Date: 8/3/22 EXP: 1a	STUDY ON NUMPY, SCIPY, STATMODELS AND PANDAS PACKAGES

AIM:

To explore the functions in Numpy , SciPy , Statsmodels and pandas packages and prepare a study with an example for each function.

NUMPY:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python. NumPy arrays are stored at one continuous place in memory unlike lists.

FUNCTIONS:

1. numpy.linspace

numpy.linspace**(*start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0*)** function returns evenly spaced numbers over a specified interval defined by the first two arguments of the function (***start*** and ***stop*** — required arguments). The number of samples generated is specified by the third argument ***num***.

2. numpy.digitize

Numpy.digitize**(*x, bins, right=False*)** function has two arguments: (1) an input array ***x***, and (2) an array of ***bins,*** returning the indices of the bins to which each value in input array belongs.

3. numpy.repeat

Numpy.repeat(***a, repeats, axis=None*)** function repeats the elements of an array. The number of repetitions is specified by the second argument ***repeats*.**

4. numpy.nan

Numpy library includes several constants such as not a number (Nan), infinity (inf) or pi. In computing, not a number is a numeric data type that can be interpreted as a value that is undefined. We can use not a number to represent missing or null values in Pandas.

PANDAS:

Pandas is a data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

FUNCTIONS:

1. Query

We sometimes need to filter a dataframe based on a condition or apply a mask to get certain values. One easy way to filter a dataframe is query function.

2. Insert

When we want to add a new column to a dataframe, it is added at the end by default. However, pandas offers the option to add the new column in any position using **insert** function.

3. Cumsum

Pandas provides an easy-to-use function to calculate cumulative sum which is **cumsum**.

4. Sample

Sample method allows you to select values randomly from a **Series** or **DataFrame**. It is useful when we want to select a random sample from a distribution.

SCIPY:

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely.

FUNCTIONS:

1. cbrt()

This is used to return the cube root of the given number.

2. comb()

It is known as combinations and returns the combination of a given value.

3. exp10()

This method gives the number with raise to 10 power of the given number.

4. exprel()

It is known as the Relative Error Exponential Function. It returns the error value for a given variable. If x is near zero, then exp(x) is near 1.

STATSMODELS:

It is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct.

FUNCTIONS:

1. get_rdataset()

This is used to download any dataset.

2. add_constant()

It is used to add a constant column to import dataset.

3. OLS(y,x).fit()

This is a type of linear least square method for estimating unknown parameter in linear regression.

4.linear_rainbow()

Rainbow test has power against many different from of non-linearity.

RESULT:

Thus, study on Python packages for data science was done.

Date: 8/3/22 EXP: 1b	READ DATA USING INBUILT PACKAGES

AIM:

To use inbuilt packages to read data from text file, Excel and the web.

SOURCE CODE AND OUTPUTS:

Data From Text File:

```
file1 = open("/home/Grade/Downloads/test","r+")

print(file1.read())
```

```
In [1]: file1 = open("/home/student/Downloads/test","r+")
        print(file1.read())

        Cloud python packages
```

Data from Excel Sheet:

```
import pandas as pd
data = pd.read_csv("/home/Grade/Downloads/cov.csv")
print("Total rows: {0}".format(len(data)))
print(list(data))

import pandas as pd
df = pd.read_csv(r"/home/Grade/Downloads/cov.csv")
print(df)

import pandas as pd
df = pd.read_csv(r"/home/Grade/Downloads/cov.csv")
data= pd.DataFrame(df, columns= ['Confirmed','Deaths'])
print(data)
```

```
In [4]: import pandas as pd
        data = pd.read_csv("/home/student/Downloads/cov.csv")
        print("Total rows: {0}".format(len(data)))
        print(list(data))

        Total rows: 187
        ['Country/Region', 'Confirmed', 'Deaths', 'Recovered', 'Active', 'New cases', 'New deaths', 'New recovered', 'Deaths
        / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered', 'Confirmed last week', '1 week change', '1 week % i
        ncrease', 'WHO Region']
```

```
In [9]: import pandas as pd
        df = pd.read_csv(r"/home/student/Downloads/cov.csv")
        print(df)
```

```
          Country/Region  Confirmed  Deaths  Recovered  Active  New cases  \
0            Afghanistan      36263    1269      25198    9796        106
1                Albania       4880     144       2745    1991        117
2                Algeria      27973    1163      18837    7973        616
3                Andorra        907      52        803      52         10
4                 Angola        950      41        242     667         18
..                   ...        ...     ...        ...     ...        ...
182  West Bank and Gaza      10621      78       3752    6791        152
183      Western Sahara         10       1          8       1          0
184               Yemen       1691     483        833     375         10
185              Zambia       4552     140       2815    1597         71
186            Zimbabwe       2704      36        542    2126        192

     New deaths  New recovered  Deaths / 100 Cases  Recovered / 100 Cases  \
0            10             18                3.50                  69.49
1             6             63                2.95                  56.25
2             8            749                4.16                  67.34
3             0              0                5.73                  88.53
4             1              0                4.32                  25.47
..          ...            ...                 ...                    ...
182           2              0                0.73                  35.33
183           0              0               10.00                  80.00
184           4             36               28.56                  49.26
185           1            465                3.08                  61.84
186           2             24                1.33                  20.04
```

```
In [10]: import pandas as pd
         df = pd.read_csv(r"/home/student/Downloads/cov.csv")
         data= pd.DataFrame(df, columns= ['Confirmed','Deaths'])
         print(data)
```

```
     Confirmed  Deaths
0        36263    1269
1         4880     144
2        27973    1163
3          907      52
4          950      41
..         ...     ...
182      10621      78
183         10       1
184       1691     483
185       4552     140
186       2704      36

[187 rows x 2 columns]
```

Data from Web:

```
import requests
from bs4 import BeautifulSoup
r = requests.get("https://www.mitindia.edu/en/")
soup = BeautifulSoup(r.content, 'html.parser')
lines = soup.find_all('p')
for line in lines:
    print(line.text)
```

```
In [7]: import requests
        from bs4 import BeautifulSoup
        r = requests.get("https://www.mitindia.edu/en/")
        soup = BeautifulSoup(r.content, 'html.parser')
        lines = soup.find_all('p')
        for line in lines:
            print(line.text)
```

```
In 1949,Shri.C.Rajam, gave the newly independent India-Madras Institute of Technology, so that MIT could establish t
he strong technical base it needed to take its place in the world. It was the rare genius and daring of its founder
that made MIT offer courses like Aeronautical Engineering, Automobile Engineering, Electronics Engineering and Instr
ument Technology for the first time in our country. Now it also provides technical education in other engineering fi
elds such as Rubber and Plastic Technology & Production Technology. It was merged with Anna University in the year 1
978. Sixty years hence, while it continues to be a pioneer in courses that it gave birth to, it is already renowned
for producing the crème de la crème of the scientific community in more nascent courses such as Computer Science and
Information Technology MIT has produced great scientist like Dr.A.P.J.Abdul Kalam, versatile genius like Sujatha and
many more. The broad-based education, coupled with practice-oriented training in their speciality, has enabled the s
tudents of MIT to handle with skill and success a wide variety of technical problems. The Madras Institute of Techno
logy has developed into an important centre of engineering education and earned an excellent reputation both in Indi
a and abroad.MIT had received many awards which includes an award for the Best Overall Performance, awarded by India
n Society of Technical Education (ISTE) during the year 1999.
Copyright © 2016. All Rights Reserved. Commercial use and distribution of the contents of the website is not allowed
without express and prior written consent of Madras Institute of Technology.No part of this website may be reproduce
d without Prior Notice.
Designed And Maintained by WebTeam MitIndia
```

RESULT:

Thus, data was read from files, excel and the web.

Date: 15/3/22 EXP:2a-i	DESCRIPTIVE ANALYTICS

AIM:

a) To use Iris Dataset and Explore various commands for doing descriptive analytics
b) To use Unique dataset assigned to you explore the commands in descriptive analytics
c) From the COVID dataset perform the following
   i. Find correlation between features and visualize it
   ii. Perform dimensionality reduction
   iii. Detect outliers
   iv. Fill missing data
   v. Data Partitioning
   vi. Perform the following statistical test for the dataset
      - Normality test
      - Correlation test
      - Parametric statistical Hypothesis test
      - Nonparametric statistical Hypothesis test

SOURCE CODE:

a) Descriptive Analysis on Iris Dataset:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

df = pd.read_csv("C:/Users/91812/Desktop/SEM 6/DSCC lab/iris.csv")

df.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

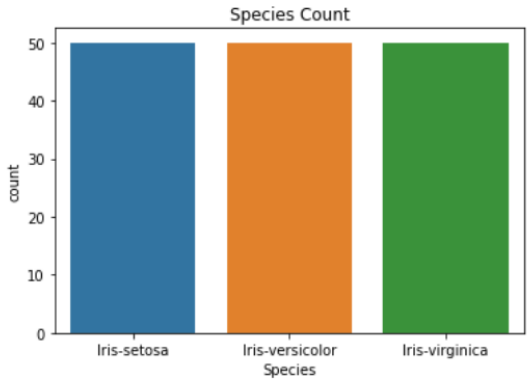
df.describe()

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

plt.title('Species Count')

sns.countplot(df['Species'])

```
<AxesSubplot:title={'center':'Species Count'}, xlabel='Species', ylabel='count'>
```

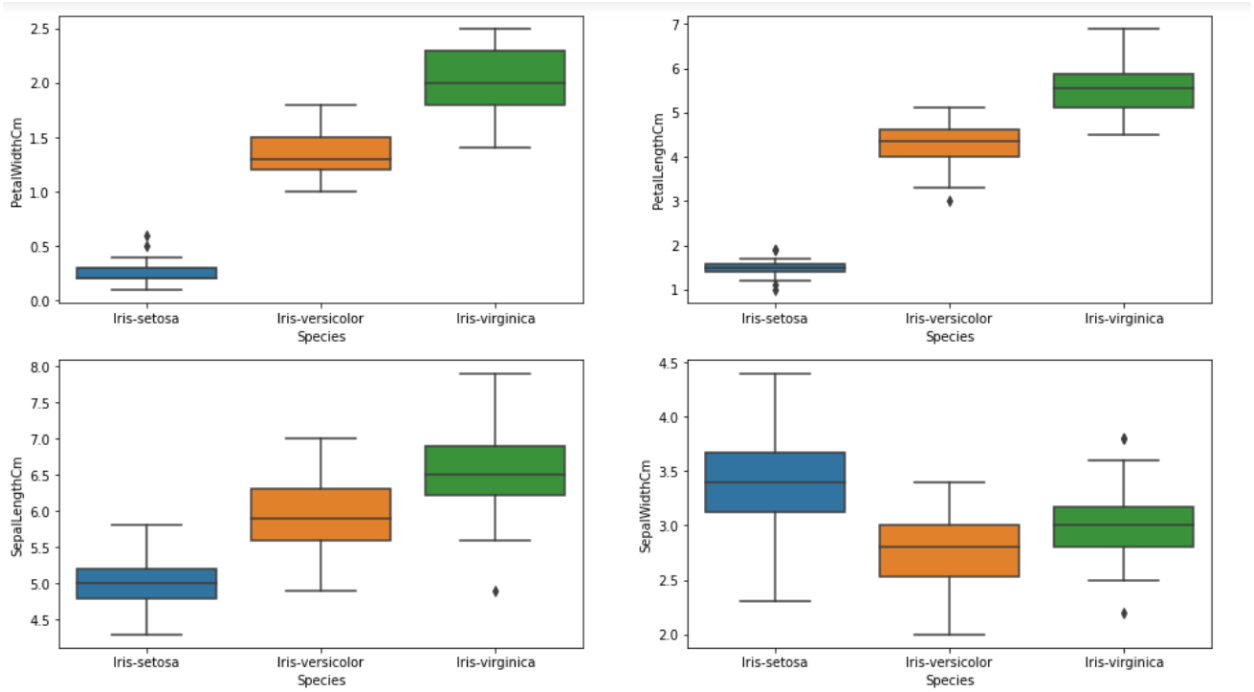


This further visualizes that species are well balanced Each species (Iris virginica, setosa, versicolor) has 50 as it's count

df.groupby('Species').agg(['mean', 'median'])

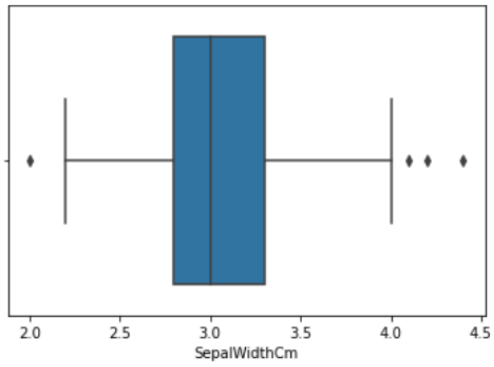
	SepalLengthCm		SepalWidthCm		PetalLengthCm		PetalWidthCm	
	mean	median	mean	median	mean	median	mean	median
Species								
Iris-setosa	5.006	5.0	3.418	3.4	1.464	1.50	0.244	0.2
Iris-versicolor	5.936	5.9	2.770	2.8	4.260	4.35	1.326	1.3
Iris-virginica	6.588	6.5	2.974	3.0	5.552	5.55	2.026	2.0

```
fig, axes = plt.subplots(2, 2, figsize=(16,9))
sns.boxplot( y='PetalWidthCm' ,x= 'Species', data=df, orient='v' , ax=axes[0, 0])
sns.boxplot( y='PetalLengthCm',x= 'Species', data=df, orient='v' , ax=axes[0, 1])
sns.boxplot( y='SepalLengthCm',x= 'Species', data=df, orient='v' , ax=axes[1, 0])
sns.boxplot( y='SepalWidthCm' ,x= 'Species', data=df, orient='v' , ax=axes[1, 1])
plt.show()
```




```
sns.boxplot(x='SepalWidthCm', data=df)
```

```
<AxesSubplot:xlabel='SepalWidthCm'>
```



In the above graph, the values above 4 and below 2 are acting as outliers.

b) Descriptive Analysis on Unique Dataset:

```
import matplotlib.pyplot as plt
import numpy as np
from google.colab import files
uploaded = files.upload()
import io
ds=pd.read_csv(io.BytesIO(uploaded['cov.csv']))
ds.shape
ds.head()
```

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
```

```
[ ] from google.colab import files
    uploaded = files.upload()
    import io
    ds=pd.read_csv(io.BytesIO(uploaded['cov.csv']))
    ds.shape
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving cov.csv to cov (5).csv
(187, 15)
```

```
[ ] ds.head()
```

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase	WHO Region
0	Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69.49	5.04	35526	737	2.07	Eastern Mediterranean
1	Albania	4880	144	2745	1991	117	6	63	2.95	56.25	5.25	4171	709	17.00	Europe
2	Algeria	27973	1163	18837	7973	616	8	749	4.16	67.34	6.17	23691	4282	18.07	Africa
3	Andorra	907	52	803	52	10	0	0	5.73	88.53	6.48	884	23	2.60	Europe
4	Angola	950	41	242	667	18	1	0	4.32	25.47	16.94	749	201	26.84	Africa

```
ds.describe()
```

DESCRIPTIVE ANALYTICS

DESCRIPTIVE ANALYSIS

```
[ ] ds.describe()
```

	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase
count	1.870000e+02	187.000000	1.870000e+02	1.870000e+02	187.000000	187.000000	187.000000	187.000000	187.000000	187.00	1.870000e+02	187.000000	187.000000
mean	8.813094e+04	3497.518717	5.063148e+04	3.400194e+04	1222.957219	28.957219	933.812834	3.019519	64.820535	inf	7.868248e+04	9448.459893	13.606203
std	3.833187e+05	14100.002482	1.901882e+05	2.133262e+05	5710.374790	120.037173	4197.719635	3.454302	26.287694	NaN	3.382737e+05	47491.127684	24.509838
min	1.000000e+01	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	1.000000e+01	-47.000000	-3.840000
25%	1.114000e+03	18.500000	6.265000e+02	1.415000e+02	4.000000	0.000000	0.000000	0.945000	48.770000	1.45	1.051500e+03	49.000000	2.775000
50%	5.059000e+03	108.000000	2.815000e+03	1.600000e+03	49.000000	1.000000	22.000000	2.150000	71.320000	3.62	5.020000e+03	432.000000	6.890000
75%	4.046050e+04	734.000000	2.260600e+04	9.149000e+03	419.500000	6.000000	221.000000	3.875000	86.885000	6.44	3.708050e+04	3172.000000	16.855000
max	4.290259e+06	148011.000000	1.846641e+06	2.816444e+06	56336.000000	1076.000000	33728.000000	28.560000	100.000000	inf	3.834677e+06	455582.000000	226.320000

ds.corr(method='pearson')

CORRELATION AND VISUALIZATION

```
[ ] ds.corr(method='pearson')
```

	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase
Confirmed	1.000000	0.934698	0.906377	0.927018	0.909720	0.871683	0.859252	0.063550	-0.064815	0.025175	0.999127	0.954710	-0.010161
Deaths	0.934698	1.000000	0.832098	0.871586	0.806975	0.814161	0.765114	0.251565	-0.114529	0.169006	0.939082	0.855330	-0.034708
Recovered	0.906377	0.832098	1.000000	0.682103	0.818942	0.820338	0.919203	0.048438	0.026610	-0.027277	0.899312	0.910013	-0.013697
Active	0.927018	0.871586	0.682103	1.000000	0.851190	0.781123	0.673887	0.054380	-0.132618	0.058386	0.931459	0.847642	-0.003752
New cases	0.909720	0.806975	0.818942	0.851190	1.000000	0.935947	0.914765	0.020104	-0.078666	-0.011637	0.896084	0.959993	0.030791
New deaths	0.871683	0.814161	0.820338	0.781123	0.935947	1.000000	0.889234	0.060399	-0.062792	-0.020750	0.862118	0.894915	0.025293
New recovered	0.859252	0.765114	0.919203	0.673887	0.914765	0.889234	1.000000	0.017090	-0.024293	-0.023340	0.839692	0.954321	0.032662
Deaths / 100 Cases	0.063550	0.251565	0.048438	0.054380	0.020104	0.060399	0.017090	1.000000	-0.168920	0.334594	0.069894	0.015095	-0.134534
Recovered / 100 Cases	-0.064815	-0.114529	0.026610	-0.132618	-0.078666	-0.062792	-0.024293	-0.168920	1.000000	-0.295381	-0.064600	-0.063013	-0.394254
Deaths / 100 Recovered	0.025175	0.169006	-0.027277	0.058386	-0.011637	-0.020750	-0.023340	0.334594	-0.295381	1.000000	0.030460	-0.013763	-0.049083
Confirmed last week	0.999127	0.939082	0.899312	0.931459	0.896084	0.862118	0.839692	0.069894	-0.064600	0.030460	1.000000	0.941448	-0.015247
1 week change	0.954710	0.855330	0.910013	0.847642	0.959993	0.894915	0.954321	0.015095	-0.063013	-0.013763	0.941448	1.000000	0.026594
1 week % increase	-0.010161	-0.034708	-0.013697	-0.003752	0.030791	0.025293	0.032662	-0.134534	-0.394254	-0.049083	-0.015247	0.026594	1.000000

c) From the Unique dataset picked by you from UCI/Kaggle perform the following:

i. Find correlation between features and visualize it

import seaborn as sns

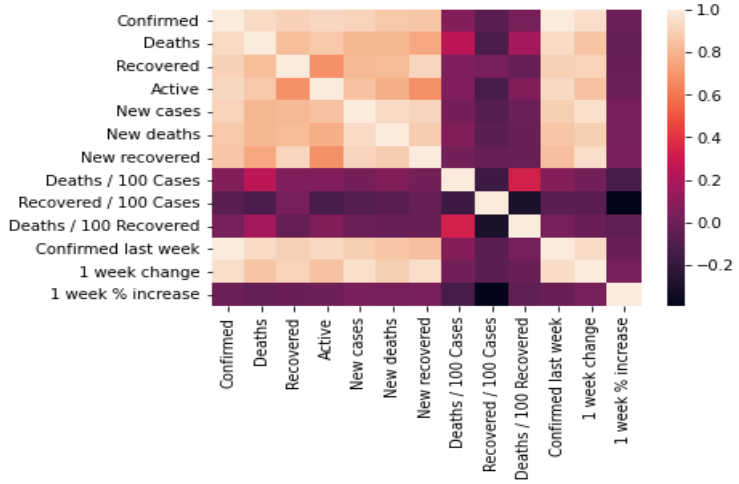
import pandas as pd

corr = ds.corr()

sns.heatmap(corr)

```
import seaborn as sns
import pandas as pd
corr = ds.corr()
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bdc17c2d0>
```

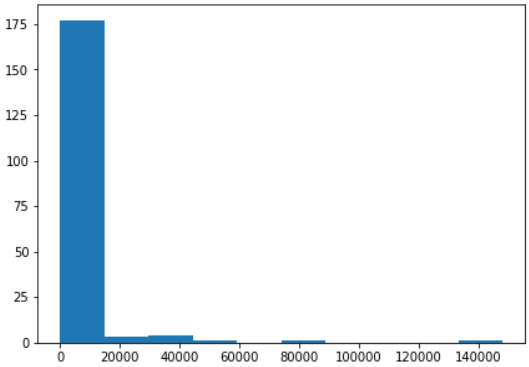


plt.figure(figsize=(7,5))

plt.hist(ds.Deaths)

```
[ ]  plt.figure(figsize=(7,5))
     plt.hist(ds.Deaths)
```

```
(array([177.,   3.,   4.,   1.,   0.,   1.,   0.,   0.,   0.,   1.]),
 array([     0. ,  14801.1,  29602.2,  44403.3,  59204.4,  74005.5,
         88806.6, 103607.7, 118408.8, 133209.9, 148011. ]),
 <a list of 10 Patch objects>)
```

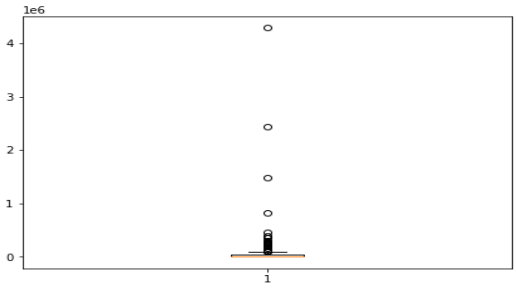


plt.figure(figsize=(7,5))

plt.boxplot(ds.Confirmed)

```
plt.figure(figsize=(7,5))
plt.boxplot(ds.Confirmed)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f5be461c090>],
 'caps': [<matplotlib.lines.Line2D at 0x7f5be4624110>,
  <matplotlib.lines.Line2D at 0x7f5be4624650>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f5be462a150>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f5be4624bd0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f5be461c650>,
  <matplotlib.lines.Line2D at 0x7f5be461cb90>]}
```

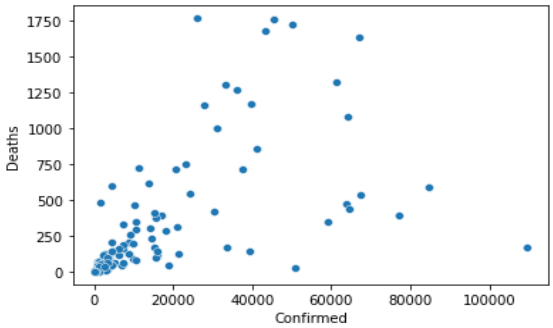


import pandas as pd

import seaborn as sns

sns.scatterplot(x="Confirmed", y="Deaths", data=ds);

```
import pandas as pd
import seaborn as sns
sns.scatterplot(x="Confirmed", y="Deaths", data=ds);
```

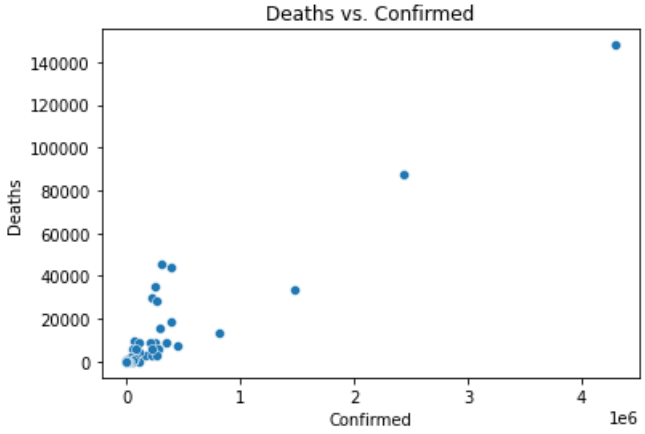


ax.set_title("Deaths vs. Confirmed")

ax.set_xlabel("Confirmed")

```
ax = sns.scatterplot(x="Confirmed", y="Deaths", data=ds)
ax.set_title("Deaths vs. Confirmed")
ax.set_xlabel("Confirmed")
```

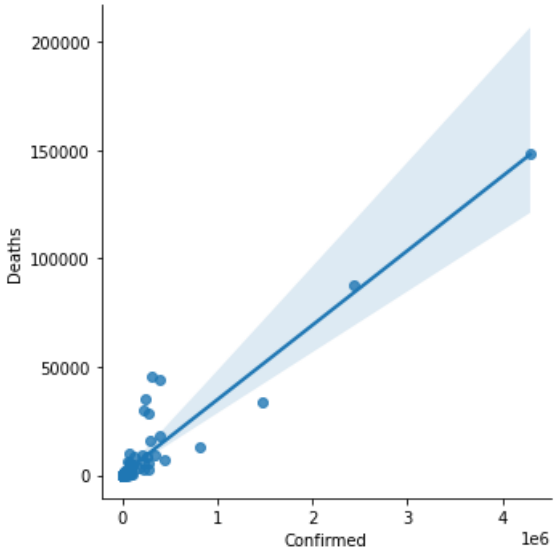
```
Text(0.5, 0, 'Confirmed')
```



sns.lmplot(x="Confirmed", y="Deaths", data=ds)

```
[ ]  sns.lmplot(x="Confirmed", y="Deaths", data=ds)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5be126a210>
```



ii. Perform dimensionality reduction

DIMENSIONALITY REDUCTION

```
ds.corr(method='pearson')
#eliminate column with correlation > 0.5
```

	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase
Confirmed	1.000000	0.934698	0.906377	0.927018	0.909720	0.871683	0.859252	0.063550	-0.064815	0.025175	0.999127	0.954710	-0.010161
Deaths	0.934698	1.000000	0.832098	0.871586	0.806975	0.814161	0.765114	0.251565	-0.114529	0.169006	0.939082	0.855330	-0.034708
Recovered	0.906377	0.832098	1.000000	0.682103	0.818942	0.820338	0.919203	0.048438	0.026610	-0.027277	0.899312	0.910013	-0.013697
Active	0.927018	0.871586	0.682103	1.000000	0.851190	0.781123	0.673887	0.054380	-0.132618	0.058386	0.931459	0.847642	-0.003752
New cases	0.909720	0.806975	0.818942	0.851190	1.000000	0.935947	0.914765	0.020104	-0.078666	-0.011637	0.896084	0.959993	0.030791
New deaths	0.871683	0.814161	0.820338	0.781123	0.935947	1.000000	0.889234	0.060399	-0.062792	-0.020750	0.862118	0.894915	0.025293
New recovered	0.859252	0.765114	0.919203	0.673887	0.914765	0.889234	1.000000	0.017090	-0.024293	-0.023340	0.839692	0.954321	0.032662
Deaths / 100 Cases	0.063550	0.251565	0.048438	0.054380	0.020104	0.060399	0.017090	1.000000	-0.168920	0.334594	0.069894	0.015095	-0.134534
Recovered / 100 Cases	-0.064815	-0.114529	0.026610	-0.132618	-0.078666	-0.062792	-0.024293	-0.168920	1.000000	-0.295381	-0.064600	-0.063013	-0.394254
Deaths / 100 Recovered	0.025175	0.169006	-0.027277	0.058386	-0.011637	-0.020750	-0.023340	0.334594	-0.295381	1.000000	0.030460	-0.013763	-0.049083
Confirmed last week	0.999127	0.939082	0.899312	0.931459	0.896084	0.862118	0.839692	0.069894	-0.064600	0.030460	1.000000	0.941448	-0.015247
1 week change	0.954710	0.855330	0.910013	0.847642	0.959993	0.894915	0.954321	0.015095	-0.063013	-0.013763	0.941448	1.000000	0.026594
1 week % increase	-0.010161	-0.034708	-0.013697	-0.003752	0.030791	0.025293	0.032662	-0.134534	-0.394254	-0.049083	-0.015247	0.026594	1.000000

iii. Detect outliers

import seaborn as sb

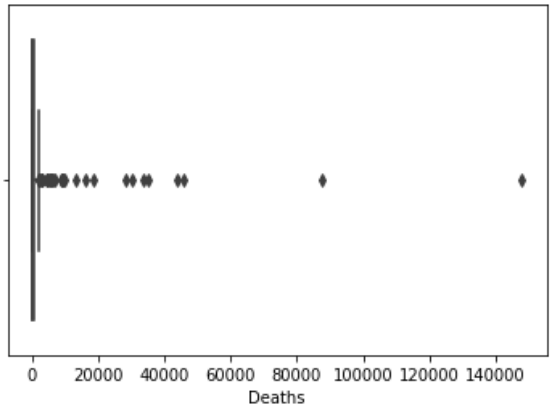
import matplotlib.pyplot as plt

sb.boxplot(x='Deaths', data=ds)

DETECT OUTLIERS

```
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(x='Deaths', data=ds)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f145c0d2ad0>
```



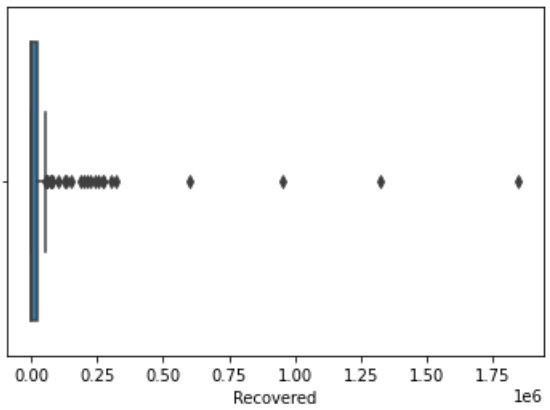
import seaborn as sb

import matplotlib.pyplot as plt

sb.boxplot(x='Recovered', data=ds)

```
[ ] import seaborn as sb
    import matplotlib.pyplot as plt
    sb.boxplot(x='Recovered', data=ds)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f145c09c610>
```



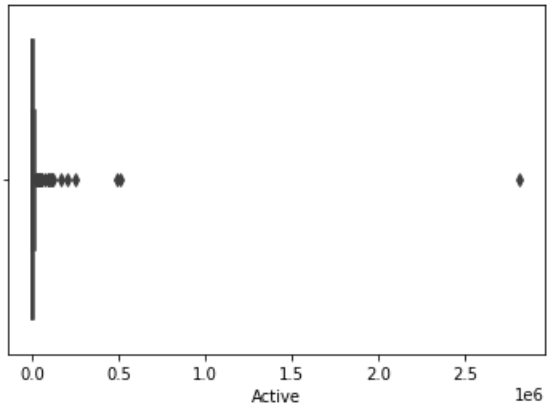
import seaborn as sb

import matplotlib.pyplot as plt

sb.boxplot(x='Active', data=ds)

```
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(x='Active', data=ds)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f145c015f10>
```



#removing outliers

import sklearn

import numpy as np

from sklearn.datasets import load_boston

import pandas as pd

import seaborn as sb

Q1 = np.percentile(ds['Deaths'], 25, interpolation = 'midpoint')

Q3 = np.percentile(ds['Deaths'], 75, interpolation = 'midpoint')

IQR = Q3 - Q1

print("Old Shape: ", ds.shape)

upper = np.where(ds['Deaths'] >= (Q3+1.5*IQR))

lower = np.where(ds['Deaths'] <= (Q1-1.5*IQR))

ds.drop(upper[0], inplace = True)

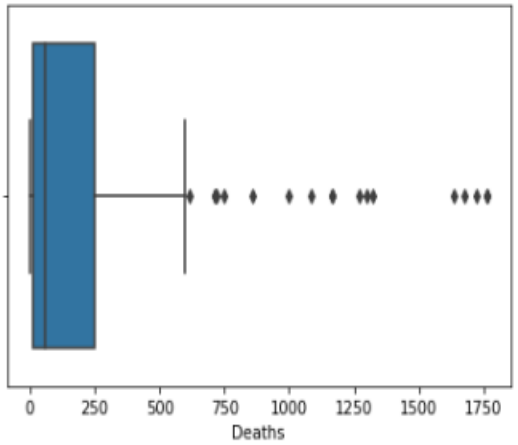
ds.drop(lower[0], inplace = True)

print("New Shape: ", ds.shape)

sb.boxplot(x='Deaths', data=ds)

```
#removing outliers
import sklearn
import numpy as np
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sb

Q1 = np.percentile(ds['Deaths'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(ds['Deaths'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", ds.shape)

upper = np.where(ds['Deaths'] >= (Q3+1.5*IQR))

lower = np.where(ds['Deaths'] <= (Q1-1.5*IQR))

ds.drop(upper[0], inplace = True)
ds.drop(lower[0], inplace = True)

print("New Shape: ", ds.shape)

sb.boxplot(x='Deaths', data=ds)
```

```
Old Shape:  (187, 15)
New Shape:  (154, 15)
<matplotlib.axes._subplots.AxesSubplot at 0x7f145cba4f50>
```



iv. Fill missing values

#checking for missing data

ds.isnull().sum()

```
[ ]  #checking for missing data
     ds.isnull().sum()

     Country/Region            0
     Confirmed                 0
     Deaths                    0
     Recovered                 0
     Active                    0
     New cases                 0
     New deaths                0
     New recovered             0
     Deaths / 100 Cases        0
     Recovered / 100 Cases     0
     Deaths / 100 Recovered    0
     Confirmed last week       0
     1 week change             0
     1 week % increase         0
     WHO Region                0
     dtype: int64
```

ds.value_counts("Deaths")

```
ds.value_counts("Deaths")

Deaths
0       17
11       4
1        3
2        3
7        3
        ..
54       1
53       1
52       1
50       1
1764     1
Length: 117, dtype: int64
```

No misssing data found

v. Data Partitioning

print('WHO Region: ',np.unique(ds['WHO Region']))

```
[ ]  print('WHO Region: ',np.unique(ds['WHO Region']))

     WHO Region:  ['Africa' 'Americas' 'Eastern Mediterranean' 'Europe' 'South-East Asia'
      'Western Pacific']
```

ds.iloc[:,0].values

```
[ ]  ds.iloc[:,0].values
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua and Barbuda', 'Armenia', 'Australia', 'Austria',
       'Azerbaijan', 'Bahamas', 'Bahrain', 'Barbados', 'Belarus',
       'Belize', 'Benin', 'Bhutan', 'Bosnia and Herzegovina', 'Botswana',
       'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
       'Cabo Verde', 'Cambodia', 'Cameroon', 'Central African Republic',
       'Chad', 'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)',
       'Costa Rica', "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus',
       'Czechia', 'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic',
       'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'Gabon', 'Gambia',
       'Georgia', 'Ghana', 'Greece', 'Greenland', 'Grenada', 'Guatemala',
       'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Holy See',
       'Honduras', 'Hungary', 'Iceland', 'Ireland', 'Israel', 'Jamaica',
       'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
       'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia',
       'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
       'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta', 'Mauritania',
       'Mauritius', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
       'Morocco', 'Mozambique', 'Namibia', 'Nepal', 'New Zealand',
       'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
       'Oman', 'Panama', 'Papua New Guinea', 'Paraguay', 'Poland',
       'Portugal', 'Qatar', 'Rwanda', 'Saint Kitts and Nevis',
       'Saint Lucia', 'Saint Vincent and the Grenadines', 'San Marino',
       'Sao Tome and Principe', 'Senegal', 'Serbia', 'Seychelles',
       'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia', 'Somalia',
       'South Korea', 'South Sudan', 'Sri Lanka', 'Sudan', 'Suriname',
       'Syria', 'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand',
       'Timor-Leste', 'Togo', 'Trinidad and Tobago', 'Tunisia', 'Uganda',
       'Ukraine', 'United Arab Emirates', 'Uruguay', 'Uzbekistan',
       'Venezuela', 'Vietnam', 'West Bank and Gaza', 'Western Sahara',
       'Yemen', 'Zambia', 'Zimbabwe'], dtype=object)
```

vi. Perform the following statistical test for the dataset

#NORMALITY TEST

from scipy.stats import shapiro

ds=np.random.normal(loc=20, scale=5, size=150)

stat, p=shapiro(ds)

print('stat=%.3f, p=%.3f\n' % (stat,p))

if p>0.05:

    print("Might be Gaussian")

else:

    print("Not Gaussian")

```
#NORMALITY TEST
from scipy.stats import shapiro
ds=np.random.normal(loc=20, scale=5, size=150)
stat, p=shapiro(ds)
print('stat=%.3f, p=%.3f\n' % (stat,p))
if p>0.05:
    print("Might be Gaussian")
else:
    print("Not Gaussian")
```

```
stat=0.994, p=0.778

Might be Gaussian
```

#CORRELATION TEST

from scipy.stats import pearsonr

data1=ds['Confirmed']

data2=ds['Deaths']

stat,p=pearsonr(data1,data2)

print('Staistics = %3f,p = %3f' %(stat,p))

if p>0.05:

    print('Probably independent')

else:

    print('Probably dependent')

```
#CORRELATION TEST
from scipy.stats import pearsonr
data1=ds['Confirmed']
data2=ds['Deaths']
stat,p=pearsonr(data1,data2)
print('Staistics = %3f,p = %3f' %(stat,p))
if p>0.05:
    print('Probably independent')
else:
    print('Probably dependent')
```

```
Staistics = 0.934698,p = 0.000000
Probably dependent
```

#PARAMETRIC STATISTICAL HYPOTHESIS TEST

from scipy.stats import ttest_ind

data1=ds['Confirmed']

data2=ds['Deaths']

stat,p=ttest_ind(data1,data2)

print('Staistics = %3f,p = %3f' %(stat,p))

if p>0.05:

    print('Probably the same distribution')

else:

    print('Probably different distributions')

```
#PARAMETRIC STATISTICAL HYPOTHESIS TEST
from scipy.stats import ttest_ind
data1=ds['Confirmed']
data2=ds['Deaths']
stat,p=ttest_ind(data1,data2)
print('Staistics = %3f,p = %3f' %(stat,p))
if p>0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
Staistics = 3.017235,p = 0.002726
Probably different distributions
```

#NON PARAMETRIC HYPOTHESIS TEST

from scipy.stats import mannwhitneyu

data1=ds['Confirmed']

data2=ds['Deaths']

stat,p=mannwhitneyu(data1,data2)

print('Staistics = %3f,p = %3f' %(stat,p))

if p>0.05:

    print('Probably the same distribution')

else:

    print('Probably different distributions')

```
#NON PARAMETRIC HYPOTHESIS TEST
from scipy.stats import mannwhitneyu
data1=ds['Confirmed']
data2=ds['Deaths']
stat,p=mannwhitneyu(data1,data2)
print('Staistics = %3f,p = %3f' %(stat,p))
if p>0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
Staistics = 5900.500000,p = 0.000000
Probably different distributions
```

RESULT:

Thus, descriptive analysis was performed on the datasets.

Date: 22/3/22 EXP: 2a-ii	UNIVARIATE ANALYSIS

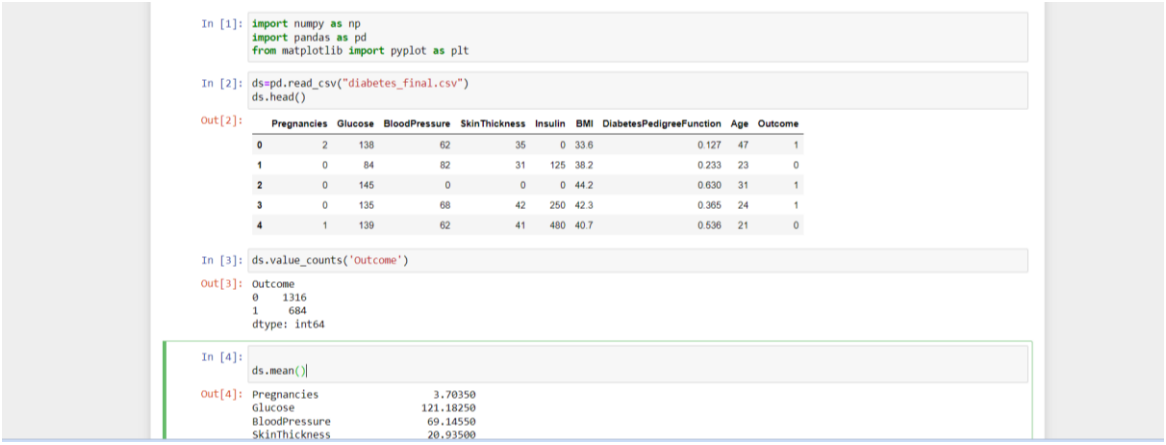
a) Univariate Analysis on Pima Indian Diabetes Dataset

AIM:

To use the diabetes data set from UCI and Pima Indians Diabetes data set and Perform the following: Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

SOURCE CODE:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
ds=pd.read_csv("diabetes_final.csv")
ds.head()
ds.value_counts('Outcome')
ds.mean()
```



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

In [2]:
```
ds=pd.read_csv("diabetes_final.csv")
ds.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

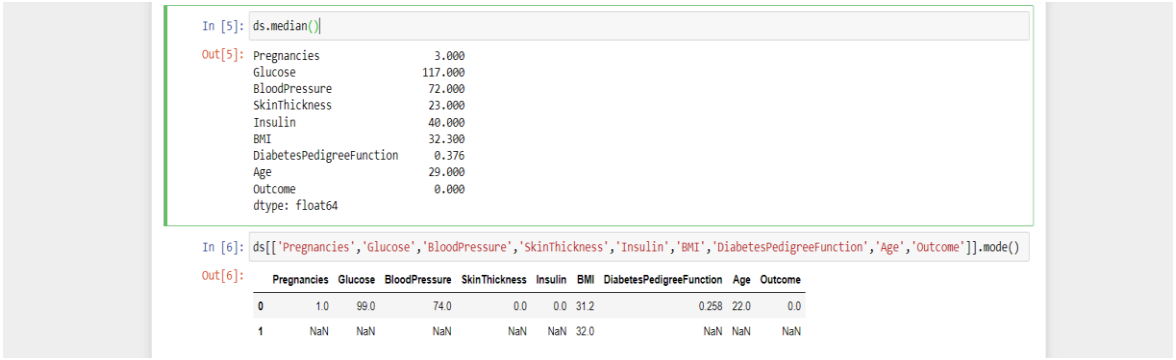
In [3]:
```
ds.value_counts('Outcome')
```

Out[3]:
```
Outcome
0    1316
1     684
dtype: int64
```

In [4]:
```
ds.mean()
```

Out[4]:
```
Pregnancies                 3.70350
Glucose                   121.18250
BloodPressure              69.14550
SkinThickness              20.93500
```

```
ds.median()
ds[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']].mode()
```



In [5]:
```
ds.median()
```

Out[5]:
```
Pregnancies                 3.000
Glucose                   117.000
BloodPressure              72.000
SkinThickness              23.000
Insulin                    40.000
BMI                        32.300
DiabetesPedigreeFunction    0.376
Age                        29.000
Outcome                     0.000
dtype: float64
```

In [6]:
```
ds[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']].mode()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1.0	99.0	74.0	0.0	0.0	31.2	0.258	22.0	0.0
1	NaN	NaN	NaN	NaN	NaN	32.0	NaN	NaN	NaN

ds.std()

ds['Pregnancies'].var()

ds.skew()

ds.kurtosis()

```
In [7]: ds.std()
Out[7]: Pregnancies                 3.306063
        Glucose                    32.068636
        BloodPressure              19.188315
        SkinThickness              16.103243
        Insulin                   111.180534
        BMI                         8.149901
        DiabetesPedigreeFunction    0.323553
        Age                        11.786423
        Outcome                     0.474498
        dtype: float64

In [8]: ds['Pregnancies'].var()
Out[8]: 10.930052776388179

In [9]: ds.skew()
Out[9]: Pregnancies                 0.982366
        Glucose                     0.158806
        BloodPressure              -1.854476
        SkinThickness               0.207228
        Insulin                     1.996084
        BMI                        -0.090455
        DiabetesPedigreeFunction    1.811979
        Age                         1.181267
        Outcome                     0.666633
        dtype: float64
```

```
In [10]: ds.kurtosis()
Out[10]: Pregnancies                 0.409868
         Glucose                     0.560371
         BloodPressure               5.328490
         SkinThickness               0.155580
         Insulin                     5.128262
         BMI                         4.131722
         DiabetesPedigreeFunction    5.006840
         Age                         0.826383
         Outcome                    -1.557158
         dtype: float64
```

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

ds=pd.read_csv("pima_indian_diabetes.csv")

ds.head()

ds.value_counts('Outcome')

ds.mean()

```
In [3]: ds=pd.read_csv("pima_indian_diabetes.csv")
        ds.head()
Out[3]:
```

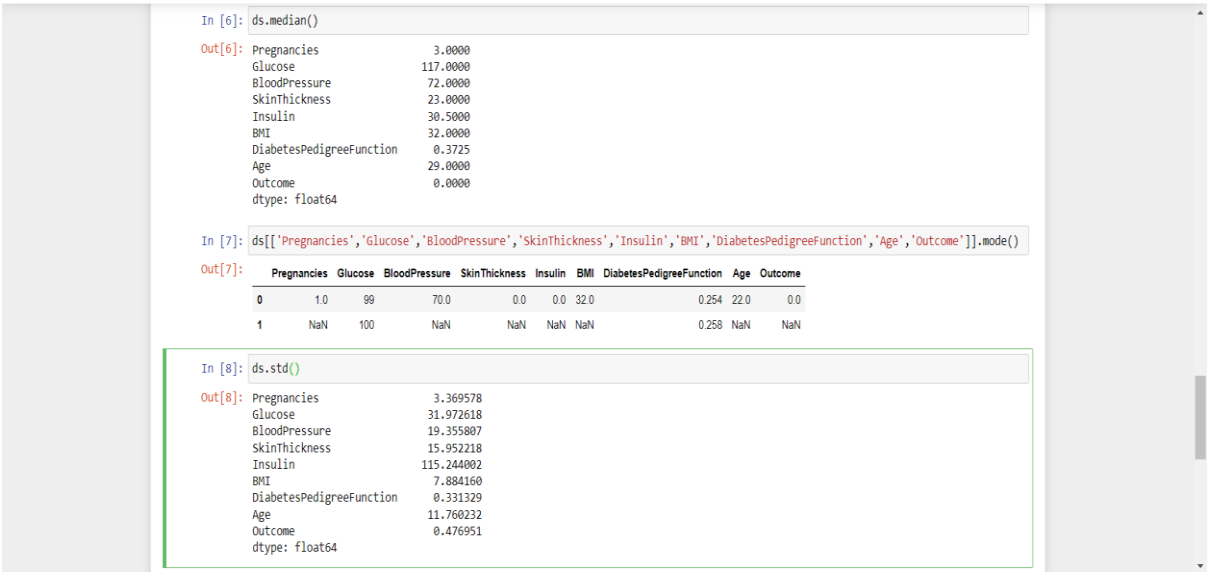
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [4]: ds.value_counts('Outcome')
Out[4]: Outcome
        0    500
        1    268
        dtype: int64

In [5]: ds.mean()
Out[5]: Pregnancies                 3.845052
        Glucose                   120.894531
        BloodPressure              69.105469
        SkinThickness              20.536458
        Insulin                    79.799479
        BMI                        31.992578
        DiabetesPedigreeFunction    0.471876
        Age                        33.240885
        Outcome                     0.348958
        dtype: float64
```

ds.median()

ds[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']].mode()

ds.std()



```
In [6]: ds.median()

Out[6]: Pregnancies                 3.0000
        Glucose                   117.0000
        BloodPressure              72.0000
        SkinThickness              23.0000
        Insulin                    30.5000
        BMI                        32.0000
        DiabetesPedigreeFunction    0.3725
        Age                        29.0000
        Outcome                     0.0000
        dtype: float64

In [7]: ds[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']].mode()

Out[7]:
```

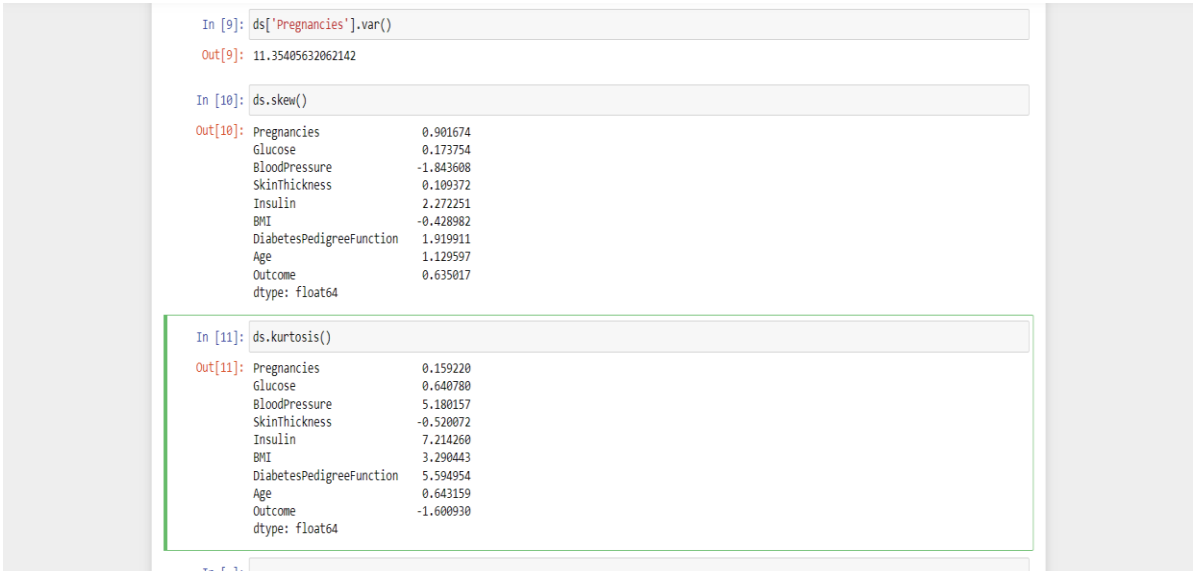
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1.0	99	70.0	0.0	0.0	32.0	0.254	22.0	0.0
1	NaN	100	NaN	NaN	NaN	NaN	0.258	NaN	NaN

```
In [8]: ds.std()

Out[8]: Pregnancies                 3.369578
        Glucose                    31.972618
        BloodPressure              19.355807
        SkinThickness              15.952218
        Insulin                   115.244002
        BMI                         7.884160
        DiabetesPedigreeFunction    0.331329
        Age                        11.760232
        Outcome                     0.476951
        dtype: float64
```

ds['Pregnancies'].var()

ds.skew()

ds.kurtosis()



```
In [9]: ds['Pregnancies'].var()

Out[9]: 11.35405632062142

In [10]: ds.skew()

Out[10]: Pregnancies                 0.901674
         Glucose                     0.173754
         BloodPressure              -1.843608
         SkinThickness               0.109372
         Insulin                     2.272251
         BMI                        -0.428982
         DiabetesPedigreeFunction    1.919911
         Age                         1.129597
         Outcome                     0.635017
         dtype: float64

In [11]: ds.kurtosis()

Out[11]: Pregnancies                 0.159220
         Glucose                     0.640780
         BloodPressure               5.180157
         SkinThickness              -0.520072
         Insulin                     7.214260
         BMI                         3.290443
         DiabetesPedigreeFunction    5.594954
         Age                         0.643159
         Outcome                    -1.600930
         dtype: float64
```

b) Univariate Analysis

AIM:

To use the COVID dataset and perform the following: Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

SOURCE CODE:

print(ds.sum())

```
[ ]  print(ds.sum())

Country/Region          AfghanistanAlbaniaAlgeriaAndorraAngolaAntigua ...
Confirmed                                                      16480485
Deaths                                                           654036
Recovered                                                       9468087
Active                                                          6358362
New cases                                                        228693
New deaths                                                         5415
New recovered                                                    174623
Deaths / 100 Cases                                               564.65
Recovered / 100 Cases                                         12121.44
Deaths / 100 Recovered                                             inf
Confirmed last week                                            14713623
1 week change                                                   1766862
1 week % increase                                               2544.36
WHO Region              Eastern MediterraneanEuropeAfricaEuropeAfricaA...
dtype: object
```

ds['WHO Region'].value_counts()

```
[ ]  ds['WHO Region'].value_counts()

Europe                  56
Africa                  48
Americas                35
Eastern Mediterranean   22
Western Pacific         16
South-East Asia         10
Name: WHO Region, dtype: int64
```

ds.value_counts()

```
ds.value_counts()
```

Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recov
Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69.49
Pakistan	274289	5842	241026	27421	1176	20	3592	2.13	87.87
Nepal	18752	48	13754	4950	139	3	626	0.26	73.35
Netherlands	53413	6160	189	47064	419	1	0	11.53	0.35
New Zealand	1557	22	1514	21	1	0	1	1.41	97.24
Georgia	1137	16	922	199	6	0	2	1.41	81.09
Germany	207112	9125	190314	7673	445	1	259	4.41	91.89
Ghana	33624	168	29801	3655	655	0	307	0.50	88.63
Greece	4227	202	1374	2651	34	0	0	4.78	32.51
Zimbabwe	2704	36	542	2126	192	2	24	1.33	20.04

Length: 187, dtype: int64

ds[['Confirmed', 'Deaths', 'Recovered', 'Active']].mean()

```
[ ] ds[['Confirmed', 'Deaths', 'Recovered', 'Active']].mean()

Confirmed    88130.935829
Deaths        3497.518717
Recovered    50631.481283
Active       34001.935829
dtype: float64
```

ds[['Confirmed', 'Deaths', 'Recovered', 'Active']].median()

```
[ ] ds[['Confirmed', 'Deaths', 'Recovered', 'Active']].median()

Confirmed    5059.0
Deaths        108.0
Recovered    2815.0
Active       1600.0
dtype: float64
```

from scipy import stats

x = stats.mode(ds.Deaths)

print(x)

```
[ ] from scipy import stats
    x = stats.mode(ds.Deaths)
    print(x)

ModeResult(mode=array([0]), count=array([17]))
```

ds.var()

```
[ ] ds.var()

/usr/local/lib/python3.7/dist-packages/ipykernel_launche
  """Entry point for launching an IPython kernel.
Confirmed                 1.469332e+11
Deaths                    1.988101e+08
Recovered                 3.617155e+10
Active                    4.550806e+10
New cases                 3.260838e+07
New deaths                1.440892e+04
New recovered             1.762085e+07
Deaths / 100 Cases        1.193221e+01
Recovered / 100 Cases     6.910429e+02
Deaths / 100 Recovered             NaN
Confirmed last week       1.144291e+11
1 week change             2.255407e+09
1 week % increase         6.007321e+02
dtype: float64
```

print(ds.std())

```
print(ds.std())

Confirmed                 383318.663831
Deaths                     14100.002482
Recovered                 190188.189643
Active                    213326.173371
New cases                   5710.374790
New deaths                   120.037173
New recovered               4197.719635
Deaths / 100 Cases             3.454302
Recovered / 100 Cases         26.287694
Deaths / 100 Recovered              NaN
Confirmed last week       338273.676567
1 week change              47491.127684
1 week % increase             24.509838
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_laur
  """Entry point for launching an IPython kernel.
```

from scipy.stats import skew

ds.skew(axis = 0, skipna = True)

```
from scipy.stats import skew
ds.skew(axis = 0, skipna = True)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_lau

Confirmed                 8.725676
Deaths                    7.464481
Recovered                 6.983644
Active                   12.182067
New cases                 7.720320
New deaths                5.970033
New recovered             6.769567
Deaths / 100 Cases        3.352173
Recovered / 100 Cases    -0.823366
Deaths / 100 Recovered         NaN
Confirmed last week       8.865198
1 week change             7.692012
1 week % increase         6.114613
dtype: float64
```

from scipy.stats import kurtosis

import numpy as np

import pylab as p

x1 = np.linspace(-5, 5, 1000)

y1 = 1./(np.sqrt(2.*np.pi)) * np.exp(-.5*(x1)**2)

p.plot(x1, y1, '*')

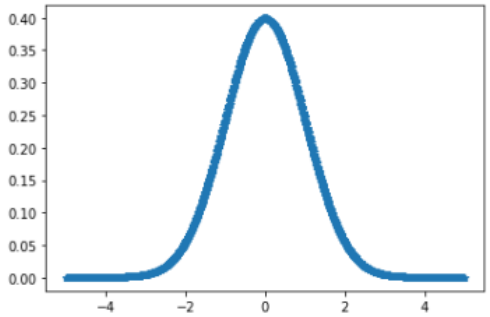
print('\nKurtosis for normal distribution :', kurtosis(y1))

print('\nKurtosis for normal distribution :', kurtosis(y1, fisher = False))

print('\nKurtosis for normal distribution :', kurtosis(y1, fisher = True))

```
from scipy.stats import kurtosis
import numpy as np
import pylab as p
x1 = np.linspace( -5, 5, 1000 )
y1 = 1./(np.sqrt(2.*np.pi)) * np.exp( -.5*(x1)**2  )
p.plot(x1, y1, '*')
print( '\nKurtosis for normal distribution :', kurtosis(y1))
print( '\nKurtosis for normal distribution :', kurtosis(y1, fisher = False))
print( '\nKurtosis for normal distribution :',  kurtosis(y1, fisher = True))
```

```
Kurtosis for normal distribution : -0.3073930877422071

Kurtosis for normal distribution : 2.692606912257793

Kurtosis for normal distribution : -0.3073930877422071
```



c) Univariate Analysis

AIM:

To use the COVID dataset and perform the following:

SOURCE CODE:

A. Univariate Linear Regression

```
from sklearn.linear_model import LinearRegression
x=ds['Active'].to_numpy().reshape((-1,1))
y=ds['Deaths'].to_numpy()
model = LinearRegression().fit(x,y)
r_sq=model.score(x,y)
print("R square value :",r_sq)
```

```
[ ]  #UNIVARIATE LINEAR REGRESSION
     from sklearn.linear_model import LinearRegression
     x=ds['Active'].to_numpy().reshape((-1,1))
     y=ds['Deaths'].to_numpy()
     model = LinearRegression().fit(x,y)
     r_sq=model.score(x,y)
     print("R square value :",r_sq)

     R square value : 0.7596620120838735
```

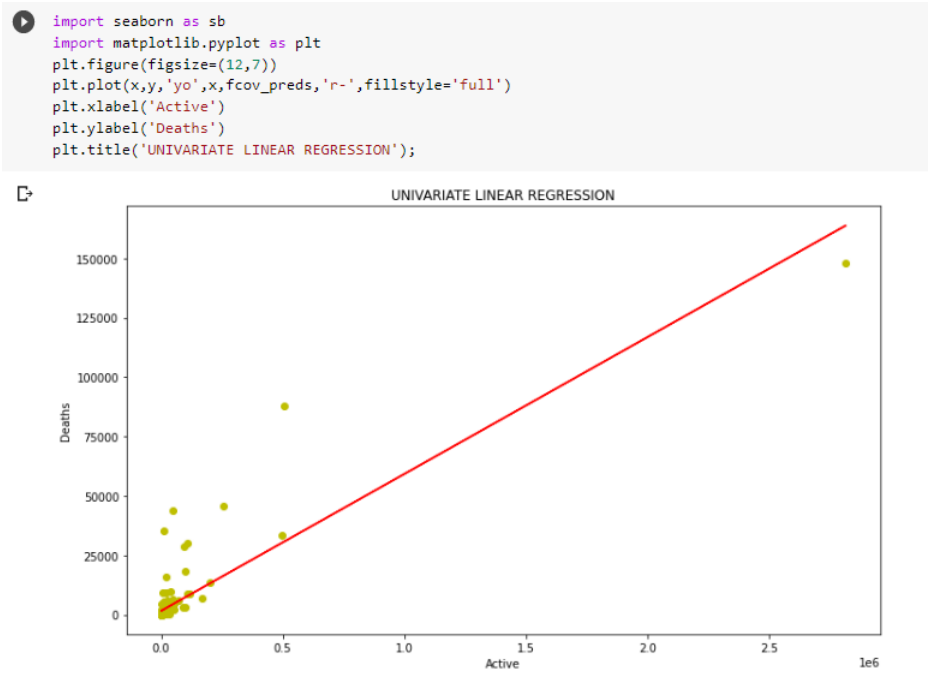
```
import numpy as np
import pandas as pd
x = ds['Active']
y = ds['Deaths']
def mean(vals):
    return sum(vals)/float(len(vals))
def variance(vals):
    val_mean = mean(vals)
    return sum([(x-val_mean)**2 for x in vals])
def covariance(x,x_mean,y,y_mean):
    cov = 0.0
    for i in range(len(x)):
        cov += (x[i]-x_mean) * (y[i]-y_mean)
    return cov
def getBetas(x,y):
```

```
    x_mean = mean(x)
    y_mean = mean(y)
    cov = covariance(x,x_mean,y,y_mean)
    x_var = variance(x)
    b1 = cov/x_var
    b0 = np.mean(y) - b1 * np.mean(x)
    return b0,b1
def predict(x,y):
    intercept,coeff = getBetas(x,y)
    preds = intercept+(coeff*x)
    return preds
intercept, coeff = getBetas(x,y)
print(f'Intercept: {intercept}\n\nCoefficient: {coeff}')
fcov_preds = predict(x,y)
```

```
import numpy as np
import pandas as pd
x = ds['Active']
y = ds['Deaths']

def mean(vals):
    return sum(vals)/float(len(vals))

def variance(vals):
    val_mean = mean(vals)
    return sum([(x-val_mean)**2 for x in vals])

def covariance(x,x_mean,y,y_mean):
    cov = 0.0
    for i in range(len(x)):
        cov += (x[i]-x_mean) * (y[i]-y_mean)
    return cov

def getBetas(x,y):
    x_mean = mean(x)
    y_mean = mean(y)
    cov = covariance(x,x_mean,y,y_mean)
    x_var = variance(x)
    b1 = cov/x_var
    b0 = np.mean(y) - b1 * np.mean(x)
    return b0,b1

def predict(x,y):
    intercept,coeff = getBetas(x,y)
    preds = intercept+(coeff*x)
    return preds

intercept, coeff = getBetas(x,y)
print(f'Intercept: {intercept}\n\nCoefficient: {coeff}')
fcov_preds = predict(x,y)
```

```
Intercept: 1538.7241343664866

Coefficient: 0.05760832536327234
```

```
import seaborn as sb
import matplotlib.pyplot as plt
plt.figure(figsize=(12,7))
plt.plot(x,y,'yo',x,fcov_preds,'r-',fillstyle='full')
plt.xlabel('Active')
plt.ylabel('Deaths')
plt.title('UNIVARIATE LINEAR REGRESSION');
```



B. Summary statistics

print(ds.describe())

```
[ ]  #SUMMARY STATISTICS
     print(ds.describe())
```

	Confirmed	Deaths	Recovered	Active	New cases
count	1.870000e+02	187.000000	1.870000e+02	1.870000e+02	187.000000
mean	8.813094e+04	3497.518717	5.063148e+04	3.400194e+04	1222.957219
std	3.833187e+05	14100.002482	1.901882e+05	2.133262e+05	5710.374790
min	1.000000e+01	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	1.114000e+03	18.500000	6.265000e+02	1.415000e+02	4.000000
50%	5.059000e+03	108.000000	2.815000e+03	1.600000e+03	49.000000
75%	4.046050e+04	734.000000	2.260600e+04	9.149000e+03	419.500000
max	4.290259e+06	148011.000000	1.846641e+06	2.816444e+06	56336.000000

	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases
count	187.000000	187.000000	187.000000	187.000000
mean	28.957219	933.812834	3.019519	64.820535
std	120.037173	4197.719635	3.454302	26.287694
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.945000	48.770000
50%	1.000000	22.000000	2.150000	71.320000
75%	6.000000	221.000000	3.875000	86.885000
max	1076.000000	33728.000000	28.560000	100.000000

	Deaths / 100 Recovered	Confirmed last week	1 week change
count	187.00	1.870000e+02	187.000000
mean	inf	7.868248e+04	9448.459893
std	NaN	3.382737e+05	47491.127684
min	0.00	1.000000e+01	-47.000000
25%	1.45	1.051500e+03	49.000000
50%	3.62	5.020000e+03	432.000000
75%	6.44	3.708050e+04	3172.000000
max	inf	3.834677e+06	455582.000000

	1 week % increase
count	187.000000
mean	13.606203
std	24.509838
min	-3.840000
25%	2.775000
50%	6.890000
75%	16.855000
max	226.320000

print(ds.describe(include=['object']))

```
print(ds.describe(include=['object']))
```

```
       Country/Region WHO Region
count             187         187
unique            187           6
top       Afghanistan      Europe
freq                1          56
```

print(ds.describe(include= 'all'))

```
print(ds.describe(include='all'))
```

```
       Country/Region     Confirmed         Deaths     Recovered  \
count             187  1.870000e+02     187.000000  1.870000e+02
unique            187           NaN            NaN           NaN
top       Afghanistan           NaN            NaN           NaN
freq                1           NaN            NaN           NaN
mean              NaN  8.813094e+04    3497.518717  5.063148e+04
std               NaN  3.833187e+05   14100.002482  1.901882e+05
min               NaN  1.000000e+01       0.000000  0.000000e+00
25%               NaN  1.114000e+03      18.500000  6.265000e+02
50%               NaN  5.059000e+03     108.000000  2.815000e+03
75%               NaN  4.046050e+04     734.000000  2.260600e+04
max               NaN  4.290259e+06  148011.000000  1.846641e+06

              Active     New cases   New deaths  New recovered  \
count   1.870000e+02    187.000000   187.000000     187.000000
unique           NaN           NaN          NaN            NaN
top              NaN           NaN          NaN            NaN
freq             NaN           NaN          NaN            NaN
mean    3.400194e+04   1222.957219    28.957219     933.812834
std     2.133262e+05   5710.374790   120.037173    4197.719635
min     0.000000e+00      0.000000     0.000000       0.000000
25%     1.415000e+02      4.000000     0.000000       0.000000
50%     1.600000e+03     49.000000     1.000000      22.000000
75%     9.149000e+03    419.500000     6.000000     221.000000
max     2.816444e+06  56336.000000  1076.000000   33728.000000

        Deaths / 100 Cases  Recovered / 100 Cases  Deaths / 100 Recovered  \
count           187.000000             187.000000                  187.00
unique                 NaN                    NaN                     NaN
top                    NaN                    NaN                     NaN
```

ds.agg({

"Deaths": ["min", "max", "median", "skew"],

"Active": ["min", "max", "median", "mean"],

"Recovered": ["min", "max", "median", "mean"],

"Confirmed": ["min", "max", "median", "mean"], })

```
ds.agg(
    {
        "Deaths": ["min", "max", "median", "skew"],
        "Active": ["min", "max", "median", "mean"],
        "Recovered": ["min", "max", "median", "mean"],
        "Confirmed": ["min", "max", "median", "mean"],
    }
)
```

	Deaths	Active	Recovered	Confirmed
min	0.000000	0.000000e+00	0.000000e+00	1.000000e+01
max	148011.000000	2.816444e+06	1.846641e+06	4.290259e+06
median	108.000000	1.600000e+03	2.815000e+03	5.059000e+03
skew	7.464481	NaN	NaN	NaN
mean	NaN	3.400194e+04	5.063148e+04	8.813094e+04

C. Frequency

ds['Country/Region'].value_counts()

```
ds['Country/Region'].value_counts()
```

```
Afghanistan    1
Pakistan       1
Nepal          1
Netherlands    1
New Zealand    1
              ..
Georgia        1
Germany        1
Ghana          1
Greece         1
Zimbabwe       1
Name: Country/Region, Length: 187, dtype: int64
```

#FREQUENCY TABLE

ds['Country/Region'].value_counts()

```
[ ] #FREQUENCY TABLE
    ds['Country/Region'].value_counts()
```

```
Afghanistan    1
Pakistan       1
Nepal          1
Netherlands    1
New Zealand    1
              ..
Georgia        1
Germany        1
Ghana          1
Greece         1
Zimbabwe       1
Name: Country/Region, Length: 187, dtype: int64
```

import pandas as pd

my_tab = pd.crosstab(index=ds["Country/Region"], columns="count")

my_tab

```
import pandas as pd
my_tab = pd.crosstab(index=ds["Country/Region"], columns="count")
my_tab
```

Country/Region	col_0 count
Afghanistan	1
Albania	1
Algeria	1
Andorra	1
Angola	1
...	...
West Bank and Gaza	1
Western Sahara	1
Yemen	1
Zambia	1
Zimbabwe	1

187 rows × 1 columns

ds.groupby(["Country/Region"])['Active'].count()

```
ds.groupby(["Country/Region"])['Active'].count()
```

```
Country/Region
Afghanistan           1
Albania               1
Algeria               1
Andorra               1
Angola                1
                     ..
West Bank and Gaza    1
Western Sahara        1
Yemen                 1
Zambia                1
Zimbabwe              1
Name: Active, Length: 187, dtype: int64
```

my_crosstab = pd.crosstab(index=ds["Country/Region"], columns=ds["Active"], margins=True)

my_crosstab

```
my_crosstab = pd.crosstab(index=ds["Country/Region"], columns=ds["Active"], margins=True)
my_crosstab
```

Active	0	1	2	4	8	9	12	13	15	18	...	107514	108928	117163	170537	201097	254427	495499
Country/Region																		
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Albania	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Algeria	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Andorra	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Angola	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Western Sahara	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Yemen	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Zambia	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
Zimbabwe	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
All	5	3	3	1	1	2	1	2	1	1	...	1	1	1	1	1	1	1

188 rows × 174 columns

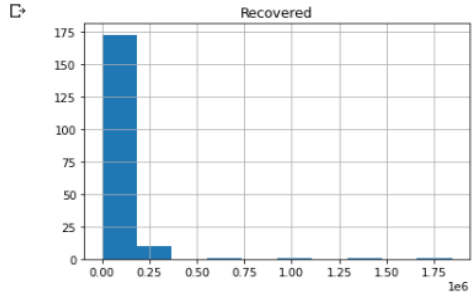
D. Charts

import matplotlib.pyplot as plt

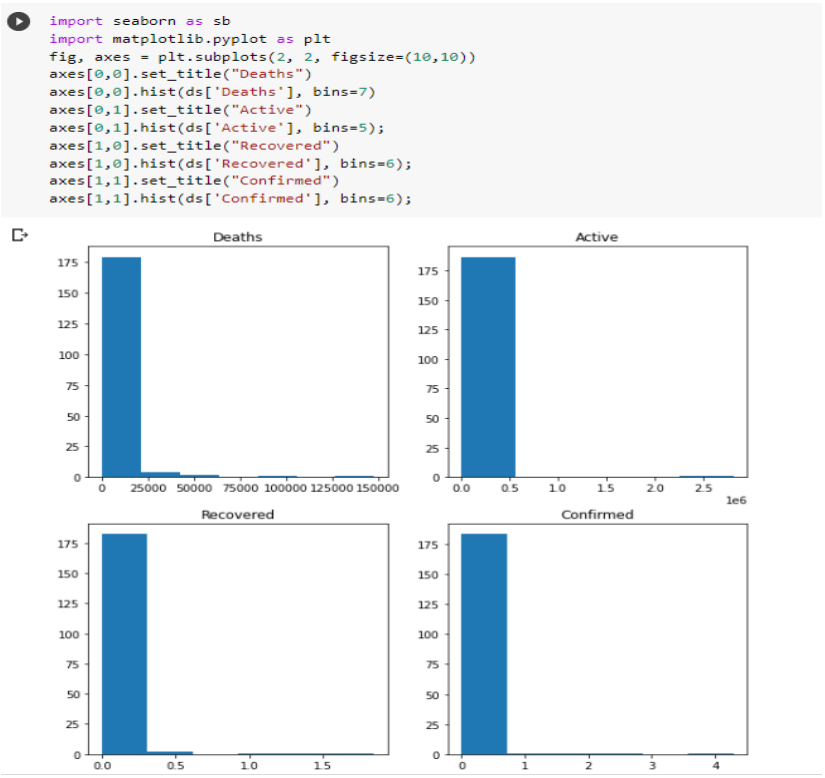
ds.hist(column="Recovered")

plt.show()

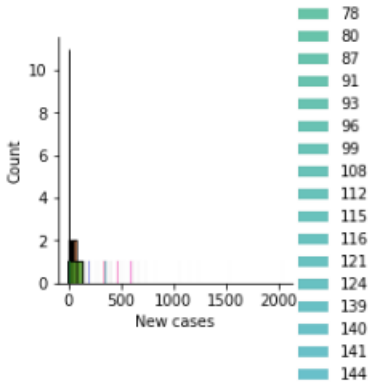
```
#CHARTS
import matplotlib.pyplot as plt
ds.hist(column="Recovered")
plt.show()
```



```
import seaborn as sb
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Deaths")
axes[0,0].hist(ds['Deaths'], bins=7)
axes[0,1].set_title("Active")
axes[0,1].hist(ds['Active'], bins=5);
axes[1,0].set_title("Recovered")
axes[1,0].hist(ds['Recovered'], bins=6);
axes[1,1].set_title("Confirmed")
axes[1,1].hist(ds['Confirmed'], bins=6);
```




```
import seaborn as sb
import matplotlib.pyplot as plt
plot = sb.FacetGrid(ds, hue="Deaths")
plot.map(sb.histplot, "Active").add_legend()
plot = sb.FacetGrid(ds, hue="Deaths")
plot.map(sb.histplot, "Recovered").add_legend()
plot = sb.FacetGrid(ds, hue="Deaths")
plot.map(sb.histplot, "Confirmed").add_legend()
plot = sb.FacetGrid(ds, hue="Deaths")
plot.map(sb.histplot, "New cases").add_legend()
plt.show()
```

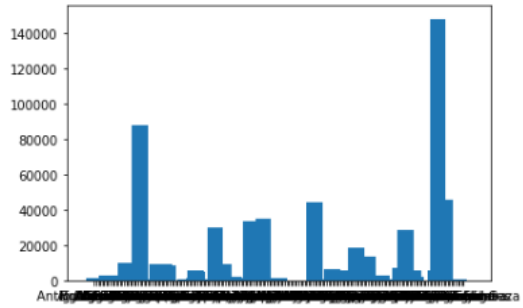


import matplotlib.pyplot as plt

plt.bar(ds["Country/Region"],ds["Deaths"],width=8)

```
import matplotlib.pyplot as plt
plt.bar(ds["Country/Region"],ds["Deaths"],width=8)
```

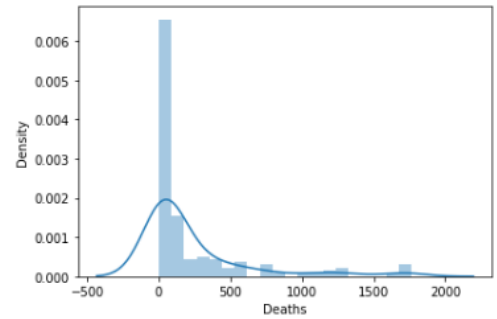
```
<BarContainer object of 187 artists>
```



sns.distplot(ds['Deaths'])

```
sns.distplot(ds['Deaths'])
```

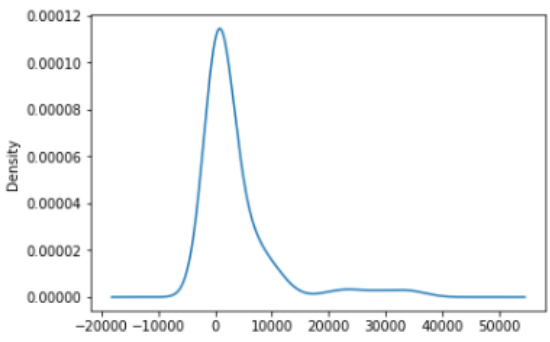
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f145c77fc10>
```



ds['Active'].plot(kind='density')

```
[ ]  ds['Active'].plot(kind='density')
```

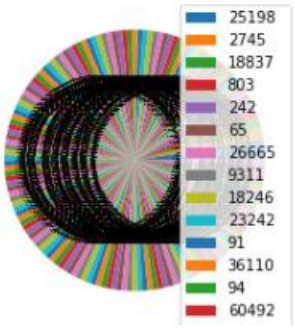
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f145d27dcd0>
```



```
plt.pie(ds['Recovered'].value_counts(),autopct='.3%f')
plt.legend(ds['Recovered'])
```

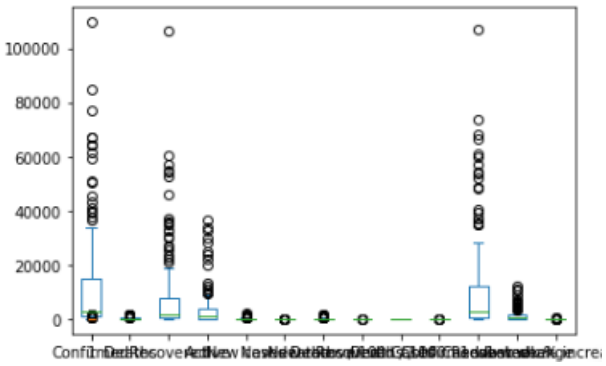
```
plt.pie(ds['Recovered'].value_counts(),autopct='.3%f')
plt.legend(ds['Recovered'])
```

```
<matplotlib.legend.Legend at 0x7f145c827190>
```



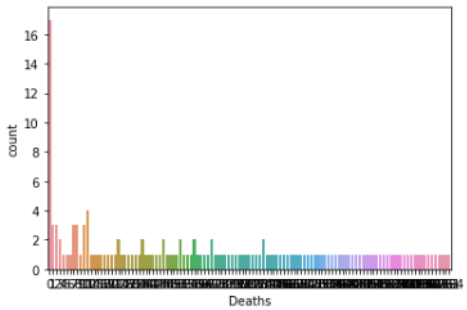

```
ds.plot.box()
plt.boxplot(ds['Deaths'])
plt.show()
```

```
ds.plot.box()
plt.boxplot(ds['Deaths'])
plt.show()
```




```
import seaborn as sb
import matplotlib.pyplot as plt
sb.countplot(x='Deaths', data=ds, )
plt.show()
```

```
import seaborn as sb
import matplotlib.pyplot as plt
sb.countplot(x='Deaths', data=ds, )
plt.show()
```



```
import seaborn as sb
import matplotlib.pyplot as plt
sb.pairplot(ds.drop(['Active'], axis = 1), hue='Deaths', height=2)
```

E. Categorical Variable Analysis

print(ds.describe(include=[‘object’])

```
print(ds.describe(include=['object']))
```

```
       Country/Region WHO Region
count             187        187
unique            187          6
top       Afghanistan     Europe
freq                1         56
```

F. Univariate Function Optimization

from scipy.optimize import minimize_scalar

import matplotlib.pyplot as plt

import numpy as np

def func(x):

    x = x**2+2*np.sin(x*np.pi)

    return x

x = ds['Deaths']

y = func(x)

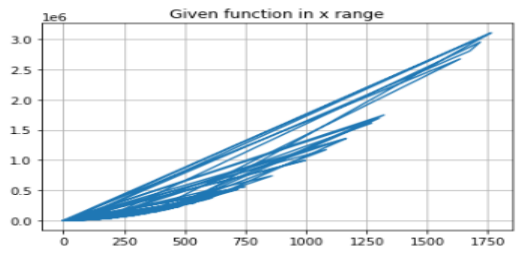
plt.title("Given function in x range")

plt.plot(x, y)

plt.grid()

plt.show()

```
#UNIVARIATE FUNCTION OPTIMISATION
from scipy.optimize import minimize_scalar
import matplotlib.pyplot as plt
import numpy as np
def func(x):
    x = x**2+2*np.sin(x*np.pi)
    return x
x = ds['Deaths']
y = func(x)
plt.title("Given function in x range")
plt.plot(x, y)
plt.grid()
plt.show()
```



result = minimize_scalar(func, method="brent")

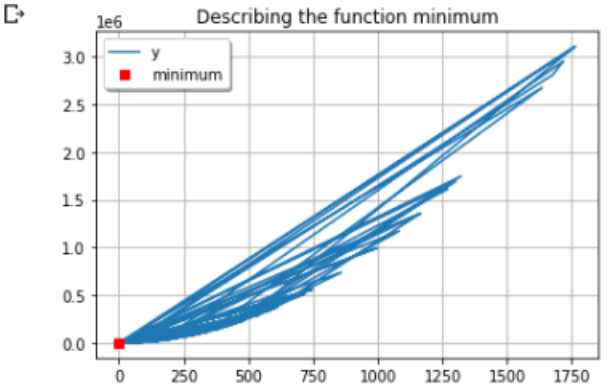
print(result)

```
result = minimize_scalar(func, method="brent")
print(result)
```

```
     fun: -1.773036468129433
    nfev: 14
     nit: 10
 success: True
       x: -0.4538535458172619
```

```
plt.title("Describing the function minimum")
plt.plot(x, y, label="y")
plt.plot(result['x'], result['fun'], 'sr', label="minimum")
plt.legend(loc='best', fancybox=True, shadow=True)
plt.grid()
plt.show()
```

```
plt.title("Describing the function minimum")
plt.plot(x, y, label="y")
plt.plot(result['x'], result['fun'], 'sr', label="minimum")
plt.legend(loc='best', fancybox=True, shadow=True)
plt.grid()
plt.show()
```

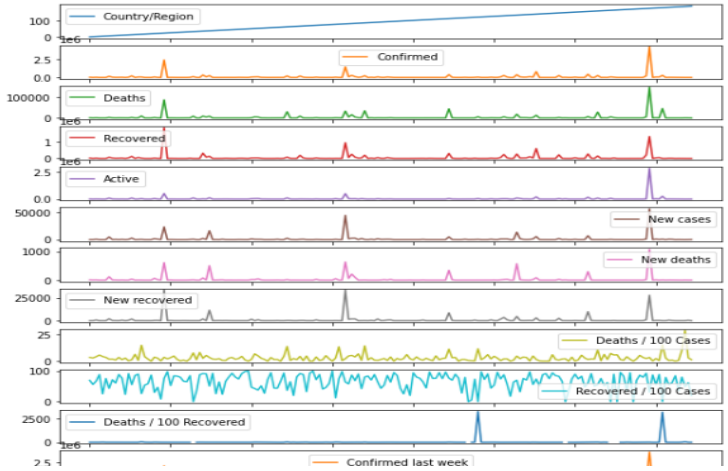


G. Perform Feature Extraction with Univariate Statistical Tests and summarize the selected features

ds.plot(subplots=True, figsize=(10, 12))

```
ds.plot(subplots=True, figsize=(10, 12))
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fee583dc5d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee5838cb90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee583dc150>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee58670e10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee583309d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee582e7d10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee58296b90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee58265390>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee582653d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee5821e810>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee5818ed90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee58156110>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee5810e450>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fee580c6790>],
      dtype=object)
```



from sklearn.datasets import make_regression

from sklearn.tree import DecisionTreeRegressor

X,y = make_regression(n_samples=1000 , n_features=10, n_informative =5 , random_state = 1)

model = DecisionTreeRegressor()

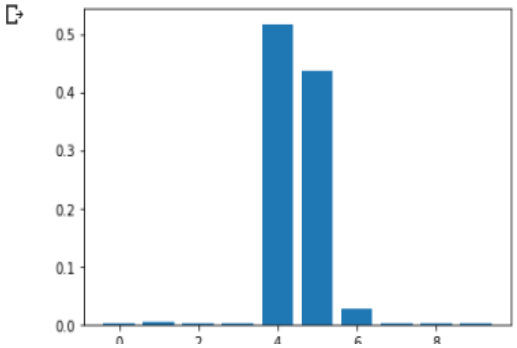
model.fit(X,y)

importance = model.feature_importances_

plt.bar([x for x in range (len (importance))],importance)

plt.show()

```
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
X,y = make_regression(n_samples=1000 , n_features=10, n_informative =5 , random_state = 1)
model = DecisionTreeRegressor()
model.fit(X,y)
importance = model.feature_importances_
plt.bar([x for x in range (len (importance))],importance)
plt.show()
```



pred = model.predict(X)

errors = abs (pred - (y))

print("Mean Absolute Error = " , round(np.mean(errors),2), 'degrees')

mape = 100 * (errors/y)

accuracy = 100 - (np.mean(mape))

print("Accuracy :" , round (accuracy , 2),"%")

```
pred = model.predict(X)
errors = abs (pred - (y))
print("Mean Absolute Error = " , round(np.mean(errors),2), 'degrees')
mape = 100 * (errors/y)
accuracy = 100 - (np.mean(mape))
print("Accuracy :" , round (accuracy , 2),"%")
```

```
Mean Absolute Error =  0.0 degrees
Accuracy : 100.0 %
```

Result:

Thus, univariate analysis was performed on the datasets.

Date: 29.03.22 EXP: 2B	UNIVARIATE TIME SERIES ANALYSIS

AIM:

To COVID Dataset, perform the following

- Create time series features from the dataset you imported.
- Plot feature importance
- Forecast and evaluate the data

SOURCE CODE:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from google.colab import files

uploaded = files.upload()

import io

ds=pd.read_csv(io.BytesIO(uploaded['cov_time.csv']),parse_dates=True,index_col="ObservationDate")

ds.shape

ds.head()

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

```
[ ] from google.colab import files
    uploaded = files.upload()
    import io
    ds=pd.read_csv(io.BytesIO(uploaded['cov_time.csv']),parse_dates=True,index_col="ObservationDate")
    ds.shape
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current brov
rerun this cell to enable.
Saving cov_time.csv to cov_time (1).csv
(306429, 7)
```

```
ds.head()
```

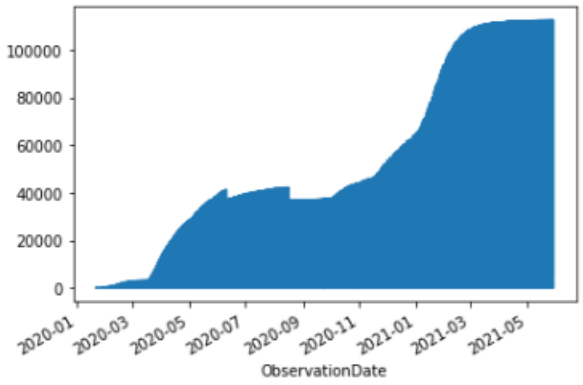
ObservationDate	SNo	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
2020-01-22	1	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
2020-01-22	2	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2020-01-22	3	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
2020-01-22	4	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
2020-01-22	5	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

A. Create time series features from the dataset you imported.

ds['Deaths'].plot()

```
ds['Deaths'].plot()
```

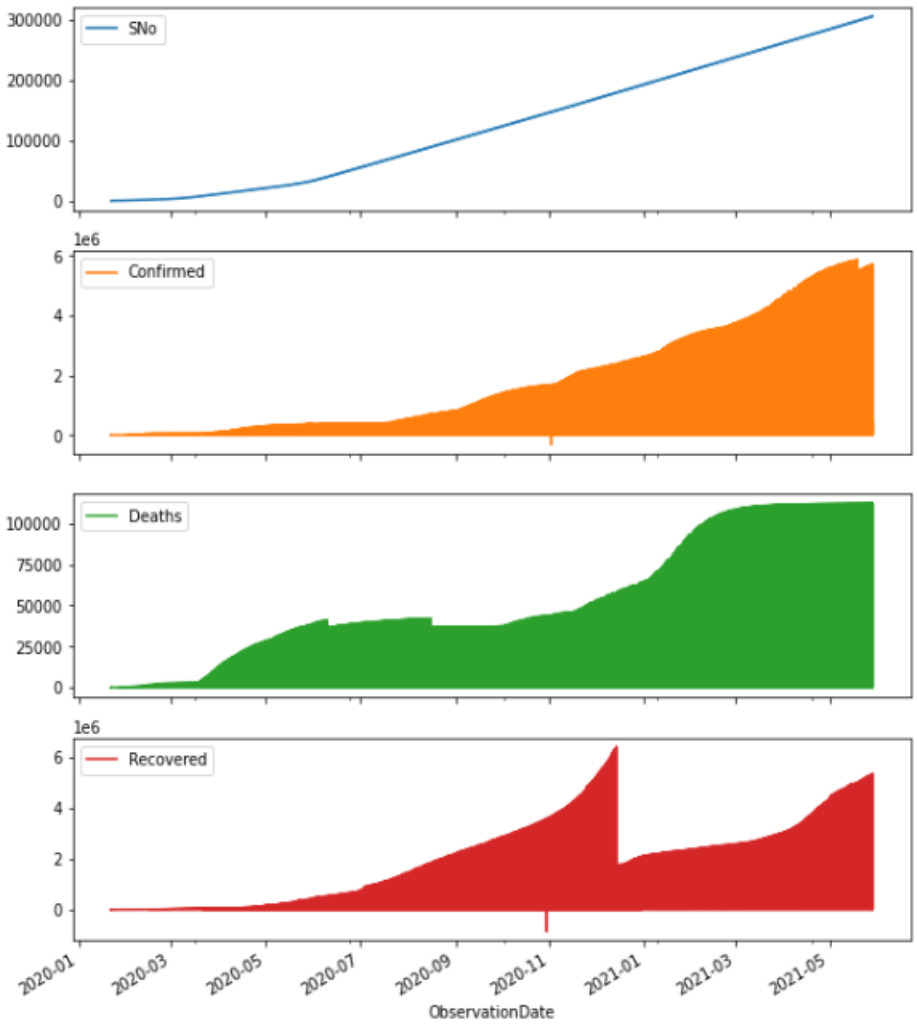
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4831ce610>
```



ds.plot(subplots=True, figsize=(10, 12))

```
ds.plot(subplots=True, figsize=(10, 12))
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fd4832a39d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd4805c9450>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd4831dc3d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd480539b10>],
      dtype=object)
```



Resampling the time series data based on monthly 'M' frequency

ds_month = ds.resample("M").mean()

using subplot

fig, ax = plt.subplots(figsize=(10, 6))

plotting bar graph

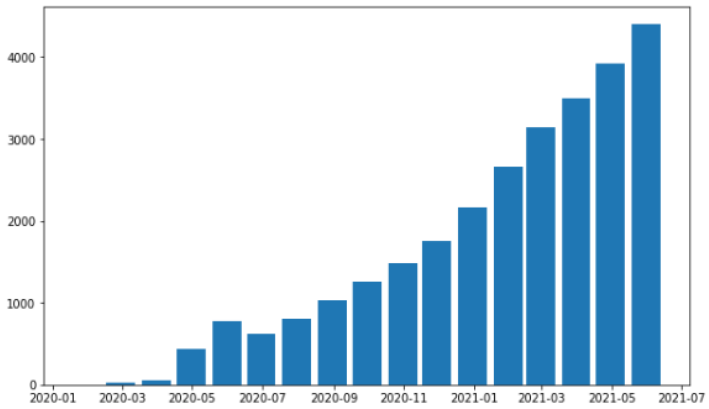
ax.bar(ds_month['2020':].index,

ds_month.loc['2020':, "Deaths"],

width=25, align='center')

```
# Resampling the time series data based on monthly 'M' frequency
ds_month = ds.resample("M").mean()
# using subplot
fig, ax = plt.subplots(figsize=(10, 6))
# plotting bar graph
ax.bar(ds_month['2020':].index,
       ds_month.loc['2020':, "Deaths"],
       width=25, align='center')
```

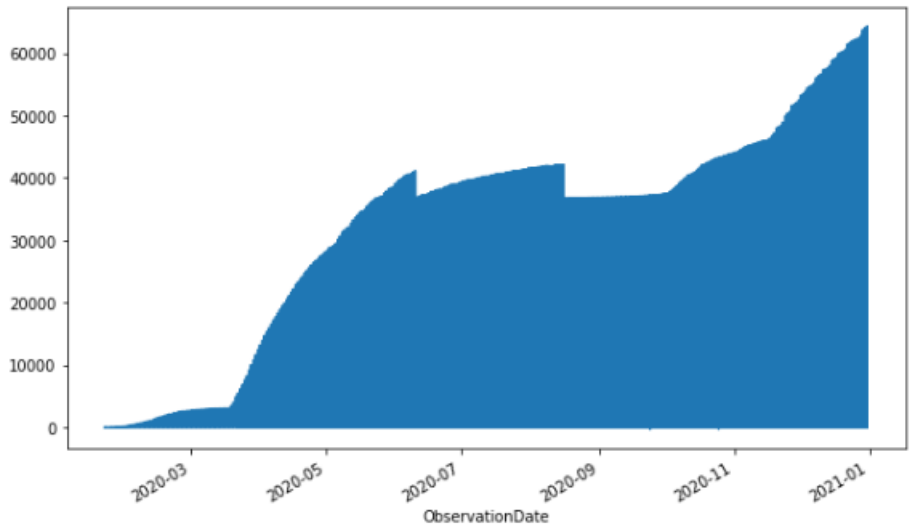
```
<BarContainer object of 17 artists>
```



ds['2020']['Deaths'].plot(figsize=(10, 6))

```
[ ] ds['2020']['Deaths'].plot(figsize=(10, 6))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Ir
  """Entry point for launching an IPython kernel.
<matplotlib.axes._subplots.AxesSubplot at 0x7fd47dc36cd0>
```

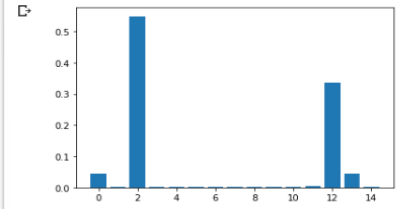


B. Plot feature importance

```
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
X, y = make_regression(n_samples=1000, n_features=15, n_informative=5, random_state=1)
model = DecisionTreeRegressor()
# fit the model
model.fit(X, y)
# get importance
importance = model.feature_importances_
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

Plot Feature Importance

```
[10] from sklearn.datasets import make_regression
     from sklearn.tree import DecisionTreeRegressor

[11] X, y = make_regression(n_samples=1000, n_features=15, n_informative=5, random_state=1)
     model = DecisionTreeRegressor()
     # fit the model
     model.fit(X, y)
     # get importance
     importance = model.feature_importances_

     plt.bar([x for x in range(len(importance))], importance)
     plt.show()
```



C. Forecast and evaluate the data

```
pred = model.predict(X)
errors = abs(pred - y)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
mape = 100 * (errors / y)# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
[ ] pred = model.predict(X)
    errors = abs(pred - y)
    print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
    mape = 100 * (errors / y)# Calculate and display accuracy
    accuracy = 100 - np.mean(mape)
    print('Accuracy:', round(accuracy, 2), '%.')

    Mean Absolute Error: 0.0 degrees.
    Accuracy: 100.0 %.
```

RESULT:

Thus, univariate analysis on time series dataset was done.

Date: 29/3/22 EXP: 3a	OPENSTACK

AIM:

To understand OpenStack deployment, its implementation and its applications.

THEORY:

OpenStack:

OpenStack is an open source platform that uses pooled virtual resources to build and manage private and public clouds. The tools that comprise the OpenStack platform, called "projects," handle the core cloud-computing services of compute, networking, storage, identity, and image services. More than a dozen optional projects can also be bundled together to create unique, deployable clouds.In virtualization, resources such as storage, CPU, and RAM are abstracted from a variety of vendor-specific programs and split by a hypervisor before being distributed as needed. OpenStack uses a consistent set of application programming interfaces (APIs) to abstract those virtual resources 1 step further into discrete pools used to power standard cloud computing tools that administrators and users interact with directly.

How does OpenStack work:

OpenStack is essentially a series of commands known as scripts. Those scripts are bundled into packages called projects that relay tasks that create cloud environments. In order to create those environments, OpenStack relies on 2 other types of software:

- Virtualization that creates a layer of virtual resources abstracted from hardware
- A base operating system (OS) that carries out commands given by OpenStack scripts

OpenStack components:

OpenStack's architecture is made up of numerous open source projects. These projects are used to set up OpenStack's undercloud and overcloud—used by sys admins and cloud users, respectively. Underclouds contain the core components sys admins need to set up and manage end users' OpenStack environments, known as overclouds.

There are 6 stable, core services that handle, compute, networking, storage, identity, and images while more than a dozen optional ones vary in developmental maturity. Those 6 core services are the infrastructure that allows the rest of the projects to handle dashboarding, orchestration, bare-metal provisioning, messaging, containers, and governance.

1. Nova : Nova is a full management and access tool to OpenStack compute resources—handling scheduling, creation, and deletion.

2. Neutron : Neutron connects the networks across other OpenStack services.

3. Swift : Swift is a highly fault-tolerant object storage service that stores and retrieves unstructured data objects using a RESTful API.

4. Cinder : Cinder provides persistent block storage accessible through a self-service API.

5. Keystone : Keystone authenticates and authorizes all OpenStack services. It's also the endpoint catalog for all services.

6. Glance : Glance stores and retrieves virtual machine disk images from a variety of locations.

OpenStack Deployment:

It is mostly deployed as infrastructure as service in both public and private cloud where virtual services and other resources are made available to users. The software platform consists of interrelated components that control diverse, multivendor hardware lifecycle management tools and packages to help install and maintain the lifecycle of deployments.

Applications Of OpenStack:

- Private Cloud
- Public Cloud
- Containers
- Network Functions Virtualization

RESULT:

Thus, OpenStack was studied.

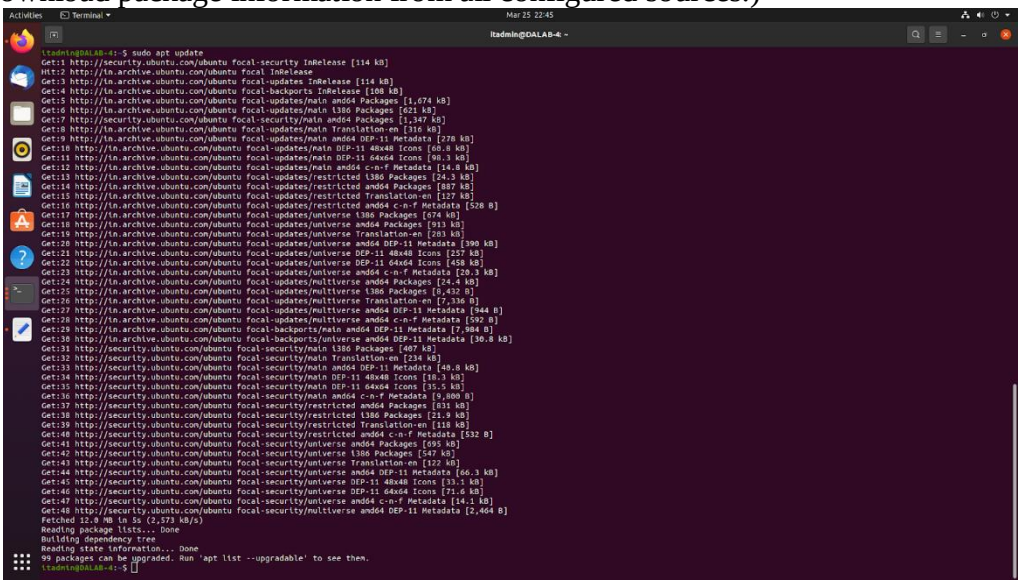
Date: 05/04/22 EXP: 3b	OPENSTACK INSTALLATION

AIM:

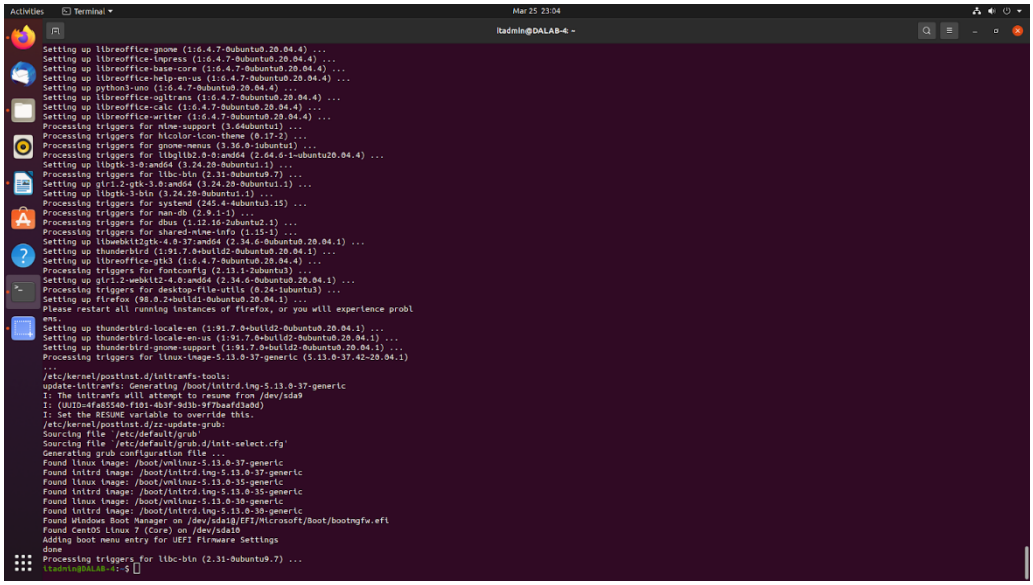
To implement infrastructure as Service using Open Stack.

STEPS:

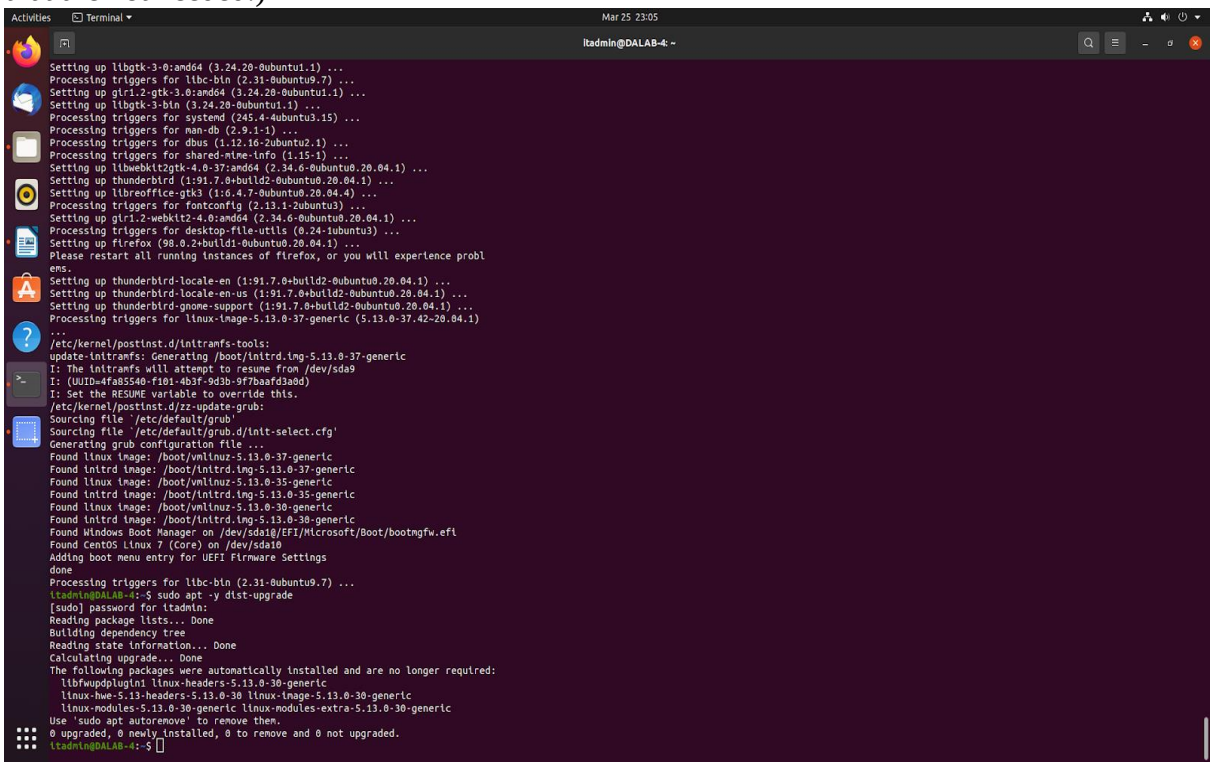
1. Login to sudo ("superuser do") user
2. Open Terminal
3. In Terminal type : sudo apt – update
   (This download package information from all configured sources.)



4. sudo apt -y upgrade
   (upgrade everything in your system, all the packages, and the kernel to the latest versions as supported by your repositories)



5. sudo apt -y dist-upgrade
   (It updates existing packages, installs new dependencies that are not in the system, and deletes those that are not needed.)



6. sudo useradd -s /bin/bash -d /opt/stack -m stack
   (add user called stack to run DevStack)
   DevStack should be run as a non-root user with sudo enabled.

```
(base) student@DALAB-4:~/Desktop$ sudo useradd -s /bin/bash -d /opt/stack -m stack
[sudo] password for student:
student is not in the sudoers file.  This incident will be reported.
(base) student@DALAB-4:~/Desktop$ su - itadmin
Password:
itadmin@DALAB-4:~$ sudo useradd -s /bin/bash -d /opt/stack -m stack
[sudo] password for itadmin:
itadmin@DALAB-4:~$
```

7. echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

(Since this user will be making many changes to your system, it should have sudo privileges)

```
itadmin@DALAB-4:~$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD: ALL
itadmin@DALAB-4:~$
```

8. sudo su - stack
   (Once you have created the stack user, it's time to log in to it)

```
itadmin@DALAB-4:~$ sudo su - stack
stack@DALAB-4:~$ sudo su -
```

9. sudo apt -y install git

(install git)

```
stack@DALAB-4:~$ sudo apt -y install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 linux-headers-5.13.0-30-generic
  linux-hwe-5.13-headers-5.13.0-30 linux-image-5.13.0-30-generic
  linux-modules-5.13.0-30-generic linux-modules-extra-5.13.0-30-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,465 kB of archives.
After this operation, 38.4 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all 1:2.25.1-1ubuntu3.2 [884 kB]
9% [2 git-man 114 kB/884 kB 13%]
pdates/main amd64 git amd64 1:2.25.1-1ubuntu3.2 [4,554 kB]
Fetched 5,465 kB in 4min 3s (22.5 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 213052 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.25.1-1ubuntu3.2_all.deb ...
Unpacking git-man (1:2.25.1-1ubuntu3.2) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.2_amd64.deb ...
Unpacking git (1:2.25.1-1ubuntu3.2) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.25.1-1ubuntu3.2) ...
Setting up git (1:2.25.1-1ubuntu3.2) ...
Processing triggers for man-db (2.9.1-1) ...
```

10. git clone https://git.openstack.org/openstack-dev/devstack

(The devstack repo contains a script that installs OpenStack and templates for configuration files. Download/clone devstack from its repository to your system)

```
stack@DALAB-4:~$ git clone https://github.com/openstack-dev/devstack.git
Cloning into 'devstack'...
remote: Enumerating objects: 48499, done.
remote: Counting objects: 100% (1725/1725), done.
remote: Compressing objects: 100% (686/686), done.
remote: Total 48499 (delta 1177), reused 1444 (delta 1033), pack-reused 46774
Receiving objects: 100% (48499/48499), 15.48 MiB | 4.31 MiB/s, done.
Resolving deltas: 100% (33808/33808), done.
```

11. cd devstack
    (Downloaded DevStack and need to setup our configuration files for it. Need to first navigate to the devstack folder, by running)

```
stack@DALAB-4:~$ cd devstack
stack@DALAB-4:~/devstack$
```

```
stack@DALAB-4:~$ cd devstack
stack@DALAB-4:~/devstack$ nano local.conf
stack@DALAB-4:~/devstack$ vi local.conf
```

12. vi local.conf

Inside local.conf add the following:

[[local|localrc]]

 # Password for KeyStone, Database, RabbitMQ and Service

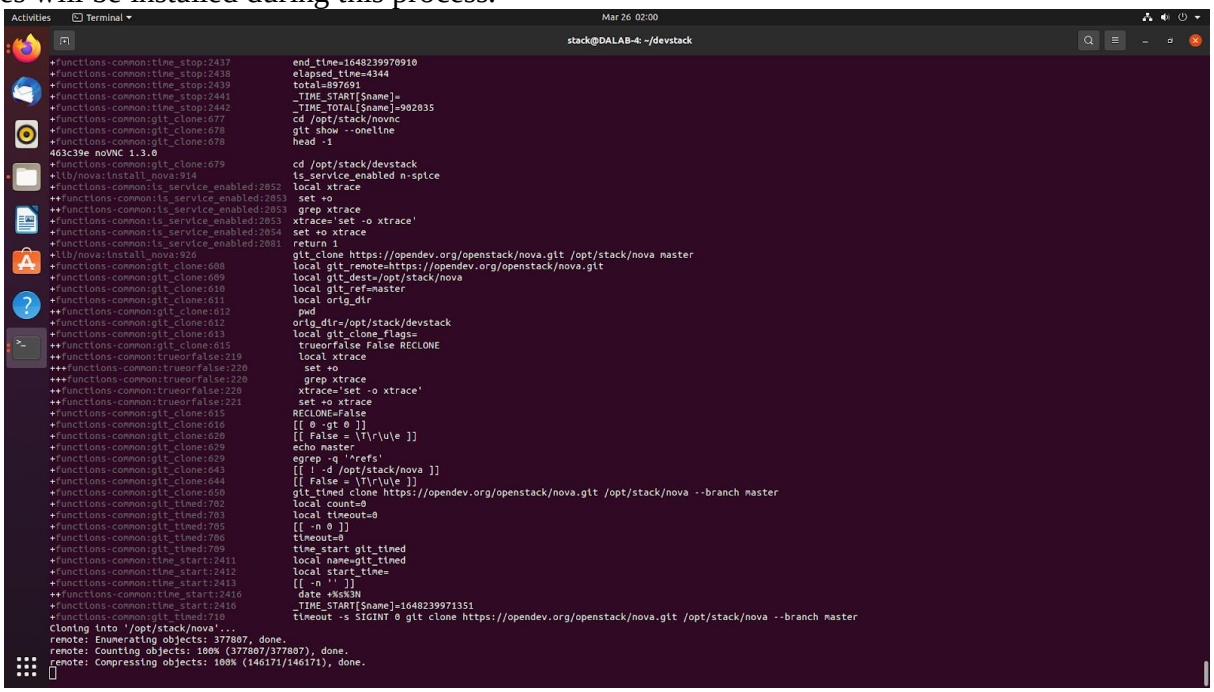
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

Once the above command is pasted/added in local.conf file press Esc key and type :wq to write it /save it and quit

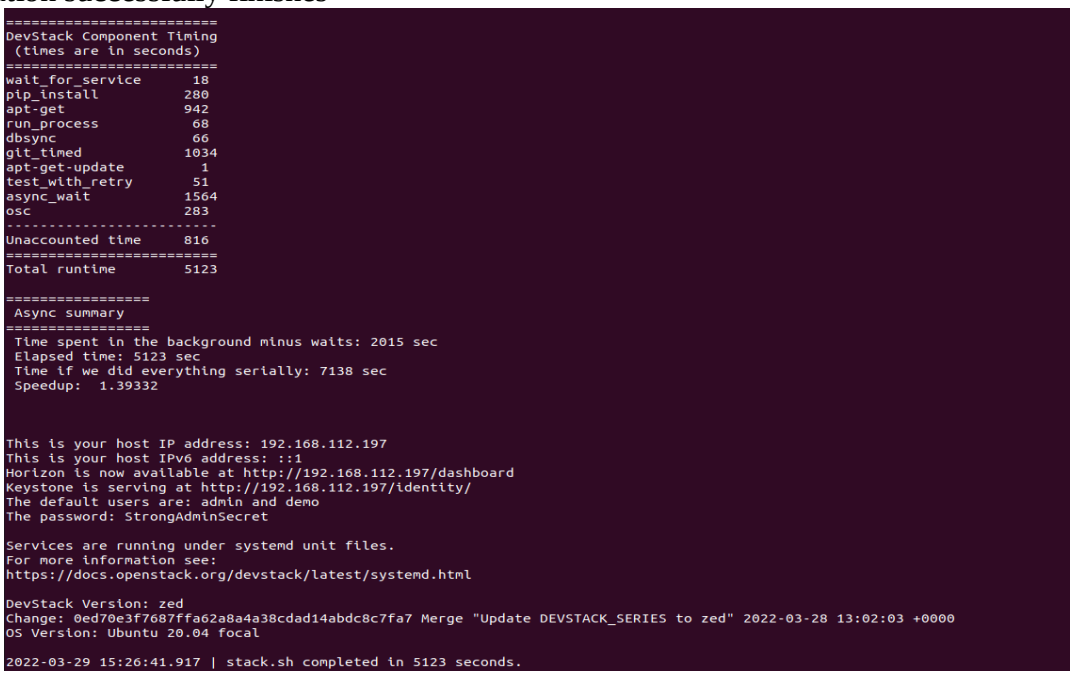
Run this script to setup OpenStack

./stack.sh

This will take a 1 - 2 hours, largely depending on the speed of your internet connection. Many git trees and packages will be installed during this process.



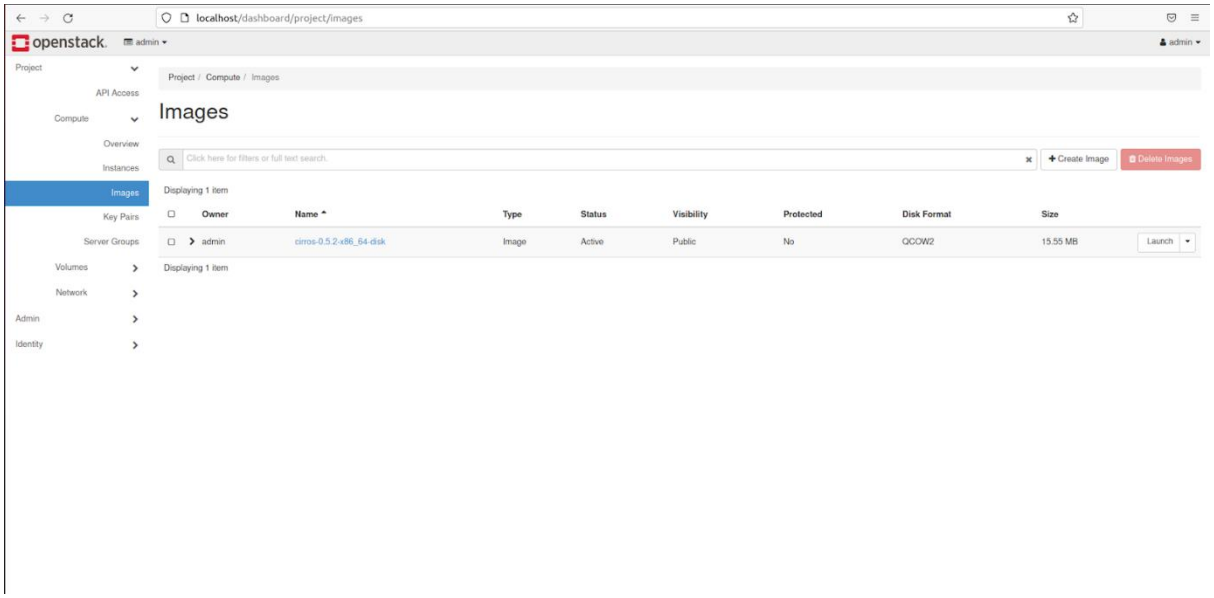
After installation successfully finishes



Since we have successfully setup OpenStack using Devstack, let's access it via our browser.

http://localhost/dashboard

Enter the credentials. We can also log in as admin here, by having User Name as admin & for Password using the one we added to local.conf file.



RESULT:

Thus, OpenStack was installed successfully.

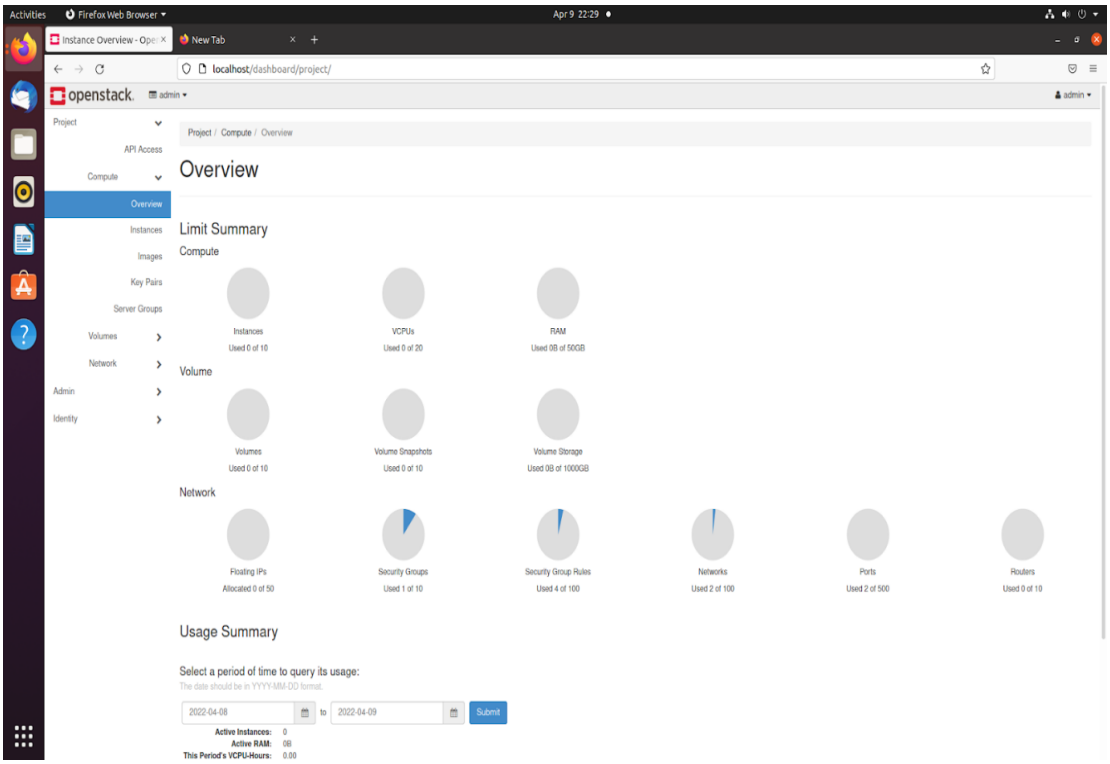
Date: 09/04/22 EXP: 4	CREATION OF VMS IN OPENSTACK

AIM:

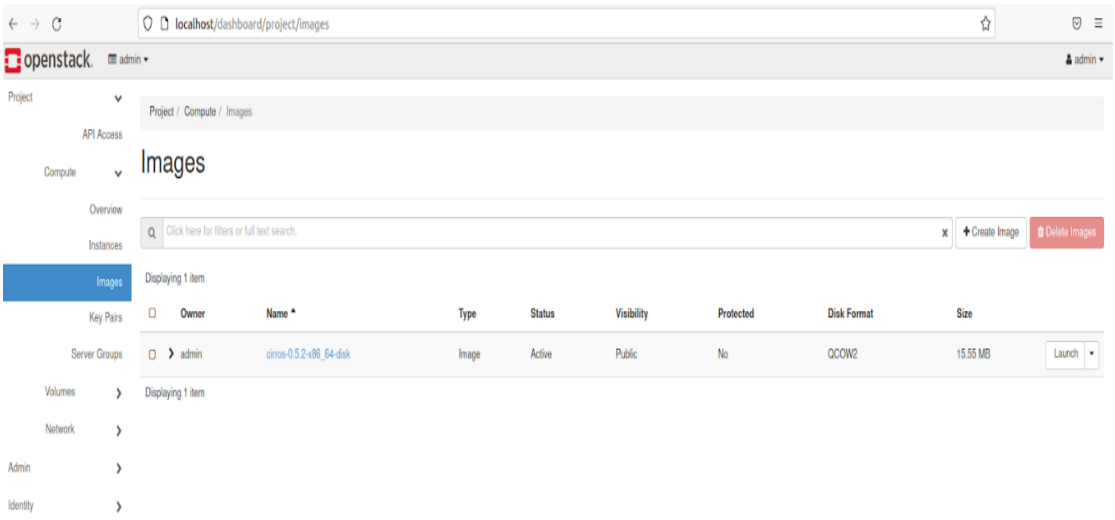
To create VM in OpenStack and execute a simple application in OpenStack.

STEPS:

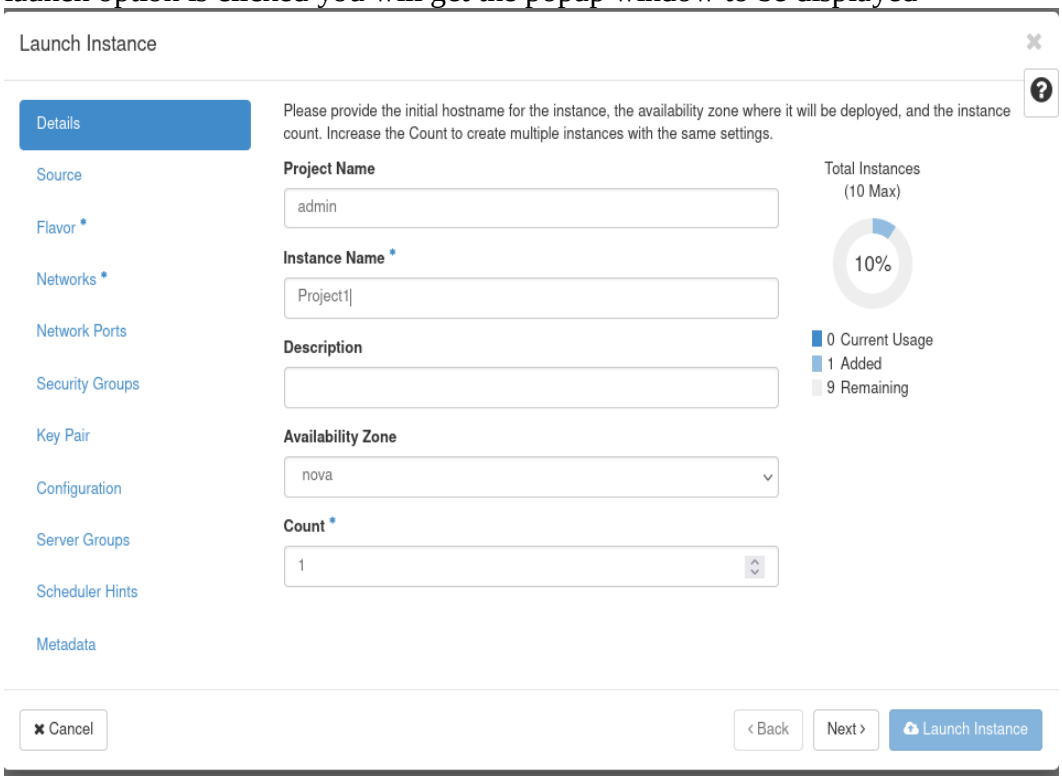
1. Once you login in dashboard with credentials you will get screen like the following



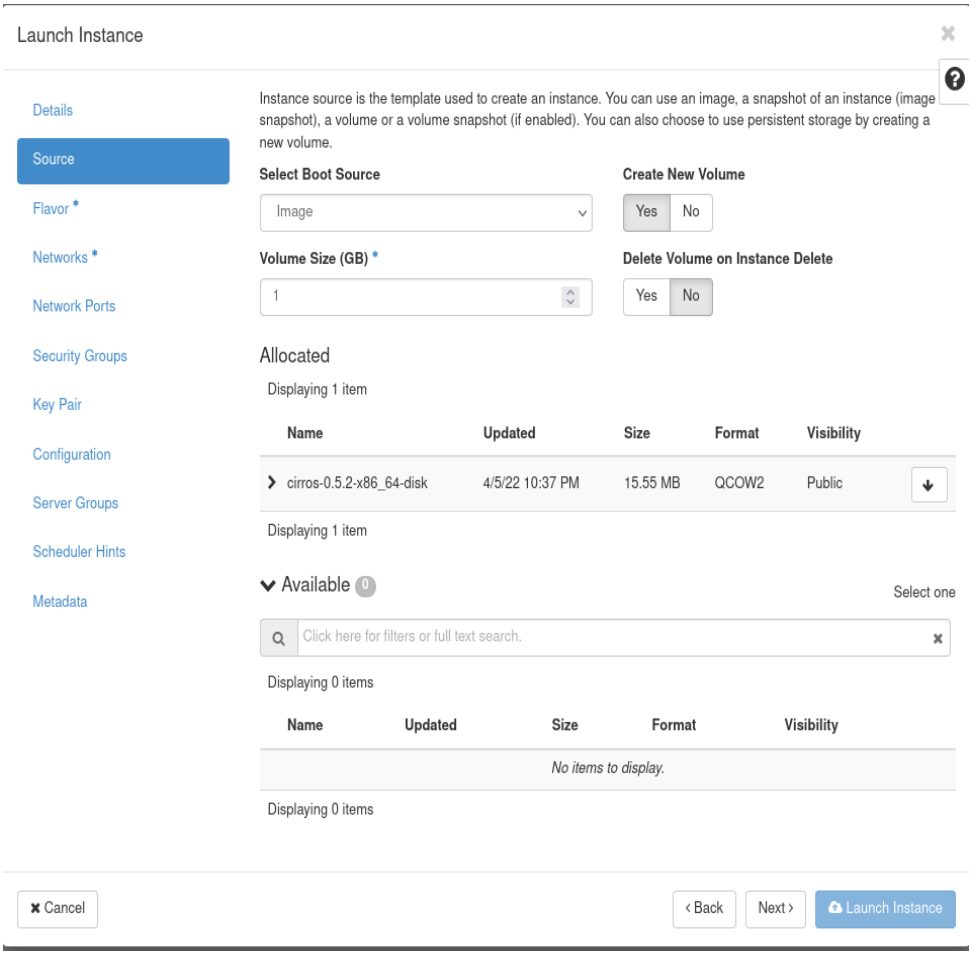
2. Launch an Existing Image Instance in OpenStack



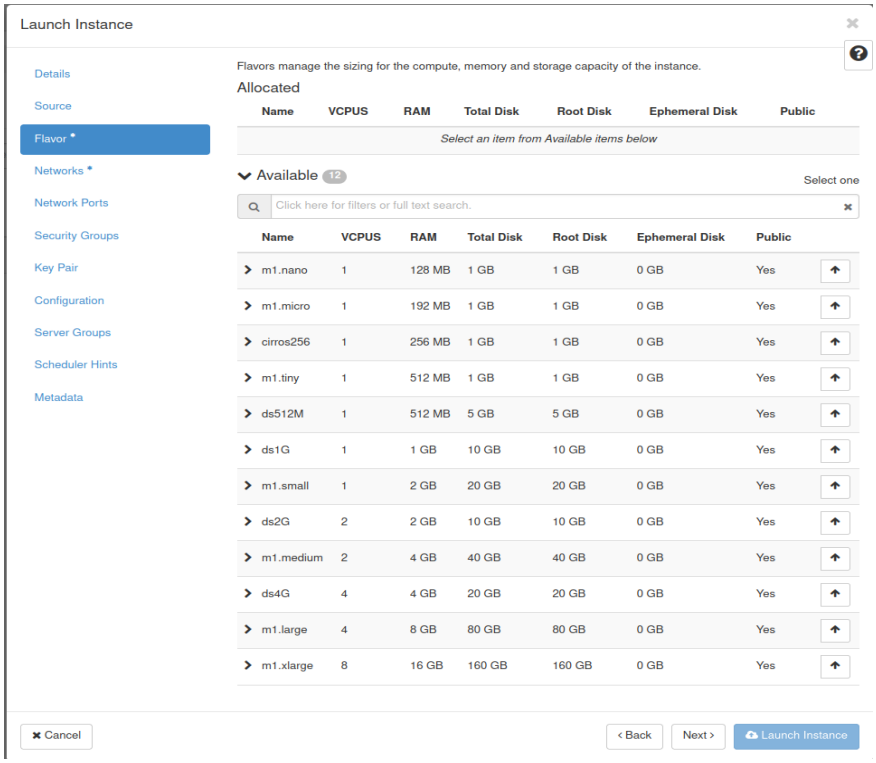
3. Once launch option is clicked you will get the popup window to be displayed



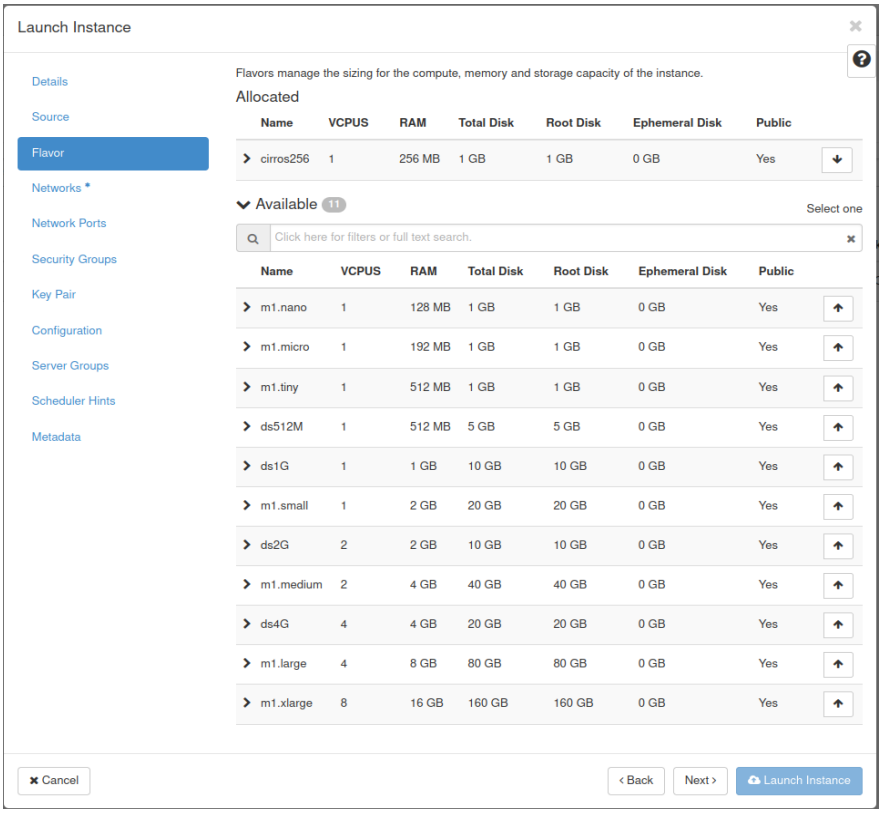
4. Give instance name of your choice rest all fields leave it as default values and click next
5. Source tab will be next screen in the launch instance window where storage is created and click next



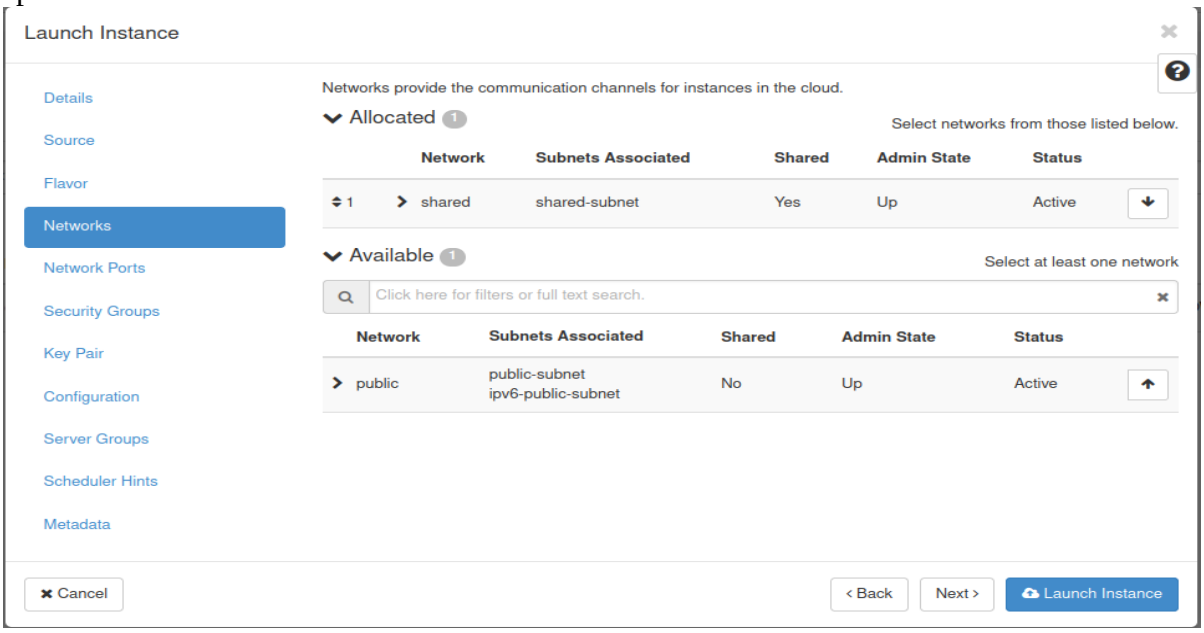
6. Allocate the virtual machine resources by adding a flavor best suited for your needs and click on **Next** to move on.



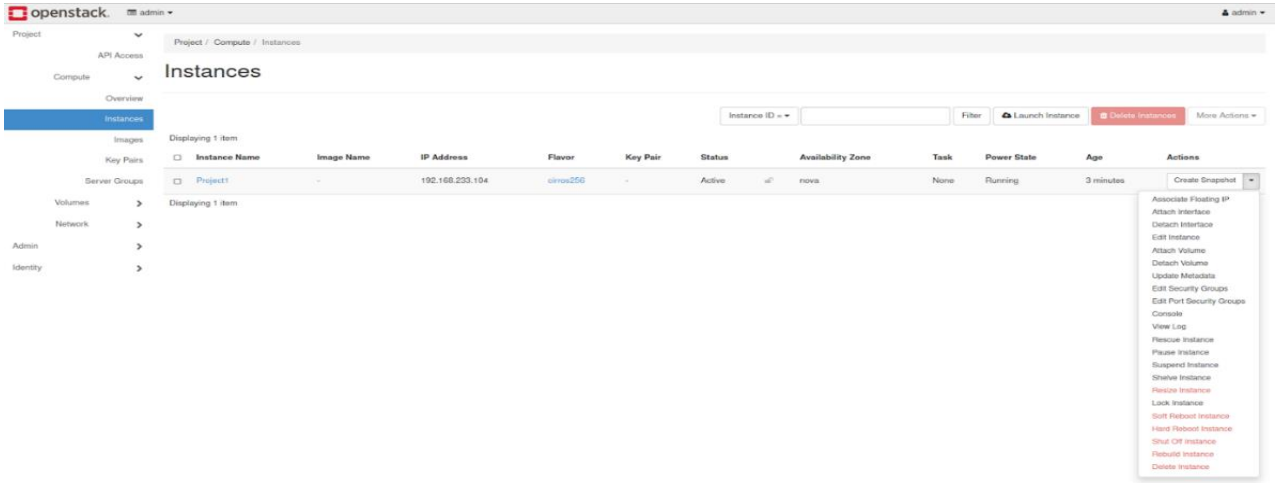
7. From the available instance add the instance which we already launched (by clicking on the upward arrow mark) so the instance move under allocated.



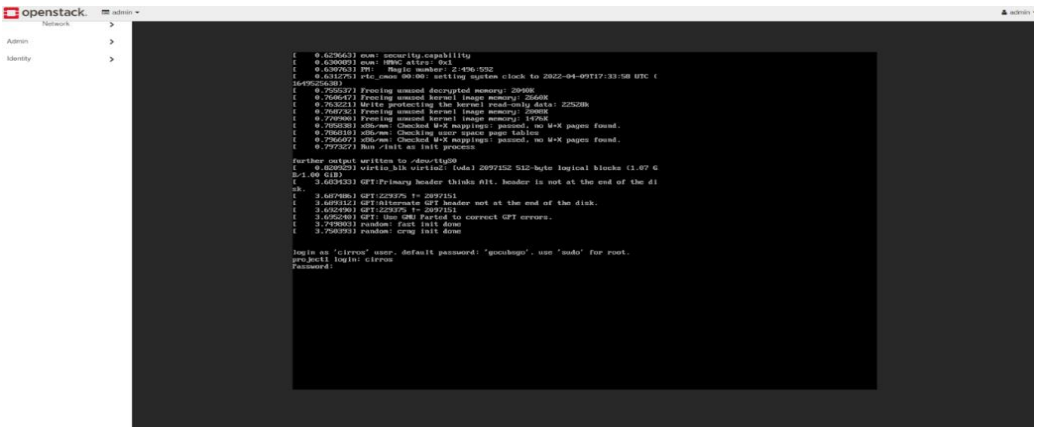
8. Finally, add the shared network from the available networks in openstack to your instance using the upward arrow mark button and hit on **Launch Instance** to start the virtual machine.



9. Once the launch instance is clicked. In the right tab click instance you see the instance added under it.



10. Click console

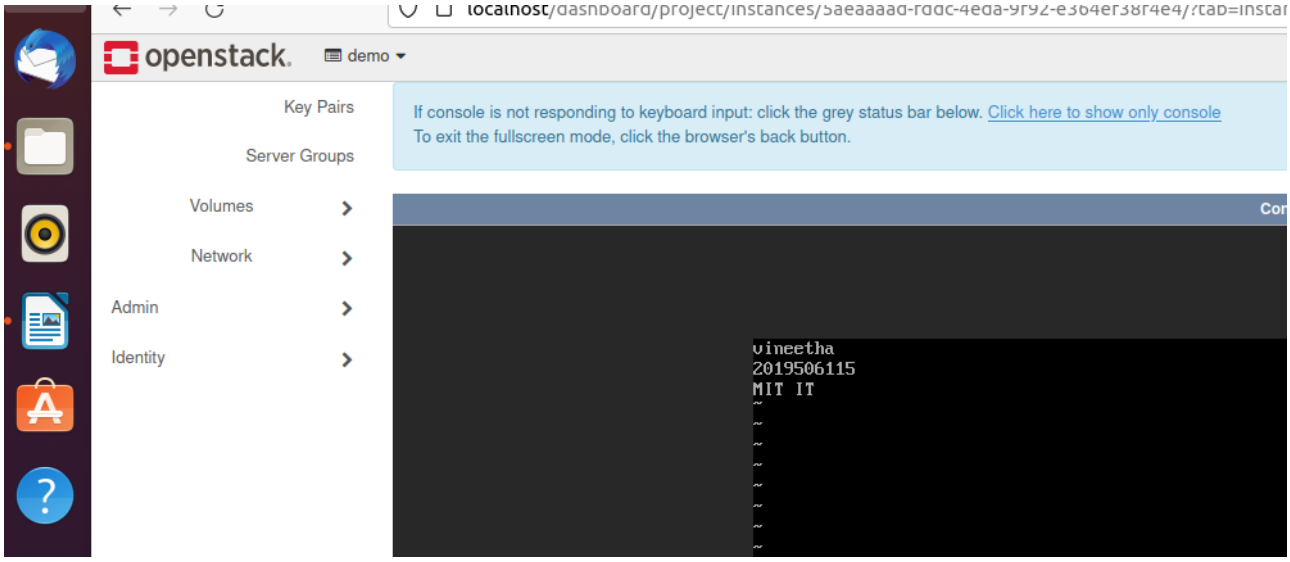


11. In console window

Create a text file using the following command:

vi test.txt

a) File will be opened
b) To enable to writable format press i or insert key
c) Type some content .
d) To save it, press Escape key and then press shift key type :wq
e) Now in console window type ls (this will list the files already available)
f) To display the text content in file use the command
g) Cat filename.extension
h) cat test.txt



RESULT:

Thus, VM was created in OpenStack and a simple application was executed.

Date: 12/4/22 EXP: 5	MONGODB- BASICS

1) AIM:

To perform Study experiment on MongoDB.

THEORY:

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

i. Database : Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
ii. Collection : Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
iii. Document : A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.
iv. _id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide _id while inserting the document. If you don't provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

Advantages of MongoDB over RDBMS:

- **Schema less** − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- **Ease of scale-out** − MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

Why Use MongoDB:

- **Document Oriented Storage** − Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

Where to Use MongoDB:

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

RESULT:

Thus, MongoDB basics were studied.

2)AIM:

To pick any system , create database and its corresponding tables, also populate the tables with data. Apply insert /search operations.

SOURCE CODE:

1. Show dbs

```
test> db
test
test> show databases
admin      41 kB
config   61.4 kB
local      41 kB
test> use dbb
switched to db dbb
dbb> db
dbb
dbb> show databases
admin      41 kB
config   61.4 kB
local      41 kB
```

2. Db.createCollection(“Grade”)

```
dbb> db.createCollection("Grade",{capped:true,size:10,max:10})
{ ok: 1 }
dbb> show databases
admin     41 kB
config   111 kB
dbb     49.2 kB
local     41 kB
```

3. db.Grade.insert()

```
dbb> db.grade.insertOne({"Name":"Thanu","Age":20,"CGPA":9.87})
{
  acknowledged: true,
  insertedId: ObjectId("6255e150f12516a28494779a")
}
dbb> db.grade.insert({"Name":"Thanuja","Age":21,"CGPA":8})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6255e173f12516a28494779b") }
}
```

4. db.Grade.insertMany()

```
dbb> db.grade.insertMany([{"Name":"Vijju","Age":21,"CGPA":9},{"Name":"Divya","Age":20,"CGPA":8.7}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6255e22af12516a28494779c"),
    '1': ObjectId("6255e22af12516a28494779d")
  }
}
```

5. db.Grade.find()

```
dbb> db.grade.find()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e173f12516a28494779b"),
    Name: 'Thanuja',
    Age: 21,
    CGPA: 8
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 21,
    CGPA: 9
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
```

```
dbb> db.grade.find().pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e173f12516a28494779b"),
    Name: 'Thanuja',
    Age: 23,
    CGPA: 8
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
```

6. db.Grade.find ()

```
dbb> db.grade.find({$and:[{"CGPA":{$gt:8.8}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  }
]
dbb> db.grade.find({$or:[{"CGPA":{$lt:8.8}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
```

```
dbb> db.grade.find({$and:[{"CGPA":{$gt:8.8}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  }
]
dbb> db.grade.find({$or:[{"CGPA":{$lt:8.8}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
dbb> db.grade.find({$nor:[{"CGPA":{$lt:8.8}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  }
]
dbb> db.grade.find({"CGPA":{$not:{$lt:8.8}}}).pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  }
]
```

7. db.Grade.update ()

```
dbb> db.grade.update({"Name":"Vijju"},{$set:{"Age":23}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbb> db.grade.find().pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e173f12516a28494779b"),
    Name: 'Thanuja',
    Age: 23,
    CGPA: 8
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
```

8. db.Grade.remove()

```
dbb> db.grade.remove({"Name":"Thanuja"})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
dbb> db.grade.find().pretty()
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
```

9. Aggregate

```
dbb> db.grade.aggregate([{$match:{"Age":{$gt:21}}}]).pretty()
[
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  }
]
dbb> db.grade.aggregate([{'$sort':{'Age':1}}])
[
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  },
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  }
]
```

```
dbb> db.grade.aggregate([{'$sort':{'Age':-1}}])
[
  {
    _id: ObjectId("6255e22af12516a28494779c"),
    Name: 'Vijju',
    Age: 23,
    CGPA: 9
  },
  {
    _id: ObjectId("6255e150f12516a28494779a"),
    Name: 'Thanu',
    Age: 20,
    CGPA: 9.87
  },
  {
    _id: ObjectId("6255e22af12516a28494779d"),
    Name: 'Divya',
    Age: 20,
    CGPA: 8.7
  }
]
```

10. Other functions:

```
dbb> var mapfunction=function(){emit(this.age,this.grade)}

dbb> var reducefunction=function(key,values){return Array.sum(values)}

dbb> db.grade.mapReduce(mapfunction,reducefunction,{'out':'Result'})
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'Result', ok: 1 }
dbb> db.result.find()

dbb> db.grade.distinct("Name")
[ 'Divya', 'Thanu', 'Vijju' ]
dbb> db.grade.distinct("Age")
[ 20, 23 ]
dbb> db.grade.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
3
dbb> db.grade.countDocuments()
3
```

RESULT:

Thus, basic operations were executed.

Date: 19/4/22 EXP: 6	BIVARIATE ANALYSIS

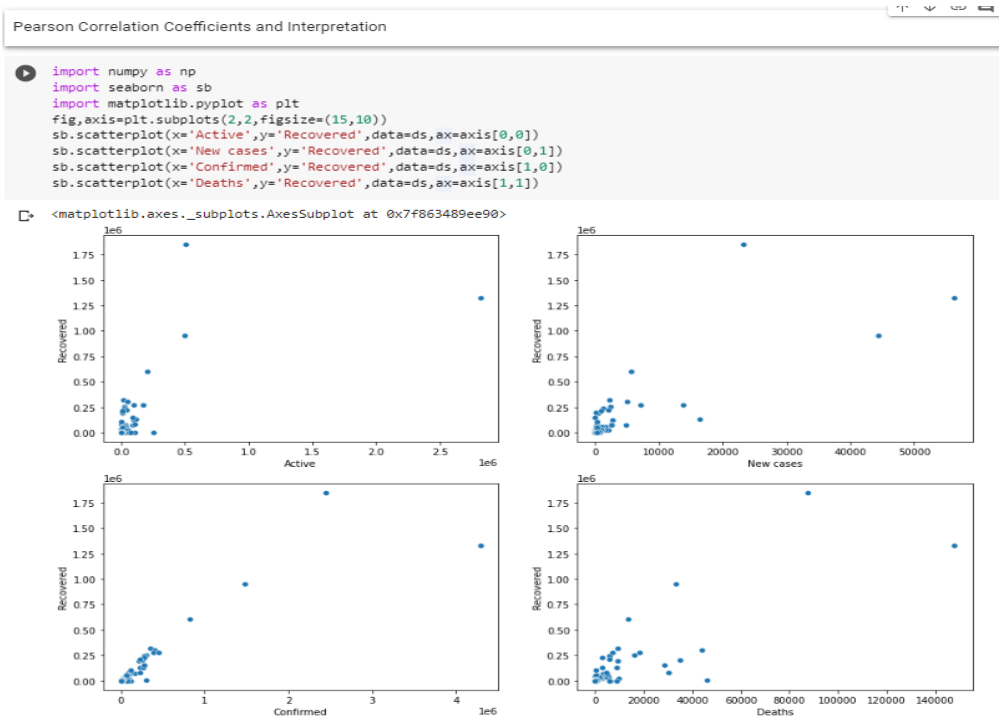
AIM:

To perform bivariate analysis on the COVID dataset.

SOURCE CODE:

a. Pearson Correlation Coefficients and Interpretation

```
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
fig,axis=plt.subplots(2,2,figsize=(15,10))
sb.scatterplot(x='Active',y='Recovered',data=ds,ax=axis[0,0])
sb.scatterplot(x='New cases',y='Recovered',data=ds,ax=axis[0,1])
sb.scatterplot(x='Confirmed',y='Recovered',data=ds,ax=axis[1,0])
sb.scatterplot(x='Deaths',y='Recovered',data=ds,ax=axis[1,1])
```



b. Simple Linear Regression

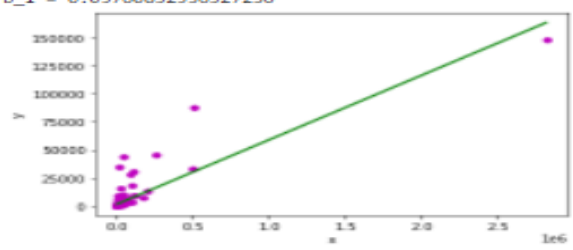
```
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)
    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)
    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)
def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
            marker = "o", s = 30)
    # predicted response vector
    y_pred = b[0] + b[1]*x
    # plotting the regression line
    plt.plot(x, y_pred, color = "g")
    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')
    # function to show plot
    plt.show()
```

```
def main():
    # observations / data
    x = ds['Active']
    y = ds['Deaths']
    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {}  \
        \nb_1 = {}".format(b[0], b[1]))
    # plotting regression line
    plot_regression_line(x, y, b)
if __name__ == "__main__":
    main()
```

Simple Linear Regression

```
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)
    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)
    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)
def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
                marker = "o", s = 30)
    # predicted response vector
    y_pred = b[0] + b[1]*x
    # plotting the regression line
    plt.plot(x, y_pred, color = "g")
    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')
    # function to show plot
    plt.show()
def main():
    # observations / data
    x = ds['Active']
    y = ds['Deaths']
    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {}  \
        \nb_1 = {}".format(b[0], b[1]))
    # plotting regression line
    plot_regression_line(x, y, b)
if __name__ == "__main__":
    main()
```

```
Estimated coefficients:
b_0 = 1538.724134366486
b_1 = 0.05760832536327236
```



c. Chi-square test

from scipy.stats import chisquare

chisquare(ds['Deaths'])

```
[ ]  from scipy.stats import chisquare
     chisquare(ds['Deaths'])

     Power_divergenceResult(statistic=10572830.630873531, pvalue=0.0)
```

from scipy.stats import chi2_contingency

contingency= pd.crosstab(ds['Deaths'], ds['Confirmed'])

print(contingency)

```
from scipy.stats import chi2_contingency
contingency= pd.crosstab(ds['Deaths'], ds['Confirmed'])
print(contingency)
```

```
Confirmed  10       12       14       17       18       20       23       \
Deaths
0                0        1        1        1        1        1        1
1                1        0        0        0        0        0        0
2                0        0        0        0        0        0        0
3                0        0        0        0        0        0        0
4                0        0        0        0        0        0        0
...            ...      ...      ...      ...      ...      ...      ...
35112            0        0        0        0        0        0        0
44022            0        0        0        0        0        0        0
45844            0        0        0        0        0        0        0
87618            0        0        0        0        0        0        0
148011           0        0        0        0        0        0        0

Confirmed  24       27       48        ...  293606   301708   347923   389717   \
Deaths                                 ...
0                2        1        0 ...        0        0        0        0
1                0        0        0 ...        0        0        0        0
2                0        0        1 ...        0        0        0        0
3                0        0        0 ...        0        0        0        0
4                0        0        0 ...        0        0        0        0
...            ...      ...      ... ...      ...      ...      ...      ...
35112            0        0        0 ...        0        0        0        0
44022            0        0        0 ...        0        0        0        0
45844            0        0        0 ...        0        1        0        0
87618            0        0        0 ...        0        0        0        0
148011           0        0        0 ...        0        0        0        0

Confirmed  395489   452529   816680   1480073  2442375  4290259
Deaths
0                0        0        0        0        0        0
1                0        0        0        0        0        0
2                0        0        0        0        0        0
3                0        0        0        0        0        0
4                0        0        0        0        0        0
...            ...      ...      ...      ...      ...      ...
35112            0        0        0        0        0        0
44022            1        0        0        0        0        0
45844            0        0        0        0        0        0
87618            0        0        0        0        1        0
148011           0        0        0        0        0        1

[150 rows x 184 columns]
```

import seaborn as sns

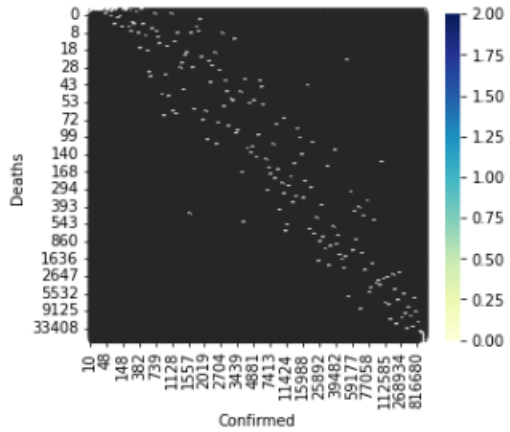
import matplotlib.pyplot as plt

plt.figure(figsize=(5,4))

sns.heatmap(contingency, annot=True, cmap="YlGnBu")

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(5,4))
sns.heatmap(contingency, annot=True, cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f03a28fff90>
```



c, p, dof, expected = chi2_contingency(contingency)

alpha=0.05

print("p value is :"+str(p))

if p <= alpha:

    print('Dependent (reject H0)')

else:

    print('Independent (H0 holds true)')

```
[9] c, p, dof, expected = chi2_contingency(contingency)
    alpha=0.05
    print("p value is :"+str(p))
    if p <= alpha:
        print('Dependent (reject H0)')
    else:
        print('Independent (H0 holds true)')
```

```
p value is :0.07854642350413846
Independent (H0 holds true)
```

d. T-test

from scipy import stats

df1=ds['Deaths']

df2=df1.mean()

t_value,p_value=stats.ttest_1samp(df1,df2)

one_tailed_p_value=float("{:.6f}".format(p_value/2))

print('Test statistic is %f'%float("{:.6f}".format(t_value)))

print('p-value for one tailed test is %f'%one_tailed_p_value)

alpha = 0.05

if one_tailed_p_value<=alpha:

print('We reject the null hypothesis H0.')

print('We do not reject the null hypothesis H0.')

T-Test

```
from scipy import stats
df1=ds['Deaths']
df2=df1.mean()
t_value,p_value=stats.ttest_1samp(df1,df2)
one_tailed_p_value=float("{:.6f}".format(p_value/2))
print('Test statistic is %f'%float("{:.6f}".format(t_value)))
print('p-value for one tailed test is %f'%one_tailed_p_value)
alpha = 0.05
if one_tailed_p_value<=alpha:
    print('We reject the null hypothesis H0.')
    print('We do not reject the null hypothesis H0.')
```

```
Test statistic is 0.000000
p-value for one tailed test is 0.500000
```

e. Analysis of Variance

import scipy.stats as stats

fvalue, pvalue = stats.f_oneway(ds['Active'], ds['Deaths'], ds['Recovered'])

print(fvalue, pvalue)

Analysis of Variance

```
import scipy.stats as stats
fvalue, pvalue = stats.f_oneway(ds['Active'], ds['Deaths'], ds['Recovered'])
print(fvalue, pvalue)
```

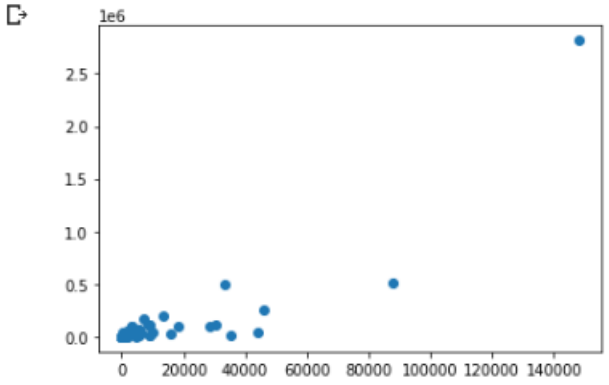
```
3.915327453634628 0.02048404348035053
```

f. Scatterplots

plt.scatter(ds['Deaths'], ds['Active'])

plt.show()

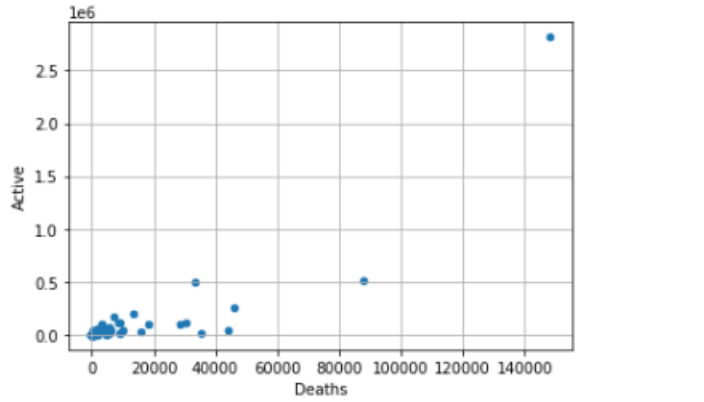
```
plt.scatter(ds['Deaths'], ds['Active'])
plt.show()
```



ds.plot(kind ="scatter", x ='Deaths', y ='Active')

plt.grid()

```
ds.plot(kind ="scatter", x ='Deaths', y ='Active')
plt.grid()
```



RESULT:

Thus, bivariate analysis was performed on the dataset.