

Title: Problem for Covid - 19 Data Analysis Project using Python

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

```
In [113]: import pandas as pd
data=pd.read_csv("https://raw.githubusercontent.com/SR1968/Datasets/main/covid-data.csv",sep=",")
data.head()
```

```
Out[113]:
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	gdp_per_capita	extreme_poverty
0	AFG	Asia	Afghanistan	31/12/19	NaN	0.0	NaN	NaN	NaN	0.0	NaN	...	1803.987
1	AFG	Asia	Afghanistan	01/01/20	NaN	0.0	NaN	NaN	NaN	0.0	NaN	...	1803.987
2	AFG	Asia	Afghanistan	02/01/20	NaN	0.0	NaN	NaN	NaN	0.0	NaN	...	1803.987
3	AFG	Asia	Afghanistan	03/01/20	NaN	0.0	NaN	NaN	NaN	0.0	NaN	...	1803.987
4	AFG	Asia	Afghanistan	04/01/20	NaN	0.0	NaN	NaN	NaN	0.0	NaN	...	1803.987

5 rows × 14 columns

1. High Level Data Understanding:

```
In [114]: # a. Find no. of rows & columns in the dataset
data.shape
```

```
Out[114]: (57394, 14)
```

```
In [115]: # b. Data types of columns.
data.dtypes
```

```
Out[115]:
```

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_and_over	float64
aged_70_and_over	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 [116]: # c. Info & describe of data in dataframe.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57394 entries, 0 to 57393
Data columns (total 14 columns):
#   Column              Non-Null Count  Dtype
--  --
0   iso_code             57871 non-null  object
1   continent            56748 non-null  object
2   location             56739 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   56562 non-null  float64
7   total_deaths         44368 non-null  float64
8   new_deaths           56465 non-null  float64
9   new_deaths_smoothed  56552 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  44996 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          4498 non-null   float64
18  icu_patients_per_million  4498 non-null   float64
19  hosp_patients         5085 non-null   float64
20  hosp_patients_per_million  5085 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           22817 non-null  float64
26  new_tests             21787 non-null  float64
27  total_tests_per_thousand  22817 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         22882 non-null  float64
32  positive_rate          23211 non-null  float64
33  stringency_index       47847 non-null  float64
34  population             57871 non-null  float64
35  population_density      54371 non-null  float64
36  median_age             51034 non-null  float64
37  aged_65_and_over       50265 non-null  float64
38  aged_70_and_over       50788 non-null  float64
39  gdp_per_capita          56367 non-null  float64
40  extreme_poverty         33571 non-null  float64
41  cardiovasc_death_rate   51913 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  24376 non-null  float64
46  hospital_beds_per_thousand  54936 non-null  float64
47  life_expectancy         56336 non-null  float64
48  human_development_index  49247 non-null  float64
dtypes: float64(14), object(4)
memory usage: 21.5+ MB
```

1. Low Level Data Understanding :

```
In [117]: # a. Find count of unique values in location column.
data.location.unique()
```

```
Out[117]: 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', 'Farland 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',
       'Saint Martin (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 [118]: # b. Find which continent has maximum frequency using values counts.
data.continent.value_counts().head(1)
```

```
Out[118]: Europe    14828
Name: continent, dtype: int64
```

```
In [119]: # c. Find maximum & mean value in 'total_cases'.
maximum=data["total_cases"].max()
print("maximum ",maximum)
m=data["total_cases"].mean()
print("mean ",m)
```

```
maximum    55154651.0
m         167797.3688753392
```

```
In [120]: # d. Find 25%,50% & 75% quartile value in 'total_deaths'.
data["total_deaths"].head(100)
data["total_deaths"].describe()
```

```
Out[120]: count      4.436880e+04
mean       6.858609e+03
std        5.578808e+04
min         1.000000e+00
25%        1.309000e+01
50%        8.409000e+01
75%        7.270000e+02
max         1.328537e+06
Name: total_deaths, dtype: float64
```

```
In [121]: # e. Find which continent has maximum 'human_development_index'.
data1=data.groupby(by="continent").mean()
data1.sort_values(by="human_development_index",ascending=False).head(1)
```

```
Out[121]:
```

	continent	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
Europe	Europe	49483.089996	967.35581	926.678845	3771.910925	22.257118	21.717531	4884.905461	70.410528	68.910528

1 rows × 11 columns

```
In [122]: # f. Find which continent has minimum 'gdp_per_capita'.
data1.sort_values(by="gdp_per_capita").head(1)
```

```
Out[122]:
```

	continent	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
Africa	Africa	14121.679667	147.185947	147.204671	389.456834	3.529233	3.537624	825.859243	8.675412	8.675412

1 rows × 11 columns

1. 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.

```
In [123]: data=data[['continent','location','date','total_cases','total_deaths','gdp_per_capita','human_development_index']]
data
```

```
Out[123]:
```

	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 × 8 columns

1. Data Cleaning

```
In [124]: # a. Remove all duplicates observations
data.loc[data.duplicated(),:]
data.drop_duplicates()
```

```
Out[124]:
```

	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 × 8 columns

```
In [125]: #b. Find missing values in all columns
data.isnull().sum()
#data.shape
```

```
Out[125]:
```

continent	646
location	0
date	0
total_cases	3636
total_deaths	13026
gdp_per_capita	7827
human_development_index	8147
dtype:	int64

```
In [103]: #c. Remove all observations where continent value is missing
data.dropna()
data.dropna(subset=["continent"],inplace=True)
#Tip : using subset parameter in dropna
data.isnull().sum()
#data.shape
```

```
Out[103]:
```

continent	0
location	0
date	0
total_cases	3609
total_deaths	12964
gdp_per_capita	6784
human_development_index	7581
dtype:	int64

```
In [104]: #d. Fill all missing values with 0
data.fillna(0,inplace=True)
data.isnull().sum()
```

```
Out[104]:
```

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

1. Date time format :

```
In [105]: # a. Convert date column in datetime format using pandas to.datetime
data["date"]= pd.to_datetime(data["date"])
data
```

```
Out[105]:
```

	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 × 8 columns

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

```
In [106]: # b. Create new column month after extracting month data from date column.
data["month"]=pd.DatetimeIndex(data["date"]).month
data
```

```
Out[106]:
```

	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 × 9 columns

1. Data Aggregation: a. Find max value in all columns using groupby function on 'continent' column b. Store the result in a new dataframe named 'df_groupby'. (Use df_groupby dataframe for all further analysis)

```
In [107]: df_groupby=data.groupby(by="continent").max()
df_groupby.reset_index(inplace = True)
df_groupby
```

```
Out[107]:
```

	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	0.965	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

1. Feature Engineering : a. Create a new feature 'total_deaths_to_total_cases' by ratio of 'total_deaths' column to 'total_cases'

```
In [108]: df_groupby["total_deaths_to_total_cases"]=df_groupby['total_deaths']/df_groupby['total_cases']
df_groupby
```

```
Out[108]:
```

	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	0.027004
1	Asia	Yemen	2020-12-11	8874290.0	130519.0	116935.600	0.933	12	0.014708
2	Europe	Vatican	2020-12-11	1991233.0	52147.0	94277.965	0.965	12	0.026188
3	North America	United States Virgin Islands	2020-12-11	11205486.0	247220.0	54225.446	0.926	12	0.022062
4	Oceania	Wallis and Futuna	2020-12-11	27750.0	907.0	44648.710	0.939	12	0.032885
5	South America	Venezuela	2020-12-11	5876464.0	166014.0	22767.037	0.843	12	0.028251

1. Data Visualization :

```
In [109]: # a. Perform Univariate analysis on 'gdp_per_capita' column by plotting histogram using seaborn dist plot.
```



```
In [110]: # b. sns.scatter plot of 'total_cases' & 'gdp_per_capita'
sns.scatterplot(x='total_cases', y='gdp_per_capita', data=df_groupby)
```

