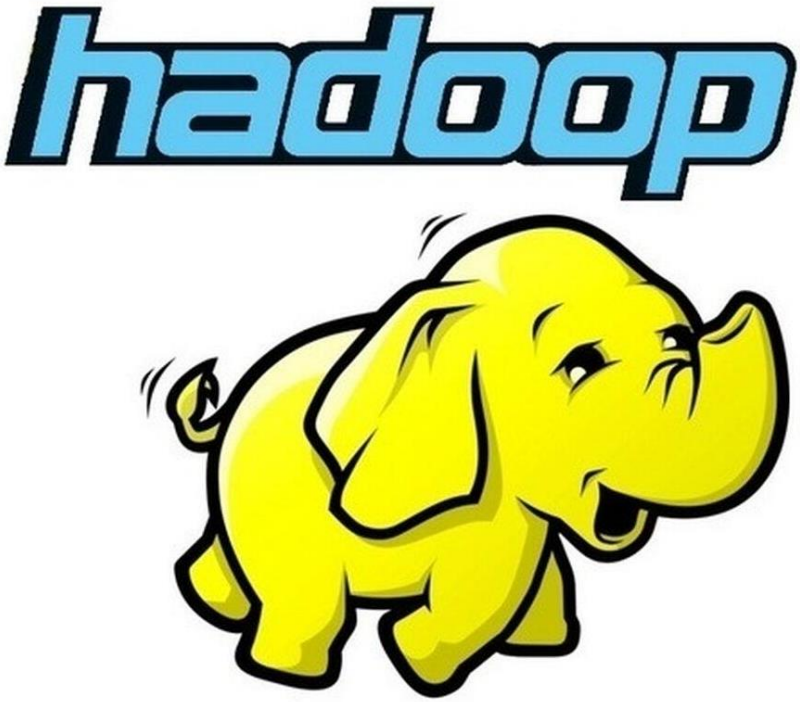


MDSC - 304(P)

Assignment - 2

MOVIE LENS DATA ANALYSIS

Krishnakanth G

20233

TABLE OF CONTENTS

MOVIE LENS DATA ANALYSIS

INTRODUCTION

The Movie Lens Dataset is most commonly used in recommender systems, which attempt to predict user movie ratings based on the ratings of other users. In this assignment, I wrote four key performance indicators (KPI's) in Hadoop MapReduce to analyze the Movie lens data.

MOVIE LENS DATA

Movie Lens' data is organized into three categories: movies, ratings, and users.

Movies

- Data format >> MovieID::Title::Genres
- MovieID is a unique id given to each movie ranging between 1 and 3952.
- Titles are identical to titles provided by the IMDB (including year of release).
- Genres are pipe, separated and are selected from the following genres: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film, Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War and Western.

Ratings

- Data format >> UserID::MovieID::Rating::Timestamp
- UserIDs is a unique id given to each user ranging between 1 and 6040.
- MovieID is a unique id given to each movie ranging between 1 and 3952.
- Ratings are made on a 5, star scale.
- Timestamp is represented in seconds since the epoch as returned by time.

Users

- Data format >> UserID::Gender::Age::Occupation::Zip-code
- Gender is denoted by a "M" for male and "F" for female.
- Age is chosen from the following ranges: 1: "Under 18", 18: "18, 24", 25: "25, 34", 35: "35, 44", 45: "45, 49", 50: "50, 55", and 56: "56+".
- Occupation is chosen from the following choices: 0: "other or not specified", 1: "academic/ educator", 2: "artist", 3: "clerical/admin", 4: "college/grad student", 5: "customer service", 6: "doctor/health care", 7: "executive/managerial", 8: "farmer", 9: "homemaker", 10: "K, 12 student", 11: "lawyer", 12: "programmer", 13: "retired", 14:

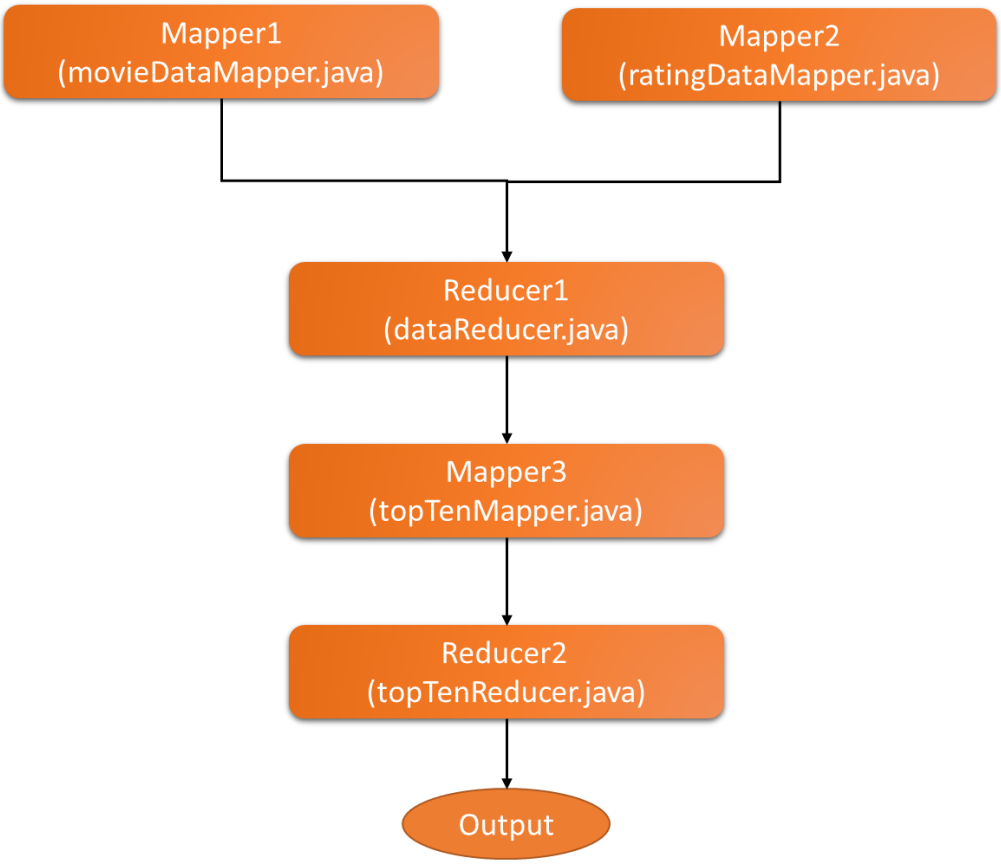
"sales/marketing", 15: "scientist", 16: "self, employed", 17: "technician/engineer", 18: "tradesman/craftsman", 19: "unemployed", and 20: "writer".

MOVIE LENS DATA ANALYSIS USING HADOOP MAPREDUCE

In this Assignment I am using four different key performance indicators (KPI's) for analyzing Movie Lens data. I explained each KPI briefly in the following sections:

KPI1 (Recommend top Ten most watched movies)

To overcome this problem, we need to count how many users watched each movie, which requires processing data from movies file & ratings file and finally output the top ten viewed movies. In MapReduce, we employ the concept of **Job chaining** to accomplish these multiple tasks.

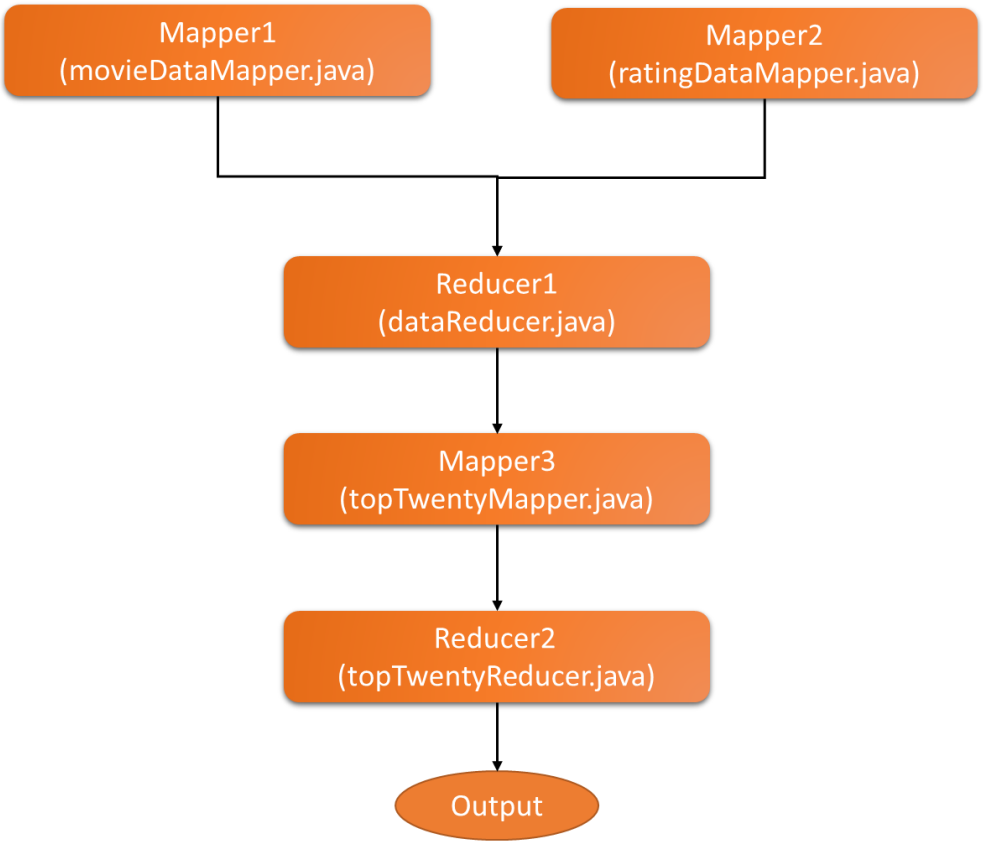


- From movieDataMapper, we obtain movie ID as key and concatenation of movie name & genre as value, and from ratingDataMapper, we get movie ID as key and 1 as value.
- At dataReducer, both mappers output is handled by combing and counting the views of each movie. Then, as an output, we obtain the movie name as a key and the total views as a value.

- Now the output of reducer (movie name & genre and total views) is given as input to toptenMapper where we find top ten local records.
- The output from the mapper is given to topTenReducer which finds the top ten global records and outputs total views as key and move name & genre as value.
- Tree Map class of java is used to find the top ten records.

KPI2 (Recommend top Twenty most Rated movies)

To overcome this problem, we need to know how many users watched and rated each movie, which requires processing data from movies file & ratings file and finally output the top twenty viewed movies. In MapReduce, we employ the concept of **Job chaining** to accomplish these multiple tasks.

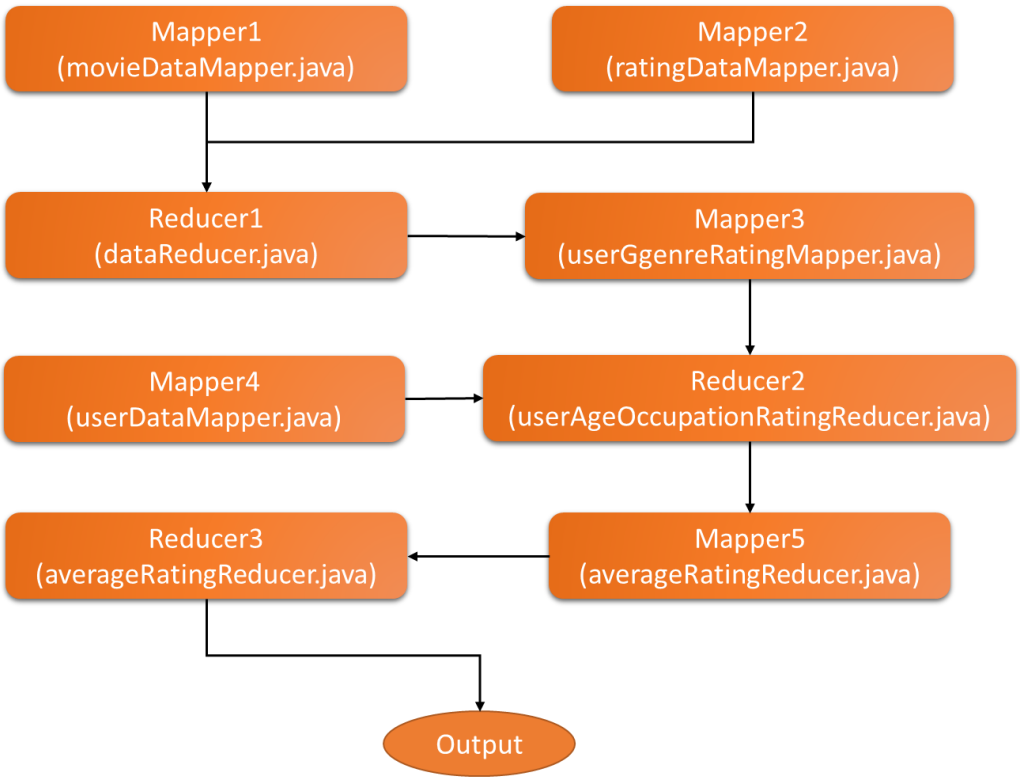


- From movieDataMapper, we obtain movie ID as key and concatenation of movie name & genre as value, and from ratingDataMapper, we get movie ID as key and rating as value.
- At dataReducer, both mappers output is handled by combining and calculating average rating for each movie. Then, as an output, we obtain the movie name & genre as a key and the average rating as value.
- Now the output of reducer (movie name & genre and average rating) is given as input to topTwentyMapper where we find top twenty local records.

- The output from the mapper is given to topTwentyReducer which finds the top ten global records and outputs average rating as key and move name & genre as value.
- Tree Map class of java is used to find the top twenty records.

KPI3 (Recommend the genres ranked by Average Rating, for each profession and age group)

To overcome this problem, we have to find average rating of a genre for each profession and age group. For that we have to process all the three files i.e., movies, ratings and users. We use the concept of **Job chaining** in MapReduce for doing these multiple tasks.

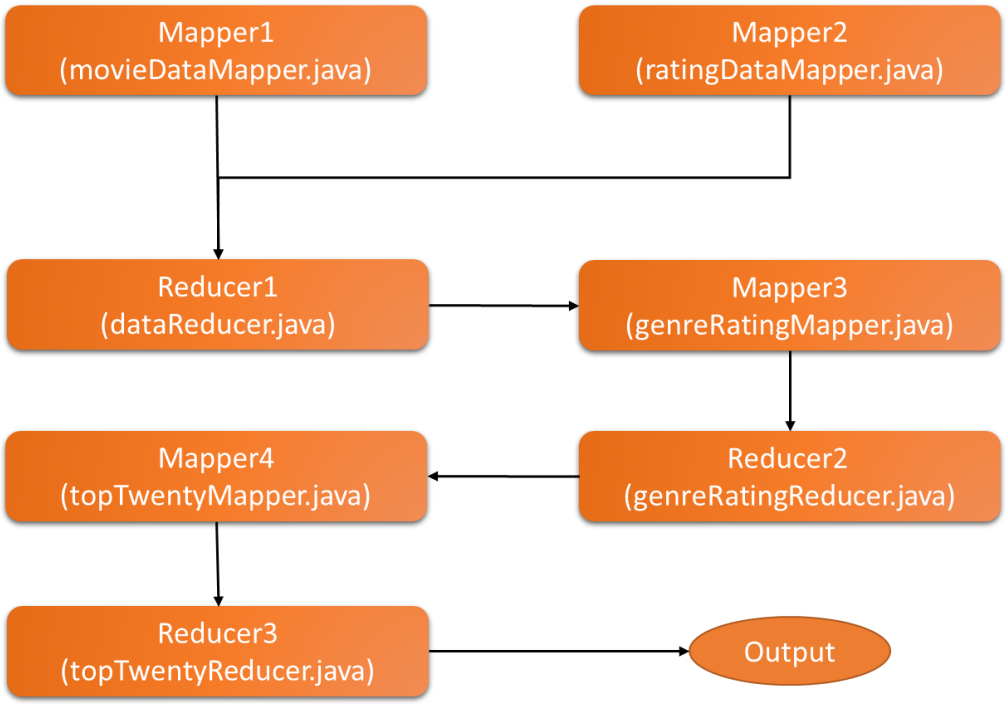


- From movieDataMapper, we get movie ID as key and genre as value, and from ratingDataMapper, we get movie ID as key and concatenation of user ID & rating as value.
- The output of both mappers is handled at dataReducer by creating a list of ratings for each movie with output, user ID as the key, and a concatenation of genre & ratings as the value.
- Output of the reducer is given as input to userGenreRatingMapper, which outputs user ID as key and concatenation of genre & rating as value.
- User's data is given as input to userDataMapper, which outputs user ID as key and concatenation of age & occupation as value.

- Now the output from both the mappers is given as input to the reducer userAgeOccupationRatingReducer, which outputs concatenation of age & occupation & genre as key and rating as value.
- The output of the reducer is given to averageRatingMapper where filtration based on age groups is done and concatenation of age & occupation & genre as key and rating as value are given as output.
- This is the final reducer which gets the input from the mapper and calculates the average rating. It produces average rating as key and concatenation of age & occupation & genre as value as the final output.

KPI4 (Recommend the genres ranked by Average Rating)

To overcome this problem, we have to find average rating of a genre which requires processing data from movies file & ratings file and then output the top twenty rated movies according to genre. We use the concept of **Job chaining** in MapReduce for doing these multiple tasks.



- From movieDataMapper, we obtain movie ID as key and concatenation of movie name & genre as value, and from ratingDataMapper, we get movie ID as key and rating as value.
- At dataReducer, both mappers output is handled by combining and calculating average rating for each movie. Then, as an output, we obtain the genre as key and the average rating by users for a movie as value.
- Now the output of reducer (genre and average rating by users for a movie) is given as input to genreRatingMapper which outputs the same to genreRatingReducer which

calculates average across common genres and output genre as key and average rating as value.

- Now the output of reducer (genre and average rating) is given as input to topTwentyMapper where we find top twenty local records.
- The output from the mapper is given to topTwentyReducer which finds the top ten global records and outputs average rating as key and genre as value.
- Tree Map class of java is used to find the top twenty records.

MOVIE LENS DATA ANALYSIS USING PYSPARK

I converted the four KPIs implemented in Hadoop MapReduce to PySpark, which is easier and faster to compute. In addition, the output for each KPI is the same in both platforms. To do PySpark work, I used the Data Bricks platform. In Hadoop, selecting the top ten or twenty records we need a separate MapReduce job but in Spark, it can be done in a single line. In comparison to Hadoop MapReduce, implementing KPIs in Spark is really convenient. Spark also outperforms Hadoop in terms of computing speed.

INFERENCES

KPI1

Command to run KPI1 in Hadoop

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hadoop jar KPI1.jar KPI1 /movie/movies /movie/ratings /movie/Im1 /movie/KPI1
2021-10-01 16:48:55,550 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
```

Output of KPI1

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hdfs dfs -cat /movie/KPI1/p*
2578    Silence of the Lambs, The (1991):Drama|Thriller
2583    Back to the Future (1985):Comedy|Sci-Fi
2590    Matrix, The (1999):Action|Sci-Fi|Thriller
2649    Terminator 2: Judgment Day (1991):Action|Sci-Fi|Thriller
2653    Saving Private Ryan (1998):Action|Drama|War
2672    Jurassic Park (1993):Action|Adventure|Sci-Fi
2883    Star Wars: Episode VI - Return of the Jedi (1983):Action|Adventure|Romance|Sci-Fi|War
2990    Star Wars: Episode V - The Empire Strikes Back (1980):Action|Adventure|Drama|Sci-Fi|War
2991    Star Wars: Episode IV - A New Hope (1977):Action|Adventure|Fantasy|Sci-Fi
3428    American Beauty (1999):Comedy|Drama
```

The KPI1 is to determine the top ten most viewed movies among users, and the KPI1 output is shown above. If a new user is looking for a movie, we can recommend any of the ten movies listed above. However, this KPI1 is not applicable to all types of users. Let's have a look at KPI2.

KPI2

Command to run KPI2 in Hadoop

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hadoop jar KPI2.jar KPI2 /movie/movies /movie/ratings /movie/Im2 /movie/KPI2
2021-10-01 16:55:14,613 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
```

Output of KPI2

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hdfs dfs -cat /movie/KPI2/p*
4.410714        World of Apu, The (Apur Sansar) (1959)::Drama
4.4128222       Casablanca (1942)::Drama|Romance|War
4.415608        Double Indemnity (1944)::Crime|Film-Noir
4.425647        To Kill a Mockingbird (1962)::Drama
4.4269404       Wallace & Gromit:: The Best of Aardman Animation (1996)
4.44989 Dr. Strangelove or:: How I Learned to Stop Worrying and Love the Bomb (1963)
4.452083        Third Man, The (1949)::Mystery|Thriller
4.4536943       Star Wars:: Episode IV - A New Hope (1977)
4.473913        Paths of Glory (1957)::Drama|War
4.4761906       Rear Window (1954)::Mystery|Thriller
4.4777246       Raiders of the Lost Ark (1981)::Action|Adventure
4.4914894       Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)::Film-Noir
4.5079365       Wrong Trousers, The (1993)::Animation|Comedy
4.5104165       Schindler's List (1993)::Drama|War
4.517106        Usual Suspects, The (1995)::Crime|Thriller
4.520548        Close Shave, A (1995)::Animation|Comedy|Thriller
4.5249662       Godfather, The (1972)::Action|Crime|Drama
4.554558        Shawshank Redemption, The (1994)::Drama
4.5605097       Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)::Action|Drama
4.6086955       Sanjuro (1962)::Action|Adventure
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$
```

The KPI2 is to determine the top twenty most rated movies by (more than 40) users, and the KPI2 output is shown above. If a new user is looking for a movie, we can recommend any of the twenty films listed above. This KPI2 is applicable for some types of users, however it fails if a user wishes to watch a movie from a specific genre. Let's have a look at KPI3.

KPI3

Command to run KPI3 in Hadoop

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hadoop jar KPI3.jar KPI3 /movie/movies /movie/ratings  /movie/Im3.1 /movie/users /movie/Im3.2 /movie/KPI3
2021-10-02 13:39:54,461 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2021-10-02 13:39:55,485 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your appl
```

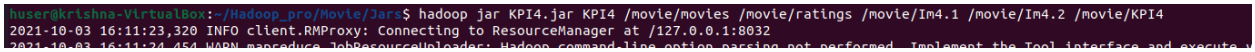
Output of KPI3

```
huser@krishna-VirtualBox:~/Hadoop_pro/Movie/Jars$ hdfs dfs -cat /movie/KPI3/p*
3.0822510822510822      18::0::Action
3.4917127071823204      18::0::Action|Adventure
3.5     18::0::Action|Adventure|Animation
2.6666666666666665      18::0::Action|Adventure|Animation|Children's|Fantasy
3.4444444444444446      18::0::Action|Adventure|Animation|Horror|Sci-Fi
2.1818181818181817      18::0::Action|Adventure|Children's|Comedy
2.1818181818181817      18::0::Action|Adventure|Children's|Sci-Fi
3.2241379310344827      18::0::Action|Adventure|Comedy
2.796875        18::0::Action|Adventure|Comedy|Crime
4.0     18::0::Action|Adventure|Comedy|Horror
3.5238095238095237      18::0::Action|Adventure|Comedy|Horror|Sci-Fi
3.8210526315789473      18::0::Action|Adventure|Comedy|Romance
3.813953488372093       18::0::Action|Adventure|Comedy|Sci-Fi
2.0     18::0::Action|Adventure|Comedy|War
2.7096774193548385      18::0::Action|Adventure|Crime
3.9642857142857144      18::0::Action|Adventure|Crime|Drama
2.6363636363636362      18::0::Action|Adventure|Crime|Thriller
2.838709677419355       18::0::Action|Adventure|Drama
3.3333333333333335      18::0::Action|Adventure|Drama|Romance
4.531914893617022       18::0::Action|Adventure|Drama|Sci-Fi|War
3.0     18::0::Action|Adventure|Drama|Thriller
3.24    18::0::Action|Adventure|Fantasy
3.9210526315789473      18::0::Action|Adventure|Fantasy|Sci-Fi
3.3793103448275863      18::0::Action|Adventure|Horror
3.0476190476190474      18::0::Action|Adventure|Horror|Thriller
3.64    18::0::Action|Adventure|Mystery
2.5     18::0::Action|Adventure|Mystery|Sci-Fi
3.2     18::0::Action|Adventure|Romance
4.318181818181818       18::0::Action|Adventure|Romance|Sci-Fi|War
```

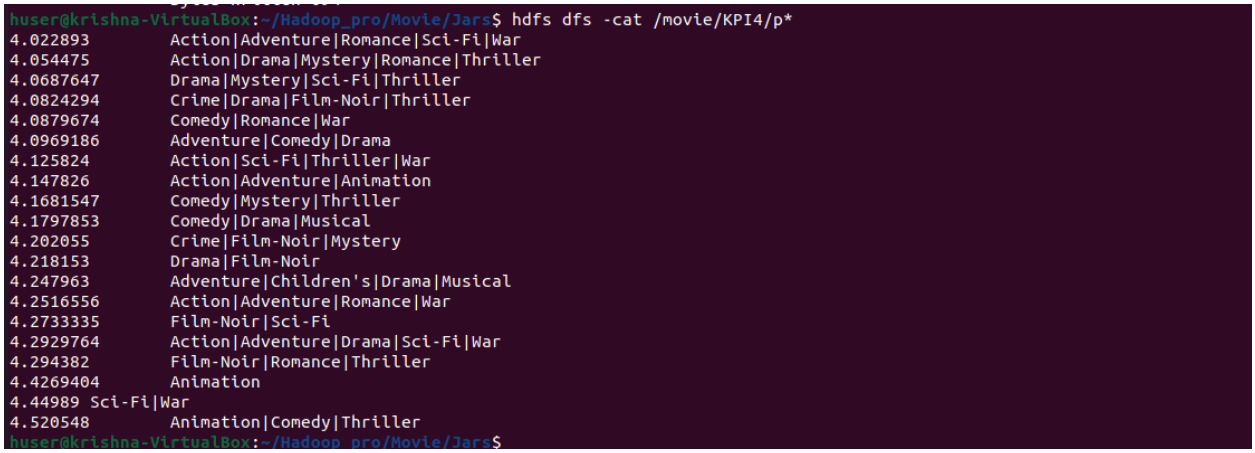
The KPI3 is to rank genres by average rating for each profession & age group, and the output for KPI3 is shown above. If a new user expresses an interest in a movie, we can recommend any movie from the output based on the user's age and profession. This KPI3 is applicable to all types of users. Let's take a look at one more KPI.

KPI4

Command to run KPI4 in Hadoop



Output of KPI4



The goal of KPI4 is to discover the top twenty genres ranked by average rating, and the result is shown above. If a new user shows an interest in a movie, we can recommend any of the twenty films listed above based on the user's genre preferences. This KPI4 is applicable to all types of users.

Code Link: https://github.com/krishnakanth-G/Hadoop/tree/main/Movie_lens_data_analysis

CONCLUSION

Following the analysis of the Movie Lens data using the four KPIs. In my opinion KPI3 and KPI4, which can propose movies to users based on their preferences, are the best approaches among all of these. If a person is interested in one of the genres, we can utilize KPI4 to select the top movie in that genre and recommend it to that user. If a user is interested in one of the genres and we have other information such as their age and profession, we can utilize KPI4 to recommend the top-rated movie in that genre for that profession and age group. The main learning from all those KPIs is job chaining which helps in making our job easy and flexible.

REFERENCES

https://github.com/srjsunny/Movie_Lens_Data-Analysis