VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

BIG DATA ANALYTICS

Submitted by

UDITI SINGH(1BM21CS260)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING

in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Feb-2024 to July-2024

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

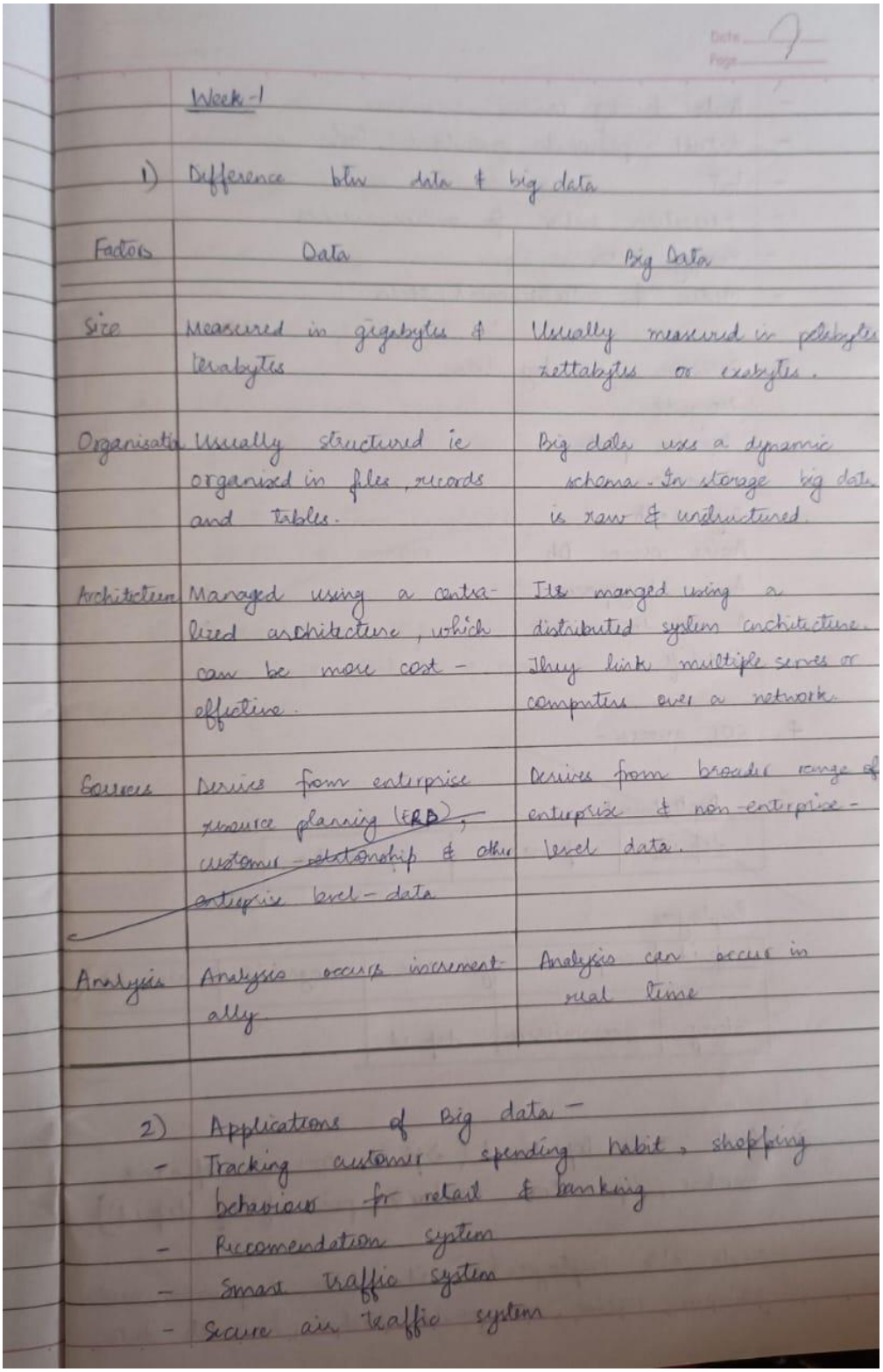
This is to certify that the Lab work entitled "LAB COURSE **BIG DATA ANALYTICS**" carried out by **UDITI SINGH (1BM21CS260),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024. The Lab report has been approved as it satisfies the academic requirements in respect of a **Big Data Analytics - (22CS6PEBDA)** work prescribed for the said degree.

Saritha A N
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

1.QUESTION AND ANSWER



Week-1

1) Difference btw data & big data

Factors	Data	Big Data
Size	Measured in gigabytes & terabytes	Usually measured in petabytes zettabytes or exabytes.
Organisation	Usually structured ie organised in files, records and tables.	Big data uses a dynamic schema. In storage big data is raw & unstructured.
Architecture	Managed using a centralized architecture, which can be more cost-effective.	It is manged using a distributed system architecture. They link multiple servers or computers over a network.
Sources	Derives from enterprise resource planning (ERP), customer relationship & other enterprise level-data	Derives from broader range of enterprise & non-enterprise-level data.
Analysis	Analysis occurs incrementally	Analysis can occur in real time

2) Applications of Big data -
- Tracking customer spending habit, shopping behaviour for retail & banking
- Recomendation system
- Smart traffic system
- Secure air traffic system

- Auto driving car
- Virtual personal assistant Tool
- IoT
- Education sector for online courses
- Energy sector
- Media & entertainment sector

3. Databases in big data
Mongodb
Apache spark
Cassandra
Dynamodb
Azure cosmos Db
Amazon keyspaces
Amazon documentdb
Redshift

4. SQL queries -

Department

dep_id	dep name	dep_location

Employees

emp_id	emp name	job name	manger_id	hire_date
salary	commission	dep_id		

a) Create Table Department (dept_id varchar, dep name varchar, dep location varchar2, primary key (dep id))

create table Employees (emp_id varchar, emp name varchar, job name varchar, manger_id varchar, hire date date,

salary int, commission int, dept_id varchar, primary key (dept_id, emp_id), foreign key dept_id extends Department))

b) select salary from employees

c) select avg(salary) from employees

d) select emp_name, emp_id from employees where job_name = 'Clerk' OR job_name = 'manager'

e) select * from employees where salary between 24000 to 50000

5) Installation steps -

sudo apt install -y mongodb

mongosh

2.MongoDB- CRUD Demonstration

Inserting into database

```
test> use Student
switched to db Student
Student> db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
```

Displaying inserted values

```
}
Student> db.Student.find()
[
  {
    _id: ObjectId('660a86053f257f0a2b66fd9b'),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId('660a86063f257f0a2b66fd9c'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('660a86063f257f0a2b66fd9d'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('660a86063f257f0a2b66fd9e'),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId('660a86083f257f0a2b66fd9f'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'abhinav@gmail.com'
  }
```

Updating values

```
Student> db.Student.update({RollNo:10},{$set:{email:"abhinav@gmail.com"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Student> db.Student.update({RollNo:11, Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Creating Customers database and inserting.

```
Student> db.createCollection("Customers");
{ ok: 1 }
Student> db.Customers.insert({cust_id:1,Balance:200, Type:"S"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660a87f33f257f0a2b66fda0') }
}
Student>

Student> db.Customers.insert({cust_id:1,Balance:1000, Type:"Z"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660a87f33f257f0a2b66fda1') }
}
Student>

Student> db.Customers.insert({cust_id:2,Balance:100, Type:"Z"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660a87f33f257f0a2b66fda2') }
}
Student>

Student> db.Customers.insert({cust_id:2,Balance:1000, Type:"C"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660a87f33f257f0a2b66fda3') }
}
Student>

Student> db.Customers.insert({cust_id:2,Balance:500, Type:"C"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660a87f33f257f0a2b66fda4') }
}
```

Updating.

```
Student> db.Customers.aggregate (
...
... {$group : { _id : "$cust_id",
...
... minAccBal :{$min:"$Balance"},
... maxAccBal :{$max:"$Balance"} }});
[
  { _id: 3, minAccBal: 500, maxAccBal: 500 },
  { _id: 2, minAccBal: 50, maxAccBal: 1000 },
  { _id: 1, minAccBal: 200, maxAccBal: 1000 }
]
Student> db.Customers.aggregate(
... {$match:{Type:"Z"}},
... {$group:{_id:"$cust_id",
... TotAccBal:{$sum:"$Balance"}}},
... {$match:{TotAccBal:{$gt:1200}}});
```

3. Perform the following DB operations using Cassandra.

```
bmscecse@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Students WITH REPLICATION={
   ... 'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES

students system_auth            system_schema system_views
system system_distributed system_traces system_virtual_schema

cqlsh> SELECT * FROM system.schema_keyspaces;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table
schema_keyspaces does not exist"
cqlsh> use Students;
cqlsh:students> create table Students_info(Roll_No int Primary key,StudName
text,DateOfJoining timestamp,last_exam_Percent double);
cqlsh:students> describe tables;

students_info

cqlsh:students> describe table students;
Table 'students' not found in keyspace 'students'
cqlsh:students> describe table students_info;

CREATE TABLE students.students_info (
        roll_no int PRIMARY KEY,
        dateofjoining timestamp,
        last_exam_percent double,
        studname text
) WITH additional_write_policy = '99p'
        AND bloom_filter_fp_chance = 0.01
        AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
        AND cdc = false
        AND comment = ''
        AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold':
'32', 'min_threshold': '4'}
        AND compression = {'chunk_length_in_kb': '16', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
        AND memtable = 'default'
        AND crc_check_chance = 1.0
```

```
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

cqlsh:students> Begin batch insert into Students_info(Roll_no,
StudName,DateOfJoining, last_exam_Percent) values(1,'Sadhana','2023-10-09', 98)
insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent)
values(2,'Rutu','2023-10-10', 97)
insert into Students_info(Roll_no, StudName,DateOfJoining,
last_exam_Percent) values(3,'Rachana','2023-10-10', 97.5)
insert into Students_info(Roll_no, StudName,DateOfJoining,
last_exam_Percent) values(4,'Charu','2023-10-06', 96.5) apply batch;
cqlsh:students> select * from students_info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 | Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |                97 |      Rutu
       4 | 2023-10-05 18:30:00.000000+0000 |                96.5 |  Charu
       3 | 2023-10-09 18:30:00.000000+0000 |                97.5 | Rachana

(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 | Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |                97 |      Rutu
       3 | 2023-10-09 18:30:00.000000+0000 |                97.5 | Rachana
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute
this query as it might involve data filtering and thus may have unpredictable
performance. If you want to execute this query despite the performance unpredictability,
use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
```

```
          4 | 2023-10-05 18:30:00.000000+0000 |                 96.5 | Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;

 roll_no | studname
---------+----------
          1 | Sadhana
          2 |        Rutu

(2 rows)
cqlsh:students> SELECT Roll_no as "USN" from Students_info;

 USN
-----
   1
   2
   4
   3

(4 rows)
cqlsh:students> update students_info set StudName='Shreya' where Roll_no=3;
cqlsh:students> select * from students_info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
          1 | 2023-10-08 18:30:00.000000+0000 |                98 | Sadhana
          2 | 2023-10-09 18:30:00.000000+0000 |                97 |      Rutu
          4 | 2023-10-05 18:30:00.000000+0000 |                96.5 |  Charu
          3 | 2023-10-09 18:30:00.000000+0000 |                97.5 |  Shreya

(4 rows)
cqlsh:students> update students_info set roll_no=8 where Roll_no=3;
InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY
KEY part roll_no found in SET part"
cqlsh:students> delete last_exam_percent from students_info where roll_no=2;
cqlsh:students> select * from students_info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
          1 | 2023-10-08 18:30:00.000000+0000 |                98 | Sadhana
          2 | 2023-10-09 18:30:00.000000+0000 |              null |    Rutu
          4 | 2023-10-05 18:30:00.000000+0000 |                96.5 |  Charu
          3 | 2023-10-09 18:30:00.000000+0000 |                97.5 |  Shreya
```

(4 rows)
cqlsh:students> delete from students_info where roll_no=2;
cqlsh:students> select * from students_info;

```
 roll_no | dateofjoining                  | last_exam_percent | studname
---------+--------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 | Sadhana
       4 | 2023-10-05 18:30:00.000000+0000 |              96.5 |  Charu
       3 | 2023-10-09 18:30:00.000000+0000 |              97.5 |  Shreya

(3 rows)
```

4.Employee Database

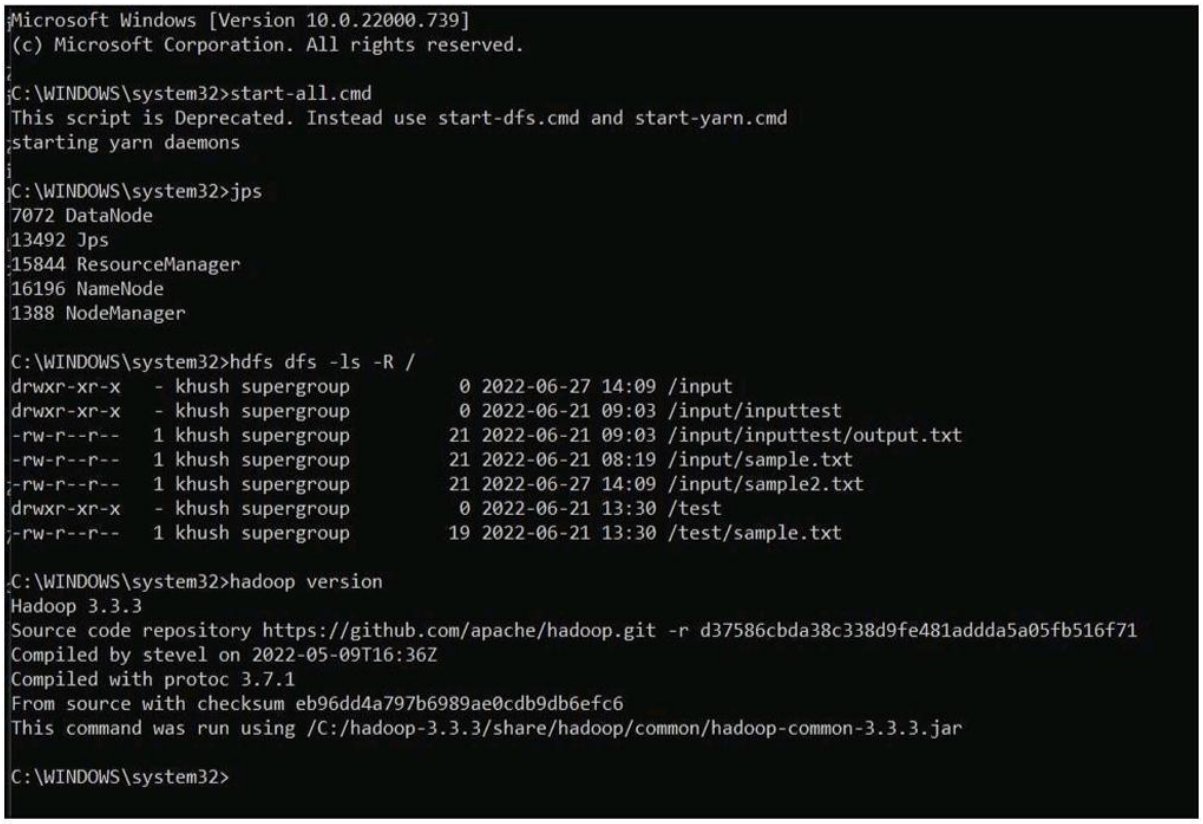
```
cqlsh> create keyspace Employee with replication ={
   ... 'class':'SimpleStrategy',
   ... 'replication_factor':1
   ... };
cqlsh> use Employee
   ... ;
cqlsh:employee> create table Employee_info(
            ... Name text,
            ... Emp_Id int PRIMARY KEY,
            ... Designation text,
            ... DateofJoining timestamp,
            ... Department text
            ... ,Salary int
            ... );
```

```
cqlsh:employee> begin batch insert into Employee_info(Name,Emp_Id,Designation,Da
teofJoining,Department,Salary) values('Raj',121,'Tester','2012-03-29','Testing',
40000) insert into Employee_info(Name,Emp_Id,Designation,DateofJoining,Departmen
t,Salary) values('Anand',122,'Developer','2013-02-27','SE',60000) insert into Em
ployee_info(Name,Emp_Id,Designation,DateofJoining,Department,Salary) values('Sha
nthi',123,'Developer','2014-04-12','SE',80000) insert into Employee_info(Name,Em
p_Id,Designation,DateofJoining,Department,Salary) values('Priya',124,'Analyst','
2012-05-29','Data',50000) apply batch;
cqlsh:employee> update Employee_info set Name='Rajesh' where Emp_Id=121;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name    |
 salary
--------+---------------------------------+------------+-------------+---------+
--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer | Shanthi |
  80000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |   Anand |
  60000
    121 | 2012-03-28 18:30:00.000000+0000 |    Testing |      Tester |  Rajesh |
  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |   Priya |
  50000

(4 rows)
```

5.Hadoop Installation



6.Hadoop Hdfs commands

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort.

Starting namenodes on [localhost]

Starting datanodes

Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]

Starting resourcemanager

Starting nodemanagers

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop dfs -mkdir /sadh

WARNING: Use of this script to execute dfs is deprecated.

WARNING: Attempting to execute replacement "hdfs dfs" instead.

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -mkdir /sadh

mkdir: `/sadh': File exists

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /

Found 1 items

drwxr-xr-x - hadoop supergroup 0 2024-05-13 14:27 /sadh

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/example/Welcome.txt /sadh/WC.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /sadh/WC.txt

hiiii

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt /home/hadoop/Desktop/example/WWC.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt /home/hadoop/Desktop/example/WWC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/example/Welcome.txt /sadh/WC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /sadh/WC.txt /sadh/WC2.txt /home/hadoop/Desktop/example/Merge.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -getfacl /sadh/

file: /sadh

owner: hadoop

group: supergroup

user::rwx

group::r-x

other::r-x

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /sadh /WC2.txt

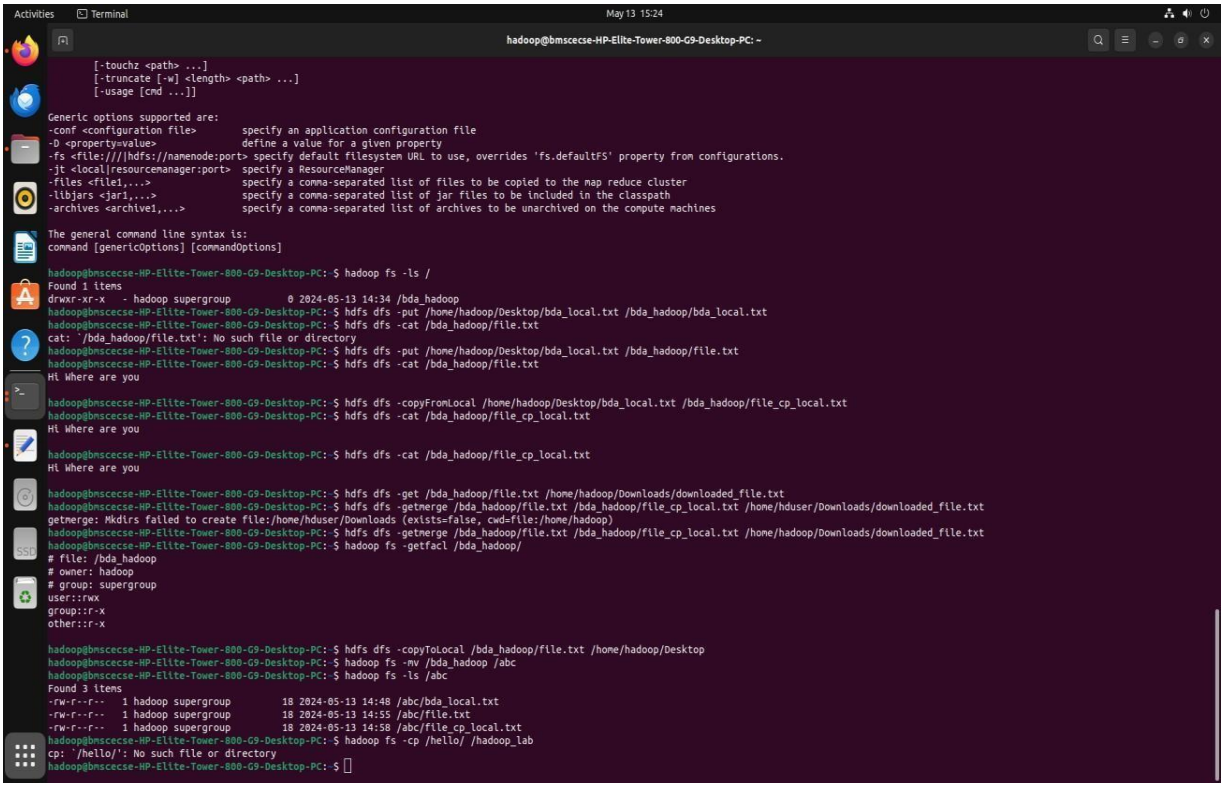
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh /WC2.txt

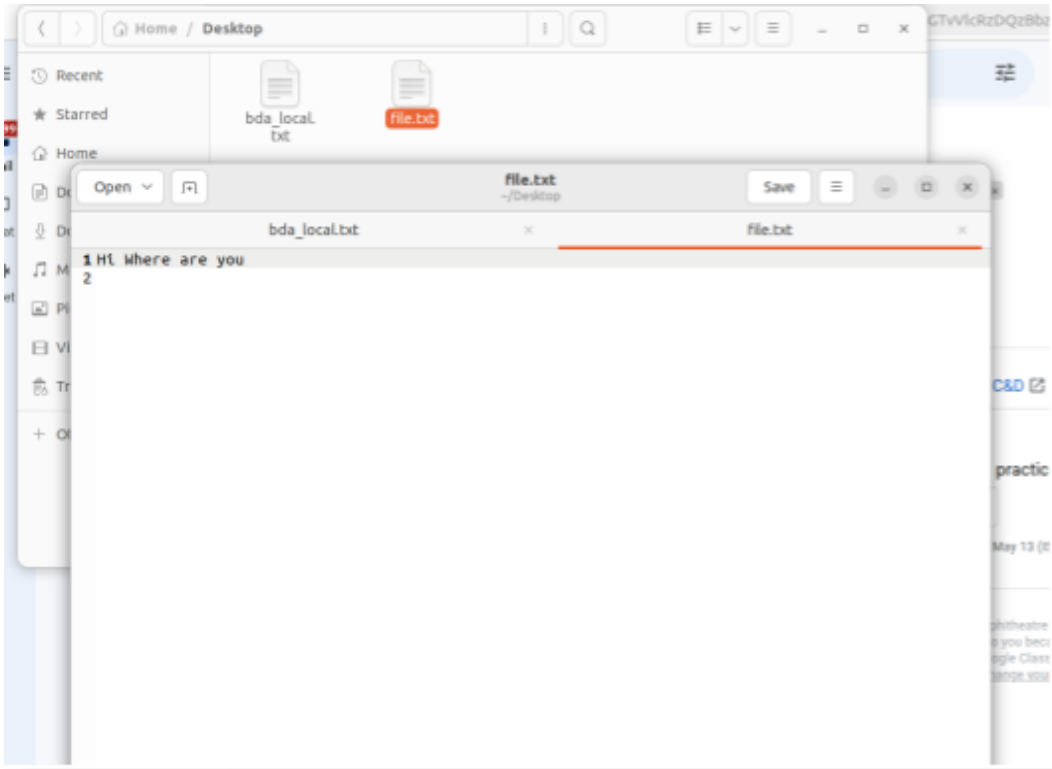
ls: `/sadh': No such file or directory

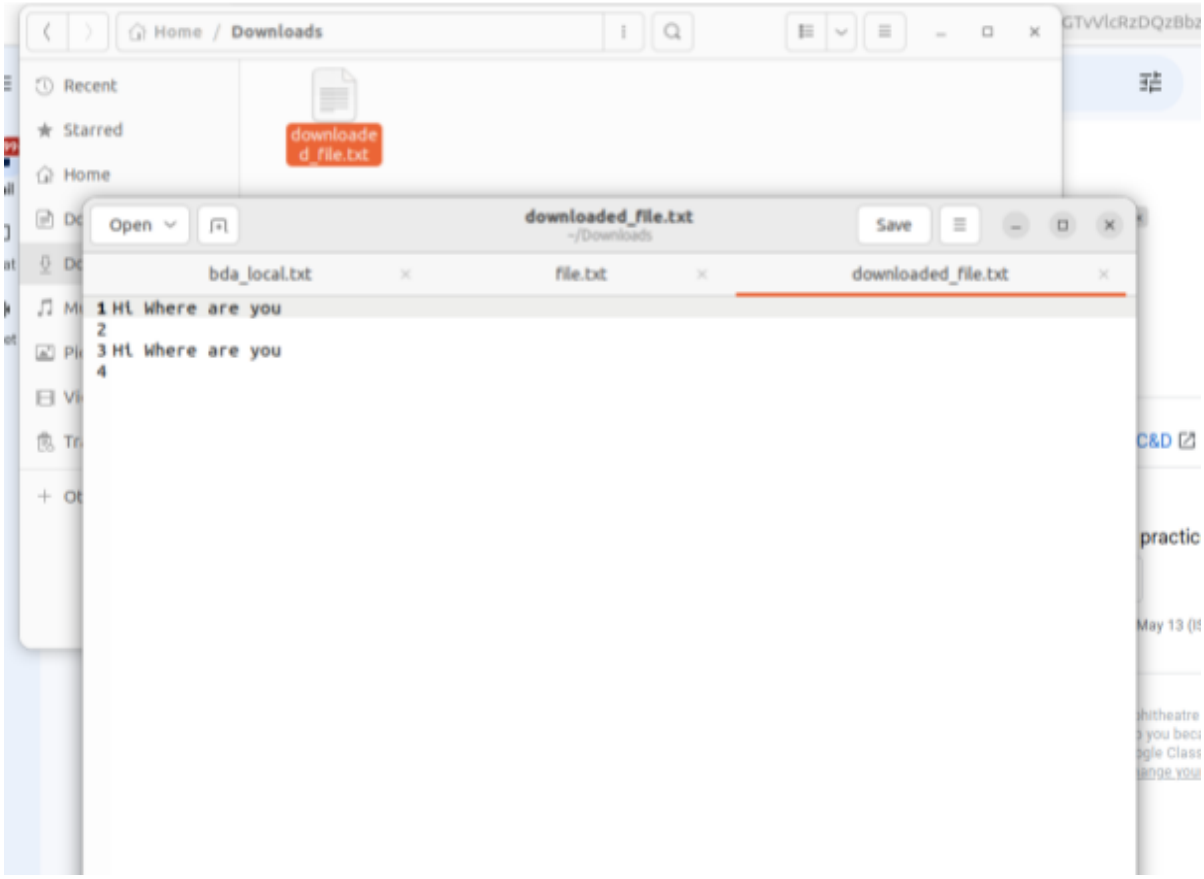
Found 2 items

-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 14:51 /WC2.txt/WC.txt
-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 15:03 /WC2.txt/WC2.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /WC2.txt/ /WC.txt

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
```




Home / Desktop
Recent
Starred
Home
bda_local.txt
file.txt
Open
file.txt
~/Desktop
Save
bda_local.txt
file.txt
1 Hi Where are you
2


Home / Downloads
Recent
Starred
Home
downloaded_file.txt
Open
downloaded_file.txt
~/Downloads
Save
bda_local.txt
file.txt
downloaded_file.txt
1 Hi Where are you
2
3 Hi Where are you
4

7. Implement WordCount Program on Hadoop framework

Mapper Code:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text,
IntWritable> {
public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter rep) throws IOException
{
String line = value.toString();
for (String word : line.split(" "))
{
if (word.length() > 0)
{
output.collect(new Text(word), new IntWritable(1));
} } } }
```

Reducer Code:

```
//  Importing  libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.mapred.MapReduceBase;
import  org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WCReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
// Reduce function
public void reduce(Text key, Iterator<IntWritable> value,
OutputCollector<Text, IntWritable> output,
Reporter rep) throws IOException
{
int count = 0;
// Counting the frequency of each words
while (value.hasNext())
{
IntWritable i = value.next();
count += i.get();
```

```
}
output.collect(key, new IntWritable(count));
} }
```

Driver Code: You have to copy paste this program into the WCDriver Java Class file.

```
//  Importing  libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WCDriver extends Configured implements Tool {
public int run(String args[]) throws IOException
{
if (args.length < 2)
{
System.out.println("Please give valid inputs");
return -1;
}
```

```
JobConf conf = new JobConf(WCDriver.class);
FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));
conf.setMapperClass(WCMapper.class);
conf.setReducerClass(WCReducer.class);
conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(IntWritable.class);
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
JobClient.runJob(conf);
return 0;
}
// Main Method
public static void main(String args[]) throws Exception
{
int exitCode = ToolRunner.run(new WCDriver(), args);
System.out.println(exitCode);
}
}
```

8.From the following link extract the weather data https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all Create a Map Reduce program to

a) find average temperature for each year from NCDC data set.

AverageDriver

```
package temp;

import  org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver {

public static void main(String[] args) throws Exception {

if (args.length != 2) {

System.err.println("Please Enter the input and output parameters");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(AverageDriver.class);

job.setJobName("Max temperature");

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
job.setMapperClass(AverageMapper.class);
job.setReducerClass(AverageReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

AverageMapper

```
package temp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
public static final int MISSING = 9999;
public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
int temperature;
String line = value.toString();
String year = line.substring(15,
19); if (line.charAt(87) == '+') {
temperature = Integer.parseInt(line.substring(88, 92));
} else {
```

```
temperature = Integer.parseInt(line.substring(87, 92));
}
String quality = line.substring(92, 93);
if (temperature != 9999 && quality.matches("[01459]"))
context.write(new Text(year), new IntWritable(temperature));
}
}
```

AverageReducer

```
package temp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class AverageReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
int max_temp = 0;
int count = 0;
for (IntWritable value : values) {
max_temp += value.get();
count++;
}
context.write(key, new IntWritable(max_temp / count));
}}
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\avgtemp.jar temp.AverageDriver /input_dir/temp.txt /avgtemp_outputdir
2021-05-15 14:52:50,635 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-15 14:52:51,005 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-15 14:52:51,111 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621060230696_0005
2021-05-15 14:52:51,735 INFO input.FileInputFormat: Total input files to process : 1
2021-05-15 14:52:52,751 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621060230696_0005
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-15 14:52:53,237 INFO conf.Configuration: resource-types.xml not found
2021-05-15 14:52:53,238 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-15 14:52:53,312 INFO impl.YarnClientImpl: Submitted application application_1621060230696_0005
2021-05-15 14:52:53,352 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621060230696_0005/
2021-05-15 14:52:53,353 INFO mapreduce.Job: Running job: job_1621060230696_0005
2021-05-15 14:53:06,640 INFO mapreduce.Job: Job job_1621060230696_0005 running in uber mode : false
2021-05-15 14:53:06,643 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-15 14:53:12,758 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-15 14:53:19,860 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-15 14:53:25,967 INFO mapreduce.Job: Job job_1621060230696_0005 completed successfully
2021-05-15 14:53:26,096 INFO mapreduce.Job: Counters: 54
      File System Counters
            FILE: Number of bytes read=72210
            FILE: Number of bytes written=674341
            FILE: Number of read operations=0
            FILE: Number of large read operations=0
            FILE: Number of write operations=0
            HDFS: Number of bytes read=894860
            HDFS: Number of bytes written=8
            HDFS: Number of read operations=8
            HDFS: Number of large read operations=0
            HDFS: Number of write operations=2
            HDFS: Number of bytes read erasure-coded=0
      Job Counters
            Launched map tasks=1
            Launched reduce tasks=1
            Data-local map tasks=1
            Total time spent by all maps in occupied slots (ms)=3782
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /avgtemp_outputdir
Found 2 items
-rw-r--r--   1 Anusree supergroup          0 2021-05-15 14:53 /avgtemp_outputdir/_SUCCESS
-rw-r--r--   1 Anusree supergroup          8 2021-05-15 14:53 /avgtemp_outputdir/part-r-00000

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /avgtemp_outputdir/part-r-00000
1901    46

C:\hadoop-3.3.0\sbin>
```

b) Find the mean max temperature for every month

MeanMaxDriver.class

```
package meanmax;

import  org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MeanMaxDriver {
public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.err.println("Please Enter the input and output parameters");
System.exit(-1);
}
Job job = new Job();
job.setJarByClass(MeanMaxDriver.class);
job.setJobName("Max temperature");
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(MeanMaxMapper.class);
job.setReducerClass(MeanMaxReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

MeanMaxMapper.class

```
package meanmax;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
public static final int MISSING = 9999;
public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
int temperature;
String line = value.toString();
String month = line.substring(19, 21);
if (line.charAt(87) == '+') {
temperature = Integer.parseInt(line.substring(88, 92));
} else {
temperature = Integer.parseInt(line.substring(87, 92));
}
String quality = line.substring(92, 93);
if (temperature != 9999 && quality.matches("[01459]"))
context.write(new Text(month), new IntWritable(temperature));
}
}
```

MeanMaxReducer.class

```
package meanmax;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Reducer;
public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
int max_temp = 0;
int total_temp = 0;
int count = 0;
int days = 0;
for (IntWritable value : values) {
int temp = value.get();
if (temp > max_temp)
max_temp = temp;
count++;
if (count == 3) {
total_temp += max_temp;
max_temp = 0;
count = 0;
days++;
}
}
context.write(key, new IntWritable(total_temp / days));
}
}
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\meanmax.jar meanmax.MeanMaxDriver /input_dir/temp.txt /meanmax_output
2021-05-21 20:28:05,250 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-21 20:28:06,662 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-21 20:28:06,916 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621608943095_0001
2021-05-21 20:28:08,426 INFO input.FileInputFormat: Total input files to process : 1
2021-05-21 20:28:09,107 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621608943095_0001
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-21 20:28:10,029 INFO conf.Configuration: resource-types.xml not found
2021-05-21 20:28:10,030 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-21 20:28:10,676 INFO impl.YarnClientImpl: Submitted application application_1621608943095_0001
2021-05-21 20:28:11,005 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621608943095_0001/
2021-05-21 20:28:11,006 INFO mapreduce.Job: Running job: job_1621608943095_0001
2021-05-21 20:28:29,385 INFO mapreduce.Job: Job job_1621608943095_0001 running in uber mode : false
2021-05-21 20:28:29,389 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-21 20:28:40,664 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-21 20:28:50,832 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-21 20:28:58,965 INFO mapreduce.Job: Job job_1621608943095_0001 completed successfully
2021-05-21 20:28:59,178 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=59082
                FILE: Number of bytes written=648091
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=894860
                HDFS: Number of bytes written=74
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8077
                Total time spent by all reduces in occupied slots (ms)=7511
                Total time spent by all map tasks (ms)=8077
                Total time spent by all reduce tasks (ms)=7511
                Total vcore-milliseconds taken by all map tasks=8077
                Total vcore-milliseconds taken by all reduce tasks=7511
                Total megabyte-milliseconds taken by all map tasks=8270848
                Total megabyte-milliseconds taken by all reduce tasks=7691264
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /meanmax_output/*
01      4
02      0
03      7
04      44
05      100
06      168
07      219
08      198
09      141
10      100
11      19
12      3

C:\hadoop-3.3.0\sbin>
```

9. For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

Driver-TopN.class

```
package samples.topn;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class TopN {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();

if (otherArgs.length != 2) {

System.err.println("Usage: TopN <in> <out>");

System.exit(2);

}

Job job = Job.getInstance(conf);
```

```
job.setJobName("Top N");
job.setJarByClass(TopN.class);
job.setMapperClass(TopNMapper.class);
job.setReducerClass(TopNReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
private static final IntWritable one = new IntWritable(1);
private Text word = new Text();
private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";
public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
StringTokenizer itr = new StringTokenizer(cleanLine);
while (itr.hasMoreTokens()) {
this.word.set(itr.nextToken().trim());
context.write(this.word, one);
}
}
}
```

```
}
```

TopNCombiner.class

```
package samples.topn;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values)
sum += val.get();
context.write(key, new IntWritable(sum));
}
}
```

TopNMapper.class

```
package samples.topn;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
```

```
private static final IntWritable one = new IntWritable(1);
private Text word = new Text();
private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";
public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
StringTokenizer itr = new StringTokenizer(cleanLine);
while (itr.hasMoreTokens()) {
this.word.set(itr.nextToken().trim());
context.write(this.word, one);
}
}
}
```

TopNReducer.class

```
package samples.topn;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import utils.MiscUtils;
public class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
private Map<Text, IntWritable> countMap = new HashMap<>();
```

```
public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values)
sum += val.get();
this.countMap.put(new Text(key), new IntWritable(sum));
}
protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
throws IOException, InterruptedException {
Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);
int counter = 0;
for (Text key : sortedMap.keySet()) {
if (counter++ == 20)
break;
context.write(key, sortedMap.get(key));
}
}
}
```

```
C:\hadoop-3.3.0\sbin>jps
11072 DataNode
20528 Jps
5620 ResourceManager
15532 NodeManager
6140 NameNode

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - Anusree supergroup          0 2021-05-08 19:46 /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -copyFromLocal C:\input.txt /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /input_dir
Found 1 items
-rw-r--r--   1 Anusree supergroup         36 2021-05-08 19:48 /input_dir/input.txt

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input.txt
hello
world
hello
hadoop
bye
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\sort.jar samples.topn.TopN /input_dir/input.txt /output_dir
2021-05-08 19:54:54,582 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-08 19:54:55,291 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1620483374279_0001
2021-05-08 19:54:55,821 INFO input.FileInputFormat: Total input files to process : 1
2021-05-08 19:54:56,261 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1620483374279_0001
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-08 19:54:56,843 INFO conf.Configuration: resource-types.xml not found
2021-05-08 19:54:56,843 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-08 19:54:57,387 INFO impl.YarnClientImpl: Submitted application application_1620483374279_0001
2021-05-08 19:54:57,507 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1620483374279_0001/
2021-05-08 19:54:57,508 INFO mapreduce.Job: Running job: job_1620483374279_0001
2021-05-08 19:55:13,792 INFO mapreduce.Job: Job job_1620483374279_0001 running in uber mode : false
2021-05-08 19:55:13,794 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-08 19:55:20,020 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-08 19:55:27,116 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-08 19:55:33,199 INFO mapreduce.Job: Job job_1620483374279_0001 completed successfully
2021-05-08 19:55:33,334 INFO mapreduce.Job: Counters: 54
       File System Counters
              FILE: Number of bytes read=65
              FILE: Number of bytes written=530397
              FILE: Number of read operations=0
              FILE: Number of large read operations=0
              FILE: Number of write operations=0
              HDFS: Number of bytes read=142
              HDFS: Number of bytes written=31
              HDFS: Number of read operations=8
              HDFS: Number of large read operations=0
              HDFS: Number of write operations=2
              HDFS: Number of bytes read erasure-coded=0
```