ERODE SENGUNTHAR
ENGINEERING COLLEGE

(APPROVED BY AICTE, NEW DELHI & PERMANENTLY AFFILIATED TO ANNA UNIVERSITY, CHENNAI.
ACCREDITED BY NBA, NEW DELHI, NAAC WITH GRADE "A" & IE(I),KOLKATA)

PERUNDURAI, ERODE – 638 057

An Autonomous Institution

BONAFIDE CERTIFICATE

Register No: .....

Certified that this is the Bonafide Record of Work Done By

Name of the Student :.....

Branch :.....

Lab Code/Name :.....

Year/Semester :.....

Faculty Incharge Head of the Department

Submitted for the End Semester Practical Examination held on .....

Internal Examiner *External Examiner*

INDEX

S.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	MARKS	FACULTY SIGN
1		Hypothesis Test using R			
2		K Means Clustering using R			
3		Implementation of Linear and Logistic Regression			
4		Time Series Analysis using R			
5		Data Analysis-Visualization using R			
6		Install and Configure Hadoop			
7		Map Reduce using Hadoop			
8		Implementation of queries using MongoDB and Cassandra			
9		Using Apache Spark for Data Analytics			
		CONTENT BEYOND THE SYLLABUS			
10		Filter and Sort Sales Data using Pig			

EX NO: 1 DATE:	HYPOTHESIS TEST USING R

AIM:

To write a R Program to implement Hypothesis Test

PROGRAM:

```
x<-rnorm(100)
t.test(x,mu=5)

x1<-rnorm(100)
y1<-rnorm(100)
t.test(x1,y1)

x2<-rnorm(100)
t.test(x2,mu=2,alternative='greater')

x3<-rnorm(100)
wilcox.test(x3,exact=FALSE)

cor.test(mtcars$mpg,mtcars$hp)
```

OUTPUT:

```
        One Sample t-test

data:  x
t = -49.019, df = 99, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 -0.1830910  0.2201926
sample estimates:
 mean of x
0.01855082


        Welch Two Sample t-test

data:  x1 and y1
t = 0.60676, df = 191.79, p-value = 0.5447
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2093324  0.3953476
sample estimates:
  mean of x   mean of y
 0.01631522 -0.07669237


data:  x2
t = -21.291, df = 99, p-value = 1
alternative hypothesis: true mean is greater than 2
95 percent confidence interval:
 -0.2129391        Inf
sample estimates:
  mean of x
-0.05284852


        Wilcoxon signed rank test with continuity correction

data:  x3
V = 2318, p-value = 0.4777
alternative hypothesis: true location is not equal to 0
```

```
        Pearson's product-moment correlation

data:  mtcars$mpg and mtcars$hp
t = -6.7424, df = 30, p-value = 1.788e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8852686 -0.5860994
sample estimates:
       cor
-0.7761684
```

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the R Program to implement Hypothesis Test has been executed and Verified Successfully.

EX NO: 2 DATE:	K-MEANS CLUSTERING USING R

AIM:

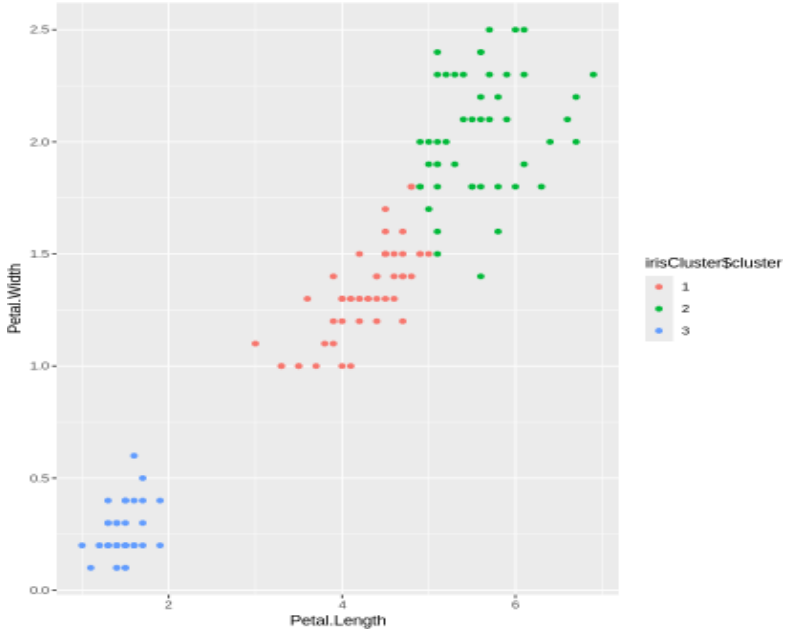
To write a R Program to implement K Means Clustering

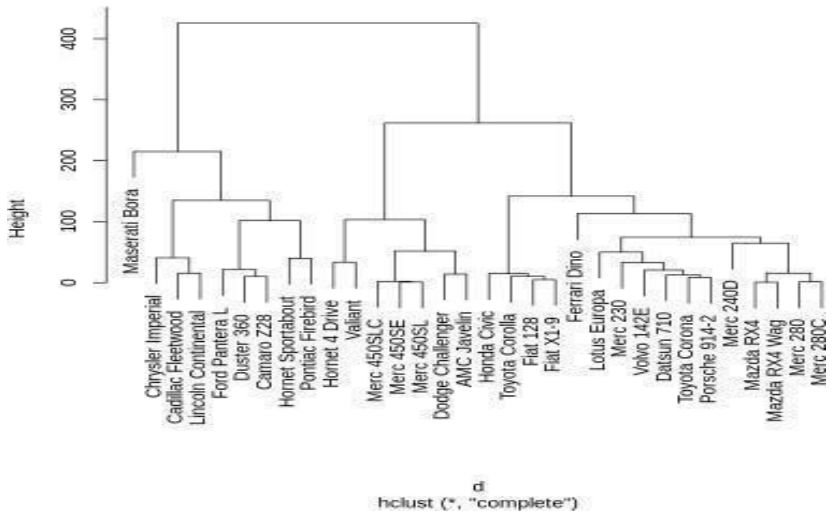
PROGRAM:

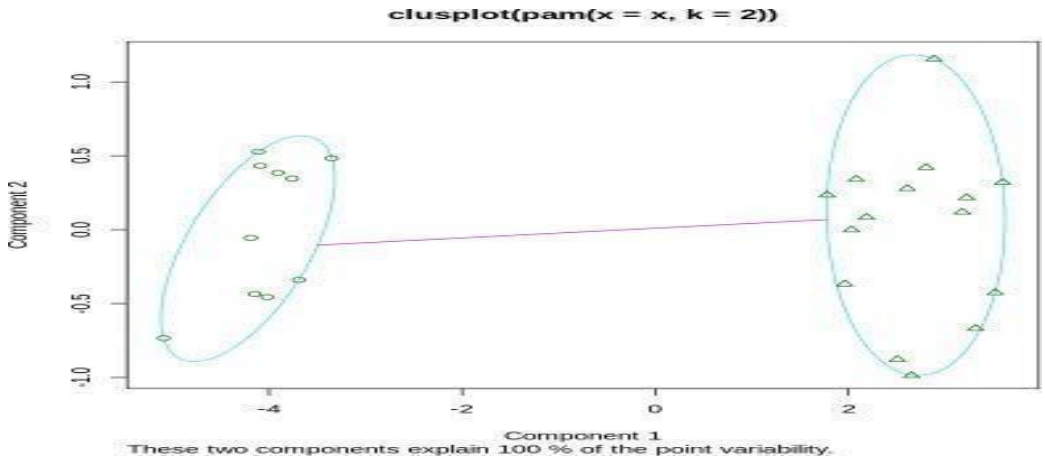
```
library(cluster)
library(ggplot2)

set.seed(20)
irisCluster<-kmeans(iris[,3:4],3,nstart=20)
irisCluster

irisCluster$cluster<-as.factor(irisCluster$cluster)
ggplot(iris,aes(Petal.Length,Petal.Width,color=irisCluster$cluster))+geom_p
oint()

d<-dist(as.matrix(mtcars))
hc<-hclust(d)
plot(hc)

x<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5),rnor
m(15,5,0.5)))
clusplot(pam(x,2))

x4<-cbind(x,rnorm(25),rnorm(25))
clusplot(pam(x4,2))
```

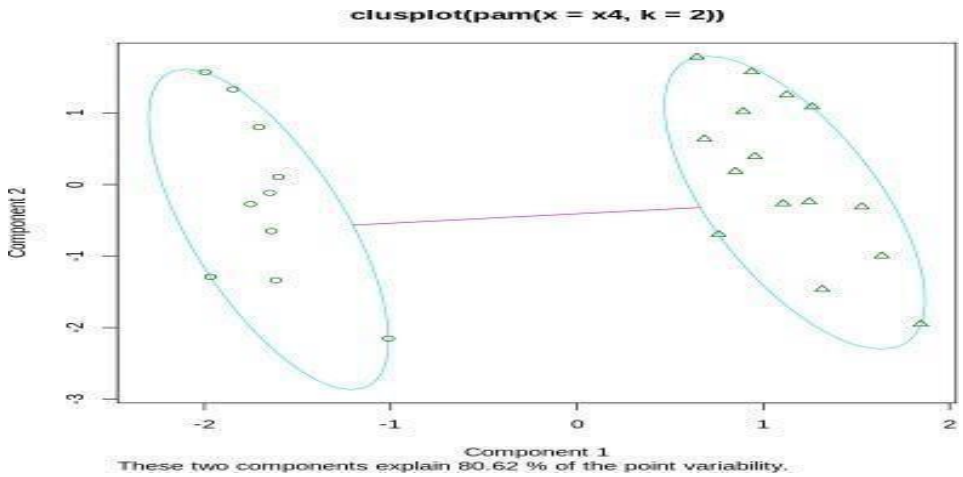
OUTPUT:

```
K-means clustering with 3 clusters of sizes 52, 48, 50

Cluster means:
  Petal.Length Petal.Width
1     4.269231    1.342308
2     5.595833    2.037500
3     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2
[112] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
[149] 2 2

Within cluster sum of squares by cluster:
[1] 13.05769 16.29167  2.02200
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```




Height
0
100
200
300
400
Maserati Bora
Chrysler Imperial
Cadillac Fleetwood
Lincoln Continental
Ford Pantera L
Duster 360
Camaro Z28
Hornet Sportabout
Pontiac Firebird
Hornet 4 Drive
Valiant
Merc 450SLC
Merc 450SE
Merc 450SL
Dodge Challenger
AMC Javelin
Honda Civic
Toyota Corolla
Fiat 128
Fiat X1-9
Ferrari Dino
Lotus Europa
Merc 230
Volvo 142E
Datsun 710
Toyota Corona
Porsche 914-2
Merc 240D
Mazda RX4
Mazda RX4 Wag
Merc 280
Merc 280C
d
hclust (*, "complete")


clusplot(pam(x = x, k = 2))
Component 2
1.0
0.5
0.0
-0.5
-1.0
-4
-2
0
2
Component 1
These two components explain 100 % of the point variability.



Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the R Program to implement K Means clustering has been executed and Verified successfully

EX NO: 3 DATE:	IMPLEMENTATION OF LINEAR AND LOGISTIC REGRESSION

AIM:

To write an R Program to implement Linear and Logistic Regression.

PROGRAM:

```
dataset=read.csv("/content/data-marketing-budget-12mo.csv",header=T,
    colClasses=c("numeric","numeric","numeric"))
head(dataset)

simple.fit = lm(Sales~Spend,data=dataset)
summary(simple.fit)

multi.fit = lm(Sales~Spend+Month, data=dataset)
summary(multi.fit)

input<- mtcars [,c("am","cyl","hp","wt")]
print(head(input))
input<- mtcars [,c("am","cyl","hp","wt")]
am.data =glm(formula = am ~ cyl+hp+wt,data = input,family = binomial)
print(summary(am.data))
```

OUTPUT:

	Month	Spend	Sales
	<dbl>	<dbl>	<dbl>
A data.frame: 6 × 3			
1	1	1000	9914
2	2	4000	40487
3	3	5000	54324
4	4	4500	50044
5	5	3000	34719
6	6	4000	42551

```
Call:
lm(formula = Sales ~ Spend, data = dataset)

Residuals:
   Min     1Q Median     3Q    Max
 -3385  -2097    258   1726   3034

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1383.4714  1255.2404   1.102    0.296
Spend         10.6222     0.1625  65.378 1.71e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2313 on 10 degrees of freedom
Multiple R-squared:  0.9977,    Adjusted R-squared:  0.9974
F-statistic:  4274 on 1 and 10 DF,  p-value: 1.707e-14
```

```
Call:
lm(formula = Sales ~ Spend + Month, data = dataset)

Residuals:
     Min       1Q   Median       3Q      Max
-1793.73 -1558.33    -1.73  1374.19  1911.58

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -567.6098  1041.8836  -0.545  0.59913
Spend         10.3825     0.1328  78.159 4.65e-14 ***
Month        541.3736   158.1660   3.423  0.00759 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1607 on 9 degrees of freedom
Multiple R-squared:  0.999,     Adjusted R-squared:  0.9988
F-statistic:  4433 on 2 and 9 DF,  p-value: 3.368e-14
```

```
                  am cyl  hp    wt
Mazda RX4          1   6 110 2.620
Mazda RX4 Wag      1   6 110 2.875
Datsun 710         1   4  93 2.320
Hornet 4 Drive     0   6 110 3.215
Hornet Sportabout  0   8 175 3.440
Valiant            0   6 105 3.460

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/ Programming & Execution	**20**	
Record	**20**	
Viva Voce	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus the R Program to implement Linear and Logistic Regression has been executed and Verified Successfully.

EX NO: 4 DATE:	TIME-SERIES ANALYSIS USING R

AIM:

To write a R Program to implement Time series analysis for the given data.

PROGRAM:

```
kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)
kings

kingstimeseries <- ts(kings)
kingstimeseries

births<-scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
birthstimeseries <- ts(births, frequency=12, start=c(1946,1))
birthstimeseries

souvenir<-scan("http://robjhyndman.com/tsdldata/data/fancy.dat")
souvenirtimeseries <- ts(souvenir, frequency=12, start=c(1987,1))
Souvenirtimeseries
```

OUTPUT:

1. 60 2. 43 3. 67 4. 50 5. 56 6. 42 7. 50 8. 65 9. 68 10. 43 11. 65 12. 34 13. 47 14. 34 15. 49 16. 41 17. 13 18. 35 19. 53 20. 56 21. 16 22. 43 23. 69 24. 59 25. 48 26. 59 27. 86 28. 55 29. 68 30. 51 31. 33 32. 49 33. 67 34. 77 35. 81 36. 67 37. 71 38. 81 39. 68 40. 70 41. 77 42. 56

A Time Series:
1. 60 2. 43 3. 67 4. 50 5. 56 6. 42 7. 50 8. 65 9. 68 10. 43 11. 65 12. 34 13. 47 14. 34 15. 49 16. 41 17. 13 18. 35 19. 53 20. 56 21. 16 22. 43 23. 69 24. 59 25. 48 26. 59 27. 86 28. 55 29. 68 30. 51 31. 33 32. 49 33. 67 34. 77 35. 81 36. 67 37. 71 38. 81 39. 68 40. 70 41. 77 42. 56

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
A Time Series: 14 × 12												
1946	26.663	23.598	26.931	24.740	25.806	24.364	24.477	23.901	23.175	23.227	21.672	21.870
1947	21.439	21.089	23.709	21.669	21.752	20.761	23.479	23.824	23.105	23.110	21.759	22.073
1948	21.937	20.035	23.590	21.672	22.222	22.123	23.950	23.504	22.238	23.142	21.059	21.573
1949	21.548	20.000	22.424	20.615	21.761	22.874	24.104	23.748	23.262	22.907	21.519	22.025
1950	22.604	20.894	24.677	23.673	25.320	23.583	24.671	24.454	24.122	24.252	22.084	22.991
1951	23.287	23.049	25.076	24.037	24.430	24.667	26.451	25.618	25.014	25.110	22.964	23.981
1952	23.798	22.270	24.775	22.646	23.988	24.737	26.276	25.816	25.210	25.199	23.162	24.707
1953	24.364	22.644	25.565	24.062	25.431	24.635	27.009	26.606	26.268	26.462	25.246	25.180
1954	24.657	23.304	26.982	26.199	27.210	26.122	26.706	26.878	26.152	26.379	24.712	25.688
1955	24.990	24.239	26.721	23.475	24.767	26.219	28.361	28.599	27.914	27.784	25.693	26.881
1956	26.217	24.218	27.914	26.975	28.527	27.139	28.982	28.169	28.056	29.136	26.291	26.987
1957	26.589	24.848	27.543	26.896	28.878	27.390	28.065	28.141	29.048	28.484	26.634	27.735
1958	27.132	24.924	28.963	26.589	27.931	28.009	29.229	28.759	28.405	27.945	25.912	26.619
1959	26.076	25.286	27.660	25.951	26.398	25.565	28.865	30.000	29.261	29.012	26.992	27.897

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
A Time Series: 7 × 12												
1987	1664.81	2397.53	2840.71	3547.29	3752.96	3714.74	4349.61	3566.34	5021.82	6423.48	7600.60	19756.21
1988	2499.81	5198.24	7225.14	4806.03	5900.88	4951.34	6179.12	4752.15	5496.43	5835.10	12600.08	28541.72
1989	4717.02	5702.63	9957.58	5304.78	6492.43	6630.80	7349.62	8176.62	8573.17	9690.50	15151.84	34061.01
1990	5921.10	5814.58	12421.25	6369.77	7609.12	7224.75	8121.22	7979.25	8093.06	8476.70	17914.66	30114.41
1991	4826.64	6470.23	9638.77	8821.17	8722.37	10209.48	11276.55	12552.22	11637.39	13606.89	21822.11	45060.69
1992	7615.03	9849.69	14558.40	11587.33	9332.56	13082.09	16732.78	19888.61	23933.38	25391.35	36024.80	80721.71
1993	10243.24	11266.88	21826.84	17357.33	15997.79	18601.53	26155.15	28586.52	30505.41	30821.33	46634.38	104660.67

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/ Programming & Execution	**20**	
Record	**20**	
Viva Voce	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus the R Program to implement Time Series Analysis has been executed and verified successfully.

EX NO: 5 DATE:	DATA ANALYSIS-VISUALIZATION USING R

AIM:

To Write an R Program to implement Data Visualization to explore Various Charts.

PROGRAM:

Code For Histogram:

```
library(RColorBrewer)
data(VADeaths)
par(mfrow=c(2,3))
hist(VADeaths,breaks=10, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
hist(VADeaths,breaks=3 ,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
hist(VADeaths,breaks=7, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
hist(VADeaths,,breaks= 2, col=brewer.pal(8,"Set3"),main="Set3 8 colors")
hist(VADeaths,col=brewer.pal(8,"Greys"),main="Greys 8 colors")
hist(VADeaths,col=brewer.pal(8,"Greens"),main="Greens 8 colors")
```

Code For Line Chart:

```
data(AirPassengers)
plot(AirPassengers,type="l")
```

Code for Bar Chart:

```
data("iris")
barplot(iris$Petal.Length) #Creating simple Bar Graph
barplot(iris$Sepal.Length,col = brewer.pal(3,"Set1"))
barplot(table(iris$Species,iris$Sepal.Length),col = brewer.pal(3,"Set1"))
```

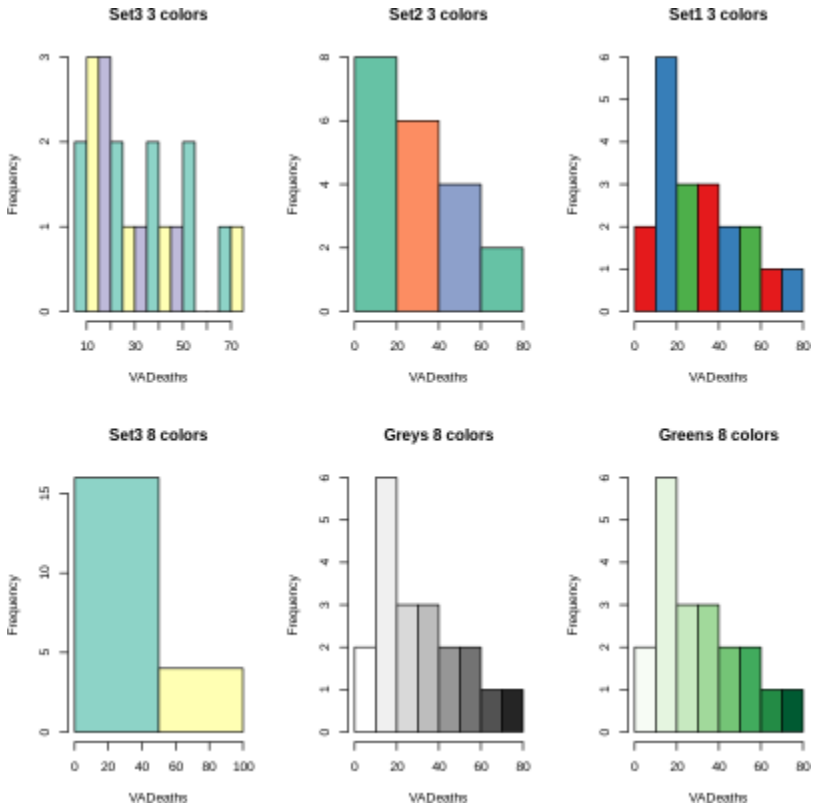
Code for Box plot:

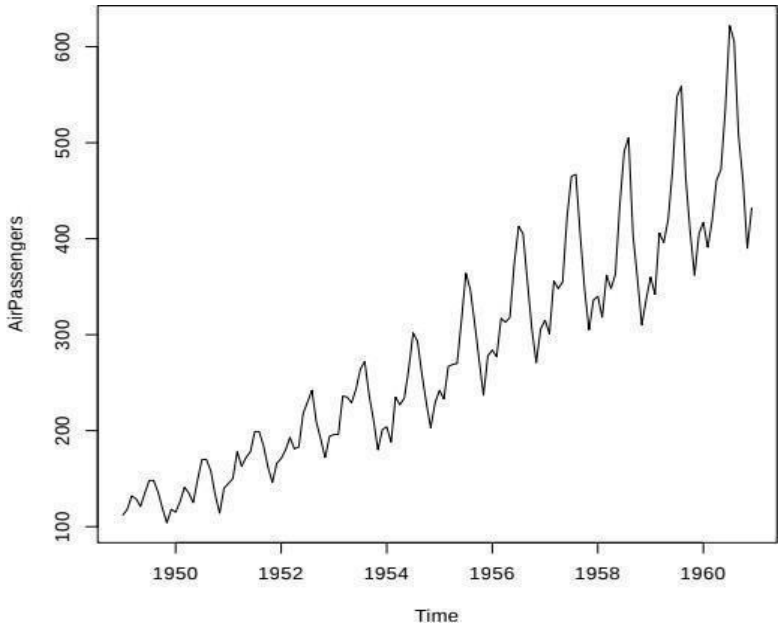
```
data(iris)
par(mfrow=c(2,2))
boxplot(iris$Sepal.Length,col="red")
boxplot(iris$Sepal.Length~iris$Species,col="red")
boxplot(iris$Sepal.Length~iris$Species,col=heat.colors(3))
boxplot(iris$Sepal.Length~iris$Species,col=topo.colors(3))
boxplot(iris$Petal.Length~iris$Species)
```

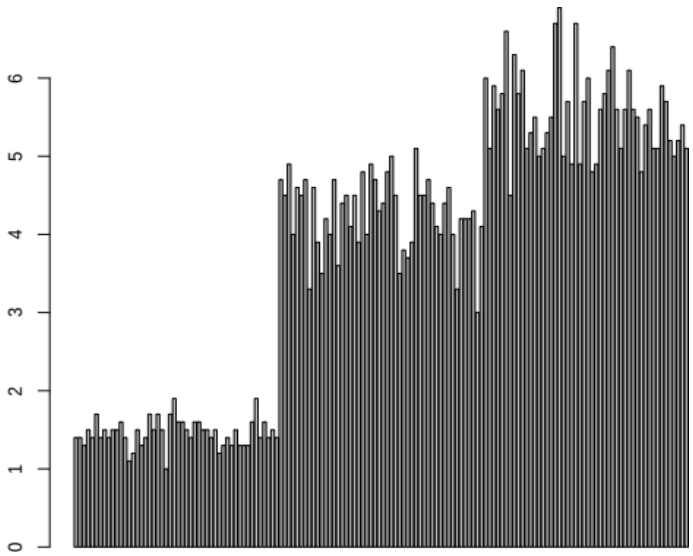
Code for Scatter Plot:

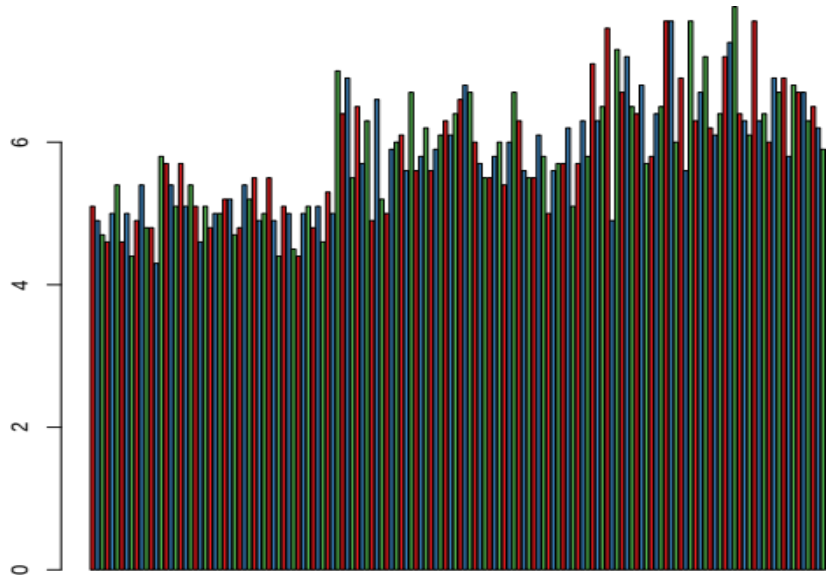
```
plot(x=iris$Petal.Length)
plot(x=iris$Petal.Length,y=iris$Species)
```

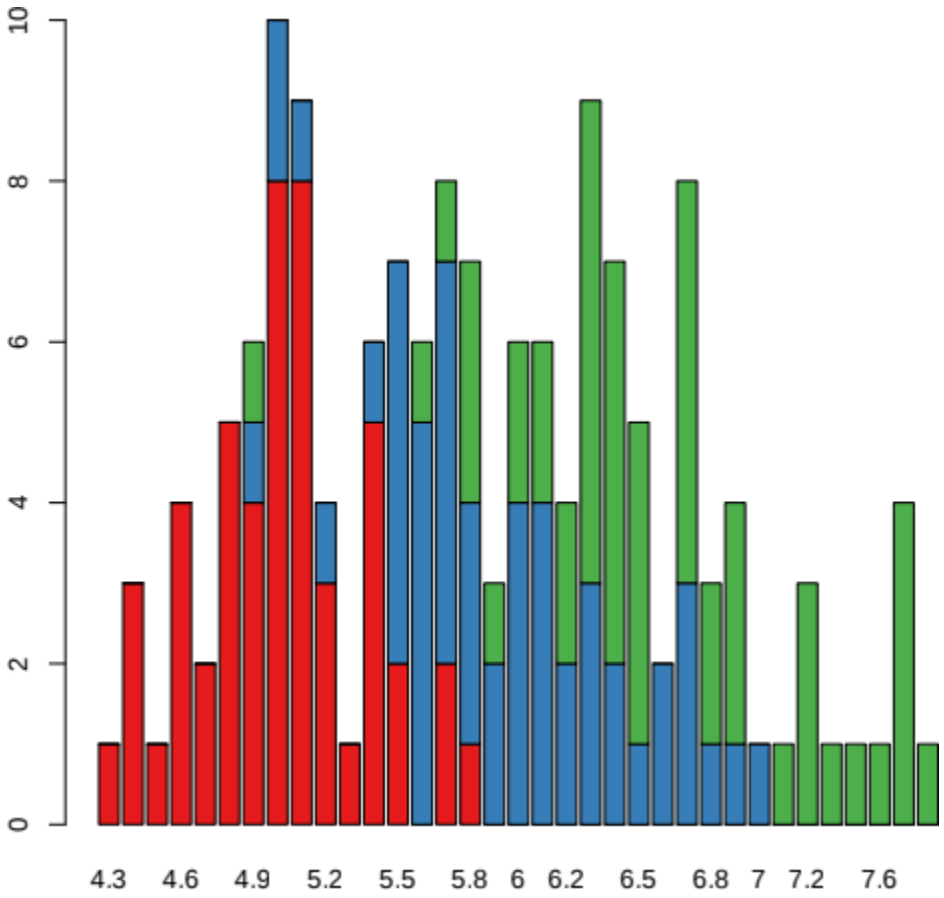
OUTPUT:

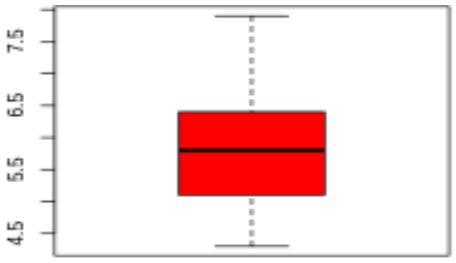


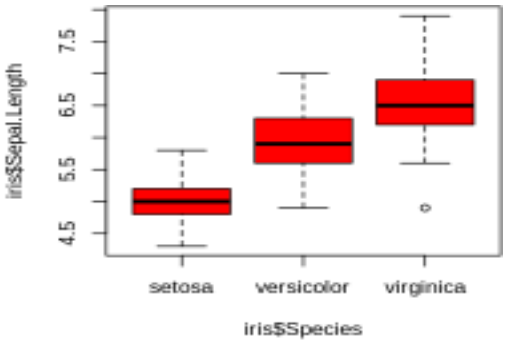

AirPassengers
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
Time

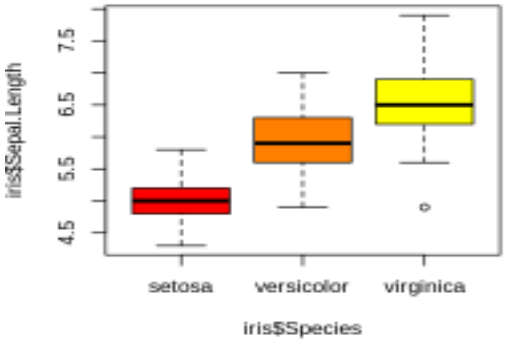

0
1
2
3
4
5
6

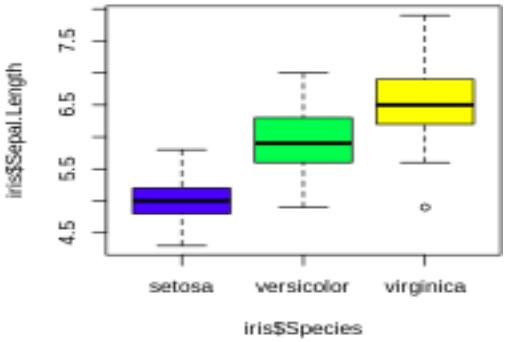

0
2
4
6

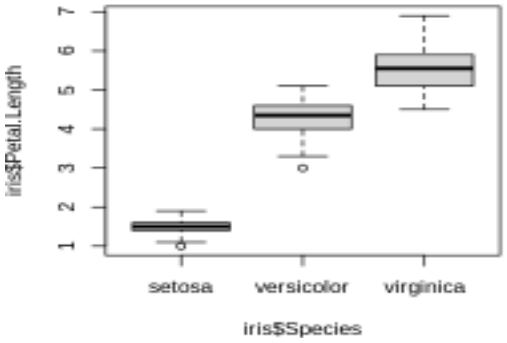

0
2
4
6
8
10
4.3
4.6
4.9
5.2
5.5
5.8
6
6.2
6.5
6.8
7
7.2
7.6

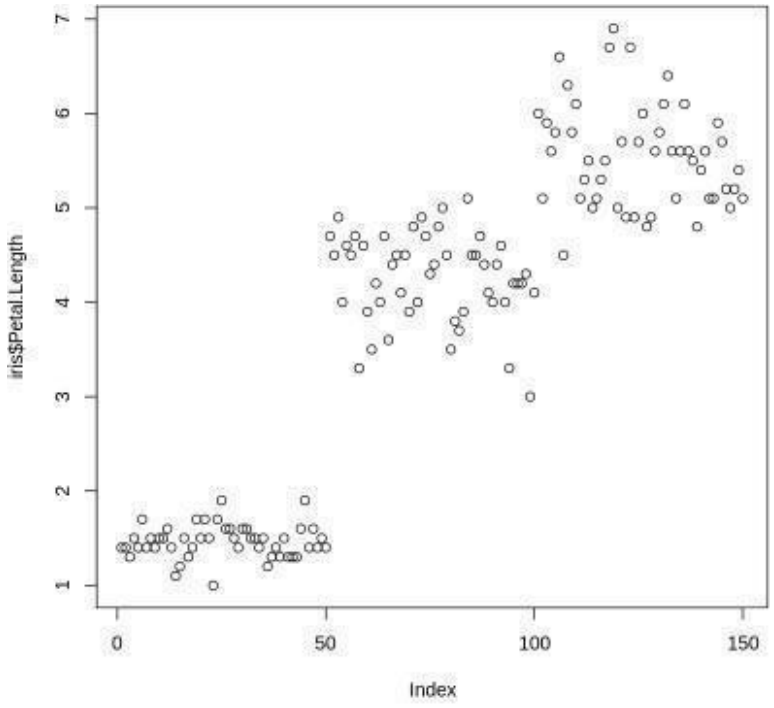

4.5
5.5
6.5
7.5

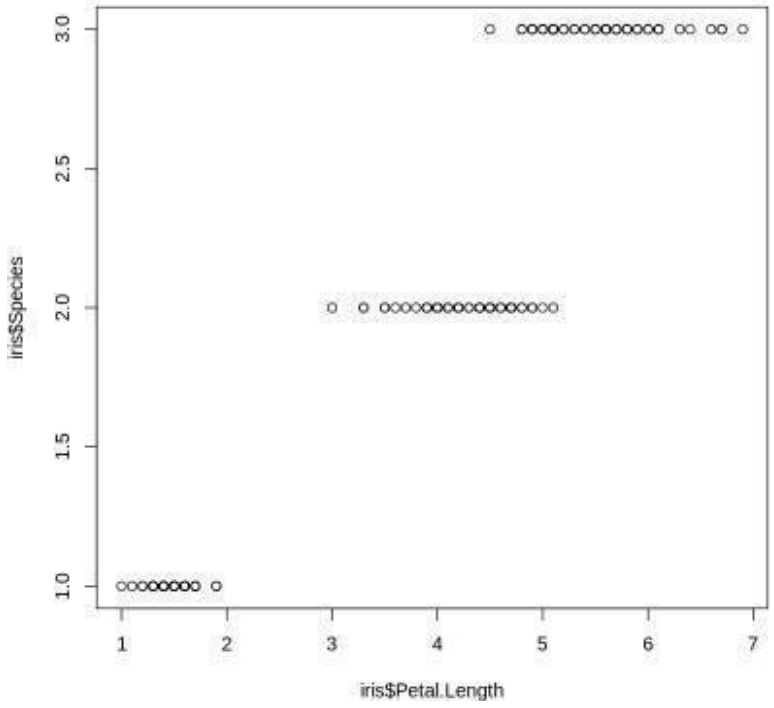

iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species


iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species


iris$Sepal.Length
4.5
5.5
6.5
7.5
setosa
versicolor
virginica
iris$Species


iris$Petal.Length
1
2
3
4
5
6
7
setosa
versicolor
virginica
iris$Species


iris$Petal.Length
Index
1
2
3
4
5
6
7
0
50
100
150


iris$Species
iris$Petal.Length
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5
6
7

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the R Program to implement Data Visualization to explore various charts has been executed and verified successfully.

EX NO: 6 DATE:	INSTALL AND CONFIGURE HADOOP

AIM:

To install and Configure Hadoop

SETUP:

Setting up Hadoop on Windows

PREREQUISITES:

- Windows 64 bit OS
- Java JDK
- Administrator Access

INSTALLATION:

1. Download Hadoop for windows from the official site

2. Extract the ZIP to your Chosen directory (Hadoop installation)

3. Set Environment Variables:

   'HADOOP_HOME'

   'JAVA_HOME'

   Edit the "Path" variable and add:

   '%HADOOP_HOME%\bin'

   '%HADOOP_HOME%\sbin'

   'JAVA_HOME%\bin'

4. Configure Hadoop:

   a. Open 'hadoop-env.cmd',set 'JAVA_HOME'.

   b. Create/edit 'core-site.xml' with:

```
xml
<property>
        <name>fs.defaultFS</name>
        <value>hdfs:localhost:9000</value>
</property>
```

c. Create/edit ‘hdfs-site.xml’ with:

```
xml
<property>
        <name>dfs.replication</name>
        <value>1</value>
</property>
```

d. create/edit ‘yarn-site.xml’ with:

```
xml
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>
<property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>HADOOP_HOME,YARN_HOME</value>
</property>
```

Formatting HDFS:

5. Open a command prompt and run :

```
hdfs namenode -format
```

Starting Hadoop Services:

6. In Command prompt run

```
cd %HADOOP_HOME%
```

start-dfs.cmd
start-yarn.cmd

Testing Hadoop:

7. Open a browser and visit ‘http://localhost:9870’ to see Hadoop namenode Web interface.

Running a Map Reduce Example:

8. To test, run a MapReduce example (replace jar file)
   Hadoop jar
   share/hadoop/mapreduce/hadoop*-mapreduce-examples.jar
   PI 16 1000

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the Install and Configure hadoop Successfully.

EX NO: 7 DATE:	MAP REDUCE USING HADOOP

AIM:

To implement an word count Program using Map Reduce

PREPARE:

1. Download MapReduceClient.jar
2. Download Input_file.txt

Place both files in "C://"

HADOOP OPERATIONS:

1. Open cmd in administrative mode and move to "C:/Hadoop-2.8.0/sbin" and start the cluster.

2. Start-all.cmd

3. Create an input directory in HDFS

   hadoop fs -mkdir /input_dir

4. Copy the input text file named input_file.txt in the input directory of HDFS

   hadoop fs -put C:input_file.txt/input_dir

   Verify the file input_file.txt is available in HDFS input directory.

   Hadoop fs -ls/input_dir/

5. Run MapReduceClient.jar and also provide input and output directories.

hadoop jar C:/MapReduceClient.jar wordcount/input_dir/output_dir

6. Verify the content for generated output file

   hadoop hdfs -cat /output_dir/

Some other useful commands:

7. To leave Safe mode:

   hadoop hdfsadmin -safemode leave

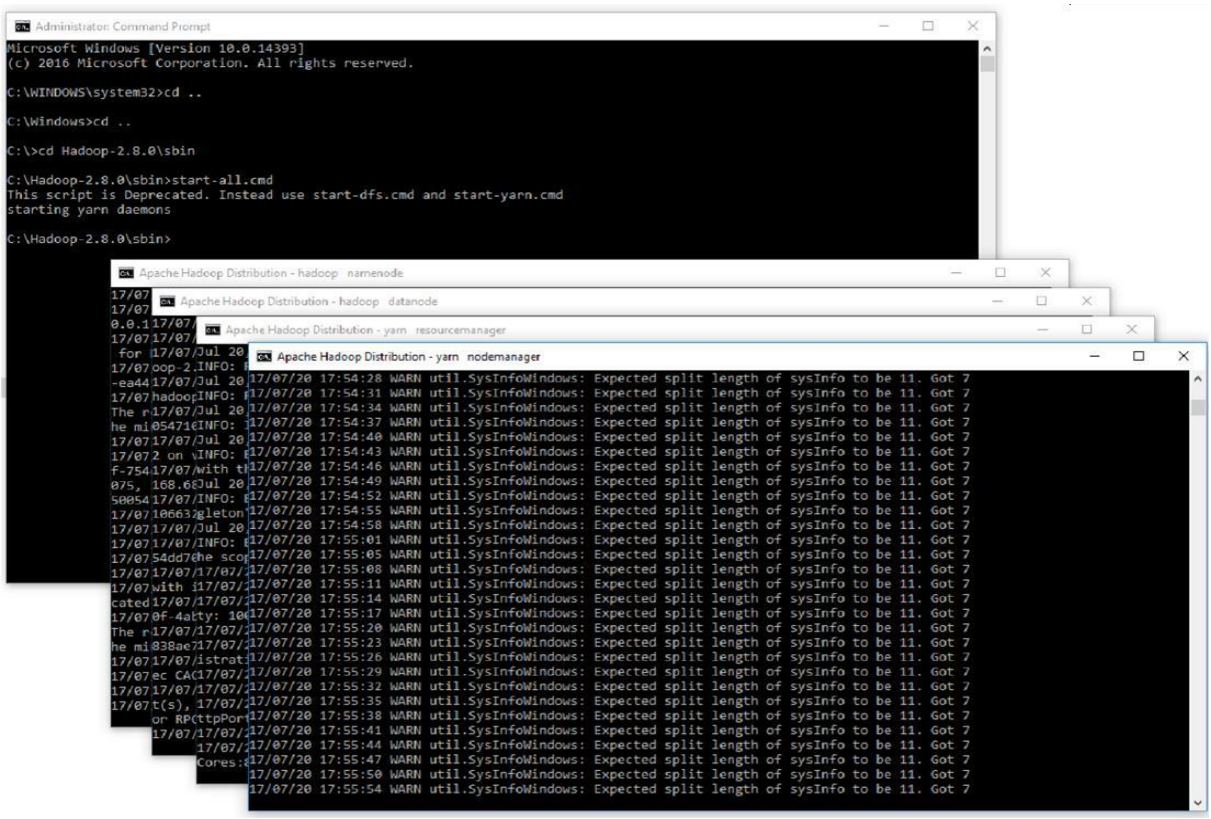
8. To delete file from HDFS directory

   hadoop fs -rm -r/input_dir/input_file.txt

9. To delete directory from HDFS directory

   hadoop fs -rm -r/input_dir

OUTPUT:


Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>cd ..
C:\Windows>cd ..
C:\>cd Hadoop-2.8.0\sbin
C:\Hadoop-2.8.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\Hadoop-2.8.0\sbin>
Apache Hadoop Distribution - hadoop namenode
Apache Hadoop Distribution - hadoop datanode
Apache Hadoop Distribution - yarn resourcemanager
Apache Hadoop Distribution - yarn nodemanager
17/07/20 17:54:28 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:31 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:34 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:37 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:40 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:43 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:46 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:49 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:52 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:55 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:58 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:01 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:05 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:08 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:11 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:14 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:17 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:20 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:23 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:26 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:29 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:32 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:35 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:38 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:41 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:44 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:47 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:50 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:54 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7

```
C:\>hadoop fs -ls /input_dir/
Found 1 items
-rw-r--r--   1 Muhammad.Bilal supergroup       1888 2017-07-20 18:31 /input_dir/input_file.txt

C:\>hadoop dfs -cat /input_dir/input_file.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
```

May June July August September

hadoop dfs -cat /output_dir/*

```
C:\>hadoop dfs -cat /output_dir/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
23      12
24      6
25      18
26      36
27      12
28      24
29      6
30      24
31      24
32      18
33      6
34      30
35      6
36      12
38      24
39      66
40      18
41      24
42      6
43      12
45      6

C:\>
```

hadoop dfsadmin –safemode leave

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the word count Program using MapReduce has been executed and verified Successfully.

EX NO: 8 DATE:	IMPLEMENTATION OF QUERIES USING MONGODB AND CASSANDRA

AIM:

To implement an application that stores big data in HBase /MongoDB/ Pig using Hadoop /R/Cassandra

PROGRAM:

Set up the Environment:

You should have hadoop,HBase,MongoDB,Pig,R, and Cassandra installed and Configured.

Sample dataset:

Let assume you have a simple dataset named "sample_data.csv" like this:

Name, Age, City
John,30,New York
Alice, 25, Los Angles
Bob, 35, Chicago

Hadoop MapReduce (Python)

Python map reduce script to process the dataset and store it in HBase:

```
#mapper
import sys
for line in sys.stdin:
    line = line.strip()
    fields = line.split(",")
```

```
        Name,age,city = fields
        print(f‘{name}\t{age}\t{city}’)

#reducer
import happybase
connection = happybase.connect(host = localhost,port = 9090)
table = connection.table('my_table')
for line in sys.stdin:
        name, age, city = line.strip().split("\t')
        table.put(name, ('info:age': age, 'info:city': city)
```

Pig Script:

A Pig script to perform some transformations and store the data in MongoDB:

```
        data = LOAD 'sample data.csv' USING PigStorage(") AS
(name:chararray, age:int, city:chararray);
        filtered data = FILTER data BY age >= 30;
        STORE filtered_data INTO
'mongodb://localhost:27017/mydb.mycollection' USING
org.apache.pig.backend.hadoop.executionengine. mapReducelayer.
PigMongoSt orage();
```

R Script:

An R script to analyze the data:

```
        library(rhbase)
        hbase <- HBaseSnew(host = "localhost", port = 9090)
        data <- hbasesget("my_table")
        print(data)
```

Cassandra (CQL):

You would create a Cassandra keyspace, define a table schema, and insert data using CQL commands. Below is a simplified example.

```
cql
CREATE KEYSPACE mykeyspace WITH replication = ('class':
'SimpleStrategy', 'replication_factor': 1};
 USE mykeyspace;
CREATE TABLE mytable (name TEXT PRIMARY KEY, age
INT, city TEXT);
INSERT INTO mytable (name, age, city) VALUES (John', 30,
'New York');
```

Running the Code:

Run the Python MapReduce script using Hadoop, execute the Pig script, run the R script, and execute the CQL commands in Cassandra.

.

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus working with Cassandra and MongoDB with Pig and Hadoop has been executed and verified Successfully.

EX NO: 9 DATE:	USING APACHE SPARK FOR DATA ANALYTICS

AIM:

To write a program to use apache spark for data Analytics

PROGRAM:

Creating Spark Session:

```
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.sql.types import Row
from datetime import datetime
```

Initializing Spark Session:

```
Sc = sparkContext()
spark=SparkSession.builder.appName("Python Spark SQL basic example").config("spark.some.config.option","some-value").getOrCreate()
```

Creation of Spark RDD:

```
srecord=sc.parallelize([

Row(roll_no=1,name="john",passed=True,marks={'Math':89,'Physics':87,'Chemistry':96},sports=['chess','football'],DoB=datetime(2012,5,1,12,1,5)),
```

```
row(roll_no=2,name="Vignesh",passed=False,marks={'Math':95,'Physics':6
6,'Chemistry':77},sports=['carrom','tennis'],DoB=datetime(2012,5,12,14,2,5)
),

Row(roll_no=3,name="Sidharth",passed=True,marks={'Math':95,'Physics':1
00,'Chemistry':95},sports=['football','kabadi'],DoB=datetime(2012,5,14,12,2
,5))
])
```

Creating a DataFrame:

```
srdf=srecord.toDF()
srdf.show()
```

Create Temporary View:

```
srdf.createOrReplaceTempView('records')
spark.sql("SELECT * FROM records").show()
re=spark.sql("SELECT * FROM records")
type(re)
```

Accessing Elements of a List or Dictionary within DataFrame:

```
spark.sql('SELECT roll_no,marks["Physics"],sports[1] FROM
records').show()
```

Usage of Where Clause:

```
spark.sql("SELECT * FROM records where passed= True").show()
spark.sql('SELECT * FROM records wheremarks["Chemistry"]<40').show()
```

Creating Global View:

```
srdf.createGlobalTempView('globalrecord')
spark.sql("SELECT  *  FROM  global_temp.globalrecord").show()
```

Dropping Columns From Data Frame:

```
srdf.columns
srdf=srdf.drop('passed')
```

Few more queries:

```
    spark.sql("SELECTround((marks.Physics+marks.Chemistry+marks.M
ath)/3)avg_marks FROM records").show()
    srdf=spark.sql("SELECT
*,round((marks.Physics+marks.Chemistry+marks.Math)/3)avg_marks
FROM records")
    srdf.show()
```

OUTPUT:

```
+-------+--------+------+--------------------+------------------+--------------------+
|roll_no|    name|passed|               marks|            sports|                DoB|
+-------+--------+------+--------------------+------------------+--------------------+
|      1|    john|  true|{Math -> 89, Chem...| [chess, football]|2012-05-01 12:01:05|
|      2| Vignesh| false|{Math -> 95, Chem...|  [carrom, tennis]|2012-05-12 14:02:05|
|      3|Sidharth|  true|{Math -> 95, Chem...|[football, kabadi]|2012-05-14 12:02:05|
+-------+--------+------+--------------------+------------------+--------------------+
```

```
+-------+---------------+---------+
|roll_no|marks[Physics]|sports[1]|
+-------+---------------+---------+
|      1|             87| football|
|      2|             66|   tennis|
|      3|            100|   kabadi|
+-------+---------------+---------+
```

```
+-------+--------+------+--------------------+------------------+--------------------+
|roll_no|    name|passed|               marks|            sports|                DoB|
+-------+--------+------+--------------------+------------------+--------------------+
|      1|    john|  true|{Math -> 89, Chem...| [chess, football]|2012-05-01 12:01:05|
|      3|Sidharth|  true|{Math -> 95, Chem...|[football, kabadi]|2012-05-14 12:02:05|
+-------+--------+------+--------------------+------------------+--------------------+
```

```
+-------+----+------+-----+------+---+
|roll_no|name|passed|marks|sports|DoB|
+-------+----+------+-----+------+---+
+-------+----+------+-----+------+---+
```

```
+-------+--------+------+--------------------+-----------------+-------------------+
|roll_no|    name|passed|               marks|           sports|                DoB|
+-------+--------+------+--------------------+-----------------+-------------------+
|      1|    john|  true|{Math -> 89, Chem...| [chess, football]|2012-05-01 12:01:05|
|      2| Vignesh| false|{Math -> 95, Chem...|  [carrom, tennis]|2012-05-12 14:02:05|
|      3|Sidharth|  true|{Math -> 95, Chem...|[football, kabadi]|2012-05-14 12:02:05|
+-------+--------+------+--------------------+-----------------+-------------------+
```

```
['roll_no', 'name', 'passed', 'marks', 'sports', 'DoB']
```

```
+---------+
|avg_marks|
+---------+
|     91.0|
|     79.0|
|     97.0|
+---------+
```

```
+-------+--------+------+--------------------+-----------------+-------------------+---------+
|roll_no|    name|passed|               marks|           sports|                DoB|avg_marks|
+-------+--------+------+--------------------+-----------------+-------------------+---------+
|      1|    john|  true|{Math -> 89, Chem...| [chess, football]|2012-05-01 12:01:05|     91.0|
|      2| Vignesh| false|{Math -> 95, Chem...|  [carrom, tennis]|2012-05-12 14:02:05|     79.0|
|      3|Sidharth|  true|{Math -> 95, Chem...|[football, kabadi]|2012-05-14 12:02:05|     97.0|
+-------+--------+------+--------------------+-----------------+-------------------+---------+
```

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the program to initialize the spark session has been executed and verified successfully.

EX NO: 10 DATE:	FILTER AND SORT SALES DATA USING PIG

AIM:

To filter and sort the sales data using Pig latin in Pig.

PREREQUISITES:

Running this program, we need the following requirements

Hadoop
Pig(Download it from Apache Pig website)

Before writing the Pig Script, Ensure that the hadoop is properly installed in your system.

Configure the environment variables (PIG_HOME, HADOOP_HOME, PATH)

PROGRAM:

1. **Prepare the sales dataset in the format sales.csv**

Transaction_id,customer,amount

1, Alice, 500
2, Bob, 1500
3, Charlie, 700
4, Diana, 2000
5, Eve, 1200

2. **Upload this file to HDFS:**

Upload this csv file to the hdfs by using the Commands:

hdfs dfs -mkdir /input

hdfs dfs -put sales.csv /input/

3. **Write the script named filter_and_sort and save this file in the extension .pig**

After uploading dataset to hdfs,write pig script in save it in the extension .pig

Pig Script:

Load the sales data from the HDFS:

```
sales_data = LOAD '/input/sales.csv' USING PigStorage(',') AS
(transaction_id:int, customer:chararray, amount:float);
```

Filter transactions with amounts greater than 1000:

```
filtered_data = FILTER sales_data BY amount > 1000;
```

Sort the filtered data by amount in descending order:

```
sorted_data = ORDER filtered_data BY amount DESC;
```

Store the result in HDFS:

```
STORE sorted_data INTO '/output/sorted_sales' USING PigStorage(',');
```

4. **Run the Pig Script:**

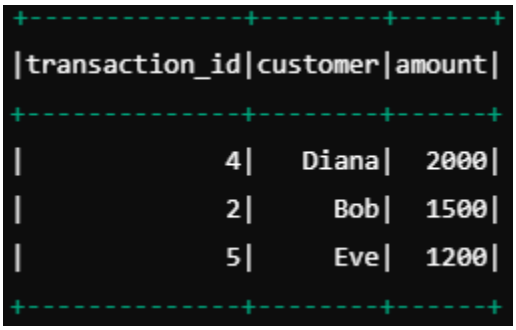
Run the above pig script by using the command in the CMD:

Verify Output directory:

hdfs dfs -ls /output/sorted_sales

View the output:

OUTPUT:



transaction_id	customer	amount
4	Diana	2000
2	Bob	1500
5	Eve	1200

Dept. of AI&DS

Description	Max Marks	Awarded
Aim	5	
Software/Tools Required & Algorithm	10	
Coding/ Programming & Execution	20	
Record	20	
Viva Voce	10	
Result	10	
TOTAL	75	

RESULT:

Thus the program to sort and filter the given dataset using Pig Latin has been executed and verified Successfully.