MADRAS INSTITUTE OFTECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5612- DATA ANALYTICS AND CLOUD COMPUTING LABORATORY

RECORD

REGISTER NUMBER: 2020506087

NAME : SHRI KAANTH P

SEMESTER : VI

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044

BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by Mr./Ms. SHRI KAANTH P Register Number (2020506087) of **Sixth** Semester, **B.Tech Information Technology** in the **IT5612-Data Analytics and Cloud Computing Laboratory** during the academic period from **January 2023 to June 2023**

Submitted for Practical Examination held on _____

Date:

Course Instructor

Dr. D.Sangeetha

Internal Examiner

External Examiner

TABLE OF CONTENTS

EXP NO :1.a DATE: 25-01-23	STUDY ON NUMPY, SCIPY, STATSMODELS AND PANDAS PACKAGES

Aim:

To study about the Python packages required to work with data analytics.

Theory:

a. Pandas

Pandas is an open-source library that is made mainly for working with relational and labelled data. It provides various data structures and operations for manipulating data.

Functions in Pandas

S. No	Description	Example
1.	read_csv( ) : This helps to read a csv [comma-separated-values] file into a pandas dataframe. Itcan also read files separated by delimiter	df = pd.read_csv (‘data.csv’)
2.	head( ) : The head(n) is used to return the first n rows of a dataset. By default, the function returns 5rows of the dataframe	df.head(10)
3.	describe( ) : It is used to generate descriptive statistics of data in a pandas data frame or services	df.describe( )
4.	memory_usage( ) : It returns a pandas series having the memory usage of each column in a dataframe	df.memory_usage(deep = = true)
5.	loc[ ] : It helps to access a group of rows and columns in a dataset	df.loc[0:4, [‘Name’, ‘Age’]]

b. Numpy

Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transformation and matrices.

Functions in Numpy

S. No	Description	Example
1.	numpy.concatenate( ) : This function is used to join two arrays of same size, either in a row-wise or column-wise way	res = np.concatenate((arr1, arr2), axis =1)
2.	numpy.char.add( ) : It concatenates the data value of two arrays, merge them and represent a new array as a result	res = numpy.char.add([‘python’], [‘java])
3.	numpy.median( ) : It calculates the median of an ordered array	med = np.median(a1)
4.	numpy.average( ) : It returns the average of all data values of the passed array	avg = np.average(a1)
5.	numpy reshape : This function allows us to change the dimension of the array without hampering the data value	res = a1.reshape(3, 2)

c. Scipy

Scipy contains a variety of sub packages which help to solve the most common issue related to scientific computation. It is built on top of the numpy library.

Functions in Scipy

S. No	Description	Example
1.	linalg.solve( ) : This gives the solutions of linear equations inn the matrix form	res = linalg.solve(a,b)
2.	linalg.inv( ) : This returns the inverse of the given matrix.	res = linalg.inv(a)
3.	linalg.det( ) : This returns the determinant of the given matrix.	res = linalg.det(ar)
4.	linalg.eig( ) : This returns the eigen values and eigen vectors of the given matrix.	eval, evec = linalg.eig(ar)
5.	special.logsumexp(x) : This computes the log of sum of exponential input element	np.log(np.sum (np.exp(a)))

d. Stats Model

It provides classes and functions for the estimation of many different statistical models for conducting statistical test and statistical data exploration

Functions in Stats Model:

S. No	Description	Example
1.	get_rdataset : It is used to download any dataset we want	data = sm.datasets. get_rdataset("Tom","Chris").data
2.	add_constant(X) : It is used to add a constant column to input dataset	res = sm.addconstant(res)
3.	OLS(y, x).fit( ) : It is a type of linear square method for estimating unknown parameters in linear regression	res = sm.OLS(y, x).fit( )
4.	linear_rainbow( ) : The null hypothesis is the fit of the model using full sample. It is the same as using a central subset. Rainbow test has power against many different forms of non-linearity	print(sm.stats.linear_rainbow.—doc--)

Result:

Thus, a study on the Python packages used to work with data analysis has been made.

EXP NO :1.b DATE: 01-02-23	READ DATA/DATASET

Aim:

To write python programs to read and display content from text file, csv file and web.

a. Reading from Text File

CODE:

```
file1 = open("file.txt","r")
file1.read()
```

OUTPUT:

```
'Welcome to Data Analytics Laboratory'
```

b. Reading from CSV File

CODE:

```
import csv
with open("sample.csv",'r') as file:
    reader=csv.reader(file)
    for i in reader:
        print(i)
```

OUTPUT:

```
['Number', 'Square']
['1', '1']
['2', '4']
['3', '9']
['4', '16']
['5', '25']
```

c. Reading from Web:

CODE:

```
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)
```

OUTPUT:

```
In 1949,Shri.C.Rajam, gave the newly independent India-Madras Institute of Technology, so that MIT could establish the strong t
echnical base it needed to take its place in the world. It was the rare genius and daring of its founder that made MIT offer co
urses like Aeronautical Engineering, Automobile Engineering, Electronics Engineering and Instrument Technology for the first ti
me in our country. Now it also provides technical education in other engineering fields such as Rubber and Plastic Technology &
Production Technology. It was merged with Anna University in the year 1978. Sixty years hence, while it continues to be a pione
er 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 K
alam, versatile genius like Sujatha and many more. The broad-based education, coupled with practice-oriented training in their
speciality, has enabled the students of MIT to handle with skill and success a wide variety of technical problems. The Madras I
nstitute of Technology has developed into an important centre of engineering education and earned an excellent reputation both
in India and abroad.MIT had received many awards which includes an award for the Best Overall Performance, awarded by Indian So
ciety 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 ex
press and prior written consent of Madras Institute of Technology.No part of this website may be reproduced without Prior Notic
e.
Designed And Maintained by WebTeam MitIndia
```

RESULT:

Thus, the Python programs to read and display content from text file, CSV file, and web have been written and verified.

EXP NO : 2.a **DATE:** 01-02-23	**DESCRIPTIVE ANALYSIS**

Aim:

To explore the various commands for performing descriptive data analysis on the Iris Dataset.

Codes and Output:

i. Import the necessary packages and dataset

```
import pandas as pd;
df=pd.read_csv("archive/city_hour.csv");
df=df[["PM2.5","PM10","NO2","SO2","O3","AQI_Bucket"]]
```

ii. Head

```
df.head()
```

	PM2.5	PM10	NO2	SO2	O3	AQI_Bucket
0	NaN	NaN	40.01	122.07	NaN	NaN
1	NaN	NaN	27.75	85.90	NaN	NaN
2	NaN	NaN	19.32	52.83	NaN	NaN
3	NaN	NaN	16.45	39.53	153.58	NaN
4	NaN	NaN	14.90	32.63	NaN	NaN

iii. Getting Information about the dataset

Shape parameter

```
df.shape
```

```
(707875, 6)
```

Info method

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 707875 entries, 0 to 707874
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   PM2.5       562787 non-null  float64
 1   PM10        411138 non-null  float64
 2   NO2         590753 non-null  float64
 3   SO2         577502 non-null  float64
 4   O3          578667 non-null  float64
 5   AQI_Bucket  578795 non-null  object
dtypes: float64(5), object(1)
memory usage: 32.4+ MB
```

Describe method

```
df.describe()
```

	PM2.5	PM10	NO2	SO2	O3
count	562787.000000	411138.000000	590753.000000	577502.000000	578667.000000
mean	67.622994	119.075804	28.885157	14.038307	34.798979
std	74.730496	104.224752	29.162194	19.305540	29.806379
min	0.010000	0.010000	0.010000	0.010000	0.010000
25%	26.200000	52.380000	10.810000	4.880000	13.420000
50%	46.420000	91.500000	20.320000	8.370000	26.240000
75%	79.490000	147.520000	36.350000	14.780000	47.620000
max	999.990000	1000.000000	499.510000	199.960000	497.620000

iv. Checking Missing Values

isnull( ) method

```
df.isnull().sum()
```

```
PM2.5          145088
PM10           296737
NO2            117122
SO2            130373
O3             129208
AQI_Bucket     129080
dtype: int64
```

v. Checking Duplicates

drop_duplicates( )

```
dup=df.drop_duplicates()
dup
```

	PM2.5	PM10	NO2	SO2	O3	AQI_Bucket
38289	25.91	84.90	58.41	140.10	47.69	Poor
38290	27.27	98.29	71.93	144.20	71.96	Poor
38291	27.27	102.68	66.79	124.37	88.85	Poor
38292	27.30	102.68	60.64	170.01	97.76	Poor
38294	23.84	121.94	70.40	31.13	94.66	Poor
...	...	...	...	...	...	...
707869	8.25	33.25	24.05	1.85	41.38	Satisfactory
707871	17.25	49.25	33.20	2.02	25.58	Satisfactory
707872	36.00	71.00	30.80	1.77	26.15	Good
707873	15.75	63.00	28.90	0.75	15.82	Good
707874	15.00	66.00	26.85	2.10	17.05	Good

347512 rows × 6 columns

Series.value_counts( )

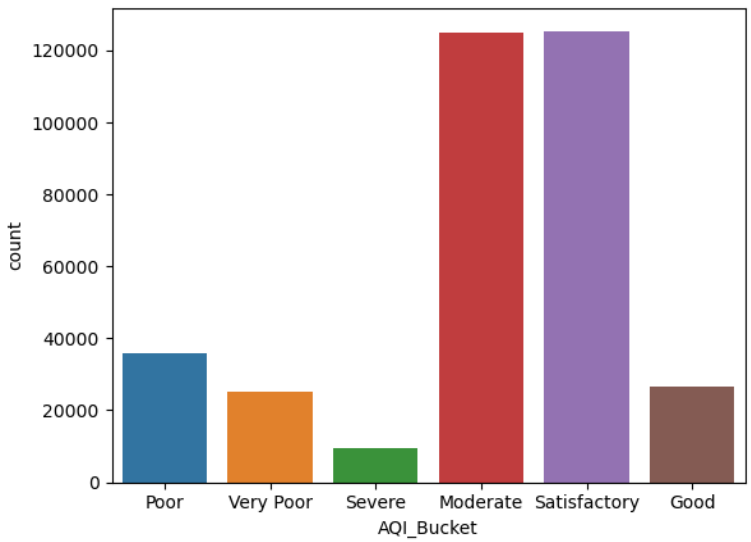
```
df.value_counts("AQI_Bucket")
```

```
AQI_Bucket
Satisfactory    125448
Moderate        124879
Poor             35861
Good             26480
Very Poor        25241
Severe            9648
dtype: int64
```

vi. Data Visualization

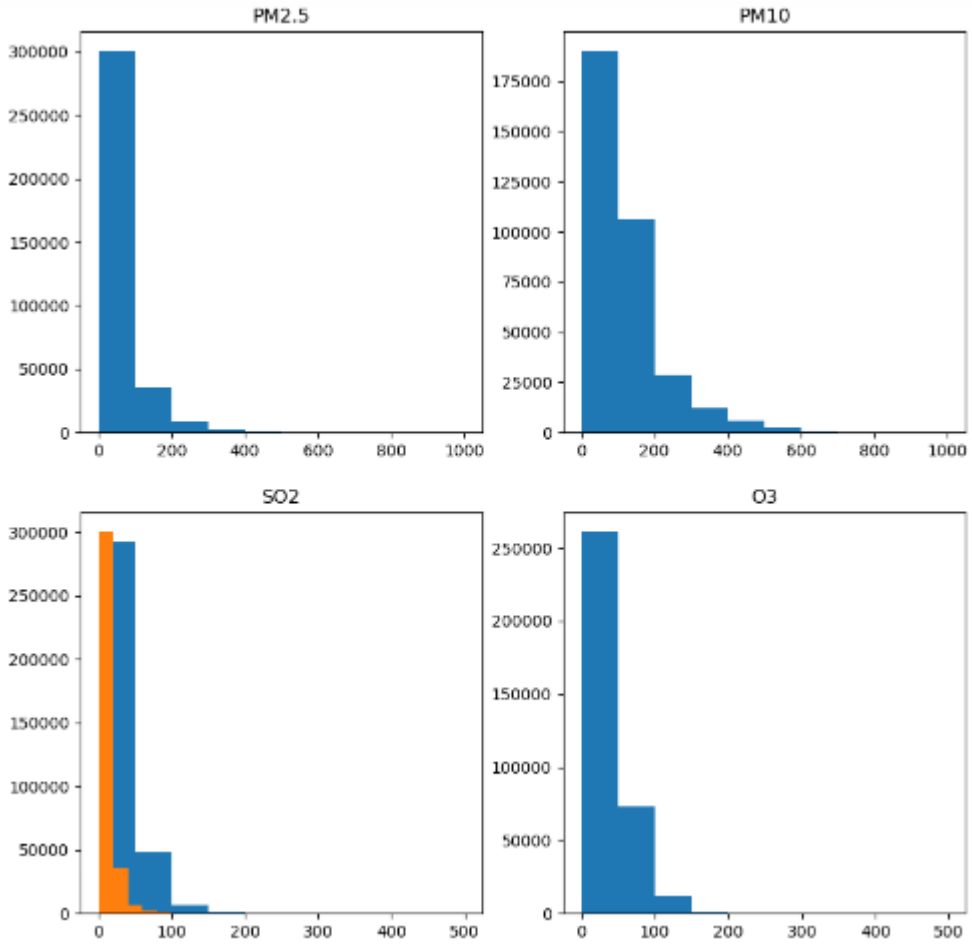
Data Visualization using countplot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='AQI_Bucket', data=df, )
plt.show()
```



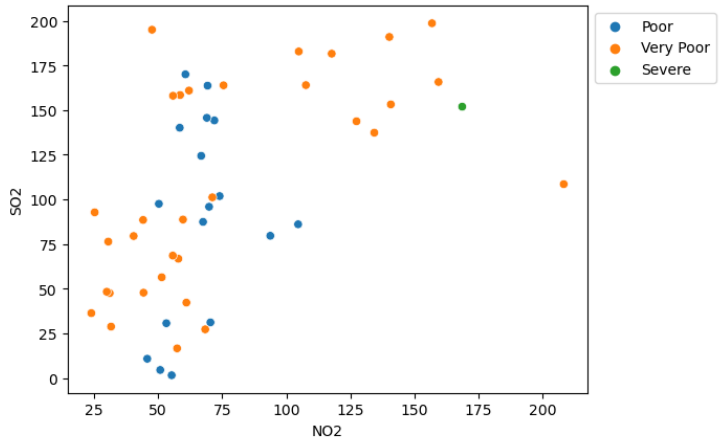
Data Visualization using histogram

```
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("PM2.5")
axes[0,0].hist(df['PM2.5'], bins=10)
axes[0,1].set_title("PM10")
axes[0,1].hist(df['PM10'], bins=10);
axes[1,0].set_title("NO2")
axes[1,0].hist(df['NO2'], bins=10);
axes[1,0].set_title("SO2")
axes[1,0].hist(df['SO2'], bins=10);
axes[1,1].set_title("O3")
axes[1,1].hist(df['O3'], bins=10);
```



Data Visualization using scatterplot

```
sns.scatterplot(x='NO2', y='SO2', hue='AQI_Bucket', data=df.head(50), )

plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```



Result:

Thus, the various commands for performing descriptive data analysis on the Iris Dataset have been executed and verified.

EXP NO : 2.b **DATE:** 08-02-23	**UNIVARIATE ANALYSIS**

Aim:

To perform univariate analysis on the Air Quality dataset.

i. Frequency

```
import pandas as pd
df=pd.read_csv("archive/city_hour.csv")
df['AQI'].value_counts()
```

```
68.0      3389
102.0     3389
64.0      3275
72.0      3183
66.0      3172
          ...
740.0        1
661.0        1
1157.0       1
767.0        1
8.0          1
Name: AQI, Length: 884, dtype: int64
```

ii. Mean

```
df['O3'].mean()
```

```
36.05122673403284
```

iii. Median

```
df['O3'].mean()
```

```
27.96
```

iv. Mode

```
df['O3'].mode()
```

```
0    22.14
Name: O3, dtype: float64
```

v. Variance

```
df['O3'].var()
```

```
848.4269834914085
```

vi. Standard Deviation

```
df['O3'].std()
```

```
29.127769971135937
```

vii. Skewness

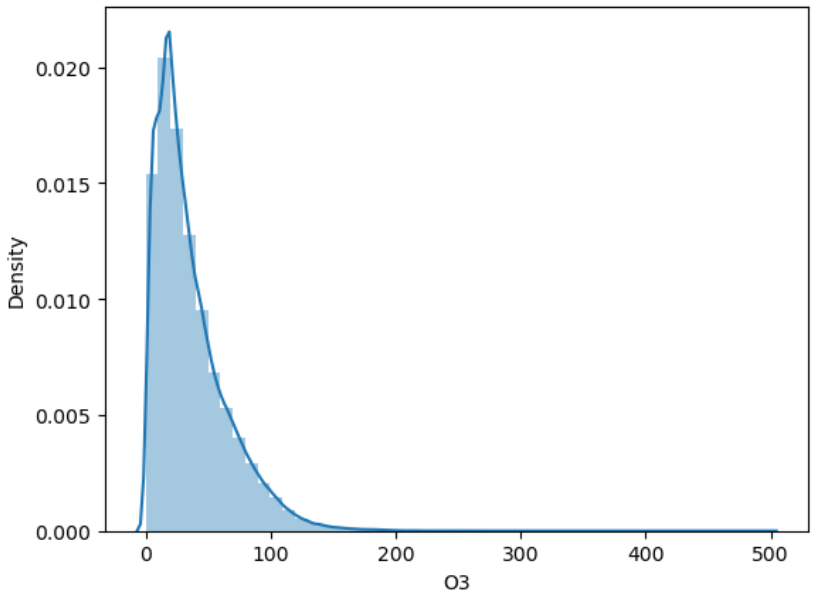
```
from scipy.stats import skew
import numpy as np
import pylab as p
y1 = df['AQI']
print("Skewness = ", skew(y1))
```

```
Skewness =  2.958968947212698
```

viii. Kurtosis

```
import seaborn as sns
sns.distplot(df['O3'], hist=True, kde=True)
df['O3'].kurt()
```

```
4.72922057433085
```



Result:

Thus, univariate analysis on the Air Quality dataset has been performed and verified.

EXP NO : 2.c DATE: 08-02-23	UNIVARIATE TIME SERIES ANALYSIS

Aim:

To perform univariate time series analysis on the Air Quality dataset.

i. Import Packages and Dataset

```
import pandas as pd

import numpy as np

df1=pd.read_csv("archive/time_series_dataset.csv",parse_dates=["Date"],
index_col=["Date"])

df1
```

ii. TIME SERIES FEATURES

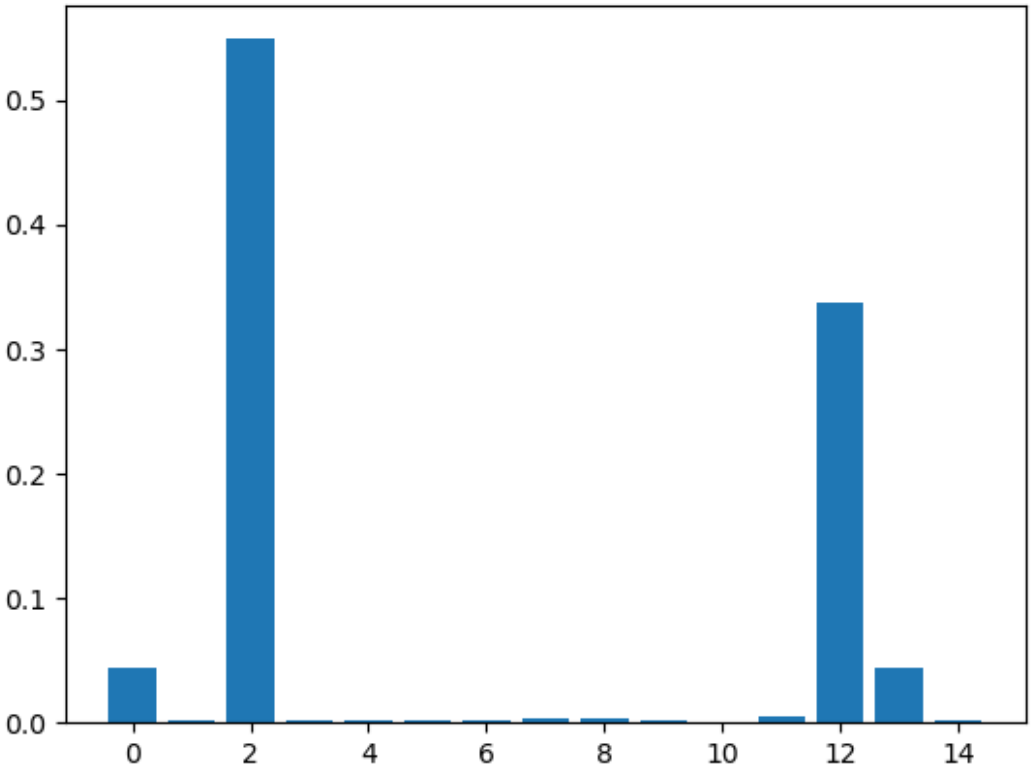
a. Mean in the period of 3 months

```
df1.resample('3M').mean()
```

Date	Temperature
2023-01-31	23.000000
2023-04-30	24.666667
2023-07-31	24.000000
2023-10-31	27.000000

b. Feature Importance

```
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
X, y = make_regression(n_samples=1000, n_features=15,
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()
```



c. Forecast and Evaluate Data

```
pred = model.predict(X)
errors = abs(pred - y)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 0.0 degrees.
```

```
mape = 100 * (errors / y)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 100.0 %.
```

Result:

Thus, univariate time series analysis has been performed on the Air Quality dataset and has been verified.

EXP NO : 2.d **DATE:** 15-02-23	**BIVARIATE ANALYSIS**

Aim:

To perform bivariate analysis on the given dataset.

i. Import necessary packages and dataset

```
import pandas as pd
import numpy as np;
from scipy.stats import pearsonr
df=pd.read_csv("archive/city_hour.csv")
df=df[["AQI_Bucket"]]
encD=pd.get_dummies(df,columns=['AQI_Bucket'])
```

ii. Pearson Correlation Coefficients and Interpretation

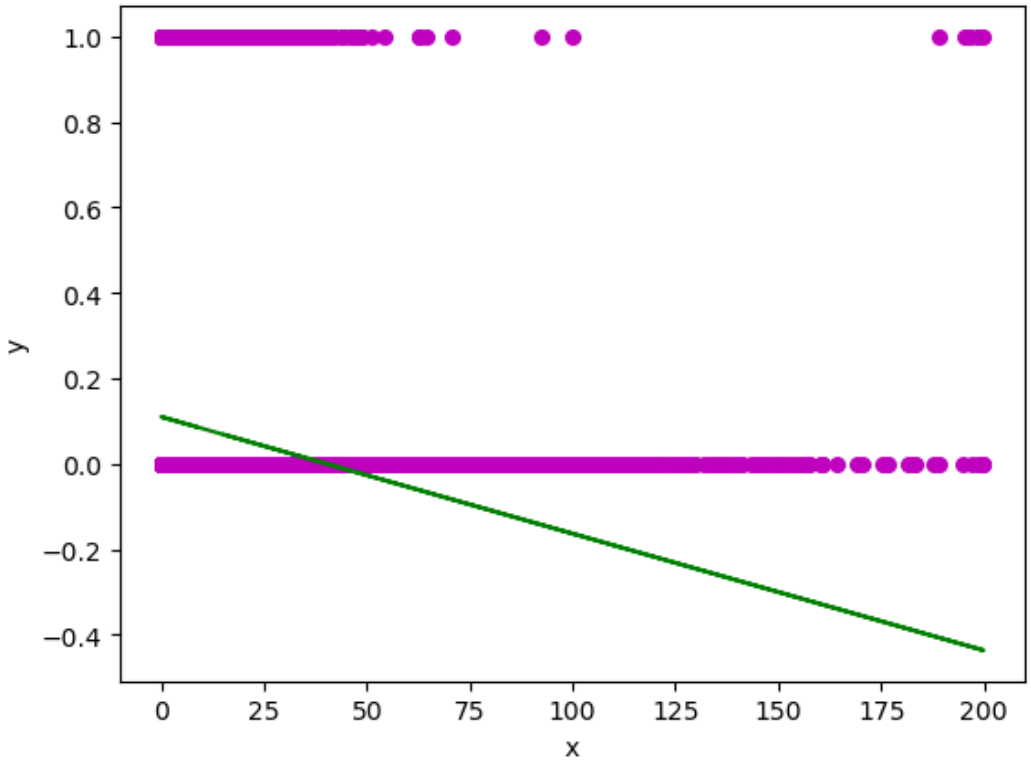
```
encD.corr()
```

	AQI_Bucket_Good	AQI_Bucket_Moderate	AQI_Bucket_Poor	AQI_Bucket_Satisfactory	AQI_Bucket_Severe	AQI_Bucket_Very Poor
AQI_Bucket_Good	1.000000	-0.238048	-0.103960	-0.197548	-0.045576	-0.084458
AQI_Bucket_Moderate	-0.238048	1.000000	-0.287591	-0.546489	-0.126078	-0.233641
AQI_Bucket_Poor	-0.103960	-0.287591	1.000000	-0.238662	-0.055061	-0.102035
AQI_Bucket_Satisfactory	-0.197548	-0.546489	-0.238662	1.000000	-0.104629	-0.193891
AQI_Bucket_Severe	-0.045576	-0.126078	-0.055061	-0.104629	1.000000	-0.044732
AQI_Bucket_Very Poor	-0.084458	-0.233641	-0.102035	-0.193891	-0.044732	1.000000

iii. Simple Linear Regression

```
import matplotlib.pyplot as plt
def estimate_coef(x, y):
 n = np.size(x)
 m_x = np.mean(x)
 m_y = np.mean(y)
 SS_xy = np.sum(y*x) - n*m_y*m_x
 SS_xx = np.sum(x*x) - n*m_x*m_x
 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):
 plt.scatter(x, y, color = "m",
 marker = "o", s = 30)
 # predicted response vector
 y_pred = b[0] + b[1]*x
 plt.plot(x, y_pred, color = "g")
 plt.xlabel('x')
 plt.ylabel('y')
 plt.show( )
def main():
 x = encD["SO2"]
 y = encD["AQI_Bucket_Good"]
 b = estimate_coef(x, y)
 print("Estimated coefficients:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))
 plot_regression_line(x, y, b)
if __name__ == "__main__":
 main( )
```

```
Estimated coefficients:
b_0 = 0.1100817841186347
b_1 = -0.002734473404634446
```



iv. Chi-Squared Test

```
from scipy.stats import chi2_contingency
stat, p, dof, expected = chi2_contingency(encD["AQI"])
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 1.0
Independent (H0 holds true)
```

v. T-Test

```
from scipy.stats import ttest_ind
param1 = encD["O3"]
param2 = encD["AQI_Bucket_Good"]
stat, p = ttest_ind(param1, param2)
print("Statistics: %.3f, p = %.3f" % (stat,p))
```

```
Statistics: 471.017, p = 0.000
```

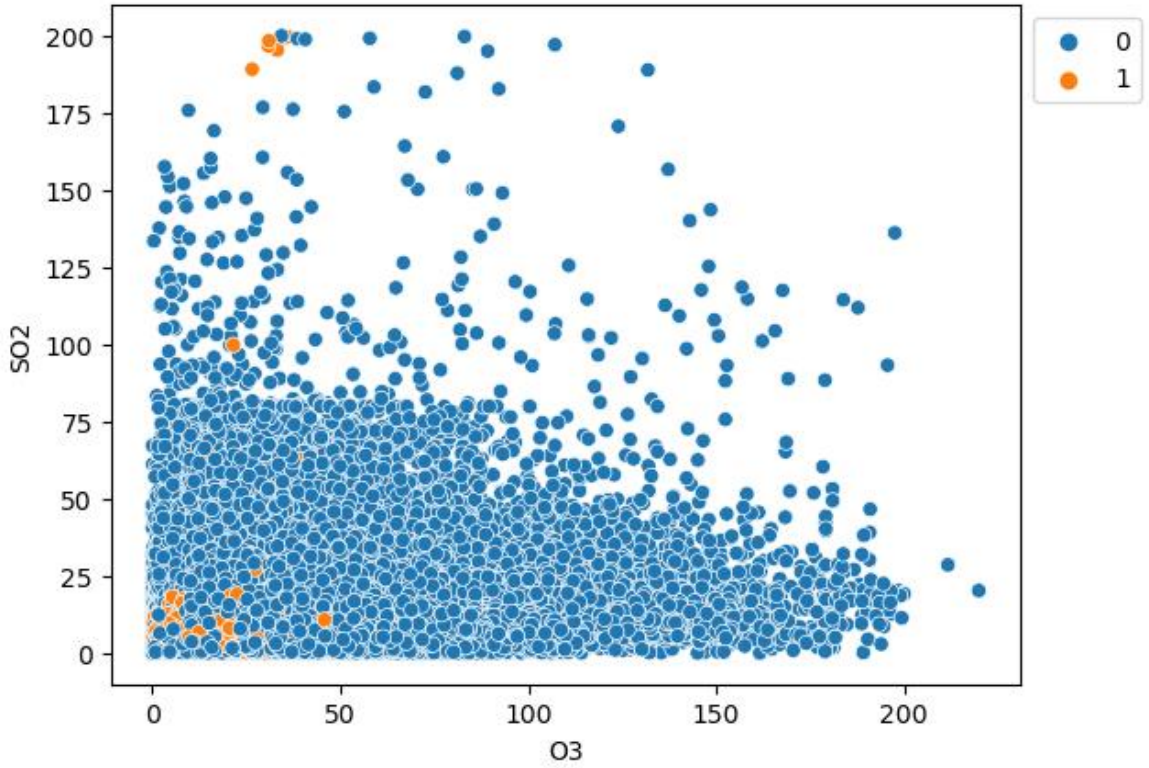
vi. Analysis of Variance

```
import scipy.stats as stats
fvalue, pvalue = stats.f_oneway(encD["O3"],
encD["AQI_Bucket_Good"],encD["AQI_Bucket_Moderate"],encD["AQI_Bucket
_Poor"],encD["AQI_Bucket_Satisfactory"],encD["AQI_Bucket_Severe"],encD["A
QI_Bucket_Very Poor"] )
print("fvalue: ", fvalue, ", pvalue: ", pvalue)
```

```
fvalue:  220625.20408816423 , pvalue:  0.0
```

vii. Scatterplots

```
import seaborn as sns
sns.scatterplot(x = "O3", y = "SO2", hue= "AQI_Bucket_Good", data = encD)
plt.legend(bbox_to_anchor = (1,1), loc = 2)
plt.show( )
```



Result:

Thus, bivariate analysis has been successfully carried out on the Air Quality dataset.

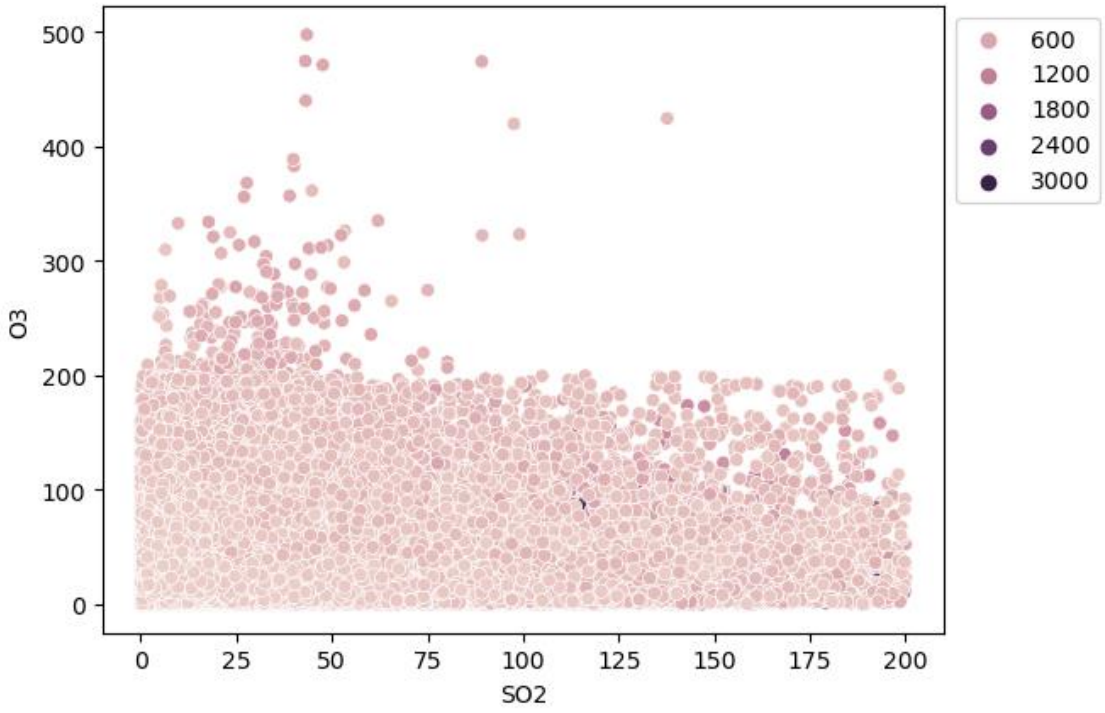
EXP NO : 2.e **DATE:** 15-02-23	**MULTIVARIATE ANALYSIS**

Aim:

To perform bivariate analysis on the given dataset.

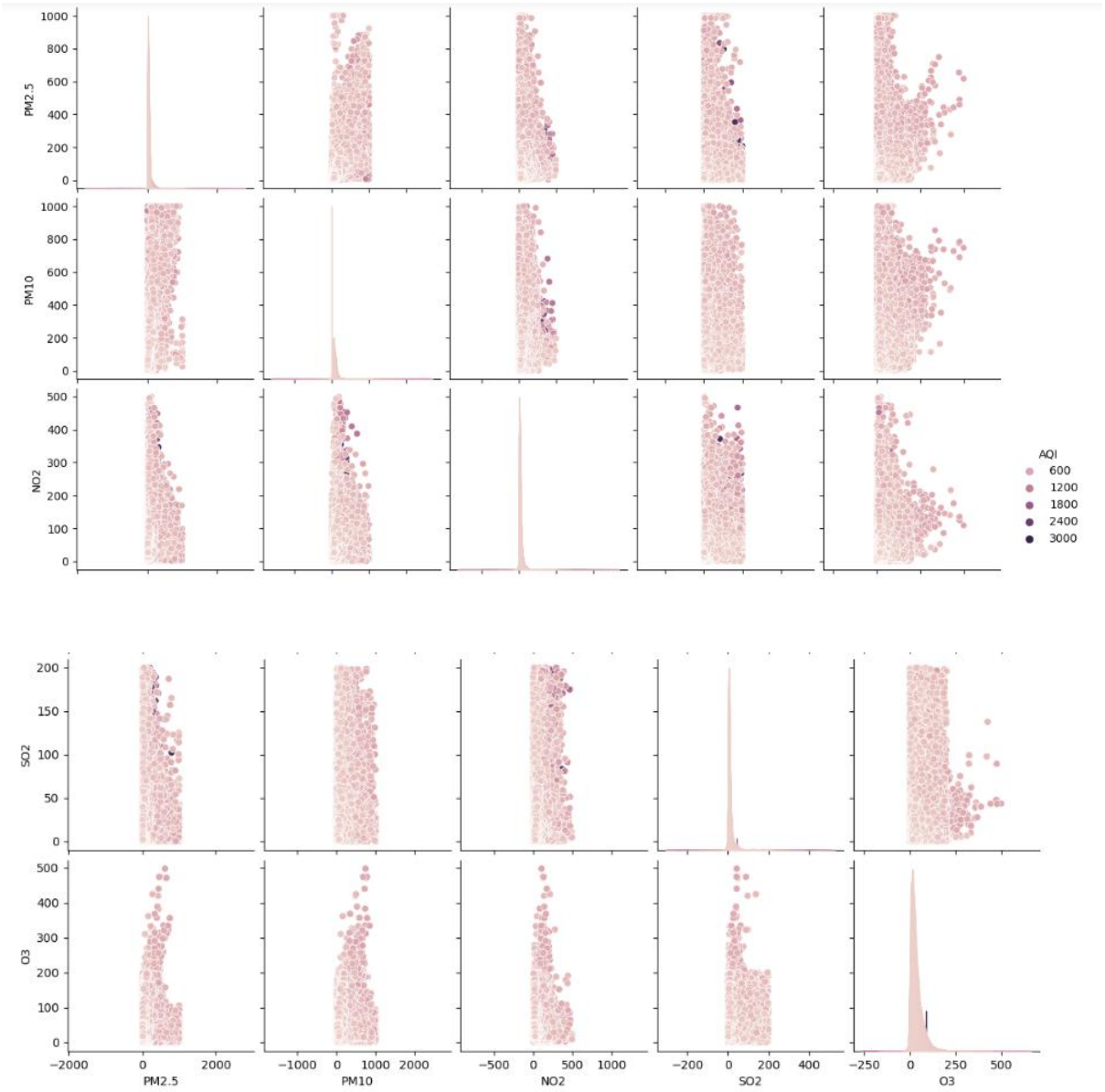
i. Scatterplot

```
sns.scatterplot(x='SO2',y='O3',hue='AQI',data=df,)
plt.legend(bbox_to_anchor=(1,1),loc=2)
plt.show()
```



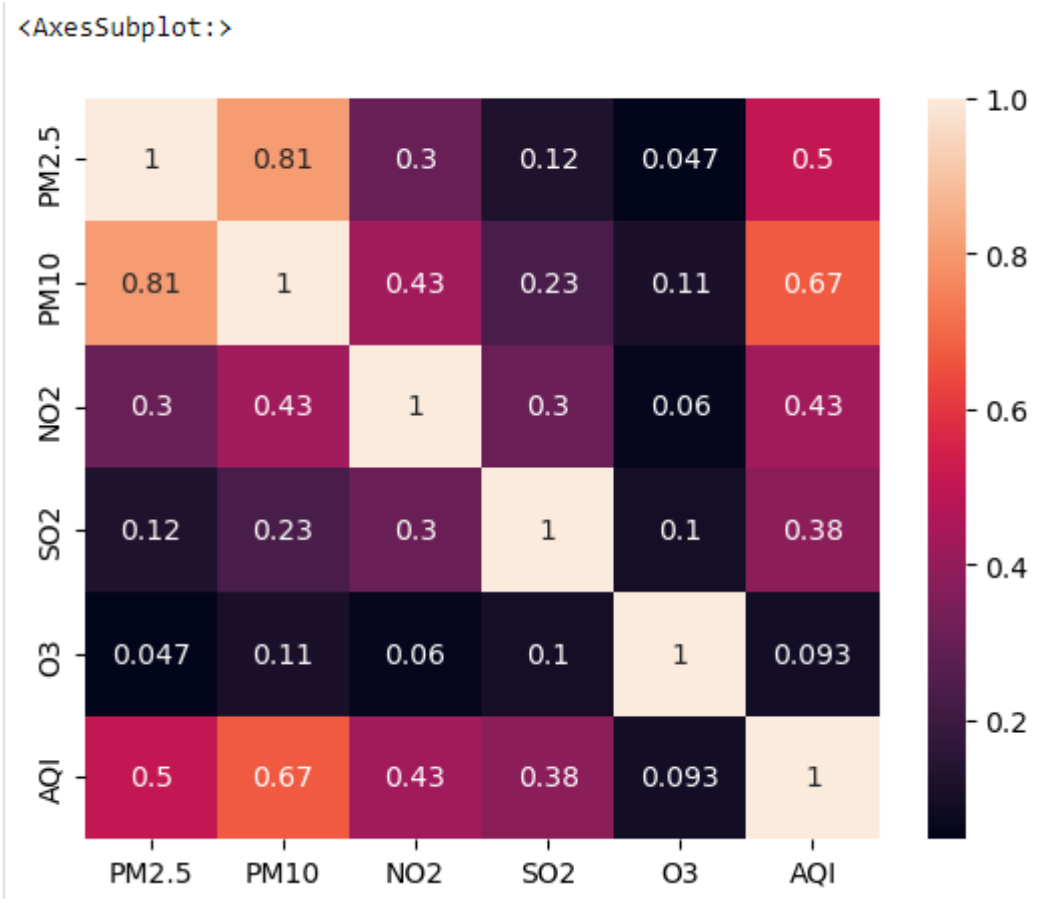
ii. Pair Plots

```
sns.pairplot(df,hue='AQI')
```



iii. Heat Map

```
sns.heatmap(df.corr(method="pearson"),annot=True)
```



Result:

Thus, multivariate analysis has been successfully carried out on the Air Quality dataset.

EXP NO : 3 DATE: 22-02-23	CLASSIFICATION OF NAÏVE BAYES

Aim:

To build models to perform Gaussian and Multinomial Naïve Bayes classification for the given dataset.

GAUSSIAN NAIVE BAYES

i. Import necessary packages and dataset

```
import pandas as pd
import numpy as np
df=pd.read_csv("dataset.csv")
df.head()
```

	PM2.5	PM10	NO2	SO2	O3	AQI
0	104.00	148.50	23.00	15.30	117.62	3.0
1	94.50	142.00	16.25	17.00	136.23	3.0
2	82.75	126.50	14.83	15.40	149.92	3.0
3	68.50	117.00	13.60	21.80	161.70	3.0
4	69.25	112.25	11.80	21.38	161.68	3.0

ii. Exploratory Data Analysis

a. Shape

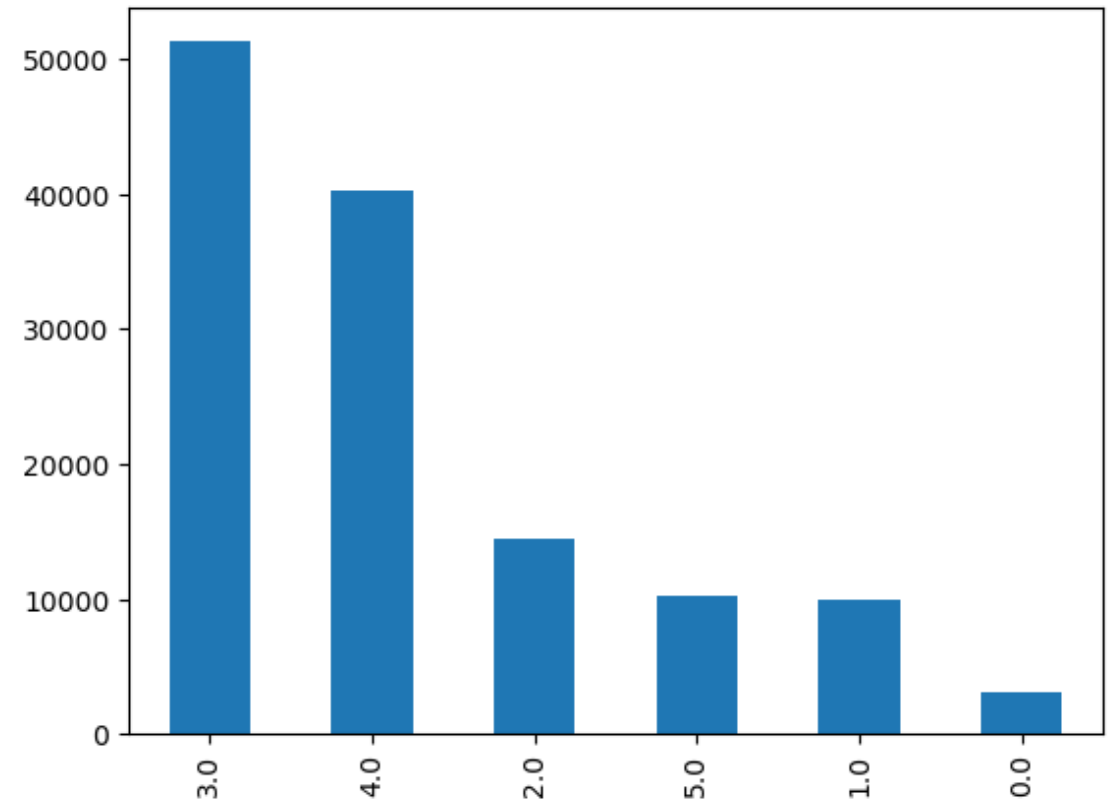
```
df.shape
```

```
(129277, 6)
```

b. Bar plot

```
import seaborn as sns
import matplotlib.pyplot as plt
df['AQI'].value_counts().plot.bar()
```

```
<AxesSubplot:>
```



iii. Identify Numerical and Categorical Data

```
import numpy as np
numeric_data = df.select_dtypes(include=[np.number])
categorical_data = df.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Numerical attributes: ", numeric_data.columns)

print("Number of categorical variables: ", categorical_data.shape[1])
print("Categorical attributes: ",categorical_data.columns)
```

```
Number of numerical variables:  6
Numerical attributes:  Index(['PM2.5', 'PM10', 'NO2', 'SO2', 'O3', 'AQI'], dtype='object')
Number of categorical variables:  0
Categorical attributes:  Index([], dtype='object')
```

iv. Identify the Missing Values

```
df.isnull().sum()
```

```
PM2.5    0
PM10     0
NO2      0
SO2      0
O3       0
AQI      0
dtype: int64
```

v. Data preprocessing

```
from sklearn import preprocessing
#Feature scaling - min max scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df)
encD=df
encD.head()
```

	PM2.5	PM10	NO2	SO2	O3	AQI
0	104.00	148.50	23.00	15.30	117.62	3.0
1	94.50	142.00	16.25	17.00	136.23	3.0
2	82.75	126.50	14.83	15.40	149.92	3.0
3	68.50	117.00	13.60	21.80	161.70	3.0
4	69.25	112.25	11.80	21.38	161.68	3.0

vi. Split dataset into train and test set

```
cols=['PM2.5',"PM10","NO2","SO2","O3"]
X=encD[cols]
y=encD['AQI']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

vii. Train the model

```
from sklearn.naive_bayes import GaussianNB
gNBmodel = GaussianNB()
gNBmodel.fit(X_train,y_train)
```

```
GaussianNB()
```

viii. Test model and display result

```
y_pred = gNBmodel.predict(X_test)
y_pred
```

```
array([3., 3., 4., ..., 5., 3., 5.])
```

ix. Accuracy Score

```
print('Accuracy of Gaussian Naive Bayes on test set: {:.2f}'.format
(gNBmodel.score(X_test, y_test)))
```

```
Accuracy of Gaussian Naive Bayes on test set: 0.61
```

x. Check for overfitting and underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = gNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

```
0.4338498826805559
0.42874823538512113

Underfitting is present
```

xi. Confusion matrix

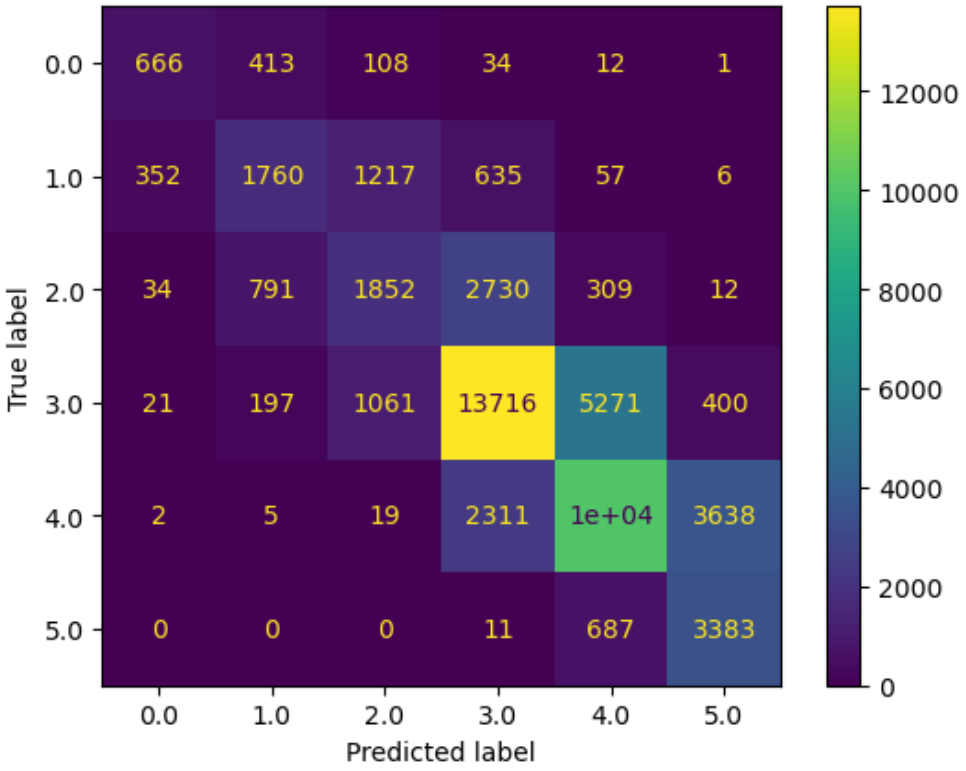
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[  666,   413,   108,    34,    12,     1],
       [  352,  1760,  1217,   635,    57,     6],
       [   34,   791,  1852,  2730,   309,    12],
       [   21,   197,  1061, 13716,  5271,   400],
       [    2,     5,    19,  2311, 10000,  3638],
       [    0,     0,     0,    11,   687,  3383]], dtype=int64)
```

xii. Analyse model performance visually

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=gNBmodel.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=gNBmodel.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2124a9cbeb0>
```



xiii. Classification report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.62      0.54      0.58      1234
         1.0       0.56      0.44      0.49      4027
         2.0       0.44      0.32      0.37      5728
         3.0       0.71      0.66      0.68     20666
         4.0       0.61      0.63      0.62     15975
         5.0       0.45      0.83      0.59      4081

    accuracy                           0.61     51711
   macro avg       0.56      0.57      0.55     51711
weighted avg       0.61      0.61      0.60     51711
```

MULTINOMIAL NAÏVE BAYES

i. Train the model

```
from sklearn.naive_bayes import MultinomialNB
mNBmodel = MultinomialNB()
mNBmodel.fit(X_train,y_train)
```

```
MultinomialNB()
```

ii. Test model and display result

```
y_pred = mNBmodel.predict(X_test)
y_pred
```

```
array([3., 4., 4., ..., 2., 2., 3.])
```

iii. Accuracy Score

```
print('Accuracy of Multinomial Naive Bayes on test
set:{:.2f}'.format(mNBmodel.score(X_test, y_test)))
```

```
Accuracy of Multinomial Naive Bayes on test set:0.31
```

iv. Checking for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = mNBmodel.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
if mae_train < mae_test:
 print("Overfitting is present")
else:
 print("Underfitting is present")
```

```
0.9832787561560478
0.9877008760225097

Overfitting is present
```

v. Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[ 687,  246,  210,   67,    8,    5],
       [1438, 1124,  695,  561,  128,   48],
       [ 708, 1696, 1084, 1189,  615,  428],
       [ 628, 2617, 3531, 5921, 4117, 3616],
       [ 135,  951, 1275, 4402, 4877, 4590],
       [   1,   34,   52,  503, 1393, 2131]], dtype=int64)
```

vi. Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.19      0.56      0.29      1234
         1.0       0.17      0.28      0.21      4027
         2.0       0.16      0.19      0.17      5728
         3.0       0.47      0.28      0.35     20666
         4.0       0.43      0.30      0.36     15975
         5.0       0.20      0.52      0.29      4081

    accuracy                           0.31     51711
   macro avg       0.27      0.36      0.28     51711
weighted avg       0.37      0.31      0.32     51711
```

Result:

Thus, the models to perform Gaussian and Multinomial Naïve Bayes classification for the given dataset have been built successfully.

EXP NO : 4 DATE: 22-02-23	SVM CLASSIFICATION

Aim:

To build models to perform classification using SVM classifier for the given dataset.

Codes and Output:

```
import pandas as pd
import numpy as np
df=pd.read_csv("dataset.csv")
```

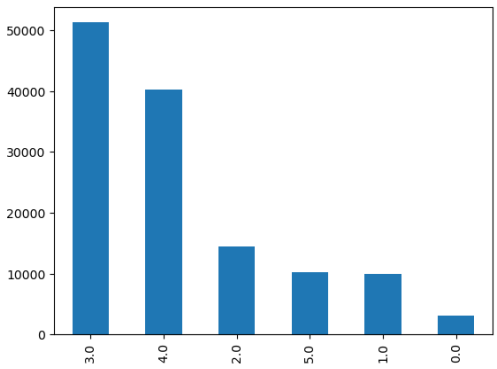
i. Exploratory Data Analysis

```
df.shape
```

```
(707875, 6)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
df['AQI'].value_counts().plot.bar()
```

```
<AxesSubplot:>
```



ii. Explore Missing Values in Variables

```
df.isnull().sum()
```

```
PM2.5    0
PM10     0
NO2      0
SO2      0
O3       0
AQI      0
dtype: int64
```

iii. Split data into separate training and test set

```
from sklearn.model_selection import train_test_split

cols=['PM2.5',"PM10","NO2","SO2","O3"]

X=df[cols]

y=df['AQI']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=40)
```

iv. Run SVM with default Hyperparameter

```
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

svc=SVC()

svc.fit(X_train,y_train)

y_pred=svc.predict(X_test)

print('Model accuracy score with default hyperparameters: {0:0.4f}'.

format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.6655
```

v. Compare the Train-Set and Test-Set Accuracy

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.6655
```

vi. Check for Overfitting and Underfitting

```
from sklearn.metrics import mean_absolute_error
y_train_pred = svc.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_pred)
print(mae_train)
print(mae_test)
if mae_train < mae_test:
    print("Overfitting is present")
else:
    print("Underfitting is present")
```

vii. Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

viii. Classification report

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0.0	0.81	0.49	0.61	615
1.0	0.65	0.56	0.60	1988
2.0	0.54	0.35	0.42	2942
3.0	0.69	0.78	0.73	10093
4.0	0.67	0.73	0.70	8158
5.0	0.66	0.46	0.54	2060
accuracy			0.67	25856
macro avg	0.67	0.56	0.60	25856
weighted avg	0.66	0.67	0.66	25856

Result:

Thus, the models to perform classification using SVM classifier for the Air Quality dataset have been built successfully.

EXP NO : 5 DATE: 01-03-23	LOGISTIC REGRESSION

Aim:

To perform logistic regression on the Air Quality dataset.

Codes and Output:

i. Import necessary packages and dataset

```
import pandas as pd
import numpy as np
df=pd.read_csv("dataset.csv")
```

ii. Split data into separate training and test set

```
from sklearn.model_selection import train_test_split
cols=['PM2.5',"PM10","NO2","SO2","O3"]
X=df[cols]
y=df['AQI']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
```

iii. Train the model

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
y_pred = lr_model.predict(X_test)
```

iv. Accuracy Score

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.4583462252475248
```

v. Confusion Matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[  51,  374,    8,  187,    1,    0],
       [ 122,  837,   94,  981,    1,    0],
       [  31,  570,   89, 2181,   42,    0],
       [  32,  528,  177, 8694,  832,    0],
       [   5,  113,   73, 5614, 2178,    5],
       [   0,    2,    4,  657, 1371,    2]], dtype=int64)
```

vi. Classification report

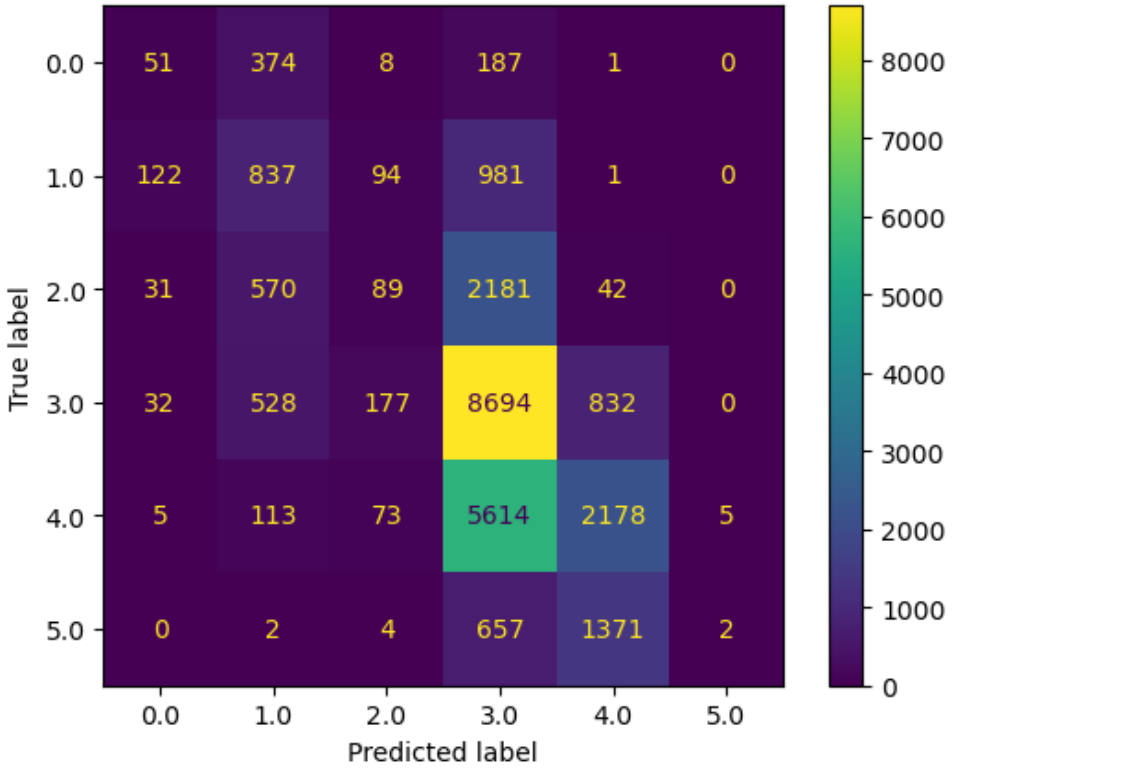
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.21      0.08      0.12       621
         1.0       0.35      0.41      0.38      2035
         2.0       0.20      0.03      0.05      2913
         3.0       0.47      0.85      0.61     10263
         4.0       0.49      0.27      0.35      7988
         5.0       0.29      0.00      0.00      2036

    accuracy                           0.46     25856
   macro avg       0.33      0.27      0.25     25856
weighted avg       0.42      0.46      0.39     25856
```

vii. Analyse model performance visually

```
from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred,
labels=lr_model.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,

display_labels=lr_model.classes_)

disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x14519a28940>
```



Result:

Thus, logical regression has been carried out on the Air Quality dataset successfully.

EXP NO : 6 DATE: 01-03-23	MULTIPLE REGRESSION

Aim:

To build a multiple regression model on the Air Quality dataset.

i. **Import necessary packages and dataset**

```
import pandas as pd
import numpy as np
df=pd.read_csv("dataset.csv")
```

ii. **Data preprocessing**

```
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df)
encD=df
encD.head()
```

	PM2.5	PM10	NO2	SO2	O3	AQI
0	104.00	148.50	23.00	15.30	117.62	3.0
1	94.50	142.00	16.25	17.00	136.23	3.0
2	82.75	126.50	14.83	15.40	149.92	3.0
3	68.50	117.00	13.60	21.80	161.70	3.0
4	69.25	112.25	11.80	21.38	161.68	3.0

iii. **Split dataset into train and test set**

```
cols=['PM2.5',"PM10","NO2","SO2","O3"]
X=encD[cols]
y=encD['AQI']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

iv. **Train the model**

```
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression()
linear_regression.fit(X_train, y_train)
```

```
LinearRegression()
```

v. **Test model and display result**

```
y_pred=linear_regression.predict(X_test)
y_pred
```

```
array([[3.81702004],
       [2.80993375],
       [2.87657045],
       ...,
       [3.17608338],
       [2.21710344],
       [2.44041965]])
```

Result:

Thus, multiple regression has been carried out on the Air Quality dataset successfully.

EXP NO : 7.a **DATE:** 08-03-23	**OPENSTACK**

Aim:

To understand OpenStack deployment, its implementation and its applications.

Theory:

OpenStack is a cloud operating system that controls large pools of compute, storage and networking resources throughout a data-centre, all managed and provisioned through APIs with common authentication mechanisms.

A dashboard is also available, giving administrators control while empowering their users to provide resources through a web interface. It began in 2010 as a joint project of Ra despace hosting and NASA. It was managed by the OpenStack Foundation, a non-profit entity.

Working of OpenStack

It is essentially a series of commands known as scripts. These scripts are bundled into packages called projects that relay tasks that create cloud environments. To create these environments, OpenStack relies on

- Virtualization that creates a layer of virtual resources abstracted from hardware
- A base as that carries out commands given by OpenStack scripts

OpenStack uses virtualized resources to build clouds. It doesn't execute commands, rather delays them to base OS.

Components of OpenStack

OpenStack's architecture is made up of numerous open source projects. These are used to set up OpenStack's undercloud and overcloud. Overcloud is used by cloud users and undercloud is used by system admins. Underclouds contain the core components system admins need to set up and manage end user's environments called overclouds.

There are 6 stable, core services that handle computing, networking, storage, identity and images. These make up the infrastructure of OpenStack that allows the rest of the projects to handle dashboarding, orchestration, etc. The 6 components are:

Nova: It is a full management tool that helps compute resource-handling, scheduling, creation, deletion, etc.

Neutron: It connects the networks across other services

Swift: It is a highly fault-tolerant object storage service that stores and retrieves unstructured data objects

Cinder: Provides persistent block storage

Keystone: Authenticates and authorises all services

Glance: Stores and retrieves VM disc images from various locations

Deployment of OpenStack

It is mostly deployed as infrastructure-as-a-Service (IaaS) in both public and private clouds where virtual servers and other resources are made available to users. The software platform consists of interrelated components control diverse, multivendor hardware. Lifecycle management tools and packaging are used to help instances maintain the lifecycle of deployments. Frameworks for the above are Tripleo, OpenStack-helm, kolla-ansible, kayole, etc.

Challenges in Implementation

- Installation challenges
- Documentation
- Upgrading OpenStack
- Long Term Support

Applications of OpenStack

- Private clouds and Public clouds
- Network function virtualization
- Containers

Result:

Thus, OpenStack deployment, its implementation and its applications have been studied.

EXP NO : 7.b DATE: 08-03-23	OPENSTACK INSTALLATION

Aim:

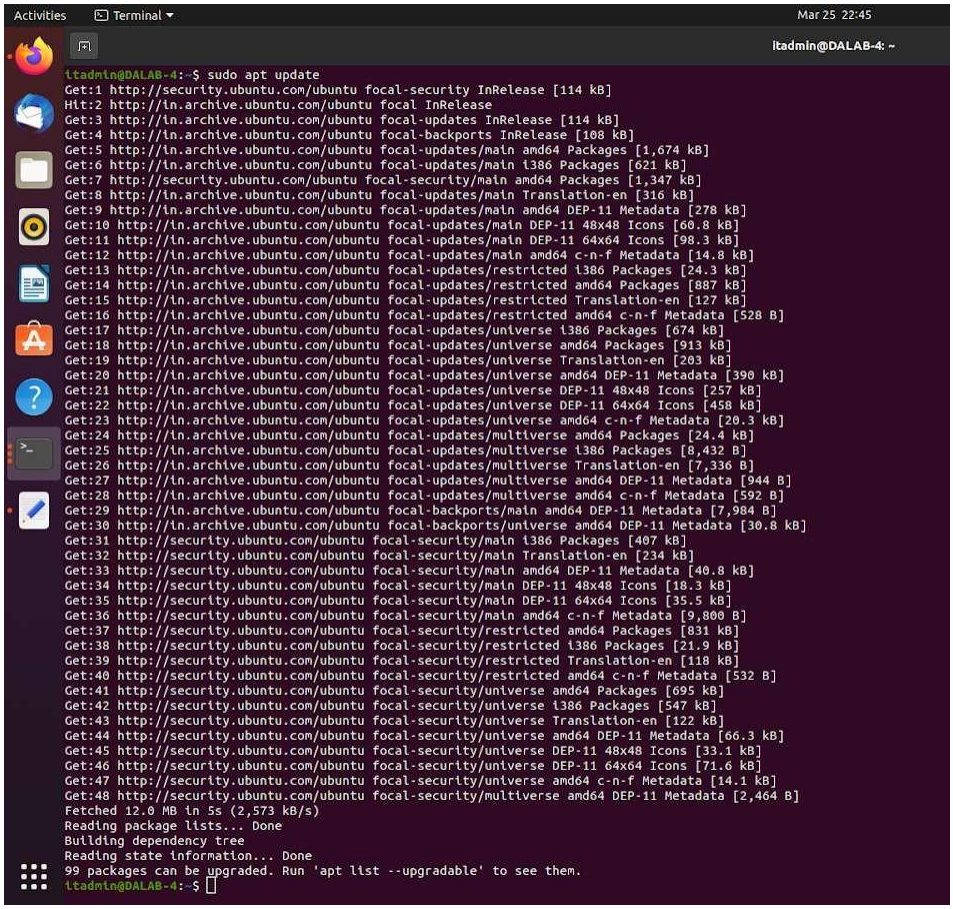
To install OpenStack and implement Infrastructure-as-a-service using it.

Installation of OpenStack in Ubuntu

1. Login to sudo ("superuser do") user
2. Open Terminal
3. In terminal, type:

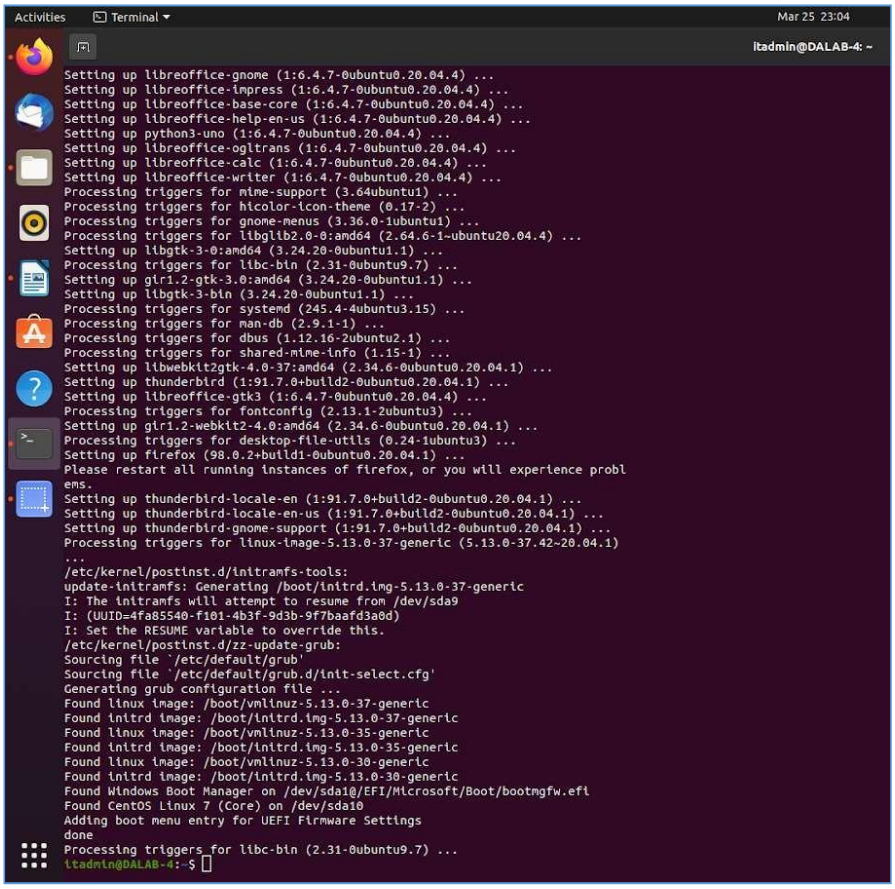
```
sudo apt update
```

This downloads package information from all configured sources.



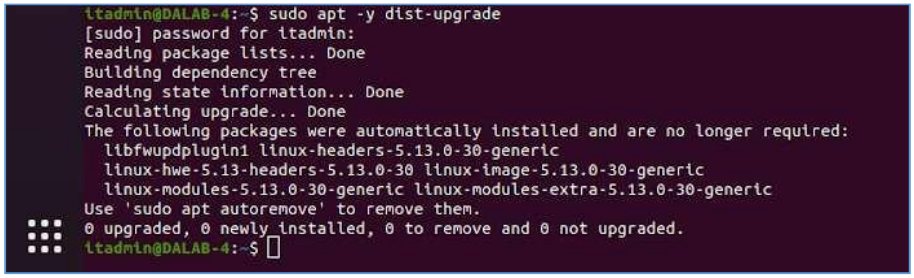
4. Upgrade everything in the system, all the packages, and the kernel to the latest versions assupported by the repositories:

```
sudo apt –y upgrade
```



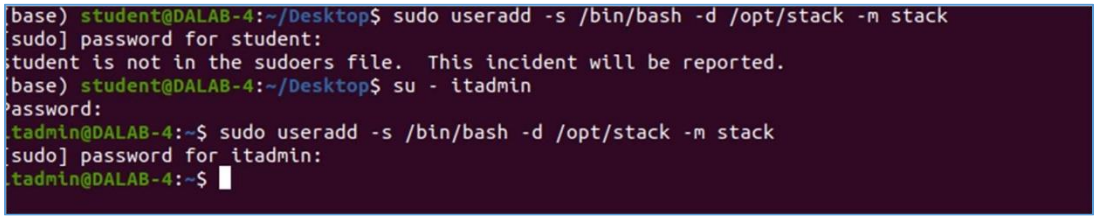
5. Upgrade existing packages, installs new dependencies that are not in the system, and deletesthose that are not needed:

```
sudo apt –y dist-upgrade
```



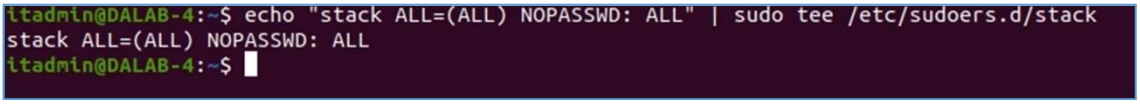
6. Add user called stack to run DevStack. It should be run as non-root user with sudo enabled:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
```



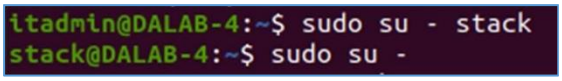
7. User should have sudo privileges to make changes to the system:

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



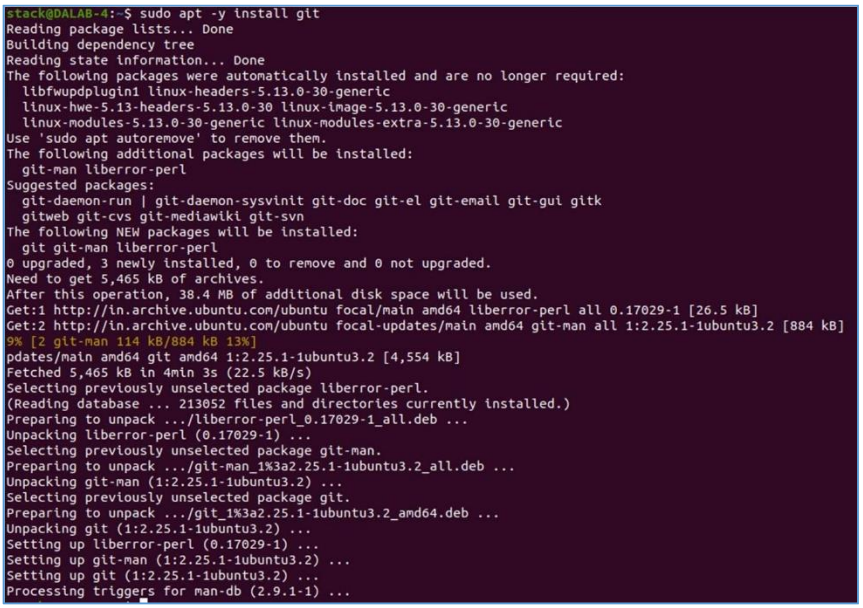
8. Log in to the stack once user is created:

```
sudo su - stack
```



9. Install git:

```
sudo apt -y install git
```



10. Download devstack from its repository into system:

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

```
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. Download devstack setup configurations files for it. Need to navigate devstack folder by running:

```
cd devstack
nano local.conf
vi local.conf
```

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

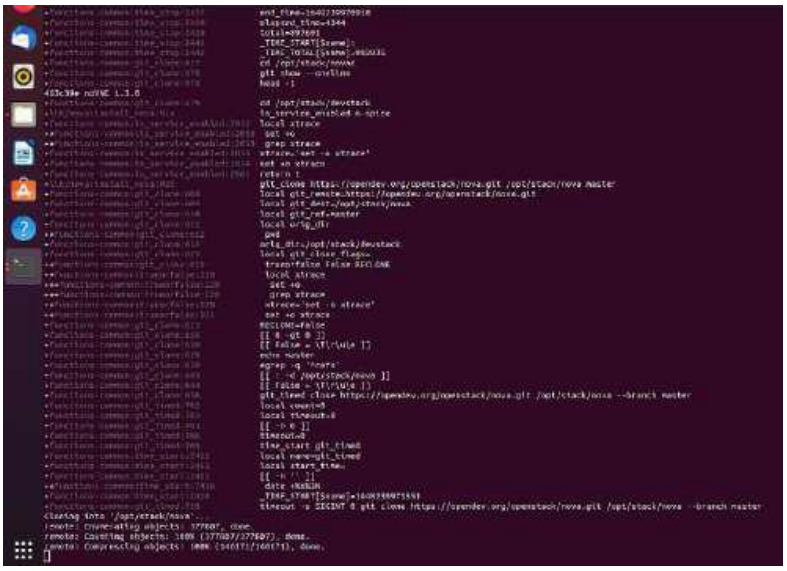
12. Add following inside local.conf:

```
[[local|localrc]]
# Password for KeyStone, Database, RabbitMQ and
Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

After setting above, in local.conf, press Esc key and type :wq to write/save and quit

13.Run following to setup Openstack on system:

```
./stack.sh
```

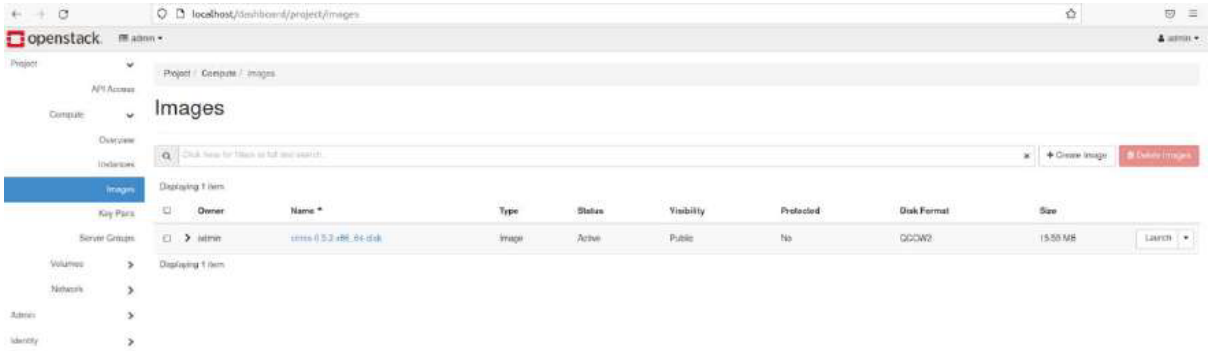


After installation, terminal:

```
=========================
DevStack Component Timing
 (times are in seconds)
=========================
wait_for_service      18
pip_install          280
apt-get              942
run_process           68
dbsync                66
git_timed           1034
apt-get-update         1
test_with_retry       51
async_wait          1564
osc                  283
-------------------------
Unaccounted time     816
=========================
Total runtime       5123

=================
 Async summary
=================
 Time spent in the background minus waits: 2015 sec
 Elapsed time: 5123 sec
 Time if we did everything serially: 7138 sec
 Speedup:  1.39332


This is your host IP address: 192.168.112.197
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.112.197/dashboard
Keystone is serving at http://192.168.112.197/identity/
The default users are: admin and demo
The password: StrongAdminSecret

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: zed
Change: 0ed70e3f7687ffa62a8a4a38cdad14abdc8c7fa7 Merge "Update DEVSTACK_SERIES to zed" 2022-03-28 13:02:03 +0000
OS Version: Ubuntu 20.04 focal

2022-03-29 15:26:41.917 | stack.sh completed in 5123 seconds.
```

14. Browse URL on browser:
    http://localhost/dashboard

15. Enter credentials. Log in as admin using username ‘admin’ and password as given in local.conf file:



Result:

Thus, OpenStack has been successfully installed.

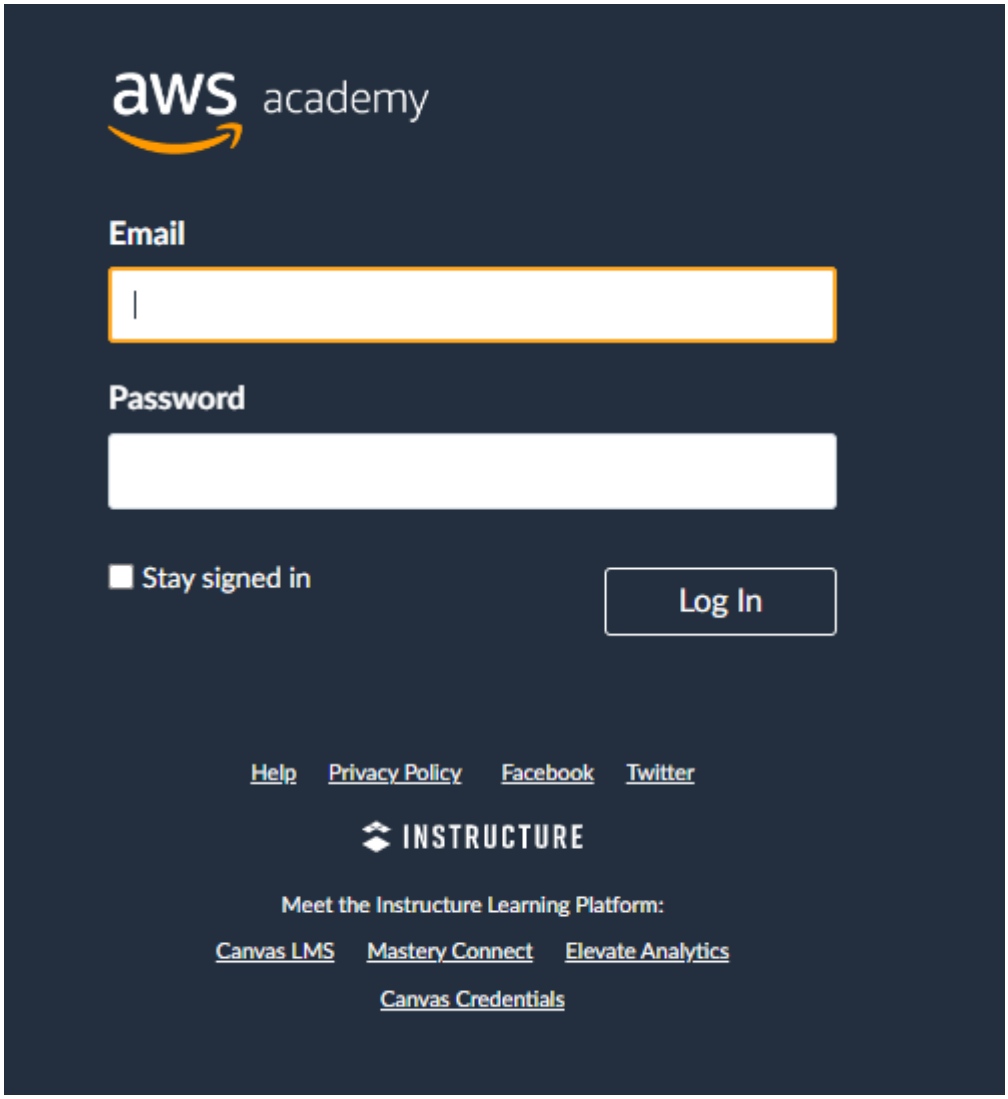
EXP NO : 8 DATE: 15-03-23	CREATION OF VM INSTANCES IN AWS

Aim:

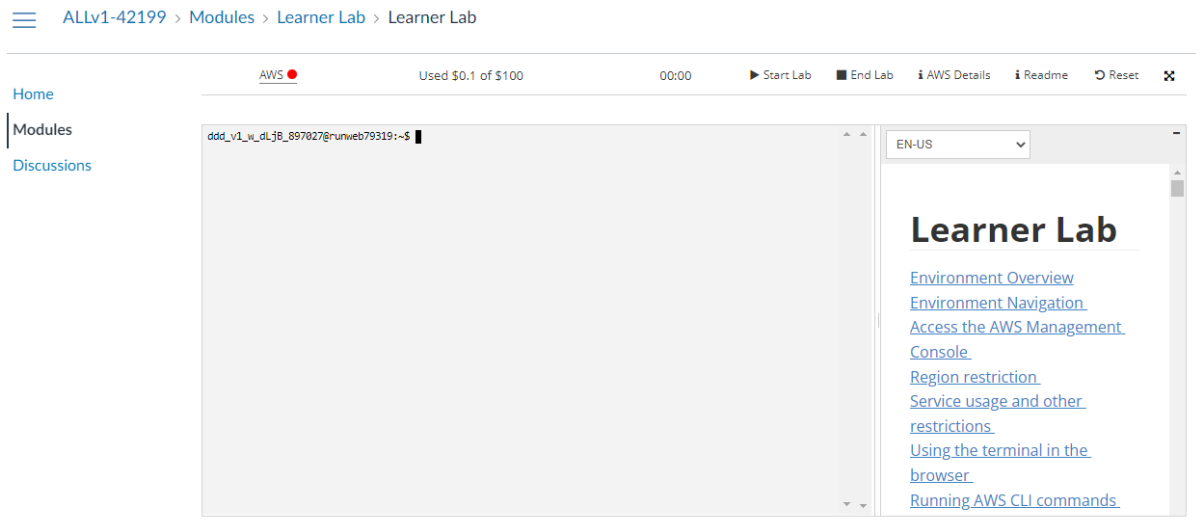
To deploy a VM in AWS and execute a simple application on it.

DEPLOYING VM IN AWS:

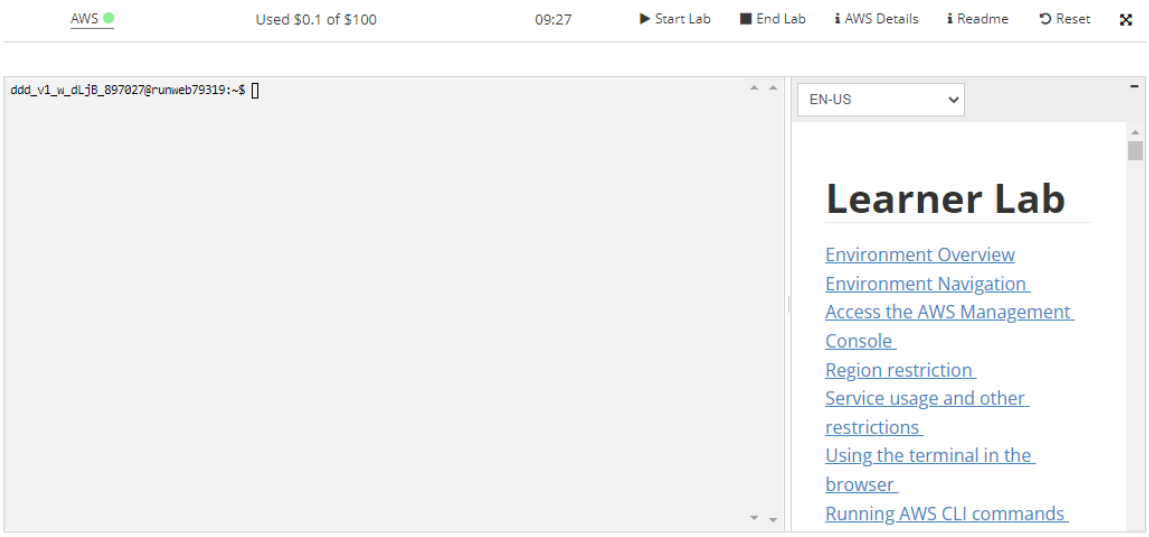
1. Login to aws academy using credentials.



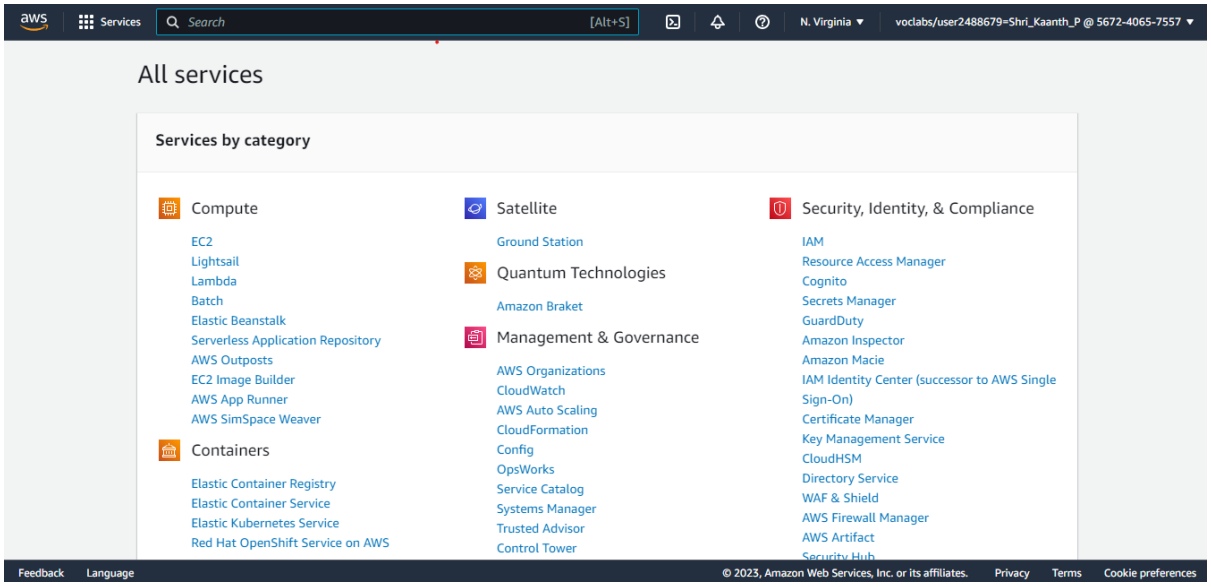
2. In the dashboard, select Modules → Learner Lab. The Learner lab will be opened.



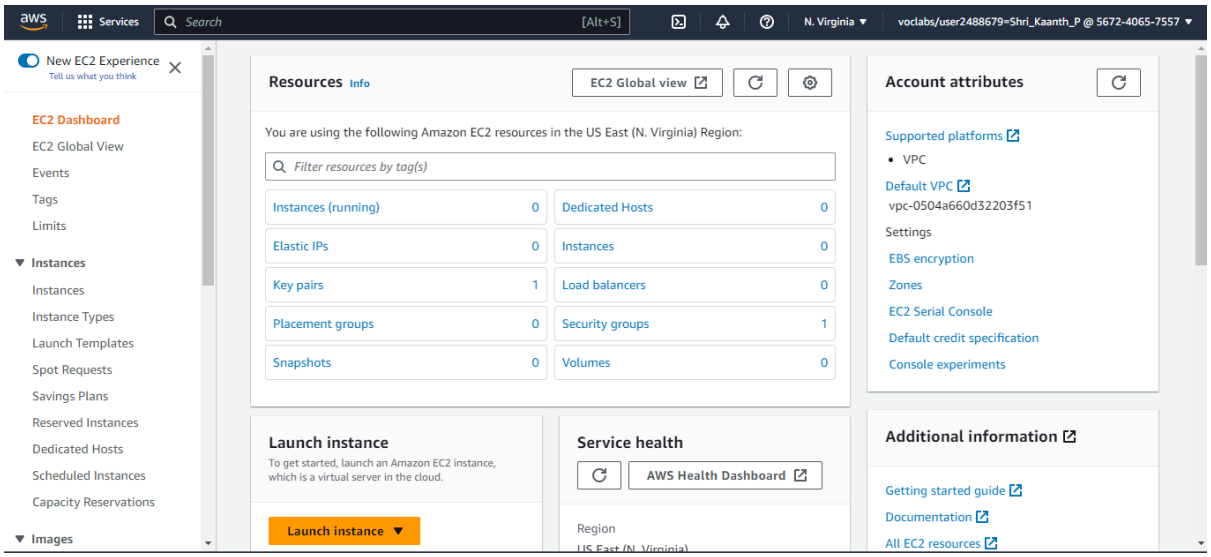
3. The red signal indicates that the lab has not started yet. To start the learner lab, click **Start Lab**. Wait for some time until the lab starts. The yellow signal indicates that the lab is being started.After the lab starts, the red signal changes to green, indicating the lab has started.



4. Click on **AWS to open** AWS Management Console. In the following page, click on **EC2.**

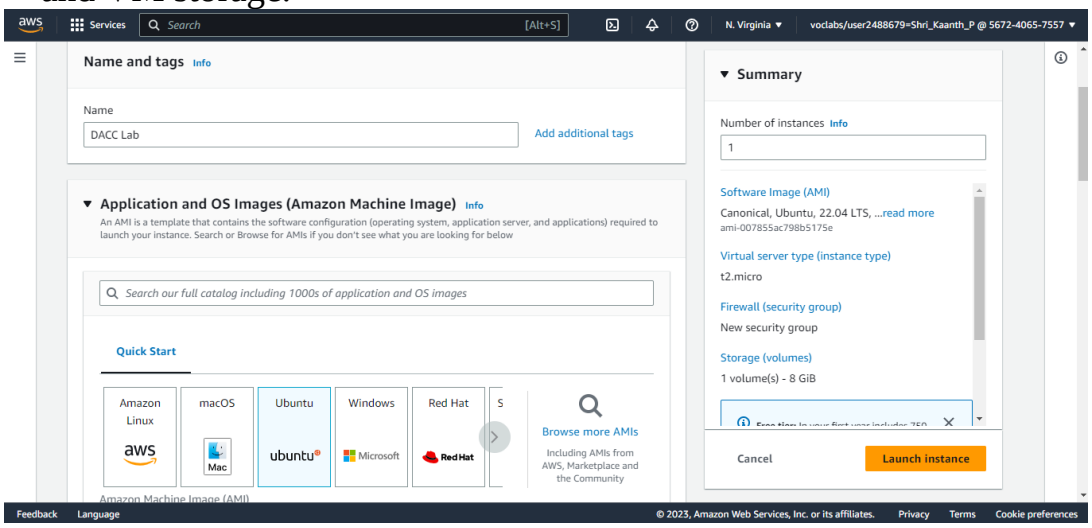


5. Click on **Instances** to view the EC2 instances created. To create a new instance, click **Launch Instance**

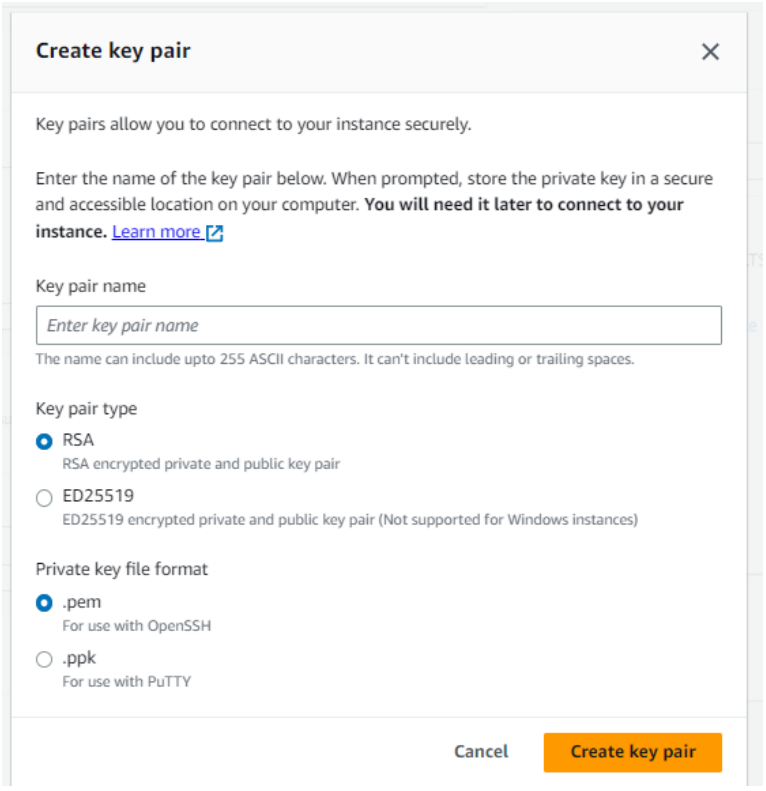


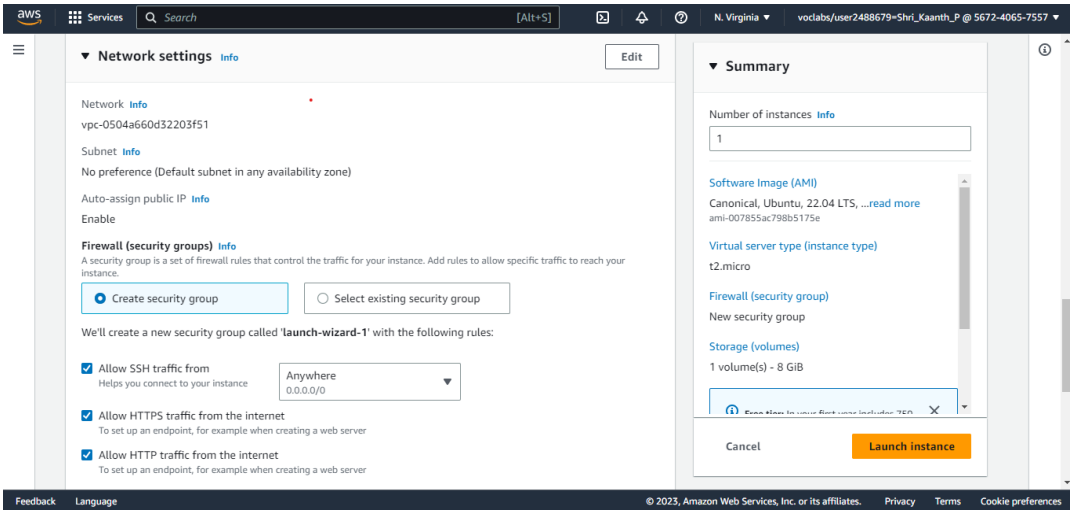
EXP NO : 09 DATE: 15-03-23	MONGODB BASICS

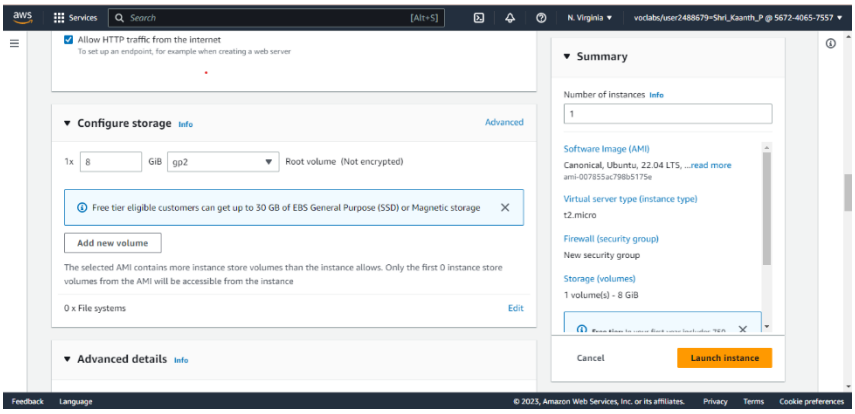
6. Select name, Application and OS image, appropriate networok settings and VM storage.



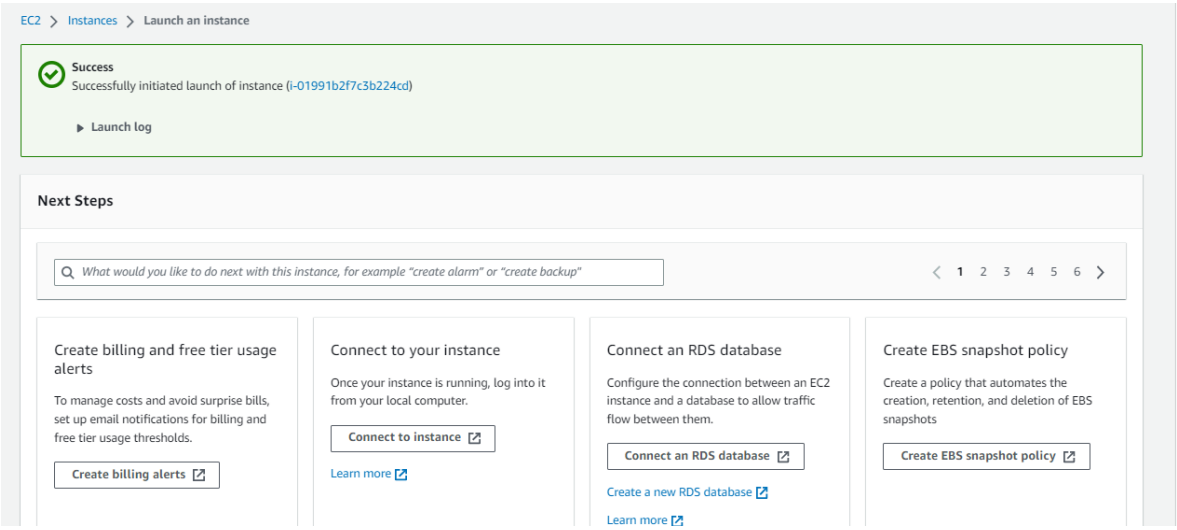
For key-pair attribute, click on **Create new key pair .** Enter the name of the key pair and select .pem for using in OpenSSH and click **Create key pair**. A .pem file will be downloaded.



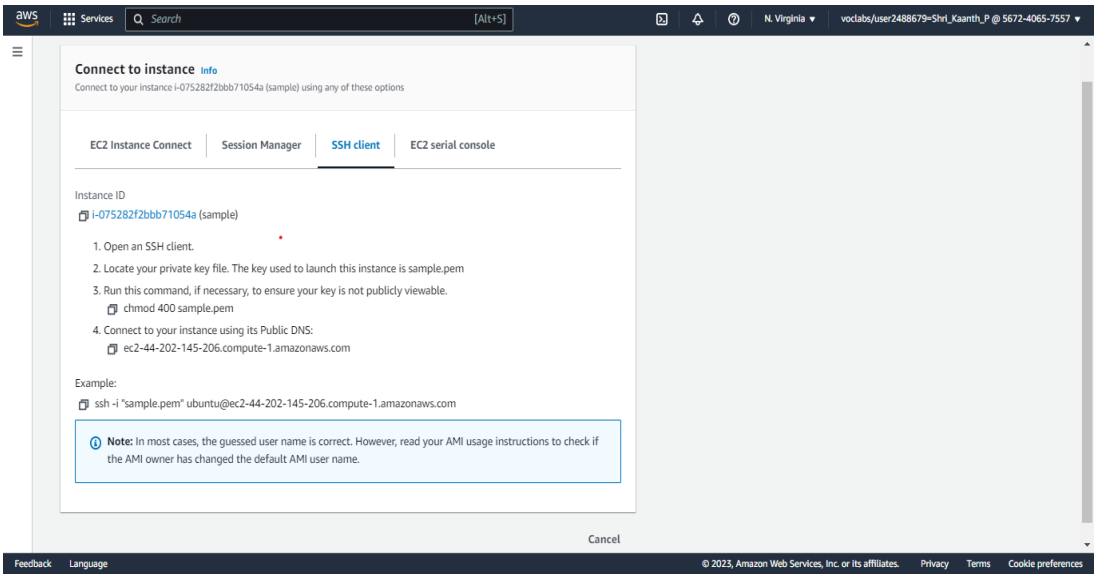




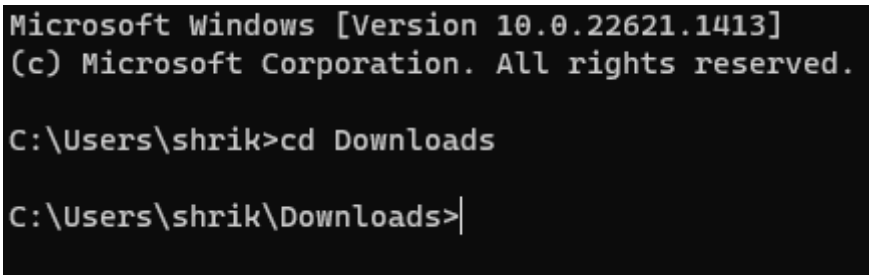
7. Click **Launch instance** to create the new instance. Your EC2 instance is created. After creating EC2 instance, click **Connect to insance.**



8. Go to **SSH client** section and copy the example given.



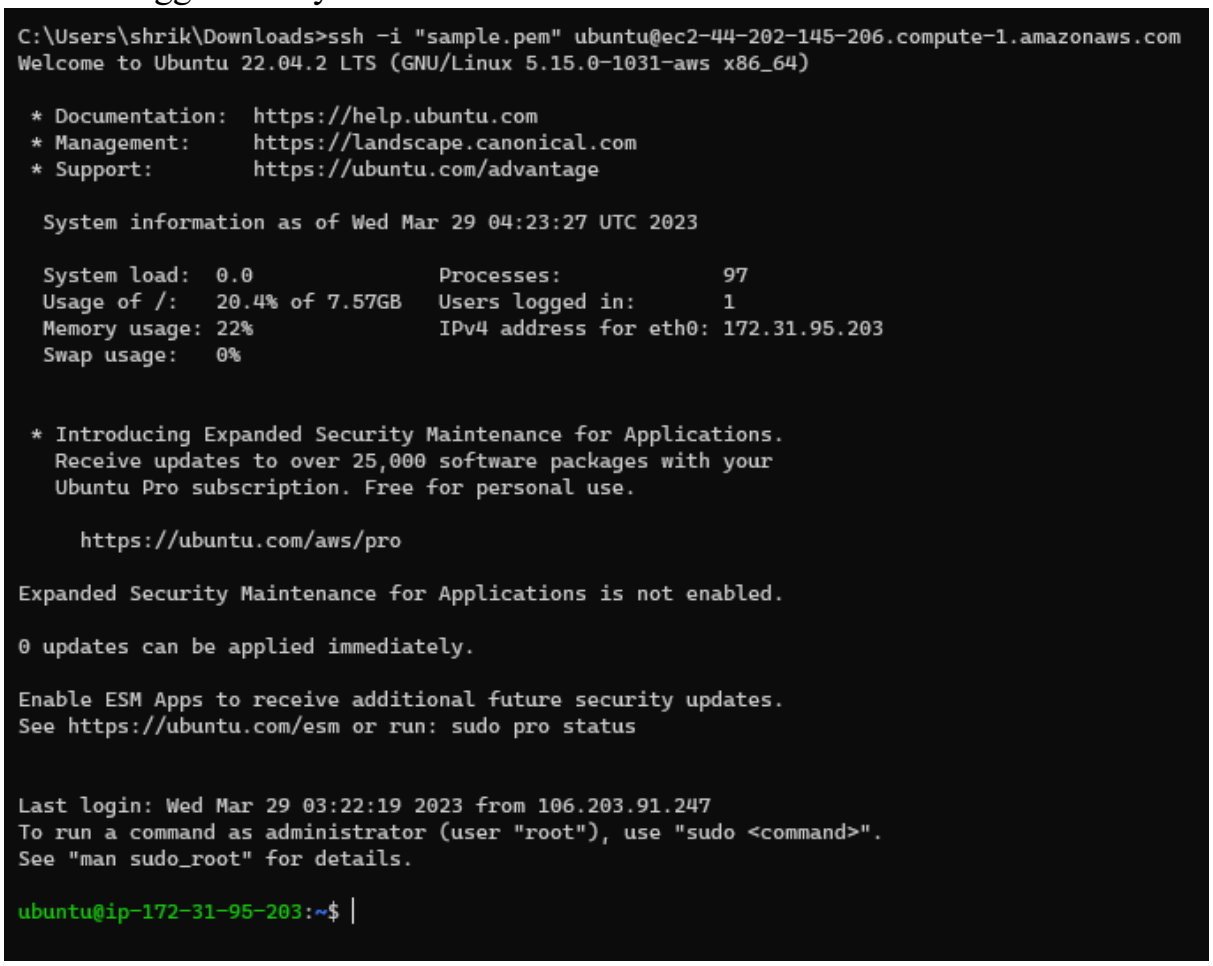
9. Now, open command prompt using start menu. Then go to the directory where the .pem file is saved. By default, it is saved in **Downloads** folder.



10. Paste the command that is copied from the SSH client section after creating the instance and press Enter.

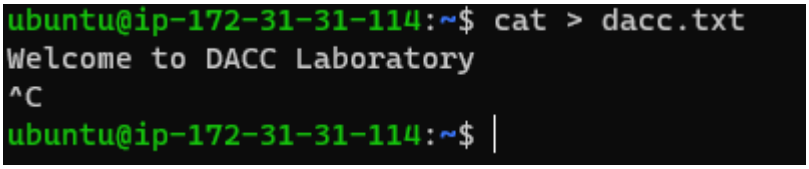
```
ssh -i "dacc lab keypair.pem" ubuntu@ec2-54-147-58-180.compute-1.amazonaws.com
```

You've logged in to your virtual machine.



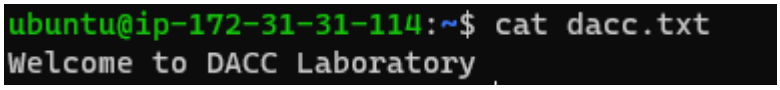
11. Enter the command below to create and edit a new file. Enter the file content and press Ctrl+c.

```
cat > sample.txt
```



12. Enter the command below to display the file content.

```
cat dacc.txt
```



Result:

Thus, VM has been deployed in OpenStack and a simple application has been executed.

EXP NO : 09 **DATE:** 29-03-23	**MONGODB BASICS**

1. Study on MongoDB

Aim:

To study about the basics of MongoDB and its operations.

Theory:

MongoDB is a general-purpose document database designed for modern application development and for the cloud. Its scale-out architecture allows you to meet the increasing demand for your system by adding more nodes to share the load.

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.

Document

MongoDB stores data as JSON documents. The document data model maps naturally to objects in application code, making it simple for developers to learn and use. 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. Documents can be nested to express hierarchical relationships and to store structures such as arrays.

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.

id is a 12 bytes hexadecimal number which assures the uniqueness of every document. The 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.

Sample document

```
{
        _id: ObjectId("6418761848304d22ba39626b"),
        emp_name: 'Ram',
        emp_id: 1001,
        age: 25,
        dept: 'Sales'
}
```

Operations

1. Use

To create and use a database. If the database exists, it returns existing database

```
use employee
```

```
local> use employee
switched to db employee
```

2. Create

To create/insert a new document to a collection

insertOne( )

```
db.data.insertOne({emp_name:”Ram”,emp_id:1001,age:25,
dept:”Sales”})
```

```
employee> db.data.insertOne({emp_name:"Ram",emp_id:1001,age:25,dept:"Sales"})
{
  acknowledged: true,
  insertedId: ObjectId("6418761848304d22ba39626b")
}
```

insertMany( )

```
db.data.insertMany([{emp_name:”Sam”,emp_id:1002,age:30,dept:”Accounts”}
,{emp_name:”Raju”,emp_id:1003,age:35,dept:”Management”},{emp_name:”B
alu”,emp_id:1004,age:40,dept:”Production”}])
```

```
employee> db.data.insertMany([{emp_name:"Sam",emp_id:1002,age:30,dept:"Accounts"},{emp_name:"Raju",emp_id:1003,dept:"Management",age:35},{emp_id:1004,emp_name:"Balu",a
ge:40,dept:"Production"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("641876be48304d22ba39626c"),
    '1': ObjectId("641876be48304d22ba39626d"),
    '2': ObjectId("641876be48304d22ba39626e")
  }
}
```

3. Read

To retrieve documents from a collection

```
db.data.find()
```

```
employee> db.data.find()
[
  {
    _id: ObjectId("6418761848304d22ba39626b"),
    emp_name: 'Ram',
    emp_id: 1001,
    age: 25,
    dept: 'Sales'
  },
  {
    _id: ObjectId("641876be48304d22ba39626c"),
    emp_name: 'Sam',
    emp_id: 1002,
    age: 30,
    dept: 'Accounts'
  },
  {
    _id: ObjectId("641876be48304d22ba39626d"),
    emp_name: 'Raju',
    emp_id: 1003,
    dept: 'Management',
    age: 35
  },
  {
    _id: ObjectId("641876be48304d22ba39626e"),
    emp_id: 1004,
    emp_name: 'Balu',
    age: 40,
    dept: 'Production'
  }
]
```

4. Update

To modify existing documents in a collection

```
db.data.updateOne({emp_id:1003},{$set:{age:50}})
```

```
employee> db.data.updateOne({emp_id:1003},{$set:{age:50}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.data.find({emp_id:1003})
[
  {
    _id: ObjectId("641876be48304d22ba39626d"),
    emp_name: 'Raju',
    emp_id: 1003,
    dept: 'Management',
    age: 50
  }
]
```

5. Delete

To remove documents from a collection

```
db.data.deleteMany({age:{$lt:38}})
```

```
employee> db.data.updateOne({emp_id:1003},{$set:{age:50}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.data.find({emp_id:1003})
[
  {
    _id: ObjectId("641876be48304d22ba39626d"),
    emp_name: 'Raju',
    emp_id: 1003,
    dept: 'Management',
    age: 50
  }
]
```

Result:

A student management system has been designed using MongoDB and CRUD operations have been performed on it successfully.

EXP NO : 10 **DATE:** 12-04-23	**HADOOP INSTALLATION**

Aim:

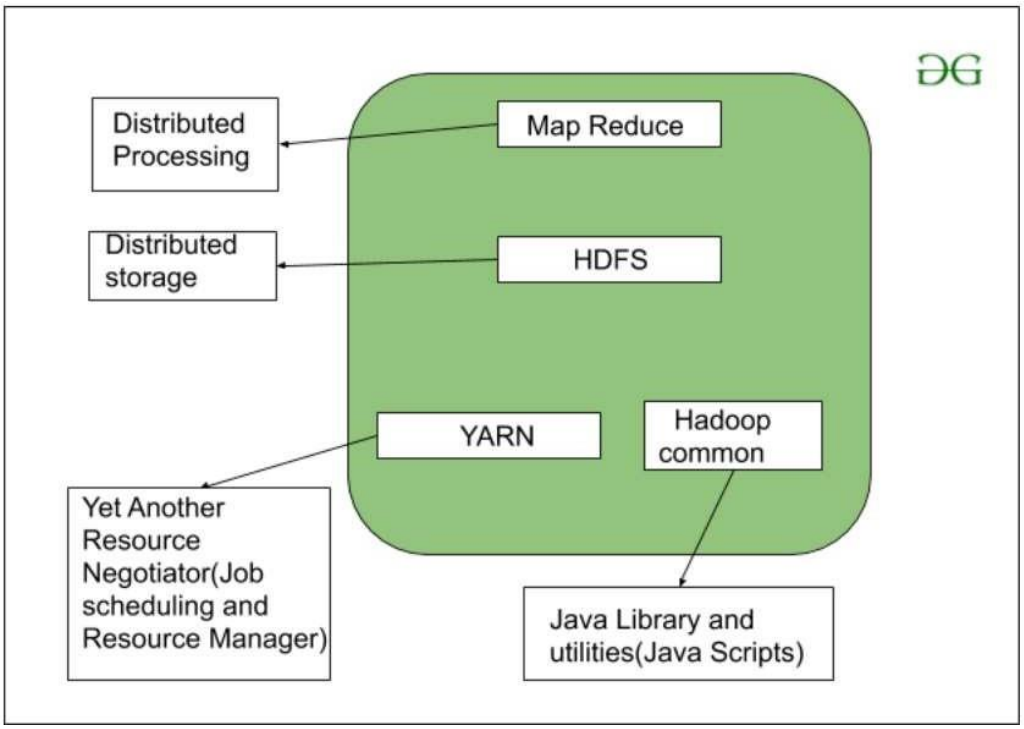
To study about the architecture of Hadoop and its components.

Theory:

Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google.

The Hadoop Architecture Mainly consists of 4 components:

- MapReduce
- HDFS(Hadoop distributed File System)
- YARN(Yet Another Resource Framework)
- Common Utilities or Hadoop Common



Components of Hadoop Architecture

1. **MapReduce**

MapReduce is an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop fast. MapReduce has 2 main tasks which are divided phase-wise. In first phase, Map is utilized and in next phase Reduce is utilized.

The Input is provided to the Map( ) function and then its output is used as an input to the Reduce( ) function to receive the final output. The Input is a set of Data. The Map() function breaks data blocks into tuples that are key-value pairs. These key-value pairs are now sent as input to the Reduce( ). The Reduce( ) function then combines the broken key-value pairs based on its key value and form set of tuples. Some operation like sorting, summation, etc. is performed which is then sent to the final Output Node which generates the final output.

2. **HDFS**

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave. Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

The NameNode manages the file system namespace by executing an operation like the opening, renaming and closing the files. The HDFS cluster contains several DataNodes each of which contains multiple data blocks that store data. The DataNodes read and write requests from the file system's clients.

3. **YARN**

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job scheduler is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and processing can be maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information

like job timing, etc. And the use of resource manager is to manage all the resources that are made available for running a Hadoop cluster.

4. Hadoop Common

Hadoop common or Common utilities are nothing but Java library and Java files or the java scripts that are needed for all the other components present in a Hadoop cluster. These utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common

INSTALL AND CONFIGURE HADOOP IN ITS TWO OPERATING MODES:

HADOOP SINGLE-NODE CLUSTER INSTALLATION

1. Download the Java 8 Package. Save this file in your home directory.
2. Extract the Java Tar File.

```
tar -xvf jdk-8u101-linux-i586.tar.gz
```

3. Download the Hadoop 2.7.3 Package.

```
https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

4. Extract the Hadoop tar File.

```
tar -xvf hadoop-2.7.3.tar.gz
```

5. Add the Hadoop and Java paths in the bash file (.bashrc).

   Open. bashrc file. Now, add Hadoop and Java Path as shown below.

```
vi .bashrc
```

```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

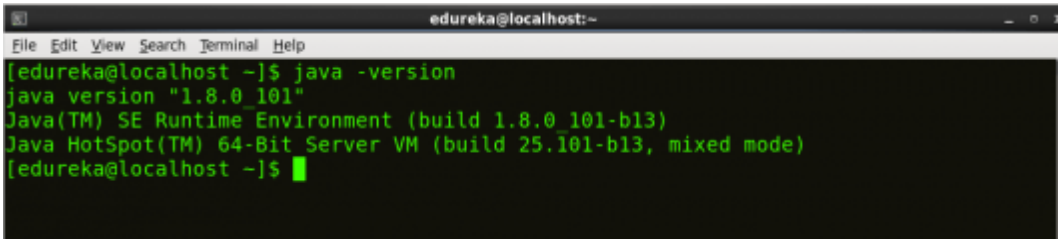
Then, save the bash file and close it. For applying all these changes to the current Terminal, execute the source command.

```
source .bashrc
```

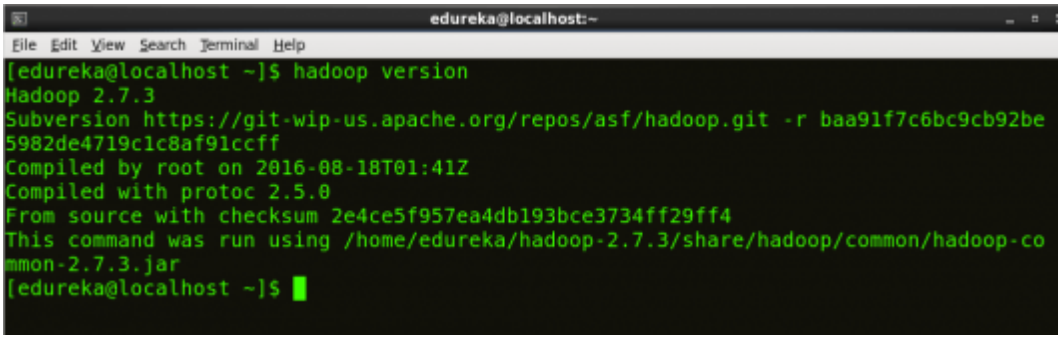
```
edureka@localhost:~
File  Edit  View  Search  Terminal  Help
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$
```

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.

```
java -version
```

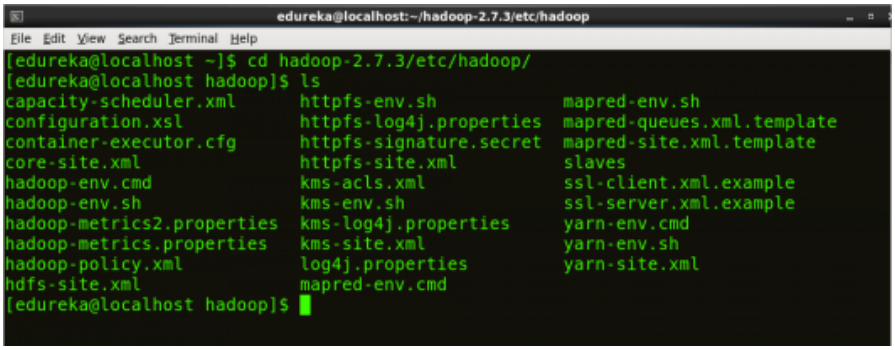



```
hadoop version
```



6. Edit the Hadoop Configuration files.

```
cd hadoop-2.7.3/etc/hadoop/
ls
```



7. Open core-site.xml and edit the property mentioned below inside configuration tag:

vi core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

8. Open hdfs-site.xml and edit the property mentioned below inside configuration tag:

vi hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
</configuration>
```

9. Open mapred-site.xml and edit the property mentioned below inside configuration tag:

vi mapred-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

10. Open yarn-site.xml and edit the property mentioned below inside configuration tag:

```
vi yarn-site.xml
```

```
<?xml version="1.0">
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

11. Edit hadoop-env.sh and add the Java Path as mentioned below:

```
vi hadoop-env.sh
```

```
# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

12. Go to Hadoop home directory and format the NameNode.

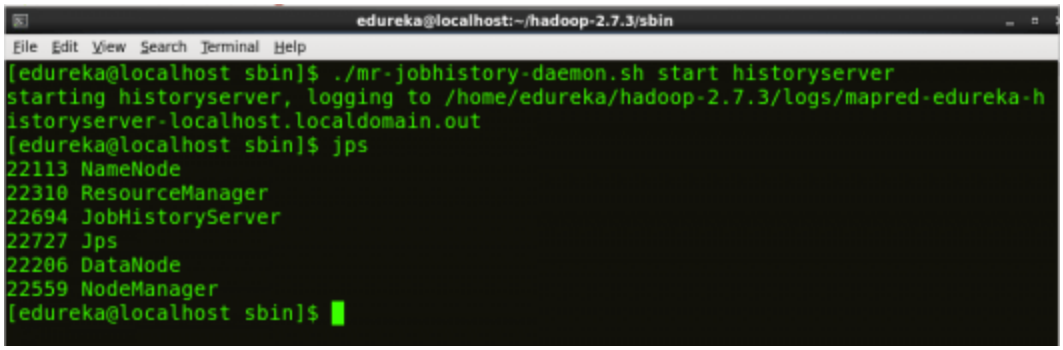
```
cd
cd Hadoop-2.7.3
bin/Hadoop namenode -format
```

13. Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons.

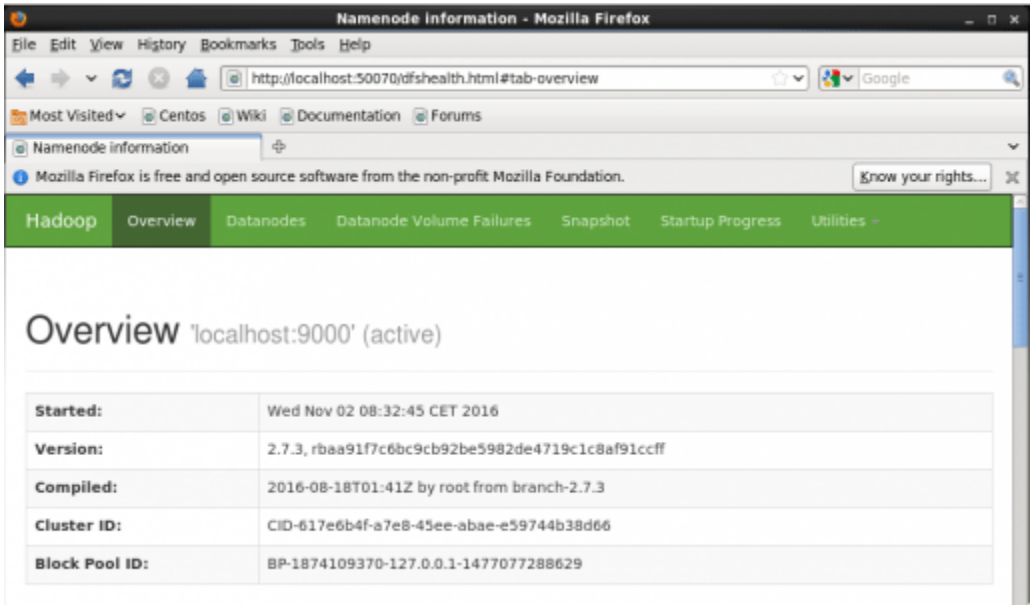
```
cd hadoop-2.7.3/sbin
./start-all.sh
```

14. To check that all the Hadoop services are up and running, run the below command.

```
jps
```



15. Now open the Mozilla browser and go to localhost:50070/dfshealth.html to check the NameNode interface.

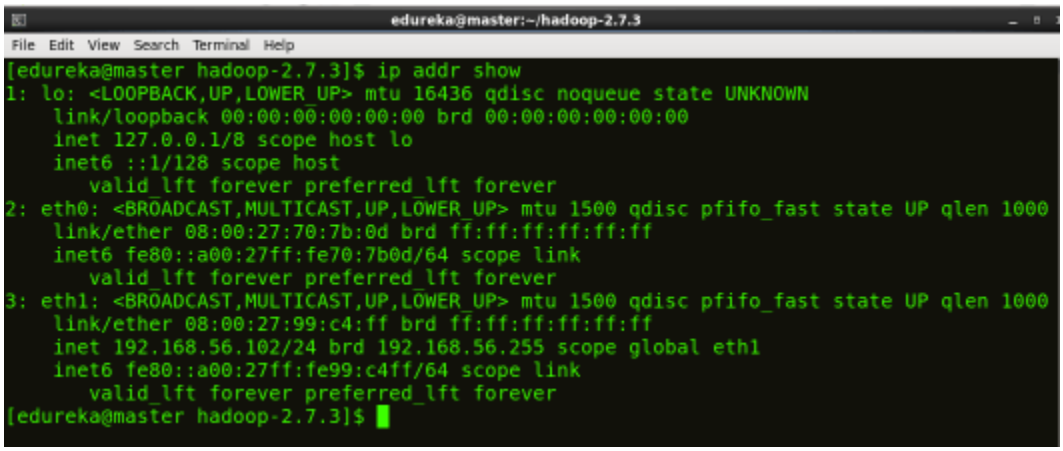


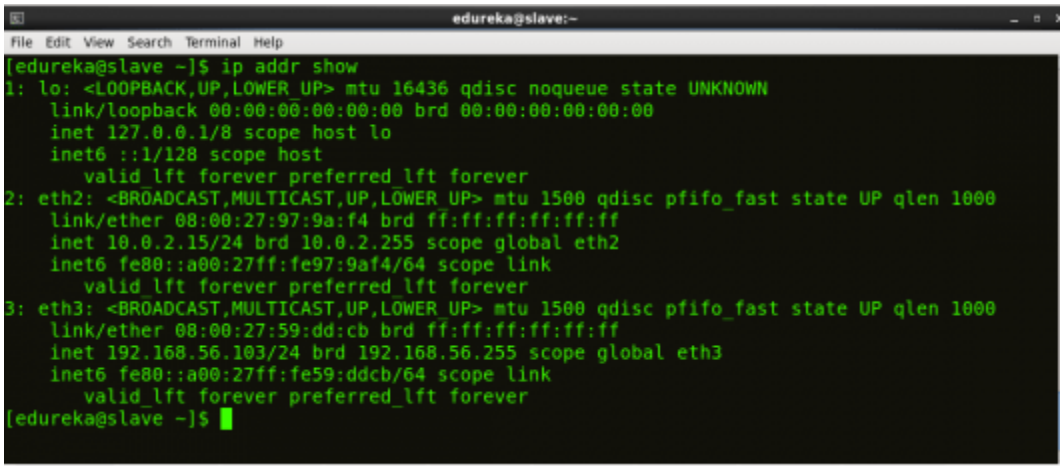
Thus, Single-node Hadoop cluster is installed.

HADOOP MULTI -NODE CLUSTER INSTALLATION

1. Check the IP address of all machines.

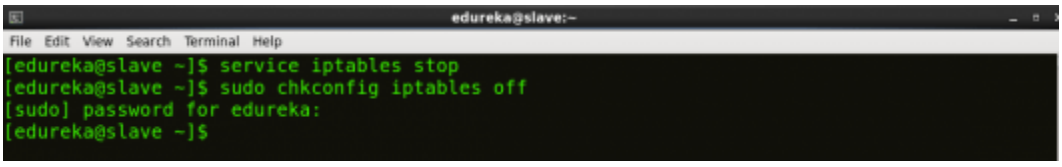
```
ip addr show
```





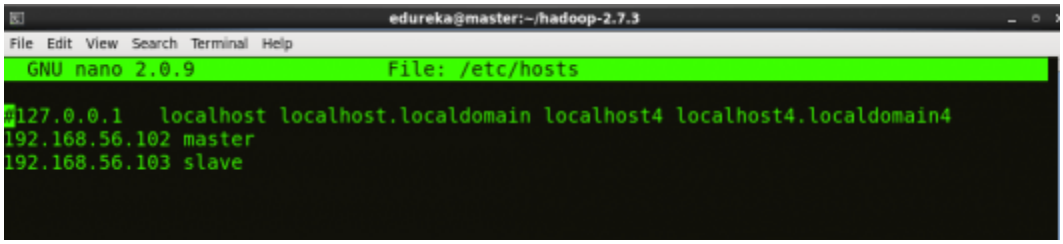
2. Disable the firewall restrictions.

```
service iptables stop
sudo chkconfig iptables off
```



3. Open hosts file to add master and data node with their respective IP addresses.

```
sudo nano /etc/hosts
```

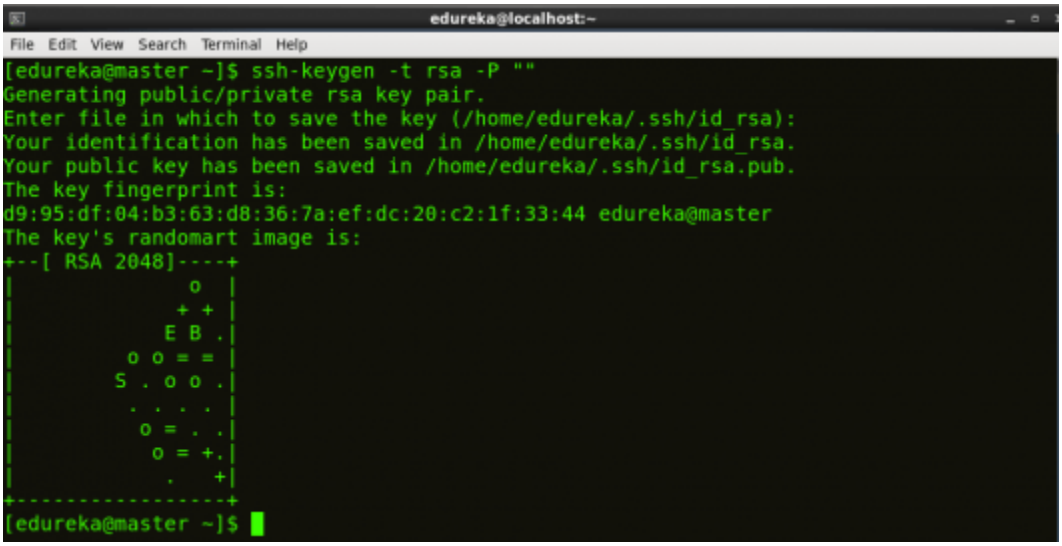


4. Restart the sshd service.

```
service sshd restart
```

5. Create the SSH Key in the master node. (Press enter button when it asks you to enter a filename to save the key).

```
ssh-keygen -t rsa -P “”
```



6. Copy the generated ssh key to master node's authorized keys.

```
cat $HOME/.ssh/id_rsa.pub >> HOME/.ssh/authorized_keys
```

7. Copy the master node's ssh key to slave's authorized keys.

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub edureka@slave
```

8. Download the Java 8 Package. Save this file in your home directory.

9. Extract the Java Tar File.

```
tar -xvf jdk-8u101-linux-i586.tar.gz
```

10. Download the Hadoop 2.7.3 Package.

```
https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

11. Extract the Hadoop tar File.

```
tar -xvf hadoop-2.7.3.tar.gz
```

12. Add the Hadoop and Java paths in the bash file (.bashrc).

Open. bashrc file. Now, add Hadoop and Java Path as shown below.

```
vi .bashrc
```

```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

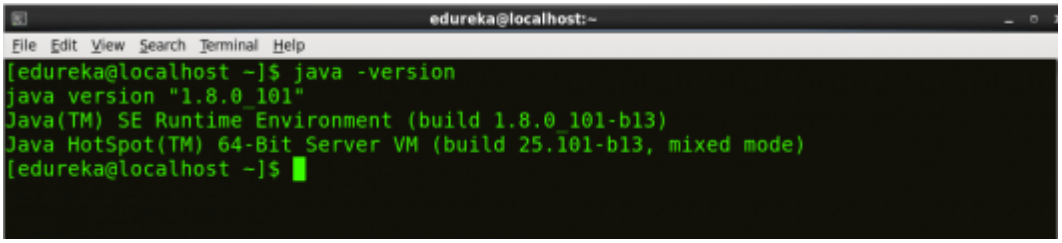
Then, save the bash file and close it. For applying all these changes to the current Terminal, execute the source command.

```
source .bashrc
```

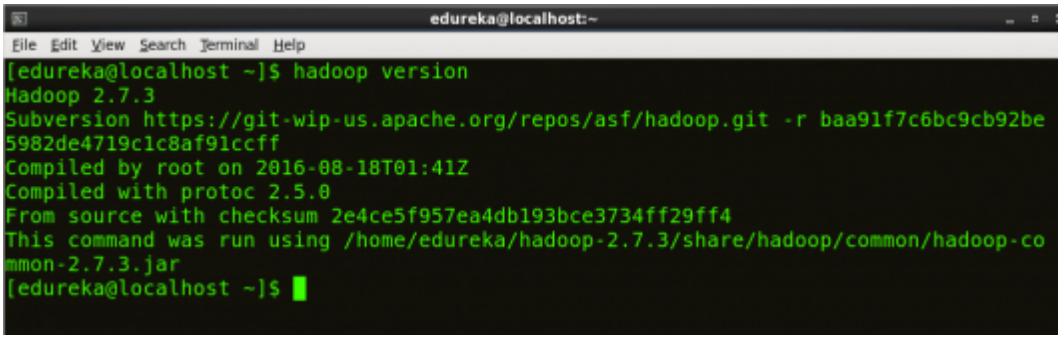
```
edureka@localhost:~
File  Edit  View  Search  Terminal  Help
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$
```

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.

```
java -version
```

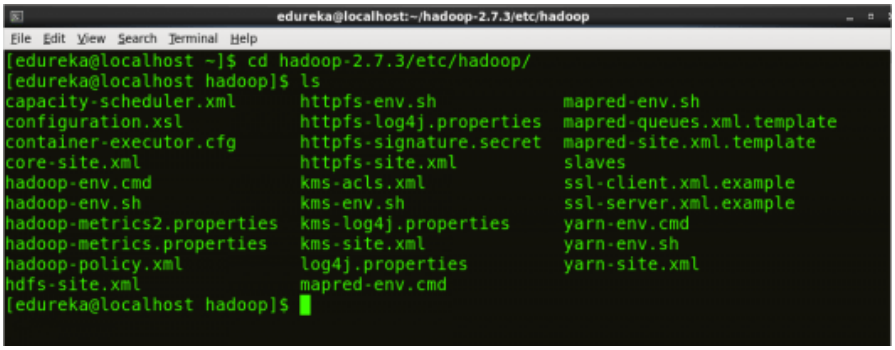



```
hadoop version
```



Edit the Hadoop Configuration files.

```
cd hadoop-2.7.3/etc/hadoop/
ls
```



13. Create masters file and edit as follows in both master and slave machines as below:

```
sudo gedit masters
```

14. Edit slaves file in master machine as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/slaves
```

15. Edit slaves file in slave machine as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/slaves
```

16. Edit core-site.xml on both master and slave machines as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/core-site.xml
```

17. Edit hdfs-site.xml on master machine as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/hdfs-site.xml
```

18. Edit core-site.xml on slave machine as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/core-site.xml
```

19. Copy mapred-site from the template in configuration folder and the edit mapred-site.xml on both master and slave machines as follows:

```
cp mapred-sitw.xml.template mapred-site.xml
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/mapred-site.xml
```

20.Edit yarn-site.xml on both master and slave machines as follows:

```
sudo gedit /home/edureka/hadoop-2.7.3/etc/hadoop/yarn-site.xml
```

21.Format the namenode (Only on master machine).

```
hadoop namenode -format
```

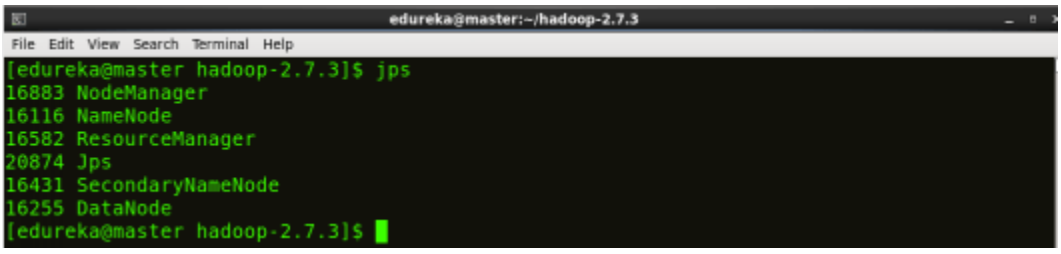
22.Start all daemons (Only on master machine).

```
./sbin/start-all.sh
```

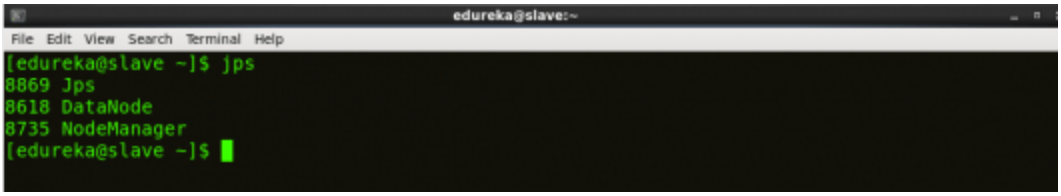
23.Check all the daemons running on both master and slave machines.

```
jps
```

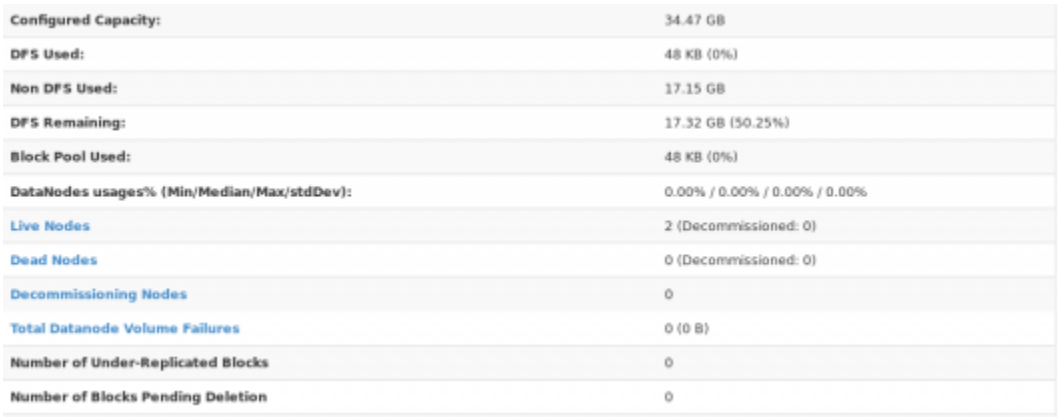
On master:



On slave:



24. Open the browser and go to master:50070/dfshealth.html on your master machine, this will give you the NameNode interface.



Configured Capacity:	34.47 GB
DFS Used:	48 KB (0%)
Non DFS Used:	17.15 GB
DFS Remaining:	17.32 GB (50.25%)
Block Pool Used:	48 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0

Thus, Multi-node Hadoop cluster is installed.

RESULT:

Thus, Hadoop has been successfully installed.

EXP NO : 11 DATE: 19-04-23	VISUALIZATION TOOLS IN PYTHON

Aim:

To study about the visualization tools in python that are used for data analytics.

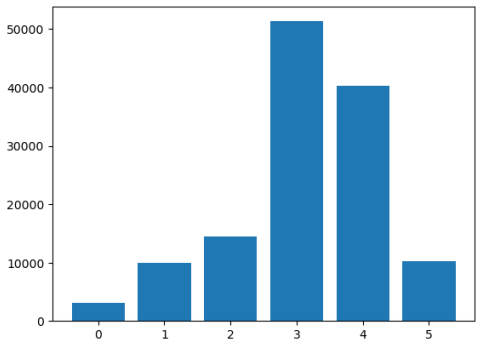
IMPORTING PACKAGES AND DATASET

```
import pandas as pd
import numpy as np
df = pd.read_csv("dataset.csv")
df.head()
```

	PM2.5	PM10	NO2	SO2	O3	AQI
0	104.00	148.50	23.00	15.30	117.62	3.0
1	94.50	142.00	16.25	17.00	136.23	3.0
2	82.75	126.50	14.83	15.40	149.92	3.0
3	68.50	117.00	13.60	21.80	161.70	3.0
4	69.25	112.25	11.80	21.38	161.68	3.0

1. BARPLOT

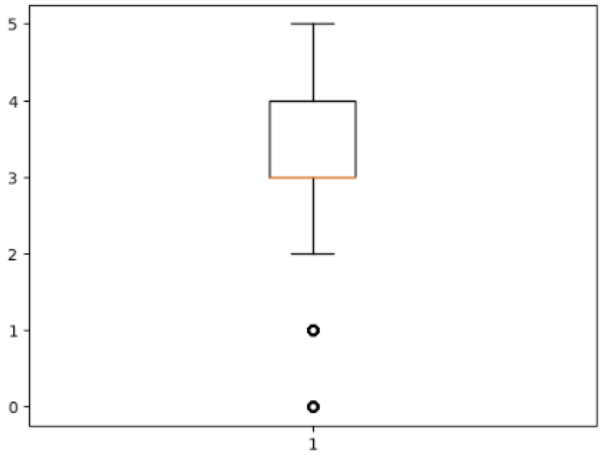
```
import matplotlib.pyplot as plt
plot = df['AQI'].value_counts()
plt.bar(plot.index,plot.values)
```



2. BOXPLOT

```
plt.boxplot(df['AQI'])
```

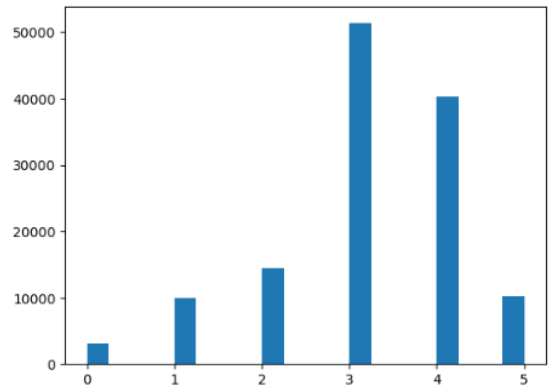
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2193de25e20>,
  <matplotlib.lines.Line2D at 0x2193de39160>],
 'caps': [<matplotlib.lines.Line2D at 0x2193de39430>,
  <matplotlib.lines.Line2D at 0x2193de39700>],
 'boxes': [<matplotlib.lines.Line2D at 0x2193de25b50>],
 'medians': [<matplotlib.lines.Line2D at 0x2193de399d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2193de39ca0>],
 'means': []}
```



3. HISTOGRAM

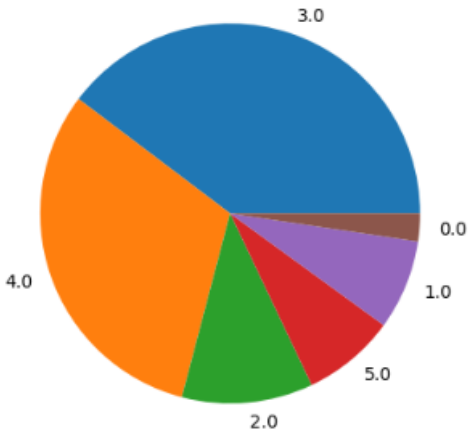
```
plt.hist(df['AQI'],bins=20)
```

```
(array([ 3047.,     0.,     0.,     0.,  9896.,     0.,     0.,     0.,
        14425.,     0.,     0.,     0., 51330.,     0.,     0.,     0.,
        40336.,     0.,     0., 10243.]),
 array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  , 2.25, 2.5 ,
        2.75, 3.  , 3.25, 3.5 , 3.75, 4.  , 4.25, 4.5 , 4.75, 5.  ]),
 <BarContainer object of 20 artists>)
```



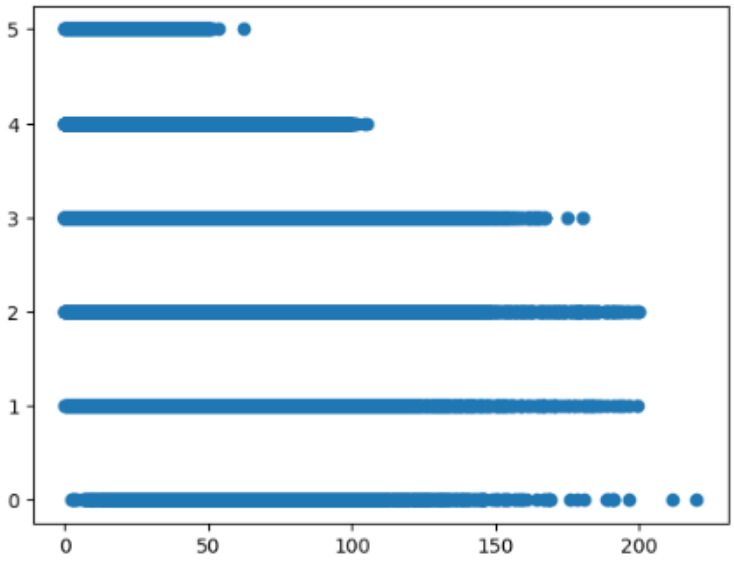
4. PIECHART

```
plt.pie(df['AQI'].value_counts(),labels=df['AQI'].value_counts().index)
```



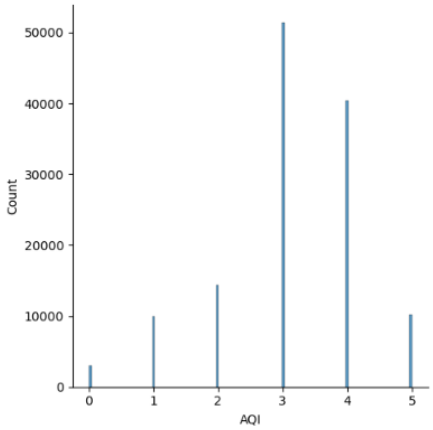
5. SCATTERPLOT

```
plt.scatter(df['O3'],df['AQI'])
```



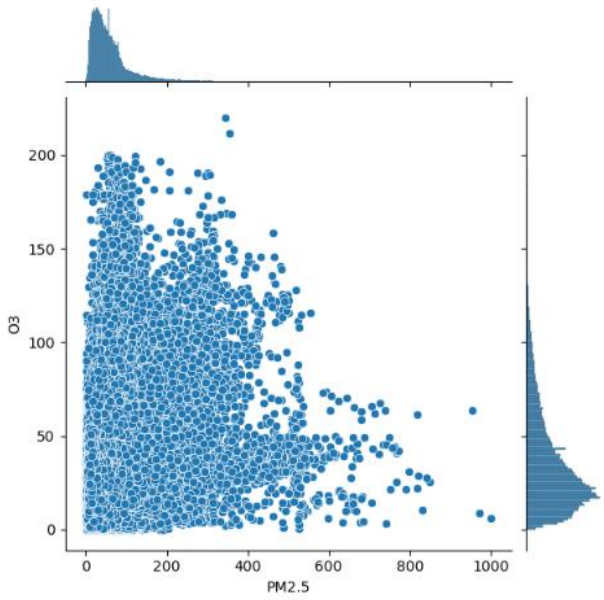
6. SEABORN

```
import seaborn as sns
sns.displot(df['AQI'])
```



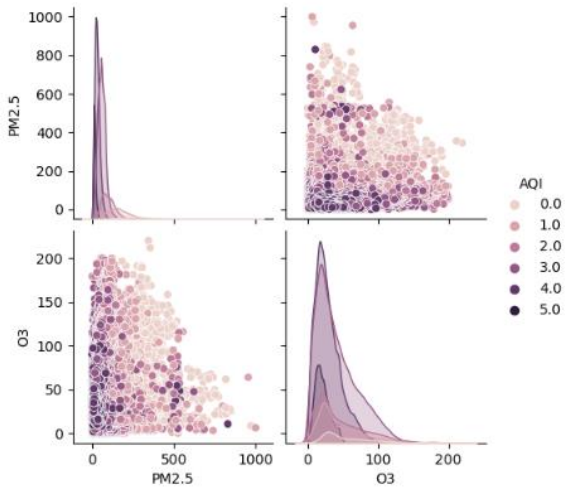
7. JOINTPLOT

```
sns.jointplot(x='PM2.5',y='O3',data=df)
```



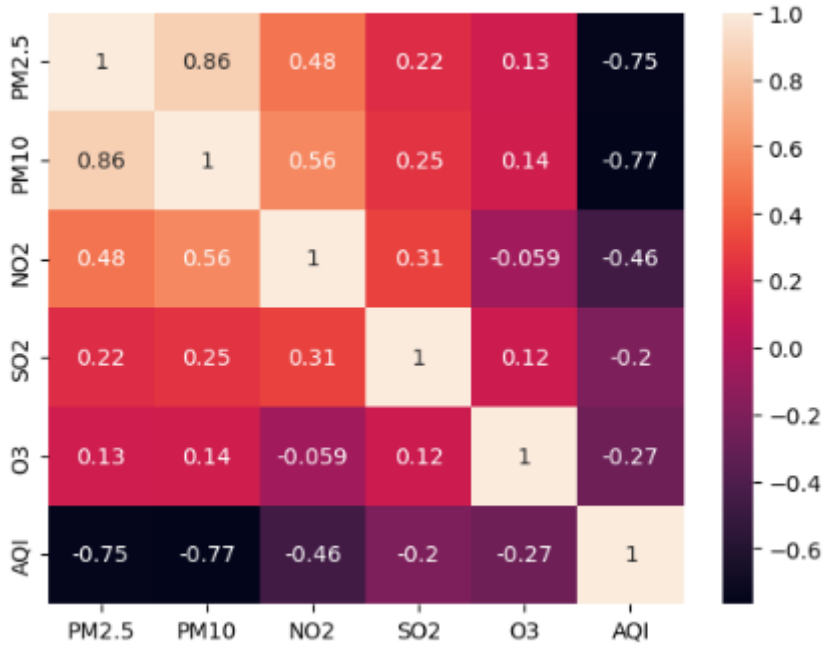
8. PAIRPLOT

```
sns.pairplot(df,vars=['PM2.5','O3'],hue='AQI')
```



9. HEATMAP

```
sns.heatmap(df.corr(),annot=True)
```



RESULT:

Thus, a study on the visualization tools in Python used for data analysis has been made.