VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

Mahantesh Gattina (1BM19CS219)

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 **Mahantesh Gattina (1BM19CS219),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 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. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Index Sheet

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

Program 1: Employee Database using Cassandra

1. Create a keyspace by name Employee

CREATE KEYSPACE employee WITH REPLICATION={ 'class' : 'SimpleStrategy', 'replication_factor' : 1};

USE employee;

2. 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, designation text, date_of_joining timestamp, salary double, dept_name text);

3. Insert the values into the table in batch

BEGIN BATCH

INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES(100,'TANYA','MANAGER','2020-09-11',30000,'TESTING')

... INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES(111,'SRIRAM','ASSOCIATE','2020-06-22',25000,'DEVELOPING')

... INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES(121,'SHIVA','MANAGER','2020-03-30',35000,'HR')

... APPLY BATCH;

SELECT * FROM employee_info;

4. Update Employee name and Department of Emp-Id 121

UPDATE employee_info SET emp_name = 'SHAAN' WHERE emp_id = 121; SELECT * FROM employee_info;

5. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

ALTER TABLE employee_info ADD projects text;

6. Update the altered table to add project names.

UPDATE employee_info SET projects = 'chat app' WHERE emp_id = 111;

UPDATE employee_info SET projects = 'campusx' WHERE emp_id = 121;

UPDATE employee_info SET projects = 'canteen app' WHERE emp_id = 100;

SELECT * FROM employee_info;

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

INSERT INTO
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
VALUES(110,'SAM','ASSOCIATE','2020-01-11',33000,'TESTING') USING TTL 15;

SELECT TTL(emp_name) from employee_info WHERE emp_id = 110; SELECT * FROM
employee_info;

Program 2:

Library Database using Cassandra

1.Create a keyspace by name Library

CREATE KEYSPACE library WITH REPLICATION={ 'class' : 'SimpleStrategy', 'replication_factor' : 1};

USE library;

2. Create a column family by name Library-Info with attributes Stud_Id Primary Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue

create table library_info(stud_id int, counter_value Counter, stud_name text,book_name text, date_of_issue timestamp, book_id int, PRIMARY KEY(stud_id,stud_name,book_name,date_of_issue,book_id));

3. Insert the values into the table in batch

UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 111 and stud_name = 'SAM' and book_name = 'ML' and date_of_issue = '2020-10-11'and book_id = 200;

UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21'and book_id = 300;

UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 113 and stud_name = 'AYMAN' and book_name = 'OOMD' and date_of_issue = '2020-04-01'and book_id = 400;

SELECT * FROM library_info;

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

UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21'and book_id = 300;

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

SELECT * FROM library_info WHERE stud_id = 112;

6. Export the created column to a csv file

COPY Library_Info(Stud_Id,Stud_Name,Book_Name,Book_Id,Date_Of_Issue,Counter_val ue) TO 'e:\libraryInfo.csv';

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

```
create table library_info2(stud_id int, counter_value Counter, stud_name text,book_name text,
date_of_issue timestamp, book_id int, PRIMARY
KEY(stud_id,stud_name,book_name,date_of_issue,book_id));

COPY library_info2(stud_id,stud_name,book_name,book_id,date_of_issue,counter _value)
FROM 'e:\libraryInfo.csv';
```

Program 3 :

Student MongoDB Program

```
> use mySTUD;

switched to db mySTUD

> db.getCollectionNames()

[ ]

> db.createCollection("Student");

{ "ok" : 1 }

> db.getCollectionNames()

[ "Student" ]

> db.Student.insert({_id: 1, Name:"John", USN: "1B22CS001",Semester: 6,Dept_name: "CSE", CGPA: 9.6, Hobbies : ["Reading","Gardening"]})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({_id: 4, Name:"Arthur", USN: "1B22CS041",Semester: 6,Dept_name: "CSE", CGPA: 8.6, Hobbies : ["Novel Reading"]})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({_id: 3, Name:"Horris", USN: "1B22EE021",Semester: 5,Dept_name: "EEE", CGPA: 9.3, Hobbies : ["eSports"]})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({_id: 7, Name:"Hritik", USN: "1B22CS014",Semester: 5,Dept_name: "CSE", CGPA: 8.7, Hobbies : ["Reading"]})

WriteResult({ "nInserted" : 1 })

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

{
        "_id" : 1,
        "Name" : "John",
        "USN" : "1B22CS001",
        "Semester" : 6,
```

```
        "Dept_name" : "CSE",
        "CGPA" : 9.6,
        "Hobbies" : [
                "Reading",
                "Gardening"
        ]
}
{
        "_id" : 4,
        "Name" : "Arthur",
        "USN" : "1B22CS041",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.6,
        "Hobbies" : [
                "Novel Reading"
        ]
}
{
        "_id" : 3,
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
        "Hobbies" : [
                "eSports"
        ]
```

```
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
        "Semester" : 5,
        "Dept_name" : "CSE",
        "CGPA" : 8.7,
        "Hobbies" : [
                "Reading"
        ]
}
```


```
> db.Student.update({_id: 3, Name:"Horris", USN: "1B22EE021",Semester: 5,Dept_name:
"EEE", CGPA: 9.3},{$set:{Hobbies:"Skating"}},{upset:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find().pretty()
```


```
{
        "_id" : 1,
        "Name" : "John",
        "USN" : "1B22CS001",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 9.6,
        "Hobbies" : [
                "Reading",
                "Gardening"
        ]
}
```

```
{
        "_id" : 4,
        "Name" : "Arthur",
        "USN" : "1B22CS041",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.6,
        "Hobbies" : [
                "Novel Reading"
        ]
}
{
        "_id" : 3,
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
        "Hobbies" : "Skating"
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
        "Semester" : 5,
        "Dept_name" : "CSE",
        "CGPA" : 8.7,
        "Hobbies" : [
```

```
                "Reading"
        ]
}
> db.Student.find({},{StudName:1,Semester:1,_id:0});
{ "Semester" : 6 }
{ "Semester" : 6 }
{ "Semester" : 5 }
{ "Semester" : 5 }
> db.Student.find({},{Name:1,Semester:1,_id:0});
{ "Name" : "John", "Semester" : 6 }
{ "Name" : "Arthur", "Semester" : 6 }
{ "Name" : "Horris", "Semester" : 5 }
{ "Name" : "Hritik", "Semester" : 5 }

> db.Student.find({Semester:{$eq:5}}).pretty();
{
        "_id" : 3,
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
        "Hobbies" : "Skating"
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
```

```
        "Semester" : 5,
        "Dept_name" : "CSE",
        "CGPA" : 8.7,
        "Hobbies" : [
                "Reading"
        ]
}
> db.Student.count();
4
> db.Student.find().sort({Name:-1}).pretty();
{
        "_id" : 1,
        "Name" : "John",
        "USN" : "1B22CS001",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 9.6,
        "Hobbies" : [
                "Reading",
                "Gardening"
        ]
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
        "Semester" : 5,
        "Dept_name" : "CSE",
```

```
        "CGPA" : 8.7,
        "Hobbies" : [
                "Reading"
        ]
}
{
        "_id" : 3,
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
        "Hobbies" : "Skating"
}
{
        "_id" : 4,
        "Name" : "Arthur",
        "USN" : "1B22CS041",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.6,
        "Hobbies" : [
                "Novel Reading"
        ]
}
```


```
(base) bmsce@bmsce-Precision-T1700:~$ mongoexport --host localhost --db mySTUD
--collection Student --type=csv --fields="_id,Name,USN,Semester,Dept_name,CGPA,Hobbies"
--out /home/bmsce/Desktop/output.csv
```

2022-05-06T12:13:37.350+0530 connected to: localhost

2022-05-06T12:13:37.351+0530 exported 4 records

(base) bmsce@bmsce-Precision-T1700:~$ mongo

MongoDB shell version v3.6.8

connecting to: mongodb://127.0.0.1:27017

Implicit session: session { "id" : UUID("aabd8226-3ced-43d4-97fb-b0d55827849c") }

MongoDB server version: 3.6.8

Server has startup warnings:

2022-05-06T11:28:08.073+0530 I STORAGE [initandlisten]

2022-05-06T11:28:08.073+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine

2022-05-06T11:28:08.073+0530 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem

2022-05-06T11:28:13.281+0530 I CONTROL [initandlisten]

2022-05-06T11:28:13.281+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.

2022-05-06T11:28:13.281+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.

2022-05-06T11:28:13.281+0530 I CONTROL [initandlisten]

> use mySTUD;

switched to db mySTUD

> db.Student.update({_id:4},{$set:{Location:”Network”}})

2022-05-06T12:16:35.289+0530 E QUERY [thread1] SyntaxError: illegal character @(shell):1:42

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

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

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

{

"_id" : 1,

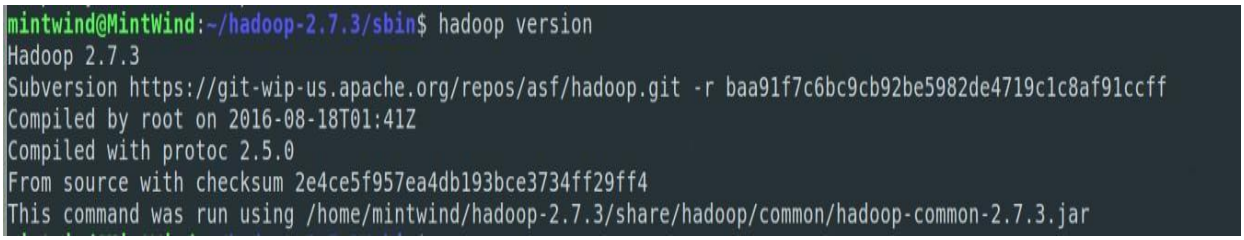
"Name" : "John",

```
        "USN" : "1B22CS001",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 9.6,
        "Hobbies" : [
                "Reading",
                "Gardening"
        ]
}
{
        "_id" : 4,
        "Name" : "Arthur",
        "USN" : "1B22CS041",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.6,
        "Hobbies" : [
                "Novel Reading"
        ],
        "Location" : "Network"
}
{
        "_id" : 3,
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
```

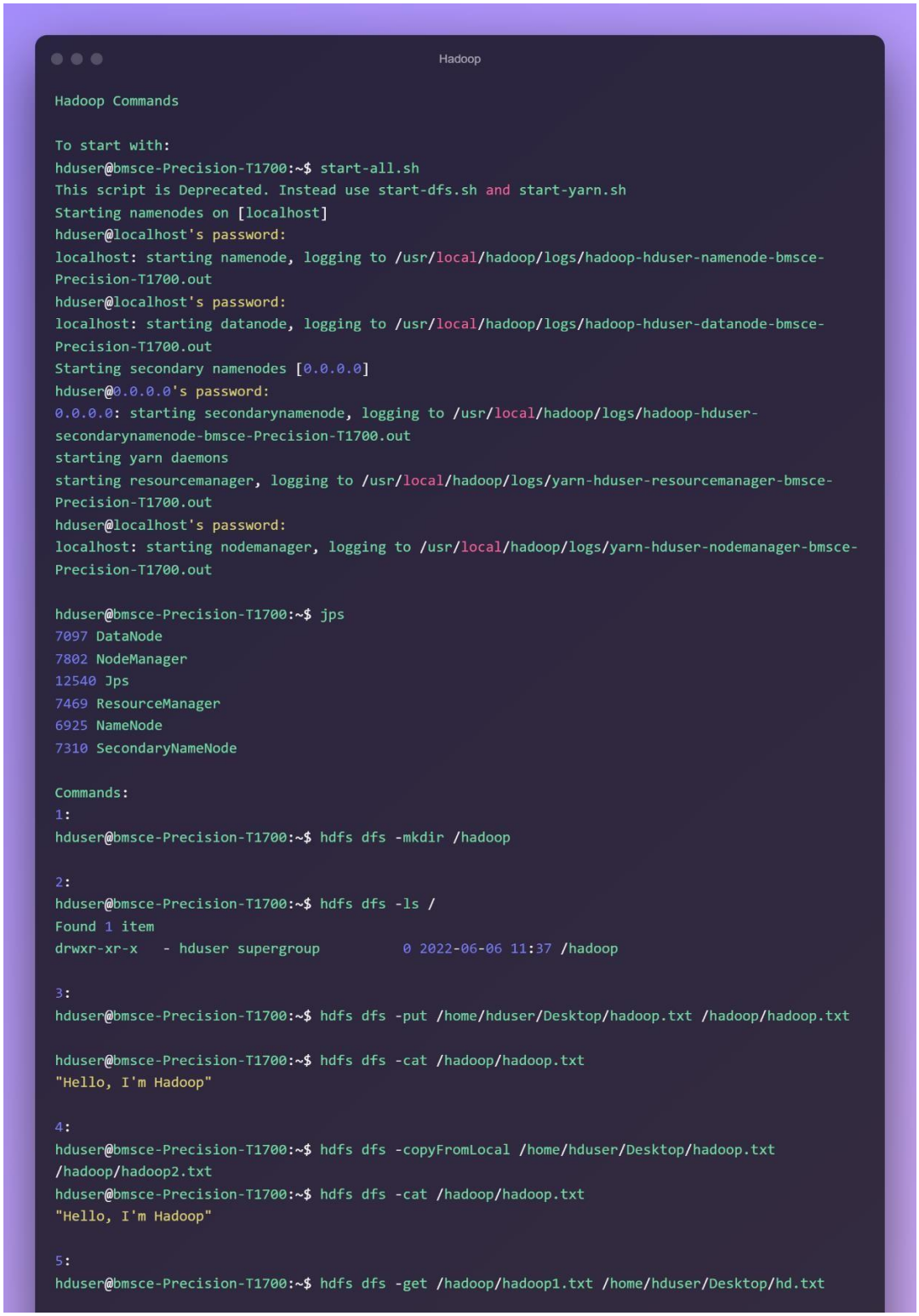
```
        "Hobbies" : "Skating"
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
        "Semester" : 5,
        "Dept_name" : "CSE",
        "CGPA" : 8.7,
        "Hobbies" : [
                "Reading"
        ]
}
> db.Student.find().sort({Name:1}).pretty();
{
        "_id" : 4,
        "Name" : "Arthur",
        "USN" : "1B22CS041",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.6,
        "Hobbies" : [
                "Novel Reading"
        ],
        "Location" : "Network"
}
{
        "_id" : 3,
```

```
        "Name" : "Horris",
        "USN" : "1B22EE021",
        "Semester" : 5,
        "Dept_name" : "EEE",
        "CGPA" : 9.3,
        "Hobbies" : "Skating"
}
{
        "_id" : 7,
        "Name" : "Hritik",
        "USN" : "1B22CS014",
        "Semester" : 5,
        "Dept_name" : "CSE",
        "CGPA" : 8.7,
        "Hobbies" : [
                "Reading"
        ]
}
{
        "_id" : 1,
        "Name" : "John",
        "USN" : "1B22CS001",
        "Semester" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 9.6,
        "Hobbies" : [
                "Reading",
                "Gardening" }
```

Screenshot of Hadoop installed:



Execution of HDFS Commands for interaction with Hadoop Environment:



```
hduser@bmsce-Precision-T1700:~$ hdfs dfs -getmerge /hadoop/hadoop.txt /hadoop/hadoop2.txt
/home/hduser/Desktop/hd_merge.txt
hduser@bmsce-Precision-T1700:~$ ls Desktop/hd_merge.txt
Desktop/hd_merge.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -getfacl /hadoop
# file: /hadoop
# owner: hduser
# group: supergroup
user::rwx
group::r-x
other::r-x


6:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /hadoop/hadoop.txt
/home/hduser/Desktop/hd2.txt

hduser@bmsce-Precision-T1700:~$ ls Desktop/hd2.txt
Desktop/hd2.txt

7:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /hadoop/hadoop.txt
"Hello, I'm Hadoop"

8:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir /hadoop/AA
hduser@bmsce-Precision-T1700:~$ hdfs dfs -mv /hadoop/hadoop.txt /hadoop/AA/hadoop.txt
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /hadoop/AA
Found 1 items
-rw-r--r--   1 hduser supergroup         18 2022-06-06 11:41 /hadoop/AA/hadoop.txt

9:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cp /hadoop/AA/hadoop.txt /hadoop/hadoop2.txt
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /hadoop/hadoop2.txt
Hello, I'm Hadoop

To stop Hadoop:
hduser@bmsce-Precision-T1700:~$ stop-all.sh
This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh
Stopping namenodes on [localhost]
hduser@localhost's password:
localhost: stopping namenode
hduser@localhost's password:
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
hduser@0.0.0.0's password:
0.0.0.0: stopping secondarynamenode
stopping yarn daemons
stopping resourcemanager
hduser@localhost's password:
localhost: stopping nodemanager
no proxyserver to stop
```

Map Reduce program for weather data:

(a) Average temperature for each year:

mapper.py

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    line = line.strip()
    year = line[15:19]

    if line[87] == '+':
        temperature = int(line[88:92])
    else:
        temperature = int(line[87:92])

    quality = line[92:93]

    if temperature != 9999 and quality in "[01459]":
        print(year+"\t"+str(temperature))
```

reducer.py

```
#!/usr/bin/python
import sys
cur_year = None
average_temp = 0
count = 0

for line in sys.stdin:
    line = line.strip()
    year, temperature = line.split("\t",1)
    if cur_year == None:
        cur_year = year
    elif cur_year != year:
        print(cur_year+"\t"+str(average_temp // count))
        average_temp = 0
        count = 0
    average_temp += int(temperature)
    count += 1

if cur_year == year:
    print(cur_year+"\t"+str(average_temp // count))
```

Output:

```
mintwind@MintWind:~$ hdfs dfs -ls /prog3
Found 2 items
-rw-r--r--   1 mintwind supergroup          0 2022-07-10 16:00 /prog3/_SUCCESS
-rw-r--r--   1 mintwind supergroup          8 2022-07-10 16:00 /prog3/part-00000
mintwind@MintWind:~$ hdfs dfs -cat /prog3/part-00000
1901    46
```

(a) Mean max temperature for every month:

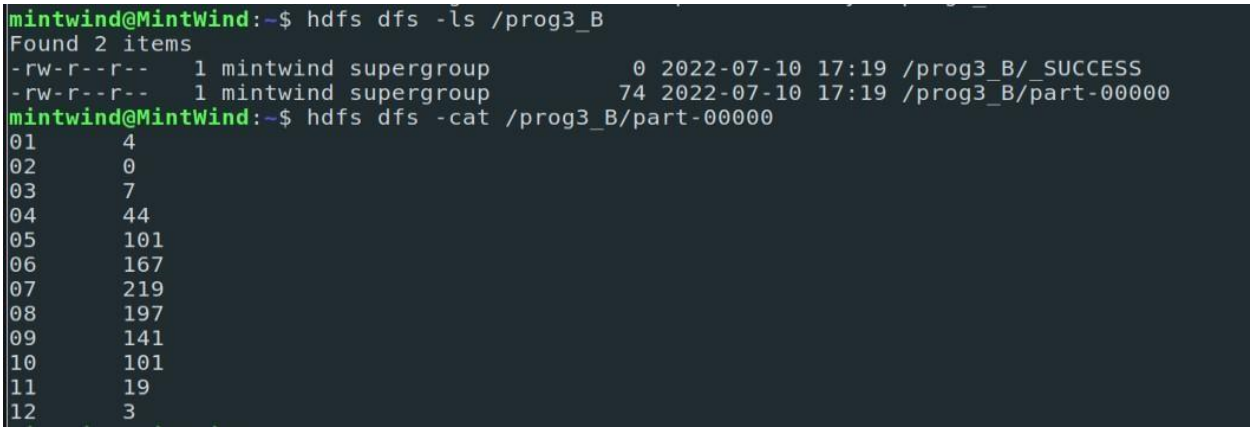
mapper.py

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    line = line.strip()
    month = line[19:21]

    if line[87] == '+':
        temperature = int(line[88:92])
    else:
        temperature = int(line[87:92])

    quality = line[92]

    if temperature != 9999 and quality in "[01459]":
        print(month+"\t"+str(temperature))
```

reducer.py

```
#!/usr/bin/python
import sys

cur_month = None
max_temp = 0
temp_sum = 0
count = 0
days = 0

for line in sys.stdin:
    line = line.strip()
    month, temperature = line.split("\t", 1)
    if cur_month == None:
        cur_month = month
    elif cur_month != month:
        print(cur_month+"\t"+str(temp_sum//days))
        cur_month = month
        max_temp = 0
        temp_sum = 0
        count= 0
        days = 0
    if int(temperature) > max_temp:
        max_temp = int(temperature)
    count += 1

    if count == 3:
        temp_sum += max_temp
        max_temp = 0
        count=0
        days += 1

if cur_month == month:
        print(cur_month+"\t"+str(temp_sum//days))
```

Output:


mintwind@MintWind:~$ hdfs dfs -ls /prog3_B
Found 2 items
-rw-r--r--   1 mintwind supergroup   0 2022-07-10 17:19 /prog3_B/_SUCCESS
-rw-r--r--   1 mintwind supergroup   74 2022-07-10 17:19 /prog3_B/part-00000
mintwind@MintWind:~$ hdfs dfs -cat /prog3_B/part-00000
01	4
02	0
03	7
04	44
05	101
06	167
07	219
08	197
09	141
10	101
11	19
12	3

Map Reduce program - Top N:

mapper.py

```python
#!/usr/bin/python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print(word+"\t"+str(1))
```

reducer.py

```python
#!/usr/bin/python
import sys

current_word = None
current_count = 0
word = None
word_map = []
N = 20

for line in sys.stdin:
    line = line.strip()
    word, count = line.split("\t", 1)
    try:
        count = int(count)
    except ValueError:
        continue

    if current_word == word:
        current_count += 1
    else:
        if current_word:
            word_map.append([(current_count), current_word])
        current_count = count
        current_word = word

if current_word == word:
    word_map.append([(current_count), current_word])

word_map.sort(reverse=True)
for v, k in word_map:
    print("%s\t%d" % (k, v))
```

Output:

```
mintwind@MintWind:~$ hdfs dfs -ls /prog2
Found 2 items
-rw-r--r--   1 mintwind supergroup          0 2022-07-10 15:13 /prog2/_SUCCESS
-rw-r--r--   1 mintwind supergroup         31 2022-07-10 15:13 /prog2/part-00000
mintwind@MintWind:~$ hdfs dfs -cat /prog2/part-00000
hello   2
world   1
hadoop  1
bye     1
```

Map Reduce program to demonstrating join operation:

mapper.py

```python
#!/usr/bin/python

import sys

for line in sys.stdin:
    dept_ID = "-1"  # default sorted as first
    dept_Name = "-1"  # default sorted as first
    no_Emp = "-1"  # default sorted as first

    line = line.strip()

    splits = line.split("\t")

    if splits[-1].isdigit():  # dept strength data
        dept_ID = splits[0]
        no_Emp = str(splits[1])
    else:
        dept_ID = splits[0]
        dept_Name = str(splits[1])

    print('%s^%s^%s' % (dept_ID, dept_Name, no_Emp))
```

reducer.py

```python
#!/usr/bin/python

import sys

new_list = {}

for line in sys.stdin:
    line = line.strip()
    dept_ID, dept_Name, no_Emp = line.split("^")
    if dept_ID not in new_list.keys():
        new_list[dept_ID] = [dept_Name, int(no_Emp)]
    else:
        if dept_Name != -1:
            new_list[dept_ID][0] = dept_Name
        if no_Emp != -1:
            if new_list[dept_ID][1] != -1:
                new_list[dept_ID][1] += int(no_Emp)
            else:
                new_list[dept_ID][1] = int(no_Emp)


for i in new_list:
    print(i+"\t"+new_list[i][0]+"\t"+str(new_list[i][1]))
```

Output:

```
mintwind@MintWind:~$ hdfs dfs -ls /prog4
Found 2 items
-rw-r--r--   1 mintwind supergroup          0 2022-07-10 18:40 /prog4/_SUCCESS
-rw-r--r--   1 mintwind supergroup         47 2022-07-10 18:40 /prog4/part-00000
mintwind@MintWind:~$ hdfs dfs -cat /prog4/part-00000
C13     Manufacturing   249
B12     HR      99
A11     Finance 49
```

Word count on Scala shell:

Scala-Shell

```
val data=sc.textFile("D:\\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;
```

Output:

```
scala> val data=sc.textFile("D:\\sparkdata.txt")
data: org.apache.spark.rdd.RDD[String] = D:\sparkdata.txt MapPartitionsRDD[6] at textFile at <console>:23

scala> data.collect
res5: Array[String] = Array(Hello World BMSCE Lion Tiger Fish)

scala> val splitdata = data.flatMap(line => line.split(" "))
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[7] at flatMap at <console>:23

scala> splitdata.collect
res6: Array[String] = Array(Hello, World, BMSCE, Lion, Tiger, Fish)

scala> val mapdata = splitdata.map(word => (word,1))
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[8] at map at <console>:23

scala> mapdata.collect
res7: Array[(String, Int)] = Array((Hello,1), (World,1), (BMSCE,1), (Lion,1), (Tiger,1), (Fish,1))

scala> val reducedata = mapdata.reduceByKey(_+_)
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[9] at reduceByKey at <console>:23

scala> reducedata.collect
res8: Array[(String, Int)] = Array((Fish,1), (Hello,1), (Lion,1), (BMSCE,1), (World,1), (Tiger,1))
```

RDD and Flat Map count how many times each word appears strictly greater than 4 times:

Scala-Shell

```
val textFile = sc.textFile("D:\\sparkdata2.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)_*)

println(sorted)

for((k,v)<-sorted)
{
if(v>4)
{
print(k+",")
print(v)
println()
}
}
```

Output:

```
scala> val textFile = sc.textFile("D:\\sparkdata2.txt")
textFile: org.apache.spark.rdd.RDD[String] = D:\sparkdata2.txt MapPartitionsRDD[21] at textFile at <console>:24

scala> val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[24] at reduceByKey at <console>:24

scala> import scala.collection.immutable.ListMap
import scala.collection.immutable.ListMap

scala> val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(Spark -> 6, data -> 4, computations -> 4, shells -> 4, "" -> 3, memory -> 3, on -> 3, is -> 2, can -> 2, with -> 2, Shells -> 2, you -> 2, that ->
 2, a -> 2, many -> 2, disk -> 2, in -> 2, distributed -> 2, and -> 2, the -> 2, however, -> 1, hoc -> 1, this -> 1, analysis. -> 1, distributing -> 1, Python -> 1, provides -> 1, interact -> 1, few -> 1, Becaus
e -> 1, load -> 1, takes -> 1, into -> 1, interactive -> 1, using -> 1, machines -> 1, Scala -> 1, manipulate -> 1, Unlike -> 1, other -> 1, single -> 1, worker -> 1, shells, -> 1, processing. -> 1, enable -> 1,
 run -> 1, allow -> 1, most -> 1, let -> 1, across -> 1, or -> 1, to -> 1, automatically -> 1, ad -> 1, which -> 1, of -> 1, care -> 1, comes -> 1, both -> 1, Spark...

scala> println(sorted)
ListMap(Spark -> 6, data -> 4, computations -> 4, shells -> 4,  -> 3, memory -> 3, on -> 3, is -> 2, can -> 2, with -> 2, Shells -> 2, you -> 2, that -> 2, a -> 2, many -> 2, disk -> 2, in -> 2, distributed -> 2
, and -> 2, the -> 2, however, -> 1, hoc -> 1, this -> 1, analysis. -> 1, distributing -> 1, Python -> 1, provides -> 1, interact -> 1, few -> 1, Because -> 1, load -> 1, takes -> 1, into -> 1, interactive -> 1,
 using -> 1, machines -> 1, Scala -> 1, manipulate -> 1, Unlike -> 1, other -> 1, single -> 1, worker -> 1, shells, -> 1, processing. -> 1, enable -> 1, run -> 1, allow -> 1, most -> 1, let -> 1, across -> 1, or
 -> 1, to -> 1, automatically -> 1, ad -> 1, which -> 1, of -> 1, care -> 1, comes -> 1, both -> 1, Spark?s -> 1, seconds -> 1, nodes, -> 1, machine, -> 1)

scala> for((k,v)<-sorted)
     | {
     | if(v>4)
     | {
     | print(k+",")
     | print(v)
     | println()
     | }
     | }
Spark,6
```