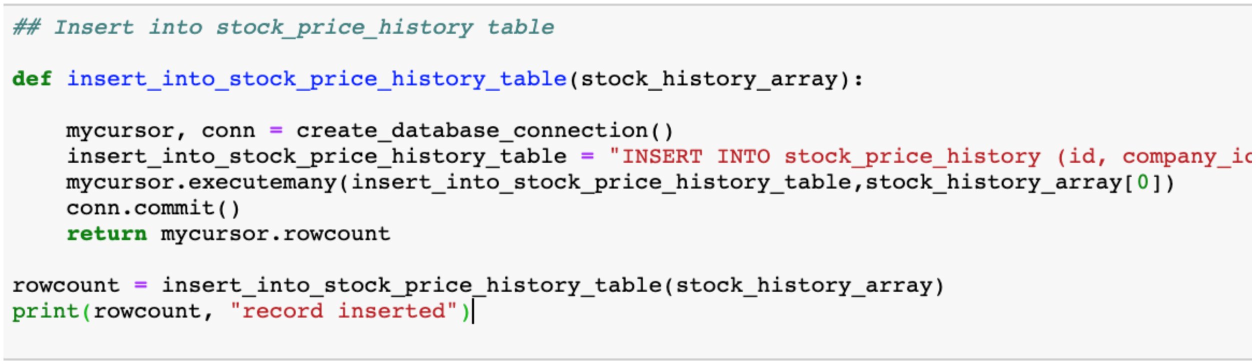
```python
## Construct data for inserting into stock_price_history table
def get_yahoo_stock_price_history_data():

    mycursor, conn = create_database_connection()
    # Get source id to insert it into stock_price_history table
    sql = "SELECT id FROM source where source_name='Yahoo Finance'"
    mycursor.execute(sql)
    price_source_id = mycursor.fetchone()[0]

    stock_history_array = []
    stock_history = ()

    # Loop through each company
    for i in company_data:
        ## Get Market information
        try:
            page = requests.get('https://finance.yahoo.com/quote/'+i[1], timeout=5)
            stock_market_soup = BeautifulSoup(page.text, 'html.parser')
            market = stock_market_soup.find('div',{'id':'quote-header-info'}).find('span').text
        except:
            print("Could not scrape for some reason")
        print(i[1],market)
        time.sleep(1)

        ## Get Stock_history
        try:
            page = requests.get('https://finance.yahoo.com/quote/'+i[1]+'/history', timeout=5)
            stock_history_soup = BeautifulSoup(page.text, 'html.parser')
        except:
            print("Could not scrape for some reason")
        count = 0
    #     stock_history = (uuid_value,i[0],i[1],market,)
        for item in stock_history_soup.find('table',{'data-test':'historical-prices'}).find('tbody'):
            uuid_value = str(uuid.uuid4())
            stock_history = (uuid_value,i[0],i[1],market,price_source_id,)
            if(count < 10):
                count = count + 1
                for value in item.find_all('span'):
                    stock_history = stock_history + (value.text,)
                stock_history_array.append(stock_history)
#         time.sleep(1)

    return stock_history_array
stock_history_array = get_yahoo_stock_price_history_data()
print(stock_history_array)
```

10. Next I would be doing some data cleaning to insert empty string for records where value doesn't exist and also to convert date into a standard format.



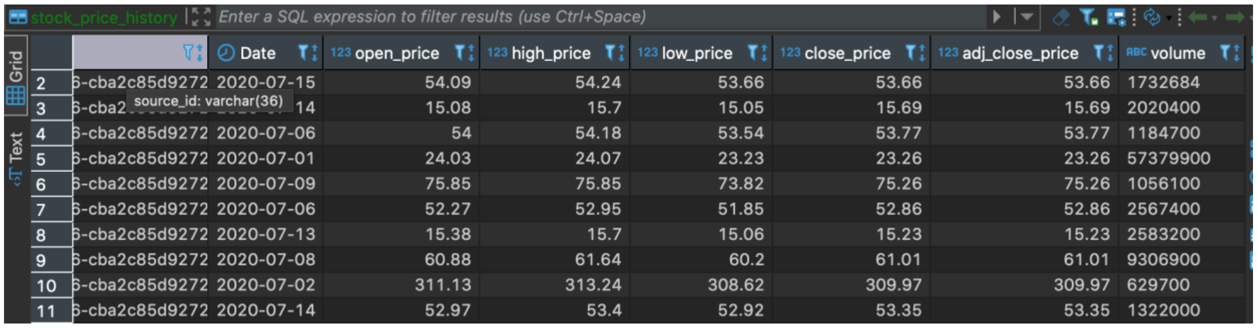
```
## Data cleaning to inserting empty string if some of the values doesn't exist
def data_cleaning_stock_history_array(stock_history_array):
    for i in range(len(stock_history_array)):
        while(len(stock_history_array[i])<12):
            stock_history_array[i] = stock_history_array[i] +('',)

    ## Format date correctly to be in standard format
    for i in range(len(stock_history_array)):
        stock_history_array_list = list(stock_history_array[i])
        stock_history_array_list[5] = parse(stock_history_array_list[5]).date()
        stock_history_array[i] = tuple(stock_history_array_list)
    return stock_history_array

stock_history_array = data_cleaning_stock_history_array(stock_history_array)
print(stock_history_array)
```

11. Inserting data into stock_price_history table



```
## Insert into stock_price_history table

def insert_into_stock_price_history_table(stock_history_array):

    mycursor, conn = create_database_connection()
    insert_into_stock_price_history_table = "INSERT INTO stock_price_history (id, company_ic
    mycursor.executemany(insert_into_stock_price_history_table,stock_history_array[0])
    conn.commit()
    return mycursor.rowcount

rowcount = insert_into_stock_price_history_table(stock_history_array)
print(rowcount, "record inserted")
```

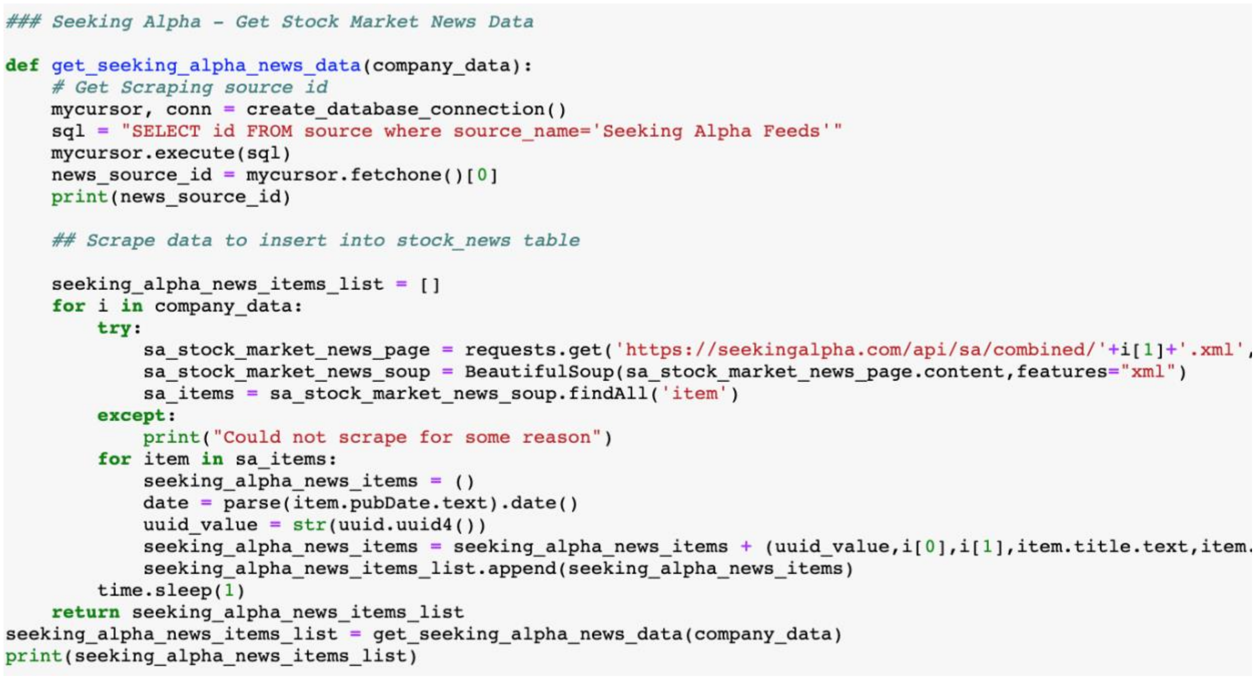
Results:



stock_price_history | Enter a SQL expression to filter results (use Ctrl+Space)

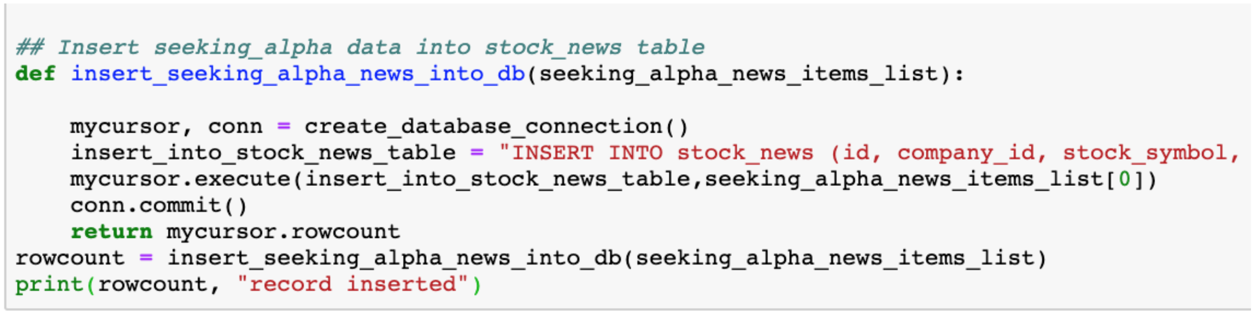
		Date	open_price	high_price	low_price	close_price	adj_close_price	volume
2	6-cba2c85d9272	2020-07-15	54.09	54.24	53.66	53.66	53.66	1732684
3	6-cba2 source_id: varchar(36)	14	15.08	15.7	15.05	15.69	15.69	2020400
4	6-cba2c85d9272	2020-07-06	54	54.18	53.54	53.77	53.77	1184700
5	6-cba2c85d9272	2020-07-01	24.03	24.07	23.23	23.26	23.26	57379900
6	6-cba2c85d9272	2020-07-09	75.85	75.85	73.82	75.26	75.26	1056100
7	6-cba2c85d9272	2020-07-06	52.27	52.95	51.85	52.86	52.86	2567400
8	6-cba2c85d9272	2020-07-13	15.38	15.7	15.06	15.23	15.23	2583200
9	6-cba2c85d9272	2020-07-08	60.88	61.64	60.2	61.01	61.01	9306900
10	6-cba2c85d9272	2020-07-02	311.13	313.24	308.62	309.97	309.97	629700
11	6-cba2c85d9272	2020-07-14	52.97	53.4	52.92	53.35	53.35	1322000

12. Scrape news data from **Seeking Alpha (**https://seekingalpha.com/page/feeds)



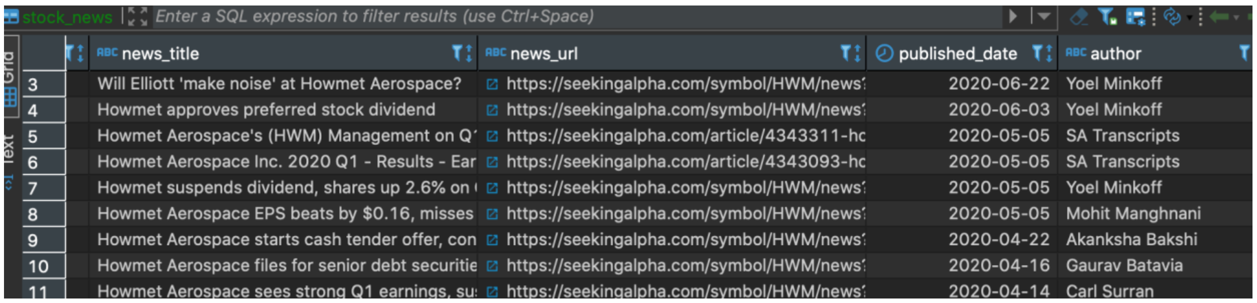
```
### Seeking Alpha - Get Stock Market News Data

def get_seeking_alpha_news_data(company_data):
    # Get Scraping source id
    mycursor, conn = create_database_connection()
    sql = "SELECT id FROM source where source_name='Seeking Alpha Feeds'"
    mycursor.execute(sql)
    news_source_id = mycursor.fetchone()[0]
    print(news_source_id)

    ## Scrape data to insert into stock_news table

    seeking_alpha_news_items_list = []
    for i in company_data:
        try:
            sa_stock_market_news_page = requests.get('https://seekingalpha.com/api/sa/combined/'+i[1]+'.xml',
            sa_stock_market_news_soup = BeautifulSoup(sa_stock_market_news_page.content,features="xml")
            sa_items = sa_stock_market_news_soup.findAll('item')
        except:
            print("Could not scrape for some reason")
        for item in sa_items:
            seeking_alpha_news_items = ()
            date = parse(item.pubDate.text).date()
            uuid_value = str(uuid.uuid4())
            seeking_alpha_news_items = seeking_alpha_news_items + (uuid_value,i[0],i[1],item.title.text,item.
            seeking_alpha_news_items_list.append(seeking_alpha_news_items)
        time.sleep(1)
    return seeking_alpha_news_items_list
seeking_alpha_news_items_list = get_seeking_alpha_news_data(company_data)
print(seeking_alpha_news_items_list)
```

13. Insert seeking alpha news data into stock_news table



```
## Insert seeking_alpha data into stock_news table
def insert_seeking_alpha_news_into_db(seeking_alpha_news_items_list):

    mycursor, conn = create_database_connection()
    insert_into_stock_news_table = "INSERT INTO stock_news (id, company_id, stock_symbol,
    mycursor.execute(insert_into_stock_news_table,seeking_alpha_news_items_list[0])
    conn.commit()
    return mycursor.rowcount
rowcount = insert_seeking_alpha_news_into_db(seeking_alpha_news_items_list)
print(rowcount, "record inserted")
```

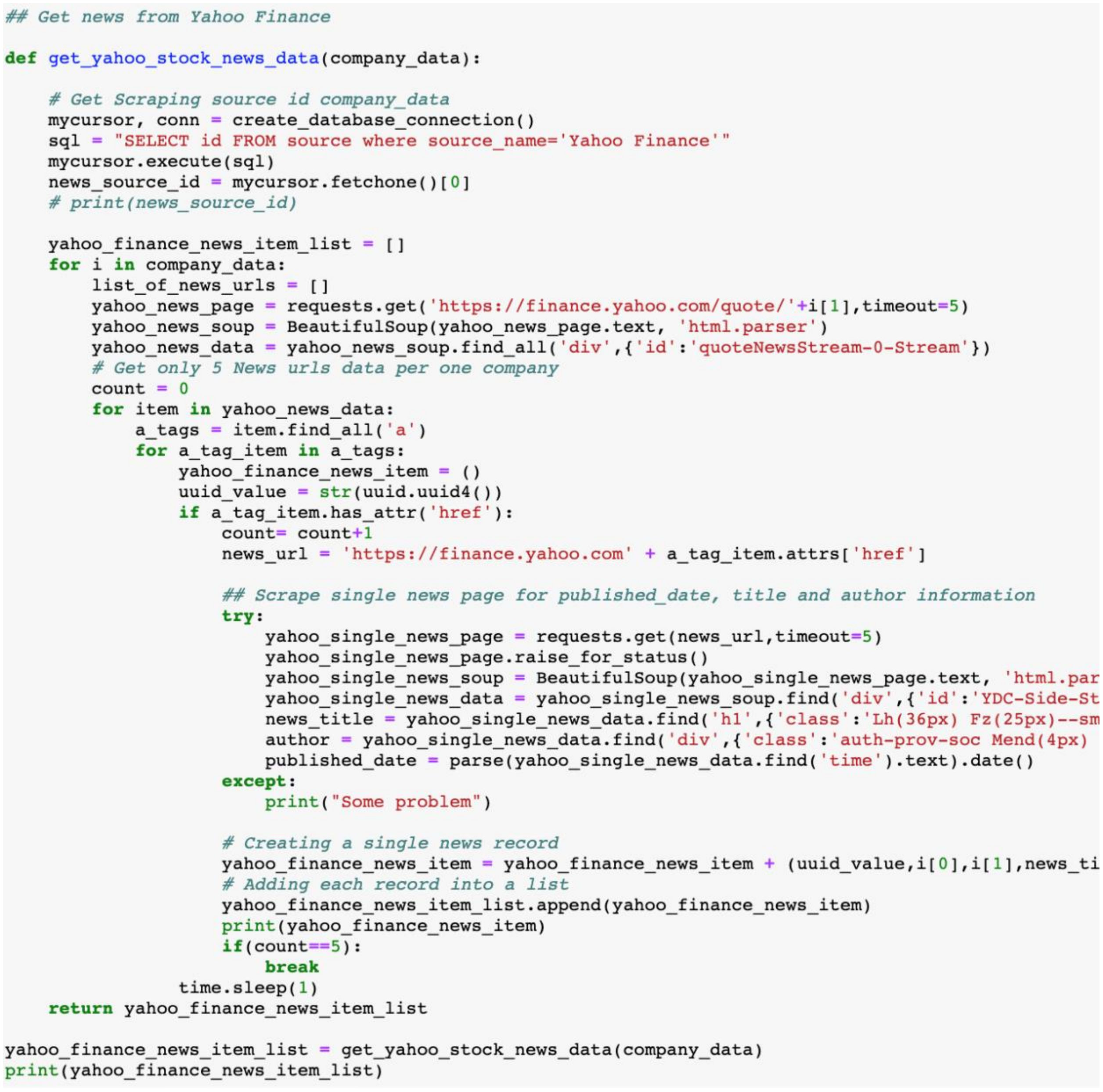
Results:



stock_news | Enter a SQL expression to filter results (use Ctrl+Space)

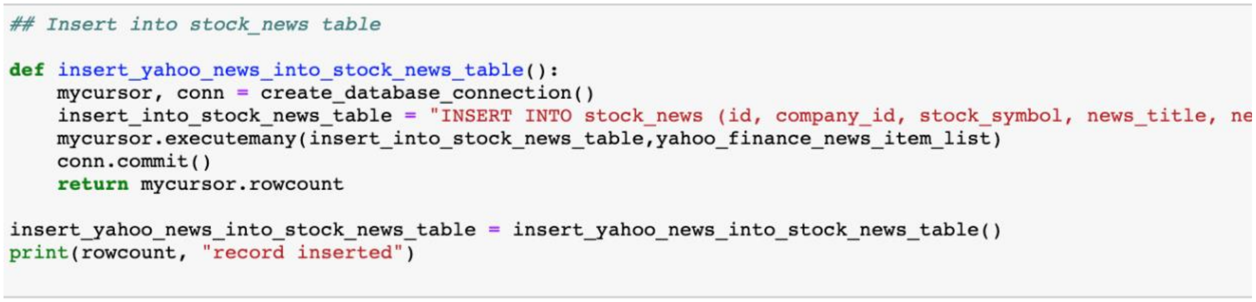
	news_title	news_url	published_date	author
3	Will Elliott 'make noise' at Howmet Aerospace?	https://seekingalpha.com/symbol/HWM/news?	2020-06-22	Yoel Minkoff
4	Howmet approves preferred stock dividend	https://seekingalpha.com/symbol/HWM/news?	2020-06-03	Yoel Minkoff
5	Howmet Aerospace's (HWM) Management on Q1	https://seekingalpha.com/article/4343311-hc	2020-05-05	SA Transcripts
6	Howmet Aerospace Inc. 2020 Q1 - Results - Ear	https://seekingalpha.com/article/4343093-hc	2020-05-05	SA Transcripts
7	Howmet suspends dividend, shares up 2.6% on	https://seekingalpha.com/symbol/HWM/news?	2020-05-05	Yoel Minkoff
8	Howmet Aerospace EPS beats by $0.16, misses	https://seekingalpha.com/symbol/HWM/news?	2020-05-05	Mohit Manghnani
9	Howmet Aerospace starts cash tender offer, con	https://seekingalpha.com/symbol/HWM/news?	2020-04-22	Akanksha Bakshi
10	Howmet Aerospace files for senior debt securitie	https://seekingalpha.com/symbol/HWM/news?	2020-04-16	Gaurav Batavia
11	Howmet Aerospace sees strong Q1 earnings, su	https://seekingalpha.com/symbol/HWM/news?	2020-04-14	Carl Surran

14. Scraping top 5 news data from **Yahoo Finance** (https://finance.yahoo.com) for each company



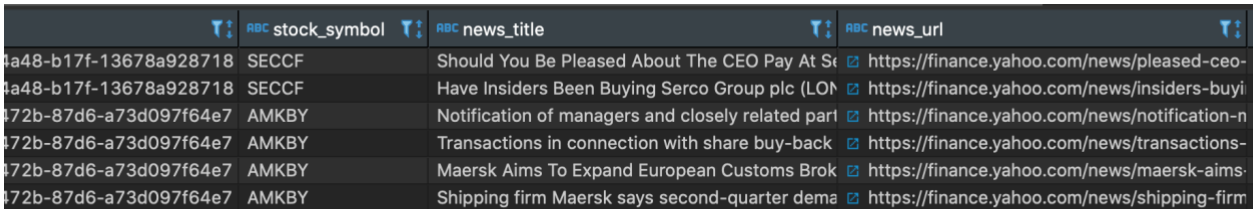
```
## Get news from Yahoo Finance

def get_yahoo_stock_news_data(company_data):

    # Get Scraping source id company_data
    mycursor, conn = create_database_connection()
    sql = "SELECT id FROM source where source_name='Yahoo Finance'"
    mycursor.execute(sql)
    news_source_id = mycursor.fetchone()[0]
    # print(news_source_id)

    yahoo_finance_news_item_list = []
    for i in company_data:
        list_of_news_urls = []
        yahoo_news_page = requests.get('https://finance.yahoo.com/quote/'+i[1],timeout=5)
        yahoo_news_soup = BeautifulSoup(yahoo_news_page.text, 'html.parser')
        yahoo_news_data = yahoo_news_soup.find_all('div',{'id':'quoteNewsStream-0-Stream'})
        # Get only 5 News urls data per one company
        count = 0
        for item in yahoo_news_data:
            a_tags = item.find_all('a')
            for a_tag_item in a_tags:
                yahoo_finance_news_item = ()
                uuid_value = str(uuid.uuid4())
                if a_tag_item.has_attr('href'):
                    count= count+1
                    news_url = 'https://finance.yahoo.com' + a_tag_item.attrs['href']

                    ## Scrape single news page for published_date, title and author information
                    try:
                        yahoo_single_news_page = requests.get(news_url,timeout=5)
                        yahoo_single_news_page.raise_for_status()
                        yahoo_single_news_soup = BeautifulSoup(yahoo_single_news_page.text, 'html.par
                        yahoo_single_news_data = yahoo_single_news_soup.find('div',{'id':'YDC-Side-St
                        news_title = yahoo_single_news_data.find('h1',{'class':'Lh(36px) Fz(25px)--sm
                        author = yahoo_single_news_data.find('div',{'class':'auth-prov-soc Mend(4px)
                        published_date = parse(yahoo_single_news_data.find('time').text).date()
                    except:
                        print("Some problem")

                    # Creating a single news record
                    yahoo_finance_news_item = yahoo_finance_news_item + (uuid_value,i[0],i[1],news_ti
                    # Adding each record into a list
                    yahoo_finance_news_item_list.append(yahoo_finance_news_item)
                    print(yahoo_finance_news_item)
                    if(count==5):
                        break
                time.sleep(1)
    return yahoo_finance_news_item_list

yahoo_finance_news_item_list = get_yahoo_stock_news_data(company_data)
print(yahoo_finance_news_item_list)
```

15. Insert yahoo news into stock_news table



```
## Insert into stock_news table

def insert_yahoo_news_into_stock_news_table():
    mycursor, conn = create_database_connection()
    insert_into_stock_news_table = "INSERT INTO stock_news (id, company_id, stock_symbol, news_title, ne
    mycursor.executemany(insert_into_stock_news_table,yahoo_finance_news_item_list)
    conn.commit()
    return mycursor.rowcount

insert_yahoo_news_into_stock_news_table = insert_yahoo_news_into_stock_news_table()
print(rowcount, "record inserted")
```

Results:



	stock_symbol	news_title	news_url
4a48-b17f-13678a928718	SECCF	Should You Be Pleased About The CEO Pay At Se	https://finance.yahoo.com/news/pleased-ceo-
4a48-b17f-13678a928718	SECCF	Have Insiders Been Buying Serco Group plc (LON	https://finance.yahoo.com/news/insiders-buyi
472b-87d6-a73d097f64e7	AMKBY	Notification of managers and closely related part	https://finance.yahoo.com/news/notification-n
472b-87d6-a73d097f64e7	AMKBY	Transactions in connection with share buy-back	https://finance.yahoo.com/news/transactions-
472b-87d6-a73d097f64e7	AMKBY	Maersk Aims To Expand European Customs Brok	https://finance.yahoo.com/news/maersk-aims-
472b-87d6-a73d097f64e7	AMKBY	Shipping firm Maersk says second-quarter dema	https://finance.yahoo.com/news/shipping-firm

5. Project breakdown for collaborative work

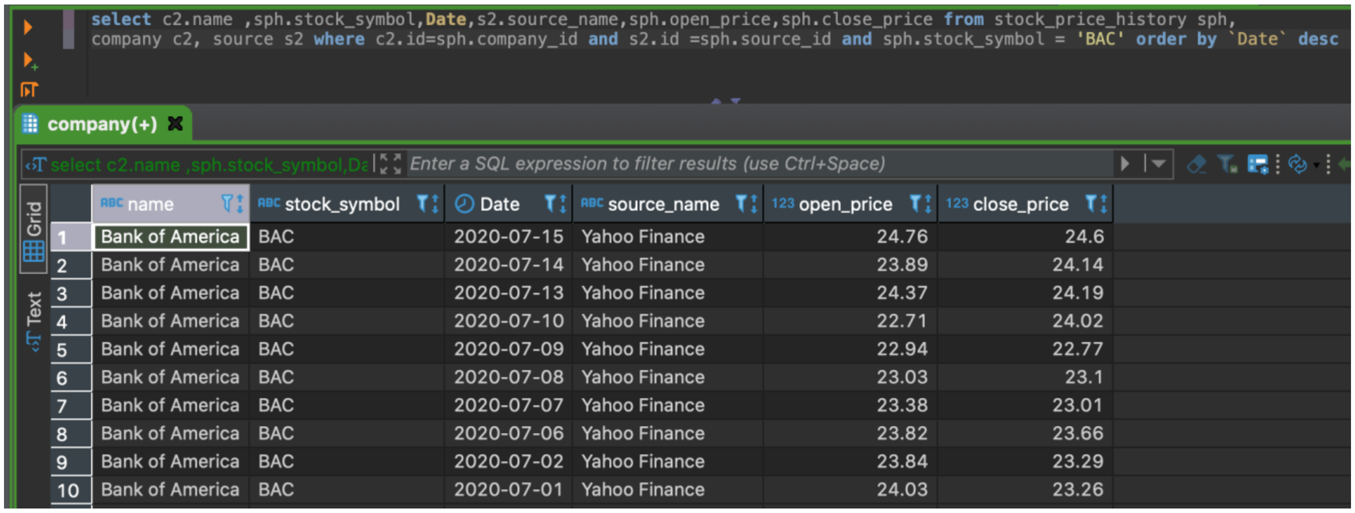
According to me, this project can be broken down into 4 parts.

1. Research work to help us as a team decide on what kind of data needs to be captured in order to assess company's performance.
2. Creating data models based on the research work done.
3. Scraping data from relevant web sources.
4. Building a standard ETL job to scrape data on a regular basis.

In this project, there will be lot of code that can be re used to scrape data, so a common library with all useful methods can be created and it can be shared among the team so that they can work easily and in a standard process.

6. Sample Query

Get the past 10 days open and close stock price for Bank of America company in descending order and sourced from Yahoo Finance



```sql
select c2.name ,sph.stock_symbol,Date,s2.source_name,sph.open_price,sph.close_price from stock_price_history sph,
company c2, source s2 where c2.id=sph.company_id and s2.id =sph.source_id and sph.stock_symbol = 'BAC' order by `Date` desc
```

	name	stock_symbol	Date	source_name	open_price	close_price
1	Bank of America	BAC	2020-07-15	Yahoo Finance	24.76	24.6
2	Bank of America	BAC	2020-07-14	Yahoo Finance	23.89	24.14
3	Bank of America	BAC	2020-07-13	Yahoo Finance	24.37	24.19
4	Bank of America	BAC	2020-07-10	Yahoo Finance	22.71	24.02
5	Bank of America	BAC	2020-07-09	Yahoo Finance	22.94	22.77
6	Bank of America	BAC	2020-07-08	Yahoo Finance	23.03	23.1
7	Bank of America	BAC	2020-07-07	Yahoo Finance	23.38	23.01
8	Bank of America	BAC	2020-07-06	Yahoo Finance	23.82	23.66
9	Bank of America	BAC	2020-07-02	Yahoo Finance	23.84	23.29
10	Bank of America	BAC	2020-07-01	Yahoo Finance	24.03	23.26