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 thought about several ideas. However, I explored three main ideas:

1. Predicting currency fx changes to maximise buys and sells.
As I have two children in university in Canada 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 and collecting the data was to burdensome for the scope of this project.
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 times. Researching for datasets I could not find anything current. Although, there are so me pay sites which may have data. As I would be required pay for the data I decided aga inst 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. I feel this lends it's self to the machine learning section as there are so many stories about the disease so would give m e an opportunity to show case what I learned.

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 with some work. 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. The data is quite clean so I was concerned about showing the data cleaning side of things

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)

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)

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 predictor 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. However, the earlier dates would certainly have 0 as there may not have been cases for a country at the start date of teh data.

Planned 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
- Mortality: high/low when cases per 100k deaths reaches a threshold to be set

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

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

I will need to understand the number of columns, count of records and the type of objects being used: like strings, dates, intergers and floats. I will review the null records and get a count to understand the completeness of the data, and build functions to assit with explori ng the data; like creating a rolling n day average and calulation for the total on a per 100, 00 of the population for comparative purposes.

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 #variable to use for select the number of top and bottom

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
```

As I create variables used in multiple places I endavour to capture them here so I can easily make updates. This saves time when when making changes.

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 shows there are a lot of columns of which most will be irrelivant for this work. There are 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_a

    # 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 for yeara 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 to the COVID data.

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 the 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]:

```
# 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")
```

I # out any testing logic here once the logic was working.

In [11]:

```
# 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
```

From the tests I can see both datasets contain quite alot of columns with data, so I will look at each source specifically the COVID data as this source will be critical to the project

COVID Raw Data

In [12]:

```
# 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:
(217509, 67)

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

There 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: 217509 entries, 0 to 217508
Data columns (total 15 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   iso_code                  217509 non-null  object
 1   location                  217509 non-null  object
 2   date                      217509 non-null  datetime64[ns]
 3   total_cases               208605 non-null  float64
 4   new_cases                 208278 non-null  float64
 5   total_deaths              189549 non-null  float64
 6   new_deaths                189456 non-null  float64
 7   icu_patients              27686 non-null   float64
 8   total_tests               79387 non-null   float64
 9   total_vaccinations        60807 non-null   float64
 10  people_fully_vaccinated   55366 non-null   float64
 11  population                216237 non-null  float64
 12  population_density        193145 non-null  float64
 13  gdp_per_capita            178046 non-null  float64
 14  extreme_poverty           116148 non-null  float64
dtypes: datetime64[ns](1), float64(12), object(2)
memory usage: 24.9+ MB
```

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

I will create a function that shows unique attributes in a list

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 reviewed. These are related to OWID codes used for regional aggregations of country data, they can be removed.

In [19]:

```
# 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: (13753, 15)
Non OWID data shape: (203756, 15)
```

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

In [20]:
```
# 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 [21]:
```
# 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 [22]:
```
# 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 203,756 rows of data and 15 columns
Number of columns appears manageable
```

Let mes deal with the NaN (null) data next

In [23]:
```
covid_data.head(5)
```

Out[23]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_patien
9381	ABW	Aruba	2020-03-13	2.0	2.0	NaN	NaN	Na
9382	ABW	Aruba	2020-03-14	2.0	0.0	NaN	NaN	Na
9383	ABW	Aruba	2020-03-15	2.0	0.0	NaN	NaN	Na
9384	ABW	Aruba	2020-03-16	2.0	0.0	NaN	NaN	Na
9385	ABW	Aruba	2020-03-17	3.0	1.0	NaN	NaN	Na

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

Out[24]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_pa
217504	ZWE	Zimbabwe	2022-09-14	256939.0	35.0	5596.0	0.0	
217505	ZWE	Zimbabwe	2022-09-15	256939.0	0.0	5596.0	0.0	
217506	ZWE	Zimbabwe	2022-09-16	256939.0	0.0	5596.0	0.0	
217507	ZWE	Zimbabwe	2022-09-17	256988.0	49.0	5598.0	2.0	
217508	ZWE	Zimbabwe	2022-09-18	256996.0	8.0	5598.0	0.0	

In [25]:

```
# 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                8580
new_cases                  8914
total_deaths              27442
new_deaths                27714
icu_patients             176070
total_tests              124557
total_vaccinations       150943
people_fully_vaccinated  156190
population                    0
population_density        12501
gdp_per_capita            27600
extreme_poverty           89498
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, I will need to convert these to 0's

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

In [26]:
```
#Which fields have nan's
covid_data[covid_data.isna().any(axis=1)]
```

Out[26]:

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	icu_pa
9381	ABW	Aruba	2020-03-13	2.0	2.0	NaN	NaN	
9382	ABW	Aruba	2020-03-14	2.0	0.0	NaN	NaN	
9383	ABW	Aruba	2020-03-15	2.0	0.0	NaN	NaN	
9384	ABW	Aruba	2020-03-16	2.0	0.0	NaN	NaN	
9385	ABW	Aruba	2020-03-17	3.0	1.0	NaN	NaN	
...	...	...	...	...	...	...	...	
217504	ZWE	Zimbabwe	2022-09-14	256939.0	35.0	5596.0	0.0	
217505	ZWE	Zimbabwe	2022-09-15	256939.0	0.0	5596.0	0.0	
217506	ZWE	Zimbabwe	2022-09-16	256939.0	0.0	5596.0	0.0	
217507	ZWE	Zimbabwe	2022-09-17	256988.0	49.0	5598.0	2.0	
217508	ZWE	Zimbabwe	2022-09-18	256996.0	8.0	5598.0	0.0	

194960 rows × 15 columns

In [27]:
```
# 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[27]:
```
9381      False
9382      False
9383      False
9384      False
9385      False
          ...
217504    False
217505    False
217506    False
217507    False
217508    False
Name: population, Length: 203756, 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 much

In [28]:
```
# drop the NaN for the population and gdp columns
covid_data_before_na = covid_data # allows me to have access to pre change da
covid_data = covid_data.dropna(subset=["population","population_density","gdp
#https://www.datasciencelearner.com/pandas-dropna-remove-nan-rows-python/
```

In [29]:
```
#review NaN data after removing records
covid_data.isna().sum()
```

Out[29]:
```
iso_code                      0
location                      0
date                          0
total_cases                3242
new_cases                  3552
total_deaths              15471
new_deaths                15733
icu_patients             147573
total_tests              100628
total_vaccinations       125828
people_fully_vaccinated  130768
population                    0
population_density            0
gdp_per_capita                0
extreme_poverty           61001
dtype: int64
```

Next I look at the value "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_pa
9381	ABW	Aruba	2020-03-13	2.0	2.0	0.0	0.0	
9382	ABW	Aruba	2020-03-14	2.0	0.0	0.0	0.0	
9383	ABW	Aruba	2020-03-15	2.0	0.0	0.0	0.0	
9384	ABW	Aruba	2020-03-16	2.0	0.0	0.0	0.0	
9385	ABW	Aruba	2020-03-17	3.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	
217504	ZWE	Zimbabwe	2022-09-14	256939.0	35.0	5596.0	0.0	
217505	ZWE	Zimbabwe	2022-09-15	256939.0	0.0	5596.0	0.0	
217506	ZWE	Zimbabwe	2022-09-16	256939.0	0.0	5596.0	0.0	
217507	ZWE	Zimbabwe	2022-09-17	256988.0	49.0	5598.0	2.0	
217508	ZWE	Zimbabwe	2022-09-18	256996.0	8.0	5598.0	0.0	

175259 rows × 15 columns

Final data set looks good and complete.

Final review of the COVID dataset

In [32]:

```
# 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)
```

```
0

        new_deaths  icu_patients  total_tests  total_vaccinations  \
9381           0.0           0.0          0.0                 0.0
9382           0.0           0.0          0.0                 0.0
9383           0.0           0.0          0.0                 0.0
9384           0.0           0.0          0.0                 0.0
9385           0.0           0.0          0.0                 0.0
...            ...           ...          ...                 ...
217504         0.0           0.0          0.0                 0.0
217505         0.0           0.0          0.0                 0.0
217506         0.0           0.0          0.0                 0.0
217507         2.0           0.0          0.0                 0.0

217508         0.0           0.0          0.0                 0.0

        people_fully_vaccinated  population  population_density  \
9381                        0.0    106536.0             584.800
9382                        0.0    106536.0             584.800
9383                        0.0    106536.0             584.800
```

Looks much better there are now no nulls for the Total_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)
```

First thing we can drop many of the year columns as the covid data does not go back that far. Clean up the column headers so they are consistent with the COVID data i.e no spaces

In [34]:

```
# 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]

gdp_data.drop('Unnamed:_66', axis=1, inplace=True)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 7 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
dtypes: float64(3), object(4)
memory usage: 14.7+ 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  
0  NY.GDP.MKTP.CD  3.310056e+09  2.496648e+09           NaN  
1  NY.GDP.MKTP.CD  9.975340e+11  9.216459e+11  1.082096e+12  
2  NY.GDP.MKTP.CD  1.879945e+10  2.011614e+10           NaN  
3  NY.GDP.MKTP.CD  7.945430e+11  7.844457e+11  8.358084e+11  
4  NY.GDP.MKTP.CD  6.930910e+10  5.361907e+10  7.254699e+10
```

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']

The gdp_data DataFrame has 7 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
2019	1.000000	0.999882	0.99950
2020	0.999882	1.000000	0.99963
2021	0.999500	0.999630	1.00000

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 %d
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): 193

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

No of rows:  175,259
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:  7
```

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
```

First create a rolling n_days average per 100k of teh population. I need use the record "date" and go back 14days. Except if the 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
9381     2020-02-28
9382     2020-02-29
9383     2020-03-01
9384     2020-03-02
9385     2020-03-03
            ...
217504   2022-08-31
217505   2022-09-01
217506   2022-09-02
217507   2022-09-03
217508   2022-09-04
Name: date, Length: 175259, dtype: datetime64[ns]
its working
```

In [41]:

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 175259 entries, 9381 to 217508
Data columns (total 16 columns):
 #   Column                    Non-Null Count   Dtype         
---  ------                    --------------   -----         
 0   iso_code                  175259 non-null  object        
 1   location                  175259 non-null  object        
 2   date                      175259 non-null  datetime64[ns]
 3   n_day_start_date          175259 non-null  datetime64[ns]
 4   total_cases               175259 non-null  float64       
 5   new_cases                 175259 non-null  float64       
 6   total_deaths              175259 non-null  float64       
 7   new_deaths                175259 non-null  float64       
 8   icu_patients              175259 non-null  float64       
 9   total_tests               175259 non-null  float64       
 10  total_vaccinations        175259 non-null  float64       
 11  people_fully_vaccinated   175259 non-null  float64       
 12  population                175259 non-null  float64       
 13  population_density        175259 non-null  float64       
 14  gdp_per_capita            175259 non-null  float64       
 15  extreme_poverty           175259 non-null  float64       
dtypes: datetime64[ns](2), float64(12), object(2)
memory usage: 22.7+ MB
None

the new column is now appearing
```

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

Create the total_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
----------------------------------------------------------------------------
-------------------------
```

For the regressions I need to 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 and create the top deaths and the bottom deaths sets of data. In the top_n death set look at the minimum value to set the lower threshold for the "HigH" mortalility classificaion In the bottom_n deaths look at the max value to set the upper 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: 1.478435070432998
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 to cahnge
```

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

Add the 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, "

covid_data.tail()
```

Out[47]:

	iso_code	location	date	n_day_start_date	total_cases	new_cases	total_deaths_
217504	ZWE	Zimbabwe	2022-09-14	2022-08-31	256939.0	35.0	
217505	ZWE	Zimbabwe	2022-09-15	2022-09-01	256939.0	0.0	
217506	ZWE	Zimbabwe	2022-09-16	2022-09-02	256939.0	0.0	
217507	ZWE	Zimbabwe	2022-09-17	2022-09-03	256988.0	49.0	
217508	ZWE	Zimbabwe	2022-09-18	2022-09-04	256996.0	8.0	

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)
```

ases	total_deaths	...	14_days_rolling_new_deaths	icu_patients	total_tests	total_vaccinations	people_fully_vaccina
2.0	0.0	...	2.0	0.0	0.0	0.0	
2.0	0.0	...	2.0	0.0	0.0	0.0	
2.0	0.0	...	2.0	0.0	0.0	0.0	
2.0	0.0	...	2.0	0.0	0.0	0.0	
3.0	0.0	...	3.0	0.0	0.0	0.0	
4.0	0.0	...	4.0	0.0	0.0	0.0	
4.0	0.0	...	4.0	0.0	0.0	0.0	

In [49]:
```
covid_data_country.tail(20)
```

220.0	228.0	...	220.0	0.0	0.0	173244.0	835
220.0	228.0	...	220.0	0.0	0.0	0.0	
220.0	228.0	...	220.0	0.0	0.0	0.0	
178.0	228.0	...	178.0	0.0	0.0	173263.0	835
178.0	228.0	...	178.0	0.0	0.0	173271.0	835
122.0	228.0	...	122.0	0.0	0.0	173283.0	835
122.0	228.0	...	122.0	0.0	0.0	173306.0	835
122.0	228.0	...	122.0	0.0	0.0	173334.0	835

I observe the earlier data in the .head() function shows empty mortality data but the later .tail() "high"'s can be seen and the matches the logic expected

In [50]:
```
# are there any null values introduced by this process
print(covid_data.shape)
print()
print(covid_data.isna().sum())
```

```
(175259, 21)

iso_code                          0
location                          0
date                              0
n_day_start_date                  0
total_cases                       0
new_cases                         0
total_deaths_per_100k             0
total_cases_per_100k              0
14_days_rolling_new_cases         0
total_deaths                      0
new_deaths                        0
14_days_rolling_new_deaths        0
icu_patients                      0
total_tests                       0
total_vaccinations                0
people_fully_vaccinated           0
population                        0
population_density                0
gdp_per_capita                    0
extreme_poverty                   0
mortality                     36069
dtype: int64
```

In [51]:
```
# need to balnk out the NaN's created in the mortality description field
covid_data = covid_data.fillna("")

print(covid_data.shape)
print()
```

```
(175259, 21)
```

All the new columns and records are the same in the datset now. Let's review the .corr() function for some quick insights

In [52]:
```
# Correlations of the covid_data
covid_data.corr()
```

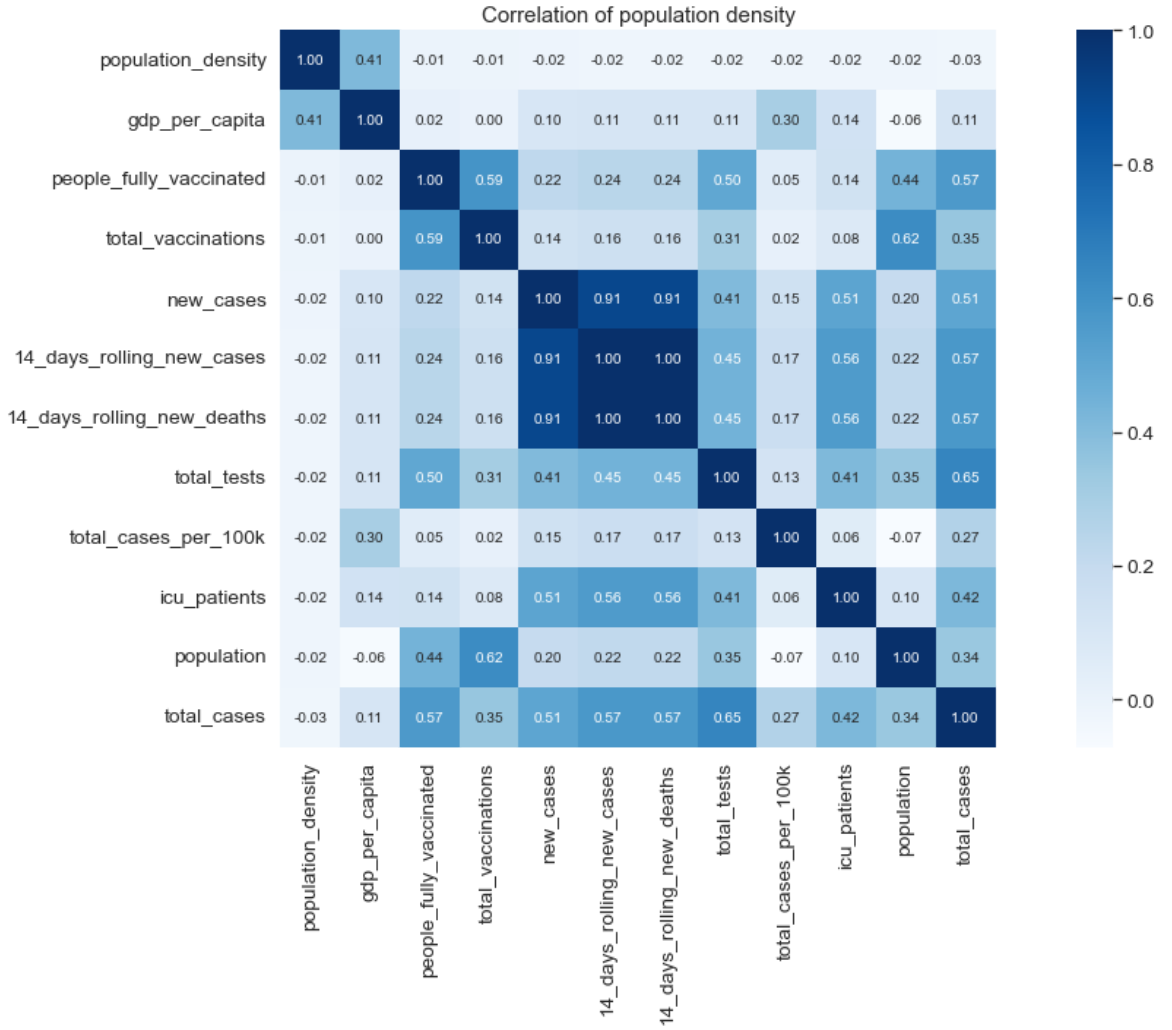
Out[52]:

	total_cases	new_cases	total_deaths_per_100k	total_cases_per_10
total_cases	1.000000	0.508076	0.279401	0.2661
new_cases	0.508076	1.000000	0.143518	0.1483
total_deaths_per_100k	0.279401	0.143518	1.000000	0.6352
total_cases_per_100k	0.266118	0.148353	0.635226	1.0000
14_days_rolling_new_cases	0.569669	0.905137	0.159026	0.1716
total_deaths	0.897248	0.453178	0.359821	0.1586
new_deaths	0.413840	0.555171	0.135286	0.0242
14_days_rolling_new_deaths	0.569669	0.905137	0.159026	0.1716
icu_patients	0.418339	0.510785	0.110853	0.0578
total_tests	0.650437	0.407629	0.144085	0.1261
total_vaccinations	0.348946	0.142295	0.026652	0.0174
people_fully_vaccinated	0.565007	0.220498	0.079058	0.0545
population	0.339868	0.200565	-0.048390	-0.0705
population_density	-0.025283	-0.015911	-0.069656	-0.0182
gdp_per_capita	0.113198	0.103189	0.141799	0.2982
extreme_poverty	-0.068280	-0.061047	-0.247828	-0.2379

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

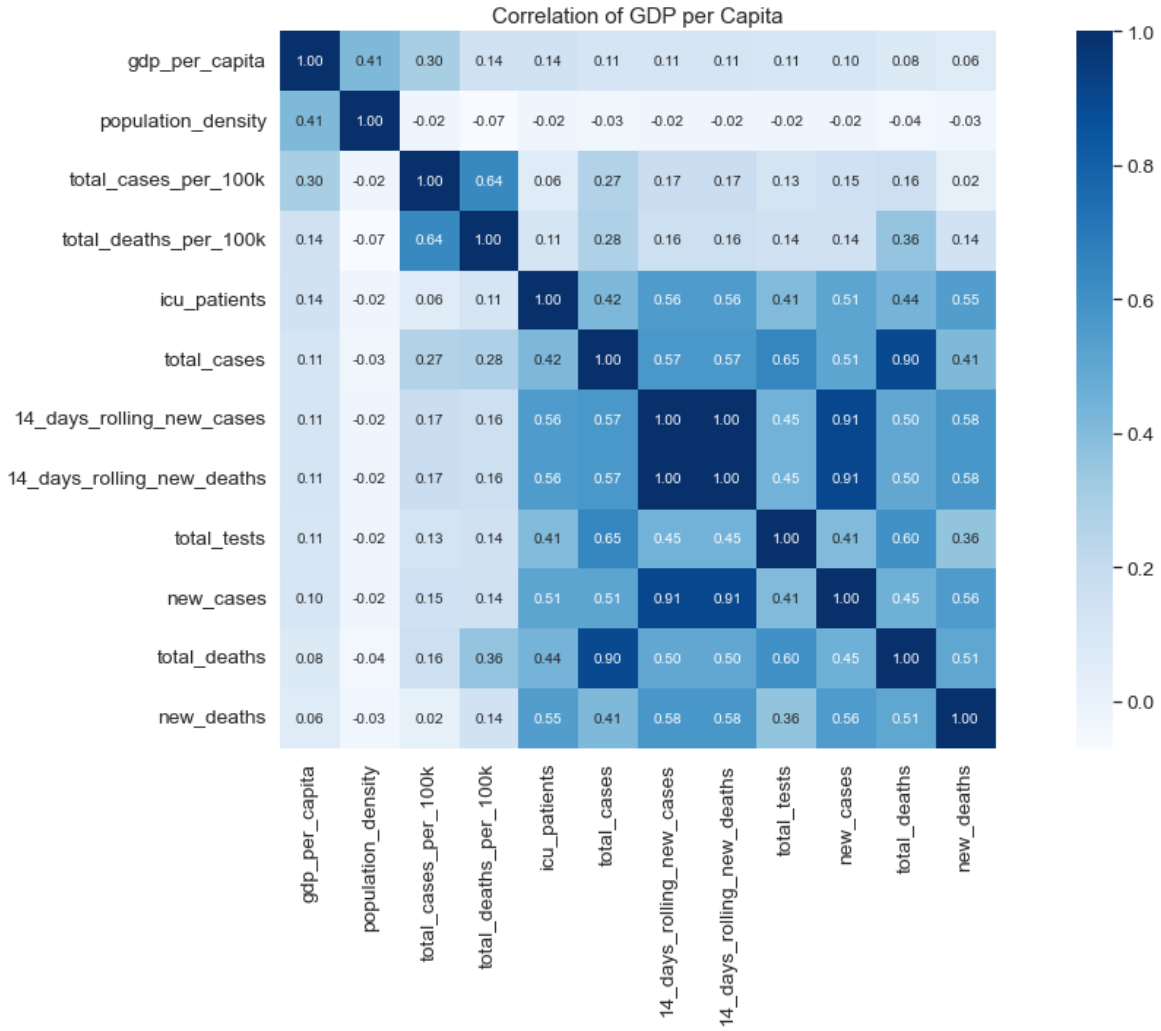
In [53]:

```
#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()
```



In [54]:

```
#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 [55]:
```
# 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()
```

Summary of data exploration and preparation: the data 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 data looks to be a good representation of the data. Next step is to pefrom regession and machine learning, although given the low correlation I am seeing not sure how fruitful it will be

Analysis

Basic Charts

In [56]:
```
# 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)
```

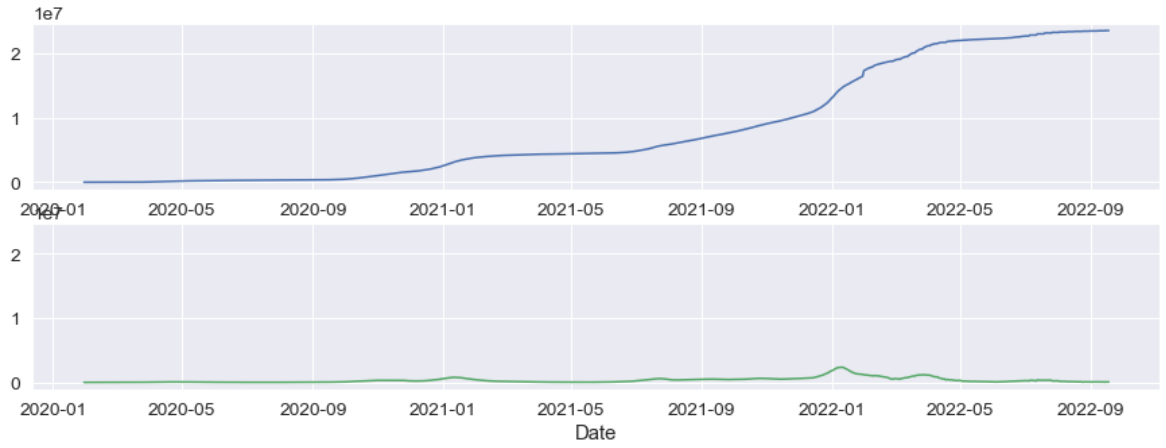
```
ABW AFG AGO ALB ARE ARG ARM ATG AUS AUT AZE BDI BEL BEN BFA BGD BGR BHR BHS
BIH BLR BLZ BMU BOL BRA BRB BRN BTN BWA CAF CAN CHE CHL CHN CIV CMR COD COG
COL COM CPV CRI CYM CYP CZE DEU DJI DMA DNK DOM DZA ECU EGY ERI ESP EST ETH
FIN FJI FRA FSM GAB GBR GEO GHA GIN GMB GNB GNQ GRC GRD GTM GUY HKG HND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

Select a country code above to test the chart below

In [57]:

```
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 [58]:

```
# 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
```

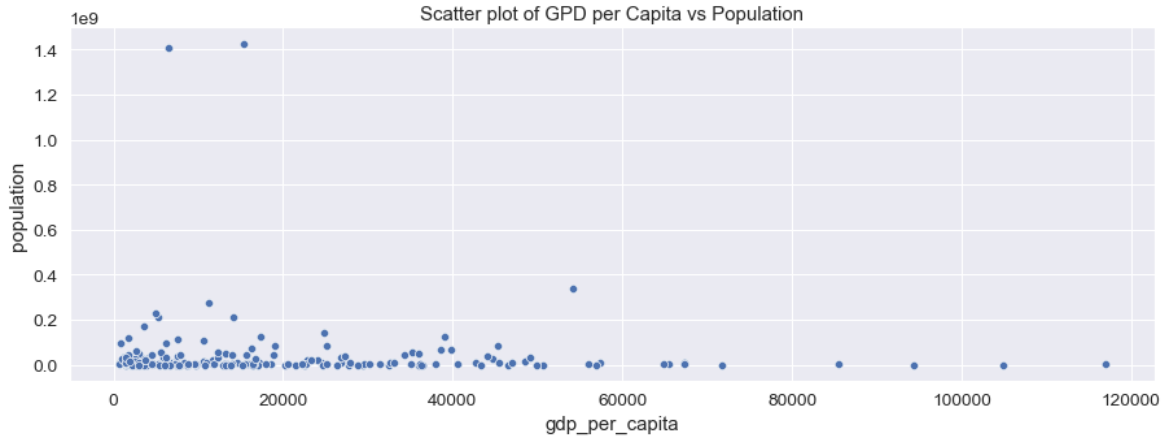
In [59]:

```
# 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
(568, 13)
----------------------------------------------------------------------------
---------------------------

            date iso_code     location  total_cases  total_cases_per_100k
\
9674  2020-12-31      ABW        Aruba       5489.0           5152.249005
10039 2021-12-31      ABW        Aruba      20461.0          19205.714500
10298 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_densit
y  \
9674           49.0              45.993842    106536.0             584.80
0
10039         181.0             169.895622    106536.0             584.80
0
```

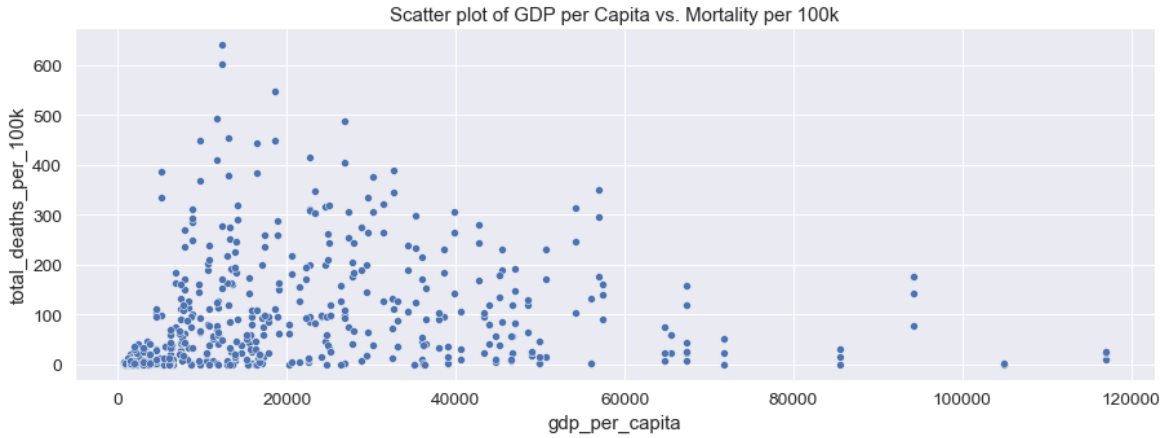
I can see that we are getting multiple dates and requested columns of data back so good to move forward

In [60]:
```
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 [61]:
```
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()
```

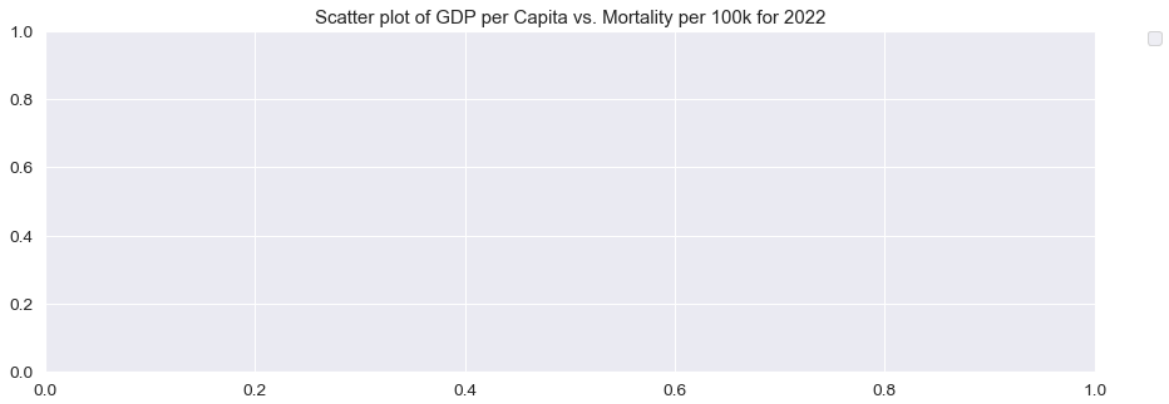


looks like an intersting visual and that we can what appears to be a pattern between countries, GDP and mortality

In [62]:
```
#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 [63]:
```
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 [64]:
```
# 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 [65]:

```
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_1
00k  \
130375 2020-12-31      MNE     Montenegro      48247.0           7684.368
624
115863 2020-12-31      LUX     Luxembourg      46415.0           7260.046
205
168755 2020-12-31      SMR     San Marino       2333.0           6913.411
960
50029  2020-12-31      CZE        Czechia     718661.0           6837.390
291
15336  2020-12-31      BHR        Bahrain      92675.0           6333.439
261
72445  2020-12-31      GEO        Georgia     227420.0           6051.655
411
```

In [66]:

```
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_1
00k  \
130375 2020-12-31      MNE     Montenegro      48247.0           7684.368
624
115863 2020-12-31      LUX     Luxembourg      46415.0           7260.046
205
```

```
168755 2020-12-31      SMR     San Marino       2333.0           6913.411
960
50029  2020-12-31      CZE        Czechia     718661.0           6837.390
291
15336  2020-12-31      BHR        Bahrain      92675.0           6333.439
261
72445  2020-12-31      GEO        Georgia     227420.0           6051.655
411
```

In [67]:

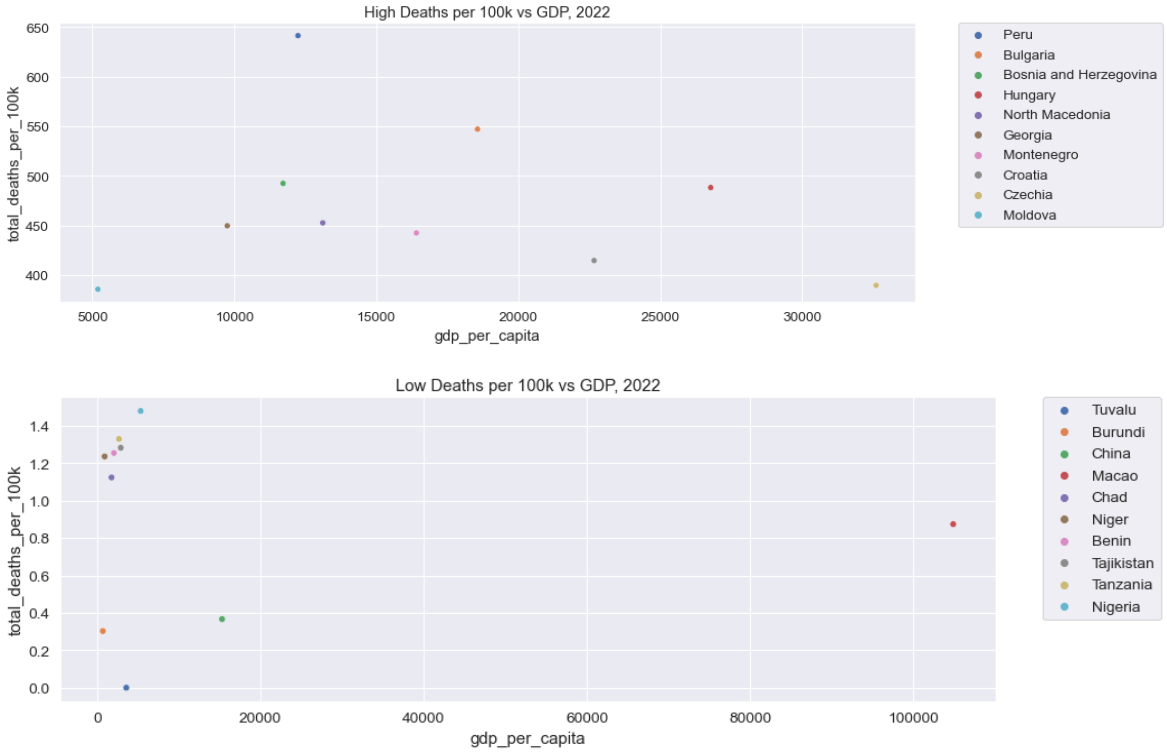
```
# min value in the Top mortality (top deaths) data
print("min of 2020 top deaths/ 100k: "+str(top_df_2020_deaths_per_100k["total
print("max of 2020 top deaths/ 100k: "+str(top_df_2020_deaths_per_100k["total
print("-"*100)
print("min of 2021 top deaths/ 100k: "+str(top_df_2021_deaths_per_100k["total
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(bot_df_2022_deaths_per_100k["to
print("min of 2022 lowest deaths/ 100k: "+str(bot_df_2020_deaths_per_100k["to
```

```
min of 2020 top deaths/ 100k: 110.02244016295403
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: 1.478435070432998
min of 2022 lowest deaths/ 100k: 0.0
```

In [68]:

```
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
```

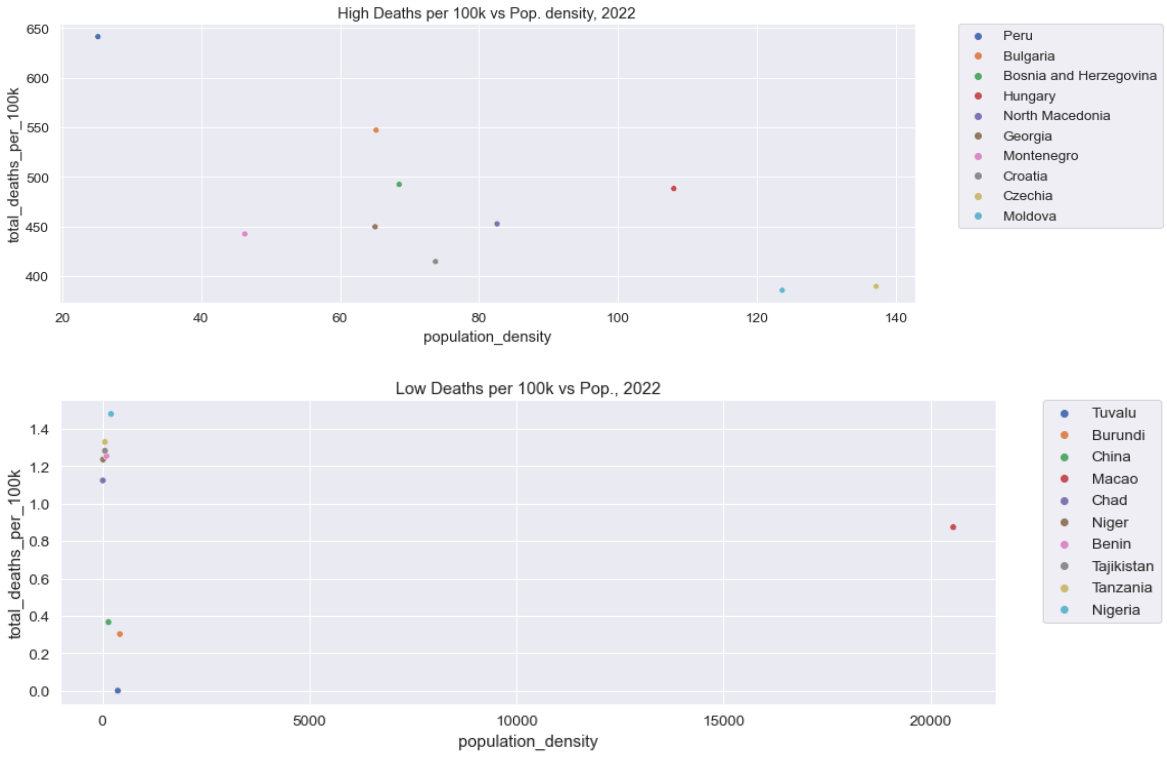


this looks strange, high mortality and gdp does not look related. it appears from the top chart that the higher mortality countries also have higher gdp, for the most part these countries look like smaller nations. lets look by population density

In [69]:

```
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
```



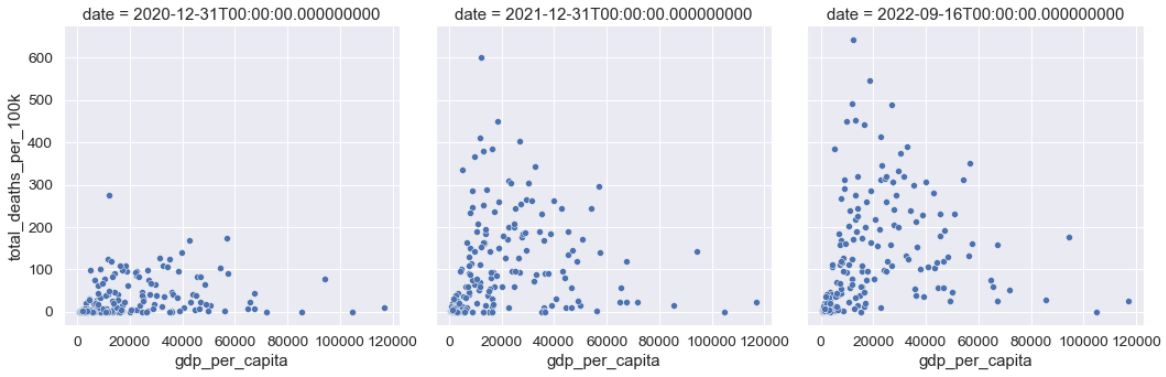
this looks strange as well, higher mortality and lower density looks negatively related, it's difficult to tell because of the outlier, Macao, in the lower chart. it appears from the top chart that the lesser dense countries also have higer mortality, for the most part these countries look like smaller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

In [70]:
```
# 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: 24.055793038171487
max 2020 top deaths/ 100k: 174.83553606353345
----------------------------------------------------------------------------
-------------------------
max 2022 lowest deaths/ 100k: 6.5339380778536755
min 2022 lowest deaths/ 100k: 0.36650704501721454
```

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

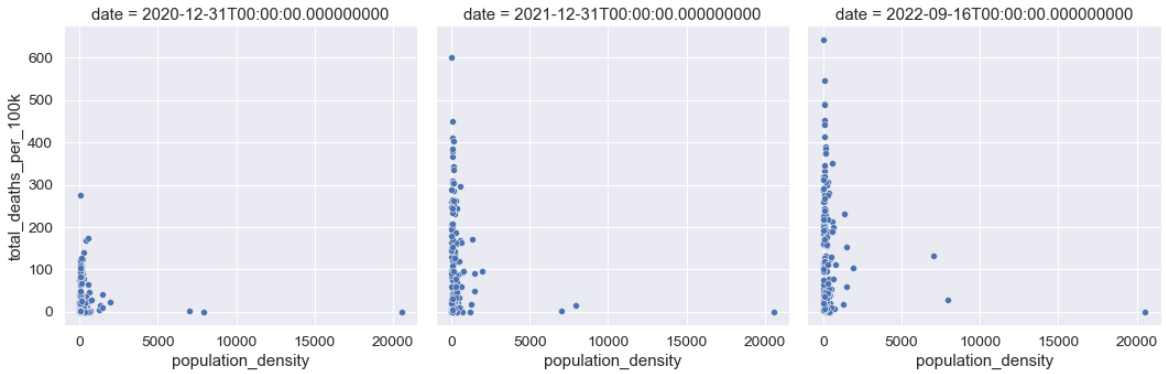
In [71]:
```
#Mortality per 100k and GDP per capita

#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 these are expected results; 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 [72]:

```
#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 not have a more significant impact on mortality

In [73]:
```
covid_temp = covid_data_sns
covid_temp.sort_values("population_density", axis = 0, ascending = False, inp
covid_temp
```

```
C:\Users\Phillip\anaconda3\lib\site-packages\pandas\util\_decorators.py:31
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  return func(*args, **kwargs)
```

Out[73]:

	date	iso_code	location	total_cases	total_cases_per_100k	total_deaths	total_deat
117458	2022-09-16	MAC	Macao	793.0	115.495473	6.0	
117199	2021-12-31	MAC	Macao	79.0	11.505854	0.0	
116834	2020-12-31	MAC	Macao	46.0	6.699611	0.0	
175866	2022-09-16	SGP	Singapore	1871900.0	34324.116180	1605.0	
175607	2021-12-31	SGP	Singapore	279405.0	5123.313041	828.0	
...	...	...	...	...	...	...	
135346	2021-12-31	NAM	Namibia	147974.0	5848.425647	3633.0	
135605	2022-09-16	NAM	Namibia	169253.0	6689.442646	4077.0	
130083	2022-09-16	MNG	Mongolia	981963.0	29331.748603	2130.0	
129824	2021-12-31	MNG	Mongolia	692621.0	20688.951670	1986.0	
129459	2020-12-31	MNG	Mongolia	1195.0	35.695275	1.0	

568 rows × 13 columns

Regression analysis

In [74]:
```
# Machine Learning
```

In [75]:
```
# Machine Learning KNN data
```

In [76]:
```
covid_data_sns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 568 entries, 117458 to 129459
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   date                     568 non-null    datetime64[ns]
 1   iso_code                 568 non-null    object        
 2   location                 568 non-null    object        
 3   total_cases              568 non-null    float64       
 4   total_cases_per_100k     568 non-null    float64       
 5   total_deaths             568 non-null    float64       
 6   total_deaths_per_100k    568 non-null    float64       
 7   population               568 non-null    float64       
 8   population_density       568 non-null    float64       
 9   gdp_per_capita           568 non-null    float64       
 10  extreme_poverty          568 non-null    float64       
 11  people_fully_vaccinated  568 non-null    float64       
 12  mortality                568 non-null    object        
dtypes: datetime64[ns](1), float64(9), object(3)
memory usage: 62.1+ KB
```

In [77]:
```
date_list = unique(covid_data_sns["date"])
print("we have the expected 3 dates selected")
```

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

In [78]:
```
#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[78]:

	total_cases_per_100k	total_deaths_per_100k	population	population_density	gdp_per_
117458	115.495473	0.873862	686607.0	20546.766	10486
117199	11.505854	0.000000	686607.0	20546.766	10486
116834	6.699611	0.000000	686607.0	20546.766	10486
175866	34324.116180	29.430101	5453600.0	7915.731	8553
175607	5123.313041	15.182632	5453600.0	7915.731	8553
...	...	...	...	...	
135346	5848.425647	143.588268	2530151.0	3.078	954
135605	6689.442646	161.136628	2530151.0	3.078	954
130083	29331.748603	63.624214	3347782.0	1.980	1184
129824	20688.951670	59.322859	3347782.0	1.980	1184
129459	35.695275	0.029871	3347782.0	1.980	1184

568 rows × 8 columns

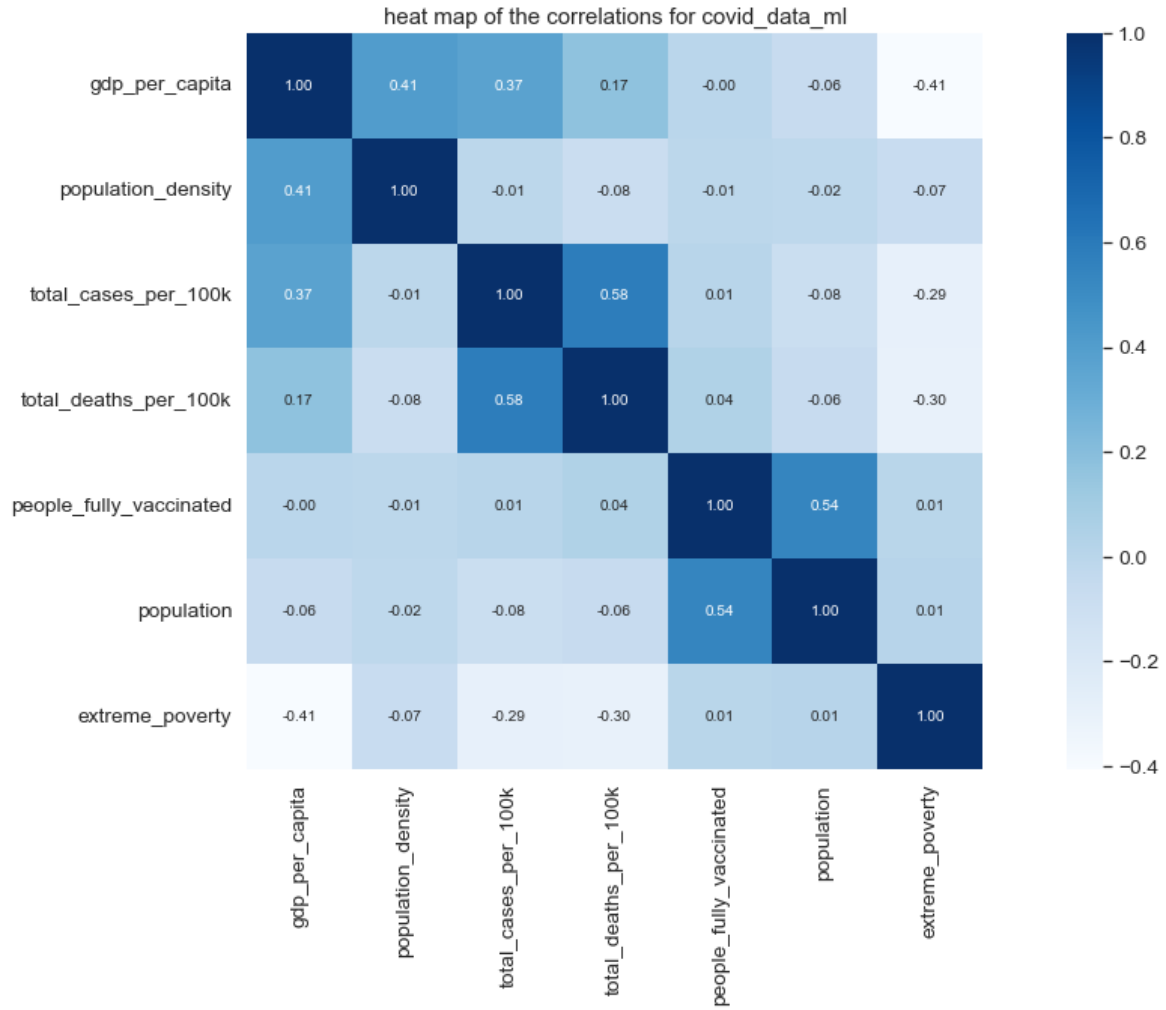
In [79]:
```
covid_data_ml2.corr()
```

Out[79]:

	total_cases_per_100k	total_deaths_per_100k	population	population_de
total_cases_per_100k	1.000000	0.583920	-0.084512	-0.00
total_deaths_per_100k	0.583920	1.000000	-0.058735	-0.07
population	-0.084512	-0.058735	1.000000	-0.02
population_density	-0.009071	-0.076922	-0.021465	1.00
gdp_per_capita	0.372847	0.171162	-0.055781	0.40
extreme_poverty	-0.289993	-0.301442	0.011374	-0.07
people_fully_vaccinated	0.006755	0.041362	0.543958	-0.00

In [80]:

```
#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 [81]:

```
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=3)

#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_))
```

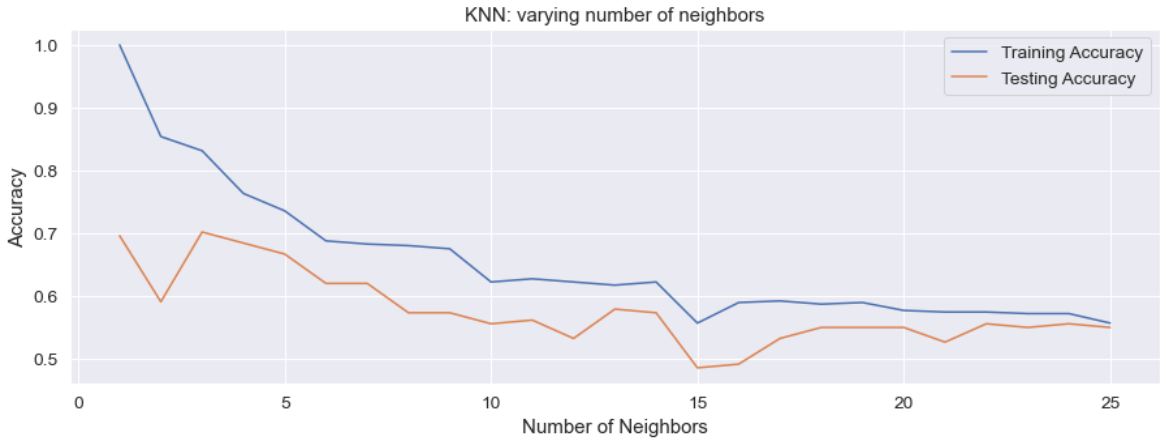
```
training and testing the data

The knn score:
0.7017543859649122

Confusion matrix:
[[18  7 15]
 [17 54  3]
 [ 7  2 48]]

Classification report:
              precision    recall  f1-score   support

                   0.43      0.45      0.44        40
        high       0.86      0.73      0.79        74
         low       0.73      0.84      0.78        57

    accuracy                           0.70       171
   macro avg       0.67      0.67      0.67       171
weighted avg       0.71      0.70      0.70       171
```

The knn score suggest there are somewhat 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 [82]:
```
#model complexity
train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1,26) #(1,26)
```

In [83]:
```
# 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 [84]:
```
# 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()
```



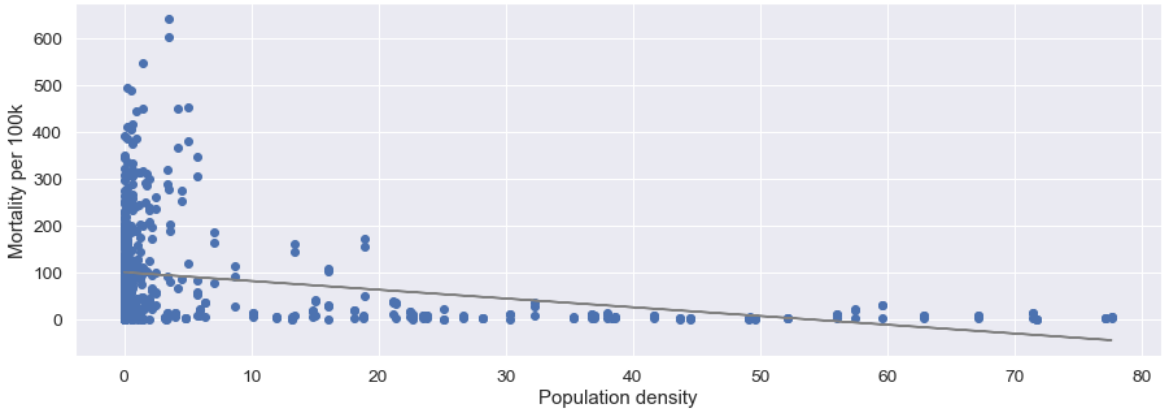
Tried with 5, 6, then 13 but this shows k of 3 is a good choice as this displays the highest testing accuracy and training score

Supervised learning with regression

In [85]:

```
#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()
```

```
[99.77058688 99.77058688 99.77058688 99.77058688 99.77058688 99.77058688
 99.77058688 99.77058688 99.77058688 99.77058688]
```

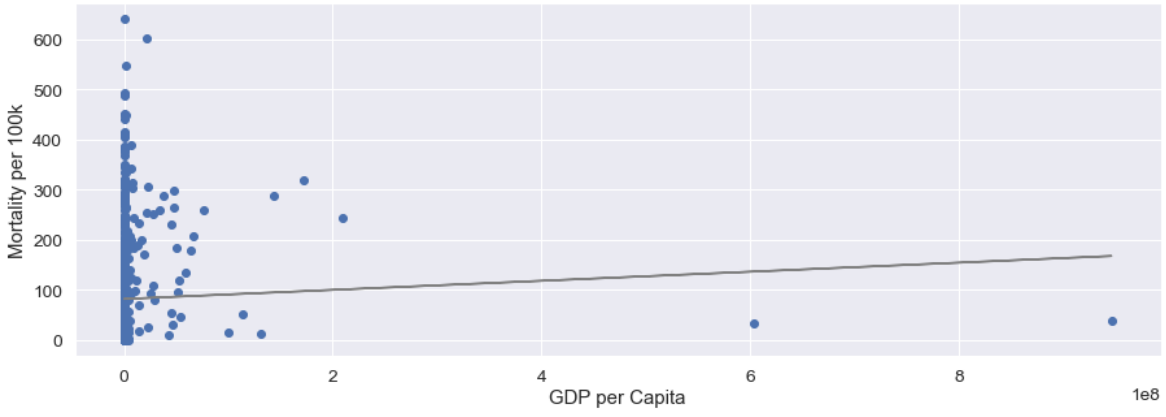


Weak negative correlation. The higher the population density the less likey the mortality from COVID, this is unexpected. I would expect the line to be positive

In [86]:

```
# 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()
```

```
[82.12525445 82.12525445 82.12525445 82.12525445 82.53305718 82.12525445
 82.12525445 82.12525445 82.12525445 82.5474371  82.12525445]
```



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 and negative.

In [87]:

```
#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 are large gaps between the predictions and test data")
print("The model only explains about %5.2f"%(r_score*100)+"% of mortality lev
```

```
Predictions: [138.90695365 135.60344594   6.59927013 256.25435076], Actual
Values: [263.50216941 310.60841126   1.49632314 333.87747515]
There are large gaps between the predictions and test data
The model only explains about 51.84% of mortality level variance
```

Results summary

Per the charts and analysis above, the results are not encoraging based on my inital hy pothesis: that higher population density and lower GDP per capita for a country would hav e a negative impact on COVID mortality (higher deaths). I beleive there may be some out liers, as seen in the scatter plot data, that should be reviewed further. This would potential ly provide better results.

Overall, the data shows some correlations but fairly weak. the k score looked promising at .702 and the Classification report F1 score of 0.79/0.78 was ok performance. The confu sion matrix results were ok (18 true positive and 15 for the false negative while 7 false po sitives compared to 48 true negative. I think this may have been skewed by the fairly wid

e grouping I gave for "high" mortality vs "low".

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

I reviewed the hypertuning and I' not condifent it will imporve the performace of the rep ort by much and given the lower accuracy I do not think over fitting is an issue. For those reasons I chose not to use the hyper parameter tuning and focus on the machine learning module

Insights

(Point out at least 5 insights in bullet points)

- I would like to have used the country data better 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/)

Other Techniques Demonstrated

Merge DataFrames COVID_data to GDP data

In [88]:
```
covid__data_merge1 = covid_data[covid_data["date"].isin([last_date_n])]

#rename location to Country_Name
covid__data_merge1.rename(columns = {'location':'Country_Name'}, inplace = Tr

print("\nAfter modifying column:\n", covid__data_merge1.columns)
```

```
After modifying column:
 Index(['iso_code', 'Country_Name', 'date', 'n_day_start_date', 'total_case
s',
       'new_cases', 'total_deaths_per_100k', 'total_cases_per_100k',
       '14_days_rolling_new_cases', 'total_deaths', 'new_deaths',
       '14_days_rolling_new_deaths', 'icu_patients', 'total_tests',
       'total_vaccinations', 'people_fully_vaccinated', 'population',
       'population_density', 'gdp_per_capita', 'extreme_poverty', 'mortalit
y'],
      dtype='object')

C:\Users\Phillip\anaconda3\lib\site-packages\pandas\core\frame.py:5039: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  return super().rename(
```

In [89]:
```
gdp_data.head()
```

Out[89]:

	Country_Name	Country_Code	Indicator_Name	Indicator_Code	2019	202
0	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD	3.310056e+09	2.496648e+0
1	Africa Eastern and Southern	AFE	GDP (current US$)	NY.GDP.MKTP.CD	9.975340e+11	9.216459e+1
2	Afghanistan	AFG	GDP (current US$)	NY.GDP.MKTP.CD	1.879945e+10	2.011614e+1
3	Africa Western and Central	AFW	GDP (current US$)	NY.GDP.MKTP.CD	7.945430e+11	7.844457e+1
4	Angola	AGO	GDP (current US$)	NY.GDP.MKTP.CD	6.930910e+10	5.361907e+1

In [90]:
```
#GDP NaN values
gdp_data.isna().sum()
```

Out[90]:
```
Country_Name       0
Country_Code       0
Indicator_Name     0
Indicator_Code     0
2019              11
2020              15
2021              37
dtype: int64
```

In [91]:
```
#deal with Nan values for GDP data
#fill NaN with an average
fill_value = pd.DataFrame({col: gdp_data.mean(axis=1) for col in gdp_data.col
gdp_data.fillna(fill_value, inplace=True)


gdp_data.head()
```

```
C:\Users\Phillip\AppData\Local\Temp/ipykernel_4320/1101080937.py:3: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeric
_only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
  fill_value = pd.DataFrame({col: gdp_data.mean(axis=1) for col in gdp_dat
a.columns})
```

Out[91]:

	Country_Name	Country_Code	Indicator_Name	Indicator_Code	2019	202
0	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD	3.310056e+09	2.496648e+0
1	Africa Eastern and Southern	AFE	GDP (current US$)	NY.GDP.MKTP.CD	9.975340e+11	9.216459e+1
2	Afghanistan	AFG	GDP (current US$)	NY.GDP.MKTP.CD	1.879945e+10	2.011614e+1
3	Africa Western and Central	AFW	GDP (current US$)	NY.GDP.MKTP.CD	7.945430e+11	7.844457e+1
4	Angola	AGO	GDP (current US$)	NY.GDP.MKTP.CD	6.930910e+10	5.361907e+1

In [92]:

```
#merge Covid to Gdp
df_covid_merge =pd.merge(covid__data_merge1,gdp_data, how="outer")

print(df_covid_merge.columns)
df_covid_merge.head()
```

```
Index(['iso_code', 'Country_Name', 'date', 'n_day_start_date', 'total_cas
es',
       'new_cases', 'total_deaths_per_100k', 'total_cases_per_100k',
       '14_days_rolling_new_cases', 'total_deaths', 'new_deaths',
       '14_days_rolling_new_deaths', 'icu_patients', 'total_tests',
       'total_vaccinations', 'people_fully_vaccinated', 'population',
       'population_density', 'gdp_per_capita', 'extreme_poverty', 'mortal
ity',
       'Country_Code', 'Indicator_Name', 'Indicator_Code', '2019', '202
0',
       '2021'],
      dtype='object')
```

Out[92]:

	iso_code	Country_Name	date	n_day_start_date	total_cases	new_cases	total_deaths_
0	ABW	Aruba	2022-09-16	2022-09-02	42970.0	0.0	21
1	AFG	Afghanistan	2022-				

In [93]:

```
#regedit to find a strin of characters:

# I did not use this code in the filtering above becuase tehre were other eas

test = "how am i to find OWID_ or owid or 'OWID' any other set of strings"

headers=covid_data_raw["iso_code"].tolist()
unique_headers = unique(headers)


owid = re.findall("OWID",str(headers))
print(unique(owid))

if owid:
  print("I found a match")
else:
  print("No match")
```

```
AFG OWID_AFR ALB DZA AND AGO AIA ATG ARG ARM ABW OWID_ASI AUS AUT AZE BHS B
HR BGD BRB BLR BEL BLZ BEN BMU BTN BOL BES BIH BWA BRA VGB BRN BGR BFA BDI
KHM CMR CAN CPV CYM CAF TCD CHL CHN COL COM COG COK CRI CIV HRV CUB CUW CYP
CZE COD DNK DJI DMA DOM ECU EGY SLV GNQ ERI EST SWZ ETH OWID_EUR OWID_EUN F
RO FLK FJI FIN FRA PYF GAB GMB GEO DEU GHA GIB GRC GRL GRD GUM GTM GGY GIN
GNB GUY HTI OWID_HIC HND HKG HUN ISL IND IDN OWID_INT IRN IRQ IRL IMN ISR I
TA JAM JPN JEY JOR KAZ KEN KIR OWID_KOS KWT KGZ LAO LVA LBN LSO LBR LBY LIE
LTU OWID_LIC OWID_LMC LUX MAC MDG MWI MYS MDV MLI MLT MHL MRT MUS MEX FSM M
DA MCO MNG MNE MSR MAR MOZ MMR NAM NRU NPL NLD NCL NZL NIC NER NGA NIU OWID
_NAM PRK MKD OWID_CYN MNP NOR OWID_OCE OMN PAK PLW PSE PAN PNG PRY PER PHL
PCN POL PRT PRI QAT ROU RUS RWA SHN KNA LCA SPM VCT WSM SMR STP SAU SEN SRB
SYC SLE SGP SXM SVK SVN SLB SOM ZAF OWID_SAM KOR SSD ESP LKA SDN SUR SWE CH
E SYR TWN TJK TZA THA TLS TGO TKL TON TTO TUN TUR TKM TCA TUV UGA UKR ARE G
BR USA VIR OWID_UMC URY UZB VUT VAT VEN VNM WLF ESH OWID_WRL YEM ZMB ZWE
OWID
None
I found a match
```