Experiment No 3

Aim: Write an application using HiveQL for flight information system which will include

a) Creating, Dropping, and altering Database tables.
b) Creating an external Hive table.
c) Load table with data, insert new values and field in the table, Join tables with Hive
d) Create index on Flight Information Table
e) Find the average departure delay per day in 2008.

Theory:

The Hive concept of a database is essentially just a catalog or namespace of tables. However, they are very useful for larger clusters with multiple teams and users, as a way of avoiding table name collisions. It's also common to use databases to organize production tables into logical groups.

a) Creating, Dropping, and altering Database tables.

The simplest syntax for creating a database is shown in the following example:

```
hive> CREATE DATABASE demo;
```

Hive will throw an error if financials already exists. You can suppress these warnings with this variation:


```
hive> CREATE DATABASE IF NOT EXISTS demo;
```

While normally you might like to be warned if a database of the same name already exists, the IF NOT EXISTS clause is useful for scripts that should create a database on-the-fly, if necessary, before proceeding.

```
hive> drop database demo;
```

b) Creating an external Hive table.

Create Table

In Hive, we can create a table by using the conventions similar to the SQL. It supports a wide range of flexibility where the data files for tables are stored. It provides two types of table: -

- Internal table

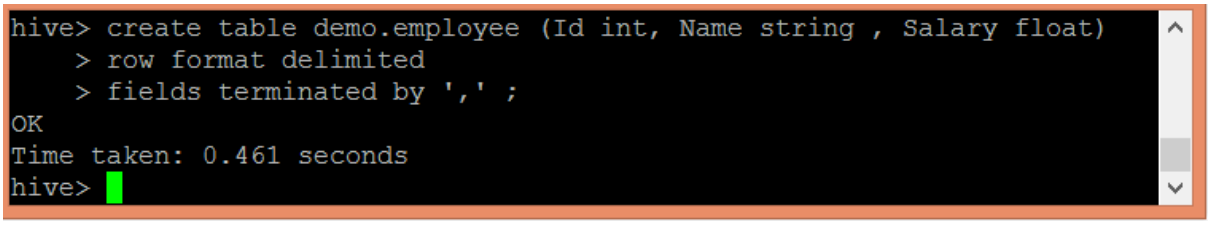
- External table

Internal Table

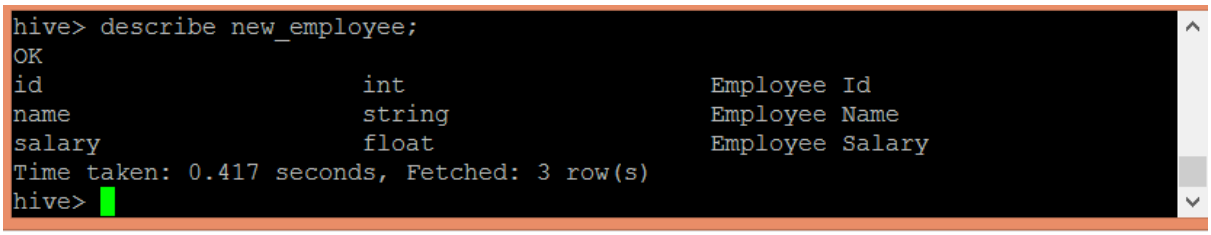
The internal tables are also called managed tables as the lifecycle of their data is controlled by the Hive. By default, these tables are stored in a subdirectory under the directory defined by hive.metastore.warehouse.dir (i.e. /user/hive/warehouse). The internal tables are not flexible enough to share with other tools like Pig. If we try to drop the internal table, Hive deletes both table schema and data.

Let's create an internal table by using the following command:-

```
hive> create table demo.employee (Id int, Name string , Sa
lary float)
     >row format delimited
     >fields terminated by ',' ;
```




```
hive> describe new_employee;
```



External Table

The external table allows us to create and access a table and a data externally. The external keyword is used to specify the external table, whereas the location keyword is used to determine the location of loaded data.

As the table is external, the data is not present in the Hive directory. Therefore, if we try to drop the table, the metadata of the table will be deleted, but the data still exists.

To create an external table, follow the below steps: -

- Let's create a directory on HDFS by using the following command: -

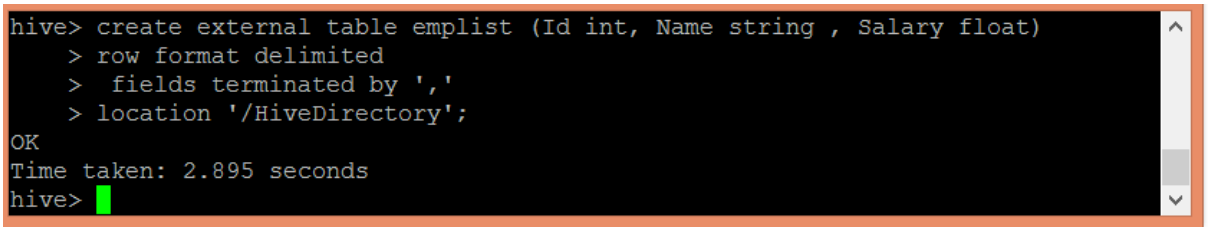
```
hdfs dfs -mkdir /HiveDirectory
```

- Now, store the file on the created directory.

```
hdfs dfs -put hive/emp_details /HiveDirectory
```

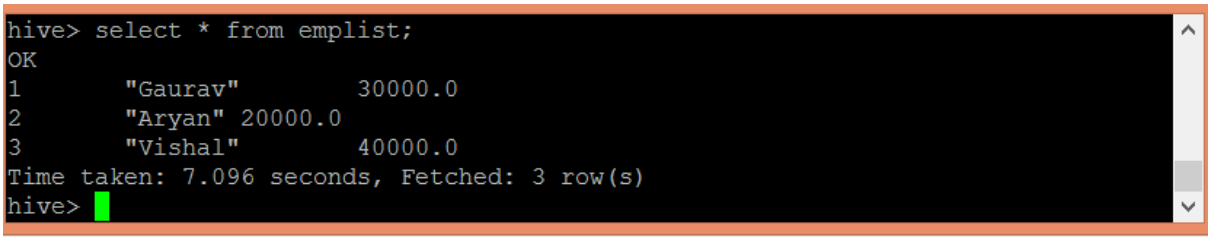
- Let's create an external table using the following command: -

```
hive> create external table emplist (Id int, Name string , Salary float)
row format delimited
 fields terminated by ','
location '/HiveDirectory';
```



- Now, we can use the following command to retrieve the data: -

```
select * from emplist;
```



c) Load table with data, insert new values and field in the table, Join tables with Hive

Hive - Alter Table

In Hive, we can perform modifications in the existing table like changing the table name, column name, comments, and table properties. It provides SQL like commands to alter the table.

Adding column

In Hive, we can add one or more columns in an existing table by using the following query

```
Alter table table_name add columns(column_name datatype);
```

Let's see the schema of the table.

```
hive> describe employee_data;
```

Now, add a new column to the table by using the following command: -

```
Hive>Alter table employee_data add columns (age int);
```

```
hive> Alter table employee_data add columns ( age int);
OK
Time taken: 1.025 seconds
hive> describe employee_data;
OK
id                      int
name                    string
salary                  float
age                     int
Time taken: 0.308 seconds, Fetched: 4 row(s)
hive>
```

d) Create index on Flight Information Table

The Airline On-Time Performance Data, "contains on-time arrival data for non-stop domestic flights by major air carriers, and provides such additional items as departure and arrival delays, origin and destination airports, flight numbers, scheduled and actual departure and arrival times, cancelled or diverted flights, taxi-out and taxi-in times, air time, and non-stop distance."

The data is provided in the form of .csv files from year 1987 to 2008. It's a huge dataset(2 decades old) which contains around 120 million rows of flight details and sums up to about 12GB when uncompressed.

This dataset can be used to work on cool travel ideas like:

- When is the best time of day/day of week/time of year to fly to minimise delays?
- Do older planes suffer more delays?
- How does the number of people flying between different locations change over time?
- How well does weather predict plane delays?
- Can you detect cascading failures as delays in one airport create delays in others? Are there critical links in the system?

LOAD DATA

First we have to place data into the data_types_table with LOAD DATA command. The syntax for the LOAD DATA command is given below.

```
"LOAD DATA [LOCAL] INPATH 'path to file' [OVERWRITE] INTO
          TABLE 'table name' [PARTITION partition column1
          = value1, partition column2 = value2,...]
```

In the above syntax optional LOCAL keyword tells Hive to copy data from the input file on the local file system into the Hive data warehouse directory. Without the LOCAL keyword, the data is simply moved (not copied) into the warehouse directory.

Creating an index is common practice with relational databases when we want to speed access to a column or set of columns in your database. Without an index, the database system has to read all rows in the table to find the data we have selected. Indexes become even more

essential when the tables grow extremely large. Hive supports index creation on tables. In Listing 18, we list the steps necessary to index the FlightInfo2008 table.

```
(A) CREATE INDEX f08_index ON TABLE flightinfo2008
          (Origin) AS 'COMPACT' WITH DEFERRED REBUILD;
(B) ALTER INDEX f08_index ON flightinfo2008 REBUILD;
(C) hive (flightdata)> SHOW INDEXES ON FlightInfo2008;
OK
f08index                flightinfo2008          origin
          flightdata__flightinfo2008_f08index__ compact
Time taken: 0.079 seconds, Fetched: 1 row(s)
```

e) Find the average departure delay per day in 2008.

The concept of windowing, introduced in the SQL:2003 standard, allows the SQL programmer to create a frame from the data against which aggregate and other window functions can operate. HiveQL now supports windowing per the SQL standard. One question we had when we first discovered this data set was, "What exactly is the average flight delay per day?" So we created a query in below Listing that produces the average departure delay per day in 2008.

```
(A) hive (flightdata)> CREATE VIEW avgdepdelay AS
                    > SELECT DayOfWeek, AVG(DepDelay) FROM
          FlightInfo2008 GROUP BY DayOfWeek;
OK
Time taken: 0.121 seconds
(B) hive (flightdata)> SELECT * FROM avgdepdelay;
...
OK
1       10.269990244459473
2       8.97689712068735
3       8.289761053658728
4       9.772897177836702
5       12.158036387869656
6       8.645680904903614
7       11.568973392595312
Time taken: 18.6 seconds, Fetched: 7 row(s)
```