Problem for Covid - 19 Data Analysis Project using Python

Import the necessary libraries

In [1]:
```
#install lib
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
url = 'https://raw.githubusercontent.com/SR1608/Datasets/main/covid-data.csv'
```

1 . Import the dataset using Pandas

In [2]:
```
df = pd.read_csv(url)
df
```

Out[2]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	gdp_per
0	AFG	Asia	Afghanistan	31/12/19	NaN	0.0	NaN	NaN	0.0	NaN	...	18
1	AFG	Asia	Afghanistan	01/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	18
2	AFG	Asia	Afghanistan	02/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	18
3	AFG	Asia	Afghanistan	03/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	18
4	AFG	Asia	Afghanistan	04/01/20	NaN	0.0	NaN	NaN	0.0	NaN	...	18
...	...	...	...	...	...	...	...	...	...	...	...	
57389	NaN	NaN	International	13/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	
57390	NaN	NaN	International	14/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	
57391	NaN	NaN	International	15/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	
57392	NaN	NaN	International	16/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	
57393	NaN	NaN	International	17/11/20	696.0	NaN	NaN	7.0	NaN	NaN	...	

57394 rows × 49 columns

2. High level Data Understanding:

a. Find no. of rows and columns in the dataset

In [3]:
```
print("no. of rows :",len(df))
print("no. of columns :",len(df.columns))
```

```
no. of rows : 57394
no. of columns : 49
```

In [4]:
```
df.shape
```

Out[4]:
```
(57394, 49)
```

b. Data types of columns.

In [5]:
```
df.dtypes
```

Out[5]:
```
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
```

c. Info & describe of data in dataframe.

In [6]:
```
df.info()
```

```
 31  tests_per_case                 22862 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 [7]:
```
df.describe(include='all')
```

Out[7]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...
count	57071	56748	57394	57394	5.375800e+04	56465.000000	55652.000000	4.436800e+04	56465.000000	55652.000000	...
unique	215	6	216	323	NaN	NaN	NaN	NaN	NaN	NaN	...
top	AFG	Europe	Afghanistan	30/10/20	NaN	NaN	NaN	NaN	NaN	NaN	...
freq	323	14828	323	215	NaN	NaN	NaN	NaN	NaN	NaN	...
mean	NaN	NaN	NaN	NaN	1.677974e+05	1953.576941	1920.431953	6.858639e+03	47.054317	46.835439	...
std	NaN	NaN	NaN	NaN	1.693038e+06	18269.650340	17777.391785	5.578081e+04	390.853776	378.272794	...
min	NaN	NaN	NaN	NaN	1.000000e+00	-8261.000000	-552.000000	1.000000e+00	-1918.000000	-232.143000	...
25%	NaN	NaN	NaN	NaN	1.800000e+02	0.000000	0.857000	1.300000e+01	0.000000	0.000000	...
50%	NaN	NaN	NaN	NaN	2.070000e+03	14.000000	19.429000	8.400000e+01	0.000000	0.286000	...
75%	NaN	NaN	NaN	NaN	2.235675e+04	235.000000	245.286000	7.270000e+02	4.000000	4.000000	...
max	NaN	NaN	NaN	NaN	5.515465e+07	646281.000000	584981.857000	1.328537e+06	10600.000000	9027.714000	...

11 rows × 49 columns

3. Low Level Data Understanding:

a. Find count of unique values in location column.

In [8]:
```
df.location.nunique()
```

Out[8]: 216

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

In [9]:
```
b=df["continent"].dropna()
```

In [10]:
```
a=b.value_counts()
```

In [11]:
```
for i in b:
    if a[i]==a.max():
        print(f"{i} continent has maximum frequency",":",a.max())
        break
```

```
Europe continent has maximum frequency : 14828
```

c. Find maximum and mean value in total_case.

In [12]:
```
print("Maximum values :",df["total_cases"].max())
```

```
Maximum values : 55154651.0
```

In [13]:
```
print("Mean values :",df["total_cases"].mean())
```

```
Mean values : 167797.3688753302
```

d. Find 25%,50%,and 75% quartile values in total_deaths.

In [14]:
```
df["total_deaths"].quantile([0.25,0.5,0.75])
```

Out[14]:
```
0.25     13.0
0.50     84.0
0.75    727.0
Name: total_deaths, dtype: float64
```

e. Find which continent has maximum human_development_index.

In [15]:
```
a=df[["continent",'human_development_index']].dropna()
a[a['human_development_index'] == a['human_development_index'].max()].drop_duplicates()
```

Out[15]:

	continent	human_development_index
38632	Europe	0.953

f. Find which continent has minimum gdp_per_capita.

In [16]:
```
a=df[["continent",'gdp_per_capita']].dropna()
a[a['gdp_per_capita'] == a['gdp_per_capita'].min()].drop_duplicates()
```

Out[16]:

	continent	gdp_per_capita
10259	Africa	661.24

4.Filter the dataframe with only this columns

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

In [17]:
```
a= df.filter(['continent','location','date','total_cases','total_deaths','gdp_per_capita','human_development_index'])
```

In [18]:
```
df2=a
```

In [19]:
```
df2
```

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

5.Data Cleaning

a.Remove all duplicates observations.

In [20]:
```
df2.drop_duplicates()
```

Out[20]:

	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

b. Find missing values in all columns.

In [21]:
```
df2.isnull()
```

Out[21]:

	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

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

In [22]:
```
df4=df2.dropna(subset=['continent'])
df4
```

Out[22]:

	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

d. Fill all missing values with 0.

In [23]:
```
df3=df4.fillna(0)
df3
```

Out[23]:

	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
...	...	...	...	...	...	...	...
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

6. Data time format .

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

In [24]:
```
import datetime as dt
df3["date"]=pd.to_datetime(df3["date"])
df3["date"]=df3["date"]
```

In [25]:
```
df3
```

Out[25]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index
0	Asia	Afghanistan	2019-12-31	0.0	0.0	1803.987	0.498
1	Asia	Afghanistan	2020-01-01	0.0	0.0	1803.987	0.498
2	Asia	Afghanistan	2020-02-01	0.0	0.0	1803.987	0.498
3	Asia	Afghanistan	2020-03-01	0.0	0.0	1803.987	0.498
4	Asia	Afghanistan	2020-04-01	0.0	0.0	1803.987	0.498
...	...	...	...	...	...	...	...
56743	Africa	Zimbabwe	2020-11-13	8696.0	255.0	1899.775	0.535
56744	Africa	Zimbabwe	2020-11-14	8765.0	257.0	1899.775	0.535
56745	Africa	Zimbabwe	2020-11-15	8786.0	257.0	1899.775	0.535
56746	Africa	Zimbabwe	2020-11-16	8786.0	257.0	1899.775	0.535
56747	Africa	Zimbabwe	2020-11-17	8897.0	257.0	1899.775	0.535

56748 rows × 7 columns

In [26]:
```
dates = pd.to_datetime(df3['date'])
```

In [27]:
```
dates
```

Out[27]:
```
0       2019-12-31
1       2020-01-01
2       2020-02-01
3       2020-03-01
4       2020-04-01
           ...
56743   2020-11-13
56744   2020-11-14
56745   2020-11-15
56746   2020-11-16
56747   2020-11-17
Name: date, Length: 56748, dtype: datetime64[ns]
```

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

In [28]:
```
df3['month'] = pd.DatetimeIndex(df3['date']).month
```

In [29]:
```
df3
```

Out[29]:

	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
0	Asia	Afghanistan	2019-12-31	0.0	0.0	1803.987	0.498	12
1	Asia	Afghanistan	2020-01-01	0.0	0.0	1803.987	0.498	1
2	Asia	Afghanistan	2020-02-01	0.0	0.0	1803.987	0.498	2
3	Asia	Afghanistan	2020-03-01	0.0	0.0	1803.987	0.498	3
4	Asia	Afghanistan	2020-04-01	0.0	0.0	1803.987	0.498	4
...	...	...	...	...	...	...	...	...
56743	Africa	Zimbabwe	2020-11-13	8696.0	255.0	1899.775	0.535	11
56744	Africa	Zimbabwe	2020-11-14	8765.0	257.0	1899.775	0.535	11
56745	Africa	Zimbabwe	2020-11-15	8786.0	257.0	1899.775	0.535	11
56746	Africa	Zimbabwe	2020-11-16	8786.0	257.0	1899.775	0.535	11
56747	Africa	Zimbabwe	2020-11-17	8897.0	257.0	1899.775	0.535	11

56748 rows × 8 columns

7. Data Aggregation:

a. Find max value in all columns using groupby function on "continent" column.

In [30]:
```
df3.reset_index(inplace = True)
b=df3.groupby(df3['continent']).apply(max)
b
```

Out[30]:

continent	index	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
Africa	56747	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12
Asia	56261	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12
Europe	55230	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12
North America	54477	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12
Oceania	55833	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12
South America	55478	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12

b. Store the result in a new dataframe named "df_groupby'.

In [31]:
```
df_groupby = b
df_groupby
```

Out[31]:

continent	index	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month
Africa	56747	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12
Asia	56261	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12
Europe	55230	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12
North America	54477	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12
Oceania	55833	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12
South America	55478	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12

8. Feature Engineering :

a. Create a new feature "total_deaths_to_total_cases" by ratio of "total_deaths" column to "total_cases".

In [32]:
```
df_groupby["total_deaths_to_total_cases"] = df_groupby["total_deaths"]/df_groupby["total_cases"]
```

In [33]:
```
df_groupby
```

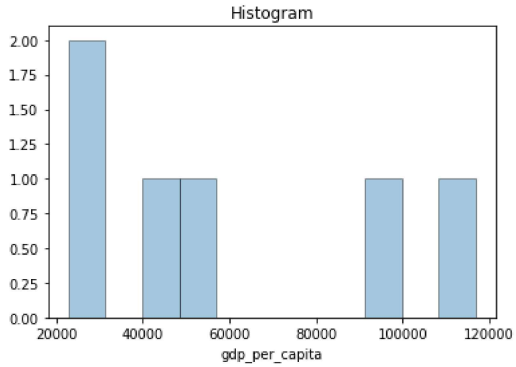
Out[33]:

continent	index	continent	location	date	total_cases	total_deaths	gdp_per_capita	human_development_index	month	total_deaths_to_total_cases
Africa	56747	Africa	Zimbabwe	2020-12-11	752269.0	20314.0	26382.287	0.797	12	0.027004
Asia	56261	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12	0.014708
Europe	55230	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.953	12	0.026188
North America	54477	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12	0.022062
Oceania	55833	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12	0.032685
South America	55478	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12	0.028251

9. Data Visualization :

a. Perform Univariate analysis on "gdp_per_capita" column by plotting histogram using seaborn dist plot.

In [34]:
```
sns.distplot(df_groupby['gdp_per_capita'],kde=False,hist=True,bins=11,hist_kws=dict(edgecolor="k", linewidth=1))
plt.title("Histogram")
plt.show()
```

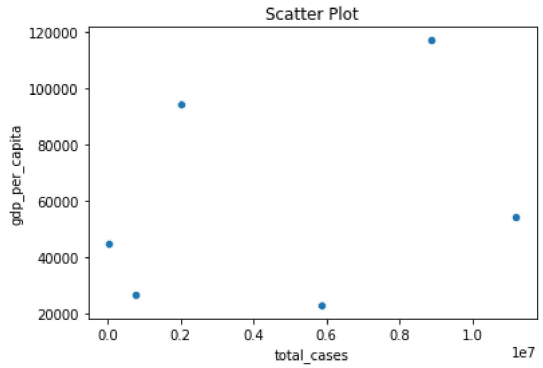


In [35]:
```
df_groupby.mean()
```

Out[35]:
```
index                          5.567100e+04
total_cases                    4.787915e+06
total_deaths                   1.028535e+05
gdp_per_capita                 5.987284e+04
human_development_index        8.985000e-01
month                          1.200000e+01
total_deaths_to_total_cases    2.514954e-02
dtype: float64
```

b. Plot a scatter plot of "total_cases" & "gdp_per_capita".

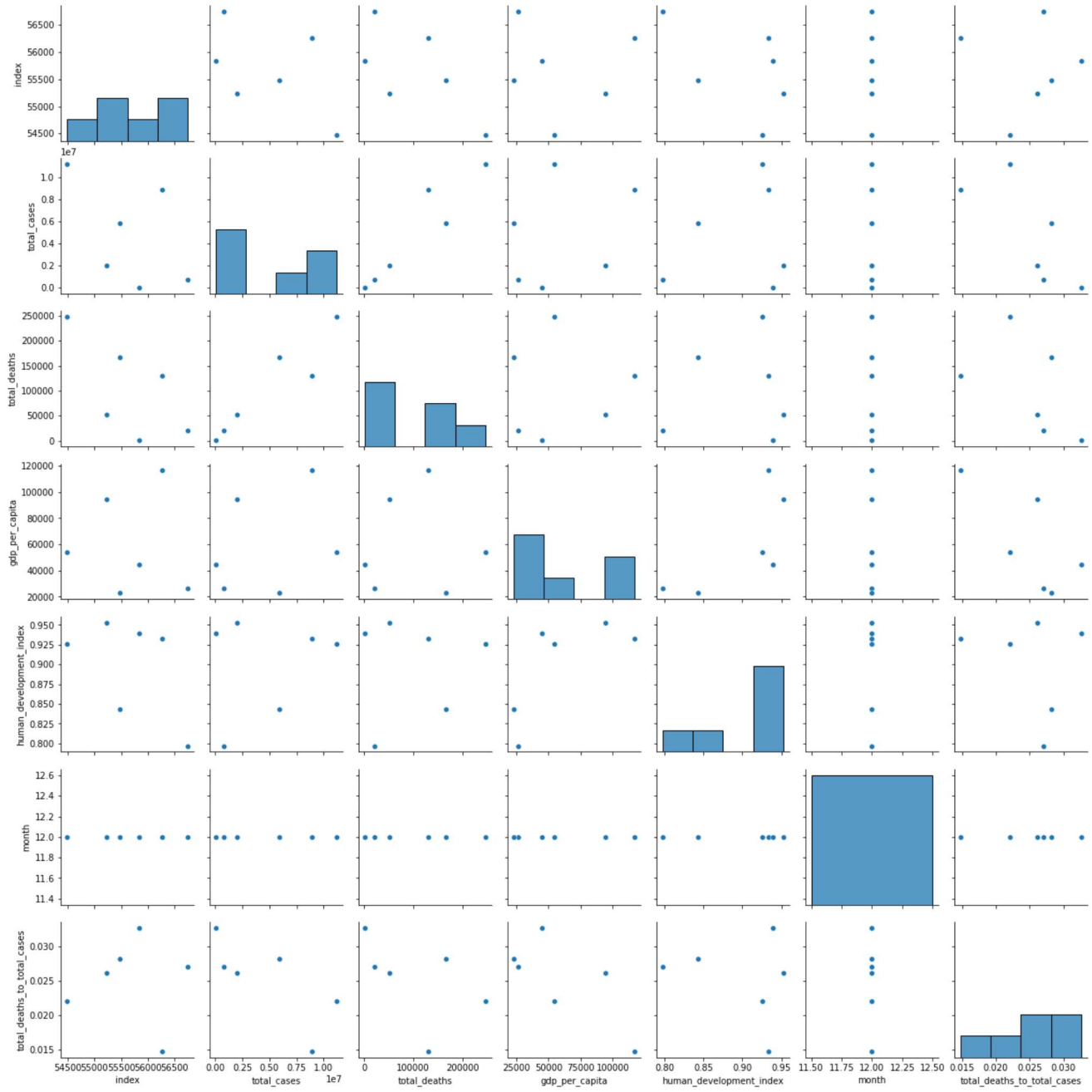
In [36]:
```
sns.scatterplot(x = 'total_cases', y= "gdp_per_capita", data=df_groupby)
plt.title("Scatter Plot")
plt.xlabel('total_cases')
plt.ylabel('gdp_per_capita')
plt.show()
```



c. Plot Pairplot on df_groupby dataset.

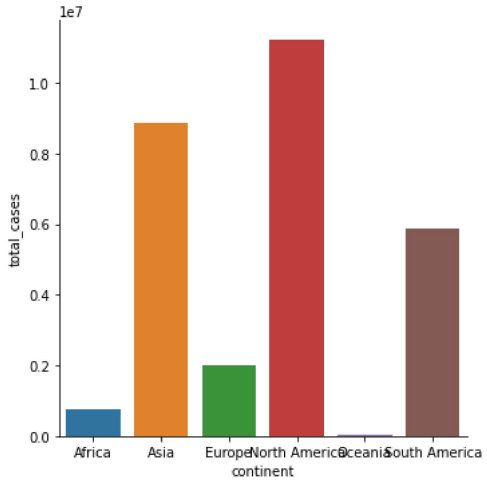
In [37]:
```
sns.pairplot(df_groupby)
```

Out[37]: <seaborn.axisgrid.PairGrid at 0x28659da75e0>



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

In [38]:
```
g = sns.catplot(x= "continent",y ="total_cases",kind="bar", data=df_groupby)
```



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

In [39]:
```
df3.to_csv('df_groupby.csv', index=False)
```