ASSIGNMENT 7.3

Problem Statement:

Explain the below concepts with an example in brief.

- Hive Data Definitions
- Hive Data Manipulations
- HiveQL Manipulations

Answer:

- **Hive Data Definitions:**

Data definition part of Hive are mainly used for creating, altering and dropping databases, tables, indexes and views. Let's have a look at these operations briefly.

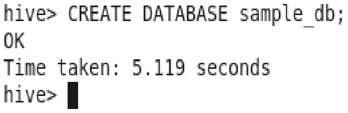
Database operations in Hive:

Database in Hive is a catalog or namespace of relations or tables. They are in fact useful with many users of multiple teams in large clusters. A *default* database will be used in case user don't specify any database.

Here is the syntax for creating a database in Hive:

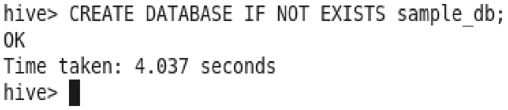
CREATE DATABASE <database name>;

Example:



```
hive> CREATE DATABASE sample_db;
OK
Time taken: 5.119 seconds
hive> █
```

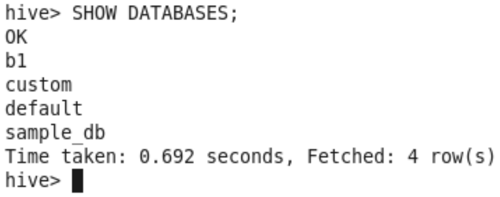
Hive will throw an error if sample_db already exists. We can suppress this warning by adding the clause, IF NOT EXISTS as shown below:



```
hive> CREATE DATABASE IF NOT EXISTS sample_db;
OK
Time taken: 4.037 seconds
hive> █
```

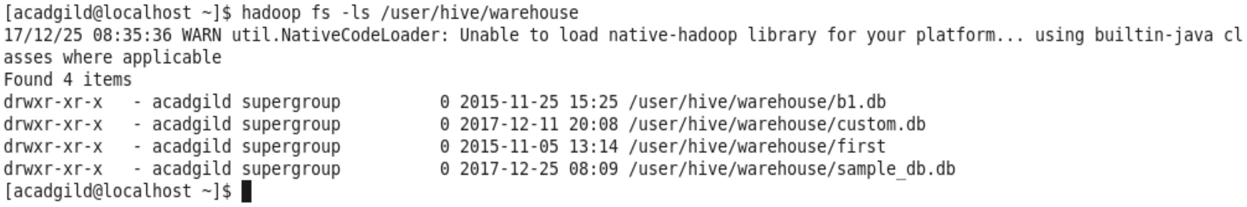
We can see a list of existing databases by following command:

SHOW DATABASES;



```
hive> SHOW DATABASES;
OK
b1
custom
default
sample_db
Time taken: 0.692 seconds, Fetched: 4 row(s)
hive> █
```

Hive will create a separate directory in HDFS for every database that gets created by a user. Tables in a database will be stored as subdirectories of database directory. The location of database directory is specified by this property: *hive.metastore.warehouse.dir,* and it can be modified by user at anytime. The default value for this property will be */user/hive/warehouse.*

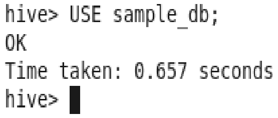


```
[acadgild@localhost ~]$ hadoop fs -ls /user/hive/warehouse
17/12/25 08:35:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 4 items
drwxr-xr-x   - acadgild supergroup          0 2015-11-25 15:25 /user/hive/warehouse/b1.db
drwxr-xr-x   - acadgild supergroup          0 2017-12-11 20:08 /user/hive/warehouse/custom.db
drwxr-xr-x   - acadgild supergroup          0 2015-11-05 13:14 /user/hive/warehouse/first
drwxr-xr-x   - acadgild supergroup          0 2017-12-25 08:09 /user/hive/warehouse/sample_db.db
[acadgild@localhost ~]$ █
```

When you launch hive from system terminal, by default, it points to *default* database and all the tables, views or indexes you create will be stored under this database. If you wish to work on a different database, you can do so with USE clause as shown below:

Syntax: USE <database name>;

Example:



```
hive> USE sample_db;
OK
Time taken: 0.657 seconds
hive> █
```

There is an option to drop a database when it is no longer required.

Syntax: DROP DATABASE IF EXISTS <database name>;

Example:

DROP DATABASE IF EXISTS sample_db;

Hive doesn't allow you to delete a database if it contains any tables. You either have to drop all tables before dropping a database or use CASCADE clause which causes Hive to drop tables and then drop the database as shown below:

DROP DATABASE IF EXISTS sample_db CASCADE;

Tables in Hive:

Hive offers to create tables of two types: managed tables (internal tables) and external tables

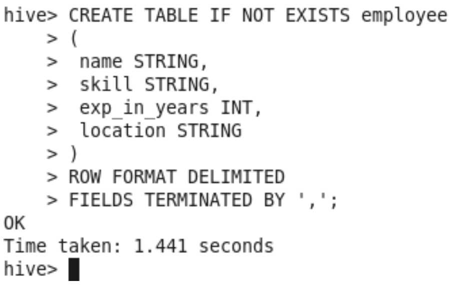
- Managed tables (internal tables)
- External tables

Managed tables: It is the default table in Hive. If we don’t specify as external while creating a table, it gets created as a managed table. When a user drops a table both the table data and metadata for that table will be deleted from the HDFS.

Syntax to create a managed table:

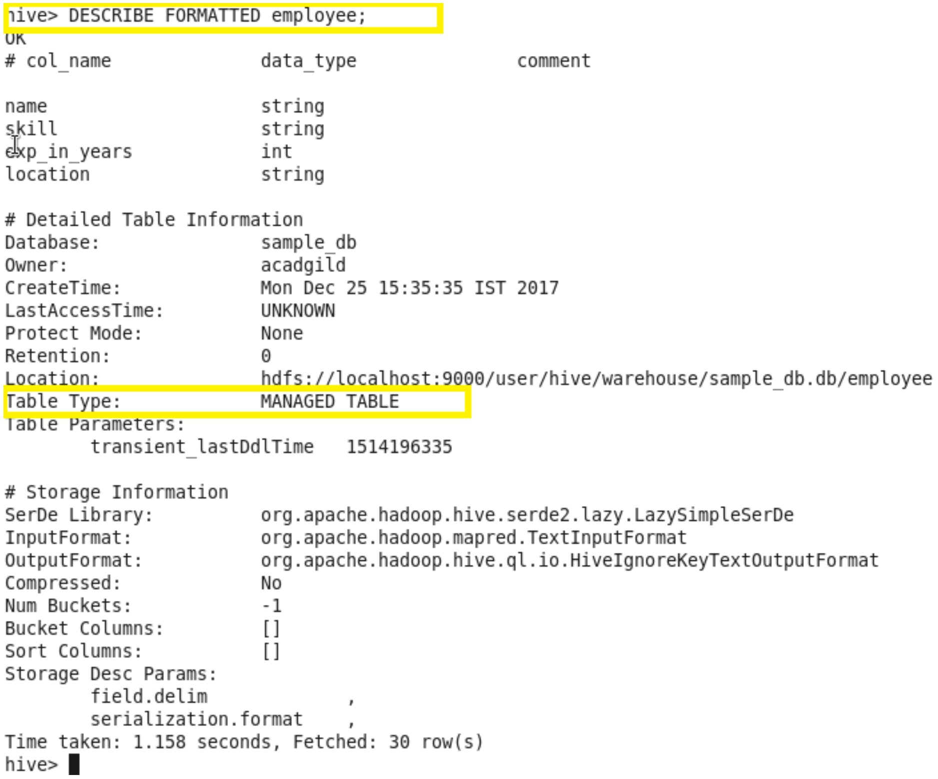
CREATE TABLE <table name>
(column_name_1 data type,
column_name_2 data type,
….
column_name_n data type)
row format delimited
fields terminated by',';

Example:



```
hive> CREATE TABLE IF NOT EXISTS employee
    > (
    >  name STRING,
    >  skill STRING,
    >  exp_in_years INT,
    >  location STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 1.441 seconds
hive> █
```

We have successfully created the table and to check the details of the table type the below command:



```
hive> DESCRIBE FORMATTED employee;
OK
# col_name              data_type               comment

name                    string
skill                   string
exp_in_years            int
location                string

# Detailed Table Information
Database:               sample_db
Owner:                  acadgild
CreateTime:             Mon Dec 25 15:35:35 IST 2017
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://localhost:9000/user/hive/warehouse/sample_db.db/employee
Table Type:             MANAGED TABLE
Table Parameters:
        transient_lastDdlTime   1514196335

# 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: 1.158 seconds, Fetched: 30 row(s)
hive>
```

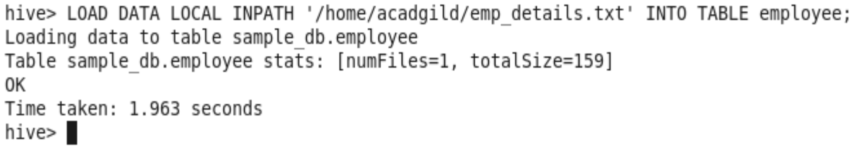
In the above image we can see MANAGED_TABLE as the entry for the option Table type which means that we have created a Managed table.

Let’s try to load a sample dataset into the table by using the below command:

Syntax:

LOAD DATA LOCAL INPATH ‘<path of the file>’ INTO TABLE <table name>;

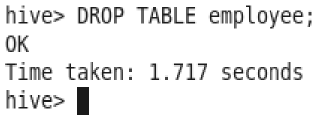
Example:



```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/emp_details.txt' INTO TABLE employee;
Loading data to table sample_db.employee
Table sample_db.employee stats: [numFiles=1, totalSize=159]
OK
Time taken: 1.963 seconds
hive>
```

Now let us drop the above created table by using DROP command.

Syntax: DROP TABLE <table name>;



```
hive> DROP TABLE employee;
OK
Time taken: 1.717 seconds
hive>
```

External tables: These kind of tables are used when data has to be used outside Hive. External table is used when we want to delete a table’s metadata keeping its data as it is.

Here is the **syntax** to create an external table:

CREATE EXTRERNAL TABLE <table name>

(column_name_1 data type,

column_name_2 data type,

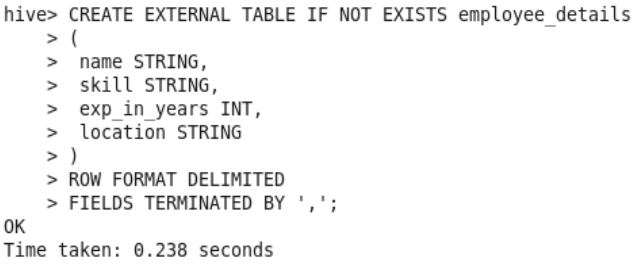
….

column_name_n data type)

row format delimited

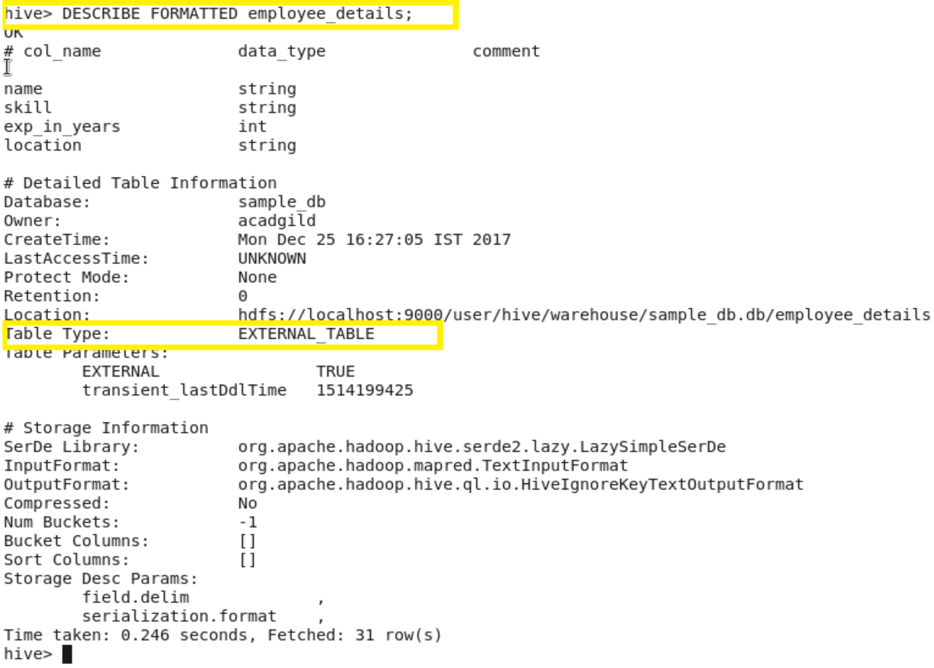
fields terminated by',';

Example:



```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS employee_details
    > (
    >  name STRING,
    >  skill STRING,
    >  exp_in_years INT,
    >  location STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.238 seconds
```

Let us check the details regarding the table using the below command:



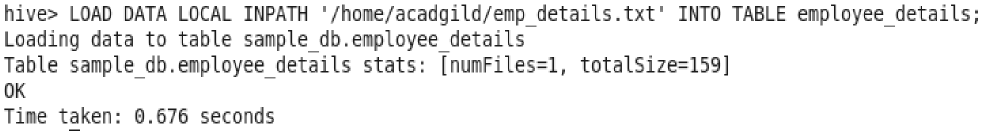
```
hive> DESCRIBE FORMATTED employee_details;
OK
# col_name              data_type               comment

name                    string
skill                   string
exp_in_years            int
location                string

# Detailed Table Information
Database:               sample_db
Owner:                  acadgild
CreateTime:             Mon Dec 25 16:27:05 IST 2017
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://localhost:9000/user/hive/warehouse/sample_db.db/employee_details
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1514199425

# 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: 0.246 seconds, Fetched: 31 row(s)
hive> █
```

Now let us load some data into the table using the below command:

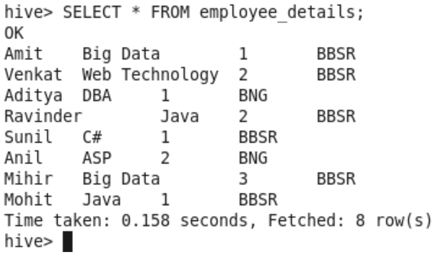
Syntax:

LOAD DATA LOCAL INPATH '<path of the file>' INTO TABLE <table name>;

Example:



```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/emp_details.txt' INTO TABLE employee_details;
Loading data to table sample_db.employee_details
Table sample_db.employee_details stats: [numFiles=1, totalSize=159]
OK
Time taken: 0.676 seconds
```

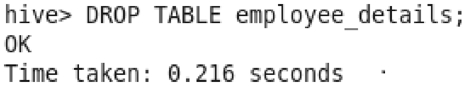


```
hive> SELECT * FROM employee_details;
OK
Amit    Big Data        1       BBSR
Venkat  Web Technology  2       BBSR
Aditya  DBA     1       BNG
Ravinder        Java    2       BBSR
Sunil   C#      1       BBSR
Anil    ASP     2       BNG
Mihir   Big Data        3       BBSR
Mohit   Java    1       BBSR
Time taken: 0.158 seconds, Fetched: 8 row(s)
hive> █
```

Now let us drop the above created table by using DROP command.

Syntax: DROP TABLE <table name>;

Example:



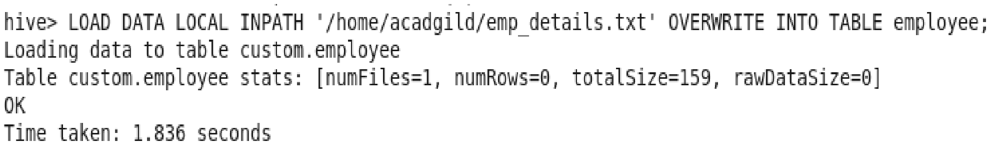
```
hive> DROP TABLE employee_details;
OK
Time taken: 0.216 seconds
```

- **<u>HiveQL Data Manipulations:</u>**

Data manipulation language of Hive is mainly used to insert data into tables and to extract data from tables into a file system.

Loading Data into Managed Tables:

Hive doesn't support row level insert operation. Instead, it offers bulk load of data from external files or other tables in Hive. We saw how to load data into a managed table in the previous section with the use of OVERWRITE keyword:



```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/emp_details.txt' OVERWRITE INTO TABLE employee;
Loading data to table custom.employee
Table custom.employee stats: [numFiles=1, numRows=0, totalSize=159, rawDataSize=0]
OK
Time taken: 1.836 seconds
```

The LOCAL keyword indicates the Hive system that data file or directory is present in the local file system. The data is **copied** into the final location. If we don't use LOCAL keyword, the path refers to distributed file system (HDFS) and in that case, data will be **moved** from given path to final location, say a database table. The reason for this sort of notion is Hive assumes that we don't want to keep copies of data in the distributed file system once it's imported to Hive tables.

Data Insertion into Tables:

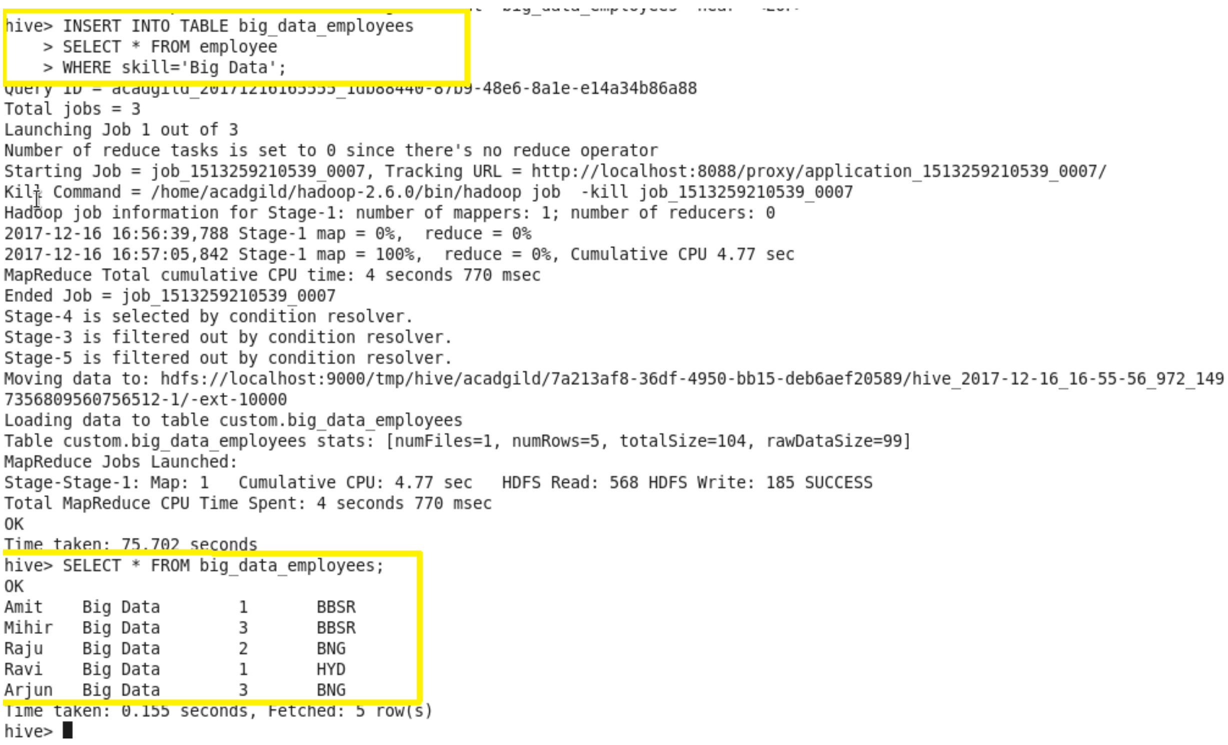
The INSERT statement is used to load data into a table via a query. For example, I am using the *employee* table from previous topic (Data definitions part) as well as a new table called *big_data_employees* which contains details of employees who have skills pertaining to 'Big Data'. Let's see the query to achieve the latter part.

Example query:

INSERT INTO TABLE big_data_employees

SELECT * FROM employee

WHERE skill = 'Big Data';



```
hive> INSERT INTO TABLE big_data_employees
    > SELECT * FROM employee
    > WHERE skill='Big Data';
Query ID = acadgild_20171216165555_1db88440-87b9-48e6-8a1e-e14a34b86a88
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_1513259210539_0007, Tracking URL = http://localhost:8088/proxy/application_1513259210539_0007/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513259210539_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-12-16 16:56:39,788 Stage-1 map = 0%,  reduce = 0%
2017-12-16 16:57:05,842 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.77 sec
MapReduce Total cumulative CPU time: 4 seconds 770 msec
Ended Job = job_1513259210539_0007
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive/acadgild/7a213af8-36df-4950-bb15-deb6aef20589/hive_2017-12-16_16-55-56_972_149
7356809560756512-1/-ext-10000
Loading data to table custom.big_data_employees
Table custom.big_data_employees stats: [numFiles=1, numRows=5, totalSize=104, rawDataSize=99]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.77 sec   HDFS Read: 568 HDFS Write: 185 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 770 msec
OK
Time taken: 75.702 seconds
hive> SELECT * FROM big_data_employees;
OK
Amit    Big Data        1       BBSR
Mihir   Big Data        3       BBSR
Raju    Big Data        2       BNG
Ravi    Big Data        1       HYD
Arjun   Big Data        3       BNG
Time taken: 0.155 seconds, Fetched: 5 row(s)
hive> 
```

The keyword 'INTO' in the above query indicates that, Hive appends the data from one table to the other. If the keyword 'OVERWRITE' is used in place of INTO, then Hive replaces existing records with new ones. This feature is available only in Hive v0.8.0 or above.

Exporting Data:

We can export data from Hive tables to external file system by tweaking the previous INSERT query as shown below:

Example query:

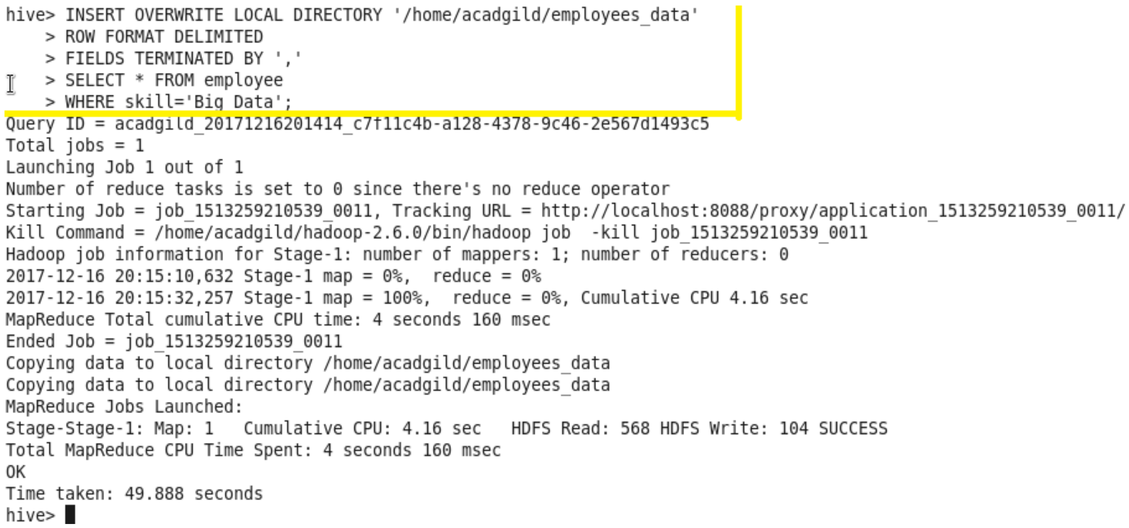
INSERT INTO LOCAL DIRECTORY '/home/acadgild/employees_data'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

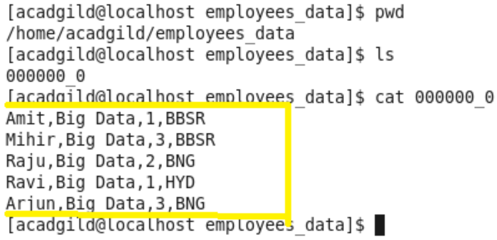
SELECT * FROM employee

WHERE skill = 'Big Data';



```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/employees_data'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > SELECT * FROM employee
    > WHERE skill='Big Data';
Query ID = acadgild_20171216201414_c7f11c4b-a128-4378-9c46-2e567d1493c5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1513259210539_0011, Tracking URL = http://localhost:8088/proxy/application_1513259210539_0011/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513259210539_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-12-16 20:15:10,632 Stage-1 map = 0%,  reduce = 0%
2017-12-16 20:15:32,257 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.16 sec
MapReduce Total cumulative CPU time: 4 seconds 160 msec
Ended Job = job_1513259210539_0011
Copying data to local directory /home/acadgild/employees_data
Copying data to local directory /home/acadgild/employees_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.16 sec   HDFS Read: 568 HDFS Write: 104 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 160 msec
OK
Time taken: 49.888 seconds
hive> 
```

Let us navigate to the path given in the query above and verify the file contents:



```
[acadgild@localhost employees_data]$ pwd
/home/acadgild/employees_data
[acadgild@localhost employees_data]$ ls
000000_0
[acadgild@localhost employees_data]$ cat 000000_0
Amit,Big Data,1,BBSR
Mihir,Big Data,3,BBSR
Raju,Big Data,2,BNG
Ravi,Big Data,1,HYD
Arjun,Big Data,3,BNG
[acadgild@localhost employees_data]$ 
```