VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

NIHARIKA B S (1BM19CS100)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019
May-2022 to July-2022
(Autonomous Institution under VTU)

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**" was carried out by **NIHARIKA B S(1BM19CS100),** who is bona fide 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 the course **BIG DATA ANALYTICS (20CS6PEBDA)** work prescribed for the said degree.

Name of the Lab-In charge	**PALLAVI GB**
Designation	Assistant Professor
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

`

Index Sheet

Course Outcome

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

Cassandra Lab Program 1: -

Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

```
cqlsh> create keyspace Employee2 with replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe Employee2;

CREATE KEYSPACE employee2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;
```

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> create keyspace Employee2 with replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe Employee2;

CREATE KEYSPACE employee2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;
```

3. Insert the values into the table in batch

```
cqlsh> begin batch insert into Employee2.Employee_Info(emp_id,date_of_jocqlshcqlsh> begin batch insert into Employee2.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(1,'2
022-04-27','Deployment','Team lead','Prem Sai',1000000.0);apply batch;
cqlsh> select * from Employee2.Employee_Info;

 emp_id | date_of_joining                 | dept_name  | designation | emp_name | salary
--------+---------------------------------+------------+-------------+----------+--------
      1 | 2022-04-26 18:30:00.000000+0000 | Deployment |   Team lead | Prem Sai |  1e+06

cqlsh> begin batch insert into Employee2.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(2,'2020-03-07','Development','Manager','Tarun',1500000.0);insert into Employee2.E
mployee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(3,'2021-03-29','R&D','Web developer','Nithish',750000.0); apply batch;
cqlsh> select * from Employee2.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation   | emp_name | salary
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |         R&D | Web developer |  Nithish | 7.5e+05

(3 rows)
```

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

5. Sort the details of Employee records based on salary

```
cqlsh> update Employee2.Employee_Info set emp_name = 'Harsha',dept_name='Testing' where emp_id = 3;
cqlsh> select * from Employee2.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation   | emp_name | salary
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | 7.5e+05

cqlsh> create table Employee2.emp(Emp_Id int ,Salary double,primary key(Emp_Id,Salary));
cqlsh> begin batch
   ... insert into Employee2.emp(emp_Id,salary) values(1,1000000);
   ... insert into Employee2.emp(emp_Id,salary) values(2,1500000);
   ... insert into Employee2.emp(emp_Id,salary) values(3,700000);
   ... apply batch;
cqlsh> select * from Employee2.emp;

 emp_id | salary
--------+---------
      1 |   1e+06
      2 | 1.5e+06
      3 |   7e+05

(3 rows)
cqlsh> paging off;
Disabled Query paging.
cqlsh> select * from Employee2.emp where emp_id in(1,2,3) order by salary;

 emp_id | salary
--------+---------
      3 |   7e+05
      1 |   1e+06
      2 | 1.5e+06

(3 rows)
```

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.

7. Update the altered table to add project names.

```
cqlsh> alter table Employee2.Employee_Info add Projects set<text>;
scqlsh> select * from Employee2.Employee_Info ;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects | salary
--------+---------------------------------+-------------+----------------+----------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai |     null |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun |     null | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha |     null | 7.5e+05

cqlsh> update Employee2.Employee_Info set projects= projects+{'abc','xyz'} where emp_id=1;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun |           null | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha |           null | 7.5e+05

(3 rows)
cqlsh> update Employee2.Employee_Info set projects= projects+{'def','pqr'} where emp_id=2;
cqlsh> update Employee2.Employee_Info set projects= projects+{'gjd','ads'} where emp_id=3;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05
```

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

//BEFORE 15 seconds

```
cqlsh> insert into Employee2.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(10,'2020-02-27','Development','Intern','XYZ',150000.0) using TTL 15;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
     10 | 2020-02-26 18:30:00.000000+0000 | Development |         Intern |      XYZ |           null | 1.5e+05
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05

cqlsh> select * from Employee2.Employee_Info ;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05

(3 rows)
```

Cassandra Lab Program 2: -

Perform the following DB operations using Cassandra.

1.Create a key space by name Library

```
cqlsh> create keyspace Library with replication = {'class':'SimpleStrategy','replication_factor':2};
```

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 timestamp,primary key(Stud_id,Stud_Name,Book_Name,Book_Id,Date_o
f_issue));
```

3.Insert the values into the table in batch

```
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 1 and Stud_Nam
e = 'Prem Sai' and Book_Name = 'Big Data Analysis' and Book_Id = 1000 and Date_of_issue='2022-04-29';
cqlsh:library> select * from Library_INfo;

 stud_id | stud_name | book_name         | book_id | date_of_issue                   | counter_value
---------+-----------+-------------------+---------+---------------------------------+---------------
       1 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-28 18:30:00.000000+0000 |             1

(1 rows)
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'OOMD' and Book_Id = 1020 and Date_of_issue='2022-05-04';
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'BDA' and Book_Id = 1100 and Date_of_issue='2022-03-06';
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'BDA' and Book_Id = 1100 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 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-28 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-03-05 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-05-03 18:30:00.000000+0000 |             1
     112 |     Tarun |              OOMD |    1020 | 2022-05-03 18:30:00.000000+0000 |             1
```

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 = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'BDA' and Book_Id = 1100 and Date_of_issue='2022-04-30';
cqlsh:library> select * from Library_INfo;

 stud_id | stud_name | book_name         | book_id | date_of_issue                   | counter_value
---------+-----------+-------------------+---------+---------------------------------+---------------
       1 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-19 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-04-29 18:30:00.000000+0000 |             2
```

5. Write a query to show that a student with id 112 has taken a book “BDA” 3 times.

```
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'BDA' and Book_Id = 1100 and Date_of_issue='2022-04-30';
cqlsh:library> select * from Library_INfo;

 stud_id | stud_name | book_name         | book_id | date_of_issue                   | counter_value
---------+-----------+-------------------+---------+---------------------------------+---------------
       1 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-19 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-04-29 18:30:00.000000+0000 |             2
```

6. Export the created column to a csv file

```
cqlsh:library> update Library_Info set Counter_value = Counter_value+1 where Stud_id = 112 and Stud_N
ame = 'Tarun' and Book_Name = 'BDA' and Book_Id = 1100 and Date_of_issue='2022-04-30';
cqlsh:library> select * from Library_INfo;

 stud_id | stud_name | book_name         | book_id | date_of_issue                   | counter_value
---------+-----------+-------------------+---------+---------------------------------+---------------
       1 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-19 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-04-29 18:30:00.000000+0000 |             2
```

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

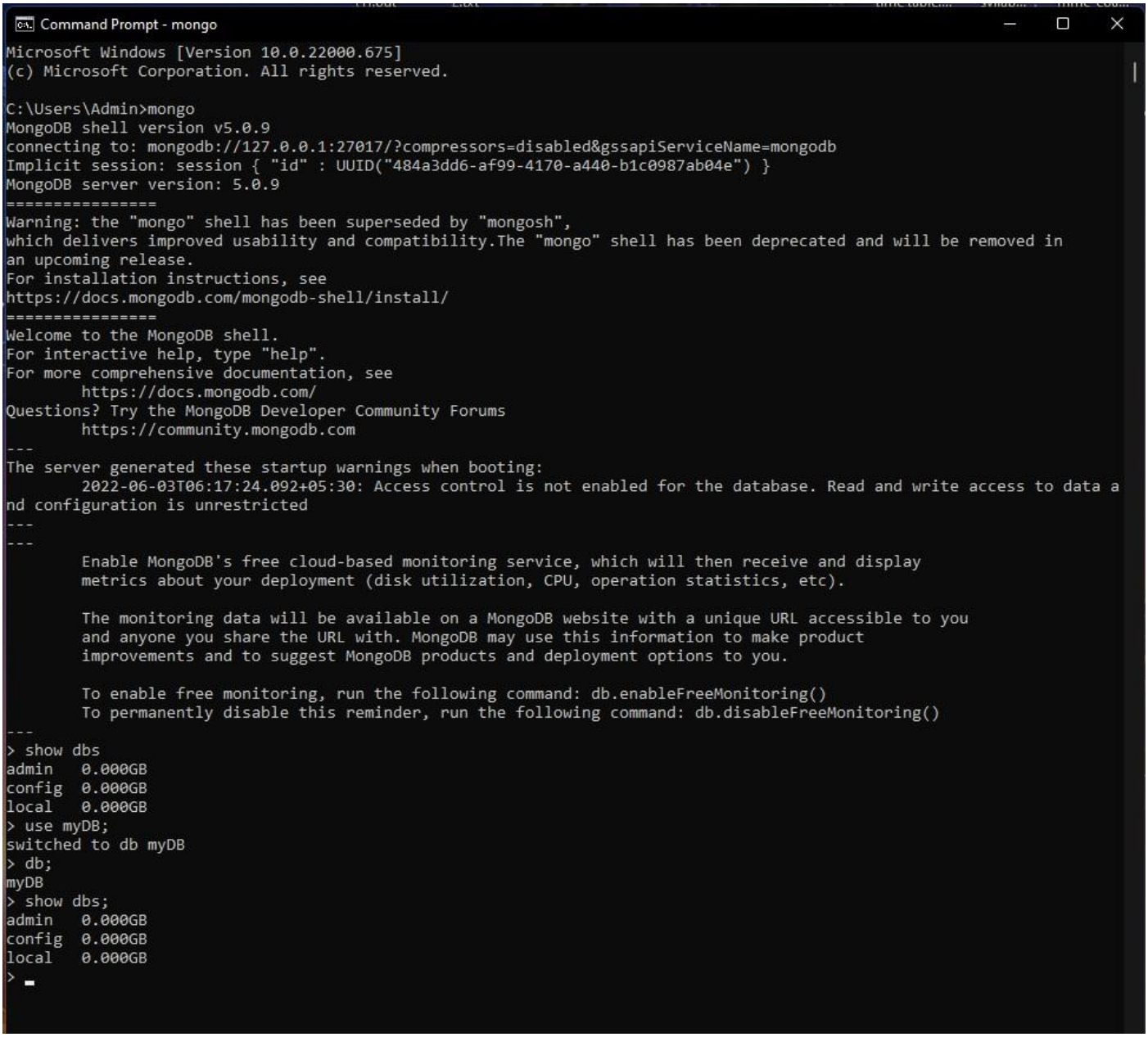
```
cqlsh:library> copy library_Info2(Stud_Id,Stud_Name,Book_Name,Book_Id,date_of_issue,counter_value) fr
om  '/home/bmsce/BDA/lib.csv';
Using 11 child processes

Starting copy of library.library_info2 with columns [stud_id, stud_name, book_name, book_id, date_of_
issue, counter_value].
Processed: 2 rows; Rate:       3 rows/s; Avg. rate:       5 rows/s
2 rows imported from 1 files in 0.401 seconds (0 skipped).
cqlsh:library> select * from library_info2;

 stud_id | stud_name | book_name         | book_id | date_of_issue                   | counter_value
---------+-----------+-------------------+---------+---------------------------------+---------------
       1 |  Prem Sai | Big Data Analysis |    1000 | 2022-04-19 18:30:00.000000+0000 |             1
     112 |     Tarun |               BDA |    1100 | 2022-04-29 18:30:00.000000+0000 |             2
```

MongoDB Lab Program 1 (CRUD Demonstration): -

Execute the queries and upload a document with output.

I. CREATE DATABASE IN MONGODB.

use myDB; db; (Confirm the existence of your database) show dbs; (To list all databases)



II.CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

1. 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(…);

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

db.Student.drop();

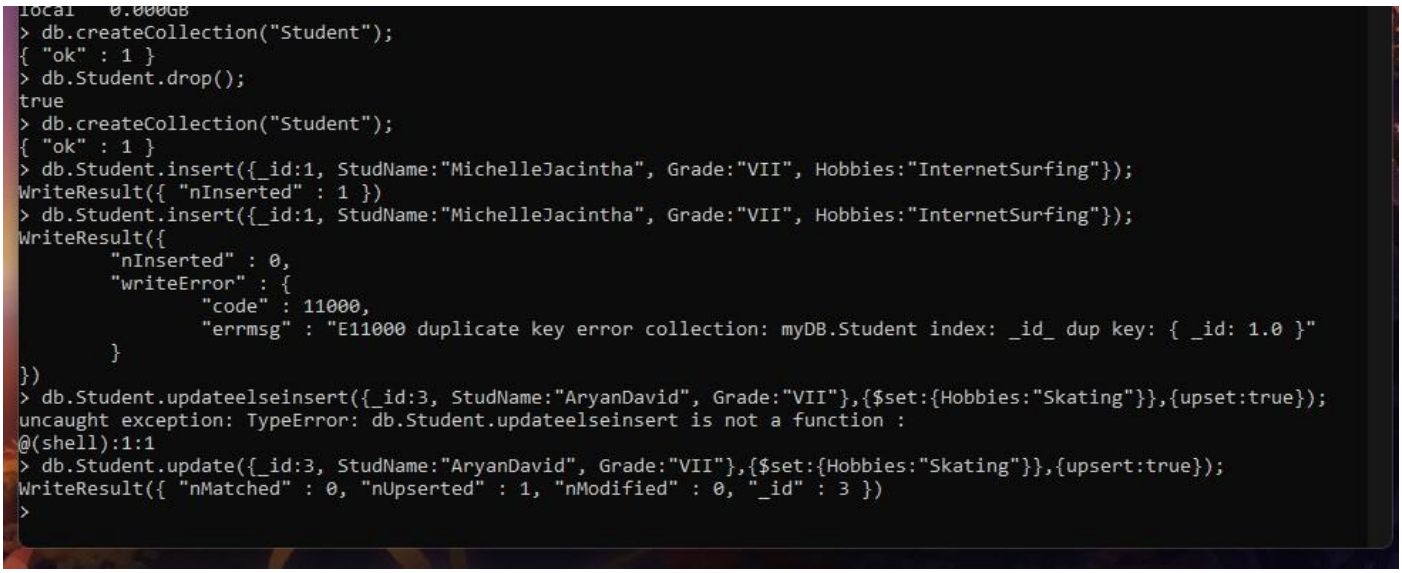
3. Create a collection by the name “Students” and store the following data in it.

db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetS urfing"});

4. 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:&quo t;Skatin g"}},{upsert:true});

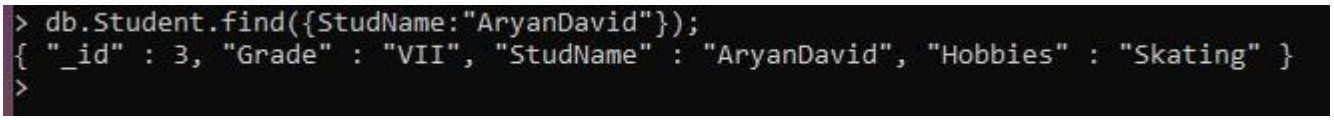


5. FIND METHOD

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

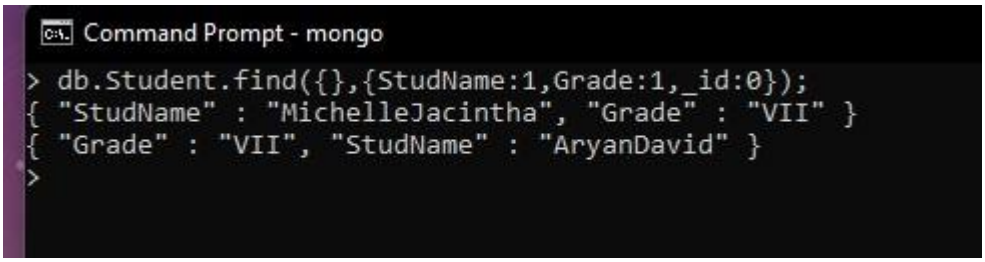
db.Student.find({StudName:"Aryan David"});

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



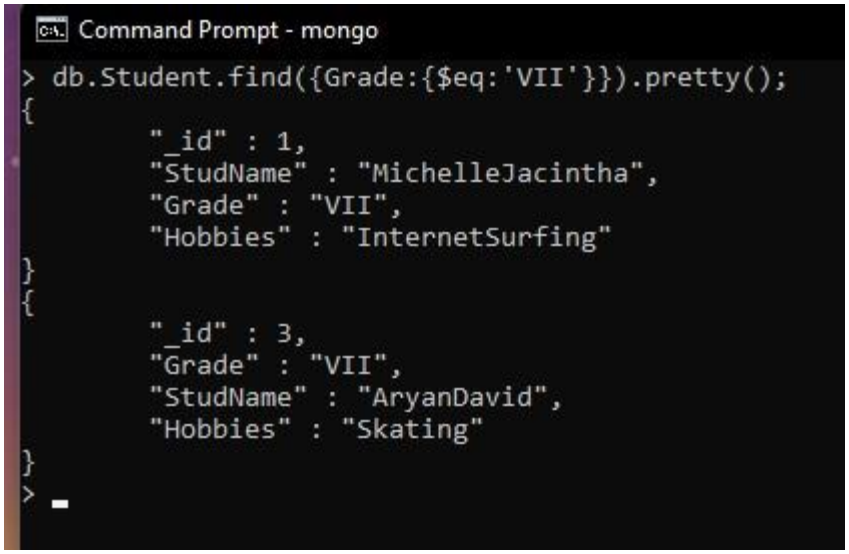
B. To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

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



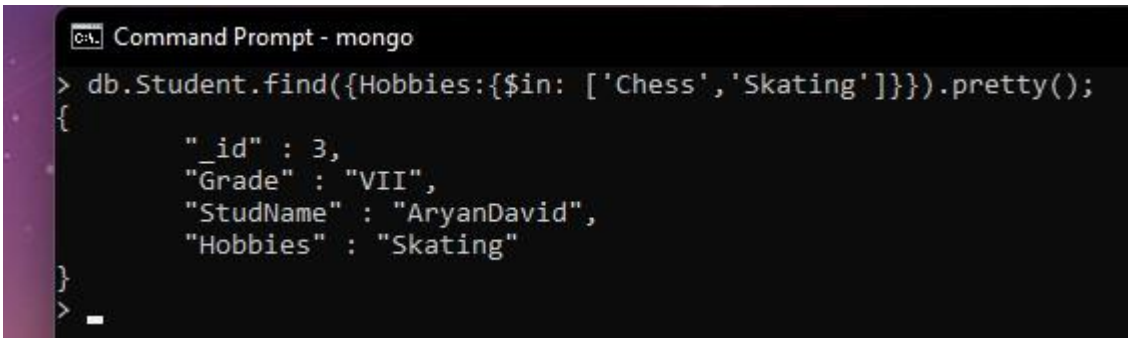
C. To find those documents where the Grade is set to ‘VII’

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

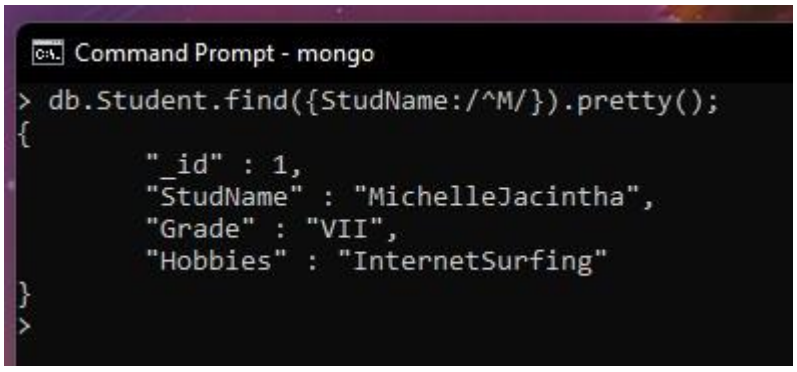


D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

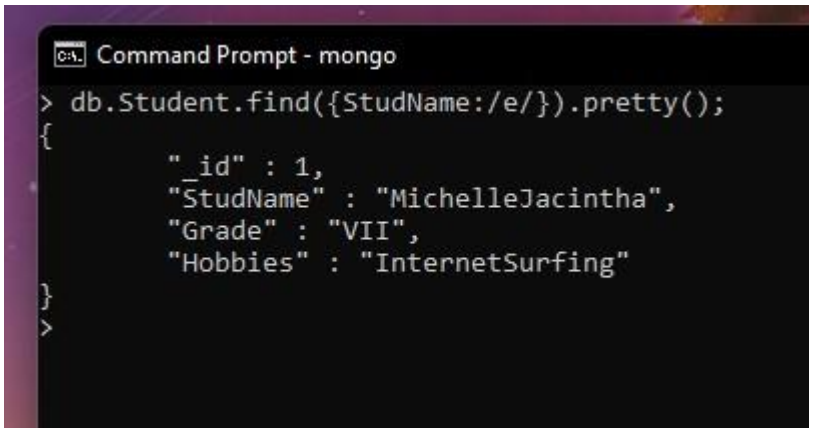
db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();



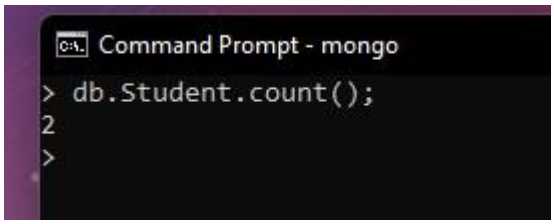
E. To find documents from the Students collection where the StudName begins with “M”. db.Student.find({StudName:/^M/}).pretty();



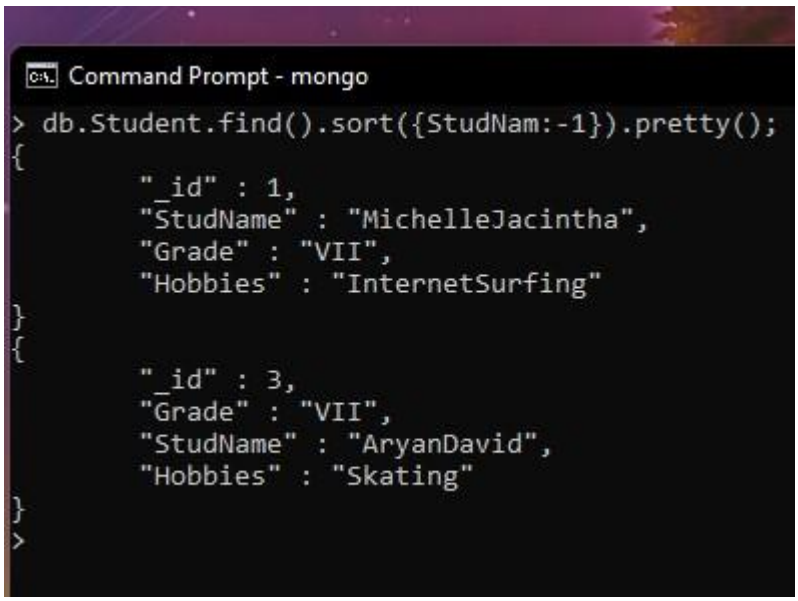
F. To find documents from the Students collection where the StudNamehas an “e” in any position. db.Student.find({StudName:/e/}).pretty();



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();

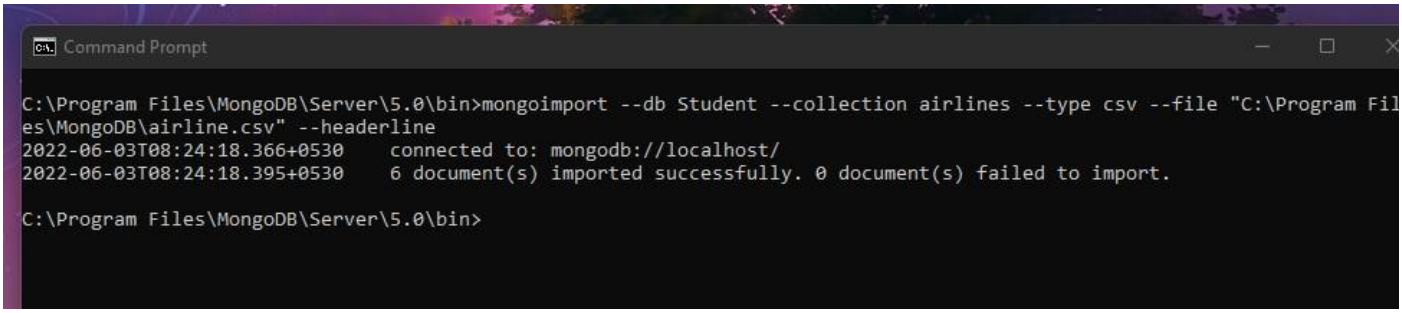


III. Import data from a CSV file

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



IV. 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”

V. 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”})

```
switched to db Student
> db.Students.save({StudName:"Vamsi",Grade:"VII"})
WriteResult({ "nInserted" : 1 })
>
```

VI. Add a new field to existing Document: db.Students.update({_id:4},{$set:{Location:”Network”}})

```
> db.Students.update({_id:4},{$set:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

VII. Remove the field in an existing Document

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

```
Command Prompt - mongo
> db.Students.update({_id:4},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

VIII. 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();

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

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

IX. to set a particular field value to NULL

```
> db.Students.update({_id:3},{$set:{Location:null}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

X Count the number of documents in Student Collections

```
> db.Student.count()
0
>
```

XI. 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 1 st two documents from the Students Collections db.Students.find().skip(2).pretty()

```
> db.Students.find().sort({StudName:1}).pretty();
{
        "_id" : ObjectId("629979944de3211e43081306"),
        "StudName" : "Vamsi",
        "Grade" : "VII"
}
>
```

XII. 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'] })

```
Command Prompt - mongo
> 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 })
>
```

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().

```
> db.food.find({fruits:['grapes','mango','apple']}).pretty()
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
>
```

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

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

```
> 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}})

```
> db.food.find ( {"fruits": {$size:2}} )
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
>
```

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}})

```
> db.food.find({_id:1},{"fruits":{$slice:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
>
```

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”]}})

```
> db.food.find({fruits:{$all:["mango","grapes"]}})
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
>
```

update on Array: using particular id replace the element present in the 1 st 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

XII. 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 (

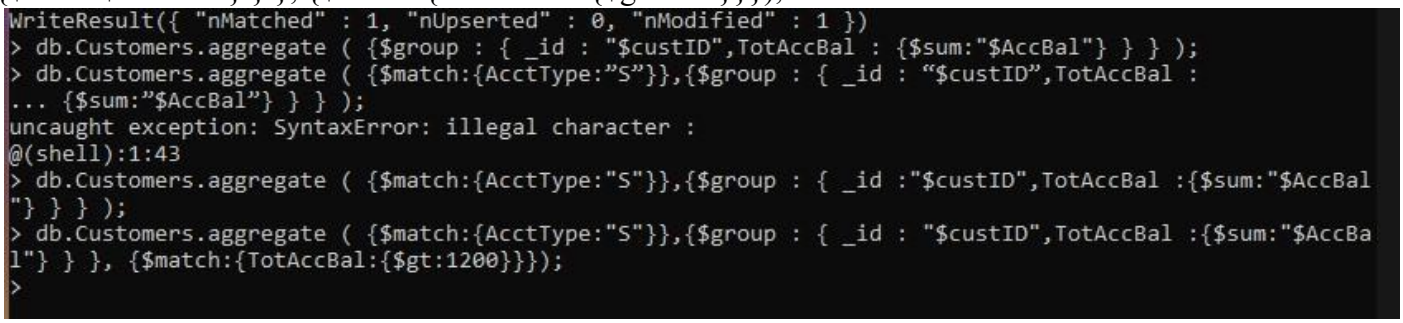
{$match:{AcctType:”S”}},{$group : { _id : “$custID”,TotAccBal :

{$sum:”$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}}});



MongoDB Lab Program 2 (CRUD Demonstration): -

1) Using MongoDB

i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)). ii) Insert required documents to the collection.

iii)First Filter on “Dept_Name:CSE” and then group it on “Semester” and

compute the Average CPGA for that semester and flter those documents where the “Avg_CPGA” is greater than 7.5.

iv)Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV fle “Output.txt”.

```
> db.createCollection("Student");
{ "ok" : 1 }
```

```
> db.Student.insert({_id:1,name:"ananya",USN:"1BM19CS095",Sem:6,Dept_Name:"CSE",CGPA:"8.1",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,name:"bharath",USN:"1BM19CS002",Sem:6,Dept_Name:"CSE",CGPA:"8.3",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,name:"chandana",USN:"1BM19CS006",Sem:6,Dept_Name:"CSE",CGPA:"7.1",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,name:"hrithik",USN:"1BM19CS010",Sem:6,Dept_Name:"CSE",CGPA:"8.6",Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,name:"kanika",USN:"1BM19CS090",Sem:6,Dept_Name:"CSE",CGPA:"9.2",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:2},{$set:{CGPA:9.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:3},{$set:{CGPA:8.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:4},{$set:{CGPA:6.5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:5},{$set:{CGPA:8.6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.aggregate({$match:{Dept_Name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
> db.Student.aggregate({$match:{Dept_Name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
{ "_id" : 6, "AvgCGPA" : 8.26 }
--fields _id , Name , USN , Sem , Dept_Name , CGPA , Hobbies
2022-04-20T15:13:53.933+0530    csv flag is deprecated; please use --type=csv instead
2022-04-20T15:13:53.935+0530    connected to: localhost
2022-04-20T15:13:53.935+0530    exported 5 records
```

```
1 _id,Name,USN,Sem,Dept_Name,CGPA,Hobbies
2 1,,1BM19CS095,6,CSE,9,Badminton
3 2,,1BM19CS002,6,CSE,9.1,Swimming
4 3,,1BM19CS006,6,CSE,8.1,Cycling
5 4,,1BM19CS010,6,CSE,6.5,Reading
6 5,,1BM19CS090,6,CSE,8.6,Cycling
```

2)Create a mongodb collection Bank.

Demonstrate the following by choosing fields of your choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors
5. Updation

```
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729" ] }
> db.Bank.getIndexes()
[
        {
                "v" : 2,
```

```
@(shell):1:20
> db.Bank.update({_id:625d78659329139694f188a6}, {$set: {CustID:5}},{upsert:true});
uncaught exception: SyntaxError: identifier starts immediately after numeric literal :
@(shell):1:20
> db.Bank.update({_id:"625d78659329139694f188a6"}, {$set: {CustID:5}},{upsert:true});
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "625d78659329139694f188a6"
})
```

```
> db.createCollection("Bank");
{ "ok" : 1 }
> db.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
uncaught exception: TypeError: db.insert is not a function :
@(shell):1:1
> db.Bank.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:2, Name:"Vishvesh Bhat", Type:"Savings", Contact:["6325985615", "080-23651452"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:3, Name:"Vaishak Bhat", Type:"Savings", Contact:["8971456321", "080-33529458"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Pramod P Parande", Type:"Current", Contact:["9745236589", "080-56324587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Shreyas R S", Type:"Current", Contact:["9445678321","044-65611729", "080-25639856"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231", "080-22364587" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729", "080-25639856" ] }
> db.Bank.updateMany({CustID:1},{$pop:{Contact:1}} );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729", "080-25639856" ] }
```

1) Using MongoDB,

i) Create a database for Faculty and Create a Faculty Collection(Faculty_id, Name, Designation ,Department, Age, Salary, Specialization(Set)). ii) Insert required documents to the collection.

iii) First Filter on “Dept_Name:MECH” and then group it on “Designation” and

compute the Average Salary for that Designation and flter those documents

where the “Avg_Sal” is greater than 650000. iv) Demonstrate usage of import and export commands

Write MongoDB queries for the following:

1)To display only the product name from all the documents of the product collection.

2)To display only the Product ID, ExpiryDate as well as the quantity from the document of the product collection where the _id column is 1.

3)To fnd those documents where the price is not set to 15000.

4)To fnd those documents from the Product collection where the quantity is set to 9 and the product name is set to ‘monitor’.

5)To fnd documents from the Product collection where the Product name ends in ‘d’.

```
}
> db.createCollection("faculty");
{ "ok" : 1 }
> db.faculty.insert({_id:1,name:"Dr. Balaraman Ravindran",designation:"Professor",department:"CSE",age:45,salary:100000,specialization:['pytho
n','mysql','sklearn', 'tensorflow']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:2,name:"Dr. Mahadev Ghorki",designation:"Assistant Professor",department:"CSE",age:35,salary:80000,specialization:['p
ython','numpy','sklearn', 'tensorflow', 'java']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:3,name:"Dr. Praveen Borade",designation:"Associate Professor",department:"ME",age:40,salary:75000,specialization:['au
tocad', 'aerodynamics', 'thermal physics']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:4,name:"Dr. Madhav Nayak",designation:"Assistant Professor",department:"ME",age:37,salary:95000,specialization:['auto
cad', 'flight-dynamics', 'Finite Element Analysis']});
WriteResult({ "nInserted" : 1 })
> db.faculty.aggregate ( {$match:{department:"ME"}}, {$group : {_id : "$designation", AverageSal :{$avg:"$salary"} } },  {$match:{AverageSal:{
$gt:50000}}});
{ "_id" : "Associate Professor", "AverageSal" : 75000 }
{ "_id" : "Assistant Professor", "AverageSal" : 95000 }
> db.createCollection("product");
{ "ok" : 1 }
> db.product.insert({pid:1,pname:"keyboard",mdate:2001,price:1800,quantity:2});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:2,pname:"mouse",mdate:2005,price:1500,quantity:5});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:3,pname:"monitor",mdate:2015,price:10000,quantity:9});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:4,pname:"motherboard",mdate:2021,price:15000,quantity:4});
WriteResult({ "nInserted" : 1 })
> db.product.find({},{pname:1,_id:0})
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
```

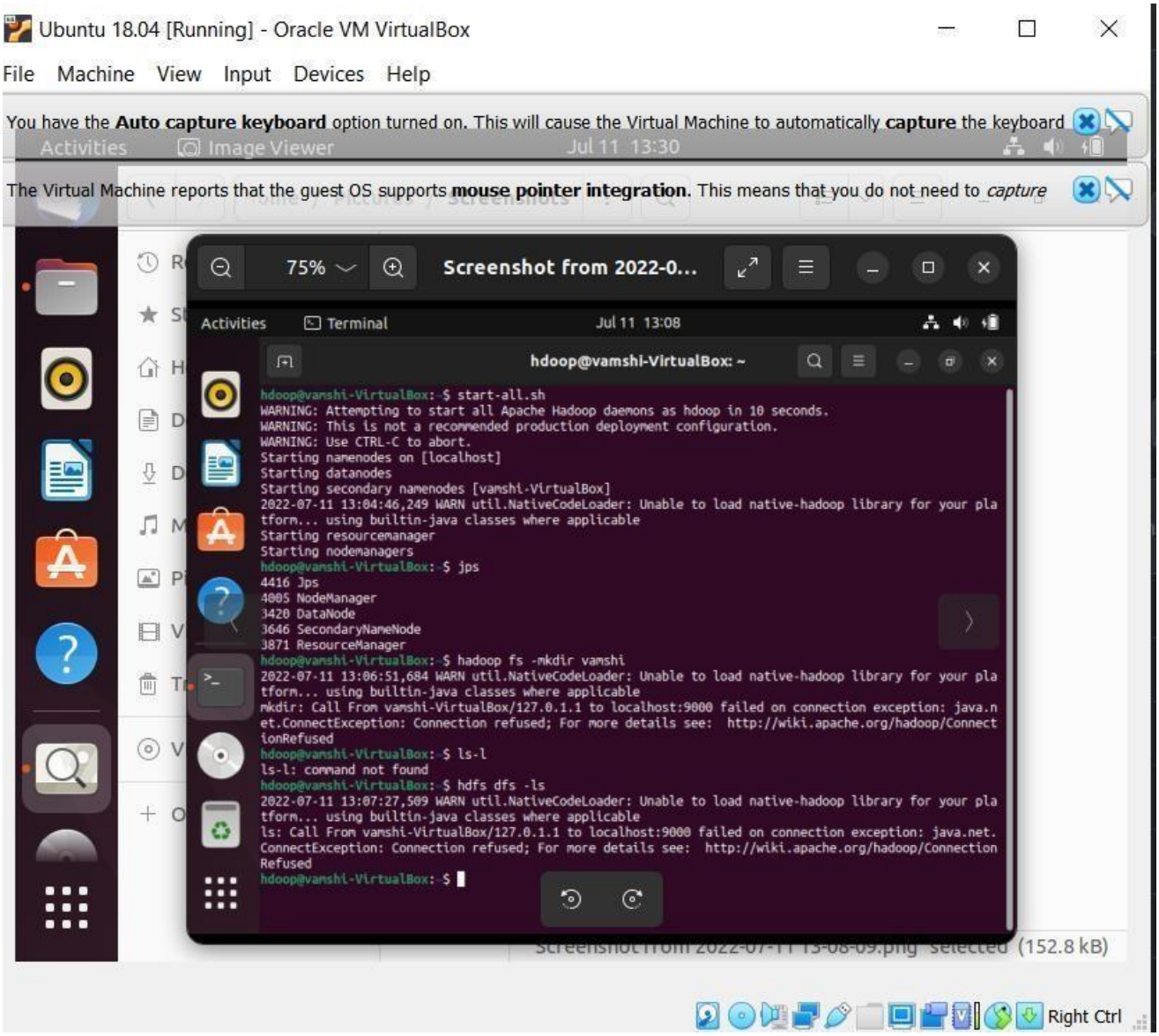
3)Create a mongodb collection Hospital. Demonstrate the following by choosing felds of choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors Updation

5.

```
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
> db.product.find({$and:[{quantity:{$eq:9}},{pname:{$eq:"monitor"}}]},{pname:1,_id:0})
{ "pname" : "monitor" }
> db.product.find({pname:/d$/},{pname:1,quantity:1,_id:0})
{ "pname" : "keyboard", "quantity" : 2 }
{ "pname" : "motherboard", "quantity" : 4 }
> db.createCollection("hospital");
{ "ok" : 1 }
> db.hospital.insert({_id:1, Name: "Anshuman Agarwal", age:23, diseases:["fever", "diarrhoea", "wheezing", "gastritis"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:2, Name: "Pinky Chaubey", age:35, diseases:["fever","nausea", "food infection", "indigestion", "kidney stones"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:3, Name: "Amresh Chowpati", age:63, diseases:["hyperglycemia", "diabetes mellitus", "food poisoning", "cold"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.updateMany({},{$pull:{diseases:"fever"}});
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
> db.hospital.updateOne({_id:1},{$pop:{diseases:-1}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.hospital.find({"diseases.2":"nausea"});
> db.hospital.find({"diseases.1":"nausea"});
> d.hospital.find({});
uncaught exception: ReferenceError: d is not defined :
@(shell):1:1
> db.hospital.find({});
{ "_id" : 1, "Name" : "Anshuman Agarwal", "age" : 23, "diseases" : [ "wheezing", "gastritis" ] }
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
{ "_id" : 3, "Name" : "Amresh Chowpati", "age" : 63, "diseases" : [ "hyperglycemia", "diabetes mellitus", "food poisoning", "cold" ] }
> db.hospital.find({"diseases.0":"nausea"});
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
> db.hospital.update({_id:3},{$set:{'diseases.1':'sarscov'}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Screenshot of Hadoop installed :



6. Create a Map Reduce program to

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

b) find the mean max temperature for every month

CODE:

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;

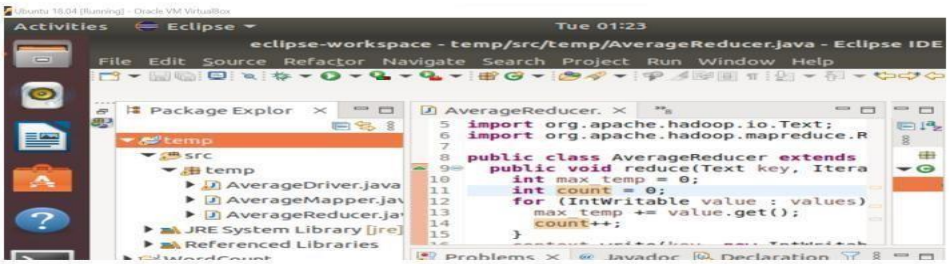
```
import java.io.IOException; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable; import
org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Mapper;

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
```

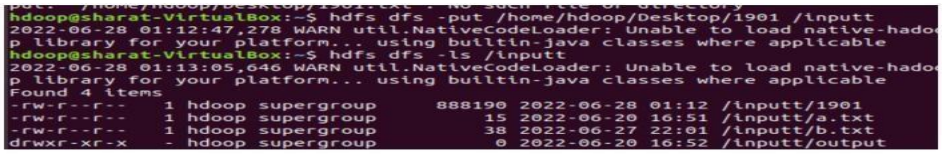
```
IntWritable(temperature)); }
}
```

AverageReducer

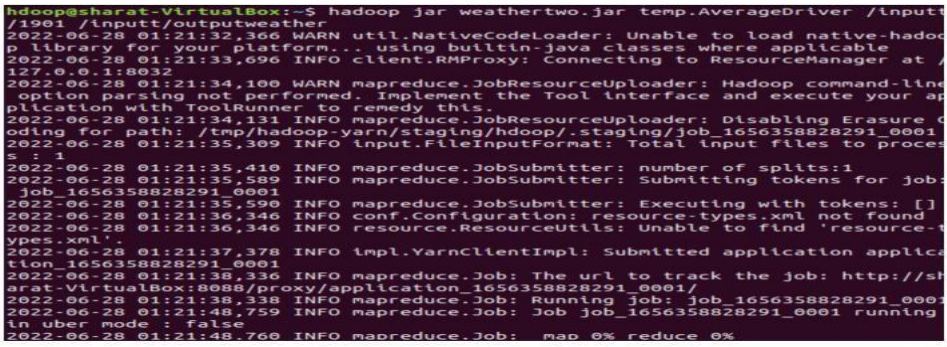
```
package temp;

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)); } }
```

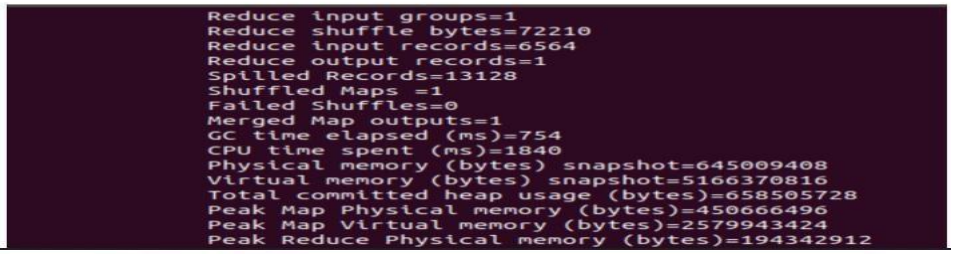
OUTPUT:





-





b)

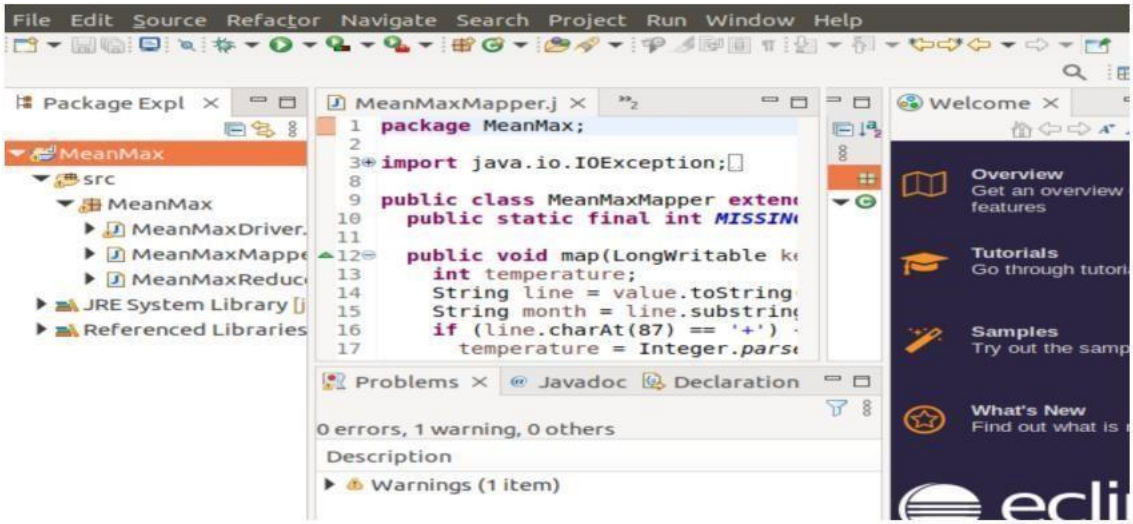
CODE:

MeanMaxDriver.class **package** meanmax;

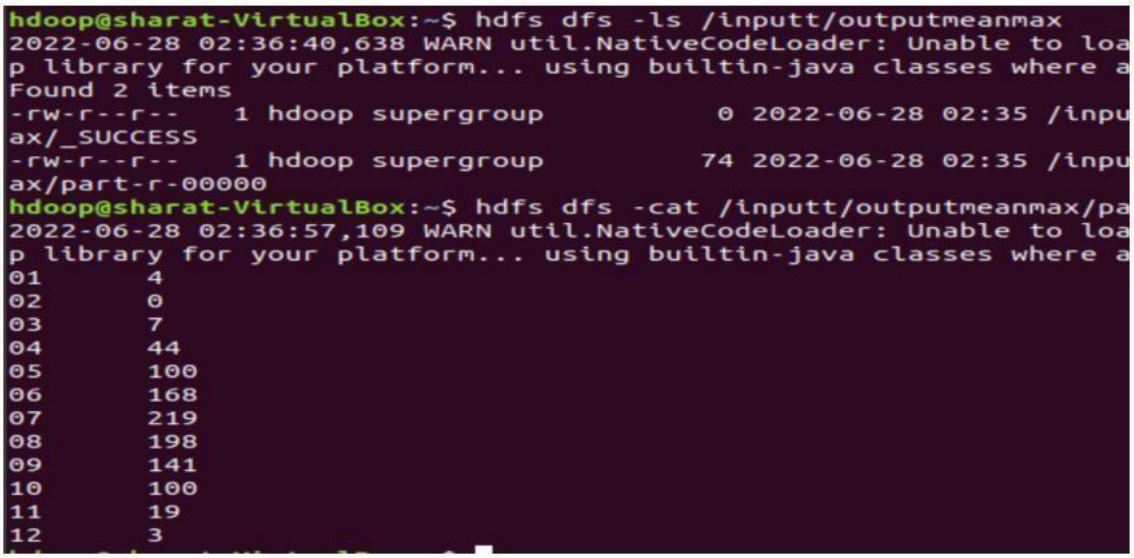
```
import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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)); } }
```

OUTPUT:




```
hdoop@sharat-VirtualBox:~$ hadoop jar MeanMaxweather2.jar MeanMax.MeanMaxDriver
 /inputt/1901 /inputt/outputmeanmax
2022-06-28 02:35:15,863 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
2022-06-28 02:35:16,403 INFO client.RMProxy: Connecting to ResourceManager at /
127.0.0.1:8032
2022-06-28 02:35:16,741 WARN mapreduce.JobResourceUploader: Hadoop command-line
 option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
2022-06-28 02:35:16,774 INFO mapreduce.JobResourceUploader: Disabling Erasure C
oding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1656363425892_0001
2022-06-28 02:35:17,464 INFO input.FileInputFormat: Total input files to proces
s : 1
2022-06-28 02:35:17,959 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-28 02:35:18,176 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1656363425892_0001
2022-06-28 02:35:18,177 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-28 02:35:18,417 INFO conf.Configuration: resource-types.xml not found
2022-06-28 02:35:18,418 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
2022-06-28 02:35:18,932 INFO impl.YarnClientImpl: Submitted application applica
tion 1656363425892_0001
```



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.

CODE:

Driver-TopN.class **package** samples.topn;

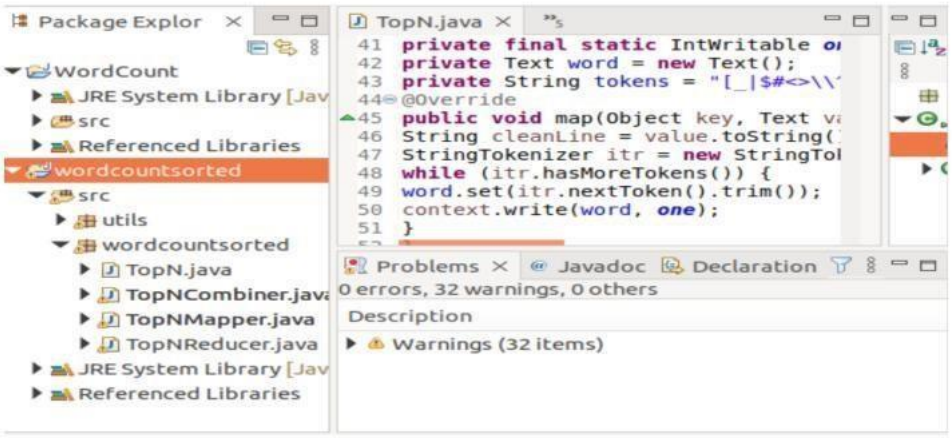
```
import java.io.IOException; import java.util.StringTokenizer; import
org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser;

public class TopN { public static void main(String[] args)
    throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs(); if
(otherArgs.length != 2) {
            System.err.println("Usage: TopN <in> <out>"); System.exit(2);
        }
        Job job = Job.getInstance(conf); job.setJobName("Top N"); job.setJarByClass(TopN.class);
        job.setMapperClass(TopNMapper.class); job.setReducerClass(TopNReducer.class);
        job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new
Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1); }

                        public static class TopNMapper extends Mapper<Object, Text,
Text, IntWritable> { private static final IntWritable one = new IntWritable(1); private Text word = new
        Text();

        private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-
:()?!\"']";

        public void map(Object key, Text value, Mapper<Object,
Text, Text, IntWritable>.Context context) throws IOException,
InterruptedException {
            String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " "); StringTokenizer itr =
new StringTokenizer(cleanLine); while (itr.hasMoreTokens()) {
            this.word.set(itr.nextToken().trim()); context.write(this.word, one); }
        }
    }
}
```

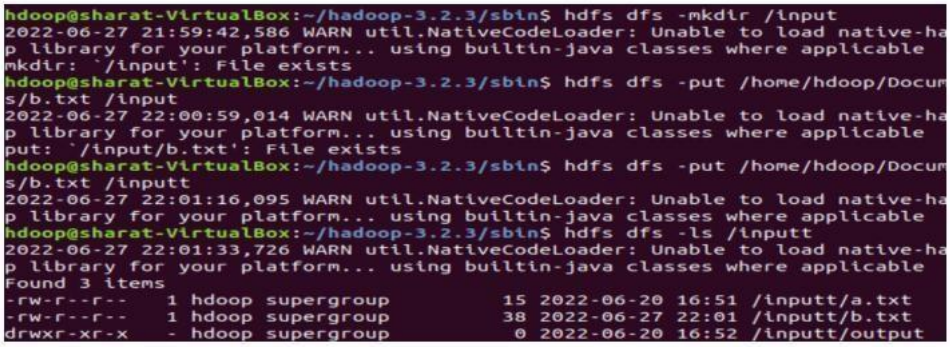
TopNCombiner.class **package** samples.topn;

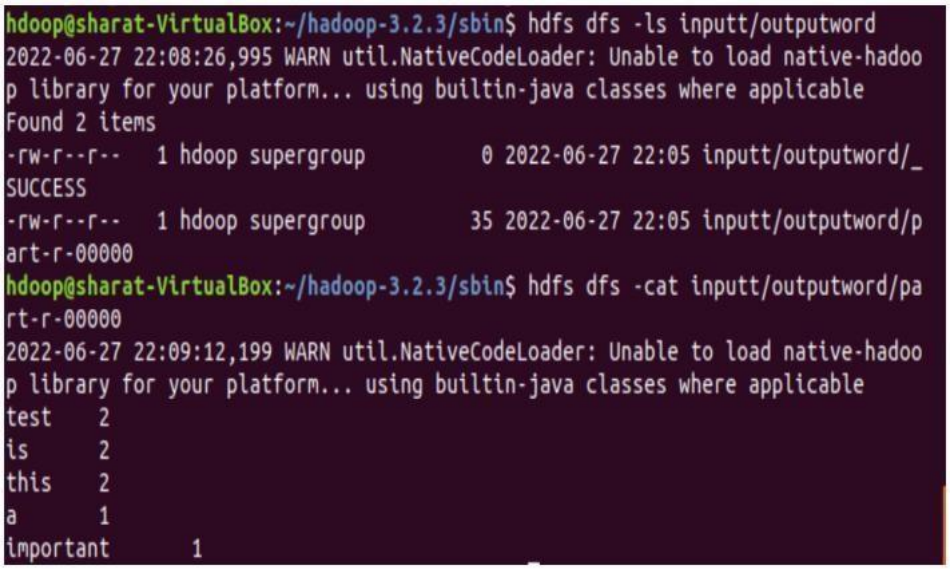
```
import java.io.IOException; import org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Reducer;

public class TopNCombiner extends Reducer<Text, IntWritable,
Text, IntWritable> { public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException { int sum = 0;
        for (IntWritable val : values) sum += val.get();
        context.write(key, new IntWritable(sum)); }
}



TopNMapper.class package samples.topn;

import java.io.IOException; import java.util.StringTokenizer; import
org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Mapper;

public class TopNMapper extends Mapper<Object, Text, Text,
IntWritable> { private static final IntWritable one = new IntWritable(1); private Text word = new Text();

    private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-
:()?!\"']";

    public vo```\\id map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)
throws IOException,
InterruptedException {
        String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " "); StringTokenizer itr =
new StringTokenizer(cleanLine); while (itr.hasMoreTokens()) { this.word.set(itr.nextToken().trim());
context.write(this.word, one);
        }
    }
}
TopNReducer.class package samples.topn;

import java.io.IOException; import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer; import utils.MiscUtils;

public class TopNReducer extends Reducer<Text, IntWritable,
Text, IntWritable> { private Map<Text, IntWritable> countMap = new HashMap<>();

    public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException { int sum = 0;
        for (IntWritable val : values) sum += val.get(); this.countMap.put(new Text(key),
        new IntWritable(sum));
```

```
    } protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws

    IOException,

InterruptedException {
        Map<Text, IntWritable> sortedMap =
MiscUtils.sortByValues(this.countMap); int counter = 0;
        for (Text key : sortedMap.keySet()) { if
        (counter++ == 20) break;
            context.write(key, sortedMap.get(key)); }
    }
}
```

OUTPUT:







8. Create a Map Reduce program to demonstrating join operation CODE:

// 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 Name input'");

Path AInputPath = new Path(args[0]);
Path BInputPath = new Path(args[1]);
Path outputPath = new Path(args[2]);

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

Posts.class);

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

User.class);

FileOutputFormat.setOutputPath(conf, outputPath); conf.setPartitionerClass(KeyPartitioner.class);
conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

conf.setMapOutputKeyClass(TextPair.class); conf.setReducerClass(JoinReducer.class);

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);
}
} }
```

OUTPUT:

```
hdoop@sharat-VirtualBox:~$ hdfs dfs -copyFromLocal DeptName.txt DeptStrength.tx
t /
2022-06-28 01:49:34,172 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
copyFromLocal: `DeptStrength.txt': No such file or directory
hdoop@sharat-VirtualBox:~$ hdfs dfs -copyFromLocal DeptName.txt DeptEmpStrength
.txt /
2022-06-28 01:50:03,670 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
copyFromLocal: `/DeptName.txt': File exists
hdoop@sharat-VirtualBox:~$ hdfs dfs -copyFromLocal DeptEmpStrength.txt /
2022-06-28 01:50:14,698 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
copyFromLocal: `/DeptEmpStrength.txt': File exists
```

```
hdoop@sharat-VirtualBox:~$ hadoop jar MapReduceJoin.jar /DeptEmpStrength.txt /D
eptName.txt /output_mapreducejoin
2022-06-28 01:54:22,260 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
2022-06-28 01:54:22,634 INFO client.RMProxy: Connecting to ResourceManager at /
127.0.0.1:8032
2022-06-28 01:54:22,756 INFO client.RMProxy: Connecting to ResourceManager at /
127.0.0.1:8032
2022-06-28 01:54:22,936 INFO mapreduce.JobResourceUploader: Disabling Erasure C
oding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1656358828291_0002
2022-06-28 01:54:23,108 INFO mapred.FileInputFormat: Total input files to proce
ss : 1
2022-06-28 01:54:23,121 INFO mapred.FileInputFormat: Total input files to proce
ss : 1
2022-06-28 01:54:23,607 INFO mapreduce.JobSubmitter: number of splits:4
2022-06-28 01:54:23,771 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1656358828291_0002
2022-06-28 01:54:23,772 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-28 01:54:23,909 INFO conf.Configuration: resource-types.xml not found
2022-06-28 01:54:23,909 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
2022-06-28 01:54:23,967 INFO impl.YarnClientImpl: Submitted application applica
tion_1656358828291_0002
```

```
                Bytes Written=85
hdoop@sharat-VirtualBox:~$ hdfs dfs -ls /outputoutput_mapreducejoin
2022-06-28 01:55:29,436 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
ls: `/outputoutput_mapreducejoin': No such file or directory
hdoop@sharat-VirtualBox:~$ hdfs dfs -ls /output_mapreducejoin
2022-06-28 01:55:36,422 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 hdoop supergroup          0 2022-06-28 01:54 /output_mapreducejo
in/_SUCCESS
-rw-r--r--   1 hdoop supergroup         85 2022-06-28 01:54 /output_mapreducejo
in/part-00000
hdoop@sharat-VirtualBox:~$ hdfs dfs -cat /output_mapreducejoin/part-00000
2022-06-28 01:56:01,106 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
A11     50              Finance
B12     100             HR
C13     250             Manufacturing
Dept_ID Total_Employee          Dept_Name
```

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

CODE:

```
package wordcount


import org.apache.spark.SparkConf import
org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD.rddToPairRDDFunctions


object WordCount {
def main(args: Array[String]) = { //Start the Spark context val conf = new
SparkConf().setAppName("WordCount").setMaster("local")
val sc = new SparkContext(conf) //Read some example file to a test RDD
val test =sc.textFile("input.txt")
test.flatMap { line => //for
each line line.split(" ") //split
the line in word by word.
  } .map { word =>
//for each word
(word, 1) //Return a key/value tuple, with the word as key and 1 as
value .reduceByKey(_ + _) //Sum
all of the value with same
key .saveAsTextFile("output.txt") //Save to
a text file //Stop the Spark context
sc.stop
}
}
```

OUTPUT:

```
scala> val test=sc.textFile("/home/hdoop/spark_word_count.txt")
test: org.apache.spark.rdd.RDD[String] = /home/hdoop/spark_word_count.txt Map
rtitionsRDD[1] at textFile at <console>:23

scala> test.collect;
[Stage 0:>                                                          (0 + 0) /
[Stage 0:>                                                          (0 + 2) /

res4: Array[String] = Array(This is a test, This is an evaluation, do you wan
a test, why do you want a test)

scala> val count=test.flatMap(line=>line.split(" "))
count: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at
nsole>:23

scala> count.collect
res5: Array[String] = Array(This, is, a, test, This, is, an, evaluation, do,
u, want, a, test, why, do, you, want, a, test)

scala> val map_frequency=count.map(entry=>(entry,1))
map_frequency: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3]
 map at <console>:23

scala> map_frequency.collect
res6: Array[(String, Int)] = Array((This,1), (is,1), (a,1), (test,1), (This,
 (is,1), (an,1), (evaluation,1), (do,1), (you,1), (want,1), (a,1), (test,1),
hy,1), (do,1), (you,1), (want,1), (a,1), (test,1))
```

```
scala> map_frequency.reduceByKey(_+_)
res7: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey
t <console>:24

scala> map_frequency.collect
res8: Array[(String, Int)] = Array((This,1), (is,1), (a,1), (test,1), (This,1
 (is,1), (an,1), (evaluation,1), (do,1), (you,1), (want,1), (a,1), (test,1),
hy,1), (do,1), (you,1), (want,1), (a,1), (test,1))

scala>

scala> val final_output=map_frequency.reduceByKey(_+_)
final_output: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[5] at red
eByKey at <console>:23

scala> final_output.collect
[Stage 4:>                                                          (0 + 2) /

res9: Array[(String, Int)] = Array((is,2), (evaluation,1), (This,2), (why,1),
want,2), (test,3), (you,2), (a,3), (do,2), (an,1))
```

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

CODE:

```
val textFile = sc.textFile("/home/Desktop/test.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()
    }
}
```

OUTPUT: