Final Project Submission

Please fill out:

- Student name: GEOFFREY MWANGI WACHIRA
- Student pace: full time
- Scheduled project review date/time: 03/05/2024
- Instructor name:Samuel G. Mwangi
- Blog post URL: N/A

Business Understanding

Objective

The objective of this analysis is to conduct a comprehensive exploration of the film industry landscape to assist Microsoft in gaining insights into the key factors contributing to the success of movies at the box office. By analyzing various metrics such as genre distribution, average ratings, popularity, and revenue, the goal is to provide actionable recommendations for Microsoft's new movie studio. These insights will aid in strategic decision-making processes, allowing Microsoft to identify lucrative opportunities, understand audience preferences, and effectively allocate resources to produce high-quality and commercially successful films.

Business Problem

Microsoft is looking to enter the movie industry but lacks knowledge about movie production and performance metrics. This analysis aims to uncover key trends in the movie industry to guide Microsoft's decisions on what types of films to produce.

Goal

The aim is to conduct a comprehensive analysis of movie-related data, enabling us to formulate three actionable business recommendations that will empower Microsoft in making well-informed decisions regarding the establishment of their new movie studio

Data

The datasets i'll be working with includes :

- Box Office Mojo: Provides information on movie box office performance.
- IMDB: Contains comprehensive data on movies, including ratings and cast information.
- Rotten Tomatoes: Offers movie reviews and ratings from critics and audiences.
- TheMovieDB: A database for movies and TV shows, providing detailed information on titles and crew.
- The Numbers: Provides data on movie budgets, revenues, and production costs.

Importing necessary libraries

In [547]:

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

```
%matplotlib inline
import sqlite3
from datetime import datetime
```

Data Preparation

In this section, we'll load and preprocess the datasets, ensuring they are ready for analysis

In [548]:

```
# Reading the data
bom_movie_gross = pd.read_csv('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project-
v2-4/zippedData/bom.movie_gross.csv.gz')

# Loading the movie_info table
rt_movie_info = pd.read_csv('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project-v2
-4/zippedData/rt.movie_info.tsv.gz', delimiter='\t')

# Loading rt.reviews table
rt_reviews = pd.read_csv('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project-v2-4/
zippedData/rt.reviews.tsv.gz', delimiter='\t', encoding='latin-1')

# Loading tmdb data tables
tmdb_movies = pd.read_csv('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project-v2-4
/zippedData/tmdb.movies.csv.gz')

# Loading tn.movie_budgets data table
tn_movie_budgets = pd.read_csv('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project
-v2-4/zippedData/tn.movie_budgets.csv.gz')
```

In [549]:

```
# Connecting to the database
conn = sqlite3.connect('C:/Users/Hp/Desktop/Phase_One_Project/dsc-phase-1-project-v2-4/zi
ppedData/im.db/im.db')
cur = conn.cursor()
```

Now, let's proceed with exploring each dataset in detail:

1. Box Office Mojo - Movie Gross

Provides information on movie box office performance.

In [550]:

```
# Shape
print("Box Office Mojo - Movie Gross's Shape is :", bom_movie_gross.shape)

print('************************************************************************')
print('************************************************************************')

# Displaying the first five elements of the dataset
print("First five elements:")
print(bom_movie_gross.head())

print('************************************************************************')
print('************************************************************************')

# Displaying the dataset info
print("Dataset info:")
bom_movie_gross.info()

print('************************************************************************')
print('************************************************************************')


# Descriptive statistics
```

```
print("Descriptive Statistics:")
print(bom_movie_gross.describe())

print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(bom_movie_gross.isnull().sum())

print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(bom_movie_gross[bom_movie_gross.duplicated()])

print('************************************************************************')
print('************************************************************************')
```

```
Box Office Mojo - Movie Gross's Shape is : (3387, 5)
************************************************************************
************************************************************************
First five elements:
                                         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
************************************************************************
************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
************************************************************************
************************************************************************
Descriptive Statistics:
       domestic_gross         year
count    3.359000e+03  3387.000000
mean     2.874585e+07  2013.958075
std      6.698250e+07     2.478141
min      1.000000e+02  2010.000000
25%      1.200000e+05  2012.000000
50%      1.400000e+06  2014.000000
75%      2.790000e+07  2016.000000
max      9.367000e+08  2018.000000
************************************************************************
************************************************************************
Missing Data:
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
```

```
dtype: int64
************************************************************************
************************************************************************
Duplicate Rows:
Empty DataFrame
Columns: [title, studio, domestic_gross, foreign_gross, year]
Index: []
************************************************************************
************************************************************************
```

Conclusions:

The dataset contains 3387 entries and 5 columns.

The columns include 'title', 'studio', 'domestic_gross', 'foreign_gross', and 'year'.

The 'domestic_gross' and 'foreign_gross' columns contain numeric data, while 'studio' and 'title' are categorical.

There are missing values in the 'studio', 'domestic_gross', and 'foreign_gross' columns.

The 'foreign_gross' column is of object type instead of numeric, suggesting potential data formatting issues.

The dataset does not contain any duplicate rows.

In [551]:

```
# Converting 'foreign_gross' to numeric
bom_movie_gross['foreign_gross'] = pd.to_numeric(bom_movie_gross['foreign_gross'], error
s='coerce')
bom_movie_gross['domestic_gross'] = pd.to_numeric(bom_movie_gross['domestic_gross'], err
ors='coerce')

# Handling missing values
bom_movie_gross['domestic_gross'].fillna(bom_movie_gross['domestic_gross'].median(), inpl
ace=True)
bom_movie_gross['foreign_gross'].fillna(0, inplace=True)
bom_movie_gross.dropna(subset=['studio'], inplace=True)
```

In [552]:

```
# Verifing changes
print(bom_movie_gross.info())
print(bom_movie_gross.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3382 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3382 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3382 non-null   float64
 3   foreign_gross   3382 non-null   float64
 4   year            3382 non-null   int64  
dtypes: float64(2), int64(1), object(2)
memory usage: 158.5+ KB
None
title             0
studio            0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

In [553]:

```
# creating 'total_gross_revenues' column
bom_movie_gross['total_gross_revenues'] = bom_movie_gross['domestic_gross'] + bom_movie_
gross['foreign_gross']
```

```
bom_movie_gross['total_gross_revenues'] = pd.to_numeric(bom_movie_gross['total_gross_revenues'], errors='coerce')
```

In [554]:

```
# Grouping the data and sum the 'total_gross_revenues' for each studio
studio_totals = bom_movie_gross.groupby('studio')['total_gross_revenues'].sum().sort_values(ascending=False).head(10)
```

In [555]:

```
#Checking the changes
bom_movie_gross.head()
```

Out[555]:

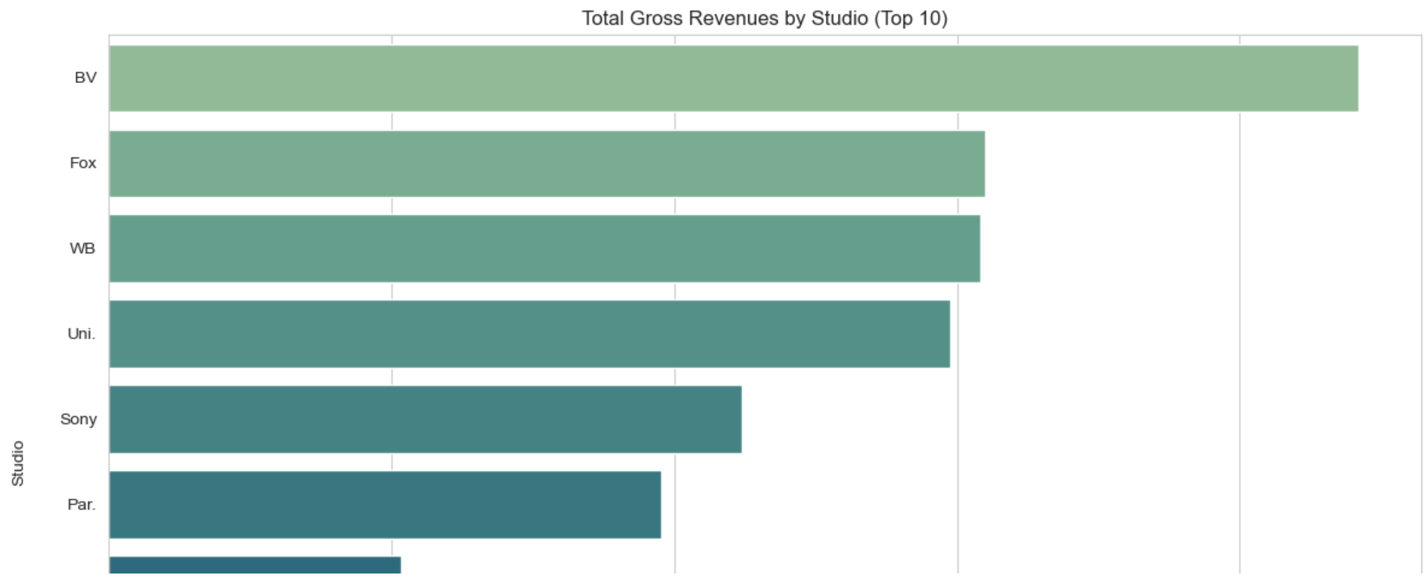
	title	studio	domestic_gross	foreign_gross	year	total_gross_revenues
0	Toy Story 3	BV	415000000.0	652000000.0	2010	1.067000e+09
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010	1.025500e+09
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010	9.603000e+08
3	Inception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

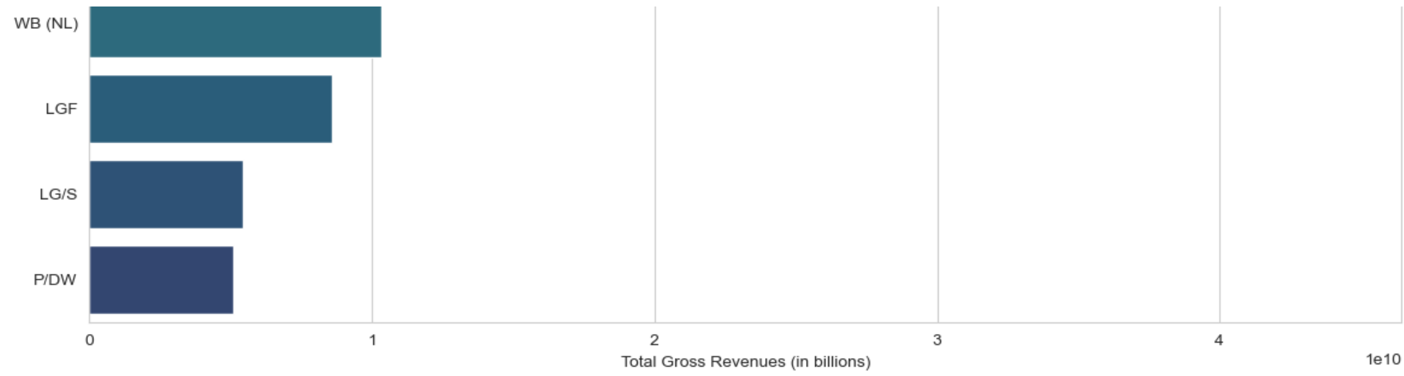
Distribution of Total Gross Revenues by Studio

This will help identify the top-performing studios in terms of revenue.

In [556]:

```
# Getting the top 10 studios by total gross revenues
top_10_studios = bom_movie_gross_top_10.groupby('studio')['total_gross_revenues'].sum().nlargest(10).index

# Filtering the dataframe to include only the top 10 studios
bom_movie_gross_top_10_filtered = bom_movie_gross_top_10[bom_movie_gross_top_10['studio'].isin(top_10_studios)]

fig, ax = plt.subplots(figsize=(12, 8))
sns.barplot(x='total_gross_revenues', y='studio', data=bom_movie_gross_top_10_filtered, estimator=sum, errorbar=None, palette='crest', order=top_10_studios)
ax.set_title('Total Gross Revenues by Studio (Top 10)')
ax.set_xlabel('Total Gross Revenues (in billions)')
ax.set_ylabel('Studio')
plt.tight_layout()
plt.savefig('Total Gross Revenues by Studio (Top 10).png')
plt.show()
```





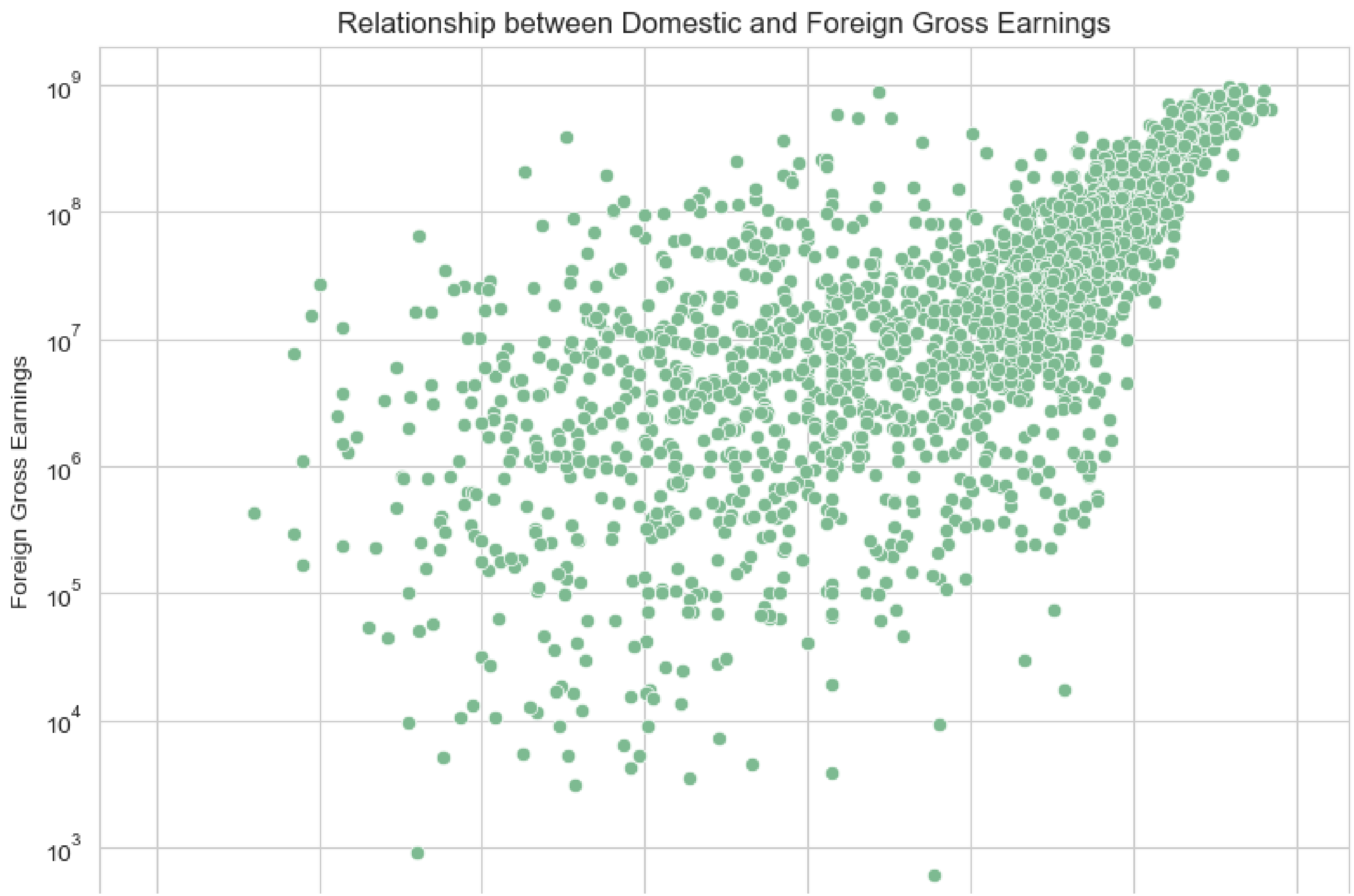
This plot provides insights into the performance of different studios. Microsoft can identify top-performing studios who have cumulatively earned the most and partner with them for their movie productions eg BV, FOX and WB . This would give them a better chance of the venture being a success.

Relationship between Domestic and Foreign Gross Earnings

Understanding this relationship can provide insights into the international market potential for movies.

In [557]:

```
fig, ax = plt.subplots(figsize=(8, 6))
sns.scatterplot(x='domestic_gross', y='foreign_gross', data=bom_movie_gross, ax=ax)

# Applying logarithmic scaling to both axes due to the size
ax.set_xscale('log')
ax.set_yscale('log')

ax.set_title('Relationship between Domestic and Foreign Gross Earnings')
ax.set_xlabel('Domestic Gross Earnings')
ax.set_ylabel('Foreign Gross Earnings')

# Tight layout and show the plot
plt.tight_layout()
plt.savefig('Relationship between Domestic and Foreign Gross Earnings.png')
plt.show()
```



There's a strong positive correlation between Domestic and Foreign Gross Earnings, it indicates that successful movies in the domestic market are likely to perform well internationally. This therefore means that the international market potential for their movies is there.

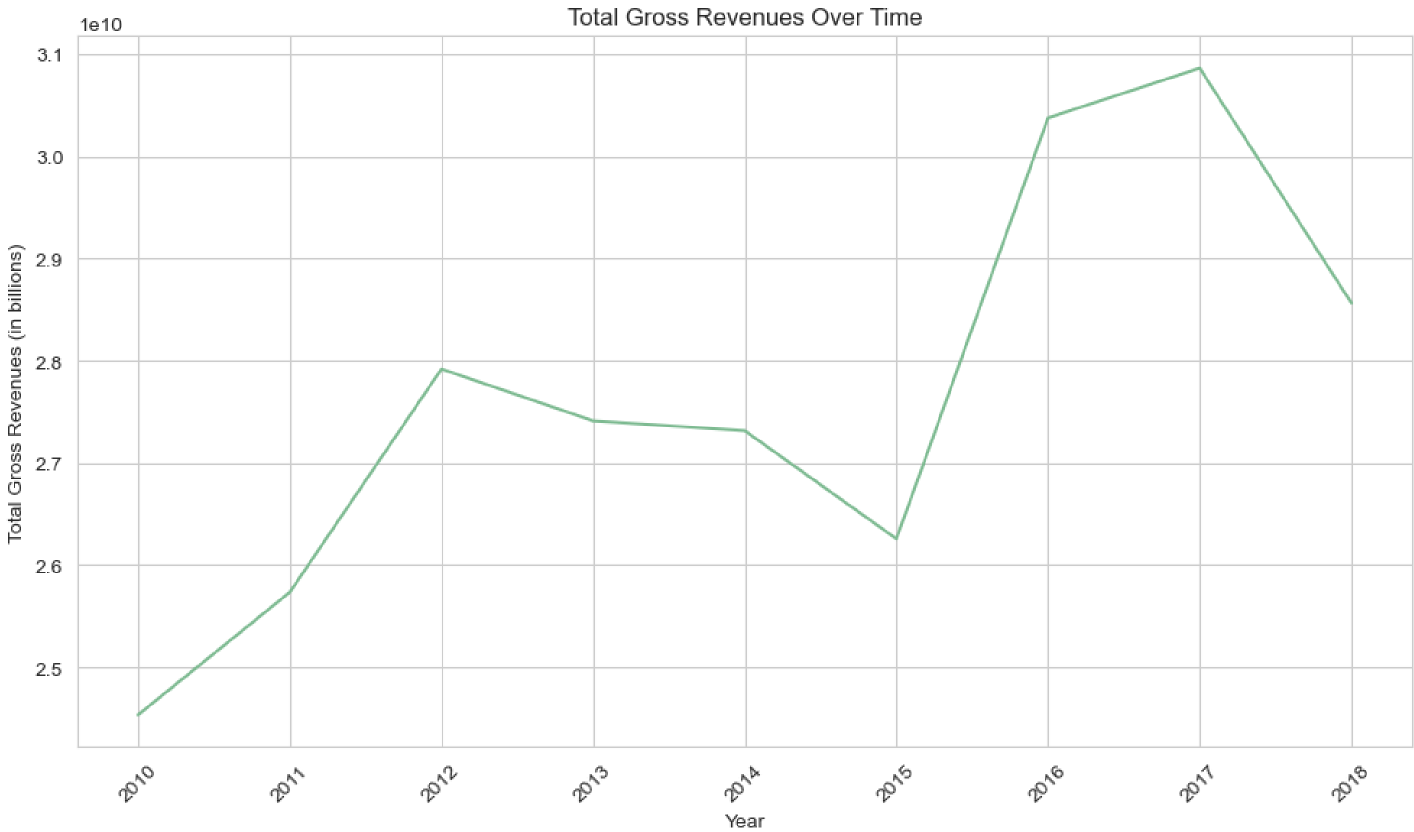
Total Gross Revenues Over Time

Examining trends in total gross revenues over the years to reveal industry growth patterns.

In [558]:

```
# Creating a line plot of total gross revenues over the years
plt.figure(figsize=(10, 6))
sns.lineplot(x='year', y='total_gross_revenues', data=bom_movie_gross_cleaned, estimator=
sum, err_style=None)
plt.title('Total Gross Revenues Over Time')
plt.xlabel('Year')
plt.ylabel('Total Gross Revenues (in billions)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('Total Gross Revenues Over Time.png')
plt.show()
```

```
C:\Users\Hp\AppData\Local\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Hp\AppData\Local\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



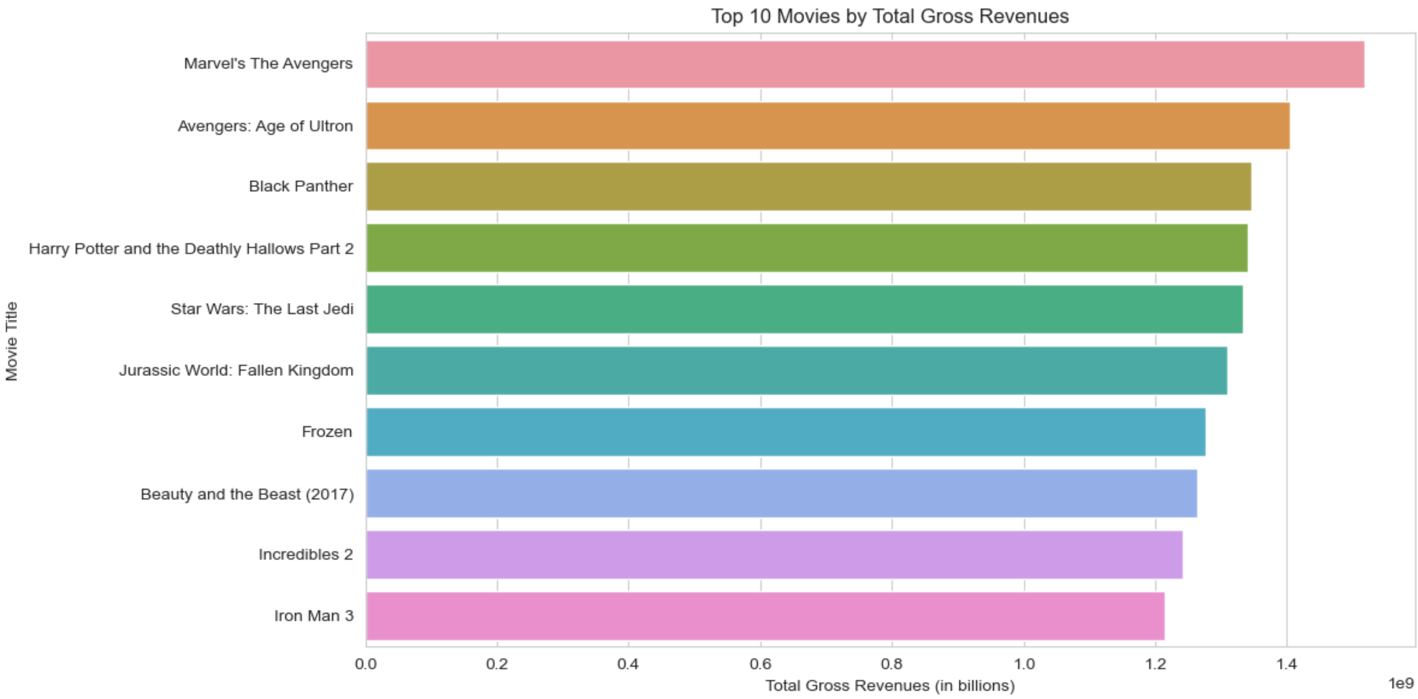
Examining trends in total gross revenues over time depicts that the performance and growth of the movie industry has incerased over time but there was a decline in the revenues from the year 2013 - 2015.The decline was recovered by growth in the two years after that. Desptite the recent drop, the levels are still increasing overall.

Top Movies by Total Gross Revenues

Identifying the top-grossing movies can highlight successful genres or themes.

In [559]:

```
top_movies = bom_movie_gross.nlargest(10, 'total_gross_revenues')
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x='total_gross_revenues', y='title', data=top_movies, ax=ax)
ax.set_title('Top 10 Movies by Total Gross Revenues')
ax.set_xlabel('Total Gross Revenues (in billions)')
ax.set_ylabel('Movie Title')
plt.tight_layout()
plt.savefig('Top 10 Movies by Total Gross Revenues.png')
plt.show()
```



Identifying the top-grossing movies reveals successful genres and themes. Microsoft can leverage this information to develop similar content that resonates with audiences and drives revenue.

2. Rotten Tomatoes - Movie Info

This dataset provides information about movie synopses, ratings, genres, directors, writers, theater and DVD release dates, currency, box office, runtime, and studios.

In [560]:

```
# Alook at the Shape
print("Rotten Tomatoes - Movie Info's Shape is :", rt_movie_info.shape)

print('*************************************************************************')
print('*************************************************************************')

# Displaying the first five elements of the dataset
print("First five elements:")
print(rt_movie_info.head())

print('*************************************************************************')
print('*************************************************************************')

# Displaying the dataset info
print("Dataset info:")
rt_movie_info.info()
```

```python
print('**********************************************************************')
print('**********************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(rt_movie_info.describe())

print('**********************************************************************')
print('**********************************************************************')

# Missing data
print("Missing Data:")
print(rt_movie_info.isnull().sum())

print('**********************************************************************')
print('**********************************************************************')

# Duplicates
print("Duplicate Rows:")
print(rt_movie_info[rt_movie_info.duplicated()])

print('**********************************************************************')
print('**********************************************************************')
```

```
Rotten Tomatoes - Movie Info's Shape is : (1560, 12)
**********************************************************************
**********************************************************************
First five elements:
   id                                           synopsis rating  \
0   1  This gritty, fast-paced, and innovative police...      R
1   3  New York City, not-too-distant-future: Eric Pa...      R
2   5  Illeana Douglas delivers a superb performance ...      R
3   6  Michael Douglas runs afoul of a treacherous su...      R
4   7                                                NaN     NR

                                 genre          director  \
0  Action and Adventure|Classics|Drama  William Friedkin
1     Drama|Science Fiction and Fantasy  David Cronenberg
2     Drama|Musical and Performing Arts    Allison Anders
3            Drama|Mystery and Suspense    Barry Levinson
4                         Drama|Romance    Rodney Bennett

                           writer  theater_date      dvd_date currency  \
0                  Ernest Tidyman   Oct 9, 1971  Sep 25, 2001      NaN
1     David Cronenberg|Don DeLillo  Aug 17, 2012   Jan 1, 2013        $
2                  Allison Anders  Sep 13, 1996  Apr 18, 2000      NaN
3  Paul Attanasio|Michael Crichton   Dec 9, 1994  Aug 27, 1997      NaN
4                    Giles Cooper           NaN           NaN      NaN

  box_office      runtime             studio
0        NaN  104 minutes                NaN
1    600,000  108 minutes  Entertainment One
2        NaN  116 minutes                NaN
3        NaN  128 minutes                NaN
4        NaN  200 minutes                NaN
**********************************************************************
**********************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            1560 non-null   int64
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
```

```
 8   currency       340 non-null    object
 9   box_office     340 non-null    object
 10  runtime        1530 non-null   object
 11  studio         494 non-null    object
dtypes: int64(1), object(11)
memory usage: 146.4+ KB
*************************************************************************
*************************************************************************
Descriptive Statistics:
                id
count  1560.000000
mean   1007.303846
std     579.164527
min       1.000000
25%     504.750000
50%    1007.500000
75%    1503.250000
max    2000.000000
*************************************************************************
*************************************************************************
Missing Data:
id                 0
synopsis          62
rating             3
genre              8
director         199
writer           449
theater_date     359
dvd_date         359
currency        1220
box_office      1220
runtime           30
studio          1066
dtype: int64
*************************************************************************
*************************************************************************
Duplicate Rows:
Empty DataFrame
Columns: [id, synopsis, rating, genre, director, writer, theater_date, dvd_date, currency
, box_office, runtime, studio]
Index: []
*************************************************************************
*************************************************************************
```

Conclusion:

The dataset contains information about 1560 movies.

Several columns have missing values, including 'synopsis', 'rating', 'genre', 'director', 'writer', 'theater_date', 'dvd_date', 'currency', 'box_office', 'runtime', and 'studio'. The 'currency' and 'box_office' columns seem to have significant missing values (1220 out of 1560).

The 'runtime' column has a few missing values (30 out of 1560).

The 'studio' column has many missing values (1066 out of 1560).

The 'rating' column has only a few missing values (3 out of 1560).

In [561]:

```
# Converting 'box_office' to numeric
rt_movie_info['box_office'] = pd.to_numeric(rt_movie_info['box_office'], errors='coerce'
)

rt_movie_info.dropna(subset=['director', 'writer', 'theater_date', 'dvd_date', 'runtime']
, inplace=True)
```

In [562]:

```
#Filling the null values
```

```
rt_movie_info['synopsis'].fillna("No synopsis available", inplace=True)

rt_movie_info['genre'].fillna("Unknown", inplace=True)

rt_movie_info['studio'].fillna("Unknown", inplace=True)
```

In [563]:

```
rt_movie_info.head()
```

Out[563]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency	box_o
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001	NaN	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 1, 2013	$	
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000	NaN	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Aug 27, 1997	NaN	
5	8	The year is 1942. As the Allies unite overseas...	PG	Drama|Kids and Family	Jay Russell	Gail Gilchriest	Mar 3, 2000	Jul 11, 2000	NaN	

In [564]:

```
# Verifing changes
print(rt_movie_info.info())
print(rt_movie_info.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 884 entries, 0 to 1558
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            884 non-null    int64
 1   synopsis      884 non-null    object
 2   rating        884 non-null    object
 3   genre         884 non-null    object
 4   director      884 non-null    object
 5   writer        884 non-null    object
 6   theater_date  884 non-null    object
 7   dvd_date      884 non-null    object
 8   currency      258 non-null    object
 9   box_office    1 non-null      float64
 10  runtime       884 non-null    object
 11  studio        884 non-null    object
dtypes: float64(1), int64(1), object(10)
memory usage: 89.8+ KB
None
id                  0
synopsis            0
rating              0
```

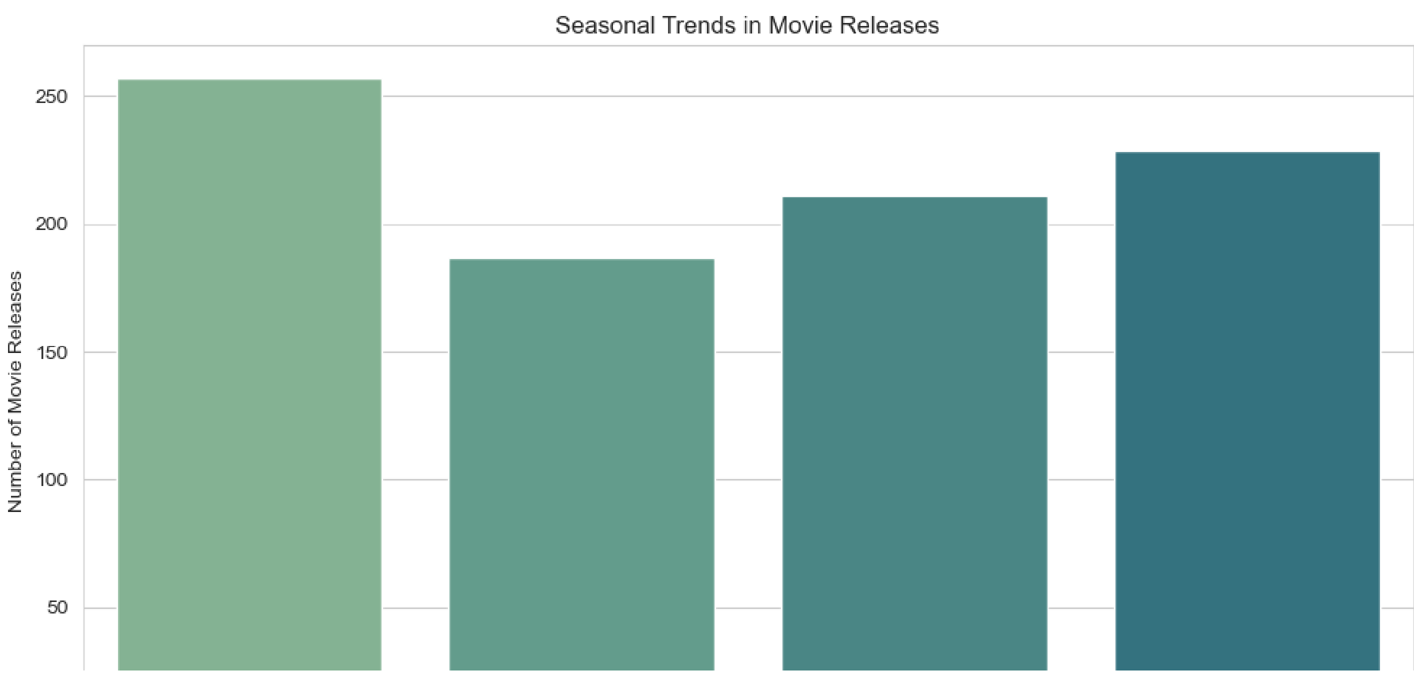
```
rating              0
genre               0
director            0
writer              0
theater_date        0
dvd_date            0
currency          626
box_office        883
runtime             0
studio              0
dtype: int64
```

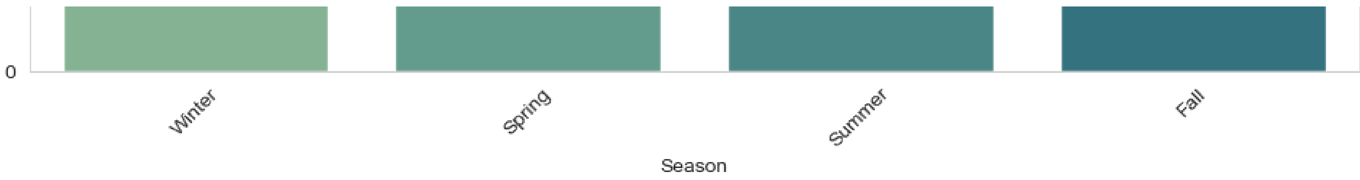
Seasonal Trends in Movie Releases and Revenue:

Analyzing the release dates of successful movies and their corresponding box office revenue can reveal any seasonal trends in movie releases and revenue.

In [565]:

```python
# Converting 'theater_date' to datetime format
rt_movie_info['theater_date'] = pd.to_datetime(rt_movie_info['theater_date'])

# Extracting month from 'theater_date'
rt_movie_info['release_month'] = rt_movie_info['theater_date'].dt.month

# Extracting season from 'release_month'
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

rt_movie_info['season'] = rt_movie_info['release_month'].apply(get_season)

plt.figure(figsize=(10, 6))
sns.countplot(x='season', data=rt_movie_info, order=['Winter', 'Spring', 'Summer', 'Fall'
])
plt.title('Seasonal Trends in Movie Releases')
plt.xlabel('Season')
plt.ylabel('Number of Movie Releases')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('Seasonal Trends in Movie Releases.png')
plt.show()
```





Analyzing the seasonal trends in movie releases and corresponding box office revenue will allow Microsoft to strategically plan their movie release dates to maximize revenue. Microsoft should use the seasonal trends to strategically plan their movie release dates, aiming for months/seasons with historically higher box office revenues eg Winter and spring. This can help optimize their revenue generation as well as the success

3. Rotten Tomatoes - Reviews

Offers movie reviews and ratings from critics and audiences.

In [566]:

```
# Shape
print("Shape:", rt_reviews.shape)
print('************************************************************************')
print('************************************************************************')

# Displaying the dataset info
print("Dataset info:")
rt_reviews.info()

print('************************************************************************')
print('************************************************************************')

# Displaying the first five elements of the dataset
print("First five elements:")
print(rt_reviews.head())

print('************************************************************************')
print('************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(rt_reviews.describe())
print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(rt_reviews.isnull().sum())
print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(rt_reviews[rt_reviews.duplicated()])
print('************************************************************************')
print('************************************************************************')
```

```
Shape: (54432, 8)
************************************************************************
************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54432 entries, 0 to 54431
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          54432 non-null  int64
 1   review      48869 non-null  object
 2   rating      40915 non-null  object
 3   fresh       54432 non-null  object
 4   critic      51710 non-null  object
```

```
 5   top_critic  54432 non-null  int64
 6   publisher   54123 non-null  object
 7   date        54432 non-null  object
dtypes: int64(2), object(6)
memory usage: 3.3+ MB
*************************************************************************
*************************************************************************
First five elements:
   id                                             review rating   fresh  \
0   3  A distinctly gallows take on contemporary fina...    3/5   fresh
1   3  It's an allegory in search of a meaning that n...    NaN  rotten
2   3  ... life lived in a bubble in financial dealin...    NaN   fresh
3   3  Continuing along a line introduced in last yea...    NaN   fresh
4   3             ... a perverse twist on neorealism...     NaN   fresh

           critic  top_critic         publisher               date
0      PJ Nabarro           0   Patrick Nabarro  November 10, 2018
1  Annalee Newitz           0           io9.com       May 23, 2018
2    Sean Axmaker           0  Stream on Demand    January 4, 2018
3   Daniel Kasman           0              MUBI  November 16, 2017
4             NaN           0      Cinema Scope   October 12, 2017
*************************************************************************
*************************************************************************
Descriptive Statistics:
                 id    top_critic
count  54432.000000  54432.000000
mean    1045.706882      0.240594
std      586.657046      0.427448
min        3.000000      0.000000
25%      542.000000      0.000000
50%     1083.000000      0.000000
75%     1541.000000      0.000000
max     2000.000000      1.000000
*************************************************************************
*************************************************************************
Missing Data:
id                0
review         5563
rating        13517
fresh             0
critic         2722
top_critic        0
publisher       309
date              0
dtype: int64
*************************************************************************
*************************************************************************
Duplicate Rows:
         id                                             review rating   fresh  \
8129    304  Friends With Kids is a smart, witty and potty-...    NaN   fresh
14575   581                                                NaN  4.5/5   fresh
26226  1055                                                NaN    4/5   fresh
35162  1368                                                NaN    2/5  rotten
35166  1368                                                NaN    2/5  rotten
40567  1535                                                NaN    2/5  rotten
42381  1598  This tired, neutered action thriller won't cau...    2/5  rotten
49487  1843                                                NaN  0.5/5  rotten
49492  1843                                                NaN  0.5/5  rotten

      critic  top_critic        publisher               date
8129     NaN           0   Liverpool Echo      June 29, 2012
14575    NaN           0      Film Threat   December 6, 2005
26226    NaN           0      Film Threat   December 6, 2005
35162    NaN           0      Film Threat   December 6, 2005
35166    NaN           0      Film Threat   December 8, 2002
40567    NaN           0      Film Threat   December 6, 2005
42381    NaN           0  Empire Magazine  November 14, 2008
49487    NaN           0      Film Threat   December 6, 2005
49492    NaN           0      Film Threat   December 8, 2002
*************************************************************************
*************************************************************************
```

Conclusion:

The dataset contains information about 54432 movie reviews.

Columns like 'review', 'rating', 'critic', and 'publisher' have missing values.

The 'rating' column contains ratings in different formats (e.g., 3/5, 4.5/5, 4/5, 2/5, 0.5/5).

Some duplicate rows are present in the dataset.

In [567]:

```
# Handling missing values
rt_reviews['review'].fillna("No review available", inplace=True)
rt_reviews['critic'].fillna("Unknown", inplace=True)
rt_reviews['publisher'].fillna("Unknown", inplace=True)
rt_reviews.dropna(subset=['rating'], inplace=True)

# Converting 'rating' column to string
rt_reviews['rating'] = rt_reviews['rating'].astype(str)

rt_reviews['rating'] = rt_reviews['rating'].str.replace('/5', '')

# Converting ratings to numeric format
rt_reviews['rating'] = pd.to_numeric(rt_reviews['rating'], errors='coerce')

# Dropping rows with missing 'rating' values
rt_reviews.dropna(subset=['rating'], inplace=True)


# Removing duplicate rows
rt_reviews.drop_duplicates(inplace=True)
```

In [568]:

```
# Verifying changes
print(rt_reviews.info())
print(rt_reviews.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 17785 entries, 0 to 54431
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   id          17785 non-null  int64  
 1   review      17785 non-null  object 
 2   rating      17785 non-null  float64
 3   fresh       17785 non-null  object 
 4   critic      17785 non-null  object 
 5   top_critic  17785 non-null  int64  
 6   publisher   17785 non-null  object 
 7   date        17785 non-null  object 
dtypes: float64(1), int64(2), object(5)
memory usage: 1.2+ MB
None
id            0
review        0
rating        0
fresh         0
critic        0
top_critic    0
publisher     0
date          0
dtype: int64
```

4. The Movie Database (TMDB) -

A database for movies and TV shows, providing detailed information on titles and crew.

In [569]:

```
# Shape
print("Shape:", tmdb_movies.shape)

print('*************************************************************************')
print('*************************************************************************')

# Displaying the dataset info
print("Dataset info:")
tmdb_movies.info()

print('*************************************************************************')
print('*************************************************************************')

print("First five elements:")
print(tmdb_movies.head())

print('*************************************************************************')
print('*************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(tmdb_movies.describe())

print('*************************************************************************')
print('*************************************************************************')

# Missing data
print("Missing Data:")
print(tmdb_movies.isnull().sum())

print('*************************************************************************')
print('*************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(tmdb_movies[tmdb_movies.duplicated()])

print('*************************************************************************')
print('*************************************************************************')
```

```
Shape: (26517, 10)
*************************************************************************
*************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  int64  
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
*************************************************************************
*************************************************************************
First five elements:
   Unnamed: 0            genre_ids     id original_language  \
0           0      [12, 14, 10751]  12444                en   
1           1  [14, 12, 16, 10751]  10191                en   
2           2        [12, 28, 878]  10138                en   
3           3      [16, 35, 10751]    862                en   
4           4        [28, 878, 12]  27205                en   
```

```
                                     original_title  popularity release_date  \
0  Harry Potter and the Deathly Hallows: Part 1      33.533   2010-11-19
1                      How to Train Your Dragon      28.734   2010-03-26
2                                    Iron Man 2      28.515   2010-05-07
3                                     Toy Story      28.005   1995-11-22
4                                     Inception      27.920   2010-07-16

                                          title  vote_average  vote_count
0  Harry Potter and the Deathly Hallows: Part 1           7.7       10788
1                      How to Train Your Dragon           7.7        7610
2                                    Iron Man 2           6.8       12368
3                                     Toy Story           7.9       10174
4                                     Inception           8.3       22186
*************************************************************************
*************************************************************************
Descriptive Statistics:
        Unnamed: 0             id    popularity  vote_average    vote_count
count  26517.00000   26517.000000  26517.000000  26517.000000  26517.000000
mean   13258.00000  295050.153260      3.130912      5.991281    194.224837
std     7654.94288  153661.615648      4.355229      1.852946    960.961095
min        0.00000      27.000000      0.600000      0.000000      1.000000
25%     6629.00000  157851.000000      0.600000      5.000000      2.000000
50%    13258.00000  309581.000000      1.374000      6.000000      5.000000
75%    19887.00000  419542.000000      3.694000      7.000000     28.000000
max    26516.00000  608444.000000     80.773000     10.000000  22186.000000
*************************************************************************
*************************************************************************
Missing Data:
Unnamed: 0           0
genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
vote_average         0
vote_count           0
dtype: int64
*************************************************************************
*************************************************************************
Duplicate Rows:
Empty DataFrame
Columns: [Unnamed: 0, genre_ids, id, original_language, original_title, popularity, relea
se_date, title, vote_average, vote_count]
Index: []
*************************************************************************
*************************************************************************
```

Conclusions

There are no missing values in any of the columns. No duplicate rows were found in the dataset The 'release_date' column is in string format and should be converted to datetime for further analysis..

This dataset contains information about movie release dates, titles, production budgets, domestic gross revenues, and worldwide gross revenues.

In [570]:

```
# Converting 'release_date' to datetime format
tmdb_movies['release_date'] = pd.to_datetime(tmdb_movies['release_date'])

# Droping unnecessary columns
tmdb_movies.drop(columns=['Unnamed: 0', 'genre_ids'], inplace=True)
```

In [571]:

```
# Verifying changes
```

```
print(tmdb_movies.info())
print(tmdb_movies.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 26517 non-null  int64
 1   original_language  26517 non-null  object
 2   original_title     26517 non-null  object
 3   popularity         26517 non-null  float64
 4   release_date       26517 non-null  datetime64[ns]
 5   title              26517 non-null  object
 6   vote_average       26517 non-null  float64
 7   vote_count         26517 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(2), object(3)
memory usage: 1.6+ MB
None
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
vote_average         0
vote_count           0
dtype: int64
```

In [572]:

```
tmdb_movies.head()
```

Out[572]:

	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count
0	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186

5. The Numbers:

Provides data on movie budgets, revenues, and production costs.

In [573]:

```
# Shape
print("Shape:", tn_movie_budgets.shape)

print('**********************************************************************')
print('**********************************************************************')

# Displaying the dataset info
print("Dataset info:")
tn_movie_budgets.info()

print('**********************************************************************')
print('**********************************************************************')

print("First five elements:")
print(tn_movie_budgets.head())
```

```
print('************************************************************************')
print('************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(tn_movie_budgets.describe())

print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(tn_movie_budgets.isnull().sum())

print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(tn_movie_budgets[tn_movie_budgets.duplicated()])

print('************************************************************************')
print('************************************************************************')
```

```
Shape: (5782, 6)
************************************************************************
************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
************************************************************************
************************************************************************
First five elements:
   id  release_date                                        movie  \
0   1  Dec 18, 2009                                       Avatar   
1   2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides   
2   3   Jun 7, 2019                                 Dark Phoenix   
3   4   May 1, 2015                      Avengers: Age of Ultron   
4   5  Dec 15, 2017            Star Wars Ep. VIII: The Last Jedi   

  production_budget domestic_gross worldwide_gross  
0      $425,000,000   $760,507,625  $2,776,345,279  
1      $410,600,000   $241,063,875  $1,045,663,875  
2      $350,000,000    $42,762,350    $149,762,350  
3      $330,600,000   $459,005,868  $1,403,013,963  
4      $317,000,000   $620,181,382  $1,316,721,747  
************************************************************************
************************************************************************
Descriptive Statistics:
                id
count  5782.000000
mean     50.372363
std      28.821076
min       1.000000
25%      25.000000
50%      50.000000
75%      75.000000
max     100.000000
************************************************************************
************************************************************************
```

```
Missing Data:
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
************************************************************************
************************************************************************
Duplicate Rows:
Empty DataFrame
Columns: [id, release_date, movie, production_budget, domestic_gross, worldwide_gross]
Index: []
************************************************************************
************************************************************************
```

Conclusions

There are no missing values in any of the columns. No duplicate rows were found in the dataset.

The dataset contains information about movie releases, including movie titles, release dates, production budgets, domestic and worldwide gross revenues.

The 'release_date' column is in string format and should be converted to datetime

The 'production_budget', 'domestic_gross', and 'worldwide_gross' columns are currently in string format and should be converted to numeric

The 'production_budget' column represents the budget allocated for producing the movie, which can be compared with the gross revenue to assess profitability.

In [574]:

```
# Converting 'release_date' to datetime format
tn_movie_budgets['release_date'] = pd.to_datetime(tn_movie_budgets['release_date'])

monetary_columns = ['production_budget', 'domestic_gross', 'worldwide_gross']
for col in monetary_columns:
    tn_movie_budgets[col] = tn_movie_budgets[col].astype(str).str.replace('$', '').str.r
eplace(',', '').astype(float)

# Calculating profit
tn_movie_budgets['profit'] = tn_movie_budgets['worldwide_gross'] - tn_movie_budgets['pro
duction_budget']

# Filling NaN values with 0
tn_movie_budgets.fillna(0, inplace=True)
```

In [575]:

```
# Display the DataFrame
print(tn_movie_budgets)
```

```
      id release_date                                        movie  \
0      1   2009-12-18                                       Avatar
1      2   2011-05-20  Pirates of the Caribbean: On Stranger Tides
2      3   2019-06-07                                 Dark Phoenix
3      4   2015-05-01                      Avengers: Age of Ultron
4      5   2017-12-15            Star Wars Ep. VIII: The Last Jedi
...   ..          ...                                          ...
5777  78   2018-12-31                                       Red 11
5778  79   1999-04-02                                    Following
5779  80   2005-07-13                Return to the Land of Wonders
5780  81   2015-09-29                         A Plague So Pleasant
5781  82   2005-08-05                            My Date With Drew

      production_budget  domestic_gross  worldwide_gross        profit
0           425000000.0     760507625.0     2.776345e+09  2.351345e+09
1           410600000.0     241063875.0     1.045664e+09  6.350639e+08
```

```
1          410600000.0   241063875.0   1.045664e+09  6.350639e+08
2          350000000.0    42762350.0   1.497624e+08 -2.002376e+08
3          330600000.0   459005868.0   1.403014e+09  1.072414e+09
4          317000000.0   620181382.0   1.316722e+09  9.997217e+08
...                ...           ...            ...           ...
5777            7000.0           0.0   0.000000e+00 -7.000000e+03
5778            6000.0       48482.0   2.404950e+05  2.344950e+05
5779            5000.0        1338.0   1.338000e+03 -3.662000e+03
5780            1400.0           0.0   0.000000e+00 -1.400000e+03
5781            1100.0      181041.0   1.810410e+05  1.799410e+05

[5782 rows x 7 columns]
```

In [576]:

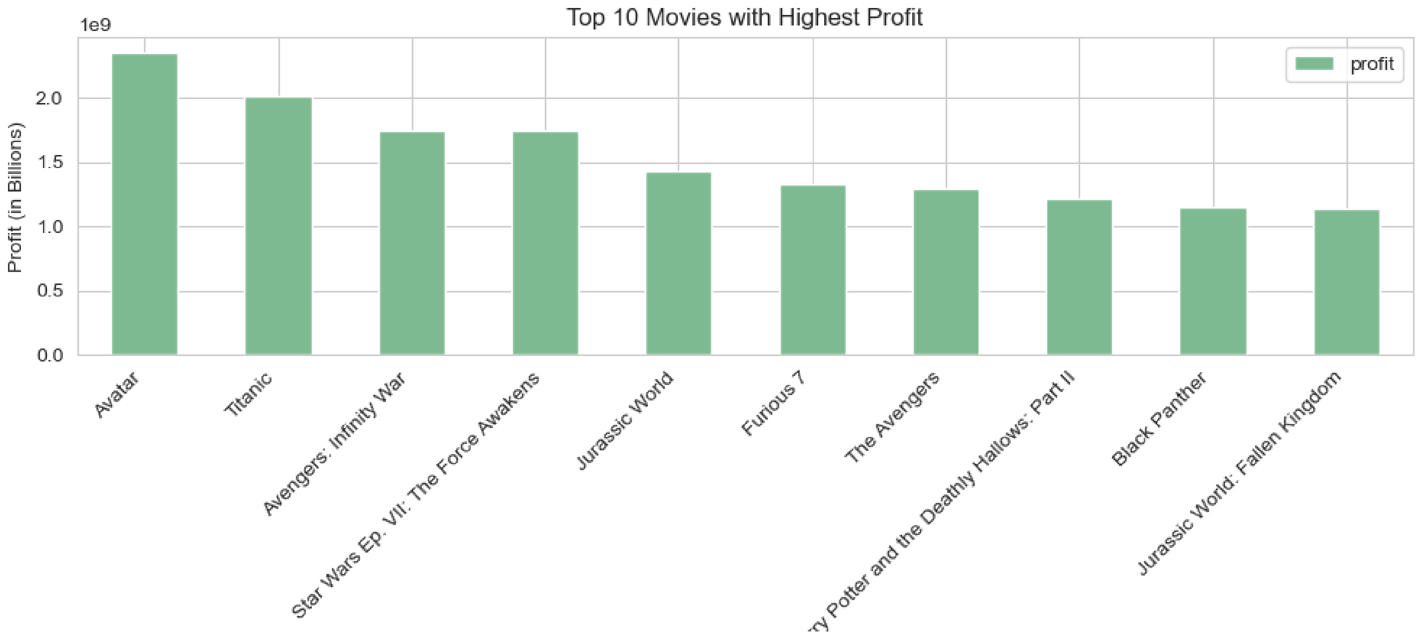
```python
# Confirming the changes
tn_movie_budgets.head()
```

Out[576]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	profit
0	1	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09	2.351345e+09
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	6.350639e+08
2	3	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	-2.002376e+08
3	4	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	1.072414e+09
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	9.997217e+08

In [577]:

```python
# Plot of top 10 movies with highest profit
fig, ax = plt.subplots(figsize=(10, 5))
most_profitable_movies.plot(kind='bar', x='movie', y='profit', ax=ax)

colors = sns.color_palette("YlOrBr", len(most_profitable_movies))

ax.set_title('Top 10 Movies with Highest Profit')
ax.set_xlabel('Movie')
ax.set_ylabel('Profit (in Billions)')

ax.set_xticklabels(most_profitable_movies['movie'], rotation=45, ha='right')
plt.tight_layout()
plt.savefig('Top 10 Movies with Highest Profit.png')

# Show the plot
plt.show()
```



Avatar and Titanic are the best movies profit-wise

In [578]:

```
extract_year = lambda date_str: date_str.year

# Applying the function to the release_date column
tn_movie_budgets['release_year'] = tn_movie_budgets['release_date'].apply(extract_year)
```

In [579]:

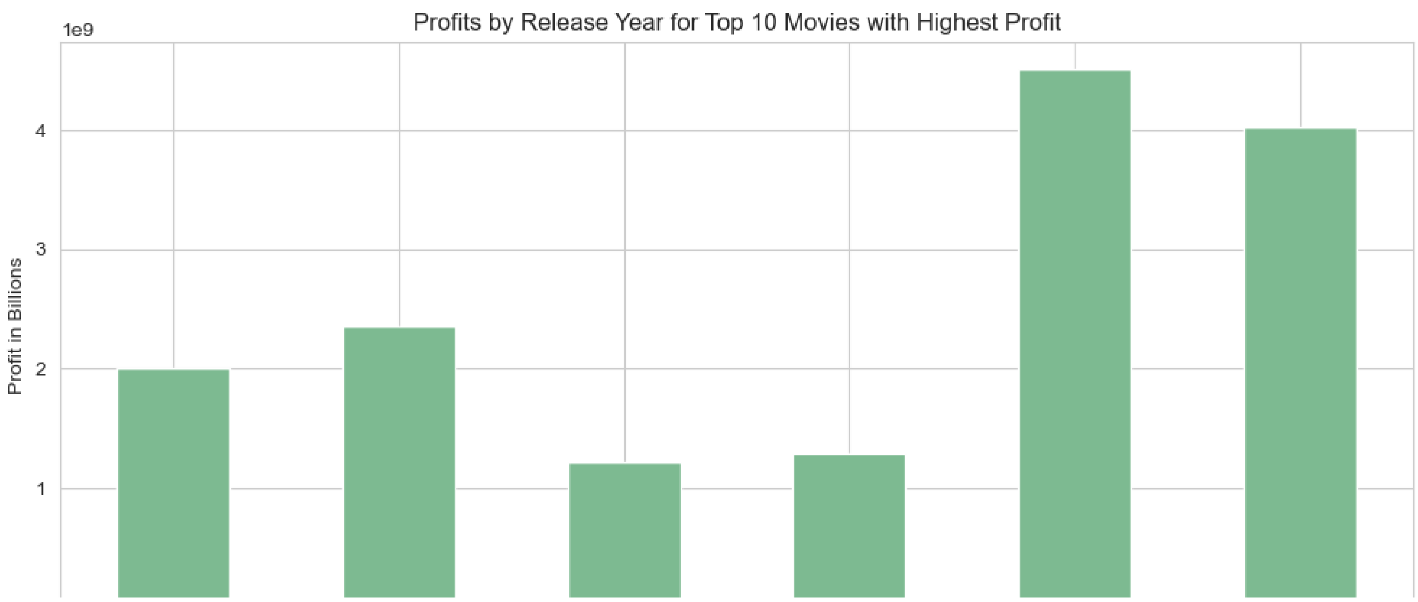
```
# Checking the results
tn_movie_budgets.head()
```

Out[579]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	profit	release_year
0	1	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09	2.351345e+09	2009
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	6.350639e+08	2011
2	3	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	-2.002376e+08	2019
3	4	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	1.072414e+09	2015
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	9.997217e+08	2017

In [580]:

```
# Grouping profits by release year for top 10 movies with highest profit
profit_by_year = most_profitable_movies.groupby(tn_movie_budgets['release_year'])['profi
t'].sum().nlargest(10)

# Plotting profits by release year for top 10 movies with highest profit arranged by year
desc
profit_by_year.sort_index().plot(kind='bar', figsize=(10, 5))

plt.title('Profits by Release Year for Top 10 Movies with Highest Profit')
plt.xlabel('Release Year')
plt.ylabel('Profit in Billions')
plt.tight_layout()
plt.savefig('Profits by Release Year for Top 10 Movies with Highest Profit.png')
plt.show()
```



This shows that the movie industry has become more and more profitable over the last few years.

Data Base Overview - IMDB

In [581]:

```
cur.execute("SELECT name FROM sqlite_master WHERE type='table';")
tables = cur.fetchall()
for table in tables:
    print(table[0])
```

```
movie_basics
directors
known_for
movie_akas
movie_ratings
persons
principals
writers
```

movie_basics table

In [582]:

```
movie_basics = pd.read_sql_query("SELECT * FROM movie_basics", conn)
movie_basics
```

Out[582]:

	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
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [583]:

```
movie_basics = pd.read_sql_query("SELECT * FROM movie_basics", conn)
movie_basics
```

Out[583]:

	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
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [584]:

```
# Shape
print("Shape:", movie_basics.shape)

print('************************************************************************')
print('************************************************************************')

# Displaying the dataset info
print("Dataset info:")
movie_basics.info()

print('************************************************************************')
print('************************************************************************')

print("First five elements:")
print(movie_basics.head())

print('************************************************************************')
print('************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(movie_basics.describe())

print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(movie_basics.isnull().sum())

print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(movie_basics.duplicated().sum())

print('************************************************************************')
print('************************************************************************')
```

```
Shape: (146144, 6)
************************************************************************
************************************************************************
Dataset info:
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         146144 non-null  object 
 1   primary_title    146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
************************************************************************
************************************************************************
First five elements:
    movie_id                    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  
************************************************************************
************************************************************************
Descriptive Statistics:
          start_year  runtime_minutes
count  146144.000000    114405.000000
mean     2014.621798        86.187247
std         2.733583       166.360590
min      2010.000000         1.000000
25%      2012.000000        70.000000
50%      2015.000000        87.000000
75%      2017.000000        99.000000
max      2115.000000     51420.000000
************************************************************************
************************************************************************
Missing Data:
movie_id               0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
************************************************************************
************************************************************************
Duplicate Rows:
0
************************************************************************
************************************************************************
```

Conclusions

The data had no duplicates

The data has missing values, dropping all the collumns with NaN values seems okay.

In [585]:

```
movie_basics.dropna(inplace=True)
```

In [586]:

```
#Checking the changes
print(movie_basics.isnull().sum())
```

```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

In [587]:

```
movie_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 112232 entries, 0 to 146139
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         112232 non-null  object 
 1   primary_title    112232 non-null  object 
 2   original_title   112232 non-null  object 
 3   start_year       112232 non-null  int64  
 4   runtime_minutes  112232 non-null  float64
 5   genres           112232 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.0+ MB
```

movie_ratings

In [588]:

```
movie_ratings = pd.read_sql_query("SELECT * FROM movie_ratings", conn)
movie_ratings
```

Out[588]:

	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
...	...	...	...
73851	tt9805820	8.1	25
73852	tt9844256	7.5	24
73853	tt9851050	4.7	14
73854	tt9886934	7.0	5
73855	tt9894098	6.3	128

73856 rows × 3 columns

In [589]:

```
# Shape
print("Shape:", movie_ratings.shape)

print('***********************************************************************')
print('***********************************************************************')

# Displaying the dataset info
print("Dataset info:")
```

```
movie_ratings.info()

print('************************************************************************')
print('************************************************************************')

print("First five elements:")
print(movie_ratings.head())

print('************************************************************************')
print('************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(movie_ratings.describe())

print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(movie_ratings.isnull().sum())

print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(movie_ratings.duplicated().sum())

print('************************************************************************')
print('************************************************************************')
```

```
Shape: (73856, 3)
************************************************************************
************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   movie_id       73856 non-null  object 
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
************************************************************************
************************************************************************
First five elements:
     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
************************************************************************
************************************************************************
Descriptive Statistics:
       averagerating      numvotes
count   73856.000000  7.385600e+04
mean        6.332729  3.523662e+03
std         1.474978  3.029402e+04
min         1.000000  5.000000e+00
25%         5.500000  1.400000e+01
50%         6.500000  4.900000e+01
75%         7.400000  2.820000e+02
max        10.000000  1.841066e+06
************************************************************************
************************************************************************
Missing Data:
movie_id         0
averagerating    0
```

```
numvotes        0
dtype: int64
**************************************************************************
**************************************************************************
Duplicate Rows:
0
**************************************************************************
**************************************************************************
```

Conclusions

The data has no missing or duplicate values.

In [590]:

```
#Combining the movie_basics and movie_ratings tables

movie_basics_combined = pd.merge(movie_basics,movie_ratings,how='left')
movie_basics_combined
```

Out[590]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	77.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.2	43.0
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	6.9	4517.0
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5	119.0
4	tt0111414	A Thin Life	A Thin Life	2018	75.0	Comedy	NaN	NaN
...	...	...	...	...	...	...	...	...
112227	tt9916160	Drømmeland	Drømmeland	2019	72.0	Documentary	6.5	11.0
112228	tt9916170	The Rehearsal	O Ensaio	2019	51.0	Drama	NaN	NaN
112229	tt9916186	Illenau - die Geschichte einer ehemaligen Heil...	Illenau - die Geschichte einer ehemaligen Heil...	2017	84.0	Documentary	NaN	NaN
112230	tt9916190	Safeguard	Safeguard	2019	90.0	Drama,Thriller	NaN	NaN
112231	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama	NaN	NaN

112232 rows × 8 columns

In [591]:

```
# Shape
print("Shape:", movie_basics_combined.shape)

print('**************************************************************************')
print('**************************************************************************')

# Displaying the dataset info
print("Dataset info:")
movie_basics_combined.info()

print('**************************************************************************')
print('**************************************************************************')
```

```python
print("First five elements:")
print(movie_basics_combined.head())

print('************************************************************************')
print('************************************************************************')

# Descriptive statistics
print("Descriptive Statistics:")
print(movie_basics_combined.describe())

print('************************************************************************')
print('************************************************************************')

# Missing data
print("Missing Data:")
print(movie_basics_combined.isnull().sum())

print('************************************************************************')
print('************************************************************************')

# Duplicates
print("Duplicate Rows:")
print(movie_basics_combined.duplicated().sum())

print('************************************************************************')
print('************************************************************************')
```

```
Shape: (112232, 8)
************************************************************************
************************************************************************
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112232 entries, 0 to 112231
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         112232 non-null  object 
 1   primary_title    112232 non-null  object 
 2   original_title   112232 non-null  object 
 3   start_year       112232 non-null  int64  
 4   runtime_minutes  112232 non-null  float64
 5   genres           112232 non-null  object 
 6   averagerating    65720 non-null   float64
 7   numvotes         65720 non-null   float64
dtypes: float64(3), int64(1), object(4)
memory usage: 6.9+ MB
************************************************************************
************************************************************************
First five elements:
    movie_id                    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  tt0100275         The Wandering Soap Opera       La Telenovela Errante   
4  tt0111414                      A Thin Life                 A Thin Life   

   start_year  runtime_minutes                genres  averagerating  numvotes  
0        2013            175.0    Action,Crime,Drama            7.0      77.0  
1        2019            114.0       Biography,Drama            7.2      43.0  
2        2018            122.0                 Drama            6.9    4517.0  
3        2017             80.0  Comedy,Drama,Fantasy            6.5     119.0  
4        2018             75.0                Comedy            NaN       NaN  
************************************************************************
************************************************************************
Descriptive Statistics:
          start_year  runtime_minutes  averagerating      numvotes
count  112232.000000    112232.000000   65720.000000  6.572000e+04
mean     2014.402078        86.261556       6.320902  3.954674e+03
std         2.639042       167.896646       1.458878  3.208823e+04
min      2010.000000         1.000000       1.000000  5.000000e+00
25%      2012.000000        70.000000       5.500000  1.600000e+01
```

```
50%      2014.000000      87.000000      6.500000  6.200000e+01
75%      2017.000000      99.000000      7.300000  3.520000e+02
max      2022.000000   51420.000000     10.000000  1.841066e+06
*************************************************************************
*************************************************************************
Missing Data:
movie_id               0
primary_title          0
original_title         0
start_year             0
runtime_minutes        0
genres                 0
averagerating      46512
numvotes           46512
dtype: int64
*************************************************************************
*************************************************************************
Duplicate Rows:
0
*************************************************************************
*************************************************************************
```

Conclusion

There are missing data in averagerating and numvotes

In [592]:

```
# dropping NaN Values
movie_basics_combined.dropna(inplace=True)
```

In [593]:

```
movie_basics_combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 65720 entries, 0 to 112227
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   movie_id         65720 non-null  object 
 1   primary_title    65720 non-null  object 
 2   original_title   65720 non-null  object 
 3   start_year       65720 non-null  int64  
 4   runtime_minutes  65720 non-null  float64
 5   genres           65720 non-null  object 
 6   averagerating    65720 non-null  float64
 7   numvotes         65720 non-null  float64
dtypes: float64(3), int64(1), object(4)
memory usage: 4.5+ MB
```

In [594]:

```
# Confirming for missing values
movie_basics_combined.isnull().sum()
```

Out[594]:

```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
dtype: int64
```

In [595]:

```
movie_basics_combined.head()
```

Out[595]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	77.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.2	43.0
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	6.9	4517.0
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5	119.0
5	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy	8.1	263.0

In [596]:

```
#genres with highest average ratings
highest_ratings = movie_basics_combined.groupby('genres')['averagerating'].mean().sort_v
alues(ascending=False).head(10)
highest_ratings
```

Out[596]:

```
genres
Comedy,Documentary,Fantasy      9.4
Documentary,Family,Musical      9.3
Game-Show                       9.0
Drama,Short                     8.8
Documentary,News,Sport          8.8
Documentary,News,Reality-TV     8.8
Action,Adventure,Musical        8.7
Biography,History,Music         8.5
Adventure,Crime                 8.5
Mystery,News,Thriller           8.4
Name: averagerating, dtype: float64
```

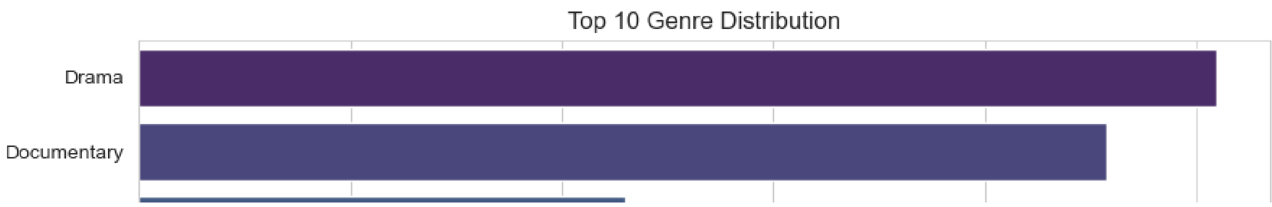
Visualizations and Recommendations

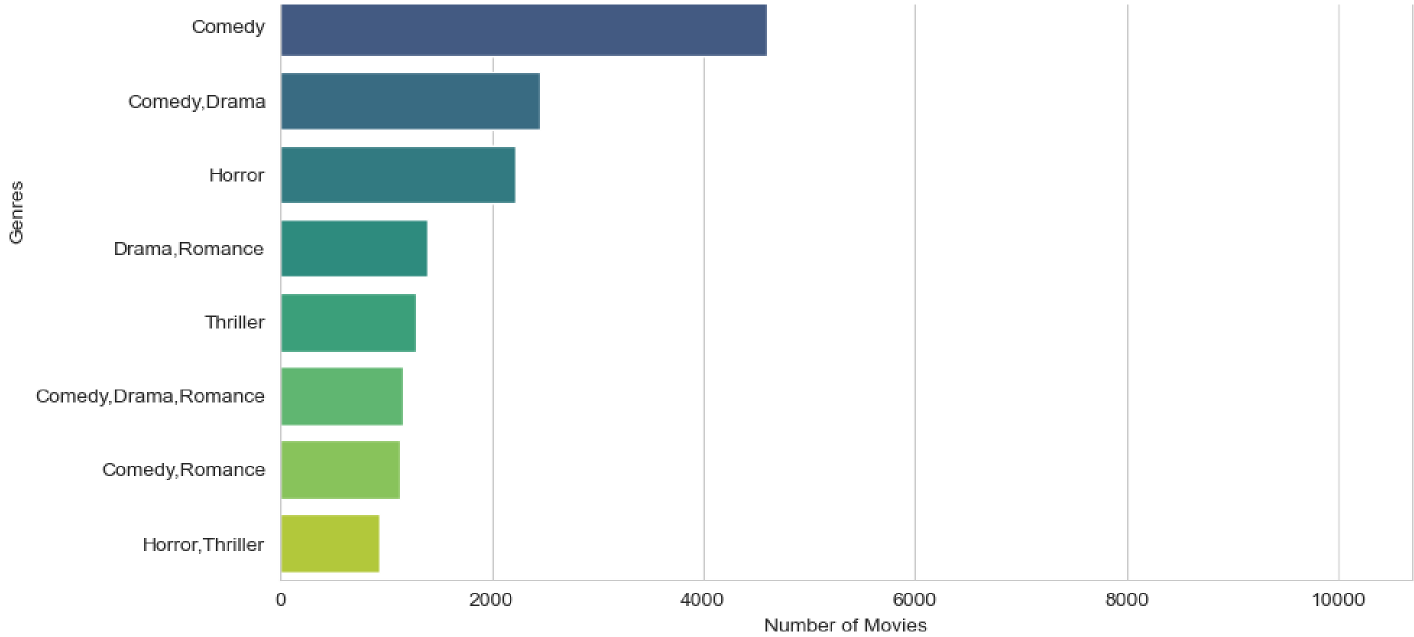
Genre Distribution:

A bar plot showing the distribution of movies across different genres can provide an overview of the most common genres in the dataset.

In [597]:

```
plt.figure(figsize=(10, 6))
top_genres = movie_basics_combined['genres'].value_counts().head(10)
sns.barplot(y=top_genres.index, x=top_genres.values, palette='viridis')
plt.title('Top 10 Genre Distribution')
plt.xlabel('Number of Movies')
plt.ylabel('Genres')
plt.tight_layout()
plt.savefig('Top 10 Genre Distribution.png')
plt.show()
```





Conclusions

This plot shows the distribution of the top 10 movie genres in terms of the number of movies produced.

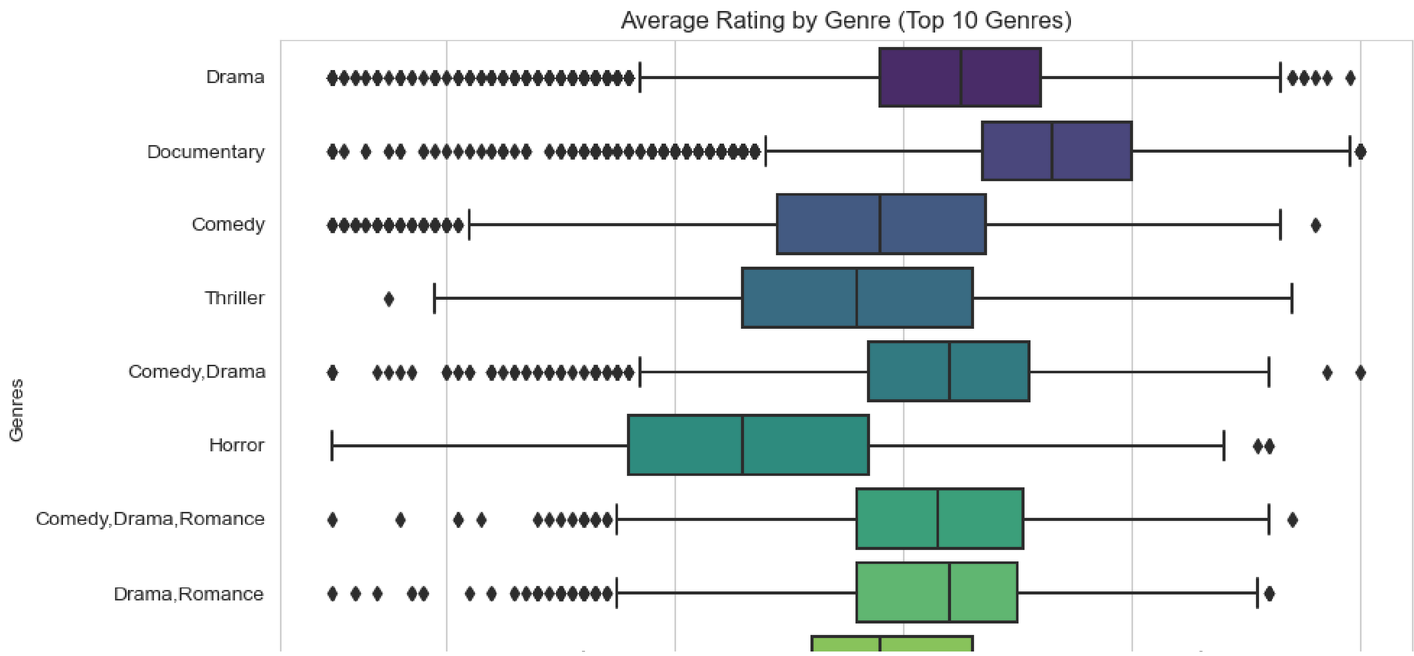
Recommendation: Microsoft should consider producing movies in genres that have a higher representation in the market. For example, genres like Drama,Documentaryy, andComedyn are popular and have a larger audience base.

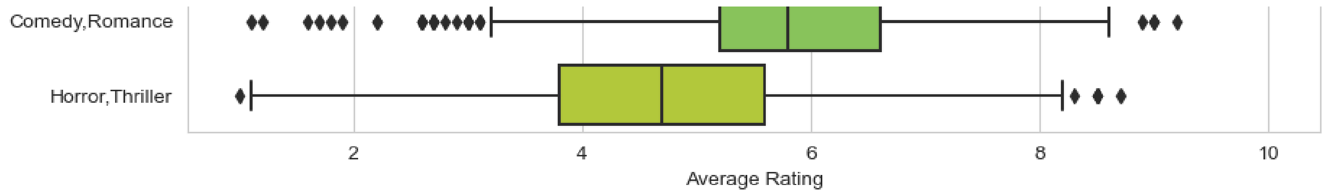
Average Rating by Genre

A box plot or bar plot showing the average rating of movies for each genre can help identify which genres tend to receive higher ratings.

In [598]:

```
plt.figure(figsize=(10, 6))
top_genres = movie_basics_combined['genres'].value_counts().head(10).index
sns.boxplot(data=movie_basics_combined[movie_basics_combined['genres'].isin(top_genres)],
y='genres', x='averagerating', palette='viridis')
plt.title('Average Rating by Genre (Top 10 Genres)')
plt.xlabel('Average Rating')
plt.ylabel('Genres')
plt.tight_layout()
plt.savefig('Average Rating by Genre (Top 10 Genres).png')
plt.show()
```





Conclusions

This box plot displays the distribution of average ratings for movies in the top 10 genres.

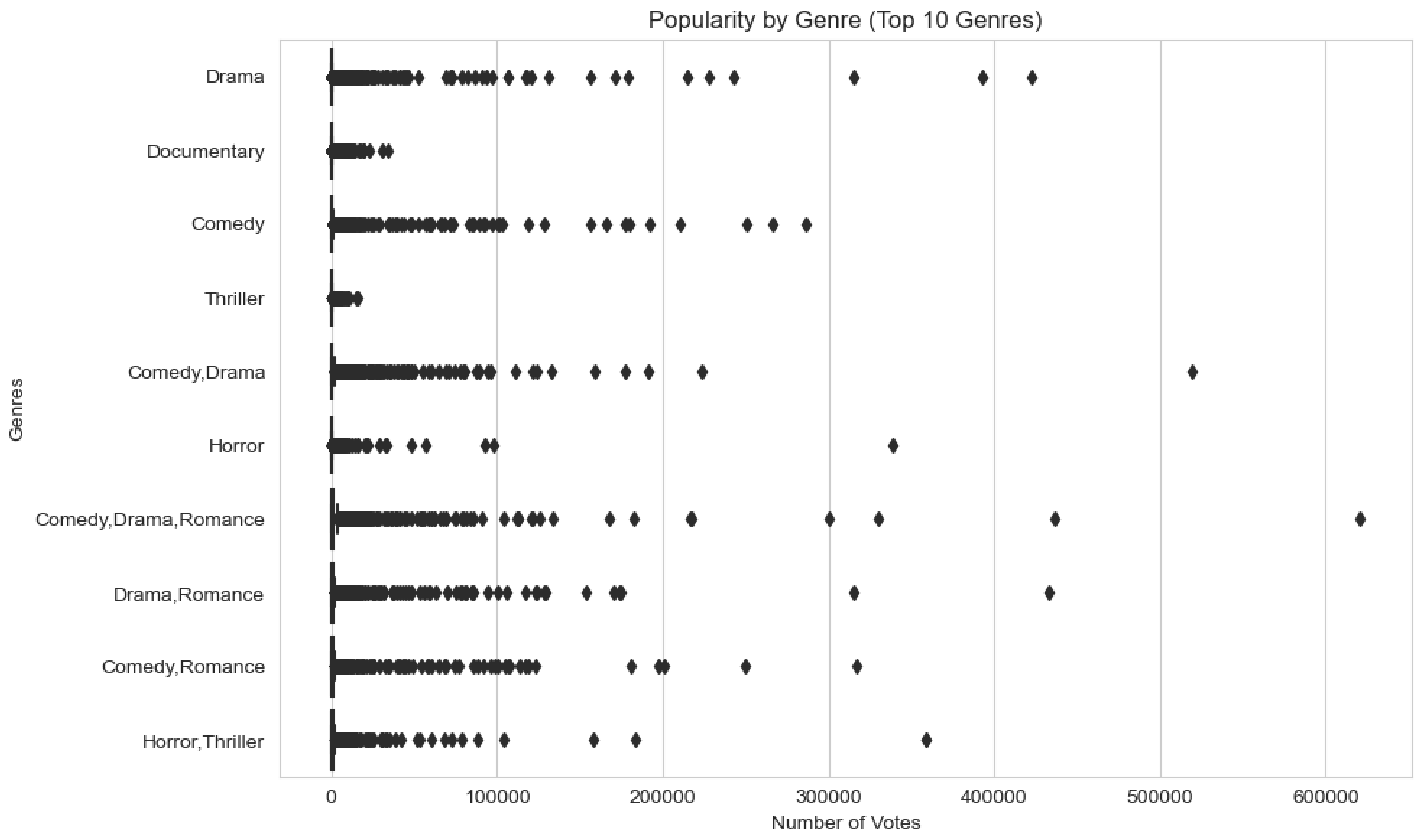
Recommendation: Microsoft should aim to produce movies in genres that tend to receive higher average ratings. For instance, genres like Documentary, Drama, and Comedy show higher average ratings, indicating audience liking.

Popularity by Genre:

A bar plot or box plot showing the popularity (e.g., number of votes) of movies for each genre can highlight which genres are more popular among viewers.

In [599]:

```
plt.figure(figsize=(10, 6))
top_genres = movie_basics_combined['genres'].value_counts().head(10).index
sns.boxplot(data=movie_basics_combined[movie_basics_combined['genres'].isin(top_genres)],
y='genres', x='numvotes', palette='viridis')
plt.title('Popularity by Genre (Top 10 Genres)')
plt.xlabel('Number of Votes')
plt.ylabel('Genres')
plt.tight_layout()
plt.savefig('Popularity by Genre (Top 10 Genres).png')
plt.show()
```



Conclusions

This box plot illustrates the distribution of popularity (measured by the number of votes) for movies in the top 10 genres.

Recommendation: Microsoft should focus on genres that have higher popularity,measuredd by the number of

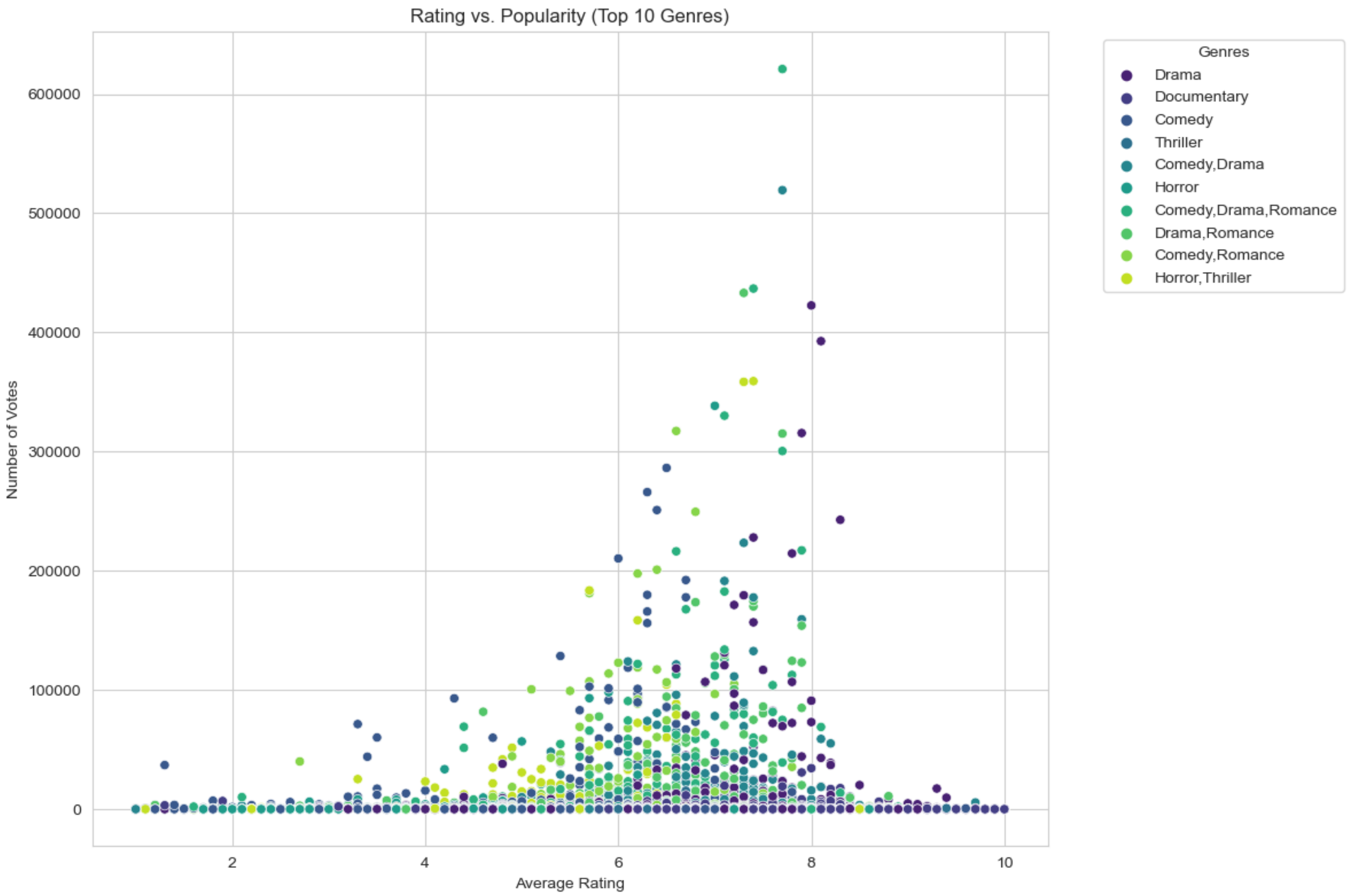
votes. Genres like Drama, Comedy, andRomancen attract more audience engagement and interest.

Rating vs. Popularity:

A scatter plot showing the relationship between average rating and popularity can help understand if there is a correlation between critical acclaim and audience engagement.

In [600]:

```
plt.figure(figsize=(12, 8))
top_genres = movie_basics_combined['genres'].value_counts().head(10).index
sns.scatterplot(data=movie_basics_combined[movie_basics_combined['genres'].isin(top_genre
s)], x='averagerating', y='numvotes', hue='genres', palette='viridis')
plt.title('Rating vs. Popularity (Top 10 Genres)')
plt.xlabel('Average Rating')
plt.ylabel('Number of Votes')
plt.legend(title='Genres', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.savefig('Rating vs. Popularity (Top 10 Genres).png')
plt.show()
```



Conclusions

This scatter plot explores the relationship between average ratings and popularity for movies in the top 10 genres.

Recommendation: Microsoft should prioritize genres that strike a balance between high ratings and popularity. For example, genres like Drama and Comedy tend to have both high ratings and popularity, suggesting that theyaree wel liked by theh audiences.

In [ ]: