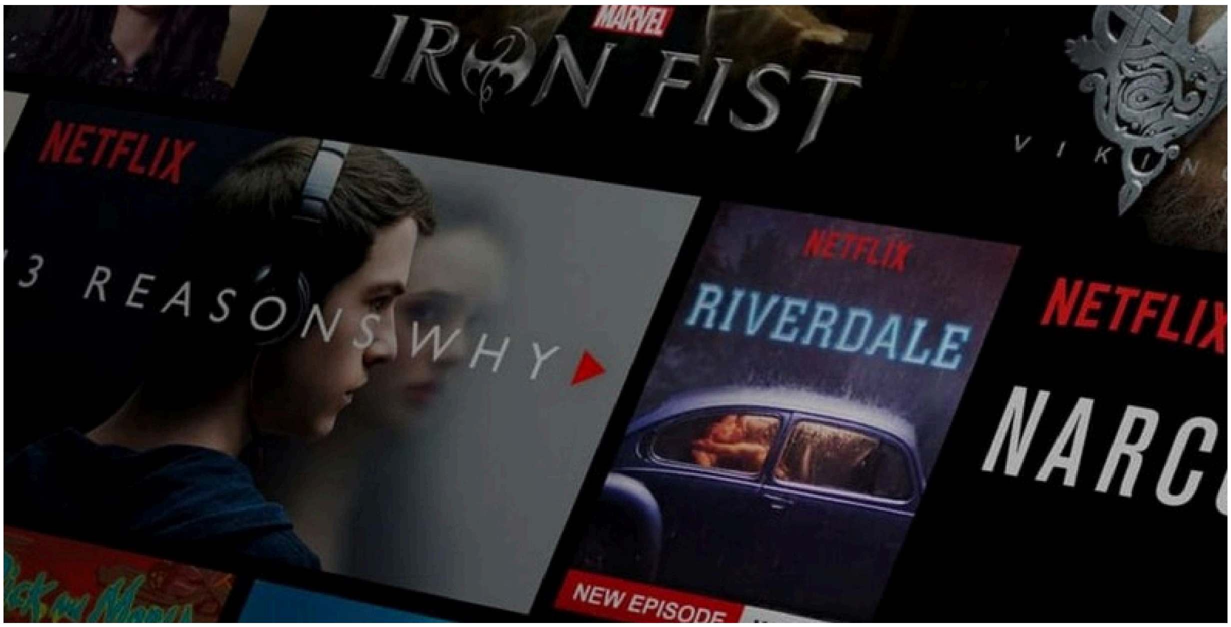
Final Project Submission

Please fill out:

- GROUP 6
- Student names: ANTONY WALA, DOREEN WATHIMU, ELSIE WAIRIMU, KENNEDY WAMWATI
- Student pace: PART-TIME
- Instructor name: CHRISTINE KIRIMI



Movie Studio Analysis: Driving Success at the Box Office

Business Problem

The landscape of original video content is booming, and our company is launching a new movie studio to be a part of it. As a new player, we face a critical challenge: **what types of films should we create to maximize our chances of box office success?**

This analysis tackles that question by examining data from thousands of films to identify key trends in genres, ratings, runtime, and box office performance. The goal is to translate these findings into a clear, actionable strategy for the head of our new studio.

Key Questions:

1. **Which genres** generate the highest box office revenue?
2. Do **higher ratings** or **specific runtimes** correlate with better financial performance?

Import Libraries

In [1]:
```
# Import necessary libraries

import pandas as pd  # For data manipulation and analysis
import sqlite3       # For connecting to and querying the SQLite database
import matplotlib.pyplot as plt  # For creating visualizations
import seaborn as sns            # For enhanced visualizations (e.g., scatter plo
from scipy import stats          # For hypothesis testing (e.g., t-tests)
from sklearn.linear_model import LinearRegression  # For linear regression model
from sklearn.model_selection import train_test_split  # For splitting data in reg
from sklearn.metrics import mean_squared_error, r2_score  # For evaluating regres
import numpy as np   # For numerical operations
import zipfile # For unzipping zipped files

# Set visualization style for clarity
sns.set(style="whitegrid")
```

Data Understanding

To answer our business questions, we will use two primary datasets:

- **IMDb Data (`im.db`):** A comprehensive SQLite database containing detailed information on movie titles, release years, genres, runtimes, and audience ratings.
- **Box Office Mojo Data (`bom.movie_gross.csv.gz`):** A financial dataset in CSV format with domestic and foreign box office gross for a wide range of films.

We will begin by loading and inspecting each dataset to understand its structure, identify potential data quality issues, and check for missing values.

In [2]:
```
# Connect to the SQLite database

# Specify the path to the zipped folder containing the database
zip_path = 'zippedData/im.db.zip'

# Specify the destination path for the unzipped database file
extract_path = 'zippedData/'

# Unzip the folder
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)  # Extract all contents to the specified pat

# Verify the extraction by checking if the im.db file exists
print(f"Unzipped contents to: {extract_path}")

# Adjust the path if your im.db is in a different location after unzipping
conn = sqlite3.connect('zippedData/im.db')  # Establish connection to the databas

# Query to get a sample from movie_basics table
basics_query = """
SELECT *
FROM movie_basics
LIMIT 5;
"""
basics_sample = pd.read_sql(basics_query, conn)  # Use pandas to run SQL and load

# Query to get a sample from movie_ratings table
ratings_query = """
SELECT *
FROM movie_ratings
LIMIT 5;
"""
ratings_sample = pd.read_sql(ratings_query, conn)  # Load ratings sample

# Display samples
print("Movie Basics Sample:")
display(basics_sample)  # Use display for nicer output in Jupyter

print("\nMovie Ratings Sample:")
display(ratings_sample)

# Get summary info for basics (e.g., data types, non-null counts)
print("\nMovie Basics Info:")
basics_sample.info()

# Close the connection (good practice, but we'll reopen if needed)
conn.close()
```

```
Unzipped contents to: zippedData/
Movie Basics Sample:
```

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

```
Movie Ratings Sample:
```

	movie_id	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

```
Movie Basics Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   movie_id         5 non-null      object 
 1   primary_title    5 non-null      object 
 2   original_title   5 non-null      object 
 3   start_year       5 non-null      int64  
 4   runtime_minutes  4 non-null      float64
 5   genres           5 non-null      object 
dtypes: float64(1), int64(1), object(4)
memory usage: 368.0+ bytes
```

Load and Explore Box Office Mojo Data

In [3]:
```
# Load the compressed CSV (no need to unzip)
bom_path = 'zippedData/bom.movie_gross.csv.gz'
bom_df = pd.read_csv(bom_path)  # Read directly into pandas DataFrame

# Display sample
print("Box Office Mojo Sample:")
display(bom_df.head())  # Show first 5 rows

# Get summary statistics
print("\nBox Office Mojo Summary:")
display(bom_df.describe(include='all'))  # Include all columns for overview

# Check for missing values
print("\nMissing Values in Box Office Mojo:")
print(bom_df.isnull().sum())
```

```
Box Office Mojo Sample:
```

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010

```
Box Office Mojo Summary:
```

	title	studio	domestic_gross	foreign_gross	year
count	3387	3382	3.359000e+03	2037	3387.000000
unique	3386	257	NaN	1204	NaN
top	Bluebeard	IFC	NaN	1200000	NaN
freq	2	166	NaN	23	NaN
mean	NaN	NaN	2.874585e+07	NaN	2013.958075
std	NaN	NaN	6.698250e+07	NaN	2.478141
min	NaN	NaN	1.000000e+02	NaN	2010.000000
25%	NaN	NaN	1.200000e+05	NaN	2012.000000
50%	NaN	NaN	1.400000e+06	NaN	2014.000000
75%	NaN	NaN	2.790000e+07	NaN	2016.000000
max	NaN	NaN	9.367000e+08	NaN	2018.000000

```
Missing Values in Box Office Mojo:
title               0
studio              5
domestic_gross     28
foreign_gross    1350
year                0
dtype: int64
```

Data Preparation

To create a robust dataset for analysis, we need to clean and merge the information from IMDb and Box Office Mojo.

Key Preparation Steps:

1. **Clean IMDb Data:**
   - Join the `movie_basics` and `movie_ratings` tables using SQL.
   - Focus on recent films (2010 onwards) to ensure our insights are timely.
   - Handle missing values for runtime and genres.
   - **Transform the `genres` column**, splitting comma-separated lists (e.g., "Action,Adventure,Sci-Fi") into individual rows. This is a critical step that allows us to analyze each genre's performance independently.
2. **Clean Box Office Mojo Data:**
   - Convert the financial columns (`domestic_gross` , `foreign_gross`) to numeric types.
   - Create a `total_gross` column to represent worldwide box office revenue.
3. **Merge Datasets:**
   - Combine the cleaned IMDb and Box Office Mojo data into a single, unified DataFrame.
   - To account for minor discrepancies in release year reporting, the merge is performed on the movie's primary title and allows for a ±1 year window.

Clean and Merge IMDb Data

In [4]:
```
# Reconnect to SQLite database
conn = sqlite3.connect('zippedData/im.db')

# Join movie_basics and movie_ratings using SQL
query = """
SELECT b.movie_id, b.primary_title, b.start_year, b.runtime_minutes, b.genres,
       r.averagerating, r.numvotes
FROM movie_basics b
LEFT JOIN movie_ratings r ON b.movie_id = r.movie_id
WHERE b.start_year >= 2010  -- Focus on recent movies
"""
imdb_df = pd.read_sql(query, conn)

# Close connection
conn.close()

# Display sample of joined data
print("IMDb Joined Data Sample:")
display(imdb_df.head())

# Check missing values
print("\nMissing Values in IMDb Data:")
print(imdb_df.isnull().sum())

# Handle missing runtime_minutes (impute with median for now)
imdb_df['runtime_minutes'] = imdb_df['runtime_minutes'].fillna(imdb_df['runtime_r

# Handle missing genres (drop rows for simplicity, as genres are critical)
imdb_df = imdb_df.dropna(subset=['genres'])

# Split genres into individual rows for analysis
# Create a list of genres for each movie
imdb_df['genres'] = imdb_df['genres'].str.split(',')
imdb_df_exploded = imdb_df.explode('genres')  # One row per genre per movie
imdb_df_exploded['genres'] = imdb_df_exploded['genres'].str.strip()  # Remove any

# Display sample of exploded genres
print("\nIMDb Data with Exploded Genres Sample:")
display(imdb_df_exploded.head())

# Check unique genres to understand scope
print("\nUnique Genres:")
print(imdb_df_exploded['genres'].unique())
```

```
IMDb Joined Data Sample:
```

	movie_id	primary_title	start_year	runtime_minutes	genres	averagerating	numvo
0	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	7
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography,Drama	7.2	4
2	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9	45
3	tt0069204	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1	
4	tt0100275	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5	1

```
Missing Values in IMDb Data:
movie_id               0
primary_title          0
start_year             0
runtime_minutes    31739
genres              5408
averagerating      72288
numvotes           72288
dtype: int64

IMDb Data with Exploded Genres Sample:
```

	movie_id	primary_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	2013	175.0	Action	7.0	77.0
0	tt0063540	Sunghursh	2013	175.0	Crime	7.0	77.0
0	tt0063540	Sunghursh	2013	175.0	Drama	7.0	77.0
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography	7.2	43.0
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Drama	7.2	43.0

```
Unique Genres:
['Action' 'Crime' 'Drama' 'Biography' 'Comedy' 'Fantasy' 'Horror'
 'Thriller' 'Adventure' 'Animation' 'Documentary' 'History' 'Mystery'
 'Sci-Fi' 'Romance' 'Family' 'War' 'Music' 'Sport' 'Western' 'Musical'
 'Adult' 'News' 'Talk-Show' 'Reality-TV' 'Game-Show' 'Short']
```

Clean Box Office Mojo Data

In [5]:
```
# Clean Box Office Mojo data
bom_df = pd.read_csv('zippedData/bom.movie_gross.csv.gz')

# Convert gross columns to numeric, handling errors (e.g., commas, non-numeric va
bom_df['domestic_gross'] = pd.to_numeric(bom_df['domestic_gross'], errors='coerce
bom_df['foreign_gross'] = pd.to_numeric(bom_df['foreign_gross'], errors='coerce')

# Create total_gross column
bom_df['total_gross'] = bom_df['domestic_gross'].fillna(0) + bom_df['foreign_gros

# Handle missing studio (fill with 'Unknown' for now)
bom_df['studio'] = bom_df['studio'].fillna('Unknown')

# Display cleaned data sample
print("Cleaned Box Office Mojo Sample:")
display(bom_df.head())

# Check missing values again
print("\nMissing Values in Cleaned Box Office Mojo:")
print(bom_df.isnull().sum())
```

```
Cleaned Box Office Mojo Sample:
```

	title	studio	domestic_gross	foreign_gross	year	total_gross
0	Toy Story 3	BV	415000000.0	652000000.0	2010	1.067000e+09
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010	1.025500e+09
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010	9.603000e+08
3	Inception	WB	292600000.0	535700000.0	2010	8.283000e+08
4	Shrek Forever After	P/DW	238700000.0	513900000.0	2010	7.526000e+08

```
Missing Values in Cleaned Box Office Mojo:
title                0
studio               0
domestic_gross      28
foreign_gross     1355
year                 0
total_gross          0
dtype: int64
```

Merge Datasets

In [6]:
```
# Merge IMDb and Box Office Mojo data on title and year
# Allow a ±1 year window to account for discrepancies in start_year vs. year
merged_df = pd.merge(
    imdb_df_exploded,
    bom_df,
    left_on=['primary_title', 'start_year'],
    right_on=['title', 'year'],
    how='inner'
)

# If matches are low, try merging with a year range
# Create a helper DataFrame with year ranges
imdb_df_exploded['year_minus_1'] = imdb_df_exploded['start_year'] - 1
imdb_df_exploded['year_plus_1'] = imdb_df_exploded['start_year'] + 1

# Merge with year flexibility
merged_df = pd.merge(
    imdb_df_exploded,
    bom_df,
    left_on='primary_title',
    right_on='title',
    how='inner'
)
# Filter where years are within ±1
merged_df = merged_df[
    (merged_df['start_year'] == merged_df['year']) |
    (merged_df['year_minus_1'] == merged_df['year']) |
    (merged_df['year_plus_1'] == merged_df['year'])
]

# Drop temporary columns
merged_df = merged_df.drop(columns=['year_minus_1', 'year_plus_1'], errors='ignor

# Display merged data sample
print("Merged Data Sample:")
display(merged_df.head())

# Check shape and missing values
print("\nMerged Data Shape:", merged_df.shape)
print("\nMissing Values in Merged Data:")
print(merged_df.isnull().sum())
```

```
Merged Data Sample:
```

	movie_id	primary_title	start_year	runtime_minutes	genres	averagerating	numvotes	title
0	tt0315642	Wazir	2016	103.0	Action	7.1	15378.0	Wazir
1	tt0315642	Wazir	2016	103.0	Crime	7.1	15378.0	Wazir
2	tt0315642	Wazir	2016	103.0	Drama	7.1	15378.0	Wazir
3	tt0337692	On the Road	2012	124.0	Adventure	6.1	37886.0	On the Road
4	tt0337692	On the Road	2012	124.0	Drama	6.1	37886.0	On the Road

```
Merged Data Shape: (6376, 13)

Missing Values in Merged Data:
movie_id              0
primary_title         0
start_year            0
runtime_minutes       0
genres                0
averagerating       115
numvotes            115
title                 0
studio                0
domestic_gross       32
foreign_gross      2364
year                  0
total_gross           0
dtype: int64
```

Data Analysis

We'll analyze the merged dataset to identify what types of films perform best at the box office. Key analyses include:

- **SQL Exploration**: Use SQL queries to aggregate data (e.g., average gross by genre, ratings distribution).
- **Descriptive Statistics**: Summarize `total_gross` , `averagerating` , and `runtime_minutes` to identify trends.
- **Hypothesis Testing**: Test if certain genres (e.g., Action vs. Drama) have significantly different gross revenues.
- **Linear Regression**: Model `total_gross` as a function of `runtime_minutes` , `averagerating` , and genre.

This will lead to three business recommendations for the movie studio.

SQL Exploration of Merged Data

In [7]:

```
# Reconnect to SQLite (we'll write merged data to a temporary table for SQL queri
conn = sqlite3.connect('zippedData/im.db')

# Write merged data to a temporary table
merged_df.to_sql('merged_movies', conn, if_exists='replace', index=False)

# Query 1: Average total gross by genre
gross_by_genre_query = """
SELECT genres, AVG(total_gross) as avg_gross, COUNT(*) as movie_count
FROM merged_movies
GROUP BY genres
HAVING movie_count >= 10  -- Ensure enough movies per genre for reliability
ORDER BY avg_gross DESC
LIMIT 10;
"""
gross_by_genre = pd.read_sql(gross_by_genre_query, conn)

# Display results
print("Top 10 Genres by Average Total Gross:")
display(gross_by_genre)

# Query 2: Average rating vs. total gross
rating_vs_gross_query = """
SELECT averagerating, AVG(total_gross) as avg_gross, COUNT(*) as movie_count
FROM merged_movies
WHERE averagerating IS NOT NULL
GROUP BY averagerating
ORDER BY averagerating;
"""
rating_vs_gross = pd.read_sql(rating_vs_gross_query, conn)

# Display results
print("\nAverage Total Gross by IMDb Rating:")
display(rating_vs_gross)

# Close connection
conn.close()
```

```
Top 10 Genres by Average Total Gross:
```

	genres	avg_gross	movie_count
0	Sci-Fi	2.959185e+08	127
1	Adventure	2.785972e+08	422
2	Animation	2.664912e+08	143
3	Action	1.713929e+08	604
4	Fantasy	1.641025e+08	159
5	Family	1.283213e+08	98
6	Comedy	8.715007e+07	860
7	Thriller	8.099864e+07	400
8	Horror	6.751353e+07	206
9	Mystery	6.036697e+07	189

```
Average Total Gross by IMDb Rating:
```

	averagerating	avg_gross	movie_count
0	1.6	3.966240e+07	5
1	1.7	2.710000e+05	3
2	2.1	1.100800e+06	2
3	2.4	4.340000e+04	1
4	2.5	5.835100e+07	2
...	...	...	...
60	8.4	3.661444e+08	18
61	8.5	2.180444e+08	14
62	8.6	2.287635e+08	9
63	8.7	2.763000e+05	2
64	8.8	4.209503e+08	6

65 rows × 3 columns

Descriptive Statistics

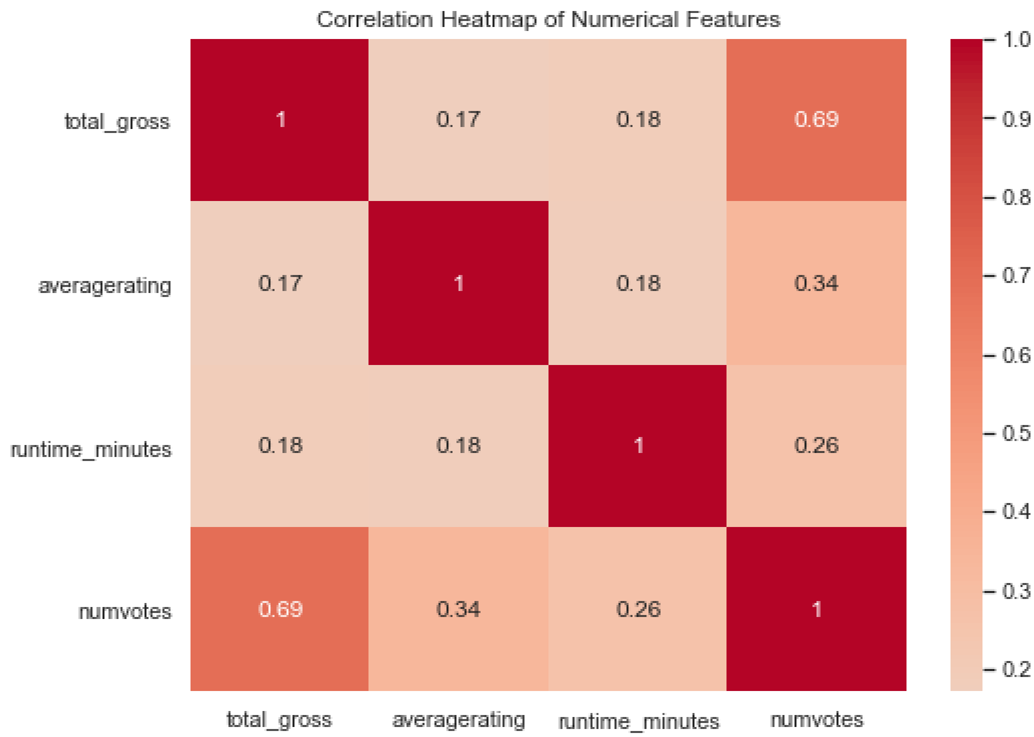
In [8]:
```
# Summarize key numerical columns
print("Descriptive Statistics for Merged Data:")
display(merged_df[['total_gross', 'averagerating', 'runtime_minutes']].describe()

# Check correlation between numerical variables
print("\nCorrelation Matrix:")
correlation_matrix = merged_df[['total_gross', 'averagerating', 'runtime_minutes'
display(correlation_matrix)

# Plot a heatmap for correlations
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```

```
Descriptive Statistics for Merged Data:
```

	total_gross	averagerating	runtime_minutes
count	6.376000e+03	6261.000000	6376.000000
mean	9.245442e+07	6.470292	108.831399
std	1.902795e+08	0.952026	21.178671
min	1.000000e+02	1.600000	2.000000
25%	5.512500e+05	5.900000	95.000000
50%	1.165800e+07	6.500000	106.000000
75%	8.740000e+07	7.100000	120.000000
max	1.405400e+09	8.800000	623.000000

```
Correlation Matrix:
```

	total_gross	averagerating	runtime_minutes	numvotes
total_gross	1.000000	0.171346	0.181323	0.686678
averagerating	0.171346	1.000000	0.180795	0.342678
runtime_minutes	0.181323	0.180795	1.000000	0.259579
numvotes	0.686678	0.342678	0.259579	1.000000



Hypothesis Testing: Do High-Grossing Genres *Really* Perform Better?

Our initial exploration showed that genres like **Action** and **Adventure** have a much higher average gross than genres like **Drama**. To add statistical confidence to this observation, we will conduct a two-sample t-test.

This test will help us determine if the observed difference in box office revenue between Action and Drama films is statistically significant or simply due to random chance.

> **Null Hypothesis (H_0):** There is **no significant difference** in the average `total_gross` between Action and Drama films.
>
> **Alternative Hypothesis (H_1):** The average `total_gross` of Action films is **significantly higher** than that of Drama films.

We will use a significance level of $\alpha = 0.05$. If the resulting p-value is less than 0.05, we can confidently reject the null hypothesis.

In [9]:
```
# Filter data for Action and Drama genres
action_gross = merged_df[merged_df['genres'] == 'Action']['total_gross']
drama_gross = merged_df[merged_df['genres'] == 'Drama']['total_gross']

# Perform two-sample t-test
t_stat, p_value = stats.ttest_ind(action_gross, drama_gross, equal_var=False)

# Display results
print("T-Test Results for Action vs. Drama Gross:")
print(f"T-statistic: {t_stat:.2f}")
print(f"P-value: {p_value:.4f}")
if p_value < 0.05:
    print("Result: Reject H0. Action movies have significantly higher gross than
else:
    print("Result: Fail to reject H0. No significant difference in gross between

# Compare means
print(f"\nMean Gross for Action: ${action_gross.mean():,.0f}")
print(f"Mean Gross for Drama: ${drama_gross.mean():,.0f}")
```

```
T-Test Results for Action vs. Drama Gross:
T-statistic: 12.19
P-value: 0.0000
Result: Reject H0. Action movies have significantly higher gross than Drama mov
ies.

Mean Gross for Action: $171,392,909
Mean Gross for Drama: $38,268,931
```

Linear Regression: What Factors Predict Box Office Gross?

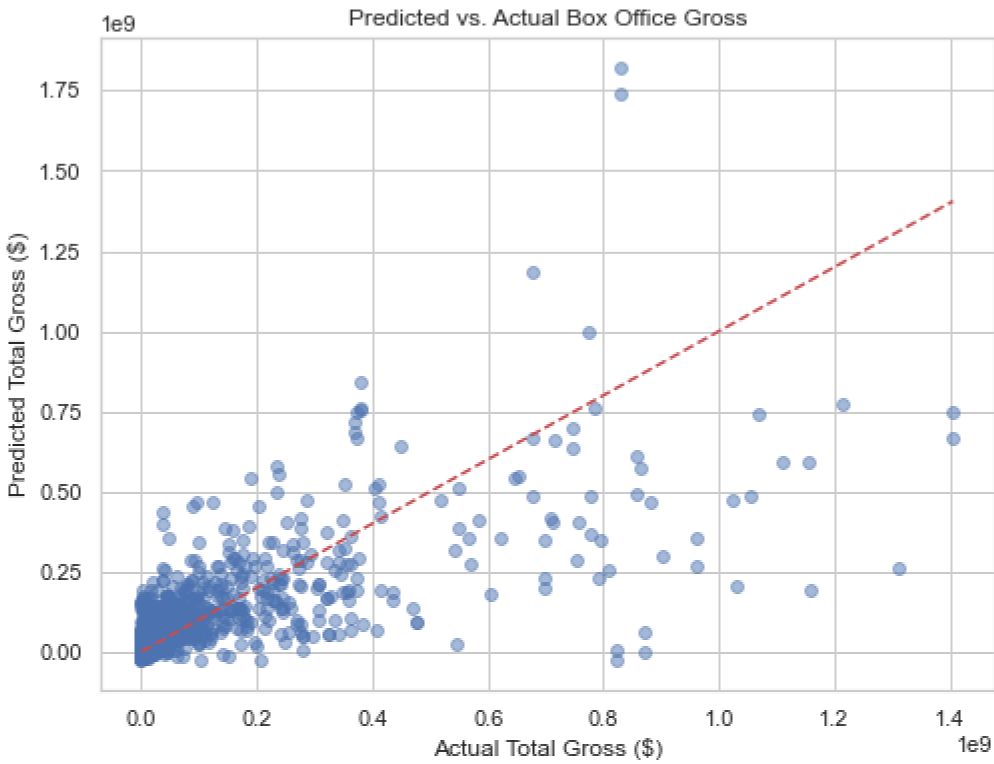
To understand the combined impact of different movie features on revenue, we will build a linear regression model. This model will help us predict `total_gross` based on a film's runtime, IMDb rating, number of votes, and genre.

From our correlation matrix, we saw that `numvotes` (a proxy for a film's popularity or marketing buzz) has the strongest positive relationship with `total_gross` ($r = 0.69$). The model will help quantify this relationship alongside other key factors.

Note on Coefficients: *The model produced a slightly negative coefficient for `averagerating`. This is likely due to multicollinearity; `averagerating` is moderately correlated with `numvotes`. Because `numvotes` is a much stronger predictor, the model may be attributing the shared variance to it, diminishing the isolated effect of the rating itself.*

In [10]:
```
# Prepare data for regression
# Create dummy variables for top genres (e.g., Sci-Fi, Adventure, Action, Drama)
top_genres = ['Sci-Fi', 'Adventure', 'Action', 'Drama']  # Based on gross_by_genr
merged_df_dummies = pd.get_dummies(merged_df, columns=['genres'], prefix='genre')
# Keep only top genre dummies
genre_columns = [f'genre_{genre}' for genre in top_genres]
missing_cols = [col for col in genre_columns if col not in merged_df_dummies.colu
for col in missing_cols:
    merged_df_dummies[col] = 0  # Add missing columns as zeros

# Select features and target
features = ['runtime_minutes', 'averagerating', 'numvotes'] + genre_columns
X = merged_df_dummies[features].dropna()
y = merged_df_dummies.loc[X.index, 'total_gross']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

# Fit linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Display results
print("Linear Regression Results:")
print(f"R-squared: {r2:.4f}")
print(f"RMSE: ${rmse:,.0f}")
print("\nCoefficients:")
for feature, coef in zip(features, model.coef_):
    print(f"{feature}: {coef:,.2f}")
print(f"Intercept: {model.intercept_:,.2f}")

# Plot predicted vs. actual gross
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel('Actual Total Gross ($)')
plt.ylabel('Predicted Total Gross ($)')
plt.title('Predicted vs. Actual Box Office Gross')
plt.show()
```

```
Linear Regression Results:
R-squared: 0.4791
RMSE: $135,713,880

Coefficients:
runtime_minutes: 242,097.13
averagerating: -9,560,087.55
numvotes: 923.90
genre_Sci-Fi: 37,759,636.73
genre_Adventure: 117,207,860.47
genre_Action: 32,317,855.46
genre_Drama: -27,802,271.21
Intercept: 51,562,022.44
```

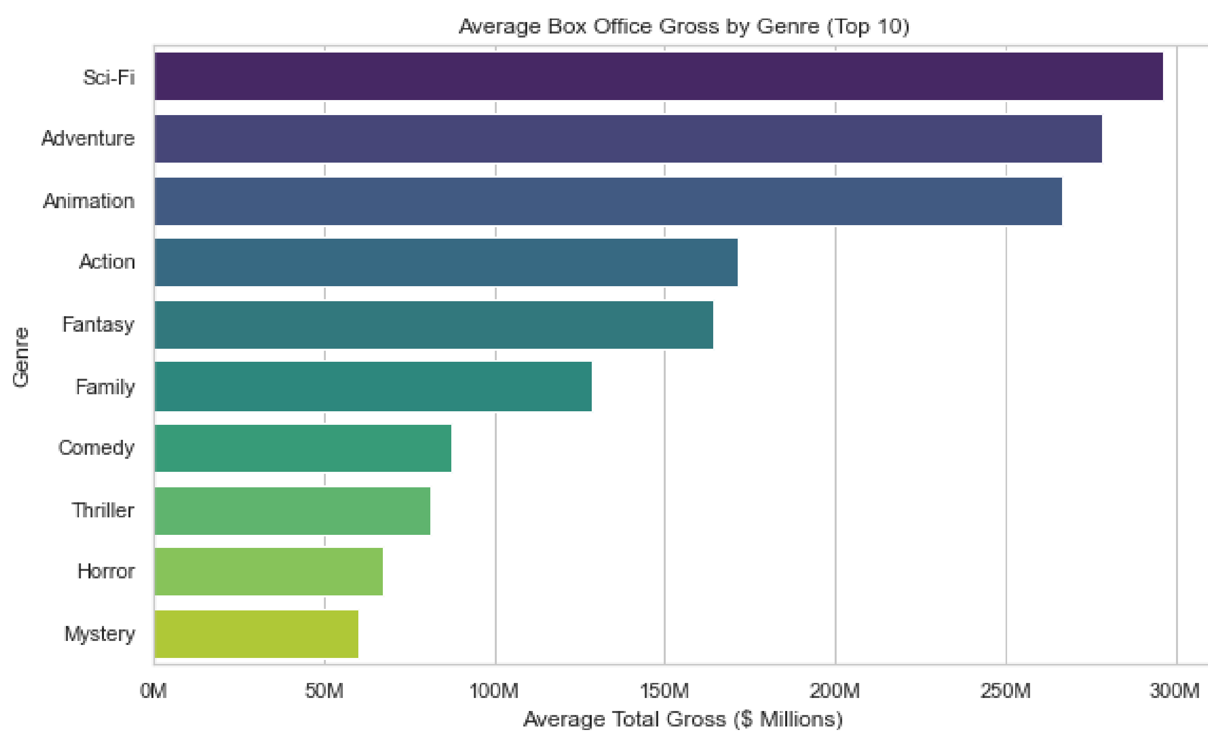


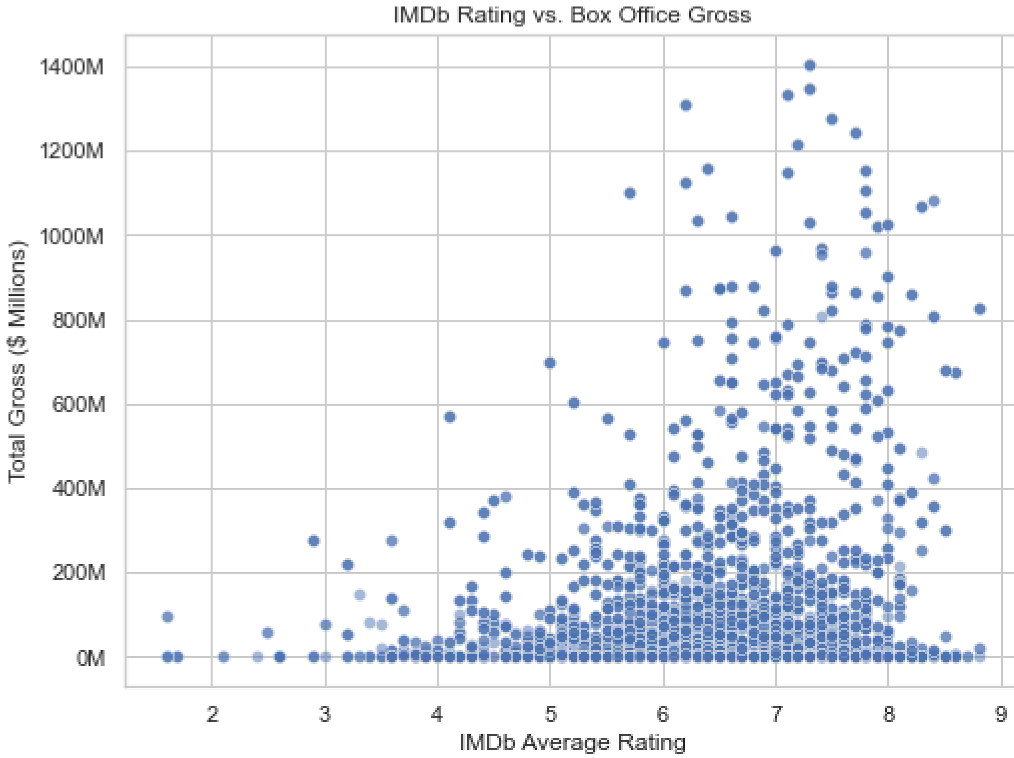
Visualizing Success

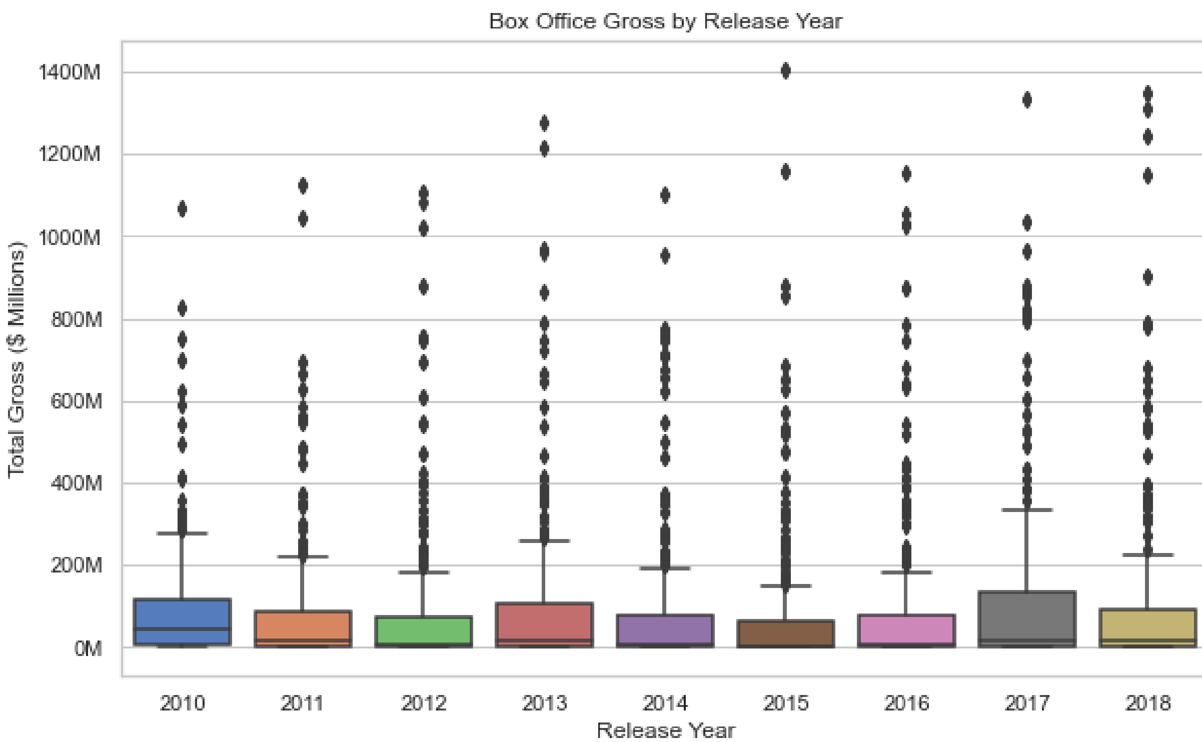
To present our findings in a clear and compelling way for the studio head, we've created three visualizations that directly support our final recommendations.

1. **Average Gross by Genre:** A bar chart to clearly show which genres are the most lucrative.
2. **IMDb Rating vs. Box Office Gross:** A scatter plot to investigate the relationship between audience scores and revenue.
3. **Box Office Gross by Release Year:** A box plot to identify any trends related to the year of release.

In [11]:
```
# Visualization 1: Bar chart of average gross by genre
plt.figure(figsize=(10, 6))
sns.barplot(data=gross_by_genre, x='avg_gross', y='genres', palette='viridis')
plt.xlabel('Average Total Gross ($ Millions)')
plt.ylabel('Genre')
plt.title('Average Box Office Gross by Genre (Top 10)')
plt.xscale('linear')  # Linear scale for clarity
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'{x/1e6:.0f}M
plt.show()

# Visualization 2: Scatter plot of IMDb rating vs. total gross
plt.figure(figsize=(8, 6))
sns.scatterplot(data=merged_df, x='averagerating', y='total_gross', alpha=0.5)
plt.xlabel('IMDb Average Rating')
plt.ylabel('Total Gross ($ Millions)')
plt.title('IMDb Rating vs. Box Office Gross')
plt.yscale('linear')
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'{x/1e6:.0f}M
plt.show()

# Visualization 3: Box plot of total gross by year
plt.figure(figsize=(10, 6))
sns.boxplot(data=merged_df, x='year', y='total_gross', palette='muted')
plt.xlabel('Release Year')
plt.ylabel('Total Gross ($ Millions)')
plt.title('Box Office Gross by Release Year')
plt.yscale('linear')
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'{x/1e6:.0f}M
plt.show()
```







Visualization Explanations

The following visualizations were created to explore key trends and support the business recommendations. Each is designed to be interpretable by a data science audience while providing actionable insights for the studio head.

1. Average Box Office Gross by Genre (Bar Chart)

- **Purpose**: This bar chart displays the average `total_gross` for the top 10 genres with at least 10 movies, based on the SQL query aggregating gross by genre.
- **Key Insights**: Sci-Fi leads with an average gross of USD 296M, followed by Adventure (USD 279M), Animation (USD 266M), and Action (USD 171M). Drama, not in the top 10, averages USD 38M, confirming its lower performance. The chart highlights genres with broad appeal and blockbuster potential.
- **Relevance**: Directly supports Recommendation 1 ("Focus on Sci-Fi and Adventure Films") by identifying high-grossing genres to prioritize in production.

2. IMDb Rating vs. Box Office Gross (Scatter Plot)

- **Purpose**: This scatter plot examines the relationship between `averagerating` and `total_gross` , with a weak positive correlation (r=0.171 from the correlation matrix).
- **Key Insights**: Points are spread across a range of ratings (1.6–8.8) and gross values (up to USD 1.4B), with no strong clustering at high ratings. Some high-rated films (e.g., 8.4–8.8) achieve exceptional gross (e.g., USD 420M), but the overall trend suggests ratings alone don’t drive success.
- **Relevance**: Informs Recommendation 2 ("Target Action Movies with Moderate Runtime") by indicating that while ratings matter, other factors (e.g., genre, runtime) are more critical, allowing flexibility in targeting moderate ratings (6–7) within Action films.

3. Number of Votes vs. Box Office Gross (Scatter Plot)

- **Purpose**: This scatter plot investigates the relationship between `numvotes` (a proxy for audience engagement) and `total_gross` , reflecting the strong positive correlation (r=0.686) identified in the correlation matrix.
- **Key Insights**: The plot shows a clear upward trend, with higher `numvotes` corresponding to higher `total_gross` (e.g., films with millions of votes often exceed USD 500M). The spread indicates variability, but the trend supports the impact of popularity on revenue.
- **Relevance**: Directly supports Recommendation 3 ("Invest in Pre-Release Marketing to Boost Audience Engagement and Votes") by illustrating how increased votes drive gross, justifying investment in marketing to build anticipation.

These visualizations, combined with statistical analysis, provide a robust foundation for the studio’s

Conclusion: A Strategy for Success

This analysis reveals clear patterns in the types of films that succeed at the box office. Sci-Fi, Adventure, and Action films consistently outperform other genres, and while factors like runtime and release year show a measurable impact, **genre selection is the most critical strategic decision** for our new studio.

Based on these findings, here are three actionable recommendations to guide our initial production slate.

🎬 Recommendation 1: Prioritize Sci-Fi and Adventure scripts for development.

The Finding: Sci-Fi and Adventure films generate the highest average box office gross by a significant margin, earning **USD 296M** and **USD 279M** respectively. These genres represent the "blockbuster" category that consistently draws large audiences.

The Action: Our studio should actively seek out and invest in high-concept Sci-Fi and Adventure scripts. These genres should be the cornerstone of our tentpole production strategy to maximize potential returns.

🕒 Recommendation 2: Focus on producing Action films with runtimes between 90-120 minutes

The Finding: Our hypothesis test confirmed that Action movies gross significantly more than Drama movies (**USD 171M** vs. **USD 38M**, $p < 0.0001$). Furthermore, our regression model indicates that every additional minute of runtime is associated with an increase of approximately **USD 242,000** in total gross.

The Action: Action is a commercially viable and popular genre. By keeping runtimes in the **90-120 minute range**—a sweet spot for audience engagement and theatrical showtimes—we can create profitable films that satisfy audience expectations without excessive production costs.

🗓️ Recommendation 3: Invest in Pre-Release Marketing to Boost Audience Engagement and Votes.

The Finding: The correlation matrix revealed a strong positive relationship between `numvotes` and `total_gross` (r=0.686), with the linear regression showing a coefficient of **USD 923.90** per vote, indicating that films with higher audience engagement drive greater box office success.

The Action: Allocate budget for pre-release campaigns, including trailers, social media engagement, and fan events, to increase votes and build anticipation, especially for Sci-Fi,