**Prac 1**

$ hdfs dfs -ls

$ hadoop fs -touchz Demo.txt

$ hadoop fs -ls

$ hadoop fs -copyFromLocal test.txt Demo.txt

$ hadoop fs -test -d Demo

$ echo s7

$ gedit test1.txt

$ gedit test2.txt

$ hadoop fs -touchz Demo.txt

$ hadoop fs -appendToFile test1.txt test2.txt Demo.txt

$ hadoop fs -cat Demo.txt

$ hadoop fs -usage test

$ hadoop fs -count -v /

$ hadoop fs -find

-name Demodirectory

/user/cloudera/Demodirectory

$ hadoop fs -help count

-count [q] [-h] [-v] [x]

**Prac 2**

A)

$ sudo -u hdfs hadoop fs -copyFromLocal wordcount input /wordcountinputhdfss

$ hadoop fs -cat /wordcount inputhdfss

$ hadoop fs -cat /wordcount outputdir3/part-r-00000

$ sudo -u hdfs hadoop jar wordcount input.jar WordCountDriver /wordcount inputhdfss /wordcountoutputdir3

B)

$ hdfs dfs -mkdir /expt4

$ sudo -u hdfs hdfs dfs -mkdir /expt4

$ sudo -u hdfs hdfs dfs -mkdir /expt4/input

$ sudo -u hdfs hdfs dfs -mkdir /expt4/input/M

$ sudo -u hdfs hdfs dfs -mkdir /expt4/input/N

$ hdfs dfs ls /expt4/input

$ sudo -u hdfs hdfs dfs -copyFromLocal m-matrix.txt /expt4/input/M

$ sudo -u hdfs hdfs dfs -copyFromLocal n-matrix.txt /expt4/input/N

$ hdfs dfs ls /expt4/input/M

$ javac MatrixMultiplication.java -cp $(hadoop classpath)

$ gedit Manifest.txt

$ jar -cfm MatrixMultiplication.jar Manifest.txt \*.class

$ sudo u hdfs hadoop jar MatrixMultiplication.jar

$ hdfs dfs -cat /expt4/interim/part-r-00000

$ hdfs dfs -cat /expt4/output/part-r-00000

**Prac 4**

$ sudo -uhive hive

hive> show databases;

hive> use demo;

hive> create table student (name string, rollno int, dept string)

> row format delimited

> fields terminated by ',';

hive>

hive> load data local inpath 'Desktop/student.txt' into table student;

hive> select from student;

Shreyas 33 mca

Nishant 61 mca

Sid 66 mca

Harsh 48 mca

Anirudh 63 mca

Manav 5 mca

Omkar 12 mca

Sen 27 mca

hive>

hive> describe student;

hive>

hive> show tables;

**Prac 6**

$ hdfs dfs -ls

$ hdfs dfs -put Downloads/web-Google.txt

$ hdfs dfs -ls

$ pyspark

def computeContribs (neighbors, rank):

for neighbor in neighbors: yield (neighbor, rank/len (neighbors))

links = sc.textFile('web-Google.txt')\

.map(lambda line: line.split())\

.map(lambda pages: (pages [0], pages [1]))\

.distinct()\

.groupByKey()\

.persist()

 ranks-links.map(lambda (page, neighbors): (page, 1.0))

for x in xrange(10): contribs-links\ .join(ranks)\ .flatMap(lambda (page, (neighbors, rank)): computeContribs (neighbors, rank)) ranks=contribs\ reduceByKey(lambda v1, v2: v1+v2)\ .map(lambda (page, contrib): (page, contrib\* 0.85 + 0.15))

for rank in ranks.collect(): print rank