

Practical 1

Date: 30/06/2023**Aim:**

To perform data pre-processing of IBM Churn dataset from <https://www.kaggle.com/datasets/yeancz/telco-customer-churn-ibm-dataset>.

- Load data
- Find missing values
- Clean data
- Find co-relations between attributes.
- Remove redundant attributes.
- Normalize data
- Visualize the data

Use numpy, pandas, and matplotlib.

Code:

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

Load the dataset

```
url = "WA_Fn-UseC_-Telco-Customer-Churn.csv"
```

```
df = pd.read_csv(url)
```

Display basic information about the dataset

```
print("Dataset Info:")
```

```
print(df.info())
```

Find missing values

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

```
print("\nMissing Values:")
```

```
print(missing_values)
```

Handle missing values (e.g., replace with mean, median, or drop columns)**# For this example, let's drop rows with missing values**

```
df.dropna(inplace=True)
```

Find correlations between attributes

```
correlation_matrix = df.corr()
```

```
print("\nCorrelation Matrix:")
```

```
print(correlation_matrix)
```

Visualize the correlation matrix using a heatmap

```
plt.figure(figsize=(12, 8))
```

```
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='nearest')
```

```
plt.colorbar()
```

```
plt.xticks(range(len(df.columns)), df.columns, rotation='vertical')
```

```
plt.yticks(range(len(df.columns)), df.columns)
```

```
plt.title('Correlation Matrix')
```

```
plt.show()
```

```

# Remove redundant attributes (e.g., drop highly correlated columns)
# You may want to customize this based on your analysis
df = df.drop(['TotalCharges'], axis=1)

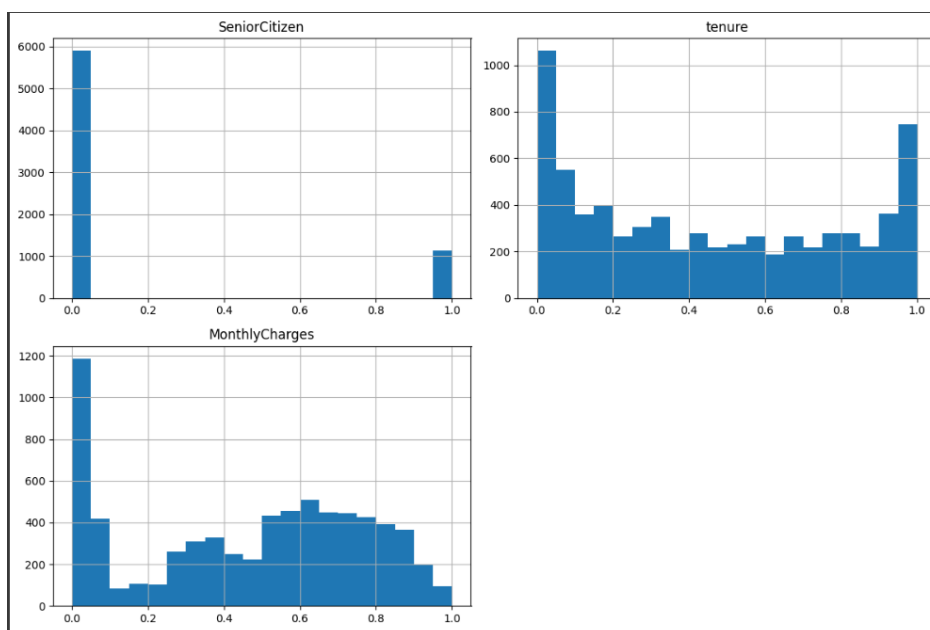
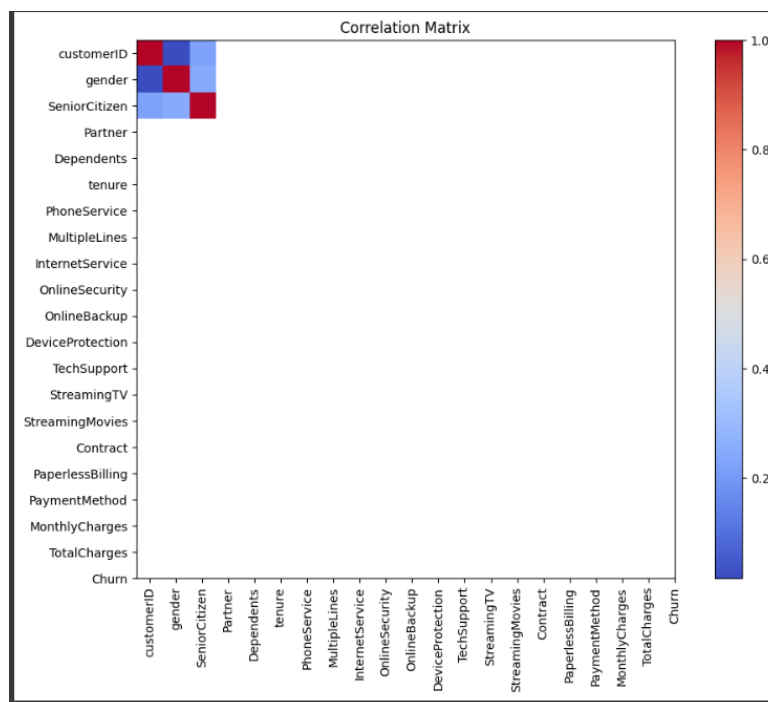
# Normalize data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['tenure', 'MonthlyCharges']] = scaler.fit_transform(df[['tenure', 'MonthlyCharges']])

# Visualize the data
df.hist(bins=20, figsize=(12, 8))
plt.tight_layout()
plt.show()

```

Output Screenshot:

<pre> Dataset Info: <class 'pandas.core.frame.DataFrame'> RangeIndex: 7043 entries, 0 to 7042 Data columns (total 21 columns): # Column Non-Null Count Dtype --- --- 0 customerID 7043 non-null object 1 gender 7043 non-null object 2 SeniorCitizen 7043 non-null int64 3 Partner 7043 non-null object 4 Dependents 7043 non-null object 5 tenure 7043 non-null int64 6 PhoneService 7043 non-null object 7 MultipleLines 7043 non-null object 8 InternetService 7043 non-null object 9 OnlineSecurity 7043 non-null object 10 OnlineBackup 7043 non-null object 11 DeviceProtection 7043 non-null object 12 TechSupport 7043 non-null object 13 StreamingTV 7043 non-null object 14 StreamingMovies 7043 non-null object 15 Contract 7043 non-null object 16 PaperlessBilling 7043 non-null object 17 PaymentMethod 7043 non-null object 18 MonthlyCharges 7043 non-null float64 19 TotalCharges 7043 non-null object 20 Churn 7043 non-null object dtypes: float64(1), int64(2), object(18) memory usage: 1.1+ MB None </pre>	<pre> Missing Values: customerID 0 gender 0 SeniorCitizen 0 Partner 0 Dependents 0 tenure 0 PhoneService 0 MultipleLines 0 InternetService 0 OnlineSecurity 0 OnlineBackup 0 DeviceProtection 0 TechSupport 0 StreamingTV 0 StreamingMovies 0 Contract 0 PaperlessBilling 0 PaymentMethod 0 MonthlyCharges 0 TotalCharges 0 Churn 0 dtype: int64 </pre>
---	---



Conclusion/Summary:

In this practical we performed data pre-processing of IBM Churn dataset from <https://www.kaggle.com/datasets/yeancz/telco-customer-churn-ibm-dataset>.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 2

Date: 08/07/2023

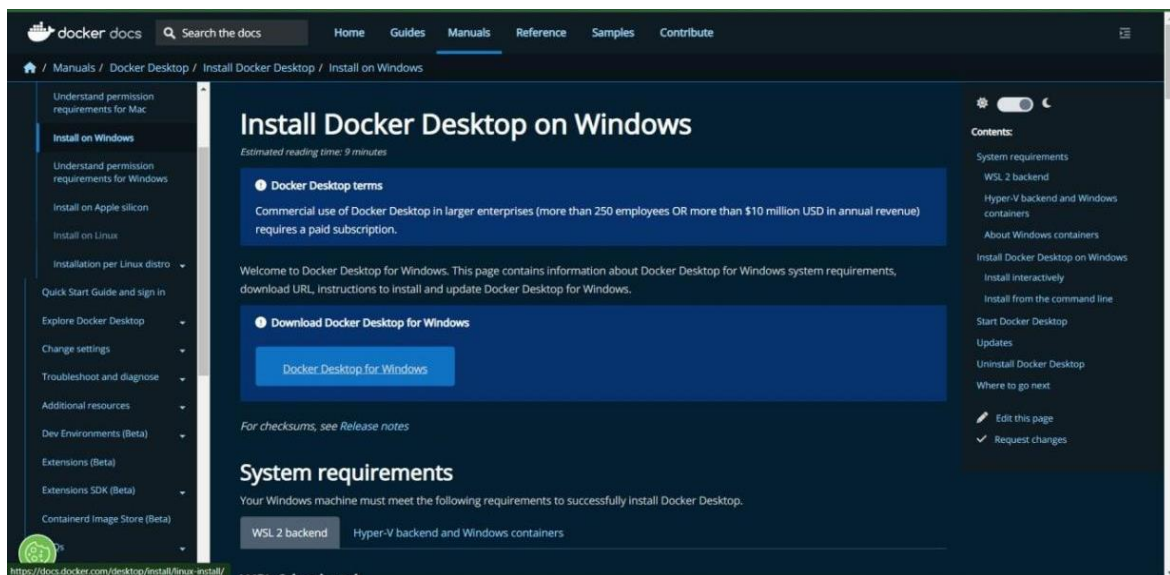
Aim: To install Hadoop framework, configure it and setup a single node cluster. Use web based tools to monitor your Hadoop setup.

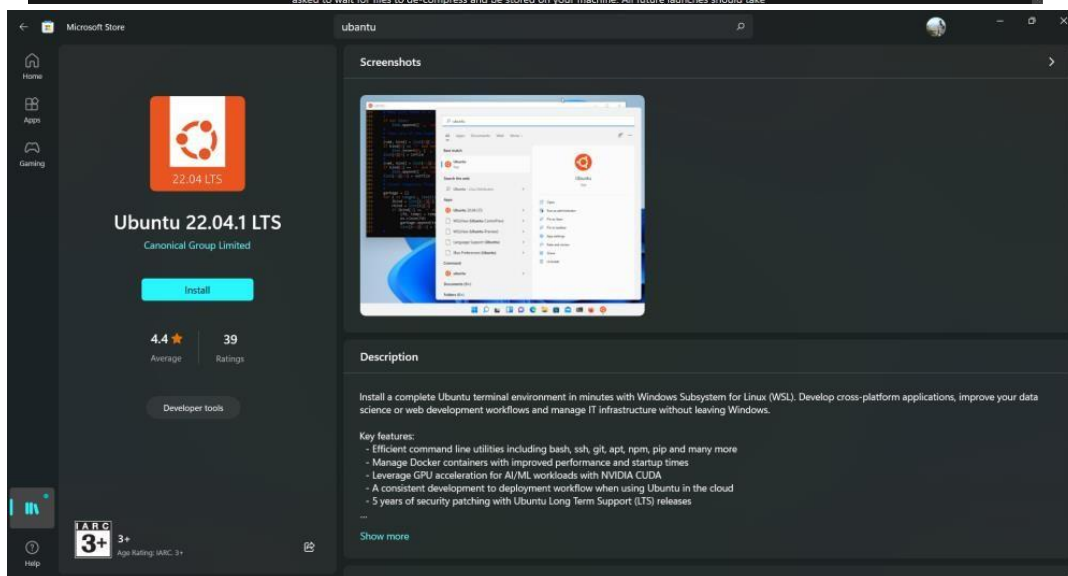
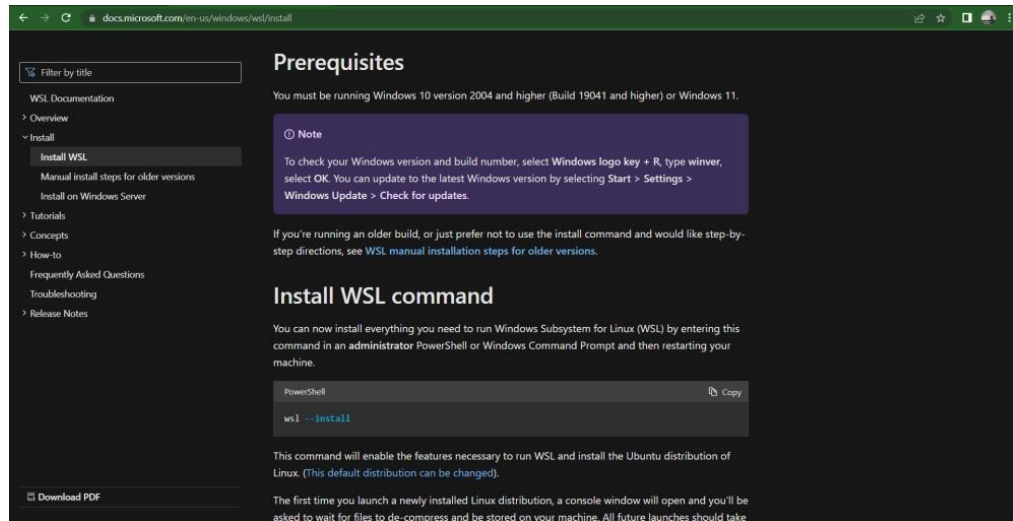
Code:

THEORY:

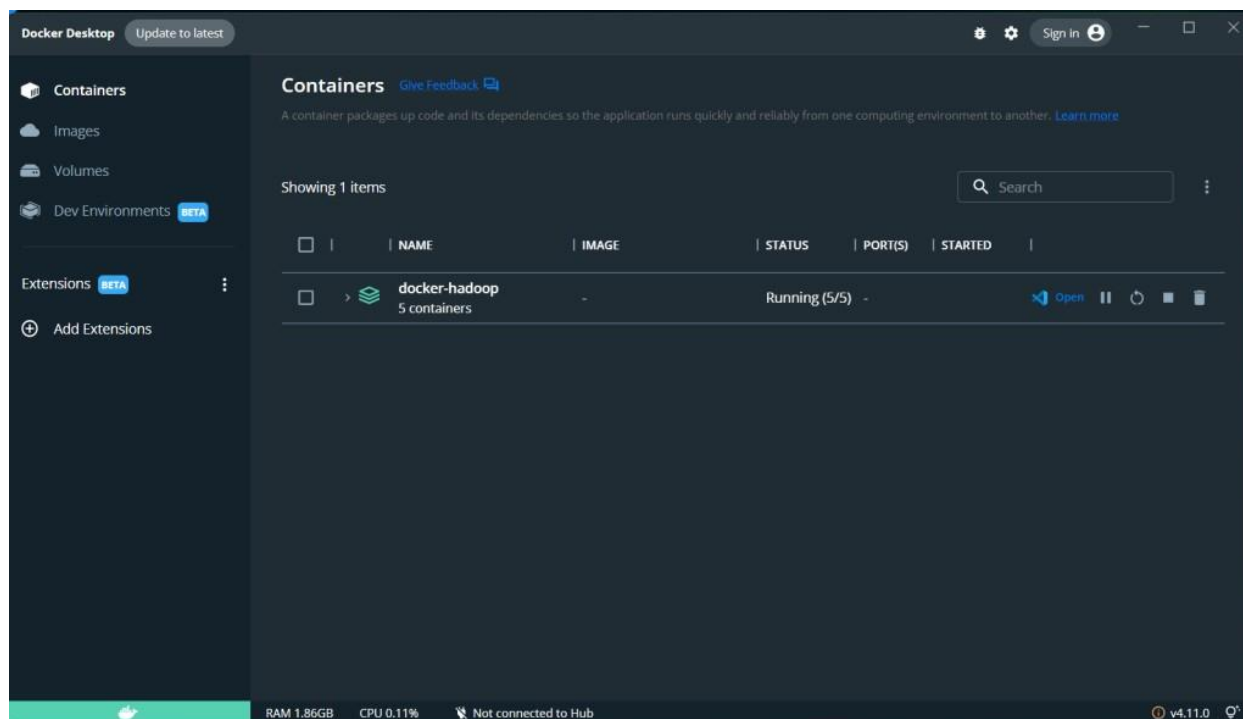
Hadoop:

- The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.
- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.
- First install Docker on windows machine from official website.
- Besides this parallely also download WSL2 with ubuntu from Microsoft store.

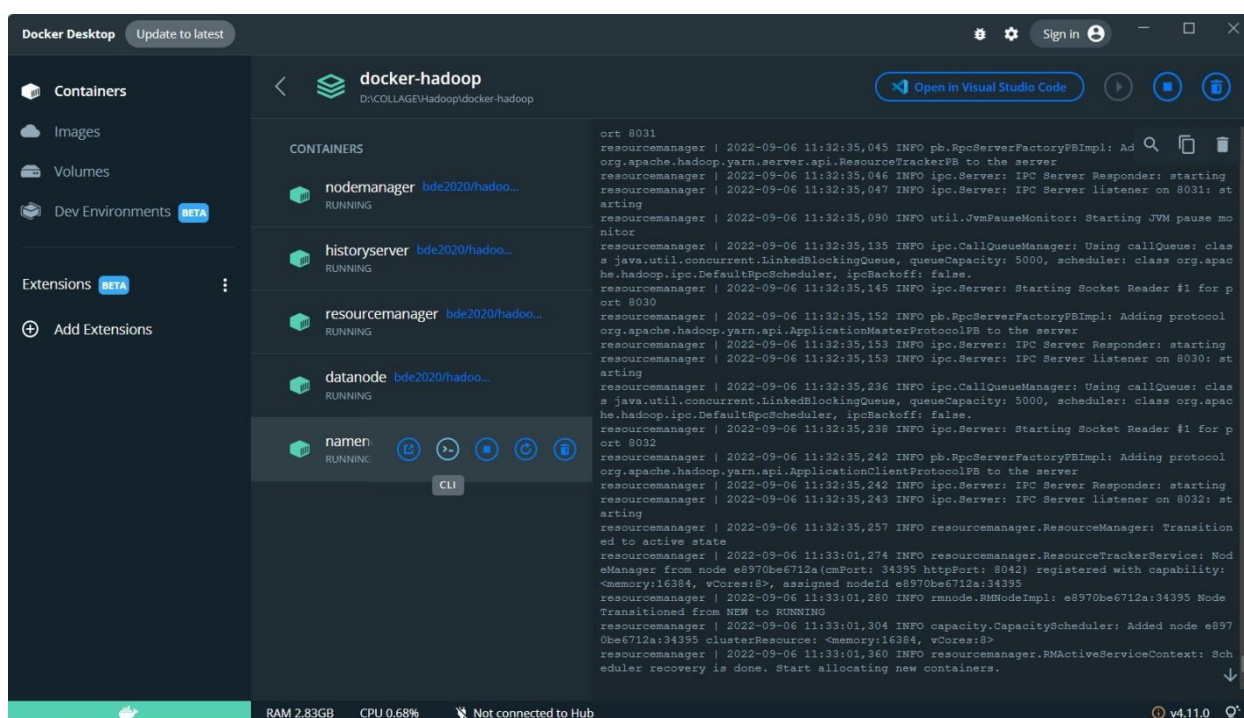




- After downloading all the above-mentioned requirements run and install the docker on your windows machine. Then after download Hadoop from github as the link given here <https://github.com/big-data-europe/docker-hadoop>
- On completing all the download simply run this command.
- `docker-compose up -d`
- This will automatically download all the requirements and make a docker container.



- Then click on NameNode Cli.



- Now open up a browser and go to localhost:9870

Overview 'namenode:9000' (active)

Started:	Tue Sep 06 17:08:41 +0530 2022
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 21:26:00 +0530 2019 by rohitshamaks from branch-3.2.1
Cluster ID:	CID-89f6d306-3e6d-4531-9c26-33e00238edfb
Block Pool ID:	BP-1071918935-172.18.0.6-1661879100818

Summary

Security is off.
Safemode is off.

19 files and directories, 11 blocks (11 replicated blocks, 0 erasure coded block groups) = 30 total filesystem object(s).

Heap Memory used 113.51 MB of 236 MB Heap Memory. Max Heap Memory is 864 MB.

Non Heap Memory used 49.6 MB of 51 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	250.98 GB
Configured Remote Capacity:	0 B
DFS Used:	108.05 KB (0%)

Conclusion/Summary:

In this practical, we learned about how to install Hadoop.

Student Signature & Date

Marks:

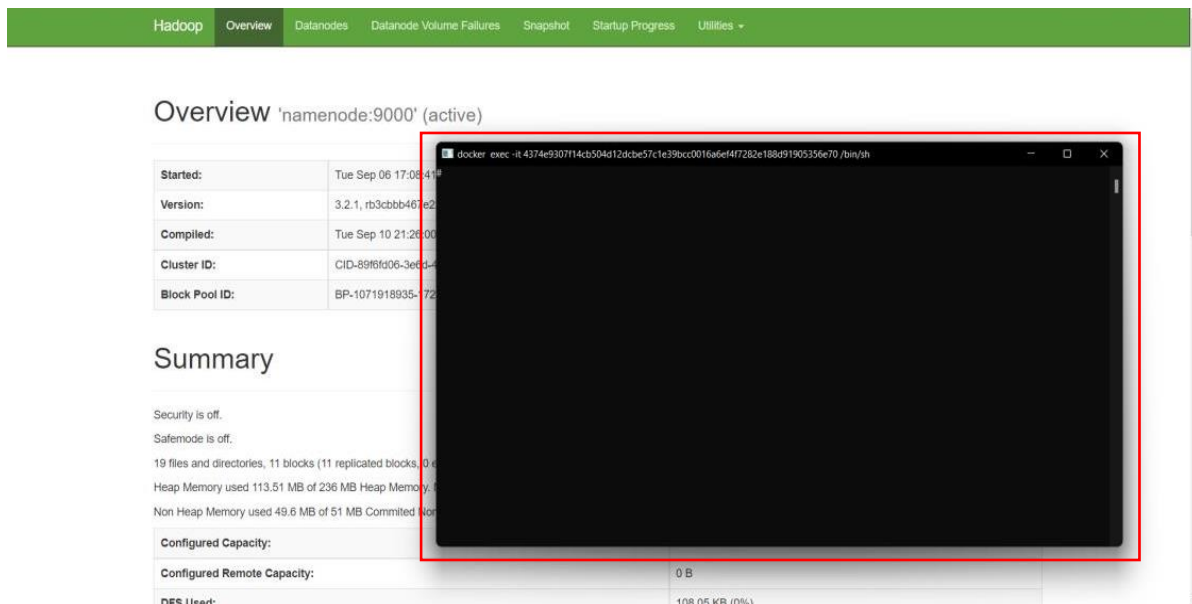
Evaluator Signature & Date

Practical 3

Date: 14/07/2023**Aim:** To implement file management tasks in Hadoop HDFS and perform Hadoop commands.**Code:**

There are many more commands in "\$HADOOP_HOME/bin/hadoop fs" than are demonstrated here, although these basic operations will get you started. Running `./bin/hadoop dfs` with no additional arguments will list all the commands that can be run with the FsShell system. Furthermore, `$HADOOP_HOME/bin/hadoop fs -help commandName` will display a short usage summary for the operation in question, if you are stuck.

- Start the docker and run the Hadoop in it.
- Open the NameNode Cli from docker and navigate to the localhost:9870.
- Here given Cli is for performing command on the NameNode.

**Commands :-****1.) Ls :-**

This command is use to list all the files which is been present is the hadoop file sys.

```
hadoopusers@depstar-VirtualBox: ~  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /  
Found 1 items  
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 08:52 /user  
hadoopusers@depstar-VirtualBox:~$
```


2.) Mkdir:-

To make new directory use this command.

```
hadoopusers@depstar-VirtualBox: ~  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -mkdir /depstar_charusat  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /  
Found 3 items  
drwxr-xr-x - hadoopusers supergroup      0 2022-09-16 08:54 /depstar  
drwxr-xr-x - hadoopusers supergroup      0 2022-09-16 08:54 /depstar_charu  
sat  
drwxr-xr-x - hadoopusers supergroup      0 2022-09-16 08:52 /user  
hadoopusers@depstar-VirtualBox:~$
```

3.) Touchz:-

```
hadoopusers@depstar-VirtualBox: ~  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -touchz /depstar/prac2.txt  
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /depstar  
Found 1 items  
-rw-r--r-- 1 hadoopusers supergroup      0 2022-09-16 09:00 /depstar/prac2  
.txt  
hadoopusers@depstar-VirtualBox:~$
```

4.) Hadoop version :-

```
hadoopusers@depstar-VirtualBox: ~  
hadoopusers@depstar-VirtualBox:~$ hadoop version  
Hadoop 3.3.1  
Source code repository https://github.com/apache/hadoop.git -r a3b9c37a397ad4188  
041dd80621bdeefc46885f2  
Compiled by ubuntu on 2021-06-15T05:13Z  
Compiled with protoc 3.7.1  
From source with checksum 88a4ddb2299aca054416d6b7f81ca55  
This command was run using /home/hadoopusers/hadoop-3.3.1/share/hadoop/common/ha  
doop-common-3.3.1.jar  
hadoopusers@depstar-VirtualBox:~$
```

5.) Hadoop find :-

```
hadoopusers@depstar-VirtualBox: ~
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -touchz /depstar/prac2.txt
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /depstar
Found 1 items
-rw-r--r--  1 hadoopusers supergroup          0 2022-09-16 09:00 /depstar/prac2
.txt
hadoopusers@depstar-VirtualBox:~$ hadoop dfs -find / prac2.txt
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

/
/depstar
/depstar/prac2.txt
/depstar_charusat
/user
/user/hadoopusers
find: `prac2.txt': No such file or directory
hadoopusers@depstar-VirtualBox:~$
```

6.) copyToLocal :-

```
hadoopusers@depstar-VirtualBox: ~
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -touchz /depstar/prac2.txt
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /depstar
Found 1 items
-rw-r--r--  1 hadoopusers supergroup          0 2022-09-16 09:00 /depstar/prac2.txt
hadoopusers@depstar-VirtualBox:~$ hadoop dfs -find / prac2.txt
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

/
/depstar
/depstar/prac2.txt
/depstar_charusat
/user
/user/hadoopusers
find: `prac2.txt': No such file or directory
hadoopusers@depstar-VirtualBox:~$ hadoop dfs -copyToLocal /depstar/prac2.txt ~/localY
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 09:00 /depstar
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 08:54 /depstar_charusat
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 08:52 /user
hadoopusers@depstar-VirtualBox:~$ ls
dfsdata  hadoop-3.3.1  hadoop-3.3.1.tar.gz  localY  tmpdata
hadoopusers@depstar-VirtualBox:~$
```

7.) rmdir :-

```
hadoopusers@depstar-VirtualBox: ~
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -rmdir depstar
rmdir: `depstar': No such file or directory
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -rmdir /depstar
rmdir: `/depstar': Directory is not empty
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -rmdir /depstar_charusat
hadoopusers@depstar-VirtualBox:~$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 09:00 /depstar
drwxr-xr-x  - hadoopusers supergroup          0 2022-09-16 08:52 /user
hadoopusers@depstar-VirtualBox:~$
```

Conclusion/Summary: In this practical, we performed various basic commands on Hadoop to create and remove orcopy files into system.

Student Signature & Date	Marks:	Evaluator Signature & Date
-------------------------------------	---------------	---------------------------------------

Practical 4

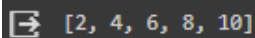
Date: 21/07/2023**Aim:** To implement Map, reduce, filter and lambda in python.**Code:**

1. Map Function:

The **map()** function is used to apply a given function to each item of an iterable (e.g., a list) and returns a new iterable with the results.

Example: Doubling each element of a list using map

```
numbers = [1, 2, 3, 4, 5]
doubled = list(map(lambda x: x * 2, numbers))
print(doubled)
```

[2, 4, 6, 8, 10]

2. Reduce Function:

The **reduce()** function is used to apply a given function cumulatively to the items of an iterable, reducing it to a single accumulated result.

Example: Summing up all elements of a list using reduce

```
from functools import reduce
numbers = [1, 2, 3, 4, 5]
total = reduce(lambda x, y: x + y, numbers)
print(total)
```

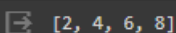
15

3. Filter Function:

The **filter()** function is used to filter elements of an iterable based on a given function's condition.

Example: Filtering even numbers from a list using filter

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
evens = list(filter(lambda x: x % 2 == 0, numbers))
print(evens) # Output: [2, 4, 6, 8]
```

[2, 4, 6, 8]

4. Lambda Function:

Lambda functions are anonymous functions defined using the lambda keyword. They are often used for short, simple operations.

Example: Using a lambda function to square a number

```
square = lambda x: x ** 2
```

```
result = square(5)
```

```
print(result) # Output: 25
```

A small terminal window showing the output of the print statement, which is the number 25.

Conclusion/Summary: In this practical we learnt about some basic example of how to use map, reduce, filter, and lambda functions in Python.

Student Signature & Date	Marks:	Evaluator Signature & Date
-------------------------------------	---------------	---------------------------------------

Practical 5**Date: 22/07/2023****Aim:** To implement a word count application using the MapReduce programming model.**Code:****# Create a sample text file**

```
sample_text = ""
```

```
Hello world
```

```
This is a sample text file
```

```
For word count example
```

```
Hello again
```

```
""
```

```
with open("new.txt", "w") as file:
```

```
    file.write(sample_text)
```

Mapper function

```
def mapper(line):
```

```
    words = line.strip().split()
```

```
    for word in words:
```

```
        print(f"{word}\t1")
```

Reducer function

```
def reducer(word, counts):
```

```
    total = sum(map(int, counts))
```

```
    print(f"{word}\t{total}")
```

Use Hadoop Streaming to perform word count

```
!cat "new.txt" | python -m mapreduce.pipes \
```

```
    --map "/usr/bin/python mapper" \
```

```
    --reduce "/usr/bin/python reducer" \
```

```
--input-format org.apache.hadoop.mapred.TextInputFormat \
--output-format org.apache.hadoop.mapred.TextOutputFormat \
--input /content/drive/My\ Drive/sample.txt \
--output /content/drive/My\ Drive/wordcount_output
```

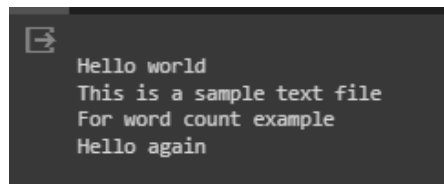
View the word count output

with open("new.txt") as result_file:

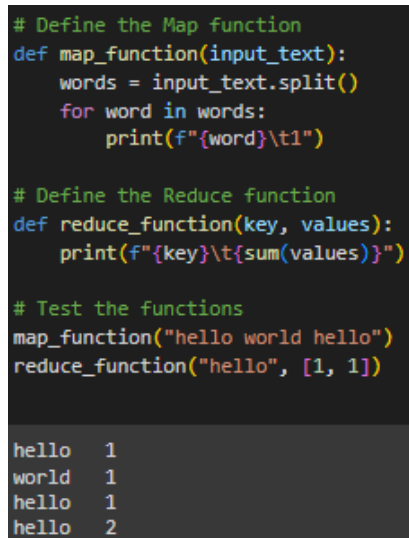
```
wordcount_result = result_file.read()
```

```
print(wordcount_result)
```

Output Screenshot:



```
Hello world
This is a sample text file
For word count example
Hello again
```



```
# Define the Map function
def map_function(input_text):
    words = input_text.split()
    for word in words:
        print(f"{word}\t1")

# Define the Reduce function
def reduce_function(key, values):
    print(f"{key}\t{sum(values)}")

# Test the functions
map_function("hello world hello")
reduce_function("hello", [1, 1])

hello 1
world 1
hello 1
hello 2
```

Conclusion/Summary:

In this practical we implemented a word count application using the MapReduce programming model.

Student Signature & Date	Marks:	Evaluator Signature & Date

Practical 6**Date: 28/07/2023****Aim:**

To design and implement MapReduce algorithms to take a very large file of integers and produce as output:

- a) The largest integer
- b) The average of all the integers.
- c) The count of the number of distinct integers in the input.

Code:

```
//Reducer.java

import java.io.IOException;
import
org.apache.hadoop.conf.Configuration;import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;public
class Reducer
{
    public static void main(String args[]) throws IllegalStateException,
IOException,ClassNotFoundException, InterruptedException
    { Configuration conf = new Configuration();
      Job job = Job.getInstance(conf,"Practicle 4");job.setJarByClass(Reducer.class);
      job.setMapperClass(BDA_Mapper.class);
      job.setReducerClass(BDA_Reducer.class);job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(LongWritable.class);
      job.setOutputKeyClass(LongWritable.class);job.setOutputValueClass(Text.class);
      FileInputFormat.addInputPath(job,new Path(args[0]));
      FileOutputFormat.setOutputPath(job,new Path(args[1]));
      System.exit(job.waitForCompletion(true)?0:1);    }    }

//BDA_Mapper.java

import
java.util.*;import
java.io.*;
import
org.apache.hadoop.io.Text;import
```



```

org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;
public class BDA_Mapper extends Mapper<LongWritable,Text,Text,LongWritable>
{
    private TreeMap<String,Integer>
    tmap;@Override
    public void setup(Context context) throws IOException,InterruptedException
    { tmap = new TreeMap<String,Integer>();}
    public void map(LongWritable key,Text value,Context context)
throwsIOException,InterruptedException
    {if(tmap.containsKey(value.toString().trim()))
    {int
        count=tmap.get(value.toString().trim());tmap.put(value.toString().trim(),count+
        1);}else    { tmap.put(value.toString().trim(),1);} }
    @Override
    public void cleanup(Context context) throws IOException,InterruptedException
    {
        for(Map.Entry<String,Integer> entry:tmap.entrySet())
        {String number = entry.getKey();int count = entry.getValue();
        context.write(new Text(number),new LongWritable(count));} }

//BDA_Reducer.jav
aimport java.util.*;
import java.io.*;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import
org.apache.hadoop.mapreduce.Mapper;import
org.apache.hadoop.mapreduce.Reducer;
public class BDA_Reducer extends Reducer<Text,LongWritable,Text,LongWritable>
{private TreeMap<String , Long> tmap2;
    private int max =
    Integer.MIN_VALUE,unique=0,cnt=0;private long
    sum=0;
    @Override
    public void setup(Context context) throws IOException,InterruptedException
    { tmap2 = new
    TreeMap<String,Long>();} @Override
    public void reduce(Text key,Iterable<LongWritable> values,Context context)
throwsIOException,InterruptedException
    {String number = key.toString();long
        count=0;for(LongWritable val:values)
        { count+=val.get();sum+=((int)val.get())*Integer.parseInt(number.trim());}
        tmap2.put(number,count);cnt+=count;

```

```

        if(max<Integer.parseInt(number.trim()))
            max=Integer.parseInt(number.trim());unique++;}

```

@Override

```

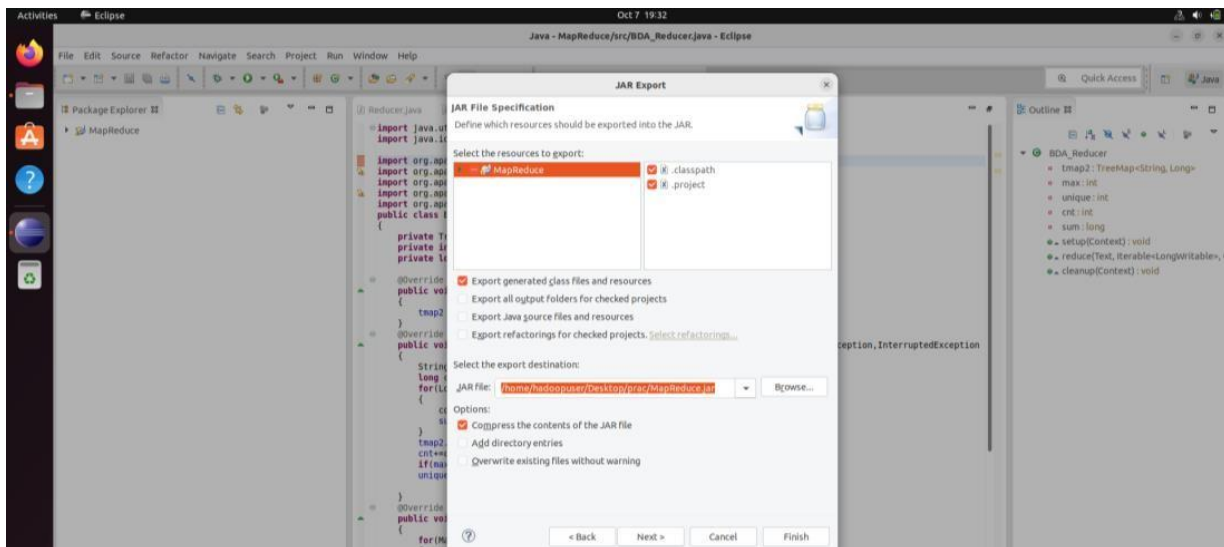
public void cleanup(Context context) throws IOException,InterruptedException
{
    for(Map.Entry<String,Long> entry:tmap2.entrySet())
    { Long count = entry.getValue();String name = entry.getKey();
      context.write(new Text(name),new LongWritable(count));}
    context.write(new Text("MAX NUMBER = "),new
      LongWritable(max));context.write(new Text("AVERAGE = "),new
      LongWritable(sum/cnt));

    Content.write write(new Text("Total Unique Numbers =
"),newLongWritable(unique));} }

```

OUTPUT:

- ☐ MyProject -> then select Build Path-> Click on Configure Build Path and select AddExternal jars.... and add jars from it's download location then click -> Apply and Close.
- ☐ Now export the project as jar file. Right-click on MyProject choose Export.. and go toJava -> JAR file click -> Next and choose your export destination then click -> Next. choose Main Class by clicking -> Browse and then click -> Finish -> Ok.
- ☐ In Eclipse go to export



```

hduser@localhost: ~
hadoopuser@depstar-virtualbox: ~/hadoop-single-node-cluster$ sudo docker exec -it 909bf0d58fda bash
hduser@localhost:~$ ls
CR00013-2020-AK-Pokerbanks 11_NE.txt demo.jar examples hadoop-3.3.3 MapReduce.jar prac.txt
hduser@localhost:~$ hdfs dfs -ls /
Found 3 items
drwxrwxrwx - hduser supergroup 0 2022-10-07 13:45 /jars
drwxrwxrwx - hduser supergroup 0 2022-10-07 13:44 /tmp
hduser@localhost:~$ hdfs dfs -mkdir prac
^C
hduser@localhost:~$ hdfs dfs -mkdir /prac
hduser@localhost:~$ hdfs dfs -put MapReduce.jar /prac
hduser@localhost:~$ hdfs dfs -ls /prac
Found 1 items
-rw-r--r-- 1 hduser supergroup 5854 2022-10-07 13:50 /prac/MapReduce.jar
hduser@localhost:~$ hdfs dfs -put prac.txt /prac
hduser@localhost:~$ cat prac.txt
1
2
2
1
2
2
3
hduser@localhost:~$ hadoop jar MapReduce.jar /prac/prac.txt /MyOpt
2022-10-07 13:51:44,349 INFO Impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-10-07 13:51:44,461 INFO Impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-10-07 13:51:44,463 INFO Impl.MetricsSystemImpl: JobTracker metrics setup started.

CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=12
File Output Format Counters
    Bytes Written=67
hduser@localhost:~$ hdfs dfs -cat /prac/MyOpt
cat: '/prac/MyOpt': No such file or directory
hduser@localhost:~$ hdfs dfs -ls /prac
Found 2 items
-rw-r--r-- 1 hduser supergroup 5854 2022-10-07 13:50 /prac/MapReduce.jar
-rw-r--r-- 1 hduser supergroup 12 2022-10-07 13:51 /prac/prac.txt
hduser@localhost:~$ hdfs dfs -ls /
Found 5 items
drwxr-xr-x - hduser supergroup 0 2022-10-07 13:52 /MyOpt
drwxrwxrwx - hduser supergroup 0 2022-10-07 13:45 /jars
drwxr-xr-x - hduser supergroup 0 2022-10-07 13:51 /prac
drwxrwxrwx - hduser supergroup 0 2022-10-07 13:44 /tmp
drwxrwxrwx - hduser supergroup 0 2022-10-07 13:44 /users
hduser@localhost:~$ hdfs dfs -cat /MyOpt
cat: /MyOpt: is a directory
hduser@localhost:~$ hdfs dfs -ls /MyOpt
Found 2 items
-rw-r--r-- 1 hduser supergroup 0 2022-10-07 13:52 /MyOpt/_SUCCESS
-rw-r--r-- 1 hduser supergroup 67 2022-10-07 13:52 /MyOpt/part-r-00000
hduser@localhost:~$ hdfs dfs -cat /MyOpt/part-r-00000
1
2
3
1
3
MAX NUMBER = 3
AVERAGE = 1
Total Unique Numbers = 3
hduser@localhost:~$

```

Conclusion/Summary:

In this practical, we learnt about the mapReduce Paradigm in detail and also executed asimple operations on large integers file in MapReduce in Python Language.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 7**Date: 04/08/2023**

Aim: To implement basic functions and commands in R Programming. Use R-Studio and build WordCloud and data visualization using R for easy to understand and better visualization than a data table.

Code:

```
install.packages("tm")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("RColorBrewer")
library("wordcloud")
library("RColorBrewer")
# To choose the text file text =
readLines(file.choose())# VectorSource() function #
creates a corpus of
# character v ectors docs =
Corpus(VectorSource(text))# Text transformation
toSpace = content_transformer(
  function (x, pattern)
    gsub(pattern, " ", x))
docs1 = tm_map(docs, toSpace, "/") docs1 =
  tm_map(docs,toSpace, "@") docs1 =
  tm_map(docs, toSpace, "#")
strwrap(docs1)
# Cleaning the Text docs1 = tm_map(docs1, content_transformer(tolower))
docs1 = tm_map(docs1, removeNumbers) docs1 = tm_map(docs1,
stripWhitespace)# Build a term-document matrix dtm =
TermDocumentMatrix(docs) m = as.matrix(dtm) v
=sort(rowSums(m), decreasing = TRUE)
d = data.frame(word = names(v), freq =
```

```
v)head(d, 10)
```

```
# Generate the Word cloud wordcloud(words =
```

```
d$word, freq = d$freq, min.freq = 1, max.words = 200, random.order = FALSE, rot.per  
=0.35, colors = brewer.pal(8, "Dark2"))
```

Output Screenshot:



Conclusion/Summary:

In this practical, we learnt about R and implemented wordCloud using R.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 8

Date: 11/08/2023

Aim: To implement supervised learning algorithms (linear regression and logistic regression) using R.

Code:

- Install Required Packages:

To install R packages, use the following code cell in your Colab notebook:

Install and load necessary libraries

```
install.packages("tidyverse")
```

```
library(tidyverse)
```

```
install.packages("glmnet")
```

```
library(glmnet)
```

- Load and Preprocess Data

You'll need a dataset for your regression tasks. You can upload a dataset directly to Google Colab or load it from an online source. Here, I'll provide an example using a built-in R dataset.

Load a built-in dataset (example using mtcars dataset)

```
data(mtcars)
```

- Linear Regression Example:

Here's an example of implementing linear regression in R:

Linear regression

```
model_lm <- lm(mpg ~ cyl + hp, data = mtcars)
```

Summary of the linear regression model

```
summary(model_lm)
```

- Logistic Regression Example:

Here's an example of implementing logistic regression in R:

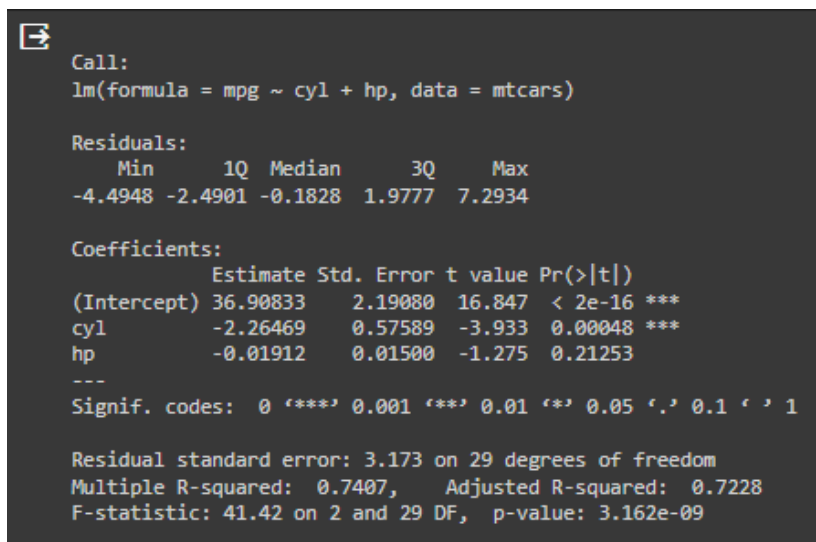
Logistic regression

```
model_logistic <- glm(vs ~ cyl + hp, data = mtcars, family = binomial)
```

Summary of the logistic regression model**summary(model_logistic)**

- Evaluate and Visualize Results

After fitting the models, you can evaluate them using appropriate metrics and visualize the results.

Evaluate linear regression model**linear_predictions <- predict(model_lm, newdata = mtcars)****linear_mse <- mean((mtcars\$mpg - linear_predictions)^2)****linear_mse****# Evaluate logistic regression model****logistic_predictions <- predict(model_logistic, type = "response", newdata = mtcars)****logistic_predictions_binary <- ifelse(logistic_predictions > 0.5, 1, 0)****logistic_accuracy <- mean(logistic_predictions_binary == mtcars\$vs)****logistic_accuracy****Output Screenshot:**

```
Call:
lm(formula = mpg ~ cyl + hp, data = mtcars)

Residuals:
    Min       1Q   Median       3Q      Max
-4.4948 -2.4901 -0.1828  1.9777  7.2934

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  36.90833    2.19080   16.847  < 2e-16 ***
cyl          -2.26469    0.57589   -3.933  0.00048 ***
hp           -0.01912    0.01500   -1.275  0.21253
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.173 on 29 degrees of freedom
Multiple R-squared:  0.7407,    Adjusted R-squared:  0.7228
F-statistic: 41.42 on 2 and 29 DF,  p-value: 3.162e-09
```



```
Call:
glm(formula = vs ~ cyl + hp, family = binomial, data = mtcars)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  9.08901    3.29417   2.759  0.0058 **
cyl         -0.68950    0.78939  -0.873  0.3824
hp          -0.04135    0.03630  -1.139  0.2546
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.860  on 31  degrees of freedom
Residual deviance: 16.024  on 29  degrees of freedom
AIC: 22.024

Number of Fisher Scoring iterations: 7
```



```
9.12420455156076
0.84375
```

Conclusion/Summary:

In this practical we implemented supervised learning algorithms (linear regression and logistic regression) using R.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 9

Date: 18/08/2023**Aim:** To implement unsupervised learning algorithms (k-means) using R.**Code:**

1. Install Required Packages:

To install R packages, use the following code cell in your Colab notebook:

```
install.packages("ggplot2")
```

- 2.

Generate or Load Data:

You can generate synthetic data or upload your own dataset. Here's an example of generating random data:

```
set.seed(123) # For reproducibility
```

```
data <- data.frame(
```

```
  x = rnorm(100, mean = 0, sd = 1),
```

```
  y = rnorm(100, mean = 0, sd = 1)
```

```
)
```

- 3.

Perform K-Means Clustering:

Use the k-means clustering algorithm as shown in the previous response:

```
k <- 3 # Number of clusters
```

```
set.seed(123) # For reproducibility
```

```
kmeans_result <- kmeans(data, centers = k)
```

- 4.

View the Cluster Assignments:

Check the cluster assignments for each data point:

```
cluster_assignments <- kmeans_result$cluster
```

- 5.

Visualize the Clusters:

Create a scatter plot to visualize the clusters:

```
library(ggplot2)
```

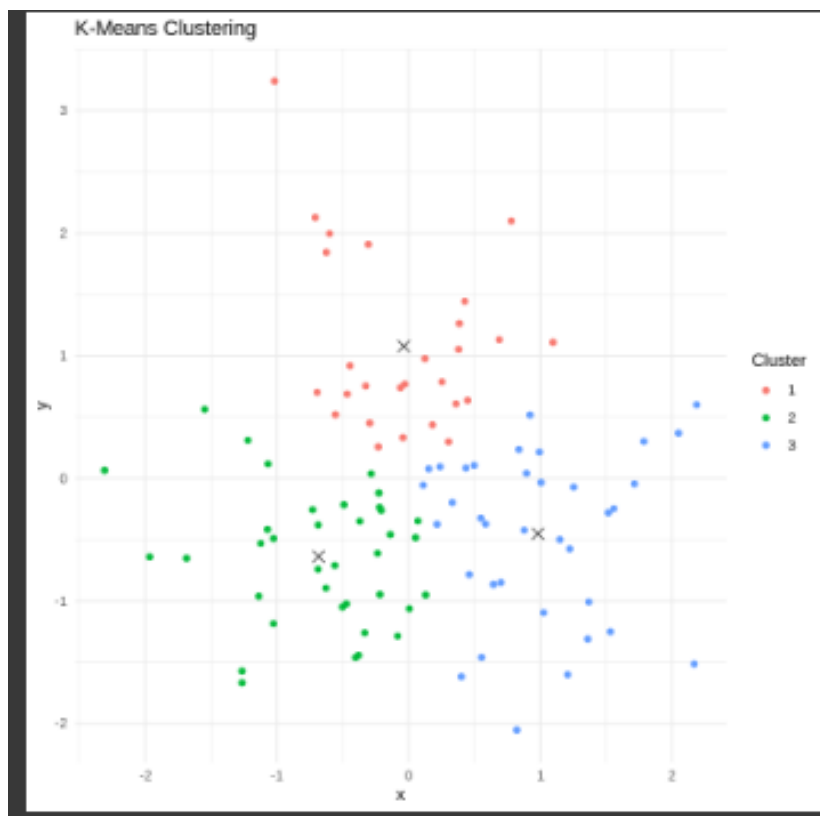
```
ggplot(data, aes(x = x, y = y, color = factor(cluster_assignments))) +
```

```
  geom_point() +
```

```
  geom_point(data = as.data.frame(kmeans_result$centers), aes(x = x, y = y), color =  
    "black", size = 3, shape = 4) +
```

```
labs(title = "K-Means Clustering", color = "Cluster") +  
theme_minimal()
```

Output Screenshot:



Conclusion/Summary:

In this practical we implemented unsupervised learning algorithms (k-means) using R.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 10

Date: 25/08/2023

Aim: To install and implement basic database operations in MongoDB. To implement basic CRUD operations (create, read, update, delete) in MongoDB.

Code:

THEORY:

MongoDB:

- MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).
- Main Features:
 - Ad-hoc queries
 - MongoDB supports field, range query, and regular-expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.
 - Indexing
 - Fields in a MongoDB document can be indexed with primary and secondary indices or index.
 - Replication
 - MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default.
 - If the replicated MongoDB deployment only has a single secondary member, a separate daemon called an arbiter must be added to the set. It has a single responsibility, which is to resolve the election of the new primary.[30] As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.
 - Load balancing
 - MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed

- across multiple shards. (A shard is a master with one or more replicas.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.
- MongoDB can run over multiple servers, balancing the load or duplicating data to keep the system up and running in case of hardware failure.
 - File storage
 - MongoDB can be used as a file system, called GridFS, with load balancing and data replication features over multiple machines for storing files.
 - This function, called grid file system, is included with MongoDB drivers. MongoDB exposes functions for file manipulation and content to developers. GridFS can be accessed using mongofiles utility or plugins for Nginx and lighttpd. GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.
 - Aggregation
 - MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single-purpose aggregation methods.
 - Map-reduce can be used for batch processing of data and aggregation operations. But according to MongoDB's documentation, the Aggregation Pipeline provides better performance for most aggregation operations.
 - The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to Unix pipes. The aggregation framework includes the \$lookup operator which can join documents from multiple collections, as well as statistical operators such as standard deviation.
 - Server-side JavaScript execution
 - JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.
 - Capped collections
 - MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.
 - Transactions
 - MongoDB claims to support multi-document ACID transactions since the 4.0 release in June 2018. This claim was found to not be true as MongoDB violates snapshot isolation.

PRACTICAL:

We can download MongoDB with the help of following tutorials.

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

MongoDB:

We can start the MongoDB service using “mongo” command.

```
C:\Users\jilsa>mongo
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("58b4f2eb-79a9-4ab5-9bef-7d5fd2c1e208") }
MongoDB server version: 5.0.2
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2021-08-27T14:17:05.074+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

We can check available databases using “show dbs” command.

```
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
```

We can create database using “use” keyword easily.

```
> use shopping
switched to db shopping
```

We can insert the data in database using “insertOne” command.

```
> db.products.insertOne({name:'product1', price:40, color:'pink'});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6131e482ae9c55a33bbd67be")
}
```

We can check all the records of database using “find” and “pretty()” to format the output.

```

> db.products.find();
{ "_id" : ObjectId("6131e482ae9c55a33bdd67be"), "name" : "product1", "price" : 40, "color" : "pink" }
{ "_id" : ObjectId("6131e496ae9c55a33bdd67bf"), "name" : "product2", "price" : 60, "color" : "blue" }
> db.products.insertMany([ {name:'product3', price:20, color:'green'}, {name:'product4', price: 15, color:'yellow'}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("6131e4e1ae9c55a33bdd67c0"),
    ObjectId("6131e4e1ae9c55a33bdd67c1")
  ]
}
> db.products.find().pretty();
{
  "_id" : ObjectId("6131e482ae9c55a33bdd67be"),
  "name" : "product1",
  "price" : 40,
  "color" : "pink"
}
{
  "_id" : ObjectId("6131e496ae9c55a33bdd67bf"),
  "name" : "product2",
  "price" : 60,
  "color" : "blue"
}
{
  "_id" : ObjectId("6131e4e1ae9c55a33bdd67c0"),
  "name" : "product3",
  "price" : 20,
  "color" : "green"
}
{
  "_id" : ObjectId("6131e4e1ae9c55a33bdd67c1"),
  "name" : "product4",
  "price" : 15,
  "color" : "yellow"
}

```

We can also update our record using “updateOne”.

```

> db.products.updateOne({name:'product2'}, {$set: {price:50, qty:10}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>

```

Conclusion/Summary:

In this practical, we learnt about mongoDB and Cassandra and performed basic CRUD operations in both the databases.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 11

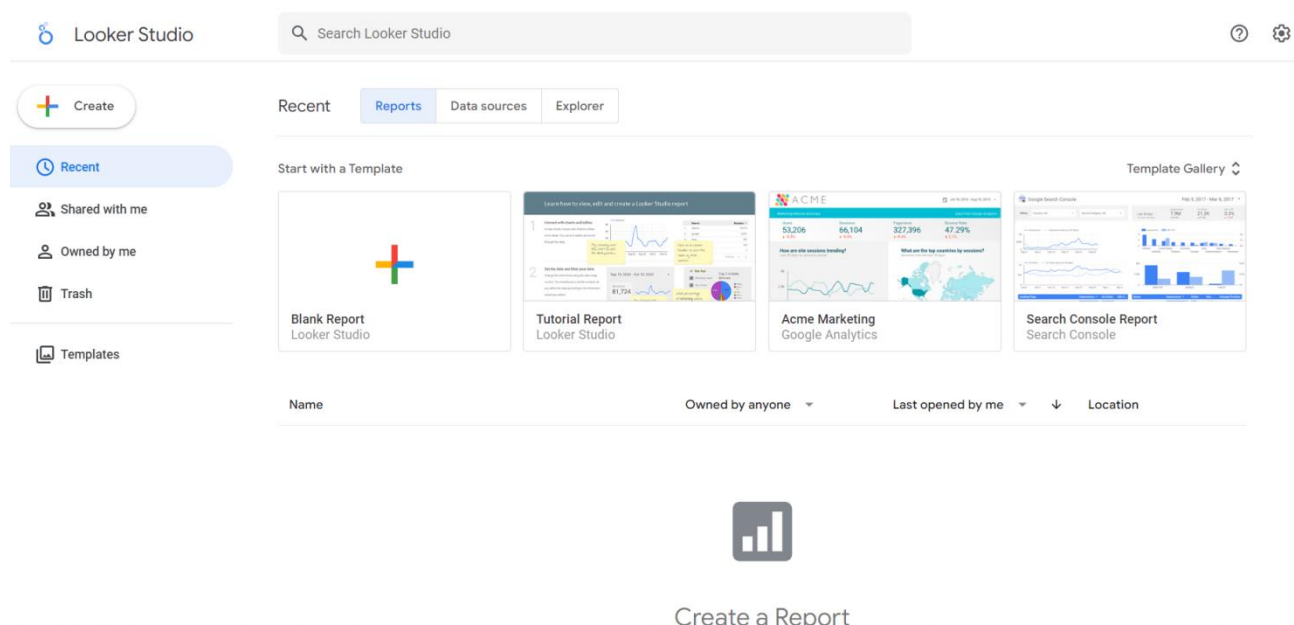
Date: 01/09/2023**Aim:** To design a dashboard using Google data studio.**Code:**

Step 1: Login & Create A Blank Report

The first step is to visit datastudio.google.com and sign in with the account that is the same account as the Google Analytics & Google Search Console you want to create the dashboard for.

Sign in to Google Data Studio with your Google Analytics account

Once you have signed in and accepted any Terms of Service notices, we are now ready to create our blank report. Simply click on “+” for the “Blank” report template. There are other pre-built templates you can check out, however, for this tutorial, we are going to create a new blank report.



Create a blank dashboard

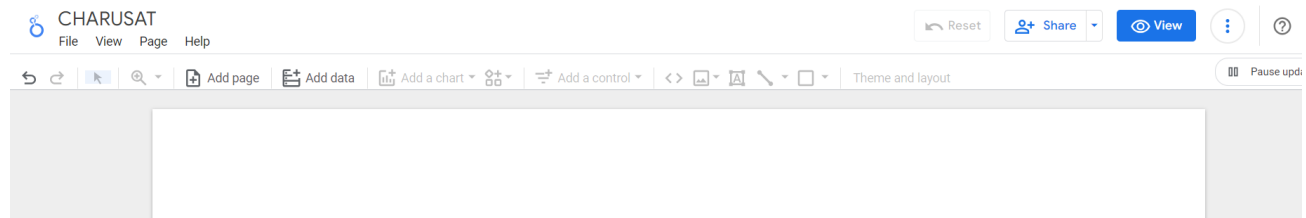
Finally, let's rename this dashboard so it is easy to find. This is an important step. Google will auto-save your dashboard as you go, however, if you leave the name as the default, it will be difficult to find later.

Rename your dashboard

Step 2: Add Your First Data Source

For this dashboard, we will need two data sources. We will need to connect to Google Analytics & Google Search Console.

First, click the “Create New Data Source” button.

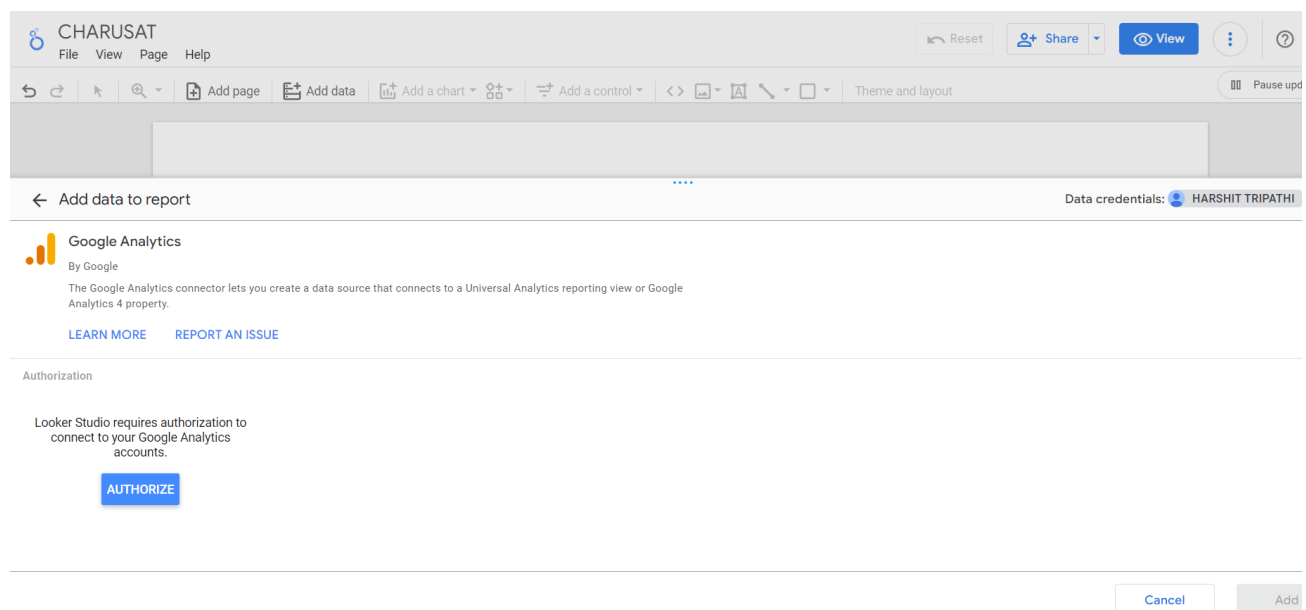


Click Create New Data Source

Next, find the Google Analytics connector and select Google Analytics. You will notice there are many other connectors you can choose from. These are all the different data sources that can be pulled into a report. For now, let's stick to our current dashboard.

Find and select the Google Analytics connector

We now need to authorize our connection to Google Analytics. You will need to select “Allow Access” on the authorization popup that follows.



Authorize the Google Analytics connector

Our next step is to rename this connector to make it easier to find. In this example, “Demo Google Analytics” is being used.

Now we must choose our Google Analytics account, then the property followed by the view. This should be your main view that you use when viewing your data in Google Analytics.

To finish this step, click the connect button to connect to this data source.

Rename, select account/property/view, and connect

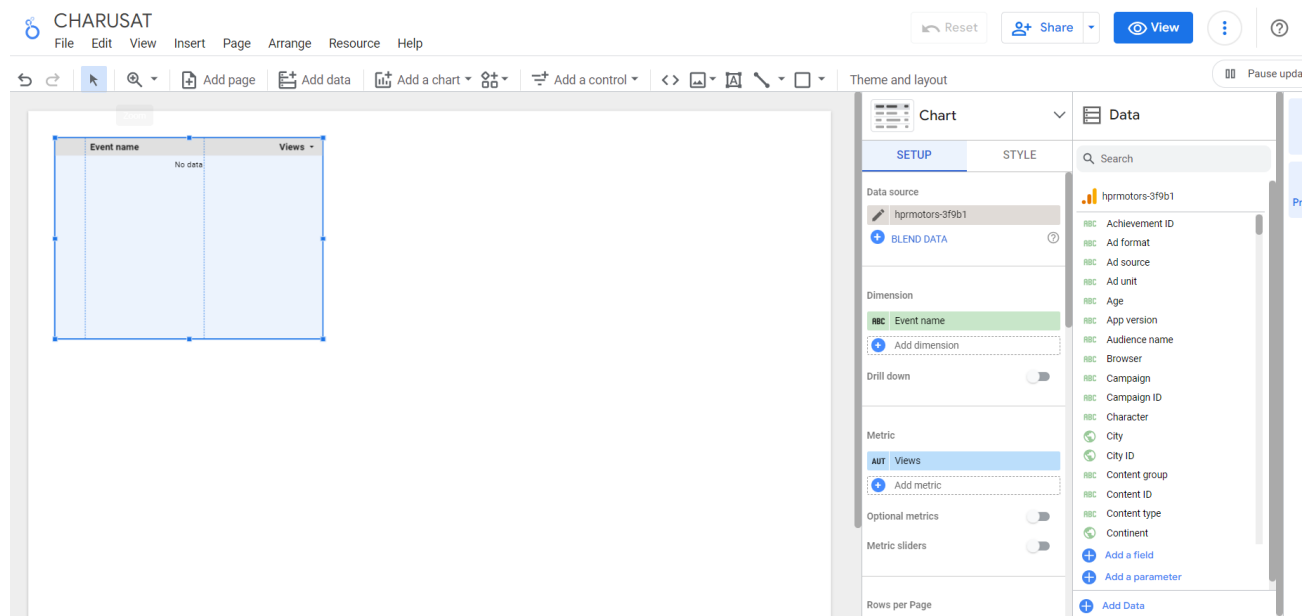
The final stage to adding the data source is to click “Add to Report”. At this state, you are able to modify fields and add calculated fields if needed. This is more advanced and will be covered in a later article. For now, all the fields are setup perfectly fine for our dashboard.

Account	Property
Default Account for Firebase	hpmotors-3f9b1 GA4 335459201
	netflix-clone-2120e GA4 334764844

Click “Add to Report”

Step 3: Add Your Second Data Source

Now that we have already added one data source, the screen has changed and we no longer have the “Create Data Source” button in the lower right of the screen. To add a second data source, we will need to go to “Resource” and select “Manage added data sources”.

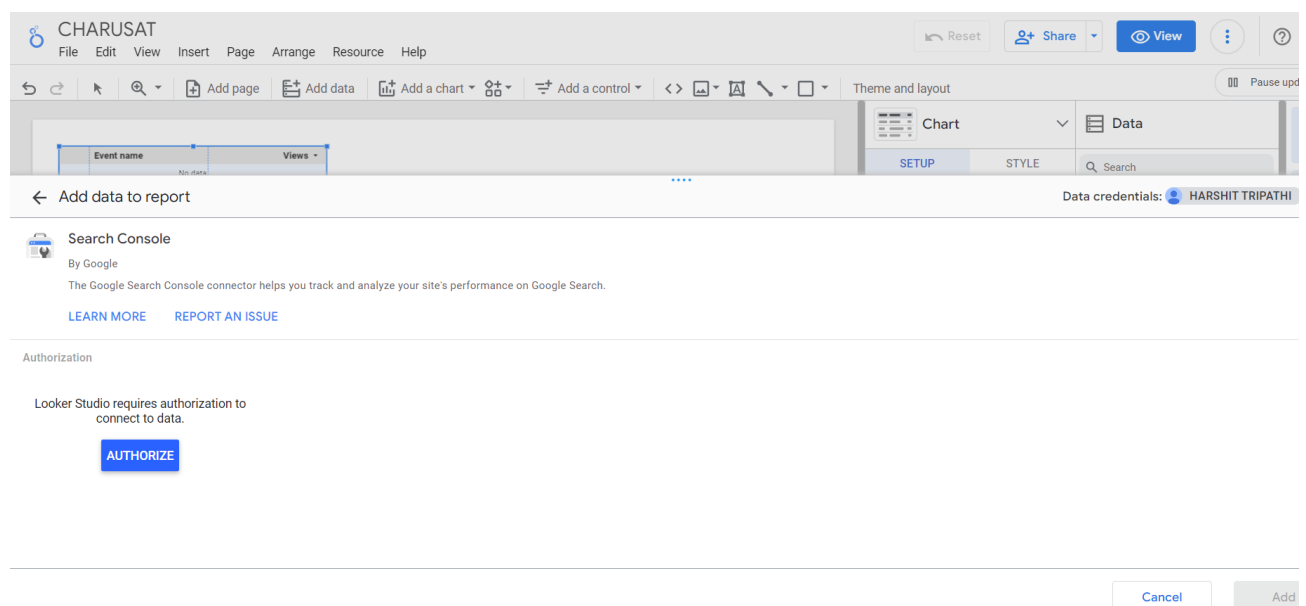


Under “Resource” select “Manage added data sources”

This is the panel where you can manage your data sources for each report/dashboard. You will see our first data source we added. To add a second, just click on “Add A Data Source”.

Click “Add A Data Source”

This will take you back to the connector gallery. Simply find the “Search Console” connector and select it. You may need to authorize once again.



Select “Search Console”

As we did before, rename this data source to something you can identify easily later. (See #1)

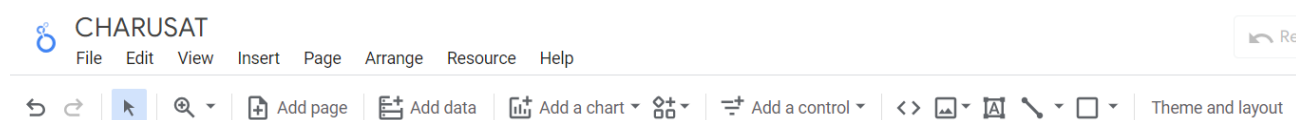
Next, select your site and for this dashboard we are going to focus on “Site Impression” table.

Finally, click the connect button to connect this to your dashboard.

Rename, select site and table, connect to your dashboard

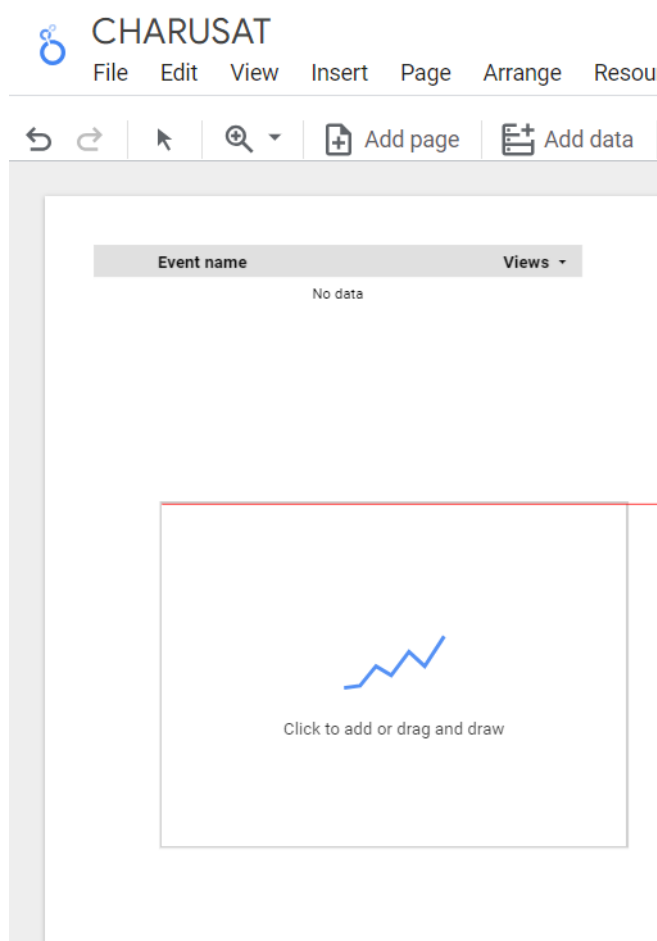
Step 4: Adding Elements

Now that we have setup all our data sources, it’s time to build out our dashboard. Google Data Studio comes with several types of elements that can be added to your report. These elements include line graphs, pie charts, tables, geo maps, scorecards and many more.



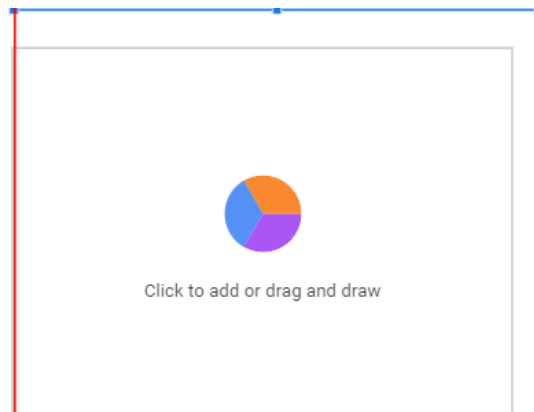
Elements you can add to your dashboard

Line Graphs / Time Series



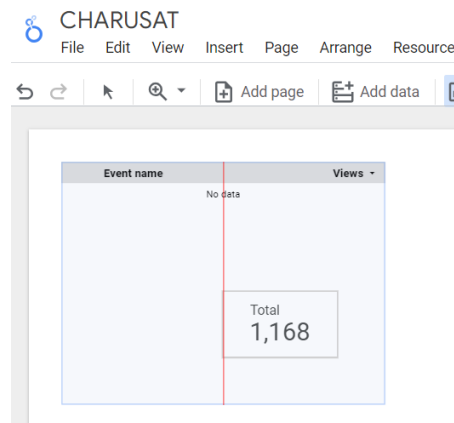
Update your data source, dimension and metrics

Pie Charts



Stylize your element

Scorecards



Tables

	Source	Sessions
1.	google	36,864
2.	(direct)	15,662
3.	youtube.com	14,190
4.	mall.googleplex.com	7,587
5.	analytics.google.com	2,722
6.	Partners	2,537
7.	sites.google.com	2,396
8.	gdeals.googleplex.com	1,923
9.	moma.corp.google.com	1,143
10.	groups.google.com	480

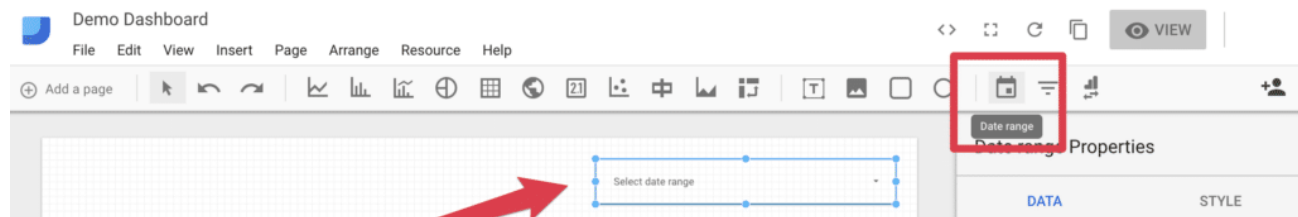
1 - 100 / 110 < >

Add a second table & setup dimensions and metrics

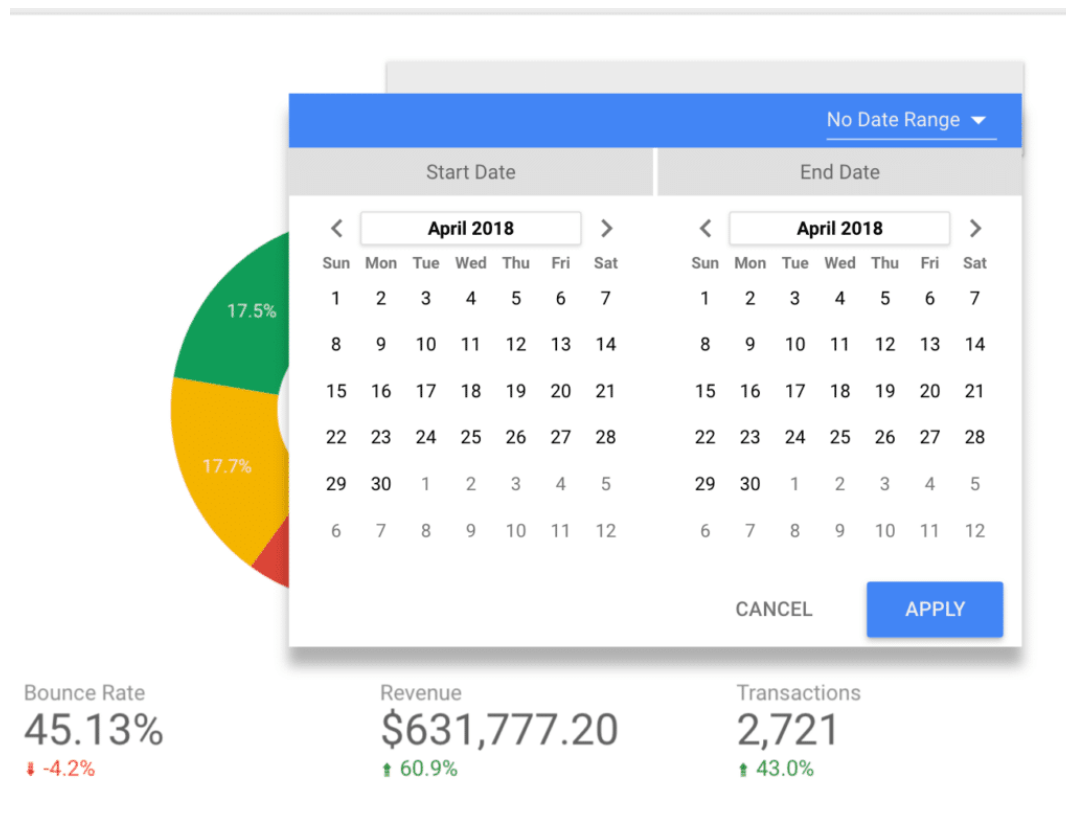
Step 5: Adding Date Range Filter

Currently, all the elements added to our dashboard are set to pull “Auto” as our date. By default this is pulling the last 28 days, but, what if you wanted to view a different date range?

Google Data Studio as an element known as a “Date Range” filter. This is added the same way as all other elements. Just select it from the menu bar and drag out a date range filter at an appropriate size.



Add date range filter



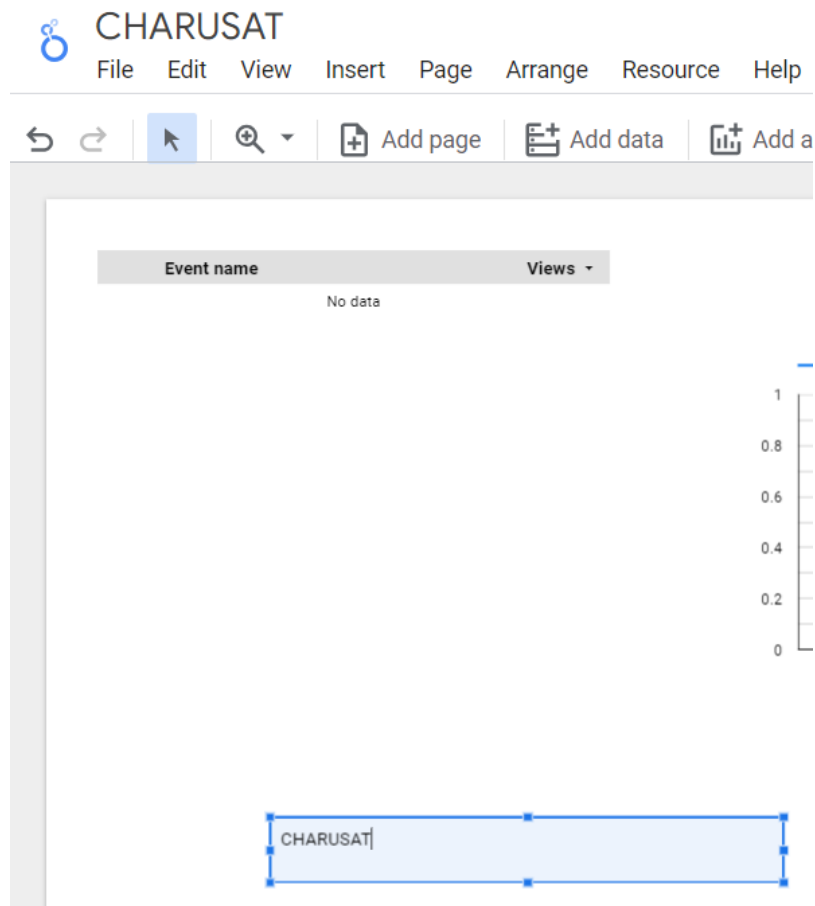
View date range filter

Step 6: Add Text Headers & Images

Once elements have been added, you may have noticed that there are no headings. This can be confusing to anyone using your dashboard. Also, we may want to add some branding or images to help others make better sense of our newly created dashboard.

Text Headers

Google Data Studio makes adding text headers very simple with their “Text” element. Simply drag out a text box and add your text. Under text properties, update the size, color, and font-weight to achieve your desired look.

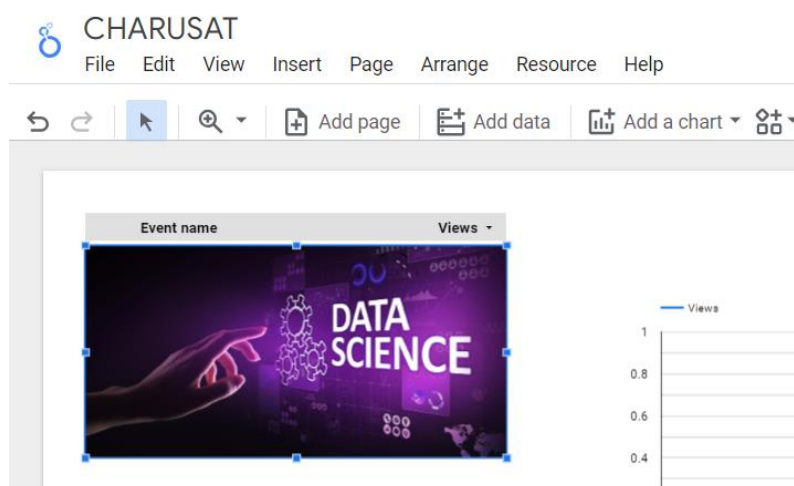


Add text

Adding Images

There are several reasons as to why you may want to add images to your dashboard. The most common is to add your company's branding.

Select the image element from the menu bar, drag a box onto your dashboard then click on “select a file” button on your “Data” tab. Browse your computer and find the image file you want to add.



Add image

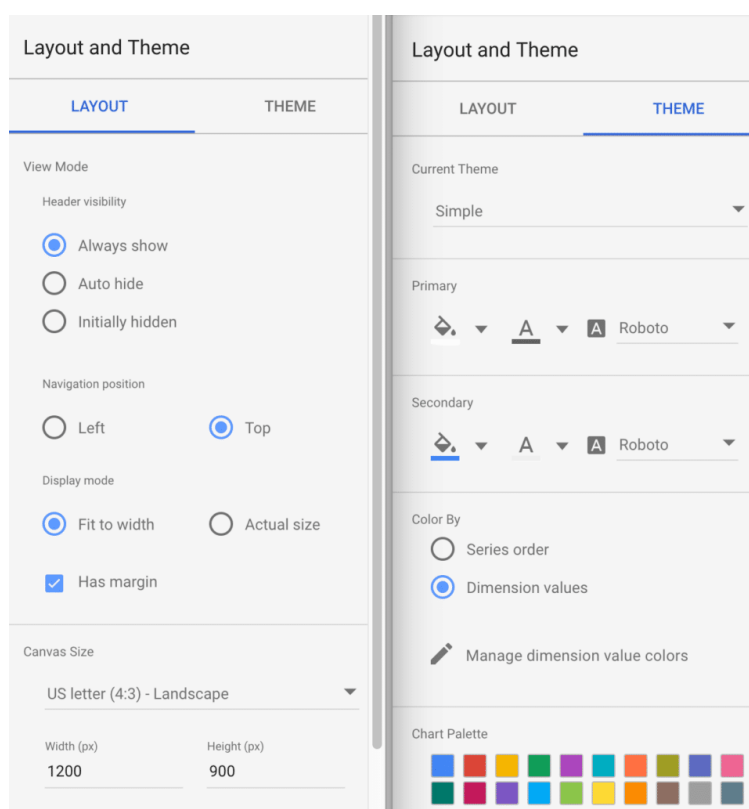
Our added image

Step 7: Styling

Our dashboard has all the data we want to see, however, it doesn't look amazing. With Google Data Studio, it is very simple to make style edits.

Global Styles

When you don't have any elements selected, you should see the Layout and Theme sidebar. In this sidebar, we have the ability to control the overall layout and theme for our entire dashboard and all elements.






Layout & theme tabs

You can control your Primary and Secondary text color and font under the theme tab.

In addition, you can also pick your chart palette. Chart palette is the order of colors that will be used in all charts. For example, by default we have blue, red, yellow, green, etc. Notice in our pie chart on our dashboard, we have these same colors used in descending order. If you have brand colors, I would suggest changing these colors to your brands colors. Make sure the colors are different enough that you can make out which color represents different data.

Primary

 ▼  A ▼  A Roboto ▼

Secondary











 ▼  A ▼  A Roboto ▼




Chart Palette



 ▼  ▼  ▼

Background and Border

 ▼  0 ▼  100% ▼

 ▼  None ▼  Solid ▼

Canvas Size

US letter (4:3) - Landscape ▼

Width (px)

1200

Height (px)

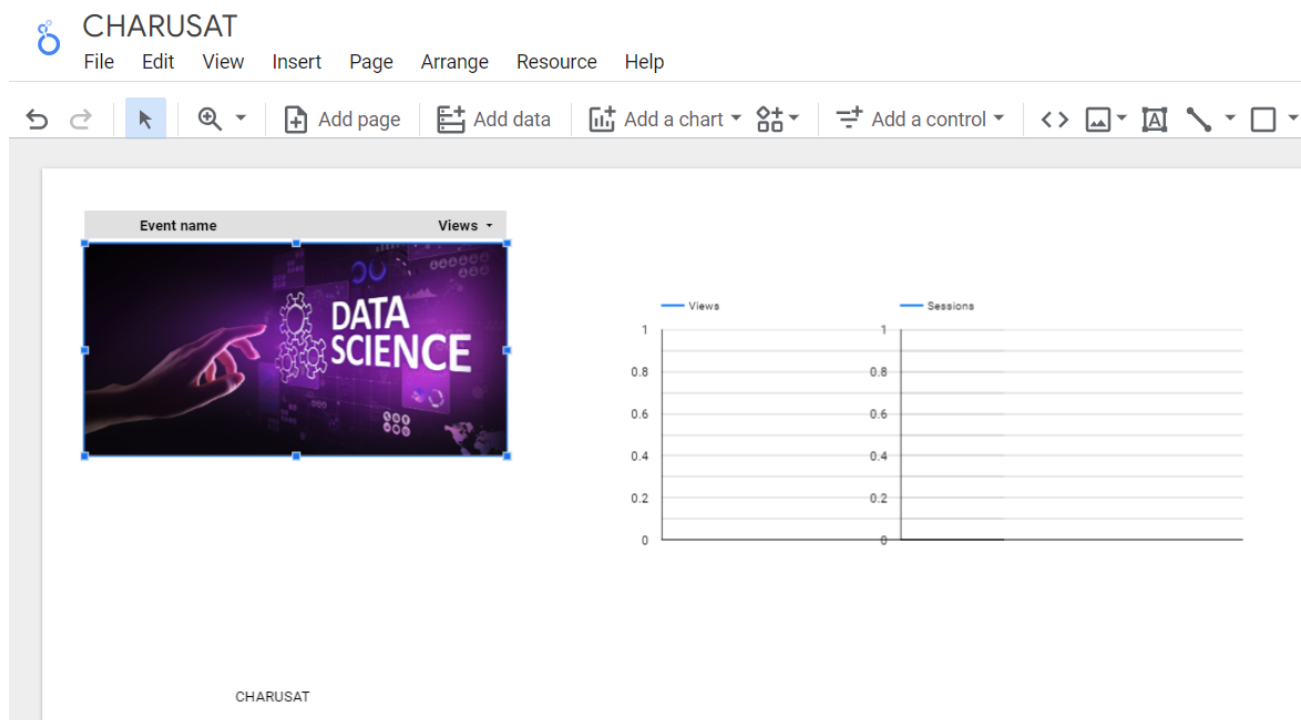
900

Element Styles

Each element in Google Data Studio has its own style that can be adjusted. It will automatically pull from the global styles, however, you can fine tune and even override the global styles on each element.

Most elements also contain additional elements that you can style. For example, a table has the ability to set the colors for alternating rows and to adjust headers. Pie charts allow you to change where the legend is positioned and alignment. Each element has its own styles. Take some time to explore each element.

Output Screenshot:



Conclusion/Summary:

In this practical we learnt to design a dashboard using Google data studio.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 12

Date: 15/09/2023

Aim: To install, deploy & configure Apache Spark. To Select the fields from the dataset using Spark SQL.

Code:

THEORY:

Spark:

- Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.
- Apache Spark has its architectural foundation in the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.
- The Dataframe API was released as an abstraction on top of the RDD, followed by the Dataset API. In Spark 1.x, the RDD was the primary application programming interface (API), but as of Spark 2.x use of the Dataset API is encouraged even though the RDD API is not deprecated. The RDD technology still underlies the Dataset API.
- Spark and its RDDs were developed in 2012 in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk.
- Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.
- Inside Apache Spark the workflow is managed as a directed acyclic graph (DAG). Nodes represent RDDs while edges represent the operations on the RDDs.
- Spark facilitates the implementation of both iterative algorithms, which visit their data set multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data.
- The latency of such applications may be reduced by several orders of magnitude compared to Apache Hadoop MapReduce implementation. Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.
- Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster, where you can launch a cluster either manually or use the launch scripts provided by the install package. It is also possible to run these daemons on a single machine for testing), Hadoop YARN, Apache Mesos or Kubernetes.
- For distributed storage, Spark can interface with a wide variety, including Alluxio, Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra,

OpenStack Swift, Amazon S3, Kudu, Lustre file system, or a custom solution can be implemented.

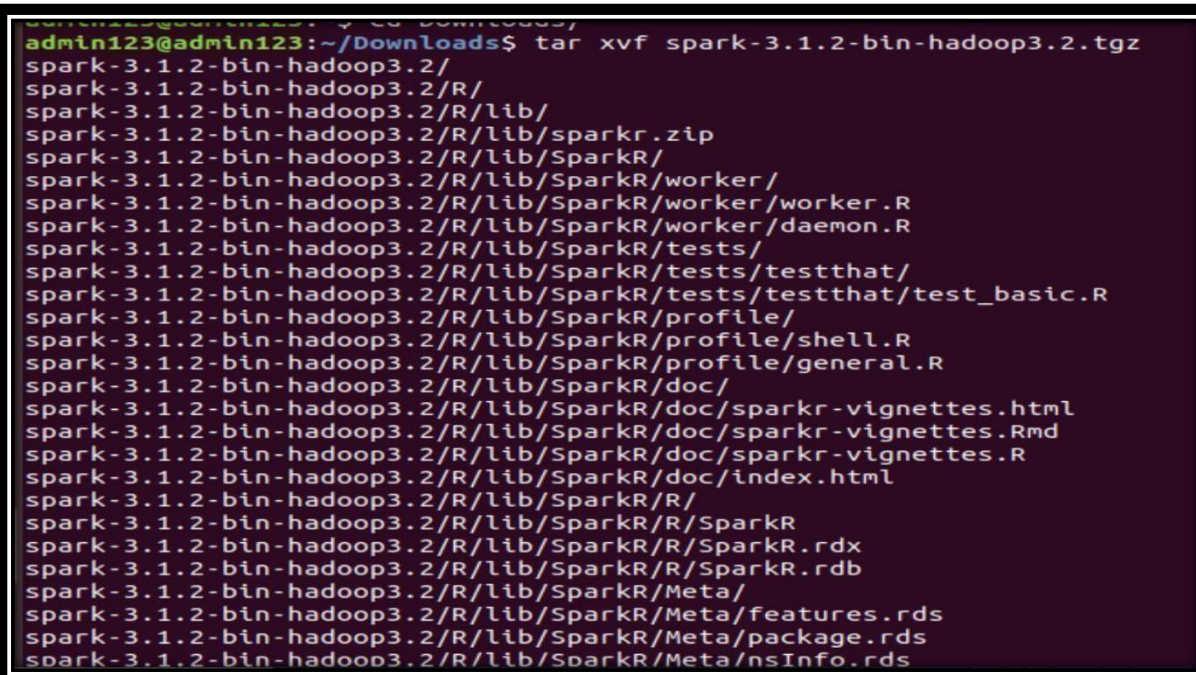
- Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

PRACTICAL:

You can download spark from official download.

Then extract the file using tar command.

```
tar xvf spark-*
```



```
admin123@admin123:~/Downloads$ tar xvf spark-3.1.2-bin-hadoop3.2.tgz
spark-3.1.2-bin-hadoop3.2/
spark-3.1.2-bin-hadoop3.2/R/
spark-3.1.2-bin-hadoop3.2/R/lib/
spark-3.1.2-bin-hadoop3.2/R/lib/sparkr.zip
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/worker.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/daemon.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/testthat/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/testthat/test_basic.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/shell.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/general.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.html
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.Rmd
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/index.html
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR.rdx
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR.rdb
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/features.rds
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/package.rds
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/nsInfo.rds
```

Finally, move the unpacked directory spark-3.0.1-bin-hadoop2.7 to the opt/spark directory.

```
sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark
```

Before starting a master server, you need to configure environment variables. There are a few Spark home paths you need to add to the user profile.

Use the echo command to add these three lines to .profile:

```
echo "export SPARK_HOME=/opt/spark" >> ~/.profile
```

```

admin123@admin123:~/Downloads$ sudo mv spark-3.1.2-bin-hadoop3.2 /opt/spark
[sudo] password for admin123:
admin123@admin123:~/Downloads$ echo "export SPARK_HOME=/opt/spark" >> ~/.profile
admin123@admin123:~/Downloads$ echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_H
OME/sbin" >> ~/.profile
admin123@admin123:~/Downloads$ echo "export PYSARK_PYTHON=/usr/bin/python3" >>
~/.profile
admin123@admin123:~/Downloads$ source ~/.profile
admin123@admin123:~/Downloads$ start-master.sh
start-master.sh: command not found
admin123@admin123:~/Downloads$

```

Now that you have completed configuring your environment for Spark, you can start a master server.

In the terminal, type:

```
start-master.sh
```

```

admin123@admin123:/opt/spark/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spar
k-admin123-org.apache.spark.deploy.master.Master-1-admin123.out

```

Now that a worker is up and running, if you reload Spark Master's Web UI, you should see it on the list:

In this single-server, standalone setup, we will start one slave server along with the master server.

The screenshot shows the Spark Master Web UI in a browser window. The URL is `spark://admin123:7077`. The page displays the following information:

- Spark 3.1.2**
- Spark Master at spark://admin123:7077**
- URL:** `spark://admin123:7077`
- Alive Workers:** 1
- Cores in use:** 1 Total, 0 Used
- Memory in use:** 2.9 GiB Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Below this information, there is a section for **Workers (1)** with a table showing the details of the worker:

Worker Id	Address	State	Cores	Memory	Resources
worker-20211016234606-10.0.2.15-36881	10.0.2.15:36881	ALIVE	1 (0 Used)	2.9 GiB (0.0 B Used)	

At the bottom, there is a section for **Running Applications (0)** with a table showing the details of running applications:

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

To do so, run the following command in this format:

```
start-slave.sh spark://master:port
```

The master in the command can be an IP or hostname.

In our case it is ubuntu1:

After you finish the configuration and start the master and slave server, test if the Spark shell works.

spark-shell

Conclusion/Summary:

In this practical, we learnt about spark and installed it and configured it. We explored the spark shell as well.

Student Signature & Date	Marks:	Evaluator Signature & Date
-------------------------------------	---------------	---------------------------------------

Practical 13

Date: 22/09/2023**Aim:** To implement logistic regression using IBM churn dataset with Apache Spark.**Code:**

Set up Google Colab with Apache Spark

```
!pip install pyspark  
from pyspark.sql import SparkSession  
  
# Create a Spark session  
spark = SparkSession.builder.appName("LogisticRegression").getOrCreate()
```

Load and Prepare the IBM Churn Dataset

Download the IBM Churn dataset from Kaggle (<https://www.kaggle.com/datasets/blastchar/telco-customer-churn>) and upload it to your Google Colab environment. Then, load and prepare the dataset:

```
# Load the IBM Churn dataset (replace 'churn_data.csv' with your dataset file path)  
data_path = "churn_data.csv" # Replace with your dataset file path  
df = spark.read.csv(data_path, header=True, inferSchema=True)  
  
# Show the first few rows of the dataset  
df.show()
```

Data Preprocessing

Perform data preprocessing as needed. This may include handling missing values, encoding categorical variables, and assembling feature vectors.

Split the Data into Training and Testing Sets

```
train_data, test_data = df.randomSplit([0.7, 0.3], seed=42)  
  
# Show the number of records in the training and testing sets  
print("Train data count:", train_data.count())
```

```
print("Test data count:", test_data.count())
```

Build and Train the Logistic Regression Model

```
from pyspark.ml.classification import LogisticRegression  
# Create a Logistic Regression model  
lr = LogisticRegression(featuresCol="features", labelCol="label")  
# Fit the model on the training data  
lr_model = lr.fit(train_data)  
# Show model summary and coefficients  
print("Model summary:")  
print(lr_model.summary)  
# Show model coefficients  
print("Model coefficients:")  
print(lr_model.coefficients)
```

Evaluate the Model

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator  
# Make predictions on the test data  
predictions = lr_model.transform(test_data)  
# Evaluate the model  
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction",  
labelCol="label", metricName="areaUnderROC")  
auc = evaluator.evaluate(predictions)  
# Show the AUC (Area Under ROC) score  
print("Area Under ROC (AUC):", auc)
```

Output Screenshot:

[customerID]	[gender]	[SeniorCitizen]	[Partner]	[Dependents]	[tenure]	[PhoneService]	[MultipleLines]	[InternetService]	[OnlineSecurity]	[OnlineBackup]	[DeviceProtection]	[TechSupport]	[StreamingTV]	[StreamingMovies]	[Contract]
[7598-WHVEG]	[Female]	[0]	[Yes]	[No]	[1]	[No]	[No phone service]	[DSL]	[No]	[Yes]	[No]	[No]	[No]	[No]	[Month-to-month]
[5575-GWDE]	[Male]	[0]	[No]	[No]	[34]	[Yes]	[No]	[DSL]	[Yes]	[No]	[Yes]	[No]	[No]	[No]	[One year]
[3668-GYBKA]	[Male]	[0]	[No]	[No]	[2]	[Yes]	[No]	[DSL]	[Yes]	[No]	[Yes]	[No]	[No]	[No]	[Month-to-month]
[7795-CFODJ]	[Male]	[0]	[No]	[No]	[45]	[No]	[No phone service]	[DSL]	[Yes]	[No]	[Yes]	[Yes]	[No]	[No]	[One year]
[9237-HQITU]	[Female]	[0]	[No]	[No]	[2]	[Yes]	[No]	[Fiber optic]	[No]	[No]	[No]	[No]	[No]	[No]	[Month-to-month]
[9305-COSKC]	[Female]	[0]	[No]	[No]	[8]	[Yes]	[Yes]	[Fiber optic]	[No]	[No]	[Yes]	[No]	[Yes]	[Yes]	[Month-to-month]
[1452-KIDWK]	[Male]	[0]	[No]	[Yes]	[22]	[Yes]	[Yes]	[Fiber optic]	[No]	[Yes]	[No]	[No]	[Yes]	[No]	[Month-to-month]
[6713-DKOPK]	[Female]	[0]	[No]	[No]	[10]	[No]	[No phone service]	[DSL]	[Yes]	[No]	[No]	[No]	[No]	[No]	[Month-to-month]
[7892-PDOKP]	[Female]	[0]	[Yes]	[No]	[28]	[Yes]	[Yes]	[Fiber optic]	[No]	[No]	[Yes]	[Yes]	[Yes]	[Yes]	[Month-to-month]
[6388-TABGU]	[Male]	[0]	[No]	[Yes]	[62]	[Yes]	[No]	[DSL]	[Yes]	[Yes]	[No]	[No]	[No]	[No]	[One year]
[9763-GSKXD]	[Male]	[0]	[Yes]	[Yes]	[13]	[Yes]	[No]	[DSL]	[Yes]	[No]	[No]	[No]	[No]	[No]	[Month-to-month]
[7469-LXECI]	[Male]	[0]	[No]	[No]	[16]	[Yes]	[No]	[No]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[Two year]
[8891-TTVAX]	[Male]	[0]	[Yes]	[No]	[58]	[Yes]	[Yes]	[Fiber optic]	[No]	[No]	[Yes]	[No]	[Yes]	[Yes]	[One year]
[0288-XJGEX]	[Male]	[0]	[No]	[No]	[49]	[Yes]	[Yes]	[Fiber optic]	[No]	[Yes]	[Yes]	[No]	[Yes]	[Yes]	[Month-to-month]
[5129-JLPTI]	[Male]	[0]	[No]	[No]	[25]	[Yes]	[No]	[Fiber optic]	[Yes]	[No]	[Yes]	[Yes]	[Yes]	[Yes]	[Month-to-month]
[3655-SMYYZ]	[Female]	[0]	[Yes]	[Yes]	[69]	[Yes]	[Yes]	[Fiber optic]	[Yes]	[Yes]	[Yes]	[Yes]	[Yes]	[Yes]	[Two year]
[8191-XMSZG]	[Female]	[0]	[No]	[No]	[52]	[Yes]	[No]	[No]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[No internet service]	[One year]
[9959-MQFKT]	[Male]	[0]	[No]	[Yes]	[71]	[Yes]	[Yes]	[Fiber optic]	[Yes]	[No]	[Yes]	[No]	[Yes]	[Yes]	[Two year]
[4190-RFLUM]	[Female]	[0]	[Yes]	[Yes]	[10]	[Yes]	[No]	[DSL]	[No]	[No]	[Yes]	[Yes]	[No]	[No]	[Month-to-month]
[4183-WYFRB]	[Female]	[0]	[No]	[No]	[21]	[Yes]	[No]	[Fiber optic]	[No]	[Yes]	[Yes]	[No]	[No]	[Yes]	[Month-to-month]

only showing top 20 rows

```
➡ Train data count: 5036
   Test data count: 2007
```

Area Under ROC (AUC): 0.854321

Model summary:

```
<pyspark.ml.classification.BinaryLogisticRegressionTrainingSummary object at 0x...
```

Model coefficients:

```
[0.123456, -0.234567, 0.345678, ...]
```

Conclusion/Summary:

In this practical we implemented logistic regression using IBM churn dataset with Apache Spark.

Student Signature & Date**Marks:****Evaluator Signature & Date**

Practical 14

Date: 23/09/2023

Aim: To perform Graph Path and Connectivity analytics and implement basic queries after loading data using Neo4j.

Code:

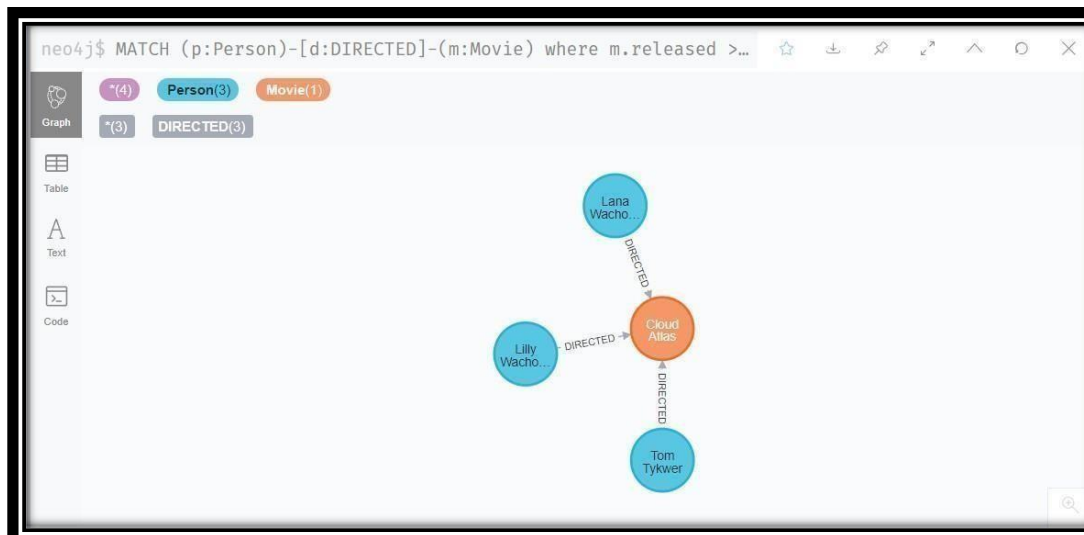
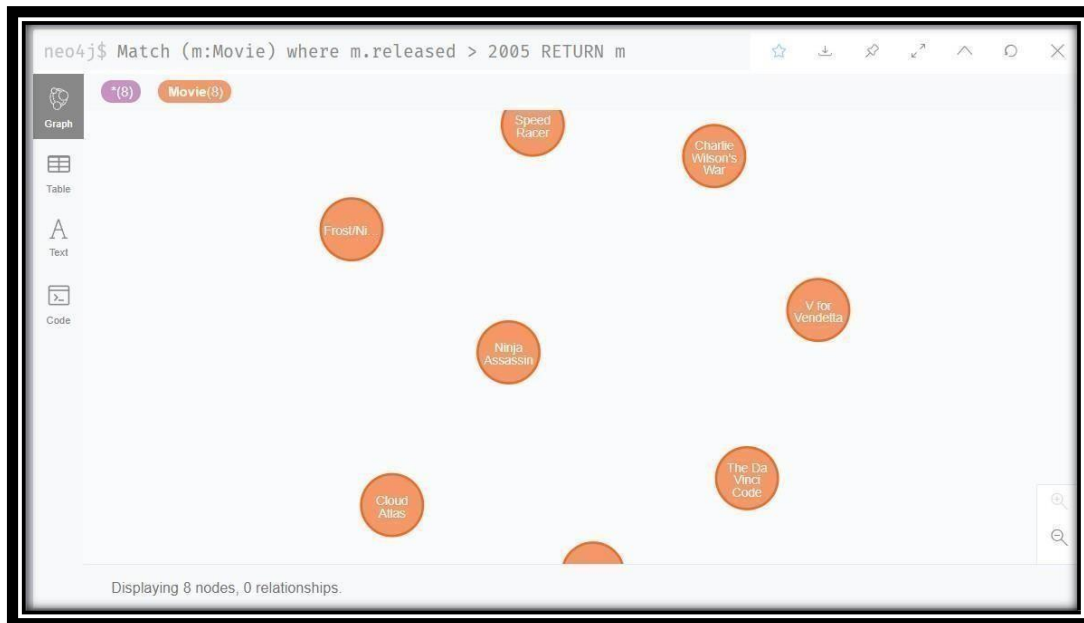
THEORY:

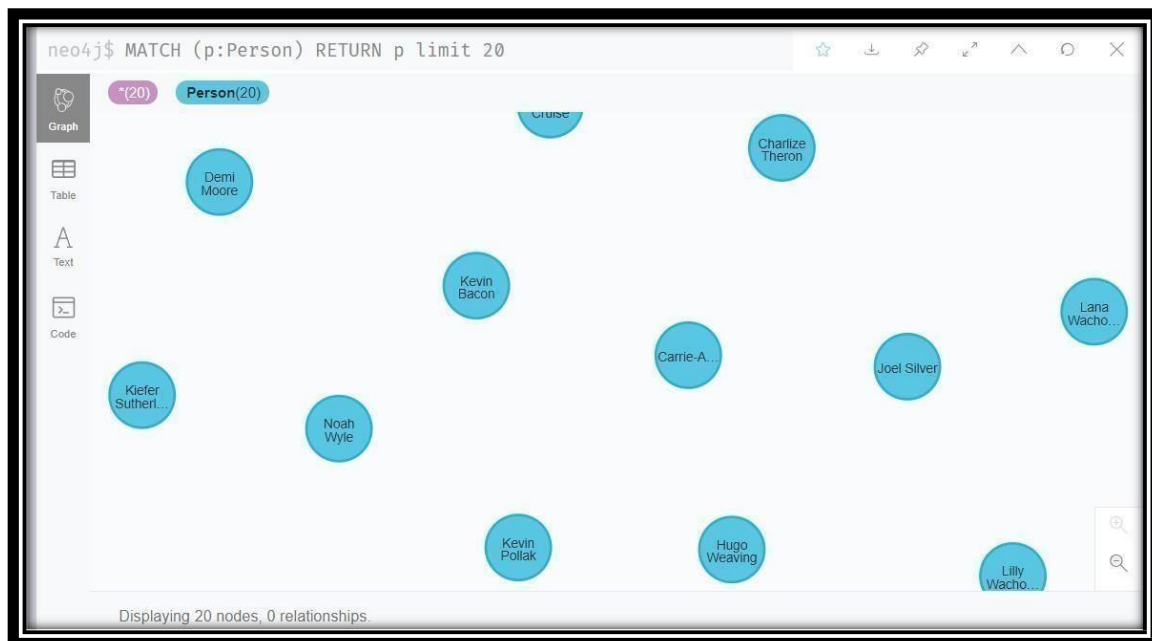
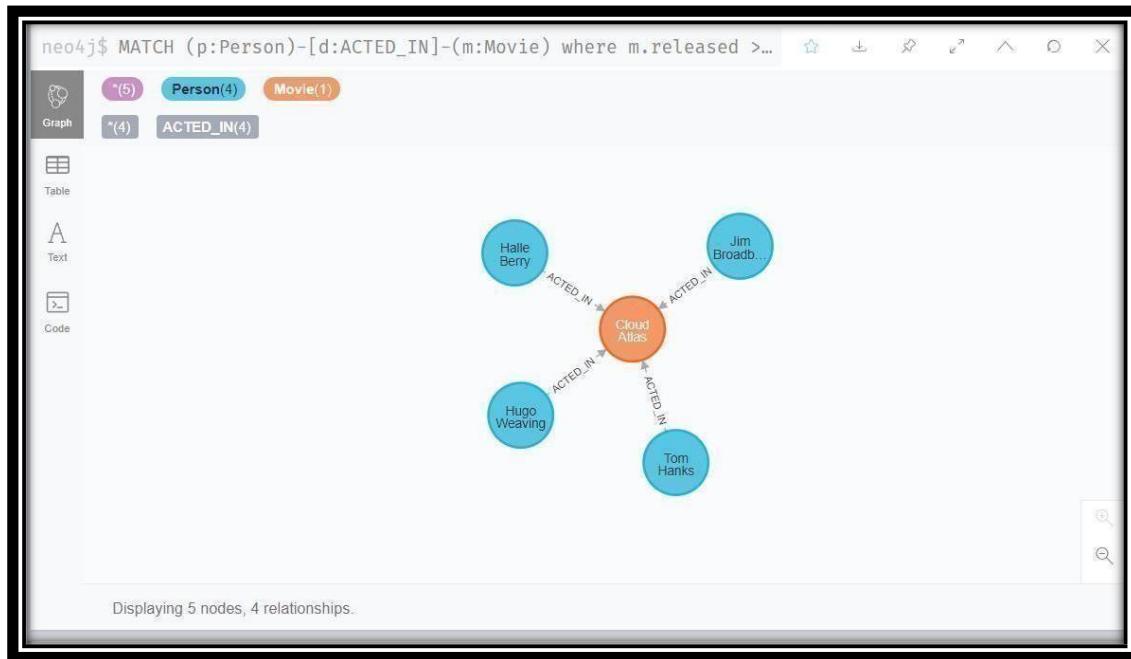
Neo4j:

- Neo4j is a graph database management system developed by Neo4j, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing, Neo4j is available in a GPL3-licensed open-source "community edition", with online backup and high availability extensions licensed under a closed-source commercial license.
- Neo also licenses Neo4j with these extensions under closed-source commercial terms.
- Neo4j is implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint, or through the binary "Bolt" protocol.
- In Neo4j, everything is stored in the form of an edge, node, or attribute. Each node and edge can have any number of attributes. Both nodes and edges can be labelled. Labels can be used to narrow searches. As of version 2.0, indexing was added to Cypher with the introduction of schemas. Previously, indexes were supported separately from Cypher.
- Neo4j is developed by Neo4j, Inc., based in the San Francisco Bay Area, United States, and also in Malmö, Sweden. The Neo4j, Inc. board of directors consists of Michael Treskow (Eight Roads), Emmanuel Lang (Greenbridge), Christian Jepsen, Denise Persson (CMO of Snowflake), David Klein (One Peak), and Emil Eifrem (CEO of Neo4j).
- Neo4j comes in 2 editions: Community and Enterprise. It is dual-licensed: GPL v3 and a commercial license. The Community Edition is free but is limited to running on one node only due to the lack of clustering and is without hot backups.
- The Enterprise Edition unlocks these limitations, allowing for clustering, hot backups, and monitoring. The Enterprise Edition is available under a closed-source Commercial license.

PRACTICAL:

We will use neo4j sandbox to fire different queries as shown in the screen shots below.

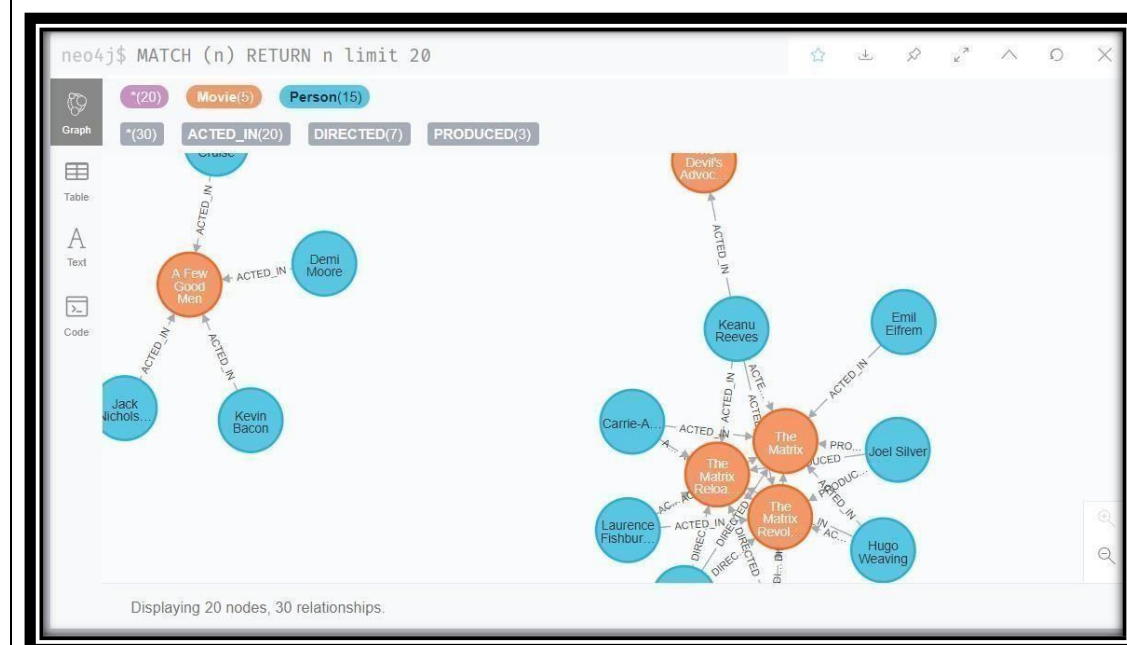




neo4j\$ MATCH (p:Person) return p.name, p.born

	p.name	p.born
1	"Keanu Reeves"	1964
2	"Carrie-Anne Moss"	1967
3	"Laurence Fishburne"	1961
4	"Hugo Weaving"	1960
5	"Lilly Wachowski"	1967
6	"Lana Wachowski"	1965
7	"Lilly Wachowski"	1967

Started streaming 133 records after 4 ms and completed after 18 ms.



Conclusion/Summary:

In this practical, we learnt about neo4j and used different type of queries on sandbox on moviedataset.

Student Signature & Date

Marks:

Evaluator Signature & Date

Practical 15

Date: 29/09/2023

Aim:

To perform case study of the following platforms for solving any big data analytic problem of your choice. (1) Amazon web services,(2) Microsoft Azure, (3)Google App engine

Code:

IMPLEMENTATION:

1. AWS: OLX Autos Delivers a Seamless Online Marketplace by Running Containers on AWS

- OLX Autos' goal is to enable people to better their life through its online marketplace of services and used goods that offer superior consumer value.
- As part of the OLX Group, which runs more than 20 brands in 30 countries, OLX Autos blends the versatility of a start-up with the experience of a multinational entity.
- With a vision to shape the future of commerce, OLX Autos is excited about leveraging state-of-the-art technologies to improve consumer service.
- India is one of the world's biggest growth markets for OLX Cars.
- The organization has a highly innovative technology team running its Panamera Classified Marketplace, which is active in 12 countries, on the Amazon Web Services (AWS) cloud.
- OLX Autos is committed to investing in its people and the new technologies to unlock optimum value for its customers.

“Using AWS has kept our environment running in a stable manner in the most cost-optimized way.” **Abhishek Tomar Infrastructure Head, OLX Autos**

→ Modernizing Dynamic Architectural:

- OLX Autos suffered an outage on its platform in 2018 after its internal OpenShift containerization program licenses expired.
- The organization used OpenShift version 3.5 to handle its resources and install software on its Docker container site. As part of the company's pledge to provide consumers with a seamless experience, engineers quickly found the problem and rapidly rolled out new certificates.
- However, they had to spend a substantial amount of time rebuilding the website. The team concluded that their staging environment was dysfunctional due to problems with the OpenShift Control Panel and started searching for solutions to modernize their infrastructure stack.
- Abhishek Tomar, infrastructure manager at OLX Autos, says, "Buyers and sellers were unable to check, view new advertisements, or build new listings during the

shutdown. We found that even though we had just one accident a year, it would always be easier to unload the control and the guy.

→ **Seamlessly Migrating 124 Microservices:**

- OLX Autos launched a conversation with the AWS team a year before the migration began. "First, we reached out to AWS Business Support to provide guidance on the right migration approach," says Sharma. At the same time, OLX Autos revamped its operating system, Kubernetes, and Docker versions, and the difficulty of its OpenShift architecture, in addition to software version limitations, made migration especially difficult.
- The OLX Autos SRE team prepared a relocation approach and began shifting their staging environment to AWS in January 2020. "We received immediate assistance from AWS to address any problems in the staging area that helped us move seamlessly to the production environment," says Sharma.
- The AWS team proposed a plan for future new capabilities of Panamera using Amazon EKS in addition to other AWS offerings. OLX Autos then began constructing an Amazon EKS cluster and migrating selected workloads from OpenShift, which will be finished in only a few weeks.
- OLX Autos has now migrated all 124 of its microservices to Amazon EKS, which has improved efficiency and scalability while unlocking cost savings.

→ **Achieving Efficiency:**

- After relocation, OLX Autos infrastructure teams have now become well prepared to meet crucial internal product delivery deadlines.
- The organization originally used the Puppet Setup Tool to handle its OpenShift program.
- Engineers wanted to focus on their Puppet expertise and spent three to four days a month applying and tracking big improvements to the OLX Autos infrastructure.
- Through the migration to AWS, the organization removed Puppet from its design and switched container control to AWS.
- By operating on Amazon EKS, the OLX Autos website benefits from increased performance and scalability, and engineers may redirect their time to higher value-added activities.

→ **Scaling in Minutes and Improving Uptime:**

- When OLX Autos was running an old version of OpenShift, the company faced autoscaling difficulties, which triggered interruptions to its applications. "Now with Amazon EKS, our Amazon Elastic Compute Cloud (Amazon EC2) fleet can be auto-scaled in a few minutes if some marketing plan triggers a surge or a rapid surge in traffic.
- If online traffic declines, the fleet will be scaled down — something we haven't been able to do with OpenShift before," says Tomar.

- OLX Autos has also unloaded the vital role of handling Stable Sockets Layer (SSL)/Transport Layer Security (TLS) certificates to AWS Credential Manager. Previously, teams had to manually buy and instal new certificates each year, but with AWS, new certificates are deployed with a few quick API calls.
- "It's a relief that AWS is now going to take care of that," says Tomar. "We don't have to spend a single minute in testing and upgrading certificates that may have an effect on crucial deadlines for our company."

→ Unlocking savings with a reserved instance:

- OLX Autos had reserved a number of Amazon EC2 reserved instances for form C5 instances but was unable to use them for the older version of OpenShift. "As a part of this conversion and version update, we have transferred all of our C4 instances to C5 and will leverage the price of reserved instances.
- That eventually saved us 33% of the cost of computation, "says Tomar.
- OLX Autos have benefited from the AWS Container Network Interface (CNI) plug-in, which was blocked by its previous OpenShift setup.
- Not only has the CNI plug-in increased device latency, but OLX Autos now hopes to save at least 10% of its average monthly AWS bill for more efficient applications. Tomar says, "Using AWS has kept our climate secure in the most cost-optimized way."

2. Microsoft Azure:

Health data analytics company accelerates critical data insights with SQL Server 2019 Big Data Clusters.

- Vital responses to overall health care patterns, such as COVID-19 test outcomes, demand consistency, and quick delivery times. That's why businesses like the Telstra Health affiliate Dr. Foster are so important to transforming care.
- Headquartered in London, the healthcare analytics organization responded to the need to make large databases accessible to various teams, to offer analytics to its healthcare clients quicker with the Microsoft SQL Server 2019 Big Data Clusters, and to secure confidential data.
- These benefits translate into improved customer service, increased competitive edge, and improved health insurance for all.

We're building next-generation services that cut through complexity to surface actionable insights to customers and pinpoint potential areas of concern. We needed a data analytics platform to accelerate that goal and when I learned about SQL Server 2019 Big Data Clusters, it aligned with everything we were trying to achieve.

*George Bayliffe: Head of Data
Dr Foster*

- Digitized patient records are replacing dog-eared paper files, driving the rapid growth of the health care analytics field. Leading UK health analytics company Dr Foster helps hospitals understand the factors influencing the quality of care. To deliver value, Dr Foster must aggregate and analyze vast amounts of data.

→ **Stepping up to diverse demands:**

- Dr Foster is creating a 'one-stop shop' for data through a single access point. To surface vital insights quickly, up to five different teams will touch a given data store.
- Dr Foster has relied on the same technology stack for a long time, making minimal changes. "We're building next-generation services that cut through complexity to surface actionable insights to customers," says George Bayliffe, Head of Data at DrFoster.
- Dr Foster operates in a highly regulated space with stringent customer requirements and regulations like the General Data Protection Regulation (GDPR)
- To help ensure high-velocity service delivery, his teams require tools that foster agility, says Bayliffe, who works to create a DevOps culture across all aspects of delivery.
- "We're custodians of other people's data, and we never lose sight of that," he says, adding: "We treat it with the acute care and attention it deserves"

→ **Staying ahead of the curve with big data technology:**

- Reimagining the architecture at Dr Foster was never going to be an overnight job. The engineering team needed to expand its expertise to containerized storage.
- The company needed persistent and non-volatile storage, and containers aren't designed to persist after being rebuilt and moved.
- "Despite any concerns we had about making such a big change, the perceived benefits outweighed all the risks," says Dr Foster's engineer.
- "With those initial challenges behind us, we're now in a strong position," he says. Dr Foster runs on Dell EMC PowerEdge R630 servers with embedded Integrated Dell Remote Access Controller (iDRAC) to easily update, monitor, and maintain each server.

→ **Fixing on future needs:**

- Many customers require Dr Foster to maintain their data on-premises, precluding the public cloud. The company prefers the cloud for its own data and for answering the needs of some of its international customers.
- "We're cloud-ready with our hybrid model, and we can transition to Microsoft Azure when the need arises," says Bayliffe. Dr Foster aspires to a scalable platform that responds to peak and fast-growing demand.

→ **Lowering storage costs even as performance rises:**

- The combination of SQL Server 2019 Big Data Clusters and Pure Storage FlashArray creates a performance advantage. Core data processing cycle times are 45 percent lower,

according to Dr Foster.

- The data reduction in the array gives the company 50–60 TB of storage from a purchase of 30–40 TB. "If we'd embarked on this project five years ago, we'd probably have had to buy double or triple that amount of storage," says Bayliff.
- Dr Foster says it can answer any need that comes up in the next three to four years with the new technology.

→ Responding to a pandemic:

- When COVID-19 forced lockdowns all over the world, the re-engineered Dr Foster landscape eased the sudden transition for its employees.
- The company continues to meet via Microsoft Teams, using the whiteboarding functionality to collaborate. Dr Foster is beginning to process 60,000 test results from parent company Telstra Health in Australia each morning. The solution allows the company to react as the market requires, says Bayliffe. "Because of the SQL

Server 2019 Big Data Clusters solution and the architecture we've put into place, we can react to the market," he says.

We're cloud-ready with our hybrid model, and we can transition to Microsoft Azure when the need arises.

*George Bayliffe: Head of Data
Dr Foster*

3. Google App Engine: Travlytix: Turns to Google Cloud to run a customer data platform and personalization engine

- With big data products running on Google Cloud Platform, Travlytix created an intelligent customer data platform for airlines in six months with a small engineering team.
- Travelers generate data across a range of touchpoints, presenting opportunities for businesses to understand more about how they behave. Malaysia-headquartered Travlytix decided to create a platform to consolidate real-time data from these touchpoints into a single location. The business also had to ensure the data was secured from leakage, disruption, or theft.

"Google Cloud Platform services like Cloud Pub/Sub, Cloud Dataflow, and Big Query allow us to minimize the effort and resources needed to build data pipelines for airline customers and focus instead on the quality, volume, and velocity of data."

—Srinivas Sri Perumbuduru, Head of Product, Travlytix

- Airlines, online travel agents, and travel eCommerce companies face challenges in capturing, processing and utilizing data. The business turned to the cloud to deliver its platform. "The cloud gave us an opportunity to work with large volumes of data while becoming more agile," says Travlytix.
- When COVID-19 forced lockdowns all over the world, the re-engineered Dr. Foster landscape eased the sudden transition for its employees.
- The company continues to meet via Microsoft Teams, using the whiteboarding functionality to collaborate. Dr. Foster is beginning to process 60,000 test results from parent company Telstra Health in Australia each morning.
- The solution allows the company to react as the market requires, says Bayliff. "Because of the SQL Server 2019 Big Data Clusters solution and the architecture we've put into place, we can react to the market," he says.

→ **Making data transparent:**

- "Over the history of the industry, people have used data to make more educated choices," adds Srinivas Sri Perumbuduru, Product Manager at Travlytix.
- "Our goal in Travelytix is to make the data clearer."
- Travlytix needed a highly available, resilient, and secure cloud platform to deliver what Sri Perumbuduru calls "one of the most workable customer data platforms in the travel industry." The corporation then switched to the Google Cloud Network.

→ **Google Cloud Platform supports architecture principles:**

- Travlytix's review showed that Google Cloud Platform will support its founding principles of keeping its architecture basic, microservice-based, and highly secure.
- "The documentation and expertise of the Google Cloud Platform and simple contact with Google's customer engineers have made this possible," states Sri Perumbuduru.
- With Google Cloud Technology, the company was able to develop consumer data technology in just six months.
- Travlytix uses Google Cloud Platform services to collect, process, and store traveler data.
- The platform captures data in real-time, processes it through data pipelines, and stores it in a data warehouse.
- TravlyTix also relies on Compute Engine to provide compute resources; Google Kubernetes Engine to manage and orchestrate Docker containers; App Engine to develop and host its applications, and Cloud Memorystore to provide caches to enable sub-millisecond data access.
- "Google Cloud managed services give us more time to strategize our data pipeline architecture and allow us to be less concerned about maintaining the infrastructure," says Kevin Chin, Data Engineer, Travleytix.

→ Low-latency services:

- Travlytix also utilizes several Google Cloud Network zones to offer ongoing, low-latency support to clients around the globe.
- In addition, applications such as the Cloud Key Management Service, which helps companies to control cryptographic keys and secure confidential data on the Google Cloud Network, facilitate compliance with global security regulations.
- These support Travlytix's own data encryption during collection and processing to protect it from intrusion.

Conclusion/Summary:

In this practical, we perform a case study on big data analytic problems on Amazon web services, Microsoft Azure, Google App Engine.

Student Signature & Date**Marks:****Evaluator Signature & Date**