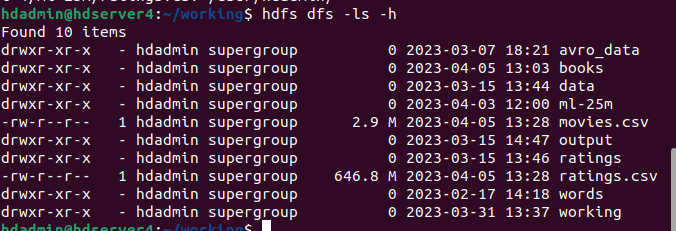
**Assignment 7**

Using HDFS

Q1. List the contents of the your home directory and use the human readable option. Show the command and the results.

hdfs dfs -ls -h

Ratings Distribution

We will calculate the distribution of each rating value using Pig Latin.

Load Ratings

Q2. Create the ratings relation by loading the data with full schema specification and filtering out the header row. Show all the statements you used to create this relation.  
  
ratings = LOAD '/user/hdadmin/ratings.csv' USING PigStorage(',') AS (userId:int, movieId:int, rating:double, timestamp:long);

ratings = FILTER ratings BY NOT (userId is null);

Compute Distribution

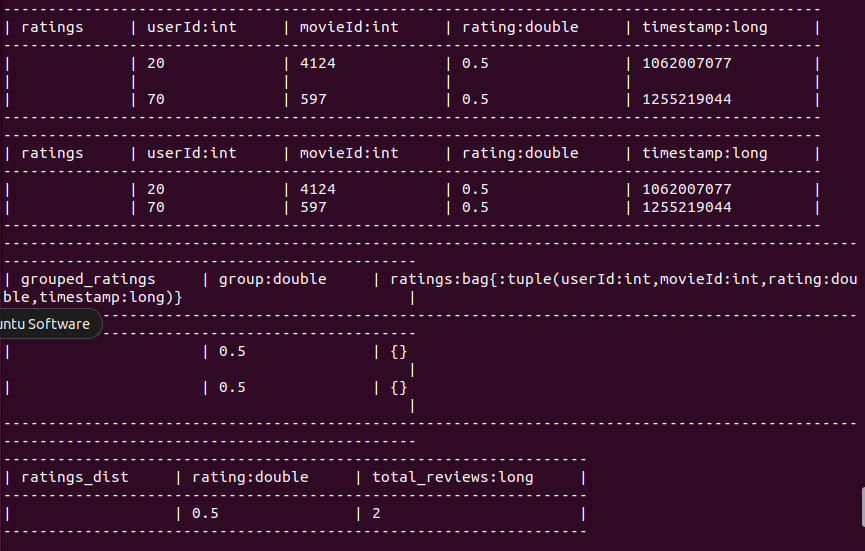
Q3. Create the ratings\_dist relation with each tuple having the fields (<rating>, <total\_reviews>). Show all of the statements you used to create this relation.

grouped\_ratings = GROUP ratings BY rating;

ratings\_dist = FOREACH grouped\_ratings GENERATE group AS rating, COUNT(ratings) AS total\_reviews;

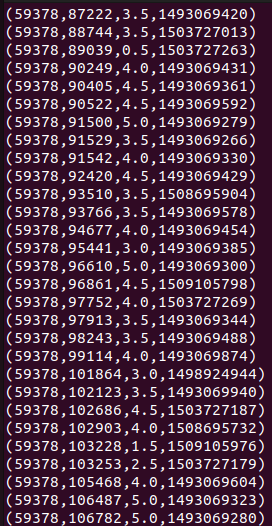
Q4. Illustrate the dataflow processing and place the pretty printed-results in your report.

ILLUSTRATE ratings\_dist;



\*\*\*Q5. Print the ratings\_dist relation to the screen. Show the command and the results.

DUMP ratings\_dist;



Movie Ratings Analysis

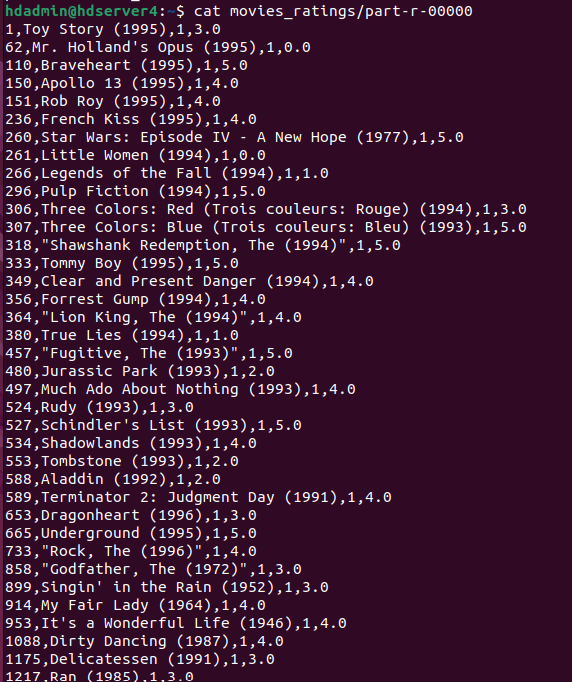
We compute the movie ratings using Pig.

Load Movies

The problem with our movie data is that the movie titles have commas in them so we can't just use PigStorage. Instead, you can use CSVExcelStorageLinks to an external site. fully qualified as org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES\_MULTILINE') to preserve the entire title surrounded by double quotes.

\*\*\*Q6. Create the movies relation by loading the data with full schema specification and filtering out the header row. Show the statements you used to create this relation.

movies = LOAD '/user/hdadmin/movies.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES\_MULTILINE') AS (movieId:int, title:chararray, genres:chararray);



Compute Movie Ratings

Each tuple in the final results should have the following fields (<movie id>, <movie title>, <total ratings>, <average rating>).

Q7. Describe in plain english the order of operations for your data flow solution.

1. Join the ratings and movies relations on the movieId field to combine the rating and movie information for each review.
2. Group the joined relation by the movieId and title fields to group all reviews for each movie together.
3. For each group, calculate the total number of ratings and the average rating using the COUNT and AVG functions.
4. Generate a new relation with one tuple for each group, where each tuple has the fields (<movie id>, <movie title>, <total ratings>, <average rating>).

Q8. Create the movieRatings relation and store it as movie\_ratings on HDFS. Show all the statements you used to create the relation.

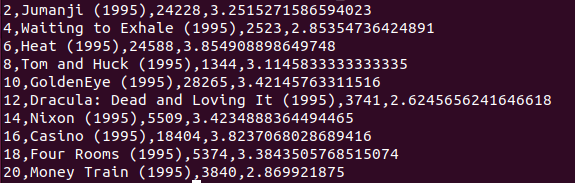
joined\_data = FOREACH (JOIN movies BY movieId, ratings BY movieId) GENERATE $0 AS movieId, $1 AS title, $5 AS rating;

movie\_ratings = FOREACH (GROUP joined\_data BY movieId) GENERATE group AS id, MAX($1.title) AS title, COUNT($1.rating) AS total, AVG($1.rating) AS avg;

STORE movie\_ratings INTO 'movie\_ratings' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES\_MULTILINE');

\*\*\*Q9. Illustrate the movieRatings relation and place the pretty-printed results in your report.

ILLUSTRATE movie\_ratings;



Analyze Movie Ratings

Instead of just looking up the movie ratings by movieId, let us find the most popular movies in the data.

\*\*\*Q10. Load the movie\_ratings data back in and find the top 5 movies by total number of ratings. Show all of the Pig Latin statements needed and the output tuples.

movieRatings = LOAD '/user/hdadmin/movie\_ratings' USING PigStorage(',') AS (movieId:int, title:chararray, total\_ratings:int, average\_rating:double);

top5\_movies = LIMIT (ORDER movieRatings BY total\_ratings DESC) 5;

DUMP top5\_movies;

