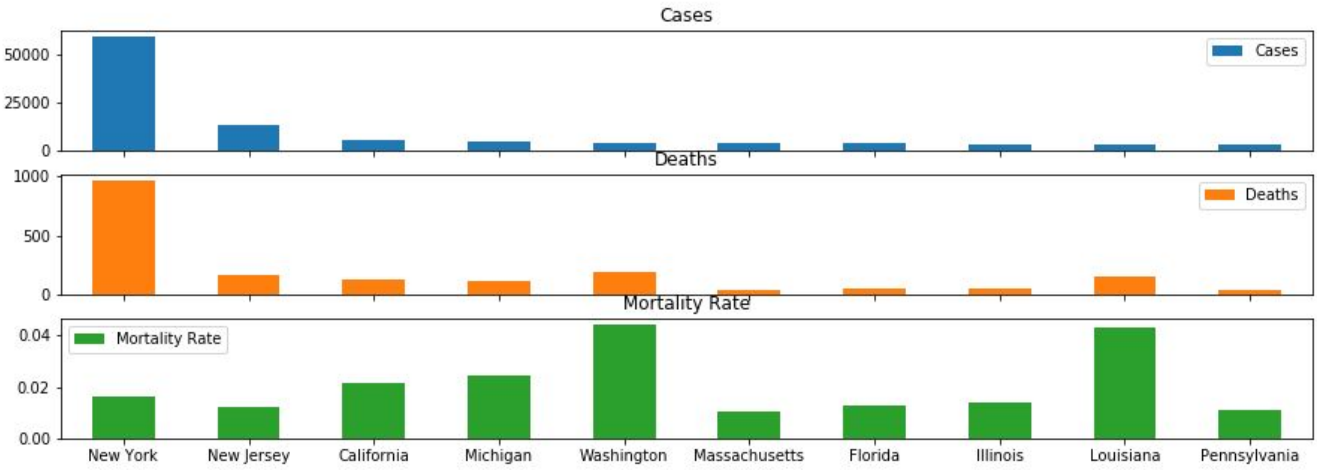
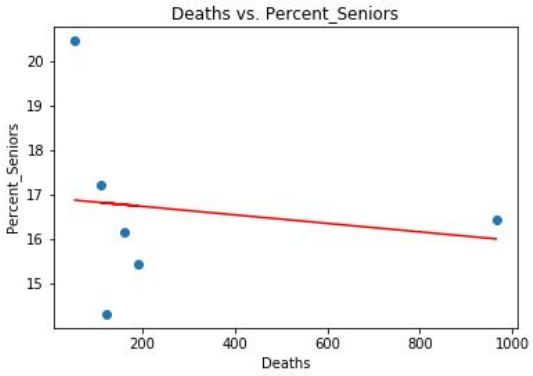
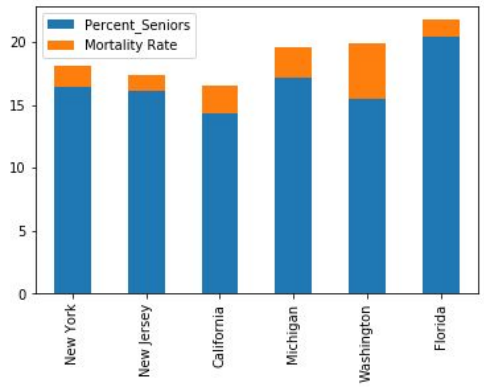
Top States COVID-19 Database

The purpose of the Database was to grab the data for deaths and cases by state including population, mortality rate, percent seniors as well as time series of cases and deaths over time. Then refine the data into a more usable form. The database can be used to analyze case and death counts by top states for COVID-19.



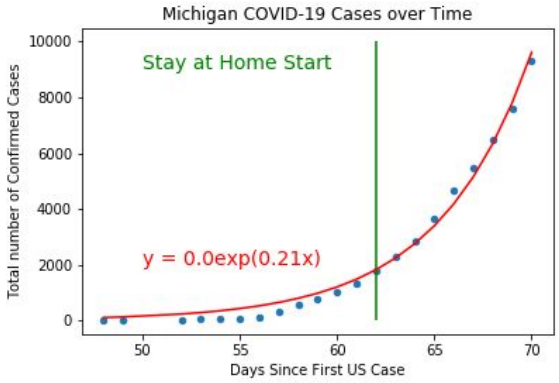
It allows one to see that although a relationship between the mortality rate of COVID-19 and the percentage of seniors in the population would seem to be strong the R-squared is only -0.42 which shows a weak relationship.

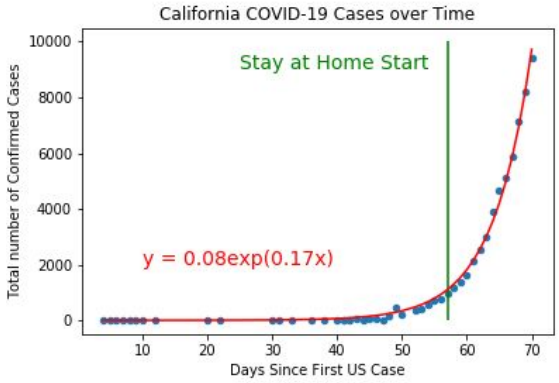


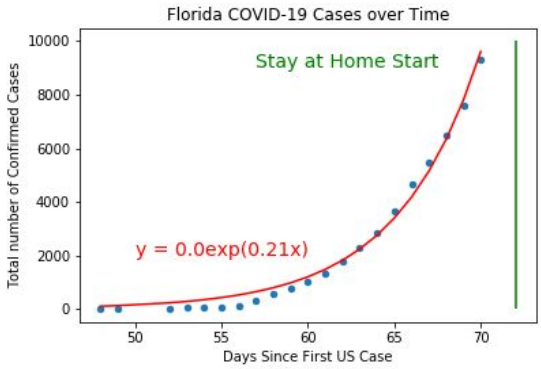


State	order	date
California	Stay at home	2020-03-19
New Jersey	Stay at home	2020-03-21
New York	Stay at home	2020-03-22
Washington	Stay at home	2020-03-23
Michigan	Stay at home	2020-03-24
Florida	Stay at home	2020-04-03

The data retrieved from the John Hopkins github after cleaning allowed for the creation of plots showing cases overtime as compared to an exponential growth rate while showing when shelter in place measures were put in.







States that put in stay at home measures closer to the first US COVID-19 case show lower growth rates in cases.

Extract from Data Sources

1. John Hopkins University Github
    a. https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_daily_reports/03-25-2020.csv
    b. Git clone and copy to local
2. Nytimes COVID-19 stay-at-home article
    a. https://www.nytimes.com/interactive/2020/us/coronavirus-stay-at-home-order.html
    b. Scrape from the link above using python, beautifulsoup, and pandas in a jupyter notebook.
    c. sah_measures = soup.find_all('div', class_="state-wrap") will retrieve the parent divs for all states
    d. for each_measure in sah_measures:

```
# scrape the article header
state = each_measure.find('h3').text
```

```
# scrape the article subheader
sah_order = each_measure.find('p', class_='l-order').text

# scrape the datetime
effective_date = each_measure.find('span', class_='l-date').text

# print article data
print('-----------------')
print(state)
print(sah_order)
print(effective_date)

# Dictionary to be inserted into MongoDB
state_measures = {
    'state': state,
    'sah_order': sah_order,
    'effective_date': effective_date,
}
```

e. This will give you a dictionary of the stay-at-home orders and their effective date by state

3. Census API
    a. 2018 census acs1 variables: https://api.census.gov/data/2018/acs/acs1/variables.json
    b. Use census wrapper to get overall population and population of male and female over 65 years per state
        i. Use tags: "B01003_001E", "B01001_020E", "B01001_021E", "B01001_022E", "B01001_023E", "B01001_024E", "B01001_025E" for males. These tags bin populations by ages of around 3 years. Similar set of tags for females.

Transform into Useable Data

1. John Hopkins cases and deaths by state overtime
    a. Import different depencies like datetime, pandas
    b. What we want to understand is the growth rates of coronavirus cases at the top five cities and John Hopkins's dataset has the daily report. **First** is having a function that self-generate all dates during March and create a for loop to add in a list (*Even though the first covid-19 case happened at WA back in Jan 22nd, 20, this dataset does not have latitude and longitude until March.

```
def daterange(start_date, end_date):
      for n in range(int ((end_date - start_date).days)):
            yield start_date + timedelta(n)
start_date = date(2020, 3, 1)
end_date = date(2020, 3, 22)
for single_date in daterange(start_date, end_date):
    dates_list.append(single_date.strftime("%m-%d-%Y")+'.csv')
```

c. Second, create another for loop to load all the csv into DataFrame(**include initial date) .

```
init_path='csse_covid_19_daily_reports/'+dates_list[0]
init_date_df=pd.read_csv(init_path)
for each_date in dates_list[1:]:
    loop_date_path='csse_covid_19_daily_reports/'+each_date
    new_date_df=pd.read_csv(loop_date_path)
    init_date_df=pd.concat([init_date_df, new_date_df], axis=0, sort=True)

covid_start_df=init_date_df.rename(columns={"Country/Region":"Country_Region",
```

d. Third, because the author of this dataset has changed the datetime format several times, we need to create a function to clean the datetime format so that we can use the date to plot the gmap later.

```
def func(x):
    print(x)
    if x[4] == '-':
        x = x[:10]
        y = datetime.strptime(x, '%Y-%m-%d').date()
    else:
        x=x[:10]
        y = datetime.strptime(x, '%m/%d/%y').date()
    return y
```

```
combine_df['date'] = combine_df['Last_Update'].apply(func)
combine_df['date']
```

e. Then filter down to only US and the top six states.

```
combine_df_US=combine_df.loc[combine_df['Country_Region'] == 'US']
NY=combine_df_US[combine_df_US['Province_State']=="New York"]
NJ=combine_df_US[combine_df_US['Province_State']=="New Jersey"]
CA=combine_df_US[combine_df_US['Province_State']=="California"]
MI=combine_df_US[combine_df_US['Province_State']=="Michigan"]
WA=combine_df_US[combine_df_US['Province_State']=="Washington"]
FL=combine_df_US[combine_df_US['Province_State']=="Florida"]
```

2. Nytimes Stay-at-Home orders by state

a. Create a dataframe out of the state measures dictionary
b. Split the text string for the state column of the dataframe by using
    i. Str.split on ‘ About ‘
    ii. That will give you 2 more columns of state and population
c. Split the text string for the date column of the dataframe by using
    i. Str.split on ‘ at ‘
    ii. That will give you
d. Create a new dataframe with only the state, order, and date columns
e. Create a list with the subset of states to analyze
f. Filter the dataframe so that it only contains rows where the state column is in the subset list.
    i. top_six_df = state_measures_df[state_measures_df['State'].isin(top_six)]
g. If there are duplicates in the dataframe filter them out now using iloc
h. Change the date column to a datetime format
    i. Add the year to the date since they only contain the month and day
        1. top_six_df['date'] = top_six_df['date'] + ' 2020'
    ii. Use pd.to_datetime() to change the format
        1. top_six_df['date'] =
        2. pd.to_datetime(top_six_df.date, format='%B %d %Y')
    iii. The data is now cleaned and ready to use.

3. Census senior citizen percentage of population
    a. Put results in two separate dataframes, one for male and one for female
    b. Filter states to look only at top 6 states affected by covid-19
    c. Since we are only looking at total population over 65, sum all columns with age related populations to get the overall population over 65 for each sex, make that into a separate column for each dataframe
    d. Join two sex tables on state
    e. Sum male and female total senior citizen population columns

<u>Load the final Database</u>

1. Gather all csv’s:
    a. Top State with Shelter in place data

```
top_six_df.to_csv('top_six.csv')
```

State	order	date
California	Stay at home	3/19/2020
Florida	Stay at home	4/3/2020
Michigan	Stay at home	3/24/2020
New Jerse	Stay at home	3/21/2020
New York	Stay at home	3/22/2020
Washingtc	Stay at home	3/23/2020

b. John Hopkins cases and deaths by state

State	Cases	Deaths
New York	59513	965
New Jerse	13386	161
California	5735	124
Michigan	4635	112
Washingtc	4319	191
Massachus	4257	44
Florida	4238	55
Illinois	3558	50
Louisiana	3540	152
Pennsylvar	3408	38

c. Census Data

	Name	Population	Percent_Seniors
16	California	39557045.0	14.326998
21	Florida	21299325.0	20.464423
25	New York	19542209.0	16.436550
26	New Jersey	8908520.0	16.145095
30	Michigan	9995915.0	17.211561
36	Washington	7535591.0	15.446526

2. Loaded the information into a relational database on PG Admin.

3. SQL database was chosen due to the relationships between the datasets which will make analysis clearer and easier.

```
from sqlalchemy import create_engine
```

```
rds_connection_string = "postgres:postgres@localhost:5432/Covid-ETL"
engine = create_engine(f'postgresql://{rds_connection_string}')
```

```
census_data.to_sql(name='final', con=engine, if_exists='append', index=False)
```

Data Output | Explain | Messages

	State text	Total Cases double precision	Total Deaths double precision	Case Growth Rate double precision	Death Growth Rate double precision	Population double precision	Percent_Seniors double precision	Shelter Start Date date
1	New York	83948	1941	0.16861635042753018	0.2661486105367089	19542209	16.43655023851193	2020-03-22
2	California	9399	199	0.1676379569104854	0.17520463171531225	39557045	14.32699788368924	2020-03-19
3	New Jer...	22255	355	0.21695462730421397	0.25670185870109075	8908520	16.145094808116276	2020-03-21
4	Washing...	5608	234	0.11825983131757993	0.09863709143440642	7535591	15.446525693870598	2020-03-23
5	Florida	6956	87	0.19990243581998726	0.18330076944132512	21299325	20.464423168339838	2020-03-24
6	Michigan	9315	335	0.20842243035127395	0.2975239062712573	9995915	17.2115609226369	2020-03-24