```
-----------------------------------------------------------------
-----------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------
-- Programming Hive  - Additional Hive Exercise (Optional)

-- 1. In this optional lab exercise, we will work with the
MovieLens dataset
--    The movielens dataset is a collection of movie ratings data
and has been widely used in the industry and
--    academia for experimenting with recommendation algorithms
and we see many publications using this dataset
--    to benchmark the performance of their algorithms
-- 2. For access to full-sized movielens data, go to
http://grouplens.org/datasets/movielens/
-----------------------------------------------------------------
-----------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------

-----------------------------------------------
-- Loading User Ratings Data into Hive - u.data
-----------------------------------------------

-- 1. Upload movielens.tgz file to linux sandbox /home/lab

-- 2. Extract the data from the MovieLens dataset

$ cd /home/lab
$ tar -zxvf movielens.tgz
$ ll

-- 3. Examine the files

$ cd ml-data
$ more u.data

-----------------------------
-- Table description "u.data"
--
-- field_1     userid
-- field_2     movieid
-- field_3     rating
-- field_4     unixtime
-----------------------------

-- 4. Create a database called ml and table called user_ratings
(tab-delimited)
hive> CREATE DATABASE ml;
hive> CREATE TABLE ml.userratings
     (userid INT, movieid INT, rating INT,
      unixtime BIGINT) ROW FORMAT DELIMITED
```

```
      FIELDS TERMINATED BY '\t'
      STORED AS TEXTFILE;

-- 5.  Move the u.data file into HDFS
$ hadoop fs -put /home/lab/ml-data/u.data /user/lab/u.data

-- 6. Load the u.data into hive table
hive> LOAD DATA INPATH '/user/lab/u.data'
      INTO TABLE ml.userratings;

-- Verify that data was loaded
hive> SELECT * FROM ml.userratings LIMIT 10;


---------------------------------
-- loading movies data into hive
---------------------------------

-- 1. Move the movies data u.item into hadoop

$ hadoop fs -put /home/lab/ml-data/u.item /user/lab/u.item


-- 2. Exam the file
$ hadoop fs -cat /user/lab/u.item | head -n 5

-- 3. Create a table called movies
-- Read the README file for u.item column description

hive> CREATE TABLE ml.movies
         (movieid INT,
          movie_title STRING,
          release_date STRING,
          v_release_date STRING,
          imdb_url STRING,
          cat_unknown INT,
          cat_action INT,
          cat_adventure INT,
          cat_animation INT,
          cat_children INT,
          cat_comedy INT,
          cat_crime INT,
          cat_documentary INT,
          cat_drama INT,
          cat_fantasy INT,
          cat_fill_noir INT,
          cat_horror INT,
          cat_musical INT,
          cat_mystery INT,
          cat_romance INT,
          cat_scifi INT,
          cat_thriller INT,
```

```
          cat_war INT,
          cat_western INT)
      ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE;

-- 4. Load the u.item into hive table ml.Movies
hive> LOAD DATA INPATH '/user/lab/u.item'
      INTO TABLE ml.Movies;

3. verify
$ hive
hive> SELECT * from ml.Movies limit 20;

4. examine the data in hdfs
$ hadoop fs â€"ls /apps/hive/warehouse
$ hadoop fs â€"ls /apps/hive/warehouse/ml.db/userratings
$ hadoop fs â€"ls /apps/hive/warehouse/ml.db/movies


--
------------------------
-- Simple analysis
------------------------

-- 1. how many records in both tables?
SELECT count(*) FROM ml.movies;
SELECT count(*) FROM ml.userratings;

-- 2. find the name of all movies released in 1990
SELECT movie_title FROM ml.movies WHERE release_date LIKE '%1990'
limit 20;

-- 3. list the movieid of the 10 most rated films in user_ratings
table
SELECT count(1) AS ratings, movieid
FROM ml.userratings
GROUP BY movieid
ORDER BY ratings DESC limit 10;

-- 4. use a join to list the titles of the movies you found in
step 3
SELECT count(*) AS ratings, movie_title
 FROM ml.movies JOIN ml.userratings
 ON (movies.movieid = userratings.movieid)
 GROUP BY movie_title ORDER BY ratings DESC LIMIT 10;

-- 5. do any movies have no ratings? (hint: outer join and IS
NULL)
SELECT  movie_title
FROM ml.movies LEFT OUTER JOIN ml.userratings
ON (movies.movieid = userratings.movieid)
```

```
WHERE userratings.movieid IS NULL;

-- 6. what is the highest rated sci_fi mvoie
SELECT movie_title, avg(rating) AS avgrating
FROM ml.movies JOIN ml.userratings
ON (movies.movieid = userratings.movieid)
WHERE cat_scifi = 1
GROUP BY movie_title
ORDER BY avgrating DESC LIMIT 1;

-- 7. what is the highest rated sci_fi movie that has at least 10
user ratings

SELECT movie_title, avgrating FROM
(SELECT movie_title, avg(rating) AS avgrating, count(1)
   AS numratings
   FROM movies JOIN userratings
   ON (movies.movieid = userratings.movieid)
WHERE cat_scifi = 1
GROUP BY movie_title) t
WHERE numratings > 10
ORDER BY avgrating DESC LIMIT 1;


-----------------------------------------
-- Partitioning and bucketing data in hive
-----------------------------------------

-- 1. load action.txt, comedy.txt, thriller.txt into hdfs
-- You need to download these files from course shell and then
upload to the sandbox first

$ less action.txt
$ less comedy.txt
$ less thriller.txt

$ hadoop fs -put /home/lab/action.txt /user/lab/action
$ hadoop fs -put /home/lab/comedy.txt /user/lab/comedy
$ hadoop fs -put /home/lab/thriller.txt /user/lab/thriller



-- 2. create a table called movies_partition with 4 columns
(movieid, movie_title, release_date, imdb_url) that is
partitioned on genre


hive> CREATE TABLE ml.movies_part
      (movieid int,
      movie_name string,
      release_date string,
      imdb_url string)
```

```
PARTITIONED BY (genre string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';


-- 3. load each file into a partition

hive> LOAD DATA INPATH '/user/lab/action'
      INTO TABLE ml.movies_part
      PARTITION(genre='action');
hive> LOAD DATA INPATH '/user/lab/comedy'
      INTO TABLE ml.movies_part
      PARTITION(genre='comedy');
hive> LOAD DATA INPATH '/user/lab/thriller'
      INTO TABLE ml.movies_part
      PARTITION(genre='thriller');

-- 4. describe the structure of the table and list the partitions
(hint: describe and show partitions command)
hive> DESCRIBE ml.movies_part;
hive> SHOW PARTITIONS ml.movies_part;

-- 5. look at the hive warehouse to see the 3 subdirectories

hive> dfs -ls /apps/hive/warehouse/ml.db/movies_part

-- 6. create a table called rating_buckets with the same column
definitions as user_ratings, but with 8 buckets, clustered on
movieid
hive> CREATE TABLE ml.rating_buckets
         (userid int,
          movieid int,
          rating int,
          unixtime int)
      CLUSTERED BY (movieid) INTO 8 BUCKETS;

-- 7. use insert overwrite table to load the rows in user_ratings
into rating_buckets. dont' forget to set mapred.reduce.tasks to 8

hive> SET mapred.reduce.tasks = 8;
hive> INSERT OVERWRITE TABLE ml.rating_buckets
      SELECT *
      FROM ml.userratings CLUSTER BY movieid;


-- 8. view the 8 fiels that were created.
$ hadoop fs -ls /user/hive/warehouse/rating_buckets

-- 9. count the rows in bucket 3 using tablesample
hive> SELECT count(1) FROM ml.rating_buckets
      TABLESAMPLE (BUCKET 3 OUT OF 8);
```