Movie Studio Box Office Analysis: Comprehensive Exploratory Data Analysis

Domestic Box Office For 2025 and 2024 from scraped https://www.boxofficemojo.com/

In [4]:

```
#import all the neccessary libraries
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
import itertools
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re


# Suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

In [1]:

```
#Webscrapping
def extract_movie_details(url):
    # Send a request to the URL
    response = requests.get(url)

    # Parse the HTML content
    soup = BeautifulSoup(response.text, "html.parser")

    # Find the summary details section
    summary_details = soup.find('div', class_='mojo-summary-values')

    if not summary_details:
        return "Could not find summary details section"

    # Extract specific details
    details = {}

    # MPAA Rating
    mpaa_elem = summary_details.find('span', string='MPAA')
    if mpaa_elem:
        details['MPAA'] = mpaa_elem.find_next_sibling('span').text.strip()

    # Running Time
    runtime_elem = summary_details.find('span', string='Running Time')
    if runtime_elem:
        details['Running Time'] = runtime_elem.find_next_sibling('span').te

    # Genres
    genres_elem = summary_details.find('span', string='Genres')
    if genres_elem:
        genres = genres_elem.find_next_sibling('span').text.strip()
        details['Genres'] = [genre.strip() for genre in genres.split('\n') :

    # In Release
    release_elem = summary_details.find('span', string='In Release')
```

```
  if release_elem:
    details['In Release'] = release_elem.find_next_sibling('span').text

  return details

def scrape_box_office_data(years):
 data = []
 for year in years:
  url = f'https://www.boxofficemojo.com/year/{year}/?grossesOption=totalG
  response = requests.get(url)
  soup = BeautifulSoup(response.text, "html.parser")
  # Find the table rows
  rows = soup.find_all('tr')[1:]  # Skip the header rows

  # Prepare lists to store data


  # Extract data from each row
  for row in rows:
    cols = row.find_all('td')

    # Check if row has enough columns
    if len(cols) >= 12:
      # Find the movie link
      movie_link = cols[1].find('a', class_='a-link-normal')

      # Base movie data
      movie_data = {
        'Rank': cols[0].text.strip(),
        'Year': year,
        'Movie': movie_link.text.strip() if movie_link else 'N/A',
        'Movie Link': "https://www.boxofficemojo.com" + movie_link[
        'Total Gross': cols[5].text.strip().replace('$', '').replac
        'Max Theaters': cols[6].text.strip(),
        'Opening Weekend Gross': cols[7].text.strip().replace('$',
        'Opening Weekend % of Total': cols[8].text.strip(),
        'Opening Theaters': cols[9].text.strip(),
        'Open Date': cols[10].text.strip(),
        'Distributor': cols[12].text.strip()
      }

      # Get additional movie details
      if movie_link:
        additional_details = extract_movie_details("https://www.box
        movie_data.update(additional_details)

      data.append(movie_data)

  # Create DataFrame
 df = pd.DataFrame(data)
 # Display the DataFrame
 return df

# Example usage
years = [2024, 2025]
box_office_data = scrape_box_office_data(years)

# Display the DataFrame
box_office_data.head(10)
```

Out[1]:

	Rank	Year	Movie	Movie Link	Total Gross	I Thea
0	1	2024	Inside Out 2	https://www.boxofficemojo.com/release/rl363819...	652980194	4,
1	2	2024	Deadpool & Wolverine	https://www.boxofficemojo.com/release/rl410809...	636745858	4,
2	3	2024	Wicked	https://www.boxofficemojo.com/release/rl119947...	473231120	3,
3	4	2024	Moana 2	https://www.boxofficemojo.com/release/rl862748...	460364069	4,
4	5	2024	Despicable Me 4	https://www.boxofficemojo.com/release/rl260351...	361004205	4,
5	6	2024	Beetlejuice Beetlejuice	https://www.boxofficemojo.com/release/rl336511...	294100435	4,
6	7	2024	Dune: Part Two	https://www.boxofficemojo.com/release/rl687152...	282144358	4,
7	8	2024	Twisters	https://www.boxofficemojo.com/release/rl132471...	267762265	4
8	9	2024	Mufasa: The Lion King	https://www.boxofficemojo.com/release/rl151109...	253981541	4,
9	10	2024	Sonic the Hedgehog 3	https://www.boxofficemojo.com/release/rl886211...	236100420	3,

In [5]:

```
# Assuming box_office_data is already defined
df = box_office_data.copy()
```

Data Preparation and Initial Exploration

In [6]:

```
# Data Cleaning Functions
def clean_currency(x):
    """
    Convert currency string to float by removing commas
    Handle cases with '-' or empty strings
    """
    if pd.isna(x) or x == '-':
        return 0.0
    return float(str(x).replace(',', ''))

def clean_theaters(x):
    """
    Convert theater count string to integer by removing commas
    Handle cases with '-' or empty strings
    """
    if pd.isna(x) or x == '-':
        return 0
    return int(str(x).replace(',', ''))
def clean_genres(genres):
    """
    Clean and standardize genre entries
    """
    # Handle different possible input types
    if isinstance(genres, str):
        # Remove brackets, quotes, and split
        return [genre.strip().strip("'") for genre in genres.strip('[]').spl
    elif isinstance(genres, list):
        # Clean list entries
        return [genre.strip().strip("'") for genre in genres]
    else:
        # If not a string or list, return empty list
        return []

# Convert Running Time from format "1 hr 36 min" to total minutes
def convert_runtime(rt):
    if isinstance(rt, str):
        hrs = re.search(r'(\d+)\s*hr', rt)
        mins = re.search(r'(\d+)\s*min', rt)
        total = 0
        if hrs:
            total += int(hrs.group(1)) * 60
        if mins:
            total += int(mins.group(1))
        return total
    return np.nan
# Load Data
def load_and_prepare_data(filepath):
    """
    Load box office data and prepare it for analysis
    """
    # Clean numerical columns
    df['Total Gross'] = df['Total Gross'].apply(clean_currency)
    df['Opening Weekend Gross'] = df['Opening Weekend Gross'].apply(clean_cu
    df['Max Theaters'] = df['Max Theaters'].apply(clean_theaters)
    df['Opening Theaters'] = df['Opening Theaters'].apply(clean_theaters).as

    # Split genres
    df['Genres'] = df['Genres'].apply(clean_genres)

    # Extract numeric running time
    df['Running Time Numeric'] = df["Running Time"].apply(convert_runtime)

    # Clean percentage column: remove '%' and convert to float
    df["Opening Weekend % of Total"] =df["Opening Weekend % of Total"].apply
        lambda x: np.nan if (pd.isna(x) or x == '-' or str(x).strip() == ''
        else float(str(x).replace('%',''))/100
    )
    return df

# Load the dataset
df1 = load_and_prepare_data(df)

# Save to CSV
df1.to_csv('box_office_data_cleaned.csv', index=False)
df1.head(5)
```

Out[6]:

	Rank	Year	Movie	Movie Link	Total Gross	The
0	1	2024	Inside Out 2	https://www.boxofficemojo.com/release/rl363819...	652980194.0	
1	2	2024	Deadpool & Wolverine	https://www.boxofficemojo.com/release/rl410809...	636745858.0	
2	3	2024	Wicked	https://www.boxofficemojo.com/release/rl119947...	473231120.0	
3	4	2024	Moana 2	https://www.boxofficemojo.com/release/rl862748...	460364069.0	
4	5	2024	Despicable Me 4	https://www.boxofficemojo.com/release/rl260351...	361004205.0	

In [7]:

```
print(df1.isnull().sum())
```

```
Rank                           0
Year                           0
Movie                          0
Movie Link                     0
Total Gross                    0
Max Theaters                   0
Opening Weekend Gross          0
Opening Weekend % of Total    28
Opening Theaters               0
Open Date                      0
Distributor                    0
MPAA                          81
Running Time                   5
Genres                         0
In Release                     0
Running Time Numeric           5
dtype: int64
```

In [8]:

```
def handle_missing_data(df):
    # Percentage data
    df['Opening Weekend % of Total'] = df['Opening Weekend % of Total'].fil
        df['Opening Weekend % of Total'].median()
    )

    # MPAA ratings
    df['MPAA'] = df['MPAA'].fillna('Unknown')

    # Running time
    runtime_median = df['Running Time Numeric'].median()
    df['Running Time Numeric'] = df['Running Time Numeric'].fillna(runtime_m
    df['Running Time'] = df['Running Time'].fillna(f"{runtime_median//60} h

    return df

df_clean = handle_missing_data(df1.copy())
```

Comprehensive Visual Analysis

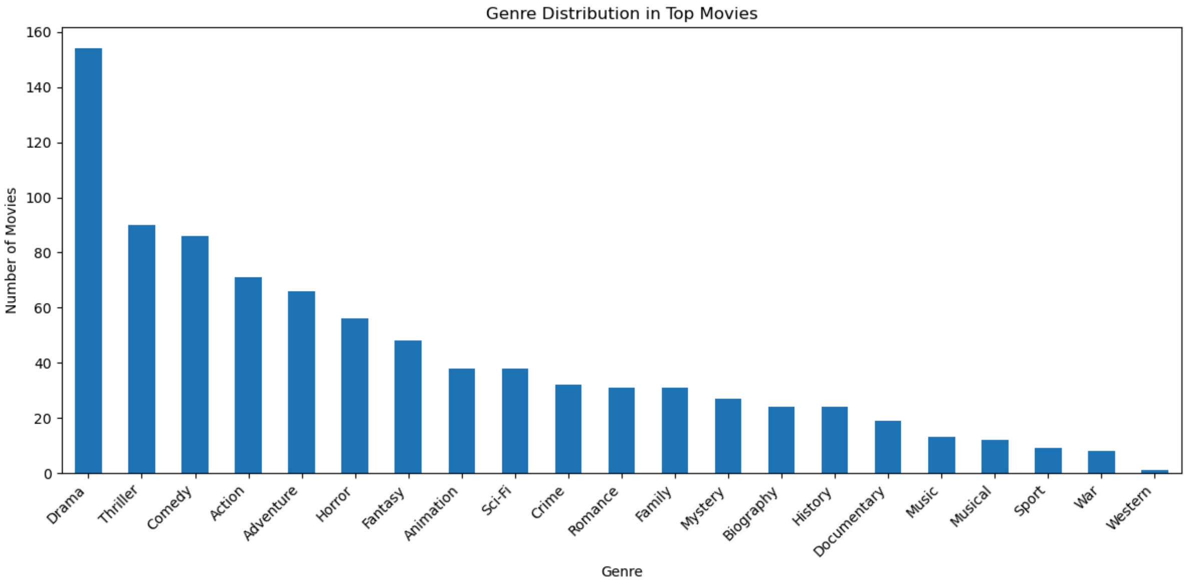
In [9]:

```
# Explode genres for individual analysis
df_exploded = df_clean.explode('Genres')

# Filter relevant columns
analysis_df = df_exploded[['Total Gross', 'Genres', 'MPAA', 'Running Time Nu
```

1. Genre Frequency Analysis

In [10]:

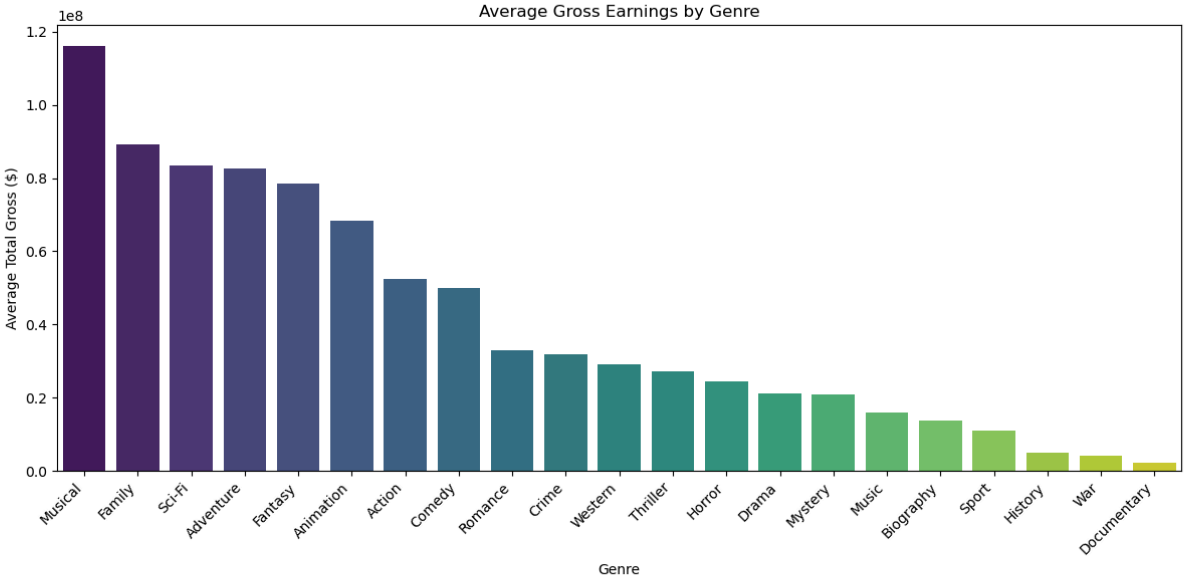
```
# Genre Frequency Analysis
genre_counts = df_exploded['Genres'].value_counts()

plt.figure(figsize=(12, 6))
genre_counts.plot(kind='bar')
plt.title('Genre Distribution in Top Movies')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



2. Genre Performance **Analysis**

In [11]:

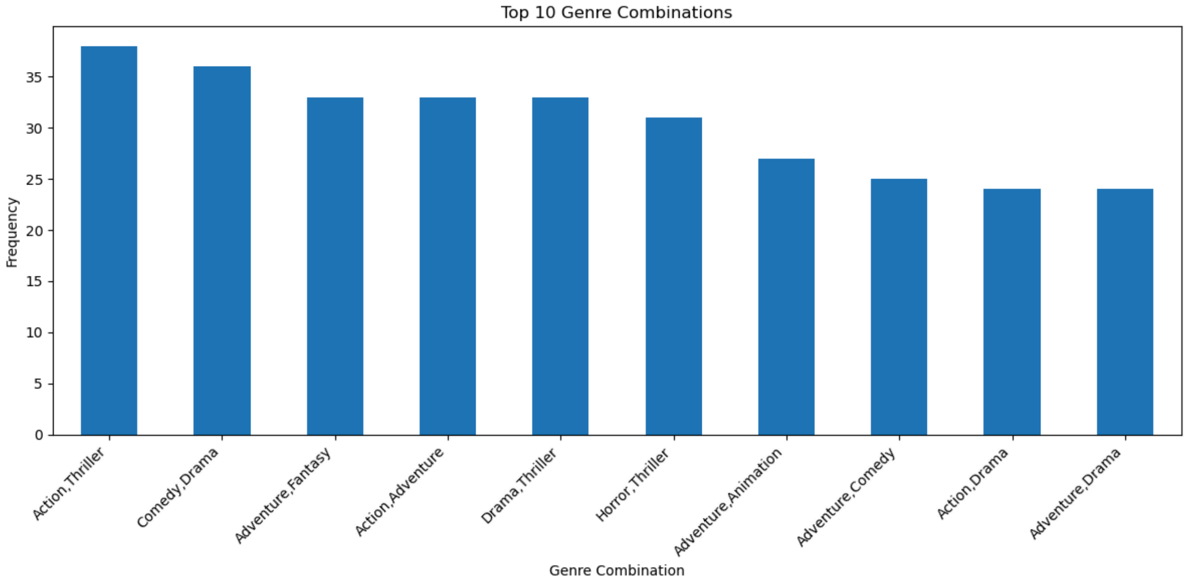
```
# Genre Performance Analysis
genre_gross_analysis = df_exploded.groupby('Genres')['Total Gross'].agg([
    'mean',      # Average gross
    'count',     # Number of movies
    'sum'        # Total gross
]).sort_values('mean', ascending=False)

# Define a color palette
palette = sns.color_palette("viridis", len(genre_gross_analysis))

plt.figure(figsize=(12, 6))
sns.barplot(
    x=genre_gross_analysis.index,
    y=genre_gross_analysis['mean'],
    palette=palette
)
plt.title('Average Gross Earnings by Genre')
plt.xlabel('Genre')
plt.ylabel('Average Total Gross ($)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



3. Genre Combination Analysis

In [12]:

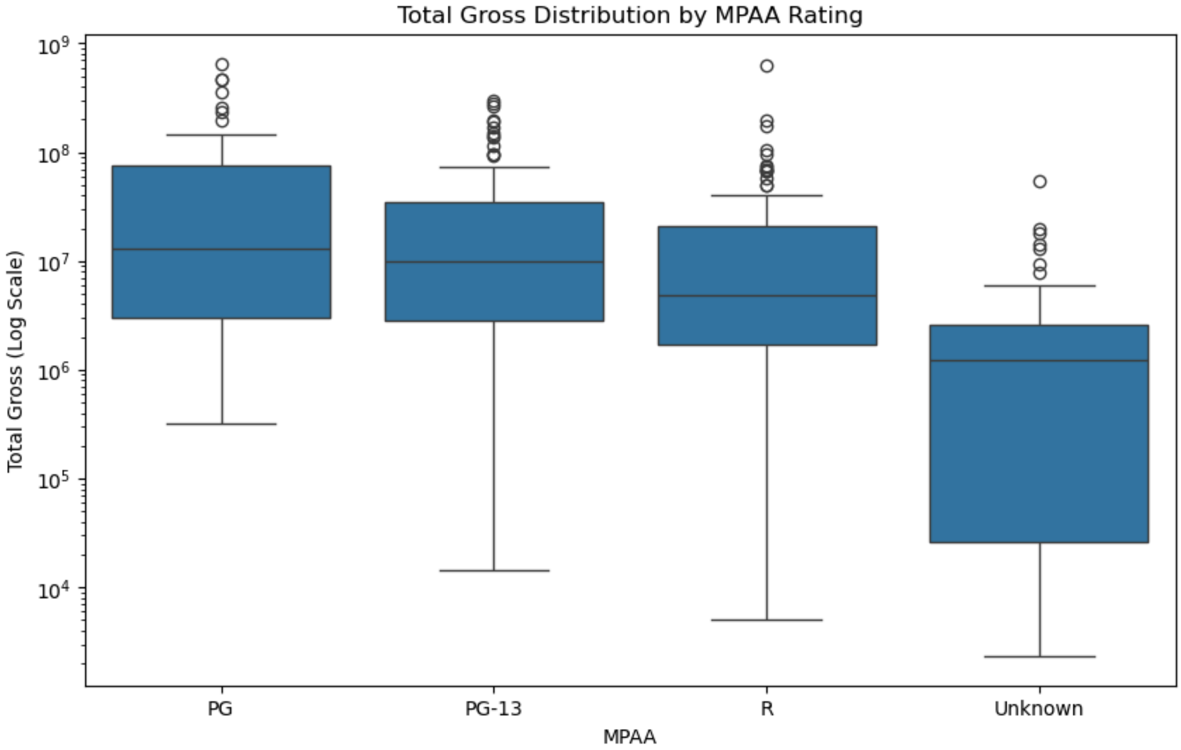
```
# Genre Combination Analysis
def get_genre_combinations(genres_list):
    return [','.join(sorted(combo)) for r in range(2, len(genres_list)+1)
            for combo in itertools.combinations(genres_list, r)]

genre_combinations = df_clean['Genres'].apply(get_genre_combinations)
genre_combo_exploded = pd.DataFrame({'Genre Combinations': [combo for sublis
genre_combo_counts = genre_combo_exploded['Genre Combinations'].value_count

plt.figure(figsize=(12, 6))
genre_combo_counts.plot(kind='bar')
plt.title('Top 10 Genre Combinations')
plt.xlabel('Genre Combination')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



4. MPAA Rating Impact

In [13]:

```
plt.figure(figsize=(10,6))
sns.boxplot(data=df_clean, x='MPAA', y='Total Gross', order=['PG', 'PG-13',
plt.yscale('log')
plt.title('Total Gross Distribution by MPAA Rating')
plt.ylabel('Total Gross (Log Scale)')
plt.show()
```



5. MPAA Rating Detailed Analysis

In [14]:

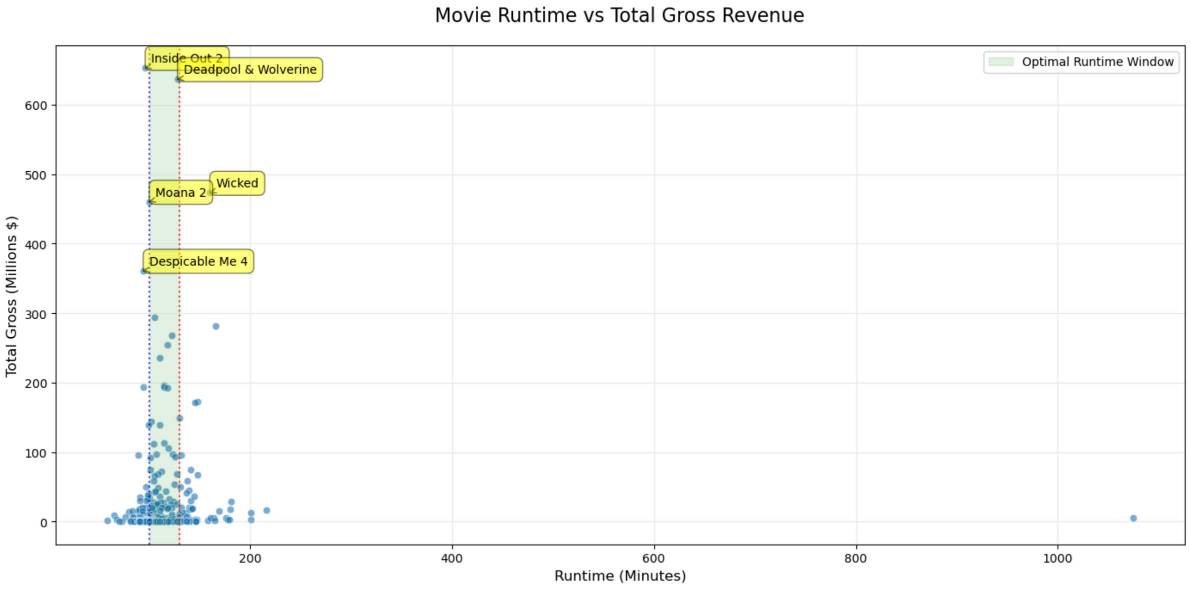
```
# MPAA Rating Detailed Analysis
mpaa_performance = df_clean.groupby('MPAA')['Total Gross'].agg([
    'mean', 'median', 'count'
]).sort_values('mean', ascending=False)
print("\nMPAA Rating Performance:")
print(mpaa_performance)
```

```
MPAA Rating Performance:
                 mean      median  count
MPAA
PG       8.157146e+07  13029994.0     43
PG-13    4.258695e+07   9891552.5     68
R        2.246168e+07   4789743.0    121
Unknown  2.842226e+06   1223881.0     81
G        2.378444e+06   2378444.0      2
```

6. Runtime Analysis

In [15]:

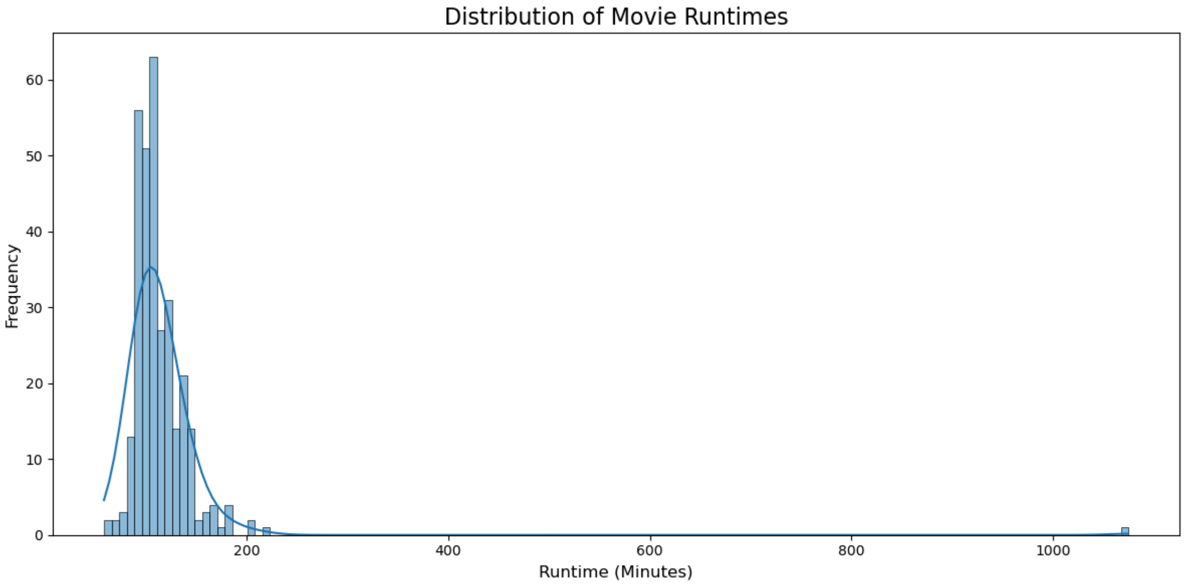
```python
plt.figure(figsize=(14, 7))
scatter = plt.scatter(df_clean['Running Time Numeric'],
                      df_clean['Total Gross']/1e6,
                      alpha=0.6,
                      edgecolors='w',
                      linewidth=0.5)

plt.title('Movie Runtime vs Total Gross Revenue', fontsize=16, pad=20)
plt.xlabel('Runtime (Minutes)', fontsize=12)
plt.ylabel('Total Gross (Millions $)', fontsize=12)

# Add optimal runtime range
plt.axvspan(100, 130, color='green', alpha=0.1, label='Optimal Runtime Windo
plt.axvline(x=100, color='blue', linestyle=':', alpha=0.7)
plt.axvline(x=130, color='red', linestyle=':', alpha=0.7)

# Label select points (top performers)
top_movies = df_clean.nlargest(5, 'Total Gross')
for i, row in top_movies.iterrows():
    plt.annotate(row['Movie'],
                 xy=(row['Running Time Numeric'], row['Total Gross']/1e6),
                 xytext=(5, 5), textcoords='offset points',
                 bbox=dict(boxstyle='round,pad=0.5', fc='yellow', alpha=0.5),
                 arrowprops=dict(arrowstyle='->'))

plt.legend()
plt.grid(alpha=0.2)
plt.tight_layout()
plt.show()
```



7.Runtime Distribution

In [16]:

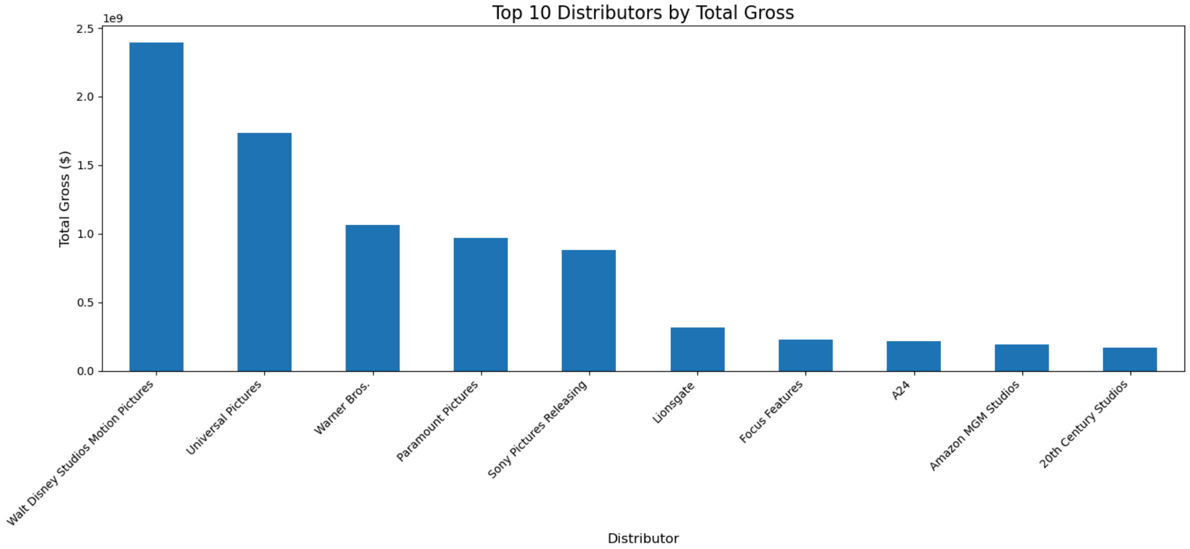
```
# Runtime Distribution
plt.figure(figsize=(12, 6))
sns.histplot(df_clean['Running Time Numeric'], kde=True)
plt.title('Distribution of Movie Runtimes', fontsize=16)
plt.xlabel('Runtime (Minutes)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.tight_layout()
plt.show()
```



8. Top Distributors by Total Gross

In [17]:

```
# Top Distributors by Total Gross
distributor_performance = df_clean.groupby('Distributor')['Total Gross'].ag
    'mean', 'sum', 'count'
]).sort_values('sum', ascending=False).head(10)

plt.figure(figsize=(15, 7))
distributor_performance['sum'].plot(kind='bar')
plt.title('Top 10 Distributors by Total Gross', fontsize=16)
plt.xlabel('Distributor', fontsize=12)
plt.ylabel('Total Gross ($)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

print("\nTop Distributor Performance:")
print(distributor_performance)
```

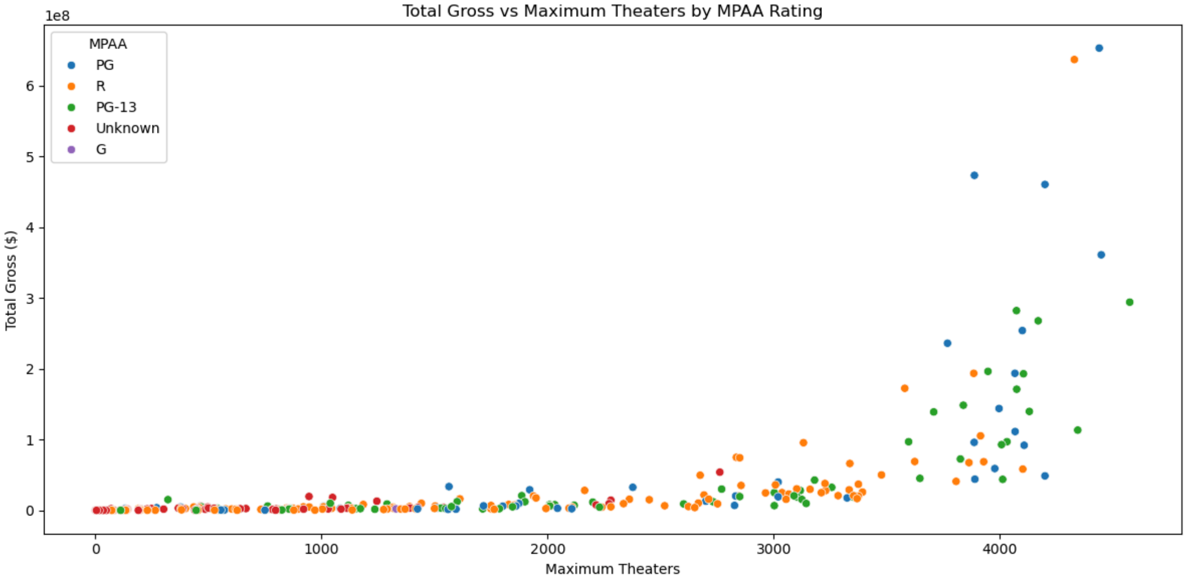


```
Top Distributor Performance:
                                             mean           sum  count
Distributor
Walt Disney Studios Motion Pictures  1.843241e+08  2.396213e+09     13
Universal Pictures                   1.334153e+08  1.734399e+09     13
Warner Bros.                         8.192627e+07  1.065042e+09     13
Paramount Pictures                   1.079831e+08  9.718478e+08      9
Sony Pictures Releasing              4.398833e+07  8.797666e+08     20
Lionsgate                            1.862698e+07  3.166587e+08     17
Focus Features                       2.040290e+07  2.244319e+08     11
A24                                  1.327958e+07  2.124734e+08     16
Amazon MGM Studios                   2.365588e+07  1.892470e+08      8
20th Century Studios                 1.711302e+08  1.711302e+08      1
```

9. Total Gross vs Maximum Theaters by MPAA Rating

In [18]:

```
#Total Gross vs Maximum Theaters by MPAA Rating
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df_clean, x='Max Theaters', y='Total Gross', hue='MPAA
plt.title('Total Gross vs Maximum Theaters by MPAA Rating')
plt.xlabel('Maximum Theaters')
plt.ylabel('Total Gross ($)')
plt.tight_layout()
plt.show()
```



Key Insights and Recommendations

Genre Strategy

The genre distribution analysis of Domestic Box Office rankings for 2024 and 2025 reveals that Drama, Thriller, and Comedy dominate in terms of movie count. However, when examining Median Total Gross (USD), the highest-earning genres were Musical, Family, and Sci-Fi, indicating that while some genres are more common, they do not necessarily generate the highest revenue. Additionally, an analysis of Genre Combinations highlights that Action and Thriller frequently appear together, suggesting that hybrid genres, particularly those blending action elements, remain popular. This insight can inform strategic decisions on film production and marketing to balance commercial success with audience preferences.

MPAA Rating Considerations

The analysis of MPAA ratings indicates that PG and PG-13 rated movies tend to generate higher and more consistent gross earnings, making them ideal for targeting a broader audience. The data also reveals that these ratings exhibit more frequent outliers, suggesting variability in performance but also the potential for significant box office success. Specifically, PG-rated films have the highest average gross $81.57M, followedbyPG-13films$42.59M. In contrast, R-rated movies show lower earnings with a median gross of $4.79M, while G-rated and "Unknown" rated movies have the lowest financial performance. These findings highlight the strategic advantage of focusing on PG and PG-13 ratings to maximize both audience reach and box office returns.

Runtime Optimization

The analysis of movie runtime suggests that the optimal duration for maximizing audience engagement and box office performance falls between 100 to 130 minutes. Films within this range are more likely to balance storytelling depth and viewer retention. Additionally, the distribution of movie runtimes indicates that the majority of films are between 70 and 200 minutes, with extreme runtimes (either too short or too long) being less common. To optimize viewership and financial success, it is advisable to avoid excessively short or long films, as they may not align with audience expectations or industry standards.

Distribution Strategy

The analysis highlights the importance of partnering with top-performing distributors to maximize box office success. Walt Disney Studios and Universal Pictures emerge as the leading distributors, consistently delivering high-grossing films. Their strong performance suggests that collaborating with established and well-reputed distribution companies can significantly impact a movie's financial success. By aligning with these industry giants, filmmakers can enhance reach, marketing effectiveness, and overall audience engagement.

Theaters and Revenue Relationship

The scatter plot "Total Gross vs Maximum Theaters by MPAA Rating" reveals a clear positive correlation between the number of theaters a movie is released in and its total gross revenue. Movies that secure wider theatrical releases tend to generate higher earnings, emphasizing the importance of maximizing theater count for financial success. This trend suggests that expanding distribution reach can significantly enhance a film’s box office performance, making strategic theater placement a crucial factor in revenue optimization.