**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



# LAB REPORT on

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

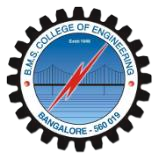
# Ishika Singhal (1BM19CS064)

***in partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

# COMPUTER SCIENCE AND ENGINEERING



# B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019 May-2022 to July-2022

**(Autonomous Institution under VTU)**

**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 “**BIG DATA ANALYTICS**” carried out by **Ishika Singhal(1BM19CS064),** 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 2022. The Lab report has been approved as it satisfies the academic requirements in respect of Big data analytics - (20CS6PEBDA) work prescribed for the said degree.

Name of the Lab-In charge ANTARA ROY CHOUDHURY

Designation Assistant Professor

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

# Index Sheet

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| **1** | **MongoDB- CRUD Demonstration** | **5** |
| **2** | **Cassandra Lab Program 1: - Student Database** | **16** |
| **3** | **Cassandra Lab Program 2: - Library Database** | **20** |
| **4** | **Hadoop Installation** | **22** |
| **5** | **Hadoop Commands** | **23** |
| **6** | **Hadoop Programs: Word Count** | **26** |
| **7** | **Hadoop Programs: Top N** | **32** |
| **8** | **Hadoop Programs: Average Temperature** | **37** |
| **9** | **Hadoop Programs: Join** | **44** |
| **10** | **Scala Programs: Word Count** | **53** |
| **11** | **Scala Programs: Word Count greater than 4** | **54** |

**Course Outcome**

|  |  |
| --- | --- |
| CO1 | Apply the concept of NoSQL, Hadoop or Spark for a given task |
| CO2 | Analyze the Big Data and obtain insight using data analytics mechanisms. |
| CO3 | Design and implement Big data applications by applying NoSQL, Hadoop or Spark |

**LAB 1:**

**I.CREATE DATABASE IN MONGODB. > use khushilDB**

**switched to db khushilDB**

**db;**

khushilDB

**show dbs;**

**admin 0.000GB**

**config 0.000GB**

**local 0.000GB**

## II. CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

1. To create a collection by the name “Student”. Let us take a look at the collection list prior to the creation of the new collection

“Student”.

## db.createCollection(“Student”); => *sql equivalent* CREATE TABLE STUDENT(…);

**{ "ok" : 1 }**

2.To drop a collection by the name “Student”.

## db.Student.drop(); 3.Create a collection by the name “Students” and store the following data in it. db.Student.insert({\_id:1,StudName:"MichelleJacintha",Gra de:"VII",Hobbies:"InternetSurfing"});

WriteResult({ "nInserted" : 1 })

4.Insert the document for “AryanDavid” in to the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from “Skating” to “Chess”. ) Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

## db.Student.update({\_id:3,StudName:"AryanDavid",Grade:" VII"},{$set:{Hobbies:"Skating"}},{upsert:true});

**WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified"**

**: 0, "\_id" : 3 })**

5.FIND METHOD

A. To search for documents from the “Students” collection based on certain search criteria.

## db.Student.find({StudName:"AryanDavid"}); ({cond..},{columns.. column:1, columnname:0} )

**{ "\_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid",**

**"Hobbies" : "Skating" }**

1. To display only the StudName and Grade from all the documents of the Students collection. The identifier\_id should be suppressed and NOT displayed. **db.Student.find({},{StudName:1,Grade:1,\_id:0});**

**{ "StudName" : "MichelleJacintha", "Grade" : "VII" }**

**{ "Grade" : "VII", "StudName" : "AryanDavid" }**

1. To find those documents where the Grade is set to ‘VII’

## db.Student.find({Grade:{$eq:'VII'}}).pretty();

**{**

**"\_id" : 1,**

**"StudName" : "MichelleJacintha",**

**"Grade" : "VII",**

**"Hobbies" : "InternetSurfing"**

**}**

**{**

**"\_id" : 3,**

**"Grade" : "VII",**

**"StudName" : "AryanDavid",**

**"Hobbies" : "Skating"**

**}**

D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

## db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();

**{**

**"\_id" : 3,**

**"Grade" : "VII",**

**"StudName" : "AryanDavid",**

**"Hobbies" : "Skating"**

**}**

E. To find documents from the Students collection where the StudName begins with “M”.

## db.Student.find({StudName:/^M/}).pretty();

**{**

**"\_id" : 1,**

**"StudName" : "MichelleJacintha",**

**"Grade" : "VII",**

**"Hobbies" : "InternetSurfing"**

**}**

F. To find documents from the Students collection where the StudNamehas an “e” in any position.

## db.Student.find({StudName:/e/}).pretty();

**{**

**"\_id" : 1,**

**"StudName" : "MichelleJacintha",**

**"Grade" : "VII",**

**"Hobbies" : "InternetSurfing"**

**}**

G. To find the number of documents in the Students collection.

## db.Student.count();

**2**

H. To sort the documents from the Students collection in the descending order of StudName.

## db.Student.find().sort({StudName:-1}).pretty();

**{**

**"\_id" : 1,**

**"StudName" : "MichelleJacintha",**

**"Grade" : "VII",**

**"Hobbies" : "InternetSurfing"**

**}**

**{**

**"\_id" : 3,**

**"Grade" : "VII",**

**"StudName" : "AryanDavid",**

**"Hobbies" : "Skating"**

**}**

## III. Import data from a CSV file

Given a CSV file “sample.txt” in the D:drive, import the file into the MongoDB collection, “SampleJSON”. The collection is in the database “test”.

**mongoimport --db Student --collection airlines --type csv – headerline --file /home/hduser/Desktop/airline.csv**

## IV. Export data to a CSV file

This command used at the command prompt exports MongoDB JSON documents from “Customers” collection in the “test” database into a CSV file “Output.txt” in the D:drive.

## mongoexport --host localhost --db Student --collection airlines --csv --out /home/hduser/Desktop/output.txt – fields “Year”,”Quarter”

1. **Save Method :**

**Save() method will insert a new document, if the document with the \_id does not exist. If it exists it will replace the exisiting document.**

db.Student.save({StudName:”Vamsi”, Grade:”VI”})

WriteResult({ "nInserted" : 1 })

1. **Add a new field to existing Document:**

db.Student.update({\_id:ObjectId("625695cc7d129fb98b44c8a1")}, {$set:{Location:"Network"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

1. **Remove the field in an existing Document** db.Student.update({\_id:ObjectId("625695cc7d129fb98b44c8a1")},

{$unset:{Location:”Network”}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

## VIII. Finding Document based on search criteria suppressing few fields

db.Student.find({\_id:1},{StudName:1,Grade:1,\_id:0});

{ "StudName" : "MichelleJacintha", "Grade" : "VII" }

## To find those documents where the Grade is not set to ‘VII’ db.Student.find({Grade:{$ne:'VII'}}).pretty();

{

"\_id" : ObjectId("625695cc7d129fb98b44c8a1"),

"StudName" : "Vamsi",

"Grade" : "VI"

}

**To find documents from the Students collection where the StudName ends with s.**

db.Student.find({StudName:/s$/}).pretty();

{

"\_id" : 1,

"StudName" : "MichelleJacintha",

"Grade" : "VII",

"Hobbies" : "InternetSurfing"

}

## IX. to set a particular field value to NULL

db.Student.update({\_id:3},{$set:{Location:null}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

## X. Count the number of documents in Student Collections

db.Student.count()

3

**XI. Count the number of documents in Student Collections with grade :VII**

db.Student.count({Grade:”VII”})

## 2 retrieve first 3 documents

db.Student.find({Grade:”VII”}).limit(1).pretty();

{

"\_id" : 1,

"StudName" : "MichelleJacintha",

"Grade" : "VII",

"Hobbies" : "InternetSurfing"

}

## Sort the document in Ascending order

db.Student.find().sort({StudName:1}).pretty();

{

"\_id" : 3,

"Grade" : "VII",

"StudName" : "AryanDavid",

"Hobbies" : "Skating",

"Location" : null

}

{

"\_id" : 1,

"StudName" : "MichelleJacintha",

"Grade" : "VII",

"Hobbies" : "InternetSurfing"

}

{

"\_id" : ObjectId("625695cc7d129fb98b44c8a1"),

"StudName" : "Vamsi",

"Grade" : "VI"

}

**Note:**  **for desending order :** db.Students.find().sort({StudName:-

1}).pretty();

## to Skip the 1st two documents from the Students Collections

db.Student.find().skip(2).pretty()

{

"\_id" : ObjectId("625695cc7d129fb98b44c8a1"),

"StudName" : "Vamsi",

"Grade" : "VI"

}

XII. Create a collection by name “food” and add to each document add a “fruits” array

db.food.insert( { \_id:1, fruits:['grapes','mango','apple'] } ) db.food.insert( { \_id:2, fruits:['grapes','mango','cherry'] } ) db.food.insert( { \_id:3, fruits:['banana','mango'] } )

{ "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }

{ "\_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }

{ "\_id" : 3, "fruits" : [ "banana", "mango" ] }

## To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and

**“apple”.**

db.food.find ( {fruits: ['grapes','mango','apple'] } ). pretty();

{ "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }

**To find in “fruits” array having “mango” in the first index position.**

db.food.find ( {“fruits.1'':grapes'} )

**To find those documents from the “food” collection where the size of the array is two.**

db.food.find ( {“fruits”: {$size:2}} )

{ "\_id" : 3, "fruits" : [ "banana", "mango" ] }

## To find the document with a particular id and display the first two elements from the array “fruits”

db.food.find({\_id:1},{“fruits”:{$slice:2}})

{ "\_id" : 1, "fruits" : [ "grapes", "mango" ] }

## To find all the documets from the food collection which have elements mango and grapes in the array “fruits”

db.food.find({fruits:{$all:[“mango”,”grapes”]}})

{ "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }

{ "\_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }

## update on Array: using particular id replace the element present in the 1st index position of the fruits array with apple

db.food.update({\_id:3},{$set:{'fruits.1':'apple'}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

insert new key value pairs in the fruits array

db.food.update({\_id:2},{$push:{price:{grapes:80,mango:200,cherr y:100}}})

{ "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }

{ "\_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ], "price" : [ {

"grapes" : 80, "mango" : 200, "cherry" : 100 } ] }

{ "\_id" : 3, "fruits" : [ "banana", "apple" ] }

Note: perform query operations using - pop, addToSet, pullAll and pull

**LAB 2:**

Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

create keyspace "Employee" with replication = {'class':'SimpleStrategy','replication\_factor':1}; cqlsh> use Employee;

1. Create a column family by name Employee-Info with attributes

Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining,

Salary, Dept\_Name

create table Employee\_Info(Emp\_id int PRIMARY KEY,Emp\_name text,Date\_of\_Joining timestamp,Salary float,Dept\_Name text) ;

1. Insert the values into the table in batch

cqlsh:employee> begin batch

... insert into

Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N

ame) values(1,'Khushil','2021-04-23',50000,'CSE')

... insert into

Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N

ame) values(2,'Tarun','2020-06-21',10000,'ISE')

... insert into

Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N

ame) values(3,'Suresh','2011-02-12',30000,'ECE')

... insert into

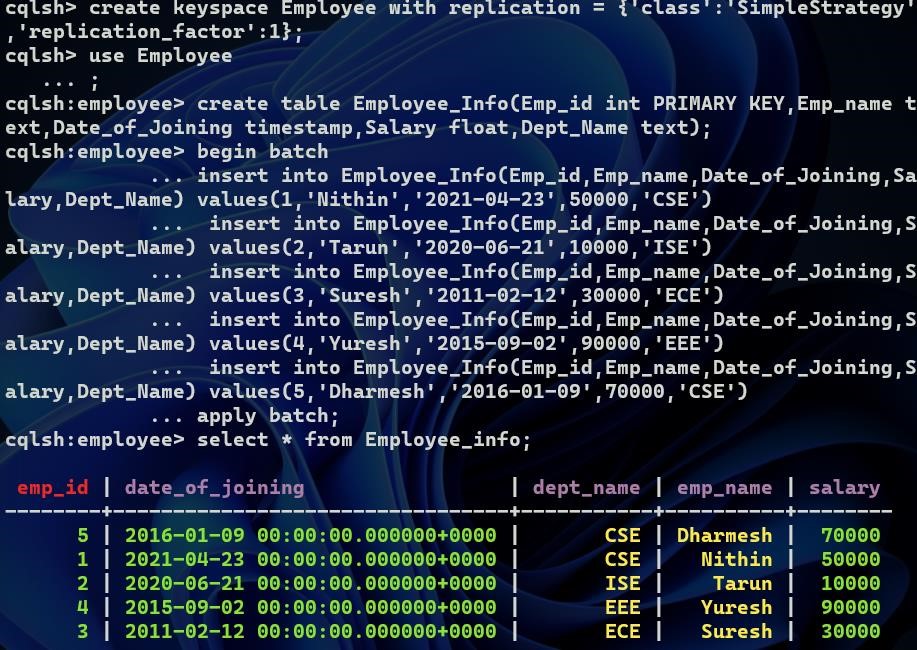
Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N ame) values(4,'Yuresh','2015-09-02',90000,'EEE')

... insert into

Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N

ame) values(5,'Dharmesh','2016-01-09',70000,'CSE')

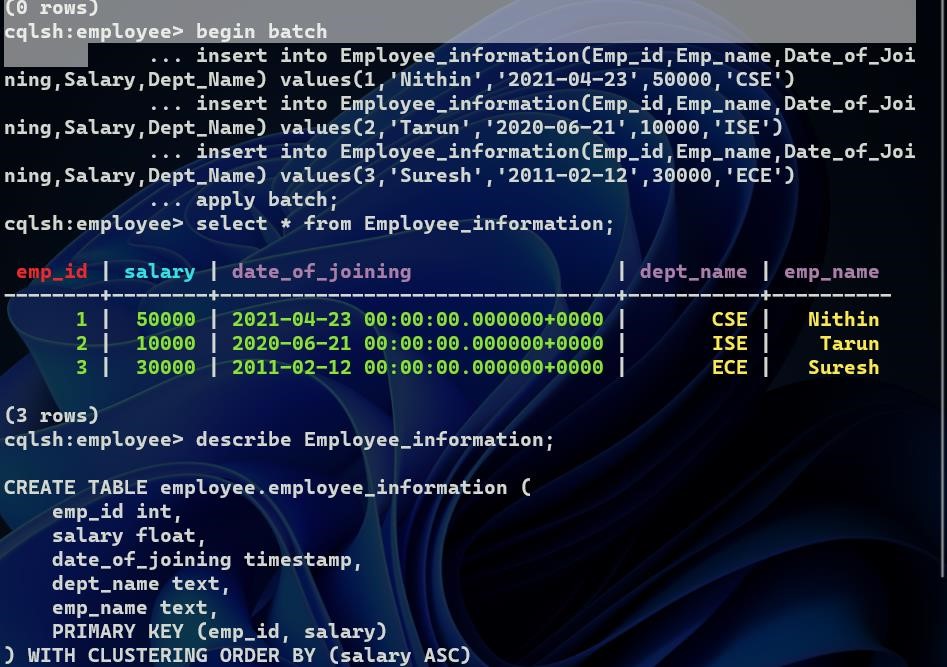
... apply batch;



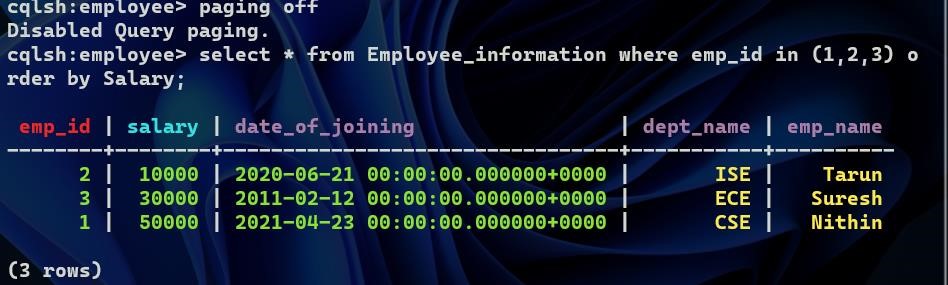
1. Update Employee name and Department of Emp-Id 1 update employee\_info set Dept\_Name='Mech',emp\_name='Sreekar' where emp\_id=1;
2. cqlsh:employee> select \* from employee\_info;



1. Sort the details of Employee records based on salary



cqlsh:employee> select \* from Employee\_information where emp\_id in (1,2,3) order by Salary;



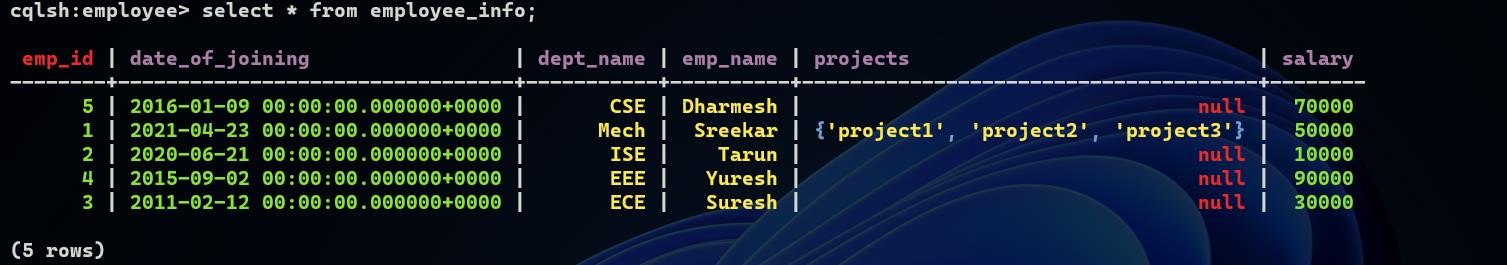
1. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

cqlsh:employee> alter table employee\_info add projects set<text>;

1. Update the altered table to add project names.

cqlsh:employee> update employee\_info set

projects=projects+{'project1','project2','project3'} where emp\_id=1;



8 Create a TTL of 15 seconds to display the values of Employees.



**LAB 3:**

1.Create a key space by name Library



2.Create a column family by name Library-Info with attributes Stud\_Id Primary Key, Counter\_value of type Counter,



3. Insert the values into the table in batch



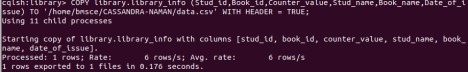
4.Display the details of the table created and increase the value of the counter



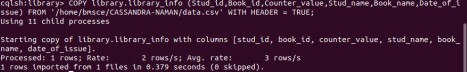
5.Write a query to show that a student with id 112 has taken a book “BDA” 2 times.



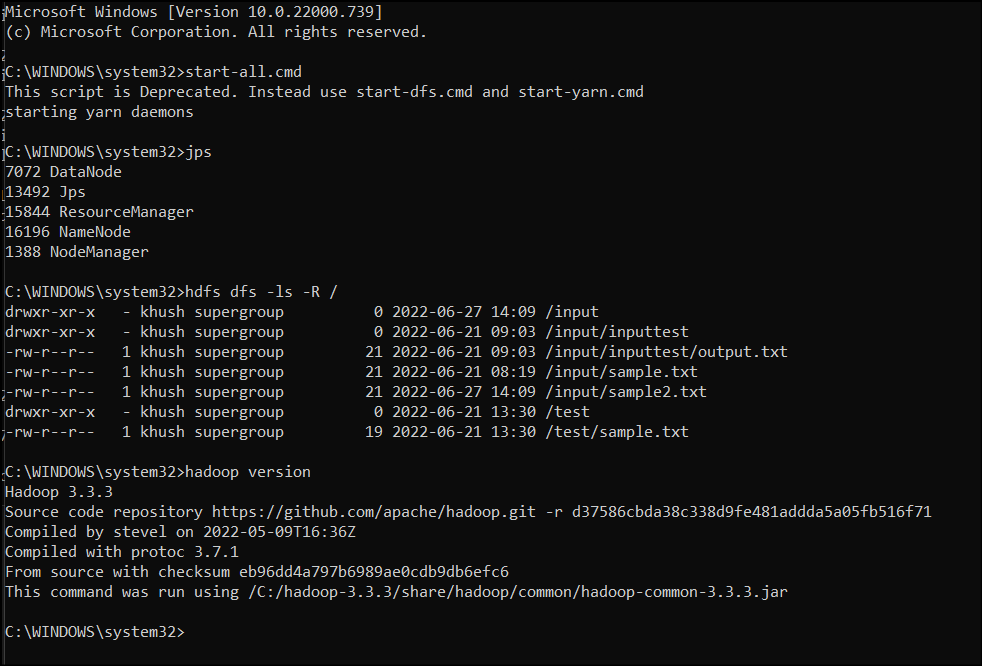
6.Export the created column to a csv file



7.Import a given csv dataset from local file system into Cassandra column family



Hadoop Installation



Hadoop Commands

hdusersbmsce-OptiPlus-3000:-$ sudo su hduser

[sudo] password for hduser:

hdusersbmsce-OptiPlus-3000: $ start-all.sh

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

22/06/06 14:43:45 WARN util.NativeCodeLoader: Unable to load native-hadoop Library for your platform... using builtin-java classes where applicable

Starting namenodes on [localhost]

localhost: nanenade running as process 3396. Stop it first.

localhost: datanode running as process 3564, Stop it first.

starting secondary nanenodes [0.0.0.0)

0.0.0.0: secondarynamenode running as process 3773. Stop it first.

O22/06/06 14:43:47 WARN uttt.NativeCodeLoader: Unable to load native-hadoop library for your

starting yarn daemons

resource process 3932. Stop it first.

Localhost: running as process 4255. stop it first.

6003 Jps

3932 ResourceManager

3773 SecondaryNameNode

4255 NodeManager

hdusersbmsce-OptiPlus-3060:-$ hdfs dfs -mkdir /khushil

hdusersbmsce-OptiPlus-3060: $ hdfs dfs -ls /

22/06/06 14:45:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable Found 19 itens

drwxr-xr-x hduser supergroup

02022-06-06 11:44 /AAA

drwxr-xr-x -hduser supergroup

2022-06-03 12:17 /Army

drwxr-xr-x hduser supergroup

02022-06-06 11:40 /Avnit

drwxr-xr-x -hduser supergroup

02022-05-31 10:44 /88

drwxr-xr-x -hduser supergroup

02022-06-01 15:03 /Cath

drwxr-xr-x -hduser supergroup

drwxr-xr-x hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x -hduser supergroup

drwxr-xr-x - hduser supergroup

drwxr-xr-x -hduser supergroup

82022-06-04 10:06 /FFF

02022-06-06 14:40 /Kmrv

02022-06-06 14:44 /Khushil

02022-06-01 15:03 /Neha

02022-06-04 09:54 /WC.txt

0 2022-06-04 09:54 /welcone.txt

02022-06-06 11:36 /abc

62022-06-03 12:13 /akash

0 2022-06-03 15:12 /darshan

0 2022-06-04 09:31 /ghh

8 2022-06-06 11:45 /hello

drwxr-xr-x -hduser supergroup

62022-06-04 09:35 /rahul

drwxr-xr-x -hduser supergroup

02022-06-03 12:11 /shre

drwxr-xr-x .hduser supergroup

02022-06-03 12:41 /shreshtha

hdusersbmsce-OptiPlus-3060:-$ hdfs dfs put /home/hduser/Desktop/6b.txt /Khushil/WC.txt

22/05/06 14:46:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using butltin-java classes where applicable hduserabesce-OptiPlex-3060:-$ hdfs dfs cat /Khushil/WC.txt

22/06/06 14:47:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable hello fron of

hdusersbmsce-OptiPlus-3040:-$ hdfs dfs-get /Khushil/WC.txt /home/hduser/Downloads/newic.txt

22/05/06 14:51:43 WARN util.NativeCodeLoader: Unable to load nattve-hadoop library for your platform... using builtin-java classes where applicable

hdusersbmsce-OptiPlus-3066:-$ cd Downloads

hdusersbmsce-OptiPlus-3060:-/Downloads$ cat newwMC.Ext

hello from 6E

hdusersbmsce-OptiPlus-3060:-$ hdfs dfs -1s /Khushil/

22/06/06 14:54:04 WARN util.NativeCodeLoader: Unable to load native-hadoop Library for your platform... using builtin java classes where applicable

Found 2 itens

-rw-r--r-- 1 hduser supergroup

23 2822-06-06 14:46 /Khushil/MC.txt

1 hduser supergroup

23 2022-06-06 14:58 /Khushil/newwc.txt

hdusersbmsce-OptiPlus-3060:-5 hdfs drs -getmerge /Khushil/wc.txt /Khushil/newwc.txt /bone/hduser/Desktop/newmerge.txt

22/06/06 14:55:18 NARN util.NativeCodeLoader: Unable to load nattve-hadoop library for your platform... using butitin-Java classes where applicable

hduserabesce-OptiPlex-3060:~$ cd Desktop

hduser@besce-OptiPlex-3060:-/Desktops cat newmerge.txt

hello from 68

D

B

hello from 68

D

B

hdusersbmsce-OptiPlus-3060:-/Desktops hadoop fs getfacl /Khushil/

22/06/06 14:56:24 WARN util.NativeCodeLoader: Unable to load native hadoop library for your platform... using builtin java classes where applicable

# file: /Khushil

# owner: hduser

# group: supergroup

user::rwx

group::r-x

other::r-x

hdusersbmsce-OptiPlus-3060:-/Desktop5 hdfs dfs copyToLocal /Khushil/HC.txt /home/hduser/Desktop

22/05/06 14:58:09 WARN util.NativeCodeLoader: Unable to load native-hadoop Library for your platform... using butltin-java classes where applicable

hdusersbmsce-OptiPlus-3000:-/Desktop5 cat MC.txt

hello fron 68

hdusersbmsce-OptiPlus-3060:-/Desktops hdfs dfs -cat /Khushil/MC.txt

22/06/06 14:58:59 WARN util.NativeCodeLoader: Unable to load native-hadoop Library for your platform... ustng bulltin-Java classes where applicable hello from GB

B

hdusersbmsce-OptiPlus-3060:-/Desktop5 hadoop fs - /Khushil /FFF 22/06/06 14:59:46 WARN util.NativeCodeLoader: Unable to load native-hadoop Library for your platform... using builtin-java classes where applicable hduseransce-OptiPlex-3060:-/Desktops hadoop fs-Ls /FFF 22/05/06 15:00:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using butltin-java classes where applicable Found 2 itens drwxr-xr-x -hduser supergroup TWEE 1 hduser supergroup 02022-05-06 14:50 /FFF/Khushil 17 2022-05-04 10:06 /FFF/MC.txt

hdusersbmsce-OptiPlus-3060:-/Desktops hadoop fs cp /FFF/ /LLL

22/06/06 15:09:34 WARN util.NativeCodeLoader: Unable to load native hadoop library for your platform... using butltin-java classes where applicable

hdusersbmsce-OptiPlus-3060:-/Desktops hadoop fs -Ls /LLL

22/06/06 15:10:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 2 1tens

drwxr-xr-x -hduser supergroup

hdusersbmsce-OptiPlus-3000:-/Desktops

02022-06-06 15:09 /LLL/KHUSHIL

17 2022-00-00 15:09 /LLL/MC.txt

Hadoop Programs

1. Word Count

WCMapper Java Class file.

// Importing libraries

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> {

    // Map function

    public void map(LongWritable key, Text value, OutputCollector<Text,

                IntWritable> output, Reporter rep) throws IOException

    {

        String line = value.toString();

        // Splitting the line on spaces

        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:

// 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

    {

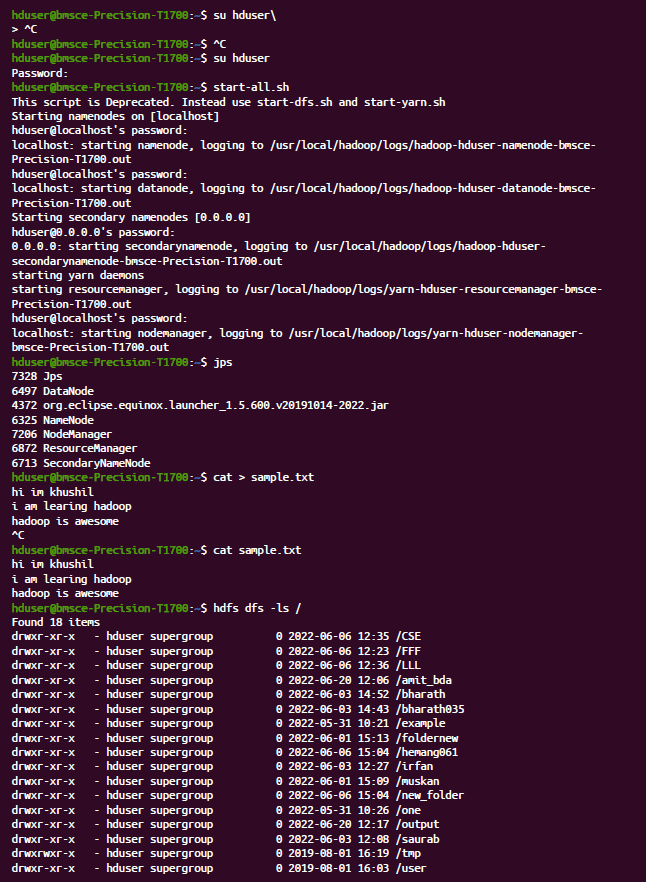
        int exitCode = ToolRunner.run(new WCDriver(), args);

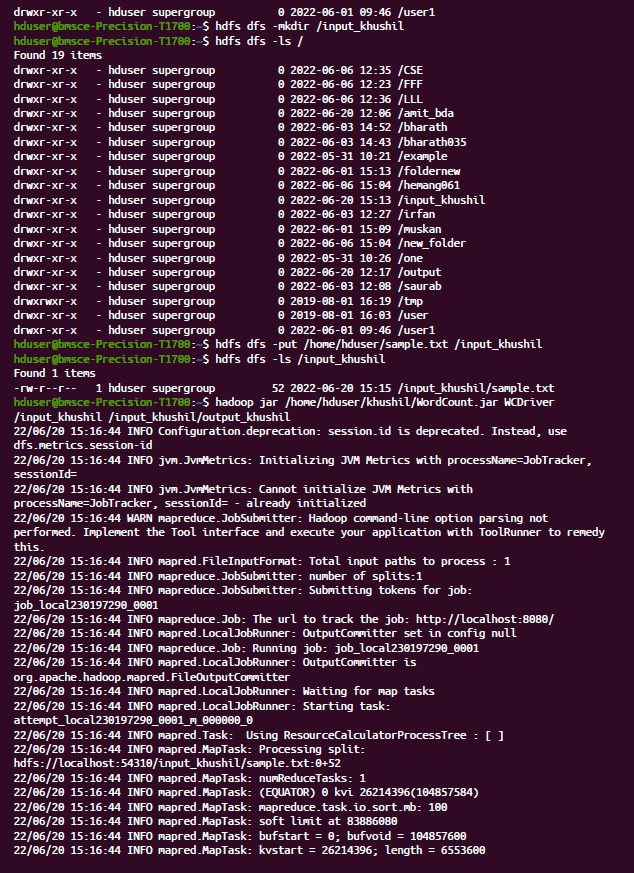
        System.out.println(exitCode);

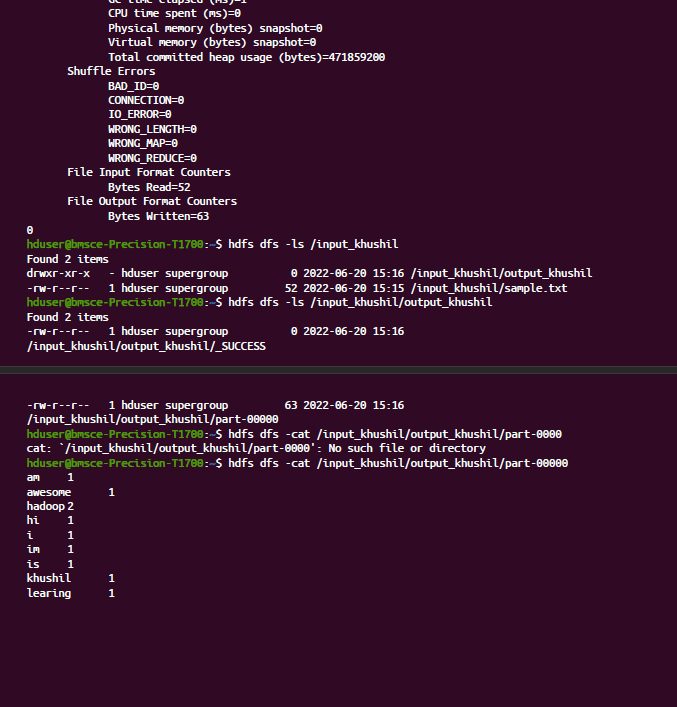
    }

}

Output :







1. Top N

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));  
  }  
}

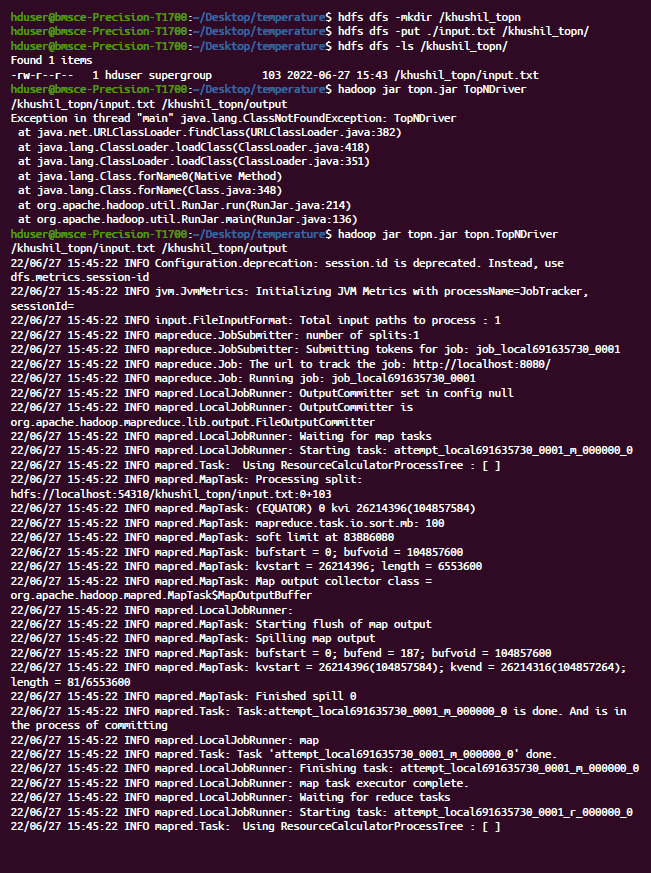
TopNMapper.class

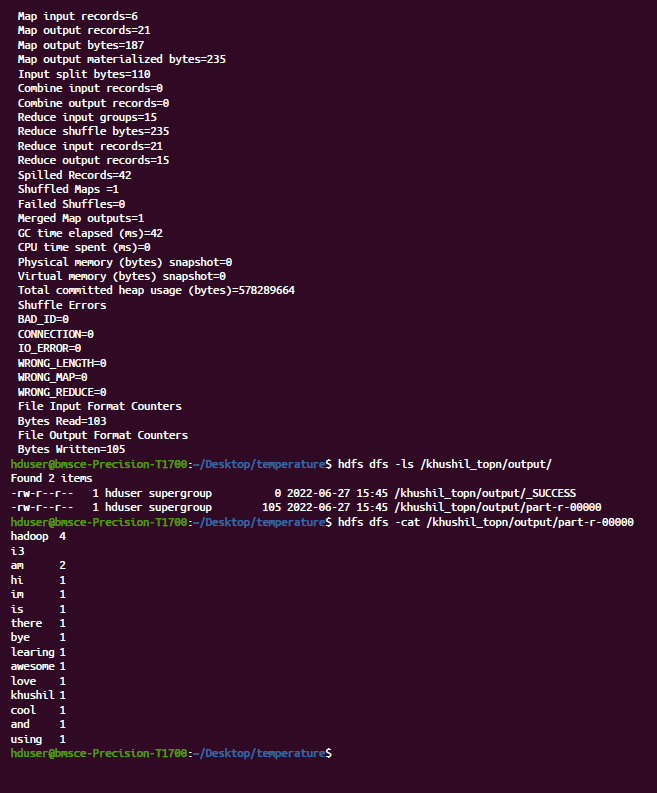
**package** samples.topn;  
  
**import** java.io.IOException;  
**import** java.util.StringTokenizer;  
**import** org.apache.hadoop.io.IntWritable;  
**import** org.apache.hadoop.io.Text;  
**import** org.apache.hadoop.mapreduce.Mapper;  
  
**public** **class** TopNMapper **extends** Mapper<Object, Text, Text, IntWritable> {  
  **private** **static** **final** IntWritable one = **new** IntWritable(1);  
    
  **private** Text word = **new** Text();  
    
  **private** String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";  
    
  **public** **vo```\\id** 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));  
    }   
  }  
}

Output:





1. Average Temperature

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);  
  }  
}

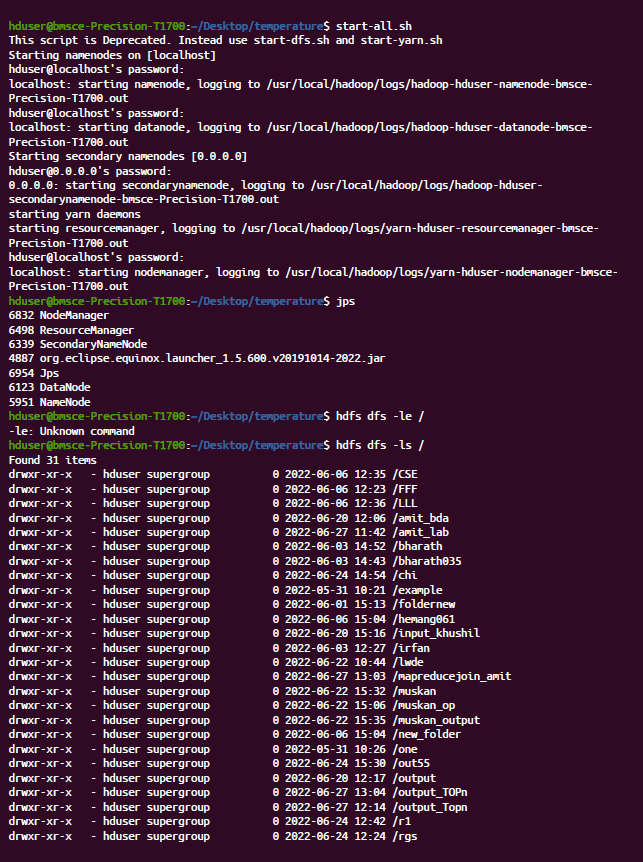
**AverageMapper**

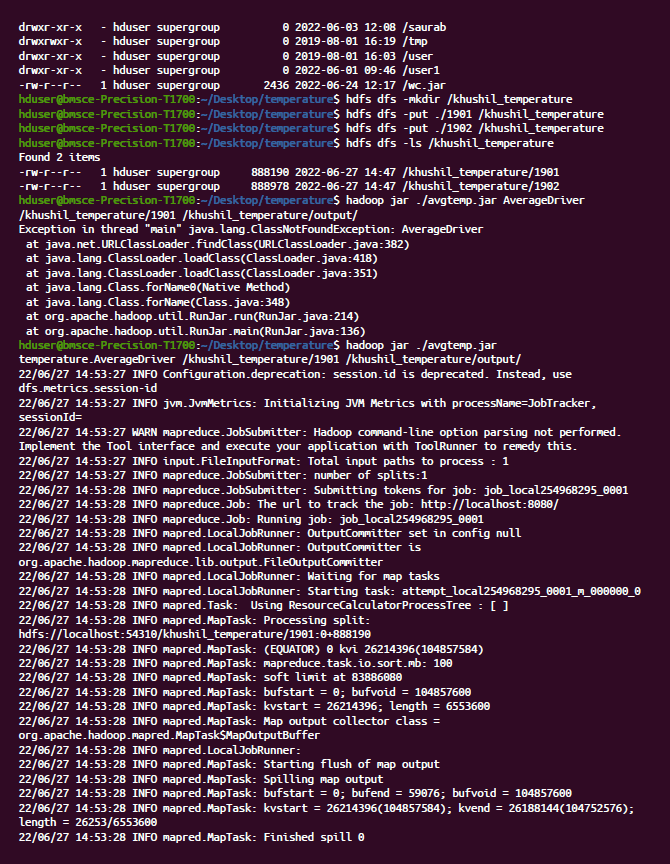
**package** temp;  
  
**import** java.io.IOException;  
**import** org.apache.hadoop.io.IntWritable;  
**import** org.apache.hadoop.io.LongWritable;  
**import** org.apache.hadoop.io.Text;  
**import** org.apache.hadoop.mapreduce.Mapper;  
  
**public** **class** AverageMapper **extends** Mapper<LongWritable, 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));   
  }  
}

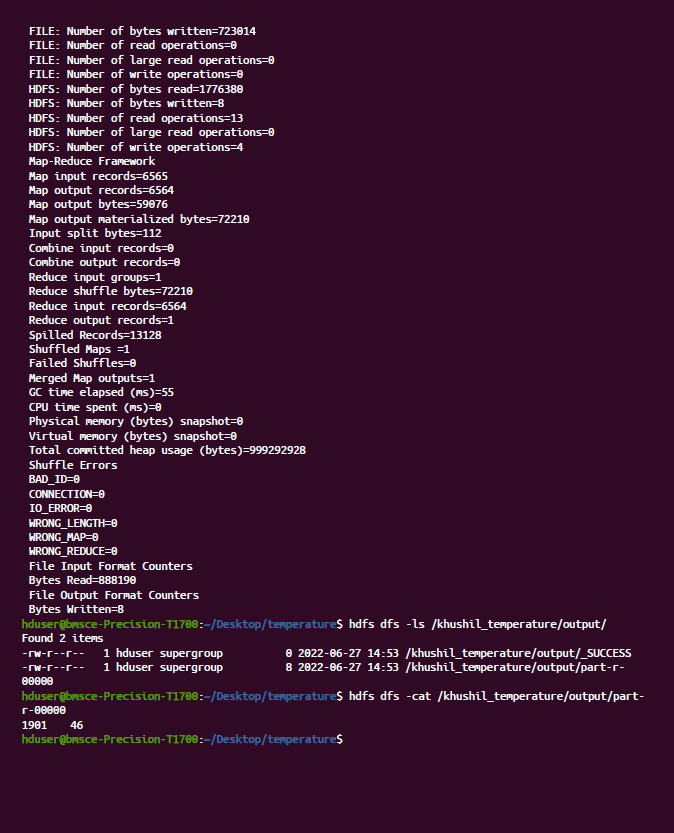
AverageReducer

**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));  
  }  
}

Output:







1. Join

// JoinDriver.java

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.\*;

import org.apache.hadoop.mapred.lib.MultipleInputs;

import org.apache.hadoop.util.\*;

public class JoinDriver extends Configured implements Tool {

public static class KeyPartitioner implements Partitioner<TextPair, Text> {

@Override

public void configure(JobConf job) {

}

@Override

public int getPartition(TextPair key, Text value, int numPartitions) {

return (key.getFirst().hashCode() & Integer.MAX\_VALUE) %

numPartitions;

}

}

@Override

public int run(String[] args) throws Exception {

if (args.length != 3) {

System.out.println("Usage: <Department Emp Strength input>

<Department Name input> <output>");

return -1;

}

JobConf conf = new JobConf(getConf(), getClass());

conf.setJobName("Join 'Department Emp Strength input' with 'Department Name

input'");

Path AInputPath = new Path(args[0]);

Path BInputPath = new Path(args[1]);

Path outputPath = new Path(args[2]);

MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,

Posts.class);

MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,

User.class);

FileOutputFormat.setOutputPath(conf, outputPath);

conf.setPartitionerClass(KeyPartitioner.class);

conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

conf.setMapOutputKeyClass(TextPair.class);

conf.setReducerClass(JoinReducer.class);

conf.setOutputKeyClass(Text.class);

JobClient.runJob(conf);

return 0;

}

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

int exitCode = ToolRunner.run(new JoinDriver(), args);

System.exit(exitCode);

}

}

// JoinReducer.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.\*;

public class JoinReducer extends MapReduceBase implements Reducer<TextPair, Text, Text,

Text> {

@Override

public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text>

output, Reporter reporter)

throws IOException

{

Text nodeId = new Text(values.next());

while (values.hasNext()) {

Text node = values.next();

Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());

output.collect(key.getFirst(), outValue);

}

}

}

// User.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.\*;

import org.apache.hadoop.io.IntWritable;

public class User extends MapReduceBase implements Mapper<LongWritable, Text, TextPair,

Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,

Reporter reporter)

throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t");

output.collect(new TextPair(SingleNodeData[0], "1"), new

Text(SingleNodeData[1]));

}

}

// Posts.java

import java.io.IOException;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

public class Posts extends MapReduceBase implements Mapper<LongWritable, Text, TextPair,

Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,

Reporter reporter)

throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t");

output.collect(new TextPair(SingleNodeData[3], "0"), new

Text(SingleNodeData[9]));

}

}

// TextPair.java

import java.io.\*;

import org.apache.hadoop.io.\*;

public class TextPair implements WritableComparable<TextPair> {

private Text first;

private Text second;

public TextPair() {

set(new Text(), new Text());

}

public TextPair(String first, String second) {

set(new Text(first), new Text(second));

}

public TextPair(Text first, Text second) {

set(first, second);

}

public void set(Text first, Text second) {

this.first = first;

this.second = second;

}

public Text getFirst() {

return first;

}

public Text getSecond() {

return second;

}

@Override

public void write(DataOutput out) throws IOException {

first.write(out);

second.write(out);

}

@Override

public void readFields(DataInput in) throws IOException {

first.readFields(in);

second.readFields(in);

}

@Override

public int hashCode() {

return first.hashCode() \* 163 + second.hashCode();

}

@Override

public boolean equals(Object o) {

if (o instanceof TextPair) {

TextPair tp = (TextPair) o;

return first.equals(tp.first) && second.equals(tp.second);

}

return false;

}

@Override

public String toString() {

return first + "\t" + second;

}

@Override

public int compareTo(TextPair tp) {

int cmp = first.compareTo(tp.first);

if (cmp != 0) {

return cmp;

}

return second.compareTo(tp.second);

}

// ^^ TextPair

// vv TextPairComparator

public static class Comparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator();

public Comparator() {

super(TextPair.class);

}

@Override

public int compare(byte[] b1, int s1, int l1,

byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);

int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);

int cmp = TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);

if (cmp != 0) {

return cmp;

}

return TEXT\_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,

b2, s2 + firstL2, l2 - firstL2);

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

}

static {

WritableComparator.define(TextPair.class, new Comparator());

}

public static class FirstComparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator();

public FirstComparator() {

super(TextPair.class);

}

@Override

public int compare(byte[] b1, int s1, int l1,

byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);

int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);

return TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

@Override

public int compare(WritableComparable a, WritableComparable b) {

if (a instanceof TextPair && b instanceof TextPair) {

return ((TextPair) a).first.compareTo(((TextPair) b).first);

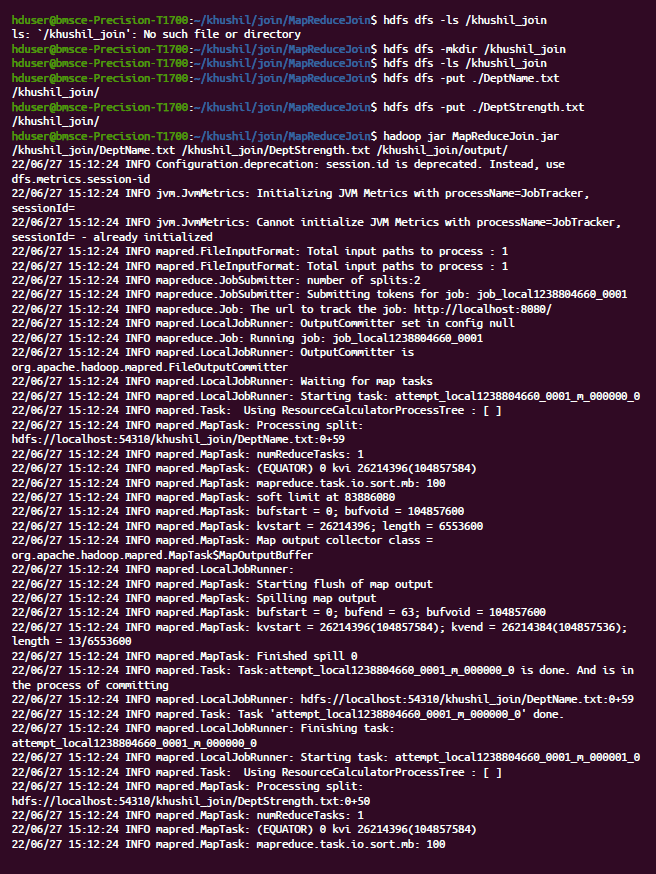
}

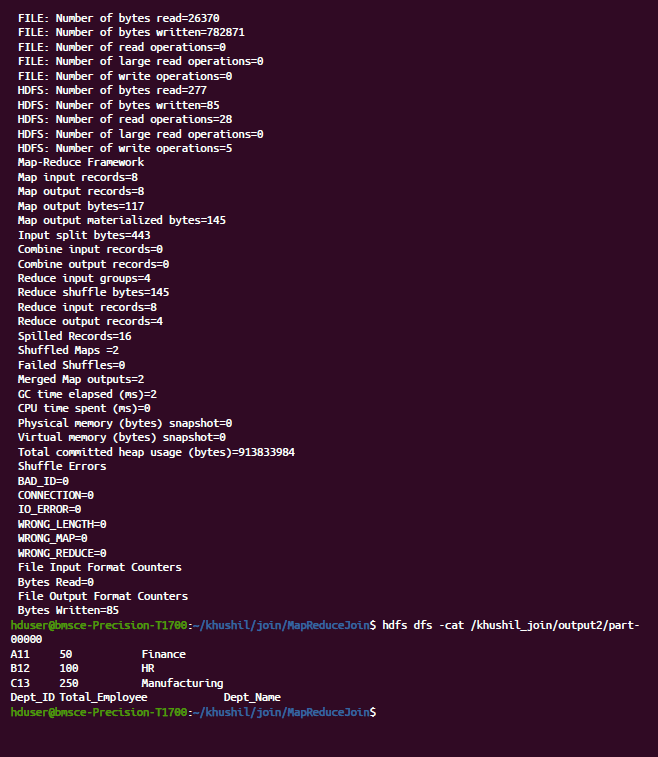
return super.compare(a, b);

}

}

}

Output: 



Scala Programming:

Lab 9:

val data=sc.textFile("sparkdata.txt")

data.collect;

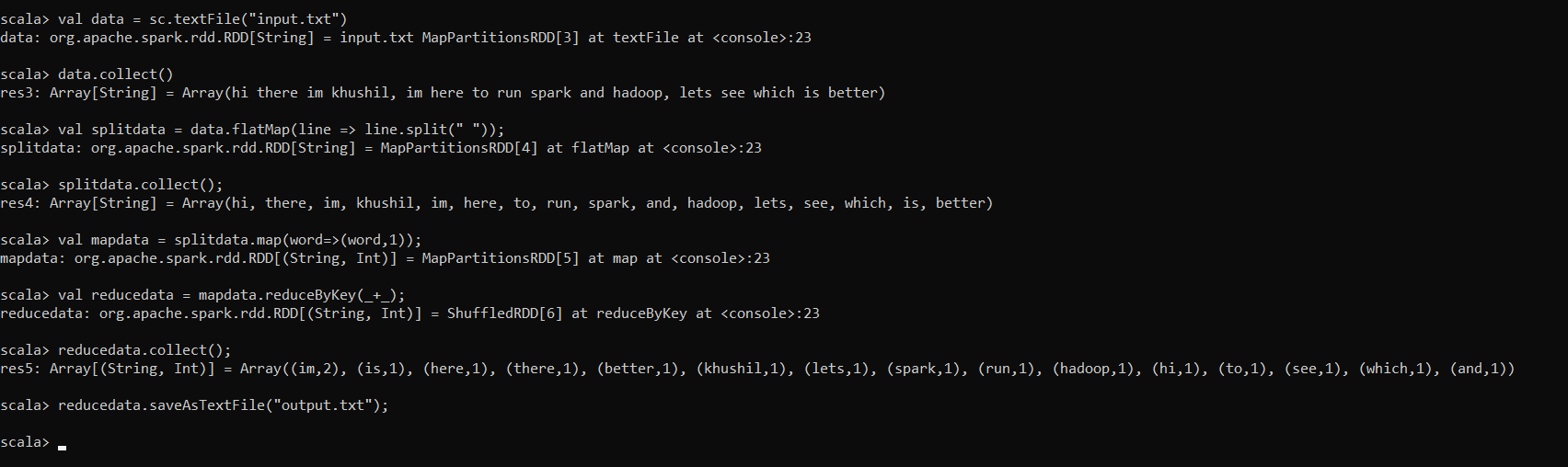
val splitdata = data.flatMap(line =**>** line.split(" "));

splitdata.collect;

val mapdata = splitdata.map(word =**>** (word,1));

mapdata.collect;

val reducedata = mapdata.reduceByKey(\_+\_); reducedata.collect;



Lab 10:

val textFile = sc.textFile("/home/bhoom/Desktop/wc.txt")

val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_ + \_)

import scala.collection.immutable.ListMap

val sorted=ListMap(counts.collect.sortWith(\_.\_2 > \_.\_2):\_\*)// sort in descending order based on values

println(sorted)

for((k,v)<-sorted)

{

if(v>4)

{

print(k+",")

print(v)

println()

}}

