Project Notebook

April 24, 2024

You may use this notebook for your project or you may develop your project on your own machine. Either way, be sure to submit all your code to Vocareum via this notebook or upload any code used for your project as a part of the sumbission.

If you intend to use this notebook for your report (pdf) submission; be sure to look into markdown text for any discussion you need: Jupyter Documentation

1 Reading all the csv files and combing into a database

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtick
import re
import time
sns.set()
#
action_movie = pd.read_csv('action.csv')
crime_movie = pd.read_csv('crime.csv')
adventure_movie = pd.read_csv('adventure.csv')
thriller_movie = pd.read_csv('thriller.csv')
family_movie = pd.read_csv('family.csv')
mystery_movie = pd.read_csv('mystery.csv')
scifi_movie = pd.read_csv('scifi.csv')
history_movie = pd.read_csv('history.csv')
sports_movie = pd.read_csv('sports.csv')
animation_movie = pd.read_csv('animation.csv')
war_movie = pd.read_csv('war.csv')
biography_movie = pd.read_csv('biography.csv')
horror_movie = pd.read_csv('horror.csv')
fantasy_movie = pd.read_csv('fantasy.csv')
romance_movie = pd.read_csv('romance.csv')
film_noir_movie = pd.read_csv('film-noir.csv')

# Add genre column to each dataframe
action_movie['genre'] = 'Action'
```

```
crime_movie['genre'] = 'Crime'
adventure_movie['genre'] = 'Adventure'
thriller_movie['genre'] = 'Thriller'
family_movie['genre'] = 'Family'
mystery_movie['genre'] = 'Mystery'
scifi_movie['genre'] = 'Sci-Fi'
history_movie['genre'] = 'History'
sports_movie['genre'] = 'Sports'
animation_movie['genre'] = 'Animation'
war_movie['genre'] = 'War'
biography_movie['genre'] = 'Biography'
horror_movie['genre'] = 'Horror'
fantasy_movie['genre'] = 'Fantasy'
romance_movie['genre'] = 'Romance'
film_noir_movie['genre'] = 'Film-Noir'
# Concatenate all dataframes
df = pd.concat([action_movie, crime_movie, adventure_movie, thriller_movie,
                family_movie, mystery_movie, scifi_movie, history_movie,
                sports_movie, animation_movie, war_movie, biography_movie,
                horror_movie, fantasy_movie, romance_movie, film_noir_movie])

# Reset index
df = df.reset_index(drop=True)

# Preview the dataframe
#df
```

2 Cleaning the Data

Remove unwanted values not needed for evaluation. Preparing and formatting the dataframe.

In [2]:
```
# copy of the DataFrame to avoid SettingWithCopyWarning if slicing is used earlier
df = df.copy()

# Remove unwanted year values
unwanted_values = ['I', 'II', 'V', 'III', 'VII', 'IV', 'XXIII', 'IX', 'XV', 'VI', 'X',
df = df[~df['year'].isin(unwanted_values)]

# Fill NaN values and handle data types directly using .loc for safer operations
df.loc[:, 'year'] = df['year'].fillna(0).astype(int)
df.loc[:, 'year'] = df.loc[:, 'year'].replace(0, np.nan)

# Drop rows with NaN values in 'year' column safely
df.dropna(subset=['year'], inplace=True)

# Convert 'year' back to integer, ensure it's done in the DataFrame
df.loc[:, 'year'] = df['year'].astype(int)
```

```
# Drop the 'movie_id' column
df = df.drop(columns=['movie_id'])
df = df.drop(columns=['director_id'])
df = df.drop(columns=['star_id'])

# Display the DataFrame
#df
df_copy = df
#df
```

2.1 Format the runtimes

```
In [3]: # Drop rows where 'runtime' is NaN
        df = df.dropna(subset=['runtime'])

        # operate on a copy to avoid SettingWithCopyWarning if df is derived from another Data
        df = df.copy()

        # Convert all 'runtime' entries to string and replace unwanted characters using .loc f
        df.loc[:, 'runtime'] = df['runtime'].astype(str).str.replace('min', '', regex=False)
        df.loc[:, 'runtime'] = df.loc[:, 'runtime'].str.replace(',', '', regex=False)

        # Convert the cleaned 'runtime' strings to integers using .loc for direct modification
        df.loc[:, 'runtime'] = df['runtime'].astype(int)

        # Convert 'runtime' from integer minutes to timedelta using .loc for direct modificati
        df.loc[:, 'runtime'] = pd.to_timedelta(df['runtime'], unit='m')

        # Print DataFrame to confirm changes
        #df
```

2.2 Format the names of actors

```
In [4]: # First, ensure all entries in 'star' are strings (convert NaNs to an empty string or
        df['star'] = df['star'].astype(str)

        # Replace newlines and extra spaces
        df['star'] = df['star'].str.replace('\n', ', ', regex=False)  # Replace newlines with
        df['star'] = df['star'].str.replace(',  ', ', ', regex=False)  # Clean up potential do

        # Strip whitespace from each name and rejoin with clean commas
        df['star'] = df['star'].apply(lambda x: ', '.join(name.strip() for name in x.split(','

        # Split 'star' into individual names and explode into rows
        star_list = df['star'].str.split(', ').explode()

        # Count occurrences of each star
```

```
star_counts = star_list.value_counts()

# Convert to DataFrame for better manipulation and storage
star_counts_df = star_counts.reset_index()
star_counts_df.columns = ['Star Name', 'Occurrences']
#Drop if blank or nan and reindex
star_counts_df = star_counts_df[(star_counts_df['Star Name'].str.strip() != '') & (sta
star_counts_df.reset_index(drop=True, inplace=True)

# Display the resulting DataFrame
#print(star_counts_df.head())
#df
```

3 The dataframe that we are working with

In [5]: df

Out[5]:

	movie_name	year	certificate	runtime	genre	rating	description	director
0	Black Panther: Wakanda Forever	2022	PG-13	0 days 02:41:00	Action	6.9	The people of Wakanda fight to protect their h...	Ryan Coogler
1	Avatar: The Way of Water	2022	PG-13	0 days 03:12:00	Action	7.8	Jake Sully lives with his newfound family form...	James Cameron
2	Plane	2023	R	0 days 01:47:00	Action	6.5	A pilot finds himself caught in a war zone aft...	Jean-François Richet
3	Everything Everywhere All at Once	2022	R	0 days 02:19:00	Action	8.0	A middle-aged Chinese immigrant is swept up in...	
5	Ant-Man and the Wasp: Quantumania	2023	PG-13	0 days 02:05:00	Action	6.6	Scott Lang and Hope Van Dyne, along with Hank ...	
...	...	...	...	...	...	...	...	
368295	Black Diamonds	1940	NaN	0 days 01:00:00	Film-Noir	5.5	A reporter on a visit to his hometown hears of...	
368296	The Gentleman from Louisiana	1936	NaN	0 days 01:07:00	Film-Noir	4.5	In Victorian-era USA, a horse-jockey becomes a...	
368297	El cerco	1955	NaN	0 days 01:17:00	Film-Noir	6.2	A group of robbers assault a factory in the po...	
368298	Three Silent Men	1940	NaN	0 days 01:12:00	Film-Noir	5.0	Foreign scientist is selling a secret weapon t...	
368299	Destination Big House	1950	NaN	0 days 01:00:00	Film-Noir	6.3	School teacher Janet Brooks innocently involve...	

```
3       Dan Kwan, \nDaniel Scheinert
5                       Peyton Reed
...                               ...
368295               Christy Cabanne
368296                 Irving Pichel
368297               Miguel Iglesias
368298                Thomas Bentley
368299                  George Blair

                                                     star     votes  \
0       Letitia Wright, , Lupita Nyong'o, , Danai Guri...  204835.0
1       Sam Worthington, , Zoe Saldana, , Sigourney We...  295119.0
2       Gerard Butler, , Mike Colter, , Tony Goldwyn, ...   26220.0
3       Michelle Yeoh, , Stephanie Hsu, , Jamie Lee Cu...  327858.0
5       Paul Rudd, , Evangeline Lilly, , Jonathan Majo...    5396.0
...                                                   ...       ...
368295  Richard Arlen, , Andy Devine, , Kathryn Adams,...      33.0
368296  Eddie Quillan, , Charles 'Chic' Sale, , Charlo...      21.0
368297  José Guardiola, , Isabel de Castro, , Ángel Jo...      46.0
368298  Sebastian Shaw, , Derrick De Marney, , Patrici...      79.0
368299  Dorothy Patrick, , Robert Rockwell, , Jimmy Ly...      40.0

        gross(in $)
0               NaN
1               NaN
2               NaN
3               NaN
5               NaN
...             ...
368295          NaN
368296          NaN
368297          NaN
368298          NaN
368299          NaN

[254475 rows x 11 columns]
```

3.1 Clean and store the directors column by occurrence

```
In [6]: # Clean the 'director' column by replacing newlines with ', '
        df['director'] = df['director'].str.replace('\n', ', ', regex=False)

        # Clean any potential double spaces just in case
        df['director'] = df['director'].str.replace(',  ', ', ', regex=False)

        # Split 'director' into individual names and explode into rows
        director_list = df['director'].str.split(', ').explode()
```

```
# Count occurrences of each director
director_counts = director_list.value_counts()

# Convert to DataFrame for better manipulation and storage
director_counts_df = director_counts.reset_index()
director_counts_df.columns = ['Director Name', 'Occurrences']

director_counts_df = director_counts_df[(director_counts_df['Director Name'].str.strip
director_counts_df.reset_index(drop=True, inplace=True)

# Display the resulting DataFrame
#print(director_counts_df.head())
```

3.2 Clean and store popular genres over time by year

In [7]:
```
# Ensure the data types are correct
df['year'] = df['year'].astype(int)
df['genre'] = df['genre'].astype(str)

# Initialize a new DataFrame to store the results
popular_genres = pd.DataFrame(columns=['year', 'popular_genre', 'number_of_release'])
popular_genres_filtered = popular_genres[~popular_genres['year'].isin([2025, 2024])]

# Group the data by 'year' and 'genre', count occurrences, and reset index
genre_counts = df.groupby(['year', 'genre']).size().reset_index(name='number_of_releas

# Find the most popular genre for each year
popular_genres = genre_counts.loc[genre_counts.groupby('year')['number_of_release'].id

# Reset the index of the final DataFrame for clean output
popular_genres.reset_index(drop=True, inplace=True)

# Sort the results by year for better readability
popular_genres.sort_values('year', ascending=False, inplace=True)

# Display the resulting DataFrame
#popular_genres
```

4 Visualizations & Analysis

4.1 Most popular genre for each year

In [8]:
```
popular_genres = popular_genres.reset_index(drop=True)
popular_genres
```

Out[8]:

	year	genre	number_of_release
0	2025	Action	2
1	2024	Adventure	2

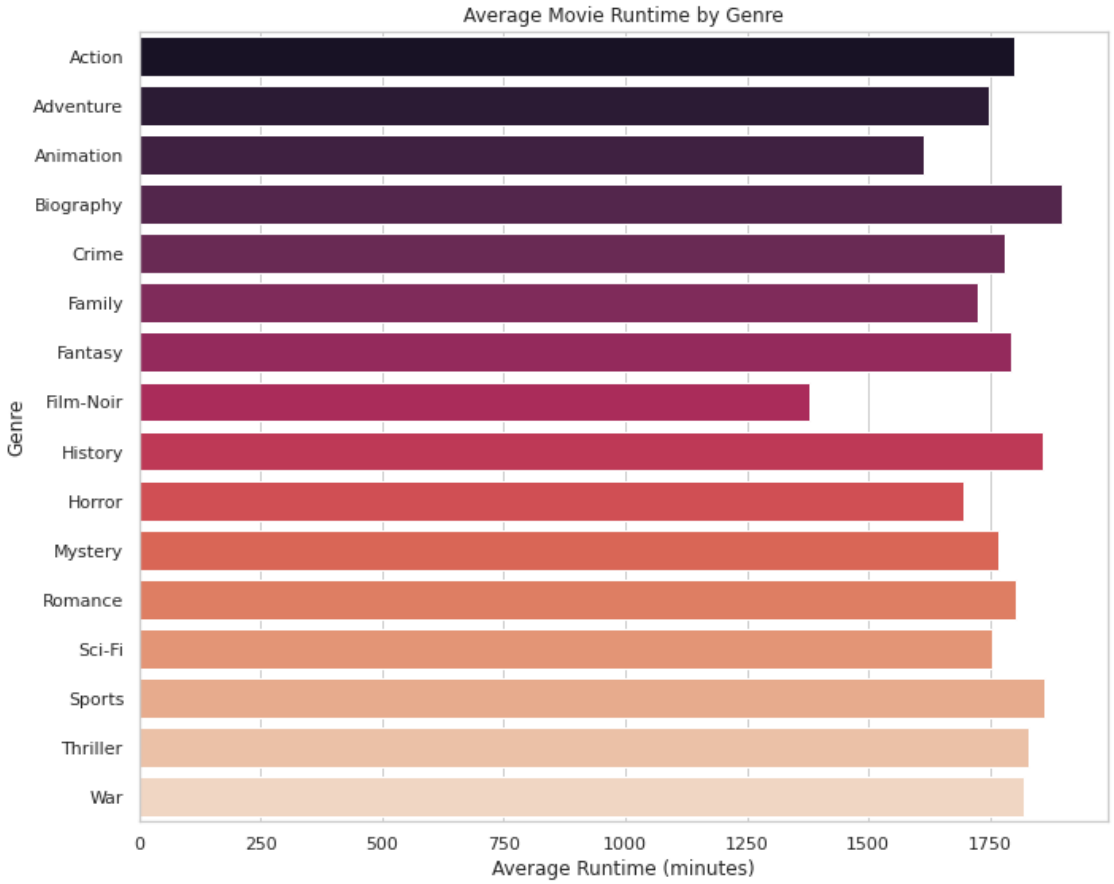
```
2    2023   Thriller               290
3    2022   Thriller              1565
4    2021   Thriller              1365
..    ...        ...                ...
116  1908  Adventure                 1
117  1907  Biography                 1
118  1906     Action                 1
119  1903  Biography                 1
120  1894    Romance                 1

[121 rows x 3 columns]
```

Note: The years 2024 and 2025 do not have sufficient data entries since dataset has not been updated for 2024 and 2025 data is based on movies that may still be in production.

4.2 What genre has the highest run time?

Visualize genre and their runtimes by using grouby

```
In [9]: # Step 1: Ensure 'runtime' is treated as a string and convert runtime to minutes
        df['runtime'] = df['runtime'].astype(str)  # Convert to string if not already
        df['runtime_minutes'] = df['runtime'].str.extract(r'(\d+):(\d+):(\d+)').apply(
            lambda x: int(x[1]) * 60 + int(x[2]), axis=1)

        # Step 2: Calculate average runtime by genre
        average_runtime_by_genre = df.groupby('genre')['runtime_minutes'].mean()
        sns.set(style="whitegrid", rc={"figure.figsize": (10, 8)})

        # Create a bar plot
        sns.barplot(
            x=average_runtime_by_genre.values,
            y=average_runtime_by_genre.index,
            palette="rocket"
        )

        # Labeling the axes and title
        plt.xlabel('Average Runtime (minutes)')
        plt.ylabel('Genre')
        plt.title('Average Movie Runtime by Genre')

        # Adjust layout to not cut off labels or titles and display the plot
        plt.tight_layout()
        plt.show()
```



4.3 Highest rated movies by year on Imdb

Creating a new dataframe with respect to ratings and find the highest rating for each year

```
In [10]: import pandas as pd

         df = df.sort_values(by='year', ascending=False)

         # Create an empty DataFrame with the same columns as df
         highest_rated_movies = pd.DataFrame(columns=df.columns)

         # Loop through each unique year in the 'year' column
         for year in df['year'].unique():
             df_year = df[df['year'] == year]  # Filter by year
             if not df_year.empty:  # Check if the resulting DataFrame is not empty
                 df_year_sorted = df_year.sort_values(by='rating', ascending=False)  # Sort by
                 highest_rated_movie = df_year_sorted.iloc[0]  # Get the row with the highest
                 # Append the row to the new DataFrame using concat
                 highest_rated_movies = pd.concat([highest_rated_movies, pd.DataFrame([highest_
```

```
# Remove rows where 'rating' might be NaN
highest_rated_movies = highest_rated_movies.dropna(subset=['rating'])

# Reset the index of the new DataFrame
highest_rated_movies.reset_index(drop=True, inplace=True)

# Display the DataFrame containing the highest rated movies by year
highest_rated_movies
```

Out[10]:

	movie_name	year	certificate	runtime	genre	rating	description	director
0	The Universal Quest	2023	NaN	0 days 02:25:00	Adventure	10.0	"The Universal Quest" is a sci-fi epic that ta...	Juwel Chowdhury
1	Love Mein	2022	NaN	0 days 01:00:00	Action	10.0	This is 2016's love story where the only daugh...	Kumud Pant
2	Low Heat	2021	NaN	0 days 00:48:00	Action	9.9	The foul mouth quick quipping Jamal becomes in...	John Carroll DeShazier
3	The Puzzling Secret	2020	NaN	0 days 02:50:00	Mystery	10.0	Add a Plot	Juwel Chowdhury
4	Journey of Eternity	2019	NaN	0 days 01:40:00	History	10.0	After his beloved fiancé was forcibly taken aw...	Frank Gilbert
..	...	...	...	...	...	...	...	...
114	The Fairylogue and Radio-Plays	1908	NaN	0 days 02:00:00	Fantasy	5.2	L. Frank Baum would appear in a white suit and...	Francis Boggs, , Otis Turner
115	Life and Passion of Christ	1907	Not Rated	0 days 00:45:00	History	6.7	Depicting well-known incidents in the life of ...	Ferdinand Zecca
116	The Story of the Kelly Gang	1906	Not Rated	0 days 01:10:00	History	6.0	Story of Ned Kelly, an infamous 19th-century A...	Charles Tait
117	The Passion Play	1903	NaN	0 days 00:45:00	Biography	6.5	The story of Jesus Christ from the proclamatio...	Lucien Nonguet, , Ferdinand Zecca
118	Miss Jerry	1894	NaN	0 days 00:45:00	Romance	5.3	The adventures of a female reporter in the 1890s.	Alexander Black

```
                                                  star  votes gross(in $)  \
0                        Juwel Chowdhury, , Ricky Riyaf    6.0         NaN
1    Kumud Pant, , Pooja Kimaya, , Atul Sharma, , B...    6.0         NaN
2    Clinton D. Walker, , Dustin Bennett, , Ricardo...    7.0         NaN
3                        Juwel Chowdhury, , Ricky Riyaf    6.0         NaN
4    Abee Sargis, , Tarik Akreyî, , Kadhim Al-Qurai...    9.0         NaN
..                                                 ...    ...         ...
114  L. Frank Baum, , Frank Burns, , George E. Wils...   67.0         NaN
115                                                nan   60.0         NaN
116  Elizabeth Tait, , John Tait, , Nicholas Brierl...  810.0         NaN
117                    Madame Moreau, , Monsieur Moreau  575.0         NaN
118  Blanche Bayliss, , William Courtenay, , Chaunc...  204.0         NaN

    runtime_minutes
0              1500
1                 0
2              2880
3              3000
4              2400
..              ...
114               0
115            2700
116             600
117            2700
118            2700

[119 rows x 12 columns]
```

4.3.1 Top rated movie of each year for the last 10 years

```
In [11]: top_Rmovies = highest_rated_movies.head(10)
         top_Rmovies = top_Rmovies[['movie_name', 'director', 'genre', 'rating', 'year']]
         top_Rmovies
```

```
Out[11]:                    movie_name                    director      genre  \
         0           The Universal Quest              Juwel Chowdhury  Adventure
         1                     Love Mein                  Kumud Pant     Action
         2                      Low Heat      John Carroll DeShazier     Action
         3           The Puzzling Secret              Juwel Chowdhury    Mystery
         4           Journey of Eternity                Frank Gilbert    History
         5         The Trees of the East                 Mike Ellwood   Thriller
         6  I Found My Heart in Sante Fe                 Bona Fajardo    Romance
         7              My Friend Violet         Matthew S. Robinson   Thriller
         8                  Heavy Makeup              Chris Morrissey     Horror
         9            The Killer Monroes  Joe Lujan, , Sharry Flaherty   Thriller

           rating  year
```

```
0   10.0  2023
1   10.0  2022
2    9.9  2021
3   10.0  2020
4   10.0  2019
5    9.8  2018
6    9.8  2017
7    9.6  2016
8    9.6  2015
9    9.6  2014
```

4.4 Highest Voted movies on Imdb by year

Create a new dataframe with the highest voted movies on Imdb

```
In [12]: # Sort the DataFrame by 'year' descending to process the latest year first
         df = df.sort_values(by='year', ascending=False)

         # Create an empty DataFrame with the same columns as df for storing results
         highest_voted_movies = pd.DataFrame(columns=df.columns)

         # Loop through each unique year in the 'year' column
         for year in df['year'].unique():
             # Filter the DataFrame by year
             df_year = df[df['year'] == year]
             # Check if the resulting DataFrame is not empty
             if not df_year.empty:
                 # Sort the yearly DataFrame by 'votes' in descending order to find the highes
                 df_year_sorted = df_year.sort_values(by='votes', ascending=False)
                 # Get the row with the highest votes
                 highest_voted_movie = df_year_sorted.iloc[0]
                 # Append the row to the new DataFrame using concat
                 highest_voted_movies = pd.concat([highest_voted_movies, highest_voted_movie.to

         # Reindex
         highest_voted_movies= highest_voted_movies.dropna(subset=['votes'])
         highest_voted_movies.reset_index(drop=True, inplace=True)

         # Display the DataFrame containing the highest voted movies by year
         highest_voted_movies
```

```
Out[12]:                         movie_name  year certificate          runtime  \
         0                           Pathaan  2023   Not Rated  0 days 02:26:00
         1                        The Batman  2022       PG-13  0 days 02:56:00
         2          Spider-Man: No Way Home  2021       PG-13  0 days 02:28:00
         3                             Tenet  2020       PG-13  0 days 02:30:00
         4                             Joker  2019           R  0 days 02:02:00
         ..                              ...   ...         ...              ...
```

```
114  The Fairylogue and Radio-Plays  1908         NaN  0 days 02:00:00
115      Life and Passion of Christ  1907   Not Rated  0 days 00:45:00
116     The Story of the Kelly Gang  1906   Not Rated  0 days 01:10:00
117                 The Passion Play  1903         NaN  0 days 00:45:00
118                       Miss Jerry  1894         NaN  0 days 00:45:00

        genre rating                                        description  \
0     Thriller    6.6  An Indian spy takes on the leader of a group o...
1       Action    7.8  When a sadistic serial killer begins murdering...
2       Sci-Fi    8.2  With Spider-Man's identity now revealed, Peter...
3       Sci-Fi    7.3  Armed with only one word, Tenet, and fighting ...
4     Thriller    8.4  A mentally troubled stand-up comedian embarks ...
..         ...    ...                                                ...
114  Adventure    5.2  L. Frank Baum would appear in a white suit and...
115  Biography    6.6  Depicting well-known incidents in the life of ...
116  Biography    6.0  Story of Ned Kelly, an infamous 19th-century A...
117  Biography    6.5  The story of Jesus Christ from the proclamatio...
118    Romance    5.3  The adventures of a female reporter in the 1890s.

                             director  \
0                      Siddharth Anand
1                          Matt Reeves
2                            Jon Watts
3                    Christopher Nolan
4                        Todd Phillips
..                                 ...
114       Francis Boggs, , Otis Turner
115                    Ferdinand Zecca
116                       Charles Tait
117  Lucien Nonguet, , Ferdinand Zecca
118                    Alexander Black

                                                 star      votes  \
0    Shah Rukh Khan, , Deepika Padukone, , John Abr...   120080.0
1    Robert Pattinson, , Zoë Kravitz, , Jeffrey Wri...   672146.0
2    Tom Holland, , Zendaya, , Benedict Cumberbatch...   770509.0
3    John David Washington, , Robert Pattinson, , E...   516838.0
4    Joaquin Phoenix, , Robert De Niro, , Zazie Bee...  1310698.0
..                                                 ...        ...
114  L. Frank Baum, , Frank Burns, , George E. Wils...       67.0
115                                                nan       61.0
116  Elizabeth Tait, , John Tait, , Nicholas Brierl...      818.0
117                   Madame Moreau, , Monsieur Moreau      575.0
118  Blanche Bayliss, , William Courtenay, , Chaunc...      204.0

     gross(in $) runtime_minutes
0            NaN            1560
1            NaN            3360
```

```
2    804747988.0            1680
3     58456624.0            1800
4    335451311.0             120
..           ...             ...
114          NaN               0
115          NaN            2700
116          NaN             600
117          NaN            2700
118          NaN            2700

[119 rows x 12 columns]
```

4.4.1 Top most voted movie by year for the last 10 years

```
In [13]: top_Vmovies = highest_voted_movies.head(10)
         top_Vmovies = top_Vmovies[['movie_name', 'director', 'genre', 'votes', 'year']]
         top_Vmovies
```

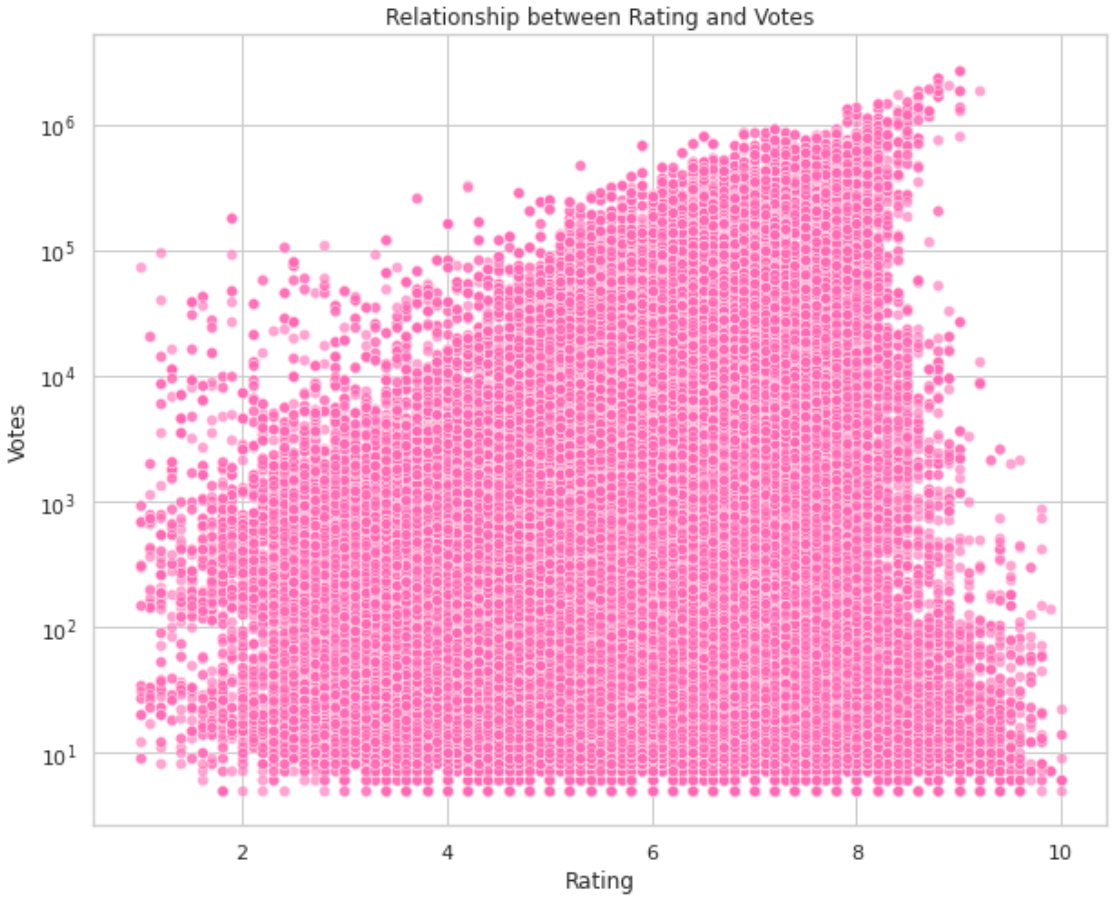
Out[13]:

	movie_name	director	genre	votes	year
0	Pathaan	Siddharth Anand	Thriller	120080.0	2023
1	The Batman	Matt Reeves	Action	672146.0	2022
2	Spider-Man: No Way Home	Jon Watts	Sci-Fi	770509.0	2021
3	Tenet	Christopher Nolan	Sci-Fi	516838.0	2020
4	Joker	Todd Phillips	Thriller	1310698.0	2019
5	Avengers: Infinity War	Anthony Russo, , Joe Russo	Sci-Fi	1095314.0	2018
6	Logan	James Mangold	Sci-Fi	772831.0	2017
7	Deadpool	Tim Miller	Action	1049045.0	2016
8	Mad Max: Fury Road	George Miller	Sci-Fi	1013611.0	2015
9	Interstellar	Christopher Nolan	Sci-Fi	1859284.0	2014

4.5 What is the relationship between votes and ratings?

I wanted to plot the relationship between total votes and ratings and how they are/are not different. Votes signifies how popular a movie may be. Ratings signifies how good the movie may be.

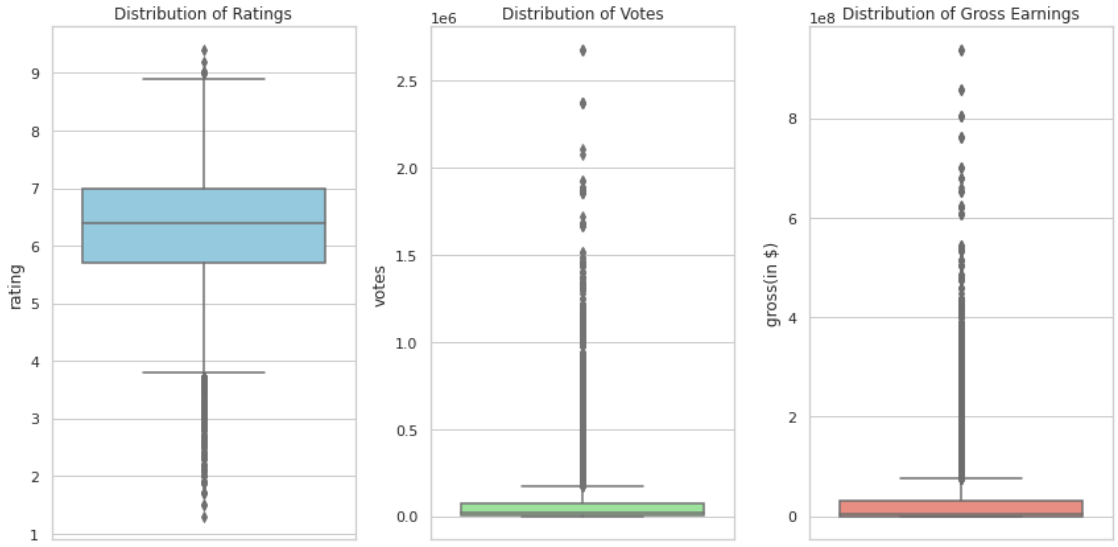
```
In [14]: # Cleaning
         df['rating'] = pd.to_numeric(df['rating'], errors='coerce')  # Convert rating to nume
         df['votes'] = pd.to_numeric(df['votes'], errors='coerce')  # Convert votes to numeric
         df.dropna(subset=['rating', 'votes'], inplace=True)  # Drop rows where either value i

         # Visualization
         plt.figure(figsize=(10, 8))
         sns.scatterplot(x='rating', y='votes', data=df, alpha=0.6, color= 'hotpink')
         plt.title('Relationship between Rating and Votes')
         plt.xlabel('Rating')
         plt.ylabel('Votes')
         plt.yscale('log')
         plt.grid(True)
         plt.show()
```



4.6 Identifying a correlation between rates, votes and gross earnings of a movie.

How are all 3 of these factors distributed and related? This helps for an industry analysis. I chose boxplot to plot these 3 variables from df to see their distribution.

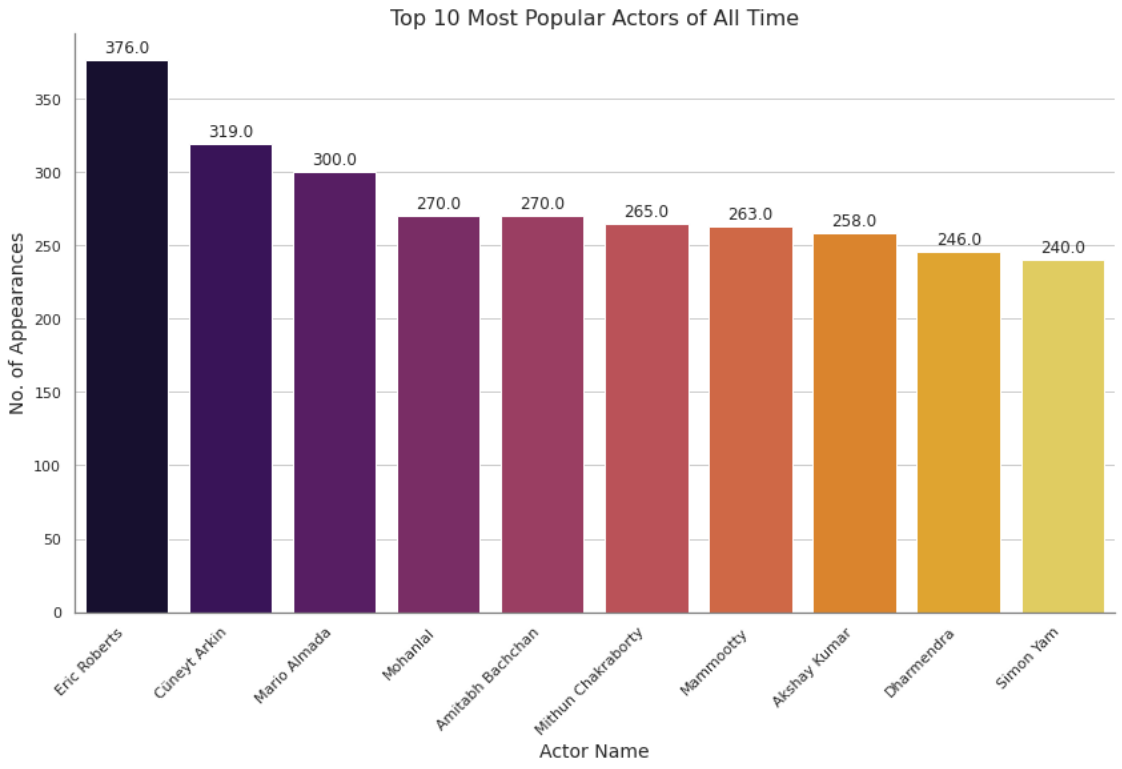
In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame containing the movie data

# First, let's ensure the necessary data columns are in the right format and clean:
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')  # Convert to numeric, ha
df['votes'] = pd.to_numeric(df['votes'], errors='coerce')
df['gross(in $)'] = pd.to_numeric(df['gross(in $)'], errors='coerce')

# Drop rows with NaN values in these columns for cleaner visualization
df = df.dropna(subset=['rating', 'votes', 'gross(in $)'])

# Create box plots for ratings, votes, and gross earnings
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
sns.boxplot(y='rating', data=df, color='skyblue')
plt.title('Distribution of Ratings')

plt.subplot(1, 3, 2)
sns.boxplot(y='votes', data=df, color='lightgreen')
plt.title('Distribution of Votes')

plt.subplot(1, 3, 3)
sns.boxplot(y='gross(in $)', data=df, color='salmon')
plt.title('Distribution of Gross Earnings')

plt.tight_layout()
plt.show()
```



4.7 Most popular actors of all time

Visualization created using the stars column cleaned and stored in a separate column with each star being stored separately with the total number of times they occur in a datset.

```
In [16]: import matplotlib.pyplot as plt
         import numpy as np  # For generating a range of colors

         # Assuming top_actors is already defined as the top 10 entries from star_counts_df
         top_actors = star_counts_df.head(10)

         # Create the bar plot with each bar having a different color
         plt.figure(figsize=(14, 8))  # Make the figure larger: width=14 inches, height=8 inch
         barplot = sns.barplot(x='Star Name', y='Occurrences', data=top_actors, palette="infer

         # Set the x-ticks to be more readable
         plt.xticks(rotation=45, ha='right')  # ha='right' helps with alignment of long names

         # Setting labels and title
         plt.xlabel("Actor Name", fontsize=14)  # Increase font size for clarity
         plt.ylabel("No. of Appearances", fontsize=14)
         plt.title("Top 10 Most Popular Actors of All Time", fontsize=16)

         # Adding value labels on each bar
         for p in barplot.patches:
             barplot.annotate(format(p.get_height(), '.1f'),
                              (p.get_x() + p.get_width() / 2., p.get_height()),
                              ha = 'center', va = 'center',
                              xytext = (0, 9),
                              textcoords = 'offset points')

         # Remove background color
         plt.gca().set_facecolor('none')  # Set the background color to none (transparent with
         plt.gca().spines['top'].set_visible(False)  # Hide the top spine
         plt.gca().spines['right'].set_visible(False)  # Hide the right spine
         plt.gca().spines['left'].set_color('gray')  #  change the color of the left spine to
         plt.gca().spines['bottom'].set_color('gray')  #  change the color of the bottom spine

         # Show the plot
         plt.show()
```

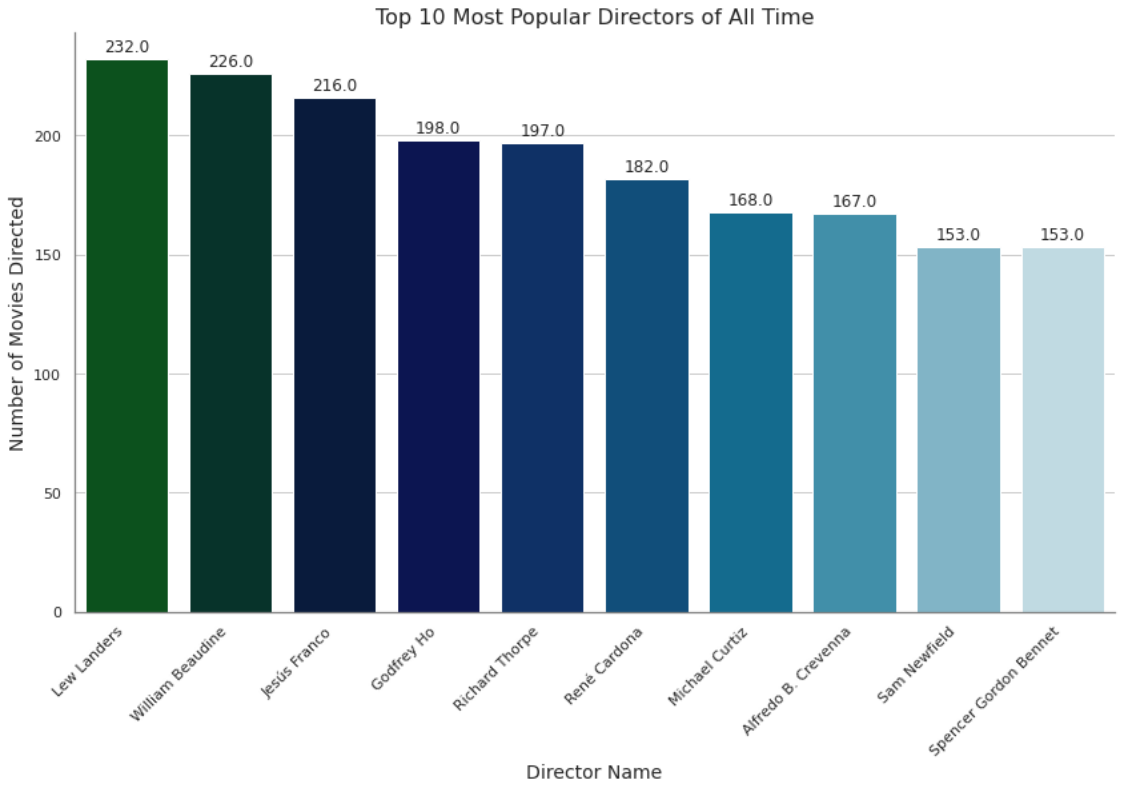


4.8 Most popular directors of all time, by number of movies directed

Visualization created using the directors column cleaned and stored in a separate column with each director being stored separately with the total number of times they occur in a datset.

```
In [17]: top_dir = director_counts_df.head(10)

         # Create the bar plot with each bar having a different color
         plt.figure(figsize=(14, 8))
         barplot = sns.barplot(x='Director Name', y='Occurrences', data=top_dir, palette="ocea

         # Set the x-ticks to be more readable
         plt.xticks(rotation=45, ha='right')

         # Setting labels and title
         plt.xlabel("Director Name", fontsize=14)  # Increase font size for clarity
         plt.ylabel("Number of Movies Directed", fontsize=14)
         plt.title("Top 10 Most Popular Directors of All Time", fontsize=16)

         # Adding value labels on each bar
         for p in barplot.patches:
             barplot.annotate(format(p.get_height(), '.1f'),
                              (p.get_x() + p.get_width() / 2., p.get_height()),
```

```
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

    # Remove background color
    plt.gca().set_facecolor('none')  # Set the background color to none (transparent with
    plt.gca().spines['top'].set_visible(False)  # Hide the top spine
    plt.gca().spines['right'].set_visible(False)  # Hide the right spine
    plt.gca().spines['left'].set_color('gray')  #  change the color of the left spine to
    plt.gca().spines['bottom'].set_color('gray')  #  change the color of the bottom spine

    # Show the plot
    plt.show()
```



4.9 Get the highest grossing movie for each year

In [18]:
```
# Initialize an empty DataFrame with the same columns as df
highest_revenue_movies = pd.DataFrame(columns=df.columns)

# Iterate through each unique year in the DataFrame, excluding the years 2025 and 202
for year in df['year'].unique():
    if year not in [2025, 2024]:  # Exclude the specified years
```

```
            df_year = df[df['year'] == year]  # Filter the DataFrame by the current year
            if not df_year.empty:  # Check if the resulting DataFrame is not empty
                # Sort by 'gross(in $)' in descending order to find the movie with the hi
                df_year_sorted = df_year.sort_values(by='gross(in $)', ascending=False)
                highest_revenue_movie = df_year_sorted.iloc[0]  # Get the first row, the
                # Create a DataFrame from the highest revenue movie and concatenate it wi
                highest_revenue_movies = pd.concat([highest_revenue_movies, highest_revenu

columns_to_drop = ['description', 'director','star']
highest_revenue_movies.drop(columns=columns_to_drop, inplace=True)

# Sort the DataFrame by year in descending order
highest_revenue_movies = highest_revenue_movies.sort_values(by='year', ascending=Fals
highest_revenue_movies = highest_revenue_movies.reset_index(drop=True)

highest_revenue_movies
```

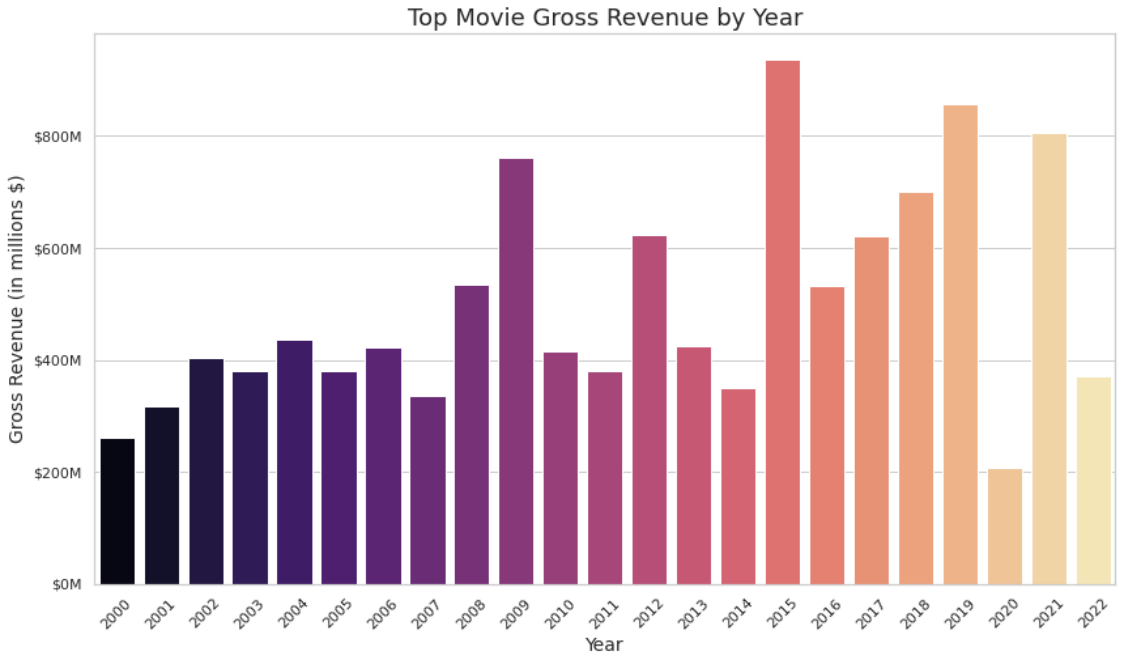
Out[18]:

	movie_name	year	certificate	runtime	genre	rating	votes	gross(in $)	runtime_minutes
0	Minions: The Rise of Gru	2022	PG	0 days 01:27:00	Animation	6.5	70007.0	369695210.0	1620
1	Spider-Man: No Way Home	2021	PG-13	0 days 02:28:00	Adventure	8.2	770492.0	804747988.0	1680
2	Bad Boys for Life	2020	R	0 days 02:04:00	Action	6.5	163933.0	206305244.0	240
3	Avengers: Endgame	2019	PG-13	0 days 03:01:00	Action	8.4	1148100.0	858373000.0	60
4	Black Panther	2018	PG-13	0 days 02:14:00	Adventure	7.3	785813.0	700059566.0	840
..	...	...	...	...	...	...	...	...	...
104	A Romance of the Redwoods	1917	Not Rated	0 days 01:10:00	History	6.1	835.0	424719.0	600
105	20,000 Leagues Under the Sea	1916	Passed	0 days 01:25:00	Adventure	6.1	1860.0	8000000.0	1500
106	The Birth of a Nation	1915	TV-PG	0 days 03:15:00	War	6.2	25213.0	10000000.0	900
107	The Squaw Man	1914	Not Rated	0 days 01:14:00	Romance	5.7	995.0	533446.0	840
108	Traffic in Souls	1913	TV-PG	0 days 01:28:00	Crime	6.0	696.0	430000.0	1680

[109 rows x 9 columns]

4.10 Highest gross values for the last 10 years

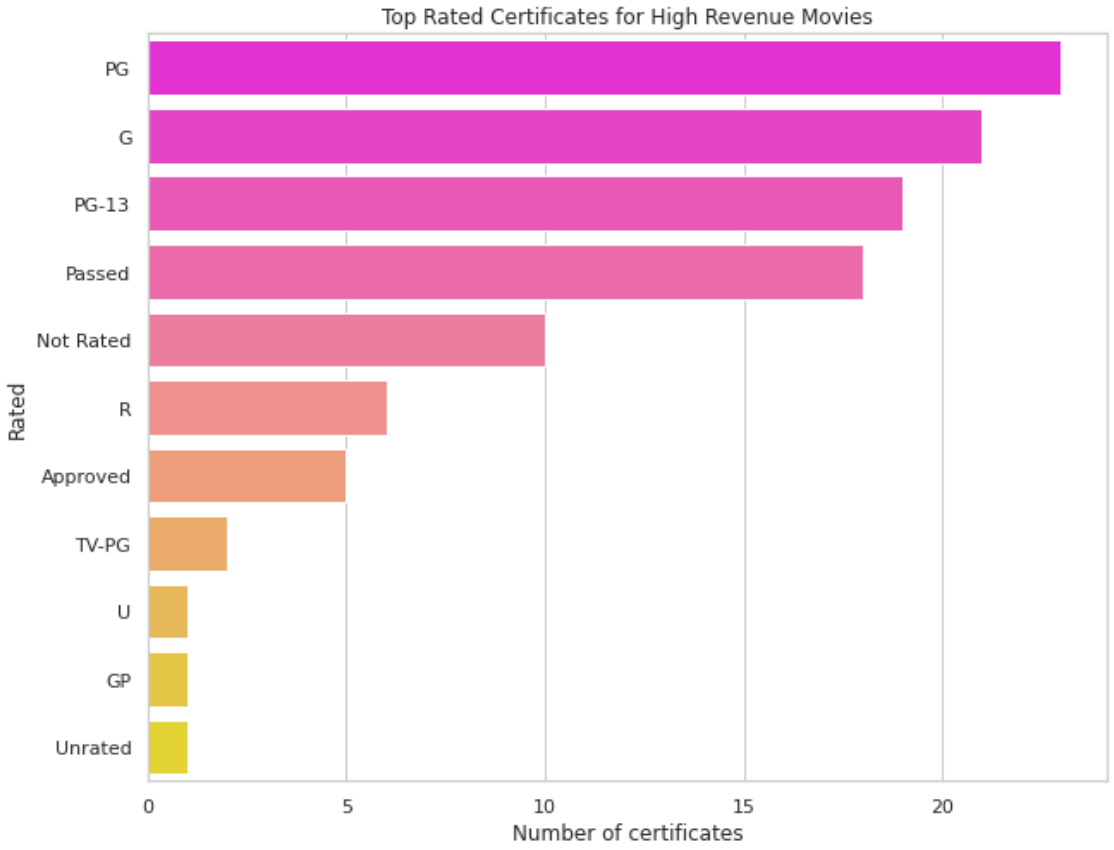
In [19]:
```
# Create the bar chart using seaborn with a larger figure size for better visibility
top_grossing = highest_revenue_movies.sort_values('year', ascending=False).head(23)
plt.figure(figsize=(14, 8))  # Set the figure size (width, height) in inches
sns.set(style="whitegrid")  # Apply the whitegrid style for a clean background with g

# Create the bar plot
ax = sns.barplot(x='year', y='gross(in $)', data=top_grossing, palette="magma")

# Format y-axis tick labels in millions and adjust label format to show commas for th
ax.yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'${x/1e6:,.0f}M'))

# Set the title and axis labels with increased font size for clarity
ax.set_title('Top Movie Gross Revenue by Year', fontsize=18)
ax.set_xlabel('Year', fontsize=14)
ax.set_ylabel('Gross Revenue (in millions $)', fontsize=14)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

#Adjust the margins to ensure the x-axis labels are fully visible
plt.subplots_adjust(bottom=0.15)

# Display the plot
plt.show()
```



4.11 What certificates do high revenue movies earn more commonly?

What kind of movies are most watched/most popular?

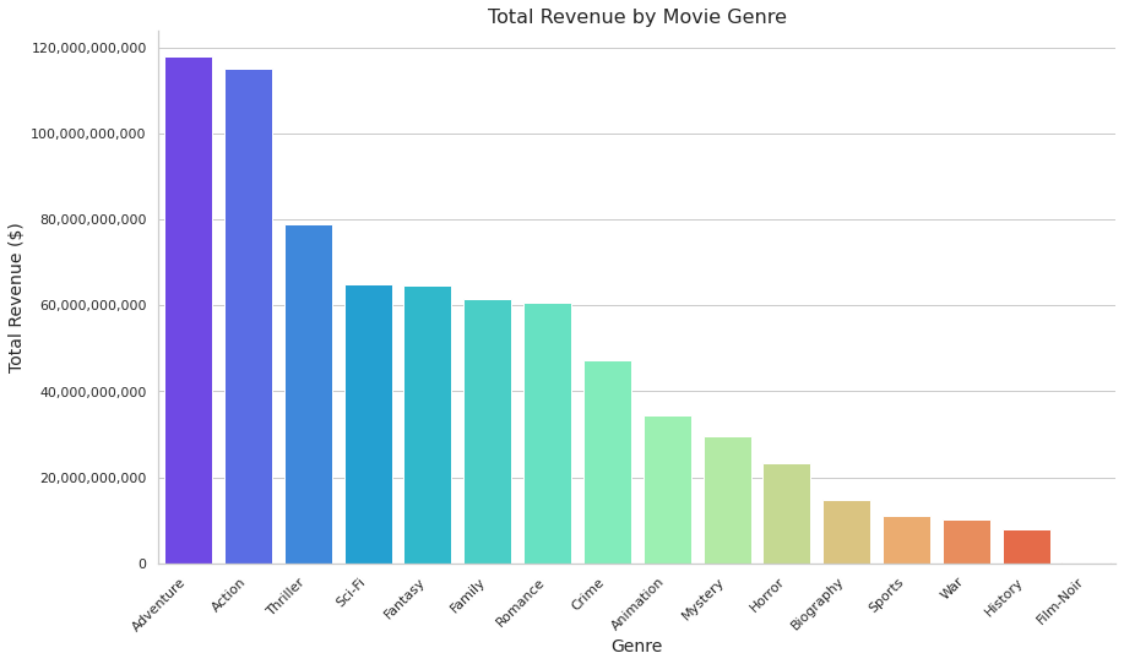
```
In [20]: certificate_counts_high_revenue = highest_revenue_movies['certificate'].value_counts()

         certificate_counts_high_revenue_df = certificate_counts_high_revenue.to_frame().reset_
         certificate_counts_high_revenue_df.columns = ['certificate', 'count']

         plt.figure(figsize=(10, 8))
         sns.barplot(x='count', y='certificate', data=certificate_counts_high_revenue_df, palet
         plt.title('Top Rated Certificates for High Revenue Movies')
         plt.xlabel('Number of certificates')
         plt.ylabel('Rated')
         plt.show()
```



4.12 What genre of movie grosses the highest?

I am curious to see what genre of movie grosses the highest by plotting the total gross values for each genre using a bar graph and using groupby to calculate total gross for each genre

In [21]:
```
# Group by 'genre' and sum up the revenues
genre_revenue = df.groupby('genre')['gross(in $)'].sum().reset_index()

# Sort the results by revenue in descending order
genre_revenue_sorted = genre_revenue.sort_values('gross(in $)', ascending=False)

# Create the bar plot
plt.figure(figsize=(14, 8))
sns.set(style="whitegrid")
ax = sns.barplot(x='genre', y='gross(in $)', data=genre_revenue_sorted, palette="rainb

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha="right")

# Set plot title and axis labels
plt.title('Total Revenue by Movie Genre', fontsize=16)
plt.xlabel('Genre', fontsize=14)
plt.ylabel('Total Revenue ($)', fontsize=14)

# Format the y-axis as currency
ax.get_yaxis().set_major_formatter(
    plt.matplotlib.ticker.FuncFormatter(lambda x, p: format(int(x), ','))
)

# Remove the right and top spines
sns.despine()

# Show the plot
plt.show()
```



4.13 Who is the best director?

I can explore who is the best director by getting average votes and average ratings for each director. I will only consider them for further analysis if they have more than 20 movies in order to make the calculations more fair.

```
In [22]: # Create a copy of the original DataFrame
         df_director = df_copy[['director', 'rating', 'votes']].copy()

         # Split 'director' column into individual directors
         df_director['director'] = df_director['director'].str.split('\n')

         # Explode the DataFrame to create separate rows for each director
         df_director = df_director.explode('director')
         df_director['director'] = df_director['director'].str.replace(',', '')

         # Count occurrences of each director
         director_occurrences = df_director['director'].value_counts()

         # Add 'occurrences' column to the DataFrame
         df_director['occurrences'] = df_director['director'].map(director_occurrences)

         # Group by director and calculate total ratings and total votes
         director_totals = df_director.groupby('director').agg({'rating': 'sum', 'votes': 'sum
         director_totals.columns = ['director', 'total_ratings', 'total_votes']
         df_director = df_director[df_director['occurrences'] >= 20 ]
```

```
# Merge the totals back to the original DataFrame
df_director = pd.merge(df_director, director_totals, on='director', how='left')

# Calculate average ratings and average votes
df_director['average_ratings'] = df_director['total_ratings'] / df_director['occurren
df_director['average_votes'] = df_director['total_votes'] / df_director['occurrences'

# Display the resulting DataFrame
df_director
```

Out[22]:

	director	rating	votes	occurrences	total_ratings	\
0	James Cameron	7.8	295119.0	37.0	182.3	
1	Jean-François Richet	6.5	26220.0	22.0	147.7	
2	Louis Leterrier	NaN	NaN	24.0	128.7	
3	Russell Mulcahy	5.6	9026.0	48.0	263.9	
4	Matt Reeves	7.8	672146.0	24.0	151.8	
...	...	...	...	...	...	
78731	Christy Cabanne	5.5	33.0	121.0	578.1	
78732	Irving Pichel	4.5	21.0	65.0	407.3	
78733	Miguel Iglesias	6.2	46.0	24.0	112.2	
78734	Thomas Bentley	5.0	79.0	21.0	54.6	
78735	George Blair	6.3	40.0	96.0	538.9	

	total_votes	average_ratings	average_votes
0	15786815.0	4.927027	426670.675676
1	779414.0	6.713636	35427.909091
2	5619511.0	5.362500	234146.291667
3	1623486.0	5.497917	33822.625000
4	7377902.0	6.325000	307412.583333
...	...	...	...
78731	28498.0	4.777686	235.520661
78732	60346.0	6.266154	928.400000
78733	2889.0	4.675000	120.375000
78734	1856.0	2.600000	88.380952
78735	7647.0	5.613542	79.656250

[78736 rows x 8 columns]

4.13.1 Best director by average rating

In [23]:
```
#geting the mean of average ratings and votes of each director if they have multiple
director_avg_stats = df_director.groupby('director').agg({
    'average_ratings': 'mean',
    'average_votes': 'mean'
}).reset_index()

# Sort the DataFrame by average_ratings in descending order
highest_rated_directors = director_avg_stats.sort_values(by='average_ratings', ascendi
```

```
# Display the resulting DataFrame
highest_rated_directors=highest_rated_directors.reset_index(drop =True)
highest_rated_directors
```

Out[23]:

	director	average_ratings	average_votes
0	Haruo Sotozaki	8.360000	16545.600000
1	Quentin Tarantino	8.071429	757350.523810
2	Satyajit Ray	8.059091	2381.227273
3	Upendra	7.935000	2144.450000
4	Akira Kurosawa	7.730769	49984.435897
...	...	...	...
2068	Dmitriy Lazarev	0.000000	0.000000
2069	Ding M. De Jesus	0.000000	0.000000
2070	Rock Parsons	0.000000	0.000000
2071	Consuelo P. Osorio	0.000000	0.000000
2072	Drew Kochera	0.000000	0.000000

[2073 rows x 3 columns]

4.13.2 Best director by votes (popularity)

In [24]:
```
#getting the mean of average ratings and votes of each director if they have multiple
director_avg_stats = df_director.groupby('director').agg({
    'average_ratings': 'mean',
    'average_votes': 'mean'
}).reset_index()

# Sort the DataFrame by average_ratings in descending order
highest_voted_directors = director_avg_stats.sort_values(by='average_votes', ascending
highest_voted_directors=highest_voted_directors.reset_index(drop =True)
# Display the resulting DataFrame
highest_voted_directors
```

Out[24]:

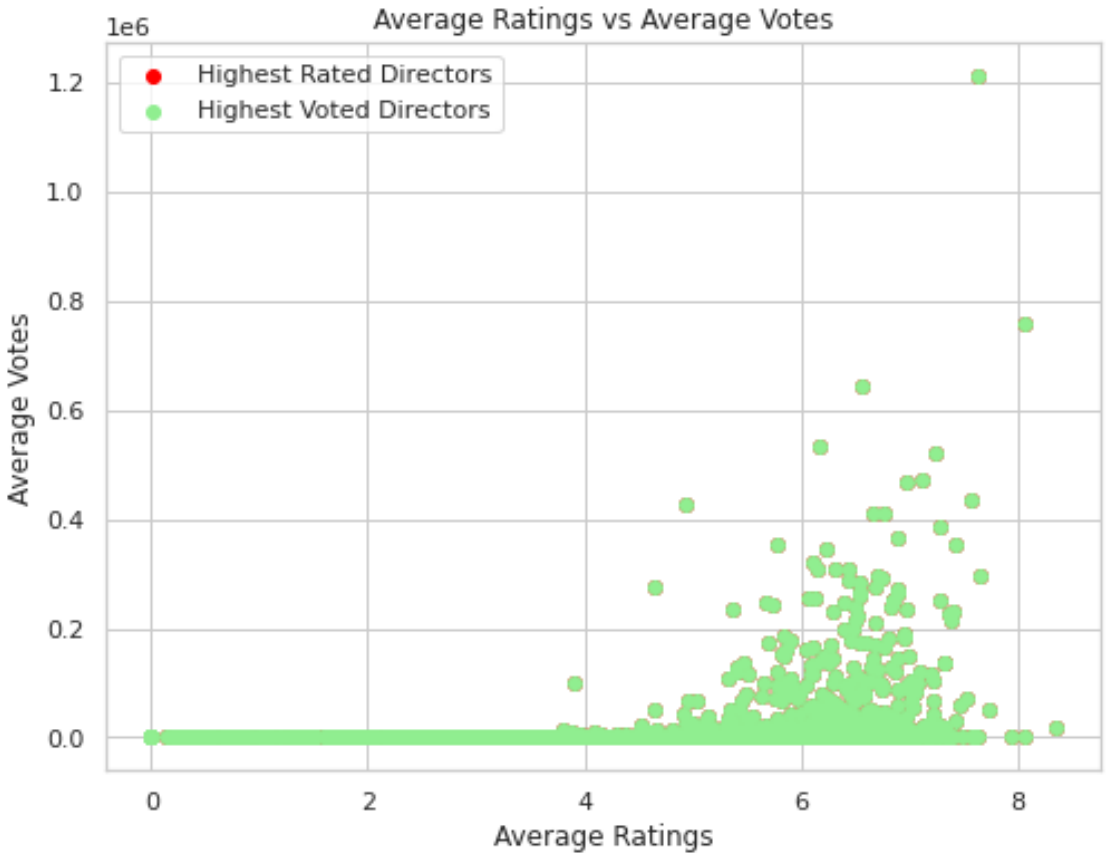
	director	average_ratings	average_votes
0	Christopher Nolan	7.629032	1.211750e+06
1	Quentin Tarantino	8.071429	7.573505e+05
2	Peter Jackson	6.563636	6.447772e+05
3	David Fincher	6.181481	5.351790e+05
4	J.J. Abrams	7.245000	5.220088e+05
...	...	...	...
2068	Hun Choi	0.000000	0.000000e+00
2069	Rock Parsons	0.000000	0.000000e+00
2070	Brando Improta	0.000000	0.000000e+00
2071	Roman Tramvay	0.000000	0.000000e+00
2072	Jim Ardent	0.000000	0.000000e+00

[2073 rows x 3 columns]

4.14 What is the relationship between highly voted and rated directors?

Do higher rated directors recieve higher votes? Is there some sort of bias when rating the movie of a highly rated director?

```
In [25]: import matplotlib.pyplot as plt

         # Plot scatter plot of average ratings vs average votes
         plt.figure(figsize=(8, 6))
         plt.scatter(highest_rated_directors['average_ratings'], highest_rated_directors['avera
         plt.scatter(highest_voted_directors['average_ratings'], highest_voted_directors['avera
         plt.xlabel('Average Ratings')
         plt.ylabel('Average Votes')
         plt.title('Average Ratings vs Average Votes')
         plt.legend()
         plt.grid(True)

         # Show the plot
         plt.show()
```



4.15 Relationship between run time and gross usd earned

I am curious to explore if a shorter movie means more revenue since people are more likely to watch something that takes less time.

```
In [26]: # cleaning
         df['gross(in $)'] = pd.to_numeric(df['gross(in $)'], errors='coerce')  # Convert gros
         df.dropna(subset=['runtime_minutes', 'gross(in $)'], inplace=True)  # Drop rows where

         # Visualization
         plt.figure(figsize=(10, 6))
         sns.regplot(x='runtime_minutes', y='gross(in $)', data=df, scatter_kws={'alpha':0.5,
         plt.title('Relationship between Runtime and Gross Earnings')
         plt.xlabel('Runtime in Minutes')
         plt.ylabel('Gross Earnings ($)')
         plt.grid(True)
         plt.show()
```