Project Report - Project 3 Data Analysis using Map/Reduce and Query Processing

DBMS Models and Implementation

CSE 4331-001

Yahia Elsaad

Dev Patel

Mohamad Nabih Alkhateeb

Table Of Contents

Overall Status

For this project, all group members successfully completed both major components: the MapReduce program along with trend analysis of genre combinations we determined from our program along with a SQL query with EXPLAIN PLAN analysis using Oracle on Omega. We approached the project by first understanding the structure of the IMDb dataset, identifying the relevant tables (title_basics, title_ratings, title_principals, and name_basics), and verifying access and schema using DESC commands.

Task 1 – MapReduce Analysis

Map Function:
The Map function is responsible for scanning through each line of the IMDb dataset and identifying movies that match our specific criteria. For Task 1, we wanted to count highly-rated movies (rating ≥ 7.0) that fit within three particular genre combinations—Action & Thriller, Adventure & Drama, and Comedy & Romance—across three defined time periods: 1991–2000, 2001–2010, and 2011–2020. The Map function filters out any entry that doesn't meet those criteria, ensuring we only process relevant movie data. Its main goal is to tag valid entries with a custom key that includes both the time period and the genre combo, and assign a value of 1 to each valid match. These outputs will then be passed to the Reducer for aggregation.

Implementation:
To implement the Map function, we started by reading in each line of the input file and splitting it using the ‘;’ as the delimiter to access each field. We then extracted key values such as the title type, release year, rating, and genres. We filtered out any rows that weren't movies, had ratings below 7.0, or didn't fall into the specified year ranges. For the genre check, we used simple string matching to verify if both genres in any of the three combinations were present. If a line met all these conditions, we constructed a key using the format [year-range],GenreCombo and wrote it to the context with a value of 1. This approach allowed us to efficiently tag and pass only valid entries forward to the Reducer while skipping over any unnecessary data.

Reduce Function:
The Reduce function is responsible for taking all the intermediate key-value pairs output by the Map function and combining them to produce the final result. Specifically, for each unique genre and time-period combination (e.g., [2001–2010],Action;Thriller), the Reduce function receives a list of values—all 1s—that represent individual matching movies. Its job is to simply sum these values to get the total count of highly-rated movies for that genre combo within the given decade. This step is essential to aggregate all the relevant data and produce a clear, human-readable output.

Implementation:
To implement the Reduce function, we defined a new class that extends the Hadoop Reducer class, taking in Text keys and IntWritable values. Within the reduce() method, we wrote a simple loop to iterate through the list of values for each key, summing them to get the total count. Once the sum was calculated, we wrote the result back to the Hadoop context with the original key and the final total. This was a

straightforward step in the project, but it really brought everything together—transforming individual matching records into meaningful insights.

Task 2 – SQL Query & Query Plan Analysis

- For Task 2, we focused on writing a SQL query that could filter and display the top 5 highest-rated movies that fit the Action/Thriller genre combination, released between 2011 and 2020. We carefully selected conditions that ensured we only retrieved high-quality, popular movies—specifically those with at least 150,000 user votes and an average rating of 7.0 or higher. To make the results more meaningful, we also included the lead actor or actress for each movie by joining with the title_principals and name_basics tables and filtering for ordering = '1'. We applied additional SQL*Plus formatting using SET LINESIZE, COLUMN FORMAT, and ORDER BY clauses to keep the output well-organized and easy to interpret. This helped ensure our results were not just accurate, but also clean and presentable for analysis and reporting.
- We didn't stop at just writing the query—we also analyzed its execution strategy using Oracle's EXPLAIN PLAN along with DBMS_XPLAN.DISPLAY() to understand how the database engine optimized the query behind the scenes. We found that Oracle used a combination of hash joins, nested loops, and full table scans to efficiently handle the multi-table join and apply the filters in the right order. The plan revealed operations like WINDOW SORT PUSHED RANK, which explained how Oracle determined the top 5 movies based on rating using a window function. This part of the task gave us valuable experience interpreting how SQL queries are executed at a deeper level, and allowed us to confirm that our query was both logically correct and performance-efficient.

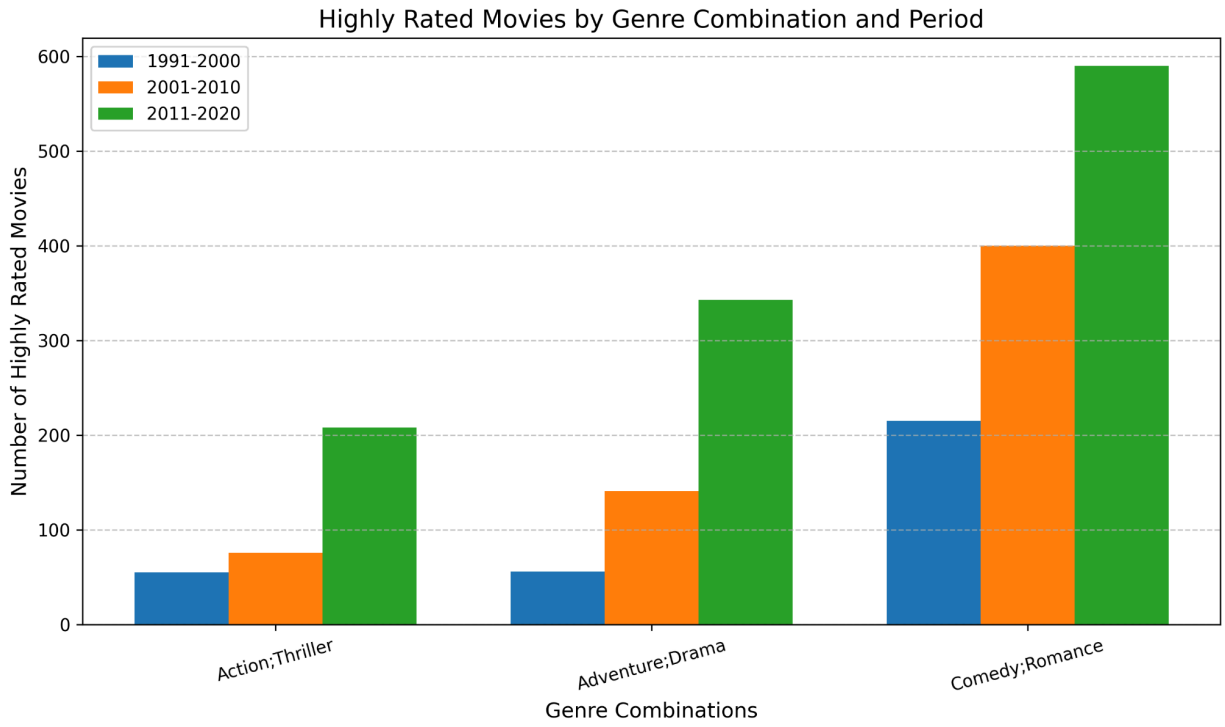
Challenges Faced

- A notable difficulty was interpreting and aligning the EXPLAIN PLAN output with the actual SQL query, especially understanding internal operations like WINDOW SORT PUSHED RANK, which were not directly visible in the original query. Additionally, the output formatting in SQL*Plus required extra care to ensure the final result and plan were presented clearly and consistently for the report. Another difficulty we encountered was during the setup phase. All group members faced configuration errors when trying to start Hadoop services, most notably issues where a node would not appear or fail to register properly (usually DataNode or NameNode). Resolving this required identifying missing steps (such as formatting HDFS correctly and running the services in proper order), and verifying environment variables like JAVA_HOME.

Overall, this project improved our knowledge of relational database optimization and distributed data processing. We overcame technical setup and interpretation issues to successfully provide valuable insights from real-world data. The hands-on experience with Hadoop and Oracle SQL gave us important exposure to tools and ways to problem-solving that are relevant to the world of databases.

Analysis Results

TASK 1 - M/R:



1. Grouped Bar Chart Trend Analysis:
This chart shows a clear upward trend in the number of highly rated movies across all genre combinations over time.

> Specifically…
> Comedy/Romance: Saw the highest increase across decades: from 215 (1991–2000) to 590 (2011–2020).

This statistic suggests a major, growing tendency in production or taste for sentimental, lighthearted entertainment (comedy/romance).

> Adventure/Drama:
> Also experienced a significant rise, from 56 to 343.

This statistic also shows that movies within this genre that are daring and emotionally intense have grown in popularity.

Action/Thriller:

This statistic demonstrates that, while beginning with less high-rated films (55), it has quadrupled in the last ten years to 208.

Suggests a surge in well-crafted action thrillers, especially post-2010.

Inference #1 - There is an ongoing increase in all genre combinations, which could be due to greater audience targeting, higher-quality production, or even wider distribution made possible by digital platforms (based on bar chart):

The number of highly rated films in each of the three genre combinations (Comedy/Romance, Adventure/Drama, and Action/Thriller) increased consistently and noticeably over the three decades shown in the grouped bar chart. This pattern most likely results from a various number of changing audience habits and industry changes, including:

1. Increased Audience Targeting:

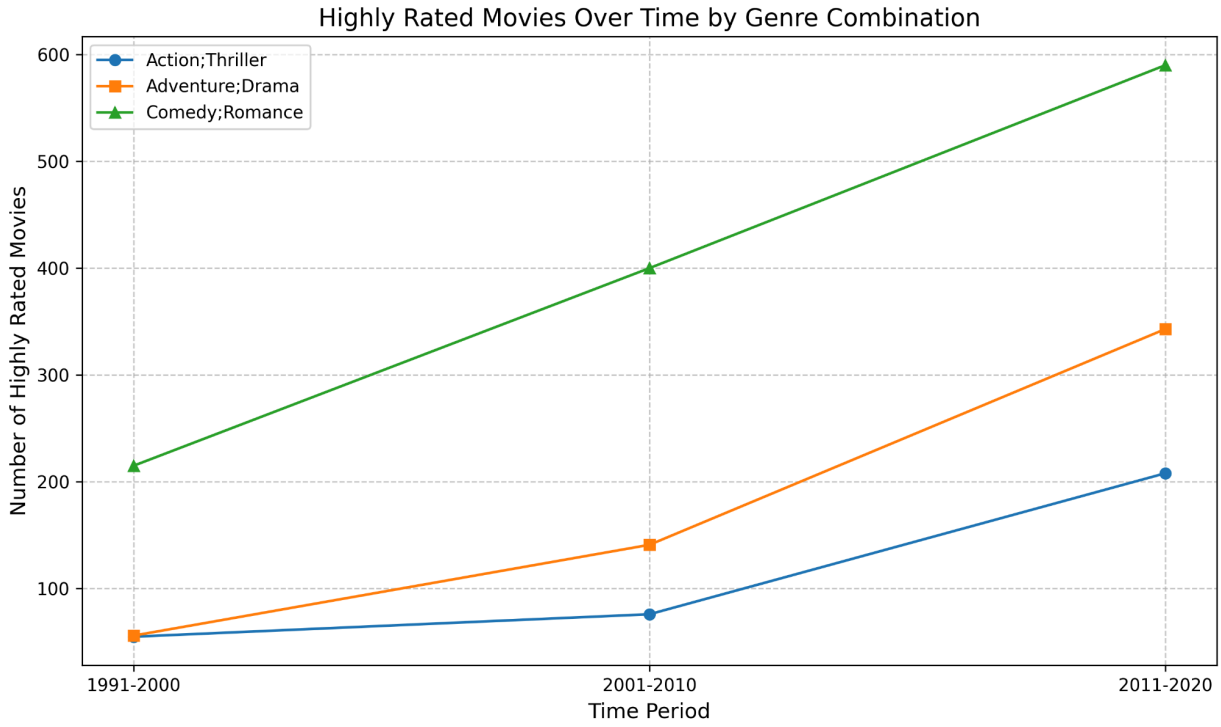
The movie business is now more data-driven, using analytics to find out what viewers want and create material that appeals to particular audience groups. In order to increase consumer satisfaction and engagement, streaming services like Netflix, Prime Video, and Hulu frequently make efforts in genre combinations (such as action + thriller, comedy + romance) depending on user viewing history.

2. Improved Production:

Particularly in genres like Action/Thriller and Adventure/Drama, the quality of films has increased due to advancements in filmmaking technology (visual effects, sound design, and cinematography). Increased funding for genre-bending movies has made it possible for greater casts, better screenplays, and better storytelling—all of which frequently translate into higher earnings at the box office.

3. More Distribution through Digital Platforms:

Today's films may rapidly reach audiences around the world through streaming, in contrast to times past when movie theaters were the main/only choice. Now, even niche or genre-blended movies can find a market, increasing their visibility, reviews, and likelihood of getting good ratings.



2. Line Chart Trend Analysis: The growth acceleration between decades is most dramatic in the final period (2011–2020), highlighting a tipping point in genre evolution and audience engagement:

Comedy/Romance:

Dominates every decade and displays the highest climb.

Could be a reflection of the increased interest for romantic comedies around the world or of social tendencies toward more relatable, human stories.

Adventure/Drama:

The sharp spike between 2001–2010 and 2011–2020 corresponds with the rise in popularity of epics or fantasy dramas during that time period (for example, films such as The Lord of the Rings and Game of Thrones).

Action/Thriller:

Shows the slowest growth out of all the genres at first, but it has increased significantly during the past ten years. This could be a reflection of the popularity of high-budget movie series (such as Marvel movies, John Wick, and Mission Impossible).

Inference #2: The line chart displays a sharp rise trend over the past ten years for all genre combinations in addition to increases in the quantity of highly rated films. This steeper slope suggests a turning point in the industry and audience trends:

1. Higher Production Value (More Cinematic Experiences):
Movies in the 2010s started to more closely resemble blockbuster standards in every genre. Comedy and romance, two genres that have historically had lower budgets, began to receive better production teams and more investments, which improved their quality and ratings.

2. Genre Innovation & Blending:
The 2010s saw a creative blending of genres, in contrast to previous decades, including:

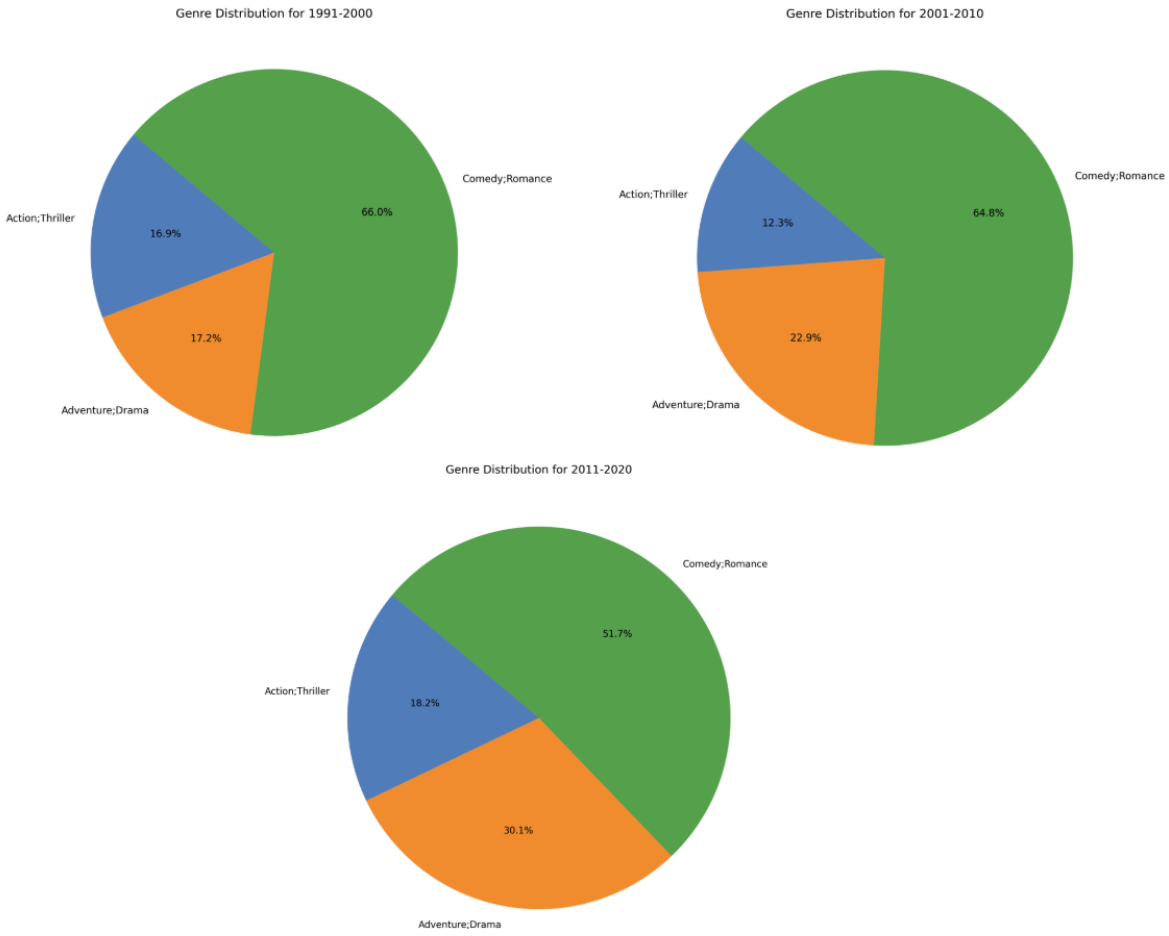
- Sci-fi-inspired romantic comedies (for example: 2013 movie Her)
- Dramas that combine elements of fantasy and adventure (like The Shape of Water)

As newer methods for storytelling drew in larger audiences, this combination probably helped explain the spike in positive feedback.

3. Streaming Platforms as Genre Catalysts:
By making major investments in genre-diverse programming of their own, streaming services like Netflix, HBO, and Amazon Prime created space for unusual themes to succeed. Movies like Bird Box and Marriage Story or television series like Stranger Things showed how non-theatrical releases continued to influence conversation and ratings.

4. Global Accessibility & Cultural Relevance:
The 2010s saw a real globalization of media, with dubbed and subtitled content becoming commonplace, opening up new audiences for genres like foreign thrillers and K-dramas (romance).

This worldwide accessibility probably increased viewership and ratings, particularly for films that were universally exciting or emotionally impactful.



3. Pie Charts (per period) Trend Analysis:

1991–2000:
Comedy/Romance dominates with ~69% of the total highly rated movies. Adventure/Drama and Action/Thriller are nearly equal, each contributing around 17% and 18% respectively. This suggests a strong audience preference for feel-good, romantic content during this decade.

2001–2010:
Comedy/Romance still leads, but its share drops slightly to ~63%. Adventure/Drama rises significantly, now accounting for ~22%. Action/Thriller sees a modest increase to ~12%. This period shows a growing interest in adventurous and dramatic narratives while romance-comedy maintains mass appeal.

2011–2020:
Comedy/Romance remains the leader but drops to ~50% of the share. Adventure/Drama makes a big leap to ~29%. Action/Thriller also climbs to ~18%. This indicates a diversifying taste in

movie genres. While romantic comedies still hold strong, audiences are increasingly engaging with intense action and dramatic adventures.

Inference #3 (based on Pie Charts): Audience preferences have changed over the years, there has been a noticeable proportional shift in genre choice. Adventure/Drama has gradually gained ground, although Comedy/Romance continues to dominate.

1. Romance and Comedy: Still Popular but Declining
By the 2010s, this genre had fallen from a huge ~69% share in the 1990s to about 50%. The drop is a sign that other genres are catching up, not that the films are becoming less good or appealing—in fact, the total number of films went up. This might imply:

- More varied emotional experiences—not just humorous or romantic stories—are what audiences are hankering about.
- More ambitious action and drama storytelling formats may be posing a greater threat to the genre.

2. Adventure/Drama: It's Rising!
Nearly doubled its proportional share, rising from around 17% in 1991–2000 to over 29% by 2011–2020. This consistent increase suggests a wider cultural need for compelling, character-driven narratives. Potential drivers:

- The popularity of lengthy narratives in epic sagas, such as Game of Thrones, The Lord of the Rings, and Harry Potter.
- The increasing interest of audiences in emotionally charged stories that examine moral complexity, resiliency, and development.

3. Action/Thriller: More Respected Despite Stability
Action/Thriller increases from about 12% to about 18%, although its growth isn't as sharp. Because of series like John Wick, The Dark Knight, Mission Impossible, and others, the genre is currently receiving high ratings and critical acclaim, suggesting that it is evolving beyond simple entertainment. Most likely impacted by:

- Increased production values
- Character depth and creative writing
- Intercultural appeal and global box office success

TASK 2 - SQL:

English version of Sql query:

The objective of the following query is to identify the top 5 highest-rated movies released between 2011 and 2020 that belong to both the "Action" and "Thriller" genres. To ensure relevance and quality (correct conditions), the query only considers:

- Movies (titleType = 'movie')
- That have received at least 150,000 votes
- With an average rating of 7.0 or higher
- And includes the lead actor or actress for each movie, identified as the person with ordering = 1 in the title_principals table
- The results are sorted in descending order of average rating, and only the top 5 movies are returned.

SQL Query:

```
SET ECHO ON;
SET LINESIZE 200;
SET PAGESIZE 100;
COLUMN title FORMAT A35;
COLUMN lead_actor_or_actress FORMAT A25;
SPOOL 4331-5331_Proj3Spring25_team_6.sql

SELECT
  b.primarytitle AS title,
  r.averagerating,
  n.primaryname AS lead_actor_or_actress
FROM
  imdb00.title_basics b
JOIN
  imdb00.title_ratings r ON b.tconst = r.tconst
JOIN
  imdb00.title_principals p ON b.tconst = p.tconst
JOIN
  imdb00.name_basics n ON p.nconst = n.nconst
WHERE
  b.titletype = 'movie'
  AND b.startyear BETWEEN 2011 AND 2020
  AND LOWER(b.genres) LIKE '%action%'
  AND LOWER(b.genres) LIKE '%thriller%'
  AND r.numvotes >= 150000
  AND r.averagerating >= 7.0
```

```
  AND p.ordering = '1'
ORDER BY
  r.averagerating DESC
FETCH FIRST 5 ROWS ONLY;
```

SQL Query Output:

```
TITLE                              AVERAGERATING   LEAD_ACTOR_OR_ACTRESS
—------------------------------  —-------------  ---------------------------
Skyfall                                      7.8     Daniel Craig
Mission: Impossible - Fallout                7.7     Tom Cruise
The Raid: Redemption                         7.6     Iko Uwais
Train to Busan                               7.6     Gong Yoo
Upgrade                                      7.5     Logan Marshall-Green
```

EXPLAIN Query plan:

```
EXPLAIN PLAN FOR
SELECT
  b.primarytitle AS title, r.averagerating, n.primaryname AS
lead_actor_or_actress
FROM
  imdb00.title_basics b
JOIN
  imdb00.title_ratings r ON b.tconst = r.tconst
JOIN
  imdb00.title_principals p ON b.tconst = p.tconst
JOIN
  imdb00.name_basics n ON p.nconst = n.nconst
WHERE
  b.titletype = 'movie'
  AND b.startyear BETWEEN 2011 AND 2020
  AND LOWER(b.genres) LIKE '%action%'
  AND LOWER(b.genres) LIKE '%thriller%'
  AND r.numvotes >= 150000
  AND r.averagerating >= 7.0
  AND p.ordering = '1'
ORDER BY
  r.averagerating DESC
FETCH FIRST 5 ROWS ONLY;

--then run this to see results of the explanation:
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

EXPLAIN Query plan output:

```
-------------------------------------------------------------------------------
----
| Id  | Operation                  | Name             | Rows | Bytes | Cost |
Time |
-------------------------------------------------------------------------------
----
|  0  | SELECT STATEMENT           |                  |   5  | 10215 | 133K |
00:00:06 |
|* 1  |  VIEW                      |                  |   5  | 10215 | 133K |
00:00:06 |
|* 2  |   WINDOW SORT PUSHED RANK  |                  |   6  | 1080  | 133K |
00:00:06 |
|  3  |    NESTED LOOPS            |                  |   6  | 1080  | 133K |
00:00:06 |
|  4  |     NESTED LOOPS           |                  |   6  | 1080  | 133K |
00:00:06 |
|  5  |      HASH JOIN             |                  |   6  |  840  | 133K |
00:00:06 |
|  6  |       NESTED LOOPS         |                  |   4  |  460  | 2614 |
00:00:01 |
|  7  |        TABLE ACCESS FULL | TITLE_RATINGS    | 657  | 11169 | 1084|
00:00:01 |
|  8  |        INDEX UNIQUE SCAN | SYS_C00547784    |   1  |       |     |
00:00:01 |
|* 9  |       TABLE ACCESS BY INDEX ROWID | TITLE_BASICS | 1 | 98   |   2 |
00:00:01 |
|*10  |     TABLE ACCESS FULL    | TITLE_PRINCIPALS | 8191K| 195M  | 131K|
00:00:06 |
| 11  |      INDEX UNIQUE SCAN   | SYS_C00547785    |   1  |       |     |
00:00:01 |
|*12  |     TABLE ACCESS BY INDEX ROWID | NAME_BASICS | 1 | 40   |   2 |
00:00:01 |
-------------------------------------------------------------------------------
----

Predicate Information (identified by operation id):
---------------------------------------------------
  1 - filter("from$_subquery$_008"."rowlimit_$$_rownumber"<=5)
  2 - filter(ROW_NUMBER() OVER (ORDER BY
INTERNAL_FUNCTION("r"."averagerating") DESC) <=5)
  5 - access("b"."tconst"="r"."tconst")
  6 - filter("r"."numvotes">=150000 AND "r"."averagerating">=7.0)
  9 - access("b"."tconst"="r"."tconst")
 10 - filter("b"."titletype"='movie' AND TO_NUMBER("b"."startyear")>=2011 AND
             TO_NUMBER("b"."startyear")<=2020 AND
```

```
             LOWER("b"."genres") LIKE U'%action%' AND
             LOWER("b"."genres") LIKE U'%thriller%')
 11 - filter("p"."ordering"='1')
 12 - access("p"."nconst"="n"."nconst")

34 rows selected.
```

Discussion in how the query plan is evaluated:

The SQL query is designed to retrieve the top 5 highest-rated Action/Thriller movies released between 2011 and 2020 that have at least 150,000 user votes and a rating of 7.0 or higher. Additionally, it ensures that only the lead actor or actress (based on ordering = '1') is returned alongside each title. The query execution plan, determined using EXPLAIN PLAN FOR and viewed using SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());, reveals how Oracle evaluates and optimizes the execution.

Oracle begins by wrapping the query in a view and applying a window function using WINDOW SORT PUSHED RANK. This allows the database to assign row numbers based on descending average ratings and apply the FETCH FIRST 5 ROWS ONLY (to identify the top 5 movies) constraint. The optimizer then uses a combination of nested loops and a hash join to process joins between the four required tables: title_basics, title_ratings, title_principals, and name_basics. The join operations are carefully chosen to minimize cost, especially with the use of foreign keys like tconst and nconst.

To further improve efficiency, Oracle performs a full table scan on title_ratings to filter rows early based on numvotes >= 150000 and averagerating >= 7.0. Additional filtering occurs through genre matching using LOWER(b.genres) LIKE '%action%' AND '%thriller%', makes sure both genres are present in the movie's genre list. Other predicates such as titletype = 'movie' and startyear BETWEEN 2011 AND 2020 are applied to narrow down the dataset. The title_principals table is filtered with ordering = '1' to extract only the lead actor/actress. Finally, Oracle uses an INDEX UNIQUE SCAN to efficiently retrieve the actor names from name_basics using the nconst key.

File Descriptions

MapReduce.java

- We created this Java file to apply the full implementation of the MapReduce logic for Task 1. The Mapper function processes the IMDb dataset line by line, filtering out only "movie" entries that are highly rated (rating ≥ 7.0) and fall within three specific genre combinations: Comedy/Romance, Adventure/Drama, and Action/Thriller. It also categorizes movies into one of three 10-year time periods (1991–2000, 2001–2010, 2011–2020). The Reducer function then aggregates the counts of qualifying movies for each unique genre-period key. The output is used to analyze genre popularity trends over time (which we get into in Analysis Results).

input_sample.txt

- We also created a sample input file used to manually verify the correctness of the MapReduce program. It contains a small and manageable subset of IMDb-formatted entries with fields such as titleType, startYear, genres, and ratings. The entries were chosen to cover different combinations of genres and years to test all filtering criteria in the Mapper. We manually counted and validated each output to ensure the logic and grouping were functioning correctly and as intended.

4331-5331_Proj3Spring25_6.sql

- This SQL script implements Task 2 by querying the IMDb Oracle database to find the top 5 highest-rated Action/Thriller movies from 2011–2020 with at least 150,000 votes and a rating ≥ 7.0. It includes the lead actor/actress for each movie and captures the query plan using EXPLAIN PLAN. The file also contains formatting commands and an English query description, with all output spooled as required.

plot1.py

- Generates a grouped bar chart showing the count of highly rated movies per genre (Comedy/Romance, Adventure/Drama, Action/Thriller) across three time periods. Saves the chart as a PNG file.

plot2.py

- Creates a line graph to visualize how each genre's count of highly rated movies changes over the three time periods. Saves the chart as a PNG file.

plot3.py

- Generates three pie charts (one per decade) showing genre share distribution among highly rated movies. Outputs each chart together as one PNG file with proper label spacing.

Project_Report.docx / Google Doc

- We created this written report compiling the overall explanation, design approach, and outcomes of the project. It includes detailed analysis for both Task 1 and Task 2, with SQL query results, formatted outputs, and EXPLAIN PLAN interpretation. In addition to describing the process and technical steps, the report provides trend analysis supported by visualizations like bar charts, line graphs, and pie charts. We then conclude this document with project reflections, challenges encountered, and insights into genre trends in highly rated movies.

Division of Labor

We divided the workload for this project based on each member's strengths and the nature of the tasks. Our team held several meetings outside class to clarify the project goals, understand the dataset and code structure, and assign responsibilities accordingly.

Dev Patel: Led the implementation of the MapReduce logic in Java, including the Mapper and Reducer functions, and created all data visualizations (bar, pie, and line charts). He also coordinated testing and contributed to documenting Task 1. Total hours spent: **20 hours.**

Yahia Elsaad: Focused on Task 2 by developing the SQL query to extract the top 5 Action-Thriller movies and analyzing the EXPLAIN PLAN results. He also cleaned and formatted the output, ensured consistency, and helped finalize the report. Total hours spent: **18 hours.**

Mohamad Nabih Alkhateeb: Set up the Hadoop environment and helped the team resolve configuration issues. He contributed to writing and debugging the MapReduce logic and assisted during the SQL phase and report editing. Total hours spent: **17 hours.**

All progress was tracked and shared via GitHub, which allowed us to collaborate efficiently and stay updated on each other's contributions.