# Movie Recommendation app

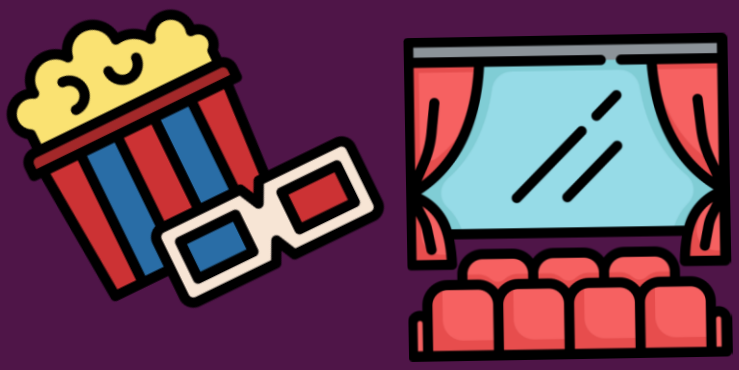
Final project in Jungle's Data Science academy

Shkumbim Mazrekaj

# Table of contents

# Key objectives

1. Find the data of movies
2. Make the necessary data cleaning
3. Load the data into a SQL database
4. Do the query that will be used
5. Integrate the SQL database in python and write the query
6. Make the frontend where the tables will be displayed
7. Designing the final product



# The Dataset

I got the dataset from https://grouplens.org/datasets/movielens/latest/ it has two files one with 100K rows at the ratings table and 9K movies and the other with 33 million rows on the ratings table and 86K movies.

For the risk of having insufficient computing power i used the smaller dataset.

It has 4 tables:

1. **Movies-** where it shows the title and genres of the said movie.
2. **Ratings-** it has the movie_id, user_id of the user who did the rating, rating of said movie and timestamp it's the seconds counted from the specific date: 1/1/1970 of when the rating was submitted.
3. **Tags-** it has the tags on a specific movie_id and timestamp of when the tag was put.
4. **Links**- it has the imdbl_id and tmdbl_id of a specific movie_id

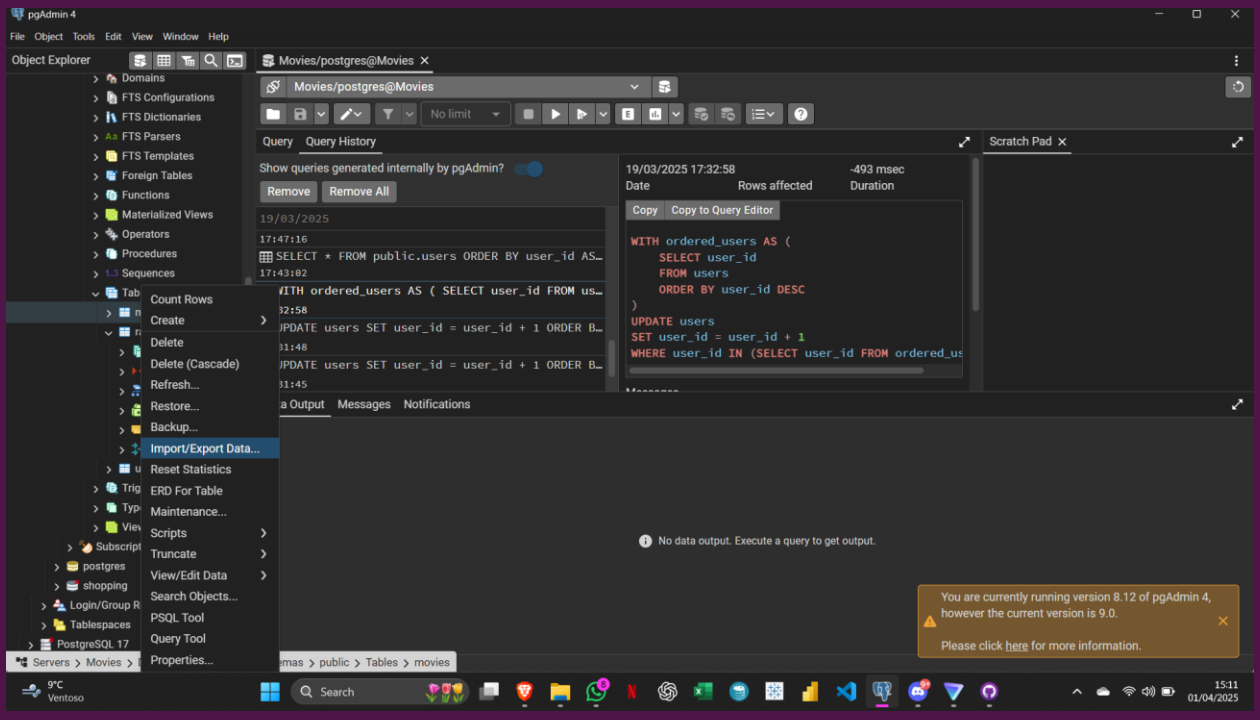
For my purpose i will only use the first two tables where i will use the average rating to rank up the movies from best to worst.

The data is licensed for personal use and fulfills GDPR rules.

# Data Preparation

I first used Python to load the data into a DataFrame, where I removed missing values from the ratings table. Then, I applied algorithms to extract the movie release year from the titles, as the year would contribute to the recommendation search.

Next, I prepared the environment to transfer the cleaned data to SQL. I created the movies and ratings tables, set the necessary variables, and used the export data option to load the datasets.




After that, I used SQL for the remaining cleaning. I converted the timestamp column into an actual date with second-level precision and created the `num_rates` column by grouping the number of ratings per movie.

With that, the data processing phase was complete.

# Backend file

The libraries needed:

```
from sqlalchemy import create_engine
import pandas as pd
```

Now let's connect python with the SQL database using this code:

```
engine = create_engine("postgresql://postgres:[redacted]@localhost/Movies")
```

With the connection done we are ready to write the query which will be inside the get_recomendation function:

```
def get_recommendations(genres, start, end):
    # If multiple genres are selected, filter using multiple LIKE conditions
    genre_conditions = " AND ".join(["m.genres LIKE %s"] * len(genres))
    query = f"""
    SELECT m.title, m.genres, AVG(r.rating) AS avg_rating
    FROM movies m
    JOIN ratings r ON m.movie_id = r.movie_id
    WHERE ({genre_conditions})
      AND m.num_rates >= 50
      AND m.release_year >= %s
      AND m.release_year <= %s
    GROUP BY m.movie_id
    ORDER BY avg_rating DESC
    LIMIT 10;
    """

    # Convert genres into SQL wildcard format (e.g., "%Action%", "%Comedy%")
    genre_params = tuple(f"%{genre}%" for genre in genres)

    return pd.read_sql(query, engine, params=genre_params + (start, end))
```

The table will be displayed by selecting the desired genres and year range, ordered by average rating, with the condition that a movie must have more than 50 ratings.

# Frontend file

For the frontend i used streamlit an online package designed for interactive and easy to use interface.

The libraries used:

```
import streamlit as st
import pandas as pd
from projekti import get_recommendations  # Import backend function
import base64
```

I used the genres list to put into buttons:

```
genres = [
    'Adventure', 'Children', 'Fantasy', 'Animation', 'Action',
    'Horror', 'Musical', 'Sci-Fi', 'Romance', 'Thriller', 'Documentary', 'Drama',
    'War', 'Comedy', 'Crime', 'IMAX', 'Mystery', 'Western', 'Film-Noir'
]

st.subheader("Select Genres:")
selected_genres = st.multiselect("Genres", genres, default=["Action"])
```

I set Action as the default genre.

Then i created the slider for the year range:

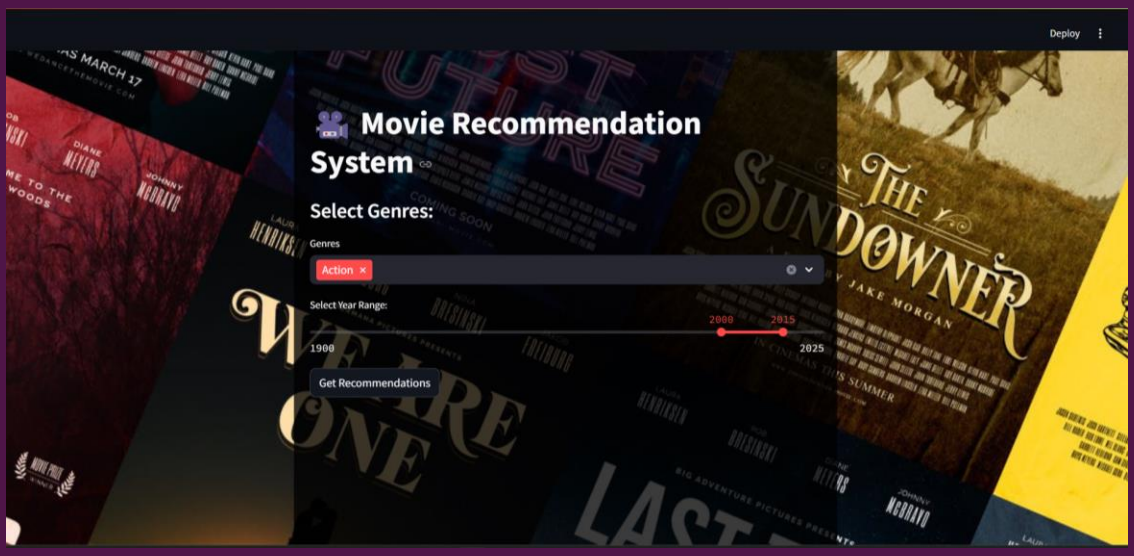
```
start_year, end_year = st.slider("Select Year Range:", 1900, 2025, (2000, 2015))
```

Now the main part:

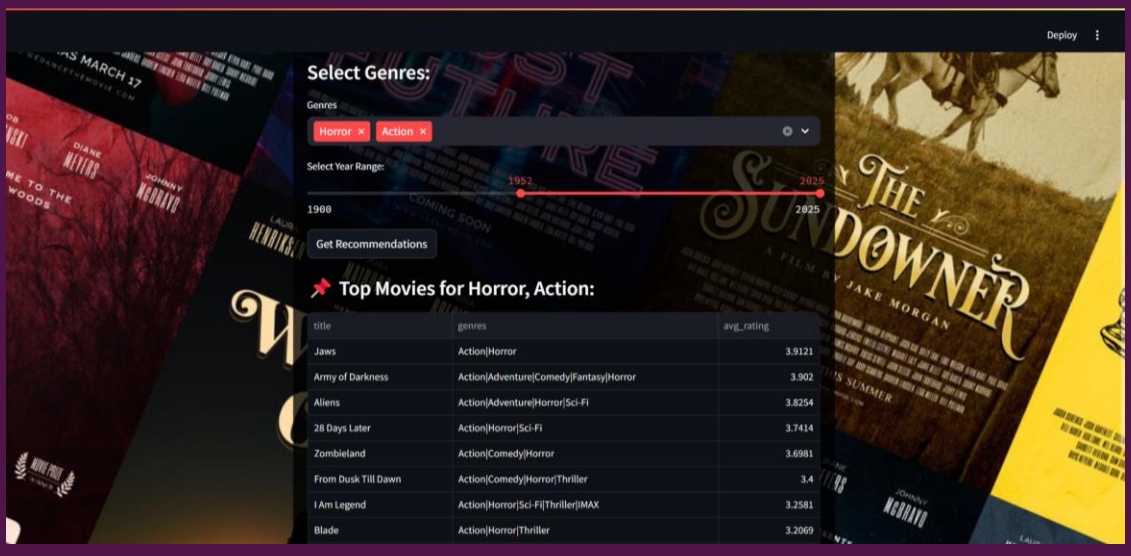
```
if st.button("Get Recommendations"):
    if selected_genres:
        recommendations = get_recommendations(selected_genres, start_year, end_year)
        if recommendations.empty:
            st.warning("No movies found. Try different genres or adjust the year range.")
        else:
            st.write(f"### 📌 Top Movies for {', '.join(selected_genres)}:")
            st.dataframe(recommendations)
    else:
        st.warning("Please select at least one genre.")
```

# Design

I added a background, some colors and some containers. And the final product looks like this:




How the recommendation will look:




The DataFrame displays the title, genres, and average rating in decreasing order.

Movies that don’t match the selected genres or fall outside the specified year range are excluded.

The interface is user-friendly, and the data can be updated if the SQL server is connected to an external source.

Thank you 😊.