Using HIVE Tool

Hive

- Hive is a data warehousing tool that is built on top of Hadoop.
- It provides a SQL-like interface to query data stored in Hadoop Distributed File System (HDFS) or other compatible data stores.
- Hive allows users to write queries in a familiar language called HiveQL (similar to SQL), which is then translated into MapReduce jobs that run on the Hadoop cluster.
- Hive is designed for handling large datasets and is optimized for batch processing, which makes it a great choice for running analytical queries on big data.
- It also supports custom user-defined functions (UDFs) that can be used to extend its capabilities.
- Hive is part of the Hadoop ecosystem and works in conjunction with other Hadoop components like HDFS, MapReduce, and YARN.
- It can be used for a variety of tasks such as data analysis, data mining, and business intelligence reporting.

Cloudera

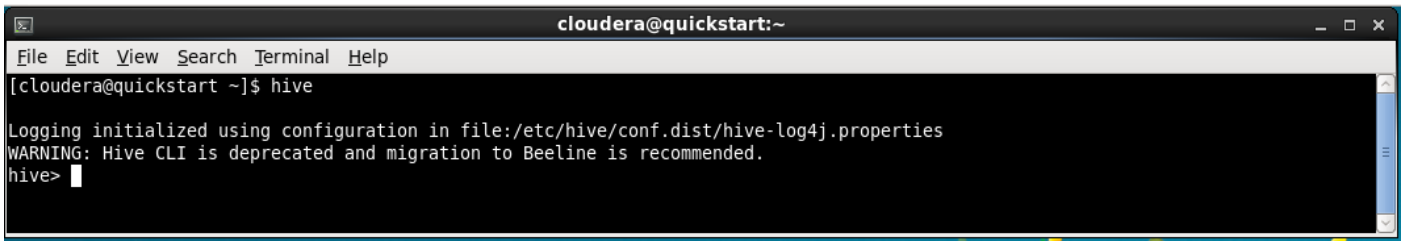
- Cloudera is a company that provides a comprehensive data management and analytics platform built on top of Hadoop.
- The Cloudera platform includes various tools and services that make it easier to store, process, analyse, and manage large volumes of structured and unstructured data.

To Begin:

Start Oracle VirtualBox and boot up the Cloudera VM

Once Cloudera is properly booted, open a new Terminal window to begin the Hive Practical

Then access the Hive CLI using the 'hive' command

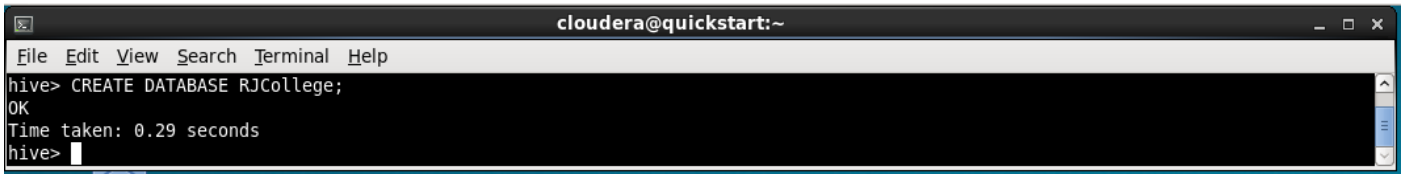


Example 1:

1. Create a data warehouse database named 'RJCollege' using Hive.

To create a database in Hive we use the command

- CREATE DATABASE RJCollege;

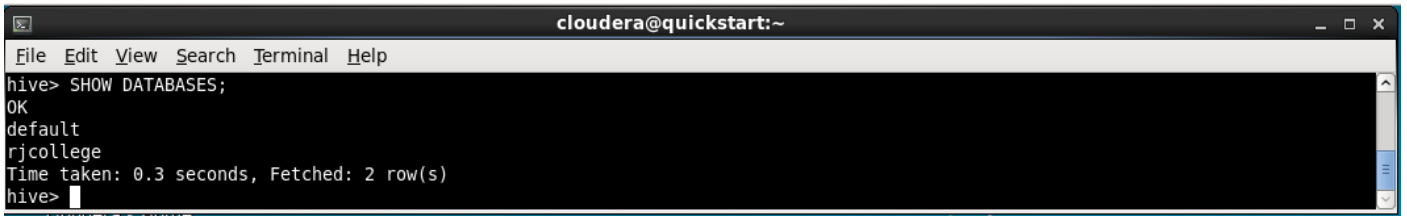


2. Check the creation of a data warehouse database.

To check if the database was created we use the command

- SHOW DATABASES;

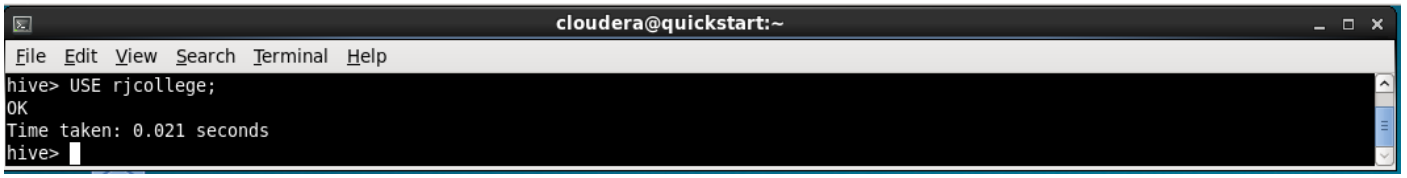
This will return a list of databases which should contain our newly created database RJCollege



3. Create a table named 'student' in the RJCollege warehouse.

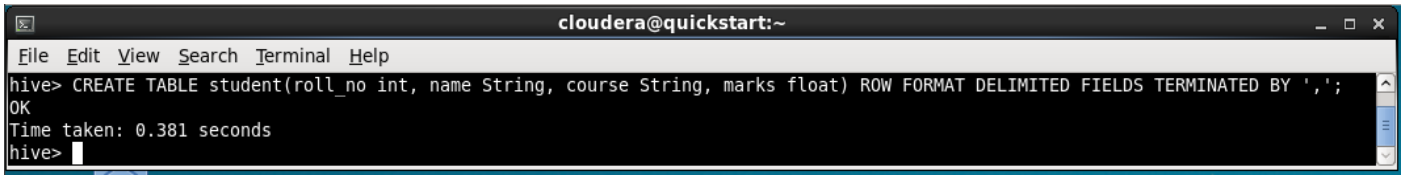
To create a table inside our database we first need to use or activate our database using command

- USE rjcollege;



Then to create a table named student we use command

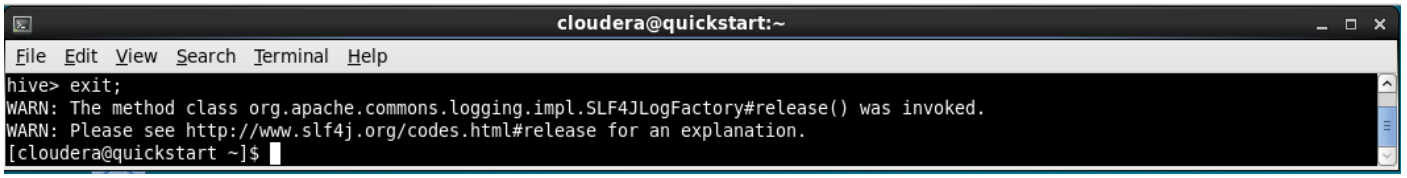
- CREATE TABLE student(roll_no int, name String, course String, marks float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’;



4. Create the data file named 'studData.txt' with students data and enter any 5 students data and copy it to HDFS

To do this task we first need to come out of Hive CLI

- exit;



To create a text file we can use the gedit command

- gedit studData.txt

Then we can enter the following data then save and close the gedit window

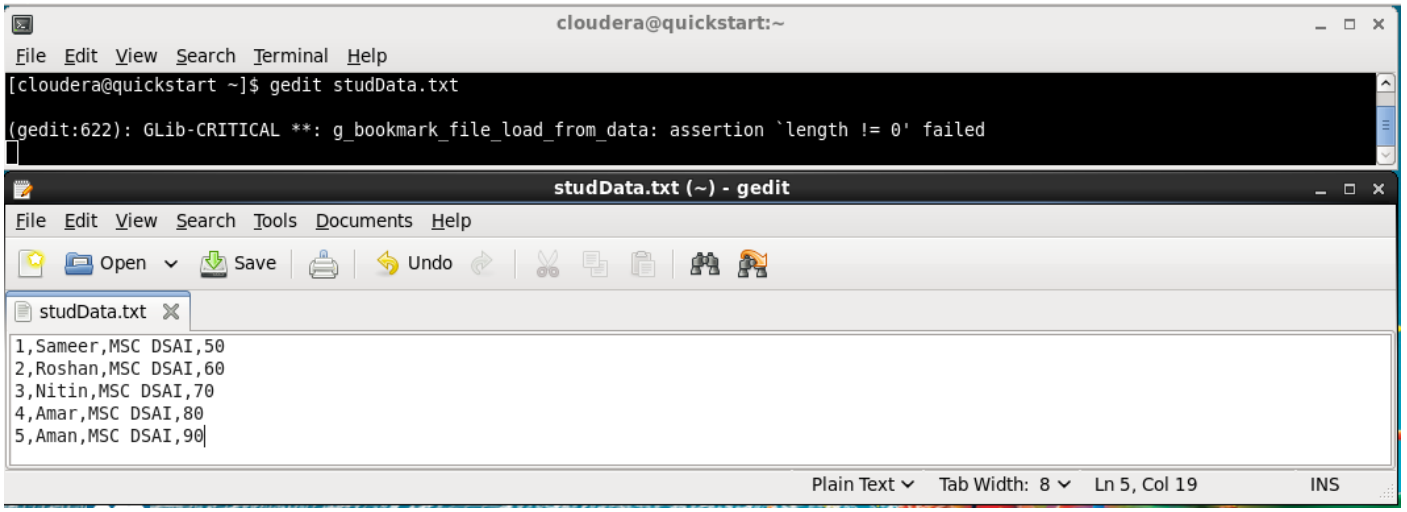
1,Sameer,MSC DSAI,50

2,Roshan, MSC DSAI, 60

3,Nitin,MSC DSAI,70

4,Amar,MSC DSAI,80

5,Aman,MSC DSAI,90



To copy it to HDFS we first need to create a directory in hdfs then move our file to that location

- hdfs dfs -mkdir /hadoop
- hdfs dfs -mkdir /hadoop/data
- hdfs dfs -put studData.txt /hadoop/data

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hdfs dfs -mkdir /hadoop
[cloudera@quickstart ~]$ hdfs dfs -mkdir /hadoop/data
[cloudera@quickstart ~]$ hdfs dfs -put studData.txt /hadoop/data
[cloudera@quickstart ~]$
```

5. Display the schema of the student table

To get the table details, go back into Hive CLI and run command

- DESCRIBE FORMATTED rjcollege;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> USE rjcollege;
OK
Time taken: 0.283 seconds
hive> DESCRIBE FORMATTED student;
OK
# col_name              data_type               comment

roll_no                 int
name                    string
course                  string
marks                   float
```

6. Display the list of all tables or confirm the creation of a student table

To get a list of tables, run command

- SHOW TABLES;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> SHOW TABLES;
OK
student
Time taken: 0.041 seconds, Fetched: 1 row(s)
hive>
```

7. Load data of studData.txt into the Hive table

For loading the data from a text file from hdfs into hive table we use

- LOAD DATA INPATH '/hadoop/data/studData.txt' INTO TABLE student;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> LOAD DATA INPATH '/hadoop/data/studData.txt' INTO TABLE student;
Loading data to table rjcollege.student
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/rjcollege.db/student/studData.txt': User does no
t belong to hive
Table rjcollege.student stats: [numFiles=1, totalSize=100]
OK
Time taken: 0.536 seconds
hive>
```

8. Display all students information/results

To display the data of student table we use

- SELECT * FROM student;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT * FROM student;
OK
1       Sameer  MSC DSAI        50.0
2       Roshan  MSC DSAI        60.0
3       Nitin   MSC DSAI        70.0
4       Amar    MSC DSAI        80.0
5       Aman    MSC DSAI        90.0
Time taken: 0.333 seconds, Fetched: 5 row(s)
hive>
```

Example 2

1. Create the csv file to store the data of the LIC Insurance policy customers and enter some data

Dsd

- gedit custDetails.csv

  3272981,suman,04/29/2001,suman@gmail.com,3079875121,vikhroli,M

  3272982,foram,04/30/2001,foram@gmail.com,3079876062,dombivali,F

  3272983,savri,05/01/2001,savri@gmail.com,3079877003,mulund,F

  3272984,siddhesh,05/02/2001,siddhesh@gmail.com,3079877944,ghatkopar,M

  3272985,jayesh,05/03/2001,jayesh@gmail.com,3079878885,vikhroli,M

- gedit policySaleDetails.csv

  18641,3272981,83475,04/29/2021,yearly,2584846

  18642,3272983,83476,04/30/2021,quarterly,54545445

  18643,3272984,83477,05/01/2021,monthly,774474

  18644,3272985,83473,05/02/2021,weekly,7452558

18645,3272982,83474,05/03/2021,yearly,7474785

- gedit policyDetails.csv

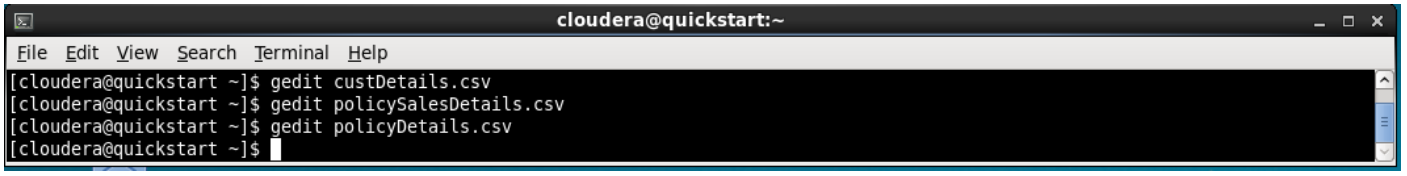
83473,trade,Health,34,20,30%

83474,union,Motor,18,5,40%

83475,backwardhome,home,35,50,40%

83476,Lpg,fire,16,10,60%

83477,smallvillage,travel,5,1,40%

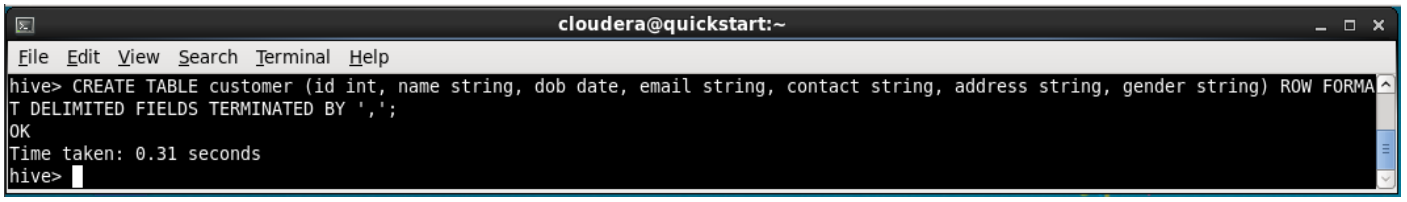


2. Create a data warehouse database named 'LICDW' using Hive

- CREATE DATABASE licdw;

3. Create an Internal/managed table for CustDetails using Hive

- USE licdw;
- CREATE TABLE customer (id int, name string, dob date, email string, contact string, address string, gender string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';



4. Load the data of custDetailsData.csv into the CustDetails table

- LOAD DATA LOCAL INPATH '/home/cloudera/custDetails.csv' INTO TABLE customer;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> LOAD DATA LOCAL INPATH '/home/cloudera/custDetails.csv' INTO TABLE customer;
Loading data to table licdw.customer
Table licdw.customer stats: [numFiles=1, totalSize=323]
OK
Time taken: 0.958 seconds
hive>
```

5. Display all records of custDetails table

- SELECT * FROM customer;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT * FROM customer;
OK
3272981 suman   NULL    suman@gmail.com 3079875121      vikhroli        M
3272982 foram   NULL    foram@gmail.com 3079876062      dombivali       F
3272983 savri   NULL    savri@gmail.com 3079877003      mulund  F
3272984 siddhesh        NULL    siddhesh@gmail.com      3079877944      ghatkopar       M
3272985 jayesh  NULL    jayesh@gmail.com        3079878885      vikhroli        M
Time taken: 0.605 seconds, Fetched: 5 row(s)
hive>
```

6. Create a policySaleDetails table as an external table using Hive

- CREATE EXTERNAL TABLE policy_detail(id int, name string, type string, age_criteria int, tenure int, maturity string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/home/cloudera/policy_detail_data';

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE EXTERNAL TABLE policy_detail(id int, name string, type string, age_criteria int, tenure int, maturity string) ROW FOR
MAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/home/cloudera/policy_detail_data'
    > ;
OK
Time taken: 0.125 seconds
hive>
```

7. Load the data of PolicySaleDetailsData.csv file to PolicySaleDetails table

- LOAD DATA LOCAL INPATH '/home/cloudera/policySalesDetails.csv' INTO TABLE policy_detail;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> LOAD DATA LOCAL INPATH '/home/cloudera/policySalesDetails.csv' INTO TABLE policy_detail;
Loading data to table licdw.policy_detail
Table licdw.policy_detail stats: [numFiles=1, totalSize=235]
OK
Time taken: 0.313 seconds
hive>
```

8. Display the schema and data details of the PolicySaleDetails table

- SELECT * FROM policy_detail;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> SELECT * FROM policy_detail;
OK
18641   3272981 83475   NULL    NULL    2584846
18642   3272983 83476   NULL    NULL    54545445
18643   3272984 83477   NULL    NULL    774474
18644   3272985 83473   NULL    NULL    7452558
18645   3272982 83474   NULL    NULL    7474785
NULL    NULL    NULL    NULL    NULL    NULL
Time taken: 0.082 seconds, Fetched: 6 row(s)
hive>
```

9. Skip header line of dataset file while loading data in Hive table

- ALTER TABLE policy_detail SET TBLPROPERTIES("skip.header.line.count"="1");

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> ALTER TABLE policy_detail SET TBLPROPERTIES("skip.header.line.count"="1");
OK
Time taken: 0.155 seconds
hive>
```

10. Insert some records in both internal and external Hive tables using the Insert command

- INSERT INTO customer VALUES(3272986, 'sameer', '12/09/1998', 'sam@gmail.com', 12456853233, 'Dombivli', 'M');

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> INSERT INTO customer VALUES(3272986, 'sameer', '12/09/1998', 'sam@gmail.com', 12456853233, 'Dombivli', 'M');
Query ID = cloudera_20230403095353_b2e1f70a-5baa-4aca-aa31-533094bab149
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1680529255881_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1680529255881_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1680529255881_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
```

- INSERT INTO policy_detail VALUES(83478, 'smallvillage', 'travel', 5, 1, '40%');

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> INSERT INTO policy_detail VALUES(83478, 'smallvillage', 'travel', 5, 1, '40%');
Query ID = cloudera_20230403095656_c84a2418-5da3-4f43-9eec-305f11d744c5
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1680529255881_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1680529255881_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1680529255881_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
```

11. Create the PolicyDetails temporary table

- CREATE TEMPORARY TABLE policy_detail_temp(id int, name string, type string, age_criteria int, tenure int, maturity string

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE TEMPORARY TABLE policy_detail_temp(id int, name string, type string, age_criteria int, tenure int, maturity string);
OK
Time taken: 0.037 seconds
hive>
```

12. Insert 2 records in the PolicyDetails table using insert command

- INSERT INTO policy_detail_temp VALUES(32768, 'trade', 'health', 24, 30, '40%');
- INSERT INTO policy_detail_temp VALUES(32769, 'union', 'motor', 18, 5, '30%');

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> INSERT INTO policy_detail_temp VALUES(32768, 'trade', 'health', 24, 30, '40%');
Query ID = cloudera_20230403105858_82e75e82-6974-44b4-a04a-d9456a4f7f28
Total jobs = 3
Launching Job 1 out of 3
```

13. Check the existing hive version, if it is 4.x then try transaction table creation

We can check hive version using command

- ! hive --version

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> ! hive --version;
Hive 1.1.0-cdh5.4.2
Subversion file:///data/jenkins/workspace/generic-package-rhel64-6-0/topdir/BUILD/hive-1.1.0-cdh5.4.2 -r Unknown
Compiled by jenkins on Tue May 19 17:00:20 PDT 2015
From source with checksum a59d7ebf7b9b9e41446f10bcf5274d93
hive>
```

14. Create PolicyDetailsDup table using PolicyDetails table, using CTAS statement

- CREATE TABLE policy_details_dup AS SELECT * FROM policy_detail;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE TABLE policy_details_dup AS SELECT * FROM policy_detail;
Query ID = cloudera_20230403110909_e548d3b7-f655-4acc-a47f-b48c763f6259
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1680529255881_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1680529255881_0005/
```

15. Execute describe and select statements for PolicyDetailsDup table

- DESCRIBE policy_details_dup;
- SELECT * FROM policy_details_dup;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> DESCRIBE policy_details_dup;
OK
id                      int
name                    string
type                    string
age_criteria            int
tenure                  int
maturity                string
Time taken: 0.085 seconds, Fetched: 6 row(s)
hive> SELECT * FROM policy_details_dup;
OK
18642   3272983 83476   NULL    NULL    54545445
18643   3272984 83477   NULL    NULL    774474
18644   3272985 83473   NULL    NULL    7452558
18645   3272982 83474   NULL    NULL    7474785
NULL    NULL    NULL    NULL    NULL    NULL
Time taken: 0.067 seconds, Fetched: 5 row(s)
hive>
```

16. Create a PolicyDetailsLike table using the existing PolicyDetails table

- CREATE TABLE policy_details_like LIKE policy_detail;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> CREATE TABLE policy_details_like LIKE policy_detail;
OK
Time taken: 0.102 seconds
hive>
```

17. Execute describe and select statements for PolicyDetailsLike table

- DESCRIBE policy_details_like;
- SELECT * FROM policy_details_like;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> DESCRIBE policy_details_like;
OK
id                      int
name                    string
type                    string
age_criteria            int
tenure                  int
maturity                string
Time taken: 0.148 seconds, Fetched: 6 row(s)
hive> SELECT * FROM policy_details_like;
OK
Time taken: 0.06 seconds
hive>
```

18. Display the list of customers and their mail ids from the custDetails table

- SELECT name, email FROM customer;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT name, email from customer;
OK
sameer  sam@gmail.com
suman   suman@gmail.com
foram   foram@gmail.com
savri   savri@gmail.com
siddhesh        siddhesh@gmail.com
jayesh  jayesh@gmail.com
Time taken: 0.073 seconds, Fetched: 6 row(s)
hive>
```

19. Get the count of the total number of customers

- SELECT COUNT(*) FROM customer;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.88 sec   HDFS Read: 7467 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 880 msec
OK
6
Time taken: 34.162 seconds, Fetched: 1 row(s)
hive>
```

20. Display the premium paid details of customer having id 3272982

- SELECT * FROM policy_sales WHERE cust_id = 3272982;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT * FROM policy_sales WHERE cust_id = 3272982;
OK
18645   3272982 83474   NULL    yearly  7474785
Time taken: 0.203 seconds, Fetched: 1 row(s)
hive>
```

21. Display the policy with maximum benefit

- SELECT * FROM policy_detail ORDER BY maturity DESC LIMIT 1;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.76 sec   HDFS Read: 7383 HDFS Write: 25 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 760 msec
OK
83476   Lpg     fire    16      10      60%
Time taken: 33.833 seconds, Fetched: 1 row(s)
hive>
```

22. Display the details of policy having tuners in the range of 24 to 48 months

- SELECT * FROM policy_detail WHERE tenure BETWEEN 24 AND 48;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> SELECT * FROM policy_details WHERE tenure BETWEEN 20 AND 48;
OK
83473   trade   Health  34      20      30%
Time taken: 0.055 seconds, Fetched: 1 row(s)
hive>
```

23. Get each customer's name, policy purchased and its type

- SELECT c.name, pd.name, pd.type FROM licdw.customer c INNER JOIN policy_sales ps ON ps.cust_id = c.id INNER JOIN policy_details pd ON pd.id = ps.policy_id;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
Stage-Stage-5: Map: 1   Cumulative CPU: 1.25 sec   HDFS Read: 8509 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 250 msec
OK
suman   backwardhome    home
foram   union   Motor
savri   Lpg     fire
siddhesh        smallvillage    travel
jayesh  trade   Health
Time taken: 32.253 seconds, Fetched: 5 row(s)
hive>
```

24. Display all policy types using distinct clause

- SELECT DISTINCT type FROM policy_details;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.85 sec   HDFS Read: 6828 HDFS Write: 30 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 850 msec
OK
Health
Motor
fire
home
travel
Time taken: 31.904 seconds, Fetched: 5 row(s)
hive>
```

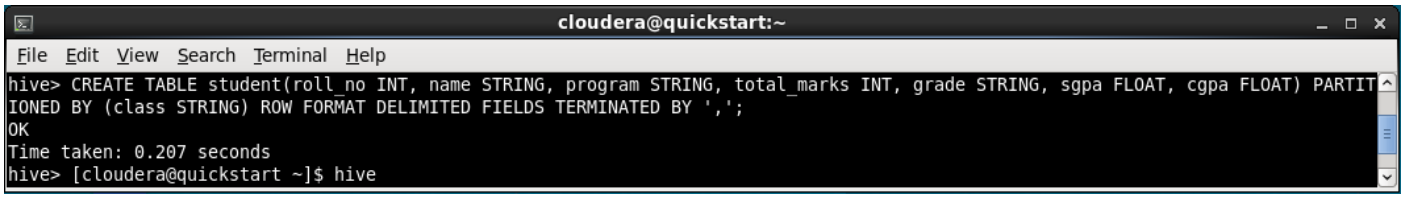
Partitioning Hive Tables

1. Create and use a database named ad ‘college’

- CREATE DATABASE college;
- USE college;

2. Create the partitioned table named ‘student’ to store the students information that is partitioned by the class values

- CREATE TABLE student(roll_no INT, name STRING, program STRING, total_marks INT, grade STRING, sgpa FLOAT, cgpa FLOAT) PARTITIONED BY (class STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';



3. Create a csv file for student information and load it in the Hive table

- gedit studinfo.csv

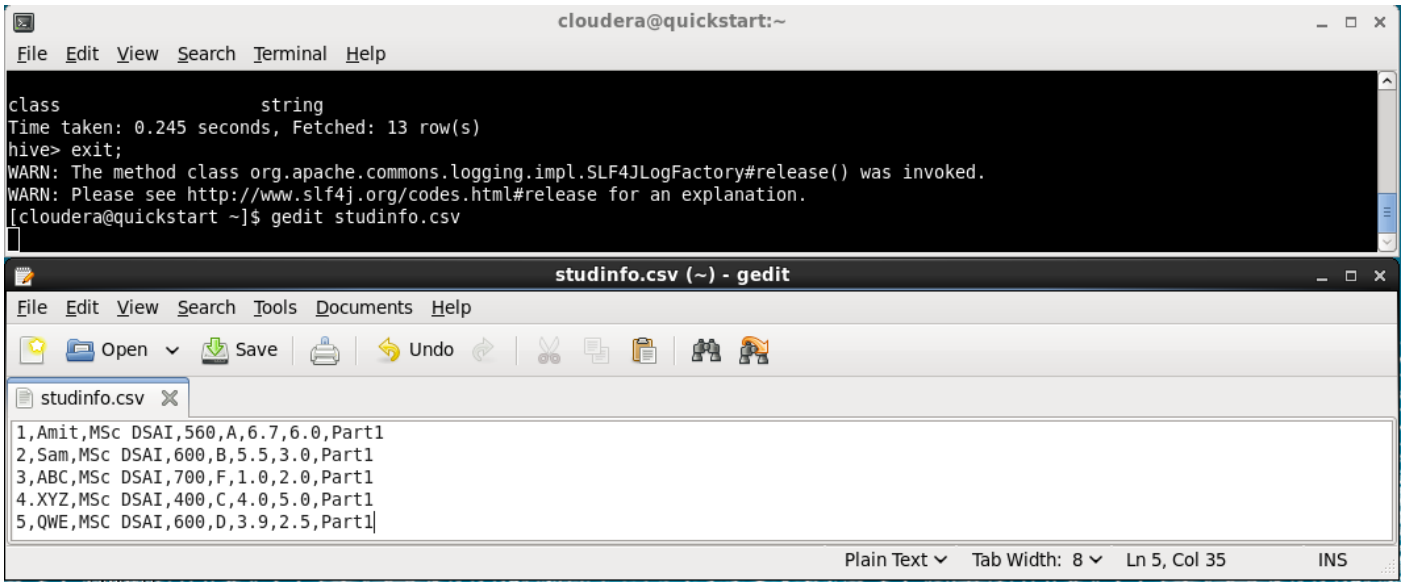
1,Amit,MSc DSAI,560,A,6.7,6.0,Part1

2,Sam,MSc DSAI,600,B,5.5,3.0,Part1

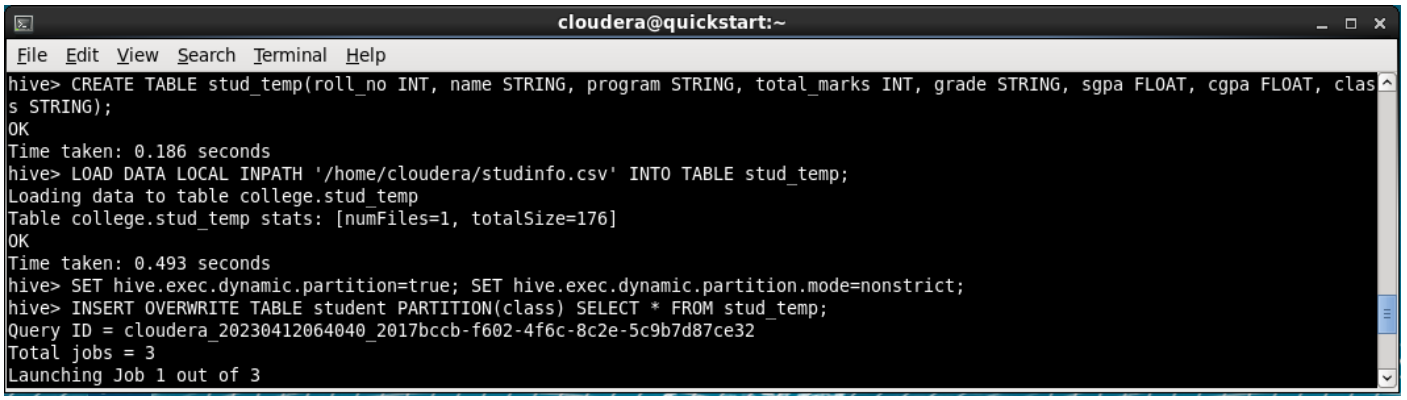
3,ABC,MSc DSAI,700,F,1.0,2.0,Part1

4.XYZ,MSc DSAI,400,C,4.0,5.0,Part1

5,QWE,MSC DSAI,600,D,3.9,2.5,Part1



- CREATE TABLE stud_temp(roll_no INT, name STRING, program STRING, total_marks INT, grade STRING, sgpa FLOAT, cgpa FLOAT, class STRING);
- LOAD DATA LOCAL INPATH '/home/cloudera/studinfo.csv' INTO TABLE stud_temp;
- SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict;
- INSERT OVERWRITE TABLE student PARTITION(class) SELECT * FROM stud_temp;



4. Display the schema with and without Formatted option

- DESCRIBE FORMATTED student;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> DESCRIBE FORMATTED student;
OK
# col_name              data_type               comment

roll_no                 int
name                    string
program                 string
total_marks             int
grade                   string
sgpa                    float
cgpa                    float

# Partition Information
# col_name              data_type               comment

class                   string

# Detailed Table Information
Database:               college
Owner:                  cloudera
CreateTime:             Wed Apr 12 06:21:48 PDT 2023
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://quickstart.cloudera:8020/user/hive/warehouse/college.db/student
Table Type:             MANAGED_TABLE
```

- DESCRIBE student;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> DESCRIBE student;
OK
roll_no                 int
name                    string
program                 string
total_marks             int
grade                   string
sgpa                    float
cgpa                    float
class                   string

# Partition Information
# col_name              data_type               comment

class                   string
Time taken: 0.084 seconds, Fetched: 13 row(s)
hive>
```

5. Display all partitions of the student table

- SHOW PARTITIONS student;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> SHOW PARTITIONS student;
OK
class=__HIVE_DEFAULT_PARTITION__
Time taken: 0.073 seconds, Fetched: 1 row(s)
hive>
```

6. Create a partitioned table named 'StudProg' that is partitioned by the program and class

- CREATE TABLE stud_prog(roll_no INT, name STRING) PARTITIONED BY (program STRING, class STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE TABLE stud_prog(roll_no INT, name STRING) PARTITIONED BY (program STRING, class STRING) ROW FORMAT DELIMITED FIELDS T
ERMINATED BY ','
    > ;
OK
Time taken: 0.15 seconds
hive>
```

7. Display all partitions of the StudProg table

- DESCRIBE stud_prog;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> DESCRIBE stud_prog;
OK
roll_no                 int
name                    string
program                 string
class                   string

# Partition Information
# col_name              data_type               comment

program                 string
class                   string
Time taken: 0.054 seconds, Fetched: 10 row(s)
hive>
```

8. Display all partitions of the StudProg table from HDFS

- hdfs dfs -ls ‘/user/hive/warehouse/college.db/student’

```
cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/college.db/student
Found 1 items
drwxrwxrwx   - cloudera hive          0 2023-04-12 06:41 /user/hive/warehouse/college.db/student/class=__HIVE_DEFAULT_PARTITION__
[cloudera@quickstart ~]$
```

9. Insert data in partitioned hive table ‘student’ using insert statement

- INSERT INTO student PARTITION(class='Part1') VALUES(6,'asdf','MSC DSAI',350,0.6,2.5,'Part1');

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> INSERT INTO student PARTITION(class='Part1') VALUES(6,'asdf','MSC DSAI',350,0.6,2.5,'Part1');
Query ID = cloudera_20230412072929_4a9cdb2c-5323-481c-becf-e18e2c726e85
Total jobs = 3
Launching Job 1 out of 3
```

10. Display all records of the hive table named student

Hive Queries

1. Create and use a database named ‘jdk’.

- CREATE DATABASE jdk;
- USE jdk;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> CREATE DATABASE jdk;
OK
Time taken: 0.025 seconds
hive> USE jdk;
OK
Time taken: 0.016 seconds
hive>
```

2. Create a table named ‘people’ with the following attributes.

Attributes:

Name, mobileNumber, dob,salary

- CREATE TABLE people(name STRING, m_no STRING, dob DATE, salary INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

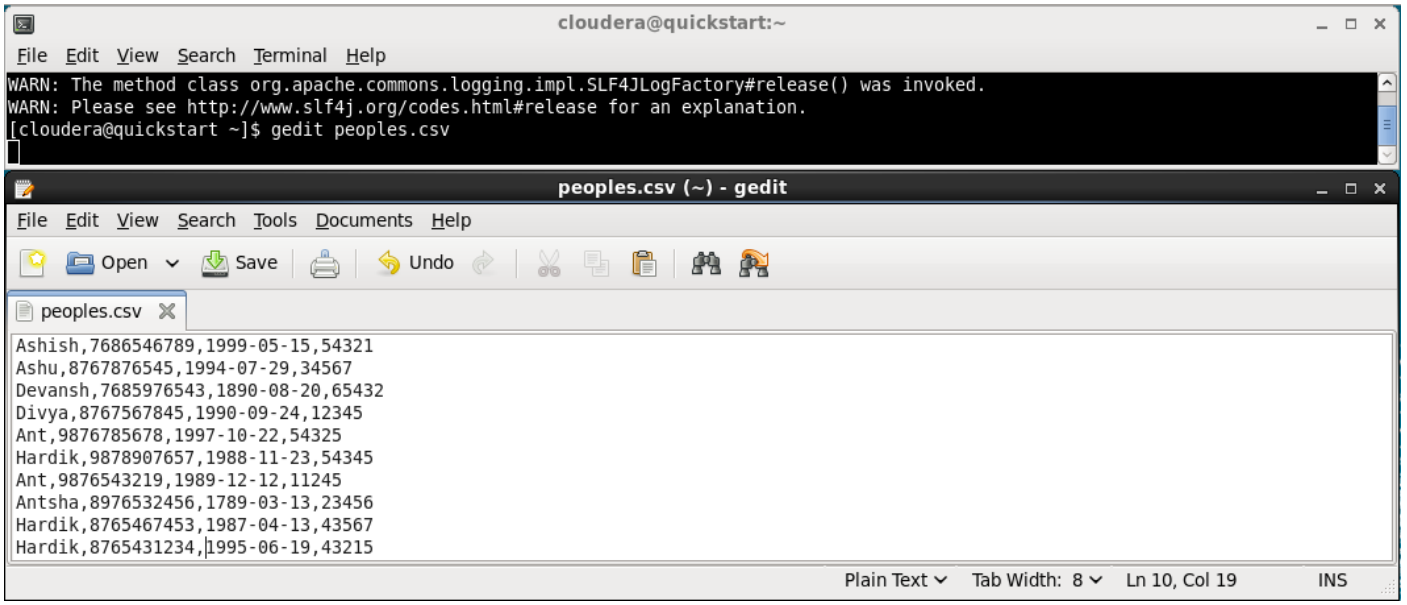
```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
hive> CREATE TABLE people(name STRING, m_no STRING, dob DATE, salary INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 0.085 seconds
hive>
```

3. Create a csv file and save the data.

- gedit peoples.csv

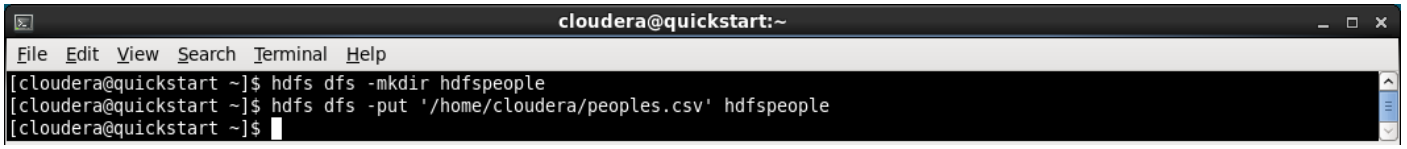
Ashish,7686546789,1999-05-15,54321
Ashu,8767876545,1994-07-29,34567

Devansh,7685976543,1890-08-20,65432
Divya,8767567845,1990-09-24,12345
Ant,9876785678,1997-10-22,54325
Hardik,9878907657,1988-11-23,54345
Ant,9876543219,1989-12-12,11245
Antsha,8976532456,1789-03-13,23456
Hardik,8765467453,1987-04-13,43567
Hardik,8765431234,1995-06-19,43215



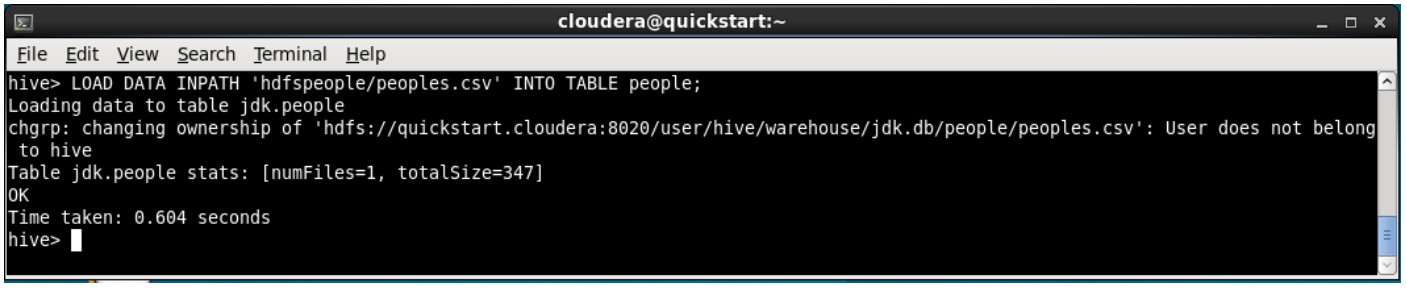
4. Copy the above csv file in the hdfs directory named ‘hdfspeople’.

- hdfs dfs -mkdir hdfspeople
- hdfs dfs -put '/home/cloudera/peoples.csv' hdfspeople



5. Load data in hive table ‘people’ .

- LOAD DATA INPATH 'hdfspeople/peoples.csv' INTO TABLE people;



6. Display all records of the ‘people’ table.

- SELECT * FROM people;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT * FROM people;
OK
Ashish  7686546789      1999-05-15      54321
Ashu    8767876545      1994-07-29      34567
Devansh 7685976543      1890-08-20      65432
Divya   8767567845      1990-09-24      12345
Ant     9876785678      1997-10-22      54325
Hardik  9878907657      1988-11-23      54345
Ant     9876543219      1989-12-12      11245
Antsha  8976532456      1789-03-13      23456
Hardik  8765467453      1987-04-13      43567
Hardik  8765431234      1995-06-19      43215
Time taken: 0.309 seconds, Fetched: 10 row(s)
hive>
```

7. Display name and dob from ‘people’ table.

- SELECT name, dob FROM people;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> SELECT name, dob FROM people;
OK
Ashish  1999-05-15
Ashu    1994-07-29
Devansh 1890-08-20
Divya   1990-09-24
Ant     1997-10-22
Hardik  1988-11-23
Ant     1989-12-12
Antsha  1789-03-13
Hardik  1987-04-13
Hardik  1995-06-19
Time taken: 0.068 seconds, Fetched: 10 row(s)
hive>
```

8. Display name and salary from ‘people’ table in an ascending order of name column .

- SELECT name, salary FROM people ORDER BY name ASC;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
Total MapReduce CPU Time Spent: 1 seconds 850 msec
OK
Ant     11245
Ant     54325
Antsha  23456
Ashish  54321
Ashu    34567
Devansh 65432
Divya   12345
Hardik  43215
Hardik  43567
Hardik  54345
Time taken: 29.336 seconds, Fetched: 10 row(s)
hive>
```

9. Display name and salary from ‘people’ table in descending order of salary column.

- SELECT name, salary FROM people ORDER BY salary DESC;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
Total MapReduce CPU Time Spent: 1 seconds 710 msec
OK
Devansh 65432
Hardik  54345
Ant     54325
Ashish  54321
Hardik  43567
Hardik  43215
Ashu    34567
Antsha  23456
Divya   12345
Ant     11245
Time taken: 23.069 seconds, Fetched: 10 row(s)
hive>
```

10. Display distinct date of births from ‘people’ table.

- SELECT DISTINCT dob FROM people;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
Total MapReduce CPU Time Spent: 1 seconds 770 msec
OK
1789-03-13
1890-08-20
1987-04-13
1988-11-23
1989-12-12
1990-09-24
1994-07-29
1995-06-19
1997-10-22
1999-05-15
Time taken: 23.142 seconds, Fetched: 10 row(s)
hive>
```

11. Display the person with maximum salary. :

- SELECT name, salary FROM people p1 WHERE p1.salary IN (SELECT MAX(salary) FROM people);

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
Total MapReduce CPU Time Spent: 3 seconds 330 msec
OK
Devansh 65432
Time taken: 49.594 seconds, Fetched: 1 row(s)
hive>
```

12. Find and display the average salary.

- SELECT AVG(salary) FROM people;

```
cloudera@quickstart:~
File Edit View Search Terminal Help
Total MapReduce CPU Time Spent: 1 seconds 830 msec
OK
39681.8
Time taken: 25.069 seconds, Fetched: 1 row(s)
hive>
```

13. Display second highest salary.

- SELECT DISTINCT salary FROM (SELECT salary, DENSE_RANK() OVER(ORDER BY salary DESC) as rank FROM people) t WHERE rank = 2;

```
54345
Time taken: 48.529 seconds, Fetched: 1 row(s)
hive>
```

14. Create and partition a table named ‘emppartition’ on the mobile number column and load data into it.

- CREATE TABLE emppartition(emd_id INT, name STRING) PARTITIONED BY(m_no STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
- Set hive.exec.dynamic.partition.mode=nonstrict;
- INSERT INTO emppartition PARTITION(m_no) VALUES(1, ‘sam’, ‘1234567890’);

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE TABLE emppartition(emd_id INT, name STRING) PARTITIONED BY(m_no STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
;
OK
Time taken: 0.158 seconds
hive> INSERT INTO emppartition PARTITION(m_no) VALUES(1, 'sam', '1234567890');
FAILED: SemanticException [Error 10096]: Dynamic partition strict mode requires at least one static partition column. To turn this
 off set hive.exec.dynamic.partition.mode=nonstrict
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> INSERT INTO emppartition PARTITION(m_no) VALUES(1, 'sam', '1234567890');
Query ID = cloudera_20230412122121_4eed9b5e-eec2-473f-9bce-fb5d0efba1b8
```

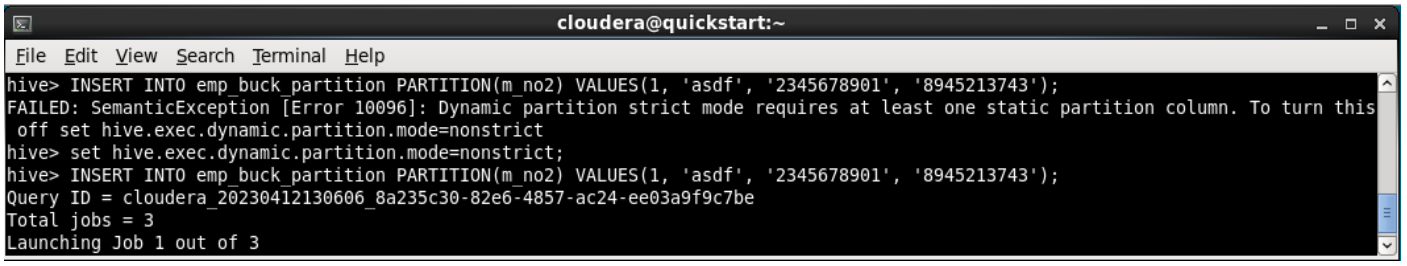
15. Create a table named ‘empbuckNoPartition’ with only bucketing on the mobile number column and load data into it.

- CREATE TABLE emp_buck_no_partition(emp_id INT, name STRING, m_no STRING) CLUSTERED BY (m_no) INTO 5 BUCKETS;
- INSERT INTO emp_buck_no_partition VALUES(1, 'xyz', '9283746251');

```
cloudera@quickstart:~
File Edit View Search Terminal Help
hive> CREATE TABLE emp_buck_no_partition(emp_id INT, name STRING, m_no STRING) CLUSTERED BY (m_no) INTO 5 BUCKETS;
OK
Time taken: 0.061 seconds
hive> INSERT INTO emp_buck_no_partition VALUES(1, 'xyz', '9283746251');
Query ID = cloudera_20230412125858_27200d23-6e51-417f-8d0d-d0df0165e5cb
Total jobs = 3
Launching Job 1 out of 3
```

16. Create a table named ‘empbuckwithPartition’ with partitioning and bucketing on the mobile number column and load data into it.

- CREATE TABLE emp_buck_partition(emp_id INT, name STRING, m_no STRING) PARTITIONED BY(m_no2 STRING) CLUSTERED BY(m_no) INTO 5 BUCKETS;
- set hive.exec.dynamic.partition.mode=nonstrict;
- INSERT INTO emp_buck_partition PARTITION(m_no2) VALUES(1, 'asdf', '2345678901', '8945213743');



17. Display data from ‘emppartition’ , ‘empbuckNoPartition’ and ‘empbuckwithPartition’ tables.

- SELECT * FROM emppartition;
- SELECT * FROM emp_buck_no_partition;
- SELECT * FROM emp_buck_partition;