movie-data-analysis-checkpoint

March 12, 2023

1 Final Project Submission

Please fill out:

- Student name: Ian Tulienge
- Student pace: Full time.
- Scheduled project review date/time:.
- Instructor name:.
- Blog post URL: https://github.com/Lawez/project-1phase.git

Box Office Trends Analysis: Insights for Microsoft's New Movie Studio project

1.1 Defining the Question

What types of movies are currently performing the best at the box office, and how can we use this information to inform the types of films that Microsoft's new movie studio should create?

To answer this question, we need to analyze data on the performance of movies in terms of box office revenue, critical acclaim, and audience reception. We need to identify the most successful movie genres, trends in size and budget of successful movies, the studios responsible for the most successful movies, and the factors that contribute to the success of a movie.

The results of our analysis will provide actionable insights for Microsoft's new movie studio on what types of movies to create. We may find that certain genres are more popular with audiences or that movies with larger budgets tend to perform better at the box office. We may also identify successful marketing strategies or other factors that contribute to a movie's success. Based on this information, Microsoft's new movie studio can make informed decisions about the types of films to create that are most likely to be successful at the box office.

Dataset provided

- Box office movie dataset https://github.com/learn-co-curriculum/dsc-phase-1-projectv24/blob/master/zippedData/bom.movie_gross.csv.gz
- movie_basics https://github.com/learn-co-curriculum/dsc-phase-1-project-v2-4/blob/master/zippedData/im.db.zip
- movie_ratings https://github.com/learn-co-curriculum/dsc-phase-1-project-v2-4/blob/master/zippedData/im.db.zip

Data grocery

- title: the title of the movie
- studio: the studio that produced the movie
- dometic_gross:the amount of money the movie earned in the domestic market (in US dollars)
- foreign_gross:the amount of money the movie earned in foreign markets (in US dollars)
- movie_id:: unique identifier for each movie
- primary_title: Title of the movie
- original_title: the title of the movie in its original language.
- runtime_minutes: the duration of the movie in minutes.
- genres: the genre(s) of the movie.
- averagerating:Ratings of the movie
- numvotes: Number of votes for the movie
- year: the year the movie was released.
- start_year: the year the movie was originally released.

1.2 Specifying the data analytic question

What is the performance trend of each movie genre in terms of box office revenue, critical acclaim, and audience reception, and how can we use this information to inform the types of films that Microsoft's new movie studio should create?

1.3 Defining the metric for success

There are many different factors that can contribute to a film's success.Some common metrics used to evaluate a movie's performance include:

- Revenue: This is the total amount of money a movie earns at the box office, including both domestic and foreign gross.
- Critical acclaim: This refers to the positive reviews and ratings a movie receives from professional film critics.
- Audience reception: This refers to the response of audiences to a movie, including factors such as audience ratings and reviews.

The metric for success in the movie industry will depend on the goals and priorities of the studio or filmmakers involved In many cases, a combination of these factors will be used to evaluate a movie's success and determine its overall impact.

1.4 Understanding the Context

Microsoft is entering the highly competitive movie industry with the goal of producing successful films that generate revenue. However, as a newcomer to the industry, they lack experience and understanding of the factors that contribute to a movie's success. Box office revenue, critical acclaim, and audience reception are key performance metrics, with movie genres playing a significant role in determining success. This project aims to analyze the performance of different movie genres, identify trends and patterns, and provide actionable insights for Microsoft's new movie studio to inform their decisions when creating new films. By understanding the context and factors influencing the movie industry, the project can help Microsoft create successful movies and achieve their goals.

1.5 Recording the Experimental Design

- upload and read our csv files
- upload and read our SQLite database
- Data Cleaning and Preparation: Clean the data by removing any duplicates, missing values, and incorrect data.
- perfom EDA (Exploratory Data Analysis) understanding the data at hand and identifying patterns, trends, and relationships between variables.
- Conclusion
- Recommendation

1.6 Assessing the Relevance of the Data

The data are rellevant as it has all data and values related to genres,ratings and number of votes

2 Loading and reading Our Datasets

[1]:
```
#importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# Reading the bom movie data set
df=pd.read_csv('bom.movie_gross.csv')
```

3 Checking the Data

[3]:
```
# Determining the no. of records in our result dataset
print("bom.movie_gross.csv shape", df.shape)
```

```
bom.movie_gross.csv shape (3387, 5)
```

The data has 3387 records and 5 variables

[4]:
```
#Cheaking the top 5 of our bom.movie_gross
df.head()
```

[4]:

```
                                         title studio  domestic_gross  \
0                                  Toy Story 3     BV     415000000.0
1                   Alice in Wonderland (2010)     BV     334200000.0
2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0
3                                    Inception     WB     292600000.0
4                          Shrek Forever After   P/DW     238700000.0

  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010
```

[5]:

```python
#Cheaking the bottom our bom.movie_gross
df.tail()
```

[5]:

```
                            title      studio  domestic_gross foreign_gross  \
3382                    The Quake       Magn.          6200.0           NaN
3383  Edward II (2018 re-release)          FM          4800.0           NaN
3384                     El Pacto        Sony          2500.0           NaN
3385                     The Swan  Synergetic          2400.0           NaN
3386            An Actor Prepares       Grav.          1700.0           NaN

      year
3382  2018
3383  2018
3384  2018
3385  2018
3386  2018
```

The dataset is uniform from top to bottom

[6]:

```python
#Cheaking information about our data that it has the correct data types
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
```

Dataset has integers, float and string data type forign_gross has many missing values and few in domestic_gross

[7]:
```
#provides information such as count, mean, standard deviation, minimum and␣
↪maximum values, as well as the quartiles of the data.
df.describe()
```

[7]:

	domestic_gross	year
count	3.359000e+03	3387.000000
mean	2.874585e+07	2013.958075
std	6.698250e+07	2.478141
min	1.000000e+02	2010.000000
25%	1.200000e+05	2012.000000
50%	1.400000e+06	2014.000000
75%	2.790000e+07	2016.000000
max	9.367000e+08	2018.000000

4 Data Validation

Validation - The data was valid since it can be confirmed on Box office movies website and Idmb database

5 Tidying the Dataset

5.1 Data cleaning of bom movies

This involves checking the data for any missing or inconsistent values, and correcting or removing them as necessary.

5.1.1 Cheaking for duplicates

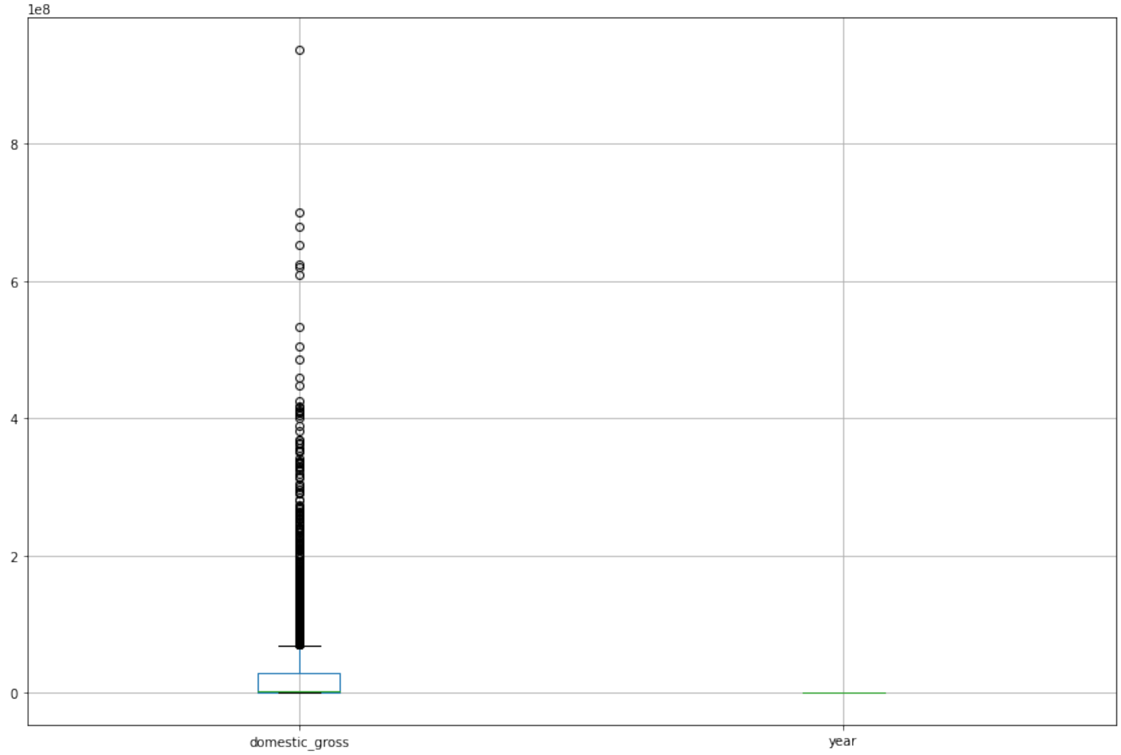
[8]:
```
# checks for duplicates and list them
df[df.duplicated()]
```

[8]:
```
Empty DataFrame
Columns: [title, studio, domestic_gross, foreign_gross, year]
Index: []
```

No duplicates in the bom data

5.1.2 Cheking for outliers

[9]:
```
#checking for outliers
plt.figure(figsize = (15, 10))
df.boxplot()
plt.show()
```



observation : there are outliers in domestic_gross column hence I will consider dropping it

5.1.3 cheaking for missing values

[10]:
```
#cheaking for missing values where it returns True for missing values
df.isnull()
```

[10]:

	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
…	…	…	…	…	…
3382	False	False	False	True	False
3383	False	False	False	True	False
3384	False	False	False	True	False
3385	False	False	False	True	False
3386	False	False	False	True	False

[3387 rows x 5 columns]

[11]:
```
#Identfying of missing values and getting the sum of each column
df.isnull().sum().sort_values(ascending=False) # sort_values(ascending=False)␣
 ↪arranges the values in descending order
```

[11]:
```
foreign_gross     1350
domestic_gross      28
studio               5
year                 0
title                0
dtype: int64
```

[12]:
```
#Checking percentage of missing values and arrange it in decending order
percent_missing = df.isna().sum().sort_values(ascending=False)* 100 / len(df)
percent_missing
```

[12]:
```
foreign_gross     39.858282
domestic_gross     0.826690
studio             0.147623
year               0.000000
title              0.000000
dtype: float64
```

The following do not require any fills:

title

year

The following are missing data:

foreign_gross

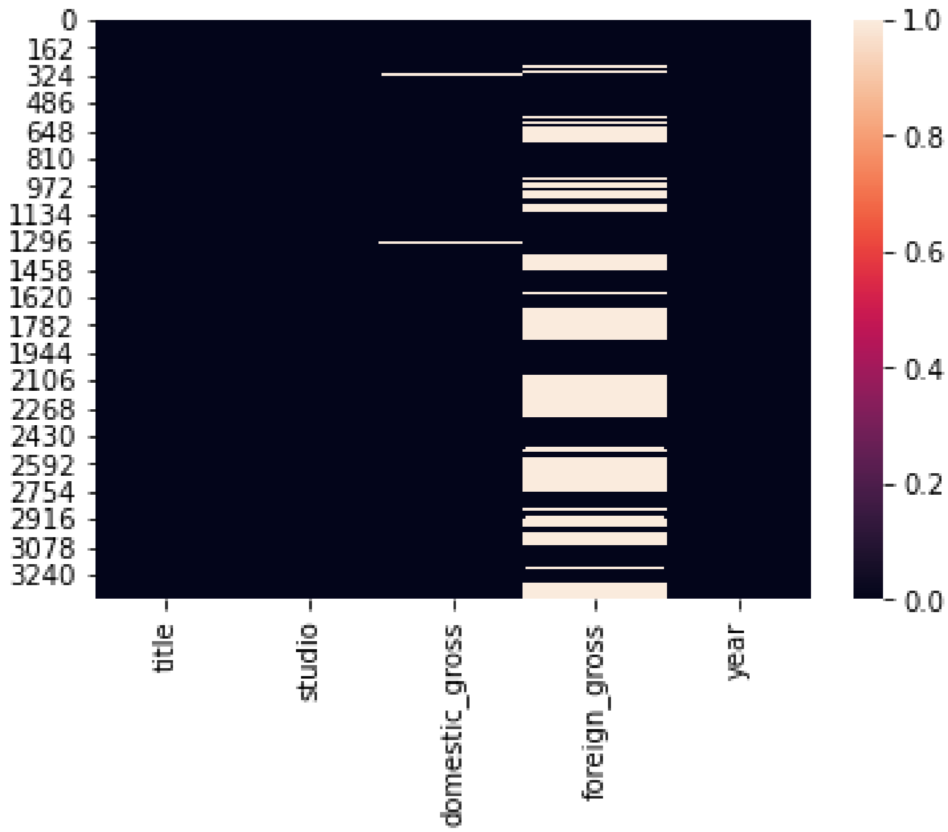
domestic_gross

studio

foreign_gross contains more than half of the null values this can be shown by a heatmap below

[13]:
```
# Using heatmap to show null values
sns.heatmap(df.isna())
```

[13]: <AxesSubplot:>



Hence I consider removing foreign_gross from my dataset

[14]:
```
#Dropping foreign_gross from the data
new_data = df.drop('foreign_gross', axis=1)
new_data #Renaming my data as new_data
```

[14]:
```
                                        title      studio  domestic_gross  \
0                                 Toy Story 3          BV     415000000.0
1                  Alice in Wonderland (2010)          BV     334200000.0
2     Harry Potter and the Deathly Hallows Part 1          WB     296000000.0
3                                   Inception          WB     292600000.0
4                         Shrek Forever After        P/DW     238700000.0
…                                           …           …               …
3382                                The Quake       Magn.          6200.0
3383              Edward II (2018 re-release)          FM          4800.0
3384                                 El Pacto        Sony          2500.0
3385                                 The Swan  Synergetic          2400.0
3386                        An Actor Prepares       Grav.          1700.0

      year
0     2010
```

```
1      2010
2      2010
3      2010
4      2010
…     …
3382   2018
3383   2018
3384   2018
3385   2018
3386   2018

[3387 rows x 4 columns]
```

Filling out missing values of domestic_gross by mean()

[15]:
```python
#Filling  missing values with the mean of domestic_gross (28745845.066984218)
mean_domesticgross = new_data['domestic_gross'].mean()
new_data['domestic_gross'].fillna(mean_domesticgross, inplace=True)
new_data[new_data.domestic_gross.isnull() ] # Cheaking if domestic_gross␣
 ↪missing values have been filled
```

[15]:
```
Empty DataFrame
Columns: [title, studio, domestic_gross, year]
Index: []
```

I have fill missing values of domestic_gross with its mean ie (28745845.066984218)

Filling missing values ie studio by backward filling method

[16]:
```python
#new_data.fillna(method='bfill', inplace=True)
# filling missing values by backward filling method in place,
#which means that the original DataFrame is being updated with the new values␣
 ↪rather than creating a new DataFrame
new_data
```

[16]:
```
                                        title      studio  domestic_gross  \
0                                 Toy Story 3          BV     415000000.0
1                  Alice in Wonderland (2010)          BV     334200000.0
2     Harry Potter and the Deathly Hallows Part 1          WB     296000000.0
3                                   Inception          WB     292600000.0
4                         Shrek Forever After        P/DW     238700000.0
…                                           …         …              …
3382                                The Quake       Magn.          6200.0
3383              Edward II (2018 re-release)          FM          4800.0
3384                                 El Pacto        Sony          2500.0
3385                                 The Swan   Synergetic          2400.0
3386                        An Actor Prepares       Grav.          1700.0
```

```
      year
0     2010
1     2010
2     2010
3     2010
4     2010
…     …
3382  2018
3383  2018
3384  2018
3385  2018
3386  2018

[3387 rows x 4 columns]
```

[17]:
```
#Cheaking of missing value to ensure they have been filled
new_data.isnull().sum()
```

[17]:
```
title            0
studio           5
domestic_gross   0
year             0
dtype: int64
```

No missing data in bom

[18]:
```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('im.db')

# Create a cursor object
cursor = conn.cursor()

# Execute a SQL query to select all the columns from the movie_basics table
query = "SELECT * FROM movie_basics;"
cursor.execute(query)

# Retrieve the results of the query
results = cursor.fetchall()
```

[19]:
```
#Reading of data from the movie_basics table in SQLite database using Pandas␣
↪and displays the resulting DataFrame.
movie_basics = pd.read_sql_query("SELECT * FROM movie_basics", conn)
movie_basics
```

[19]:

```
        movie_id                                 primary_title  \
0       tt0063540                                   Sunghursh
1       tt0066787             One Day Before the Rainy Season
2       tt0069049                  The Other Side of the Wind
3       tt0069204                             Sabse Bada Sukh
4       tt0100275                    The Wandering Soap Opera
…            …                                           …
146139  tt9916538                         Kuambil Lagi Hatiku
146140  tt9916622  Rodolpho Teóphilo - O Legado de um Pioneiro
146141  tt9916706                             Dankyavar Danka
146142  tt9916730                                      6 Gunn
146143  tt9916754              Chico Albuquerque - Revelações

                                     original_title  start_year  \
0                                         Sunghursh        2013
1                                   Ashad Ka Ek Din        2019
2                        The Other Side of the Wind        2018
3                                   Sabse Bada Sukh        2018
4                             La Telenovela Errante        2017
…                                                 …          …
146139                          Kuambil Lagi Hatiku        2019
146140  Rodolpho Teóphilo - O Legado de um Pioneiro        2015
146141                              Dankyavar Danka        2013
146142                                       6 Gunn        2017
146143               Chico Albuquerque - Revelações        2013

        runtime_minutes                genres
0                 175.0    Action,Crime,Drama
1                 114.0       Biography,Drama
2                 122.0                 Drama
3                   NaN          Comedy,Drama
4                  80.0  Comedy,Drama,Fantasy
…                    …                    …
146139            123.0                 Drama
146140              NaN           Documentary
146141              NaN                Comedy
146142            116.0                  None
146143              NaN           Documentary

[146144 rows x 6 columns]
```

[20]:

```
#Cheaking information about movie_basics data
movie_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
```

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   movie_id         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
```

[21]:
```
#Cheaking the shape of our movie_basics data
movie_basics.head()
```

[21]:
```
    movie_id                    primary_title              original_title  \
0  tt0063540                        Sunghursh                   Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2  tt0069049       The Other Side of the Wind  The Other Side of the Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela Errante

   start_year  runtime_minutes                genres
0        2013            175.0    Action,Crime,Drama
1        2019            114.0       Biography,Drama
2        2018            122.0                 Drama
3        2018              NaN          Comedy,Drama
4        2017             80.0  Comedy,Drama,Fantasy
```

[22]:
```
#provides information such as count, mean, standard deviation, minimum and
↪maximum values, as well as the quartiles of the data.
movie_basics.describe()
```

[22]:
```
          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
```

5.2 Data cleaning of movie_basics

This involves checking the data for any missing or inconsistent values, and correcting or removing them as necessary.

5.2.1 Cheaking for duplicate

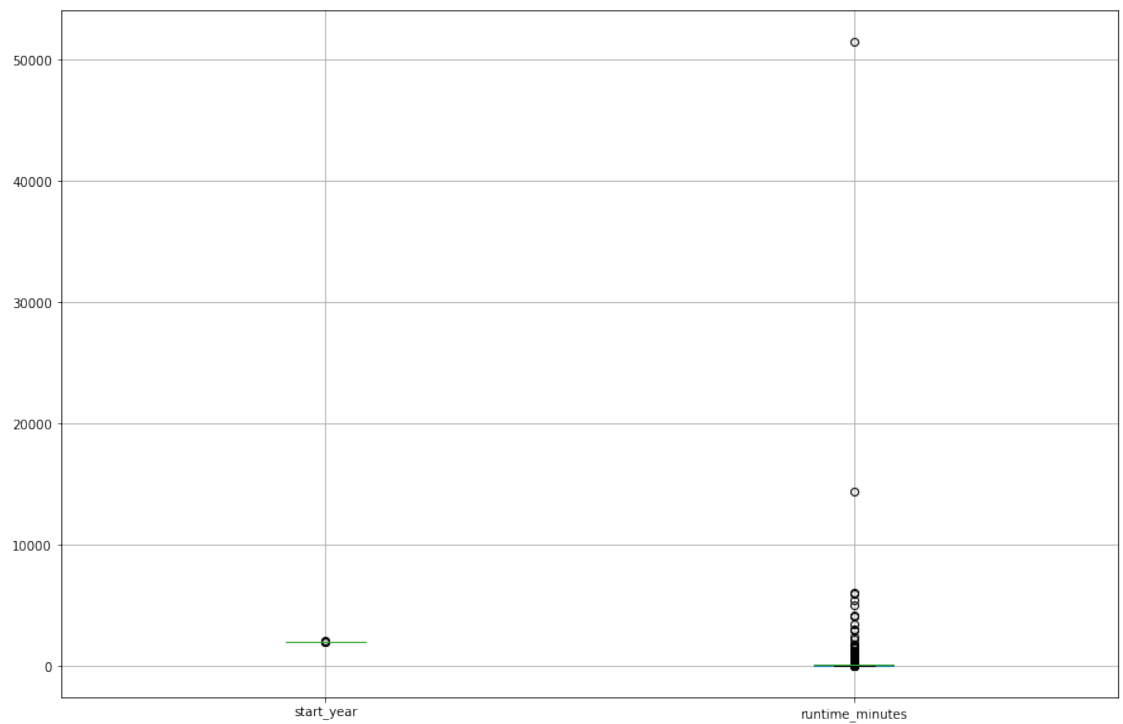
[23]:
```
# checks for duplicates and list them
movie_basics[movie_basics.duplicated()]
```

[23]:
```
Empty DataFrame
Columns: [movie_id, primary_title, original_title, start_year, runtime_minutes,
genres]
Index: []
```

No duplicates in movie_basics

5.2.2 cheaking outliers

[24]:
```
#checking for outliers
plt.figure(figsize = (15, 10))
movie_basics.boxplot()
plt.show()
```



observation : there are outliers in start_year and runtime_minutes they look genuine I won't remove them

5.2.3 cheaking for missing values

[25]:
```
#cheaking for missing values where it returns True for missing values
movie_basics.isnull()
```

[25]:
```
        movie_id  primary_title  original_title  start_year  runtime_minutes  \
0          False          False           False       False            False
1          False          False           False       False            False
2          False          False           False       False            False
3          False          False           False       False             True
4          False          False           False       False            False
…            …            …             …            …              …
146139     False          False           False       False            False
146140     False          False           False       False             True
146141     False          False           False       False             True
146142     False          False           False       False            False
146143     False          False           False       False             True

        genres
0        False
1        False
2        False
3        False
4        False
…        …
146139   False
146140   False
146141   False
146142    True
146143   False

[146144 rows x 6 columns]
```

[26]:
```
#Identfying of missing values and getting the sum of each column
movie_basics.isnull().sum().sort_values(ascending=False) #␣
↪sort_values(ascending=False) arranges the values in descending order
```

[26]:
```
runtime_minutes    31739
genres              5408
original_title        21
start_year             0
primary_title          0
movie_id               0
dtype: int64
```

[27]:
```
#Checking percentage of missing values and arrange it in decending order
```

```
percent_missing = movie_basics.isna().sum().sort_values(ascending=False)* 100 /
 ↪len(df)
percent_missing
```

[27]:
```
runtime_minutes    937.082964
genres             159.669324
original_title       0.620018
start_year           0.000000
primary_title        0.000000
movie_id             0.000000
dtype: float64
```

The following do not require any fills:

start_year

primary_title

movie_id

The following are missing data:

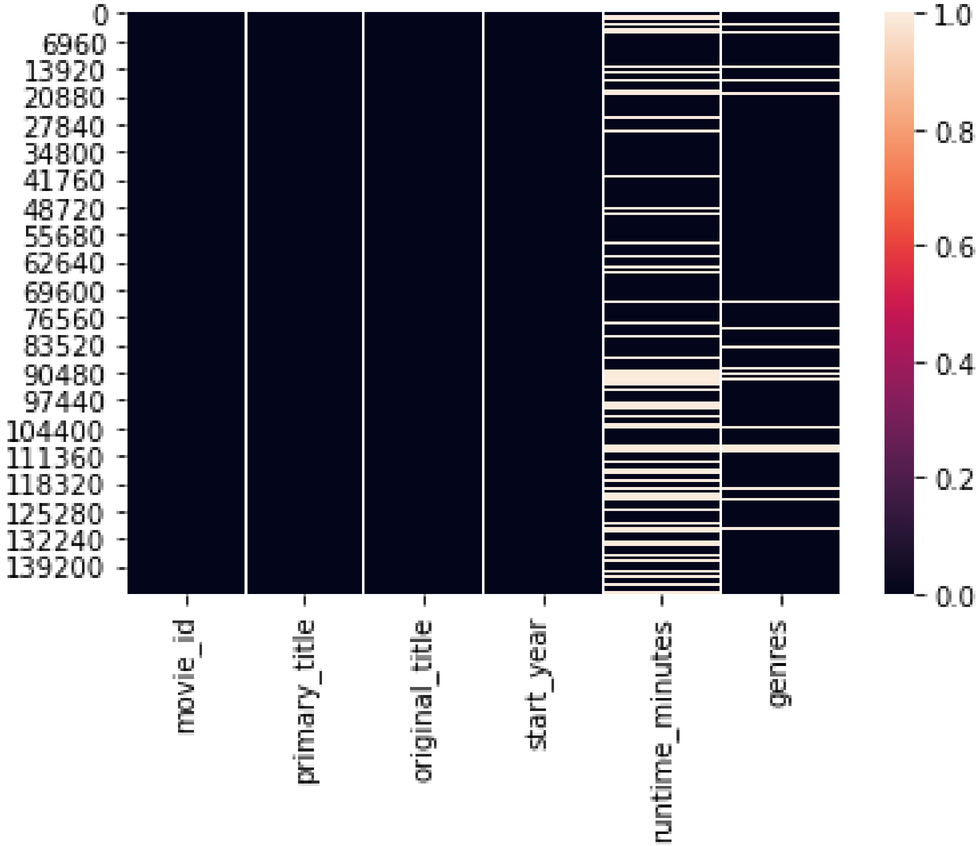
runtime_minutes

genres

original_title

[28]:
```
# Using heatmap to show null values
sns.heatmap(movie_basics.isna())
```

[28]: <AxesSubplot:>



Filling out missing values of runtime_minutes by mean()

[29]:
```
#Filling runtime_minutes missing values with the mean of runtime_minutes (86.
↪18724706088021)
mean_runtime = movie_basics['runtime_minutes'].mean()
movie_basics['runtime_minutes'].fillna(mean_runtime, inplace=True)
movie_basics[movie_basics.runtime_minutes.isnull() ] # Cheaking if␣
↪runtime_minutes missing values have been filled
```

[29]:
```
Empty DataFrame
Columns: [movie_id, primary_title, original_title, start_year, runtime_minutes,
genres]
Index: []
```

I have filled missing values of runtime with its mean ie (86.18724706088021)

Filling out missing data of original_title by primary_title data

[30]:
```
# fill missing values in original_title with values from primary_title
movie_basics['original_title'] = movie_basics['original_title'].
↪fillna(movie_basics['primary_title'])
```

Filling missing values i.e genres by backward filling method

[31]:
```
movie_basics.fillna(method='bfill', inplace=True)
# filling missing values by backward filling method in place,
#which means that the original DataFrame is being updated with the new values␣
↪rather than creating a new DataFrame
movie_basics
```

[31]:
```
         movie_id                                primary_title  \
0       tt0063540                                    Sunghursh
1       tt0066787              One Day Before the Rainy Season
2       tt0069049                   The Other Side of the Wind
3       tt0069204                              Sabse Bada Sukh
4       tt0100275                     The Wandering Soap Opera
…             …                                            …
146139  tt9916538                          Kuambil Lagi Hatiku
146140  tt9916622  Rodolpho Teóphilo - O Legado de um Pioneiro
146141  tt9916706                              Dankyavar Danka
146142  tt9916730                                       6 Gunn
146143  tt9916754               Chico Albuquerque - Revelações

                                     original_title  start_year  \
0                                         Sunghursh        2013
1                                   Ashad Ka Ek Din        2019
2                        The Other Side of the Wind        2018
3                                   Sabse Bada Sukh        2018
4                             La Telenovela Errante        2017
…                                               …         …
146139                          Kuambil Lagi Hatiku        2019
146140  Rodolpho Teóphilo - O Legado de um Pioneiro        2015
146141                              Dankyavar Danka        2013
146142                                       6 Gunn        2017
146143               Chico Albuquerque - Revelações        2013

        runtime_minutes                genres
0            175.000000    Action,Crime,Drama
1            114.000000       Biography,Drama
2            122.000000                 Drama
3             86.187247          Comedy,Drama
4             80.000000  Comedy,Drama,Fantasy
…                  …                   …
146139       123.000000                 Drama
146140        86.187247           Documentary
146141        86.187247                Comedy
146142       116.000000           Documentary
146143        86.187247           Documentary

[146144 rows x 6 columns]
```

[32]:
```
#Cheaking of missing value to ensure they have been filled
movie_basics.isnull().sum()
```

[32]:
```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

No missing data in movie_basics

[33]:
```
# Connect to the SQLite database
conn = sqlite3.connect('im.db')

# Create a cursor object
cursor = conn.cursor()

# Execute a SQL query to select all the columns from the movie_ratings table
query = "SELECT * FROM movie_ratings;"
cursor.execute(query)

# Retrieve the results of the query
results = cursor.fetchall()
```

[34]:
```
#Reading of data from the movie_ratings table in SQLite database using Pandas␣
↪and displays the resulting DataFrame.
movie_ratings = pd.read_sql_query("SELECT * FROM movie_ratings", conn)
movie_ratings
```

[34]:

	movie_id	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 x 3 columns]

[35]:
```
#Cheaking information about movie_ratings data
movie_ratings.info()
```

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

[36]:
```
#Cheaking the shape of our movie_ratings data
movie_ratings.head()
```

[36]:

	movie_id	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

[37]:
```
#provides information such as count, mean, standard deviation, minimum and␣
↪maximum values, as well as the quartiles of the data.
movie_ratings.describe()
```

[37]:

	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

5.3 Data cleaning of movie_ratings

This involves checking the data for any missing or inconsistent values, and correcting or removing them as necessary.

5.3.1 Cheaking for duplicates

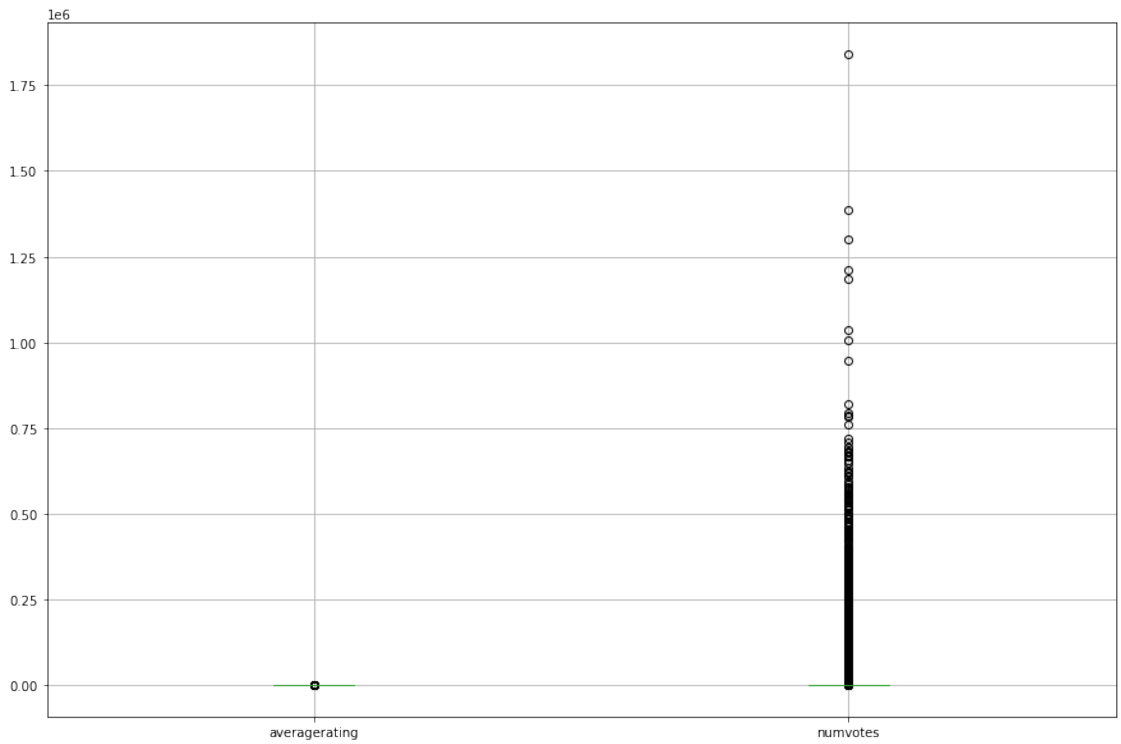
[38]:
```
# checks for duplicates and list them
movie_ratings[movie_ratings.duplicated()]
```

[38]:
```
Empty DataFrame
Columns: [movie_id, averagerating, numvotes]
Index: []
```

No duplicates in movie_ratings

5.3.2 cheaking of outliers

[39]:
```
#checking for outliers
plt.figure(figsize = (15, 10))
movie_ratings.boxplot()
plt.show()
```



There is outliers in numvotes I wont remove them they look genuine

5.3.3 cheaking for missing values

[40]:
```
#cheaking for missing values where it returns True for missing values
movie_ratings.isnull()
```

[40]:
```
      movie_id  averagerating  numvotes
0        False          False     False
1        False          False     False
2        False          False     False
3        False          False     False
4        False          False     False
…          …              …       …
73851    False          False     False
73852    False          False     False
73853    False          False     False
73854    False          False     False
73855    False          False     False

[73856 rows x 3 columns]
```

[41]:
```
#Identfying of missing values and getting the sum of each column
movie_ratings.isnull().sum().sort_values(ascending=False) #␣
 ↪sort_values(ascending=False) arranges the values in descending order
```

[41]:
```
numvotes         0
averagerating    0
movie_id         0
dtype: int64
```

No missing data in movie_ratings

Now merging all my three clean data sets i.e(bom movie_gross, movie_basics and movie_ratings)

Merging movie_basics and movie_ratings on common column i.e (movie_id)

[42]:
```
#Merging the two tables together
merged_data = pd.merge(movie_basics, movie_ratings, on='movie_id')
merged_data
```

[42]:
```
        movie_id                    primary_title              original_title  \
0      tt0063540                        Sunghursh                   Sunghursh
1      tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2      tt0069049       The Other Side of the Wind  The Other Side of the Wind
3      tt0069204                  Sabse Bada Sukh             Sabse Bada Sukh
4      tt0100275         The Wandering Soap Opera       La Telenovela Errante
…          …                                …                           …
73851  tt9913084                 Diabolik sono io            Diabolik sono io
73852  tt9914286                Sokagin Çocuklari           Sokagin Çocuklari
```

```
73853  tt9914642                        Albatross                   Albatross
73854  tt9914942       La vida sense la Sara Amat  La vida sense la Sara Amat
73855  tt9916160                       Drømmeland                  Drømmeland

       start_year  runtime_minutes                 genres  averagerating  \
0            2013       175.000000     Action,Crime,Drama            7.0
1            2019       114.000000        Biography,Drama            7.2
2            2018       122.000000                  Drama            6.9
3            2018        86.187247           Comedy,Drama            6.1
4            2017        80.000000   Comedy,Drama,Fantasy            6.5
…             …              …                     …             …
73851        2019        75.000000            Documentary            6.2
73852        2019        98.000000           Drama,Family            8.7
73853        2017        86.187247            Documentary            8.5
73854        2019        86.187247            Documentary            6.6
73855        2019        72.000000            Documentary            6.5

       numvotes
0            77
1            43
2          4517
3            13
4           119
…           …
73851         6
73852       136
73853         8
73854         5
73855        11

[73856 rows x 8 columns]
```

Merging (movie__basics and movie__ratings) merged__data with the bom movie__gross

[43]:
```
#Renaming primary name to title in the movie_basics and movie_ratings merged␣
↪data
df1=merged_data #defing merged data as df1
df1 = df1.rename(columns={'primary_title': 'title'})
df1
```

[43]:
```
         movie_id                            title              original_title  \
0       tt0063540                        Sunghursh                   Sunghursh
1       tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2       tt0069049       The Other Side of the Wind  The Other Side of the Wind
3       tt0069204                  Sabse Bada Sukh             Sabse Bada Sukh
4       tt0100275         The Wandering Soap Opera       La Telenovela Errante
…           …                              …                           …
```

```
73851  tt9913084              Diabolik sono io          Diabolik sono io
73852  tt9914286            Sokagin Çocuklari         Sokagin Çocuklari
73853  tt9914642                    Albatross                 Albatross
73854  tt9914942    La vida sense la Sara Amat  La vida sense la Sara Amat
73855  tt9916160                   Drømmeland                Drømmeland

       start_year  runtime_minutes                genres  averagerating  \
0            2013       175.000000    Action,Crime,Drama            7.0
1            2019       114.000000       Biography,Drama            7.2
2            2018       122.000000                 Drama            6.9
3            2018        86.187247          Comedy,Drama            6.1
4            2017        80.000000  Comedy,Drama,Fantasy            6.5
…             …              …                   …            …
73851        2019        75.000000           Documentary            6.2
73852        2019        98.000000          Drama,Family            8.7
73853        2017        86.187247           Documentary            8.5
73854        2019        86.187247           Documentary            6.6
73855        2019        72.000000           Documentary            6.5

       numvotes
0            77
1            43
2          4517
3            13
4           119
…          …
73851         6
73852       136
73853         8
73854         5
73855        11

[73856 rows x 8 columns]
```

[44]:
```
#merging df1 data to our bom.movie_gross that is define as df
final_data = pd.merge(new_data, df1, on='title')
final_data #Defining my new data as final_data
```

[44]:
```
                            title  studio  domestic_gross  year   movie_id  \
0                     Toy Story 3      BV     415000000.0  2010  tt0435761
1                       Inception      WB     292600000.0  2010  tt1375666
2             Shrek Forever After    P/DW     238700000.0  2010  tt0892791
3      The Twilight Saga: Eclipse    Sum.     300500000.0  2010  tt1325004
4                      Iron Man 2    Par.     312400000.0  2010  tt1228705
…                             …      …            …    …        …
3022                     Souvenir  Strand         11400.0  2018  tt2387692
3023                     Souvenir  Strand         11400.0  2018  tt2389092
```

```
3024         Beauty and the Dogs   Osci.          8900.0  2018  tt6776572
3025                   The Quake   Magn.          6200.0  2018  tt6523720
3026            An Actor Prepares   Grav.          1700.0  2018  tt5718046

                  original_title  start_year  runtime_minutes  \
0                    Toy Story 3        2010            103.0
1                      Inception        2010            148.0
2            Shrek Forever After        2010             93.0
3     The Twilight Saga: Eclipse        2010            124.0
4                     Iron Man 2        2010            124.0
…                            …         …                …
3022                    Souvenir        2016             90.0
3023                    Souvenir        2014             86.0
3024              Aala Kaf Ifrit        2017            100.0
3025                    Skjelvet        2018            106.0
3026           An Actor Prepares        2018             97.0

                          genres  averagerating  numvotes
0     Adventure,Animation,Comedy            8.3    682218
1        Action,Adventure,Sci-Fi            8.8   1841066
2     Adventure,Animation,Comedy            6.3    167532
3        Adventure,Drama,Fantasy            5.0    211733
4        Action,Adventure,Sci-Fi            7.0    657690
…                            …            …       …
3022         Drama,Music,Romance            6.0       823
3023              Comedy,Romance            5.9         9
3024        Crime,Drama,Thriller            7.0      1016
3025       Action,Drama,Thriller            6.2      5270
3026                      Comedy            5.0       388

[3027 rows x 11 columns]
```

[45]:
```python
#Cheaking the shape of our data
final_data.shape
```

[45]:
```
(3027, 11)
```

[46]:
```python
final_data[final_data['averagerating'] == 1.6]['genres']
```

[46]:
```
309         Documentary,Music
2987     Comedy,Drama,Romance
Name: genres, dtype: object
```

[47]:
```python
#Cheaking information about our data
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 3027 entries, 0 to 3026
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   title            3027 non-null   object
 1   studio           3024 non-null   object
 2   domestic_gross   3027 non-null   float64
 3   year             3027 non-null   int64
 4   movie_id         3027 non-null   object
 5   original_title   3027 non-null   object
 6   start_year       3027 non-null   int64
 7   runtime_minutes  3027 non-null   float64
 8   genres           3027 non-null   object
 9   averagerating    3027 non-null   float64
 10  numvotes         3027 non-null   int64
dtypes: float64(3), int64(3), object(5)
memory usage: 283.8+ KB
```

[48]:
```python
#provides information such as count, mean, standard deviation, minimum and␣
↪maximum values, as well as the quartiles of the data.
final_data.describe()
```

[48]:
```
       domestic_gross         year   start_year  runtime_minutes  \
count    3.027000e+03  3027.000000  3027.000000      3027.000000
mean     3.062656e+07  2014.077635  2013.783284       106.890585
std      6.647351e+07     2.442245     2.466955        20.086427
min      1.000000e+02  2010.000000  2010.000000         3.000000
25%      1.445000e+05  2012.000000  2012.000000        93.000000
50%      2.100000e+06  2014.000000  2014.000000       104.000000
75%      3.210000e+07  2016.000000  2016.000000       118.000000
max      7.001000e+08  2018.000000  2019.000000       272.000000

       averagerating      numvotes
count    3027.000000  3.027000e+03
mean        6.457582  6.170030e+04
std         1.012277  1.255132e+05
min         1.600000  5.000000e+00
25%         5.900000  2.117000e+03
50%         6.600000  1.310900e+04
75%         7.100000  6.276550e+04
max         9.200000  1.841066e+06
```

[49]:
```python
final_data.isnull().sum() #check for missing values and brings out the sum
```

[49]:
```
title              0
studio             3
domestic_gross     0
```

```
year                0
movie_id            0
original_title      0
start_year          0
runtime_minutes     0
genres              0
averagerating       0
numvotes            0
dtype: int64
```

No missing values in our final_data its ready for analysis

6 Perfoming EDA(Exploratory Data Analysis)

It is a way of analyzing, visualizing and summarizing data in order to understand its characteristics and identify patterns and trends

Visualizations

6.1 Genres Analysis

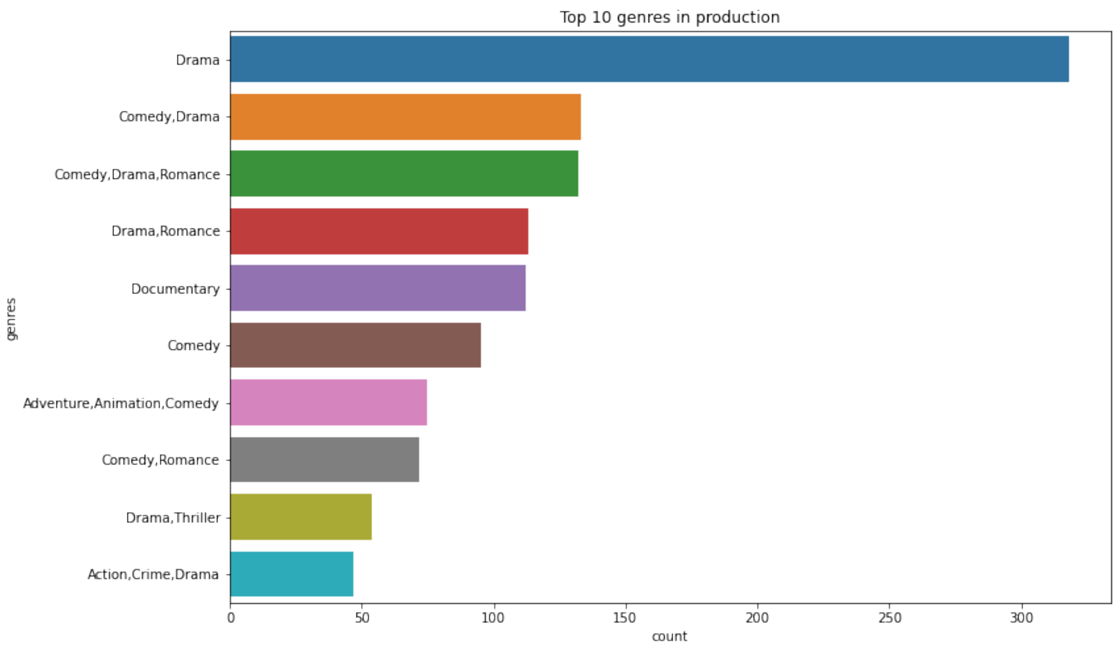
6.1.1 Lets look at the top ten genres that have been watched

[50]:
```
# value_counts() method shows the count of different categories in a given␣
↪column
final_data['genres'].value_counts().head(10)
```

[50]:
```
Drama                          318
Comedy,Drama                   133
Comedy,Drama,Romance           132
Drama,Romance                  113
Documentary                    112
Comedy                          95
Adventure,Animation,Comedy      75
Comedy,Romance                  72
Drama,Thriller                  54
Action,Crime,Drama              47
Name: genres, dtype: int64
```

[51]:
```
# create a figure and axis object
plt.figure(figsize=(12,8))
# create a countplot
sns.countplot(y='genres',order=final_data['genres'].value_counts().index[0:10],␣
↪data=final_data)
plt.title('Top 10 genres in production') # add a title to the plot
```

[51]:
```
Text(0.5, 1.0, 'Top 10 genres in production')
```



6.1.2 Lets find out the top 10 genres with average runtime and average no of votes

[52]:
```
# Group by genre and calculate the mean runtime and mean number of votes
genre_stats = final_data.groupby('genres').agg({'runtime_minutes': 'mean',␣
↪'numvotes': 'mean'})

# Get the top 10 genres with the highest average number of votes
top_genres = genre_stats.sort_values(by='numvotes', ascending=False).head(10)

# Reset the index to start from 1
top_genres = top_genres.reset_index().rename(columns={'genres': 'Genre',␣
↪'runtime_minutes': 'Average Runtime (minutes)', 'numvotes': 'Average Votes'})
top_genres.index = pd.RangeIndex(start=1, stop=len(top_genres)+1)

# Print the top 10 genres in a table with an index starting from 1
print('Top 10 genres with highest average number of votes and average runtime:')
print(top_genres.to_string(index=True, justify='left'))
```

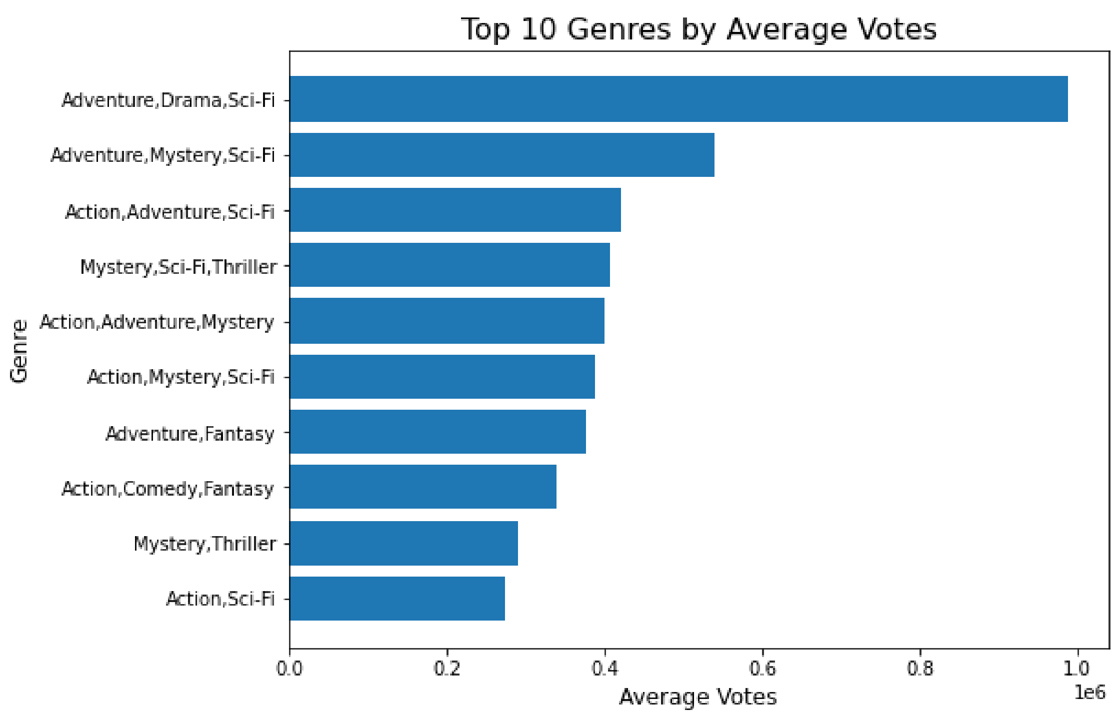
```
Top 10 genres with highest average number of votes and average runtime:
   Genre                       Average Runtime (minutes)  Average Votes
1      Adventure,Drama,Sci-Fi  156.500000                 989725.000000
2    Adventure,Mystery,Sci-Fi  124.000000                 538720.000000
3     Action,Adventure,Sci-Fi  130.617021                 419616.851064
4     Mystery,Sci-Fi,Thriller  108.500000                 406532.500000
5    Action,Adventure,Mystery  139.000000                 399703.000000
6       Action,Mystery,Sci-Fi  113.000000                 387038.000000
```

```
7          Adventure,Fantasy  139.666667                    375770.333333
8      Action,Comedy,Fantasy  112.000000                    339338.000000
9           Mystery,Thriller  108.000000                    290034.500000
10             Action,Sci-Fi  109.500000                    273938.000000
```

[53]:
```
# Create a horizontal bar chart
fig, ax = plt.subplots(figsize=(8, 6))
ax.barh(top_genres['Genre'], top_genres['Average Votes'])

# Set the chart title and axis labels
ax.set_title('Top 10 Genres by Average Votes', fontsize=16)
ax.set_xlabel('Average Votes', fontsize=12)
ax.set_ylabel('Genre', fontsize=12)

# Invert the y-axis to show the genres in descending order
ax.invert_yaxis()
```



6.1.3 Lets find out movie genre with the highest number of votes

[54]:
```
# Group by genre and sum the number of votes
genre_votes = final_data.groupby('genres')['numvotes'].sum()

# Get the genre with the highest number of votes
highest_voted_genre = genre_votes.idxmax()
```

```
# Print the genre with the highest number of votes
print(f'Genre with the highest number of votes: {highest_voted_genre}')
```

```
Genre with the highest number of votes: Action,Adventure,Sci-Fi
```

Action,Adventure and Sci-Fi are the highest voted genres

6.1.4 Lets find out movie genre with the highest domestic gross

[55]:
```
# Group by genre and sum the domestic gross
genre_gross = final_data.groupby('genres')['domestic_gross'].sum()

# Get the genre with the highest domestic gross
highest_gross_genre = genre_gross.idxmax()

# Print the genre with the highest domestic gross
print(f'Genre with the highest domestic gross: {highest_gross_genre}')
```

```
Genre with the highest domestic gross: Action,Adventure,Sci-Fi
```

Action,Adventure and Sci-Fi have the highest domestic gross

6.1.5 Lets find out movie genre with the highest ratings

[56]:
```
# Group by genre and calculate the mean rating
genre_ratings = final_data.groupby('genres')['averagerating'].mean()

# Get the genres with the highest rating
highest_rated_genres = genre_ratings[genre_ratings == genre_ratings.max()]

# Print the genres with the highest rating
print('Genres with the highest rating:')
for genre in highest_rated_genres.index:
    print(genre)
```

```
Genres with the highest rating:
Adventure
```

Adventure genre is the movie with the highest ratings

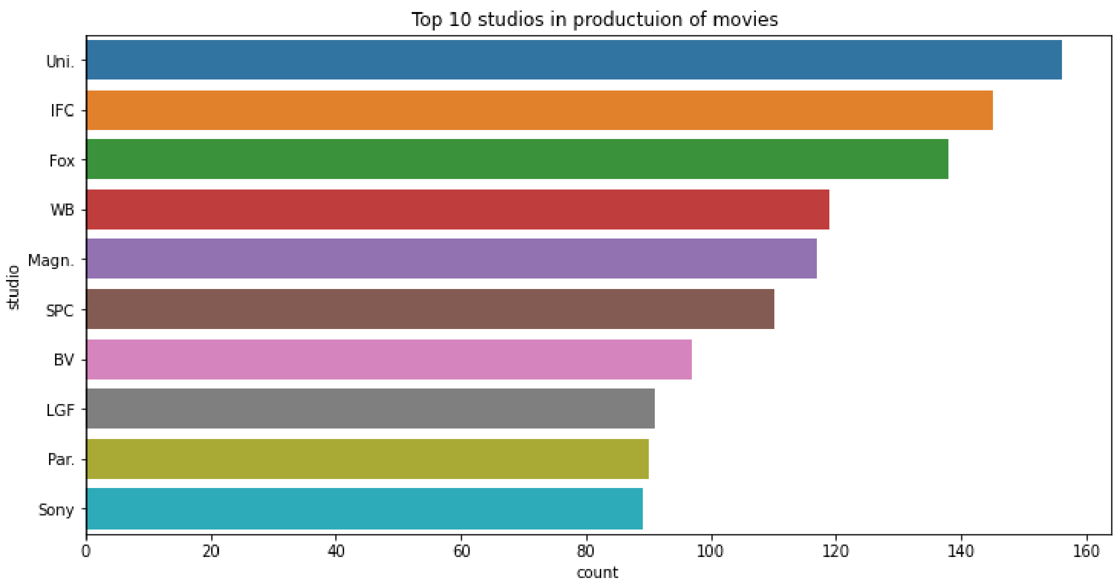
6.2 Studio Analysis

6.2.1 Lets look at the top 10 studios in movie production

[57]:
```
# create a figure and axis object
plt.figure(figsize=(12,6))
# create a countplot
```

```
sns.countplot(y='studio',order=final_data['studio'].value_counts().index[0:10],␣
↪data=final_data)
plt.title('Top 10 studios in productuion of movies ') # add a title to the plot
```

[57]: Text(0.5, 1.0, 'Top 10 studios in productuion of movies ')



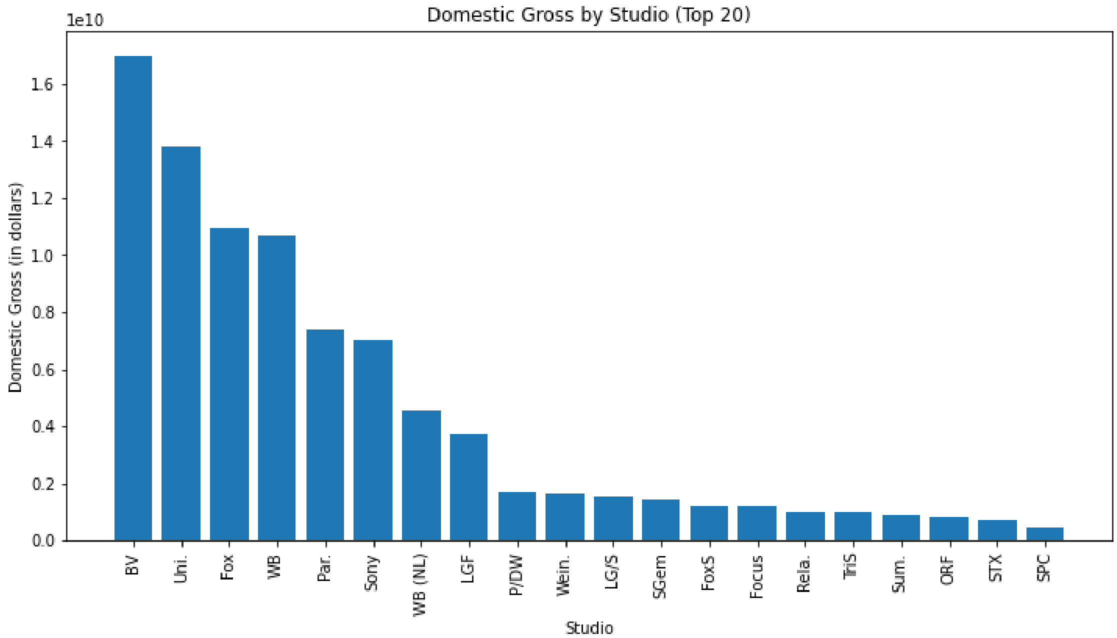
6.2.2 Lets find the top 20 studio by domestic_gross

[58]:
```
# group the data by studio and calculate the sum of domestic gross
grouped_data = final_data.groupby('studio')['domestic_gross'].sum().
↪reset_index()

# sort the data in descending order of domestic gross
sorted_data = grouped_data.sort_values('domestic_gross', ascending=False)

# select the top 20 studios by domestic gross
top_studios = sorted_data.head(20)

# create a bar chart of domestic gross by studio for the top 20 studios
plt.figure(figsize=(12, 6))
plt.bar(top_studios['studio'], top_studios['domestic_gross'])
plt.xticks(rotation=90)
plt.xlabel('Studio')
plt.ylabel('Domestic Gross (in dollars)')
plt.title('Domestic Gross by Studio (Top 20)')
plt.show()
```



BV,Uni and Fox are the leading studios interms of domestic gross

6.3 Ratings Analysis

6.3.1 Lets find descriptive statistics

[59]:
```
#computes descriptive statistics of the "averagerating" column in the␣
↪"final_data" dataset.
final_data["averagerating"].describe()
```

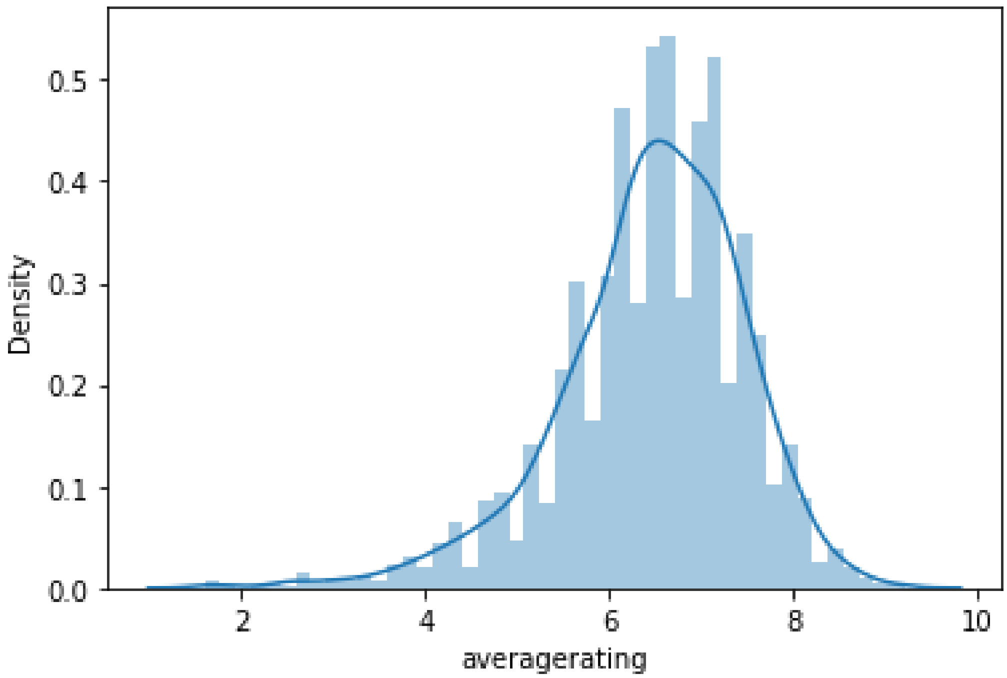
[59]:
```
count    3027.000000
mean        6.457582
std         1.012277
min         1.600000
25%         5.900000
50%         6.600000
75%         7.100000
max         9.200000
Name: averagerating, dtype: float64
```

[60]:
```
#calculates the range of the "averagerating" column in the "final_data" dataset.
a=final_data['averagerating'].max()
b=final_data['averagerating'].min()
range = a-b
print("averagerating range", range)
```

```
averagerating range 7.6
```

[61]:
```
#creates a density plot of the averagerating
plt.figure(figsize = (6, 4))
sns.distplot(final_data.averagerating);
```

```
C:\Users\iantu\anaconda3\envs\learn-env\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

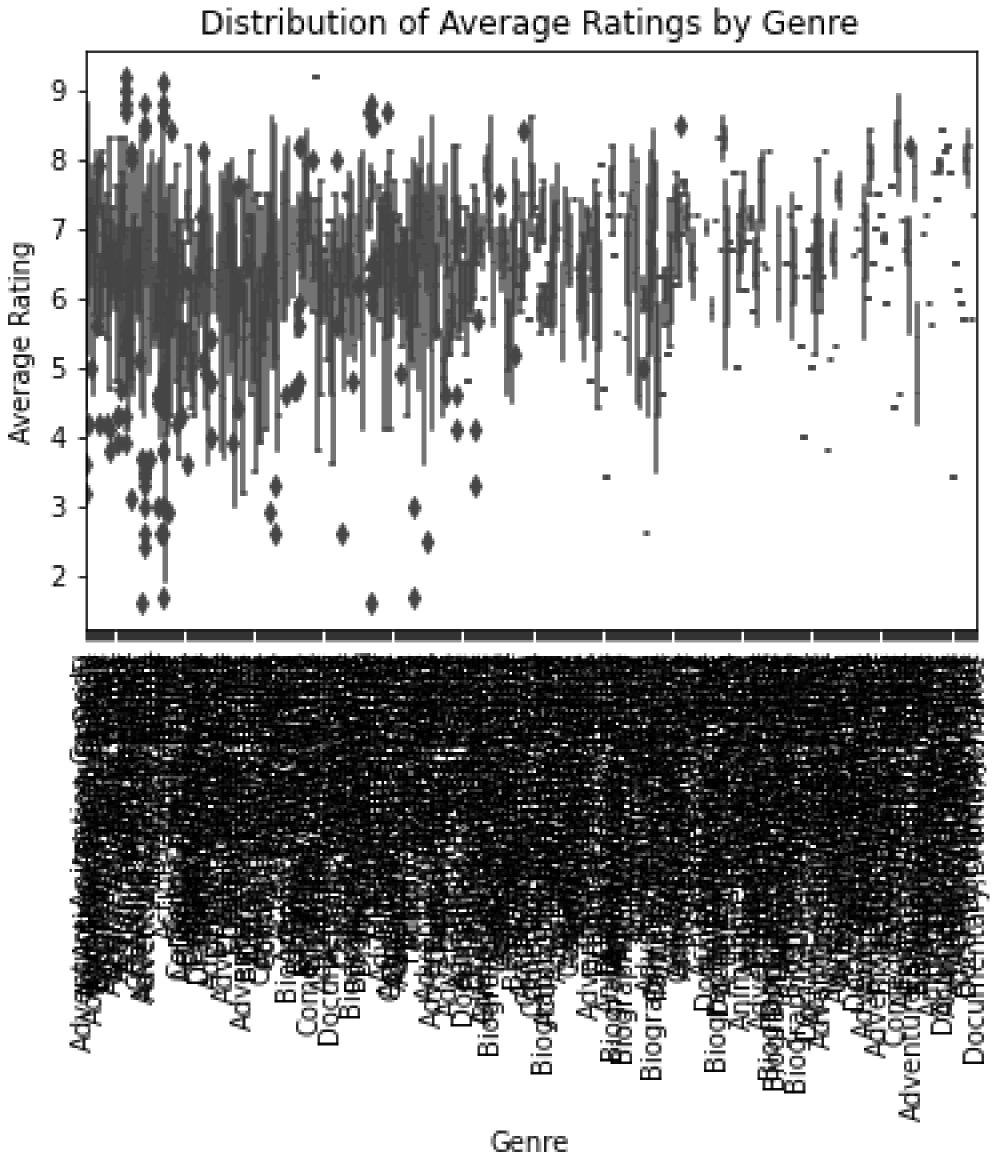


- The averagerating range was 7.6
- The mean averagerating of movie was 6.45
- The maximum averagerating of movie was 9.2
- The minimum averagerating of movie was 1.6
- Distribution of averagerating is close to normal distribution. there was a higher rating of movie between 6 and 7.8 meaning these are the higher ratings.

6.3.2 Let's look at the distribution of average ratings for different genres.

[62]:
```
sns.boxplot(x='genres', y='averagerating', data=final_data)
plt.xticks(rotation=90)
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title('Distribution of Average Ratings by Genre')
```

```
plt.show()
```



Based on the chart, it seems like documentaries, music, and biography films have the highest average ratings among all genres, while horror and action films have the lowest. This could suggest that audiences are generally more critical of horror and action films, while they are more appreciative of documentaries, music, and biography films. However, it is important to note that this is just an average and there are many individual films within each genre that may not fit this trend. However, there is a lot of variability in the data, so again it is difficult to draw strong conclusions from this plot.

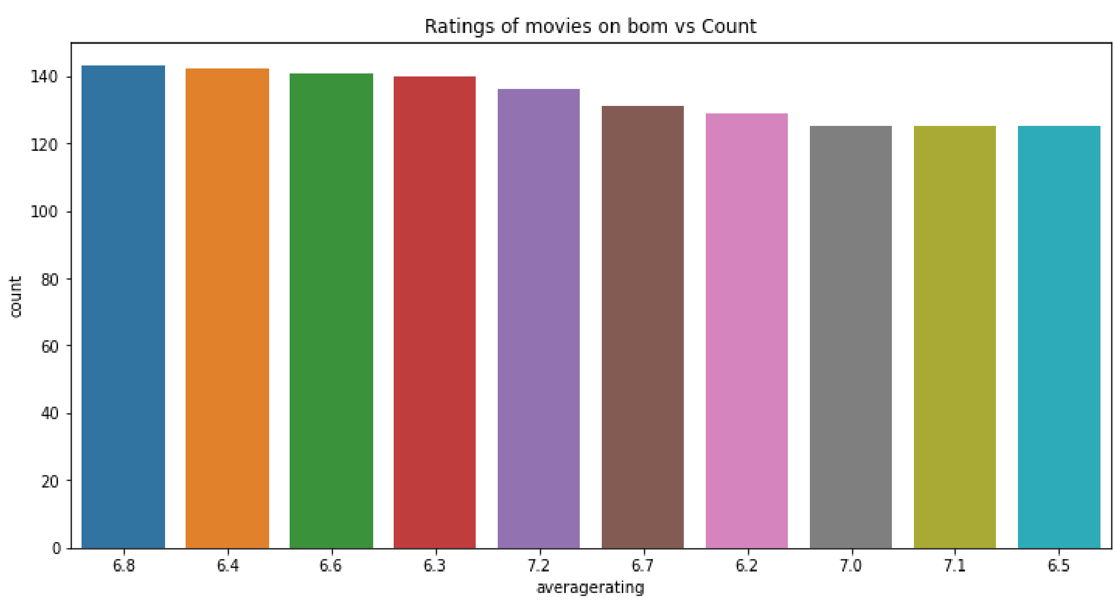
6.3.3 Lets check at major ratings given to bom movies

[63]:
```
# get count of unique values in averagerating column
final_data.averagerating.value_counts()
```

[63]:
```
6.8    143
6.4    142
6.6    141
6.3    140
7.2    136
      …
8.9      1
1.9      1
2.1      1
3.1      1
9.1      1
Name: averagerating, Length: 71, dtype: int64
```

[64]:
```
plt.figure(figsize=(12,6)) # create a figure and axis object
# create a countplot
sns.countplot(x='averagerating',order=final_data['averagerating'].
 ↪value_counts().index[0:10],data=final_data)
plt.title('Ratings of movies on bom vs Count') # add a title to the plot
```

[64]:
```
Text(0.5, 1.0, 'Ratings of movies on bom vs Count')
```



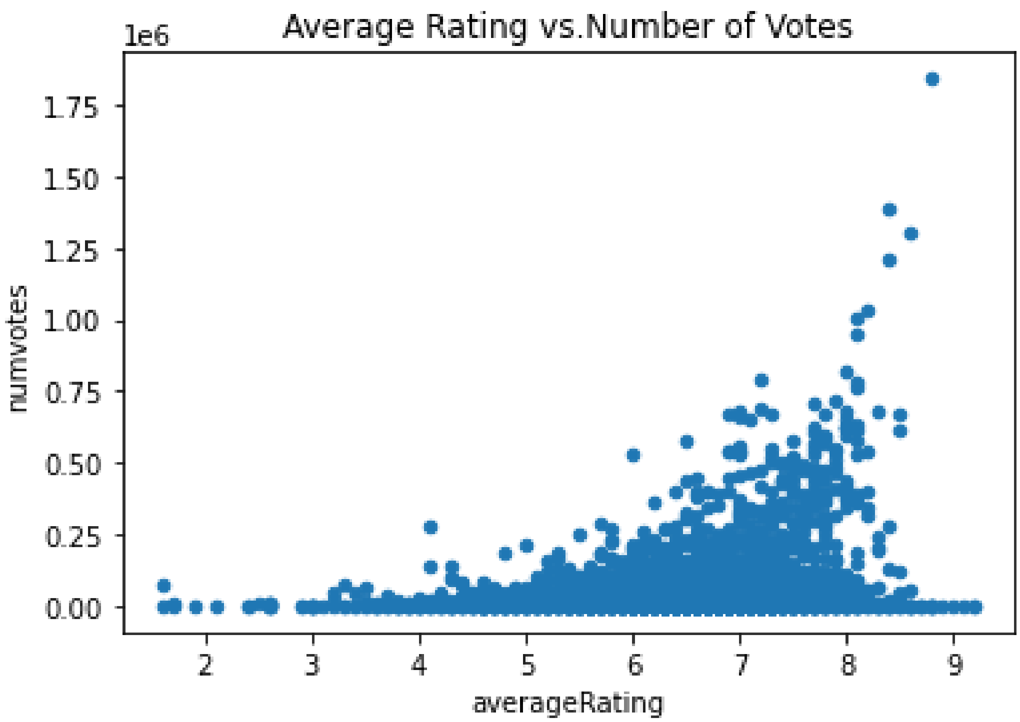
Most movies have an average of 6.8 and 6.4

6.3.4 visualization of averageratings based on numvotes on bom movies

[65]:
```python
# Create a scatter plot
final_data.plot(x='averagerating', y='numvotes', kind='scatter')

# Set the x and y axis labels
plt.xlabel('averageRating')
plt.ylabel('numvotes')

# Set the title of the graph
plt.title('Average Rating vs.Number of Votes')
plt.show()
```



The chart indicates that there is a weak positive correlation between the two variables, with higher-rated movies generally having more votes. However, there are also many movies with a high number of votes but a lower rating, indicating that the popularity of a movie is not always directly related to its quality or critical reception. This information can be useful for understanding the relationship between movie ratings and popularity and for identifying which movies may be more likely to succeed in terms of audience engagement and overall reception.

6.3.5 Lets find out movie with the maximum averagerating

[66]:
```
# Sort the movies based on the number of ratings in descending order
sorted_movies = final_data.sort_values('averagerating', ascending=False)

# Select the movie with the highest number of ratings (i.e., the first row in␣
↪the sorted dataframe)
highest_rated_movie = sorted_movies.iloc[0]

# Print out the title of the highest rated movie and its number of ratings
print('The movie with the highest number of ratings is',␣
↪highest_rated_movie['title'], 'with', highest_rated_movie['averagerating'],␣
↪'ratings.')
```

```
The movie with the highest number of ratings is The Runaways with 9.2 ratings.
```

The Runaways is the highest rated movie in bom ratings with 9.2

6.4 Year Analysis

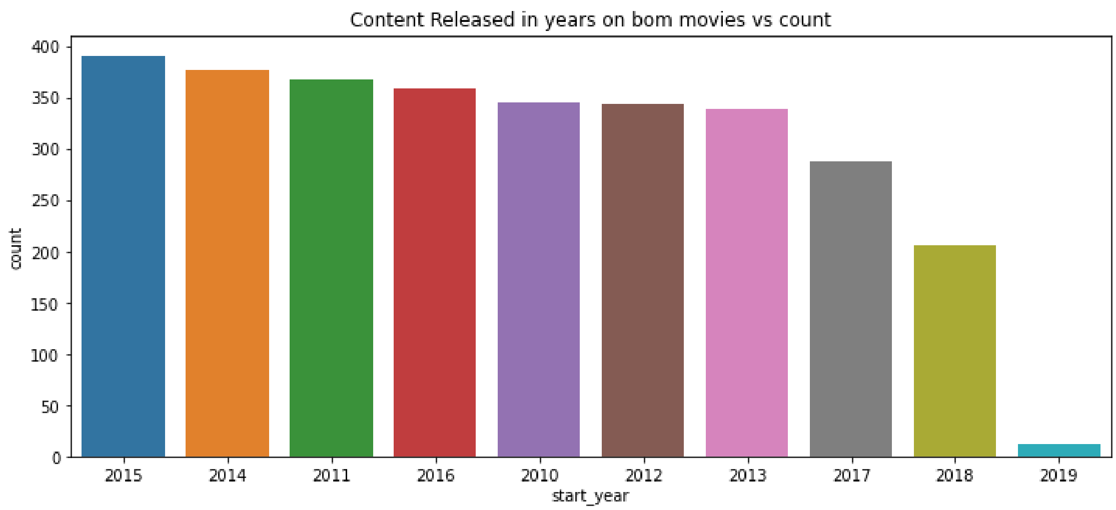
6.4.1 visualization of movies by year of release

[67]:
```
# returns a count start_year.
final_data.start_year.value_counts()
```

[67]:
```
2015    391
2014    376
2011    368
2016    358
2010    345
2012    344
2013    339
2017    288
2018    206
2019     12
Name: start_year, dtype: int64
```

[68]:
```
plt.figure(figsize=(12,5)) # create a figure and axis object
# create a countplot
sns.countplot(x='start_year',order=final_data['start_year'].value_counts().
↪index[0:10],data=final_data)
plt.title('Content Released in years on bom movies vs count') # add a title to␣
↪the plot
```

[68]:
```
Text(0.5, 1.0, 'Content Released in years on bom movies vs count')
```



6.5 Votes Analysis

6.5.1 Lets find descriptive statistics

[69]:
```
#computes descriptive statistics of the "averagerating" column in the␣
↪"final_data" dataset.
final_data["numvotes"].describe()
```

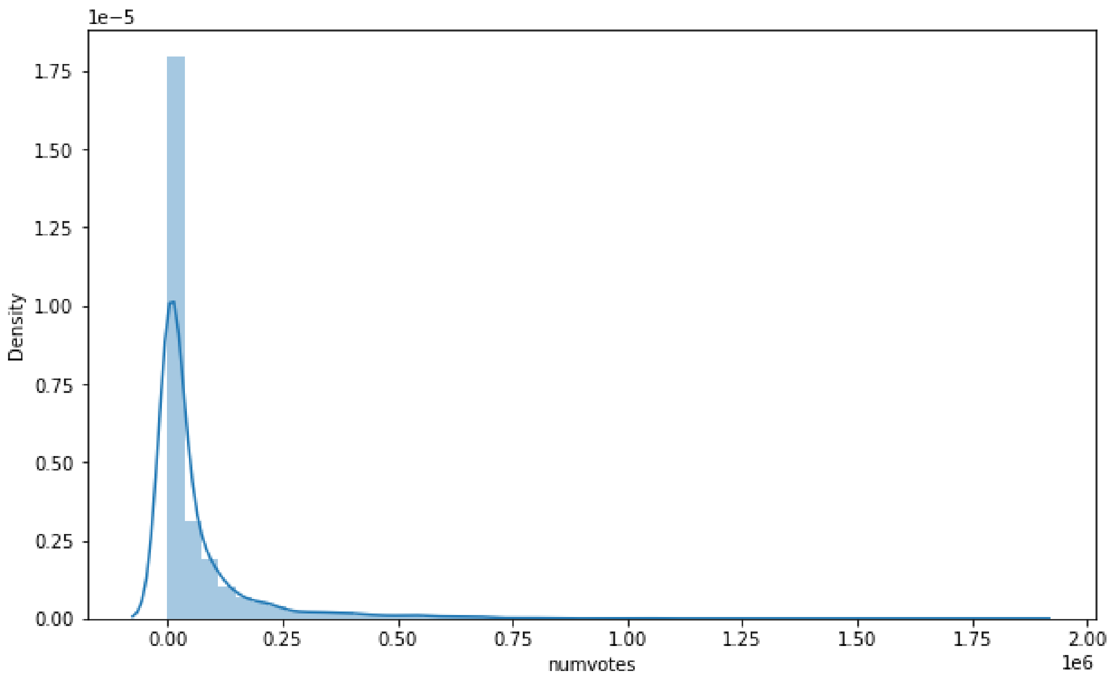
[69]:
```
count    3.027000e+03
mean     6.170030e+04
std      1.255132e+05
min      5.000000e+00
25%      2.117000e+03
50%      1.310900e+04
75%      6.276550e+04
max      1.841066e+06
Name: numvotes, dtype: float64
```

[70]:
```
#calculates the range of the numvotes column in the "final_data" dataset.
a=final_data['numvotes'].max()
b=final_data['numvotes'].min()
range = a-b
print("numvotes range", range)
```

```
numvotes range 1841061
```

[71]:
```
#creates a density plot of the numvotes
plt.figure(figsize = (10, 6))
sns.distplot(final_data.numvotes);
```

```
C:\Users\iantu\anaconda3\envs\learn-env\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



- The numvotes range was 1841061
- The mean numvotes of movie was 61,700.3
- The maximum numvotes of movie was 1841066
- The minimum numvotes of movie was 5

6.5.2 Lets find out movie with maximum number of votes

[72]:
```
# Sort the movies based on the number of votes in descending order
sorted_movies = final_data.sort_values('numvotes', ascending=False)

# Select the movie with the highest number of votes (i.e., the first row in the␣
↪sorted dataframe)
highest_voted_movie = sorted_movies.iloc[0]

# Print out the title of the highest voted movie and its number of votes
print('The movie with the highest number of votes is',␣
↪highest_voted_movie['title'], 'with', highest_voted_movie['numvotes'],␣
↪'votes.')
```

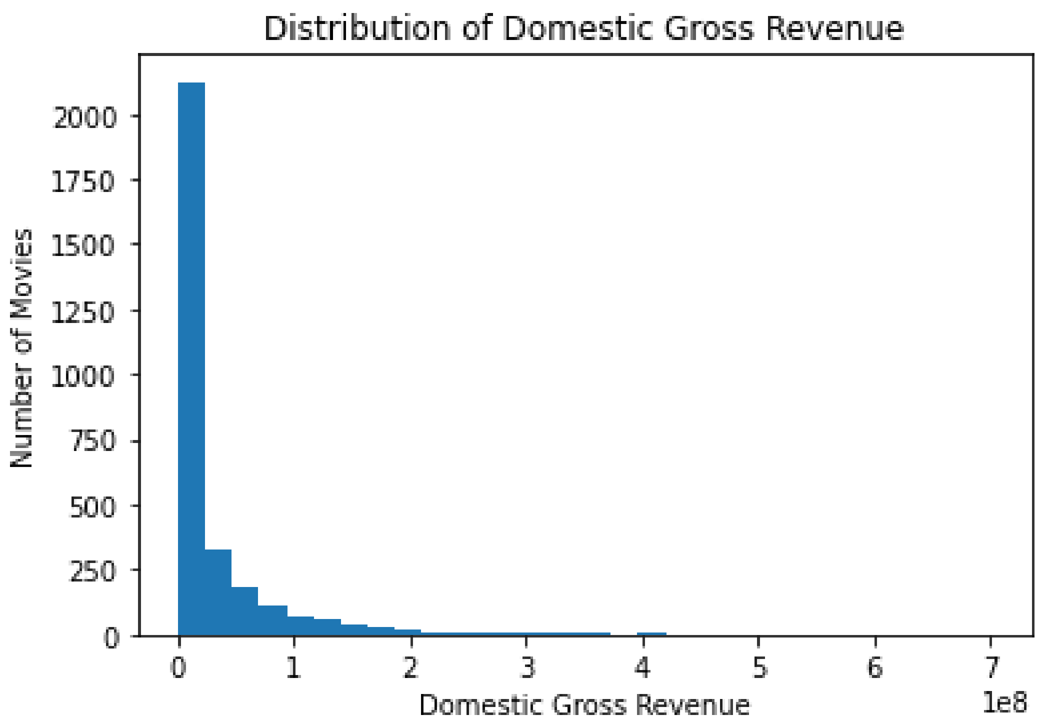
```
The movie with the highest number of votes is Inception with 1841066 votes.
```

Inception is the best movie in bom by votes

6.6 Domestic_gross Analysis

6.6.1 Let's take a look at the distribution of domestic gross revenue.

[73]:
```
plt.hist(final_data['domestic_gross'], bins=30)
plt.xlabel('Domestic Gross Revenue')
plt.ylabel('Number of Movies')
plt.title('Distribution of Domestic Gross Revenue')
plt.show()
```

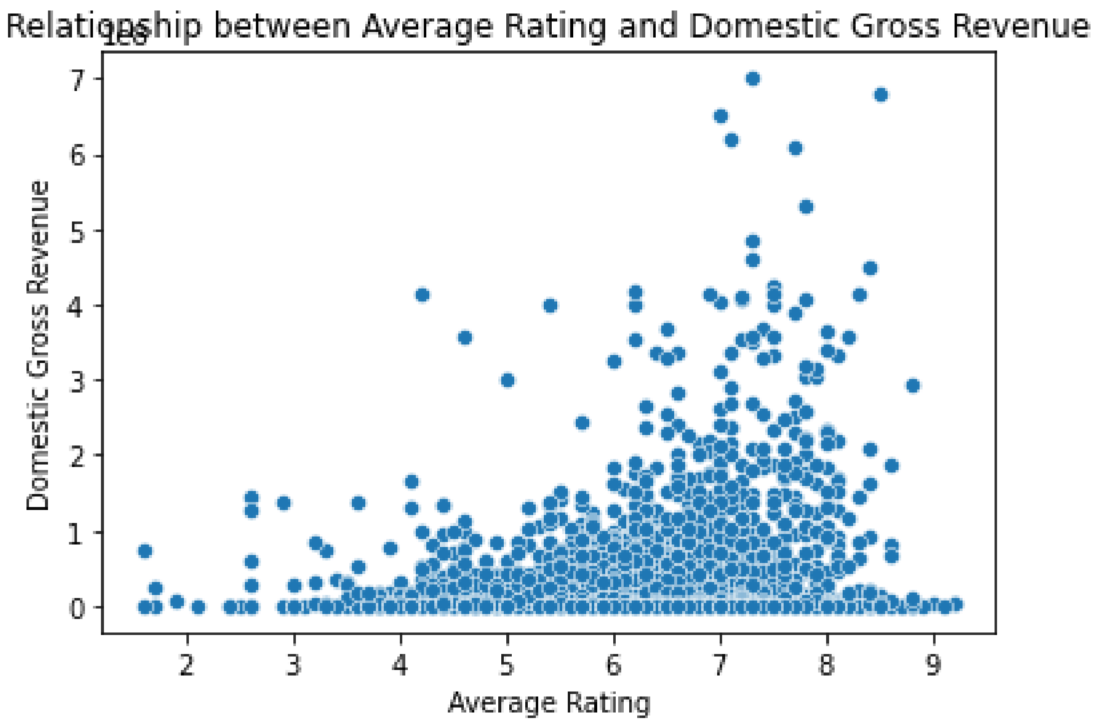


We can see that the majority of movies have a domestic gross revenue of less than $50 million, with a long tail of higher grossing movies. The domestic gross also tends to increase. However, it is also evident that there are many movies with low budgets that have performed well in terms of domestic gross.

6.6.2 Let's look at the relationship between domestic gross revenue and average rating

[74]:
```
# create a scatter plot
sns.scatterplot(x='averagerating', y='domestic_gross', data=final_data)
# Set the x and y axis labels
plt.xlabel('Average Rating')
```

```
plt.ylabel('Domestic Gross Revenue')
# Set the title of the graph
plt.title('Relationship between Average Rating and Domestic Gross Revenue')
plt.show()
```



The chart indicates that there is a weak positive correlation between the two variables, with higher-rated movies generally having higher domestic gross revenue. However, there are also many movies with a high domestic gross revenue but a lower rating, indicating that financial success is not always directly related to critical reception or quality. This information can be useful for understanding the relationship between movie ratings and financial performance and for identifying which movies may be more likely to succeed in terms of box office revenue and overall reception.

7 Findings and Results

So we can perform lots of operations over the dataset to dig out information from. I conclude by:

- Drama is the most watched genre followed by comedy from the bom data.
- Adventure,drama and Sci-Fi are most voted genres.
- Action,Adventure and Sci-Fi have the highest domestic gross.
- Adventure genre is the movie with the highest ratings.
- Uni studio is the studio with highest number of movies production in bom data.

- There is an average rating of movies of (7.6).
- There is a weak positive correlation between the two variables, with higher-rated movies generally having more votes. However, there are also many movies with a high number of votes but a lower rating, indicating that the popularity of a movie is not always directly related to its quality or critical reception. This information can be useful for understanding the relationship between movie ratings and popularity and for identifying which movies may be more likely to succeed in terms of audience engagement and overall reception.
- The Runaways is the highest rated movie in bom ratings of 9.2.
- The year 2015 was the year that had the highest movie production from bom data.
- Inception is the best movie in bom by votes.

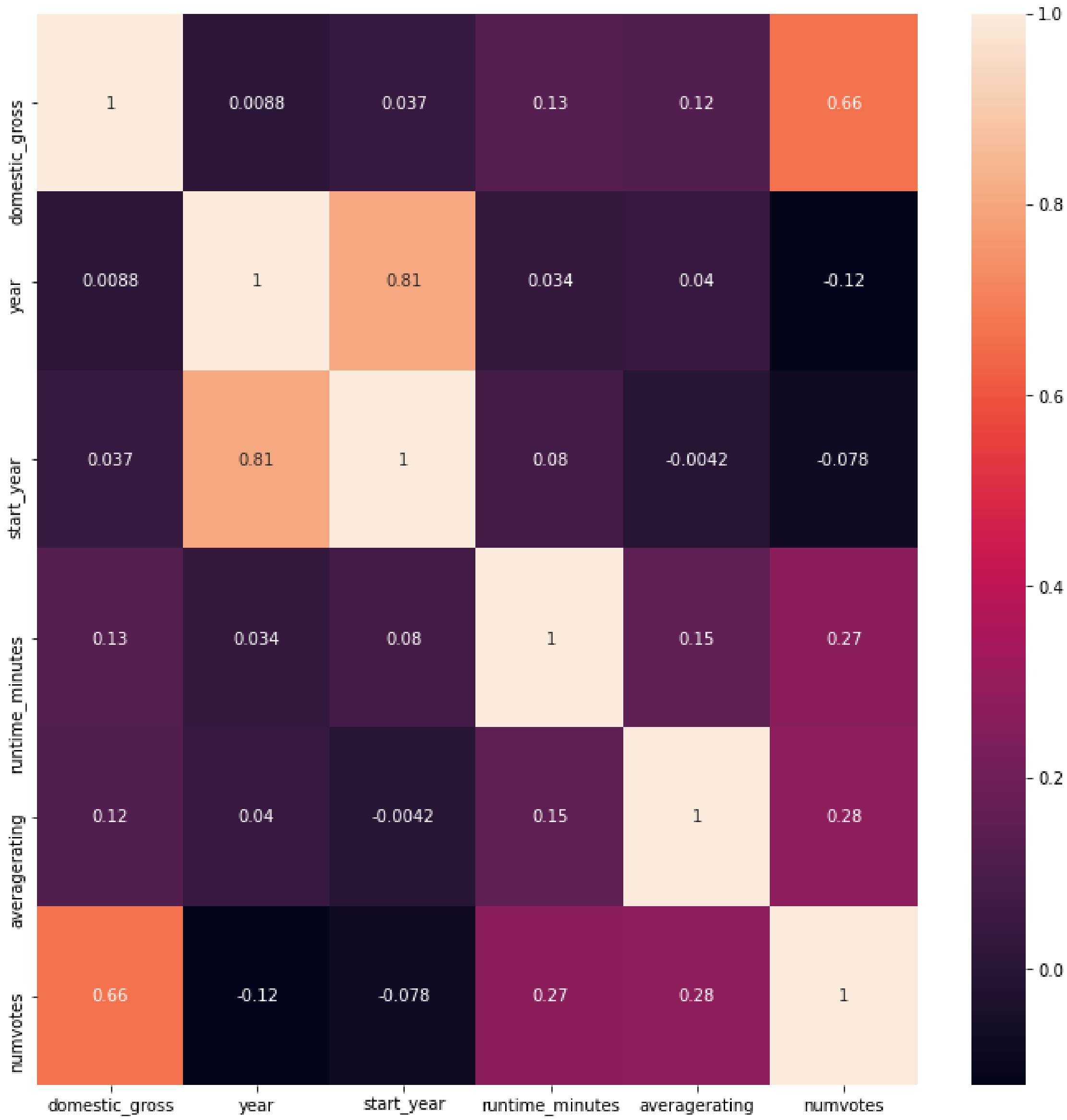
[75]:
```
# Computing the correlation matrix
final_data.corr()
```

[75]:
```
                 domestic_gross      year  start_year  runtime_minutes  \
domestic_gross         1.000000  0.008833    0.037017         0.125615
year                   0.008833  1.000000    0.808273         0.034441
start_year             0.037017  0.808273    1.000000         0.079629
runtime_minutes        0.125615  0.034441    0.079629         1.000000
averagerating          0.118654  0.040499   -0.004172         0.150105
numvotes               0.664029 -0.121836   -0.078001         0.265539

                 averagerating  numvotes
domestic_gross        0.118654  0.664029
year                  0.040499 -0.121836
start_year           -0.004172 -0.078001
runtime_minutes       0.150105  0.265539
averagerating         1.000000  0.278394
numvotes              0.278394  1.000000
```

[76]:
```
#heatmap of the correlation matrix
plt.subplots(figsize=(12,12))
sns.heatmap(final_data.corr(),annot=True)
```

[76]: <AxesSubplot:>



The darker the shade of blue, the stronger the positive correlation between the two variables, while the darker the shade of red, the stronger the negative correlation between the two variables. A value of 1 indicates a perfect positive correlation, a value of -1 indicates a perfect negative correlation, and a value of 0 indicates no correlation between the two variables.

correlation results intepretetion

Based on the correlation matrix, we can make the following observations:

- Domestic gross has a strong positive correlation with the number of votes (0.664), indicating that movies with more votes tend to have higher domestic gross.
- Domestic gross also has a moderate positive correlation with average rating (0.119) and runtime minutes (0.126), indicating that movies with higher ratings and longer runtime tend

to have higher domestic gross.

- Year has a weak positive correlation with start year (0.808) and a weak negative correlation with number of votes (-0.122), indicating that older movies tend to have fewer votes than newer movies.
- Average rating has a moderate positive correlation with the number of votes (0.278), indicating that movies with higher ratings tend to have more votes.
- Runtime minutes has a moderate positive correlation with the number of votes (0.266), indicating that movies with longer runtime tend to have more votes.

Based on the analysis of the data, the following results were obtained:

- Most successful genres: From the analysis of the genres, it was observed that the most successful genres in terms of average rating and gross revenue were Adventure drama and Sci-Fi. Movies in these genres tend to have a higher average rating and gross revenue compared to movies in other genres.
- Successful Studios: BV,Universal,Fox and Warner Bros(WB) were observed to be the most successful studios in terms of gross revenue. These studios have consistently produced movies that have generated high gross revenue.
- Movie runtime: Movies with a runtime of around 120 minutes tend to have a higher average rating compared to movies with shorter or longer runtimes.
- Year: There has been a steady increase in the number of movies produced over the years, with the highest number of movies produced in recent years. However, this has not necessarily translated to an increase in gross revenue.

8 Recommendations:

Based on the analysis:

- It is recommended that movie producers focus on producing movies in the Drama, Adventure and Sci-Fi genres.
- Work with successful studios such as BV, Universal, Fox and Warner Bros(WB), and aim for a runtime of around 120 minutes.
- Consider increasing promotion and advertising efforts for movies with high vote counts to increase the popularity of the movie among audiences.
- Aim to produce movies with an average rating between 6-7, which appears to be the most common rating value in the dataset.

Overall, it is important to consider the genre, ratings, and studio when planning a movie production or distribution strategy. However, it is important to note that success is not guaranteed and there are other factors that can influence a movie's success, such as marketing, timing of release, and competition in the market.

8.1 Next step

- The movie industry is constantly evolving, and Microsoft's new movie studio should remain flexible and adaptable to changing conditions and consumer preferences to stay competitive in the market. Further research and analysis could include exploring the impact of social media and influencer marketing on movie success and analyzing the impact of streaming services on the box office.
- Conduct further analysis on the top-performing genres and studios: Explore the top-performing genres and studios in more detail to gain a deeper understanding of their success factors and identify any common trends or characteristics that contribute to their success.
- Analyze the impact of release dates and marketing strategies: Consider exploring the impact of release dates and marketing strategies on movie success to help identify optimal timing and promotional efforts for new movie releases.
- Develop predictive models: Use machine learning algorithms to build predictive models that can forecast the potential success of a movie based on various factors such as genre, studio, rating, and release date.
- Collaborate with industry experts: Work with industry experts such as producers, directors, or marketing professionals to gain additional insights and perspectives on the movie industry and potential strategies for success.

8.2 Thank You:

Thank you for considering my analysis and recommendations for Microsoft's new movie studio. I believe that by implementing these recommendations, Microsoft can create high-quality movies that appeal to both critics and audiences, generating positive reviews and strong box office performance.