**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

***Submitted by***

**VRISHANK J VASIST (1BM21CS246)**

***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 **VRISHANK J VASIST (1BM21CS246),** who is 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.

Name of the Lab-Incharge               **Dr. Jyothi S Nayak**

Designation Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | **MongoDB- CRUD Demonstration**  **( Practice and Self Study)** | **5** |
| **2** | **Perform the following DB operations using Cassandra-Student Database** | **8** |
| **3** | **Cassandra-Employee Database** | **12** |
| **4** | **Implement WordCount Program on Hadoop framework** | **13** |
| **5** | **HDFS Commands** | **17** |
| **6** | **Create a Map Reduce program to a) find average temperature for each year from NCDC data set. b) find the mean max temperature for every month** | **21** |
| **7** | **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** | **29** |

**Course Outcome**

|  |  |
| --- | --- |
| C0 | Apply the concepts of NoSQL, Hadoop, Spark for a given task |
| C1 | Analyse data analytic techniques for a given problem . |
| C2 | Analyse data analytic techniques for a given problem . |

1. **MongoDB- CRUD Demonstration( Practice and Self Study)**

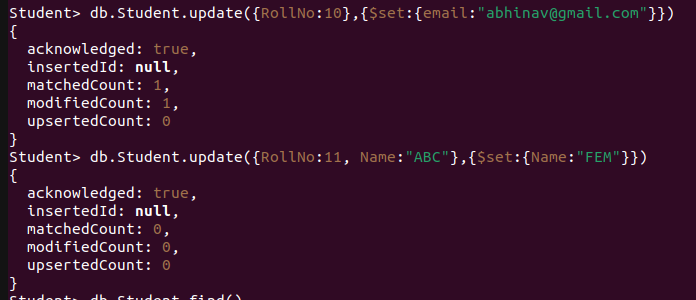
Inserting into database

****

Displaying inserted values



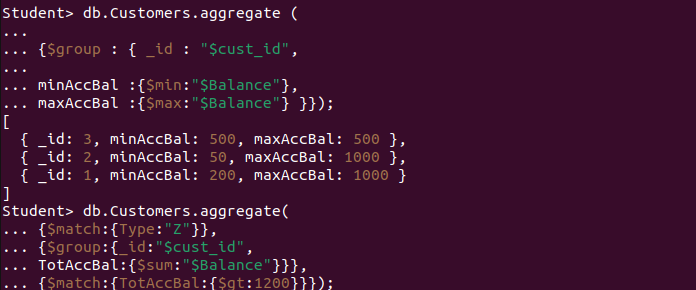
Updating values



Creating Customers database and inserting.



Updating.



1. **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

---------+---------------------------------+-------------------+----------

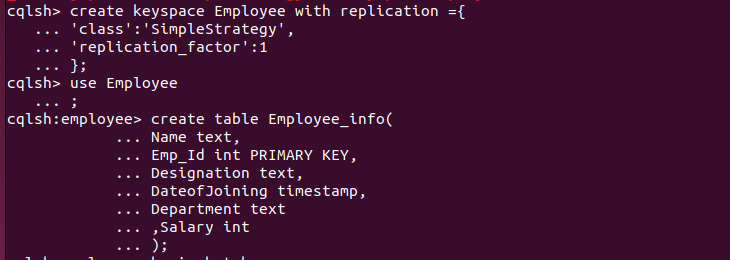
1 | 2023-10-08 18:30:00.000000+0000 | 98 | Sadhana

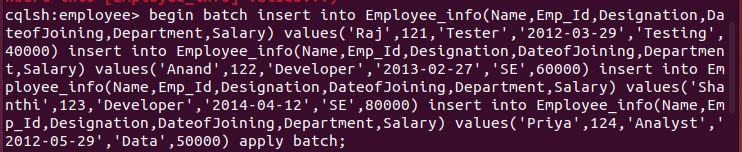
4 | 2023-10-05 18:30:00.000000+0000 | 96.5 | Charu

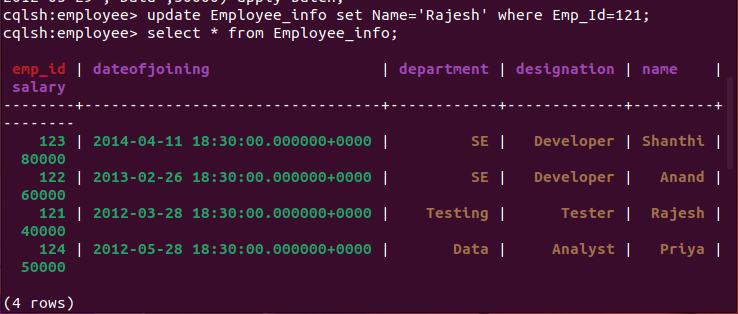
3 | 2023-10-09 18:30:00.000000+0000 | 97.5 | Shreya

(3 rows)

1. **Employee Database**







1. **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

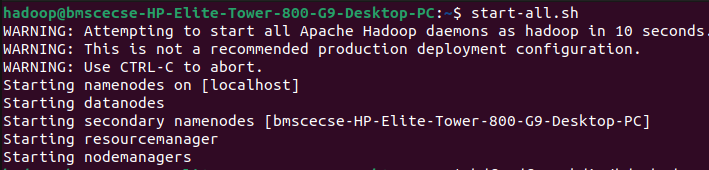
ls: `/sadh': No such file or directory

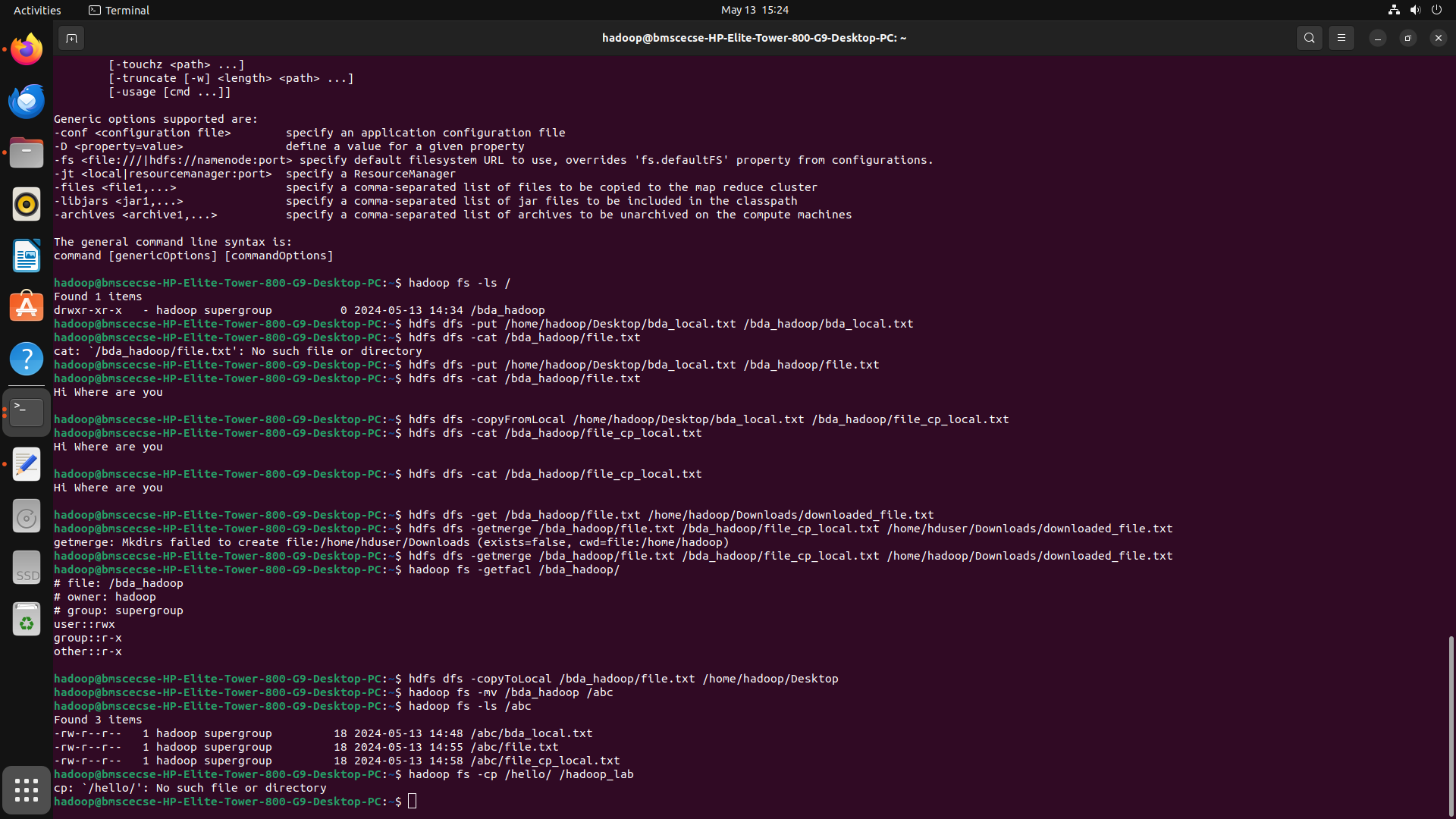
Found 2 items

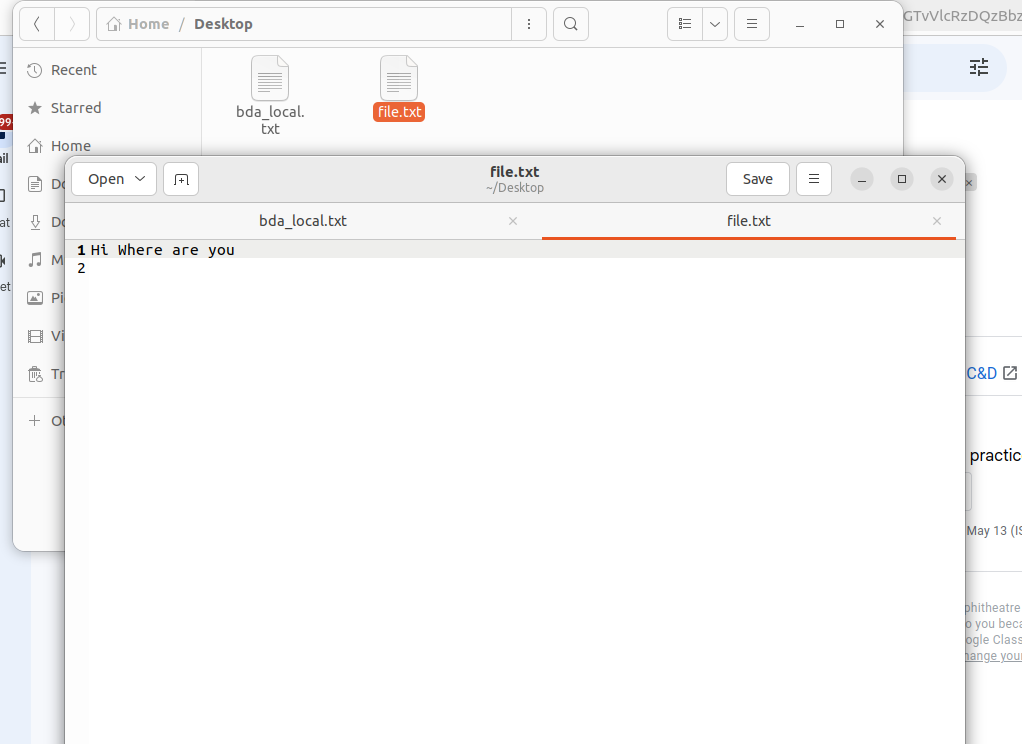
-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 14:51 /WC2.txt/WC.txt

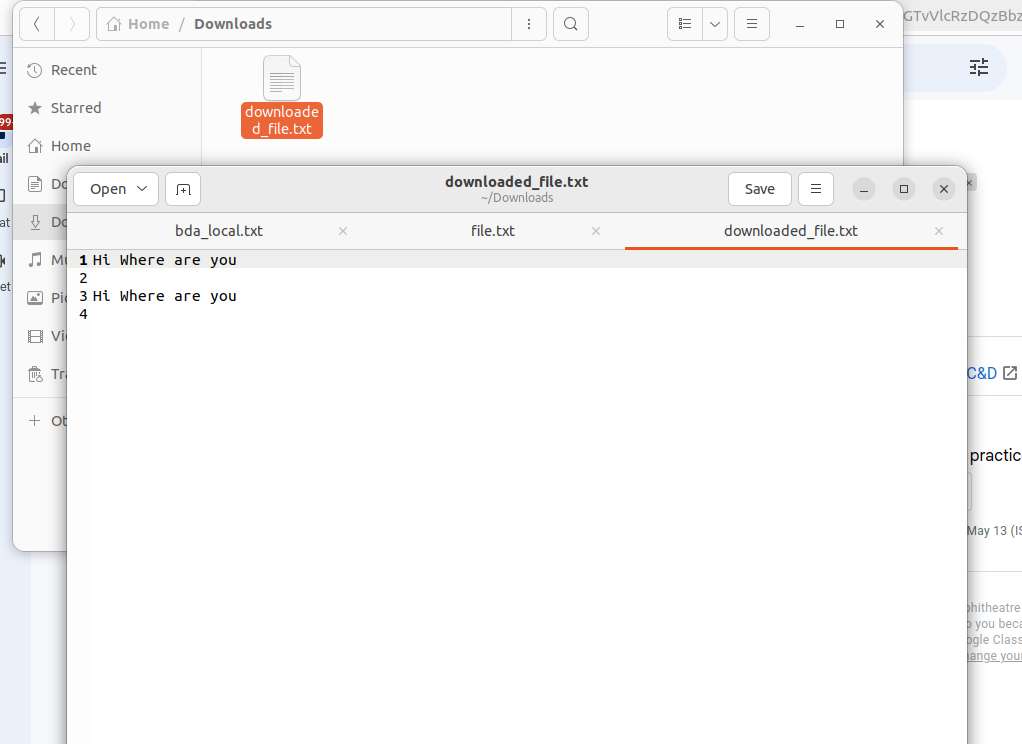
-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 15:03 /WC2.txt/WC2.txt

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









1. **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);

}

}

1. **From the following link extract the weather data** [**https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all**](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) {

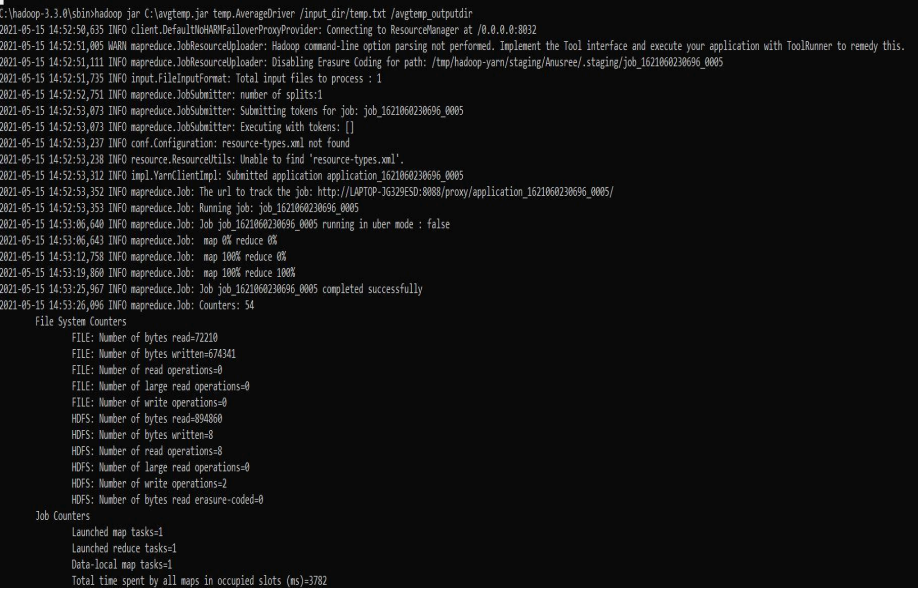
max\_temp += value.get();

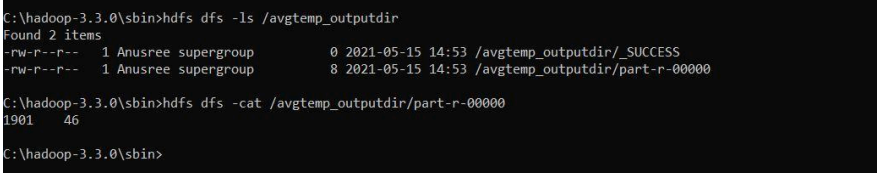
count++;

}

context.write(key, new IntWritable(max\_temp / count));

}}





**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++;

}

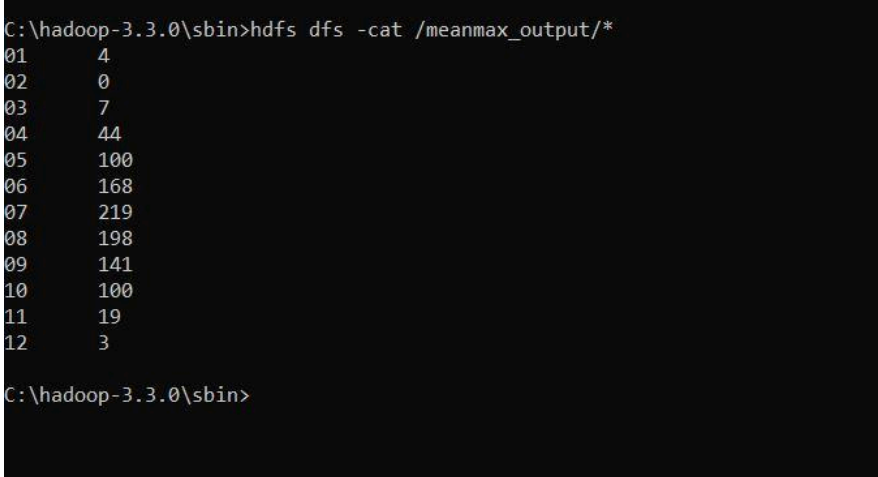
}

context.write(key, new IntWritable(total\_temp / days));

}

}





1. **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)

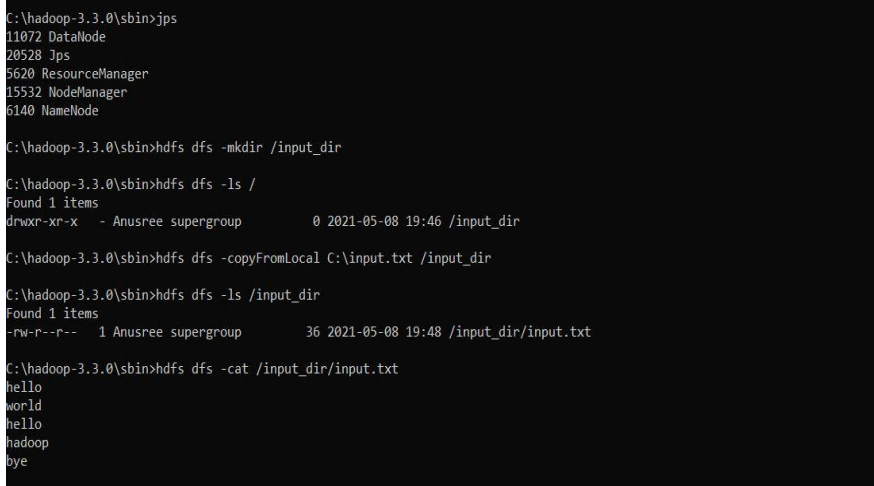
break;

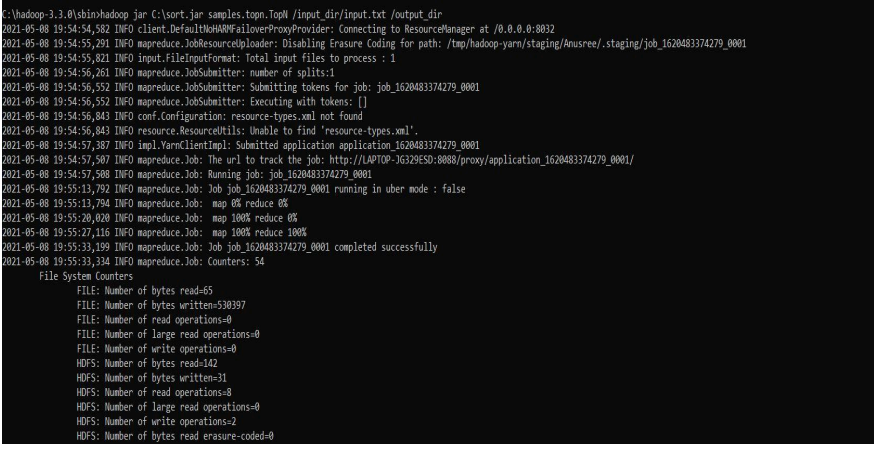
context.write(key, sortedMap.get(key));

}

}

}

****

****