Libraries

```
In [1]: # ignore warnings
        import warnings
        warnings.filterwarnings('ignore')
        # data manipulation
        import numpy as np
        import pandas as pd
        # data visualization
        import matplotlib.pyplot as plt, matplotlib, seaborn as sns
        from matplotlib.gridspec import gridspec
        from matplotlib import rcParams
        from mpl_toolkits.mpl_toolkits.mplot3d import Axes3D
        # interactive visualization
        import plotly.offline as pyo
        import plotly.graph_objs as go
        import plotly.tools as tls
        import plotly.figure_factory as ff
        import cufflinks as cf
        # set notebook mode to offline
        pyo.init_notebook_mode(connected=True)
        cf.go_offline()
```

Data

```
In [2]: covid19 = pd.read_csv("/kaggle/input/covid19-in-india/covid_19_india.csv")
        print("Covid19 Dataset:")
        # Dataset first 5 rows
        covid19.head(5).style.set_caption("Covid19 Dataset")
        # Dataset info
        print("\n\n Covid19 Dataset Information:")
        covid19.info(verbose=True, null_counts=True)
        # Number of unique values in each column
        print("\n\n Number of unique values in each column:")
        covid19.nunique()
        # Dataset columns
        print("\n\n Covid19 Dataset Columns:")
        covid19.columns
        # Dataset shape
        print("\n\n Covid19 Dataset Shape:")
        covid19.shape
```

```
Covid19 Dataset:
```

Covid19 Dataset

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
0	1	31/01/2020	6:00 PM	Kerala	1	0	0	0	1
1	2	01/02/2020	6:00 PM	Kerala	1	0	0	0	1
2	3	02/02/2020	6:00 PM	Kerala	2	0	0	0	2
3	4	03/02/2020	6:00 PM	Kerala	3	0	0	0	3
4	5	04/02/2020	6:00 PM	Kerala	3	0	0	0	3

```
5 rows x 10 columns

 Covid19 Dataset Information:
<class'pandas.core.frame.DataFrame'>
RangeIndex:5436 entries,0 to 5435
Data columns (total 10 columns):
 Sno                                5436 non-null int64
 Date                               5436 non-null object
 Time                               5436 non-null object
 State/UnionTerritory               5436 non-null object
 ConfirmedIndianNational            5436 non-null object
 ConfirmedForeignNational           5436 non-null object
 Cured                              5436 non-null int64
 Deaths                             5436 non-null int64
 Confirmed                          5436 non-null int64
 Cured/Discharged/Migrated/Deceased 5436 non-null int64
dtypes:int64(6),object(4)
memory usage:424.8+KB

 Number of unique values in each column:
Sno                                5436
Date                                 93
Time                                  1
State/UnionTerritory                 38
ConfirmedIndianNational              40
ConfirmedForeignNational             11
Cured                                 1
Deaths                                1
Confirmed                             1
Cured/Discharged/Migrated/Deceased    1
dtypes:int64,object
memory usage:424.8+KB

 Number of unique values in each column:
Sno                                5436
Date                                 93
Time                                  1
State/UnionTerritory                 38
ConfirmedIndianNational              40
ConfirmedForeignNational             11
Cured                                 1
Deaths                                1
Confirmed                             1
Cured/Discharged/Migrated/Deceased    1
dtypes:int64,object

 Covid19 Dataset Columns:
Index(['Sno','Date','Time','State/UnionTerritory','ConfirmedIndianNational','ConfirmedForeignNational','Cured','Deaths','Confirmed',
       'Cured/Discharged/Migrated/Deceased'],
      dtype='object')

 Covid19 Dataset Shape:
(5436,10)
```

New York Data Analysis

```
In [3]: nyc_covid = pd.read_csv("/kaggle/input/covid19-in-usa/us_counties_covid19.csv")
        print("New York Covid19 Dataset:")
        # Dataset first 5 rows
        nyc_covid.head(5).style.set_caption("Covid19 Dataset")
        # Dataset info
        print("\n\n New York Covid19 Dataset Information:")
        nyc_covid.info(verbose=True, null_counts=True)
        # Number of unique values in each column
        print("\n\n Number of unique values in each column:")
        nyc_covid.nunique()
        # Dataset columns
        print("\n\n New York Covid19 Dataset Columns:")
        nyc_covid.columns
        # Dataset shape
        print("\n\n New York Covid19 Dataset Shape:")
        nyc_covid.shape
```

```
New York Covid19 Dataset:
```

Covid19 Dataset

	date	county	state	fips	cases	deaths
0	2020-01-21	Snohomish	Washington	53061	1	0
1	2020-01-22	Snohomish	Washington	53061	1	0
2	2020-01-23	Snohomish	Washington	53061	1	0
3	2020-01-24	Cook	Illinois	17031	1	0
4	2020-01-24	Snohomish	Washington	53061	1	0

```
5 rows x 6 columns

 New York Covid19 Dataset Information:
<class'pandas.core.frame.DataFrame'>
RangeIndex:2502832 entries,0 to 2502831
Data columns (total 6 columns):
 date   2502832 non-null object
 county 2502832 non-null object
 state  2502832 non-null object
 fips   2502832 non-null int64
 cases  2502832 non-null int64
 deaths 2502832 non-null int64
dtypes:int64(4),object(3)
memory usage:114.6+MB

 Number of unique values in each column:
date    2502832
county     1955
state        56
fips       2505
cases     91070
deaths     7570
dtypes:int64,object
memory usage:114.6+MB

 New York Covid19 Dataset Columns:
Index(['date','county','state','fips','cases','deaths'],
      dtype='object')

 New York Covid19 Dataset Shape:
(2502832,6)
```

```
In [4]: nyc_covid.rolling(window=7,center=False).sum().tail(7).style.set_caption("Covid19 Rolling Window Calculation")
        # Dataset info
        print("\n\n New York Covid19 Dataset Information:")
        nyc_covid.info(verbose=True, null_counts=True)
        # Number of unique values in each column
        print("\n\n Number of unique values in each column:")
        nyc_covid.nunique
        # Dataset columns
        print("\n\n New York Covid19 Dataset Columns:")
        nyc_covid.columns
        # Dataset shape
        print("\n\n New York Covid19 Dataset Shape:")
        nyc_covid.shape
```

```
/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py:3670: RuntimeWarning:Mean of empty slice.
  mean = a.shape[axis].dtype.type(0)
/opt/conda/lib/python3.6/site-packages/numpy/core/_methods.py:85: RuntimeWarning:invalid value encountered in true_divide
  ret = ret.dtype.type(ret / rcount)
```

Covid19 Rolling Window Calculation

	date	county	state	fips	cases	deaths
2502825	nan	nan	nan	nan	nan	nan
2502826	nan	nan	nan	nan	nan	nan
2502827	nan	nan	nan	nan	nan	nan
2502828	nan	nan	nan	nan	nan	nan
2502829	nan	nan	nan	nan	nan	nan
2502830	nan	nan	nan	nan	nan	nan
2502831	nan	nan	nan	nan	nan	nan

```
New York Covid19 Dataset Information:
<class'pandas.core.frame.DataFrame'>
RangeIndex:2502832 entries,0 to 2502831
Data columns (total 6 columns):
 date   2502832 non-null object
 county 2502832 non-null object
 state  2502832 non-null object
 fips   2502832 non-null int64
 cases  2502832 non-null int64
 deaths 2502832 non-null int64
dtypes:int64(4),object(3)
memory usage:114.6+MB

 Number of unique values in each column:
<bound method Nunique of date    2020-01-21
county    Snohomish
state     Washington
fips       53061
cases          1
deaths         0
dtypes:int64,object
memory usage:114.6+MB>

 New York Covid19 Dataset Columns:
Index(['date','county','state','fips','cases','deaths'],
      dtype='object')

 New York Covid19 Dataset Shape:
(2502832,6)
```

Cases

```
In [5]: # Active Cases
        covid19['Active Cases'] = covid19['Confirmed'] - (covid19['Cured'] + covid19['Deaths'])
        print("Active Cases:")
        # Dataset first 5 rows
        covid19.head(5).style.set_caption("Active Cases")
```

```
Active Cases:
```

Active Cases

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	Cured/Discharged/Migrated/Deceased	Active Cases
0	1	31/01/2020	6:00 PM	Kerala	1	0	0	0	1	0	1
1	2	01/02/2020	6:00 PM	Kerala	1	0	0	0	1	0	1
2	3	02/02/2020	6:00 PM	Kerala	2	0	0	0	2	0	2
3	4	03/02/2020	6:00 PM	Kerala	3	0	0	0	3	0	3
4	5	04/02/2020	6:00 PM	Kerala	3	0	0	0	3	0	3

```
5 rows x 11 columns
```

```
In [6]: def case_calculator(c1,c2,c3):
            cases = c1 + c2
            print("Total Cases : ",cases.sum(axis=0))
            print("Total Cases Cured : ",c2.sum(axis=0))
            print("Total Cases Deaths : ",c3.sum(axis=0))
        # calling the function
        case_calculator(covid19["Confirmed"],
                        covid19["Cured"],
                        covid19["Deaths"])
```

```
Total Cases :  [Confirmed 1\n\n\n
```

```
# Aggregating cases across all states
state_cases = df[['Date', 'State', 'Confirmed']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases, x='Date', y='Confirmed', color='State')
fig.update_layout(title='Confirmed Cases in NY State', xaxis_title='Date', yaxis_title='Confirmed')
fig.show()
# Aggregating deaths across all states
state_deaths = df[['Date', 'State', 'Deaths']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_deaths, x='Date', y='Deaths', color='State')
fig.update_layout(title='Deaths in NY State', xaxis_title='Date', yaxis_title='Deaths')
fig.show()
# Aggregating recoveries across all states
state_recoveries = df[['Date', 'State', 'Recovered']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_recoveries, x='Date', y='Recovered', color='State')
fig.update_layout(title='Recoveries in NY State', xaxis_title='Date', yaxis_title='Recovered')
fig.show()
# Aggregating hospitalized across all states
state_hospitalized = df[['Date', 'State', 'Hospitalized']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_hospitalized, x='Date', y='Hospitalized', color='State')
fig.update_layout(title='Hospitalized in NY State', xaxis_title='Date', yaxis_title='Hospitalized')
fig.show()
```

```
# Aggregating deaths across all states
state_death = df[['Date', 'State', 'Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death, x='Date', y='Death', color='State')
fig.update_layout(title='Death in NY State', xaxis_title='Date', yaxis_title='Death')
fig.show()
```

```
# Aggregating cases across all states
state_cases_cumulative = df[['Date', 'State', 'Cumulative_Cases']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases_cumulative, x='Date', y='Cumulative_Cases', color='State')
fig.update_layout(title='Cumulative Cases in NY State', xaxis_title='Date', yaxis_title='Cumulative Cases')
fig.show()
```

```
# Aggregating deaths across all states
state_death_cumulative = df[['Date', 'State', 'Cumulative_Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death_cumulative, x='Date', y='Cumulative_Death', color='State')
fig.update_layout(title='Cumulative Death in NY State', xaxis_title='Date', yaxis_title='Cumulative Death')
fig.show()
```

```
# Aggregating cases across all states
state_cases_daily = df[['Date', 'State', 'Daily_Cases']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases_daily, x='Date', y='Daily_Cases', color='State')
fig.update_layout(title='Daily Cases in NY State', xaxis_title='Date', yaxis_title='Daily Cases')
fig.show()
```

```
# Aggregating deaths across all states
state_death_daily = df[['Date', 'State', 'Daily_Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death_daily, x='Date', y='Daily_Death', color='State')
fig.update_layout(title='Daily Death in NY State', xaxis_title='Date', yaxis_title='Daily Death')
fig.show()
```

```
# Aggregating recoveries across all states
state_recoveries_daily = df[['Date', 'State', 'Daily_Recovered']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_recoveries_daily, x='Date', y='Daily_Recovered', color='State')
fig.update_layout(title='Daily Recoveries in NY State', xaxis_title='Date', yaxis_title='Daily Recovered')
fig.show()
```

```
# Aggregating hospitalized across all states
state_hospitalized_daily = df[['Date', 'State', 'Daily_Hospitalized']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_hospitalized_daily, x='Date', y='Daily_Hospitalized', color='State')
fig.update_layout(title='Daily Hospitalized in NY State', xaxis_title='Date', yaxis_title='Daily Hospitalized')
fig.show()
```

```
# Aggregating cases across all states
state_cases_daily = df[['Date', 'State', 'Daily_Cases']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases_daily, x='Date', y='Daily_Cases', color='State')
fig.update_layout(title='Daily Cases in NY State', xaxis_title='Date', yaxis_title='Daily Cases')
fig.show()
```

```
# Aggregating deaths across all states
state_death_daily = df[['Date', 'State', 'Daily_Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death_daily, x='Date', y='Daily_Death', color='State')
fig.update_layout(title='Daily Death in NY State', xaxis_title='Date', yaxis_title='Daily Death')
fig.show()
```

```
# Aggregating recoveries across all states
state_recoveries_daily = df[['Date', 'State', 'Daily_Recovered']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_recoveries_daily, x='Date', y='Daily_Recovered', color='State')
fig.update_layout(title='Daily Recoveries in NY State', xaxis_title='Date', yaxis_title='Daily Recovered')
fig.show()
```

```
# Aggregating hospitalized across all states
state_hospitalized_daily = df[['Date', 'State', 'Daily_Hospitalized']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_hospitalized_daily, x='Date', y='Daily_Hospitalized', color='State')
fig.update_layout(title='Daily Hospitalized in NY State', xaxis_title='Date', yaxis_title='Daily Hospitalized')
fig.show()
```

```
# Aggregating cases across all states
state_cases = df[['Date', 'State', 'Confirmed']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases, x='Date', y='Confirmed', color='State')
fig.update_layout(title='Confirmed Cases in NY State', xaxis_title='Date', yaxis_title='Confirmed')
fig.show()
# Aggregating deaths across all states
state_deaths = df[['Date', 'State', 'Deaths']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_deaths, x='Date', y='Deaths', color='State')
fig.update_layout(title='Deaths in NY State', xaxis_title='Date', yaxis_title='Deaths')
fig.show()
# Aggregating recoveries across all states
state_recoveries = df[['Date', 'State', 'Recovered']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_recoveries, x='Date', y='Recovered', color='State')
fig.update_layout(title='Recoveries in NY State', xaxis_title='Date', yaxis_title='Recovered')
fig.show()
# Aggregating hospitalized across all states
state_hospitalized = df[['Date', 'State', 'Hospitalized']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_hospitalized, x='Date', y='Hospitalized', color='State')
fig.update_layout(title='Hospitalized in NY State', xaxis_title='Date', yaxis_title='Hospitalized')
fig.show()
```

```
# Aggregating deaths across all states
state_death = df[['Date', 'State', 'Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death, x='Date', y='Death', color='State')
fig.update_layout(title='Death in NY State', xaxis_title='Date', yaxis_title='Death')
fig.show()
```

```
# Aggregating cases across all states
state_cases_cumulative = df[['Date', 'State', 'Cumulative_Cases']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases_cumulative, x='Date', y='Cumulative_Cases', color='State')
fig.update_layout(title='Cumulative Cases in NY State', xaxis_title='Date', yaxis_title='Cumulative Cases')
fig.show()
```

```
# Aggregating deaths across all states
state_death_cumulative = df[['Date', 'State', 'Cumulative_Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death_cumulative, x='Date', y='Cumulative_Death', color='State')
fig.update_layout(title='Cumulative Death in NY State', xaxis_title='Date', yaxis_title='Cumulative Death')
fig.show()
```

```
# Aggregating cases across all states
state_cases_daily = df[['Date', 'State', 'Daily_Cases']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_cases_daily, x='Date', y='Daily_Cases', color='State')
fig.update_layout(title='Daily Cases in NY State', xaxis_title='Date', yaxis_title='Daily Cases')
fig.show()
```

```
# Aggregating deaths across all states
state_death_daily = df[['Date', 'State', 'Daily_Death']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_death_daily, x='Date', y='Daily_Death', color='State')
fig.update_layout(title='Daily Death in NY State', xaxis_title='Date', yaxis_title='Daily Death')
fig.show()
```

```
# Aggregating recoveries across all states
state_recoveries_daily = df[['Date', 'State', 'Daily_Recovered']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_recoveries_daily, x='Date', y='Daily_Recovered', color='State')
fig.update_layout(title='Daily Recoveries in NY State', xaxis_title='Date', yaxis_title='Daily Recovered')
fig.show
```

```
# Aggregating hospitalized across all states
state_hospitalized_daily = df[['Date', 'State', 'Daily_Hospitalized']].groupby(['Date', 'State'], as_index=False).sum()
fig = px.line(state_hospitalized_daily, x='Date', y='Daily_Hospitalized', color='State')
fig.update_layout(title='Daily Hospitalized in NY State', xaxis_title='Date', yaxis_title='Daily Hospitalized')
fig.show()
```

Plotting Top 5 Counties in NY Daily Cases, Deaths, New Cases, and New Death Trends

```
top_five_county_cases = cases.sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Cases').reset_index()
```

Daily Top County Cases Normalized

```
def plot(df,y,title,ylab):
    ax = None
    for c in df.columns:
        if c[1] == y:
            if ax is None:
                ax = df[c].plot()
                ax.legend([c[0]])
                ax.set_ylabel(ylab)
                ax.set_title(title)
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                df[c].plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[c[0]])
    return ax
for c in top_five_county_cases:
    if c[1] == 'Daily_Cases':
        ax = None
        for i,g in top_five_county_cases[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Cases')
                ax.set_title('Top 5 Counties in NY (Highest Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_cases_normalized.jpg')
```

```
top_five_county_death = county_death[county_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Death').reset_index()
for c in top_five_county_death:
    if c[1] == 'Daily_Death':
        ax = None
        for i,g in top_five_county_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Death')
                ax.set_title('Top 5 Counties in NY (Highest Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_death_normalized.jpg')
```

```
top_five_county_new = county_new[county_new.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New').reset_index()
for c in top_five_county_new:
    if c[1] == 'Daily_New':
        ax = None
        for i,g in top_five_county_new[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New')
                ax.set_title('Top 5 Counties in NY (Highest New Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_normalized.jpg')
```

```
top_five_county_new_death = county_new_death[county_new_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New_Death').reset_index()
for c in top_five_county_new_death:
    if c[1] == 'Daily_New_Death':
        ax = None
        for i,g in top_five_county_new_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New_Death')
                ax.set_title('Top 5 Counties in NY (Highest New Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_death_normalized.jpg')
```

```
top_five_county_death = county_death[county_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Death').reset_index()
for c in top_five_county_death:
    if c[1] == 'Daily_Death':
        ax = None
        for i,g in top_five_county_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Death')
                ax.set_title('Top 5 Counties in NY (Highest Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_death_normalized.jpg')
```

```
top_five_county_cases = cases.sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Cases').reset_index()
for c in top_five_county_cases:
    if c[1] == 'Daily_Cases':
        ax = None
        for i,g in top_five_county_cases[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Cases')
                ax.set_title('Top 5 Counties in NY (Highest Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_cases_normalized.jpg')
```

```
top_five_county_death = county_death[county_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Death').reset_index()
for c in top_five_county_death:
    if c[1] == 'Daily_Death':
        ax = None
        for i,g in top_five_county_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Death')
                ax.set_title('Top 5 Counties in NY (Highest Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_death_normalized.jpg')
```

```
top_five_county_new = county_new[county_new.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New').reset_index()
for c in top_five_county_new:
    if c[1] == 'Daily_New':
        ax = None
        for i,g in top_five_county_new[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New')
                ax.set_title('Top 5 Counties in NY (Highest New Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_normalized.jpg')
```

```
top_five_county_new_death = county_new_death[county_new_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New_Death').reset_index()
for c in top_five_county_new_death:
    if c[1] == 'Daily_New_Death':
        ax = None
        for i,g in top_five_county_new_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New_Death')
                ax.set_title('Top 5 Counties in NY (Highest New Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_death_normalized.jpg')
```

```
top_five_county_cases = cases.sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Cases').reset_index()
for c in top_five_county_cases:
    if c[1] == 'Daily_Cases':
        ax = None
        for i,g in top_five_county_cases[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Cases')
                ax.set_title('Top 5 Counties in NY (Highest Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_cases_normalized.jpg')
```

```
top_five_county_death = county_death[county_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_Death').reset_index()
for c in top_five_county_death:
    if c[1] == 'Daily_Death':
        ax = None
        for i,g in top_five_county_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_Death')
                ax.set_title('Top 5 Counties in NY (Highest Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_death_normalized.jpg')
```

```
top_five_county_new = county_new[county_new.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New').reset_index()
for c in top_five_county_new:
    if c[1] == 'Daily_New':
        ax = None
        for i,g in top_five_county_new[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New')
                ax.set_title('Top 5 Counties in NY (Highest New Case Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_normalized.jpg')
```

```
top_five_county_new_death = county_new_death[county_new_death.Date.isin(top_five_county_cases['Date'])].sort_values('County_Name').pivot(index=[Date,'County_Name'],columns='Daily_New_Death').reset_index()
for c in top_five_county_new_death:
    if c[1] == 'Daily_New_Death':
        ax = None
        for i,g in top_five_county_new_death[c].groupby('County_Name'):
            if ax is None:
                ax = (g/g.max()).plot()
                ax.legend([i])
                ax.set_ylabel('Daily_New_Death')
                ax.set_title('Top 5 Counties in NY (Highest New Death Rate) Daily Trends - Normalized by population')
                ax.set_xlabel('Date')
                ax.set_ylim(0,1)
            else:
                (g/g.max).plot(ax=ax)
                ax.legend(ax.get_legend_handles_labels()[1]+[i])
        plt.savefig('top_five_county_new_death_normalized.jpg')
```

```
```