VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data Analytics

Submitted by

RAKSHITHA G (1BM17CS071)

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

<u>CERTIFICATE</u>

This is to certify that the Lab work entitled "BIG DATA ANALYTICS" carried out by **RAKSHITHA G (1BM17CS071),** 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.

Dr. Pallavi G B   **Dr. Jyothi S Nayak**

Assistant Professor   Professor and Head

Department of CSE   Department of CSE

BMSCE,Bengaluru

`

Index Sheet

10	Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark	47-48

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

DB operations using Cassandra (Employee):

cqlsh> create keyspace mployee_space WITH REPLICATION = {'class' :

'SimpleStrategy','replication_factor':2};

CREATE TABLE employee_space.employee_info (emp_id int PRIMARY

KEY,emp_name text,designation text,date_of_joining timestamp,salary

float,dept_name text);

cqlsh> begin batch INSERT INTO

employee_space.employee_info(emp_id,emp_name,designation,date_of_join

i ng,salary,dept_name) VALUES(1,'Damodar','Manager','2022-01-

24',100000,'Marketing');

 ... apply batch;

cqlsh> begin batch INSERT INTO

employee_space.employee_info(emp_id,emp_name,designation,date_of_join

i ng,salary,dept_name) VALUES(2,'Mahalaxmi','Accountant','2021-01-24',200000,'Accounts');

 ... INSERT INTO

employee_space.employee_info(emp_id,emp_name,designation,date_of_joini ng,salary,dept_name) VALUES(3,'Mahesh','Manager','2021-03-24',500000,'Marketing');

 ... INSERT INTO

employee_space.employee_info(emp_id,emp_name,designation,date_of_join i ng,salary,dept_name) VALUES(4,'Nidhi','Administrator','2021-05-24',500000,'Administration');

 ... INSERT INTO

employee_space.employee_info(emp_id,emp_name,designation,date_of_join i ng,salary,dept_name) VALUES(5,'Rahul','Administrator','2009-05-24',2000000,'Administration');

 ... apply batch;

cqlsh> use employee_space;

cqlsh:employee_space> select * from employee_info;

emp_id | date_of_joining | dept_name | designation |

emp_name | salary

     + + + + + 5 | 2009-05-23 18:30:00.000000+0000 | Administration |

Administrator | Rahul | 2e+06

 1 | 2022-01-23 18:30:00.000000+0000 | Marketing | Manager |

Damodar | 1e+05

 2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant |

Mahalaxmi | 2e+05

 4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator |
Nidhi | 5e+05

 3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager |
Mahesh | 5e+05

(5 rows)

cqlsh:employee_space> update employee_info set emp_name='Radha' where
emp_id=1;

cqlsh:employee_space> update employee_info set dept_name='Development'
where emp_id=1;

cqlsh:employee_space> select * from employee_info;

emp_id | date_of_joining | dept_name | designation |
emp_name | salary

      + + + + + 5 | 2009-05-23 18:30:00.000000+0000 | Administration |
Administrator | Rahul | 2e+06

 1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager |
Radha | 1e+05

 2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant |
Mahalaxmi | 2e+05

 4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator |
Nidhi | 5e+05

 3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager |
Mahesh | 5e+05

(5 rows)

```
cqlsh:employee_space> alter table employee_info add projects set<text>;

cqlsh:employee_space> update employee_info set projects=projects+{'Web
development','machine learning'} where emp_id=2;

cqlsh:employee_space> select * from employee_info;
```

emp_id | date_of_joining | dept_name | designation |
emp_name | projects | salary

+ + + + + +

5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator |
Rahul | null | 2e+06

1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager |
Radha | null | 1e+05

2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant |
Mahalaxmi | {'Web development', 'machine learning'} | 2e+05 4 |
2021-05-23 18:30:00.000000+0000 | Administration | Administrator |
Nidhi | null | 5e+05

3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager |
Mahesh | null | 5e+05

(5 rows)

```
cqlsh:employee_space> update employee_info set projects=projects+{'Web
development','machine learning','cybersecurity'} where emp_id=5;

cqlsh:employee_space> select * from employee_info;
```

emp_id | date_of_joining | dept_name | designation |
emp_name | projects | salary

+ + + + + +

 5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | Rahul | {'Web development', 'cybersecurity', 'machine learning'} | 2e+06

1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | Radha | null | 1e+05

 2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | Mahalaxmi | {'Web development', 'machine learning'} | 2e+05 4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | Nidhi | null | 5e+05

 3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | Mahesh | null | 5e+05

(5 rows)

cqlsh:employee_space> INSERT INTO employee_space.employee_info(emp_id,emp_name,designation,date_of_join i ng,salary,dept_name) VALUES(6,'Harshitha','Manager','2022-01-24',100000,'Marketing') using ttl 15;

cqlsh:employee_space> select * from employee_info;

emp_id | date_of_joining | dept_name | designation | emp_name | projects | salary

+ + + + + +

 5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | Rahul | {'Web development', 'cybersecurity', 'machine learning'} | 2e+06

1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | Radha

| null | 1e+05

 2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant |

Mahalaxmi | {'Web development', 'machine learning'} | 2e+05 4 |

2021-05-23 18:30:00.000000+0000 | Administration | Administrator |

Nidhi | null | 5e+05

 6 | 2022-01-23 18:30:00.000000+0000 | Marketing | Manager |
Harshitha | null | 1e+05

 3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager |

Mahesh | null | 5e+05

(6 rows)

cqlsh:employee_space> select * from employee_info;

emp_id | date_of_joining | dept_name | designation |

emp_name | projects | salary

+ + + + + +

 5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator |

Rahul | {'Web development', 'cybersecurity', 'machine learning'} | 2e+06

1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | Radha

| null | 1e+05

 2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant |

Mahalaxmi | {'Web development', 'machine learning'} | 2e+05 4 |

2021-05-23 18:30:00.000000+0000 | Administration | Administrator |

Nidhi | null | 5e+05

 3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager |

Mahesh | null | 5e+05

(5 rows)

DB operations using Cassandra (Library)

cqlsh> create keyspace library_space WITH

REPLICATION={'class':'SimpleStrategy','replication_factor':2};

cqlsh> use library_space;

cqlsh:library_space> create table library_info(stud_id int,counter_value

counter,stud_name text,book_name text,book_id int,date_of_issue

timestamp,PRIMARY

KEY(stud_id,stud_name,book_name,book_id,date_of_issue));

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=1 and stud_name='abc' and book_name='book1' and

book_id=11 and date_of_issue='2022-01-30';

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=2 and stud_name='def' and book_name='book2' and

book_id=12 and date_of_issue='2022-03-30';

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=3 and stud_name='ghi' and book_name='book3' and

book_id=13 and date_of_issue='2022-05-30';

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=4 and stud_name='jkl' and book_name='book4' and

book_id=14 and date_of_issue='2022-07-30';

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=5 and stud_name='mno' and book_name='book5' and

book_id=15 and date_of_issue='2022-09-30';

cqlsh:library_space> select * from library_info;

stud_id | stud_name | book_name | book_id | date_of_issue |

counter_value

        + + + + +

 5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 |

1

 1 | abc | book1 | 11 | 2022-01-29 18:30:00.000000+0000

| 1

 2 | def | book2 | 12 | 2022-03-29 18:30:00.000000+0000

| 1

 4 | jkl | book4 | 14 | 2022-07-29 18:30:00.000000+0000 | 1

 3 | ghi | book3 | 13 | 2022-05-29 18:30:00.000000+0000 |

1

(5 rows)

cqlsh:library_space> update library_info set counter_value=counter_value+1

where stud_id=5 and stud_name='mno' and book_name='book5' and

book_id=15 and date_of_issue='2022-09-30';

cqlsh:library_space> select * from library_info;

stud_id | stud_name | book_name | book_id | date_of_issue |

counter_value

        + + + + +

 5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 |

2

 1 | abc | book1 | 11 | 2022-01-29 18:30:00.000000+0000

| 1

 2 | def | book2 | 12 | 2022-03-29 18:30:00.000000+0000 |

1

 4 | jkl | book4 | 14 | 2022-07-29 18:30:00.000000+0000 | 1

 3 | ghi | book3 | 13 | 2022-05-29 18:30:00.000000+0000 |

1

(5 rows)

cqlsh:library_space> copy

library_info(stud_id,stud_name,book_name,book_id,date_of_issue,counter_

v alue) to '/home/bmscecse/Desktop/bda.csv';

Using 11 child processes

Starting copy of library_space.library_info with columns [stud_id, stud_name,

book_name, book_id, date_of_issue, counter_value].

Processed: 5 rows; Rate: 45 rows/s; Avg. rate: 45 rows/s

5 rows exported to 1 files in 0.121 seconds.

cqlsh:library_space> create table library_info_copy(stud_id int,counter_value

counter,stud_name text,book_name text,book_id int,date_of_issue

timestamp,PRIMARY

KEY(stud_id,stud_name,book_name,book_id,date_of_issue));

cqlsh:library_space> copy

library_info_copy(stud_id,stud_name,book_name,book_id,date_of_issue,cou

n ter_value) from '/home/bmscecse/Desktop/new.csv';

Using 11 child processes

Starting copy of library_space.library_info_copy with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].

Processed: 5 rows; Rate: 8 rows/s; Avg. rate: 12 rows/s

5 rows imported from 1 files in 0.406 seconds (0 skipped).

cqlsh:library_space> select * from library_info where counter_value=2 allow filtering;

stud_id | stud_name | book_name | book_id | date_of_issue | counter_value

+ + + + +

 5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 | 2

MongoDB- CRUD Demonstration

use my_db

switched to db my_db

db.Student.insert({_id:1,name:"Michael",grade :"VII",hobbies:"reading"})

WriteResult({ "nInserted" : 1 })

db.Student.update({_id:1},{$set:{hobbies:"crick et"}},{upsert:true})

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

```
db.Student.find()
{ "_id" : 1, "name" : "Michael", "grade" : "VII",
"hobbies" : "cricket" }
db.Student.insert({id:1,name:"Latha",grade:"VII
I",hobbies:"Singing"})
WriteResult({ "nInserted" : 1 })
db.Student.find({name:"Latha"}).pretty()
{
"_id" :
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
db.Student.find({},{name:1,grade:1,_id:0}) {
"name" : "Michael", "grade" : "VII" } { "name"
: "Latha", "grade" : "VIII" }
db.Student.find({grade:{$eq:"VII"}}).pretty() {
"_id" : 1, "name" : "Michael", "grade" : "VII",
"hobbies" : "cricket" }
db.Student.find({name:/^L/}).pretty()
{
```

```
"_id" :
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
db.Student.find({name:/a/}).pretty() { "_id" :
1, "name" : "Michael", "grade" : "VII",
"hobbies" : "cricket" }
{
"_id" :
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
db.Student.count()
2
db.Student.find().sort({name:1}).pretty()
{
```

```
"_id" :
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
{ "_id" : 1, "name" : "Michael", "grade" : "VII",
"hobbies" : "cricket" }
db.Student.save({name:"Ratan",grade:"VII",_id
: 1})
WriteResult({ "nMatched" : 1, "nUpserted" : 0,
"nModified" : 1 })
db.Student.find()
{ "_id" : 1, "name" : "Ratan", "grade" : "VII" }
{ "_id" :
ObjectId("6253f120f7936958d67f3c07"), "id" :
1, "name" : "Latha", "grade" : "VIII", "hobbies" :
"Singing" }
db.Student.update({_id:1},{$set:{location:"net
work"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0,
"nModified" : 1 })
```

```
db.Student.update({_id:1},{$unset:{location:"

n etwork"}})

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

"nModified" : 1 })

db.Student.find({name:/n$/}).pretty() { "_id"

: 1, "name" : "Ratan", "grade" : "VII" }

db.Student.find({grade:"VII"}).limit(3).pretty()

{ "_id" : 1, "name" : "Ratan", "grade" : "VII" }

db.Student.count({grade:"VIII"})

1

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

{

"_id" :

ObjectId("6253f120f7936

958d67f3c07"),

"id" : 1,

"name" : "Latha",

"grade" : "VIII",

"hobbies" : "Singing"

}

{ "_id" : 1, "name" : "Ratan", "grade" : "VII" }

db.Student.find().sort({name:-1}).pretty() {

"_id" : 1, "name" : "Ratan", "grade" : "VII" } {

"_id" :
```

```
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
db.Student.find().skip(1).pretty()
{
"_id" :
ObjectId("6253f120f7936
958d67f3c07"),
"id" : 1,
"name" : "Latha",
"grade" : "VIII",
"hobbies" : "Singing"
}
db.createCollection("food")
{ "ok" : 1 }
db.food.insert({_id:1,fruits:['grapes','mango']}
) WriteResult({ "nInserted" : 1 })
db.food.insert({_id:2,fruits:['grapes','mango','
c herry']})
WriteResult({ "nInserted" : 1 })
```
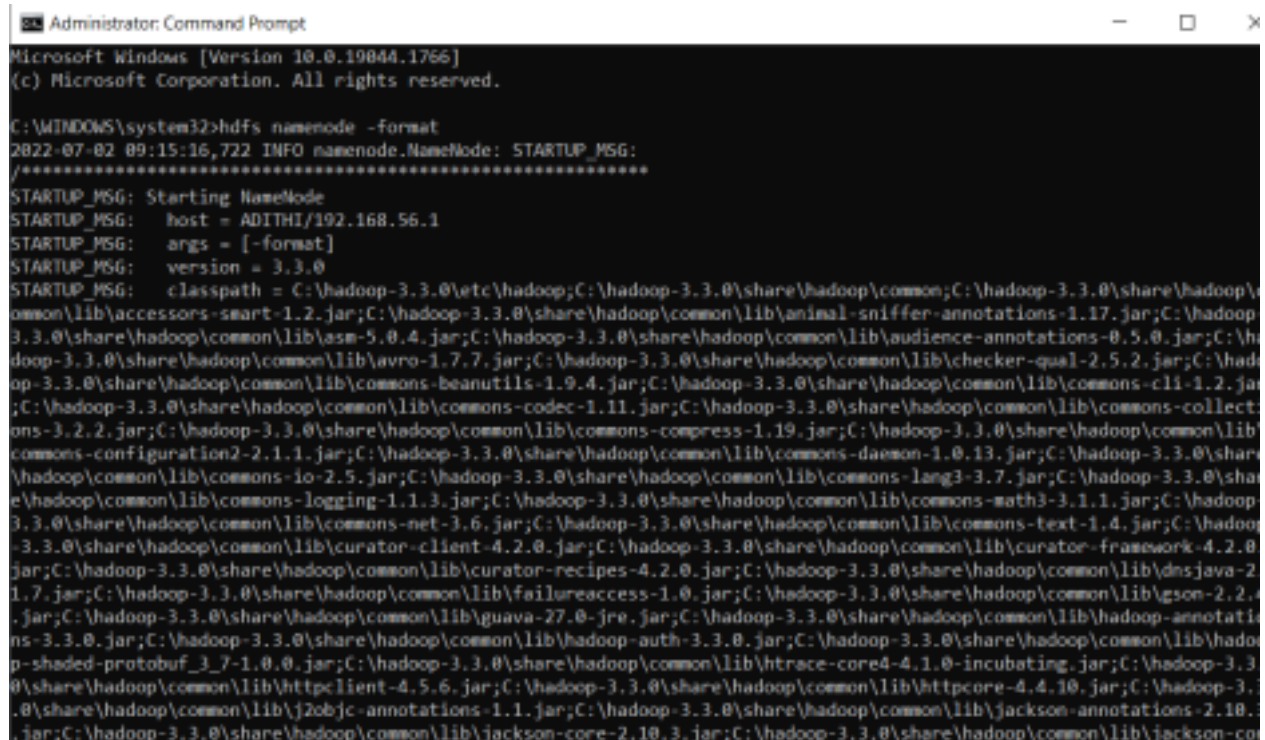
```
db.food.insert({_id:3,fruits:['banana','cherry']}
) WriteResult({ "nInserted" : 1 })
db.food.find({fruits:['grapes','mango']}) {
"_id" : 1, "fruits" : [ "grapes", "mango" ] }
db.food.find({'fruits':{$size:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
{ "_id" : 3, "fruits" : [ "banana", "cherry" ] }
db.food.find({_id:2},{'fruits':{$slice:2}})
{ "_id" : 2, "fruits" : [ "grapes", "mango" ] }
db.food.find({fruits:{$all:['grapes','mango']}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] } {
"_id" : 2, "fruits" : [ "grapes", "mango",
"cherry" ] }
db.food.update({_id:3},{$set:{'fruits.1':'apple'}}
) WriteResult({ "nMatched" : 1, "nUpserted" :
0, "nModified" : 1 })
db.food.find()
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango",
"cherry" ] }
{ "_id" : 3, "fruits" : [ "banana", "apple" ] }
db.food.update({_id:2},{$push:{price:{grapes:8
0,mango:200,cherry:100}}})
```

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

"nModified" : 1 })

Screenshot of Hadoop installed

Execution of HDFS Commands for interaction with Hadoop Environment.

$ start-all.sh

$ jps

4193 ResourceManager

4691 Jps

3876 SecondaryNameNode

4566 NodeManager

3050 NameNode

3391 DataNode

```
$ hdfs dfs -mkdir /xyz


$ hadoop fs -ls /

Found 13 items

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:48 /FFF drwxr-xr-x

- hduser supergroup 0 2022-06-04 10:10 /abc drwxr-xr-x - hduser

supergroup 0 2022-06-03 14:39 /folder1 drwxr-xr-x - hduser

supergroup 0 2022-06-03 15:00 /folder2 drwxr-xr-x - hduser

supergroup 0 2022-06-03 15:00 /folder3 drwxr-xr-x - hduser

supergroup 0 2022-06-01 14:48 /lab1 drwxr-xr-x - hduser

supergroup 0 2019-10-24 11:08 /output drwxr-xr-x - hduser

supergroup 0 2022-06-01 14:48 /pratibha drwxr-xr-x - hduser

supergroup 0 2019-10-24 10:47 /rgs drwxr-xr-x - hduser supergroup

0 2022-06-03 12:05 /test drwxrwxr-x - hduser supergroup 0

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

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

2022-06-06 11:33 /xyz


$ hdfs dfs -put /home/hduser/Desktop/welcome.txt /xyz/Wel.txt


$ hdfs dfs -cat /abc/WC.txt

Hadoop lab
```

```
$ hdfs dfs -copyFromLocal /home/hduser/Desktop/welcome.txt /xyz/Wel.txt

copyFromLocal: `/xyz/Wel.txt': File exists

$ hdfs dfs -get /abc/WC.txt /home/hduser/Downloads/wwc.txt


$ hdfs dfs -getmerge /abc/WC.txt /abc/WC2.txt
/home/hduser/Desktop/Merge.txt


$ hadoop fs -getfacl /abc/

# file: /abc

# owner: hduser

# group: supergroup

 user::rwx

group::r-x

other::r-x


$ hdfs dfs -copyToLocal /abc/WC2.txt /home/hduser/Desktop


$ hadoop fs -mv /abc /FFF

$ hdfs dfs -ls /FFF

Found 3 items

-rw-r--r-- 1 hduser supergroup 11 2022-06-04 09:42 /FFF/WC.txt

-rw-r--r-- 1 hduser supergroup 20 2022-06-04 09:48 /FFF/WC2.txt

drwxr-xr-x - hduser supergroup 0 2022-06-04 10:10 /FFF/abc
```

```
$ hadoop fs -cp /FFF/ /xxx

$ hadoop fs -ls /xxx
Found 3 items

-rw-r--r-- 1 hduser supergroup 11 2022-06-06 12:19 /xxx/WC.txt

-rw-r--r-- 1 hduser supergroup 20 2022-06-06 12:19 /xxx/WC2.txt

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:19 /xxx/abc
```

To use Hadoop to find the average temperature and mean max temperature for each year from NCDC data set.

Average temperature:

AverageDriver:

```java
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");
```

```java
        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:

```java
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,
```

```java
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:

```java
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,
```
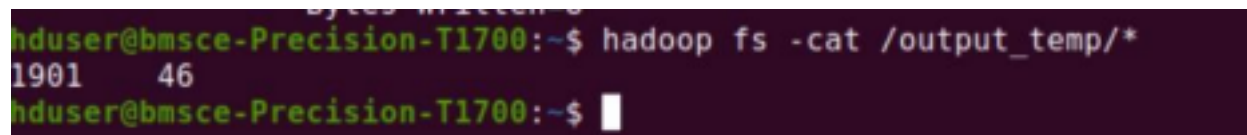
Text, IntWritable>.Context context) throws IOException, InterruptedException

{

int max_temp = 0;
int count = 0;

for (IntWritable value : values) {

max_temp += value.get();

count++;

}

context.write(key, new IntWritable(max_temp / count));

}

}

```
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output_temp/*
1901    46
hduser@bmsce-Precision-T1700:~$
```

Mean Max:

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;

```java
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;

```java
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;
```
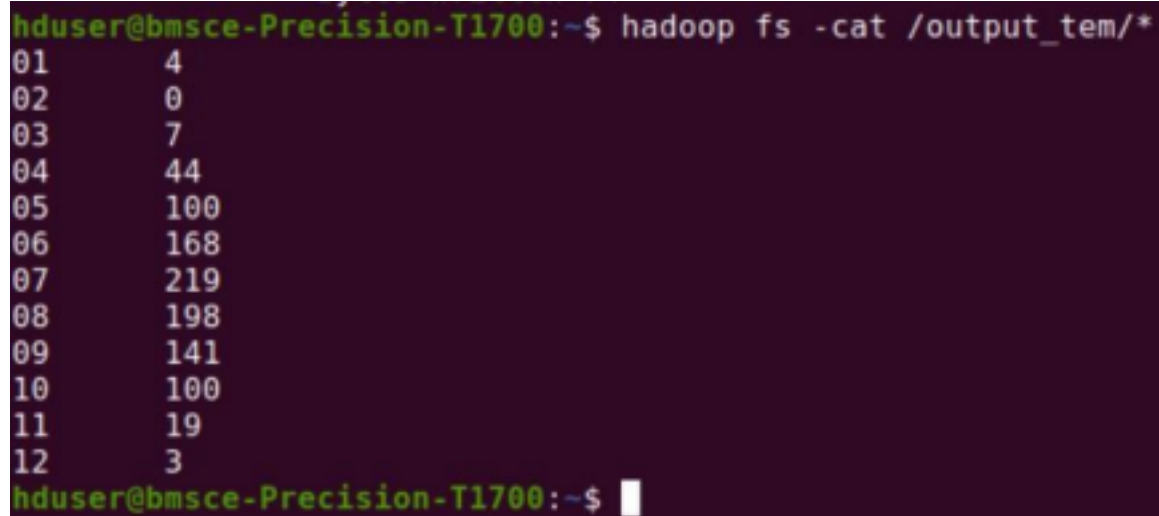
```java
import org.apache.hadoop.io.Text;

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));

}

}

```
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output_tem/*
01      4
02      0
03      7
04      44
05      100
06      168
07      219
08      198
09      141
10      100
11      19
12      3
hduser@bmsce-Precision-T1700:~$
```

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.

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;

```java
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();
```

```java
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

```java
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> {
```

```java
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

```java
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 = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-
:()?!\"&#39;]";

public void map(Object key, Text value, Mapper<Object, Text, Text,
IntWritable>.Context

context) throws IOException, InterruptedException {
```

```java
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

```java
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,
```

```java
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));

}

}

}
```

```
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output8/*
how      5
your     4
is       4
brother  1
are      1
hi       1
sister   1
family   1
you      1
job      1
hduser@bmsce-Precision-T1700:~$
```

Create a Map Reduce program to demonstrating join

operation // 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;

}

```java
}
@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);
```

```java
        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

    {
```

```java
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

```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,
```

```java
Text&gt; {

@Override

public void map(LongWritable key, Text value, OutputCollector&lt;TextPair,
Text&gt; output,

 Reporter reporter)

throws IOException

{

String valueString = value.toString();
String[] SingleNodeData = valueString.split(&quot;\t&quot;);

output.collect(new TextPair(SingleNodeData[0], &quot;1&quot;), 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&lt;LongWritable,
Text,

TextPair,

Text&gt; {

@Override

public void map(LongWritable key, Text value, OutputCollector&lt;TextPair,
Text&gt; output,

 Reporter reporter)
```

```java
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) {
```

```java
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();

}
```

```java
@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();
```

```java
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 {
```

```java
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);

}

} }
```

```
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output_mapreduce/*
A11      50                 Finance
B12      100                HR
C13      250                Manufacturing
Dept_ID Total_Employee         Dept_Name
hduser@bmsce-Precision-T1700:~$
```

Program to print word count on scala shell and print "Hello world" on scala IDE

Word Count:

```
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;
```



```
scala> reducedata.collect;
res8: Array[(String, Int)] = Array(("",1), (hello,5), (lab,3), (begin,3), (spark
,5), (9,1))
```

Hello World:

```
object ExPrint {

    def main(args: Array[String]) {

       println("Hello World!");

    }
```

```
    }
    package word_count

 object count {
    def main(args : Array[String]){
    println("Hello World!");

    }
 }
```

```
Problems  Tasks  Console 
<terminated> count$ [Scala Application] /usr/lib/jvm/java
Hello World!
```

**Using RDD and FlatMap count how many times each word appears
in a file and write out a list of words whose count is strictly greater
than 4 using Spark**

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()

}

}
```

```
scala> println(sorted)
Map(hello -> 5, spark -> 5, lab -> 3, begin -> 3,  -> 1, 9 -> 1)

scala> for((k,v)<-sorted)
     | {
     | if(v>4)
     | {
     | print(k+",")
     | print(v)
     | println()
     | }
     | }
hello,5
spark,5
```