Final Project Submission

Please fill out:

- Student name: Beatrice Wambui Kariuki
- Student pace: self paced / part time / full time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

Overview

Microsoft have decided to create a new movie studio.They require more insights into which types of genres are doing best at the box office. This project uses descriptive statistical analysis on data gathered from IMDb website to gain insight into which combination of genres will be most successful in the indusry. Three seperate datasets were used for this analysis to gain insights.Correlation was checked between average ratings and no of votes and there was no signficant relationship between the two.The number of movies released in a given year was also analysed to check the trends within the given time frame.The best performing studio was the Zeit studio with the highest total gross sales and this was due to them producing the best title the return of Xander cage which propelled the studio's success.Runtime through the start years was also analysed and there was a decline in the total runtimes over the years which is a recommendation to microsoft to know the ideal lenth of movies to be produced.The best genre combination was The best genre is Animation,Mystery,Thriller with an average rating of 9.20 whereas the best genre combination is Talk-Show with an average rating of 6.42.Other recommendations that Microsoft can adopt to be able to be successful would be to stay informed about the industry,to be creative in the production of movies as this would translate to both financial and ratings success.Aligning the genre combinations with the target audience.

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don’t know anything about creating movies.We are expected to explore the types of films that are currently doing the best at the box office.Then translate those findings into actionable insights that the head of Microsoft's new movie studio can use to help decide what type of films to create.The data analytical questions were based on the data being analysed to provide meaningful insights to Microsoft.The questions are key and form part of both movie and financial success.

Data Understanding

The datasets being used are .csv files namely title.basics,title.ratings and bom.movies_gross.All the files are available in the IMDB website.

In [2]:
```
#importing the relevant libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
pwd
```

Out[3]:
```
'C:\\Users\\lenovo\\Documents\\Flatiron1\\Phase_1\\dsc-phase1-project\\Micros
oft-Movie-Analysis'
```

Movie Gross Data

The movie gross dataset contains various records mainly the movie title,studio,the gross sales of the movie and the year of the movie

In [4]:
```
# load 'bom.movie_gross.csv' as a dataframe
# copy the file path to access the file since its in a different directory

movie_gross = pd.read_csv('./zippedData/bom.movie_gross.csv',encoding='utf-8')
movie_gross.head()
```

Out[4]:

	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

In [5]:
```
movie_gross.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
```

In [6]:
```
movie_gross.shape
```

Out[6]:
```
(3387, 5)
```

In [7]:
```
movie_gross.describe()
```

Out[7]:

	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

In [8]:
```
movie_gross['foreign_gross'].dtype
```

Out[8]:
```
dtype('O')
```

In [9]:
```
movie_gross['foreign_gross'].nunique()
```

Out[9]:
```
1204
```

The Title Basics Data

The title basic dataset contains records of the movie primary and original titles,the start year the runtime(mins),genres and tconst which is a unique identifier for titles in the IDMb database hence making referencing easier when searching for movies.

In [14]:
```
# load 'title.basics.csv' as a dataframe
# copy the file path to access the file since its in a different directory

title_basics = pd.read_csv('./zippedData/title.basics.csv',encoding='utf-8')
title_basics.head()
```

Out[14]:

	tconst	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

In [15]:
```
title_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146143 non-null  object 
 2   original_title   146122 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
```

In [16]:
```
title_basics.shape
```

Out[16]:
```
(146144, 6)
```

The Title Ratings Data

The title ratings dataset contains records of tconst,average rating and number of votes of movies

In [17]:
```
# load 'title.ratings.csv' as a dataframe
# copy the file path to access the file since its in a different directory


title_ratings= pd.read_csv('./zippedData/title.ratings.csv',encoding='utf-8')
title_ratings.head()
```

Out[17]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

In [18]:
```
title_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   tconst         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
```

In [19]:
```
title_ratings.shape
```

Out[19]: (73856, 3)

Data Cleaning

Movie_gross Data

In [20]:
```
movie_gross.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
```

In [21]:
```
# summing up the no of duplicate records in the dataset
movie_gross.duplicated().sum()
```

Out[21]: 0

In [22]:
```
# checking for sum of null values in the columns within in the dataset
movie_gross.isna().sum()
```

Out[22]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

Depending on the nature of your data,handling missing values,common strategies are: 1.Removing rows with missing values. 2.Filling missing values with a specific value (e.g., mean, median, or mode).

In [23]:
```
# checking for the records with the NaN values
missing_rows = movie_gross[movie_gross.isna().any(axis=1)]
missing_rows
```

Out[23]:

	title	studio	domestic_gross	foreign_gross	year
210	Outside the Law (Hors-la-loi)	NaN	96900.0	3300000	2010
222	Flipped	WB	1800000.0	NaN	2010
230	It's a Wonderful Afterlife	UTV	NaN	1300000	2010
254	The Polar Express (IMAX re-issue 2010)	WB	673000.0	NaN	2010
267	Tiny Furniture	IFC	392000.0	NaN	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

1380 rows × 5 columns

In [24]:
```
def missing_values(movie_gross):
    """A simple function to identify data has missing values"""
    # identify the total missing values per column
    miss = movie_gross.isna().sum()

    # calculate percentage of the missing values
    percentage_miss = (movie_gross.isna().sum() / len(movie_gross))

    # creating a dataframe 'missing'
    missing = pd.DataFrame({"Missing Values": miss, "Percentage": percentage_mi

    # remove values that are missing
    missing.drop(missing[missing["Percentage"] == 0].index, inplace = True)

    return missing


missing_data = missing_values(movie_gross)
missing_data
```

Out[24]:

	index	Missing Values	Percentage
1	studio	5	0.001476
2	domestic_gross	28	0.008267
3	foreign_gross	1350	0.398583

In [22]:
```
# Drop the the necessary columns based on the analysis needed.In this case i ch
# 1.The NaN values the column studioincase i would need to do an analysis of th
# 2.Fill the NaN values of both the domestic_gross and foreign_gross with the m
```

In [25]:
```
movie_gross['studio'].fillna('Unknown', inplace=True)
movie_gross.isna().sum()
```

Out[25]:
```
title              0
studio             0
domestic_gross    28
foreign_gross   1350
year               0
dtype: int64
```

In [26]:
```
# Before filling the NaN values with the mean of the column,its best to replace
movie_gross['domestic_gross'] = movie_gross['domestic_gross'].replace(np.nan, 0
movie_gross['domestic_gross'] = movie_gross['domestic_gross'].astype(int)
movie_gross['domestic_gross'] = movie_gross['domestic_gross'].replace(0,movie_g
movie_gross.isna().sum()
```

Out[26]:
```
title              0
studio             0
domestic_gross     0
foreign_gross   1350
year               0
dtype: int64
```

In [27]:
```
movie_gross['domestic_gross'].dtype
```

Out[27]: dtype('float64')

In [28]:
```
movie_gross['foreign_gross'] = movie_gross['foreign_gross'].str.replace(",",""
movie_gross['foreign_gross'] = movie_gross['foreign_gross'].replace(np.nan, 0)
movie_gross['foreign_gross'] = movie_gross['foreign_gross'].astype('float64')
movie_gross['foreign_gross'] = movie_gross['foreign_gross'].replace(0,movie_gro
movie_gross.isna().sum()
```

Out[28]:
```
title             0
studio            0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

In [29]:
```
movie_gross['foreign_gross'].dtype
```

Out[29]: dtype('float64')

In [30]:
```
movie_gross.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          3387 non-null   object
 2   domestic_gross  3387 non-null   float64
 3   foreign_gross   3387 non-null   float64
 4   year            3387 non-null   int64
dtypes: float64(2), int64(1), object(2)
memory usage: 132.4+ KB
```

In [31]:
```
movie_gross.describe()
```

Out[31]:

	domestic_gross	foreign_gross	year
count	3.387000e+03	3.387000e+03	3387.000000
mean	2.874388e+07	6.297790e+07	2013.958075
std	6.670498e+07	1.075504e+08	2.478141
min	1.000000e+02	6.000000e+02	2010.000000
25%	1.225000e+05	1.160000e+07	2012.000000
50%	1.400000e+06	4.502979e+07	2014.000000
75%	2.850821e+07	4.502979e+07	2016.000000
max	9.367000e+08	9.605000e+08	2018.000000

In [32]:
```
movie_gross['Total_gross'] = movie_gross['domestic_gross'] + movie_gross['fore:
movie_gross.head()
```

Out[32]:

	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

Title_basics Data

In [33]:
```
title_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146143 non-null  object 
 2   original_title   146122 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
```

In [34]:
```
title_basics.duplicated().sum()
```

Out[34]:
```
0
```

In [35]:
```
title_basics.isna().sum()
```

Out[35]:
```
tconst                 0
primary_title          1
original_title        22
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
```

In [36]:
```
# checking for the records with the NaN values
missingrows_title_basics =title_basics[title_basics.isna().any(axis=1)]
missingrows_title_basics
```

Out[36]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
6	tt0112502	Bigfoot	Bigfoot	2017	NaN	Horror,Thriller
8	tt0139613	O Silêncio	O Silêncio	2012	NaN	Documentary,History
16	tt0187902	How Huang Fei-hong Rescued the Orphan from the...	How Huang Fei-hong Rescued the Orphan from the...	2011	NaN	NaN
21	tt0250404	Godfather	Godfather	2012	NaN	Crime,Drama
...	...	...	...	...	...	...
146138	tt9916428	The Secret of China	The Secret of China	2019	NaN	Adventure,History,War
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	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

33912 rows × 6 columns

In [37]:
```
title_basics['primary_title'].fillna('No Primary Title',inplace=True)
title_basics['original_title'].fillna('No Original Title',inplace=True)
title_basics['genres'].fillna('No Genre',inplace=True)
```

In [38]:
```
title_basics.isna().sum()
```

Out[38]:
```
tconst                 0
primary_title          0
original_title         0
start_year             0
runtime_minutes    31739
genres                 0
dtype: int64
```

In [39]:
```
title_basics['runtime_minutes'].dtype
```

Out[39]:
```
dtype('float64')
```

In [40]:
```
title_basics['runtime_minutes'].unique()
```

Out[40]:
```
array([1.750e+02, 1.140e+02, 1.220e+02,       nan, 8.000e+01, 7.500e+01,
       8.300e+01, 8.200e+01, 1.360e+02, 1.000e+02, 1.800e+02, 8.900e+01,
       6.000e+01, 1.600e+02, 1.040e+02, 1.200e+02, 1.100e+02, 9.100e+01,
       1.340e+02, 4.400e+01, 4.000e+01, 9.700e+01, 5.900e+01, 4.500e+01,
       8.600e+01, 9.500e+01, 9.000e+01, 1.030e+02, 9.600e+01, 8.800e+01,
       1.020e+02, 1.090e+02, 9.900e+01, 8.400e+01, 1.240e+02, 9.800e+01,
       1.010e+02, 1.370e+02, 5.700e+01, 1.190e+02, 1.080e+02, 9.200e+01,
       2.800e+02, 8.700e+01, 1.320e+02, 1.810e+02, 1.440e+02, 1.070e+02,
       1.120e+02, 9.300e+01, 1.130e+02, 1.170e+02, 1.270e+02, 1.500e+02,
       1.150e+02, 1.050e+02, 1.410e+02, 1.280e+02, 8.500e+01, 5.600e+01,
       9.400e+01, 7.600e+01, 1.230e+02, 1.630e+02, 8.100e+01, 1.160e+02,
       1.060e+02, 1.290e+02, 1.390e+02, 7.700e+01, 1.250e+02, 1.610e+02,
       7.800e+01, 1.430e+02, 1.300e+02, 2.000e+02, 1.180e+02, 1.310e+02,
       1.690e+02, 7.900e+01, 6.700e+01, 1.210e+02, 7.400e+01, 1.110e+02,
       1.330e+02, 7.200e+01, 1.460e+02, 5.500e+01, 1.400e+02, 6.500e+01,
       1.260e+02, 7.000e+01, 5.200e+01, 5.100e+01, 6.300e+01, 6.100e+01,
       5.000e+01, 5.800e+01, 7.300e+01, 4.800e+01, 1.300e+01, 1.500e+01,
       6.600e+01, 6.800e+01, 1.580e+02, 5.300e+01, 7.100e+01, 1.420e+02,
       6.200e+01, 4.700e+01, 2.000e+01, 6.900e+01, 1.560e+02, 1.540e+02,
       2.700e+01, 1.100e+01, 8.000e+00, 1.480e+02, 4.900e+01, 6.400e+01,
       3.100e+01, 1.350e+02, 5.400e+01, 1.600e+01, 2.880e+02, 4.600e+01,
       1.970e+02, 1.450e+02, 1.510e+02, 2.080e+02, 2.220e+02, 4.300e+01,
       1.550e+02, 3.000e+01, 1.620e+02, 1.740e+02, 2.260e+02, 5.000e+00,
       4.000e+00, 2.600e+01, 1.200e+01, 1.920e+02, 2.600e+02, 1.650e+02,
       1.380e+02, 2.250e+02, 2.900e+01, 2.760e+02, 1.400e+01, 7.000e+00,
       1.000e+00, 3.300e+01, 1.490e+02, 3.400e+01, 9.000e+00, 1.520e+02,
       2.100e+01, 1.000e+01, 1.700e+01, 2.400e+01, 4.200e+01, 1.950e+02,
       6.000e+00, 1.470e+02, 2.000e+00, 1.780e+02, 3.000e+00, 1.760e+02,
       2.500e+01, 1.800e+01, 3.500e+02, 2.410e+02, 2.800e+01, 1.900e+01,
       2.960e+02, 3.880e+02, 2.700e+02, 2.150e+02, 2.200e+01, 2.570e+02,
       3.600e+01, 1.570e+02, 2.720e+02, 1.680e+02, 1.320e+03, 1.640e+02,
       1.700e+02, 1.720e+02, 1.530e+02, 3.100e+02, 3.900e+01, 1.830e+02,
       4.100e+01, 1.900e+02, 3.500e+01, 2.200e+02, 1.670e+02, 3.200e+01,
       1.590e+02, 2.640e+02, 3.700e+01, 2.500e+02, 3.800e+01, 3.450e+03,
       2.300e+01, 2.780e+02, 3.560e+02, 3.300e+02, 1.820e+02, 4.200e+03,
       1.800e+03, 1.960e+02, 3.640e+02, 1.730e+02, 7.610e+02, 1.850e+02,
       2.370e+02, 2.330e+02, 1.660e+02, 2.560e+02, 2.940e+02, 2.400e+03,
       5.000e+02, 1.669e+03, 6.050e+02, 8.400e+02, 2.400e+02, 3.210e+02,
       1.860e+02, 2.310e+02, 2.300e+02, 1.440e+03, 3.600e+02, 2.050e+02,
       1.710e+02, 2.010e+02, 3.200e+02, 2.100e+02, 2.180e+02, 2.440e+02,
       3.530e+02, 2.540e+02, 1.990e+02, 1.930e+02, 3.000e+02, 1.880e+02,
       2.240e+02, 3.240e+02, 1.840e+02, 2.360e+02, 1.770e+02, 7.240e+02,
       2.430e+02, 2.210e+02, 8.420e+02, 5.450e+02, 2.910e+02, 3.330e+02,
       1.980e+02, 1.910e+02, 2.350e+02, 3.170e+02, 1.440e+04, 2.380e+02,
       1.890e+02, 2.450e+02, 4.800e+02, 2.280e+02, 2.130e+02, 2.480e+02,
       4.040e+02, 2.110e+02, 2.320e+02, 4.500e+02, 4.160e+02, 2.270e+02,
       2.040e+02, 3.380e+02, 2.850e+02, 4.760e+02, 3.540e+02, 1.790e+02,
       3.190e+02, 3.630e+02, 3.820e+02, 7.800e+02, 2.020e+02, 3.410e+02,
       3.077e+03, 2.520e+02, 2.140e+02, 3.230e+02, 1.200e+03, 4.100e+02,
       2.090e+02, 3.250e+02, 6.000e+02, 3.340e+02, 2.170e+02, 2.580e+02,
       1.151e+03, 2.650e+02, 2.290e+02, 5.490e+02, 7.200e+02, 5.200e+02,
       4.850e+02, 6.000e+03, 2.905e+03, 5.460e+03, 2.030e+02, 4.980e+03,
       2.840e+02, 2.770e+02, 2.120e+02, 4.670e+02, 6.070e+02, 5.400e+02,
       3.790e+02, 3.040e+02, 2.060e+02, 2.740e+02, 3.020e+02, 2.160e+02,
       1.559e+03, 9.000e+02, 2.470e+02, 4.080e+03, 2.070e+02, 2.820e+02,
       1.870e+02, 3.460e+02, 2.550e+02, 1.940e+02, 4.060e+02, 7.880e+02,
       2.420e+02, 1.260e+03, 2.340e+02, 3.120e+02, 2.230e+02, 2.190e+02,
```

```
2.160e+03, 4.240e+02, 6.230e+02, 1.834e+03, 6.017e+03, 2.870e+02,
7.460e+02, 1.184e+03, 9.120e+02, 3.960e+02, 2.630e+02, 3.830e+02,
3.590e+02, 6.530e+02, 2.390e+02, 5.142e+04, 4.950e+02, 2.460e+02,
1.100e+03, 6.010e+02, 6.600e+02, 8.080e+02, 2.950e+02, 2.610e+02,
2.690e+02, 4.470e+02])
```

In [41]:
```
title_basics['runtime_minutes'].value_counts()
```

Out[41]:
```
runtime_minutes
90.0     7131
80.0     3526
85.0     2915
100.0    2662
95.0     2549
         ...
319.0       1
354.0       1
476.0       1
338.0       1
447.0       1
Name: count, Length: 367, dtype: int64
```

In [42]:
```
title_basics['runtime_minutes'] = title_basics['runtime_minutes'].replace(np.na
title_basics['runtime_minutes'] = title_basics['runtime_minutes'].astype('floa
title_basics['runtime_minutes'] = title_basics['runtime_minutes'].replace(0,ti
```

In [43]:
```
title_basics.isna().sum()
```

Out[43]:
```
tconst             0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

In [44]:
```
title_basics['runtime_minutes'].dtype
```

Out[44]:
```
dtype('float64')
```

Title_ratings Data

In [45]:
```
title_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   tconst         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
```

In [46]:
```
title_ratings.duplicated().sum()
```

Out[46]:
```
0
```

Merging Datasets

We will first merge the title_basics and title_ratings datasets to get combined_title dataset.The primary key is the tconst hence an inner join.

In [47]:
```
combined_title_df = title_basics.join(title_ratings,rsuffix="_ratings" , how=":
combined_title_df
```

Out[47]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.000000	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.000000	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.000000	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	67.469427	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.000000	Comedy,Drama,Fantasy
...	...	...	...	...	...	..
73851	tt4206656	MarchFourth Marching Band in China	MarchFourth Marching Band in China	2014	66.000000	Documentary,Music
73852	tt4206658	El Bumbún	El Bumbún	2014	85.000000	Drama
73853	tt4206724	70 Acres in Chicago: Cabrini Green	70 Acres in Chicago: Cabrini Green	2014	53.000000	Documentary,History,News
73854	tt4207014	Amante de lo ajeno	Amante de lo ajeno	2012	99.000000	Drama
73855	tt4207078	Nazar Palmus	Nazar Palmus	2016	67.469427	Fantasy,Romance,Thriller

73856 rows × 9 columns

In [48]:
```
combined_title_df['title_comparison'] = combined_title_df['primary_title'] == c
combined_title_df
```

Out[48]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.000000	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.000000	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.000000	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	67.469427	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.000000	Comedy,Drama,Fantasy
...	...	...	...	...	...	..
73851	tt4206656	MarchFourth Marching Band in China	MarchFourth Marching Band in China	2014	66.000000	Documentary,Music
73852	tt4206658	El Bumbún	El Bumbún	2014	85.000000	Drama
73853	tt4206724	70 Acres in Chicago: Cabrini Green	70 Acres in Chicago: Cabrini Green	2014	53.000000	Documentary,History,News
73854	tt4207014	Amante de lo ajeno	Amante de lo ajeno	2012	99.000000	Drama
73855	tt4207078	Nazar Palmus	Nazar Palmus	2016	67.469427	Fantasy,Romance,Thriller

73856 rows × 10 columns

In [49]:
```
combined_title_df['title_comparison'].value_counts('false')
```

Out[49]:
```
title_comparison
True     0.885629
False    0.114371
Name: proportion, dtype: float64
```

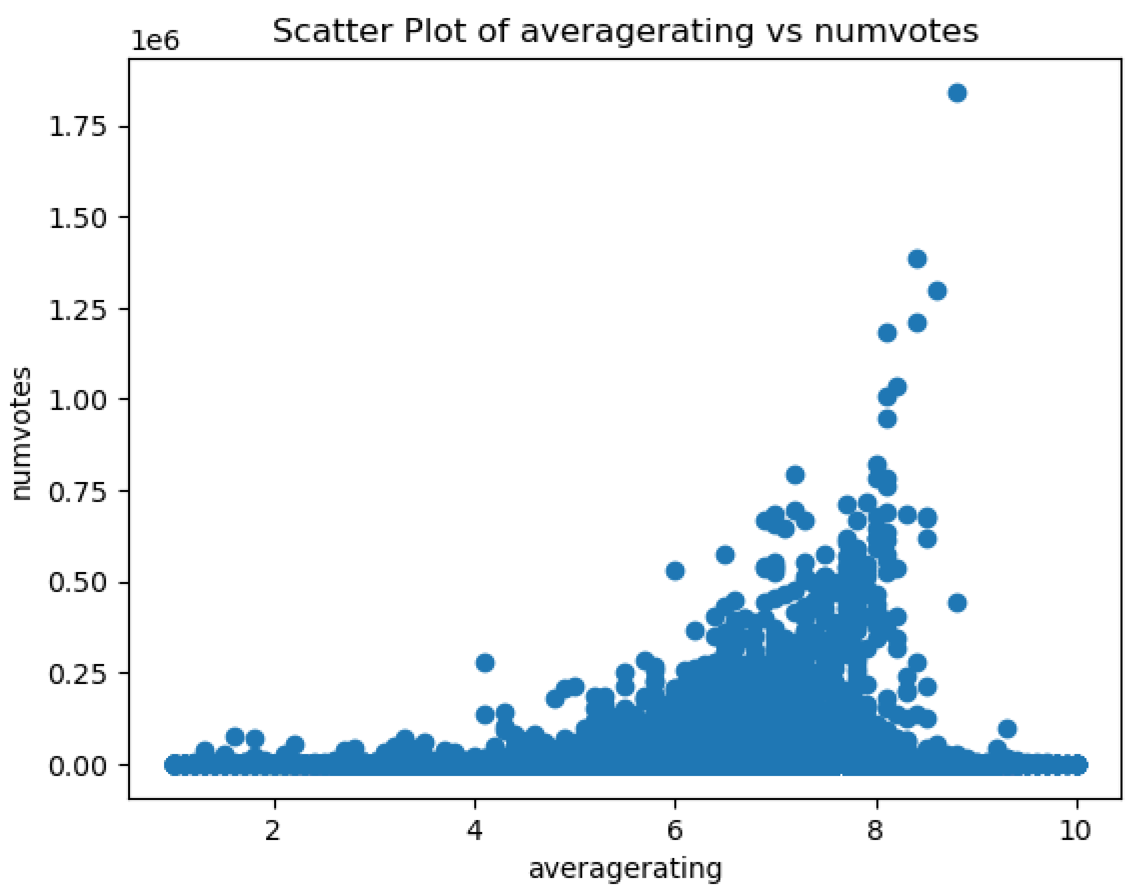
Analysis

In [50]:
```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

%matplotlib inline
```

In [51]:
```
# Calculate the correlation between 'averagerating' and 'numvotes'
correlation = combined_title_df['averagerating'].corr(combined_title_df['numvot

print("Correlation between 'averagerating' and 'numvotes':", correlation)
```

```
Correlation between 'averagerating' and 'numvotes': 0.04447809440198375
```

In [52]:
```
# Scatter plot
plt.scatter(combined_title_df['averagerating'], combined_title_df['numvotes'])
plt.title('Scatter Plot of averagerating vs numvotes')
plt.xlabel('averagerating')
plt.ylabel('numvotes')
plt.show()
```



The linear relationship between the two variables is weak since its close to 0.Hence there is no significant relationship between the average ratings and the num votes.Changes in average rating have almost no impact on the number of votes a title receives and vice versa.

Title Distribution Per Year

In [52]:
```
title = movie_gross.groupby('year').agg({'title': ['count']})
title.columns = ['Title Count']
title = title.sort_values('Title Count', ascending = False)
title.head()
```

Out[52]:

year	Title Count
2015	450
2016	436
2012	400
2011	399
2014	395

2015:Year with the highest number of movie releases at 450. This could be due to a variety of factors, including successful films, increased production, or strong market demand for movies during that year.

2018:Had the lowest number of movie releases at 308. This decline in the number of titles could be attributed to industry-specific factors, changes in audience preferences, or economic conditions affecting film production.

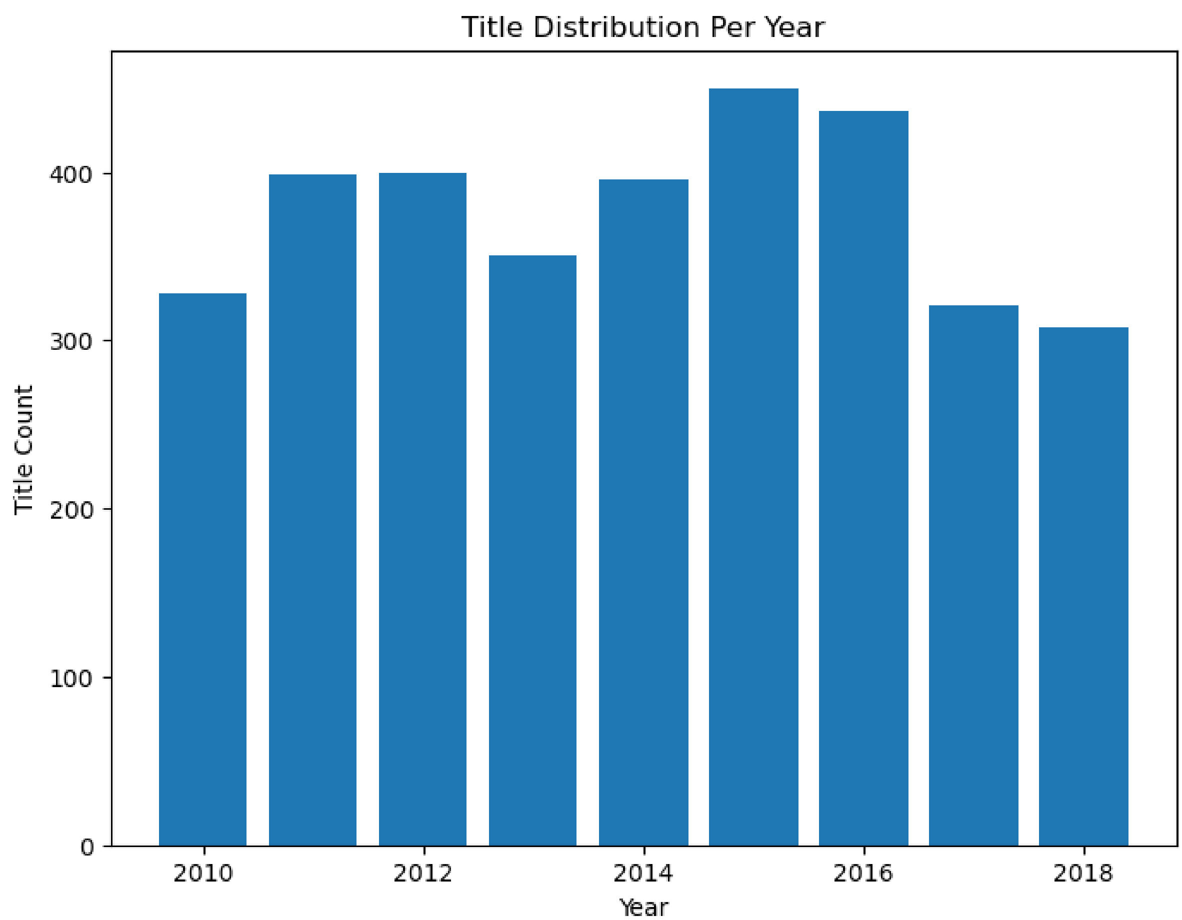
Consistency: Years like 2016, 2012, and 2011 also had relatively high numbers of movie releases, indicating a consistent level of film production during those years.

2010-2014: These years had varying numbers of movie releases but generally stayed above 300 titles, demonstrating a steady level of film production.

2013: Had a drop in the number of movie releases compared to the surrounding years. This could be due to factors specific to that year.

The film industry is adaptable to changes, with periods of growth, decline, and recovery. It responds to changing circumstances, audience preferences, and external factors like economic conditions.Its best to understand the dynamics of the film industry over the years.

In [53]:
```
# Create the plot
fig, ax = plt.subplots(figsize=(8, 6))

# Group the movie_gross DataFrame by year and count the titles per year
title_counts = movie_gross.groupby('year')['title'].count()

# Define labels for the x-axis
labels = title_counts.index

# Plot vertical bars of fixed width using the 'bar' function
ax.bar(labels, title_counts)

# Give a title to the bar graph and label the axes
ax.set_title("Title Distribution Per Year")
ax.set_ylabel("Title Count")
ax.set_xlabel("Year")

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



In [54]:
```
movie_gross.max()
```

Out[54]:
```
title              xXx: The Return of Xander Cage
studio                                      Zeit.
domestic_gross                        936700000.0
foreign_gross                         960500000.0
year                                         2018
Total_gross                          1518900000.0
dtype: object
```

In [55]:
```
movie_gross.min()
```

Out[55]:
```
title                '71
studio                3D
domestic_gross     100.0
foreign_gross      600.0
year                2010
Total_gross       4900.0
dtype: object
```

Best Performing Studio

In [56]:
```
# Grouping by 'studio' and summing 'domestic_gross' and 'foreign_gross'
movie_gross.groupby(['studio'])['Total_gross']
movie_gross.head()
```

Out[56]:

	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

In [57]:
```
# Grouping by 'studio' and 'year', and summing 'domestic_gross' and 'foreign_gr
movie_gross.groupby(['studio'])['Total_gross']
movie_gross.min()
```

Out[57]:
```
title                '71
studio                3D
domestic_gross     100.0
foreign_gross      600.0
year                2010
Total_gross       4900.0
dtype: object
```

In [58]:
```
# Grouping by 'studio' and 'year', and summing 'domestic_gross' and 'foreign_gr
movie_gross.groupby(['studio'])['Total_gross']
movie_gross.max()
```

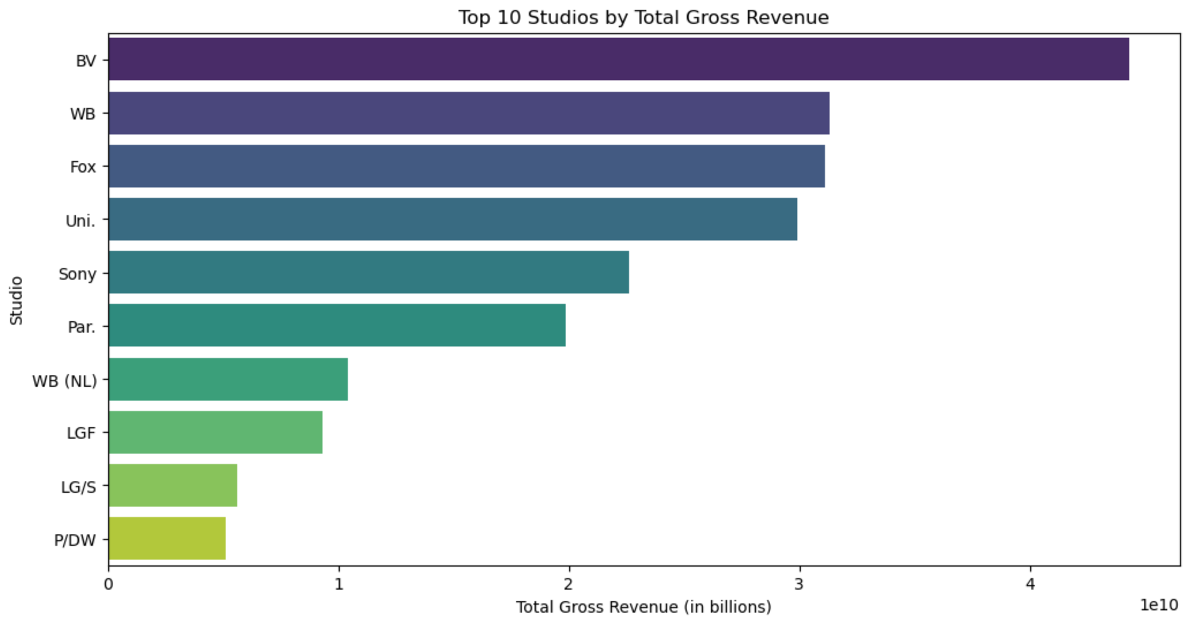
Out[58]:
```
title                xXx: The Return of Xander Cage
studio                                        Zeit.
domestic_gross                          936700000.0
foreign_gross                           960500000.0
year                                           2018
Total_gross                            1518900000.0
dtype: object
```

In [59]:
```
# Group by 'studio' and calculate the sum of 'Total_gross' for each studio
studio_total_gross = movie_gross.groupby(['studio'])['Total_gross'].sum().sort_
studio_total_gross
```

Out[59]:
```
studio
BV              4.430294e+10
WB              3.128625e+10
Fox             3.109543e+10
Uni.            2.989225e+10
Sony            2.261367e+10
                    ...
FOAK            1.243000e+05
IVP             1.121000e+05
Darin Southa    9.840000e+04
ITL             5.290000e+04
WOW             4.940000e+04
Name: Total_gross, Length: 258, dtype: float64
```

The studio with the highest average revenue is the BV whereas the studio with the lowest is WOW.Microsoft can conclude that BV studio have been successful in the market whereas WOW have not been successful hence the low earnings.This can be used as a measure of studio performance analysis and also inderstanding why BV studio are doing best to be the best player in the industry.

In [60]:
```
# Select the top 10 studios
top_10_studios = studio_total_gross.head(10)

# Creating a new figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Plotting
sns.barplot(x=top_10_studios.values, y=top_10_studios.index, palette='viridis',
ax.set_title("Top 10 Studios by Total Gross Revenue")
ax.set_xlabel('Total Gross Revenue (in billions)')
ax.set_ylabel('Studio')

plt.show()
```



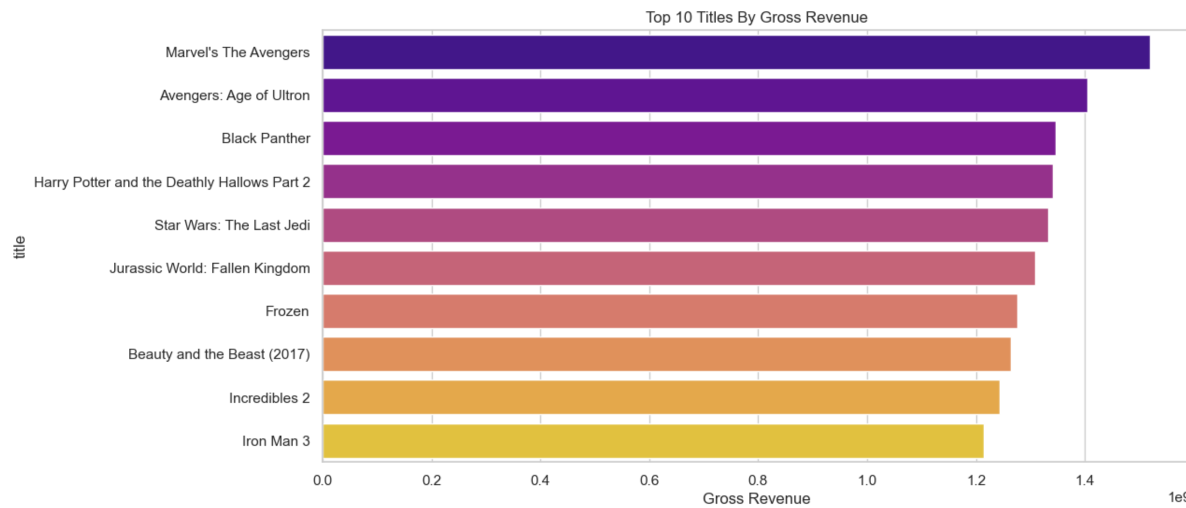
Best Performing Titles

In [61]:
```
# Grouping by 'title' and summing 'domestic_gross' and 'foreign_gross'
movie_gross.groupby(['title'])['Total_gross']
movie_gross.head()
```

Out[61]:

	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

In [102]:
```
# Group by 'title' and calculate the sum of 'Total_gross' for each studio
title_total_gross = movie_gross.groupby(['title'])['Total_gross'].sum().sort_va

# Select the top 10 studios
top_10_titles = title_total_gross.head(10)

# Creating a new figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Plotting
sns.barplot(x=top_10_titles.values, y=top_10_titles.index, palette='plasma', ax
ax.set_title("Top 10 Titles By Gross Revenue")
ax.set_xlabel('Gross Revenue')
ax.set_ylabel('title')

plt.show()
```



In [62]:
```
movie_gross.groupby(['title'])['Total_gross']
movie_gross.min()
```

Out[62]:
```
title                 '71
studio                 3D
domestic_gross      100.0
foreign_gross       600.0
year                 2010
Total_gross        4900.0
dtype: object
```

In [63]:
```
movie_gross.groupby(['title'])['Total_gross']
movie_gross.max()
```

Out[63]:
```
title             xXx: The Return of Xander Cage
studio                                     Zeit.
domestic_gross                       936700000.0
foreign_gross                        960500000.0
year                                        2018
Total_gross                         1518900000.0
dtype: object
```

Run Minutes Per Genre

In [64]:
```
# Grouping by 'studio' and summing 'domestic_gross' and 'foreign_gross'
#movie_gross.groupby(['studio'])['Total_gross']
#movie_gross.head()
```

In [65]:
```
combined_title_df.describe()
```

Out[65]:

	start_year	runtime_minutes	averagerating	numvotes
count	73856.000000	73856.000000	73856.000000	7.385600e+04
mean	2012.973137	83.451360	6.332729	3.523662e+03
std	2.382247	65.034231	1.474978	3.029402e+04
min	2010.000000	1.000000	1.000000	5.000000e+00
25%	2011.000000	67.469427	5.500000	1.400000e+01
50%	2013.000000	82.000000	6.500000	4.900000e+01
75%	2014.000000	96.000000	7.400000	2.820000e+02
max	2026.000000	14400.000000	10.000000	1.841066e+06

In [66]:
```
combined_title_df.head(2)
```

Out[66]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	tconst_r
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	tt103
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	tt103

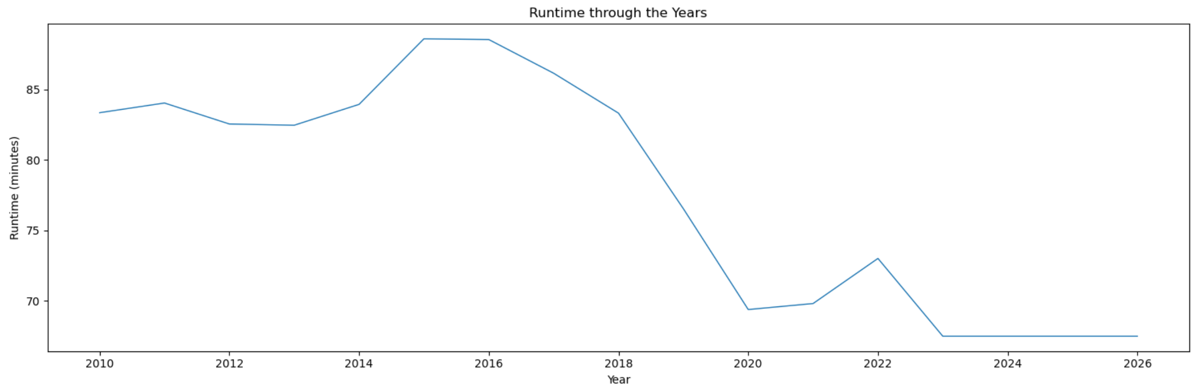
In [67]:
```
combined_title_df['genres'].count()
```

Out[67]: 73856

Movie Runtime Through The Years

We would like to recommendation for how long Microsoft's film should be. We looked at the runtime of movies through the start year to find an ideal length.

In [68]:
```
fig = plt.figure(figsize = (15,5))

# Create a line plot to visualize runtime through the decades
g = sns.lineplot(x='start_year', y='runtime_minutes', errorbar=None, data=comb
#Set plot title and axis labels
g.set(title = 'Runtime through the Years',
           ylabel = "Runtime (minutes)",
           xlabel = "Year")
plt.tight_layout()
fig.savefig("Runtime_decades.png")
plt.show()
```



In [69]:
```
#Grouping by genres
genre_ratings = combined_title_df.groupby('genres')['averagerating'].mean().res
genre_ratings.head()
```

Out[69]:

	genres	averagerating
0	Action	6.342155
1	Action,Adventure	6.356452
2	Action,Adventure,Animation	6.346465
3	Action,Adventure,Biography	6.489474
4	Action,Adventure,Comedy	6.394737

In [70]:
```
#genre with the highest averagerating
best_genre = genre_ratings.sort_values(by='averagerating', ascending=False).ilo
print(f"The best genre is {best_genre['genres']} with an average rating of {bes
```

```
The best genre is Animation,Mystery,Thriller with an average rating of 9.20
```

In [125]:
```
# Sort the DataFrame by average rating
best_genre = genre_ratings.sort_values(by='averagerating', ascending=False)
best_genre
```

Out[125]:

	genres	averagerating
358	Animation,Mystery,Thriller	9.2
275	Adventure,Romance,Sport	9.0
352	Animation,Music,Romance	8.9
804	Family,Fantasy,Horror	8.8
549	Comedy,Sci-Fi,Western	8.7
...	...	...
122	Action,Horror,Music	3.2
30	Action,Animation,Music	3.2
420	Biography,Fantasy,Horror	3.1
465	Comedy,Documentary,Sci-Fi	3.0
329	Animation,Family,Music	2.9

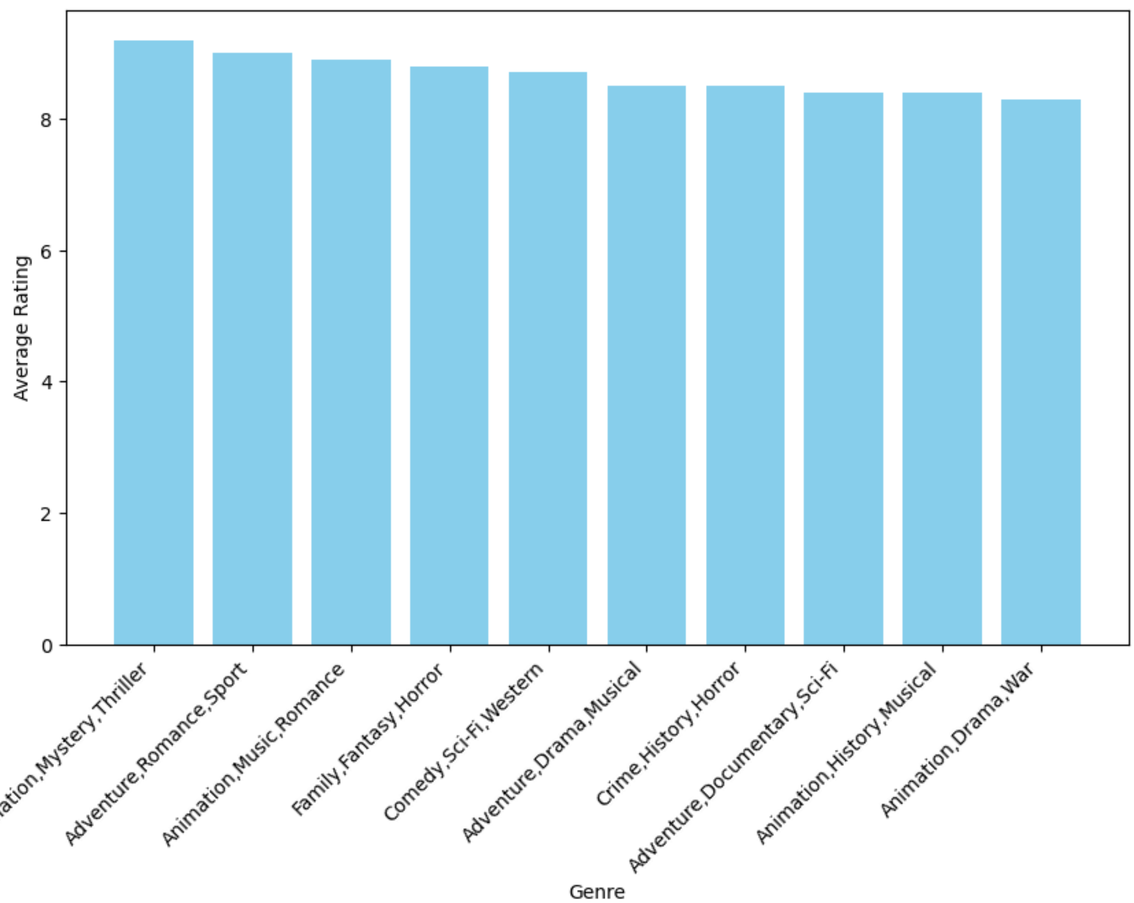
947 rows × 2 columns

In [71]:
```
top10_genre = best_genre.head(10)
top10_genre
```

Out[71]:
```
genres           Animation,Mystery,Thriller
averagerating                           9.2
Name: 358, dtype: object
```

In [73]:

```
# Sort the DataFrame by average rating
best_genre = genre_ratings.sort_values(by='averagerating', ascending=False)

top10_genre = best_genre.head(10)

# Plotting the top 10 genres
plt.figure(figsize=(10, 6))
plt.bar(top10_genre['genres'], top10_genre['averagerating'], color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title = ("Top 10 Genre by Average Rating")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabil
plt.show()
```



Best Genre Combinations

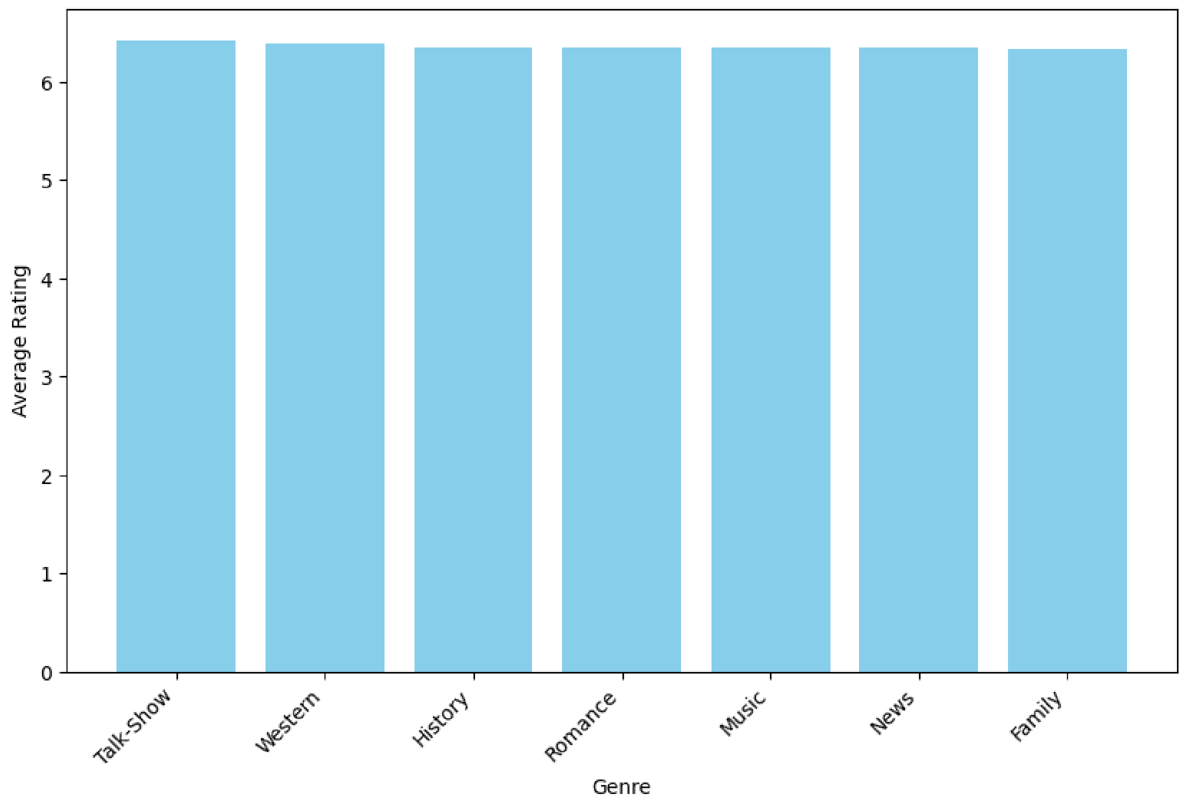
In [74]:
```
# Split the genres into a list
combined_title_df['genres'] = combined_title_df['genres'].str.split(',')

# Explode the DataFrame to create one row per genre in each title
exploded_df = combined_title_df.explode('genres')

# Group by the 'genres' column and calculate the average rating
genre_ratings = exploded_df.groupby('genres')['averagerating'].mean().reset_in

#to find the genre combination with the highest average rating.
#This will tell you which combination of genres tends to have the best ratings.
best_genre_combination = genre_ratings.sort_values(by='averagerating', ascendi

#print the best genre combination and its average rating
print(f"The best genre combination is {best_genre_combination['genres']} with a
```

```
The best genre combination is Talk-Show with an average rating of 6.42
```

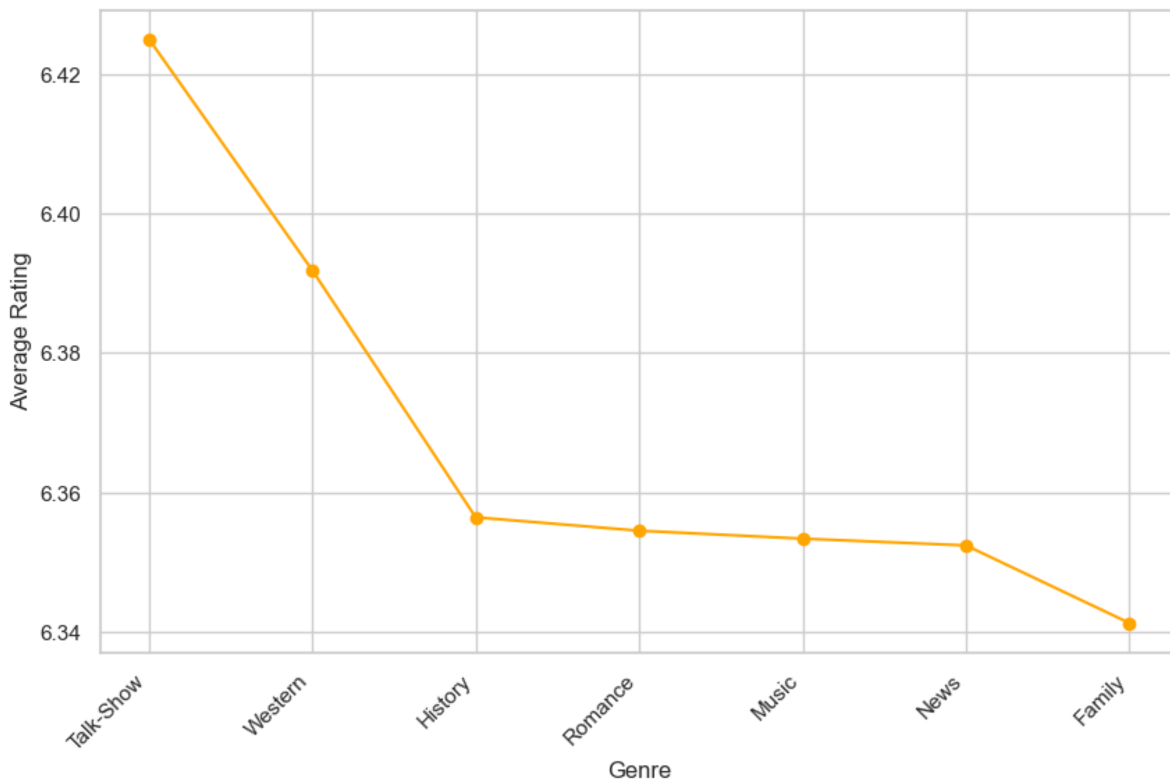
In [75]:
```
best_genre_combination.head(2)
```

Out[75]:
```
genres           Talk-Show
averagerating        6.425
Name: 24, dtype: object
```

In [76]:
```
best_genre_combination = genre_ratings.sort_values(by='averagerating', ascendi

top10_genrecombo = best_genre_combination.head(7)

# Plotting the top 7 genre combinations
plt.figure(figsize=(10, 6))
plt.bar(top10_genrecombo['genres'], top10_genrecombo['averagerating'], color='s
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title = ('Top 10 Genre by Average Rating')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabil
plt.show()
```



In [146]:
```
# Assuming you have already defined genre_ratings
best_genre_combination = genre_ratings.sort_values(by='averagerating', ascendi

top10_genrecombo = best_genre_combination.head(7)

# Plotting a line plot for the top 7 genre combinations
plt.figure(figsize=(10, 6))
plt.plot(top10_genrecombo['genres'], top10_genrecombo['averagerating'], marker=
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title = ('Top 7 Genre Combinations by Average Rating')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabil
plt.grid(True)  # Add grid for better visualization
plt.show()
```



In [ ]: