Movie Database

DJ Bonifacic

Ian Pearlstein

Diego Robayo

Executive Summary

The end of the year movie season is upon us, so we decided to create a database involving different stats for films. Our first step for this ETL process was to find our data. Using Kaggle, we were able to find csv datasets related to movie sales, ratings and genre.

Next, we needed to transform our data, so we could eventually merge it together. We cleaned the three files using pandas by dropping columns, removing NaNs, setting movie titles to lowercase and splitting data into two columns using delimiters. After that, we merged our datasets based on the title of each movie. We also created a unique ID as well. Once the data was ready, we split our merged data back into movie_genre, movie_id and movie_rating tables. At this point, each table had the unique title ID that would help merge them back together when querying.

Lastly, we needed to load our data. We created a relational database in pgAdmin with the three different data tables listed above and loaded each one. Once the data was ready, we were able to create different queries based off the year, title, gross sales, etc. The reason we created three tables in pgAdmin with a title ID link is to segment the data into a cleaner format. The three tables can be easily merged with the ID shared between them.

Data Sources

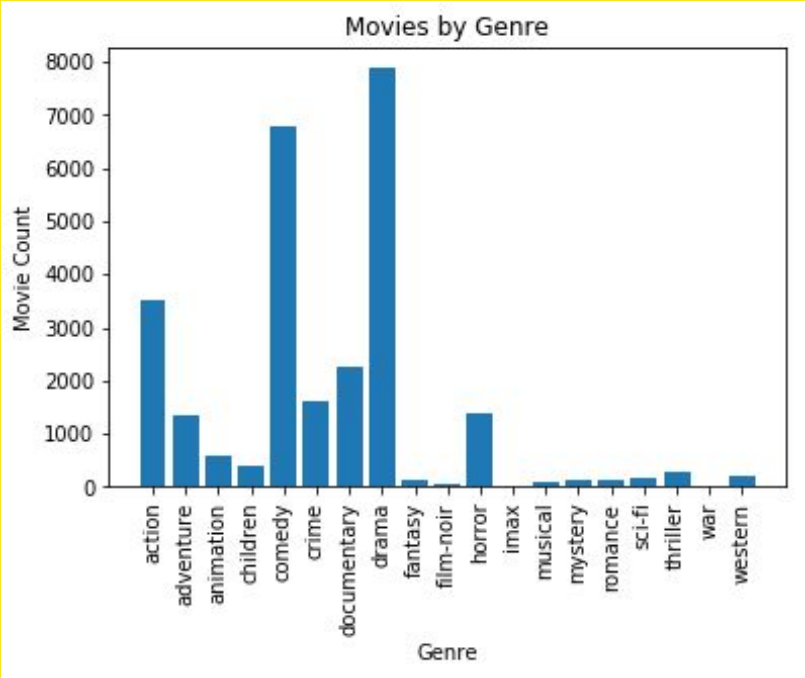
kaggle

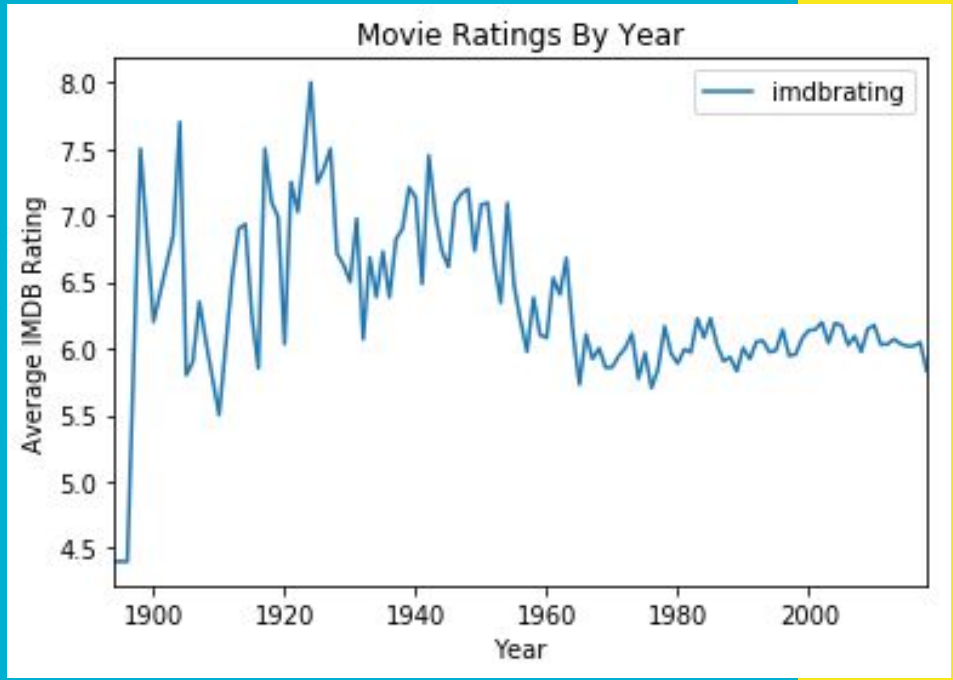
Movie Sales

Movie Genres

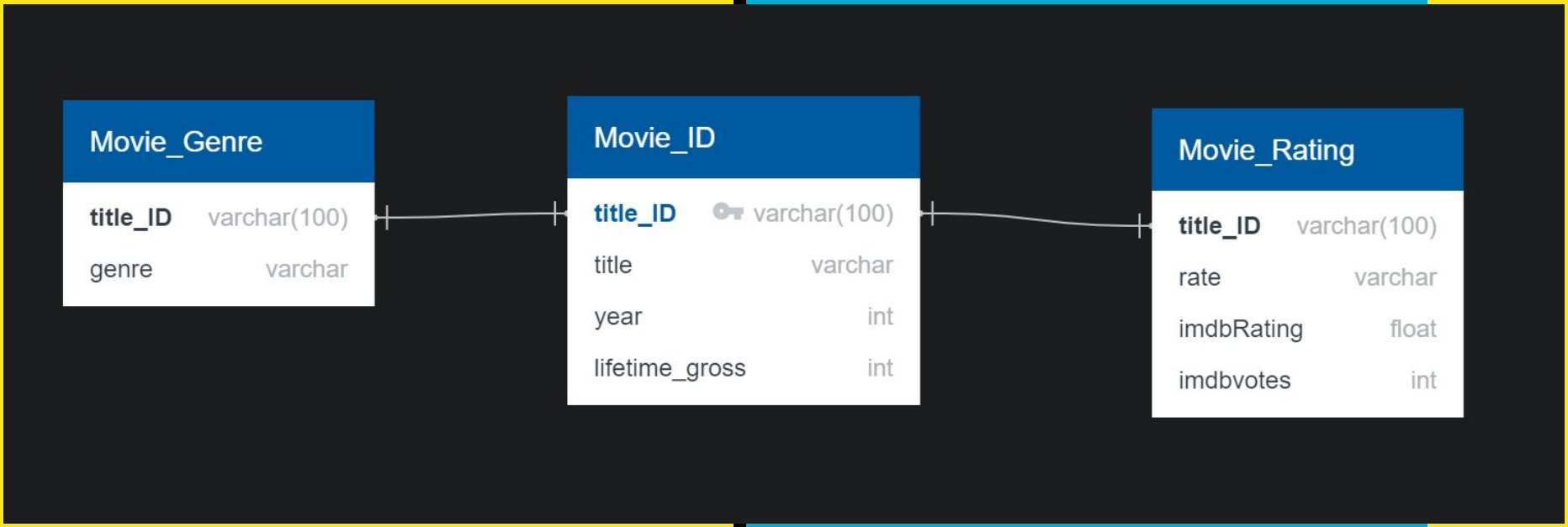
Movie Ratings

Graphs


Movies by Genre
Movie Count
8000
7000
6000
5000
4000
3000
2000
1000
0
action
adventure
animation
children
comedy
crime
documentary
drama
fantasy
film-noir
horror
imax
musical
mystery
romance
sci-fi
thriller
war
western
Genre


Movie Ratings By Year
imdbrating
Average IMDB Rating
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
1900
1920
1940
1960
1980
2000
Year

Entity Relationship Diagram



Data Dictionary

Table	Field Name	Data Type	Description	Example
movie_genre	title_id	varchar	Primary key of the table	34863
movie_genre	genre	varchar	Movie genre	Drama
movie_id	title_id	varchar	Primary key of the table	tt3842023
movie_id	title	varchar	Movie title	Wild Tales
movie_id	year	INT	Year when the movie was released	2013
movie_id	lifetime_gross	varchar	Lifetime box office money	659363944
movie_rating	title_id	varchar	Primary key of the table	tt3842023
movie_rating	rated	varchar	Rate of the film's suitability for certain audiences based on its content	R
movie_rating	imbd rating	float	Movie rating on imdb	8.2
movie_rating	imdb votes	INT	Amount of votes on imdb	565

Schema

```
--CREATE SCHEMA

CREATE TABLE movie_genre (
        title_id varchar(100),
        genre varchar(100)
);


CREATE Table movie_rating (
        title_id varchar(100),
        rated varchar(100),
        imdbrating float,
        imdbvotes INT
);

CREATE Table movie_id (
        title_id varchar(100),
        title varchar(100),
        year INT,
        lifetime_gross INT
);
```

5 Queries

--Query the data to return all the rows from 2015:

```
SELECT title, title_id, year
        FROM movie_id
        WHERE year = 2015;
```

-- Query the data to sort by highest grossing film in 2014:

```
SELECT title, title_id, lifetime_gross
        FROM movie_id
        ORDER BY lifetime_gross DESC;
```

-- Query the data to sort by highest rating in 2014:

```
SELECT movie_rating.title_id, title, imdbrating, imdbvotes
        FROM movie_rating
        inner join movie_id
        ON movie_rating.title_id=movie_id.title_id
        ORDER BY imdbrating DESC;
```

5 Queries

-- Query return highest grossing comedies:

```
SELECT distinct movie_id.title_id, title, lifetime_gross, genre
   FROM movie_id
   LEFT JOIN movie_genre
   ON movie_id.title_id = movie_genre.title_id
   WHERE genre = 'comedy' and year = 2014
   ORDER BY lifetime_gross desc;
```

-- Query returns count of movies in each genre:

```
 SELECT genre
        count(1) as movies
        FROM movie_genre
        GROUP BY genre;
```