VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
BELAGAVI 590018

BIG DATA ANALYTICS LAB RECORD

By

Namratha V (1BM17CS151)

Under the Guidance of

Prof. Latha NR
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

2020-2021

INDEX

SL NO.	DATE	PROGRAM	PAGE NO.
1.	24-09-2020	MongoDB : Student Database	3
2.	05-10-2020	MongoDB : Customer Database	7
3.	12-10-2020	Cassandra : Employee Keyspace	11
4.	02-11-2020	Cassandra : Library Keyspace	13
5.	09-11-2020	Hadoop : Word Count	15
6.	07-12-2020	Hadoop : Average Temperature	18
7.	14-12-2020	Hive : Employee Table	20

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:"Nam",ContactNo:9482141788,EmailId:"nam@gmail.c
om"},
{RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.c
om"},
{RollNo:12,Age:30,Name:"Amy",ContactNo:9482141766,EmailId:"amy@gmail.c
om"},
{RollNo:13,Age:21,Name:"Penny",ContactNo:9482141755,EmailId:"penny@gma
il.com"},
{RollNo:14,Age:26,Name:"Leo",ContactNo:9442141788,EmailId:"leo@gmail.c
om"}]);
db.Student.find()
```

```
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:"Nam",ContactNo:9482141788,EmailId:"nam@gmail.com"},
                       {RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.com"},
                       {RollNo:12,Age:30,Name:"Amy",ContactNo:9482141766,EmailId:"amy@gmail.com"},
                       {RollNo:13,Age:21,Name:"Penny",ContactNo:9482141755,EmailId:"penny@gmail.com"},
                       {RollNo:14,Age:26,Name:"Leo",ContactNo:9442141788,EmailId:"leo@gmail.com"}]);
db.Student.find()
//3. Write query to update Email-Id of a student with rollno 10
db.Student.update({RollNo:10},{$set:{EmailId:"namratha@gmail.com"}});
db.Student.find({RollNo:10})
//4. Replace the student name from "ABC" to "FEM" of rollno 11
db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})
```

Student 0.001 sec.

	_id	RollNo	Age	Name	ContactNo	EmailId
1	ObjectId("5...	10.0	21.0	Nam	9482141788.0	nam@gma...
2	ObjectId("5...	11.0	25.0	ABC	9482141778.0	abc@gmail...
3	ObjectId("5...	12.0	30.0	Amy	9482141766.0	amy@gmai...
4	ObjectId("5...	13.0	21.0	Penny	9482141755.0	penny@g...
5	ObjectId("5...	14.0	26.0	Leo	9442141788.0	leo@gmail....

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

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

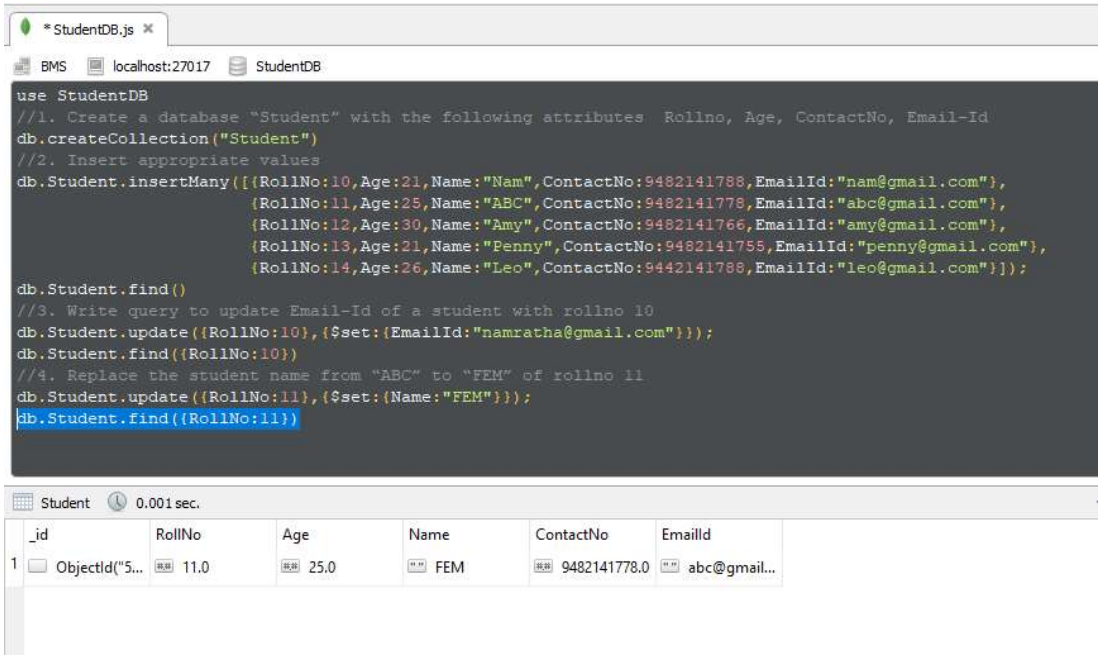
```
use StudentDB
//1. Create a database "Student" with the following attributes  Rollno, Age, ContactNo, Email-Id
db.createCollection("Student")
//2. Insert appropriate values
db.Student.insertMany([{RollNo:10,Age:21,Name:"Nam",ContactNo:9482141788,EmailId:"nam@gmail.com"},
                       {RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.com"},
                       {RollNo:12,Age:30,Name:"Amy",ContactNo:9482141766,EmailId:"amy@gmail.com"},
                       {RollNo:13,Age:21,Name:"Penny",ContactNo:9482141755,EmailId:"penny@gmail.com"},
                       {RollNo:14,Age:26,Name:"Leo",ContactNo:9442141788,EmailId:"leo@gmail.com"}]);
db.Student.find()
//3. Write query to update Email-Id of a student with rollno 10
db.Student.update({RollNo:10},{$set:{EmailId:"namratha@gmail.com"}});
db.Student.find({RollNo:10})
//4. Replace the student name from "ABC" to "FEM" of rollno 11
db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})
```

Student 0.002 sec.

	_id	RollNo	Age	Name	ContactNo	EmailId
1	ObjectId("5...	10.0	21.0	Nam	9482141788.0	namratha...

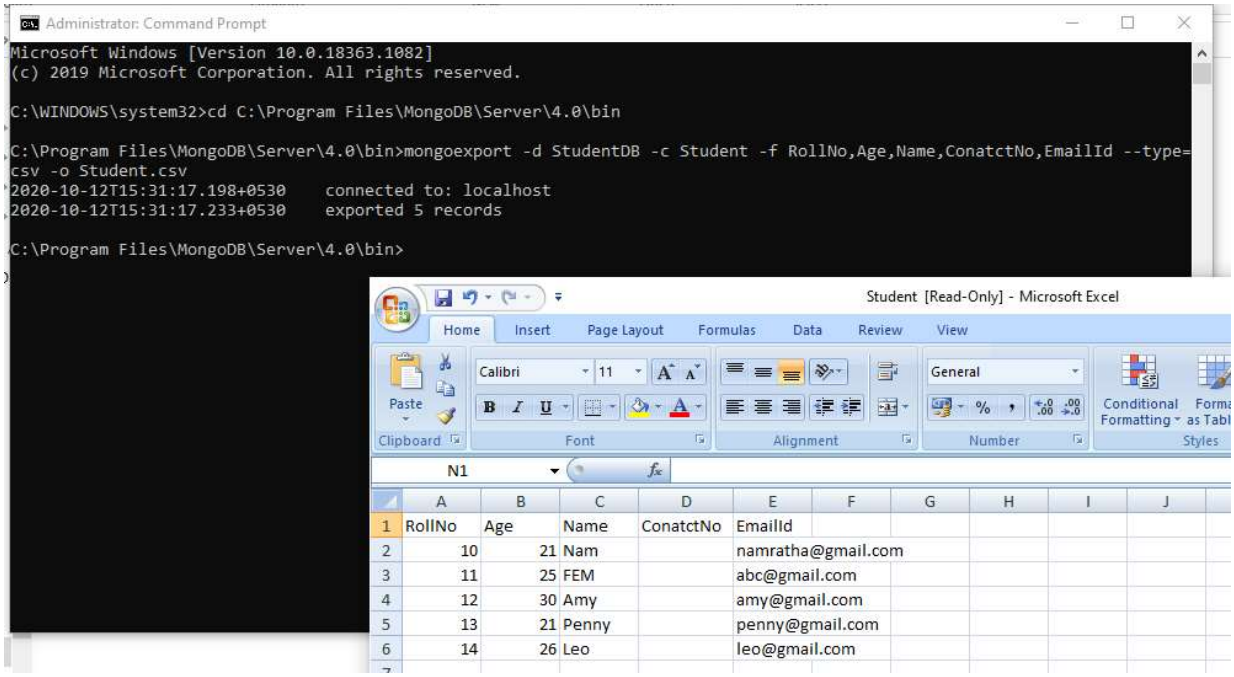
4. Replace the student name from “ABC” to “FEM” of rollno 11

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



5. Export the created table into local file system

```
mongoexport -d StudentDB -c Student -f
RollNo,Age,Name,ConatctNo,EmailId --type=csv -o Student.csv
```



6. Drop the table

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

```
* StudentDB.js
BMS   localhost:27017   StudentDB
//2. Insert appropriate values
db.Student.insertMany([{RollNo:10,Age:21,Name:"Nam",ContactNo:9482141788,EmailId:"nam@gmail.com"},
                       {RollNo:11,Age:25,Name:"ABC",ContactNo:9482141778,EmailId:"abc@gmail.com"},
                       {RollNo:12,Age:30,Name:"Amy",ContactNo:9482141766,EmailId:"amy@gmail.com"},
                       {RollNo:13,Age:21,Name:"Penny",ContactNo:9482141755,EmailId:"penny@gmail.com"},
                       {RollNo:14,Age:26,Name:"Leo",ContactNo:9442141788,EmailId:"leo@gmail.com"}]);
db.Student.find()
//3. Write query to update Email-Id of a student with rollno 10
db.Student.update({RollNo:10},{$set:{EmailId:"namratha@gmail.com"}});
db.Student.find({RollNo:10})
//4. Replace the student name from "ABC" to "FEM" of rollno 11
db.Student.update({RollNo:11},{$set:{Name:"FEM"}});
db.Student.find({RollNo:11})
//5. Export the created table into local file system
mongoexport -d StudentDB -c Student -f RollNo,Age,Name,ConatctNo,EmailId --type=csv -o Student.csv
//6. Drop the table
db.Student.drop()

0.221 sec.
true
```

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

```
mongoimport -d StudentDB -c Student --type csv --file Student.csv
–headerline
```

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d StudentDB -c Student --type csv --file Student.csv --headerline
2020-10-12T15:37:04.986+0530    connected to: localhost
2020-10-12T15:37:05.156+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.**

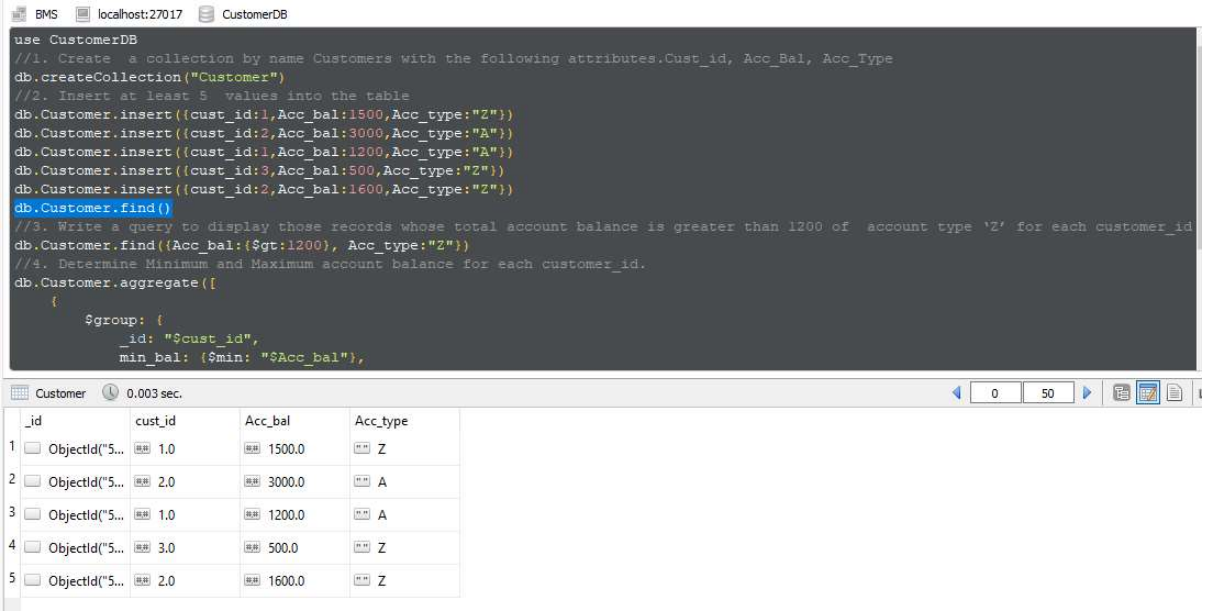
```
use CustomerDB
```

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()
```



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"})
```

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.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()
//3. Write a query to display those records whose total account balance is gr
db.Customer.find({Acc_bal:{$gt:1200}, Acc_type:"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"}
        }
    }
]);
```

Customer 0.819 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
```

```
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
2020-10-12T16:07:31.012+0530    connected to: localhost
2020-10-12T16:07:31.016+0530    exported 5 records

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

Customer [Read-Only] - Microsoft Exce

A1 cust_id

	A	B	C
1	cust_id	Acc_bal	Acc_type
2	1	1500	Z
3	2	3000	A
4	1	1200	A
5	3	500	Z
6	2	1600	Z

6. Drop the table

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

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

//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
//6. Drop the table
db.Customer.drop()
//7. Import a given csv dataset from local file system into mongodb collection
mongoimport -d CustomerDB -c Customer --type csv --file Customer.csv --headerline
```

0.228 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
```

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d CustomerDB -c Customer --type csv --file Customer.csv --headerlin
e
2020-10-12T16:08:43.814+0530    connected to: localhost
2020-10-12T16:08:45.037+0530    [########################] CustomerDB.Customer  69B/69B (100.0%)
2020-10-12T16:08:45.038+0530    [########################] CustomerDB.Customer  69B/69B (100.0%)
2020-10-12T16:08:45.038+0530    imported 5 documents

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

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

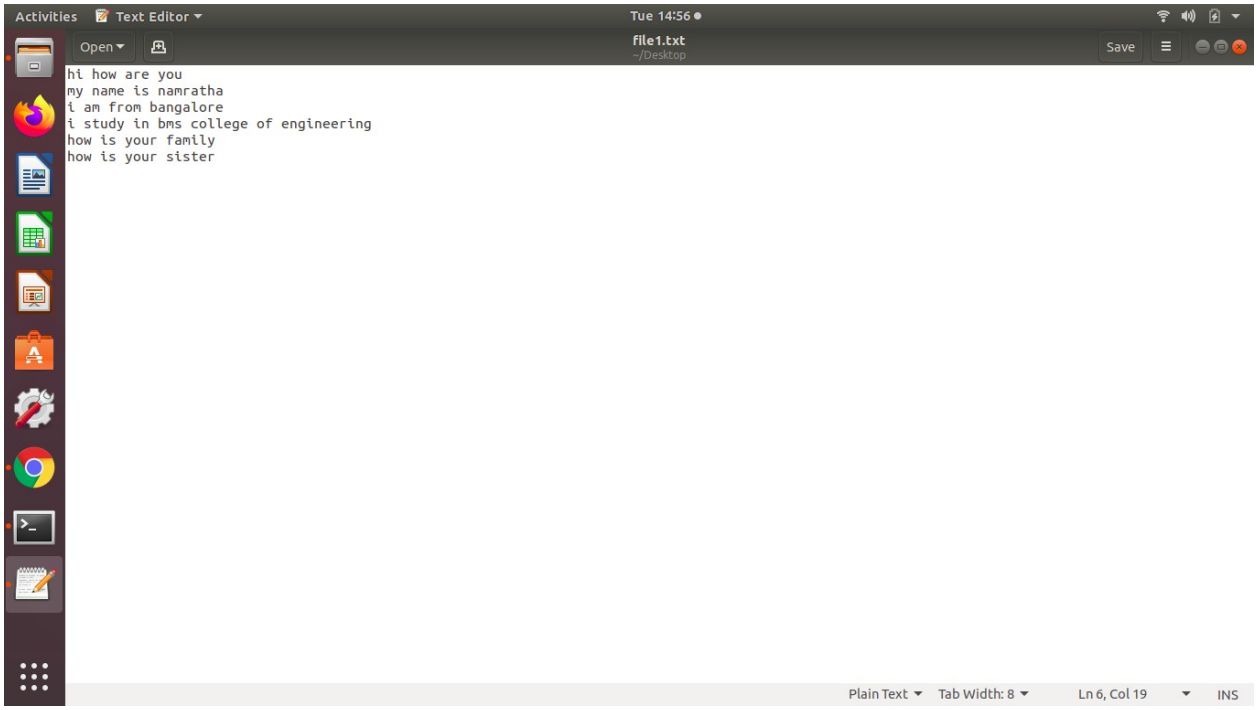
1. Starting Hadoop Cluster

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

```
(base) lenovo@lenovo-ThinkPad-Edge-E431:~$ su hduser
Password:
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ cd\
>
hduser@lenovo-ThinkPad-Edge-E431:~$ start-all.sh
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ jps
10051 NameNode
10244 DataNode
10645 ResourceManager
11462 Jps
10475 SecondaryNameNode
10991 NodeManager
```

2. Creating a file to count words



3. Moving file to Hadoop system

```
$ hadoop fs –mkdir /rgs1
$ hadoop fs –ls /
$ hadoop fs –copyFromLocal /home/lenovo/Desktop/Nam-BDA-LAB/WordCount/wordcount_file.txt /rgs1/wc_test.txt
```

```
hduser@lenovo-ThinkPad-Edge-E431:/$ hadoop fs -mkdir /rgs1
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ 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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/19 22:52:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2020-12-08 16:24 /rgs1
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ hadoop fs -copyFromLocal /home/lenovo/Desktop/Nam-BDA-LAB/WordCount/wordcount_file.txt /rgs1/wc_test.txt
```

4. Running the JAR file

```
$ hadoop jar //home/lenovo/Desktop/Nam-BDA-LAB/WordCount/wordcount.jar WordCount /rgs1/wc_test /rgs1/output/
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ hadoop jar /home/lenovo/Desktop/Nam-BDA-LAB/WordCount/wordcount.jar WordCount /rgs1/wc_test.txt /rgs1/outp
ut/
```

5. Output

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

```
hduser@lenovo-ThinkPad-Edge-E431:~$ hadoop fs -ls /rgs1/
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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/19 23:04:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
Found 4 items
drwxr-xr-x   - hduser supergroup          0 2020-12-08 16:24 /rgs1/output
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:22 /rgs1/test.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:31 /rgs1/test1.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-19 22:59 /rgs1/wc_test.txt
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ hadoop fs -cat /rgs1/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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/19 23:07:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
am      1
are     1
bangalore       1
bms     1
college 1
engineering     1
family  1
from    1
hi      1
how     3
```

6. Stopping Hadoop

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

```
hduser@lenovo-ThinkPad-Edge-E431:~$ 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
```

```
(base) lenovo@lenovo-ThinkPad-Edge-E431:~$ su hduser
Password:
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ cd\
>
hduser@lenovo-ThinkPad-Edge-E431:~$ start-all.sh
```

```
hduser@lenovo-ThinkPad-Edge-E431:~$ jps
15779 DataNode
15589 NameNode
16535 NodeManager
```

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

```
$ hadoop fs –copyFromLocal /home/lenovo/Desktop/Nam-BDA-LAB/LAB8/1901 /rgs1/AT_test.txt
$ hadoop –ls /rgs1
```

```
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ hadoop fs -copyFromLocal /home/lenovo/Desktop/Nam-BDA-LAB/LAB8/1901   /rgs1/AT_test.txt
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ hadoop fs -ls /rgs1
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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/20 09:04:57 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
Found 5 items
-rw-r--r--   1 hduser supergroup     888190 2020-12-20 09:04 /rgs1/AT_test.txt
drwxr-xr-x   - hduser supergroup          0 2020-12-08 16:24 /rgs1/output
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:22 /rgs1/test.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:31 /rgs1/test1.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-19 22:59 /rgs1/wc_test.txt
```

3. Running the JAR file

```
$ hadoop jar home/lenovo/Desktop/Nam-BDA-LAB/LAB8/Average.jar
AverageDriver /rgs1/AT_test.txt /rgs1/output2
```

```
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ hadoop jar /home/lenovo/Desktop/Nam-BDA-LAB/LAB8/Average.jar AverageDriver /rgs1/AT_test.txt /r
gs1/output2
```

4. Output

```
$ hadoop fs –ls /rgs1
$ hadoop fs –cat /rgs1/output2/part-r-00000
```

```
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ hadoop fs -ls /rgs1
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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/20 09:10:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
Found 6 items
-rw-r--r--   1 hduser supergroup     888190 2020-12-20 09:04 /rgs1/AT_test.txt
drwxr-xr-x   - hduser supergroup          0 2020-12-08 16:24 /rgs1/output
drwxr-xr-x   - hduser supergroup          0 2020-12-20 09:09 /rgs1/output2
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:22 /rgs1/test.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-08 15:31 /rgs1/test1.txt
-rw-r--r--   1 hduser supergroup        131 2020-12-19 22:59 /rgs1/wc_test.txt
```

```
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ hadoop fs -cat /rgs1/output2/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 reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/20 09:11:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
1901    46
```

5. Stopping Hadoop

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

```
hduser@lenovo-ThinkPad-Edge-E431:/home/lenovo$ 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_151 COMMENT ‘EMPLOYEE Details’ WITH DBPROPERTIES(‘creator’=’Namratha’);
>SHOW DATABASES;
>DESCRIBE DATABASE EMPLOYEE_151;
>USE EMPLOYEES_151;
> CREATE EXTERNAL TABLE IF NOT EXISTS EMPLOYEE_151(EMP_ID INT,EMP_NAME STRING,DESIGNATION STRING,SALARY FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T' LOCATION '/EMPLOYEE_INFO';
>DESCRIBE FORMATTED EMPLOYEE_151;
```

```
1 CREATE DATABASE IF NOT EXISTS EMPLOYEES_151 COMMENT 'EMPLOYEE Details' WITH DBPROPERTIES('creator'='Namratha');
2 SHOW DATABASES;
3 DESCRIBE DATABASE EMPLOYEES_151;
```

	db_name	comment	location	owner_name	owner_type	paramete
1	employees_151	EMPLOYEE Details	hdfs://namenode:8020/user/hive/warehouse/employees_151.db	root	USER	

```
4 USE EMPLOYEES_151;
5 CREATE EXTERNAL TABLE IF NOT EXISTS EMPLOYEE_151 (Emp_ID INT,Emp_Name STRING,Designation STRING,Salary FLOAT)
6   ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/EMPLOYEE_INFO';
7 DESCRIBE FORMATTED EMPLOYEE_151;
```

	col_name	data_type	comme
3	emp_id	int	
4	emp_name	string	
5	designation	string	
6	salary	float	
7		NULL	NULL
8	# Detailed Table Information	NULL	NULL
9	Database:	default	NULL
10	Owner:	root	NULL
11	CreateTime:	Sun Dec 20 04:38:18 UTC 2020	NULL
12	LastAccessTime:	UNKNOWN	NULL
13	Retention:	0	NULL
14	Location:	hdfs://namenode:8020/EMPLOYEE_INFO	NULL
15	Table Type:	EXTERNAL_TABLE	NULL
16	Table Parameters:	NULL	NULL
17		EXTERNAL	TRUE
18		transient_lastDdlTime	160843
19		NULL	NULL

2. Load data into table from a given file

```
>INSERT INTO TABLE EMPLOYEE_151
VALUES(1,’Nam’,’Manager’,1000000),(2,’Amy’,’Clerk’,50000),(3,’Pen
ny’,’Intern’,20000),(4,’Shelly’,’HR’,35000);
>SELECT * FROM EMPLOYEE_151;
```

```
LOAD DATA LOCAL INPATH '/home/lenovo/Desktop/Nam-BDA-LAB/LAB9/Employee.txt' OVERWRITE INTO TABLE EMPLOYEE_151;
INSERT INTO TABLE employee_151 VALUES(1,'Nam','Manager',1000000),(2,'Amy','Clerk',5000),(3,'Penny','Intern',20000),
(4,'Shelly','HR',35000);
SELECT * FROM employee_151;
```

Execute 5000

Query History | Saved Queries | Query Builder | Results

Grid | Chart | Columns | Search | Export | Expand

	employee_151.emp_id	employee_151.emp_name	employee_151.designation	employee_151.salary
1	1	Nam	Manager	1000000
2	2	Amy	Clerk	5000
3	3	Penny	Intern	20000
4	4	Shelly	HR	35000

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 EMPLOYEE_151 WHERE
SALARY>30000;
>SELECT * FROM EMPLOYEE_VIEW;
```

```
CREATE VIEW EMPLOYEE_VIEW AS SELECT * FROM employee_151 WHERE Salary>30000;
SELECT * FROM EMPLOYEE_VIEW;
```

	employee_view.emp_id	employee_view.emp_name	employee_view.designation	employee_view.salary
1	1	Nam	Manager	1000000
2	4	Shelly	HR	35000

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 EMPLOYEE_151 ADD COLUMNS (DEPT_ID INT);
>DESCRIBE FROMATTED EMPLOYEE_151;
> INSERT INTO TABLE EMPLOYEE_151
VALUES(1,'Nam','Manager',10000000,1),(2,'Amy','Clerk',50000,2),(3
,'Penny','Intern',20000,3),(4,'Shelly','HR',35000,3);
>SELECT * FROM EMPLOYEE_151;
```

```
ALTER TABLE EMPLOYEE_151 ADD COLUMNS (Dept_ID INT);
DESCRIBE FORMATTED EMPLOYEE_151;
```

	col_name	data_type	comme
1	# col_name	data_type	comme
2		NULL	NULL
3	emp_id	int	
4	emp_name	string	
5	designation	string	
6	salary	float	
7	dept_id	int	
8		NULL	NULL
9	# Detailed Table Information	NULL	NULL
10	Database:	default	NULL
11	Owner:	root	NULL
12	CreateTime:	Sun Dec 20 04:38:18 UTC 2020	NULL
13	LastAccessTime:	UNKNOWN	NULL
14	Retention:	0	NULL
15	Location:	hdfs://namenode:8020/EMPLOYEE_INFO	NULL
16	Table Type:	EXTERNAL_TABLE	NULL

```
INSERT INTO TABLE EMPLOYEE_151 VALUES(1,'Nam','Manager',10000000,1),(2,'Amy','Clerk',50000,2),(3,'Penny','Intern',20000,3),(4,'Shelly','HR',35000,3);
SELECT * FROM EMPLOYEE_151;
```

1	Nam	Manager	10000000	1
2	Amy	Clerk	50000	2
3	Penny	Intern	20000	3
4	Shelly	HR	35000	3

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 EMPLOYEE_151 WHERE SALARY >
30000 GROUP BY DEPT_ID;
```

```
SELECT DEPT_ID, COUNT(DEPT_ID) FROM EMPLOYEE_151 WHERE SALARY > 30000 GROUP BY DEPT_ID;
```

dept_id	_c1
NULL	0
1	1
2	1
3	1

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

```
CREATE EXTERNAL TABLE IF NOT EXISTS DEPARTMENT_151(DEPT_ID INT,
DEPT_NAME STRING, EMP_ID INT) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\T' LOCATION '/DEPARTMENT';
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS DEPARTMENT_151(DEPT_ID INT, DEPT_NAME STRING, EMP_ID INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T' LOCATION '/DEPARTMENT';
DESCRIBE FORMATTED DEPARTMENT_151;
```

col_name	data_type	comment
# col_name	data_type	comment
	NULL	NULL
dept_id	int	
dept_name	string	
emp_id	int	

```
INSERT INTO TABLE DEPARTMENT_151 VALUES(1,'Management',1),(2,'Finance',2),(3,'HR',3),(3,'HR',4);
SELECT * FROM DEPARTMENT_151;
```

	department_151.dept_id	department_151.dept_name	department_151.emp_id
1	1	Management	1
2	2	Finance	2
3	3	HR	3
4	3	HR	4

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

```
SELECT * FROM EMPLOYEE_151 JOIN DEPARTMENT_151 ON
EMPLOYEE_151.DEPT_ID = DEPARTMENT_151.DEPT_ID;
```

```
SELECT * FROM EMPLOYEE_151 JOIN DEPARTMENT_151 ON EMPLOYEE_151.DEPT_ID = DEPARTMENT_151.DEPT_ID;
```

employee_151.emp_id	employee_151.emp_name	employee_151.designation	employee_151.salary	employee_151.dept_id	department_151.dept_id	department_151.dept_name	department_151.emp_id
1	Nam	Manager	10000000	1	1	Management	1
2	Amy	Clerk	50000	2	2	Finance	2
3	Penny	Intern	20000	3	3	HR	3
4	Shelly	HR	35000	3	3	HR	4