VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

M Vamshi Krishna (1BM19CS080)

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 **M Vamshi Krishna(1BM19CS080),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (20CS6PEBDA)** work prescribed for the said degree.

Dr.Pallavi G B
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Index Sheet

Course Outcome

CO 1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO 2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO 3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

BDA LAB 1

```
bmsce@bmsce-OptiPlex-3060:~$ mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c2e3109b-0341-483b-ba3a-f9fb3b1aed87") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten]
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten]
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> db;
test
> use lab1DB;
switched to db lab1DB
> db;
lab1DB
```

```
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
myDB    0.000GB
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.drop();
true
> db.getCollectionNames()
[ ]
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.insert({_id:1, StudName:"Jeevan", Grade:"VI",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2, StudName:"Vamsi", Grade:"VI", Hobbies:["Watching Movies", "Reading Novels", "Drugs"]})
WriteResult({ "nInserted" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
```

```
> db.Student.find({}).pretty();
{
        "_id" : 1,
        "StudName" : "Jeevan",
        "Grade" : "VI",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 2,
        "StudName" : "Vamsi",
        "Grade" : "VI",
        "Hobbies" : [
                "Watching Movies",
                "Reading Novels",
                "Drugs"
        ]
}
> db.Student.insert({_id:3, StudName:"Sharat", Grade:"V", Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
```

```
> db.Student.update({_id:5, StudName:"Kusum", Grade:"VI"}, {$set:{Hobbies: ["Golf", "Sea Shell Collection"]}}, {upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 5 })
> db.Student.find({}).pretty();
{
        "_id" : 1,
        "StudName" : "Jeevan",
        "Grade" : "VI",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 2,
        "StudName" : "Vamsi",
        "Grade" : "VI",
        "Hobbies" : [
                "Watching Movies",
                "Reading Novels",
                "Drugs"
        ]
}
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ]
}
> db.Student.find({StudName:"Kusum"});
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
```

```
> db.Student.find({StudName:"Kusum"}, {StudName:1, Grade:1}).pretty();
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum" }
> db.Student.count();
4
> db.Student.find({}).sort({StudName:1});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
> db.Student.find({}).sort({StudName:-1});
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
> db.Student.update({_id:5}, {$set:{Location:"NIHMANS"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({StudName:"Kusum"}).pretty();
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
```

```
> db.Student.find({}).skip(2).pretty();
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
> db.Student.find().skip(2).pretty();
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
> db.createCollection("food")
{ "ok" : 1 }
> db.food.insert({_id:1,fruits:['avacado','dragon fruit']})
WriteResult({ "nInserted" : 1 })
```

```
> db.food.insert({_id:1,fruits:['avacado','dragon fruit']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['strawberry','dragon fruit']})
WriteResult({ "nInserted" : 1 })
> db.food.find({'fruits.1':'avacado'}).pretty()
> db.food.find().pretty()
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({'fruits.1':"avacado"}).pretty()
> db.food.find({'fruits.1':"avacado"})
> db.food.find({'fruits.0':"avacado"})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':"avacado"}).pretty()
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':"avacado"}).pretty();
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':{$size:2}}).pretty();
> db.food.find({'fruits':{$size:2}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({_id:2},{'fruits':{$slice:2}});
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({_id:2},{'fruits':{$slice:1}});
{ "_id" : 2, "fruits" : [ "strawberry" ] }
> db.food.find({fruits:{$all:["avacado"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({fruits:{$all:["avacado","dragon fruit"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({fruits:{$all:["dragon fruit"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
```

```
> show collections;
Student
customer
food
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}})
{ "_id" : 2, "TotalAccBal" : 20000000 }
{ "_id" : 1, "TotalAccBal" : 10000000 }
> db.customer.find()
{ "_id" : ObjectId("6253f945d7ce1043c6d5c8cc"), "custID" : 1, "AcctBal" : 10000000, "AcctType" : "FD" }
{ "_id" : ObjectId("6253f963d7ce1043c6d5c8cd"), "custID" : 2, "AcctBal" : 20000000, "AcctType" : "FD" }
{ "_id" : ObjectId("6253f973d7ce1043c6d5c8ce"), "custID" : 3, "AcctBal" : 30000000, "AcctType" : "RD" }
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}},{$match:{TotAccBal:{$gt:10000000}}});
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}},{$match:{TotalAccBal:{$gt:10000000}}});
{ "_id" : 2, "TotalAccBal" : 20000000 }
> quit()
```

BDA- Lab 2

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 filter 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 file “Output.txt”.

```
bmsce@bmsce-Precision-T1700:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("4419b91e-5b22-4f43-a52c-ac40a0bf73a6") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-04-20T19:31:53.425+0530 I STORAGE  [initandlisten]
2022-04-20T19:31:53.426+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-04-20T19:31:53.426+0530 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten]
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten]
> use Niharika_db
switched to db Niharika_db
```

```
> db.Student.insert({_id:1,Name:"Aravind",USN:"1BM19CS001",Sem:6,Dept_name:"CSE",CGPA:"9.6",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name:"Aman",USN:"1BM19EC002",Sem:7,Dept_name:"ECE",CGPA:"9.1",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,Name:"Latha",USN:"1BM19CS003",Sem:6,Dept_name:"CSE",CGPA:"8.1",Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,Name:"Sam",USN:"1BM19CS004",Sem:6,Dept_name:"CSE",CGPA:"6.5",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,Name:"Suman",USN:"1BM19CS005",Sem:5,Dept_name:"CSE",CGPA:"7.6",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.update({_id:1},{$set:{CGPA:9.0}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:2},{$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},{$set:{CGPA:8.6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.aggregate({$match:{Dept_name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
{ "_id" : 5, "AvgCGPA" : 8.6 }
{ "_id" : 6, "AvgCGPA" : 7.866666666666667 }
bmsce@bmsce-Precision-T1700:~$ mongoexport --host localhost --db Niharika_db --collection Student --csv --out /home/bmsce/Desktop/output.txt --fields "_id","Name","USN","Sem","Dept-name","CGPA","Hobbies";
2022-04-20T15:04:30.836+0530    csv flag is deprecated; please use --type=csv instead
2022-04-20T15:04:30.836+0530    connected to: localhost
2022-04-20T15:04:30.836+0530    exported 5 records
```

```
Open
output.txt
~/Desktop
_id,Name,USN,Sem,Dept-name,CGPA,Hobbies
1,Aravind,1BM19CS001,6,,9,Badminton
2,Aman,1BM19EC002,7,,9.1,Swimming
3,Latha,1BM19CS003,6,,8.1,Reading
4,Sam,1BM19CS004,6,,6.5,Cycling
5,Suman,1BM19CS005,5,,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

```
> 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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
```

```
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
> db.Bank.updateMany({},{$pull:{Contact:"080-25639856"}} );
{ "acknowledged" : true, "matchedCount" : 5, "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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> db.Bank.createIndex({Name:1, Type:1},{name:});
uncaught exception: SyntaxError: expected expression, got '}' :
@(shell):1:43
> db.Bank.createIndex({Name:1, Type:1},{name:"Find current account holders"});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> 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.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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5 }
> db.Bank.update({_id:"625d78659329139694f188a6", CustID:5}, {$set: {Name:"Sumantha K S", Type:"Savings", Contact:["9856321478","011-65897458"
]}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 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", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5, "Contact" : [ "9856321478", "011-65897458" ], "Name" : "Sumantha K S", "Type" : "Savings"
}
>
```

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 filter 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 find those documents where the price is not set to 15000.
4) To find those documents from the Product collection where the quantity is set to 9 and theproduct name is set to ‘monitor’.
5) To find 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 fields of yourchoice.

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

BDA LAB 3

Program 1. Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

```
cqlsh> CREATE KEYSPACE Employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> describe keyspace
No keyspace specified and no current keyspace
cqlsh> describe Employee;
```

2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

```
cqlsh> create table Employee.Employee_Info(Emp_Id int Primary Key,Emp_Name text,Designation text,Date_of_Joining timestamp,Salary dou
ble,Dept_Name text);
```

```
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining | dept_name | designation | emp_name | salary
--------+-----------------+-----------+-------------+----------+--------

(0 rows)
```

3. Insert the values into the table in batch

```
cqlsh> begin batch insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(1,'2021-06-
03','Deployment','Manager','Kusum',1500000.50); apply batch;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name  | designation | emp_name | salary
--------+---------------------------------+------------+-------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 | Deployment |     Manager |    Kusum | 1.5e+06

(1 rows)
```

```
cqlsh> begin batch
   ... insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(2,'2020-09-03','Develop
ment','Web developer','Karan',1700000.50);
   ... insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(121,'2019-05-03','R&D',
'Intern','Kia',2000000.50);
   ... apply batch;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation   | emp_name | salary
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |       Manager |    Kusum | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |         R&D |        Intern |      Kia |   2e+06

(3 rows)
```

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

```
cqlsh> update Employee.Employee_Info SET emp_name='Kushi',dept_name='Testing' where emp_id=121;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | salary
--------+---------------------------------+-------------+----------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |   2e+06

(3 rows)
```

5. Sort the details of Employee records based on salary

```
cqlsh> create table Employee.emp(Emp_Id int,Emp_name text,Designation text,Date_Of_Joining timestamp,Salary double,Dept_Name text,primary ke
y(Emp_Id,Salary));
cqlsh> begin batch
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(1,1500000.50,'2021-06-03','Deployment','
Manager','Kusum');
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(2,1100000.50,'2022-05-03','Development',
'Web Developer','Karan');
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(121,1900000.50,'2022-05-03','R&D','Inter
n','Kia');
   ... apply batch;
cqlsh> select * from Employee.emp;

 emp_id | salary  | date_of_joining                 | dept_name   | designation    | emp_name
--------+---------+---------------------------------+-------------+----------------+----------
      1 | 1.5e+06 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum
      2 | 1.1e+06 | 2022-05-03 00:00:00.000000+0000 | Development | Web Developer |    Karan
    121 | 1.9e+06 | 2022-05-03 00:00:00.000000+0000 |         R&D |         Intern |      Kia

(3 rows)
cqlsh> paging off;
Disabled Query paging.
cqlsh> select * from Employee.emp where emp_id in (1,2,121) order by salary;

 emp_id | salary  | date_of_joining                 | dept_name   | designation    | emp_name
--------+---------+---------------------------------+-------------+----------------+----------
      2 | 1.1e+06 | 2022-05-03 00:00:00.000000+0000 | Development | Web Developer |    Karan
      1 | 1.5e+06 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum
    121 | 1.9e+06 | 2022-05-03 00:00:00.000000+0000 |         R&D |         Intern |      Kia

(3 rows)
cqlsh>
```

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

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects | salary
--------+---------------------------------+-------------+----------------+----------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |     null | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan |     null | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |     null |   2e+06

(3 rows)
```

7. Update the altered table to add project names.

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum | {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan |           null | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |           null |   2e+06

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                        | salary
--------+---------------------------------+-------------+----------------+----------+---------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                            null |   2e+06

(3 rows)

cqlsh> update Employee.Employee_Info set projects=projects+{'lab','jkl'} where emp_id=121;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                        | salary
--------+---------------------------------+-------------+----------------+----------+---------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                {'lab', 'jkl'} |   2e+06

(3 rows)
```

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

```
cqlsh> insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(11,'2019-05-05','R&D','
Intern','Kajal',1000000.50) using TTL 15;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                        | salary
--------+---------------------------------+-------------+----------------+----------+---------------------------------+---------
     11 | 2019-05-05 00:00:00.000000+0000 |         R&D |         Intern |    Kajal |                            null |   1e+06
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                {'lab', 'jkl'} |   2e+06

(4 rows)
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                        | salary
--------+---------------------------------+-------------+----------------+----------+---------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                {'lab', 'jkl'} |   2e+06

(3 rows)
cqlsh>
```

LAB-4

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':3};
cqlsh> Use LIBRARY;
cqlsh:library>
```

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

```
cqlsh:library> create table library_info(stud_id int, counter_value Counter, stud_name text,book_name text, date_of_issue timestamp, book_id int, PRIMARY KEY(stud_id,stud_name,book_name,date_of
_issue,book_id));
```

```
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue | book_id | counter_value
---------+-----------+-----------+---------------+---------+---------------

(0 rows)
```

3. Insert the values into the table in batch

```
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 111 and stud_name = 'SAM' and book_name = 'ML' and date_of_issue = '2020-10-11'and book_id = 200;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21'and book_id = 300
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 113 and stud_name = 'AYMAN' and book_name = 'OOMD' and date_of_issue = '2020-04-01'and book_id = 400;
```

```
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |       SAM |        ML | 2020-10-10 18:30:00.000000+0000 |     200 |             1
     113 |     AYMAN |      OOMD | 2020-03-31 18:30:00.000000+0000 |     400 |             1
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             1

(3 rows)
```

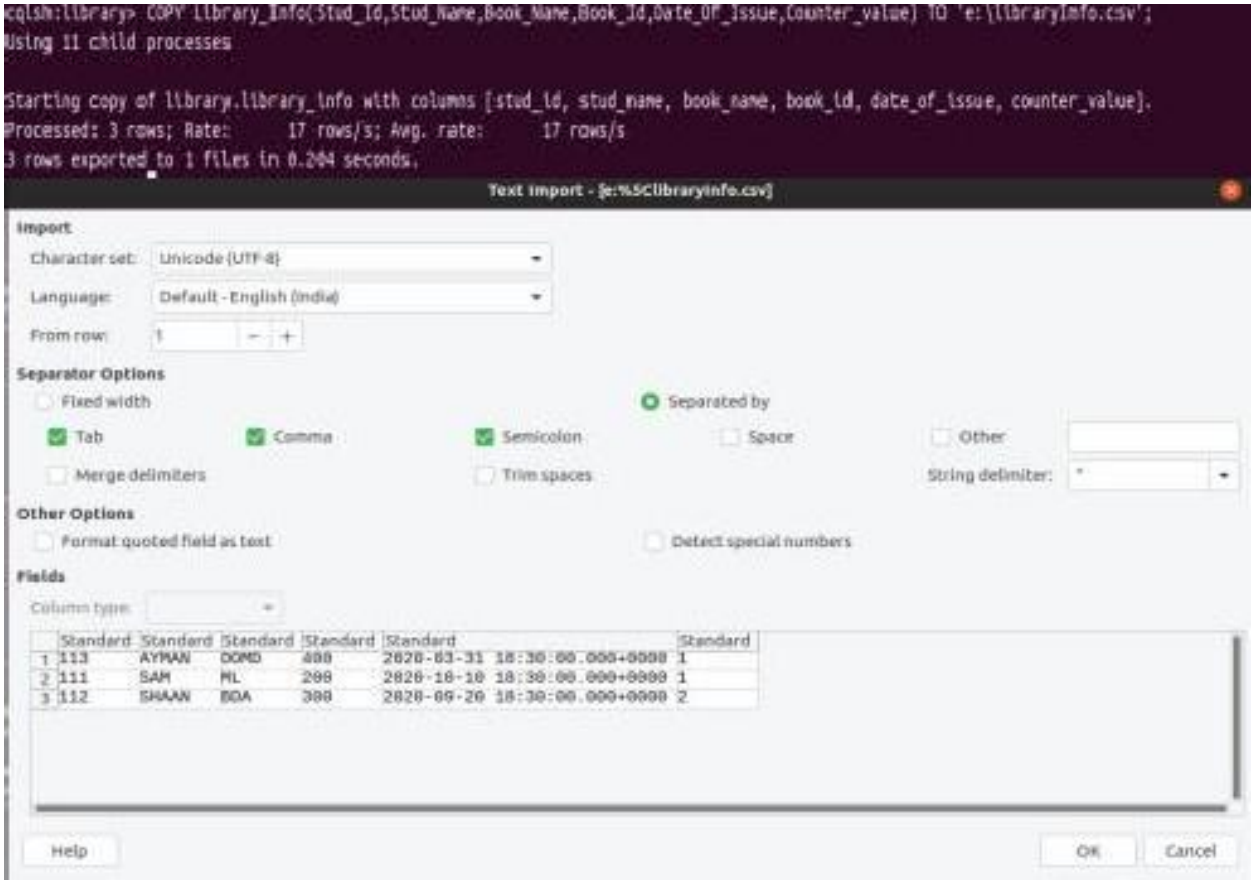
4. Display the details of the table created and increase the value of the counter

```
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21'and book_id = 300;
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |       SAM |        ML | 2020-10-10 18:30:00.000000+0000 |     200 |             1
     113 |     AYMAN |      OOMD | 2020-03-31 18:30:00.000000+0000 |     400 |             1
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             2

(3 rows)
```

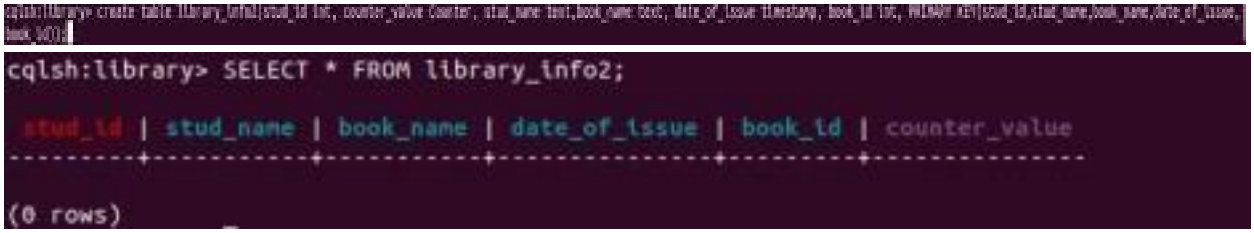
5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

```
cqlsh:library> SELECT * FROM library_info WHERE stud_id = 112;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             2

(1 rows)
```

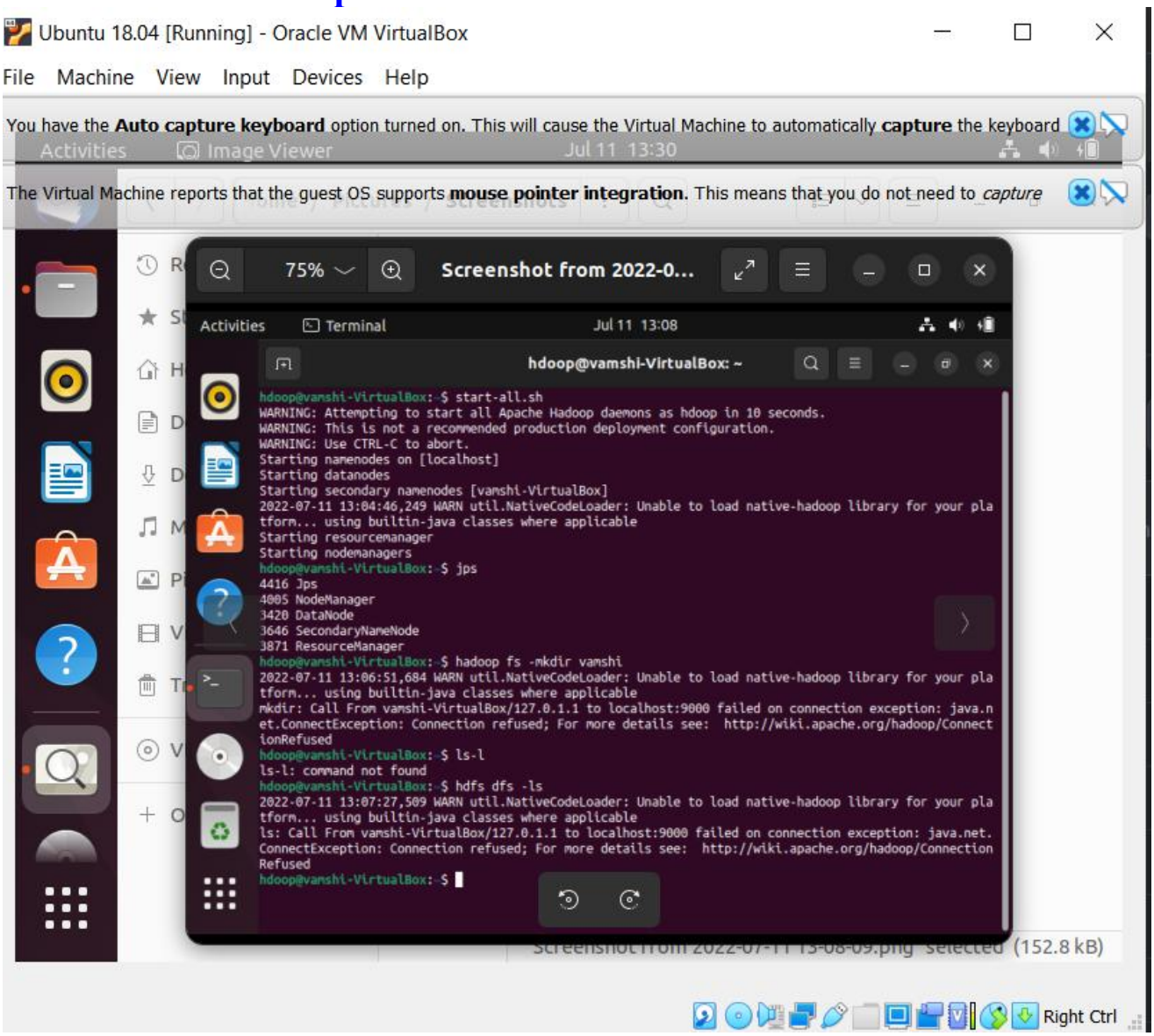
6. Export the created column to a csv file



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



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

a)

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;
```

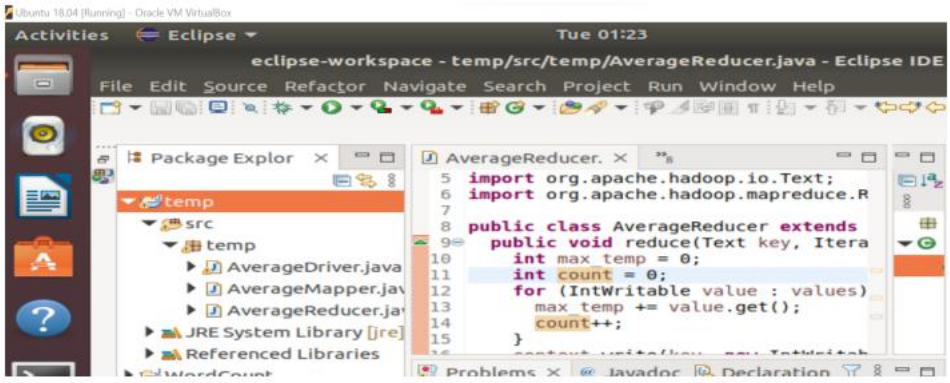
```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

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:



```
hdoop@sharat-VirtualBox:~$ hadoop jar weathertwo.jar temp.AverageDriver /inputt
/1901 /inputt/outputweather
2022-06-28 01:21:32,366 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
2022-06-28 01:21:33,696 INFO client.RMProxy: Connecting to ResourceManager at /
127.0.0.1:8032
2022-06-28 01:21:34,100 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 01:21:34,131 INFO mapreduce.JobResourceUploader: Disabling Erasure C
oding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1656358828291_0001
2022-06-28 01:21:35,309 INFO input.FileInputFormat: Total input files to proces
s : 1
2022-06-28 01:21:35,410 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-28 01:21:35,589 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1656358828291_0001
2022-06-28 01:21:35,590 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-28 01:21:36,346 INFO conf.Configuration: resource-types.xml not found
2022-06-28 01:21:36,346 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
2022-06-28 01:21:37,378 INFO impl.YarnClientImpl: Submitted application applica
tion_1656358828291_0001
2022-06-28 01:21:38,336 INFO mapreduce.Job: The url to track the job: http://sh
arat-VirtualBox:8088/proxy/application_1656358828291_0001/
2022-06-28 01:21:38,338 INFO mapreduce.Job: Running job: job_1656358828291_0001
2022-06-28 01:21:48,759 INFO mapreduce.Job: Job job_1656358828291_0001 running
in uber mode : false
2022-06-28 01:21:48,760 INFO mapreduce.Job:  map 0% reduce 0%
```

```
                Reduce input groups=1
                Reduce shuffle bytes=72210
                Reduce input records=6564
                Reduce output records=1
                Spilled Records=13128
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=754
                CPU time spent (ms)=1840
                Physical memory (bytes) snapshot=645009408
                Virtual memory (bytes) snapshot=5166370816
                Total committed heap usage (bytes)=658505728
                Peak Map Physical memory (bytes)=450666496
                Peak Map Virtual memory (bytes)=2579943424
                Peak Reduce Physical memory (bytes)=194342912
```

```
                Bytes Written=8
hdoop@sharat-VirtualBox:~$ hdfs dfs -ls  /inputt/outputweather
2022-06-28 01:22:16,506 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 01:21 /inputt/outputweath
er/_SUCCESS
-rw-r--r--   1 hdoop supergroup          8 2022-06-28 01:21 /inputt/outputweath
```

```
hdoop@sharat-VirtualBox:~$ hdfs dfs -cat /inputt/outputweather/part-r-00000
2022-06-28 01:23:07,585 WARN util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
1901    46
```

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

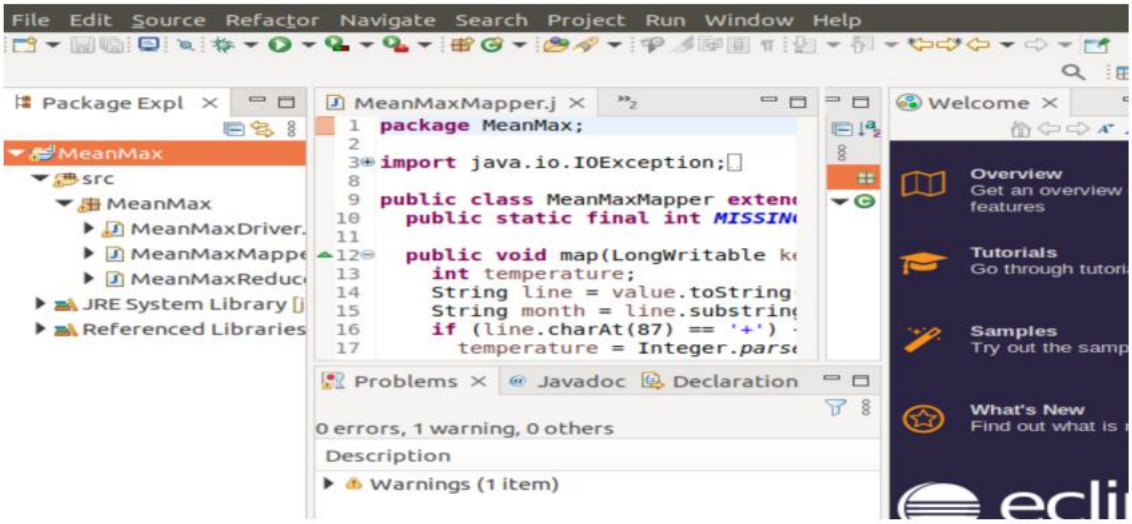
```
package meanmax;

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));
  }
}
```

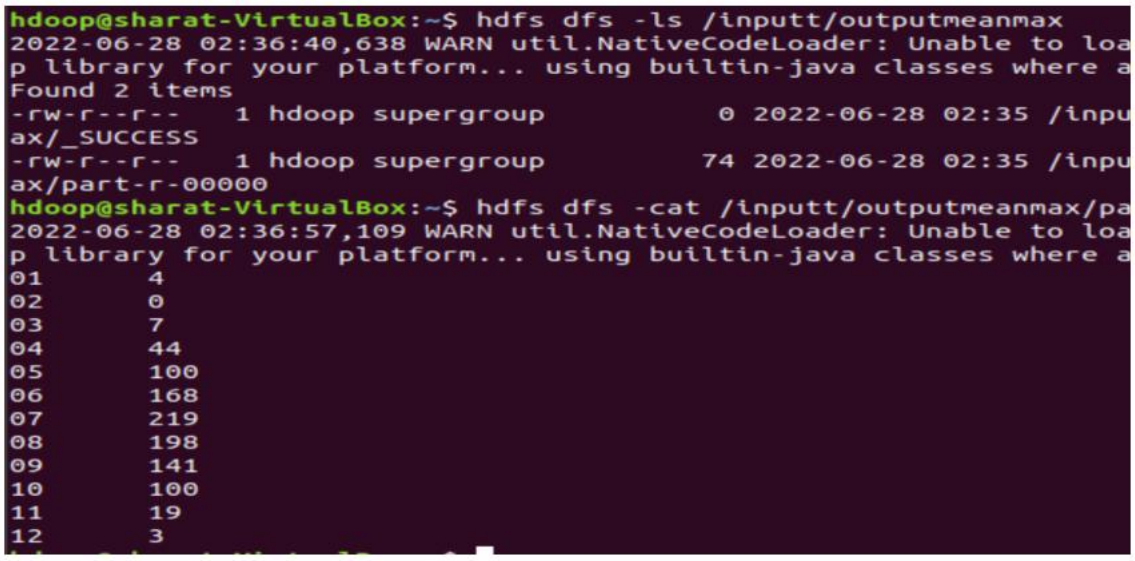
MeanMaxReducer.class

```
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();
```

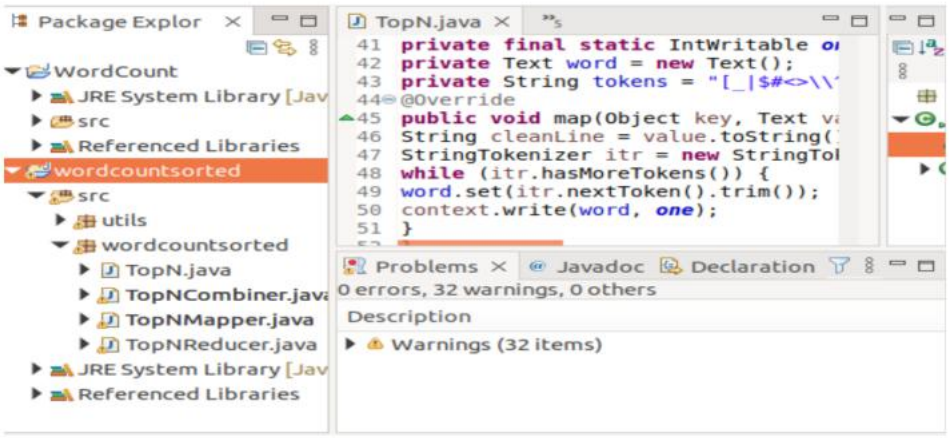
```
  private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-
:()?!\"']";

  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);
    }
  }
}
```

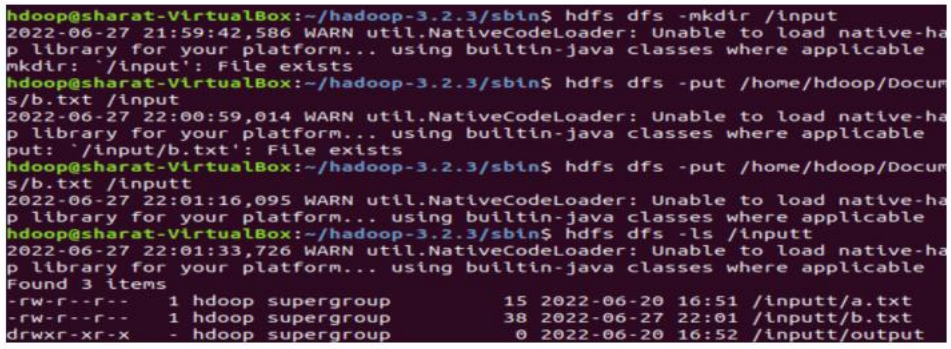
TopNReducer.class

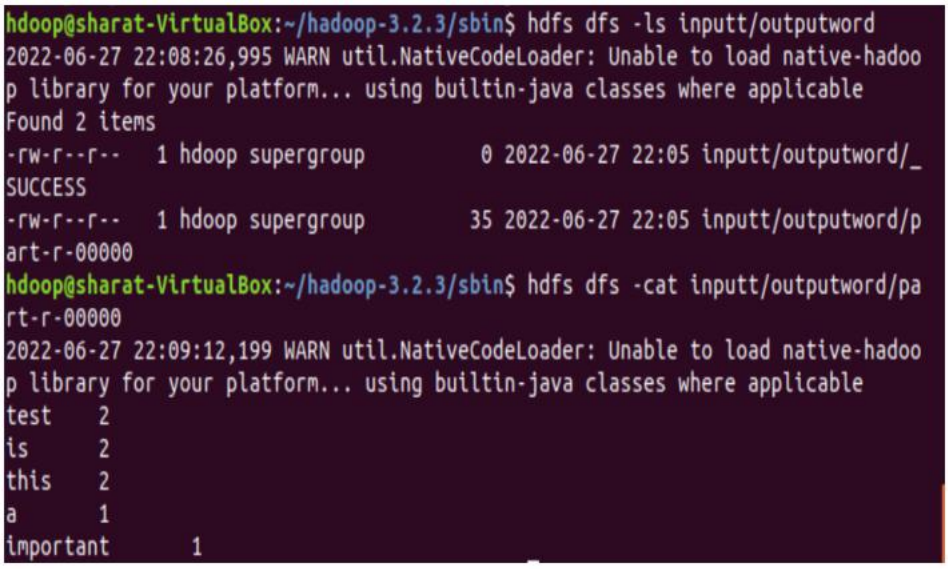
```
package samples.topn;

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
```

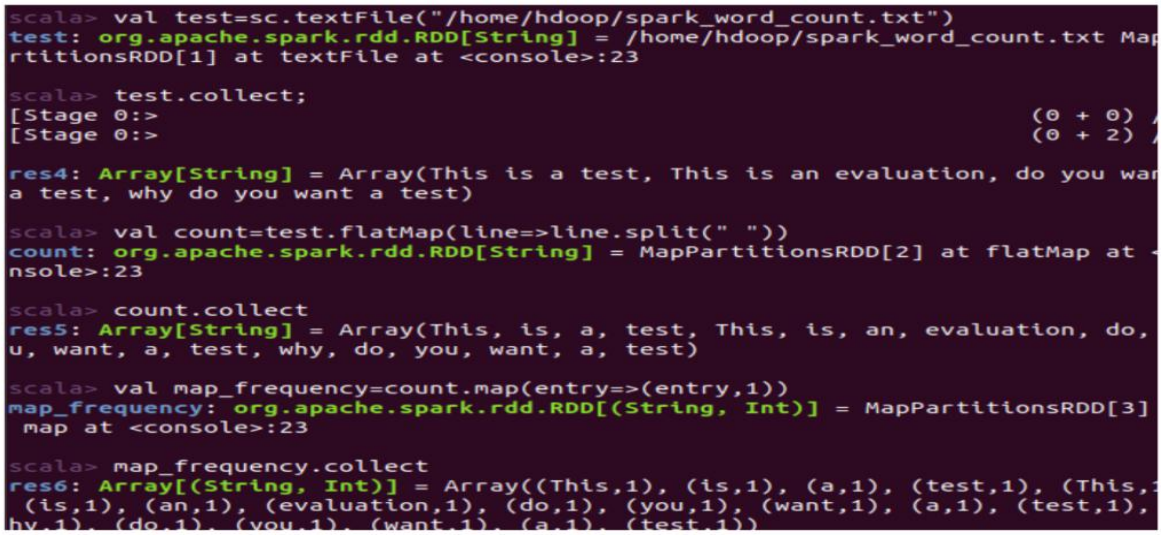
9. Program to print word count on scala shell and print “Hello world” on scala IDE

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> 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:

```
sorted: scala.collection.immutable.ListMap[String,Int] = ListMa
> 3, is -> 2, This -> 2, want -> 2, do -> 2, why -> 1, you -> 1

scala> for((k,v)<-sorted)
     | {
     | if(v>4)
     | {
     | print(k+",")
     | print(v)
     | println()
     | }
     | }
test,5
```

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

scala> val frequency=word_count.flatMap((line)=>line.split(" ")).map(word=>(w
d,1)).reduceByKey(_+_)
frequency: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduce
Key at <console>:23

scala> val sorted=ListMap(frequency.collect.sortWith(_._2>_._2):_*)
<console>:23: error: not found: value ListMap
       val sorted=ListMap(frequency.collect.sortWith(_._2>_._2):_*)
                  ^

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

scala> val sorted=ListMap(frequency.collect.sortWith(_._2>_._2):_*)
[Stage 0:>                                                          (0 + 2)

sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(test -> 5,
> 3, is -> 2, This -> 2, want -> 2, do -> 2, why -> 1, you -> 1, an -> 1)
```


```
sorted: scala.collection.immutable.ListMap[String,Int] = ListMa
> 3, is -> 2, This -> 2, want -> 2, do -> 2, why -> 1, you -> 1

scala> for((k,v)<-sorted)
     | {
     | if(v>4)
     | {
     | print(k+",")
     | print(v)
     | println()
     | }
     | }
test,5
```