**MovieLens Project**

Q1 The dataset provided - MovieLens data sets are collected by the GroupLens Research Project at the University of Minnesota. It represents users' reviews of movies. This data set consists of: \* 100,000 ratings (1-5) from 943 users on 1682 movies. \* Each user has rated at least 20 movies. \* Simple demographic info for the users (age, gender, occupation, zip) u.data -- The full u data set, 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1.

The data is randomly ordered. This is a tab separated list of user id item id rating timestamp The time stamps are Unix seconds since 1/1/1970 UTC u.user -- Demographic information about the users; this is a pipe (|) separated list of user id | age | gender | occupation | zip code The user ids are the ones used in the u.data data set. Paste the code for each step and the output of the Query Queries to be performed   
  
1. Create an external table u\_data for u.data in HDFS.

2. See the field descriptions of u\_data table

3. Show all the data in the newly created u\_data table

4. Show the numbers of item reviewed by each user in the newly created u\_data table

5. Show the numbers of users reviewed each item in the newly created u\_data table

6. Create an external table u\_user for u.user in HDFS .

7. See the field descriptions of u\_user table

8. Show all the data in the newly created user table

9. Count the number of data in the u\_user table

10. Count the number of user in the u\_user table genderwise   
11. Join u\_data table and u\_user tables based on userid - Perform a reduce side join and map side join for the same and compare the time taken in both cases.

12. Create a partitioned table u\_user\_partitioned, partitioned by occupation column

13. Find out the total number of male and total number of female only for the most common occupation – you can hard code the occupation/ use subqueries. - Perform the query on both un-partitioned table and partitioned table. - Compare and report the performance differences.

use hivedb;

set hive.cli.print.current.db;

set hive.cli.print.current.db=true;

1. create a u\_data table

create EXTERNAL TABLE IF NOT EXISTS u\_data ( userId INT, movieId INT, rating INT, time STRING )

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

2. See the field descriptions of u\_data table

describe u\_data;

3. load data u.data from location /home/data/cts into u\_data table from a local text file

LOAD DATA INPATH 'datasets/u.data' OVERWRITE INTO table u\_data;

4. show all the data in the newly created u\_data table

select \* from u\_data;

5. show the numbers of item reviewed by each user in the newly created u\_data table

select movieid, count(userid) AS no from u\_data group by movieid order by no;

6. show the numbers of users reviewed each item in the newly created u\_data table

select userid, count(movieid) AS no from u\_data group by userid order by no;

7. create a u\_user table

create External TABLE u\_user ( userId INT, age INT, gender String, occupation STRING, zip INT )

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

8. see the field descriptions of u\_user table

describe u\_user;

9. load data u.user from /home/data/cts location into u\_user table from a local text file

LOAD DATA INPATH 'datasets/u.user' OVERWRITE INTO table u\_user;

10. show all the data in the newly created user table

select \* from u\_user;

11. count the number of data in the u\_user table

select count(\*) from u\_user;

12. count the number of user in the u\_user table genderwise

select gender,count(\*) from u\_user group by gender;

13. join u\_data table and u\_user tables based on userid and show the top 10 results

select \* from u\_user usr JOIN u\_data mov ON usr.userid=mov.userid;

14. Create a partitioned table u\_user\_partitioned, partitioned by occupation column

create External TABLE u\_user\_partitioned (userId INT,age INT,gender String,zip INT)

PARTITIONED BY (occupation STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

STORED AS TEXTFILE;

set hive.exec.dynamic.partition.mode=nonstrict;

INSERT OVERWRITE table u\_user\_partitioned partition(occupation)

SELECT userId, age, gender, zip, occupation

FROM u\_user;

15. Find out the total number of male and total number of female only for the most common occupation – you can hard code the occupation/ use subqueries.

- Perform the query on both un-partitioned table and partitioned table. - Compare and report the performance differences.

select \* from u\_user\_partitioned;

select occupation,count(\*) from u\_user\_partitioned group by occupation;

administrator 79

artist 28

doctor 7

educator 95

engineer 67

entertainment 18

executive 32

healthcare 16

homemaker 7

lawyer 12

librarian 51

marketing 26

none 9

other 105

programmer 66

retired 14

salesman 12

scientist 31

student 196

technician 27

writer 45

Time taken: 23.811 seconds, Fetched: 21 row(s)

select gender,occupation,count(\*) from u\_user where

occupation = 'student'

group by gender,occupation;

F student 60

M student 136

Time taken: 22.449 seconds, Fetched: 2 row(s)

select gender,occupation,count(\*) from u\_user\_partitioned where

occupation = 'student'

group by gender,occupation;

F student 60

M student 136

Time taken: 22.215 seconds, Fetched: 2 row(s)

hive (hivedb)>

**Performance difference = .234 seconds**