Microsoft Movie Studio

Student name: Ashley Simiyu

Student pace: Part-time

Scheduled project 5/11/2023

Instructor name: Samwel Jane

Blog post URL:

Overview

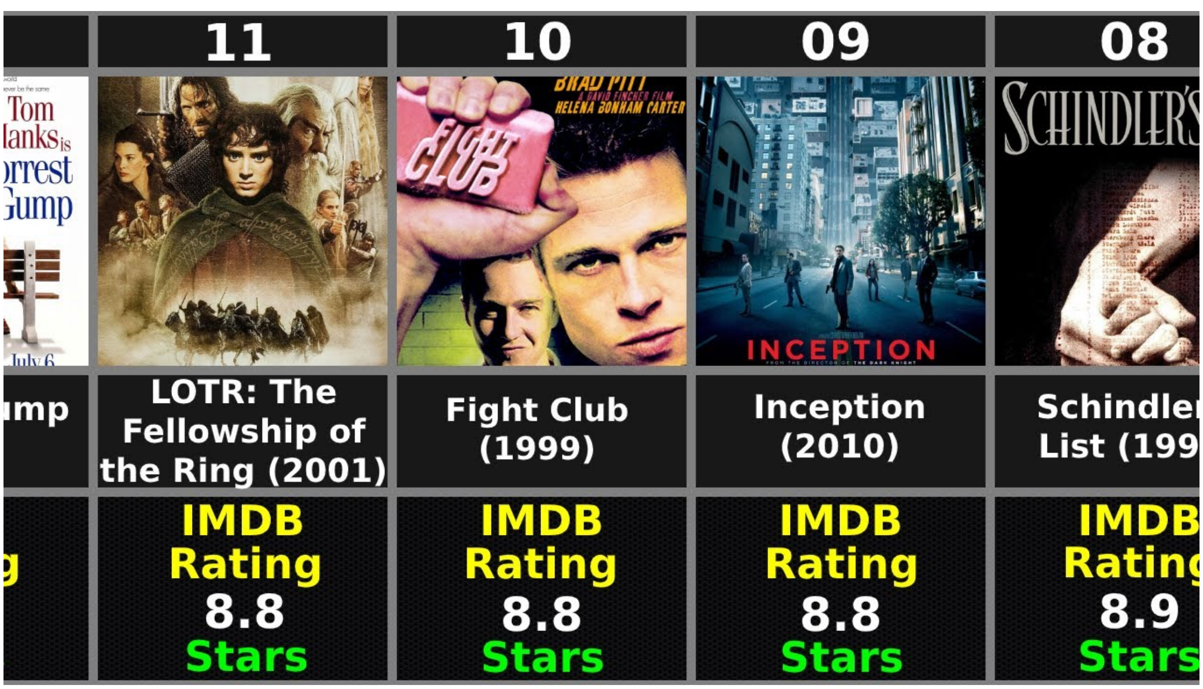
For this project, exploratory data analysis (EDA) will be used to generate insights for Microsoft.

Business Problem

Microsoft wants to start making movies, but they're not sure what kinds of movies are currently popular and successful at the box office. They want to know what types of movies people are watching and enjoying, so they can make informed decisions about the kinds of movies they should create.

Data Understanding

From the provided data sources (Box Office Mojo, IMDB, Rotten Tomatoes, TheMovieDB, The Numbers),find the most suitable dataset to use is one that provides information on movie titles, genres, ratings, and box office gross.



In [233]:
```
#import the necessary libraries
import pandas as pd
import numpy as np
```

In [234]:
```
#load the files
df_basics = pd.read_csv('title.basics.csv')
df_ratings = pd.read_csv('title.ratings.csv')
df_gross = pd.read_csv('bom.movie_gross.csv')
```

In [165]:
```
df_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    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
```

In [166]:
```
df_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 [167]:
```
df_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
```

Title Basics Data

The title_basics dataset contains information about movie titles, genres as well as how long the movie ran for.

In [168]:
```
df_basics.shape
```

Out[168]:
```
(146144, 6)
```

In [169]:
```
df_basics.head()
```

Out[169]:

	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 [170]:
```
# find the most frequent movie genres produced
df_basics['genres'].value_counts()
```

Out[170]:
```
Documentary                     32185
Drama                           21486
Comedy                           9177
Horror                           4372
Comedy,Drama                     3519
                                ...  
Adventure,Romance,Thriller          1
Animation,Documentary,Horror        1
Comedy,Sport,Western                1
Action,Animation,Mystery            1
Crime,Mystery,Western               1
Name: genres, Length: 1085, dtype: int64
```

In [171]:
```
# find the most frequent runtime_minutes of the movies produced
df_basics['runtime_minutes'].value_counts()
```

Out[171]:
```
90.0     7131
80.0     3526
85.0     2915
100.0    2662
95.0     2549
         ...
382.0       1
724.0       1
808.0       1
287.0       1
540.0       1
Name: runtime_minutes, Length: 367, dtype: int64
```

Ratings Data

The ratings dataset provides the ratings given and the number of votes of that rating.

In [172]:
```
df_ratings.shape
```

Out[172]:
```
(73856, 3)
```

In [173]:
```
df_ratings.head()
```

Out[173]:

	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 [174]:
```
# check how many ratings are greater than 7
high_ratings = df_ratings['averagerating'] > 7.0
high_ratings.value_counts()
```

Out[174]:
```
False    49211
True     24645
Name: averagerating, dtype: int64
```

Movie Gross

The movie gross dataset includes information on the title of the movie, the studio that produced it, the domestic and foreign gross and the year the movie got released.

In [175]:
```
df_gross.shape
```

Out[175]:
```
(3387, 5)
```

In [176]:
```
df_gross.head()
```

Out[176]:

	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 [177]:
```
# find the most used studio for movie production
df_gross['studio'].value_counts()
```

Out[177]:
```
IFC      166
Uni.     147
WB       140
Fox      136
Magn.    136
        ...
AZ         1
RME        1
Asp.       1
KS         1
CFI        1
Name: studio, Length: 257, dtype: int64
```

In [178]:
```
df_gross['year'].value_counts()
```

Out[178]:
```
2015    450
2016    436
2012    400
2011    399
2014    395
2013    350
2010    328
2017    321
2018    308
Name: year, dtype: int64
```

Data Preparation

Merging Data

Before cleaning the datasets, first I would merge the datasets for easier cleaning process of the data as one-merged dataset.

In [235]:
```
# merge the title_basics and ratings dataframes

merged_df = df_basics.merge(df_ratings.set_index('tconst'),on = 'tconst', how =
```

In [236]:
```
# concatenate the merged dataframes to movie_gross
concat_df = pd.concat([merged_df, df_gross], axis = 1)
concat_df
```

Out[236]:

	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
...	...	...	...	...	...	...
73851	tt9913084	Diabolik sono io	Diabolik sono io	2019	75.0	Documentary
73852	tt9914286	Sokagin Çocuklari	Sokagin Çocuklari	2019	98.0	Drama,Family
73853	tt9914642	Albatross	Albatross	2017	NaN	Documentary
73854	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	NaN	NaN
73855	tt9916160	Drømmeland	Drømmeland	2019	72.0	Documentary

73856 rows × 13 columns

Data Cleaning

Once all the datasets have been merged and concatenated, then the cleaning process would be easier and less strenuous. I will address any missing values, data types, and any other inconsistencies.

In [237]:
```
# check the null values in the concatenated dataframe
concat_df.isna().sum()
```

Out[237]:
```
tconst                0
primary_title         0
original_title        0
start_year            0
runtime_minutes    7620
genres              804
averagerating         0
numvotes              0
title             70469
studio            70474
domestic_gross    70497
foreign_gross     71819
year              70469
dtype: int64
```

In [238]:
```
concat_df.shape
```

Out[238]:
```
(73856, 13)
```

In [239]:
```
concat_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73856 entries, 0 to 73855
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           73856 non-null  object 
 1   primary_title    73856 non-null  object 
 2   original_title   73856 non-null  object 
 3   start_year       73856 non-null  int64  
 4   runtime_minutes  66236 non-null  float64
 5   genres           73052 non-null  object 
 6   averagerating    73856 non-null  float64
 7   numvotes         73856 non-null  int64  
 8   title            3387 non-null   object 
 9   studio           3382 non-null   object 
 10  domestic_gross   3359 non-null   float64
 11  foreign_gross    2037 non-null   object 
 12  year             3387 non-null   float64
dtypes: float64(4), int64(2), object(7)
memory usage: 7.9+ MB
```

In [240]:
```
# drop the unnecessary columns
concat_df.drop(columns = ['original_title', 'title', 'year'], axis = 1, inplace
```

In [241]:
```
# replace the nullvalues in the runtime_minutes with 0
concat_df['runtime_minutes'].fillna(0, inplace = True)

#filling the nullvalues in genres with 'unknown'
concat_df['genres'].fillna('Unknown', inplace=True)
```

In [242]:
```
# checkthe data in the first five rows
concat_df.head()
```

Out[242]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating	num
0	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography,Drama	7.2	
2	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9	
3	tt0069204	Sabse Bada Sukh	2018	0.0	Comedy,Drama	6.1	
4	tt0100275	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5	

In [243]:
```
## check the data from the 1038th to 1069th row
concat_df.iloc[1038:1070]
```

Out[243]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerat
1038	tt1107855	You Have the Right to Remain Violent	2010	88.0	Action,Drama,Thriller	
1039	tt1109467	Standing Silent	2011	84.0	Documentary	
1040	tt1109488	Mars	2010	90.0	Animation,Comedy,Sci-Fi	
1041	tt1109574	Between Us	2012	90.0	Drama	
1042	tt1109582	A Day of Violence	2010	91.0	Crime,Thriller	
1043	tt1109587	Driving Me Crazy	2012	0.0	Comedy,Drama,Romance	
1044	tt1109594	Kalamity	2010	98.0	Drama,Thriller	
1045	tt1109624	Paddington	2014	95.0	Adventure,Comedy,Family	
1046	tt1110208	The Bend	2011	85.0	Drama	
1047	tt1111235	Trance	2010	83.0	Horror	
1048	tt1111313	The Elephant in the Living Room	2010	96.0	Documentary	
1049	tt1111884	Code Blue	2010	93.0	Crime,Drama	
1050	tt1111900	Voices Unbound: The Story of the Freedom Writers	2010	90.0	Documentary	
1051	tt1112291	Turn It Up!	2014	86.0	Documentary	
1052	tt1113829	George Harrison: Living in the Material World	2011	208.0	Biography,Documentary,Music	
1053	tt1114710	He Ain't Like That	2010	110.0	Thriller	
1054	tt1114731	Seres: Genesis	2010	110.0	Action,Adventure,Sci-Fi	
1055	tt1114732	Soundtrack	2015	90.0	Thriller	
1056	tt1116183	Carmen's Kiss	2010	90.0	Drama,Romance,Thriller	
1057	tt1116184	Jackass 3D	2010	95.0	Action,Comedy,Documentary	
1058	tt1117390	A Man Without a Country	2012	115.0	Comedy,Documentary	
1059	tt1117593	Kluge	2010	0.0	Thriller	
1060	tt1117668	King of Paper Chasin'	2011	124.0	Crime,Drama,Music	

	tconst	primary_title	start_year	runtime_minutes	genres	averagerat
1061	tt1119192	The Justice of Wolves	2010	94.0	Drama,Mystery	
1062	tt1119630	La revolución es un sueño eterno	2012	110.0	Biography,History	
1063	tt1120919	A Mormon President	2011	0.0	Documentary	
1064	tt1120985	Blue Valentine	2010	112.0	Drama,Romance	
1065	tt1121096	Seventh Son	2014	102.0	Action,Adventure,Fantasy	
1066	tt1121986	Money Fight	2012	119.0	Action,Drama	
1067	tt1122614	And Everything Is Going Fine	2010	89.0	Documentary	
1068	tt1123373	Detective Dee: The Mystery of the Phantom Flame	2010	123.0	Action,Adventure,Drama	
1069	tt1124035	The Ides of March	2011	101.0	Drama,Thriller	

In [244]:
```
# check the data in the last 20 rows
concat_df.tail(20)
```

Out[244]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating
73836	tt9903716	Jessie	2019	106.0	Horror,Thriller	8.5
73837	tt9903952	BADMEN with a good behavior	2018	87.0	Comedy,Horror	9.2
73838	tt9904014	Lost in Klessin	2018	90.0	War	7.3
73839	tt9904820	American Terror Story	2019	76.0	Horror	2.6
73840	tt9904844	Ott Tänak: The Movie	2019	125.0	Documentary	8.7
73841	tt9905412	Ottam	2019	120.0	Drama	8.1
73842	tt9905462	Pengalila	2019	111.0	Drama	8.4
73843	tt9905476	Hand Rolled	2019	90.0	Documentary	9.3
73844	tt9905796	July Kaatril	2019	0.0	Romance	9.0
73845	tt9906218	Unstoppable	2019	84.0	Documentary	8.1
73846	tt9908960	Pliusas	2018	90.0	Comedy	4.2
73847	tt9910502	Hayatta Olmaz	2019	97.0	Comedy	7.0
73848	tt9910930	Jeg ser deg	2019	75.0	Crime,Documentary	6.1
73849	tt9911774	Padmavyuhathile Abhimanyu	2019	130.0	Drama	8.4
73850	tt9913056	Swarm Season	2019	86.0	Documentary	6.2
73851	tt9913084	Diabolik sono io	2019	75.0	Documentary	6.2
73852	tt9914286	Sokagin Çocuklari	2019	98.0	Drama,Family	8.7
73853	tt9914642	Albatross	2017	0.0	Documentary	8.5
73854	tt9914942	La vida sense la Sara Amat	2019	0.0	Unknown	6.6
73855	tt9916160	Drømmeland	2019	72.0	Documentary	6.5

In [245]:
```
#convert the datatype of foreign gross from object to float
concat_df['foreign_gross'] = pd.to_numeric(concat_df['foreign_gross'], errors='
```

In [246]:
```
#find the median of the foreign gross
median_gross = concat_df['foreign_gross'].median()

# fill in the missing values with the foreign_gross median
concat_df['foreign_gross'].fillna(median_gross, inplace = True)
```

In [247]:
```
# check if there are other null values
concat_df.isna().sum()
```

Out[247]:
```
tconst                 0
primary_title          0
start_year             0
runtime_minutes        0
genres                 0
averagerating          0
numvotes               0
studio             70474
domestic_gross     70497
foreign_gross          0
dtype: int64
```

There are still missing values in the studio and domestic_gross column

In [248]:
```
#find the median of the domestic_gross
median_gross = concat_df['domestic_gross'].median()

# fill in the missing values with the domestic_gross median
concat_df['domestic_gross'].fillna(median_gross, inplace = True)
```

In [249]:
```
#filling the nullvalues in studio with 'unknown'
concat_df['studio'].fillna('Unknown', inplace=True)
```

In [250]:
```
concat_df.isna().sum()
```

Out[250]:
```
tconst             0
primary_title      0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
studio             0
domestic_gross     0
foreign_gross      0
dtype: int64
```

Now the concatenated dataset has been cleaned! I can now head to analyse the data.

Data Analysis

Perform various data exploration and visualization tasks to gain insights of the data provided.

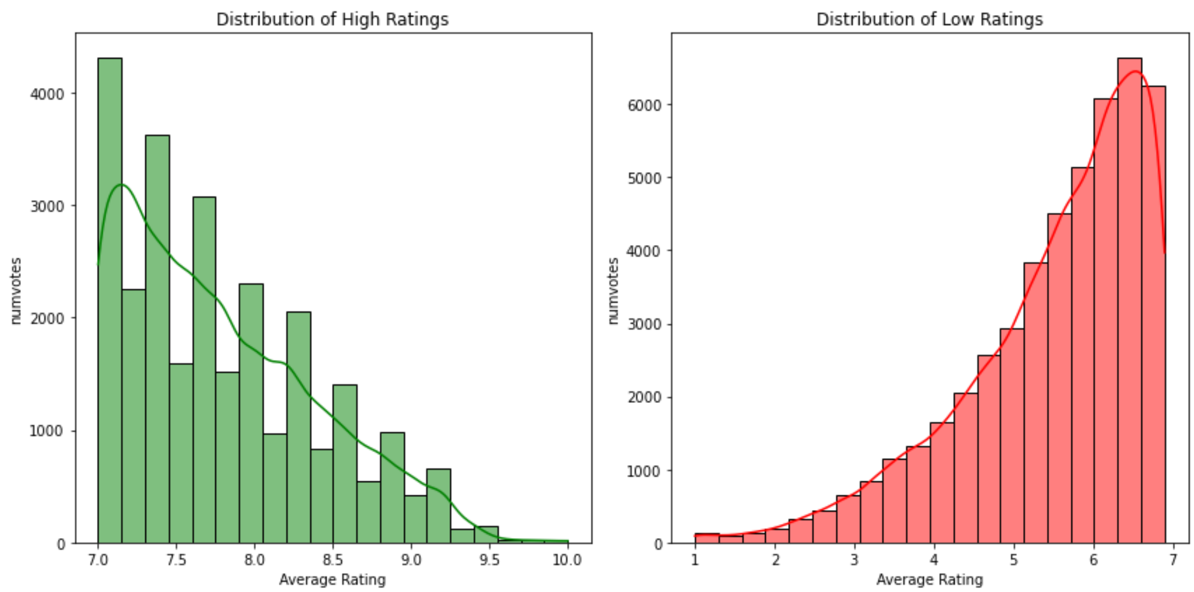
In [251]:
```python
#import the necessary libraries for plot analysis
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [252]:
```python
# check how the data looks like
concat_df.iloc[57900:57920]
```

Out[252]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerati
57900	tt5943878	Pagdi	2016	0.0	Action	3
57901	tt5943940	Abduction	2017	90.0	Action,Comedy,Drama	3
57902	tt5944422	Truth or Dare	2016	90.0	Drama,Romance	6
57903	tt5944670	Pinkwashing Exposed: Seattle Fights Back!	2015	0.0	Unknown	8
57904	tt5944812	Dead Sunrise	2017	120.0	Adventure,Horror,Sci-Fi	7
57905	tt5945054	Isäni tähtien takaa	2016	80.0	Documentary	6
57906	tt5945222	Dugma: The Button	2016	58.0	Documentary	7
57907	tt5945282	Cahier africain	2016	118.0	Documentary	7
57908	tt5945286	Raving Iran	2016	84.0	Documentary	7
57909	tt5945584	Lamparina da Aurora	2017	74.0	Drama,Thriller	7
57910	tt5945724	The Garden	2017	97.0	Drama	6
57911	tt5945946	1st Strike	2016	99.0	Drama	4
57912	tt5946128	Dear Zindagi	2016	151.0	Drama,Romance	7
57913	tt5946552	Addicted to Porn: Chasing the Cardboard Butterfly	2017	82.0	Documentary,Drama,History	4
57914	tt5946668	4/20 Massacre	2018	84.0	Action,Horror	3
57915	tt5946852	People's Garage	2016	162.0	Action,Drama	7
57916	tt5946936	Surga Yang Tak Dirindukan 2	2017	121.0	Drama	7
57917	tt5946974	3 Srikandi	2016	0.0	Biography,Sport	6
57918	tt5947284	Holy God	2017	25.0	Documentary	6
57919	tt5947332	Cryptic Road	2016	84.0	Mystery,Sci-Fi,Thriller	7

In [253]:
```
#Create a histogram to visualize the distribution of 'averagerating'

# Filter the data for high and low ratings
high_ratings = concat_df[concat_df['averagerating'] >= 7]
low_ratings = concat_df[concat_df['averagerating'] < 7]

# Create two subplots
plt.figure(figsize=(12, 6))

# Subplot 1: High Ratings
plt.subplot(1, 2, 1)
sns.histplot(data=high_ratings, x='averagerating', bins=20, kde=True, color='gr
plt.title('Distribution of High Ratings')
plt.xlabel('Average Rating')
plt.ylabel('numvotes')

# Subplot 2: Low Ratings
plt.subplot(1, 2, 2)
sns.histplot(data=low_ratings, x='averagerating', bins=20, kde=True, color='rec
plt.title('Distribution of Low Ratings')
plt.xlabel('Average Rating')
plt.ylabel('numvotes')


plt.tight_layout()
plt.show()
```



The distribution of Low ratings is negative skewness while the High ratings distribution has a positive skweness

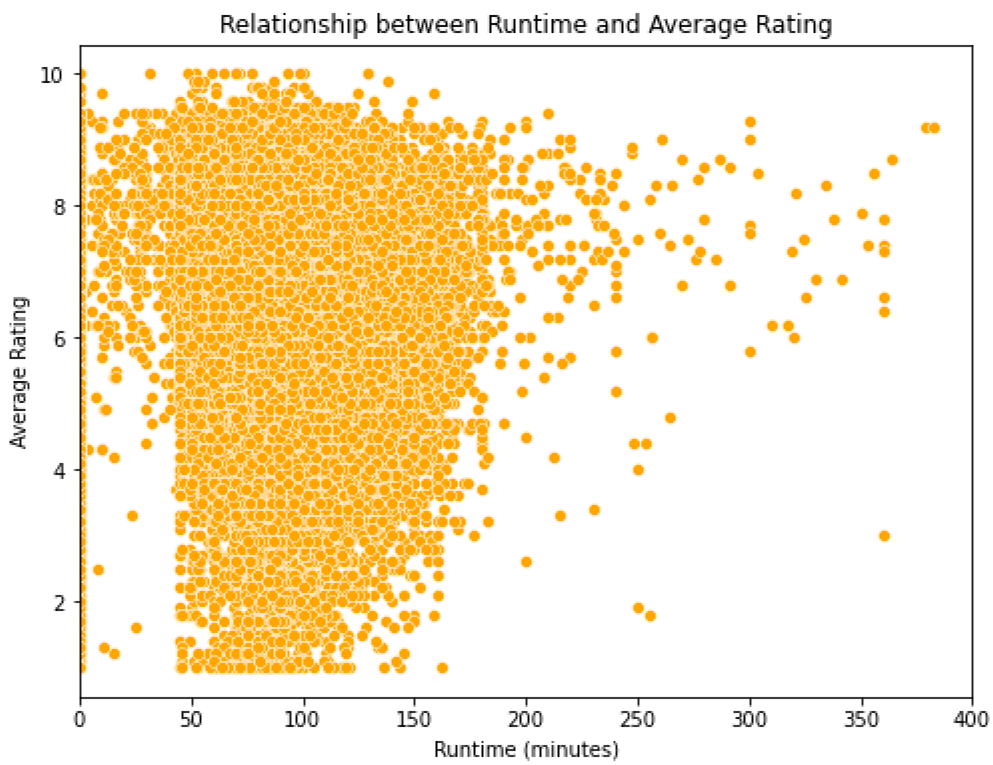
In [254]:
```
concat_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73856 entries, 0 to 73855
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           73856 non-null  object 
 1   primary_title    73856 non-null  object 
 2   start_year       73856 non-null  int64  
 3   runtime_minutes  73856 non-null  float64
 4   genres           73856 non-null  object 
 5   averagerating    73856 non-null  float64
 6   numvotes         73856 non-null  int64  
 7   studio           73856 non-null  object 
 8   domestic_gross   73856 non-null  float64
 9   foreign_gross    73856 non-null  float64
dtypes: float64(4), int64(2), object(4)
memory usage: 6.2+ MB
```

As you can see above, the data type for genres was a float, which should not be the case. I will change the its data type to string.

Voila! Now the data set is perfect to work with.

In [256]:
```
# plot the relationship between runtime_minutes and avarageratingusing a scatt
plt.figure(figsize=(8, 6))
sns.scatterplot(data=concat_df, x=('runtime_minutes'), y='averagerating', color
plt.title('Relationship between Runtime and Average Rating')
plt.xlabel('Runtime (minutes)')
plt.ylabel('Average Rating')
plt.xlim(0, 400)
plt.show()
```



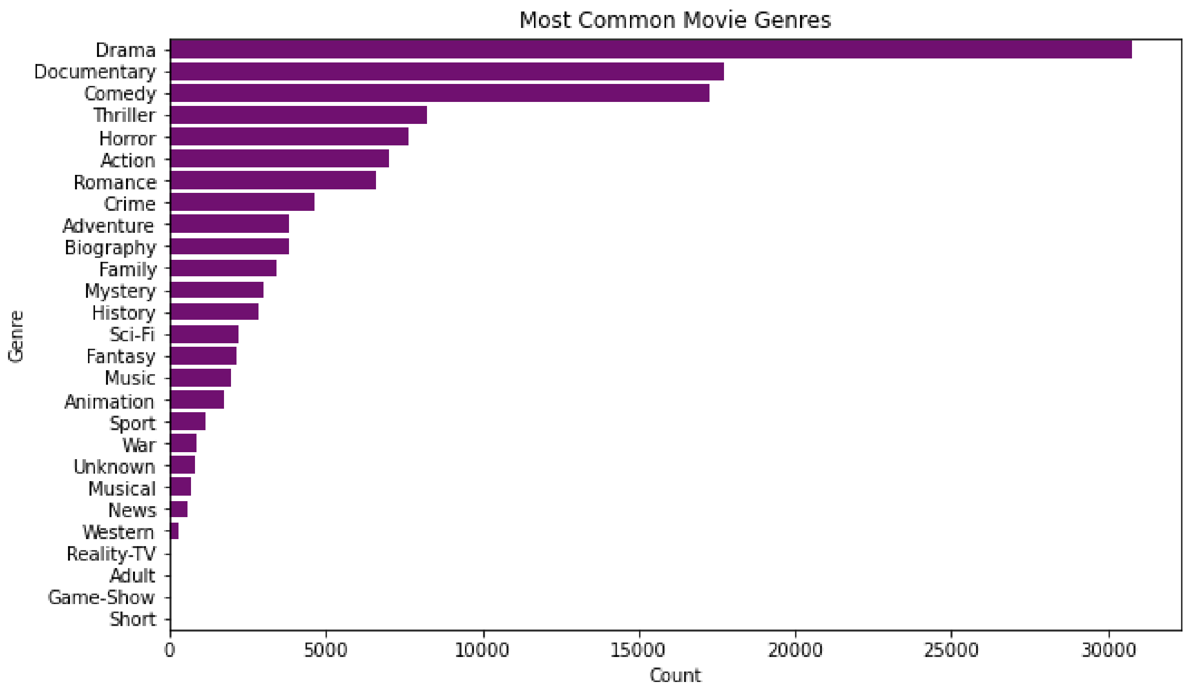
The high movie ratings are for those movies that run in less than 3 hours(180 minutes)

In [232]:
```
concat_df.head()
```

Out[232]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating	numvotes	studio
0	tt0063540	Sunghursh	2013	175.0	[nan]	7.0	77	BV
1	tt0066787	One Day Before the Rainy Season	2019	114.0	[nan]	7.2	43	BV
2	tt0069049	The Other Side of the Wind	2018	122.0	[nan]	6.9	4517	WB
3	tt0069204	Sabse Bada Sukh	2018	0.0	[nan]	6.1	13	WB
4	tt0100275	The Wandering Soap Opera	2017	80.0	[nan]	6.5	119	P/DW

In [259]:
```
# plot the most common movie genres
genre_counts = concat_df['genres'].str.split(',').explode().value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(y=genre_counts.index, x=genre_counts.values, orient='h', color='pur
plt.title('Most Common Movie Genres')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

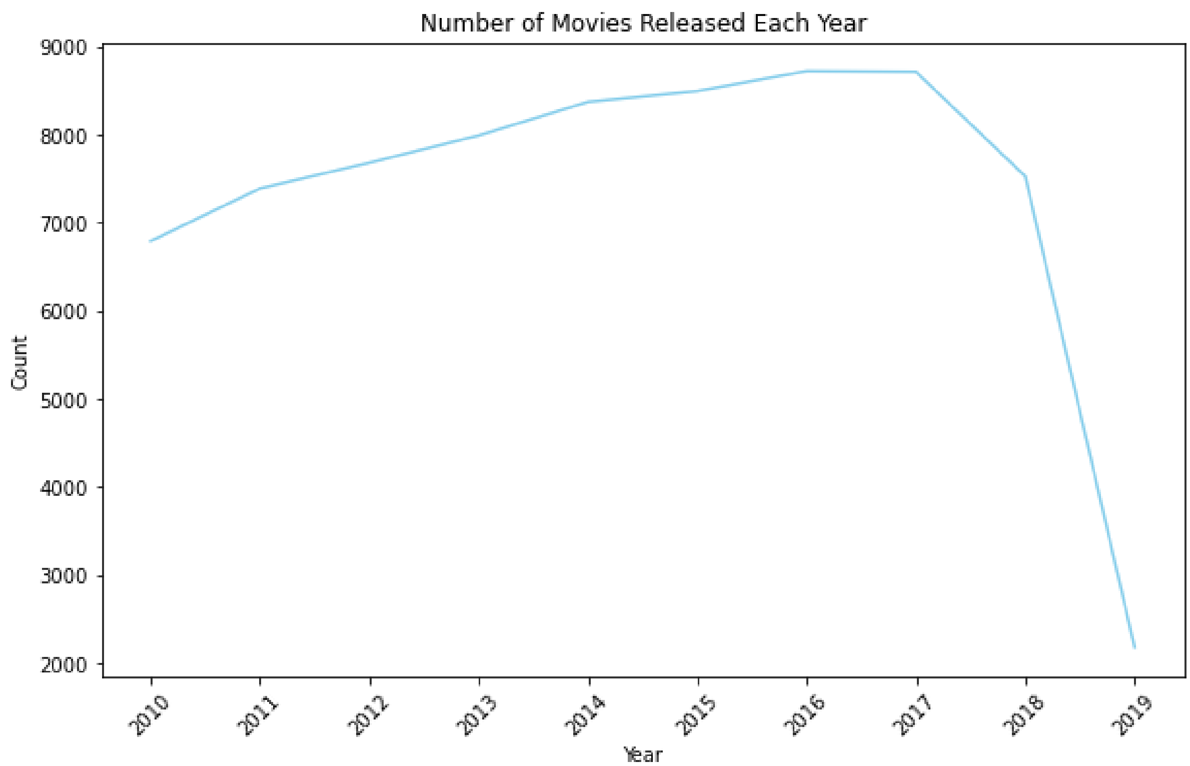


In [262]:
```
concat_df.describe()
```

Out[262]:

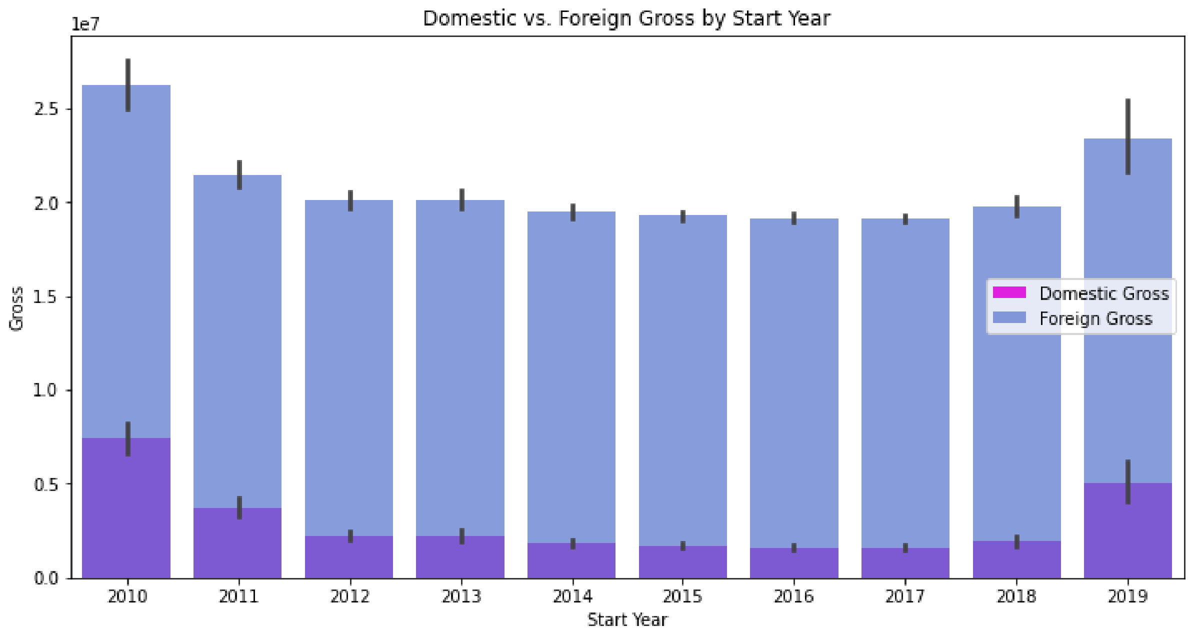
	start_year	runtime_minutes	averagerating	numvotes	domestic_gross	foreign_gros
count	73856.000000	73856.000000	73856.000000	7.385600e+04	7.385600e+04	7.385600e+0
mean	2014.276132	84.888228	6.332729	3.523662e+03	2.643700e+06	2.044505e+0
std	2.614807	199.608940	1.474978	3.029402e+04	1.537727e+07	2.458699e+0
min	2010.000000	0.000000	1.000000	5.000000e+00	1.000000e+02	6.000000e+0
25%	2012.000000	75.000000	5.500000	1.400000e+01	1.400000e+06	1.890000e+0
50%	2014.000000	90.000000	6.500000	4.900000e+01	1.400000e+06	1.890000e+0
75%	2016.000000	101.000000	7.400000	2.820000e+02	1.400000e+06	1.890000e+0
max	2019.000000	51420.000000	10.000000	1.841066e+06	9.367000e+08	9.605000e+0

In [113]:
```
year_counts = concat_df['start_year'].value_counts().sort_index()
plt.figure(figsize=(10, 6))
sns.lineplot(x=year_counts.index, y=year_counts.values, color='skyblue')
plt.title('Number of Movies Released Each Year')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(year_counts.index, rotation=45)
plt.show()
```



The number of movies released are drastically dropping in the two years i.e 2017-2019

In [115]:
```
# Grouped bar plot to compare domestic and foreign gross per start_year
plt.figure(figsize=(12, 6))

# Grouped bar plot for domestic gross
sns.barplot(x='start_year', y='domestic_gross', data=concat_df,color='magenta',

# Grouped bar plot for foreign gross next to domestic gross
sns.barplot(x='start_year', y='foreign_gross', data=concat_df, color='royalblue

# Add labels and legends
plt.title('Domestic vs. Foreign Gross by Start Year')
plt.xlabel('Start Year')
plt.ylabel('Gross')
plt.legend(loc='right')

plt.show()
```



For each start year, the foreign gross is higher than the domestic gross, hence most money is earned from the foreign gross as compared to domestic gross

Insights

From this analysis, there are a couple of things that needs to be considered

What are the target audience demographics and preferences for movie genres? From the 'Domestic vs. Foreign Gross by Start Year' graph, it shows that the biggest target audience should be the foreigners. The foreigners(globally) are the biggest contributors to the gross. This means for every movie produced, it svhould be able to have subtitles for the non-language speakers.

Are there specific genres or themes that have gained popularity? There are three most highly rated movie genres. And those are Drama, Documentary and Comedy.

How long should a movie be? For most part, the most reccommended duration for a movie is less than 180 minutes. That way the audience won't lose the attention of the message of the movie.

Challenges

Little time for analysis

Conclusion

In summary,I may not reccommend Microsoft into entering the film industry. With the data used for analysis, it doesn't really shade thelight on the expenses incurred as weel as the budget for producing a movie. Also the data maybe abut outdated, since the data sources are not from