Final Project Submission

Please fill out:

- Student name: VIVIAN KERUBO MOSOMI
- Student pace: PART TIME
- Scheduled project review date/time:
- Instructor name: FAITH ROTICH
- Blog post URL:

1.Fetching the data needed

The cell below is importing all the necessary packages needed.

In [43]:
```
#Importing pandas
import pandas as pd

#For visualizations
import matplotlib.pyplot as plt
import seaborn as sns
```

Studying the title basics data

In [44]:
```
title_basics_df = pd.read_csv('zippedData/imdb.title.basics.csv.gz')
title_basics_df
```

Out[44]:

	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
...	...	...	...	...	...	...
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	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [45]:
```
# The last five rows

title_basics_df.tail()
```

Out[45]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
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	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

In [46]:
```
#Number of columns

title_basics_df.columns
```

Out[46]:
```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')
```

In [47]:
```
# The Data Types of

title_basics_df.dtypes
```

Out[47]:
```
tconst              object
primary_title       object
original_title      object
start_year           int64
runtime_minutes    float64
genres              object
dtype: object
```

In [48]:
```
# Info of the data

title_basics_df.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 [49]:
```
# Summary statistics of the data

title_basics_df.describe()
```

Out[49]:

	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

Cleaning the title basics data

Separating the genres columns to avoid multiple values in a row

In [50]:
```
"""
The splitting below is to split the genres on one column to different ones for better analysis.
It returns each split element as a separate column
"""
```

Out[50]:
```
'\nThe splitting below is to split the genres on one column to different ones for better analysis.\nIt returns each split element as a separate column\n'
```

In [51]:
```
# Splitting the genres column to have multiple columns
split_genres = title_basics_df['genres'].str.split(',',expand=True)

#Concatenating the original dataframe with the split df
title_df_normalized = pd.concat([title_basics_df, split_genres],axis=1)

#Renaming columns
title_df_normalized.columns = ['tconst','primary_title','original_title','start_year','runtime_minutes','genres','Genr

title_df_normalized
```

Out[51]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	Genre1	Genre2	Genre3
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	Action	Crime	Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	Biography	Drama	None
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	Drama	None	None
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	Comedy	Drama	None
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	Comedy	Drama	Fantasy
...	...	...	...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama	Drama	None	None
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary	Documentary	None	None
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy	Comedy	None	None
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN	NaN	NaN	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary	Documentary	None	None

146144 rows × 9 columns

In [52]:
```
title_df_normalized.columns = title_df_normalized.columns.str.strip()
title_df_normalized.columns
```

Out[52]:
```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'Genre1', 'Genre2', 'Genre3'],
      dtype='object')
```

In [53]:
```
# Dropping the genres column to avoid duplication

title_df_normalized.drop(columns=['genres'],axis=1,inplace=True)

title_df_normalized
```

Out[53]:

	tconst	primary_title	original_title	start_year	runtime_minutes	Genre1	Genre2	Genre3
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	None
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	None	None
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy	Drama	None
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	None	None
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary	None	None
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy	None	None
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN	NaN	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary	None	None

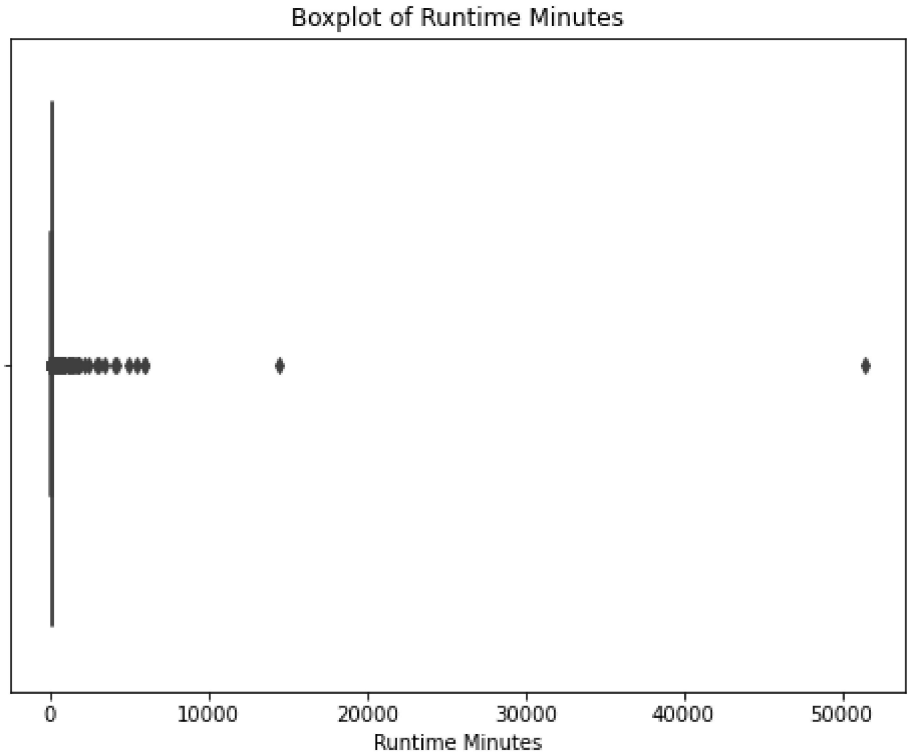
146144 rows × 8 columns

Dealing with null values

In [54]:
```
# Checking for null values again

title_df_normalized.isna().sum()
```

Out[54]:
```
tconst                0
primary_title         0
original_title       21
start_year            0
runtime_minutes   31739
Genre1             5408
Genre2            86766
Genre3           116708
dtype: int64
```

In [55]:
```
plt.figure(figsize=(8, 6))

# Boxplot to check for outliers in 'runtime_minutes'
sns.boxplot(x=title_df_normalized['runtime_minutes'])

#Labelling the plot
plt.title('Boxplot of Runtime Minutes')
plt.xlabel('Runtime Minutes')

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



In [56]:
```
"""
From the boxplot above, the numerical column runtime_minutes has outliers
Because median is less affected by outliers, we'll use it to fill in null values as below
"""
```

Out[56]:
```
"\nFrom the boxplot above, the numerical column runtime_minutes has outliers\nBecause median is less affected by out
liers, we'll use it to fill in null values as below\n"
```

In [57]:
```
# Filling in the null values in the 'Original title' column with 'Blank Genre'
title_df_normalized['original_title'].fillna('Missing Title',inplace=True)
```

In [58]:
```
# Filing missing values in the 'runtime_minutes' column with the median
title_df_normalized['runtime_minutes'].fillna(title_df_normalized['runtime_minutes'].median(), inplace=True)

# Filling missing values in 'Genre1', 'Genre2', and 'Genre3' with 'Blank Genre'
title_df_normalized['Genre1'].fillna('Blank Genre', inplace=True)
title_df_normalized['Genre2'].fillna('Blank Genre', inplace=True)
title_df_normalized['Genre3'].fillna('Blank Genre', inplace=True)
```

In [59]:
```
# Checking if null values still exist

title_df_normalized.isna().sum()
```

Out[59]:
```
tconst            0
primary_title     0
original_title    0
start_year        0
runtime_minutes   0
Genre1            0
Genre2            0
Genre3            0
dtype: int64
```

In [60]:
```
"""
From the above, now we don't have null values in the title basics dataset
"""
```

Out[60]:
```
"\nFrom the above, now we don't have null values in the title basics dataset\n"
```

In [61]:
```
#Checking the whole data set
title_df_normalized
```

Out[61]:

	tconst	primary_title	original_title	start_year	runtime_minutes	Genre1	Genre2	Genre3
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	Blank Genre
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	Blank Genre	Blank Genre
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	87.0	Comedy	Drama	Blank Genre
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	Blank Genre	Blank Genre
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	87.0	Documentary	Blank Genre	Blank Genre
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	87.0	Comedy	Blank Genre	Blank Genre
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	Blank Genre	Blank Genre	Blank Genre
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	87.0	Documentary	Blank Genre	Blank Genre

146144 rows × 8 columns

Dealing with duplicates

In [62]:
```
# Checking for duplicates

title_df_normalized.duplicated().sum()
```

Out[62]:
```
0
```

In [63]:
```
"""
There are no duplicates from the above code
"""
```

Out[63]:
```
'\nThere are no duplicates from the above code\n'
```

Studying the title ratings data

In [64]:
```
title_ratings_df = pd.read_csv('zippedData/imdb.title.ratings.csv.gz')

title_ratings_df
```

Out[64]:

	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
...	...	...	...
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 [65]:
```
# First five rows

title_ratings_df.head()
```

Out[65]:

	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 [66]:
```
#Number of columns

title_ratings_df.columns
```

Out[66]:
```
Index(['tconst', 'averagerating', 'numvotes'], dtype='object')
```

In [67]:
```
# The data types

title_ratings_df.dtypes
```

Out[67]:
```
tconst            object
averagerating    float64
numvotes           int64
dtype: object
```

In [68]:
```
# Info of the title ratings data

title_ratings_df.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 [69]:
```
# Summary statistics

title_ratings_df.describe()
```

Out[69]:

	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

Cleaning the title ratings data

In [70]:
```
# Check for null values

title_ratings_df.isna().sum()  # No null values
```

Out[70]:
```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

In [71]:
```
# Checking for duplicates

title_ratings_df.duplicated().sum()  # No duplicates
```

Out[71]:
```
0
```

In [72]:
```
"""
Title ratings has no null values or duplicate values
"""
```

Out[72]:
```
'\nTitle ratings has no null values or duplicate values\n'
```

Studying the movie_gross data

In [73]:
```
movie_gross_df = pd.read_csv('zippedData/bom.movie_gross.csv.gz')

movie_gross_df
```

Out[73]:

	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
...	...	...	...	...	...
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

3387 rows × 5 columns

In [74]:
```
# First five rows

movie_gross_df.head()
```

Out[74]:

	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 [75]:
```
# Number of columns

movie_gross_df.columns
```

Out[75]:
```
Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype='object')
```

In [76]:
```
# The data types

movie_gross_df.dtypes
```

Out[76]:
```
title              object
studio             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object
```

In [77]:
```
# Info of the movie gross data

movie_gross_df.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 [78]:
```
# Summary statistics

movie_gross_df.describe()
```

Out[78]:

	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

Cleaning the movie gross data

In [79]:
```
# Checking missing values

movie_gross_df.isna().sum()
```

Out[79]:
```
title              0
studio             5
domestic_gross    28
foreign_gross   1350
year               0
dtype: int64
```

In [80]:
```
# Filling in missing values

# studio column
movie_gross_df['studio'].fillna('Missing',inplace=True)

# Domestic gross
movie_gross_df['domestic_gross'].fillna(0,inplace=True)

# Foreign gross
movie_gross_df['foreign_gross'].fillna(0,inplace=True)
```

In [81]:
```
# Checking if missing values are still there

movie_gross_df.isna().sum()
```

Out[81]:
```
title             0
studio            0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

In [82]:
```
# Checking for duplicates

movie_gross_df.duplicated().sum()
```

Out[82]:
```
0
```

In [83]:
```
"""
For movie gross there are no duplicates and I've filled in missing values
"""
```

Out[83]:
```
"\nFor movie gross there are no duplicates and I've filled in missing values\n"
```

Combining the data for And Cleaning it

In [84]:
```
merged_df = pd.merge(title_df_normalized, title_ratings_df, on='tconst')
merged_df = pd.merge(merged_df, movie_gross_df, left_on=["primary_title", "primary_title"], right_on=["title", "title"

merged_df
```

Out[84]:

	tconst	primary_title	original_title	start_year	runtime_minutes	Genre1	Genre2	Genre3	averagerating	numvotes	title
0	tt0315642	Wazir	Wazir	2016	103.0	Action	Crime	Drama	7.1	15378	Wazir
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure	Drama	Romance	6.1	37886	On the Road
2	tt4339118	On the Road	On the Road	2014	89.0	Drama	Blank Genre	Blank Genre	6.0	6	On the Road
3	tt5647250	On the Road	On the Road	2016	121.0	Drama	Blank Genre	Blank Genre	5.7	127	On the Road
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure	Comedy	Drama	7.3	275300	The Secret Life of Walter Mitty
...	...	...	...	...	...	...	...	...	...	...	...
3022	tt8331988	The Chambermaid	La camarista	2018	102.0	Drama	Blank Genre	Blank Genre	7.1	147	The Chambermaid
3023	tt8404272	How Long Will I Love U	Chao shi kong tong ju	2018	101.0	Romance	Blank Genre	Blank Genre	6.5	607	How Long Will I Love U
3024	tt8427036	Helicopter Eela	Helicopter Eela	2018	135.0	Drama	Blank Genre	Blank Genre	5.4	673	Helicopter Eela
3025	tt9078374	Last Letter	Ni hao, Zhihua	2018	114.0	Drama	Romance	Blank Genre	6.4	322	Last Letter
3026	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary	Music	Blank Genre	8.8	2067	Burn the Stage: The Movie

3027 rows × 15 columns

In [85]:
```
# Checking for missing values in the merged data frame

merged_df.isnull().sum()
```

Out[85]:
```
tconst             0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
Genre1             0
Genre2             0
Genre3             0
averagerating      0
numvotes           0
title              0
studio             0
domestic_gross     0
foreign_gross      0
year               0
dtype: int64
```

In [86]:
```
"""
The combined data has no missing values because each data set is clean
"""
```

Out[86]:
```
'\nThe combined data has no missing values because each data set is clean\n'
```

In [87]:
```
# Checking for duplicates in the combined dataset

merged_df.duplicated().sum()    #There are no duplicates in the combined data set.
```

Out[87]:
```
0
```

In [88]:
```
# I'm dropping the column title as it has similar information with primary title.

merged_df.drop(columns=['title'],axis=1,inplace=True)

merged_df
```

Out[88]:

	tconst	primary_title	original_title	start_year	runtime_minutes	Genre1	Genre2	Genre3	averagerating	numvotes	studio	dor
0	tt0315642	Wazir	Wazir	2016	103.0	Action	Crime	Drama	7.1	15378	Relbig.	
1	tt0337692	On the Road	On the Road	2012	124.0	Adventure	Drama	Romance	6.1	37886	IFC	
2	tt4339118	On the Road	On the Road	2014	89.0	Drama	Blank Genre	Blank Genre	6.0	6	IFC	
3	tt5647250	On the Road	On the Road	2016	121.0	Drama	Blank Genre	Blank Genre	5.7	127	IFC	
4	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure	Comedy	Drama	7.3	275300	Fox	
...	...	...	...	...	...	...	...	...	...	...	...	
3022	tt8331988	The Chambermaid	La camarista	2018	102.0	Drama	Blank Genre	Blank Genre	7.1	147	FM	
3023	tt8404272	How Long Will I Love U	Chao shi kong tong ju	2018	101.0	Romance	Blank Genre	Blank Genre	6.5	607	WGUSA	
3024	tt8427036	Helicopter Eela	Helicopter Eela	2018	135.0	Drama	Blank Genre	Blank Genre	5.4	673	Eros	
3025	tt9078374	Last Letter	Ni hao, Zhihua	2018	114.0	Drama	Romance	Blank Genre	6.4	322	CL	
3026	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary	Music	Blank Genre	8.8	2067	Trafalgar	

3027 rows × 14 columns

In [89]:
```
"""
Data preprocess and cleaning is now done.The next step is to come up with solutions for Microsoft using the results we
"""
```

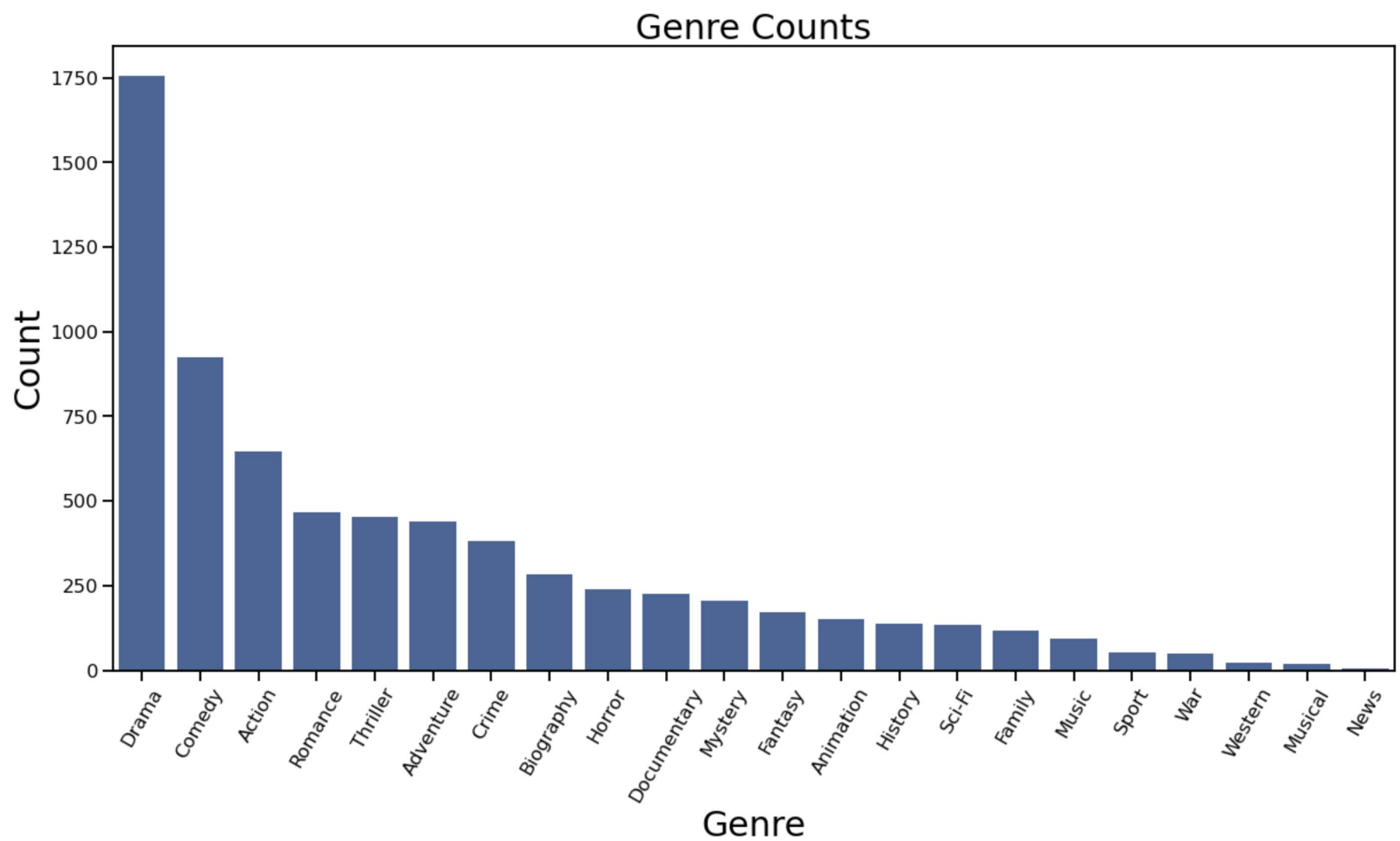
Out[89]:
```
'\nData preprocess and cleaning is now done.The next step is to come up with solutions for Microsoft using the resul
ts we have above\n'
```

Getting the best performing films using the count of genres

In [90]:
```
"""
To do that, we have to fetch all genres and their popularity(their counts)
I found the melt function in pandas which converts data from wide to long format
I'll use that for the genre columns
This will create two columns
The genre columns and their values. This will help us in visualization
"""
```

Out[90]:
```
"\nTo do that, we have to fetch all genres and their popularity(their counts)\nI found the melt function in pandas w
hich converts data from wide to long format \nI'll use that for the genre columns\nThis will create two columns\nThe
genre columns and their values. This will help us in visualization\n"
```

In [91]:
```
#Getting the counts of each genre

melted_df = pd.melt(merged_df, value_vars=['Genre1', 'Genre2', 'Genre3'])

genre_counts = melted_df['value'].value_counts()

genre_counts = genre_counts[genre_counts.index != 'Blank Genre']

# Creating a bar plot using Seaborn
plt.figure(figsize=(20,10))
sns.set_context('talk')
sns.barplot(x=genre_counts.index, y=genre_counts.values, color='#4060A0')

# The labels
plt.title('Genre Counts',fontsize=30)
plt.xlabel('Genre',fontsize=30)
plt.ylabel('Count',fontsize=30)
plt.xticks(rotation=60)

plt.show()
```



In [92]:
```
"""
The visualization above shows that drama, comedy and action are the most produced films.
"""
```

Out[92]:
```
'\nThe visualization above shows that drama, comedy and action are the most produced films.\n'
```

Getting genres with high ratings(Top five movie ratings)

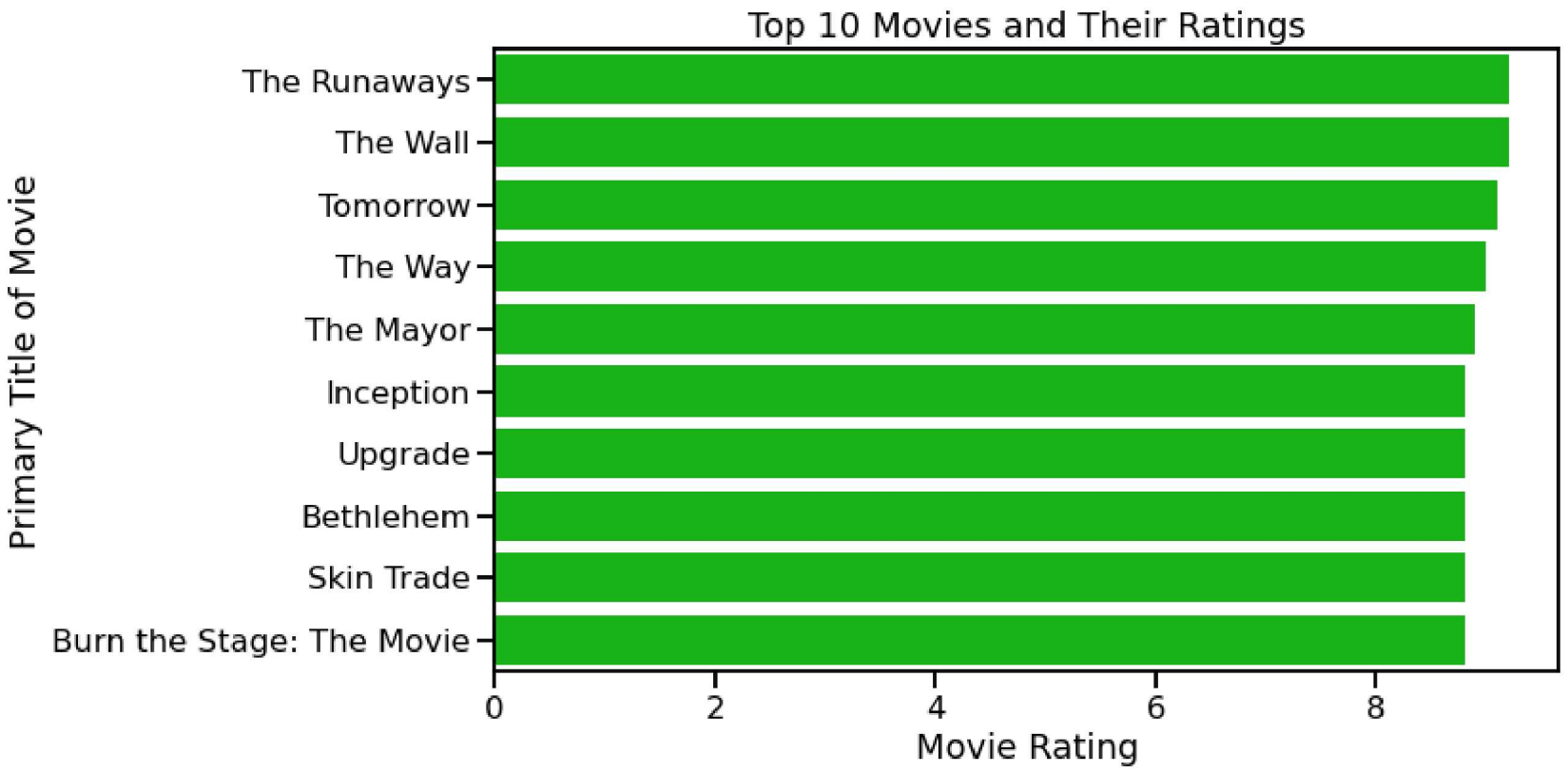
In [93]:
```
"""
Next, I'm getting the movie with the highest rating then from that I'll be able to get the genres of the movie.
"""
```

Out[93]:
```
"\nNext, I'm getting the movie with the highest rating then from that I'll be able to get the genres of the movi
e.\n"
```

In [94]:
```
merged_df['averagerating'].value_counts().head()
```

Out[94]:
```
6.8    143
6.4    142
6.6    141
6.3    140
7.2    136
Name: averagerating, dtype: int64
```

In [95]:
```
sorted_df = merged_df.sort_values(by='averagerating',ascending=False).head(10)

# A horizontal bar plot

plt.figure(figsize=(10,6))
sns.barplot(x=sorted_df['averagerating'],y=sorted_df['primary_title'],color='#00CC00',order=sorted_df['primary_title']
sns.set_context('talk')

# Labelling of the plot

plt.xlabel('Movie Rating')
plt.ylabel('Primary Title of Movie')
plt.title('Top 10 Movies and Their Ratings')

plt.show()
```



In [96]:
```
"""
Now because we have the top 10 movies with the highest rating, we can find their genres.
Although the ratings are very close so their visualization doesn't show alot of difference among the 10
"""
```

Out[96]:
```
"\nNow because we have the top 10 movies with the highest rating, we can find their genres.\nAlthough the ratings ar
e very close so their visualization doesn't show alot of difference among the 10\n"
```

In [97]:
```
#The code below accesses the genres of the top 10 movies with high ratings

sorted_df[['tconst','primary_title','averagerating','Genre1','Genre2','Genre3']]
```

Out[97]:

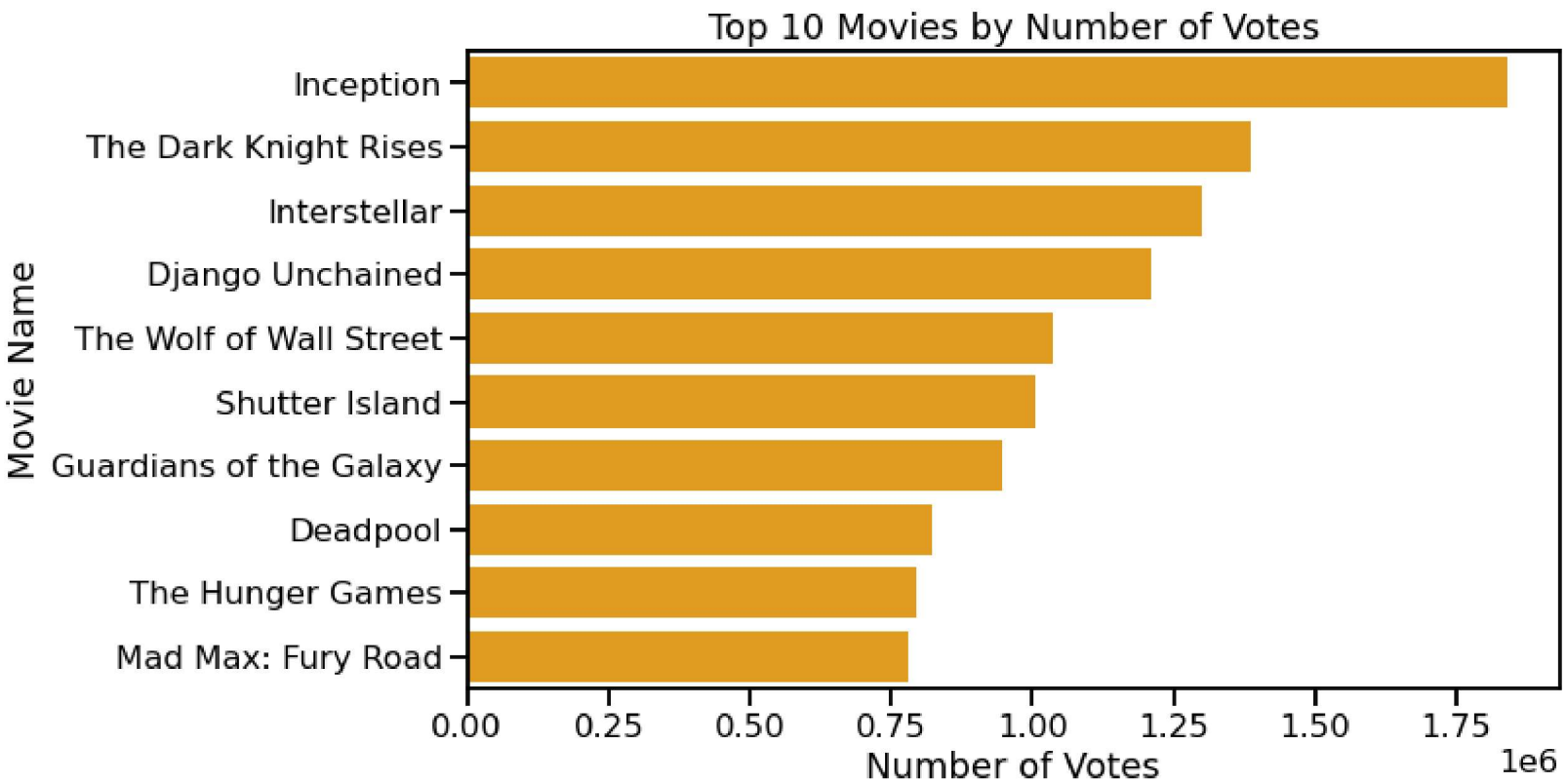
	tconst	primary_title	averagerating	Genre1	Genre2	Genre3
173	tt6168914	The Runaways	9.2	Adventure	Blank Genre	Blank Genre
658	tt1455256	The Wall	9.2	Documentary	Blank Genre	Blank Genre
2039	tt2831326	Tomorrow	9.1	Drama	Blank Genre	Blank Genre
638	tt6216234	The Way	9.0	Documentary	Blank Genre	Blank Genre
1186	tt1744662	The Mayor	8.9	Comedy	Documentary	Drama
514	tt1375666	Inception	8.8	Action	Adventure	Sci-Fi
2935	tt6739824	Upgrade	8.8	Drama	Blank Genre	Blank Genre
2150	tt5065924	Bethlehem	8.8	Comedy	Blank Genre	Blank Genre
834	tt1576702	Skin Trade	8.8	Documentary	Blank Genre	Blank Genre
3026	tt9151704	Burn the Stage: The Movie	8.8	Documentary	Music	Blank Genre

In [98]:
```
"""
From the genres, we can see that Documentary ranks highest. So apart from drama, comedy and action in genre counts, th
documentaries too.
"""
```

Out[98]:
```
'\nFrom the genres, we can see that Documentary ranks highest. So apart from drama, comedy and action in genre count
s, they can film \ndocumentaries too.\n'
```

Getting genres with highest number of votes(Top 5 genres)

In [99]:
```
# Sorting the dataframe in descending order by 'averagerating'
sorted_df = merged_df.sort_values(by='numvotes', ascending=False).head(10)


# Creating a horizontal bar plot

plt.figure(figsize=(10, 6))
sns.barplot(x=sorted_df['numvotes'], y=sorted_df['primary_title'], color='#FFA500',order=sorted_df['primary_title'])
sns.set_context('talk')


# Labelling the plot
plt.xlabel('Number of Votes')
plt.ylabel('Movie Name')
plt.title('Top 10 Movies by Number of Votes')

plt.show()
```



In [ ]:
```
```

In [100]:
```
sorted_df[['primary_title','numvotes','Genre1','Genre2','Genre3','numvotes']]
```

Out[100]:

	primary_title	numvotes	Genre1	Genre2	Genre3	numvotes
514	Inception	1841066	Action	Adventure	Sci-Fi	1841066
483	The Dark Knight Rises	1387769	Action	Thriller	Blank Genre	1387769
101	Interstellar	1299334	Adventure	Drama	Sci-Fi	1299334
1355	Django Unchained	1211405	Drama	Western	Blank Genre	1211405
163	The Wolf of Wall Street	1035358	Biography	Crime	Drama	1035358
253	Shutter Island	1005960	Mystery	Thriller	Blank Genre	1005960
1536	Guardians of the Galaxy	948394	Action	Adventure	Comedy	948394
607	Deadpool	820847	Action	Adventure	Comedy	820847
531	The Hunger Games	795227	Action	Adventure	Sci-Fi	795227
532	Mad Max: Fury Road	780910	Action	Adventure	Sci-Fi	780910

In [101]:
```
"""
From the visualization above, action and adventure rank highest. Since we had action in our top 3 genre counts, they c
film adventure oftenly.
"""
```

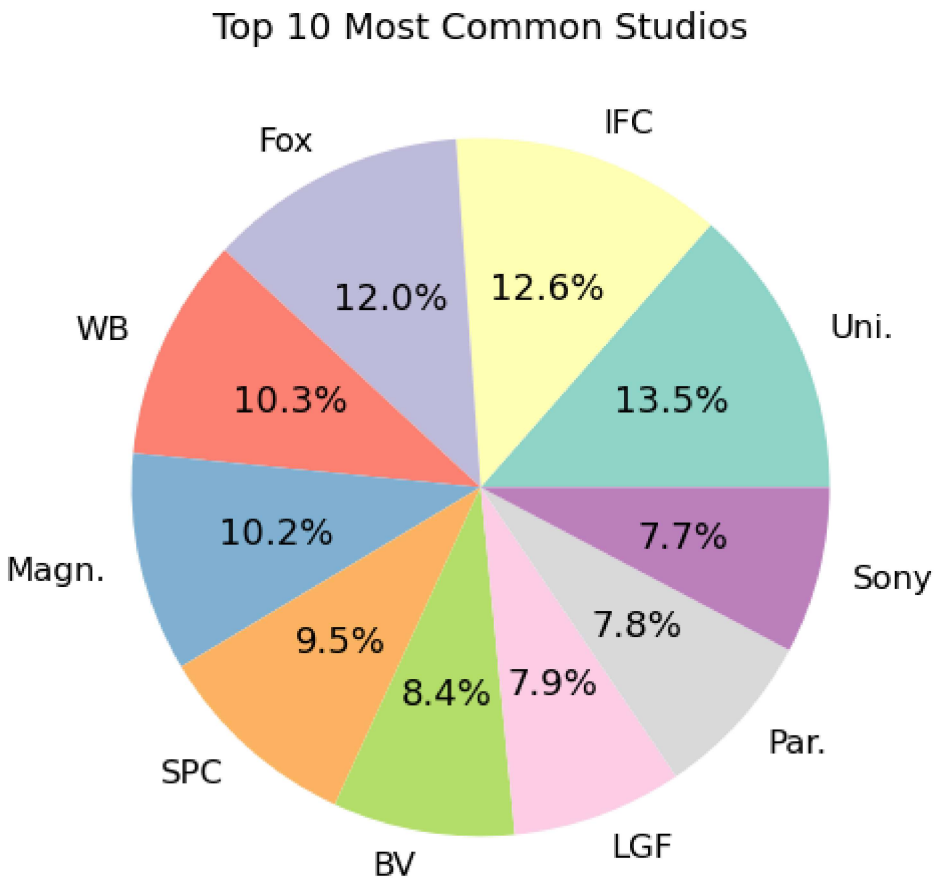
Out[101]:
```
'\nFrom the visualization above, action and adventure rank highest. Since we had action in our top 3 genre counts, t
hey can also \nfilm adventure oftenly.\n'
```

Getting The Best Studios

In [102]:
```
# This is code to confirm if our graph is giving us the correct minimum and maximum studios

print(merged_df['studio'].value_counts().head(10).idxmin())
print(merged_df['studio'].value_counts().head(10).idxmax())
```

```
Sony
Uni.
```

In [103]:
```
# Counting the occurrences of each studio
studio_counts = merged_df['studio'].value_counts()

# Limit the display to top 10 studios
top_n_studios = 10
studio_counts = studio_counts.head(top_n_studios)

# Using a Seaborn color palette for the pie chart
colors = sns.color_palette("Set3", len(studio_counts))

# Create a pie chart with Seaborn colors
plt.figure(figsize=(8, 8))
plt.pie(studio_counts.values, labels=studio_counts.index, autopct='%1.1f%%', colors=colors)
plt.title('Top 10 Most Common Studios')
plt.show()
```

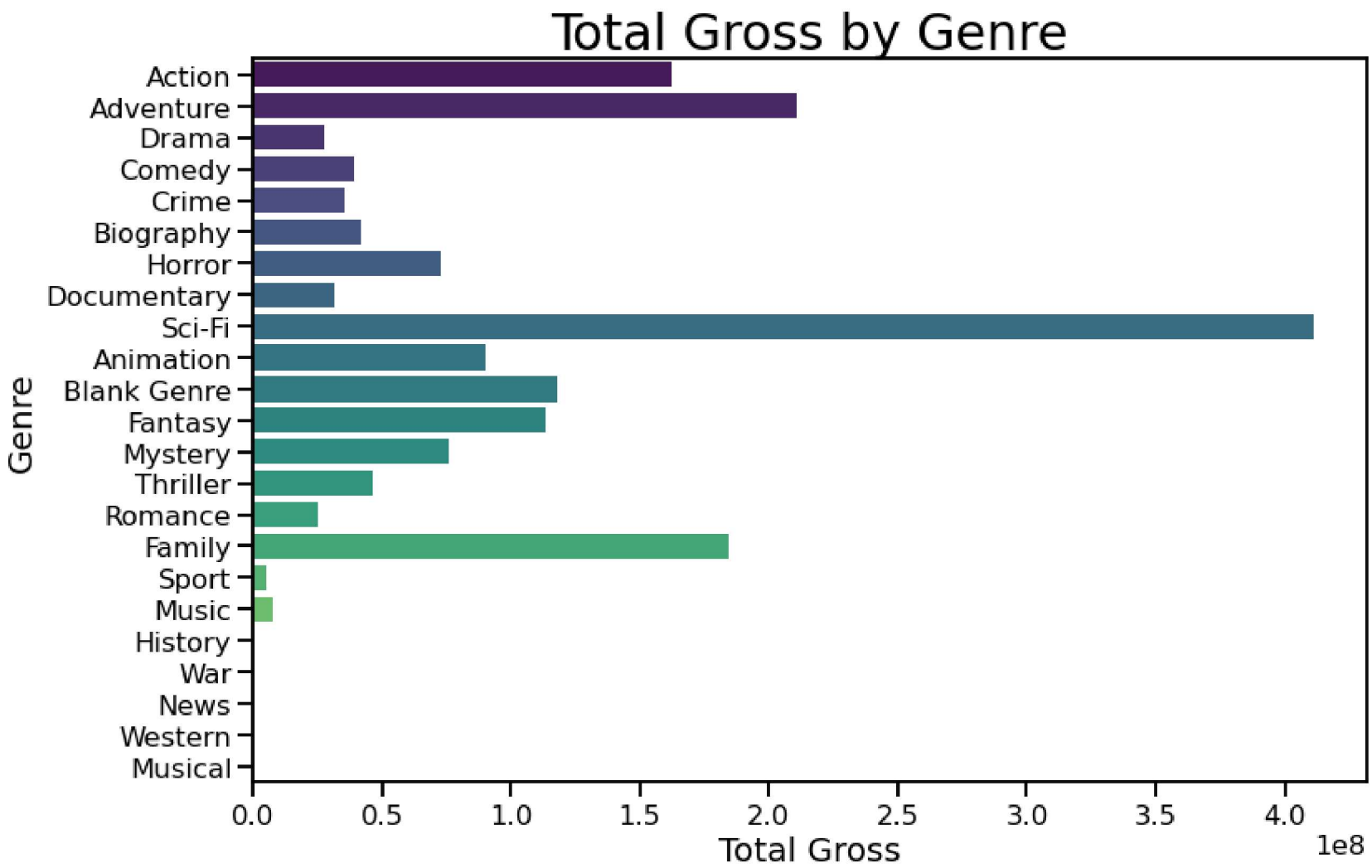


In [104]:
```
"""
Uni. Studio is the most common for filming.
The company can benchmark from Uni studio to know how they film because they most likely have success and expertise ir
"""
```

Out[104]:
```
'\nUni. Studio is the most common for filming.\nThe company can benchmark from Uni studio to know how they film beca
use they most likely have success and expertise in various genres.\n'
```

Perfomance on Both Domestic And Foreign Markets

In [105]:
```
# Convert 'domestic_gross' and 'foreign_gross' to numeric
merged_df['domestic_gross'] = pd.to_numeric(merged_df['domestic_gross'], errors='coerce')
merged_df['foreign_gross'] = pd.to_numeric(merged_df['foreign_gross'], errors='coerce')

# Calculate the total gross
merged_df['total_gross'] = merged_df['domestic_gross'] + merged_df['foreign_gross']

# Melt the DataFrame to have a single "Genre" column
melted_df = pd.melt(merged_df, value_vars=['Genre1', 'Genre2', 'Genre3'])

# Sorting 'total_gross' column in descending order
genre_total_gross = merged_df['total_gross'].sort_values(ascending=False)

# Create a horizontal bar plot
plt.figure(figsize=(12, 8))
sns.set_context('talk')
sns.barplot(x=genre_total_gross, y='value', data=melted_df, ci=None, palette='viridis')

# The labels
plt.title('Total Gross by Genre', fontsize=30)
plt.xlabel('Total Gross', fontsize=20)
plt.ylabel('Genre', fontsize=20)


plt.show()
```



In [106]:
```
"""
Above, we have Sci-Fi as the movie genre that has alot of revenue both domestically and internationally
"""
```

Out[106]:
```
'\nAbove, we have Sci-Fi as the movie genre that has alot of revenue both domestically and internationally\n'
```

Recommendations

1. Microsoft can focus on the following 6 genre of movies using the genre counts, ratings and voting analysis:
    - Drama
    - Comedy
    - Action
    - Adventure
    - Documentary
    - Sci-Fi
2. They can learn also get experience and help the following studios:
    - Uni.
    - IFC
    - Fox

- WB

  Those are the top 4 studios.

3. For more profits both locally and internationally, they can do more Scientific Fiction films

From that, they can start their movie studio