Database Project: Oscars & Movies Dataset

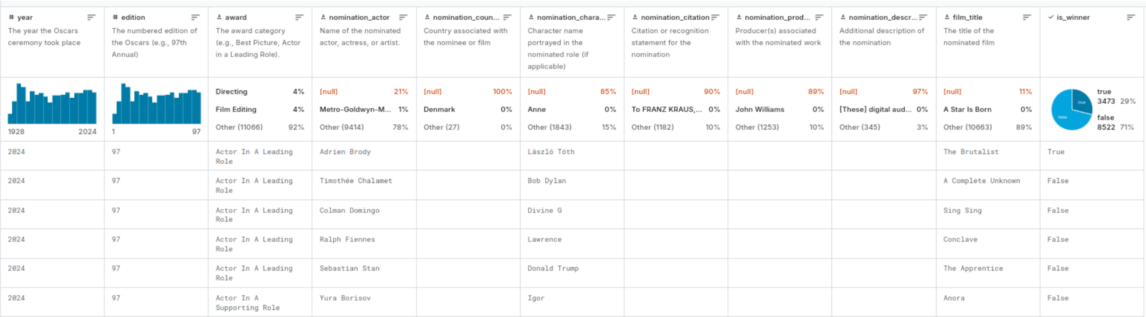
Danila Pechenev & Gwenn Garrigues

Context and Objectives

- The **goal** of the project is to build a fully structured relational database from real-world movie industry datasets.
- Initial dataset: **Oscars Dataset** (1928 - 2024, 12k entries).
- Extended with: **Movies Dataset** (1M entries).
- Tasks:
  - Modeling the data: CDM (ER) and LDM.
  - Preprocessing and cleaning the datasets.
  - Creating the tables and inserting data into them.
  - Creating meaningful queries.

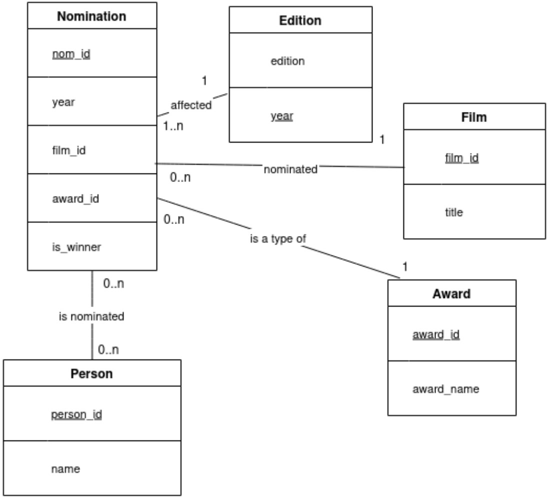
Why the Oscars Dataset?

- Rich dataset: awards, films, actors, winners, character names.
- Entities can be easily linkable with other related datasets.
- Raw, messy real-world data that needs to be preprocessed first.

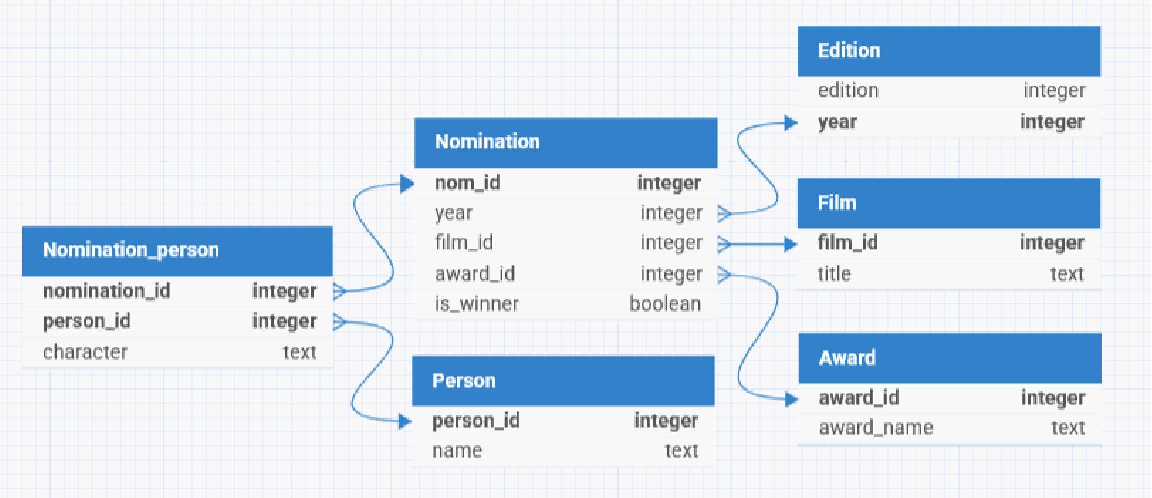


# year	# edition	award	nomination_actor	nomination_coun...	nomination_chara...	nomination_citation	nomination_prod...	nomination_descr...	film_title	✓ is_winner
The year the Oscars ceremony took place	The numbered edition of the Oscars (e.g., 97th Annual)	The award category (e.g., Best Picture, Actor in a Leading Role).	Name of the nominated actor, actress, or artist.	Country associated with the nominee or film	Character name portrayed in the nominated role (if applicable)	Citation or recognition statement for the nomination	Producer(s) associated with the nominated work	Additional description of the nomination	The title of the nominated film	
1928 2024	1 97	Directing 4% Film Editing 4% Other (11066) 92%	[null] 21% Metro-Goldwyn-M... 1% Other (9414) 78%	[null] 100% Denmark 0% Other (27) 0%	[null] 85% Anne 0% Other (1843) 15%	[null] 90% To FRANZ KRAUS,... 0% Other (1182) 10%	[null] 89% John Williams 0% Other (1253) 10%	[null] 97% [These] digital aud... 0% Other (345) 3%	[null] 11% A Star Is Born 0% Other (10663) 89%	true 3473 29% false 8522 71%
2024	97	Actor In A Leading Role	Adrien Brody		László Tóth				The Brutalist	True
2024	97	Actor In A Leading Role	Timothée Chalamet		Bob Dylan				A Complete Unknown	False
2024	97	Actor In A Leading Role	Colman Domingo		Divine G				Sing Sing	False
2024	97	Actor In A Leading Role	Ralph Fiennes		Lawrence				Conclave	False
2024	97	Actor In A Leading Role	Sebastian Stan		Donald Trump				The Apprentice	False
2024	97	Actor In A Supporting Role	Yura Borisov		Igor				Anora	False

CDM — Oscars Dataset



LDM — Oscars Dataset



Nomination_person	
nomination_id	integer
person_id	integer
character	text

Nomination	
nom_id	integer
year	integer
film_id	integer
award_id	integer
is_winner	boolean

Person	
person_id	integer
name	text

Edition	
edition	integer
year	integer

Film	
film_id	integer
title	text

Award	
award_id	integer
award_name	text

Dataset Preprocessing Overview

- Loaded the raw `oscars.csv` dataset from Kaggle and inspected dimensions and missing values.
- Removed nominations without an associated film ($\approx$ 11% with `film_title = NaN`).
- Dropped irrelevant or unusable text columns: `nomination_citation`, `nomination_description`, `acceptance_speech_text`, `nomination_country`, `acceptance_speech_url`.
- Unified nominee information by merging `nomination_actor` and `nomination_producers` into `nomination_people`.
- Parsed `nomination_people` to extract clean individual names: normalization, splitting, filtering groups/organizations/countries, and reconstructing incomplete patterns.
- Created structured list-valued column `people_list` and removed intermediate text fields.

Database Creation

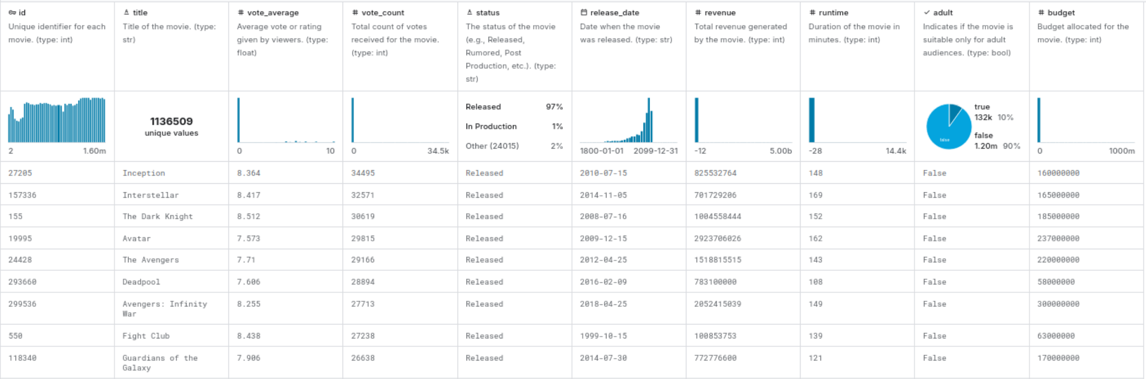
- **PostgreSQL DBMS.** After preprocessing, the cleaned data is imported into a PostgreSQL database.
- **Schema creation.** A dedicated schema `films` is created to organize all tables.
- **Table creation.** Tables are created according to the logical data model (LDM):
  - primary keys defined with `INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY`
  - foreign keys, unique constraints, and `NOT NULL` constraints
  - all attributes except `character` are `NOT NULL`
  - foreign keys include `ON DELETE CASCADE` to maintain consistency

Table Population

Improved approach: **bulk insert**

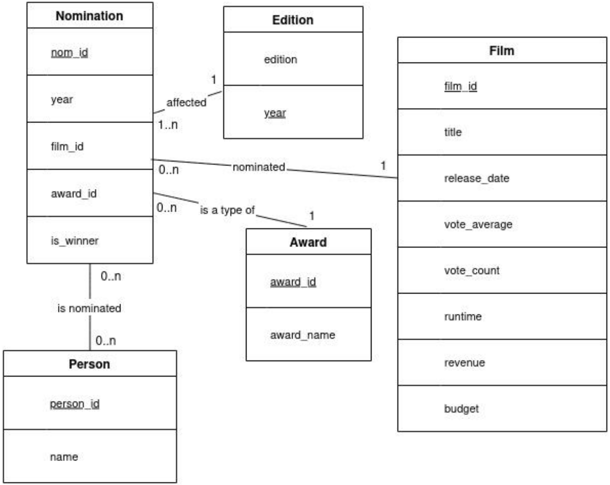
- Group thousands of inserts into one efficient operation.
- Create a temporary table `tmp_oscars` that mirrors the original structure.
- Bulk insert: perform a single COPY FROM STDIN using `copy_expert` (psycopg2).
- Populate the actual tables directly inside PostgreSQL:
  - Insert unique (edition, year) into `Edition`.
  - Insert unique awards into `Award`.
  - Insert unique film titles into `Film`.
  - `Nomination`: link film, award, year, winner flag.
  - `Person`: extract distinct names.
  - `Nomination_person`: connect nominations and people (with character name).

Adding new dataset - Full TMDB Movies Dataset

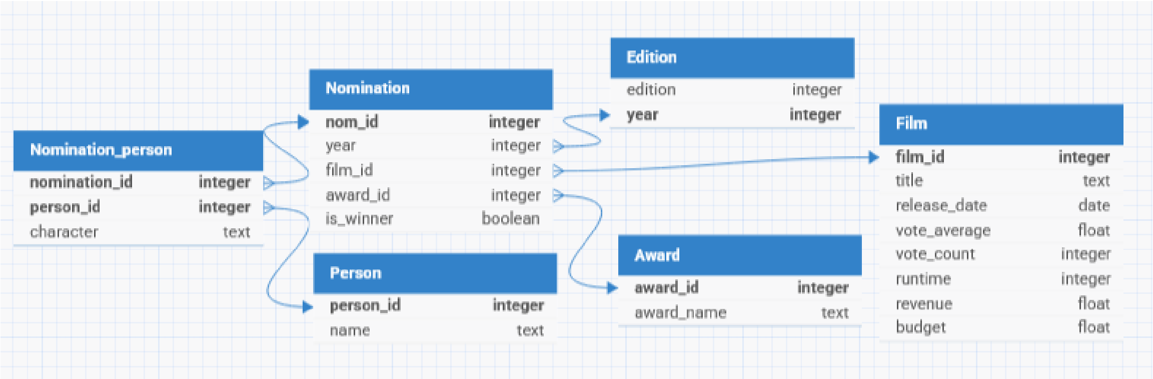


id	title	vote_average	vote_count	status	release_date	revenue	runtime	adult	budget
Unique identifier for each movie. (type: int)	Title of the movie. (type: str)	Average vote or rating given by viewers. (type: float)	Total count of votes received for the movie. (type: int)	The status of the movie (e.g., Released, Rumored, Post Production, etc.). (type: str)	Date when the movie was released. (type: str)	Total revenue generated by the movie. (type: int)	Duration of the movie in minutes. (type: int)	Indicates if the movie is suitable only for adult audiences. (type: bool)	Budget allocated for the movie. (type: int)
2 1.60m	1136509 unique values	0 10	0 34.5k	Released 97% In Production 1% Other (24015) 2%	1800-01-01 2099-12-31	-12 5.00b	-28 14.4k	true 132k 10% false 1.20m 90%	0 1000m
27205	Inception	8.364	34495	Released	2010-07-15	825532764	148	False	160000000
157336	Interstellar	8.417	32571	Released	2014-11-05	701729206	169	False	165000000
155	The Dark Knight	8.512	30619	Released	2008-07-16	1004558444	152	False	185000000
19995	Avatar	7.573	29815	Released	2009-12-15	2923706026	162	False	237000000
24428	The Avengers	7.71	29166	Released	2012-04-25	1518815515	143	False	220000000
293660	Deadpool	7.606	28894	Released	2016-02-09	783100000	108	False	58000000
299536	Avengers: Infinity War	8.255	27713	Released	2018-04-25	2052415039	149	False	300000000
550	Fight Club	8.438	27238	Released	1999-10-15	100853753	139	False	63000000
118340	Guardians of the Galaxy	7.906	26638	Released	2014-07-30	772776600	121	False	170000000

CDM — Movies + Oscars



LDM — Movies + Oscars



Nomination_person	
nomination_id	**integer**
person_id	**integer**
character	text

Nomination	
nom_id	**integer**
year	integer
film_id	integer
award_id	integer
is_winner	boolean

Person	
person_id	**integer**
name	text

Edition	
edition	integer
year	**integer**

Award	
award_id	**integer**
award_name	text

Film	
film_id	**integer**
title	text
release_date	date
vote_average	float
vote_count	integer
runtime	integer
revenue	float
budget	float

Movies Dataset — Preprocessing

- Removed irrelevant rows:
  - adult films, missing titles, unreleased films.
- Removed duplicates using:

$$(\texttt{title}, \texttt{release\_year})$$

- Cleaned numerical values:
  - $\texttt{budget}, \texttt{revenue}, \texttt{runtime} = 0 \rightarrow \text{NaN}$
  - $\texttt{vote\_count} = 0 \rightarrow \texttt{vote\_average} = \text{NaN}$
- Converted `release_date` to datetime.

Matching the Movies and Oscars Datasets

- Matching film of both dataset:
  - using identifiable pair (`title`, `release_year`)
  - exact lowercase title match,
  - release date closest to first Oscar nomination.
- Result:

4733 matches out of 5090 Oscar films

Film Table Modification

- Added new attributes to `film`:

  `release_date`, `vote_average`, `vote_count`, `runtime`, `revenue`, `budget`

- Updated constraint:
  - Many movie with the same title ? Remove the unique constraint `title`
  - New unique key:

    (`title`, `release_date`)

- New attributes may be `NULL` when no match exists.

Table Population

- Matching all existing Oscar films with Movies dataset information.
- Added a large sample from Movies dataset:

  10 000 additional films

- For consistency :
  - uniqueness on (`title`, `release_date`)
  - generating unique identifier for each film
  - no duplicated films inserted.

Query 1 — Most Nominated Films

```
SELECT film.title, COUNT(Nomination.nom_id) AS nom_total
FROM film
JOIN nomination ON film.film_id = nomination.film_id
GROUP BY film.title
ORDER BY nom_total desc;
```

Film	**Nominations**
A Star Is Born	25
West Side Story	18
Titanic	16
Moulin Rouge	15

Query 2 — Actors Who Played “Joker”

```
SELECT person.name
FROM person
JOIN nomination_person ON person.person_id = nomination_person.person_id
WHERE nomination_person.character = 'Joker'
```

Actor
Heath Ledger

Query 3 — Highest Win Ratio

```
WITH film_stats AS (
SELECT film.title, n.year, COUNT(n.nom_id) AS total_noms,
SUM(n.is_winner::INT) AS total_wins
FROM film
JOIN nomination AS n ON film.film_id = n.film_id
GROUP BY film.title, n.year
HAVING COUNT(n.nom_id) >= 5
)
SELECT title, year, total_wins, total_noms,
total_wins / total_noms AS conversion_rate
FROM film_stats
ORDER BY conversion_rate DESC, total_noms DESC LIMIT 1
```

Film	**Year**	**Wins**	**Noms**	**Ratio**
Return of the King	2003	11	11	1.00

Query 4 — Top Rated Films (Votes over 10k)

```
WITH film_win AS ( select film_id, count(*) AS nb_win
FROM nomination WHERE is_winner=true GROUP BY film_id)
SELECT title, vote_average, nb_win
FROM film F JOIN film_win fw USING (film_id)
WHERE vote_average IS NOT NULL AND vote_count > 10000
ORDER BY vote_average DESC
LIMIT 10;
```

Film	**Rating**	**Wins**
The Godfather	8.707	3
The Godfather II	8.591	6
Schindler's List	8.573	7
Spirited Away	8.539	1

Query 5 — Lowest-Budget Oscar Winner

```
SELECT f.title, f.budget
FROM film f
JOIN nomination n ON f.film_id = n.film_id
JOIN award a ON n.award_id = a.award_id
WHERE n.is_winner AND f.budget != ’NaN’
ORDER BY  f.budget ASC
LIMIT 1;
```

Film	**Budget**
Kiss of the Spider Woman	11

Conclusion

- Final product: easily extendable, normalized and multi-source database.
- Movies dataset integration enables meaningful queries:
  - film rating vs. award performance,
  - financial analysis of Oscar winners.
- Future extension:
  - Detailed People dataset (date of birth, country, ...).