Microsoft Movie Analysis

Author: Marissa Bush

Overview

This EDA gives insight on what successful movie studios are doing well and what specific actions Microsoft can do to achieve similar aims.

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.

Question 1: How many films have the top studios made from 2010-2019, and which studio brings in the most earnings? In other words, what are the studios that will be Microsoft's biggest competition?

Question 2: Is there a positive correlation between film length and domestic gross?

Question 3: What are the most popular movie genres?

Data Understanding

Three sets of data were collected to answer these questions - box office mojo movie gross data, imdb title basics data, and imdb title ratings data.

In [1]:

```
# Import standard packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/Users/marissabush/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/statsmodels/t
ools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions
in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [2]:

```
# Load csv files
bom_mg_df = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')
imdb_tr_df = pd.read_csv('data/zippedData/imdb.title.ratings.csv.gz')
imdb_tb_df = pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
```

BOM Movie Gross Data

In [3]:

```
# Function to get data frame info
```

```
def df_scope(m_df):
    #print name,.shape, .info, .describe
    for name, df in m_df.items():
        print('=' * 100)
        print(name)
        print(m_df.shape, '\n')
        print(m_df.info(), '\n')
        print(m_df.describe(include='all'))
```

In [4]:

```
df_scope(bom_mg_df)
df_scope(imdb_tr_df)
df_scope(imdb_tb_df)
```

```
==========================================================================================
==========
title
(3387, 5)

<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
None

            title studio  domestic_gross foreign_gross         year
count        3387   3382    3.359000e+03          2037  3387.000000
unique       3386    257             NaN          1204          NaN
top     Bluebeard    IFC             NaN       1200000          NaN
freq            2    166             NaN            23          NaN
mean          NaN    NaN    2.874585e+07           NaN  2013.958075
std           NaN    NaN    6.698250e+07           NaN     2.478141
min           NaN    NaN    1.000000e+02           NaN  2010.000000
25%           NaN    NaN    1.200000e+05           NaN  2012.000000
50%           NaN    NaN    1.400000e+06           NaN  2014.000000
75%           NaN    NaN    2.790000e+07           NaN  2016.000000
max           NaN    NaN    9.367000e+08           NaN  2018.000000
==========================================================================================
==========
studio
(3387, 5)

<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
None

            title studio  domestic_gross foreign_gross         year
count        3387   3382    3.359000e+03          2037  3387.000000
unique       3386    257             NaN          1204          NaN
top     Bluebeard    IFC             NaN       1200000          NaN
freq            2    166             NaN            23          NaN
mean          NaN    NaN    2.874585e+07           NaN  2013.958075
std           NaN    NaN    6.698250e+07           NaN     2.478141
```

```
min           NaN    NaN    1.000000e+02          NaN  2010.000000
25%           NaN    NaN    1.200000e+05          NaN  2012.000000
50%           NaN    NaN    1.400000e+06          NaN  2014.000000
75%           NaN    NaN    2.790000e+07          NaN  2016.000000
max           NaN    NaN    9.367000e+08          NaN  2018.000000
==========================================================================================
===========
domestic_gross
(3387, 5)

<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
None

             title studio  domestic_gross foreign_gross         year
count         3387   3382    3.359000e+03          2037  3387.000000
unique        3386    257             NaN          1204          NaN
top      Bluebeard    IFC             NaN       1200000          NaN
freq             2    166             NaN            23          NaN
mean           NaN    NaN    2.874585e+07           NaN  2013.958075
std            NaN    NaN    6.698250e+07           NaN     2.478141
min            NaN    NaN    1.000000e+02           NaN  2010.000000
25%            NaN    NaN    1.200000e+05           NaN  2012.000000
50%            NaN    NaN    1.400000e+06           NaN  2014.000000
75%            NaN    NaN    2.790000e+07           NaN  2016.000000
max            NaN    NaN    9.367000e+08           NaN  2018.000000
==========================================================================================
===========
foreign_gross
(3387, 5)

<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
None

             title studio  domestic_gross foreign_gross         year
count         3387   3382    3.359000e+03          2037  3387.000000
unique        3386    257             NaN          1204          NaN
top      Bluebeard    IFC             NaN       1200000          NaN
freq             2    166             NaN            23          NaN
mean           NaN    NaN    2.874585e+07           NaN  2013.958075
std            NaN    NaN    6.698250e+07           NaN     2.478141
min            NaN    NaN    1.000000e+02           NaN  2010.000000
25%            NaN    NaN    1.200000e+05           NaN  2012.000000
50%            NaN    NaN    1.400000e+06           NaN  2014.000000
75%            NaN    NaN    2.790000e+07           NaN  2016.000000
max            NaN    NaN    9.367000e+08           NaN  2018.000000
==========================================================================================
===========
year
(3387, 5)
```

```
<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
None

            title studio  domestic_gross foreign_gross         year
count        3387   3382    3.359000e+03          2037  3387.000000
unique       3386    257             NaN          1204          NaN
top     Bluebeard    IFC             NaN       1200000          NaN
freq            2    166             NaN            23          NaN
mean          NaN    NaN    2.874585e+07           NaN  2013.958075
std           NaN    NaN    6.698250e+07           NaN     2.478141
min           NaN    NaN    1.000000e+02           NaN  2010.000000
25%           NaN    NaN    1.200000e+05           NaN  2012.000000
50%           NaN    NaN    1.400000e+06           NaN  2014.000000
75%           NaN    NaN    2.790000e+07           NaN  2016.000000
max           NaN    NaN    9.367000e+08           NaN  2018.000000
==========================================================================================
===========
tconst
(73856, 3)

<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
None

           tconst  averagerating      numvotes
count       73856   73856.000000  7.385600e+04
unique      73856            NaN           NaN
top     tt2055720            NaN           NaN
freq            1            NaN           NaN
mean          NaN       6.332729  3.523662e+03
std           NaN       1.474978  3.029402e+04
min           NaN       1.000000  5.000000e+00
25%           NaN       5.500000  1.400000e+01
50%           NaN       6.500000  4.900000e+01
75%           NaN       7.400000  2.820000e+02
max           NaN      10.000000  1.841066e+06
==========================================================================================
===========
averagerating
(73856, 3)

<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
None
```

```
           tconst  averagerating      numvotes
count       73856   73856.000000  7.385600e+04
unique      73856            NaN           NaN
top     tt2055720            NaN           NaN
freq            1            NaN           NaN
mean          NaN       6.332729  3.523662e+03
std           NaN       1.474978  3.029402e+04
min           NaN       1.000000  5.000000e+00
25%           NaN       5.500000  1.400000e+01
50%           NaN       6.500000  4.900000e+01
75%           NaN       7.400000  2.820000e+02
max           NaN      10.000000  1.841066e+06
==========================================================================================
===========
numvotes
(73856, 3)

<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
None

           tconst  averagerating      numvotes
count       73856   73856.000000  7.385600e+04
unique      73856            NaN           NaN
top     tt2055720            NaN           NaN
freq            1            NaN           NaN
mean          NaN       6.332729  3.523662e+03
std           NaN       1.474978  3.029402e+04
min           NaN       1.000000  5.000000e+00
25%           NaN       5.500000  1.400000e+01
50%           NaN       6.500000  4.900000e+01
75%           NaN       7.400000  2.820000e+02
max           NaN      10.000000  1.841066e+06
==========================================================================================
===========
tconst
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
unique     146144        136071         137773            NaN
top     tt5222638          Home         Broken            NaN
freq            1            24             19            NaN
mean          NaN           NaN            NaN    2014.621798
std           NaN           NaN            NaN       2.733583
min           NaN           NaN            NaN    2010.000000
25%           NaN           NaN            NaN    2012.000000
50%           NaN           NaN            NaN    2015.000000
75%           NaN           NaN            NaN    2017.000000
```

```
max           NaN           NaN            NaN    2115.000000

        runtime_minutes       genres
count     114405.000000       140736
unique              NaN         1085
top                 NaN  Documentary
freq                NaN        32185
mean          86.187247          NaN
std          166.360590          NaN
min            1.000000          NaN
25%           70.000000          NaN
50%           87.000000          NaN
75%           99.000000          NaN
max        51420.000000          NaN
============================================================================================
primary_title
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
unique     146144        136071         137773            NaN
top     tt5222638          Home         Broken            NaN
freq            1            24             19            NaN
mean          NaN           NaN            NaN    2014.621798
std           NaN           NaN            NaN       2.733583
min           NaN           NaN            NaN    2010.000000
25%           NaN           NaN            NaN    2012.000000
50%           NaN           NaN            NaN    2015.000000
75%           NaN           NaN            NaN    2017.000000
max           NaN           NaN            NaN    2115.000000

        runtime_minutes       genres
count     114405.000000       140736
unique              NaN         1085
top                 NaN  Documentary
freq                NaN        32185
mean          86.187247          NaN
std          166.360590          NaN
min            1.000000          NaN
25%           70.000000          NaN
50%           87.000000          NaN
75%           99.000000          NaN
max        51420.000000          NaN
============================================================================================
original_title
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
unique     146144        136071         137773            NaN
top     tt5222638          Home         Broken            NaN
freq            1            24             19            NaN
mean          NaN           NaN            NaN    2014.621798
std           NaN           NaN            NaN       2.733583
min           NaN           NaN            NaN    2010.000000
25%           NaN           NaN            NaN    2012.000000
50%           NaN           NaN            NaN    2015.000000
75%           NaN           NaN            NaN    2017.000000
max           NaN           NaN            NaN    2115.000000

        runtime_minutes       genres
count     114405.000000       140736
unique              NaN         1085
top                 NaN  Documentary
freq                NaN        32185
mean          86.187247          NaN
std          166.360590          NaN
min            1.000000          NaN
25%           70.000000          NaN
50%           87.000000          NaN
75%           99.000000          NaN
max        51420.000000          NaN
==========================================================================================
===========
start_year
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
unique     146144        136071         137773            NaN
top     tt5222638          Home         Broken            NaN
freq            1            24             19            NaN
mean          NaN           NaN            NaN    2014.621798
std           NaN           NaN            NaN       2.733583
min           NaN           NaN            NaN    2010.000000
25%           NaN           NaN            NaN    2012.000000
50%           NaN           NaN            NaN    2015.000000
75%           NaN           NaN            NaN    2017.000000
max           NaN           NaN            NaN    2115.000000

        runtime_minutes       genres
count     114405.000000       140736
unique              NaN         1085
top                 NaN  Documentary
freq                NaN        32185
mean          86.187247          NaN
std          166.360590          NaN
```

```
min           1.000000         NaN
25%          70.000000         NaN
50%          87.000000         NaN
75%          99.000000         NaN
max       51420.000000         NaN
==========================================================================================
===========
runtime_minutes
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
unique     146144        136071         137773            NaN
top     tt5222638          Home         Broken            NaN
freq            1            24             19            NaN
mean          NaN           NaN            NaN    2014.621798
std           NaN           NaN            NaN       2.733583
min           NaN           NaN            NaN    2010.000000
25%           NaN           NaN            NaN    2012.000000
50%           NaN           NaN            NaN    2015.000000
75%           NaN           NaN            NaN    2017.000000
max           NaN           NaN            NaN    2115.000000

        runtime_minutes       genres
count     114405.000000       140736
unique              NaN         1085
top                 NaN  Documentary
freq                NaN        32185
mean          86.187247          NaN
std          166.360590          NaN
min            1.000000          NaN
25%           70.000000          NaN
50%           87.000000          NaN
75%           99.000000          NaN
max        51420.000000          NaN
==========================================================================================
===========
genres
(146144, 6)

<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
None

           tconst primary_title original_title     start_year  \
count      146144        146144         146123  146144.000000
```

```
unique      146144         136071         137773            NaN
top      tt5222638           Home         Broken            NaN
freq             1             24             19            NaN
mean           NaN            NaN            NaN    2014.621798
std            NaN            NaN            NaN       2.733583
min            NaN            NaN            NaN    2010.000000
25%            NaN            NaN            NaN    2012.000000
50%            NaN            NaN            NaN    2015.000000
75%            NaN            NaN            NaN    2017.000000
max            NaN            NaN            NaN    2115.000000

        runtime_minutes        genres
count     114405.000000        140736
unique              NaN          1085
top                 NaN   Documentary
freq                NaN         32185
mean          86.187247           NaN
std          166.360590           NaN
min            1.000000           NaN
25%           70.000000           NaN
50%           87.000000           NaN
75%           99.000000           NaN
max        51420.000000           NaN
```

In [5]:

```
# View bom_mg_df
bom_mg_df.head(2)
```

Out[5]:

	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

In [6]:

```
imdb_tr_df.head(2)
```

Out[6]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559

In [7]:

```
imdb_tb_df.head(2)
```

Out[7]:

	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

In [8]:

```
# Combine both IMDB data frames on common column
imdb_df = pd.merge(imdb_tr_df, imdb_tb_df, on='tconst', how='inner')
imdb_df.shape
```

Out[8]:

```
(73856, 8)
```

In [9]:

```
# View data frame
imdb_df.head(2)
```

Out[9]:

	tconst	averagerating	numvotes	primary_title	original_title	start_year	runtime_minutes	genres
0	tt10356526	8.3	31	Laiye Je Yaarian	Laiye Je Yaarian	2019	117.0	Romance
1	tt10384606	8.9	559	Borderless	Borderless	2019	87.0	Documentary

Combined Dateframe

In [10]:

```
# View bom_mg_df column names
bom_mg_df.columns
```

Out[10]:

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

In [11]:

```
# Rename both primary title and start year to match with bom_mg_df
imdb_df.rename(columns = {'primary_title':'title'}, inplace = True)
imdb_df.rename(columns = {'start_year':'year'}, inplace = True)
```

In [12]:

```
# Double check column names
imdb_df.tail(2)
```

Out[12]:

	tconst	averagerating	numvotes	title	original_title	year	runtime_minutes	genres
73854	tt9886934	7.0	5	The Projectionist	The Projectionist	2019	81.0	Documentary
73855	tt9894098	6.3	128	Sathru	Sathru	2019	129.0	Thriller

In [13]:

```
# Merge both data frames on two common columns, 'title' and 'year'
df = imdb_df.merge(bom_mg_df, on = ['title','year'], how = 'inner')
df.head(2)
```

Out[13]:

	tconst	averagerating	numvotes	title	original_title	year	runtime_minutes	genres	studio	dome
0	tt1043726	4.2	50352	The Legend of Hercules	The Legend of Hercules	2014	99.0	Action,Adventure,Fantasy	LG/S	
1	tt1171222	5.1	8296	Baggage Claim	Baggage Claim	2013	96.0	Comedy	FoxS	

In [14]:

```
# View combined data frame shape
df.shape
```

Out[14]:

```
(1847, 11)
```

In [15]:

In [15]:

```
# View columns in new data frame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1847 entries, 0 to 1846
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   tconst           1847 non-null   object
 1   averagerating    1847 non-null   float64
 2   numvotes         1847 non-null   int64
 3   title            1847 non-null   object
 4   original_title   1847 non-null   object
 5   year             1847 non-null   int64
 6   runtime_minutes  1843 non-null   float64
 7   genres           1845 non-null   object
 8   studio           1845 non-null   object
 9   domestic_gross   1837 non-null   float64
 10  foreign_gross    1269 non-null   object
dtypes: float64(3), int64(2), object(6)
memory usage: 173.2+ KB
```

Data Preparation

To begin the data cleaning process I chose to examine and drop any duplicates in the two columns, 'tconst' and 'original_title'. Then find all missing values, check the percentages and drop those, as well.

In [16]:

```
# Check for duplicates and missing values for combined df
```

In [17]:

```
# View data frame
df.head(2)
```

Out[17]:

	tconst	averagerating	numvotes	title	original_title	year	runtime_minutes	genres	studio	dome
0	tt1043726	4.2	50352	The Legend of Hercules	The Legend of Hercules	2014	99.0	Action,Adventure,Fantasy	LG/S	
1	tt1171222	5.1	8296	Baggage Claim	Baggage Claim	2013	96.0	Comedy	FoxS	

In [18]:

```
# View tconst column for duplicates
df['tconst'].duplicated().sum()
```

Out[18]:

0

In [19]:

```
# View original title column for duplicates
df['original_title'].duplicated().sum()
```

Out[19]:

20

In [20]:

```
# Drop duplicates
```

```
# Drop duplicates
df.drop_duplicates(subset = ['original_title'], inplace = True)
df.shape
```

Out[20]:

```
(1827, 11)
```

In [21]:

```
# View any missing values
df.isnull().sum().sort_values(ascending=False)
```

Out[21]:

```
foreign_gross      572
domestic_gross      10
studio               2
genres               1
runtime_minutes      0
year                 0
original_title       0
title                0
numvotes             0
averagerating        0
tconst               0
dtype: int64
```

In [22]:

```
# Divide by length of df to view percentage missing
len(df)
df.isnull().sum().sort_values(ascending = False)/len(df)
```

Out[22]:

```
foreign_gross      0.313082
domestic_gross     0.005473
studio             0.001095
genres             0.000547
runtime_minutes    0.000000
year               0.000000
original_title     0.000000
title              0.000000
numvotes           0.000000
averagerating      0.000000
tconst             0.000000
dtype: float64
```

In [23]:

```
# Drop foreign gross column (30% missing) and columns I don't need
# for analysis
df.drop('foreign_gross', axis = 1, inplace = True)
df.drop('tconst', axis = 1, inplace = True)
df.drop('year', axis = 1, inplace = True)
df.drop('original_title', axis = 1, inplace = True)
```

In [24]:

```
# Drop missing values from the other columns
df.dropna(subset=['genres', 'runtime_minutes', 'domestic_gross', 'studio'], inplace=True
)
df.shape
```

Out[24]:

```
(1816, 7)
```

In [25]:

```
# Double check for missing values
df.isnull().sum().sort_values(ascending=False)
```

Out[25]:

```
domestic_gross     0
studio             0
genres             0
runtime_minutes    0
title              0
numvotes           0
averagerating      0
dtype: int64
```

In [26]:

```
# rename column for more understanding
df.rename(columns = {'start_year':'release_date'}, inplace = True)
```

In [27]:

```
# view df
df.head(2)
```

Out[27]:

	averagerating	numvotes	title	runtime_minutes	genres	studio	domestic_gross
0	4.2	50352	The Legend of Hercules	99.0	Action,Adventure,Fantasy	LG/S	18800000.0
1	5.1	8296	Baggage Claim	96.0	Comedy	FoxS	21600000.0

In [28]:

```
# View descriptive stats for any significant outliers
df.describe()
```

Out[28]:

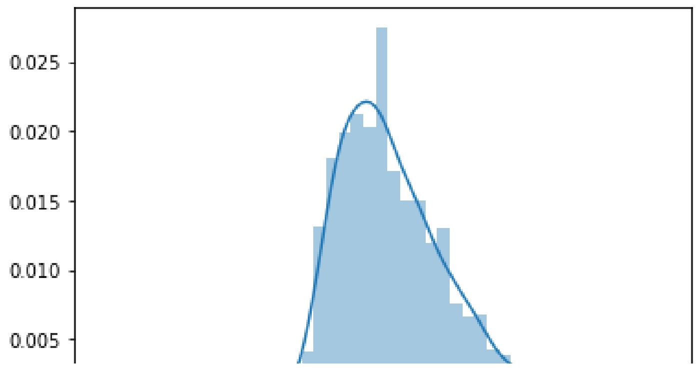
	averagerating	numvotes	runtime_minutes	domestic_gross
count	1816.000000	1.816000e+03	1816.000000	1.816000e+03
mean	6.423073	9.294794e+04	110.953194	4.299649e+07
std	0.998854	1.510763e+05	19.794412	7.751079e+07
min	1.600000	6.000000e+00	25.000000	3.000000e+02
25%	5.800000	8.013750e+03	97.000000	5.835000e+05
50%	6.500000	3.638950e+04	108.000000	1.080000e+07
75%	7.100000	1.071020e+05	123.000000	5.242500e+07
max	8.800000	1.841066e+06	189.000000	7.001000e+08

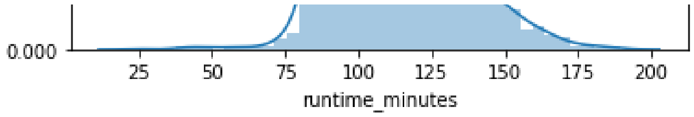
In [29]:

```
# Check for outliers
sns.distplot(df['runtime_minutes'])
```

Out[29]:

<AxesSubplot:xlabel='runtime_minutes'>





In [30]:

```
# Remove outliers
df = df[df.runtime_minutes != 272]
df = df[df.runtime_minutes != 25]
```

Data Modeling

Question 1: How many films have the top studios made from 2010-2019, and which studio brings in the most earnings? In other words, what are the studios that will be Microsoft's biggest competition?

In [31]:

```
# View df
df.head(2)
```

Out[31]:

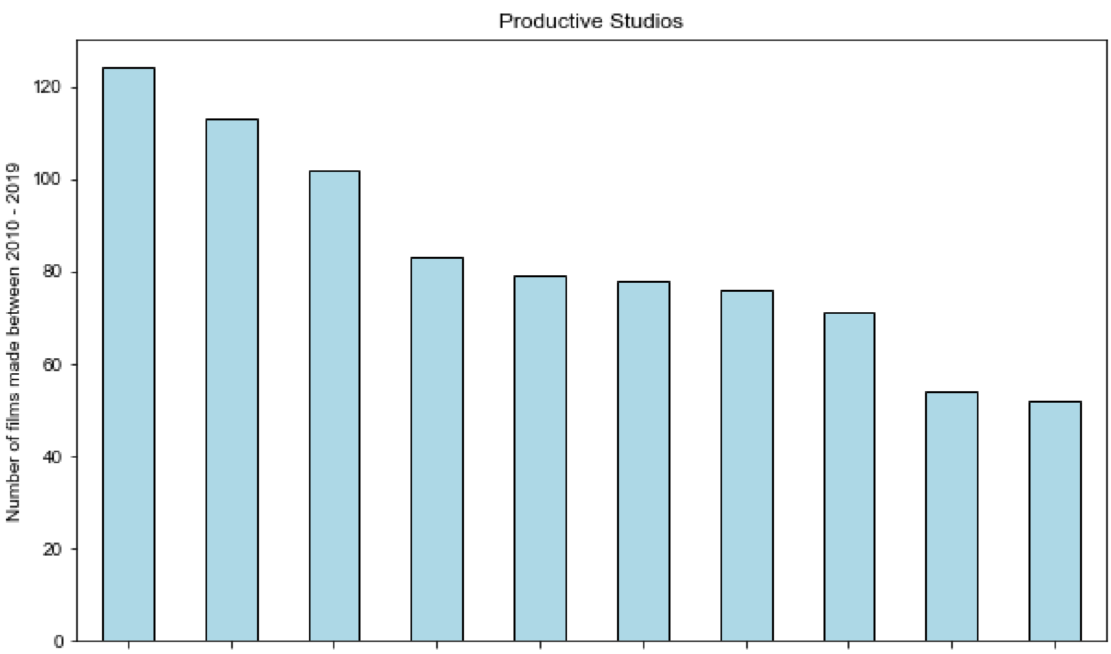
	averagerating	numvotes	title	runtime_minutes	genres	studio	domestic_gross
0	4.2	50352	The Legend of Hercules	99.0	Action,Adventure,Fantasy	LG/S	18800000.0
1	5.1	8296	Baggage Claim	96.0	Comedy	FoxS	21600000.0

In [32]:

```
# How many films per studio
top_studios = df['studio'].value_counts().head(10)
top_studios.plot(kind = 'bar', figsize = (10,6),
                 color = 'lightblue', edgecolor = 'black'
)

plt.xlabel('Top 10 Studios')
plt.ylabel('Number of films made between 2010 - 2019')
plt.xticks(rotation = 50)
plt.title('Productive Studios')

sns.set()
;
```

Out[32]:

''



In [33]:

```
# Descriptive stats for top_studios
top_studios.describe()
```

Out[33]:

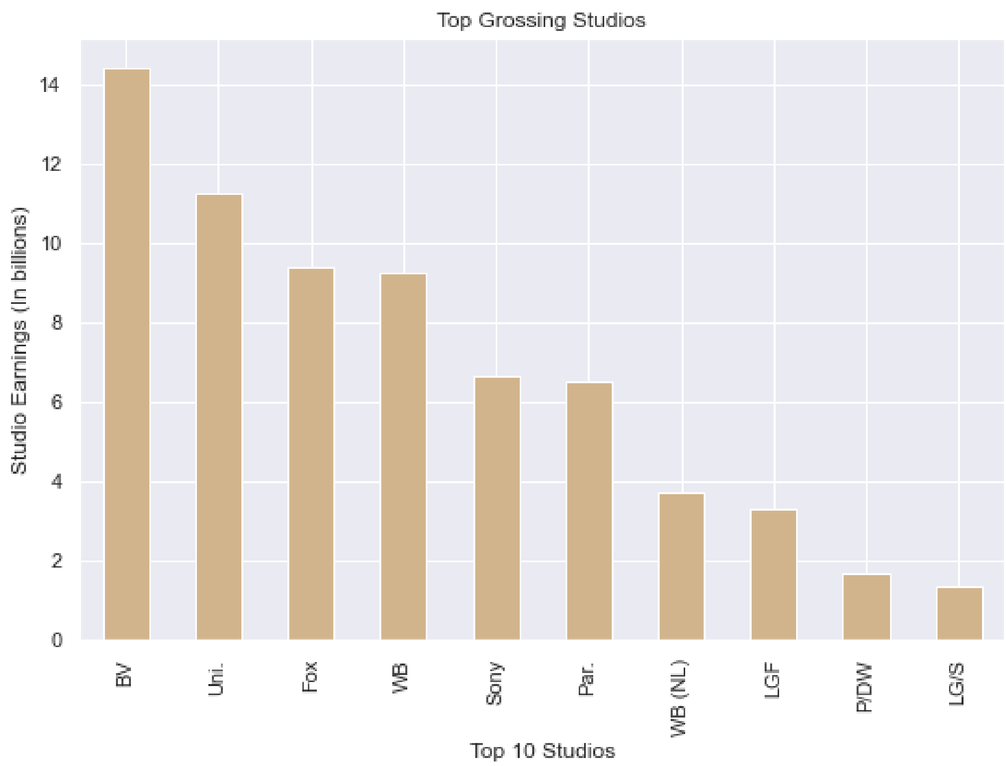
```
count     10.000000
mean      83.200000
std       23.517369
min       52.000000
25%       72.250000
50%       78.500000
75%       97.250000
max      124.000000
Name: studio, dtype: float64
```

In [34]:

```
# Group top studios and their domestic gross sum, divide by a billion
studio_gross = df.groupby('studio').domestic_gross.sum().sort_values(ascending = False).
head(10)
studio_gross = studio_gross/1000000000
```

In [35]:

```
# Bar graph of studios and domestic gross sum
studio_gross.plot.bar(figsize=(9,6), color = 'tan',
                      edgecolor = 'white'
                     )
plt.xlabel('Top 10 Studios')
plt.ylabel('Studio Earnings (In billions)')
plt.title('Top Grossing Studios')
sns.set();
```



The most productive studios from 2010-2019 made over 100 films in that time with Buena Vista studios bringing in the highest earnings.

Question 2: Is there a correlation between film length and domestic gross?

In [36]:

In [36]:

```
# View domestic gross column stats
df['domestic_gross'].describe()
```

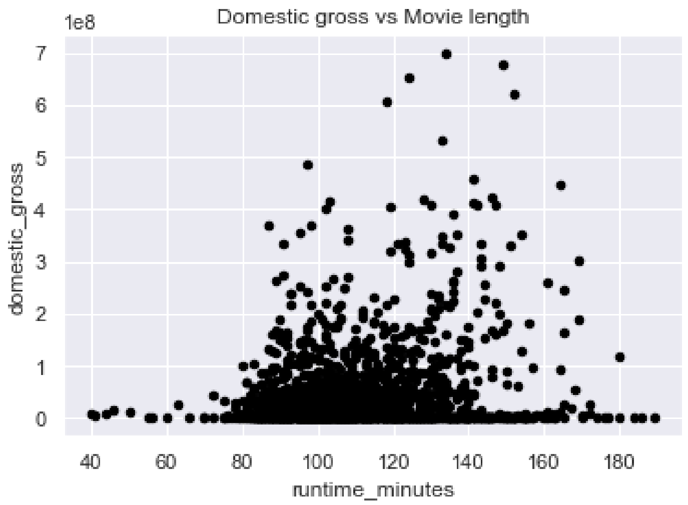
Out[36]:

```
count    1.815000e+03
mean     4.299158e+07
std      7.753186e+07
min      3.000000e+02
25%      5.830000e+05
50%      1.080000e+07
75%      5.245000e+07
max      7.001000e+08
Name: domestic_gross, dtype: float64
```

In [37]:

```
# Scatter plot of domestic gross vs movie length
plt.figure(figsize = (9, 6))
df.plot.scatter(x = 'runtime_minutes', y = 'domestic_gross', color = 'black')
plt.title("Domestic gross vs Movie length")

plt.show()

sns.set();
```

```
<Figure size 648x432 with 0 Axes>
```



It looks there is a slight positive correlation between the two with the higher grossing films being in the 2 - 2.5 hour range. Perhaps making a film that length would be a good move.

Question 3: What are the most popular movie genres?

In [38]:

```
# View df
df.head(1)
```

Out[38]:

	averagerating	numvotes	title	runtime_minutes	genres	studio	domestic_gross
0	4.2	50352	The Legend of Hercules	99.0	Action,Adventure,Fantasy	LG/S	18800000.0

In [39]:

```
# Split genres column by column
df['genres'] = df['genres'].str.split(',')
```

In [40]:

```
# Expand df by adding different genres in separate rows
```

```
# Expand df by adding different genres in separate rows
df_explode = df.explode('genres')
```

In [41]:

```
# View df
df_explode.head()
```

Out[41]:

	averagerating	numvotes	title	runtime_minutes	genres	studio	domestic_gross
0	4.2	50352	The Legend of Hercules	99.0	Action	LG/S	18800000.0
0	4.2	50352	The Legend of Hercules	99.0	Adventure	LG/S	18800000.0
0	4.2	50352	The Legend of Hercules	99.0	Fantasy	LG/S	18800000.0
1	5.1	8296	Baggage Claim	96.0	Comedy	FoxS	21600000.0
2	7.6	326657	Moneyball	133.0	Biography	Sony	75600000.0

In [42]:

```
# View new df 'genres' column by value counts
df_explode['genres'].value_counts()
```

Out[42]:

```
Drama          945
Comedy         651
Action         519
Adventure      361
Romance        298
Thriller       282
Crime          264
Biography      185
Horror         141
Mystery        127
Fantasy        123
Animation      113
Sci-Fi         108
Documentary     95
Family          81
History         71
Music           56
Sport           35
War             22
Musical         12
Western         11
News             1
Name: genres, dtype: int64
```

In [43]:

```
# Group the data by genres and view sum stats
df_explode.groupby('genres').sum()
```

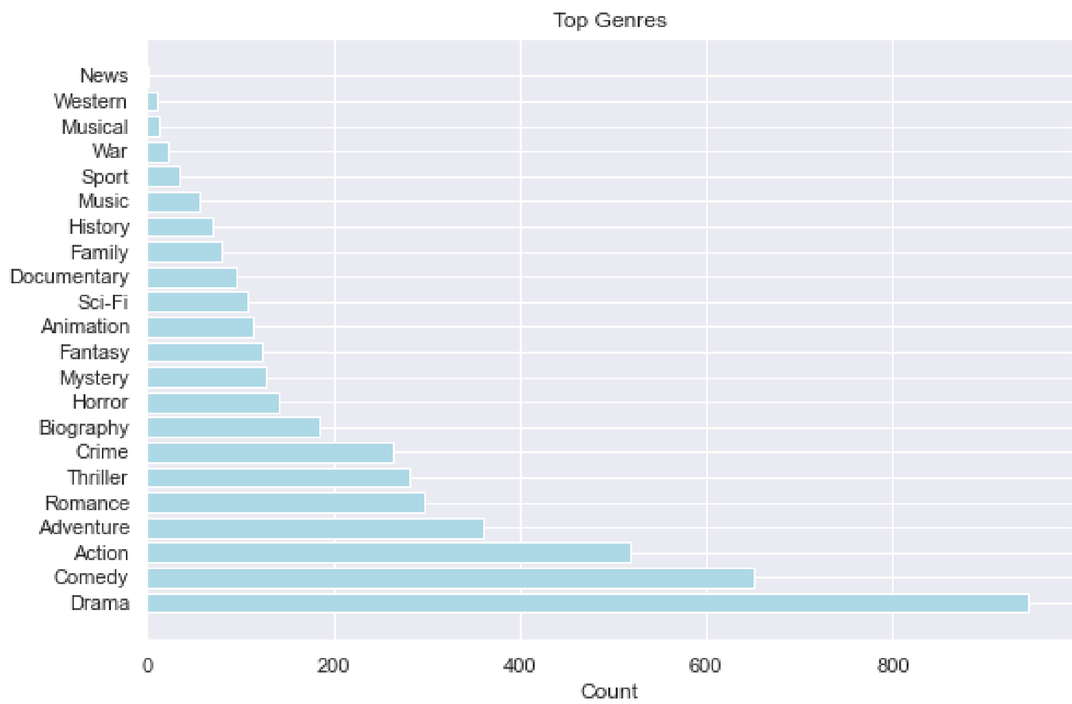
Out[43]:

genres	averagerating	numvotes	runtime_minutes	domestic_gross
Action	3282.6	75396162	60952.0	3.627845e+10
Adventure	2333.4	66684656	40314.0	4.055522e+10
Animation	742.2	12358231	10734.0	1.315974e+10
Biography	1298.3	16012128	21547.0	5.209095e+09
Comedy	4019.2	49300577	69862.0	2.965430e+10
Crime	1703.5	25611913	30020.0	8.177104e+09
Documentary	[illegible]	[illegible]	[illegible]	[illegible]

genres	averagerating	numvotes	runtime_minutes	domestic_gross
Documentary	694.2	900089	9012.0	4.210960e+08
Drama	6227.8	71150037	108416.0	2.209239e+10
Family	490.0	5468132	8600.0	4.914445e+09
Fantasy	768.8	16568762	13763.0	8.269386e+09
History	488.0	4898372	8537.0	1.894073e+09
Horror	807.8	10511836	13958.0	4.907666e+09
Music	372.8	3476117	6213.0	1.466830e+09
Musical	72.9	482775	1577.0	3.839827e+08
Mystery	797.6	14373218	13609.0	4.332947e+09
News	6.7	1167	75.0	1.320000e+04
Romance	1864.3	15923321	33351.0	5.875371e+09
Sci-Fi	708.3	32014751	12637.0	1.433445e+10
Sport	246.2	2518741	4236.0	9.992940e+08
Thriller	1767.5	32018330	31040.0	1.140090e+10
War	144.1	630269	2581.0	2.184931e+08
Western	74.5	2026472	1283.0	5.061511e+08

In [44]:

```
# Make a horizontal bar chart with the df_explode['genres'] value counts
fig, ax = plt.subplots(figsize = (9,6))

genre_types = df_explode['genres'].value_counts()

ax.barh(y=genre_types.index,
    width=genre_types.values, color = 'lightblue', edgecolor = 'white'
)
ax.set_xlabel('Count')
ax.set_title('Top Genres')
sns.set();
```

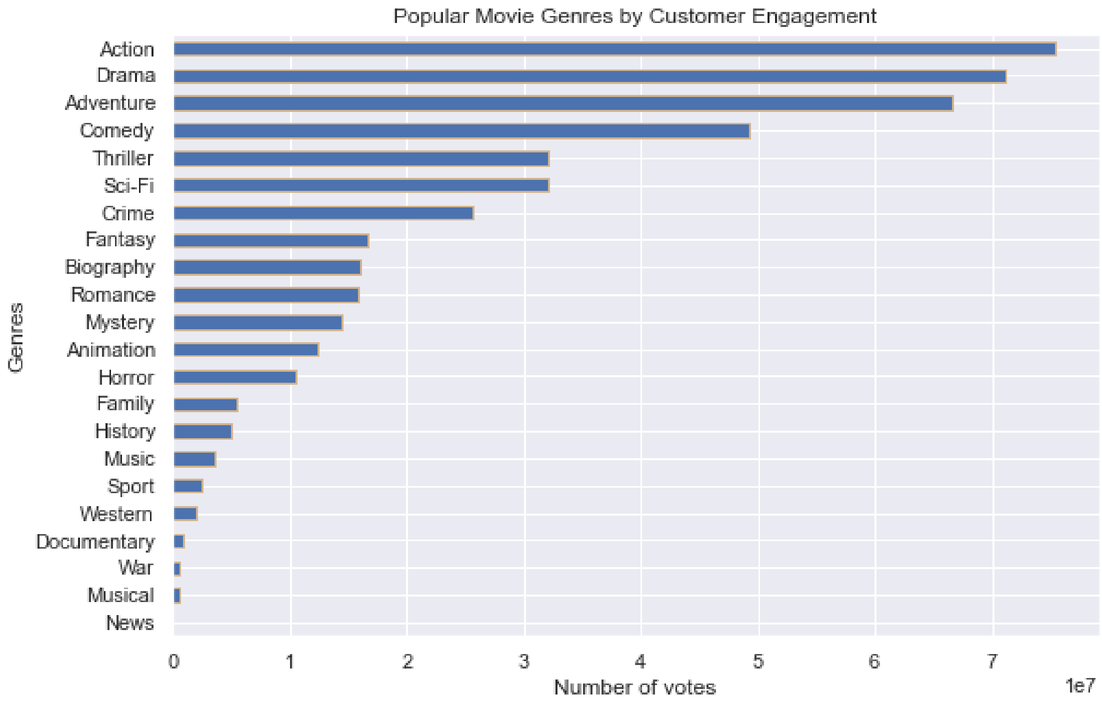


Results show drama, action, and comedy are the most frequently made movies.

In [45]:

```
# Graph by genre and number of votes - or number of customer interaction
df_explode.groupby(['genres'])['numvotes'].sum().sort_values().plot(kind
                                                        ='barh',figsize=(9,
```

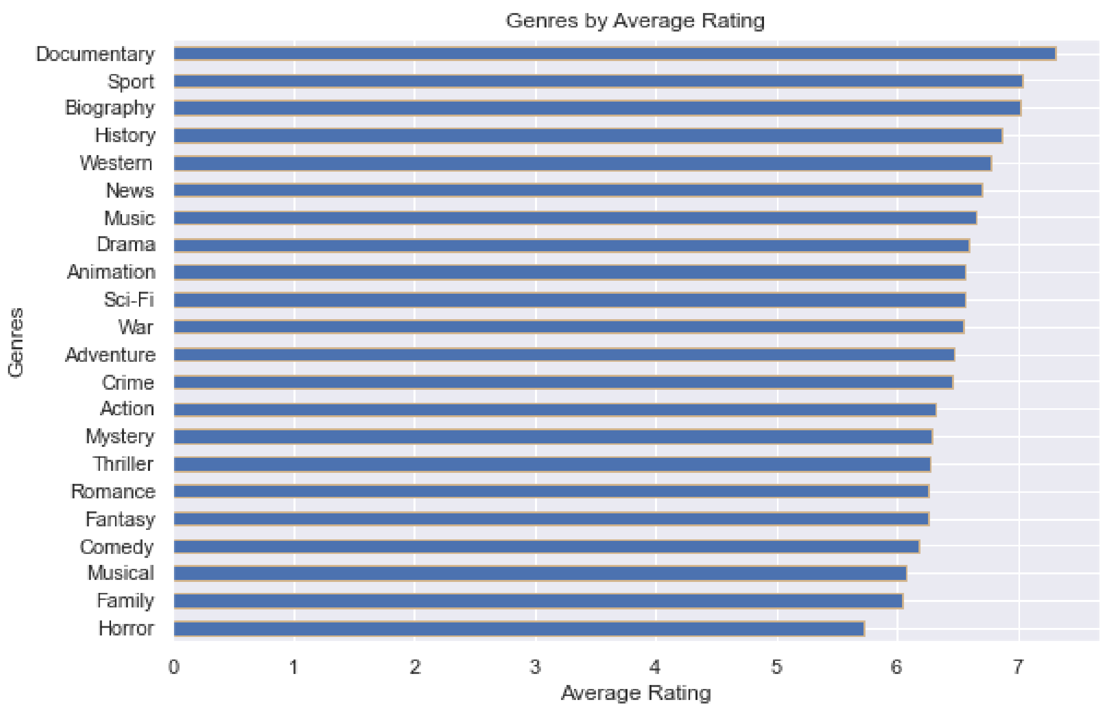
```
6),
                                                                      edgecolor = 'tan')

plt.title('Popular Movie Genres by Customer Engagement')
plt.xlabel("Number of votes")
plt.ylabel('Genres')
sns.set();
```



Results show that action, drama, and adventure films result in a lot more online engagement compared the other genres.

In [46]:

```
# Use df_explode['genres'] and ['averagerating'] to display average rating
df_explode.groupby(['genres'])['averagerating'].mean().sort_values().plot(kind='barh',
                                                                        figsize=(9,6),
                                                                        edgecolor = 'tan'
)
plt.title("Genres by Average Rating")
plt.xlabel('Average Rating')
plt.ylabel('Genres')
sns.set();
```



Results seem to show that documentary, sport, and biography come out as the top three genres with the highest ratings. However, this would be inaccurate to conclude due the low number of votes for those particular genres.

Looking at the previous three graphs, it appears action, drama, and comedy are the most successful genres with audiences.

Evaluation

The visualizations show that the top movie studios today have made an average of 102.7 films between 2010-2019 with that being about 11.4 films a year. The other visualization shows that movies that make a higher domestic gross are between 2 - 2.5 hours long. With the final visualizations, it looks like action, drama, and comedy are the most frequently made films that also create the most 'buzz'/customer engagement.

Conclusions

With all this in mind, I would recommend Microsoft to make a movie that is between 2 to 2.5 hours long and also to consider a film in the action, drama, or comedy genre. Additionally, I would recommend making about 11.4 films a year in order to compete with the top studios. This analysis has gaps due to the small data set and with only including domestic gross as a measure of earnings. To improve this project, I would like to work with foreign gross and cost of production data to understand the bigger picture of potential earnings per film.

In [ ]:

```
```