Project Report-- **##Project 3: Data Analysis using Map/Reduce + Query Processing##**
Spring 2023

Division of Labor:

Team 31

· Sakshi(sx3702@mavs.uta.edu, 1001993702) (installation of Hadoop, MapReduce function, query plan) ~20hours+
· Spoorthi Seri(sxs2684@mavs.uta.edu,1002032680) (documentation, UTM-virtual machine) ~10hours
· Neha Thokala(nxt0631@mavs.uta.edu, 1002030631) (trend analysis, oracle run) ~10hours

Overall Status: Successfully completed both the tasks and documented the results.

- **[Task 1]** Completed the pre-requisite steps to setup a development environment by installing Hadoop on an Ubuntu OS
  - Used UTM virtual box on MacBook.
- **[Task 1]** Implemented a Map/Reduce program to count number of movies with different genre combinations in JAVA by following the steps provided.
  - Initially ran the basic Wordcount program – created, compiled jar (javac *.java file)
  - Incrementally developed to generate the required results (part-r-00000 file).
  - After final changes, generated a CSV file/graph from the output and analyzed the counts to determine trends.
  - After completion, verified the counts matched with the oracle table as well.
- **[Task 2]** Completed the pre-requisite to access oracle database for task 2.
  - Followed the steps provided on how to access oracle on Omega
- **[Task 2]** Created an SQL query to get top 5 movies based on movie rating with the expected criteria (at least 100k votes within a given date range and genre combination)
  - Ran an explain statement of the query to understand how the query performed.

Validated the results by filing simple select query on the title_basics table on the imdb00 database on Omega with the correct where clauses for titletype, startyear and genres;

SELECT count(*) FROM imdb00.TITLE_BASICS tb WHERE tb.STARTYEAR >= '2000' AND tb.STARTYEAR <= '2006' AND tb.GENRES LIKE '%Action%' AND tb.GENRES LIKE '%Drama%' AND tb.titletype = 'movie'

Analysis Results

[Task 1] Analysis of genre combinations with given date ranges

Task1 (mapReduce) **Output File: part-r-00000**

Year Range	Genre	Number of Movies
2000-2006	Action/Drama	1240
2000-2006	Adventure/Sci-Fi	67
2000-2006	Comedy/Romance	1620
2007-2013	Action/Drama	1811
2007-2013	Adventure/Sci-Fi	153
2007-2013	Comedy/Romance	2402
2014-2020	Action/Drama	2326
2014-2020	Adventure/Sci-Fi	247
2014-2020	Comedy/Romance	2784

Scenario1:

Data Range	Genre Combinations	No. of Movies
2000-2006	Action/Drama	1240
2007-2013	Action/Drama	1811
2014-2020	Action/Drama	2326
2000-2006	Adventure/Sci-Fi	67
2007-2013	Adventure/Sci-Fi	153
2014-2020	Adventure/Sci-Fi	247
2000-2006	Comedy/Romance	1620
2007-2013	Comedy/Romance	2402
2014-2020	Comedy/Romance	2784

Inference of Scenerio1:

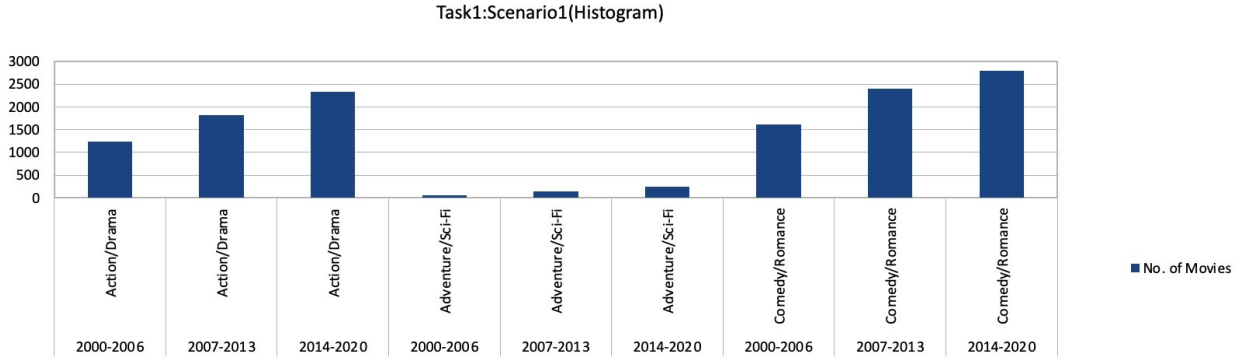
-All three genres (Action/Drama, Adventure/Sci-Fi, and Comedy/Romance) show an increasing trend in the number.

-Comedy/Romance is the most popular genre combination, with consistently high counts in all three time periods. It also shows the strongest increasing trend over time, with a total count of 2784 movies in the 2014-2020 period.

-Action/Drama and Adventure/Sci-Fi are less popular than Comedy/Romance, but still show an increasing trend in the number of movies produced over time.

-Adventure/Sci-Fi is the least popular genre combination, with the lowest counts in all three time periods. It shows a relatively flat trend over time, with a total count of only 467 movies across all three periods.

-Overall, there is an upward trend in the number of movies produced across all genres and time periods



Scenario2:

Data Range	Genre Combinations	No. of Movies
2000-2006	Action/Drama	1240
	Adventure/Sci-Fi	67
	Comedy/Romance	1620
2007-2013	Action/Drama	1811
	Adventure/Sci-Fi	153
	Comedy/Romance	2402
2014-2020	Action/Drama	2326
	Adventure/Sci-Fi	247
	Comedy/Romance	2784

Inference of Scenerio2:

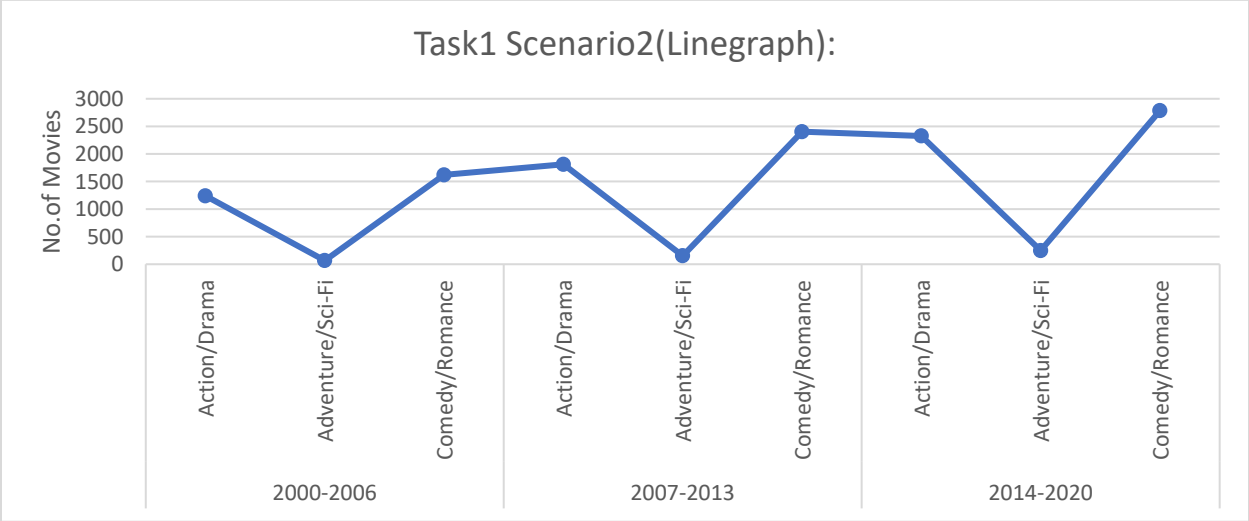
-In all three time periods, Comedy/Romance is the most popular genre combination with the highest count of movies produced. It is followed by Action/Drama, which has the second-highest count in all three time periods.

-Adventure/Sci-Fi is the least popular genre combination in all three time periods, with the lowest count of movies produced.

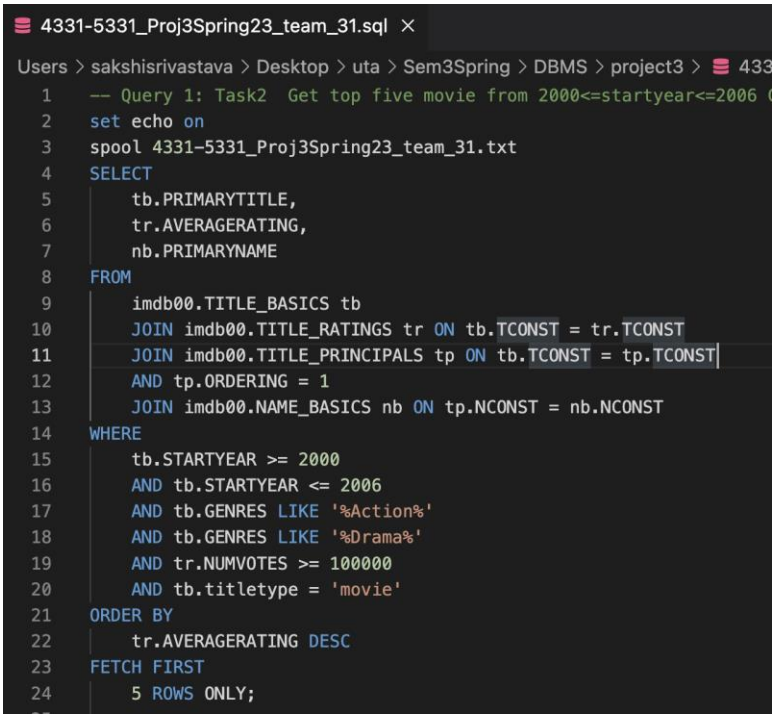
-Over time, the count of movies produced in all three genre combinations has increased. However, the rate of increase is not the same for all three genres. Comedy/Romance shows the highest rate of increase, followed by Action/Drama and then Adventure/Sci-Fi.

-The largest increase in the number of movies produced was seen in the Comedy/Romance genre combination, which saw a 71% increase in the number of movies produced between the 2000-2006 and 2014-2020 time periods.

-Adventure/Sci-Fi saw the smallest increase in the number of movies produced over time, with only a 268% increase between 2000-2006 and 2014-2020. This is likely due to the relatively low popularity of this genre combination compared to the other two genres.



[Task 2] Analysis of query explanation

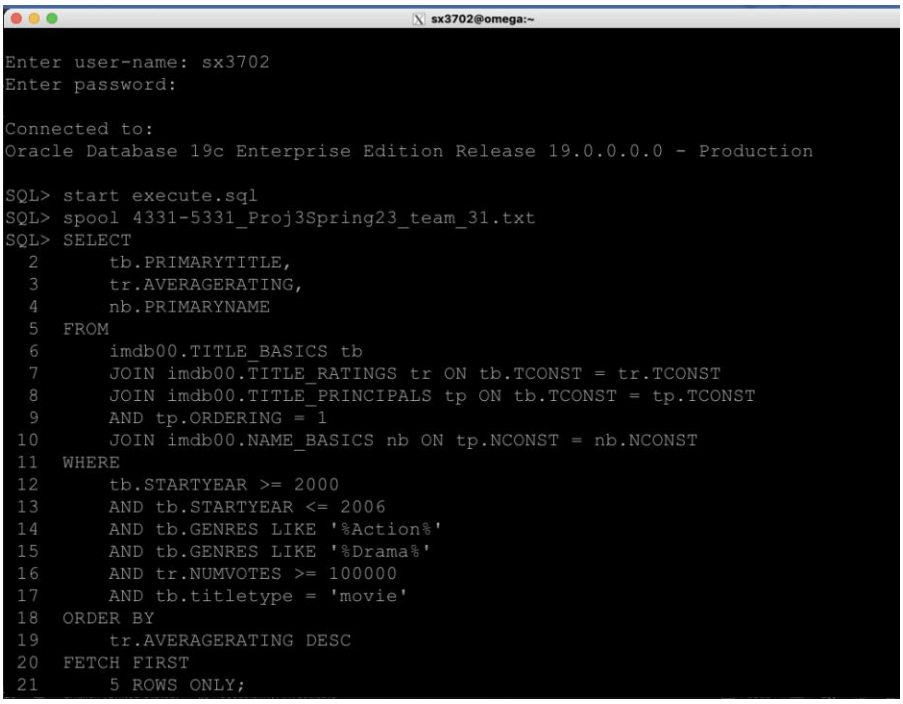


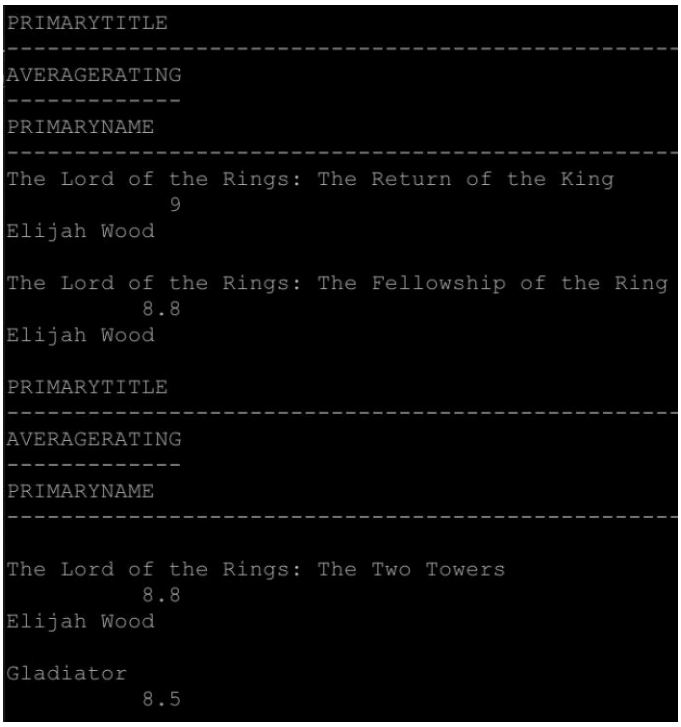
4331-5331_Proj3Spring23_team_31.sql

Users > sakshisrivastava > Desktop > uta > Sem3Spring > DBMS > project3 > 4

```
25
26  EXPLAIN PLAN
27  SET
28      statement_id = 'Query_1' FOR
29  SELECT
30      tb.PRIMARYTITLE,
31      tr.AVERAGERATING,
32      nb.PRIMARYNAME
33  FROM
34      imdb00.TITLE_BASICS tb
35      JOIN imdb00.TITLE_RATINGS tr ON tb.TCONST = tr.TCONST
36      JOIN imdb00.TITLE_PRINCIPALS tp ON tb.TCONST = tp.TCONST
37      AND tp.ORDERING = 1
38      JOIN imdb00.NAME_BASICS nb ON tp.NCONST = nb.NCONST
39  WHERE
40      tb.STARTYEAR >= 2000
41      AND tb.STARTYEAR <= 2006
42      AND tb.GENRES LIKE '%Action%'
43      AND tb.GENRES LIKE '%Drama%'
44      AND tr.NUMVOTES >= 100000
45      AND tb.titletype = 'movie'
46  ORDER BY
47      tr.AVERAGERATING DESC
48  FETCH FIRST
49      5 ROWS ONLY;
50
51  SELECT
52      PLAN_TABLE_OUTPUT
53  FROM
54      TABLE(DBMS_XPLAN.DISPLAY(NULL, 'Query_1', 'BASIC'));
55
56  SPOOL OFF
57  set echo off
```

[Task 2] Oracle run screenshot.





```
PRIMARYTITLE
--------------------------------------------------------------------
AVERAGERATING
-------------
PRIMARYNAME
--------------------------------------------------------------------
Russell Crowe

Oldboy
         8.4
Choi Min-sik
```

```
SQL> EXPLAIN PLAN
  2  SET
  3      statement_id = 'Query_1' FOR
  4  SELECT
  5      tb.PRIMARYTITLE,
  6      tr.AVERAGERATING,
  7      nb.PRIMARYNAME
  8  FROM
  9      imdb00.TITLE_BASICS tb
 10      JOIN imdb00.TITLE_RATINGS tr ON tb.TCONST = tr.TCONST
 11      JOIN imdb00.TITLE_PRINCIPALS tp ON tb.TCONST = tp.TCONST
 12      AND tp.ORDERING = 1
 13      JOIN imdb00.NAME_BASICS nb ON tp.NCONST = nb.NCONST
 14  WHERE
 15      tb.STARTYEAR >= 2000
 16      AND tb.STARTYEAR <= 2006
 17      AND tb.GENRES LIKE '%Action%'
 18      AND tb.GENRES LIKE '%Drama%'
 19      AND tr.NUMVOTES >= 100000
 20      AND tb.titletype = 'movie'
 21  ORDER BY
 22      tr.AVERAGERATING DESC
 23  FETCH FIRST
 24      5 ROWS ONLY;

Explained.
```

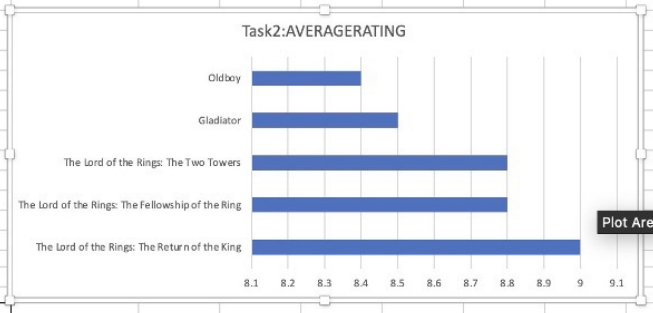
```
SQL>
SQL> SELECT
  2      PLAN_TABLE_OUTPUT
  3  FROM
  4      TABLE(DBMS_XPLAN.DISPLAY(NULL, 'Query_1', 'BASIC'));
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 2239367442

-------------------------------------------------------------------
| Id  | Operation                          | Name              |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                   |
|   1 |  VIEW                              |                   |
|   2 |   WINDOW SORT PUSHED RANK          |                   |
|   3 |    NESTED LOOPS                    |                   |
|   4 |     NESTED LOOPS                   |                   |
|   5 |      HASH JOIN                     |                   |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
|   6 |       NESTED LOOPS                 |                   |
|   7 |        NESTED LOOPS                |                   |
|   8 |         TABLE ACCESS FULL          | TITLE_RATINGS     |
|   9 |         INDEX UNIQUE SCAN          | SYS_C00547784     |
|  10 |        TABLE ACCESS BY INDEX ROWID| TITLE_BASICS      |
|  11 |       TABLE ACCESS FULL            | TITLE_PRINCIPALS  |
|  12 |      INDEX UNIQUE SCAN             | SYS_C00547785     |
|  13 |     TABLE ACCESS BY INDEX ROWID    | NAME_BASICS       |
-------------------------------------------------------------------

20 rows selected.
```

```
SQL>
SQL> SPOOL OFF
SQL> set echo off
```

Task2: SQL Queries Get top five movie from 2000<=startyear<=2006 GENRES='Action,Drama'

```
SELECT
tb.PRIMARYTITLE,
tr.AVERAGERATING,
nb.PRIMARYNAME
FROM
imdb00.TITLE_BASICS tb
JOIN imdb00.TITLE_RATINGS tr ON tb.TCONST = tr.TCONST
JOIN imdb00.TITLE_PRINCIPALS tp ON tb.TCONST = tp.TCONST
AND tp.ORDERING = 1
JOIN imdb00.NAME_BASICS nb ON tp.NCONST = nb.NCONST
WHERE
tb.STARTYEAR >= 2000
AND tb.STARTYEAR <= 2006
AND tb.GENRES LIKE '%Action%'
AND tb.GENRES LIKE '%Drama%'
AND tr.NUMVOTES >= 100000
AND tb.titletype = 'movie'
ORDER BY
tr.AVERAGERATING DESC
FETCH FIRST
5 ROWS ONLY;
```



OutputReceived:

PRIMARYTITLE	AVERAGERATING	PRIMARYNAME
The Lord of the Rings: The Return of the King	9	Elijah Wood
The Lord of the Rings: The Fellowship of the Ring	8.8	Elijah Wood
The Lord of the Rings: The Two Towers	8.8	Elijah Wood
Gladiator	8.5	Russell Crowe
Oldboy	8.4	Choi Min-sik

Inference of Task2: SQL Queries:

1) The top 3 movies are all part of "The Lord of the Rings" trilogy, with "The Return of the King" being the highest-rated movie with an average rating of 9.0.

2) The other two movies in the top 5 are "Gladiator" with an average rating of 8.5 and "Oldboy" with an average rating of 8.4.

3) Elijah Wood is the first-billed cast member in all three "The Lord of the Rings" movies, and Russell Crowe and Choi Min-sik are the first-billed cast members in "Gladiator" and "Oldboy", respectively.

File descriptions:

[Task 1] Map/Reduce mapper step which receives an input movie row and map's it into a key with date range as the prefix and genre combination as the suffix and a value of 1.

IMDbData.java

Users > sakshisrivastava > Desktop > uta > Sem3Spring > DBMS > project3 > IMDbData.java

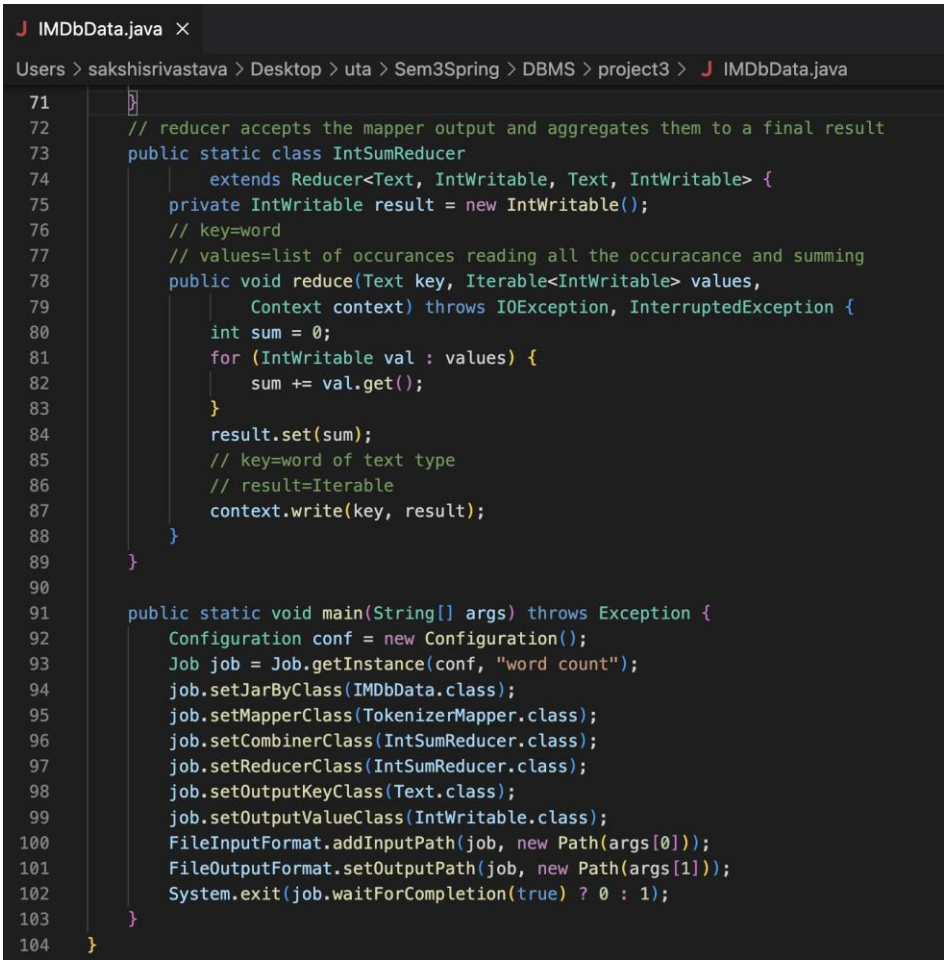
```java
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

public class IMDbData {
    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        // Define the date ranges and genre categories
        int[][] dateRanges = { { 2000, 2006 }, { 2007, 2013 }, { 2014, 2020 } };
        String[][] genreCategories = { { "Comedy", "Romance" }, { "Action", "Drama" }, { "Adventure", "Sci-Fi" } };
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        // mapper is tranformer
        // object key associated with this mapper execution is unique,value=sentence
        // context knows which file it is and which split it is
        // mapper emits all the occurance of the word inside one sentence
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            FileSplit fileSplit = (FileSplit) context.getInputSplit();
            // we can use log that will tell which file name is running inside this map
            String filename = fileSplit.getPath().getName();
            // StringTokenizer itr = new StringTokenizer(value.toString());
            // Split the sentence into individual parts
            String[] parts = value.toString().split(";");
            // Erro solution: Null /N is coming
            if (parts.length > 3 && parts[3].matches("-?\\d+(\\.\\d+)?")
                && parts[1].equals("movie") ) {
                // Extract the year value from the sentence
                int year = Integer.parseInt(parts[3]);
                // Extract the genres from the sentence
                List<String> genres = Arrays.asList(parts[4].split(","));
```

IMDbData.java

Users > sakshisrivastava > Desktop > uta > Sem3Spring > DBMS > project3 > IMDbData.java

```java
                List<String> genres = Arrays.asList(parts[4].split(","));
                // Find the date range that the year falls in
                String dateRangeLabel = "Unknown";
                for (int i = 0; i < dateRanges.length; i++) {
                    if (year >= dateRanges[i][0] && year <= dateRanges[i][1]) {
                        dateRangeLabel = dateRanges[i][0] + "-" + dateRanges[i][1];
                        break;
                    }
                }
                // Find the genre category that matches the genres in the sentence
                String genreCategoryLabel = "Unknown";
                for (int i = 0; i < genreCategories.length; i++) {
                    List<String> categoryGenres = Arrays.asList(genreCategories[i]);
                    if (genres.containsAll(categoryGenres)) {
                        genreCategoryLabel = String.join("/", categoryGenres);
                        break;
                    }
                }
                // Create the key with the specified format
                if (!dateRangeLabel.equals("Unknown") && !genreCategoryLabel.equals("Unknown")) {
                    String key1 = dateRangeLabel + " | " + genreCategoryLabel;
                    System.out.println(key1); // Output: 2007-2013 | Drama/Action
                    // word is text type , key1 is in string format
                    word.set(key1);
                    context.write(word, one);
                } else {
                    System.out.println("Unable to create key");
                }

            }
        }
    }
```

[Task 1] Map/Reduce reducer step which aggregates the occurrences of the keys generated in the mapper step.



[Task 1] Map/Reduce program summary

- IMDBData.java is a program for processing IMDb movie data with the Apache Hadoop MapReduce framework.
- The program consists of a main class IMDbData and two nested classes TokenizerMapper and IntSumReducer.
- The main function configures and runs the MapReduce job, serving as the entry point of the program.
- The TokenizerMapper class extends the Mapper class provided by Hadoop and overrides the map function to generate intermediate key-value pairs from input text data where the prefix is a date range and suffix is a combination of genres
    - ***2000-2006|Action/Drama***
- Each line of text data is split into individual parts, the year and genres of the movie are extracted, and the movie is categorized based on these factors, with each category represented as a key-value pair.

- The IntSumReducer class extends the Reducer class provided by Hadoop and overrides the reduce function to aggregate intermediate key-value pairs and produce final output key-value pairs.
- The main function sets up the Hadoop job by configuring input and output paths, map and reduce classes, and output key and value classes.
- The job is executed using the job.waitForCompletion() method, which submits the job to the Hadoop cluster and waits for it to complete. -The program returns 0 if the job completes successfully and 1 if not.
- The user provides input and output paths as command-line arguments. -The Hadoop cluster runs the MapReduce job and produces the output.

Logical errors and how we handled them:

- Encountered new lines (\N) which caused array out of bound errors. ***Solution***: Added below if condition to prevent processing any data with length less than 3 in IMDbData.java file.

```
39
40              // If Null is coming
41              if (parts.length > 3 && parts[3].matches("-?\\d+(\\.\\d+)?")
42                      && parts[1].equals("movie") ) {
```

- Initially the Map/Reduce program generated very high counts, we resolved the logical error by adding 'movie' in the JAVA filtering logic. After this change the counts matched with the SQL query counts.
- Giving wrong argument for the input data led to empty data problem even though Hadoop ran successfully without any errors (0%map and 100% reducer)
- Resolving JAR file build and compilation errors. We faced dependency and type mismatch errors when using java.util.* instead of the exact imports for java.util.Arrays and java.util.List

Installation errors and how we handled them:

- Resolved JAVA class path errors when setting up HADOOP by installing usr/local/bin directory and adding exports in ~/.bashrc file
- While installing Ubuntu faced issues as MacBook required java-11-openjdk-arm64 unlike all other systems which used amd64. Downloaded all dependency file with arm64.

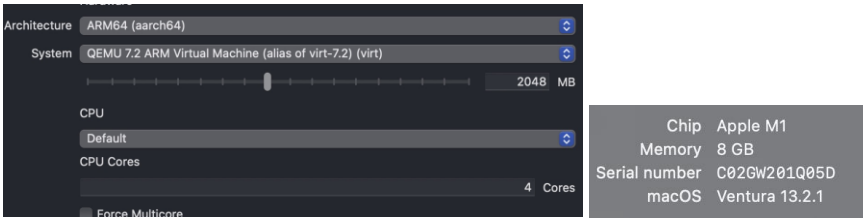
```
sx3702@sakshiubuntu:~$ source ~/.bashrc
bash: export: `=/usr/lib/jvm/java-11-openjdk-arm64': not a valid identifier
bash: export: `=/usr/local/hadoop/': not a valid identifier
sx3702@ubunto:~/Downloads$ hadoop-3.3.5/bin/hdfs namenode -format
ERROR: JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64 does not exist.
```

```
117 fi
118 export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-arm64
119 export HADOOP_HOME=/usr/local/hadoop
120 export PATH=$PATH:$HADOOP_HOME/bin
121 export PATH=$PATH:$HADOOP_HOME/sbin
122 export HADOOP_MAPRED_HOME=$HADOOP_HOME
123 export HADOOP_COMMON_HOME=$HADOOP_HOME
```

- Got permission denied in Ubuntu while doing pdsh, solution re-configure ubuntu with SSH external.

```
sx3702@ubunto:/usr/local$ export PDSH_RCMD_TYPE=ssh
sx3702@ubunto:/usr/local$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as sx3702 in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
localhost: sx3702@localhost: Permission denied (publickey,password).
pdsh@ubunto: localhost: ssh exited with exit code 255
Starting datanodes
localhost: sx3702@localhost: Permission denied (publickey,password).
pdsh@ubunto: localhost: ssh exited with exit code 255
Starting secondary namenodes [ubunto]
ubunto: sx3702@ubunto: Permission denied (publickey,password).
pdsh@ubunto: ubunto: ssh exited with exit code 255
2023-04-17 21:51:51,144 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Starting resourcemanager
resourcemanager is running as process 7531.  Stop it first and ensure /tmp/hadoo
p-sx3702-resourcemanager.pid file is empty before retry.
Starting nodemanagers
localhost: sx3702@localhost: Permission denied (publickey,password).
pdsh@ubunto: localhost: ssh exited with exit code 255
```

- Mac Book keeps on getting hang due to storage shortage. iCloud Drive is always full. After multiple attempts found balanced memory resource on UTM with 2048 MB and 4 core



Future Improvements

Here are some possible future improvements

1. Use of Combiner Class: Currently, the program uses the same reducer class for combiner as well. Using a separate combiner class can help in reducing the amount of data sent from mapper to reducer, thus improving the overall performance.
2. Better Error Handling: The current program only checks for a few specific conditions before emitting the output. However, there can be other cases that might result in an error. Therefore, adding more error handling code can improve the robustness of the program.

3. Use of Custom Data Types: The program uses Text and IntWritable as input and output types respectively. However, in some cases, using custom data types can provide better abstraction and make the code more readable and maintainable.
4. Use of Advanced Techniques: This program uses basic MapReduce techniques to analyze the data. However, there are other advanced techniques like Apache Spark, Apache Flink, or Hadoop Streaming, which can help in processing large datasets more efficiently.
5. More Complex Analysis: The current program only finds the count of movies based on their genre and release date. However, there are many other interesting insights that can be obtained from the IMDb dataset. For example, analyzing the ratings of movies based on their genre and release date can help in identifying popular genres or trends over time.

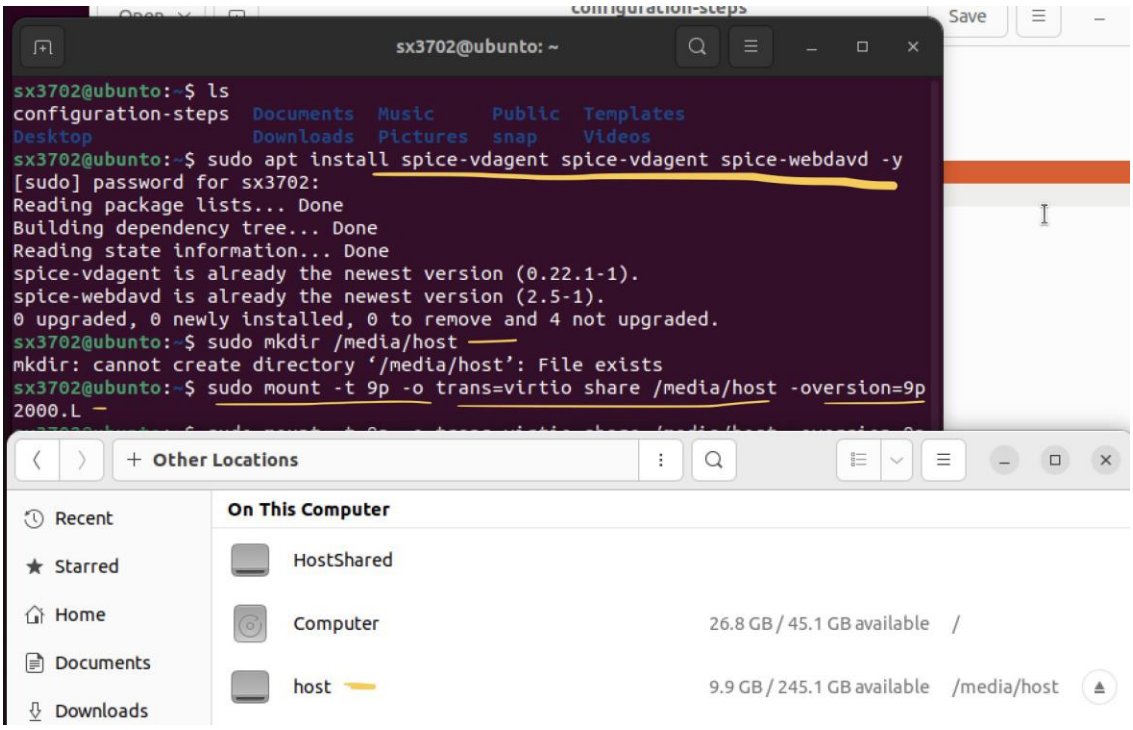
Screenshots

Mounted mac file directory to UTM /media/host folder

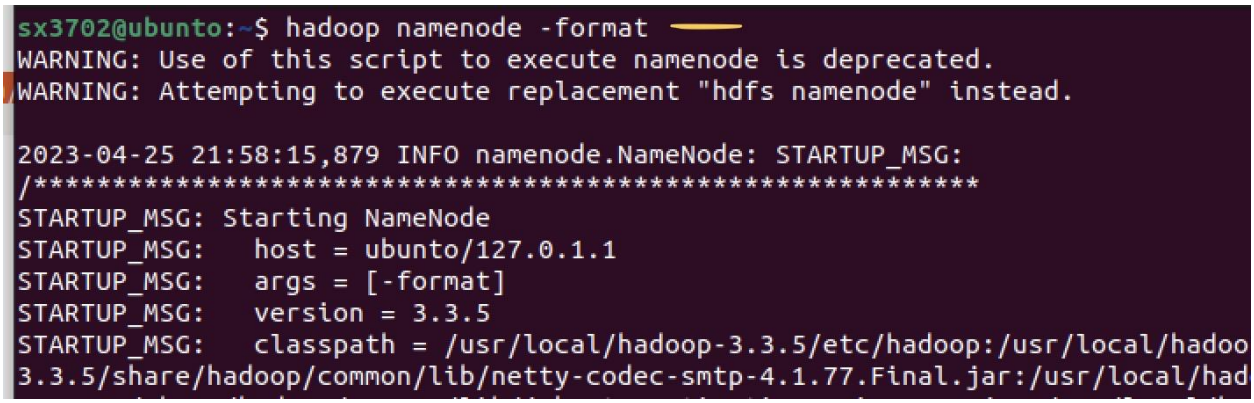
```
Remote disk share fromm mac to UTM -->
sudo apt install spice-vdagent spice-vdagent spice-webdavd -y
(make sure project3 folder is shared before starting UTM)
sudo mkdir /media/host
sudo mount -t 9p -o trans=virtio share /media/host -oversion=9p2000.L
```

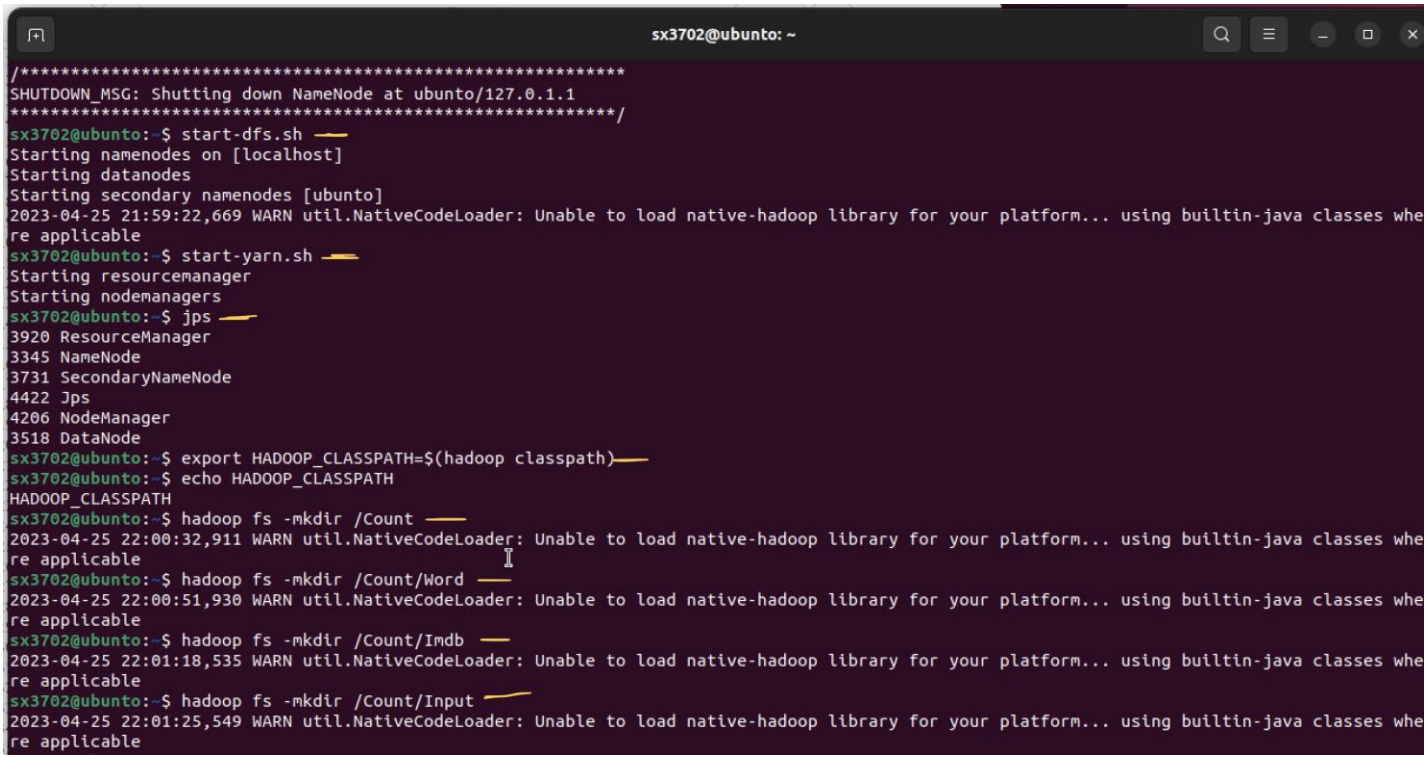
Hadoop restart and connect with server.

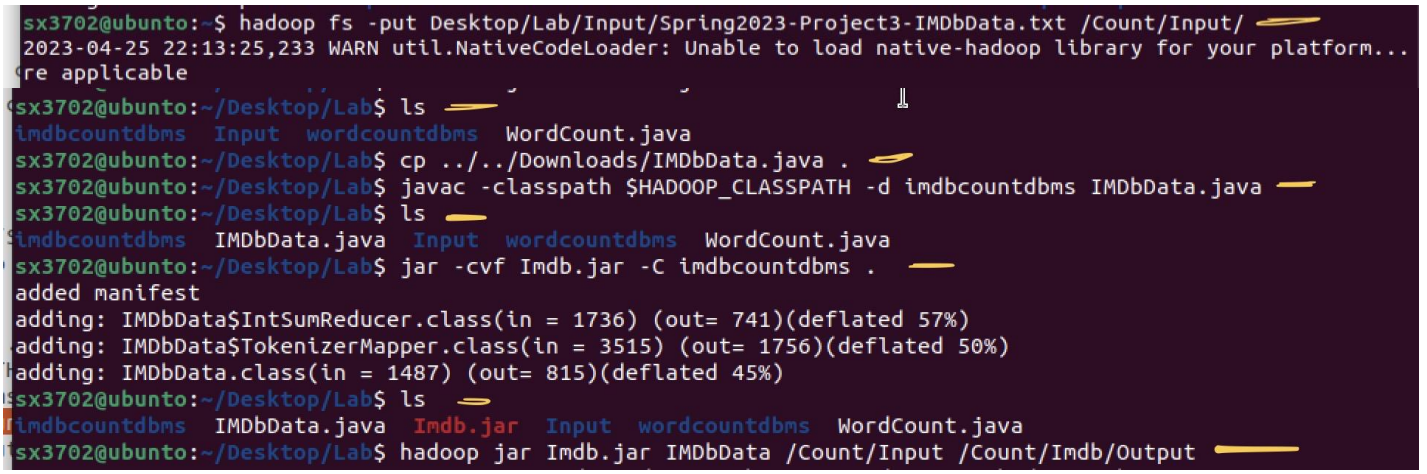
```
hadoop restart---------->
hadoop namenode -format
start-yarn.sh
start-dfs.sh
jps
export HADOOP_CLASSPATH=$(hadoop classpath)
hadoop fs -mkdir /Count
hadoop fs -mkdir /Count/Word
hadoop fs -mkdir /Count/Imdb
hadoop fs -mkdir /Count/Input
hadoop fs -put Desktop/Lab/Input/Spring2023-Project3-IMDbData.txt /Count/Input
copy imdb.java from hostshared to download folder

cd Desktop/Lab
cp ../../Downloads/IMDbData.java .
javac -classpath $HADOOP_CLASSPATH -d imdbcountdbms IMDbData.java
jar -cvf Imdb.jar -C imdbcountdbms .
hadoop jar Imdb.jar IMDbData /Count/Input /Count/Imdb/Output
hadoop dfs -cat /Count/Imdb/Output/*
```



Hadoop execution steps




sx3702@ubunto: ~
/**
SHUTDOWN_MSG: Shutting down NameNode at ubunto/127.0.1.1
**/
sx3702@ubunto:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubunto]
2023-04-25 21:59:22,669 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
sx3702@ubunto:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
sx3702@ubunto:~$ jps
3920 ResourceManager
3345 NameNode
3731 SecondaryNameNode
4422 Jps
4206 NodeManager
3518 DataNode
sx3702@ubunto:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
sx3702@ubunto:~$ echo HADOOP_CLASSPATH
HADOOP_CLASSPATH
sx3702@ubunto:~$ hadoop fs -mkdir /Count
2023-04-25 22:00:32,911 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
sx3702@ubunto:~$ hadoop fs -mkdir /Count/Word
2023-04-25 22:00:51,930 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
sx3702@ubunto:~$ hadoop fs -mkdir /Count/Imdb
2023-04-25 22:01:18,535 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
sx3702@ubunto:~$ hadoop fs -mkdir /Count/Input
2023-04-25 22:01:25,549 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

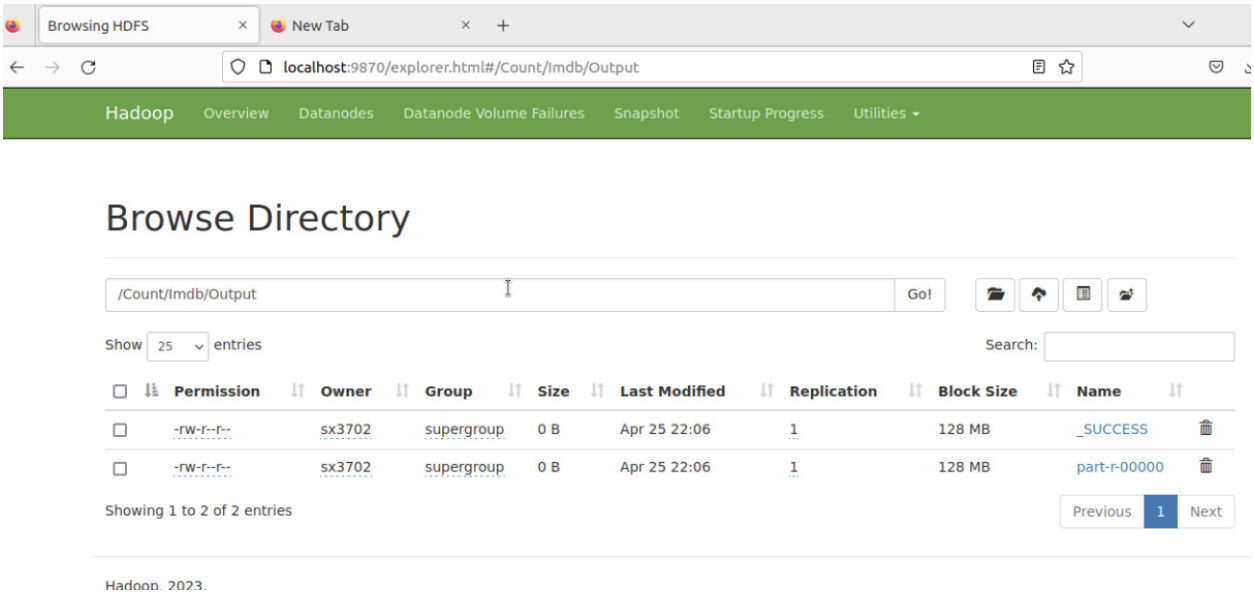

sx3702@ubunto:~$ hadoop fs -put Desktop/Lab/Input/Spring2023-Project3-IMDbData.txt /Count/Input/
2023-04-25 22:13:25,233 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... re applicable
sx3702@ubunto:~/Desktop/Lab$ ls
imdbcountdbms Input wordcountdbms WordCount.java
sx3702@ubunto:~/Desktop/Lab$ cp ../../Downloads/IMDbData.java .
sx3702@ubunto:~/Desktop/Lab$ javac -classpath $HADOOP_CLASSPATH -d imdbcountdbms IMDbData.java
sx3702@ubunto:~/Desktop/Lab$ ls
imdbcountdbms IMDbData.java Input wordcountdbms WordCount.java
sx3702@ubunto:~/Desktop/Lab$ jar -cvf Imdb.jar -C imdbcountdbms .
added manifest
adding: IMDbData$IntSumReducer.class(in = 1736) (out= 741)(deflated 57%)
adding: IMDbData$TokenizerMapper.class(in = 3515) (out= 1756)(deflated 50%)
adding: IMDbData.class(in = 1487) (out= 815)(deflated 45%)
sx3702@ubunto:~/Desktop/Lab$ ls
imdbcountdbms IMDbData.java Imdb.jar Input wordcountdbms WordCount.java
sx3702@ubunto:~/Desktop/Lab$ hadoop jar Imdb.jar IMDbData /Count/Input /Count/Imdb/Output

```
1682459971055_0003
2023-04-25 22:14:29,006 INFO input.FileInputFormat: Total input files to process : 1
2023-04-25 22:14:29,518 INFO mapreduce.JobSubmitter: number of splits:4
2023-04-25 22:14:29,663 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1682459971055_0003
2023-04-25 22:14:29,663 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-04-25 22:14:29,791 INFO conf.Configuration: resource-types.xml not found
2023-04-25 22:14:29,791 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-04-25 22:14:30,176 INFO impl.YarnClientImpl: Submitted application application_1682459971055_0003
2023-04-25 22:14:30,226 INFO mapreduce.Job: The url to track the job: http://ubunto:8088/proxy/application_1
2023-04-25 22:14:30,227 INFO mapreduce.Job: Running job: job_1682459971055_0003
2023-04-25 22:14:35,427 INFO mapreduce.Job: Job job_1682459971055_0003 running in uber mode : false
2023-04-25 22:14:35,429 INFO mapreduce.Job:  map 0% reduce 0%
2023-04-25 22:14:55,045 INFO mapreduce.Job:  map 52% reduce 0%
2023-04-25 22:14:57,067 INFO mapreduce.Job:  map 100% reduce 0%
2023-04-25 22:15:00,094 INFO mapreduce.Job:  map 100% reduce 100%
2023-04-25 22:15:01,143 INFO mapreduce.Job: Job job_1682459971055_0003 completed successfully
2023-04-25 22:15:01,636 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=1124
                FILE: Number of bytes written=1384076
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=497433541
                HDFS: Number of bytes written=284
                HDFS: Number of read operations=17
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
```

Hadoop final output

```
sx3702@ubunto:~/Desktop/Lab$ hadoop dfs -cat /Count/Imdb/Output2/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2023-04-25 22:23:32,154 WARN util.NativeCodeLoader: Unable to load nati
re applicable
2000-2006 | Action/Drama        1240
2000-2006 | Adventure/Sci-Fi    67
2000-2006 | Comedy/Romance      1620
2007-2013 | Action/Drama        1811
2007-2013 | Adventure/Sci-Fi    153
2007-2013 | Comedy/Romance      2402
2014-2020 | Action/Drama        2326
2014-2020 | Adventure/Sci-Fi    247
2014-2020 | Comedy/Romance      2784
```

Hadoop file system



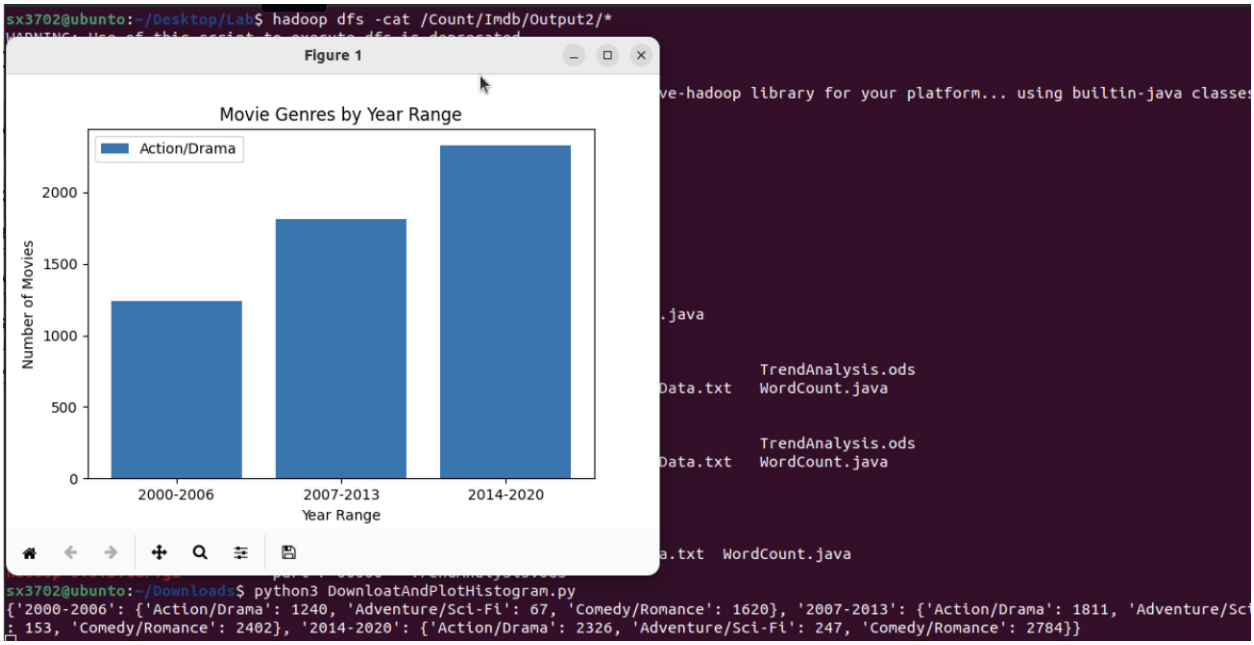
Additional Work

[Task 1] Wrote python script to automate manual work of downloading part-r-0000 map reduce output from Hadoop localhost that has code to convert csv format to json and directly display analysis graph without human interweaving.

DownloatAndPlotHistogram.py

Users > sakshisrivastava > Desktop > uta > Sem3Spring > DBMS > project3 > DownloatAndPlotHistogr

```python
#!pip3 install matplotlib
import matplotlib.pyplot as plt
data = {}
with open('/home/sx3702/Downloads/part-r-00000', 'r') as f:
    for line in f:
        year_genre, count = line.strip().split('\t')
        year, genre = year_genre.split(' | ')
        count = int(count)
        if year not in data:
            data[year] = {}
        data[year][genre] = count
print(data)
# data = {
#     '2000-2006': {'Action/Drama': 167371, 'Adventure/Sci-Fi': 11891, 'Comedy/Rom
#     '2007-2013': {'Action/Drama': 330421, 'Adventure/Sci-Fi': 27804, 'Comedy/Rom
#     '2014-2020': {'Action/Drama': 474792, 'Adventure/Sci-Fi': 49363, 'Comedy/Rom
# }
def plot_histogram(start_date, end_date, genres):
    #genres = ['Action/Drama', 'Adventure/Sci-Fi', 'Comedy/Romance']
    data_range = {}
    for year in data:
        if start_date <= year <= end_date:
            for genre in genres:
                if genre in data[year]:
                    if year in data_range:
                        data_range[year][genre] = data[year][genre]
                    else:
                        data_range[year] = {genre: data[year][genre]}
    fig, ax = plt.subplots()
    for genre in genres:
        genre_data = [data_range[year][genre] for year in data_range]
        ax.bar(data_range.keys(), genre_data, label=genre)
    ax.set_xlabel('Year Range')
    ax.set_ylabel('Number of Movies')
    ax.set_title('Movie Genres by Year Range')
    ax.legend()
    plt.show()
    plt.show()
plot_histogram('2000-2006', '2014-2020', ['Action/Drama'])
plot_histogram('2000-2006', '2014-2020', ['Adventure/Sci-Fi'])
plot_histogram('2000-2006', '2014-2020', ['Comedy/Romance'])
```


sx3702@ubunto:~/Desktop/Lab$ hadoop dfs -cat /Count/Imdb/Output2/*
Figure 1
Movie Genres by Year Range
Action/Drama
Number of Movies
0
500
1000
1500
2000
2000-2006
2007-2013
2014-2020
Year Range
ve-hadoop library for your platform... using builtin-java classes
.java
TrendAnalysis.ods
Data.txt WordCount.java
TrendAnalysis.ods
Data.txt WordCount.java
a.txt WordCount.java
sx3702@ubunto:~/Downloads$ python3 DownloatAndPlotHistogram.py
{'2000-2006': {'Action/Drama': 1240, 'Adventure/Sci-Fi': 67, 'Comedy/Romance': 1620}, '2007-2013': {'Action/Drama': 1811, 'Adventure/Sci
: 153, 'Comedy/Romance': 2402}, '2014-2020': {'Action/Drama': 2326, 'Adventure/Sci-Fi': 247, 'Comedy/Romance': 2784}}

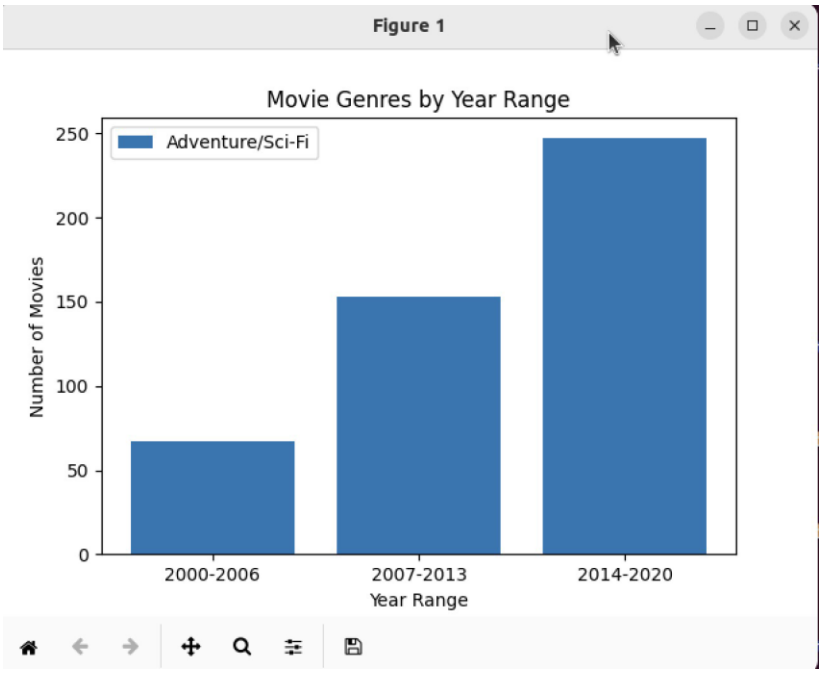


Figure 1
Movie Genres by Year Range
Adventure/Sci-Fi
Number of Movies
0
50
100
150
200
250
2000-2006
2007-2013
2014-2020
Year Range

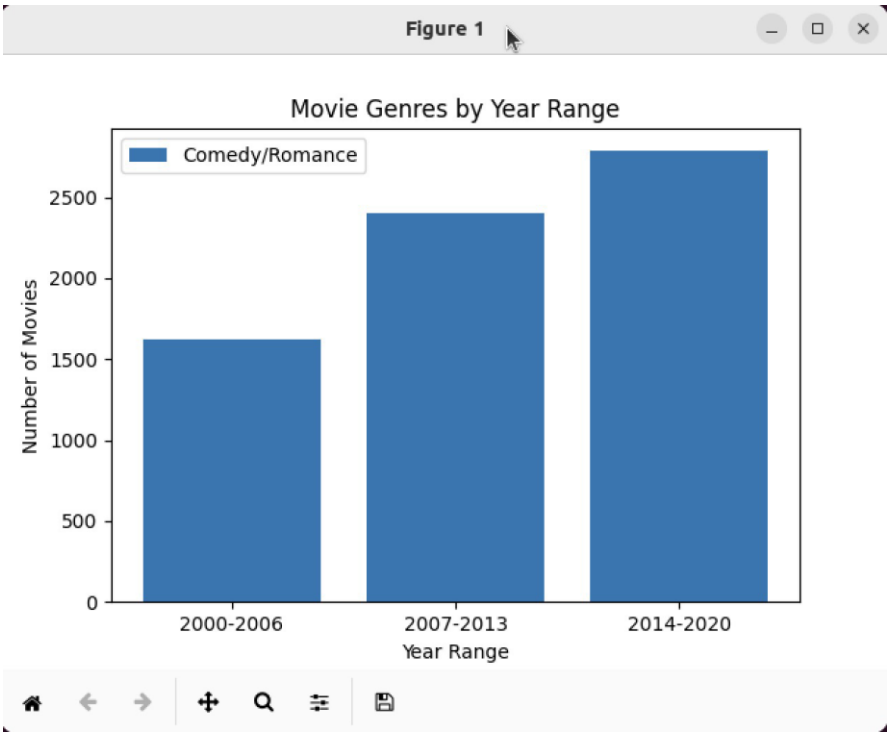


Figure 1
Movie Genres by Year Range
Comedy/Romance
Number of Movies
2500
2000
1500
1000
500
0
2000-2006
2007-2013
2014-2020
Year Range