example

MICROSOFT MOVIE ANALYSIS

Final Project Submission

Please fill out:

- Student name:
- Student pace: self paced / part time / full time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

Overview

This notebook analyzes three movie databases containing datapoints across thousands of movies. The following characteristics are typically good indicators of success for movies: runtime minutes, genre, average rating and gross earnings. This analysis will help guide movie production decisions for the new Microsoft movie studio.

We found adventure movies to be the genre with the highest average audience rating and gross earnings. Our analysis also suggested that movies with a runtime duration of 90-130 minutes had a higher audience average rating.

1. Business understanding

a) Business problem

- Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don't know anything about creating movies. You are charged with exploring what types of films are currently doing the best at the box office.

Problem Statement:

- Translate the findings into actionable insights which the head of Microsoft's new movie studio can use to help decide what type of films to create.

b) Main Objective

- Identify the characteristics of the films that are currently doing better use the data to help create a new movie studio.

c) Other Objectives

1. Determine the relationship between genres and the average rating.
2. Determine the relationship between genres and the gross earnings.

3. Determine the relationship between genres and number of votes.
4. Determine the relationship between the runtime duration of the movie and average rating
5. Determine the distribution of domestic and foreign gross
6. Identify the highly rated genres

Data understanding

- Data for this analysis is taken from 2 of the largest online movie databases with datapoints on hundreds of thousands of movies.

- The data for the analysis questions primarily come from two sources: IMDb and Box Office Mojo.

- Data from IMDb can be used to analyze genre preferences and audience demographics, while Box Office Mojo data enables the assessment of box office success metrics like gross earnings. By integrating insights from both sources, Microsoft can gain a comprehensive understanding of the movie landscape and make informed decisions regarding genre selection, runtime optimization and audience targeting.

- The dataset contains the following information regarding movies:

1. tconst: A unique identifier for each movie.

2. primary_title: This is the common and popular movie title.
3. original_title: Origianl title of the movie, may be in a different language.
4. start_year: The year when production of the movie began.

5. runtime_minutes: The duration of the movie in minutes

6. genres: This is the the classification of the movie
7. averagerating: The mean rating given to the movie by viewers.
8. numvotes: The number of votes contributed by viewers, used to calculate the average rating.

9. title: The title under which the movie was produced.
10. studio: The production studio responsible for creating the movie.

11. domestic_gross: The revenue generated from the sale of movie tickets within the country of origin.

12. foreign_gross: The revenue generated from the sale of movie tickets within the country of origin.

13. year: This is the year the movie was produced * the movie.se?

Loading data

```
# Importing standard packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Loading datasets

Box office mojo

```
# Loading the csv file

f1 = r"C:\Users\ADMIN\Desktop\dsc-phase-1-project\zippedData\
bom.movie_gross.csv.gz"
df1 = pd.read_csv(f1)

# Previewing a sample

df1.head()
```

```
                                             title studio  domestic_gross
\
0                                  Toy Story 3     BV     415000000.0

1                    Alice in Wonderland (2010)     BV     334200000.0

2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0

3                                    Inception     WB     292600000.0

4                           Shrek Forever After   P/DW     238700000.0


  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010
```

```
# Shape of our data set

df1.shape
```

```
(3387, 5)
```

IMDB (ratings)

```
#Loading the csv file

f2 = r"C:\Users\ADMIN\Desktop\dsc-phase-1-project\zippedData\
imdb.title.ratings.csv.gz"
df2 = pd.read_csv(f2)

# Previewing a sample

df2.head()
```

```
       tconst  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
```

```
# Shape of our data set

df2.shape
```

```
(73856, 3)
```

IMDB (basics)

```
# Loading the csv file

f3 = r"C:\Users\ADMIN\Desktop\dsc-phase-1-project\zippedData\
imdb.title.basics.csv.gz"
df3 = pd.read_csv(f3)

# Previewing a sample

df3.head()
```

```
      tconst                    primary_title
original_title  \
0  tt0063540                        Sunghursh
Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek
Din
2  tt0069049       The Other Side of the Wind  The Other Side of the
Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada
Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela
Errante

   start_year  runtime_minutes                genres
```

```
0         2013             175.0    Action,Crime,Drama
1         2019             114.0       Biography,Drama
2         2018             122.0                 Drama
3         2018               NaN          Comedy,Drama
4         2017              80.0  Comedy,Drama,Fantasy
```

```
# Shape of our data_set
```

```
df3.shape
```

```
(146144, 6)
```

Merging the three dataframes

```
# Merging df2 and df3 based on the common column 'tconst'
```

```
merged_df = pd.merge(df2, df3, on='tconst' , how = 'inner')
```

```
merged_df.head()
```

```
       tconst  averagerating  numvotes           primary_title  \
0  tt10356526            8.3        31        Laiye Je Yaarian
1  tt10384606            8.9       559              Borderless
2   tt1042974            6.4        20               Just Inès
3   tt1043726            4.2     50352  The Legend of Hercules
4   tt1060240            6.5        21                Até Onde?

           original_title  start_year  runtime_minutes  \
0        Laiye Je Yaarian        2019            117.0
1              Borderless        2019             87.0
2               Just Inès        2010             90.0
3  The Legend of Hercules        2014             99.0
4               Até Onde?        2011             73.0

                     genres
0                   Romance
1               Documentary
2                     Drama
3  Action,Adventure,Fantasy
4           Mystery,Thriller
```

To merge the merged dataset with df1, I will create a common column between the two dataframes.

```
# Renaming the column 'title' in df1 to 'primary title'
```

```
df1.rename(columns={'title': 'primary_title'}, inplace=True)
```

```
# Converting the column 'foreign_gross' to a numerical d.type
```

```
column_name = 'foreign_gross'
df1[column_name] = pd.to_numeric(df1[column_name], errors='coerce')

# Merging the merged dataset to df1 (bom_movie_gross)

merged_df1 = pd.merge(merged_df, df1, on='primary_title' , how =
'inner')
```

I opted for an inner join which merges the datasets based on matched values, ensuring a more cohesive and reliable dataset. Using an outer join results in numerous missing values, diminishing the credibility of our dataset.

```
# Reassigning the variable 'merged_df1' to df

df = merged_df1
```

Data inspection

Let's inspect our merged dataframe.

```
# Previewing the first five rows

df.head()
```

```
      tconst  averagerating  numvotes                     primary_title
\
0  tt1043726            4.2     50352            The Legend of Hercules

1  tt1171222            5.1      8296                      Baggage Claim

2  tt1181840            7.0      5494  Jack and the Cuckoo-Clock Heart

3  tt1210166            7.6    326657                           Moneyball

4  tt1212419            6.5     87288                           Hereafter


                  original_title  start_year  runtime_minutes  \
0         The Legend of Hercules        2014             99.0
1                Baggage Claim        2013             96.0
2  Jack et la mécanique du coeur        2013             94.0
3                      Moneyball        2011            133.0
4                      Hereafter        2010            129.0

                      genres  studio  domestic_gross  foreign_gross
year
0   Action,Adventure,Fantasy    LG/S      18800000.0     42400000.0
2014
1                     Comedy    FoxS      21600000.0       887000.0
2013
```

```
2  Adventure,Animation,Drama  Shout!              NaN      3400000.0
2014
3      Biography,Drama,Sport    Sony      75600000.0     34600000.0
2011
4      Drama,Fantasy,Romance      WB      32700000.0     72500000.0
2010
```

```
# Previewing the last five rows

df.tail()
```

```
         tconst  averagerating  numvotes       primary_title  \
3022  tt3399916            6.3      4185      The Dead Lands
3023  tt3616916            6.7     28167           The Wave
3024  tt3748512            7.4      4977  Hitchcock/Truffaut
3025  tt7008872            7.0     18768          Boy Erased
3026  tt7048622            7.7     11168          The Insult

          original_title  start_year  runtime_minutes
genres  \
3022      The Dead Lands        2014            107.0
Action,Adventure
3023              Bølgen        2015            105.0
Action,Drama,Thriller
3024  Hitchcock/Truffaut        2015             79.0
Documentary
3025          Boy Erased        2018            115.0
Biography,Drama
3026           L'insulte        2017            113.0
Crime,Drama,Thriller

     studio  domestic_gross  foreign_gross  year
3022  Magn.          5200.0            NaN  2015
3023  Magn.        177000.0            NaN  2016
3024  Cohen        260000.0            NaN  2015
3025  Focus       6800000.0      5000000.0  2018
3026  Cohen       1000000.0            NaN  2018
```

```
# Checking the shape of our dataframe
df.shape
```

```
(3027, 12)
```

```
# Checking the info and uniformity of our dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3027 entries, 0 to 3026
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   tconst            3027 non-null   object
 1   averagerating     3027 non-null   float64
 2   numvotes          3027 non-null   int64
 3   primary_title     3027 non-null   object
 4   original_title    3027 non-null   object
 5   start_year        3027 non-null   int64
 6   runtime_minutes   2980 non-null   float64
 7   genres            3020 non-null   object
 8   studio            3024 non-null   object
 9   domestic_gross    3005 non-null   float64
 10  foreign_gross     1828 non-null   float64
 11  year              3027 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 283.9+ KB
```

```
# Checking data numerical summaries
df.describe()
```

```
       averagerating      numvotes   start_year  runtime_minutes  \
count    3027.000000  3.027000e+03  3027.000000      2980.000000
mean        6.457582  6.170030e+04  2013.783284       107.217114
std         1.012277  1.255132e+05     2.466955        20.073886
min         1.600000  5.000000e+00  2010.000000         3.000000
25%         5.900000  2.117000e+03  2012.000000        94.000000
50%         6.600000  1.310900e+04  2014.000000       105.000000
75%         7.100000  6.276550e+04  2016.000000       118.000000
max         9.200000  1.841066e+06  2019.000000       272.000000

       domestic_gross  foreign_gross         year
count    3.005000e+03   1.828000e+03  3027.000000
mean     3.064033e+07   7.843093e+07  2014.077635
std      6.671629e+07   1.387062e+08     2.442245
min      1.000000e+02   6.000000e+02  2010.000000
25%      1.390000e+05   4.700000e+06  2012.000000
50%      2.000000e+06   2.125000e+07  2014.000000
75%      3.250000e+07   8.170000e+07  2016.000000
max      7.001000e+08   9.464000e+08  2018.000000
```

Data cleaning

```
# Making a copy of the merged data set to retain an original copy.

df_copy = df.copy()
```

a) Checking for completeness of our data.

```
# Number of missing values per column
df.isna().sum()
```

```
tconst                0
averagerating         0
numvotes              0
primary_title         0
original_title        0
start_year            0
runtime_minutes      47
genres                7
studio                3
domestic_gross       22
foreign_gross      1199
year                  0
dtype: int64
```

```
# Checking the proportion of our missing data
df.isnull().mean()
```

```
tconst             0.000000
averagerating      0.000000
numvotes           0.000000
primary_title      0.000000
original_title     0.000000
start_year         0.000000
runtime_minutes    0.015527
genres             0.002313
studio             0.000991
domestic_gross     0.007268
foreign_gross      0.396102
year               0.000000
dtype: float64
```

- The proportion of missing data for the columns: runtime_minutes, genres, studio and domestic_gross are insignificant compared to the complete data. For the foreign_gross column, the proportion of missing values accounts for almost 40% of our data.
- For the 'runtime_minutes' and 'domestic_gross' columns, I will impute missing values with the mean since it provides a representative measure for our data.
- For the 'genres' and 'studio' columns, I will impute the missing values with 'Not recorded' as they are deemed insignificant compared to the available data. This approach allows us to maintain the completeness of the dataset without introducing bias.
- For the 'foreign_gross' column, I will impute missing values with 0 to retain information about the missingness of the data while still including observations with available data."

```
# Repalcing the missing values in 'runtime_minutes' and
'domestic_gross' with the mean

df[['runtime_minutes' , 'domestic_gross']]= df[['runtime_minutes' ,
'domestic_gross']].fillna(df[['runtime_minutes' ,
'domestic_gross']].mean())
```

```
# Filling missing values in 'genres' column with 'Not recorded'

df['genres'].fillna('Not recorded', inplace=True)

# Filling missing values in 'foreign_gross' with 0

df['foreign_gross'].fillna(0, inplace=True)

df.head()
```

```
      tconst  averagerating  numvotes                     primary_title
\
0  tt1043726            4.2     50352           The Legend of Hercules

1  tt1171222            5.1      8296                     Baggage Claim

2  tt1181840            7.0      5494  Jack and the Cuckoo-Clock Heart

3  tt1210166            7.6    326657                        Moneyball

4  tt1212419            6.5     87288                        Hereafter


                  original_title  start_year  runtime_minutes  \
0         The Legend of Hercules        2014             99.0
1                    Baggage Claim        2013             96.0
2  Jack et la mécanique du coeur        2013             94.0
3                      Moneyball        2011            133.0
4                      Hereafter        2010            129.0

                       genres  studio  domestic_gross  foreign_gross
year
0   Action,Adventure,Fantasy    LG/S    1.880000e+07     42400000.0
2014
1                     Comedy    FoxS    2.160000e+07       887000.0
2013
2  Adventure,Animation,Drama  Shout!    3.064033e+07      3400000.0
2014
3      Biography,Drama,Sport    Sony    7.560000e+07     34600000.0
2011
4      Drama,Fantasy,Romance      WB    3.270000e+07     72500000.0
2010
```

```
# Checking for duplicates

num_of_duplicates = df.duplicated().sum()
print ("The number of duplicates in our dataframe is:" ,
num_of_duplicates)
```

```
The number of duplicates in our dataframe is: 0
```

b) Validity

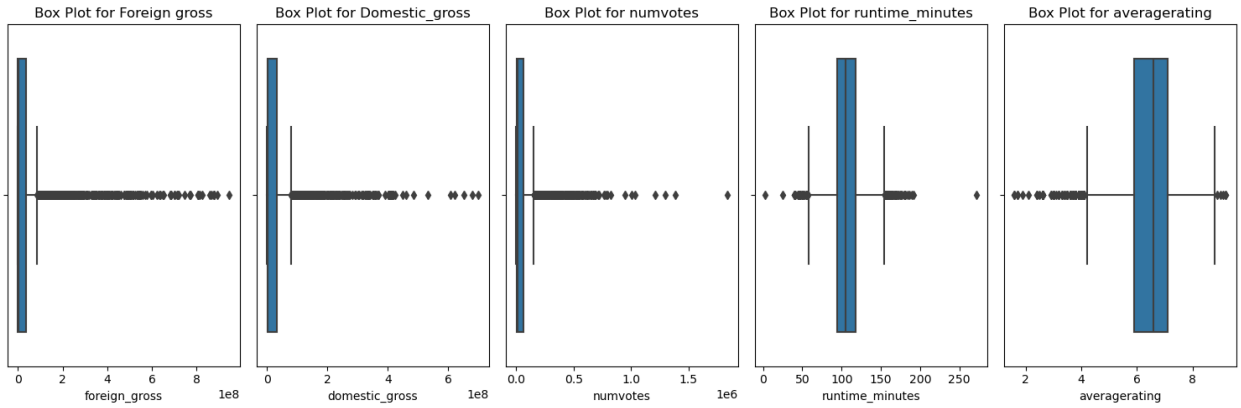
- Some columns are not important, hence can be dropped, so as to remain with only the necessary columns. I will drop the columns 'start_year' and 'original_title' to prevent redundancy.

```
# Dropping the start_year , original_title columns

df.drop(columns = ["start_year", "original_title"], inplace = True)

# Making 'tconst' the index of the dataframe.
df.set_index("tconst", inplace = True)

# Previewing our dataset
df.head()
```

```
           averagerating  numvotes                     primary_title  \
tconst
tt1043726            4.2     50352           The Legend of Hercules
tt1171222            5.1      8296                     Baggage Claim
tt1181840            7.0      5494  Jack and the Cuckoo-Clock Heart
tt1210166            7.6    326657                        Moneyball
tt1212419            6.5     87288                        Hereafter

           runtime_minutes                       genres  studio
domestic_gross  \
tconst

tt1043726             99.0   Action,Adventure,Fantasy    LG/S
1.880000e+07
tt1171222             96.0                     Comedy    FoxS
2.160000e+07
tt1181840             94.0  Adventure,Animation,Drama  Shout!
3.064033e+07
tt1210166            133.0      Biography,Drama,Sport    Sony
7.560000e+07
tt1212419            129.0      Drama,Fantasy,Romance      WB
3.270000e+07

           foreign_gross  year
tconst
tt1043726     42400000.0  2014
tt1171222       887000.0  2013
tt1181840      3400000.0  2014
tt1210166     34600000.0  2011
tt1212419     72500000.0  2010
```

```
# Detection of outliers using the IQR method

numeric_columns = ['foreign_gross', 'domestic_gross',
'averagerating' , 'runtime_minutes' , 'numvotes']
```

```
# Loop through each numeric column
for column in numeric_columns:
    # Calculate IQR
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1

    # Identify outliers using IQR
    outliers = (df[column] < (q1 - 1.5 * iqr)) | (df[column] > (q3 +
1.5 * iqr))

# 9. outlier plot
# Set up the figure with subplots
fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(15, 5))

# Box plot for 'foreign_gross'
sns.boxplot(x=df['foreign_gross'], ax=axes[0])
axes[0].set_title('Box Plot for Foreign gross')

# Box plot for 'domestic_gross'
sns.boxplot(x=df['domestic_gross'], ax=axes[1])
axes[1].set_title('Box Plot for Domestic_gross')

# Box plot for 'numvotes'
sns.boxplot(x=df['numvotes'], ax=axes[2])
axes[2].set_title('Box Plot for numvotes')

# Box plot for 'runtime_minutes'
sns.boxplot(x=df['runtime_minutes'], ax=axes[3])
axes[3].set_title('Box Plot for runtime_minutes')

# Box plot for 'averagerating'
sns.boxplot(x=df['averagerating'], ax=axes[4])
axes[4].set_title('Box Plot for averagerating')

# Adjust layout
plt.tight_layout()
plt.show()
```

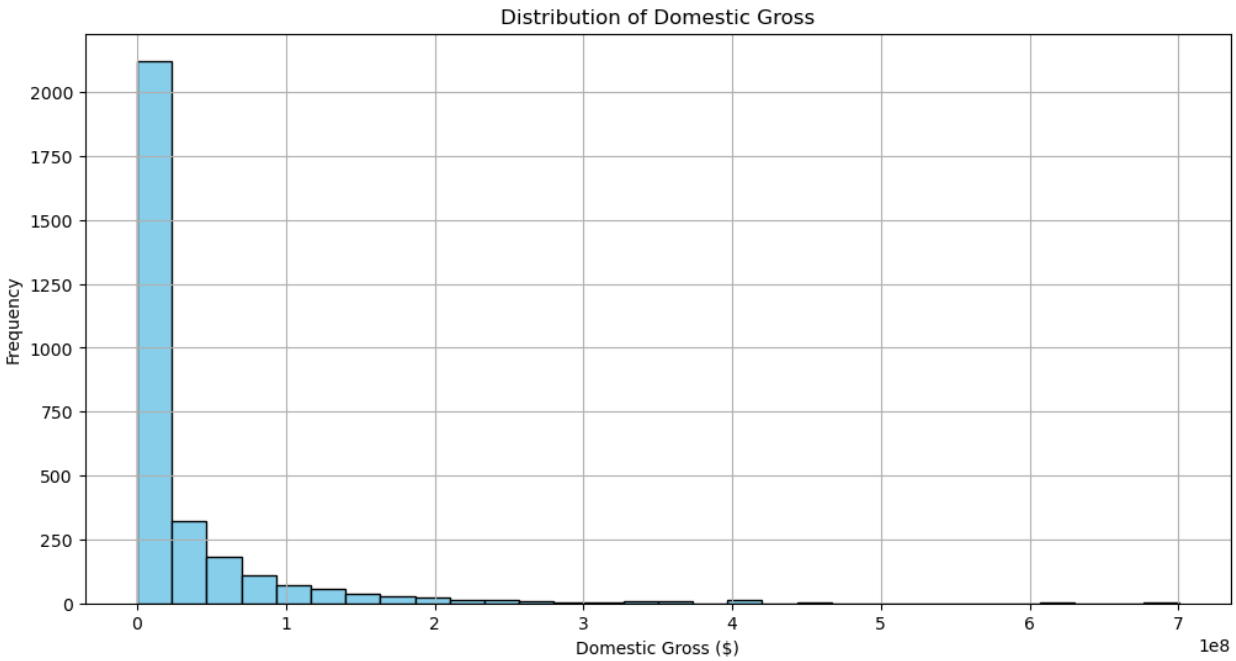


- I will not address any outliers in the dataset, as they are representative of the study.

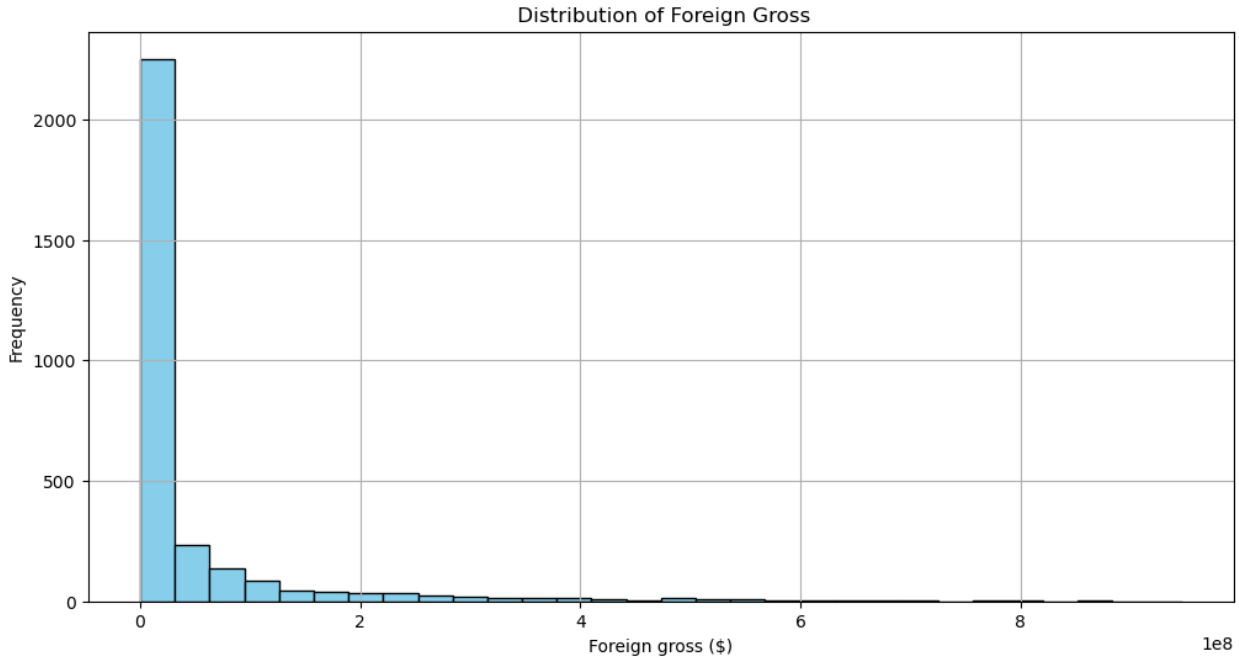
Exploratory data analysis

- Distribution of the gross earnings (domestic and foreign)

```
# Plotting a histogram for domestic gross

plt.figure(figsize=(12, 6))
plt.hist(df['domestic_gross'], bins=30, color='skyblue',
edgecolor='black')
plt.xlabel('Domestic Gross ($)')
plt.ylabel('Frequency')
plt.title('Distribution of Domestic Gross')
plt.grid(True)
plt.show()
```



```
# Plotring a histogram for foreign gross

plt.figure(figsize=(12, 6))
plt.hist(df['foreign_gross'], bins=30, color='skyblue',
edgecolor='black')
plt.xlabel('Foreign gross ($)')
plt.ylabel('Frequency')
plt.title(' Distribution of Foreign Gross')
plt.grid(True)
plt.show()
```



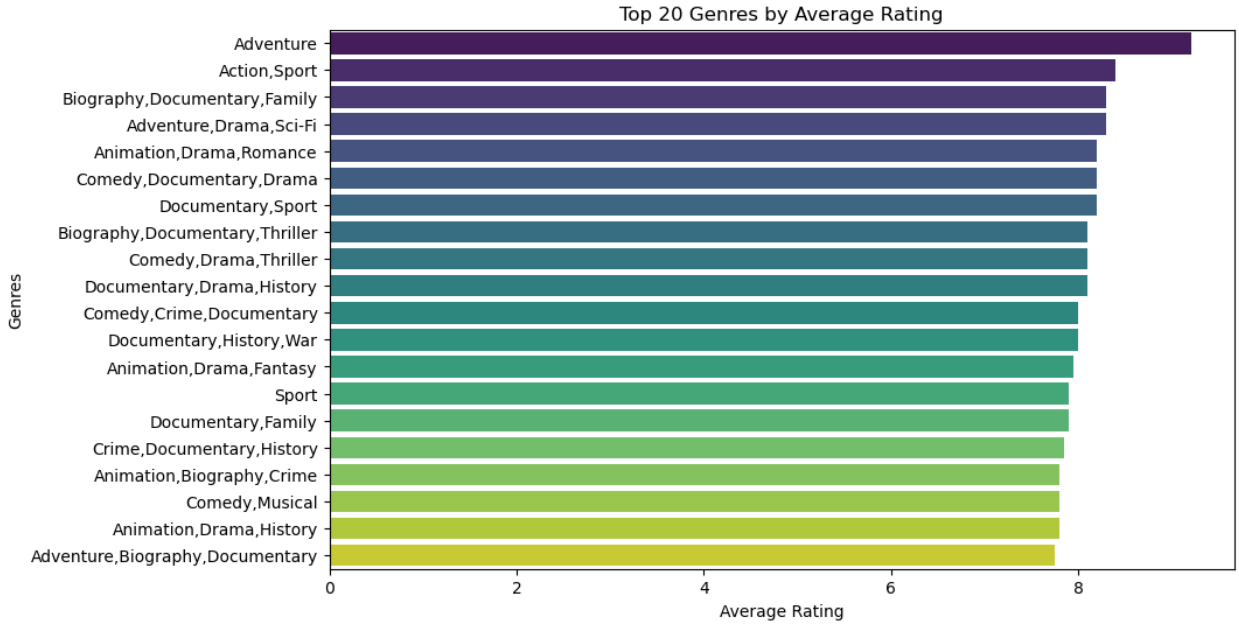
- The distribution of domestic and foreign gross in our dataset exhibits positive skewness. This indicates that the majority of movies yielded lower earnings in both domestic and foreign markets. A select few movies generated significantly higher gross earnings.
- From the histograms above, the foreign gross yielded higher earnings on average compared to the domestic gross. This suggests that, on average, movies in the dataset tended to perform better in international markets than in their domestic market.

Most successful genres based on average rating and gross earnings.

- Average rating

```
# Group by the genres and calculate the average rating for each genre.
Arranging from the highest average rating to the lowest.

genre_means = df.groupby('genres')
['averagerating'].mean().sort_values(ascending = False).head(20)

# Plotting the bar chart
plt.figure(figsize = (10, 6))
```

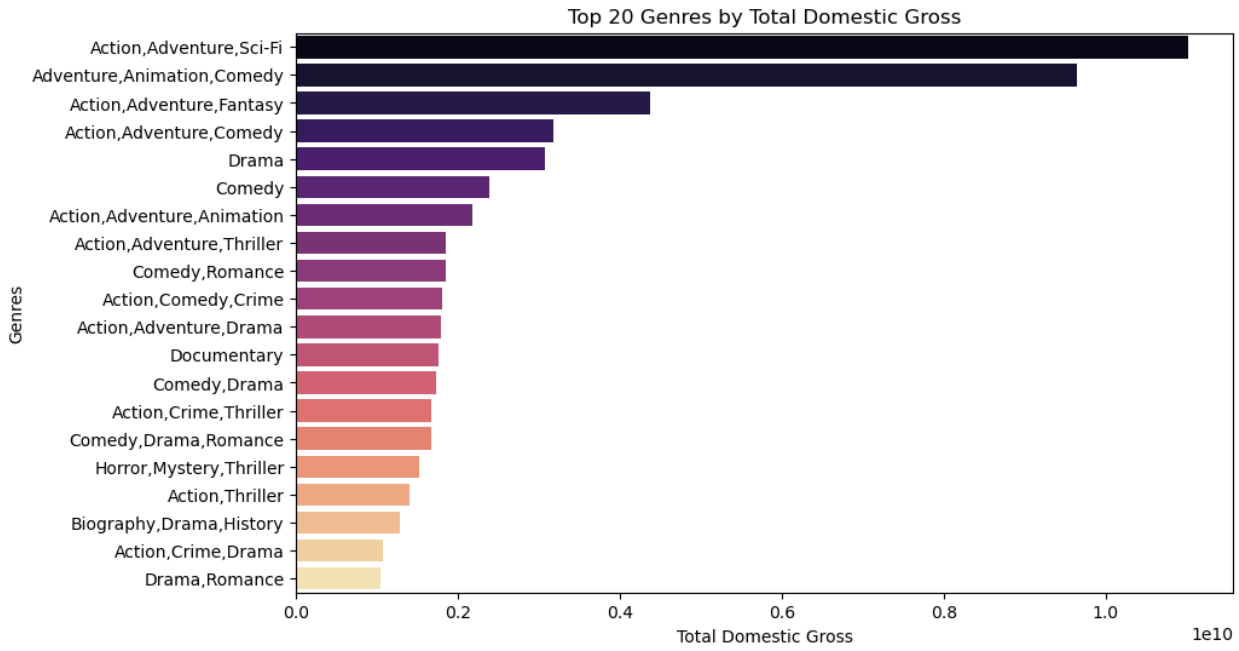
```
sns.barplot(x = genre_means.values, y = genre_means.index,
palette='viridis')
plt.xlabel('Average Rating')
plt.ylabel('Genres')
plt.title('Top 20 Genres by Average Rating')
plt.show()
```



From this movie data set, adventure movies have the highest audience average rating.

- Domestic gross

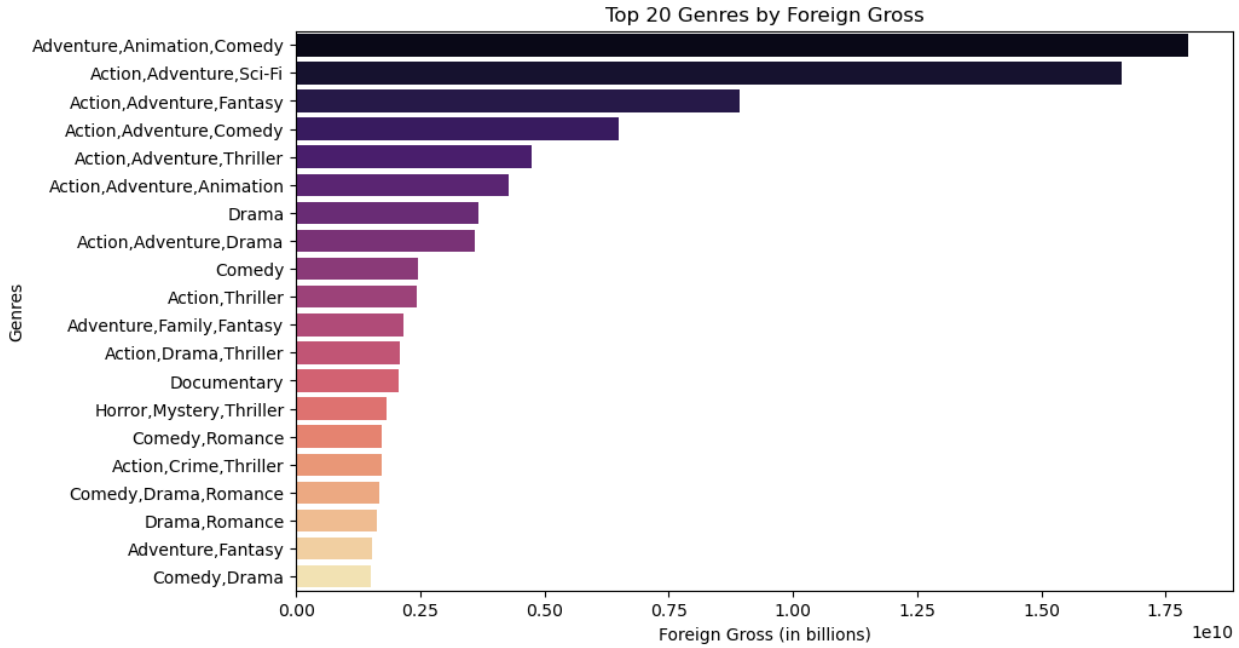
```
# Group by genre and calculate the total domestic gross for each
genre.
genre_domestic_gross = df.groupby('genres')
['domestic_gross'].sum().sort_values(ascending = False).head(20)

# Plot the bar chart
plt.figure(figsize = (10, 6))
sns.barplot(x = genre_domestic_gross.values, y =
genre_domestic_gross.index, palette='magma')
plt.xlabel('Total Domestic Gross')
plt.ylabel('Genres')
plt.title('Top 20 Genres by Total Domestic Gross')
plt.show()
```



The genres action, adventure and sci-fi gave the highest domestic gross earnings.

- Foreign gross

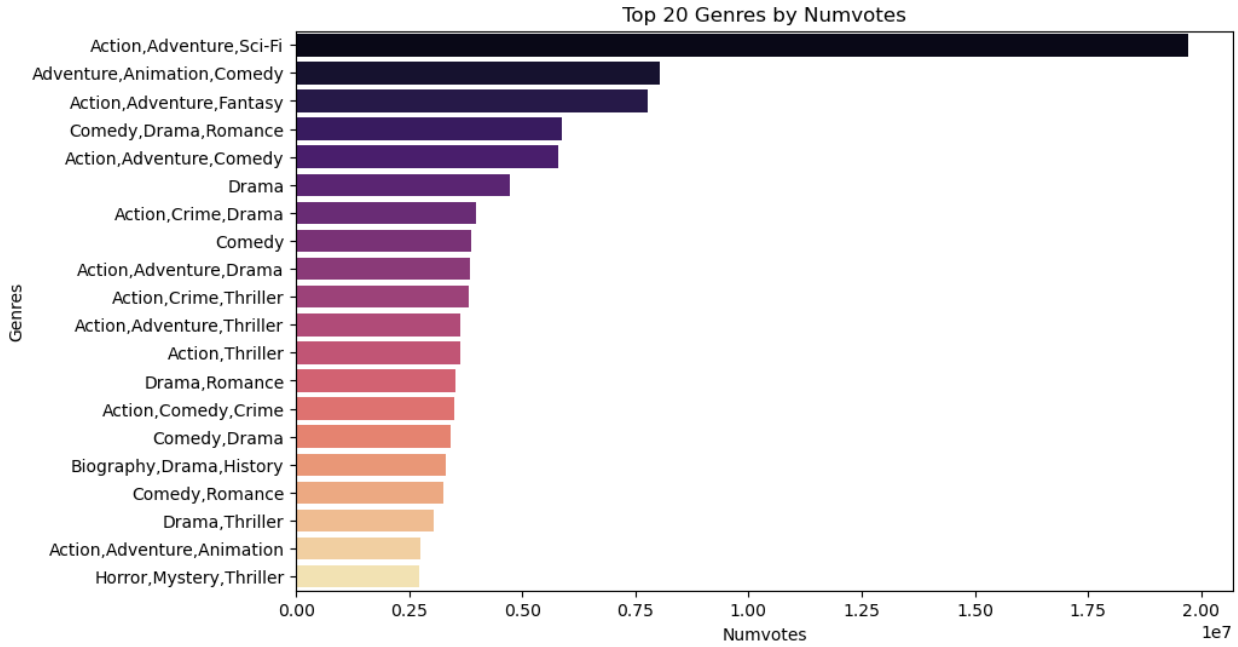
```
# Group by genre and calculate the total foreign gross for each genre
genre_foreign_gross = df.groupby('genres')
['foreign_gross'].sum().sort_values(ascending = False).head(20)

# Plotting the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x = genre_foreign_gross.values, y =
genre_foreign_gross.index, palette = 'magma')
plt.xlabel('Foreign Gross (in billions)')
plt.ylabel('Genres')
plt.title('Top 20 Genres by Foreign Gross')
plt.show()
```



The genres adventure, animation and comedy gave the highest foreign gross earnings.

- Numvotes (number of votes)

```
# Group by genre and calculate the total numvotes for each genre
genre_numvotes = df.groupby('genres')
['numvotes'].sum().sort_values(ascending = False).head(20)

# Plotting the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x = genre_numvotes.values, y = genre_numvotes.index,
palette = 'magma')
plt.xlabel('Numvotes')
plt.ylabel('Genres')
plt.title('Top 20 Genres by Numvotes')
plt.show()
```

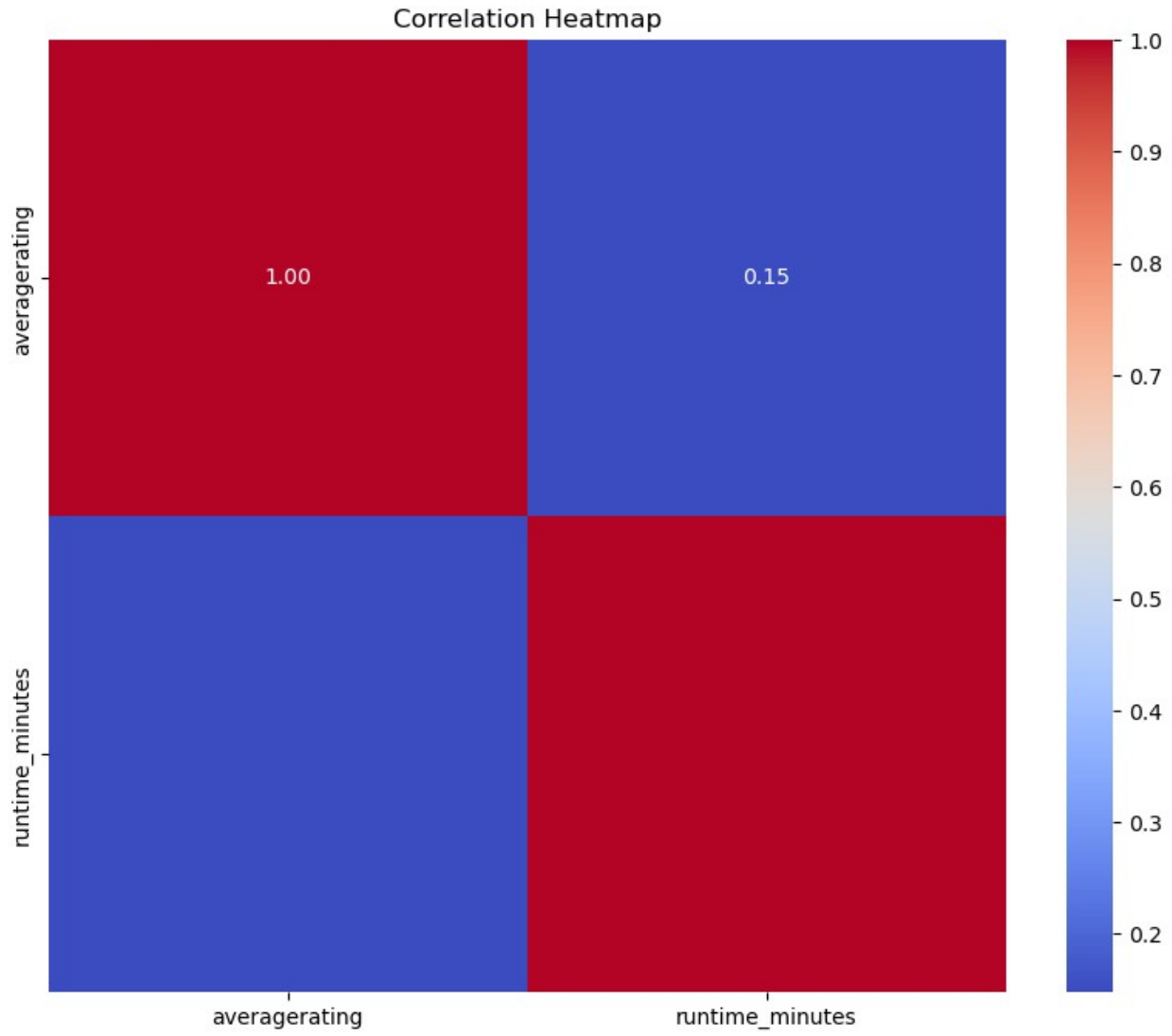


The genres Action, Adventure, and Sci-Fi garnered the highest number of votes among viewers.

Relationship between the average rating and the runtime of the movie.

```
# Assigning our two variables 'averagerating' and 'runtime_minutes'
data_set = df[['averagerating' , 'runtime_minutes']]
correlation_matrix = data_set.corr()

# Plotting the correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Heatmap')
plt.show
```

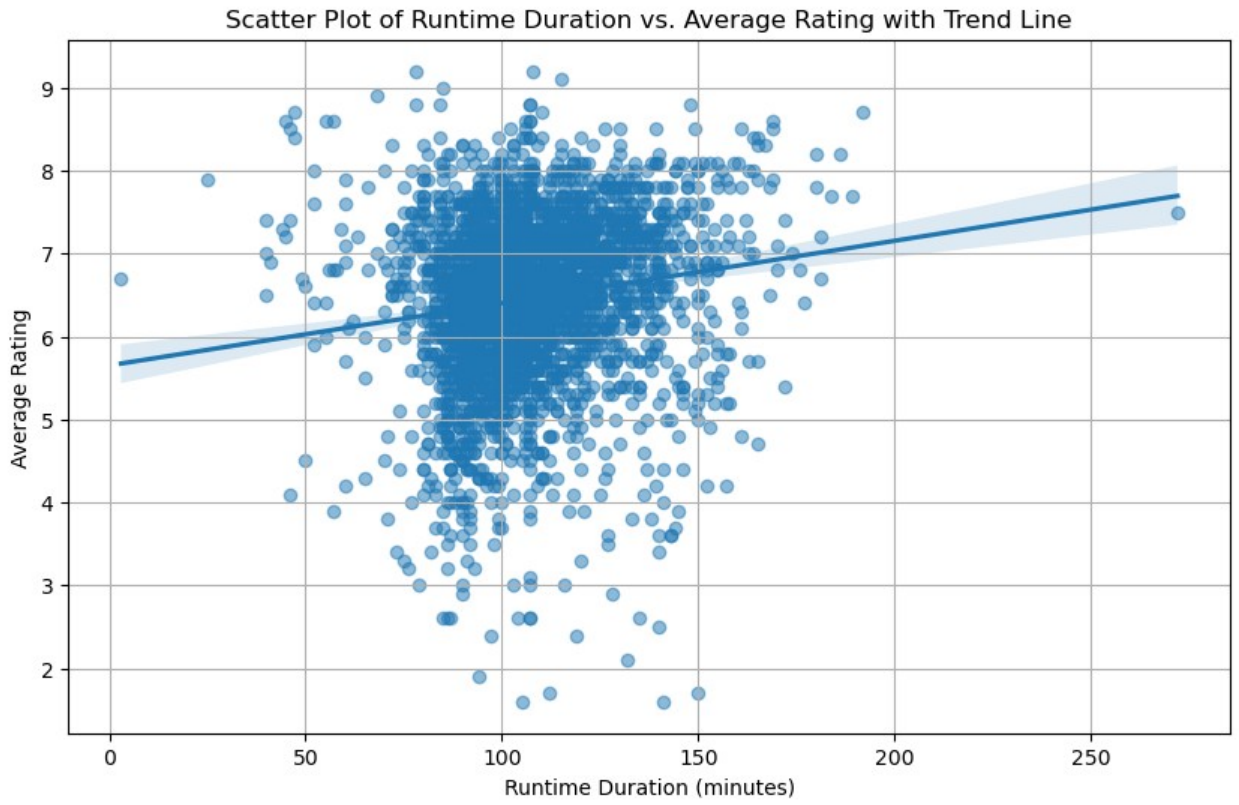
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



The above heatmap shows a weak positive correlation between the average rating and runtime minutes. It suggests that there may be some association between the duration of a movie and its rating but the relationship is not particularly strong.

The scatter plot below has been used to visualize the same observation made above.

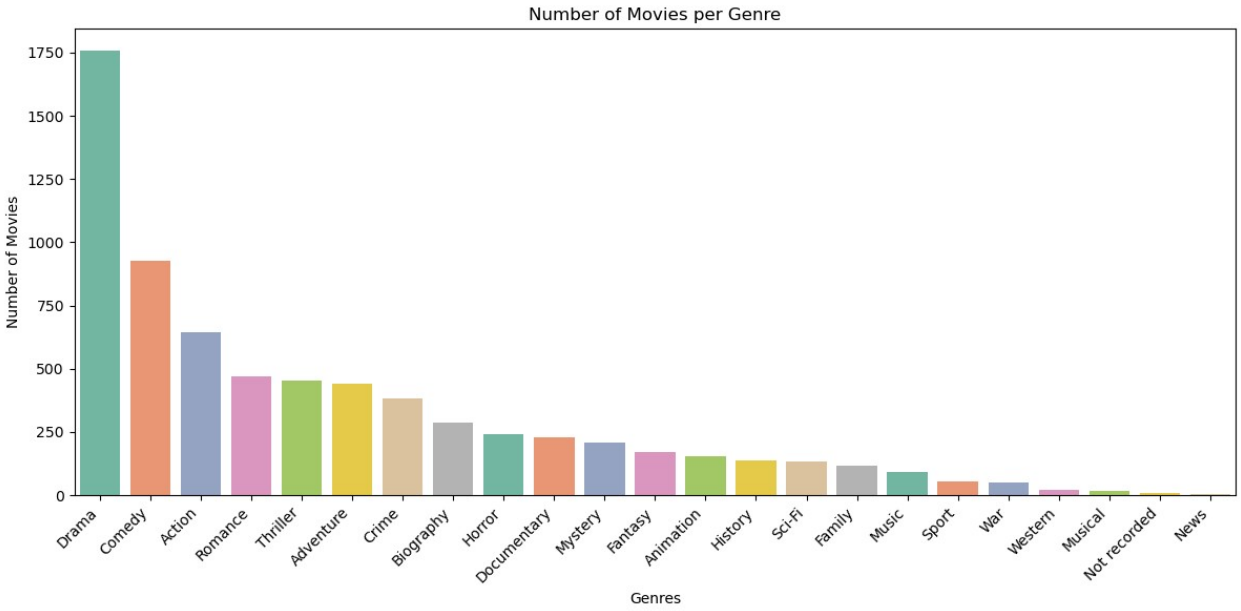
```
# Plotting runtime_minutes vs. average rating with a trend line

plt.figure(figsize=(10, 6))
sns.regplot(x = 'runtime_minutes', y = 'averagerating', data = df,
scatter_kws = {'alpha':0.5})
plt.title('Scatter Plot of Runtime Duration vs. Average Rating with
Trend Line')
plt.xlabel('Runtime Duration (minutes)')
plt.ylabel('Average Rating')
plt.grid(True)
plt.show()
```



The scatter plot above illustrates that movies with higher ratings tend to fall within the range of 90 to 140 minutes in duration.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df' is your DataFrame containing movie data with a
'genres' column
# Split genres string into a list of genres
df['genres'] = df['genres'].str.split(',')

# Explode the 'genres' column so that each genre becomes a separate
row
df_exploded = df.explode('genres')

# Count the number of movies for each genre
genre_counts = df_exploded['genres'].value_counts()

# Plot the bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x=genre_counts.index, y=genre_counts.values,
palette='Set2')
plt.xlabel('Genres')
plt.ylabel('Number of Movies')
```

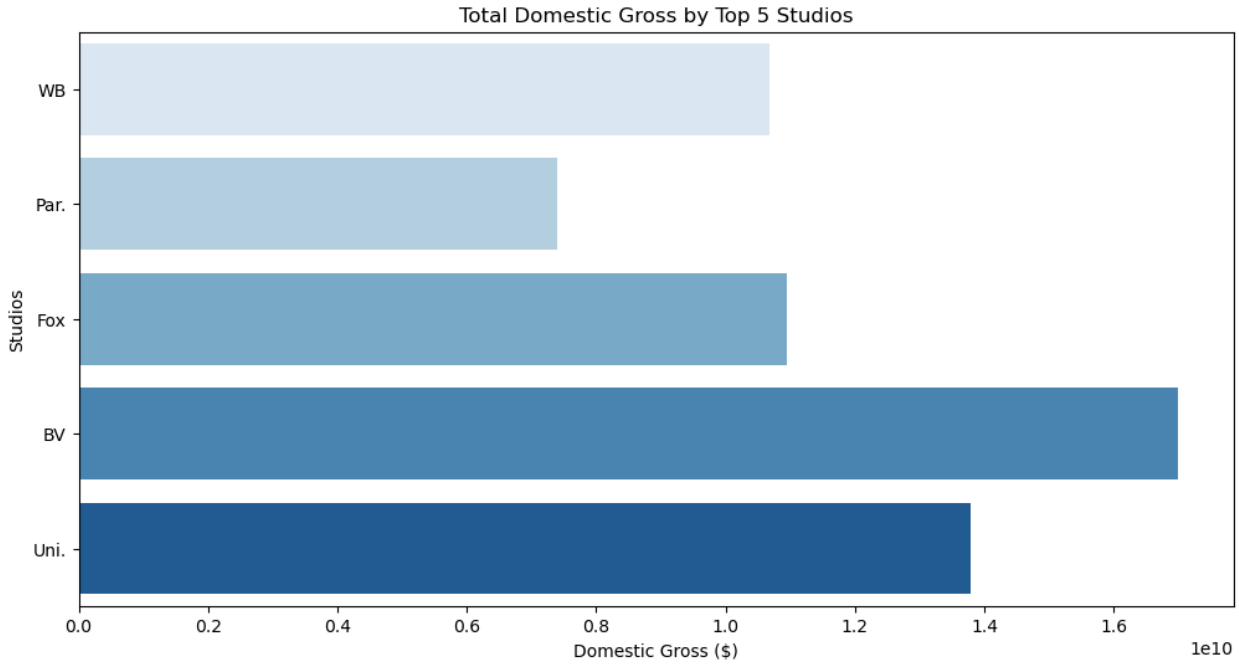
```
plt.title('Number of Movies per Genre')
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



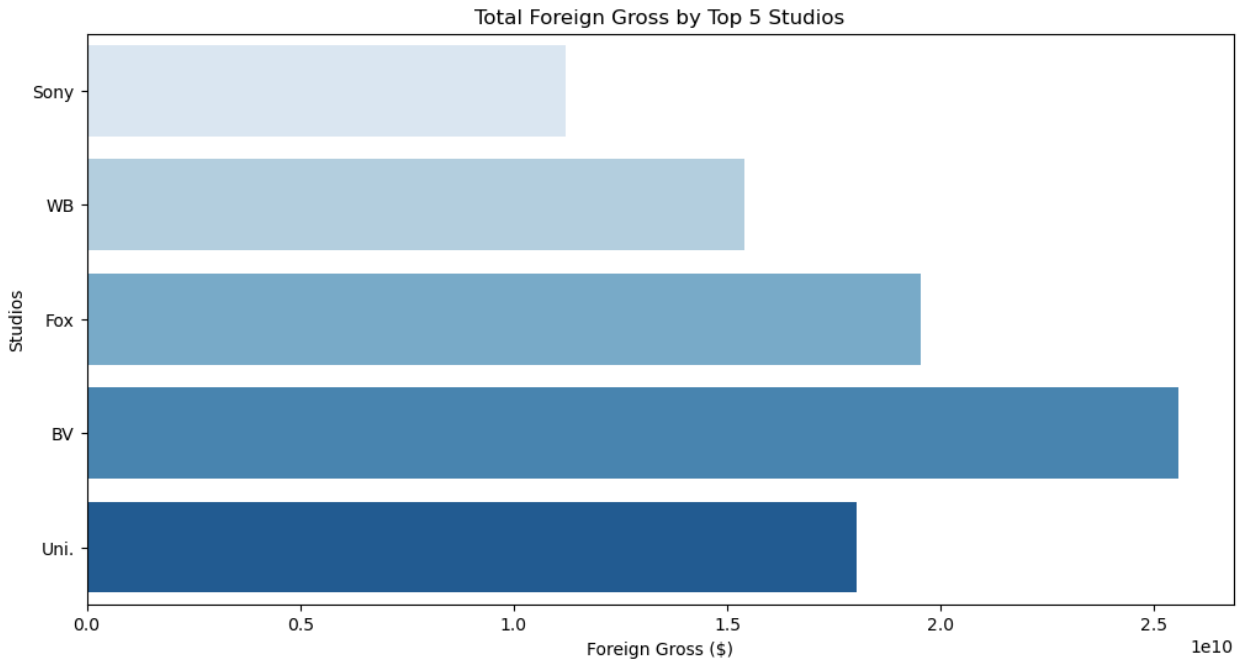
Drama movies were the most produced, followed by comedy and action.

Relationship between existing studios and gross earnings

```
# Calculate total domestic gross by studio

domestic_gross_by_studio = df.groupby('studio')
['domestic_gross'].sum().sort_values(ascending=False)
top_5_domestic_studios = domestic_gross_by_studio.head(5).index

# Filter the data for the top 5 domestic studios
domestic_data_top_5 = df[df['studio'].isin(top_5_domestic_studios)]

# Plot for top 5 domestic gross
plt.figure(figsize=(12, 6))
sns.barplot(x='domestic_gross', y='studio', data=domestic_data_top_5,
estimator=sum, errorbar=None, palette='Blues')
plt.xlabel('Domestic Gross ($)')
plt.ylabel('Studios')
plt.title('Total Domestic Gross by Top 5 Studios')
plt.show()
```




```
# Calculate total foreign gross by studio

foreign_gross_by_studio = df.groupby('studio')
['foreign_gross'].sum().sort_values(ascending=False)
top_5_foreign_studios = foreign_gross_by_studio.head(5).index

# Filter the data for the top 5 foreign studios
foreign_data_top_5 = df[df['studio'].isin(top_5_foreign_studios)]

# Plot for top 5 foreign gross
plt.figure(figsize=(12, 6))
sns.barplot(x='foreign_gross', y='studio', data=foreign_data_top_5,
estimator=sum, errorbar=None, palette='Blues')
plt.xlabel('Foreign Gross ($)')
plt.ylabel('Studios')
plt.title('Total Foreign Gross by Top 5 Studios')
plt.show()
```



Based on the two bar graphs, we observe that BV (Buena Vista), Fox, and Universal Studios achieved the highest gross earnings, with BV ranking at the top.

Conclusions

- The genres loved by majority of people locally and internationally are action and adventure movies.
- Action, Adventure and Sci-fi are the highest selling genres internationally and the highest number of votes.
- Adventure, Animation and Comedy are the highest selling genres locally.
- Foreign gross yielded higher earnings on average compared to the domestic gross.
- Movies with higher ratings tend to fall within the range of 90 to 140 minutes in duration.
- Buena Vista (BV) is the highest ranking movie studio in terms of domestic and foreign earnings.

Limitations

- The runtime length for movies below an average rating of 8.0 are not taken into consideration therefore, we cannot draw specific conclusions that a 90 minute movie will help contribute to a higher rating, rather we can conclude that most higher rated films are within this runtime.
- A relatively large proportion of missing data for foreign gross earnings.
- This analysis is incomplete as it hasn't considered crucial factors like movie production budgets, profits, and seasonal trends, which are vital for making informed decisions in movie production.

Recommendations

- Action, adventure and sci-fi movies have the highest number of people watching them locally and internationally and should thus be given priority.
- Further reasearch and analysis can be done looking into the methods adopted by the high ranking movie studios.
- Additional analysis could be done comparing production budget to viewer rating. This modeling could help predict whether higher production values trend towards higher ratings by viewers.