PHASE ONE PROJECT FINAL SUBMISSION Caroline Gesaka Nyairo Part Time

1.Data Colllection and Data Loading a. Find the correct dataset to use. b. Load the datasets. c. Merge the datasets.

In [1]:
```
#a.Find the correct datasets to use
#tn.movie_budgets has info on budget-analyze financial performance
#imdb.title basics has info on movie basics.ie genres titles and release years-
#will show relationship between budget allocation and box office success and also d
#bom.movie_gross-compliments the budget and genre info
#imdb.title.ratings csv-show additional factors to consider.

#(import the libraries)

import pandas as pd #for data manipulation
import numpy as np #for numeric computation
import matplotlib.pyplot as plt #for ploting our data and creating visualization
%matplotlib inline
import seaborn as sns # data visualization
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#b. load and read the datasets
movie_budgets_df= pd.read_csv(r"C:\Users\Caro\Desktop\PROJECTS\PhaseOne Final Proje
title_basics_df= pd.read_csv(r"C:\Users\Caro\Desktop\PROJECTS\PhaseOne Final Projec
movie_gross_df = pd.read_csv(r"C:\Users\Caro\Desktop\PROJECTS\PhaseOne Final Projec
title_ratings_df = pd.read_csv(r"C:\Users\Caro\Downloads\dsc-phase-1-project-master
```

In [3]:
```
#c. Merge the four datasets
# Merge movie_budgets with title_basics
merged_data_df= pd.merge(movie_budgets_df, title_basics_df, left_on='movie', right_

# Merge merged_data with movie_gross
merged_data_df = pd.merge(merged_data_df, movie_gross_df, left_on='movie', right_or

# Merge merged_data with title_ratings
merged_data_df = pd.merge(merged_data_df, title_ratings_df, on='tconst', how='inner

# Print the first few rows of the merged dataset
merged_data_df.head()
```

Out[3]:

	id	release_date	movie	production_budget	domestic_gross_x	worldwide_gross	tconst
0	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875	tt1298650
1	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	tt2395427
2	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200	tt4154756
3	9	Nov 17, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209	tt0974015
4	10	Nov 6, 2015	Spectre	$300,000,000	$200,074,175	$879,620,923	tt2379713

2.DATA CLEANING a.Handle the missing values. b.Check and remove duplicates. c.Convert Datatypes (if neccessary)

In [4]:
```
#a. Handling the missing values
# Check for missing values before dropping
print("Missing values  :\n", merged_data_df.isnull().sum())
#there are missing values

#b..check and remove duplicates
# Check for duplicate rows
print("Number of duplicate rows:", merged_data_df.duplicated().sum())
#there are no duplicates

#c.Convert datatypes(if necessary)
#regex= regular expressions
# Convert financial columns to numeric
financial_columns = ['production_budget', 'domestic_gross_x', 'worldwide_gross', 'f
merged_data_df[financial_columns] = merged_data_df[financial_columns].replace('[\$,
```

```
Missing values  :
 id                    0
release_date           0
movie                  0
production_budget      0
domestic_gross_x       0
worldwide_gross        0
tconst                 0
primary_title          0
original_title         0
start_year             0
runtime_minutes       30
genres                 7
title                  0
studio                 0
domestic_gross_y       1
foreign_gross        198
year                   0
averagerating          0
numvotes               0
dtype: int64
Number of duplicate rows: 0
```

In [5]:
```
#a. Handling my missing values(runtime minutes,genre,domestic gross and foreign gro
# Impute missing values for 'runtime_minutes' with the median
median_runtime = merged_data_df['runtime_minutes'].median()
merged_data_df['runtime_minutes'].fillna(median_runtime, inplace=True)

# fill missing values for 'genres' with the mode
mode_genre = merged_data_df['genres'].mode()[0]
merged_data_df['genres'].fillna(mode_genre, inplace=True)

# fill missing value for 'domestic_gross_y' with the mean
mean_domestic_gross = merged_data_df['domestic_gross_y'].mean()
merged_data_df['domestic_gross_y'].fillna(mean_domestic_gross, inplace=True)

# fill missing value for 'foreign_gross' with 0 (assuming it represents no foreign
merged_data_df['foreign_gross'].fillna(0, inplace=True)

#print("Missing values after cleaning :\n",
merged_data_df.isnull().sum()
```

Out[5]:
```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross_x     0
worldwide_gross      0
tconst               0
primary_title        0
original_title       0
start_year           0
runtime_minutes      0
genres               0
title                0
studio               0
domestic_gross_y     0
foreign_gross        0
year                 0
averagerating        0
numvotes             0
dtype: int64
```

3.Exploratory Data Analysis(EDA) a. Explore the cleaned dataset. b.Analyz distributions,relationships between variables and identify patterns. c.Visualize the data. d.Handle Outliers e.Calculate summary statistics to summariye the dataset.

In [6]:
```
#a.Explore the cleaned dataset
#Load the cleaned dataset
merged_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1413 entries, 0 to 1412
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 1413 non-null   int64  
 1   release_date       1413 non-null   object 
 2   movie              1413 non-null   object 
 3   production_budget  1413 non-null   float64
 4   domestic_gross_x   1413 non-null   float64
 5   worldwide_gross    1413 non-null   float64
 6   tconst             1413 non-null   object 
 7   primary_title      1413 non-null   object 
 8   original_title     1413 non-null   object 
 9   start_year         1413 non-null   int64  
 10  runtime_minutes    1413 non-null   float64
 11  genres             1413 non-null   object 
 12  title              1413 non-null   object 
 13  studio             1413 non-null   object 
 14  domestic_gross_y   1413 non-null   float64
 15  foreign_gross      1413 non-null   float64
 16  year               1413 non-null   int64  
 17  averagerating      1413 non-null   float64
 18  numvotes           1413 non-null   int64  
dtypes: float64(7), int64(4), object(8)
memory usage: 209.9+ KB
```

In [7]:
```
# Summary statistics for numerical columns
summary_stats = merged_data_df.describe()
summary_stats.head()
```

Out[7]:

	id	production_budget	domestic_gross_x	worldwide_gross	start_year	runtime_
count	1413.000000	1.413000e+03	1.413000e+03	1.413000e+03	1413.000000	1413
mean	51.659590	4.628144e+07	6.012142e+07	1.503449e+08	2013.644020	107
std	28.606088	5.517378e+07	8.444932e+07	2.327543e+08	2.531381	19
min	1.000000	5.000000e+04	0.000000e+00	0.000000e+00	2010.000000	3
25%	27.000000	1.000000e+07	7.018188e+06	1.753600e+07	2011.000000	95

In [8]:
```
#b. Analyz the distributions
# Frequency distribution of genres
genre_counts = merged_data_df['genres'].value_counts()
print(genre_counts)
```

```
genres
Drama                          101
Adventure,Animation,Comedy      61
Comedy,Drama,Romance            51
Comedy,Drama                    50
Action,Adventure,Sci-Fi         44
                              ...
Biography,Drama,Musical          1
Fantasy,Horror,Thriller          1
Comedy,Mystery                   1
Mystery,Thriller                 1
Comedy,Thriller                  1
Name: count, Length: 216, dtype: int64
```

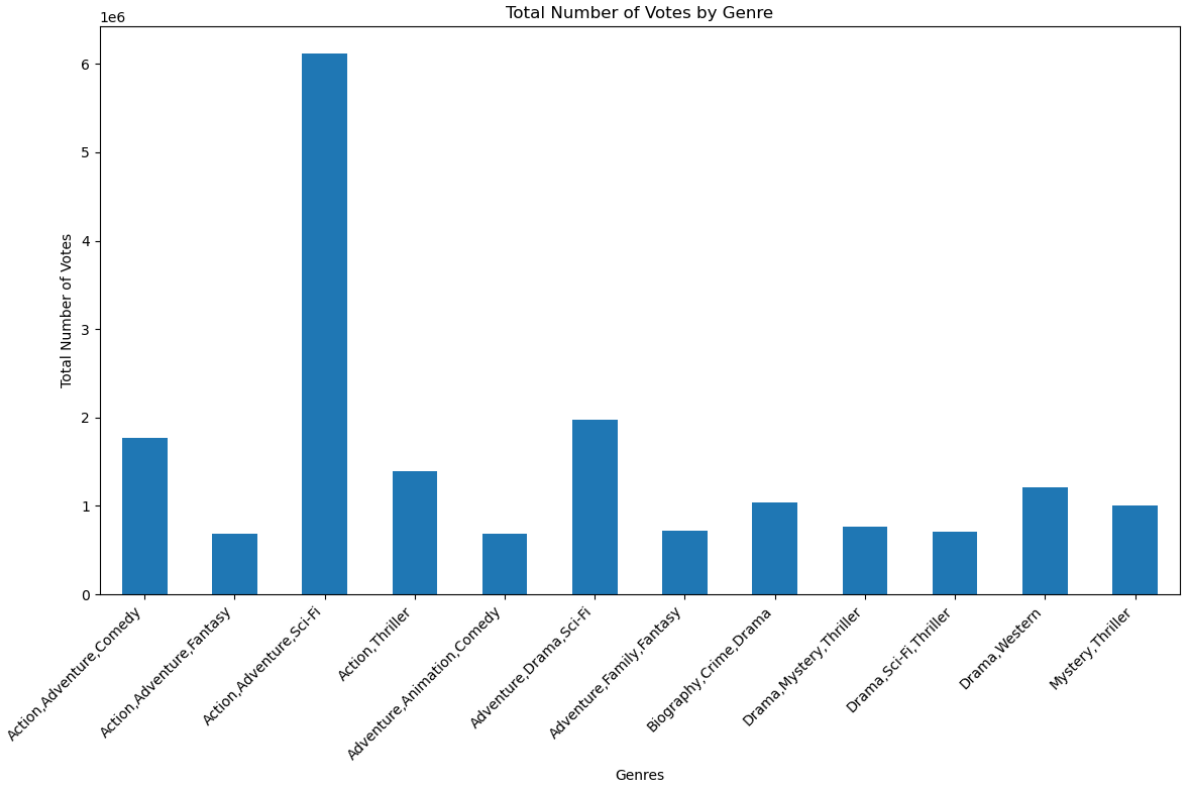
In [10]:
```
merged_data_df_sorted=merged_data_df.sort_values(by='numvotes', ascending=False).he
merged_data_df_sorted
```

Out[10]:

	id	release_date	movie	production_budget	domestic_gross_x	worldwide_gross	tcor
93	38	Jul 16, 2010	Inception	160000000.0	292576195.0	8.355246e+08	tt13756
5	11	Jul 20, 2012	The Dark Knight Rises	275000000.0	448139099.0	1.084439e+09	tt13458
90	32	Nov 5, 2014	Interstellar	165000000.0	188017894.0	6.663794e+08	tt08166
194	69	Dec 25, 2012	Django Unchained	100000000.0	162805434.0	4.499483e+08	tt18537
197	76	Dec 25, 2013	The Wolf of Wall Street	100000000.0	116900694.0	3.898704e+08	tt09938
250	52	Feb 19, 2010	Shutter Island	80000000.0	128012934.0	2.994618e+08	tt11308
77	14	Aug 1, 2014	Guardians of the Galaxy	170000000.0	333172112.0	7.708675e+08	tt20153
350	56	Feb 12, 2016	Deadpool	58000000.0	363070709.0	8.010256e+08	tt14310
245	38	Mar 23, 2012	The Hunger Games	80000000.0	408010692.0	6.779234e+08	tt13921
113	75	May 15, 2015	Mad Max: Fury Road	150000000.0	153636354.0	3.700980e+08	tt13921
315	36	Oct 3, 2014	Gone Girl	61000000.0	167767189.0	3.685672e+08	tt22679
12	19	Dec 14, 2012	The Hobbit: An Unexpected Journey	250000000.0	303003568.0	1.017004e+09	tt09036
179	16	Oct 4, 2013	Gravity	110000000.0	274092705.0	6.936987e+08	tt14544
36	48	May 3, 2013	Iron Man 3	200000000.0	408992272.0	1.215392e+09	tt13008
111	70	May 6, 2011	Thor	150000000.0	181030624.0	4.493266e+08	tt08003
35	47	Jun 18, 2010	Toy Story 3	200000000.0	415004880.0	1.068880e+09	tt04357
187	46	Oct 2, 2015	The Martian	108000000.0	228433663.0	6.552714e+08	tt36593
2	7	Apr 27, 2018	Avengers: Infinity War	300000000.0	678815482.0	2.048134e+09	tt41547
128	10	Jul 22, 2011	Captain America: The First Avenger	140000000.0	176654505.0	3.705698e+08	tt04583
79	16	Apr 4, 2014	Captain America: The Winter Soldier	170000000.0	259746958.0	7.144019e+08	tt18438

In [11]:
```
#1 Number of votes vs Genres
# Grouped bar plot for numvotes by genres
plt.figure(figsize=(12, 8))
merged_data_df_sorted.groupby('genres')['numvotes'].sum().plot(kind='bar')
```

```
plt.xlabel('Genres')
plt.ylabel('Total Number of Votes')
plt.title('Total Number of Votes by Genre')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [12]:
```
#identifying outliers in the above
# Calculate quartiles
Q1 = np.percentile(merged_data_df['numvotes'], 25)
Q3 = np.percentile(merged_data_df['numvotes'], 75)

# Calculate IQR
IQR = Q3 - Q1

# Calculate lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = merged_data_df[(merged_data_df['numvotes'] < lower_bound) | (merged_data

# Display outliers
outliers.head()
```

Out[12]:

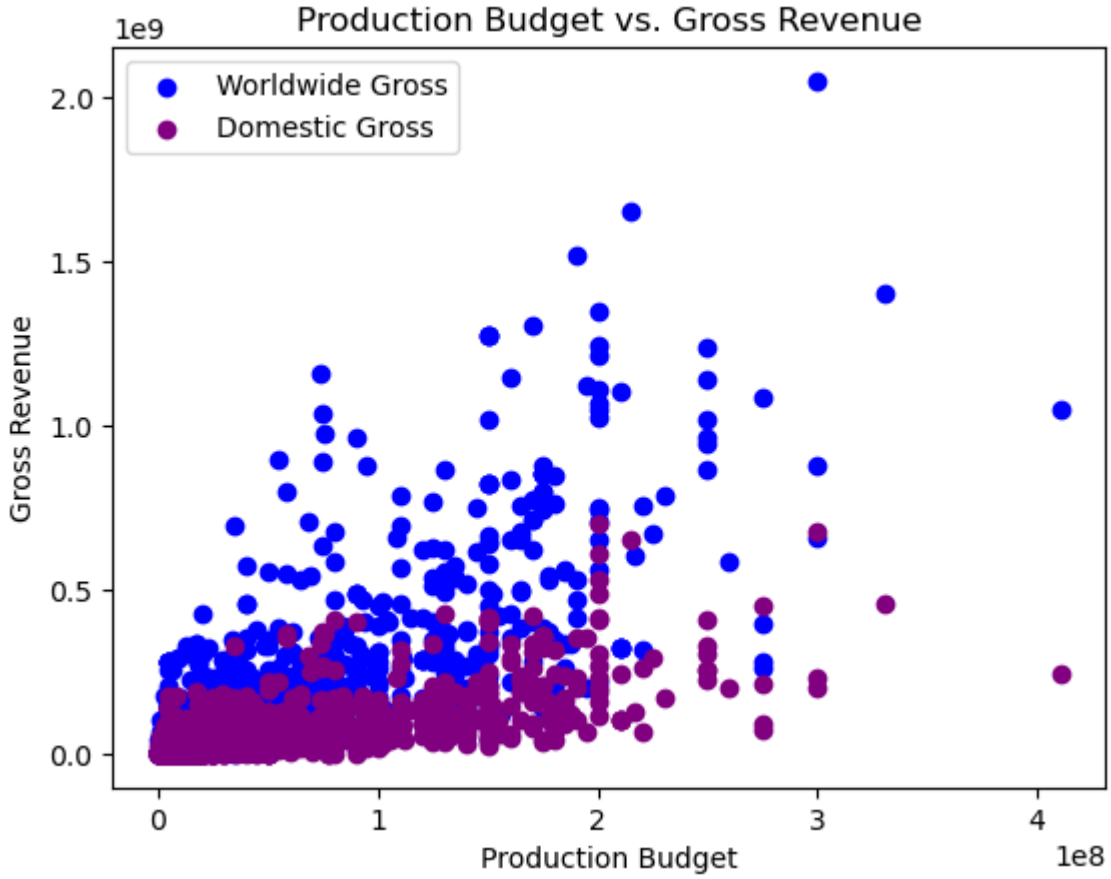
	id	release_date	movie	production_budget	domestic_gross_x	worldwide_gross	tconst
0	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	tt1298650
1	4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	tt2395427
2	7	Apr 27, 2018	Avengers: Infinity War	300000000.0	678815482.0	2.048134e+09	tt4154756
3	9	Nov 17, 2017	Justice League	300000000.0	229024295.0	6.559452e+08	tt0974015
4	10	Nov 6, 2015	Spectre	300000000.0	200074175.0	8.796209e+08	tt2379713

In [13]:
```
#Conclusion and Recommendations on genres vs numvotes
recommendations = """
###
Conclusions:
The bar graph shows that Action,Adventure and Sci-Fi have the most numvotes than th
Movies with significantly high number of votes shows movies that have gaine popular
Movies with significantly less number of votes are less popular.
Lower outliers may represent niche films.
Certain movie geres when combined with other genres boost popularity of the niche g


Recommendations:
1. Focus on the high number of votes on movie genres as they indicate popularity.
2. Utilize the Niche films to target niche audiences effectively.
3. Pair up popular movie genres with the niche genres to maximizes on increasing pc

"""
print(recommendations)
```

```
###
Conclusions:
The bar graph shows that Action,Adventure and Sci-Fi have the most numvotes than t
he rest of the genres.
Movies with significantly high number of votes shows movies that have gaine popula
rity.
Movies with significantly less number of votes are less popular.
Lower outliers may represent niche films.
Certain movie geres when combined with other genres boost popularity of the niche
genres


Recommendations:
1. Focus on the high number of votes on movie genres as they indicate popularity.
2. Utilize the Niche films to target niche audiences effectively.
3. Pair up popular movie genres with the niche genres to maximizes on increasing p
opularity
```

In [14]:
```
#2 Product budget vs Gross Revenue
# Scatter plot of budget vs. worldwide gross
```

```
plt.scatter(merged_data_df['production_budget'], merged_data_df['worldwide_gross'],

# Scatter plot of budget vs. domestic gross
plt.scatter(merged_data_df['production_budget'], merged_data_df['domestic_gross_y']

plt.xlabel('Production Budget')
plt.ylabel('Gross Revenue')
plt.title('Production Budget vs. Gross Revenue')
plt.legend()
plt.show()
```

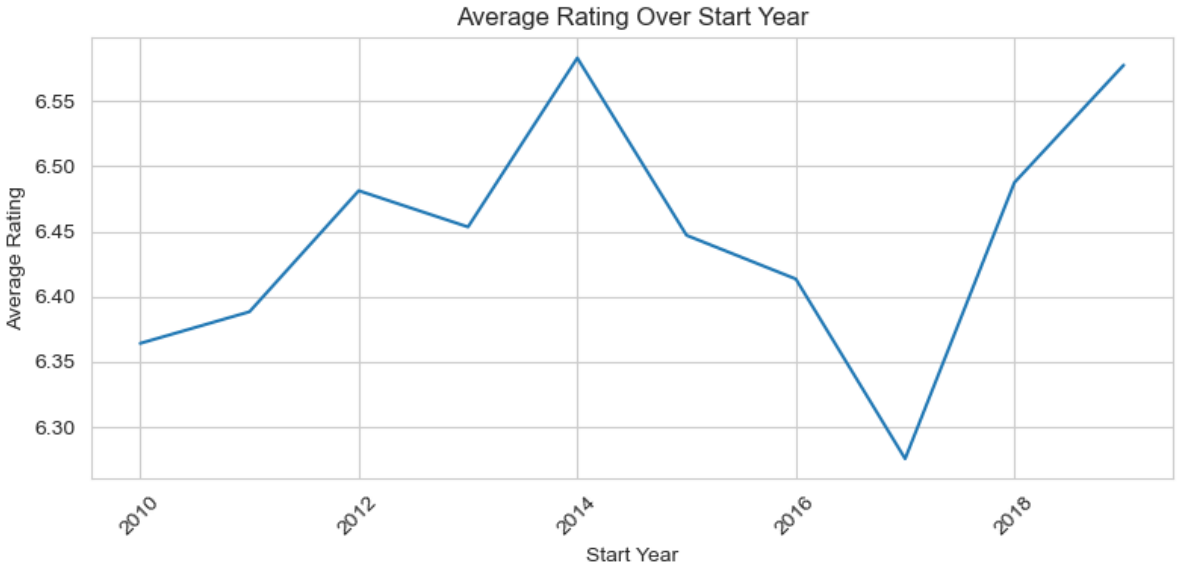


In [15]:
```
#Conclusion and Recommendations on Production revenue vs Gross revenue
recommendations = """
###Conclusions:
Scatter plot suggests +ve correlation between production revenue and gross revenue.
Movies with higher production budget recieve higher grosss revenue.
As production budget increase,gross revenue increases as well.
Some movies with high production budgets may not perform well at box office resulti


Recommendations:
1. Allocate resourses carefully based on thorough analysis.
2. Production companies can valuable resources and expertise to optimiye productior
3. Production companies make their content to meet target audiences demands,
hence increasing success in maximizing gross revenues.
"""
print(recommendations)
```

```
###Conclusions:
Scatter plot suggests +ve correlation between production revenue and gross revenu
e.
Movies with higher production budget recieve higher grosss revenue.
As production budget increase,gross revenue increases as well.
Some movies with high production budgets may not perform well at box office result
ing to low gross revenue.


Recommendations:
1. Allocate resoursces carefully based on thorough analysis.
2. Production companies can valuable resources and expertise to optimiye productio
n outcomes and maximize gross revenues.
3. Production companies make their content to meet target audiences demands,
hence increasing success in maximizing gross revenues.
```

In [16]:

```
#3.Start year vs Average rating
# Set the style of seaborn
sns.set_style("whitegrid")

# Create a line plot for start year vs. average rating
plt.figure(figsize=(8, 4))
sns.lineplot(x='start_year', y='averagerating', data=merged_data_df, ci=None)
plt.title('Average Rating Over Start Year')
plt.xlabel('Start Year')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



In [17]:

```
 #Conclusion and Recommendations on Start year vs Average rating
recommendations = """
###
Conclusions:
The line plot illustrates the trends of average ratings of movies over the years.
Variation shows changes in the reception of movies over the years.
Average ratings change over time depending on audience preferences.



Recommendations:
1. Access trends and identify patterns so as to make more informed decisions.
2. Change in movie quality or audince preference may vary over the years,
hence producers should make informed decisions regarding release strategies.
```

```
3.Audience engagement through movie trailers can help them
analyz audience expectations and improve in the production of the movies to audienc
"""
print(recommendations)
```

```
###
Conclusions:
The line plot illustrates the trends of average ratings of movies over the years.
Variation shows changes in the reception of movies over the years.
Average ratings change over time depending on audience preferences.



Recommendations:
1. Access trends and identify patterns so as to make more informed decisions.
2. Change in movie quality or audince preference may vary over the years,
hence producers should make informed decisions regarding release strategies.
3.Audience engagement through movie trailers can help them
analyz audience expectations and improve in the production of the movies to audien
ce preferences.
```

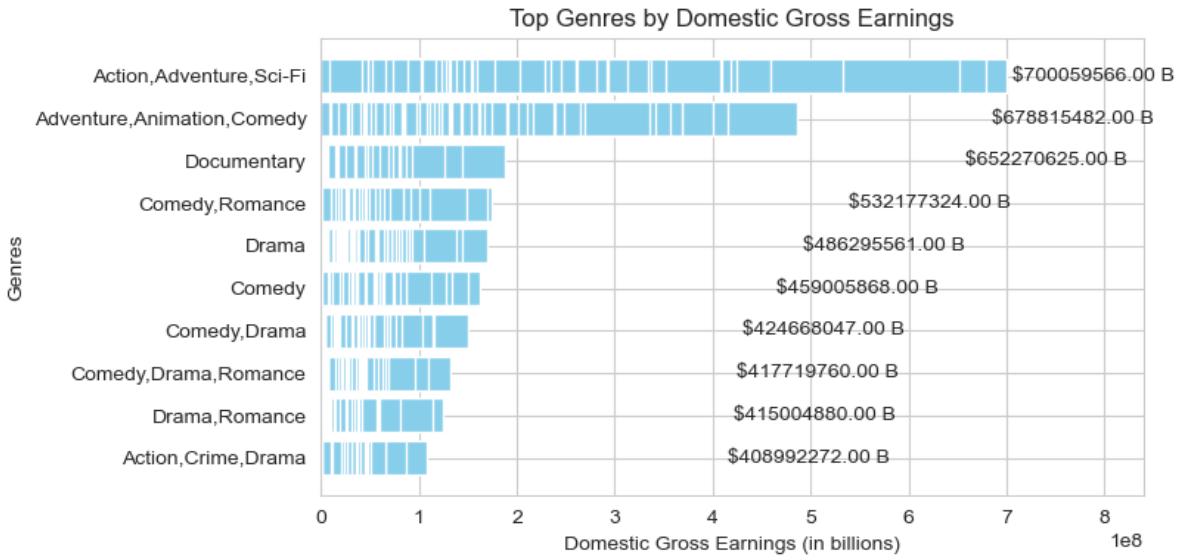
In [19]:
```
# Sort domestic gross earnings in descending
merged_data_df_sorted = merged_data_df.sort_values(by='domestic_gross_x', ascending

# Add limit
top_genres_df = merged_data_df_sorted['genres'].value_counts().head(10).index

# Filter to include only the top genres
top_genre_data_df = merged_data_df_sorted[merged_data_df_sorted['genres'].isin(top_
top_genre_data_df.head(10)
```

Out[19]:

	id	release_date	movie	production_budget	domestic_gross_x	worldwide_gross	tconst
31	42	Feb 16, 2018	Black Panther	200000000.0	700059566.0	1.348258e+09	tt1825683
2	7	Apr 27, 2018	Avengers: Infinity War	300000000.0	678815482.0	2.048134e+09	tt4154756
21	34	Jun 12, 2015	Jurassic World	215000000.0	652270625.0	1.648855e+09	tt0369610
33	45	Dec 16, 2016	Rogue One: A Star Wars Story	200000000.0	532177324.0	1.049103e+09	tt3748528
34	46	Jun 17, 2016	Finding Dory	200000000.0	486295561.0	1.021215e+09	tt2277860
1	4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	tt2395427
143	38	Nov 22, 2013	The Hunger Games: Catching Fire	130000000.0	424668047.0	8.648680e+08	tt1951264
76	13	Jun 22, 2018	Jurassic World: Fallen Kingdom	170000000.0	417719760.0	1.305773e+09	tt4881806
35	47	Jun 18, 2010	Toy Story 3	200000000.0	415004880.0	1.068880e+09	tt0435761
36	48	May 3, 2013	Iron Man 3	200000000.0	408992272.0	1.215392e+09	tt1300854

In [20]:

```
#4. genres vs domestic earnings
# Plotting genre vs domestic earnings
plt.figure(figsize=(8, 4))
plt.barh(top_genre_data_df['genres'], top_genre_data_df['domestic_gross_x'], color=
plt.xlabel('Domestic Gross Earnings (in billions)')
plt.ylabel('Genres')
plt.title('Top Genres by Domestic Gross Earnings')
plt.gca().invert_yaxis()  # Invert y-axis to display the genre with the highest ear

#maximum gross earnings for the top genres
max_earnings = top_genre_data_df.head(10)['domestic_gross_x'].max()

# set limit
plt.xlim(0, max_earnings * 1.2)

plt.tight_layout()

# Add labels to the bars displaying the actual values of 10 domestic gross earnings
for index, value in enumerate(top_genre_data_df.head(10)['domestic_gross_x']):
    plt.text(value, index, f' ${value:.2f} B', va='center')

plt.show()
```



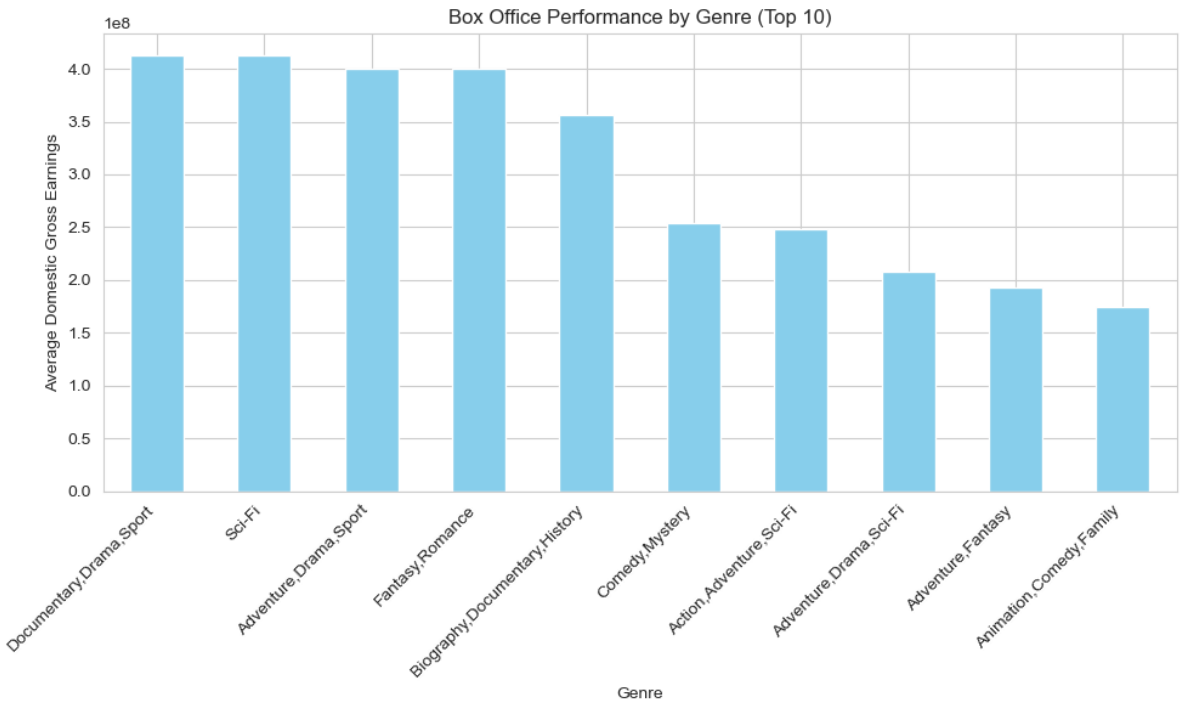
In [21]:
```
#Conclusion and Recommendations on Genres and Domestic Gross
recommendations = """
###
The domestic gross earnings for each genre are represented by the lengths of the ba
Longer bars indicate higher domestic gross earnings for movies in those genres.
Action,Adventure,Sci-fi movies have higher domestic grooss than comedy,Drama.
Audience prefence can influence financial performance in the domestic market.


Recommendations:
1. Prioritize genres with high domestic gross.
2. Allocate resourses in genres that are highly performing.
3. Encourage creativity and innovation to diffrenciate between movies categories.
3. Market research on evolving audience preferences,trends and patterns.
"""
print(recommendations)
```

```
###
The domestic gross earnings for each genre are represented by the lengths of the b
ars.
Longer bars indicate higher domestic gross earnings for movies in those genres.
Action,Adventure,Sci-fi movies have higher domestic grooss than comedy,Drama.
Audience prefence can influence financial performance in the domestic market.


Recommendations:
1. Prioritize genres with high domestic gross.
2. Allocate resourses in genres that are highly performing.
3. Encourage creativity and innovation to diffrenciate between movies categories.
3. Market research on evolving audience preferences,trends and patterns.
```

In [22]:
```
# 5.Box office performance by genre
genre_box_office = merged_data_df.groupby('genres')['domestic_gross_x'].mean().sort
genre_box_office.head(10)
```

Out[22]:
```
genres
Documentary,Drama,Sport          4.125634e+08
Sci-Fi                           4.125634e+08
Adventure,Drama,Sport            4.007380e+08
Fantasy,Romance                  4.007380e+08
Biography,Documentary,History    3.564617e+08
Comedy,Mystery                   2.544643e+08
Action,Adventure,Sci-Fi          2.480170e+08
Adventure,Drama,Sci-Fi           2.082258e+08
Adventure,Fantasy                1.928914e+08
Animation,Comedy,Family          1.750288e+08
Name: domestic_gross_x, dtype: float64
```

In [23]:
```
# Group the data by genre then calculate the mean domestic gross earnings for each
genre_box_office = merged_data_df.groupby('genres')['domestic_gross_x'].mean().sort

# Plotting
plt.figure(figsize=(10, 6))
genre_box_office.head(10).plot(kind='bar', color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Average Domestic Gross Earnings')
plt.title('Box Office Performance by Genre (Top 10)')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```

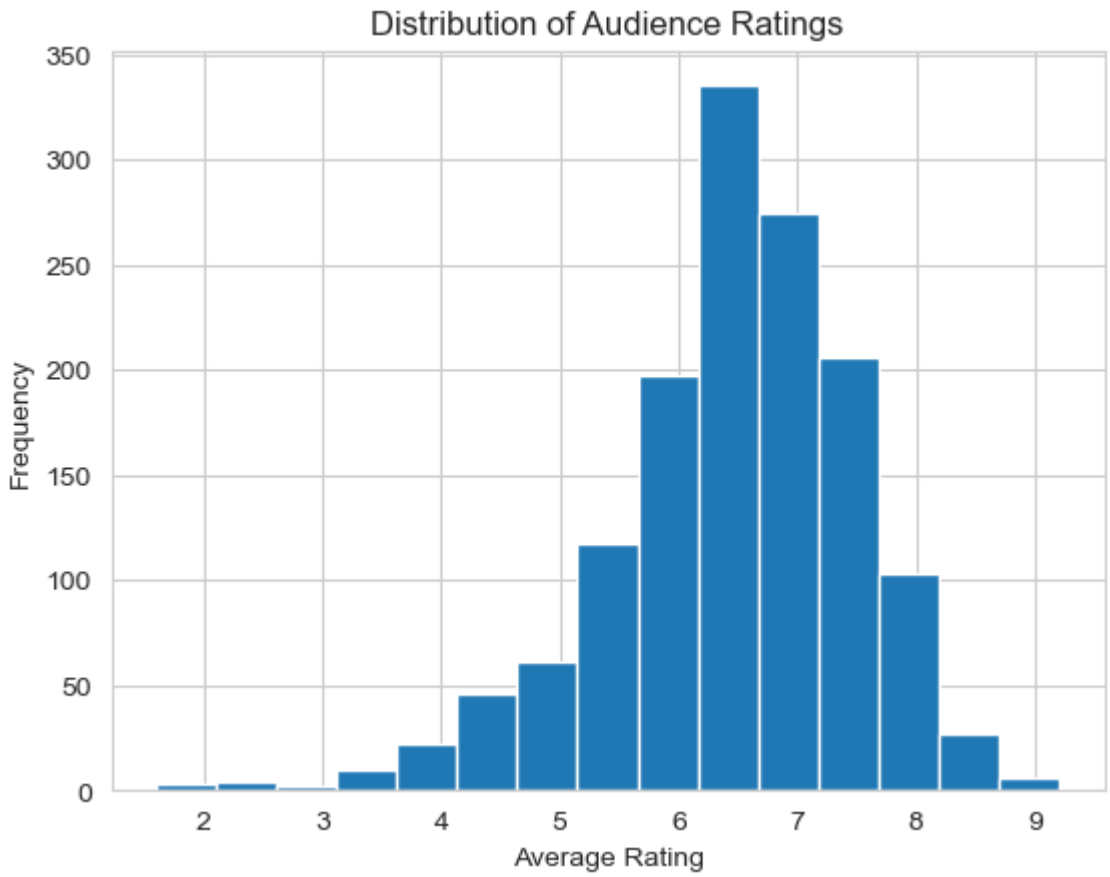


In [24]:
```
 #Conclusion and Recommendations on Box office performance by genre
recommendations = """
###
Certain genres generate higher value domestic gross earnings compared to others.
Genre preference has influence on audience ticket purchase.
Top performing genres can be used by producers to mximize box office revenue.
The popularity of the genre reflects on audience preferences.


Recommendations:
1. Maximize audience involvement in marketing campaigns.
2. Allocate resourses in genres that are highly performing.
3. Collaborate with industries to increase box office reavenue via fanbase and expe
3. Market research on evolving audience preferences,trends and patterns.
```

```
"""
print(recommendations)
```

```
###
Certain genres generate higher value domestic gross earnings compared to others.
Genre preference has influence on audience ticket purchase.
Top performing genres can be used by producers to mximize box office revenue.
The popularity of the genre reflects on audience preferences.


Recommendations:
1. Maximize audience involvement in marketing campaigns.
2. Allocate resourses in genres that are highly performing.
3. Collaborate with industries to increase box office reavenue via fanbase and exp
ertise.
3. Market research on evolving audience preferences,trends and patterns.
```

In [25]:
```
# 6 Distribution of audience ratings
plt.hist(merged_data_df['averagerating'], bins=15)
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.title('Distribution of Audience Ratings')
plt.show()
```



In [26]:
```
#identifying outliers in the above
# Calculate quartiles
Q1 = np.percentile(merged_data_df['averagerating'], 25)
Q3 = np.percentile(merged_data_df['averagerating'], 75)

# Calculate IQR
IQR = Q3 - Q1

# Calculate lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
# Identify outliers
outliers = merged_data_df[(merged_data_df['averagerating'] < lower_bound) | (merged

# Display outliers
outliers.head()
```

Out[26]:

	id	release_date	movie	production_budget	domestic_gross_x	worldwide_gross	tconst
115	83	Jul 1, 2010	The Last Airbender	150000000.0	131772187.0	319713881.0	tt0938283
227	46	Apr 15, 2011	Rio	90000000.0	143619809.0	487519809.0	tt573482(
259	11	Nov 11, 2011	Jack and Jill	79000000.0	74158157.0	150519217.0	tt0810913
360	77	Dec 25, 2014	Into the Woods	56200000.0	128002372.0	213116401.0	tt2201083
396	7	Jul 28, 2017	The Emoji Movie	50000000.0	86089513.0	216562312.0	tt4877122

In [27]:

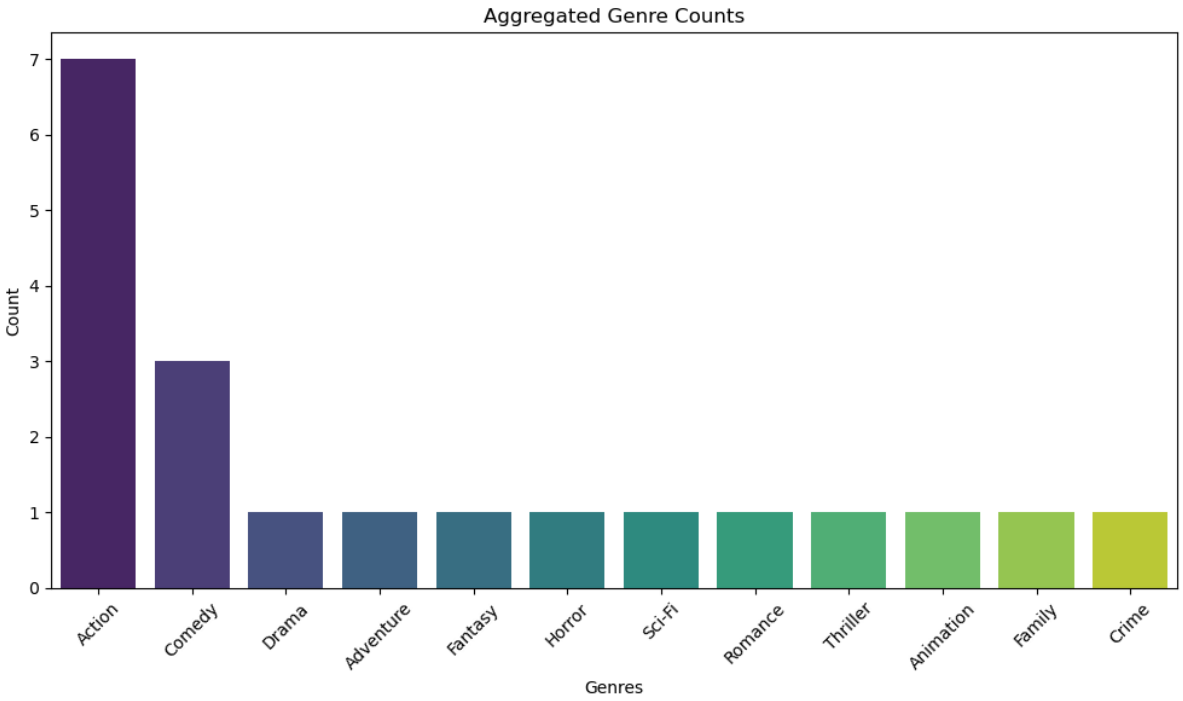
```
 #Conclusion and Recommendations on AverageRatings vs movies frequency
recommendations = """
###
Conclusions:
The histogram is symmetrical and ratings are evenly distributed at the center.
A peak at high rating indicates that a significant number of movies are well reciev
Apeak at the lower rating suggests lower audience satisifaction.
Movies with most frequency recieve average ratings.
Some movies have very low rating and others very high.
Outliers indicate that budget and ratings might affect movie production and recepti

Recommendations:
1. Continue producing movies that align with audience preference to maintain high r
2. Optimize runtime durations to align with audience preferences and improve the ov
3.Factors like production budget allocation should be optimized.
"""
print(recommendations)
```

```
###
Conclusions:
The histogram is symmetrical and ratings are evenly distributed at the center.
A peak at high rating indicates that a significant number of movies are well recie
ved by the audience.
Apeak at the lower rating suggests lower audience satisifaction.
Movies with most frequency recieve average ratings.
Some movies have very low rating and others very high.
Outliers indicate that budget and ratings might affect movie production and recept
ion.

Recommendations:
1. Continue producing movies that align with audience preference to maintain high
ratings.
2. Optimize runtime durations to align with audience preferences and improve the o
verall quality of movies.
3.Factors like production budget allocation should be optimized.
```

In [40]:

```
#7 genre vs counts
# Sample movie data
movie_data = pd.DataFrame({
    'movie_id': range(1, 21),
```

```
    'title': ['Movie' + str(i) for i in range(1, 21)],
    'genres': ['Action', 'Comedy', 'Drama', 'Action, Comedy', 'Action, Drama',
               'Comedy, Drama', 'Action, Adventure', 'Adventure, Fantasy', 'Fantasy
               'Action, Comedy, Drama', 'Horror', 'Sci-Fi', 'Comedy, Romance', 'Rom
               'Thriller', 'Action, Thriller', 'Animation, Family', 'Family, Fantas
               'Action, Adventure, Sci-Fi', 'Crime, Drama']
})

# Bar plot with aggregated counts
plt.figure(figsize=(10, 6))
movie_data['genres_agg'] = movie_data['genres'].apply(lambda x: x.split(',')[0])  #
genre_counts = movie_data['genres_agg'].value_counts()
sns.barplot(x=genre_counts.index, y=genre_counts.values, palette='viridis')
plt.title('Aggregated Genre Counts')
plt.xlabel('Genres')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

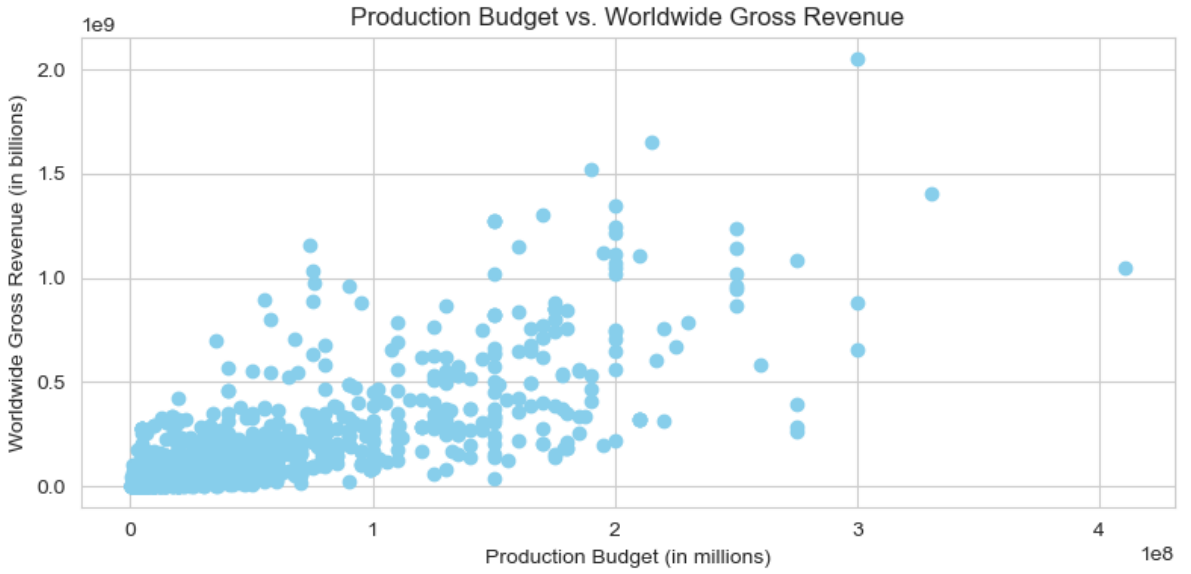


In [28]:

```
# Conclusion and Recommendations on Genres vs Count
recommendations = """
###
Conclusions:
Action movie has the highest count and comedy has a higher count than the other mov
Recommendations:
1. Focus on genres with the highest counts over different start years to attract mo
2. Consider the popularity of movie genres when planning production budgets and mar
3. Explore opportunities for diversifying movie genres to appeal to a wider audienc
"""
print(recommendations)
```

```
###
Conclusions:
Action movie has the highest count and comedy has a higher count than the other mo
vie genres.
Recommendations:
1. Focus on genres with the highest counts over different start years to attract m
ore viewers.
2. Consider the popularity of movie genres when planning production budgets and ma
rketing strategies.
3. Explore opportunities for diversifying movie genres to appeal to a wider audien
ce demographic.
```

In [29]:
```
#8. Correlation betweenworls widegross production budget

# Scatter plot
plt.figure(figsize=(8, 4))
plt.scatter(merged_data_df['production_budget'], merged_data_df['worldwide_gross'],
plt.xlabel('Production Budget (in millions)')
plt.ylabel('Worldwide Gross Revenue (in billions)')
plt.title('Production Budget vs. Worldwide Gross Revenue')
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [30]:
```
# Conclusion and Recommendations on worldwide grosss by production budget.
Conclusions = """
###
The scatter plot is clustered.
As the production budget increases the worldwide gross increases.
This is a possitive correlation.
"""
print(Conclusions)
```

```
###
The scatter plot is clustered.
As the production budget increases the worldwide gross increases.
This is a possitive correlation.
```

In [31]:
```
#correlation between ratings and box office gross

rating_correlation = merged_data_df['averagerating'].corr(merged_data_df['domestic_
print("Correlation between ratings and box office gross:", rating_correlation)
```

```
Correlation between ratings and box office gross: 0.19520412734142323
```

In [32]:
```
# Conclusion and Recommendations on ratings and box office gross
Conclusions = """
###
This is a negative correlation.
Showing you cannot draw  meaningful insights using ratings by box office gross

"""
print(Conclusions)
```

```
###
This is a negative correlation.
Showing you cannot draw  meaningful insights using ratings by box office gross
```

In [33]:
```
Overall_Recommendation= """
CONCLUSIONS
1.Movies with higher production budgets have higher box office earning.
2.Investing more in production leads to higher box office revenues.
3.Frequency distribution of genres reveal that certain movie genre perfom better th
4.Ratings are clustered around a certain range.
5.Rating shows audience preferences and genre popularity.
6.Box office performance fluctuats overtime hence market research should be done.

RECOMMENDATIONS
1.Focus on producing movies that have demonstrated strong perfomance in box office.
2.Allocate resources strategically by investing in high potential movie projects wi
3.Monitor market trends and audience preferrence to optimize release timings.
4.Pay attention to Factors beyound audience ratings.


"""
print(Overall_Recommendation)
```

```
CONCLUSIONS
1.Movies with higher production budgets have higher box office earning.
2.Investing more in production leads to higher box office revenues.
3.Frequency distribution of genres reveal that certain movie genre perfom better t
han others in terms of domestic earnings.
4.Ratings are clustered around a certain range.
5.Rating shows audience preferences and genre popularity.
6.Box office performance fluctuats overtime hence market research should be done.

RECOMMENDATIONS
1.Focus on producing movies that have demonstrated strong perfomance in box offic
e.
2.Allocate resources strategically by investing in high potential movie projects w
ith higher production budget.
3.Monitor market trends and audience preferrence to optimize release timings.
4.Pay attention to Factors beyound audience ratings.
```