Data Science
Assignment – 1 Data Wrangling

1. Import all the required Python Libraries,
2. Locate an open source data from the web. Provide a clear description of the data and ist source.
3. Load the Dataset into pandas dataframe

```
In [1]: import pandas

In [2]: data_frame = pandas.read_csv('/home/mohit/DSBDA Assignments/DataSet/covid_worldwide.csv')
        data_frame
```

Out[2]:

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
1	2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
2	3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
3	4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
4	5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593
...	...	...	...	...	...	...	...	...
226	227	Diamond Princess	712	13	699	0	NaN	NaN
227	228	Vatican City	29	NaN	29	0	NaN	799
228	229	Western Sahara	10	1	9	0	NaN	626,161
229	230	MS Zaandam	9	2	7	0	NaN	NaN
230	231	Tokelau	5	NaN	NaN	5	NaN	1,378

231 rows × 8 columns

> Source :- Kaggle.com , covid_worldwide.csv

4. Data Preprocessing : Check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable description. Check the dimension of the data frame.

- Checking missing values (null values) and calculating the occurences of null values.
- Provide variable descriptions. Types of variables etc. Check the dimension of each frame.

```
data_frame.shape
```

```
(200, 8)
```

-> Number of Rows :- 200, Number of Columns :- 8

Count of occurences of null values in each column

```
data_frame.isnull().sum()
```

```
Serial Number       0
Country             0
Total Cases         0
Total Deaths        6
Total Recovered    21
Active Cases       19
Total Test         18
Population          3
dtype: int64
```

```
data_frame.tail(10)
```

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
221	222	Tuvalu	2,805	NaN	NaN	2,805	NaN	12,066
222	223	Saint Helena	2,166	NaN	2	2,164	NaN	6,115
223	224	Falkland Islands	1,930	NaN	1,930	0	8,632	3,539
224	225	Montserrat	1,403	8	1,376	19	17,762	4,965
225	226	Niue	747	NaN	746	1	NaN	1,622
226	227	Diamond Princess	712	13	699	0	NaN	NaN
227	228	Vatican City	29	NaN	29	0	NaN	799
228	229	Western Sahara	10	1	9	0	NaN	626,161
229	230	MS Zaandam	9	2	7	0	NaN	NaN

Processing of NaN Values by replacing it by zero.

```
data_frame.fillna(0)
```

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
0	1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
1	2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
2	3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
3	4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
4	5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593
...	...	...	...	...	...	...	...	...
226	227	Diamond Princess	712	13	699	0	0	0
227	228	Vatican City	29	0	29	0	0	799
228	229	Western Sahara	10	1	9	0	0	626,161
229	230	MS Zaandam	9	2	7	0	0	0
230	231	Tokelau	5	0	0	5	0	1,378

231 rows × 8 columns

Processing of NaN values by substituting it with mean.

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN

➢

NaN Column : -

18	45	'2020/12/18'	90	112	NaN

```
x = data_frame['Calories'].mean()
data_frame['Calories'].fillna(x, inplace=True)
data_frame
```

Modified Column : -

18	45	'2020/12/18'	90	112	304.68

5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types . If the variables in the data set are not in the correct data type, apply proper type conversion.

   Converting the data type of Total Cases from object into int64 to perform computations on it.

```
data_frame.dtypes
```

```
Serial Number       int64
Country            object
Total Cases        object
Total Deaths       object
Total Recovered    object
Active Cases       object
Total Test         object
Population         object
dtype: object
```

For Computations the column must be converted into int64.

```
data_frame['Total Cases'] = pandas.to_numeric(data_frame['Total Cases'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~/.local/lib/python3.10/site-packages/pandas/_libs/lib.pyx:2369, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "104,196,861"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Cell In[21], line 1
----> 1 data_frame['Total Cases'] = pandas.to_numeric(data_frame['Total Cases'])

File ~/.local/lib/python3.10/site-packages/pandas/core/tools/numeric.py:185, in to_numeric(arg, errors, downcast)
    183 coerce_numeric = errors not in ("ignore", "raise")
    184 try:
--> 185     values, _ = lib.maybe_convert_numeric(
    186         values, set(), coerce_numeric=coerce_numeric
    187     )
    188 except (ValueError, TypeError):
    189     if errors == "raise":

File ~/.local/lib/python3.10/site-packages/pandas/_libs/lib.pyx:2411, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "104,196,861" at position 0
```

Correction:

```
df['Total Cases'] = [column.replace(',', '') for column in df['Total Cases']]
```

```
df['Total Cases'] = pandas.to_numeric(df['Total Cases'])
df.dtypes
```

Outcome:

```
df['Total Cases'] = pandas.to_numeric(df['Total Cases'])
df.dtypes
```

```
Serial Number       int64
Country            object
Total Cases         int64
Total Deaths       object
Total Recovered    object
Active Cases       object
Total Test         object
Population         object
dtype: object
```

Filtering data and removing values (‘ Total Cases’) less than 10, 000, after applying conversion.

```
for index in data_frame.index:
    if data_frame.loc[index, 'Total Cases'] < 10000:
        data_frame.drop(index, inplace = True)

data_frame.tail(10)
```

	Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
190	191	Marshall Islands	15584	17	15,528	39	NaN	60,057
191	192	CAR	15368	113	15,200	55	81,294	5,016,678
192	193	Gambia	12586	372	12,189	25	155,686	2,558,482
193	194	Saint Martin	12233	63	1,399	10,771	112,382	39,730
194	195	Vanuatu	12014	14	11,976	24	24,976	321,832
195	196	Greenland	11971	21	2,761	9,189	164,926	56,973
196	197	Yemen	11945	2,159	9,124	662	329,592	31,154,867
197	198	Caribbean Netherlands	11661	38	10,476	1,147	30,126	26,647
198	199	Sint Maarten	11010	89	10,905	16	62,056	43,966
199	200	Eritrea	10189	103	10,086	0	23,693	3,662,244

Function to apply conversion to given column:
Example :- remove_commas

```
def remove_commas(val):
    s = str(val).replace(',', '')
    return int(s)

data_frame['Total Deaths'].apply(remove_commas)
```

```
0      1132935
1       530740
2       164233
3       165711
4       697074
        ...
195         21
196       2159
197         38
198         89
199        103
Name: Total Deaths, Length: 200, dtype: int64
```

>> ndim() Method :-

```
data_frame.ndim
```

```
2
```

>> Demonstration of describe() method.

```
data_frame.describe()
```

	Serial Number
count	231.000000
mean	116.000000
std	66.828138
min	1.000000
25%	58.500000
50%	116.000000
75%	173.500000
max	231.000000

>> reset_index() method

```
data_frame.columns
```

```
Index(['Serial Number', 'Country', 'Total Cases', 'Total Deaths',
       'Total Recovered', 'Active Cases', 'Total Test', 'Population'],
      dtype='object')
```

```
data_frame.set_index('Serial Number')
```

Serial Number	Country	Total Cases	Total Deaths	Total Recovered	Active Cases	Total Test	Population
1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593
...	...	...	...	...	...	...	...
227	Diamond Princess	712	13	699	0	NaN	NaN
228	Vatican City	29	NaN	29	0	NaN	799
229	Western Sahara	10	1	9	0	NaN	626,161
230	MS Zaandam	9	2	7	0	NaN	NaN
231	Tokelau	5	NaN	NaN	5	NaN	1,378

231 rows × 7 columns

6. Turn categorical variables into quantitative variables in Python.

Categorical data is a collection of information that is divided into groups. Categorical variables represent types of data which may be divided into groups. Examples of categorical variables are race, sex, age group, and educational level. While the latter two variables may also be considered in a numerical manner by

using exact values for age and highest grade completed, it is often more informative to categorize such variables into a relatively small number of groups.

Smartphones data frame :

```
smartphones_df = pandas.read_csv('DataSets/flipkart_smartphones.csv')
smartphones_df[['brand', 'model', 'colour', 'original_price', 'ratings', 'reviews', 'battery_type']]
```

	brand	model	colour	original_price	ratings	reviews	battery_type
0	VIVO	VIVO T1 44W	Starry Sky	19990	4.5	6044	Lithium
1	APPLE	APPLE IPHONE 11	White	48900	4.6	10818	NaN
2	VIVO	VIVO T1 44W	Midnight Galaxy	20990	4.4	3750	Lithium
3	XIAOMI	POCO M4 5G	Power Black	15999	4.2	4185	Lithium Polymer
4	XIAOMI	REDMI 10	Caribbean Green	14999	4.3	12084	Lithium Polymer
5	XIAOMI	POCO M4 5G	Cool Blue	15999	4.2	4185	Lithium Polymer
6	XIAOMI	POCO C31	Shadow Gray	11999	4.3	11672	Lithium
7	XIAOMI	POCO M4 5G	Yellow	15999	4.2	4185	Lithium Polymer
8	XIAOMI	REDMI 10	Midnight Black	14999	4.3	12084	Lithium Polymer
9	VIVO	VIVO T1 44W	Midnight Galaxy	23990	4.3	483	Lithium
10	VIVO	VIVO T1 44W	Starry Sky	20990	4.4	3750	Lithium
11	INFINIX	INFINIX HOT 20 PLAY	Racing Black	11999	4.4	486	Lithium

Dummy Variable: A **dummy** variable in pandas is an indicator variable that takes only the value, `0`, or,`1`, to indicate whether a separate categorical variable can take a specific value or not.

To create a dummy variable in a given DataFrame in pandas, we make use of the `get_dummies()` function

```
pandas.get_dummies(smartphones_df['battery_type']).head(10)
```

	Lithium	Lithium Ion	Lithium Polymer
0	1	0	0
1	0	0	0
2	1	0	0
3	0	0	1
4	0	0	1
5	0	0	1
6	1	0	0
7	0	0	1
8	0	0	1
9	1	0	0