**BDT Practical 6**

**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.

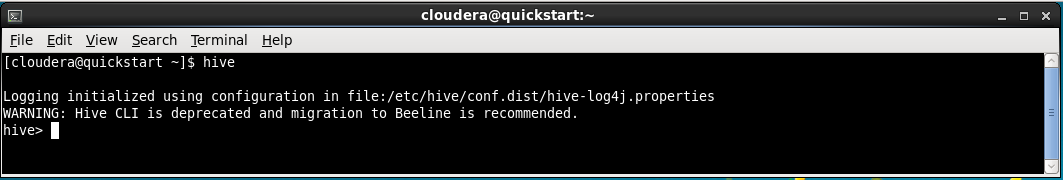
**To Begin:**

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

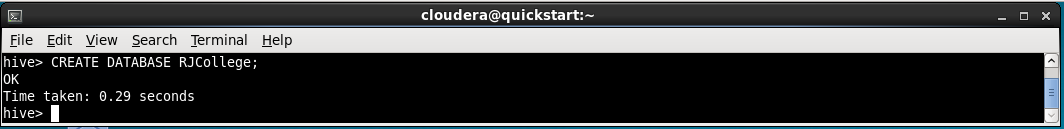


**Example 1:**

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

To create a database in Hive we use the command

* CREATE DATABASE RJCollege;

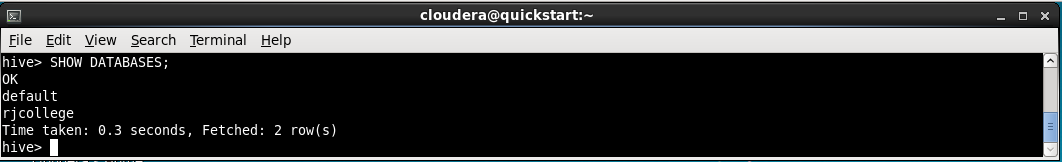


**2. Check the creation of a data warehouse database.**

To check if the database was created we use the command

* SHOW DATABASES;

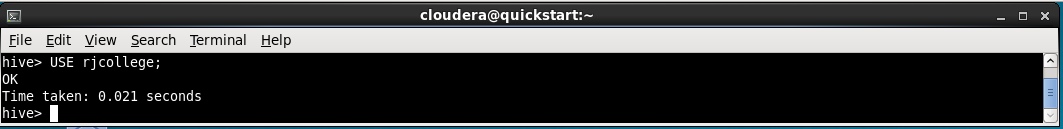
This will return a list of databases which should contain our newly created database RJCollege



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

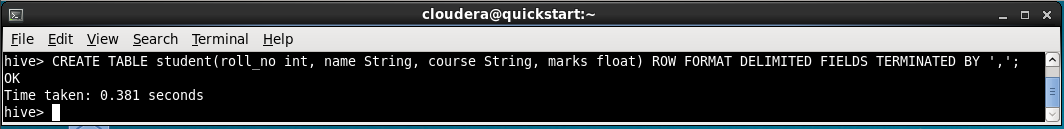
To create a table inside our database we first need to use or activate our database using command

* USE rjcollege;



Then to create a table named student we use command

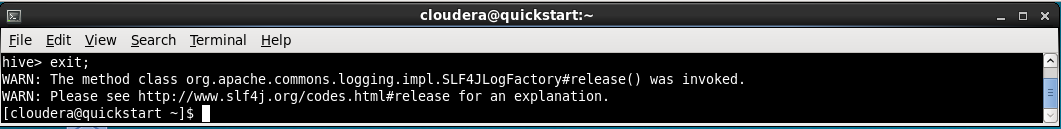
* CREATE TABLE student(roll\_no int, name String, course String, marks float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’;



**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

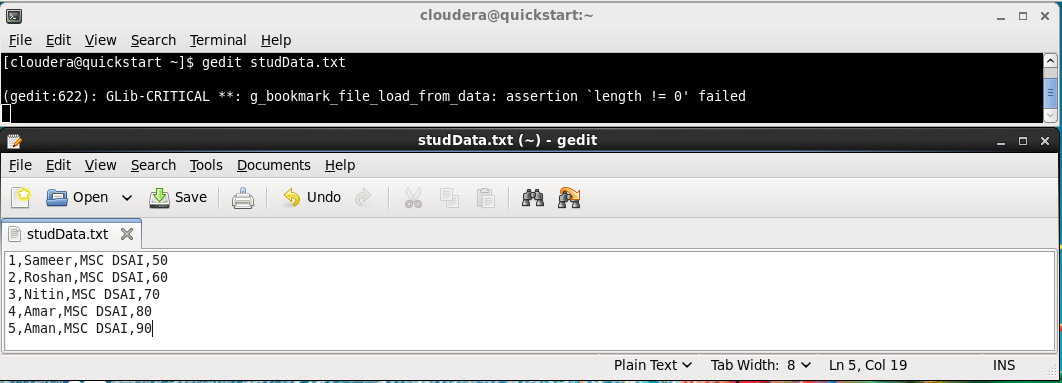
1,Sameer,MSC DSAI,50

2,Roshan, MSC DSAI, 60

3,Nitin,MSC DSAI,70

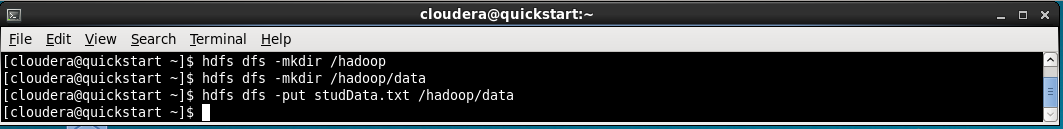
4,Amar,MSC DSAI,80

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

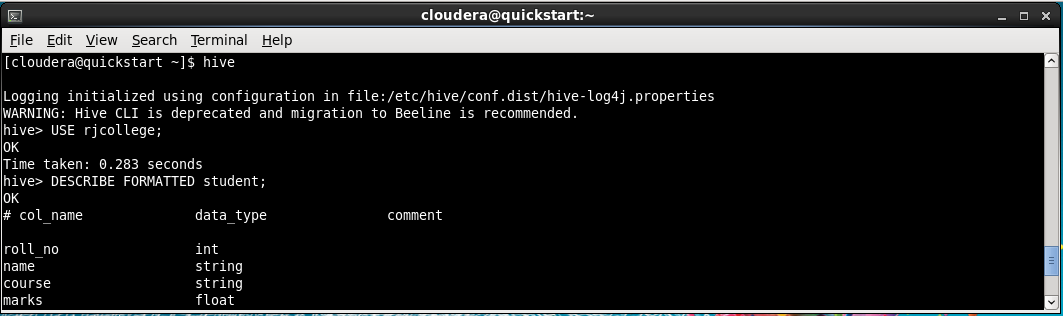
* hdfs dfs -mkdir /hadoop
* hdfs dfs -mkdir /hadoop/data
* hdfs dfs -put studData.txt /hadoop/data



**5. Display the schema of the student table**

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

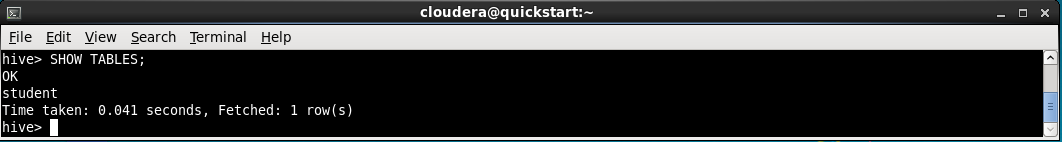
* DESCRIBE FORMATTED rjcollege;



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

To get a list of tables, run command

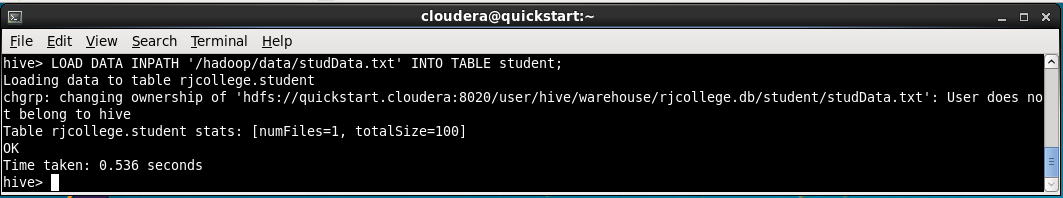
* SHOW TABLES;



**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

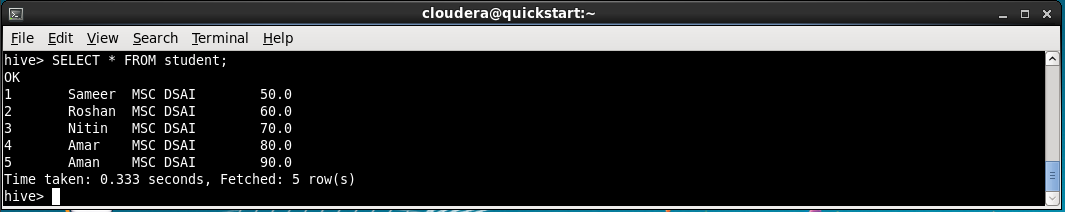
* LOAD DATA INPATH ‘/hadoop/data/studData.txt’ INTO TABLE student;



**8. Display all students information/results**

To display the data of student table we use

* SELECT \* FROM student;



**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

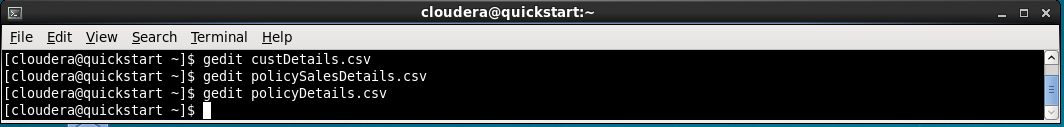
83473,trade,Health,34,20,30%

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

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

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

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

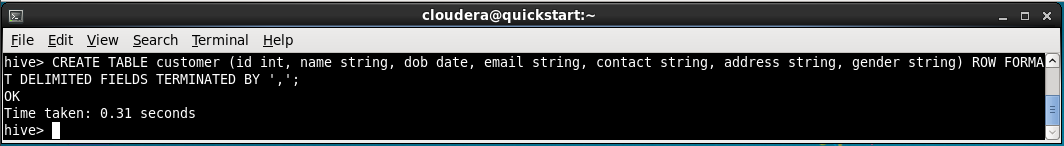


**2. Create a data warehouse database named ‘LICDW’ using Hive**

* CREATE DATABASE licdw;

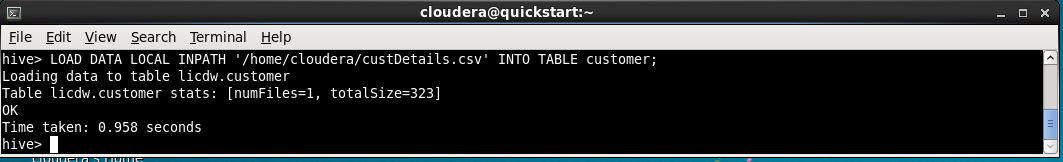
**3. Create an Internal/managed table for CustDetails using Hive**

* 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**

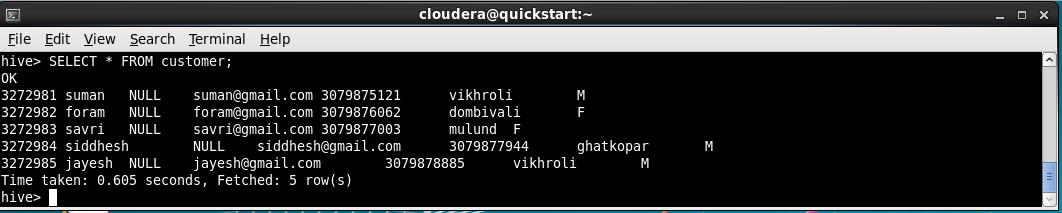
* LOAD DATA LOCAL INPATH '/home/cloudera/custDetails.csv' INTO TABLE customer;



**cre**

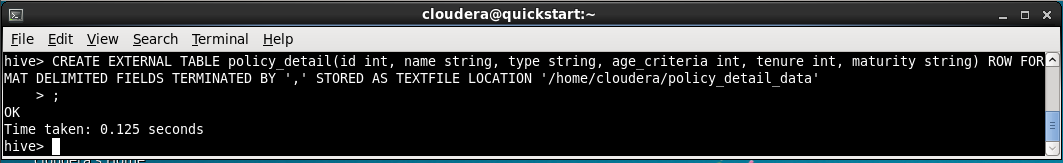
**5. Display all records of custDetails table**

* SELECT \* FROM customer;



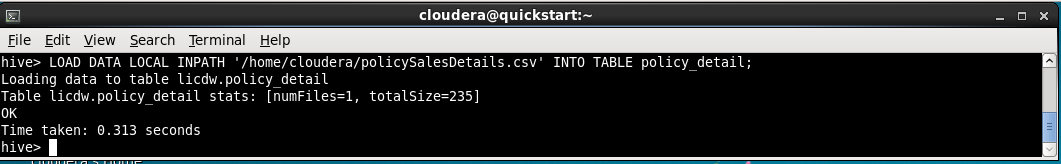
**6. Create a policySaleDetails table as an external table using Hive**

* 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';

****

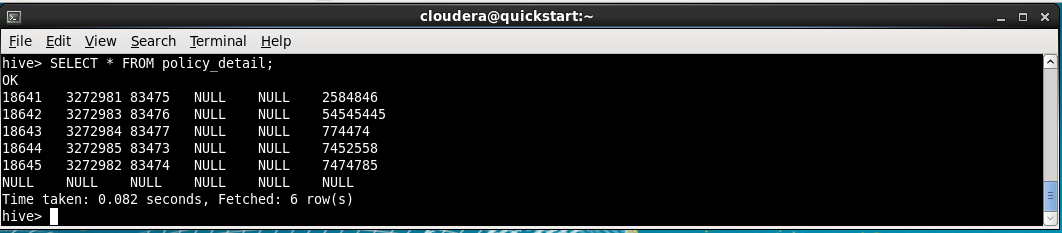
**7. Load the data of PolicySaleDetailsData.csv file to PolicySaleDetails table**

* LOAD DATA LOCAL INPATH '/home/cloudera/policySalesDetails.csv' INTO TABLE policy\_detail;

****

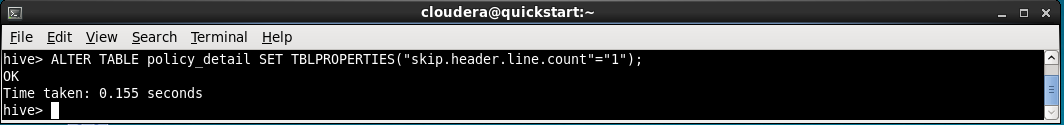
**8. Display the schema and data details of the PolicySaleDetails table**

* SELECT \* FROM policy\_detail;

****

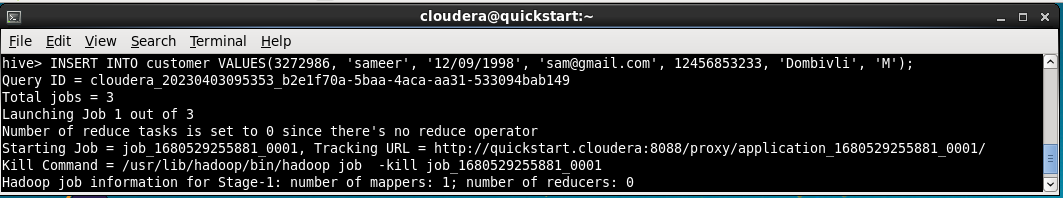
**9. Skip header line of dataset file while loading data in Hive table**

* ALTER TABLE policy\_detail SET TBLPROPERTIES("skip.header.line.count"="1");

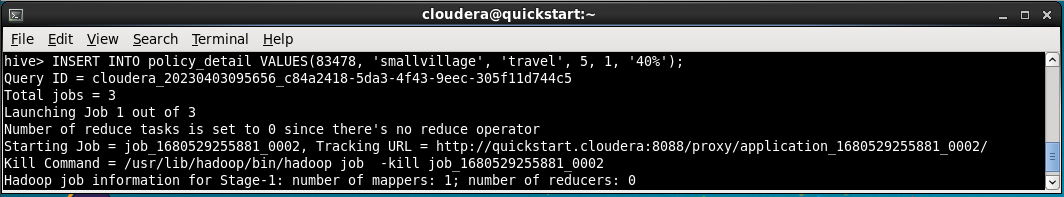
****

**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');

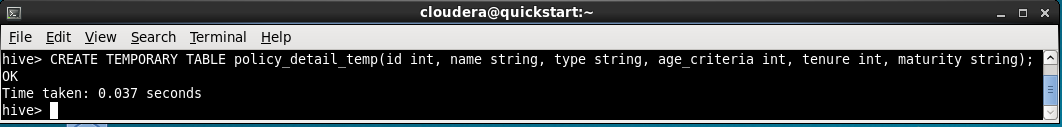


* INSERT INTO policy\_detail VALUES(83478, 'smallvillage', 'travel', 5, 1, '40%');



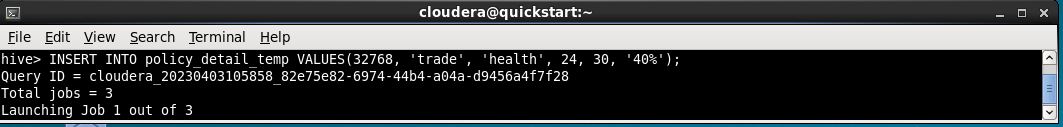
**11. Create the PolicyDetails temporary table**

* CREATE TEMPORARY TABLE policy\_detail\_temp(id int, name string, type string, age\_criteria int, tenure int, maturity string

****

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

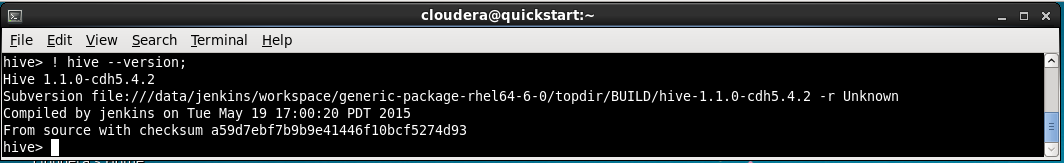
* INSERT INTO policy\_detail\_temp VALUES(32768, 'trade', 'health', 24, 30, '40%');
* INSERT INTO policy\_detail\_temp VALUES(32769, 'union', 'motor', 18, 5, '30%');



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

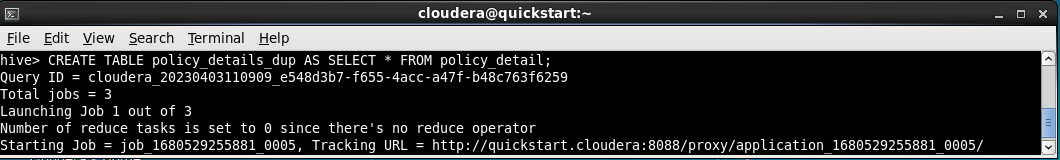
We can check hive version using command

* ! hive --version



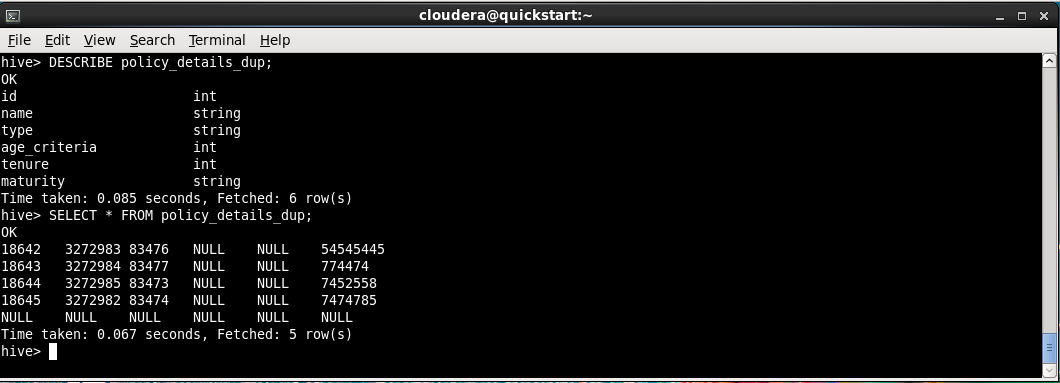
**14. Create PolicyDetailsDup table using PolicyDetails table, using CTAS statement**

* CREATE TABLE policy\_details\_dup AS SELECT \* FROM policy\_detail;

****

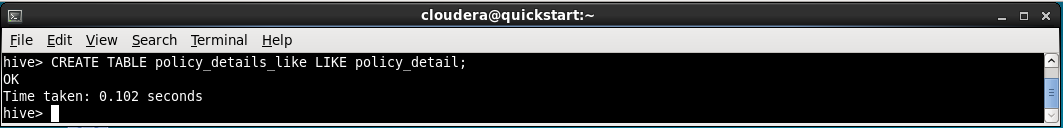
**15. Execute describe and  select statements for PolicyDetailsDup table**

* DESCRIBE policy\_details\_dup;
* SELECT \* FROM policy\_details\_dup;

****

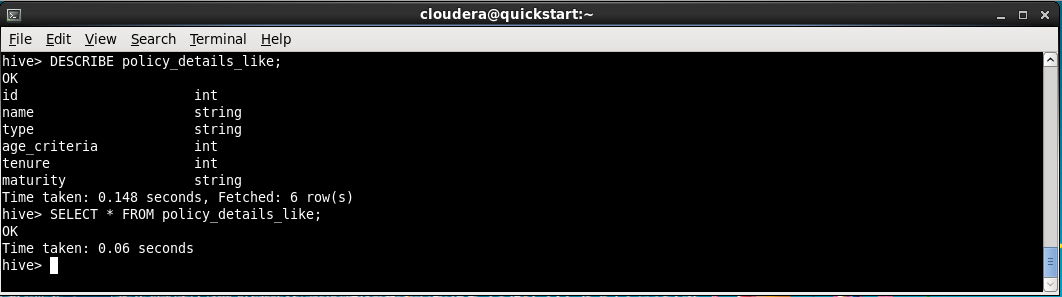
**16. Create a PolicyDetailsLike table using the existing PolicyDetails table**

* CREATE TABLE policy\_details\_like LIKE policy\_detail;

****

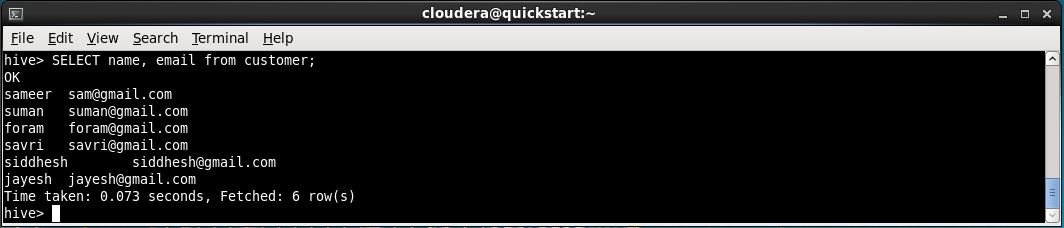
**17. Execute describe and  select statements for PolicyDetailsLike table**

* DESCRIBE policy\_details\_like;
* SELECT \* FROM policy\_details\_like;

****

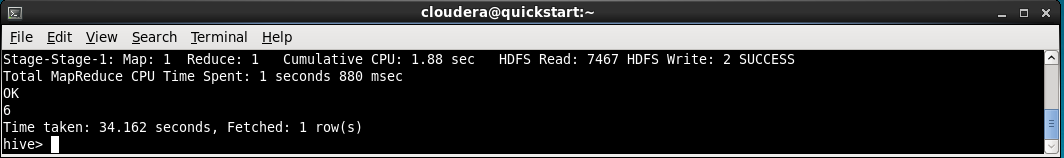
**18. Display the list of customers and their mail ids from the custDetails table**

* SELECT name, email FROM customer;

****

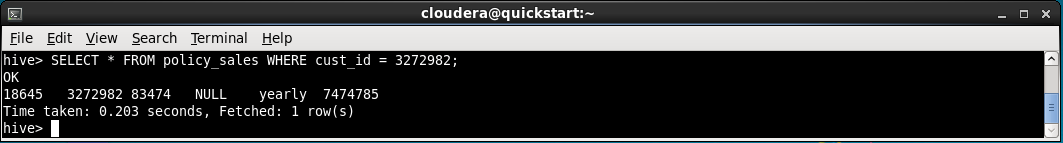
**19. Get the count of the total number of customers**

* SELECT COUNT(\*) FROM customer;

****

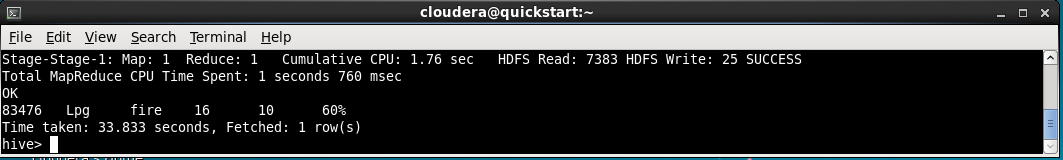
**20. Display the premium paid details of customer having id 3272982**

* SELECT \* FROM policy\_sales WHERE cust\_id = 3272982;

****

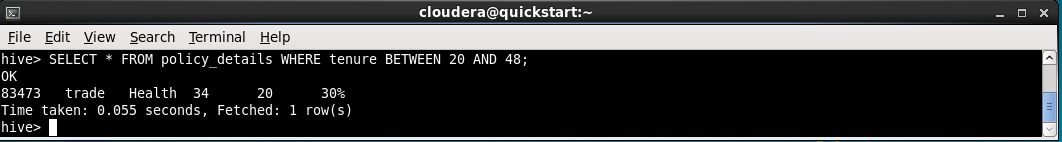
**21. Display the policy with maximum benefit**

* SELECT \* FROM policy\_detail ORDER BY maturity DESC LIMIT 1;



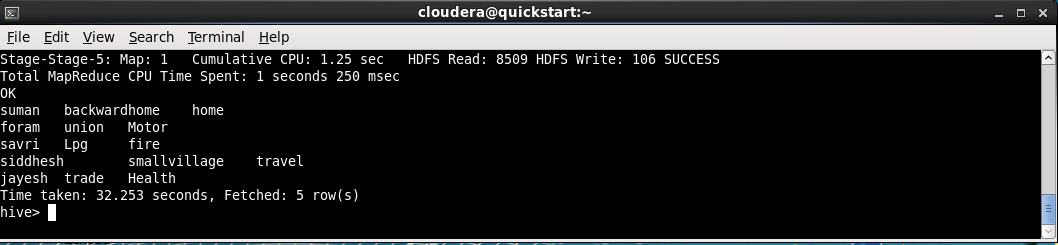
**22. Display the details of policy having tuners in the range of 24 to 48 months**

* SELECT \* FROM policy\_detail WHERE tenure BETWEEN 24 AND 48;

****

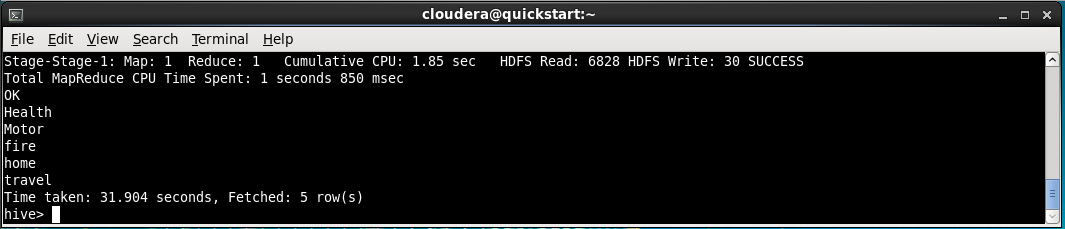
**23. Get each customer's name, policy purchased and its type**

* 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;



**24. Display all policy types using distinct clause**

* SELECT DISTINCT type FROM policy\_details;



**Partitioning Hive Tables**

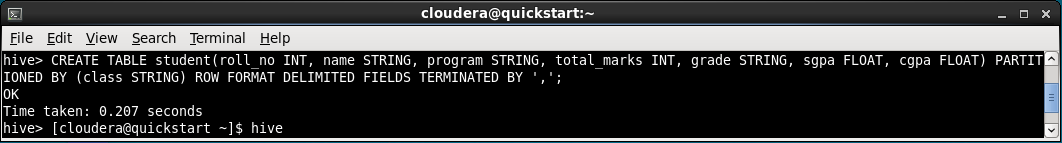
**1. Create and use a database named ad ‘college’**

* 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**

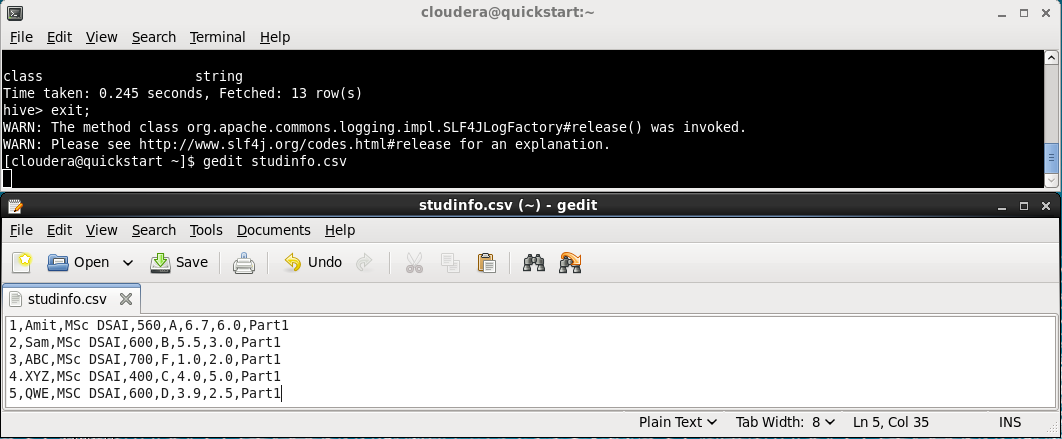
* gedit studinfo.csv

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

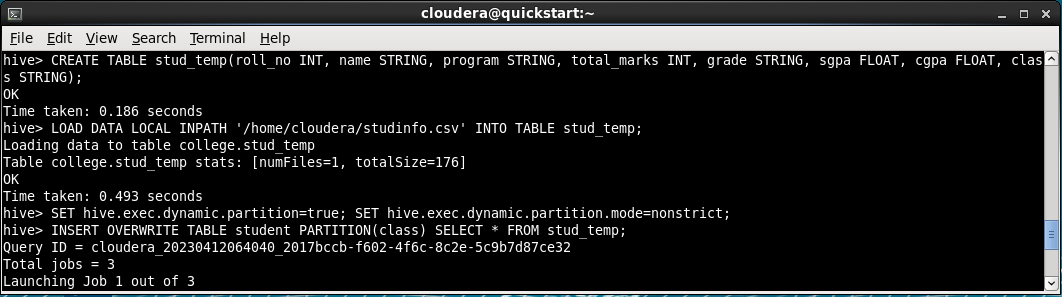
2,Sam,MSc DSAI,600,B,5.5,3.0,Part1

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

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

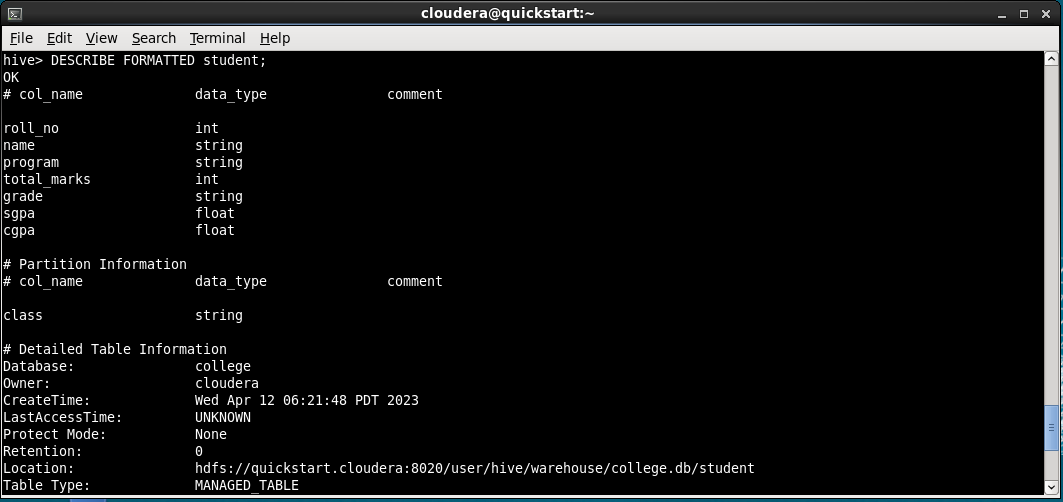
5,QWE,MSC DSAI,600,D,3.9,2.5,Part1

* 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;

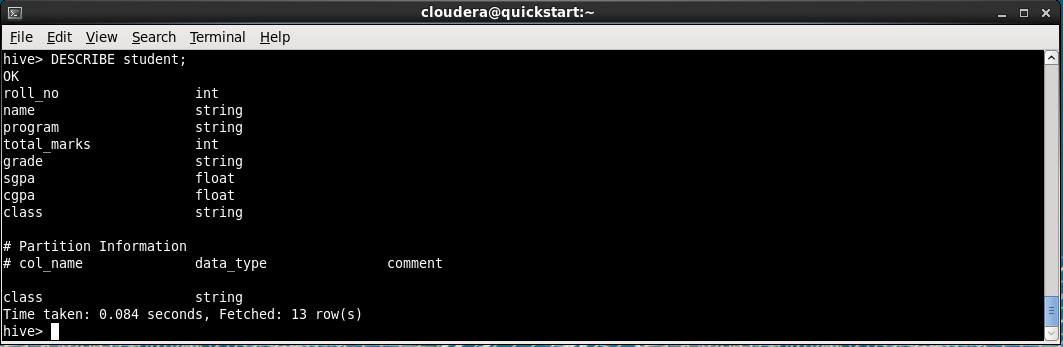


**4. Display the schema with and without Formatted option**

* DESCRIBE FORMATTED student;

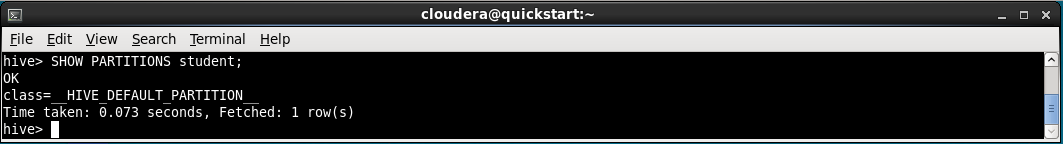


* DESCRIBE student;



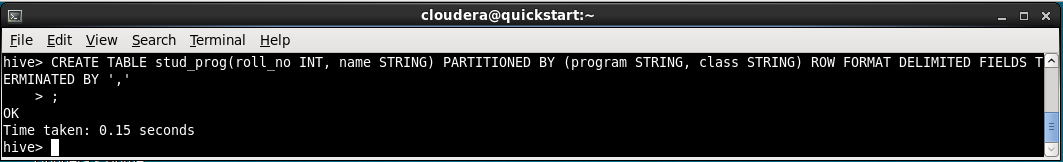
**5. Display all partitions of the student table**

* SHOW PARTITIONS student;



**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 ',';



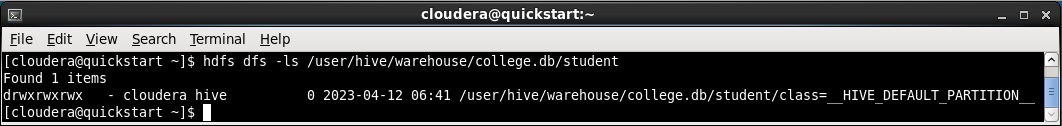
**7. Display all partitions of the StudProg table**

* DESCRIBE stud\_prog;



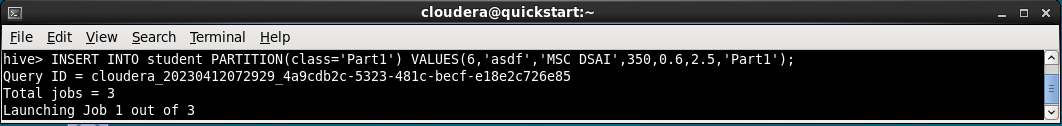
**8. Display all partitions of the StudProg table from HDFS**

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



**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');

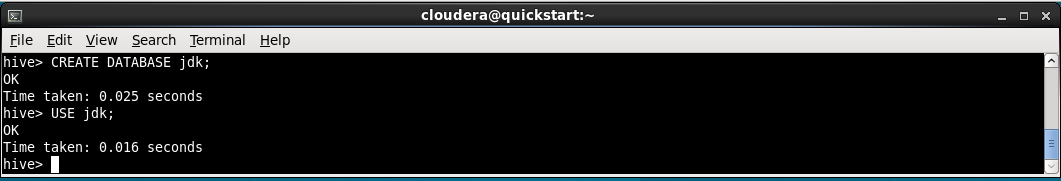


**10. Display all records of the hive table named student**

**Hive Queries**

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

* CREATE DATABASE jdk;
* USE jdk;

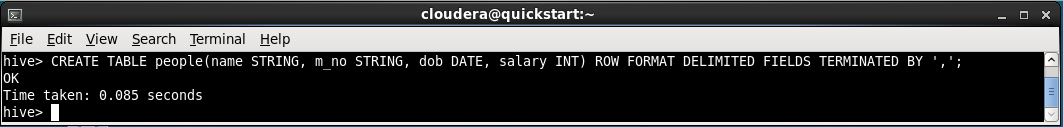


**2. Create a table named ‘people’ with the following attributes.**

**Attributes:**

**Name, mobileNumber, dob,salary**

* CREATE TABLE people(name STRING, m\_no STRING, dob DATE, salary INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';



**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

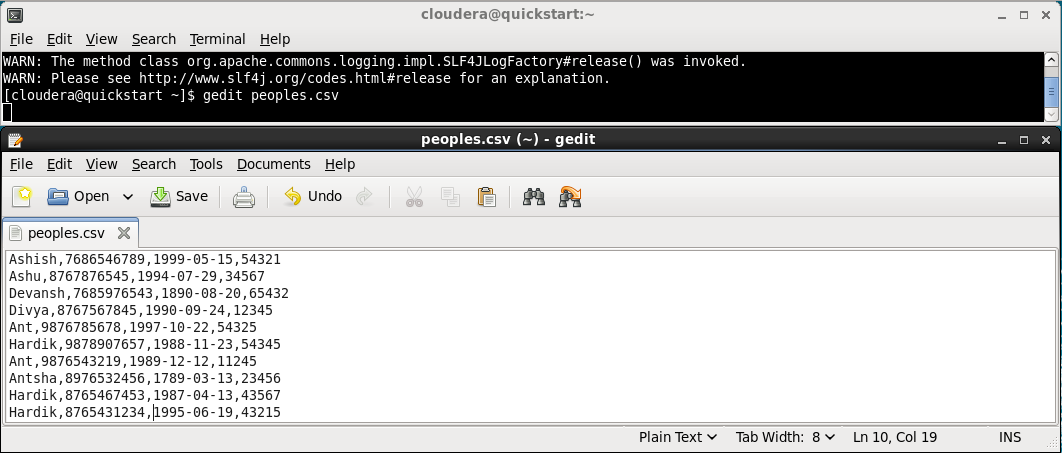
Ant,9876785678,1997-10-22,54325

Hardik,9878907657,1988-11-23,54345

Ant,9876543219,1989-12-12,11245

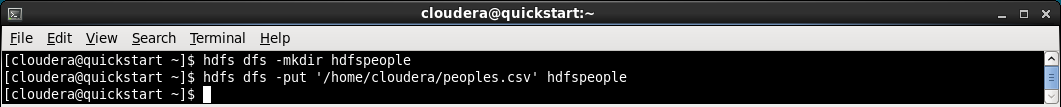
Antsha,8976532456,1789-03-13,23456

Hardik,8765467453,1987-04-13,43567

Hardik,8765431234,1995-06-19,43215

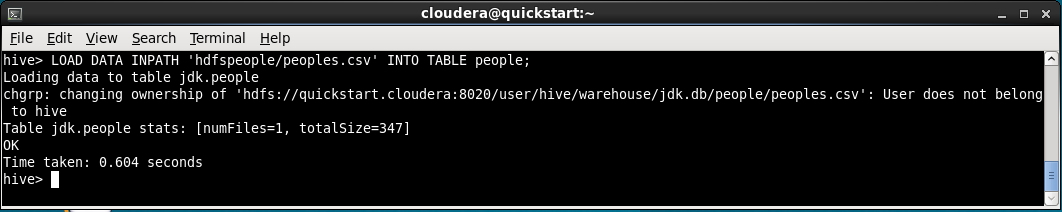
**4. Copy the above csv file in the hdfs directory named ‘hdfspeople’.**

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



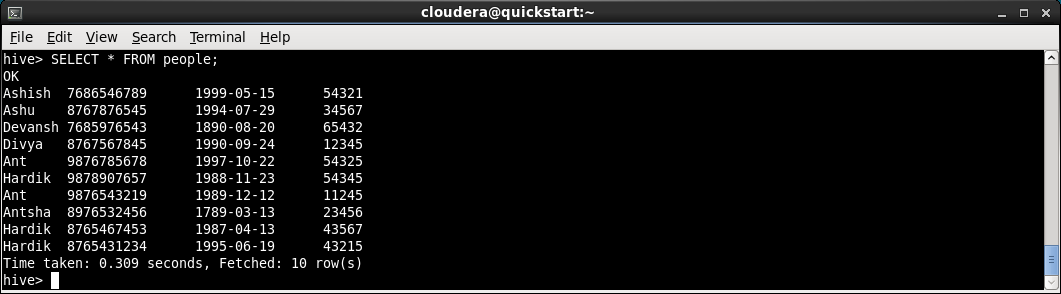
**5. Load data in hive table ‘people’ .**

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



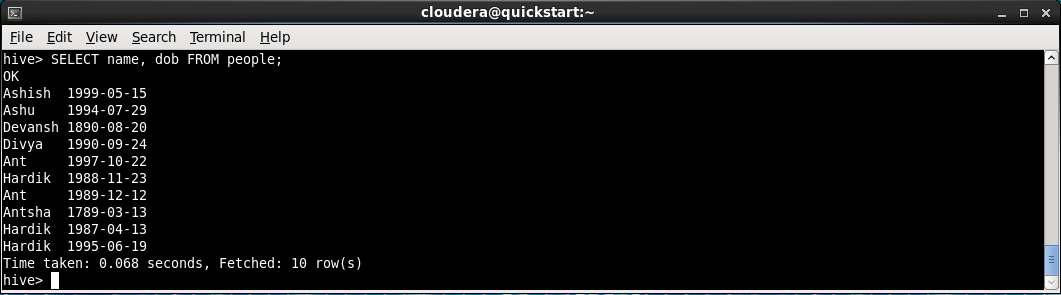
**6. Display all records of the ‘people’ table.**

* SELECT \* FROM people;



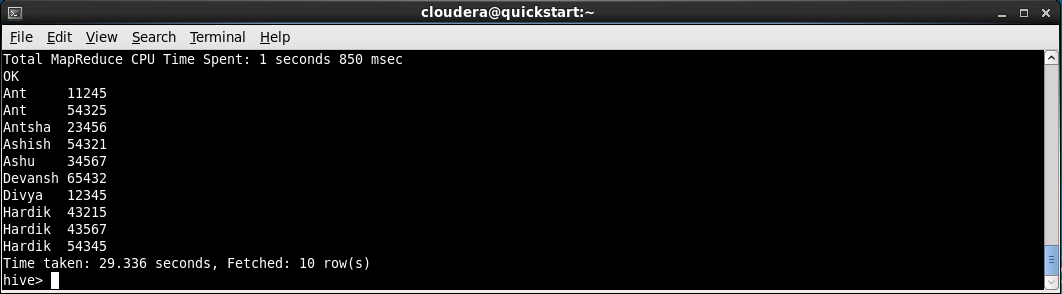
**7. Display name and dob from ‘people’ table.**

* SELECT name, dob FROM people;



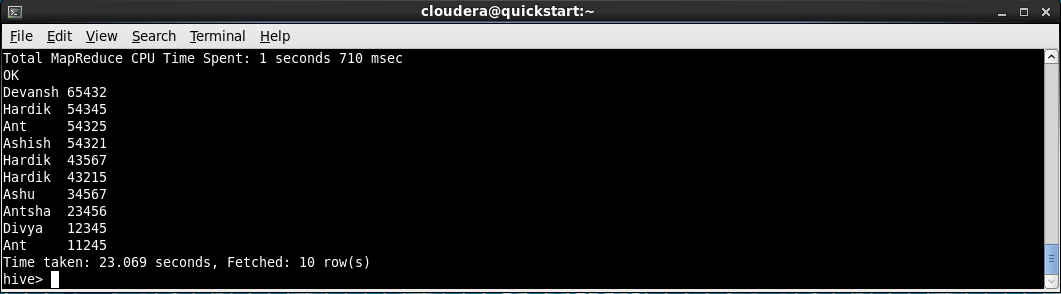
**8. Display name and salary from ‘people’ table in an ascending order of name column .**

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



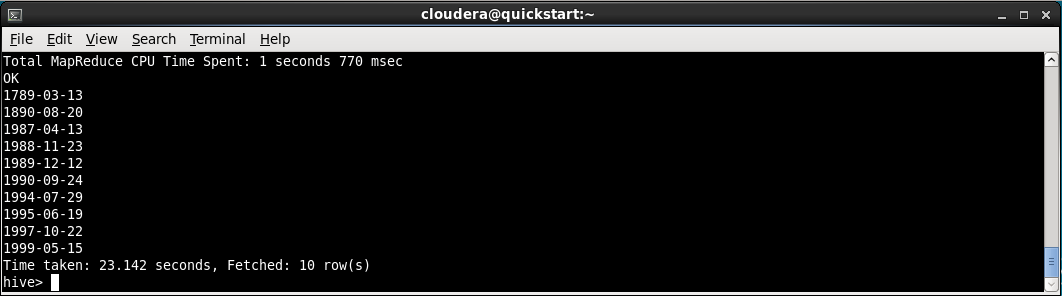
**9. Display name and salary from ‘people’ table in descending order of salary column.**

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



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

* SELECT DISTINCT dob FROM people;



**11. Display the person with maximum salary. :**

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



**12. Find and display the average salary.**

* SELECT AVG(salary) FROM people;



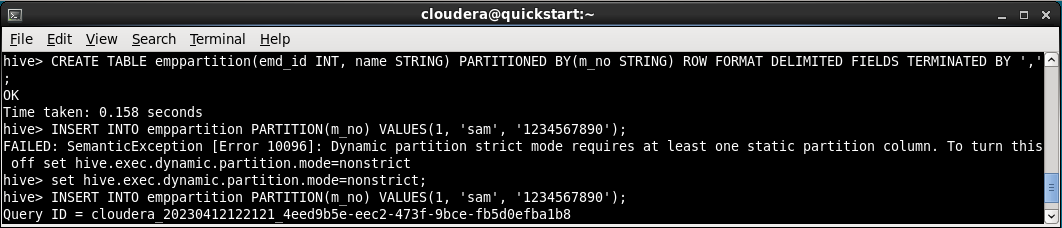
**13. Display second highest salary.**

* SELECT DISTINCT salary FROM (SELECT salary, DENSE\_RANK() OVER(ORDER BY salary DESC) as rank FROM people) t WHERE rank = 2;

****

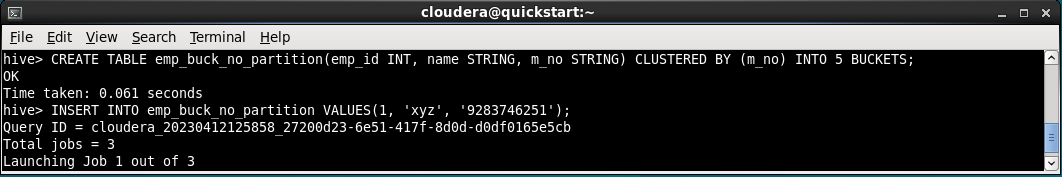
**14. Create and partition a table named ‘emppartition’ on the mobile number column and load data into it.**

* 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’);



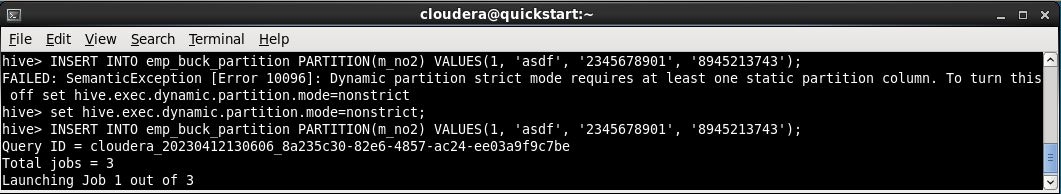
**15. Create a table named ‘empbuckNoPartition’ with only bucketing on the mobile number column and load data into it.**

* 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');



**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;

