Analysis of MovieLens Dataset using HDFS & Hive

Data Preparation -

Creating a directory named 'movielens-data' inside Hadoop in Hortonworks Sandbox and downloading data inside local file system –

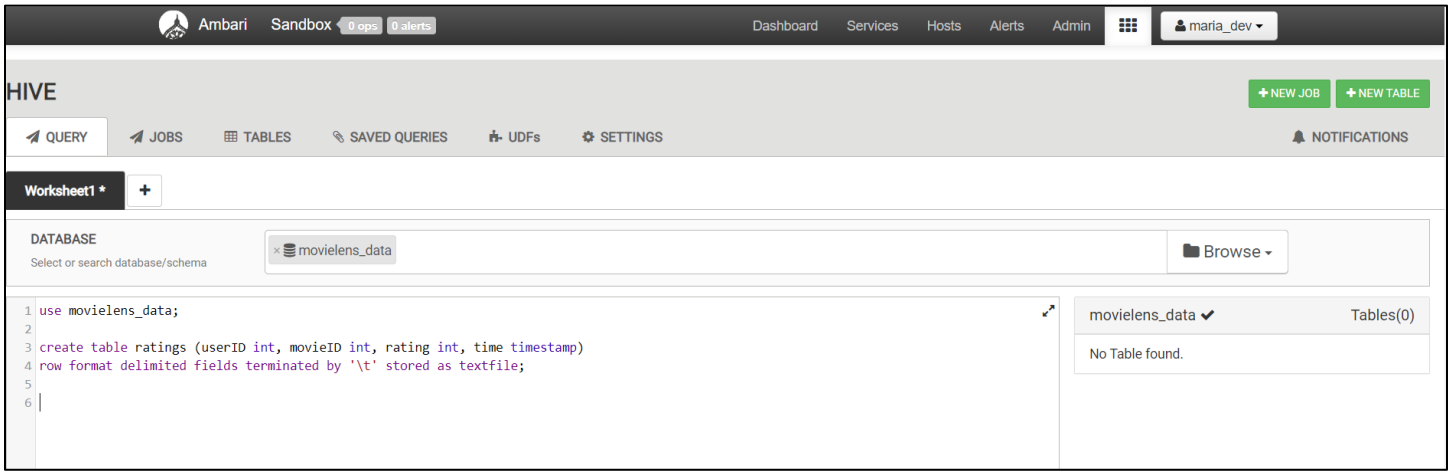
```
[maria_dev@sandbox-hdp ~]$ hadoop fs -mkdir movielens-data
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - maria_dev hdfs          0 2020-05-21 04:53 movielens-data
[maria_dev@sandbox-hdp ~]$ pwd
/home/maria_dev
[maria_dev@sandbox-hdp ~]$ wget http://media.sundog-soft.com/hadoop/ml-100k/u.da
ta
--2020-05-21 04:56:07--  http://media.sundog-soft.com/hadoop/ml-100k/u.data
Resolving media.sundog-soft.com (media.sundog-soft.com)... 52.216.86.203
Connecting to media.sundog-soft.com (media.sundog-soft.com)|52.216.86.203|:80...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: 2079229 (2.0M) [application/octet-stream]
Saving to: 'u.data'

100%[======================================>] 2,079,229   1.11MB/s   in 1.8s

2020-05-21 04:56:09 (1.11 MB/s) - 'u.data' saved [2079229/2079229]
```

```
[maria_dev@sandbox-hdp ~]$ wget http://media.sundog-soft.com/hadoop/ml-100k/u.item
--2020-05-21 04:56:49--  http://media.sundog-soft.com/hadoop/ml-100k/u.item
Resolving media.sundog-soft.com (media.sundog-soft.com)... 52.216.146.107
Connecting to media.sundog-soft.com (media.sundog-soft.com)|52.216.146.107|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 236344 (231K) [application/octet-stream]
Saving to: 'u.item'

100%[=========================================================================================================

2020-05-21 04:56:49 (1.57 MB/s) - 'u.item' saved [236344/236344]

[maria_dev@sandbox-hdp ~]$ ls
u.data  u.item
[maria_dev@sandbox-hdp ~]$
```

Copying files from local file system to HDFS –

```
[maria_dev@sandbox-hdp ~]$ hadoop fs -copyFromLocal u.data movielens-data/u.data
[maria_dev@sandbox-hdp ~]$ hadoop fs -copyFromLocal u.item movielens-data/u.item
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - maria_dev hdfs          0 2020-05-21 05:06 movielens-data
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls movielens-data
Found 2 items
-rw-r--r--   1 maria_dev hdfs    2079229 2020-05-21 05:06 movielens-data/u.data
-rw-r--r--   1 maria_dev hdfs     236344 2020-05-21 05:06 movielens-data/u.item
[maria_dev@sandbox-hdp ~]$
```

Creating a table from Ambari UI inside Hive for 'u.data' file to be imported –



Loading data from HDFS to the Hive table –

```
use movielens_data;

LOAD DATA INPATH '/user/maria_dev/movielens-data/u.data' OVERWRITE INTO TABLE ratings
```

Now, from the second file 'u.item' only first two columns are needed – so I will create a temp table to dump the entire dataset. Then I will select only those two columns which I need from the temp table to overwrite into a new 'movienames' dataset

To load in the movie names data from 'u.item' – first I will create a temp table

```
use movielens_data;

create table temp_movienames (col_value string);
```

And Load in the data into the temp table from HDFS –

```
use movielens_data;

LOAD DATA INPATH '/user/maria_dev/movielens-data/u.item' OVERWRITE INTO TABLE temp_movienames
```

The data loaded is as follows –

```
use movielens_data;


select * from temp_movienames limit 20;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

temp_movienames.col_value
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Copycat%20(1995)|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-1995||http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995

Now we will create the movienames table to hold the actual movienames data needed for analysis –

```
use movielens_data;


create table movienames (movieID int, movieName string);

```

I will create a query to Extract Data from 'temp_movienames' and Store It to 'movienames'

```
use movielens_data;

insert overwrite table movienames
SELECT
  split(col_value, '\\|')[0] movieID,
  split(col_value, '\\|')[1] movieName

from temp_movienames;
```

Movienames has been loaded as follows –

```
10 use movielens_data;
11
12 select * from movienames limit 10;
13
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

movienames.movieid	movienames.moviename
1	Toy Story (1995)
2	GoldenEye (1995)
3	Four Rooms (1995)
4	Get Shorty (1995)
5	Copycat (1995)
6	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
7	Twelve Monkeys (1995)

Loading the third table 'userInfo' directly from the Ambari UI

Hive | Query | Saved Queries | History | UDFs | Upload Table

Upload from Local | Upload from HDFS

File type: CSV | Select from local: Choose File u.user

Database: default | Table name: userInfo

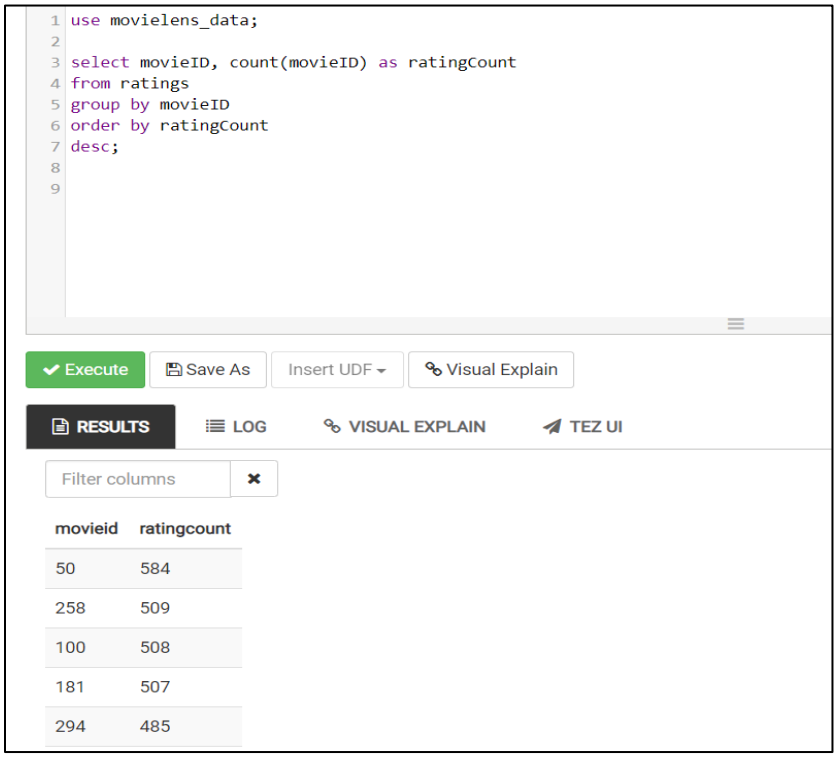
Stored as: TEXTFILE

userID	age	gender	occupation	zipcode
INT	INT	CHAR 1	STRING	INT
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

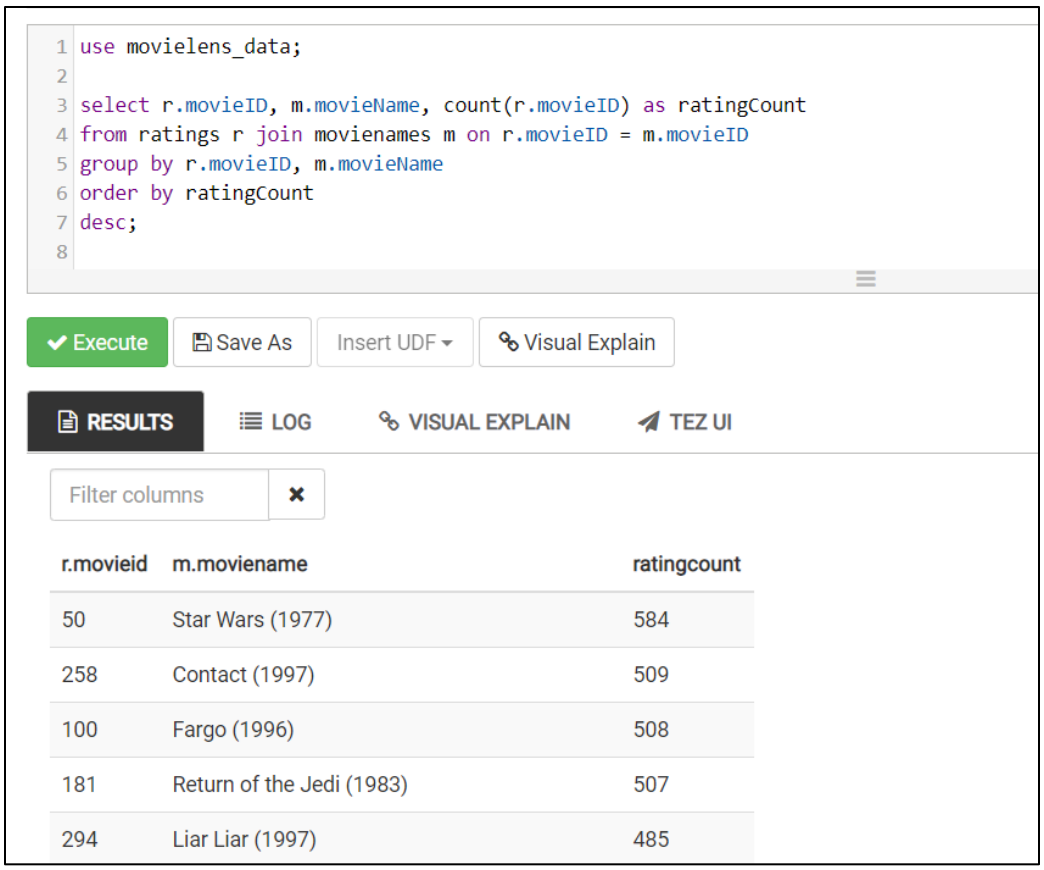
Data Analytics -

Now since the data has been prepared in Hive, it is time to do some Analytics –

1. How many times each movie is rated?



2. Now the ratings table can be joined with the movienames table to get the highest rated movienames –



3. Movie with the Highest Average rating? Here we select the movies who are rated by more than 10 people –

```
use movielens_data;

select * from (select m.movieName, CAST(avg(r.rating) AS decimal(10,2)) as avgrating, count(r.movieID) as ratingCount
from ratings r join movienames m on r.movieID = m.movieID
group by m.movieName) s
where s.ratingCount > 10
order by s.avgrating
desc;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

s.moviename	s.avgrating	s.ratingcount
Close Shave, A (1995)	4.49	112
Schindler's List (1993)	4.47	298
Wrong Trousers, The (1993)	4.47	118
Casablanca (1942)	4.46	243
Shawshank Redemption, The (1994)	4.45	283

4. Movies produced each year?

```
use movielens_data;

select s.year, count(*) as cnt from (select *, cast(substr(moviename, length(moviename) -4, 4) as int) as year from movienames) s
group by s.year
order by cnt
desc;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

s.year	cnt
1996	297
1995	294
1994	237
1997	235
1993	130
1998	53

5. Movie with the Lowest Average rating? Again, we select the movies who are rated by more than 10 people –

```
use movielens_data;

select * from (select m.movieName, CAST(avg(r.rating) AS decimal(10,1)) as avgrating, count(r.movieID) as ratingCount
from ratings r join movienames m on r.movieID = m.movieID
group by m.movieName) s
where s.ratingCount > 10
order by s.avgrating;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

s.moviename	s.avgrating	s.ratingcount
Children of the Corn: The Gathering (1996)	1.3	19
Amityville II: The Possession (1982)	1.6	14
Body Parts (1991)	1.6	13
Free Willy 3: The Rescue (1997)	1.7	27
Robocop 3 (1993)	1.7	11
Lawnmower Man 2: Beyond Cyberspace (1996)	1.7	21

6. Average age of users who rated per Movies –

```
use movielens_data;

create view if not exists movieWiseRatings as
select m.moviename, r.userid from movieNames m join ratings r on m.movieid = r.movieid;

select w.moviename, cast(avg(u.age) as decimal(10,1)) as avgAge
from movieWiseRatings w join userInfo u on w.userid = u.userid
group by w.moviename;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

w.moviename	avgage
'Til There Was You (1997)	24.7
1-900 (1994)	31
101 Dalmatians (1996)	32.4
12 Angry Men (1957)	35.2
187 (1997)	29.1

7. What is the percentage of Male & Female users who rated for each movie –

```
use movielens_data;

create view if not exists movieWiseUsers as
select m.moviename, r.userid from movieNames m join ratings r on m.movieid = r.movieid;

with cte as (select w.moviename, case when gender = 'M' then 1 end as M,
case when gender = 'F' then 1 end as F from userInfo u join movieWiseUsers w on u.userid = w.userid)

select moviename, cast((sum(m)/count(*)) * 100 as int) as pect_male, cast((sum(f)/count(*)) * 100 as int) as pect_female
from cte group by moviename;

```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

moviename	pect_male	pect_female
'Til There Was You (1997)	44	55
1-900 (1994)	80	20
101 Dalmatians (1996)	60	39
12 Angry Men (1957)	79	20
187 (1997)	75	24
2 Days in the Valley (1996)	81	18

8. Most users belonged to what occupation –

```
use movielens_data;

select occupation, count(*) as cnt from userinfo
group by occupation
order by cnt desc;
```

Execute | Save As | Insert UDF | Visual Explain

RESULTS | LOG | VISUAL EXPLAIN | TEZ UI

Filter columns

occupation	cnt
student	196
other	105
educator	95
administrator	79
engineer	67
programmer	66
librarian	51

References –

https://www.cloudera.com/tutorials/how-to-process-data-with-apache-hive/.html

Udemy course - https://www.udemy.com/course/the-ultimate-hands-on-hadoop-tame-your-big-data/

https://stackoverflow.com/questions/58609228/regexp-extract-value-from-pipe-delimited-string