```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from bokeh.plotting import figure, show, output_file
from bokeh.layouts import column

print("\nEM 624 Fall 2022 - Final Project\n-----------------------------------\n")
print("---COVID-19 Data Analysis---\n")
print("---Ran by Shashank Khanna---\n------------------------------------\n\n")

###### ------- Setting the columns and rows to max width ------ #######

pd.set_option('display.width', 1100)
pd.set_option('display.max_columns', 900)
pd.set_option('display.max_rows', 9000)

###### ------- Reading the covid files provided by Prof. Lipizzi ------- #######

covid_counties_file = pd.read_csv('COVID-19_US-counties.csv')
covid_housings_file = pd.read_excel('CO-EST2019-ANNHU.xlsx', header=None) #deleting
the file's headers and substituting custom column names
covid_gdp_file = pd.read_excel('GDP_Counties.xlsx', header=None ,) #deleting the
file's headers and substituting custom column names
covid_census_file = pd.read_csv('cc-est2019-alldata.csv' , encoding='ISO-8859-1')
#ISO 8859-1 is a single-byte encoding that can represent the first 256 Unicode
characters.

# The date and fips columns are being removed because they don't give any useful
information.
covid_counties_file.drop(['fips'], axis=1, inplace=True)
covid_counties_file.drop(['date'], axis=1, inplace=True)

###### -------- Converting all the files into a dataframe -------- ######

covid_counties_df = pd.DataFrame(covid_counties_file)
covid_housings_df = pd.DataFrame(covid_housings_file)
covid_gdp_df = pd.DataFrame(covid_gdp_file)
covid_census_df = pd.DataFrame(covid_census_file)

####### ------ Preprocessing the COVID-19_US-COUNTIES file ------- ########

covid_counties_file.drop(['county'], axis=1, inplace=True)
# finding the total number of cases and fatalities by state by the use of pivot
function
pivot_states_file = pd.pivot_table(covid_counties_file, index=['state'],
values=['cases', 'deaths'],aggfunc=np.sum)
pivot_df = pd.DataFrame(pivot_states_file).reset_index() # resetting the index to
make further processing easier
pivot_df.sort_values(by=['cases'], ascending=False , inplace=True) # using
sort_values function to order the dataset by cases

pivot_df = pivot_df.drop(33) # removing the data for New York state as it is not
useful to us for further analysis

#printing the table
print('\n\nThe following table contains the cases and deaths with respect to the
States : \n\n',pivot_df)
```

```
d = pivot_df['deaths'] # adding the list of deaths into a variable 'd'
c = pivot_df['cases'] # adding the list of cases into a variable 'c'
mortality = (d / c)*100 # calculating the mortality rate by dividing the number of
cases by the number of deaths
pivot_df['mortality'] = mortality # inserting a column "mortality" into the
pivot_df dataframe

# preparing the data for using in the matplot graphs
states = pivot_df['state'].head(10) # using head to select only the top 10 data
deaths = pivot_df['deaths'].head(10)
cases = pivot_df['cases'].head(10)

######## ------- Matplot Multiple Bar Graph ------- #######

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(12, 7))
# Set position of bar on X axis
br1 = np.arange(len(cases))
br2 = [x + barWidth for x in br1]
# Verical Bar Plots
plt.bar(br1, cases, color ='r', width = barWidth,
        edgecolor ='grey', label ='Cases')
plt.bar(br2, deaths, color ='b', width = barWidth,
        edgecolor ='grey', label ='Deaths')
# Adding Xticks
plt.xlabel('States', fontweight='bold', fontsize=15)
plt.ylabel('Cases and Deaths', fontweight='bold', fontsize=15)
plt.xticks([r + barWidth for r in range(len(cases))],
           states)
plt.xticks(rotation=90) #rotating the x axis lables so that they fit into the graph
without overlapping
plt.title('Cases and Deaths V/s States Plot') # title of the graph
plt.legend() # displaying the legend of the graph
plt.show() # showing the graph

####### ------ Preprocessing the CO-EST2019-ANNHU file ------- #######

covid_housings_df.drop([0, 1, 2, 3, 4], inplace=True) # removing the 1st 5 rows
from the housing table as it is not useful to us.
covid_housings_df.drop([1, 2], axis=1, inplace=True) # removing the 2nd and 3rd
column from the table as it is not useful
# renaming the headers of the columns of the housing table as we had removed them
previously
covid_housings_df.rename(columns={0: 'County', 3: 'FY2010', 4: 'FY2011', 5:
'FY2012',6: 'FY2013', 7: 'FY2014', 8: 'FY2015', 9: 'FY2016', 10: 'FY2017', 11:
'FY2018', 12: 'FY2019'},inplace=True)
covid_housings_df.reset_index(inplace=True) # inplace works like COMMIT in SQL it
makes sure that the changes stay.
covid_housings_df.drop(['index'], axis=1, inplace=True) # dropping the index column
of the table to make it more presentable
covid_housings_df.drop(covid_housings_df.tail(6).index,inplace=True) # removing the
last 6 rows of the table as they are not useful for processing
covid_housings_df.sort_values(by=['FY2019'], ascending=False, inplace=True) #
sorting the table in descending order by FY2019 data
# printing the top 50 data from the table
print('\nFollowing is the table for Housing Unit Estimates as per county from Year
2010 - 19 : \n\n',covid_housings_df.head(50))
```

```
######## -------- Matplot Bar Graph for Housing Units Table ------- ########

# set width of bar
barWidth = 0.4
housing_county = covid_housings_df['County'].head(10)
housing_2019 = covid_housings_df['FY2019'].head(10)
fig = plt.figure(figsize=(12,7))
plt.bar(housing_county,housing_2019)
# Verical Bar Plot
plt.xlabel('Counties', fontweight='bold', fontsize=15)
plt.ylabel('Housing Units in 2019', fontweight='bold', fontsize=15)
# adding xticks
plt.xticks(rotation=90)
plt.title('Housing Units of 2019 V/s County plot')
plt.show()

####### -------  Preprocessing the GDP_Counties file ------- #######

covid_gdp_df.drop([0, 1, 2, 3, 4, 5], inplace=True)  # removing header rows
covid_gdp_df.rename(columns={0: 'state', 1: "2015 GDP", 2: "2016 GDP", 3: "2017
GDP", 4: "2018 GDP", 5: "2018 Rank", 6: "2016 Change",
                                        7: "2017 Change" , 8: "2018 Change", 9:
"Rank Change"}, inplace=True) # renaming the columns with custom values
covid_gdp_df.drop(["2018 Rank" , "2016 Change" , "2017 Change" , "2018 Change" ,
"Rank Change"], axis=1, inplace=True)  # removing unnecessary columns
covid_gdp_df = covid_gdp_df.set_index('state')  # setting the index as the column
"state"
covid_gdp_table = covid_gdp_df # renaming the dataframe as a table
covid_gdp_table.reset_index(inplace=True)
covid_gdp_table.sort_values(by=['2018 GDP'], ascending=False, inplace=True) #
sorting the table in descending order
covid_gdp_table = covid_gdp_table.drop(1890) # dropping the state of New York as it
is not part of our analysis
covid_gdp_table = covid_gdp_table.drop(1921)
print('\n\nFollowing table contains the GDP values for all States : \n\
n',covid_gdp_table.head(50))

####### ------ Matplot Multi-line graph for GDP table ------ #######

# inputting the state data from gdp table into a variable x
x = covid_gdp_table['state'].head(10)
# # inputting the yearly GDP data from gdp table into a variable y1, y2, y3, y4
y1 = covid_gdp_table['2018 GDP'].head(10)
y2 = covid_gdp_table['2017 GDP'].head(10)
y3 = covid_gdp_table['2016 GDP'].head(10)
y4 = covid_gdp_table['2015 GDP'].head(10)
# setting the labels for each line plot
plt.plot(x,y1, label="2018 GDP")
plt.plot(x,y2, label="2017 GDP")
plt.plot(x,y3, label="2016 GDP")
plt.plot(x,y4, label="2015 GDP")
# Line Plot
plt.xlabel("States" ,fontweight='bold', fontsize=15)
plt.ylabel("GDP",fontweight='bold', fontsize=15)
# setting xticks
plt.xticks(rotation=90)
plt.title('GDP for years 2015 - 18 V/s States plots ')
plt.legend()
plt.show()

###### ------ Preprocessing the cc-est2019-alldata file ------- #######

covid_census_df.drop(["SUMLEV", "STATE", "COUNTY", "YEAR", "AGEGRP" , "CTYNAME"],
axis=1, inplace=True) # removing the following columns as they are not required for
processing
covid_census_df.sort_values(by=['TOT_POP'], ascending=False , inplace=True)
# using pivot table function to obtain the total
pivot_census_file = pd.pivot_table(covid_census_df, index=['STNAME'],
values=['TOT_POP', 'TOT_MALE', 'TOT_FEMALE'],aggfunc=np.sum)
pivot_census_df = pd.DataFrame(pivot_census_file).reset_index()
pivot_census_df.sort_values(by='TOT_POP', ascending=False ,inplace=True)
pivot_census_df.rename(columns = {'STNAME':'state'}, inplace = True)
pivot_census_df.rename(columns = {'TOT_POP':'Total'}, inplace = True)
pivot_census_df.rename(columns = {'TOT_MALE':'Male'}, inplace = True)
pivot_census_df.rename(columns = {'TOT_FEMALE':'Female'}, inplace = True)
pivot_census_df = pivot_census_df.drop(32)
# printing the table
print('\n\nFollowing is the table for the population distribution of all the
respective states : \n\n')
print(pivot_census_df.head(50))

###### ------ Matplot Bar Graph ------ ######

barWidth = 0.4
# inputting the variables with the data for the plot from the census table
census_states = pivot_census_df['state'].head(15)
census = pivot_census_df['Total'].head(15)
fig = plt.figure(figsize=(12,7))
plt.bar(census_states,census)
# Bar Plot
plt.xlabel('States', fontweight='bold', fontsize=15)
plt.ylabel('Total Population', fontweight='bold', fontsize=15)
#settings xticks
plt.xticks(rotation=90)
plt.title('Total Population V/s States Plot')
plt.show()

###### ----- MERGING OF TABLES TO FIND RELATIONSHIP BETWEEN THEM ------ ######

#Merging the population table with the cases and deaths table
covid_merge_1 = pivot_df.merge(pivot_census_df, on='state') # using "state" as the
primary key to merge
# printing the merged table
print("\n\nFollowing is the merged population and COVID19 cases/deaths table : \n\
n",covid_merge_1.head(50))

#Merging the table containing the population , covid19 cases and deaths with the
GDP table
covid_merge_2 = covid_merge_1.merge(covid_gdp_table, on='state')
covid_final_merge = covid_merge_2.drop_duplicates(subset=['state'], keep=False) #
removing the duplicates, getting rid of any similar values that come after the 1st
print("\n\nFollowing is the Merged population, COVID19 cases/deaths, and GDP Table:
\n\n",covid_final_merge.head(50),'\n\nThe following Warning can be ignored\n')

####### ----- PROCESSING THE ADDITIONAL FILES DOWNLOADED FROM THE NET -------
#######
```

```
# reading the pollution file
pollution_file = pd.read_csv('2019-Annual (2).csv')
# converting the csv file to a dataframe for further processing
pollution_table = pd.DataFrame(pollution_file)
 # removing the columns from the pollution table as it is not useful to us.
pollution_table.drop(['Edition','Report Type', 'Score','Lower CI','Upper
CI','Source','Source Year' ], axis=1, inplace=True)
# renaming the state column so we can merge the tables
pollution_table.rename(columns = {'State Name':'state'}, inplace = True)
airpollution = pollution_table['Measure Name'].values == 'Air Pollution' #
selecting all the values from the table which contain measure name as Air pollution
pollution_df = pollution_table[airpollution]
pollution_df.reset_index(inplace= True)
pollution_df.drop(['index'], axis=1, inplace=True)
pollution_df = pollution_df.drop(51) # the value for Dominica republic is null
hence not useful
pollution_df = pollution_df.drop(50) # this is the value for the whole of USA ,
hence not required.
pollution_df = pollution_df.sort_values(by='Value', ascending=False) # sorting the
table as per air quality values

#Merging the pollution and cases/deaths table
covid_merge_3 = covid_merge_1.merge(pollution_df, on='state')
#printing the merged table
print("\n\nFollowing table contains the Merged population, COVID19 cases/deaths,
GDP and Pollution Table : \n\n",covid_merge_3.head(50))

#reading the housing units file
housing_est_file = pd.read_excel('NST-EST2019-ANNHU (1).xlsx' , header=None)
# converting the file into a dataframe for processing
housings_table = pd.DataFrame(housing_est_file)
housings_table.drop([0, 1, 2, 3, 4,5,6,7,8], inplace=True) # removing the 1st 8
rows from the housing table as it is not useful to us.
housings_table.drop([1, 2,3,4,5,6,7,8,9,10,11], axis=1, inplace=True) # removing
the all columns that are not useful
# renaming the columns as given below
housings_table.rename(columns={0: 'state', 12: 'FY2019'} , inplace=True)
housings_df = housings_table.reset_index()
housings_df.drop(['index'],axis=1, inplace=True)
housings_df.drop(housings_df.tail(5).index,inplace=True) # removing the last 6 rows
of the table as they are not useful for processing
housings_df.sort_values(by=['FY2019'], ascending=False , inplace=True) # sorting in
descending order by FY2019 data

# merging the housing data
covid_merge_4 = covid_merge_3.merge(housings_df, on='state')

# setting the variable from the merged table for further use, using only the top 10
values
merge_states = covid_merge_4['state'].head(10)
merge_cases = covid_merge_4['cases'].head(10)
merge_deaths = covid_merge_4['deaths'].head(10)
merge_census = covid_merge_4['Total'].head(10)
merge_housing = covid_merge_4['FY2019'].head(10)
merge_mortality = covid_merge_4['mortality'].head(10)
merge_pollution = covid_merge_4['Value'].head(10)

# find avg number of people per household
avg_per_house = merge_census / merge_housing
```

```
# printing the table
print("\n\nFollowing table contains the Merged Final Table : \n\
n",covid_merge_4.head(50))

###### ----- BOKEH PLOTS  ------ ######

# initializing the bokeh plot for states v/s mortality
p1 = figure(width=900,
            height=700,
            x_range=merge_states,
            title='State-wise Distribution of Mortality Rate',
            y_axis_label='Mortality Rate (in %)', x_axis_label='State')
# setting a veritcal bar plot with varible as x = states data and top = mortality
rate data
p1.vbar(x=merge_states, top=merge_mortality, width=0.6, color="#ed7b11" )

# initializing the bokeh plot for states v/s average person per house
p2 = figure(width=700,
            height=900,
            y_range=merge_states,
            title='Average Number of people per household as per States',
            x_axis_label='Number of people per household', y_axis_label='States')
# setting a horizontal bar plot with varible as x = states data and right = average
person per household data
p2.hbar(y=merge_states, right=avg_per_house, height=0.8, width=0.6,
color="#42f566")

# initializing the bokeh plot for states v/s average person per house
p3 = figure(width=900, height=700, x_range=merge_states, title='Pollution V/S
States', x_axis_label='States',
            y_axis_label="Pollution")
# setting a line graph with x axis = states data and y axis = pollution data
p3.line(merge_states, merge_pollution, line_width=2)

# save the plots into the output file
output_file("output_project.html")
# showing the bokeh plots
show(column(p1,p2,p3))
```