RNCP Report

> ***Info:*** This report will take you through the journey of how a movie could win an Oscars' Best Picture Awards, from a newly mined Data Analyst who at some point in her life, also dreamt of being an actress.

Table of contents

📌 **Learning Highlights:**

- Exposure to new tools:
  - `copy`, `fuzzywuzzy`, `selenium`, `sqlalchemy`
- Test test test:
  - to avoid high cost at tiny error (e.g. 10 hours of scraping)
- Learning never ends:
  - `Tableau` is more interactive but not any easier than `matplotlib` or `seaborn`

📌 **Special Thanks:**

- Our teachers Kseniia, Rafaat, Andy & Angela for their knowledge and guidance
- The entire 1601 Data Analytics classmates for their friendship, support & suggestions

Introduction

▼ 📽️ Why Oscars?

Till date, I still remember back when I was learning English, I was searching for English movies to watch so I could learn English while having a break from study. There wasn't as much information as we have nowadays but IMDb was there and it provided me with information, ratings, and synopsis that helped me choose the movies, usually with highest ratings and reviews. When my English was getting better, I enjoyed reading trivia, quotes, and behind the scenes of my favourite movies.

With the movies chosen from IMDb, not only did I enjoy the cinematography but also did I learn many life lessons. For this reason, I often look out for the Oscars, as it is one of the best awards that could be given to outstanding films and individuals. It always amazes me that there are so many thoughts, layers, easter eggs, and hard work behind each movie that I learn to appreciate the art, the cast and the crew more and more.

In recent years, the Oscars is no longer just an awards show but it has also become a vehicle to advocate social movement, drawing attention to various topics such as climatic changes, diversity or social equality.

In this project, I would like to explore the quantitative relationship between the inherent characteristics of IMDb datasets and likely outcome to win an Oscars of

a movie. Certainly, as with many things in life, we need qualitative factors to explain events and I will add them where it is crucial to the case.

▼ 🤝 Business Use Case

Oscars, while it is an awards, it is a big part of the film-making industry where millions of people depend upon and billions of dollars are generated. If data can pinpoint the potential winners, many businesses will benefit from it in terms of box office, brand sponsorship, merchandise sales, and even decisions on future film collaboration, aside from the fact that it could also shed a light and drive social movements at times.

Data Collection

▼ 🔽 IMDb datasets

- IMDb has readily available datasets for academic researches and because of this, it does not contain aspects such as producers, budget and box-office numbers. However, it has average ratings, numbers of votes, principle crew and genres that would be interesting to interpret.
- Key Columns Descriptions:

Table Reference	Columns	Descriptions	Comments
actor_name	`nconst`	**unique ID** of individuals regardless of role (directors, actors, composers, etc)	**used** for flagging if the person has been nominated
	`primaryName`	individuals' name	
	`knownForTitles`	the top 4 works a person is known for	**used** for calculating the average ratings of their best 4 works/ titles

Table Reference	Columns	Descriptions	Comments
title_akas		names of the same movies in different languages	**not used** 90% of data (region/ languages) are 'Others'
title_basics	`tconst`	unique ID of each title regardless of type (movie, series, talk-show)	**used** as primary key for all derived tables of this project
	`startYear`	year of release	
	`runtimeMinutes`	length in minutes	
	`genres`	genres (1 -3 genres per title identified)	**used** as part of analysis
title_crew		directors of each title	**not used** as it is a subset of another table
title_prin	`tconst`	ID of title	*same as above*
	`nconst`	ID of principle crew	1 `tconst` has many `nconst`
	`category`	role of each individual	e.g. director, actor
title_rating	`tconst`	uniqueID of title	*same as above*
	`averageRating`	total rating in IMDB, weighted with votes count	**used** as part of analysis
	`numVotes`	total number of votes	**used** as part of analysis

▼ 🕸 Web Scraping

- Web Scraping was used to extract the historical records of:
  - Nominated & winning movies, including those who won for sound effects or costume design
  - Nominated & winning individuals, mainly directors, actors, actresses and composers
- Python code to scrape Oscars best pictures. Similar processes were used for actors, actress, composers (other categories such as costume designer, VFX

effects, etc were not exhaustive enough for full datasets and hence will skew the data if included)

```python
url_oscars_winner = 'https://www.imdb.com/list/ls055265443/?sort=release_date,desc&st_dt=&mode=simple&page=1&
ref_=ttls_vw_smp'
response_winner = requests.get(url_oscars_winner)
soup_winner = bs4.BeautifulSoup(response_winner.content, 'lxml')
```

Python

```python
titleId_winner = soup_winner.find_all('div', class_ = "col-title")
titleId_winner_block = [element for element in titleId_winner]
titleId_winner_list = [element.find('a')['href'][7:-1] for element in titleId_winner_block]
len(titleId_winner_list)
```

Python

```python
titleId_winner_list_full = []
for i in range(1, 9):
    url_oscars_winner = f'https://www.imdb.com/list/ls055265443/?sort=release_date,desc&st_dt=&mode=simple&page={i}&
    ref_=ttls_vw_smp'
    response_winner = requests.get(url_oscars_winner)
    soup_winner = bs4.BeautifulSoup(response_winner.content, 'lxml')
    titleId_winner = soup_winner.find_all('div', class_ = "col-title")
    titleId_winner_block = [element for element in titleId_winner]
    titleId_winner_list = [element.find('a')['href'][7:-1] for element in titleId_winner_block]
    titleId_winner_list_full += titleId_winner_list
```

Python

- Python code to clean scraping results and put them into a dataframe for later use

```python
oscars_winner_list = pd.DataFrame(titleId_winner_list_full)
oscars_winner_list.drop_duplicates()
oscars_winner_list.columns = ['tconst']
oscars_winner_list['Winner'] = 'TRUE'
oscars_winner_list.to_csv("./Data/oscars_winner_imdb_list.csv")
```

Python

- Key Columns Descriptions:

Table Reference	Columns	Descriptions
Nominee	`tconst`	imdb `tconst` ID of nominated title
	`Nominated`	True
Winner	`tconst`	imdb `tconst` ID of winning title
	`Winner`	True

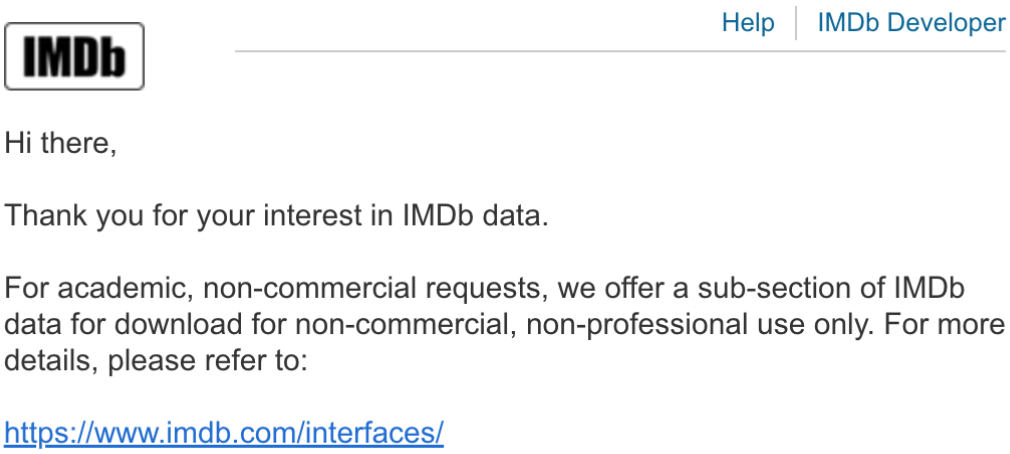
▼ 🔢 The Production House & Box Office Number

- Budget and financial numbers on movies would produce interesting insights as high investment could lead to hiring the best crew and cast. Same goes with production house for the style & influence. Unfortunately, within the timeframe and the information being exclusive, these have been proven

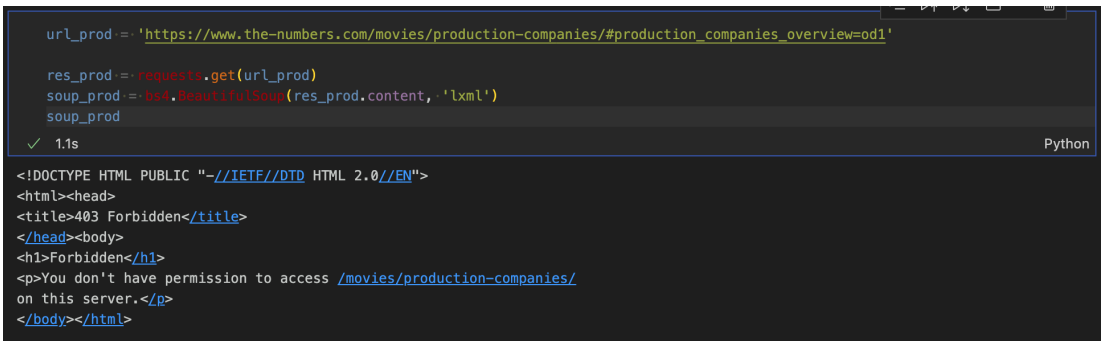
difficult to obtain. Other than that, the numbers will have to be adjusted for inflation and given the project timeline, this could probably require longer time

- One has to have IMDBPro account to obtain these numbers

- IMDb rejected my request for a developer account for box office numbers, even only for academic purpose



- Attempt to scrape from The Numbers are forbidden. Upon request, the data is payable at $475 for academic use



Data Cleaning

▼ ⚙️ Clean Null

- Python function to clean `\N` value for all applicable dataframes and change them into required data types for analysis & sample output:

```python
def clean_N(df, col, replace, type):
    """
    Replace \N value in a column, replace it with a value of choice and reset the type of the column
    """
    df[col] = df[col].replace('\\N', replace)
    df[col] = df[col].astype(type)
    return df[col].value_counts().sort_index()
```

✓ 0.0s Python

```python
clean_N(actor_name_cp, 'birthYear', 1000, 'int64')
clean_N(actor_name_cp, 'deathYear', 2023, 'int64')
```

Python

Outputs are collapsed ···

```python
# actor_name_cp.dtypes
```

Python

```
nconst              object
primaryName         object
birthYear            int64
deathYear            int64
primaryProfession   object
knownForTitles      object
dtype: object
```

▼ ✨ Drop Irrelevant Datas & Clean Invalid Entries

- Oscars started in 1927 and thus anything released before that are irrelevant

```python
actor_name_cp = actor_name_cp.rename({'deathYear':'lastSeen'}, axis=1)
actor_name_clean = actor_name_cp[(actor_name_cp['birthYear'] >= 1900) & (actor_name_cp['lastSeen'] > 1926)]
```

Python

- Oscars movies have to be more than 40 minutes so anything shorter were dropped. Column `runtimeMitutes` contained texts and these were removed manually as the invalid records were small (only 10 lines)

```python
title_basics['runtimeMinutes'].fillna(0, inplace=True)
title_basics[title_basics['runtimeMinutes'].str.contains('Animation').fillna(False)]
```

✓ 7.2s Python

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
2996552	tt13704268	tvEpisode	Bay of the Triffids/Doctor of Doom\tBay of the...	0	\N	1800	2024	Animation,Comedy,Family	NaN

- Titles data contained non-movies types such as Reality Show, Shorts, News, etc. These rows were also dropped

```python
movie_type = ['movie','tvMovie']
title_movies_clean = title_basics[((title_basics['endYear'] < 2023)
                                   | (title_basics['startYear'] > 1926))
                                  & (title_basics['titleType'].isin(movie_type))]
```

✓ 1.5s Python

▼ 💱 Split Genres & Encoding Genres

- Genres columns contain strings of genres types, ranging from 0 to 3 genres per title. I broke them down for further processing:

```
titleId_genres = title_movies_clean.genres.str.split(",", expand=True)
titleId_genres.columns = ['genres1','genres2','genres3']

titleId_basics_genres_sep = pd.concat([title_movies_clean, titleId_genres], axis=1)
titleId_basics_genres_sep.head()
```

✓ 3.0s Python

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	genres1
3816	tt0003854	movie	Dodge City Trail	Dodge City Trail	0	1936	2024	56	Adventure,Music,Western	Adventure
11551	tt0011715	movie	Sol de gloria	Sol de gloria	0	1928	2024	0	Drama,Romance	Drama
11636	tt0011801	movie	Tötet nicht mehr	Tötet nicht mehr	0	2019	2024	0	Action,Crime	Action
13082	tt0013274	movie	Istoriya grazhdanskoy voyny	Istoriya grazhdanskoy voyny	0	2021	2024	133	Documentary	Documentary

- Replacing `Nan` with `'None'` to ensure no empty cells for future processing and avoid SQL errors

```
titleId_basics_genres_sep = titleId_basics_genres_sep.fillna(np.nan).replace([np.nan], 'None')
titleId_basics_genres_sep[titleId_basics_genres_sep['genres3'] == 'None']
```

✓ 2.4s Python

titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	genres1	genres2	genres3
movie	Sol de gloria	Sol de gloria	0	1928	2024	0	Drama,Romance	Drama	Romance	None
movie	Tötet nicht mehr	Tötet nicht mehr	0	2019	2024	0	Action,Crime	Action	Crime	None
movie	Istoriya grazhdanskoy voyny	Istoriya grazhdanskoy voyny	0	2021	2024	133	Documentary	Documentary	None	None
movie	A Hero for a Night	A Hero for a Night	0	1927	2024	60	Comedy	Comedy	None	None
movie	Mustalaishurmaaja	Mustalaishurmaaja	0	1929	2024	69	Drama,Romance	Drama	Romance	None

- Since genres are text, I used `LabelEncoder()` to convert them to numbers. Encoded genres were then put into dataframe from array for final database creation

```
# LE from 1 list of all genres
le = LabelEncoder()
le.fit(all_genres)
encoded_genres1 = pd.DataFrame(le.transform(titleId_basics_genres_sep.genres1))
encoded_genres2 = pd.DataFrame(le.transform(titleId_basics_genres_sep.genres2))
encoded_genres3 = pd.DataFrame(le.transform(titleId_basics_genres_sep.genres3))
type(encoded_genres1)
```

✓ 0.9s Python

```
pandas.core.frame.DataFrame
```

```
genres_df = pd.concat([encoded_genres1, encoded_genres2, encoded_genres3], axis = 1)
genres_df.columns = ['encoded_genres1','encoded_genres2','encoded_genres3']
genres_df.head(30)
# genres_df.info()
```

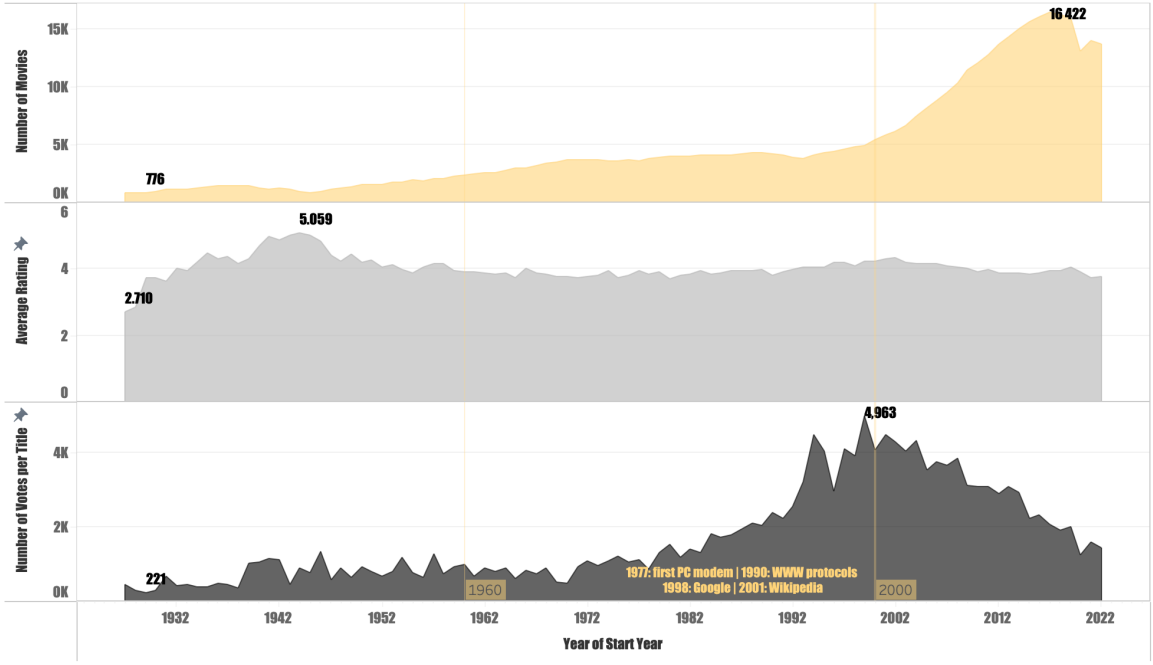
✓ 0.1s Python

	encoded_genres1	encoded_genres2	encoded_genres3
0	2	16	29
1	8	22	20
2	0	6	20
3	7	20	20

Data Exploration & Visualisation

▼ 📈 Trend

- Number of Movies: we see that while Oscars started way back in 1927, only till around 1960 that the number of movies slowly **picked up** to around 2300 movies/ year due to the advent of auto zoom camera equipment. Thereafter, it once again **picked up sharply** from 2000, after the arrival of personal PC modem, WWW sites, Google and Wikipedia where the public has easier access to media and hence more awareness to the industry. We also see a **dip** in 2020 (by around 3000 titles due to the Covid pandemic)
- Average Rating: generally the average rating stays **rather stable** though the year. However, it was higher before 1960, showing perhaps that the audience, as time goes, are given more choices/ options to watch, and hence are more refined and selective in their taste. It is also worth noting that the average here is about 3.9, which is very low because there are presence of movies without ratings. The **mode of average rating is 5.2**, meaning that most of movies are more than “average”.
- Number of Votes per Title: we saw that the number of movies **exploded over time**, and so did the number of votes per title. We can infer that the public has more access to the internet and are actually sending feedback to the movie, which is informative for the movies makers.

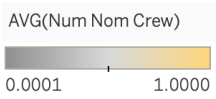


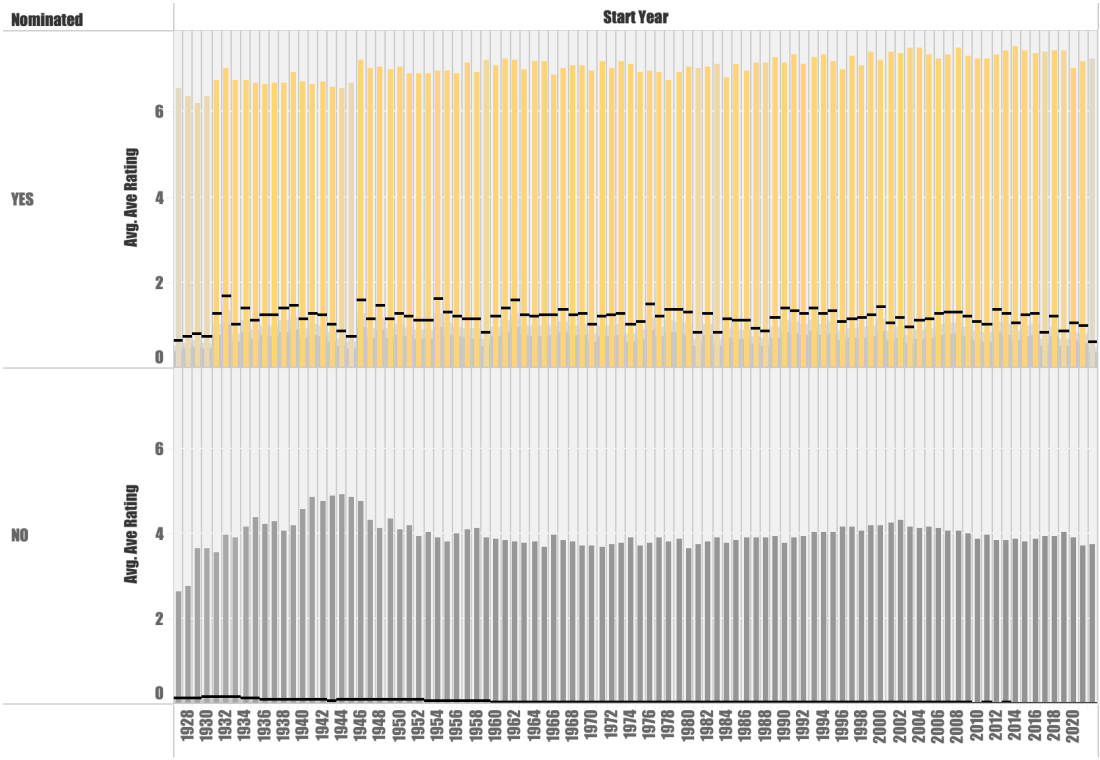
▼ 📍Nomination

- To win the Oscars, one first has to be nominated so I looked into the difference between the nominated titles and the ones that are not, in terms of their crew, as rather often, a winning title is the result of a strong team, talent & reputation wise.

▼ 👥 Nominated Crew

- A movie with crew that has track record of nomination might influent the public ratings and by force of the crowd, could be voted to win the Oscars.
- Comparing the number of nominated crew in a title, here we see that in general, **a nominated title will have at least 1 member that had been nominated herself/ himself (black horizontal bar)**, whereas for the non-nominated, the value is close to 0. Towards the present day, the average of nominated crew per title is almost non-existing, meaning perhaps, without a nominated crew in the team, it would be much harder to win. The same observation goes for the average rating (vertical bar) - non-nominated titles have average of 3.9, while the nominated ones is rated at 7.1



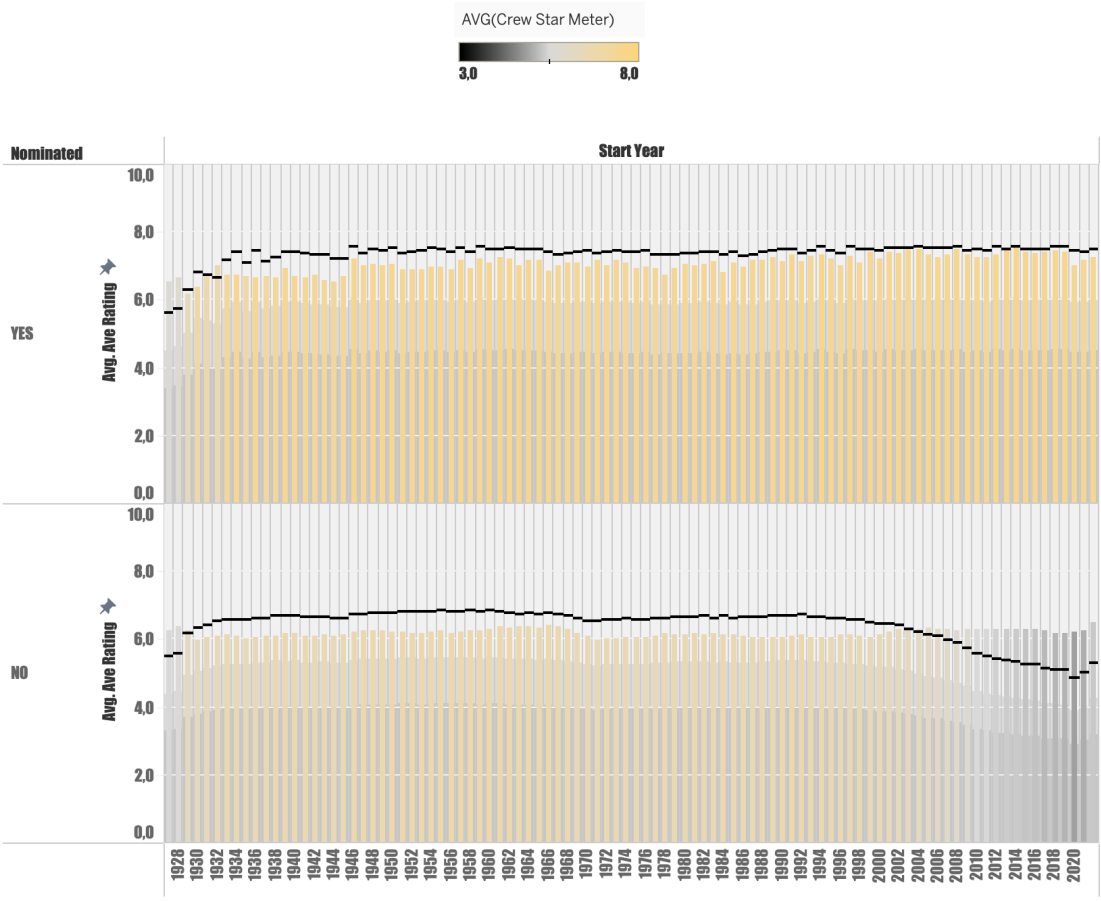


▼ ⭐ Star Meter

- Similar to having a nominated cast, the reputation of the crew would be a factor for the title to win. We are now going to look at the correlation between the average of ratings of movies based on what I called “star meter”, which depends on the principle crew of the movie, and it is derived by the average ratings of the top 4 movies the individual is known for.
- For example:
  - Given the titles & ratings of the name `nconst` = nm0000006 in the table below, the personal rating of nm0000006 is = 7.5 out of 4 titles ratings
  - The star meter of a movie title is then calculated as the average personal rating of the principle crew

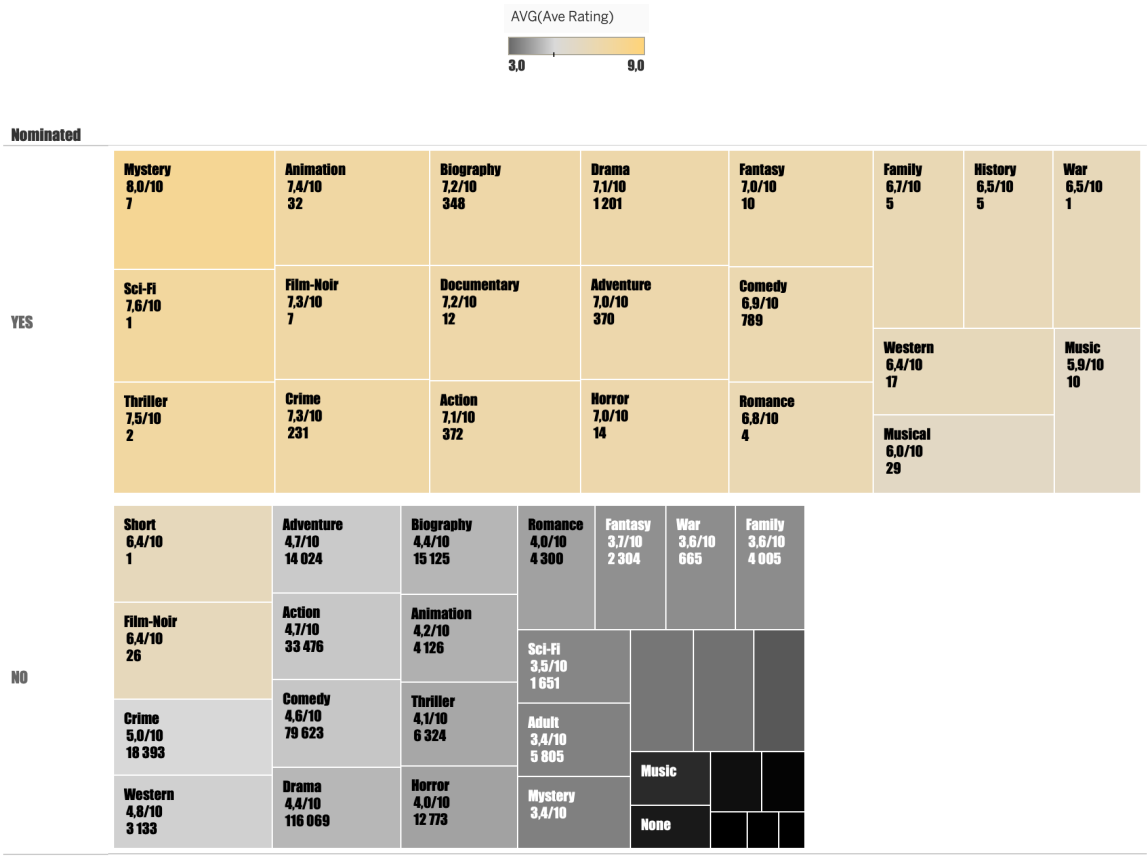
nconst (nameID)	titleKnownFor	average_rating
nm0000006	tt0038787	7
	tt0034583	8
	tt0038109	9
	tt0036855	6

- Comparing the star meters of the nominated vs non-nominated movies, we see that for the nominated movies, the **star meters (black line)** were in the range of 7.1 to 7.3 in the early 40s; however, after that it became constant between 7.4 and 7.5. Meanwhile, when we look at the non-nominated titles, the star meters were mostly below 7 and it seems to be going down from late 90s onwards. This is because there were more low ratings in these years (note: ratings lower than 2 are filtered out as they have few cast and perhaps low exposure)



▼ 🎭 Genres

- **Nomination**: from all the records, of those that are nominated, Drama, Comedy and Adventure had the most entries but they do not have the highest rating; meanwhile genres like Mystery, Thriller or Animation, Film-Noir or Sci-Fi had very few being nominated but their ratings are very high. This indicates that for a genres that are a little out of norm (i.e. eliciting extreme sensation/ imagination) will have to be extremely well-done in order to be nominated. (Note: only the first genre recorded for each title is considered as it is the main categorisation)

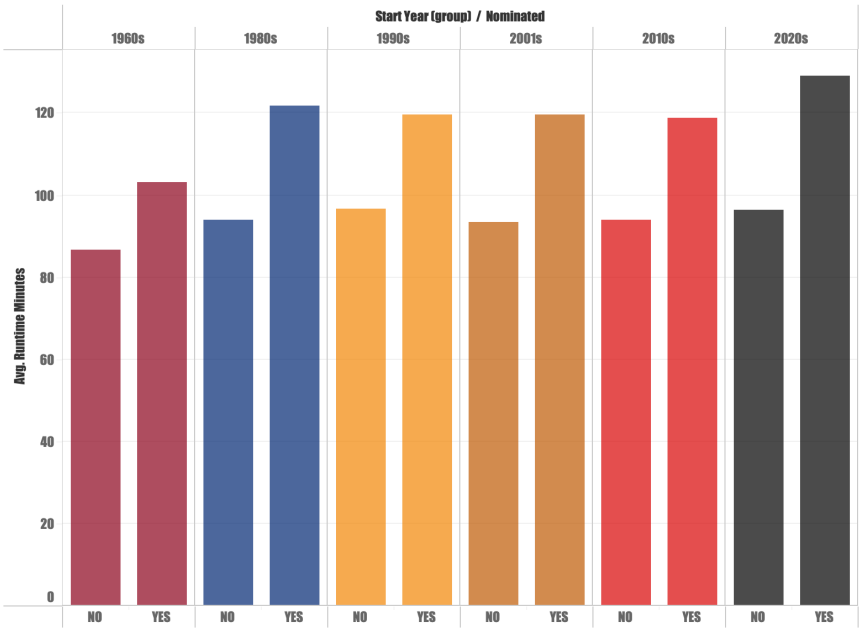


- Here is how the genres change over time - essentially Drama, Comedy and Adventure appear very frequently
- **Note:** 1960s includes the years before, trimmed to reduce complexity due to the sheer size of number of titles
- **Link**: https://youtu.be/f_voqR22Usg

https://youtu.be/f_voqR22Usg

▼ 🕰️ Running Length

- In terms of running time, between the nominated & non-nominated movies, it is very clear that the nominated movies are on the average 25 mins longer (average running time = 118 minutes). This could show that with 25 minutes extra, the stories could be told in more details or the plots can be more comprehensive, and hence, creating a longer lasting impact on the audience.

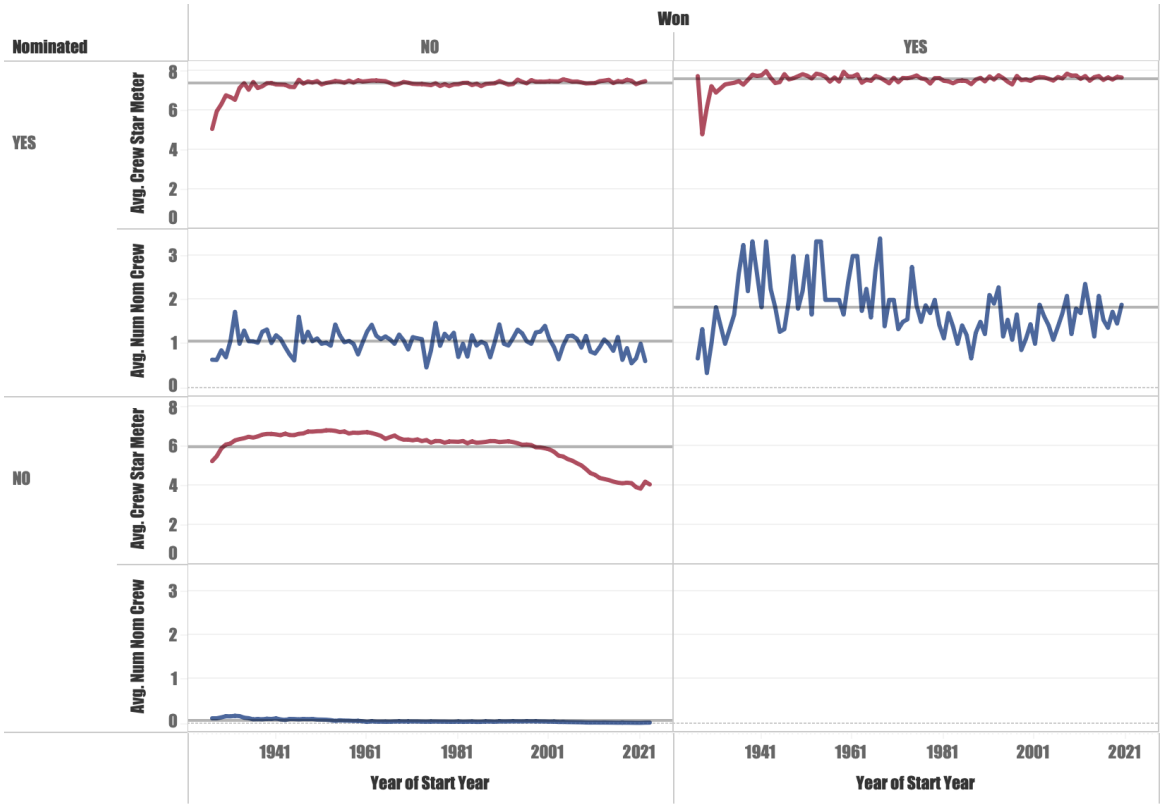


▼ 🏅 Winning

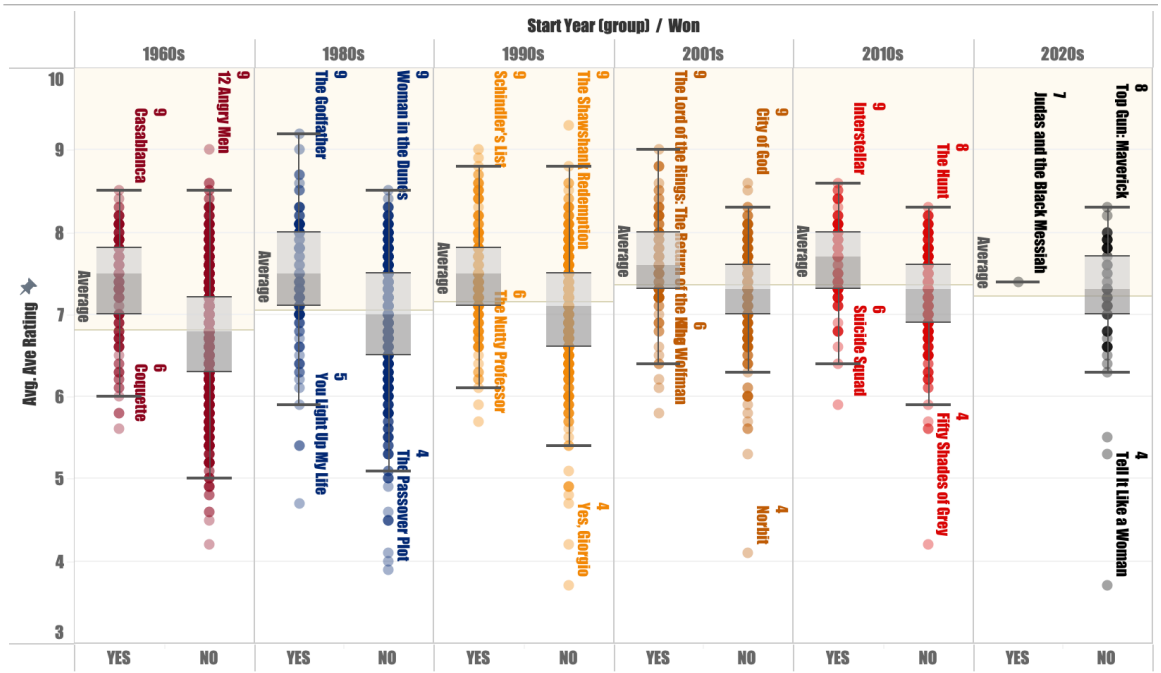
- Given the number of titles produced each year, the winning chance is slim, however we see certain genres having better chance, and they are also the top 3 genres with the most titles produced. Drama has the highest **winning chance at 0.05%,** Comedy at **0.03%**, Biography and Adventure at **0.02%.**

	Won / Nominated		
	NO		YES
Genres1	YES	NO	YES
Drama	0,20%	24,99%	0,05%
Comedy	0,14%	17,15%	0,03%
Biography	0,05%	3,25%	0,02%
Adventure	0,06%	3,02%	0,02%
Action	0,07%	7,21%	0,01%
Crime	0,04%	3,96%	0,01%
Horror	0,00%	2,75%	0,00%
Film-Noir	0,00%	0,01%	0,00%
Documentary	0,00%	21,66%	0,00%
Music	0,00%	0,87%	0,00%
Fantasy	0,00%	0,50%	0,00%
Animation	0,01%	0,89%	0,00%
Adult		1,25%	
Family	0,00%	0,86%	
Game-Show		0,00%	
History	0,00%	0,26%	
Musical	0,01%	0,48%	
Mystery	0,00%	0,59%	
News		0,01%	
None		5,79%	
Reality-TV		0,06%	
Romance	0,00%	0,93%	
Sci-Fi	0,00%	0,36%	
Short		0,00%	
Sport		0,23%	
Talk-Show		0,01%	
Thriller	0,00%	1,36%	
War	0,00%	0,14%	
Western	0,00%	0,67%	

- There are up to 10 Best Pictures nominees a year and that means the possibility to win once nominated is 10%. What sets the nominees and winners apart, I wondered? Between being nominated and winning, there is small difference in terms of crew star meter, showing that for a nominated film, the cast's track records are solid. When it came to number of nominated crew, **we see a difference in number of crew that are previously nominated, with the winning titles having almost double number of crew**, and in the past before 1960, triple but likely due to the number of movies produced being much smaller than our present days.



- Looking deeper into each decades of the nominated & winning titles, we see that the average ratings are increasing (except for the 2020s where we still have a few more years to the end of the decade, so the numbers are only based on existing movies of 2021 and 2022). This does signal that with more titles produced, more technology entering our life, to win an Oscars, one must go beyond expectation to win the public's favour

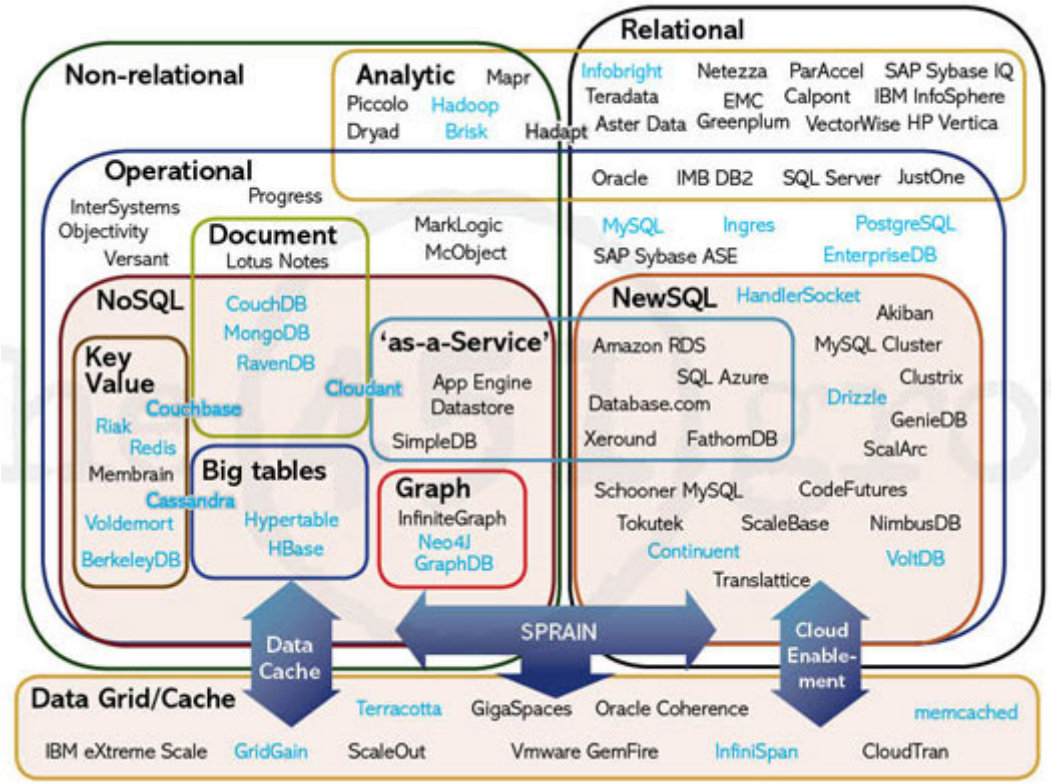


Database Type

A good database is crucial to any data process, so data can be stored, retrieved, connected to serve different purposes and end customers, whether internal or external

▼ 🔣 Types of Databases

There are many types of databases but they are mainly divided into relational database and non-relational database, within which we have subsets of tools and for different usage:



▼ 🏢 Database comparison

SQL and NoSQL are essentially different in the following aspects:

SQL		NoSQL
Relational	**Model**	Non-relational
Structured Tables	**Data**	Semi-structured
Strict Schema	**Flexibility**	Dynamic Schema
ACID compliance (Atomicity, Consistency, Isolation, Durability)	**Transactions**	Mostly BASE (Basically Available, Soft-state, Eventually consistent)

SQL		NoSQL
Strong	**Consistency**	Eventual to Strong
Prioritises consistencies	**Availability**	Basic Availability
Vertically scaled	**Scale**	Horizontal by data partitioning

▼ 🤔 Choice of Database

As I am working on tables, with related keys, and vertically expanded, I opted for **SQL** database for data modelling

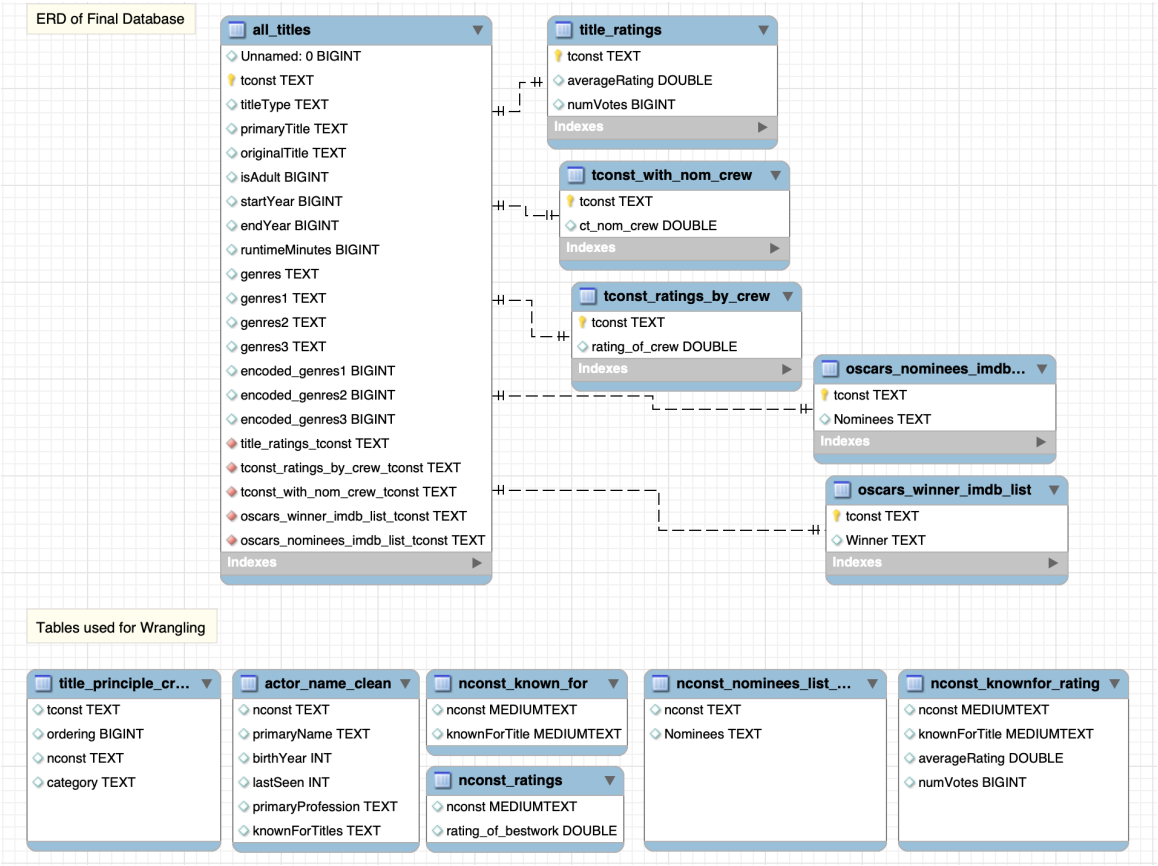
Entities Relationship Diagram

▼ 👣 Entities Relationship Diagram

This is the final relationship diagram when all information are put into place for analysis.

- all_titles: cleaned table `title_basics` , with genres split and encoded, from imdb
- title_ratings: cleaned table `title_rating` , from imdb
- tconst_with_nom_crew: SQL derived table from table `title_prin` and nominated list (`nom_nconst`) scraped from imbd
- tconst_ratings_by_crew: SQL derived table from table `title_prin` , `title_rating`
- oscars_nominees_imdb_list: Scraped from imdb into table/ dataframe
- oscars_winner_imdb_list: Scraped from imdb into table/ dataframe

In order to have the final databased, I also used another 5 tables as intermediaries listed at the bottom of the image.



Queries & Final Database

❓Queries

- Since the original table listed the titles each individual is known for in a string, I split & put them into row in order to calculate the average score for each individual based on the ratings of the titles.

```
-- 1. CREATE A TABLE THAT MAPS EACH INDIVIDUAL WITH THE MOVIES THEY ARE KNOWN FOR:
CREATE TABLE nconst_known_for
AS (
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 1), ', ', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 2), ',', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 3), ',', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 4), ',', -1) knownForTitle FROM actor_name_clean
);
```

nconst	knownForTitle
nm0000002	tt0037382
nm0000003	tt0054452
nm0000004	tt0078723

- The next step is to look up the ratings for each `knownForTitle` above to calculate the star meter of each individual/ nconst

```
-- 2. CREATE A TABLE THAT MAPS THE MOVIES AND THE CORRESPONDING RATINGS:
CREATE TABLE nconst_knownfor_rating
AS
(SELECT nconst, knownForTitle, averageRating, numVotes
FROM nconst_known_for nkf
LEFT JOIN title_ratings tr
ON nkf.knownForTitle = tr.tconst);
```

```
-- 3. AVERAGE RATING FOR THE MOVIES A PERSON HAS DONE, WEIGHTED WITH NUMBER OF VOTE COUNTS
CREATE TABLE nconst_ratings
AS
  (SELECT
     nconst,
     (sum(averageRating * numVotes)/ sum(numVotes)) as nconst_rating
  FROM(
     SELECT *, ROW_NUMBER() OVER (PARTITION BY nconst ORDER BY averageRating DESC) rn
     FROM nconst_knownfor_rating) x
  GROUP BY nconst);
-- nconst_rating = average rating of the top 4 titles they are known for, weighted by number of votes
```

```
ALTER TABLE nconst_ratings RENAME COLUMN nconst_rating TO rating_of_bestwork;
```

nconst	rating_of_bestwork
nm0000002	7.537273771697488
nm0000003	7.255458823529411
nm0000004	7.585644647710464

- With the rating of each individual, now I can compute the score of title based on the crew's historical work

```
-- 4. SCORE OF EACH TITLE BASED ON THE RATING OF THE MAIN CREW (NCONST_RATING)
CREATE TABLE tconst_ratings_by_crew
AS (
SELECT tpc.tconst, avg(nconst_rating)
FROM title_principle_crew tpc
LEFT JOIN nconst_ratings nr
ON tpc.nconst = nr.nconst
GROUP BY tconst);
```

```
ALTER TABLE tconst_ratings_by_crew RENAME COLUMN `avg(nconst_rating)` TO rating_of_crew;
```

tconst	rating_of_crew
tt0017534	7.443602872263254
tt0017637	7.861560383386581
tt0017668	7.391269005445723

- As we need the number of nominated crew for each title, we need to join tables too

```
-- 6. CREATE TABLE THAT COUNTS THE CREW WHO ARE NOMINATED IN A TITLE
-- (i.e. IF A TITLE HAS A STAR-STUDDED CAST)
CREATE TABLE tconst_with_nom_crew
AS
(SELECT x.tconst, sum(x.nom_crew) as ct_nom_crew  -- count of nominated crew in a title
FROM
   (SELECT tconst, nl.Nominees as nom_crew
   FROM  title_principle_crew tpc
   LEFT JOIN nconst_nominees_list_446 nl
   ON tpc.nconst = nl.nconst) x
GROUP BY x.tconst);

UPDATE tconst_with_nom_crew
SET ct_nom_crew = 0
WHERE ct_nom_crew IS NULL;
SELECT * FROM tconst_with_nom_crew;
```

tconst	ct_nom_crew
tt0003854	0
tt0011715	0
tt0011801	0
tt0013274	0
tt0014985	0
tt0015152	0
tt0015414	0
tt0015724	0
tt0015737	0
tt0016029	1

▼ 🖐 Final Database

- With all the tables ready, I combined the tables to create the final dataset

```
-- 7. CREATE FINAL DATABASE:
CREATE TABLE imdb_megadata_2
AS
  (SELECT  a.*,
           tr.averageRating as ave_rating,
           tr.numVotes as num_votes,
           cc.ct_nom_crew as count_nom_crew,
           tc.rating_of_crew as crew_star_meter,
           ond.Nominees as nominated,
           ow.Winner as won
   FROM all_titles_2 a
   LEFT JOIN title_ratings tr ON a.tconst = tr.tconst
   LEFT JOIN tconst_with_nom_crew cc ON a.tconst = cc.tconst
   LEFT JOIN tconst_ratings_by_crew tc ON a.tconst = tc.tconst
   LEFT JOIN oscars_winner_imdb_list ow ON a.tconst = ow.tconst
   LEFT JOIN oscars_nominees_imdb_list ond ON a.tconst = ond.tconst);

   SELECT * FROM imdb_megadata_2;
   SELECT * FROM all_titles;
```

tconst	...	primaryTitle	originalTitle	...	startYear	endYear	runtime...	genres	genres1	genres2	genres3	e...	en...	en...	ave...	num_...	co...	crew_star...	nomi...	won
tt0021079	m	Little Caesar	Little Caesar	0	1931	2024	79	Actio...	Action	Crime	Drama	0	6	8	7.2	13686	1	7.562134...	1	NULL
tt0018937	m	A Girl in Every Port	A Girl in Every Port	0	1928	2024	78	Actio...	Action	Adventure	Comedy	0	2	5	6.6	760	1	6.865098...	NULL	NULL
tt0021867	m	Finn and Hattie	Finn and Hattie	0	1931	2024	78	Com...	Comedy	None	None	5	20	20	6.2	66	2	6.928074...	NULL	NULL
tt0023088	m	The Kid from Spain	The Kid from Spain	0	1932	2024	96	Com...	Comedy	Musical	Romance	5	17	22	6.5	570	1	7.323457...	NULL	NULL
tt0019752	m	The Careless Age	The Careless Age	0	1929	2024	65	Drama	Drama	None	None	8	20	20	5.7	50	0	7.126265...	NULL	NULL
tt0022386	m	The Sin of Madelon Cl...	The Sin of Madelon Cl...	0	1931	2024	75	Crim...	Crime	Drama	Music	6	8	16	6.6	979	1	6.826906...	1	1

Conclusion

In conclusion, we see that factors such as average ratings, number of nominated crew in a title, reputation of the crew, run time and genres make a huge difference to be nominated for an Oscars, and to towards winning one, a title would fare better if it has higher number of crew who had been nominated.

To win an Oscars, one should minimally/ optimally aim for:

- **Average rating: 7.5**, by the public
- Number of **nominated crew or cast: 2**
- **Star meter: 7.6**, i.e. the crew have consistently work on titles with rating of 7.5

- **Run time: 123 minutes** so there is enough time to explore the topic and tell the story
- **Genres: drama, comedy, biography or adventure**. If the genres belongs to the likes of sci-fi or mystery, it would need to have a strong storyline or cast to drive

Limitations & Further Improvements

▼ 🧗 Enhance we must

While we have drawn several conclusions, I believe we could have more insights with the following features:

- Budget - to see the investment poured into a movie to ensure good visual effects, thorough casting, advertisement, etc
- Box office number - to see the financial response & interest from the public
- Number of nominations per individuals - to support the reputation of a cast/ crew more accurately
- Full nominated/ winning list - to account for all the credits of all crew, including visual effects, costume, or screenplay department

- Credibility from other awards, for example BAFTA awards and perhaps Razzie awards too

▼ 📄 Sources

- https://www.imdb.com/interfaces/
- https://www.imdb.com/list/ls055265443/
- https://www.imdb.com/list/ls055903720/
- https://www.imdb.com/list/ls096107034/
- https://www.imdb.com/list/ls026557238/

▼ 💃 Fun Facts

- The colour scheme of the categorical charts are inspired by Everything Everywhere All At Once poster (`tconst` = “tt6710474”) who just won the Oscars 2023 on 12th March, and its statistics are below:
  - ✔️ Average rating: 8
  - ✔️ Runtime: 139 minutes
  - ✔️ Genres: Action, Adventure, Comedy, Fantasy, Sci-fi
  - ✔️ Star meter: 7.79
  - ❌ Number of nominated crew: 0