SI 206 Final Project Report

Group Name: The Best Group
By: Oliver Li, Emmaline Smith, David Hu
https://github.com/oliverli99/si206_finalproject

1) Goals for our project

Our overall goal for our project was to understand the effects that COVID-19 is having on society and investigate claims about how the spread of the disease will be mitigated with warmer temperatures.

2) Goals that were achieved

We successfully created the visualizations to gain a better understanding of the situation. Based upon the visualizations that we have created, we were able to successfully correlate the weather with the countries with the number of cases per country by continent. We also wanted to be able to understand the effects of COVID-19 on the economic markets. Thus, we took historical stock data and tried to understand how COVID-19 was possibly making a difference. We used stock tickers from several different industries in order to diversify and see if it made a difference. Although it is not easy to tell which individual stocks were affected because the time scale was small, if we look at VOO, or the S&P 500, we can tell that there was a huge dropoff on March 18th. This could be due to a variety of reasons. It does make sense as during this time the United States did not have enough resources to effectively deal with the problem (i.e., not enough medical masks and tests being manufactured). Lastly, although it was not in our original goal for our project, we also managed to understand the type of music that different countries were listening to and the most popular genres. We expected to see a lot more somber songs and genres given the time period we're living in, however, it seemed that rock and pop was still topping the charts.

3) Problems that we faced

The main problems that we faced were with finding relevant API's and manipulating the data in a way that made the two sources relate. We also had a lot of trouble deciding what made sense to investigate in the beginning. Another problem that we had while doing this project was the requirement that one of the API's must contain two separate tables and be joined by a primary key. This seemed like a strange requirement since most of the API's we were using have data that can all be stored in one table so we solved this by adding another API that had different types of data that could be categorized in two different tables. We finally solved this by finding an API by LastFM that contained data that had API endpoints that made sense to split into two tables. We ended up having two different tables, one for the top tracks and one for the top artists.

The date and country columns could both be considered the shared keys for the two tables.

4) Calculation Data

The calculation data is stored in the CSV files attached to the assignment. They consist of a few things. The first file 'tempvscasesbycovid.csv' shows the average temperature by country and the number of cases. the direct correlation between the two. The second output file that we created was 'avg_temp_by_country.csv' which contains the average temperature by each country across all the data that we have gathered. We created a similar output file, avg_new_cases.csv,' which contains the average number of *new* cases per day per country. For the stock data, we output the average stock price per ticker in a file called 'average_stock_price.csv.' For the tracks and artists API, there is an output file called 'numcases_vs_listeners_by_genre.csv.' This contains the number of cases and the number of listeners by genre for each country. There is also a csv named 'artistcount'.csv.' This contains the artist vs. the number of times that artist appears in the database.

5) The visualization that you created (i.e. screenshot or image file)

The visualizations will be attached into this zipped folder. Descriptions of them are as follows:

1) Average_stock_price: Fitted histogram with differing stock prices
2) Average_temp_num_cases: Linear regression plot of the average temperature and the number of cases of each country
3) Histogram_cases & Histogram_new_cases: Histograms of the total and average new number of cases
4) Numcases_listeners_by_genre: Plots the number of listeners by genre against the total number of cases for that country.
5) Stock_time_series: Plots time series of stocks
6) Top_artist_count: histogram of the counts of how many times an artist shows up
7) Weather_vs_Cases_pair_plot: Pair Plot of the temperature vs. the number of cases.
8) Weather_vs_Cases_Regressin: The number of cases vs. the temperature of each country by continent.

6) Instructions for running the code

The following are instructions for running the code.

1) Open the SIfinalproject.ipynb file using Jupyter notebook.
2) Hit **Shift + enter** in order to run each cell. Run the first cell.
3) Going to the next cell titled "Weather Data." If you want to add entries for today's weather for each country into the database, hit **Shit + enter.** You should

comment out the three lines of code if you don't want to append data to the current database when you run the cell.

4) Going to the next cell titled "COVID-19 Data." Hit **Shift + enter** if you would like to append the COVID-19 data**.** Similarly to the above cell, you want to comment the three lines of code as indicated by the comment in the code if you don't want to append COVID-19 data to the current database when you run the cell. *Be wary* that this may take a long time since the API has since gotten a lot of traffic.
5) Going to the next cell titled "Stock Data." Hit **Shift + enter**. An input prompt should show up, and you must enter a *valid* stock ticker. This will return data for the last 20 days for that stock ticker. This will only have to be done every 20 business days/days for that specific stock ticker.
6) Going to the next cell titled "Music Data." Run this once a day *only*. Hit **Shift + enter.** This will prompt the user to choose what list of 20 countries of data to add for that specific day.
7) Going to the next cell titled "Querying, Calculating, and Visualizations," again all you need to do is just run it by hitting **CTRL + enter.** This should re-generate all of the visualizations and output data for the current day if applicable.

7) Documentation for each function that you wrote. This includes the input and output for each function.

List of Classes: WeatherData, COVIDCases, Stocks, Calculate

WeatherData: contains all the functions from beginning to end to accessing the WeatherBit API data, manipulating the data, and storing it into the SQL database 'data.db.'

def __init__(self):

"""
Initializes each Weather Data object. Creates the city_urls attribute which contains all request urls for all capital cities around the world.
"""

def create_request_url_weather(self, city):

"""
Takes in a city, returns request URL
"""

def fit_weather_list(self, country_weather):

"""
Take in a dictionary 'country_weather'. Refits the names of countries to correct names.
Returns a dictionary with fitted list
"""

def get_weather_data(self):

"""

Gets weather data using the list of urls.

Returns a dictionary called country_weather

"""

def create_insert_weather_tuple(self, country_weather):

"""

Takes in a dictionary, returns a tuple of values with only 20 top most influential countries to insert into database

"""

def create_and_append_weather(self, insertweathertuple):

"""

Take in tuples, and append country/weather data to the country_weather table in the data database. No return value.

"""

def convert_to_fahrenheit(dictionary):

"""

Convert a dictionary from celsius to fahrenheit by modifying the values. Returns dictionary with all values in fahrenheit.

"""

COVIDCases: contains all the functions from beginning to end to accessing the COVID-19 API data, manipulating the data, and storing it into the SQL database 'data.db.'

def fit_list(self, dictionary):

"""

Takes in a dictionary, returns a dictionary with corrected key names.

"""

def get_country_cases_dict(self):

"""

Returns a dictionary for 73 most influential countries and corresponding number of cases.

"""

def get_covid_info_by_country_today(self, country):

"""

Take in country, returns COVID data for that country.

"""

def create_and_insert_cases_tuple(self):

"""

Creates/returns the tuple for 20 most influential countries to insert into the database.

"""

def create_and_append_cases(self, insertcasestuple):

"""

Take in tuple, append COVID-19 data to the cases table in the data database.
"""

Stocks: contains all the functions from beginning to end to accessing the Alpha Vantage (Stock) API data, manipulating the data, and storing it into the SQL database 'data.db.'

def __init__(self, tickerlist):
"""
Initializes Stock object. Creates past_20_days attribute, a list of the past 20 dates.
"""

def get_past_20_days(self):
"""
Return a list of the past 20 dates
"""

def get_last_20_days_stock_data(self):
"""
Uses the stock object attribute to create a request URL for the past 20 dates, return a list of tuples with that stock
for the past 20 days.
"""

def create_and_append_stocks(self, inserttuple):
"""
Take in a tuple, append the data inside the tuple into a stocks table.
"""

Music Data: Not a class but a cell with all the data collection and manipulation for get_top_tracks and get_top_artists.

def request_url_geo_top_artists(country):
"""
Pass in the country. Returns request url for geo top artists API endpoint.
"""

def request_url_geo_top_tracks(country):
"""
Pass in the country. Returns request url for geo top tracks API endpoint.
"""

def get_top_artists(request_url):
"""
Pass in a request url. Returns a list of tuples to prepare data for database and visualization.

"""

def get_top_tracks(request_url):

"""

Pass in a request url. Returns a list of tuples to prepare data for database and visualization.

"""

def create_and_append_song_data():

"""

Creates tracks & artists table if they don't exist. Append data to tracks and artists table. Returns nothing.

"""

Calculate: contains all the queries that perform the calculations upon all the tables in the SQL database 'data.db.' Also contains queries to create other visualizations.

def avg_temp_by_country(self):

"""

Calculates the average temperature by country. Returns a tuple that contains the query result.

"""

def historic_cases(self):

"""

Calculates the cases by date and country. Returns a tuple that contains the query result.

"""

def avg_new_cases_per_day_by_country(self):

"""

Calculates the average new cases by country. Returns a tuple that contains the query result.

"""

def join_calc(self):

"""

Calculates using a JOIN the average weather, total number of cases. Returns a tuple that contains the query result.

"""

def avg_stock_price(self):

"""

Calculates average stock price per stock. Returns a tuple that contains the query result.

"""

def stocks(self):

"""

Select all data from the stocks table.
"""

def check_length_cases(self):
"""
Checks the number of entries of the cases table.
"""
def check_length_stocks(self):
"""
Checks the number of entries of the stocks table.
"""
def check_length_weather(self):
"""
Checks the number of entries of the country_weather table.
"""

def artist_calculate(self):
"""
Returns the artist, and number of times that artist shows up
"""

def track_calculate(self):
"""
Returns country, cases, # listeners, artist, and genre
"""

8) Resource Documentation

Date	**Issue Description**	**Location of Resource**	**Result (did it solve the issue?)**
3/24/2020	Weather Data API	WeatherBit	Yes, it provided the necessary weather data for each city.
3/24/2020	COVID-19 Data API	RapidAPI	Yes, it provided the necessary COVID-19 data by country.

3/24/2020	Calling a class function inside of __init__	https://stackoverflow.com/questions/12646326/calling-a-class-function-inside-of-init	Yes, it confirmed what I was doing made sense.
3/24/2020	Stopping plots from plotting on top of each other. Saving plots into a pdf.	https://stackoverflow.com/questions/36018681/stop-seaborn-plotting-multiple-figures-on-top-of-one-another	Yes, it resolved the issue and we were able to output plots separately and save them to pdfs locally.
3/31/2020	Stock Data API	Alpha Vantage	Yes, it provided the necessary stock data given the ticker.
4/05/2020	Figuring out how to get the last 20 days	https://stackoverflow.com/questions/20573459/getting-the-date-of-7-days-ago-from-current-date-in-python/20573492	Yes, it helped create a function to grab the last 20 days/dates.
4/17/2020	Last.FM API	https://www.last.fm/api/	Yes, it helped us meet the requirement of having two tables with the same key.