Project Outbreak

Goal:

Examine any correlations between world health epidemics and on global stock markets (US, UK and Asia). Visualize any major market fluctuations which may have been due to worldwide illnesses and health scares.

1. Epidemic Data
    a. Outbreak of the Epidemic based on Year, Month
    b. Number of cases reported
    c. Impact on state (if US) or country (non US)
2. Country GeoJson
    a. Geodata data package provides geojson polygons for all the world's countries.
3. Stock Data (US/UK)
    a. Time Series chart of Major Stock Market Indexes over time
    b. Major Market indexes… U.S. (Dow Jones Industrial, S&P500, Nasdaq), U.K. (FTSE100), Japan (Nikkei 225), Hong Kong (Heng Seng)
    c. Open, High, Low, Close and Volume

Datasets :

Zika:https://www.kaggle.com/cdc/zika-virus-epidemic/version/1#
Ebola:https://www.kaggle.com/imdevskp/ebola-outbreak-20142016-complete-dataset
SARS:https://www.kaggle.com/imdevskp/sars-outbreak-2003-complete-dataset#sars_2003_complete_dataset_clean.csv
Cholera(Yamen):https://www.kaggle.com/tentotheminus9/yemen-data#Yemen%20Cholera%20Outbreak%20Epidemiology%20Data%20-%20Data_Country_Level.csv
Country GeoJson: https://github.com/datasets/geo-countries/blob/master/data/countries.geojson
COVID19 vs SARS vs MERS vs EBOLA vs H1N1
https://www.kaggle.com/imdevskp/covid19-vs-sars-vs-mers-vs-ebola-vs-h1n1/comments#763757

New Libraries: (Planned to use, not used currently)

Fusion.js : https://www.fusioncharts.com/fusiontime/examples/update-chart-using-api-methods
Chart.js: https://www.chartjs.org/

Market Indexes (add API Keys):

Time Series stock market information to be sourced from Alpha Vantage. A provider of free API's for real-time and historical stock market data.

Dow Jones Industrial (U.S.)
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=DJI&outputsize=full&apikey=

Nikkei 225 (Japan)
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=^N225&outputsize=full&apikey=

FTSE100 (U.K)
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=^FTSE&outputsize=full&apikey=

S&P 500 (U.S)
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=^GSPC&outputsize=full&apikey=

Hang Seng (Hong Kong)
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=^HSI&outputsize=full&apikey=

GitHub Link:

https://github.com/maheshdivan/project-outbreak

Instruction for local data load:

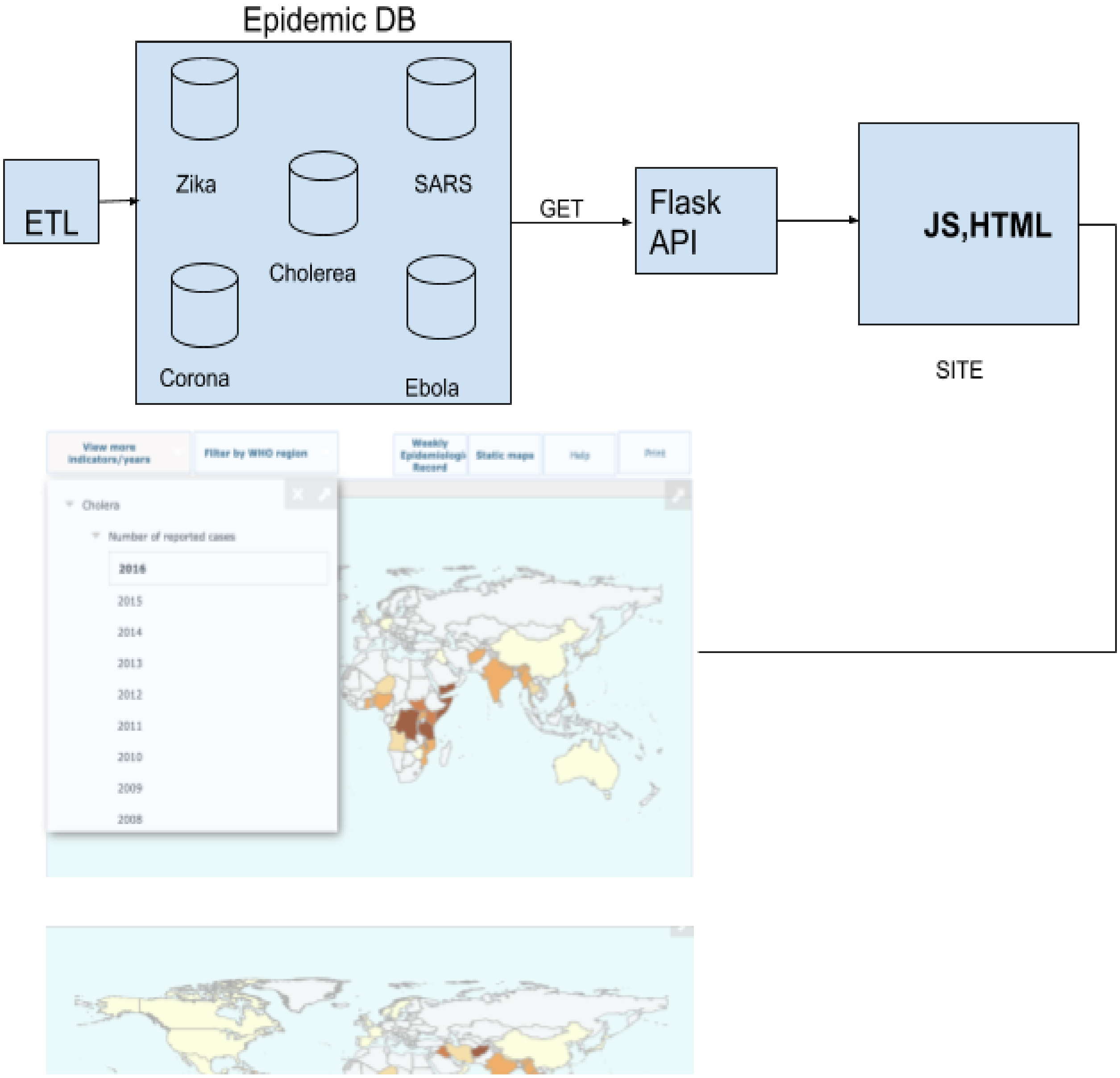
Clone the repo: https://github.com/maheshdivan/project-outbreak

To perform local data load:

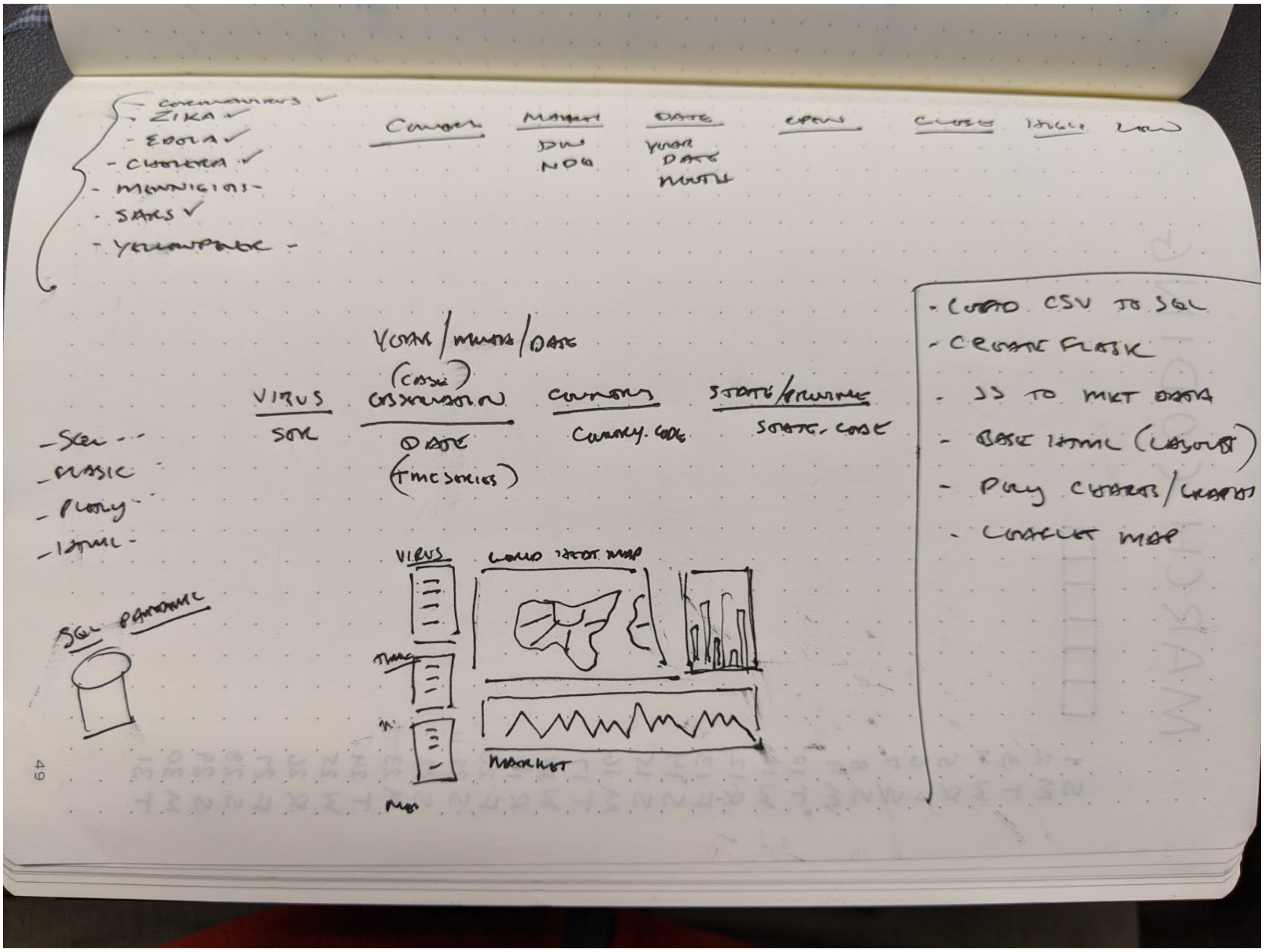
1. Open pgadmin and create one database
    a. Marketing_db
2. Multiple files for ETL:
    a. /ETL/ETL-Ebola_v3.ipynb
    b. /ETL/market.ipynb
3. Openl files in Jupyter Notebook and edit to account for local data file source and database connection string:
    a. epidemic_zika='/path/to/cdc_zika.csv'
    b. market_data='path/to/markets.csv'

    c. Rds_connection_string = "pgadminID:pgadminPWD@localhost:5432/marketing_db"
4. Open /API/Market/app.py and edit to account for local connection to local postgres DB
    a. conn = psycopg2.connect(host='localhost',user='postgres',password='postgres',dbname='marketing_db')
5. Run “python app.py” for local Flask server. All visualizations can pull data from`http://127.0.0.1:5000`;

Technical Architecture


Epidemic DB
ETL
Zika
SARS
Cholerea
Corona
Ebola
GET
Flask API
JS,HTML
SITE
View more indicators/years
Filter by WHO region
Weekly Epidemiological Record
Static maps
Help
Print
Cholera
Number of reported cases
2016
2015
2014
2013
2012
2011
2010
2009
2008

Rough Sketch:



- Coronavirus ✓
- Zika ✓
- Ebola ✓
- Cholera ✓
- Meningitis -
- SARS ✓
- Yellow Fever -

Company	Market	Date	Open	Close	High	Low
	DJI	Year				
	NDQ	Date				
		Month				

- SQL -
- Flask -
- Plotly -
- HTML -

Year / Month / Date

Virus	(Case) Observation	Country	State/Province
SQL	Date (Time series)	Country. Code	State. Code

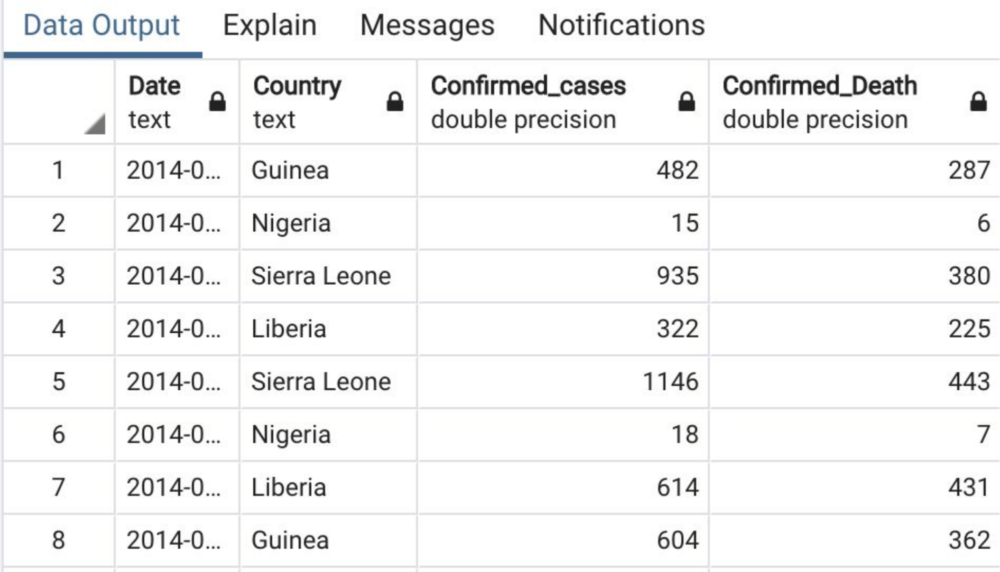
SQL Database

- Convert CSV to SQL
- Create Flask
- JS to MKT data
- Basic HTML (Layout)
- Plotly Charts/Graphs
- Leaflet Map

Technical Design:

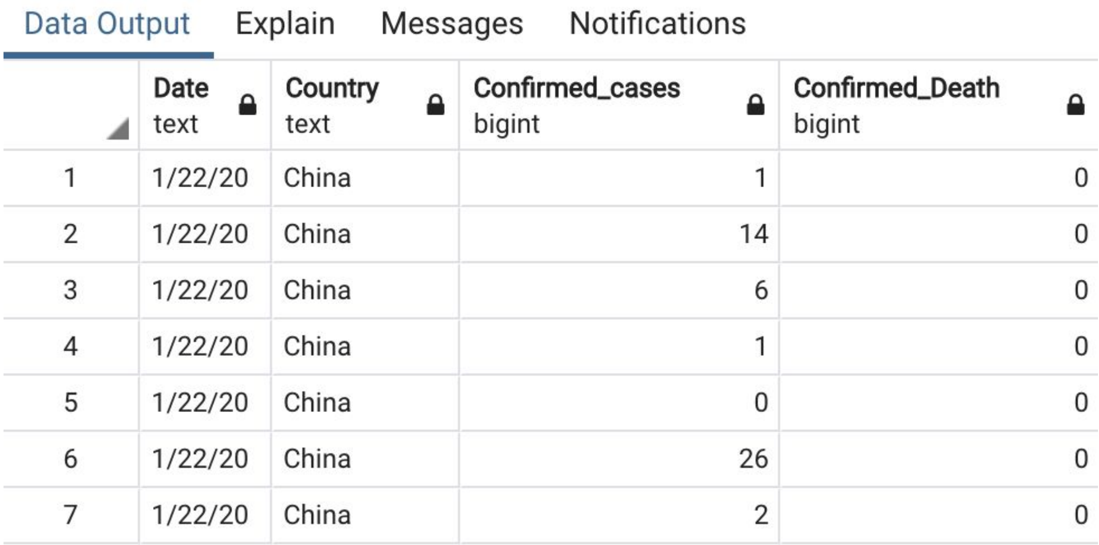
Perform Extract, Transform and Load of the Data into PGAdmin 4.

Epidemic : Ebola


Data Output
Explain
Messages
Notifications

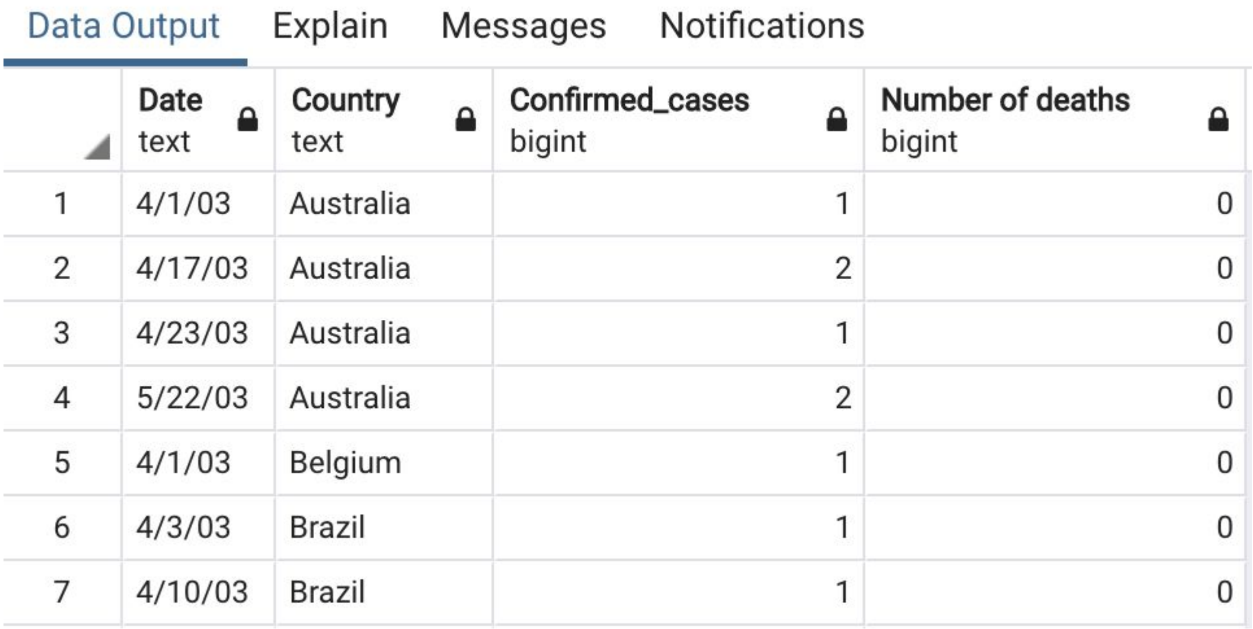
	Date text	Country text	Confirmed_cases double precision	Confirmed_Death double precision
1	2014-0...	Guinea	482	287
2	2014-0...	Nigeria	15	6
3	2014-0...	Sierra Leone	935	380
4	2014-0...	Liberia	322	225
5	2014-0...	Sierra Leone	1146	443
6	2014-0...	Nigeria	18	7
7	2014-0...	Liberia	614	431
8	2014-0...	Guinea	604	362

Corona Table:


Data Output
Explain
Messages
Notifications

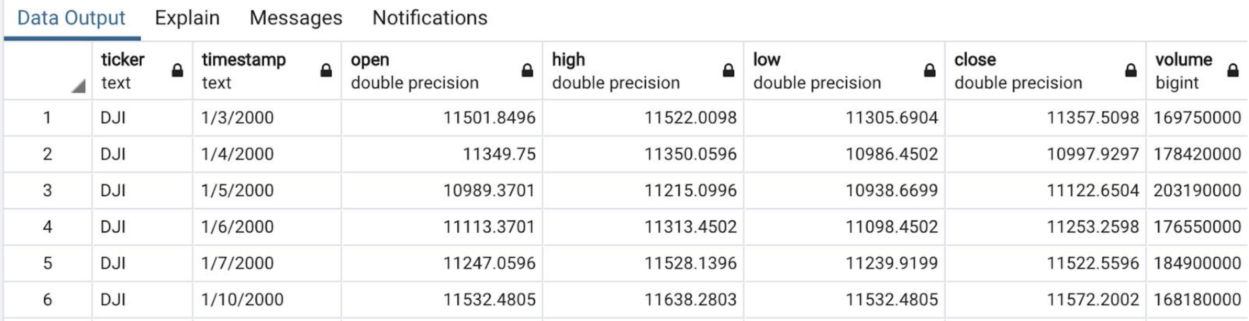
	Date text	Country text	Confirmed_cases bigint	Confirmed_Death bigint
1	1/22/20	China	1	0
2	1/22/20	China	14	0
3	1/22/20	China	6	0
4	1/22/20	China	1	0
5	1/22/20	China	0	0
6	1/22/20	China	26	0
7	1/22/20	China	2	0

SARS Table:


Data Output
Explain
Messages
Notifications

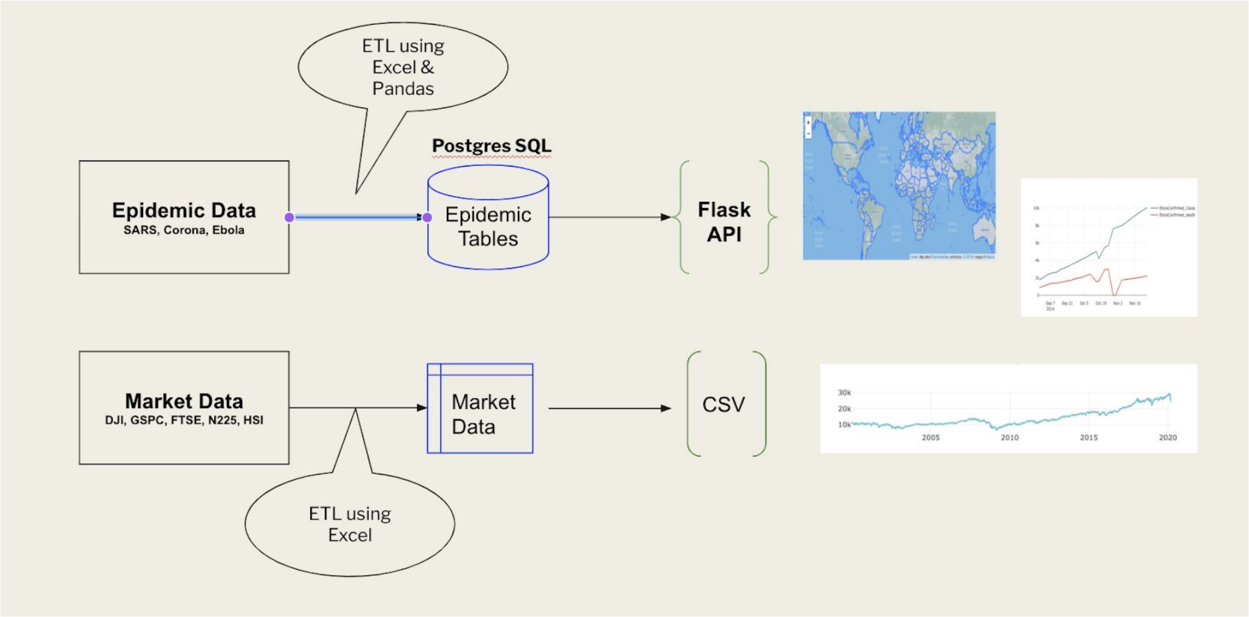
	Date text	Country text	Confirmed_cases bigint	Number of deaths bigint
1	4/1/03	Australia	1	0
2	4/17/03	Australia	2	0
3	4/23/03	Australia	1	0
4	5/22/03	Australia	2	0
5	4/1/03	Belgium	1	0
6	4/3/03	Brazil	1	0
7	4/10/03	Brazil	1	0

Market Data:


Data Output
Explain
Messages
Notifications

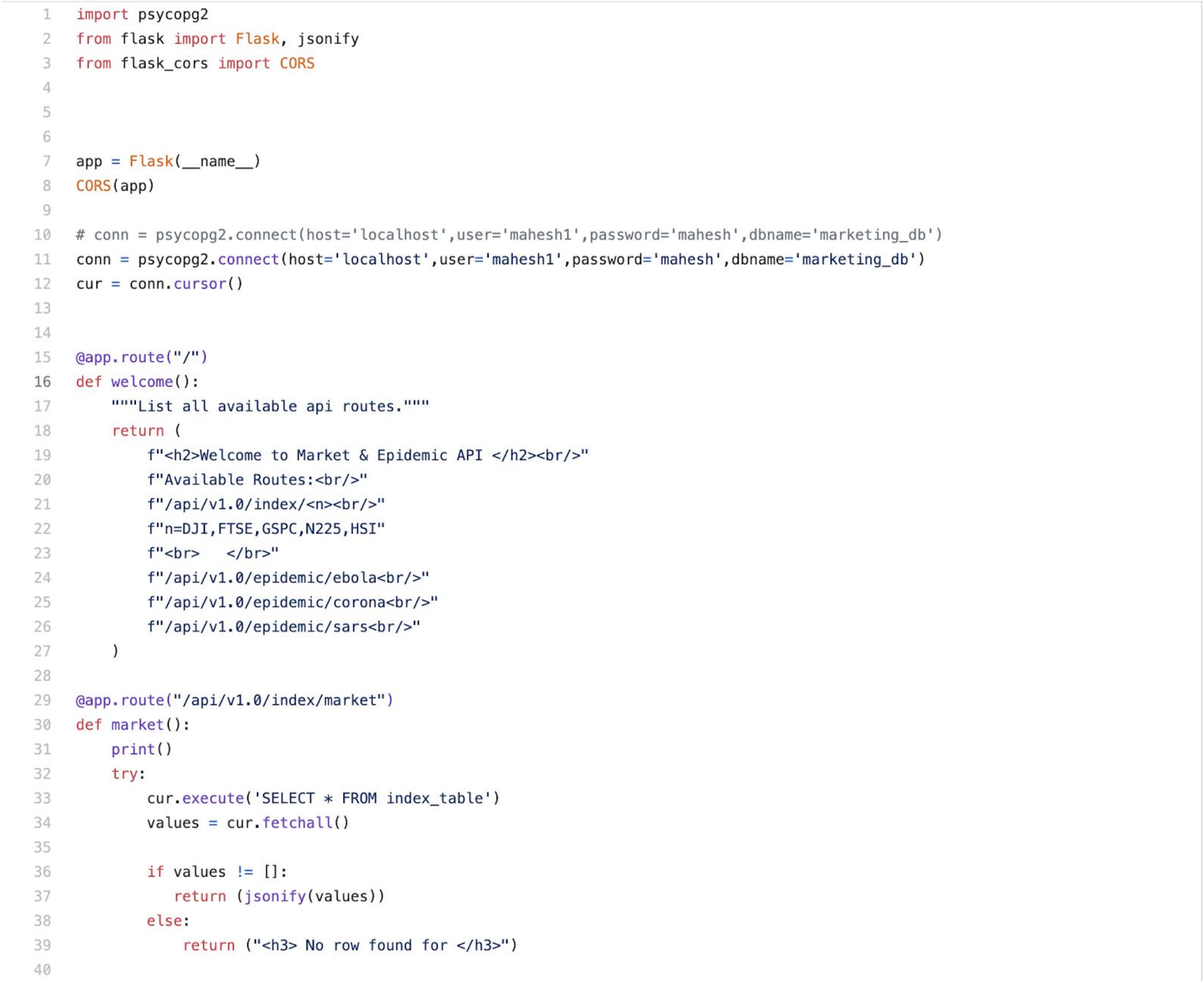
	ticker text	timestamp text	open double precision	high double precision	low double precision	close double precision	volume bigint
1	DJI	1/3/2000	11501.8496	11522.0098	11305.6904	11357.5098	169750000
2	DJI	1/4/2000	11349.75	11350.0596	10986.4502	10997.9297	178420000
3	DJI	1/5/2000	10989.3701	11215.0996	10938.6699	11122.6504	203190000
4	DJI	1/6/2000	11113.3701	11313.4502	11098.4502	11253.2598	176550000
5	DJI	1/7/2000	11247.0596	11528.1396	11239.9199	11522.5596	184900000
6	DJI	1/10/2000	11532.4805	11638.2803	11532.4805	11572.2002	168180000

Data Munging:


ETL using Excel & Pandas
Postgres SQL
Epidemic Data
SARS, Corona, Ebola
Epidemic Tables
Flask API
Market Data
DJI, GSPC, FTSE, N225, HSI
Market Data
CSV
ETL using Excel
30k
20k
10k
2005
2010
2015
2020

Flask API:

project-outbreak/API/Market/

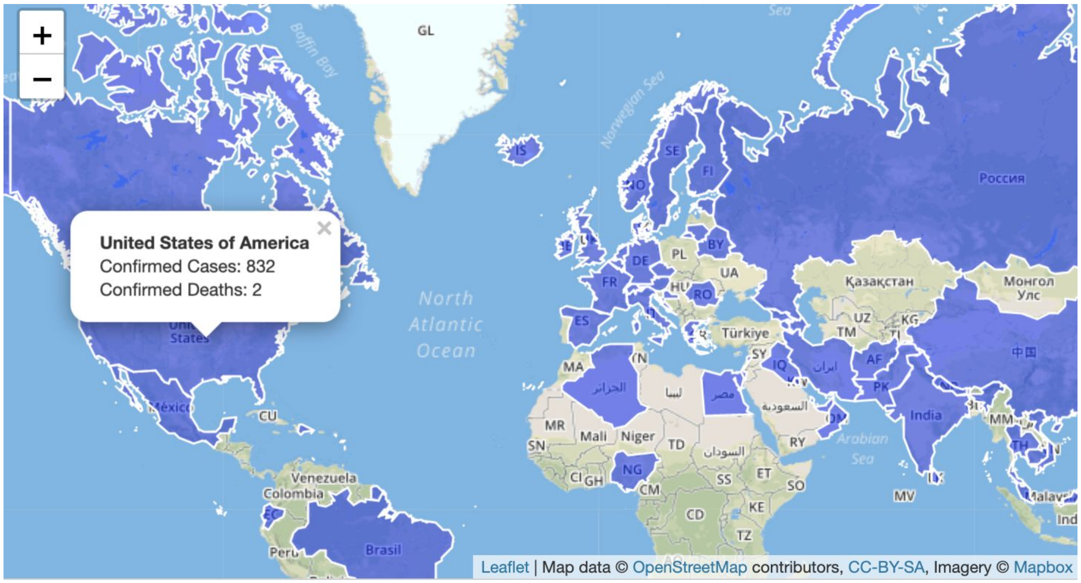


```python
import psycopg2
from flask import Flask, jsonify
from flask_cors import CORS



app = Flask(__name__)
CORS(app)

# conn = psycopg2.connect(host='localhost',user='mahesh1',password='mahesh',dbname='marketing_db')
conn = psycopg2.connect(host='localhost',user='mahesh1',password='mahesh',dbname='marketing_db')
cur = conn.cursor()


@app.route("/")
def welcome():
    """List all available api routes."""
    return (
        f"<h2>Welcome to Market & Epidemic API </h2><br/>"
        f"Available Routes:<br/>"
        f"/api/v1.0/index/<n><br/>"
        f"n=DJI,FTSE,GSPC,N225,HSI"
        f"<br>   </br>"
        f"/api/v1.0/epidemic/ebola<br/>"
        f"/api/v1.0/epidemic/corona<br/>"
        f"/api/v1.0/epidemic/sars<br/>"
    )

@app.route("/api/v1.0/index/market")
def market():
    print()
    try:
        cur.execute('SELECT * FROM index_table')
        values = cur.fetchall()

        if values != []:
           return (jsonify(values))
        else:
            return ("<h3> No row found for </h3>")

```

Data Visualizations

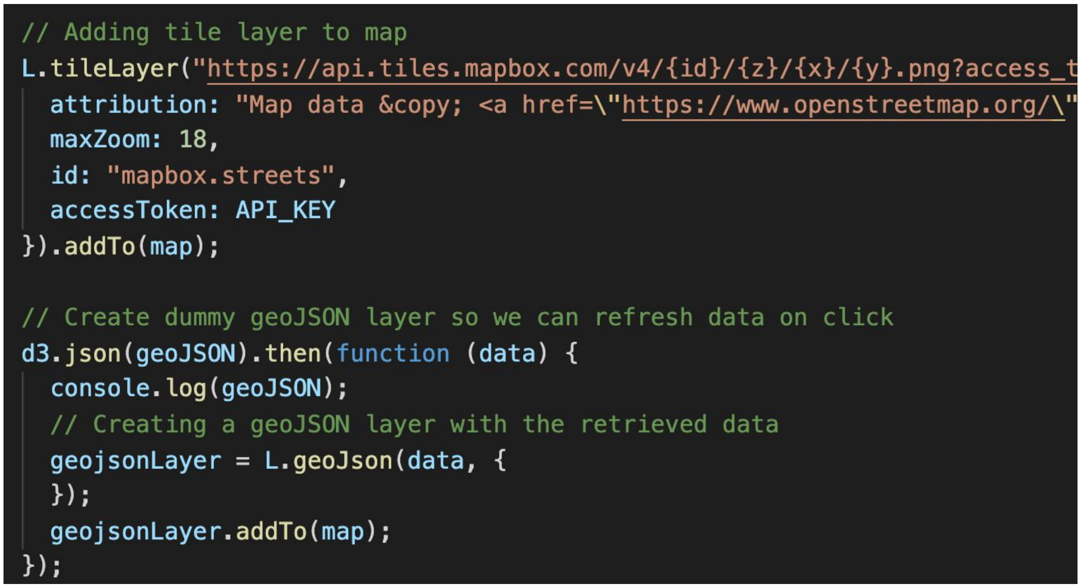
project-outbreak/static/js/

Leaflet

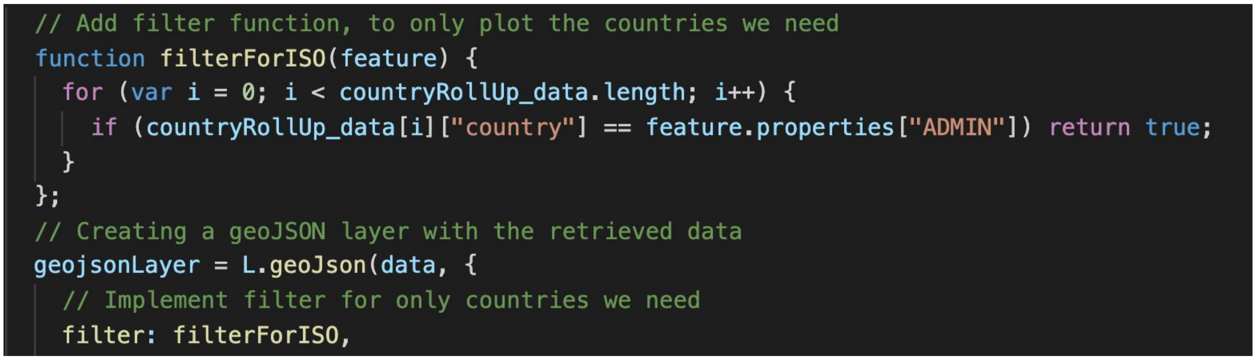

+
−
GL
Baffin Bay
Norwegian Sea
IS
SE
FI
NO
Россия
BY
PL
DE
UA
FR
HU
RO
ES
Қазақстан
Монгол Улс
UZ
KG
TM
Türkiye
SY
IQ
ایران
AF
PK
中国
India
North Atlantic Ocean
MA
الجزائر
ليبيا
مصر
السعودية
MR
Mali
Niger
TD
السودان
SN
GH
NG
CM
ET
SS
SO
CD
KE
TZ
RY
Arabian Sea
MV
MM
TH
United States
México
CU
Venezuela
Colombia
Peru
Brasil

United States of America
Confirmed Cases: 832
Confirmed Deaths: 2

Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

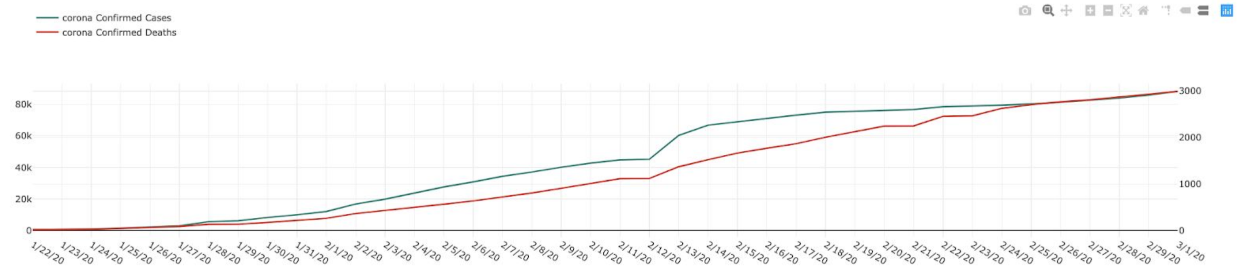


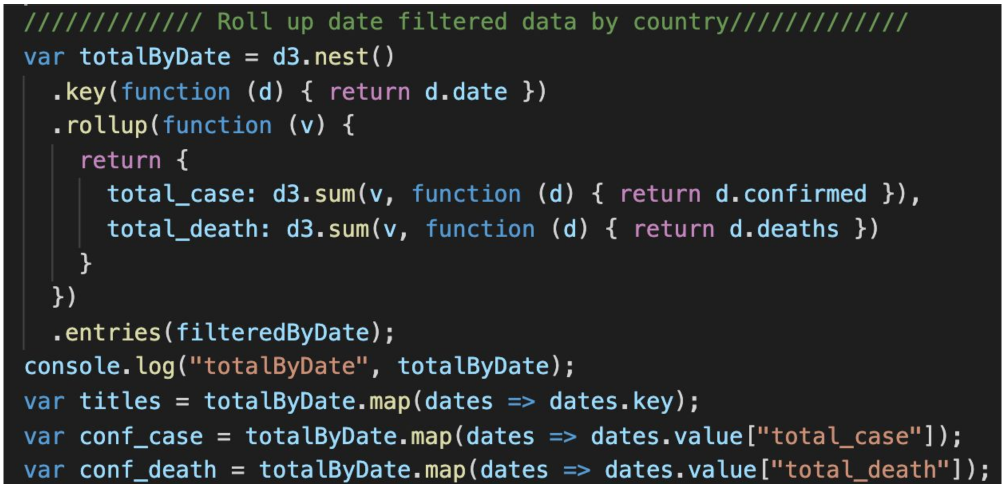
```
// Adding tile layer to map
L.tileLayer("https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_t
  attribution: "Map data &copy; <a href=\"https://www.openstreetmap.org/\"
  maxZoom: 18,
  id: "mapbox.streets",
  accessToken: API_KEY
}).addTo(map);

// Create dummy geoJSON layer so we can refresh data on click
d3.json(geoJSON).then(function (data) {
  console.log(geoJSON);
  // Creating a geoJSON layer with the retrieved data
  geojsonLayer = L.geoJson(data, {
  });
  geojsonLayer.addTo(map);
});
```



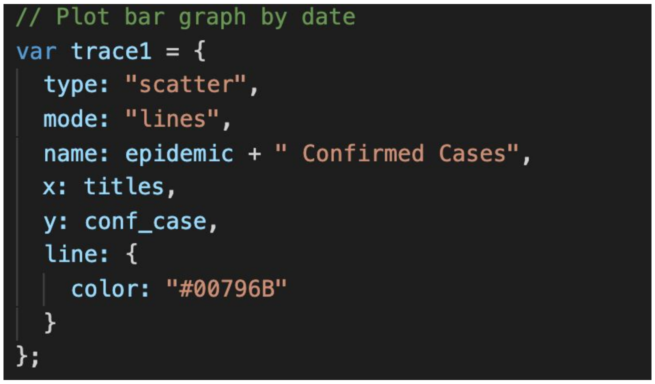
```
// Add filter function, to only plot the countries we need
function filterForISO(feature) {
  for (var i = 0; i < countryRollUp_data.length; i++) {
    if (countryRollUp_data[i]["country"] == feature.properties["ADMIN"]) return true;
  }
};
// Creating a geoJSON layer with the retrieved data
geojsonLayer = L.geoJson(data, {
  // Implement filter for only countries we need
  filter: filterForISO,
```

Plotly Epidemic Graph:



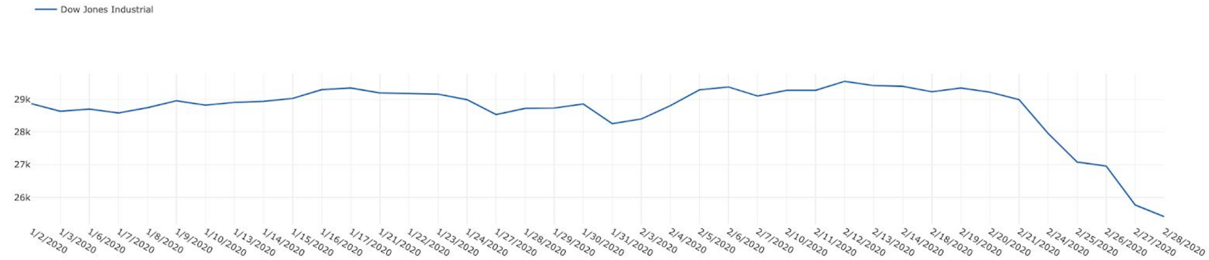


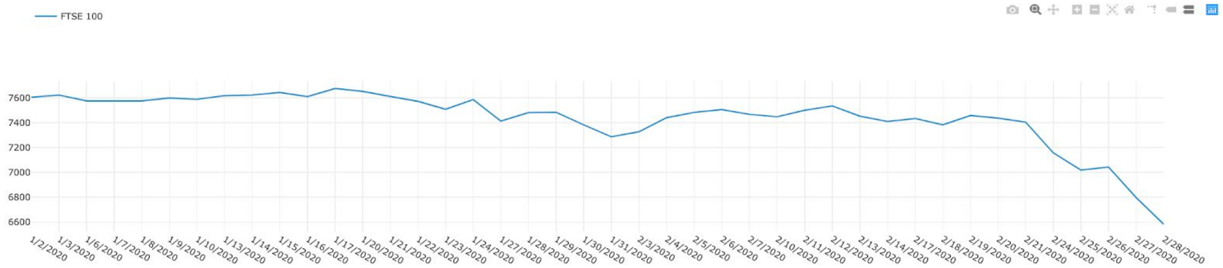
```
///////////// Roll up date filtered data by country/////////////
var totalByDate = d3.nest()
  .key(function (d) { return d.date })
  .rollup(function (v) {
    return {
      total_case: d3.sum(v, function (d) { return d.confirmed }),
      total_death: d3.sum(v, function (d) { return d.deaths })
    }
  })
  .entries(filteredByDate);
console.log("totalByDate", totalByDate);
var titles = totalByDate.map(dates => dates.key);
var conf_case = totalByDate.map(dates => dates.value["total_case"]);
var conf_death = totalByDate.map(dates => dates.value["total_death"]);
```

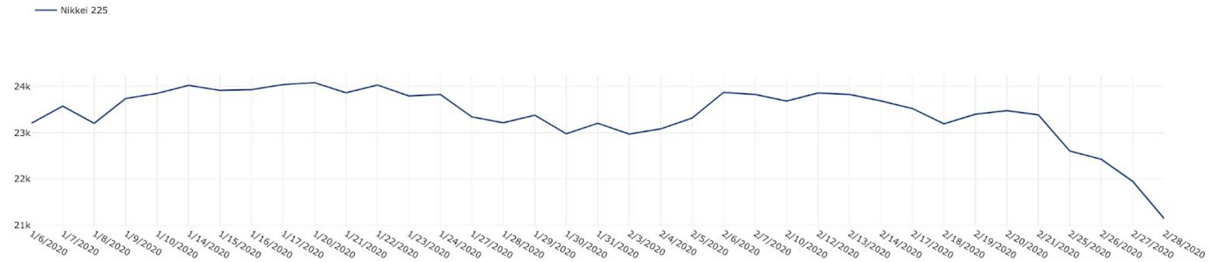


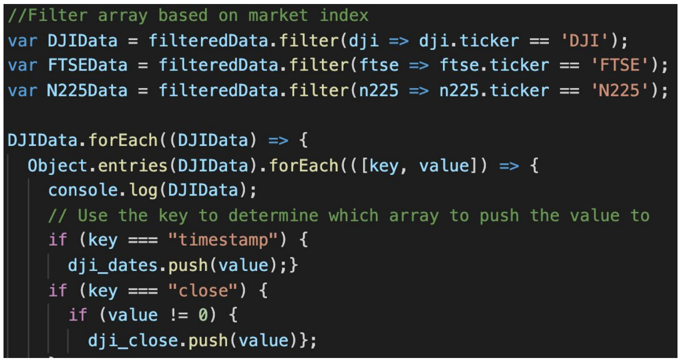
```
// Plot bar graph by date
var trace1 = {
  type: "scatter",
  mode: "lines",
  name: epidemic + " Confirmed Cases",
  x: titles,
  y: conf_case,
  line: {
    color: "#00796B"
  }
};
```

Plotly Market Graph:

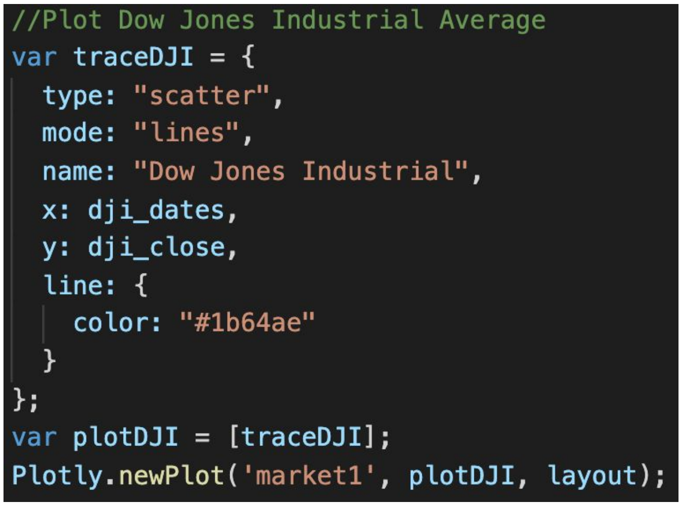








```
//Filter array based on market index
var DJIData = filteredData.filter(dji => dji.ticker == 'DJI');
var FTSEData = filteredData.filter(ftse => ftse.ticker == 'FTSE');
var N225Data = filteredData.filter(n225 => n225.ticker == 'N225');

DJIData.forEach((DJIData) => {
  Object.entries(DJIData).forEach(([key, value]) => {
    console.log(DJIData);
    // Use the key to determine which array to push the value to
    if (key === "timestamp") {
      dji_dates.push(value);}
    if (key === "close") {
      if (value != 0) {
        dji_close.push(value)};
```



```
//Plot Dow Jones Industrial Average
var traceDJI = {
  type: "scatter",
  mode: "lines",
  name: "Dow Jones Industrial",
  x: dji_dates,
  y: dji_close,
  line: {
    color: "#1b64ae"
  }
};
var plotDJI = [traceDJI];
Plotly.newPlot('market1', plotDJI, layout);
```

How run the application :

API:
Run Flask API by running python3 app.py under
https://github.com/maheshdivan/project-outbreak/tree/master/API/Market

Main APP:
Run python -m http.server from the index.html location