Project Report - Phillip Marsh

GitHub URL

Phillip's GutHub can be found at: [Phillip's GitHub Repository (https://github.com/PhillipNM/UCDPA_PhillipMarsh)](https://github.com/PhillipNM/UCDPA_PhillipMarsh)

document should contain between 1,500 and 2,00 words

Abstract

(short overview of the entire project)

For this project I chose to review COVID data as I was somewhat familiar with the underlying data but only from creating metrics on the data. I wanted to gain some further understanding of the situation and felt there would be a lot of data options available. The results did not turn out as I planned but the excercise was rewarding but very challenging. Trying to cover such a large scope of skills with in python and the huge amount of imformation on tips and tricks, although many sites are not that useful and I spent hours between the DataCamp videos and onlineadvice sites. It turns out that population density and economic prosperity of a country does not have much of an impact of a disease like COVID which I guess is why people are not panicking each flu season. I would have loved to added some insights into the impact of masking, and lock downs but tring to join that periodic data in with this daily data was too much of a challenge for this short period of time.

Introduction

(Explain why you chose this project use case)

After considering serveral ideas and researching the available dataset I decide on a dataset I am fairly familiar with from a reporting point of view (as part of the business continuity team) but that I had not done much with the other than create some metrics using Tableau. I wondered if we could predict confidently that countries with lower population densities or high GDP per capita fared better than higher density countries or lower GDP.

Datasets

(Provide a description of your dataset and source. Also justify why you chose this source)

Deciding on the dataset

I had several ideas, however, I explored three main ideas:

1. Predicting currency fx changes to maximise buys and sells.
As I have two children in Canada in university the fx rate for USD to CAD is always top of mind. After exploring this for a bit the challenge to understand the market conditions that I could use for making predictions did not seem to fit well with what I needed for this proje ct.
2. Flight delays, cancellations and the average compensation. Are the airlines "gaming" t he system to not pay-out customers given the turmoil in travel I thought it would be interes ting to compare recent cancellations, delays and reasons and compensations vs. pre-covi d data. I researched for datasets but could not find anything current, although there were some sites that may have had data; I would have to pay for and for this reason I decided against this topic.
3. COVID data. This idea would have plenty of source data out there but would it offer th e ability to make predictions and not just forecasting trends.

COVID Data

I picked the COVID idea as there is good data and the types of calculations and techniques required would lend itself to the project easily. This data is something we are all very familiar with at this time. Governments, countries, organizations and corporations have struggled with rules and regulations trying to balance controlling the epidemic vs. economic stability.

I reviewed a couple of sources and in the end selected "Our World In Data" (OWID). OWID has a comprehensive set of publicly available data specifically for COVID. In working with the FIL business continuity team, I assited with the COVID response. I came across this data source and found it very useful. In the end this is the source we used to provide global situational updates for the senior members in the organization so they could decide on stay at home and return to office responses for each jurisdiction across the organization.

source of covid data: [https://github.com/owid/covid-19-data/tree/master/public/data (https://github.com/owid/covid-19-data/tree/master/public/data)](https://github.com/owid/covid-19-data/tree/master/public/data)

Originally I downloaded a (.csv) copy of the data to use but the file was large (I was getting an error that the file was to big for my type of GitHub repository account). this occured when I pushed the data to my GitHub repository. I then researched how I could link to an external csv file, and this solved the problem. This file creats the opportunity to use current data. However, I noticed that the most current days data is not 100% populated so I have adjusted to used the most recent data - two days.

source of GDP data: [https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=false (https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=false)](https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=false)

the file is a zip file which is difficult to connect to so in this instance I downloaded the file and unzipped it.

Implementation Process

(describe your entire process in detail)

Hypothesis

My hypothesis is that countries with higher population density and lower GDP have higher mortality rates than for higher density higher GDP countries. It would also be interesting to see how lower density and higher GDP countries fared and if density and GDP are a predicor of mortaility for a disease like COVID

The implementation process I followed was
- Gather Data
- Transform & clean
- Explore
- Analyze and build models

Gather Data
There are several measures I need for my analysis if any of the data sets include 0 values for total I will use the prior days data as total are cumlative

Measures for each country:
- Highest Cases per 100k people: for year end 2020, 2021 and latest 2022
- Highest Deaths per 100k people: for year end 2020, 2021 and latest 2022
- Lowest Cases per 100k people: for year end 2020, 2021 and latest 2022
- Lowest Deaths per 100k people: for year end 2020, 2021 and latest 2022
- Look at the 14 day rolling average cases per 100k people over time
- Look at the 14 day rolling average deaths per 100k people over time
- Population density
- GDP per person

Transform & Clean and Explore
Review data for size and complexity, NaNs and missing values. Use techiques like

- .head()
- .tail()
- .info()
- .shape()
- .isna().sum()

to understand the number of columns, count of records and the type of object being used, like strings, dates, intergers and floats. Review the null records and get a sum to understa nd the completeness of the data, and functions to assit with exploring the data like creatin g a rolling n day average and calulation for the total on a per 100,00 of the population for comparatives

Analyze and build models
Take the top 20: Categorize as High, Low for mortality and add to the data set. This will allow some of the linear regression models for correlations

Run agaisnt the machine learnng logic for insights

Import and review the data

In [1]:
```
# Import packages needed for project:
import pandas as pd
import requests
import io
import datetime as dt
from datetime import datetime
from datetime import timedelta
import numpy as np
from collections import Counter
import re
import sklearn

# Visualization
import matplotlib.pyplot as plt
# import matplotlib.animation as animation
import seaborn as sns

# Machine learning
#from sklearn.module import Model
from sklearn.linear_model import LinearRegression, LogisticRegression, Ridge,
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

create global variables

In [2]:
```
#how many columns are too many to wrangle
column_count_limit=30 #number of columns deemed to be managble for exploring
#this will allow a use to run a calculation to high light if a detset has a l

#number of days used in rolling average default = 14 but user could change to
#is relevant
days_calc = 14 #n days for calculations.

top_n_parameter = 10 # was 10 #variable to use for select the number of top a

pop_per_100k = 100000 #varibale to set for total cases and deaths per populat

#for calculations relating to mortality
high_deaths_per_100k = 50 # was 50
low_deaths_per_100k = 10 # was 10
# I decide on this after reviewing the min and maxk values for the topn recor
```

Gather data

In [3]:
```
# Import COVID data

# Link and download COVID dataset from OWID
url = "https://covid.ourworldindata.org/data/owid-covid-data.csv"
download = requests.get(url).content

# Create the COVID as a pandas dataframe
covid_data_raw = pd.read_csv(io.StringIO(download.decode('utf-8')),parse_date
#source: https://stackoverflow.com/questions/59004960/converting-date-format-
```

review of covid header details:

In [4]:
```
covid_data_raw.head()
```

Out[4]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	tota
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	

5 rows × 67 columns

a quick review show there are a lot of columns of which most will be irrelivant. There also records with NaN which will have to be dealt with as they would impact calulations.

In [5]:
```
# Import World Bank GDP data
    # source: https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_d

    # Create the GDP raw file as a pandas dataframe, headers start on row 4
gdp_data_raw = pd.read_csv("/Users/Phillip/UDCPA_PhillipMarsh/data/API_NY.GDP
```

review of global gdp details:

In [6]:
```
gdp_data_raw.head()
```

Out[6]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962
0	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD	NaN	NaN	NaN
1	Africa Eastern and Southern	AFE	GDP (current US$)	NY.GDP.MKTP.CD	2.129059e+10	2.180847e+10	2.370702e+10
2	Afghanistan	AFG	GDP (current US$)	NY.GDP.MKTP.CD	5.377778e+08	5.488889e+08	5.466667e+08
3	Africa Western and Central	AFW	GDP (current US$)	NY.GDP.MKTP.CD	1.040414e+10	1.112789e+10	1.194319e+10
4	Angola	AGO	GDP (current US$)	NY.GDP.MKTP.CD	NaN	NaN	NaN

5 rows × 67 columns

a quick review shows there are also alot of columns of year dat most of which would not be relevant. This data also uses 3 digit ISO codes which means I can use it to join to data if need be.

create global calculations to be used in the analysis

there are a few calculations that will be used repeatedly and it makes sense to put them at the start of teh project so they are easy to find if changes need to be made

In [7]:
```
#global calculation

# What are the range of dates in data
beg_date = min(covid_data_raw["date"]) #starting point of the available data
end_date = max(covid_data_raw["date"]) #most recent data in the file

#calculate the lastest observation form the covid data, this data is dynamic
#it can take time for new data to roll in.  This report is using the last dat
last_date = end_date - timedelta(2)
last_date_n = str(last_date)

print("The COVID data starts on "+str(beg_date)+" and the most recent date is
```

```
The COVID data starts on 2020-01-01 00:00:00 and the most recent date is 20
22-09-18 00:00:00
```

Exploring the data

Review the headers, number of headers, type of data to undestand more about the data available

In [8]:
```
# name of a dataframe with comment before and after

def name_obj(df, comment, comment2=""):
    """Create statement naming the dataframe around comment and comment2

    Args:
        df (dataFrame): the name of the dataframe
        comment (string): comment string which goes before the name of the da
        comment2 (string): comment string which goes after the name of the da
    """
    name =[x for x in globals() if globals()[x] is df][0]
    return (comment+name+comment2)

covid_data_raw_name = name_obj(covid_data_raw,"Dataframe Name is:")
gdp_data_raw_name = name_obj(gdp_data_raw,"Dataframe Name is:")

#example: test the function
print("There are two primary sourced datasets used in this project:")
print(covid_data_raw_name)
print(gdp_data_raw_name)
```

```
There are two primary sourced datasets used in this project:
Dataframe Name is:covid_data_raw
Dataframe Name is:gdp_data_raw
```

In [9]:
```
# create functions for reviewing dataframe headers

# create a function to make list from the column header names of a dataframe
def column_headers_list(df):
    """create a list of column headers

    Args:
        df (DataFrame): the name of the dataframe to use

    Returns:
        list of column headers
    """

    columns_lst = df.columns.tolist() # create a list of the column headers f

    return columns_lst
```

In [10]:

```python
# Count the number of items in the list from the column header names list of

#test the function "column_headers_list"

# Raw Covid data
columns_lst_test = column_headers_list(covid_data_raw)
columns_len_test = len(columns_lst_test)

# Test function
#print(columns_lst_test)
#print(columns_len_test)


# Raw gdp data
columns_lst_test = column_headers_list(gdp_data_raw)
#print("There are :"+str(columns_len_test)+" header records")

# Test function
#print(columns_lst_test)
#print("There are :"+str(columns_len_test)+" header records")
```

In [11]:

```python
# create a function determine if the data set is too wide
def columns_comment(xlist,column_count_limit=30):
    """Use column_len to decide if the dataframe is too large to manage

    Args:
        xlist(list): list to review
        columns_len(int): from column_headers_list function
        column_count_limit(float): limi number of columns to compare
    """
    columns_len = len(xlist)

    if columns_len>column_count_limit:
        comment = "There are many columns ("+str(columns_len)+"), Drop a some
    else:
        comment = "Number of columns appears manageable"

    return comment, columns_len
```

we can see both datasets contain quite alot of columns with data

COVID Raw Data

In [12]:

```python
# review Covid raw data

# show the column headers and the number of columns

data = covid_data_raw

columns_len = data.shape[1] # count the number of columns in the list

covid_name = name_obj(data,"The headers from the", "Dataframe are:")
print(column_headers_list(data))

print()

description_covid_raw = name_obj(data,"The "," DataFrame has "+str(columns_le
#print(description_covid_raw)

print()

#print(columns_comment(column_headers_list(df))) #xlist,column_count_limit=30
```

```
['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
'new_cases_smoothed', 'total_deaths', 'new_deaths', 'new_deaths_smoothed',
'total_cases_per_million', 'new_cases_per_million', 'new_cases_smoothed_per
_million', 'total_deaths_per_million', 'new_deaths_per_million', 'new_death
s_smoothed_per_million', 'reproduction_rate', 'icu_patients', 'icu_patients
_per_million', 'hosp_patients', 'hosp_patients_per_million', 'weekly_icu_ad
missions', 'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
'weekly_hosp_admissions_per_million', 'total_tests', 'new_tests', 'total_te
sts_per_thousand', 'new_tests_per_thousand', 'new_tests_smoothed', 'new_tes
ts_smoothed_per_thousand', 'positive_rate', 'tests_per_case', 'tests_unit
s', 'total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated',
'total_boosters', 'new_vaccinations', 'new_vaccinations_smoothed', 'total_v
accinations_per_hundred', 'people_vaccinated_per_hundred', 'people_fully_va
ccinated_per_hundred', 'total_boosters_per_hundred', 'new_vaccinations_smoo
thed_per_million', 'new_people_vaccinated_smoothed', 'new_people_vaccinated
_smoothed_per_hundred', 'stringency_index', 'population', 'population_densi
ty', 'median_age', 'aged_65_older', 'aged_70_older', 'gdp_per_capita', 'ext
reme_poverty', 'cardiovasc_death_rate', 'diabetes_prevalence', 'female_smok
ers', 'male_smokers', 'handwashing_facilities', 'hospital_beds_per_thousan
d', 'life_expectancy', 'human_development_index', 'excess_mortality_cumulat
ive_absolute', 'excess_mortality_cumulative', 'excess_mortality', 'excess_m
ortality_cumulative_per_million']
```

In [13]:

```
#test function columns_comment()

# test for covid data
columns_lst_covid = column_headers_list(covid_data_raw) #list of headers

comment_covid = columns_comment(columns_lst_covid,)[0] #Comment string
header_len_covid = columns_comment(columns_lst_covid,column_count_limit=30)[1

print("for the COVID raw file")
print(comment_covid)

print("-"*100)
```

```
for the COVID raw file
There are many columns (67), Drop a some of them to imporve performance and
the size of the file
---------------------------------------------------------------------------
-------------------------
```

In [14]:

```
# test for gdp data
columns_lst_gdp = column_headers_list(gdp_data_raw)

comment_gdp = columns_comment(columns_lst_gdp)[0]
header_len_gdp = columns_comment(columns_lst_gdp,column_count_limit=30)[1]

print("for the gdp raw file")
print(comment_gdp)

print("-"*100)
```

```
for the gdp raw file
There are many columns (67), Drop a some of them to imporve performance and
the size of the file
---------------------------------------------------------------------------
-------------------------
```

use the shape function to summarize the total number of rows and columns for each dataset:

In [15]:

```
# Understanding the data

# Information (shape) on are the records + columns

# covid raw data
print("The COVID data shape shows:")
print(covid_data_raw.shape)
print()
print("The GDP data shape shows:")
# gdp raw data
print(gdp_data_raw.shape)
```

```
The COVID data shape shows:
(217284, 67)

The GDP data shape shows:
(266, 67)
```

There are also a lot of records for the COVID data, we should limit the number of days to review, but lets remove many of the columns and create a new covid_data DataFrame from the raw file

In [16]:

```
# drop columns
    #source: https://datatofish.com/drop-columns-pandas-dataframe/#:~:text=He
covid_data = covid_data_raw.drop([
    'continent',
    'new_cases_smoothed',
    'new_deaths_smoothed',
    'new_cases_smoothed_per_million',
    'new_deaths_smoothed_per_million',
    'icu_patients_per_million',
    'hosp_patients',
    'hosp_patients_per_million',
    'weekly_icu_admissions',
    'weekly_icu_admissions_per_million',
    'weekly_hosp_admissions',
    'weekly_hosp_admissions_per_million',
    'total_tests_per_thousand',
    'new_tests_per_thousand',
    'new_tests_smoothed',
    'tests_per_case',
    'tests_units',
    'new_vaccinations_smoothed',
    'total_vaccinations_per_hundred',
    'people_vaccinated_per_hundred',
    'people_fully_vaccinated_per_hundred',
    'total_boosters_per_hundred',
    'new_vaccinations_smoothed_per_million',
    'new_people_vaccinated_smoothed',
    'new_people_vaccinated_smoothed_per_hundred',
    'stringency_index','median_age',
    'aged_65_older',
    'aged_70_older',
    'cardiovasc_death_rate',
    'diabetes_prevalence',
    'female_smokers',
    'male_smokers',
    'handwashing_facilities',
    'hospital_beds_per_thousand',
    'life_expectancy',
    'human_development_index',
    'excess_mortality_cumulative_absolute',
    'excess_mortality_cumulative',
    'excess_mortality',
    'excess_mortality_cumulative_per_million',
    'total_cases_per_million',
    'new_cases_per_million',
    'total_deaths_per_million',
    'new_deaths_per_million',
    'reproduction_rate',
    'people_vaccinated',
    'total_boosters',
    'new_vaccinations',
    'new_tests_smoothed_per_thousand',
    'new_tests',
    'positive_rate'
    ],
    axis=1)
```

```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217284 entries, 0 to 217283
Data columns (total 15 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   iso_code                 217284 non-null  object
 1   location                 217284 non-null  object
 2   date                     217284 non-null  datetime64[ns]
 3   total_cases              208374 non-null  float64
 4   new_cases                208084 non-null  float64
 5   total_deaths             189322 non-null  float64
 6   new_deaths               189248 non-null  float64
 7   icu_patients             27684 non-null   float64
 8   total_tests              79387 non-null   float64
 9   total_vaccinations       60774 non-null   float64
 10  people_fully_vaccinated  55335 non-null   float64
 11  population               216013 non-null  float64
 12  population_density       192944 non-null  float64
 13  gdp_per_capita           177859 non-null  float64
```

Using the .info() function we now have 14 columns that look relevant to the analysis, how many countries are there

In [17]:

```
# all but the first three columns are float objects; two: "iso_code", "locati
# create a function to get a list of unique values

# Function to get unique values


def unique(list1):

    # Print directly by using * symbol
    print(*Counter(list1))
```

In [18]:
```
#list of covid countrie ISO code

# sort by ISO_code and Date
covid_data = covid_data.sort_values(['iso_code', 'location', 'date'])

#create a list of country codes from the covid data
Country_lst_covid_1 = covid_data["iso_code"].tolist()

# list the country codes
country_iso_list = unique(Country_lst_covid_1)
```

```
ABW AFG AGO AIA ALB AND ARE ARG ARM ATG AUS AUT AZE BDI BEL BEN BES BFA BGD
BGR BHR BHS BIH BLR BLZ BMU BOL BRA BRB BRN BTN BWA CAF CAN CHE CHL CHN CIV
CMR COD COG COK COL COM CPV CRI CUB CUW CYM CYP CZE DEU DJI DMA DNK DOM DZA
ECU EGY ERI ESH ESP EST ETH FIN FJI FLK FRA FRO FSM GAB GBR GEO GGY GHA GIB
GIN GMB GNB GNQ GRC GRD GRL GTM GUM GUY HKG HND HRV HTI HUN IDN IMN IND IRL
IRN IRQ ISL ISR ITA JAM JEY JOR JPN KAZ KEN KGZ KHM KIR KNA KOR KWT LAO LBN
LBR LBY LCA LIE LKA LSO LTU LUX LVA MAC MAR MCO MDA MDG MDV MEX MHL MKD MLI
MLT MMR MNE MNG MNP MOZ MRT MSR MUS MWI MYS NAM NCL NER NGA NIC NIU NLD NOR
NPL NRU NZL OMN OWID_AFR OWID_ASI OWID_CYN OWID_EUN OWID_EUR OWID_HIC OWID_
INT OWID_KOS OWID_LIC OWID_LMC OWID_NAM OWID_OCE OWID_SAM OWID_UMC OWID_WRL
PAK PAN PCN PER PHL PLW PNG POL PRI PRK PRT PRY PSE PYF QAT ROU RUS RWA SAU
SDN SEN SGP SHN SLB SLE SLV SMR SOM SPM SRB SSD STP SUR SVK SVN SWE SWZ SXM
SYC SYR TCA TCD TGO THA TJK TKL TKM TLS TON TTO TUN TUR TUV TWN TZA UGA UKR
URY USA UZB VAT VCT VEN VGB VIR VNM VUT WLF WSM YEM ZAF ZMB ZWE
```

Review of this object shows there are some ISO_Codes that are more than the standard 3 char length, these should be removed these are related to OWID codes for regional aggregations of country data, they can be removed

In [19]:
```
# Review of this object shows there are some ISO_Codes that are more than the
# these are related to OWID codes for regional aggregations of country data,
# convert to strin and find the OWID character pattern or the pattern is not

#find_non_ISO_codes = re.complie(r')
#re.search(r"'OWID",country_iso_list)
```

In [20]:
```
# drop the "OWID_" records

# how many records have the OWID ISO_Code?
covid_data_owid = covid_data[covid_data["iso_code"].str.contains("OWID")] #OW
print("OWID data shape: "+str(covid_data_owid.shape))

covid_data = covid_data[covid_data["iso_code"].str.contains("OWID")==False] #
print("Non OWID data shape: "+str(covid_data.shape))
```

```
OWID data shape: (13739, 15)
Non OWID data shape: (203545, 15)
```

We can now see much fewer record and I have kept a copy of the OWID aggregate data in case there is time to look at this data further

In [21]:
```
# list the country ISO Codes again
Country_lst_covid_1 = covid_data["iso_code"].tolist()

#re-run the unique records; OWID records are no longer displayed
country_ISO_list = unique(Country_lst_covid_1)
```

```
ABW AFG AGO AIA ALB AND ARE ARG ARM ATG AUS AUT AZE BDI BEL BEN BES BFA BGD
BGR BHR BHS BIH BLR BLZ BMU BOL BRA BRB BRN BTN BWA CAF CAN CHE CHL CHN CIV
CMR COD COG COK COL COM CPV CRI CUB CUW CYM CYP CZE DEU DJI DMA DNK DOM DZA
ECU EGY ERI ESH ESP EST ETH FIN FJI FLK FRA FRO FSM GAB GBR GEO GGY GHA GIB
GIN GMB GNB GNQ GRC GRD GRL GTM GUM GUY HKG HND HRV HTI HUN IDN IMN IND IRL
IRN IRQ ISL ISR ITA JAM JEY JOR JPN KAZ KEN KGZ KHM KIR KNA KOR KWT LAO LBN
LBR LBY LCA LIE LKA LSO LTU LUX LVA MAC MAR MCO MDA MDG MDV MEX MHL MKD MLI
MLT MMR MNE MNG MNP MOZ MRT MSR MUS MWI MYS NAM NCL NER NGA NIC NIU NLD NOR
NPL NRU NZL OMN PAK PAN PCN PER PHL PLW PNG POL PRI PRK PRT PRY PSE PYF QAT
ROU RUS RWA SAU SDN SEN SGP SHN SLB SLE SLV SMR SOM SPM SRB SSD STP SUR SVK
SVN SWE SWZ SXM SYC SYR TCA TCD TGO THA TJK TKL TKM TLS TON TTO TUN TUR TUV
TWN TZA UGA UKR URY USA UZB VAT VCT VEN VGB VIR VNM VUT WLF WSM YEM ZAF ZMB
ZWE
```

this look better the 3 digit codes line up nicely and all look correct now

In [22]:
```
# show the column headers and the number of columns

df_head = covid_data

columns_len = df_head.shape[1] # count the number of columns in the list

#print(name_obj(df_head,"The headers from the ", " DataFrame are:"))
#column_headers_list(df_head)
```

In [23]:
```
# print a summary of covid_data

# Summary of covid_data_raw file
rows = covid_data.shape[0]
cols = covid_data.shape[1]

print("The raw data file has {} rows of data".format(f"{rows:,d}"),"and {} co
if cols>column_count_limit:
    print("There are many columns, drop a some of them to imporve performance
else:
    print("Number of columns appears manageable")
```

```
The raw data file has 203,545 rows of data and 15 columns
Number of columns appears manageable
```

lets deal with the nulls

In [24]:
```
covid_data.head(5)
```

Out[24]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_patien
9371	ABW	Aruba	2020-03-13	2.0	2.0	NaN	NaN	Na
9372	ABW	Aruba	2020-03-14	2.0	0.0	NaN	NaN	Na
9373	ABW	Aruba	2020-03-15	2.0	0.0	NaN	NaN	Na
9374	ABW	Aruba	2020-03-16	2.0	0.0	NaN	NaN	Na
9375	ABW	Aruba	2020-03-17	3.0	1.0	NaN	NaN	Na

In [25]:
```
covid_data.tail(5)
```

Out[25]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_pa
217279	ZWE	Zimbabwe	2022-09-13	256904.0	16.0	5596.0	0.0	
217280	ZWE	Zimbabwe	2022-09-14	256939.0	35.0	5596.0	0.0	
217281	ZWE	Zimbabwe	2022-09-15	256939.0	0.0	5596.0	0.0	
217282	ZWE	Zimbabwe	2022-09-16	256939.0	0.0	5596.0	0.0	
217283	ZWE	Zimbabwe	2022-09-17	256988.0	49.0	5598.0	2.0	

In [26]:

```
# Review Null data
# pd.set_option('display.max_rows',None)
print("Null data:")
print(covid_data.isna().sum())
```

```
Null data:
iso_code                        0
location                        0
date                            0
total_cases                  8586
new_cases                    8883
total_deaths                27444
new_deaths                  27697
icu_patients               175861
total_tests                124346
total_vaccinations         150753
people_fully_vaccinated    155998
population                      0
population_density          12489
gdp_per_capita              27574
extreme_poverty             89406
dtype: int64
```

review of the columns and null data shows iso_code, loaction (country), date, population are fully populated

population density is not populated for everything, will need to confirm that for the selected date that his is improved

total_cases, new_cases, total_deaths, new_deaths etc I would expect null data as data would not be available for all countries from the start of the data period, need to convert these to 0's

need to review "gdp_per_capita" and "population_density" data as that could impact report later on

In [27]:

```
#Which fields have nan's
covid_data[covid_data.isna().any(axis=1)]
```

9374	ABW	Aruba	2020-03-16	2.0	0.0	NaN	NaN
9375	ABW	Aruba	2020-03-17	3.0	1.0	NaN	NaN
...	...	...	...	...	...	...	...
217279	ZWE	Zimbabwe	2022-09-13	256904.0	16.0	5596.0	0.0
217280	ZWE	Zimbabwe	2022-09-14	256939.0	35.0	5596.0	0.0
217281	ZWE	Zimbabwe	2022-09-15	256939.0	0.0	5596.0	0.0
217282	ZWE	Zimbabwe	2022-09-16	256939.0	0.0	5596.0	0.0
217283	ZWE	Zimbabwe	2022-09-17	256988.0	49.0	5598.0	2.0

In [28]:

```
# review "gdp_per_capita" and "population_density"

covid_data_gdp_pop = covid_data[["date","iso_code","location","population","p

covid_data_gdp_pop["population"].isnull()

#covid_data_gdp_pop[covid_data_gdp_pop.isna().any(axis=1)]
```

Out[28]:

```
9371      False
9372      False
9373      False
9374      False
9375      False
          ...
217279    False
217280    False
217281    False
217282    False
217283    False
Name: population, Length: 203545, dtype: bool
```

as i only plan on using certain dates, specifically year end and the most recent for the current year, dropping these NaN records should not impact the report too much

In [29]:

```
# drop the NaN for the population and gdp columns
covid_data.dropna(subset=["population","population_density","gdp_per_capita"]
#https://www.datasciencelearner.com/pandas-dropna-remove-nan-rows-python/
```

Out[29]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_
9371	ABW	Aruba	2020-03-13	2.0	2.0	NaN	NaN	
9372	ABW	Aruba	2020-03-14	2.0	0.0	NaN	NaN	
9373	ABW	Aruba	2020-03-15	2.0	0.0	NaN	NaN	
9374	ABW	Aruba	2020-03-16	2.0	0.0	NaN	NaN	
9375	ABW	Aruba	2020-03-17	3.0	1.0	NaN	NaN	
...	...	...	...	...	...	...	...	
217279	ZWE	Zimbabwe	2022-09-13	256904.0	16.0	5596.0	0.0	

next look at "total" columns that have NaNs that should really be 0s. Many of these NaNs are form the earlier dates when there weren't many cases

In [30]:

```
# change the NaN's to 0 for the remainder valuations
covid_data = covid_data.fillna(0)
```

In [31]:

```
covid_data
```

Out[31]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_
9371	ABW	Aruba	2020-03-13	2.0	2.0	0.0	0.0	
9372	ABW	Aruba	2020-03-14	2.0	0.0	0.0	0.0	
9373	ABW	Aruba	2020-03-15	2.0	0.0	0.0	0.0	
9374	ABW	Aruba	2020-03-16	2.0	0.0	0.0	0.0	
9375	ABW	Aruba	2020-03-17	3.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	
217279	ZWE	Zimbabwe	2022-09-13	256904.0	16.0	5596.0	0.0	

Summary of the COVID dataset

In [32]:

```python
# How many records am I dealing with

#total_records = covid_data.count(axis=1)
#print(total_records)
#print("")

# show the countries/ locations in the data
print("ISO codes and Country")
print(covid_data.pivot_table(index = ["iso_code", "location"], aggfunc ="size

print("")
# df.size
print("Size:")
print(covid_data.size)

print("")
# df.isnull()
column_picker ="total_deaths"
covid_ttl_deaths = covid_data.filter(["iso_code", "location",column_picker])
bool_series_null =pd.isnull(covid_ttl_deaths[column_picker])

print("Null",column_picker,": ")
print(covid_ttl_deaths[bool_series_null])
#print(covid_data.isnull())

print("")
# df.notnull()
bool_series = pd.notnull(covid_ttl_deaths[column_picker])
print("Not null:")
print(covid_ttl_deaths[bool_series])

print("")
# df.describe()
print("Describe:")
print(covid_data.describe)
```

```
Length: 229, dtype: int64

Size:
3053175

Null total_deaths :
Empty DataFrame
Columns: [iso_code, location, total_deaths]
Index: []

Not null:
       iso_code  location  total_deaths
9371        ABW     Aruba           0.0
9372        ABW     Aruba           0.0
9373        ABW     Aruba           0.0
9374        ABW     Aruba           0.0
9375        ABW     Aruba           0.0
...         ...       ...           ...
217279      ZWE  Zimbabwe        5596.0
217280      ZWE  Zimbabwe        5596.0
```

Looks much better there are now now nulls for the Ttoa_deaths which will allow caculations to be performed

GDP Raw Data

In [33]:

```
# review GDP data

#look at info for gdp data
print("shape of the GDP raw data:")
print(gdp_data_raw.shape)
print()
```

```
shape of the GDP raw data:
(266, 67)
```

drop many of the year columns as the covid data does not go back that far

In [34]:

```python
# do not need most of the columns so will remove cols 4:63
gdp_data = gdp_data_raw.drop(gdp_data_raw.iloc[:,4:63],axis = 1)

#gdp_data = gdp_data_1.drop(gdp_data_raw.iloc[:,7],axis = 1)

#convert the spaces " " to underscore "_" consitent with the COVID data
gdp_data.columns = [c.replace(' ', '_') for c in gdp_data.columns]

print(gdp_data.info())
print()
print(gdp_data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country_Name    266 non-null    object 
 1   Country_Code    266 non-null    object 
 2   Indicator_Name  266 non-null    object 
 3   Indicator_Code  266 non-null    object 
 4   2019            255 non-null    float64
 5   2020            251 non-null    float64
 6   2021            229 non-null    float64
 7   Unnamed:_66     0 non-null      float64
dtypes: float64(4), object(4)
memory usage: 16.8+ KB
None

                 Country_Name Country_Code     Indicator_Name  \
0                       Aruba          ABW  GDP (current US$)   
1 Africa Eastern and Southern          AFE  GDP (current US$)   
2                 Afghanistan          AFG  GDP (current US$)   
3  Africa Western and Central          AFW  GDP (current US$)   
4                      Angola          AGO  GDP (current US$)   

  Indicator_Code          2019          2020          2021  Unnamed:_66  
0 NY.GDP.MKTP.CD  3.310056e+09  2.496648e+09           NaN          NaN  
1 NY.GDP.MKTP.CD  9.975340e+11  9.216459e+11  1.082096e+12          NaN  
2 NY.GDP.MKTP.CD  1.879945e+10  2.011614e+10           NaN          NaN  
3 NY.GDP.MKTP.CD  7.945430e+11  7.844457e+11  8.358084e+11          NaN  
4 NY.GDP.MKTP.CD  6.930910e+10  5.361907e+10  7.254699e+10          NaN  
```

In [35]:

```
# show the column headers and the number of columns

columns_len = gdp_data.shape[1] # count the number of columns in the list
gdp_data_name = name_obj(gdp_data,"The headers from the ", " Dataframe are:")

print(gdp_data_name)
columns_lst_gdp = column_headers_list(gdp_data)
print(columns_lst_gdp)

print()

print(name_obj(gdp_data,"The "," DataFrame has "+str(columns_len)+" columns")

print()

#print(columns_comment())
```

```
The headers from the gdp_data Dataframe are:
['Country_Name', 'Country_Code', 'Indicator_Name', 'Indicator_Code', '201
9', '2020', '2021', 'Unnamed:_66']

The gdp_data DataFrame has 8 columns
```

GDP data looks much better and is ready if I need it

In [36]:

```
# Correlations of the gdp_data
gdp_data.corr()
```

Out[36]:

	2019	2020	2021	Unnamed:_66
2019	1.000000	0.999882	0.99950	NaN
2020	0.999882	1.000000	0.99963	NaN
2021	0.999500	0.999630	1.00000	NaN
Unnamed:_66	NaN	NaN	NaN	NaN

Summary of data

In [37]:

```
# Summary of Covid data
print("summary of Covid data")
print()

# Number of unique countries
n = covid_data.iso_code.nunique()
print("No of unique countries (covid_data):",n)

print("")

# Number of unique dates
n = covid_data.date.nunique()



print("No of unique dates: ",n)
print("From: ",beg_date.strftime("%b %d %Y")," to: ",end_date.strftime("%b %c
print("")

# Number of records
rec = covid_data.shape[0]
col = covid_data.shape[1]

print("No of rows: ",f"{rec:,d}")
print("No of columns: ",f"{col:,d}")
#source: https://stackoverflow.com/questions/60934535/format-integer-with-com
print("")
```

```
summary of Covid data

No of unique countries (covid_data): 229

No of unique dates:  992
From:  Jan 01 2020  to:  Sep 18 2022

No of rows:  203,545
No of columns:  15
```

In [38]:

```
# Summary of GDP data

print("summary of GDP Data")
print()


# Number of unique countries
n = gdp_data.Country_Code.nunique()
print("No of unique countries: ",n)
print("")

# Number of records
rec = gdp_data.shape[0]
col = gdp_data.shape[1]

print("No of rows: ",f"{rec:,d}")
print("No of columns: ",f"{col:,d}")
```

```
summary of GDP Data

No of unique countries:  266

No of rows:  266
No of columns:  8
```

calculations for reporting

In [39]:

```
# Calculations for report:

# date calculations
# There needs to be a n_day (number of days) total for certain total columns
# comparative data against 100k of a countries population
```

to create a rolling ndays average per 100k, I need to identify the date and go back 14days unless that date is with in 14days of the start of the dataset

In [40]:

```
# n day calculations can't begin until the nth day after the first date in th
first_calc_date = beg_date +  timedelta(days=days_calc)
print("Begining Date: "+str(beg_date)+"; Earliest starting date for calculati

# calculate the start date for the n days data for each record
n_day_start = covid_data["date"] - timedelta(days=days_calc)
print()
print("show that the dates are populating with different results")
print(n_day_start)
print("its working")
print()

# Insert a column with the n day start date, this shows when the n days rolli
covid_data.insert(loc=3, column="n_day_start_date", value=n_day_start, allow_
#false will not allow the column to be entered more than once
```

```
Begining Date: 2020-01-01 00:00:00; Earliest starting date for calculation
s: 2020-01-15 00:00:00

show that the dates are populating with different results
9371     2020-02-28
9372     2020-02-29
9373     2020-03-01
9374     2020-03-02
9375     2020-03-03
            ...
217279   2022-08-30
217280   2022-08-31
217281   2022-09-01
217282   2022-09-02
217283   2022-09-03
Name: date, Length: 203545, dtype: datetime64[ns]
its working
```

In [41]:

```
print(covid_data.info())
print("the new column is now appearing")
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 203545 entries, 9371 to 217283
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   iso_code                 203545 non-null  object
 1   location                 203545 non-null  object
 2   date                     203545 non-null  datetime64[ns]
 3   n_day_start_date         203545 non-null  datetime64[ns]
 4   total_cases              203545 non-null  float64
 5   new_cases                203545 non-null  float64
 6   total_deaths             203545 non-null  float64
 7   new_deaths               203545 non-null  float64
 8   icu_patients             203545 non-null  float64
 9   total_tests              203545 non-null  float64
 10  total_vaccinations       203545 non-null  float64
 11  people_fully_vaccinated  203545 non-null  float64
 12  population               203545 non-null  float64
 13  population_density       203545 non-null  float64
```

create the n_rolling days functions and insert into the DataFrame

In [42]:

```
# n days totals
# (https://stackoverflow.com/questions/28236305/how-do-i-sum-values-in-a-colu
# https://python.tutorialink.com/calculate-14-day-rolling-average-on-data-wit

covid_data.sort_values(['iso_code','date'], ascending=(True,True), inplace=Tr

# Rolling new cases
rolling_new_cases = covid_data.groupby(['iso_code'])['new_cases'].transform(l

# Insert a column with the "n" rolling new cases

#new_column string name
new_column = str(days_calc)+"_days_rolling_new_cases"
print(new_column)

print(new_column in covid_data.columns) # Test for existing column# True

#  delete new column, use if re-runing with out resetting the data,
#if time allows will create if statement to check if column is available then
#del covid_data[str(days_calc)+"_days_rolling_new_cases"]

# insert new_column
covid_data.insert(loc=6, column=str(days_calc)+"_days_rolling_new_cases", val

print("-"*100)

# Rolling new deaths
rolling_new_deaths = covid_data.groupby(['iso_code'])['new_deaths'].transform


# Insert a column with the "n" rolling new deaths

#new_column string name
new_column_2 = str(days_calc)+"_days_rolling_new_deaths"
print(new_column_2)

print(new_column_2 in covid_data.columns) # Test for existing column# True

# delete new column
#del covid_data[str(days_calc)+"_days_rolling_new_deaths"]

# insert new_column
covid_data.insert(loc=9, column=str(days_calc)+"_days_rolling_new_deaths", va

print("-"*100)


#repeat for new deaths
#still need to create calculations for:
    #total_cases_per_100k per 100k of the population (total_cases/population
    #total_deaths_per_100k of the population (total_deaths/population * 100,0
    #total_cases_per_100sqkm of the country (total_cases/total country sqkm *
    #total_deaths_per_100sqkm of the country (total_deaths/total country sqkm
#these will be used to use machine learning to establish if the GDP or pop de
#merge in the gdp data if required
```

```
14_days_rolling_new_cases
False
----------------------------------------------------------------------------
----------------------------
14_days_rolling_new_deaths
False
----------------------------------------------------------------------------
----------------------------
```

make the totals 100k of the population so we can compare countries if need be

In [43]:

```
# Cases per 100K of population
total_cases_per_100k = covid_data.total_cases/covid_data.population * pop_per

# Insert a column total cases per 100k

#new_column string name
new_column_3 = "total_cases_per_100k"
print(new_column_3)

print(new_column_3 in covid_data.columns) # Test for existing column# True

#  delete new column
#del covid_data["total_cases_per_100k"]

# insert new_column
covid_data.insert(loc=6, column="total_cases_per_100k", value=total_cases_per

print("-"*100)
```

```
total_cases_per_100k
False
------------------------------------------------------------------------------
-------------------------
```

In [44]:

```
# deaths per 100K of population
total_deaths_per_100k = covid_data.total_deaths/covid_data.population * pop_p

# Insert a column total deaths per 100k

#new_column string name
new_column_4 = "total_deaths_per_100k"
print(new_column_4)

print(new_column_4 in covid_data.columns) # Test for existing column# True

#  delete new column
#del covid_data["total_deaths_per_100k"]

# insert new_column
covid_data.insert(loc=6, column="total_deaths_per_100k", value=total_deaths_p

print("-"*100)
```

```
total_deaths_per_100k
False
---------------------------------------------------------------------------
-------------------------
```

to be able to group for regressions create a subset of the data to look at

Take the data for year end for 2020,2021 and the most recent data from 2022 review this data for and create the top deaths and the lowest deaths sets of data in the top deaths look at the minimum value to set the threshold for the "HigH" mortalility classificaion in the bottom deaths look at the max value to set the threshold for the "Low" mortalility classificaion

In [45]:

```
# create a classification for mortality if the total_deaths per 100k is high
# obesrvations of the deaths for 2020 and 2021, get the min value of the top
#covid_data[covid_data["date"].isin(["2020-12-31","2021-12-31","2022-09-15"])

#last_date_n = str(last_date_n = str(last_date))

# filter data for the dates:
covid_data_observe = covid_data[covid_data["date"].isin(["2020-12-31","2021-1
covid_data_observe_20_21 =covid_data[covid_data["date"].isin(["2020-12-31","2
covid_data_observe_22 =covid_data[covid_data["date"].isin([last_date_n])]

#top and bottom observations
top_20_21 = covid_data_observe_20_21.nlargest(n=top_n_parameter, columns=["to
bot_22 = covid_data_observe_22.nsmallest(n=top_n_parameter, columns=["total_d

# min value in the Top mortality (top deaths) data
print("min of 2020/21 top deaths/ 100k: "+str(top_20_21["total_deaths_per_100
print("max of 2020/21 top deaths/ 100k: "+str(top_20_21["total_deaths_per_100
print("these records look very high, it could be due to an outlier, I have re
print("-"*100)

# max value in the bottom mortality (bottom deaths) data
print("max of 2022 lowest deaths/ 100k: "+str(bot_22["total_deaths_per_100k"]
print("min of 2022 lowest deaths/ 100k: "+str(bot_22["total_deaths_per_100k"]
```

```
min of 2020/21 top deaths/ 100k: 308.80746576160647
max of 2020/21 top deaths/ 100k: 601.1779992283662
these records look very high, it could be due to an outlier, I have recalcu
lted below again once more of the data is cleaned
---------------------------------------------------------------------------
-------------------------
max of 2022 lowest deaths/ 100k: 1.235510373891575
min of 2022 lowest deaths/ 100k: 0.0
```

In [46]:

```
# classifiers for deaths beig high should be above 50 per 100k and below 10 p
print("high_deaths ="+str(high_deaths_per_100k))
print("low_deaths ="+str(low_deaths_per_100k))
# set these variables at the top of the project
```

```
high_deaths =50
low_deaths =10
```

add this classification to the covid_data

In [47]:

```
# create a calculation to insert the classification group of "Low" (10,25,50)
#if the total_deaths_per_100k >= 50 then "High Deaths" elseif total_deaths_pe

covid_data.loc[covid_data["total_deaths_per_100k"] <= low_deaths_per_100k, "m
covid_data.loc[covid_data["total_deaths_per_100k"] < high_deaths_per_100k, "m
covid_data.loc[covid_data["total_deaths_per_100k"] >= high_deaths_per_100k, "

covid_data.tail()
```

Out[47]:

	iso_code	location	date	n_day_start_date	total_cases	new_cases	total_deaths_pe
217279	ZWE	Zimbabwe	2022-09-13	2022-08-30	256904.0	16.0	34.
217280	ZWE	Zimbabwe	2022-09-14	2022-08-31	256939.0	35.0	34.
217281	ZWE	Zimbabwe	2022-09-15	2022-09-01	256939.0	0.0	34.
217282	ZWE	Zimbabwe	2022-09-16	2022-09-02	256939.0	0.0	34.
217283	ZWE	Zimbabwe	2022-09-17	2022-09-03	256988.0	49.0	35.

5 rows × 21 columns

It can be observerd from the .tail function that observations between 10 to 50 will be blank

In [48]:

```
# Review covid data using the iso code for a country
covid_data_country = covid_data[covid_data["iso_code"]=="ABW"]

covid_data_country.head(20)
```

Out[48]:

	iso_code	location	date	n_day_start_date	total_cases	new_cases	total_deaths_per
9371	ABW	Aruba	2020-03-13	2020-02-28	2.0	2.0	
9372	ABW	Aruba	2020-03-14	2020-02-29	2.0	0.0	
9373	ABW	Aruba	2020-03-15	2020-03-01	2.0	0.0	
9374	ABW	Aruba	2020-03-16	2020-03-02	2.0	0.0	
9375	ABW	Aruba	2020-03-17	2020-03-03	3.0	1.0	
9376	ABW	Aruba	2020-03-18	2020-03-04	4.0	1.0	
9377	ABW	Aruba	2020-03-19	2020-03-05	4.0	0.0	

In [49]:

```
covid_data_country.tail(20)
```

Out[49]:

	iso_code	location	date	n_day_start_date	total_cases	new_cases	total_deaths_pe
10270	ABW	Aruba	2022-08-29	2022-08-15	42792.0	42.0	213.
10271	ABW	Aruba	2022-08-30	2022-08-16	42792.0	0.0	213.
10272	ABW	Aruba	2022-08-31	2022-08-17	42848.0	56.0	213.
10273	ABW	Aruba	2022-09-01	2022-08-18	42848.0	0.0	213.
10274	ABW	Aruba	2022-09-02	2022-08-19	42848.0	0.0	213.
10275	ABW	Aruba	2022-09-03	2022-08-20	42848.0	0.0	213.
10276	ABW	Aruba	2022-09-04	2022-08-21	42848.0	0.0	213.

we can observe that the earlier data in the .head() function shows empty mortality data but the later .tail() "high"'s can be seen

In [50]:
```
print(covid_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 203545 entries, 9371 to 217283
Data columns (total 21 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   iso_code                     203545 non-null  object
 1   location                     203545 non-null  object
 2   date                         203545 non-null  datetime64[ns]
 3   n_day_start_date             203545 non-null  datetime64[ns]
 4   total_cases                  203545 non-null  float64
 5   new_cases                    203545 non-null  float64
 6   total_deaths_per_100k        203545 non-null  float64
 7   total_cases_per_100k         203545 non-null  float64
 8   14_days_rolling_new_cases    203545 non-null  float64
 9   total_deaths                 203545 non-null  float64
 10  new_deaths                   203545 non-null  float64
 11  14_days_rolling_new_deaths   203545 non-null  float64
 12  icu_patients                 203545 non-null  float64
 13  total_tests                  203545 non-null  float64
 14  total_vaccinations           203545 non-null  float64
 15  people_fully_vaccinated      203545 non-null  float64
 16  population                   203545 non-null  float64
 17  population_density           203545 non-null  float64
 18  gdp_per_capita               203545 non-null  float64
 19  extreme_poverty              203545 non-null  float64
 20  mortality                    203545 non-null  object
dtypes: datetime64[ns](2), float64(16), object(3)
memory usage: 34.2+ MB
None
```

all the new columns are showing up in the datset now. Let's review the .corr() function for some quick insights

In [51]:

```
# Correlations of the covid_data
covid_data.corr()
```

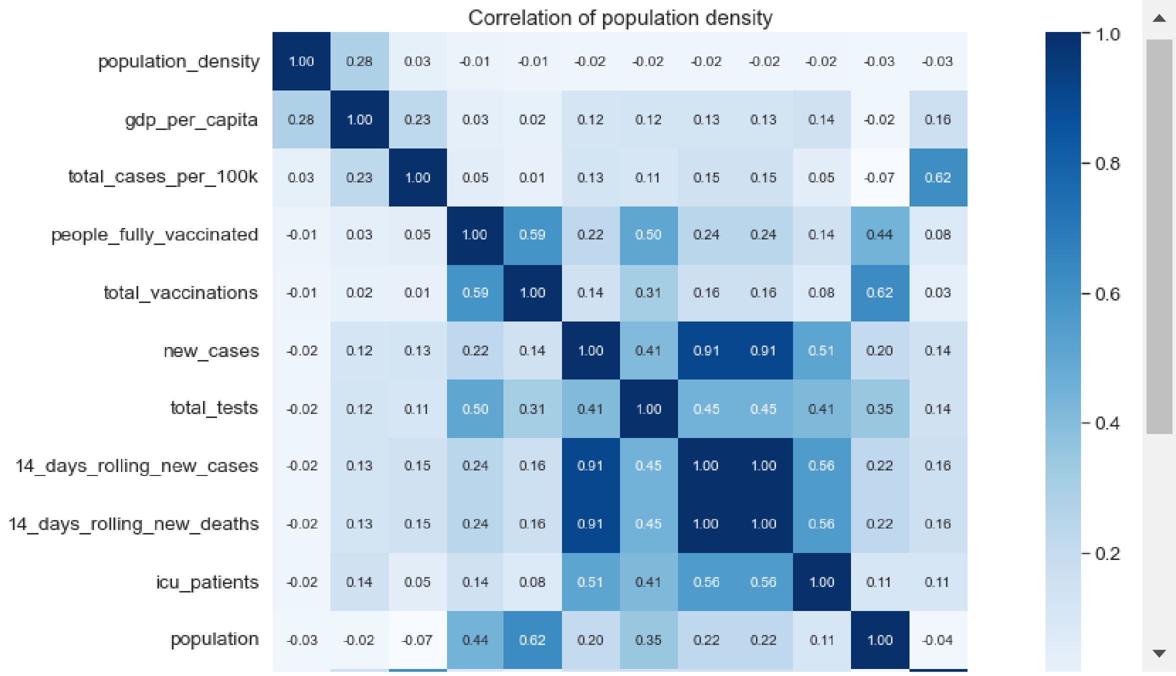
Out[51]:

	total_cases	new_cases	total_deaths_per_100k	total_cases_per_10
total_cases	1.000000	0.511007	0.274637	0.2282
new_cases	0.511007	1.000000	0.142463	0.1273
total_deaths_per_100k	0.274637	0.142463	1.000000	0.6158
total_cases_per_100k	0.228270	0.127374	0.615828	1.0000
14_days_rolling_new_cases	0.571959	0.905718	0.157589	0.1473
total_deaths	0.897945	0.455584	0.352094	0.1332
new_deaths	0.418564	0.556613	0.136376	0.0174
14_days_rolling_new_deaths	0.571959	0.905718	0.157589	0.1473
icu_patients	0.421160	0.510926	0.110637	0.0482
total_tests	0.653270	0.409020	0.143523	0.1071
total_vaccinations	0.351515	0.144864	0.029522	0.0131
people_fully_vaccinated	0.566829	0.223022	0.079996	0.0455
population	0.345073	0.204812	-0.038970	-0.0683
population_density	-0.026215	-0.017614	-0.026077	0.0250
gdp_per_capita	0.134295	0.117210	0.157560	0.2273
extreme_poverty	-0.050959	-0.048274	-0.216721	-0.2143

this does not look very promising but it's hard to read. Let's review as a heat map

In [52]:

```
#Correlation of population density
corr = covid_data.corr()
plt.figure(figsize=(20, 9))
k = 12 #number of variables for heatmap
cols = corr.nlargest(k, 'population_density')['population_density'].index
cm = np.corrcoef(covid_data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws
plt.title("Correlation of population density", size = 15)
plt.show()
```

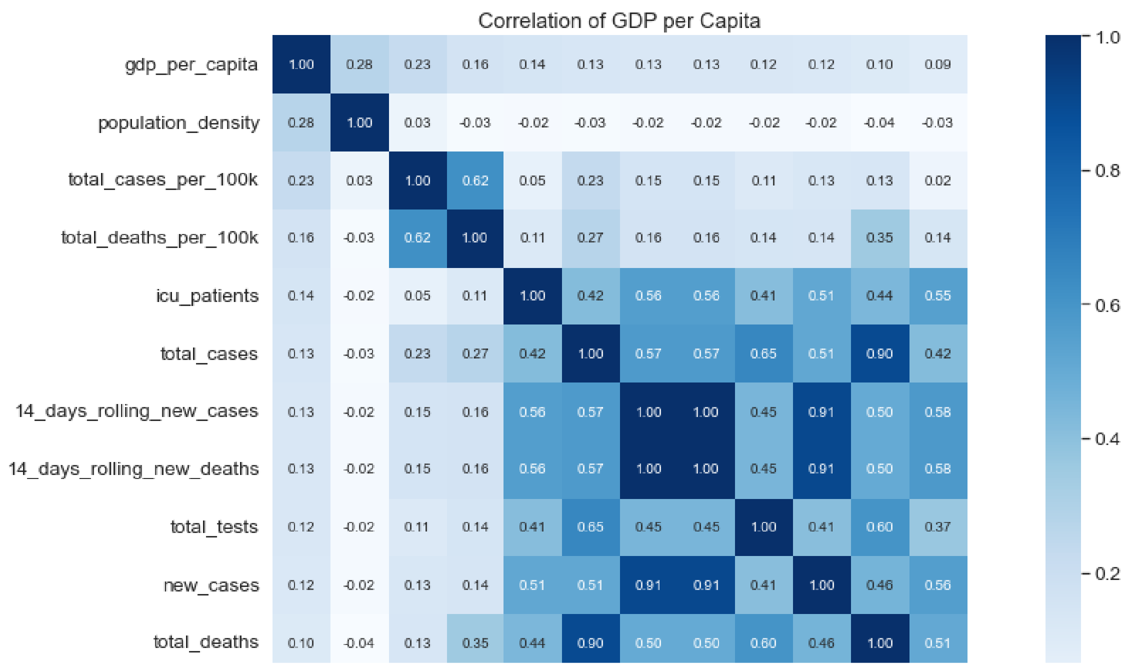


population_density	1.00	0.28	0.03	-0.01	-0.01	-0.02	-0.02	-0.02	-0.02	-0.02	-0.03	-0.03
gdp_per_capita	0.28	1.00	0.23	0.03	0.02	0.12	0.12	0.13	0.13	0.14	-0.02	0.16
total_cases_per_100k	0.03	0.23	1.00	0.05	0.01	0.13	0.11	0.15	0.15	0.05	-0.07	0.62
people_fully_vaccinated	-0.01	0.03	0.05	1.00	0.59	0.22	0.50	0.24	0.24	0.14	0.44	0.08
total_vaccinations	-0.01	0.02	0.01	0.59	1.00	0.14	0.31	0.16	0.16	0.08	0.62	0.03
new_cases	-0.02	0.12	0.13	0.22	0.14	1.00	0.41	0.91	0.91	0.51	0.20	0.14
total_tests	-0.02	0.12	0.11	0.50	0.31	0.41	1.00	0.45	0.45	0.41	0.35	0.14
14_days_rolling_new_cases	-0.02	0.13	0.15	0.24	0.16	0.91	0.45	1.00	1.00	0.56	0.22	0.16
14_days_rolling_new_deaths	-0.02	0.13	0.15	0.24	0.16	0.91	0.45	1.00	1.00	0.56	0.22	0.16
icu_patients	-0.02	0.14	0.05	0.14	0.08	0.51	0.41	0.56	0.56	1.00	0.11	0.11
population	-0.03	-0.02	-0.07	0.44	0.62	0.20	0.35	0.22	0.22	0.11	1.00	-0.04

Type *Markdown* and LaTeX: α^2

In [53]:

```
#Correlation of gdp
corr = covid_data.corr()
plt.figure(figsize=(20, 9))
k = 12 #number of variables for heatmap
cols = corr.nlargest(k, 'gdp_per_capita')['gdp_per_capita'].index
cm = np.corrcoef(covid_data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws
plt.title("Correlation of GDP per Capita", size = 15)
plt.show()
```



darker shadeing represent positive correlation. from this we can infer that population density and gdp are not correlated to the mortality rate of a country. gdp appears to have slightly better correllation than the population density

In [54]:

```
# not sure if we need some sort of index key to view the data, I created one
pk = covid_data["iso_code"]+str(covid_data['date'])

print(pk.head())

#insert pk into covid_data
#del covid_data["pk"] #delete pk column
#covid_data.insert(0, 'pk', pk)


#covid_data["pk"]
#covid_data.info()
```

```
9371    ABW9371     2020-03-13\n9372     2020-03-14\n9...
9372    ABW9371     2020-03-13\n9372     2020-03-14\n9...
9373    ABW9371     2020-03-13\n9372     2020-03-14\n9...
9374    ABW9371     2020-03-13\n9372     2020-03-14\n9...
9375    ABW9371     2020-03-13\n9372     2020-03-14\n9...
Name: iso_code, dtype: object
```

summary of data exploration and preparation: the datre is ready for anlyis but my confidence level is not high after reviewing the .corr() results. I decided to leave the gdp data source at this point as the COVID gdp per capita looks to be a good representation of the data. next step is to pefrom regession and machine learning although given the little correlation I am seeing not sure how fruitful it will be

Analysis

Basic Charts

In [55]:

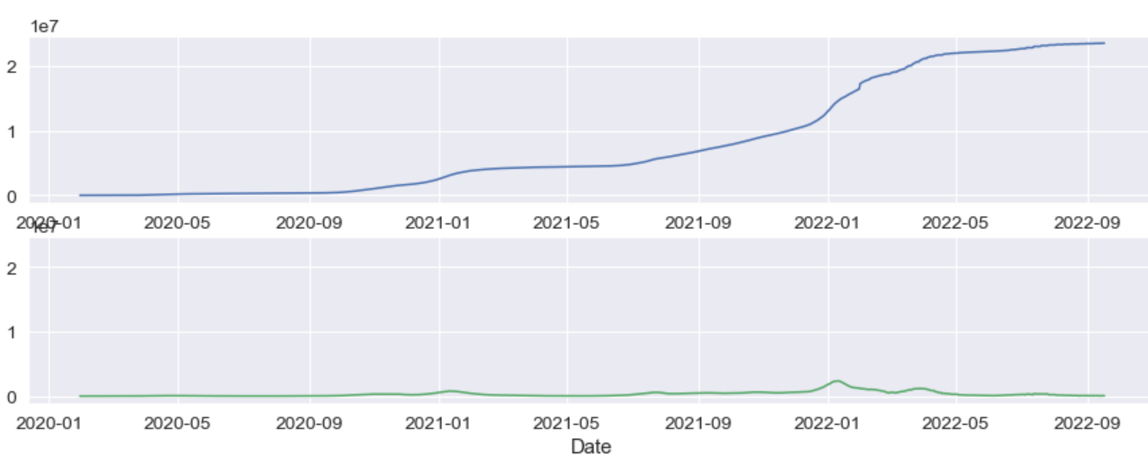
```
# Set chart sizes to wide

plt.rcParams['figure.figsize'] = [15, 5]

#list of iso codes
Country_review = covid_data["iso_code"].tolist()

# list the country codes
country_iso_list = unique(Country_review)
```

```
ABW AFG AGO AIA ALB AND ARE ARG ARM ATG AUS AUT AZE BDI BEL BEN BES BFA BGD
BGR BHR BHS BIH BLR BLZ BMU BOL BRA BRB BRN BTN BWA CAF CAN CHE CHL CHN CIV
CMR COD COG COK COL COM CPV CRI CUB CUW CYM CYP CZE DEU DJI DMA DNK DOM DZA
ECU EGY ERI ESH ESP EST ETH FIN FJI FLK FRA FRO FSM GAB GBR GEO GGY GHA GIB
GIN GMB GNB GNQ GRC GRD GRL GTM GUM GUY HKG HND HRV HTI HUN IDN IMN IND IRL
IRN IRQ ISL ISR ITA JAM JEY JOR JPN KAZ KEN KGZ KHM KIR KNA KOR KWT LAO LBN
LBR LBY LCA LIE LKA LSO LTU LUX LVA MAC MAR MCO MDA MDG MDV MEX MHL MKD MLI
MLT MMR MNE MNG MNP MOZ MRT MSR MUS MWI MYS NAM NCL NER NGA NIC NIU NLD NOR
NPL NRU NZL OMN PAK PAN PCN PER PHL PLW PNG POL PRI PRK PRT PRY PSE PYF QAT
ROU RUS RWA SAU SDN SEN SGP SHN SLB SLE SLV SMR SOM SPM SRB SSD STP SUR SVK
SVN SWE SWZ SXM SYC SYR TCA TCD TGO THA TJK TKL TKM TLS TON TTO TUN TUR TUV
TWN TZA UGA UKR URY USA UZB VAT VCT VEN VGB VIR VNM VUT WLF WSM YEM ZAF ZMB
ZWE
```

In [56]:

```
country_use_code = "GBR"

#covid_data = covid_data.set_index("pk")

iso_use = country_use_code

print("Total Cases vs "+new_column+" for: "+iso_use)


# Basic plot of total covid cases over time
fig, ax=plt.subplots(2,1, sharey=True)


data=covid_data[covid_data["iso_code"]==iso_use]
data1=covid_data[covid_data["iso_code"]==iso_use]

ax[0].plot(data["date"], data["total_cases"], color='b')
print()
ax[1].plot(data["date"], data[new_column], color='g') #using the new_column f

ax[1].set_xlabel("Date")

plt.show()
```

```
Total Cases vs 14_days_rolling_new_cases for: GBR
```



the charts above give an indication of the total mortality per 100k people and rolling 14 day spikes representing the waves over time. we can see that the total increases sharply between the end of 2020, 2021 and is now leveling off.

analysis with seaborn

I'm going to create a subset of the data as mentioned between year end totals/100k population to compare and see if the gdp, martality calssification impacts the results

In [57]:

```
# create a subset of the COVID data for use with seaborn analysis

covid_data_small = covid_data[['date',
                               'iso_code',
                               'location',
                               'total_cases',
                               'total_cases_per_100k',
                               'total_deaths',
                               'total_deaths_per_100k',
                               'population',
                               'population_density',
                               'gdp_per_capita',
                               'extreme_poverty',
                               'people_fully_vaccinated',
                               'mortality'
                              ]]


#covid_data_small.fillna(0)
covid_data_small

last_date_n = str(last_date)

print("last date to use: "+last_date_n)
```

```
last date to use: 2022-09-16 00:00:00
```

In [58]:

```python
# filter on dates for analysis
covid_data_sns = covid_data_small[covid_data_small["date"].isin(["2020-12-31"

print("covid_data_sns.shape")
print(covid_data_sns.shape)
print("-"*100)
print()
print(covid_data_sns.head())
print("-"*100)
print(covid_data_sns.tail())
print("-"*100)
print()
print("Null data")
print(covid_data_sns.isna().sum())
print("-"*100)
print()
print(covid_data_sns.corr())
print("-"*100)
```

```
covid_data_sns.shape
(658, 13)
---------------------------------------------------------------------------
-------------------------

             date iso_code     location  total_cases  total_cases_per_100k
\
9664   2020-12-31      ABW        Aruba       5489.0           5152.249005
10029  2021-12-31      ABW        Aruba      20461.0          19205.714500
10288  2022-09-16      ABW        Aruba      42970.0          40333.783885
311    2020-12-31      AFG  Afghanistan      52330.0            130.500504
676    2021-12-31      AFG  Afghanistan     158084.0            394.229728

       total_deaths  total_deaths_per_100k  population  population_density
\
9664           49.0              45.993842    106536.0             584.800
10029         181.0             169.895622    106536.0             584.800
10288         228.0             214.012165    106536.0             584.800
311          2189.0               5.458926  40099462.0              54.422
676          7356.0              18.344386  40099462.0              54.422

       gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
9664        35973.781              0.0                      0.0
10029       35973.781              0.0                      0.0      high
10288       35973.781              0.0                  83557.0      high
311          1803.987              0.0                      0.0
676          1803.987              0.0                      0.0
---------------------------------------------------------------------------
-------------------------
              date iso_code  location  total_cases  total_cases_per_100k  \
216111  2021-12-31      ZMB    Zambia     254274.0           1305.768848
216370  2022-09-16      ZMB    Zambia     333363.0           1711.913214
216658  2020-12-31      ZWE  Zimbabwe      13867.0             86.703843
217023  2021-12-31      ZWE  Zimbabwe     213258.0           1333.402195
217282  2022-09-16      ZWE  Zimbabwe     256939.0           1606.518989

        total_deaths  total_deaths_per_100k  population  population_density
\
```

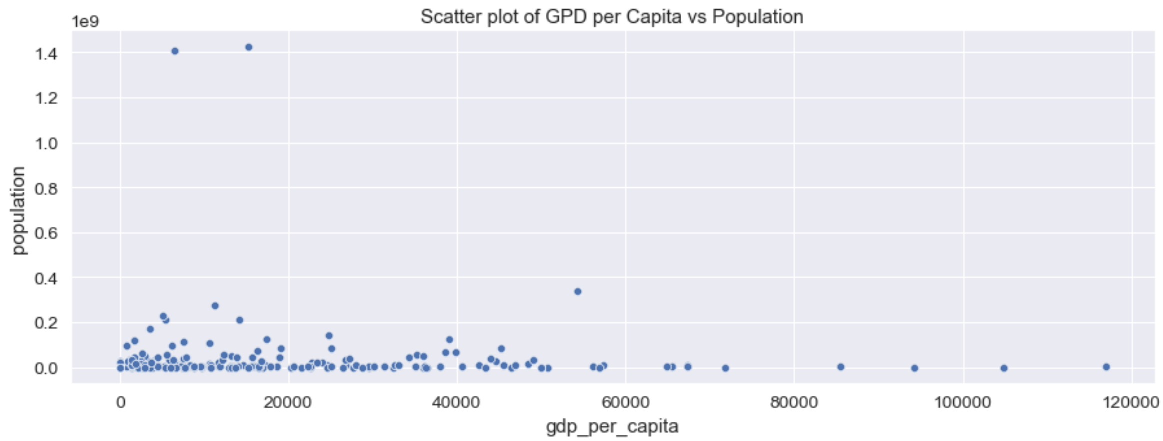
```
216111       3734.0            19.175145  19473125.0             22.995
216370       4017.0            20.628430  19473125.0             22.995
216658        363.0             2.269669  15993524.0             42.729
217023       5004.0            31.287664  15993524.0             42.729
217282       5596.0            34.989162  15993524.0             42.729

        gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
216111        3689.251             57.5                1217415.0
216370        3689.251             57.5                      0.0
216658        1899.775             21.4                      0.0
217023        1899.775             21.4                3135168.0
217282        1899.775             21.4                      0.0
---------------------------------------------------------------------------
-------------------------

Null data
date                       0
iso_code                   0
location                   0
total_cases                0
total_cases_per_100k       0
total_deaths               0
total_deaths_per_100k      0
population                 0
population_density         0
gdp_per_capita             0
extreme_poverty            0
people_fully_vaccinated    0
mortality                  0
dtype: int64
---------------------------------------------------------------------------
-------------------------

                         total_cases  total_cases_per_100k  total_deaths  \
total_cases                 1.000000              0.217462      0.886175
total_cases_per_100k        0.217462              1.000000      0.098189
total_deaths                0.886175              0.098189      1.000000
total_deaths_per_100k       0.251780              0.565010      0.323133
population                  0.371592             -0.084021      0.391657
population_density         -0.028417              0.038308     -0.038251
gdp_per_capita              0.167965              0.281003      0.116082
extreme_poverty            -0.066459             -0.265953     -0.062166
people_fully_vaccinated     0.492317             -0.002906      0.515926

                         total_deaths_per_100k  population  \
total_cases                           0.251780    0.371592
total_cases_per_100k                  0.565010   -0.084021
total_deaths                          0.323133    0.391657
total_deaths_per_100k                 1.000000   -0.048313
population                           -0.048313    1.000000
population_density                   -0.033790   -0.025052
gdp_per_capita                        0.187821   -0.022257
extreme_poverty                      -0.265442    0.028340
people_fully_vaccinated               0.044485    0.545371

                         population_density  gdp_per_capita  extreme_povert
y  \
```

```
total_cases                      -0.028417        0.167965        -0.06645
9
total_cases_per_100k              0.038308        0.281003        -0.26595
3
total_deaths                     -0.038251        0.116082        -0.06216
6
total_deaths_per_100k            -0.033790        0.187821        -0.26544
2
population                       -0.025052       -0.022257         0.02834
0
population_density                1.000000        0.269128        -0.06812
3
gdp_per_capita                    0.269128        1.000000        -0.31493
8
extreme_poverty                  -0.068123       -0.314938         1.00000
0
people_fully_vaccinated          -0.008559        0.009211         0.01489
8

                         people_fully_vaccinated
total_cases                             0.492317
total_cases_per_100k                   -0.002906
total_deaths                            0.515926
total_deaths_per_100k                   0.044485
population                              0.545371
population_density                     -0.008559
gdp_per_capita                          0.009211
extreme_poverty                         0.014898
people_fully_vaccinated                 1.000000
--------------------------------------------------------------------------
-------------------------
```

I can see that we are getting multiple dates and requested columns of data back so good to move forward

In [59]:

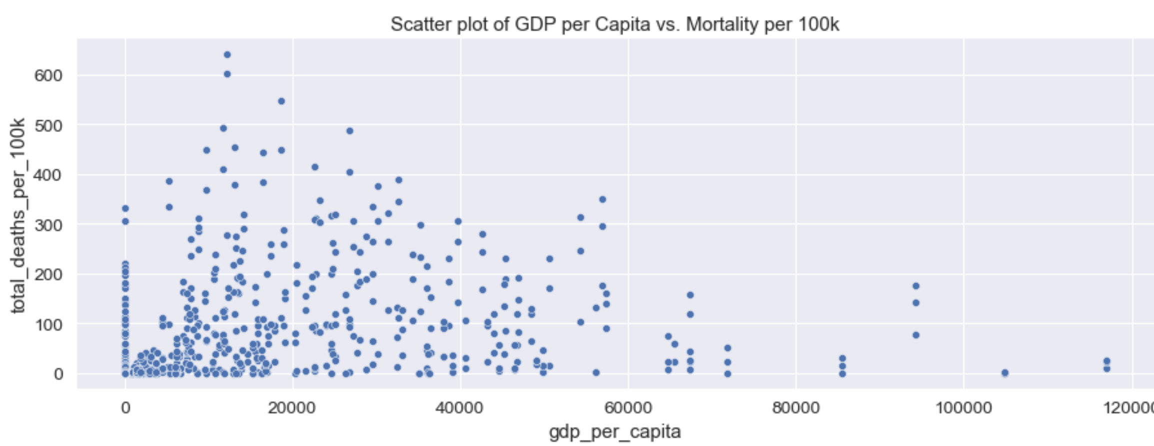
```
sns.scatterplot(x="gdp_per_capita",y="population",data=covid_data_sns)
plt.title("Scatter plot of GPD per Capita vs Population")
plt.show()
```



looks like a few outliers with a few high gdp nodes with relatively low populations

In [60]:

```
sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=covid_data_
#plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.title("Scatter plot of GDP per Capita vs. Mortality per 100k", size = 15)
plt.show()
```



looks like an intersting visual and that we can what appears to be a pattern between countries, GDP and mortality

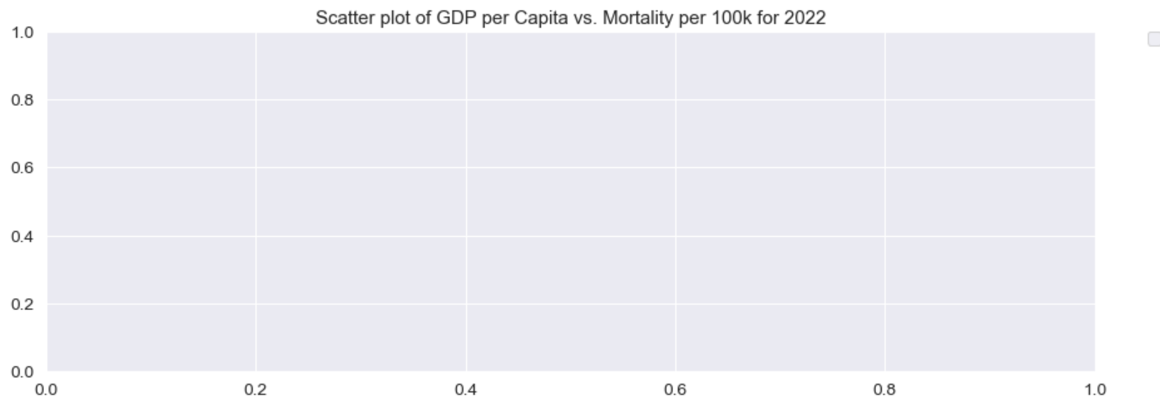
In [61]:

```
#create data sets for each year
df_2020 = covid_data_sns[covid_data_sns["date"].isin(["2020-12-31"])]
df_2021 = covid_data_sns[covid_data_sns["date"].isin(["2021-12-31"])]
df_2022 = covid_data_sns[covid_data_sns["date"].isin(["2022-09-15"])]
```

In [62]:

```
sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=df_2022, hu
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.title("Scatter plot of GDP per Capita vs. Mortality per 100k for 2022", s
plt.show()
```

```
No handles with labels found to put in legend.
```



only displaying a single point in time gets rid of some noise and it would appear looking at the latest data here that there appears to be a relationship between gdp and mortality rates. it might be easier to review by the top and bottom countries

In [63]:

```
# Top n data; use: top_n_parameter
#https://datascientyst.com/get-top-10-highest-lowest-values-pandas/
#df.nlargest; df.nsmallest


print("Top countries by cases and deaths:")
print()

df_2020 = covid_data_sns[covid_data_sns["date"].isin(["2020-12-31"])]
df_2021 = covid_data_sns[covid_data_sns["date"].isin(["2021-12-31"])]
df_2022 = covid_data_sns[covid_data_sns["date"].isin([last_date_n])]
```

```
Top countries by cases and deaths:
```

In [64]:

```python
print("creating a sets of top n cases and deaths per 100k of the population")
print()
print("Bottom countries by cases and deaths:")
print()

top_df_2020_cases_per_100k = df_2020.nlargest(n=top_n_parameter, columns=["to
print("top_df_2020_cases_per_100k")
print(top_df_2020_cases_per_100k)
print("-"*100)

top_df_2020_deaths_per_100k = df_2020.nlargest(n=top_n_parameter, columns=["t
print("top_df_2020_deaths_per_100k")
print(top_df_2020_deaths_per_100k)
print("-"*100)

top_df_2021_cases_per_100k = df_2021.nlargest(n=top_n_parameter, columns=["to
print("top_df_2021_cases_per_100k")
print(top_df_2021_cases_per_100k)
print("-"*100)

top_df_2021_deaths_per_100k = df_2021.nlargest(n=top_n_parameter, columns=["t
print("top_df_2021_deaths_per_100k")
print(top_df_2021_deaths_per_100k)
print("-"*100)

top_df_2022_cases_per_100k = df_2022.nlargest(n=top_n_parameter, columns=["to
print("top_df_2022_cases_per_100k")
print(top_df_2022_cases_per_100k)
print("-"*100)

top_df_2022_deaths_per_100k = df_2022.nlargest(n=top_n_parameter, columns=["t
print("top_df_2022_deaths_per_100k")
print(top_df_2022_deaths_per_100k)
```

```
creating a sets of top n cases and deaths per 100k of the population

Bottom countries by cases and deaths:

top_df_2020_cases_per_100k
             date iso_code      location  total_cases  total_cases_per_100
k  \
4061   2020-12-31      AND       Andorra       8049.0          10184.22451
1
130241 2020-12-31      MNE    Montenegro      48247.0           7684.36862
4
115744 2020-12-31      LUX    Luxembourg      46415.0           7260.04620
5
168582 2020-12-31      SMR    San Marino       2333.0           6913.41196
0
49976  2020-12-31      CZE       Czechia     718661.0           6837.39029
1
15321  2020-12-31      BHR       Bahrain      92675.0           6333.43926
1
```

```
75180  2020-12-31      GIB      Gibraltar       2040.0          6244.26079
0
72368  2020-12-31      GEO        Georgia     227420.0          6051.65541
1
205157 2020-12-31      USA  United States   20221641.0          6000.52925
0
177403 2020-12-31      SVN       Slovenia     122152.0          5763.49078
3

        total_deaths  total_deaths_per_100k   population  population_densit
y  \
4061            84.0             106.283372      79034.0              163.75
5
130241         682.0             108.623114     627859.0               46.28
0
115744         495.0              77.425894     639321.0              231.44
7
168582          59.0             174.835536      33746.0              556.66
7
49976        11580.0             110.172918   10510750.0              137.17
6
15321          352.0              24.055793    1463265.0             1935.90
7
75180            7.0              21.426385      32670.0             3457.10
0
72368         2505.0              66.658151    3757980.0               65.03
2
205157      350544.0             104.019724  336997624.0               35.60
8
177403        2697.0             127.252396    2119410.0              102.61
9

        gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
4061             0.000              0.0                      0.0      high
130241       16409.288              1.0                      0.0      high
115744       94277.965              0.2                      0.0      high
168582       56861.470              0.0                      0.0      high
49976        32605.906              0.0                      1.0      high
15321        43290.705              0.0                      0.0
75180            0.000              0.0                      0.0
72368         9745.079              4.2                      0.0      high
205157       54225.446              1.2                  44827.0      high
177403       31400.840              0.0                      0.0      high
---------------------------------------------------------------------------
-------------------------
top_df_2020_deaths_per_100k
             date iso_code                location  total_cases  \
154752 2020-12-31      PER                    Peru    1015137.0
168582 2020-12-31      SMR              San Marino       2333.0
19050  2020-12-31      BEL                 Belgium     646496.0
204187 2020-12-31      GBR          United Kingdom    2488780.0
177403 2020-12-31      SVN                Slovenia     122152.0
97104  2020-12-31      ITA                   Italy    2107166.0
25461  2020-12-31      BIH  Bosnia and Herzegovina     110985.0
145044 2020-12-31      MKD         North Macedonia      83329.0
111986 2020-12-31      LIE           Liechtenstein       2221.0
49976  2020-12-31      CZE                 Czechia     718661.0
```

```
        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
154752           3010.893634       93070.0             276.045372  3371547
2.0
168582           6913.411960          59.0             174.835536     3374
6.0
19050            5567.760016       19528.0             168.179258  1161142
0.0
204187           3699.080751       94998.0             141.195796  6728104
0.0
177403           5763.490783        2697.0             127.252396   211941
0.0
97104            3556.978835       74159.0             125.183300  5924033
0.0
25461            3393.058210        4050.0             123.817505   327094
3.0
145044           3961.765391        2503.0             119.001773   210333
0.0
111986           5689.182612          44.0             112.707805     3903
9.0
49976            6837.390291       11580.0             110.172918  1051075
0.0

        population_density  gdp_per_capita  extreme_poverty  \
154752              25.129       12236.706              3.5
168582             556.667       56861.470              0.0
19050              375.564       42658.576              0.2
204187             272.898       39753.244              0.2
177403             102.619       31400.840              0.0
97104              205.859       35220.084              2.0
25461               68.496       11713.895              0.2
145044              82.600       13111.214              5.0
111986             237.012           0.000              0.0
49976              137.176       32605.906              0.0

        people_fully_vaccinated mortality
154752                      0.0      high
168582                      0.0      high
19050                      21.0      high
204187                      0.0      high
177403                      0.0      high
97104                       0.0      high
25461                       0.0      high
145044                      0.0      high
111986                      0.0      high
49976                       1.0      high
---------------------------------------------------------------------------
-------------------------
top_df_2021_cases_per_100k
             date iso_code    location  total_cases  total_cases_per_100k
\
4426   2021-12-31      AND     Andorra      23740.0          30037.705291
130606 2021-12-31      MNE  Montenegro     170034.0          27081.558121
75545  2021-12-31      GIB   Gibraltar       8701.0          26632.996633
176809 2021-12-31      SVK    Slovakia    1371082.0          25168.449646
72733  2021-12-31      GEO     Georgia     934741.0          24873.495867
```

```
168947 2021-12-31      SMR  San Marino       8202.0          24305.102827
50341  2021-12-31      CZE     Czechia    2475729.0          23554.256357
173557 2021-12-31      SYC  Seychelles      24788.0          23281.675589
177768 2021-12-31      SVN    Slovenia     464048.0          21895.150065
129691 2021-12-31      MNG    Mongolia     692621.0          20688.951670

        total_deaths  total_deaths_per_100k  population  population_density
\
4426           140.0             177.138953     79034.0             163.755
130606        2411.0             384.003415    627859.0              46.280
75545          100.0             306.091215     32670.0            3457.100
176809       16635.0             305.362597   5447622.0             113.128
72733        13800.0             367.218559   3757980.0              65.032
168947         100.0             296.331417     33746.0             556.667
50341        36129.0             343.733796  10510750.0             137.176
173557         134.0             125.857049    106470.0             208.354
177768        5589.0             263.705465   2119410.0             102.619
129691        1986.0              59.322859   3347782.0               1.980

        gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
4426             0.000              0.0                      0.0      high
130606       16409.288              1.0                 272853.0      high
75545            0.000              0.0                      0.0      high
176809       30155.152              0.7                      0.0      high
72733         9745.079              4.2                      0.0      high
168947       56861.470              0.0                      0.0      high
50341        32605.906              0.0                6661738.0      high
173557       26382.287              1.1                      0.0      high
177768       31400.840              0.0                1188990.0      high
129691       11840.846              0.5                2163572.0      high
---------------------------------------------------------------------------
-------------------------
top_df_2021_deaths_per_100k
             date iso_code               location  total_cases  \
155117 2021-12-31      PER                   Peru    2296831.0
30414  2021-12-31      BGR               Bulgaria     747108.0
25826  2021-12-31      BIH  Bosnia and Herzegovina     291313.0
88064  2021-12-31      HUN                Hungary    1256415.0
130606 2021-12-31      MNE             Montenegro     170034.0
145409 2021-12-31      MKD        North Macedonia     225049.0
72733  2021-12-31      GEO                Georgia     934741.0
50341  2021-12-31      CZE                Czechia    2475729.0
127837 2021-12-31      MDA                Moldova     376155.0
46648  2021-12-31      HRV                Croatia     715245.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
155117           6812.394618      202690.0             601.177999  3371547
2.0
30414           10849.873974       30955.0             449.543906   688586
8.0
25826            8906.086104       13442.0             410.951826   327094
3.0
88064           12939.677558       39186.0             403.572231   970978
6.0
130606          27081.558121        2411.0             384.003415    62785
9.0
```

```
145409         10699.652456       7960.0             378.447509   210333
0.0
72733          24873.495867      13800.0             367.218559   375798
0.0
50341          23554.256357      36129.0             343.733796  1051075
0.0
127837         12286.600124      10275.0             335.619136   306150
6.0
46648          17616.286158      12538.0             308.807466   406013
5.0

        population_density  gdp_per_capita  extreme_poverty  \
155117              25.129       12236.706              3.5
30414               65.180       18563.307              1.5
25826               68.496       11713.895              0.2
88064              108.043       26777.561              0.5
130606              46.280       16409.288              1.0
145409              82.600       13111.214              5.0
72733               65.032        9745.079              4.2
50341              137.176       32605.906              0.0
127837             123.655        5189.972              0.2
46648               73.726       22669.797              0.7

        people_fully_vaccinated mortality
155117               22083549.0      high
30414                 1914910.0      high
25826                       0.0      high
88064                       0.0      high
130606                 272853.0      high
145409                      0.0      high
72733                       0.0      high
50341                 6661738.0      high
127837                 982152.0      high
46648                 1953540.0      high
---------------------------------------------------------------------------
-------------------------
top_df_2022_cases_per_100k
             date iso_code                   location  total_cases  \
65515  2022-09-16      FRO             Faeroe Islands      34658.0
49669  2022-09-16      CYP                     Cyprus     582381.0
75804  2022-09-16      GIB                  Gibraltar      20069.0
169206 2022-09-16      SMR                 San Marino      20552.0
4685   2022-09-16      AND                    Andorra      46147.0
13160  2022-09-16      AUT                    Austria    5008515.0
52480  2022-09-16      DNK                    Denmark    3285290.0
89256  2022-09-16      ISL                    Iceland     205284.0
178027 2022-09-16      SVN                   Slovenia    1153964.0
166687 2022-09-16      SPM  Saint Pierre and Miquelon       3166.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
65515           65530.933293          28.0              52.942066     5288
8.0
49669           64997.371672        1178.0             131.472187    89600
7.0
75804           61429.445975         108.0             330.578512     3267
0.0
```

```
169206          60902.032834         118.0             349.671072     3374
6.0
4685            58388.794696         155.0             196.118126     7903
4.0
13160           56136.168666       20664.0             231.605134   892208
2.0
52480           56118.129766        6993.0             119.451884   585424
0.0
89256           55431.973753         213.0              57.515493    37033
5.0
178027          54447.416970        6802.0             320.938374   211941
0.0
166687          53816.080231           1.0              16.998130      588
3.0

        population_density  gdp_per_capita  extreme_poverty  \
65515               35.308           0.000              0.0
49669              127.657       32415.132              0.0
75804             3457.100           0.000              0.0
169206             556.667       56861.470              0.0
4685               163.755           0.000              0.0
13160              106.749       45436.686              0.7
52480              136.520       46682.515              0.2
89256                3.404       46482.958              0.2
178027             102.619       31400.840              0.0
166687               0.000           0.000              0.0

        people_fully_vaccinated mortality
65515                       0.0      high
49669                       0.0      high
75804                       0.0      high
169206                      0.0      high
4685                        0.0      high
13160                       0.0      high
52480                       0.0      high
89256                       0.0      high
178027                      0.0      high
166687                      0.0
---------------------------------------------------------------------------
-------------------------
top_df_2022_deaths_per_100k
             date iso_code                location  total_cases  \
155376 2022-09-16      PER                    Peru    4131137.0
30673  2022-09-16      BGR                Bulgaria    1251331.0
26085  2022-09-16      BIH  Bosnia and Herzegovina     397822.0
88323  2022-09-16      HUN                 Hungary    2070443.0
145668 2022-09-16      MKD         North Macedonia     342075.0
72992  2022-09-16      GEO                 Georgia    1762206.0
130865 2022-09-16      MNE              Montenegro     278134.0
46907  2022-09-16      HRV                 Croatia    1223641.0
50600  2022-09-16      CZE                 Czechia    4073515.0
128096 2022-09-16      MDA                 Moldova     583183.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
155376          12252.941320      216264.0             641.438447  3371547
2.0
```

```
30673           18172.451171       37675.0             547.135089   688586
8.0
26085           12162.303042       16108.0             492.457374   327094
3.0
88323           21323.260883       47409.0             488.259988   970978
6.0
145668          16263.496456        9521.0             452.663158   210333
0.0
72992           46892.373030       16900.0             449.709684   375798
0.0
130865          44298.799571        2778.0             442.456029    62785
9.0
46907           30137.938763       16834.0             414.616755   406013
5.0
50600           38755.702495       40951.0             389.610637  1051075
0.0
128096          19048.892930       11808.0             385.692532   306150
6.0

        population_density  gdp_per_capita  extreme_poverty  \
155376              25.129       12236.706              3.5
30673               65.180       18563.307              1.5
26085               68.496       11713.895              0.2
88323              108.043       26777.561              0.5
145668              82.600       13111.214              5.0
72992               65.032        9745.079              4.2
130865              46.280       16409.288              1.0
46907               73.726       22669.797              0.7
50600              137.176       32605.906              0.0
128096             123.655        5189.972              0.2

        people_fully_vaccinated mortality
155376                      0.0      high
30673                 2070947.0      high
26085                       0.0      high
88323                       0.0      high
145668                      0.0      high
72992                       0.0      high
130865                      0.0      high
46907                       0.0      high
50600                 6888750.0      high
128096                      0.0      high
```

In [65]:

```
print("creating a sets of bottom n cases and deaths per 100k of the populatio
print()

# Bottom n data; use: top_n_parameter
#https://datascientyst.com/get-top-10-highest-lowest-values-pandas/
#df.nlargest; df.nsmallest
print("Bottom countries by cases and deaths:")
print()

bot_df_2020_cases_per_100k = df_2020.nsmallest(n=top_n_parameter, columns=["t
print("bot_df_2020_cases_per_100k")
print(top_df_2020_cases_per_100k)
print("-"*100)
print()

bot_df_2020_deaths_per_100k = df_2020.nsmallest(n=top_n_parameter, columns=["
print("bot_df_2020_deaths_per_100k")
print(top_df_2020_deaths_per_100k)
print("-"*100)
print()

bot_df_2021_cases_per_100k = df_2021.nsmallest(n=top_n_parameter, columns=["t
print("bot_df_2021_cases_per_100k")
print(top_df_2021_cases_per_100k)
print("-"*100)
print()

bot_df_2021_deaths_per_100k = df_2021.nsmallest(n=top_n_parameter, columns=["
print("bot_df_2021_deaths_per_100k")
print(top_df_2021_deaths_per_100k)
print("-"*100)
print()

bot_df_2022_cases_per_100k = df_2022.nsmallest(n=top_n_parameter, columns=["t
print("bot_df_2022_cases_per_100k")
print(top_df_2022_cases_per_100k)
print("-"*100)
print()

bot_df_2022_deaths_per_100k = df_2022.nsmallest(n=top_n_parameter, columns=["
print("bot_df_2022_deaths_per_100k")
print(top_df_2022_deaths_per_100k)
print("-"*100)
print()
```

```
creating a sets of bottom n cases and deaths per 100k of the population

Bottom countries by cases and deaths:

bot_df_2020_cases_per_100k
             date iso_code       location  total_cases  total_cases_per_100
k  \
4061   2020-12-31      AND        Andorra       8049.0           10184.22451
1
130241 2020-12-31      MNE     Montenegro      48247.0            7684.36862
```

```
4
115744 2020-12-31      LUX     Luxembourg      46415.0           7260.04620
5
168582 2020-12-31      SMR     San Marino       2333.0           6913.41196
0
49976  2020-12-31      CZE        Czechia     718661.0           6837.39029
1
15321  2020-12-31      BHR        Bahrain      92675.0           6333.43926
1
75180  2020-12-31      GIB      Gibraltar       2040.0           6244.26079
0
72368  2020-12-31      GEO        Georgia     227420.0           6051.65541
1
205157 2020-12-31      USA  United States   20221641.0           6000.52925
0
177403 2020-12-31      SVN       Slovenia     122152.0           5763.49078
3

        total_deaths  total_deaths_per_100k   population  population_densit
y  \
4061            84.0             106.283372      79034.0               163.75
5
130241         682.0             108.623114     627859.0                46.28
0
115744         495.0              77.425894     639321.0               231.44
7
168582          59.0             174.835536      33746.0               556.66
7
49976        11580.0             110.172918   10510750.0               137.17
6
15321          352.0              24.055793    1463265.0              1935.90
7
75180            7.0              21.426385      32670.0              3457.10
0
72368         2505.0              66.658151    3757980.0                65.03
2
205157      350544.0             104.019724  336997624.0                35.60
8
177403        2697.0             127.252396    2119410.0               102.61
9

        gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
4061             0.000              0.0                      0.0      high
130241       16409.288              1.0                      0.0      high
115744       94277.965              0.2                      0.0      high
168582       56861.470              0.0                      0.0      high
49976        32605.906              0.0                      1.0      high
15321        43290.705              0.0                      0.0
75180            0.000              0.0                      0.0
72368         9745.079              4.2                      0.0      high
205157       54225.446              1.2                  44827.0      high
177403       31400.840              0.0                      0.0      high
---------------------------------------------------------------------------
-----------------------

bot_df_2020_deaths_per_100k
             date iso_code                location  total_cases  \
```

```
154752 2020-12-31      PER                    Peru    1015137.0
168582 2020-12-31      SMR              San Marino       2333.0
19050  2020-12-31      BEL                 Belgium     646496.0
204187 2020-12-31      GBR          United Kingdom    2488780.0
177403 2020-12-31      SVN                Slovenia     122152.0
97104  2020-12-31      ITA                   Italy    2107166.0
25461  2020-12-31      BIH  Bosnia and Herzegovina     110985.0
145044 2020-12-31      MKD         North Macedonia      83329.0
111986 2020-12-31      LIE           Liechtenstein       2221.0
49976  2020-12-31      CZE                 Czechia     718661.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
154752           3010.893634       93070.0             276.045372  3371547
2.0
168582           6913.411960          59.0             174.835536     3374
6.0
19050            5567.760016       19528.0             168.179258  1161142
0.0
204187           3699.080751       94998.0             141.195796  6728104
0.0
177403           5763.490783        2697.0             127.252396   211941
0.0
97104            3556.978835       74159.0             125.183300  5924033
0.0
25461            3393.058210        4050.0             123.817505   327094
3.0
145044           3961.765391        2503.0             119.001773   210333
0.0
111986           5689.182612          44.0             112.707805     3903
9.0
49976            6837.390291       11580.0             110.172918  1051075
0.0

        population_density  gdp_per_capita  extreme_poverty  \
154752              25.129       12236.706              3.5
168582             556.667       56861.470              0.0
19050              375.564       42658.576              0.2
204187             272.898       39753.244              0.2
177403             102.619       31400.840              0.0
97104              205.859       35220.084              2.0
25461               68.496       11713.895              0.2
145044              82.600       13111.214              5.0
111986             237.012           0.000              0.0
49976              137.176       32605.906              0.0

        people_fully_vaccinated mortality
154752                      0.0      high
168582                      0.0      high
19050                      21.0      high
204187                      0.0      high
177403                      0.0      high
97104                       0.0      high
25461                       0.0      high
145044                      0.0      high
111986                      0.0      high
49976                       1.0      high
```

```
--------------------------------------------------------------------------
-------------------------

bot_df_2021_cases_per_100k
             date iso_code    location  total_cases  total_cases_per_100k  \
4426   2021-12-31      AND     Andorra      23740.0          30037.705291
130606 2021-12-31      MNE  Montenegro     170034.0          27081.558121
75545  2021-12-31      GIB   Gibraltar       8701.0          26632.996633
176809 2021-12-31      SVK    Slovakia    1371082.0          25168.449646
72733  2021-12-31      GEO     Georgia     934741.0          24873.495867
168947 2021-12-31      SMR  San Marino       8202.0          24305.102827
50341  2021-12-31      CZE     Czechia    2475729.0          23554.256357
173557 2021-12-31      SYC  Seychelles      24788.0          23281.675589
177768 2021-12-31      SVN    Slovenia     464048.0          21895.150065
129691 2021-12-31      MNG    Mongolia     692621.0          20688.951670

        total_deaths  total_deaths_per_100k  population  population_density  \
4426           140.0             177.138953     79034.0             163.755
130606        2411.0             384.003415    627859.0              46.280
75545          100.0             306.091215     32670.0            3457.100
176809       16635.0             305.362597   5447622.0             113.128
72733        13800.0             367.218559   3757980.0              65.032
168947         100.0             296.331417     33746.0             556.667
50341        36129.0             343.733796  10510750.0             137.176
173557         134.0             125.857049    106470.0             208.354
177768        5589.0             263.705465   2119410.0             102.619
129691        1986.0              59.322859   3347782.0               1.980

        gdp_per_capita  extreme_poverty  people_fully_vaccinated mortality
4426             0.000              0.0                      0.0      high
130606       16409.288              1.0                 272853.0      high
75545            0.000              0.0                      0.0      high
176809       30155.152              0.7                      0.0      high
72733         9745.079              4.2                      0.0      high
168947       56861.470              0.0                      0.0      high
50341        32605.906              0.0                6661738.0      high
173557       26382.287              1.1                      0.0      high
177768       31400.840              0.0                1188990.0      high
129691       11840.846              0.5                2163572.0      high
--------------------------------------------------------------------------
-------------------------

bot_df_2021_deaths_per_100k
             date iso_code                location  total_cases  \
155117 2021-12-31      PER                    Peru    2296831.0
30414  2021-12-31      BGR                Bulgaria     747108.0
25826  2021-12-31      BIH  Bosnia and Herzegovina     291313.0
88064  2021-12-31      HUN                 Hungary    1256415.0
130606 2021-12-31      MNE              Montenegro     170034.0
145409 2021-12-31      MKD         North Macedonia     225049.0
72733  2021-12-31      GEO                 Georgia     934741.0
50341  2021-12-31      CZE                 Czechia    2475729.0
127837 2021-12-31      MDA                 Moldova     376155.0
46648  2021-12-31      HRV                 Croatia     715245.0
```

```
        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
155117           6812.394618      202690.0             601.177999  3371547
2.0
30414           10849.873974       30955.0             449.543906   688586
8.0
25826            8906.086104       13442.0             410.951826   327094
3.0
88064           12939.677558       39186.0             403.572231   970978
6.0
130606          27081.558121        2411.0             384.003415    62785
9.0
145409          10699.652456        7960.0             378.447509   210333
0.0
72733           24873.495867       13800.0             367.218559   375798
0.0
50341           23554.256357       36129.0             343.733796  1051075
0.0
127837          12286.600124       10275.0             335.619136   306150
6.0
46648           17616.286158       12538.0             308.807466   406013
5.0

        population_density  gdp_per_capita  extreme_poverty  \
155117              25.129       12236.706              3.5
30414               65.180       18563.307              1.5
25826               68.496       11713.895              0.2
88064              108.043       26777.561              0.5
130606              46.280       16409.288              1.0
145409              82.600       13111.214              5.0
72733               65.032        9745.079              4.2
50341              137.176       32605.906              0.0
127837             123.655        5189.972              0.2
46648               73.726       22669.797              0.7

        people_fully_vaccinated mortality
155117               22083549.0      high
30414                 1914910.0      high
25826                       0.0      high
88064                       0.0      high
130606                 272853.0      high
145409                      0.0      high
72733                       0.0      high
50341                 6661738.0      high
127837                 982152.0      high
46648                 1953540.0      high
---------------------------------------------------------------------------
-------------------------

bot_df_2022_cases_per_100k
             date iso_code                 location  total_cases  \
65515  2022-09-16      FRO           Faeroe Islands      34658.0
49669  2022-09-16      CYP                   Cyprus     582381.0
75804  2022-09-16      GIB                Gibraltar      20069.0
169206 2022-09-16      SMR               San Marino      20552.0
4685   2022-09-16      AND                  Andorra      46147.0
13160  2022-09-16      AUT                  Austria    5008515.0
```

```
52480  2022-09-16      DNK                     Denmark    3285290.0
89256  2022-09-16      ISL                     Iceland     205284.0
178027 2022-09-16      SVN                    Slovenia    1153964.0
166687 2022-09-16      SPM  Saint Pierre and Miquelon       3166.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
65515           65530.933293          28.0              52.942066     5288
8.0
49669           64997.371672        1178.0             131.472187    89600
7.0
75804           61429.445975         108.0             330.578512     3267
0.0
169206          60902.032834         118.0             349.671072     3374
6.0
4685            58388.794696         155.0             196.118126     7903
4.0
13160           56136.168666       20664.0             231.605134   892208
2.0
52480           56118.129766        6993.0             119.451884   585424
0.0
89256           55431.973753         213.0              57.515493    37033
5.0
178027          54447.416970        6802.0             320.938374   211941
0.0
166687          53816.080231           1.0              16.998130      588
3.0

        population_density  gdp_per_capita  extreme_poverty  \
65515               35.308           0.000              0.0
49669              127.657       32415.132              0.0
75804             3457.100           0.000              0.0
169206             556.667       56861.470              0.0
4685               163.755           0.000              0.0
13160              106.749       45436.686              0.7
52480              136.520       46682.515              0.2
89256                3.404       46482.958              0.2
178027             102.619       31400.840              0.0
166687               0.000           0.000              0.0

        people_fully_vaccinated mortality
65515                       0.0      high
49669                       0.0      high
75804                       0.0      high
169206                      0.0      high
4685                        0.0      high
13160                       0.0      high
52480                       0.0      high
89256                       0.0      high
178027                      0.0      high
166687                      0.0
---------------------------------------------------------------------------
-------------------------

bot_df_2022_deaths_per_100k
             date iso_code               location  total_cases  \
155376 2022-09-16      PER                   Peru    4131137.0
```

```
30673  2022-09-16      BGR                Bulgaria   1251331.0
26085  2022-09-16      BIH  Bosnia and Herzegovina    397822.0
88323  2022-09-16      HUN                 Hungary   2070443.0
145668 2022-09-16      MKD         North Macedonia    342075.0
72992  2022-09-16      GEO                 Georgia   1762206.0
130865 2022-09-16      MNE              Montenegro    278134.0
46907  2022-09-16      HRV                 Croatia   1223641.0
50600  2022-09-16      CZE                 Czechia   4073515.0
128096 2022-09-16      MDA                 Moldova    583183.0

        total_cases_per_100k  total_deaths  total_deaths_per_100k  populati
on  \
155376          12252.941320      216264.0             641.438447  3371547
2.0
30673           18172.451171       37675.0             547.135089   688586
8.0
26085           12162.303042       16108.0             492.457374   327094
3.0
88323           21323.260883       47409.0             488.259988   970978
6.0
145668          16263.496456        9521.0             452.663158   210333
0.0
72992           46892.373030       16900.0             449.709684   375798
0.0
130865          44298.799571        2778.0             442.456029    62785
9.0
46907           30137.938763       16834.0             414.616755   406013
5.0
50600           38755.702495       40951.0             389.610637  1051075
0.0
128096          19048.892930       11808.0             385.692532   306150
6.0

        population_density  gdp_per_capita  extreme_poverty  \
155376              25.129       12236.706              3.5
30673               65.180       18563.307              1.5
26085               68.496       11713.895              0.2
88323              108.043       26777.561              0.5
145668              82.600       13111.214              5.0
72992               65.032        9745.079              4.2
130865              46.280       16409.288              1.0
46907               73.726       22669.797              0.7
50600              137.176       32605.906              0.0
128096             123.655        5189.972              0.2

        people_fully_vaccinated mortality
155376                      0.0      high
30673                 2070947.0      high
26085                       0.0      high
88323                       0.0      high
145668                      0.0      high
72992                       0.0      high
130865                      0.0      high
46907                       0.0      high
50600                 6888750.0      high
128096                      0.0      high
---------------------------------------------------------------------------
```

In [66]:

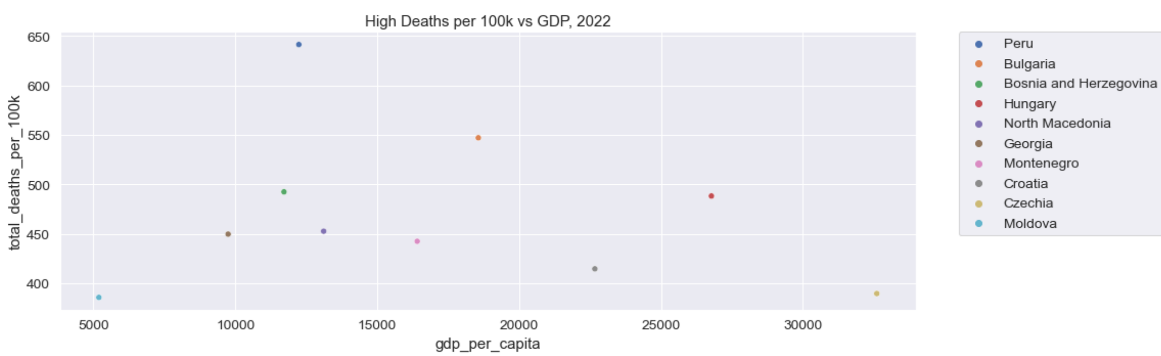
```python
# min value in the Top mortality (top deaths) data
print("min of 2020 top deaths/ 100k: "+str(top_df_2020_deaths_per_100k["total
print("max of 2020 top deaths/ 100k: "+str(top_df_2020_deaths_per_100k["total
print("-"*100)
print("min of 2021 top deaths/ 100k: "+str(top_df_2021_deaths_per_100k["total
print("max of 2021 top deaths/ 100k: "+str(top_df_2021_deaths_per_100k["total
print("-"*100)

# max value in the bottom mortality (bottom deaths) data
print("max of 2022 lowest deaths/ 100k: "+str(top_df_2022_deaths_per_100k["to
print("min of 2022 lowest deaths/ 100k: "+str(top_df_2020_deaths_per_100k["to
```

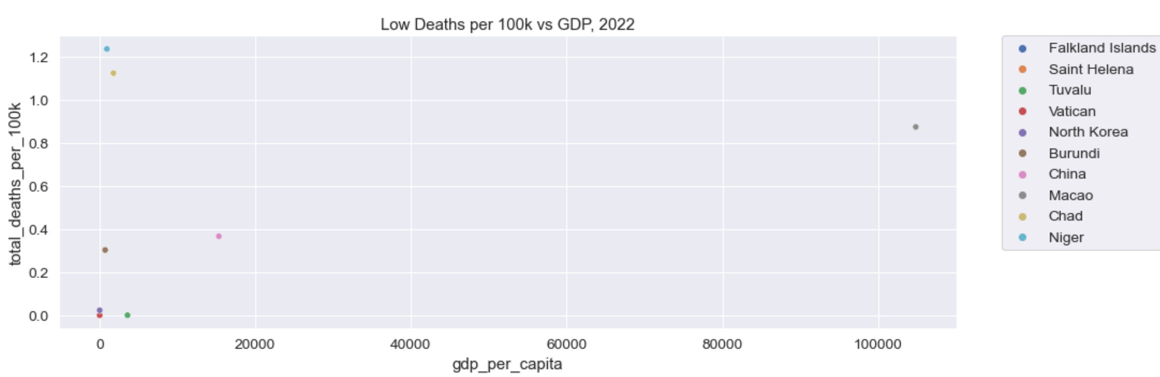
```
min of 2020 top deaths/ 100k: 110.17291820279239
max of 2020 top deaths/ 100k: 276.04537169166724
---------------------------------------------------------------------------
-------------------------
min of 2021 top deaths/ 100k: 308.80746576160647
max of 2021 top deaths/ 100k: 601.1779992283662
---------------------------------------------------------------------------
-------------------------
max of 2022 lowest deaths/ 100k: 641.4384470132882
min of 2022 lowest deaths/ 100k: 110.17291820279239
```

In [67]:

```
print("Top and Bottom Mortality vs GDP per capita")
print()

#Top deaths vs gdp
sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=top_df_2022
plt.title("High Deaths per 100k vs GDP, 2022", size = 15)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()

#bottom deaths vs gdp
sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=bot_df_2022
plt.title("Low Deaths per 100k vs GDP, 2022", size = 15)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```

```
Top and Bottom Mortality vs GDP per capita
```



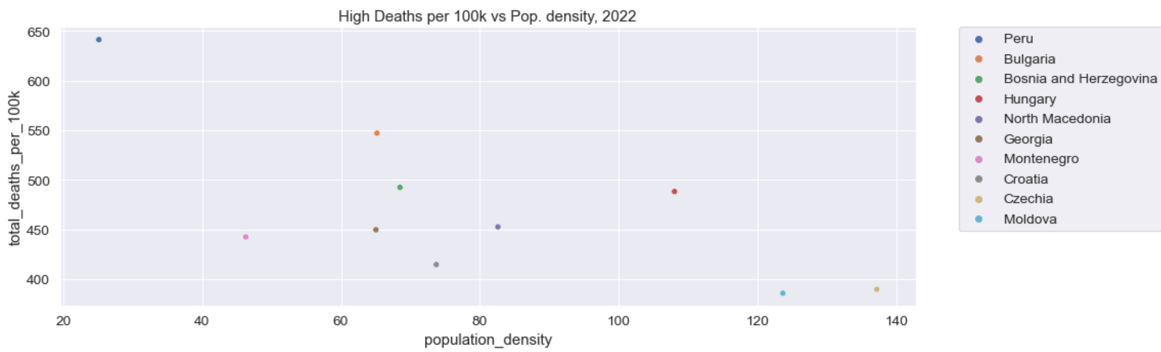


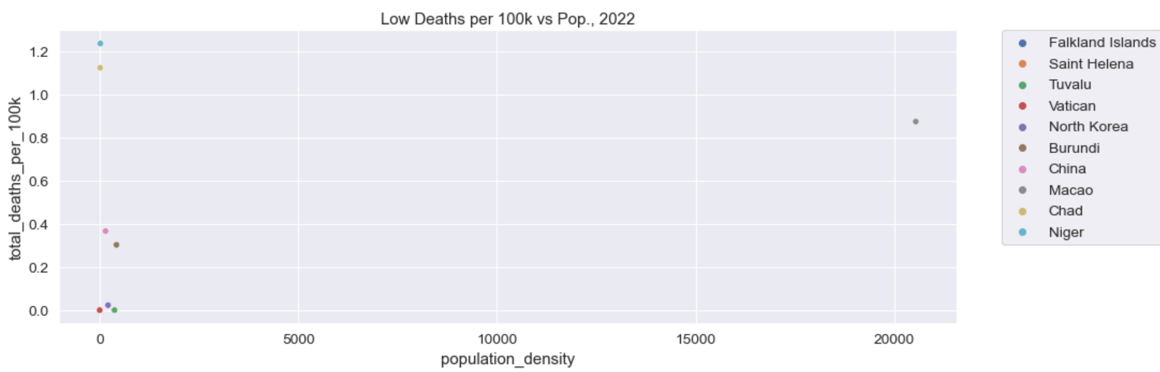
this looks strange, high mortality and gdp does not look related. it appears from the top chart that the top gdp countries also have higer mortality, for the most part these countries look like smaller nations. lets look by population density

In [68]:

```
print("Top and Bottom Mortality vs Population Density")

#Top deaths vs population_density
sns.scatterplot(x="population_density",y="total_deaths_per_100k",data=top_df_
plt.title("High Deaths per 100k vs Pop. density, 2022", size = 15)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()

#bottom deaths vs population_density
sns.scatterplot(x="population_density",y="total_deaths_per_100k",data=bot_df_
plt.title("Low Deaths per 100k vs Pop., 2022", size = 15)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```

```
Top and Bottom Mortality vs Population Density
```





this looks strange as well, higher mortality and lower density looks negatively related, it's difficult to tell because of the outlier, Macao. it appears from the top chart that the lesser dense countries also have higer mortality, for the most part these countries look like smaller nations.

In [69]:

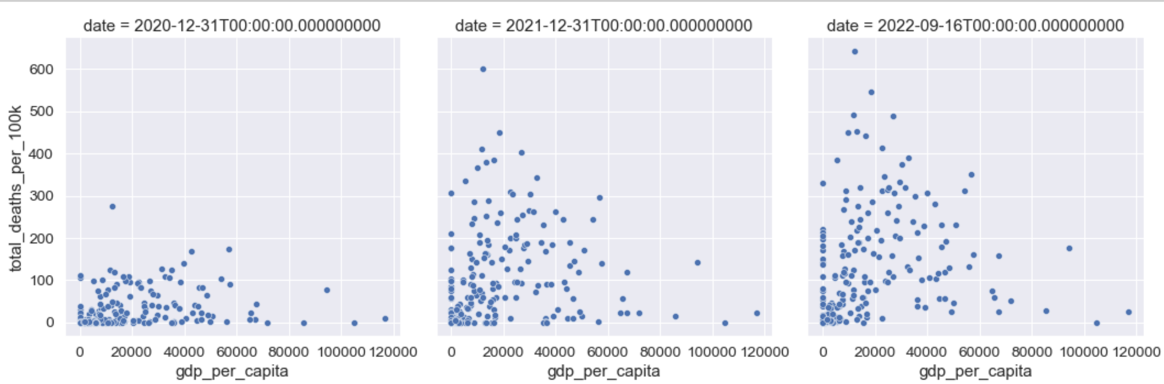
```
# calculate the threshold to use for the "high" and "low" mortality column

# min value in the Top mortality (top deaths) data
print("min 2020 top deaths/ 100k: "+str(top_df_2020_cases_per_100k["total_dea
print("max 2020 top deaths/ 100k: "+str(top_df_2020_cases_per_100k["total_dea
print("-"*100)

# max value in the bottom mortality (bottom deaths) data
print("max 2022 lowest deaths/ 100k: "+str(bot_df_2022_cases_per_100k["total_
print("min 2022 lowest deaths/ 100k: "+str(bot_df_2022_cases_per_100k["total_
```

```
min 2020 top deaths/ 100k: 21.426385062748697
max 2020 top deaths/ 100k: 174.83553606353345
---------------------------------------------------------------------------
-------------------------
max 2022 lowest deaths/ 100k: 6.5339380778536755
min 2022 lowest deaths/ 100k: 0.02310188288431166
```

I used this to set the group bads for the "Mortality" column. Using the 2020 top 10 records to set the lower limit of "high" mortality and current 2022 bottom 10 records to set the higher limit for the "low" mortality

In [70]:

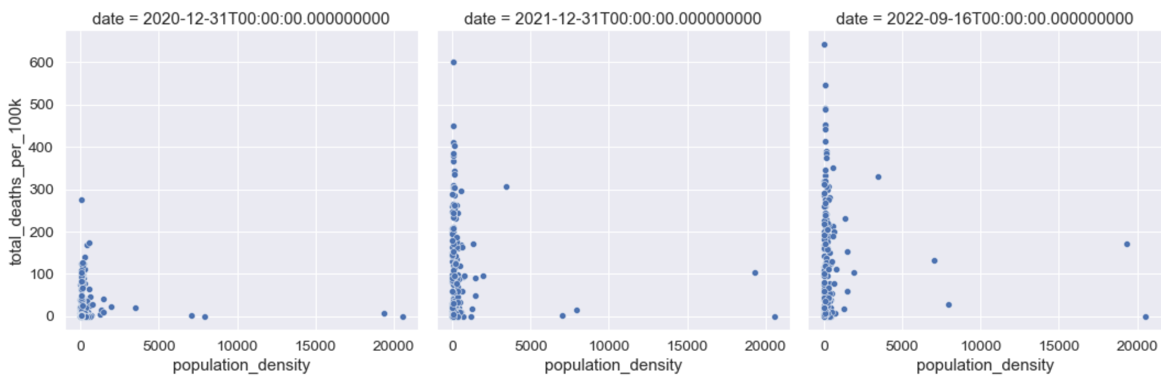
```
#Mortality per 100k and GDP per capita

#High mortality vs gdp
sns.relplot(x="gdp_per_capita",
            y="total_deaths_per_100k",
            data=covid_data_sns,
            kind="scatter",
            col = "date")
plt.show()

#bottom deaths vs gdp
#sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=bot_df_202
#plt.show()
```



we can observe from this that there are expected results that over time lower gdp per capita records had higher mortality per 100k of the population. However, there are some interesting results where lower gdp did not have a high mortality. notice the skew to the upper left over time which suggests lower gdp does impact higher mortality

In [71]:

```
#Mortality per 100k and population density

#High mortality vs pop density
sns.relplot(x="population_density",
            y="total_deaths_per_100k",
            data=covid_data_sns,
            kind="scatter",
            col = "date")
plt.show()

#bottom deaths vs gdp
#sns.scatterplot(x="gdp_per_capita",y="total_deaths_per_100k",data=bot_df_202
#plt.show()
```



we can observe here that a higher density in the population does have a more significant impact on mortality

In [72]:

```
covid_temp = covid_data_sns
covid_temp.sort_values("population_density", axis = 0, ascending = False, inp
covid_temp
```

Out[72]:

	date	iso_code	location	total_cases	total_cases_per_100k	total_deaths	total_d
117338	2022-09-16	MAC	Macao	793.0	115.495473	6.0	
116714	2020-12-31	MAC	Macao	46.0	6.699611	0.0	
117079	2021-12-31	MAC	Macao	79.0	11.505854	0.0	
128404	2020-	MCO	Monaco	875.0	2385.106035	3.0	

Regression analysis

In [73]:
```
# Machine Learning
```

In [74]:
```
# Machine learning KNN data
```

In [75]:
```
covid_data_sns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 658 entries, 117338 to 190952
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   date                      658 non-null    datetime64[ns]
 1   iso_code                  658 non-null    object
 2   location                  658 non-null    object
 3   total_cases               658 non-null    float64
 4   total_cases_per_100k      658 non-null    float64
 5   total_deaths              658 non-null    float64
 6   total_deaths_per_100k     658 non-null    float64
 7   population                658 non-null    float64
 8   population_density        658 non-null    float64
 9   gdp_per_capita            658 non-null    float64
 10  extreme_poverty           658 non-null    float64
 11  people_fully_vaccinated   658 non-null    float64
 12  mortality                 658 non-null    object
dtypes: datetime64[ns](1), float64(9), object(3)
memory usage: 72.0+ KB
```

In [76]:
```
date_list = unique(covid_data_sns["date"])
print("we have the expected 3 dates selected")
```

```
2022-09-16 00:00:00 2020-12-31 00:00:00 2021-12-31 00:00:00
we have the expected 3 dates selected
```

In [77]:

```
#create 2 sets of data for the review removing date, iso code, location, as t
#and even fewer value columns in the second set

covid_data_ml = covid_data_sns

covid_data_ml1 = covid_data_ml.drop(["date","iso_code","location"],axis=1)

covid_data_ml2 = covid_data_ml.drop(["date","iso_code","location","total_case

covid_data_ml2
```

Out[77]:

	total_cases_per_100k	total_deaths_per_100k	population	population_density	gdp_per_
117338	115.495473	0.873862	686607.0	20546.766	1048
116714	6.699611	0.000000	686607.0	20546.766	1048
117079	11.505854	0.000000	686607.0	20546.766	1048
128404	2385.106035	8.177506	36686.0	19347.500	
128769	13588.289811	103.581748	36686.0	19347.500	
...	...	...	...	...	
195759	0.000000	0.000000	1849.0	0.000	
131520	1041.430835	22.639801	4417.0	0.000	
163966	129.533679	0.000000	5404.0	0.000	
25158	42335.055793	142.290122	26706.0	0.000	
190952	24691.436414	43.876943	23859912.0	0.000	

658 rows × 8 columns

In [78]:

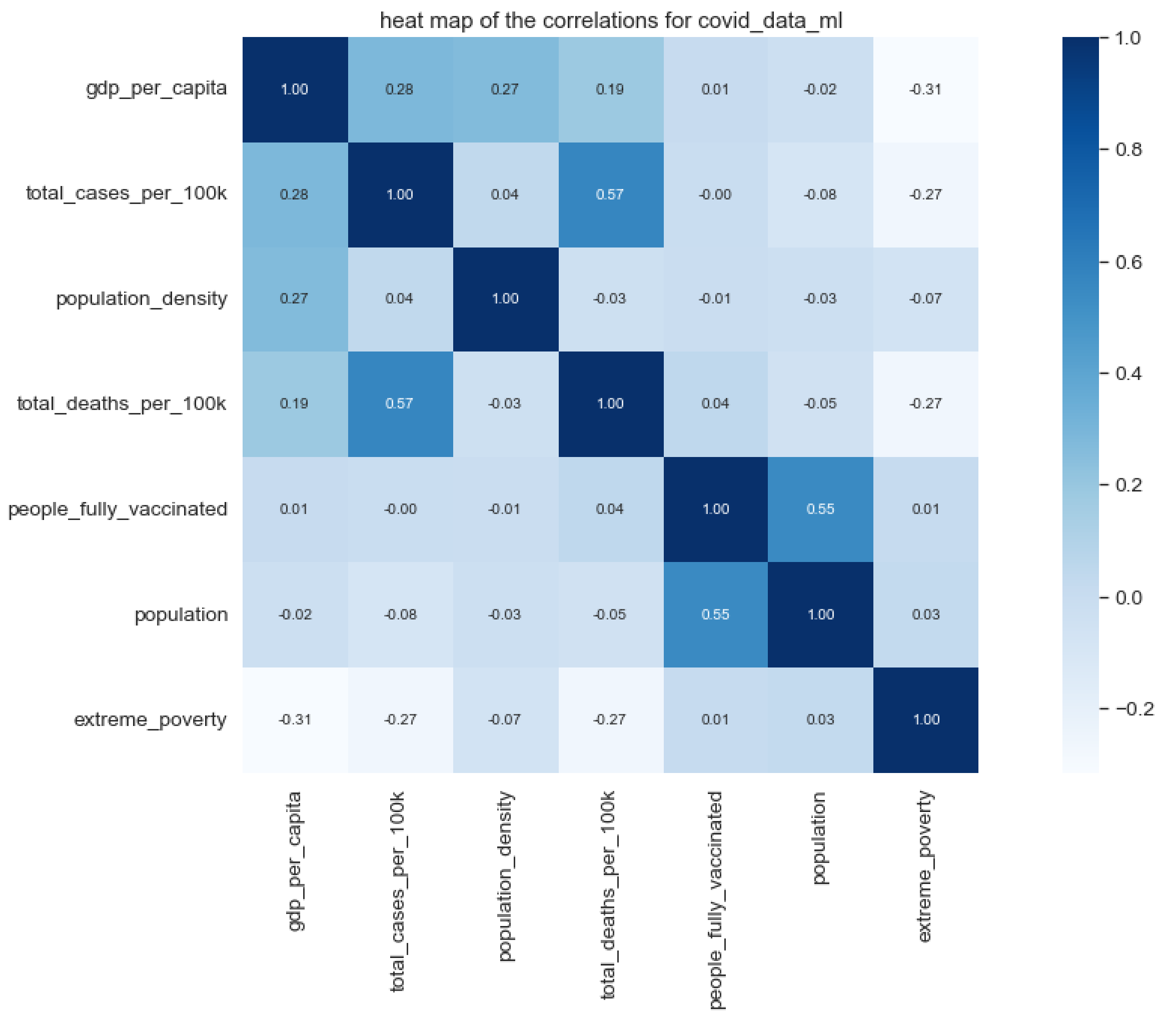
```
covid_data_ml2.corr()
```

Out[78]:

	total_cases_per_100k	total_deaths_per_100k	population	population_de
total_cases_per_100k	1.000000	0.565010	-0.084021	0.03
total_deaths_per_100k	0.565010	1.000000	-0.048313	-0.03
population	-0.084021	-0.048313	1.000000	-0.02
population_density	0.038308	-0.033790	-0.025052	1.0(
gdp_per_capita	0.281003	0.187821	-0.022257	0.2(
extreme_poverty	-0.265953	-0.265442	0.028340	-0.0(
people_fully_vaccinated	-0.002906	0.044485	0.545371	-0.0(

In [79]:

```
#heat map of the correlations for covid_data_ml2

Corr_data = covid_data_ml2

#Correlation of gdp
corr = Corr_data.corr()
plt.figure(figsize=(20, 9))

k = 12 #number of variables for heatmap

cols = corr.nlargest(k, 'gdp_per_capita')['gdp_per_capita'].index
cm = np.corrcoef(Corr_data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws
plt.title("heat map of the correlations for covid_data_ml")
plt.show()
```



Results

(Include the charts and describe them)

Supervised learning with classification

In [80]:
```
print("training and testing the data")
#from datacamp
print()

#covid_data_ml
#covid_data_ml1
#covid_data_ml2

ml_data = covid_data_ml1

X = ml_data.drop("mortality",axis=1).values #drop target value
y = ml_data["mortality"].values #target observations

#split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_
knn = KNeighborsClassifier(n_neighbors=6)

#fit the classiier to the training data
knn.fit(X_train, y_train)

#print the accuracy
print("The knn score:")
print(knn.score(X_test, y_test))
print()

y_pred_ = knn.predict(X_test)

print("Confusion matrix:")
print(confusion_matrix(y_test, y_pred_))
print()

print("Classification report:")
print(classification_report(y_test, y_pred_))
```

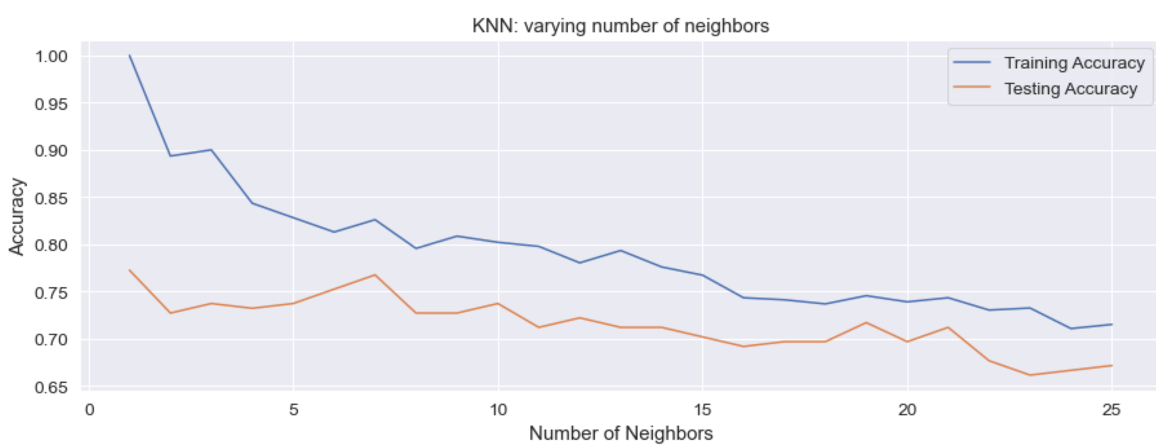
```
training and testing the data

The knn score:
0.7525252525252525

Confusion matrix:
[[104  11]
 [ 38  45]]

Classification report:
              precision    recall  f1-score   support

                   0.73      0.90      0.81       115
        high       0.80      0.54      0.65        83

    accuracy                           0.75       198
   macro avg       0.77      0.72      0.73       198
weighted avg       0.76      0.75      0.74       198
```

The knn score suggest there are tight relationships with the data. However, the "high" mortality classification prediction is not as high suggesting mortality from COVID is not that correlated to the gdp per capita or the population density

In [81]:
```
#model complexity
train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1,26)
```

In [82]:
```
# Loop through neighbors array
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors=neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor]=knn.score(X_train, y_train)
    test_accuracies[neighbor]=knn.score(X_test,y_test)
```

In [83]:
```
# plot training and test values
#plt.figure(figuresize=(8,6))
plt.title("KNN: varying number of neighbors")
plt.plot(neighbors, train_accuracies.values(), label="Training Accuracy")
plt.plot(neighbors, test_accuracies.values(), label="Testing Accuracy")
plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")

plt.show()
```



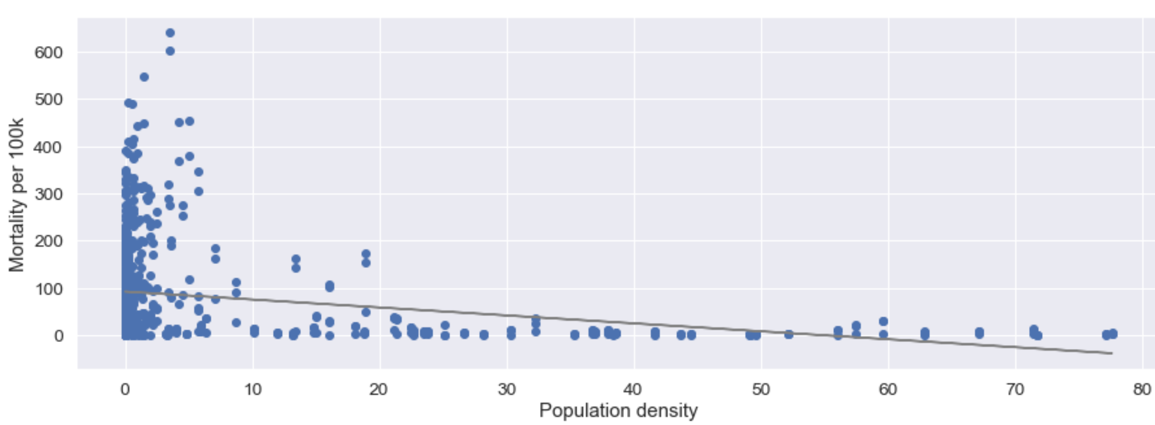
this shows k of 6 is a good choice as this displays the highest testing accuracy and thetraining score

Supervised learning with regression

In [84]:

```
#training and testing the data
#from datacamp

#covid_data_ml
#covid_data_ml1
#covid_data_ml2
#print(ml_data)

ml_data = covid_data_ml1


X = ml_data.drop("total_deaths_per_100k",axis=1).values #drop target value
y = ml_data["total_deaths_per_100k"].values #target observations


# predicting mortality using population density

#predict using pop_density (6)
X_pop_d = X[:,6]
#print(y.shape, X_pop_d.shape) # check shape
# reshape
X_pop_d = X_pop_d.reshape(-1,1)
#print(X_pop_d.shape) #check shape

#regression model
reg = LinearRegression()
reg.fit(X_pop_d,y)
predictions = reg.predict(X_pop_d)
print(predictions[:10])

#plot Total_deaths per 100k vs. population density with regression
plt.scatter(X_pop_d, y)
plt.plot(X_pop_d, predictions, color = "gray")
plt.ylabel("Mortality per 100k")
plt.xlabel("Population density")
plt.show()
```

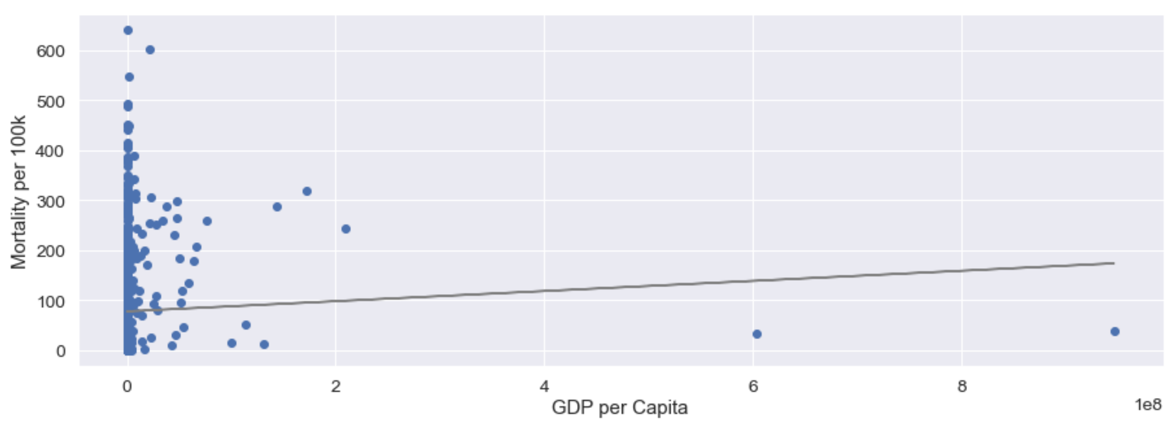
```
[91.77032418 91.77032418 91.77032418 91.77032418 91.77032418 91.77032418
 91.77032418 91.77032418 91.77032418 91.77032418]
```



Weak negative correlation. The higher the population density the less likey the mortality from COVID, this is unexpected.

In [85]:
```
# predicting mortality using gdp

#predict using gdp_per_capita (7)
X_gdp_c = X[:,7]

#print(ml_data)

# reshape
X_gdp_c = X_gdp_c.reshape(-1,1)
#print(X_gdp_c.shape) #check shape

#regression model
reg = LinearRegression()
reg.fit(X_gdp_c,y)
predictions = reg.predict(X_gdp_c)
print(predictions[:10])

#plot Total_deaths per 100k vs. population density
plt.scatter(X_gdp_c, y)
plt.plot(X_gdp_c, predictions, color = "gray")
plt.ylabel("Mortality per 100k")
plt.xlabel("GDP per Capita")
plt.show()
```

```
[77.93492175 77.93492175 77.93492175 77.93492175 77.93492175 77.93492175
 77.93492175 78.39156498 77.93492175 77.93492175]
```



Weak positive correlation. The higher the gdp per capita the less likey the mortality from COVID, this is somewhat expected, I would have expected the line to be steeper.

In [86]:

```
#Linear regression using all features

# need to drop mortality
covid_data_sns.drop(["date","iso_code","location","total_cases", "total_death


ml_data_r = covid_data_sns.drop(["date","iso_code","location","total_cases",

ml_data_r



X = ml_data_r.drop("total_deaths_per_100k",axis=1).values #drop target value
y = ml_data_r["total_deaths_per_100k"].values #target observations


#split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_
knn = KNeighborsClassifier(n_neighbors=5)

#fit the linear regression to the training data
reg_all = LinearRegression()
reg_all.fit(X_train, y_train)


#predict on the test set
y_pred = reg_all.predict(X_test)

r_score = reg_all.score(X_test, y_test)

print("Predictions: {}, Actual Values: {}".format(y_pred[:4], y_test[:4]))
print("There is a large gap between the predictions and test data")
print("The model only explains about %5.2f"%(r_score*100)+"% of mortality lev
```

```
Predictions: [ 43.48915448  46.5803073  101.09051936  36.24860988], Actual
Values: [2.31018829e-02 2.78315855e+00 4.03572231e+02 1.25033429e+01]
There is a large gap between the predictions and test data
The model only explains about 28.32% of mortality level variance
```

Results summary

Per the charts and analysis above, the results are not encoraging based on my inital hypothesis: that higher population density and lower GDP per capita for a country would have a negative impact on COVID mortality (higher deaths). I beleive there may be some outliers as we have seen in the scatter plot data that need to be removed which would potentially provide better results.

```
Overall, the data shows some correlations but fairly weak. the k score l
ooked promising at 81.13 and the Classification report F1 score of 0.81
 was ok to good performance and teh confusion matrix results were good
 (96 true positive and 19 for the false negative while there were 18 fal
se positives compared to 65 true negative.  I think this mat have beed s
kewed by the fairly wide grouping I gave for "high" mortality vs "low".

Given the flatness of the regression line it would make sense to review
 some of the outlier data and rerun maybe with a wider set of data.
```

In [ ]:

Insights

(Point out at least 5 insights in bullet points)

- Being able to use country data better in the machine learning would probably give better insights into the correlation
- Finding data and cleaning data is very challenging
- I really expected there to be a tighter correlation between the data and need more time to review the data for items that could be corrected
- Intersting excercise, seeing what others have put out online; it shows there is a very long way to go to get to an intermediate level
- The amount of information to learn about python is daunting and takes patience

References

HTML Code help: W3 Schools (https://www.w3schools.com/html/html_links.asp)

Our World in Data (OWID): https://ourworldindata.org/coronavirus#explore-the-global-situation (https://ourworldindata.org/coronavirus#explore-the-global-situation)

The World Bank GDP: https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=false (https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?year_high_desc=false)

Python:

formatting numbers: https://pythonguides.com/python-format-number-with-commas/#:~:text=Python%20format%20number%20with%20commas%20Let%20us%20see,comma (https://pythonguides.com/python-format-number-with-commas/#:~:text=Python%20format%20number%20with%20commas%20Let%20us%20see,comma

formatting dates: https://stackabuse.com/how-to-format-dates-in-python/ (https://stackabuse.com/how-to-format-dates-in-python/)

```
In [ ]:
```

```
In [ ]:
```