VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data Analytics

Submitted by

Ajith Kumar G (1BM19CS009)

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 **Ajith Kumar G (1BM19CS009),** 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
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**

1. DB operations using Cassandra – Employee:

Cassandra- Employee

```
cqlsh:system> CREATE KEYSPACE employee with replication = {
          ... 'class':'SimpleStrategy','replication_factor':1};
cqlsh:system> USE employee;

cqlsh:employee> CREATE TABLE employee_info (
           ... emp_id text,
           ... emp_name text,
           ... designation text,
           ... date_of_joining date,
           ... salary float,
           ... dept_name text,
           ... PRIMARY KEY(emp_id)
           ... );

cqlsh:employee> DESC TABLES employee_info

cqlsh:employee> BEGIN BATCH
           ... INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
           ... VALUES ('120','John','Software Engineering','2020-01-01',80000,'Development')
           ... INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
           ... VALUES ('121','Harry','Debugger','2020-04-11',60000,'Development')
           ... INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
           ... VALUES ('122','Clark','Tester','2020-02-21',75000,'Testing')
           ... APPLY BATCH;

cqlsh:employee> SELECT * FROM employee_info;

 emp_id | date_of_joining | dept_name   | designation          | emp_name | salary
--------+-----------------+-------------+----------------------+----------+--------
    120 |      2020-01-01 | Development | Software Engineering |     John |  80000
    121 |      2020-04-11 | Development |             Debugger |    Harry |  60000
    122 |      2020-02-21 |     Testing |               Tester |    Clark |  75000

(3 rows)

cqlsh:employee> UPDATE employee_info SET emp_name='Potter',dept_name='Testing'
           ... WHERE emp_id='121';

cqlsh:employee> ALTER TABLE employee_info ADD Projects set<text>;

cqlsh:employee> UPDATE employee_info SET Projects={'SQL','QT'} WHERE emp_id='120';

cqlsh:employee> UPDATE employee_info SET Projects={'UI/UX','PYPY3'} WHERE emp_id='121';

cqlsh:employee> UPDATE employee_info SET Projects={'Voice Module','DATACENTER'} WHERE emp_id='122';

cqlsh:employee> SELECT * FROM employee_info;

 emp_id | date_of_joining | dept_name   | designation          | emp_name | projects                        | salary
--------+-----------------+-------------+----------------------+----------+---------------------------------+--------
    120 |      2020-01-01 | Development | Software Engineering |     John |                  {'QT', 'SQL'} |  80000
    121 |      2020-04-11 |     Testing |             Debugger |   Potter |              {'PYPY3', 'UI/UX'} |  60000
    122 |      2020-02-21 |     Testing |               Tester |    Clark | {'DATACENTER', 'Voice Module'} |  75000
```

```
cqlsh:employee> UPDATE employee_info SET emp_name='Potter',dept_name='Testing'
           ... WHERE emp_id='121';

cqlsh:employee> ALTER TABLE employee_info ADD Projects set<text>;

cqlsh:employee> UPDATE employee_info SET Projects={'SQL','QT'} WHERE emp_id='120';

cqlsh:employee> UPDATE employee_info SET Projects={'UI/UX','PYPY3'} WHERE emp_id='121';

cqlsh:employee> UPDATE employee_info SET Projects={'Voice Module','DATACENTER'} WHERE emp_id='122';

cqlsh:employee> SELECT * FROM employee_info;

 emp_id | date_of_joining | dept_name   | designation          | emp_name | projects                      | salary
--------+-----------------+-------------+----------------------+----------+-------------------------------+--------
    120 |      2020-01-01 | Development | Software Engineering |     John |                 {'QT', 'SQL'} |  80000
    121 |      2020-04-11 |     Testing |             Debugger |   Potter |             {'PYPY3', 'UI/UX'} |  60000
    122 |      2020-02-21 |     Testing |               Tester |    Clark | {'DATACENTER', 'Voice Module'} |  75000

(3 rows)

cqlsh:employee> INSERT INTO employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES
('123','James','System Design Lead','2020-03-02',90000,'Development') USING TTL 15;

cqlsh:employee> SELECT TTL(emp_name) FROM employee_info WHERE emp_id='123' ;

 ttl(emp_name)
---------------
              9
(1 rows)

cqlsh:employee> SELECT TTL(emp_name) FROM employee_info WHERE emp_id='123' ;

 ttl(emp_name)
---------------
              6
(1 rows)

cqlsh:employee> SELECT * FROM employee_info ;

 emp_id | date_of_joining | dept_name   | designation          | emp_name | projects                      | salary
--------+-----------------+-------------+----------------------+----------+-------------------------------+--------
    123 |      2020-03-02 | Development |   System Design Lead |    James |                          null |  90000
    120 |      2020-01-01 | Development | Software Engineering |     John |                 {'QT', 'SQL'} |  80000
    121 |      2020-04-11 |     Testing |             Debugger |   Potter |             {'PYPY3', 'UI/UX'} |  60000
    122 |      2020-02-21 |     Testing |               Tester |    Clark | {'DATACENTER', 'Voice Module'} |  75000

(4 rows)

cqlsh:employee> SELECT TTL(emp_name) FROM employee_info WHERE emp_id='123' ;

 ttl(emp_name)
---------------

(0 rows)
cqlsh:employee> SELECT * FROM employee_info ;

 emp_id | date_of_joining | dept_name   | designation          | emp_name | projects                      | salary
--------+-----------------+-------------+----------------------+----------+-------------------------------+--------
    120 |      2020-01-01 | Development | Software Engineering |     John |                 {'QT', 'SQL'} |  80000
    121 |      2020-04-11 |     Testing |             Debugger |   Potter |             {'PYPY3', 'UI/UX'} |  60000
    122 |      2020-02-21 |     Testing |               Tester |    Clark | {'DATACENTER', 'Voice Module'} |  75000

(3 rows)
```

2. DB operations using Cassandra – Library:

Cassandra- Employee

```
cqlsh> CREATE KEYSPACE Library with replication={ 'class':'SimpleStrategy', 'replication_factor':1};
cqlsh> USE Library
cqlsh:library> CREATE TABLE library_info(
           ... stud_id int,
           ... stud_name text,
           ... book_id int,
           ... book_name text,
           ... counter_value counter,
           ... date_of_issue date,
           ... PRIMARY KEY((stud_id,book_id),stud_name,book_name,date_of_issue)
           ... );

cqlsh:library> DESC library_info;
CREATE TABLE library.library_info (
    stud_id int,
    book_id int,
    stud_name text,
    book_name text,
    date_of_issue date,
    counter_value counter,
    PRIMARY KEY ((stud_id, book_id), stud_name, book_name, date_of_issue)
) WITH CLUSTERING ORDER BY (stud_name ASC, book_name ASC, date_of_issue ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32',
'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1  WHERE book_id=100 and stud_id=112 and stud_name='Krishna'
and book_name='BDA'and date_of_issue='2020-02-02' ;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1  WHERE book_id=100 and stud_id=112 and stud_name='Krishna'
and book_name='BDA'and date_of_issue='2020-02-02' ;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1  WHERE book_id=201 and stud_id=132 and stud_name='Arthur'
and book_name='CNS'and date_of_issue='2020-02-05' ;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1  WHERE book_id=111 and stud_id=202 and stud_name='Alan' and
book_name='OOMD'and date_of_issue='2020-09-12' ;

cqlsh:library> SELECT * FROM library_info;
```

```
 stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
---------+---------+-----------+-----------+---------------+---------------
     132 |     201 |    Arthur |       CNS |    2020-02-05 |             1
     112 |     100 |   Krishna |       BDA |    2020-02-02 |             2
     202 |     111 |      Alan |      OOMD |    2020-09-12 |             1
(3 rows)


cqlsh:library> SELECT * from library_info WHERE stud_id=112 and book_id=100;

 stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
---------+---------+-----------+-----------+---------------+---------------
     112 |     100 |   Krishna |       BDA |    2020-02-02 |             2

cqlsh:library> COPY library_info(stud_id,book_id,stud_name,book_name,date_of_issue,counter_value) TO 'LIB.csv';
Using 1 child processes

Starting copy of library.library_info with columns [stud_id, book_id, stud_name, book_name, date_of_issue, counter_value].

Processed: 3 rows; Rate:       1 rows/s; Avg. rate:       1 rows/s
3 rows exported to 1 files in 2.418 seconds.

cqlsh:library> TRUNCATE library_info;
cqlsh:library> SELECT * FROM library_info;

 stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
---------+---------+-----------+-----------+---------------+---------------
(0 rows)

cqlsh:library> COPY library_info(stud_id,book_id,stud_name,book_name,date_of_issue,counter_value) FROM 'LIB.csv';
Using 1 child processes

Starting copy of library.library_info with columns [stud_id, book_id, stud_name, book_name, date_of_issue, counter_value].

Processed: 3 rows; Rate:       3 rows/s; Avg. rate:       5 rows/s
3 rows imported from 1 files in 0.593 seconds (0 skipped).
cqlsh:library> SELECT * FROM library_info;

 stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
---------+---------+-----------+-----------+---------------+---------------
     132 |     201 |    Arthur |       CNS |    2020-02-05 |             1
     112 |     100 |   Krishna |       BDA |    2020-02-02 |             2
     202 |     111 |      Alan |      OOMD |    2020-09-12 |             1

(3 rows)
```

3. MongoDB- CRUD Demonstration:

(i) Mongo-1

Mongo-1

```
> use mySTUD
switched to db mySTUD

> 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: 2, Name:"Wick", USN: "1B22IS301",Semester: 4,Dept_name: "ISE", CGPA:
8.3, Hobbies : ["Reading","Gardening"]})
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: 4, Name:"Arthur", USN: "1B22CS041",Semester: 6,Dept_name: "CSE", CGPA:
8.6, Hobbies : ["Novel Reading"]})
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id: 5, Name:"Tess", USN: "1B22ME011",Semester: 5,Dept_name: "ME", CGPA:
9.1, Hobbies : ["DIY"]})
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id: 6, Name:"Sylvia", USN: "1B22CS013",Semester: 5,Dept_name: "CSE", CGPA:
9.1, Hobbies : ["DIY"]})
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" : 2,
    "Name" : "Wick",
    "USN" : "1B22IS301",
    "Semester" : 4,
    "Dept_name" : "ISE",
    "CGPA" : 8.3,
    "Hobbies" : [
        "Reading",
        "Gardening"
    ]
}
```

```
{
    "_id" : 3,
    "Name" : "Horris",
    "USN" : "1B22EE021",
    "Semester" : 5,
    "Dept_name" : "EEE",
    "CGPA" : 9.3,
    "Hobbies" : [
        "eSports"
    ]
}
{
    "_id" : 4,
    "Name" : "Arthur",
    "USN" : "1B22CS041",
    "Semester" : 6,
    "Dept_name" : "CSE",
    "CGPA" : 8.6,
    "Hobbies" : [
        "Novel Reading"
    ]
}
{
    "_id" : 5,
    "Name" : "Tess",
    "USN" : "1B22ME011",
    "Semester" : 5,
    "Dept_name" : "ME",
    "CGPA" : 9.1,
    "Hobbies" : [
        "DIY"
    ]
}

> db.Student.aggregate({$match :{Dept_name:"CSE"}},{$group: {_id:"$Semester",AvgCGPA:
{$avg:"$CGPA"}}},{$match :{AvgCGPA:{$gt:7.5}}})
{ "_id" : 5, "AvgCGPA" : 8.899999999999999 }
{ "_id" : 6, "AvgCGPA" : 9.1 }
```

(ii)Mongo-2

```
Mongo-2

> use mySTUD
switched to db mySTUD

> db.createCollection("Bank")
{ "ok" : 1 }

> db.Bank.insert({name: "Arka", type:"savings", transactions: ["+1000", "-100", "+5000"],
Balance:1400})
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({name: "Derek", type:"savings", transactions: ["-100", "+300", "+500"],
Balance:5500})
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({name: "Shreastha", type:"savings", transactions: ["+200", "-300", "+60",
"-70"], Balance:8000})
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({name: "Harries", type:"savings", transactions: ["+600", "-7000"],
Balance:11000})
WriteResult({ "nInserted" : 1 })



> db.Bank.update({name:"Derek"},{$pull:{transactions:{$in:["+500"]}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Bank.find().pretty()
{
    "_id" : ObjectId("626649b716596fe24c1442bb"),
    "name" : "Arka",
    "type" : "savings",
    "transactions" : [
        "+1000",
        "-100"
    ],
    "Balance" : 1400
}
{
    "_id" : ObjectId("626649d116596fe24c1442bc"),
    "name" : "Derek",
    "type" : "savings",
    "transactions" : [
        "-100",
        "+300"
    ],
    "Balance" : 5500
}
```

```
{
    "_id" : ObjectId("626649f116596fe24c1442bd"),
    "name" : "Shreastha",
    "type" : "savings",
    "transactions" : [
        "+200",
        "-300",
        "+60",
        "-70"
    ],
    "Balance" : 8000
}
{
    "_id" : ObjectId("62664a1216596fe24c1442be"),
    "name" : "Harries",
    "type" : "savings",
    "transactions" : [
        "+600",
        "-7000"
    ],
    "Balance" : 11000
}

> db.Bank.update({name: "Shreastha"},{$pop:{transactions:-1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Bank.find().pretty()
{
    "_id" : ObjectId("626649b716596fe24c1442bb"),
    "name" : "Arka",
    "type" : "savings",
    "transactions" : [
        "+1000",
        "-100"
    ],
    "Balance" : 1400
}
{
    "_id" : ObjectId("626649d116596fe24c1442bc"),
    "name" : "Derek",
    "type" : "savings",
    "transactions" : [
        "-100",
        "+300"
    ],
    "Balance" : 5500
}
```

```
{
    "_id" : ObjectId("626649f116596fe24c1442bd"),
    "name" : "Shreastha",
    "type" : "savings",
    "transactions" : [
        "-300",
        "+60",
        "-70"
    ],
    "Balance" : 8000
}
{
    "_id" : ObjectId("62664a1216596fe24c1442be"),
    "name" : "Harries",
    "type" : "savings",
    "transactions" : [
        "+600",
        "-7000"
    ],
    "Balance" : 11000
}
```

4. Screenshot of Hadoop installed:

```
mintwind@MintWind:~/hadoop-2.7.3/sbin$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /home/mintwind/hadoop-2.7.3/share/hadoop/common/hadoop-common-2.7.3.jar
```

5. Execution of HDFS Commands for interaction with Hadoop Environment:

```
Hadoop

Hadoop Commands

To start with:
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/hadoop-hduser-namenode-bmsce-
Precision-T1700.out
hduser@localhost's password:
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-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-bmsce-Precision-T1700.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-bmsce-
Precision-T1700.out
hduser@localhost's password:
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-bmsce-
Precision-T1700.out

hduser@bmsce-Precision-T1700:~$ jps
7097 DataNode
7802 NodeManager
12540 Jps
7469 ResourceManager
6925 NameNode
7310 SecondaryNameNode

Commands:
1:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir /hadoop

2:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /
Found 1 item
drwxr-xr-x   - hduser supergroup          0 2022-06-06 11:37 /hadoop

3:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -put /home/hduser/Desktop/hadoop.txt /hadoop/hadoop.txt

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

4:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyFromLocal /home/hduser/Desktop/hadoop.txt
/hadoop/hadoop2.txt
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /hadoop/hadoop.txt
"Hello, I'm Hadoop"

5:
hduser@bmsce-Precision-T1700:~$ hdfs dfs -get /hadoop/hadoop1.txt /home/hduser/Desktop/hd.txt
```

```
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
```

6. Map Reduce program for weather data:

(a) Average temperature for each year:

mapper.py

```python
#!/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

```python
#!/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
```

(b) Mean max temperature for every month:

mapper.py

```python
#!/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

```python
#!/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
```

7. 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
```

8. 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
```

9. 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))
```

10. 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
```