

What is HIVE:

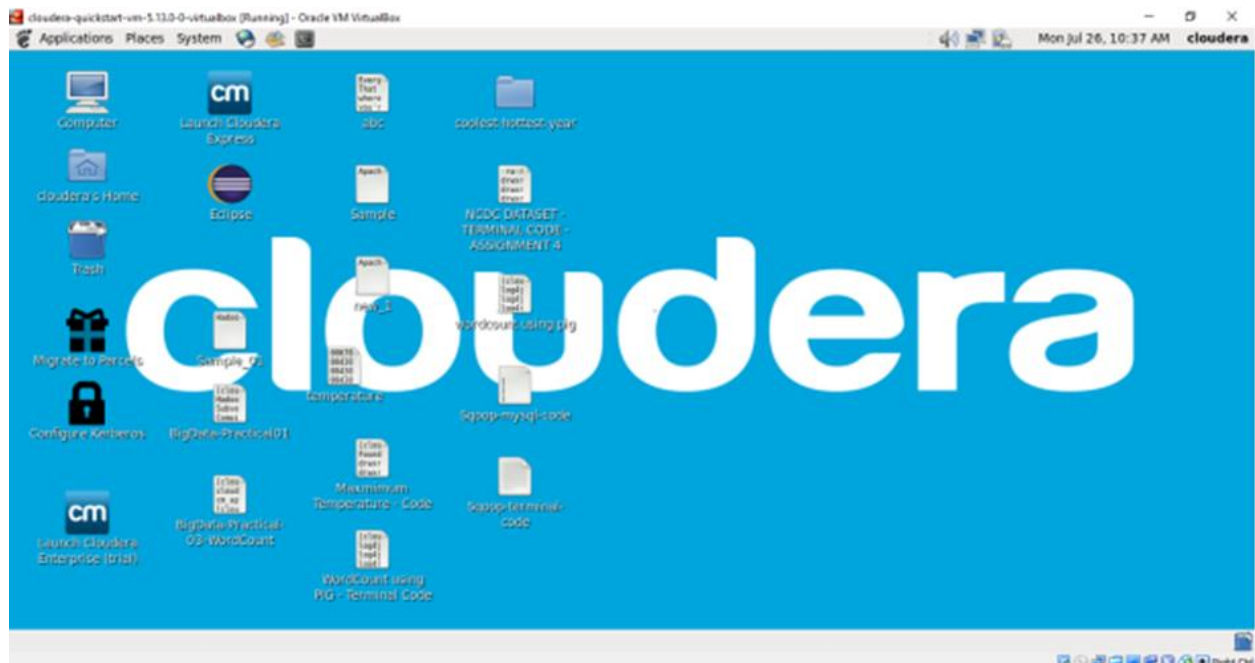
Hive is a data warehouse system which is used to analyse structured data. It is built on the top of Hadoop. It was developed by Facebook.

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).

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.

```
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
Time taken: 0.627 seconds, Fetched: 1 row(s)
```

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

Show databases;

```
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
Time taken: 0.627 seconds, Fetched: 1 row(s)
```

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: 2.833 seconds  
hive> use rjc;  
OK  
Time taken: 0.186 seconds
```

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: 0.022 seconds, Fetched: 2 row(s)
```

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

Use rjc;

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

show tables;

8. Creating a database “rjc”.

create database rjc;

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.

use rjc;

```
hive> create database rjc;  
OK  
Time taken: 2.833 seconds  
hive> use rjc;  
OK  
Time taken: 0.186 seconds
```

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: 0.46 seconds
```

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: 0.161 seconds, Fetched: 4 row(s)
```

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;

```
hive> describe formatted employee;
OK
# col_name          data_type          comment
id                  int
name                string
salary             float
age                 int

# Detailed Table Information
Database:           rjc
Owner:              cloudera
CreateTime:         Thu Mar 10 19:31:57 PST 2022
LastAccessTime:     UNKNOWN
Protect Mode:       None
Retention:          0
Location:           hdfs://quickstart.cloudera:8020/user/hive/warehouse/rjc.
db/employee
Table Type:         MANAGED_TABLE
Table Parameters:
    transient_lastDdlTime 1646969517

# 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      ,
    serialization.format
Time taken: 0.093 seconds, Fetched: 30 row(s)
```

By default hive creates Internal table or Managed Table.

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

create external table employee2 (ID int, name string, salary float, age int)

- row format delimited
- fields terminated by ','
- stored as textfile;

```
hive> create external table emloyee2 (ID int, name string, salary float, age int
)
> row format delimited
> fields terminated by ','
> stored as textfile;
OK
Time taken: 0.053 seconds
```

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: 0.092 seconds, Fetched: 4 row(s)
```

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

create external table employee3 (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: 0.084 seconds
hive> █
```

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: 0.086 seconds, Fetched: 4 row(s)
hive> █
```

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

show tables;


```
hive> show tables;  
OK  
employee2  
employee  
emptable  
Time taken: 0.011 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: 0.204 seconds
```

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  
employee2  
employee  
emptable  
Time taken: 0.011 seconds, Fetched: 3 row(s)  
hive>
```

22. First we will see the fields of emtable then we will add new column as **surname** in emtable using below command,

describe emtable;

Alter table emtable add columns (surname string);

describe emtable;

```
hive> Alter table emtable change name first_name string;
OK
Time taken: 0.177 seconds
hive> describe emtable;
OK
id                int
first_name        string
salary            float
age               int
surname            string
Time taken: 0.088 seconds, Fetched: 5 row(s)
hive> 
```

23. Now we will change field name of the emtable to first_name using alter command,

Alter table emtable change name first_name string;

describe emtable;

```
hive> Alter table emptable change name first_name string;
OK
Time taken: 0.177 seconds
hive> describe emptable;
OK
id                int
first_name        string
salary           float
age              int
surname           string
Time taken: 0.088 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

```
hive>
>
> create database rjcstudent;
OK
Time taken: 0.035 seconds
hive> show databases;
OK
default
rjc
rjcstudent
Time taken: 0.017 seconds, Fetched: 3 row(s)
```

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

gedit Student.csv

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

26. Creating a new database as rjcstudent.

create database rjcstudent;

show databases;

Using rjcstudent database.

use rjcstudent;

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: 0.152 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: 0.071 seconds, Fetched: 9 row(s)
hive>
```

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 student
> partition(Course = 'HADOOP')
> ;
Loading data to table rjcstudent.student partition (course=HADOOP)
Partition rjcstudent.student{course=HADOOP} stats: [numFiles=1, numRows=0, totalSize=99, rawDataSize=0]
OK
Time taken: 0.945 seconds
```

It is partitioning based on Hadoop.

select * from student;

```
hive> SELECT * FROM STUDENT;
OK
NULL    NAME    NULL    HADOOP
1       REHAN   NULL    HADOOP
2       RISHI   NULL    HADOOP
3       SHIVAM  NULL    HADOOP
4       ANAND   NULL    HADOOP
5       PRINCE  NULL    HADOOP
Time taken: 0.532 seconds, Fetched: 6 row(s)
hive>
```

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

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

```
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student
> partition(Course ='PYTHON')
> ;
Loading data to table rjcstudent.student partition (course=PYTHON)
Partition rjcstudent.student{course=PYTHON} stats: [numFiles=1, numRows=0, totalSize=99, rawDataSize=0]
OK
Time taken: 0.367 seconds
hive> SELECT * FROM STUDENT;
OK
NULL    NAME    NULL    HADOOP
1       REHAN   NULL    HADOOP
2       RISHI   NULL    HADOOP
3       SHIVAM  NULL    HADOOP
4       ANAND   NULL    HADOOP
5       PRINCE  NULL    HADOOP
NULL    NAME    NULL    PYTHON
1       REHAN   NULL    PYTHON
2       RISHI   NULL    PYTHON
3       SHIVAM  NULL    PYTHON
4       ANAND   NULL    PYTHON
5       PRINCE  NULL    PYTHON
Time taken: 0.087 seconds, Fetched: 12 row(s)
```

```

hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student
> partition(Course = 'ML')
> ;
Loading data to table rjcstudent.student partition (course=ML)
Partition rjcstudent.student{course=ML} stats: [numFiles=1, numRows=0, totalSize=99, rawDataSize=0]
OK
Time taken: 0.374 seconds
hive> SELECT * FROM STUDENT;
OK
NULL    NAME    NULL    HADOOP
1       REHAN   NULL    HADOOP
2       RISHI   NULL    HADOOP
3       SHIVAM  NULL    HADOOP
4       ANAND   NULL    HADOOP
5       PRINCE  NULL    HADOOP
NULL    NAME    NULL    ML
1       REHAN   NULL    ML
2       RISHI   NULL    ML
3       SHIVAM  NULL    ML
4       ANAND   NULL    ML
5       PRINCE  NULL    ML
NULL    NAME    NULL    PYTHON
1       REHAN   NULL    PYTHON
2       RISHI   NULL    PYTHON
3       SHIVAM  NULL    PYTHON
4       ANAND   NULL    PYTHON
5       PRINCE  NULL    PYTHON
Time taken: 0.078 seconds, Fetched: 18 row(s)
hive>

```

Drop: to drop the entire table we can use drop table command.

Syntax: DROP table tablename;

```

hive> DROP table student;
OK
Time taken: 0.774 seconds
hive> create table student (ID int, Name string, Age int)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count" = "1");

```

```

hive> create table student (ID int, Name string, Age int)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count" = '1');
OK
Time taken: 0.06 seconds
hive> SELECT * FROM STUDENT;
OK
Time taken: 0.061 seconds

```

```
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student
      > partition(Course="Hadoop")
      > ;
Loading data to table default.student partition (course=Hadoop)
Partition default.student{course=Hadoop} stats: [numFiles=1, numRows=0, totalSize=127, rawDataSize=0]
OK
Time taken: 1.007 seconds
hive> select * from student;
OK
NULL      Name      NULL      Hadoop
1         Akshata NULL      Hadoop
2         Sarita  NULL      Hadoop
3         Priti   NULL      Hadoop
4         Shivani NULL      Hadoop
5         Kajal   NULL      Hadoop
6         Ajay    NULL      Hadoop
Time t
```

```
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student
      > partition(Course="Java");;
Loading data to table default.student partition (course=Java)
Partition default.student{course=Java} stats: [numFiles=1, numRows=0, totalSize=127, rawDataSize=0]
OK
Time taken: 0.42 seconds
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student partition(Course="Python");
Loading data to table default.student partition (course=Python)
Partition default.student{course=Python} stats: [numFiles=1, numRows=0, totalSize=127, rawDataSize=0]
OK
Time taken: 0.558 seconds
```

```
hive> drop table student;
OK
Time t;
OK
Time taken: 0.053 seconds
```



```
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: 0.096 seconds
```

```
hive> load data local inpath '/home/cloudera/Documents/Student.csv' into table student;
Loading data to table default.student
Table default.student stats: [numFiles=1, totalSize=127]
OK
Time taken: 0.209 seconds
hive> select * from student;
OK
1      Akshata Hadoop  21
2      Sarita  Java   22
3      Priti   Java   23
4      Shivani Python 25
5      Kajal  Hadoop  21
6      Ajay   Python  23
Time taken: 0.076 seconds, Fetched: 6 row(s)
```

```
hive> create database hiveql;
OK
Time taken: 0.154 seconds
```

Creating a new table as employee.

create table employee(ID int, Name string, Dept string, yoj int, salary float,
Country string)

- row format delimited
- fields terminated by ',';
- tblproperties(("skip.header.line.count"="1"));

```

hive> 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");
OK
Time taken: 0.299 seconds
hive> describe employee;
OK
id                int
name              string
department        string
yoy              int
salary           float
Time taken: 0.182 seconds, Fetched: 5 row(s)

```

Loading the data into employee 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 employee;

```

hive> select * from employee where salary >=25000;
OK
1      Akshata IT      2018      50000.0
3      Priti  HR      2012      34000.0
4      Shivani SC     2015      45000.0
Time taken: 0.246 seconds, Fetched: 3 row(s)

```

```

hive> select * from employee where salary <25000;
OK
2      Sarita Sales   2022      23000.0
Time taken: 0.109 seconds, Fetched: 1 row(s)

```

```

hive> select ID ,name,salary+5000 from employee;
OK
1      Akshata 55000.0
2      Sarita  28000.0
3      Priti   39000.0
4      Shivani 50000.0
NULL   NULL    NULL
Time taken: 0.07 seconds, Fetched: 5 row(s)

```

```
hive> select max(Salary)from employee;
Query ID = cloudera_20220314210707_1f8c8a5d-d609-44de-aa5e-7437d774bbb3
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_1644894610889_0012, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1644894610889_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1644894610889_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-14 21:08:09,884 Stage-1 map = 0%, reduce = 0%
2022-03-14 21:08:18,811 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.86 se
c
2022-03-14 21:08:29,591 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.96
sec
MapReduce Total cumulative CPU time: 1 seconds 960 msec
Ended Job = job_1644894610889_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 1.96 sec HDFS Read: 6880 HD
FS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 960 msec
OK
50000.0
Time taken: 34.969 seconds, Fetched: 1 row(s)
```

```

hive> select min(Salary)from employee;
Query ID = cloudera_20220314210808_a7f2dda9-084c-4d61-8f70-f52d55661d33
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_1644894610889_0013, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1644894610889_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1644894610889_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-14 21:09:01,568 Stage-1 map = 0%, reduce = 0%
2022-03-14 21:09:09,159 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.77 se
c
2022-03-14 21:09:17,760 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.84
sec
MapReduce Total cumulative CPU time: 1 seconds 840 msec
Ended Job = job_1644894610889_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 1.84 sec HDFS Read: 6903 HD
FS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 840 msec
OK
23000.0
Time taken: 27.823 seconds, Fetched: 1 row(s)

```

```

hive> select ID,Name,sqrt(Salary)from employee;
OK
1      Akshata 223.60679774997897
2      Sarita  151.65750888103102
3      Priti   184.39088914585776
4      Shivani 212.13203435596427
NULL   NULL    NULL
Time taken: 0.082 seconds, Fetched: 5 row(s)

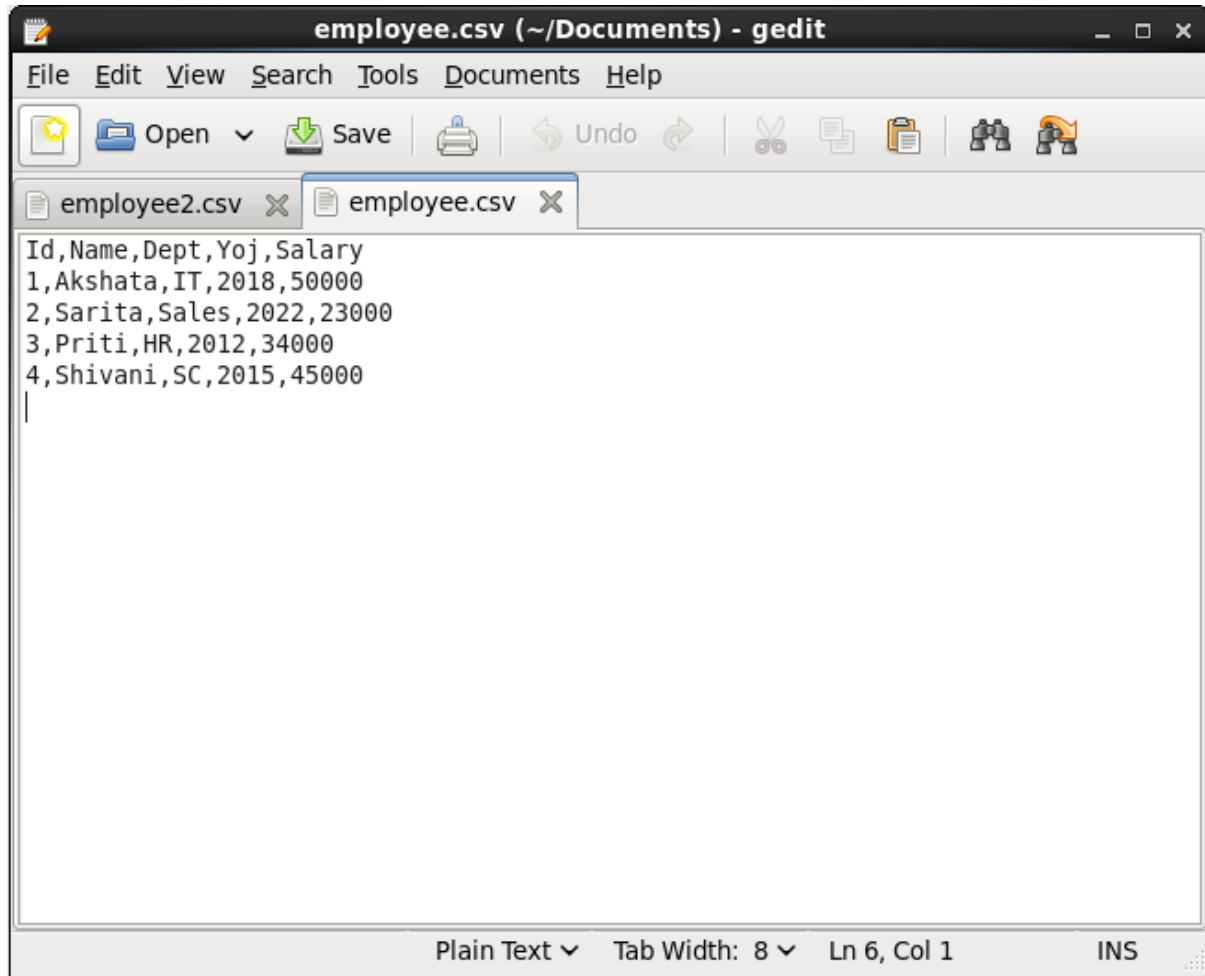
```

```

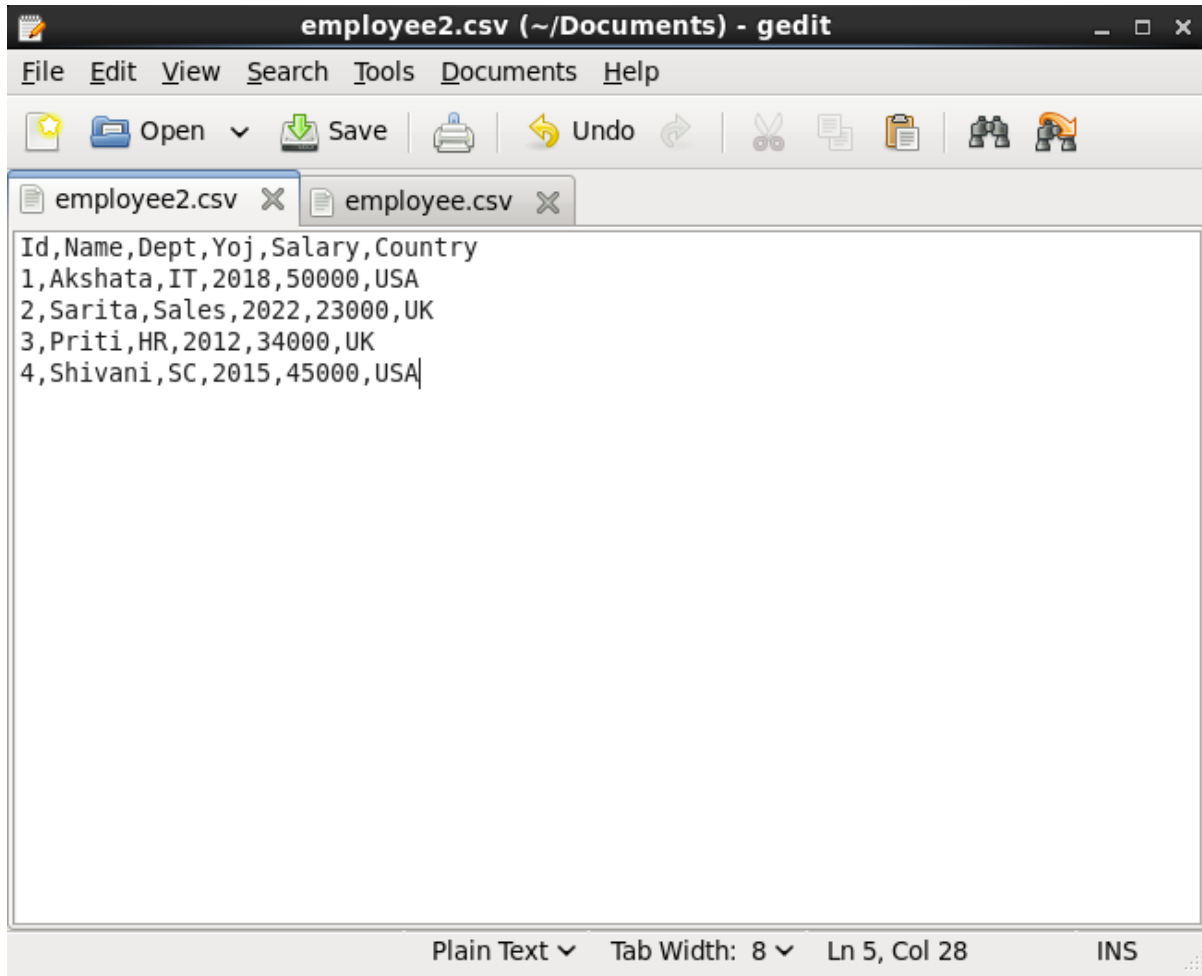
hive> select ID,upper(Name)from employee;
OK
1      AKSHATA
2      SARITA
3      PRITI
4      SHIVANI
NULL   NULL
Time taken: 0.072 seconds, Fetched: 5 row(s)

```

Name: Atharva Suroshe
Roll No: 40



```
Id,Name,Dept,Yoj,Salary
1,Akshata,IT,2018,50000
2,Sarita,Sales,2022,23000
3,Priti,HR,2012,34000
4,Shivani,SC,2015,45000
```



Now creating employee.csv file.

```
gedit employee.csv
```

Creating new database for performing querying operations.

Create database hiveql;

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 empgrp(ID int,Name string,Department string,YOJ int,Salary float,Country string)
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ','
> tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.071 seconds
```

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/employee2.csv' into table empgrp;
Loading data to table default.empgrp
Table default.empgrp stats: [numFiles=1, totalSize=142]
OK
Time taken: 0.372 seconds
hive> select * from empgrp;
OK
1      Akshata IT      2018      50000.0 USA
2      Sarita Sales    2022      23000.0 UK
3      Priti HR        2012      34000.0 UK
4      Shivani SC      2015      45000.0 USA
Time taken: 0.044 seconds, Fetched: 4 row(s)
```

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;

```

hive> select Country,sum(Salary) from empgrp group by Country;
Query ID = cloudera_20220314211717_43a54ee2-d4d8-4df0-9fea-1c1a8e97bb
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_1644894610889_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1644894610889_0014/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1644894610889_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-14 21:17:56,866 Stage-1 map = 0%, reduce = 0%
2022-03-14 21:18:04,436 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.81 sec
2022-03-14 21:18:12,942 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.88 sec
MapReduce Total cumulative CPU time: 1 seconds 880 msec
Ended Job = job_1644894610889_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 1.88 sec HDFS Read: 7321 HD
FS Write: 23 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 880 msec
OK
UK      57000.0
USA     95000.0
Time taken: 28.81 seconds, Fetched: 2 row(s)

```

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;


```

hive> select * from empgrp order by Salary desc;
Query ID = cloudera_20220314211919_ff2ca655-14b8-4c7f-9a6f-0cda9ca4fa59
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_1644894610889_0016, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1644894610889_0016/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1644894610889_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-14 21:19:51,589 Stage-1 map = 0%, reduce = 0%
2022-03-14 21:20:00,259 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.74 se
c
2022-03-14 21:20:08,901 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.83
sec
MapReduce Total cumulative CPU time: 1 seconds 830 msec
Ended Job = job_1644894610889_0016
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 1.83 sec HDFS Read: 7186 HD
FS Write: 118 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 830 msec
OK
1      Akshata IT      2018      50000.0 USA
4      Shivani SC      2015      45000.0 USA
3      Priti HR       2012      34000.0 UK
2      Sarita Sales    2022      23000.0 UK
Time taken: 28.749 seconds, Fetched: 4 row(s)

```

Instead of order by if we have sort by,

Select * from employee 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.

Name: Atharva Suroshe
Roll No: 40

Name: Atharva Suroshe
Roll No: 40