**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

**Rohit Satheesh Nair (1BM19CS206)**

***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**

**May-2022 to July-2022**

**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 **Rohit Satheesh Nair(1BM19CS206),** 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 a **BIG DATA ANALYTICS - (20CS6PEBDA)** work prescribed for the said degree.

**Prof. Pallavi GB**              **Dr. Jyothi S Nayak**

Assistant Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **01** | **MongoDB** | **4** |
| **02** | **Cassandra Lab Program 1 : Student Database** | **14** |
| **03** | **Cassandra Lab Program 2 : Library Database** | **22** |
| **04** | **Wordcount Program - Hadoop** | **26** |
| **05** | **Wordcount Program - Mapreducer** | **31** |
| **06** | **Hadoop TopN Program** | **36** |
| **07** | **Average Temperature** | **42** |
| **08** | **MeanMax Temperature** | **46** |
| **09** | **Join Operation Using Mapreduce** | **50** |
| **10** | **Wordcount using Scala** | **61** |
| **11** | **Wordcount greater than 4 using Scala** | **62** |

**Course Outcome**

|  |  |
| --- | --- |
| CO 1 | 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**

bmsce@bmsce-Precision-T1700: ~$ mongo

MongoDB shell version v3.6.8

connecting to: mongodb://127.0.0.1:27017

Implicit session: session {“id”: UUID("d66acdb3-8482-417d-8b75-

d65dae4b53ee")}

MongoDB server version: 3.6.8

Server has startup warnings:

2022-04-11T18:49:15.627+0530 I STORAGE [initandlisten]

2022-04-11T18:49:15.627+0530 I STORAGE [initandlisten] \*\*

WARNING: Using the XFS filesystem is strongly recommended with

the WiredTiger storage engine

2022-04-11T18:49:15.627+0530 I STORAGE [initandlisten] \*\*

See http://dochub.mongodb.org/core/prodnotes-filesystem

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten]

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten] \*\*

WARNING: Access control is not enabled for the database.

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten] \*\*

Read and write access to data and configuration is unrestricted.

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten]

> use Student

switched to db Student

> db.createCollection("student");

{ "ok" : 1 }

>

db.Student.insert({\_id:1,StudName:"Megha",Grade:"vii",Hobbies:"Inter

netSurfing"});

WriteResult({ "nInserted" : 1 })

>

db.Student.update({\_id:3,StudName:"Ayan",Grade:"vii"},{$set:{Hobbie

s:"skating"}},{upsert:true});

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" :

3 })

> db.Student.find({StudName:"Ayan"});

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

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

{ "StudName" : "Megha", "Grade" : "vii" }

{ "Grade" : "vii", "StudName" : "Ayan" }

> db.Student.find({Grade:{$eq:'vii'}}).pretty();

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

> db.Student.find({Grade:{$eq:'vii'}});

{ "\_id" : 1, "StudName" : "Megha", "Grade" : "vii", "Hobbies" :

"InternetSurfing" }

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

> db.Student.find({Grade:{$eq:'vii'}}).pretty();

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

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

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

> db.Student.find({Hobbies:{$in:['skating']}}).pretty();

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

> db.Student.count();

2

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

> db.Student.save({StudName:"Vamsi",Greade:"vi"})

WriteResult({ "nInserted" : 1 })

> db.Students.update({\_id:4},{$set:{Location:"Network"}})

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

> db.Students.update({\_id:4},{$unset:{Location:"Network"}})

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

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

{ "StudName" : "Megha", "Grade" : "vii" }

> db.Student.find({Grade:{$ne:'VII'}}).pretty();

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{ "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating"

}

{

"\_id" : ObjectId("6253f413e88b8c9e787b194e"),

"StudName" : "Vamsi",

"Greade" : "vi"

}

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

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

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

> db.Students.count()

0

> db.Students.count({Grade:"VII"})

0

> db.Student.find({Grade:"VII"}).limit(3).pretty();

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

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

> db.Student.count({Grade:"VII"})

0

> db.Students.count({Grade:"vii"})

0

> db.Student.count()

3

> db.Student.count({Grade:"vii"})

2

> db.Student.find({Grade:"vii"}).limit(3).pretty();

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{

"\_id" : 3,

"Grade" : "vii",

"StudName" : "Ayan",

"Hobbies" : "skating",

"Location" : null

}

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

{

"\_id" : 3,

"Grade" : "vii",

"StudName" : "Ayan",

"Hobbies" : "skating",

"Location" : null

}

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{

"\_id" : ObjectId("6253f413e88b8c9e787b194e"),

"StudName" : "Vamsi",

"Greade" : "vi"

}

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

{

"\_id" : ObjectId("6253f413e88b8c9e787b194e"),

"StudName" : "Vamsi",

"Greade" : "vi"

}

> db.food.insert( { \_id:1, fruits:['grapes','mango','apple';] } )

2022-04-11T15:05:51.894+0530 E QUERY [thread1] SyntaxError:

missing ] after element list @(shell):1:57

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

> db.food.insert({\_id:3,fruits:['banana','mango']})

WriteResult({ "nInserted" : 1 })

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

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

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

> db.food.find({"fruits":{$size:2}})

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

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

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

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

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

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

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

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

>

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

}}})

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

>

> db.createCollection("Customers");

{

"ok" : 0,

"errmsg" : "a collection 'bhuvana.Customers' already exists",

"code" : 48,

"codeName" : "NamespaceExists"

}

db.Customers.insert({\_custID:1,AcctBal:'100000',AcctType:"saving"});

WriteResult({ "nInserted" : 1 })

>

db.Customers.aggregate({$group:{\_id:"$custID",TotAccBal:{$sum:"$A

ccBal"}}});

{ "\_id" : null, "TotAccBal" : 0 }

db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"

$custID",TotAccBal:{$sum:"$AccBal"}}});

{ "\_id" : null, "TotAccBal" : 0 }

db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"

$custID",TotAccBal:{$sum:"$AccBal"}}},{$match:{TotAccBal:{$gt:1

200}}});

**LAB 2**

cqlsh:employee> CREATE KEYSPACE employee WITH

REPLICATION={ 'class' : 'SimpleStrategy', 'replication\_factor' : 1};

cqlsh:employee> USE employee;

cqlsh:employee> create table employee\_info(emp\_id int PRIMARY

KEY, emp\_name text,

... designation text, date\_of\_joining timestamp, salary double

PRIMARY KEY, dept\_name text);

cqlsh:employee> CREATE TABLE employee\_info(emp\_id int,

emp\_name text, designation text, date\_of\_joining timestamp, salary

double, dept\_name text, PRIMARY KEY(emp\_id, salary));

cqlsh:employee> BEGIN BATCH INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(100,'Jogesh','MANAGER','2021-09-

11',30000,'TESTING');

... INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(111,'Tamara','ASSOCIATE','2021-06-

22',25000,'DEVELOPING');

... INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(121,'Elenor','MANAGER','2021-03-

30',35000,'HR');

... INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(115,'Charu','ASSISTANT','2021-12-

30',20000,'DEVELOPING');

... INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(105,'Santosh','ASSOCIATE','2021-06-

25',25000,'TESTING');

... APPLY BATCH;

cqlsh:employee> SELECT \* FROM employee\_info

... ;

emp\_id | salary | date\_of\_joining | dept\_name | designation |

emp\_name

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

-----

105 | 25000 | 2021-06-24 18:30:00.000000+0000 | TESTING |

ASSOCIATE | Santosh

111 | 25000 | 2021-06-21 18:30:00.000000+0000 | DEVELOPING |

ASSOCIATE | Tamara

121 | 35000 | 2021-03-29 18:30:00.000000+0000 | HR |

MANAGER | Elenor

115 | 20000 | 2021-12-29 18:30:00.000000+0000 | DEVELOPING |

ASSISTANT | Charu

100 | 30000 | 2021-09-10 18:30:00.000000+0000 | TESTING |

MANAGER | Jogesh

(5 rows)

cqlsh:employee> UPDATE employee\_info SET emp\_name = 'Jayesh',

dept\_name = 'DEVELOPING' WHERE emp\_id = 121;

cqlsh:employee> UPDATE employee\_info SET emp\_name = 'Jayesh',

dept\_name = 'DEVELOPING' WHERE emp\_id = 121 AND salary =

35000;

cqlsh:employee> SELECT \* FROM employee\_info ;

emp\_id | salary | date\_of\_joining | dept\_name | designation |

emp\_name

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

-----

105 | 25000 | 2021-06-24 18:30:00.000000+0000 | TESTING |

ASSOCIATE | Santosh

111 | 25000 | 2021-06-21 18:30:00.000000+0000 | DEVELOPING |

ASSOCIATE | Tamara

121 | 35000 | 2021-03-29 18:30:00.000000+0000 | DEVELOPING |

MANAGER | Jayesh

115 | 20000 | 2021-12-29 18:30:00.000000+0000 | DEVELOPING |

ASSISTANT | Charu

100 | 30000 | 2021-09-10 18:30:00.000000+0000 | TESTING |

MANAGER | Jogesh

(5 rows)

cqlsh:employee> SELECT \* FROM employee\_info WHERE emp\_id in

(105, 111, 121, 115, 100) order by salary;

cqlsh:employee> paging off

Disabled Query paging.

cqlsh:employee> SELECT \* FROM employee\_info WHERE emp\_id in

(105, 111, 121, 115, 100) order by salary;

emp\_id | salary | date\_of\_joining | dept\_name | designation |

emp\_name

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

-----

115 | 20000 | 2021-12-29 18:30:00.000000+0000 | DEVELOPING |

ASSISTANT | Charu

105 | 25000 | 2021-06-24 18:30:00.000000+0000 | TESTING |

ASSOCIATE | Santosh

111 | 25000 | 2021-06-21 18:30:00.000000+0000 | DEVELOPING |

ASSOCIATE | Tamara

100 | 30000 | 2021-09-10 18:30:00.000000+0000 | TESTING |

MANAGER | Jogesh

121 | 35000 | 2021-03-29 18:30:00.000000+0000 | DEVELOPING |

MANAGER | Jayesh

(5 rows)

cqlsh:employee> ALTER TABLE employee\_info ADD projects text;

cqlsh:employee> UPDATE employee\_info SET projects = 'Chat App'

WHERE emp\_id = 111;

cqlsh:employee> UPDATE employee\_info SET projects = 'Chat App'

WHERE emp\_id = 111 and salary = 25000;

cqlsh:employee> UPDATE employee\_info SET projects = 'Discord Bot'

WHERE emp\_id = 115 and salary = 20000;

cqlsh:employee> UPDATE employee\_info SET projects = 'Campus

Portal' WHERE emp\_id = 105 and salary = 25000;

cqlsh:employee> UPDATE employee\_info SET projects = 'YouTube

Downloader' WHERE emp\_id = 100 and salary = 30000;

cqlsh:employee> UPDATE employee\_info SET projects = 'Library

Management System ' WHERE emp\_id = 121 and salary = 35000;

cqlsh:employee> SELECT \* FROM employee\_infor

... ;

cqlsh:employee> SELECT \* FROM employee\_info ;

emp\_id | salary | date\_of\_joining | dept\_name | designation |

emp\_name | projects

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

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

105 | 25000 | 2021-06-24 18:30:00.000000+0000 | TESTING |

ASSOCIATE | Santosh | Campus Portal

111 | 25000 | 2021-06-21 18:30:00.000000+0000 | DEVELOPING |

ASSOCIATE | Tamara | Chat App

121 | 35000 | 2021-03-29 18:30:00.000000+0000 | DEVELOPING |

MANAGER | Jayesh | Library Management System

115 | 20000 | 2021-12-29 18:30:00.000000+0000 | DEVELOPING |

ASSISTANT | Charu | Discord Bot

100 | 30000 | 2021-09-10 18:30:00.000000+0000 | TESTING |

MANAGER | Jogesh | YouTube Downloader

(5 rows)

cqlsh:employee> INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

...

... ;

cqlsh:employee> INSERT INTO

...

employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,de

pt\_name)

... VALUES(110,'SAM','ASSOCIATE','2021-01-

11',28000,'TESTING') USING TTL 15;

cqlsh:employee> SELECT TTL(emp\_name) from employee\_info

WHERE emp\_id = 110;

ttl(emp\_name)

---------------

3

(1 rows)

cqlsh:employee> SELECT \* FROM employee\_info;

emp\_id | salary | date\_of\_joining | dept\_name | designation |

emp\_name | projects

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

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

105 | 25000 | 2021-06-24 18:30:00.000000+0000 | TESTING |

ASSOCIATE | Santosh | Campus Portal

111 | 25000 | 2021-06-21 18:30:00.000000+0000 | DEVELOPING |

ASSOCIATE | Tamara | Chat App

121 | 35000 | 2021-03-29 18:30:00.000000+0000 | DEVELOPING |

MANAGER | Jayesh | Library Management System

115 | 20000 | 2021-12-29 18:30:00.000000+0000 | DEVELOPING |

ASSISTANT | Charu | Discord Bot

100 | 30000 | 2021-09-10 18:30:00.000000+0000 | TESTING |

MANAGER | Jogesh | YouTube Downloader

**LAB 3**

cqlsh:library> CREATE KEYSPACE library WITH replication =

{'class':

'SimpleStrategy','replication\_factor':1}; cqlsh:library> USE library ;

cqlsh:library> CREATE TABLE Library\_info(stud\_id int, stud\_name

text, book\_name text, book\_id text, date\_of\_issue timestamp,

counter\_value counter, PRIMARY KEY(stud\_id,stud\_name,

book\_name, book\_id, date\_of\_issue));

cqlsh:library> BEGIN COUNTER BATCH

... UPDATE library\_info set counter\_value +=1 where stud\_id =

111 and stud\_name = 'Manju' and book\_name = 'Human Behaviour' and

book\_id = '52e43' and date\_of\_issue = '2021-09-12';

... UPDATE library\_info set counter\_value +=1 where stud\_id =

112 and stud\_name = 'Kishore' and book\_name = 'Engineering

Mathematics-1' and book\_id = '52e44' and date\_of\_issue = '2021-04-10';

... UPDATE library\_info set counter\_value +=1 where stud\_id =

113 and stud\_name = 'Maitri' and book\_name = 'Dan Brown and

book\_id = '52e45' and date\_of\_issue = '2021-02-01';

... UPDATE library\_info set counter\_value +=1 where stud\_id =

114 and stud\_name = 'Ramesh' and book\_name = 'EME' and book\_id =

'52e46' and date\_of\_issue = '2021-04-03';

... APPLY BATCH;

cqlsh:library> SELECT \* FROM library\_info ;

stud\_id | stud\_name | book\_name | book\_id | date\_of\_issue

| counter\_value

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

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

114 | Ramesh | EME | 52e46 | 2021-04-02

18:30:00.000000+0000 | 1

111 | Manju | Human Behaviour | 52e43 | 2021-09-11

18:30:00.000000+0000 | 1

113 | Maitri | Dan Brown | 52e45 | 2021-01-31

18:30:00.000000+0000 | 1

112 | Kishore| Engineering Mathematics-1 | 52e44 | 2021-04-09

18:30:00.000000+0000 | 1

(4 rows)

cqlsh:library> UPDATE library\_info set counter\_value += 1 where

stud\_id = 112 and stud\_name = 'Kishore' and book\_name = 'Engineering

Mathematics-1' and book\_id = '52e44' and date\_of\_issue = '2021-04-09';

cqlsh:library> SELECT \* FROM library\_info ;

stud\_id | stud\_name | book\_name | book\_id | date\_of\_issue

| counter\_value

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

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

114 | Ramesh | EME | 52e46 | 2021-04-02

18:30:00.000000+0000 | 1

111 | Manoj | Human Behaviour | 52e43 | 2021-09-11

18:30:00.000000+0000 | 1

113 | Maitri | Dan Brown | 52e45 | 2021-01-31

18:30:00.000000+0000 | 1

112 | Kishore| Engineering Mathematics-1 | 52e44 | 2021-04-09

18:30:00.000000+0000 | 2

cqlsh:library> copy library\_info(stud\_id,stud\_name, book\_name,

book\_id, date\_of\_issue,counter\_value) to 'library\_info.csv' ;

Using 11 child processes

Starting copy of library.library\_info with columns [stud\_id, stud\_name,

book\_name, book\_id, date\_of\_issue, counter\_value]. Processed: 6 rows;

Rate: 39 rows/s; Avg. rate: 39 rows/s 6 rows exported to 1 files in

0.165 seconds.

cqlsh:library> copy library\_info(stud\_id,stud\_name, book\_name,

book\_id, date\_of\_issue,counter\_value) from 'library\_info.csv' ;

Using 11 child processes

Starting copy of library.library\_info with columns [stud\_id, stud\_name,

book\_name, book\_id, date\_of\_issue, counter\_value]. Processed: 6 rows;

Rate: 10 rows/s; Avg. rate: 15 rows/s 6 rows imported from 1 files

in 0.392 seconds (0 skipped).

**LAB 4**

hduser@bmsce-Precision-T1700:~$ start-all.sh

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

Starting namenodes on [localhost]

hduser@localhost's password:

localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoophduser-namenode-bmsce-Precision-T1700.out

hduser@localhost's password:

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoophduser-datanode-bmsce-Precision-T1700.out

Starting secondary namenodes [0.0.0.0]

hduser@0.0.0.0's password:

0.0.0.0: starting secondarynamenode, logging to

/usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-bmscePrecision-T1700.out

starting yarn daemons

starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduserresourcemanager-bmsce-Precision-T1700.out

hduser@localhost's password:

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarnhduser-nodemanager-bmsce-Precision-T1700.out

hduser@bmsce-Precision-T1700:~$ jps

7747 NodeManager

7045 DataNode

7416 ResourceManager

7257 SecondaryNameNode

6874 NameNode

7886 Jps

hduser@bmsce-Precision-T1700:~$ mkdir

mkdir: missing operand

Try 'mkdir --help' for more information.

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir/hadoop

-mkdir/hadoop: Unknown command

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir/lab6

-mkdir/lab6: Unknown command

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir /lab6

hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /

Found 6 items

drwxr-xr-x - hduser supergroup 0 2022-05-31 09:45 /ff

drwxr-xr-x - hduser supergroup 0 2022-05-31 09:16 /j

drwxr-xr-x - hduser supergroup 0 2022-06-01 09:31 /lab6

drwxr-xr-x - hduser supergroup 0 2022-05-31 09:57 /ss

drwxrwxr-x - hduser supergroup 0 2019-08-01 16:19 /tmp

drwxr-xr-x - hduser supergroup 0 2019-08-01 16:03 /user

hduser@bmsce-Precision-T1700:~$ hdfs dfs -put

/home/hduser/Desktop/Welcome.txt/abc/WC.txt

put: `.': No such file or directory

hduser@bmsce-Precision-T1700:~$ hdfs dfs -put

/home/hduser/Desktop/Welcome.txt/abc/WC.txt

put: `.': No such file or directory

hduser@bmsce-Precision-T1700:~$ hdfs dfs -put

/home/hduser/Desktop/Welcome.txt /lab6/WC.txt

hduser@bmsce-Precision-T1700:~$ sudo nano xyz.txt

[sudo] password for hduser:

hduser@bmsce-Precision-T1700:~$ sudo nano xyz.txt

hduser@bmsce-Precision-T1700:~$ hadoop fs -copyFromlocal xyz.txt

/lab6

-copyFromlocal: Unknown command

hduser@bmsce-Precision-T1700:~$ hadoop fs -copyFromLocal xyz.txt

/lab6

hduser@bmsce-Precision-T1700:~$ hadoop fs -ls/lab6

-ls/lab6: Unknown command

hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /lab6

Found 2 items

-rw-r--r-- 1 hduser supergroup 0 2022-06-01 09:40 /lab6/WC.txt

-rw-r--r-- 1 hduser supergroup 24 2022-06-01 09:45 /lab6/xyz.txt

hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /lab6/xyz.txt

Hello My name is Anitej

hduser@bmsce-Precision-T1700:~$ hdfs dfs -get /lab6/xyz.txt

/home/hduser/Downloads/WWC.txt

get: `/home/hduser/Downloads/WWC.txt': File exists

hduser@bmsce-Precision-T1700:~$ hdfs dfs -get /lab6/WC.txt

/home/hduser/Downloads/WWC.txt

get: `/home/hduser/Downloads/WWC.txt': File exists

hduser@bmsce-Precision-T1700:~$ hdfs dfs -get /lab6/xyz.txt

/home/hduser/Downloads/WWC.txt

get: `/home/hduser/Downloads/WWC.txt': File exists

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /lab6/xyz.xt

/home/hduser/Desktop

copyToLocal: `/lab6/xyz.xt': No such file or directory

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /lab6/xyz.txt

/home/hduser/Desktop

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /lab6/xyz.txt

Hello My name is Anitej

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /lab6/xyz.txt

/home/hduser/Desktop

copyToLocal: `/home/hduser/Desktop/xyz.txt': File exists

hduser@bmsce-Precision-T1700:~$ hdoop fs -mv /lab6 /FFF

hdoop: command not found

hduser@bmsce-Precision-T1700:~$ hadoop fs -mv /lab6 /FFF

hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /FFF

Found 2 items

-rw-r--r-- 1 hduser supergroup 0 2022-06-01 09:40 /FFF/WC.txt

-rw-r--r-- 1 hduser supergroup 24 2022-06-01 09:45 /FFF/xyz.txt

hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /lab6/ /LLL

cp: `/lab6/': No such file or directory

hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /CSE/ /LLL

cp: `/CSE/': No such file or directory

hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /lab6/ /LLL

cp: `/lab6/': No such file or directory

hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /lab6/ LLL

cp: `LLL': No such file or directory

**LAB 5**

Mapper Code: You have to copy paste this program into the 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&lt;LongWritable,

Text, Text,

IntWritable&gt; {

// Map function

public void map(LongWritable key, Text value,

OutputCollector&lt;Text,

IntWritable&gt; output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces

for (String word : line.split(“ “))

{

if (word.length() &gt; 0)

{

output.collect(new Text(word), new IntWritable(1));

} } } }

Reducer Code: You have to copy paste this program into the

WCReducer Java Class file

// 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&lt;Text,

IntWritable, Text, IntWritable&gt; {

// Reduce function

public void reduce(Text key, Iterator&lt;IntWritable&gt; value,

OutputCollector&lt;Text, IntWritable&gt; 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 &lt; 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);

}

**LAB 6**

**Driver-TopN**

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);

}

**TopN Combiner 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));

}

}

**TopN Mapper 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);

}

}

}

**TopN Reducer 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)

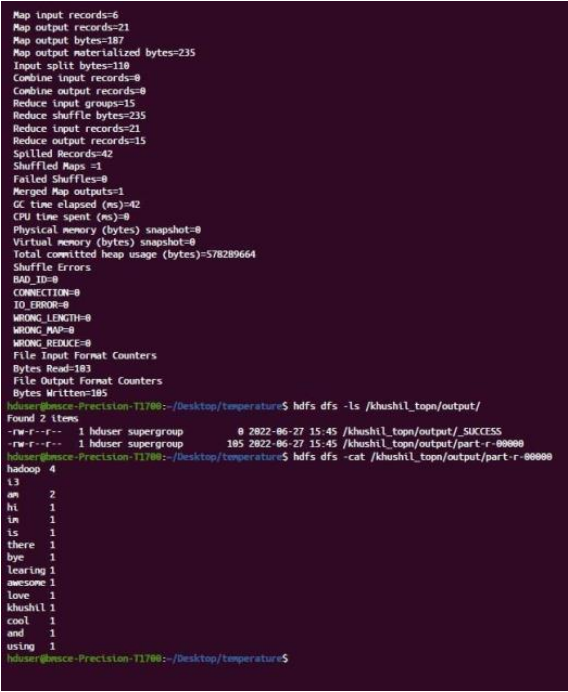
break;

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

}

}

}



**LAB 7**

**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);

}

}

**Average Mapper**

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));

}

}

**Average Reducer**

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));

}

}

**LAB 8**

**MeanMax Driver**

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);

}

}

**MeanMax Mapper**

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));

}

}

**MeanMax Reducer**

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));

}

}

**LAB 9**

**Driver**

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);

}

}

**Reducer**

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);

}

}

}

**Posts**

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]));

}

}

**TextPairs**

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);

}

} }

**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]));

}

}

**LAB 10**

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 11**

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()

}

}