HIA302 Group Project Health Data Collections and Preparation

Name of Project: Data Preparation of Covid-19 dataset in Malaysia

Group: B

Group Members:

1) Lee Sin Yee
2) Hanis Binti Hasri

Objective: The objective of this project is to perform descriptive analysis (statistics) with Python on the dataset and suggest better management of the pandemic COVID-19 situation.

Pull data from the following GitHub link:

https://github.com/MoH-Malaysia/covid19-public

Group Project Github Link:
https://github.com/CindyLee84/HIA302-GROUP-PROJECT

Table of Content

Contribution

Topic	**Contributors**
Introduction Data Preparation Processes	Sin Yee & Hanis
Descriptive Statistic <u>Dataset 1: Covid-19 Age Group</u> No missing values detected - No Cleaning required covid_cases_age.describe( ) - find mean, standard deviation of each age group who diagnosed with Covid-19 infection <u>Dataset 2: Malaysia cases</u> find missing values from numpy import isnan Importing the SimpleImputer class for filling NaN To analyze the date frame of covid new cases <u>Dataset 3: Malaysia's State Covid-19 cases</u> find missing values from numpy import isnan Importing the SimpleImputer class for filling NaN List the states involved in the dataset find the percentage of covid-19 new cases in each state To investigate the states with the highest active cases of covid-19 (Jupyter Notebook) Do a comparison on covid cases between partially vaccination, fully vaccination and with booster in all the states Plot multiple line chart to compare the partially-vaccinated cases, fully-vaccinated cases and boosted cases in Selangor	Sin Yee

Plot a boxplot to compare the covid child cases, adolescent case, adult case and elderly case <u>Dataset 4: Covid-19 Death Age group</u> Find the missing values from numpy import isnan Importing the SimpleImputer class for filling NaN Find the age group that contributed the highest death in Johor Find the age group that contributed the highest death in Malaysia(vertical and horizontal barplot) <u>Dataset 5: hospital covid-19 cases</u> Compare the mean of covid beds in for state in Malaysia population Plot a barplot to compare the number of beds in all states of Malaysia for COVID-19 cases Compare the number of individuals or patients with covid in the hospital for each state in Malaysia population Plot a barplot and line chart to compare among admitted cases and discharged cases of covid-19 in all states <u>Dataset 6: rtk-ag test versus pcr test in Malaysia</u> Plot a boxplot to do a comparison on both rtk-ag and pcr test <u>Dataset 7: COVID-19 Tests in the state level</u> plot a barplot to compare the antigen rapid test "rtk-ag" and pcr test in all states of Malaysia. plot a barplot to compare the antigen rapid test "rtk-ag" and pcr test in all states of Malaysia after comparing with state population data Plot a barplot for the number of rtk-ag test in all states of Malaysia(Jupyter notebook) Plot a barplot for the number of pcr test in all states of Malaysia (Jupyter notebook) <u>Dataset 8: serious symptoms caused by different vaccines</u> Plot a barplot to compare the serious symptoms among vaccines sinovac, pfizer, astrazeneca and cansino Find the month that reported with highest number of serious symptoms - suspected anaphylaxis	

Dataset 9: Descriptive analysis of new deaths dataset Dataset 10: Descriptive analysis of ICU situation dataset Dataset 11: Descriptive analysis of the combination of dataset 9, 10 and new cases (state) dataset.	Hanis
Summary	Sin Yee & Hanis
Recommendation	Sin Yee & Hanis
Conclusion	Sin Yee & Hanis

Introduction

The data was taken from the Ministry of Health Malaysia COVID-19 dataset. It comprises extensive data from the beginning of the pandemic, until this report is written. The dataset was updated daily.

Following are the information provided in the dataset:

1. Epidemic
    a. cases_age
    b. cases_malaysia
    c. cases_state
    d. clusters
    e. deaths_malaysia
    f. deaths_state
    g. death_age
    h. hospital
    i. icu
    j. pkrc
    k. tests_malaysia
    l. tests_state
2. Vaccination
    a. aefi_serious
    b. aefi
    c. vax_malaysia
    d. vax_outcomes_capita
    e. vax_school
    f. vax_state
3. Mysejahtera
    a. checkin_malaysia_time
    b. checkin_malaysia
    c. checkin_state
    d. trace_malaysia
4. Notebooks
    a. Charts_casetrend
    b. sample_deaths_breakthrough
5. Static
    a. Population

Data Preparation Processes

Data preparation process is the process of cleaning and transforming raw data prior to data processing and analysis[1]. It is an important process that usually involves reformatting data, making corrections to data and the combining of data sets to improve the quality of data[1]. Data preparation helps to eliminate bias, for example, removing the possible outliers and standardizing the data formats contributed to enrich data. There are few key steps to data preparation process:

i) Gathering data/data collection

ii) Cleansing data

iii) Transform and enrich data

v) Store the data

Gathering data or data collection is the first step that involves collecting data from various sources such as databases, files, and external repositories[4]. Raw Data that is extracted originally exists in the form of complex unstructured and semi-structured sources such as PDF, text files and csv files. In this project, we extracted the datasets from github.com/MoH-Malaysia/covid19-public The advantages of increased access to data are less manual work required, provide faster insights, and therefore shortened the time to do analysis. Data cleaning involves removing duplicates and outliers, filling in missing values (NAN), conforming data to a standardized pattern, and masking private or sensitive data entries[3]. Once data has been cleansed, it must be validated by testing for possible errors[3].

Transforming data is the process of updating the format or data entries in order to produce a well-defined outcome, or to make the data more easily understood by a wider audience[2]. Enriching data refers to adding and connecting data with other related information to provide deeper insights. The preprocessed data can be transformed through scaling, decomposition, or aggregation[4]. Since most datasets contain features that differ greatly in terms of range, units, or magnitude, scaling is recommended to suppress this effect by normalizing all features to a similar level of magnitude[4]. With the help of scaling, some features which are too big will not be allowed to be applied as the main predictor[5]. If some values in the dataset are complicated, decomposing them into different constituent parts may be more valuable to a Machine Learning model[4]. Features of aggregation can be applied to connect related features together and decrease the dimensionality of an input set. Furthermore, in certain circumstances, combining multiple features into a single feature can be more effective for an algorithm[4].

To demonstrate by using python, step 1 of data cleaning is to load the data set by importing libraries for data preprocessing. **Numpy** is the library applied for all mathematical functions[5]. **Pandas** is the best function for importing and managing datasets[5]. **Matplotlib** is a tool to create charts[5]. These libraries can be imported with a shortcut alias as the following:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

Once the data set is downloaded and named as a .csv file, the data can be loaded into a pandas Dataframe to explore the data and do some basic cleaning functions removing data that are not required which causes the data processing slower[5]. Such cleaning includes removing the first line which contains extraneous text instead of column titles[5]. This text should be removed because it prevents the data set from being executed properly by the panda library[5]. The 2nd step of data cleaning is to explore the data set by reviewing the data set manually to avoid errors in the data analysis and the modeling process[5]. To ease the process, we create a Dataframe with the names of the columns, data types, the first row's values and description from the data dictionary. Extra attention should be paid on the columns that are poorly formatted which require more pre-processing work to be done[5].

The third step of data cleaning is to prepare features for Machine Learning where we need to handle missing values because the mathematics functions in most machine learning models predict that the data is numerical and should contain no missing values. Moreover, the **scikit-learn** library shows error if we try to apply a model like linear regression and logistic regression using data that contain missing or non-numeric values.

There are several ways to fill up missing values which are listed in the following and these decisions depend on the type of data, what are the objectives of the data, and the cause of values missing[5].

(i) Remove the lines with the data if data set is big enough and the percentage of missing values is high (for example, over 50%)[5];

(ii) Fill all null variables with 0 when dealing with numerical values[5];

(iii) Apply the Imputerclass from the scikit-learn library to fill in missing values with the data's (mean, median)[5]

(iv) Decide to fill up missing values with whatever value comes directly after it in the same column[5].

Inconsistent format of dates, such as dd/mm/yy and mm/dd/yy tend to appear in the same columns[5]. These date values might not be in the right data type, and therefore this will not effectively perform manipulations and get accurate results from it. In this case, the datetime package can be used to fix the type of the date[5].

The final step of data cleaning is to store or save the data to a csv file[5]. It is recommended to store the final output of each section or stage of the workflow in a separate CSV file[5]. In this way, we will be able to make changes in your data processing flow without having to recalculate everything[5]. After preparation, the data can be stored or channeled into a third-party application such as a business intelligence application to ease processing and data analysis[3].

Descriptive Statistic

Dataset 1: Covid-19 Age Group

```
url = 'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases
_age.csv'
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/e
pidemic/cases_malaysia.csv')
data
data.describe ( )
```

The function of describe( ) computes a summary of statistics pertaining to the DataFrame column[6]

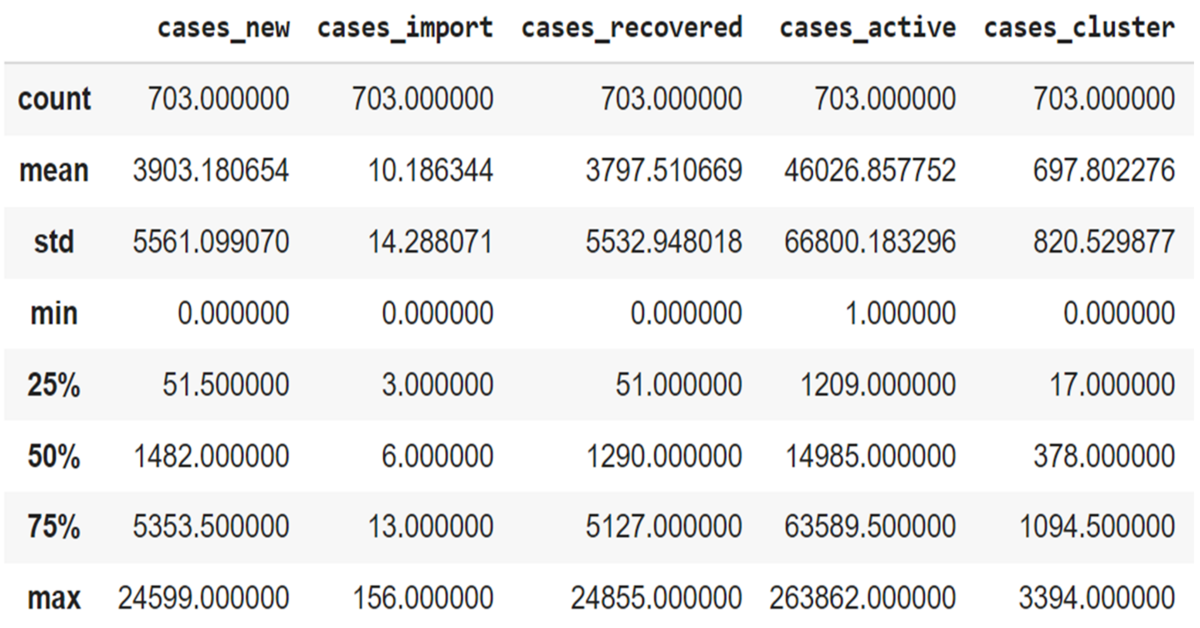
There are many methods used to compute descriptive statistics and other related operations on DataFrame. Most of these are functions of aggregations, for example, sum( ) and mean( ) and std( )[6]. sum( ) is the function that returns the sum of the values for the requested axis[6]. By default, the axis is index (axis=0)[6]. Mean( ) is the operation that returns the average value[6]. std( ) is the function that returns the Bressel standard deviation of the numerical columns[6].

Dataset 2: Malaysia cases (Data Cleaning)

```
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_mala
ysia.csv'
import pandas as pd
Msia_case =
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemi
c/cases_malaysia.csv')
Msia_case
```

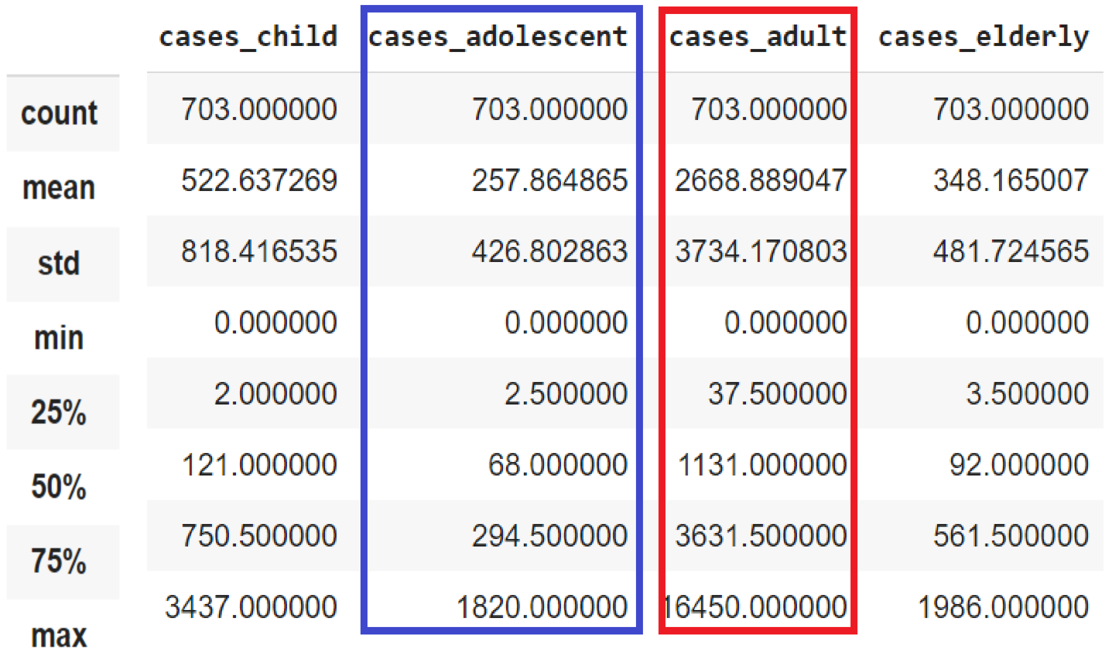
The three tables below were generated from Dataset 2:

New Cases, Import Cases, Recovered Cases, Active Cases and Cluster Cases in Malaysia



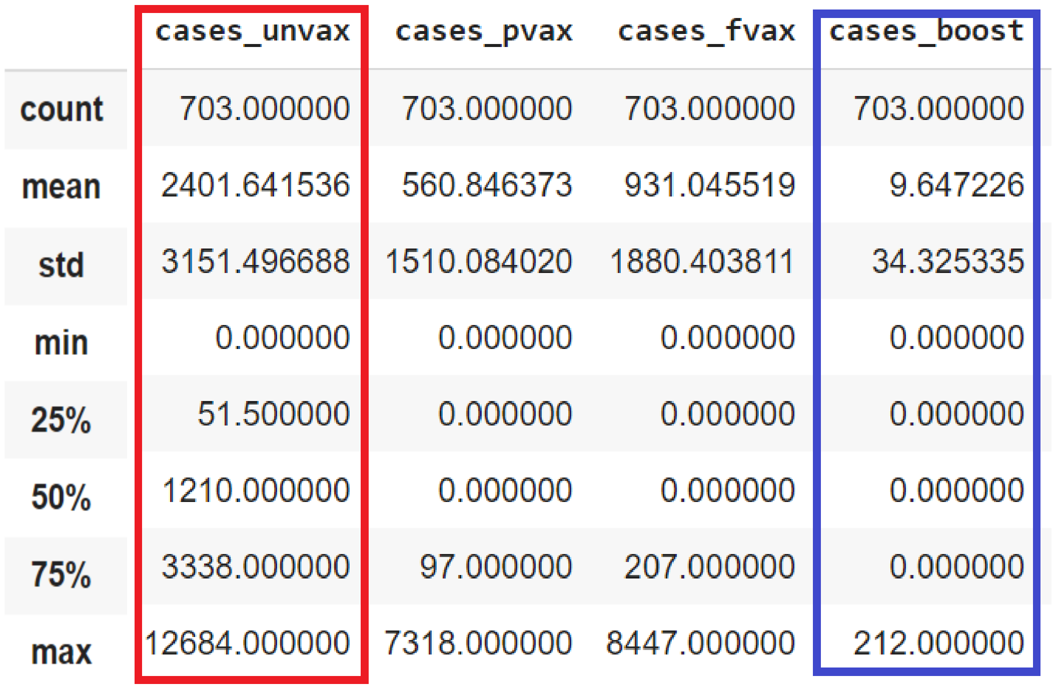
	cases_new	cases_import	cases_recovered	cases_active	cases_cluster
count	703.000000	703.000000	703.000000	703.000000	703.000000
mean	3903.180654	10.186344	3797.510669	46026.857752	697.802276
std	5561.099070	14.288071	5532.948018	66800.183296	820.529877
min	0.000000	0.000000	0.000000	1.000000	0.000000
25%	51.500000	3.000000	51.000000	1209.000000	17.000000
50%	1482.000000	6.000000	1290.000000	14985.000000	378.000000
75%	5353.500000	13.000000	5127.000000	63589.500000	1094.500000
max	24599.000000	156.000000	24855.000000	263862.000000	3394.000000

Population: Child Cases, Adolescent Cases, Adult Cases and Elderly Cases



	cases_child	cases_adolescent	cases_adult	cases_elderly
count	703.000000	703.000000	703.000000	703.000000
mean	522.637269	257.864865	2668.889047	348.165007
std	818.416535	426.802863	3734.170803	481.724565
min	0.000000	0.000000	0.000000	0.000000
25%	2.000000	2.500000	37.500000	3.500000
50%	121.000000	68.000000	1131.000000	92.000000
75%	750.500000	294.500000	3631.500000	561.500000
max	3437.000000	1820.000000	16450.000000	1986.000000

<u>Unvaccinated cases, partially-partially vaccinated cases, fully vaccinated and booster cases</u>



	cases_unvax	cases_pvax	cases_fvax	cases_boost
count	703.000000	703.000000	703.000000	703.000000
mean	2401.641536	560.846373	931.045519	9.647226
std	3151.496688	1510.084020	1880.403811	34.325335
min	0.000000	0.000000	0.000000	0.000000
25%	51.500000	0.000000	0.000000	0.000000
50%	1210.000000	0.000000	0.000000	0.000000
75%	3338.000000	97.000000	207.000000	0.000000
max	12684.000000	7318.000000	8447.000000	212.000000

In dataset 2 Malaysia cases, by applying the data.describe ( ), we are able to extract the value of count, mean, standard deviation, minimum, 25% range, 50% range, 75% range and maximum new cases, import cases, recovered cases, active cases, cluster cases, child cases, adolescent cases, adult cases, unvaccinated cases, partially vaccinated cases, fully vaccinated cases, booster cases and different category of cluster cases in Malaysia.

However, we have detected many missing values (NaN) in the dataset:

```
# find missing values

from numpy import isnan
from pandas import read_csv

# load dataset

import pandas as pd
Msia_case=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemi
c/cases_malaysia.csv')
Msia_case.isnull( ).sum( )
```

```
date                         0

cases_new                    0

cases_import                 0

cases_recovered              0

cases_active                 0

cases_cluster                1

cases_unvax                  1

cases_pvax                   1

cases_fvax                   1

cases_boost                  1

cases_child                  1

cases_adolescent             1

cases_adult                  1

cases_elderly                1

cases_0_4                    1

cases_5_11                   1

cases_12_17                  1

cases_18_29                  1

cases_30_39                  1

cases_40_49                  1

cases_50_59                  1

cases_60_69                  1

cases_70_79                  1
```

```
cases_80                    1
cluster_import            342
cluster_religious         342
cluster_community         342
cluster_highRisk          342
cluster_education         342
cluster_detentionCentre   342
cluster_workplace         342
dtype: int64
```

Therefore, we fill the missing values with mean by using SimpleImputer class:

```
import pandas as pd
import numpy as np
```

# Importing the SimpleImputer class

```
from sklearn.impute import SimpleImputer
# Imputer object using the mean strategy and
# missing_values type for imputation
imputer = SimpleImputer(missing_values = np.nan,
                        strategy ='mean')
```

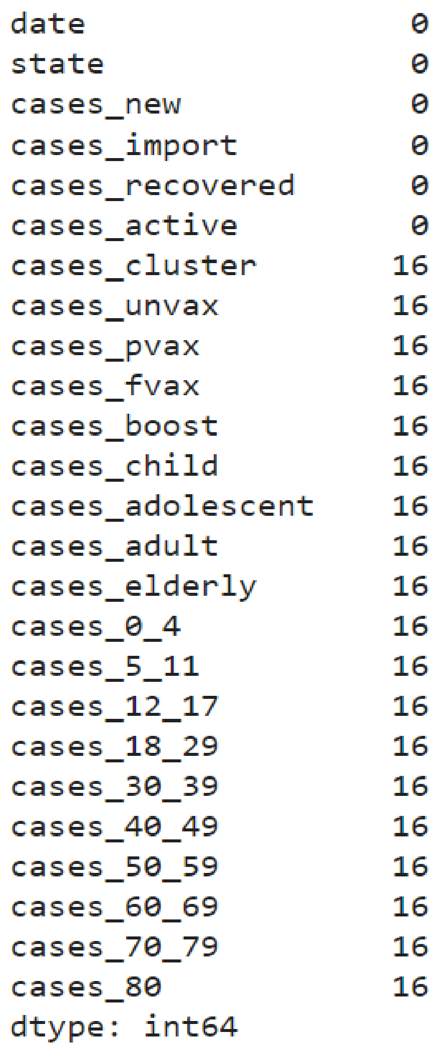
*The complete codes is documented in Jupyter Notebook HIA302 Group Project

<u>Dataset 3: Malaysia's State Covid-19 cases (Data Cleaning)</u>

```
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_state
.csv'
import pandas as pd
state_case =
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemi
c/cases_state.csv')
state_case
```

In this dataset, we also found some missing values (NaN) as shown below:

state_case.isnull( ).sum( )



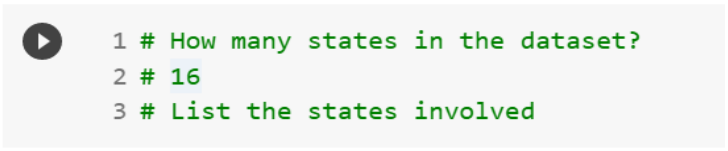
```
date                0
state               0
cases_new           0
cases_import        0
cases_recovered     0
cases_active        0
cases_cluster      16
cases_unvax        16
cases_pvax         16
cases_fvax         16
cases_boost        16
cases_child        16
cases_adolescent   16
cases_adult        16
cases_elderly      16
cases_0_4          16
cases_5_11         16
cases_12_17        16
cases_18_29        16
cases_30_39        16
cases_40_49        16
cases_50_59        16
cases_60_69        16
cases_70_79        16
cases_80           16
dtype: int64
```

Data cleaning done by filling the missing values with mean values:

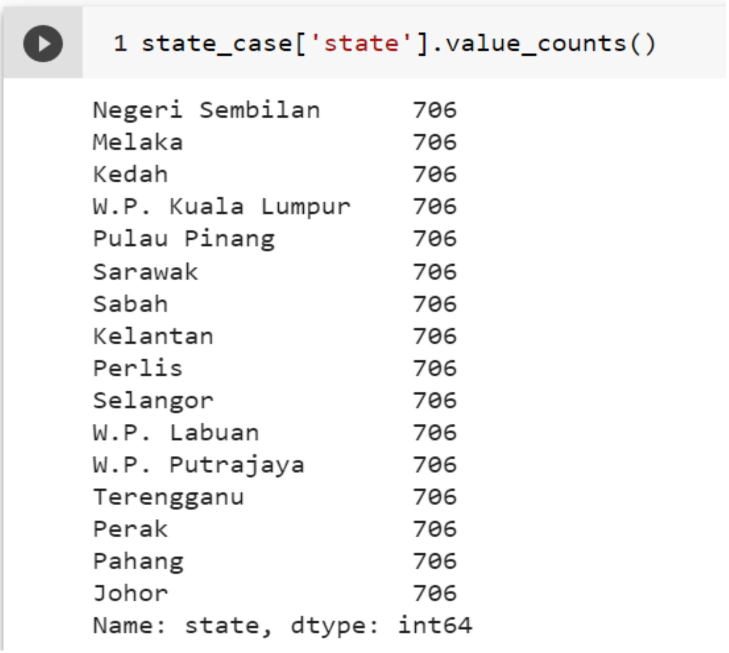
```
import pandas as pd
import numpy as np

# Importing the SimpleImputer class

from sklearn.impute import SimpleImputer

# Imputer object using the mean strategy and
# missing_values type for imputation

imputer = SimpleImputer(missing_values = np.nan,
                    strategy ='mean')
```

*refer the complete data cleaning codes and output in Jupyter Notebook HIA302 Group Project

<u>List the states that are involved in the dataset:</u>



```
# How many states in the dataset?
# 16
# List the states involved
```

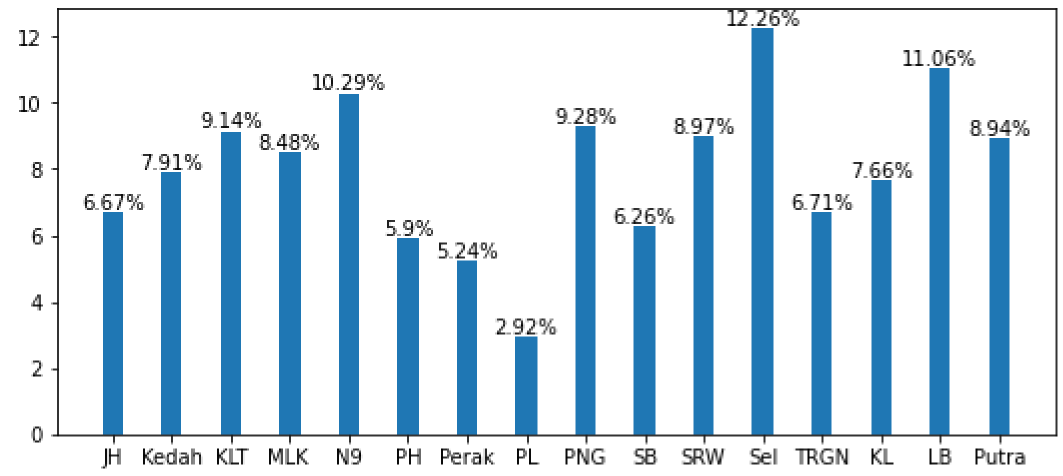


```
state_case['state'].value_counts()
```

```
Negeri Sembilan      706
Melaka               706
Kedah                706
W.P. Kuala Lumpur    706
Pulau Pinang         706
Sarawak              706
Sabah                706
Kelantan             706
Perlis               706
Selangor             706
W.P. Labuan          706
W.P. Putrajaya       706
Terengganu           706
Perak                706
Pahang               706
Johor                706
Name: state, dtype: int64
```

# Plot a barplot to compare Malaysia's states new COVID-19 cases

import matplotlib.pyplot as plt



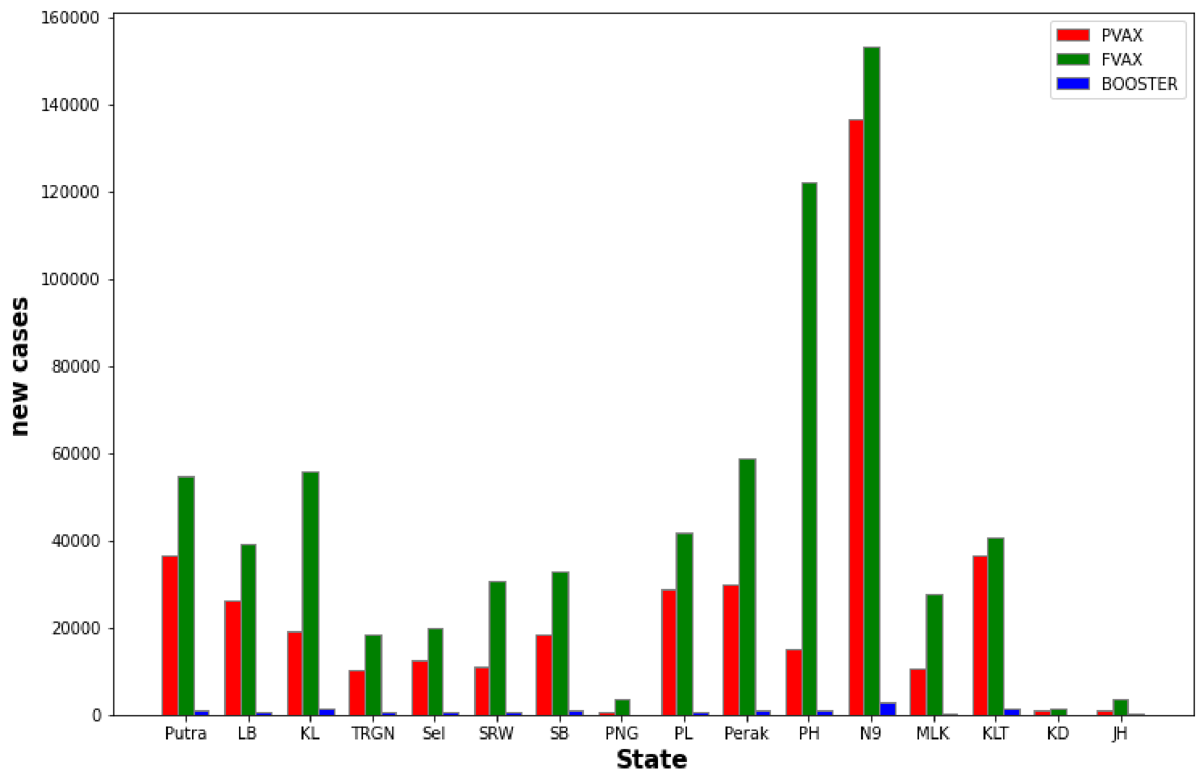
Putra	*Putrajaya*
LB	*Labuan*
KL	*Kuala Lumpur*
TRGN	*Terengganu*
Sel	*Selangor*
SRW	*Sarawak*
SB	*Sabah*
PNG	*Penang*
PL	*Perlis*
Perak	*Perak*
PH	*Pahang*
N9	*Negeri Sembilan*
MLK	*Melaka*
KLT	*Kelantan*
KD	*Kedah*
JH	*Johor*

*Note: Abbreviation of the States in the dataset

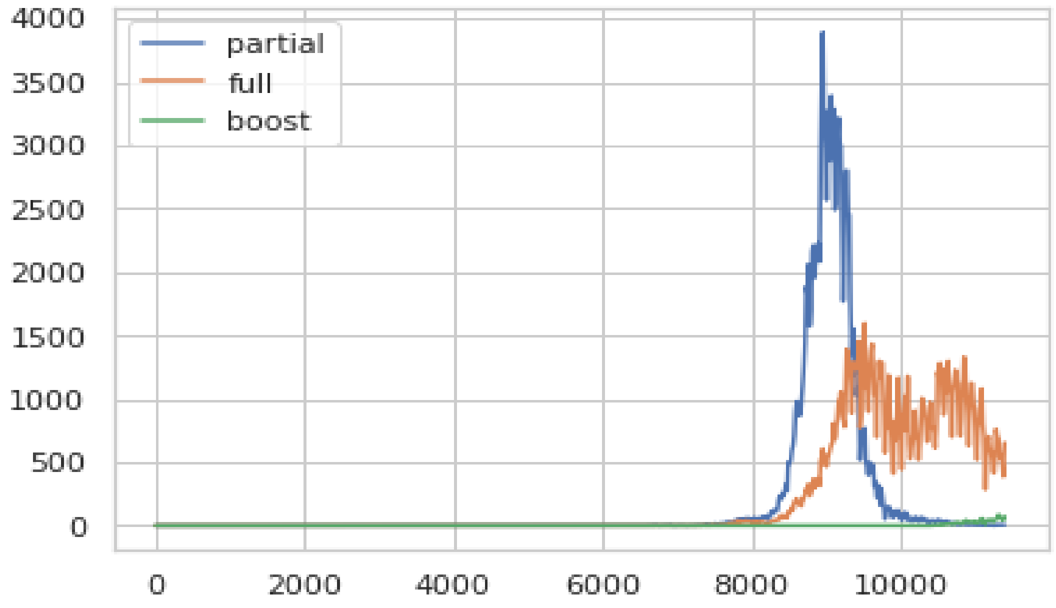
```
# Plot another barplot to compare among cases that are partially vaccinated (pvac), fully vacc
inated and complete dose with booster in all states.
import numpy as np
import matplotlib.pyplot as plt

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

# set height of bar
PVAX = [39, 39, 1094, 352, 3890, 291, 1085, 766, 31, 628, 220, 412, 676, 376, 613, 958]
FVAX = [62, 24, 443, 385, 1596, 3610, 872, 651, 134, 536, 344, 213, 256, 613, 460, 671]
BOOSTER = [5, 3, 27, 18, 64, 34, 22, 19, 2, 27, 26, 15, 23, 33, 37, 21]
# Set position of bar on X axis
br1 = np.arange(len(PVAX))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

# Make the plot
plt.bar(br1, PVAX, color ='r', width = barWidth,
        edgecolor ='grey', label ='PVAX')
plt.bar(br2, FVAX, color ='g', width = barWidth,
        edgecolor ='grey', label ='FVAX')
plt.bar(br3, BOOSTER, color ='b', width = barWidth,
        edgecolor ='grey', label ='BOOSTER')
```

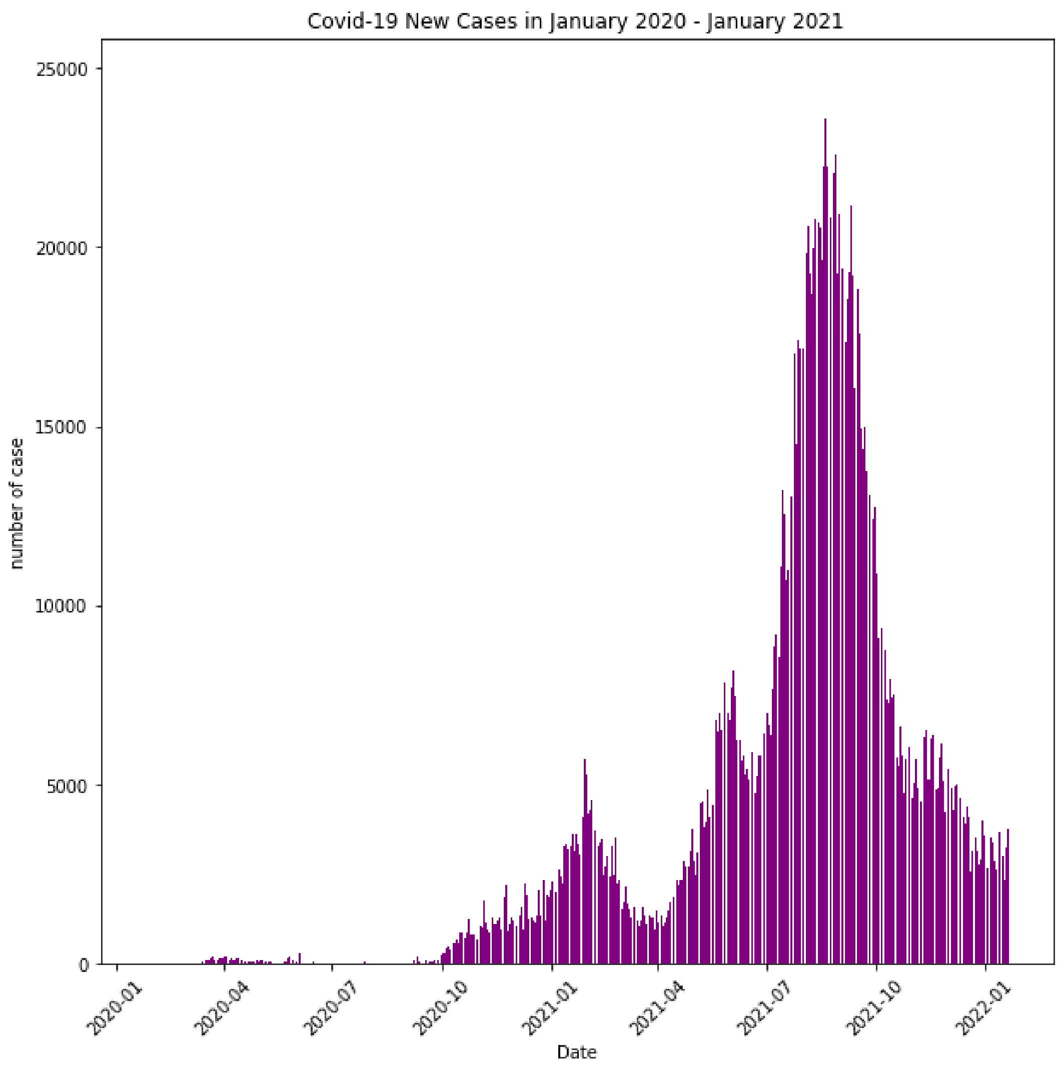
```
# Adding Xticks
plt.xlabel('State', fontweight ='bold', fontsize = 15)
plt.ylabel('new cases', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(PVAX))],
        ['Putra', 'LB', 'KL', 'TRGN', 'Sel', 'SRW', 'SB', 'PNG', 'PL', 'Perak', 'PH', 'N9', 'MLK', 'KL
T', 'KD','JH'])
plt.legend( )
plt.show( )
```

Partially-vaccination, fully-vaccination, and with booster covid-19 Cases



Selangor Partial-vaccinated, fully vaccinated, and with booster Covid-19 cases





Above demonstrated the disease pattern according to the dataframe from January 2020 to January 2021, the libraries imported:

```
import os
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

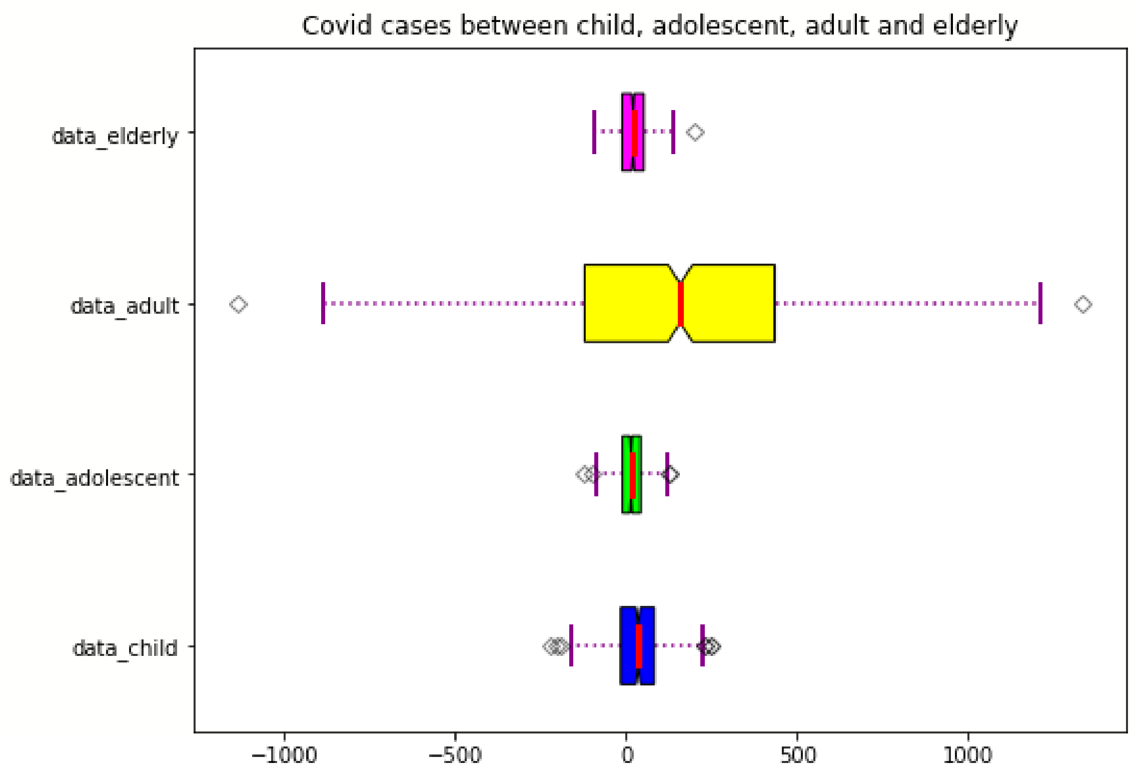
Standard deviation values can be used to plot a boxplot as shown below:

```
import matplotlib.pyplot as plt
import numpy as np
```

# Creating dataset

```
np.random.seed(10)
data_child = np.random.normal(32.545048, 79.708481, 600)
data_adolescent = np.random.normal(16.024501 ,41.433625, 600)
data_adult = np.random.normal(165.724037, 419.303178, 600)
data_elderly = np.random.normal(21.800636, 48.383413, 600)
data = [data_child, data_adolescent, data_adult, data_elderly]
```

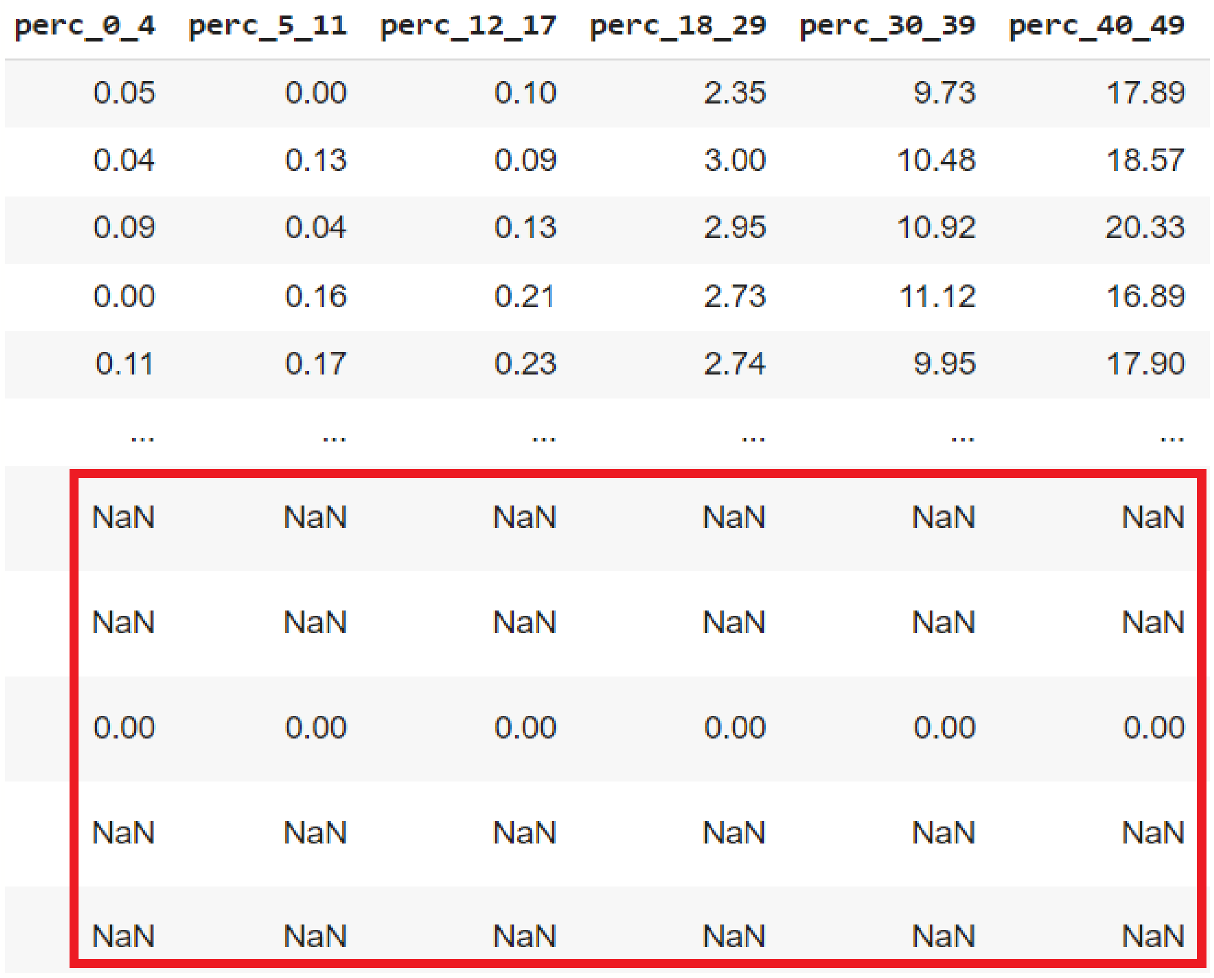
A Box Plot which is also known as Whisker plot is created to demonstrate the summary of the set of data values having properties of **minimum**, **first quartile**, **median**, **third quartile** and **maximum**[8]. In the box plot, a box is plotted from the first quartile to the third quartile, a vertical line exists and goes through the box at the median[8].



Dataset 4: Covid-19 Death Age group (Data Cleaning)

```
url=
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/deaths_age
.csv'
import pandas as pd
deaths_age=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemi
c/deaths_age.csv')
deaths_age
```

From the deaths age dataset, we found that there are some missing values in the state of Labuan and Putrajaya detected for the attribute of age group percentage.



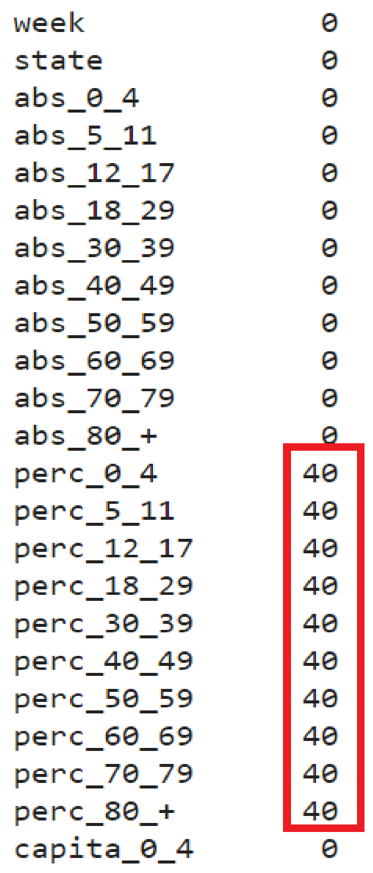
perc_0_4	perc_5_11	perc_12_17	perc_18_29	perc_30_39	perc_40_49
0.05	0.00	0.10	2.35	9.73	17.89
0.04	0.13	0.09	3.00	10.48	18.57
0.09	0.04	0.13	2.95	10.92	20.33
0.00	0.16	0.21	2.73	11.12	16.89
0.11	0.17	0.23	2.74	9.95	17.90
...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
0.00	0.00	0.00	0.00	0.00	0.00
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN

While creating Data Frame from a csv file, many blank columns are imported as null values into the Data Frame which later creates errors while processing that data frame[7]. Pandas isnull( ) method is used to detect and manage NULL values in a dataframe[7].

```
from numpy import isnan
from pandas import read_csv
```

# load dataset

```
url =
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/deaths_age.csv'
import pandas as pd
deaths_age =
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/deaths_age.csv')
deaths_age.isnull().sum()
```



```
week           0
state          0
abs_0_4        0
abs_5_11       0
abs_12_17      0
abs_18_29      0
abs_30_39      0
abs_40_49      0
abs_50_59      0
abs_60_69      0
abs_70_79      0
abs_80_+       0
perc_0_4      40
perc_5_11     40
perc_12_17    40
perc_18_29    40
perc_30_39    40
perc_40_49    40
perc_50_59    40
perc_60_69    40
perc_70_79    40
perc_80_+     40
capita_0_4     0
```

From the above evaluation, we found that there are 40 missing values in each category of percentage for all age groups.

There are two methods for data cleaning, first we can choose to drop the nan value in the csv file :

# making data frame from csv file

```
import pandas as pd
deaths_age
=pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epide
mic/deaths_age.csv')
```

# making new data frame with dropped NA values

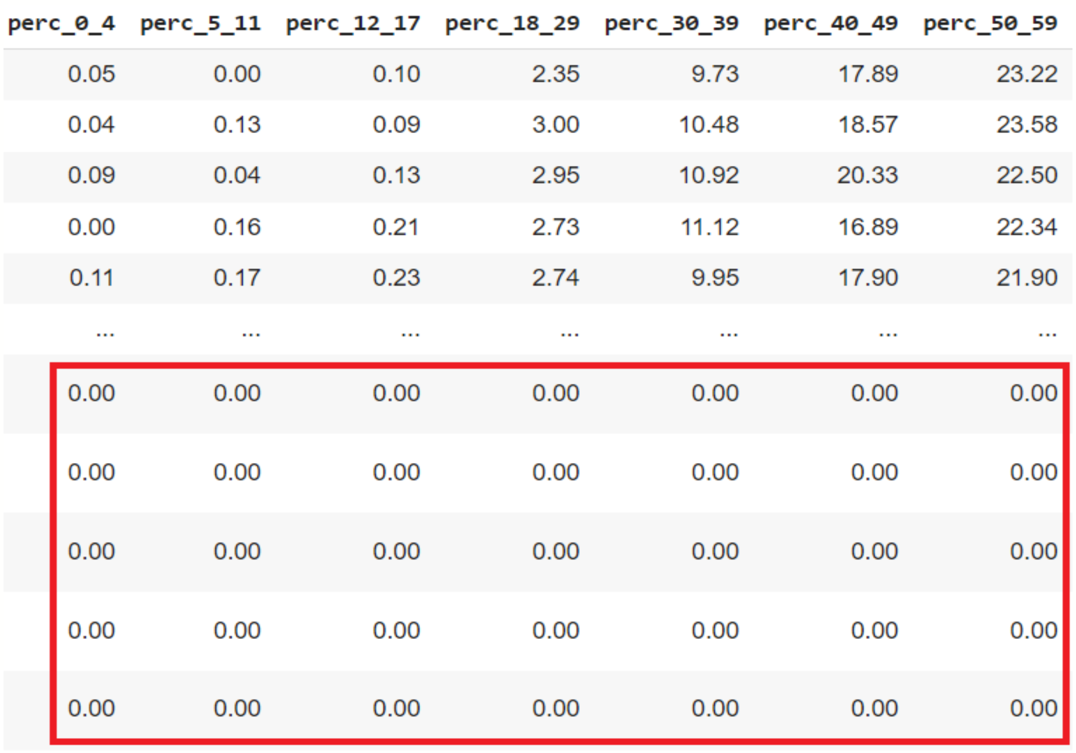
```
new_deaths_age = deaths_age.dropna(axis = 0, how ='any')
new_deaths_age
```

#Now we compare sizes of data frames so that we can come to know how many rows had at least 1 Null value

```
print("Old data frame length:", len(deaths_age))
print("New data frame length:", len(new_deaths_age))
print("Number of rows with at least 1 NA value: ", (len(deaths_age)-len(new_deaths_age)))
```

Output:

```
Old data frame length: 432
New data frame length: 392
Number of rows with at least 1 NA value:  40
```

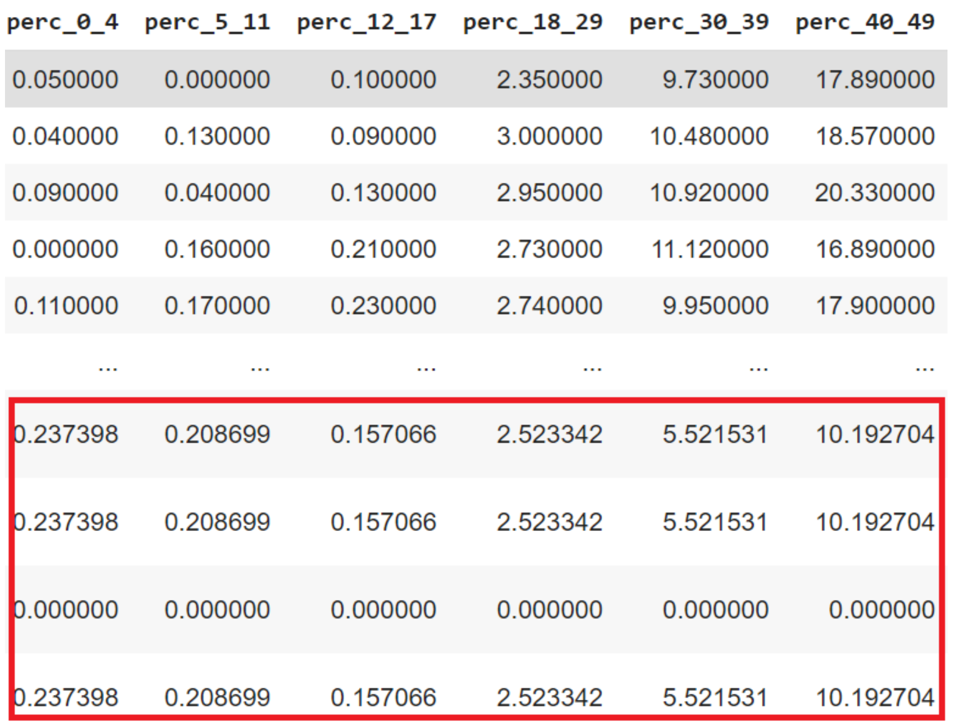


perc_0_4	perc_5_11	perc_12_17	perc_18_29	perc_30_39	perc_40_49	perc_50_59
0.05	0.00	0.10	2.35	9.73	17.89	23.22
0.04	0.13	0.09	3.00	10.48	18.57	23.58
0.09	0.04	0.13	2.95	10.92	20.33	22.50
0.00	0.16	0.21	2.73	11.12	16.89	22.34
0.11	0.17	0.23	2.74	9.95	17.90	21.90
...	...	...	...	...	...	...
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00

Second method is to use mean to find the na values by column:

```
import pandas as pd
import numpy as np

# Importing the SimpleImputer class

from sklearn.impute import SimpleImputer

# Imputer object using the mean strategy and

# missing_values type for imputation

imputer = SimpleImputer(missing_values = np.nan,
                        strategy ='mean')
deaths_age=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemi
c/deaths_age.csv')
deaths_age['perc_0_4'] = deaths_age['perc_0_4'].fillna(deaths_age['perc_0_4'].mean())
deaths_age['perc_5_11']= deaths_age['perc_5_11'].fillna(deaths_age['perc_5_11'].mean())
deaths_age['perc_12_17']= deaths_age['perc_12_17'].fillna(deaths_age['perc_12_17'].mean())
deaths_age['perc_18_29']= deaths_age['perc_18_29'].fillna(deaths_age['perc_18_29'].mean())
deaths_age['perc_30_39']= deaths_age['perc_30_39'].fillna(deaths_age['perc_30_39'].mean())
deaths_age['perc_40_49']= deaths_age['perc_40_49'].fillna(deaths_age['perc_40_49'].mean())
deaths_age['perc_50_59']= deaths_age['perc_50_59'].fillna(deaths_age['perc_50_59'].mean())
deaths_age['perc_60_69']= deaths_age['perc_60_69'].fillna(deaths_age['perc_60_69'].mean())
deaths_age['perc_70_79']= deaths_age['perc_70_79'].fillna(deaths_age['perc_70_79'].mean())
deaths_age['perc_80_+']= deaths_age['perc_80_+'].fillna(deaths_age['perc_80_+'].mean())
```

deaths_age output:



perc_0_4	perc_5_11	perc_12_17	perc_18_29	perc_30_39	perc_40_49
0.050000	0.000000	0.100000	2.350000	9.730000	17.890000
0.040000	0.130000	0.090000	3.000000	10.480000	18.570000
0.090000	0.040000	0.130000	2.950000	10.920000	20.330000
0.000000	0.160000	0.210000	2.730000	11.120000	16.890000
0.110000	0.170000	0.230000	2.740000	9.950000	17.900000
...	...	...	...	...	...
0.237398	0.208699	0.157066	2.523342	5.521531	10.192704
0.237398	0.208699	0.157066	2.523342	5.521531	10.192704
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.237398	0.208699	0.157066	2.523342	5.521531	10.192704

We decided to apply the second method for cleaning the data for the age group of covid-19 death cases, which is to fill with mean value to replace the missing values.

# From the value count below we found that the data is included “Malaysia” in the attribute of state,

#Therefore, we need to extract only data for “Malaysia” in this case

deaths_age['state'].value_counts( )

```
Pahang              24
Perak               24
Sarawak             24
Johor               24
Kelantan            24
Sabah               24
Klang Valley        24
Negeri Sembilan     24
```

```
W.P. Labuan         24
W.P. Putrajaya      24
Kedah               24
Melaka              24
Perlis              24
Selangor            24
Malaysia            24
Pulau Pinang        24
W.P. Kuala Lumpur   24
Terengganu          24
Name: state, dtype: int64
```

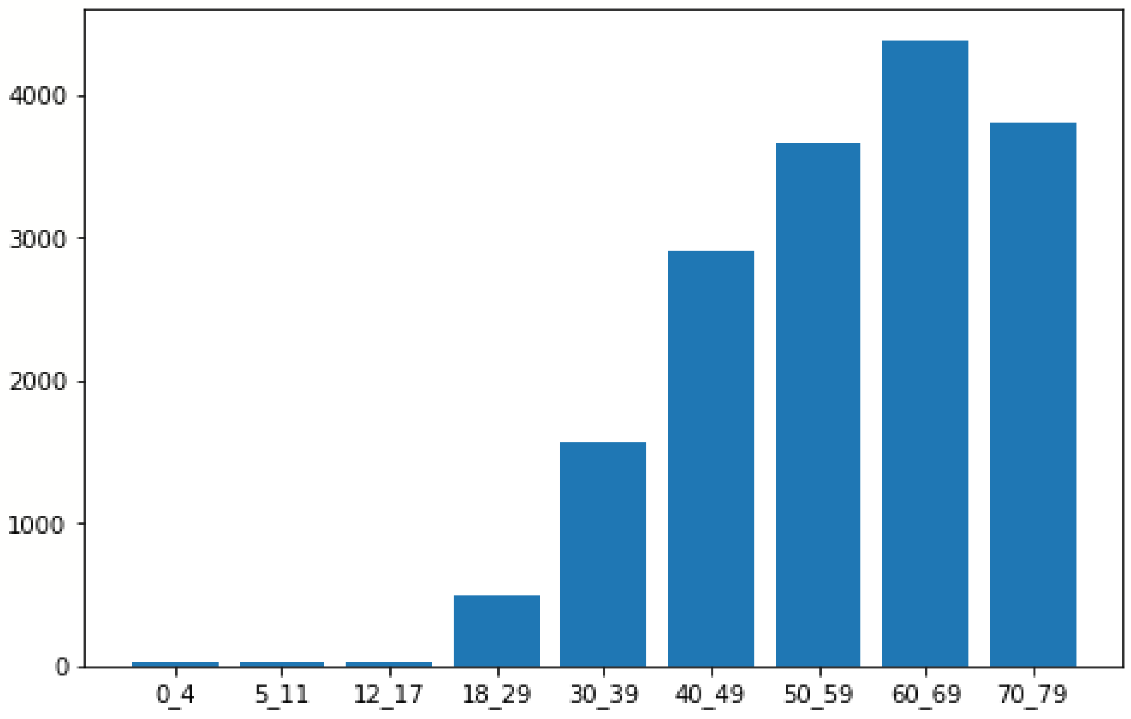
# to find the age-group which reported with highest death in Malaysia:

```
Malaysia_deaths = deaths_age.query('state=="Malaysia"')
Malaysia_deaths
```

# Plot a barplot to demonstrate a clearer picture on death cases of each age group in Malaysia.

```
from matplotlib import pyplot as plt
```

Age group of Malaysia Covid-19 Death Cases



#we could also perform horizontal plot with labeled number of death cases:

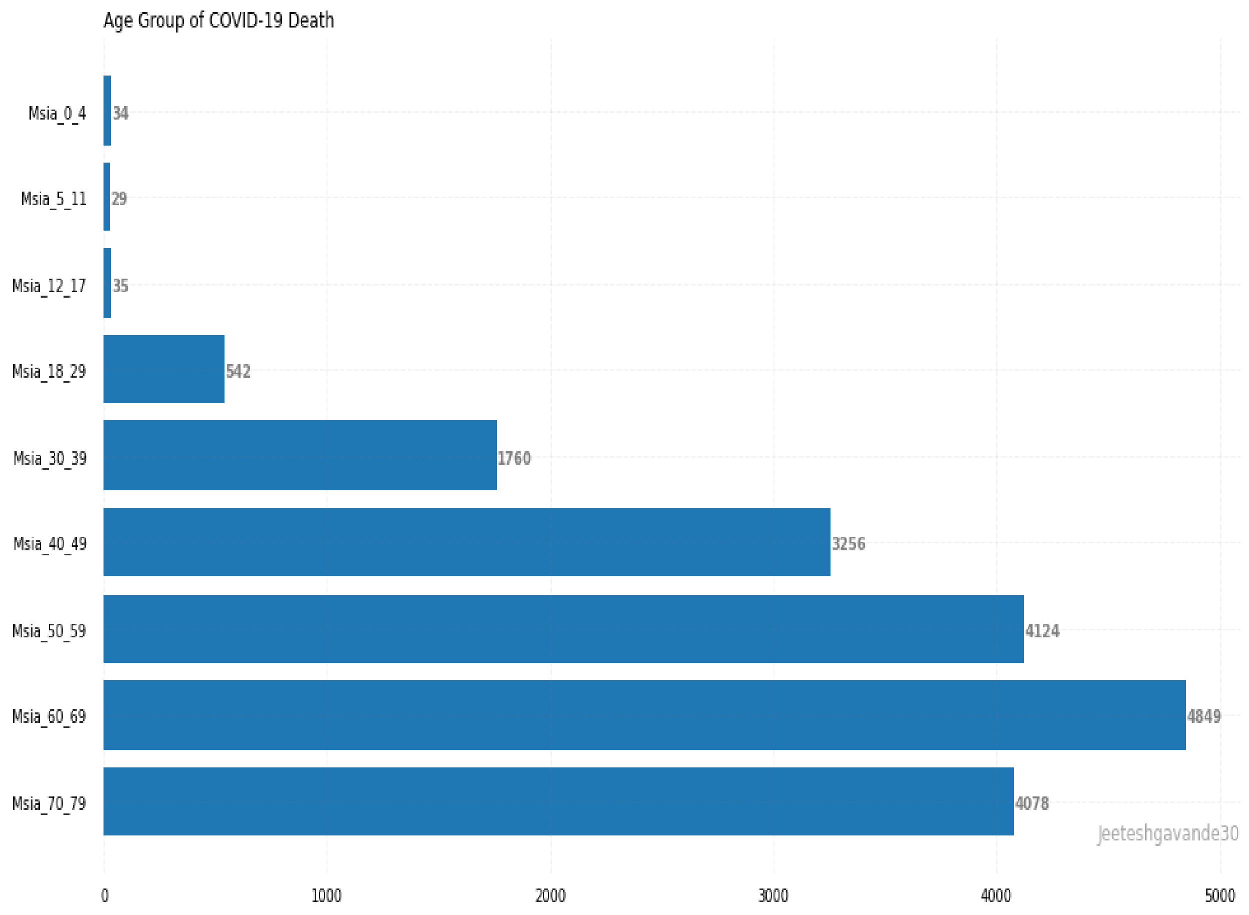
# Figure Size

fig, ax = plt.subplots(figsize =(16, 9))

# Horizontal Bar Plot

ax.barh(country_death, deaths_agegroup)

***complete codes is documented in Jupyter Notebook HIA302 Group Project**

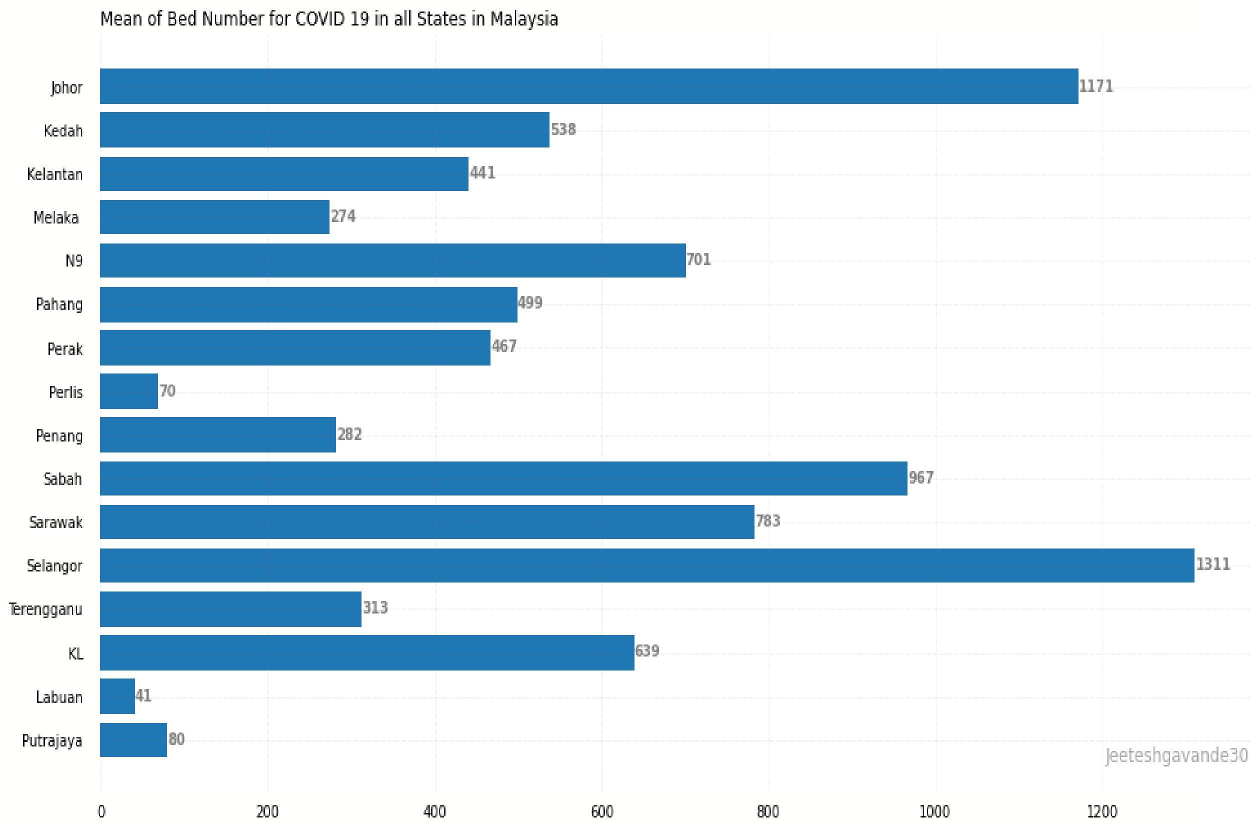


Dataset 5: hospital covid-19 cases

url=
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/hospital.csv'

import pandas as pd
hospital_covid=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/hospital.csv')
hospital_covid

# Plot a barplot to compare the number of beds in all states of Malaysia for COVID-19 cases using mean( ) - refer to Jupyter Notebook for codes



# hosp_x: total number of individuals in category x in hospitals; this is a stock variable altered by flows from admissions and discharges

# hosp_covid: total number of individuals in category covid in hospitals

Compare the number of individuals or patients with covid in the hospital each state in Malaysia population

```
hospital_covid.groupby('state').hosp_covid.sum()
```

```
state
Johor                  352442
Kedah                  148896
Kelantan               160163
Melaka                  81561
Negeri Sembilan        171173
Pahang                  98867
Perak                  132345
Perlis                  14743
Pulau Pinang           105237
Sabah                  248992
Sarawak                169045
Selangor               491337
Terengganu              69597
W.P. Kuala Lumpur      224703
W.P. Labuan              5254
W.P. Putrajaya           9122
Name: hosp_covid, dtype: int64
```

#To take into account “Malaysia state population”

```
'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/static/population.csv'
import pandas as pd
population=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/static/population.csv')
population
```

# to find the percentage of number of individuals/patients with covid-19 infection in hospital of each state for Malaysia population

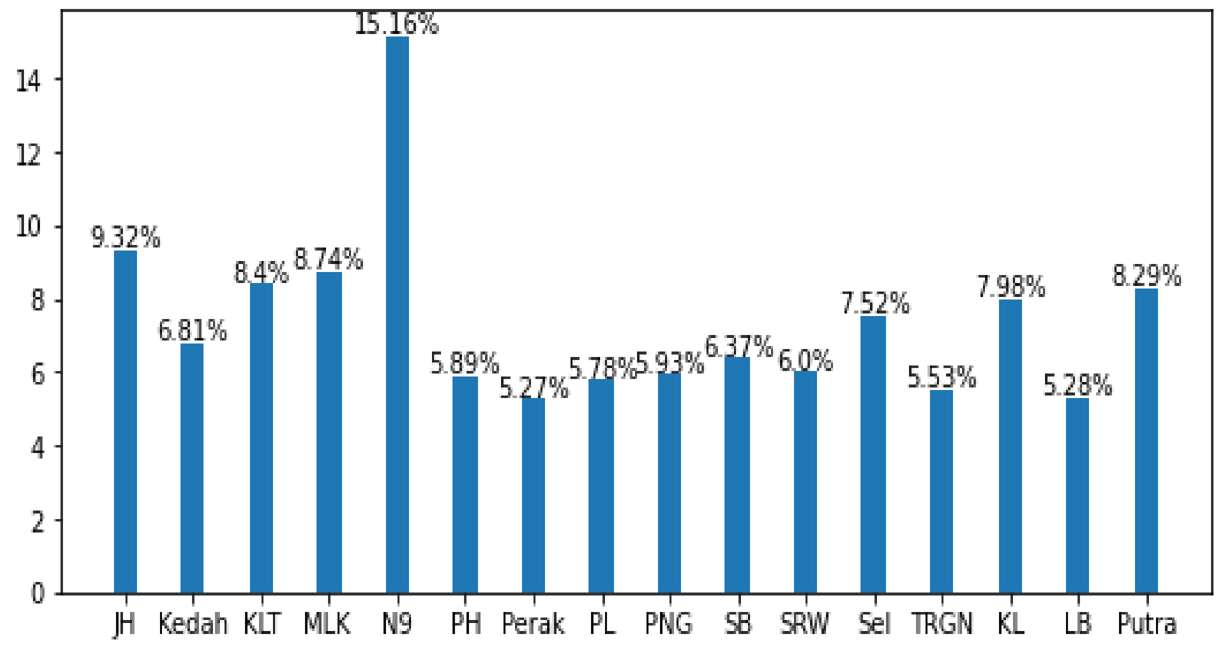
# (the total number of individuals infected by covid in hospital/state population) * 100

percent_covid_individuals_Johor=(352442/3781000)*100
The same calculation done for the all the states involved in the dataset

# percentage of number of individuals or patients with covid in the hospital of each state

from matplotlib import pyplot as plt
import numpy as np

***complete codes is documented in Jupyter Notebook HIA302 Group Project**



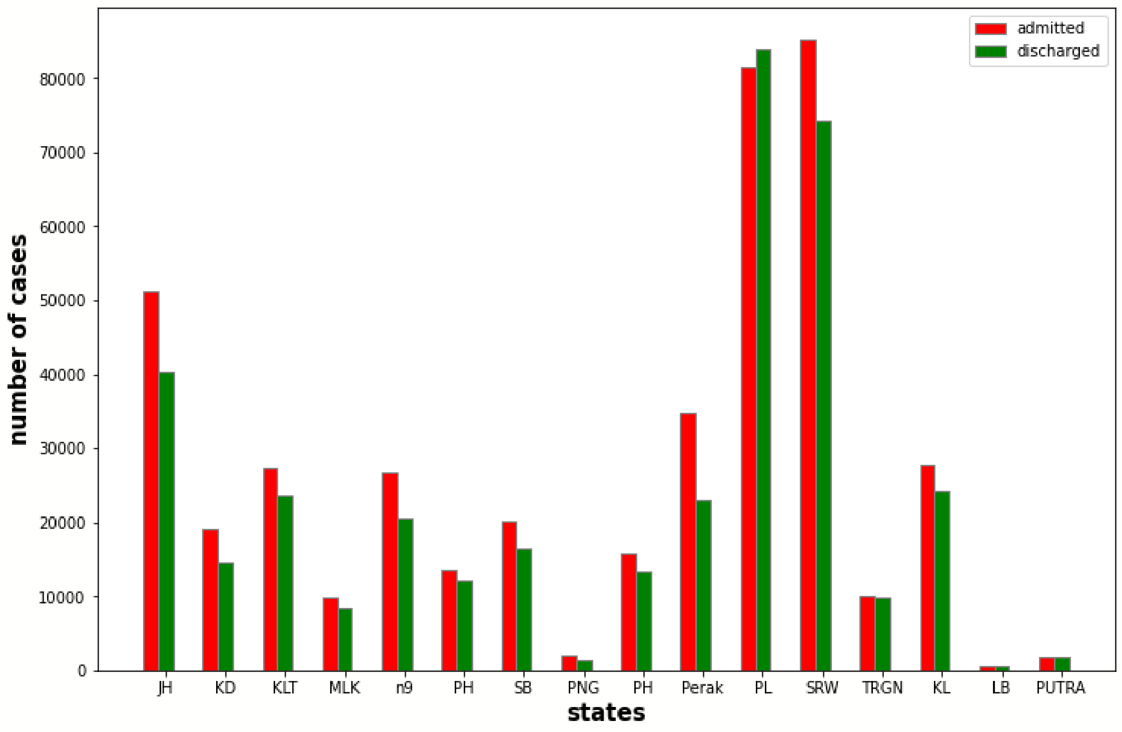
# Plot another barplot to compare among admitted cases and discharged cases of covid-19 in all states before taking static population data into account.

import numpy as np
import matplotlib.pyplot as plt

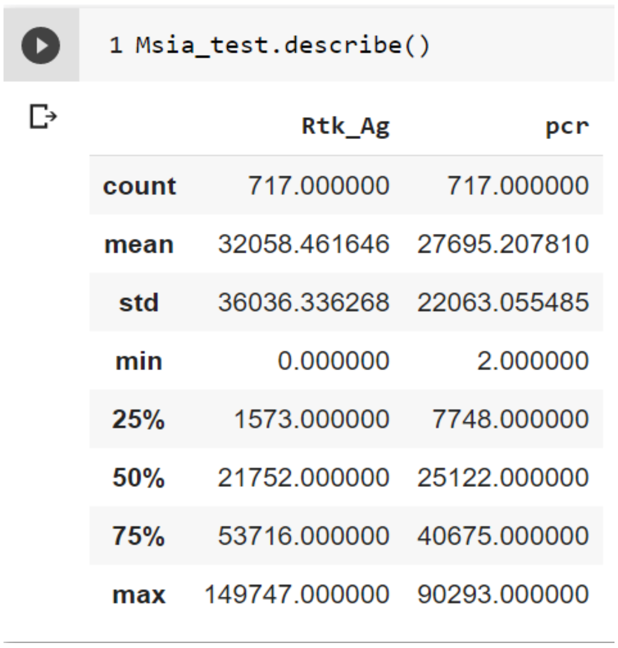
# set width of bar

barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

```
# set height of bar

admitted_covid=
[51286,19153,27370,9747,26681,13491,20094,2033,15723,34822,81586,85294,10118,27749
,497,1865]

discharged_covid=
[40346,14594,23682,8488,20638,12127,16372,1352,13401,23040,83932,74372,9821,24181,
502,1781]

# Set position of bar on X axis

br1 = np.arange(len(admitted_covid))
br2 = [x + barWidth for x in br1]

# Make the plot

plt.bar(br1, admitted_covid, color ='r', width = barWidth,
        edgecolor ='grey', label ='admitted')
plt.bar(br2, discharged_covid, color ='g', width = barWidth,
        edgecolor ='grey', label ='discharged')

# Adding Xticks

plt.xlabel('states', fontweight ='bold', fontsize = 15)
plt.ylabel('number of cases', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(admitted_covid))],
          ['JH', 'KD', 'KLT', 'MLK', 'n9', 'PH', 'SB', 'PNG', 'PH', 'Perak', 'PL', 'SRW', 'TRGN', 'KL',
'LB','PUTRA'])
plt.legend( )
plt.show( )
```


admitted
discharged
number of cases
states
0
10000
20000
30000
40000
50000
60000
70000
80000
JH
KD
KLT
MLK
n9
PH
SB
PNG
PH
Perak
PL
SRW
TRGN
KL
LB
PUTRA

Dataset 6 & 7: Malaysia covid 19- test



```
1 Msia_test.describe()
```

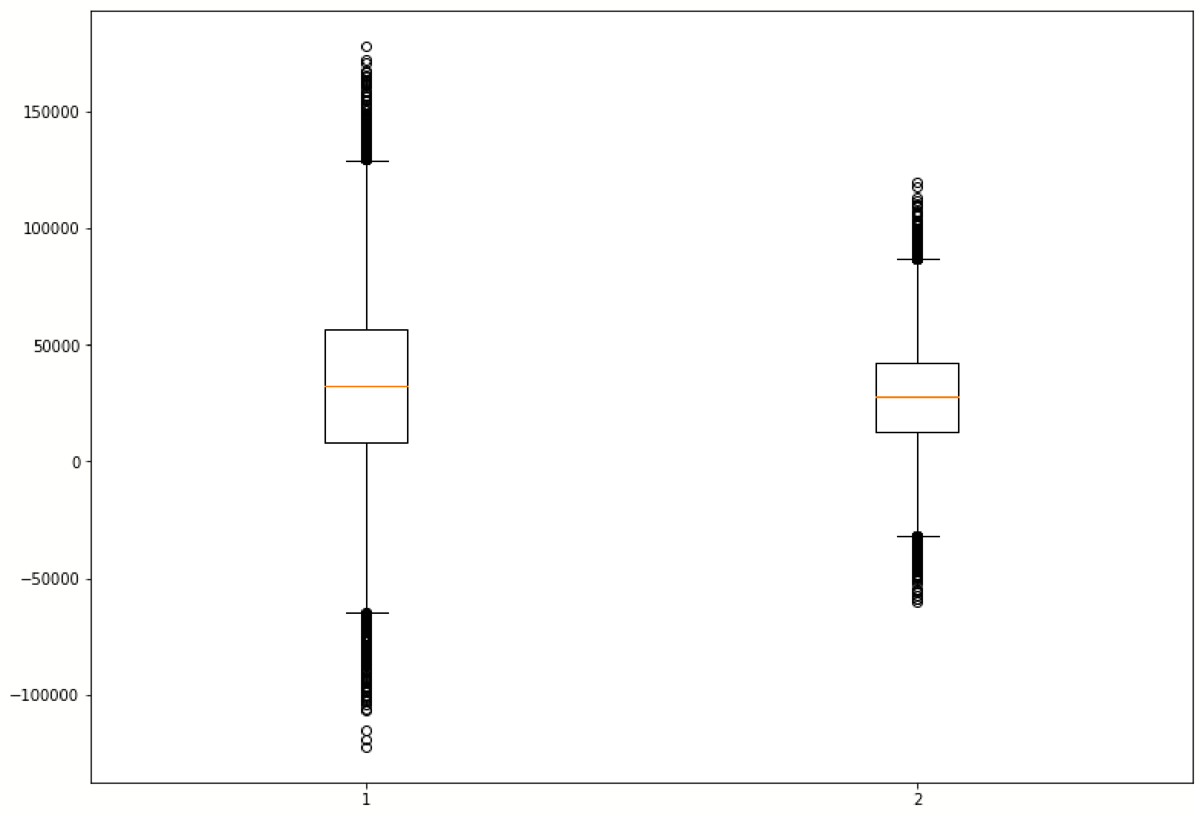
```
             Rtk_Ag            pcr
count     717.000000     717.000000
mean    32058.461646   27695.207810
std     36036.336268   22063.055485
min         0.000000       2.000000
25%      1573.000000    7748.000000
50%     21752.000000   25122.000000
75%     53716.000000   40675.000000
max    149747.000000   90293.000000
```

# There are 2 tests are performed to test the existence of covid-19 infection,

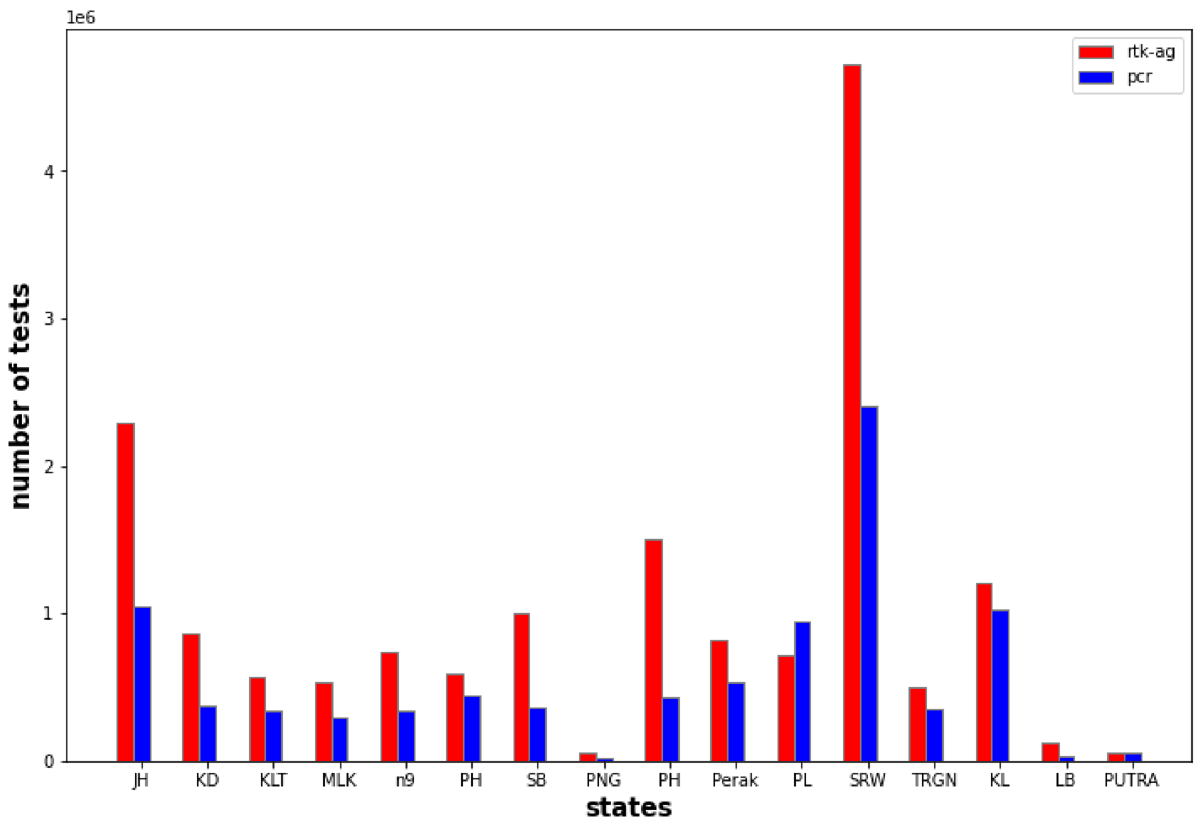
# Which test is more commonly used in Malaysia? rtk-ag or pcr?

# Plot a boxplot to do a comparison on both tests:

Boxplot can also be used to detect outliers. It demonstrates whether or not the data is symmetrical, how consistently the data is grouped, and if and how the data is skewed. Median or 50th Percentile is the middle value of the dataset[9]. First quartile or 25th Percentile is the middle number between the smallest number (not the "minimum") and the median of the dataset[9]. Third quartile or 75th Percentile is the middle value between the median and the highest value (not the "maximum") of the dataset[9]. InterQuartile Range (IQR) is defined as the 25th to the 75th percentile[9]. IQR indicated how spread the middle values are[9]. **Outliers** is an observation point that is distant from other observations[9]. In the case below, the outlier is detected at >170000 and < -60000 number of tests with rtk-ag test.



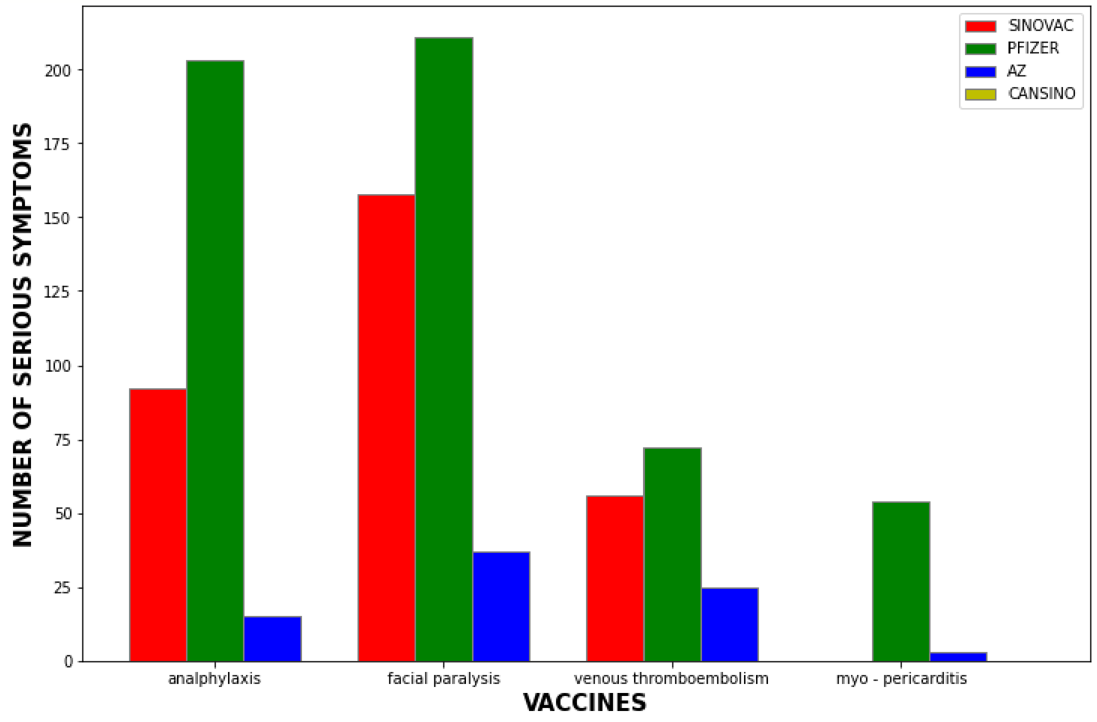
# plot a barplot to compare the antigen rapid test "rtk-ag" and "pcr test" in all states of Malaysia - refer to jupyter notebook for detail codes



Dataset 8: Serious Symptoms caused by Vaccines

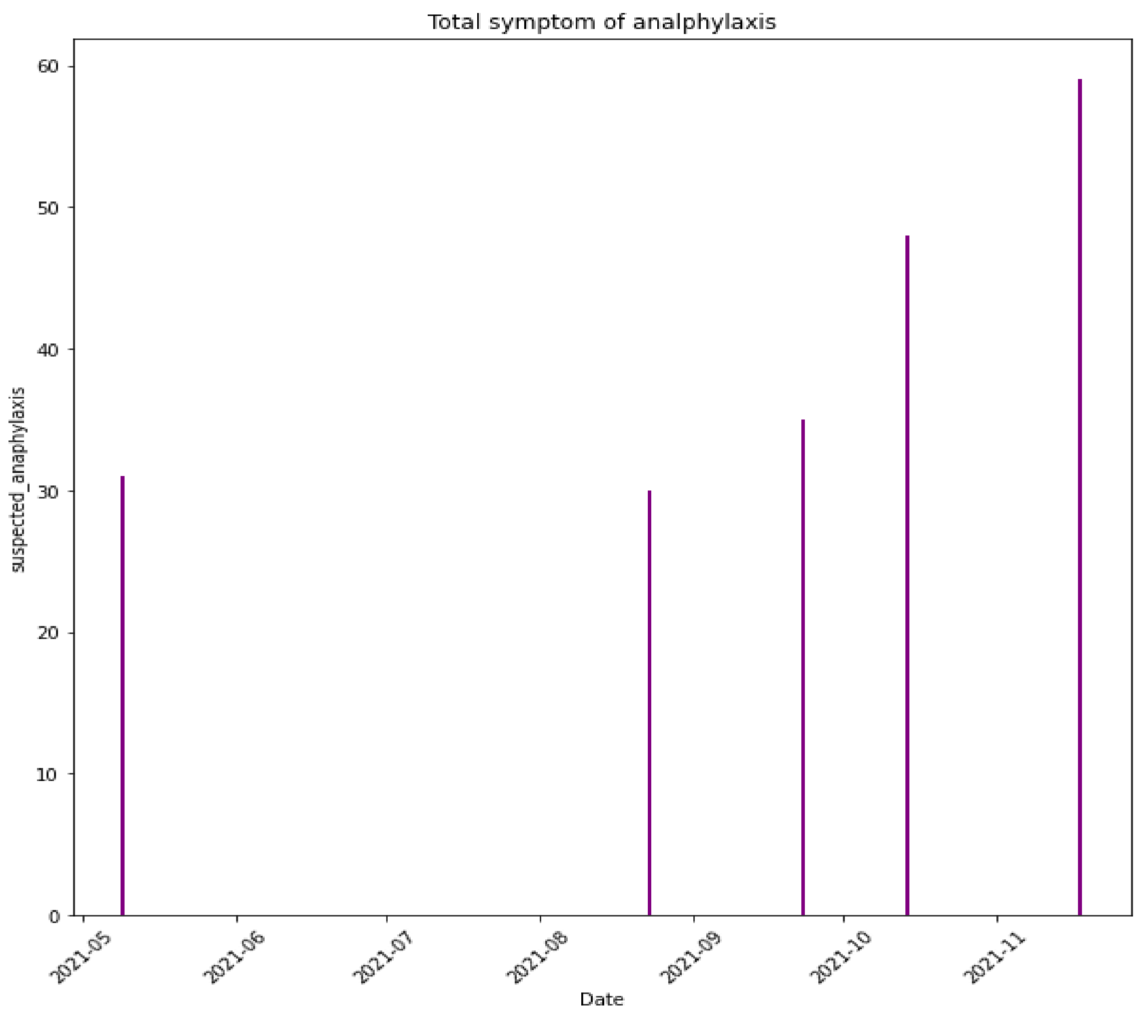
```
import pandas as pd
serious_symptom=
pd.read_csv('https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/vaccinat
ion/aefi_serious.csv')
serious_symptom

# Plot a barplot to compare the serious symptoms among vaccines sinovac, pfizer,
astrazeneca and cansino.
import numpy as np
import matplotlib.pyplot as plt

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

# set height of bar
SINOVAC = [92,158,56,0]
PFIZER = [203,211,72,54]
AZ = [15,37,25,3]
CANSINO = [0,0,0,0]

# Set position of bar on X axis
br1 = np.arange(len(SINOVAC))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
br4 = [x + barWidth for x in br3]

# Make the plot
plt.bar(br1, SINOVAC, color ='r', width = barWidth,
        edgecolor ='grey', label ='SINOVAC')
plt.bar(br2, PFIZER, color ='g', width = barWidth,
        edgecolor ='grey', label ='PFIZER')
plt.bar(br3, AZ, color ='b', width = barWidth,
        edgecolor ='grey', label ='AZ')
plt.bar(br4, CANSINO, color ='y', width = barWidth,
        edgecolor ='grey', label ='CANSINO')
```

```
# Adding Xticks
plt.xlabel('VACCINES', fontweight ='bold', fontsize = 15)
plt.ylabel('NUMBER OF SERIOUS SYMPTOMS', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(SINOVAC))],
        ['analphylaxis', 'facial paralysis', 'venous thromboembolism', 'myo - pericarditis'])
plt.legend( )
plt.show( )
```



# Find the month that reported with highest number of serious symptoms - suspected anaphylaxis

```
# Import necessary packages
import os
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Create figure and plot space
fig, ax = plt.subplots(figsize=(10, 10))

# Add x-axis and y-axis
ax.bar(serious_symptom.index.values,
        serious_symptom['suspected_anaphylaxis'],
        color='purple')

# Set title and labels for axes
ax.set(xlabel="Date",
       ylabel="suspected_anaphylaxis",
       title="Total symptom of anaphylaxis")

# Rotate tick marks on x-axis
plt.setp(ax.get_xticklabels(), rotation=45)

plt.show()
```



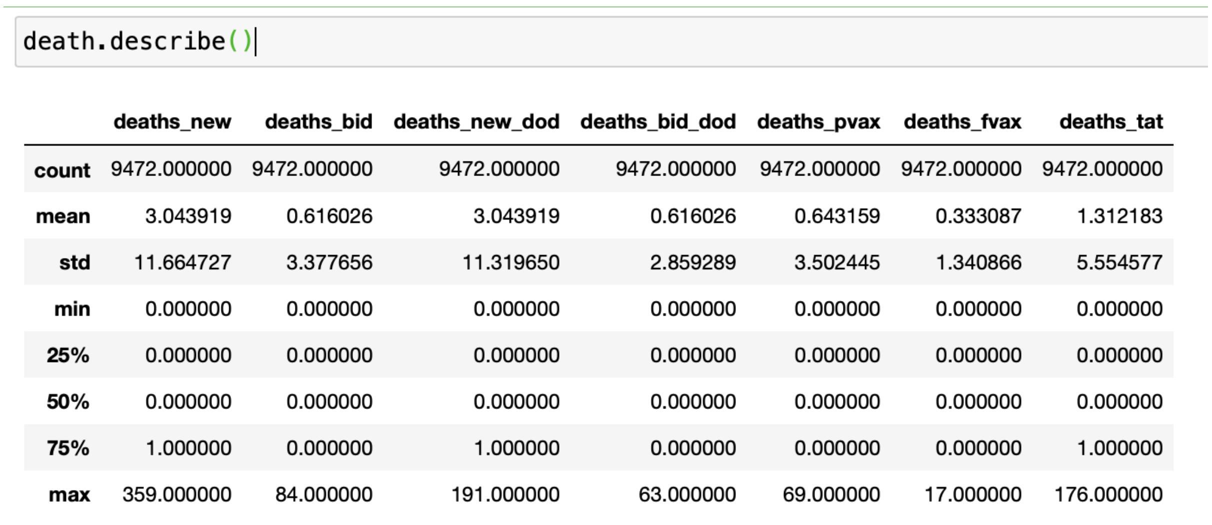
Dataset 9: deaths_state

Link to the dataset:
https://raw.githubusercontent.com/hhanis/covid19-public/main/epidemic/deaths_state.csv

In this dataset, there are no NaN values.

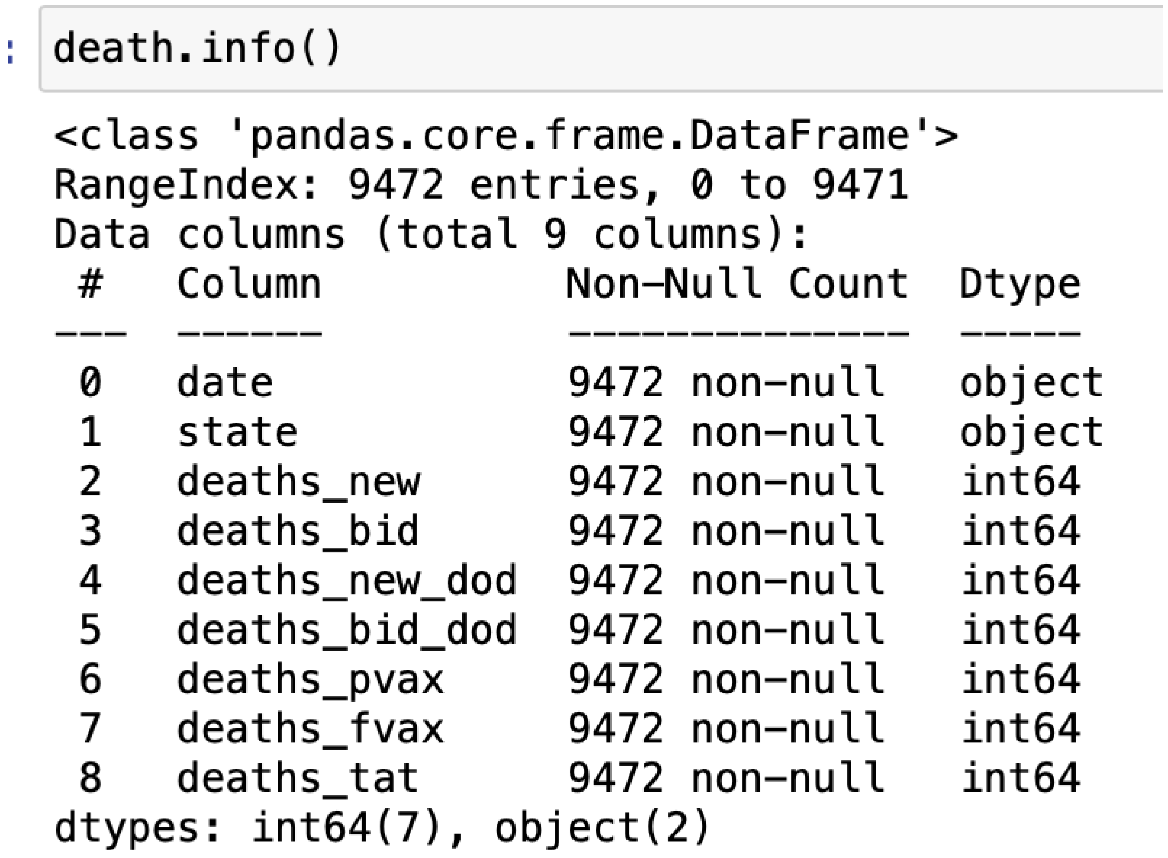
#To see the overview of this dataset

death.describe()



```
death.describe()
```

	deaths_new	deaths_bid	deaths_new_dod	deaths_bid_dod	deaths_pvax	deaths_fvax	deaths_tat
count	9472.000000	9472.000000	9472.000000	9472.000000	9472.000000	9472.000000	9472.000000
mean	3.043919	0.616026	3.043919	0.616026	0.643159	0.333087	1.312183
std	11.664727	3.377656	11.319650	2.859289	3.502445	1.340866	5.554577
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000
max	359.000000	84.000000	191.000000	63.000000	69.000000	17.000000	176.000000



```
death.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9472 entries, 0 to 9471
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   date            9472 non-null   object
 1   state           9472 non-null   object
 2   deaths_new      9472 non-null   int64 
 3   deaths_bid      9472 non-null   int64 
 4   deaths_new_dod  9472 non-null   int64 
 5   deaths_bid_dod  9472 non-null   int64 
 6   deaths_pvax     9472 non-null   int64 
 7   deaths_fvax     9472 non-null   int64 
 8   deaths_tat      9472 non-null   int64 
dtypes: int64(7), object(2)
```

Dataset 10: ICU

Link to dataset:
https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/icu.csv

#To see the overview of this dataset

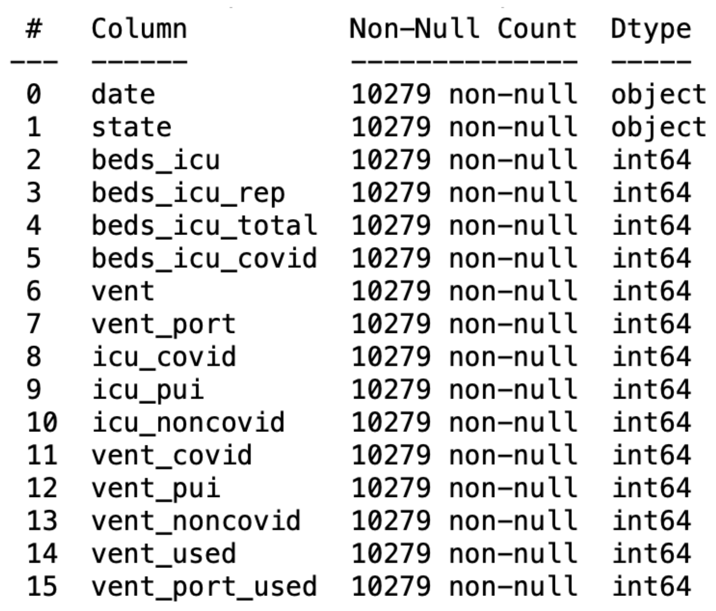
```
import pandas as pd
import numpy as np
import seaborn as sns
```

#to access the dataset

```
death =
pd.read_csv('https://raw.githubusercontent.com/hhanis/covid19-public/main/epidemic/deaths_state.csv')
```

#after accessing the dataset, we identify null values

```
death.isna().sum()
```



```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            10279 non-null  object
 1   state           10279 non-null  object
 2   beds_icu        10279 non-null  int64
 3   beds_icu_rep    10279 non-null  int64
 4   beds_icu_total  10279 non-null  int64
 5   beds_icu_covid  10279 non-null  int64
 6   vent            10279 non-null  int64
 7   vent_port       10279 non-null  int64
 8   icu_covid       10279 non-null  int64
 9   icu_pui         10279 non-null  int64
 10  icu_noncovid    10279 non-null  int64
 11  vent_covid      10279 non-null  int64
 12  vent_pui        10279 non-null  int64
 13  vent_noncovid   10279 non-null  int64
 14  vent_used       10279 non-null  int64
 15  vent_port_used  10279 non-null  int64
```

#No NaN values in this dataset

Dataset 11: Combination of deaths_new, icu and cases_state dataset

#The objective of this combination was to see the correlation between new cases, icu admissions and the number of death

#loading the new cases in all states, also checking the NaN values

```
new =
pd.read_csv("https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_state.csv")
```

```
new.isna().sum()
```

```
date                 0
state                0
cases_new            0
cases_import         0
cases_recovered      0
cases_active         0
cases_cluster        0
cases_unvax          0
cases_pvax           0
cases_fvax           0
cases_boost          0
cases_child          0
cases_adolescent     0
cases_adult          0
cases_elderly        0
cases_0_4            0
cases_5_11           0
cases_12_17          0
cases_18_29          0
cases_30_39          0
cases_40_49          0
cases_50_59          0
cases_60_69          0
cases_70_79          0
cases_80             0
dtype: int64
```

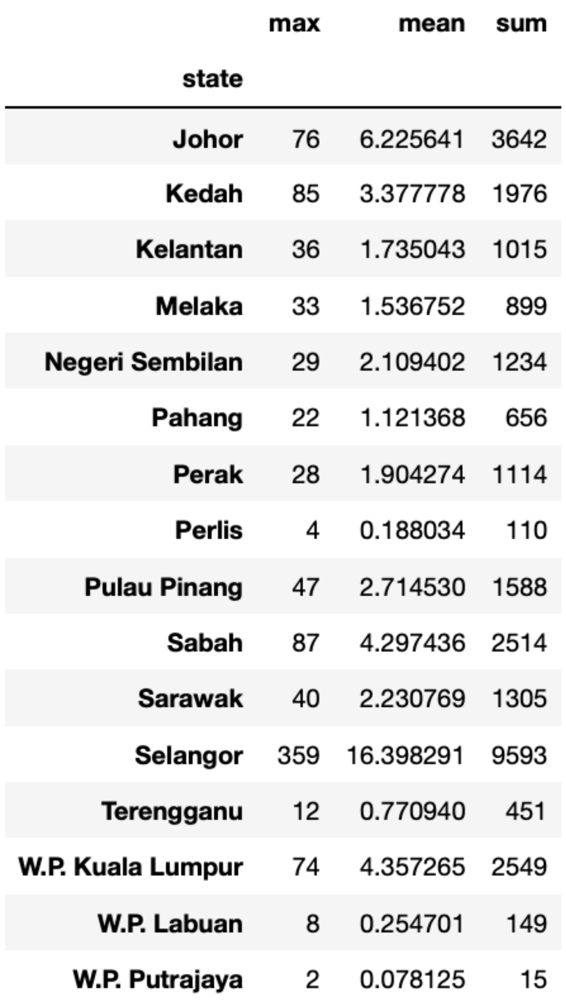
#Dataframe 1 = combining three dataset

```
from functools import reduce

dfs = [icu, death, new]

df = reduce(lambda left,right: pd.merge(left,right,on=['date','state']), dfs)

df.isna().sum()
```

#no NaN values in this dataset

#What is the maximum number of deaths in one day in each state?
#what is the mean number of death in each state
#what is total number of new deaths in each state

```
df.groupby('state').deaths_new.agg(['max','mean','sum'])
```

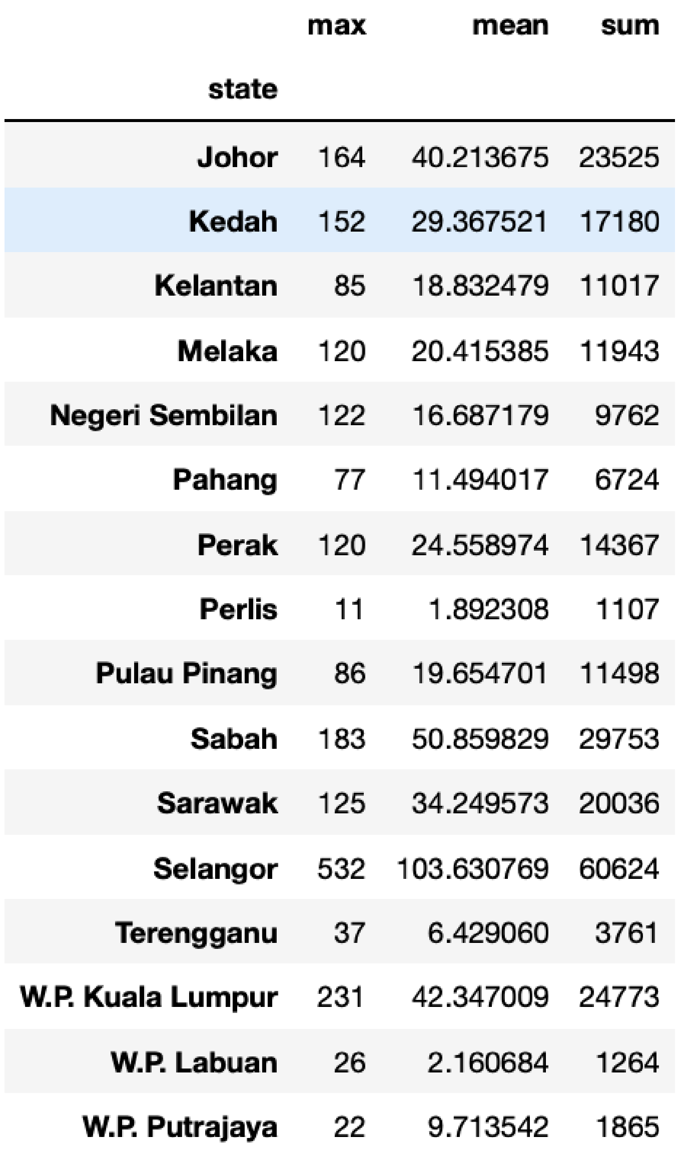


	max	mean	sum
state			
Johor	76	6.225641	3642
Kedah	85	3.377778	1976
Kelantan	36	1.735043	1015
Melaka	33	1.536752	899
Negeri Sembilan	29	2.109402	1234
Pahang	22	1.121368	656
Perak	28	1.904274	1114
Perlis	4	0.188034	110
Pulau Pinang	47	2.714530	1588
Sabah	87	4.297436	2514
Sarawak	40	2.230769	1305
Selangor	359	16.398291	9593
Terengganu	12	0.770940	451
W.P. Kuala Lumpur	74	4.357265	2549
W.P. Labuan	8	0.254701	149
W.P. Putrajaya	2	0.078125	15

#This table shows that the highest number of deaths in a day occurred in Selangor (359 cases). The mean for death in Selangor is 16.39

#show the summary of icu admission in each state
#What is the maximum number of ICU admission in one day in each state?
#What is the mean number of ICU admission in each state?
#What is the total number of ICU admission in each state?

df.groupby('state').icu_covid.agg(['max','mean','sum'])

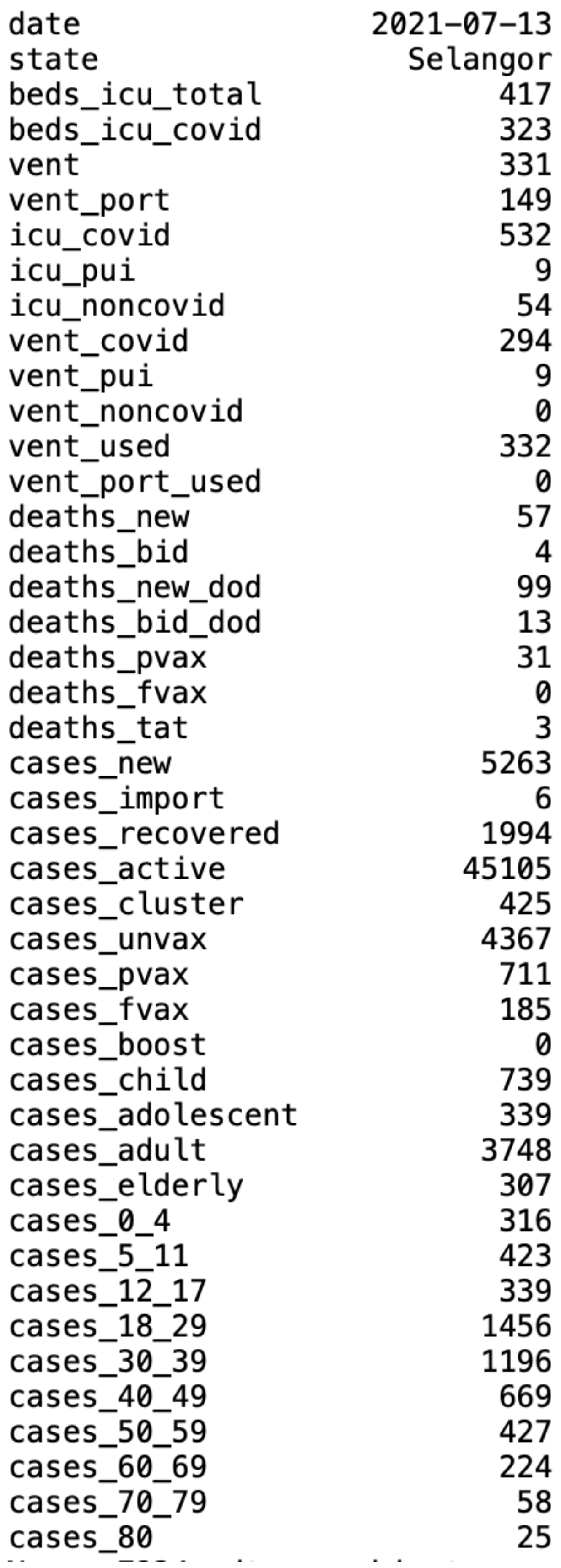


	max	mean	sum
state			
Johor	164	40.213675	23525
Kedah	152	29.367521	17180
Kelantan	85	18.832479	11017
Melaka	120	20.415385	11943
Negeri Sembilan	122	16.687179	9762
Pahang	77	11.494017	6724
Perak	120	24.558974	14367
Perlis	11	1.892308	1107
Pulau Pinang	86	19.654701	11498
Sabah	183	50.859829	29753
Sarawak	125	34.249573	20036
Selangor	532	103.630769	60624
Terengganu	37	6.429060	3761
W.P. Kuala Lumpur	231	42.347009	24773
W.P. Labuan	26	2.160684	1264
W.P. Putrajaya	22	9.713542	1865

#This table shows that the highest number of ICU admission in a day occurred in Selangor (532 cases). The mean for ICU admission in Selangor was 103.63

#when and where was the highest icu admission in a day?

df.loc[df['icu_covid'].idxmax()]

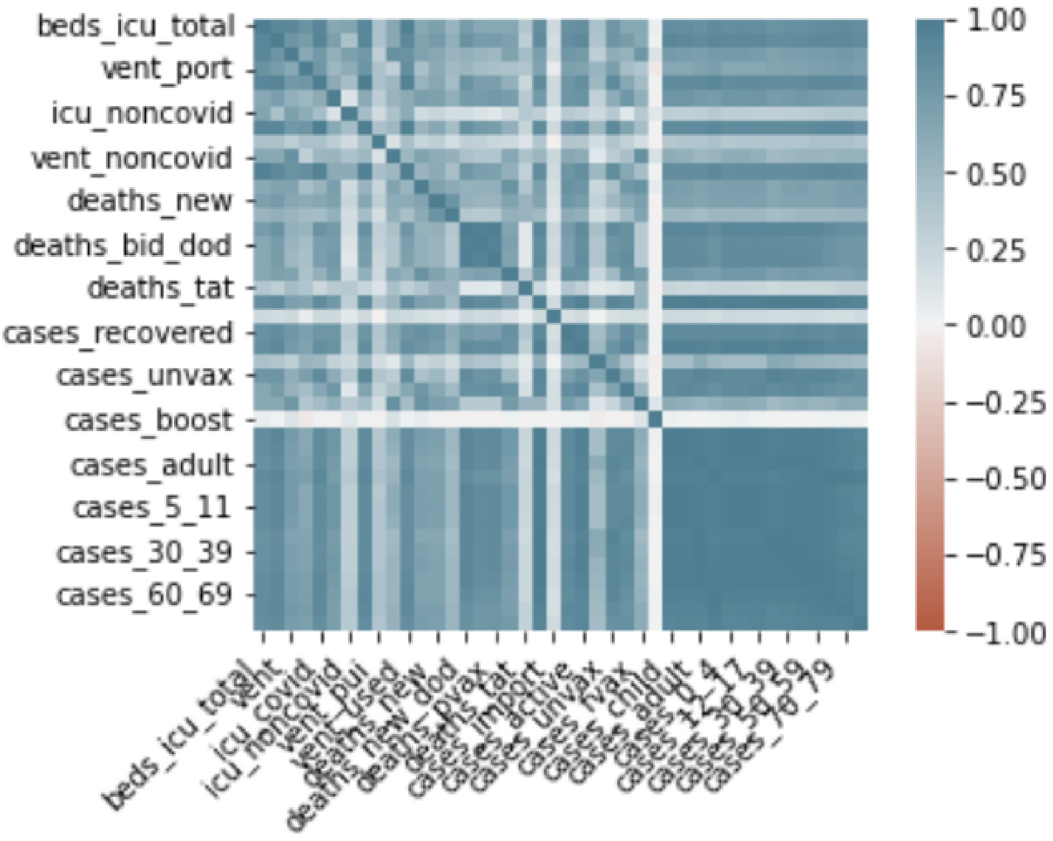


```
date                 2021-07-13
state                  Selangor
beds_icu_total              417
beds_icu_covid              323
vent                        331
vent_port                   149
icu_covid                   532
icu_pui                       9
icu_noncovid                 54
vent_covid                  294
vent_pui                      9
vent_noncovid                 0
vent_used                   332
vent_port_used                0
deaths_new                   57
deaths_bid                    4
deaths_new_dod               99
deaths_bid_dod               13
deaths_pvax                  31
deaths_fvax                   0
deaths_tat                    3
cases_new                  5263
cases_import                  6
cases_recovered            1994
cases_active              45105
cases_cluster               425
cases_unvax                4367
cases_pvax                  711
cases_fvax                  185
cases_boost                   0
cases_child                 739
cases_adolescent            339
cases_adult                3748
cases_elderly               307
cases_0_4                   316
cases_5_11                  423
cases_12_17                 339
cases_18_29                1456
cases_30_39                1196
cases_40_49                 669
cases_50_59                 427
cases_60_69                 224
cases_70_79                  58
cases_80                     25
```

#Since Selangor has the highest cases in Malaysia, let’s do a correlation between variables in Selangor

#we will choose only few attributes: date, number of icu bed, number of patients in icu, number of deaths, number of new case
#To get better view in Selangor, we have to take out ‘Selangor’ datas based on the ‘state’ variable and create independent dataframe

```
selangor = df.loc[df['state']=='Selangor']
sel_new=selangor.iloc[:,[0,2,3,4,5,6,12,13,14,21]]
sel_new.corr().astype(float)

#selangor heatmap
sel = sns.heatmap(
    corr_selangor,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20,220, n=200),
    square=True
)

sel.set_xticklabels(
    sel.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);

#correlation for selangor
corr_selangor = selangor.corr().astype(float)

#selangor heatmap
sel = sns.heatmap(
    corr_selangor,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20,220, n=200),
    square=True
)
sel.set_xticklabels(
    sel.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```



#For more efficient data analytics, we only choose few attributes to be analyzed: date, number of icu bed, number of patients in icu, number of deaths, number of new case

#the numbers in the square bracket in the code below indicates the location of the attributes mentioned above

```
sel_new=selangor.iloc[:,[0,2,3,4,5,6,12,13,14,21]]
sel_new
```

	date	beds_icu_total	beds_icu_covid	vent	vent_port	icu_covid	vent_used	vent_port_used	deaths_new	cases_new
11	2020-03-24	104	42	56	7	23	56	0	1	27
26	2020-03-25	104	42	56	7	19	56	7	0	53
41	2020-03-26	104	42	56	7	24	56	7	0	77
56	2020-03-27	104	42	56	7	25	56	0	1	36
71	2020-03-28	104	42	56	7	31	33	0	1	34
...	...	...	...	...	...	...	...	...	...	...
8898	2021-10-25	245	158	480	17	102	128	13	4	756
8914	2021-10-26	245	158	480	17	90	135	8	9	929
8930	2021-10-27	245	158	480	17	89	118	23	1	1222
8946	2021-10-28	245	158	480	17	90	119	26	1	1431
8962	2021-10-29	245	158	480	17	90	133	17	8	1328

585 rows × 10 columns

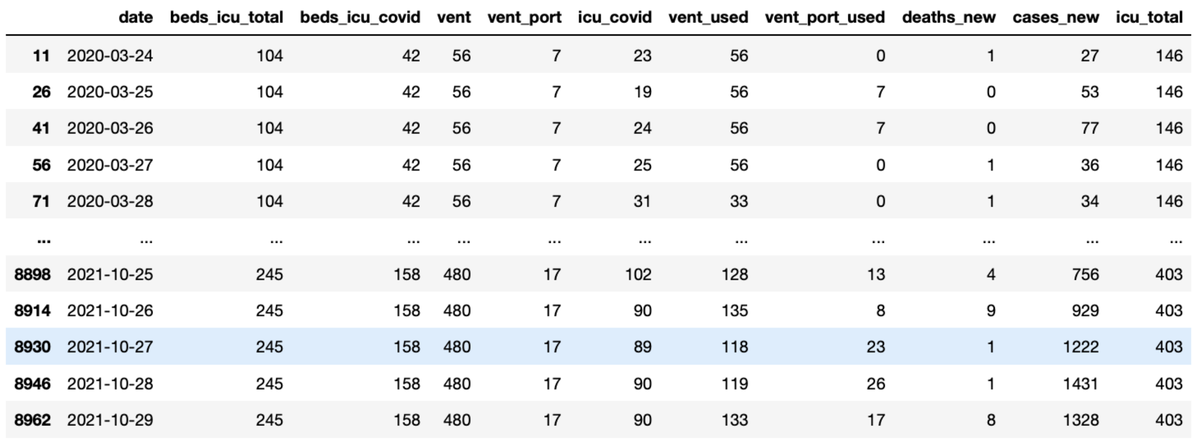
#To see the situation on the ground more clearly, we can combine the number of ICU beds (beds_icu_total + beds_icu_covid) to get the total number of ICU beds in Selangor. We also notice the number of ICU beds are increasing in trend, possibly to accommodate the increased number of cases in Selangor.

To add values from two different columns, we have to add 1 more column to put the new calculated value

#create new column

```
sel_new["icu_total"]=sel_new["beds_icu_total"]+sel_new["beds_icu_covid"]
```

#this will add one more column at the end of the table with the new calculated values



	date	beds_icu_total	beds_icu_covid	vent	vent_port	icu_covid	vent_used	vent_port_used	deaths_new	cases_new	icu_total
11	2020-03-24	104	42	56	7	23	56	0	1	27	146
26	2020-03-25	104	42	56	7	19	56	7	0	53	146
41	2020-03-26	104	42	56	7	24	56	7	0	77	146
56	2020-03-27	104	42	56	7	25	56	0	1	36	146
71	2020-03-28	104	42	56	7	31	33	0	1	34	146
...	...	...	...	...	...	...	...	...	...	...	...
8898	2021-10-25	245	158	480	17	102	128	13	4	756	403
8914	2021-10-26	245	158	480	17	90	135	8	9	929	403
8930	2021-10-27	245	158	480	17	89	118	23	1	1222	403
8946	2021-10-28	245	158	480	17	90	119	26	1	1431	403
8962	2021-10-29	245	158	480	17	90	133	17	8	1328	403

In addition to the added column for total number of ICU beds, we also calculated the percentage of ICU bed usage to get a better view of the situation in Selangor. Line graph is plotted to show the trend of percentage ICU bed usage and the number of death cases in Selangor.

#according to the README file, the total number of ICU bed is the sum of 'beds_icu_total' and 'beds_icu_covid'
#Hence, we need to create a new column to put in value for the total number of ICU beds.

```
sel_new["icu_total"]=sel_new["beds_icu_total"]+sel_new["beds_icu_covid"]
sel_new
```

#We want to see the percentage of ICU usage throughout the pandemic. another column is added to put in the percentage value

```
sel_new["icu_perc"]=sel_new["icu_covid"]/sel_new["icu_total"]*100
```

#since already taken out selangor from the dataset, the date becomes unique for each data. However, first we have to change the datatype of 'date' into date format. then, we can convert the date to 'index'

#changing the datatype to date format

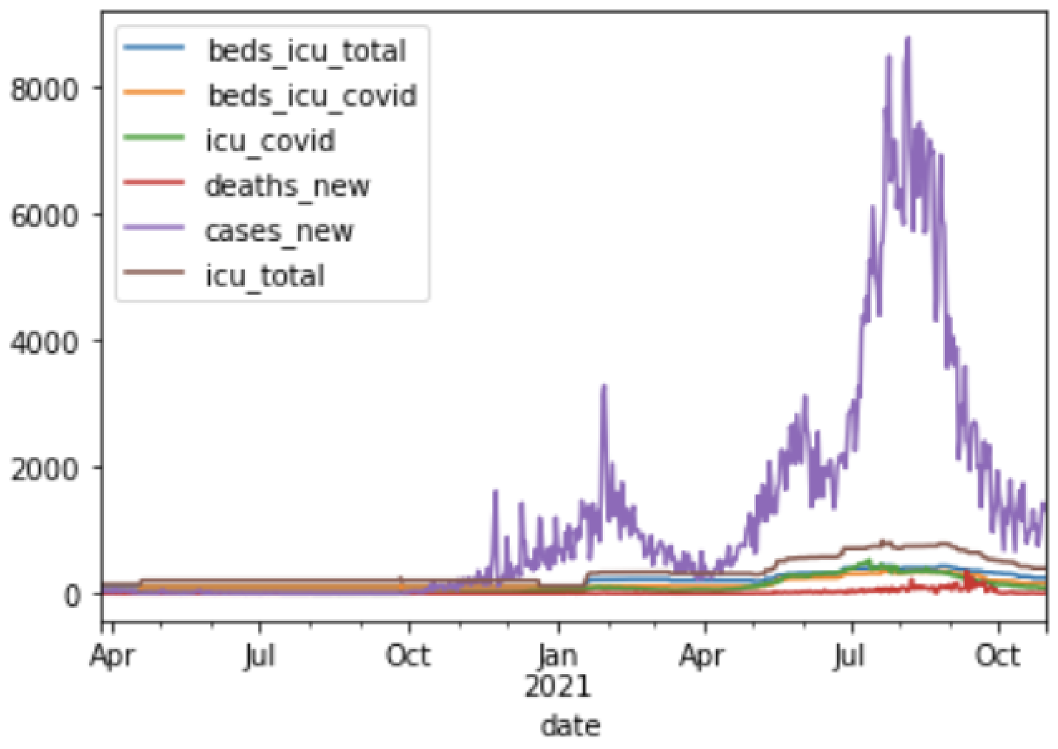
```
sel_new['date'] = pd.to_datetime(sel_new['date'], format='%Y-%m-%d')
```

#since dates all are unique for each data, we convert the data to become index

```
sel_new.set_index('date', inplace=True)
```

	beds_icu_total	beds_icu_covid	vent	vent_port	icu_covid	vent_used	vent_port_used	deaths_new	cases_new	icu_total	icu_perc
date											
2020-03-24	104	42	56	7	23	56	0	1	27	146	15.753425
2020-03-25	104	42	56	7	19	56	7	0	53	146	13.013699
2020-03-26	104	42	56	7	24	56	7	0	77	146	16.438356
2020-03-27	104	42	56	7	25	56	0	1	36	146	17.123288
2020-03-28	104	42	56	7	31	33	0	1	34	146	21.232877
...	...	...	...	...	...	...	...	...	...	...	...
2021-10-25	245	158	480	17	102	128	13	4	756	403	25.310174
2021-10-26	245	158	480	17	90	135	8	9	929	403	22.332506
2021-10-27	245	158	480	17	89	118	23	1	1222	403	22.084367
2021-10-28	245	158	480	17	90	119	26	1	1431	403	22.332506
2021-10-29	245	158	480	17	90	133	17	8	1328	403	22.332506

```
sel_new.plot()
```



The graph shows the peak number of new cases was in between July-September 2021. During this period, ICU admission was also at its peak in the pandemic.

Summary

Dataset 1: Covid-19 Age Group

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_malaysia.csv'

Dataset 2: Malaysia cases

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_malaysia.csv'

Dataset 3: Malaysia's States Covid-19 cases

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/cases_state.csv'

Dataset 4: Covid-19 Death Age group

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/deaths_age.csv'

Dataset 5: hospital covid-19 cases

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/hospital.csv'

Dataset 6: rtk-ag test versus pcr test in Malaysia

https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/tests_malaysia.csv

Dataset 7: rtk-ag test versus pcr test at the state level

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/tests_state.csv'

Dataset 8: serious symptoms caused by different vaccines

'https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/vaccination/aefi_serious.csv'

1. From the Dataset 2 Malaysia Cases above, we found that the recovered covid-19 cases are comparable or slightly lesser than new covid cases.
2. The unvaccinated cases are higher than partially vaccinated and fully vaccinated cases. The lowest reported covid-19 cases are booster cases.

3. In the aspect of population, the highest reported covid-19 cases fall in the category of adult cases, followed by children’s cases. The lowest number of populations reported with covid-19 cases is the adolescent population.
4. The highest age group reported with covid-19 cases is 18-29, followed by age group of 30-39. The lowest age group that reported with covid-19 cases is 80 and above, followed by the age group of 70-79.
5. The highest number of clusters reported with covid-19 cases is found to be "cluster workplace" followed by "cluster community" while the lowest number of clusters on the other hand is shown to be "cluster import".
6. Selangor is reported to have the highest covid-19 new cases 12.26% among all the states in Malaysia while Perlis is reported to have the lowest covid-19 new cases which is 2.92% from Malaysia population.
7. Within the state level, before dividing by population data, Negeri Sembilan is reported to have the highest partially,fully and booster cases. However, after taking into account the population for each state, Selangor showed to have higher partially, fully and booster cases than Negeri Sembilan.
8. When we are doing analysis for covid-19 specifically for the state of Johor, we found the age group that contributed the highest death cases(4322 cases) is between 60-69 year old and in national state which in Malaysia, the age group of 60-69 year old is also found has the highest death Covid-19 cases (4849 cases).
9. From Dataset 5, we found that Selangor is reported to contribute the highest bed occupancy in Malaysia. The number of patients in hospital is reported to be highest in Negeri Sembilan after analyzing state population data while before comparing with state population data, Selangor is shown to have the highest number of covid individuals in hospital.
10. Based on the barplot observation in Dataset 5, most of the states like Johor, Kedah, Kelantan, Melaka, Negeri Sembilan, Pahang, Perak, Perlis, Penang,Sabah, Selangor and KL have more patients admitted than discharged from hospital. The rate of admission and discharges in hospitals for Terengganu, Labuan and Putrajaya are about the same.Only Sarawak is reported to have higher discharge rate than admission rate.
11. From Dataset 6, we conclude that Malaysia performed more rtk-ag test than pcr test for identifying the existence of covid-19 infection. From Dataset 7, without considering the state population data, Selangor is found to conduct the highest number of tests compared to other states. Most states in Malaysia performed more rtk-ag test than pcr test except Sarawak. Sarawak performed more pcr tests than rtk-ag tests. However, after comparing population data for each state, Labuan is found to conduct the highest number of tests(rtk-ag) followed by Penang.
12. From Dataset 8, the brand of Pfizer is shown to cause more serious symptoms such as anaphylaxis, facial paralysis, venous thromboembolism and myo-pericarditis compared to Sinovac, Astrazeneca and Cansino. Only Pfizer and Astrazeneca are reported to cause serious symptoms of myo-pericarditis.

Dataset 9: data on death number in each state

Link to the dataset:
https://raw.githubusercontent.com/hhanis/covid19-public/main/epidemic/deaths_state.csv

In this dataset, there are no identified NaN values during the preparation process of the data.

The data has 9 columns and 9472 rows as of 14/1/2022 with the following attributes:

i. Date
ii. State
iii. Deaths_new: deaths due to COVID-19 based on date reported to the public
iv. Deaths_bid: deaths due to COVID-19 which were brought-in dead based on date reported to public (perfect subset of deaths_new)
v. Deaths_new_dod: deaths due to COVID-19 based on date of death
vi. Deaths_bid_dod: deaths due to COVID-19 which were brought-in dead based on date of death (perfect subset of deaths_new_dod)
vii. Deaths_pvax: number of partially-vaccinated individuals who died due to COVID-19 based on date of death (perfect subset of deaths_new_dod)
viii. Deaths_fvax: number of fully-vaccinated who died due to COVID-19 based on date of death (perfect subset of deaths_new_dod)
ix. Deaths_tat: median days between date of death and date of report for all deaths reported on the day

There are 16 states in the dataset: Johor, Kedah, Kelantan, Melaka, Negeri Sembilan, Pahang, Perak, Perlis, Pulau Pinang, Sabah, Sarawak, Selangor, Terengganu, W.P. Kuala Lumpur, W.P. Labuan, W.P. Putrajaya.

The information on the states were updated consistently on a daily basis.

Dataset 10: data on ICU admission in each state

https://raw.githubusercontent.com/MoH-Malaysia/covid19-public/main/epidemic/icu.csv

This data is a collection of data about the ICU cases among the COVID-19 positive cases in Malaysia.

In this dataset, there are no identified NaN values during preparation process of the data.
There are 16 attributes in the dataset

The data has 16 columns and 10183 rows as of 14/1/2022 with the following attributes:

i. Date
ii. State
iii. Beds_icu: total gazetted ICU beds
iv. Beds_icu_rep: total beds aside from (3) which are temporarily or permanently designated to be under the care of Anaesthesiology & Critical Care departments

v. Beds_icu_total: total critical care beds available (with related medical infrastructure)
vi. Beds_icu_covid: total critical care beds dedicated for COVID-19
vii. Vent: total available ventilators
viii. Vent_port: total available portable ventilators
ix. Icu_covid: total number of individuals under intensive care
x. Icu_pui: total number of individuals who are suspected/probable, COVID-19 positive
xi. Icu_noncovid: total number of individuals who are non-covid
xii. Vent_covid: total number of individuals under intensive care
xiii. Vent_pui: total number of individuals who are suspected/probable, COVID-19 positive
xiv. Vent_noncovid: total number of individuals who are non-covid
xv. Vent_used
xvi. Vent_port_used

Dataset 11: Combination of deaths_new, icu and cases_state dataset

In this dataset, we combined three dataset into one. The combination came from the deaths_new, icu and cases_state dataset. No identified NaN values in this dataset.

Recommendation

The explanations on the attributes in github.com/Moh Malaysia/Covid-19 repository are not completed. The attributes in each column of the csv file should be clarified for example in the age group category, **abs**_0_4, **perc**_0_4,**capita**_0_4 in the readme file, the definition of abs, perc and capita should be included for better understanding of the data. In the future, other than the parameters of the number of covid cases, vaccinations and hospital, treatment/medication used during hospitalization can be added for investigation. Focus can be placed more on the outcome of covid-19 cases, recovered cases/death with and without, coexisting complications such as diabetes/hypertension, as this data provide more clinical insight. In view of the data collected being quantitative data, a predictive model can be considered to predict the incidence in the future. Although the MOH-Malaysia dataset is open to the public at the moment, however its existence is not widely known. The ministry also should be transparent on how they manage the data to foresee the COVID-19 situation in Malaysia.

Conclusion

In this group project, we have provided a simple descriptive statistical analysis of the Coronavirus (Covid-19) outbreak in Malaysia. Using data of the daily and cumulative cases in Malaysia for approximately 2 years from January Year 2020 to current Year 2022, we have analyzed the pattern of the disease. Despite the simplicity of our results, we believe that these simple analysis provide an interesting insight into the statistics of the Covid-19 outbreak in Malaysia. The result will be useful in contributing to health guidelines or interventions in the hospitals and the community.

We used python programming for analyzing the Covid-19 Data in this project. This programming language is very powerful and flexible. Descriptive statistics is a study of analysis to describe,show or summarize the findings in a meaningful way[10]. It involves the calculation of parameters such as the measure of the mean, median, standard deviation, percentiles and also the construction of tables and graphs[10].

Data preparation and data cleansing are important for the success of machine learning models[4]. However, this process is time-intensive and sensitive that is full of challenges[4]. Therefore, self-service data preparation tools have been designed to increase the productivity of data scientists and further improve the performance of machine learning models[4]. Such tools allow practitioners to work within an easy-to-use visual application for cleaning, preparing, and deploying data[4].

This dataset is complete in terms of duration, i.e it was started early, during the early phase of the pandemic. This concept could be implemented in other situations, such as in dengue and other diseases as well.

References

1.https://www.actian.com/company/blog/the-six-steps-essential-for-data-preparation-and-analysis/

2. https://www.altair.com/what-is-data-preparation/

3. https://www.talend.com/resources/what-is-data-preparation/

4. https://www.topbots.com/data-preparation-for-machine-learning/

5. https://www.kdnuggets.com/2019/11/data-cleaning-preprocessing-beginners.html

6.https://www.tutorialspoint.com/python_pandas/python_pandas_descriptive_statistics.htm

7.https://www.geeksforgeeks.org/python-pandas-isnull-and-notnull/

8.https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/

9.https://medium.com/@agarwal.vishal819/outlier-detection-with-boxplots-1b6757fafa21

10.https://towardsdatascience.com/a-quick-guide-on-descriptive-statistics-using-pandas-and-seaborn-2aadc7395f32