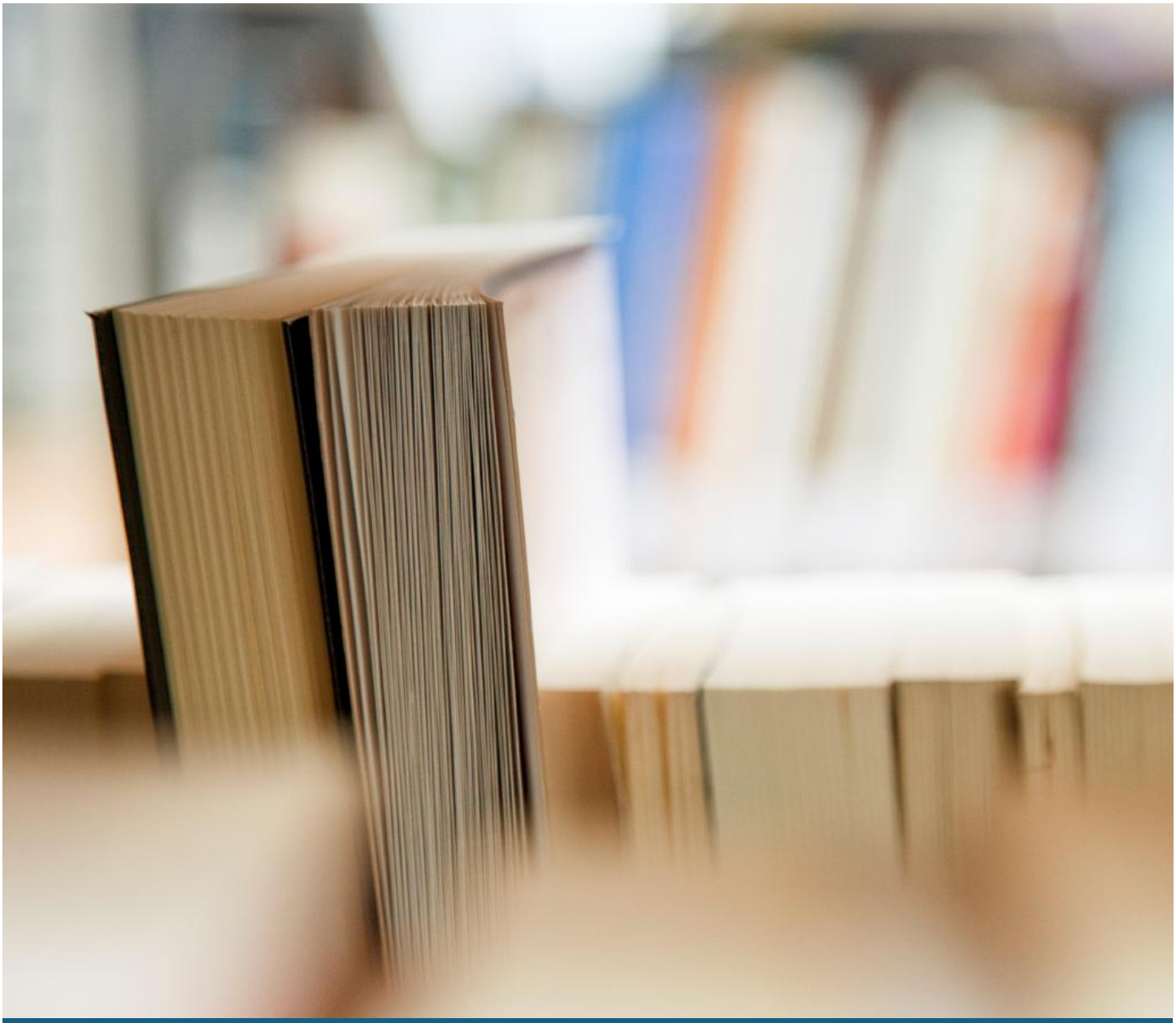

Map/Reduce Program and Query Processing

CSE 5331 – 001 Report

Inshaad Merchant, Araohat Kokate, Aindrila Bhattacharya 4/22/25 CSE 5331-001

Table Of Contents

Overall Status

Task 1 involved implementing the map and reduce functions on an IMDb dataset that was provided. This was done by setting up a Hadoop file system within WSL Ubuntu (since the coding was done on a Windows 11 OS) and then changing the map and reduce functions that was given in the WordCount.java file. The map function parses a line of the file and splits it according to semicolon and then gathers the required parts only (title type, title year, title rating and genres). It sorts out first by title type, picking out everything that is not a movie. Then it gets the movies in the proper range of years, then the rating, then it parses the genres to find the proper combination of genres. The key is set to the year range and genre combination, and the value is set to 1 for every combination encountered that fits the parameters. After sorting the <key, value> pairs, they are passed into the combiner function which counts the number of same keys in each map output and groups them together. The reducer function does the same except for all map outputs together. Other than the map function, all the code is the same as the WordCount.java file provided on the official Hadoop documentation and it suits our purposes so it remains unchanged.

Task 2 involved logging into UTA's Oracle database and running an SQL query on the IMDb database there to find the **top 5 Comedy/Romance movies in 2011 – 2020.** The query outputted the movie title, the rating, the number of votes it had, and the lead actor or actress. Some difficulties involved looking up how to push the SQL file onto Omega and execute the query using the file itself, as WSL wouldn't allow for copy-pasting multiple lines of query. It also involved putting the query into an EXPLAIN PLAN statement so to see how the query was being run.

Analysis Results

Task 1:

The results of Task 1 are as follows:

```
hadoopuser@Inshaad:~/hadooppproj3$ hdfs dfs -cat /imdboutput/part-r-00000
[1991-2000],Action,Thriller     55
[1991-2000],Adventure,Drama     56
[1991-2000],Comedy,Romance      215
[2001-2010],Action,Thriller     76
[2001-2010],Adventure,Drama     141
[2001-2010],Comedy,Romance      400
[2011-2020],Action,Thriller     208
[2011-2020],Adventure,Drama     343
[2011-2020],Comedy,Romance      590
```

According to these results, we can make this following graph:

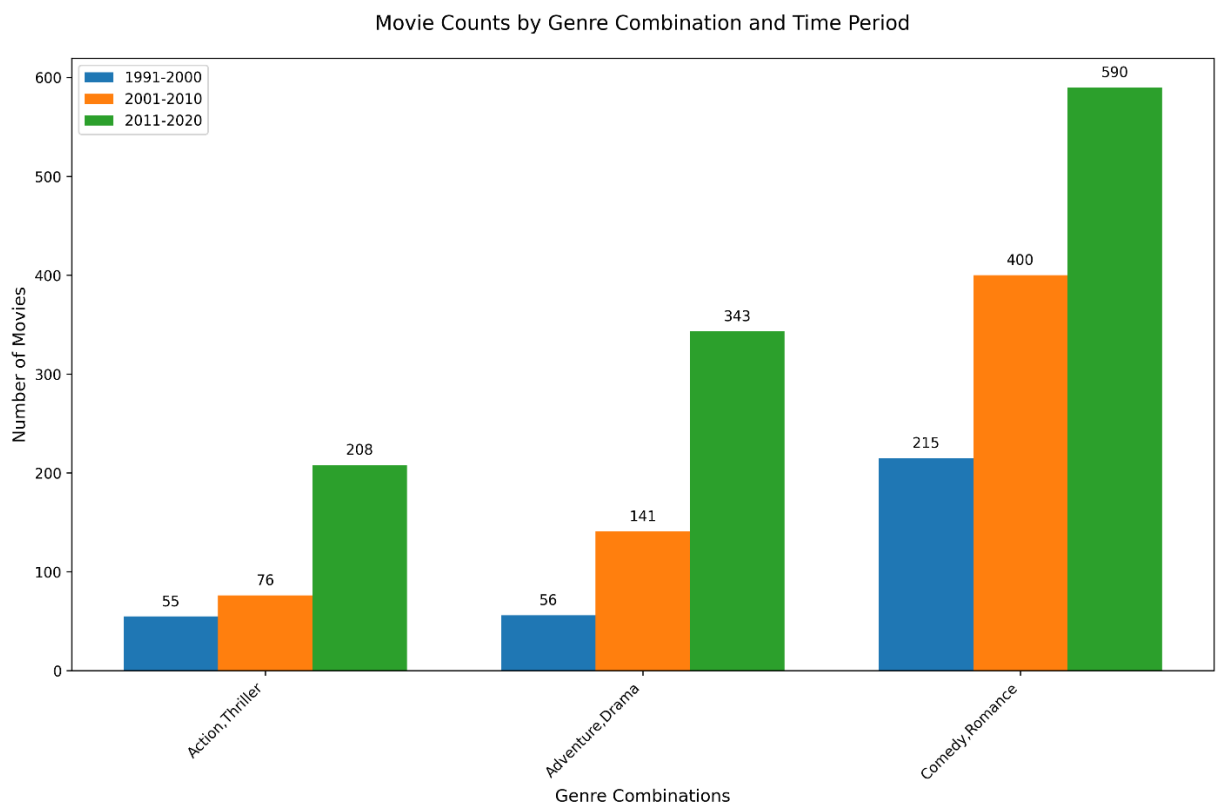

Figure 1: Movie Counts by Genre Combination and the Time Period

This shows overall that as time has gone on, there has been a general increase in the number of movies being made. This can be attributed to better filming conditions, allowing to an increase in the number of films that could be made due to better avenues of artistic expression.

We can also make the following graph:

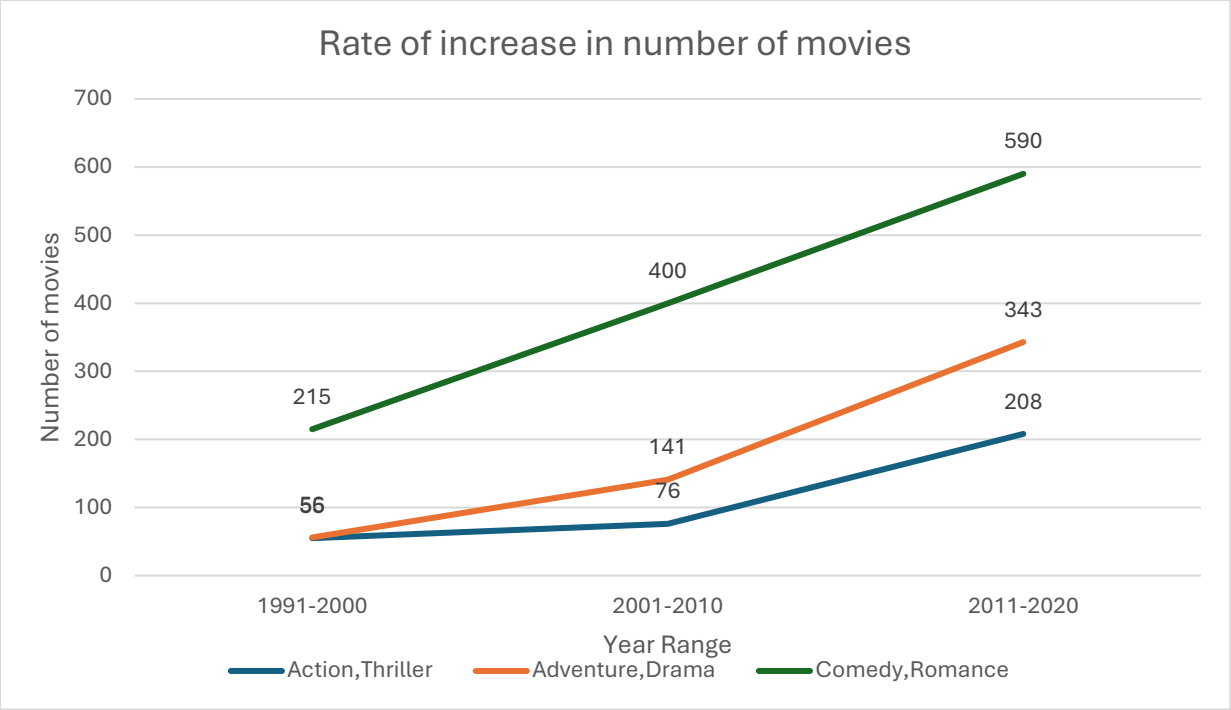

Figure 2: Rate of increase in number of movies

According to this graph, while the rate of increase in the number of comedy and romance movies is relatively steady, the rate of increase in the number of action-drama and adventure-thriller movies has a sharp increase.

Task 2:

Raw data:

```
Movie Title
--------------------------------------------------------------------------------
    Rating Number of Votes
---------- ---------------
Lead Actor/Actress
--------------------------------------------------------------------------------
The Artist
       7.9          240576
Jean Dujardin

Moonrise Kingdom
       7.8          346556
Jared Gilman

Movie Title
--------------------------------------------------------------------------------
    Rating Number of Votes
---------- ---------------
Lead Actor/Actress
--------------------------------------------------------------------------------

Silver Linings Playbook
       7.7          702627
Bradley Cooper

Midnight in Paris
       7.7          419852

Movie Title
--------------------------------------------------------------------------------
    Rating Number of Votes
---------- ---------------
Lead Actor/Actress
--------------------------------------------------------------------------------
Owen Wilson

50/50
       7.6          328825
Joseph Gordon-Levitt
```

Figure 3: Raw - Top 5 Comedy/Romance Movies from 2011-2020

Data after being cleaned up in excel for better format:

Movie Title	Rating	Number of Votes	Lead Actor/Actress
The Artist	7.9	240576	Jean Dujardin
Moonrise Kingdom	7.8	346556	Jared Gilman
Silver Linings Playbook	7.7	702627	Bradley Cooper
Midnight in Paris	7.7	419852	Owen Wilson
50/50	7.6	328825	Joseph Gordon-Levitt

Table 1: Clean - Top 5 Comedy/Romance Movies from 2011-2020

SQL Query Explain Plan Output:

```
SQL> @explain_query_plan.sql

Explained.


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.
```

The explanation of the EXPLAIN PLAN line by line is as follows:

Id	Operation	Object (Name)	Description
0	SELECT STATEMENT		Final query execution
1	VIEW		This is the *WITH ComedyRomanceMovies* statement initialized as a VIEW statement
2	WINDOW SORT PUSHED RANK		Implements the *ORDER BY averagerating DESC FETCH FIRST 5 ROWS ONLY*
3	NESTED LOOPS		Joining multiple tables
4	NESTED LOOPS		Inner join inside the join chain

5	HASH JOIN		Join between *title_basics* and *title_ratings*
6	NESTED LOOPS		Further joins (*title_principals* → *name_basics*)
7	NESTED LOOPS		More joining logic
8	TABLE ACCESS FULL	TITLE_RATINGS	Full scan of ratings table (likely because of *numvotes* >= 150000 filter)
9	INDEX UNIQUE SCAN	SYS_C00547784	Fast index lookup on *title_basics* using primary key *tconst*
10	TABLE ACCESS BY INDEX ROWID	TITLE_BASICS	Accessing rows from *title_basics* via index
11	TABLE ACCESS FULL	TITLE_PRINCIPALS	Full scan, searching for the lead actor using the ordering = '1'
12	INDEX UNIQUE SCAN	SYS_C00547785	Fast index lookup on *name_basics* using *nconst*
13	TABLE ACCESS BY INDEX ROWID	NAME_BASICS	Accessing rows from *name_basics* via index

File Descriptions

README.md – read me file explaining the project

WordCount.java – edited to parse the IMDb dataset provided with the problem.

Visualize.py – used to make Figure 1

Task1_Figure2.xlsx – used to make Figure 2

Top_comedy_romance_movies.sql – the query for the Omega database to find the proper data entries

Explain_query_plan.sql – the explain plan query

4331-5331_Proj3Spring25_team_15.sql – submittable sql

Division of Labor

Inshaad Merchant – Task 1, Task 2, graph, report

Araohat Kokate – Task 1, general debugging, report

Aindrila Bhattacharya – Task 1, Task 2, graph, report