```
!pip install contextily
!pip install mapclassify
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import contextily
import mapclassify
import folium
import fsspec

%matplotlib inline
```

```
Requirement already satisfied: contextily in /usr/local/lib/python3.10/dist-packages (1.4.0)
Requirement already satisfied: geopy in /usr/local/lib/python3.10/dist-packages (from contextily) (2.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from contextily) (3.7.1)
Requirement already satisfied: mercantile in /usr/local/lib/python3.10/dist-packages (from contextily) (1.2.1)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from contextily) (9.4.0)
Requirement already satisfied: rasterio in /usr/local/lib/python3.10/dist-packages (from contextily) (1.3.9)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from contextily) (2.31.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from contextily) (1.3.2)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.10/dist-packages (from contextily) (2023.10.1)
Requirement already satisfied: geographiclib<3,>=1.52 in /usr/local/lib/python3.10/dist-packages (from geopy->contextily) (2
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (1.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (0.12.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (4
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (1
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (23.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily) (3.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->contextily)
Requirement already satisfied: click>=3.0 in /usr/local/lib/python3.10/dist-packages (from mercantile->contextily) (8.1.7)
Requirement already satisfied: affine in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (2.4.0)
Requirement already satisfied: attrs in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (23.1.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (2023.7.22)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (0.7.2)
Requirement already satisfied: snuggs>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (1.4.7)
Requirement already satisfied: click-plugins in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (1.1.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from rasterio->contextily) (67.7.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->contextil
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->contextily) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->contextily) (2.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->c
Requirement already satisfied: mapclassify in /usr/local/lib/python3.10/dist-packages (2.6.1)
Requirement already satisfied: networkx>=2.7 in /usr/local/lib/python3.10/dist-packages (from mapclassify) (3.2.1)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.10/dist-packages (from mapclassify) (1.23.5)
Requirement already satisfied: pandas!=1.5.0,>=1.4 in /usr/local/lib/python3.10/dist-packages (from mapclassify) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.10/dist-packages (from mapclassify) (1.2.2)
Requirement already satisfied: scipy>=1.8 in /usr/local/lib/python3.10/dist-packages (from mapclassify) (1.11.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.5.0,>=1.4->
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.5.0,>=1.4->mapclassif
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->mapclassify
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->mapc
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas!=1.5
```

```
# This loads geodataframe contaning county geometry shapes
c = "https://github.com/babdelfa/gis/blob/main/counties_geometry.zip?raw=true"
import fsspec
with fsspec.open(c) as file:
    county_shapes = gpd.read_file(file)

# This loads the most recent covid19 data from Johns Hopkins University's Github
url_cases = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confir
df_cases = pd.read_csv(url_cases)
url_deaths = "https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_US.csv"
df_deaths = pd.read_csv(url_deaths)
```

```
# Drop unused columns
df_cases.drop(['UID','iso2','iso3','code3','Combined_Key'], axis=1,inplace=True)

# Unpivot table from wide to long format
cases_melt = pd.melt(df_cases,
                     id_vars=['FIPS','Admin2','Province_State','Country_Region','Lat','Long_'],
                     var_name='Date',
                     value_name='cases')

df_deaths.drop(['UID','iso2','iso3','code3','Combined_Key'],axis=1,inplace=True)
death_melt = pd.melt(df_deaths,
                       id_vars=['FIPS','Admin2','Province_State','Country_Region','Lat','Long_','Population'],
                        var_name='Date',
                        value_name= 'deaths')
# Merge two dataframes
data = pd.merge(cases_melt,death_melt,
```

```
                how='left',
                on=['FIPS','Admin2','Province_State','Country_Region','Lat','Long_','Date'])

# Tranform `Date` column to right datetime format
data['Date'] = pd.to_datetime(data['Date'])

# Rename columns for readability
data.rename(columns={'Admin2':'County','Province_State':'State','County_Region':'Country'},inplace=True)

# Filter to include data from 2020 to 2022
data = data[(data['Date'] >= '2020-01-01') & (data['Date'] <= '2022-12-31')]


from os import stat
# Create user-input prompt
print("*** MIS 433 COVID19 REPORT ***")
county = input("Enter County: ")
print(county)

# Generate information from the dataset that is relevant the user input
county_subset=data[data.County == county]
state=county_subset.State.iloc[0]
population = county_subset.Population.iloc[0]
print(f"\nPopulation of {county}, {state}: {format(population, ',d')}")
first_case_date = county_subset[county_subset['cases']>0]["Date"].min()
print(f"\nFirst reported outbreak in {county}: {(first_case_date).strftime('%B %d, %Y')}")

print(f"\n{county} County COVID-19 Summary Statistics: ")

# Subset data to generate total cases within a specific year
county_subset2020 = county_subset[county_subset.Date.dt.year==2020].copy()
county_subset2020["new_cases"] = county_subset.cases.diff()
total_cases=int(county_subset2020.new_cases.sum())
county_subset2021 = county_subset[county_subset.Date.dt.year == 2021].copy()
county_subset2021["new_cases"] = county_subset.cases.diff()
total_cases_1 = int(county_subset2021.new_cases.sum())
county_subset2022 = county_subset[county_subset.Date.dt.year == 2022].copy()
county_subset2022["new_cases"] = county_subset.cases.diff()
total_cases_2 = int(county_subset2022.new_cases.sum())
cum_cases = county_subset2022.cases.iloc[-1]

# Display covid cases statistics of the county interested by user
print(f"- Average cases in 2020: {round(county_subset2020.new_cases.mean(),2)}")
print(f"- Average cases in 2021: {round(county_subset2021.new_cases.mean(),2)}")
print(f"- Average cases in 2022: {round(county_subset2022.new_cases.mean(),2)}")
print("- Total cases in 2020: {:,}".format(total_cases))
print("- Total cases 2021: {:,}".format(total_cases_1))
print("- Total cases 2022: {:,}".format(total_cases_2))
print(f"- Cumulative total nunber of cases: {format(cum_cases,',d')} (December 31,2022)")

# Plot the trend of covid cases past 3 years
plt.figure(figsize=(8,6))
plt.plot(county_subset.Date, county_subset.cases)
plt.xticks(rotation=45)
plt.title(f"Total COVID19 Cases for {county} County")
plt.xlabel('Date')
plt.ylabel('Total Number of Cases')

# Generate interactive Covid-19 choropleth map for the state that the user-input county belongs to
subset = data[['FIPS','County','State','Population','Date','cases']].copy()
subset_2022 = subset[(subset.State == county_subset.State.iloc[0]) & (subset.Date=="2022-12-31")]
df = pd.merge(county_shapes,subset_2022,left_on='FIPS_BEA',right_on='FIPS').drop(['Date','OBJECTID','Shape_Leng','Shape_Area','FIPS','FIPS_BEA'],axis=1
df.explore(column='cases',cmap="Set2",scheme="NaturalBReaks",legend=True)
```

```
*** MIS 433 COVID19 REPORT ***
Enter County: Fairfax
Fairfax

Population of Fairfax, Virginia: 1,147,532

First reported outbreak in Fairfax: March 08, 2020

Fairfax County COVID-19 Summary Statistics:
- Average cases in 2020: 127.84
- Average cases in 2021: 199.38
- Average cases in 2022: 368.65
- Total cases in 2020: 43,977
- Total cases 2021: 72,775
- Total cases 2022: 134,558
- Cumulative total nunber of cases: 251,310 (December 31,2022)
```