CA4022 - Assignment 1

Name: Cliodhna McAuley
Student Number: 18416482
Email: cliodhna.mcauley5@mail.dcu.ie

My objective in the assignment was to download the MovieLens dataset, clean the data and decide which files to merge and use, complete some basic analysis of the data in both Apache Pig and Apache Hive, and finally complete some more complex queries on the data in Apache Hive. My assignment is available at https://github.com/mcauleyc/CA4022_Assignment_1. All code for this assignment is located in 'Code'. The data visualisation code is available in 'Visualisation.ipynb'. The screenshots of the results are stored in Screenshots.

2. Loading and Cleaning the Data

The first thing I had to do was to load the data. I had a problem reading in 'movies.csv' using PigStorage(',') because it was separating on the commas within double quotes as well which was making a lot of rows wrong. I decided to use CSVExcelStorage() instead which was better able to handle these special commas. I began cleaning the data in Pig. First, I got rid of the headers by removing null values from the 'movieId' column. I split the title column into title and year using regular expressions. I tried using the substring function for this but it caused errors and it wouldn't have worked for titles that did not contain a year like 'Cosmos' anyway. The regular expressions I used were '([\\s\\S]+)\\b(?: |$)' to isolate the title and '\\((\\d{4})\\)' to get the year. I then split the genres on '|' so that each genre was separated by commas and contained in parentheses. Here is a screenshot of the code that I used.

```
grunt> mdata = LOAD 'hdfs:/user/cliodhna/ml-latest-small/movies.csv' USING org.apache.pig.piggyban
k.storage.CSVExcelStorage() AS (movieId:int, full_title:chararray, genres:chararray);

2021-10-21 09:52:58,343 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resou
rcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publis
her.enabled
grunt> mdata = FILTER mdata BY movieId is not null;
grunt>
grunt> movies = FOREACH mdata GENERATE movieId, REGEX_EXTRACT(REPLACE(full_title, ';', '-'), '([\\
s\\S]+)\\b(?: |$)', 1) as title, REGEX_EXTRACT(full_title, '\\((\\d{4})\\)', 1) as year, STRSPLIT(
genres, '\\|') as genres;
grunt>
grunt>
grunt> rdata = LOAD 'hdfs:/user/cliodhna/ml-latest-small/ratings.csv' USING org.apache.pig.piggyba
nk.storage.CSVExcelStorage() AS (userId:int, movieId:int, rating:float, timestamp:chararray);
2021-10-21 09:52:59,192 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resou
rcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publis
her.enabled
grunt> ratings = FILTER rdata BY movieId is not null;
grunt>
grunt> all_data = JOIN movies BY movieId, ratings BY movieId;
grunt> all_data = foreach all_data generate $0 as m_movieId, $1 as title, $2 as year, $3 as genres
, $4 as userId, $5 as r_movieId, $6 as rating, $7 as timestamp;
grunt> STORE all_data INTO ' hdfs:/user/cliodhna/pig_output/4' USING org.apache.pig.piggybank.stor
age.CSVExcelStorage(';');
2021-10-21 09:55:41,784 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resou
rcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publis
```

I then loaded in 'ratings.csv' which did not need cleaning apart from removing the headers. I decided to join movies and ratings into one dataset on 'movieId'. I don't think that links or tags data were necessary for this assignment so I did not include them in my dataset. When storing the dataset to a file, I used CSVExcelStorage(';') so that the delimiter would be a semicolon. This makes it easy to differentiate the commas in titles and genres. This caused one problem in the dataset which was the title "Steins;Gate the Movie: The Burden of Déjà vu" because this contains a semicolon. I decided to change the semicolon to a hyphen when reading the title to make it easier.

3. Simple Queries in Pig

To find the most rated movie I grouped by movieId, counted how many ratings there were for each movie, made each entry unique and then put in descending order of number of ratings. In this case Forrest Gump is the most rated with 329 ratings.

```
(356,329,Forrest Gump)
(318,317,Shawshank Redemption, The)
(296,307,Pulp Fiction)
```

Next I got the movies with only 5 stars. I did this by getting the average rating and ensuring it was equal to 5. There were only a few movies with only five stars and they did not have mayy ratings, the most was two.

```
(78836,2,5.0)
(3473,2,5.0)
(1151,2,5.0)
```

Then I found the movies that had the most 4 and 5 star ratings. I did this by filtering the data to only include 4 and 5 star ratings and then repeating the first query. The Shawshank Redemption had the most, with 225.

```
(318,225,Shawshank Redemption, The)
(356,210,Forrest Gump)
(296,200,Pulp Fiction)
```

I then found the movie with the majority of 5 star ratings. I did this in the same way as the previous query but only filtered by 5 star ratings. It is similar to the previous query but the 2nd and 3rd movies are switched.

```
(318,153,Shawshank Redemption, The)
(296,123,Pulp Fiction)
(356,116,Forrest Gump)
```

Finally for my analysis with Pig, I got the user with the highest average rating. I did this in a similar way to the first query but grouped by user and added the average rating. User 53 had the highest average rating with 5.0 and she had 20 ratings.

```
(53,20,5.0)
(251,23,4.869565217391305)
```

4. Simple Queries in Hive

I created a movie_rating table in hive and loaded the cleaned data from pig into it. Here is the code for that.

```
hive> CREATE TABLE IF NOT EXISTS movie_rating(m_movieId INT, title STRING, year INT, genres STRING, use
rId INT, r_movieId INT, rating FLOAT, r_timestamp STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ';'
    > STORED AS TEXTFILE
    > LOCATION 'hdfs:/user/cliodhna/pig_output/4/';
OK
Time taken: 0.817 seconds
hive> select * from movie_rating limit 10;
OK
1       Toy Story       1995    (Adventure,Animation,Children,Comedy,Fantasy)   314     1       3.0     8
34398280
```

I ran the same queries in Hive as I did in Pig. All of the methods are very similar but each could be done in one statement and they were easier overall. All of the results are the same as well.

```
356     Forrest Gump    329
318     Shawshank Redemption, The       317
296     Pulp Fiction    307
```

```
6818    Come and See (Idi i     2
6442    Belle époque    2
```

```
318     Shawshank Redemption, The       225
356     Forrest Gump    210
296     Pulp Fiction    200
```

```
318     Shawshank Redemption, The       153
296     Pulp Fiction    123
356     Forrest Gump    116
```

```
53      20      5.0
251     23      4.869565217391305
```

5. More Complex Queries

The first query was to find out the number of ratings for each star level. I did this by grouping by rating and counting each rating. Similarly the second part wanted to find the most popular rating so I used the same query and ordered it by the count in descending order. From this you can see that 4.0 is the most common rating and 0.5 is the rarest.

```
5.0     13211
4.5     8551
4.0     26818
3.5     13136
3.0     20047
2.5     5550
2.0     7551
1.5     1791
1.0     2811
0.5     1370
```

```
4.0     26818
3.0     20047
5.0     13211
3.5     13136
4.5     8551
2.0     7551
2.5     5550
1.0     2811
1.5     1791
0.5     1370
```

Finally, I wanted to find the rating distributed by genre. I did this by grouping by genres. As you can see the most popular genre is Comedy followed by Drama and Comedy/Romance.

```
(Comedy)        7196
(Drama) 6291
(Comedy,Romance)        3967
```

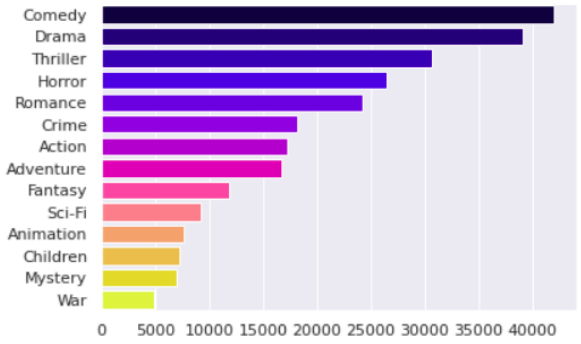
I then wanted to see how much of a difference it made if I did it based on individual genres rather than the grouping that are already there. Here is the code I used and I changed Comedy to be some of the most popular genres in the grouped categories.

```
hive> select count(*) as cnt from movie_rating
    > where genres like '%Comedy%';
```

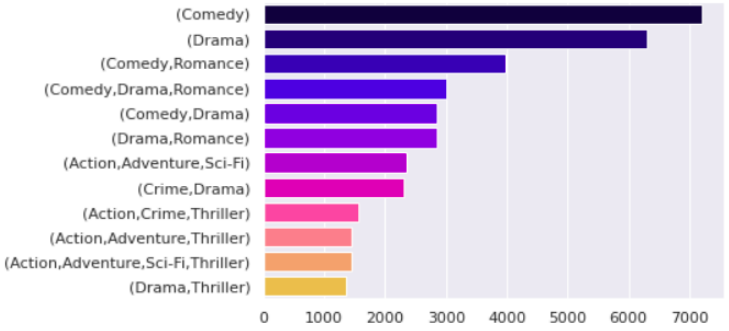
The result for this query for instance was 39053. The rest of the results were as follows, Drama: 41928, Thriller : 26452, Horror: 7291, Romance: 18124, Crime: 16681, Action: 30635, Adventure: 24161, Fantasy: 11834, Sci-Fi: 17243, Animation: 6988, Children: 9208, Mystery: 7674, War: 4859.

Visualisation

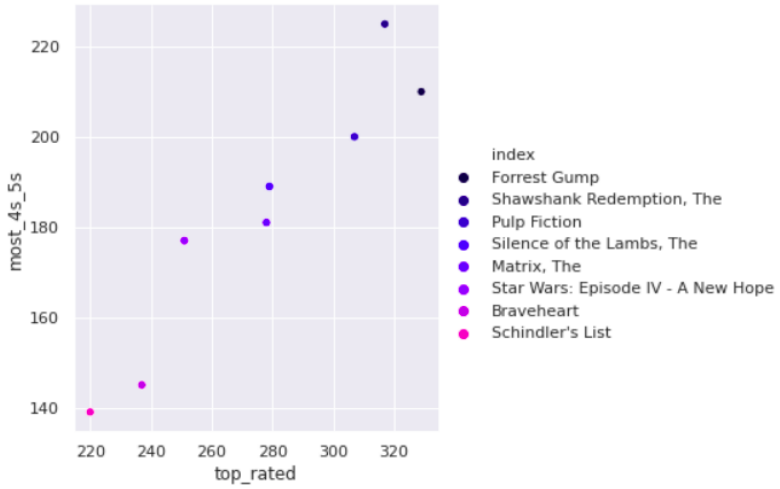
Firstly I wanted to do some basic visualisation of the distribution of ratings by genre. This makes it easier to understand which genres had the most ratings and which ones had the least. As you can see, Comedy and Drama were the most rated genres singularly.



I then wanted to see the distribution for the genres when they were grouped together and not split into individual categories. This shows that Comedy and Drama on their own are still the most popular and the following four highest all contain Comedy or Drama. Interestingly, Horror does not appear at all in the grouped genres whereas it is fourth in the individual rankings. This may be because all the other genres tend to go together more than Horror. For example, you would not often find Romantic Horror films.



Finally, I wanted to see if there was an obvious relationship between the movies with the most 4 and 5 star ratings, and the most rated movies. The top ten movies in both datasets were not the same so I removed the two different ones from each. As you can see, there is a relationship between the two. The top rated movie, ‘Forrest Gump’, has the second highest number of 4 and 5 star ratings. Similarly, ‘The Shawshank Redemption’, the movie with the most 4 and 5 stars, is the second top rated movie.



Conclusion

Overall, this assignment taught me a lot about Hadoop, Apache Pig, and Apache Hive. These new technologies were quite difficult to use at the start and very hard to set up. I was bombarded with errors in the beginning but after a lot of trial and error, and googling, I was able to fix my problems. One thing I did not like about Pig was that the error messages were very unclear, even when looking in the error logs. This made it more difficult than it needed to be to fix my problems. I also noticed that it was a lot easier to analyse the data in Hive because it is very similar to SQL. This assignment has made me feel a lot more comfortable using Hadoop in general.