Final Project Submission

Please fill out:

- Student name: Richard Gachiri Muriithi
- Student pace: part time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

Overview

This projects examines the motion picture industry, which incorporates a wide range of genres. Every year, a number of studios attempt to gain a piece of the motion picture entertainment market by releasing these films. Microsoft can use this analysis to be able to know how to enter the motion picture industry

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. You are charged with exploring what types of films are currently doing the best at the box office. You must 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.

Objectives

1. Identify the total number of votes as the year go by.
2. Identify the top ten highest generating revenue domestic genre movies.
3. Identify the top ten highest generating revenue foreign genre movies.
4. Identify the top ten most popular movie genres which can be used when they develop their online streaming platform e.g Microsoft + or MS +
5. What are the best ten high generating revenue stuidos. This will be the total reveue, combine domestic and foriegn audience.
6. Identify the top ten highest generating revenue domestic audience stuidos.
7. Identify the top ten highest generating revenue foreign audience studios.

Data Understanding

Loading data

We load data into a data structure called a **dataframe**. A dataframe contains rows and columns; it can be easily manipulated hence appropriate for data analysis.

In [1]:
```
# Your code here - remember to use markdown cells for comments as well!
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('./Data/imdb.title.ratings.csv.gz')
```

In [3]:
```
df.info() # checks for the overview of the data
```

```
<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 [4]:
```
df.shape # checks for the number of rows and columns
```

Out[4]:
```
(73856, 3)
```

In [5]:
```
df.dtypes # dtype attribute checks the typer of data
```

Out[5]:
```
tconst            object
averagerating    float64
numvotes           int64
dtype: object
```

In [6]:
```
df.describe() # check for a statistical summary of the data
```

Out[6]:

	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

In [7]:
```
df1 = pd.read_csv('./Data/imdb.title.basics.csv.gz')
```

In [8]:
```
df1.info() # checks for the overview of the data
```

```
<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 [9]:
```
df1.shape # checks for the number of rows and columns
```

Out[9]:
```
(146144, 6)
```

In [10]:
```
df1.dtypes # dtype attribute checks the typer of data
```

Out[10]:
```
tconst              object
primary_title       object
original_title      object
start_year           int64
runtime_minutes    float64
genres              object
dtype: object
```

In [11]:
```
df1.describe() # check for a statistical summary of the data
```

Out[11]:

	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

In [12]:
```
df2 = pd.read_csv('./Data/bom.movie_gross.csv.gz')
```

In [13]:
```
df2.info() # checks for the overview of the data
```

```
<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 [14]:
```
df2.shape # checks for the number of rows and columns
```

Out[14]:
```
(3387, 5)
```

In [15]:
```
df2.dtypes # dtype attribute checks the typer of data
```

Out[15]:
```
title              object
studio             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object
```

In [16]:
```
df2.describe() # check for a statistical summary of the data
```

Out[16]:

	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

The IMDB ratings, title and movie gross data

imdb.title.ratings

In [17]:
```
df.head()#checks for the first 5 rows imdb.title.ratings
```

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]:
```
df.tconst.nunique() ## all the tconst are unique from each other
```

Out[18]:
```
73856
```

In [19]:
```
df.averagerating.describe()
```

Out[19]:
```
count    73856.000000
mean         6.332729
std          1.474978
min          1.000000
25%          5.500000
50%          6.500000
75%          7.400000
max         10.000000
Name: averagerating, dtype: float64
```

In [20]:
```
df.averagerating.value_counts()
```

Out[20]:
```
7.0     2262
6.6     2251
7.2     2249
6.8     2239
6.5     2221
        ...
9.6       18
10.0      16
9.8       15
9.7       12
9.9        5
Name: averagerating, Length: 91, dtype: int64
```

imdb.title.basics

The df1 dataset, ratings contains movies with start years from 2010 to 2115 and also includes a wide varierty of primary title, original title and genres

In [21]:
```
df1.head() #checks for the first 5 rows imdb.title.basics
```

Out[21]:

	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 [22]:
```
df1.tconst.nunique() # the tconst are unique from each other
```

Out[22]:
```
146144
```

In [23]:
```
df1.primary_title.value_counts()
```

Out[23]:
```
Home                                                24
Broken                                              20
The Return                                          20
Alone                                               16
Homecoming                                          16
                                                    ..
Fuusen'inu Tinî: Nandaka fushigina kyouryuu no kuni!  1
Gila Jiwa: Illogically Sane                          1
The Rolston Sessions                                 1
Hannah Arendt                                        1
Irie the Film: Journey of a King                     1
Name: primary_title, Length: 136071, dtype: int64
```

In [24]:
```
df1.original_title.value_counts()
```

Out[24]:
```
Broken                              19
Home                                18
The Return                          17
Alone                               13
The Gift                            13
                                    ..
Desire in New York                   1
Gathering                            1
The Manhattan Front                  1
Nunuko no seisen Harajuku Story      1
Irie the Film: Journey of a King     1
Name: original_title, Length: 137773, dtype: int64
```

In [25]:
```
df1.start_year.value_counts()
```

Out[25]:
```
2017    17504
2016    17272
2018    16849
2015    16243
2014    15589
2013    14709
2012    13787
2011    12900
2010    11849
2019     8379
2020      937
2021       83
2022       32
2023        5
2024        2
2027        1
2026        1
2025        1
2115        1
Name: start_year, dtype: int64
```

In [26]:
```
df1.genres.value_counts()[:10]
```

Out[26]:
```
Documentary                 32185
Drama                       21486
Comedy                       9177
Horror                       4372
Comedy,Drama                 3519
Thriller                     3046
Action                       2219
Biography,Documentary        2115
Drama,Romance                2079
Comedy,Drama,Romance         1558
Name: genres, dtype: int64
```

bom.movie_gross

The df2 dataset, Movie gross contains title movies from between the year 2010 to the year 2018. It also contains a wide variety of the movie titles, the studios, the domestics gross and foreign gross

In [27]:
```
df2.head() #checks for the first 5 rows bom.movie_gross
```

Out[27]:

	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 [28]:
```
#df2['Years'] =  df2['year']
#df2['Years'].value_counts()

df2.year.value_counts()
```

Out[28]:
```
2015    450
2016    436
2012    400
2011    399
2014    395
2013    350
2010    328
2017    321
2018    308
Name: year, dtype: int64
```

In [29]:
```
df2.title.value_counts()
```

Out[29]:
```
Bluebeard                                                 2
Blue Valentine                                            1
Rampage (2018)                                            1
Badmaash Company                                          1
10 Cloverfield Lane                                       1
                                                         ..
Cold War 2                                                1
The Man Who Fell to Earth (35th anniversary re-issue)    1
The Girl in the Book                                      1
The Best Offer                                            1
2 Days in New York                                        1
Name: title, Length: 3386, dtype: int64
```

All the title of the movies appear once except from the Bluebeard movie title

In [30]:
```
df2.studio.value_counts()
```

Out[30]:
```
IFC      166
Uni.     147
WB       140
Magn.    136
Fox      136
        ...
CF&SR      1
CLF        1
RME        1
BGP        1
KS         1
Name: studio, Length: 257, dtype: int64
```

In [31]:
```
df2.domestic_gross.value_counts()
```

Out[31]:
```
1100000.0      32
1000000.0      30
1300000.0      30
1200000.0      25
1400000.0      23
               ..
68800.0         1
87000000.0      1
739000.0        1
336000000.0     1
727000.0        1
Name: domestic_gross, Length: 1797, dtype: int64
```

In [32]:
```
df2.domestic_gross.describe()
```

Out[32]:
```
count    3.359000e+03
mean     2.874585e+07
std      6.698250e+07
min      1.000000e+02
25%      1.200000e+05
50%      1.400000e+06
75%      2.790000e+07
max      9.367000e+08
Name: domestic_gross, dtype: float64
```

In [33]:
```
df2.foreign_gross.value_counts()
```

Out[33]:
```
1200000      23
1100000      14
4200000      12
1900000      12
1300000      11
             ..
167100000     1
193100000     1
78300000      1
166800000     1
45400000      1
Name: foreign_gross, Length: 1204, dtype: int64
```

In [34]:
```
df2.foreign_gross.describe()
```

Out[34]:
```
count        2037
unique       1204
top       1200000
freq           23
Name: foreign_gross, dtype: object
```

Data Preparation

Data Cleaning

From the above data sets we will check first for missing data values in all the datasets. After identifying the missing values we will clean the data sets by either:

1. Drop missing values, either columns or rows.
2. Fill/replace with mean/median/mode/, backward fill/forward fill, certain value
3. Interpolate

In [35]:
```
df.isnull().sum() # Gets count of missing values, imdb.title.ratings.csv
```

Out[35]:
```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

We can see that from the dataset, imdb.title.ratings they is no missing values.

In [36]:
```
df1.isnull().sum() # Gets count of missing values, imdb.title.basics.csv
```

Out[36]:
```
tconst                 0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
```

We can see that from the dataset, imdb.title.basics they is missing values in:

1. original_title, 21 missing values
2. runtime_minutes, 31739 missing values
3. genres, 5408 missing values.

In [37]:
```
df2.isnull().sum() # Gets count of missing values, born.movie_gross.cvs
```

Out[37]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

We can see that from the dataset, born.movie_gross.cvs they is missing values in:

1. studio, 5 missing values
2. domestic_gross, 28 missing values
3. foreign_gross, 1350 missing values.

In [38]:
```
#imdb.title.ratings.csv
df.isna().sum() # Gets count of missing values
```

Out[38]:
```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

In [39]:
```
#imdb.title.basics.csv
df1.isna().sum()# Gets count of missing values
```

Out[39]:
```
tconst                 0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
```

In [40]:
```
#born.movie_gross.cvs
df2.isna().sum() # Gets count of missing values
```

Out[40]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [41]:
```
#imdb.title.ratings.csv
missing_values = df.isna() # Checks missing values
missing_values.head()
```

Out[41]:

	tconst	averagerating	numvotes
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

In [42]:
```
#imdb.title.basics.csv
missing_values = df1.isna() # Checks missing values
missing_values.head()
```

Out[42]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	True	False
4	False	False	False	False	False	False

In [43]:
```
#born.movie_gross.cvs
missing_values = df2.isna() # Checks missing values
missing_values.head()
```

Out[43]:

	title	studio	domestic_gross	foreign_gross	year
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False

Below we will check the percentage of missing value of the datasets **imdb.title.ratings, imdb.title.basic and born.movie_gross**.

This will help use in knowing how to deal with the missing values in the datasets provided.

In [44]:
```
def missing_values(data):
    # identify the total missing values per column
    # sort in order
    miss = data.isnull().sum().sort_values(ascending = False)

    # calculate percentage of the missing values
    percentage_miss = (data.isnull().sum() / len(data)).sort_values(ascending

    # store in a dataframe
    missing = pd.DataFrame({"Missing Values": miss, "Percentage": percentage_m

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

    return missing
```

In [45]:
```
missing_data = missing_values(df)
missing_data
```

Out[45]:

index	Missing Values	Percentage

From the df dataframe, imdb.title.ratings, they are no missing values at all.

In [46]:
```
missing_data1 = missing_values(df1)
missing_data1
```

Out[46]:

	index	Missing Values	Percentage
0	runtime_minutes	31739	0.217176
1	genres	5408	0.037005
2	original_title	21	0.000144

From the above results, df1 dataframe imdb.title.basics we can see the columns **rutime_minutes, genres, original title** have certain percentage of missing values. They are:

1. runtime_minutes 22%
2. genres 3.7%
3. original_title 0.0144%

In [47]:
```
missing_data2 = missing_values(df2)
missing_data2
```

Out[47]:

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

From the above results, df2 dataframe bom.movie_gross we can see the columns **foreign_gross, domestic_gross, studio** have certain percentage of missing values. They are:

1. foreign_gross 39%
2. domestic_gross 0.8267%
3. studio 0.01476%

Dealing with the missing values.

df1 dataframe imdb.title.basics

In [48]:
```
missing_data1
```

Out[48]:

	index	Missing Values	Percentage
0	runtime_minutes	31739	0.217176
1	genres	5408	0.037005
2	original_title	21	0.000144

In [49]:
```
df1.shape
```

Out[49]:
```
(146144, 6)
```

For the column **runtime_mintues** with a very low percentage, we can replace it with the mean score of the runtime minutes

In [50]:
```
df1.runtime_minutes.fillna(df1.runtime_minutes.mean(), inplace = True)
```

since columns **genres, original title** have very low percentage we cand drop the rows where there is missing values

In [51]:
```
df1.dropna(axis = 0, subset=['genres'], inplace=True)
df1.dropna(axis = 0, subset=['original_title'], inplace=True)

## Using a function
#def drop_rows_missing_values(df, columns):
#  """Drops rows in columns with missing values.
#  simple function to drop the rows wtih missing values
#  """
#  return df.dropna(subset=columns, inplace = True)

#drop_rows = drop_rows_with_missing_values(df1,['genres','original_title'])
```

In [52]:
```
missing_values(df1)
```

Out[52]:

index	Missing Values	Percentage

In [53]:
```
df1.isna().sum()
```

Out[53]:
```
tconst             0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

In [54]:
```
df1.shape
```

Out[54]:
```
(140734, 6)
```

df2 dataframe bom.movie_gross

In [55]:
```
missing_data2
```

Out[55]:

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

For the column **foreign_gross** with a very low percentage, we can replace it with the mean score of the runtime minutes

In [56]:
```
df2.foreign_gross
```

Out[56]:
```
0        652000000
1        691300000
2        664300000
3        535700000
4        513900000
           ...
3382           NaN
3383           NaN
3384           NaN
3385           NaN
3386           NaN
Name: foreign_gross, Length: 3387, dtype: object
```

In [57]:
```
df2.shape
```

Out[57]:
```
(3387, 5)
```

For the column **domestic_gross, studio** with a we will drop the rows with the missing values

In [58]:
```
df2.dropna(axis = 0, subset=['domestic_gross'], inplace=True)
df2.dropna(axis = 0, subset=['studio'], inplace=True)


## Using a function to drop rows#
#def drop_rows_missing_values(df, columns):
#  """Drops rows in columns with missing values.
#  simple function to drop the rows wtih missing values
#  """
#  return df.dropna(axis = 0, subset=columns, inplace = True)

#drop_rows = drop_rows_with_missing_values(df2,['domestic_gross','studio'])
```

In [59]:
```
missing_values(df2)
```

Out[59]:

	index	Missing Values	Percentage
0	foreign_gross	1349	0.401967

For the column **foreign_gross** with a very low percentage, we can replace it with the median of its column. But the data type for this column is an object(string) thus we need to convert the datatype

In [60]:
```
df2.isna().sum()
```

Out[60]:
```
title                0
studio               0
domestic_gross       0
foreign_gross     1349
year                 0
dtype: int64
```

Since for **foreign_gross** the data type is object, we covert it float64 so as to be able to replace the missing values using the median.

In [61]:
```
df2.foreign_gross = df2.foreign_gross.str.replace(',', "")
```

In [62]:
```
df2.foreign_gross = pd.to_numeric(df2.foreign_gross)
```

In [63]:
```
df2.dtypes
```

Out[63]:
```
title              object
studio             object
domestic_gross    float64
foreign_gross     float64
year                int64
dtype: object
```

In [64]:
```
df2.foreign_gross.fillna(df2.foreign_gross.median(), inplace = True)
```

In [65]:
```
missing_values(df2)
```

Out[65]:

index	Missing Values	Percentage

In [66]:
```
df2.isna().sum()
```

Out[66]:
```
title             0
studio            0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

Merging Datasets

In [67]:
```
merge_data = pd.merge(df, df1, on = 'tconst', how = 'inner')
merge_data
```

Out[67]:

	tconst	averagerating	numvotes	primary_title	original_title	start_year	runtime_minu
0	tt10356526	8.3	31	Laiye Je Yaarian	Laiye Je Yaarian	2019	117.000(
1	tt10384606	8.9	559	Borderless	Borderless	2019	87.000(
2	tt1042974	6.4	20	Just Inès	Just Inès	2010	90.000(
3	tt1043726	4.2	50352	The Legend of Hercules	The Legend of Hercules	2014	99.000(
4	tt1060240	6.5	21	Até Onde?	Até Onde?	2011	73.000(
...	...	...	...	...	...	...	
73047	tt9805820	8.1	25	Caisa	Caisa	2018	84.000(
73048	tt9844256	7.5	24	Code Geass: Lelouch of the Rebellion - Glorifi...	Code Geass: Lelouch of the Rebellion Episode III	2018	120.000(
73049	tt9851050	4.7	14	Sisters	Sisters	2019	86.187:
73050	tt9886934	7.0	5	The Projectionist	The Projectionist	2019	81.000(
73051	tt9894098	6.3	128	Sathru	Sathru	2019	129.000(

73052 rows × 8 columns

In [68]:
```
df2.rename(columns={'title': 'original_title'}, inplace=True)
```

In [69]:
```
merged_df3 = pd.merge(merge_data, df2, on= 'original_title', how = 'inner')
merged_df3
```

Out[69]:

	tconst	averagerating	numvotes	primary_title	original_title	start_year	runtime
0	tt1043726	4.2	50352	The Legend of Hercules	The Legend of Hercules	2014	
1	tt1171222	5.1	8296	Baggage Claim	Baggage Claim	2013	
2	tt1210166	7.6	326657	Moneyball	Moneyball	2011	
3	tt1212419	6.5	87288	Hereafter	Hereafter	2010	
4	tt1229238	7.4	428142	Mission: Impossible - Ghost Protocol	Mission: Impossible - Ghost Protocol	2011	
...	...	...	...	...	...	...	
2419	tt3142688	5.8	5841	Finding Fanny	Finding Fanny	2014	
2420	tt3399916	6.3	4185	The Dead Lands	The Dead Lands	2014	
2421	tt3748512	7.4	4977	Hitchcock/Truffaut	Hitchcock/Truffaut	2015	
2422	tt7008872	7.0	18768	Boy Erased	Boy Erased	2018	
2423	tt8011712	7.4	54	The Past	The Past	2018	

2424 rows × 12 columns

In [70]:
```
merged_df3.start_year.value_counts()
```

Out[70]:
```
2014    317
2015    296
2011    295
2010    283
2012    280
2016    279
2013    257
2017    225
2018    181
2019     11
Name: start_year, dtype: int64
```

In [71]:
```
merged_df3.describe(include = 'all')
```

Out[71]:

	tconst	averagerating	numvotes	primary_title	original_title	start_year	runtime_
count	2424	2424.000000	2.424000e+03	2424	2424	2424.000000	242
unique	2424	NaN	NaN	2136	2122	NaN	
top	tt4094724	NaN	NaN	Split	Split	NaN	
freq	1	NaN	NaN	6	6	NaN	
mean	NaN	6.408581	7.334540e+04	NaN	NaN	2013.783003	10
std	NaN	1.042490	1.350694e+05	NaN	NaN	2.492569	2
min	NaN	1.600000	5.000000e+00	NaN	NaN	2010.000000	
25%	NaN	5.800000	3.919000e+03	NaN	NaN	2012.000000	9
50%	NaN	6.500000	2.121800e+04	NaN	NaN	2014.000000	10
75%	NaN	7.100000	8.138900e+04	NaN	NaN	2016.000000	11
max	NaN	9.200000	1.841066e+06	NaN	NaN	2019.000000	18

In [72]:
```
merged_df3.columns
```

Out[72]:
```
Index(['tconst', 'averagerating', 'numvotes', 'primary_title',
       'original_title', 'start_year', 'runtime_minutes', 'genres', 'studio',
       'domestic_gross', 'foreign_gross', 'year'],
      dtype='object')
```

Check for duplicate rows in the merged data set, merged_df3

In [73]:
```
duplicate_rows = merged_df3[merged_df3.duplicated()]
duplicate_rows
```

Out[73]:

tconst	averagerating	numvotes	primary_title	original_title	start_year	runtime_minutes	genre

After merging the datasets, they are now duplicates with the **merged_df3** dataset

Analysis

In [74]:
```
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Title of movies whose runtime >=180

In [75]:
```
merged_df3[merged_df3.runtime_minutes >= 180] ['primary_title']
```

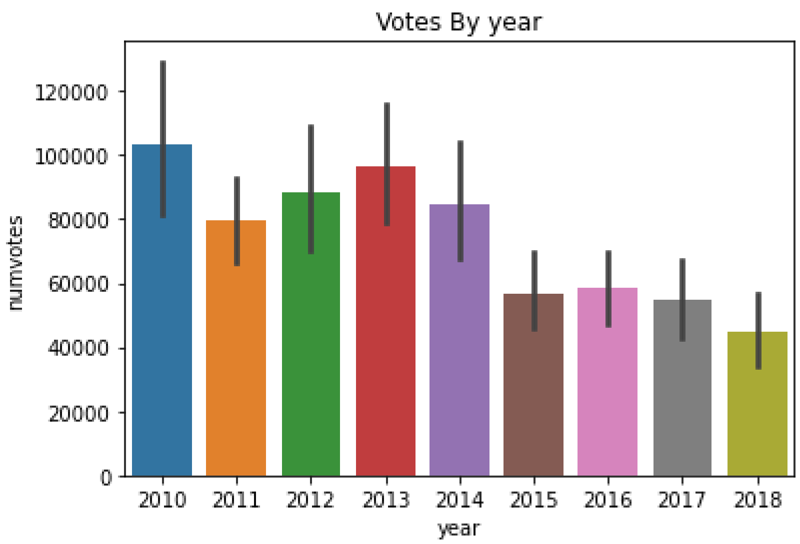
Out[75]:
```
268                            Aurora
652                Bhaag Milkha Bhaag
1489           The Wolf of Wall Street
1688    M.S. Dhoni: The Untold Story
Name: primary_title, dtype: object
```

Which year has the highet average votes

In [76]:
```
merged_df3.groupby('year')['numvotes'].mean().sort_values(ascending = False)
```

Out[76]:
```
year
2010    103439.476636
2013     96635.246964
2012     88273.095070
2014     84535.286765
2011     79584.155172
2016     58226.732673
2015     56661.488439
2017     54603.632411
2018     44555.581395
Name: numvotes, dtype: float64
```

In [77]:
```
sns.barplot(x = 'year', y = 'numvotes', data = merged_df3)
plt.title("Votes By year")
plt.show()
```



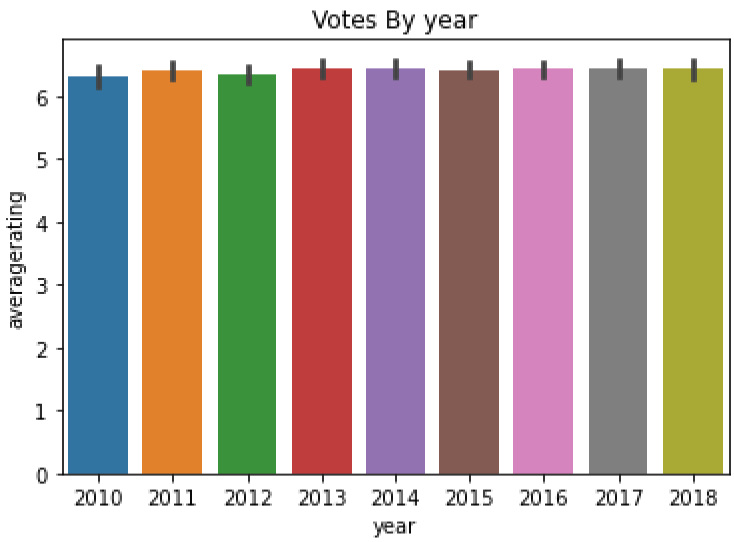
As the years go on, thet has been less and less num of votes towards to movies. This may be due to many factors, for example, some say older movies were better made than the new ones.

Year with the highest average rating

In [78]:
```
year_highest_rating = merged_df3.groupby('year')['averagerating'].mean().sort_
year_highest_rating.head(10)
```

Out[78]:
```
year
2017    6.446640
2013    6.440891
2018    6.436744
2014    6.434191
2016    6.424092
2015    6.420520
2011    6.406897
2012    6.347183
2010    6.307944
Name: averagerating, dtype: float64
```

In [79]:
```
sns.barplot(x = 'year', y = 'averagerating', data = merged_df3)
plt.title("Votes By year")
plt.show()
```



Top 20 Highest revenue movies

In [80]:
```
merged_df3['total_revenue'] = merged_df3['domestic_gross'] + merged_df3['forei
```

In [81]:
```
top_20 = merged_df3.nlargest(20, 'total_revenue')[['original_title', 'total_re
```

In [82]:

```
top_20
```

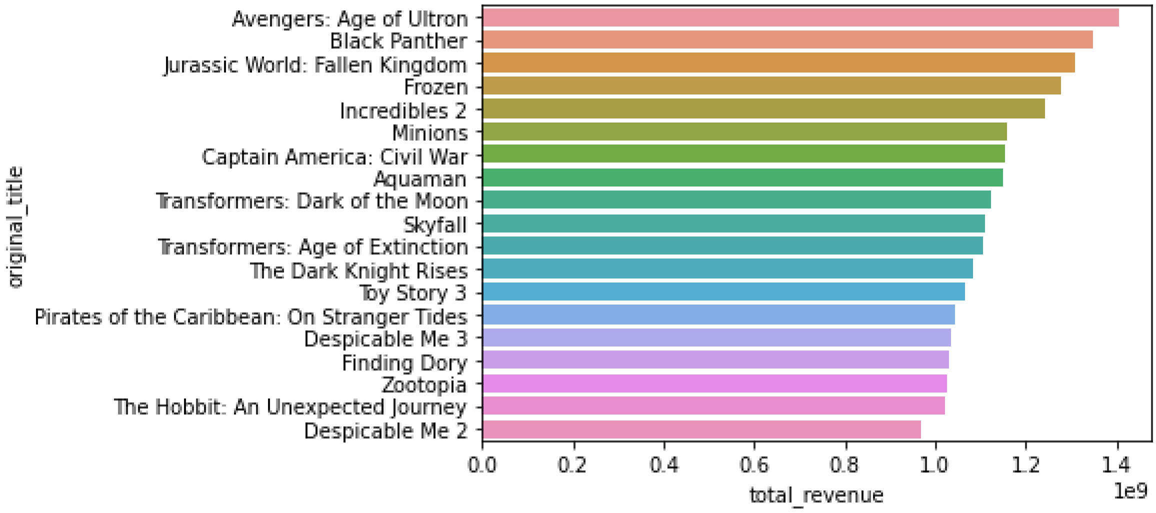
Out[82]:

	total_revenue
original_title	
Avengers: Age of Ultron	1.405400e+09
Black Panther	1.347000e+09
Jurassic World: Fallen Kingdom	1.309500e+09
Frozen	1.276400e+09
Frozen	1.276400e+09
Incredibles 2	1.242800e+09
Minions	1.159400e+09
Captain America: Civil War	1.153300e+09
Aquaman	1.147800e+09
Transformers: Dark of the Moon	1.123800e+09
Skyfall	1.108600e+09
Transformers: Age of Extinction	1.104000e+09
The Dark Knight Rises	1.084900e+09
Toy Story 3	1.067000e+09
Pirates of the Caribbean: On Stranger Tides	1.045700e+09
Despicable Me 3	1.034800e+09
Finding Dory	1.028600e+09
Zootopia	1.023800e+09
The Hobbit: An Unexpected Journey	1.021100e+09
Despicable Me 2	9.708000e+08

Created a new column called **total_revenue** to show the total movie gross of both the domestic and foreign makert

In [83]:
```
sns.barplot(x = 'total_revenue', y = top_20.index, data = top_20)
```

Out[83]:
```
<AxesSubplot:xlabel='total_revenue', ylabel='original_title'>
```



Top 10 Studio with the highest domestic_gross revenue

In [84]:
```
studio_revenue_domestic = merged_df3.groupby('studio')['domestic_gross'].mean(
```

In [85]:
```
studio_revenue_domestic
```

Out[85]:
```
studio
P/DW        1.682900e+08
BV          1.642396e+08
WB          9.107521e+07
WB (NL)     8.805417e+07
Uni.        8.651773e+07
                ...
Icar.       3.200000e+03
ALP         2.800000e+03
First       2.000000e+03
Shout!      1.500000e+03
DR          8.000000e+02
Name: domestic_gross, Length: 188, dtype: float64
```

In [86]:
```
#top_ten_domestic = studio_revenue_domestic.head(10)
plt.(figsize=(15, 6))
plt.bar(studio_revenue_domestic.index[:10], studio_revenue_domestic.values[:10

# Add labels and title to the plot.
plt.xlabel('Studio')
plt.ylabel('Domestic Gross')
plt.title('Top 10 Studio Revenues - Domestic Gross')

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

```
  File "<ipython-input-86-b12c69990729>", line 2
    plt.(figsize=(15, 6))
        ^
SyntaxError: invalid syntax
```

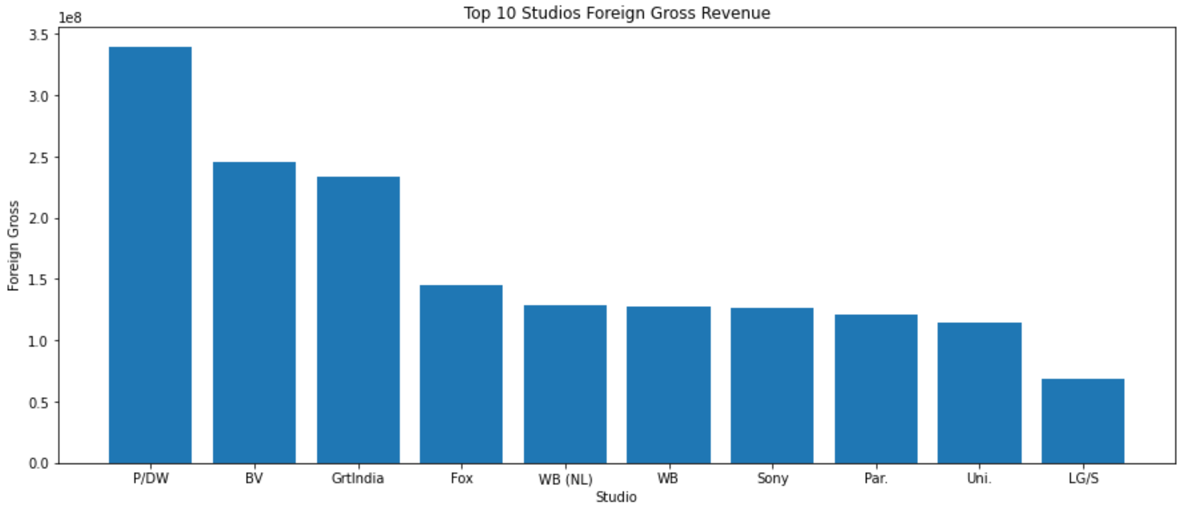
Top 10 studio with the highest foreign_gross revenue

In [87]:
```
studio_revenue_foreign = merged_df3.groupby('studio')['foreign_gross'].mean().
```

In [88]:
```
studio_revenue_foreign
```

Out[88]:
```
studio
P/DW        3.393600e+08
BV          2.459918e+08
GrtIndia    2.340000e+08
Fox         1.447408e+08
WB (NL)     1.283417e+08
                ...
Viv.        6.030000e+04
First       5.360000e+04
WOW         1.860000e+04
FOAK        1.730000e+04
ITL         1.180000e+04
Name: foreign_gross, Length: 188, dtype: float64
```

In [89]:
```
#top_ten_foreign = studio_revenue_foreign.head(10)
plt.figure(figsize=(15, 6))
plt.bar(studio_revenue_foreign.index[:10], studio_revenue_foreign.values[:10])

# Add labels and title to the plot.
plt.xlabel('Studio')
plt.ylabel('Foreign Gross')
plt.title('Top 10 Studios Foreign Gross Revenue')

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

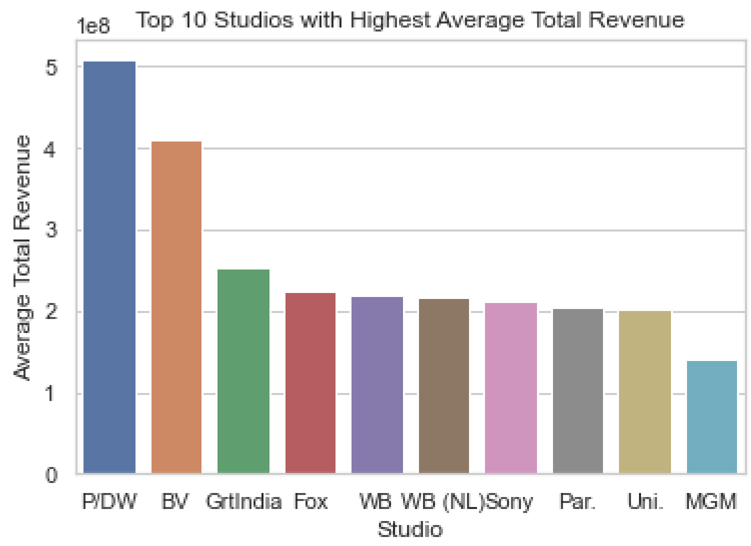


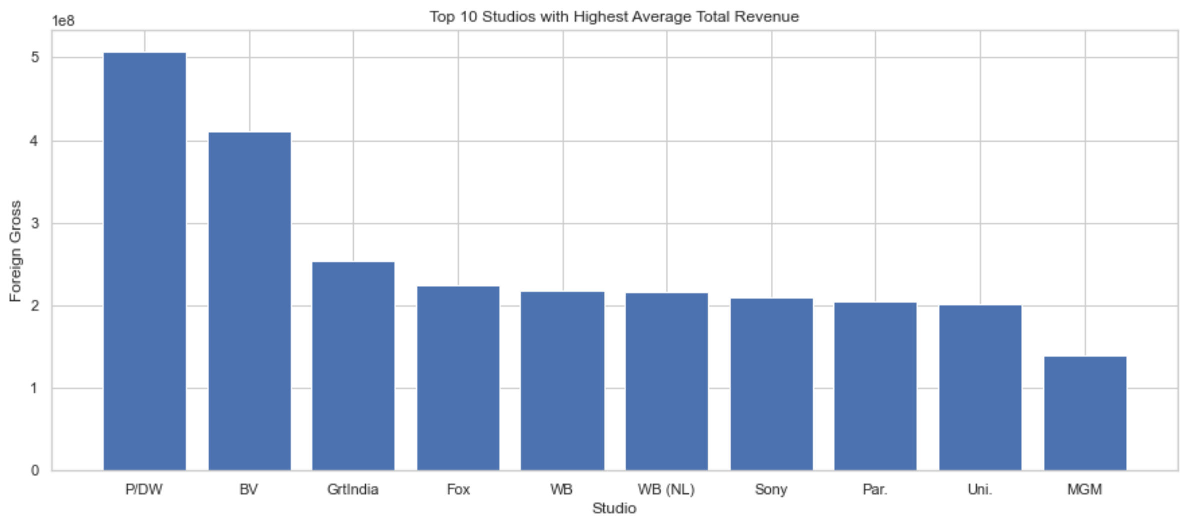
Top 10 Studios with the highest revenue

In [90]:
```
highest_studio_revenue = merged_df3.groupby('studio')['total_revenue'].mean().
highest_studio_revenue.head(10)
```

Out[90]:
```
studio
P/DW        5.076500e+08
BV          4.102313e+08
GrtIndia    2.542000e+08
Fox         2.241580e+08
WB          2.182487e+08
WB (NL)     2.163958e+08
Sony        2.104489e+08
Par.        2.046458e+08
Uni.        2.012829e+08
MGM         1.393000e+08
Name: total_revenue, dtype: float64
```

In [91]:
```
sns.set_theme(style="whitegrid")
sns.barplot(x=highest_studio_revenue.index[:10], y=highest_studio_revenue.valu

# Add labels and title to the plot.
plt.xlabel('Studio')
plt.ylabel('Average Total Revenue')
plt.title('Top 10 Studios with Highest Average Total Revenue')

# Show the plot.
plt.show()

## matplotib
plt.figure(figsize=(15, 6))
plt.bar(highest_studio_revenue.index[:10], highest_studio_revenue.values[:10])

# Add labels and title to the plot.
plt.xlabel('Studio')
plt.ylabel('Foreign Gross')
plt.title('Top 10 Studios with Highest Average Total Revenue')

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





Genres Section

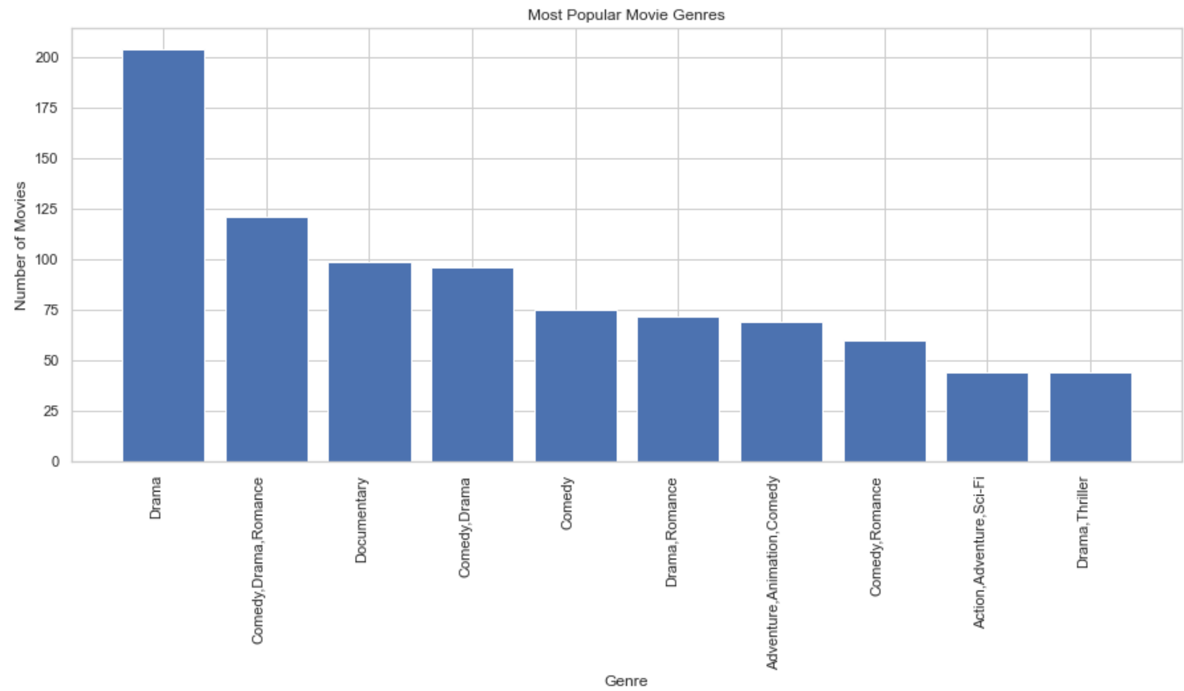
Top 10 Most Popular Movie genres

In [92]:
```
def genre_counts(merged_df):
    genre_counts = merged_df3.genres.value_counts()

    return genre_counts
```

In [93]:
```
genre_counts = genre_counts(merged_df3).head(10)
genre_counts
```

Out[93]:
```
Drama                          204
Comedy,Drama,Romance           121
Documentary                     99
Comedy,Drama                    96
Comedy                          75
Drama,Romance                   72
Adventure,Animation,Comedy      69
Comedy,Romance                  60
Action,Adventure,Sci-Fi         44
Drama,Thriller                  44
Name: genres, dtype: int64
```

In [94]:
```
plt.figure(figsize=(15,6))
plt.bar(x = genre_counts.index, height = genre_counts.values)
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.xticks(rotation = 90, ha = 'right')
plt.title('Most Popular Movie Genres')
plt.show()
```



Genres with the highest total revenue

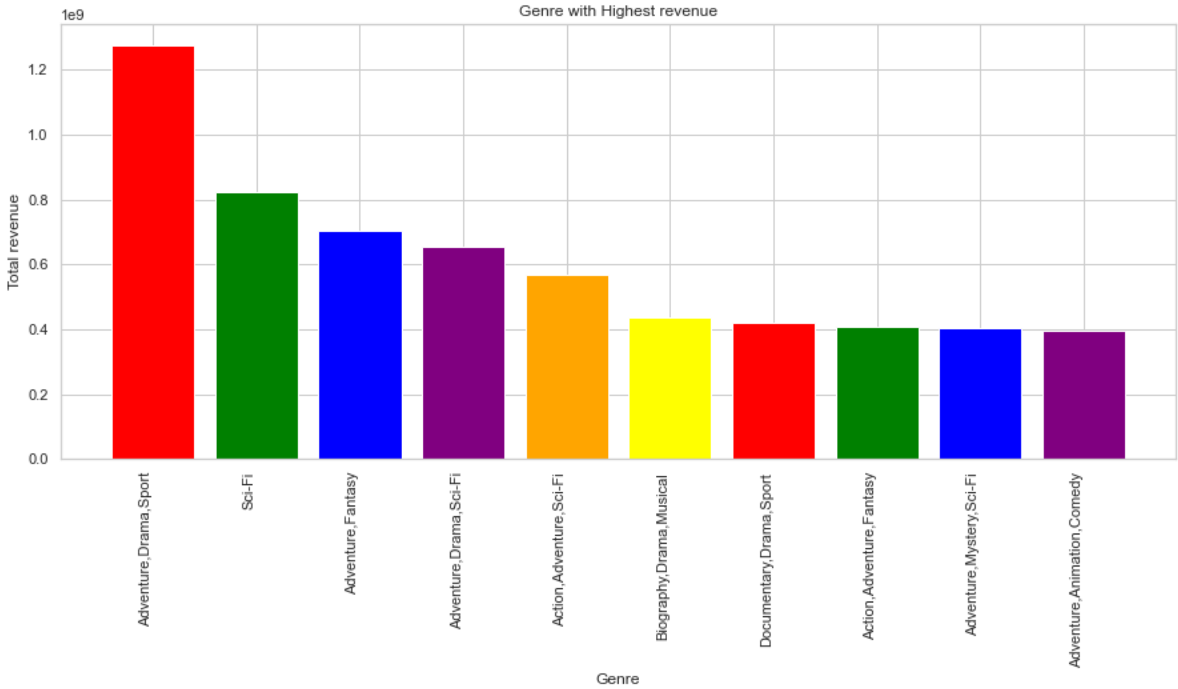
In [95]:
```
merged_df3.columns
```

Out[95]:
```
Index(['tconst', 'averagerating', 'numvotes', 'primary_title',
       'original_title', 'start_year', 'runtime_minutes', 'genres', 'studio',
       'domestic_gross', 'foreign_gross', 'year', 'total_revenue'],
      dtype='object')
```

In [96]:
```
high_revenue_genre = merged_df3.groupby('genres')['total_revenue'].mean().sort
high_revenue_genre.head(10)
```

Out[96]:
```
genres
Adventure,Drama,Sport         1.276400e+09
Sci-Fi                        8.219000e+08
Adventure,Fantasy             7.040333e+08
Adventure,Drama,Sci-Fi        6.537500e+08
Action,Adventure,Sci-Fi       5.688864e+08
Biography,Drama,Musical       4.350000e+08
Documentary,Drama,Sport       4.210750e+08
Action,Adventure,Fantasy      4.092138e+08
Adventure,Mystery,Sci-Fi      4.034000e+08
Adventure,Animation,Comedy    3.969922e+08
Name: total_revenue, dtype: float64
```

In [97]:
```
#top_ten = high_revenue_genre.head(10)
colors = ['red', 'green', 'blue', 'purple', 'orange', 'yellow']
plt.figure(figsize=(15, 6))
plt.bar(x = high_revenue_genre.index[:10], height = high_revenue_genre.values[

  # Add labels and title to the plot.
plt.xlabel('Genre')
plt.ylabel('Total revenue')
plt.xticks(rotation = 90, ha = 'right')
plt.title('Genre with Highest revenue')

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

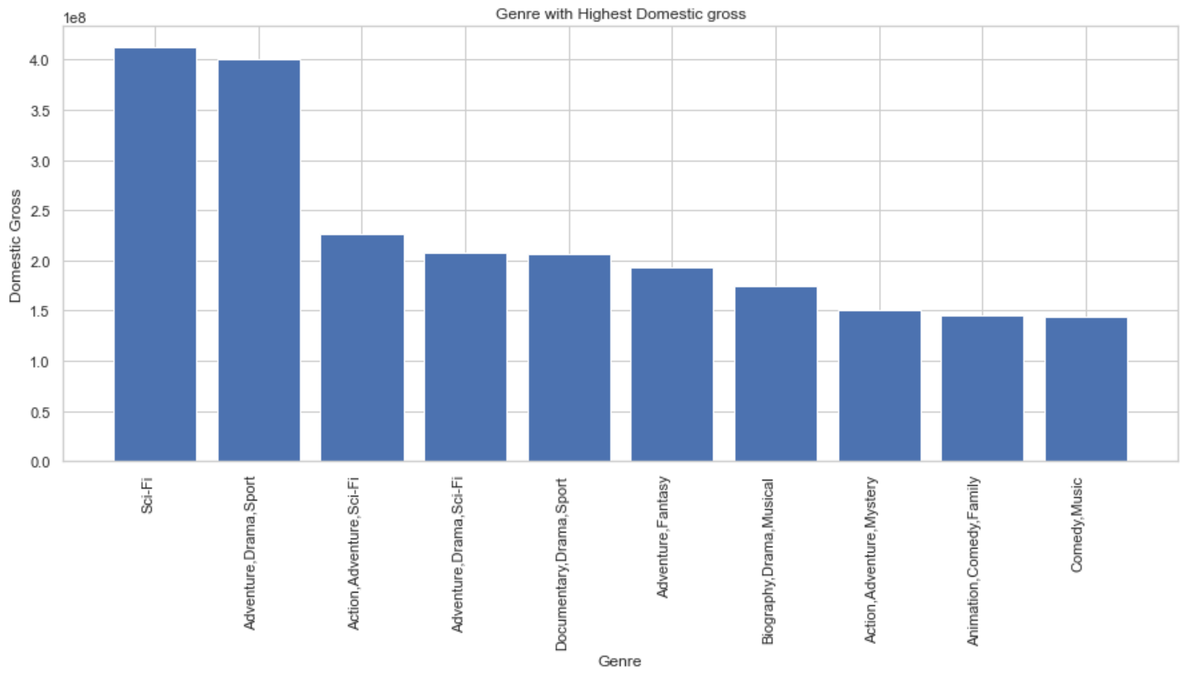


Genres with the highest doemstic gross

In [98]:
```
highest_domestic_genres = merged_df3.groupby('genres')['domestic_gross'].mean(
highest_domestic_genres.head(10)
```

Out[98]:
```
genres
Sci-Fi                         4.126000e+08
Adventure,Drama,Sport          4.007000e+08
Action,Adventure,Sci-Fi        2.258432e+08
Adventure,Drama,Sci-Fi         2.082000e+08
Documentary,Drama,Sport        2.067250e+08
Adventure,Fantasy              1.929000e+08
Biography,Drama,Musical        1.743000e+08
Action,Adventure,Mystery       1.509000e+08
Animation,Comedy,Family        1.458669e+08
Comedy,Music                   1.446000e+08
Name: domestic_gross, dtype: float64
```

In [99]:
```
plt.figure(figsize=(15, 6))
plt.bar(x = highest_domestic_genres.index[:10], height = highest_domestic_genr

  # Add labels and title to the plot.
plt.xlabel('Genre')
plt.ylabel('Domestic Gross')
plt.xticks(rotation = 90, ha = 'right')
plt.title('Genre with Highest Domestic gross')

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

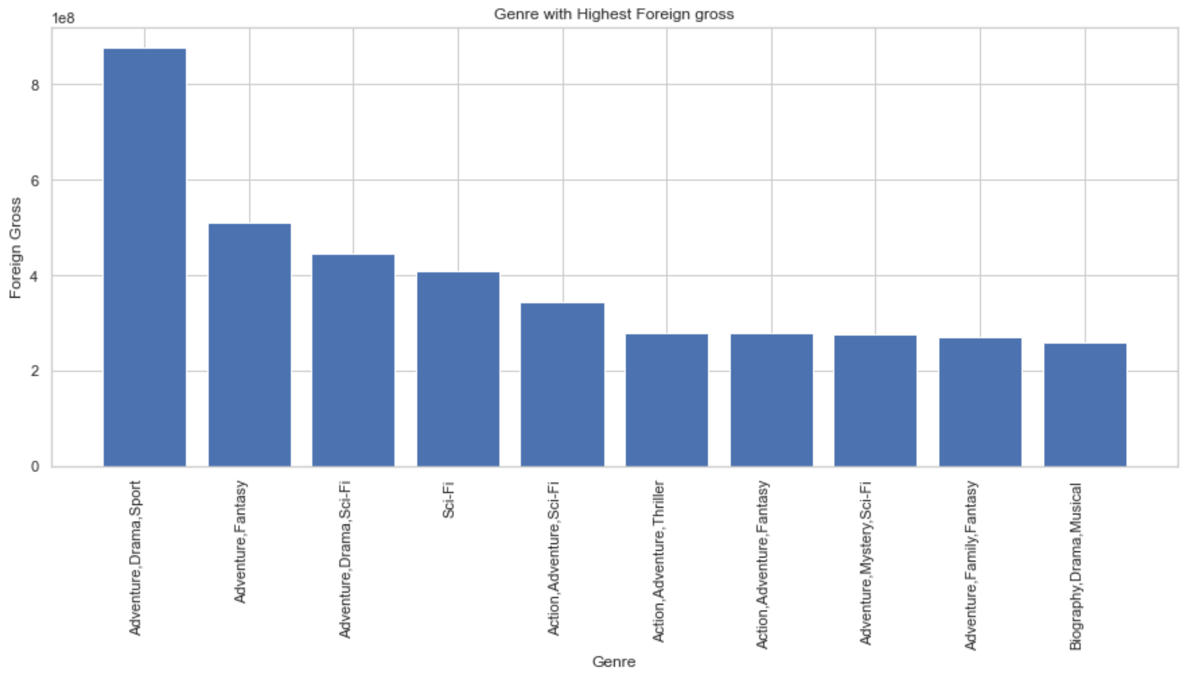


Genres with the highest foreign gross

In [100]:
```
highest_foreign_genres = merged_df3.groupby('genres')['foreign_gross'].mean().
highest_foreign_genres.head(10)
```

Out[100]:
```
genres
Adventure,Drama,Sport         8.757000e+08
Adventure,Fantasy             5.111333e+08
Adventure,Drama,Sci-Fi        4.455500e+08
Sci-Fi                        4.093000e+08
Action,Adventure,Sci-Fi       3.430432e+08
Action,Adventure,Thriller     2.804529e+08
Action,Adventure,Fantasy      2.796138e+08
Adventure,Mystery,Sci-Fi      2.769000e+08
Adventure,Family,Fantasy      2.695500e+08
Biography,Drama,Musical       2.607000e+08
Name: foreign_gross, dtype: float64
```

In [101]:
```
plt.figure(figsize=(15, 6))
plt.bar(x = highest_foreign_genres.index[:10], height = highest_foreign_genres

  # Add labels and title to the plot.
plt.xlabel('Genre')
plt.ylabel('Foreign Gross')
plt.xticks(rotation = 90, ha = 'right')
plt.title('Genre with Highest Foreign gross')

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



As you can see in both the domestic_gross and foreign_gross, **genres** with the highest revenue vary with the different audiences wacthing. This shows that the domestic and foreign audience like different **genre** differently

Does moving rating affect genre popularity

In [102]:
```
merged_df3.columns
```

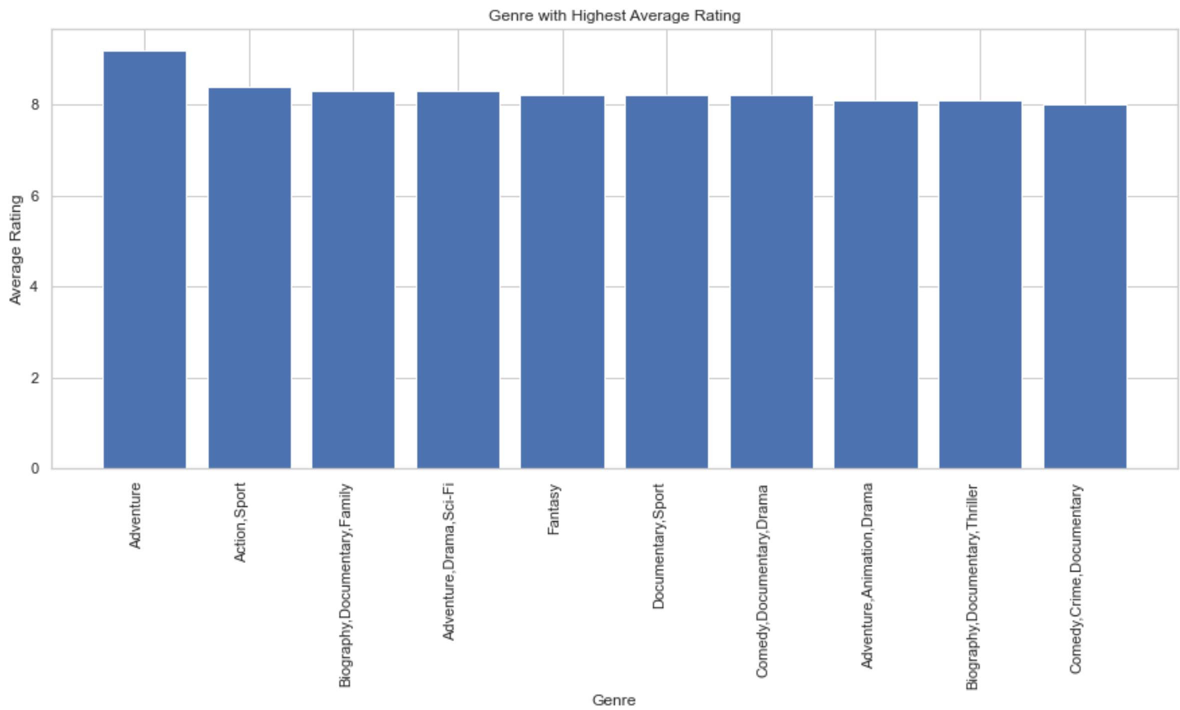
Out[102]:
```
Index(['tconst', 'averagerating', 'numvotes', 'primary_title',
       'original_title', 'start_year', 'runtime_minutes', 'genres', 'studio',
       'domestic_gross', 'foreign_gross', 'year', 'total_revenue'],
      dtype='object')
```

In [103]:
```
highest_rating_genres = merged_df3.groupby('genres')['averagerating'].mean().s
highest_rating_genres.head(10)
```

Out[103]:
```
genres
Adventure                           9.2
Action,Sport                        8.4
Biography,Documentary,Family        8.3
Adventure,Drama,Sci-Fi              8.3
Fantasy                             8.2
Documentary,Sport                   8.2
Comedy,Documentary,Drama            8.2
Adventure,Animation,Drama           8.1
Biography,Documentary,Thriller      8.1
Comedy,Crime,Documentary            8.0
Name: averagerating, dtype: float64
```

In [104]:
```
plt.subplots(figsize=(15, 6))
plt.bar(x = highest_rating_genres.index[:10], height = highest_rating_genres.v

  # Add labels and title to the plot.
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.xticks(rotation = 90, ha = 'right')
plt.title('Genre with Highest Average Rating')
```

Out[104]:
```
Text(0.5, 1.0, 'Genre with Highest Average Rating')
```



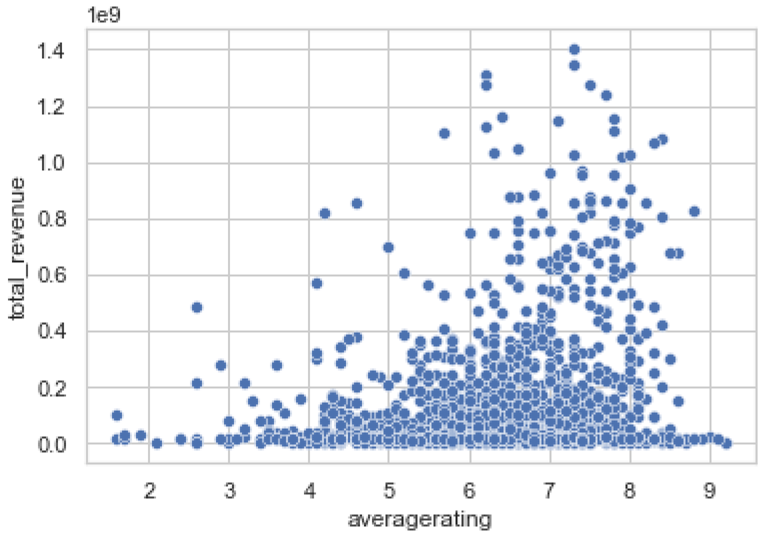
Does rating affect movie gross revenue

In [105]:
```
merged_df3.columns
```

Out[105]:
```
Index(['tconst', 'averagerating', 'numvotes', 'primary_title',
       'original_title', 'start_year', 'runtime_minutes', 'genres', 'studio',
       'domestic_gross', 'foreign_gross', 'year', 'total_revenue'],
      dtype='object')
```

In [106]:
```
sns.scatterplot(x = 'averagerating', y = 'total_revenue', data = merged_df3)
```

Out[106]:
```
<AxesSubplot:xlabel='averagerating', ylabel='total_revenue'>
```



Not necessarily, the higher the rating doesn't mean that the gross revenue of the movie will be higher.

Conclusions

This analysis leads to several recommendation on how Micrsosoft can be able to create movies and benefit well. This may be movies for the cincema or even their online digital streaming platform

- **The numvotes as the years go by:** As the years progress the number of votes keeps decreasing since 2010, this may be caused by differen factors such as audience perceiving new movies not as good as the old ones
- **The ten most popluar genres with movies are:**
  1. Drama
  2. Comdey, Drama, Romance
  3. Documentary
  4. Comdey, Drama
  5. Comdey
  6. Drama, Romanance
  7. Adventure, Animation, Comedy
  8. Comedy,Romance
  9. Action, Adventure, Sci-Fi
  10. Drama, Thriller
- **The genres with the highest total revenue globally are:** This is the total revenue of both the domestic gross and foreign gross. This factors in both audience and can be used for big box bluster movies
  1. Adventure,Drama,Sport
  2. Sci-Fi
  3. Adventure,Fantasy
  4. Adventure,Drama,Sci-Fi

5. Action,Adventure,Sci-Fi
6. Biography,Drama,Musical
7. Documentary,Drama,Sport
8. Action,Adventure,Fantasy
9. Adventure,Mystery,Sci-Fi
10. Adventure,Animation,Comedy

- **The genres with the highest domestic gross, this may vary different to total revenue and foreign gross of the genre due to preference of the domestic audience:**

  1. Sci-Fi
  2. Adventure,Drama,Sport
  3. Action,Adventure,Sci-Fi
  4. Adventure,Drama,Sci-Fi
  5. Documentary,Drama,Sport
  6. Adventure,Fantasy
  7. Biography,Drama,Musical
  8. Action,Adventure,Mystery
  9. Animation,Comedy,Family
  10. Comedy,Music

- **The genres with the highest foreign gross, this may vary different to domestic revenue of the genre due to preference of the foreign audience to the domestic audience:**

  1. Adventure,Drama,Sport
  2. Adventure,Fantasy
  3. Adventure,Drama,Sci-Fi
  4. Sci-Fi
  5. Action,Adventure,Sci-Fi
  6. Action,Adventure,Thriller
  7. Action,Adventure,Fantasy
  8. Adventure,Mystery,Sci-Fi
  9. Adventure,Family,Fantasy
  10. Biography,Drama,Musical

- **The studios with the best total revenue are:** This are the best performing stuidos when it comes to making movies that will bring high revenue for them in both the domestic and foreign market.

  1. P/DW
  2. BV
  3. GrtIndia
  4. Fox
  5. WB
  6. WB (NL)
  7. Sony
  8. Par.
  9. Uni.
  10. MGM

- **Studios with the best revenue, domestic and foreign market respetively:** This are the best perfomring studios in their respective markets. The studio may vary with the different audiences this been domestic and foreign.
- **The ratings affect on the movie revenue:** From this it is Not necessarily, the higher the rating doesn't mean that the gross revenue of the movie will be higher.

Recommendations

Following the analysis, these are some of the recommendations:

1. **For the domestic audience:** Microsoft should focus more on producing Sci-Fi since the it bring the most revenue in the domestic audience followed by Adventure,Drama,Sport
2. **For foreign audience:** Microsoft should focus more on producing Adventure,Drama,Sport followed by Adventure,Fantasy to yield more revenue for them in the long run.
3. **For the market at large:** In terms of both markets, this will be useful in creating big blockbuster movies that both audience enjoy. They should focus more on Adventure,Drama,Sport followed by Sci-Fi movies.
4. **Most popular movie genres for their online streaming services:** This will help Microsoft studios in breaking into the online streaming markets. They should focus more on providing movies based on drama followe by comedy, drama, romance genre. This would help them get more subscribers.
5. **Collabration with other studios for the market at large**: When collaborating with other studios for the market at large for things such as crossovers, the recommended studio is P/DW followed by BV. This will not only bring more revenue for them but also exposure and revenue for the other studion
6. **Collabration with other studio for the domestic and foreign audience respectively:** Studios with the best revenue, domestic and foreign. These are the best-performing studios in their respective markets. The studio may vary with the different audiences, these been domestic and foreign. Microsoft should focus on which studion they collaborate with when they make movies for the domestic audience and focus on which stuidos perfom best when they are making a movie for the foreign market.

```
In [ ]:
```