A Data Analysis of IMDB and Rotten Tomatoes

Author: Brittney Nitta-Lee

Introduction

This project analyzes data from IMDB and Rotten Tomatoes to explore which genres and movie studios are the most successful in the movie industry. Microsoft is looking to create a new movie studio to produce original video content and wants to know what type of films are doing best at the box office. This analysis provides insights for the client to help decide what movies to create.

Questions

This project will address four questions about the movie industry: 1. What genre of movies are the most popular based on user ratings? 2. What are the top grossing movie genres? 3. What movie content rating based on genre is the most popular? 4. What are the top grossing movie studios?

Data

IMDB
The analysis examines a merged form of data from Box Office Mojo by IMDbPro and IMDB. BOM (Box Office Mojo) has data on domestic gross, and IMDB has data on movie genres.

Rotten Tomatoes
Rotten Tomatoes contains data on genres, user ratings and movie content rating that is used to address popular genres based on content rating and popular genres based on user ratings.

IMDB Data

The IMDB dataset has two tables that were used in this data cleaning. Movie Basics and Movie Ratings includes data files that provide movie genres, title, start year and average votes.

In [1]:
```
import sqlite3
import pandas as pd
```

In [2]:
```
conn = sqlite3.connect('im.db')
cur = conn.cursor()
```

Datacleaning

Pandas is used to format the IMDB data into the following dataframes.

In [3]:
```
movie_gross = pd.read_csv ('bom.movie_gross.csv')

movie_gross.head()
```

Out[3]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010

In [4]:
```
movie_basics = pd.read_sql("""
    SELECT *
    FROM movie_basics;
    """, conn)
movie_basics.head()
```

Out[4]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	dor
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	

In [5]:
```
movie_ratings = pd.read_sql("""
    SELECT *
    FROM movie_ratings;
    """, conn)
movie_ratings.head()
```

Out[5]:

	movie_id	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

Movie Basics and Movie Gross both have a column for individual movie titles. I want to see if the two datasets share the same datat under Title and Primary Title columns. I merged the datasets along the title and primary title columns will show movies that share the same title.

In [6]:
```
movie_basics = movie_gross.merge(movie_basics, how='inner', left_on='title', rig
movie_basics.head()
```

Out[6]:

	title_x	studio	domestic_gross_x	foreign_gross	year	movie_id	primary_title	original_titl
0	Toy Story 3	BV	415000000.0	652000000	2010	tt0435761	Toy Story 3	Toy Story
1	Inception	WB	292600000.0	535700000	2010	tt1375666	Inception	Inceptio
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	tt0892791	Shrek Forever After	Shre Forever Afte
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	tt1325004	The Twilight Saga: Eclipse	The Twiligh Saga: Eclips
4	Iron Man 2	Par.	312400000.0	311500000	2010	tt1228705	Iron Man 2	Iron Man

In [10]:
```
imbd_df= pd.read_sql("""

SELECT *
FROM movie_basics
JOIN movie_ratings
    USING(movie_id)
;
""", conn)
imbd_df.head()
```

Out[10]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	dor
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	

In [11]:
```
Same_movie_titles = []
for title in imbd_df['primary_title'].unique():
    if title in movie_gross['title'].unique():
        Same_movie_titles.append(title)
```

In [12]:
```
print(Same_movie_titles[0:5])
len(Same_movie_titles)
```

```
['Wazir', 'On the Road', 'The Secret Life of Walter Mitty', 'A Walk Among the To
mbstones', 'Jurassic World']
```

Out[12]:
```
2598
```

In [ ]:
```
for i in range(0,3370):
```

```
    if movie_basics['year_x'][i] != movie_basics['start_year'][i]:
        movie_basics.drop(i, inplace = True)
```

I created a new dataframe to join Movie Basics and Movie ratings using Movie ID.

Since the genres are separated by commas, I wanted to split the genres up using the explode function.

In [ ]:
```
imbd_df['genres'] = imbd_df['genres'].str.split(',')
imbd_genres_df = imbd_df.explode('genres')
```

Analysis

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
#top grossing studios
sns.set_style('darkgrid')
sns.set_palette('Set2')

sns.barplot(data=movie_basics_df, x="domestic gross", y="studio", ci=None)
sns.set(rc = {'figure.figsize':(20,40)})

plt.title('Top Grossing Studios', fontsize=35, fontname='Arial')
plt.xlabel('Total Gross (Hundred Million USD)', fontsize=20, fontname='Arial')
plt.ylabel('Studio', fontsize=20, fontname='Arial')
plt.xticks(fontsize=15, fontname='Arial')
plt.yticks(fontsize=15, fontname='Arial')


sns.despine()
plt.show()
```

In [ ]:
```
#explode function to separate genres
movie_basics['genres'] = movie_basics['genres'].str.split(',')
movie_basics_genres = movie_basics.explode('genres')
movie_basics_genres
```

In [ ]:
```
#group genres, domestic gross, studio

moviebasics_group_table = (
    movie_basics_genres
    .groupby('genres')
    .sum()
    .reset_index()
    .sort_values('domestic_gross_x', ascending = False)[['genres', 'domestic_gro
)
moviebasics_group_table
```

In [ ]:
```
sns.set_style('darkgrid')
sns.set_palette('Set2')

sns.barplot(data=moviebasics_group_table, x="domestic_gross_x", y="genres", ci=N
sns.set(rc = {'figure.figsize':(15,15)})

plt.title('Top Grossing Movie Genres', fontsize=35, fontname='Arial')
```

```
plt.xlabel('Total Gross (Hundred Million USD)', fontsize=20, fontname='Arial')
plt.ylabel('Genres', fontsize=20, fontname='Arial')
plt.xticks(fontsize=15, fontname='Arial')
plt.yticks(fontsize=15, fontname='Arial')


sns.despine()
plt.show()
```

Rotten Tomatoes

I'll be using Pandas to open two Rotten Tomatoes datasets.

In [ ]:
```
rt_reviews = pd.read_csv('rt.reviews.tsv', delimiter="\t", header=0, encoding="u
rt_reviews.head(6)
```

In [ ]:
```
rt_info = pd.read_csv('rt.movie_info.tsv', delimiter = '\t', header=0, encoding=
rt_info.head()
```

Both dataset contains different columns expect for one, which is id. I will inspect the id column to confirm that these id numbers correlate to the same movie.

In [ ]:
```
rt_reviews['id'][990:999]
```

In [ ]:
```
rt_reviews['review'][995:1000]
```

In [ ]:
```
rt_info['synopsis'][18]
```

In [ ]:
```
rt_info['id'][18]
```

The movie ids are a match. Next I will merge the the movie id columns from both datasets.

In [ ]:
```
#The movie IDs from RT_reviews and RT_info match so merge on ID
rotten_tomatoes_df = rt_info.merge(rt_reviews, how='inner', on='id')
rotten_tomatoes_df.head(100)
```

I created a new dataframe to reflect the columns that I need for my analysis.

In [ ]:
```
rt_subset = rotten_tomatoes_df[['id', 'rating_x', 'genre', 'review', 'fresh']]
rt_subset.head()
```

In [ ]:
```
#What genre has the most "fresh" review?
#need count of frequency for fresh values based on content rating and genre
#to count the number of fresh values I would need to create a for loop
```

Changed column names to format data.

In [ ]:
```
rt_subset.rename(columns = {'fresh':'rating', 'rating_x':'contentrating'}, inpla
```

I used the explode function to separate genres.

In [ ]:
```
rt_subset['genre'] = rt_subset['genre'].str.split("|")
rt_subset_2 = rt_subset.explode('genre')
```

I created a dictionary to assign numerical values to fresh and rotten ratings. Numerical values are now in a new column called numeric_rating.

In [ ]:
```
rating_map = {'fresh': 1, 'rotten' : -1}
rt_subset_2['numeric_rating'] = rt_subset_2['rating'].map(rating_map)
rt_subset_2
```

In [ ]:
```
genre_numeric_rating = (
    rt_subset_2
    .groupby('genre')
    .sum()
    .reset_index()
    .sort_values('numeric_rating', ascending = False)[['genre', 'numeric_rating'
)
genre_numeric_rating
```

In [ ]:
```
#save content rating as new df
content_numeric_rating = (
    rt_subset_2
    .groupby('numeric_rating')
    .sum()
    .reset_index()
    .sort_values('numeric_contentrating', ascending = False)[['numeric_rating',
)
content_numeric_rating
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
sns.set_style('darkgrid')
sns.set_palette('Set2')

sns.barplot(data=genre_numeric_rating, x="numeric_rating", y="genre", ci=None)
sns.set(rc = {'figure.figsize':(20,15)})

plt.title('Genres With The Most Fresh Rating', fontsize=35, fontname='Arial')
plt.xlabel('Rating', fontsize=20, fontname='Arial')
plt.ylabel('Genres', fontsize=20, fontname='Arial')
plt.xticks(fontsize=15, fontname='Arial')
plt.yticks(fontsize=15, fontname='Arial')


sns.despine()
plt.show()
```

To find the content rating with the most fresh reviews, I'll create a content rating dictionary and assign numeric values.

In [ ]:
```
#What movie content rating has the most fresh reviews


contentrating_map = {'R': 0, 'PG' : 1, 'PG-13' : 2, 'NR' : 3, 'G' : 4}
rt_subset_2['numeric_contentrating'] = rt_subset_2['contentrating'].map(contentr
rt_subset_2
```

In [ ]:
```
#What movie content rating has the most fresh reviews
sns.set_style('darkgrid')
```

```
sns.set_palette('Set2')

sns.barplot(data=rt_subset_2, x="contentrating", y="numeric_rating", ci=None)
sns.set(rc = {'figure.figsize':(15,15)})

plt.title('Movie Content Rating With Most Fresh Reviews', fontsize=35, fontname=
plt.xlabel('Content Rating', fontsize=20, fontname='Arial')
plt.ylabel('Rating', fontsize=20, fontname='Arial')
plt.xticks(fontsize=15, fontname='Arial')
plt.yticks(fontsize=15, fontname='Arial')


sns.despine()
plt.show()
```