Engineering Excellence - J1

(Hadoop Cluster Setup - Movielens Data)

MovieLens Data Use Case Development Notes

TABLE OF CONTENT

Create Database & switch to it:

hive> create database movie_lens_data;

hive> use movie_lens_data;

HIVE TABLES required:

USERS, RATINGS, MOVIES

Table 1: USERS - (UserID, Name, Age, Gender, Occupation, Zip Code)

```
create table if not exists users
        (user_id bigint,
         name string,
         age int,
         gender char(1),
         occupation string,
         zip_code string)
comment 'movie lens user table'
row format delimited
fields terminated by ','
stored as textfile;
```

```
hive> desc users;
OK
user_id             bigint
name                string
age                 int
gender              char(1)
occupation          string
zip_code            string
Time taken: 0.364 seconds, Fetched: 6 row(s)
```

Table 2: MOVIES - (MovieId, Title, Genres)

```
create table if not exists movies
        (movie_id bigint,
         title string,
         genres string)
comment 'movie lens: movie table'
row format delimited
fields terminated by ','
stored as textfile;

hive> desc movies;
OK
movie_id            bigint
title               string
genres              string
Time taken: 0.446 seconds, Fetched: 3 row(s)
hive>
```

Table 3: RATINGS - (UserId, MovieId, Rating, Timestamp)

```
create table if not exists ratings
        (user_id bigint,
         movie_id bigint,
         rating float,
         time_stamp string)
comment 'movie lens: ratings table'
row format delimited
fields terminated by ','
stored as textfile;
```

hive> desc ratings;

OK

user_id bigint

movie_id bigint

rating float

time_stamp string

Time taken: 0.336 seconds, Fetched: 4 row(s)

hive>

DATA SET

This dataset describes 5-star rating and free-text tagging activity from [MovieLens] (http://movielens.org), a movie recommendation service. It contains 47644977 ratings for 34308 movies. These data were created by 247853 users between January 09, 1995 and January 29, 2016. This dataset was generated on January 29, 2016.

Ratings Data File Structure (ratings.csv)

All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

userId, movieId, rating, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Below is the sample records from ratings.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data/ratings/ratings.csv

247752,4993,0.5,1287412650

247752,5952,0.5,1287412663

247752,7153,0.5,1287412661

247752,8874,4.0,1287412729

247752,27773,2.5,1287413266

247752,30749,4.0,1287412625

Movies Data File Structure (movies.csv)

Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has the following format:

movieId, title, genres

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following:

* Action
* Adventure
* Animation
* Children's
* Comedy
* Crime
* Documentary
* Drama
* Fantasy
* Film-Noir
* Horror
* Musical
* Mystery

* Romance

* Sci-Fi

* Thriller

* War

* Western

* (no genres listed)

Below is the sample records from movies.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data_bkp/movies/movies.csv

151657,iMurders (2008),Drama|Horror|Mystery|Thriller

151661,Autoerotic (2011),Drama|Romance

151663,"Semen, a Love Sample (2005)",Comedy|Romance

151667,Romance on the Run (1938),(no genres listed)

151669,Genetic Me (2014),(no genres listed)

151671,The Chosen (2015),Thriller

151673,Hustle & Heat (2003),Action|Comedy|Crime|Romance|Thriller

Users Data File Structure (users.csv)

This file contains demographic information about the users; this is a comma separated list with following format:

userId, age, gender, occupation, zip-code

Below is the sample records from movies.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data_bkp/users/users.csv

247732,Test User 247732,27,F,Engineer,900673

247733,Test User 247733,35,F,None,425922

247734,Test User 247734,24,M,HomeMaker,885037

247735,Test User 247735,28,F,HomeMaker,849347

247736,Test User 247736,23,M,Writer,732639

247737,Test User 247737,16,M,Librarian,847508

247738,Test User 247738,16,M,Executive,585023

Add movielens data (.csv) files to HDFS:

hdfs@impetus-i0161:~$ hdfs dfs -mkdir /user/hdfs/movie_lens_data

hdfs@impetus-i0161:~$ hdfs dfs -mkdir /user/hdfs/movie_lens_data/movies

hdfs@impetus-i0161:~$ hdfs dfs -mkdir /user/hdfs/movie_lens_data/ratings

hdfs@impetus-i0161:~$ hdfs dfs -mkdir /user/hdfs/movie_lens_data/users

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/movies.csv /user/hdfs/movie_lens_data/movies

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/ratings.csv /user/hdfs/movie_lens_data/ratings

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/users.csv /user/hdfs/movie_lens_data/users

HDFS file snapshot

hdfs@impetus-i0161:~$ ll /home/hdfs/lens_data/ml-latest/users.csv

-rw-r--r-- 1 hdfs hadoop 22628 Jun 10 20:38 /home/hdfs/lens_data/ml-latest/users.csv

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/movies.csv /user/hdfs/movie_lens_data/movies

hdfs@impetus-i0161:~$ hdfs dfs -ls /user/hdfs/movie_lens_data/movies

Found 1 items

-rw-r--r-- 3 hdfs hdfs 1729811 2016-06-10 21:28 /user/hdfs/movie_lens_data/movies/movies.csv

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/ratings.csv /user/hdfs/movie_lens_data/ratings

hdfs@impetus-i0161:~$ hdfs dfs -put /home/hdfs/lens_data/ml-latest/users.csv /user/hdfs/movie_lens_data/users

hdfs@impetus-i0161:~$ hdfs dfs -ls /user/hdfs/movie_lens_data/ratings

Found 1 items

-rw-r--r--   3 hdfs hdfs 620204630 2016-06-10 21:28 /user/hdfs/movie_lens_data/ratings/ratings.csv

hdfs@impetus-i0161:~$ hdfs dfs -ls /user/hdfs/movie_lens_data/users

Found 1 items

-rw-r--r--   3 hdfs hdfs   22628 2016-06-10 21:28 /user/hdfs/movie_lens_data/users/users.csv

hdfs@impetus-i0161:~$

Load data to HIVE tables from HDFS using PIG

Load Movies Data

```
grunt> movies = LOAD 'hdfs://EETeamJ1/user/hdfs/movie_lens_data/movies/movies.csv' USING PigStorage(',') as (movie_id:long,title:chararray,genres:chararray);

grunt> STORE movies INTO 'movie_lens_data.movies' USING org.apache.hive.hcatalog.pig.HCatStorer();
```

Load Users Data

```
users = LOAD 'hdfs://EETeamJ1//user/hdfs/movie_lens_data/users/users.csv' USING PigStorage(',') as (user_id:long,name:chararray,age:int,gender:chararray,occupation:chararray,zip_code:chararray);

STORE users INTO 'movie_lens_data.users' USING org.apache.hive.hcatalog.pig.HCatStorer();
```

Load Ratings Data

```
ratings = LOAD 'hdfs://EETeamJ1//user/hdfs/movie_lens_data/ratings/ratings.csv' USING PigStorage(',') as (user_id:long,movie_id:long,rating:float,time_stamp:chararray);

STORE ratings INTO 'movie_lens_data.ratings' USING org.apache.hive.hcatalog.pig.HCatStorer();
```

Check Hive data after PIG execution

Movies data verification

hive> select count(*) from movies;

Query ID = hive_20160714163223_5b95aedb-79a9-4e67-ab1d-10bfe138fc1b

Total jobs = 1

Launching Job 1 out of 1

Tez session was closed. Reopening...

Session re-established.

Status: Running (Executing on YARN cluster with App id application_1468446400470_0020)

--

VERTICES STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED

--

Map 1 ..... SUCCEEDED 1 1 0 0 0 0

Reducer 2 ..... SUCCEEDED 1 1 0 0 0 0

--

VERTICES: 02/02 [==========>>] 100% ELAPSED TIME: 5.07 s

--

OK

34208

Time taken: 12.989 seconds, Fetched: 1 row(s)

hive>

Ratings data verification

hive> select count(*) from ratings;

Query ID = hive_20160714163436_5e2fd51a-cc21-432a-a6f4-891b3c37aae0

Total jobs = 1

Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1468446400470_0020)

VERTICES STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED

Map 1 ..... SUCCEEDED 9 9 0 0 0 0

Reducer 2 ..... SUCCEEDED 1 1 0 0 0 0

VERTICES: 02/02 [==========>>] 100% ELAPSED TIME: 15.19 s

OK

22884377

Time taken: 15.704 seconds, Fetched: 1 row(s)

hive>

Users data verification

hive>select count(*) from users;

Query ID = hive_20160714163554_7f63bae6-43ea-4320-aef1-ce5bcc8c9395

Total jobs = 1

Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1468446400470_0020)

VERTICES STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED

Map 1 ..... SUCCEEDED 1 1 0 0 0 0

Reducer 2 ..... SUCCEEDED 1 1 0 0 0 0

VERTICES: 02/02 [==========>>] 100% ELAPSED TIME: 4.65 s

OK

247753

Time taken: 5.225 seconds, Fetched: 1 row(s)

hive>

USE CASE 1: List all the movies and the number of ratings

hive> select title, count(*) from movies right outer join ratings on movies.movie_id=ratings.movie_id group by movies.movie_id, title;

Store ratings data into new table

```
CREATE TABLE mov_rating_count(movie_id bigint, title string, rating_count bigint);
INSERT OVERWRITE TABLE mov_rating_count
 SELECT movie_id, title, count(*)
 FROM movies
 RIGHT OUTER JOIN ratings
 ON movies.movie_id=ratings.movie_id
 GROUP BY movies.movie_id, title;
```

```
hive> INSERT OVERWRITE TABLE mov_rating_count
    > SELECT movie_id, title, count(*)
    > FROM movies
    > RIGHT OUTER JOIN ratings
    > ON movies.movie_id=ratings.movie_id
    > GROUP BY movies.movie_id, title;
Loading data to table movie_lens_data.mov_rating_count
Table movie_lens_data.mov_rating_count stats: [numFiles=2, numRows=33670, totalSize=1134891, rawDataSize=1101221]
OK
Time taken: 31.771 seconds
hive>
```

USE CASE 2: List all the users and the number of ratings they have done for a movie

```
hive> SELECT u.user_id, u.name, COUNT(r.rating)
 FROM users u, ratings r WHERE u.user_id=r.user_id
 GROUP BY u.user_id, u.name;
```

Store users data into new table

```
CREATE TABLE IF NOT EXISTS user_rating_count ( user_id bigint, name String, rating_count int) COMMENT 'Details how many movies a user rated.' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;
```

```
INSERT OVERWRITE TABLE user_rating_count
 SELECT u.user_id, u.name, COUNT(r.rating)
 FROM users u, ratings r WHERE u.user_id=r.user_id
 GROUP BY u.user_id, u.name;
```

```
hive> select user_id, name, rating_count from user_rating_count LIMIT 10;
OK
1       Test User 1     3
2       Test User 2     4
3       Test User 3     4
4       Test User 4     183
5       Test User 5     25
6       Test User 6     18
7       Test User 7     20
8       Test User 8     15
9       Test User 9     16
10      Test User 10    30
```

USE CASE 3: List all the Movie IDs which have been rated (Movie Id with at least one user rating it)

Deepti:

select DISTINCT ratings.movie_id,movies.title from ratings LEFT JOIN movies where ratings.movie_id = movies.movie_id ;

Optimized:

select movie_id, title from mov_rating_count;

hive> select movie_id, title from mov_rating_count LIMIT 10;

OK

2 Jumanji (1995)

3 Grumpier Old Men (1995)

5 Father of the Bride Part II (1995)

6 Heat (1995)

10 GoldenEye (1995)

13 Balto (1995)

14 Nixon (1995)

18 Four Rooms (1995)

19 Ace Ventura: When Nature Calls (1995)

23 Assassins (1995)

USE CASE 4: List all the Users who have rated the movies (Users who have rated atleast one movie)

```
hive> select user_id, name from user_rating_count LIMIT 10;
OK
1       Test User 1
2       Test User 2
3       Test User 3
4       Test User 4
5       Test User 5
6       Test User 6
7       Test User 7
8       Test User 8
9       Test User 9
10      Test User 10
Time taken: 0.134 seconds, Fetched: 10 row(s)
hive>
```

USE CASE 5: List of all the User with the max, min, average ratings they have given against any movie

hive> select user_id, max(rating), min(rating), round(avg(rating),2) from ratings group by user_id LIMIT 10;

Query ID = hive_20160714172915_16fd4bee-4dfa-4e05-8d3d-ba6769d7001d

Total jobs = 1

Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1468446400470_0021)

--

VERTICES STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED

--

Map 1 ..... SUCCEEDED 9 9 0 0 0 0

Reducer 2 ..... SUCCEEDED 2 2 0 0 0 0

--

VERTICES: 02/02 [==========================>>] 100% ELAPSED TIME: 22.99 s

--

OK

1 5.0 2.5 3.5

6 5.0 1.0 3.64

7 5.0 1.5 4.25

9 5.0 1.0 3.44

12 5.0 1.0 4.08

13 5.0 1.0 2.55

14 5.0 1.0 2.94

19 5.0 3.5 4.37

42483 5.0 1.0 3.86

42484 5.0 3.0 3.83

Time taken: 23.487 seconds, Fetched: 10 row(s)

hive>

USE CASE 6: List all the Movies with the max, min, average ratings given by any user

```
SELECT m.movie_id, m.title, MAX(r.rating),
 AVG(r.rating), MIN(r.rating) FROM movies m,
 ratings r WHERE m.movie_id=r.movie_id
 GROUP BY m.movie_id,m.title;
```

Store movies data into new table

```
CREATE TABLE IF NOT EXISTS movie_ratings ( movie_id bigint,
 title String,
 max_rating float,
 avg_rating float,
 min_rating float)
 COMMENT 'Max min avg rating of any movie.'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
 LINES TERMINATED BY '\n' STORED AS TEXTFILE;

INSERT OVERWRITE TABLE movie_ratings
 SELECT m.movie_id, m.title, MAX(r.rating),
 AVG(r.rating), MIN(r.rating) FROM movies m,
 ratings r WHERE m.movie_id=r.movie_id
 GROUP BY m.movie_id,m.title;
```

hive> select movie_id, title, max_rating, avg_rating, min_rating from movie_ratings LIMIT 20;

OK

1 Toy Story (1995) 5.0 3.8948016 0.5

2 Jumanji (1995) 5.0 3.2210855 0.5

3 Grumpier Old Men (1995) 5.0 3.1800942 0.5

4 Waiting to Exhale (1995) 5.0 2.8797274 0.5

5 Father of the Bride Part II (1995) 5.0 3.0808113 0.5

6 Heat (1995) 5.0 3.836536 0.5

7 Sabrina (1995) 5.0 3.3733666 0.5

8 Tom and Huck (1995) 5.0 3.139661 0.5

9 Sudden Death (1995) 5.0 3.015246 0.5

10 GoldenEye (1995) 5.0 3.436888 0.5

11 "American President 5.0 3.6641243 0.5

12 Dracula: Dead and Loving It (1995) 5.0 2.670864 0.5

13 Balto (1995) 5.0 3.2976334 0.5

14 Nixon (1995) 5.0 3.4313333 0.5

15 Cutthroat Island (1995) 5.0 2.7282789 0.5

16 Casino (1995) 5.0 3.7851126 0.5

17 Sense and Sensibility (1995) 5.0 3.9575002 0.5

18 Four Rooms (1995) 5.0 3.4020066 0.5

19 Ace Ventura: When Nature Calls (1995) 5.0 2.6226342 0.5

20 Money Train (1995) 5.0 2.8992693 0.5

Time taken: 0.106 seconds, Fetched: 20 row(s)

hive>