Problem for Covid - 19 Data Analysis Project using Pandas

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
Url = 'https://raw.githubusercontent.com/SR1608/Datasets/main/covid-data.csv'
```

1. Import the dataset using Pandas from above mentioned url.

In [7]:

```
df = pd.read_csv(Url)
df
```

Out[7]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	to
0	AFG	Asia	Afghanistan	31/12/19	NaN	0.0	NaN	
1	AFG	Asia	Afghanistan	01/01/20	NaN	0.0	NaN	
2	AFG	Asia	Afghanistan	02/01/20	NaN	0.0	NaN	
3	AFG	Asia	Afghanistan	03/01/20	NaN	0.0	NaN	
4	AFG	Asia	Afghanistan	04/01/20	NaN	0.0	NaN	
...	...	...	...	...	...	...	...	
57389	NaN	NaN	International	13/11/20	696.0	NaN	NaN	
57390	NaN	NaN	International	14/11/20	696.0	NaN	NaN	
57391	NaN	NaN	International	15/11/20	696.0	NaN	NaN	
57392	NaN	NaN	International	16/11/20	696.0	NaN	NaN	
57393	NaN	NaN	International	17/11/20	696.0	NaN	NaN	

57394 rows × 49 columns

2. High Level Data Understanding:

In [8]:

```
# a. Find no. of rows & columns in the dataset

sh = df.shape
print(f"The No of Rows are {sh[0]}")
print(f"The No of Columns are {sh[1]}")
```

```
The No of Rows are 57394
The No of Columns are 49
```

In [6]:

```
# b. Data types of columns.

df.dtypes
```

Out[6]:

```
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 [9]:

```
# c. Info & describe 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 [10]:

```
df.describe()
```

Out[10]:

	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_
count	5.375800e+04	56465.000000	55652.000000	4.436800e+04	56465.000000	556
mean	1.677974e+05	1953.576941	1920.431953	6.858639e+03	47.054317	
std	1.693038e+06	18269.650340	17777.391785	5.578081e+04	390.853776	3
min	1.000000e+00	-8261.000000	-552.000000	1.000000e+00	-1918.000000	-2
25%	1.800000e+02	0.000000	0.857000	1.300000e+01	0.000000	
50%	2.070000e+03	14.000000	19.429000	8.400000e+01	0.000000	

	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_s
75%	2.235675e+04	235.000000	245.286000	7.270000e+02	4.000000	
max	5.515465e+07	646281.000000	584981.857000	1.328537e+06	10600.000000	90

8 rows × 45 columns

3. Low Level Data Understanding :

In [11]:

```
# a. Find count of unique values in location column.

df.location.unique()
```

Out[11]:

```
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Aruba',
       'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
       'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin',
       'Bermuda', 'Bhutan', 'Bolivia', 'Bonaire Sint Eustatius and Saba',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei', 'Bulgaria', 'Burkina Faso',
       'Burundi', 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde',
       'Cayman Islands', 'Central African Republic', 'Chad', 'Chile',
       'China', 'Colombia', 'Comoros', 'Congo', 'Costa Rica',
       "Cote d'Ivoire", 'Croatia', 'Cuba', 'Curacao', 'Cyprus',
       'Czech Republic', 'Democratic Republic of Congo', 'Denmark',
       'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt',
       'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Ethiopia', 'Faeroe Islands', 'Falkland Islands', 'Fiji',
       'Finland', 'France', 'French Polynesia', 'Gabon', 'Gambia',
       'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland',
       'Grenada', 'Guam', 'Guatemala', 'Guernsey', 'Guinea',
       'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong',
       'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq',
       'Ireland', 'Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan',
       'Jersey', 'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
       'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia',
       'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Macedonia',
       'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
       'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico', 'Moldova',
       'Monaco', 'Mongolia', 'Montenegro', 'Montserrat', 'Morocco',
       'Mozambique', 'Myanmar', 'Namibia', 'Nepal', 'Netherlands',
       'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria',
       'Northern Mariana Islands', 'Norway', 'Oman', 'Pakistan',
       'Palestine', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
       'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Qatar',
       'Romania', 'Russia', 'Rwanda', 'Saint Kitts and Nevis',
       'Saint Lucia', 'Saint Vincent and the Grenadines', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore',
       'Sint Maarten (Dutch part)', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Korea',
       'South Sudan', 'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
       'Swaziland', 'Sweden', 'Switzerland', 'Syria', 'Taiwan',
       'Tajikistan', 'Tanzania', 'Thailand', 'Timor', 'Togo',
       'Trinidad and Tobago', 'Tunisia', 'Turkey',
       'Turks and Caicos Islands', 'Uganda', 'Ukraine',
       'United Arab Emirates', 'United Kingdom', 'United States',
       'United States Virgin Islands', 'Uruguay', 'Uzbekistan', 'Vanuatu',
       'Vatican', 'Venezuela', 'Vietnam', 'Wallis and Futuna',
       'Western Sahara', 'Yemen', 'Zambia', 'Zimbabwe', 'World',
       'International'], dtype=object)
```

In [12]:

```
# b. Find which continent has maximum frequency using values counts.

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

Out[12]:

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

In [13]:

```
#  c. Find maximum & mean value in 'total_cases'.

df['total_cases'].mean()
```

Out[13]:

```
167797.3688753302
```

In [14]:

```
df['total_cases'].max()
```

Out[14]:

```
55154651.0
```

In [15]:

```
# d. Find 25%,50% & 75% quartile value in 'total_deaths'.

df.total_deaths.describe()
```

Out[15]:

```
count    4.436800e+04
mean     6.858639e+03
std      5.578081e+04
min      1.000000e+00
25%      1.300000e+01
50%      8.400000e+01
75%      7.270000e+02
max      1.328537e+06
Name: total_deaths, dtype: float64
```

In [16]:

```
# e. Find which continent has maximum 'human_development_index'.

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

Out[16]:

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

In [17]:

```
# f. Find which continent has minimum 'gdp_per_capita'.

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

Out[17]:

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

4. Filter the dataframe with only this columns:

['continent','location','date','total_cases','total_deaths','gdp_per_capi and update the data frame.

In [18]:

```
df_original = df.copy()
```

In [19]:

```
df = df.loc[:,['continent','location','date','total_cases','total_deaths','gdp_per_c
```

In [20]:

```
df
```

Out[20]:

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

57394 rows × 7 columns

5. Data Cleaning

In [21]:

```
#  a. Remove all duplicates observations
```

```
df.duplicated().sum()
```

Out[21]: 0

In [22]:

```
df.drop_duplicates()
```

Out[22]:

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

57394 rows × 7 columns

In [23]:

```
# b. Find missing values in all columns

df.isnull().sum()
```

Out[23]:

```
continent                  646
location                     0
date                         0
total_cases               3636
total_deaths             13026
gdp_per_capita            7027
human_development_index   8147
dtype: int64
```

In [25]:

```
# c. Remove all observations where continent column value is missing

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

In [26]:

```
df.isnull().sum()
```

Out[26]:

```
continent                    0
location                     0
date                         0
total_cases               3600
total_deaths             12964
gdp_per_capita            6704
human_development_index   7501
dtype: int64
```

In [27]:

```
#  d. Fill all missing values with 0

df = df.fillna(0)
```

In [28]:

```
df.isnull().sum()
```

Out[28]:

```
continent                 0
location                  0
date                      0
total_cases               0
total_deaths              0
gdp_per_capita            0
human_development_index   0
dtype: int64
```

In [29]:

```
df.dtypes
```

Out[29]:

```
continent                   object
location                    object
date                        object
total_cases                float64
total_deaths               float64
gdp_per_capita             float64
human_development_index    float64
dtype: object
```

date should be in datetime format not as object data type.

6. Date time format :

In [30]:

```
# a. Convert date column in datetime format using pandas.to_datetime

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

In [31]:

```
df.dtypes
```

Out[31]:

```
continent                          object
location                           object
date                       datetime64[ns]
total_cases                       float64
total_deaths                      float64
gdp_per_capita                    float64
human_development_index           float64
dtype: object
```

In [32]:

```
# b. Create new column month after extracting month data from date column.

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

In [33]:

```
df
```

Out[33]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development
0	Asia	Afghanistan	2019-12-31	0.0	0.0	1803.987	
1	Asia	Afghanistan	2020-01-01	0.0	0.0	1803.987	

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development
2	Asia	Afghanistan	2020-02-01	0.0	0.0	1803.987	
3	Asia	Afghanistan	2020-03-01	0.0	0.0	1803.987	
4	Asia	Afghanistan	2020-04-01	0.0	0.0	1803.987	
...	...	...	...	...	...	...	
56743	Africa	Zimbabwe	2020-11-13	8696.0	255.0	1899.775	
56744	Africa	Zimbabwe	2020-11-14	8765.0	257.0	1899.775	
56745	Africa	Zimbabwe	2020-11-15	8786.0	257.0	1899.775	
56746	Africa	Zimbabwe	2020-11-16	8786.0	257.0	1899.775	
56747	Africa	Zimbabwe	2020-11-17	8897.0	257.0	1899.775	

56748 rows × 8 columns

7. Data Aggregation:

In [34]:

```
# a. Find max value in all columns using groupby function on 'continent' column.

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

In [35]:

```
df_groupby
```

Out[35]:

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

8. Feature Engineering :

In [36]:

```
#  a. Create a new feature 'total_deaths_to_total_cases' by ratio of 'total_deaths'

df_groupby['total_deaths_to_total_cases'] = df_groupby['total_deaths']/df_groupby['t
```

In [37]:

```
df_groupby
```

Out[37]:

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

9. Data Visualization :

In [38]:

```
import seaborn as sns
```

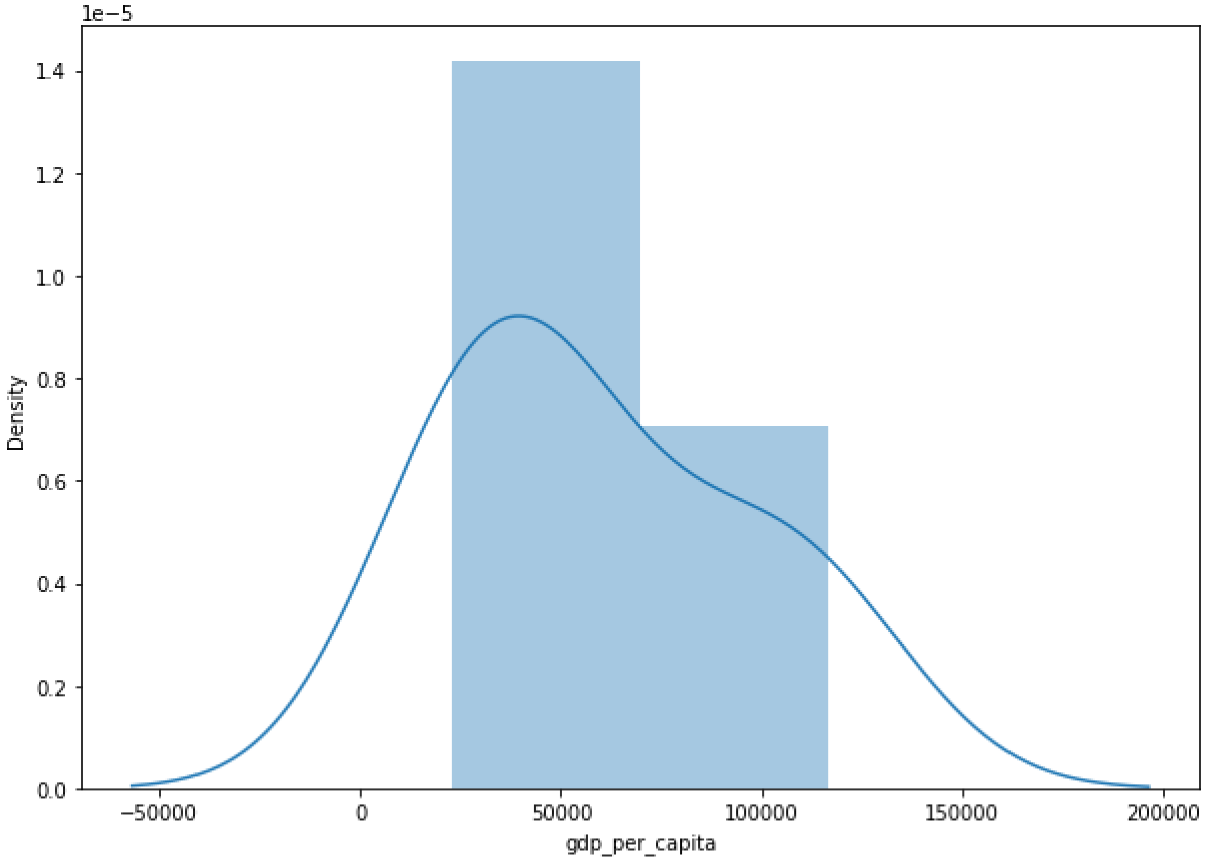
In [39]:

```
# a. Perform Univariate analysis on 'gdp_per_capita' column by plotting histogram us

plt.figure(figsize=(10,7))
sns.distplot(df_groupby['gdp_per_capita'])
```

Out[39]:

```
<AxesSubplot:xlabel='gdp_per_capita', ylabel='Density'>
```



We can see from this graph that density is too high where the gdp is bit less then 50000.

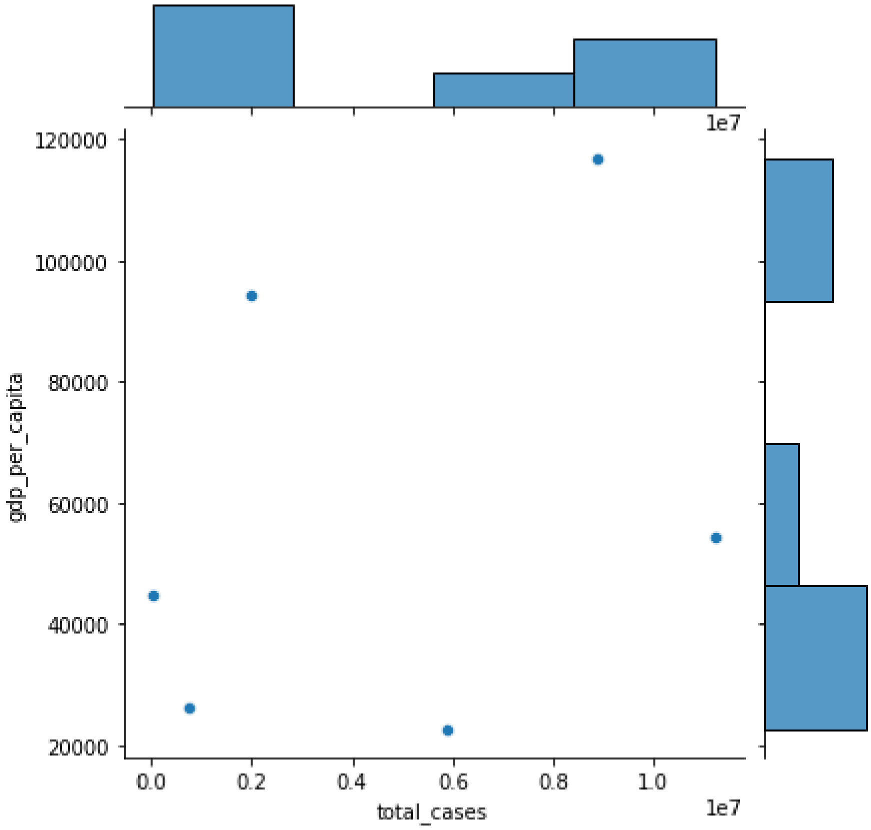
In [40]:

```
#  b. Plot a scatter plot of 'total_cases' & 'gdp_per_capita'

sns.jointplot(data=df_groupby , x= "total_cases" , y ="gdp_per_capita" ,kind='scatte
```

Out[40]:

```
<seaborn.axisgrid.JointGrid at 0x1e5e167a0a0>
```



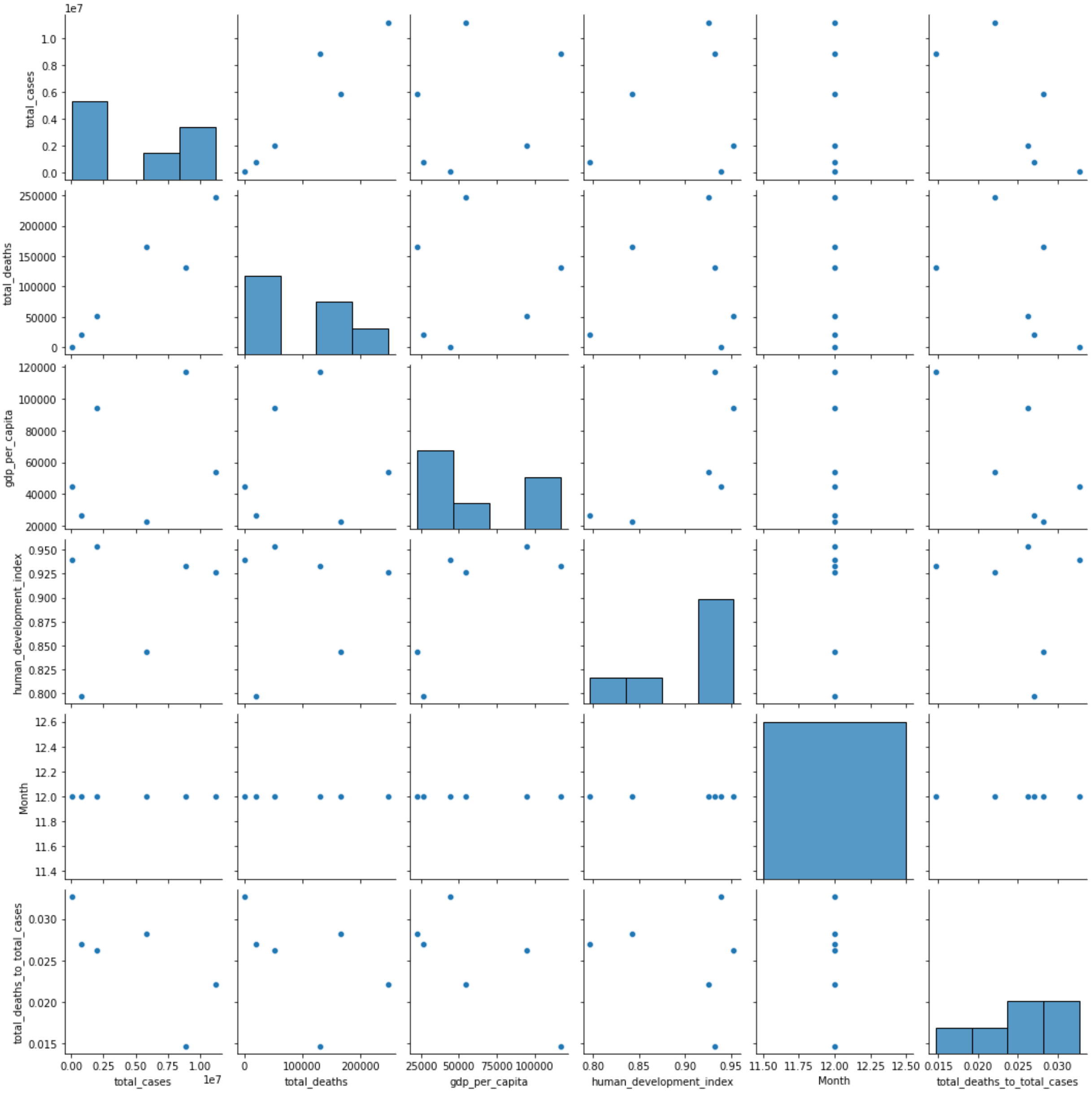
In [41]:

```
#  c. Plot Pairplot on df_groupby dataset.

sns.pairplot(data = df_groupby)
```

Out[41]:

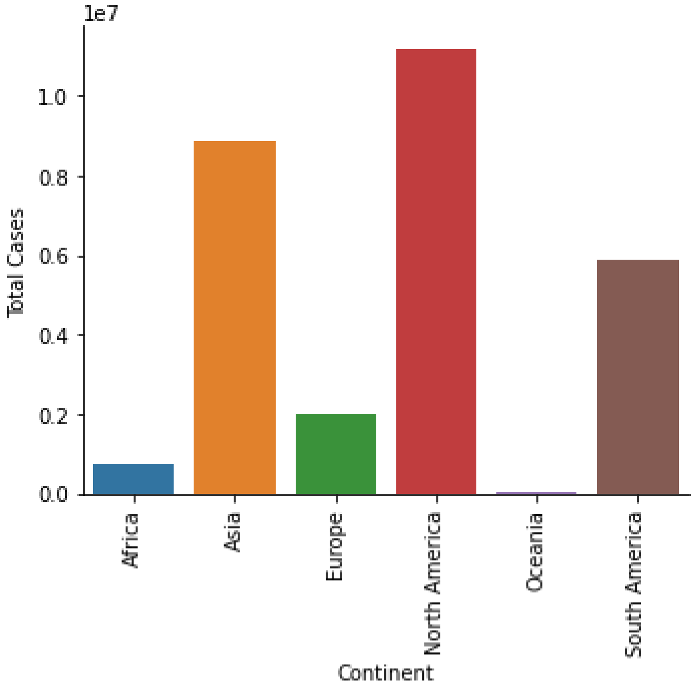
```
<seaborn.axisgrid.PairGrid at 0x1e5e17837c0>
```



In [42]:

```
# d. Plot a bar plot of 'continent' column with 'total_cases'


sns.catplot(data=df_groupby, x ="continent" , y ="total_cases" ,kind='bar')
plt.xlabel("Continent")
plt.ylabel("Total Cases")
plt.xticks(rotation=90)
plt.tight_layout()
```



We can conclude from this graph that number of cases is highest in North America and lowest in Ocenia.

10. Save the df_groupby data frame in your local drive using pandas.to_csv function

In [43]:

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