final

February 18, 2024

1 Microsoft Movie Studio (MSMS)

1.1 Authors: Ruth Nyakio Karimi

1.2 Overview

Exploratory data analysis to generate insights for Microsoft executives to reveal opportunities and factors to consider while setting up a new Microsoft Movie studio. Data used from box office and IMDB. Some of the recommendations to Microsoft include top competitors to consider, best performing movies with high budgets and high-performing movies with relatively low budgets

1.3 Business Problem

Microsoft has seen an opportunity in creating original video content and they have decided to create a new movie studio. By doing so they will be able to claim the market share and produce Microsoft original content

1.4 Data Understanding

- Importing the standard packages we need
- Data loading and data previewing ***

[1]:
```
# Importing standard packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

[2]:
```
# listing files in this directory
%ls
```

```
CONTRIBUTING.md                 awesome.gif
Final.ipynb                     dsc-phase1-project-template.ipynb
LICENSE.md                      student.ipynb
```

```
README.md                         zippedData/
Ruth_Nyakio_Project_1.ipynb
```

Data Preview

- In this section we will preview all the datasets provided
- We will also try to understand the all datasets by printing the shape and previewing the first 2 rows
- The goal is to be able to choose the best datasets for the project ##### Important datasets
- imdb.title.basics
- imdb.title.ratings
- bom.movie_gross

[3]:
```
# Exploring the data
# IMDB Title Ratings dataset

imdb_title_ratings = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/
 ↪imdb.title.ratings.csv")
print(imdb_title_ratings.shape) # get the size
imdb_title_ratings.head(2)      # preview the first 2 rows
```

```
(73856, 3)
```

[3]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559

[4]:
```
# Exploring the data
# IMDB Title Basics dataset

imdb_title_basics = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/
 ↪imdb.title.basics.csv")
print(imdb_title_basics.shape) # get the size
imdb_title_basics.info()
imdb_title_basics.head(2) # preview the first 2 rows
```

```
(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    146143 non-null  object
 2   original_title   146122 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
```

[4]:
```
      tconst                    primary_title   original_title  start_year  \
0  tt0063540                        Sunghursh        Sunghursh        2013
1  tt0066787  One Day Before the Rainy Season  Ashad Ka Ek Din        2019

   runtime_minutes              genres
0            175.0  Action,Crime,Drama
1            114.0     Biography,Drama
```

[5]:
```
# Exploring the data
# Bom Movie Gross dataset

bom_movie_gross = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/bom.
 ↪movie_gross.csv")
print(bom_movie_gross.shape)
bom_movie_gross.info()
bom_movie_gross.head(2)
```

```
(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
```

[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
```

[6]:
```
# Exploring the data
# IMDB name basics
imdb_name_basics = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/
 ↪imdb.name.basics.csv")
print(imdb_name_basics.shape)
imdb_name_basics.info()
imdb_name_basics.head(2)
```

```
(606648, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 606648 entries, 0 to 606647
Data columns (total 6 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   nconst              606648 non-null  object
 1   primary_name        606648 non-null  object
 2   birth_year          82736 non-null   float64
 3   death_year          6783 non-null    float64
 4   primary_profession  555308 non-null  object
 5   known_for_titles    576444 non-null  object
dtypes: float64(2), object(4)
memory usage: 27.8+ MB
```

[6]:

	nconst	primary_name	birth_year	death_year	primary_profession	known_for_titles
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer	tt0837562,tt2398241,tt0844471,tt0118553
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department	tt0896534,tt6791238,tt0287072,tt1682940

[7]:
```
# Exploring the data
# Movie budgets

tn_movie_budgets = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/tn.
 ↪movie_budgets.csv")
print(tn_movie_budgets.shape)
tn_movie_budgets.info()
tn_movie_budgets.head(2)
```

```
(5782, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5782 non-null   int64
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
```

```
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

[7]:
```
   id  release_date                                        movie  \
0   1  Dec 18, 2009                                       Avatar
1   2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides

  production_budget domestic_gross worldwide_gross
0      $425,000,000   $760,507,625  $2,776,345,279
1      $410,600,000   $241,063,875  $1,045,663,875
```

[8]:
```
# Exploring the data
# Movie reviews

rt_reviews = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/rt.
 ↪reviews.csv")
print(rt_reviews.shape)
rt_reviews.head(2)
```

```
(54432, 8)
```

[8]:
```
   id                                             review rating   fresh  \
0   3  A distinctly gallows take on contemporary fina…    3/5   fresh
1   3  It's an allegory in search of a meaning that n…    NaN  rotten

           critic  top_critic        publisher               date
0      PJ Nabarro           0  Patrick Nabarro  November 10, 2018
1  Annalee Newitz           0          io9.com       May 23, 2018
```

[9]:
```
# Exploring the data
# Movie dataset

tmdb_movies = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/tmdb.
 ↪movies.csv")
print(tmdb_movies.shape)
tmdb_movies.head(2)
```

```
(26517, 10)
```

[9]:
```
   Unnamed: 0            genre_ids     id original_language  \
0           0      [12, 14, 10751]  12444                en
1           1  [14, 12, 16, 10751]  10191                en

                               original_title  popularity release_date  \
0  Harry Potter and the Deathly Hallows: Part 1      33.533   2010-11-19
1                    How to Train Your Dragon      28.734   2010-03-26

                                  title  vote_average  vote_count
```

```
0  Harry Potter and the Deathly Hallows: Part 1          7.7       10788
1                    How to Train Your Dragon          7.7        7610
```

[10]:
```
# Exploring the data
# Movie info dataset

rt_movie_info = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/rt.
 ↪movie_info.csv")
print(rt_movie_info.shape)
rt_movie_info.head(2)
```

```
(1560, 12)
```

[10]:
```
   id                                           synopsis rating  \
0   1  This gritty, fast-paced, and innovative police…      R
1   3  New York City, not-too-distant-future: Eric Pa…      R

                                genre          director  \
0  Action and Adventure|Classics|Drama  William Friedkin
1    Drama|Science Fiction and Fantasy  David Cronenberg

                        writer  theater_date      dvd_date currency  \
0                Ernest Tidyman   Oct 9, 1971  Sep 25, 2001      NaN
1  David Cronenberg|Don DeLillo  Aug 17, 2012   Jan 1, 2013        $

  box_office      runtime             studio
0        NaN  104 minutes                NaN
1    600,000  108 minutes  Entertainment One
```

[11]:
```
# Exploring the data
# Title principals dataset

imdb_title_principals = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/
 ↪zippedData/imdb.title.principals.csv")
print(imdb_title_principals.shape)
imdb_title_principals.head(2)
```

```
(1028186, 6)
```

[11]:
```
      tconst  ordering     nconst  category  job   characters
0  tt0111414         1  nm0246005     actor  NaN  ["The Man"]
1  tt0111414         2  nm0398271  director  NaN          NaN
```

[12]:
```
# Exploring the data
# Title crew dataset
```

```
imdb_title_crew = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/
 ↪imdb.title.crew.csv")
print(imdb_title_crew.shape)
imdb_title_crew.head(2)
```

```
(146144, 3)
```

[12]:

	tconst	directors	writers
0	tt0285252	nm0899854	nm0899854
1	tt0438973	NaN	nm0175726,nm1802864

[13]:
```
# Exploring the data
# Title akas dataset

imdb_title_akas = pd.read_csv("/Users/r/Desktop/MS-Movie-Project/zippedData/
 ↪imdb.title.akas.csv")
print(imdb_title_akas.shape)
imdb_title_akas.head(2)
```

```
(331703, 8)
```

[13]:

	title_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик парк	BG	bg	NaN	NaN	0.0
1	tt0369610	11	Jurashikku warudo	JP	NaN	imdbDisplay	NaN	0.0

1.5 Data Conclusion

- bom_movie_gross_csv - contains name of the title, studio where the movie was produced, year of production and domestic and foreign gross revenue
- imdb_name_basics_csv - Contains details of writers and directors (birth, death year, profession, names and title they are associated with)
- imdb_title_akas_csv - Contains more details of a title
- imdb_title_basics_csv - Contains information on various title. Title id, original and primary title, year, runtime minutes and genre of the movie
- imdb_title_crew_csv - Contains titles and the directors and writers associated with the title
- imdb_title_principals_csv - Contains identities of writers, directors or writer and titles they are associated with
- imdb_title_ratings_csv - Contains movie title identity key, averating rating and number of votes for that specific title
- rt_movie_info_csv - Contains info on movies, director, writer, studio, date and synopsis
- rt_reviews_csv - Contains movie reviews, rating in scale of 5, reviewer name and date of the review
- tmdb_movies_csv - Contains details of a movie like ratiing, release date, title and genre

- tn_movie_budgets_csv - Contains budget of a movie, their domestic and foreign gross revenue and the date the movie was realease date

1.6 Data Preparation

In this section we will perform the following data preparation **Merging the datasets we need * Data understanding: Checking for nulls and empty values * Data cleaning: Replacing the nulls and missing values** *

From the above data preview we are going to use the following datasets for this project

- bom_movie_gross_csv
- imdb_title_basics_csv
- tn_movie_budgets_csv
- imdb_title_ratings_csv

[14]:
```
# Lets merge the top 3 datasets we will be using
# Let merge title basics and title ratings data

#Preview the size of the 2 datasets before merging
print(imdb_title_ratings.shape)
print(imdb_title_basics.shape)

# Merge the 2 data sets using an inner join
# We only want information of the movies whose average rating and numvotes are␣
 ↪available
title_rating = pd.merge(imdb_title_basics, imdb_title_ratings, on='tconst',␣
 ↪how="outer")

# Show the resulting and explore the new datasets by checking the nulls
print(title_rating.shape)
title_rating.head(10)
title_rating.isna().mean()*100
```

```
(73856, 3)
(146144, 6)
(146144, 8)
```

[14]:
```
tconst              0.000000
primary_title       0.000684
original_title      0.015054
start_year          0.000000
runtime_minutes    21.717621
genres              3.700460
averagerating      49.463543
numvotes           49.463543
dtype: float64
```

[15]:
```
# Merging the new dataset with the third dataset (budget dataset)
# We do an outer join
title_rating_budget = pd.merge(title_rating, tn_movie_budgets, \
                      left_on='original_title', right_on='movie', how="outer")

# Checking the size of the new dataset
title_rating_budget.shape
```

[15]: (149775, 14)

[16]:
```
title_rating_budget.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149775 entries, 0 to 149774
Data columns (total 14 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   tconst             146323 non-null  object
 1   primary_title      146322 non-null  object
 2   original_title     146301 non-null  object
 3   start_year         146323 non-null  float64
 4   runtime_minutes    114554 non-null  float64
 5   genres             140911 non-null  object
 6   averagerating      73959 non-null   float64
 7   numvotes           73959 non-null   float64
 8   id                 6989 non-null    float64
 9   release_date       6989 non-null    object
 10  movie              6989 non-null    object
 11  production_budget  6989 non-null    object
 12  domestic_gross     6989 non-null    object
 13  worldwide_gross    6989 non-null    object
dtypes: float64(5), object(9)
memory usage: 16.0+ MB
```

[17]:
```
# Lets explore the dataset by getting the mean, median, count, percentiles etc.
title_rating_budget.describe()
```

[17]:

	start_year	runtime_minutes	averagerating	numvotes	\
count	146323.000000	114554.000000	73959.000000	7.395900e+04	
mean	2014.621884	86.190696	6.332517	3.620574e+03	
std	2.733620	166.257634	1.474505	3.085529e+04	
min	2010.000000	1.000000	1.000000	5.000000e+00	
25%	2012.000000	70.000000	5.500000	1.400000e+01	
50%	2015.000000	87.000000	6.500000	4.900000e+01	
75%	2017.000000	99.000000	7.400000	2.830000e+02	
max	2115.000000	51420.000000	10.000000	1.841066e+06	

```
                id
count  6989.000000
mean     50.279296
std      28.757684
min       1.000000
25%      26.000000
50%      50.000000
75%      75.000000
max     100.000000
```

[18]:
```
# Checking for duplicates
# No duplicates found
title_rating_budget.duplicated().value_counts()
```

[18]:
```
False    149775
Name: count, dtype: int64
```

[19]:
```
# Checking for missing values
title_rating_budget.isnull().sum()
```

[19]:
```
tconst                3452
primary_title         3453
original_title        3474
start_year            3452
runtime_minutes      35221
genres                8864
averagerating        75816
numvotes             75816
id                  142786
release_date        142786
movie               142786
production_budget   142786
domestic_gross      142786
worldwide_gross     142786
dtype: int64
```

[20]:
```
# getting the propotion of the missing data as a percentage
# Columns from the title_rating_budget have alot of missing data, so we will␣
 ↪just drop the columns
title_rating_budget.isna().mean()*100
```

[20]:
```
tconst               2.304791
primary_title        2.305458
original_title       2.319479
start_year           2.304791
runtime_minutes     23.515941
genres               5.918211
```

```
averagerating        50.619930
numvotes             50.619930
id                   95.333667
release_date         95.333667
movie                95.333667
production_budget    95.333667
domestic_gross       95.333667
worldwide_gross      95.333667
dtype: float64
```

1.7 Data Cleaning

1.7.1 In this section, we will go data cleaning

- Removing outliers
- Replacing null values
- Changing datatypes
- Creating new columns to hold new data

If we drop the N/A we will lose more than 50% of our dataset

So we keep the dataset and try to replace the null values with mean and median in the following section

- We will use median where we have several missing values
- We will use mean where we have less than 50% missing values
- We will use unknown for columns containing strings
- We will use 0 for columns containing dates and currencies

[21]:
```
# Creating a new pandaframe
df = title_rating_budget
```

[22]:
```
# Trying to drop all empty rows
df.dropna(how="all", axis=1).shape
```

[22]:
```
(149775, 14)
```

[23]:
```
# Changing start-year to year (datetime)
# Changing from float to datetime

df['start_year'] = pd.to_datetime(df['start_year'], format='%Y')

# Extracting only year from the datetime format
df['start_year'] = pd.to_datetime(df['start_year']).dt.strftime('%Y')

# Previewing the data
df.tail()
```

[23]:
```
        tconst primary_title original_title start_year  runtime_minutes genres  \
149770     NaN           NaN            NaN        NaN              NaN    NaN
149771     NaN           NaN            NaN        NaN              NaN    NaN
149772     NaN           NaN            NaN        NaN              NaN    NaN
149773     NaN           NaN            NaN        NaN              NaN    NaN
149774     NaN           NaN            NaN        NaN              NaN    NaN

        averagerating  numvotes    id  release_date  \
149770            NaN       NaN  76.0  May 26, 2006
149771            NaN       NaN  77.0  Dec 31, 2004
149772            NaN       NaN  79.0   Apr 2, 1999
149773            NaN       NaN  80.0  Jul 13, 2005
149774            NaN       NaN  82.0   Aug 5, 2005

                             movie production_budget domestic_gross  \
149770                      Cavite            $7,000        $70,071
149771               The Mongol King            $7,000           $900
149772                     Following            $6,000        $48,482
149773 Return to the Land of Wonders            $5,000         $1,338
149774             My Date With Drew            $1,100       $181,041

       worldwide_gross
149770         $71,644
149771            $900
149772        $240,495
149773          $1,338
149774        $181,041
```

[24]:
```
# Creating release year to store the date the movie was released
# Extracting only year from the datetime format
df['release_year'] = pd.to_datetime(df['release_date']).dt.strftime('%Y')

# previewing the data
df.tail()
```

[24]:
```
        tconst primary_title original_title start_year  runtime_minutes genres  \
149770     NaN           NaN            NaN        NaN              NaN    NaN
149771     NaN           NaN            NaN        NaN              NaN    NaN
149772     NaN           NaN            NaN        NaN              NaN    NaN
149773     NaN           NaN            NaN        NaN              NaN    NaN
149774     NaN           NaN            NaN        NaN              NaN    NaN

        averagerating  numvotes    id  release_date  \
149770            NaN       NaN  76.0  May 26, 2006
149771            NaN       NaN  77.0  Dec 31, 2004
149772            NaN       NaN  79.0   Apr 2, 1999
149773            NaN       NaN  80.0  Jul 13, 2005
```

```
149774           NaN       NaN  82.0   Aug 5, 2005

                             movie production_budget domestic_gross  \
149770                       Cavite            $7,000        $70,071
149771             The Mongol King            $7,000           $900
149772                    Following            $6,000        $48,482
149773  Return to the Land of Wonders            $5,000         $1,338
149774            My Date With Drew            $1,100       $181,041

       worldwide_gross release_year
149770         $71,644         2006
149771            $900         2004
149772        $240,495         1999
149773          $1,338         2005
149774        $181,041         2005
```

[25]:
```
# Previewing columns we need
df.columns[-4:-1]
```

[25]:
```
Index(['production_budget', 'domestic_gross', 'worldwide_gross'],
dtype='object')
```

[26]:
```
# changing object to float on columns containing currency
# replace $ in the last 3 columns and change the values to float
df[df.columns[-4:-1:]] = df[df.columns[-4:-1:]].replace('[\$,]', '',␣
 ↪regex=True).astype(float)
df.head()
```

[26]:
```
      tconst                    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  averagerating  numvotes  \
0       2013            175.0    Action,Crime,Drama            7.0      77.0
1       2019            114.0       Biography,Drama            7.2      43.0
2       2018            122.0                 Drama            6.9    4517.0
3       2018              NaN          Comedy,Drama            6.1      13.0
4       2017             80.0  Comedy,Drama,Fantasy            6.5     119.0

   id release_date movie  production_budget  domestic_gross  worldwide_gross  \
0 NaN          NaN   NaN                NaN             NaN              NaN
1 NaN          NaN   NaN                NaN             NaN              NaN
2 NaN          NaN   NaN                NaN             NaN              NaN
3 NaN          NaN   NaN                NaN             NaN              NaN
```

```
4 NaN          NaN   NaN                NaN             NaN              NaN
```

```
  release_year
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
```

[27]:
```
# Calculating the means and medians that we will use to replacing the missing␣
 ↪data

mean_rating = df["averagerating"].mean()
mean_votes = df["numvotes"].mean()
mean_time = df["runtime_minutes"].mean()
mean_budget = df["production_budget"].median()
mean_domestic = df["domestic_gross"].median()
mean_worldwide = df["worldwide_gross"].median()
```

[28]:
```
# Replace the missing ratings with the mean and median
# We use unknown to replace columns containing strings and 0 for columns␣
 ↪containing objects
df["averagerating"].fillna(mean_rating, inplace=True)
df["numvotes"].fillna(mean_votes, inplace=True)
df["runtime_minutes"].fillna(mean_time, inplace=True)
df["tconst"].fillna('Unknown', inplace=True)
df["primary_title"].fillna('Unknown', inplace=True)
df["original_title"].fillna('Unknown', inplace=True)
df["start_year"].fillna(0, inplace=True)
df["genres"].fillna('Unknown', inplace=True)
df["movie"].fillna('Unknown', inplace=True)
df["release_date"].fillna('Unknown', inplace=True)
df["id"].fillna(-1, inplace=True)
df["production_budget"].fillna(0, inplace=True)
df["domestic_gross"].fillna(0, inplace=True)
df["worldwide_gross"].fillna(0, inplace=True)
df["release_year"].fillna(0, inplace=True)
```

[29]:
```
# Checking if we have gotten rid of all the nulls
df.isna().mean()*100
```

[29]:
```
tconst              0.0
primary_title       0.0
original_title      0.0
start_year          0.0
runtime_minutes     0.0
genres              0.0
```

```
averagerating        0.0
numvotes             0.0
id                   0.0
release_date         0.0
movie                0.0
production_budget    0.0
domestic_gross       0.0
worldwide_gross      0.0
release_year         0.0
dtype: float64
```

[30]:
```
# Exploring the data to check datatypes and null columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149775 entries, 0 to 149774
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   tconst             149775 non-null  object
 1   primary_title      149775 non-null  object
 2   original_title     149775 non-null  object
 3   start_year         149775 non-null  object
 4   runtime_minutes    149775 non-null  float64
 5   genres             149775 non-null  object
 6   averagerating      149775 non-null  float64
 7   numvotes           149775 non-null  float64
 8   id                 149775 non-null  float64
 9   release_date       149775 non-null  object
 10  movie              149775 non-null  object
 11  production_budget  149775 non-null  float64
 12  domestic_gross     149775 non-null  float64
 13  worldwide_gross    149775 non-null  float64
 14  release_year       149775 non-null  object
dtypes: float64(7), object(8)
memory usage: 17.1+ MB
```

[31]:
```
# Data preview
df.head()
```

[31]:

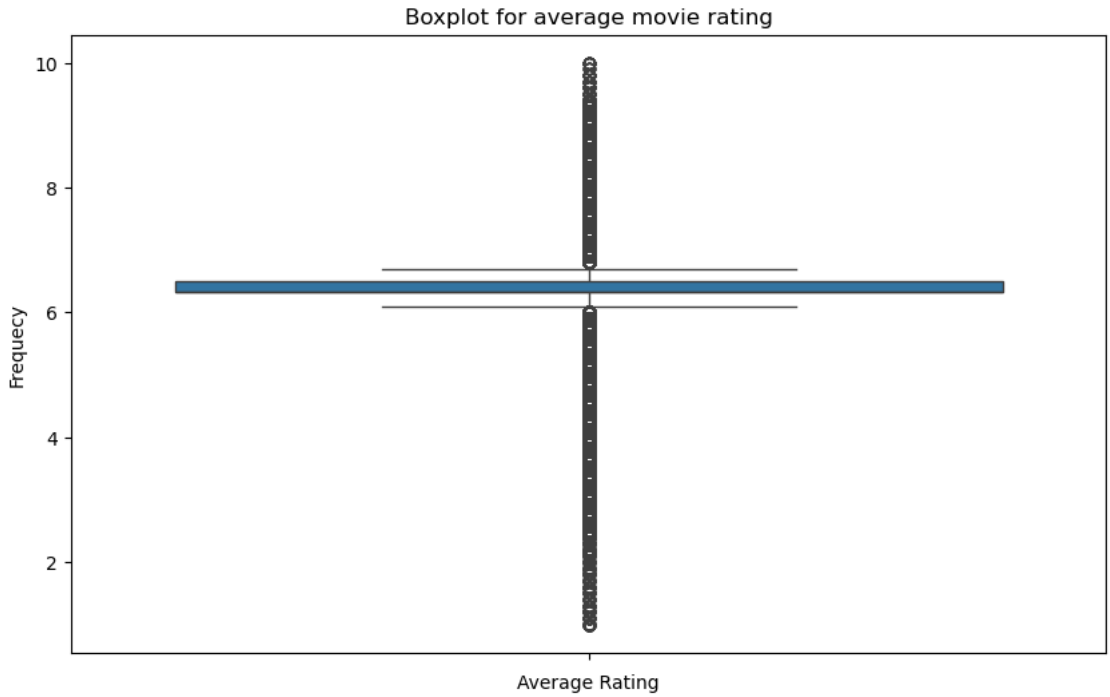
	tconst	primary_title	original_title	\
0	tt0063540	Sunghursh	Sunghursh	
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	

```
   start_year  runtime_minutes                genres  averagerating  numvotes  \
0        2013       175.000000    Action,Crime,Drama            7.0      77.0
1        2019       114.000000       Biography,Drama            7.2      43.0
2        2018       122.000000                 Drama            6.9    4517.0
3        2018        86.190696          Comedy,Drama            6.1      13.0
4        2017        80.000000  Comedy,Drama,Fantasy            6.5     119.0

     id release_date    movie  production_budget  domestic_gross  \
0 -1.0      Unknown  Unknown                0.0             0.0
1 -1.0      Unknown  Unknown                0.0             0.0
2 -1.0      Unknown  Unknown                0.0             0.0
3 -1.0      Unknown  Unknown                0.0             0.0
4 -1.0      Unknown  Unknown                0.0             0.0

   worldwide_gross release_year
0              0.0            0
1              0.0            0
2              0.0            0
3              0.0            0
4              0.0            0
```

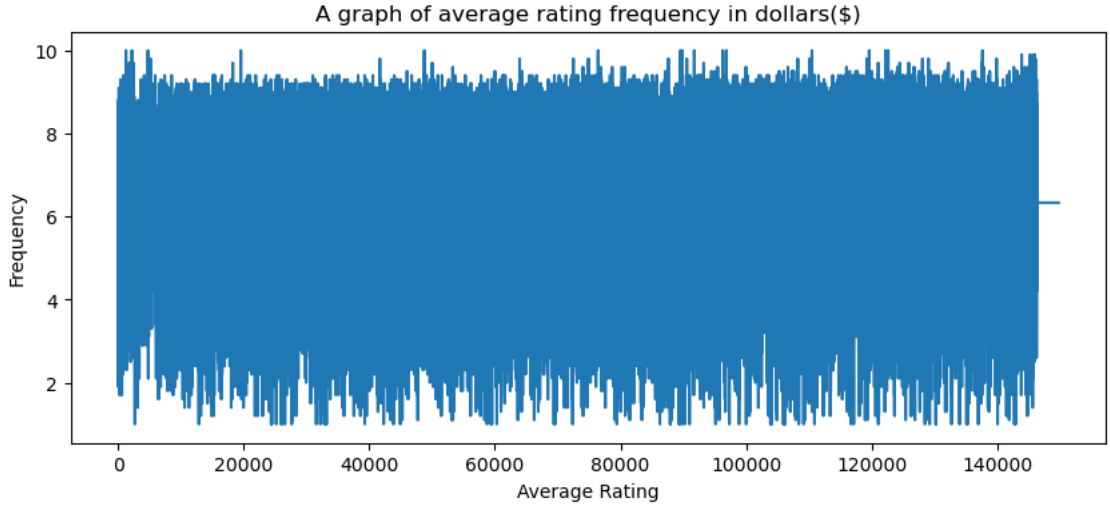
1.8 Outliers

1.8.1 In this section we explore and remove outliers

[32]:
```
# Checking for outliers using boxplot for average rating column
import seaborn as sns

plt.figure(figsize = (10,6))
plt.title('Boxplot for average movie rating')
plt.ylabel('Frequecy')
plt.xlabel('Average Rating')
sns.boxplot(data=df['averagerating']);
```



[33]:
```
# Checking for outliers using lineplot

plt.figure(figsize=(10,4))
df['averagerating'].plot()
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.title('A graph of average rating frequency in dollars($)');
```



[34]:
```
# Chose the maximum quantile to fill in the outliers for the rating columns
max_quantile=df["averagerating"].quantile(0.995)
max_quantile
```

[34]:
```
9.1
```

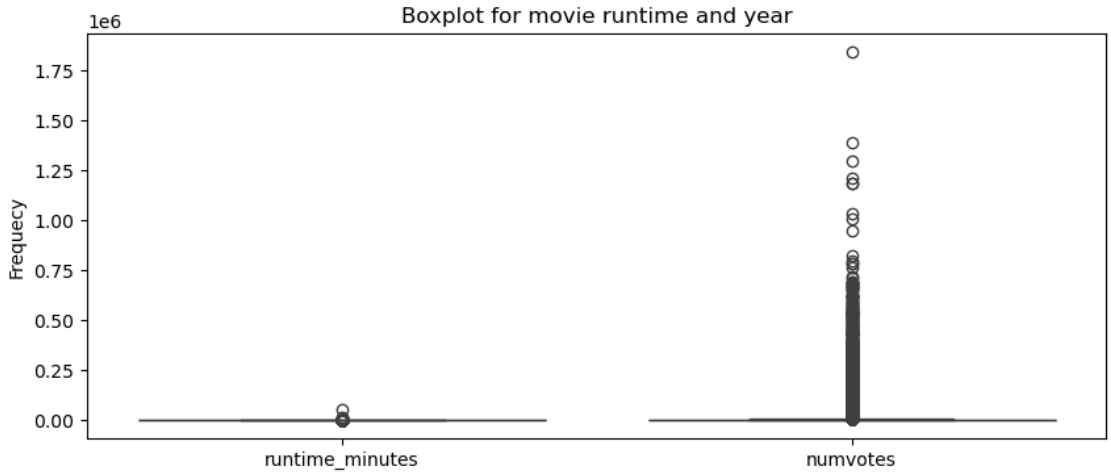
[35]:
```
# IMDB rating are from range 10 (1 to 10).
# Any rating above 10 is an outlier or wrong
# We check if we have this case

df[df["averagerating"]>10]
```

[35]:
```
Empty DataFrame
Columns: [tconst, primary_title, original_title, start_year, runtime_minutes,
genres, averagerating, numvotes, id, release_date, movie, production_budget,
domestic_gross, worldwide_gross, release_year]
Index: []
```

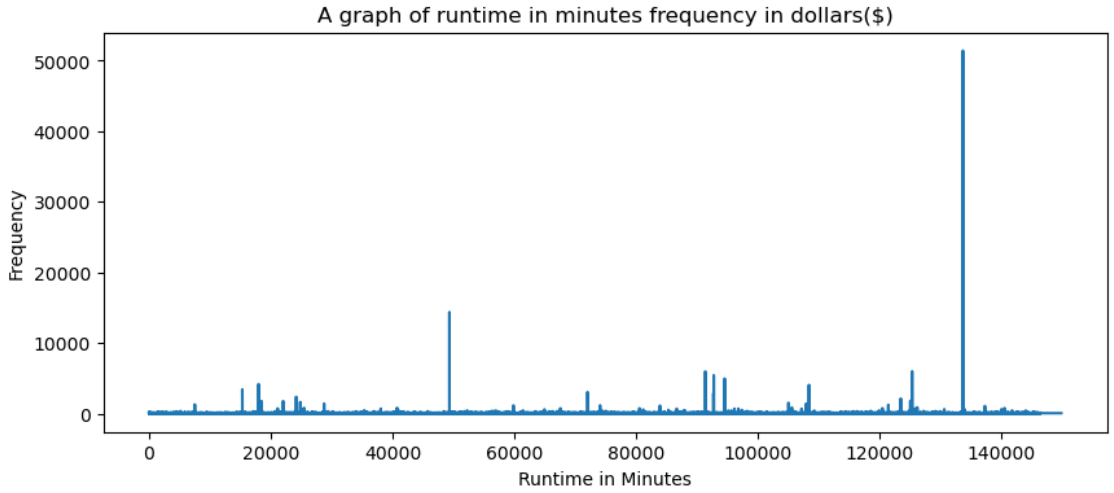
[36]:
```
# IMDB rating are from range 10 (1 to 10).
# Any rating less than 1 is an outlier or wrong
# We check if we have this case

df[df["averagerating"]< 1]
```

[36]:
```
Empty DataFrame
Columns: [tconst, primary_title, original_title, start_year, runtime_minutes,
genres, averagerating, numvotes, id, release_date, movie, production_budget,
domestic_gross, worldwide_gross, release_year]
Index: []
```

[37]:
```
# Outlier for runtimes and number of votes

import seaborn as sns

plt.figure(figsize = (10,4))
columns = ['runtime_minutes','numvotes']
plt.title('Boxplot for movie runtime and year')
plt.ylabel('Frequecy')
sns.boxplot(data=df[columns]);
```

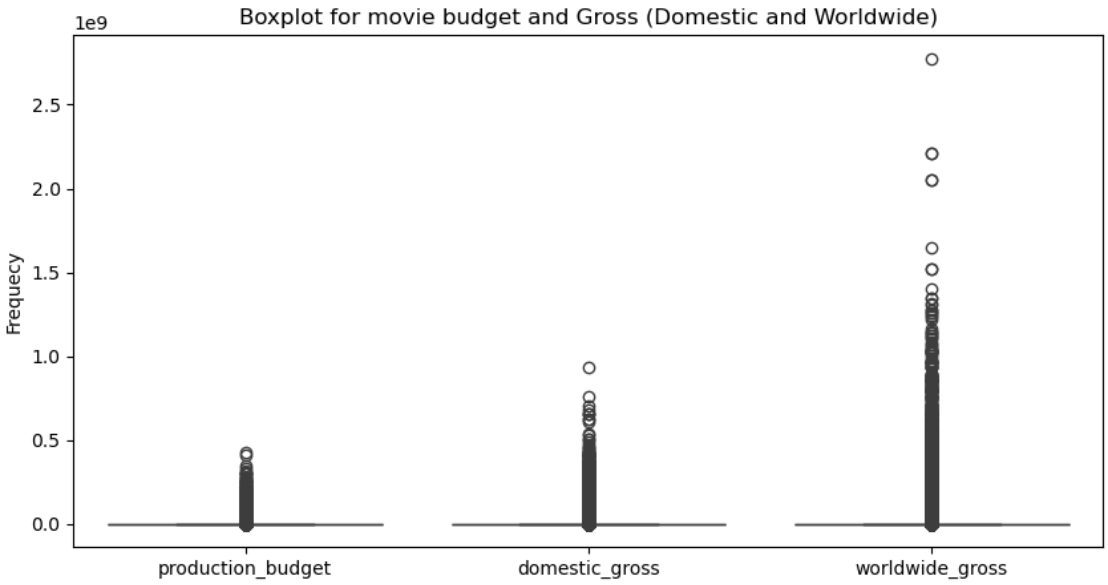


[38]:
```
# Detecting outliers using lineplot

plt.figure(figsize=(10,4))
df['runtime_minutes'].plot()
plt.xlabel('Runtime in Minutes')
plt.ylabel('Frequency')
plt.title('A graph of runtime in minutes frequency in dollars($)');
```

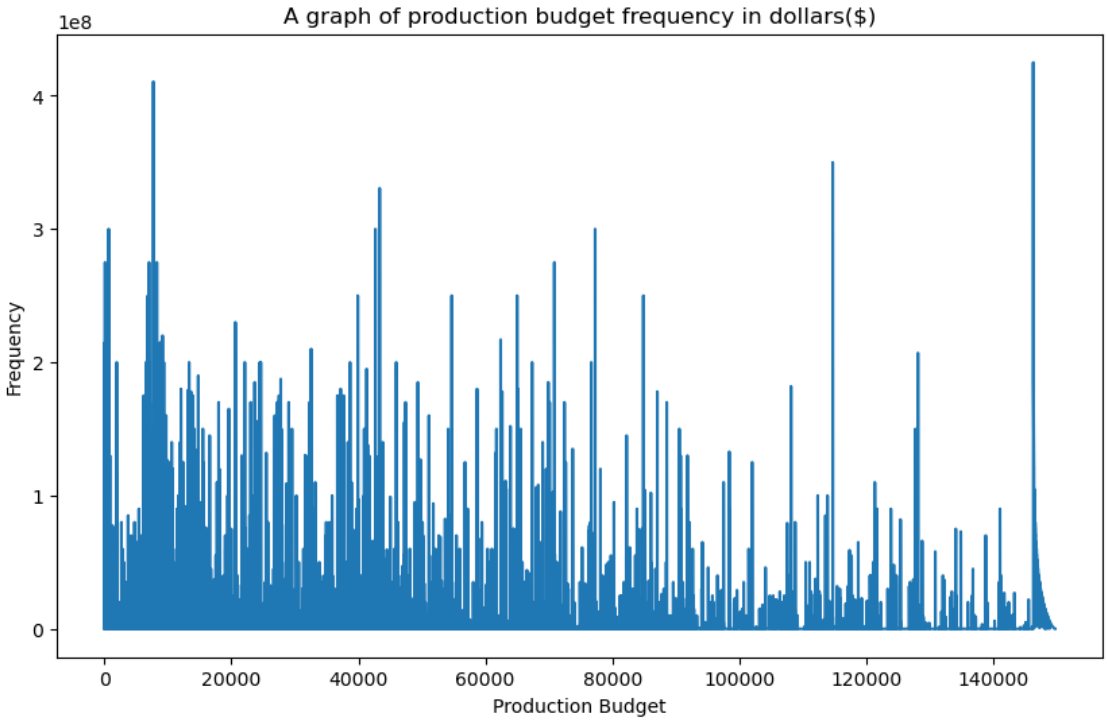


[39]:
```
# Outlier for bugdet, domestic and worldwide gross

import seaborn as sns

plt.figure(figsize = (10,5))
```

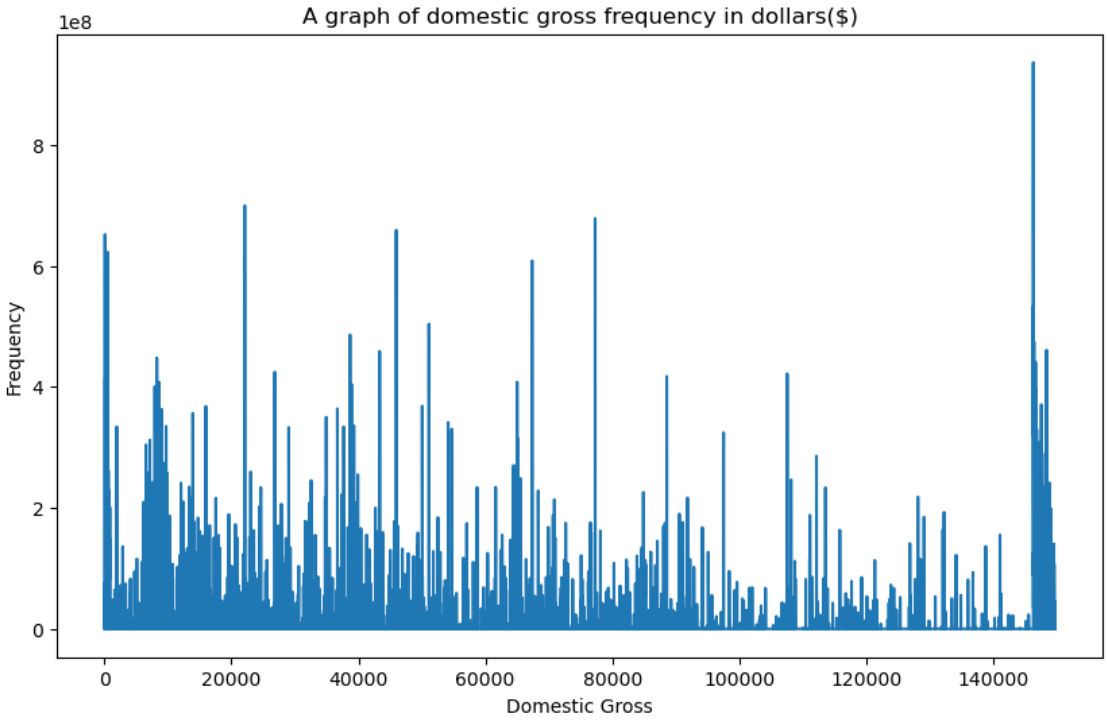
```
columns = ['production_budget','domestic_gross', 'worldwide_gross']
plt.title('Boxplot for movie budget and Gross (Domestic and Worldwide)')
plt.ylabel('Frequecy')
sns.boxplot(data=df[columns]);
```



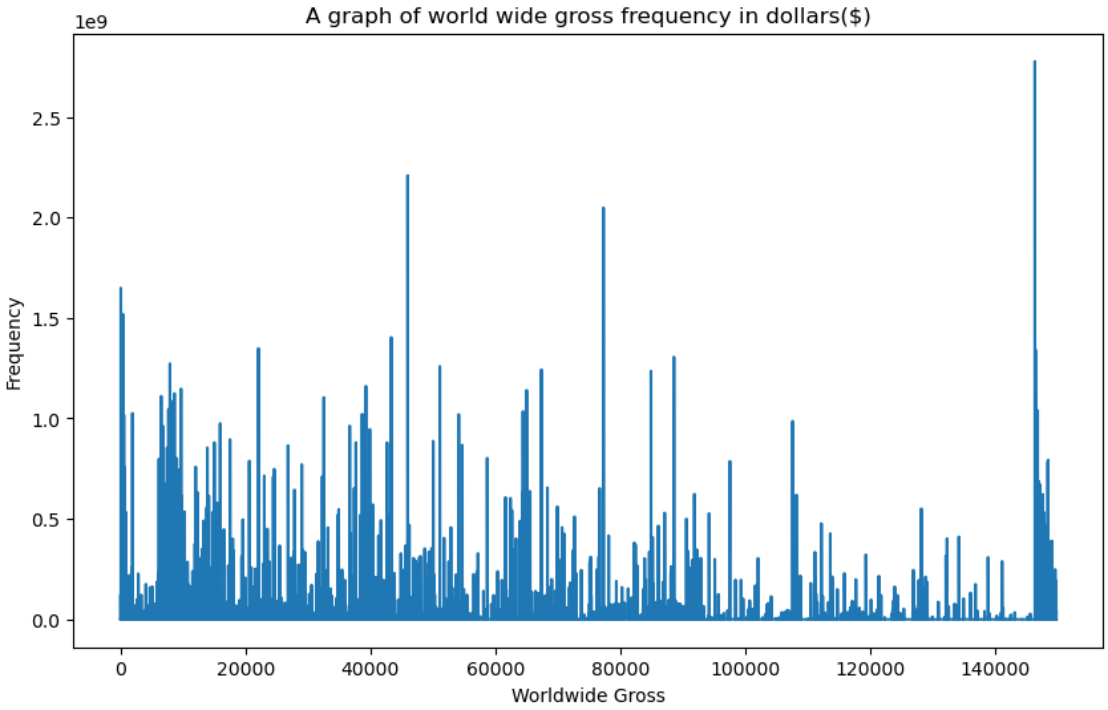
[40]:
```
# Checking for production budget outliers using a lineplot

plt.figure(figsize=(10,6))
df['production_budget'].plot()
plt.xlabel('Production Budget')
plt.ylabel('Frequency')
plt.title('A graph of production budget frequency in dollars($)');
```



[41]:
```
# Checking for domestic gross outliers using a lineplot

plt.figure(figsize=(10,6))
df['domestic_gross'].plot()
plt.xlabel('Domestic Gross')
plt.ylabel('Frequency')
plt.title('A graph of domestic gross frequency in dollars($)');
```



[42]:
```
# Checking for world wide gross outliers using a lineplot

plt.figure(figsize=(10,6))
df['worldwide_gross'].plot()
plt.xlabel('Worldwide Gross')
plt.ylabel('Frequency')
plt.title('A graph of world wide gross frequency in dollars($)');
```



Removing Outliers

Conclusion For Rating: Ratings on websites such as IMDB range from 1 to 10. There are no rating outside this range. So we wont be removing any outliers from column averagerating

Runtime Minutes: We have outliers and we will removing them using IQR method

[43]:
```
df2 = df
df2.shape
```

[43]: (149775, 15)

[44]:
```
df2.head()
```

[44]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	175.000000	Action,Crime,Drama	7.0	77.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.000000	Biography,Drama	7.2	43.0
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					

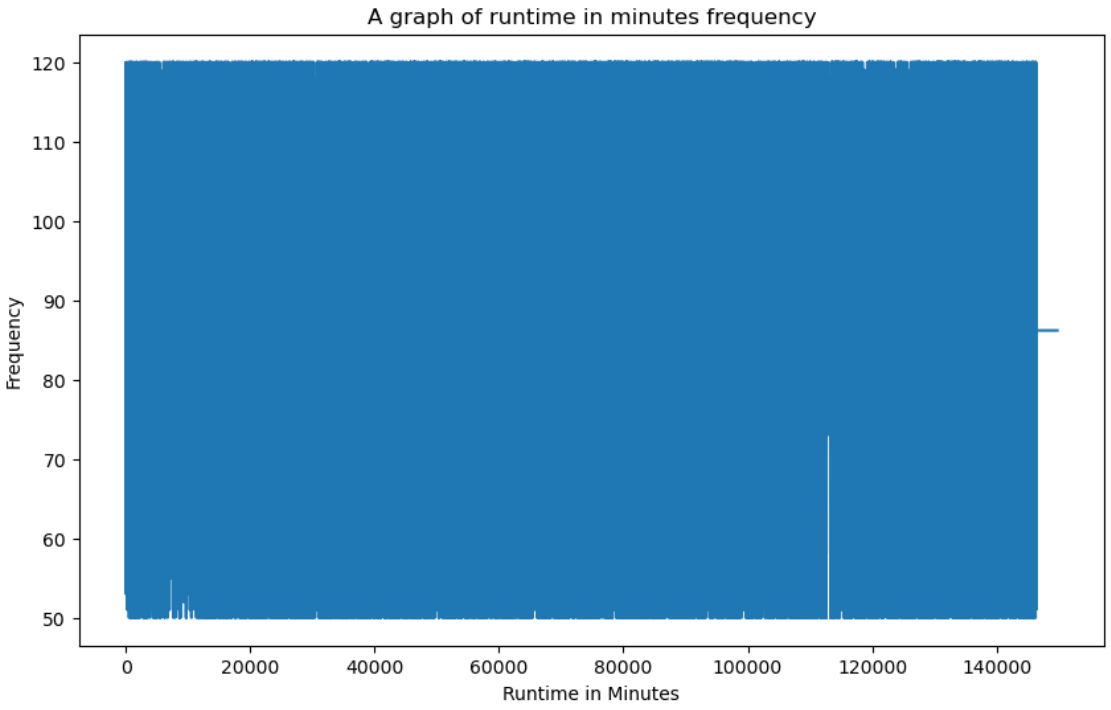
```
2       2018       122.000000                  Drama            6.9    4517.0
3       2018        86.190696           Comedy,Drama            6.1      13.0
4       2017        80.000000  Comedy,Drama,Fantasy             6.5     119.0

    id release_date    movie  production_budget  domestic_gross  \
0 -1.0      Unknown  Unknown                0.0             0.0
1 -1.0      Unknown  Unknown                0.0             0.0
2 -1.0      Unknown  Unknown                0.0             0.0
3 -1.0      Unknown  Unknown                0.0             0.0
4 -1.0      Unknown  Unknown                0.0             0.0

   worldwide_gross release_year
0              0.0            0
1              0.0            0
2              0.0            0
3              0.0            0
4              0.0            0
```

[45]:
```python
# Removing outliers for runtime_minutes using interquartile range
# Calculating the Upper and Lower Quantile

Q1 = df2['runtime_minutes'].quantile(0.25)
Q3 = df2['runtime_minutes'].quantile(0.75)

# Calculate the IQR
IQR = Q3 - Q1

# Calculate the upper and lower limits
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

# Create arrays of Boolean values indicating the outlier rows
upper_array = np.where(df2['runtime_minutes'] >= upper)[0]
lower_array = np.where(df2['runtime_minutes'] <= lower)[0]

# Removing the outliers
df2.drop(index=upper_array, inplace=True)
df2.drop(index=lower_array, inplace=True)

# Print the new shape of the DataFrame
print("New Shape: ", df2.shape)
```

```
New Shape:  (131256, 15)
```

[46]:
```python
# Checking if we have removed the outliers successfully

plt.figure(figsize=(10,6))
```

```
df2['runtime_minutes'].plot()
plt.xlabel('Runtime in Minutes')
plt.ylabel('Frequency')
plt.title('A graph of runtime in minutes frequency');
```



1.9 Data Modeling

1.10 Here we create different visuals that will help us answer the following questions

- Was the rating we have influenced by the nuumber of runtime in minutes?
- Is the domestic gross correlated with rating - Does rating affect domestic revenue?
- Does rating affect worlwide revenue?
- Does budget affect the movie rating?
- How is movie budget connected to movie revenue?
- Who are top possible Microsoft competitors?
- What movies genre perform well in terms of revenue?

[47]:
```
# Generating descriptive statistics for numerical columns (mean, median, std,␣
↪etc.).
df2.describe()
```

[47]:
```
       runtime_minutes  averagerating       numvotes             id  \
count    131256.000000  131256.000000  131256.000000  131256.000000
```

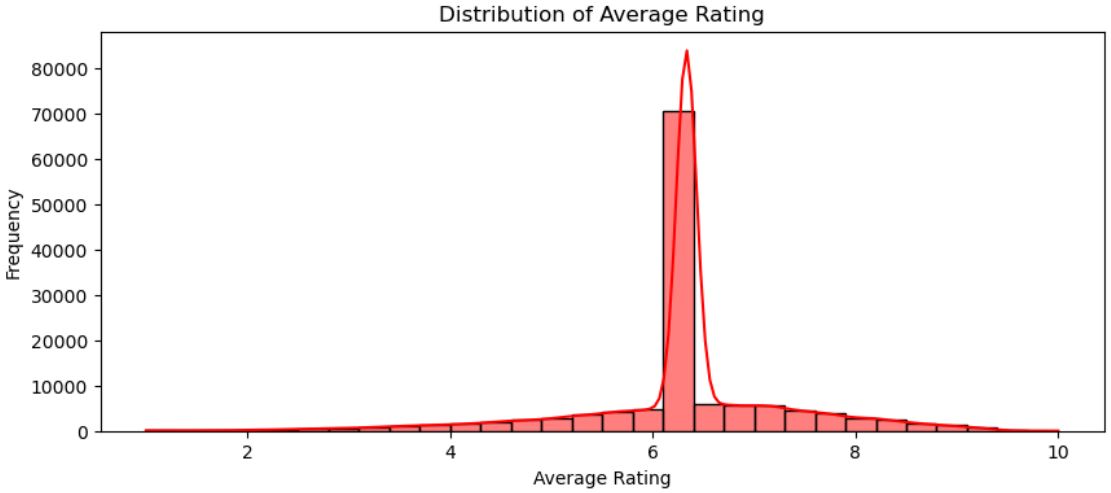
```
mean          85.284478       6.317249    3106.572124       1.493913
std           14.732725       1.051002   15071.831497      12.732728
min           50.000000       1.000000       5.000000      -1.000000
25%           80.000000       6.332517      45.000000      -1.000000
50%           86.190696       6.332517    3620.574264      -1.000000
75%           92.000000       6.500000    3620.574264      -1.000000
max          120.000000      10.000000  820847.000000     100.000000

       production_budget  domestic_gross  worldwide_gross
count       1.312560e+05    1.312560e+05     1.312560e+05
mean        1.401270e+06    1.828656e+06     3.885454e+06
std         1.029327e+07    1.589206e+07     3.830669e+07
min         0.000000e+00    0.000000e+00     0.000000e+00
25%         0.000000e+00    0.000000e+00     0.000000e+00
50%         0.000000e+00    0.000000e+00     0.000000e+00
75%         0.000000e+00    0.000000e+00     0.000000e+00
max         4.250000e+08    9.366622e+08     2.776345e+09
```

[48]:
```
# Generating summary of the data.
df2.info()
```

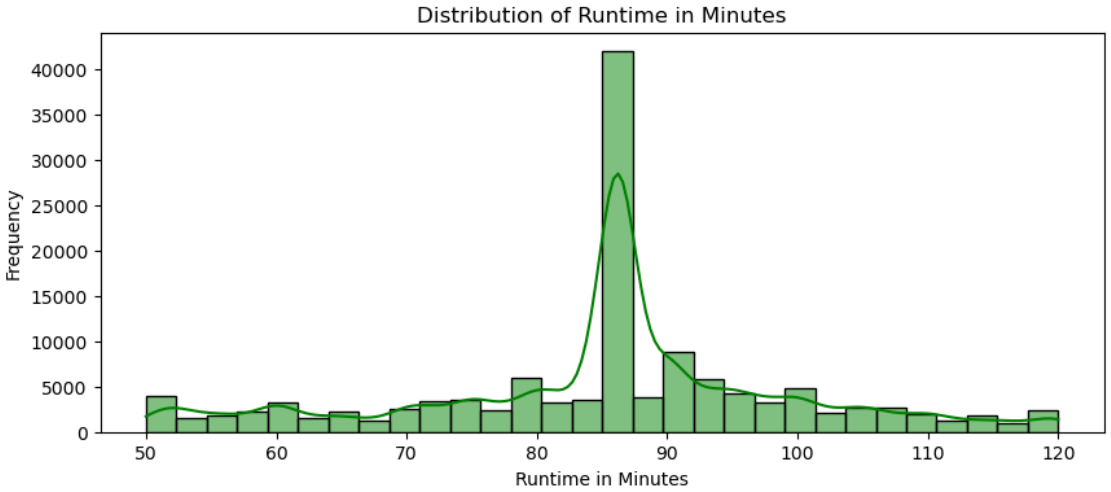
```
<class 'pandas.core.frame.DataFrame'>
Index: 131256 entries, 1 to 149774
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   tconst             131256 non-null  object
 1   primary_title      131256 non-null  object
 2   original_title     131256 non-null  object
 3   start_year         131256 non-null  object
 4   runtime_minutes    131256 non-null  float64
 5   genres             131256 non-null  object
 6   averagerating      131256 non-null  float64
 7   numvotes           131256 non-null  float64
 8   id                 131256 non-null  float64
 9   release_date       131256 non-null  object
 10  movie              131256 non-null  object
 11  production_budget  131256 non-null  float64
 12  domestic_gross     131256 non-null  float64
 13  worldwide_gross    131256 non-null  float64
 14  release_year       131256 non-null  object
dtypes: float64(7), object(8)
memory usage: 16.0+ MB
```

[49]:
```
# Plotting movie ratings distribution

plt.figure(figsize=(10,4))
```

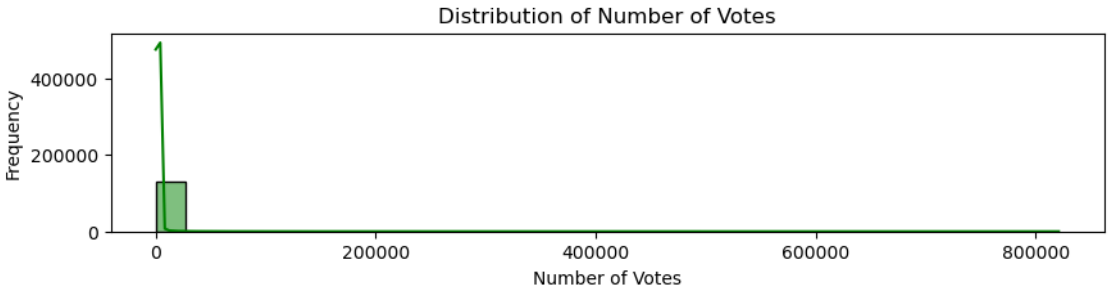
```
sns.histplot(df2['averagerating'], kde=True, bins=30, color='red')
plt.title('Distribution of Average Rating')
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.show()
```



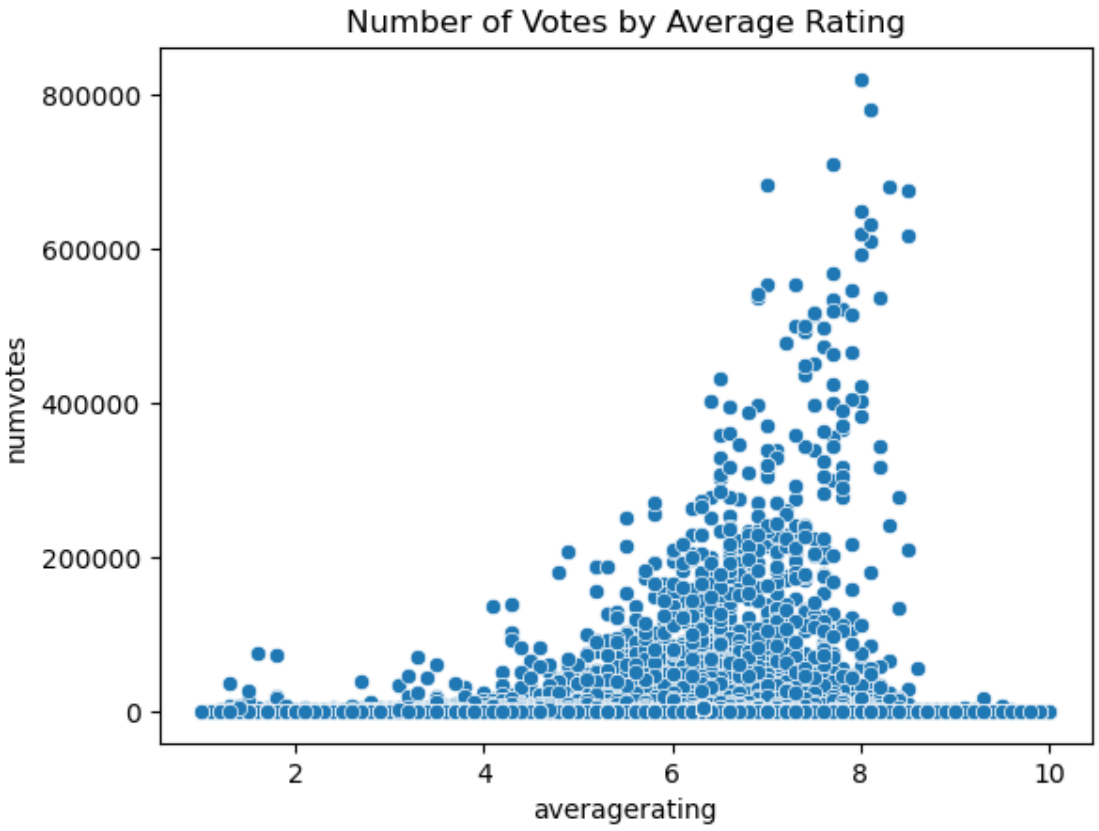
[50]:
```
# Plotting runtime_minutes distribution

plt.figure(figsize=(10, 4))
sns.histplot(df2['runtime_minutes'], kde=True, bins=30, color='green')
plt.title('Distribution of Runtime in Minutes')
plt.xlabel('Runtime in Minutes')
plt.ylabel('Frequency')
plt.show()
```



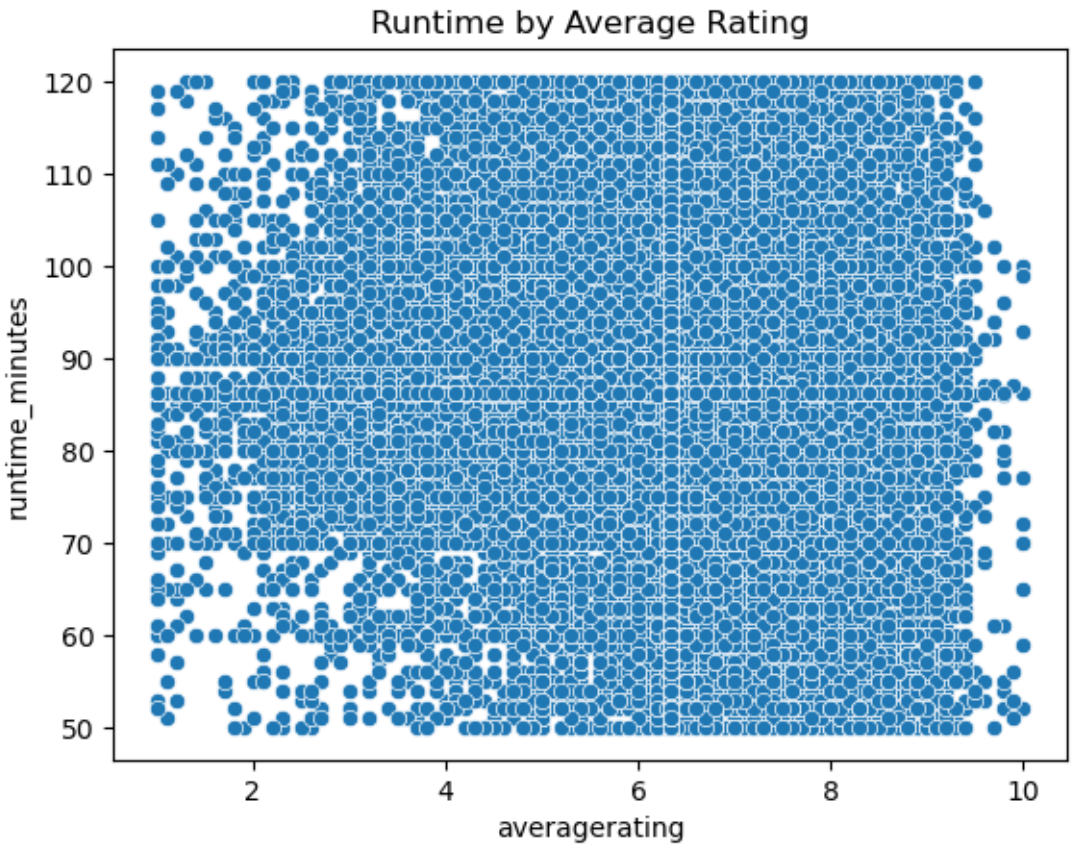
[51]:
```
# Plotting Number of Votes distribution

plt.figure(figsize=(10, 2))
sns.histplot(df2['numvotes'], kde=True, bins=30, color='green')
plt.title('Distribution of Number of Votes')
plt.xlabel('Number of Votes')
plt.ylabel('Frequency')
plt.show()
```



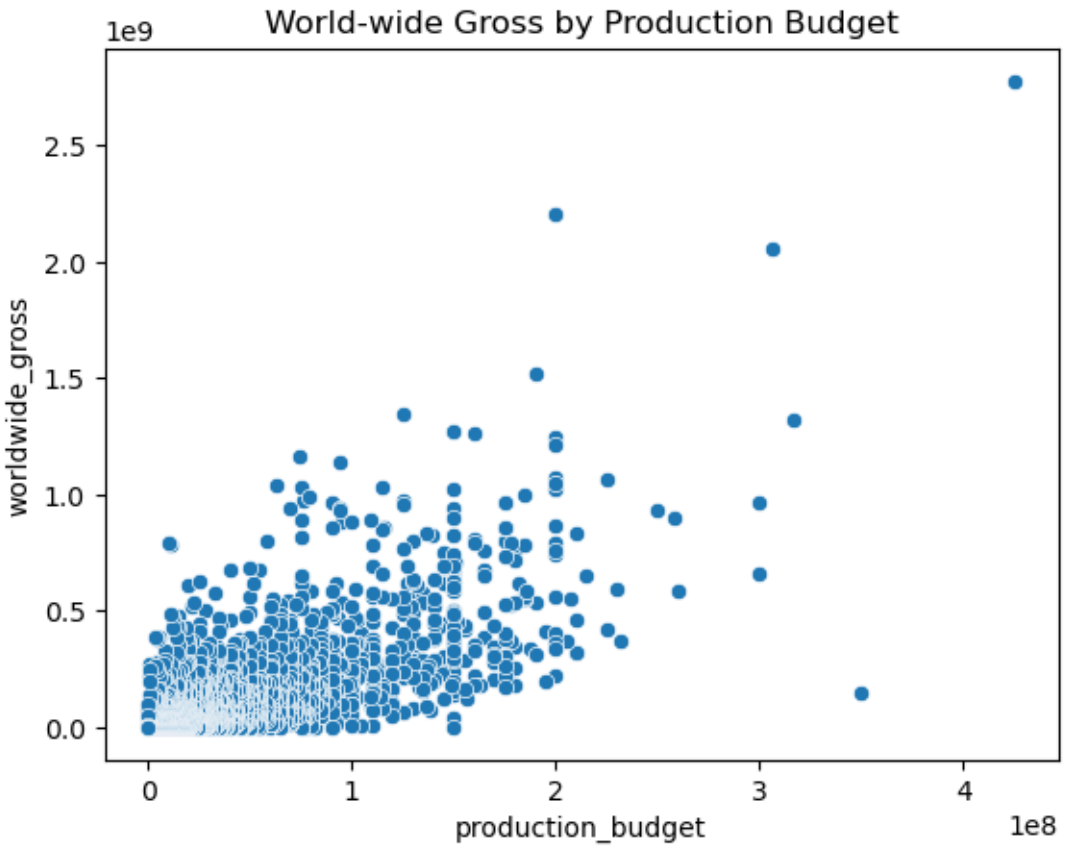
[52]:
```
# We use a scatterplot to show the relationship between average rating and␣
 ↪Runtime in Minutes
# There is no colleration between the number of votes and the average rating of␣
 ↪a movie

sns.scatterplot(x="averagerating",
                y="numvotes",
                data=df2).set(title='Number of Votes by Average Rating');
```



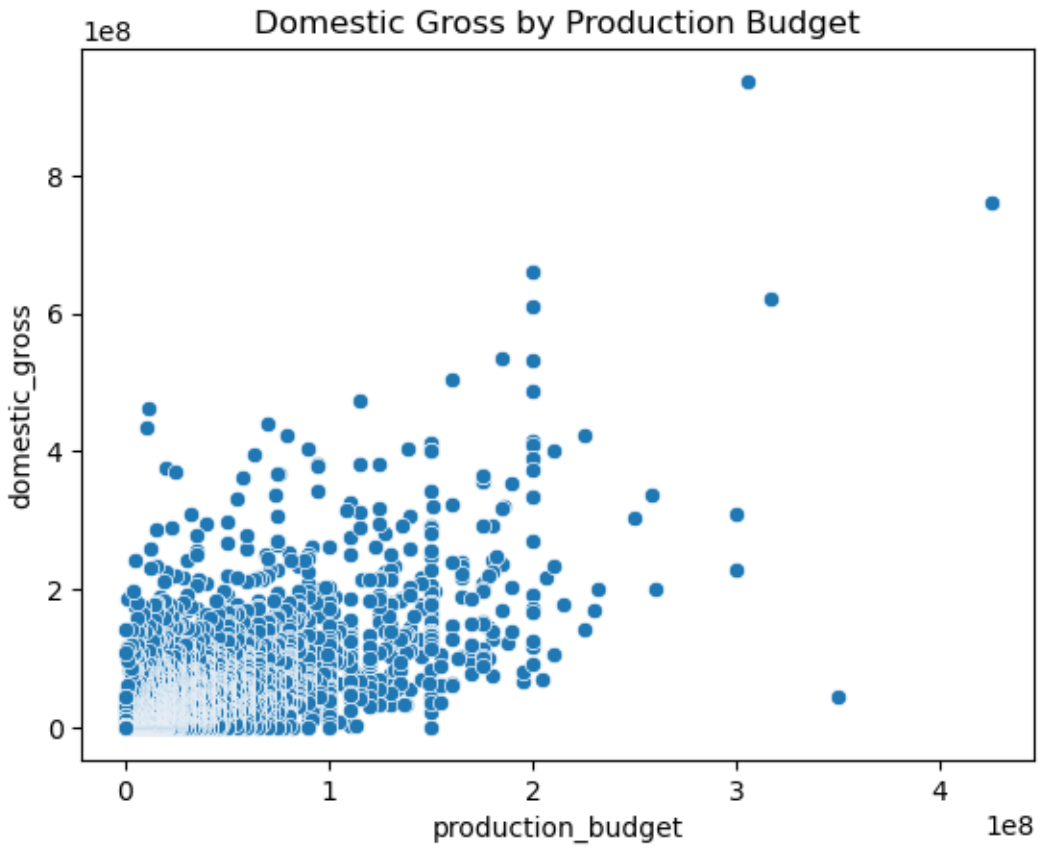
[53]:
```
# We use a scatterplot to show the relationship between average rating and␣
 ↪number of votes
# No correlation between runtime in minutes and the average rating of the movie

sns.scatterplot(x="averagerating",
                y="runtime_minutes",
                data=df2).set(title='Runtime by Average Rating');
```



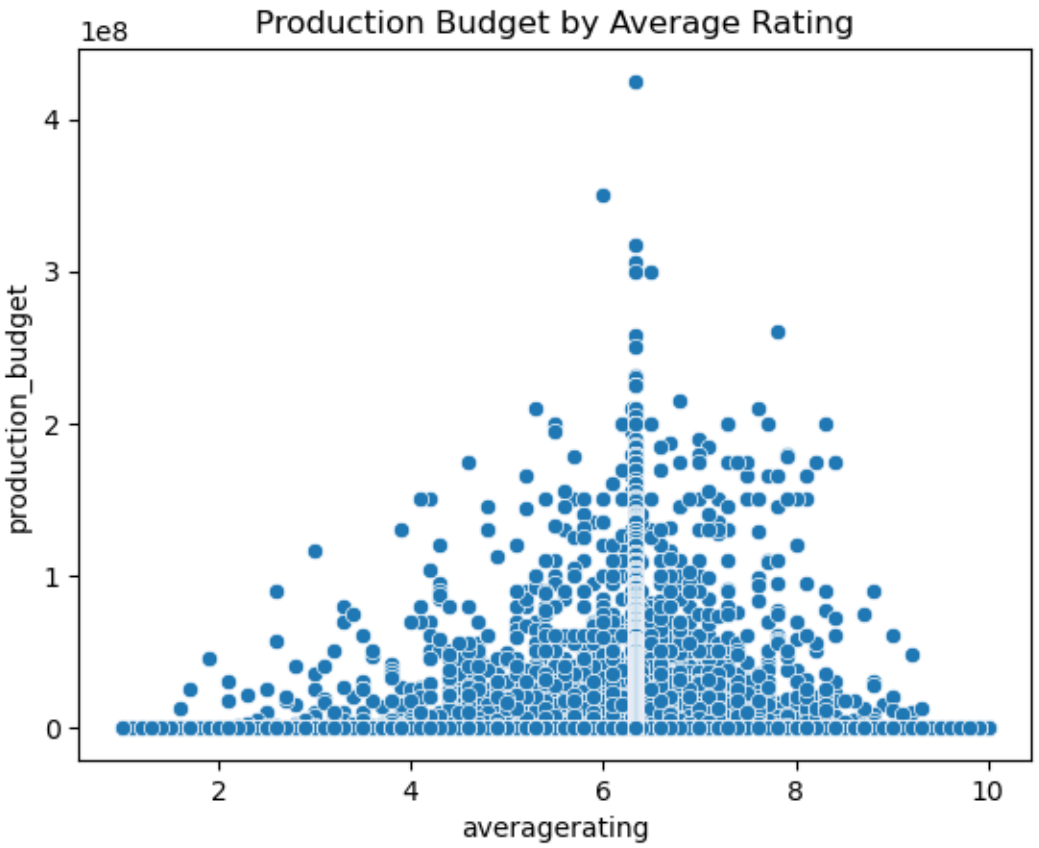
[54]:
```
# We use a scatterplot to show the relationship between production budget and␣
 ↪the worldwide gross generated for a movie
# There is a positive colleration between the production budget and the␣
 ↪worldwide gross

sns.scatterplot(x="production_budget",
                y="worldwide_gross",
                data=df2).set(title='World-wide Gross by Production Budget');
```



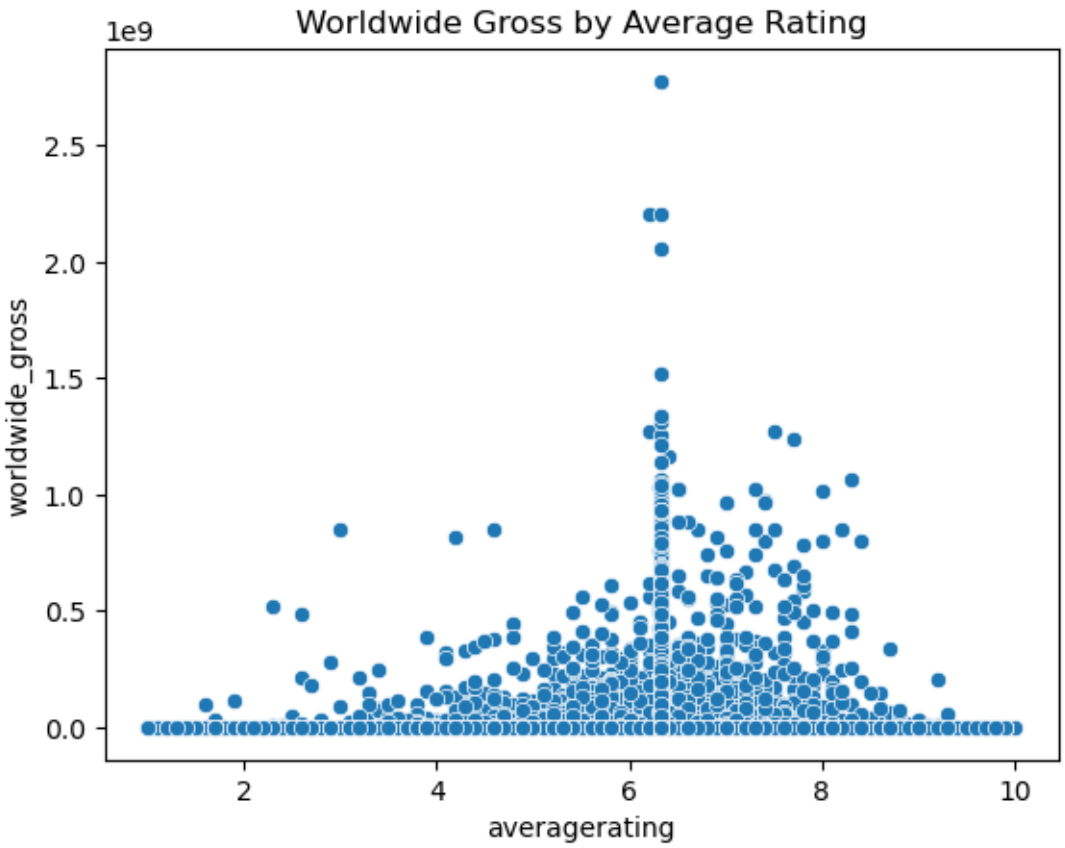
[55]:
```
# We use a scatterplot to show the relationship between production budget and␣
 ↪the domestic gross generated for a movie
# There is a positive colleration between the production budget and the␣
 ↪domestic gross

sns.scatterplot(x="production_budget",
                    y="domestic_gross",
                    data=df2).set(title='Domestic Gross by Production Budget');
```



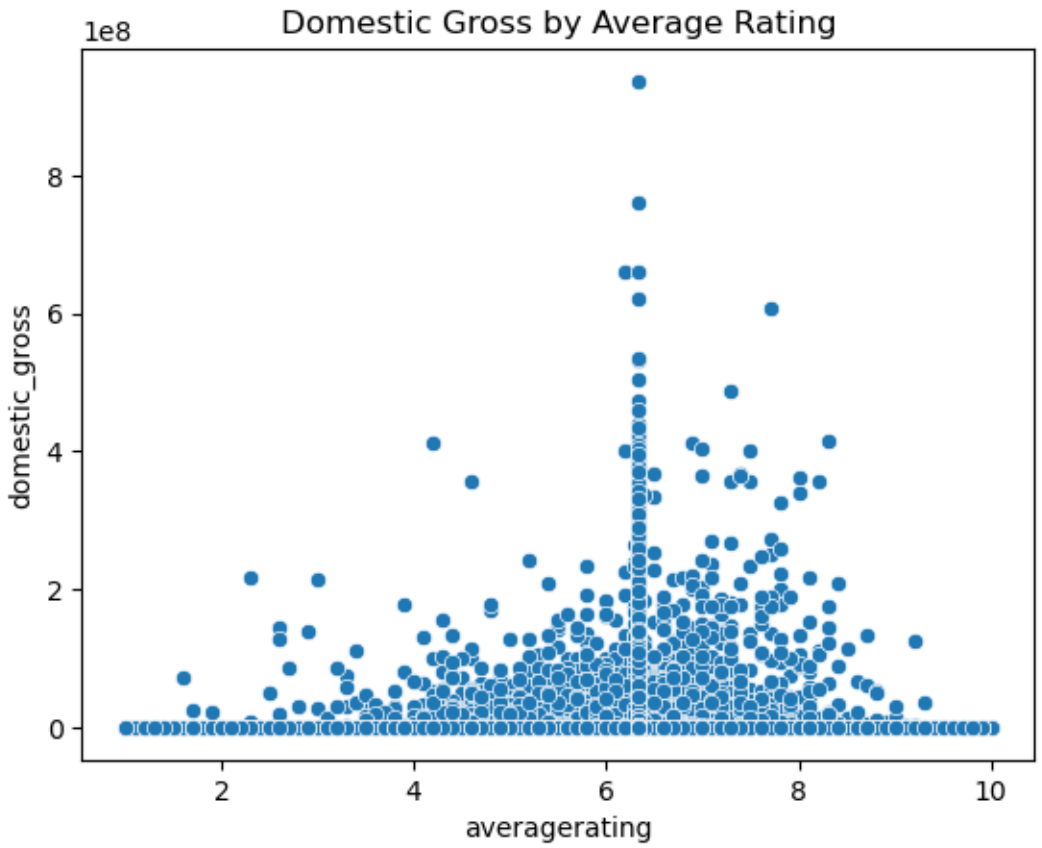
[56]:
```
# We use a scatterplot to show the relationship between production budget and␣
 ↪the average rating for a movie
# There is no clleration between the production budget and the average rating␣
 ↪of the movie

sns.scatterplot(x="averagerating",
                    y="production_budget",
                    data=df2).set(title='Production Budget by Average Rating');
```



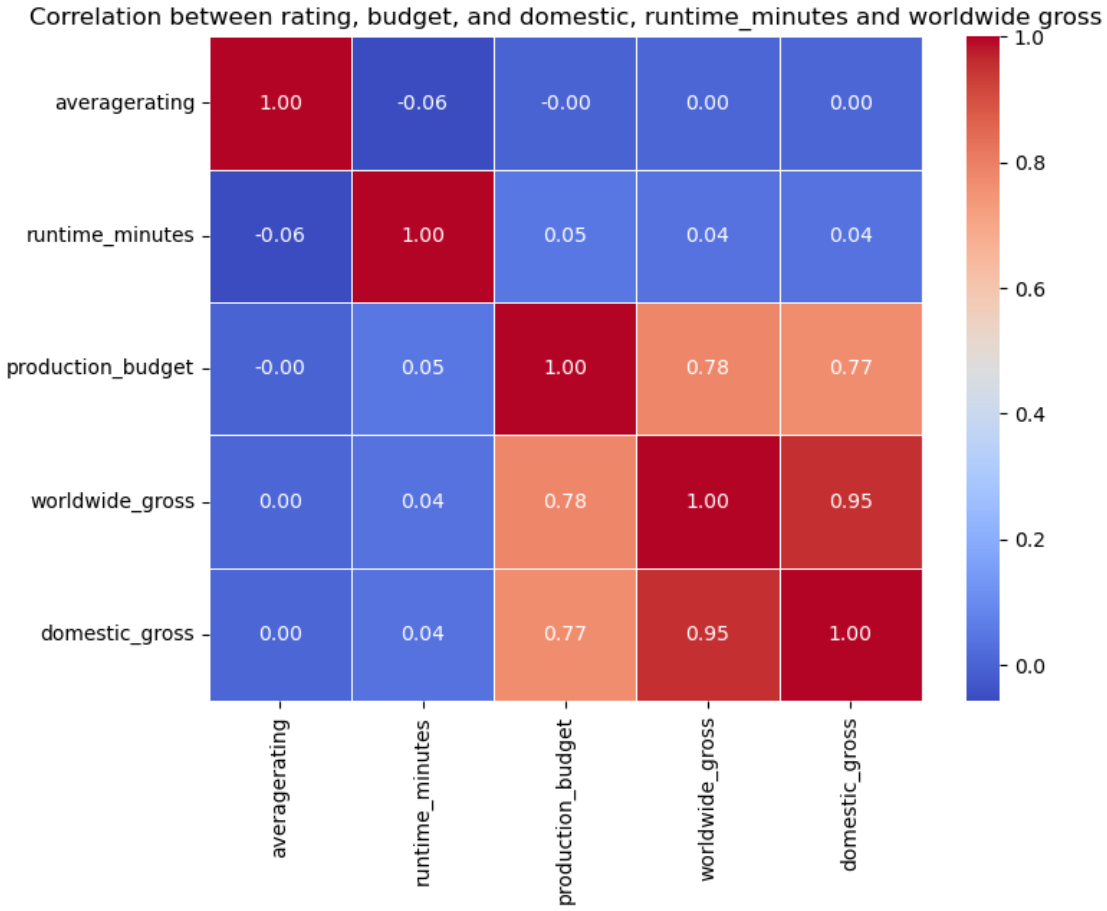
[57]:
```
# We use a scatterplot to show the relationship between worldwide gross and the␣
 ↪average rating for a movie
# There is no colleration between the worldwide gross and the average rating of␣
 ↪the movie

sns.scatterplot(x="averagerating",
                    y="worldwide_gross",
                    data=df2).set(title='Worldwide Gross by Average Rating');
```



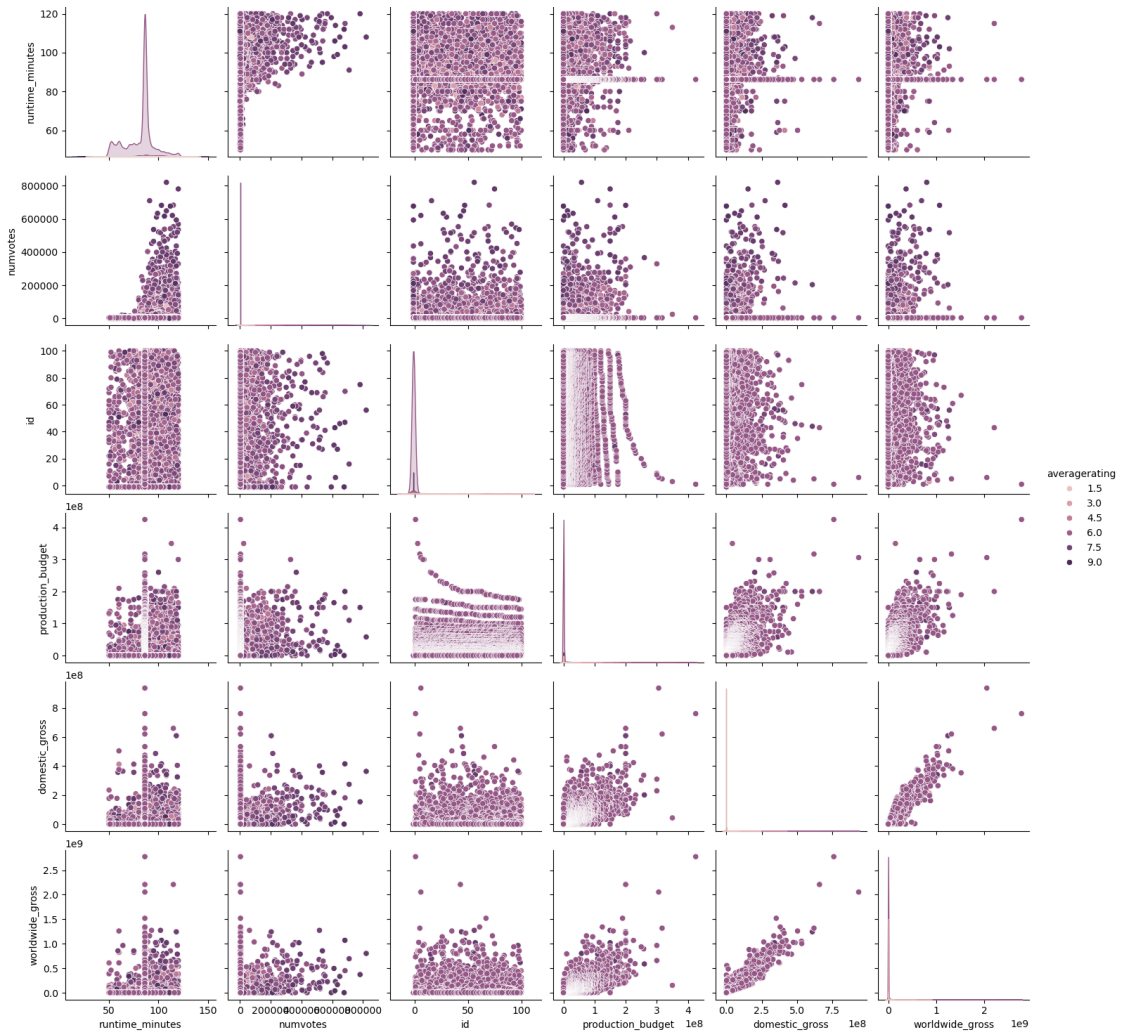
[58]:
```
# We use a scatterplot to show the relationship between domestic gross and the␣
 ↪average rating for a movie
# There isno clleration between the domestic gross and the average rating of␣
 ↪the movie

sns.scatterplot(x="averagerating",
                    y="domestic_gross",
                    data=df2).set(title='Domestic Gross by Average Rating');
```



[59]:
```
# Explore the correlation between Rating, Budget, and Domestic, runtime_minutes
 ↪and Worldwide Gross.

matrix = df2[['averagerating', 'runtime_minutes','production_budget',\
              'worldwide_gross','domestic_gross']].corr(numeric_only=False)

# Plotting the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(matrix,
            annot=True,
            cmap='coolwarm',
            fmt=".2f",
            linewidths=.5)
plt.title('Correlation between rating, budget, and domestic, runtime_minutes
 ↪and worldwide gross')
plt.show()
```



[60]:
```
# Overall data visualization to assess metrics
sns.pairplot(df2,
             hue='averagerating');
```



[61]:
```
# Check number of movies by year when the movie was released
movie_year = df.groupby(['release_year']).size().reset_index(name='Count')
movie_year.sort_values(by='Count', ascending=False).head(10)
```

[61]:

	release_year	Count
0	0	124884
91	2015	430
86	2010	298
85	2009	294
87	2011	289
90	2014	276
82	2006	276
84	2008	269
89	2013	256
88	2012	247

[62]:
```
# Check number of movies by year when the movie started/ will start
movie_year = df.groupby(['start_year']).size().reset_index(name='Count')
movie_year.sort_values(by='Count', ascending=False).head(10)
```

[62]:

	start_year	Count
8	2017	15039
7	2016	14860
9	2018	14788
6	2015	14052
5	2014	13527
4	2013	12787
3	2012	12138
2	2011	11408
1	2010	10449
10	2019	7699

1.11 Other Investigations

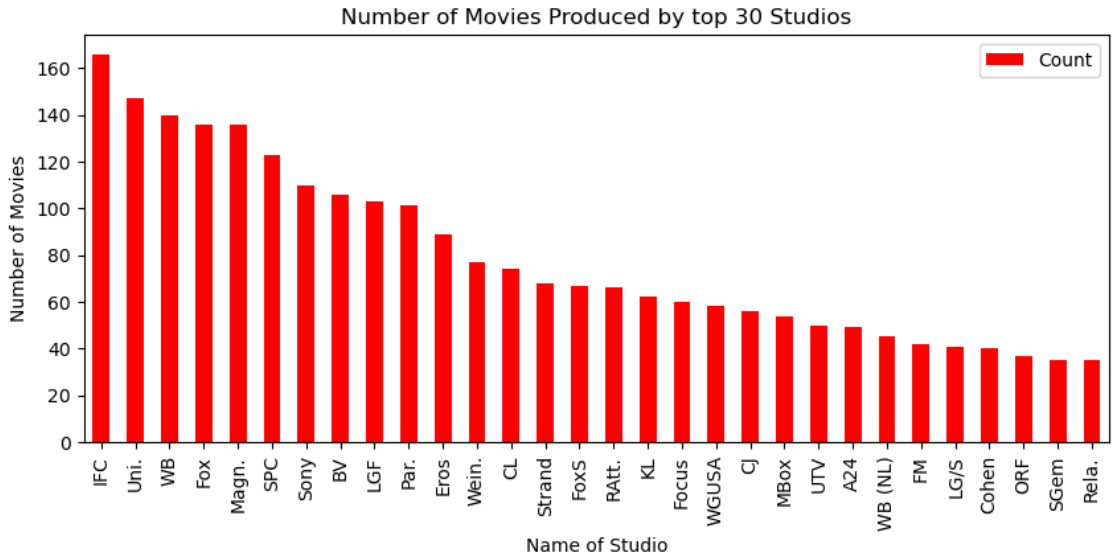
[63]:
```
bom_movie_gross
```

[63]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
…	…	…	…	…	…
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

[3387 rows x 5 columns]

[64]:
```
# Check number of movies by studio from the movie gross dataset
movie = bom_movie_gross.groupby(['studio']).size().reset_index(name='Count')
movie = movie.sort_values(by='Count', ascending=False).head(30)
movie.plot.bar(x='studio',
               y='Count',
               color = 'red',
               xlabel='Name of Studio',
               ylabel='Number of Movies',
               title = 'Number of Movies Produced by top 30 Studios',
               figsize=(10,4));
```



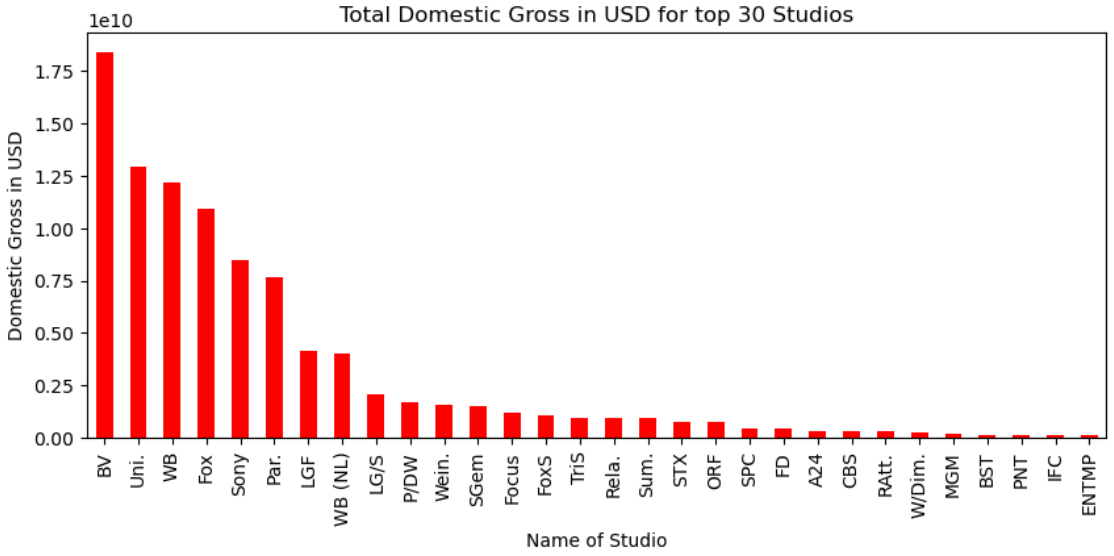
[65]:
```
# Checking the columns we need to change
bom_movie_gross.columns[-3:-1:]
```

[65]:
```
Index(['domestic_gross', 'foreign_gross'], dtype='object')
```

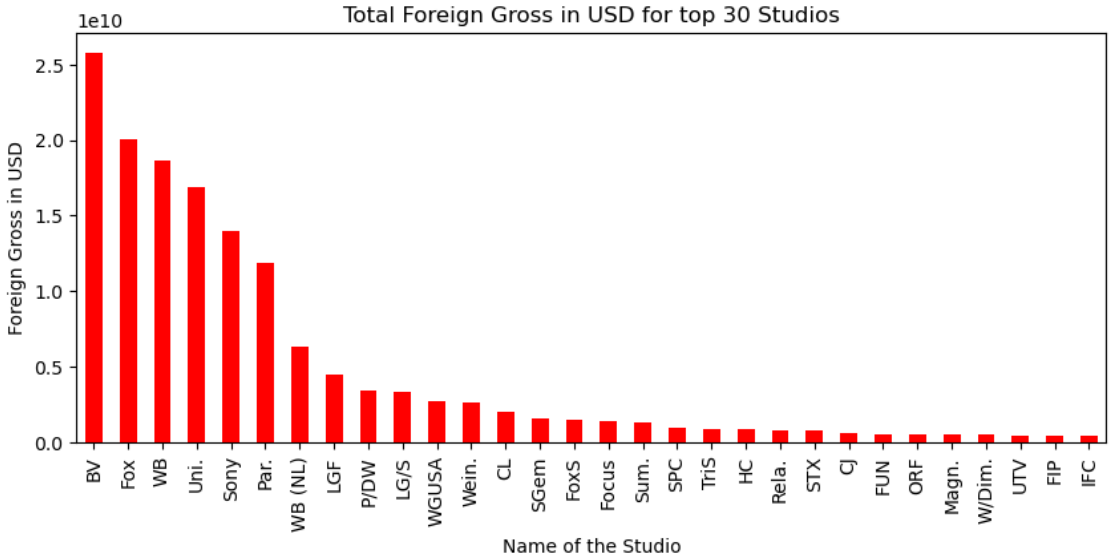
[66]:
```
# Replace $ in the columns and change the values to float
bom_movie_gross[bom_movie_gross.columns[-3:-1:]] =␣
 ↪bom_movie_gross[bom_movie_gross.columns[-3:-1:]].replace('[\$,]', '',␣
 ↪regex=True).astype(float)
```

[67]:
```
# Plotting a bar graph of studios and their total domestic gross

r = bom_movie_gross.groupby(['studio'])['domestic_gross'].sum()
r = r.sort_values(ascending=False).head(30)
r.plot.bar(color = 'red',
           xlabel='Name of Studio',
           ylabel='Domestic Gross in USD',
```

```
            title = 'Total Domestic Gross in USD for top 30 Studios',
            figsize=(10,4));
```



[68]:
```
# Plotting a bar graph of studios and their total foreign gross
r = bom_movie_gross.groupby(['studio'])['foreign_gross'].sum()
r = r.sort_values(ascending=False).head(30)
r.plot.bar(color = 'red',
           xlabel='Name of the Studio',
           ylabel='Foreign Gross in USD',
           title = 'Total Foreign Gross in USD for top 30 Studios',
           figsize=(10,4));
```

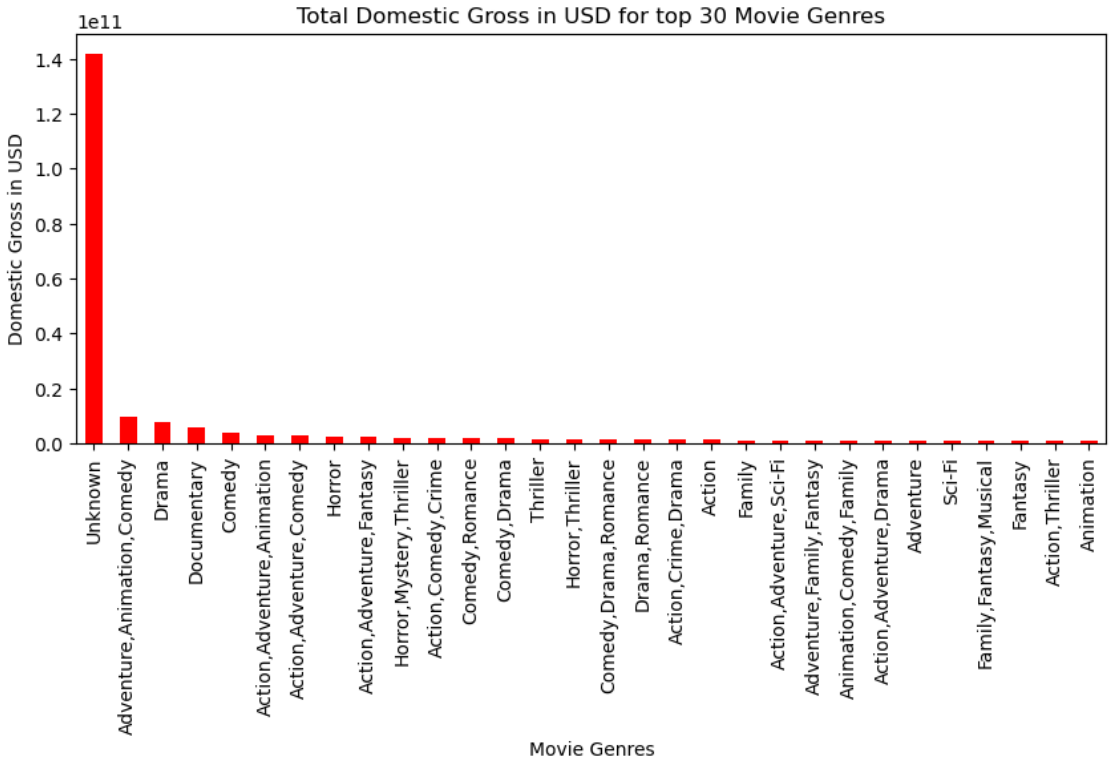


[69]:
```
# Previewing the dataset
df2.tail()
```

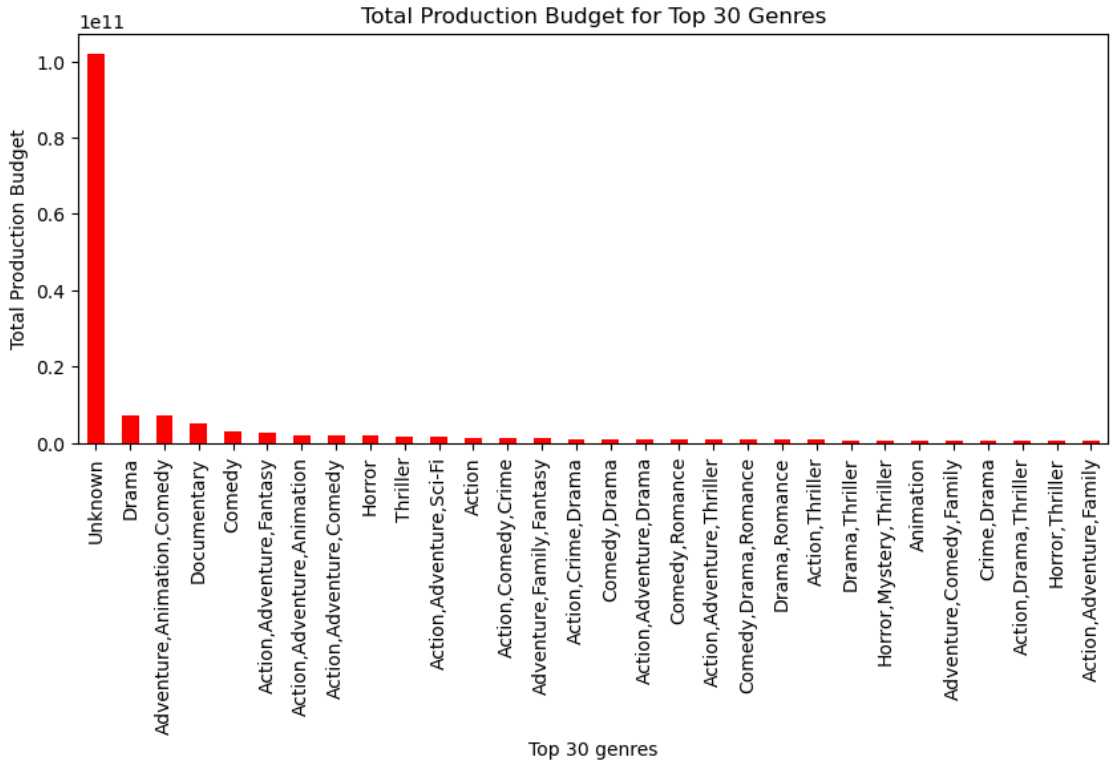
[69]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	release_year
149770	Unknown	Unknown	Unknown	0	86.190696	Unknown	6.332517	3620.574264	76.0	May 26, 2006	Cavite	7000.0	70071.0	71644.0	2006
149771	Unknown	Unknown	Unknown	0	86.190696	Unknown	6.332517	3620.574264	77.0	Dec 31, 2004	The Mongol King	7000.0	900.0	900.0	2004
149772	Unknown	Unknown	Unknown	0	86.190696	Unknown	6.332517	3620.574264	79.0	Apr 2, 1999	Following	6000.0	48482.0	240495.0	1999
149773	Unknown	Unknown	Unknown	0	86.190696	Unknown	6.332517	3620.574264	80.0	Jul 13, 2005	Return to the Land of Wonders	5000.0	1338.0	1338.0	2005
149774	Unknown	Unknown	Unknown	0	86.190696	Unknown	6.332517	3620.574264	82.0	Aug 5, 2005	My Date With Drew	1100.0	181041.0	181041.0	2005

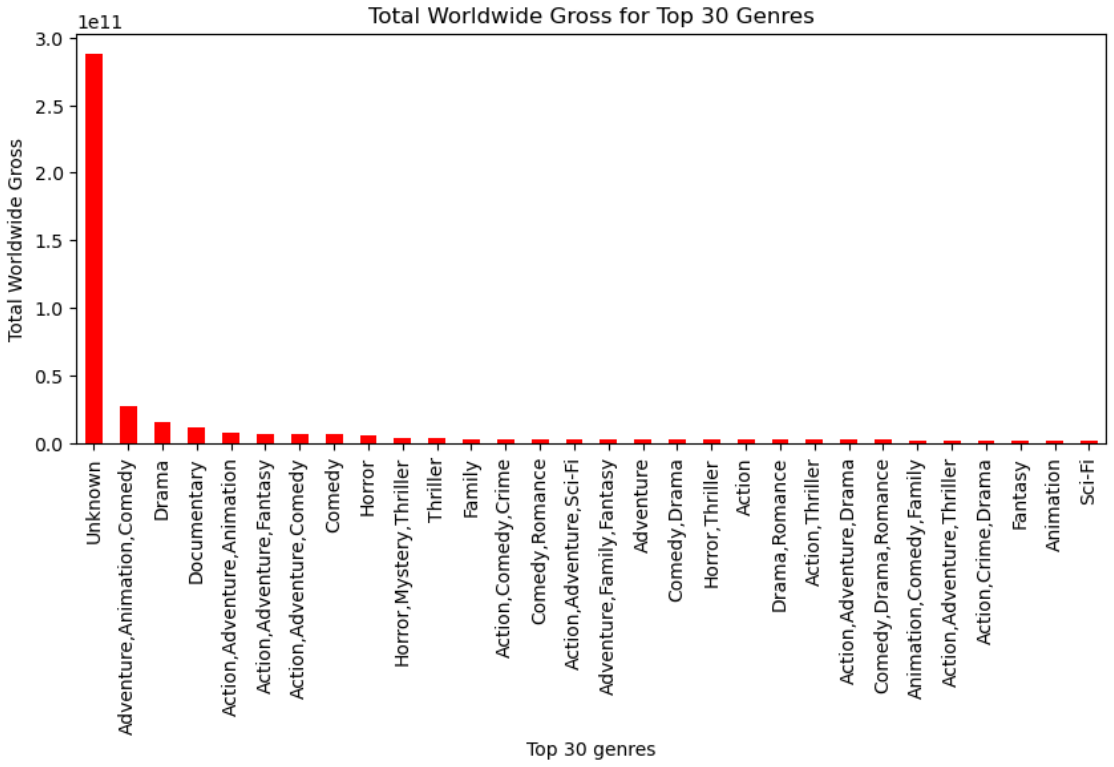
[70]:
```
# Plotting a bar graph of genres and their total domestic gross
r = df2.groupby(['genres'])['domestic_gross'].sum()
r = r.sort_values(ascending=False).head(30)
r.plot.bar(color = 'red',
           xlabel='Movie Genres',
           ylabel='Domestic Gross in USD',
           title = 'Total Domestic Gross in USD for top 30 Movie Genres',
           figsize=(10,4));
```



[71]:
```
# Plotting a bar graph of studios and their total production budget
r = df2.groupby(['genres'])['production_budget'].sum()
r = r.sort_values(ascending=False).head(30)
r.plot.bar(color = 'red',
           xlabel='Top 30 genres',
           ylabel='Total Production Budget',
           title = 'Total Production Budget for Top 30 Genres',
           figsize=(10,4));
```



[72]:
```
# Plotting a bar graph of genres and their total worlwide gross revenue
r = df2.groupby(['genres'])['worldwide_gross'].sum()
r = r.sort_values(ascending=False).head(30)
r.plot.bar(color = 'red',
           xlabel='Top 30 genres',
           ylabel='Total Worldwide Gross',
           title = 'Total Worldwide Gross for Top 30 Genres',
           figsize=(10,4));
```



1.12 Findings

- Movie rating is not influenced by runtime, budget or gross income
- Movies with higher budget attract higher gross income
- Top Studios: IFC Films, Universal, Warner Bros. Pictures, Fox and Magnolia Pictures in terms of number of movies
- Studios with Highest Gross Revenue include: BV, Universal, Warner Bros. Pictures, Fox and Sony
- Highest Gross Revenue: Adventure Animation Comedy, Drama, Documentary, Comedy Animation and Action
- Highest Production Budget: Drama, Adventure Animation Comedy, Documentary, Comedy and Action Adventure Fantasy ***

1.13 Conclusions

- Competitors to consider: IFC Films, Universal, Warner Bros. Pictures, Sony, Fox, Magnolia Pictures and Buena Vista
- Best performing movies to consider (high budget): Drama, Action Adventure Fantasy, Documentary, Comedy, Horror, Action Adventure Animation, Action Adventure Comedy

- Movie to consider with relatively low budget but high gross revenue: Horror Mystery Thriller, Family, Adventure, Comedy Romance
- Movies created should be 86 mins long on average
- To maximise profit MS can produce high quality and performing movies that require relatively low budget ***

[ ]: