Covid - 19 Data Analysis Project using Python

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [7]:
```
url="https://raw.githubusercontent.com/SR1608/Datasets/main/covid-data.csv"
df=pd.read_csv(url)
df
```

Out[7]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	gdp_per_capita	extreme_poverty	cardiovasc_death_rate	diabetes
0	AFG	Asia	Afghanistan	31/12/19	NaN	0.0	NaN	NaN	0.0	NaN	...	1803.987	NaN	597.029	
1	AFG	Asia	Afghanistan	01/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	1803.987	NaN	597.029	
2	AFG	Asia	Afghanistan	02/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	1803.987	NaN	597.029	
3	AFG	Asia	Afghanistan	03/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	1803.987	NaN	597.029	
4	AFG	Asia	Afghanistan	04/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	1803.987	NaN	597.029	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
57389	NaN	NaN	International	13/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	NaN	NaN	NaN	
57390	NaN	NaN	International	14/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	NaN	NaN	NaN	
57391	NaN	NaN	International	15/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	NaN	NaN	NaN	
57392	NaN	NaN	International	16/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	NaN	NaN	NaN	
57393	NaN	NaN	International	17/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	NaN	NaN	NaN	

57394 rows × 49 columns

High Level Data Understanding:

In [8]:
```
#Find no. of rows & columns in the dataset

print("no_of_rows = ",df.shape[0])
print("no_of_cols = ",df.shape[1])
```

```
no_of_rows =  57394
no_of_cols =  49
```

In [9]:
```
#Data types of columns

df.dtypes
```

Out[9]:
```
iso_code                                object
continent                               object
location                                object
date                                    object
total_cases                            float64
new_cases                              float64
new_cases_smoothed                     float64
total_deaths                           float64
new_deaths                             float64
new_deaths_smoothed                    float64
total_cases_per_million                float64
new_cases_per_million                  float64
new_cases_smoothed_per_million         float64
total_deaths_per_million               float64
new_deaths_per_million                 float64
new_deaths_smoothed_per_million        float64
reproduction_rate                      float64
icu_patients                           float64
icu_patients_per_million               float64
hosp_patients                          float64
hosp_patients_per_million              float64
weekly_icu_admissions                  float64
weekly_icu_admissions_per_million      float64
weekly_hosp_admissions                 float64
weekly_hosp_admissions_per_million     float64
total_tests                            float64
new_tests                              float64
total_tests_per_thousand               float64
new_tests_per_thousand                 float64
new_tests_smoothed                     float64
new_tests_smoothed_per_thousand        float64
tests_per_case                         float64
positive_rate                          float64
stringency_index                       float64
population                             float64
population_density                     float64
median_age                             float64
aged_65_older                          float64
aged_70_older                          float64
gdp_per_capita                         float64
extreme_poverty                        float64
cardiovasc_death_rate                  float64
diabetes_prevalence                    float64
female_smokers                         float64
male_smokers                           float64
handwashing_facilities                 float64
hospital_beds_per_thousand             float64
life_expectancy                        float64
human_development_index                float64
dtype: object
```

In [10]:
```
#Info of data in dataframe

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57394 entries, 0 to 57393
Data columns (total 49 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   iso_code                            57071 non-null  object
 1   continent                           56748 non-null  object
 2   location                            57394 non-null  object
 3   date                                57394 non-null  object
 4   total_cases                         53758 non-null  float64
 5   new_cases                           56465 non-null  float64
 6   new_cases_smoothed                  55652 non-null  float64
 7   total_deaths                        44368 non-null  float64
 8   new_deaths                          56465 non-null  float64
 9   new_deaths_smoothed                 55652 non-null  float64
 10  total_cases_per_million             53471 non-null  float64
 11  new_cases_per_million               56401 non-null  float64
 12  new_cases_smoothed_per_million      55587 non-null  float64
 13  total_deaths_per_million            44096 non-null  float64
 14  new_deaths_per_million              56401 non-null  float64
 15  new_deaths_smoothed_per_million     55587 non-null  float64
 16  reproduction_rate                   37696 non-null  float64
 17  icu_patients                        4490 non-null   float64
 18  icu_patients_per_million            4490 non-null   float64
 19  hosp_patients                       5005 non-null   float64
 20  hosp_patients_per_million           5005 non-null   float64
 21  weekly_icu_admissions               357 non-null    float64
 22  weekly_icu_admissions_per_million   357 non-null    float64
 23  weekly_hosp_admissions              645 non-null    float64
 24  weekly_hosp_admissions_per_million  645 non-null    float64
 25  total_tests                         22017 non-null  float64
 26  new_tests                           21787 non-null  float64
 27  total_tests_per_thousand            22017 non-null  float64
 28  new_tests_per_thousand              21787 non-null  float64
 29  new_tests_smoothed                  24612 non-null  float64
 30  new_tests_smoothed_per_thousand     24612 non-null  float64
 31  tests_per_case                      22802 non-null  float64
 32  positive_rate                       23211 non-null  float64
 33  stringency_index                    47847 non-null  float64
 34  population                          57071 non-null  float64
 35  population_density                  54371 non-null  float64
 36  median_age                          51034 non-null  float64
 37  aged_65_older                       50265 non-null  float64
 38  aged_70_older                       50768 non-null  float64
 39  gdp_per_capita                      50367 non-null  float64
 40  extreme_poverty                     33571 non-null  float64
 41  cardiovasc_death_rate               51013 non-null  float64
 42  diabetes_prevalence                 52881 non-null  float64
 43  female_smokers                      39669 non-null  float64
 44  male_smokers                        39156 non-null  float64
 45  handwashing_facilities              24176 non-null  float64
 46  hospital_beds_per_thousand          45936 non-null  float64
 47  life_expectancy                     56336 non-null  float64
 48  human_development_index             49247 non-null  float64
dtypes: float64(45), object(4)
memory usage: 21.5+ MB
```

In [11]:
```
#describe of data in dataframe

df.describe()
```

Out[11]:

	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
count	5.375800e+04	56465.000000	55652.000000	4.436800e+04	56465.000000	55652.000000	53471.000000	56401.000000	55587.000000	44096.000000
mean	1.677974e+05	1953.576941	1920.431953	6.858639e+03	47.054317	46.835439	3139.099982	39.344804	38.403211	95.335293
std	1.693038e+06	18269.650340	17777.391785	5.578081e+04	390.853776	378.272794	6183.455260	133.985155	101.649441	180.721699
min	1.000000e+00	-8261.000000	-552.000000	1.000000e+00	-1918.000000	-232.143000	0.001000	-2212.545000	-269.978000	0.000000
25%	1.800000e+02	0.000000	0.857000	1.300000e+01	0.000000	0.000000	90.033000	0.000000	0.260000	3.977750
50%	2.070000e+03	14.000000	19.429000	8.400000e+01	0.000000	0.286000	593.663000	2.194000	3.863000	20.383000
75%	2.235675e+04	235.000000	245.286000	7.270000e+02	4.000000	4.000000	3487.050000	25.941000	29.678500	90.571250
max	5.515465e+07	646281.000000	584981.857000	1.328537e+06	10600.000000	9027.714000	76541.772000	8652.658000	2472.188000	1248.014000

8 rows × 45 columns

Low Level Data Understanding

In [12]:
```
#Find count of unique values in location column.

df['location'].nunique()
```

Out[12]:
```
216
```

In [13]:
```
#Find which continent has maximum frequency using values counts.

df['continent'].value_counts()
```

Out[13]:
```
Europe           14828
Africa           13637
Asia             13528
North America     9116
South America     3404
Oceania           2235
Name: continent, dtype: int64
```

In [14]:
```
#Find maximum & mean value in 'total_cases'.

print("Maximum value: ",df['total_cases'].max())
print("Mean value: ",df['total_cases'].mean())
```

```
Maximum value:  55154651.0
Mean value:  167797.3688753302
```

In [15]:
```
#Find 25%,50% & 75% quartile value in 'total_deaths'.

print("25% ",df['total_deaths'].describe()[4])
print("50% ",df['total_deaths'].describe()[5])
print("75% ",df['total_deaths'].describe()[6])
```

```
25%  13.0
50%  84.0
75%  727.0
```

In [16]:
```
#Find which continent has maximum 'human_development_index'.

df.groupby(['continent']).agg({'human_development_index':'max'})
```

Out[16]:

continent	human_development_index
Africa	0.797
Asia	0.933
Europe	0.953
North America	0.926
Oceania	0.939
South America	0.843

In [17]:
```
#Find which continent has minimum 'gdp_per_capita'.

df.groupby(['continent']).agg({'gdp_per_capita':'min'})
```

Out[17]:

continent	gdp_per_capita
Africa	661.240
Asia	1479.147
Europe	5189.972
North America	1653.173
Oceania	2205.923
South America	6885.829

In [18]:
```
# Filter the dataframe with only this columns ['continent','location','date','total_cases','total_deaths','gdp_per_capita','human_development_index']
# and update the data frame.

df=df[['continent','location','date','total_cases','total_deaths','gdp_per_capita','human_development_index']]

df
```

Out[18]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	31/12/19	NaN	NaN	1803.987	0.498
1	Asia	Afghanistan	01/01/20	NaN	NaN	1803.987	0.498
2	Asia	Afghanistan	02/01/20	NaN	NaN	1803.987	0.498
3	Asia	Afghanistan	03/01/20	NaN	NaN	1803.987	0.498
4	Asia	Afghanistan	04/01/20	NaN	NaN	1803.987	0.498
...	...	...	...	...	...	...	...
57389	NaN	International	13/11/20	696.0	7.0	NaN	NaN
57390	NaN	International	14/11/20	696.0	7.0	NaN	NaN
57391	NaN	International	15/11/20	696.0	7.0	NaN	NaN
57392	NaN	International	16/11/20	696.0	7.0	NaN	NaN
57393	NaN	International	17/11/20	696.0	7.0	NaN	NaN

57394 rows × 7 columns

Data Cleaning

In [19]:
```
#Remove all duplicates observations

df.drop_duplicates()
```

Out[19]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	31/12/19	NaN	NaN	1803.987	0.498
1	Asia	Afghanistan	01/01/20	NaN	NaN	1803.987	0.498
2	Asia	Afghanistan	02/01/20	NaN	NaN	1803.987	0.498
3	Asia	Afghanistan	03/01/20	NaN	NaN	1803.987	0.498
4	Asia	Afghanistan	04/01/20	NaN	NaN	1803.987	0.498
...	...	...	...	...	...	...	...
57389	NaN	International	13/11/20	696.0	7.0	NaN	NaN
57390	NaN	International	14/11/20	696.0	7.0	NaN	NaN
57391	NaN	International	15/11/20	696.0	7.0	NaN	NaN
57392	NaN	International	16/11/20	696.0	7.0	NaN	NaN
57393	NaN	International	17/11/20	696.0	7.0	NaN	NaN

57394 rows × 7 columns

In [20]:
```
#Find missing values in all columns

df.isnull()
```

Out[20]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	False	False	False	True	True	False	False
1	False	False	False	True	True	False	False
2	False	False	False	True	True	False	False
3	False	False	False	True	True	False	False
4	False	False	False	True	True	False	False
...	...	...	...	...	...	...	...
57389	True	False	False	False	False	True	True
57390	True	False	False	False	False	True	True
57391	True	False	False	False	False	True	True
57392	True	False	False	False	False	True	True
57393	True	False	False	False	False	True	True

57394 rows × 7 columns

In [21]:
```
# Remove all observations where continent column value is missing

df.dropna(subset=['continent'])
```

Out[21]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	31/12/19	NaN	NaN	1803.987	0.498
1	Asia	Afghanistan	01/01/20	NaN	NaN	1803.987	0.498
2	Asia	Afghanistan	02/01/20	NaN	NaN	1803.987	0.498
3	Asia	Afghanistan	03/01/20	NaN	NaN	1803.987	0.498
4	Asia	Afghanistan	04/01/20	NaN	NaN	1803.987	0.498
...	...	...	...	...	...	...	...
56743	Africa	Zimbabwe	13/11/20	8696.0	255.0	1899.775	0.535
56744	Africa	Zimbabwe	14/11/20	8765.0	257.0	1899.775	0.535
56745	Africa	Zimbabwe	15/11/20	8786.0	257.0	1899.775	0.535
56746	Africa	Zimbabwe	16/11/20	8786.0	257.0	1899.775	0.535
56747	Africa	Zimbabwe	17/11/20	8897.0	257.0	1899.775	0.535

56748 rows × 7 columns

In [22]:
```
#Fill all missing values with 0

df.fillna(0)
```

Out[22]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	31/12/19	0.0	0.0	1803.987	0.498
1	Asia	Afghanistan	01/01/20	0.0	0.0	1803.987	0.498
2	Asia	Afghanistan	02/01/20	0.0	0.0	1803.987	0.498
3	Asia	Afghanistan	03/01/20	0.0	0.0	1803.987	0.498
4	Asia	Afghanistan	04/01/20	0.0	0.0	1803.987	0.498
...	...	...	...	...	...	...	...
57389	0	International	13/11/20	696.0	7.0	0.000	0.000
57390	0	International	14/11/20	696.0	7.0	0.000	0.000
57391	0	International	15/11/20	696.0	7.0	0.000	0.000
57392	0	International	16/11/20	696.0	7.0	0.000	0.000
57393	0	International	17/11/20	696.0	7.0	0.000	0.000

57394 rows × 7 columns

In [ ]:
```
#Convert date column in datetime format using pandas.to_datetime

df['date'] = pd.to_datetime(df['date'])
```

In [26]:
```
df
```

Out[26]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	2019-12-31	NaN	NaN	1803.987	0.498
1	Asia	Afghanistan	2020-01-01	NaN	NaN	1803.987	0.498
2	Asia	Afghanistan	2020-02-01	NaN	NaN	1803.987	0.498
3	Asia	Afghanistan	2020-03-01	NaN	NaN	1803.987	0.498
4	Asia	Afghanistan	2020-04-01	NaN	NaN	1803.987	0.498
...	...	...	...	...	...	...	...
57389	NaN	International	2020-11-13	696.0	7.0	NaN	NaN
57390	NaN	International	2020-11-14	696.0	7.0	NaN	NaN
57391	NaN	International	2020-11-15	696.0	7.0	NaN	NaN
57392	NaN	International	2020-11-16	696.0	7.0	NaN	NaN
57393	NaN	International	2020-11-17	696.0	7.0	NaN	NaN

57394 rows × 7 columns

In [ ]:
```
#Create new column month after extracting month data from date column.

df['month']= df['date'].dt.month
```

In [29]:
```
df.head()
```

Out[29]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
0	Asia	Afghanistan	2019-12-31	NaN	NaN	1803.987	0.498	12
1	Asia	Afghanistan	2020-01-01	NaN	NaN	1803.987	0.498	1
2	Asia	Afghanistan	2020-02-01	NaN	NaN	1803.987	0.498	2
3	Asia	Afghanistan	2020-03-01	NaN	NaN	1803.987	0.498	3
4	Asia	Afghanistan	2020-04-01	NaN	NaN	1803.987	0.498	4

Data Aggregation:

In [30]:
```
#Find max value in all columns using groupby function on 'continent' column

df.groupby(["continent"]).max().reset_index()
```

Out[30]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
0	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12
1	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12
2	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12
3	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12
4	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12
5	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12

In [31]:
```
#Store the result in a new dataframe named 'df_groupby'.

df_groupby = df.groupby(["continent"]).max().reset_index()
```

In [32]:
```
df_groupby
```

Out[32]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
0	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12
1	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12
2	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12
3	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12
4	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12
5	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12

Feature Engineering

In [33]:
```
# Create a new feature 'total_deaths_to_total_cases' by ratio of 'total_deaths' column to 'total_cases'

df_groupby['total_deaths_to_total_cases'] = df['total_deaths']/df['total_cases']
```

In [35]:
```
df_groupby.head()
```

Out[35]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month	total_deaths_to_total_cases
0	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12	NaN
1	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12	NaN
2	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12	NaN
3	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12	NaN
4	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12	NaN

Data Visualization

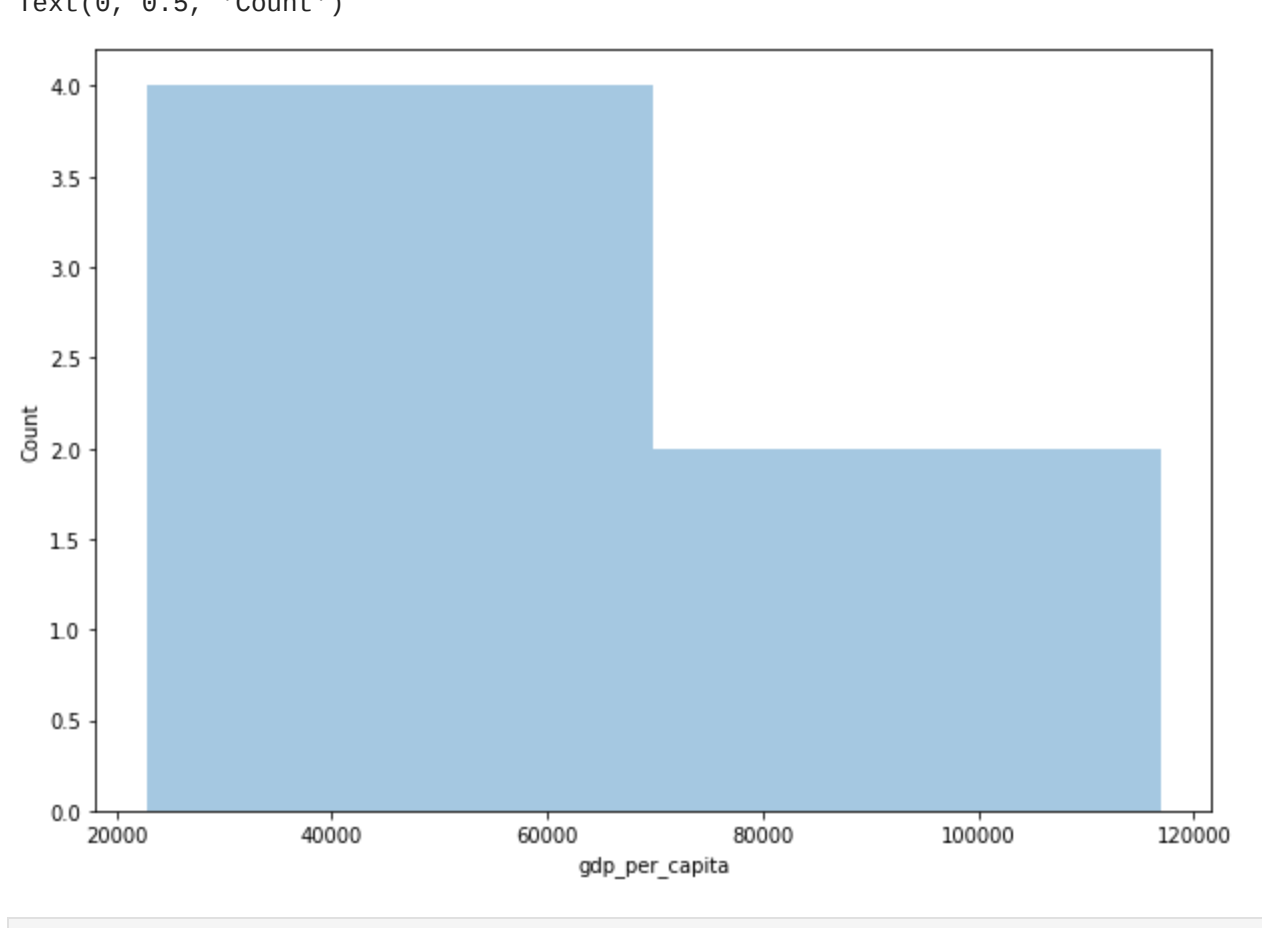
In [36]:
```
#Perform Univariate analysis on 'gdp_per_capita' column by plotting histogram using seaborn dist plot.

plt.figure(figsize=(10,7))

sns.distplot(df_groupby['gdp_per_capita'],kde = False , hist=True)
plt.xlabel("gdp_per_capita")
plt.ylabel("Count")
```

```
c:\users\yogesh\appdata\local\programs\python\python39\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

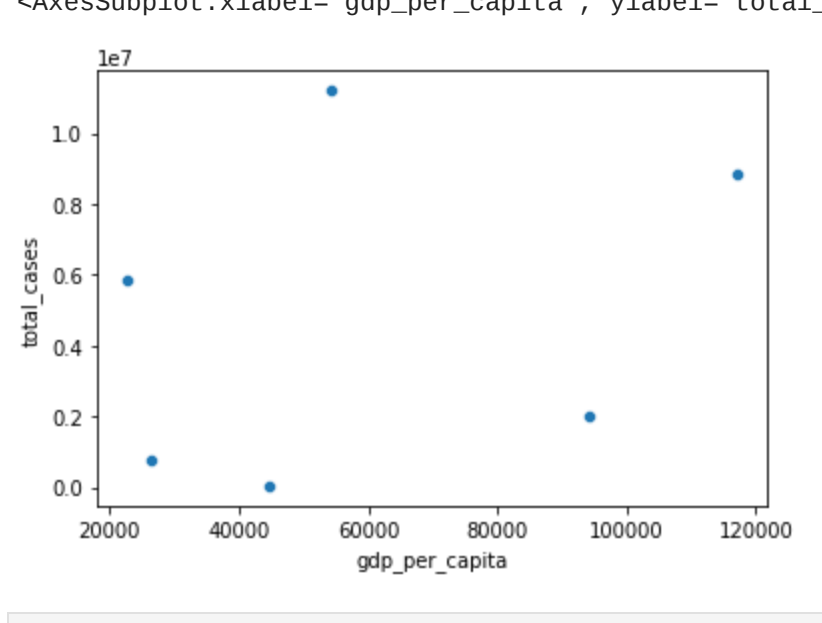
Out[36]:
```
Text(0, 0.5, 'Count')
```



In [37]:
```
# Plot a scatter plot of 'total_cases' & 'gdp_per_capita'

sns.scatterplot(data=df_groupby,x='gdp_per_capita',y='total_cases')
```

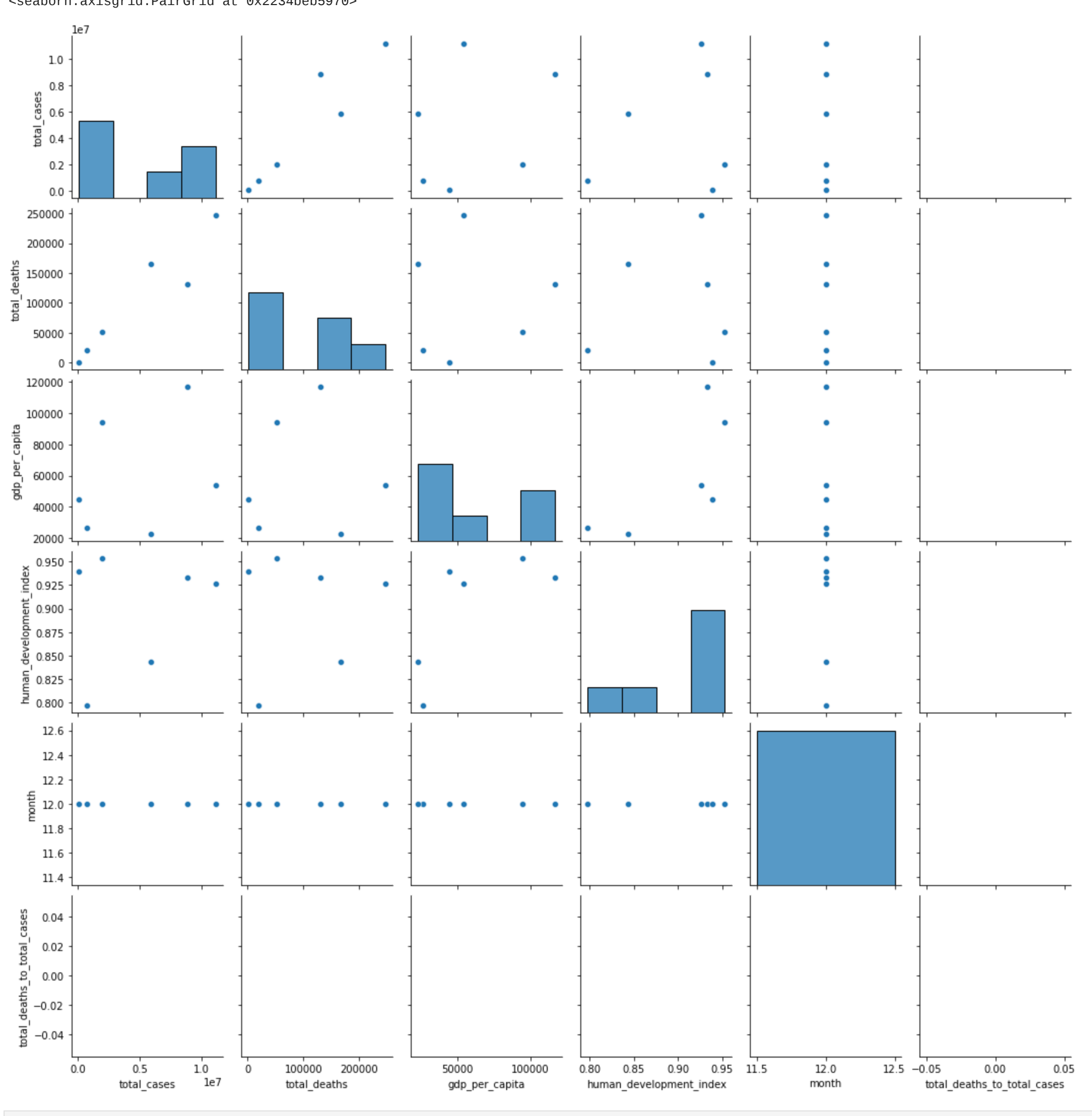
Out[37]:
```
<AxesSubplot:xlabel='gdp_per_capita', ylabel='total_cases'>
```



In [38]:
```
# Plot Pairplot on df_groupby dataset.

sns.pairplot(data=df_groupby)
```

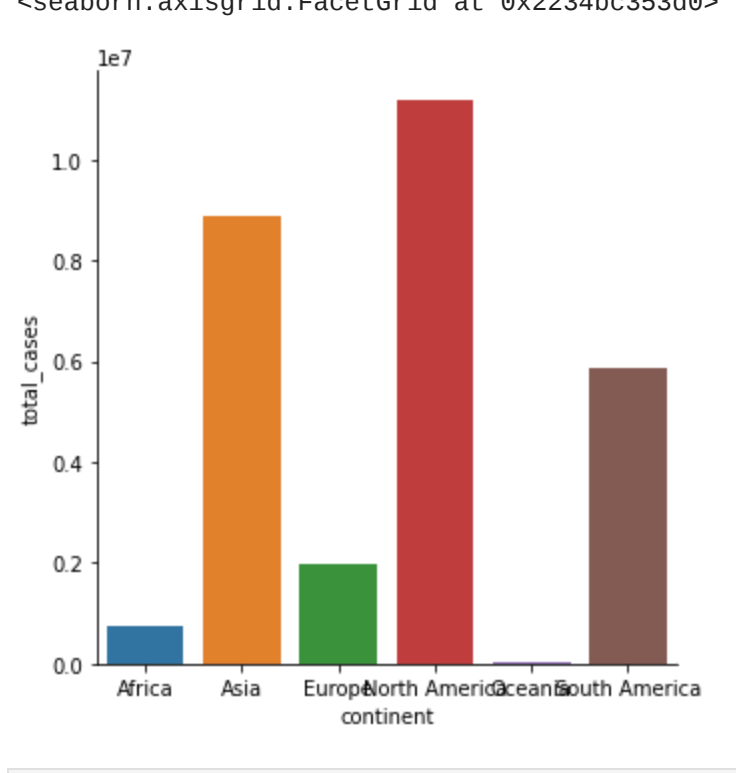
Out[38]:
```
<seaborn.axisgrid.PairGrid at 0x2234beb5970>
```



In [39]:
```
# Plot a bar plot of 'continent' column with 'total_cases' .

sns.catplot(data=df_groupby, x ="continent" , y ="total_cases" ,kind='bar')
```

Out[39]:
```
<seaborn.axisgrid.FacetGrid at 0x2234bc353d0>
```



In [40]:
```
# Save the df_groupby data frame in your local drive using pandas.to_csv function

df_groupby.to_csv('Covid - 19 Data Analysis.csv')
```

In [ ]: