**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

**Mazin Salim (1BM19CS201)**

***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 “LAB COURSE **BIG DATA ANALYTICS (****20CS6PEBDA)**” carried out by **Mazin Salim (1BM19CS201),** 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 BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| 1. | Cassandra Lab Program 1: - Employee Database | 4 |
| 2. | Cassandra Lab Program 2: - Library Database | 10 |
| 3. | MongoDB- CRUD Demonstration | 16 |
| 4. | Hadoop Installation | 23 |
| 5. | Hadoop Commands | 24 |
| 6. | Hadoop Program: Average Temperature | 30 |
| 7. | Hadoop Program: Word Count (TopN) | 37 |
| 8. | Hadoop Program: Join Operation | 42 |
| 9. | Scala Program | 51 |
| 10. | Scala Program: Word Count | 53 |

**Course Outcome**

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

**LAB 1: Cassandra Lab Program 1: - Employee Database**

**1) Perform the following DB operations using Cassandra.**

**I. Create a keyspace by name Employee**

**II. Create a column family by name**

**Employee-Info with attributes**

**Emp\_Id Primary Key, Emp\_Name,**

**Designation, Date\_of\_Joining, Salary, Dept\_Name**

**III. Insert the values into the table in batch**

**IV. Update Employee name and Department of Emp-Id 121**

**V. Sort the details of Employee records based on salary**

**VI. Alter the schema of the table Employee\_Info to add a column Projects which stores a set**

**of**

**Projects done by the corresponding Employee.**

**VII. Update the altered table to add project names.**

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

create keyspace employee\_199 with replication = {'class':'SimpleStrategy','replication\_factor':1};

cqlsh> describe keyspaces;

stud1 system\_auth employ employee drivers

harshita newstudents students1 system\_traces

employee\_199 library student lib1

system\_schema system system\_distributed bigcassandra

cqlsh> select \* from system\_schema.keyspaces;

keyspace\_name | durable\_writes | replication

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

student | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

system\_auth | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

system\_schema | True | {'class': 'org.apache.cassandra.locator.LocalStrategy'}

library | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

bigcassandra | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '3'}

lib1 | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

employee\_199 | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

stud1 | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

students1 | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

newstudents | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

system\_distributed | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '3'}

system | True | {'class': 'org.apache.cassandra.locator.LocalStrategy'}

drivers | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

system\_traces | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '2'}

harshita | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '3'}

employee | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

employ | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication\_factor': '1'}

(17 rows)

cqlsh> use employee\_199;

cqlsh:employee\_199> create table emp\_info ( emp\_id int, emp\_name text, designation text, DOJ timestamp, salary double, dept\_name text, PRIMARY KEY(emp\_id, salary));

cqlsh:employee\_199> describe tables;

emp\_info

cqlsh:employee\_199> describe table emp\_info;

CREATE TABLE employee\_199.emp\_info (

emp\_id int,

salary double,

dept\_name text,

designation text,

doj timestamp,

emp\_name text,

PRIMARY KEY (emp\_id, salary)

) ;

cqlsh:employee\_199> begin batch

... insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name) values(1,'Jack','manager','2021-02-12',5000,'webdev')

... insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name) values(2,'Mohan','clerk','2022-03-22',10000,'datacenter')

... insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name) values(3,'Sumesh','COE','2019-04-10',100000,'marketing')

... apply batch;

cqlsh:employee\_199> select \* from emp\_info;

emp\_id | salary | dept\_name | designation | doj | emp\_name

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

1 | 5000 | webdev | manager | 2021-02-11 18:30:00.000000+0000 | Jack

2 | 10000 | datacenter | clerk | 2022-03-21 18:30:00.000000+0000 | Mohan

3 | 1e+05 | marketing | COE | 2019-04-09 18:30:00.000000+0000 | Sumesh

(3 rows)

cqlsh:employee\_199> update emp\_info set emp\_name='Rohit', dept\_name='advertising' where emp\_id=3;

cqlsh:employee\_199> select \* from emp\_info;

emp\_id | salary | dept\_name | designation | doj | emp\_name

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

1 | 5000 | webdev | manager | 2021-02-11 18:30:00.000000+0000 | Jack

2 | 10000 | datacenter | clerk | 2022-03-21 18:30:00.000000+0000 | Mohan

3 | 1e+05 | advertising | COE | 2019-04-09 18:30:00.000000+0000 | Rohit

(3 rows)

cqlsh:employee\_199> select \* from emp\_info order by salary desc;

InvalidRequest: Error from server: code=2200 [Invalid query] message="ORDER BY is only supported when the partition key is restricted by an EQ or an IN."

cqlsh:employee\_199> alter table emp\_info add projects set <text>;

cqlsh:employee\_199> select \* from emp\_info;

emp\_id | salary | dept\_name | designation | doj | emp\_name | projects

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

1 | 5000 | webdev | manager | 2021-02-11 18:30:00.000000+0000 | Jack | null

2 | 10000 | datacenter | clerk | 2022-03-21 18:30:00.000000+0000 | Mohan | null

3 | 1e+05 | advertising | COE | 2019-04-09 18:30:00.000000+0000 | Rohit | null

(3 rows)

cqlsh:employee\_199> update emp\_info set projects={'data science'} where emp\_id=1;

cqlsh:employee\_199> update emp\_info set projects={'security','crypto'} where emp\_id=2;

cqlsh:employee\_199> update emp\_info set projects={'mobile app'} where emp\_id=3;

cqlsh:employee\_199> select \* from emp\_info;

emp\_id | salary | dept\_name | designation | doj | emp\_name | projects

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

1 | 5000 | webdev | manager | 2021-02-11 18:30:00.000000+0000 | Jack | {'data science'}

2 | 10000 | datacenter | clerk | 2022-03-21 18:30:00.000000+0000 | Mohan | {'crypto', 'security'}

3 | 1e+05 | advertising | COE | 2019-04-09 18:30:00.000000+0000 | Rohit | {'mobile app'}

(3 rows)

cqlsh:employee\_199> insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name) values(4,'rakesh','intern','2022-04-19',1000,'marketing',{'data science'}) using TTL 18;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Unmatched column names/values"

cqlsh:employee\_199> insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name,projects) values(4,'rakesh','intern','2022-04-19',1000,'marketing',{'data science'}) using TTL 18;

cqlsh:employee\_199> select ttl(dept\_name) from emp\_info where emp\_id=4;

ttl(dept\_name)

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

(0 rows)

cqlsh:employee\_199> insert into emp\_info(emp\_id,emp\_name,designation,DOJ,salary,dept\_name,projects) values(4,'rakesh','intern','2022-04-19',1000,'marketing',{'data science'}) using TTL 50;

cqlsh:employee\_199> select ttl(dept\_name) from emp\_info where emp\_id=4;

ttl(dept\_name)

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

46

(1 rows)

**LAB 2: Cassandra Lab Program 2: - Library Database**

**2) Perform the following DB operations using Cassandra.**

**I. Create a keyspace by name Library**

**II. 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**

**III. Insert the values into the table in batch**

**IV. Display the details of the table created and increase the value of the counter**

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

**VI. Export the created column to a csv file**

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

bmsce@bmsce-Precision-T1700:~$ cqlsh

Connected to Test Cluster at 127.0.0.1:9042.

[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]

Use HELP for help.

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

AlreadyExists: Keyspace 'library' already exists

cqlsh> create keyspace library\_199 with replication={'class':'SimpleStrategy', 'replication\_factor':1};

cqlsh> describe keyspaces;

stud1 system\_auth employ employee drivers

harshita newstudents students1 system\_traces library\_199

bigcassandra library student lib1

system\_schema system system\_distributed employee\_199

cqlsh:library> use library\_199;

cqlsh:library\_199> create table lib\_info(stud\_id int PRIMARY KEY, counter\_val counter, stud\_name text, book\_name text, book\_id int, issue\_date date);

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot mix counter and non counter columns in the same table"

cqlsh:library\_199> create table lib\_info(stud\_id int , counter\_val counter, stud\_name text, book\_name text, book\_id int, issue\_date date, PRIMARY KEY (stud\_id,stud\_name,book\_name,book\_id,issue\_date);

SyntaxException: line 1:180 mismatched input ';' expecting ')' (...stud\_id,stud\_name,book\_name,book\_id,issue\_date)[;])

cqlsh:library\_199> create table lib\_info(stud\_id int , counter\_val counter, stud\_name text, book\_name text, book\_id int, issue\_date date, PRIMARY KEY (stud\_id,stud\_name,book\_name,book\_id,issue\_date));

cqlsh:library\_199> begin batch

... insert into lib\_info(stud\_id,stud\_name,book\_name,book\_id,issue\_date) values (121,'sumit','java',1140,'2022-05-07')

... apply batch;

InvalidRequest: Error from server: code=2200 [Invalid query] message="INSERT statements are not allowed on counter tables, use UPDATE instead"

cqlsh:library\_199> update lib\_info set counter\_val=counter\_val+1 where stud\_id=1 and stud\_name='sumit' and book\_name='oomd' and book\_id=121 and issue\_date='2022-05-06';

cqlsh:library\_199> select \* from lib\_info;

stud\_id | stud\_name | book\_name | book\_id | issue\_date | counter\_val

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

1 | sumit | oomd | 121 | 2022-05-06 | 1

(1 rows)

cqlsh:library\_199> update lib\_info set counter\_val=counter\_val+1 where stud\_id=2 and stud\_name='sukesh' and book\_name='bda' and book\_id=122 and issue\_date='2022-04-06';

cqlsh:library\_199> update lib\_info set counter\_val=counter\_val+1 where stud\_id=3 and stud\_name='reddy' and book\_name='java' and book\_id=123 and issue\_date='2022-04-10';

cqlsh:library\_199> update lib\_info set counter\_val=counter\_val+1 where stud\_id=4 and stud\_name='nikhil' and book\_name='ml' and book\_id=124 and issue\_date='2022-03-10';

cqlsh:library\_199> select \* from lib\_info;

stud\_id | stud\_name | book\_name | book\_id | issue\_date | counter\_val

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

1 | sumit | oomd | 121 | 2022-05-06 | 1

2 | sukesh | bda | 122 | 2022-04-06 | 1

4 | nikhil | ml | 124 | 2022-03-10 | 1

3 | reddy | java | 123 | 2022-04-10 | 1

(4 rows)

cqlsh:library\_199> update lib\_info set counter\_val=counter\_val+1 where stud\_id=2 and stud\_name='sukesh' and book\_name='bda' and book\_id=122 and issue\_date='2022-04-06';

cqlsh:library\_199> select \* from lib\_info;

stud\_id | stud\_name | book\_name | book\_id | issue\_date | counter\_val

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

1 | sumit | oomd | 121 | 2022-05-06 | 1

2 | sukesh | bda | 122 | 2022-04-06 | 2

4 | nikhil | ml | 124 | 2022-03-10 | 1

3 | reddy | java | 123 | 2022-04-10 | 1

(4 rows)

cqlsh:library\_199> select \* from lib\_info where counter\_val=2;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

cqlsh:library\_214> select \* from lib\_info where counter\_val=2 allow filtering;

stud\_id | stud\_name | book\_name | book\_id | issue\_date | counter\_val

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

2 | sukesh | bda | 122 | 2022-04-06 | 2

(1 rows)

cqlsh:library\_199> copy lib\_info(stud\_id,counter\_val,stud\_name,book\_name,book\_id,issue\_date) to '/home/bmsce/desktop/lib\_data.csv';

Using 11 child processes

Can't open '/home/bmsce/desktop/lib\_data.csv' for writing: [Errno 2] No such file or directory: '/home/bmsce/desktop/lib\_data.csv'

cqlsh:library\_199> copy lib\_info(stud\_id,counter\_val,stud\_name,book\_name,book\_id,issue\_date) to '/home/bmsce/Desktop/lib\_data';

Using 11 child processes

Starting copy of library\_199.lib\_info with columns [stud\_id, counter\_val, stud\_name, book\_name, book\_id, issue\_date].

Processed: 4 rows; Rate: 24 rows/s; Avg. rate: 24 rows/s

4 rows exported to 1 files in 0.177 seconds.

cqlsh:library\_199> create table lib\_info1(stud\_id int , counter\_val counter, stud\_name text, book\_name text, book\_id int, issue\_date date, PRIMARY KEY (stud\_id,stud\_name,book\_name,book\_id,issue\_date));

cqlsh:library\_199> copy lib\_info(stud\_id,counter\_val,stud\_name,book\_name,book\_id,issue\_date) from 'lib\_data.csv';

Using 11 child processes

Starting copy of library\_199.lib\_info with columns [stud\_id, counter\_val, stud\_name, book\_name, book\_id, issue\_date].

Failed to import 0 rows: IOError - Can't open 'lib\_data.csv' for reading: no matching file found, given up after 1 attempts

Processed: 0 rows; Rate: 0 rows/s; Avg. rate: 0 rows/s

0 rows imported from 0 files in 0.149 seconds (0 skipped).

cqlsh:library\_199> copy lib\_info1(stud\_id,counter\_val,stud\_name,book\_name,book\_id,issue\_date) from '/home/bmsce/Desktop/lib\_data.csv';

Using 11 child processes

Starting copy of library\_199.lib\_info1 with columns [stud\_id, counter\_val, stud\_name, book\_name, book\_id, issue\_date].

Failed to import 0 rows: IOError - Can't open '/home/bmsce/Desktop/lib\_data.csv' for reading: no matching file found, given up after 1 attempts

Processed: 0 rows; Rate: 0 rows/s; Avg. rate: 0 rows/s

0 rows imported from 0 files in 0.156 seconds (0 skipped).

cqlsh:library\_199> copy lib\_info1(stud\_id,counter\_val,stud\_name,book\_name,book\_id,issue\_date) from '/home/bmsce/Desktop/lib\_data';

Using 11 child processes

Starting copy of library\_199.lib\_info1 with columns [stud\_id, counter\_val, stud\_name, book\_name, book\_id, issue\_date].

Processed: 4 rows; Rate: 7 rows/s; Avg. rate: 11 rows/s

4 rows imported from 1 files in 0.375 seconds (0 skipped).

cqlsh:library\_199> select \* from lib\_info1;

stud\_id | stud\_name | book\_name | book\_id | issue\_date | counter\_val

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

1 | sumit | oomd | 121 | 2022-05-06 | 1

2 | sukesh | bda | 122 | 2022-04-06 | 2

4 | nikhil | ml | 124 | 2022-03-10 | 1

3 | reddy | java | 123 | 2022-04-10 | 1

(4 rows)

**LAB 3: MongoDB- CRUD Demonstration**

bmsce@bmsce-Precision-T1700:~$ mongo

MongoDB shell version v3.6.8

connecting to: mongodb://127.0.0.1:27017

Implicit session: session { "id" : UUID("d66acdb3-8482-417d-8b75-d65dae4b53ee") }

MongoDB server version: 3.6.8

Server has startup warnings:

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

2022-04-11T18:49:15.627+0530 I STORAGE [initandlisten] \*\* WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine

2022-04-11T18:49:15.627+0530 I STORAGE [initandlisten] \*\* See http://dochub.mongodb.org/core/prodnotes-filesystem

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

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten] \*\* WARNING: Access control is not enabled for the database.

2022-04-11T18:49:18.771+0530 I CONTROL [initandlisten] \*\* Read and write access to data and configuration is unrestricted.

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

> use Student

switched to db Student

> db.createCollection("student");

{ "ok" : 1 }

> db.Student.insert({\_id:1,StudName:"Megha",Grade:"vii",Hobbies:"InternetSurfing"});

WriteResult({ "nInserted" : 1 })

> db.Student.update({\_id:3,StudName:"Ayan",Grade:"vii"},{$set:{Hobbies:"skating"}},{upsert:true});

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 3 })

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

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

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

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

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

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

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

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

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

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

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

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

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

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

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

> db.Student.count();

2

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

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

WriteResult({ "nInserted" : 1 })

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

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

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

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

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

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

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

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

{

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

"StudName" : "Vamsi",

"Greade" : "vi"

}

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

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

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

> db.Students.count()

0

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

0

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

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

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

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

0

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

0

> db.Student.count()

3

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

2

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

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{

"\_id" : 3,

"Grade" : "vii",

"StudName" : "Ayan",

"Hobbies" : "skating",

"Location" : null

}

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

{

"\_id" : 3,

"Grade" : "vii",

"StudName" : "Ayan",

"Hobbies" : "skating",

"Location" : null

}

{

"\_id" : 1,

"StudName" : "Megha",

"Grade" : "vii",

"Hobbies" : "InternetSurfing"

}

{

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

"StudName" : "Vamsi",

"Greade" : "vi"

}

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

{

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

"StudName" : "Vamsi",

"Greade" : "vi"

}

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

2022-04-11T15:05:51.894+0530 E QUERY [thread1] SyntaxError: missing ] after element list @(shell):1:57

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

WriteResult({ "nInserted" : 1 })

> db.Customers.aggregate({$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}});

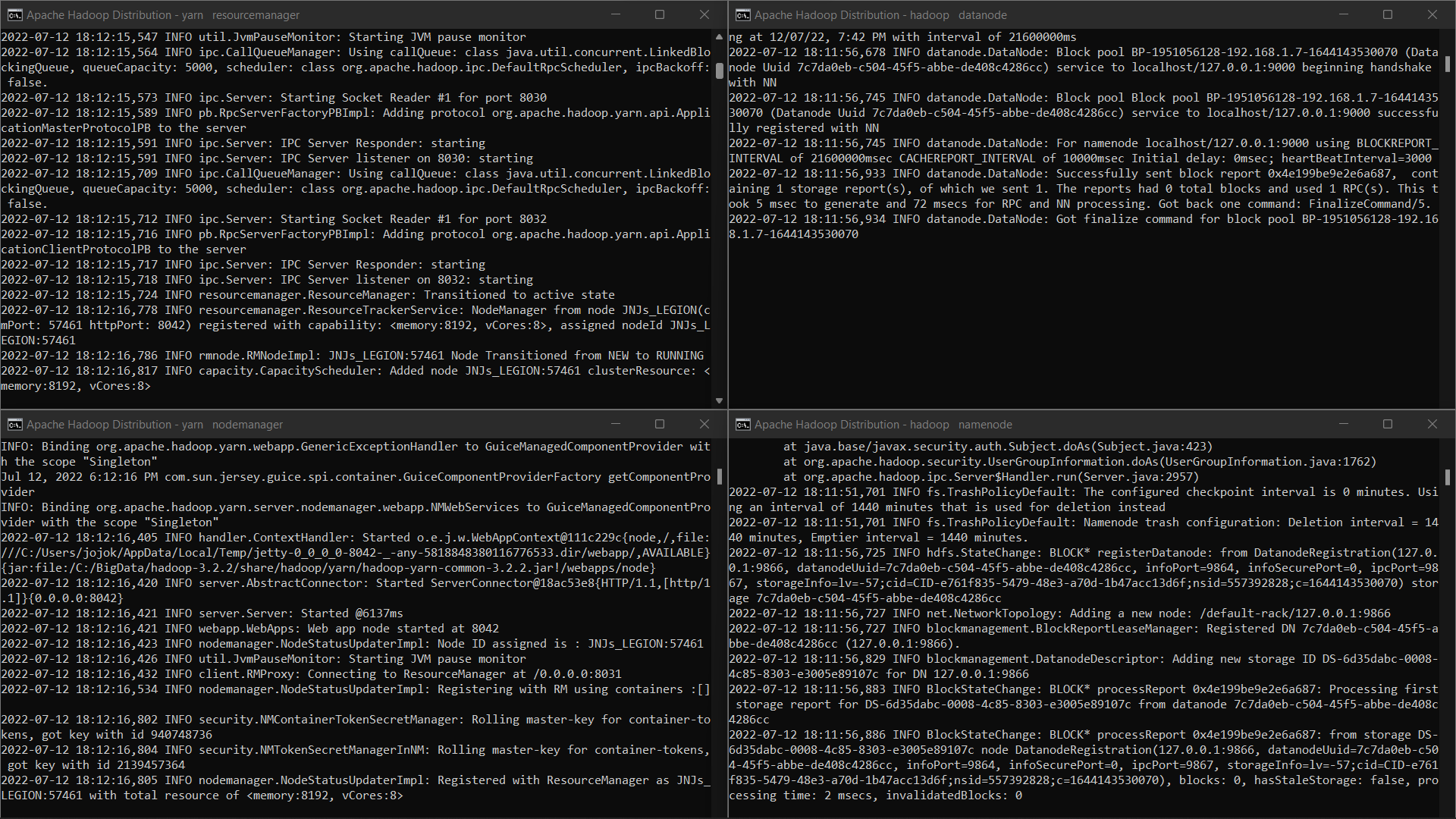
{ "\_id" : null, "TotAccBal" : 0 }

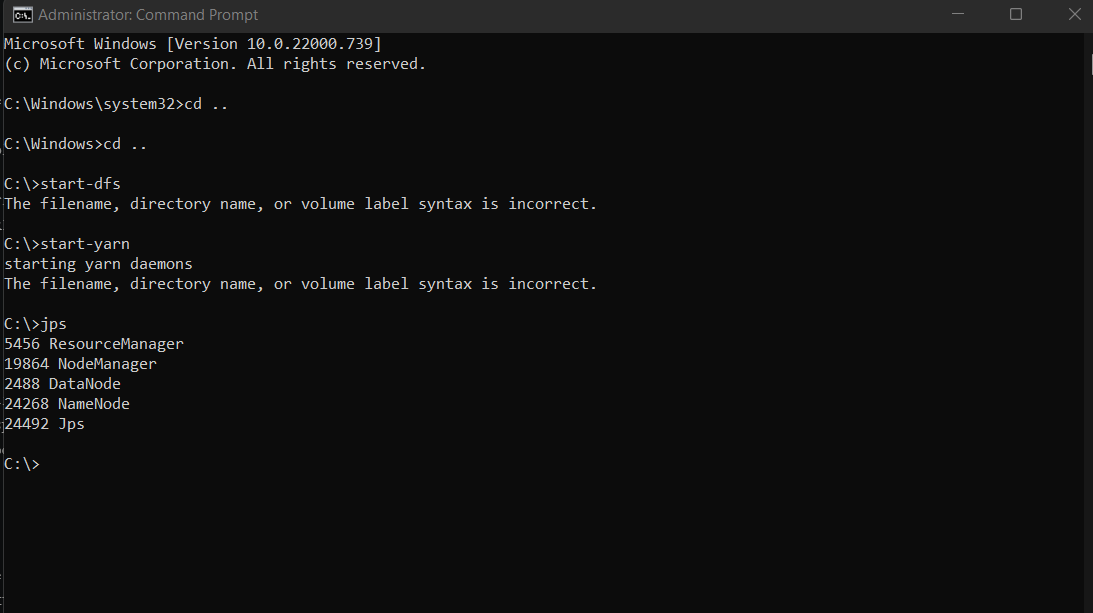
db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}});

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

db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}},{$match:{TotAccBal:{$gt:1200}}});

**LAB 4: Hadoop Installation**

****

****

**LAB 5: Hadoop Commands**

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

5072 SecondaryNameNode

4674 NameNode

4856 DataNode

5563 NodeManager

6507 Jps

5231 ResourceManager

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

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

Found 7 items

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:52 /SharmaJi

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:34 /abc

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:44 /bhavana

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:22 /lochan

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:45 /u1

-rw-r--r-- 1 hduser supergroup 19 2022-05-31 11:01 /user

drwxr-xr-x - hduser supergroup 0 2022-06-01 10:08 /vallisha

hduser@bmsce-Precision-T1700:~$ cat newfile.txt

SharmaJi

KhanwaJi

PaiJI

Kasturba

pandeyji

patilwa

Nairwa

hduser@bmsce-Precision-T1700:~$ hdfs dfs -put /home/hduser/newfile.txt /abc/joel.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /abc/joel.txt

SharmaJi

KhanwaJi

PaiJI

Kasturba

pandeyji

patilwa

Nairwa

hduser@bmsce-Precision-T1700:~$ cat > sample.txt

Hello

This is a new text file

^C

hduser@bmsce-Precision-T1700:~$ cat sample.txt

Hello

This is a new text file

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyFromLocal /home/hduser/sample.txt /abc/joel2.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /abc/joel2.txt

Hello

This is a new text file

hduser@bmsce-Precision-T1700:~$ hdfs dfs -get /abc/joel2.txt /home/hduser/joel2\_copy.txt

hduser@bmsce-Precision-T1700:~$ ls

derby.log 'Packet Tracer 7.2.1 for Linux 64 bit.tar.gz'

Desktop Pictures

Documents pig\_1564816082257.log

Downloads pig\_1599215374374.log

examples.desktop pt

first.text PT72Installer

hadoop-2.6.0.tar.gz Public

hive R

joel2\_copy.txt TCPclient.py

sample.txt TCPserver.py

lol Templates

metastore\_db toinstalledlist

Music UDPclient.py

newfile.txt UDPserver.py

hduser@bmsce-Precision-T1700:~$ hdfs dfs -getmerge /abc/joel.txt /abc/joel2.txt /home/hduser/joel\_merge.txt

hduser@bmsce-Precision-T1700:~$ ls

derby.log 'Packet Tracer 7.2.1 for Linux 64 bit.tar.gz'

Desktop Pictures

Documents pig\_1564816082257.log

Downloads pig\_1599215374374.log

examples.desktop pt

first.text PT72Installer

hadoop-2.6.0.tar.gz Public

hive R

joel2\_copy.txt TCPclient.py

joel\_merge.txt TCPserver.py

sample.txt Templates

lol toinstalledlist

metastore\_db UDPclient.py

Music UDPserver.py

newfile.txt Videos

hduser@bmsce-Precision-T1700:~$ cat joel\_merge.txt

SharmaJi

KhanwaJi

PaiJI

Kasturba

pandeyji

patilwa

Nairwa

Hello

This is a new text file

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

# file: /abc

# owner: hduser

# group: supergroup

user::rwx

group::r-x

other::r-x

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /abc/joel.txt /home/hduser/Desktop

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

Found 7 items

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:52 /SharmaJi

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:40 /abc

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:44 /bhavana

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:22 /lochan

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:45 /u1

-rw-r--r-- 1 hduser supergroup 19 2022-05-31 11:01 /user

drwxr-xr-x - hduser supergroup 0 2022-06-01 10:08 /vallisha

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

hduser@bmsce-Precision-T1700:~$ hadoop fs -mv /abc /joel

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /joel

Found 1 items

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:40 /joel/abc

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

Found 7 items

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:52 /SharmaJi

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:44 /bhavana

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:59 /joel

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:22 /lochan

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:45 /u1

-rw-r--r-- 1 hduser supergroup 19 2022-05-31 11:01 /user

drwxr-xr-x - hduser supergroup 0 2022-06-01 10:08 /vallisha

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /vallisha

Found 1 items

-rw-r--r-- 1 hduser supergroup 13 2022-06-01 09:52 /vallisha/sample1.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /joel/abc

Found 2 items

-rw-r--r-- 1 hduser supergroup 57 2022-06-04 09:37 /joel/abc/joel.txt

-rw-r--r-- 1 hduser supergroup 30 2022-06-04 09:40 /joel/abc/joel2.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /joel/abc/joel.txt

SharmaJi

KhanwaJi

PaiJI

Kasturba

pandeyji

patilwa

Nairwa

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cp /vallisha/sample1.txt /joel

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /joel

Found 2 items

drwxr-xr-x - hduser supergroup 0 2022-06-04 09:40 /joel/abc

-rw-r--r-- 1 hduser supergroup 13 2022-06-04 10:07 /joel/sample1.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /joel/sample1.txt

sample1 text

**LAB 6: Hadoop Program: Average Temperature**

**6. Create a Map Reduce program to**

**a) find average temperature for each year from the NCDC data set.**

AverageDriver

package temp;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver {

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

if (args.length != 2) {

System.err.println("Please Enter the input and output parameters");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(AverageDriver.class);

job.setJobName("Max temperature");

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setMapperClass(AverageMapper.class);

job.setReducerClass(AverageReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

AverageMapper

package temp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

public static final int MISSING = 9999;

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

IntWritable>.Context context) throws IOException, InterruptedException {

int temperature;

String line = value.toString();

String year = line.substring(15, 19);

if (line.charAt(87) == '+') {

temperature = Integer.parseInt(line.substring(88, 92));

} else {

temperature = Integer.parseInt(line.substring(87, 92));

}

String quality = line.substring(92, 93);

if (temperature != 9999 && quality.matches("[01459]"))

context.write(new Text(year), new IntWritable(temperature));

}

}

AverageReducer

package temp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class AverageReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,

IntWritable>.Context context) throws IOException, InterruptedException {

int max\_temp = 0;

int count = 0;

for (IntWritable value : values) {

max\_temp += value.get();

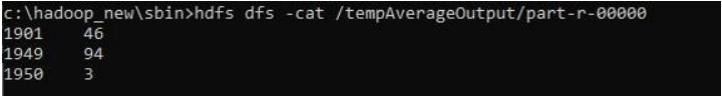
count++;

}

context.write(key, new IntWritable(max\_temp / count));

}

}

****

**b) Create a Map Reduce program to find the mean max temperature for every month**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;

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;

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;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,

IntWritable>.Context context) throws IOException, InterruptedException {

int max\_temp = 0;

int total\_temp = 0;

int count = 0;

int days = 0;

for (IntWritable value : values) {

int temp = value.get();

if (temp > max\_temp)

max\_temp = temp;

count++;

if (count == 3) {

total\_temp += max\_temp;

max\_temp = 0;

count = 0;

days++;

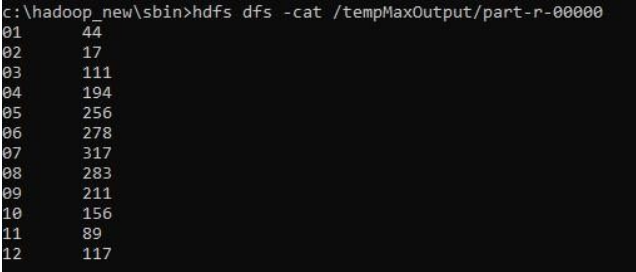
}

}

context.write(key, new IntWritable(total\_temp / days));

}

}

****

**LAB 7: Hadoop Program: Word Count (TopN)**

**7) 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.**

//Driver Code

package wordCount;

import java.io.IOException;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

public int run(String args[]) throws IOException

{

if (args.length < 2)

{

System.out.println("Please give valid inputs");

return -1;

}

JobConf conf = new JobConf(WCDriver.class);

FileInputFormat.setInputPaths(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

conf.setMapperClass(WCMapper.class);

conf.setReducerClass(WCReducer.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(IntWritable.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

return 0;

}

// Main Method

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

{

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

System.out.println(exitCode);

}

}

//Mapper Code

package wordCount;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text, IntWritable> {

// Map function

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

rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces

for (String word : line.split(" "))

{

if (word.length() > 0)

{

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

}

}

}

}

//Reducer Code

package wordCount;

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text,IntWritable, Text, IntWritable> {

// Reduce function

public void reduce(Text key, Iterator<IntWritable> value, OutputCollector<Text, IntWritable>

output,Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words

while (value.hasNext())

{

IntWritable i = value.next();

count += i.get();

}

output.collect(key, new IntWritable(count));

}

}

//Hadoop Commands

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

hduser@bmsce-Precision-T1700:~$ hadoop fs -copyFromLocal /home/hduser/Desktop/sample.txt

/joel/test.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /joel/test.txt

hi how are you

how is your job

how is your family

how is your brother

how is your sister

hduser@bmsce-Precision-T1700:~$ hadoop jar /home/hduser/Documents/wordCount.jar

wordCount.WCDriver /joel/test.txt /joel/output

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /joel/output/part-00000

are 1

brother 1

family 1

hi 1

how 5

is 4

job 1

sister 1

you 1

your 4

**LAB 8: Hadoop Program: Join Operation**

**8) 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;

}

}

@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 Nameinput'");

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

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

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

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

Posts.class);

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

User.class);

FileOutputFormat.setOutputPath(conf, outputPath);

conf.setPartitionerClass(KeyPartitioner.class);

conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

conf.setMapOutputKeyClass(TextPair.class);

conf.setReducerClass(JoinReducer.class);

conf.setOutputKeyClass(Text.class);

JobClient.runJob(conf);

return 0;

}

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

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

System.exit(exitCode);

}

}

// JoinReducer.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.Text;

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

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

Text,

Text> {

@Override

public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text>output, Reporter reporter)

throws IOException

{

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

while (values.hasNext()) {

Text node = values.next();

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

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

}

}

}

// User.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

import org.apache.hadoop.io.IntWritable;

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

TextPair,

Text> {

@Override

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

Reporter reporter)

throws IOException

{

String valueString = value.toString();

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

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

Text(SingleNodeData[1]));

}

}

//Posts.java

import java.io.IOException;

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

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

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

TextPair,

Text> {

@Override

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

Reporter reporter)

throws IOException

{

String valueString = value.toString();

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

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

Text(SingleNodeData[9]));

}

}

// TextPair.java

import java.io.\*;

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

public class TextPair implements WritableComparable<TextPair> {

private Text first;

private Text second;

public TextPair() {

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

}

public TextPair(String first, String second) {

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

}

public TextPair(Text first, Text second) {

set(first, second);

}

public void set(Text first, Text second) {

this.first = first;

this.second = second;

}

public Text getFirst() {

return first;

}

public Text getSecond() {

return second;

}

@Override

public void write(DataOutput out) throws IOException {

first.write(out);

second.write(out);

}

@Override

public void readFields(DataInput in) throws IOException {

first.readFields(in);

second.readFields(in);

}

@Override

public int hashCode() {

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

}

@Override

public boolean equals(Object o) {

if (o instanceof TextPair) {

TextPair tp = (TextPair) o;

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

}

return false;

}

@Override

public String toString() {

return first + "\t" + second;

}

@Override

public int compareTo(TextPair tp) {

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

if (cmp != 0) {

return cmp;

}

return second.compareTo(tp.second);

}

// ^^ TextPair

// vv TextPairComparator

public static class Comparator extends WritableComparator {

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

public Comparator() {

super(TextPair.class);

}

@Override

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

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

try {

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

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

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

if (cmp != 0) {

return cmp;

}

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

b2, s2 + firstL2, l2 - firstL2);

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

}

static {

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

}

public static class FirstComparator extends WritableComparator {

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

public FirstComparator() {

super(TextPair.class);

}

@Override

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

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

try {

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

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

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

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

@Override

public int compare(WritableComparable a, WritableComparable b) {

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

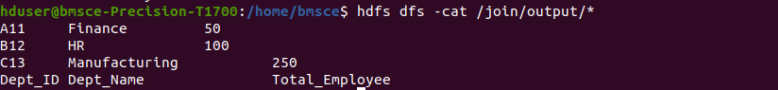
return ((TextPair) a).first.compareTo(((TextPair) b).first);

}

return super.compare(a, b);

}

} }

****

**LAB 9: Scala Program**

**9) Program to print word count on scala shell and print “Hello world” on scala IDE.**

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

data.collect;

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

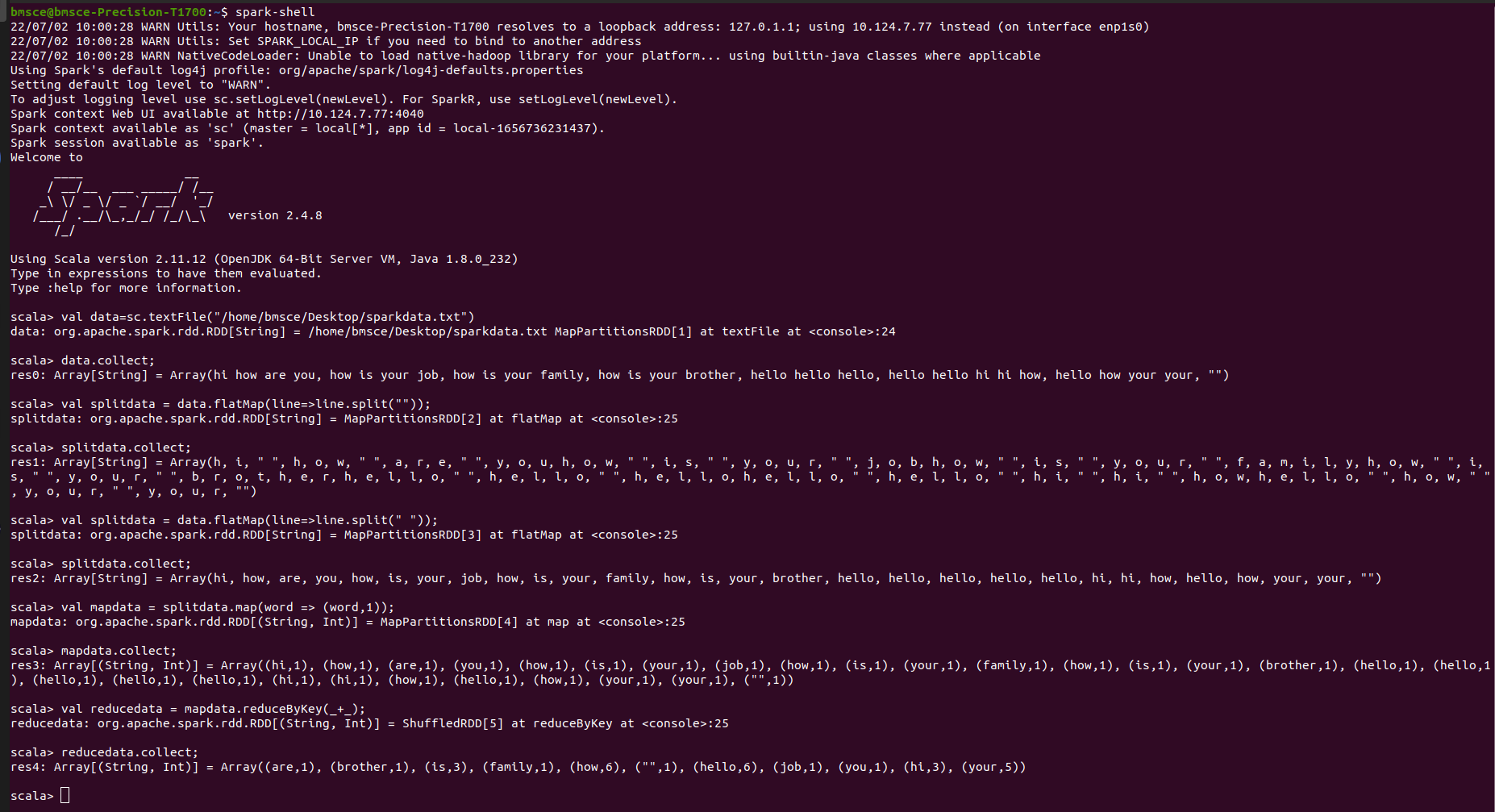
splitdata.collect;

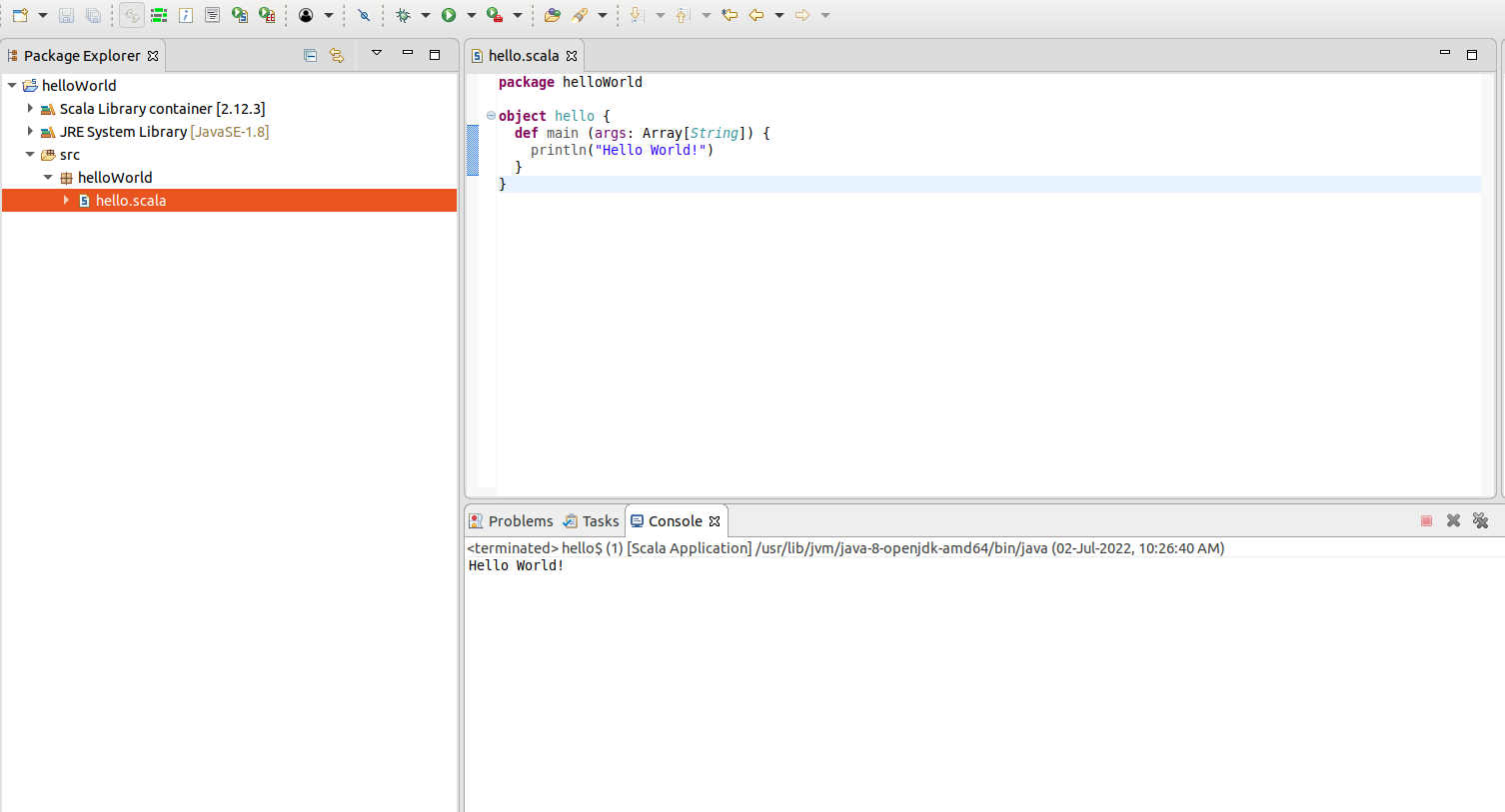
val mapdata = splitdata.map(word => (word,1));

mapdata.collect;

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

reducedata.collect;

****

****

**LAB 10: Scala Program: Word Count**

**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.**

val textFile = sc.textFile("/home/bmsce/Desktop/sparkdata.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()

}}

