Advanced HiveSQL

Question 1: create table called movies_whole with 3 columns (movieid, movie_name, genre)

HiveSQL : create table Movielens.movies_whole1(movieid string , movie_name varchar (50), genre string)row format delimited fields terminated by ',' collection items terminated by '$';

Question 2. load action_comedy_thriller file into table

HiveSQL : Load data inpath '/user/root/action_comedy_thriller' overwrite into table Movielens.movies_whole2;

Question 3. create a table called movies_part with 2 columns (movieid, movie_name) that is partitioned on genre

HiveSQL : CREATE TABLE Movielens.movies_part1(movieid string, movie_name string) PARTITIONED BY (genre string);

Question 4 : load each file (action, comedy, and thriller) into a partitions ("Action", "Comedy", and "Thriller")

HiveSQL :

- LOAD DATA LOCAL INPATH 'action' INTO TABLE Movielens.movies_part PARTITION (genre = 'Action');
- LOAD DATA LOCAL INPATH 'comedy' INTO TABLE Movielens.movies_part PARTITION (genre = 'Comedy');
- LOAD DATA LOCAL INPATH 'thriller' INTO TABLE Movielens.movies_part PARTITION (genre = 'Thriller');

Question 5 : describe the structure of the table and list the partitions (hint: describe and show partitions command)

HiveSQL : describe movies_part;

Advanced HiveSQL

```
hive> describe movies_part;
OK
movieid                 string
movie_name              string
genre                   string

# Partition Information
# col_name              data_type               comment

genre                   string
Time taken: 0.512 seconds, Fetched: 8 row(s)
```

Partition command : show partitions movies_part;

```
hive> show  partitions  movies_part;
OK
genre=Action
genre=Comedy
genre=Thriller
Time taken: 0.742 seconds, Fetched: 3 row(s)
hive>
```

Question 6: navigate to the location of movie_part on HDFS. How does the partitioned table look on HDFS? Write 1 line on what you think is happening when partitioned tables are created.

HiveSQL : dfs -ls /apps/hive/warehouse/movies_part;

```
Found 3 items
drwxrwxrwx   - root hadoop          0 2018-10-11 01:04 /apps/hive/warehouse/movies_part
/genre=Action
drwxrwxrwx   - root hadoop          0 2018-10-11 01:05 /apps/hive/warehouse/movies_part
/genre=Comedy
drwxrwxrwx   - root hadoop          0 2018-10-11 01:07 /apps/hive/warehouse/movies_part
/genre=Thriller
```

Advanced HiveSQL

I think partition will determine how the data will be stored in the table. For example, if a table has two columns, id, name and age; and is partitioned by age, all the rows having same age will be stored together.

HiveSQL 7.a : Table movie_whole2 : Select * from movies_whole1 limit 20;
Execution Time : Time taken: 0.189 seconds, Fetched: 20 row(s)

Table movie_part : Select * from movies_part limit 20;
Execution Time : Time taken: 0.31 seconds, Fetched: 20 row(s)

HiveSQl 7.b : Table movie_whole2 : Select count(*) from movies_whole2 where genre='Action';
Execution Time : Time taken: 16.216 seconds, Fetched: 1 row(s)

Table movie_part : Select count(*) from movies_part where genre='Action';
Execution Time : Time taken: 7.558 seconds, Fetched: 1 row(s)

HiveSQl 7.c: Table movie_whole2 : Select count(*) from movies_whole2;
Execution Time : Time taken: 7.575 seconds, Fetched: 1 row(s

Table movie_part : Select count(*) from movies_part;
Execution Time : Time taken: 6.676 seconds, Fetched: 1 row(s)

HiveSQl 7.d: Table movie_whole2 : Select t.year, count(*) as count from (Select regexp_extract(movie_name, '([1-2][0-9][0-9][0-9])',1) as year from movies_whole2) t group by year order by count desc limit 5;

Execution Time : Time taken: 9.218 seconds, Fetched: 5 row(s)

Table movie_part :Select t.year, count(*) as count from (Select regexp_extract(movie_name, '([1-2][0-9][0-9][0-9])',1) as year from movies_part) t group by year order by count desc limit 5;

Execution Time : Time taken: 9.001 seconds, Fetched: 1 row(s)

Advanced HiveSQL

HiveSQl 7.e: Table movie_whole2 : Select t.year, count(*) as count from (Select regexp_extract(movie_name, '([1-2][0-9][0-9][0-9])',1) as year from movies_whole1 where genre='Thriller') t group by year order by count desc limit 5;

Execution Time : Time taken: 7.082 seconds

Table movie_part : Select t.year, count(*) as count from (Select regexp_extract(movie_name, '([1-2][0-9][0-9][0-9])',1) as year from movies_part where genre='Thriller') t group by year order by count desc limit 5;

Execution Time : Time taken: 6.907 seconds, Fetched: 1 row(s)

Answer 7.1: I think partitioning table queries run faster. After doing partitioning a table or index may improve query performance, based on the types of queries.

Answer 7.2: My partitioning table queries are running faster in this circumstance. It is not primarily for query performance. I understand from lecture and other resources that partitioning is mostly for improved maintenance, fast loads, fast deletes and the ability to spread a table across multiple file groups.

Question 8: With some help from the "select" statement in 7(e) -> create a table called movie_year_temp with following columns (movieid, movie_title, movie_year).

HiveSQL : create table movie_year_temp as Select movieid, movie_name, cast(regexp_extract(movie_name, '([1-2][0-9][0-9][0-9])',1) as int) as movie_year from movies_whole2;

Output : Table default.movie_year_temp stats: [numFiles=1, numRows=879, totalSize=35907, rawDataSize=35028]
OK
Time taken: 16.704 seconds

Question 9 : Create a table called year_buckets with the same column definitions as movie_year_temp, but with 8 buckets, clustered on movie_year

HiveSQL : CREATE TABLE year_bucket1(movie_id string,movie_name varchar(50), movie_year int)CLUSTERED BY (movie_year) INTO 8 BUCKETS;

Output : OK
Time taken: 0.761 seconds

Question 10: use insert overwrite table to load the rows in movie_year_temp into year_buckets.

Advanced HiveSQL

HiveSQL : INSERT INTO TABLE year_bucket1 SELECT movieid ,movie_name , movie_year FROM movie_year_temp;

Question 11: Navigate to the location of year_buckets on HDFS. How does the partitioned table look on HDFS?

HiveSQL : dfs -ls /apps/hive/warehouse/Movielens.db/year_buckets1;

```
Found 1 items
-rwxrwxrwx   1 root hadoop      35907 2018-10-11 20:21 /apps/hive/warehouse/year_bucket
1/000000_0
```

Question 12: Using the table movie_year_temp apply the histogram function (with 5 buckets) on movie_year to get get the distribution of year values in the table.

HiveSQL: select explode(histogram_numeric(movie_year, 5)) as hist_year from movie_year_temp;

```
OK
{"x":1938.375,"y":8.0}
{"x":1951.8000000000002,"y":5.0}
{"x":1960.8333333333335,"y":6.0}
{"x":1975.5454545454545,"y":11.0}
{"x":1994.7569444444443,"y":144.0}
Time taken: 16.064 seconds, Fetched: 5 row(s)
hive>
```