**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

**NAVEENA K N(1BM20CS411)**

***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 **NAVEENA K N(1BM20CS411),** 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 **Course Title - (Course code)**work prescribed for the said degree.

Nameof the Lab-Incharge               **Dr. Jyothi S Nayak**

Designation Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Lab:1**

**Cassandra Program - 1**

**1. Create a key space by name Employee**

cqlsh> CREATE KEYSPACE Empyolees WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication\_factor' : 1 };

cqlsh> DESCRIBE KEYSPACES;

system\_schema crud project system\_distributed system\_traces

system\_auth system student empyolees

cqlsh> USE Employees;

**2. Create a column family by name Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name**

cqlsh:employees> CREATE TABLE Employee\_Info (

... Emp\_Id int PRIMARY KEY,

... Emp\_Name text,

... Designation text,

... Date\_Of\_Joining timestamp,

... Salary int,

... Dept\_Name text

... );

cqlsh:employees> DESCRIBE TABLES;

employee\_info

cqlsh:employees> DESCRIBE TABLE Employee\_Info;

CREATE TABLE employees.employee\_info (

emp\_id int PRIMARY KEY,

date\_of\_joining timestamp,

dept\_name text,

designation text,

emp\_name text,

salary int

) WITH 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';

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

cqlsh:employees> BEGIN BATCH

... INSERT INTO Employee\_Info (Emp\_Id,Emp\_Name,Designation,Date\_of\_Joining,Salary,Dept\_Name) VALUES (1,'Bruce Wayne','CEO','2022-04-22',100000,'Management')

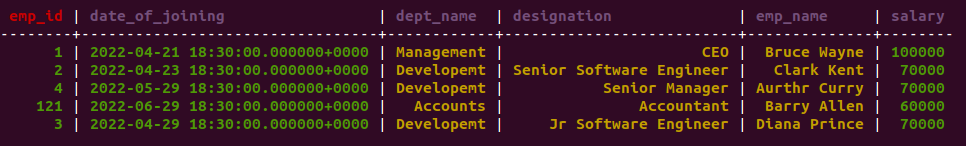
... INSERT INTO Employee\_Info (Emp\_Id,Emp\_Name,Designation,Date\_of\_Joining,Salary,Dept\_Name) VALUES (2,'Clark Kent','Senior Software Engineer','2022-04-24',70000,'Developemt')

... INSERT INTO Employee\_Info (Emp\_Id,Emp\_Name,Designation,Date\_of\_Joining,Salary,Dept\_Name) VALUES (3,'Diana Prince','Jr Software Engineer','2022-04-30',70000,'Developemt')

... INSERT INTO Employee\_Info (Emp\_Id,Emp\_Name,Designation,Date\_of\_Joining,Salary,Dept\_Name) VALUES (4,'Aurthr Curry','Senior Manager','2022-05-30',70000,'Developemt')

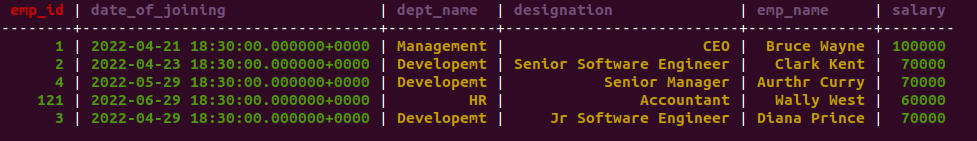
... APPLY BATCH;

cqlsh:employees> SELECT \* FROM Employee\_Info;



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

cqlsh:employees> UPDATE Employee\_Info SET Emp\_Name = 'Wally West', dept\_name = 'HR' WHERE Emp\_id = 121;



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

cqlsh:employees> CREATE TABLE Employee\_Info (

... Emp\_Id int,

... Emp\_Name text,

... Designation text,

... Date\_Of\_Joining timestamp,

... Salary int,

... Dept\_Name text,

… PRIMARY KEY (Emp\_Id , Salary)

... ) WITH CLUSTERING ORDER BY (Salary desc);

cqlsh:employee> select \* from Employee\_Info;

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

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

121 | 2022-06-29 18:30:00.000000+0000 | HR | Accountant | Wally West |60000

3 | 2022-04-29 18:30:00.000000+0000 | Development| Jr Software Manager | Diana Prince |70000

2 | 2022-04-23 18:30:00.000000+0000 | Management |Senior Software Manager|Clark Kent |70000

4 | 2022-05-29 18:30:00.000000+0000 | Development| Senior Manager | Aurthur Curry |70000

1 | 2022-04-21 18:30:00.000000+0000 | Management | CEO |Bruce Wayne |100000

**6. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.**

cqlsh:employee> ALTER TABLE Employee\_Info ADD Projects text;

cqlsh:employee> select \* from Employee\_Info;

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

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

1 | 2022-04-21 18:30:00.000000+0000 | Management | CEO |Bruce Wayne | null |100000

2 | 2022-04-23 18:30:00.000000+0000 | Management |Senior Software Manager|Clark Kent | null |70000

4 | 2022-05-29 18:30:00.000000+0000 | Development| Senior Manager | Aurthur Curry | null |70000

121 | 2022-06-29 18:30:00.000000+0000 | HR | Accountant | Wally West | null |60000

3 | 2022-04-29 18:30:00.000000+0000 | Development| Jr Software Manager | Diana Prince | null |70000

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

cqlsh:employee> UPDATE Employee\_Info SET Projects='Research' WHERE Emp\_id=1 and salary=100000.0;

cqlsh:employee> select \* from Employee\_Info;

cqlsh:employee> select \* from Employee\_Info;

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

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

1 | 2022-04-21 18:30:00.000000+0000 | Management | CEO |Bruce Wayne | Research |100000

2 | 2022-04-23 18:30:00.000000+0000 | Management |Senior Software Manager|Clark Kent | null |70000

4 | 2022-05-29 18:30:00.000000+0000 | Development| Senior Manager | Aurthur Curry | null |70000

121 | 2022-06-29 18:30:00.000000+0000 | HR | Accountant | Wally West | null |60000

3 | 2022-04-29 18:30:00.000000+0000 | Development| Jr Software Manager | Diana Prince | null |70000

cqlsh:employee> UPDATE Employee\_Info SET Projects='Data Migration' WHERE Emp\_id=2 and salary=70000.0;

cqlsh:employee> UPDATE Employee\_Info SET Projects='Data analysis' WHERE Emp\_id=3 and salary=70000.0;

cqlsh:employee> UPDATE Employee\_Info SET Projects='Reporting' WHERE Emp\_id=121 and salary=60000.0;

cqlsh:employee> UPDATE Employee\_Info SET Projects='Research' WHERE Emp\_id=4 and salary=70000.0;

cqlsh:employee> select \* from Employee\_Info;

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

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

1 | 2022-04-21 18:30:00.000000+0000 | Management | CEO |Bruce Wayne | Research |100000

2 | 2022-04-23 18:30:00.000000+0000 | Management |Senior Software Manager|Clark Kent | Data Migration |70000

4 | 2022-05-29 18:30:00.000000+0000 | Development| Senior Manager | Aurthur Curry |Data analysis |70000

121 | 2022-06-29 18:30:00.000000+0000 | HR | Accountant | Wally West |Reporting |60000

3 | 2022-04-29 18:30:00.000000+0000 | Development| Jr Software Manager | Diana Prince |Research |70000

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

cqlsh:employee> INSERT INTO Employee\_Info(Emp\_id, Emp\_Name, Designation, Date\_Of\_Joining, salary, Dept\_name) VALUES (5,'John Jones','CTO','2022-04-01',80000.0,'Space Station') using ttl 15;

cqlsh:employee> select ttl(Emp\_Name) from Employee\_Info Where Emp\_id=5;

ttl(emp\_name)

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

6

**Lab 2:**

**Cassandra Program - 2**

1. **Create a key space by name Library**

bmsce@bmsce-Precision-T1700:~$ Cassandra/apache-cassandra-3.11.0/bin

bash: Cassandra/apache-cassandra-3.11.0/bin: Is a directory

bmsce@bmsce-Precision-T1700:~$ Cassandra/apache-cassandra-3.11.0/bin/

bash: Cassandra/apache-cassandra-3.11.0/bin/: Is a directory

bmsce@bmsce-Precision-T1700:~$ cd Cassandra/apache-cassandra-3.11.0/bin/

bmsce@bmsce-Precision-T1700:~/Cassandra/apache-cassandra-3.11.0/bin$ ./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

... };

cqlsh> describe keyspaces

system\_schema system student system\_traces

system\_auth library system\_distributed

**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**

cqlsh:library> create table library\_info(stud\_id int, counter\_value counter, stud\_name text, book\_name text, book\_id int, date\_of\_issue date, primary key(stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue));

cqlsh:library> describe library\_info

CREATE TABLE library.library\_info (

stud\_id int,

stud\_name text,

book\_name text,

book\_id int,

date\_of\_issue date,

counter\_value counter,

PRIMARY KEY (stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue)

) WITH CLUSTERING ORDER BY (stud\_name ASC, book\_name ASC, book\_id 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';

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

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

cqlsh:library> update library\_info set counter\_value = counter\_value + 1 where stud\_id = 1 and stud\_name = 'Bruce' and book\_name = 'Game of Thrones' and book\_id = 1 and date\_of\_issue = '2022-04-20';

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

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

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

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

(1 rows)

cqlsh:library> update library\_info set counter\_value = counter\_value + 1 where stud\_id = 2 and stud\_name = 'Clark' and book\_name = 'Song of Ice and Fire' and book\_id = 2 and date\_of\_issue = '2022-04-21';

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

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

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

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

2 | Clark |Song of Ice and Fire | 2 | 2022-04-21 | 1

(2 rows)

cqlsh:library> update library\_info set counter\_value = counter\_value + 1 where stud\_id = 112 and stud\_name = 'Diana' and book\_name = 'BDA' and book\_id = 3 and date\_of\_issue = '2022-05-04';

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

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

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

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

2 | Clark |Song of Ice and Fire | 2 | 2022-04-21 | 1

112 | Diana | BDA | 3 | 2022-05-04 | 1

(3 rows)

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

cqlsh:library> update library\_info set counter\_value = counter\_value + 1 where stud\_id = 112 and stud\_name = 'Diana' and book\_name = 'BDA' and book\_id = 3 and date\_of\_issue = '2022-05-04';

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

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

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

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

2 | Clark |Song of Ice and Fire | 2 | 2022-04-21 | 1

112 | Diana | BDA | 3 | 2022-05-04 | 2

(3 rows)

cqlsh:library> select \* from library\_info where stud\_id = 112;

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

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

112 | Diana | BDA | 3 | 2022-05-04 | 2

(1 rows)

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

cqlsh:library> copy library\_info (stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value) to '/home/bmsce/Desktop/data.csv';

Using 11 child processes

Starting copy of library.library\_info with columns [stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value].

Processed: 4 rows; Rate: 21 rows/s; Avg. rate: 21 rows/s

4 rows exported to 1 files in 0.200 seconds.

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

cqlsh:library> copy library\_info (stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value) from '/home/bmsce/Desktop/data1.csv';

Using 11 child processes

Starting copy of library.library\_info with columns [stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value].

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

4 rows imported from 1 files in 0.381 seconds (0 skipped).

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

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

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

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

2 | Clark |Song of Ice and Fire | 2 | 2022-04-21 | 1

112 | Diana | BDA | 3 | 2022-05-04 | 2

1 | Bruce | Game of Thrones | 1 | 2022-04-20 | 1

2 | Clark |Song of Ice and Fire | 2 | 2022-04-21 | 1

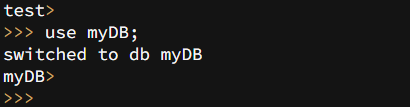
112 | Diana | BDA | 3 | 2022-05-04 | 2

**Lab: 3 WORKING WITH MONGODB**

1. **Create Database In Mongodb**

**use myDB;**

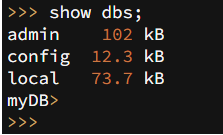
Confirm the existence of your database

****

**db;**

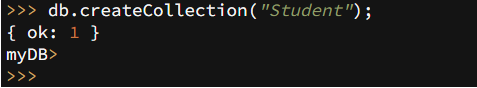
To list all databases

**show dbs;**

****

1. **CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS**
2. To create a collection by the name “Student”. Let us take a look at the collection list prior to the creation of the new collection “Student”.

**db.createCollection(“Student”); =>** *sql equivalent* **CREATE TABLE STUDENT(…);**

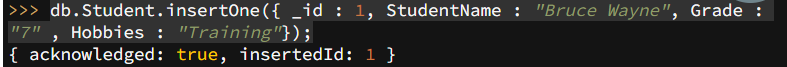
****

1. To drop a collection by the name “Student”.

**db.Student.drop();**

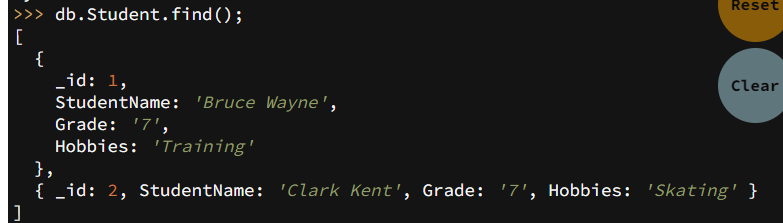
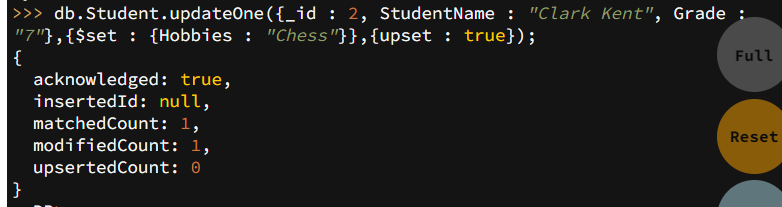
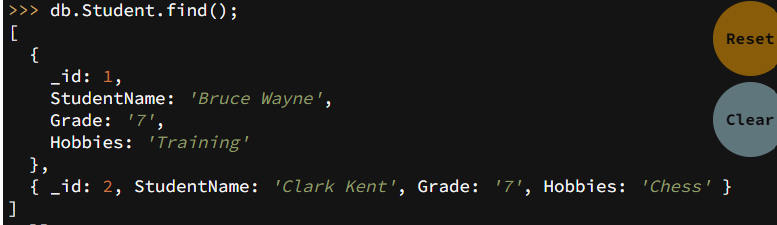
1. Create a collection by the name “Students” and store the following data in it.

**db.Student.insert({\_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});**

****

1. Insert the document for “AryanDavid” in to the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from “Skating” to “Chess”. ) Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

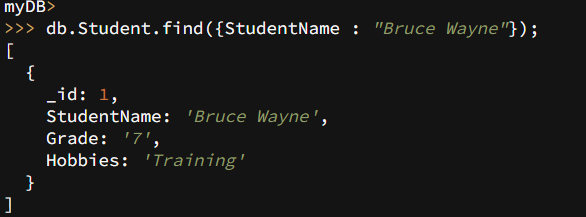
**db.Student.update({\_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});**

1. ****
2. ****
3. ****
4. FIND METHOD

A. To search for documents from the “Students” collection based on certain search criteria.

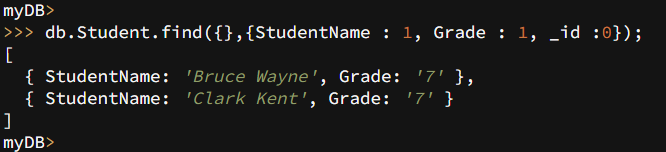
**db.Student.find({StudName:"Aryan David"});**

**({cond..},{columns.. column:1, columnname:0} )**



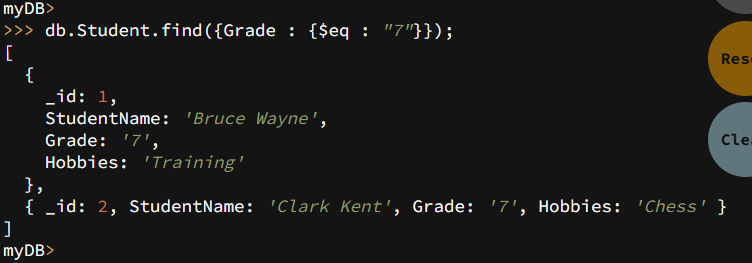
B. To display only the StudName and Grade from all the documents of the Students collection. The identifier\_id should be suppressed and NOT displayed.

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



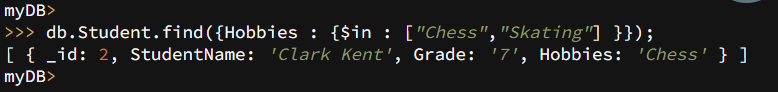
C. To find those documents where the Grade is set to ‘VII’

**db.Student.find({Grade:{$eq:'VII'}}).pretty();**



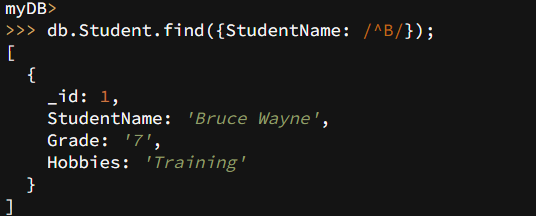
D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

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



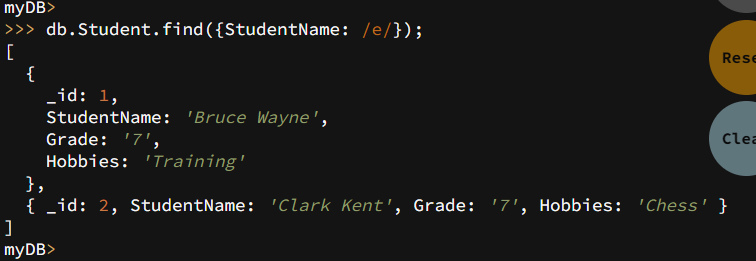
E. To find documents from the Students collection where the StudName begins with “M”.

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



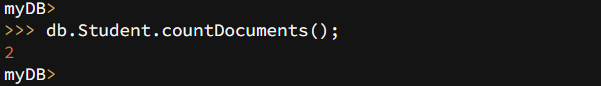
F. To find documents from the Students collection where the StudName has an “e” in any position.

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



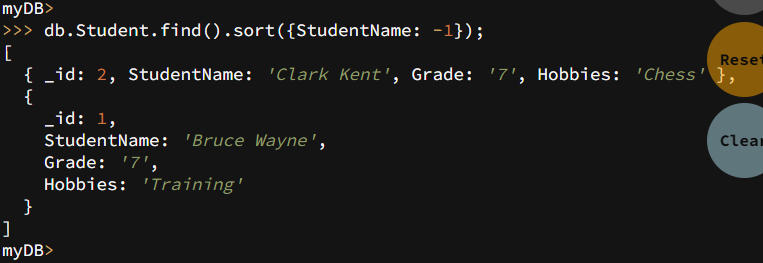
G. To find the number of documents in the Students collection.

**db.Student.count();**



H. To sort the documents from the Students collection in the descending order of StudName.

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



1. **Import data from a CSV file**

Given a CSV file “sample.txt” in the D:drive, import the file into the MongoDB collection, “SampleJSON”. The collection is in the database “test”.

**mongoimport --db Student --collection airlines --type csv –headerline --file /home/hduser/Desktop/airline.csv**

1. **Export data to a CSV file**

This command used at the command prompt exports MongoDB JSON documents from “Customers” collection in the “test” database into a CSV file “Output.txt” in the D:drive.

**mongoexport --host localhost --db Student --collection airlines --csv --out /home/hduser/Desktop/output.txt –fields “Year”,”Quarter”**

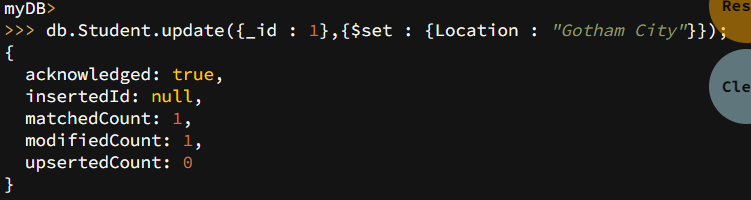
1. **Save Method :**

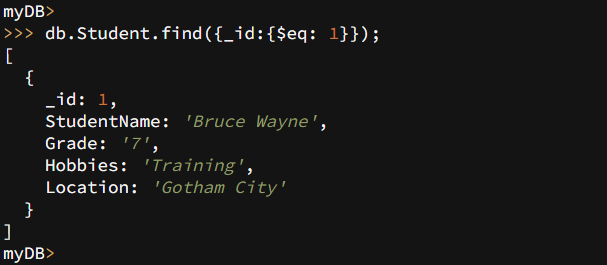
**Save() method will insert a new document, if the document with the \_id does not exist. If it exists it will replace the exisiting document.**

db.Students.save({StudName:”Vamsi”, Grade:”VI”})

1. **Add a new field to existing Document:**

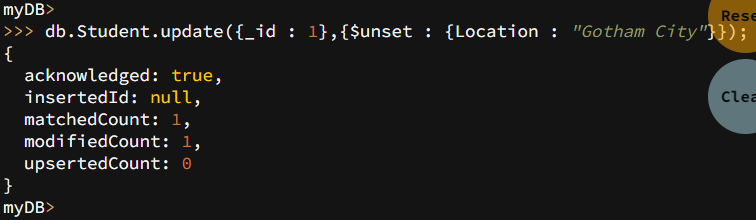
db.Students.update({\_id:4},{$set:{Location:”Network”}})





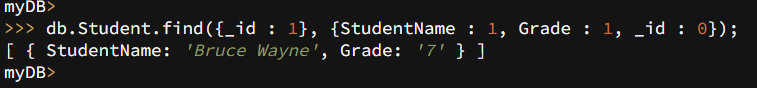
1. **Remove the field in an existing Document**

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



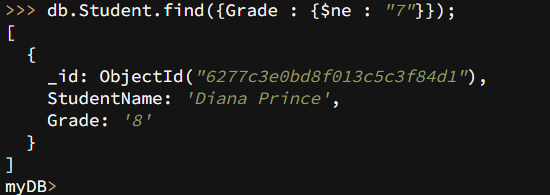
1. **Finding Document based on search criteria suppressing few fields**

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



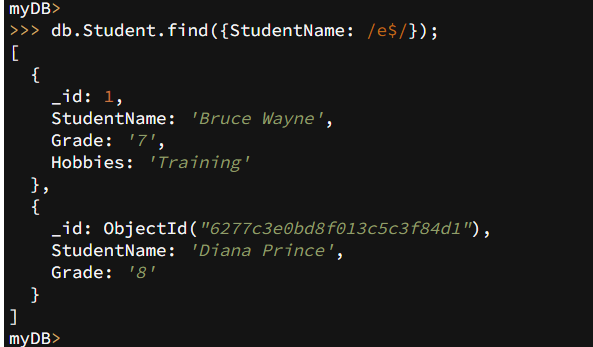
**To find those documents where the Grade is not set to ‘VII’**

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



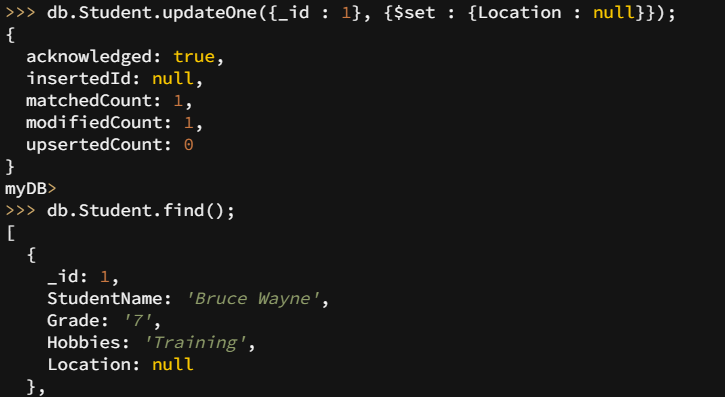
**To find documents from the Students collection where the StudName ends with s.**

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



1. **to set a particular field value to NULL**

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



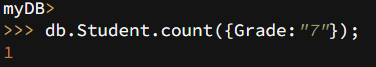
1. **Count the number of documents in Student Collections**

db.Students.count()



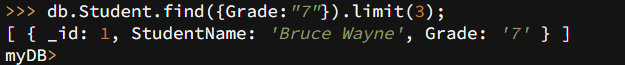
1. **Count the number of documents in Student Collections with grade :VII**

db.Students.count({Grade:”VII”})



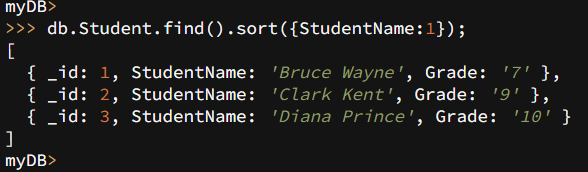
**retrieve first 3 documents**

db.Students.find({Grade:”VII”}).limit(3).pretty();



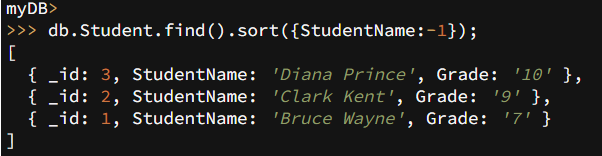
**Sort the document in Ascending order**

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



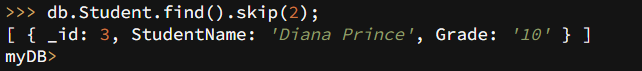
**Note:**

**for desending order :** db.Students.find().sort({StudName:-1}).pretty();



**to Skip the 1st two documents from the Students Collections**

db.Students.find().skip(2).pretty()

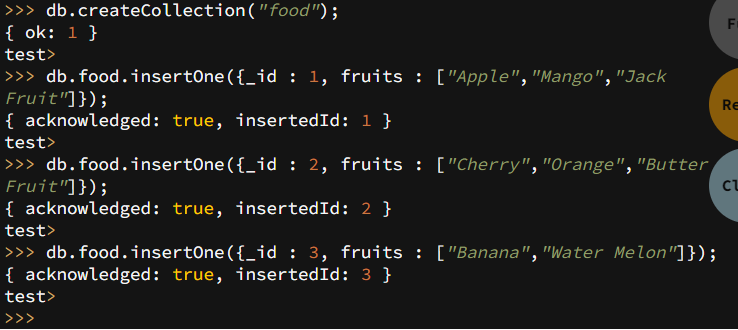


* Create a collection by name “food” and add to each document add a “fruits” array

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

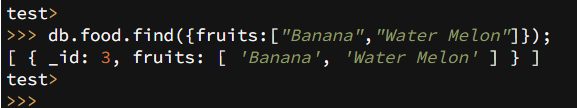
db.food.insert( { \_id:2, fruits:['grapes','mango','cherry'] } )

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



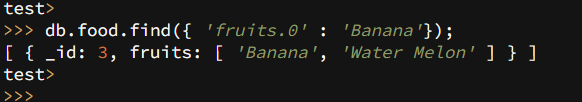
**To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”.**

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



**To find in “fruits” array having “mango” in the first index position.**

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

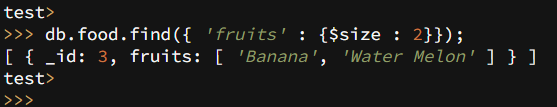


**To find those documents from the “food” collection where the size of the array is two.**

db.food.find ( {“fruits”: {$size:2}} )

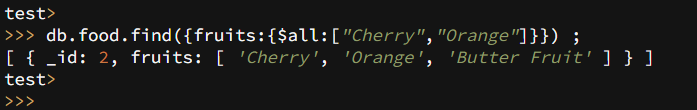
**To find the document with a particular id and display the first two elements from the array “fruits”**

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



**To find all the documets from the food collection which have elements mango and grapes in the array “fruits”**

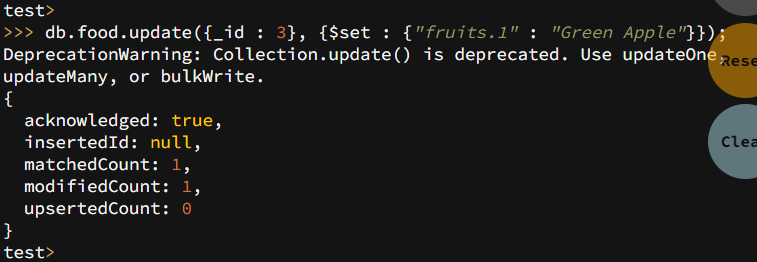
db.food.find({fruits:{$all:[“mango”,”grapes”]}})



**update on Array:**

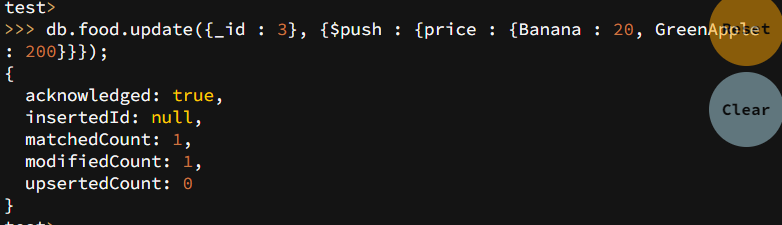
**using particular id replace the element present in the 1st index position of the fruits array with apple**

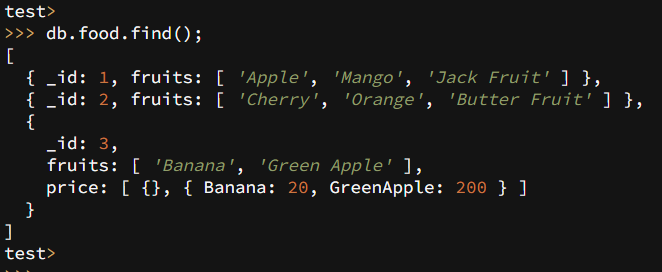
db.food.update({\_id:3},{$set:{'fruits.1':'apple'}})



insert new key value pairs in the fruits array

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



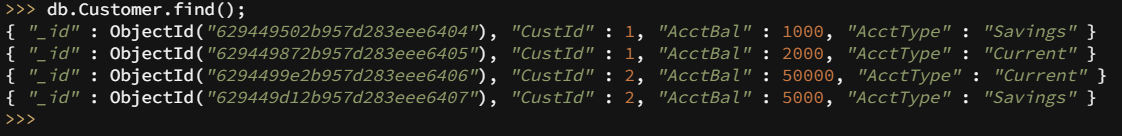


Note: perform query operations using - pop, addToSet, pullAll and pull

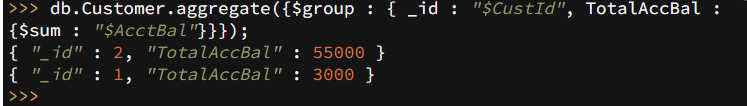
* **Aggregate Function :**

**Create a collection Customers with fields custID, AcctBal, AcctType.**

**Now group on “custID” and compute the sum of “AccBal”.**

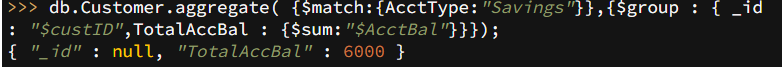
****

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



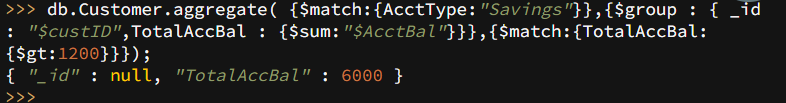
**match on AcctType:”S” then group on “CustID” and compute the sum of “AccBal”.**

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



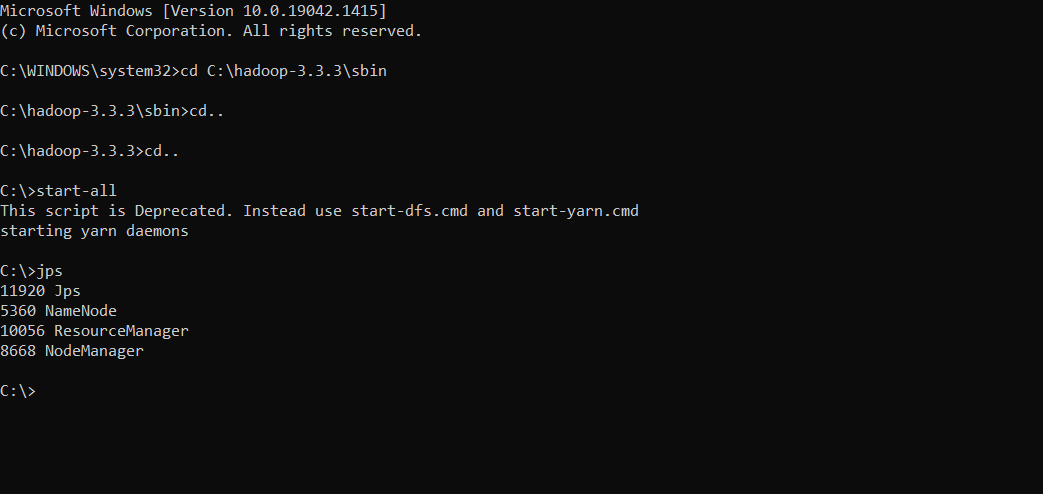
**match on AcctType:”S” then group on “CustID” and compute the sum of “AccBal” and total balance greater than 1200.**

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



**Lab:4 Hadoop Installation.**

* **Screenshot of Hadoop installed**



**Lab: 5**

**Execution of HDFS Commands for interaction with Hadoop Environment.**

c:\hadoop\_new\sbin>hdfs dfs -mkdir /temp

c:\hadoop\_new\sbin>hdfs dfs –copyFromLocal

E:\Desktop\sample.txt \temp c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 1 items -rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt

c:\hadoop\_new\sbin>hdfs dfs -cat \temp\sample.txt hello world

c:\hadoop\_new\sbin>hdfs dfs -get \temp\sample.txt

E:\Desktop\temp

c:\hadoop\_new\sbin>hdfs dfs -put E:\Desktop\temp \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 2 items -rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp c:\hadoop\_new\sbin>hdfs dfs -mv \lab1 \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin 7 supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -rm /temp/sample.txt Deleted /temp/sample.txt

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 2 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp c:\hadoop\_new\sbin>hdfs dfs -copyFromLocal

E:\Desktop\sample.txt \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:17 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -copyToLocal \temp\sample.txt E:\Desktop\sample.txt

**Lab: 7**

**For the given file, Create a Map Reduce program to a) Find the average temperature for each year from the NCDC data set.**

// AverageDriver.java package temperature; import org.apache.hadoop.io.\*;

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

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

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.java package temperature;

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

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

import java.io.IOException;

public class AverageMapper extends Mapper

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

String line = value.toString();

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

int temperature; 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 != MISSING && quality.matches("[01459]")) context.write(new Text(year),new IntWritable(temperature)); } 12 } //AverageReducer.java package temperature;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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

import java.io.IOException;

public class AverageReducer extends Reducer

{

public void reduce(Text key, Iterable values, Context context) throws IOException,InterruptedException {

int max\_temp = 0;

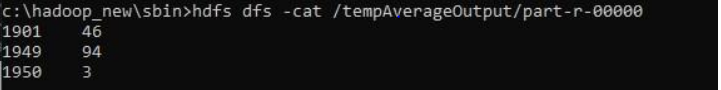
int count = 0;

for (IntWritable value : values) { max\_temp += value.get();

count+=1;

}

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



//TempDriver.java package temperatureMax;

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

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

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

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

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

public class TempDriver { 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(TempDriver.class);

job.setJobName("Max temperature");

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

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

job.setMapperClass(TempMapper.class);

job.setReducerClass(TempReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

//TempMapper.java package temperatureMax;

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

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

import java.io.IOException;

public class TempMapper extends Mapper

{ public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException { 14 String line = value.toString();

String month = line.substring(19,21);

int temperature;

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 != MISSING && quality.matches("[01459]")) context.write(new Text(month),new IntWritable(temperature)); } }

//TempReducer.java package temperatureMax;

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

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

import java.io.IOException;

public class TempMapper extends Mapper {

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{ String line = value.toString();

String month = line.substring(19,21);

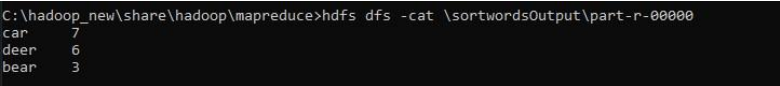
int temperature;

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 != MISSING && 15 quality.matches("[01459]")) context.write(new Text(month),new IntWritable(temperature)); } }



**Lab 8**

**Create a Map Reduce program to demonstrating join operation**

* Program

// 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&lt;TextPair,

Text&gt; {

@Override

public void configure(JobConf job) {}

@Override

public int getPartition(TextPair key, Text value, int numPartitions) {

return (key.getFirst().hashCode() &amp; Integer.MAX\_VALUE) %

numPartitions;

}

}

@Override

public int run(String[] args) throws Exception {

if (args.length != 3) {

System.out.println(&quot;Usage: &lt;Department Emp Strength input&gt;

&lt;Department Name input&gt; &lt;output&gt;&quot;);

return -1;

}

JobConf conf = new JobConf(getConf(), getClass());

conf.setJobName(&quot;Join &#39;Department Emp Strength input&#39; with

&#39;Department Name

input&#39;&quot;);

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.cl

ass);

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&lt;TextPair, Text, Text,

Text&gt; {

@Override

public void reduce (TextPair key, Iterator&lt;Text&gt; values,

OutputCollector&lt;Text, Text&gt;

output, Reporter reporter)

throws IOException

{

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

while (values.hasNext()) {

Text node = values.next();

Text outValue = new Text(nodeId.toString() + &quot;\t\t&quot; + 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&lt;LongWritable, Text, TextPair,

Text&gt; {

@Override

public void map(LongWritable key, Text value,

OutputCollector&lt;TextPair, Text&gt; output,

Reporter reporter)

throws IOException

{

String valueString = value.toString();

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

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

Text(SingleNodeData[1]));

}

}

//Posts.java

import java.io.IOException;

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

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

public class Posts extends MapReduceBase implements

Mapper&lt;LongWritable, Text, TextPair,

Text&gt; {

@Override

public void map(LongWritable key, Text value,

OutputCollector&lt;TextPair, Text&gt; output,

Reporter reporter)

throws IOException

{

String valueString = value.toString();

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

output.collect(new TextPair(SingleNodeData[3], &quot;0&quot;), new

Text(SingleNodeData[9]));

}

}

// TextPair.java

import java.io.\*;

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

public class TextPair implements WritableComparable&lt;TextPair&gt; {

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) &amp;&amp; second.equals(tp.second);

}

return false;

}

@Override

public String toString() {

return first + &quot;\t&quot; + 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 &amp;&amp; b instanceof TextPair) {

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

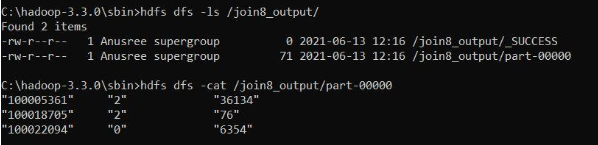
}

return super.compare(a, b);

}

} }

* output



**Lab:9**

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

* commands and outline:

hduser@bmsce-OptiPlex-3060:~$ spark-shell

22/06/28 09:34:37 WARN Utils: Your hostname, bmsce-OptiPlex-3060 resolves to a loopback address: 127.0.1.1; using 10.124.7.72 instead (on interface enp1s0)

22/06/28 09:34:37 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

22/06/28 09:34:37 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Spark context Web UI available at http://10.124.7.72:4040

Spark context available as 'sc' (master = local[\*], app id = local-1656389082904).

Spark session available as 'spark'.

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.4.8

/\_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0\_312)

Type in expressions to have them evaluated.

Type :help for more information.

scala> println("hello");

hello

scala> val data=sc.textFile("/home/hduser/Desktop/sample.txt");

data: org.apache.spark.rdd.RDD[String] = /home/hduser/Desktop/sample.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> data.collect;

res1: Array[String] = Array(hi hw are ypu, how is your job, how is your family, how is your brother, how is your sister)

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

splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:25

scala> splitdata.collect;

res2: Array[String] = Array(hi, hw, are, ypu, how, is, your, job, how, is, your, family, how, is, your, brother, how, is, your, sister)

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

mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> mapdata.collect;

res3: Array[(String, Int)] = Array((hi,1), (hw,1), (are,1), (ypu,1), (how,1), (is,1), (your,1), (job,1), (how,1), (is,1), (your,1), (family,1), (how,1), (is,1), (your,1), (brother,1), (how,1), (is,1), (your,1), (sister,1))

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

reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:25

scala> reducedata.collect;

res4: Array[(String, Int)] = Array((are,1), (brother,1), (is,4), (sister,1), (family,1), (how,4), (ypu,1), (job,1), (hi,1), (hw,1), (your,4))

**LAB-10**

**Using RDD and FlaMap 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**

* commands and output:

cala> val textFile=sc.textFile("/home/hduser/Desktop/sample.txt");

textFile: org.apache.spark.rdd.RDD[String] = /home/hduser/Desktop/sample.txt MapPartitionsRDD[8] at textFile at <console>:24

scala> val counts=textFile.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey(\_=\_)

<console>:25: error: reassignment to val

val counts=textFile.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey(\_=\_)

^

scala> val counts=textFile.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey(\_+\_)

counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:25

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] = Map(is -> 4, how -> 4, your -> 4, are -> 1, brother -> 1, sister -> 1, family -> 1, ypu -> 1, job -> 1, hi -> 1, hw -> 1)

scala> println(sorted)

Map(is -> 4, how -> 4, your -> 4, are -> 1, brother -> 1, sister -> 1, family -> 1, ypu -> 1, job -> 1, hi -> 1, hw -> 1)

scala> for((k,v)<-sorted)

| {

| if(v>4)

| {

| print(k+",")

| print(v)

| println()

| }

| }

//SINCE SAMPLE TEXT FILE DOESNT HAVE WORD WITH FREQUENCY >4,NO OUTPUT