MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5312 – DATA ANALYTICS AND CLOUD COMPUTING

LAB MANUAL

Prepared by

Dr. D. Sangeetha

Mr. K. Govindasamy

Vision of the Department

To educate students with conceptual knowledge and technical skills in the field of Information Technology with moral and ethical values to achieve excellence in an academic, industry and research centric environment.

Mission of the Department

1. To inculcate in students a firm foundation in theory and practice of IT skills coupled with the thought process for disruptive innovation and research methodologies, to keep pace with emerging technologies.
2. To provide a conducive environment for all academic, administrative, and interdisciplinary research activities using state-of-the-art technologies.
3. To stimulate the growth of graduates and doctorates, who will enter the workforce as productive IT engineers, researchers, and entrepreneurs with necessary soft skills, and continue higher professional education with competence in the global market.
4. To enable seamless collaboration with the IT industry and Government for consultancy and sponsored research.
5. To cater to cross-cultural, multinational, and demographic diversity of students.
6. To educate the students on the social, ethical, and moral values needed to make significant contributions to society.

Program Educational Objectives (PEOs)

PEO1: Demonstrate core competence in basic engineering and mathematics to design, formulate, analyze, and solve hardware/software engineering problems.

PEO2: Develop insights in foundational areas of Information Technology and related engineering to address real-world problems using digital and cognitive technologies.

PEO3: Collaborate with industry, academic and research institutions for state-of-the-art product development and research.

PEO4: Inculcate a high degree of professionalism, effective communication skills and team spirit to work on multidisciplinary projects in diverse environments.

PEO5: Practice high ethical values and technical standards.

Program Specific Outcomes (PSOs):

PSO1: Ability to apply programming principles and practices for the design of software solutions in an internet-enabled world of business and social activities.

PSO2: Ability to identify the resources to build and manage the IT infrastructure using the current technologies in order to solve real world problems with an understanding of the tradeoffs involved in the design choices.

PSO3: Ability to plan, design and execute projects for the development of intelligent systems with a focus on the future.

IT5612 DATA ANALYTICS AND CLOUD COMPUTING LABORATORY L T P C

0 0 4 2

OBJECTIVES:

To provide hands-on experience to cloud and data analytics frameworks and tools.

To use the Python packages for performing analytics.

To learn using analytical tools for real world problems.

To familiarize the usage of distributed frameworks for handling voluminous data.

To write and deploy analytical algorithms as MapReduce tasks.

LABORATORY EXERCISES:

Analytics Using Python:

1. Download, install and explore the features of NumPy, SciPy, Jupyter, Statsmodels and Pandas packages.

(i) Reading data from text file, Excel and the web.

(ii) Exploring various commands for doing descriptive analytics on Iris data set.

2. Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:

(i) Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

(ii) Bivariate analysis: Linear and logistic regression modeling

(iii) Multiple Regression analysis

Also compare the results of the above analysis for the two data sets.

3. Apply Bayesian and SVM techniques on Iris and Diabetes data set.

4. Apply and explore various plotting functions on UCI data sets.

Cloud Computing:

5. Installation of OpenStack.

6. Creation of VMs and installing applications and executing simple programs in

OpenStack.

7. Simple applications for communication across VMs.

Hadoop, MapReduce, HDFS, Hive:

8. Install and configure Hadoop in its two operating modes: Pseudo distributed and fully distributed.
9. Implement the following file management tasks in Hadoop: Adding files and directories, retrieving files and deleting files.
10. Create a retail database with the following tables: Product, Customer, Manufacturer, Shipping and Time using MongoDB and perform data replication using sharding techniques.
11. Install HIVE and implement the above retail schema definition and perform CRUD operations.

OUTCOMES:

On completion of the course, the students will be able to:

CO1: Install analytical tools and configure distributed file system.

CO2: Have skills in developing and executing analytical procedures in various distributed frameworks and databases.

CO3: Develop, implement and deploy simple applications on very large datasets.

CO4: Implement simple to complex data modeling in NoSQL databases.

CO5: Develop and deploy simple applications in OpenStack cloud.

CO6: Implement real world applications by using suitable analytical framework and tools.

Mapping of Course Outcomes (COs) with Program Outcomes (POs)

Course Outcomes (COs)	Program Outcomes (POs)											
	1	2	3	4	5	6	7	8	9	10	11	12
Install analytical tools and configu distributed file system	✓	✓	✓	✓					✓		✓	✓
Have skills in developing and executing analytical procedures in various distributed frameworks and databases	✓	✓	✓	✓	✓				✓		✓	✓
Develop, implement and deploy simple applications on very large datasets.	✓	✓	✓	✓	✓				✓		✓	✓
Implement simple to complex data modeling in NoSQL database	✓	✓	✓		✓				✓			
Develop and deploy simple applications in OpenStack cloud	✓	✓	✓	✓					✓			
Implement real world applications by using suitable analytical framework and tools	✓	✓	✓	✓	✓						✓	✓

SNO	NAME OF EXPERIMENT
1.	Identification and reading of the dataset
2. a	Descriptive Analysis
2. b	Univariate Analysis
2. c	Bivariate Analysis
2. d	Multivariate Analysis
3. a	Logistic Regression
3. b	Multiple Regression
4.	Classification using Naïve bayes classifier and Support Vector Machine
5 a.	Open stack
5 b.	Open Stack Installation
5 c.	Creation of VMs in Open Stack
6 a.	Hadoop Installation
6 b.	Hadoop File Implementation
7 a.	Hive Installation
7 b.	Hive - CRUD operations
8 a.	MongoDB basics
8 b.	MongoDB Data replication
9.	Visualisation tool in python

Exp no :	Study on Numpy,Scipy, Statmodels 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

SNO	DESCRIPTION	EXAMPLE
1.	read_csv( ) : This helps to read a csv [comma separated-values] file into a pandas dataframe. It can also read files separated by delimiter	data_e = pd.read_csv (r'c:user_ds.csv')
2.	head( ) : The head(n) is used to return the first n rows of a dataset. By default, the function returns 5 rows of the dataframe	data_e.head(10)
3.	describe( ) : It is used to generate descriptive statistics of data in a pandas data frame or services	data_e.describe( )
4.	memory_usage( ) : It returns a pandas series having the memory usage of each column in a dataframe	data_e.memory_usage(deep = = true)
5.	loc[ ] : It helps to access a group of rows and columns in a dataset	data_e.loc[0:4, ['Name', 'Age', 'State']]

---

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.

SNO	DESCRIPTION	EXAMPLE
1.	numpyreshape : This function allows us to change the dimension of the array without hampering the data value	res = arr.reshape(3, 2)
2.	numpy.concatenate( ) : This function is used to join two arrays of same size, either in a row-wise or column-wise way	res = numpy.concatenate((arr1, arr2), axis =1)
3.	numpy.char.add( ) : It concatenates the data value of two arrays, merge them and represent a new array as a result	res = numpy.concatenate((arr1, arr2), axis =1)
4.	numpy.median( ) : It calculates the median of an ordered array	med = numpy.median(X)
5.	numpy.average( ) : It returns the average of all data values of the passed array	avg = numpy.average(X)

---

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.

SNO	DESCRIPTION	EXAMPLE
1.	special.logsumexp(x) : log sum exponential computes the log of sum exponential input element	np.log(np.sum (np.exp(a)))
2.	linalg.det : linear algebra of scipy is an implementation of BLAS and ALAS LAPACK libraries. It accepts 2D arrays and gives a @D array	ar = np.array([4, 5], [5, 2]) linalg.det(ar)
3.	linalg.eig( ) : The most common problem in linear algebra is Eigen value and Eigen value	eg_val, eg_vec = linalg.eig(ar)
4.	Scipy.sparse : It is used for creating sparse matrix using multiple data structures	from scipy.sparse import csr_matrixar = np.array([0, 0], [1, 1]) print(csr_matrix(ar).data)

---

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 Scipy

SNO	DESCRIPTION	EXAMPLE
1.	get_rdataset : It is used to download any dataset we want	data = sm.datasets. get_rdataset("Guerry", "Hist").data
2.	add_constant(X) : It is used to add a constant column to input dataset	X = sm.addconstant(X)
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.:	**Identification and reading of the dataset**

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

About the dataset:

IOT based sensor system is used to detect and capture the weather and surrounding environment. Contains Humidity, Temperature, Light, Co2, Humidity Ratio, Occupancy and Date columns. These are fed into the algo to detect anomalies.

CODES AND OUTPUTS:

a. Reading from Text File

Code:

```
lines = [ ]
with open("/content/datatest.txt") as f:
  lines = f.readlines()
count = 0
for line in lines:
  count += 1
  print(f'line {count}: {line}')
```

Output

```
line 167: "305","2015-02-02 17:04:00",22.6,25,433,832,0.00423823118799312,1
line 168: "306","2015-02-02 17:05:00",22.6,25,433,836.25,0.00423823118799312,1
line 169: "307","2015-02-02 17:06:00",22.6,25,428.2,832.6,0.00423823118799312,1
line 174: "312","2015-02-02 17:10:59",22.58,25.08,433,821.5,0.0042466904885792 4,1
```

b. Reading as a csv file (reading the text file into a dataframe)

Code:

```
df=pd.read_csv("/content/datatest.txt",encoding="UTF-8")
df
```

Output

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
140	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	
141	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	
142	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	
143	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	
144	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	
145	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	

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 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
Instrument Technology for the first time 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 pioneer in courses that it gave birth to, it is already renowned
……
```

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	**Descriptive Analysis**

Aim: to explore various commands for performing descriptive data analysis on the dataset

CODES AND OUTPUTS:

Importing the dataset and the packages necessary

```
import pandas as pd
df=pd.read_csv("/content/datatest.txt",encoding="UTF-8")
```

1. Head ()

```
df.head()
```

OUTPUT

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
140	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1
141	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1
142	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1
143	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1
144	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1

2. Columns

```
Df.columns
```

OUTPUT

```
Index(['date', 'Temperature', 'Humidity', 'Light', 'CO2', 'HumidityRatio', 'Occupancy'],
dtype='object')
```

3. Isnull()

```
newdf = df.isnull().sum()

print(newdf.to_string())
```

OUTPUT

```
date             0

Temperature      0
Humidity         0
Light            0
CO2              0
HumidityRatio    0
Occupancy        0
```

4. shape

```
df.shape()
```

OUTPUT

```
(2665,7)
```

5. dtypes

```
df.dtypes
```

OUTPUT

```
date            object
Temperature     float64
Humidity        float64
Light           float64
CO2             float64
HumidityRatio   float64
Occupancy       int64
dtype: object
```

6. value_count

```
df.value_counts("Humidity")
```

OUTPUT

```
Humidity
22.500000    93
25.000000    82
22.290000    63
22.200000    62
22.700000    51
             ..
26.080000     1
26.059167     1
26.050000     1
26.020000     1
31.472500     1
Length: 725, dtype: int64
```

7. drop_duplicates

```
data = df.drop_duplicates(subset ="Humidity",)
```

OUTPUT

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
140	2015-02-02 14:19:00	23.700000	26.272000	585.200000	749.200000	0.004764	1
141	2015-02-02 14:19:59	23.718000	26.290000	578.400000	760.400000	0.004773	1
142	2015-02-02 14:21:00	23.730000	26.230000	572.666667	769.666667	0.004765	1
...	...	...	...	...	...	...	...
2798	2015-02-04 10:37:00	24.290000	25.978000	793.000000	1145.400000	0.004882	1
2799	2015-02-04 10:38:00	24.290000	25.852000	801.400000	1140.800000	0.004858	1
2804	2015-02-04 10:43:00	24.408333	25.681667	798.000000	1124.000000	0.004860	1

725 rows × 7 columns

8. Mean, median and mode

```
df['Humidity'].mean()
df['Humidity'].median()
df['Humidity'].mode()
```

OUTPUT:

```
Mean: 25.353936799785583]

Median: 25.0

Mode: 0 22.5 dtype: float64
```

9. Describe

```
df.describe()
```

OUTPUT

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
count	2665.000000	2665.000000	2665.000000	2665.000000	2665.000000	2665.000000
mean	21.433876	25.353937	193.227556	717.906470	0.004027	0.364728
std	1.028024	2.436842	250.210906	292.681718	0.000611	0.481444
min	20.200000	22.100000	0.000000	427.500000	0.003303	0.000000
25%	20.650000	23.260000	0.000000	466.000000	0.003529	0.000000
50%	20.890000	25.000000	0.000000	580.500000	0.003815	0.000000
75%	22.356667	26.856667	442.500000	956.333333	0.004532	1.000000
max	24.408333	31.472500	1697.250000	1402.250000	0.005378	1.000000

10. Info

```
df.info()
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2665 entries, 140 to 2804
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           2665 non-null   object
 1   Temperature    2665 non-null   float64
 2   Humidity       2665 non-null   float64
 3   Light          2665 non-null   float64
 4   CO2            2665 non-null   float64
 5   HumidityRatio  2665 non-null   float64
 6   Occupancy      2665 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 166.6+ KB
```

Result:

Thus, the Python programs to perform descriptive analysis has been implemented successfully.

Exp no :2 b	**Univariate Analysis**

`

Aim: to explore various commands for performing univariate data analysis on the dataset

1. Var() and std()

```
df[“Humidtiy”].var()
df[“Humidtiy”].var()
```

OUTPUT

```
5.9382005
2.4368423
```

2. Skew()

```
From scipysats import skew
y1=df[“Humidity”]
print(“Skewness: “ , skew(y1))
```

OUTPUT

```
Skewness = 0.67238354
```

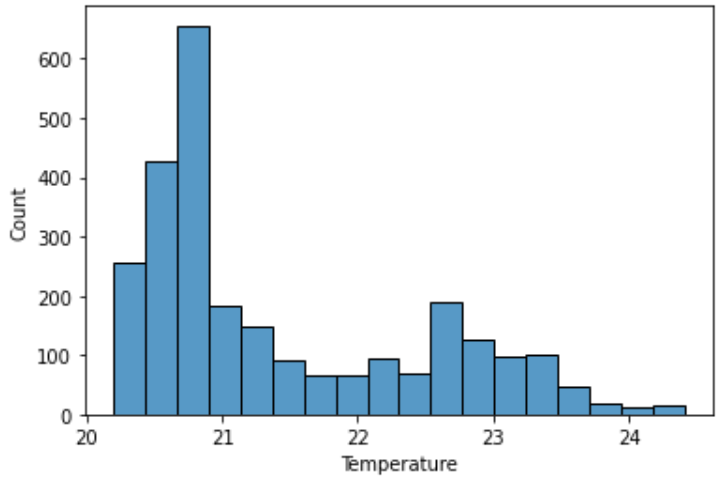
Importing seaborn and matplotlib

```
import seaborn as sns
import matplotlib.pyplot as plt
```

3. Histplot

```
sns.histplot(x="Temperature", data=df)
```

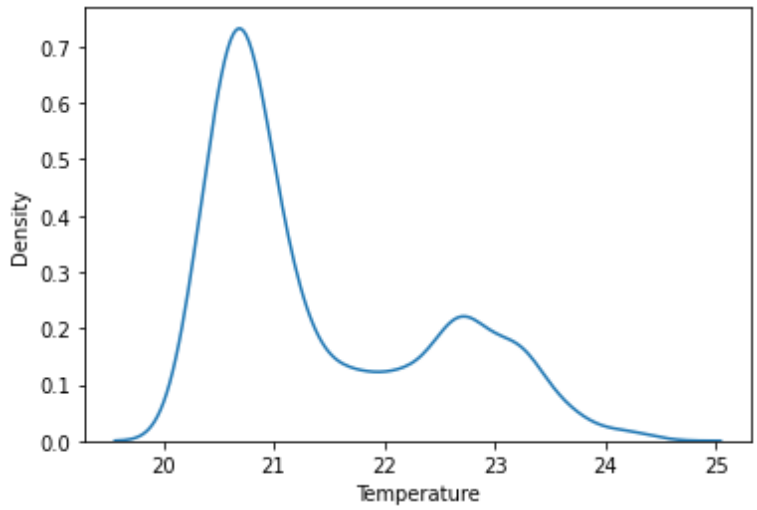
OUTPUT



4. Kdeplot

```
sns.kdeplot(x="Temperature", data=df)
```

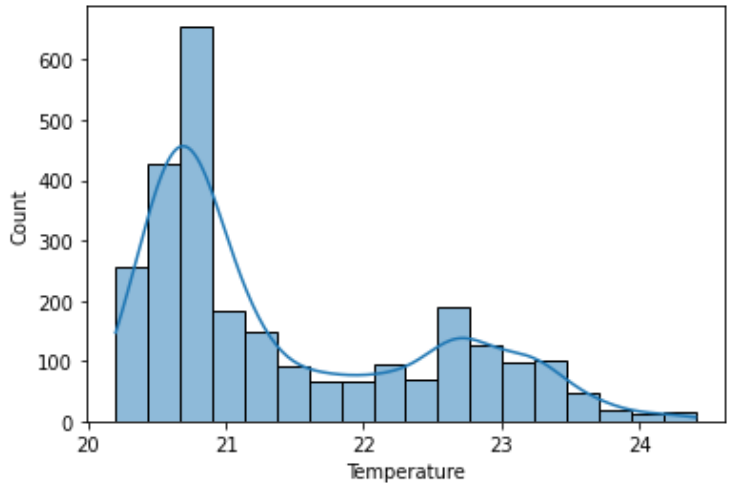
OUTPUT



5. Histplot() along with kdeplot

```
sns.histplot(x="Temperature", data=df, kde=True)
```

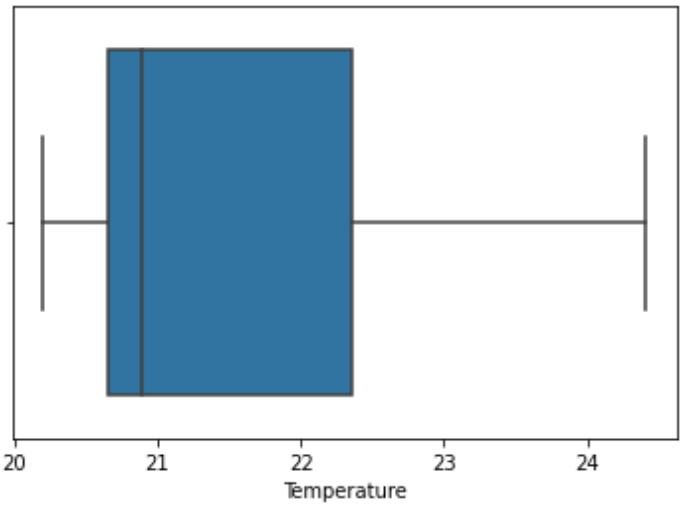
OUTPUT



6. Boxplot

```
sns.boxplot(x=df["Temperature"])
```

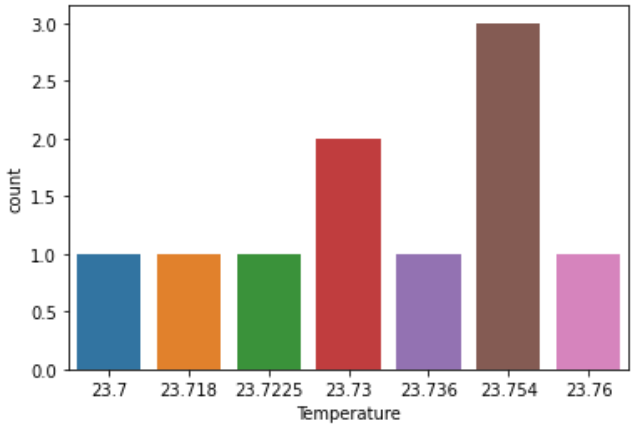
OUTPUT



7. Countplot

```
sns.countplot(x=df.head(10)["Temperature"])
```

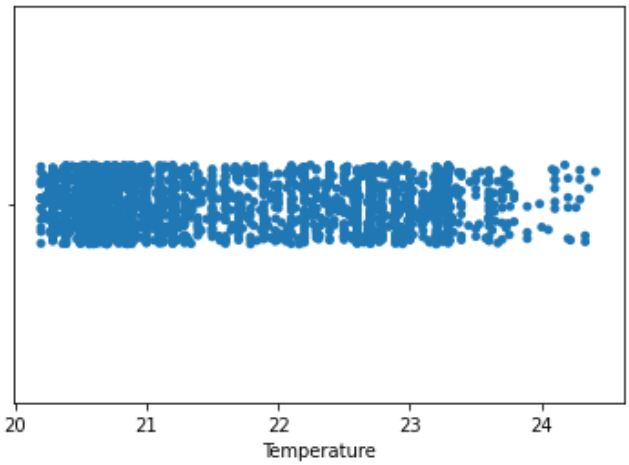
OUTPUT



8. Stripplot

```
sns.stripplot(x=df["Temperature"])
```

OUTPUT



Result:

Thus, the Python programs to perform univariate analysis has been implemented successfully.

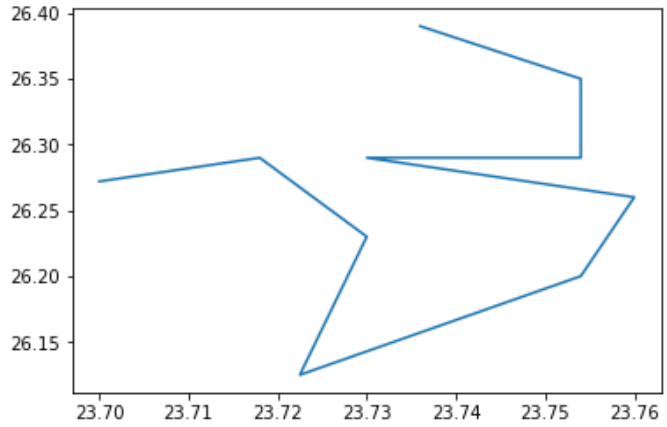
Exp no :2 c	**Bivariate Analysis**

Aim: to explore various commands for performing bivariate data analysis on the dataset

1. Lineplot

```
plt.plot(df.head(10)["Temperature"], df.head(10)["Humidity"])
plt.show()
```

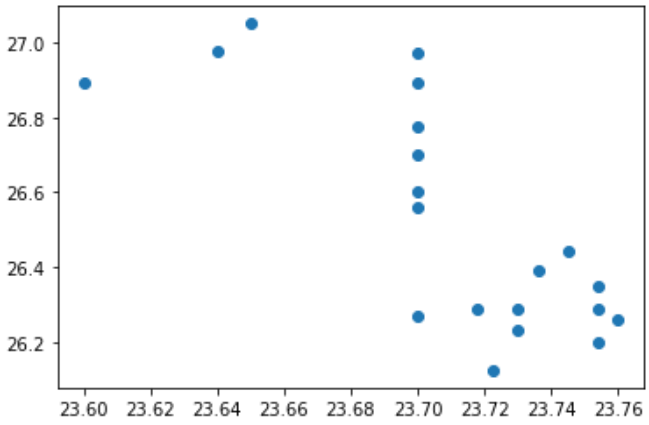
OUTPUT



2. Scatter plot

```
plt.scatter(df.head(20)["Temperature"], df.head(20)["Humidity"])
plt.show()
```

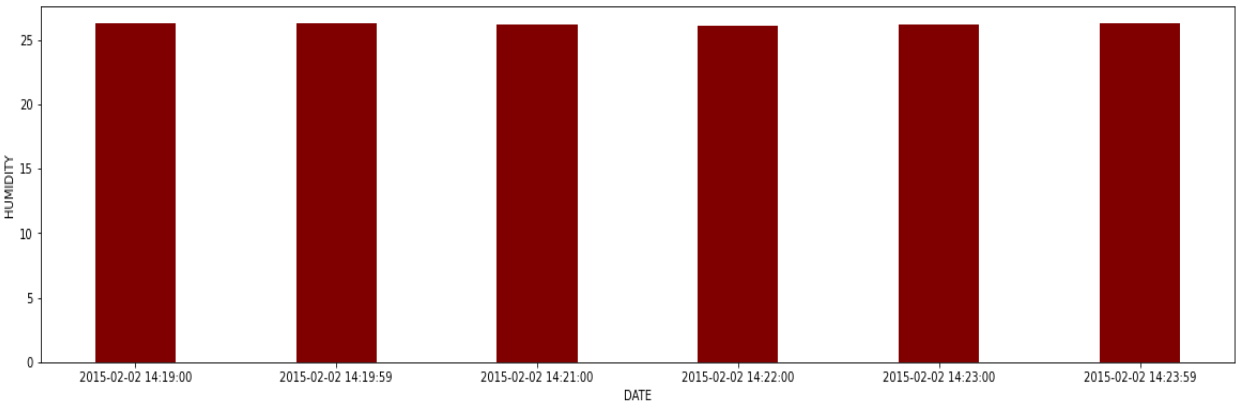
OUTPUT



3. Bar

```
fig = plt.figure(figsize = (20, 5))
plt.bar(df.head(6)["date"], df.head(6)["Humidity"], color ='maroon',width = 0.4)
plt.xlabel("DATE")
plt.ylabel("HUMIDITY")
plt.show()
```

OUTPUT



Result:

Thus, the Python programs to perform univariate analysis has been implemented successfully.

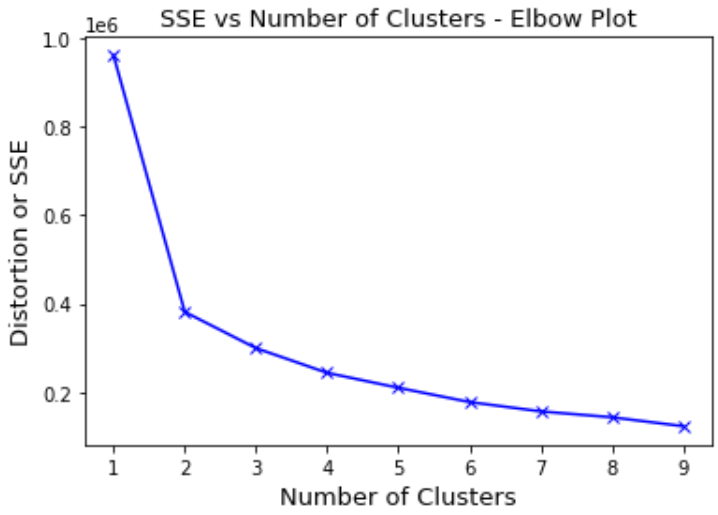
Exp no :2 c	**Multivariate Analysis**

Aim: to explore various commands for performing bivariate data analysis on the dataset

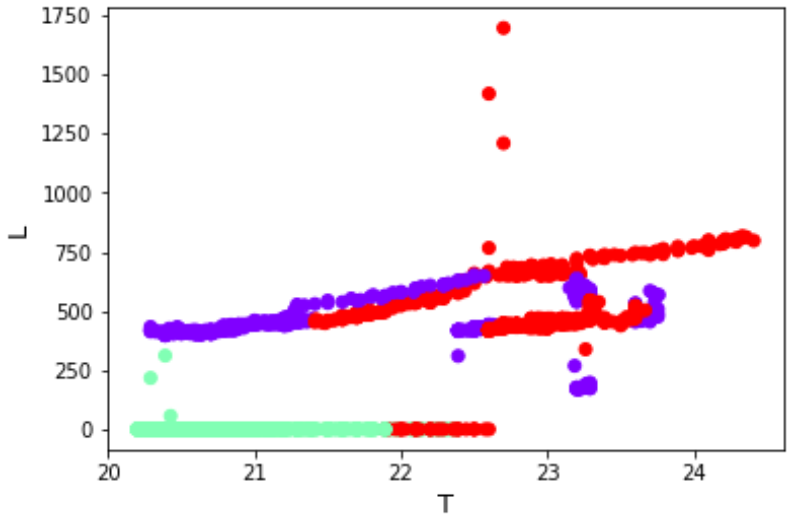
1. Clustering analysis

   First we find the number of cluster using the elbow graph and then cluster the data

```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
K = range(1,10)
distortions = []
interias = []
df1=df.drop(df.columns[[0]], axis=1)
for k in K:
  kmeans = KMeans(n_clusters=k).fit(df1)
  kmeans.fit(df1)
  distortions.append(sum(np.min(cdist(df1, kmeans.cluster_centers_, 'euclidean'), axis=1)))
  interias.append(kmeans.inertia_)
plt.plot(K, distortions, 'bx-')
plt.xlabel('Number of Clusters', fontsize=13)
plt.ylabel('Distortion or SSE', fontsize=13)
plt.title('SSE vs Number of Clusters - Elbow Plot', fontsize=13)
plt.show()
```

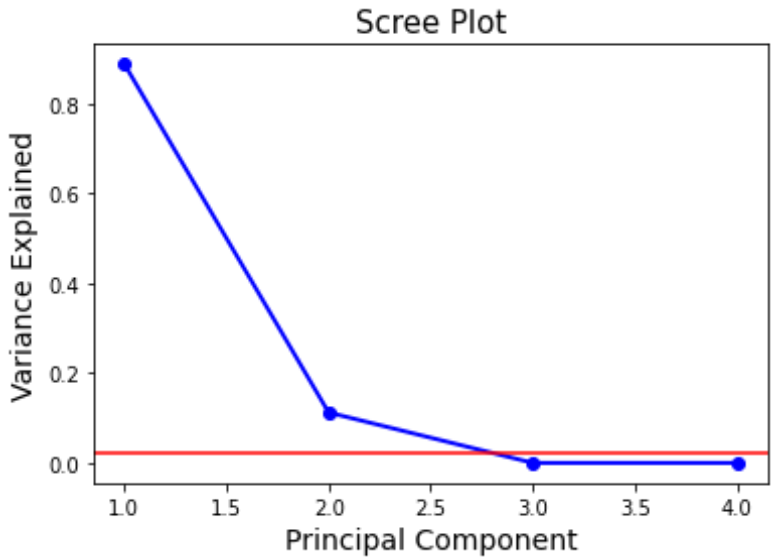


```
=kmeans = KMeans(n_clusters = 3)
kmeans.fit(df1)
df1['clusters'] = kmeans.fit_predict(df1)
plt.scatter(df1['Temperature'], df1['Light'], c=df1['clusters'], cmap='rainbow')
plt.xlabel("T", fontsize=13)
plt.ylabel("L", fontsize=13)
plt.show()
```



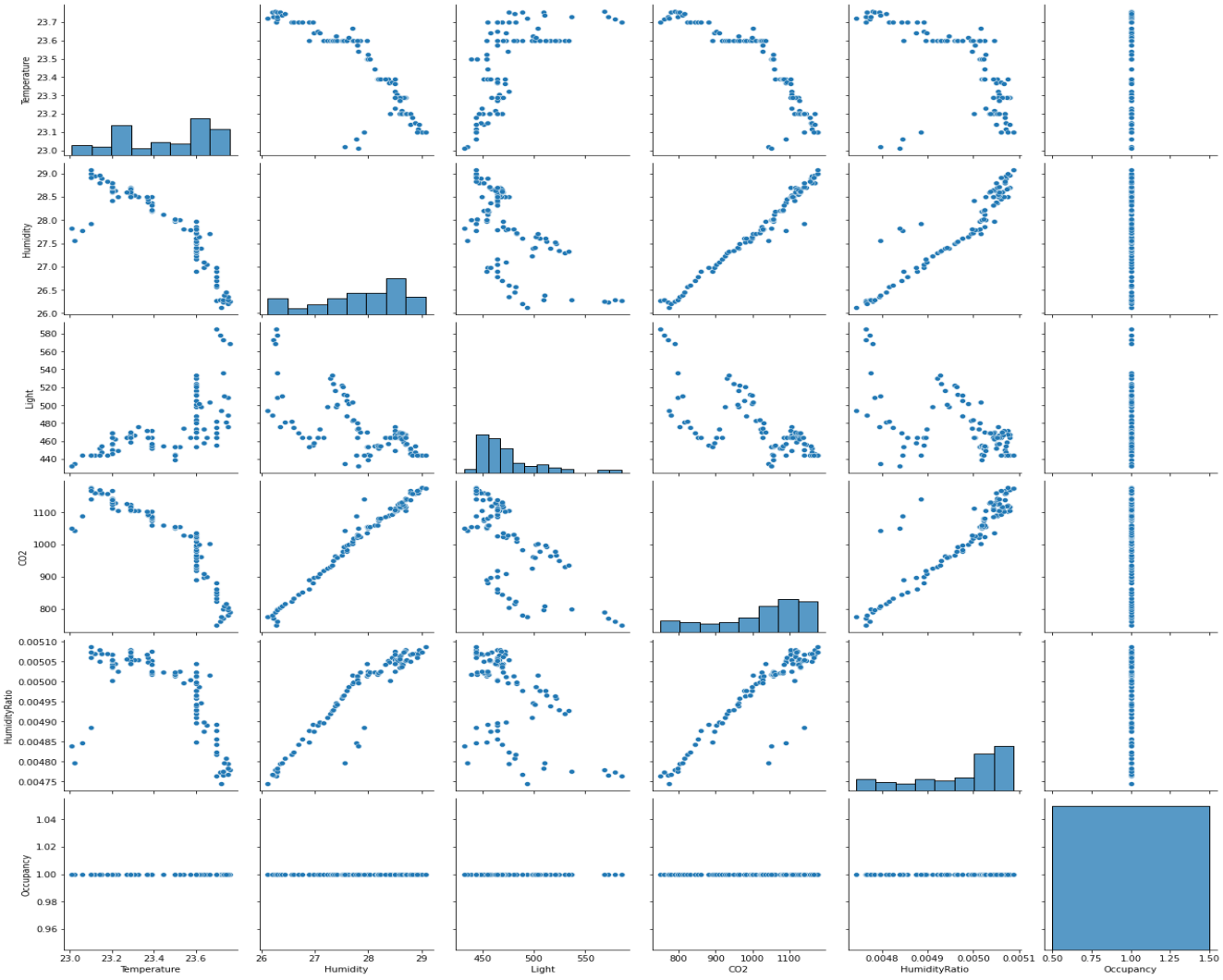
2. MULTIVARIATE ANALYSIS: PCA

```
from sklearn.decomposition import PCA
pca = PCA()
feature = pca.fit_transform(feature.drop("date", axis="columns"))
variance = pca.explained_variance_ratio_
print(variance*100)
PC_Values = np.arange(pca.n_components_) + 1
plt.plot(PC_Values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.axhline(y=0.023, color='r', linestyle='-')
plt.title("Scree Plot", fontsize=15)
plt.xlabel('Principal Component', fontsize=14)
plt.ylabel('Variance Explained', fontsize=14)
```



3. Pairplot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(df.drop(["date",],axis=1).head(100),height=3)
```



Result:

Thus, the Python programs to perform multivariate analysis has been implemented successfully.

Exp no :3 a	**Logistic Regression**

Aim: To explore various commands for performing Logistic regression on the dataset

```
df=pd.read_csv("/content/datatest.txt",encoding="UTF-8") //reading train data
df1=pd.read_csv("/content/datatest2.txt",encoding="UTF-8")//reading test data
y1=df["Occupancy"]
x1=df.drop(["Occupancy","date",],axis=1) // splitting into input and output
y=df["Occupancy"]
x=df.drop(["Occupancy","date",],axis=1) // splitting into input and output
// training the logistic regression model
 from sklearn.linear_model import LogisticRegression
  lr = LogisticRegression()
lr.fit(x1,y1)
predictions = lr.predict(x) //making predictions
score = lr.score(x, y)
print(score) //printing the accuracy score
// heatmap
from sklearn import metrics
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```

OUTPUT

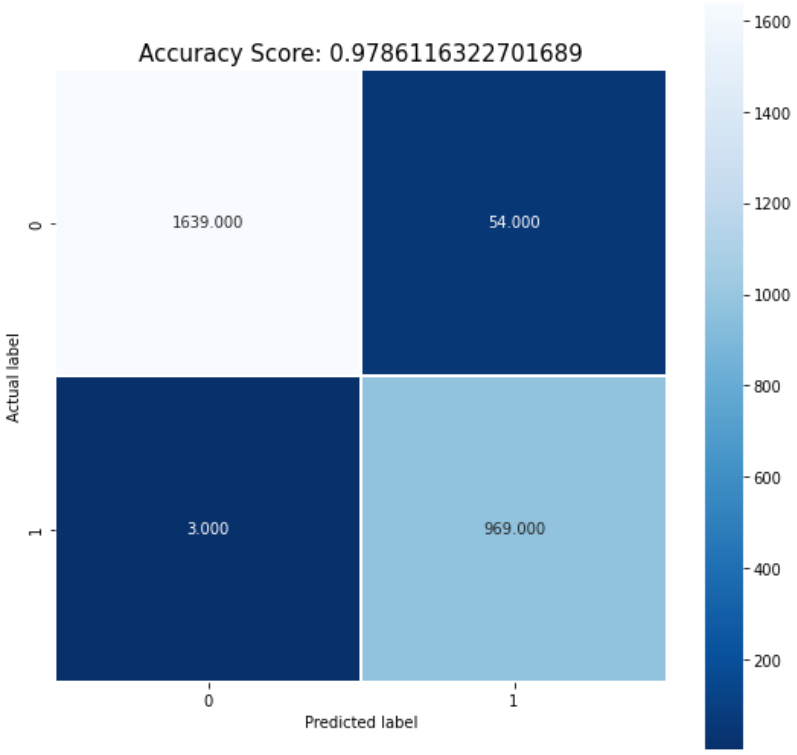
Score =0.9786116322701689

Confusion matrix

[[1639 54]

[3 969]]

Heat map



Result:

Thus, the Python programs to perform Logistic regression has been implemented successfully.

Exp no :3 b

Multiple Regression

Aim: to explore various commands for performing multiple regression on the dataset

```
from sklearn.linear_model import LinearRegression
x1=df.drop(["Occupancy","date","HumidityRatio",],axis=1)
y1=df["HumidityRatio"]
//splitting the data into input and predicted data
x=df.drop(["Occupancy","date","HumidityRatio",],axis=1)
y=df["HumidityRatio"]
//training the model and predicting values
linearReg = LinearRegression()
linearReg.fit(x1,y1)
y_pred=linearReg.predict(x)
//accuracy of the model
score = linearReg.score(x, y)
print(score)
```

OUTPUT

```
prediction: array([0.00473046, 0.00473909, 0.00473404, ..., 0.00483148, 0.00483807, 0.00484892])
Intercept: -0.005338032822997117
Coefficients: [ 2.52716004e-04  1.53192296e-04  -3.30661947e-08  9.85101138e-08]
Score: 0.9987793863852992
```

Result:

Thus, the Python programs to perform Multiple regression has been implemented successfully.

Exp no :4 a	**Classification using Naïve Bayes Classifier**

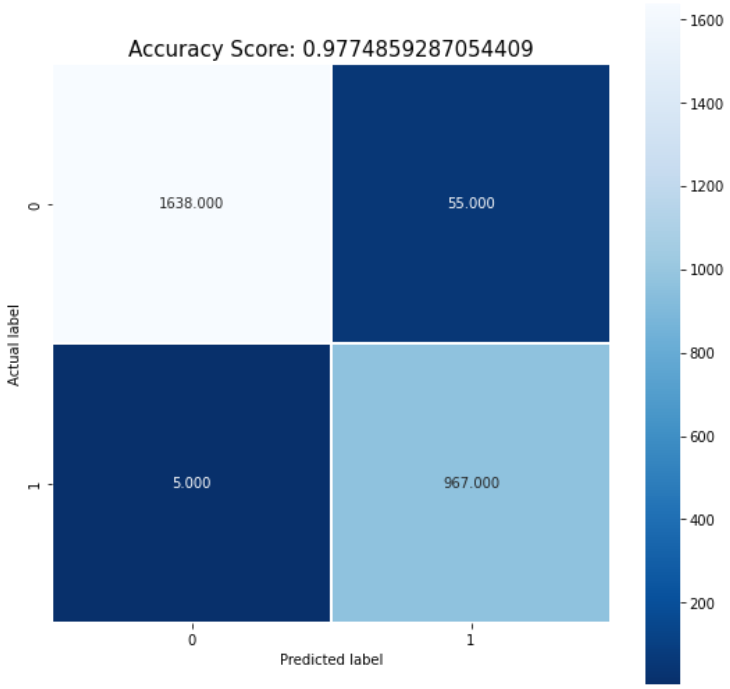
Aim: to explore commands for performing classification using naïve bayes classifier on the Dataset

```
// reading the test and train data
dfTrain = pd.read_csv('/datatraining.txt')
dfTest = pd.read_csv('/datatest.txt')
y_train = dfTrain["Occupancy"]
x_train = dfTrain.drop(["Occupancy",  "date", ], axis = 1)
y_test = dfTest["Occupancy"]
x_test = dfTest.drop(["Occupancy", "date", ], axis = 1)
// training and predicting
from sklearn.preprocessing import StandardScaler
sc = StandardScaler() //standard scaling
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
y_pred
//heatmap
score = classifier.score(x_test, y_test)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```

OUTPUT

```
//predicted value
array([1, 1, 1, ..., 1, 1, 1])
//confusion matrix
[[1638   55]
 [   5  967]]
//accuracy  score
0.9774859287054409
```

Heatmap



Result:

Thus, the Python programs to perform classification using Naïve Bayes classifier has been implemented successfully.

Exp no :4 b	**Classification using support vector machine**

Aim : to explore commands for performing classification using support vector machine on the dataset

```
// reading the test and train data
dfTrain = pd.read_csv('/datatraining.txt')
dfTest = pd.read_csv('/datatest.txt')
y_train = dfTrain["Occupancy"]
x_train = dfTrain.drop(["Occupancy", "date", ], axis = 1)
y_test = dfTest["Occupancy"]
x_test = dfTest.drop(["Occupancy", "date", ], axis = 1)
// training and predicting
from sklearn.svm import SVC
clf = SVC(kernel='rbf')
clf.fit(x_train, y_train)
y_pred=clf.predict(x_test)
y_pred
from sklearn import metrics
print(metrics.classification_report(y_test.astype('float32'), y_pred))
//heatmap
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```

OUTPUT

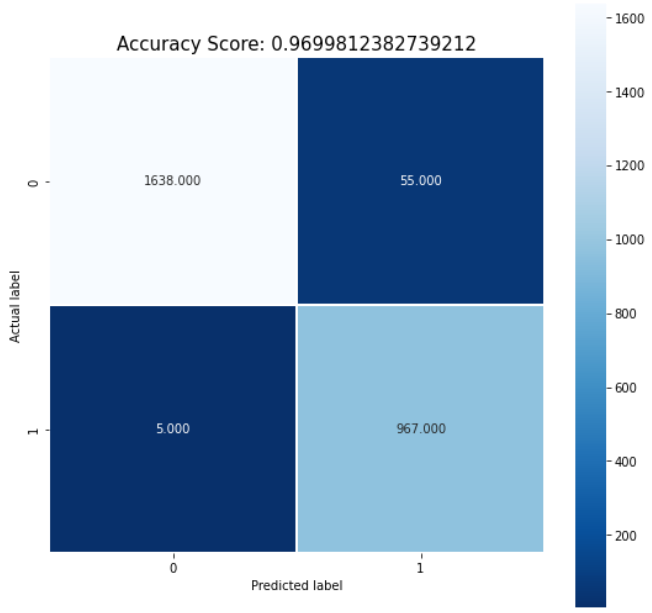
//predicted value

array([1, 1, 1, ..., 1, 1, 1])

//performance metrics

	precision	recall	f1-score	support
0.0	1.00	0.95	0.98	1693
1.0	0.93	1.00	0.96	972
accuracy			0.97	2665
macro avg	0.96	0.98	0.97	2665
weighted avg	0.97	0.97	0.97	2665

Heatmap



Result:

Thus, the Python programs to perform classification using Support Vector Machine has been implemented successfully.

Exp no :5 a	**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 underclo ud 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
- Deployment Models
- OpenStack-based public cloud
- On-premises distribution
- Hosted OpenStack Private Cloud
- Appliance based OpenStack
- 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 :5 b	OpenStack Installation

Aim:

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

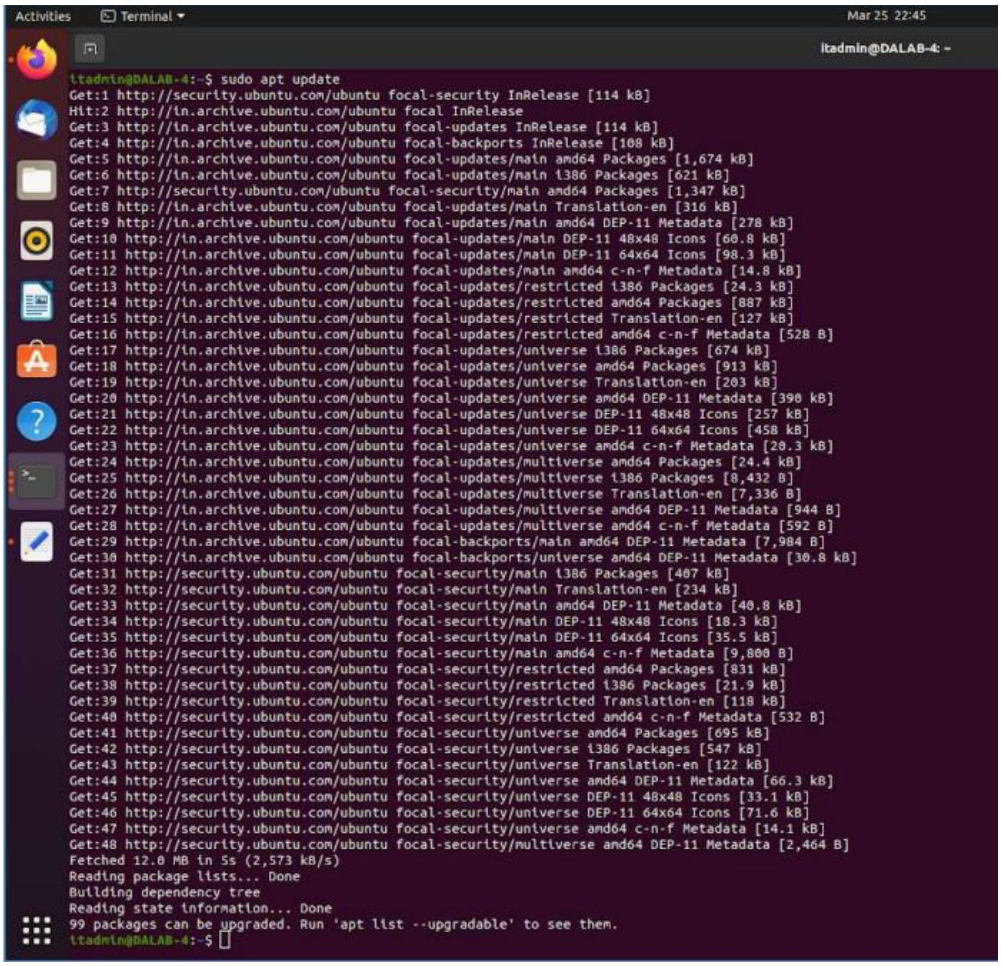
Procedure:

Installation of OpenStack in Ubuntu

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

```
sudo apt – update
```

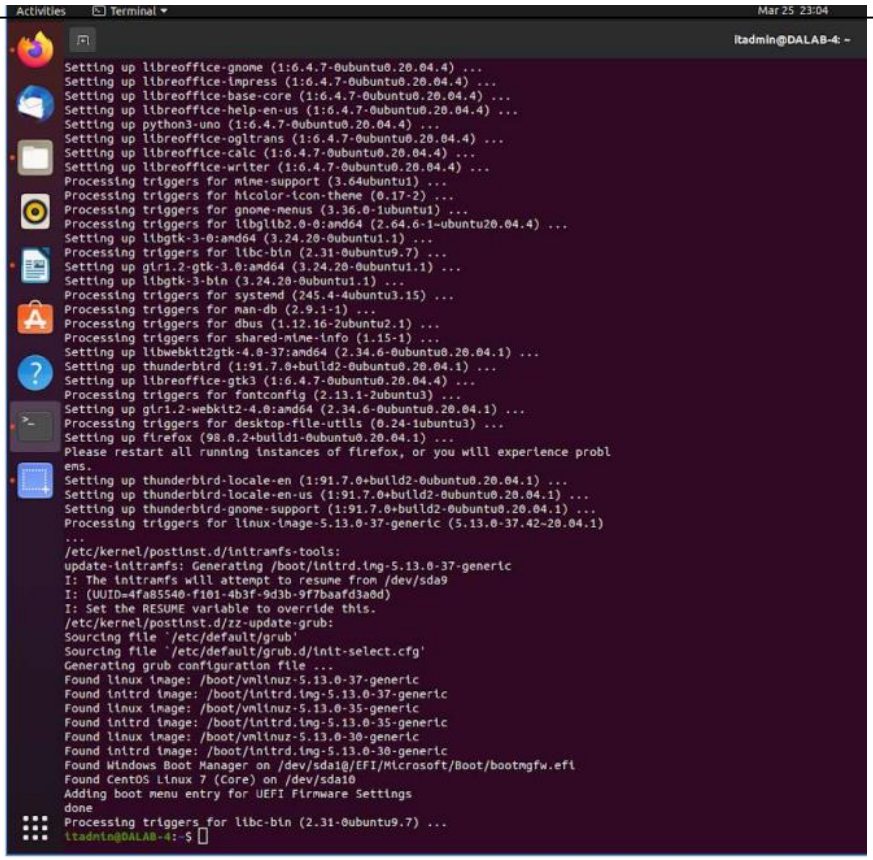
This downloads package information from all configured sources.



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

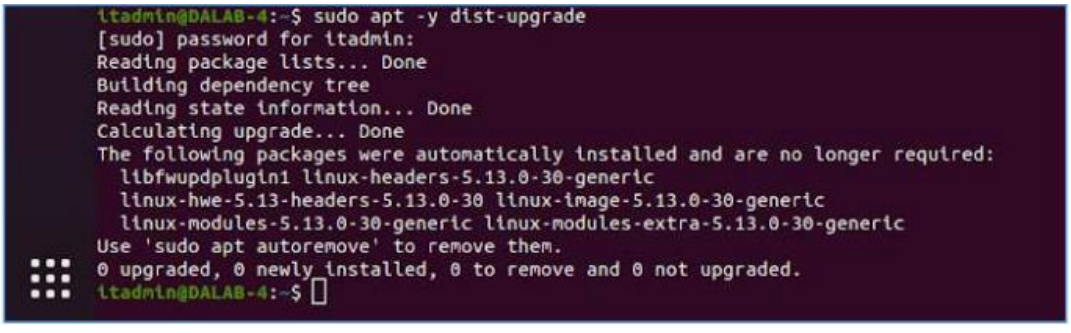
5. Upgrade existing packages, installs new dependencies that are not in the system, and deletes those that are not needed:

```
sudo apt –y upgrade
```



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

```
sudo apt –y dist-upgrad
```



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

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

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

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

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

```
sudo su – stac
```

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

9. Install git:

```
sudo apt -y 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. Download devstack from its repository into system:

```
cd 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:

```
nano local.conf

vi local.conf

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

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

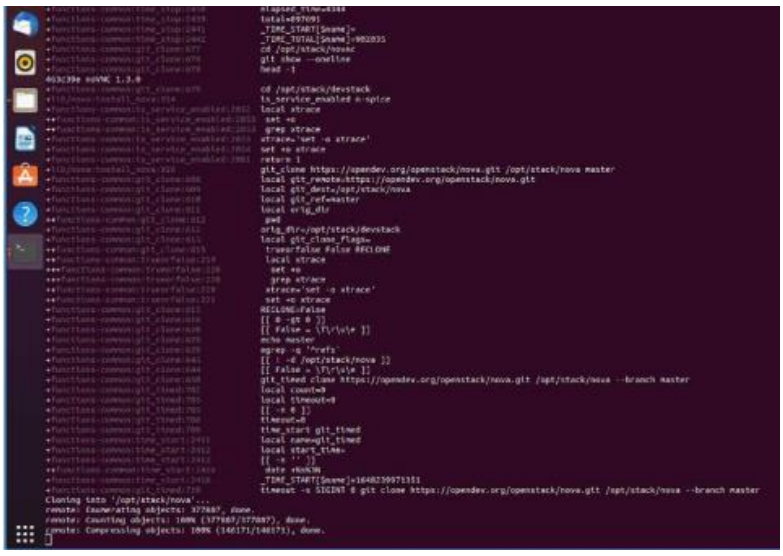
12. Add following inside local.conf:

```
[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
```



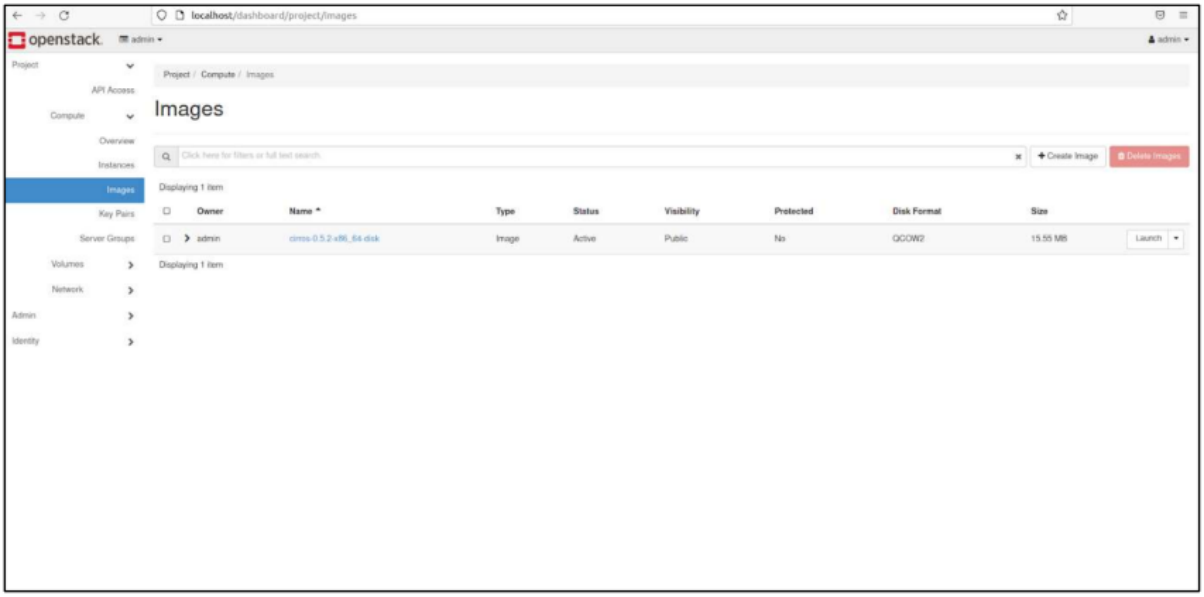
After installation, terminal:

```
=========================
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 :5 c	Creation of VMs in OpenStack

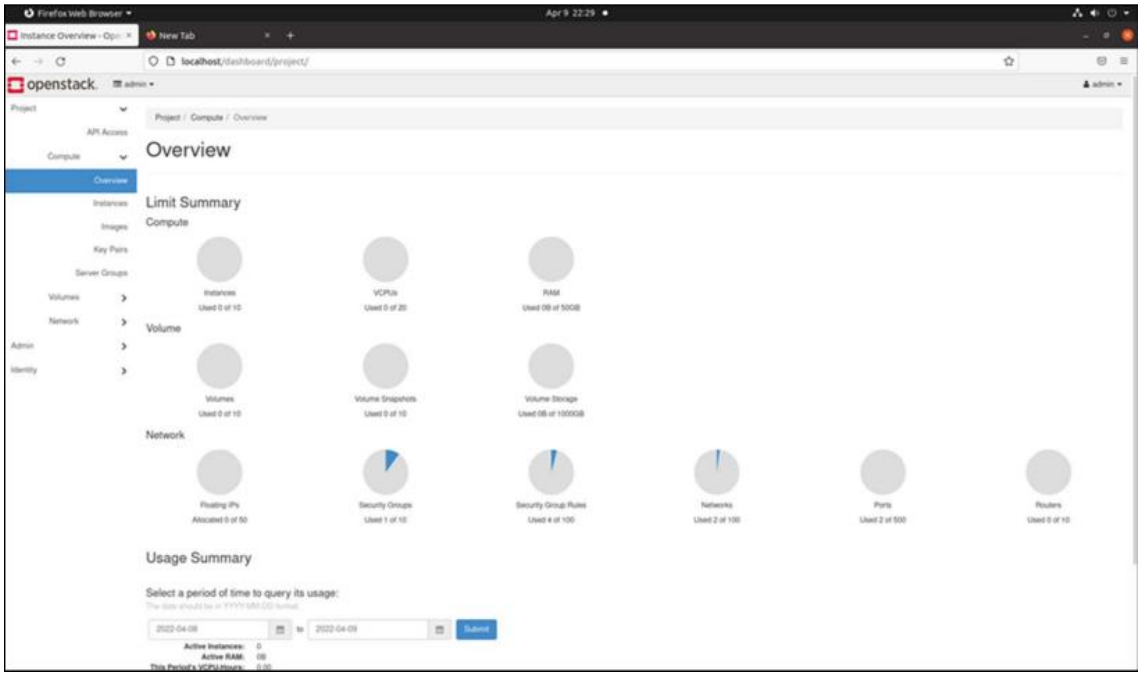
Aim:

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

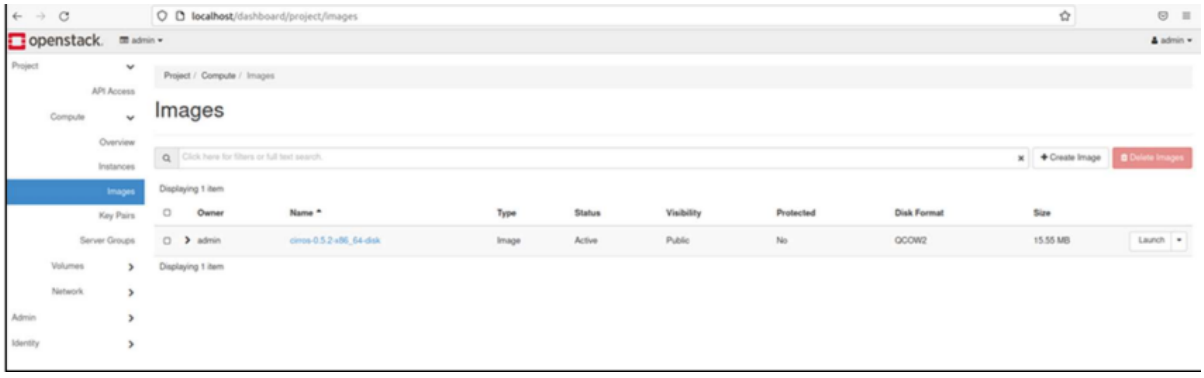
Procedure:

Deploying VM in OpenStack

1. Login to OpenStack using credential

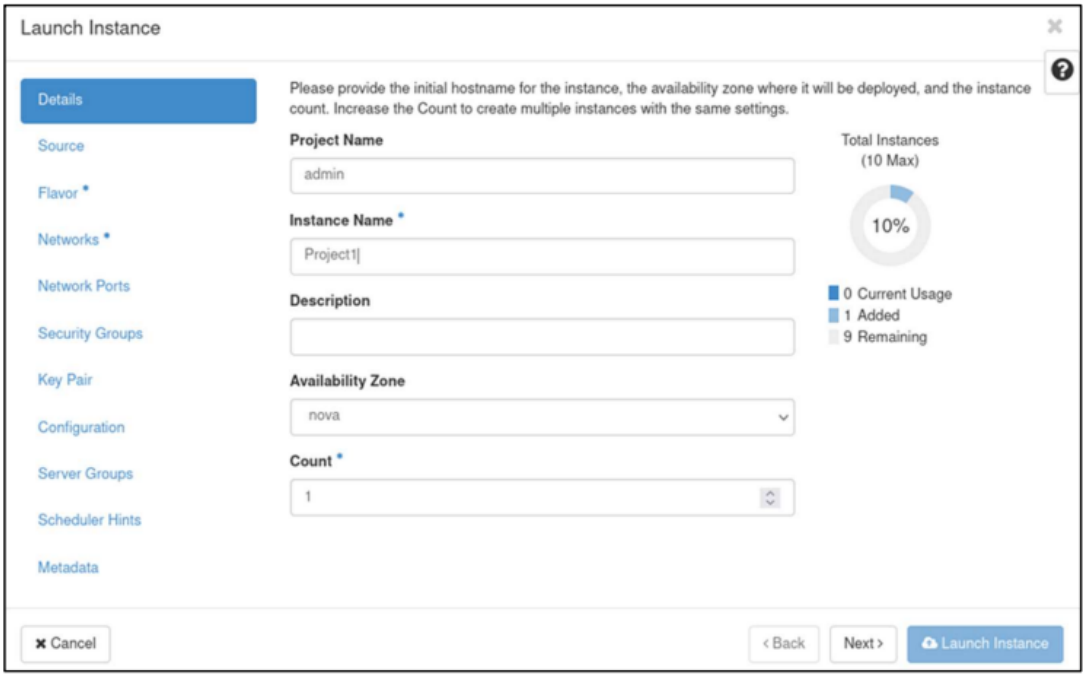


2. Launch an Existing Image Instance in OpenStack

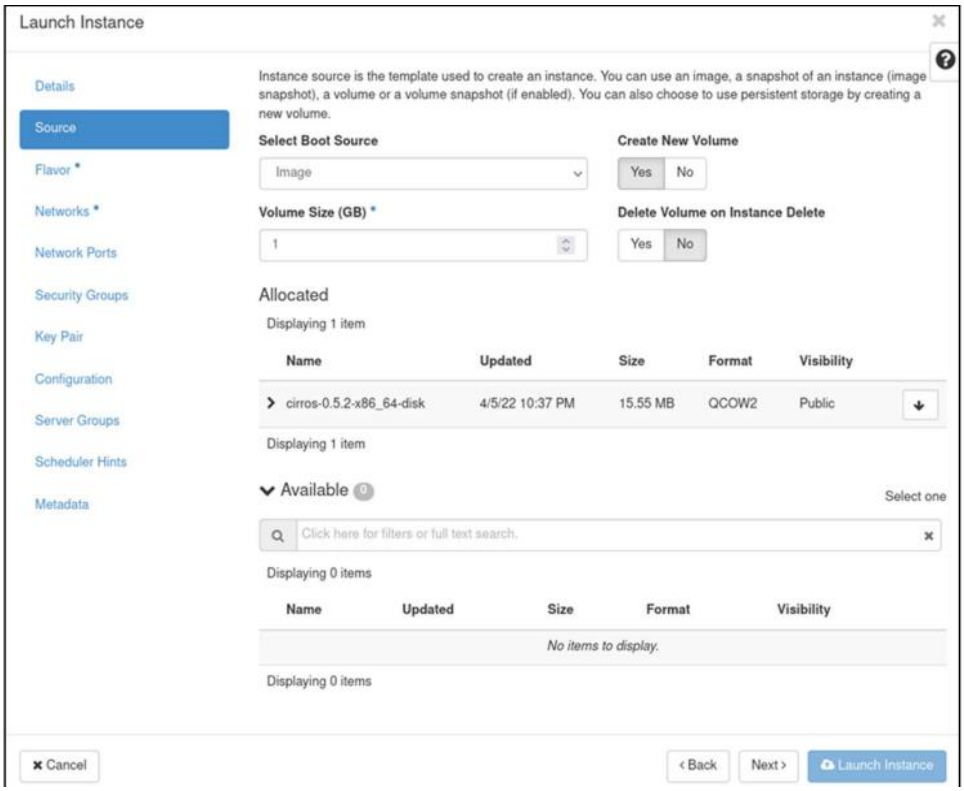


Once launch option is selected, pop-up window is displayed.

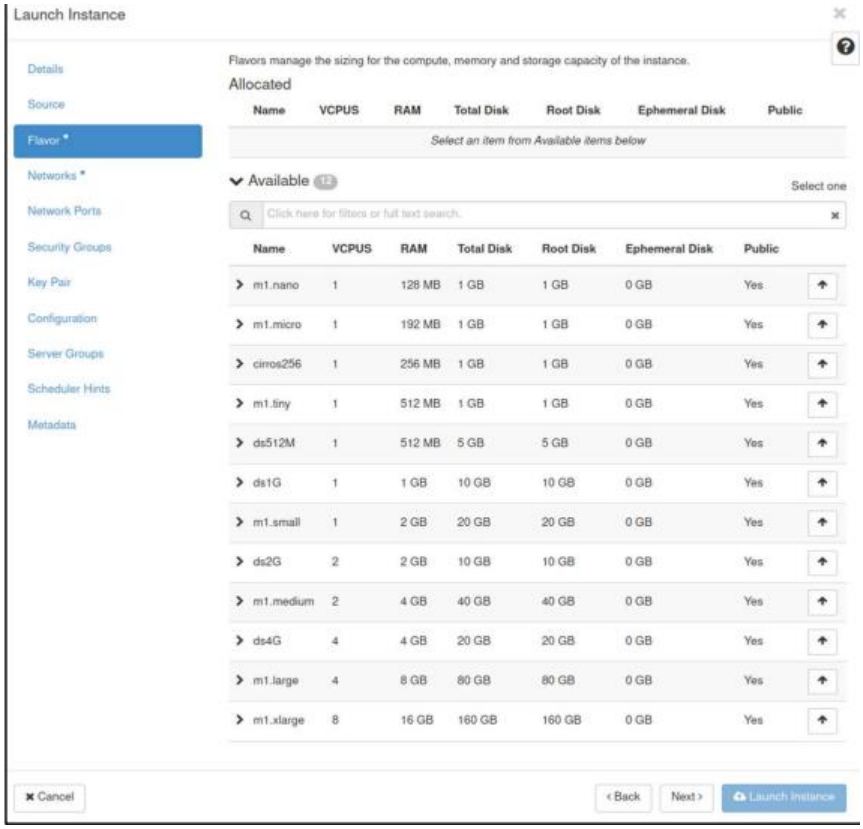
3. Give instance name of your choice. Leave the other fields as default values and click next



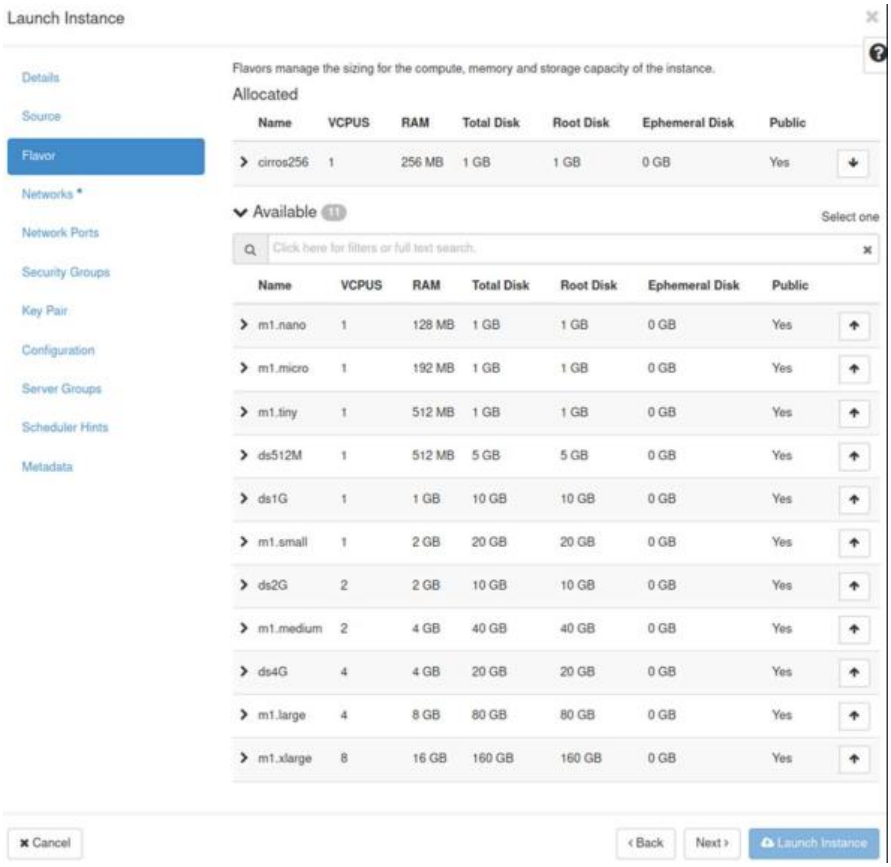
4. Source tab will be next screen in the launch instance window where storage is created and click next



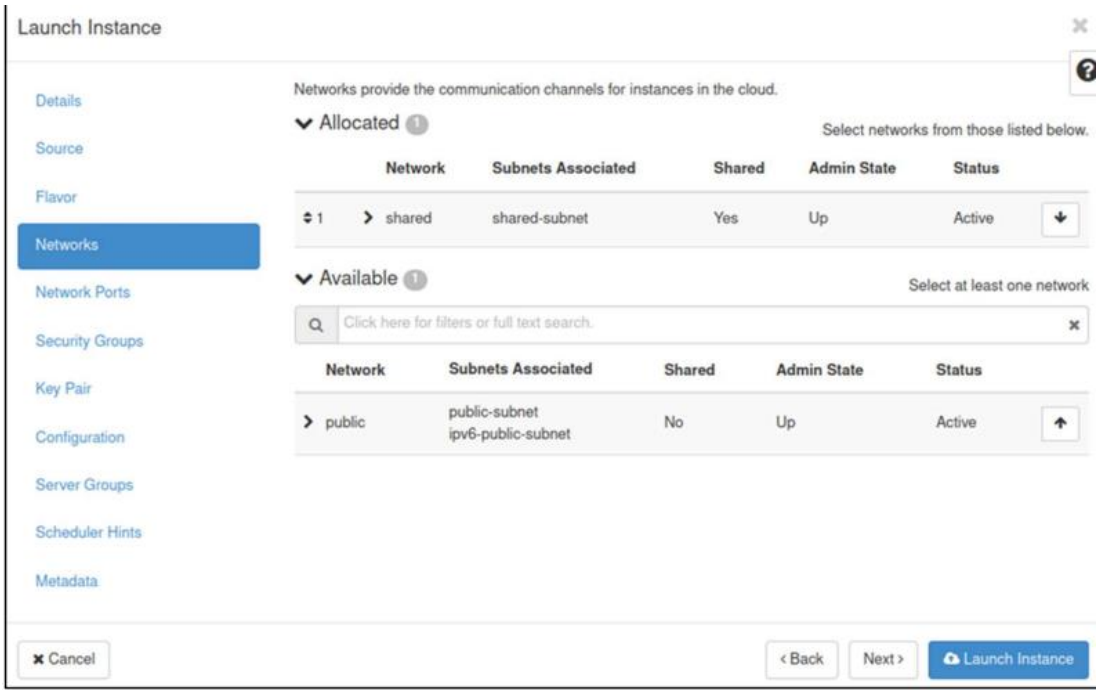
5. Allocate VM resources by adding a flavour best suitable for your needs and click on Next



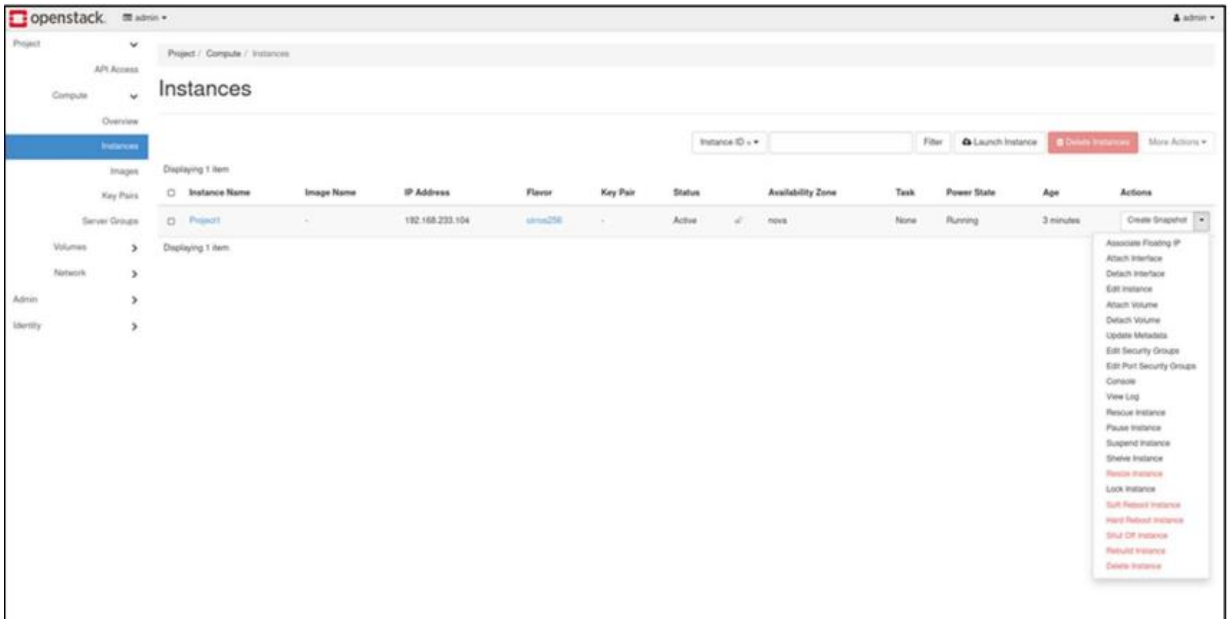
6. From available instance, add instance which already launched so that the instance moves under allocated



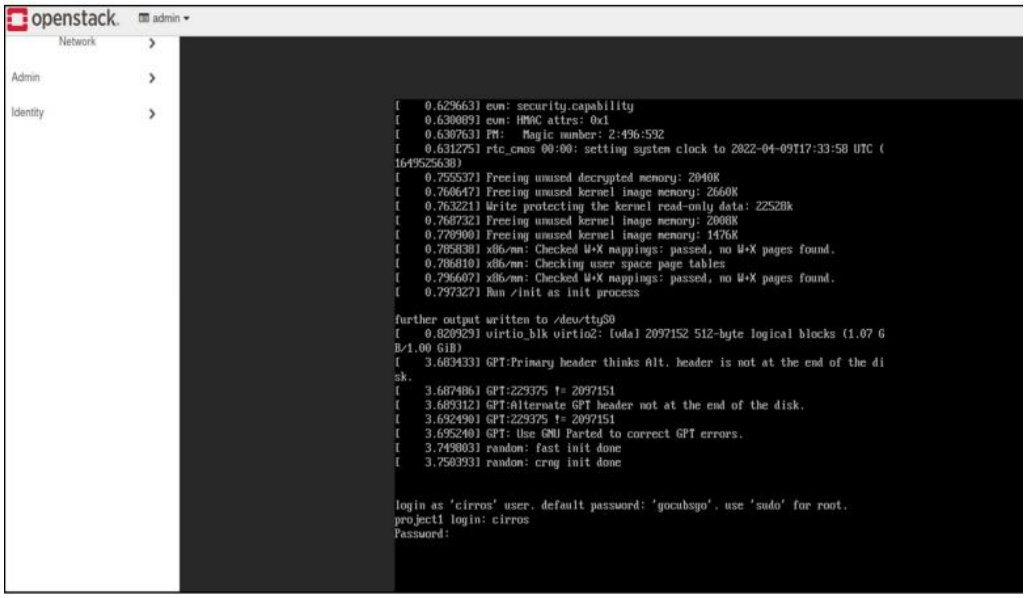
7. Finally, add shared network from available networks in OpenStack to your instance using upward arrow-mark button and hit on ‘Launch Instance’ to start the virtual machine



8. Once launch is clicked, select instance in right tab. The added instance is seen under it



9. Click ‘console’



10. Create a text file:

```
vi test.txt
```

11. In console type:

```
ls
```

12. Display text content using:

```
Cat test.txt
```

Result:

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

Exp no :6 a	**Hadoop Installation**

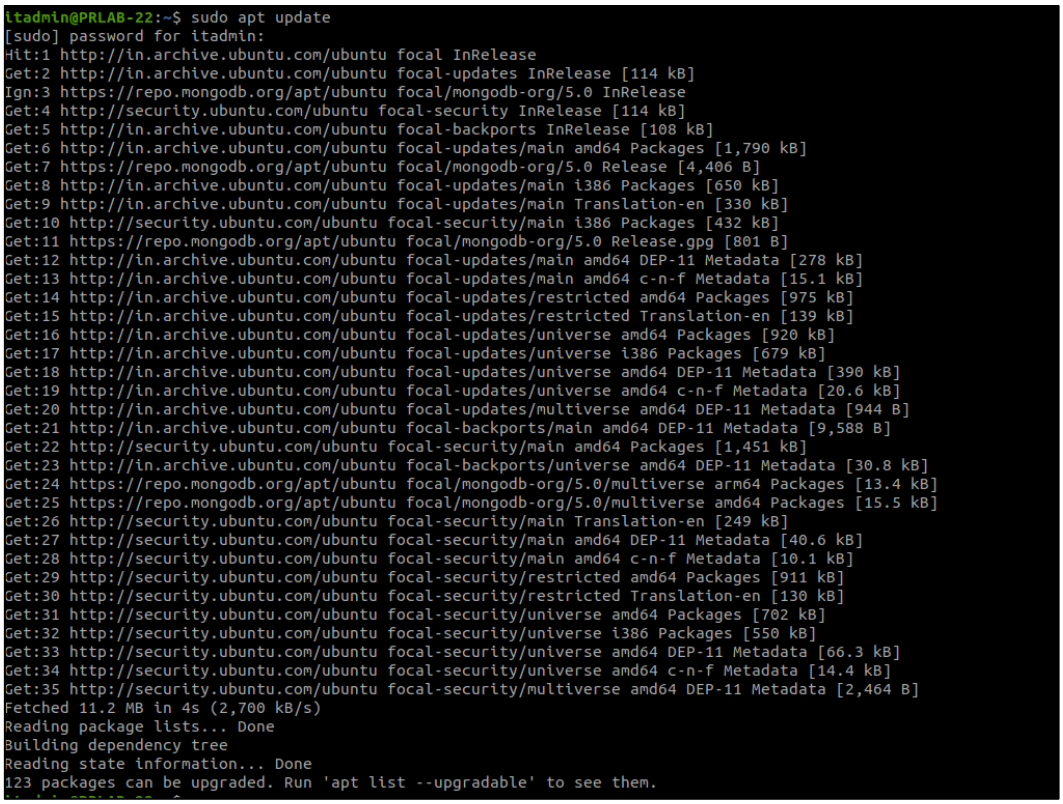
Aim:

To study about the architecture of Hadoop and its components.

Procedure:

1. Update system before installation:

```
sudo apt – update
```



2. Install OpenJDK

```
sudo apt install openjdk-11-jdk
```

```
itadmin@PRLAB-22:~$ sudo apt install openjdk-8-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-8-demo openjdk-8-source visualvm icedtea-8-plugin fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-8-jdk openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 21 newly installed, 0 to remove and 123 not upgraded.
Need to get 43.5 MB of archives.
After this operation, 162 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 java-common all 0.72 [6,816 B]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 openjdk-8-jre-headless amd64 8u312-b07-0ubuntu1~20.04 [28.2 MB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal/main amd64 ca-certificates-java all 20190405ubuntu1 [12.2 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal/main amd64 fonts-dejavu-extra all 2.37-1 [1,953 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libatk-wrapper-java all 0.37.1-1 [53.0 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libatk-wrapper-java-jni amd64 0.37.1-1 [45.1 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal/main amd64 xorg-sgml-doctools all 1:1.11-1 [12.9 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu focal/main amd64 x11proto-dev all 2019.2-1ubuntu1 [594 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu focal/main amd64 x11proto-core-dev all 2019.2-1ubuntu1 [2,620 B]
Get:10 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libice-dev amd64 2:1.0.10-0ubuntu1 [47.8 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libpthread-stubs0-dev amd64 0.4-1 [5,384 B]
Get:12 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libsm-dev amd64 2:1.2.3-1 [17.0 kB]
```

3. Verify intalled Java version

```
java -version; javac -version
```

```
itadmin@PRLAB-22:~$ java -version; javac -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1~20.04-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
javac 1.8.0_312
```

4. Install OpenSSH Server

```
sudo apt install openssh-server openssh-client -y
```

```
itadmin@PRLAB-22:~$ sudo apt install openssh-server openssh-client -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  openssh-sftp-server
Suggested packages:
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard
The following packages will be upgraded:
  openssh-client openssh-server openssh-sftp-server
3 upgraded, 0 newly installed, 0 to remove and 120 not upgraded.
Need to get 1,099 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-sftp-server amd64 1:8.2p1-4ubuntu0.5 [51.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-server amd64 1:8.2p1-4ubuntu0.5 [377 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-client amd64 1:8.2p1-4ubuntu0.5 [671 kB]
Fetched 1,099 kB in 0s (2,271 kB/s)
Preconfiguring packages ...
(Reading database ... 213340 files and directories currently installed.)
Preparing to unpack .../openssh-sftp-server_1%3a8.2p1-4ubuntu0.5_amd64.deb ...
Unpacking openssh-sftp-server (1:8.2p1-4ubuntu0.5) over (1:8.2p1-4ubuntu0.4) ...
Preparing to unpack .../openssh-server_1%3a8.2p1-4ubuntu0.5_amd64.deb ...
Unpacking openssh-server (1:8.2p1-4ubuntu0.5) over (1:8.2p1-4ubuntu0.4) ...
Preparing to unpack .../openssh-client_1%3a8.2p1-4ubuntu0.5_amd64.deb ...
Unpacking openssh-client (1:8.2p1-4ubuntu0.5) over (1:8.2p1-4ubuntu0.4) ...
Setting up openssh-client (1:8.2p1-4ubuntu0.5) ...
Setting up openssh-sftp-server (1:8.2p1-4ubuntu0.5) ...
Setting up openssh-server (1:8.2p1-4ubuntu0.5) ...
rescue-ssh.target is a disabled or a static unit, not starting it.
Processing triggers for systemd (245.4-4ubuntu3.15) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ufw (0.36-6ubuntu1) ...
```

5. Create a Hadoop user

```
sudo adduser hdoop
```

```
itadmin@PRLAB-22:~$ sudo adduser hdoop
Adding user `hdoop' ...
Adding new group `hdoop' (1002) ...
Adding new user `hdoop' (1002) with group `hdoop' ...
Creating home directory `/home/hdoop' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for hdoop
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

6. Switch to Hadoop user

```
su - hdoop
```

```
itadmin@PRLAB-22:~$ su - hdoop
Password:
hdoop@PRLAB-22:~$
```

7. Generate an SSH key pair and define the location is is to be stored in

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

```
hdoop@PRLAB-22:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/hdoop/.ssh'.
Your identification has been saved in /home/hdoop/.ssh/id_rsa
Your public key has been saved in /home/hdoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ML0nTmQKjrrLU8f7Y+ogsbBvRvyK8hur4xvz6CTYUA4 hdoop@PRLAB-22
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|       .         |
|E . . o +        |
| + o . * .       |
|ooo o . S .      |
|o==. o o o       |
|=B+o. . .        |
|*=Ooo. o         |
|O&O+.o+..        |
+----[SHA256]-----+
```

8. Use the cat command to store the public key as authorized_keys in the ssh directory

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

9. Set the permissions for your user with the chmod command

```
chmod 0600 ~/.ssh/authorized_keys
```

```
hdoop@PRLAB-22:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hdoop@PRLAB-22:~$ chmod 0600 ~/.ssh/authorized_keys
```

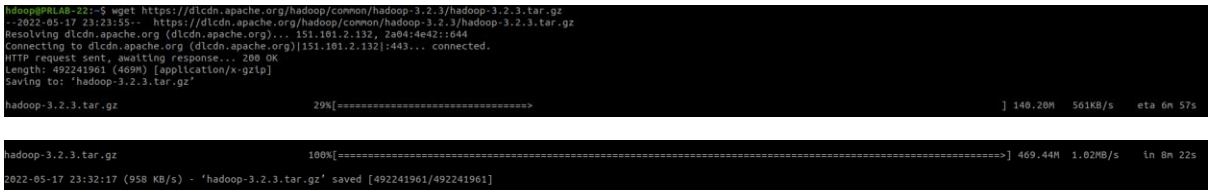
10. Verify everything is set up correctly by using the hdoop user to SSH to localhost

```
ssh localhost
```

```
hdoop@PRLAB-22:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:SaTAp+60cPT8NCkg14kfRqJ8pzcBqlN5mV2T5rgLflE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

118 updates can be applied immediately.
67 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

11. Visit the official Apache Hadoop project page, and select the version of Hadoop you want to implement. Click the link to download locally in a machine or install the Hadoop framework in the system using wget command

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
```



Extract the files to initiate the Hadoop installation.

```
tar xvzf hadoop-3.2.3.tar.gz
```

```
hadoop-3.2.3/share/doc/hadoop/hadoop-distcp/css/maven-base.css
hadoop-3.2.3/share/doc/hadoop/hadoop-distcp/css/maven-theme.css
hadoop-3.2.3/share/doc/hadoop/hadoop-distcp/css/site.css
hadoop-3.2.3/share/doc/hadoop/hadoop-distcp/css/print.css
hadoop-3.2.3/share/doc/hadoop/hadoop-distcp/dependency-analysis.html
hadoop-3.2.3/lib/
hadoop-3.2.3/lib/native/
hadoop-3.2.3/lib/native/libhdfspp.so.0.1.0
hadoop-3.2.3/lib/native/libhadooppipes.a
hadoop-3.2.3/lib/native/libhdfs.so.0.0.0
hadoop-3.2.3/lib/native/libhadooputils.a
hadoop-3.2.3/lib/native/libhadoop.so
hadoop-3.2.3/lib/native/libhdfspp.so
hadoop-3.2.3/lib/native/libhadoop.so.1.0.0
hadoop-3.2.3/lib/native/libnativetask.a
hadoop-3.2.3/lib/native/examples/
hadoop-3.2.3/lib/native/examples/wordcount-part
hadoop-3.2.3/lib/native/examples/wordcount-nopipe
hadoop-3.2.3/lib/native/examples/pipes-sort
hadoop-3.2.3/lib/native/examples/wordcount-simple
hadoop-3.2.3/lib/native/libhdfs.a
hadoop-3.2.3/lib/native/libhdfs.so
hadoop-3.2.3/lib/native/libhadoop.a
hadoop-3.2.3/lib/native/libnativetask.so.1.0.0
hadoop-3.2.3/lib/native/libnativetask.so
hadoop-3.2.3/lib/native/libhdfspp.a
hadoop-3.2.3/LICENSE.txt
```

12. Rename the extracted directory by executing the command

```
mv hadoop-3.2.3 hadoop
```

```
hdoop@PRLAB-22:~$ mv hadoop-3.2.3 hadoop
```

13. Configure Java environment variables for setting up Hadoop

```
dirname $(dirname $(readlink -f $(which java)))
```

```
hdoop@PRLAB-22:~$ dirname $(dirname $(readlink -f $(which java)))
/usr/lib/jvm/java-8-openjdk-amd64/jre
```

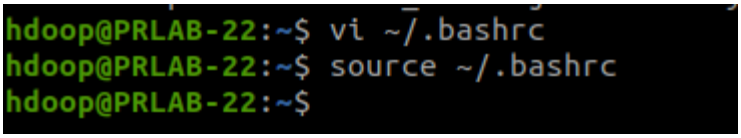
14. Open the "~/.bashrc" file in "vi" text editor. Define the Hadoop environment variables.

```
vi ~/.bashrc
```

```
#Hadoop Related Options
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hdoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

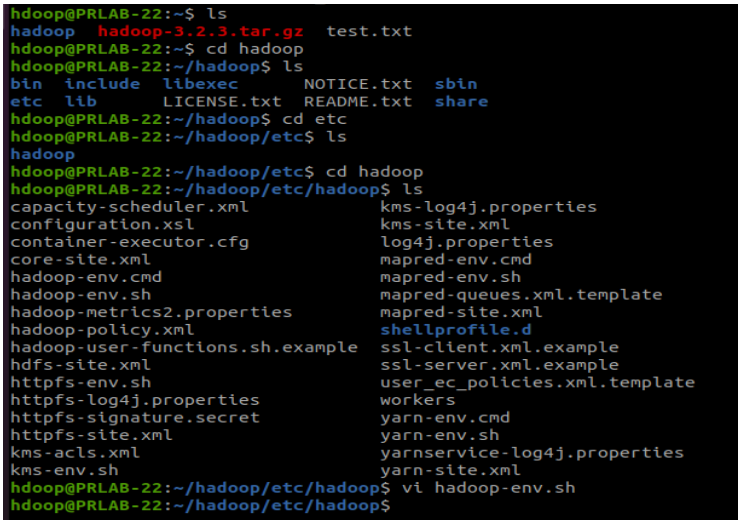
15. Apply the changes to the current running environment

```
source ~/.bashrc
```



16. Add the full path to the OpenJDK installation on your system

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



17. Configure Apache Hadoop on the Ubuntu system. For this, the step is to create two directories: datanode and namenode, inside the home directory of Hadoop.

```
mkdir -p ~/dfsdata/namenode
mkdir -p ~/dfsdata/datanode
```

18. Specify the URL for your NameNode, and the temporary directory Hadoop uses for the map and reduce process. Go to specific directory and open core-site.xml

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hdoop/tmpdata</value>
</property>
<property>

  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
```

19. Configure the file by defining the NameNode and DataNode storage directories in hdfs-site.xml

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi hdfs-site.xml
```

Add the following configuration to the file and, if needed, adjust the NameNode and DataNode directories to your custom locations.

```
<configuration>

<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property></configuration>
```

20. Use the following command to access the mapred-site.xml file and define MapReduce values:

```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi mapred-site.xml
```

Add the following configuration to change the default MapReduce framework name value to yarn

```
<configuration>

<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property> </configuration>
```

21. Edit yarn-site.xml File

Open the yarn-site.xml file in a text editor: nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

Append the following configuration to the file

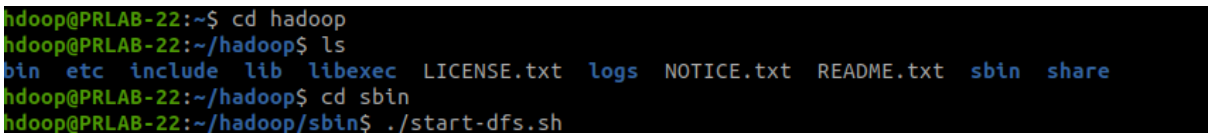
```
hdoop@PRLAB-22:~/hadoop/etc/hadoop$ vi yarn-site.xml
```

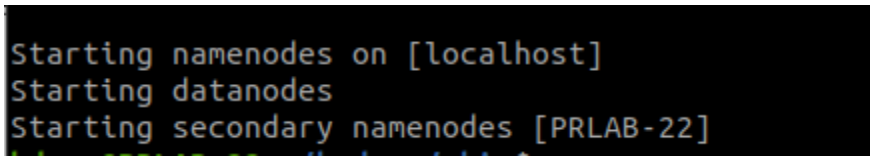
22. Format HDFS NameNode

```
hdfs namenode -format
```

23. Start Hadoop Cluster

```
./start-dfs.sh
```





24. Once the namenode, datanodes, and secondary namenode are up and running, start the YARN resource

```
./start-yarn.sh
```

```
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

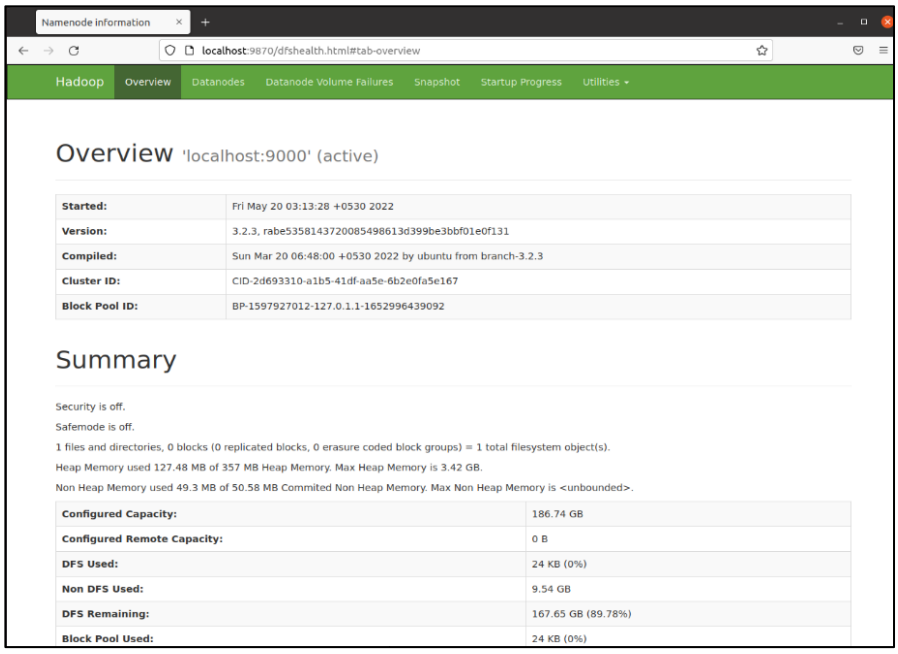
25. Check if all the daemons are active and running as Java processes

```
jps
```

```
hdoop@PRLAB-22:~/hadoop/sbin$ jps
36208 NodeManager
36627 Jps
35547 DataNode
36059 ResourceManager
35372 NameNode
35789 SecondaryNameNode
```

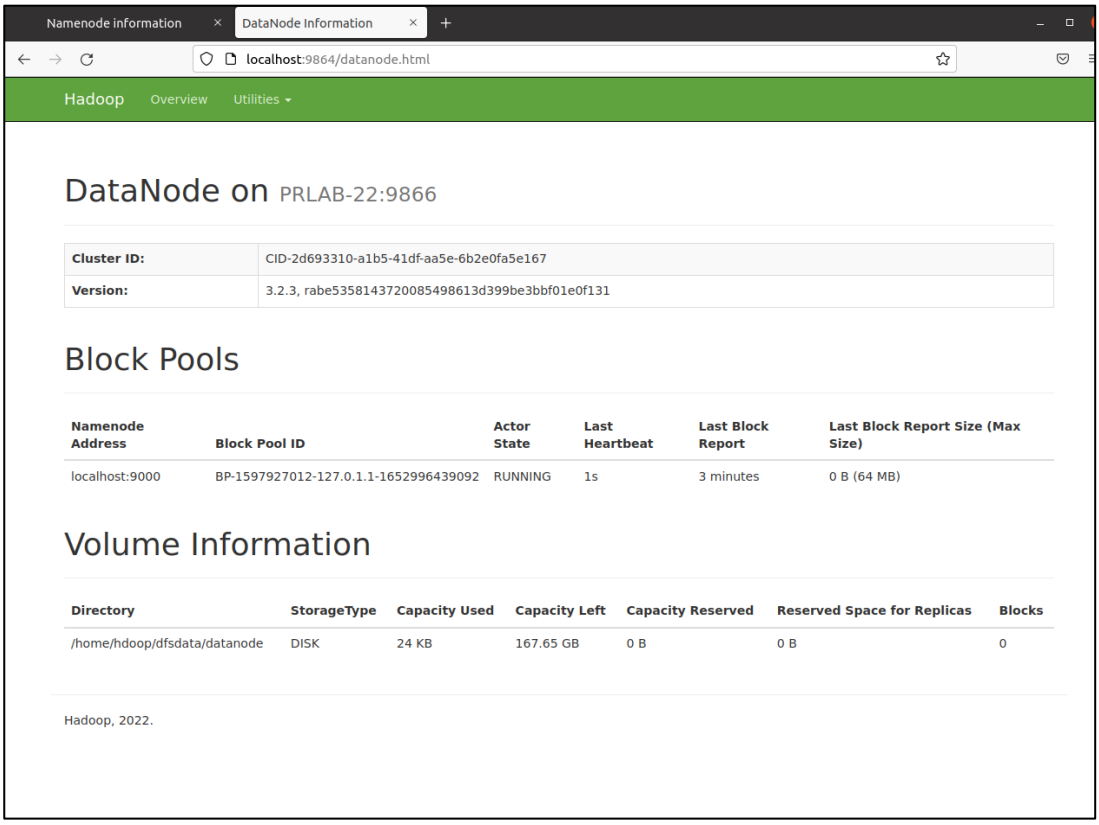
26. Access Hadoop UI from Browser

```
http://localhost:9870
```



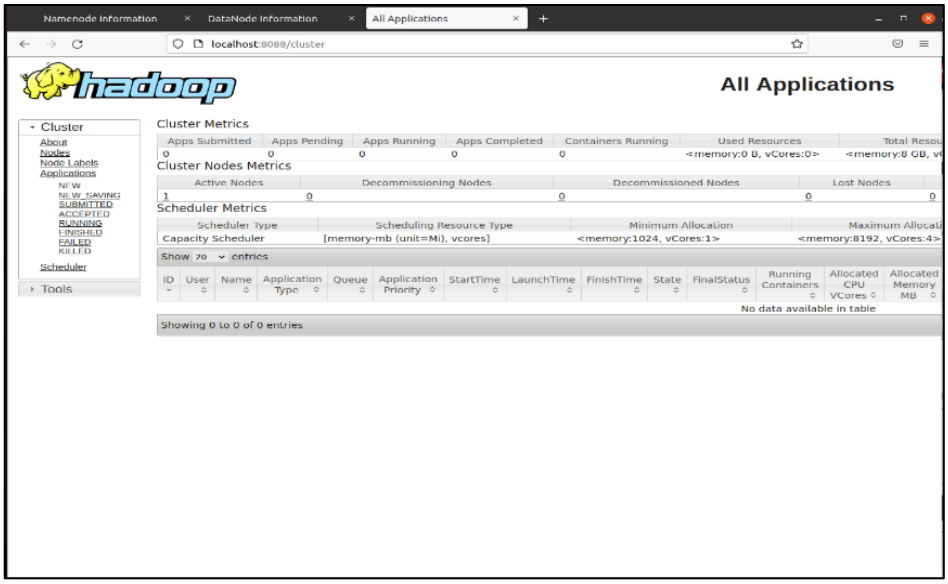
27. The default port 9864 is used to access individual DataNodes directly from browser

```
http://localhost:9864
```



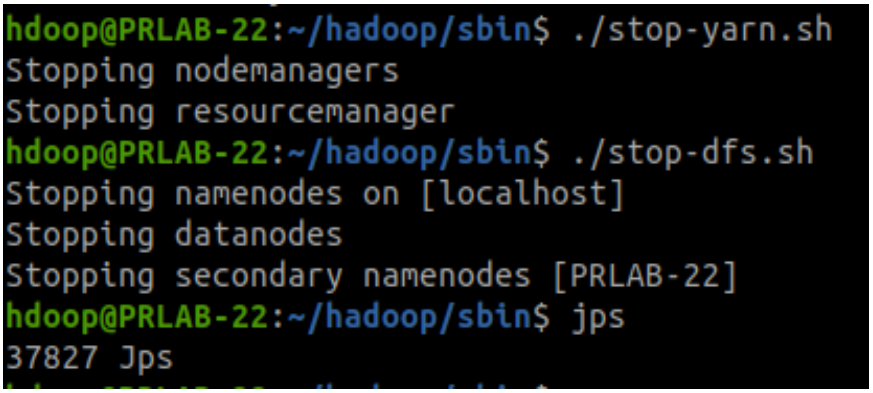
28. The YARN Resource Manager is accessible on port 8088. The Resource Manager is an invaluable tool that allows to monitor all running processes in Hadoop cluster

```
http://localhost:8088
```



29. Stop the NameNode, DataNode and YARN

```
./~stop-yarn.sh
```



Result: Thus, Hadoop has been successfully installed.

Exp no : 6 b	File Management in Hadoop

Aim: To implement file management in the Hadoop File System of Hadoop.

Codes and Output:

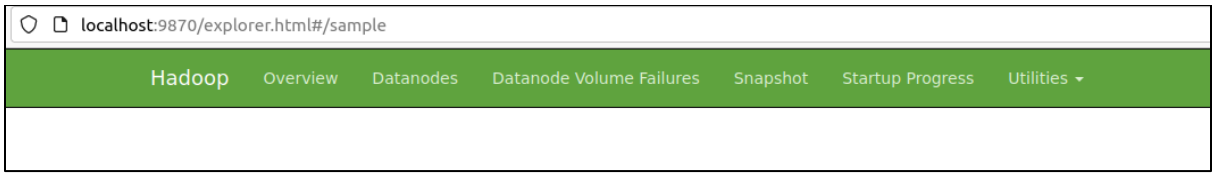
a) Adding Directories

```
hadoop fs -mkdir /sample
```

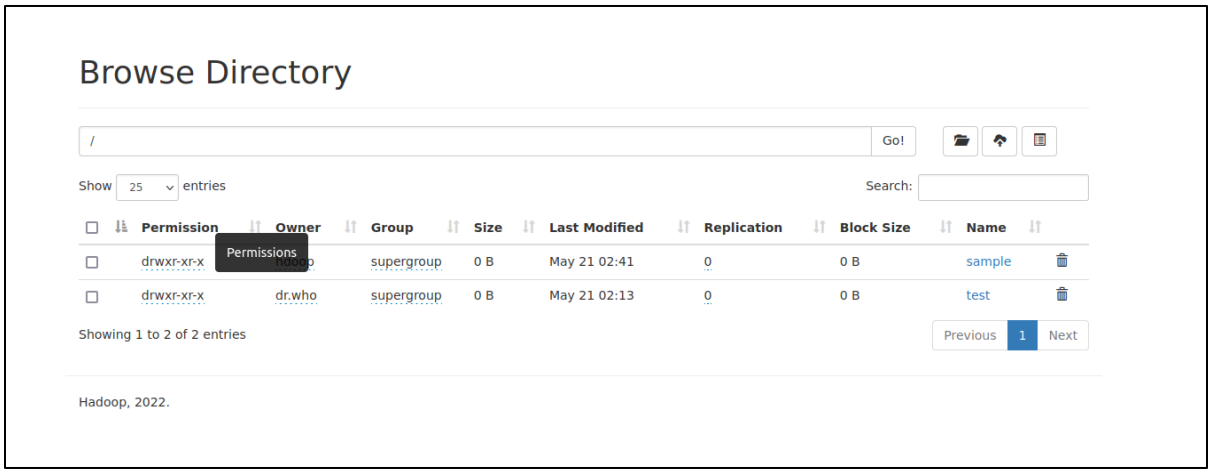
```
hdoop@PRLAB-22:~$ hadoop fs -mkdir /sample
```

Login to the web browser

```
http://localhost:9870
```



Under Utilities → click (Browse the file system option)



b) Adding Files

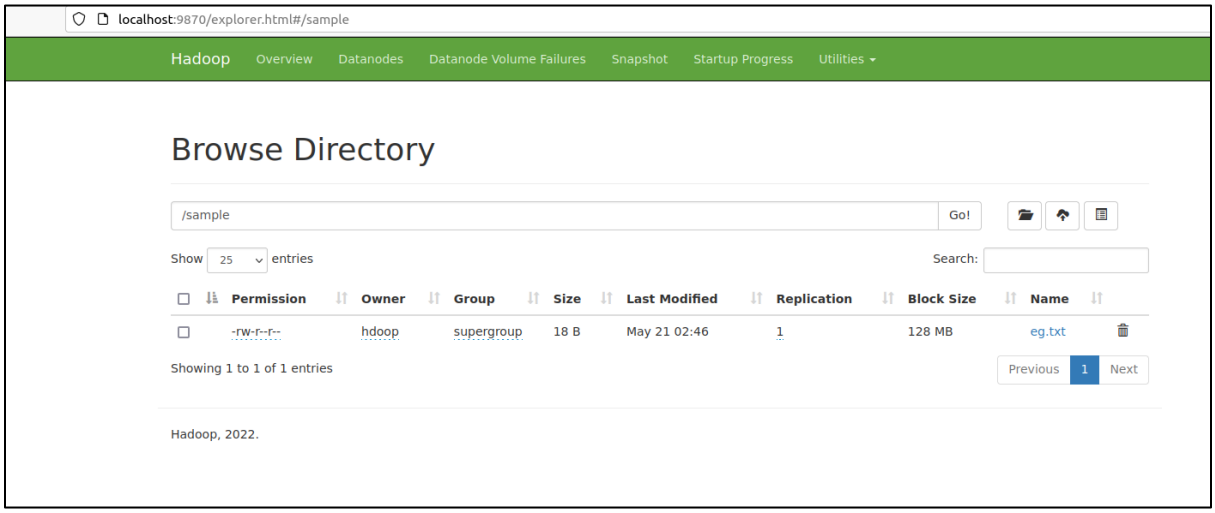
Create a new file in the terminal

```
vi eg.txt
```

```
hdoop@PRLAB-22:~$ vi eg.txt
```

Type contents and save it and exit out of the file. Move the file to the newly created directory.

```
hadoop fs -put eg.txt /sample
```



c) Retrieve File Contents

```
hadoop fs -cat /sample/eg.txt
```

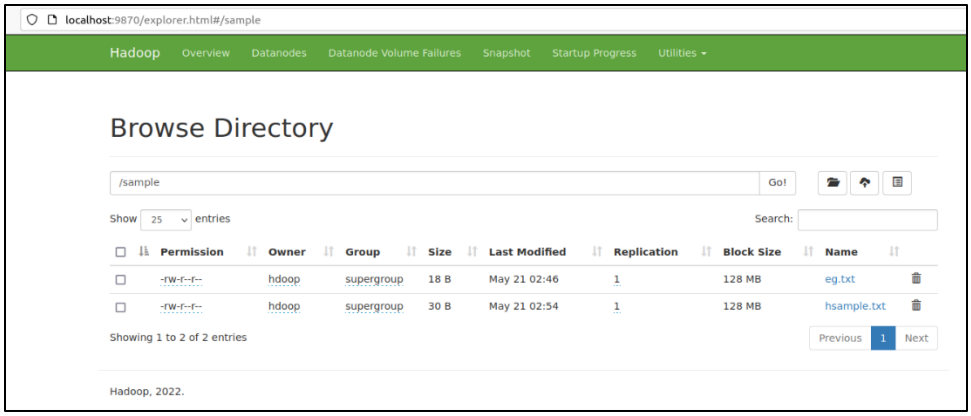
```
hdoop@PRLAB-22:~$ hadoop fs -cat /sample/eg.txt
welcome to hadoop
```

d) Delete File

Create a new file and move to the directory 'sample'

```
vi hsample.txt
hadoop fs -put hsample.txt /sample
hadoop fs -cat /sample/hsample.txt
```

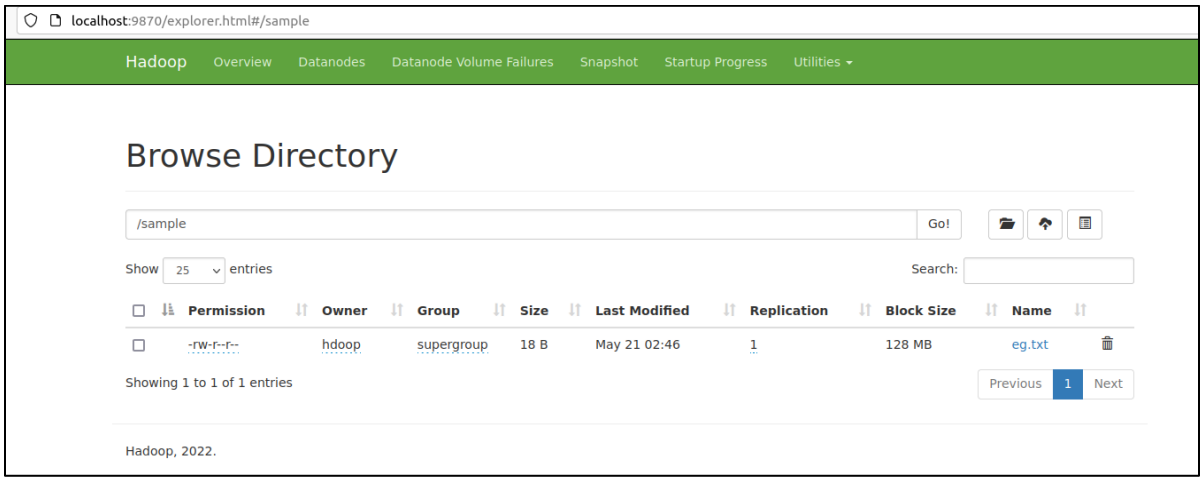
```
hdoop@PRLAB-22:~$ vi hsample.txt
hdoop@PRLAB-22:~$ hadoop fs -put hsample.txt /sample
hdoop@PRLAB-22:~$ hadoop fs -cat /sample/hsample.txt
Newly created file for hadoop
```




```
hadoop fs -rm /sample/hsample.txt
```

```
hdoop@PRLAB-22:~$ hadoop fs -rm /sample/hsample.txt
Deleted /sample/hsample.txt
```

After deletion



Result: Thus, file management in HDFS of Hadoop has been successfully carried out.

Exp. No.: 7 a	**Hive Installation**

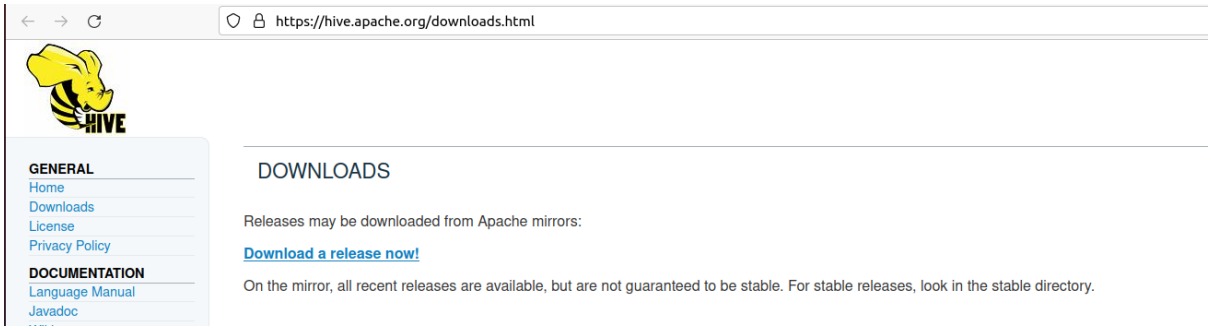
Aim:

To install Hive.

Procedure:

1. Download Apache HIVE latest release

Visit https://hive.apache.org/downloads.html



2. In terminal, switch to the user where Hadoop is installed

su – hdoop wget https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz

```
itadmin@PRLAB-22:~$ su - hdoop
Password:
hdoop@PRLAB-22:~$ wget https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
--2022-05-27 02:29:24--  https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 278813748 (266M) [application/x-gzip]
Saving to: 'apache-hive-3.1.2-bin.tar.gz'

apache-hive-3.1.2-bin.tar.gz      68%[=================================>                ] 182.46M  14.6MB/s    eta 7s
```

3. Extract the files to initiate the Hive installation

tar xvzf apache-hive-3.1.2-bin.tar.gz

```
hdoop@PRLAB-22:~$ tar xvzf apache-hive-3.1.2-bin.tar.gz
apache-hive-3.1.2-bin/LICENSE
apache-hive-3.1.2-bin/NOTICE
apache-hive-3.1.2-bin/RELEASE_NOTES.txt
apache-hive-3.1.2-bin/binary-package-licenses/asm-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.google.protobuf-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.ibm.icu.icu4j-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.sun.jersey-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/com.thoughtworks.paranamer-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javax.transaction.transaction-api-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/javolution-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/jline-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/NOTICE
apache-hive-3.1.2-bin/binary-package-licenses/org.abego.treelayout.core-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr.antlr4-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.antlr.stringtemplate-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.codehaus.janino-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.jamon.jamon-runtime-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.jruby-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.mozilla.rhino-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/org.slf4j-LICENSE
apache-hive-3.1.2-bin/binary-package-licenses/sqlline-LICENSE
apache-hive-3.1.2-bin/examples/files/2000_cols_data.csv
apache-hive-3.1.2-bin/examples/files/3col_data.txt
apache-hive-3.1.2-bin/examples/files/4col_data.txt
```

4. Configure Hive Environment Variables (bashrc)

```
vi .bashrc
```

Append the following Hive environment variables to the .bashrc file:

```
export HIVE_HOME= "/home/hdoop/apache-hive-3.1.2-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

```
hdoop@PRLAB-22: ~
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
#Hadoop Related Options
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hdoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
#Hive Related Options
export HIVE_HOME="/home/hdoop/apache-hive-3.1.2-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

5. Apply the changes to the current environment

```
source ~/.bashrc
```

```
hdoop@PRLAB-22:~$ source ~/.bashrc
```

6. Edit hive-config.sh file

```
vi $HIVE_HOME/bin/hive-config.sh
```

```
hdoop@PRLAB-22: ~/apache-hive-3.1.2-bin/bin
        HIVE_CONF_DIR=$confdir
        ;;
    --auxpath)
        shift
        HIVE_AUX_JARS_PATH=$1
        shift
        ;;
    *)
        break;
        ;;
  esac
done


# Allow alternate conf dir location.
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf}"

export HIVE_CONF_DIR=$HIVE_CONF_DIR
export HADOOP_HOME=/home/hdoop/hadoop
export HIVE_AUX_JARS_PATH=$HIVE_AUX_JARS_PATH
```

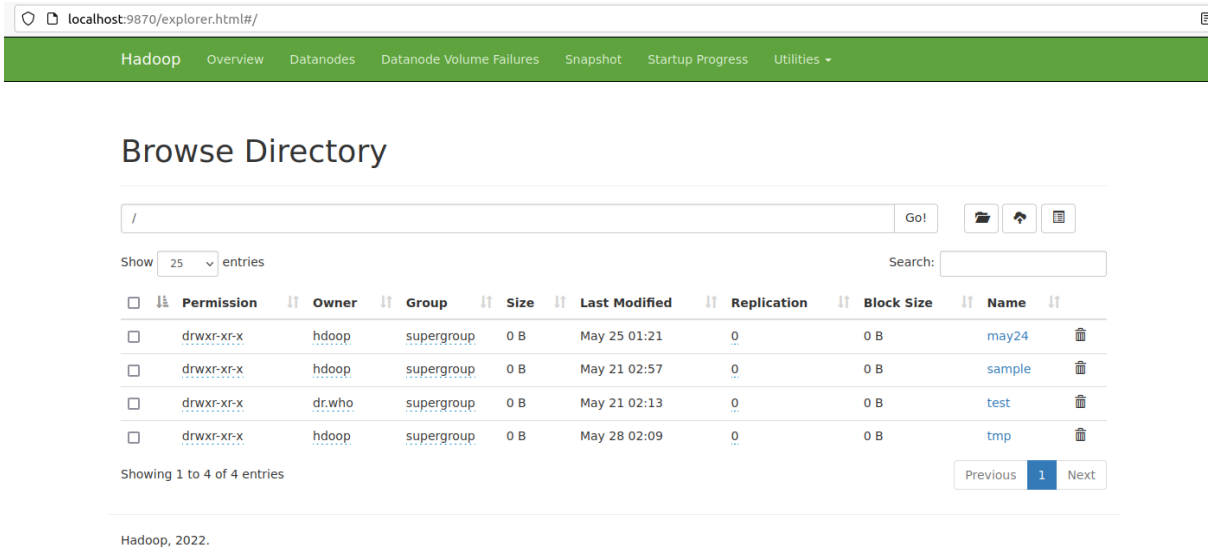
Start hadoop file system and yarn using the commands which were given during hadoop installation procedure.

```
itadmin@PRLAB-22:~$ su - hdoop
Password:
hdoop@PRLAB-22:~$ cd hadoop
hdoop@PRLAB-22:~/hadoop$ cd sbin
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [PRLAB-22]
hdoop@PRLAB-22:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hdoop@PRLAB-22:~/hadoop/sbin$ jps
37620 NodeManager
38374 Jps
36616 DataNode
36856 SecondaryNameNode
37467 ResourceManager
36060 NameNode
```

7. Create two separate directories to store data in the HDFS layer

```
hadoop fs -mkdir /tmp
```

```
hdoop@PRLAB-22:~$ hadoop fs -mkdir /tmp
hdoop@PRLAB-22:~$
```

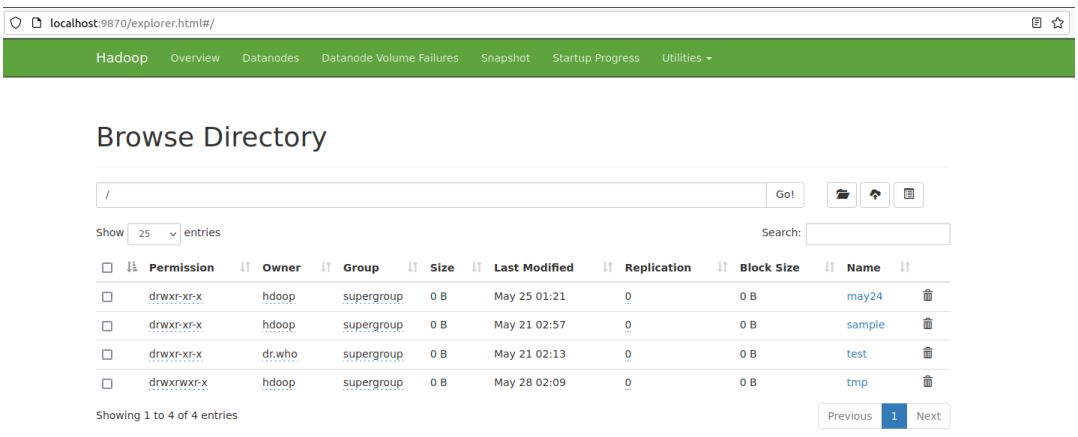



```
hadoop fs -chmod g+w /tmp
```

Add write and execute permissions to tmp group members

Check if the permissions were added correctly

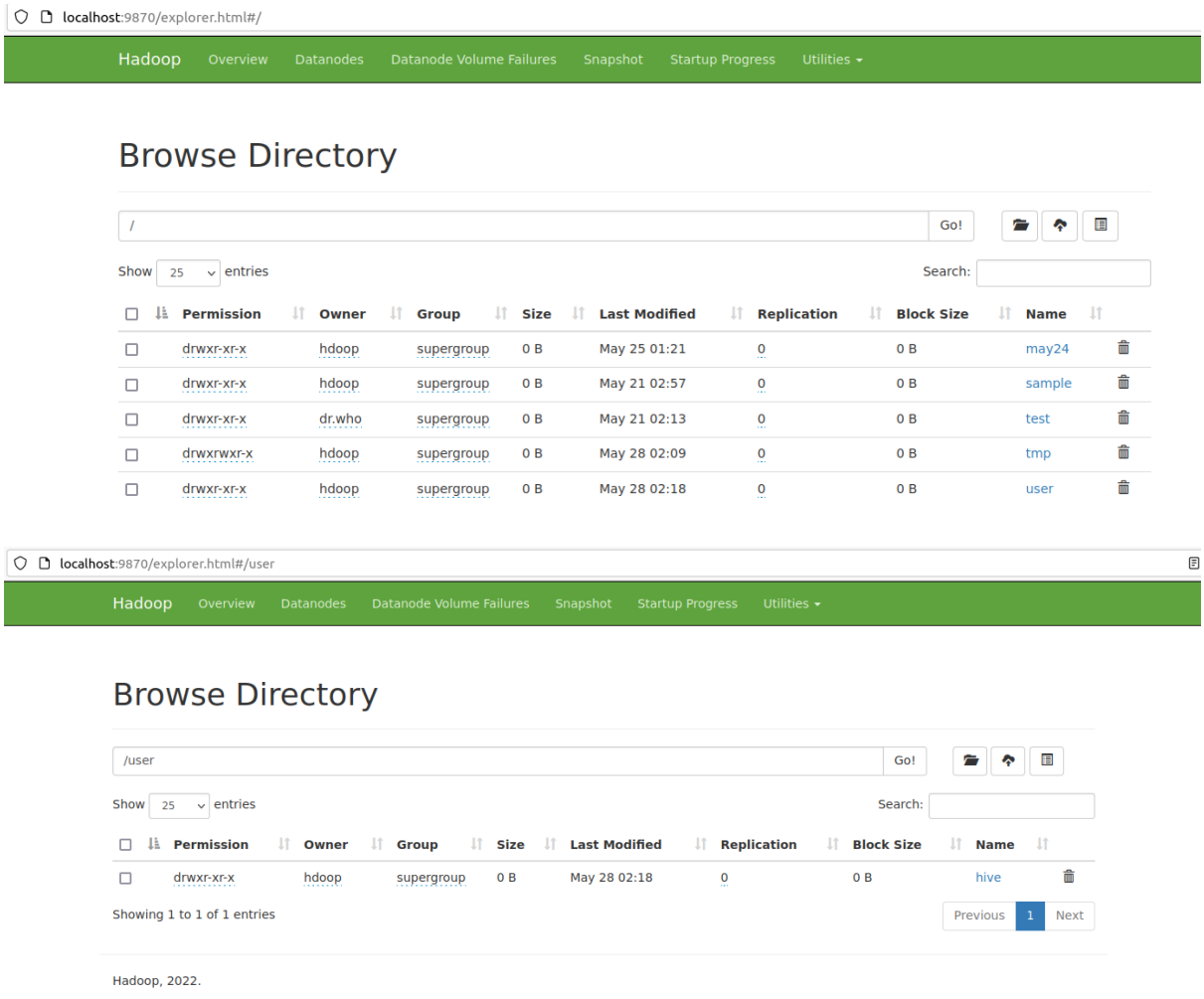
```
hadoop fs -ls /
```

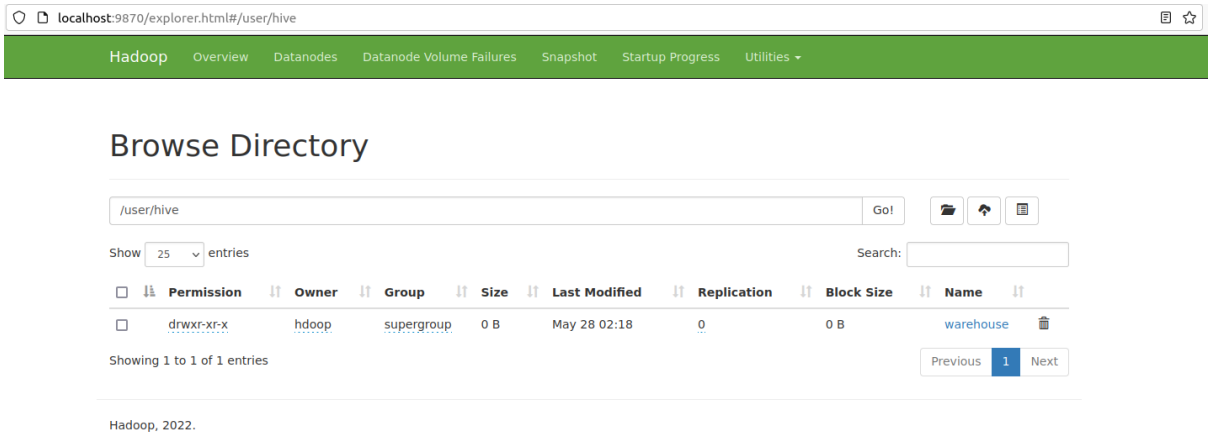


```
hdoop@PRLAB-22:~$ hadoop fs -ls /
Found 4 items
drwxr-xr-x   - hdoop  supergroup          0 2022-05-25 01:21 /may24
drwxr-xr-x   - hdoop  supergroup          0 2022-05-21 02:57 /sample
drwxr-xr-x   - dr.who supergroup          0 2022-05-21 02:13 /test
drwxrwxr-x   - hdoop  supergroup          0 2022-05-28 02:09 /tmp
hdoop@PRLAB-22:~$
```

Create warehouse Directory. Create the warehouse directory within the /user/hive/ parent directory

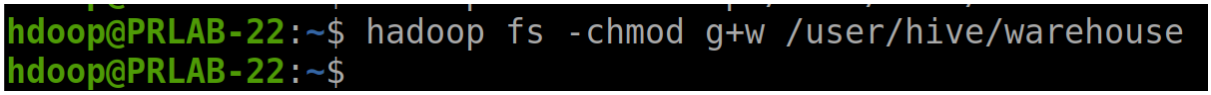
hadoop fs -mkdir -p /user/hive/warehouse



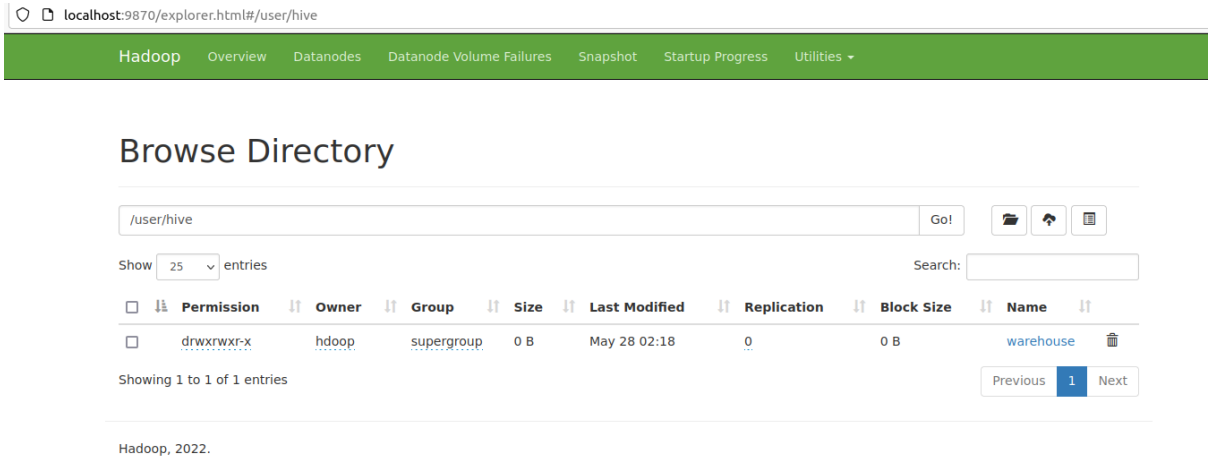


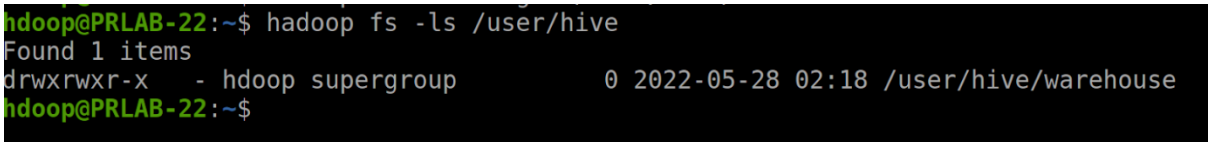
Add write and execute permissions to warehouse group members

```
hadoop fs -chmod g+w /user/hive/warehouse
```



Check if the permissions were added correctly

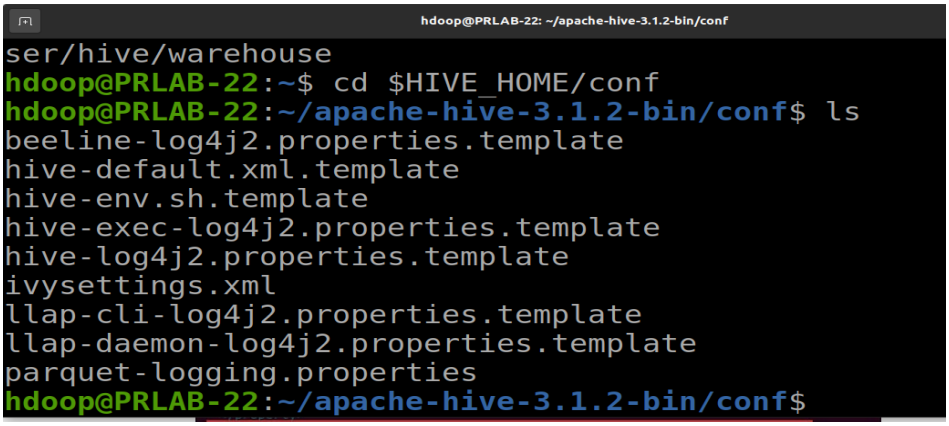




8. Configure hive-site.xml File

```
cd $HIVE_HOME/conf
```

List the files contained in the folder using the ls command



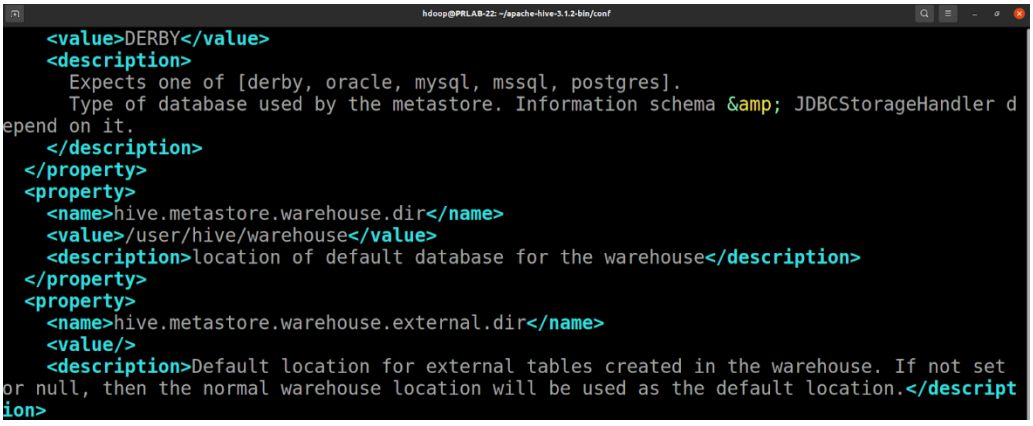
Use the hive-default.xml.template to create the hive-site.xml file:

```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$ cp hive-default.xml.template hive-site.xml
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$
```

Access the hive-site.xml file using the text editor of your choice

```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$ vi hive-site.xml
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/conf$
```

Using Hive in a stand-alone mode rather than in a real-life Apache Hadoop cluster is a safe option



9. Initiate Database

```
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```

```
hdoop@PRLAB-22:~$ $HIVE_HOME/bin/schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.
10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hdoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-
1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Precondition
s.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1357)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1338)
        at org.apache.hadoop.mapred.JobConf.setJar(JobConf.java:536)
        at org.apache.hadoop.mapred.JobConf.setJarByClass(JobConf.java:554)
        at org.apache.hadoop.mapred.JobConf.<init>(JobConf.java:448)
        at org.apache.hadoop.hive.conf.HiveConf.initialize(HiveConf.java:5141)
        at org.apache.hadoop.hive.conf.HiveConf.<init>(HiveConf.java:5104)
        at org.apache.hive.beeline.HiveSchemaTool.<init>(HiveSchemaTool.java:96)
        at org.apache.hive.beeline.HiveSchemaTool.main(HiveSchemaTool.java:1473)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
```

10. Remove the existing guava file from the Hive lib directory

```
rm $HIVE_HOME/lib/guava-19.0.jar
```

```
hdoop@PRLAB-22:~$ rm $HIVE_HOME/lib/guava-19.0.jar
hdoop@PRLAB-22:~$
```

11. Use the schematool command once again to initiate the Derby database

```
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```

```
hdoop@PRLAB-22:~$ $HIVE_HOME/bin/schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.
10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hdoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-
1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

```
Initialization script completed
schemaTool completed
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
hdoop@PRLAB-22:~$
```

12. Launch Hive Client Shell on Ubuntu

cd $HIVE_HOME/bin/hive

```
hdoop@PRLAB-22:~/apache-hive-3.1.2-bin/bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hdoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 981ad83d-79b6-4856-89f7-9a1c51cef3f2

Logging initialized using configuration in jar:file:/home/hdoop/apache-hive-3.1.2-bin/lib/hive-common-3.1.2
.jar!/hive-log4j2.properties Async: true
Exception in thread "main" java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path i
n absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D
        at org.apache.hadoop.fs.Path.initialize(Path.java:263)
        at org.apache.hadoop.fs.Path.<init>(Path.java:221)
        at org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.java:710)
        at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:627)
        at org.apache.hadoop.hive.ql.session.SessionState.beginStart(SessionState.java:591)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:747)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:683)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

```
Caused by: java.net.URISyntaxException: Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsyste
m:user.name%7D
        at java.net.URI.checkPath(URI.java:1823)
        at java.net.URI.<init>(URI.java:745)
        at org.apache.hadoop.fs.Path.initialize(Path.java:260)
        ... 12 more
```

13. Error may occur when hive-shell started before metastore_db service. To avoid this just delete or move your metastore_db and try the below command

```
$ mv metastore_db metastore_db.tmp
```

14. Once again to initiate the Derby database

```
$ schematool -dbType derby -initSchema
```

```
tar xvzf apache-hive-3.1.2-bin.tar.gz
```

Result:

Thus, Hive has been successfully installed.

Exp. No:7 b	**Hive CRUD Operations**

Aim:

To implement a schema definition in Hive and perform CRUD operations.

Procedure:

Codes and Output:

a) Table Creation

```
Create table studentnew (id int, name string, location string) cluster by (location) into 3 buckets row format delimited fields terminated by ‘,’ lines terminated by ‘\n’ stored as orc TABLEPROPERTIES (‘transactional’ = ‘true’);
```

Output:

```
hive> Set hive.support.concurrency = true;
hive> Set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
hive> set hive.compactor.worker.threads =1;
hive> create table studentnew  (id int, name string, location string) clustered by (location) into 3 bucket
s row format delimited fields terminated by ',' lines terminated by '\n' stored as orc TBLPROPERTIES ('tran
sactional'='true');
OK
Time taken: 0.807 seconds
```

b) Insertion of Records

```
INSERT INTO TABLE studentnew VALUES (101,’abc’,’GST Road’),
 (102,’A’,’Guindy’),  (103,’PRABU’,’henry road’),  (104,’KUMAR’,’gandhi road’);
```

Output:

```
hive> insert into table studentnew values(101,'abc','GST Road');
Query ID = hdoop_20220531030502_54f3eff2-c5fa-452d-85e0-c728e6b2a704
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0001, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_
0001/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 3
2022-05-31 03:05:13,825 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:05:16,916 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.7 sec
2022-05-31 03:05:22,021 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 4.5 sec
2022-05-31 03:05:24,050 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.36 sec
MapReduce Total cumulative CPU time: 6 seconds 360 msec
Ended Job = job_1653941890266_0001
Loading data to table default.studentnew
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
```

c) Retrieve of Records

```
select * from studentnew
```

Output:

```
hive> select * from studentnew;
OK
101     abc     GST Road
102     prabhu  guindy
103     gandhi  radha nagar chrompet
Time taken: 0.333 seconds, Fetched: 3 row(s)
```

d) Updation of Records

```
Update studentnew SET id = 113 WHERE id = 103;
```

Output:

```
hive> Update studentnew SET id = 113 WHERE id = 103;
Query ID = hdoop_20220531031127_9ec9ac2d-39f9-4bd5-b809-5c4ea6ceb5d2
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0007, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_0007/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0007
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 3
2022-05-31 03:11:33,595 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:11:38,685 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.12 sec
2022-05-31 03:11:42,751 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 9.56 sec
2022-05-31 03:11:43,770 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 12.97 sec
MapReduce Total cumulative CPU time: 12 seconds 970 msec
Ended Job = job_1653941890266_0007
Loading data to table default.studentnew
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 3   Cumulative CPU: 12.97 sec   HDFS Read: 43124 HDFS Write: 1801 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 970 msec
OK
Time taken: 18.984 seconds
```

```
hive> select * from studentnew;
OK
101     abc     GST Road
102     prabhu  guindy
113     gandhi  radha nagar chrompet
Time taken: 0.16 seconds, Fetched: 3 row(s)
```

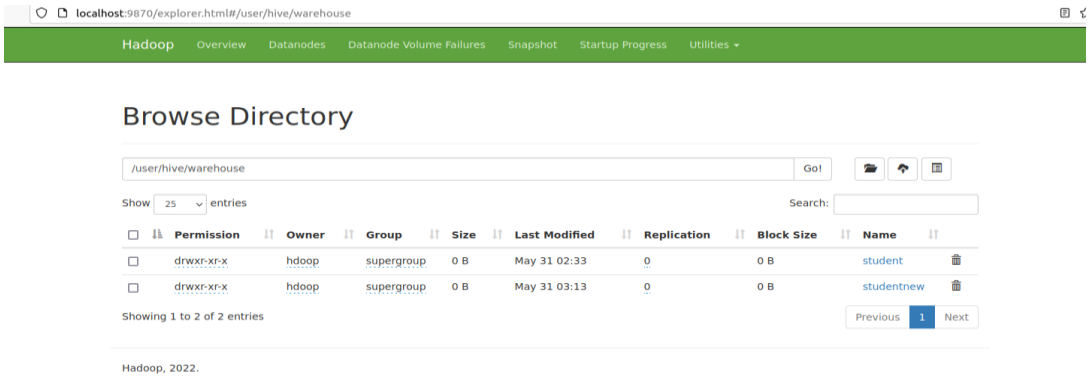
e) Deletion of Records

```
delete from studentnew where id=102;
```

Output:

```
hive> delete from studentnew where id=102;
Query ID = hdoop_20220531031339_a531bcab-2da2-4826-81af-a8d4f44dc0f4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1653941890266_0008, Tracking URL = http://PRLAB-22:8088/proxy/application_1653941890266_0008/
Kill Command = /home/hdoop/hadoop/bin/mapred job  -kill job_1653941890266_0008
Hadoop job information for Stage-1: number of mappers: 4; number of reducers: 3
2022-05-31 03:13:45,465 Stage-1 map = 0%,  reduce = 0%
2022-05-31 03:13:51,592 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 11.5 sec
2022-05-31 03:13:55,652 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 13.28 sec
2022-05-31 03:13:57,690 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.71 sec
MapReduce Total cumulative CPU time: 16 seconds 710 msec
Ended Job = job_1653941890266_0008
Loading data to table default.studentnew
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4  Reduce: 3   Cumulative CPU: 16.71 sec   HDFS Read: 51452 HDFS Write: 896 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 710 msec
OK
Time taken: 19.13 seconds
hive> select * from studentnew;
OK
101     abc     GST Road
113     gandhi  radha nagar chrompet
Time taken: 0.168 seconds, Fetched: 2 row(s)
```

View the table created in Browser-UI



Result: Thus, a schema definition has been created in Hive and CRUD operations have been performed.

Exp. No.: 8 a	**MongoDB Basics**

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(7df78ad8902c),
  'Name': 'ABC',
  'Age': 20,
  'Reg_No': '201901a',
  'Subjects': [ 'Math', 'English'],
  'Details': {   'Father' : 'XY',
                 'Mother' : 'PQ'
             }
}
```

Operations

1. Use

- To create and use a database. If the database exists, it returns existing database
- Syntax:
  - use Database_name

2. Create

- To create/insert a new document to a collection
- Syntax:
  - db.collection_name.insertOne( )
  - db.collection_name.insertMany( )

3. Read

- To retrieve documents from a collection
- Syntax:
  - db.collection_name.find( )

4. Update

- To modify existing documents in a collection
- Syntax:
  - db.collection_name.updateOne( )
  - db.collection_name.updateMany( )
  - db.collection_name.replaceOne( )

5. Delete

- To remove documents from a collection
- Syntax:
  - db.collection_name.deleteOne( )
  - db.collection_name.deleteMany( )

Result:

Thus, a study on the basics of MongoDB and its operations has been made.

2. Pick any system as an example, create database and its corresponding tables. Also populate the tables with data. Apply insert /search operations

Aim:

To design a student management system using MongoDB and perform CRUD operations on it.

Code:

i. Create and Use Database 'StuDB'

```
use StuDb
db
```

Output:

```
> use StuDb
switched to db StuDb
> db
StuDb
```

ii. Insert

➢ Insert one document into collection

```
db.StuDb.insert( { "StuId" : 1, "Name" : "Ashwin", "Age" : 19, "Marks" : { "Math" : 100, "English" : 99 } } )
Db.StuDB.find( ).pretty( )
```

Output:

```
> db.StuDb.find().pretty()
{
        "_id" : ObjectId("6255a2667bb66dce7e8855ae"),
        "StuId" : 1,
        "Name" : "Ashwin",
        "Age" : 19,
        "Marks" : {
                "Math" : 100,
                "English" : 99
        }
}
```

- Insert multiple documents into collection

```
db.StuDb.insertMany( [ { “StuId” : 2, “Name” : “Charlie”, “Age” : 20, “Marks” : { “Math” : 98, “English” : 100 } } , { “StuId” : 3, “Name” : “David”, “Age” : 20, “Marks” : { “Math” : 89, “English” : 78 } }, { “StuId” : 4, “Name” : “Davis”, “Age” : 20, “Marks” : { “Math” : 96, “English” : 95 } } ] )
```

Output:

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6255a46a7bb66dce7e8855af"),
                ObjectId("6255a46a7bb66dce7e8855b0"),
                ObjectId("6255a46a7bb66dce7e8855b1")
        ]
}
```

iii. Search

- Find document where ‘StuId’ is 3

```
db.StuDb.find( { ‘StuID’ : 3 } ).pretty( )
```

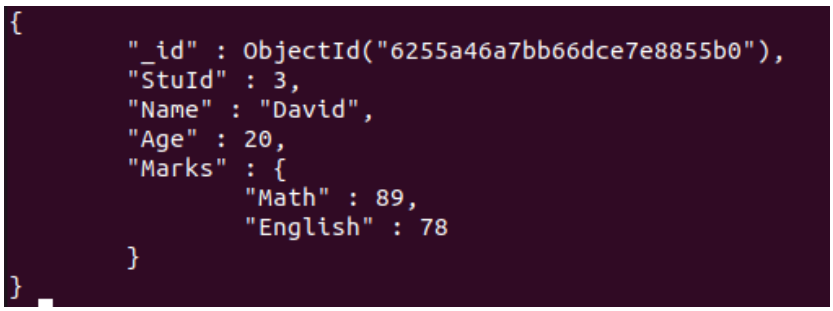
Output:

```
{
        "_id" : ObjectId("6255a46a7bb66dce7e8855b0"),
        "StuId" : 3,
        "Name" : "David",
        "Age" : 20,
        "Marks" : {
                "Math" : 89,
                "English" : 78
        }
}
```

➢ Find documents where marks in English is less than 80

```
db.StuDb.find( { 'Marks.English' : { $lt : 80 } } )
```

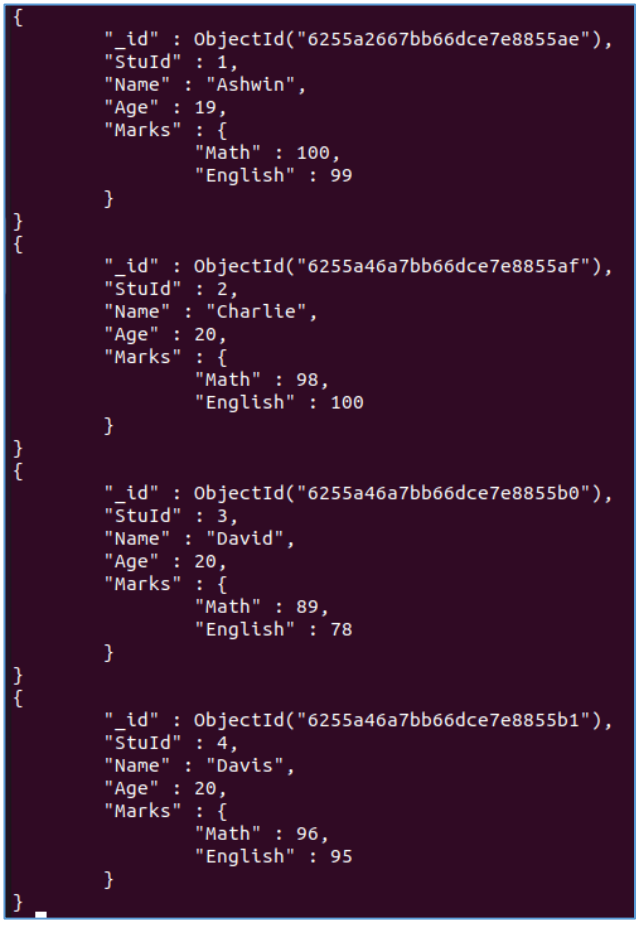
Output:



iv. View

```
db.StuDb.find( ).pretty( )
```

Output:

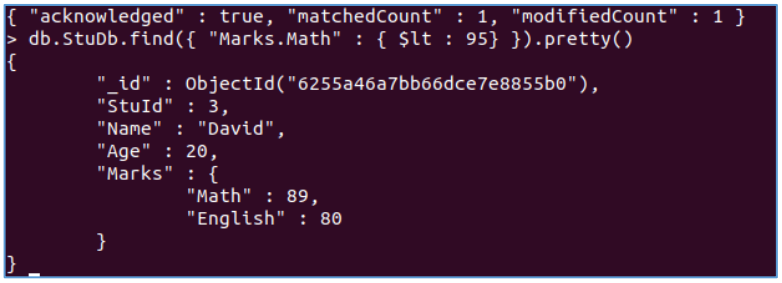


v. Update

Update the marks in English to 80 for all documents in collection where marks in math is less than 95.

```
db.StuDb.updateMany( { 'Marks.Math' : { $lt : 95 }}, { $set : { 'Marks.English' : 80 } } )
db.StuDb.find( { 'Marks.Math' : { $lt : 95 } } ).pretty( )
```

Output:

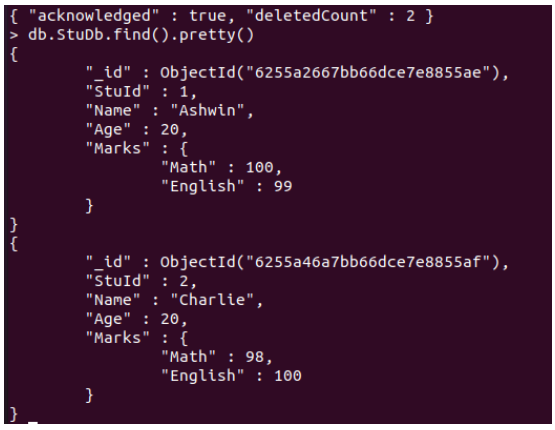


vi. Delete

Delete all documents that have marks in math less than 97.

```
db.StuDb.deleteMany( { 'Marks.Math' : { $lt : 97 } } )
db.StuDb.find( ).pretty( )
```

Output:



Result:

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

Exp. No.: 8 b

MongoDB Data Replication

Aim:

To perform data replication in MongoDB.

Procedure:

1. Creating a node in port 27021

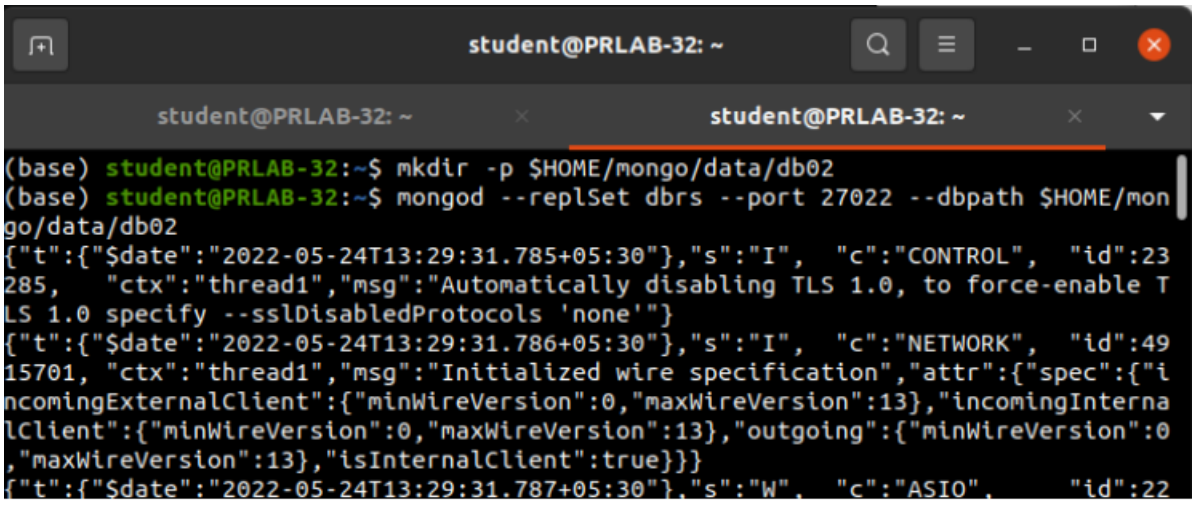
```
mkdir -p $HOME/mongo/data/db01
mongod --replSet dbrs --port 27021 --dbpath $HOME/mongo/data/db01
```

2. Open another terminal. Create node in a different port, 27022

```
mkdir -p $HOME/mongo/data/db02
mongod --replSet dbrs --port 27022 --dbpath $HOME/mongo/data/db02
```

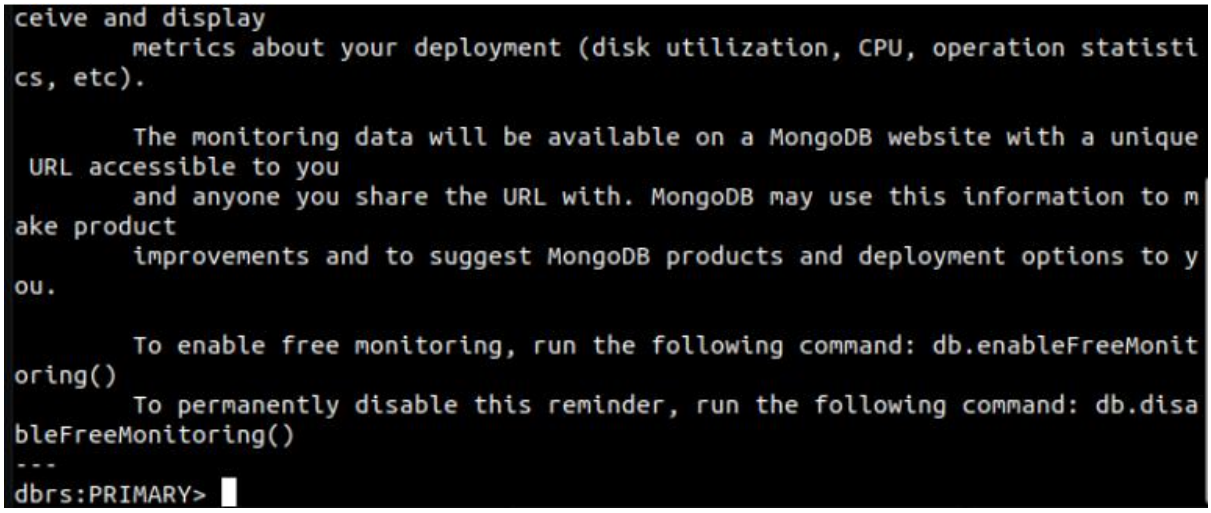
3. In third terminal, create a third node

```
mkdir -p $HOME/mongo/data/db03
mongod --replSet dbrs --port 27023 --dbpath $HOME/mongo/data/db03
```



4. Connect to any one Mongo Daemon

```
mongo --port 27021
```




```
rsconf = {
 _id: "dbrs",
 members: [
 {
 _id: 0,
 host: "127.0.0.1:27021",
 priority: 3
 },
 {
 _id: 1,
 host: "127.0.0.1:27022",
 priority: 1,
 },
 {
 _id: 2,
 host: "127.0.0.1:27023",
 priority: 2
 }
 ]
};
```

```
db.getMongo().setReadPref('secondary')
```

```
        ]
}
> rs.initiate(rsconf);
{ "ok" : 1 }
```

Result:

Thus, data replication in MongoDB has been successfully carried out.

Exp. No.: 9	**Visualization Tools in Python**

Aim:

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

Theory:

a. Matplotlib

Matplotlib is a visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Functions in Matplotlib

S. No	Function	Description
1.	matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)	Make a bar plot for two attributes
2.	matplotlib.pyplot.boxplot(x, notch=None)	Make a box and whisker plot
3.	hexbin( ) : matplotlib.pyplot.hexbin(x, y, C=None, gridsize=100, bins=None, xscale='linear', yscale='linear')	Make a 2D hexagonal binning plot of points x, y
4.	matplotlib.pyplot.hist(x, bins=None, range=None, density=None, weights=None)	Plot a histogram
5.	matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None)	Plot a pie chart
6.	matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)	Plot y versus x as lines and/or markers
7.	matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None)	A scatter plot of y vs x with varying marker size and/or color

b. Diagram

Diagrams lets you draw the cloud system architecture in Python code. It was made for prototyping a new system architecture design without any design tools. You can also describe or visualize the existing system architecture as well.

Functions in Diagram

S. No	Function	Description
1.	Diagram("Diagram Name")	Create a diagram context with Diagram class
2.	ELB("Node Name")	To perform elastic load balance
3.	Route53("dns")	To establish routing connection
4.	Cluster("Cluster Name")	To create a cluster context
5.	RDS("Node Type")	To create a relational database service instance
6.	SQS("event queue")	To create a simple queue service
7.	Edge(color="*", style="*")	To create an edge between two nodes

c. MayaVi

It is a free, easy to use scientific data visualizer. It is written in Python and uses the amazing Visualization Toolkit (VTK) for the graphics. It provides a GUI written using Tkinter. MayaVi is free and distributed and it is also cross platform.

Functions in MayaVi

S. No	Function	Description
1.	barchart(*args, **kwargs)	Plots vertical glyphs (like bars) scaled vertical, to do histogram-like plots.
2.	contour3d(*args, **kwargs)	Plots iso-surfaces for a 3D volume of data supplied as arguments
3.	contour_surf(*args, **kwargs)	Plots the contours of a surface using grid-spaced data for elevation supplied as a 2D array
4.	flow(*args, **kwargs)	Creates a trajectory of particles following the flow of a vector field
5.	imshow(*args, **kwargs)	View a 2D array as an image
6.	mesh(*args, **kwargs)	Plots a surface using grid-spaced data supplied as 2D arrays
7.	plot3d(*args, **kwargs)	Draws lines between points

d. Seaborn

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Functions in Seaborn

S. No	Function	Description
1.	seaborn.distplot(x, bins = *, kde = *)	To plot approximate probability density across the y-axis
2.	pie( )	To plot a pie chart
3.	barh( )	To plot a bar graph
4.	scatter(x, y)	To plot a scatter plot
5.	jointplot(x, y, kind = 'reg')	Creates a regression line between 2 numerical parameters in the jointplot(scatterplot) to help visualize their linear relationships
6.	pairplot(x)	To plot multiple scatter plots at a time
7.	heatmap( )	To generate a heatmap

e. VisPy

VisPy is a library designed for use by data scientists, and it's intended to make creating complex, interactive visualizations as quickly as possible. VisPy takes advantage of the extra processing power granted by GPUs. This makes the rendering of large datasets faster than other libraries. VisPy graphs and charts can be scaled up, making visualizations out of thousands or even millions of points of data.

Functions in VisPy

S. No	Function	Description
1.	vispy.color.BaseColormap(colors=None)	Class representing a colormap
2.	texture_lut()	Return a texture2D object for LUT after its value is set. Can be None
3.	vispy.geometry.MeshData( )	Class for storing and operating on 3D mesh data
4.	vispy.geometry.curves.curve3_bezier(p1, p2, p3)	Generate the vertices for a quadratic Bezier curve
5.	vispy.io.imread(filename, format=None)	Read image data from disk
6.	vispy.io.imsave(filename, im, format=None)	Save image data to disk
7.	vispy.plot.fig.Fig( )	Create a figure window

f. PyQtGraph

PyQtGraph is a pure-python graphics and GUI library built on PyQt / PySide and numpy. It is intended for use in mathematics / scientific / engineering applications. Despite being written entirely in python, the library is very fast due to its heavy leverage of NumPy for number crunching and Qt's GraphicsView framework for fast display.

Functions in PyQtGraph

S. No	Function	Description
1.	pyqtgraph.plot( )	Create a new plot window showing data
2.	PlotWidget.plot( )	Add a new set of data to an existing plot widget
3.	GraphicsLayout.addPlot( )	Add a new plot to a grid of plots
4.	pyqtgraph.image( )	To display 2D or 3D data
5.	pyqtgraph.opengl.GLViewWidget()	To create a widget to display 3D objects
6.	pyqtgraph.opengl.GLGridItem()	To create a grid
7.	mkPen( )	To build a pen with desired configurations

g. Chartify

Chartify is an open-source data visualization library from Spotify that makes it easy for data analysts to create charts and graphs. Chartify is built on top of Bokeh, which is a very popular data visualization library.

Functions in Chartify

S. No	Function	Description
1.	chartify.Chart( )	create a basic chart
2.	chartify.Chart(x_axis_type='datetime')	To set the datatype of the x-axis
3.	set_title( )	To set the title of the chart
4.	set_subtitle( )	To set the subtitle of the chart
5.	axes.set_xaxis_label( )	To set the label of the x-axis
6.	plot.scatter(dataframe, x_col, y_col, color_col)	To draw a scatter plot
7.	plot.histogram(data_frame, values_column)	To draw a histogram

Result:

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