Movie Review Analysis using PIG

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

G. Mayur Teja (AP20110010087)

G. Asish Karthikeya (AP20110010114)

K. Harika (AP20110010124)

Under the Guidance of

Sriramulu Bojjagani

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[November, 2023]

Certificate

Date: 30-Oct-23

This is to certify that the work present in this Project entitled "**Movie Review Analysis Using PIG**" has been carried out by **G. Mayur teja, G. Asish Karthikeya, K. Harika** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in the **School of Engineering and Sciences**.

Supervisor

(Signature)

Dr. Sriramulu Bojjagani

Assistant Professor,

Computer Science and Engineering.

Acknowledgements

I am grateful to Dr.Sriramulu Bojjagani for his ongoing support and fascinating report techniques. I also want to express my appreciation for them giving us their pearls of wisdom. I am grateful for his willingness to give up his time so freely.

I would also like to express my gratitude to SRM University, AP for assistance with analyzing the meteorological data, and for providing a good environment and facilities to complete this report successfully. I also appreciate the thoughtful criticism from the Books & Texts peer reviewers who remain anonymous.

I appreciate having the opportunity to work on this with everyone. My dissertation committee members have all given me a great deal of professional and personal advice and have taught me a lot about both scientific research and life in general.

Finally, I want to thank my parents for their inspiration and support throughout my academic career.

Table of Contents

Abstract

Movies have constantly functioned as a key source of entertainment since its debut in the late 18th century, delivering a diverse range of genres such as drama, comedy, science fiction, and action. The vast and diversified nature of acquired movie data needs a departure from standard analytical methodologies, advocating the use of big data analytics for complete insights. This research digs into the investigation and analysis of a movie ratings dataset using HDFS and Apache Pig, a platform for large-scale data processing. Using Pig's data processing capability, the research focuses on identifying movies with low average ratings, particularly those below 2.0, also Additionally, arranging the top 10 movies based on their highest average ratings. and gives a comprehensive assessment of their titles, average ratings, and number of ratings received. The findings shed light on cinematic works that may not have connected well with audiences, providing vital insights into the dynamics of movie tastes, and contributing to the expanding environment of data-driven film study.

Keywords: Big data, HDFS, Apache PIG

Abbreviations

SQL	Structured Query Language
HDFS	Hadoop Distributed File System

List of Figures

1.Introduction

The growth of user-generated material in the digital age has altered the landscape of information sharing and entertainment consumption. The massive volume of data created by internet movie reviews is one visible indication of this transition. Moviegoers share their thoughts on numerous platforms, producing a wealth of useful information for filmmakers, studios, and fans. It is essential to use big data technology to analyze movie reviews to get valuable insights from this enormous and dynamic dataset. The expansion of digital platforms has resulted in a flood of user-generated material, particularly in the form of movie reviews. These evaluations, which can be found on a variety of websites and forums, include a lot of information that may be used to identify patterns, trends, and attitudes associated to cinematic experiences. When confronted with the vast volume of data accessible, traditional techniques of analysis fall short, necessitating the use of modern big data technology.

Our study focuses on the use of Apache Pig and MapReduce Hadoop, two strong big data tools, to process and evaluate massive repositories of movie reviews. Apache Pig provides a high-level framework for expressing data analysis applications, allowing researchers to create complicated analysis workflows without wading into MapReduce programming complexities. Hadoop, on the other hand, is a distributed processing platform that is well-known for its scalability and efficiency when dealing with enormous datasets. The convergence of these technologies allows us to address the issues posed by large datasets by providing a scalable and parallelized method to movie review research. Our technique entails extracting important characteristics from textual data, sentiment analysis, and identifying major themes within reviews. We want to deliver a complete and effective solution for extracting meaningful insights from the large ocean of movie-related thoughts by using the parallel processing capabilities of MapReduce Hadoop and the data manipulation ease of Apache Pig.

In the subsequent sections, we will delve into the intricacies of our methodology, present the results of our analyses, and discuss the implications and potential applications of our findings in the broader context of the film industry. Through this research, we endeavor to demonstrate the efficacy of employing Pig and MapReduce Hadoop in the domain of movie review analysis, opening new avenues for the exploration of audience sentiments in the digital age.

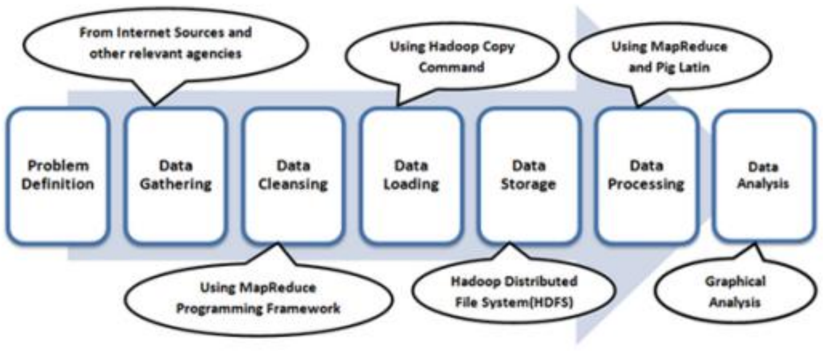
1.1 Applications:

- **Movie Recommendation System**: Analyzing movie ratings and identifying poorly rated movies can be a crucial step in enhancing a movie recommendation system. The application could use this analysis to filter out poorly rated movies from recommendations to improve user satisfaction.

- **Content Quality Assessment:** Media companies could use this script to assess the overall quality of their movie catalog. Identifying movies with low average ratings can help in deciding whether to promote, demote, or remove certain content from their offerings.

- **User Engagement Analysis**: For streaming platforms, understanding the average ratings and the number of ratings per movie can provide insights into user engagement. Low-rated movies with a high number of ratings may indicate controversial content that attracts audience attention.

Figure 1. Research Framework:


From Internet Sources and other relevant agencies
Using Hadoop Copy Command
Using MapReduce and Pig Latin
Problem Definition
Data Gathering
Data Cleansing
Data Loading
Data Storage
Data Processing
Data Analysis
Using MapReduce Programming Framework
Hadoop Distributed File System(HDFS)
Graphical Analysis

1.2 PIG: Working and Environment.

Apache Pig serves as an elevated platform crafted for the development of MapReduce programs within the Hadoop ecosystem, streamlining the intricate processes associated with large-scale data processing by offering an abstraction layer over the intricacies of traditional MapReduce programming. At its core lies Pig Latin, a scripting language meticulously designed to articulate data transformations in a concise and adaptable manner.

1. Pig Latin and Data Transformations:

Pig Latin Language: Functioning as a procedural scripting language, Pig Latin facilitates the articulation of data transformations. Its design prioritizes intuition over the complexities inherent in traditional MapReduce, enabling developers to concentrate on the logic of data processing rather than the nuances of parallelization and optimization.

Data Transformation Tasks: Pig scripts find particular efficacy in tasks such as Extract, Transform, Load (ETL), data analysis, and iterative data processing. Pig accommodates various data types, encompassing both structured and unstructured data, while providing an array of operations including filtering, joining, ordering, and more.

2. Data Sources and Versatility:

Data Sources: Pig seamlessly manages data from diverse sources, spanning databases, local files, and Hadoop's Distributed File System (HDFS). This adaptability positions it as a valuable tool for organizations dealing with a myriad of data formats and storage systems.

Extensibility: Pig's functionality can be extended by developers through the creation of their own user-defined functions (UDFs) for tasks such as reading, processing, and writing data. This inherent extensibility enhances Pig's adaptability to a spectrum of data processing requirements.

3. Parallel Execution and Optimization:

Parallel Processing: A standout feature of Pig is its adeptness in handling parallel processing. Tasks are automatically optimized by breaking them into sub-tasks that can execute concurrently across a Hadoop cluster. This capability is indispensable for managing voluminous datasets and leveraging the distributed architecture of Hadoop.

Task Optimization: Pig abstracts away the intricacies associated with parallel execution, empowering developers to devise complex data transformations without delving into the underlying architectural complexities or optimization intricacies. This characteristic renders Pig an accessible tool for a diverse range of data scientists and engineers.

4. Use Cases and Popularity:

ETL and Data Analysis: Pig finds common application in ETL tasks, where data is extracted from diverse sources, transformed, and loaded into a data warehouse or analytical system. Its straightforward nature also makes it a preferred choice for exploratory data analysis.

Iterative Data Processing: The platform's support for iterative data processing renders it suitable for algorithms requiring multiple iterations, such as machine learning algorithms.

Ease of Use: Pig's simplicity and flexibility, coupled with its support for parallel processing, position it as a favored choice among data scientists and engineers working with substantial datasets. This allows them to emphasize data analysis while minimizing the complexities associated with writing MapReduce programs.

In summary, Apache Pig's capacity to handle diverse data types, facilitate complex data operations, and execute tasks in parallel establishes it as an indispensable tool within the Hadoop ecosystem. Its design principles of simplicity, extensibility, and parallelism contribute significantly to its popularity among developers and data practitioners involved in large-scale

2.Methodology

2.1 Dataset

This research leverages two distinct datasets, comprises movie-related information and is sourced from **“github”.** The data includes details on various movies, such as titles, release dates, IMDb URLs, and genre classifications.

1. **Movie Metadata Dataset**

The first dataset, we referred it as the "Movie Metadata Dataset, It includes comprehensive information about various movies, such as titles, release dates, IMDb URLs, and genre classifications.

The dataset is structured in tabular form, with each row representing a distinct movie. The columns include:

MovieID: A unique identifier for each movie.

Title: The title of the movie.

ReleaseDate: The date of the movie's release.

URL: The IMDb URL associated with the movie.

Genres: Multiple genre columns (Genre1, Genre2, ..., GenreN) indicating the genres of each movie.

2. **User Ratings Dataset**

The second dataset we referred it as "User Ratings Dataset,"

Similar to the Movie Metadata Dataset, the User Ratings Dataset is structured in tabular form. The columns include:

UserID: A unique identifier for each user.

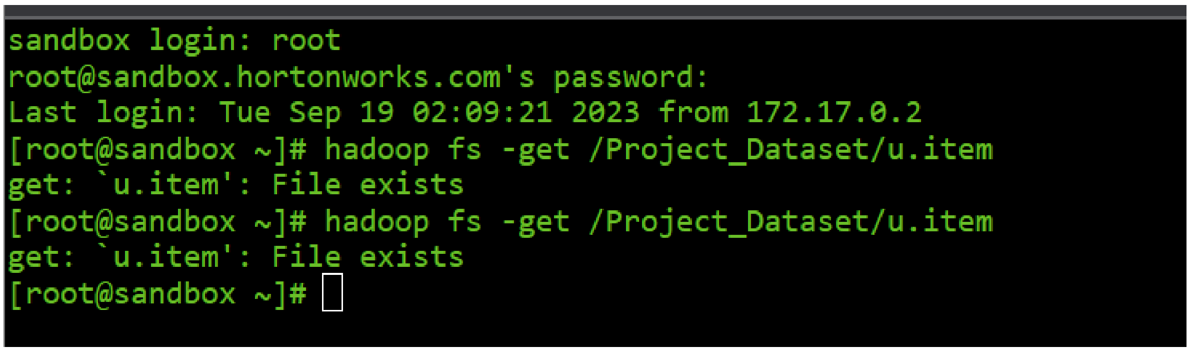
MovieID: The unique identifier corresponding to each movie.

Rating: The user-assigned rating for the movie on a numerical scale.

2.2 Procedure

2.2.1 Movie Analysis

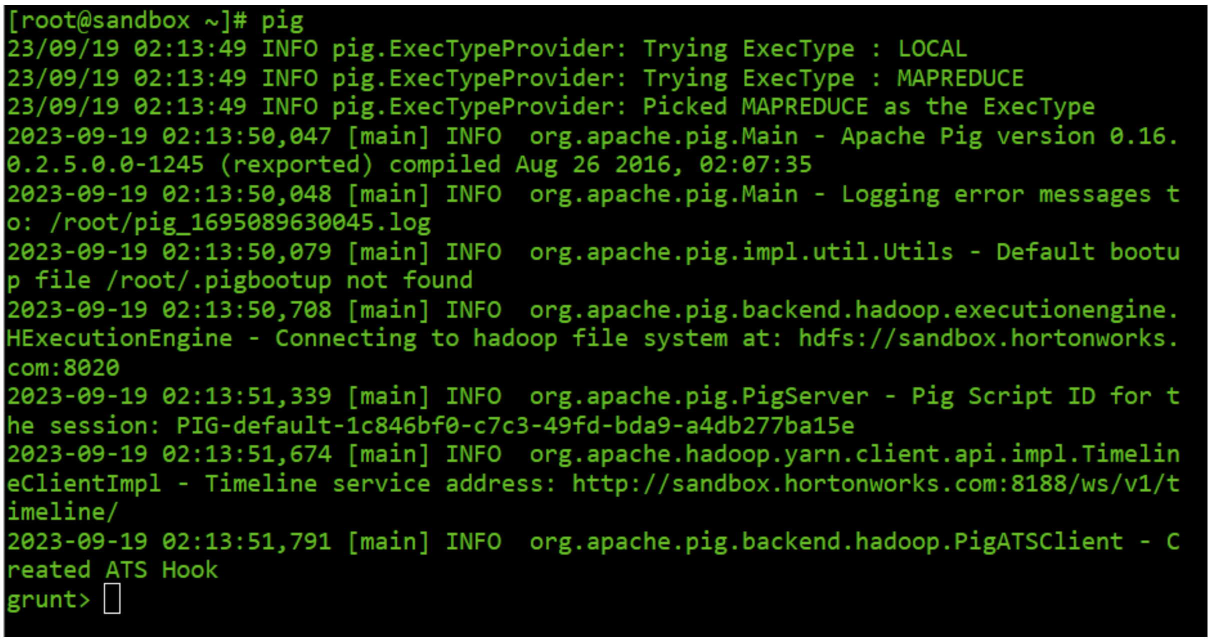
1. **Data Ingestion and HDFS Setup:** Begin by setting up the Hadoop Distributed File System (HDFS) environment. Upload the provided datasets mentioned earlier into the newly created folder using the 'get' command.



```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Tue Sep 19 02:09:21 2023 from 172.17.0.2
[root@sandbox ~]# hadoop fs -get /Project_Dataset/u.item
get: `u.item': File exists
[root@sandbox ~]# hadoop fs -get /Project_Dataset/u.item
get: `u.item': File exists
[root@sandbox ~]#
```

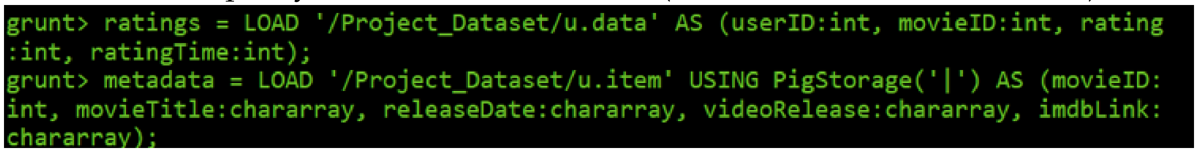
Figure 2. Sandbox Login & Data Ingestion

2. **Launching Pig:** Open the PIG shell by typing `**pig**` in the **command line** interface.



```
[root@sandbox ~]# pig
23/09/19 02:13:49 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
23/09/19 02:13:49 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
23/09/19 02:13:49 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2023-09-19 02:13:50,047 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.
0.2.5.0.0-1245 (rexported) compiled Aug 26 2016, 02:07:35
2023-09-19 02:13:50,048 [main] INFO  org.apache.pig.Main - Logging error messages t
o: /root/pig_1695089630045.log
2023-09-19 02:13:50,079 [main] INFO  org.apache.pig.impl.util.Utils - Default bootu
p file /root/.pigbootup not found
2023-09-19 02:13:50,708 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
HExecutionEngine - Connecting to hadoop file system at: hdfs://sandbox.hortonworks.
com:8020
2023-09-19 02:13:51,339 [main] INFO  org.apache.pig.PigServer - Pig Script ID for t
he session: PIG-default-1c846bf0-c7c3-49fd-bda9-a4db277ba15e
2023-09-19 02:13:51,674 [main] INFO  org.apache.hadoop.yarn.client.api.impl.Timelin
eClientImpl - Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/t
imeline/
2023-09-19 02:13:51,791 [main] INFO  org.apache.pig.backend.hadoop.PigATSClient - C
reated ATS Hook
grunt>
```

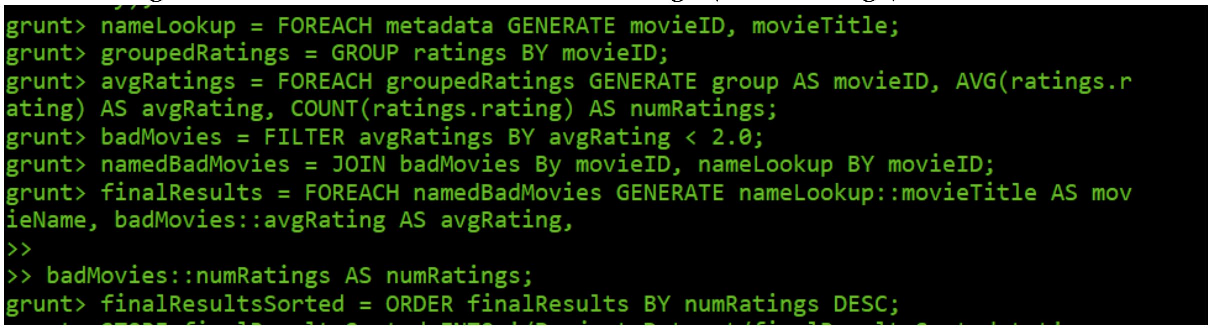
3. **Loading the Datasets**: The first line of the script loads the data located at 'Project_Dataaset'. The load function is used to load the data and PigStorage is used to specify the delimiter (comma in this case).



```
grunt> ratings = LOAD '/Project_Dataset/u.data' AS (userID:int, movieID:int, rating
:int, ratingTime:int);
grunt> metadata = LOAD '/Project_Dataset/u.item' USING PigStorage('|') AS (movieID:
int, movieTitle:chararray, releaseDate:chararray, videoRelease:chararray, imdbLink:
chararray);
```

Figure 3 .Loading Dataset

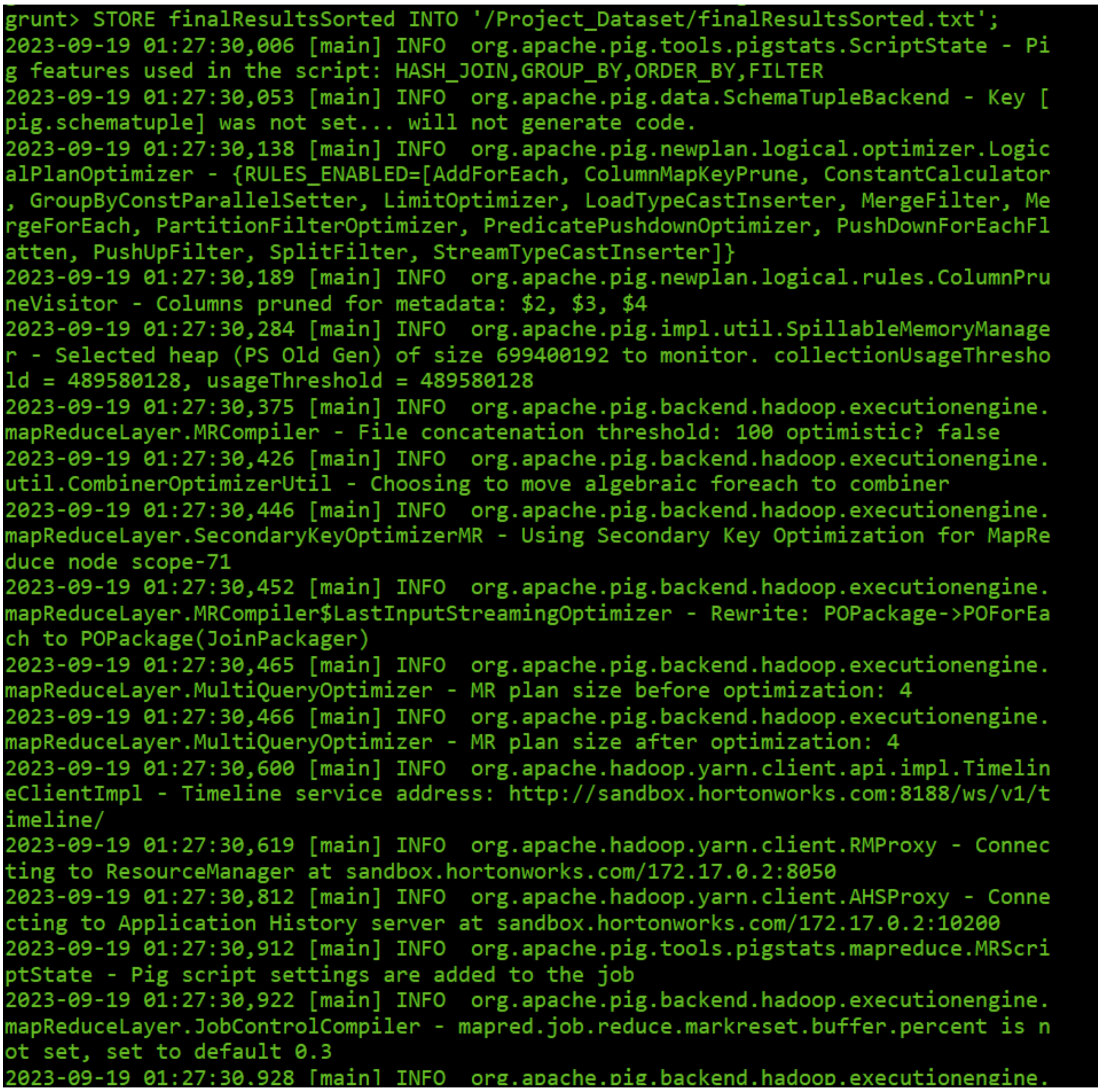
4. **Projection Operation (nameLookup):** In this step, a new relation named nameLookup is created by selecting only the movieID and movieTitle columns from the metadata relation.
5. **Grouping Operation (groupedRatings):** The ratings relation is grouped by the movieID field. This is a typical step before performing operations on groups of data.
6. **Average Calculation (avgRatings):** This step calculates the average rating (avgRating) and the total number of ratings (numRatings) for each movie. The results are stored in the avgRatings relation.
7. **Filtering Operation (badMovies):** Movies with an average rating less than 2.0 are filtered and stored in the badMovies relation**.**
8. **Join Operation (namedBadMovies):** The badMovies relation is joined with the nameLookup relation based on the common field movieID.
9. **Projection Operation (finalResults):** In this step, unnecessary columns are removed, and only the required columns (movieName, avgRating, numRatings) are selected for the final results.
10. **Ordering Operation (finalResultsSorted):** The final results are ordered in descending order based on the number of ratings (numRatings).



```
grunt> nameLookup = FOREACH metadata GENERATE movieID, movieTitle;
grunt> groupedRatings = GROUP ratings BY movieID;
grunt> avgRatings = FOREACH groupedRatings GENERATE group AS movieID, AVG(ratings.r
ating) AS avgRating, COUNT(ratings.rating) AS numRatings;
grunt> badMovies = FILTER avgRatings BY avgRating < 2.0;
grunt> namedBadMovies = JOIN badMovies By movieID, nameLookup BY movieID;
grunt> finalResults = FOREACH namedBadMovies GENERATE nameLookup::movieTitle AS mov
ieName, badMovies::avgRating AS avgRating,
>>
>> badMovies::numRatings AS numRatings;
grunt> finalResultsSorted = ORDER finalResults BY numRatings DESC;
```

Figure 4.Query

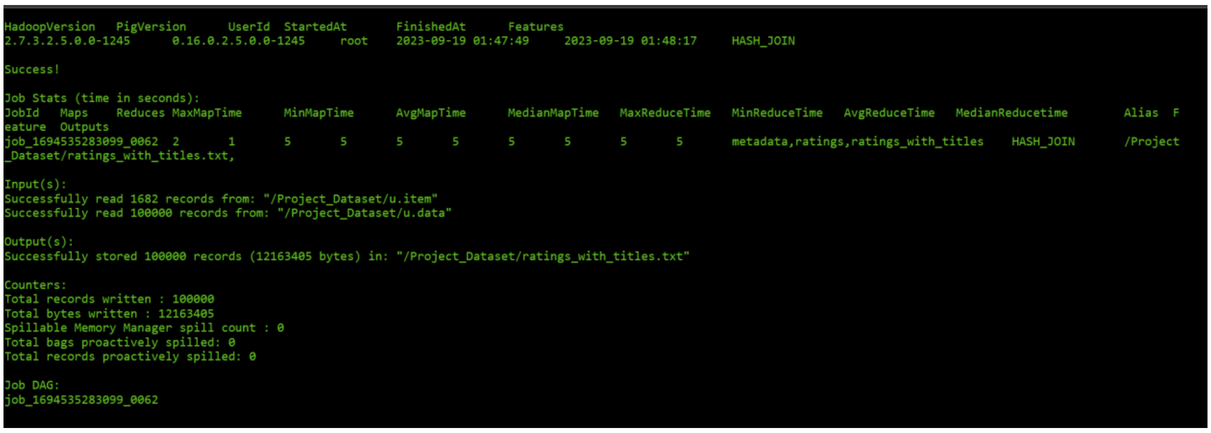
12. Storing the Results: The results are stored in to separate text file located at '/Project_Dataset/finalResultsSorted.txt'. The 'Store' function is used to store the results.



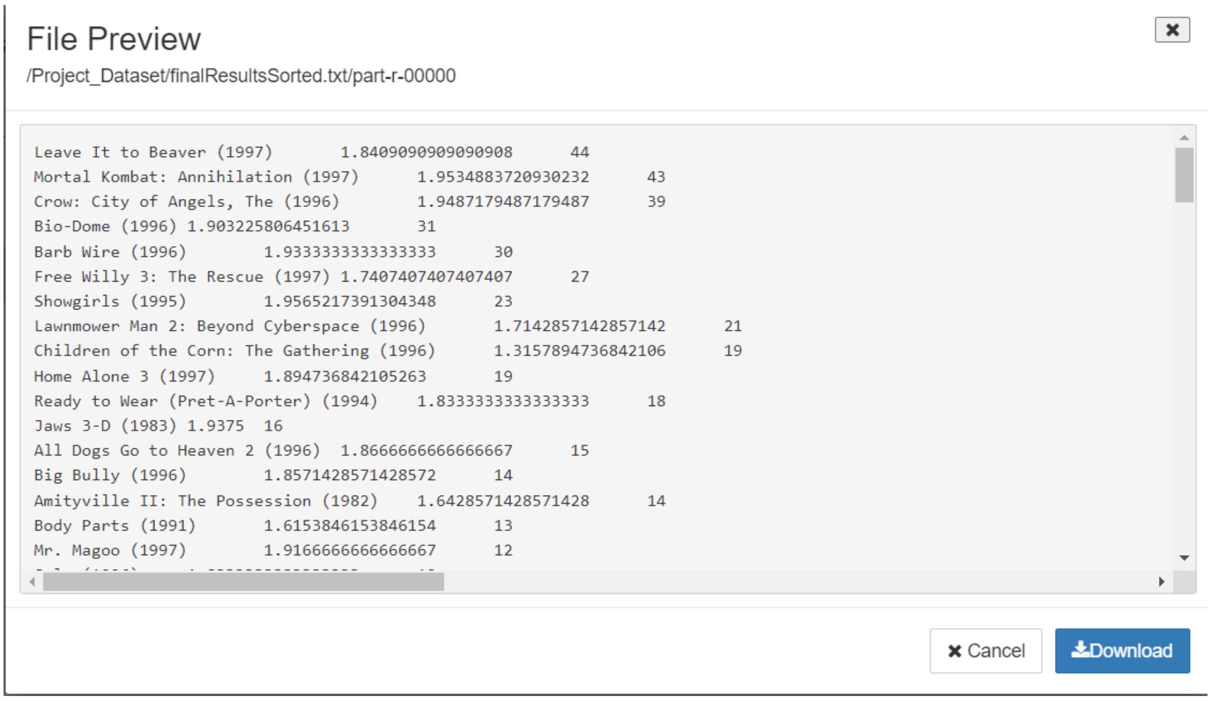
```
grunt> STORE finalResultsSorted INTO '/Project_Dataset/finalResultsSorted.txt';
2023-09-19 01:27:30,006 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pi
g features used in the script: HASH_JOIN,GROUP_BY,ORDER_BY,FILTER
2023-09-19 01:27:30,053 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [
pig.schematuple] was not set... will not generate code.
2023-09-19 01:27:30,138 [main] INFO  org.apache.pig.newplan.logical.optimizer.Logic
alPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator
, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, Me
rgeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFl
atten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2023-09-19 01:27:30,189 [main] INFO  org.apache.pig.newplan.logical.rules.ColumnPru
neVisitor - Columns pruned for metadata: $2, $3, $4
2023-09-19 01:27:30,284 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManage
r - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThresho
ld = 489580128, usageThreshold = 489580128
2023-09-19 01:27:30,375 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2023-09-19 01:27:30,426 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
util.CombinerOptimizerUtil - Choosing to move algebraic foreach to combiner
2023-09-19 01:27:30,446 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.SecondaryKeyOptimizerMR - Using Secondary Key Optimization for MapRe
duce node scope-71
2023-09-19 01:27:30,452 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.MRCompiler$LastInputStreamingOptimizer - Rewrite: POPackage->POForEa
ch to POPackage(JoinPackager)
2023-09-19 01:27:30,465 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 4
2023-09-19 01:27:30,466 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 4
2023-09-19 01:27:30,600 [main] INFO  org.apache.hadoop.yarn.client.api.impl.Timelin
eClientImpl - Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/t
imeline/
2023-09-19 01:27:30,619 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connec
ting to ResourceManager at sandbox.hortonworks.com/172.17.0.2:8050
2023-09-19 01:27:30,812 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Conne
cting to Application History server at sandbox.hortonworks.com/172.17.0.2:10200
2023-09-19 01:27:30,912 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScri
ptState - Pig script settings are added to the job
2023-09-19 01:27:30,922 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is n
ot set, set to default 0.3
2023-09-19 01:27:30,928 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
```

Figure 5. Storing the result

Output of this query :



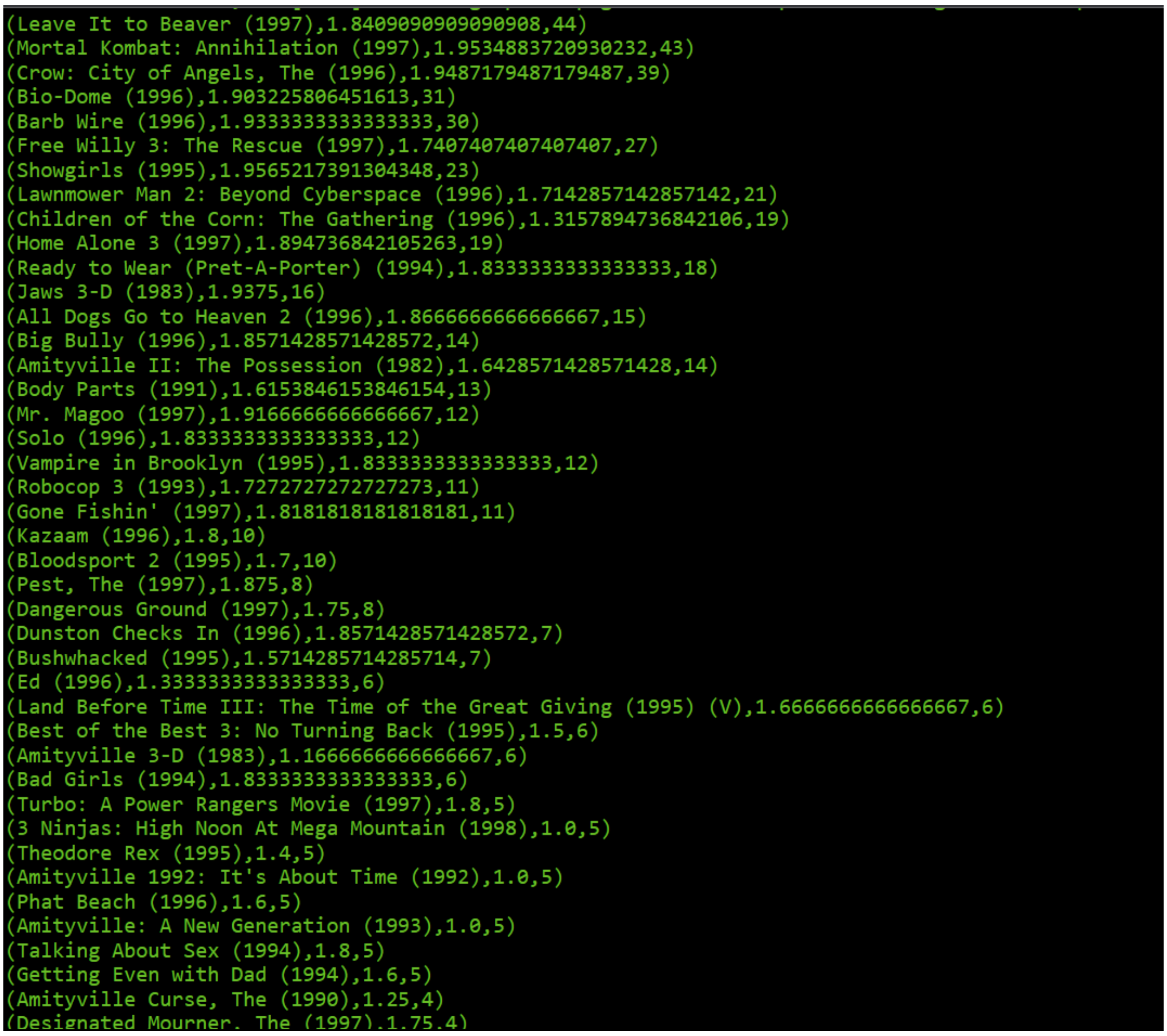
```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.7.3.2.5.0.0-1245      0.16.0.2.5.0.0-1245     root    2023-09-19 01:47:49     2023-09-19 01:48:17     HASH_JOIN

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   F
eature  Outputs
job_1694535283099_0062  2       1       5       5       5       5       5       5       5       5       metadata,ratings,ratings_with_titles    HASH_JOIN       /Project
_Dataset/ratings_with_titles.txt,

Input(s):
Successfully read 1682 records from: "/Project_Dataset/u.item"
Successfully read 100000 records from: "/Project_Dataset/u.data"

Output(s):
Successfully stored 100000 records (12163405 bytes) in: "/Project_Dataset/ratings_with_titles.txt"

Counters:
Total records written : 100000
Total bytes written : 12163405
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1694535283099_0062
```



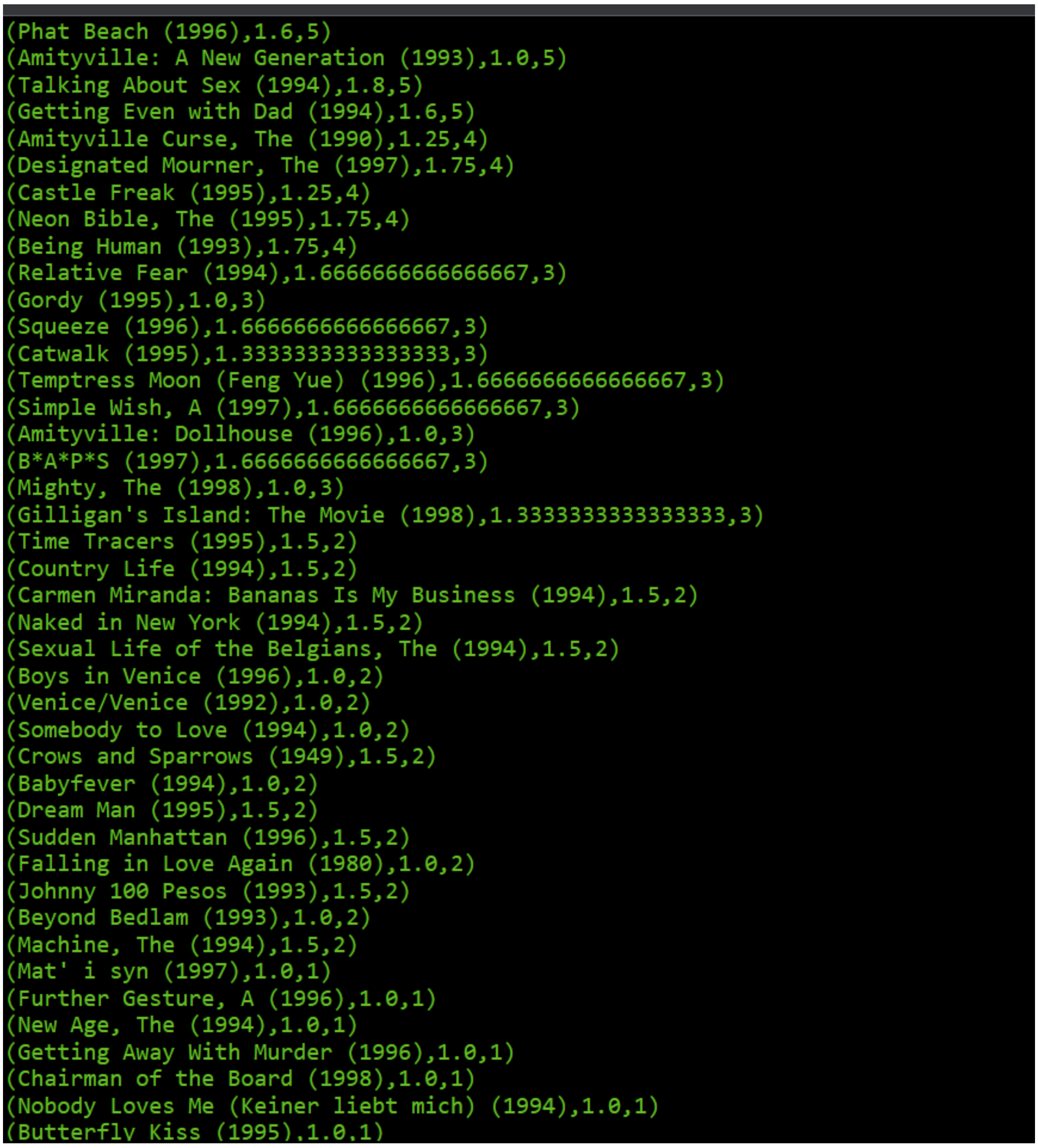
File Preview

/Project_Dataset/finalResultsSorted.txt/part-r-00000

```
Leave It to Beaver (1997)       1.8409090909090908      44
Mortal Kombat: Annihilation (1997)      1.9534883720930232      43
Crow: City of Angels, The (1996)        1.9487179487179487      39
Bio-Dome (1996) 1.903225806451613       31
Barb Wire (1996)        1.9333333333333333      30
Free Willy 3: The Rescue (1997) 1.7407407407407407      27
Showgirls (1995)        1.9565217391304348      23
Lawnmower Man 2: Beyond Cyberspace (1996)       1.7142857142857142      21
Children of the Corn: The Gathering (1996)      1.3157894736842106      19
Home Alone 3 (1997)     1.894736842105263       19
Ready to Wear (Pret-A-Porter) (1994)    1.8333333333333333      18
Jaws 3-D (1983) 1.9375  16
All Dogs Go to Heaven 2 (1996)  1.8666666666666667      15
Big Bully (1996)        1.8571428571428572      14
Amityville II: The Possession (1982)    1.6428571428571428      14
Body Parts (1991)       1.6153846153846154      13
Mr. Magoo (1997)        1.9166666666666667      12
```



```
(Leave It to Beaver (1997),1.8409090909090908,44)
(Mortal Kombat: Annihilation (1997),1.9534883720930232,43)
(Crow: City of Angels, The (1996),1.9487179487179487,39)
(Bio-Dome (1996),1.903225806451613,31)
(Barb Wire (1996),1.9333333333333333,30)
(Free Willy 3: The Rescue (1997),1.7407407407407407,27)
(Showgirls (1995),1.9565217391304348,23)
(Lawnmower Man 2: Beyond Cyberspace (1996),1.7142857142857142,21)
(Children of the Corn: The Gathering (1996),1.3157894736842106,19)
(Home Alone 3 (1997),1.894736842105263,19)
(Ready to Wear (Pret-A-Porter) (1994),1.8333333333333333,18)
(Jaws 3-D (1983),1.9375,16)
(All Dogs Go to Heaven 2 (1996),1.8666666666666667,15)
(Big Bully (1996),1.8571428571428572,14)
(Amityville II: The Possession (1982),1.6428571428571428,14)
(Body Parts (1991),1.6153846153846154,13)
(Mr. Magoo (1997),1.9166666666666667,12)
(Solo (1996),1.8333333333333333,12)
(Vampire in Brooklyn (1995),1.8333333333333333,12)
(Robocop 3 (1993),1.7272727272727273,11)
(Gone Fishin' (1997),1.8181818181818181,11)
(Kazaam (1996),1.8,10)
(Bloodsport 2 (1995),1.7,10)
(Pest, The (1997),1.875,8)
(Dangerous Ground (1997),1.75,8)
(Dunston Checks In (1996),1.8571428571428572,7)
(Bushwhacked (1995),1.5714285714285714,7)
(Ed (1996),1.3333333333333333,6)
(Land Before Time III: The Time of the Great Giving (1995) (V),1.6666666666666667,6)
(Best of the Best 3: No Turning Back (1995),1.5,6)
(Amityville 3-D (1983),1.1666666666666667,6)
(Bad Girls (1994),1.8333333333333333,6)
(Turbo: A Power Rangers Movie (1997),1.8,5)
(3 Ninjas: High Noon At Mega Mountain (1998),1.0,5)
(Theodore Rex (1995),1.4,5)
(Amityville 1992: It's About Time (1992),1.0,5)
(Phat Beach (1996),1.6,5)
(Amityville: A New Generation (1993),1.0,5)
(Talking About Sex (1994),1.8,5)
(Getting Even with Dad (1994),1.6,5)
(Amityville Curse, The (1990),1.25,4)
(Designated Mourner, The (1997),1.75,4)
```



```
(Phat Beach (1996),1.6,5)
(Amityville: A New Generation (1993),1.0,5)
(Talking About Sex (1994),1.8,5)
(Getting Even with Dad (1994),1.6,5)
(Amityville Curse, The (1990),1.25,4)
(Designated Mourner, The (1997),1.75,4)
(Castle Freak (1995),1.25,4)
(Neon Bible, The (1995),1.75,4)
(Being Human (1993),1.75,4)
(Relative Fear (1994),1.6666666666666667,3)
(Gordy (1995),1.0,3)
(Squeeze (1996),1.6666666666666667,3)
(Catwalk (1995),1.3333333333333333,3)
(Temptress Moon (Feng Yue) (1996),1.6666666666666667,3)
(Simple Wish, A (1997),1.6666666666666667,3)
(Amityville: Dollhouse (1996),1.0,3)
(B*A*P*S (1997),1.6666666666666667,3)
(Mighty, The (1998),1.0,3)
(Gilligan's Island: The Movie (1998),1.3333333333333333,3)
(Time Tracers (1995),1.5,2)
(Country Life (1994),1.5,2)
(Carmen Miranda: Bananas Is My Business (1994),1.5,2)
(Naked in New York (1994),1.5,2)
(Sexual Life of the Belgians, The (1994),1.5,2)
(Boys in Venice (1996),1.0,2)
(Venice/Venice (1992),1.0,2)
(Somebody to Love (1994),1.0,2)
(Crows and Sparrows (1949),1.5,2)
(Babyfever (1994),1.0,2)
(Dream Man (1995),1.5,2)
(Sudden Manhattan (1996),1.5,2)
(Falling in Love Again (1980),1.0,2)
(Johnny 100 Pesos (1993),1.5,2)
(Beyond Bedlam (1993),1.0,2)
(Machine, The (1994),1.5,2)
(Mat' i syn (1997),1.0,1)
(Further Gesture, A (1996),1.0,1)
(New Age, The (1994),1.0,1)
(Getting Away With Murder (1996),1.0,1)
(Chairman of the Board (1998),1.0,1)
(Nobody Loves Me (Keiner liebt mich) (1994),1.0,1)
(Butterfly Kiss (1995),1.0,1)
```

Output of the query with movie rating <2.0

2.2.2 Grouping and Aggregating: You can group the data by certain fields and perform aggregate operations. Finding the average rating for each user.

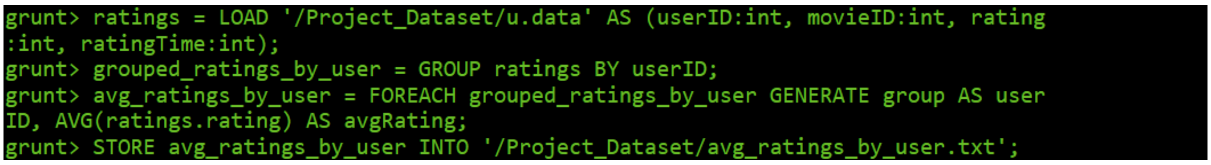
Script:

ratings = LOAD '/Project_Dataset/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

grouped_ratings_by_user = GROUP ratings BY userID;

avg_ratings_by_user = FOREACH grouped_ratings_by_user GENERATE group AS userID, AVG(ratings.rating) AS avgRating;

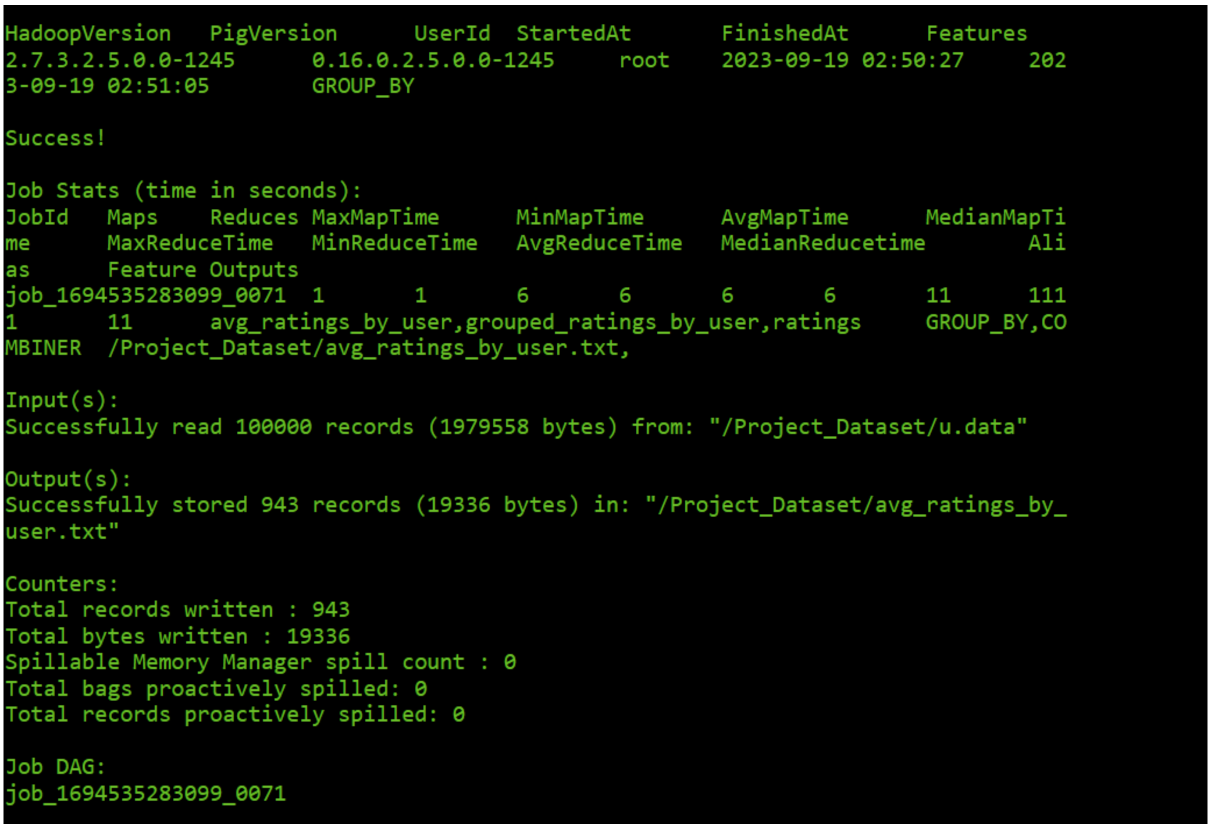
STORE avg_ratings_by_user INTO '/Project_Dataset/avg_ratings_by_user.txt';



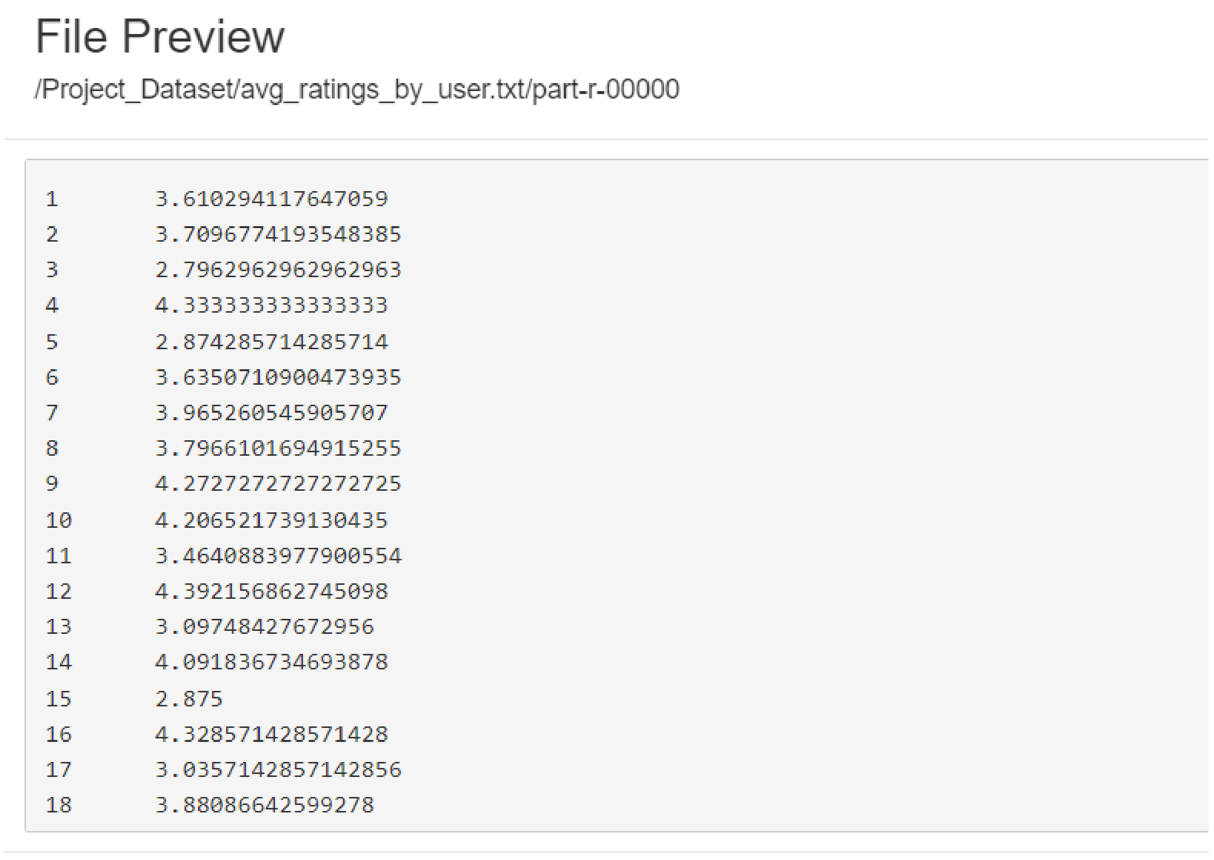
```
grunt> ratings = LOAD '/Project_Dataset/u.data' AS (userID:int, movieID:int, rating
:int, ratingTime:int);
grunt> grouped_ratings_by_user = GROUP ratings BY userID;
grunt> avg_ratings_by_user = FOREACH grouped_ratings_by_user GENERATE group AS user
ID, AVG(ratings.rating) AS avgRating;
grunt> STORE avg_ratings_by_user INTO '/Project_Dataset/avg_ratings_by_user.txt';
```

Figure 6 query

Output:



```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.7.3.2.5.0.0-1245      0.16.0.2.5.0.0-1245     root    2023-09-19 02:50:27     202
3-09-19 02:51:05        GROUP_BY

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTi
me      MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Ali
as      Feature Outputs
job_1694535283099_0071  1       1       6       6       6       6       11      111
1       11      avg_ratings_by_user,grouped_ratings_by_user,ratings     GROUP_BY,CO
MBINER  /Project_Dataset/avg_ratings_by_user.txt,

Input(s):
Successfully read 100000 records (1979558 bytes) from: "/Project_Dataset/u.data"

Output(s):
Successfully stored 943 records (19336 bytes) in: "/Project_Dataset/avg_ratings_by_
user.txt"

Counters:
Total records written : 943
Total bytes written : 19336
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1694535283099_0071
```



File Preview

/Project_Dataset/avg_ratings_by_user.txt/part-r-00000

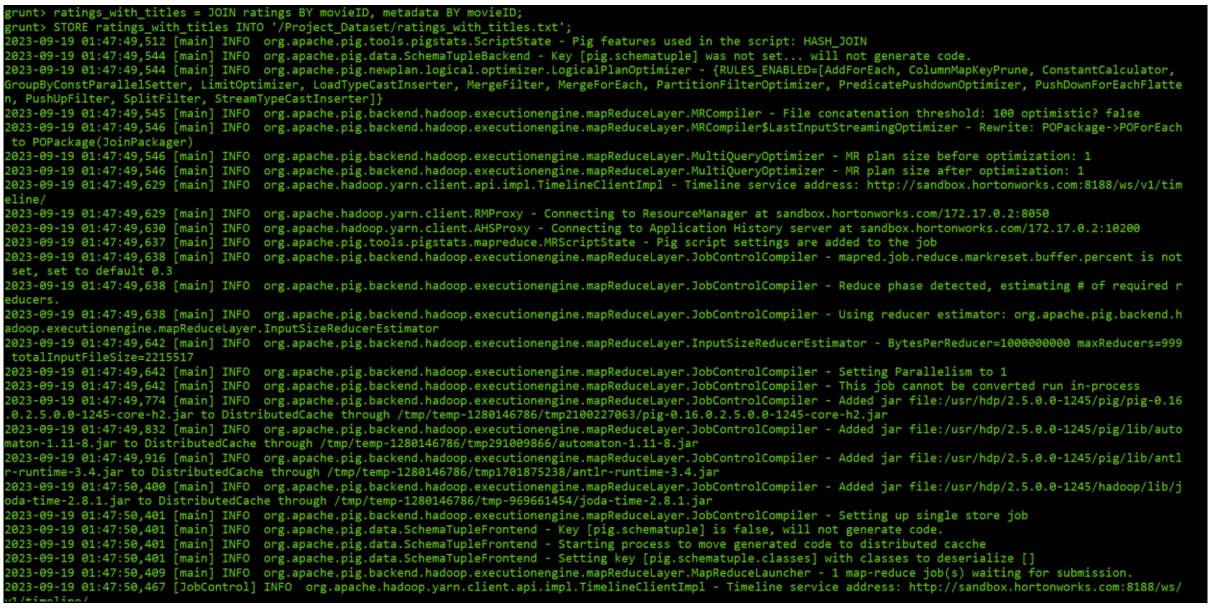
```
1       3.610294117647059
2       3.7096774193548385
3       2.7962962962962963
4       4.333333333333333
5       2.874285714285714
6       3.6350710900473935
7       3.965260545905707
8       3.7966101694915255
9       4.2727272727272725
10      4.206521739130435
11      3.4640883977900554
12      4.392156862745098
13      3.09748427672956
14      4.091836734693878
15      2.875
16      4.328571428571428
17      3.0357142857142856
18      3.88086642599278
```

2.2.3 **Joining**: You can join multiple datasets together. Joining the ratings and metadata to get movie titles in the ratings.

Script:

ratings_with_titles = JOIN ratings BY movieID, metadata BY movieID;

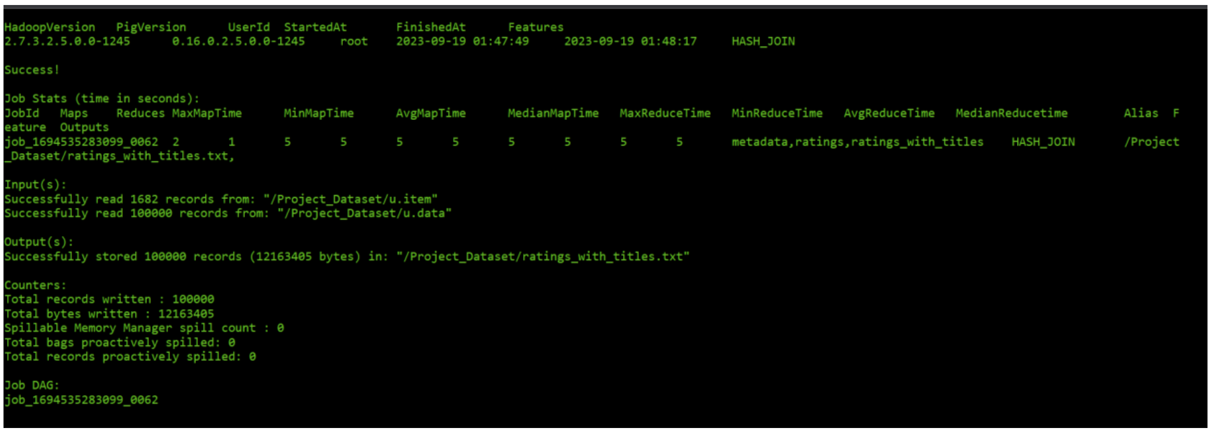
STORE ratings_with_titles INTO '/Project_Dataset/ratings_with_titles.txt';



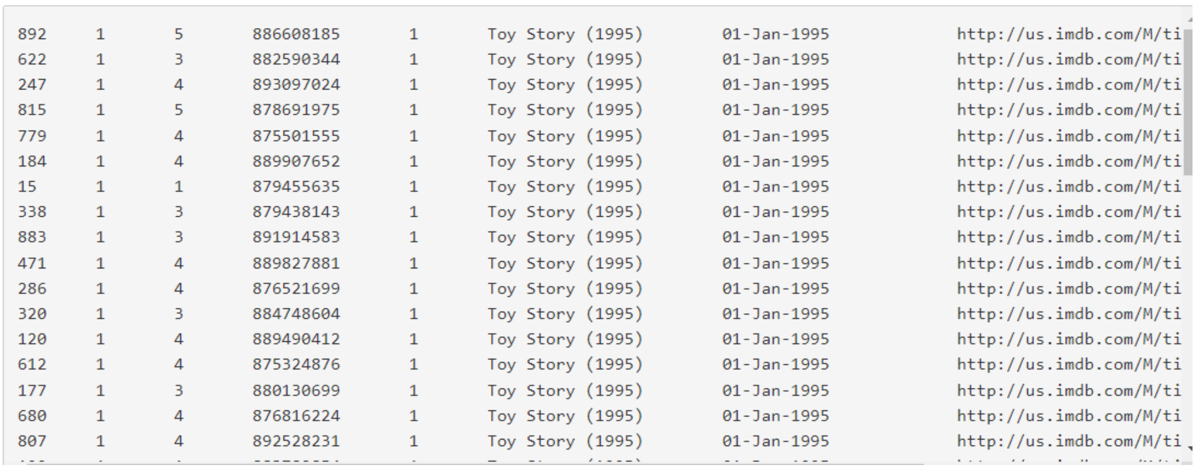
```
grunt> ratings_with_titles = JOIN ratings BY movieID, metadata BY movieID;
grunt> STORE ratings_with_titles INTO '/Project_Dataset/ratings_with_titles.txt';
2023-09-19 01:47:49,512 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: HASH_JOIN
2023-09-19 01:47:49,544 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2023-09-19 01:47:49,544 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator,
GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatte
n, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2023-09-19 01:47:49,545 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2023-09-19 01:47:49,546 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler$LastInputStreamingOptimizer - Rewrite: POPackage->POForEach
 to POPackage(JoinPackager)
2023-09-19 01:47:49,546 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2023-09-19 01:47:49,546 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2023-09-19 01:47:49,629 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/tim
eline/
2023-09-19 01:47:49,629 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbox.hortonworks.com/172.17.0.2:8050
2023-09-19 01:47:49,630 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History server at sandbox.hortonworks.com/172.17.0.2:10200
2023-09-19 01:47:49,637 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2023-09-19 01:47:49,638 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not
 set, set to default 0.3
2023-09-19 01:47:49,638 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Reduce phase detected, estimating # of required r
educers.
2023-09-19 01:47:49,638 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Using reducer estimator: org.apache.pig.backend.h
adoop.executionengine.mapReduceLayer.InputSizeReducerEstimator
2023-09-19 01:47:49,642 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator - BytesPerReducer=1000000000 maxReducers=999
 totalInputFileSize=2215517
2023-09-19 01:47:49,642 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting Parallelism to 1
2023-09-19 01:47:49,642 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - This job cannot be converted run in-process
2023-09-19 01:47:49,774 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Added jar file:/usr/hdp/2.5.0.0-1245/pig/pig-0.16
.0.2.5.0.0-1245-core-h2.jar to DistributedCache through /tmp/temp-1280146786/tmp2100227063/pig-0.16.0.2.5.0.0-1245-core-h2.jar
2023-09-19 01:47:49,832 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Added jar file:/usr/hdp/2.5.0.0-1245/pig/lib/auto
maton-1.11-8.jar to DistributedCache through /tmp/temp-1280146786/tmp291009866/automaton-1.11-8.jar
2023-09-19 01:47:49,916 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Added jar file:/usr/hdp/2.5.0.0-1245/pig/lib/antl
r-runtime-3.4.jar to DistributedCache through /tmp/temp-1280146786/tmp1701875238/antlr-runtime-3.4.jar
2023-09-19 01:47:50,400 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Added jar file:/usr/hdp/2.5.0.0-1245/hadoop/lib/j
oda-time-2.8.1.jar to DistributedCache through /tmp/temp-1280146786/tmp-969661454/joda-time-2.8.1.jar
2023-09-19 01:47:50,401 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2023-09-19 01:47:50,401 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2023-09-19 01:47:50,401 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cacche
2023-09-19 01:47:50,401 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Setting key [pig.schematuple.classes] with classes to deserialize []
2023-09-19 01:47:50,409 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
2023-09-19 01:47:50,467 [JobControl] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://sandbox.hortonworks.com:8188/ws/
```

Figure 7 Query

<u>Output:</u>



```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.7.3.2.5.0.0-1245      0.16.0.2.5.0.0-1245     root    2023-09-19 01:47:49     2023-09-19 01:48:17     HASH_JOIN

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   F
eature  Outputs
job_1694535283099_0062  2       1       5       5       5       5       5       5       5       5       metadata,ratings,ratings_with_titles    HASH_JOIN       /Project
_Dataset/ratings_with_titles.txt,

Input(s):
Successfully read 1682 records from: "/Project_Dataset/u.item"
Successfully read 100000 records from: "/Project_Dataset/u.data"

Output(s):
Successfully stored 100000 records (12163405 bytes) in: "/Project_Dataset/ratings_with_titles.txt"

Counters:
Total records written : 100000
Total bytes written : 12163405
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1694535283099_0062
```



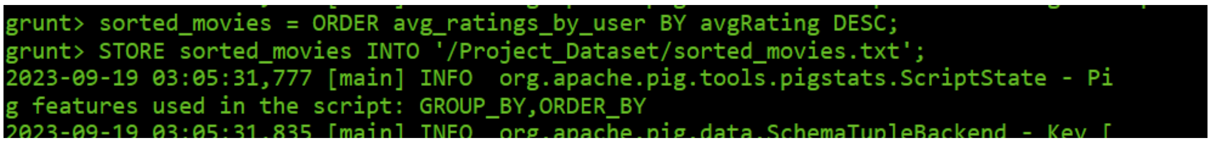
```
892    1    5    886608185    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
622    1    3    882590344    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
247    1    4    893097024    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
815    1    5    878691975    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
779    1    4    875501555    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
184    1    4    889907652    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
15     1    1    879455635    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
338    1    3    879438143    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
883    1    3    891914583    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
471    1    4    889827881    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
286    1    4    876521699    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
320    1    3    884748604    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
120    1    4    889490412    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
612    1    4    875324876    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
177    1    3    880130699    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
680    1    4    876816224    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
807    1    4    892528231    1    Toy Story (1995)    01-Jan-1995    http://us.imdb.com/M/ti
```

2.2.4 **Sorting:** You can sort the data based on certain fields. Sorting the movies based on their average ratings.

Script:

sorted_movies = ORDER avg_ratings_by_user BY avgRating DESC;

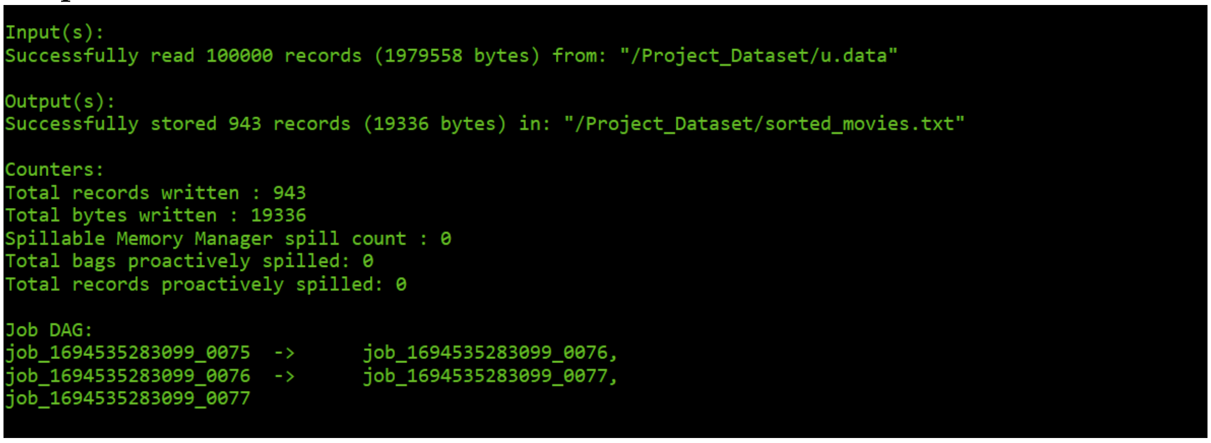
STORE sorted_movies INTO '/Project_Dataset/sorted_movies.txt';



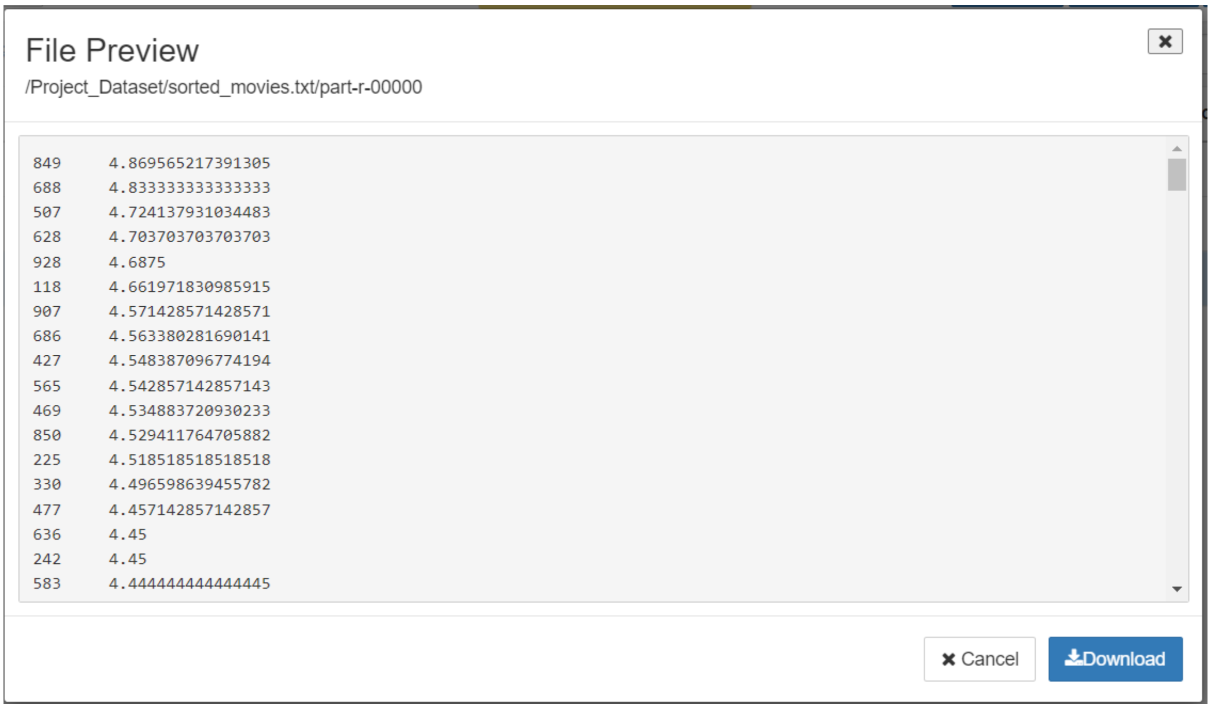
```
grunt> sorted_movies = ORDER avg_ratings_by_user BY avgRating DESC;
grunt> STORE sorted_movies INTO '/Project_Dataset/sorted_movies.txt';
2023-09-19 03:05:31,777 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pi
g features used in the script: GROUP_BY,ORDER_BY
2023-09-19 03:05:31,835 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [
```

Figure 8 Query

<u>Output:</u>



```
Input(s):
Successfully read 100000 records (1979558 bytes) from: "/Project_Dataset/u.data"

Output(s):
Successfully stored 943 records (19336 bytes) in: "/Project_Dataset/sorted_movies.txt"

Counters:
Total records written : 943
Total bytes written : 19336
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1694535283099_0075  ->      job_1694535283099_0076,
job_1694535283099_0076  ->      job_1694535283099_0077,
job_1694535283099_0077
```



File Preview
/Project_Dataset/sorted_movies.txt/part-r-00000

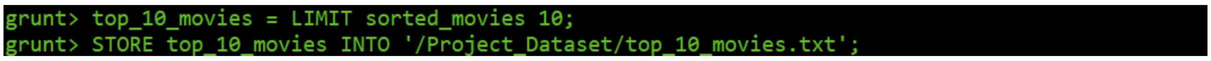
```
849     4.869565217391305
688     4.833333333333333
507     4.724137931034483
628     4.703703703703703
928     4.6875
118     4.661971830985915
907     4.571428571428571
686     4.563380281690141
427     4.548387096774194
565     4.542857142857143
469     4.534883720930233
850     4.529411764705882
225     4.518518518518518
330     4.496598639455782
477     4.457142857142857
636     4.45
242     4.45
583     4.444444444444445
```

2.2.5 **Limiting**: You can limit the number of tuples in a relation. Can get the top 10 movies with the highest average ratings.

Script:

top_10_movies = LIMIT sorted_movies 10;

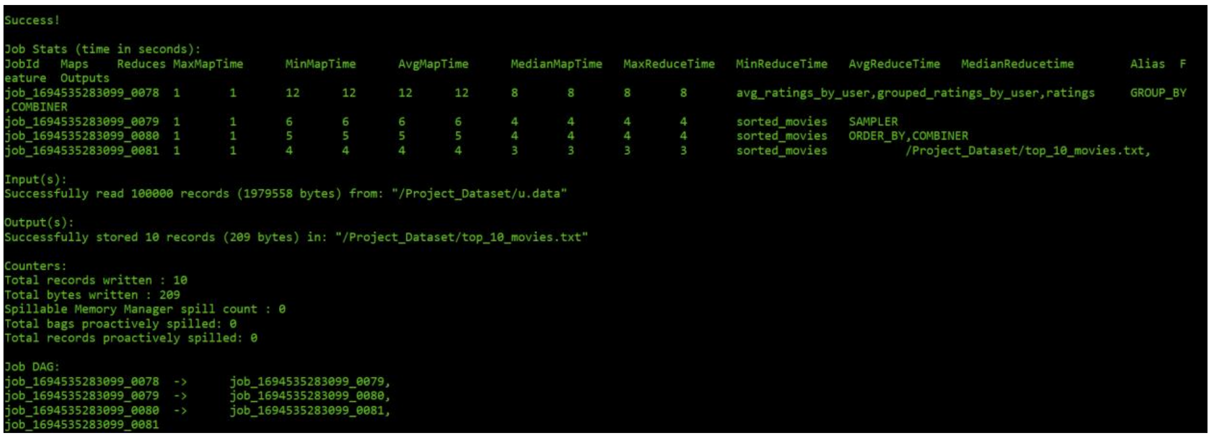
STORE top_10_movies INTO '/user/maria_dev/top_10_movies.txt';



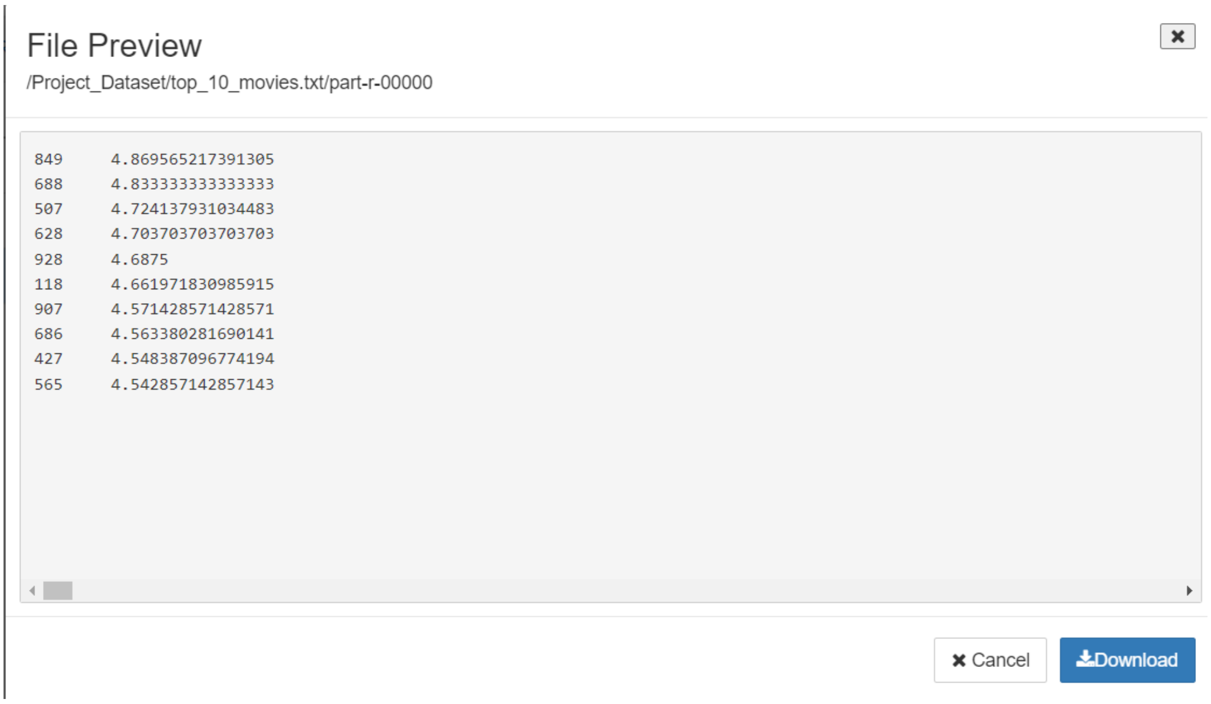
```
grunt> top_10_movies = LIMIT sorted_movies 10;
grunt> STORE top_10_movies INTO '/Project_Dataset/top_10_movies.txt';
```

Figure 9 Query

<u>Output:</u>



```
Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   F
eature  Outputs
job_1694535283099_0078  1       1       12      12      12      12      8       8       8       8       avg_ratings_by_user,grouped_ratings_by_user,ratings     GROUP_BY
,COMBINER
job_1694535283099_0079  1       1       6       6       6       6       4       4       4       4       sorted_movies   SAMPLER
job_1694535283099_0080  1       1       5       5       5       5       4       4       4       4       sorted_movies   ORDER_BY,COMBINER
job_1694535283099_0081  1       1       4       4       4       4       3       3       3       3       sorted_movies           /Project_Dataset/top_10_movies.txt,

Input(s):
Successfully read 100000 records (1979558 bytes) from: "/Project_Dataset/u.data"

Output(s):
Successfully stored 10 records (209 bytes) in: "/Project_Dataset/top_10_movies.txt"

Counters:
Total records written : 10
Total bytes written : 209
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1694535283099_0078  ->      job_1694535283099_0079,
job_1694535283099_0079  ->      job_1694535283099_0080,
job_1694535283099_0080  ->      job_1694535283099_0081,
job_1694535283099_0081
```



/Project_Dataset/top_10_movies.txt/part-r-00000

```
849     4.869565217391305
688     4.833333333333333
507     4.724137931034483
628     4.703703703703703
928     4.6875
118     4.661971830985915
907     4.571428571428571
686     4.563380281690141
427     4.548387096774194
565     4.542857142857143
```

3. Concluding Remark

In this project, we embarked on a comprehensive exploration of movie rating datasets, employing Apache Pig to perform various data processing tasks. The investigation involved a two-fold analysis focusing on identifying poorly rated movies and understanding user preferences through average ratings. Leveraging Pig Latin, a high-level scripting language for Apache Pig, enabled efficient data processing and analysis.

The project initiated with data loading and extraction of relevant information. The dataset was loaded, and pertinent fields such as userID, movieID, rating, and ratingTime were extracted. Subsequently, grouping and aggregation operations were executed to discern patterns in user behavior. Grouping the ratings data by userID facilitated the computation of average ratings for each user. The results were stored for further analysis, contributing insights into user preferences and aiding in the personalization of content recommendations. Further enriching our analysis, the project delved into the realm of data integration through joining. By merging the ratings data with metadata, which included movie titles, a more comprehensive dataset was obtained.

Sorting emerged as a crucial facet of our exploration. The sorted dataset, ordered by average ratings in descending order, unveiled the highest-rated movies. This process facilitated the identification of movies that resonated most positively with the audience, offering valuable insights into popular content.

Lastly, the project employed limiting operations to distill the wealth of information into manageable, actionable insights. By focusing on the top 10 movies with the highest average ratings, the research honed in on the most impactful findings. This selective approach is instrumental for content recommendation systems, aiding in the curation of personalized lists for users.

In conclusion, the research project demonstrated the power of Apache Pig in efficiently processing and analyzing large-scale datasets. The methodologies employed, including grouping, joining, sorting, and limiting, showcased the versatility of Pig Latin in handling diverse tasks. The outcomes of this study provide a foundation for enhancing user experiences in content recommendation systems, ensuring that users are presented with movies that align with their preferences, thereby maximizing user satisfaction. The methodologies and insights gained from this research lay a robust groundwork for future endeavors in the field of data processing and analysis.

4. Future Work

- **Sentiment Analysis:** Going beyond numerical ratings, future work could involve sentiment analysis on user reviews. Understanding the qualitative aspects of why users liked or disliked a movie could provide deeper insights into audience preferences and contribute to a more nuanced recommendation system.
- Can I also work on different sectors like Coursera reviews, Flipkart reviews and Google Play Store reviews

5. References

1. Jain, A., Bhatnagar, V., Chen, M., Mao, S., Liu, Y., Frings, S., ... Santhi, V. (2016). Movie Analytics forEffective Recommendation System using Pig with Hadoop. International Journal of Rough Sets andData Analysis, 3(2), 82–100. doi:10.4018/IJRSDA.2016040106

2. Lin, J., & Dyer, C. (2010). Data-Intensive Text Processing with MapReduce. Synthesis Lectures on Human Language Technologies, 3(1), 1–177. DOI: 10.2200/S00218ED1V01Y201003HLT007

3. Apache Pig Documentation: The official documentation provides detailed information about Apache Pig, its functionalities, and how to use it. You can find it on the Apache Pig website: https://pig.apache.org/docs/r0.17.0/

4.https://www.guru99.com/introduction-to-pig-and-hive.html
5. [Tom White], 2009. Hadoop: The Definitive Guide. First Edition. O'Reilly Media.

6. [Alan Gates], 2011.Programming Pig. First Edition. O'Reilly Media.