VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
BELAGAVI 590018

BIG DATA ANALYTICS LAB RECORD

By

Shreyas K (1BM17CS098)

Under the Guidance of

Prof. Latha N R
Assistant Professor
Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

2017-2018

INDEX

Date: 24-09-2020

1. MongoDB: Student Database

Perform the following DB operations using MongoDB

1. **Create a database "Student" with the following attributes Rollno, Age, ContactNo, Email Id.**
2. **Insert appropriate values**
3. **Write query to update Email-Id of a student with rollno 10.**
4. **Replace the student name from "ABC" to "FEM" of rollno 11.**
5. **Export the created table into local file system**
6. **Drop the table**
7. **Import a given csv dataset from local file system into mongodb collection.**

```
use StudentDB
```

1. Create a database "Student" with the following attributes Rollno, Age, ContactNo, Email-Id

```
db.createCollection("Student")
```

2. Insert appropriate values

```
db.Student.insertMany([
{RollNo:10,Age:21,Name:"Shreyas",ContactNo:9482141788,EmailId:"shreyas@gm
ail.com"},
{RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.c
om"},
{RollNo:12,Age:30,Name:"Varun",ContactNo:9482141766,EmailId:"varun@gmail.
c om"},
{RollNo:13,Age:21,Name:"Arun",ContactNo:9482141755,EmailId:"arun@gmail.c
om"},
{RollNo:14,Age:26,Name:"Rahul",ContactNo:9442141788,EmailId:"rahul@gmail.
com"}]);

db.Student.find()
```

```
db.createCollection("Student")

db.Student.insertMany([
{RollNo:10,Age:21,Name:"Shreyas",ContactNo:9482141788,EmailId:"shreyas@gmail.com"},
{RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.com"},
{RollNo:12,Age:30,Name:"Varun",ContactNo:9482141766,EmailId:"varun@gmail.com"},
{RollNo:13,Age:21,Name:"Arun",ContactNo:9482141755,EmailId:"arun@gmail.com"},
{RollNo:14,Age:26,Name:"Rahul",ContactNo:9442141788,EmailId:"rahul@gmail.com"}]);

db.Student.find()
```

Student 0.038 sec.

	_id	RollNo	Age	Name	ContactNo	EmailId
1	ObjectId("5f...	10.0	21.0	Shreyas	9482141788.0	shreyas@gm...
2	ObjectId("5f...	11.0	25.0	ABC	9482141778.0	abc@gmail....
3	ObjectId("5f...	12.0	30.0	Varun	9482141766.0	varun@gmai...
4	ObjectId("5f...	13.0	21.0	Arun	9482141755.0	arun@gmail....
5	ObjectId("5f...	14.0	26.0	Rahul	9442141788.0	rahul@gmai...

3. Write query to update Email-Id of a student with rollno 10

```
db.Student.update({RollNo:10},{$set:{EmailId:"shreyask@gmail.com"}});
db.Student.find({RollNo:10});
```

```
db.Student.insertMany([
{RollNo:10,Age:21,Name:"Shreyas",ContactNo:9482141788,EmailId:"shreyas@gmail.com"},
{RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.com"},
{RollNo:12,Age:30,Name:"Varun",ContactNo:9482141766,EmailId:"varun@gmail.com"},
{RollNo:13,Age:21,Name:"Arun",ContactNo:9482141755,EmailId:"arun@gmail.com"},
{RollNo:14,Age:26,Name:"Rahul",ContactNo:9442141788,EmailId:"rahul@gmail.com"}]);

db.Student.find()

db.Student.update({RollNo:10},{$set:{EmailId:"shreyask@gmail.com"}});
db.Student.find({RollNo:10});
```

Student 0.005 sec.

	_id	RollNo	Age	Name	ContactNo	EmailId
1	ObjectId("5f...	10.0	21.0	Shreyas	9482141788.0	shreyask@g...

4. Replace the student name from “ABC” to “DEF” of rollno 11

```
db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})
```

```
db.Student.find()

db.Student.update({RollNo:10},{$set:{EmailId:"shreyask@gmail.com"}});
db.Student.find({RollNo:10});

db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})
```

Student 0.003 sec.

	_id	RollNo	Age	Name	ContactNo	EmailId
1	ObjectId("5f...	11.0	25.0	FEM	9482141778.0	abc@gmail....

5. Export the created table into local file system

```
mongoexport --db StudentDB --collection Student --fields
RollNo,Age,Name,ContactNo,EmailId --type=csv -o StudentData.csv
```

	A	B	C	D	E	F	G	H	I	J
1	RollNo	Age	Name	ContactNc	EmailId					
2	10	21	Shreyas	9.48E+09	shreyask@gmail.com					
3	11	25	FEM	9.48E+09	abc@gmail.com					
4	12	30	Varun	9.48E+09	varun@gmail.com					
5	13	21	Arun	9.48E+09	arun@gmail.com					
6	14	26	Rahul	9.44E+09	rahul@gmail.com					

C:\Windows\System32\cmd.exe

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoexport --db StudentDB --collection Student
--type=csv --fields RollNo,Age,Name,ContactNo,EmailId --out StudentData.csv
2020-12-26T23:49:06.286+0530    connected to: localhost
2020-12-26T23:49:06.289+0530    exported 5 records

C:\Program Files\MongoDB\Server\4.0\bin>
```

6. Drop the table

```
db.Student.drop()
```

```
db.Student.update({RollNo:10},{$set:{EmailId:"shreyask@gmail.com"}});
db.Student.find({RollNo:10});

db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})

db.Student.drop()
```

0.271 sec.

```
true
```

7. Import a given csv dataset from local file system into mongodb collection

```
mongoimport --db StudentDB --collection Student --type csv --file
StudentData.csv –-headerline
```

C:\Windows\System32\cmd.exe

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db StudentDB --collection Student
--type csv --file StudentData.csv --headerline
2020-12-26T23:55:54.431+0530    connected to: localhost
2020-12-26T23:55:54.951+0530    imported 5 documents
```

Date: 05-10-2020

2. MongoDB: Customer Database

Perform the following DB operations using MongoDB.

1. **Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type**
2. **Insert at least 5 values into the table**
3. **Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.**
4. **Determine Minimum and Maximum account balance for each customer_id.**
5. **Export the created collection into local file system**
6. **Drop the table**
7. **Import a given csv dataset from local file system into mongodb collection.**

```
use CustomerDB
```

1. Create a collection by name Customers with the following attributes.Cust_id, Acc_Bal, Acc_Type

```
db.createCollection("Customer")
```

2. Insert at least 5 values into the table

```
db.Customer.insert({cust_id:1,Acc_bal:1500,Acc_type:"Z"})
db.Customer.insert({cust_id:2,Acc_bal:3000,Acc_type:"A"})
db.Customer.insert({cust_id:1,Acc_bal:1200,Acc_type:"A"})
db.Customer.insert({cust_id:3,Acc_bal:500,Acc_type:"Z"})
db.Customer.insert({cust_id:2,Acc_bal:1600,Acc_type:"Z"})
db.Customer.find()
```

BDA localhost:27017 BDA

```
use CustomerDB

db.createCollection("Customer")

db.Customer.insert({cust_id:1,Acc_bal:1500,Acc_type:"Z"})
db.Customer.insert({cust_id:2,Acc_bal:3000,Acc_type:"A"})
db.Customer.insert({cust_id:1,Acc_bal:1200,Acc_type:"A"})
db.Customer.insert({cust_id:3,Acc_bal:500,Acc_type:"Z"})
db.Customer.insert({cust_id:2,Acc_bal:1600,Acc_type:"Z"})

db.Customer.find()
```

Customer 0.007 sec.

	_id	cust_id	Acc_bal	Acc_type
1	ObjectId("5f...	1.0	1500.0	Z
2	ObjectId("5f...	2.0	3000.0	A
3	ObjectId("5f...	1.0	1200.0	A
4	ObjectId("5f...	3.0	500.0	Z
5	ObjectId("5f...	2.0	1600.0	Z

3. Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer_id.

```
db.Customer.find({Acc_bal:{$gt:1200}, Acc_type:"Z"})
```

```
db.Customer.find()

db.Customer.find({Acc_bal:{$gt:1200}, Acc_type:"Z"})
```

Customer 0.001 sec.

	_id	cust_id	Acc_bal	Acc_type
1	ObjectId("5f...	1.0	1500.0	Z
2	ObjectId("5f...	2.0	1600.0	Z

4. Determine Minimum and Maximum account balance for each customer_id.

```
db.Customer.aggregate([
        {$group: { _id: "$cust_id",min_bal: {$min: "$Acc_bal"},
        max_bal: {$max: "$Acc_bal"}}}]);
```

```
db.Customer.aggregate([
        {$group: { _id: "$cust_id",min_bal: {$min: "$Acc_bal"},
        max_bal: {$max: "$Acc_bal"}}}]);
```

Customer 0.001 sec.

	_id	min_bal	max_bal
1	3.0	500.0	500.0
2	2.0	1600.0	3000.0
3	1.0	1200.0	1500.0

5. Export the created collection into local file system

```
mongoexport -d CustomerDB -c Customer -f cust_id,Acc_bal,Acc_type
--type=csv -o Customer.csv
```

A	B	C	D	E	F	G	H
cust_id	Acc_bal	Acc_type					
1	1500	Z					
2	3000	A					
1	1200	A					
3	500	Z					
2	1600	Z					

C:\Windows\System32\cmd.exe

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoexport -d CustomerDB
-c Customer -f cust_id,Acc_bal,Acc_type --type=csv -o Customer.csv

2020-12-27T00:36:10.657+0530    connected to: localhost
2020-12-27T00:36:10.658+0530    exported 5 records
```

6. Drop the table

```
db.Customer.drop()
```

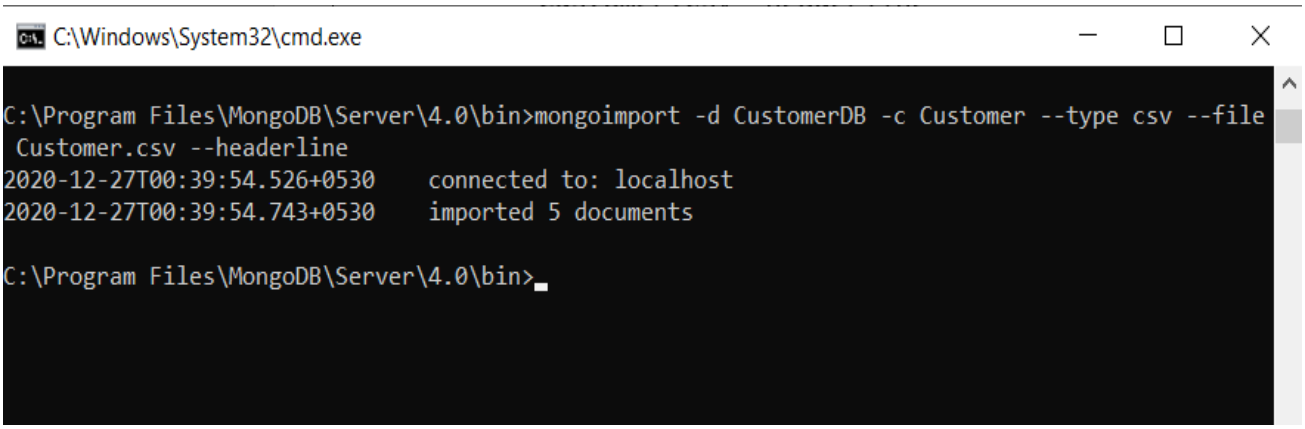
```
db.Customer.find({Acc_bal:{$gt:1200}, Acc_type:"Z"})

db.Customer.aggregate([
        {$group: { _id: "$cust_id",min_bal: {$min: "$Acc_bal"},
        max_bal: {$max: "$Acc_bal"}}}]);

db.Customer.drop()
```

0.063 sec.

```
true
```

7. Import a given csv dataset from local file system into mongodb collection

```
mongoimport -d CustomerDB -c Customer --type csv --file
Customer.csv –-headerline
```



Date: 12-10-2020

3. Cassandra: Employee Keyspace

Perform the following DB operations using Cassandra.

1. **Create a keyspace by name 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**
3. **Insert the values into the table in batch 3. Update Employee name and Department of Emp-Id 121**
4. **Sort the details of Employee records based on salary**
5. **Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.**
6. **Update the altered table to add project names.**
7. **Create a TTL of 15 seconds to display the values of Employees.**

1. Create a keyspace by name Employee

```
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.8 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> CREATE KEYSPACE Employee WITH REPLICATION={'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES;

system_schema  system   system_distributed  system_traces
system_auth    student  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> USE Employee;
cqlsh:employee> CREATE TABLE Employee_Info (Emp_Id int PRIMARY KEY, Emp_Name text,Designation text, DateOfJoining timest
amp, Salary double,Dept_Name text);
cqlsh:employee> DESCRIBE TABLES;

employee_info
```

3. Insert the values into the table in batch

```
cqlsh:employee> BEGIN BATCH INSERT INTO Employee_Info(Emp_Id , Emp_Name ,Designation , DateOfJoining ,Salary ,Dept_Name)
 VALUES(120,'Nam','Manager','2020-08-01',1000000,'Development');INSERT INTO Employee_Info(Emp_Id , Emp_Name ,Designation
 , DateOfJoining ,Salary ,Dept_Name) VALUES(121,'Amy','SE','2020-10-18',60000,'Development');INSERT INTO Employee_Info(E
mp_Id , Emp_Name ,Designation , DateOfJoining ,Salary ,Dept_Name) VALUES(122,'Penny','SDET','2020-01-08',50000,'R&D');IN
SERT INTO Employee_Info(Emp_Id , Emp_Name ,Designation , DateOfJoining ,Salary ,Dept_Name) VALUES(123,'Shelly','Data Ana
lyst','2020-10-18',40000,'R&D');INSERT INTO Employee_Info(Emp_Id , Emp_Name ,Designation , DateOfJoining ,Salary ,Dept_N
ame) VALUES(124,'Leo','Manager','2019-08-18',1000000,'HR');APPLY BATCH;
cqlsh:employee> SELECT * FROM employee_info;

 emp_id | dateofjoining                   | dept_name   | designation  | emp_name | salary
--------+---------------------------------+-------------+--------------+----------+--------
    120 | 2020-07-31 18:30:00.000000+0000 | Development |      Manager |      Nam |  1e+06
    123 | 2020-10-17 18:30:00.000000+0000 |         R&D | Data Analyst |   Shelly |  40000
    122 | 2020-01-07 18:30:00.000000+0000 |         R&D |         SDET |    Penny |  50000
    121 | 2020-10-17 18:30:00.000000+0000 | Development |           SE |      Amy |  60000
    124 | 2019-08-17 18:30:00.000000+0000 |          HR |      Manager |      Leo |  1e+06

(5 rows)
```

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

```
cqlsh:employee> UPDATE Employee_Info SET Emp_Name = 'Raj' , Dept_Name='R&D' WHERE Emp_Id=121;
cqlsh:employee> SELECT * FROM employee_info;

 emp_id | dateofjoining                   | dept_name   | designation  | emp_name | salary
--------+---------------------------------+-------------+--------------+----------+--------
    120 | 2020-07-31 18:30:00.000000+0000 | Development |      Manager |      Nam |  1e+06
    123 | 2020-10-17 18:30:00.000000+0000 |         R&D | Data Analyst |   Shelly |  40000
    122 | 2020-01-07 18:30:00.000000+0000 |         R&D |         SDET |    Penny |  50000
    121 | 2020-10-17 18:30:00.000000+0000 |         R&D |           SE |      Raj |  60000
    124 | 2019-08-17 18:30:00.000000+0000 |          HR |      Manager |      Leo |  1e+06
```

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

```
cqlsh:employee> ALTER TABLE employee_info ADD Project VARCHAR;
cqlsh:employee> DESCRIBE TABLE employee_info;

CREATE TABLE employee.employee_info (
    emp_id int PRIMARY KEY,
    dateofjoining timestamp,
    dept_name text,
    designation text,
    emp_name text,
    project text,
    salary double
```

6. Update the altered table to add project names.

```
cqlsh:employee> UPDATE employee_info SET project='EDM' WHERE emp_id=120;
cqlsh:employee> UPDATE employee_info SET project='Alexa' WHERE emp_id=121;
cqlsh:employee> UPDATE employee_info SET project='Health Monitoring System' WHERE emp_id=122;
cqlsh:employee> UPDATE employee_info SET project='Prediction App' WHERE emp_id=123;
cqlsh:employee> UPDATE employee_info SET project='Stock Management' WHERE emp_id=120;
cqlsh:employee> SELECT * FROM employee_info;

 emp_id | dateofjoining                   | dept_name   | designation  | emp_name | project                  | salary
--------+---------------------------------+-------------+--------------+----------+--------------------------+--------
    120 | 2020-07-31 18:30:00.000000+0000 | Development |      Manager |      Nam |         Stock Management |  1e+06
    123 | 2020-10-17 18:30:00.000000+0000 |         R&D | Data Analyst |   Shelly |           Prediction App |  40000
    122 | 2020-01-07 18:30:00.000000+0000 |         R&D |         SDET |    Penny | Health Monitoring System |  50000
    121 | 2020-10-17 18:30:00.000000+0000 |         R&D |           SE |      Raj |                    Alexa |  60000
    124 | 2019-08-17 18:30:00.000000+0000 |          HR |      Manager |      Leo |                     null |  1e+06
```

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

```
cqlsh:employee> INSERT INTO Employee_Info(Emp_Id , Emp_Name ,Designation , DateOfJoining ,Salary ,Dept_Name) VALUES(125,
'Joe','Software Developer','2020-10-01',60000,'Development') USING TTL 15;
cqlsh:employee> SELECT TTL(designation) FROM employee_Info where Emp_id=125;

 ttl(designation)
------------------
               13
```

Date: 02-11-2020

4. Cassandra: Library Keyspace

Perform the following DB operations using Cassandra.

1. **Create a keyspace by name 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**
3. **Insert the values into the table in batch**
4. **Display the details of the table created and increase the value of the counter**
5. **Write a query to show that a student with id 112 has taken a book "BDA" 2 times.**
6. **Export the created column to a csv file**
7. **Import a given csv dataset from local file system into Cassandra column family**

1. Create a keyspace by name Library

```
cqlsh> CREATE KEYSPACE Library WITH REPLICATION = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES;

system_schema  system   student             employee
system_auth    library  system_distributed  system_traces
```

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> USE Library;
cqlsh:library> CREATE TABLE Library_Info (Stud_id int,Counter_value counter, Stud_Name text,Book_Name text,Book_Id int,D
oi timestamp,PRIMARY KEY(Stud_id,Stud_Name,Book_Name,Book_id,doi));
cqlsh:library> DESCRIBE TABLES;

library_info
```

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='Nam' AND Book_Na
me='BDA' and Book_id=121 and Doi='2020-11-05' ;
cqlsh:library> UPDATE Library_Info SET Counter_value = Counter_value+1 WHERE Stud_id=112 and Stud_Name='Amy' AND Book_Na
me='BDA' and Book_id=122 and Doi='2020-10-05' ;
cqlsh:library> UPDATE Library_Info SET Counter_value = Counter_value+1 WHERE Stud_id=113 and Stud_Name='Penny' AND Book_
Name='DSR' and Book_id=131 and Doi='2020-11-05' ;
cqlsh:library> UPDATE Library_Info SET Counter_value = Counter_value+1 WHERE Stud_id=114 and Stud_Name='Shelly' AND Book
_Name='SQM' and Book_id=141 and Doi='2020-11-03' ;
cqlsh:library> UPDATE Library_Info SET Counter_value = Counter_value+1 WHERE Stud_id=115 and Stud_Name='Leo' AND Book_Na
me='DSR' and Book_id=132 and Doi='2020-11-04'
```

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

```
cqlsh:library> SELECT * FROM  Library_Info;

 stud_id | stud_name | book_name | book_id | doi                             | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     114 |    Shelly |       SQM |     141 | 2020-11-02 18:30:00.000000+0000 |             1
     111 |       Nam |       BDA |     121 | 2020-11-04 18:30:00.000000+0000 |             1
     113 |     Penny |       DSR |     131 | 2020-11-04 18:30:00.000000+0000 |             1
     112 |       Amy |       BDA |     122 | 2020-10-04 18:30:00.000000+0000 |             1
     115 |       Leo |       DSR |     132 | 2020-11-03 18:30:00.000000+0000 |             1
```

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

```
cqlsh:library> UPDATE Library_Info SET Counter_value = Counter_value+1 WHERE Stud_id=112 and Stud_Name='Amy' AND Book_Na
me='BDA' and Book_id=122 and Doi='2020-10-05' ;
cqlsh:library> SELECT * FROM  Library_Info;

 stud_id | stud_name | book_name | book_id | doi                             | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     114 |    Shelly |       SQM |     141 | 2020-11-02 18:30:00.000000+0000 |             1
     111 |       Nam |       BDA |     121 | 2020-11-04 18:30:00.000000+0000 |             1
     113 |     Penny |       DSR |     131 | 2020-11-04 18:30:00.000000+0000 |             1
     112 |       Amy |       BDA |     122 | 2020-10-04 18:30:00.000000+0000 |             2
     115 |       Leo |       DSR |     132 | 2020-11-03 18:30:00.000000+0000 |             1
```

6. Export the created column to a csv file

```
cqlsh:library> COPY Library_Info(Stud_id,Counter_value,Stud_Name,Book_Name,Book_id,doi) TO 'C:\Users\lenovo\Desktop\BDA\
LAB\LAB 6\libraryInfo.csv';
Using 3 child processes

Starting copy of library.library_info with columns [stud_id, counter_value, stud_name, book_name, book_id, doi].
Processed: 5 rows; Rate:      20 rows/s; Avg. rate:       1 rows/s
5 rows exported to 1 files in 3.812 seconds.
```

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

```
cqlsh:library> COPY Library_Info(Stud_id,Counter_value,Stud_Name,Book_Name,Book_id,doi) FROM 'C:\Users\lenovo\Desktop\BD
A\LAB\LAB 6\libraryInfo.csv';
Using 3 child processes
```

Date: 09-11-2020

5. Hadoop: Word Count

Hadoop program to find the word count

1. Starting Hadoop Cluster

```
$ su hduser
$ cd
$ start-all.sh
```

```
shreyas@ubuntu:~$ su hduser
Password:
hduser@ubuntu:/home/shreyas$ cd
hduser@ubuntu:~$ start-all.sh
```

```
hduser@ubuntu:~$ jps
3472 ResourceManager
3745 NodeManager
3320 SecondaryNameNode
3113 DataNode
4398 NameNode
4943 Jps
hduser@ubuntu:~$
```

2. Creating a file to count words

Open ▾ file1.txt ~/Desktop Save

```
hi how are you
how is your job
how is your family
how is your brother
how is your sister
```

3. Moving file to Hadoop system

```
$ hadoop fs –mkdir /rgs1
$ hadoop fs –ls /
$ hadoop fs –copyFromLocal /home/shreyas/Desktop/file1.txt
/prg1/test.txt
```

```
hduser@ubuntu:~$ hadoop fs -mkdir /prg1
```

```
hduser@ubuntu:~$ hadoop fs -ls /
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
20/12/26 22:24:00 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
Found 5 items
drwxr-xr-x   - hduser supergroup          0 2020-12-21 11:27 /input1
drwxr-xr-x   - hduser supergroup          0 2020-12-26 22:23 /prg1
drwxr-xr-x   - hduser supergroup          0 2020-12-08 06:30 /rgs1
drwxr-xr-x   - hduser supergroup          0 2020-12-14 01:16 /rgs2
drwxr-xr-x   - hduser supergroup          0 2020-12-14 08:55 /rgs3
hduser@ubuntu:~$ hadoop fs -copyFromLocal /home/shreyas/Desktop/file1.txt /prg1
/test.txt
```

4. Running the JAR file

```
$ hadoop jar /home/shreyas/Desktop/wordcount.jar WordCount
/prg1/test.txt /prg1/output/
```

```
hduser@ubuntu:~$ hadoop jar /home/shreyas/Desktop/wordcount.jar WordCount /prg1
/test.txt /prg1/output/
```

5. Output

```
$ hadoop fs –ls /prg1/
$ hadoop fs –cat /rgs1/output/part-r-00000
```

```
hduser@ubuntu:~$ hadoop fs -ls /prg1
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
20/12/26 23:39:17 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2020-12-26 22:54 /prg1/output
-rw-r--r--   1 hduser supergroup         89 2020-12-26 22:34 /prg1/test.txt
```

```
hduser@ubuntu:~$ hadoop fs -cat /prg1/output/part-r-00000
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
20/12/26 23:42:11 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
are     1
brother 1
family  1
hi      1
how     5
is      4
job     1
sister  1
you     1
your    4
```

6. Stopping Hadoop

```
$ stop-all.sh
```

```
hduser@ubuntu:~$ stop-all.sh
```

Date: 07-12-2020

6. Hadoop: Average Temperature

Hadoop program to find the Average Temperature

1. Starting Hadoop Cluster

```
$ su hduser
$ cd\
$ start-all.sh
$ jps
```

```
shreyas@ubuntu:~$ su hduser
Password:
hduser@ubuntu:/home/shreyas$ cd
hduser@ubuntu:~$ start-all.sh
```

```
hduser@ubuntu:~$ jps
3472 ResourceManager
3745 NodeManager
3320 SecondaryNameNode
3113 DataNode
4398 NameNode
4943 Jps
hduser@ubuntu:~$
```

2. Copying the binary file to the Hadoop file system as a text file

```
$ hadoop fs –copyFromLocal /home/shreyas/Desktop/1901 /prg2/test.txt
$ hadoop fs –ls /prg2
```

```
hduser@ubuntu:~$ hadoop fs -copyFromLocal /home/shreyas/Desktop/1901 /prg2/test
.txt
hduser@ubuntu:~$ hadoop fs -ls /prg2
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
20/12/26 23:56:42 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 hduser supergroup     888190 2020-12-26 23:56 /prg2/test.txt
```

3. Running the JAR file

```
$ hadoop jar home/shreyas/Desktop/avgtemp.jar AverageDriver
/prg2/test.txt /prg2/output
```

4. Output

```
$ hadoop fs –ls /prg2
$ hadoop fs –cat /prg2/output/part-r-0000
```

```
hduser@ubuntu:~$ hadoop jar /home/shreyas/Desktop/avgtemp.jar AverageDriver /pr
g2/test.txt prg2/output/
hduser@ubuntu:~$ hadoop fs -ls /prg2
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.uti
l.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.6.0.ja
r) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.sec
urity.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
 access operations
WARNING: All illegal access operations will be denied in a future release
20/12/27 01:03:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2020-12-27 01:02 /prg2/output
-rw-r--r--   1 hduser supergroup     888190 2020-12-27 00:58 /prg2/test.txt
hduser@ubuntu:~$ hadoop fs -cat /prg2/output/part-r-00000
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.uti
l.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.6.0.ja
r) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.sec
urity.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
 access operations
WARNING: All illegal access operations will be denied in a future release
20/12/27 01:19:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
1901    46
```

5. Stopping Hadoop

```
$ stop-all.sh
```

```
hduser@ubuntu:~$ stop-all.sh
```

Date: 14-12-2020

7. Hive: Employee Table

Write Queries in Hive to do the following

1. **Create an external table named with the following attributes -> Empl_ID ->Emp_Name -> Designation -> Salary**
2. **Load data into table from a given file**
3. **Create a view to Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.**
4. **Alter the table to add a column Dept_Id and Generate a query to retrieve the employee details in order by using Dept_Id**
5. **Generate a query to retrieve the number of employees in each department whose salary is greater than 30000**
6. **Create another table Department with attributes -> Dept_Id ->Dept_name ->Emp_Id**
7. **Display the cumulative details of each employee along with department details**

1. Create an external table named with the following attributes -> Empl_ID ->Emp_Name -> Designation -> Salary

```
>CREATE DATABASE IF NOT EXISTS EMPLOYEES COMMENT ‘EMPLOYEE
Details’ WITH DBPROPERTIES(‘creator’=’Shreyas’);
>SHOW DATABASES;
>DESCRIBE DATABASE EMPLOYEES;
>USE EMPLOYEES;
> CREATE EXTERNAL TABLE IF NOT EXISTS EMPLOYEES (EMP_ID INT,
EMP_NAME STRING, DESIGNATION STRING, SALARY FLOAT) ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\T' LOCATION '/EMPLOYEE_INFO';
>DESCRIBE FORMATTED EMPLOYEES;
```

2. Load data into table from a given file

```
>INSERT INTO TABLE EMPLOYEES VALUES (1, ’Arun’, ’Manager’, 1000000),
(2, ’Ashish’, ’Clerk’, 50000), (3, ’Arvindh’, ’Intern’, 20000),
(4, ’Shruti’, ’HR’, 35000);
>SELECT * FROM EMPLOYEES;
```

3. Create a view to Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

```
>CREATE VIEW EMPLOYEE_VIEW AS SELECT * FROM EMPLOYEES WHERE
SALARY>30000;
```

```
>SELECT * FROM EMPLOYEE_VIEW;
```

4. Alter the table to add a column Dept_Id and Generate a query to retrieve the employee details in order by using Dept_Id

```
>ALTER TABLE EMPLOYEES ADD COLUMNS (DEPT_ID INT);
>DESCRIBE FROMATTED EMPLOYEES;
```

5. Generate a query to retrieve the number of employees in each department whose salary is greater than 30000

```
SELECT DEPT_ID, COUNT(DEPT_ID) FROM EMPLOYEES WHERE SALARY >
30000 GROUP BY DEPT_ID;
```

6. Create another table Department with attributes -> Dept_Id ->Dept_name ->Emp_Id

```
CREATE EXTERNAL TABLE IF NOT EXISTS DEPARTMENTS (DEPT_ID INT,
DEPT_NAME STRING, EMP_ID INT) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\T' LOCATION '/DEPARTMENT';
```

7. Display the cumulative details of each employee along with department details

```
SELECT * FROM EMPLOYEES JOIN DEPARTMENT ON
EMPLOYEES.DEPT_ID = DEPARTMENTS.DEPT_ID;
```

```
hive> create database if not exists Employees comment 'Employee Details' with dbproperties ('creator'='Arun');
OK
Time taken: 0.179 seconds
hive> show databases;
OK
default
employees
Time taken: 0.022 seconds, Fetched: 2 row(s)
hive> DESCRIBE DATABASE EMPLOYEES
    > DESCRIBE DATABASE EMPLOYEES;
FAILED: ParseException line 2:0 missing EOF at 'DESCRIBE' near 'EMPLOYEES'
hive> DESCRIBE DATABASE EMPLOYEES;
OK
employees       Employee Details        hdfs://localhost:54310/user/hive/warehouse/employees.db hduser  USER
Time taken: 0.056 seconds, Fetched: 1 row(s)
hive> use employees;
OK
Time taken: 0.014 seconds
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS EMPLOYEES(EMP_ID INT,EMP_NAME STRING, DESIGNATION STRING, SALARY FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T' LOCATION '/EMPLOYEE_INFO';
OK
Time taken: 0.385 seconds
hive> describe formatted employees;
OK
# col_name              data_type               comment

emp_id                  int
emp_name                string
designation             string
salary                  float

# Detailed Table Information
Database:               employees
Owner:                  hduser
CreateTime:             Sat Dec 26 21:00:46 IST 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:54310/EMPLOYEE_INFO
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1608996646

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             T
        serialization.format    T
Time taken: 0.347 seconds, Fetched: 30 row(s)
hive> insert into table employees values (1,'Arun','Manager',1000000),(2,'Ashish','Clerk',50000),(3,'Arvindh','Intern',20000),(4,'Shruti','HR',35000);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20201226210346_a835ef8d-f924-4ae3-a668-53269f851079
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2020-12-26 21:03:51,219 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local369431830_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
```

```
Moving data to directory hdfs://localhost:54310/EMPLOYEE_INFO/.hive-staging_hive_2020-12-26_21-03-46_880_2733960997414858450-1/-ext-10000
Loading data to table employees.employees
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 85 HDFS Write: 253 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 5.2 seconds
hive> select * from employees;
OK
1       Arun    Manager 1000000.0
2       Ashish  Clerk   50000.0
3       Arvindh Intern  20000.0
4       Shruti  HR      35000.0
Time taken: 0.177 seconds, Fetched: 4 row(s)
hive> CREATE VIEW EMPLOYEE_VIEW AS SELECT * FROM EMPLOYEES WHERE SALARY>30000;
OK
Time taken: 0.48 seconds
hive> select * from employee_view;
OK
1       Arun    Manager 1000000.0
2       Ashish  Clerk   50000.0
4       Shruti  HR      35000.0
Time taken: 0.184 seconds, Fetched: 3 row(s)
hive> ALTER TABLE EMPLOYEES ADD COLUMNS (DEPT_ID INT);
OK
Time taken: 0.121 seconds
hive> describe formatted employees;
OK
# col_name              data_type               comment

emp_id                  int
emp_name                string
designation             string
salary                  float
dept_id                 int

# Detailed Table Information
Database:               employees
Owner:                  hduser
CreateTime:             Sat Dec 26 21:00:46 IST 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:54310/EMPLOYEE_INFO
Table Type:             EXTERNAL_TABLE
```

```
Table Parameters:
        EXTERNAL                TRUE
        last_modified_by        hduser
        last_modified_time      1608996930
        numFiles                1
        totalSize               93
        transient_lastDdlTime   1608996930

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             T
        serialization.format    T
Time taken: 0.05 seconds, Fetched: 35 row(s)
hive> SELECT DEPT_ID, COUNT(DEPT_ID) FROM EMPLOYEES WHERE SALARY>30000 GROUP BY DEPT_ID;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20201226210620_28191f63-6e08-45d0-9dda-7ab0c5b48ade
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2020-12-26 21:06:22,239 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1509417087_0002
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 878 HDFS Write: 506 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
NULL    0
Time taken: 1.536 seconds, Fetched: 1 row(s)
hive> CREATE EXTERNAL TABLE IF NOT EXISTS DEPARTMENTS(DEPT_ID INT,DEPT_NAME STRING, EMP_ID INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T' LOCATION '/DEPARTMENT';
OK
Time taken: 0.097 seconds
```

```
hive> SELECT * FROM EMPLOYEES JOIN DEPARTMENTS ON EMPLOYEES.DEPT_ID = DEPARTMENTS.DEPT_ID;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20201226212212_6bc64807-5ad0-4e26-9ad7-6f610c2e11b6
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.3.7-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2020-12-26 21:22:17     Starting to launch local task to process map join;      maximum memory = 477626368
2020-12-26 21:22:18     Dump the side-table for tag: 1 with group count: 0 into file: file:/tmp/mydir/503f8a1f-da20-4799-8320-5a20e503d551/hive_2020-12-26_21-22-12_048_5154997920250648288-1/-local-10004/H
ashTable-Stage-3/MapJoin-mapfile11--.hashtable
2020-12-26 21:22:18     Uploaded 1 File to: file:/tmp/mydir/503f8a1f-da20-4799-8320-5a20e503d551/hive_2020-12-26_21-22-12_048_5154997920250648288-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile11--.hasht
able (260 bytes)
2020-12-26 21:22:18     End of local task; Time Taken: 1.186 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2020-12-26 21:22:20,730 Stage-3 map = 100%,  reduce = 0%
Ended Job = job_local374276665_0004
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 625 HDFS Write: 253 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 8.686 seconds
hive>
```