**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

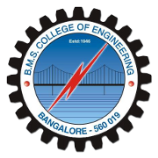
**KANALA BHUVANA REDDY (1BM19CS069)**

***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 **KANALA BHUVANA REDDY (1BM19CS069),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Course Title - (Course code)** work prescribed for the said degree.

Name of the Lab-Incharge                **Antara Roy Choudhury**

Designation Assistant Professor

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

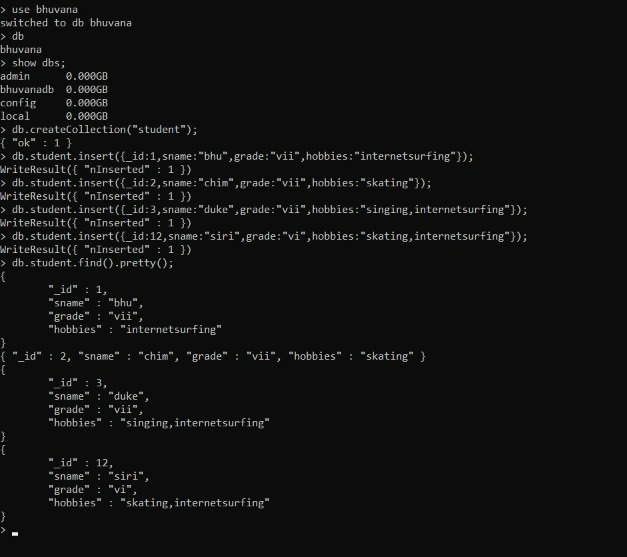
|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | **MongoDB CRUD Demonstration** | **4** |
| **2** | **Cassandra-Employee Database** | **10** |
| **3** | **Cassandra-Library Database** | **17** |
| **4** | **Screenshots of Hadoop installed** | **24** |
| **5** | **Execution of hdfs commands** | **25** |
| **6** | **Map Reduce for weather data** | **27** |
| **7** | **Map Reduce for top 10 maximum occurrences of words** | **30** |
| **8** | **Map Reduce to demonstrating join operation** | **32** |
| **9** | **WordCount on scala and “Hello World” on scala IDE** | **35** |
| **10** | **Use of RDD and FlatMap** | **36** |

**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 |

### Experiment 1

1

****



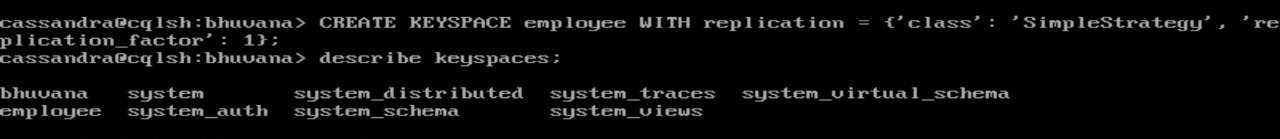
|  |
| --- |
|  |
| > db.Student.find({Grade:{$eq:'vii'}}); |
|  | { "\_id" : 1, "StudName" : "Megha", "Grade" : "vii", "Hobbies" : "InternetSurfing" } |
|  | { "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" } |
|  | > db.Student.find({Grade:{$eq:'vii'}}).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | { "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" } |
|  | > db.Student.find({Hobbies:{$in:['Chess','Skating']}}).pretty(); |
|  | > db.Student.find({Hobbies:{$in:['Skating']}}).pretty(); |
|  | > db.Student.find({Hobbies:{$in:['skating']}}).pretty(); |
|  | { "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" } |
|  | > db.Student.find({StudName:/^M/}).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | > db.Student.find({StudName:/e/}).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | > db.Student.count(); |
|  | 2 |
|  | > db.Student.find().sort({StudName:-1}).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | { "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" } |
|  | > db.Student.save({StudName:"Vamsi",Greade:"vi"}) |
|  | WriteResult({ "nInserted" : 1 }) |
|  | > db.Students.update({\_id:4},{$set:{Location:"Network"}}) |
|  | WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 }) |
|  | > db.Students.update({\_id:4},{$unset:{Location:"Network"}}) |
|  | WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 }) |
|  | > db.Student.find({\_id:1},{StudName:1,Grade:1,\_id:0}); |
|  | { "StudName" : "Megha", "Grade" : "vii" } |
|  | > db.Student.find({Grade:{$ne:'VII'}}).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | { "\_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" } |
|  | { |
|  | "\_id" : ObjectId("6253f413e88b8c9e787b194e"), |
|  | "StudName" : "Vamsi", |
|  | "Greade" : "vi" |
|  | } |
|  | > db.Student.find({StudName:/s$/}).pretty(); |
|  | > db.Students.update({\_id:3},{$set:{Location:null}}) |
|  | WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 }) |
|  | > db.Students.count() |
|  | 0 |
|  | > db.Students.count({Grade:"VII"}) |
|  | 0 |
|  | > db.Student.find({Grade:"VII"}).limit(3).pretty(); |
|  | > db.Student.update({\_id:3},{$set:{Location:null}}) |
|  | WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) |
|  | > db.Student.count({Grade:"VII"}) |
|  | 0 |
|  | > db.Students.count({Grade:"vii"}) |
|  | 0 |
|  | > db.Student.count() |
|  | 3 |
|  | > db.Student.count({Grade:"vii"}) |
|  | 2 |
|  | > db.Student.find({Grade:"vii"}).limit(3).pretty(); |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | { |
|  | "\_id" : 3, |
|  | "Grade" : "vii", |
|  | "StudName" : "Ayan", |
|  | "Hobbies" : "skating", |
|  | "Location" : null |
|  | } |
|  | > db.Student.find().sort({StudName:1}).pretty(); |
|  | { |
|  | "\_id" : 3, |
|  | "Grade" : "vii", |
|  | "StudName" : "Ayan", |
|  | "Hobbies" : "skating", |
|  | "Location" : null |
|  | } |
|  | { |
|  | "\_id" : 1, |
|  | "StudName" : "Megha", |
|  | "Grade" : "vii", |
|  | "Hobbies" : "InternetSurfing" |
|  | } |
|  | { |
|  | "\_id" : ObjectId("6253f413e88b8c9e787b194e"), |
|  | "StudName" : "Vamsi", |
|  | "Greade" : "vi" |
|  | } |
|  | > db.Student.find().skip(2).pretty() |
|  | { |
|  | "\_id" : ObjectId("6253f413e88b8c9e787b194e"), |
|  | "StudName" : "Vamsi", |
|  | "Greade" : "vi" |
|  | } |
|  | > db.food.insert( { \_id:1, fruits:['grapes','mango','apple';] } ) |
|  | 2022-04-11T15:05:51.894+0530 E QUERY [thread1] SyntaxError: missing ] after element list @(shell):1:57 |
|  | > db.food.insert({\_id:1,fruits:['grapes','mango','apple']}) |
|  | WriteResult({ "nInserted" : 1 }) |
|  | > db.food.insert({\_id:2,fruits:['grapes','mango','cherry']}) |
|  | WriteResult({ "nInserted" : 1 }) |
|  | > db.food.insert({\_id:3,fruits:['banana','mango']}) |
|  | WriteResult({ "nInserted" : 1 }) |
|  | > db.food.find({fruits:['grapes','mango','apple']}).pretty(); |
|  | { "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] } |
|  | > db.food.find({'fruits.1':'grapes'}) |
|  | > db.food.find({"fruits":{$size:2}}) |
|  | { "\_id" : 3, "fruits" : [ "banana", "mango" ] } |
|  | > db.food.find({\_id:1},{"fruits":{$slice:2}}) |
|  | { "\_id" : 1, "fruits" : [ "grapes", "mango" ] } |
|  | > db.food.find({fruits:{$all:["mango","grapes"]}}) |
|  | { "\_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] } |
|  | { "\_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] } |
|  | > db.food.update({\_id:3},{$set:{"fruits.1":"apple"}}) |
|  | WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) |
|  | > db.food.update({\_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}}) |
|  | WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) |
|  | > |
|  | > |
|  | > |
|  | > |
|  | > |
|  | > db.createCollection("Customers"); |
|  | { |
|  | "ok" : 0, |
|  | "errmsg" : "a collection 'bhuvana.Customers' already exists", |
|  | "code" : 48, |
|  | "codeName" : "NamespaceExists" |
|  | } |
|  | db.Customers.insert({\_custID:1,AcctBal:'100000',AcctType:"saving"}); |
|  | WriteResult({ "nInserted" : 1 }) |
|  | > db.Customers.aggregate({$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}}); |
|  | { "\_id" : null, "TotAccBal" : 0 } |
|  | db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}}); |
|  | { "\_id" : null, "TotAccBal" : 0 } |
|  | db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{\_id:"$custID",TotAccBal:{$sum:"$AccBal"}}},{$match:{TotAccBal:{$gt:1200}}}); |
|  | > |
|  |  |

## EXPERIMENT 2

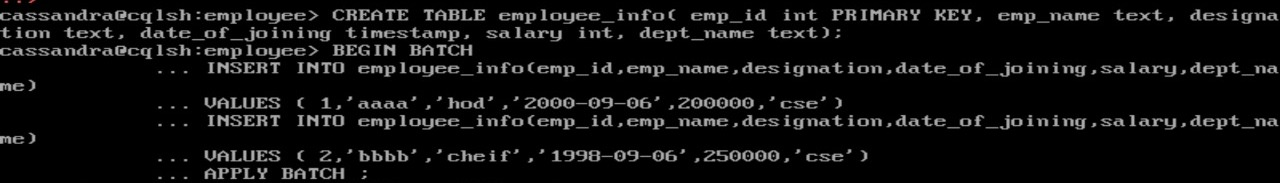


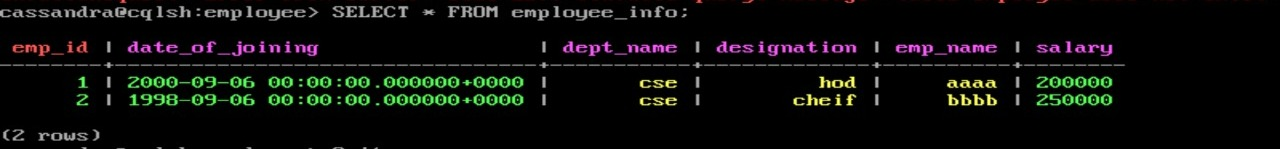
|  |
| --- |
| 1.Create a key space by name Employee |
|  | >CREATE KEYSPACE employee\_db WITH replication ={'class':'SimpleStrategy', 'replication\_factor' : 3}; |
|  | >use keyspace employee\_db; |
|  |  |
|  | 2.Create a column family by name Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name |
|  | >CREATE TABLE employee\_info( |
|  | emp\_id int primary key,emp\_name text,designation text,date\_of\_joining timestamp,salary int,dept\_name text); |
|  |  |
|  | 3.Insert the values into the table in batch |
|  | >cqlsh:employee\_db> begin batch |
|  | insert into employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,dept\_name) |
|  | values(1,'aaa','manager','2021-02-02',500000,'sales') |
|  | insert into employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,dept\_name) |
|  | values(2,'bbb','accountant','2020-02-02',100000,'sales') |
|  | insert into employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,dept\_name) |
|  | values(121,'ccc','accountant','2019-02-02',300000,'marketing') apply batch; |
|  |  |
|  | >cqlsh:employee\_db> select \* from employee\_info; |
|  | emp\_id | date\_of\_joining | dept\_name | designation | emp\_name | salary |
|  | --------+---------------------------------+-----------+-------------+----------+-------- |
|  | 1 | 2021-02-01 18:30:00.000000+0000 | sales | manager | aaa | 500000 |
|  | 2 | 2020-02-01 18:30:00.000000+0000 | sales | accountant | bbb | 100000 |
|  | 121 | 2019-02-01 18:30:00.000000+0000 | marketing | accountant | ccc | 300000 |
|  |  |
|  |  |
|  |  |
|  | 4.Update Employee name and Department of Emp-Id 121 |
|  | cqlsh:employee\_db> update employee\_info set emp\_name='xyz' where emp\_id=121; |
|  | cqlsh:employee\_db> update employee\_info set dept\_name='executive' where emp\_id=121; |
|  | cqlsh:employee\_db> select \* from employee\_info; |
|  |  |
|  | emp\_id | date\_of\_joining | dept\_name | designation | emp\_name | salary |
|  | --------+---------------------------------+-----------+-------------+----------+-------- |
|  | 1 | 2021-02-01 18:30:00.000000+0000 | sales | manager | aaa | 500000 |
|  | 2 | 2020-02-01 18:30:00.000000+0000 | sales | accountant | bbb | 100000 |
|  | 121 | 2019-02-01 18:30:00.000000+0000 | executive | accountant | xyz | 300000 |
|  |  |
|  |  |
|  | 5. Sort the details of Employee records based on salary |
|  |  |
|  | create table emp(id int, salary int,name text, primary key(id,salary)); |
|  |  |
|  | begin batch insert into emp(id,salary,name) values (1,89900,'kjl'); insert into emp(id,salary,name) values (2,70000,'uiu'); apply batch; |
|  |  |
|  | paging off; |
|  | Disabled Query paging. |
|  | cqlsh:employee> select \* from emp where id in (1,2) order by salary; |
|  |  |
|  | id | salary | name |
|  | ----+--------+------ |
|  | 2 | 70000 | uiu |
|  | 1 | 89900 | kjl |
|  |  |
|  |  |
|  | 6.Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee. |
|  | cqlsh:employee\_db> alter table employee\_info add projects set<text>; |
|  | cqlsh:employee\_db> select \* from employee\_info; |
|  |  |
|  | emp\_id | date\_of\_joining | dept\_name | designation | emp\_name | projects | salary |
|  | --------+---------------------------------+-----------+-------------+----------+----------+-------- |
|  | 1 | 2021-02-01 18:30:00.000000+0000 | sales | manager | aaa | null | 500000 |
|  | 2 | 2020-02-01 18:30:00.000000+0000 | sales | accountant | bbb | null | 100000 |
|  | 121 | 2019-02-01 18:30:00.000000+0000 | executive | accountant | xyz | null | 300000 |
|  |  |
|  | (3 rows) |
|  |  |
|  | 7.Update the altered table to add project names. |
|  | cqlsh:employee\_db> update employee\_info set projects=projects+{'1111'} where emp\_id=1; |
|  | cqlsh:employee\_db> update employee\_info set projects=projects+{'2222'} where emp\_id=2; |
|  | cqlsh:employee\_db> update employee\_info set projects=projects+{'2222'} where emp\_id=121; |
|  | cqlsh:employee\_db> select \* from employee\_info; |
|  |  |
|  | emp\_id | date\_of\_joining | dept\_name | designation | emp\_name | projects | salary |
|  | --------+---------------------------------+-----------+-------------+----------+----------+-------- |
|  | 1 | 2021-02-01 18:30:00.000000+0000 | sales | manager | aaa | {'1111'} | 500000 |
|  | 2 | 2020-02-01 18:30:00.000000+0000 | sales | accountant | bbb | {'2222'} | 100000 |
|  | 121 | 2019-02-01 18:30:00.000000+0000 | executive | accountant | xyz | {'2222'} | 300000 |
|  |  |
|  |  |
|  |  |
|  | 8 Create a TTL of 15 seconds to display the values of Employees. |
|  |  |
|  | insert into employee\_info(emp\_id,emp\_name,designation,date\_of\_joining,salary,dept\_name) |
|  | ... values(4,'abc','manager','2021-02-02',400000,'sales') using ttl 30; |
|  | cqlsh:employee\_db> select \* from employee\_info; |
|  |  |
|  | emp\_id | date\_of\_joining | dept\_name | designation | emp\_name | projects | salary |
|  | --------+---------------------------------+-----------+-------------+----------+----------+-------- |
|  | 1 | 2021-02-01 18:30:00.000000+0000 | sales | manager | aaa | {'1111'} | 500000 |
|  | 2 | 2020-02-01 18:30:00.000000+0000 | sales | accountant | bbb | {'2222'} | 100000 |
|  | 4 | 2021-02-01 18:30:00.000000+0000 | sales | manager | abc | null | 400000 |
|  | 121 | 2019-02-01 18:30:00.000000+0000 | executive | accountant | xyz | {'2222'} | 300000 |
|  |  |
|  | cqlsh:employee\_db> select ttl(emp\_name) from employee\_info where emp\_id=4; |
|  |  |
|  | ttl(emp\_name) |
|  | --------------- |
|  |  |
|  | (0 |

**1:**

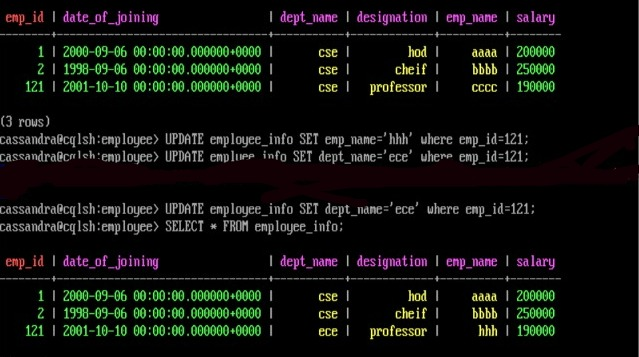
****

**2,3:**

****

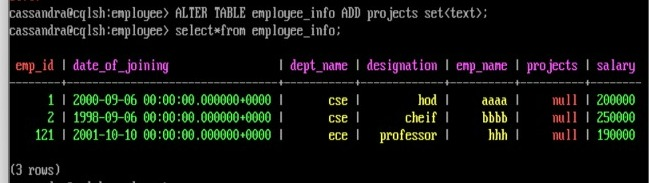
****

**4:**



**5:**

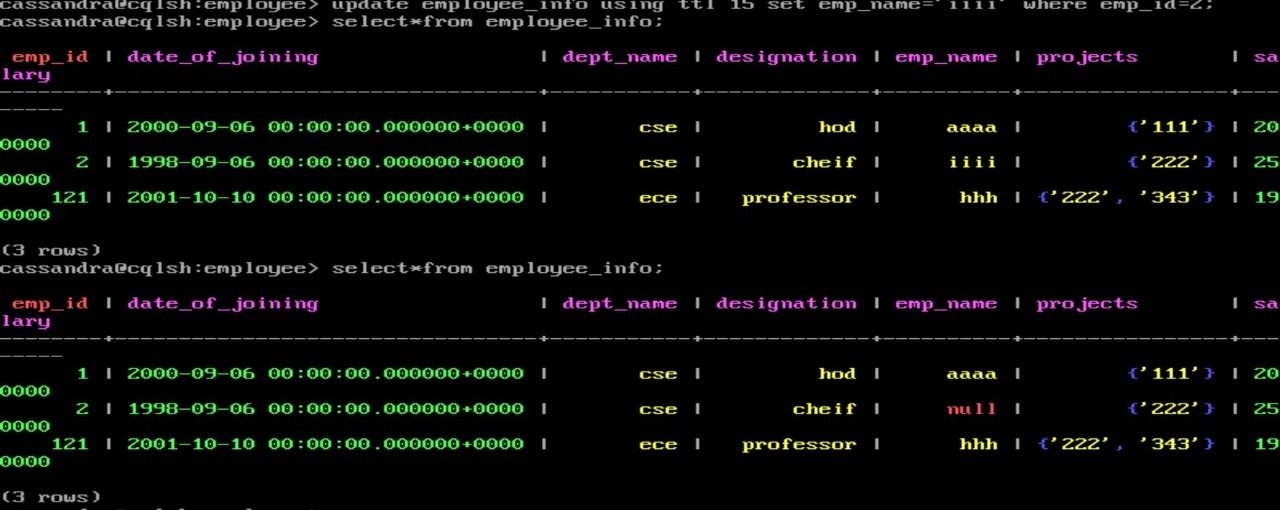
****

**6:**

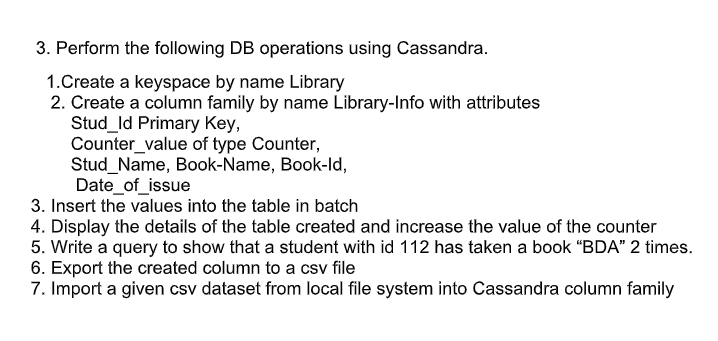
**7:**

****

**8.**

****

## EXPERIMENT 3



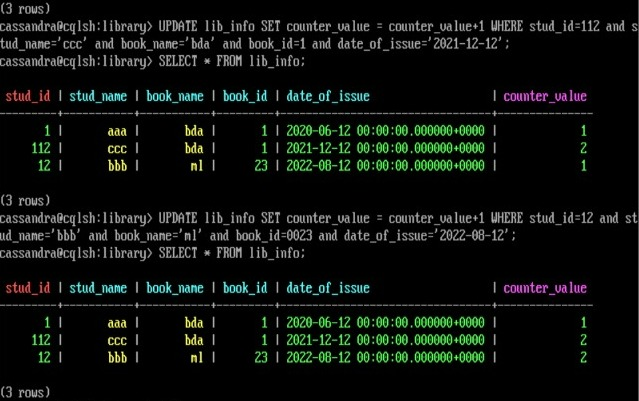
|  |
| --- |
| 1 Create a key space by name Library |
|  |  |
|  | CREATE KEYSPACE lib WITH replication={'class':'SimpleStrategy','replication\_factor' : 3}; |
|  | o/p: |
|  | describe keyspaces; |
|  |  |
|  | library1 system\_auth system system\_distributed employee1 |
|  | lib bhuvana students employee employeelab1 |
|  | "Students" newstudents employee\_db system\_traces |
|  | system\_schema lab employee\_224 students1 |
|  |  |
|  | 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. |
|  |  |
|  | 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\_of\_issue)); |
|  |  |
|  | o/p: SELECT \* FROM lib.library\_info; |
|  |  |
|  | stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value |
|  | ---------+-----------+-----------+---------------+---------+--------------- |
|  | ---------------------------------------------------------------------------------------------------- |
|  |  |
|  | 3. Insert the values into the table in batch |
|  |  |
|  | 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='2020-10-12' and book\_id=201; |
|  | UPDATE library\_info SET counter\_value = counter\_value+1 WHERE stud\_id=112 and stud\_name='ram' and book\_name='bda' and date\_of\_issue='2020-10-22' and book\_id=205; |
|  | UPDATE library\_info SET counter\_value = counter\_value+1 WHERE stud\_id=113 and stud\_name='siri' and book\_name='ml' and date\_of\_issue='2020-10-20' and book\_id=206; |
|  |  |
|  | o/p:SELECT \* FROM lib.library\_info; |
|  |  |
|  | stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value |
|  | ---------+-----------+-----------+---------------------------------+---------+--------------- |
|  | 111 | sam | ml | 2020-10-11 18:30:00.000000+0000 | 201 | 1 |
|  | 113 | siri | ml | 2020-10-19 18:30:00.000000+0000 | 206 | 1 |
|  | 112 | ram | bda | 2020-10-21 18:30:00.000000+0000 | 205 | 1 |
|  |  |
|  | ------------------------------------------------------------------------------------------------------ |
|  |  |
|  |  |
|  | 4. Display the details of the table created and increase the value of the counter |
|  |  |
|  | UPDATE library\_info SET counter\_value = counter\_value+1 WHERE stud\_id=113 and stud\_name='siri' and book\_name='ml' and date\_of\_issue='2020-10-20' and book\_id=206; |
|  |  |
|  | SELECT \* FROM lib.library\_info; |
|  |  |
|  | stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value |
|  | ---------+-----------+-----------+---------------------------------+---------+--------------- |
|  | 111 | sam | ml | 2020-10-11 18:30:00.000000+0000 | 201 | 1 |
|  | 113 | siri | ml | 2020-10-19 18:30:00.000000+0000 | 206 | 2 |
|  | 112 | ram | bda | 2020-10-21 18:30:00.000000+0000 | 205 | 1 |
|  |  |
|  |  |
|  | --------------------------------------------------------------------------------------------------- |
|  |  |
|  |  |
|  | 5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times. |
|  |  |
|  | SELECT \*FROM library\_info WHERE stud\_id=112; |
|  |  |
|  | o/p: stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value |
|  | ---------+-----------+-----------+---------------------------------+---------+--------------- |
|  | 112 | ram | bda | 2020-10-21 18:30:00.000000+0000 | 205 | 2 |
|  |  |
|  | 6. Export the created column to a csv file |
|  | COPY library\_info(stud\_id,counter\_value,stud\_name,book\_name,book\_id,date\_of\_issue) TO 'e:\libraryinfo.csv'; |
|  |  |
|  | o/p: |
|  | Using 11 child processes |
|  |  |
|  | Starting copy of lib.library\_info with columns [stud\_id, counter\_value, stud\_name, book\_name, book\_id, date\_of\_issue]. |
|  | Processed: 3 rows; Rate: 24 rows/s; Avg. rate: 24 rows/s |
|  | 3 rows exported to 1 files in 0.136 seconds. |
|  |  |
|  |  |
|  | 7. Import a given csv dataset from local file system into Cassandra column family |
|  |  |
|  | CREATE TABLE library\_info1(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\_of\_issue)); |
|  |  |
|  | SELECT \* FROM library\_info1; |
|  |  |
|  | stud\_id | stud\_name | book\_name | book\_id | date\_of\_issue | counter\_value |
|  | ---------+-----------+-----------+---------+---------------+--------------- |
|  |  |
|  |  |
|  | COPY library\_info1(stud\_id,counter\_value,stud\_name,book\_name,book\_id,date\_of\_issue) TO 'e:\libraryinfo.csv'; |
|  |  |
|  |  |
|  |  |
|  | o/p:Using 11 child processes |
|  |  |
|  | Starting copy of lib.library\_info1 with columns [stud\_id, counter\_value, stud\_name, book\_name, book\_id, date\_of\_issue]. |
|  | Processed: 3 rows; Rate: 5 rows/s; Avg. rate: 7 rows/s |
|  | 3 rows exported to 1 files in 0.135 seconds. |

**1,2:**

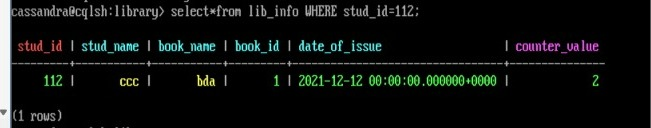
**3:**

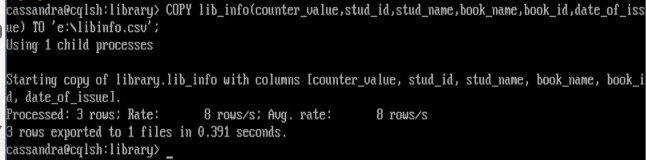
****

**4.**

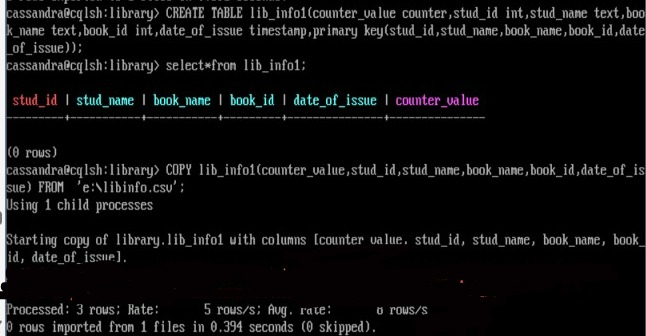
****

**5.**

****

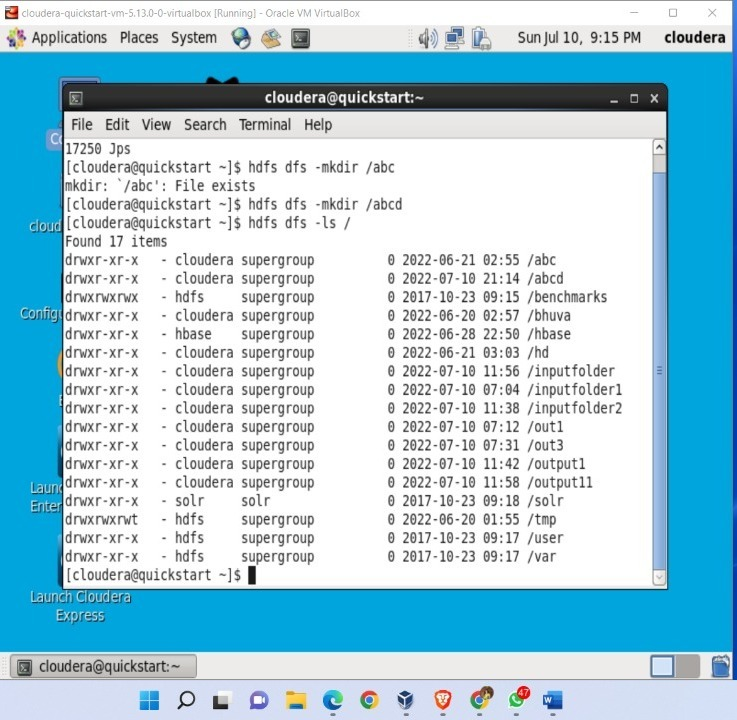
**6.**

**7.**

****

**Experiment 4**

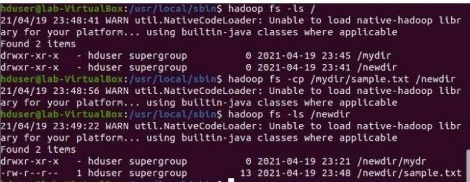
**Screenshot of Hadoop installed**



**Experiment 5**

**HDFS Commands**







**Experiment 6**

**Map Reduce for weather data**

**Driver class:**

package temperatureMax;

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

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

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

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

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

public class TempDriver {

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

if (args.length != 2) {

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

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(TempDriver.class);

job.setJobName("Max temperature");

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

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

job.setMapperClass(TempMapper.class);

job.setReducerClass(TempReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

}

}

**Mapper class:**

package temperatureMax;

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

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

import java.io.IOException;

public class TempMapper extends Mapper {

public static final int MISSING = 9999;

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

String line = value.toString();

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

int temperature;

if (line.charAt(87)=='+') temperature = Integer.parseInt(line.substring(88, 92));

else

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

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

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

}

}

**Reducer class:**

package temperatureMax;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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

import java.io.IOException;

public class TempReducer extends Reducer {

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

int max\_temp = 0;

for (IntWritable value : values) {

if(max\_temp<value.get()) {

max\_temp = value.get();

}

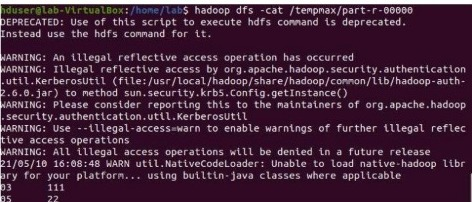
}

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

}

}

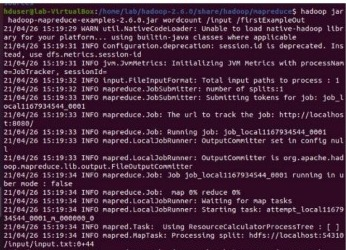
Output:

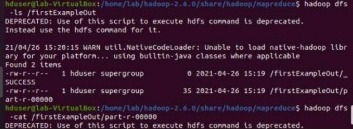




**Experiment 7**

**Map Reduce for top 10 maximum occurrences of words**







**Experiment 8**

**Map Reduce to demonstrating join operation**

**Driver Class**

package MapReduceJoin;

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, DeptNameMapper.class);

MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class, DeptEmpStrengthMapper.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);

}

}

**Mapper Class**

package MapReduceJoin;

import java.io.IOException;

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

public class DeptNameMapper 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], "0"), new Text(SingleNodeData[1]));

}

}

**Reducer Class**

package MapReduceJoin;

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);

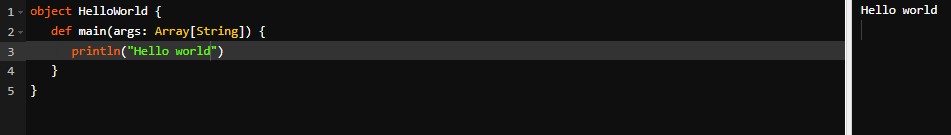
}

}

}

**Experiment 9**

**WordCount on scala and “Hello World” on scala IDE**



**Experiment 10**

**Use of RDD and FlatMap**

val text = sc.textFile("abc.txt")

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

val greaterThan4=counts.filter(x=>x.\_2>4);

greaterThan4.collect().forEach(println) **Input file:**

Hello Hello World Hello Hello Xyz Xyz Xyz Hello World Hello Xyz World World Xyz Hello World **Output:**

Hello 7

World 5

Xyz 5