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

#Importing key libraries
df1=pd.read_csv(r"C:\Users\user\Desktop\Project\title.basics.csv")
df2=pd.read_csv(r"C:\Users\user\Desktop\Project\bom.movie_gross.csv")
df3=pd.read_csv(r"C:\Users\user\Desktop\Project\title.ratings.csv")
```

Data Merging

```
#Viewing the first 5 rows for the first dataset
df1.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
```

```
#Viewing the first 5 rows for the second dataset

df2.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
```

```
#Viewing the first 5 rows for the third dataset

df3.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

df4=df1.merge(df3)
df4.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  averagerating
numvotes
0        2013            175.0    Action,Crime,Drama            7.0
77
1        2019            114.0       Biography,Drama            7.2
43
2        2018            122.0                Drama            6.9
4517
3        2018              NaN          Comedy,Drama            6.1
13
4        2017             80.0  Comedy,Drama,Fantasy            6.5
119
```

```
df2.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
```

```
#Renaming df2 title column to primary_title
df6 = df2.rename(columns={'title': 'primary_title'})

df6.head()
```

```
                                  primary_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
```

```
df7=df4.merge(df6)
```

Merged Dataset

```
df7.head(15)
```

```
       tconst                    primary_title  \
0   tt0315642                            Wazir
1   tt0337692                   On the Road
2   tt4339118                   On the Road
3   tt5647250                   On the Road
4   tt0359950  The Secret Life of Walter Mitty
5   tt0365907      A Walk Among the Tombstones
6   tt0369610                  Jurassic World
7   tt0372538                             Spy
8   tt3079380                             Spy
9   tt0376136                The Rum Diary
10  tt0376479              American Pastoral
11  tt0383010              The Three Stooges
12  tt0398286                        Tangled
13  tt0401729                  John Carter
14  tt0409379                    In Secret

                     original_title  start_year  runtime_minutes  \
0                             Wazir        2016            103.0
1                      On the Road        2012            124.0
2                      On the Road        2014             89.0
3                      On the Road        2016            121.0
4   The Secret Life of Walter Mitty        2013            114.0
5       A Walk Among the Tombstones        2014            114.0
6                    Jurassic World        2015            124.0
7                               Spy        2011            110.0
8                               Spy        2015            119.0
9                       The Rum Diary        2011            119.0
10                American Pastoral        2016            108.0
11                The Three Stooges        2012             92.0
12                          Tangled        2010            100.0
13                     John Carter        2012            132.0
14                       In Secret        2013            107.0

                       genres  averagerating  numvotes   studio  \
0           Action,Crime,Drama            7.1     15378  Relbig.
1      Adventure,Drama,Romance            6.1     37886      IFC
2                        Drama            6.0         6      IFC
3                        Drama            5.7       127      IFC
4       Adventure,Comedy,Drama            7.3    275300      Fox
5           Action,Crime,Drama            6.5    105116     Uni.
6      Action,Adventure,Sci-Fi            7.0    539338     Uni.
7           Action,Crime,Drama            6.6        78      Fox
8          Action,Comedy,Crime            7.0    213908      Fox
9                Comedy,Drama            6.2     94787       FD
10                Crime,Drama            6.1     12898      LGF
```

```
11                Comedy,Family            5.1     28570      Fox
12  Adventure,Animation,Comedy             7.8    366366       BV
13     Action,Adventure,Sci-Fi             6.6    241792       BV
14        Crime,Drama,Thriller             6.2      7045    RAtt.
```

```
    domestic_gross foreign_gross  year
0        1100000.0           NaN  2016
1         744000.0       8000000  2012
2         744000.0       8000000  2012
3         744000.0       8000000  2012
4       58200000.0     129900000  2013
5       26300000.0      26900000  2014
6      652300000.0       1,019.40  2015
7      110800000.0     124800000  2015
8      110800000.0     124800000  2015
9       13100000.0      10800000  2011
10        544000.0           NaN  2016
11      44300000.0      10500000  2012
12     200800000.0     391000000  2010
13      73100000.0     211100000  2012
14        444000.0           NaN  2014
```

```
#Checking for missing values
df7.isnull().mean()
```

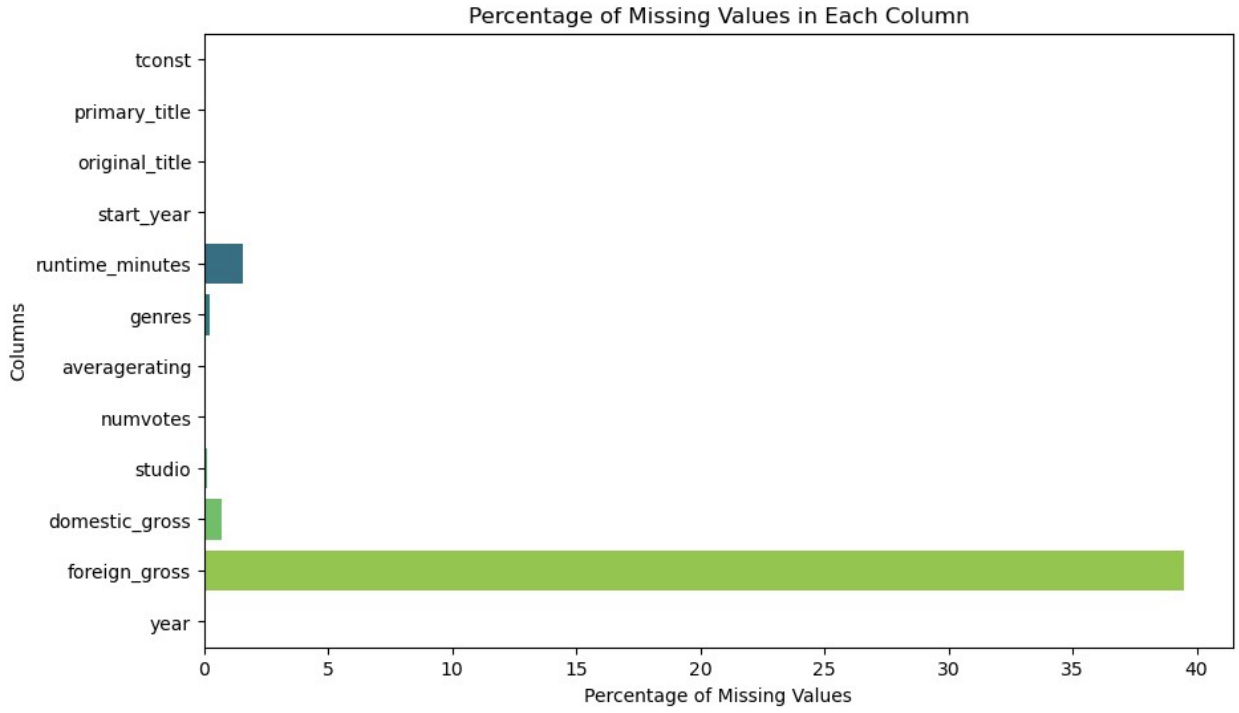
```
tconst             0.000000
primary_title      0.000000
original_title     0.000000
start_year         0.000000
runtime_minutes    0.015532
genres             0.002313
averagerating      0.000000
numvotes           0.000000
studio             0.000991
domestic_gross     0.007270
foreign_gross      0.394911
year               0.000000
dtype: float64
```

```
df7.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3026 entries, 0 to 3025
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   tconst           3026 non-null   object
 1   primary_title    3026 non-null   object
 2   original_title   3026 non-null   object
 3   start_year       3026 non-null   int64 
```

```
 4   runtime_minutes  2979 non-null   float64
 5   genres           3019 non-null   object
 6   averagerating    3026 non-null   float64
 7   numvotes         3026 non-null   int64
 8   studio           3023 non-null   object
 9   domestic_gross   3004 non-null   float64
 10  foreign_gross    1831 non-null   object
 11  year             3026 non-null   int64
dtypes: float64(3), int64(3), object(6)
memory usage: 283.8+ KB
```

```
missing_values_percentage = df7.isnull().mean() * 100

# Creating the bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=missing_values_percentage.values,
y=missing_values_percentage.index, palette='viridis')
plt.xlabel('Percentage of Missing Values')
plt.ylabel('Columns')
plt.title('Percentage of Missing Values in Each Column')
plt.show()
```




```
duplicate = df7.duplicated().sum()

#Checking for duplicates
duplicate
```

```
0
```

```
df7.shape
```

```
(3026, 12)
```

```
#Dropping the missing values for rows with the lowest amount of Data
df_cleaned = df7.dropna(subset=['primary_title', 'original_title',
'start_year', 'runtime_minutes', 'genres', 'averagerating',
'numvotes', 'studio', 'domestic_gross'])

df_cleaned.isnull().mean()
```

```
tconst             0.000000
primary_title      0.000000
original_title     0.000000
start_year         0.000000
runtime_minutes    0.000000
genres             0.000000
averagerating      0.000000
numvotes           0.000000
studio             0.000000
domestic_gross     0.000000
foreign_gross      0.401559
year               0.000000
dtype: float64
```

```
# Dropping the foreign column because more than 40 percent of the data
was missing
df_cleaned=df_cleaned.drop('foreign_gross', axis=1)

df_cleaned.isnull().mean()
```

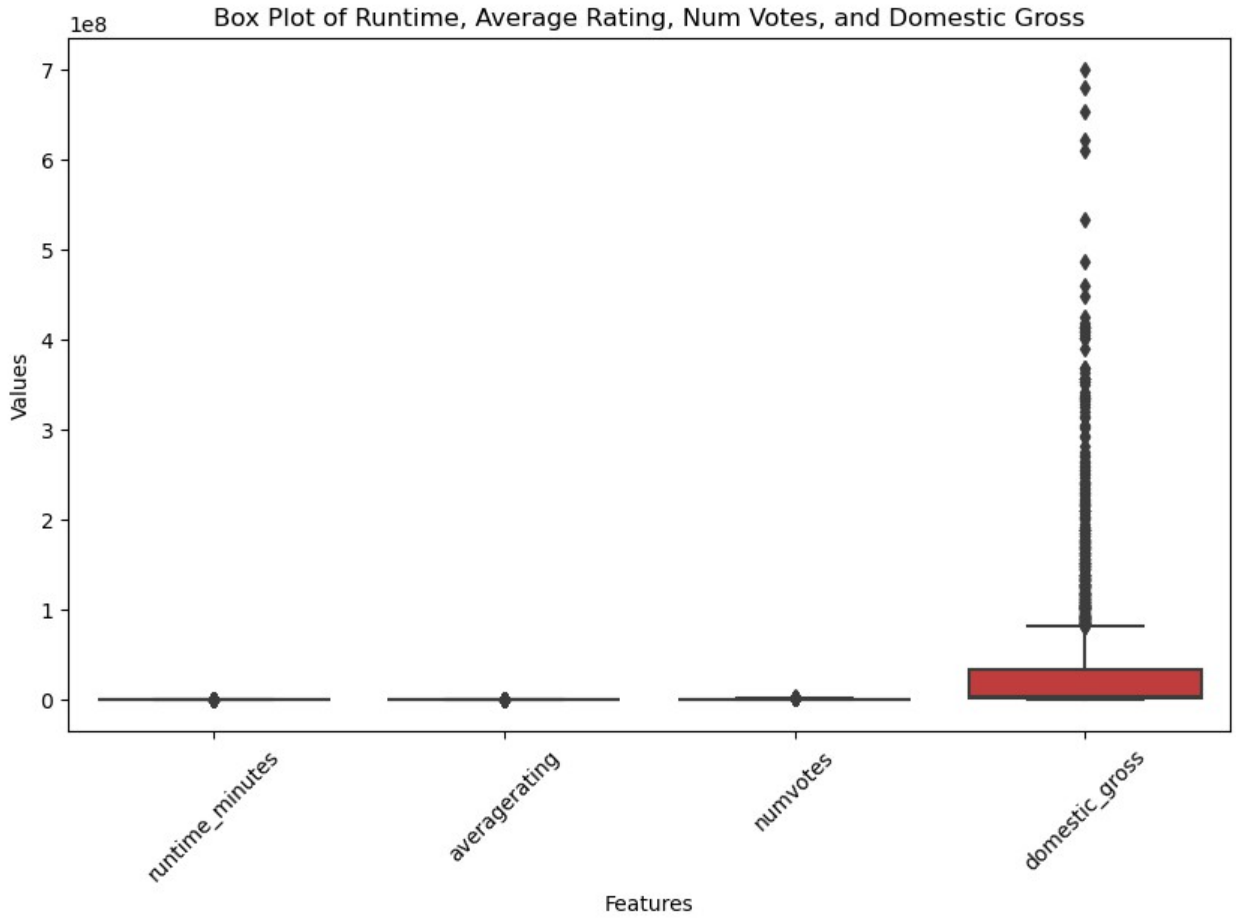
```
tconst             0.0
primary_title      0.0
original_title     0.0
start_year         0.0
runtime_minutes    0.0
genres             0.0
averagerating      0.0
numvotes           0.0
studio             0.0
domestic_gross     0.0
year               0.0
dtype: float64
```

```
df_cleaned.head()
```

```
      tconst                    primary_title  \
0  tt0315642                           Wazir
1  tt0337692                  On the Road
2  tt4339118                  On the Road
3  tt5647250                  On the Road
```

```
4  tt0359950  The Secret Life of Walter Mitty

                     original_title  start_year  runtime_minutes  \
0                             Wazir        2016            103.0
1                       On the Road        2012            124.0
2                       On the Road        2014             89.0
3                       On the Road        2016            121.0
4  The Secret Life of Walter Mitty         2013            114.0

                      genres  averagerating  numvotes   studio
domestic_gross  \
0       Action,Crime,Drama             7.1     15378  Relbig.
1100000.0
1  Adventure,Drama,Romance             6.1     37886      IFC
744000.0
2                    Drama             6.0         6      IFC
744000.0
3                    Drama             5.7       127      IFC
744000.0
4   Adventure,Comedy,Drama             7.3    275300      Fox
58200000.0

   year
0  2016
1  2012
2  2012
3  2012
4  2013
```

```
columns_of_interest = ['runtime_minutes', 'averagerating', 'numvotes',
'domestic_gross']

# Creating the box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_cleaned[columns_of_interest])
plt.title('Box Plot of Runtime, Average Rating, Num Votes, and
Domestic Gross')
plt.xlabel('Features')
plt.ylabel('Values')
plt.xticks(rotation=45)
plt.show()
#There were no outlier
```




```
df_cleaned.shape
```

```
(2951, 11)
```

```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2951 entries, 0 to 3025
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           2951 non-null   object 
 1   primary_title    2951 non-null   object 
 2   original_title   2951 non-null   object 
 3   start_year       2951 non-null   int64  
 4   runtime_minutes  2951 non-null   float64
 5   genres           2951 non-null   object 
 6   averagerating    2951 non-null   float64
 7   numvotes         2951 non-null   int64  
 8   studio           2951 non-null   object 
 9   domestic_gross   2951 non-null   float64
 10  year             2951 non-null   int64  
```

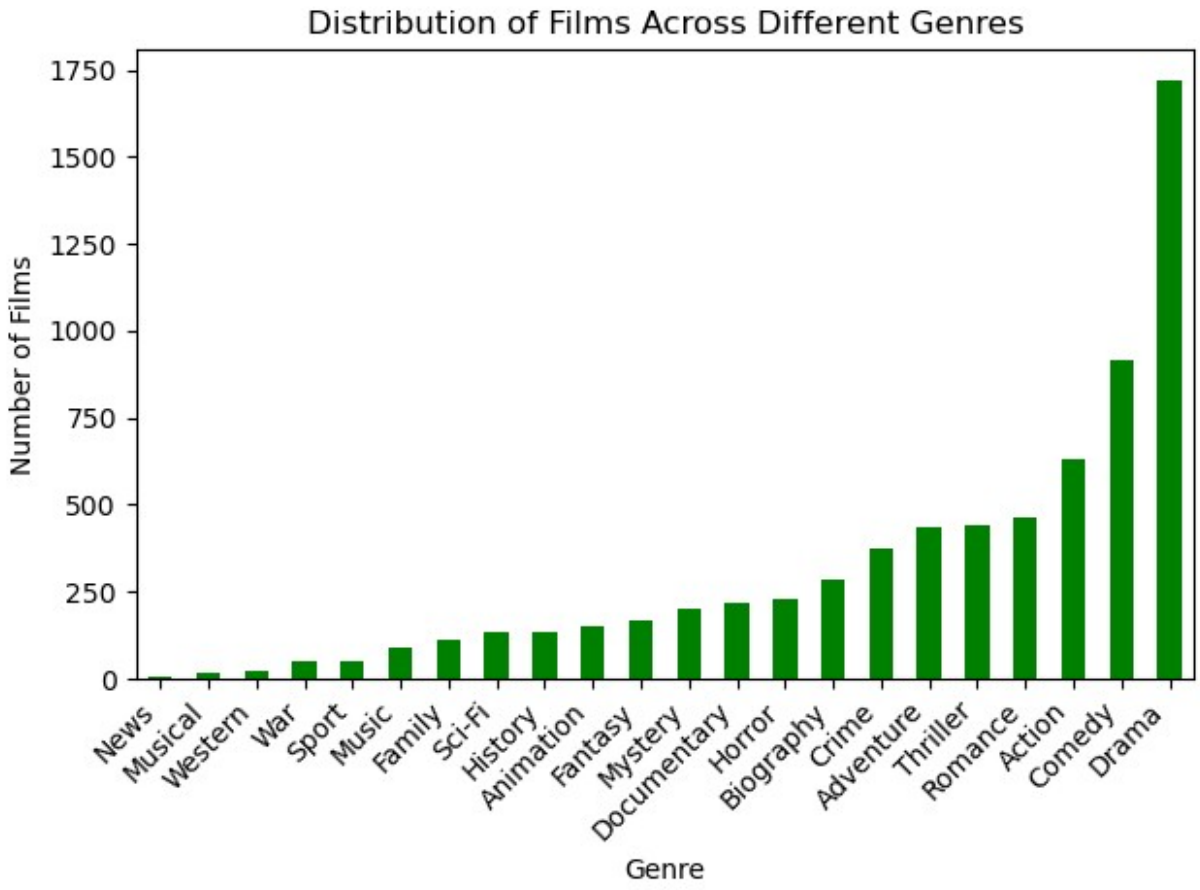
```
dtypes: float64(3), int64(3), object(5)
memory usage: 276.7+ KB
```

Data Analysis

Distribution of films across different genres

```
#Genre Analysis
# Splitting genres into individual labels
genres_split = df_cleaned['genres'].str.split(',')

# Counting the occurrence of each genre
genre_counts = {}
for genres in genres_split:
    for genre in genres:
        genre = genre.strip()
        if genre in genre_counts:
            genre_counts[genre] += 1
        else:
            genre_counts[genre] = 1

# Converting dictionary to DataFrame for easier plotting
genre_counts_df = pd.DataFrame.from_dict(genre_counts, orient='index',
columns=['Count'])
genre_counts_df = genre_counts_df.sort_values(by='Count',
ascending=True)

plt.figure(figsize=(10, 6))
genre_counts_df.plot(kind='bar', color="green", legend=None)
plt.title('Distribution of Films Across Different Genres')
plt.xlabel('Genre')
plt.ylabel('Number of Films')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



Summarizing the data

```
df_cleaned.describe()
```

```
        start_year  runtime_minutes  averagerating      numvotes  \
count  2951.000000      2951.000000    2951.000000  2.951000e+03
mean   2013.798712       107.311759       6.465673  6.320818e+04
std       2.458361        20.044198       0.994086  1.267549e+05
min    2010.000000         3.000000       1.600000  5.000000e+00
25%    2012.000000        94.000000       5.900000  2.500000e+03
50%    2014.000000       105.000000       6.600000  1.390700e+04
75%    2016.000000       118.000000       7.100000  6.674400e+04
max    2019.000000       272.000000       9.200000  1.841066e+06

       domestic_gross         year
count    2.951000e+03  2951.000000
mean     3.069066e+07  2014.090139
std      6.709869e+07     2.441868
min      1.000000e+02  2010.000000
25%      1.375000e+05  2012.000000
50%      2.000000e+06  2014.000000
```

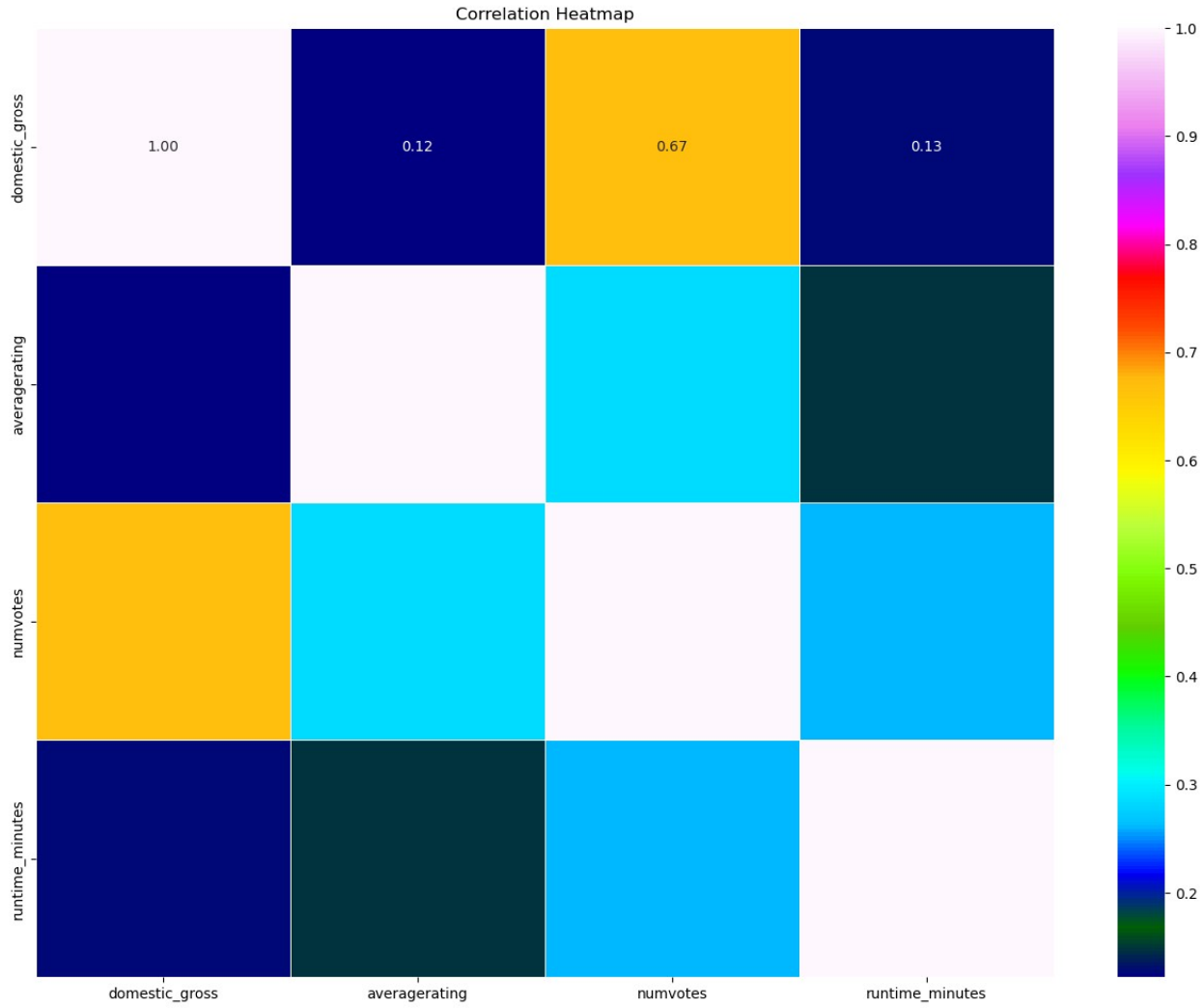
```
75%      3.245000e+07  2016.000000
max      7.001000e+08  2018.000000
```

Correlation

```
#studio_groups = df_cleaned.groupby('studio')

# Calculate correlation coefficients
correlation =df_cleaned[['domestic_gross', 'averagerating',
'numvotes', "runtime_minutes"]].corr()
correlation
```

```
                 domestic_gross  averagerating  numvotes
runtime_minutes
domestic_gross         1.000000       0.121792  0.667733
0.127272
averagerating          0.121792       1.000000  0.284124
0.148895
numvotes               0.667733       0.284124  1.000000
0.259791
runtime_minutes        0.127272       0.148895  0.259791
1.000000
```

```
plt.figure(figsize=(16, 12))
sns.heatmap(correlation, annot=True, cmap='gist_ncar', fmt=".2f",
linewidths=0.5, edgecolor="dark")
plt.title('Correlation Heatmap')
plt.show()
```



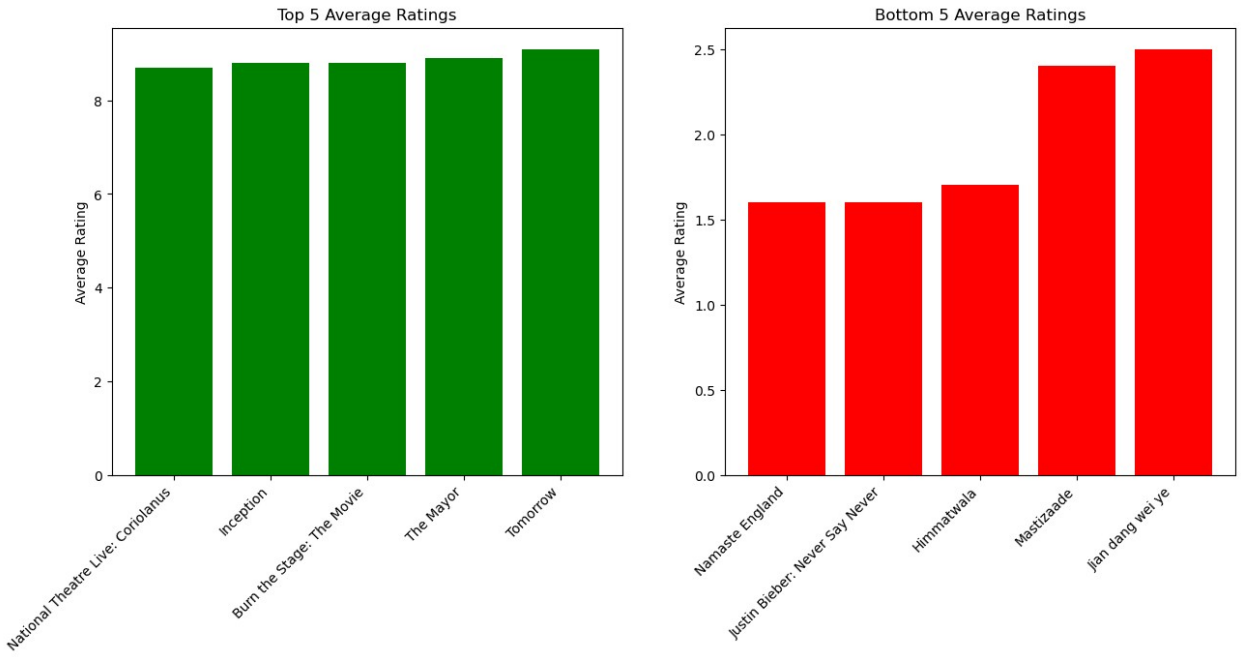
Movie With highest and lowest average rating

```
top_and_bottom_rated = df_cleaned.groupby("original_title")
["averagerating"].mean().sort_values()

top_5 = top_and_bottom_rated.tail(5)
bottom_5 = top_and_bottom_rated.head(5)

# Converting to DataFrame for visualization
top_5_df = top_5.reset_index()
bottom_5_df = bottom_5.reset_index()

# Renaming columns for clarity
top_5_df.columns = ['Original Title', 'Average Rating']
bottom_5_df.columns = ['Original Title', 'Average Rating']

# Creating subplots with side-by-side arrangement
fig, axs = plt.subplots(1, 2, figsize=(15, 6))
```

```
# Plotting top ratings
axs[0].bar(top_5_df['Original Title'], top_5_df['Average Rating'],
color='green')
axs[0].set_title('Top 5 Average Ratings')
axs[0].set_ylabel('Average Rating')
axs[0].set_xticklabels(top_5_df['Original Title'], rotation=45,
ha='right')

# Plotting bottom ratings
axs[1].bar(bottom_5_df['Original Title'], bottom_5_df['Average
Rating'], color='red')
axs[1].set_title('Bottom 5 Average Ratings')
axs[1].set_ylabel('Average Rating')
axs[1].set_xticklabels(bottom_5_df['Original Title'], rotation=45,
ha='right')
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_11680\2562747894.py:21:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  axs[0].set_xticklabels(top_5_df['Original Title'], rotation=45,
ha='right')
C:\Users\user\AppData\Local\Temp\ipykernel_11680\2562747894.py:27:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  axs[1].set_xticklabels(bottom_5_df['Original Title'], rotation=45,
ha='right')
```

```
[Text(0, 0, 'Namaste England'),
 Text(1, 0, 'Justin Bieber: Never Say Never'),
 Text(2, 0, 'Himmatwala'),
 Text(3, 0, 'Mastizaade'),
 Text(4, 0, 'Jian dang wei ye')]
```



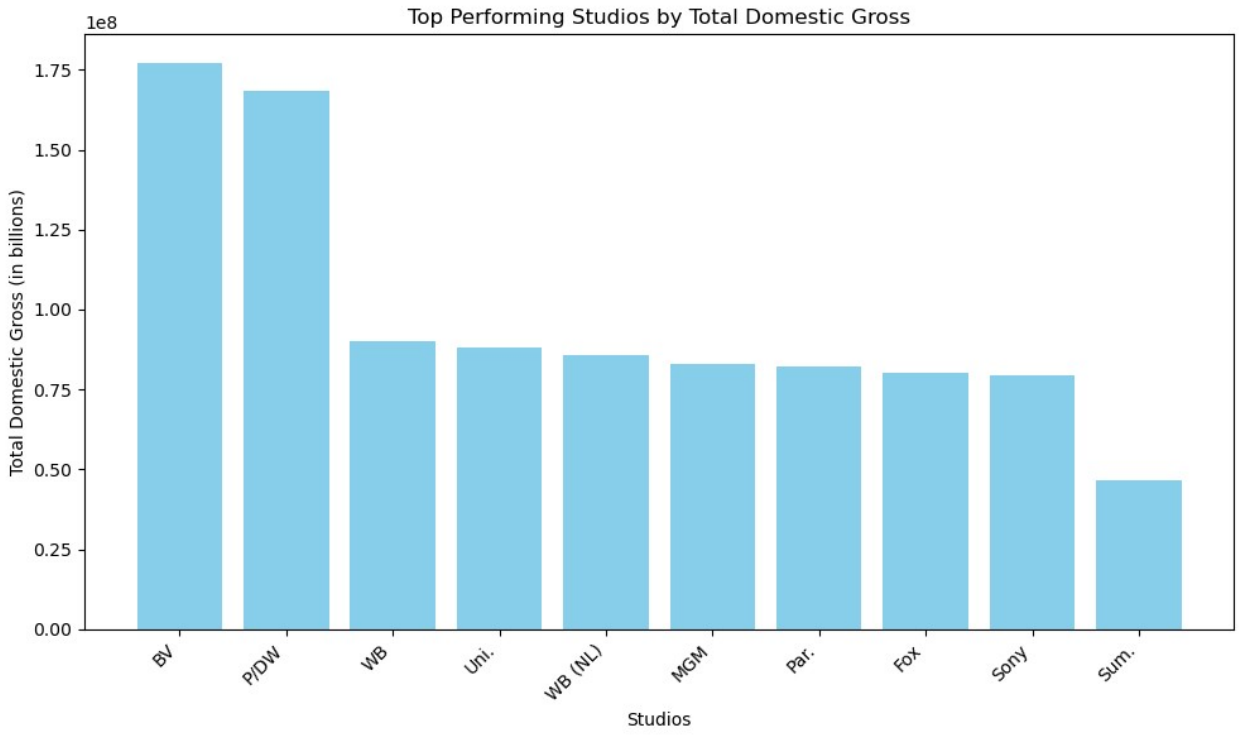
Studio performance

```
# Grouping data by studio and calculating average domestic gross and
rating
studio_performance =
df_cleaned.groupby('studio').agg({'domestic_gross': 'mean',
'averagerating': 'mean'})

# Sorting studios by average domestic gross in descending order
studio_performance =
studio_performance.sort_values(by='domestic_gross', ascending=False)

# Displaying the top performing studios
print("Top Performing Studios by Average Domestic Gross:")
print(studio_performance.head(10))

# Sorting studios by average rating in descending order
studio_performance =
studio_performance.sort_values(by='averagerating', ascending=False)

# Displaying the top rated studios
print("\nTop Rated Studios by Average Rating:")
print(studio_performance.head(10))

studio_performance_by_gross =
studio_performance.sort_values(by='domestic_gross',
ascending=False).head(10)

# Plotting bar graph for top performing studios by total domestic
```

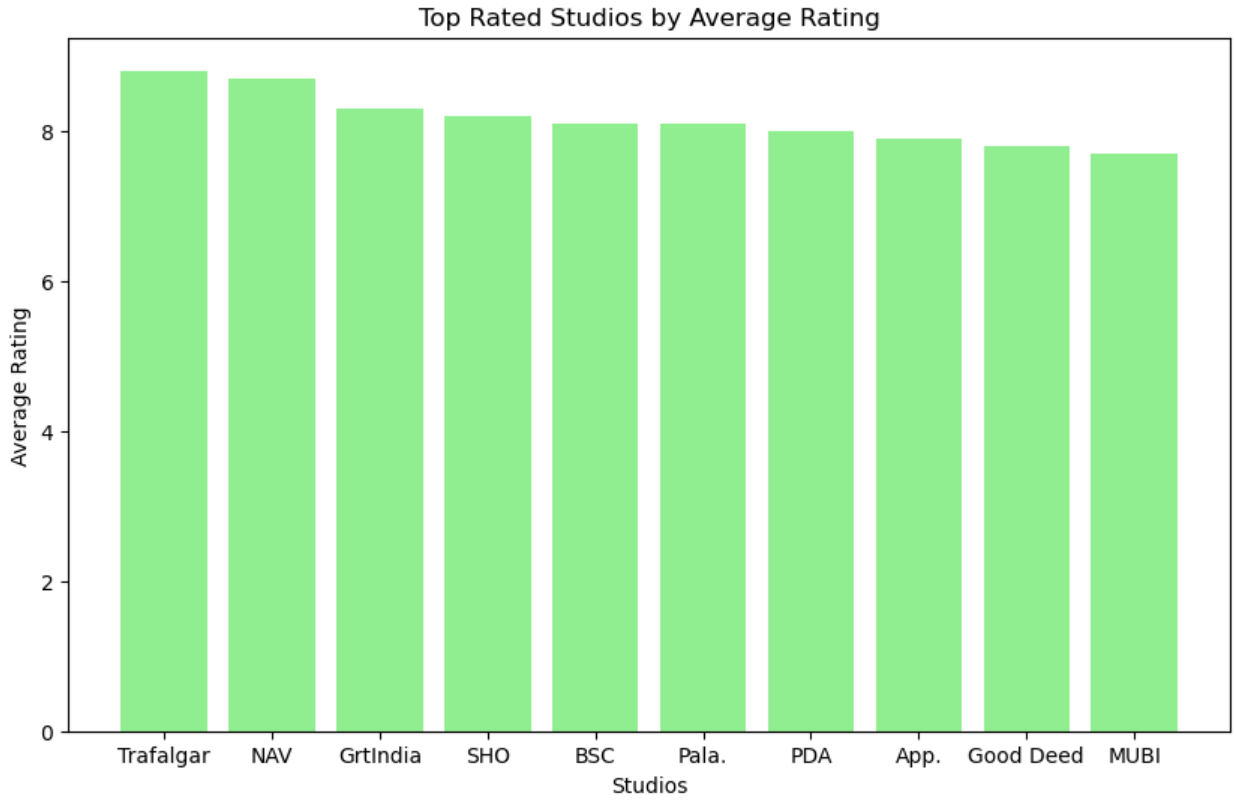
```
gross
plt.figure(figsize=(10, 6))
plt.bar(studio_performance_by_gross.index,
studio_performance_by_gross['domestic_gross'], color='skyblue')
plt.xlabel('Studios')
plt.ylabel('Total Domestic Gross (in billions)')
plt.title('Top Performing Studios by Total Domestic Gross')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Sorting studios by average rating in descending order
studio_performance_by_rating =
studio_performance.sort_values(by='averagerating',
ascending=False).head(10)

# Plotting bar graph for top rated studios by average rating
plt.figure(figsize=(10, 6))
plt.bar(studio_performance_by_rating.index,
studio_performance_by_rating['averagerating'], color='lightgreen')
plt.xlabel('Studios')
plt.ylabel('Average Rating')
plt.title('Top Rated Studios by Average Rating')
```

```
Top Performing Studios by Average Domestic Gross:
         domestic_gross  averagerating
studio
BV         1.772596e+08       6.938298
P/DW       1.682900e+08       6.760000
WB         8.991780e+07       6.547458
Uni.       8.797907e+07       6.218301
WB (NL)    8.593529e+07       6.205882
MGM        8.300000e+07       6.800000
Par.       8.204059e+07       6.424719
Fox        8.006324e+07       6.290441
Sony       7.958687e+07       6.180460
Sum.       4.637476e+07       6.547059

Top Rated Studios by Average Rating:
           domestic_gross  averagerating
studio
Trafalgar       4200000.0            8.8
NAV               26300.0            8.7
GrtIndia       20200000.0            8.3
SHO              426000.0            8.2
BSC             6700000.0            8.1
Pala.             81900.0            8.1
PDA             3620000.0            8.0
App.            3600000.0            7.9
```

```
Good Deed       6700000.0             7.8
WOW               30800.0             7.7
```



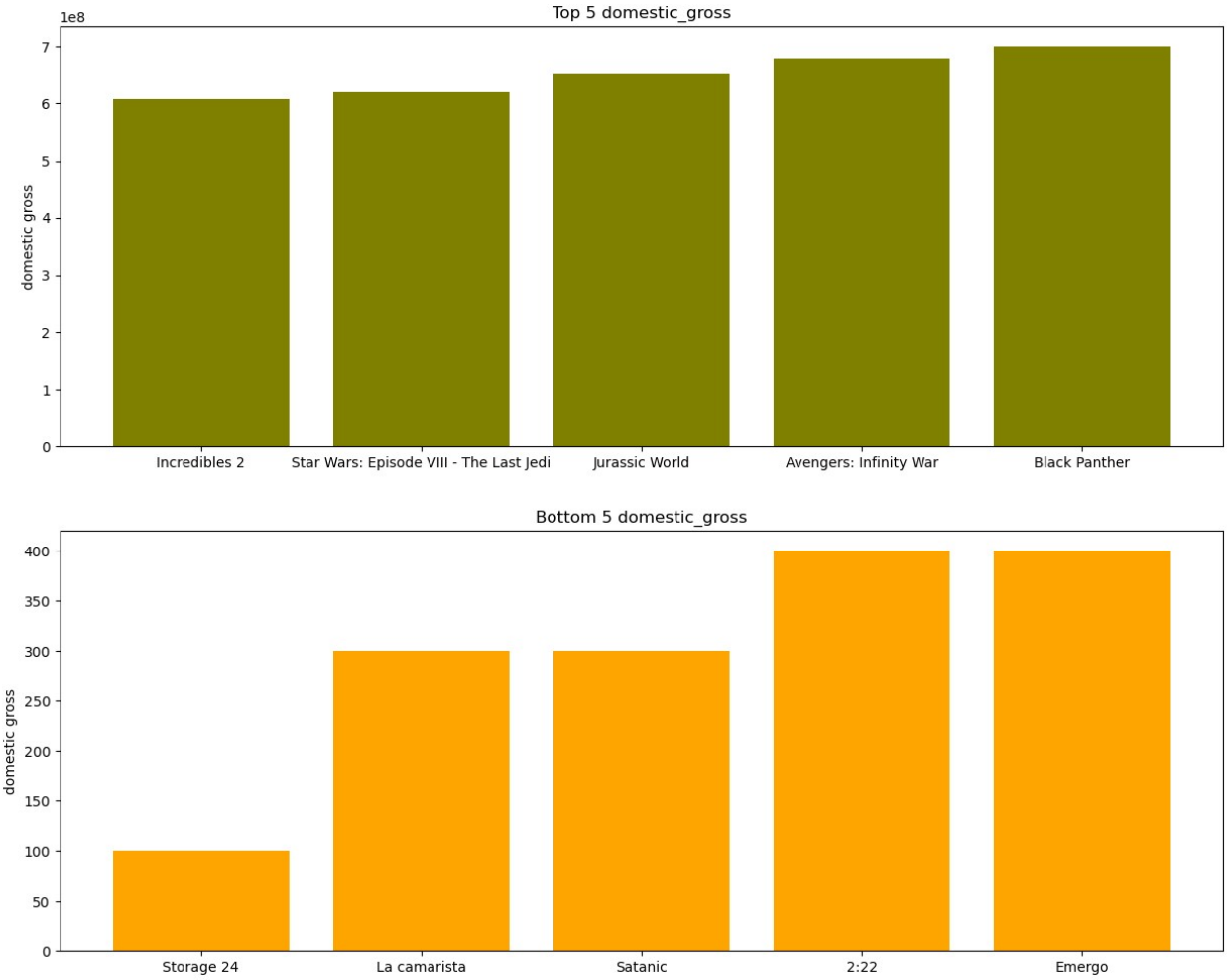

```
Text(0.5, 1.0, 'Top Rated Studios by Average Rating')
```



Movie with the highest and lowest domestic gross

```
top_and_bottom_priced=df_cleaned.groupby("original_title")
["domestic_gross"].mean().sort_values()

top_5_priced = top_and_bottom_priced.tail(5)
bottom_5_priced = top_and_bottom_priced.head(5)

# Converting to dataframe for visualization
top_5_priced_df=top_5_priced.reset_index()

bottom_5_priced_df=bottom_5_priced.reset_index()

# Renaming columns for clarity
top_5_priced_df.columns = ['Original Title', 'domestic gross']
bottom_5_priced_df.columns = ['Original Title', 'domestic gross']
fig, axs = plt.subplots(2, 1, figsize=(15, 12))

# Plotting top ratings
axs[0].bar(top_5_priced_df['Original Title'],
top_5_priced_df['domestic gross'], color='olive')
axs[0].set_title('Top 5 domestic_gross')
```

```
axs[0].set_ylabel('domestic gross')

# Plotting bottom ratings
axs[1].bar(bottom_5_priced_df['Original Title'],
bottom_5_priced_df['domestic gross'], color='orange')
axs[1].set_title('Bottom 5 domestic_gross')
axs[1].set_ylabel('domestic gross')

Text(0, 0.5, 'domestic gross')
```




```
df_cleaned.head(1)
```

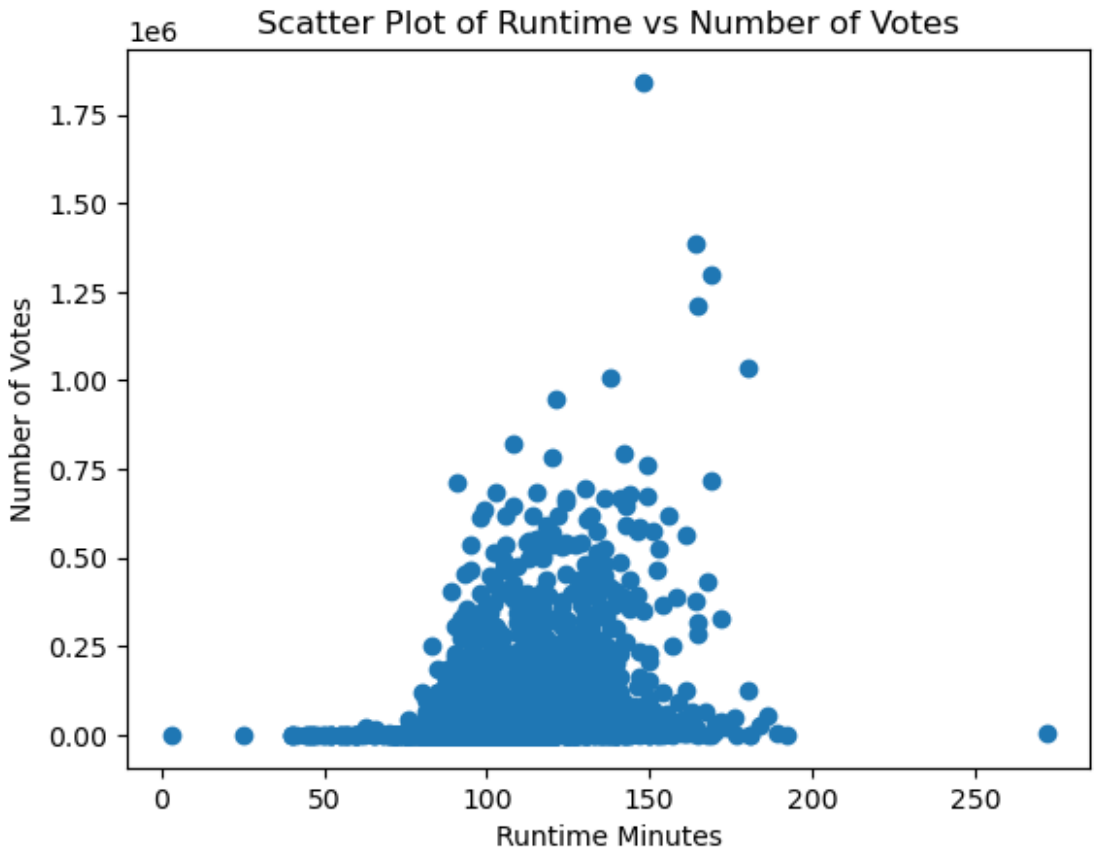
```
      tconst primary_title original_title  start_year  runtime_minutes
\
0  tt0315642         Wazir          Wazir        2016            103.0


               genres  averagerating  numvotes   studio
domestic_gross  year
```

```
0  Action,Crime,Drama            7.1     15378  Relbig.
1100000.0  2016
```

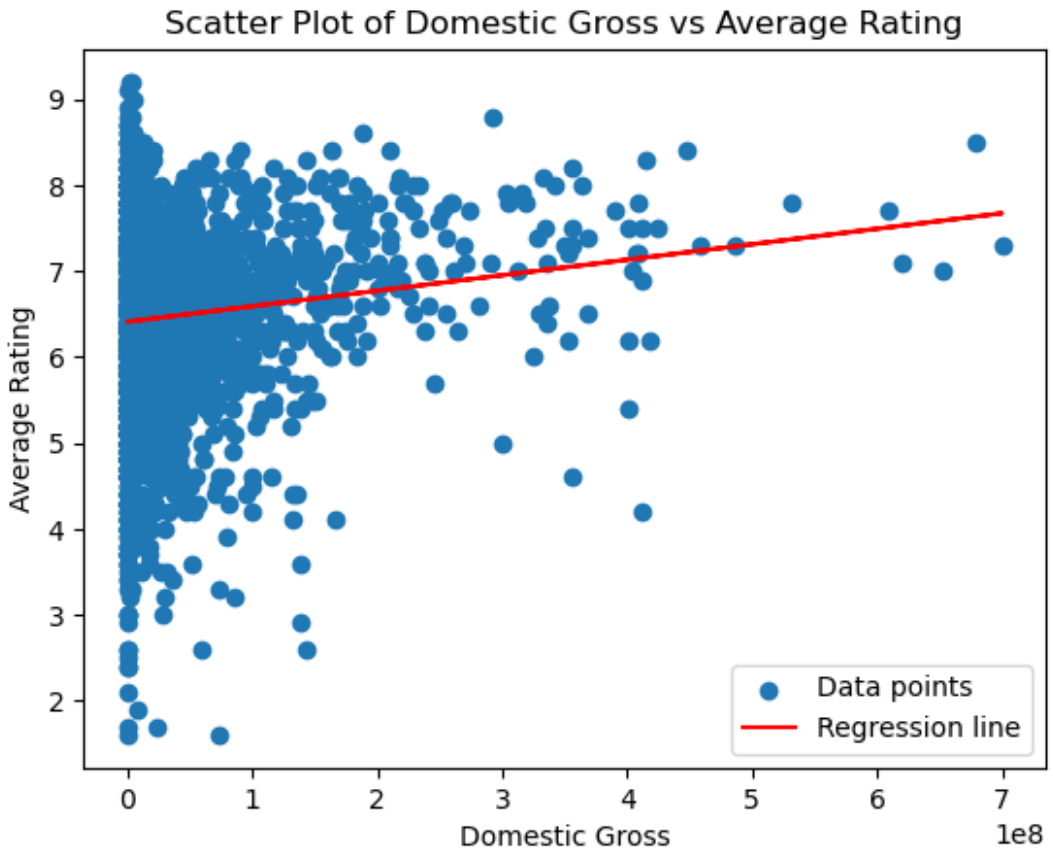
Factors that predict averagerating

```
trend = np.polyval(reg, df_cleaned["runtime_minutes"])

# Create a scatter plot
plt.scatter(x=df_cleaned["runtime_minutes"], y=df_cleaned["numvotes"],
label='Data points')

# Plot the regression line
#plt.plot(df_cleaned["runtime_minutes"], trend, color='red',
label='Regression line')

# Setting labels and title
plt.xlabel("Runtime Minutes")
plt.ylabel("Number of Votes")
plt.title("Scatter Plot of Runtime vs Number of Votes")

# Print the coefficients of the linear regression model
print("The- slope, intercept):", reg)
```

```
The- slope, intercept): [1.80438938e-09 6.41029475e+00]
```



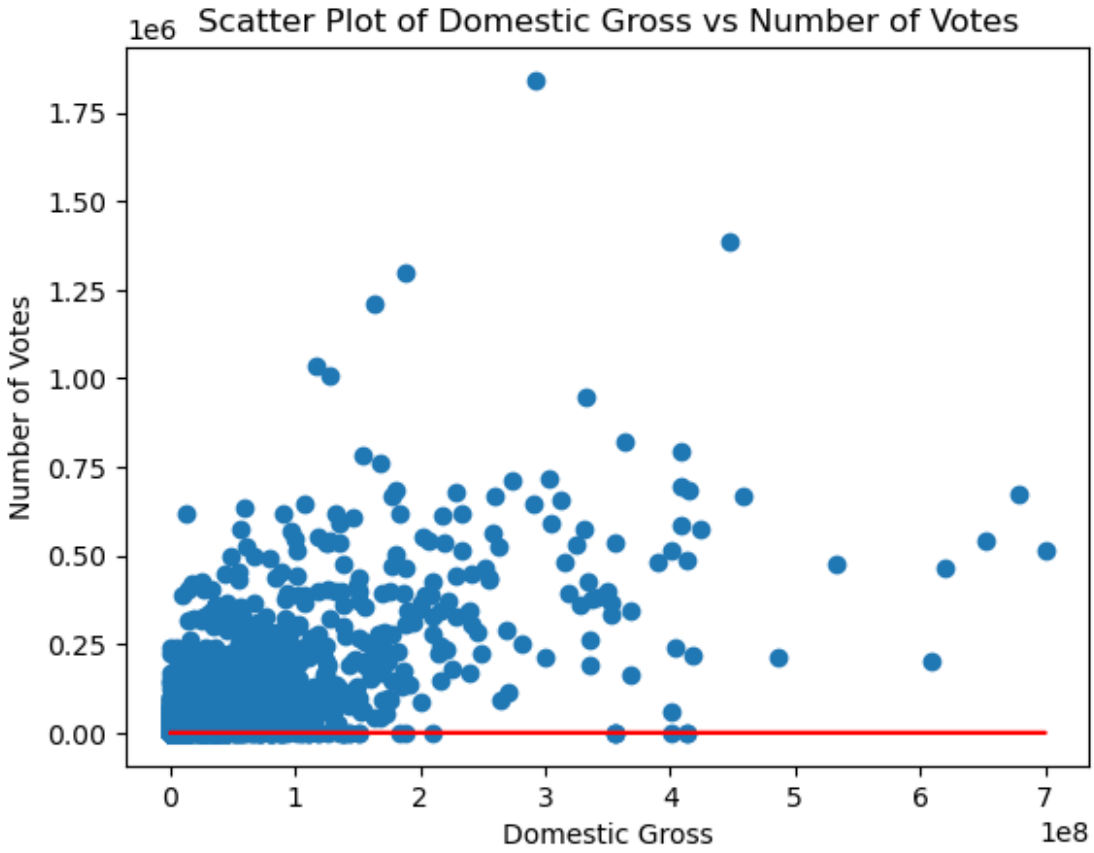

```
# Fit a linear regression model
reg = np.polyfit(df_cleaned["domestic_gross"],
df_cleaned["averagerating"], deg=1)
trend = np.polyval(reg, df_cleaned["domestic_gross"])

# Create a scatter plot
plt.scatter(x=df_cleaned["domestic_gross"],
y=df_cleaned["averagerating"], label='Data points')

# Plot the regression line
plt.plot(df_cleaned["domestic_gross"], trend, color='red',
label='Regression line')

# Setting labels and title
plt.xlabel("Domestic Gross")
plt.ylabel("Average Rating")
plt.title("Scatter Plot of Domestic Gross vs Average Rating")


# Print the coefficients of the linear regression model
print("Coefficients of the linear regression model (slope,
intercept):", reg)
```




```
Coefficients of the linear regression model (slope, intercept):
[1.80438938e-09 6.41029475e+00]
```

```
# Fit a linear regression model

trend = np.polyval(reg, df_cleaned["domestic_gross"])

# Create a scatter plot
plt.scatter(x=df_cleaned["domestic_gross"], y=df_cleaned["numvotes"],
label='Data points')

# Plot the regression line
plt.plot(df_cleaned["domestic_gross"], trend, color='red',
label='Regression line')

# Setting labels and title
plt.xlabel("Domestic Gross")
plt.ylabel("Number of Votes")
plt.title("Scatter Plot of Domestic Gross vs Number of Votes")


# Print the coefficients of the linear regression model
print("Coefficients of the linear regression model (slope,
intercept):", reg)
```

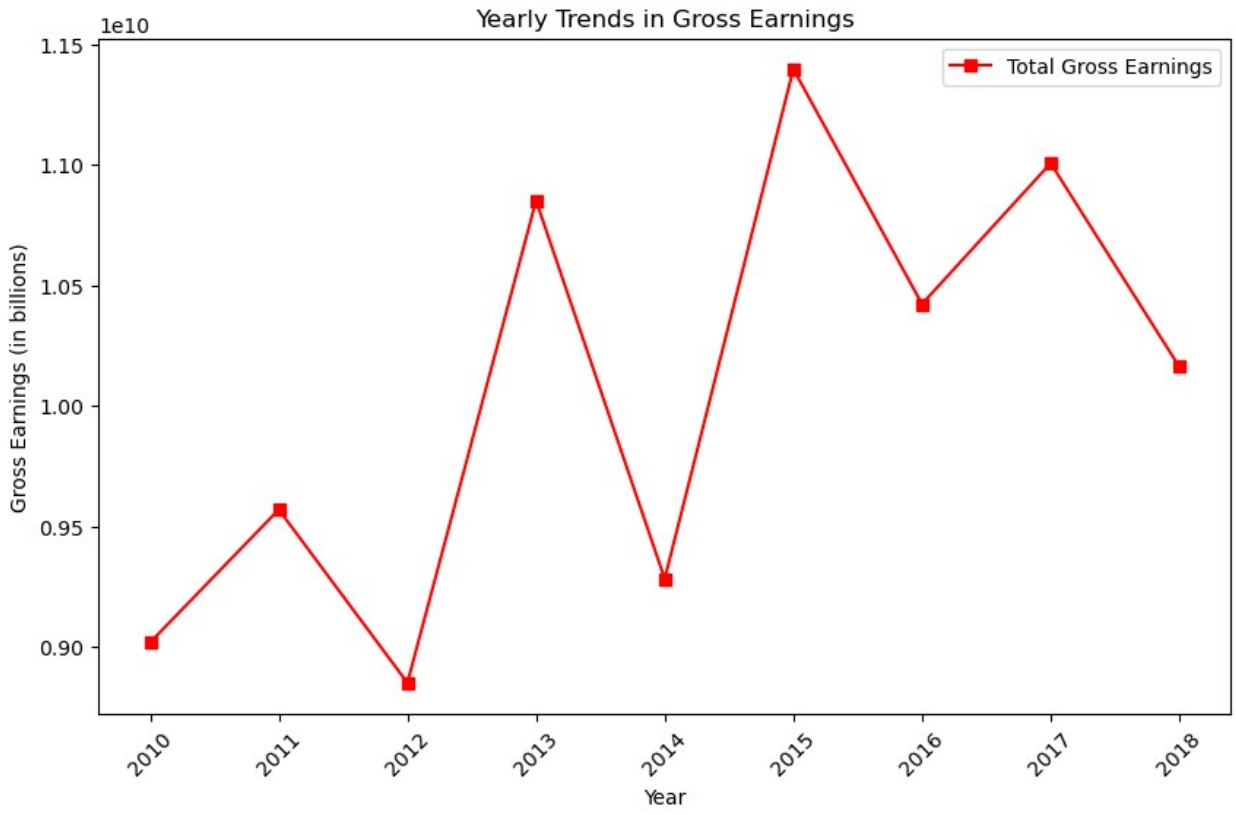
```
Coefficients of the linear regression model (slope, intercept):
[1.80438938e-09 6.41029475e+00]
```



Trend in Earnings over the years

```
grouped_data = df_cleaned.groupby('year').agg({'domestic_gross':
'sum'}).reset_index()

# Plotting trends in gross earnings over the years
plt.figure(figsize=(10, 6))

# Line chart for total domestic gross earnings
plt.plot(grouped_data['year'], grouped_data['domestic_gross'],
marker='s', color='r', label='Total Gross Earnings')

# Adding labels and title
plt.xlabel('Year')
plt.ylabel('Gross Earnings (in billions)')
plt.title('Yearly Trends in Gross Earnings')
plt.xticks(rotation=45)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x20c515dbd50>
```



Saving the df_cleaned to PC

```
file_path = r"C:\Users\user\Desktop\Project\df_cleaned.csv"

# Save the DataFrame to a CSV file
df_cleaned.to_csv(file_path, index=False)

print("CSV file saved successfully!")
```

```
CSV file saved successfully!
```