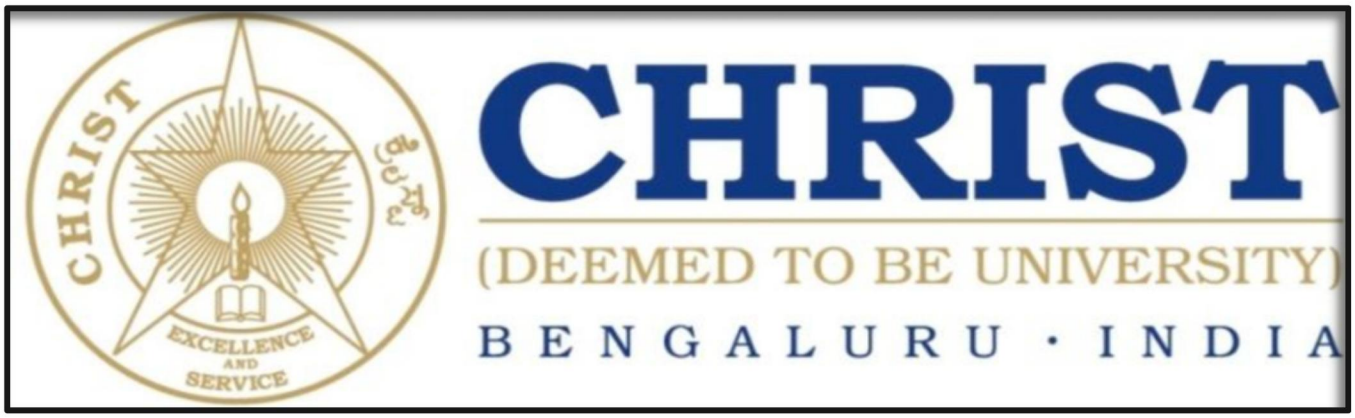


ANALYSIS OF MOVIE DATASET: TRENDS, DIRECTORS, RATINGS, AND LANGUAGE INSIGHTS

(DEMONSTRATION OF HADOOP STREAMING)

SANJAY R

2348055

26th November 2024

MDS571

Big Data Analytics

Department of Statistics and Data Science

CHRIST (Deemed to be University)

Department of Statistics and Data Science

Course:MDS571-Big Data Analytics **Lab :6**

R.SANJAY **2348055**

ANALYSIS OF MOVIE DATASET: TRENDS, DIRECTORS, RATINGS, AND LANGUAGE INSIGHTS

DEMONSTRATION OF HADOOP STREAMING

✧ INTRODUCTION:

In today's data-driven world, the ability to process and analyze vast amounts of information has become indispensable. Big data technologies have revolutionized how industries manage and extract insights from complex datasets, enabling informed decision-making and strategic planning. Among these technologies, Hadoop stands out as a powerful framework for distributed and scalable data processing, making it an essential tool for handling large datasets efficiently.

The film industry, rich in diverse data such as movie titles, directors, ratings, and languages, provides a compelling example of big data's potential to uncover patterns and trends. By analyzing this data, businesses and researchers can gain insights into audience preferences, production trends, and global market dynamics. This report explores how Hadoop's MapReduce framework can be utilized to analyze movie datasets effectively.Through the use of Python-based mapper and reducer programs, this project demonstrates how Hadoop streaming facilitates scalable data processing. The integration of distributed computing principles with real-world datasets underscores the practical value of big data tools in addressing complex analytical

challenges. This report aims to provide a comprehensive understanding of how modern technologies can unlock the full potential of data.

✧ PROBLEM DESCRIPTION:

The primary challenge addressed by this project is the efficient filtering and aggregation of a large movie dataset. In the context of the film industry, the ability to extract specific information about movies, such as details about the director, release year, and audience ratings, is crucial for various analytical purposes.

The project seeks to address two main problems:

- ✓ **Filtering Movies Based on Criteria:** Given a large dataset of movies, we need to filter movies based on specific criteria such as the director's name, the year of release, and a minimum rating threshold.
- ✓ **Language-wise Aggregation and Counting:** After filtering the movies based on the specified criteria, the next step is to categorize the movies by language and count the number of movies for each language. This will provide an overview of how many movies are available in each language, which can be useful for understanding language distribution and trends in the film industry. Additionally, each movie's details, including its title, year, rating, and director, are output for further analysis.

✧ DATASET DESCRIPTION:

The provided dataset contains a collection of movies with various details about each film. It consists of six key attributes: movie ID, title, release year, rating, director, and language. This structured dataset offers valuable insights into the movie industry, and each record is separated by tabs, making it suitable for large-scale processing, such as with Hadoop or MapReduce frameworks.

- ➢ **Movie ID:** A unique identifier assigned to each movie in the dataset. It serves as a primary key, ensuring that every movie record is distinct and can be referenced individually in analyses.
- ➢ **Title:** The name of the movie.
- ➢ **Release Year:** The year in which the movie was released.

- **Rating:** The average rating of the movie, usually based on audience feedback.
- **Director:** The person who directed the movie.
- **Language:** The primary language in which the movie was made. This feature is can be used for regional analysis or to study language-based trends in the film industry.

✧ CODE:

✓ mapper.py:

```
import sys
# Check if correct arguments are passed from Hadoop Streaming command
if len(sys.argv) != 4:
    print("Usage: python mapper.py <Director Name> <Year> <Rating>")
    sys.exit(1)
# Get dynamic inputs from command-line arguments passed by Hadoop Streaming
director_filter = sys.argv[1]    # Director filter (from user input)
year_filter = sys.argv[2]    # Year filter (from user input)
rating_filter = float(sys.argv[3])    # Rating filter (from user input)
# Process each line from the input dataset
for line in sys.stdin:
    line = line.strip()    # Remove leading and trailing whitespaces
    if not line:    # Skip empty lines
        continue
    # Split the line into columns assuming tab-separated values
    columns = line.split("\t")
    if len(columns) < 6:    # Ensure the line has sufficient columns
        continue
    # Extracting relevant columns from the movie dataset
    movie_id = columns[0]
    movie_name = columns[1]
    year = columns[2]
    rating = float(columns[3])
    director = columns[4]
    language = columns[5]
```

 # Filter the data based on the director, year, and rating provided by the user

if director == director_filter and year == year_filter and rating >= rating_filter:

 # If the movie matches the filter criteria, output it as 'filtered' data

print(f"filtered\t{director}\t{year}\t{rating}\t{movie_name}\t{language}")

 # Output language-wise movie count (this is used by the Reducer for aggregation)

print(f"{language}\t1\t{movie_name}\t{year}\t{rating}\t{director}")

✓ reducer.py:

import sys

from collections import defaultdict

 # Create a defaultdict to accumulate counts and movie details

language_count = defaultdict(lambda: {'count': 0, 'movies': []})

 # Read input line by line from the Mapper's output

for line in sys.stdin:

line = line.strip() # Strip leading/trailing whitespaces

 # Split the input line into parts based on tab delimiter

parts = line.split("\t")

 # If the record is a "filtered" movie (based on the user-defined filters)

if parts[0] == "filtered":

director = parts[1]

year = parts[2]

rating = float(parts[3])

movie_name = parts[4]

language = parts[5]

 # Output the filtered movie details

```
        print(f"Filtered Movie: {movie_name}, {year}, {rating}, {director}, {language}")

    else:
        # Handle the language-wise movie counting (non-filtered records)
        language = parts[0]
        count = int(parts[1])  # This is always 1 for each movie
        movie_name = parts[2]
        year = parts[3]
        rating = float(parts[4])
        director = parts[5]

        # Update the language count and store movie details for each language
        language_count[language]['count'] += count
        language_count[language]['movies'].append(f"{movie_name} ({year}) - {rating} by {director}")

# Print the language-wise aggregation results
for language, data in language_count.items():
    print(f"Language: {language}, Movie Count: {data['count']}")
    for movie in data['movies']:
        print(f"  {movie}")

# Summarize the total movie count per language at the end
print("\nSummary:")
for language, data in language_count.items():
    print(f"Language: {language}, Total Movies: {data['count']}")
```

✧ PROGRAM DESCRIPTION:

The Movie Dataset Analysis Program uses the Hadoop MapReduce framework to process a large movie dataset. The primary objective of the program is to aggregate the number of movies based on their rating, director, and language, leveraging the distributed processing capabilities of Hadoop. By using MapReduce, the program

efficiently handles the large dataset, ensuring scalability and high performance in a distributed environment. The program's architecture consists of two main components: the Mapper and the Reducer, which work together to process the data and produce aggregated insights.

✓ mapper.py:

The Mapper is designed to filter and prepare the data by extracting the relevant information from the movie dataset and applying dynamic filters based on the user-provided inputs for director, year, and rating. These filter values are passed via the Hadoop Streaming command using sys.argv[1], sys.argv[2], and sys.argv[3]. The Mapper reads the movie dataset line by line and splits each line into its respective columns, which include movie_name, year, rating, director, and language. The Mapper checks if each movie record matches the filter conditions and then outputs the filtered data and language-wise counts.

The filter logic is implemented as follows:

```
if director == director_filter and year == year_filter and rating >= rating_filter:
    print(f"filtered\t{director}\t{year}\t{rating}\t{movie_name}\t{language}")
```

If a movie matches the provided director, year, and rating conditions, it is sent to the Reducer as a "filtered" record. Additionally, the Mapper also outputs language-wise movie counts for aggregation in the Reducer:

```
print(f"{language}\t1\t{movie_name}\t{year}\t{rating}\t{director}")
```

This ensures that the dataset is filtered based on the input criteria while also providing a breakdown of movie counts by language.

✓ reducer.py:

Reducer Logic

The Reducer receives the filtered data and language-wise counts from the Mapper and performs the aggregation tasks. It uses a default dictionary to store the count of movies and their details for each language. The Reducer begins by processing the

filtered records sent from the Mapper. It checks if the incoming record is a filtered movie using the condition parts[0] == "filtered", and then outputs the details of the filtered movie:

```
if parts[0] == "filtered":
    director = parts[1]
    year = parts[2]
    rating = float(parts[3])
    movie_name = parts[4]
    language = parts[5]
    print(f"Filtered Movie: {movie_name}, {year}, {rating}, {director}, {language}")
```

For language-wise movie counts, the Reducer aggregates the data by updating the movie count for each language and storing the movie details in a list. The aggregation logic is as follows:

```
language_count[language]['count'] += count
language_count[language]['movies'].append(f"{movie_name} ({year}) - {rating} by {director}")
```

At the end of the process, the Reducer prints the aggregated results for each language, showing the total count of movies and listing the movies for each language:

```
for language, data in language_count.items():
    print(f"Language: {language}, Movie Count: {data['count']}")
    for movie in data['movies']:
        print(f"    {movie}")
```

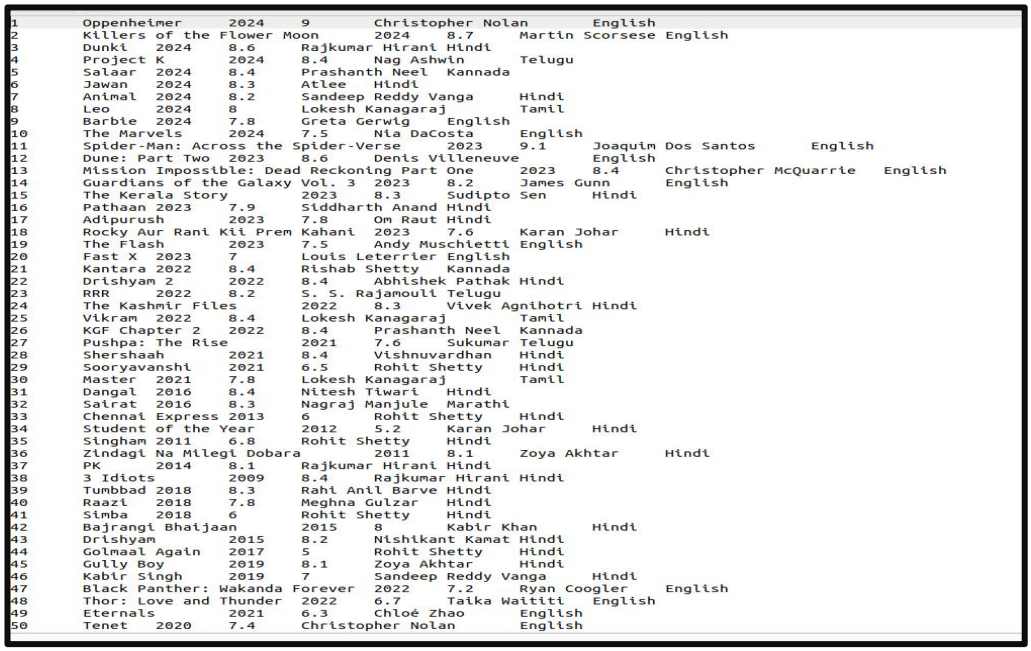
This step ensures that the Reducer outputs the final summary of the dataset, providing both the filtered movies based on user-defined criteria and the aggregated count of movies per language. The Mapper and Reducer work together to achieve the program's objective of categorizing and counting movies efficiently in a distributed Hadoop environment.

✧ PROJECT SETUP AND EXECUTION:

✓ DATASET PREPARATION:

The Movie Dataset Analysis Program relies on a single primary dataset containing movie-related information.

- **Movies Dataset:** This dataset includes key attributes such as movie_id, title, release year, rating, director, and language. Each record in the dataset provides essential details that are crucial for understanding various aspects of the movies, such as their ratings, the directors who created them, and the languages in which they were produced.



```
1       Oppenheimer     2024    9       Christopher Nolan       English
2       Killers of the Flower Moon      2024    8.7     Martin Scorsese English
3       Dunki   2024    8.6     Rajkumar Hirani Hindi
4       Project K       2024    8.4     Nag Ashwin      Telugu
5       Salaar  2024    8.4     Prashanth Neel  Kannada
6       Jawan   2024    8.3     Atlee   Hindi
7       Animal  2024    8.2     Sandeep Reddy Vanga     Hindi
8       Leo     2024    8       Lokesh Kanagaraj        Tamil
9       Barbie  2024    7.8     Greta Gerwig    English
10      The Marvels     2024    7.5     Nia DaCosta     English
11      Spider-Man: Across the Spider-Verse     2023    9.1     Joaquim Dos Santos      English
12      Dune: Part Two  2023    8.6     Denis Villeneuve        English
13      Mission Impossible: Dead Reckoning Part One     2023    8.4     Christopher McQuarrie   English
14      Guardians of the Galaxy Vol. 3  2023    8.2     James Gunn      English
15      The Kerala Story        2023    8.3     Sudipto Sen     Hindi
16      Pathaan 2023    7.9     Siddharth Anand Hindi
17      Adipurush       2023    7.8     Om Raut Hindi
18      Rocky Aur Rani Kii Prem Kahani  2023    7.6     Karan Johar     Hindi
19      The Flash       2023    7.5     Andy Muschietti English
20      Fast X  2023    7       Louis Leterrier English
21      Kantara 2022    8.4     Rishab Shetty   Kannada
22      Drishyam 2      2022    8.4     Abhishek Pathak Hindi
23      RRR     2022    8.2     S. S. Rajamouli Telugu
24      The Kashmir Files       2022    8.3     Vivek Agnihotri Hindi
25      Vikram  2022    8.4     Lokesh Kanagaraj        Tamil
26      KGF Chapter 2   2022    8.4     Prashanth Neel  Kannada
27      Pushpa: The Rise        2021    7.6     Sukumar Telugu
28      Shershaah       2021    8.4     Vishnuvardhan   Hindi
29      Sooryavanshi    2021    6.5     Rohit Shetty    Hindi
30      Master  2021    7.8     Lokesh Kanagaraj        Tamil
31      Dangal  2016    8.4     Nitesh Tiwari   Hindi
32      Sairat  2016    8.3     Nagraj Manjule  Marathi
33      Chennai Express 2013    6       Rohit Shetty    Hindi
34      Student of the Year     2012    5.2     Karan Johar     Hindi
35      Singham 2011    6.8     Rohit Shetty    Hindi
36      Zindagi Na Milegi Dobara        2011    8.1     Zoya Akhtar     Hindi
37      PK      2014    8.1     Rajkumar Hirani Hindi
38      3 Idiots        2009    8.4     Rajkumar Hirani Hindi
39      Tumbbad 2018    8.3     Rahi Anil Barve Hindi
40      Raazi   2018    7.8     Meghna Gulzar   Hindi
41      Simba   2018    6       Rohit Shetty    Hindi
42      Bajrangi Bhaijaan       2015    8       Kabir Khan      Hindi
43      Drishyam        2015    8.2     Nishikant Kamat Hindi
44      Golmaal Again   2017    5       Rohit Shetty    Hindi
45      Gully Boy       2019    8.1     Zoya Akhtar     Hindi
46      Kabir Singh     2019    7       Sandeep Reddy Vanga     Hindi
47      Black Panther: Wakanda Forever  2022    7.2     Ryan Coogler    English
48      Thor: Love and Thunder  2022    6.7     Taika Waititi   English
49      Eternals        2021    6.3     Chloé Zhao      English
50      Tenet   2020    7.4     Christopher Nolan       English
```

The dataset is structured in a tab-separated format, which allows for efficient parsing and processing by the MapReduce framework. The movie_id serves as a unique identifier for each movie, while other attributes like rating, director, and language allow for in-depth analysis and aggregation to uncover trends and insights from the movie data.

To facilitate processing with the Hadoop MapReduce framework, the datasets were placed within the Hadoop directory structure at /MDS2024/LAB.6 This setup ensured that the files were easily accessible for the program, allowing seamless integration and efficient execution of the MapReduce process.

```
hadoop@Ubuntu22:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Ubuntu22]
hadoop@Ubuntu22:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@Ubuntu22:~$ jps
6433 ResourceManager
5957 DataNode
6184 SecondaryNameNode
7209 Jps
6555 NodeManager
5837 NameNode
```

```
hadoop@Ubuntu22:~$ hadoop fs -ls  /MDS2024/LAB6/
Found 1 items
-rw-r--r--   3 hadoop supergroup       2300 2024-12-12 23:12 /MDS2024/LAB6/Movies
```

✧ EXECUTION PROCESS:

- The mapper.py and reducer.py files were created to perform the core tasks of processing and joining the movie dataset based on the movie_id. These files were saved in the specified file paths for execution within the Hadoop environment.
- The mapper.py file was stored at /home/hadoop/Labs/LAB6/mapper.py, and the reducer.py file was saved at /home/hadoop/Labs/LAB6/reducer.py.
- The next step involved executing the program within the Hadoop environment, using the following command:

~$ hadoop jar '/home/hadoop/Desktop/hadoop-streaming-3.3.6.jar' -mapper "python3 /home/hadoop/Labs/LAB6/mapper.py 'Sudipto Sen' '2023' '8.3'" -reducer "python3 /home/hadoop/Labs/LAB6/reducer.py" -input '/MDS2024/LAB6/Movies' -output '/MDS2024/LAB6/output'

```
hadoop@Ubuntu22:~$ hadoop jar '/home/hadoop/Desktop/hadoop-streaming-3.3.6.jar' -mapper "python3 /home/hadoop/Labs/LAB6/mapper.py 'Sudipto Sen' '2023' '8.3'"
 -reducer "python3 /home/hadoop/Labs/LAB6/reducer.py" -input '/MDS2024/LAB6/Movies' -output '/MDS2024/LAB6/output'
2024-12-13 23:41:36,587 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-12-13 23:41:36,811 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-12-13 23:41:36,811 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-12-13 23:41:36,843 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-12-13 23:41:37,524 INFO mapred.FileInputFormat: Total input files to process : 1
2024-12-13 23:41:37,623 INFO mapreduce.JobSubmitter: number of splits:1
2024-12-13 23:41:38,046 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1035403188_0001
2024-12-13 23:41:38,046 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-12-13 23:41:38,350 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-12-13 23:41:38,392 INFO mapreduce.Job: Running job: job_local1035403188_0001
2024-12-13 23:41:38,451 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-12-13 23:41:38,470 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-12-13 23:41:38,490 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-12-13 23:41:38,490 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup fai
lures: false
2024-12-13 23:41:38,593 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-12-13 23:41:38,606 INFO mapred.LocalJobRunner: Starting task: attempt_local1035403188_0001_m_000000_0
2024-12-13 23:41:38,677 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-12-13 23:41:38,681 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup fai
lures: false
2024-12-13 23:41:38,816 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-12-13 23:41:38,852 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/MDS2024/LAB6/Movies:0+2300
2024-12-13 23:41:38,914 INFO mapred.MapTask: numReduceTasks: 1
2024-12-13 23:41:39,112 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-12-13 23:41:39,114 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-12-13 23:41:39,115 INFO mapred.MapTask: soft limit at 83886080
2024-12-13 23:41:39,115 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-12-13 23:41:39,115 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-12-13 23:41:39,123 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
```

```
        File System Counters
                FILE: Number of bytes read=287778
                FILE: Number of bytes written=1586085
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4600
                HDFS: Number of bytes written=2832
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=51
                Map output records=51
                Map output bytes=2327
                Map output materialized bytes=2435
                Input split bytes=93
                Combine input records=0
                Combine output records=0
                Reduce input groups=7
                Reduce shuffle bytes=2435
                Reduce input records=51
                Reduce output records=65
                Spilled Records=102
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=0
                Total committed heap usage (bytes)=679477248
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2300
        File Output Format Counters
                Bytes Written=2832
2024-12-13 23:41:41,588 INFO streaming.StreamJob: Output directory: /MDS2024/LAB6/output
```

~$ hadoop fs -ls /MDS2024/LAB6/output

```
hadoop@Ubuntu22:~$ hadoop fs -ls  /MDS2024/LAB6/output
Found 2 items
-rw-r--r--   3 hadoop supergroup          0 2024-12-13 23:41 /MDS2024/LAB6/output/_SUCCESS
-rw-r--r--   3 hadoop supergroup       2832 2024-12-13 23:41 /MDS2024/LAB6/output/part-00000
```

~$ hadoop fs -cat /MDS2024/LAB6/output/part-00000

```
hadoop@Ubuntu22:~$ hadoop fs -cat /MDS2024/LAB6/output/part-00000
Filtered Movie: The Kerala Story, 2023, 8.3, Sudipto Sen, Hindi
Language: English, Movie Count: 14
  Oppenheimer (2024) - 9.0 by Christopher Nolan
  Eternals (2021) - 6.3 by Chloé Zhao
  Thor: Love and Thunder (2022) - 6.7 by Taika Waititi
  Black Panther: Wakanda Forever (2022) - 7.2 by Ryan Coogler
  Tenet (2020) - 7.4 by Christopher Nolan
  Fast X (2023) - 7.0 by Louis Leterrier
  The Flash (2023) - 7.5 by Andy Muschietti
  Guardians of the Galaxy Vol. 3 (2023) - 8.2 by James Gunn
  Mission Impossible: Dead Reckoning Part One (2023) - 8.4 by Christopher McQuarrie
  Dune: Part Two (2023) - 8.6 by Denis Villeneuve
  Spider-Man: Across the Spider-Verse (2023) - 9.1 by Joaquim Dos Santos
  The Marvels (2024) - 7.5 by Nia DaCosta
  Barbie (2024) - 7.8 by Greta Gerwig
  Killers of the Flower Moon (2024) - 8.7 by Martin Scorsese
Language: Hindi, Movie Count: 26
  Zindagi Na Milegi Dobara (2011) - 8.1 by Zoya Akhtar
  Singham (2011) - 6.8 by Rohit Shetty
  Student of the Year (2012) - 5.2 by Karan Johar
  Chennai Express (2013) - 6.0 by Rohit Shetty
  Dangal (2016) - 8.4 by Nitesh Tiwari
  Sooryavanshi (2021) - 6.5 by Rohit Shetty
  Shershaah (2021) - 8.4 by Vishnuvardhan
  Dunki (2024) - 8.6 by Rajkumar Hirani
  The Kashmir Files (2022) - 8.3 by Vivek Agnihotri
  Drishyam 2 (2022) - 8.4 by Abhishek Pathak
  Rocky Aur Rani Kii Prem Kahani (2023) - 7.6 by Karan Johar
  Adipurush (2023) - 7.8 by Om Raut
  Pathaan (2023) - 7.9 by Siddharth Anand
  The Kerala Story (2023) - 8.3 by Sudipto Sen
  Animal (2024) - 8.2 by Sandeep Reddy Vanga
  Jawan (2024) - 8.3 by Atlee
  Kabir Singh (2019) - 7.0 by Sandeep Reddy Vanga
  Gully Boy (2019) - 8.1 by Zoya Akhtar
  Golmaal Again (2017) - 5.0 by Rohit Shetty
  Drishyam (2015) - 8.2 by Nishikant Kamat
  Bajrangi Bhaijaan (2015) - 8.0 by Kabir Khan
  Simba (2018) - 6.0 by Rohit Shetty
  Raazi (2018) - 7.8 by Meghna Gulzar
  Tumbbad (2018) - 8.3 by Rahi Anil Barve
  3 Idiots (2009) - 8.4 by Rajkumar Hirani
  PK (2014) - 8.1 by Rajkumar Hirani
```

✧ **OUTPUT:**

Filtered Movie: The Kerala Story, 2023, 8.3, Sudipto Sen, Hindi

Language: English, Movie Count: 14

Oppenheimer (2024) - 9.0 by Christopher Nolan

Eternals (2021) - 6.3 by Chloé Zhao

Thor: Love and Thunder (2022) - 6.7 by Taika Waititi

Black Panther: Wakanda Forever (2022) - 7.2 by Ryan Coogler

Tenet (2020) - 7.4 by Christopher Nolan

Fast X (2023) - 7.0 by Louis Leterrier

The Flash (2023) - 7.5 by Andy Muschietti

Guardians of the Galaxy Vol. 3 (2023) - 8.2 by James Gunn

Mission Impossible: Dead Reckoning Part One (2023) - 8.4 by Christopher McQuarrie

Dune: Part Two (2023) - 8.6 by Denis Villeneuve

Spider-Man: Across the Spider-Verse (2023) - 9.1 by Joaquim Dos Santos

The Marvels (2024) - 7.5 by Nia DaCosta

Barbie (2024) - 7.8 by Greta Gerwig

Killers of the Flower Moon (2024) - 8.7 by Martin Scorsese

Language: Hindi, Movie Count: 26

Zindagi Na Milegi Dobara (2011) - 8.1 by Zoya Akhtar

Singham (2011) - 6.8 by Rohit Shetty

Student of the Year (2012) - 5.2 by Karan Johar

Chennai Express (2013) - 6.0 by Rohit Shetty

Dangal (2016) - 8.4 by Nitesh Tiwari

Sooryavanshi (2021) - 6.5 by Rohit Shetty

Shershaah (2021) - 8.4 by Vishnuvardhan

Dunki (2024) - 8.6 by Rajkumar Hirani

The Kashmir Files (2022) - 8.3 by Vivek Agnihotri

Drishyam 2 (2022) - 8.4 by Abhishek Pathak

Rocky Aur Rani Kii Prem Kahani (2023) - 7.6 by Karan Johar

Adipurush (2023) - 7.8 by Om Raut

Pathaan (2023) - 7.9 by Siddharth Anand

The Kerala Story (2023) - 8.3 by Sudipto Sen

Animal (2024) - 8.2 by Sandeep Reddy Vanga

Jawan (2024) - 8.3 by Atlee

Kabir Singh (2019) - 7.0 by Sandeep Reddy Vanga

Gully Boy (2019) - 8.1 by Zoya Akhtar

Golmaal Again (2017) - 5.0 by Rohit Shetty

Drishyam (2015) - 8.2 by Nishikant Kamat

Bajrangi Bhaijaan (2015) - 8.0 by Kabir Khan

Simba (2018) - 6.0 by Rohit Shetty

Raazi (2018) - 7.8 by Meghna Gulzar

Tumbbad (2018) - 8.3 by Rahi Anil Barve

3 Idiots (2009) - 8.4 by Rajkumar Hirani

PK (2014) - 8.1 by Rajkumar Hirani

Language: Kannada, Movie Count: 3

Kantara (2022) - 8.4 by Rishab Shetty

KGF Chapter 2 (2022) - 8.4 by Prashanth Neel

Salaar (2024) - 8.4 by Prashanth Neel

Language: Marathi, Movie Count: 1

Sairat (2016) - 8.3 by Nagraj Manjule

Language: Tamil, Movie Count: 3

Master (2021) - 7.8 by Lokesh Kanagaraj

Vikram (2022) - 8.4 by Lokesh Kanagaraj

Leo (2024) - 8.0 by Lokesh Kanagaraj

Language: Telugu, Movie Count: 3

Pushpa: The Rise (2021) - 7.6 by Sukumar

Project K (2024) - 8.4 by Nag Ashwin

RRR (2022) - 8.2 by S. S. Rajamouli

Summary:

Language: English, Total Movies: 14

Language: Hindi, Total Movies: 26

Language: Kannada, Total Movies: 3

Language: Marathi, Total Movies: 1

Language: Tamil, Total Movies: 3

Language: Telugu, Total Movies: 3

✧ CONCLUSION:

In conclusion, the Movie Dataset provides a rich and structured collection of data that allows for detailed analysis of various aspects of movies, including their ratings, directors, release years, and languages. The key attributes such as movie_id, title, rating, director, and language enable the exploration of trends, the identification of patterns, and the extraction of valuable insights. By organizing the data in a tab-separated format, it ensures easy handling and efficient processing within the Hadoop MapReduce framework. The structured nature of the dataset supports tasks like categorization, aggregation, and filtering, making it ideal for large-scale data

processing and analysis. Through leveraging the power of distributed computing, this dataset enables the processing of substantial amounts of movie-related information, ensuring that timely and actionable insights can be derived from it. This not only enhances our understanding of the movie industry but also empowers decision-making related to movie recommendations, audience preferences, and the influence of directors and languages on movie success. The use of Hadoop MapReduce for such analyses ensures scalability and fault tolerance, further bolstering the ability to work with large datasets efficiently. Therefore, this dataset serves as a valuable resource for anyone looking to gain a deeper understanding of movie trends and preferences in a data-driven manner.