**Project 1**

Analysis of Top Rated IMDB Movies vs The top 500 Most Profitable Hollywood Movies

Taken from Kaggle August 14, 2023

Sources: <https://www.kaggle.com/datasets/joebeachcapital/top-500-hollywood-movies-of-all-time>

<https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows>

Stakeholders: movie producers, aspiring filmmakers

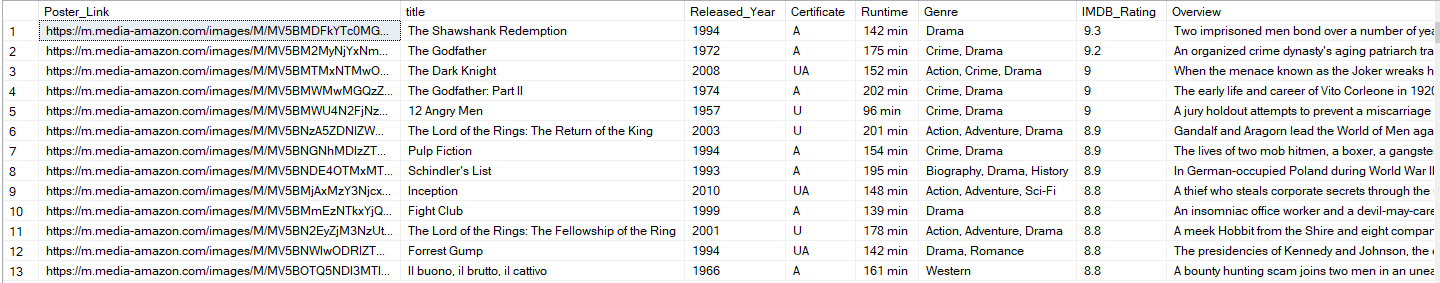
What genres are most popular?

How can user ratings be maximized?

What can I expect from my low budget? Or a high budget?

PART 1: EDA

1. Analyze ‘IMDBTop1000’ table



1. Count the number of movies for each genre and count how many unique genres are there. Some movies have multiple genres so string\_split was used

SELECT TRIM(value) AS genre\_value, COUNT(\*) AS movie\_count

FROM (

SELECT LTRIM(RTRIM(value)) AS value

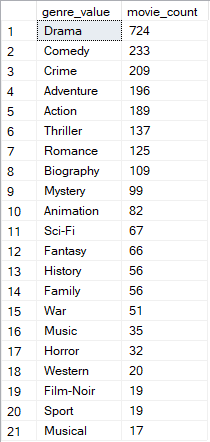
FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genres

GROUP BY TRIM(value)

ORDER BY movie\_count DESC;



There are 21 unique genres in this table and the most occurring genre is Drama with 724.

1. Get the minimum, maximum and average rating of all the movies in the table (IMDB\_Rating)

SELECT MIN(IMDB\_Rating) AS min\_rating, MAX(IMDB\_Rating) AS max\_rating, AVG(IMDB\_Rating) AS average\_rating

FROM IMDBTop1000;



Min, max, and average released\_year (rounded)

SELECT MIN(Released\_Year) AS min\_year,

MAX(Released\_Year) AS max\_year,

ROUND(AVG(Released\_Year), 0) AS avg\_year

FROM IMDBTop1000;



Min, max, and average runtime in minutes with trimming of number and conversions because Runtime is in nvarchar

SELECT MIN(CONVERT(INT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS min\_runtime,

MAX(CONVERT(INT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS max\_runtime,

AVG(CONVERT(FLOAT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS avg\_runtime

FROM IMDBTop1000;



Min, max, and average meta\_score (metascore is reviews from Metacritic by reviewers compared to IMDB’s reviews by users)

SELECT

MIN(Meta\_score) AS Min\_Meta\_score,

MAX(Meta\_score) AS Max\_Meta\_score,

AVG(CAST(Meta\_score AS FLOAT)) AS Avg\_Meta\_score

FROM IMDBTop1000;



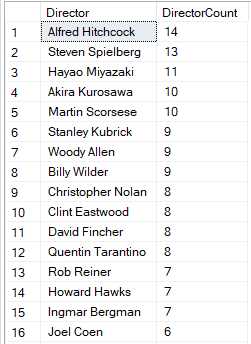
1. Count how many times each director appeared on this list

SELECT Director, COUNT(\*) AS DirectorCount

FROM IMDBTop1000

GROUP BY Director

ORDER BY DirectorCount DESC;



1. Count how many times each star appeared on this table from all 4 columns 'star1', 'star2', 'star3', 'star4'

SELECT Star1, COUNT(\*) AS StarCount

FROM (

SELECT Star1 FROM IMDBTop1000

UNION ALL

SELECT Star2 FROM IMDBTop1000

UNION ALL

SELECT Star3 FROM IMDBTop1000

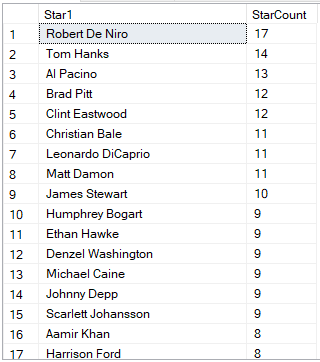
UNION ALL

SELECT Star4 FROM IMDBTop1000

) AS AllStars

GROUP BY Star1

ORDER BY StarCount DESC;



**PART 2: Insights**

1. Which genres have the highest IMDB rating? And how many movies are in each genre?

SELECT

TRIM(value) AS genre\_value,

COUNT(\*) AS movie\_count,

AVG(IMDB\_Rating) AS average\_rating

FROM (

SELECT LTRIM(RTRIM(value)) AS value, IMDB\_Rating

FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genres

GROUP BY TRIM(value)

ORDER BY average\_rating DESC;



1. Min, Max, and Average number of votes

SELECT

MIN(No\_of\_Votes) AS MinVotes,

MAX(No\_of\_Votes) AS MaxVotes,

AVG(No\_of\_Votes) AS AvgVotes

FROM IMDBTop1000;



--Vote number range and average rating split into 2

SELECT

CASE

WHEN No\_of\_Votes >= 25088 AND No\_of\_Votes < 273693 THEN '25088-273693'

WHEN No\_of\_Votes >= 273693 AND No\_of\_Votes <= 2343110 THEN '273693-2343110'

ELSE 'Other'

END AS VoteRange,

AVG(IMDB\_Rating) AS AverageRating

FROM IMDBTop1000

GROUP BY

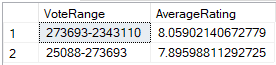
CASE

WHEN No\_of\_Votes >= 25088 AND No\_of\_Votes < 273693 THEN '25088-273693'

WHEN No\_of\_Votes >= 273693 AND No\_of\_Votes <= 2343110 THEN '273693-2343110'

ELSE 'Other'

END;



--Vote number range and average rating split into 10

SELECT

CASE

WHEN No\_of\_Votes >= 25088 AND No\_of\_Votes < 256890 THEN '25088-256890'

WHEN No\_of\_Votes >= 256890 AND No\_of\_Votes < 488692 THEN '256890-488692'

WHEN No\_of\_Votes >= 488692 AND No\_of\_Votes < 720495 THEN '488692-720495'

WHEN No\_of\_Votes >= 720495 AND No\_of\_Votes < 952296 THEN '720495-952296'

WHEN No\_of\_Votes >= 952296 AND No\_of\_Votes < 1180099 THEN '952296-1180099'

WHEN No\_of\_Votes >= 1180099 AND No\_of\_Votes < 1407901 THEN '1180099-1407901'

WHEN No\_of\_Votes >= 1407901 AND No\_of\_Votes < 1635703 THEN '1407901-1635703'

WHEN No\_of\_Votes >= 1635703 AND No\_of\_Votes < 1863506 THEN '1635703-1863506'

WHEN No\_of\_Votes >= 1863506 AND No\_of\_Votes < 2091308 THEN '1863506-2091308'

WHEN No\_of\_Votes >= 2091308 AND No\_of\_Votes <= 2343110 THEN '2091308-2343110'

ELSE 'Other'

END AS VoteRange,

AVG(IMDB\_Rating) AS AverageRating

FROM IMDBTop1000

GROUP BY

CASE

WHEN No\_of\_Votes >= 25088 AND No\_of\_Votes < 256890 THEN '25088-256890'

WHEN No\_of\_Votes >= 256890 AND No\_of\_Votes < 488692 THEN '256890-488692'

WHEN No\_of\_Votes >= 488692 AND No\_of\_Votes < 720495 THEN '488692-720495'

WHEN No\_of\_Votes >= 720495 AND No\_of\_Votes < 952296 THEN '720495-952296'

WHEN No\_of\_Votes >= 952296 AND No\_of\_Votes < 1180099 THEN '952296-1180099'

WHEN No\_of\_Votes >= 1180099 AND No\_of\_Votes < 1407901 THEN '1180099-1407901'

WHEN No\_of\_Votes >= 1407901 AND No\_of\_Votes < 1635703 THEN '1407901-1635703'

WHEN No\_of\_Votes >= 1635703 AND No\_of\_Votes < 1863506 THEN '1635703-1863506'

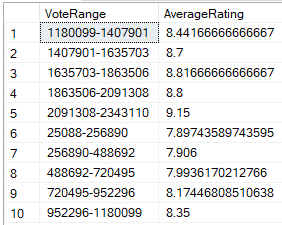
WHEN No\_of\_Votes >= 1863506 AND No\_of\_Votes < 2091308 THEN '1863506-2091308'

WHEN No\_of\_Votes >= 2091308 AND No\_of\_Votes <= 2343110 THEN '2091308-2343110'

ELSE 'Other'

END

ORDER BY VoteRange;



1. Min, max, and average runtime in minutes with trimming of number and conversions because Runtime is in nvarchar

SELECT MIN(CONVERT(INT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS min\_runtime,

MAX(CONVERT(INT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS max\_runtime,

AVG(CONVERT(FLOAT, LEFT(runtime, CHARINDEX(' ', runtime) - 1))) AS avg\_runtime

FROM IMDBTop1000;



--runtime ranges and the average rating of each split into 3

SELECT

CASE

WHEN runtime\_int >= 45 AND runtime\_int <= 137 THEN '45-137'

WHEN runtime\_int > 137 AND runtime\_int <= 229 THEN '138-229'

WHEN runtime\_int > 229 AND runtime\_int <= 321 THEN '230-321'

ELSE 'Other'

END AS RuntimeRange,

AVG(IMDB\_Rating) AS AverageRating

FROM (

SELECT

CAST(SUBSTRING(Runtime, 1, PATINDEX('%[^0-9]%', Runtime) - 1) AS INT) AS runtime\_int,

IMDB\_Rating

FROM IMDBTop1000

) AS RuntimeData

GROUP BY

CASE

WHEN runtime\_int >= 45 AND runtime\_int <= 137 THEN '45-137'

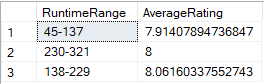
WHEN runtime\_int > 137 AND runtime\_int <= 229 THEN '138-229'

WHEN runtime\_int > 229 AND runtime\_int <= 321 THEN '230-321'

ELSE 'Other'

END

ORDER BY AverageRating;



1. Top rated movie in each genre with its corresponding Metascore

WITH TopRatedMovies AS (

SELECT

genre\_value,

MAX(title) AS top\_rated\_movie,

MAX(IMDB\_Rating) AS max\_rating

FROM (

SELECT TRIM(value) AS genre\_value,

title,

IMDB\_Rating,

ROW\_NUMBER() OVER (PARTITION BY TRIM(value) ORDER BY IMDB\_Rating DESC) AS rn

FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genre\_movies

WHERE rn = 1

GROUP BY genre\_value

),

TopRatedMoviesWithMetaScore AS (

SELECT

TRM.genre\_value AS genre,

TRM.top\_rated\_movie,

TRM.max\_rating AS max\_rating,

IMDB.Meta\_score AS meta\_score

FROM TopRatedMovies AS TRM

LEFT JOIN IMDBTop1000 AS IMDB ON TRM.top\_rated\_movie = IMDB.title

)

SELECT

genre,

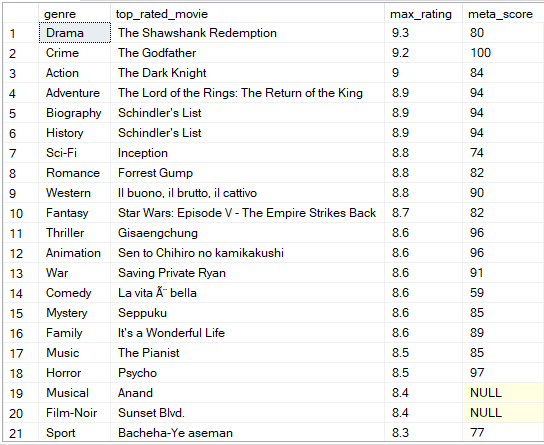
top\_rated\_movie,

max\_rating,

meta\_score

FROM TopRatedMoviesWithMetaScore

ORDER BY max\_rating DESC;



A movie like Schindler’s List for instance appeared on this list twice

--Top meta score for each genre with its corresponding IMDB\_rating

WITH TopMetaScoreMovies AS (

SELECT

genre\_value,

MAX(title) AS top\_meta\_score\_movie,

MAX(Meta\_score) AS max\_meta\_score

FROM (

SELECT TRIM(value) AS genre\_value,

title,

Meta\_score,

ROW\_NUMBER() OVER (PARTITION BY TRIM(value) ORDER BY Meta\_score DESC) AS rn

FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genre\_movies

WHERE rn = 1

GROUP BY genre\_value

),

TopMetaScoreMoviesWithIMDBRating AS (

SELECT

TMM.genre\_value AS genre,

TMM.top\_meta\_score\_movie,

TMM.max\_meta\_score AS max\_meta\_score,

IMDB.IMDB\_Rating AS IMDB\_Rating

FROM TopMetaScoreMovies AS TMM

LEFT JOIN IMDBTop1000 AS IMDB ON TMM.top\_meta\_score\_movie = IMDB.title

)

SELECT

genre,

top\_meta\_score\_movie,

max\_meta\_score,

IMDB\_Rating

FROM TopMetaScoreMoviesWithIMDBRating

ORDER BY max\_meta\_score DESC;



1. Top Rated and top Meta Score for each genre side by side

WITH TopRatedMovies AS (

SELECT

genre\_value,

MAX(title) AS top\_rated\_movie,

MAX(IMDB\_Rating) AS max\_rating

FROM (

SELECT TRIM(value) AS genre\_value,

title,

IMDB\_Rating,

ROW\_NUMBER() OVER (PARTITION BY TRIM(value) ORDER BY IMDB\_Rating DESC) AS rn

FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genre\_movies

WHERE rn = 1

GROUP BY genre\_value

),

TopMetaScoreMovies AS (

SELECT

genre\_value,

MAX(title) AS top\_meta\_score\_movie,

MAX(Meta\_score) AS max\_meta\_score

FROM (

SELECT TRIM(value) AS genre\_value,

title,

Meta\_score,

ROW\_NUMBER() OVER (PARTITION BY TRIM(value) ORDER BY Meta\_score DESC) AS rn

FROM IMDBTop1000

CROSS APPLY STRING\_SPLIT(genre, ',')

) AS genre\_movies

WHERE rn = 1

GROUP BY genre\_value

)

SELECT

TRM.genre\_value AS genre,

TRM.top\_rated\_movie,

TRM.max\_rating,

TMM.top\_meta\_score\_movie,

TMM.max\_meta\_score

FROM TopRatedMovies AS TRM

INNER JOIN TopMetaScoreMovies AS TMM ON TRM.genre\_value = TMM.genre\_value

ORDER BY TRM.genre\_value;



**PART 3: THE TOP OF THE TOP** - analysis of the highly rated movies in IMDB, their budget, genre, rating, stars, number of votes, runtime, etc.

1. Rename all columns in the MostProfitable500 table to Pascal Case (table from: <https://www.kaggle.com/datasets/joebeachcapital/top-500-hollywood-movies-of-all-time>)

EXEC sp\_rename 'MostProfitable500.[worldwide gross (m)]', 'WorldwideGrossM', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[% budget recovered]', 'BudgetRecoveredPercent', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[X times budget recovered]', 'XTimesBudgetRecovered', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[budget (millions)]', 'BudgetMillions', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[domestic gross (m)]', 'DomesticGrossM', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[domestic %]', 'DomesticPercent', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[international gross (m)]', 'InternationalGrossM', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[% of gross from international]', 'PercentGrossFromInternational', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[worldwide gross]', 'WorldwideGross', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[year]', 'Year', 'COLUMN';

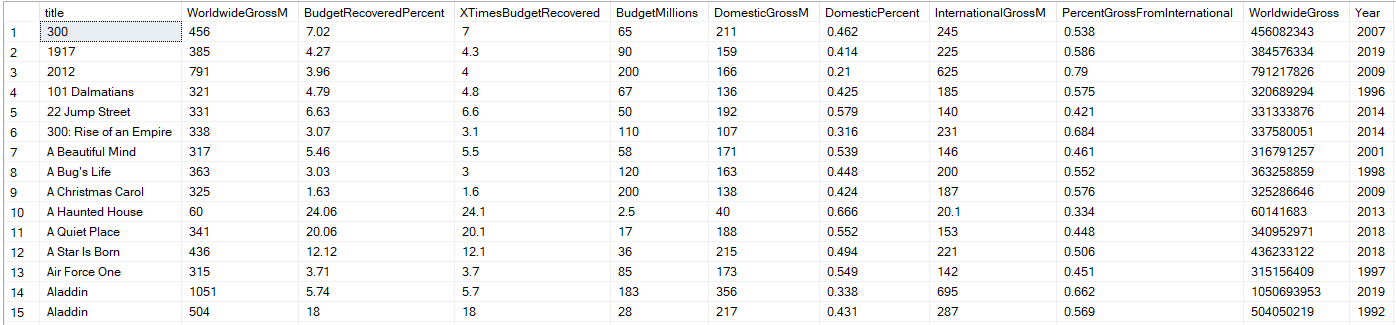
EXEC sp\_rename 'MostProfitable500.[decade]', 'Decade', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[source]', 'Source', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[budget source]', 'BudgetSource', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[force label]', 'ForceLabel', 'COLUMN';

EXEC sp\_rename 'MostProfitable500.[horror]', 'Horror', 'COLUMN';



1. Relate the 2 tables ‘IMDBTop1000’ and ‘MostProfitable500’ by joining the 2 on the title column with their similarities (e.g. the movie 300 exists on both tables) and excluding non quantifiable analytical data such as ‘overview’ and ‘poster\_link’. Ordered by IMDB\_rating first then by Meta\_score

SELECT t1.title,

t1.Released\_Year,

t1.Certificate,

t1.Runtime,

t1.Genre,

t1.IMDB\_Rating,

t1.Meta\_score,

t1.Director,

t1.Star1,

t1.Star2,

t1.Star3,

t1.Star4,

t1.No\_of\_Votes,

t2.WorldwideGrossM,

t2.BudgetRecoveredPercent,

t2.XTimesBudgetRecovered,

t2.BudgetMillions,

t2.DomesticGrossM,

t2.DomesticPercent,

t2.InternationalGrossM,

t2.PercentGrossFromInternational,

t2.WorldwideGross,

t2.Year,

t2.Decade,

t2.Source,

t2.Horror

FROM IMDBTop1000 AS t1

INNER JOIN MostProfitable500 AS t2 ON t1.title = t2.title

ORDER BY IMDB\_Rating DESC

SELECT t1.title,

t1.Released\_Year,

t1.Certificate,

t1.Runtime,

t1.Genre,

t1.IMDB\_Rating,

t1.Meta\_score,

t1.Director,

t1.Star1,

t1.Star2,

t1.Star3,

t1.Star4,

t1.No\_of\_Votes,

t2.WorldwideGrossM,

t2.BudgetRecoveredPercent,

t2.XTimesBudgetRecovered,

t2.BudgetMillions,

t2.DomesticGrossM,

t2.DomesticPercent,

t2.InternationalGrossM,

t2.PercentGrossFromInternational,

t2.WorldwideGross,

t2.Year,

t2.Decade,

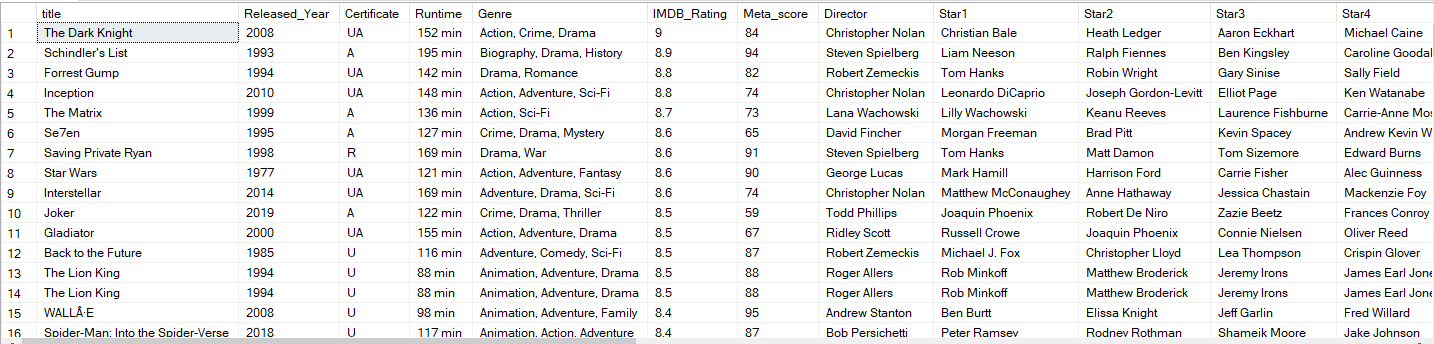
t2.Source,

t2.Horror

FROM IMDBTop1000 AS t1

INNER JOIN MostProfitable500 AS t2 ON t1.title = t2.title

ORDER BY IMDB\_Rating DESC



1. Top 20 movies in terms of Worldwide Gross. Times budget recovered and Budget(millions) ranks are also displayed

WITH Top20 AS (

SELECT \*,

RANK() OVER (ORDER BY worldwidegross DESC) AS worldwidegross\_rank,

RANK() OVER (ORDER BY xtimesbudgetrecovered DESC) AS xtimesbudgetrecovered\_rank,

RANK() OVER (ORDER BY budgetmillions DESC) AS budgetmillions\_rank

FROM MostProfitable500

)

SELECT

title, WorldwideGross, XTimesBudgetRecovered, BudgetMillions, worldwidegross\_rank, xtimesbudgetrecovered\_rank, budgetmillions\_rank

FROM Top20

WHERE worldwidegross\_rank <= 20

ORDER BY worldwidegross\_rank;

