2440016804 - Rio Pramana - LA01 - Assignment 1

Import libraries & read downloaded dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Downloaded file is in the same folder
csv_path = "covid_19_indonesia_time_series_all.csv"
covid19_df = pd.read_csv(csv_path)
```

Check dataset

In [3]:
```
covid19_df.shape
```

Out[3]: (21759, 38)

In [4]:
```
covid19_df.head(5)
```

Out[4]:

	Date	Location ISO Code	Location	New Cases	New Deaths	New Recovered	New Active Cases	Total Cases	Total Deaths	Total Recovered	...	Latitude	New Cases per Million	Total Cases per Million	New Deaths per Million	Total Deaths per Million	Total Deaths per 100rb	F
0	3/1/2020	ID-JK	DKI Jakarta	2	0	0	2	39	20	41	...	-6.204699	0.18	3.60	0.0	1.84	0.18	
1	3/2/2020	ID-JK	DKI Jakarta	2	0	0	2	41	20	41	...	-6.204699	0.18	3.78	0.0	1.84	0.18	
2	3/2/2020	IDN	Indonesia	2	0	0	2	2	0	0	...	-0.789275	0.01	0.01	0.0	0.00	0.00	
3	3/2/2020	ID-RI	Riau	1	0	0	1	2	0	3	...	0.511648	0.16	0.33	0.0	0.00	0.00	
4	3/3/2020	ID-JK	DKI Jakarta	2	0	0	2	43	20	41	...	-6.204699	0.18	3.96	0.0	1.84	0.18	

5 rows × 38 columns

In [5]:
```
covid19_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21759 entries, 0 to 21758
Data columns (total 38 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Date                        21759 non-null  object 
 1   Location ISO Code           21759 non-null  object 
 2   Location                    21759 non-null  object 
 3   New Cases                   21759 non-null  int64  
 4   New Deaths                  21759 non-null  int64  
 5   New Recovered               21759 non-null  int64  
 6   New Active Cases            21759 non-null  int64  
 7   Total Cases                 21759 non-null  int64  
 8   Total Deaths                21759 non-null  int64  
 9   Total Recovered             21759 non-null  int64  
 10  Total Active Cases          21759 non-null  int64  
 11  Location Level              21759 non-null  object 
 12  City or Regency             0 non-null      float64
 13  Province                    21117 non-null  object 
 14  Country                     21759 non-null  object 
 15  Continent                   21759 non-null  object 
 16  Island                      21117 non-null  object 
 17  Time Zone                   21117 non-null  object 
 18  Special Status              3123 non-null   object 
 19  Total Regencies             21759 non-null  int64  
 20  Total Cities                21145 non-null  float64
 21  Total Districts             21759 non-null  int64  
 22  Total Urban Villages        21142 non-null  float64
 23  Total Rural Villages        21117 non-null  float64
 24  Area (km2)                  21759 non-null  int64  
 25  Population                  21759 non-null  int64  
 26  Population Density          21759 non-null  float64
 27  Longitude                   21759 non-null  float64
 28  Latitude                    21759 non-null  float64
 29  New Cases per Million       21759 non-null  float64
 30  Total Cases per Million     21759 non-null  float64
```

```
 31  New Deaths per Million       21759 non-null  float64
 32  Total Deaths per Million     21759 non-null  float64
 33  Total Deaths per 100rb       21759 non-null  float64
 34  Case Fatality Rate           21759 non-null  object
 35  Case Recovered Rate          21759 non-null  object
 36  Growth Factor of New Cases   20572 non-null  float64
 37  Growth Factor of New Deaths  19292 non-null  float64
dtypes: float64(14), int64(12), object(12)
memory usage: 6.3+ MB
```

Copying covid19_df into a covid19_new and drop all the unnecessary columns in covid19_new

In [6]:
```
covid19_new = covid19_df.copy()
covid19_new.drop(["Location ISO Code", "New Active Cases", "Total Active Cases", "Location Level", "City or Regency",
                  "Province", "Country", "Continent", "Island", "Time Zone", "Special Status", "Total Regencies",
                  "Total Cities", "Total Districts", "Total Urban Villages", "Total Rural Villages", "Area (km2)",
                  "Population", "Population Density", "Longitude", "Latitude", "New Cases per Million",
                  "Total Cases per Million", "New Deaths per Million", "Total Deaths per Million", "Total Deaths per 100rb",
                  "Case Fatality Rate", "Case Recovered Rate", "Growth Factor of New Cases", "Growth Factor of New Deaths"],
                 axis=1, inplace=True)
```

In [7]:
```
covid19_new.shape
```

Out[7]:
```
(21759, 8)
```

In [8]:
```
covid19_new.head(5)
```

Out[8]:

	Date	Location	New Cases	New Deaths	New Recovered	Total Cases	Total Deaths	Total Recovered
0	3/1/2020	DKI Jakarta	2	0	0	39	20	41
1	3/2/2020	DKI Jakarta	2	0	0	41	20	41
2	3/2/2020	Indonesia	2	0	0	2	0	0
3	3/2/2020	Riau	1	0	0	2	0	3
4	3/3/2020	DKI Jakarta	2	0	0	43	20	41

Columns processed have matched the requirements in the assignment

1. Find out what categories exist and how many neighbourhood belong to each category by using the value_counts() method

In [9]:
```
covid19_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21759 entries, 0 to 21758
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Date             21759 non-null  object
 1   Location         21759 non-null  object
 2   New Cases        21759 non-null  int64 
 3   New Deaths       21759 non-null  int64 
 4   New Recovered    21759 non-null  int64 
 5   Total Cases      21759 non-null  int64 
 6   Total Deaths     21759 non-null  int64 
 7   Total Recovered  21759 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 1.3+ MB
```

All attributes are numeric except for "Date" and "Location". "Date" and "Location" is an object and can contain any type of Python object. We can find out what categories exist in those two columns and how many neighbourhood belong to each category by using the value_counts() method.

Column "Date"

In [10]:
```
covid19_new.Date.value_counts()
```

Out[10]:
```
1/16/2021    35
4/30/2021    35
4/23/2021    35
4/24/2021    35
4/25/2021    35
             ..
3/4/2020      4
3/5/2020      4
3/2/2020      3
3/1/2020      1
12/3/2021     1
Name: Date, Length: 643, dtype: int64
```

Column "Location"

In [11]:
```
covid19_new.Location.value_counts()
```

Out[11]:
```
DKI Jakarta                  642
Indonesia                    642
Riau                         641
Jawa Barat                   640
Banten                       637
Jawa Tengah                  635
Sulawesi Tenggara            633
Bali                         632
Kalimantan Timur             629
Daerah Istimewa Yogyakarta   627
Sumatera Utara               626
Jawa Timur                   625
Sulawesi Selatan             624
Jambi                        623
Kepulauan Riau               623
Papua                        621
Maluku                       620
Sumatera Selatan             620
Aceh                         617
Kalimantan Tengah            617
Lampung                      617
Sulawesi Tengah              617
Sulawesi Utara               617
Sumatera Barat               617
Papua Barat                  616
Maluku Utara                 616
Kalimantan Utara             614
Sulawesi Barat               614
Kalimantan Barat             614
Kalimantan Selatan           613
Kepulauan Bangka Belitung    613
Nusa Tenggara Barat          612
Bengkulu                     611
Nusa Tenggara Timur          603
Gorontalo                    591
Name: Location, dtype: int64
```

2. Shows a summary of the numerical attributes

In [12]:
```
#Calculated Summary
print(covid19_new.sum(axis=1, numeric_only=True))
```

```
0            102
1            104
2              4
3              6
4            106
          ...
21754      69339
21755     179610
21756     119842
21757     212000
21758    8507362
Length: 21759, dtype: int64
```

In [13]:
```
#Mean
print(covid19_new.mean(numeric_only=True))
```

```
New Cases            391.293580
New Deaths            13.220415
New Recovered        377.310998
Total Cases        85259.970817
Total Deaths        2648.289352
Total Recovered    76712.602463
dtype: float64
```

In [14]:
```
#Standard Deviation
print(covid19_new.std(numeric_only=True))
```

```
New Cases            2074.551043
New Deaths             76.482617
New Recovered        1999.062563
Total Cases        368513.285849
Total Deaths        11776.011067
Total Recovered    340395.710890
dtype: float64
```

One of summarization technique is called 5-number summary which includes Median (50th percentile), 1st Quartile (25th percentile), 3rd Quartile (75th percentile), Minimum, and Maximum. We can see these data summarization using .describe() function.

In [15]:
```
#Summarizing Data
print(covid19_new.describe())
```

```
          New Cases    New Deaths  New Recovered   Total Cases   Total Deaths  \
count  21759.000000  21759.000000   21759.000000  2.175900e+04   21759.000000
mean     391.293580     13.220415     377.310998  8.525997e+04    2648.289352
std     2074.551043     76.482617    1999.062563  3.685133e+05   11776.011067
min        0.000000      0.000000       0.000000  1.000000e+00       0.000000
25%        7.000000      0.000000       4.000000  1.822500e+03      50.000000
50%       41.000000      1.000000      31.000000  1.078000e+04     283.000000
75%      151.000000      5.000000     143.000000  3.646450e+04    1050.000000
max    56757.000000   2069.000000   48832.000000  4.257243e+06  143858.000000

       Total Recovered
count     2.175900e+04
mean      7.671260e+04
std       3.403957e+05
min       0.000000e+00
25%       1.038500e+03
50%       8.745000e+03
75%       3.293250e+04
max       4.105680e+06
```

3. handling missing data (Replacing missing data with the mean value or others)

In [16]:
```
covid19_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21759 entries, 0 to 21758
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Date             21759 non-null  object
 1   Location         21759 non-null  object
 2   New Cases        21759 non-null  int64 
 3   New Deaths       21759 non-null  int64 
 4   New Recovered    21759 non-null  int64 
 5   Total Cases      21759 non-null  int64 
 6   Total Deaths     21759 non-null  int64 
 7   Total Recovered  21759 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 1.3+ MB
```

From the result of .info(), we know that there are 21759 rows in the dataset and every column (columns that are being processed for this assignment) have 21759 non-null count. So, that means there are no missing data in covid_new. To make sure, we will use .isnull() paired with .values.any() and .isna() paired with .sum() to check if there are missing data:

In [17]:
```
covid19_new.isna().sum()
```

Out[17]:
```
Date              0
Location          0
New Cases         0
New Deaths        0
New Recovered     0
Total Cases       0
Total Deaths      0
Total Recovered   0
dtype: int64
```

In [18]:
```
covid19_new.isnull().values.any()
```

Out[18]:
```
False
```

The output is **False** and there are 0 missing data in each column. So, we can confirm that there are no missing data.

But, in case of missing data, we can choose two ways to handle it:

1. Drop missing values
2. Replace missing values

We can use **.dropna()** function to drop missing values, but i think for this dataset containing COVID-19 information, it's better to replace missing values instead of dropping them.

Replace missing values

In this case, we will replace missing numerical data with the mean value for each column

In [19]:
```
covid19_new["New Cases"].fillna(covid19_new["New Cases"].mean())
```

Out[19]:
```
0         2
1         2
2         2
```

```
3          1
4          2
         ...
21754      2
21755      4
21756      4
21757      4
21758    245
Name: New Cases, Length: 21759, dtype: int64
```

In [20]:
```
covid19_new["New Deaths"].fillna(covid19_new["New Deaths"].mean())
```

Out[20]:
```
0        0
1        0
2        0
3        0
4        0
        ..
21754    0
21755    0
21756    0
21757    0
21758    8
Name: New Deaths, Length: 21759, dtype: int64
```

In [21]:
```
covid19_new["New Recovered"].fillna(covid19_new["New Recovered"].mean())
```

Out[21]:
```
0          0
1          0
2          0
3          0
4          0
         ...
21754      0
21755      0
21756      0
21757      1
21758    328
Name: New Recovered, Length: 21759, dtype: int64
```

In [22]:
```
covid19_new["Total Cases"].fillna(covid19_new["Total Cases"].mean())
```

Out[22]:
```
0            39
1            41
2             2
3             2
4            43
          ...
21754      34715
21755      89849
21756      59937
21757     106045
21758    4257243
Name: Total Cases, Length: 21759, dtype: int64
```

In [23]:
```
covid19_new["Total Deaths"].fillna(covid19_new["Total Deaths"].mean())
```

Out[23]:
```
0           20
1           20
2            0
3            0
4           20
          ...
21754     1056
21755     2152
21756     3071
21757     2889
21758   143858
Name: Total Deaths, Length: 21759, dtype: int64
```

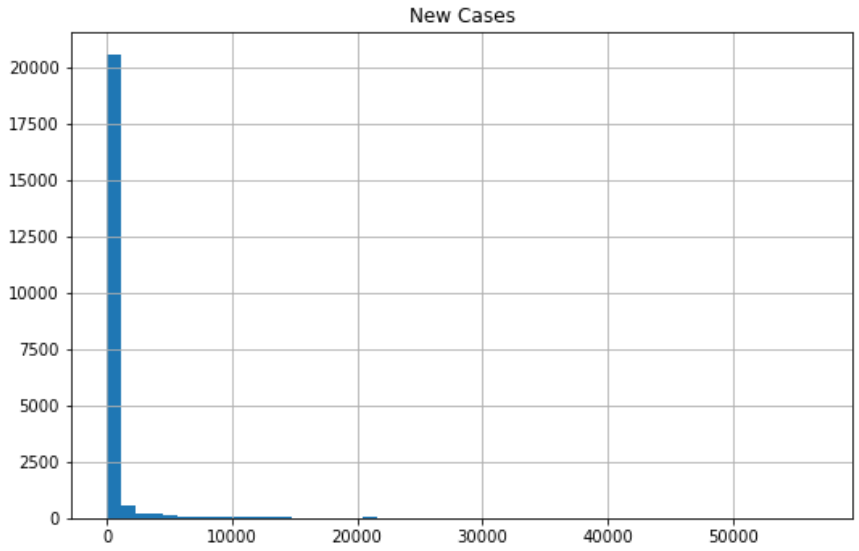
In [24]:
```
covid19_new["Total Recovered"].fillna(covid19_new["Total Recovered"].mean())
```

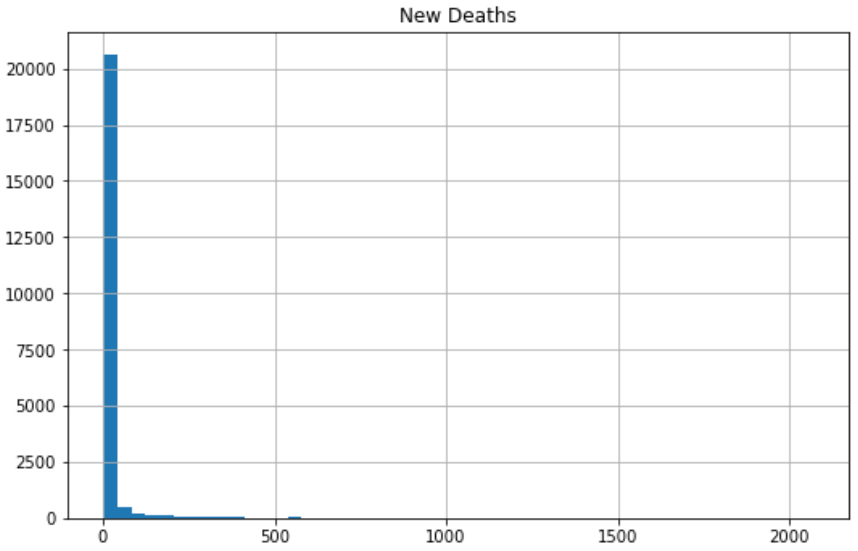
Out[24]:
```
0            41
1            41
2             0
3             3
4            41
          ...
21754     33566
21755     87605
21756     56830
21757    103061
21758   4105680
Name: Total Recovered, Length: 21759, dtype: int64
```

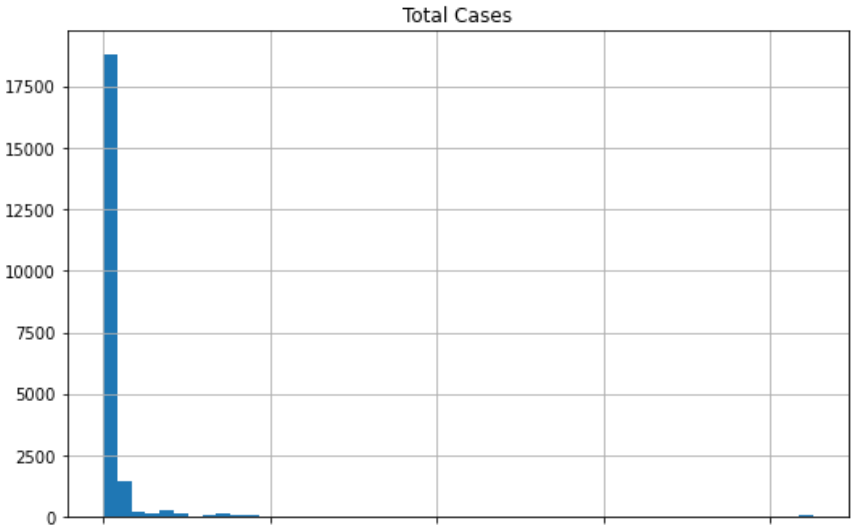
4. Plot a histogram for each numerical attribute

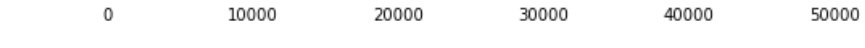
To plot a histogram for each numerical attribute, we can use .hist() and plt.show() function

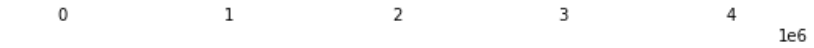
In [25]:
```
covid19_new.hist(bins=50,figsize=(20,20))
plt.show()
```

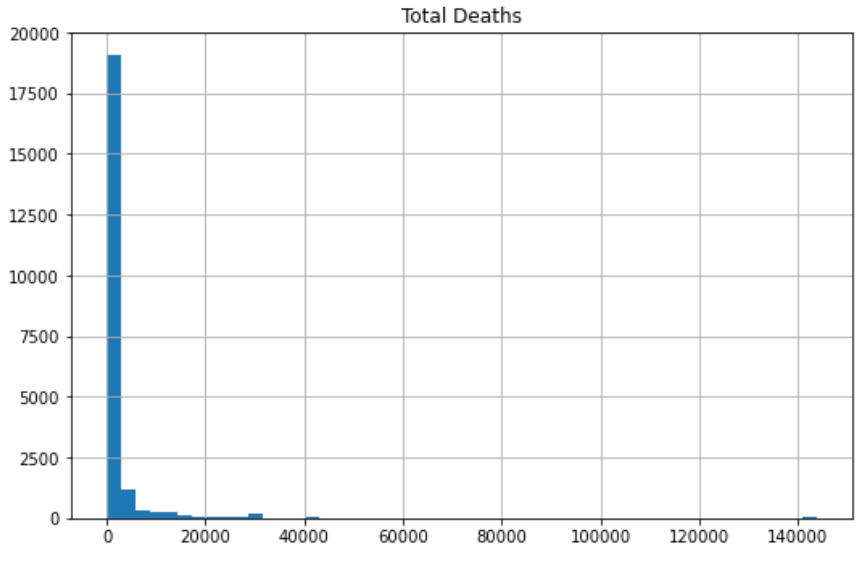


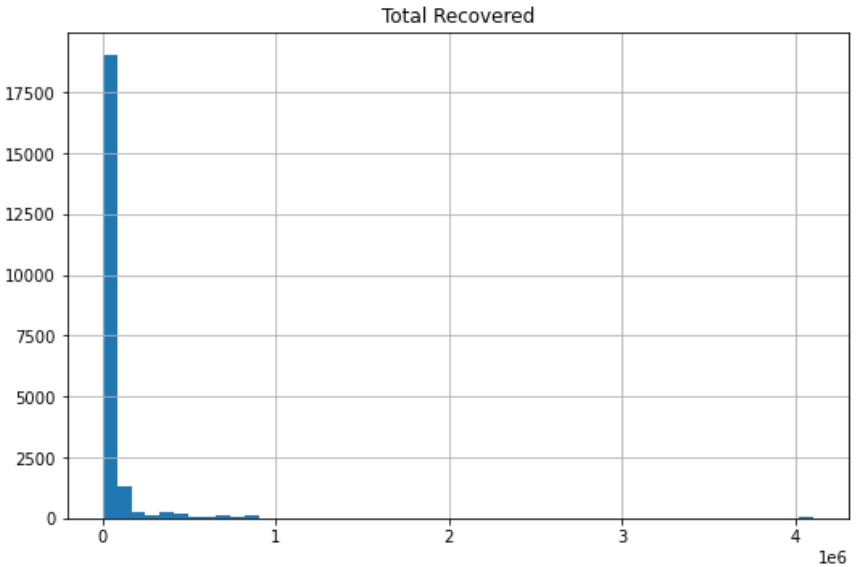





0
10000
20000
30000
40000
50000


0
1
2
3
4
1e6


Total Deaths
20000
17500
15000
12500
10000
7500
5000
2500
0
0
20000
40000
60000
80000
100000
120000
140000


Total Recovered
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
1e6