Name: Pushpa Yadav
Roll No: 46

Experiment 8 – Querying, Sorting, Aggregating data using HiveQL

What is HIVE:

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive:

These are the following features of Hive:

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

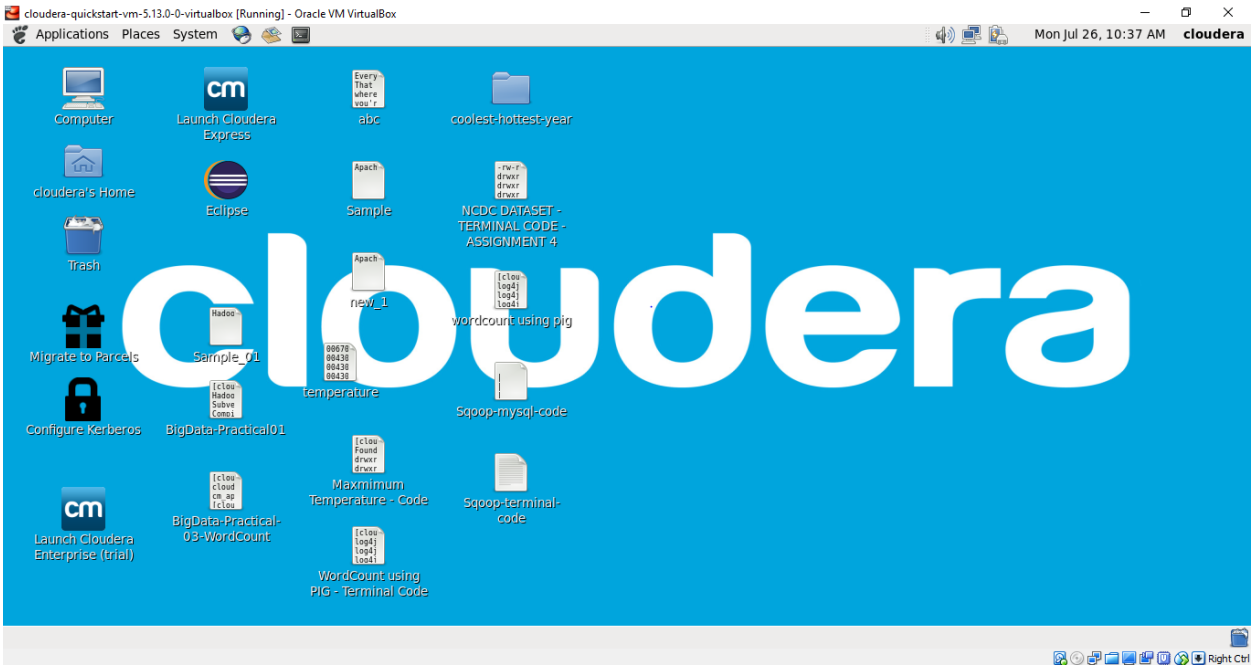
Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.

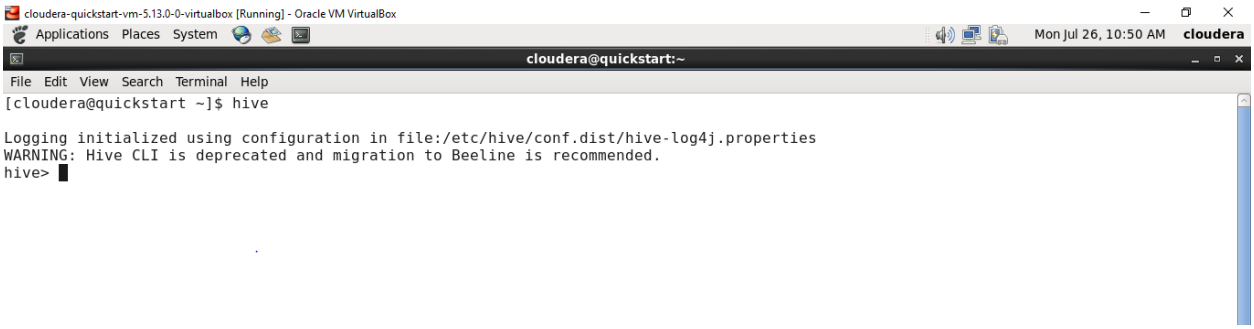
- Hive queries contain high latency.

Steps:Querying, Sorting, Aggregating data using HiveQL

1. Open the cloudera.



2. Open the terminal, Now we use **hive** command to enter the **hive shell prompt** and in hive shell we could execute all of the hive commands.



3. Now we will see the databases which are already existing using below command.
   Show databases;

```
hive> show databases;
OK
default
Time taken: 14.9 seconds, Fetched: 1 row(s)
hive>
```

4. If we want to drop the database with the entire data (rows) then we will use below command. Here we don't have any existing database rather than default so if for example I have 'office' as a database then will drop the database along with the data using command as,
   drop database office cascade;

5. Creating a database name 'RJC' using below command.
   Create database RJC;

```
hive> create database RJC;
OK
Time taken: 76.014 seconds
hive>
```

6. So if we want to see whether this RJC database is created or not we will use below command,
   show databases;

```
hive> show databases;
OK
default
rjc
Time taken: 8.867 seconds, Fetched: 2 row(s)
hive>
```

7. Now we want to check whether we have any tables inside this rjc database or not. So first we will move to this databse rjc using below command,
   Use rjc;

```
hive> use rjc;
OK
Time taken: 1.134 seconds
hive>
```

Now we have moved inside this rjc database. Now we will check out which are the tables available using below command,

show tables;

```
hive> show tables;
OK
Time taken: 2.067 seconds
hive>
```

So as we can see from the above output it give us OK as output because there are no tables created inside this rjc database. So as there are no tables we did not get anything in the output we simply got as OK.

No will drop this 'rjc' database using below command we do not mention here cascade because there are no tables in this and so there are no data available or present inside this rjc database.

drop database rjc;

and to check whether the data base dropped or not using below command,

show databases;

```
hive> drop database rjc;
OK
Time taken: 124.177 seconds
hive> show databases;
OK
default
Time taken: 0.264 seconds, Fetched: 1 row(s)
hive>
```

8. Creating a database **"rjc"**.

   create database rjc;

```
hive> create database rjc;
OK
Time taken: 2.437 seconds
hive>
```

Now let's move to this database using below command so now all the work we will do or perform it should be done within this **rjc** database.

userjc;

```
hive> use rjc;
OK
Time taken: 1.747 seconds
hive>
```

9. Now we will create a table inside this rjc database named as **employee** using below command,

   create table employee(ID int, name string, salary float, age int)

   After this we will not put semicolon , When we will be loading the data from some existing csv file or maybe some other text files so we have to mention that how that data has to be loaded here. We are simply creating the schema of the table with some certain fields or attributes along with their datatypes and then I'm mentioning

   - **row format delimited**
   - **fields terminated by ',';**

```
hive> create table employee(ID int, name string, salary float, age int)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 29.742 seconds
hive>
```

row format delimited means , every record is present in one row and

fields terminated by ',' and fields are terminated by comma. So as soon as it encounters one comma so that means that one is the value of some field and after comma it is encountering abc so that abc is the value of some another field. By default it is a "tab" character that means fields are separated by "tab".

10. So now we will see the schema of the table using below command,

    describe employee;

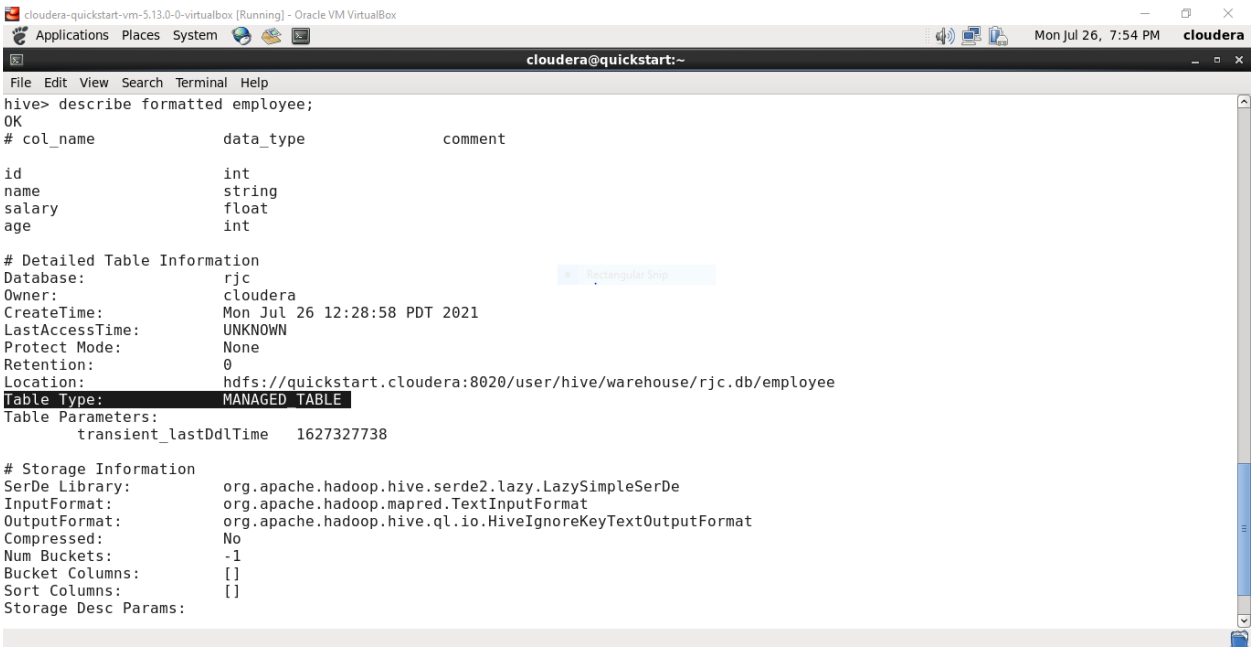
    It will give different fields of table employee along with their respective datatypes.

```
hive> describe employee;
OK
id                      int
name                    string
salary                  float
age                     int
Time taken: 34.438 seconds, Fetched: 4 row(s)
hive>
```

11. By default the internal table would store in the warehouse directory of hive. Whereas the external tables are available in the hdfs. And if we drop the internal table so then the table data and the metadata associated with that table will be deleted from the hdfs. Whereas when we drop the external table then only the metadata associated with that table will be deleted whereas the table data will be untouched by hive as it would be residing in the hdfs and it would be outside the warehouse directory of the hive.

    So Now we will check how the table which we will be created is internal table or external table using below command,

    describe formatted employee;



    By default hive createsInternal table or Managed Table.

12. Now we will create the external table using below command,

    create external table emloyee2 (ID int, name string, salary float, age int)

- **row format delimited**
- **fields terminated by ‘,’**
- **stored as textfile;**

```
hive> create external table employee2 (ID int, name string, salary float, age int)
    > row format delimited
    > fields terminated by ','
    > stored as textfile;
OK
Time taken: 29.371 seconds
hive>
```

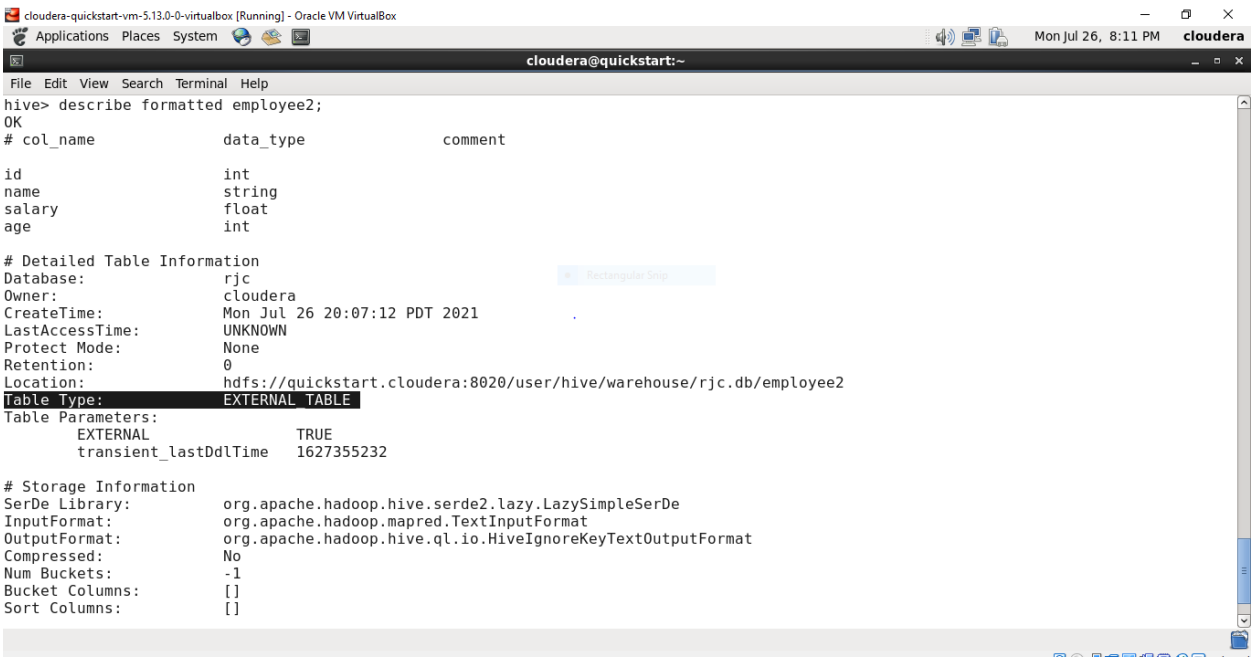
13. Checking the schema of the table using below command,

    describe employee2;

```
hive> describe employee2;
OK
id                      int
name                    string
salary                  float
age                     int
Time taken: 3.693 seconds, Fetched: 4 row(s)
hive>
```

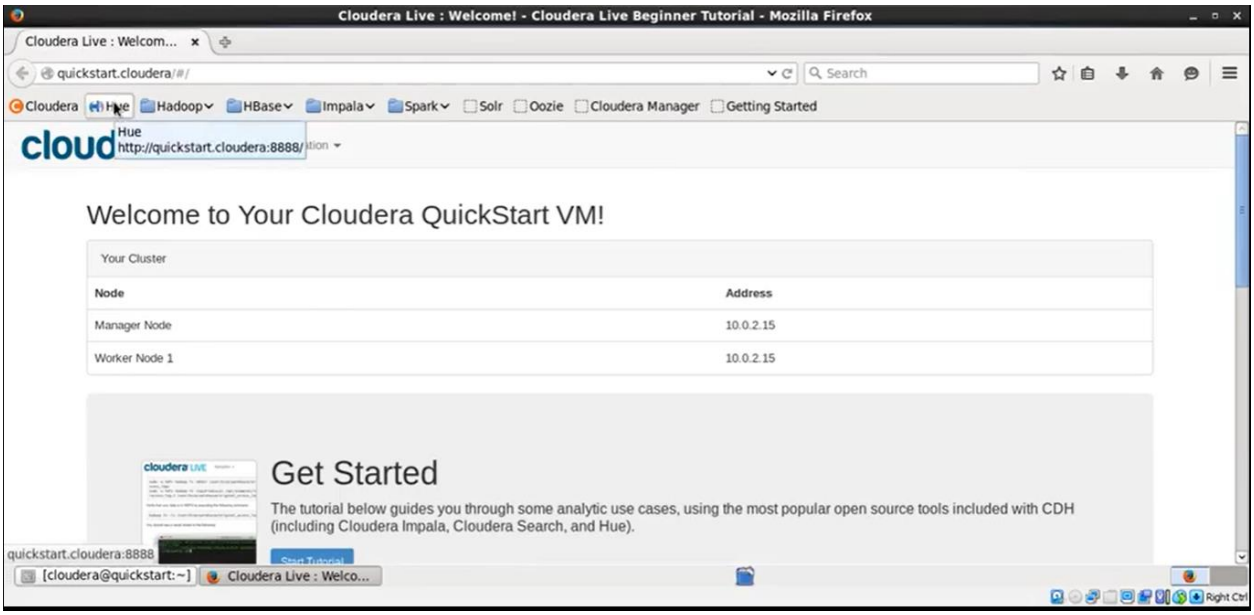
14. So Now we will check how the table which we will be created is internal table or external table using below command,

    describe formatted employee2;

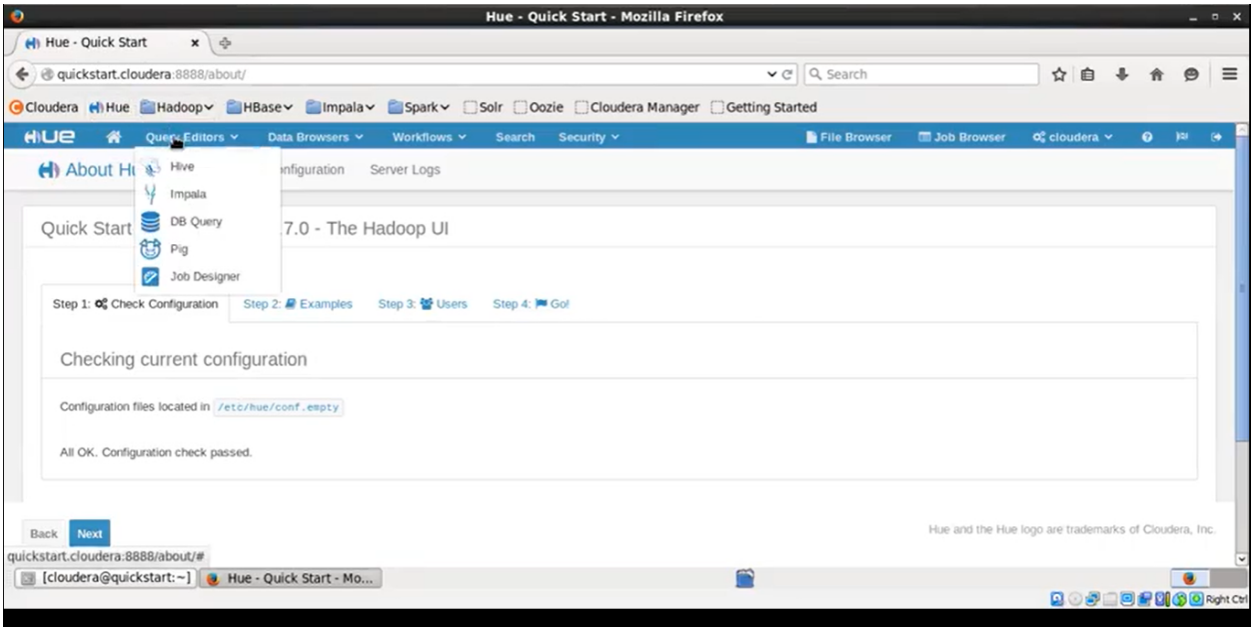


It is an External table.

15. So whatever we have done in terminal the same thing we can see in the browser as well.
    Open the browser → click on Hue



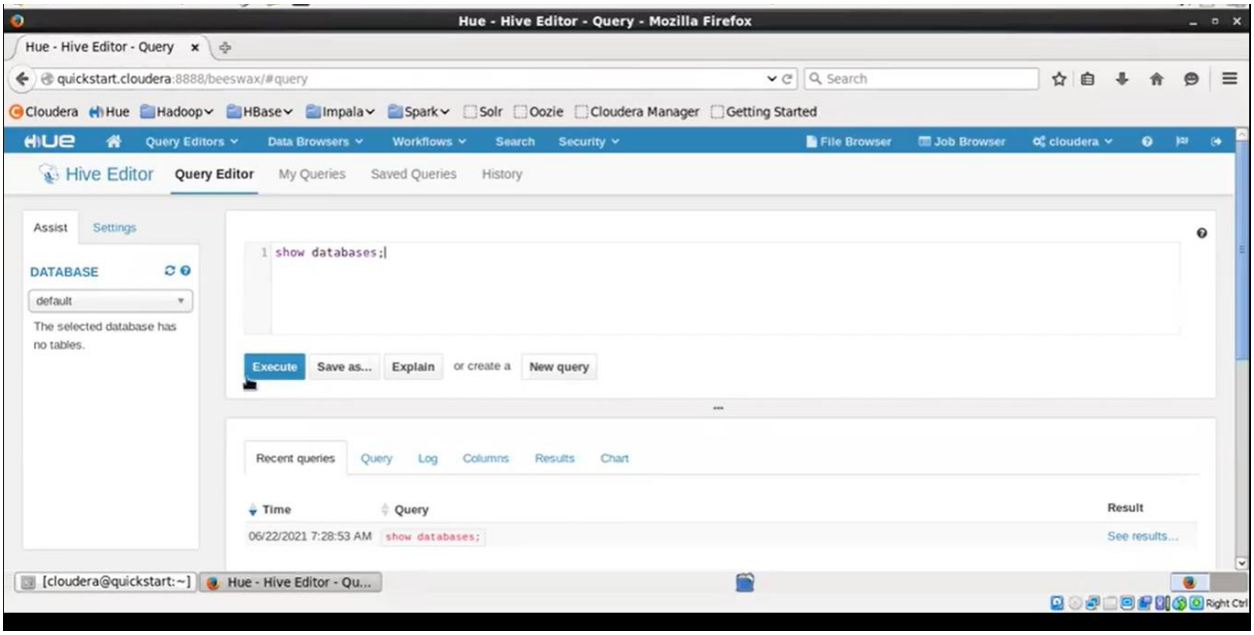
Then click on Query Editor → select Hive

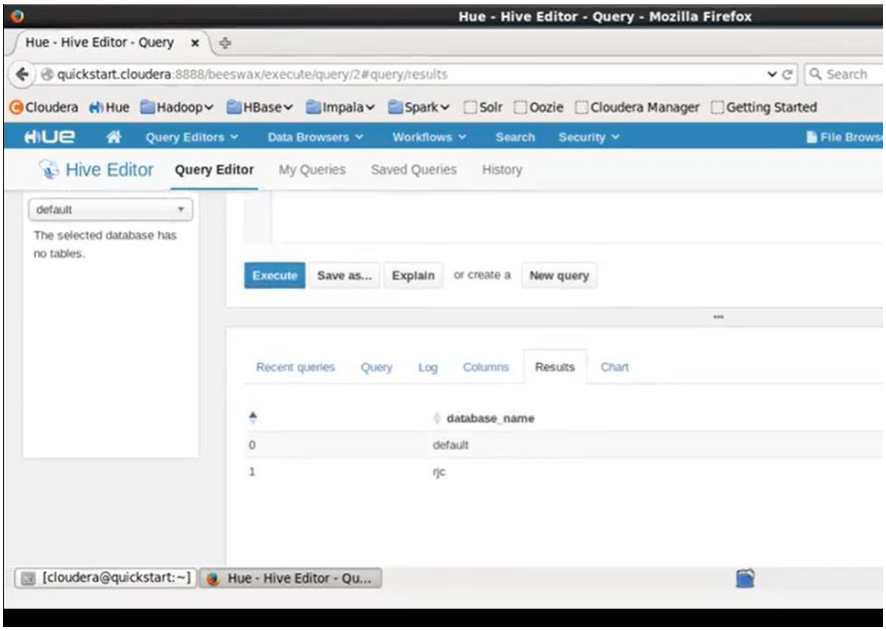


Write the query in query editor. Here we are showing the databases by using below command,

show databases;

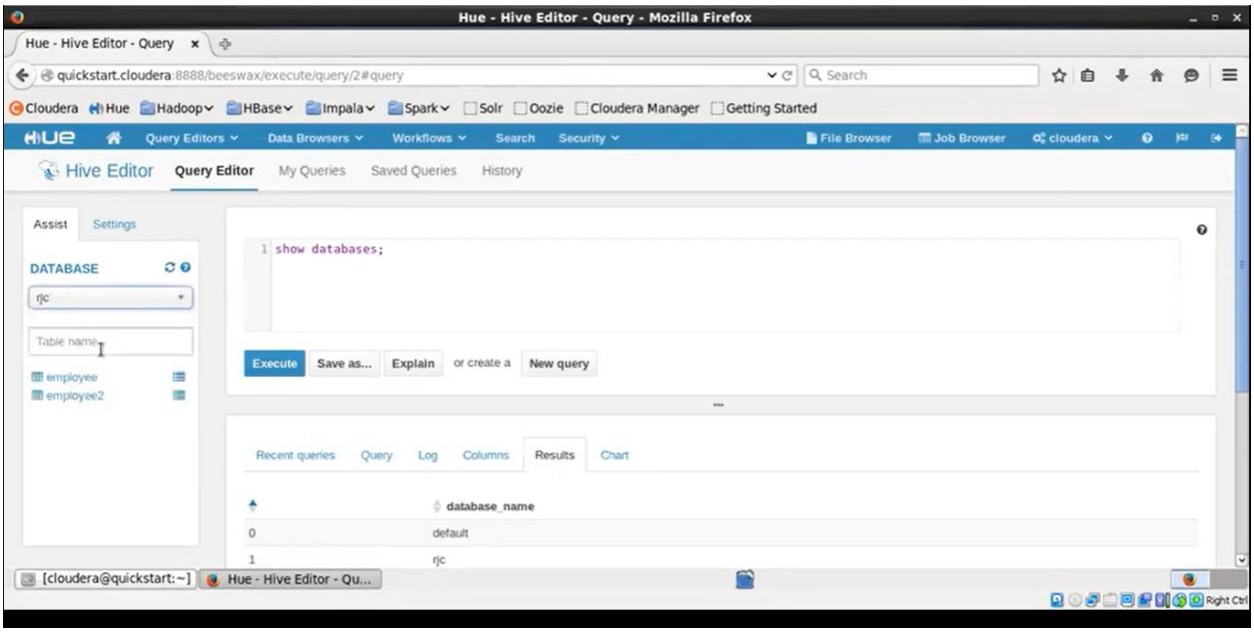
It will give the list of databases which are present.



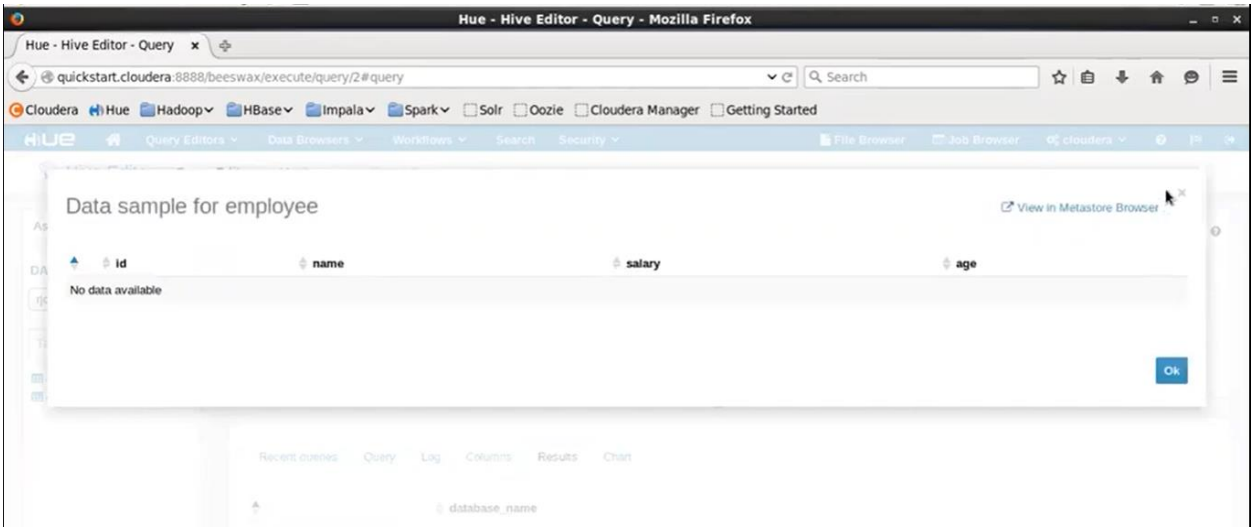


There are only two databases present i.e. default and rjc which we created earlier.

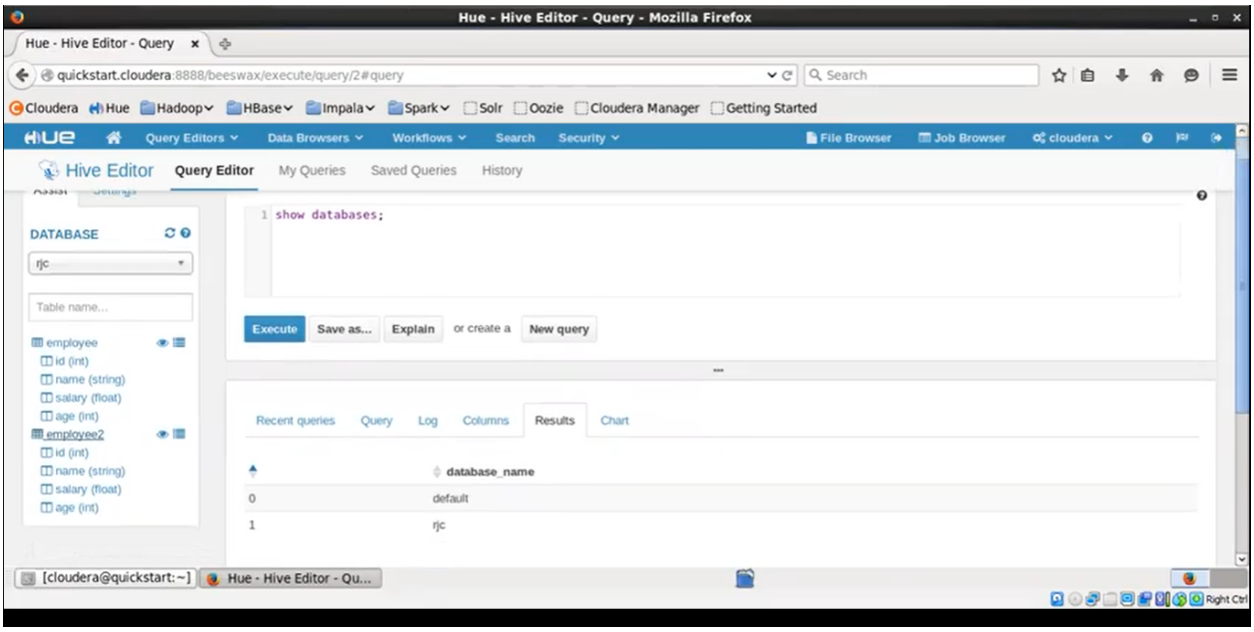
We can also see the list of databases in left side corner. And after clicking on the database name here it is “rjc” it will give the list of all tables present inside “rjc” database.



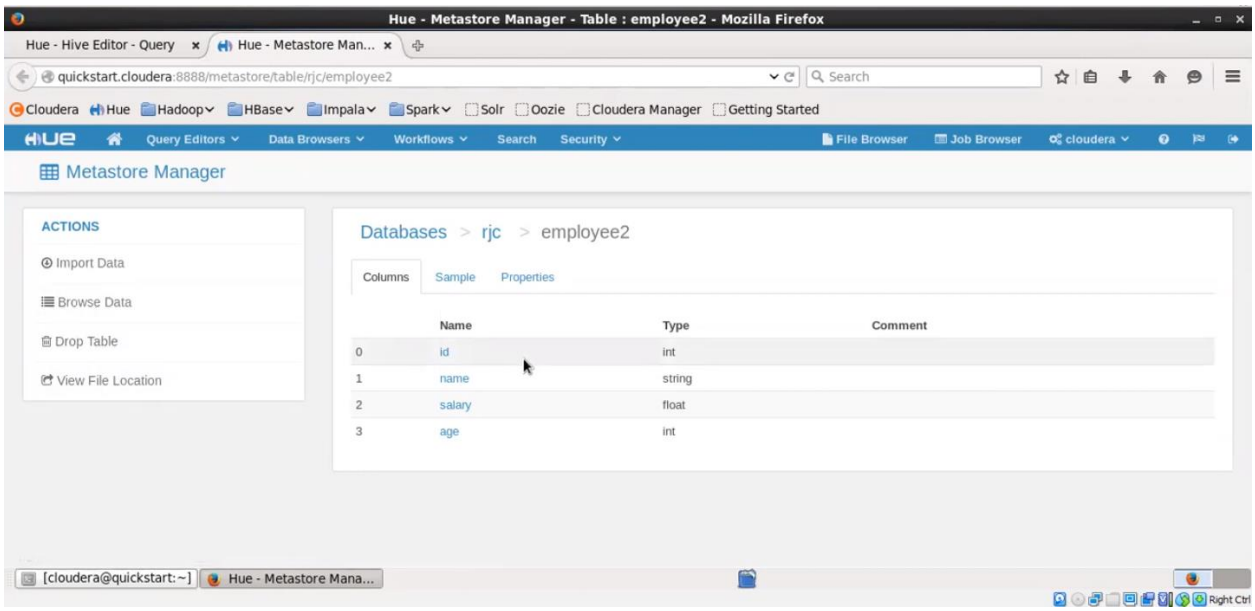
We can also Preview the sample data. Here it is showing blank because we have not inserted anything or data inside this employee table.



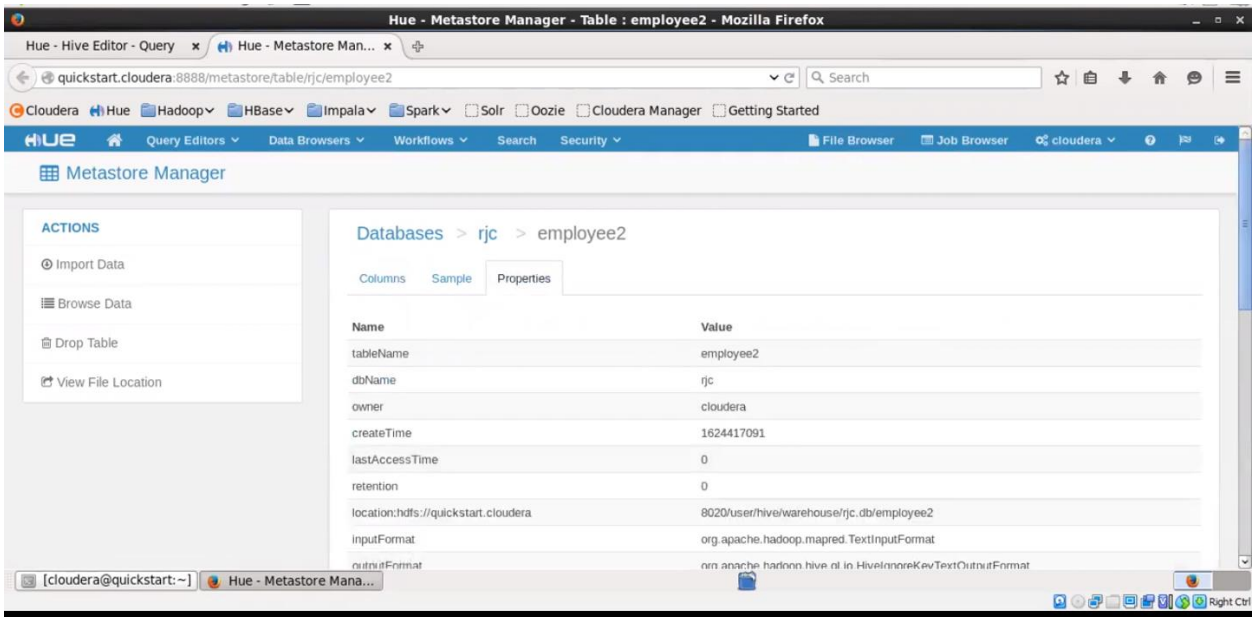
When we click on employee table it will give the fields of employee table.



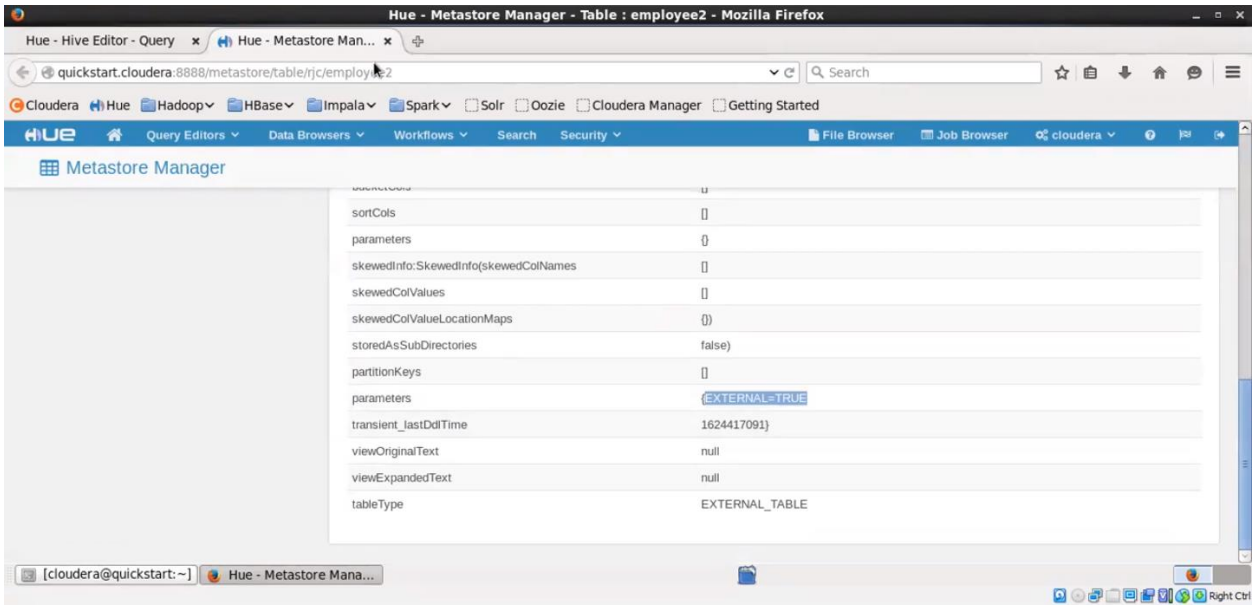
Now click on employee2 table which is an external table then click on eye type icon it will give the employee2 table Metastore information like fields and their respective datatypes.



And click on properties it will give the properties of employee2 table.



here we can also see that the External is set to TRUE.



16. Creating a new external table named as employee3 in the specific location using below command,

create external table emloyee3 (ID int, name string, salary float, age int)

- **row format delimited**
- **fields terminated by ‘,’**
- **location '/user/cloudera/vj';**

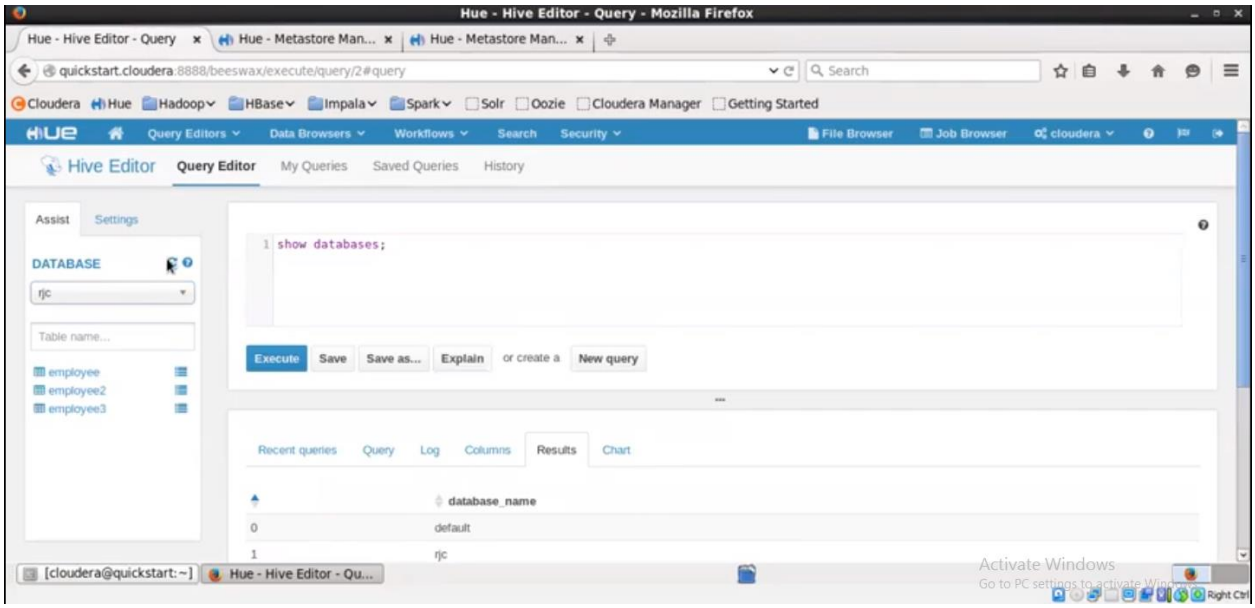
It will first create 'vj' directory inside the /user/cloudera and then inside 'vj' the employee3 table get stored.

```
hive> create external table employee3(ID int, name string, salary float, age int
)
    > row format delimited
    > fields terminated by ','
    > location '/user/cloudera/vj';
OK
Time taken: 0.159 seconds
```

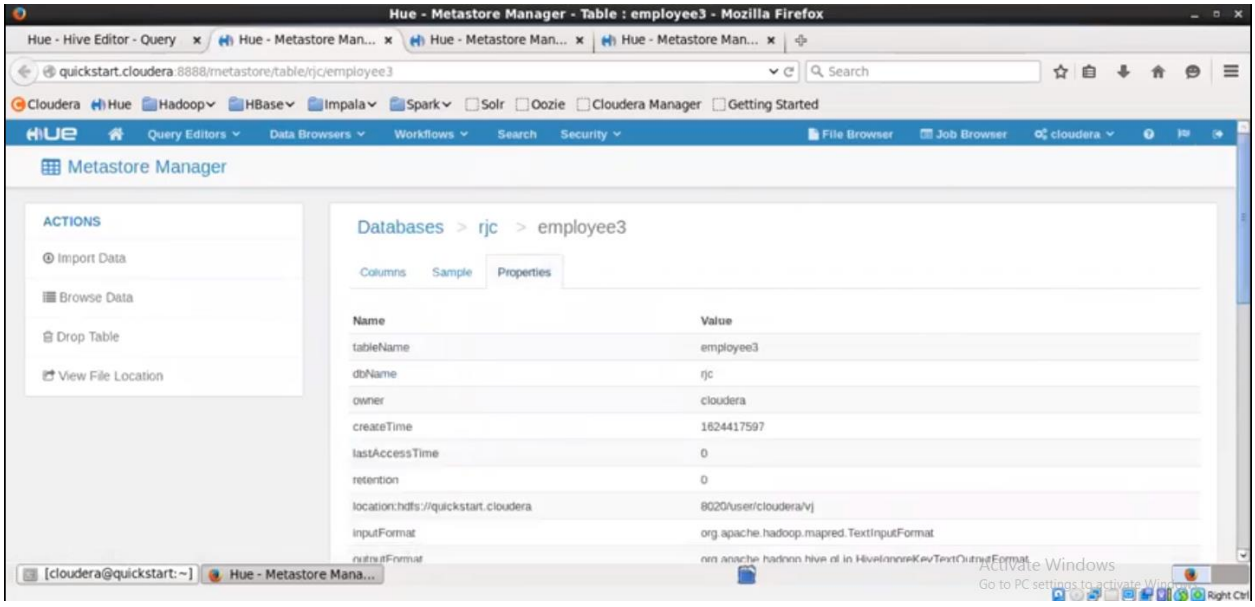
17. To see the schema of the employee3 table we use below command,
describe employee3;

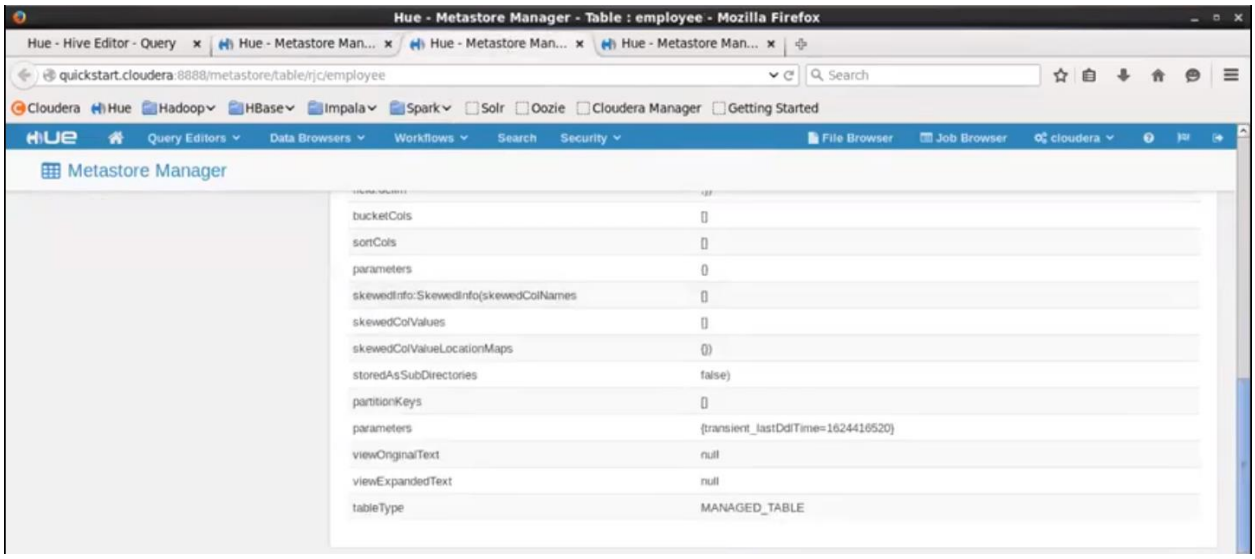
```
hive> describe employee3;
OK
id                      int
name                    string
salary                  float
age                     int
Time taken: 0.128 seconds, Fetched: 4 row(s)
hive>
```

18. Then switch to browser and Refresh and select the rjc database and refresh, Now it will display the employee3 table along with other two tables which we have created earlier.



Here we will see the properties of employee3 table here we can also see the location of the table where it is stored.





19. Now move to terminal and listing out all the tables using below command;
show tables;

```
hive> show tables;
OK
employee
employee2
employee3
Time taken: 0.023 seconds, Fetched: 3 row(s)
hive>
```

20. ALTER COMMANDS

Now we are changing the name of the **employee3** table to **emptable** using below command,

```
hive> alter table employee3 RENAME TO emptable;
OK
Time taken: 0.231 seconds
hive>
```

21. Now we will check whether the name of the **employee3** table changes to **emptable** or not using below command,
show tables;

```
hive> show tables;
OK
employee
employee2
emptable
Time taken: 0.018 seconds, Fetched: 3 row(s)
hive>
```

22. First we will see the fields of emptable then we will add new column as **surname** in emptable using below command,
describe emptable;
Alter table emptable add columns (surname string);
describe emptable;

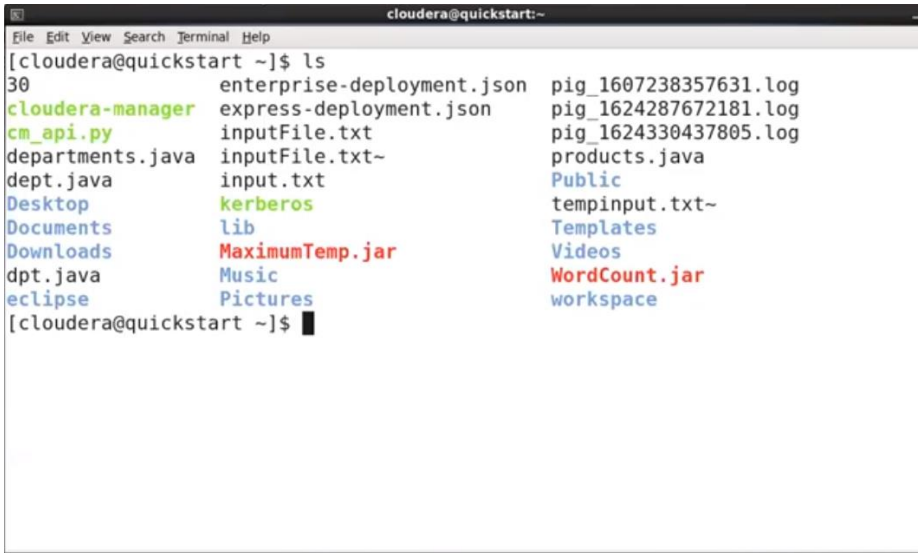
```
hive> describe emptable;
OK
id                      int
name                    string
salary                  float
age                     int
Time taken: 0.098 seconds, Fetched: 4 row(s)
hive> Alter table emptable add columns(surname string);
OK
Time taken: 0.181 seconds
hive> describe emptable;
OK
id                      int
name                    string
salary                  float
age                     int
surname                 string
Time taken: 0.098 seconds, Fetched: 5 row(s)
hive>
```

23. Now we will change field name of the emptable to first_name using alter command,
Alter table emptable change name first_name string;
describe emptable;

```
hive> alter table emptable change name first_name string;
OK
Time taken: 0.242 seconds
hive> describe emptable;
OK
id                      int
first_name              string
salary                  float
age                     int
surname                 string
Time taken: 0.094 seconds, Fetched: 5 row(s)
hive>
```

Loading the data in the table

24. Before loading the data in the table we will first create the csv file. Now open the new terminal , using **ls** command list out all the directories **-->** change the directory to document directory **-->** use **ls** command to list all the files present inside the document folder or directory

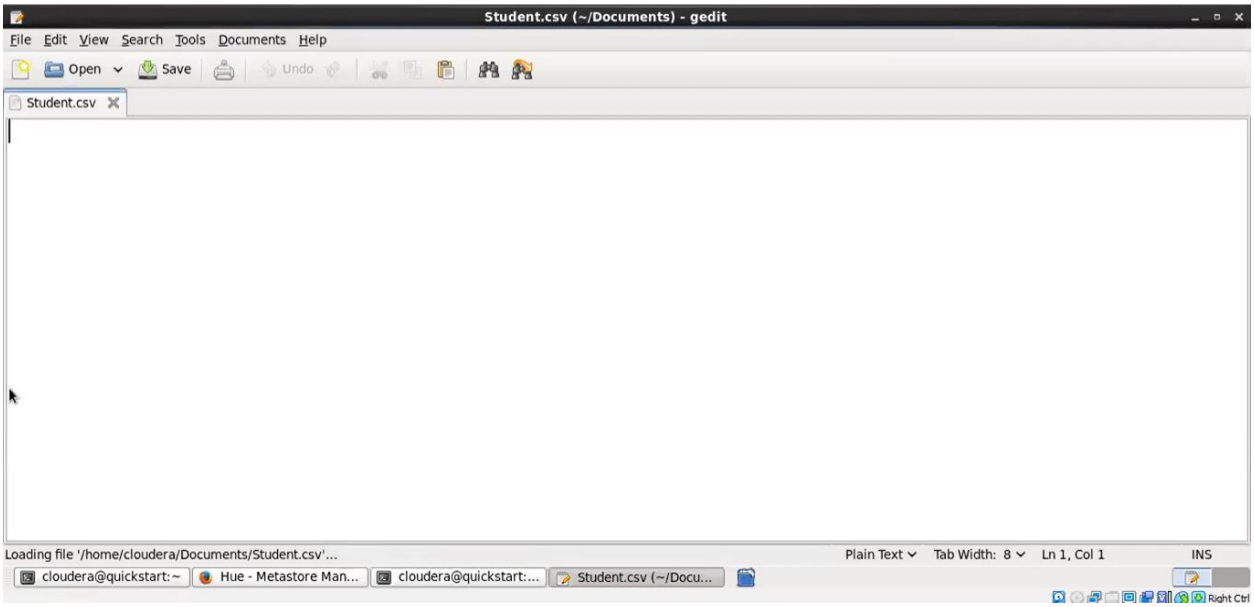



```
[cloudera@quickstart ~]$ cd Documents
[cloudera@quickstart Documents]$ ls
cloudera-manager.html  Customer.csv   employee.csv
```

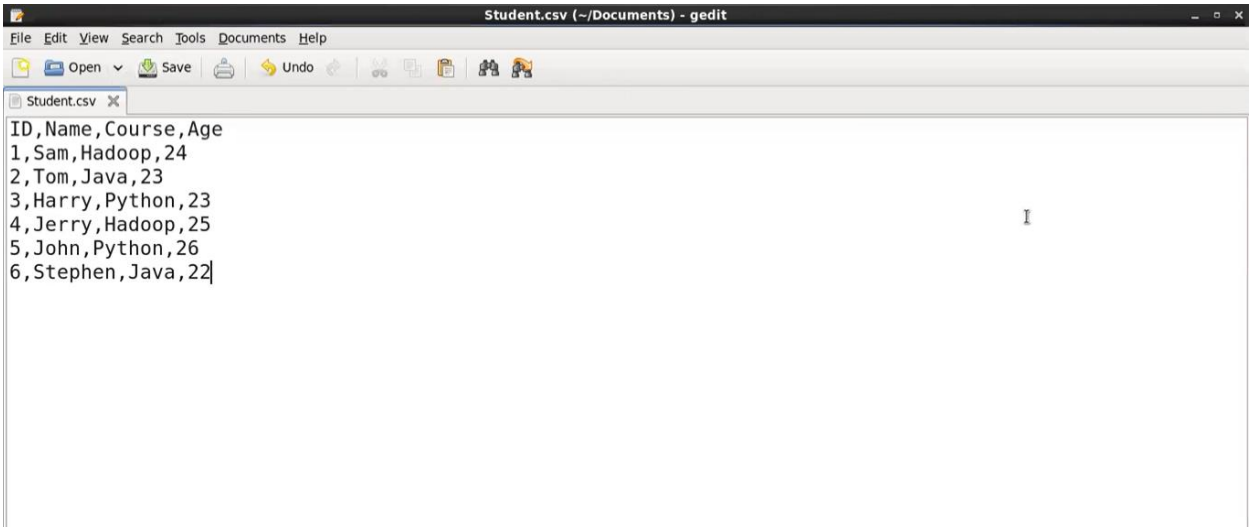
25. Now creating new file as Student.csv using below command,
gedit Student.csv

```
[cloudera@quickstart Documents]$ gedit Student.csv
```

As soon as we hit enter it will create a Student.csv file as it was not existing earlier.



Now we are populating the Student.csv file with some data and save this file.



26. Creating a new database as **rjcstudent.**

create database rjcstudent;

show databases;

```
hive> create database rjcstudent;
OK
Time taken: 0.056 seconds
hive> show databases;
OK
default
rjc
rjcstudent
Time taken: 0.01 seconds, Fetched: 3 row(s)
```

27. Using rjcstudent database.
use rjcstudent;

```
hive> use rjcstudent;
OK
Time taken: 0.028 seconds
hive>
```

28. Creating new table **student** inside rjcstudent database.
create table student (ID int, Name string, Age int)

- **partitioned by(Course string)**
- **row format delimited**
- **fields terminated by ‘,’;**

```
hive> create table student (ID int,Name string,Age int)
    > partitioned by(Course string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.178 seconds
hive>
```

29. To see the structure or schema of the table,
describe student;

```
hive> describe student;
OK
id                      int
name                    string
age                     int
course                  string

# Partition Information
# col_name              data_type               comment

course                  string
Time taken: 0.099 seconds, Fetched: 9 row(s)
```

30. Loading data in the student table from Student.csv file which we have created in document directory. Here we are partitioning based on course = 'Hadoop'.

load data local inpath '/home/cloudera/Documents/Student.csv' into table student

- **partitioned by(Course string)**
- **row format delimited**

```
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table s
tudent
    > partition(Course="Hadoop")
    > ;
Loading data to table rjcstudent.student partition (course=Hadoop)
Partition rjcstudent.student{course=Hadoop} stats: [numFiles=1, numRows=0, total
Size=120, rawDataSize=0]
OK
Time taken: 1.204 seconds
```

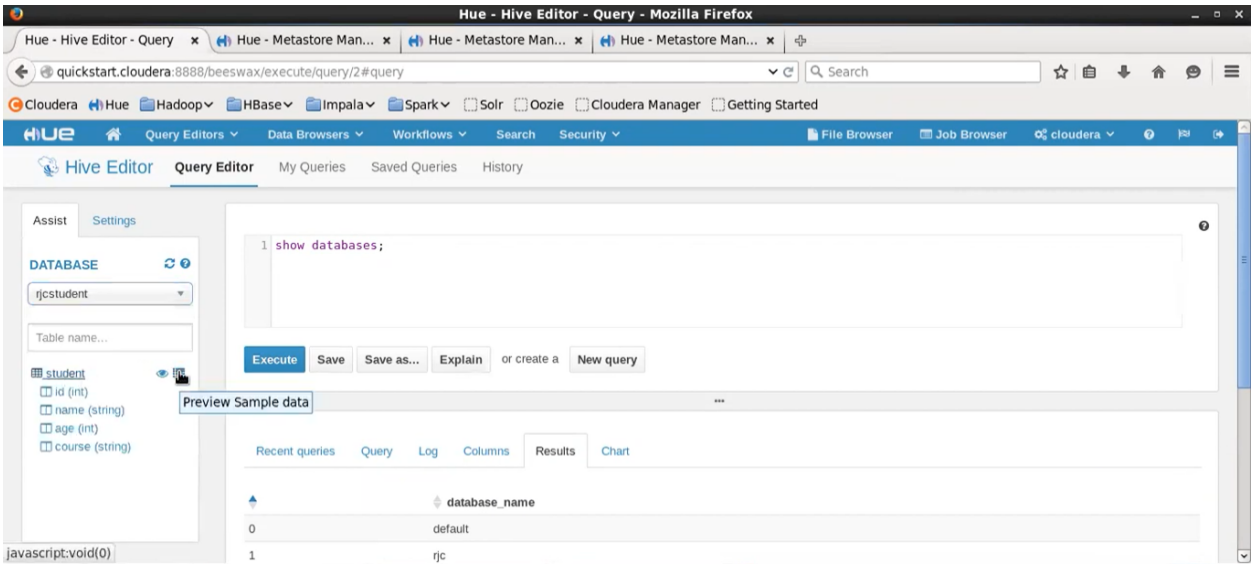
It is partitioning based on Hadoop.
select * from student;

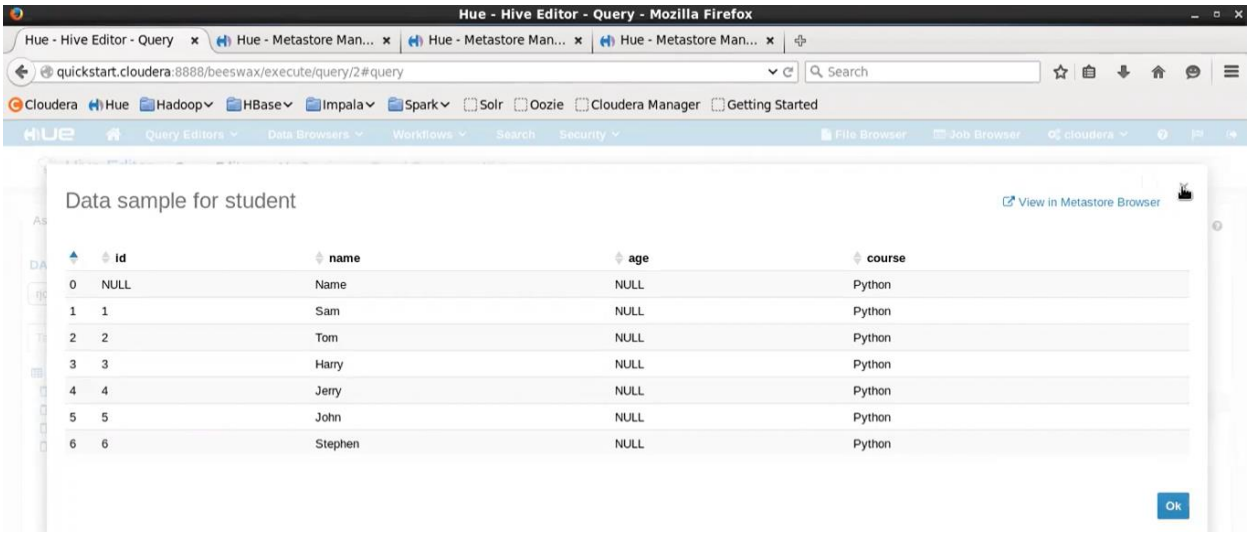
```
hive> select * from student;
OK
NULL    Name    NULL    Hadoop
1       Sam     NULL    Hadoop
2       Tom     NULL    Hadoop
3       Harry   NULL    Hadoop
4       Jerry   NULL    Hadoop
5       John    NULL    Hadoop
6       Stephen NULL    Hadoop
Time taken: 0.447 seconds, Fetched: 7 row(s)
hive>
```

31. Now we similarly partition for course = Java and course = Python.

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table s
tudent
    > partition(Course="Java");
Loading data to table rjcstudent.student partition (course=Java)
Partition rjcstudent.student{course=Java} stats: [numFiles=1, numRows=0, totalSi
ze=120, rawDataSize=0]
OK
Time taken: 0.608 seconds
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table s
tudent
    > partition(Course="Python");
Loading data to table rjcstudent.student partition (course=Python)
Partition rjcstudent.student{course=Python} stats: [numFiles=1, numRows=0, total
Size=120, rawDataSize=0]
OK
Time taken: 0.619 seconds
hive>
```

32. Now go to browser refresh the page and select database as rjcstudent and click in preview student table.





33. Now dropping the table student and creating the student table again normally (i.e. without partitioning).

drop table student;

create table student (ID int, Name String, Course string, Age int)

- **row format delimited**
- **fields terminated by ‘,’**
- **tblproperties(“skip.header.line.count” =”1”);**

```
hive> create table student(ID int, Name String, Course string, Age int)
    > row format delimited
    > fields terminated by ','
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.121 seconds
hive>
```

34. Loading data in the student table from Student.csv file which present inside /home/cloudera/Documents directory.

load data local inpath '/home/cloudera/Documents/Student.csv' into table student

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> load data local inpath '/user/cloudera/Documents/Student.csv' into table s
tudent;
FAILED: SemanticException Line 1:23 Invalid path ''/user/cloudera/Documents/Stud
ent.csv'': No files matching path file:/user/cloudera/Documents/Student.csv
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table s
tudent;
Loading data to table rjcstudent.student
Table rjcstudent.student stats: [numFiles=1, totalSize=120]
OK
Time taken: 0.445 seconds
```

select * from student;

```
hive> select * from student;
OK
1       Sam     Hadoop  24
2       Tom     Java    23
3       Harry   Python  23
4       Jerry   Hadoop  25
5       John    Python  26
6       Stephen Java    22
Time taken: 0.166 seconds, Fetched: 6 row(s)
hive>
```

35. Now creating **employee.csv f**ile**.**

gedit employee.csv

```
[cloudera@quickstart Documents]$ gedit employee.csv
```

And populating it with some data.

```
employee.csv (~/Documents) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
employee.csv
Id,Name,Dept,Yoj,Salary
1,Rose,IT,2012,26000
2,Sam,Sales,2012,22000
3,Mike,HR,2013,30000
4,Nick,SC,2013,20000
```

Querying :

36. Creating new database for performing querying operations.

Create database hiveql;

```
hiveql' in ddl statement
hive> create database hiveql;
OK
Time taken: 0.041 seconds
hive>
```

37. Using database **hiveql** and creating table **employee** inside the hiveql datanase.
create table employee(ID int, Name string, Department string, YOJ int, Salary float)

- **row format delimited**
- **fields terminated by ‘,’;**
- **tblproperties(“skip.header.line.count” =”1”);**

```
hive> create table employee(ID int, Name string, Department string, YOJ int, Sal
ary float)
    > row format delimited
    > fields terminated by ','
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.081 seconds
hive>
```

Now we will see the schema of employee table using below command,

describe employee;

```
hive> describe employee;
OK
id                      int
name                    string
department              string
yoj                     int
salary                  float
Time taken: 0.122 seconds, Fetched: 5 row(s)
```

38. Loading the data into **employee table** from **employee.csv file** which we have created earlier and it is present in /home/cloudera/Documents directory.

load data local inpath ‘/home/cloudera/Documents/employee.csv’ into table employee

Displaying the table using below command,

select * from employee;

```
hive> load data local inpath '/home/cloudera/Documents/employee.csv' into table
employee;
Loading data to table hiveql.employee
Table hiveql.employee stats: [numFiles=1, totalSize=110]
OK
Time taken: 0.264 seconds
hive> select * from employee;
OK
1       Rose    IT      2012    26000.0
2       Sam     Sales   2012    22000.0
3       Mike    HR      2013    30000.0
4       Nick    SC      2013    20000.0
Time taken: 0.071 seconds, Fetched: 4 row(s)
hive>
```

Now we Perform some operations on this employee table.

39. We are printing or displaying the rows or records of employees whose salary is greater than and equal to 25000.

select * from employee where salary >=25000;

```
hive> select * from employee where salary>=25000;
OK
1       Rose    IT      2012    26000.0
3       Mike    HR      2013    30000.0
Time taken: 0.436 seconds, Fetched: 2 row(s)
```

The Rose and Mike are two employees whose salaries are greater than and equal to 25000.

40. Now we are printing or displaying the rows or records of employees whose salary is less than 25000.

select * from employee where salary <25000;

```
hive> select * from employee where salary<25000;
OK
2       Sam     Sales   2012    22000.0
4       Nick    SC      2013    20000.0
Time taken: 0.123 seconds, Fetched: 2 row(s)
hive>
```

Aggregating

41. Arithmetic operations:

We are adding 5000 in existing salary and displaying specific columns using below command,

select ID, name, salary + 5000 from employee;

```
hive> select ID, name, salary + 5000 from employee;
OK
1       Rose    31000.0
2       Sam     27000.0
3       Mike    35000.0
4       Nick    25000.0
Time taken: 0.166 seconds, Fetched: 4 row(s)
hive>
```

42. Displaying the maximum salary using below command,

select max(salary) from employee;

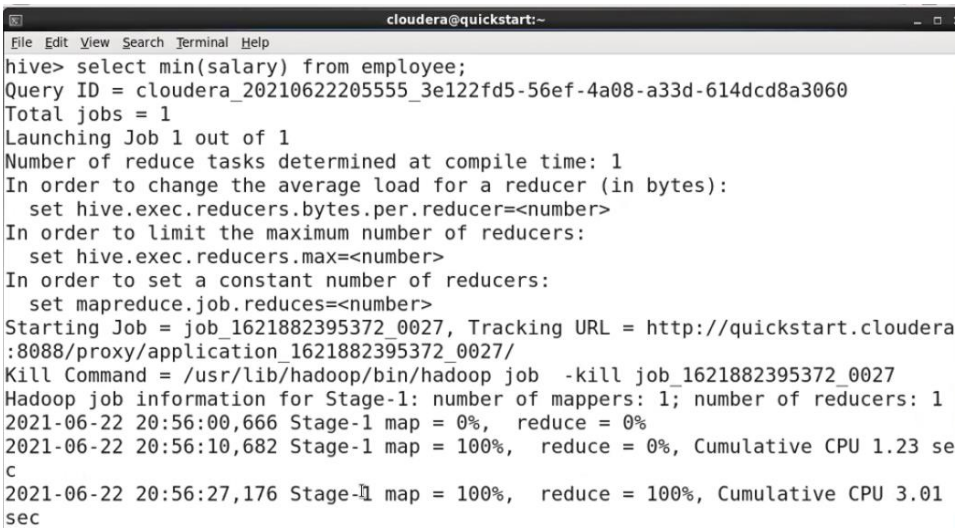
```
hive> select max(salary) from employee;
Query ID = cloudera_20210622205454_5967b01d-712a-46a2-a00e-701804626fc3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<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>
Starting Job = job_1621882395372_0026, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0026/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0026
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-06-22 20:54:44,412 Stage-1 map = 0%,  reduce = 0%
2021-06-22 20:55:00,043 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.3 sec
2021-06-22 20:55:16,926 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.6 s
ec
```

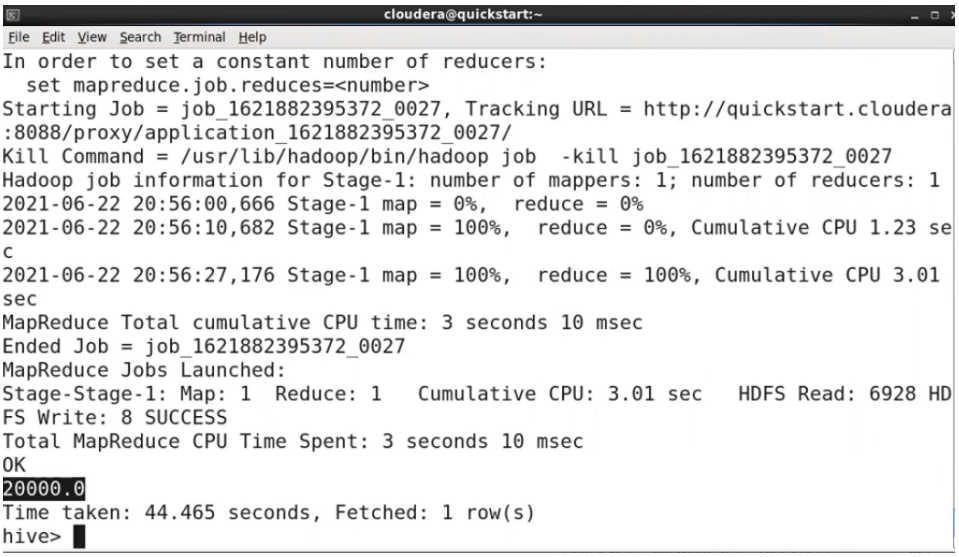
```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1621882395372_0026, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0026/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0026
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-06-22 20:54:44,412 Stage-1 map = 0%,  reduce = 0%
2021-06-22 20:55:00,043 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.3 sec
2021-06-22 20:55:16,926 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.6 s
ec
MapReduce Total cumulative CPU time: 3 seconds 600 msec
Ended Job = job_1621882395372_0026
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.6 sec   HDFS Read: 6908 HDF
S Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 600 msec
OK
30000.0
Time taken: 51.738 seconds, Fetched: 1 row(s)
hive>
```

Here we can see that 30000 is maximum salary.

43. Displaying the minimum salary using below command,

select min(salary) from employee;

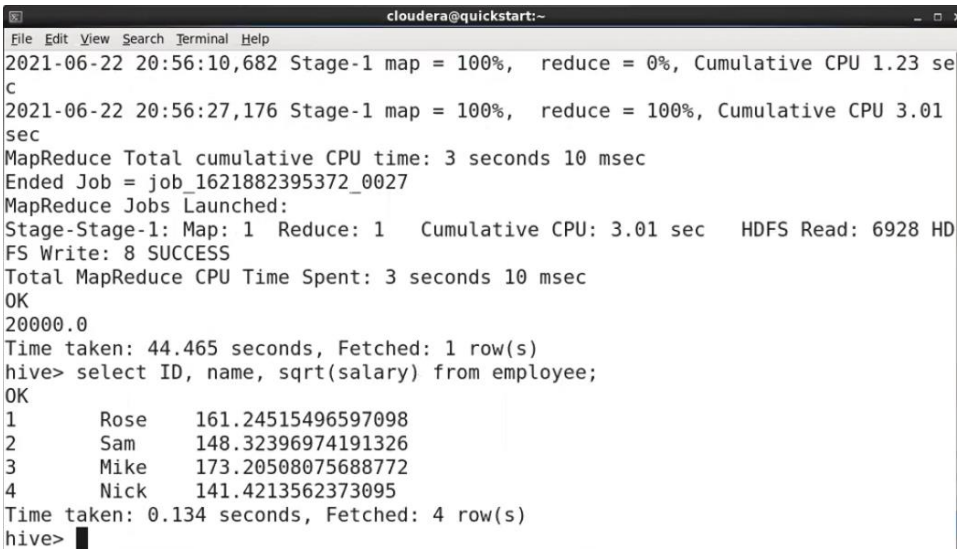




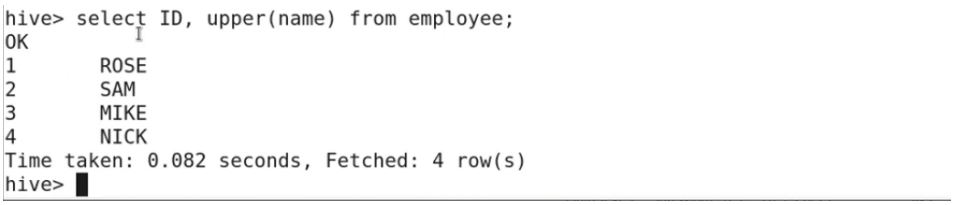
Here we can see that 20000 is minimum salary.

44. Now finding the **square root** of salary using below command,

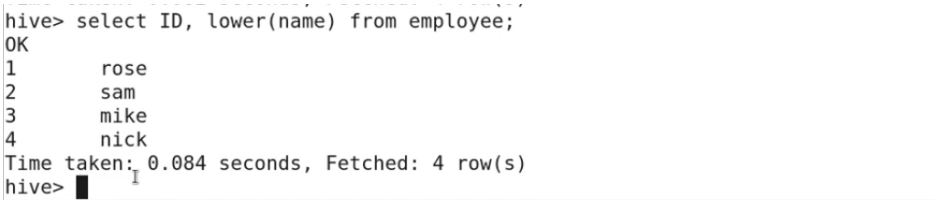
Select ID, name, sqrt(salary) from employee;



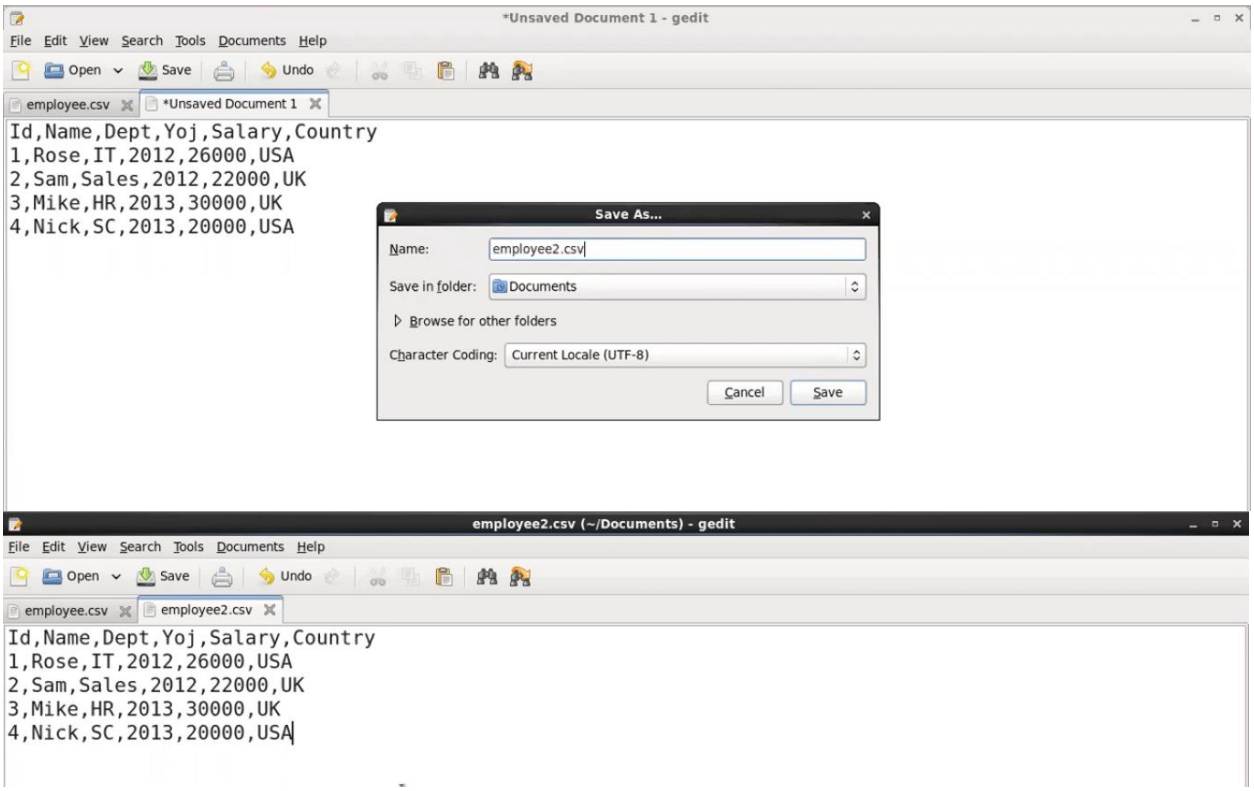
45. Now we are showing the name column in upper case using below command,
select ID, upper(name) from employee;



46. Now we are showing the name column in lower case using below command,
select ID, lower(name) from employee;

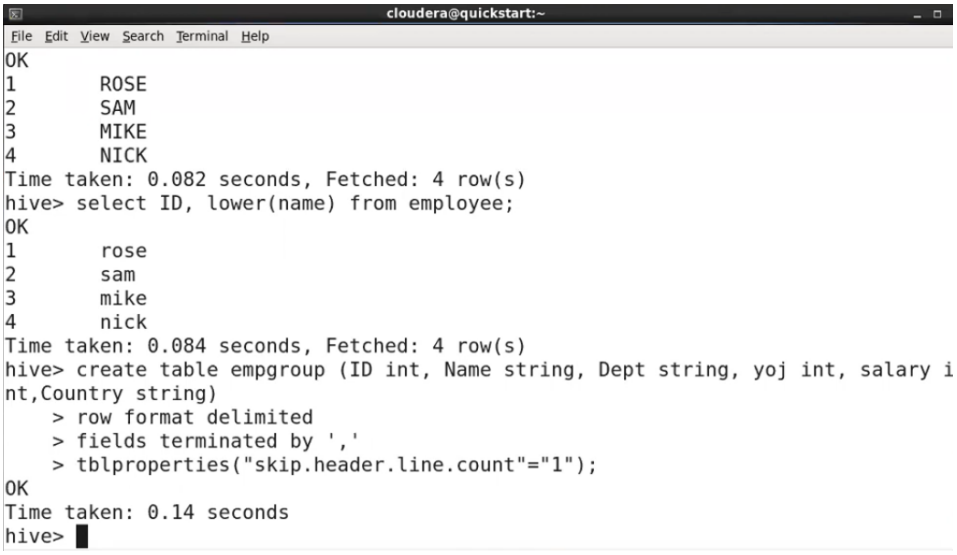


47. Creating another employee2.csv file with the same data as employee.csv but adding one more column as Country to it.



48. Creating a new table as **empgroup.**
create table empgroup (ID int, Name string, Dept string, yoj int, salary float, Country string)

- **row format delimited**
- **fields terminated by ‘,’;**
- **tblproperties(“skip.header.line.count” =”1”);**



49. Loading the data into **empgroup table** from **employee2.csv file** which we have created and it is present in /home/cloudera/Documents directory.

load data local inpath '/home/cloudera/Documents/employee2.csv' into table empgroup

Displaying the table using below command,
select * from empgroup;

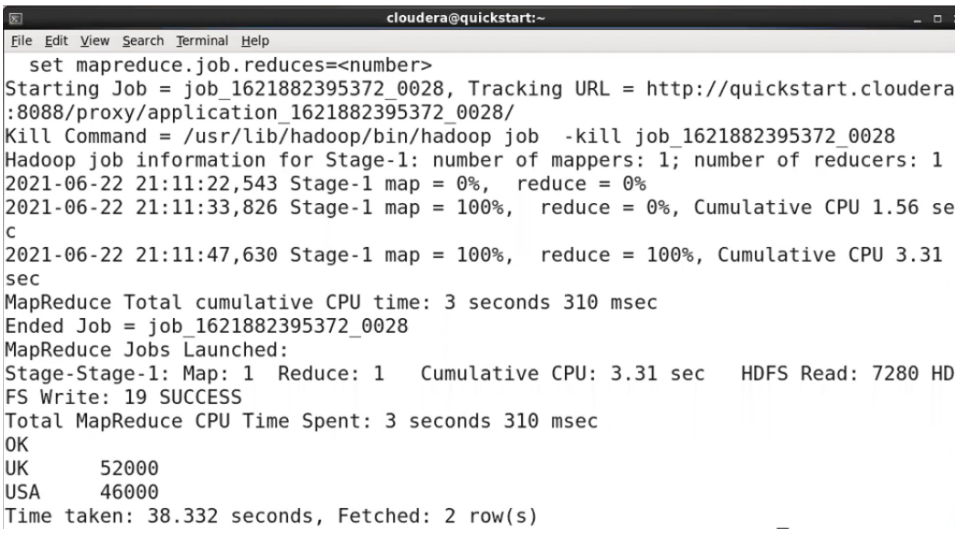
```
hive> load data local inpath '/home/cloudera/Documents/employee2.csv' into table
 empgroup;
Loading data to table hiveql.empgroup
Table hiveql.empgroup stats: [numFiles=1, totalSize=132]
OK
Time taken: 0.411 seconds
hive> select * from empgroup;
OK
1       Rose    IT      2012    26000   USA
2       Sam     Sales   2012    22000   UK
3       Mike    HR      2013    30000   UK
4       Nick    SC      2013    20000   USA
Time taken: 0.068 seconds, Fetched: 4 row(s)
hive>
```

50. Groupby clause

Now we display the total sum of salary of employees country wise using below command,

select country, sum(salary) from empgroup group by country;

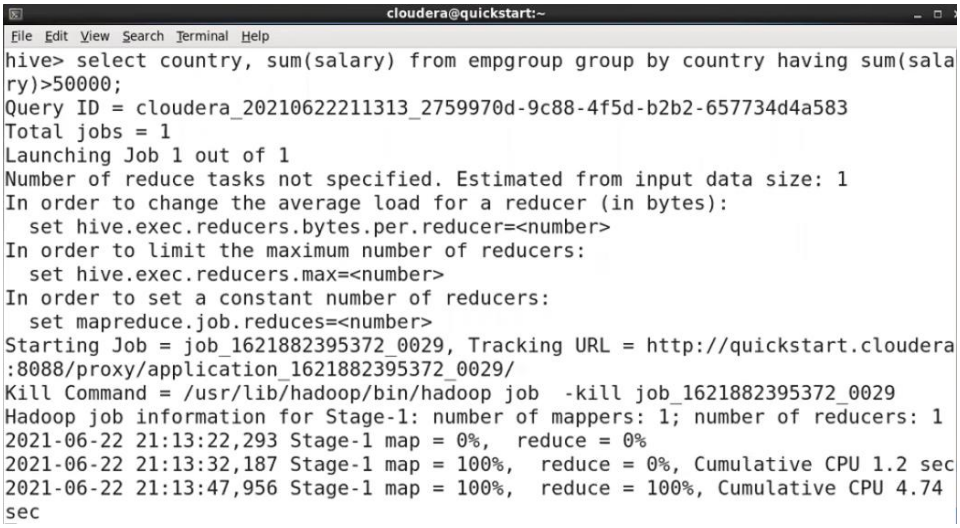
```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
2       Sam     Sales   2012    22000   UK
3       Mike    HR      2013    30000   UK
4       Nick    SC      2013    20000   USA
Time taken: 0.068 seconds, Fetched: 4 row(s)
hive> select country, sum(salary) from empgroup group by country;
Query ID = cloudera_20210622211111_c2ace1de-8169-4944-b942-3fa1dd9ec3e9
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>
Starting Job = job_1621882395372_0028, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0028/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0028
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-06-22 21:11:22,543 Stage-1 map = 0%,  reduce = 0%
```

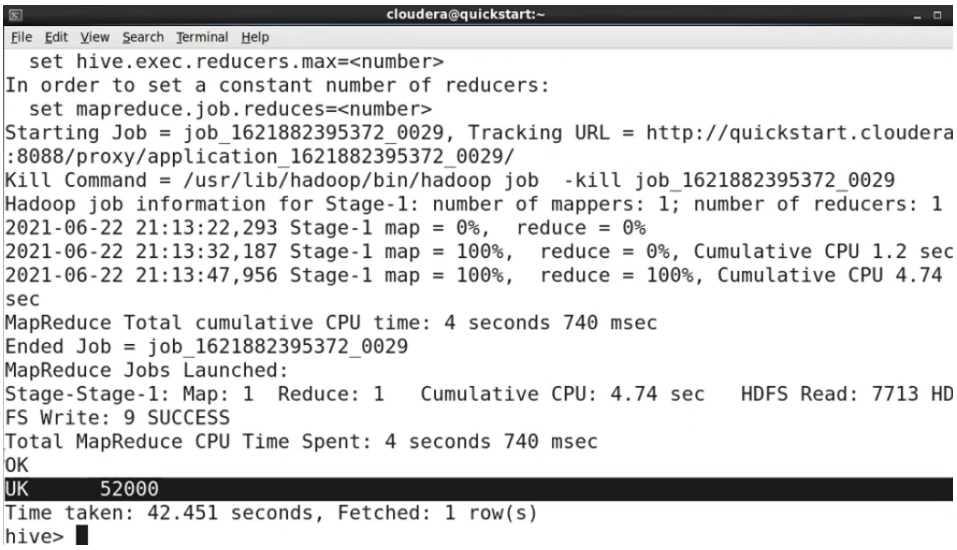


51. Groupby clause along with the having clause

Taking the total sum of salary countrywise using groupby clause and from that selecting or displaying those country whose total sum of salary>50000 using having clause.

select country, sum(salary) from empgroup group by country having sum(salary)>50000;

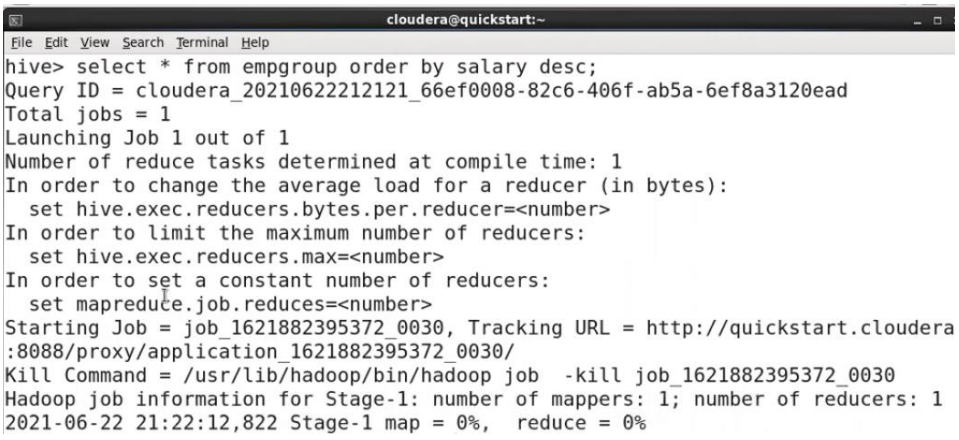


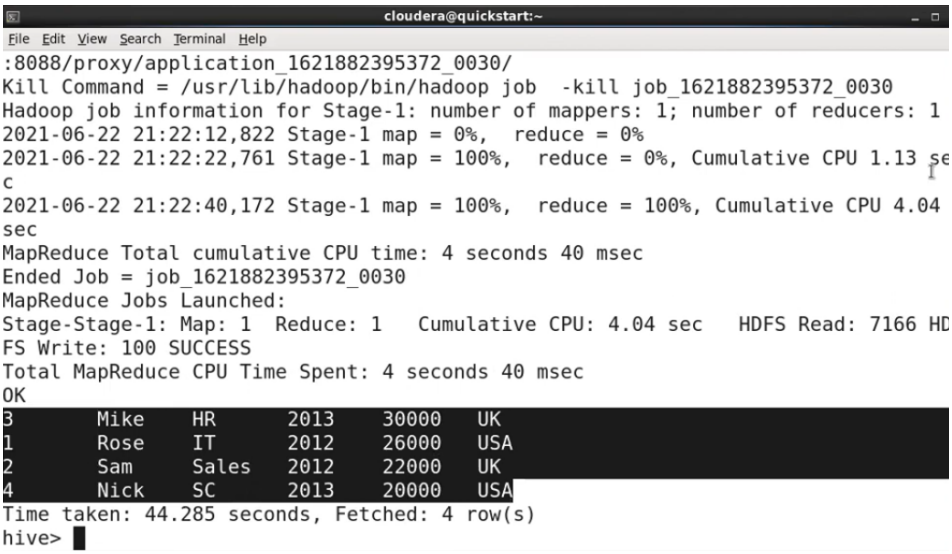


Sorting : Order by

52. Now we are displaying the table by sorting the salary in descending order.

Select * from empgroup order by salary desc;

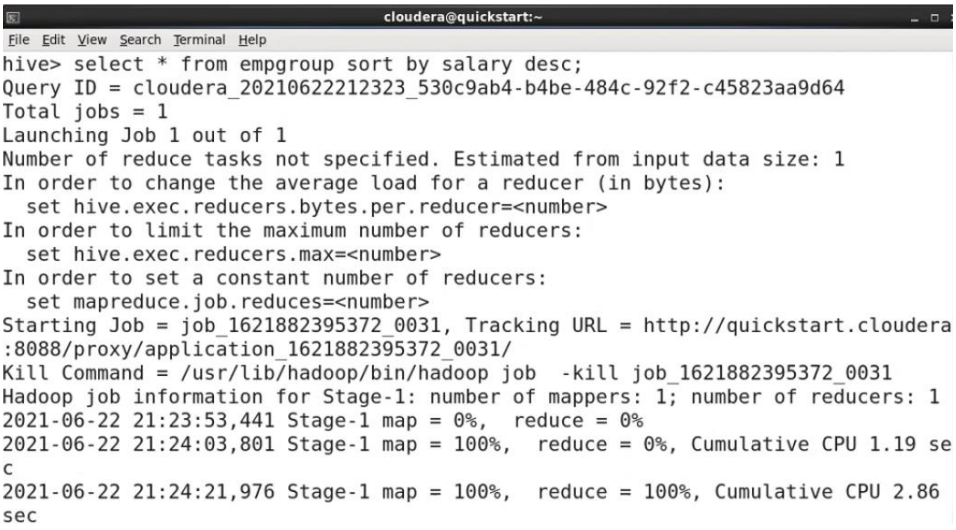


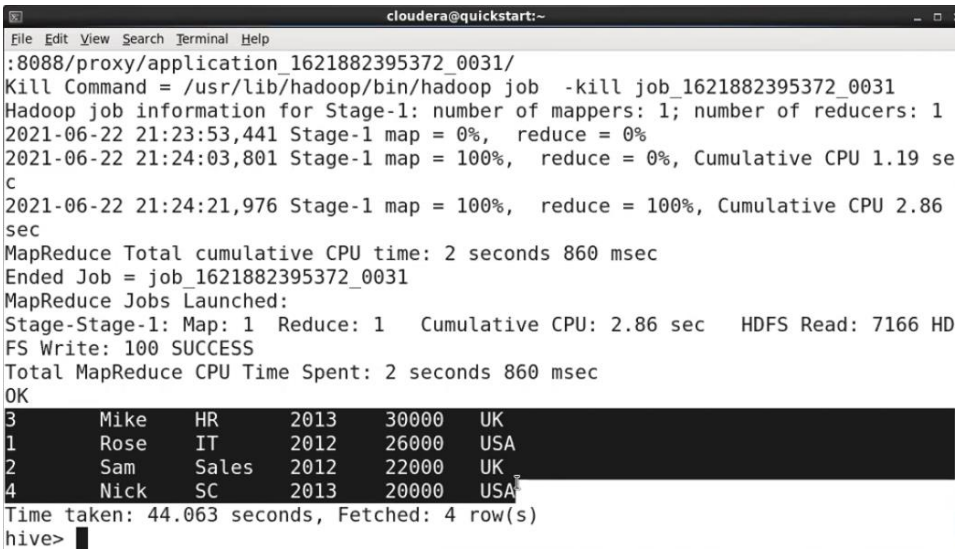


We can see that number of reducers: 1 for the order by.

53. Instead of order by if we have sort by,

Select * from empgroup sort by salary desc;





Now we can see the similar result as we got from order by and sort by so what is the difference between the two is that it depends on number of reducers in order by we got number of reducers is 1 and by using sort by here is also we got number of reducers is 1 so the difference between the two is that Order by will guarantee the total order in the output whereas sort by will only guarantee the ordering of the rows within the reducer. Order by gives us completely sorted result whereas sort by give us partially sorted result.