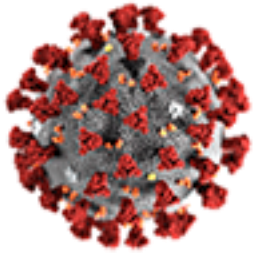
COVID19 TRACKER



Matt Mead
Allan Hunt
Emilio Bello

The Big Picture

Retrieve APIs
Convert to DF
Explore Data
Clean Data
Merge DFs
Add Calc Columns
Create DB connection
Create Postgres tables
Populate Tables

Create Database
Create Tables
Populate Tables
Define Primary Keys

Connect to Database
Mirror Tables
Run Flask Server
App Routes to Pages
App Routes to Data
Jsonify Data

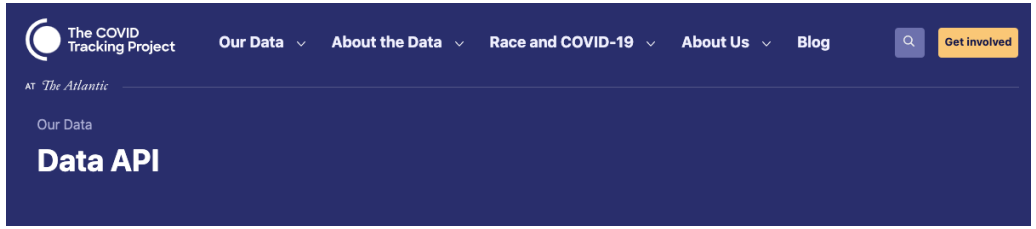
Call Data Routes
Store Data in Var
Plot
Populate HTML pages

Set Page Layout
Import JS Files
Import Libraries
Import API Key
Style

Datasets APIs

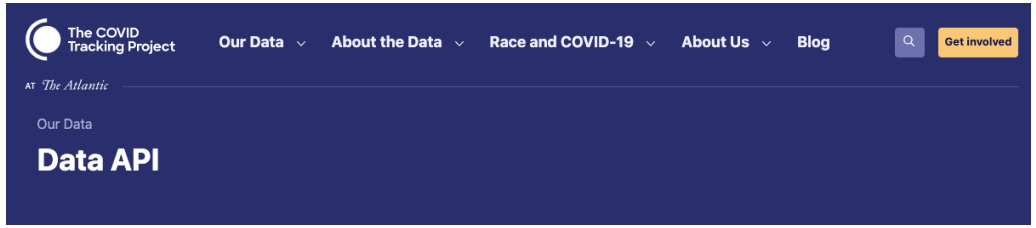
COVID information
historical by State

https://covidtracking.com/data/api



COVID information
historical US Totals

https://covidtracking.com/data/api



USA Population
by State

https://datausa.io/about/api/

Retrieve APIs
Convert to DF
Explore Data
Clean Data
Merge DFs
Add Calc Columns
Create DB connection
Create Postgres tables
Populate Tables

Importing and Inspecting the Data

In [1]:
```
# import depedencies
import requests as req
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import json
from config import password
from sqlalchemy import create_engine
```

In [2]:
```
# getting the APIs
url_covid = 'https://api.covidtracking.com/v1/states/daily.json'
url_state_population = 'https://datausa.io/api/data?drilldowns=State&measures=Population&year=latest'
```

In [3]:
```
# retrieving API and storing data in variables
covid_js = req.get(url_covid).json()
# usa_pop_js = req.get(url_usa_population).json()
state_pop_js = req.get(url_state_population).json()
print(json.dumps(state_pop_js['data'], indent=4, sort_keys=True))
```

```
[
```

In [4]:
```
# converting the data to a dataframe for data manipulation and cleaning
covid_all = pd.DataFrame(covid_js)
state_pop_df = pd.DataFrame(state_pop_js['data'])
display(covid_all.head(), state_pop_df.head())
```

	date	state	positive	probableCases	negative	pending	totalTestResultsSource	totalTestResults	hospitalizedCurrently	hospitalizedCumulative	...	po
0	20201022	AK	12877.0	NaN	535832.0	NaN	totalTestsViral	548709.0	41.0	NaN	...	5
1	20201022	AL	177064.0	22122.0	1125753.0	NaN	totalTestsViral	1280695.0	864.0	19448.0	...	13(
2	20201022	AR	102798.0	6506.0	1159242.0	NaN	totalTestsViral	1255534.0	603.0	6526.0	...	12(
3	20201022	AS	0.0	NaN	1616.0	NaN	totalTestsViral	1616.0	NaN	NaN	...	
4	20201022	AZ	234906.0	5341.0	1438997.0	NaN	totalTestsPeopleViral	1668562.0	848.0	20938.0	...	16

5 rows × 55 columns

	ID State	State	ID Year	Year	Population	Slug State
0	04000US30	Montana	2018	2018	1062305	montana
1	04000US01	Alabama	2018	2018	4887871	alabama
2	04000US04	Arizona	2018	2018	7171646	arizona
3	04000US05	Arkansas	2018	2018	3013825	arkansas
4	04000US06	California	2018	2018	39557045	california

Retrieve APIs
Convert to DF
Explore Data
Clean Data
Merge DFs
Add Calc Columns
Create DB connection
Create Postgres tables
Populate Tables

```
In [6]: state_pop_df['State'].unique()

Out[6]: array(['Montana', 'Alabama', 'Arizona', 'Arkansas', 'California',
               'Colorado', 'Connecticut', 'Delaware', 'District of Columbia',
               'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana',
               'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland',
               'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi',
               'Missouri', 'Alaska', 'Nebraska', 'Nevada', 'New Hampshire',
               'New Jersey', 'New Mexico', 'New York', 'North Carolina',
               'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania',
               'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee',
               'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington',
               'West Virginia', 'Wisconsin', 'Wyoming', 'Puerto Rico'],
              dtype=object)
```

```
deprecated_fields = ['checkTimeEt', 'commercialScore', 'dateChecked', 'dateModified',
                     'grade', 'hash', 'hospitalized', 'negativeIncrease',
                     'negativeRegularScore', 'negativeScore', 'posNeg', 'positiveScore',
                     'score', 'total', 'totalTestResultsSource']
non_required_fields = ['deathConfirmed', 'deathProbable', 'lastUpdateEt',
                       'totalTestsViral', 'positiveTestsViral', 'negativeTestsViral',
                       'positiveCasesViral', 'probableCases', 'negative', 'recovered',
                       'pending', 'totalTestEncountersViral', 'totalTestsPeopleViral',
                       'totalTestsAntibody', 'positiveTestsAntibody', 'negativeTestsAntibody',
                       'totalTestsPeopleAntibody', 'positiveTestsPeopleAntibody',
                       'negativeTestsPeopleAntibody', 'totalTestsPeopleAntigen',
                       'positiveTestsPeopleAntigen','totalTestsAntigen', 'positiveTestsAntigen',
                       'totalTestResultsIncrease', 'totalTestResults']
```

```
In [12]: # removing deprecated and non-required fields to obtain the filtered list
filtered_fields = []
for i in all_fields:
    if i not in deprecated_fields and i not in non_required_fields:
        filtered_fields.append(i)
filtered_fields
```

```
In [23]: # merging the covid_df with state_df
covid_wpop = covid_df.merge(state_df, how='inner', on='fips')
covid_wpop.head()
```

df.head()
df.shape
df.describe()
df.columns
df.info()

Retrieve APIs
Convert to DF
Explore Data
Clean Data
Merge DFs
Add Calc Columns
Create DB connection
Create Postgres tables
Populate Tables

Seven-day rolling average of new cases/deaths, by number of days since X daily cases/deaths first recorded

In [29]:
```
# indexing by state and sorting by state and date
covid_indexed = covid_wpop.set_index(['state']).sort_index().sort_values(['state', 'date'])
```

In [30]:
```
# keeping fields to be used
covid_rollavg = covid_indexed[['date', 'positive', 'positiveIncrease', 'death', 'deathIncrease']]
```

In [31]:
```
# inserting calculated column: 7 days rolling average of cases
covid_rollavg['positiveRollingAvg'] = covid_rollavg.groupby(level=0)['positiveIncrease'].rolling(window=7).mean().value
covid_rollavg
```

In [33]:
```
# new cases: looping to get the date at which the daily reported cases >= threshold for each state.
state_list = covid_rollavg.index.get_level_values('state').unique()
new_threshold = 10
dates_new = []
for i in state_list:
    new_df = covid_rollavg.loc[i]
    new_df = new_df.loc[new_df['positiveIncrease'] >= new_threshold]
    date = new_df.iloc[0].date
    dates_new.append(date)
dates_new
```

```
 Timestamp('2020-03-13 00:00:00'),
 Timestamp('2020-03-13 00:00:00'),
 Timestamp('2020-03-20 00:00:00'),
 Timestamp('2020-03-21 00:00:00'),
 Timestamp('2020-03-20 00:00:00'),
 Timestamp('2020-03-10 00:00:00'),
```

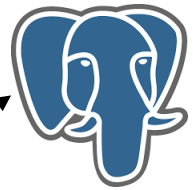
In [34]:
```
# zipping the states with the respective date
new_dict = dict(zip(state_list, dates_new))
new_dict
```

```
 'NV': Timestamp('2020-03-13 00:00:00'),
 'NY': Timestamp('2020-03-05 00:00:00'),
 'OH': Timestamp('2020-03-15 00:00:00'),
 'OK': Timestamp('2020-03-18 00:00:00'),
```

In [42]:
```
# populating new_datelapse with the date substraction for number of days since reaching the threshold
for x in covid_rollavg.index:
    covid_rollavg.iloc[x, 8] = (covid_rollavg.iloc[x, 1] - new_dict[covid_rollavg.iloc[x, 0]]).days
```

Retrieve APIs
Convert to DF
Explore Data
Clean Data
Merge DFs
Add Calc Columns
Create DB connection
Create Postgres tables
Populate Table

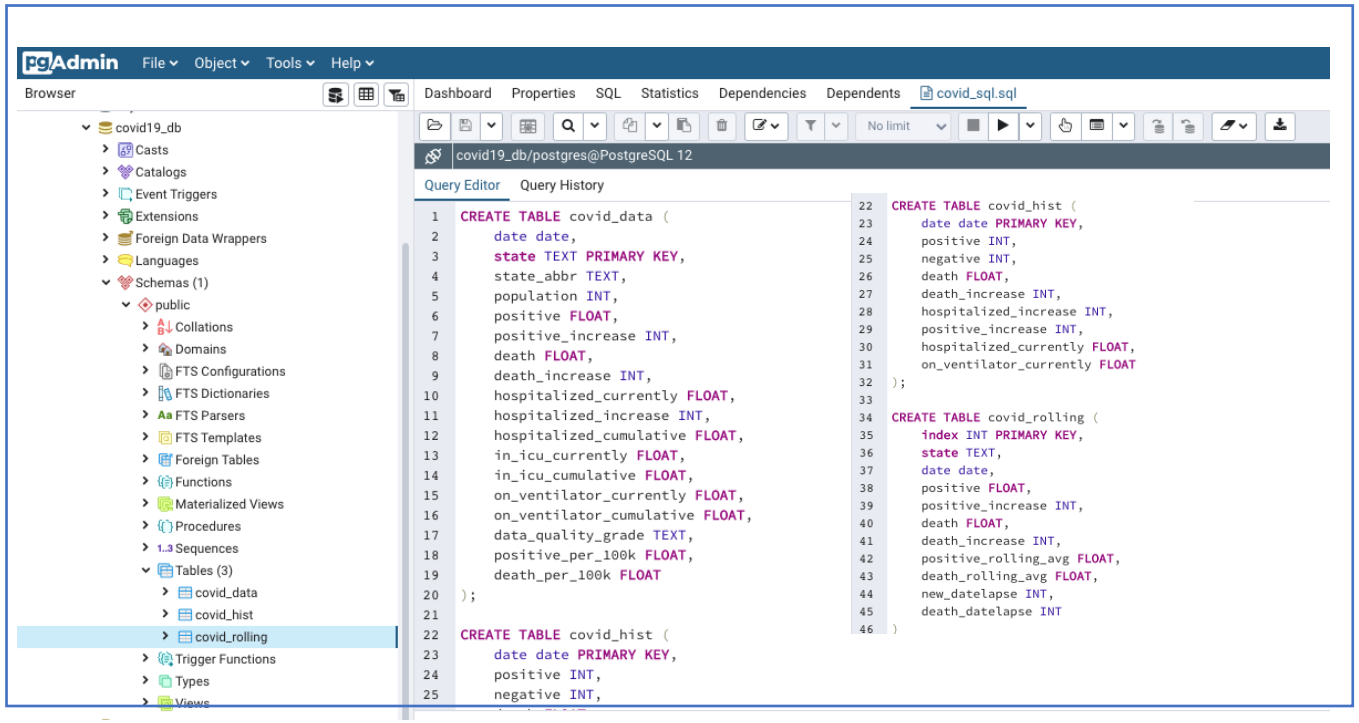
```
In [61]: # create the connection string
         rds_connection_string = "postgres:" + password + "@localhost:5432/covid19_db"

In [62]: # createing the engine connection with Postgres
         engine = create_engine(f'postgresql://{rds_connection_string}')
```

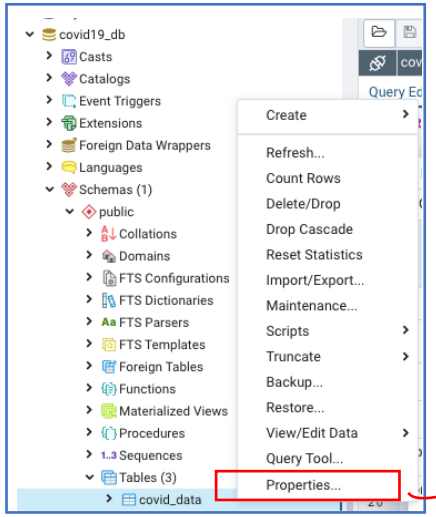


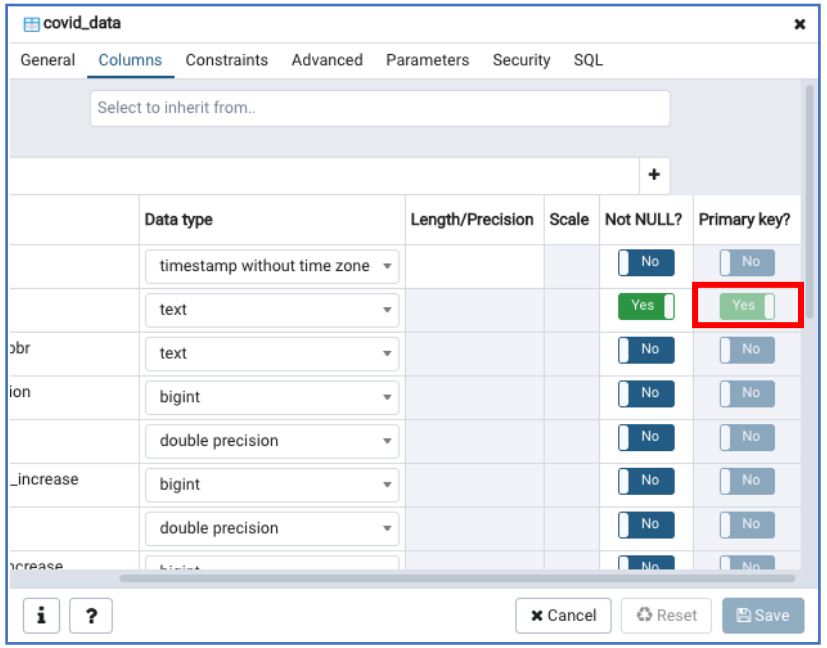
```
In [63]: # loading covid_rollingavg to SQL
         covid_rollavg.to_sql(name='covid_rolling', con=engine, if_exists='replace', index=True)
```

```
In [64]: # reading from SQL for validation
         covid_rolling = pd.read_sql_query('select * from covid_rolling', con=engine).head(15)
         covid_rolling
```

Create Database
Create Tables
Populate Tables
Define Primary Keys







Connect to Database
Mirror Tables
Run Flask Server
App Routes to Pages
App Routes to Data
Jsonify Data

```
COVID-19-Tracker — python ◂ python app.py — 80×24
(NewPythonData) Emilios-MBP:COVID-19-Tracker emiliobello$ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with fsevents reloader
 * Debugger is active!
 * Debugger PIN: 262-941-842
127.0.0.1 - - [23/Oct/2020 11:01:44] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2020 11:01:44] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /index.html HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/images/covid19.gif HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/js/config.js HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/js/app-numbers.js HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/js/app.js HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/js/app-cal.js HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2020 11:01:49] "GET /static/js/app-cal-positive.js HTTP/1.1" 304 -
```

app.py

```python
# Import the functions we need from flask
from flask import Flask
from flask import render_template
from flask import jsonify

# Import the functions we need from SQL Alchemy
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine

# Import username and password
from config import password

# Define the database connection parameters
username = 'postgres'  # Ideally this would come from config.py (or similar)
password = password  # Ideally this would come from config.py (or similar)
database_name = 'covid19_db' # Created in Week 9, Night 1, Exercise 08-Stu_CRUD
connection_string = f'postgresql://{username}:{password}@localhost:5432/{database_name}'

# Connect to the database
engine = create_engine(connection_string)
base = automap_base()
base.prepare(engine, reflect=True)

# Choose the table we wish to use
table_bystate = base.classes.covid_data
table_hist = base.classes.covid_hist
table = base.classes.covid_rolling

# Instantiate the Flask application. (Chocolate cake recipe.)
# This statement is required for Flask to do its job.
app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0 # Effectively disables page caching

```

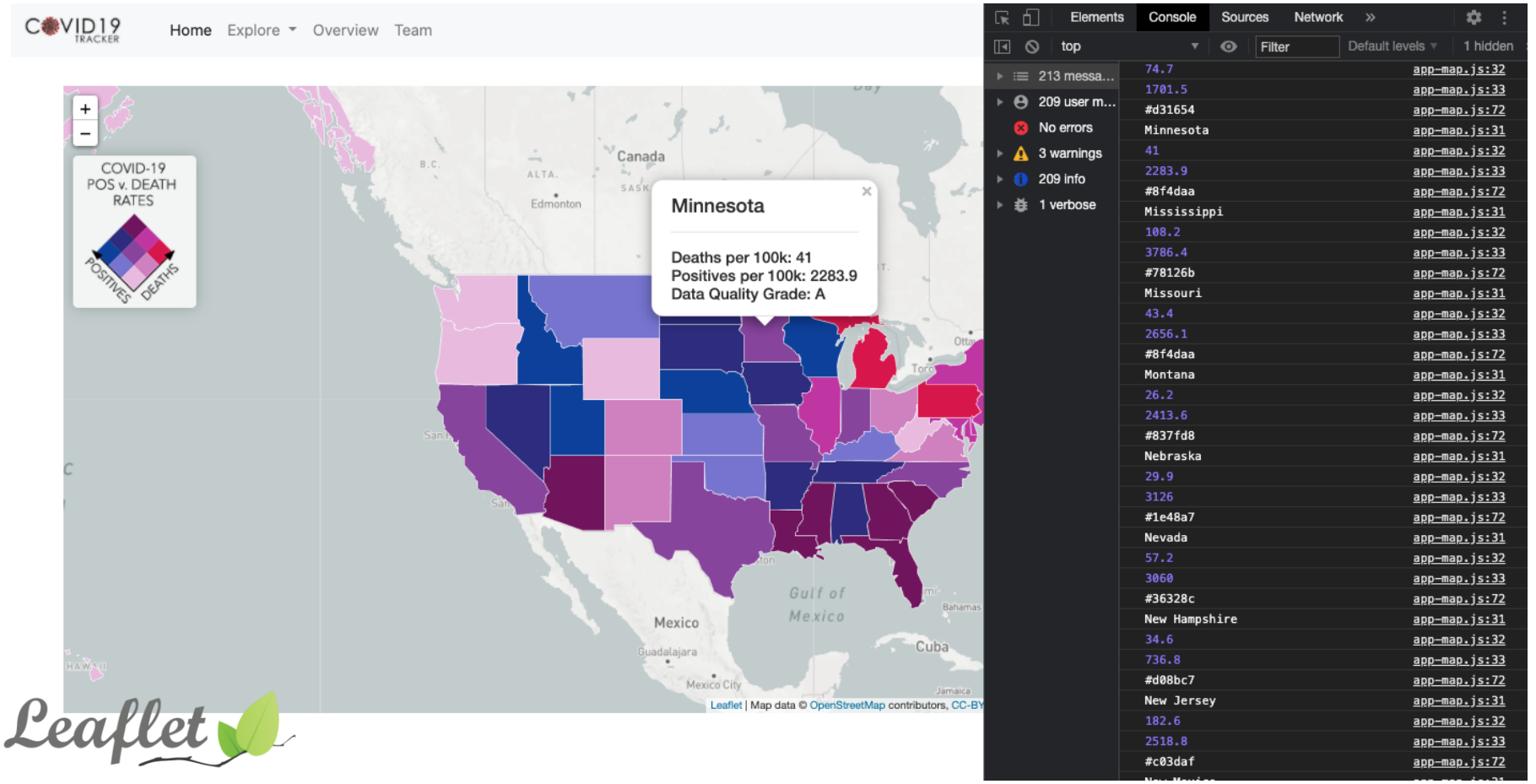
Connect to Database
Mirror Tables
Run Flask Server
App Routes to Pages
App Routes to Data
Jsonify Data

app.py

app.py > ...

```
89
90  # DEFINE APP ROUTES TO QUERY SPECIFIC DATA
91
92  @app.route("/covidhistory")
93  def QueryCovidHist():
94      ''' Querty the database for Covid History of US and return the results as JSON '''
95
96      session = Session(engine)
97      results = session.query(table_hist.date, table_hist.death_increase, table_hist.death, table_hist
98      session.close()
99
100     hist = []
101     for date, death_increase, death, positive, negative, hospitalized_currently, on_ventilator_curre
102         dict = {}
103         dict["date"] = date
104         dict["death_incr"] = death_increase
105         dict["all_death"] = death
106         dict["total_pos"] = positive
107         dict["total_neg"] = negative
108         dict["hospitalized_current"] = hospitalized_currently
109         dict["on_ventilator_currently"] = on_ventilator_currently
110         dict["hospitalized_increase"] = hospitalized_increase
111         dict["positive_increase"] = positive_increase
112         hist.append(dict)
113
114     return jsonify(hist)
115
```

Call Data Routes
Store Data in Var
Plot
Populate HTML pages



Set Page Layout
Import JS Files
Import Libraries
Import API Key
Style

```
# style.css ×
static > css > # style.css > .body
1   /* ---------- INDEX ---------- */
2
3   /* Classes */
4
5   .header-row {
6       padding-top: 10px;
7       text-align: center;
8   }
9
10  /* ---------- OVERVIEW ----------*/
11
12  /* Classes */
13  .list-header {
14      text-align: center;
15      color: #333333;
16      font-weight: lighter;
17  }
18
19  #title-resource {
20      text-align: center;
21      padding: 10px;
22      font-weight: lighter;
23  }
24
25  /* ---------- TEAM TAGS ---------- */
26
27  /* Classes */
28  .rounded-circle {
29      width: 200px;
30      height: 200px;
31  }
32
33  /* IDs */
34  #bio {
35      text-align: center;
36      padding: 20px;
37  }
```

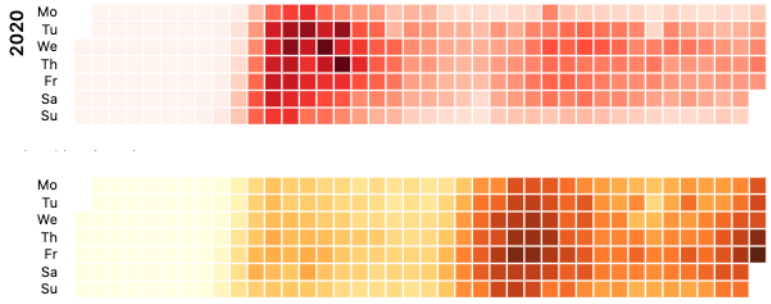
COVID19 TRACKER Home Explore Overview Team

COVID-19 Dashboard

Last Update

Thursday, October 22, 2020

Total Deaths	Total Positive	Total Hospitalized
215,761	8,449,231	41,485




```
Elements  Consol
<!DOCTYPE html>
<html lang="en">
▶ <head>…</head>
▼ <body data-new-gr-c-s-che
    <!-- NAVBAR START-->
  ▶ <nav class="navbar navbar-expand-lg navbar-light bg-
  light">…</nav>
    <!-- NAVBAR END -->
    <!-- COVID DASHBOARD HEADER START -->
  ▶ <div class="container">…</div>
    <!-- COVID DASHBOARD HEADER END -->
    <!-- NUMBERS START -->
  ▶ <div class="row">…</div>
    <!-- NUMBERS END -->
    <!-- CALENDARS SVG INPUT START -->
    <!-- DEATHS -->
  ▼ <div class="container center">
    ▼ <div class="row">
      ▼ <div class="col-12">
      ▶ <svg class="svgcal" id="svg">…</svg> == $0
        </div>
      </div>
    </div>
    <!-- POSITIVE -->
  ▶ <div class="container center">…</div>
    <!-- HOSPITALIZED -->
  ▶ <div class="container center">…</div>
    <!-- CALENDARS SVG INPUT END -->
    <!-- IMPORTS -->
    <script src="https://code.jquery.com/jquery-
    3.3.1.slim.min.js" integrity="sha384-q8i/
    X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6j
```

Set Page Layout
Import JS Files
Import Libraries
Import API Key
Style

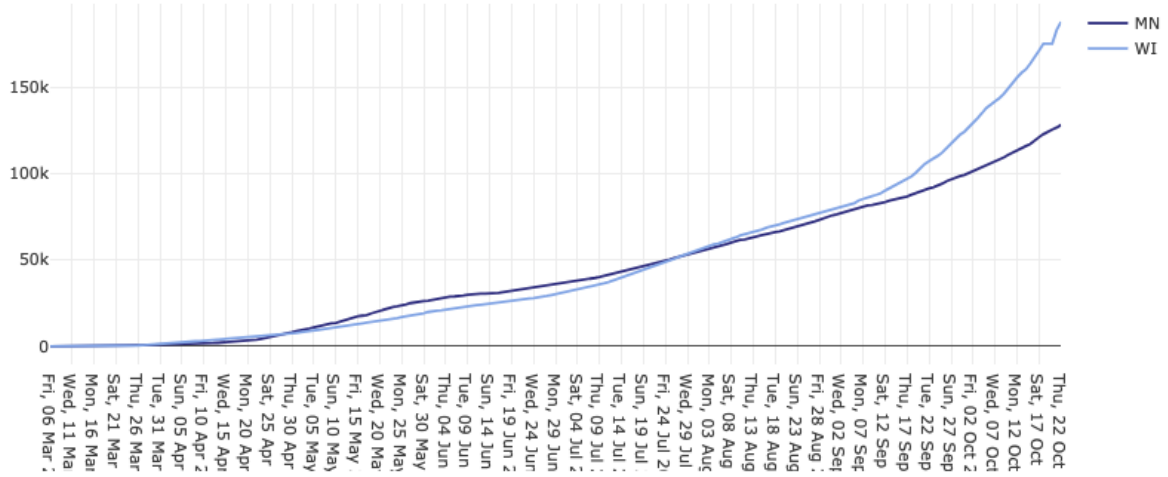
COVID19 TRACKER

Home Explore Overview Team

Select State 1: MN

Select State 2: WI

MN vs. WI cumulative new cases



MN vs. WI cumulative deaths

plotly