Phase 1 Project Data Cleaning, Exploration and Visualizations

1) Data Cleaning

Imports

In [941…

```
# Import standard packages.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

%matplotlib inline
```

Load DataFrame with Sales Figures & Production Costs

In [942…

```
# Load data file.
df = pd.read_csv('tn.movie_budgets.csv')

# Remove "$" and "," from production_budget columns & convert to millions.
df['production_budget'] = df['production_budget'].str.replace(' ', '')
df['production_budget'] = df['production_budget'].str.replace('$', '')
df['production_budget'] = df['production_budget'].str.replace(',', '')
df['production_budget'] = df['production_budget'].str[:-6]

# Remove "$" and "," from domestic_gross columns & convert to millions.
df['domestic_gross'] = df['domestic_gross'].str.replace(' ', '')
df['domestic_gross'] = df['domestic_gross'].str.replace('$', '')
df['domestic_gross'] = df['domestic_gross'].str.replace(',', '')
df['domestic_gross'] = df['domestic_gross'].str[:-6]

# Remove "$" and "," from worldwide_gross columns & convert to millions.
df['worldwide_gross'] = df['worldwide_gross'].str.replace(' ', '')
df['worldwide_gross'] = df['worldwide_gross'].str.replace('$', '')
df['worldwide_gross'] = df['worldwide_gross'].str.replace(',', '')
df['worldwide_gross'] = df['worldwide_gross'].str[:-6]

# Filter blank strings.
df = df[df.production_budget != '']
df = df[df.worldwide_gross != '']
df = df[df.domestic_gross != '']

# Change figures from str to int class type.
df['production_budget'] = df['production_budget'].astype(int)
df['domestic_gross'] = df['domestic_gross'].astype(int)
df['worldwide_gross'] = df['worldwide_gross'].astype(int)

# Create international_gross col.
df['international_gross'] = df['worldwide_gross'] - df['domestic_gross']
```

```
# Filter international_gross.
df = df[df.international_gross != 0]

# Confirm.
df
```

Out[942…

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	international_g
0	1	18-Dec-09	Avatar	425	760	2776	2
1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	3	7-Jun-19	Dark Phoenix	350	42	149	
3	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
4	5	15-Dec-17	Star Wars Ep. VIII: The Last Jedi	317	620	1316	
...	...	...	...	...	...	...	
5242	43	3-Aug-05	Junebug	1	2	3	
5243	44	1-Aug-08	Frozen River	1	2	6	
5244	45	21-Nov-01	Sidewalks of New York	1	2	3	
5246	47	29-Sep-00	The Broken Hearts Club: A Romantic Comedy	1	1	2	
5251	52	7-Dec-01	No Man's Land	1	1	2	

3252 rows × 7 columns

Load DataFrame with Genre Data

In [943…

```
# Load genre file.
genre_df = pd.read_csv('tmdb.movies.csv')

# Filter to relevant columns.
genre_df = genre_df[['genre_ids', 'title']]
```

```
# Replace genre id with corresponding genre.
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('12', 'adventure')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('14', 'fantasy')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('16', 'animation')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('18', 'drama')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('27', 'horror')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('28', 'action')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('35', 'comedy')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('36', 'history')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('37', 'western')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('53', 'thriller')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('80', 'crime')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('99', 'documentary')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('878', 'sci-fi')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('9648', 'mystery')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('10402', 'music')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('10749', 'romance')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('10751', 'family')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('10752', 'war')
genre_df['genre_ids'] = genre_df['genre_ids'].str.replace('10770', 'tv movie')

# Confirm changes.
genre_df
```

Out[943…

	genre_ids	title
0	[adventure, fantasy, family]	Harry Potter and the Deathly Hallows: Part 1
1	[fantasy, adventure, animation, family]	How to Train Your Dragon
2	[adventure, action, sci-fi]	Iron Man 2
3	[animation, comedy, family]	Toy Story
4	[action, sci-fi, adventure]	Inception
...	...	...
26512	[horror, drama]	Laboratory Conditions
26513	[drama, thriller]	_EXHIBIT_84xxx_
26514	[fantasy, action, adventure]	The Last One
26515	[family, adventure, action]	Trailer Made
26516	[thriller, horror]	The Church

26517 rows × 2 columns

Create New List that Contains Movies Found in Both DataFrames

In [944…

```
# Create list for with same title for merging.
same_title_list = []

#convert both title columns to lists for conversion later.
genre_title_list = genre_df['title'].tolist()
gross_title_list = df['movie'].tolist()
```

```
# Create duplicate title list.
for title in genre_title_list:
    for movie in gross_title_list:
        if title == movie:
            same_title_list.append(title)
        else:
            continue

# Print current list length.
print("Orignal list count: " + str(len(same_title_list)))

# Remove duplicates from list.
same_title_list = list(dict.fromkeys(same_title_list))

# Print new list length.
print("New list count: " + str(len(same_title_list)))
```

```
Orignal list count: 1565
New list count: 1276
```

Reduce Genre DataFrame to Movies Found in Both DataFrames

In [945…

```
# Reduce to genre_df to movies found in both dataframes.
genre_df = genre_df[genre_df['title'].isin(same_title_list)]

# Confirm changes.
genre_df
```

Out[945…

	genre_ids	title
1	[fantasy, adventure, animation, family]	How to Train Your Dragon
2	[adventure, action, sci-fi]	Iron Man 2
3	[animation, comedy, family]	Toy Story
4	[action, sci-fi, adventure]	Inception
5	[adventure, fantasy, family]	Percy Jackson & the Olympians: The Lightning T...
...	...	...
26323	[]	The Box
26339	[documentary, documentary]	The Judge
26413	[horror]	Wolf
26425	[music]	The Box
26508	[animation]	Jaws

1531 rows × 2 columns

Reduce Sales/Production DataFrame to Movies Found in Both DataFrames

In [946…
```
# Reduce to df to movies found in both dataframes
df = df[df['movie'].isin(same_title_list)]

# Confirm changes
df
```

Out[946…

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	international_g
0	1	18-Dec-09	Avatar	425	760	2776	2
1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
3	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
6	7	27-Apr-18	Avengers: Infinity War	300	678	2048	1
8	9	17-Nov-17	Justice League	300	229	655	
…	…	…	…	…	…	…	
5230	31	25-Sep-15	The Green Inferno	1	7	12	
5231	32	3-Feb-17	I Am Not Your Negro	1	7	9	
5232	33	19-Oct-12	The Sessions	1	6	11	
5241	42	21-Oct-11	Martha Marcy May Marlene	1	2	5	
5251	52	7-Dec-01	No Man's Land	1	1	2	

1296 rows × 7 columns

Clean Column Titles for Merging

In [947…
```
# Rename genre_df 'title' column to 'movie in prep for merge
genre_df = genre_df.rename(columns={'title' : 'movie'})

# Confirm changes
print(genre_df.head())
df
```

```
                                 genre_ids  \
1  [fantasy, adventure, animation, family]
2              [adventure, action, sci-fi]
3              [animation, comedy, family]
4              [action, sci-fi, adventure]
5             [adventure, fantasy, family]

                                                movie
1                          How to Train Your Dragon
2                                          Iron Man 2
3                                           Toy Story
4                                           Inception
5  Percy Jackson & the Olympians: The Lightning T...
```

Out[947…

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	international_g
0	1	18-Dec-09	Avatar	425	760	2776	2
1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
3	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
6	7	27-Apr-18	Avengers: Infinity War	300	678	2048	1
8	9	17-Nov-17	Justice League	300	229	655	
...	...	...	...	...	...	...	
5230	31	25-Sep-15	The Green Inferno	1	7	12	
5231	32	3-Feb-17	I Am Not Your Negro	1	7	9	
5232	33	19-Oct-12	The Sessions	1	6	11	
5241	42	21-Oct-11	Martha Marcy May Marlene	1	2	5	
5251	52	7-Dec-01	No Man's Land	1	1	2	

1296 rows × 7 columns

Merge Data Frames

In [948…

```
# Merge dataframes into one dataframe, "merged_df".
```

```
merged_df = pd.merge(df, genre_df, on = 'movie', how = 'outer')

# Print new dataframe length.
print("Combined DataFrame length is: " + str(len(merged_df)))
```

```
Combined DataFrame length is: 1565
```

In [949…

```
# Drop duplicates.
merged_df.drop_duplicates(subset = 'movie', keep = False, inplace = True)

# Confirm results.
merged_df
```

Out[949…

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	international_g
0	1	18-Dec-09	Avatar	425	760	2776	
1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
3	7	27-Apr-18	Avengers: Infinity War	300	678	2048	
8	11	20-Jul-12	The Dark Knight Rises	275	448	1084	
...	...	...	...	...	...	...	
1557	29	20-Jul-18	Unfriended: Dark Web	1	8	16	
1558	31	25-Sep-15	The Green Inferno	1	7	12	
1559	32	3-Feb-17	I Am Not Your Negro	1	7	9	
1560	33	19-Oct-12	The Sessions	1	6	11	
1561	42	21-Oct-11	Martha Marcy May Marlene	1	2	5	

1046 rows × 8 columns

Load Nick's DataFrame Which Combined Ours Together

In [950…

```
# Load file.
fnl_df = pd.read_csv('master_data_raw.csv')

# Select relevant columns for data exploration.
fnl_df = fnl_df[['movie_CH', 'production_budget_CH', 'domestic_gross_CH', 'worldwide_gr
                 'runtime_minutes', 'studio', 'genres', 'Action',
                 'Animation', 'Comedy','Family']]

# Test.
fnl_df
```

Out[950…

	movie_CH	production_budget_CH	domestic_gross_CH	worldwide_gross_CH	international_gross	r
0	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	
4	The Secret Life of Walter Mitty	91.0	58.0	187.0	129.0	
…	…	…	…	…	…	
3892	NaN	NaN	NaN	NaN	NaN	
3893	NaN	NaN	NaN	NaN	NaN	
3894	NaN	NaN	NaN	NaN	NaN	
3895	NaN	NaN	NaN	NaN	NaN	
3896	NaN	NaN	NaN	NaN	NaN	

3897 rows × 13 columns

In [951…

```
# Drop NaN & duplicates
fnl_df.dropna(axis=0, how = 'any', inplace=True)
fnl_df.drop_duplicates(subset = 'movie_CH', keep = False, inplace = True)

# Add profit column
fnl_df['profit'] = fnl_df['worldwide_gross_CH'] - fnl_df['production_budget_CH']

# Remove unused genres for graphing purposes.
fnl_df['genres'] = fnl_df['genres'].str.replace('Adventure', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Horror', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Mystery', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Thriller', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Musical', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Romance', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Fantasy', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('War', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Sci-Fi', '')
```

```
fnl_df['genres'] = fnl_df['genres'].str.replace('Western', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Drama', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Music', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Documentary', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Biography', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Sport', '')
fnl_df['genres'] = fnl_df['genres'].str.replace('Crime', '')
fnl_df['genres'] = fnl_df['genres'].str.replace(',', '')

# Confirm Results.
fnl_df
```

Out[951…

	movie_CH	production_budget_CH	domestic_gross_CH	worldwide_gross_CH	international_gro
4	The Secret Life of Walter Mitty	91.0	58.0	187.0	129
5	A Walk Among the Tombstones	28.0	26.0	62.0	36
6	Jurassic World	215.0	652.0	1648.0	996
7	The Rum Diary	45.0	13.0	21.0	8
8	The Three Stooges	30.0	44.0	54.0	10
...	...	...	...	...	
3522	Uncle Drew	18.0	42.0	46.0	4
3523	BlacKkKlansman	15.0	49.0	93.0	44
3534	Paul, Apostle of Christ	5.0	17.0	25.0	8
3541	Instant Family	48.0	67.0	119.0	52
3604	Hereditary	10.0	44.0	70.0	26

754 rows × 14 columns

2) Data Exploration

Set Up Action Genre DataFrames

In [952…

```
# Set up new df for genre.
action_df = fnl_df[(fnl_df['Action'] == True) & (fnl_df['Animation'] != True)]

# Establish new data points for genre.
action_movies = len(action_df)
action_prod = sum(action_df['production_budget_CH'])
action_salesd = sum(action_df['domestic_gross_CH'])
action_salesw = sum(action_df['worldwide_gross_CH'])
action_salesi = sum(action_df['international_gross'])
total_action_profit = action_salesw - action_prod
film_action_profit = total_action_profit / action_movies
```

```
print(f' Average profit per Action film is roughly {round(film_action_profit)}M' )
```

```
Average profit per Action film is roughly 192M
```

In [953…
```
# Set up new df for genre
a_action_df = fnl_df[(fnl_df['Action'] == True) & (fnl_df['Animation'] == True)]

# Establish new data points for genre
a_action_movies = len(a_action_df)
a_action_prod = sum(a_action_df['production_budget_CH'])
a_action_salesd = sum(a_action_df['domestic_gross_CH'])
a_action_salesw = sum(a_action_df['worldwide_gross_CH'])
a_action_salesi = sum(a_action_df['international_gross'])
a_total_action_profit = a_action_salesw - a_action_prod
a_film_action_profit = a_total_action_profit / a_action_movies

print(f' Average profit per Animated Action film is roughly {round(a_film_action_profit
```

```
Average profit per Animated Action film is roughly 328M
```

In [954…
```
# Set up new df for genre
animation_df = fnl_df[(fnl_df['Animation'] == True)]

# Establish new data points for genre
animation_movies = len(animation_df)
animation_prod = sum(animation_df['production_budget_CH'])
animation_salesd = sum(animation_df['domestic_gross_CH'])
animation_salesw = sum(animation_df['worldwide_gross_CH'])
animation_salesi = sum(animation_df['international_gross'])
total_animation_profit = animation_salesw - animation_prod
film_animation_profit = total_animation_profit / animation_movies

print(f' Average profit per Animated film is roughly {round(film_animation_profit)}M' )
```

```
Average profit per Animated film is roughly 300M
```

In [955…
```
# Set up new df for genre
comedy_df = fnl_df[(fnl_df['Comedy'] == True) & (fnl_df['Animation'] != True)]

# Establish new data points for genre
comedy_movies = len(comedy_df)
comedy_prod = sum(comedy_df['production_budget_CH'])
comedy_salesd = sum(comedy_df['domestic_gross_CH'])
comedy_salesw = sum(comedy_df['worldwide_gross_CH'])
comedy_salesi = sum(comedy_df['international_gross'])
total_comedy_profit = comedy_salesw - comedy_prod
film_comedy_profit = total_comedy_profit / comedy_movies

print(f' Average profit per Comedy film is roughly {round(film_comedy_profit)}M' )
```

```
Average profit per Comedy film is roughly 76M
```

In [956…
```
# Set up new df for genre
a_comedy_df = fnl_df[(fnl_df['Comedy'] == True) & (fnl_df['Animation'] == True)]

# Establish new data points for genre
a_comedy_movies = len(a_comedy_df)
a_comedy_prod = sum(a_comedy_df['production_budget_CH'])
a_comedy_salesd = sum(a_comedy_df['domestic_gross_CH'])
a_comedy_salesw = sum(a_comedy_df['worldwide_gross_CH'])
```

```
a_comedy_salesi = sum(a_comedy_df['international_gross'])
a_total_comedy_profit = a_comedy_salesw - a_comedy_prod
a_film_comedy_profit = a_total_comedy_profit / a_comedy_movies

print(f' Average profit per Animated Comedy film is roughly {round(a_film_comedy_profit
```

```
Average profit per Animated Comedy film is roughly 294M
```

In [957…

```
# Set up new df for genre
family_df = fnl_df[(fnl_df['Family'] == True) & (fnl_df['Animation'] != True)]

# Establish new data points for genre
family_movies = len(family_df)
family_prod = sum(family_df['production_budget_CH'])
family_salesd = sum(family_df['domestic_gross_CH'])
family_salesw = sum(family_df['worldwide_gross_CH'])
family_salesi = sum(family_df['international_gross'])
total_family_profit = family_salesw - family_prod
film_family_profit = total_family_profit / family_movies

print(f' Average profit per Family film is roughly {round(film_family_profit)}M' )
```

```
Average profit per Family film is roughly 108M
```

In [958…

```
# Set up new df for genre
a_family_df = fnl_df[(fnl_df['Family'] == True) & (fnl_df['Animation'] == True)]

# Establish new data points for genre
a_family_movies = len(a_family_df)
a_family_prod = sum(a_family_df['production_budget_CH'])
a_family_salesd = sum(a_family_df['domestic_gross_CH'])
a_family_salesw = sum(a_family_df['worldwide_gross_CH'])
a_family_salesi = sum(a_family_df['international_gross'])
a_total_family_profit = a_family_salesw - a_family_prod
a_film_family_profit = a_total_family_profit / a_family_movies

print(f' Average profit per Family film is roughly {round(a_film_family_profit)}M' )
```

```
Average profit per Family film is roughly 208M
```

In [959…

```
# Creating lists to store total movie values.
all_movies = [action_movies, comedy_movies, family_movies]
all_animated_movies = [a_action_movies, a_comedy_movies, a_family_movies]
all_profit = [total_action_profit, total_comedy_profit, total_family_profit]
all_animated_profit = [a_total_action_profit, a_total_comedy_profit, a_total_family_pro

# Calculations for presentation.
total_movies = sum(all_movies)
total_animated_movies = sum(all_animated_movies)

total_movie_profit = sum(all_profit)
total_animated_profit = sum(all_animated_profit)

film_avg_profit = total_movie_profit / total_movies
a_film_avg_profit = total_animated_profit / total_animated_movies

avg_animated_profit_increase = a_film_avg_profit / film_avg_profit
avg_increase_per_film = a_film_avg_profit - film_avg_profit
```

```
# Looking at values.
print(f'Total movies that meet our criteria {total_movies+total_animated_movies}')
print(f'You can expect roughly {round(avg_animated_profit_increase, 2)} times more prof
```

```
Total movies that meet our criteria 607
You can expect roughly 2.2 times more profit when animating (120% increase).
```

3) Create Visuals

Clean Genres Specific DataFrames for Visuals

In [960…

```
# Setting up columns for graphing.
family_df.loc[:,'genres'] = 'Family'
a_family_df.loc[:,'genres'] = 'Animation/Family'
comedy_df.loc[:,'genres'] = 'Comedy'
a_comedy_df.loc[:,'genres'] = 'Animation/Comedy'
action_df.loc[:,'genres'] = 'Action'
a_action_df.loc[:,'genres'] = 'Animated/Action'
```

```
C:\Users\helme\anaconda3\envs\learn-env\lib\site-packages\pandas\core\indexing.py:1765:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value)
```

In [961…

```
# Create copy of fnl_df for clean genres.
clean_genres_df = fnl_df.copy()

# Convert filtered dataframe data in tuples for cleaned genres.
cg_family = tuple(zip(family_df.movie_CH, family_df.genres))
cg_a_family = tuple(zip(a_family_df.movie_CH, a_family_df.genres))
cg_comedy = tuple(zip(comedy_df.movie_CH, comedy_df.genres))
cg_a_comedy = tuple(zip(a_comedy_df.movie_CH, a_comedy_df.genres))
cg_action = tuple(zip(action_df.movie_CH, action_df.genres))
cg_a_action = tuple(zip(a_action_df.movie_CH, a_action_df.genres))

# Combine tuples.
all_genres_tup = cg_family + cg_a_family + cg_comedy + cg_a_comedy + cg_action + cg_a_a

# Confirm results.
all_genres_tup
```

Out[961…

```
(('The Three Stooges', 'Family'),
 ('Real Steel', 'Family'),
 ('Furry Vengeance', 'Family'),
 ('Ramona and Beezus', 'Family'),
 ('Percy Jackson & the Olympians: The Lightning Thief', 'Family'),
 ('The Hobbit: An Unexpected Journey', 'Family'),
 ('The Last Airbender', 'Family'),
 ('The Chronicles of Narnia: The Voyage of the Dawn Treader', 'Family'),
 ('Secretariat', 'Family'),
 ('Parental Guidance', 'Family'),
 ('Monte Carlo', 'Family'),
 ('Paddington', 'Family'),
 ('Diary of a Wimpy Kid', 'Family'),
 ('The Muppets', 'Family'),
 ('Zookeeper', 'Family'),
```

```
('The Spy Next Door', 'Family'),
('Cats & Dogs: The Revenge of Kitty Galore', 'Family'),
('We Bought a Zoo', 'Family'),
('Marmaduke', 'Family'),
('You Again', 'Family'),
('Big Miracle', 'Family'),
('Letters to God', 'Family'),
('The Odd Life of Timothy Green', 'Family'),
('Dolphin Tale', 'Family'),
('Maleficent', 'Family'),
('Soul Surfer', 'Family'),
('A Wrinkle in Time', 'Family'),
('Oz the Great and Powerful', 'Family'),
('Diary of a Wimpy Kid: Rodrick Rules', 'Family'),
('Alexander and the Terrible, Horrible, No Good, Very Bad Day', 'Family'),
('Percy Jackson: Sea of Monsters', 'Family'),
('Heaven is for Real', 'Family'),
('Diary of a Wimpy Kid: Dog Days', 'Family'),
('Earth to Echo', 'Family'),
('Alice Through the Looking Glass', 'Family'),
('Night at the Museum: Secret of the Tomb', 'Family'),
('Dolphin Tale 2', 'Family'),
('Pan', 'Family'),
('The BFG', 'Family'),
('Fantastic Beasts: The Crimes of Grindelwald', 'Family'),
('Miracles from Heaven', 'Family'),
('Middle School: The Worst Years of My Life', 'Family'),
('Mary Poppins Returns', 'Family'),
('The Nutcracker and the Four Realms', 'Family'),
('Goosebumps 2: Haunted Halloween', 'Family'),
('Diary of a Wimpy Kid: The Long Haul', 'Family'),
('Hotel Transylvania', 'Animation/Family'),
('Epic', 'Animation/Family'),
('Frankenweenie', 'Animation/Family'),
('Mars Needs Moms', 'Animation/Family'),
('Despicable Me', 'Animation/Family'),
('Hotel Transylvania 2', 'Animation/Family'),
('The Secret Life of Walter Mitty', 'Comedy'),
('The Rum Diary', 'Comedy'),
('The Three Stooges', 'Comedy'),
('Dinner for Schmucks', 'Comedy'),
('Piranha 3D', 'Comedy'),
('Tower Heist', 'Comedy'),
('Hail, Caesar!', 'Comedy'),
('Ant-Man', 'Comedy'),
('Hall Pass', 'Comedy'),
('Something Borrowed', 'Comedy'),
('Furry Vengeance', 'Comedy'),
('Ramona and Beezus', 'Comedy'),
('A Thousand Words', 'Comedy'),
('The Incredible Burt Wonderstone', 'Comedy'),
('R.I.P.D.', 'Comedy'),
('Jack and Jill', 'Comedy'),
('Take Me Home Tonight', 'Comedy'),
('The Kids Are All Right', 'Comedy'),
('The Wedding Ringer', 'Comedy'),
('The Switch', 'Comedy'),
('Letters to Juliet', 'Comedy'),
('Little Fockers', 'Comedy'),
('The Green Hornet', 'Comedy'),
('Dylan Dog: Dead of Night', 'Comedy'),
('The Bounty Hunter', 'Comedy'),
('Silver Linings Playbook', 'Comedy'),
('Parental Guidance', 'Comedy'),
('Monte Carlo', 'Comedy'),
```

```
('Dark Shadows', 'Comedy'),
('Eddie the Eagle', 'Comedy'),
('Blended', 'Comedy'),
('Paddington', 'Comedy'),
('Morning Glory', 'Comedy'),
('Baggage Claim', 'Comedy'),
('A Walk in the Woods', 'Comedy'),
('You Will Meet a Tall Dark Stranger', 'Comedy'),
('The Five-Year Engagement', 'Comedy'),
('Diary of a Wimpy Kid', 'Comedy'),
('The Muppets', 'Comedy'),
('Last Vegas', 'Comedy'),
('Zookeeper', 'Comedy'),
('Get Him to the Greek', 'Comedy'),
('Anchorman 2: The Legend Continues', 'Comedy'),
('Due Date', 'Comedy'),
('Hot Tub Time Machine', 'Comedy'),
('21 Jump Street', 'Comedy'),
('Aloha', 'Comedy'),
('Kick-Ass', 'Comedy'),
('Sex and the City 2', 'Comedy'),
('2 Guns', 'Comedy'),
('The Spy Next Door', 'Comedy'),
('Date Night', 'Comedy'),
('Easy A', 'Comedy'),
('Bad Teacher', 'Comedy'),
('Cats & Dogs: The Revenge of Kitty Galore', 'Comedy'),
('The Watch', 'Comedy'),
('Our Family Wedding', 'Comedy'),
('Seeking a Friend for the End of the World', 'Comedy'),
('Movie 43', 'Comedy'),
('Rock of Ages', 'Comedy'),
('Made in Dagenham', 'Comedy'),
('Pride and Prejudice and Zombies', 'Comedy'),
('Grown Ups', 'Comedy'),
('Cop Out', 'Comedy'),
('The Other Guys', 'Comedy'),
('We Bought a Zoo', 'Comedy'),
('Marmaduke', 'Comedy'),
('Journey 2: The Mysterious Island', 'Comedy'),
('The Hangover Part II', 'Comedy'),
('The Best Exotic Marigold Hotel', 'Comedy'),
('Table 19', 'Comedy'),
('You Again', 'Comedy'),
('Somewhere', 'Comedy'),
('Deadpool', 'Comedy'),
('Quartet', 'Comedy'),
('The Odd Life of Timothy Green', 'Comedy'),
('Big Mommas: Like Father, Like Son', 'Comedy'),
('Baywatch', 'Comedy'),
('Cedar Rapids', 'Comedy'),
('Bridesmaids', 'Comedy'),
('Attack the Block', 'Comedy'),
('The Change-Up', 'Comedy'),
('Central Intelligence', 'Comedy'),
('Horrible Bosses', 'Comedy'),
('Beginners', 'Comedy'),
('Hope Springs', 'Comedy'),
('The Guard', 'Comedy'),
('The Dilemma', 'Comedy'),
('Larry Crowne', 'Comedy'),
('Warm Bodies', 'Comedy'),
('This Means War', 'Comedy'),
('The Big Short', 'Comedy'),
('One for the Money', 'Comedy'),
```

```
('American Reunion', 'Comedy'),
('Zoolander 2', 'Comedy'),
('Think Like a Man', 'Comedy'),
('30 Minutes or Less', 'Comedy'),
('Young Adult', 'Comedy'),
('Friends with Benefits', 'Comedy'),
('Johnny English Reborn', 'Comedy'),
('Our Idiot Brother', 'Comedy'),
('Ted', 'Comedy'),
('Jumping the Broom', 'Comedy'),
('The Dictator', 'Comedy'),
('Here Comes the Boom', 'Comedy'),
('Diary of a Wimpy Kid: Rodrick Rules', 'Comedy'),
('Kick-Ass 2', 'Comedy'),
('Grudge Match', 'Comedy'),
('Fun Size', 'Comedy'),
('Vampires Suck', 'Comedy'),
('The DUFF', 'Comedy'),
('Mirror Mirror', 'Comedy'),
('The Sapphires', 'Comedy'),
('Entourage', 'Comedy'),
('The Guilt Trip', 'Comedy'),
('Alexander and the Terrible, Horrible, No Good, Very Bad Day', 'Comedy'),
('Joyful Noise', 'Comedy'),
('Office Christmas Party', 'Comedy'),
('Scouts Guide to the Zombie Apocalypse', 'Comedy'),
('Moonrise Kingdom', 'Comedy'),
('Danny Collins', 'Comedy'),
('That Awkward Moment', 'Comedy'),
('Nebraska', 'Comedy'),
('RED 2', 'Comedy'),
('To Rome with Love', 'Comedy'),
('Dirty Grandpa', 'Comedy'),
('The Sessions', 'Comedy'),
('Magic Mike', 'Comedy'),
('The Big Wedding', 'Comedy'),
('Seven Psychopaths', 'Comedy'),
('Sex Tape', 'Comedy'),
('Pitch Perfect', 'Comedy'),
('Robot & Frank', 'Comedy'),
('War Dogs', 'Comedy'),
('Guardians of the Galaxy', 'Comedy'),
('Diary of a Wimpy Kid: Dog Days', 'Comedy'),
('Identity Thief', 'Comedy'),
('The Best Man Holiday', 'Comedy'),
('Dumb and Dumber To', 'Comedy'),
('Spring Breakers', 'Comedy'),
('Tammy', 'Comedy'),
('Second Act', 'Comedy'),
('Saving Mr. Banks', 'Comedy'),
('Horrible Bosses 2', 'Comedy'),
('St. Vincent', 'Comedy'),
('The Monuments Men', 'Comedy'),
('Into the Woods', 'Comedy'),
('Grown Ups 2', 'Comedy'),
('About Time', 'Comedy'),
('Don Jon', 'Comedy'),
('The Internship', 'Comedy'),
('Dear White People', 'Comedy'),
('Think Like a Man Too', 'Comedy'),
('A Haunted House', 'Comedy'),
('Magic Mike XXL', 'Comedy'),
('The Grand Budapest Hotel', 'Comedy'),
('Love the Coopers', 'Comedy'),
('Muppets Most Wanted', 'Comedy'),
```

```
('22 Jump Street', 'Comedy'),
('The Intern', 'Comedy'),
('Delivery Man', 'Comedy'),
('Enough Said', 'Comedy'),
('The Heat', 'Comedy'),
('Philomena', 'Comedy'),
('And So It Goes', 'Comedy'),
('Blockers', 'Comedy'),
('The Second Best Exotic Marigold Hotel', 'Comedy'),
('Me and Earl and the Dying Girl', 'Comedy'),
('Ted 2', 'Comedy'),
('Night at the Museum: Secret of the Tomb', 'Comedy'),
('The Boss', 'Comedy'),
('Game Night', 'Comedy'),
('Top Five', 'Comedy'),
('Mike and Dave Need Wedding Dates', 'Comedy'),
('A Haunted House 2', 'Comedy'),
('Pitch Perfect 2', 'Comedy'),
('Ride Along 2', 'Comedy'),
('Wish I Was Here', 'Comedy'),
('Hot Pursuit', 'Comedy'),
('A Hologram for the King', 'Comedy'),
('The Hundred-Foot Journey', 'Comedy'),
('Mortdecai', 'Comedy'),
('Monster Trucks', 'Comedy'),
('Crazy Rich Asians', 'Comedy'),
('Now You See Me 2', 'Comedy'),
('Trainwreck', 'Comedy'),
('The Diary of a Teenage Girl', 'Comedy'),
('Welcome to Marwen', 'Comedy'),
('American Ultra', 'Comedy'),
('Pan', 'Comedy'),
('Compadres', 'Comedy'),
('Fist Fight', 'Comedy'),
('Paul Blart: Mall Cop 2', 'Comedy'),
('Whiskey Tango Foxtrot', 'Comedy'),
('Girls Trip', 'Comedy'),
('Paper Towns', 'Comedy'),
('Ricki and the Flash', 'Comedy'),
('My Big Fat Greek Wedding 2', 'Comedy'),
('The Nice Guys', 'Comedy'),
('Dope', 'Comedy'),
('Krampus', 'Comedy'),
('Bajrangi Bhaijaan', 'Comedy'),
('Teenage Mutant Ninja Turtles: Out of the Shadows', 'Comedy'),
('Swiss Army Man', 'Comedy'),
('Florence Foster Jenkins', 'Comedy'),
('Free Fire', 'Comedy'),
('Neighbors 2: Sorority Rising', 'Comedy'),
('The Meddler', 'Comedy'),
('Can You Ever Forgive Me?', 'Comedy'),
('Kingsman: The Golden Circle', 'Comedy'),
('Bad Moms', 'Comedy'),
('Fifty Shades of Black', 'Comedy'),
('Hunt for the Wilderpeople', 'Comedy'),
('How to Be a Latin Lover', 'Comedy'),
('Rough Night', 'Comedy'),
('The Sisters Brothers', 'Comedy'),
('Middle School: The Worst Years of My Life', 'Comedy'),
('Mary Poppins Returns', 'Comedy'),
('Ant-Man and the Wasp', 'Comedy'),
('Love, Simon', 'Comedy'),
('Logan Lucky', 'Comedy'),
('The Big Sick', 'Comedy'),
('Deadpool 2', 'Comedy'),
```

```
('Goosebumps 2: Haunted Halloween', 'Comedy'),
('Diary of a Wimpy Kid: The Long Haul', 'Comedy'),
('I Feel Pretty', 'Comedy'),
('Green Book', 'Comedy'),
('A Simple Favor', 'Comedy'),
('Uncle Drew', 'Comedy'),
('Instant Family', 'Comedy'),
('Tangled', 'Animation/Comedy'),
('Toy Story 3', 'Animation/Comedy'),
('The Smurfs', 'Animation/Comedy'),
('The Boxtrolls', 'Animation/Comedy'),
('Hotel Transylvania', 'Animation/Comedy'),
('Mr. Peabody & Sherman', 'Animation/Comedy'),
('Shrek Forever After', 'Animation/Comedy'),
('Megamind', 'Animation/Comedy'),
('Frankenweenie', 'Animation/Comedy'),
('Rango', 'Animation/Comedy'),
('Cars 2', 'Animation/Comedy'),
('Yogi Bear', 'Animation/Comedy'),
('Despicable Me', 'Animation/Comedy'),
('Happy Feet Two', 'Animation/Comedy'),
('Hop', 'Animation/Comedy'),
('Arthur Christmas', 'Animation/Comedy'),
('Winnie the Pooh', 'Animation/Comedy'),
('Monsters University', 'Animation/Comedy'),
('Norm of the North', 'Animation/Comedy'),
('Alvin and the Chipmunks: Chipwrecked', 'Animation/Comedy'),
('Free Birds', 'Animation/Comedy'),
('ParaNorman', 'Animation/Comedy'),
('Ice Age: Continental Drift', 'Animation/Comedy'),
('Despicable Me 2', 'Animation/Comedy'),
('Sausage Party', 'Animation/Comedy'),
('Wreck-It Ralph', 'Animation/Comedy'),
('The Nut Job', 'Animation/Comedy'),
('Turbo', 'Animation/Comedy'),
('Penguins of Madagascar', 'Animation/Comedy'),
('The Angry Birds Movie', 'Animation/Comedy'),
('Cloudy with a Chance of Meatballs 2', 'Animation/Comedy'),
('The Smurfs 2', 'Animation/Comedy'),
('Captain Underpants: The First Epic Movie', 'Animation/Comedy'),
('Finding Dory', 'Animation/Comedy'),
('The SpongeBob Movie: Sponge Out of Water', 'Animation/Comedy'),
('Minions', 'Animation/Comedy'),
('Sherlock Gnomes', 'Animation/Comedy'),
('Rio 2', 'Animation/Comedy'),
('Smurfs: The Lost Village', 'Animation/Comedy'),
('Hotel Transylvania 2', 'Animation/Comedy'),
('The Secret Life of Pets', 'Animation/Comedy'),
('Rock Dog', 'Animation/Comedy'),
('Zootopia', 'Animation/Comedy'),
('Alvin and the Chipmunks: The Road Chip', 'Animation/Comedy'),
('Ice Age: Collision Course', 'Animation/Comedy'),
('Despicable Me 3', 'Animation/Comedy'),
('The Nut Job 2: Nutty by Nature', 'Animation/Comedy'),
('Cars 3', 'Animation/Comedy'),
('The Boss Baby', 'Animation/Comedy'),
('The Lego Batman Movie', 'Animation/Comedy'),
('The Star', 'Animation/Comedy'),
('Storks', 'Animation/Comedy'),
('Early Man', 'Animation/Comedy'),
('The Emoji Movie', 'Animation/Comedy'),
('Peter Rabbit', 'Animation/Comedy'),
('Hotel Transylvania 3: Summer Vacation', 'Animation/Comedy'),
('Smallfoot', 'Animation/Comedy'),
('A Walk Among the Tombstones', 'Action'),
```

```
('Jurassic World', 'Action'),
('John Carter', 'Action'),
('The A-Team', 'Action'),
('Real Steel', 'Action'),
('The Equalizer', 'Action'),
('Captain America: The First Avenger', 'Action'),
('Tower Heist', 'Action'),
('The Mechanic', 'Action'),
('Ant-Man', 'Action'),
('Season of the Witch', 'Action'),
('Red Tails', 'Action'),
('Man of Steel', 'Action'),
('Jack Reacher', 'Action'),
('R.I.P.D.', 'Action'),
('Thor', 'Action'),
('Warcraft', 'Action'),
('World War Z', 'Action'),
('Priest', 'Action'),
('Dracula Untold', 'Action'),
('The Legend of Tarzan', 'Action'),
('The Last Airbender', 'Action'),
('Source Code', 'Action'),
('Green Zone', 'Action'),
('The Amazing Spider-Man', 'Action'),
('Sucker Punch', 'Action'),
('Machete', 'Action'),
('The Green Hornet', 'Action'),
('Hanna', 'Action'),
('Dylan Dog: Dead of Night', 'Action'),
('The Book of Eli', 'Action'),
('The Bounty Hunter', 'Action'),
('The Legend of Hercules', 'Action'),
('Repo Men', 'Action'),
('Ghost Rider: Spirit of Vengeance', 'Action'),
('Skyfall', 'Action'),
('Jonah Hex', 'Action'),
('Lone Survivor', 'Action'),
('Seventh Son', 'Action'),
('Free State of Jones', 'Action'),
('Green Lantern', 'Action'),
('Takers', 'Action'),
('The Bourne Legacy', 'Action'),
('Kites', 'Action'),
('Jack Ryan: Shadow Recruit', 'Action'),
('The Lone Ranger', 'Action'),
('Escape Plan', 'Action'),
('Resident Evil: Afterlife', 'Action'),
('Iron Man 2', 'Action'),
('21 Jump Street', 'Action'),
('The Tourist', 'Action'),
('Kick-Ass', 'Action'),
('300: Rise of an Empire', 'Action'),
('Immortals', 'Action'),
('The Man with the Iron Fists', 'Action'),
('Den of Thieves', 'Action'),
('X-Men: First Class', 'Action'),
('2 Guns', 'Action'),
('The Spy Next Door', 'Action'),
('Looper', 'Action'),
('Cats & Dogs: The Revenge of Kitty Galore', 'Action'),
('The Watch', 'Action'),
('Pirates of the Caribbean: On Stranger Tides', 'Action'),
('Iron Man 3', 'Action'),
('Bullet to the Head', 'Action'),
('Rise of the Planet of the Apes', 'Action'),
```

```
('The Expendables', 'Action'),
('Gangster Squad', 'Action'),
('47 Ronin', 'Action'),
('Dredd', 'Action'),
('The Dark Knight Rises', 'Action'),
('Tomb Raider', 'Action'),
('The Cold Light of Day', 'Action'),
('Cloud Atlas', 'Action'),
('Pride and Prejudice and Zombies', 'Action'),
('Inception', 'Action'),
('Cop Out', 'Action'),
('The Other Guys', 'Action'),
('Suicide Squad', 'Action'),
('The Hunger Games', 'Action'),
('Mad Max: Fury Road', 'Action'),
('Taken 2', 'Action'),
('Journey 2: The Mysterious Island', 'Action'),
('Transformers: Dark of the Moon', 'Action'),
('Star Trek Into Darkness', 'Action'),
('Riddick', 'Action'),
('12 Strong', 'Action'),
('I, Frankenstein', 'Action'),
('Predators', 'Action'),
('The Wolverine', 'Action'),
('Brick Mansions', 'Action'),
('Deadpool', 'Action'),
('Faster', 'Action'),
('Battleship', 'Action'),
('Killer Elite', 'Action'),
('The Next Three Days', 'Action'),
('Big Mommas: Like Father, Like Son', 'Action'),
('Baywatch', 'Action'),
('Aquaman', 'Action'),
('Attack the Block', 'Action'),
('Central Intelligence', 'Action'),
('Drive Angry', 'Action'),
('Haywire', 'Action'),
('Sherlock Holmes: A Game of Shadows', 'Action'),
('Contraband', 'Action'),
('Exodus: Gods and Kings', 'Action'),
('Elysium', 'Action'),
('The Mortal Instruments: City of Bones', 'Action'),
('Premium Rush', 'Action'),
('Skyline', 'Action'),
('Man on a Ledge', 'Action'),
('Mortal Engines', 'Action'),
('G.I. Joe: Retaliation', 'Action'),
('Maleficent', 'Action'),
('The Commuter', 'Action'),
('Act of Valor', 'Action'),
('Lockout', 'Action'),
('Fast Five', 'Action'),
('This Means War', 'Action'),
('Contagion', 'Action'),
('One for the Money', 'Action'),
('Safe House', 'Action'),
('A Good Day to Die Hard', 'Action'),
('Abraham Lincoln: Vampire Hunter', 'Action'),
('Jupiter Ascending', 'Action'),
('30 Minutes or Less', 'Action'),
('Independence Day: Resurgence', 'Action'),
('Edge of Tomorrow', 'Action'),
('Johnny English Reborn', 'Action'),
('In Time', 'Action'),
('Wrath of the Titans', 'Action'),
```

```
('Here Comes the Boom', 'Action'),
('The Dark Tower', 'Action'),
('Kick-Ass 2', 'Action'),
('Colombiana', 'Action'),
('Pacific Rim', 'Action'),
('Ready Player One', 'Action'),
('Snowpiercer', 'Action'),
('Alex Cross', 'Action'),
('Scouts Guide to the Zombie Apocalypse', 'Action'),
('Snow White and the Huntsman', 'Action'),
('The Expendables 2', 'Action'),
('Hands of Stone', 'Action'),
('Pirates of the Caribbean: Dead Men Tell No Tales', 'Action'),
('The Maze Runner', 'Action'),
('After Earth', 'Action'),
('RED 2', 'Action'),
('Chappie', 'Action'),
('Dhoom 3', 'Action'),
('Divergent', 'Action'),
('Captain America: The Winter Soldier', 'Action'),
('End of Watch', 'Action'),
('Resident Evil: Retribution', 'Action'),
('The Amazing Spider-Man 2', 'Action'),
('X-Men: Days of Future Past', 'Action'),
('Parker', 'Action'),
('Pompeii', 'Action'),
('The Hunger Games: Catching Fire', 'Action'),
('The Hunger Games: Mockingjay - Part 1', 'Action'),
('American Assassin', 'Action'),
('King Arthur: Legend of the Sword', 'Action'),
('Thor: The Dark World', 'Action'),
('Machete Kills', 'Action'),
('Guardians of the Galaxy', 'Action'),
('The Finest Hours', 'Action'),
('Dead Man Down', 'Action'),
('Dawn of the Planet of the Apes', 'Action'),
('Transformers: Age of Extinction', 'Action'),
('Collide', 'Action'),
('San Andreas', 'Action'),
('Jane Got a Gun', 'Action'),
('3 Days to Kill', 'Action'),
('Run All Night', 'Action'),
('Valerian and the City of a Thousand Planets', 'Action'),
('Spider-Man: Homecoming', 'Action'),
('22 Jump Street', 'Action'),
('Olympus Has Fallen', 'Action'),
('The 5th Wave', 'Action'),
('The Expendables 3', 'Action'),
('White House Down', 'Action'),
('The Rover', 'Action'),
('Need for Speed', 'Action'),
('Avengers: Age of Ultron', 'Action'),
('The November Man', 'Action'),
('Gods of Egypt', 'Action'),
('The Heat', 'Action'),
('Atomic Blonde', 'Action'),
('Taken 3', 'Action'),
('The Gunman', 'Action'),
('Hell or High Water', 'Action'),
('Resident Evil: The Final Chapter', 'Action'),
('Midnight Special', 'Action'),
('Star Trek Beyond', 'Action'),
('Hitman: Agent 47', 'Action'),
('Game Night', 'Action'),
('Blackhat', 'Action'),
```

```
('Furious 7', 'Action'),
('Ride Along 2', 'Action'),
('Red Sparrow', 'Action'),
('John Wick', 'Action'),
('The Transporter Refueled', 'Action'),
('Hot Pursuit', 'Action'),
('The Purge: Anarchy', 'Action'),
('Batman v Superman: Dawn of Justice', 'Action'),
('Mortdecai', 'Action'),
('Hardcore Henry', 'Action'),
('Monster Trucks', 'Action'),
('Now You See Me 2', 'Action'),
('London Has Fallen', 'Action'),
('American Ultra', 'Action'),
('Compadres', 'Action'),
('Transformers: The Last Knight', 'Action'),
('X-Men: Apocalypse', 'Action'),
('Paul Blart: Mall Cop 2', 'Action'),
('War for the Planet of the Apes', 'Action'),
('Captain America: Civil War', 'Action'),
('Mechanic: Resurrection', 'Action'),
('Underworld: Blood Wars', 'Action'),
('Kong: Skull Island', 'Action'),
('The Equalizer 2', 'Action'),
('Solo: A Star Wars Story', 'Action'),
('The Nice Guys', 'Action'),
('Only the Brave', 'Action'),
('Bajrangi Bhaijaan', 'Action'),
('Baby Driver', 'Action'),
('Teenage Mutant Ninja Turtles: Out of the Shadows', 'Action'),
('Maze Runner: The Scorch Trials', 'Action'),
('The Darkest Minds', 'Action'),
('The Purge: Election Year', 'Action'),
('Avengers: Infinity War', 'Action'),
('Free Fire', 'Action'),
('Jason Bourne', 'Action'),
('Mile 22', 'Action'),
('The Fate of the Furious', 'Action'),
('Kingsman: The Golden Circle', 'Action'),
('Bumblebee', 'Action'),
('The Meg', 'Action'),
('Baahubali 2: The Conclusion', 'Action'),
('Jurassic World: Fallen Kingdom', 'Action'),
('Dunkirk', 'Action'),
('Sicario: Day of the Soldado', 'Action'),
('Dangal', 'Action'),
('Ant-Man and the Wasp', 'Action'),
('The Hurricane Heist', 'Action'),
('Deadpool 2', 'Action'),
('Skyscraper', 'Action'),
('Hotel Artemis', 'Action'),
('The First Purge', 'Action'),
('Proud Mary', 'Action'),
('Upgrade', 'Action'),
('Peppermint', 'Action'),
('Destroyer', 'Action'),
('Puss in Boots', 'Animated/Action'),
('The Croods', 'Animated/Action'),
('How to Train Your Dragon', 'Animated/Action'),
('The Adventures of Tintin', 'Animated/Action'),
('Megamind', 'Animated/Action'),
('Kung Fu Panda 2', 'Animated/Action'),
('Rise of the Guardians', 'Animated/Action'),
('The Lego Movie', 'Animated/Action'),
('How to Train Your Dragon 2', 'Animated/Action'),
```

```
('The Angry Birds Movie', 'Animated/Action'),
('Captain Underpants: The First Epic Movie', 'Animated/Action'),
('Kung Fu Panda 3', 'Animated/Action'),
('The Lego Ninjago Movie', 'Animated/Action'),
('Incredibles 2', 'Animated/Action'),
('The Lego Batman Movie', 'Animated/Action'),
('Kubo and the Two Strings', 'Animated/Action'))
```

In [962…

```
# Create DataFrame with cleaned genres to be merged.
df_merge_genres = pd.DataFrame(list(all_genres_tup), columns=['movie_CH', 'genres'])

# Confirm results.
df_merge_genres
```

Out[962…

	movie_CH	genres
0	The Three Stooges	Family
1	Real Steel	Family
2	Furry Vengeance	Family
3	Ramona and Beezus	Family
4	Percy Jackson & the Olympians: The Lightning T...	Family
...	...	...
602	Kung Fu Panda 3	Animated/Action
603	The Lego Ninjago Movie	Animated/Action
604	Incredibles 2	Animated/Action
605	The Lego Batman Movie	Animated/Action
606	Kubo and the Two Strings	Animated/Action

607 rows × 2 columns

Merge DataFrames For Visualizations

In [963…

```
# Merge DataFrames for boxplot.
fnl_merged_df = pd.merge(clean_genres_df, df_merge_genres, on = 'movie_CH', how = 'oute

# Drop NaN.
fnl_merged_df.dropna(axis=0, how = 'any', inplace=True)

# Confirm we still have 607 movies.
fnl_merged_df
```

Out[963…

	movie_CH	production_budget_CH	domestic_gross_CH	worldwide_gross_CH	international_gross	r
0	The Secret Life of Walter Mitty	91.0	58.0	187.0	129.0	
1	A Walk Among the Tombstones	28.0	26.0	62.0	36.0	

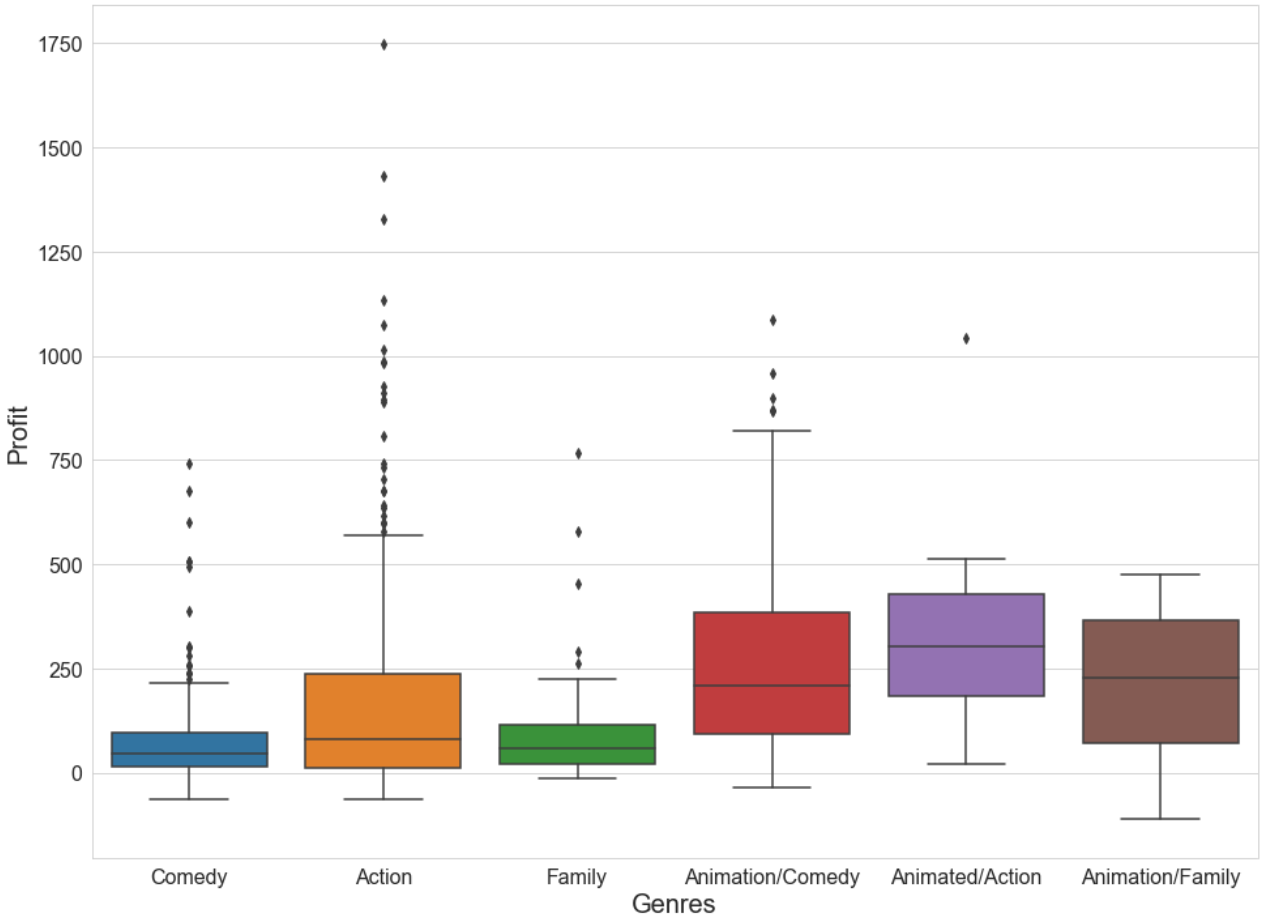
	movie_CH	production_budget_CH	domestic_gross_CH	worldwide_gross_CH	international_gross	r
2	Jurassic World	215.0	652.0	1648.0	996.0	
3	The Rum Diary	45.0	13.0	21.0	8.0	
4	The Three Stooges	30.0	44.0	54.0	10.0	
...	...	...	...	...	...	
827	Green Book	23.0	85.0	322.0	237.0	
828	A Simple Favor	20.0	53.0	97.0	44.0	
830	Destroyer	9.0	1.0	3.0	2.0	
832	Uncle Drew	18.0	42.0	46.0	4.0	
835	Instant Family	48.0	67.0	119.0	52.0	

607 rows × 15 columns

Create Boxplot

In [964…

```
fnl_merged_df = fnl_merged_df.rename(columns={'genres_y':'Genres', 'profit':'Profit'})

fig, ax = plt.subplots(figsize=(16,12))

g = sns.boxplot(data=fnl_merged_df, x = 'Genres', y='Profit')
```



Refine Boxplot for Presentation

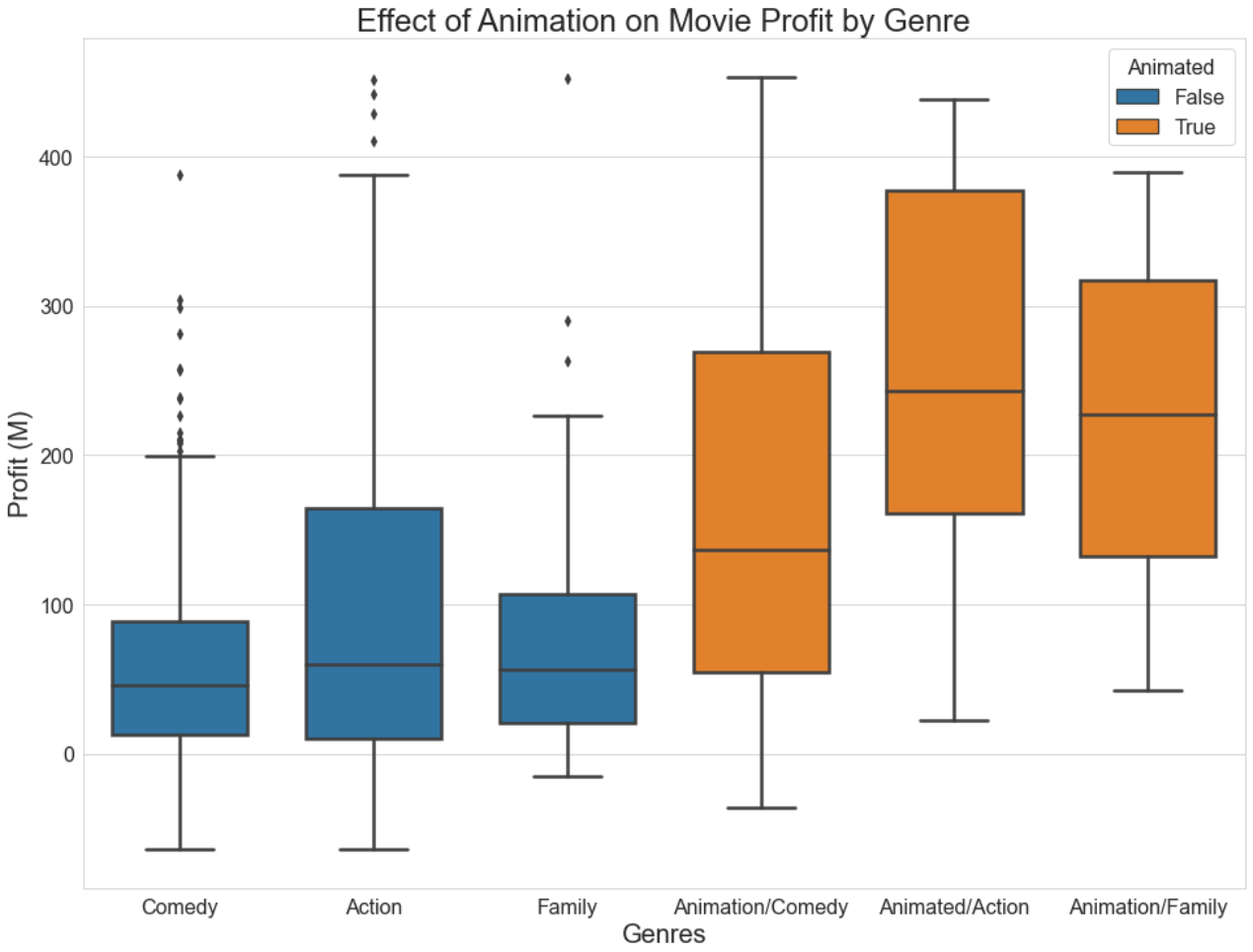
In [965…
```
# Define function to determine outliers
def outliers(df, col):
    Q1= df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    upper_limit = Q3 + 1.5 * IQR
    lower_limit = Q1 - 1.5 * IQR
    return upper_limit, lower_limit

# Apply outlier function to 'Profit' data
upper, lower = outliers(fnl_merged_df, "Profit")
print("Upper whisker: ", upper)
print("Lower Whisker: ", lower)
```

```
Upper whisker:  456.25
Lower Whisker:  -245.75
```

In [966…
```
# Filter outliers with function. Upper outliers are so high that none of the lower outl
# so I manually adjust lower whisker to -100M 'Profit'
no_outlier_df = fnl_merged_df[(fnl_merged_df['Profit'] > -100) & (fnl_merged_df['Profit
```

In [967…
```
# Set up Boxplot dimensions
plt.rc('font', size=16)
plt.rc('axes', titlesize=24)
plt.rc('axes', labelsize=20)
plt.rc('xtick', labelsize=16)
plt.rc('ytick', labelsize=16)
```

```
plt.rc('legend', fontsize=16)

# Create Boxplot
fig, ax = plt.subplots(figsize=(16,12))

g = sns.boxplot(data=no_outlier_df, x = 'Genres', y='Profit', width=.7, linewidth=2.5,
ax.set_ylabel("Profit (M)")
plt.legend(title = "Animated")
plt.title('Effect of Animation on Movie Profit by Genre');
```



Recall data for possible use in presentation

In [968…

```
print(f'You can expect roughly {round(avg_animated_profit_increase, 2)} times more prof
print(f'This comes to an average of ${round(avg_increase_per_film)}M more profit per fi
```

```
You can expect roughly 2.2 times more profit when animating (120% increase).
This comes to an average of $160M more profit per film.
```

Bonus Stats After Instructor Feedback on Slides

In [969…

```
# Check to make sure "fnl_merged_df" has 607 row and can still be used for calculations
print(f' DataFrame still has {len(fnl_merged_df.index)} entries.')

# Calculate % of negative profit non-animated films from our set.
red_films = len(fnl_merged_df[(fnl_merged_df['Profit'] < 0) & (fnl_merged_df['Animation
red_film_rate = red_films / total_movies
```

```
# Calculate % of negative profit animated films from our set.
a_red_films = len(fnl_merged_df[(fnl_merged_df['Profit'] < 0) & (fnl_merged_df['Animati
a_red_film_rate = a_red_films / total_animated_movies

# Print results
print(f' Non-Animated films generate a loss rate of {round(red_film_rate,3)} or 12.1%')
print(f' Animated films generate a loss rate of {round(a_red_film_rate,3)} or 3.8%')
print(f' Non-Animated films are more than 3x more likely to generate a loss!')
print(f' The average Non-Animated profit is ${round(film_avg_profit)}M.')
print(f' The average Animated profit is ${round(a_film_avg_profit)}M.')
```

```
DataFrame still has 607 entries.
Non-Animated films generate a loss rate of 0.121 or 12.1%
Animated films generate a loss rate of 0.038 or 3.8%
Non-Animated films are more than 3x more likely to generate a loss!
The average Non-Animated profit is $134M.
The average Animated profit is $294M.
```

 example

Project Title

Authors: Chris Helmerson, Nick Kennedy, Samira Chatrathi

Overview

In this project one-paragraph overview of the project, including the business problem, data, methods, results and recommendations.

Business Problem

Summary of the business problem you are trying to solve, and the data questions that you plan to answer to solve them.

Questions to consider:

- What are the business's pain points related to this project?
- How did you pick the data analysis question(s) that you did?
- Why are these questions important from a business perspective?

Data Understanding

Describe the data being used for this project.

Questions to consider:

- Where did the data come from, and how do they relate to the data analysis questions?
- What do the data represent? Who is in the sample and what variables are included?

- What is the target variable?
- What are the properties of the variables you intend to use?

In [4]:
```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [ ]:

In [ ]:

In [ ]:

Data Preparation

Describe and justify the process for preparing the data for analysis.

Questions to consider:

- Were there variables you dropped or created?
- How did you address missing values or outliers?
- Why are these choices appropriate given the data and the business problem?

In [ ]:

Data Modeling

Describe and justify the process for analyzing or modeling the data.

Questions to consider:

- How did you analyze or model the data?
- How did you iterate on your initial approach to make it better?
- Why are these choices appropriate given the data and the business problem?

In [ ]:

In [10]:
```
dfchris = pd.read_csv('Merged_DataFrame_CH.csv')
dfchris
```

Out[10]:

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int
0	0	1	18-Dec-09	Avatar	425	760	2776	
1	1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	2	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
3	3	7	27-Apr-18	Avengers: Infinity War	300	678	2048	
4	4	9	17-Nov-17	Justice League	300	229	655	
...	...	...	...	...	...	...	...	
1560	1560	33	19-Oct-12	The Sessions	1	6	11	
1561	1561	42	21-Oct-11	Martha Marcy May Marlene	1	2	5	
1562	1562	52	7-Dec-01	No Man's Land	1	1	2	
1563	1563	52	7-Dec-01	No Man's Land	1	1	2	
1564	1564	52	7-Dec-01	No Man's Land	1	1	2	

1565 rows × 9 columns

In [11]:
```
dfnick = pd.read_csv('final_data_nk.csv')
dfnick
```

Out[11]:

	Unnamed: 0	Unnamed: 0.1	clean_title	movie_id	primary_title	original_title	start_year	runtime_mi
0	0	20	foodfight!	tt0249516	Foodfight!	Foodfight!	2012	
1	1	33	mortal kombat	tt0293429	Mortal Kombat	Mortal Kombat	2021	

	Unnamed: 0	Unnamed: 0.1	clean_title	movie_id	primary_title	original_title	start_year	runtime_mi
2	2	40	the overnight	tt0326592	The Overnight	The Overnight	2010	
3	3	48	on the road	tt0337692	On the Road	On the Road	2012	
4	4	54	the secret life of walter mitty	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	
...	...	...	...	...	...	...	...	
3892	3892	145843	trapped	tt9877596	Trapped	Trapped	2016	
3893	3893	145937	the promise	tt9889072	The Promise	The Promise	2017	
3894	3894	145986	sublime	tt9893078	Sublime	Sublime	2019	
3895	3895	146025	columbus	tt9899880	Columbus	Columbus	2018	
3896	3896	146078	unstoppable	tt9906218	Unstoppable	Unstoppable	2019	

3897 rows × 44 columns

In [12]:
```
dfnick.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 44 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          3897 non-null   int64  
 1   Unnamed: 0.1        3897 non-null   int64  
 2   clean_title         3897 non-null   object 
 3   movie_id            3897 non-null   object 
 4   primary_title       3897 non-null   object 
 5   original_title      3896 non-null   object 
 6   start_year          3897 non-null   int64  
 7   runtime_minutes     3401 non-null   float64
 8   genres              3824 non-null   object 
 9   id                  3897 non-null   int64  
 10  release_date        3897 non-null   object 
 11  movie               3897 non-null   object 
 12  production_budget   3897 non-null   int64  
 13  domestic_gross      3897 non-null   int64  
 14  worldwide_gross     3897 non-null   int64  
 15  foreign_gross       3897 non-null   int64  
 16  title               1666 non-null   object 
 17  studio              1666 non-null   object 
 18  domestic_grossbom   1665 non-null   float64
 19  foreign_grossbom    1426 non-null   float64
 20  year                1666 non-null   float64
 21  worldwide_grossbom  1425 non-null   float64
 22  Sport               3824 non-null   object 
 23  Crime               3824 non-null   object 
 24  News                3824 non-null   object 
 25  Romance             3824 non-null   object 
 26  Sci-Fi              3824 non-null   object 
 27  Western             3824 non-null   object 
 28  Horror              3824 non-null   object 
```

```
 29  Drama              3824 non-null   object
 30  Family             3824 non-null   object
 31  nan                3824 non-null   object
 32  Mystery            3824 non-null   object
 33  Musical            3824 non-null   object
 34  Fantasy            3824 non-null   object
 35  Adventure          3824 non-null   object
 36  Documentary        3824 non-null   object
 37  Thriller           3824 non-null   object
 38  Comedy             3824 non-null   object
 39  Reality-TV         3824 non-null   object
 40  Music              3824 non-null   object
 41  History            3824 non-null   object
 42  Action             3824 non-null   object
 43  Animation          3824 non-null   object
dtypes: float64(5), int64(8), object(31)
memory usage: 1.3+ MB
```

In [13]:
```
dfchris
```

Out[13]:

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int
0	0	1	18-Dec-09	Avatar	425	760	2776	
1	1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	2	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
3	3	7	27-Apr-18	Avengers: Infinity War	300	678	2048	
4	4	9	17-Nov-17	Justice League	300	229	655	
...	...	...	...	...	...	...	...	
1560	1560	33	19-Oct-12	The Sessions	1	6	11	
1561	1561	42	21-Oct-11	Martha Marcy May Marlene	1	2	5	
1562	1562	52	7-Dec-01	No Man's Land	1	1	2	
1563	1563	52	7-Dec-01	No Man's Land	1	1	2	

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int
1564	1564	52	7-Dec-01	No Man's Land	1	1	2	

1565 rows × 9 columns

In [14]:
```
dfchris['release_date']= pd.to_datetime(dfchris['release_date'])
dfchris['release_date']
```

Out[14]:
```
0      2009-12-18
1      2011-05-20
2      2015-05-01
3      2018-04-27
4      2017-11-17
          ...
1560   2012-10-19
1561   2011-10-21
1562   2001-12-07
1563   2001-12-07
1564   2001-12-07
Name: release_date, Length: 1565, dtype: datetime64[ns]
```

In [15]:
```
import datetime
```

In [16]:
```
dfchris['release_date'] = pd.DatetimeIndex(dfchris['release_date']).year
```

In [17]:
```
dfchris
```

Out[17]:

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int
0	0	1	2009	Avatar	425	760	2776	
1	1	2	2011	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	2	4	2015	Avengers: Age of Ultron	330	459	1403	
3	3	7	2018	Avengers: Infinity War	300	678	2048	
4	4	9	2017	Justice League	300	229	655	
...	...	...	...	...	...	...	...	

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	int
1560	1560	33	2012	The Sessions	1	6	11	
1561	1561	42	2011	Martha Marcy May Marlene	1	2	5	
1562	1562	52	2001	No Man's Land	1	1	2	
1563	1563	52	2001	No Man's Land	1	1	2	
1564	1564	52	2001	No Man's Land	1	1	2	

1565 rows × 9 columns

In [18]:
```
dfnick.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 44 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          3897 non-null   int64
 1   Unnamed: 0.1        3897 non-null   int64
 2   clean_title         3897 non-null   object
 3   movie_id            3897 non-null   object
 4   primary_title       3897 non-null   object
 5   original_title      3896 non-null   object
 6   start_year          3897 non-null   int64
 7   runtime_minutes     3401 non-null   float64
 8   genres              3824 non-null   object
 9   id                  3897 non-null   int64
 10  release_date        3897 non-null   object
 11  movie               3897 non-null   object
 12  production_budget   3897 non-null   int64
 13  domestic_gross      3897 non-null   int64
 14  worldwide_gross     3897 non-null   int64
 15  foreign_gross       3897 non-null   int64
 16  title               1666 non-null   object
 17  studio              1666 non-null   object
 18  domestic_grossbom   1665 non-null   float64
 19  foreign_grossbom    1426 non-null   float64
 20  year                1666 non-null   float64
 21  worldwide_grossbom  1425 non-null   float64
 22  Sport               3824 non-null   object
 23  Crime               3824 non-null   object
 24  News                3824 non-null   object
 25  Romance             3824 non-null   object
 26  Sci-Fi              3824 non-null   object
 27  Western             3824 non-null   object
 28  Horror              3824 non-null   object
 29  Drama               3824 non-null   object
```

```
 30  Family              3824 non-null   object
 31  nan                 3824 non-null   object
 32  Mystery             3824 non-null   object
 33  Musical             3824 non-null   object
 34  Fantasy             3824 non-null   object
 35  Adventure           3824 non-null   object
 36  Documentary         3824 non-null   object
 37  Thriller            3824 non-null   object
 38  Comedy              3824 non-null   object
 39  Reality-TV          3824 non-null   object
 40  Music               3824 non-null   object
 41  History             3824 non-null   object
 42  Action              3824 non-null   object
 43  Animation           3824 non-null   object
dtypes: float64(5), int64(8), object(31)
memory usage: 1.3+ MB
```

In [19]:
```
dfnick['release_date']
```

Out[19]:
```
0       31-Dec-12
1       18-Aug-95
2       19-Jun-15
3       22-Mar-13
4       25-Dec-13
          ...
3892    20-Sep-02
3893    21-Apr-17
3894    13-Mar-07
3895     4-Aug-17
3896    12-Nov-10
Name: release_date, Length: 3897, dtype: object
```

In [20]:
```
dfnick['release_date']= pd.to_datetime(dfnick['release_date'])
dfnick['release_date']
```

Out[20]:
```
0      2012-12-31
1      1995-08-18
2      2015-06-19
3      2013-03-22
4      2013-12-25
          ...
3892   2002-09-20
3893   2017-04-21
3894   2007-03-13
3895   2017-08-04
3896   2010-11-12
Name: release_date, Length: 3897, dtype: datetime64[ns]
```

In [21]:
```
import datetime
dfnick['release_date'] = pd.DatetimeIndex(dfnick['release_date']).year
dfnick['release_date']
```

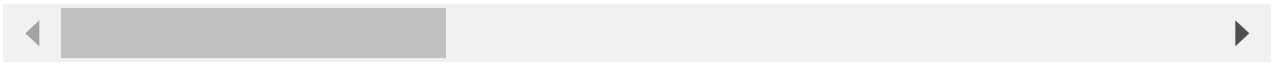
Out[21]:
```
0       2012
1       1995
2       2015
3       2013
4       2013
        ...
3892    2002
3893    2017
3894    2007
3895    2017
3896    2010
Name: release_date, Length: 3897, dtype: int64
```

In [22]:
```
dfmerged = pd.read_csv('merged_all.csv')
dfmerged
```

Out[22]:

	Unnamed: 0	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
0	0	foodfight!	tt0249516	Foodfight!	Foodfight!	2012	91.0	Acti
1	1	mortal kombat	tt0293429	Mortal Kombat	Mortal Kombat	2021	NaN	Act
2	2	the overnight	tt0326592	The Overnight	The Overnight	2010	88.0	
3	3	on the road	tt0337692	On the Road	On the Road	2012	124.0	Adve
4	4	the secret life of walter mitty	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adv
...	...	...	...	...	...	...	...	
5373	3893	the promise	tt9889072	The Promise	The Promise	2017	NaN	
5374	3894	sublime	tt9893078	Sublime	Sublime	2019	NaN	
5375	3895	columbus	tt9899880	Columbus	Columbus	2018	85.0	
5376	3896	unstoppable	tt9906218	Unstoppable	Unstoppable	2019	84.0	
5377	3896	unstoppable	tt9906218	Unstoppable	Unstoppable	2019	84.0	

5378 rows × 52 columns



In [23]:
```
dfmerged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5378 entries, 0 to 5377
Data columns (total 52 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            5378 non-null   int64
 1   clean_title           5378 non-null   object
 2   movie_id              5378 non-null   object
 3   primary_title         5378 non-null   object
 4   original_title        5377 non-null   object
 5   start_year            5378 non-null   int64
 6   runtime_minutes       4682 non-null   float64
 7   genres                5279 non-null   object
 8   id                    5378 non-null   int64
 9   release_date          5378 non-null   object
 10  movie                 5378 non-null   object
 11  production_budget     5378 non-null   int64
 12  domestic_gross        5378 non-null   int64
 13  worldwide_gross       5378 non-null   int64
 14  foreign_gross         5378 non-null   int64
 15  title                 2148 non-null   object
```

```
 16  studio                2148 non-null   object
 17  domestic_grossbom     2147 non-null   float64
 18  foreign_grossbom      1874 non-null   float64
 19  year                  2148 non-null   float64
 20  worldwide_grossbom    1873 non-null   float64
 21  Sport                 5279 non-null   object
 22  Crime                 5279 non-null   object
 23  News                  5279 non-null   object
 24  Romance               5279 non-null   object
 25  Sci-Fi                5279 non-null   object
 26  Western               5279 non-null   object
 27  Horror                5279 non-null   object
 28  Drama                 5279 non-null   object
 29  Family                5279 non-null   object
 30  nan                   5279 non-null   object
 31  Mystery               5279 non-null   object
 32  Musical               5279 non-null   object
 33  Fantasy               5279 non-null   object
 34  Adventure             5279 non-null   object
 35  Documentary           5279 non-null   object
 36  Thriller              5279 non-null   object
 37  Comedy                5279 non-null   object
 38  Reality-TV            5279 non-null   object
 39  Music                 5279 non-null   object
 40  History               5279 non-null   object
 41  Action                5279 non-null   object
 42  Animation             5279 non-null   object
 43  averagerating         3797 non-null   float64
 44  numvotes              3797 non-null   float64
 45  release_date_CH       3416 non-null   object
 46  movie_CH              3416 non-null   object
 47  production_budget_CH  3416 non-null   float64
 48  domestic_gross_CH     3416 non-null   float64
 49  worldwide_gross_CH    3416 non-null   float64
 50  international_gross   3416 non-null   float64
 51  genre_ids             3416 non-null   object
dtypes: float64(11), int64(7), object(34)
memory usage: 2.1+ MB
```

In [24]:

```
#Let's load our cleaned data frame and looking at the rows and columns

import pandas as pd
masterdf = pd.read_csv('master_data_raw.csv')
masterdf
```

Out[24]:

	Unnamed: 0	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
0	0	foodfight!	tt0249516	Foodfight!	Foodfight!	2012	91.0	Acti
1	1	mortal kombat	tt0293429	Mortal Kombat	Mortal Kombat	2021	NaN	Act
2	2	the overnight	tt0326592	The Overnight	The Overnight	2010	88.0	
3	3	on the road	tt0337692	On the Road	On the Road	2012	124.0	Adve
4	4	the secret life of walter mitty	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adv
...	...	...	...	...	...	...	...	

	Unnamed: 0	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes
3892	3892	trapped	tt9877596	Trapped	Trapped	2016	NaN
3893	3893	the promise	tt9889072	The Promise	The Promise	2017	NaN
3894	3894	sublime	tt9893078	Sublime	Sublime	2019	NaN
3895	3895	columbus	tt9899880	Columbus	Columbus	2018	85.0
3896	3896	unstoppable	tt9906218	Unstoppable	Unstoppable	2019	84.0

3897 rows × 52 columns

In [25]:
```
masterdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 52 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            3897 non-null   int64  
 1   clean_title           3897 non-null   object 
 2   movie_id              3897 non-null   object 
 3   primary_title         3897 non-null   object 
 4   original_title        3896 non-null   object 
 5   start_year            3897 non-null   int64  
 6   runtime_minutes       3401 non-null   float64
 7   genres                3824 non-null   object 
 8   id                    3897 non-null   int64  
 9   release_date          3897 non-null   object 
 10  movie                 3897 non-null   object 
 11  production_budget     3897 non-null   int64  
 12  domestic_gross        3897 non-null   int64  
 13  worldwide_gross       3897 non-null   int64  
 14  foreign_gross         3897 non-null   int64  
 15  title                 1666 non-null   object 
 16  studio                1666 non-null   object 
 17  domestic_grossbom     1665 non-null   float64
 18  foreign_grossbom      1426 non-null   float64
 19  year                  1666 non-null   float64
 20  worldwide_grossbom    1425 non-null   float64
 21  Sport                 3824 non-null   object 
 22  Crime                 3824 non-null   object 
 23  News                  3824 non-null   object 
 24  Romance               3824 non-null   object 
 25  Sci-Fi                3824 non-null   object 
 26  Western               3824 non-null   object 
 27  Horror                3824 non-null   object 
 28  Drama                 3824 non-null   object 
 29  Family                3824 non-null   object 
 30  nan                   3824 non-null   object 
 31  Mystery               3824 non-null   object 
 32  Musical               3824 non-null   object 
 33  Fantasy               3824 non-null   object 
 34  Adventure             3824 non-null   object 
 35  Documentary           3824 non-null   object 
 36  Thriller              3824 non-null   object 
 37  Comedy                3824 non-null   object 
 38  Reality-TV            3824 non-null   object 
```

```
 39  Music                 3824 non-null   object
 40  History               3824 non-null   object
 41  Action                3824 non-null   object
 42  Animation             3824 non-null   object
 43  averagerating         2938 non-null   float64
 44  numvotes              2938 non-null   float64
 45  release_date_CH       1309 non-null   object
 46  movie_CH              1309 non-null   object
 47  production_budget_CH  1309 non-null   float64
 48  domestic_gross_CH     1309 non-null   float64
 49  worldwide_gross_CH    1309 non-null   float64
 50  international_gross   1309 non-null   float64
 51  genre_ids             1309 non-null   object
dtypes: float64(11), int64(7), object(34)
memory usage: 1.5+ MB
```

In [26]:
```
#our selected release date is in object format...we want to change that to an integer
import datetime
masterdf['release_date'] = pd.DatetimeIndex(masterdf['release_date']).year
masterdf['release_date']
```

Out[26]:
```
0       2012
1       1995
2       2015
3       2013
4       2013
        ...
3892    2002
3893    2017
3894    2007
3895    2017
3896    2010
Name: release_date, Length: 3897, dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [27]:
```
#now we want to create our visualizaiton to be filtered with years between 2014 and 201

yr2014_yr2018 = masterdf[(masterdf['release_date'] >= 2014) & (masterdf['release_date']
xrange = yr2014_yr2018['release_date']
```

In [102…
```
xrange
```

Out[102…
```
2       2015
5       2014
6       2015
17      2014
21      2015
        ...
3883    2016
3885    2015
3890    2015
3893    2017
3895    2017
Name: release_date, Length: 1477, dtype: int64
```

In [28]:
```
yr2014_yr2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1477 entries, 2 to 3895
Data columns (total 52 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            1477 non-null   int64  
 1   clean_title           1477 non-null   object 
 2   movie_id              1477 non-null   object 
 3   primary_title         1477 non-null   object 
 4   original_title        1477 non-null   object 
 5   start_year            1477 non-null   int64  
 6   runtime_minutes       1356 non-null   float64
 7   genres                1457 non-null   object 
 8   id                    1477 non-null   int64  
 9   release_date          1477 non-null   int64  
 10  movie                 1477 non-null   object 
 11  production_budget     1477 non-null   int64  
 12  domestic_gross        1477 non-null   int64  
 13  worldwide_gross       1477 non-null   int64  
 14  foreign_gross         1477 non-null   int64  
 15  title                 865 non-null    object 
 16  studio                865 non-null    object 
 17  domestic_grossbom     865 non-null    float64
 18  foreign_grossbom      714 non-null    float64
 19  year                  865 non-null    float64
 20  worldwide_grossbom    714 non-null    float64
 21  Sport                 1457 non-null   object 
 22  Crime                 1457 non-null   object 
 23  News                  1457 non-null   object 
 24  Romance               1457 non-null   object 
 25  Sci-Fi                1457 non-null   object 
 26  Western               1457 non-null   object 
 27  Horror                1457 non-null   object 
 28  Drama                 1457 non-null   object 
 29  Family                1457 non-null   object 
 30  nan                   1457 non-null   object 
 31  Mystery               1457 non-null   object 
 32  Musical               1457 non-null   object 
 33  Fantasy               1457 non-null   object 
 34  Adventure             1457 non-null   object 
 35  Documentary           1457 non-null   object 
 36  Thriller              1457 non-null   object 
 37  Comedy                1457 non-null   object 
 38  Reality-TV            1457 non-null   object 
 39  Music                 1457 non-null   object 
 40  History               1457 non-null   object 
 41  Action                1457 non-null   object 
 42  Animation             1457 non-null   object 
 43  averagerating         1244 non-null   float64
 44  numvotes              1244 non-null   float64
 45  release_date_CH       586 non-null    object 
 46  movie_CH              586 non-null    object 
 47  production_budget_CH  586 non-null    float64
 48  domestic_gross_CH     586 non-null    float64
 49  worldwide_gross_CH    586 non-null    float64
 50  international_gross   586 non-null    float64
 51  genre_ids             586 non-null    object 
dtypes: float64(11), int64(8), object(33)
memory usage: 611.6+ KB
```

In [29]:
```
print(yr2014_yr2018['foreign_gross'])
```

```
2          56188
5       36090902
6      996584239
```

```
17       21449831
21         446787
          ...
3883     98883179
3885       208672
3890     18986015
3893      2327129
3895        93404
Name: foreign_gross, Length: 1477, dtype: int64
```

In [30]:
```
#mean for foreign and domestic gross

fogrssmn = yr2014_yr2018['foreign_gross'].sum()/5
dogrssmn = yr2014_yr2018['domestic_gross'].sum()/5

print(fogrssmn)
print(dogrssmn)
```

```
20393201726.4
13948292452.8
```

In [31]:
```
#median for foreign and domestic gross
import statistics

yr2014 = masterdf[(masterdf['release_date'] == 2014)]
yr2015 = masterdf[(masterdf['release_date'] == 2015)]
yr2016 = masterdf[(masterdf['release_date'] == 2016)]
yr2017 = masterdf[(masterdf['release_date'] == 2017)]
yr2018 = masterdf[(masterdf['release_date'] == 2018)]


yr2k14fs = yr2014['foreign_gross'].sum()
yr2k15fs = yr2015['foreign_gross'].sum()
yr2k16fs = yr2016['foreign_gross'].sum()
yr2k17fs = yr2017['foreign_gross'].sum()
yr2k18fs = yr2018['foreign_gross'].sum()

foreign_sums = [yr2k14fs, yr2k15fs, yr2k16fs, yr2k17fs, yr2k18fs]
foreign_median = statistics.median(foreign_sums)


yr2k14ds = yr2014['domestic_gross'].sum()
yr2k15ds = yr2015['domestic_gross'].sum()
yr2k16ds = yr2016['domestic_gross'].sum()
yr2k17ds = yr2017['domestic_gross'].sum()
yr2k18ds = yr2018['domestic_gross'].sum()

domestic_sums = [yr2k14ds, yr2k15ds, yr2k16ds, yr2k17ds, yr2k18ds]
domestic_median = statistics.median(domestic_sums)


print(foreign_median)
print(domestic_median)
```

```
18416412096
12736761698
```

In [32]:
```
#standard deviation for foreign and domestic gross

fogrssstd = statistics.stdev(foreign_sums)
dogrssstd = statistics.stdev(domestic_sums)
```

```
print(fogrssstd)
print(dogrssstd)
```

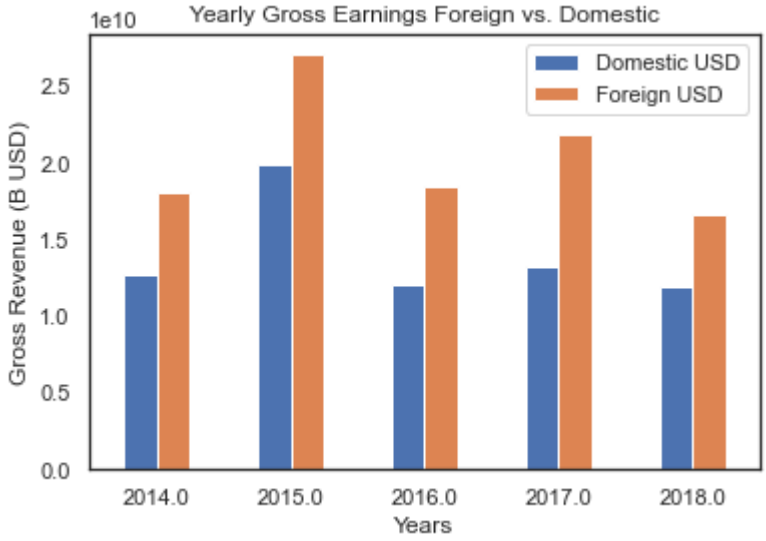
```
343105386.8836856
347979795.93352
/Users/samirachatrathi/opt/anaconda3/envs/learn-env/lib/python3.8/statistics.py:168: Run
timeWarning: overflow encountered in long_scalars
  partials[d] = partials_get(d, 0) + n
/Users/samirachatrathi/opt/anaconda3/envs/learn-env/lib/python3.8/fractions.py:420: Runt
imeWarning: overflow encountered in long_scalars
  return Fraction(a.numerator * db - b.numerator * da,
```

In [104…

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

labels = ['2014', '2015', '2016', '2017', '2018']


ax = masterdf.groupby(xrange)['domestic_gross', 'foreign_gross'].sum().sort_values('rel

ax.legend(['Domestic USD', 'Foreign USD'])

sns.set_style('white')
;
```

```
<ipython-input-104-0be25106f3ca>:8: FutureWarning: Indexing with multiple keys (implicit
ly converted to a tuple of keys) will be deprecated, use a list instead.
  ax = masterdf.groupby(xrange)['domestic_gross', 'foreign_gross'].sum().sort_values('re
lease_date').plot.bar(stacked=False, title = 'Yearly Gross Earnings Foreign vs. Domesti
c', ylabel = 'Gross Revenue (B USD)', xlabel = 'Years', rot=0)
''
```

Out[104…



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Evaluation

Evaluate how well your work solves the stated business problem.

Questions to consider:

- How do you interpret the results?
- How well does your model fit your data? How much better is this than your baseline model?
- How confident are you that your results would generalize beyond the data you have?
- How confident are you that this model would benefit the business if put into use?

In [ ]:
```
import cpi

cpi.update()
dom_grossfilt = cpi.inflate(masterdf['domestic_gross'], masterdf['release_date'], to =

for_grossfilt = cpi.inflate(masterdf['foreign_gross'], masterdf['release_date'], to= 20

dom_grossfilt()
```

Conclusions

Provide your conclusions about the work you've done, including any limitations or next steps.

Questions to consider:

- What would you recommend the business do as a result of this work?
- What are some reasons why your analysis might not fully solve the business problem?
- What else could you do in the future to improve this project?

NICK Kennedy's notebook

In [2]:
```
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
import sqlite3
%matplotlib inline
```

In [ ]:
```
#extract IMDB movie basics table from SQL database
```

In [13]:
```
!cd
```

```
C:\Users\Nick\Documents\Flatiron\phase_1\project\gitrepo\microsoft-movie-analysis-ncs
```

In [22]:
```
conn = sqlite3.connect('C:/Users/Nick/Documents/Flatiron/phase_1/project/data/im.db')
```

In [23]:
```
movie_basics_df = pd.read_sql("""
SELECT *
FROM movie_basics
""",conn )
```

In [24]:
```
movie_basics_df.head(2)
```

Out[24]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama

In [25]:
```
movie_basics_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   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
```

In [ ]:
```
#extract csvs to dataframes. bom, tn, tmdb
```

In [75]:
```
bom_raw_df = pd.read_csv('bom.movie_gross.csv')
bom_raw_df.head(2)
```

Out[75]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010

In [31]:
```
tn_raw_df = pd.read_csv('tn.movie_budgets.csv')
tn_raw_df.head(2)
```

Out[31]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875

In [ ]:
```
#Create clean title columns for matching 1) MOVIE_BASICS
```

In [32]:
```
movie_basics_df['clean_title'] = movie_basics_df['primary_title'].str.lower()
```

In [34]:
```
movie_basics_df['clean_title'] = movie_basics_df['clean_title'].str.replace("'","").rep
```

In [35]:
```
#move clean title to the front then rest index
movie_basics_df.set_index('clean_title', inplace = True)
```

In [37]:
```
movie_basics_df.reset_index(inplace = True)
movie_basics_df.head(2)
```

Out[37]:

	index	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	ge
0	0	sunghursh	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dr
1	1	one day before the rainy season	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dr

In [ ]:
```
#2) BOM RAW
```

In [76]:
```
bom_raw_df['clean_title'] = bom_raw_df['title'].str.lower()
```

In [77]:
```
bom_raw_df['clean_title'] = bom_raw_df['clean_title'].str.replace("'","").replace(',',
```

In [80]:
```
bom_raw_df.set_index('clean_title', inplace = True)
bom_raw_df.head(2)
```

Out[80]:

clean_title	title	studio	domestic_gross	foreign_gross	year
toy story 3	Toy Story 3	BV	415000000.0	652000000.0	2010
alice in wonderland (2010)	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010

In [47]:
```
#tn_budgets
```

In [50]:
```
tn_raw_df['clean_title'] = tn_raw_df['movie'].str.lower()
```

In [63]:
```
for title in tn_raw_df['clean_title']:
    title.replace("'","").replace(',', '').replace(':','').replace('.','').replace('-',
```

In [64]:
```
tn_raw_df.reset_index( inplace = True)
tn_raw_df.head(2)
```

Out[64]:

	level_0	index	clean_title	id	release_date	movie	production_budget	domestic_gross	worldwic
0	0	0	avatar	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776
1	1	1	pirates of the caribbean: on stranger tides	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045

In [61]:
```
bom_raw_df
```

Out[61]:

clean_title	title	studio	domestic_gross	foreign_gross	year
toy story 3	Toy Story 3	BV	415000000.0	652000000.0	2010
alice in wonderland (2010)	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010
harry potter and the deathly hallows part 1	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010
inception	Inception	WB	292600000.0	535700000.0	2010
shrek forever after	Shrek Forever After	P/DW	238700000.0	513900000.0	2010
...	...	...	...	...	...
the quake	The Quake	Magn.	6200.0	NaN	2018
edward ii (2018 re-release)	Edward II (2018 re-release)	FM	4800.0	NaN	2018
el pacto	El Pacto	Sony	2500.0	NaN	2018
the swan	The Swan	Synergetic	2400.0	NaN	2018
an actor prepares	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

In [65]:
```
#CLEAN title not eliminating punctuation rn so going to do it in excel
tn_raw_df.to_csv('tn_raw_clean_fix.csv')
#checking the others
bom_raw_df.to_csv('bom_raw_clean_fix.csv')
movie_basics_df.to_csv('movie_basics_clean_fix.csv')
```

In [ ]:
```
#that didn't work for the movie_basics because the file is too large so
```

In [ ]:
```
#engineer a foreign_gross for tn and worldwide_gross for bom
```

In [81]:
```
bom_raw_df['worldwide_gross'] = bom_raw_df['foreign_gross'] + bom_raw_df['domestic_gros
```

In [72]:
```
#having trouble fixing number formats so
tn_raw_df.to_csv('tn_numfix.csv')
```

In [73]:
```
tn_fixed_df = pd.read_csv('tn_numfix.csv')
tn_fixed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Unnamed: 0        5782 non-null   int64 
 1   level_0           5782 non-null   int64 
 2   index             5782 non-null   int64 
 3   clean_title       5782 non-null   object
```

```
 4   id                 5782 non-null   int64
 5   release_date       5782 non-null   object
 6   movie              5782 non-null   object
 7   production_budget  5782 non-null   int64
 8   domestic_gross     5782 non-null   int64
 9   worldwide_gross    5782 non-null   int64
dtypes: int64(7), object(3)
memory usage: 451.8+ KB
```

In [82]:
```
#now to engineer foreign gross
tn_raw_df['foreign_gross'] = tn_raw_df['worldwide_gross'] + tn_raw_df['domestic_gross']
```

NOW TO JOIN THEM and clean them

In [87]:
```
tn_raw_df.drop(['level_0', 'index'], axis = 1, inplace =True)
```

In [88]:
```
tn_raw_df.set_index('clean_title', inplace=True)
```

In [89]:
```
tn_mbasics_merge = movie_basics_df.join(tn_raw_df, on='clean_title')
tn_mbasics_merge.head(2)
```

Out[89]:

	index	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	ge
0	0	sunghursh	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dr
1	1	one day before the rainy season	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dr

In [93]:
```
tn_mbasics_merge.drop('index', axis=1, inplace=True)
```

In [90]:
```
#check the merge info
tn_mbasics_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146353 entries, 0 to 146143
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   index              146353 non-null  int64
 1   clean_title        146353 non-null  object
 2   movie_id           146353 non-null  object
 3   primary_title      146353 non-null  object
 4   original_title     146332 non-null  object
 5   start_year         146353 non-null  int64
 6   runtime_minutes    114581 non-null  float64
 7   genres             140940 non-null  object
 8   id                 3897 non-null    float64
 9   release_date       3897 non-null    object
 10  movie              3897 non-null    object
 11  production_budget  3897 non-null    object
 12  domestic_gross     3897 non-null    object
 13  worldwide_gross    3897 non-null    object
 14  foreign_gross      3897 non-null    object
dtypes: float64(2), int64(2), object(11)
memory usage: 17.9+ MB
```

In [96]:
```
#lets try merging this with bom before dropping nulls
all_merged_df = tn_mbasics_merge.join(bom_raw_df, on='clean_title', rsuffix = 'bom')
```

In [97]:
```
all_merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146355 entries, 0 to 146143
Data columns (total 20 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   clean_title         146355 non-null  object 
 1   movie_id            146355 non-null  object 
 2   primary_title       146355 non-null  object 
 3   original_title      146334 non-null  object 
 4   start_year          146355 non-null  int64  
 5   runtime_minutes     114583 non-null  float64
 6   genres              140942 non-null  object 
 7   id                  3897 non-null    float64
 8   release_date        3897 non-null    object 
 9   movie               3897 non-null    object 
 10  production_budget   3897 non-null    object 
 11  domestic_gross      3897 non-null    object 
 12  worldwide_gross     3897 non-null    object 
 13  foreign_gross       3897 non-null    object 
 14  title               3515 non-null    object 
 15  studio              3512 non-null    object 
 16  domestic_grossbom   3490 non-null    float64
 17  foreign_grossbom    2130 non-null    float64
 18  year                3515 non-null    float64
 19  worldwide_grossbom  2105 non-null    float64
dtypes: float64(6), int64(1), object(13)
memory usage: 23.4+ MB
```

In [101…
```
#now to drop NAs
all_merged_df.dropna(subset = ['release_date'], inplace =True)
```

In [102…
```
all_merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3897 entries, 20 to 146078
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   clean_title         3897 non-null   object 
 1   movie_id            3897 non-null   object 
 2   primary_title       3897 non-null   object 
 3   original_title      3896 non-null   object 
 4   start_year          3897 non-null   int64  
 5   runtime_minutes     3401 non-null   float64
 6   genres              3824 non-null   object 
 7   id                  3897 non-null   float64
 8   release_date        3897 non-null   object 
 9   movie               3897 non-null   object 
 10  production_budget   3897 non-null   object 
 11  domestic_gross      3897 non-null   object 
 12  worldwide_gross     3897 non-null   object 
 13  foreign_gross       3897 non-null   object 
 14  title               1666 non-null   object 
 15  studio              1666 non-null   object 
 16  domestic_grossbom   1665 non-null   float64
 17  foreign_grossbom    1426 non-null   float64
 18  year                1666 non-null   float64
 19  worldwide_grossbom  1425 non-null   float64
```

```
dtypes: float64(6), int64(1), object(13)
memory usage: 639.4+ KB
```

In [105…]
```
all_merged_df['foreign_gross']
```

Out[105…]
```
20                      $73,706$0
33         $122,133,227$70,433,227
40            $1,165,996$1,109,808
48              $9,313,302$720,828
54         $187,861,183$58,236,838
                    ...
145843       $6,916,869$6,916,869
145937      $10,551,417$8,224,288
145986                        $0$0
146025       $1,110,511$1,017,107
146078    $165,720,921$81,562,942
Name: foreign_gross, Length: 3897, dtype: object
```

In [106…]
```
#Looks like we need to take a closer look in excel and fix numbers
all_merged_df.to_csv('all_merged_fixup.csv')
```

In [107…]
```
final_merge_df = pd.read_csv('all_merged_fixup.csv')
```

In [108…]
```
final_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          3897 non-null   int64  
 1   clean_title         3897 non-null   object 
 2   movie_id            3897 non-null   object 
 3   primary_title       3897 non-null   object 
 4   original_title      3896 non-null   object 
 5   start_year          3897 non-null   int64  
 6   runtime_minutes     3401 non-null   float64
 7   genres              3824 non-null   object 
 8   id                  3897 non-null   int64  
 9   release_date        3897 non-null   object 
 10  movie               3897 non-null   object 
 11  production_budget   3897 non-null   int64  
 12  domestic_gross      3897 non-null   int64  
 13  worldwide_gross     3897 non-null   int64  
 14  foreign_gross       3897 non-null   int64  
 15  title               1666 non-null   object 
 16  studio              1666 non-null   object 
 17  domestic_grossbom   1665 non-null   float64
 18  foreign_grossbom    1426 non-null   float64
 19  year                1666 non-null   float64
 20  worldwide_grossbom  1425 non-null   float64
dtypes: float64(5), int64(7), object(9)
memory usage: 639.5+ KB
```

In [109…]
```
#Great now lets lets look at genres
genres_imdb_list= set(list([x for x in final_merge_df['genres']]))
```

In [110…]
```
genres_imdb_list = list(genres_imdb_list)
genres_working1 = []
for x in genres_imdb_list:
       genres_working1.append(str(x))
```

In [112…
```
genres_working2 = []
for x in genres_working1:
    for y in x.split(sep =','):
        genres_working2.append(y)
imdb_genres = list(set(genres_working2))
imdb_genres.pop(1)
imdb_genres.pop(-9)
imdb_genres
```

Out[112…
```
['Sport',
 'Crime',
 'News',
 'Romance',
 'Sci-Fi',
 'Western',
 'Horror',
 'Drama',
 'Family',
 'nan',
 'Mystery',
 'Musical',
 'Fantasy',
 'Adventure',
 'Documentary',
 'Thriller',
 'Comedy',
 'Reality-TV',
 'Music',
 'History',
 'Action',
 'Animation']
```

NOW TO BUILD BOOLEAN GENRE COLUMNS

In [113…
```
for x in imdb_genres:
    final_merge_df[x] = (final_merge_df['genres'].str.contains(x))
```

In [114…
```
final_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 43 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          3897 non-null   int64
 1   clean_title         3897 non-null   object
 2   movie_id            3897 non-null   object
 3   primary_title       3897 non-null   object
 4   original_title      3896 non-null   object
 5   start_year          3897 non-null   int64
 6   runtime_minutes     3401 non-null   float64
 7   genres              3824 non-null   object
 8   id                  3897 non-null   int64
 9   release_date        3897 non-null   object
 10  movie               3897 non-null   object
 11  production_budget   3897 non-null   int64
 12  domestic_gross      3897 non-null   int64
 13  worldwide_gross     3897 non-null   int64
 14  foreign_gross       3897 non-null   int64
 15  title               1666 non-null   object
 16  studio              1666 non-null   object
 17  domestic_grossbom   1665 non-null   float64
```

```
 18  foreign_grossbom    1426 non-null   float64
 19  year                1666 non-null   float64
 20  worldwide_grossbom  1425 non-null   float64
 21  Sport               3824 non-null   object
 22  Crime               3824 non-null   object
 23  News                3824 non-null   object
 24  Romance             3824 non-null   object
 25  Sci-Fi              3824 non-null   object
 26  Western             3824 non-null   object
 27  Horror              3824 non-null   object
 28  Drama               3824 non-null   object
 29  Family              3824 non-null   object
 30  nan                 3824 non-null   object
 31  Mystery             3824 non-null   object
 32  Musical             3824 non-null   object
 33  Fantasy             3824 non-null   object
 34  Adventure           3824 non-null   object
 35  Documentary         3824 non-null   object
 36  Thriller            3824 non-null   object
 37  Comedy              3824 non-null   object
 38  Reality-TV          3824 non-null   object
 39  Music               3824 non-null   object
 40  History             3824 non-null   object
 41  Action              3824 non-null   object
 42  Animation           3824 non-null   object
dtypes: float64(5), int64(7), object(31)
memory usage: 1.3+ MB
```

In [115…
```
#OK all set lets save this down
final_merge_df.to_csv('final_data_nk.csv')
```

In [ ]:
```
```

Now to merge in ratings info as well as Chris' data

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [8]:
```
final_data_df = pd.read_csv('final_data_NK.csv')
```

In [3]:
```
final_data_df.head(2)
```

Out[3]:

	Unnamed: 0	Unnamed: 0.1	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes
0	0	20	foodfight!	tt0249516	Foodfight!	Foodfight!	2012	91.0
1	1	33	mortal kombat	tt0293429	Mortal Kombat	Mortal Kombat	2021	NaN

2 rows × 44 columns

In [9]:
```
final_data_df.drop(['Unnamed: 0', 'Unnamed: 0.1'], axis = 1, inplace =True)
```

In [10]:
```
final_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 42 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   clean_title         3897 non-null   object 
 1   movie_id            3897 non-null   object 
 2   primary_title       3897 non-null   object 
 3   original_title      3896 non-null   object 
 4   start_year          3897 non-null   int64  
 5   runtime_minutes     3401 non-null   float64
 6   genres              3824 non-null   object 
 7   id                  3897 non-null   int64  
 8   release_date        3897 non-null   object 
 9   movie               3897 non-null   object 
 10  production_budget   3897 non-null   int64  
 11  domestic_gross      3897 non-null   int64  
 12  worldwide_gross     3897 non-null   int64  
 13  foreign_gross       3897 non-null   int64  
 14  title               1666 non-null   object 
 15  studio              1666 non-null   object 
 16  domestic_grossbom   1665 non-null   float64
 17  foreign_grossbom    1426 non-null   float64
 18  year                1666 non-null   float64
 19  worldwide_grossbom  1425 non-null   float64
 20  Sport               3824 non-null   object 
 21  Crime               3824 non-null   object 
 22  News                3824 non-null   object 
 23  Romance             3824 non-null   object 
 24  Sci-Fi              3824 non-null   object 
 25  Western             3824 non-null   object 
 26  Horror              3824 non-null   object 
 27  Drama               3824 non-null   object 
 28  Family              3824 non-null   object 
 29  nan                 3824 non-null   object 
 30  Mystery             3824 non-null   object 
 31  Musical             3824 non-null   object 
 32  Fantasy             3824 non-null   object 
 33  Adventure           3824 non-null   object 
 34  Documentary         3824 non-null   object 
 35  Thriller            3824 non-null   object 
 36  Comedy              3824 non-null   object 
 37  Reality-TV          3824 non-null   object 
 38  Music               3824 non-null   object 
 39  History             3824 non-null   object 
 40  Action              3824 non-null   object 
 41  Animation           3824 non-null   object 
dtypes: float64(5), int64(6), object(31)
memory usage: 1.2+ MB
```

In [4]:
```
import sqlite3
conn = sqlite3.connect('C:/Users/Nick/Documents/Flatiron/phase_1/project/data/im.db')
```

In [5]:
```
movie_ratings_df = pd.read_sql("""
SELECT *
FROM movie_ratings
""", conn)
movie_ratings_df.head()
```

Out[5]:

	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

In [6]:
```
movie_ratings_df.set_index('movie_id', inplace = True)
```

In [11]:
```
ratings_join_df = final_data_df.join(movie_ratings_df, on='movie_id')
```

In [12]:
```
ratings_join_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 44 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   clean_title         3897 non-null   object 
 1   movie_id            3897 non-null   object 
 2   primary_title       3897 non-null   object 
 3   original_title      3896 non-null   object 
 4   start_year          3897 non-null   int64  
 5   runtime_minutes     3401 non-null   float64
 6   genres              3824 non-null   object 
 7   id                  3897 non-null   int64  
 8   release_date        3897 non-null   object 
 9   movie               3897 non-null   object 
 10  production_budget   3897 non-null   int64  
 11  domestic_gross      3897 non-null   int64  
 12  worldwide_gross     3897 non-null   int64  
 13  foreign_gross       3897 non-null   int64  
 14  title               1666 non-null   object 
 15  studio              1666 non-null   object 
 16  domestic_grossbom   1665 non-null   float64
 17  foreign_grossbom    1426 non-null   float64
 18  year                1666 non-null   float64
 19  worldwide_grossbom  1425 non-null   float64
 20  Sport               3824 non-null   object 
 21  Crime               3824 non-null   object 
 22  News                3824 non-null   object 
 23  Romance             3824 non-null   object 
 24  Sci-Fi              3824 non-null   object 
 25  Western             3824 non-null   object 
 26  Horror              3824 non-null   object 
 27  Drama               3824 non-null   object 
 28  Family              3824 non-null   object 
 29  nan                 3824 non-null   object 
 30  Mystery             3824 non-null   object 
 31  Musical             3824 non-null   object 
 32  Fantasy             3824 non-null   object 
 33  Adventure           3824 non-null   object 
 34  Documentary         3824 non-null   object 
 35  Thriller            3824 non-null   object 
 36  Comedy              3824 non-null   object 
 37  Reality-TV          3824 non-null   object 
 38  Music               3824 non-null   object 
 39  History             3824 non-null   object 
```

```
 40  Action              3824 non-null   object
 41  Animation           3824 non-null   object
 42  averagerating       2938 non-null   float64
 43  numvotes            2938 non-null   float64
dtypes: float64(7), int64(6), object(31)
memory usage: 1.3+ MB
```

In [13]:
```
ratings_join_df.head()
```

Out[13]:

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	gen
0	foodfight!	tt0249516	Foodfight!	Foodfight!	2012	91.0	Action,Animation,Come
1	mortal kombat	tt0293429	Mortal Kombat	Mortal Kombat	2021	NaN	Action,Adventure,Fant
2	the overnight	tt0326592	The Overnight	The Overnight	2010	88.0	N
3	on the road	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Drama,Romai
4	the secret life of walter mitty	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Dra

5 rows × 44 columns

In [19]:
```
ratings_join_df['release_date'] = pd.to_datetime(ratings_join_df['release_date'])
```

In [20]:
```
ratings_join_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 44 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   clean_title         3897 non-null   object
 1   movie_id            3897 non-null   object
 2   primary_title       3897 non-null   object
 3   original_title      3896 non-null   object
 4   start_year          3897 non-null   int64
 5   runtime_minutes     3401 non-null   float64
 6   genres              3824 non-null   object
 7   id                  3897 non-null   int64
 8   release_date        3897 non-null   datetime64[ns]
 9   movie               3897 non-null   object
 10  production_budget   3897 non-null   int64
 11  domestic_gross      3897 non-null   int64
 12  worldwide_gross     3897 non-null   int64
 13  foreign_gross       3897 non-null   int64
 14  title               1666 non-null   object
 15  studio              1666 non-null   object
 16  domestic_grossbom   1665 non-null   float64
 17  foreign_grossbom    1426 non-null   float64
 18  year                1666 non-null   float64
 19  worldwide_grossbom  1425 non-null   float64
 20  Sport               3824 non-null   object
```

```
 21  Crime              3824 non-null   object
 22  News               3824 non-null   object
 23  Romance            3824 non-null   object
 24  Sci-Fi             3824 non-null   object
 25  Western            3824 non-null   object
 26  Horror             3824 non-null   object
 27  Drama              3824 non-null   object
 28  Family             3824 non-null   object
 29  nan                3824 non-null   object
 30  Mystery            3824 non-null   object
 31  Musical            3824 non-null   object
 32  Fantasy            3824 non-null   object
 33  Adventure          3824 non-null   object
 34  Documentary        3824 non-null   object
 35  Thriller           3824 non-null   object
 36  Comedy             3824 non-null   object
 37  Reality-TV         3824 non-null   object
 38  Music              3824 non-null   object
 39  History            3824 non-null   object
 40  Action             3824 non-null   object
 41  Animation          3824 non-null   object
 42  averagerating      2938 non-null   float64
 43  numvotes           2938 non-null   float64
dtypes: datetime64[ns](1), float64(7), int64(6), object(30)
memory usage: 1.3+ MB
```

In [21]:
```
!ls
```

```
CONTRIBUTING.md
LICENSE.md
README.md
Untitled.ipynb
all_merged_fixup.csv
bom.movie_gross.csv
bom_genres_fixed_tues_am.csv
bom_raw_clean_fix.csv
bomwithgenres_roughdraft.csv
dsc-phase1-project-template.ipynb
final_data_nk.csv
genre_id_key.csv
im.db
movie_basics_clean_fix.csv
nick_scratch.ipynb
notebook_nk.ipynb
rt_reviews.csv
rtmovie_info.csv
tmdb.movies.csv
tn.movie_budgets.csv
tn_numfix.csv
tn_raw_clean_fix.csv
```

In [22]:
```
!cd ..
```

In [23]:
```
cd
```

```
C:\Users\Nick
```

In [24]:
```
cd Documents
```

```
C:\Users\Nick\Documents
```

In [25]:
```
cd Flatiron/phase_1/project/data
```

```
C:\Users\Nick\Documents\Flatiron\phase_1\project\data
```

In [26]:
```
ls
```

```
 Volume in drive C has no label.
 Volume Serial Number is 7C99-DF07

 Directory of C:\Users\Nick\Documents\Flatiron\phase_1\project\data

01/25/2022  03:21 PM    <DIR>          .
01/25/2022  03:21 PM    <DIR>          ..
01/25/2022  09:24 AM    <DIR>          .ipynb_checkpoints
01/25/2022  03:19 PM        44,365,937 all_data.xlsx
06/13/2019  11:42 PM           142,555 bom.movie_gross.csv
01/25/2022  10:23 AM           494,060 bom_genres_fixed_tues_am.csv
01/24/2022  11:07 PM           467,438 bom_genres_numfix.csv
01/24/2022  01:43 PM           320,049 bomwithgenres_roughdraft.csv
01/21/2022  05:34 PM           416,742 csv_data.csv
01/23/2022  06:43 PM               260 genre_id_key.csv
01/21/2022  06:55 PM       169,443,328 im.db
01/21/2022  06:58 PM                22 im.db.zip
01/25/2022  03:21 PM           108,780 Merged_DataFrame_CH.csv
01/25/2022  01:58 PM            59,898 nick_scratch.ipynb
01/23/2022  02:47 PM           478,896 practice_join.csv
12/12/2018  01:46 AM         1,184,685 rt.movie_info.tsv
12/12/2018  01:46 AM         9,395,716 rt.reviews.tsv
01/21/2022  05:19 PM         9,157,440 rt_reviews.csv
01/25/2022  11:36 AM         1,198,944 rtmovie_info.csv
01/21/2022  04:55 PM         2,279,683 tmdb.movies.csv
06/14/2019  02:29 PM           422,521 tn.movie_budgets.csv
01/25/2022  11:44 AM         1,413,986 tn_bom_join1.csv
01/24/2022  12:38 PM           550,980 tn4xl_1
01/24/2022  12:38 PM           550,980 tn4xl_1.csv
01/25/2022  02:12 PM           623,897 tn4xl_3.csv
              22 File(s)    243,076,797 bytes
               3 Dir(s)  256,331,296,768 bytes free
```

In [28]:
```
ratings_join_df.loc[ratings_join_df['clean_title']== 'avatar',:]
```

Out[28]:

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	id	releas
1071	avatar	tt1775309	Avatar	AbatÃ¢	2011	93.0	Horror	1	2009

1 rows × 44 columns

In [29]:
```
chris_df = pd.read_csv('Merged_DataFrame_CH.csv')
```

In [31]:
```
chris_df.head()
```

Out[31]:

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	interna
0	0	1	18-Dec-09	Avatar	425	760	2776	

	Unnamed: 0	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	interna
1	1	2	20-May-11	Pirates of the Caribbean: On Stranger Tides	410	241	1045	
2	2	4	1-May-15	Avengers: Age of Ultron	330	459	1403	
3	3	7	27-Apr-18	Avengers: Infinity War	300	678	2048	
4	4	9	17-Nov-17	Justice League	300	229	655	

In [33]:
```
chris_df.drop(['Unnamed: 0', 'id'], axis = 1, inplace = True)
```

In [34]:
```
chris_df['clean_title'] = chris_df['movie'].str.lower()
```

In [36]:
```
chris_df.set_index('clean_title', inplace = True)
```

In [38]:
```
merged_all_df = ratings_join_df.join(chris_df, on='clean_title', rsuffix = '_CH')
```

In [39]:
```
merged_all_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5378 entries, 0 to 3896
Data columns (total 51 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           5378 non-null   object
 1   movie_id              5378 non-null   object
 2   primary_title         5378 non-null   object
 3   original_title        5377 non-null   object
 4   start_year            5378 non-null   int64
 5   runtime_minutes       4682 non-null   float64
 6   genres                5279 non-null   object
 7   id                    5378 non-null   int64
 8   release_date          5378 non-null   datetime64[ns]
 9   movie                 5378 non-null   object
 10  production_budget     5378 non-null   int64
 11  domestic_gross        5378 non-null   int64
 12  worldwide_gross       5378 non-null   int64
 13  foreign_gross         5378 non-null   int64
 14  title                 2148 non-null   object
 15  studio                2148 non-null   object
 16  domestic_grossbom     2147 non-null   float64
 17  foreign_grossbom      1874 non-null   float64
 18  year                  2148 non-null   float64
 19  worldwide_grossbom    1873 non-null   float64
 20  Sport                 5279 non-null   object
```

```
 21  Crime                5279 non-null   object
 22  News                 5279 non-null   object
 23  Romance              5279 non-null   object
 24  Sci-Fi               5279 non-null   object
 25  Western              5279 non-null   object
 26  Horror               5279 non-null   object
 27  Drama                5279 non-null   object
 28  Family               5279 non-null   object
 29  nan                  5279 non-null   object
 30  Mystery              5279 non-null   object
 31  Musical              5279 non-null   object
 32  Fantasy              5279 non-null   object
 33  Adventure            5279 non-null   object
 34  Documentary          5279 non-null   object
 35  Thriller             5279 non-null   object
 36  Comedy               5279 non-null   object
 37  Reality-TV           5279 non-null   object
 38  Music                5279 non-null   object
 39  History              5279 non-null   object
 40  Action               5279 non-null   object
 41  Animation            5279 non-null   object
 42  averagerating        3797 non-null   float64
 43  numvotes             3797 non-null   float64
 44  release_date_CH      3416 non-null   object
 45  movie_CH             3416 non-null   object
 46  production_budget_CH 3416 non-null   float64
 47  domestic_gross_CH    3416 non-null   float64
 48  worldwide_gross_CH   3416 non-null   float64
 49  international_gross  3416 non-null   float64
 50  genre_ids            3416 non-null   object
dtypes: datetime64[ns](1), float64(11), int64(6), object(33)
memory usage: 2.1+ MB
```

In [40]:
```
merged_all_df.sort_values('clean_title')
```

Out[40]:

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
2373	#horror	tt3526286	#Horror	#Horror	2015	101.0	Crime,Drar
325	10 cloverfield lane	tt1179933	10 Cloverfield Lane	10 Cloverfield Lane	2016	103.0	Drama,Horrc
2329	10 days in a madhouse	tt3453052	10 Days in a Madhouse	10 Days in a Madhouse	2015	111.0	
2367	12 rounds	tt3517850	12 Rounds	12 Rounds	2017	NaN	Action,Drama
577	12 strong	tt1413492	12 Strong	12 Strong	2018	130.0	Action,Dran
...	...	...	...	...	...	...	
3236	zoom	tt6117454	Zoom	Zoom	2016	NaN	Crir
3152	zoom	tt5815346	Zoom	Zoom	2016	158.0	Comedy,Drama

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
2472	zoom	tt3763866	Zoom	Zoom	2015	96.0	Animation,Come
2103	zootopia	tt2948356	Zootopia	Zootopia	2016	108.0	Adventure,Animatio
1622	zulu	tt2249221	Zulu	Zulu	2013	110.0	Crime,Drar

5378 rows × 51 columns

In [41]:
```
merged_all_df.to_csv('merged_all.csv')
```

In [42]:
```
ls
```

```
 Volume in drive C has no label.
 Volume Serial Number is 7C99-DF07

 Directory of C:\Users\Nick\Documents\Flatiron\phase_1\project\data

01/25/2022  04:31 PM    <DIR>          .
01/25/2022  04:31 PM    <DIR>          ..
01/25/2022  09:24 AM    <DIR>          .ipynb_checkpoints
01/25/2022  03:19 PM        44,365,937 all_data.xlsx
06/13/2019  11:42 PM           142,555 bom.movie_gross.csv
01/25/2022  10:23 AM           494,060 bom_genres_fixed_tues_am.csv
01/24/2022  11:07 PM           467,438 bom_genres_numfix.csv
01/24/2022  01:43 PM           320,049 bomwithgenres_roughdraft.csv
01/21/2022  05:34 PM           416,742 csv_data.csv
01/23/2022  06:43 PM               260 genre_id_key.csv
01/21/2022  06:55 PM       169,443,328 im.db
01/21/2022  06:58 PM                22 im.db.zip
01/25/2022  04:28 PM         1,270,318 merged_all.csv
01/25/2022  03:21 PM           108,780 Merged_DataFrame_CH.csv
01/25/2022  04:31 PM            79,865 Merged_DataFrame_V2.csv
01/25/2022  01:58 PM            59,898 nick_scratch.ipynb
01/23/2022  02:47 PM           478,896 practice_join.csv
12/12/2018  01:46 AM         1,184,685 rt.movie_info.tsv
12/12/2018  01:46 AM         9,395,716 rt.reviews.tsv
01/21/2022  05:19 PM         9,157,440 rt_reviews.csv
01/25/2022  11:36 AM         1,198,944 rtmovie_info.csv
01/21/2022  04:55 PM         2,279,683 tmdb.movies.csv
06/14/2019  02:29 PM           422,521 tn.movie_budgets.csv
01/25/2022  11:44 AM         1,413,986 tn_bom_join1.csv
01/24/2022  12:38 PM           550,980 tn4xl_1
01/24/2022  12:38 PM           550,980 tn4xl_1.csv
01/25/2022  02:12 PM           623,897 tn4xl_3.csv
              24 File(s)    244,426,980 bytes
               3 Dir(s)  255,235,944,448 bytes free
```

In [43]:
```
chris_df2 = pd.read_csv('Merged_DataFrame_V2.csv')
```

In [45]:
```
chris_df2['clean_title'] = chris_df2['movie'].str.lower()
```

In [46]:
```
chris_df2.set_index('clean_title', inplace = True)
```

In [48]:
```
chris_df2.drop(['Unnamed: 0', 'id'], axis = 1, inplace = True)
```

In [49]:
```
chris_df2.head(2)
```

Out[49]:

clean_title	release_date	movie	production_budget	domestic_gross	worldwide_gross	international_
avatar	Dec 18, 2009	Avatar	425	760	2776	
pirates of the caribbean: on stranger tides	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410	241	1045	

In [50]:
```
merged_all_df2 = ratings_join_df.join(chris_df2, on='clean_title', rsuffix = '_CH')
```

In [51]:
```
merged_all_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3897 entries, 0 to 3896
Data columns (total 51 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           3897 non-null   object
 1   movie_id              3897 non-null   object
 2   primary_title         3897 non-null   object
 3   original_title        3896 non-null   object
 4   start_year            3897 non-null   int64
 5   runtime_minutes       3401 non-null   float64
 6   genres                3824 non-null   object
 7   id                    3897 non-null   int64
 8   release_date          3897 non-null   datetime64[ns]
 9   movie                 3897 non-null   object
 10  production_budget     3897 non-null   int64
 11  domestic_gross        3897 non-null   int64
 12  worldwide_gross       3897 non-null   int64
 13  foreign_gross         3897 non-null   int64
 14  title                 1666 non-null   object
 15  studio                1666 non-null   object
 16  domestic_grossbom     1665 non-null   float64
 17  foreign_grossbom      1426 non-null   float64
 18  year                  1666 non-null   float64
 19  worldwide_grossbom    1425 non-null   float64
 20  Sport                 3824 non-null   object
 21  Crime                 3824 non-null   object
 22  News                  3824 non-null   object
 23  Romance               3824 non-null   object
 24  Sci-Fi                3824 non-null   object
 25  Western               3824 non-null   object
 26  Horror                3824 non-null   object
 27  Drama                 3824 non-null   object
 28  Family                3824 non-null   object
 29  nan                   3824 non-null   object
```

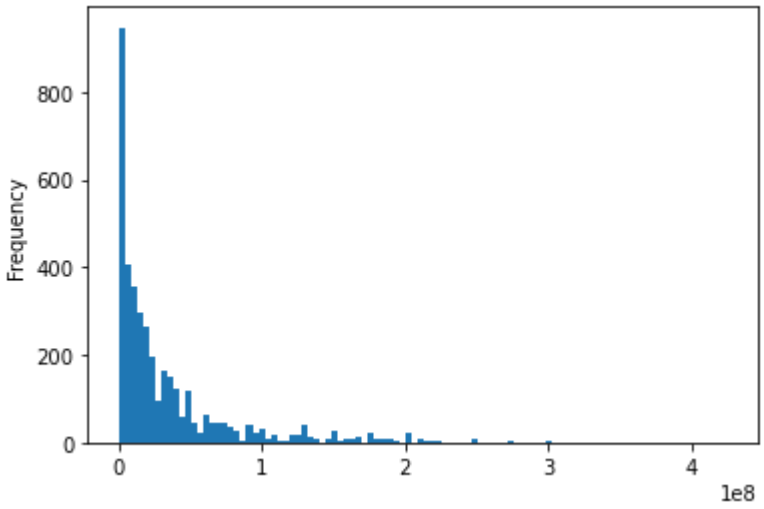
```
 30  Mystery               3824 non-null   object
 31  Musical               3824 non-null   object
 32  Fantasy               3824 non-null   object
 33  Adventure             3824 non-null   object
 34  Documentary           3824 non-null   object
 35  Thriller              3824 non-null   object
 36  Comedy                3824 non-null   object
 37  Reality-TV            3824 non-null   object
 38  Music                 3824 non-null   object
 39  History               3824 non-null   object
 40  Action                3824 non-null   object
 41  Animation             3824 non-null   object
 42  averagerating         2938 non-null   float64
 43  numvotes              2938 non-null   float64
 44  release_date_CH       1309 non-null   object
 45  movie_CH              1309 non-null   object
 46  production_budget_CH  1309 non-null   float64
 47  domestic_gross_CH     1309 non-null   float64
 48  worldwide_gross_CH    1309 non-null   float64
 49  international_gross   1309 non-null   float64
 50  genre_ids             1309 non-null   object
dtypes: datetime64[ns](1), float64(11), int64(6), object(33)
memory usage: 1.5+ MB
```

In [52]:
```
merged_all_df2.to_csv('master_data_raw.csv')
```

Exploring the full data

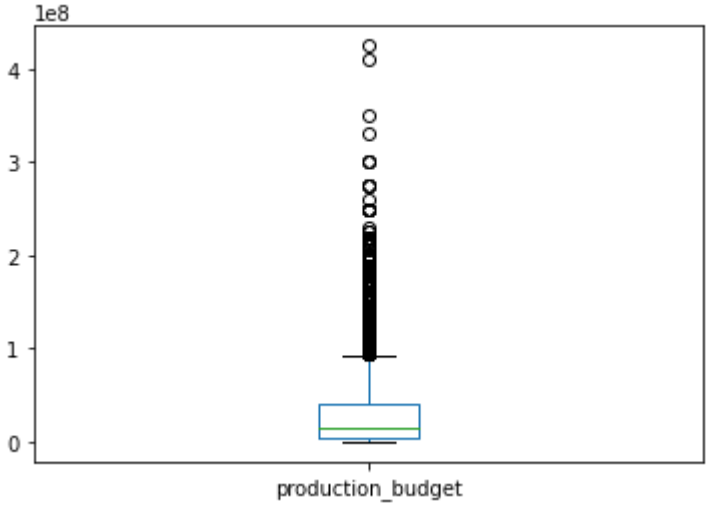
In [59]:
```
merged_all_df2['production_budget'].plot.hist(bins = 100)
```

Out[59]:
```
<AxesSubplot:ylabel='Frequency'>
```



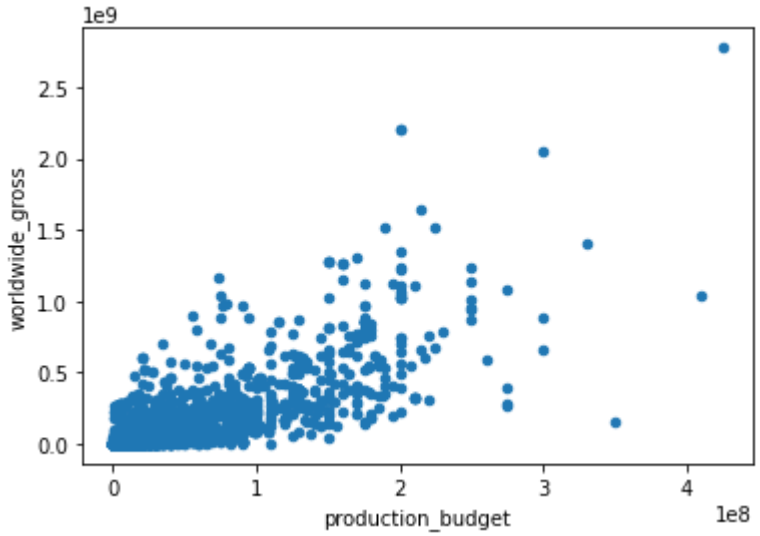
In [60]:
```
merged_all_df2['production_budget'].plot.box()
```

Out[60]:
```
<AxesSubplot:>
```



In [86]:
```
merged_all_df2[['production_budget', 'worldwide_gross']].plot.scatter(x='production_bud
```

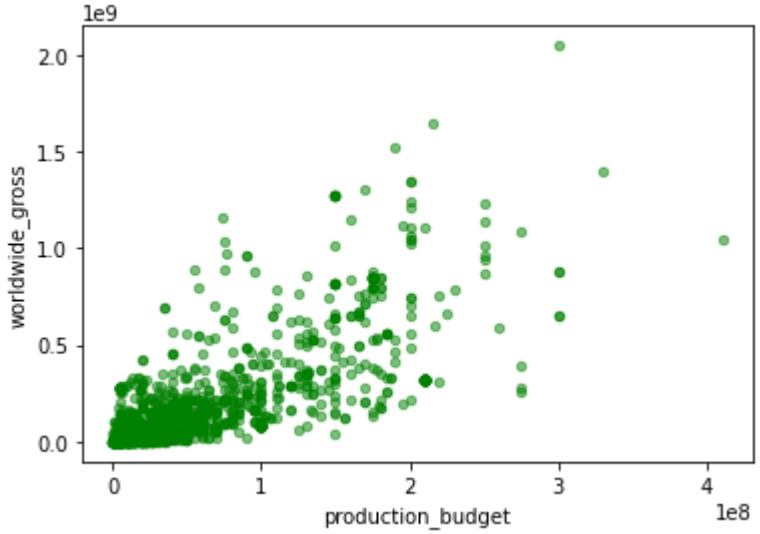
Out[86]:
```
<AxesSubplot:xlabel='production_budget', ylabel='worldwide_gross'>
```



In [88]:
```
after_2010 = merged_all_df2[merged_all_df2['year'] >=2010]
```

In [67]:
```
after_2010[['production_budget', 'worldwide_gross']].plot.scatter(x='production_budget'
```

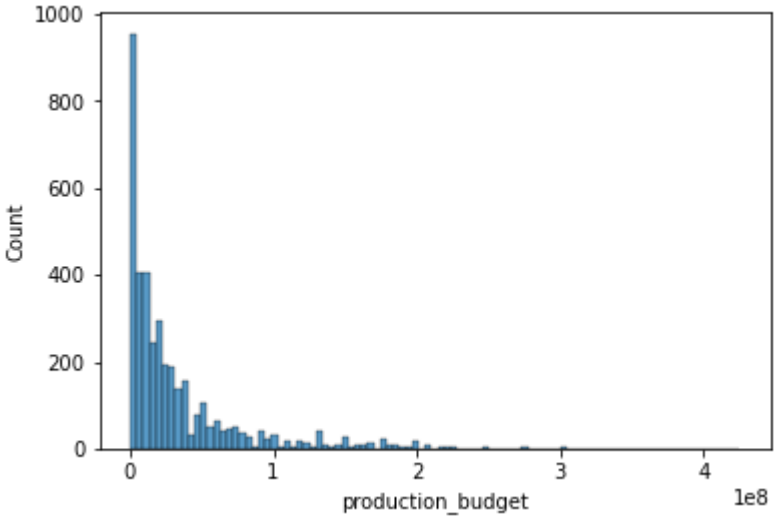
Out[67]:
```
<AxesSubplot:xlabel='production_budget', ylabel='worldwide_gross'>
```



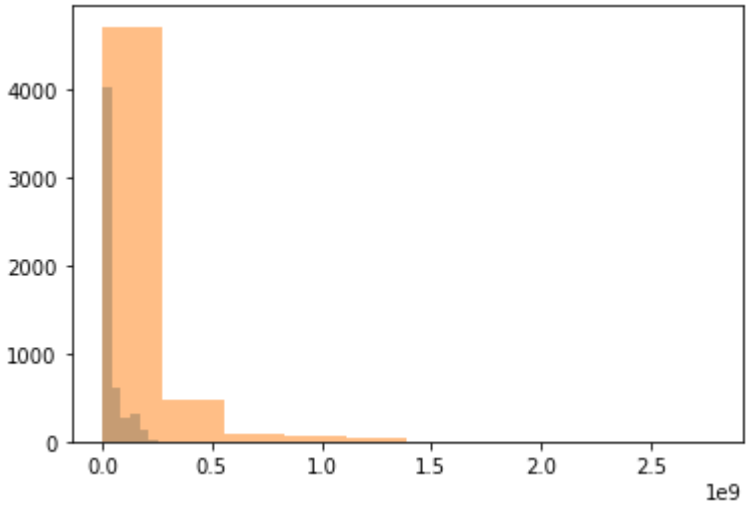
In [68]:
```
import seaborn as sns
```

In [89]:
```
sns.histplot(merged_all_df2['production_budget'])
```

Out[89]:
```
<AxesSubplot:xlabel='production_budget', ylabel='Count'>
```

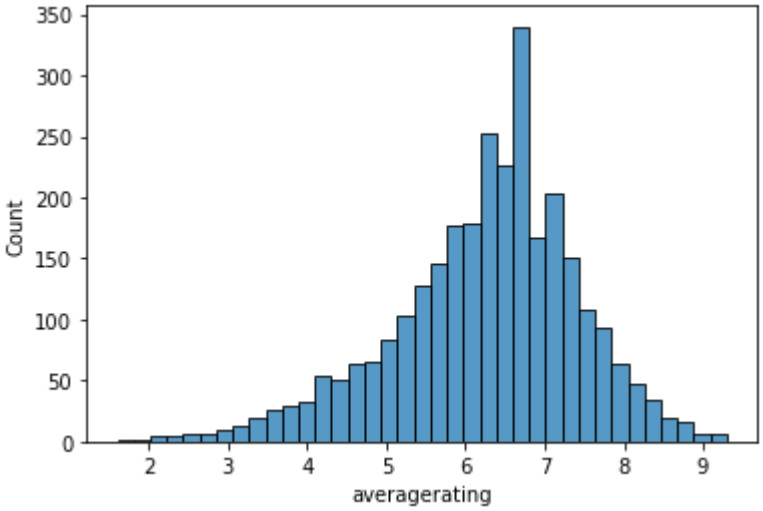


In [84]:
```
for col in ['production_budget' , 'worldwide_gross']:
    plt.hist(merged_all_df[col],  alpha = 0.5)
```



In [94]:
```
sns.histplot(merged_all_df2['averagerating'])
```

Out[94]:
```
<AxesSubplot:xlabel='averagerating', ylabel='Count'>
```



NOW merging in persons info from IMDB database

In [92]:
```
director_df = pd.read_sql("""
SELECT *
FROM directors
""", conn)
```

In [109…
```
director_df.head(2)
```

Out[109…

movie_id	person_id
tt0285252	nm0899854
tt0462036	nm1940585

```
writer_df = pd.read_sql("""
```

In [93]:
```
SELECT *
FROM directors
""", conn)
```

In [96]:
```
director_df.set_index('movie_id', inplace = True)
```

In [97]:
```
merge_wd = .join(director_df, on = 'movie_id')
```

In [98]:
```
writer_df.set_index('movie_id', inplace = True)
```

In [100…
```
merged_wd_df = merge_direct_df.join(writer_df, on ='movie_id', rsuffix = '_w')
```

merged

In [107…
```
merged_wd_df = merged_wd_df.drop_duplicates(subset = ['movie_id', 'release_date'])
```

In [108…
```
merged_wd_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3897 entries, 0 to 3896
Data columns (total 53 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           3897 non-null   object
 1   movie_id              3897 non-null   object
 2   primary_title         3897 non-null   object
 3   original_title        3896 non-null   object
 4   start_year            3897 non-null   int64
 5   runtime_minutes       3401 non-null   float64
 6   genres                3824 non-null   object
 7   id                    3897 non-null   int64
 8   release_date          3897 non-null   datetime64[ns]
 9   movie                 3897 non-null   object
 10  production_budget     3897 non-null   int64
 11  domestic_gross        3897 non-null   int64
 12  worldwide_gross       3897 non-null   int64
 13  foreign_gross         3897 non-null   int64
 14  title                 1666 non-null   object
 15  studio                1666 non-null   object
 16  domestic_grossbom     1665 non-null   float64
 17  foreign_grossbom      1426 non-null   float64
 18  year                  1666 non-null   float64
 19  worldwide_grossbom    1425 non-null   float64
 20  Sport                 3824 non-null   object
 21  Crime                 3824 non-null   object
 22  News                  3824 non-null   object
 23  Romance               3824 non-null   object
 24  Sci-Fi                3824 non-null   object
 25  Western               3824 non-null   object
 26  Horror                3824 non-null   object
 27  Drama                 3824 non-null   object
 28  Family                3824 non-null   object
 29  nan                   3824 non-null   object
 30  Mystery               3824 non-null   object
 31  Musical               3824 non-null   object
 32  Fantasy               3824 non-null   object
 33  Adventure             3824 non-null   object
 34  Documentary           3824 non-null   object
 35  Thriller              3824 non-null   object
 36  Comedy                3824 non-null   object
 37  Reality-TV            3824 non-null   object
 38  Music                 3824 non-null   object
```

```
 39  History               3824 non-null   object
 40  Action                3824 non-null   object
 41  Animation             3824 non-null   object
 42  averagerating         2938 non-null   float64
 43  numvotes              2938 non-null   float64
 44  release_date_CH       1309 non-null   object
 45  movie_CH              1309 non-null   object
 46  production_budget_CH  1309 non-null   float64
 47  domestic_gross_CH     1309 non-null   float64
 48  worldwide_gross_CH    1309 non-null   float64
 49  international_gross   1309 non-null   float64
 50  genre_ids             1309 non-null   object
 51  person_id             3805 non-null   object
 52  person_id_w           3805 non-null   object
dtypes: datetime64[ns](1), float64(11), int64(6), object(35)
memory usage: 1.6+ MB
```

In [110…
```
persons_df = pd.read_sql("""
SELECT *
FROM persons
""", conn)
```

In [111…
```
persons_df.head()
```

Out[111…

	person_id	primary_name	birth_year	death_year	primary_profession
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator

In [113…
```
persons_df.set_index('person_id', inplace=True)
```

In [114…
```
merge_final = merged_wd_df.join(persons_df, on='person_id')
```

In [115…
```
merge_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3897 entries, 0 to 3896
Data columns (total 57 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           3897 non-null   object
 1   movie_id              3897 non-null   object
 2   primary_title         3897 non-null   object
 3   original_title        3896 non-null   object
 4   start_year            3897 non-null   int64
 5   runtime_minutes       3401 non-null   float64
 6   genres                3824 non-null   object
 7   id                    3897 non-null   int64
 8   release_date          3897 non-null   datetime64[ns]
 9   movie                 3897 non-null   object
 10  production_budget     3897 non-null   int64
 11  domestic_gross        3897 non-null   int64
 12  worldwide_gross       3897 non-null   int64
 13  foreign_gross         3897 non-null   int64
```

```
 14  title                 1666 non-null   object
 15  studio                1666 non-null   object
 16  domestic_grossbom     1665 non-null   float64
 17  foreign_grossbom      1426 non-null   float64
 18  year                  1666 non-null   float64
 19  worldwide_grossbom    1425 non-null   float64
 20  Sport                 3824 non-null   object
 21  Crime                 3824 non-null   object
 22  News                  3824 non-null   object
 23  Romance               3824 non-null   object
 24  Sci-Fi                3824 non-null   object
 25  Western               3824 non-null   object
 26  Horror                3824 non-null   object
 27  Drama                 3824 non-null   object
 28  Family                3824 non-null   object
 29  nan                   3824 non-null   object
 30  Mystery               3824 non-null   object
 31  Musical               3824 non-null   object
 32  Fantasy               3824 non-null   object
 33  Adventure             3824 non-null   object
 34  Documentary           3824 non-null   object
 35  Thriller              3824 non-null   object
 36  Comedy                3824 non-null   object
 37  Reality-TV            3824 non-null   object
 38  Music                 3824 non-null   object
 39  History               3824 non-null   object
 40  Action                3824 non-null   object
 41  Animation             3824 non-null   object
 42  averagerating         2938 non-null   float64
 43  numvotes              2938 non-null   float64
 44  release_date_CH       1309 non-null   object
 45  movie_CH              1309 non-null   object
 46  production_budget_CH  1309 non-null   float64
 47  domestic_gross_CH     1309 non-null   float64
 48  worldwide_gross_CH    1309 non-null   float64
 49  international_gross   1309 non-null   float64
 50  genre_ids             1309 non-null   object
 51  person_id             3805 non-null   object
 52  person_id_w           3805 non-null   object
 53  primary_name          3805 non-null   object
 54  birth_year            1748 non-null   float64
 55  death_year            24 non-null     float64
 56  primary_profession    3796 non-null   object
dtypes: datetime64[ns](1), float64(13), int64(6), object(37)
memory usage: 1.7+ MB
```

In [116…
```
persons_df.reset_index(inplace = True)
```

In [117…
```
persons_df['person_id_w'] = persons_df['person_id']
```

In [119…
```
persons_df.set_index('person_id_w', inplace = True)
```

In [120…
```
master_merge = merge_final.join(persons_df, on='person_id_w', rsuffix = '_w')
```

FEATURE ENGINEERING PROFIT and ROI

In [203…
```
master_merge['worlwide_profit'] = master_merge['worldwide_gross'] - master_merge['produ
```

In [204…
```
master_merge['ROI'] = master_merge['worlwide_profit'].astype('float')//master_merge['pr
```

In [230…
```
master_merge.drop('nan', axis = 1, inplace = True)
```

In [275…
```
master_merge.drop(['%ROI', '% ROI', 'Reality-TV'], axis = 1, inplace = True)
```

In [276…
```
master_2010plus_df = master_merge[master_merge['year'] >= 2010]
```

In [172…
```
master_2010plus_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1666 entries, 3 to 3896
Data columns (total 66 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           1666 non-null   object
 1   movie_id              1666 non-null   object
 2   primary_title         1666 non-null   object
 3   original_title        1666 non-null   object
 4   start_year            1666 non-null   int64
 5   runtime_minutes       1563 non-null   float64
 6   genres                1640 non-null   object
 7   id                    1666 non-null   int64
 8   release_date          1666 non-null   datetime64[ns]
 9   movie                 1666 non-null   object
 10  production_budget     1666 non-null   int64
 11  domestic_gross        1666 non-null   int64
 12  worldwide_gross       1666 non-null   int64
 13  foreign_gross         1666 non-null   int64
 14  title                 1666 non-null   object
 15  studio                1666 non-null   object
 16  domestic_grossbom     1665 non-null   float64
 17  foreign_grossbom      1426 non-null   float64
 18  year                  1666 non-null   float64
 19  worldwide_grossbom    1425 non-null   float64
 20  Sport                 1640 non-null   object
 21  Crime                 1640 non-null   object
 22  News                  1640 non-null   object
 23  Romance               1640 non-null   object
 24  Sci-Fi                1640 non-null   object
 25  Western               1640 non-null   object
 26  Horror                1640 non-null   object
 27  Drama                 1640 non-null   object
 28  Family                1640 non-null   object
 29  nan                   1640 non-null   object
 30  Mystery               1640 non-null   object
 31  Musical               1640 non-null   object
 32  Fantasy               1640 non-null   object
 33  Adventure             1640 non-null   object
 34  Documentary           1640 non-null   object
 35  Thriller              1640 non-null   object
 36  Comedy                1640 non-null   object
 37  Reality-TV            1640 non-null   object
 38  Music                 1640 non-null   object
 39  History               1640 non-null   object
 40  Action                1640 non-null   object
 41  Animation             1640 non-null   object
 42  averagerating         1471 non-null   float64
 43  numvotes              1471 non-null   float64
 44  release_date_CH       1046 non-null   object
 45  movie_CH              1046 non-null   object
 46  production_budget_CH  1046 non-null   float64
 47  domestic_gross_CH     1046 non-null   float64
 48  worldwide_gross_CH    1046 non-null   float64
 49  international_gross   1046 non-null   float64
```

```
 50  genre_ids             1046 non-null   object
 51  person_id             1635 non-null   object
 52  person_id_w           1635 non-null   object
 53  primary_name          1635 non-null   object
 54  birth_year            1035 non-null   float64
 55  death_year            12 non-null     float64
 56  primary_profession    1634 non-null   object
 57  person_id_w           1635 non-null   object
 58  primary_name_w        1635 non-null   object
 59  birth_year_w          1035 non-null   float64
 60  death_year_w          12 non-null     float64
 61  primary_profession_w  1634 non-null   object
 62  worlwide_profit       1666 non-null   int64
 63  % ROI                 1666 non-null   int64
 64  %ROI                  1666 non-null   int64
 65  ROI                   1666 non-null   float64
dtypes: datetime64[ns](1), float64(16), int64(9), object(40)
memory usage: 872.0+ KB
```

exploring the data by grouping

In [132…
```
master_2010plus_df.groupby('person_id').sum().sort_values('worldwide_gross', ascending
```

Out[132…

person_id	start_year	runtime_minutes	id	production_budget	domestic_gross	worldwide_gross	fo
nm0751577	6048	432.0	40	720000000	1346646789	3902605502	
nm0634240	8053	587.0	150	750000000	1118801468	3086180484	
nm1490123	8056	495.0	259	371500000	879478118	3083488461	
nm0923736	8051	536.0	148	623600000	1128220169	2992084614	
nm0001392	6039	474.0	62	750000000	816490211	2922948044	
…	…	…	…	…	…	…	
nm0222816	2014	73.0	57	20000	0	0	
nm3781779	2014	86.0	43	10000000	0	0	
nm0908037	2013	98.0	18	600000	0	0	
nm0067009	2014	92.0	53	18000000	0	0	
nm0706987	2015	100.0	36	35000	0	0	

2940 rows × 21 columns

In [151…
```
master_2010plus_df.groupby('primary_name').mean().sort_values('worldwide_gross', ascend
```

Out[151…

primary_name	start_year	runtime_minutes	id	production_budget	domestic_gross	worldwide
Atsushi Wada	2011.000000	93.000000	1.000000	425000000.0	7.605076e+08	2.77634
Pete Meads	2012.000000	NaN	43.000000	200000000.0	6.593639e+08	2.20820

primary_name	start_year	runtime_minutes	id	production_budget	domestic_gross	worldwide
Ravi Punj	2018.000000	115.000000	43.000000	200000000.0	6.593639e+08	2.20820
Anthony Russo	2016.000000	144.000000	13.333333	240000000.0	4.488823e+08	1.30086
Chris Buck	2013.000000	102.000000	56.000000	150000000.0	4.007380e+08	1.27247
Chi-kin Kwok	2010.000000	92.000000	56.000000	150000000.0	4.007380e+08	1.27247
Adam Green	2010.000000	93.000000	56.000000	150000000.0	4.007380e+08	1.27247
Kyle Balda	2017.000000	89.000000	30.000000	75000000.0	2.646243e+08	1.03472
Jared Bush	2016.000000	108.000000	57.000000	150000000.0	3.412682e+08	1.01943
Sam Mendes	2013.500000	145.500000	31.000000	250000000.0	2.522172e+08	9.95074
Peter Jackson	2013.000000	158.000000	20.666667	250000000.0	2.721634e+08	9.74316
Zane Burden	2018.000000	60.000000	60.500000	90000000.0	4.400357e+08	9.33815
Christophe Gans	2014.000000	112.000000	60.500000	90000000.0	4.400357e+08	9.33815
Lee Unkrich	2013.500000	104.000000	24.500000	187500000.0	3.123654e+08	9.33443
Pierre Coffin	2012.666667	94.666667	48.333333	73000000.0	3.185417e+08	8.93005
Chris Renaud	2016.000000	87.000000	26.000000	75000000.0	3.683843e+08	8.86750
Alastair Siddons	2013.000000	75.000000	98.000000	175000000.0	3.564617e+08	8.54236
Russell Davidson	2018.000000	NaN	98.000000	175000000.0	3.564617e+08	8.54236
Nasir Rahim	2011.000000	NaN	98.000000	175000000.0	3.564617e+08	8.54236
Vaggelis Rigas	2011.000000	74.000000	98.000000	175000000.0	3.564617e+08	8.54236

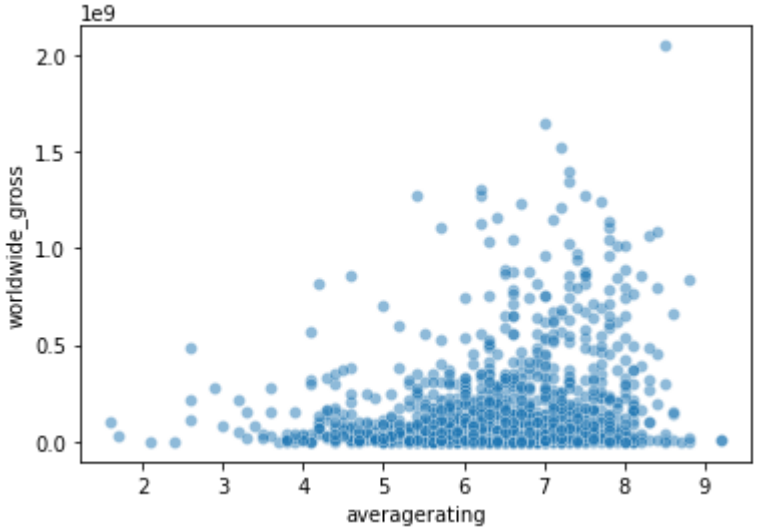
20 rows × 21 columns

master_2010plus_df.groupby('primary_name').count().sort_values('genres', ascending = False)[:20]

In [164…
```
sns.scatterplot(master_2010plus_df['averagerating'],master_2010plus_df['worldwide_gross
```

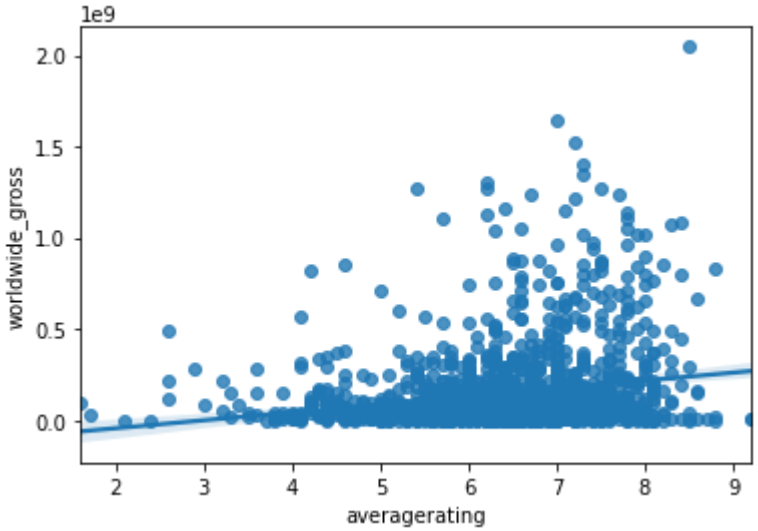
```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[164…
```
<AxesSubplot:xlabel='averagerating', ylabel='worldwide_gross'>
```



In [188…
```
sns.regplot(master_2010plus_df['averagerating'],master_2010plus_df['worldwide_gross'])
```

Out[188…
```
<AxesSubplot:xlabel='averagerating', ylabel='worldwide_gross'>
```



In [189…
```
master_2010plus_df['averagerating'].corr(master_2010plus_df['worldwide_gross'])
```

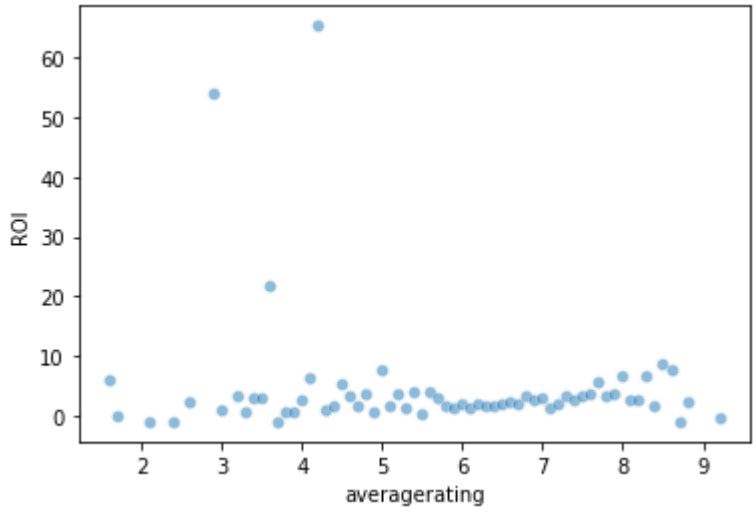
Out[189…
```
0.193461865788271
```

In [173…
```
master2010_by_rating = master_2010plus_df.groupby('averagerating').mean()
```

In [174…
```
sns.scatterplot(master2010_by_rating.index, master2010_by_rating['ROI'], alpha = 0.5)
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[174…
```
<AxesSubplot:xlabel='averagerating', ylabel='ROI'>
```



Isolating/exploring sample looking only at films from 2014->

In [277…
```
master2014_df = master_merge[master_merge['year'] > 2013]
```

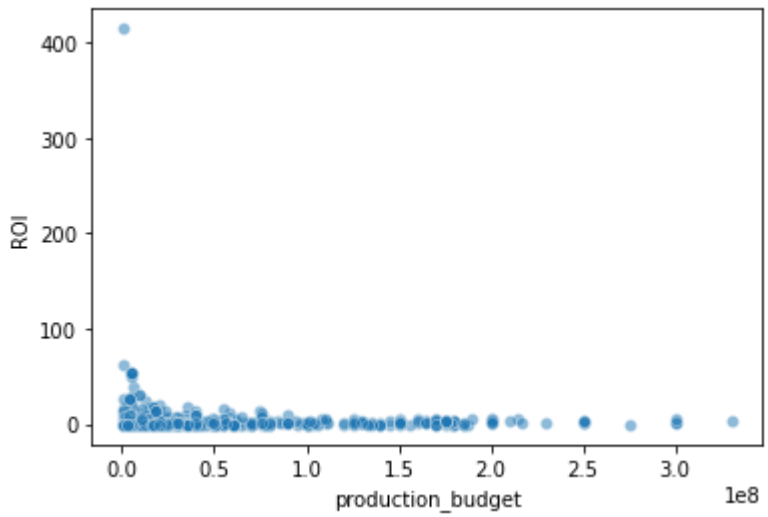
In [197…
```
master2014_df['averagerating'].corr(master2014_df['worldwide_gross'])
```

Out[197… 0.198065540652158

In [198…
```
sns.scatterplot(master2014_df['production_budget'],master2014_df['ROI'], alpha = 0.5)
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[198… <AxesSubplot:xlabel='production_budget', ylabel='ROI'>

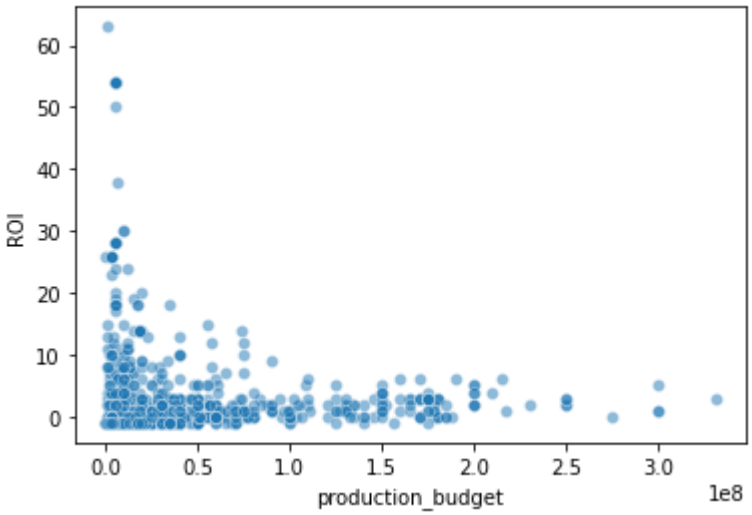


In [199…
```
master2014_roi_fix = master2014_df[master2014_df['ROI'] < 100]
```

In [200…
```
sns.scatterplot(master2014_roi_fix['production_budget'],master2014_roi_fix['ROI'], alph
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

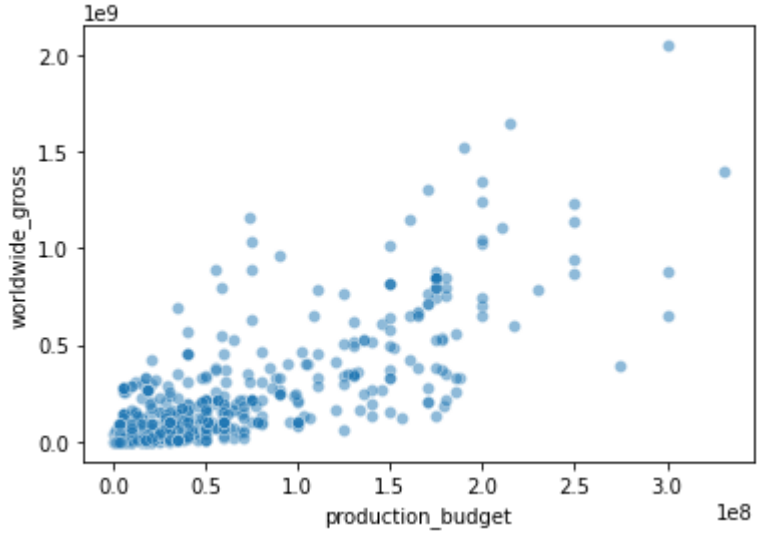
Out[200… <AxesSubplot:xlabel='production_budget', ylabel='ROI'>



In [201…
```
sns.scatterplot(master2014_df['production_budget'],master2014_df['worldwide_gross'], al
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[201… <AxesSubplot:xlabel='production_budget', ylabel='worldwide_gross'>



In [209…
```
master2014_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 882 entries, 5 to 3890
Data columns (total 66 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           882 non-null    object
 1   movie_id              882 non-null    object
 2   primary_title         882 non-null    object
```

```
 3   original_title        882 non-null    object
 4   start_year            882 non-null    int64
 5   runtime_minutes       836 non-null    float64
 6   genres                872 non-null    object
 7   id                    882 non-null    int64
 8   release_date          882 non-null    datetime64[ns]
 9   movie                 882 non-null    object
 10  production_budget     882 non-null    int64
 11  domestic_gross        882 non-null    int64
 12  worldwide_gross       882 non-null    int64
 13  foreign_gross         882 non-null    int64
 14  title                 882 non-null    object
 15  studio                882 non-null    object
 16  domestic_grossbom     882 non-null    float64
 17  foreign_grossbom      723 non-null    float64
 18  year                  882 non-null    float64
 19  worldwide_grossbom    723 non-null    float64
 20  Sport                 872 non-null    object
 21  Crime                 872 non-null    object
 22  News                  872 non-null    object
 23  Romance               872 non-null    object
 24  Sci-Fi                872 non-null    object
 25  Western               872 non-null    object
 26  Horror                872 non-null    object
 27  Drama                 872 non-null    object
 28  Family                872 non-null    object
 29  nan                   872 non-null    object
 30  Mystery               872 non-null    object
 31  Musical               872 non-null    object
 32  Fantasy               872 non-null    object
 33  Adventure             872 non-null    object
 34  Documentary           872 non-null    object
 35  Thriller              872 non-null    object
 36  Comedy                872 non-null    object
 37  Reality-TV            872 non-null    object
 38  Music                 872 non-null    object
 39  History               872 non-null    object
 40  Action                872 non-null    object
 41  Animation             872 non-null    object
 42  averagerating         779 non-null    float64
 43  numvotes              779 non-null    float64
 44  release_date_CH       514 non-null    object
 45  movie_CH              514 non-null    object
 46  production_budget_CH  514 non-null    float64
 47  domestic_gross_CH     514 non-null    float64
 48  worldwide_gross_CH    514 non-null    float64
 49  international_gross   514 non-null    float64
 50  genre_ids             514 non-null    object
 51  person_id             867 non-null    object
 52  person_id_w           867 non-null    object
 53  primary_name          867 non-null    object
 54  birth_year            526 non-null    float64
 55  death_year            1 non-null      float64
 56  primary_profession    866 non-null    object
 57  person_id_w           867 non-null    object
 58  primary_name_w        867 non-null    object
 59  birth_year_w          526 non-null    float64
 60  death_year_w          1 non-null      float64
 61  primary_profession_w  866 non-null    object
 62  worlwide_profit       882 non-null    int64
 63  % ROI                 882 non-null    int64
 64  %ROI                  882 non-null    int64
 65  ROI                   882 non-null    float64
dtypes: datetime64[ns](1), float64(16), int64(9), object(40)
memory usage: 461.7+ KB
```

In [ ]:
```
sns.scatterplot(master2014_df['production_budget'],master2014_df['worlwide_profit'], al
```

Looking to see if there were significant changes in genre over time

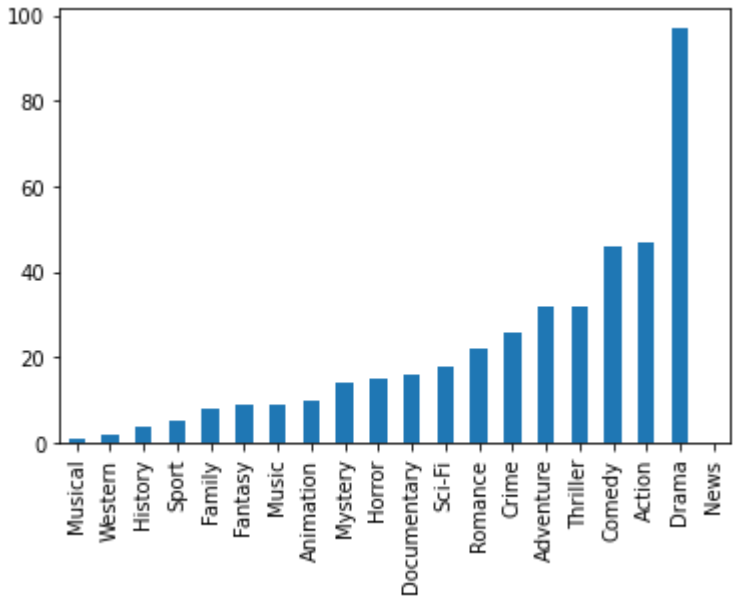
In [278…
```
#making subsets of master by year
master2014_only = master_merge[master_merge['year'] == 2014]
master2015_only = master_merge[master_merge['year'] == 2015]
master2016_only = master_merge[master_merge['year'] == 2016]
master2017_only = master_merge[master_merge['year'] == 2017]
master2018_only = master_merge[master_merge['year'] == 2018]
master2019_only = master_merge[master_merge['year'] == 2019]
master2020_only = master_merge[master_merge['year'] == 2020]
```

In [ ]:
```
master2014_only[genres].apply(pd.value_counts)
```

In [282…
```
genres2014_df = master2014_only[genres].apply(pd.value_counts)
```

In [283…
```
genres2014_df.loc[True].sort_values().plot.bar()
```

Out[283… <AxesSubplot:>



In [224…
```
genres = master_merge.columns[20:42]
```

In [239…
```
genres = list(genres)
```

In [281…
```
genres.pop(-5)
```

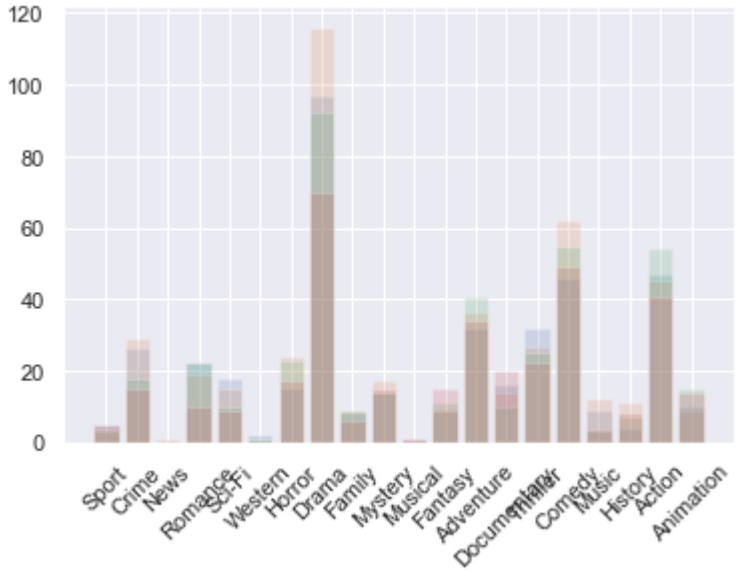
Out[281… 'Reality-TV'

In [285…
```
genres2015_df = master2015_only[genres].apply(pd.value_counts)
genres2016_df = master2016_only[genres].apply(pd.value_counts)
genres2017_df = master2017_only[genres].apply(pd.value_counts)
genres2018_df = master2018_only[genres].apply(pd.value_counts)
```

```
genres2019_df = master2019_only[genres].apply(pd.value_counts)
genres2020_df = master2020_only[genres].apply(pd.value_counts)
```

In [395…
```
sns.set()
```

In [396…
```
fig, ax = plt.subplots()

ax.bar(genres, genres2014_df.loc[True], alpha=0.2,)
ax.bar(genres, genres2015_df.loc[True], alpha=0.2)
ax.bar(genres, genres2016_df.loc[True], alpha=0.2)
ax.bar(genres, genres2017_df.loc[True], alpha=0.2)
plt.xticks(rotation = 45);
```



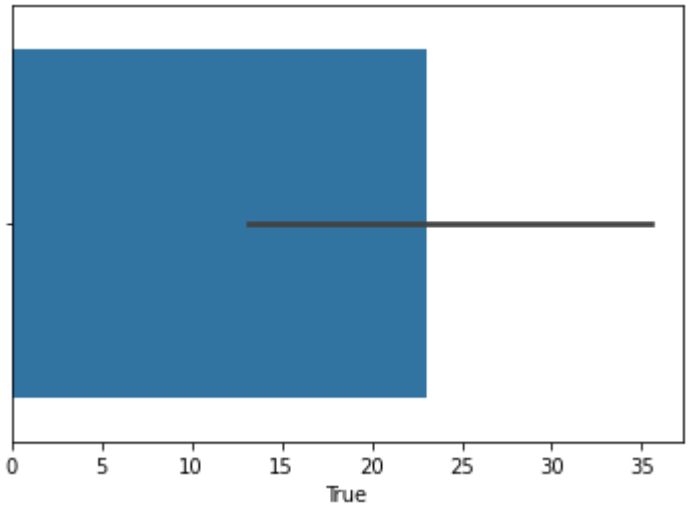
In [1]:
```
ax.bar(genres, [genres2014_df.loc[True],
 genres2015_df.loc[True], genres2016_df.loc[True], genres2017_df.loc[True]])
plt.xticks(rotation = 45);
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-3c57b909bc0f> in <module>
----> 1 ax.bar(genres, [genres2014_df.loc[True],
      2  genres2015_df.loc[True], genres2016_df.loc[True], genres2017_df.loc[True]])
      3 plt.xticks(rotation = 45);

NameError: name 'ax' is not defined
```

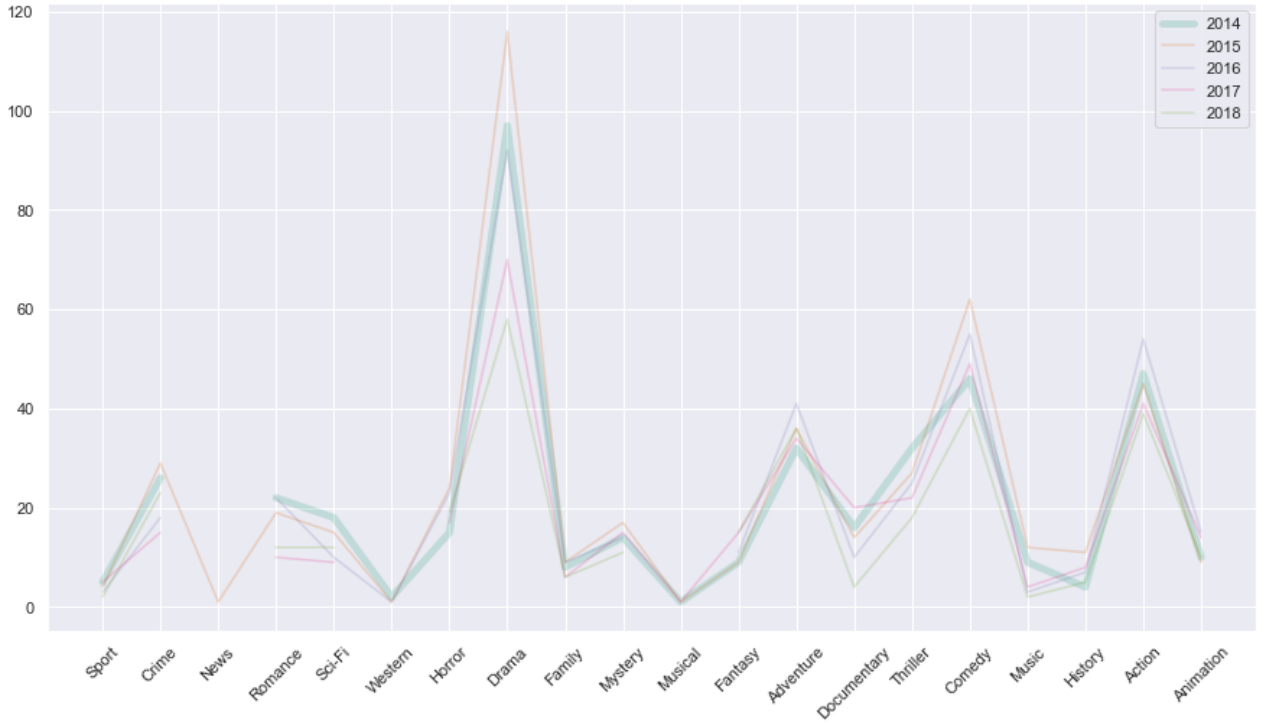
In [304…

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[304… <AxesSubplot:xlabel='True'>



In [430…
```
fig, ax = plt.subplots(figsize = (15,8))

ax.plot(genres, genres2014_df.loc[True], alpha=0.2, linewidth=5)
ax.plot(genres, genres2015_df.loc[True], alpha=0.2)
ax.plot(genres, genres2016_df.loc[True], alpha=0.2)
ax.plot(genres, genres2017_df.loc[True], alpha=0.2)
ax.plot(genres, genres2018_df.loc[True], alpha=0.2)
ax.legend(['2014','2015','2016','2017','2018'])

plt.xticks(rotation = 45);
```



Looking at Directors

In [324…
```
director_group_df = master_merge.groupby('primary_name_w').sum().sort_values('worlwide_
director_group_df.head()
```

Out[324…

	start_year	runtime_minutes	id	production_budget	domestic_gross	worldwide_gross

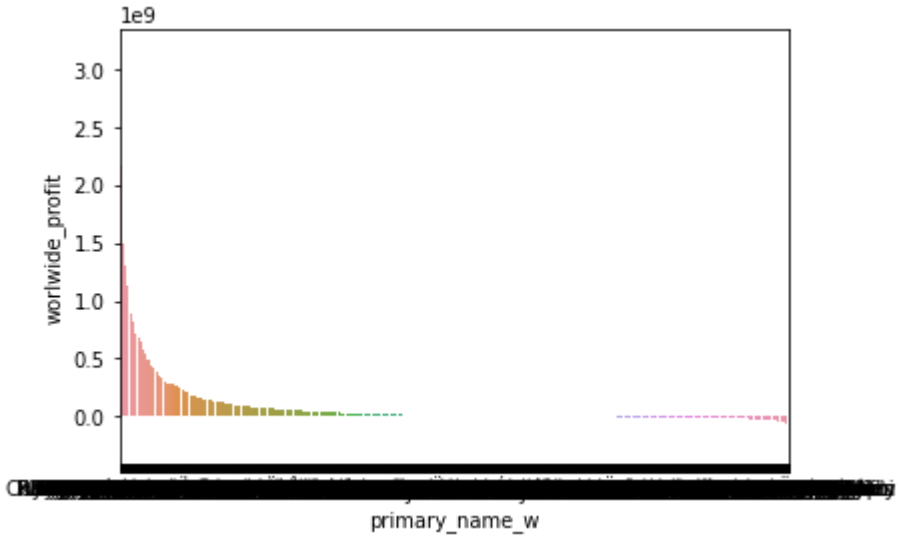
primary_name_w	start_year	runtime_minutes	id	production_budget	domestic_gross	worldwide_gross
primary_name_w						
Anthony Russo	6048	432.0	40	720000000	1346646789	3902605502
James Wan	8056	495.0	259	371500000	879478118	3083488461
Pierre Coffin	6038	284.0	145	219000000	955625140	2679017581
Joss Whedon	8051	536.0	148	623600000	1128220169	2992084614
Atsushi Wada	2011	93.0	1	425000000	760507625	2776345279

5 rows × 23 columns

In [328…
```
sns.barplot(director_group_df.index, director_group_df['worlwide_profit'].)
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[328…
```
<AxesSubplot:xlabel='primary_name_w', ylabel='worlwide_profit'>
```



In [334…
```
director_group_df['worlwide_profit'].sort_values()[-20:].sum()
```

Out[334…
```
40773839556
```

In [335…
```
director_group_df['worlwide_profit'].sort_values()[:-20].sum()
```

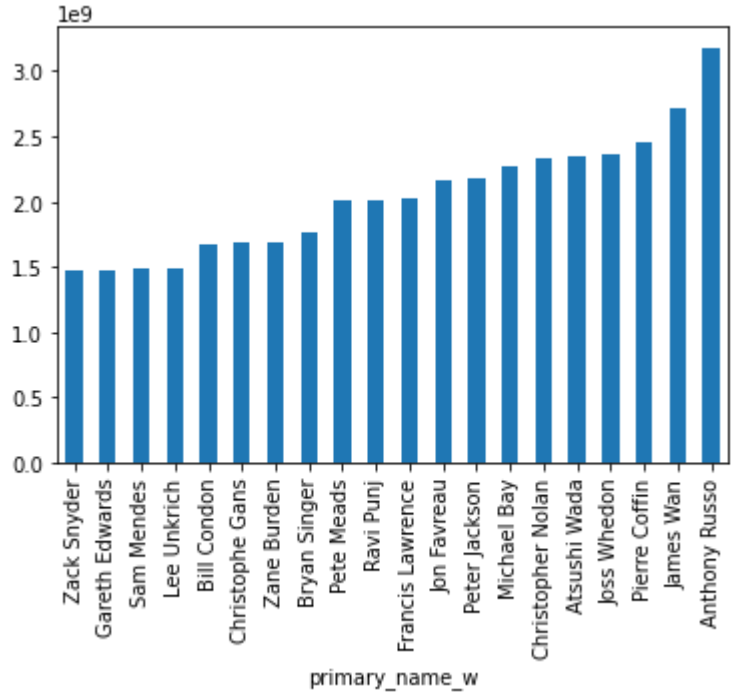
Out[335…
```
192194554972
```

In [336…
```
40773839556/192194554972
```

Out[336…
```
0.21214877581698485
```

In [337…
```
director_group_df['worlwide_profit'].sort_values()[-20:].plot.bar()
```

Out[337… <AxesSubplot:xlabel='primary_name_w'>



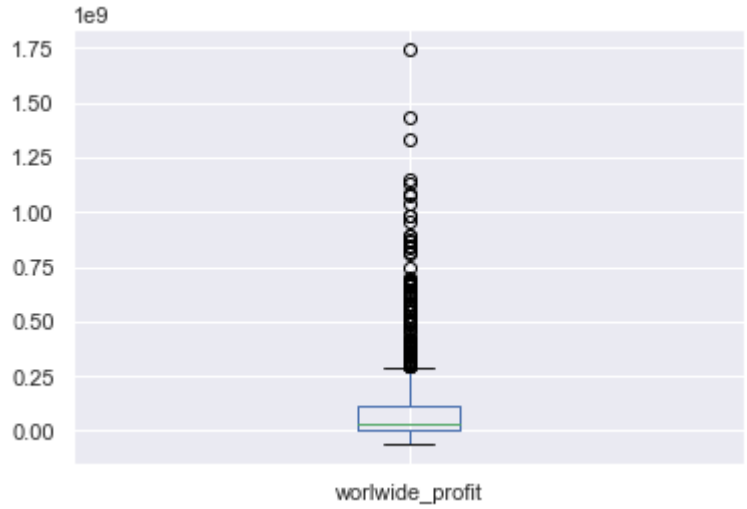
In [341…
```
master_merge['primary_name_w']
```

Out[341…
```
0       Lawrence Kasanoff
1           Simon McQuoid
2          Jed I. Goodman
3           Walter Salles
4             Ben Stiller
               ...
3892                  NaN
3893       Edwine Dorival
3894       Bill Guttentag
3895     Hatef Alimardani
3896         Nick Willing
Name: primary_name_w, Length: 3897, dtype: object
```

In [342…
```
master_merge.to_csv('master_merge.csv')
```

In [398…
```
master2014_df['worlwide_profit'].plot.box()
```

Out[398… <AxesSubplot:>



In [350…
```
a = master2014_df['worlwide_profit'].sort_values(ascending = False)[:20].sum()
```

In [351…
```
b = master2014_df['worlwide_profit'].sort_values(ascending = False)[20:].sum()
```

In [462…
```
# percentage of total global profit from top 20 movies over 21%
a/(b+a)
```

Out[462… 0.21104172782432867

In [353…
```
len(master2014_df['worlwide_profit'].sort_values(ascending = False))
```

Out[353… 882

In [355…
```
master2014_df.sort_values('worlwide_profit', ascending = False)[:20]
```

Out[355…

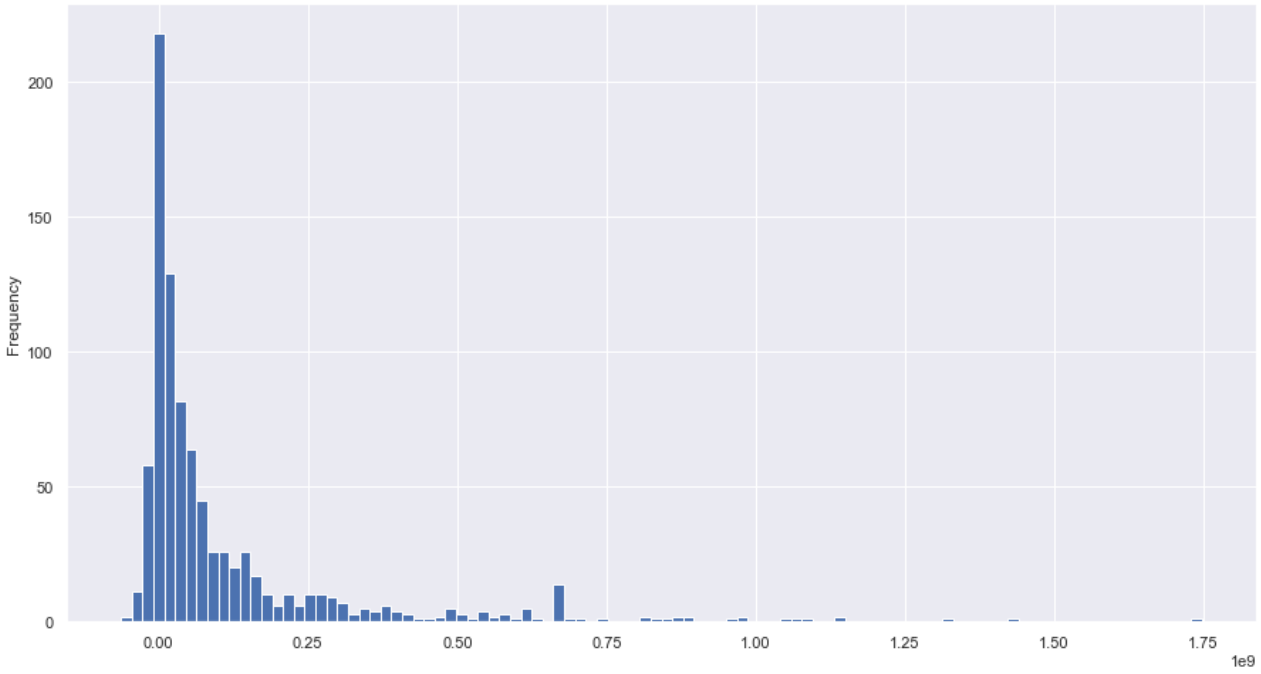
	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
2623	avengers: infinity war	tt4154756	Avengers: Infinity War	Avengers: Infinity War	2018	149.0	Action,Adv
6	jurassic world	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adv
2041	furious 7	tt2820852	Furious 7	Furious Seven	2015	137.0	Action,C
1134	black panther	tt1825683	Black Panther	Black Panther	2018	134.0	Action,Adv
2879	jurassic world: fallen kingdom	tt4881806	Jurassic World: Fallen Kingdom	Jurassic World: Fallen Kingdom	2018	128.0	Action,Adv
1661	minions	tt2293640	Minions	Minions	2015	91.0	Adventure,Animat
1786	avengers: age of ultron	tt2395427	Avengers: Age of Ultron	Avengers: Age of Ultron	2015	141.0	Action,Adv
2410	incredibles 2	tt3606756	Incredibles 2	Incredibles 2	2018	118.0	Action,Adventur
649	aquaman	tt1477834	Aquaman	Aquaman	2018	143.0	Action,Adven

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
2794	the fate of the furious	tt4630562	The Fate of the Furious	The Fate of the Furious	2017	136.0	Action,C
2336	despicable me 3	tt3469046	Despicable Me 3	Despicable Me 3	2017	89.0	Adventure,Animat
1473	transformers: age of extinction	tt2109248	Transformers: Age of Extinction	Transformers: Age of Extinction	2014	165.0	Action,Adv
2351	captain america: civil war	tt3498820	Captain America: Civil War	Captain America: Civil War	2016	147.0	Action,Adv
1654	jumanji: welcome to the jungle	tt2283362	Jumanji: Welcome to the Jungle	Jumanji: Welcome to the Jungle	2017	119.0	Action,Advent
2103	zootopia	tt2948356	Zootopia	Zootopia	2016	108.0	Adventure,Animat
2464	rogue one: a star wars story	tt3748528	Rogue One: A Star Wars Story	Rogue One	2016	133.0	Action,Adv
1003	bohemian rhapsody	tt1727824	Bohemian Rhapsody	Bohemian Rhapsody	2018	134.0	Biography,D
1647	finding dory	tt2277860	Finding Dory	Finding Dory	2016	97.0	Adventure,Animat
1997	the secret life of pets	tt2709768	The Secret Life of Pets	The Secret Life of Pets	2016	87.0	Adventure,Animat
598	deadpool	tt1431045	Deadpool	Deadpool	2016	108.0	Action,Advent

20 rows × 64 columns

In [399…
```
master2014_df['worlwide_profit'].plot.hist(figsize = (15,8), bins=100)
```

Out[399…
```
<AxesSubplot:ylabel='Frequency'>
```



isolating films that lost money

In [366…
```
lost_money_df =
master2014_df[master2014_df['worlwide_profit'] < 0]
```

In [367…
```
lost_money_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 210 entries, 21 to 3861
Data columns (total 64 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   clean_title           210 non-null    object
 1   movie_id              210 non-null    object
 2   primary_title         210 non-null    object
 3   original_title        210 non-null    object
 4   start_year            210 non-null    int64
 5   runtime_minutes       194 non-null    float64
 6   genres                208 non-null    object
 7   id                    210 non-null    int64
 8   release_date          210 non-null    datetime64[ns]
 9   movie                 210 non-null    object
 10  production_budget     210 non-null    int64
 11  domestic_gross        210 non-null    int64
 12  worldwide_gross       210 non-null    int64
 13  foreign_gross         210 non-null    int64
 14  title                 210 non-null    object
 15  studio                210 non-null    object
 16  domestic_grossbom     210 non-null    float64
 17  foreign_grossbom      117 non-null    float64
 18  year                  210 non-null    float64
 19  worldwide_grossbom    117 non-null    float64
 20  Sport                 208 non-null    object
 21  Crime                 208 non-null    object
 22  News                  208 non-null    object
 23  Romance               208 non-null    object
 24  Sci-Fi                208 non-null    object
 25  Western               208 non-null    object
```

```
 26  Horror                208 non-null    object
 27  Drama                 208 non-null    object
 28  Family                208 non-null    object
 29  Mystery               208 non-null    object
 30  Musical               208 non-null    object
 31  Fantasy               208 non-null    object
 32  Adventure             208 non-null    object
 33  Documentary           208 non-null    object
 34  Thriller              208 non-null    object
 35  Comedy                208 non-null    object
 36  Music                 208 non-null    object
 37  History               208 non-null    object
 38  Action                208 non-null    object
 39  Animation             208 non-null    object
 40  averagerating         173 non-null    float64
 41  numvotes              173 non-null    float64
 42  release_date_CH       87 non-null     object
 43  movie_CH              87 non-null     object
 44  production_budget_CH  87 non-null     float64
 45  domestic_gross_CH     87 non-null     float64
 46  worldwide_gross_CH    87 non-null     float64
 47  international_gross   87 non-null     float64
 48  genre_ids             87 non-null     object
 49  person_id             208 non-null    object
 50  person_id_w           208 non-null    object
 51  primary_name          208 non-null    object
 52  birth_year            121 non-null    float64
 53  death_year            0 non-null      float64
 54  primary_profession    208 non-null    object
 55  person_id_w           208 non-null    object
 56  primary_name_w        208 non-null    object
 57  birth_year_w          121 non-null    float64
 58  death_year_w          0 non-null      float64
 59  primary_profession_w  208 non-null    object
 60  worlwide_profit       210 non-null    int64
 61  ROI                   210 non-null    float64
 62  Biography             208 non-null    object
 63  War                   208 non-null    object
dtypes: datetime64[ns](1), float64(16), int64(7), object(40)
memory usage: 106.6+ KB
```

In [528…
```
sns.set_style('white')
```

INVESTIGATING RATINGs and STUDIOS

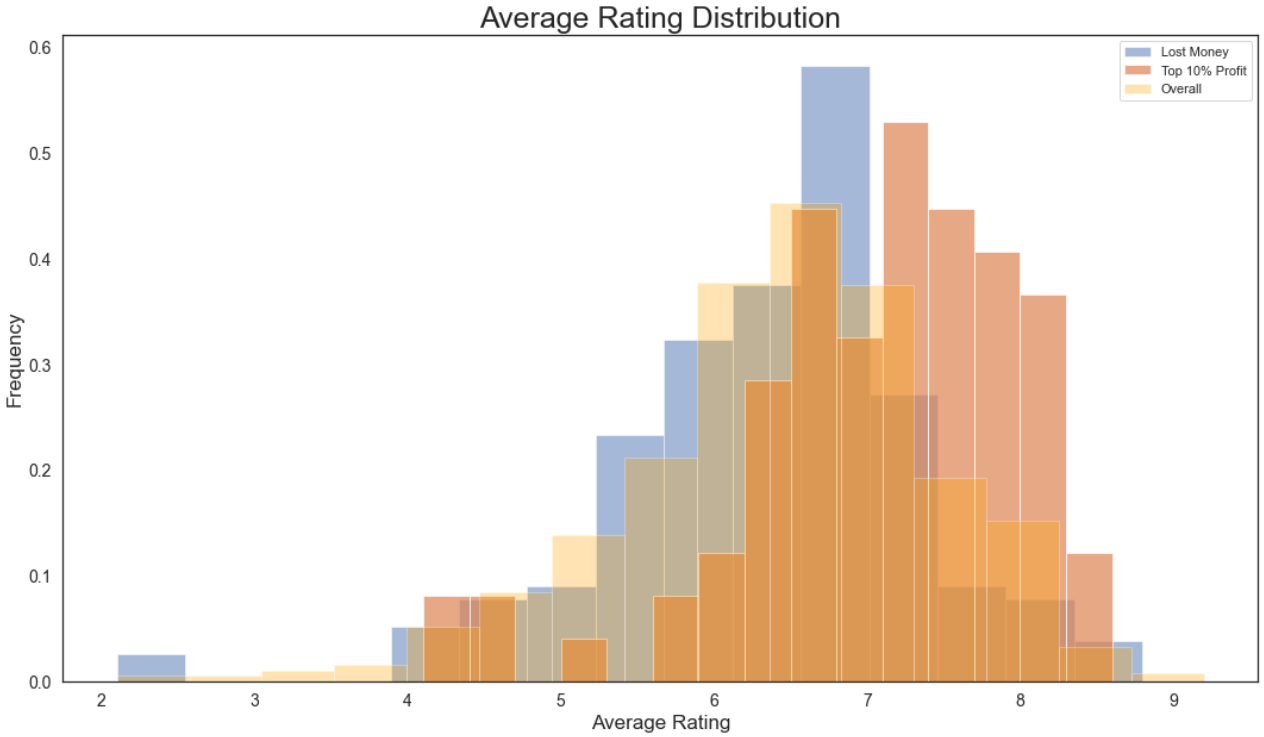
In [577…
```
top_studios=['BV', 'Uni.', 'Fox', 'WB', 'Sony', 'Par.', 'WB (NL)', 'LGF', 'Wein.',
       'FoxS', 'STX', 'LG/S', 'Focus', 'SGem', 'TriS', 'UTV', 'A24', 'ORF',
       'Studio 8', 'GrtIndia', 'Rela.', 'W/Dim.', 'MGM', 'SPC', 'Yash', 'Eros',
       'EOne', 'BH Tilt', 'PNT', 'RTWC', 'IFC', 'Neon', 'BST', 'EC', 'PFR',
       'Affirm', 'Orch.', 'RAtt.', 'MBox', 'Cleopatra', 'LGP', 'Trib.',
       'Annapurna', 'BBC', 'Cohen', 'Drft.', 'Global Road', 'DR', 'CE', 'XL',
       'Gold.', 'Free', 'ENTMP', 'TFA', 'Saban', 'RLJ', 'Alc', 'Amazon', 'BSC',
       'BG', 'Magn.', 'FCW', 'KE', 'VE']
master2014_df[master2014_df['studio'].isin(top_studios)]['averagerating'].median()
```

Out[577… 6.6

In [686…
```
fig, ax = plt.subplots(figsize=(18,10))
ax.hist(lost_money_df['averagerating'], alpha=0.5, bins = 15, density=True)
ax.hist(master2014_df.sort_values('worlwide_profit', ascending = False)['averagerating'
```

```
ax.hist(master2014_df['averagerating'], alpha = 0.3, color='orange', bins=15, density=T
ax.legend(['Lost Money', 'Top 10% Profit', 'Overall'])
ax.set(title='Rating and Profit', xlabel='Average Rating', ylabel= 'Frequency')
ax.set_title('Average Rating Distribution', size=25);
ax.set_xlabel('Average Rating', size=17)
ax.set_ylabel('Frequency', size = 17)
plt.xticks(size=14)
plt.yticks(size = 14);
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\numpy\lib\histograms.py:839: Ru
ntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\numpy\lib\histograms.py:840: Ru
ntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```



In [505…
```
master2014_df.sort_values('worlwide_profit', ascending = False)['averagerating'][:88].m
```

Out[505…
```
7.15
```

In [506…
```
master2014_df.sort_values('worlwide_profit', ascending = False)['averagerating'][:88].m
```

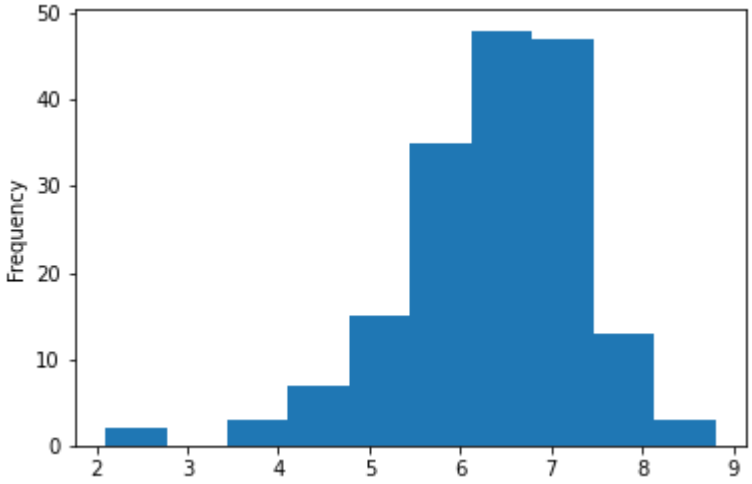
Out[506…
```
7.031707317073169
```

In [507…
```
print(lost_money_df['averagerating'].median(),lost_money_df['averagerating'].mean())
```

```
6.5 6.349132947976879
```

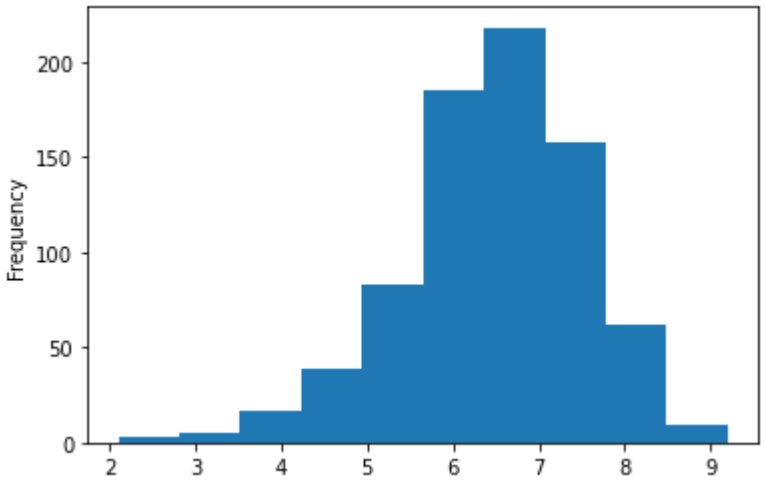
In [368…
```
lost_money_df['averagerating'].plot.hist()
```

Out[368…
```
<AxesSubplot:ylabel='Frequency'>
```



In [369…
```
master2014_df['averagerating'].plot.hist()
```

Out[369…
```
<AxesSubplot:ylabel='Frequency'>
```



In [373…
```
top_20_df = master2014_df[master2014_df['worlwide_profit'] > 742000000]
top_20_df['averagerating'].mean()
```

Out[373…
```
7.189999999999998
```

In [520…
```
top_20_df
```

Out[520…

	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
6	jurassic world	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adv
598	deadpool	tt1431045	Deadpool	Deadpool	2016	108.0	Action,Advent
649	aquaman	tt1477834	Aquaman	Aquaman	2018	143.0	Action,Adven
1003	bohemian rhapsody	tt1727824	Bohemian Rhapsody	Bohemian Rhapsody	2018	134.0	Biography,D
1134	black panther	tt1825683	Black Panther	Black Panther	2018	134.0	Action,Adv

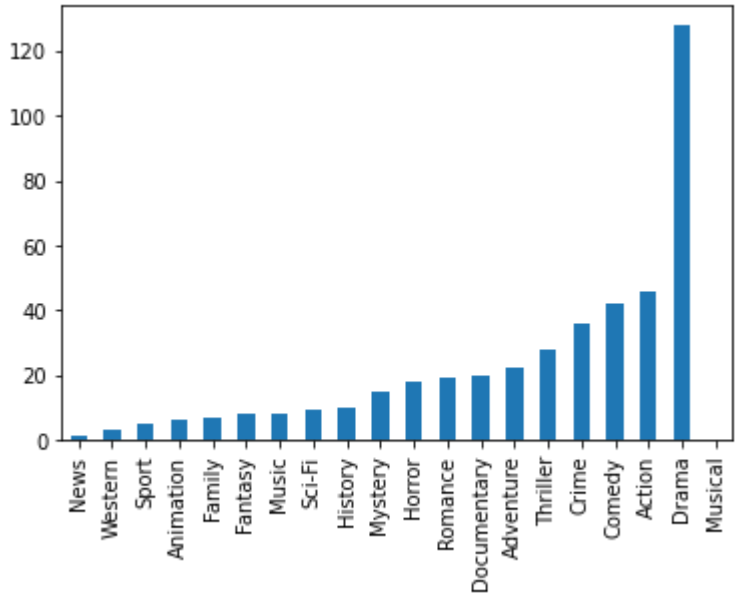
	clean_title	movie_id	primary_title	original_title	start_year	runtime_minutes	
1473	transformers: age of extinction	tt2109248	Transformers: Age of Extinction	Transformers: Age of Extinction	2014	165.0	Action,Adv
1647	finding dory	tt2277860	Finding Dory	Finding Dory	2016	97.0	Adventure,Animat
1654	jumanji: welcome to the jungle	tt2283362	Jumanji: Welcome to the Jungle	Jumanji: Welcome to the Jungle	2017	119.0	Action,Advent
1661	minions	tt2293640	Minions	Minions	2015	91.0	Adventure,Animat
1786	avengers: age of ultron	tt2395427	Avengers: Age of Ultron	Avengers: Age of Ultron	2015	141.0	Action,Adv
1997	the secret life of pets	tt2709768	The Secret Life of Pets	The Secret Life of Pets	2016	87.0	Adventure,Animat
2041	furious 7	tt2820852	Furious 7	Furious Seven	2015	137.0	Action,C
2103	zootopia	tt2948356	Zootopia	Zootopia	2016	108.0	Adventure,Animat
2336	despicable me 3	tt3469046	Despicable Me 3	Despicable Me 3	2017	89.0	Adventure,Animat
2351	captain america: civil war	tt3498820	Captain America: Civil War	Captain America: Civil War	2016	147.0	Action,Adv
2410	incredibles 2	tt3606756	Incredibles 2	Incredibles 2	2018	118.0	Action,Adventur
2464	rogue one: a star wars story	tt3748528	Rogue One: A Star Wars Story	Rogue One	2016	133.0	Action,Adv
2623	avengers: infinity war	tt4154756	Avengers: Infinity War	Avengers: Infinity War	2018	149.0	Action,Adv
2794	the fate of the furious	tt4630562	The Fate of the Furious	The Fate of the Furious	2017	136.0	Action,C
2879	jurassic world: fallen kingdom	tt4881806	Jurassic World: Fallen Kingdom	Jurassic World: Fallen Kingdom	2018	128.0	Action,Adv

20 rows × 64 columns

In [375…
```
genres_lost_money = lost_money_df[genres].apply(pd.value_counts)
```

In [380…
```
genres_lost_money.loc[True].sort_values().plot.bar()
```

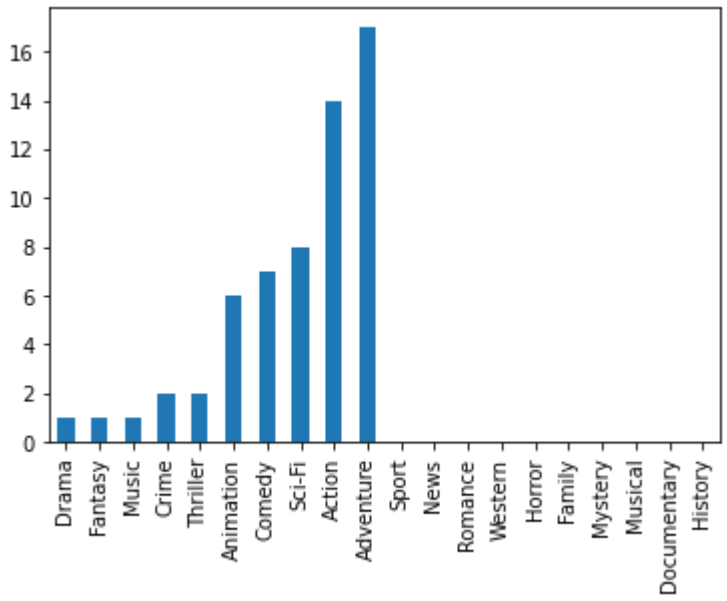
Out[380… `<AxesSubplot:>`



In [381…
```
top_20_genres = top_20_df[genres].apply(pd.value_counts)
```

In [383…
```
top_20_genres.loc[True].sort_values
().plot.bar()
```

Out[383… <AxesSubplot:>



In [530…
```
studio_2014_df = master2014_df.groupby('studio').sum().sort_values('worlwide_profit', a
studio_2014_df.index
```

Out[530…
```
Index(['BV', 'Uni.', 'Fox', 'WB', 'Sony', 'Par.', 'WB (NL)', 'LGF', 'Wein.',
       'FoxS', 'STX', 'LG/S', 'Focus', 'SGem', 'TriS', 'UTV', 'A24', 'ORF',
       'Studio 8', 'GrtIndia', 'Rela.', 'W/Dim.', 'MGM', 'SPC', 'Yash', 'Eros',
       'EOne', 'BH Tilt', 'PNT', 'RTWC', 'IFC', 'Neon', 'BST', 'EC', 'PFR',
       'Affirm', 'Orch.', 'RAtt.', 'MBox', 'Cleopatra', 'LGP', 'Trib.',
       'Annapurna', 'BBC', 'Cohen', 'Drft.', 'Global Road', 'DR', 'CE', 'XL',
       'Gold.', 'Free', 'ENTMP', 'TFA', 'Saban', 'RLJ', 'Alc', 'Amazon', 'BSC',
       'BG', 'Magn.', 'FCW', 'KE', 'VE'],
      dtype='object', name='studio')
```

In [619…
```
studio_2014_df_avg = master2014_df.groupby('studio').mean().sort_values('worlwide_profi
studio_2014_df_avg['averagerating']
```

Out[619… 8.3

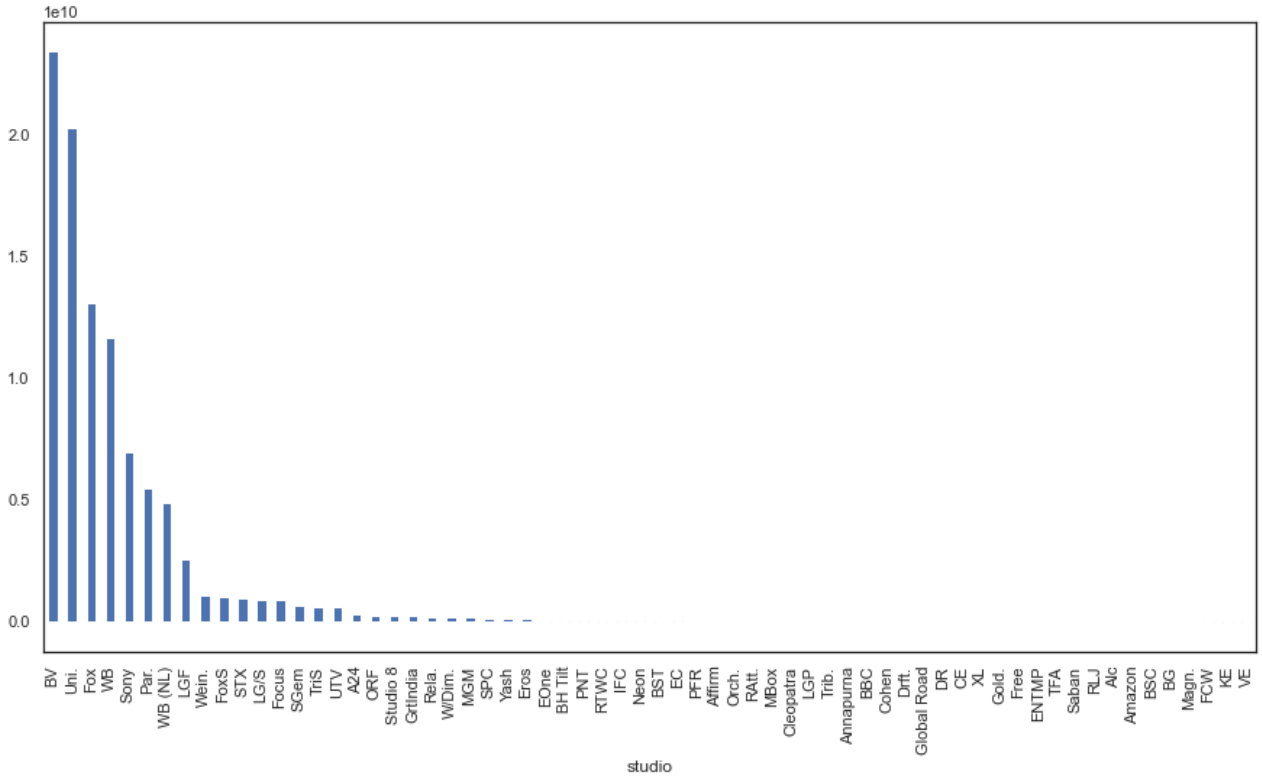
In [624…
```
studios2014_outliers_removed = studio_2014_df[10:]
studios2014_outliers_removed_mean = studio_2014_df_avg[10:]
```

In [628…
```
studios2014_outliers_removed['averagerating']
```

Out[628…
```
studio
STX            107.0
LG/S            87.7
Focus          169.0
SGem            71.9
TriS            65.7
UTV             12.6
A24            138.9
ORF            130.9
Studio 8        26.3
GrtIndia         8.3
Rela.           59.1
W/Dim.          12.9
MGM              7.2
SPC             57.8
Yash            11.6
Eros             8.0
EOne            30.2
BH Tilt         43.0
PNT             10.6
RTWC            40.6
IFC            100.9
Neon             7.5
BST             55.8
EC              17.4
PFR             16.4
Affirm          12.7
Orch.            7.9
RAtt.          141.0
MBox             7.4
Cleopatra        7.4
LGP             10.9
Trib.            6.2
Annapurna       31.8
BBC              7.3
Cohen           13.9
Drft.            6.7
Global Road      6.1
DR              23.2
CE              12.1
XL               5.7
Gold.            7.5
Free             5.8
ENTMP           19.3
TFA              5.8
Saban            6.6
RLJ             10.7
Alc              5.2
Amazon           6.8
BSC              8.1
BG              75.4
Magn.           52.9
FCW             14.4
```

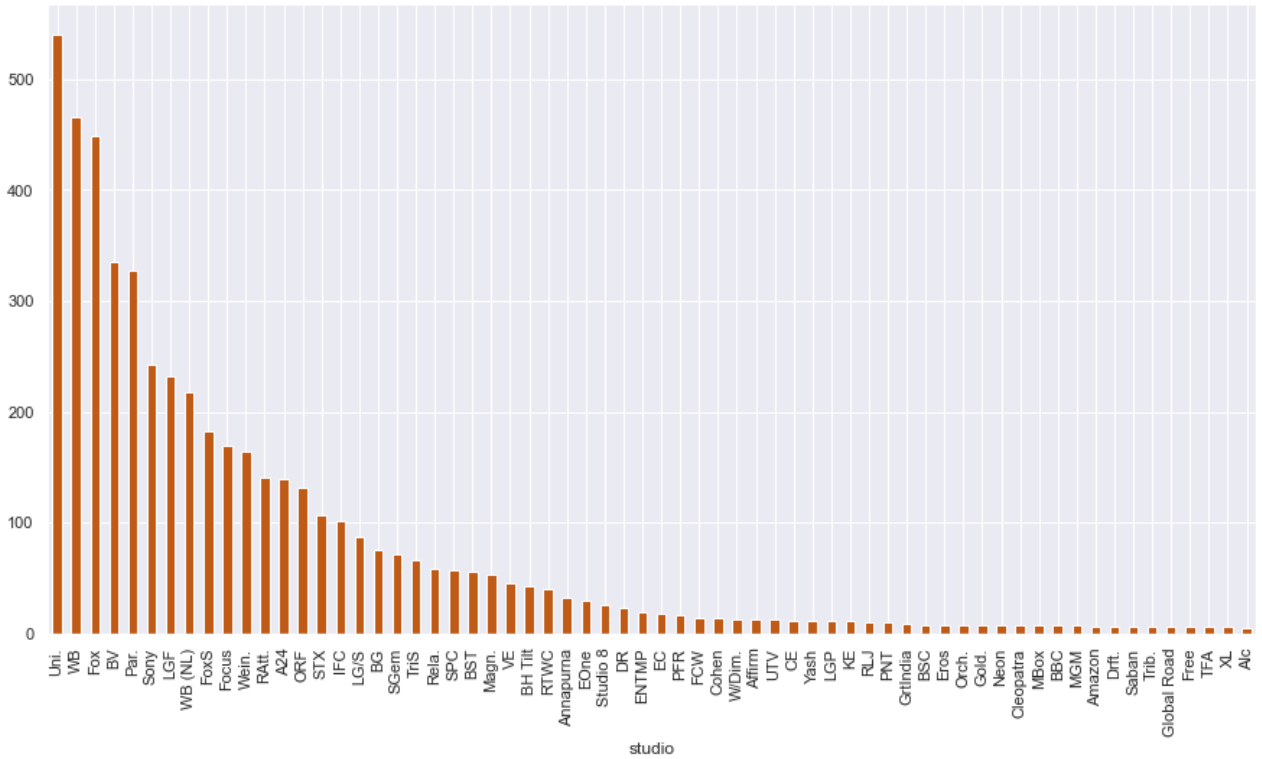
```
KE              10.9
VE              45.4
Name: averagerating, dtype: float64
```

In [591…
```
studio_2014_df['worlwide_profit'].plot.bar(figsize = (15,8))
```



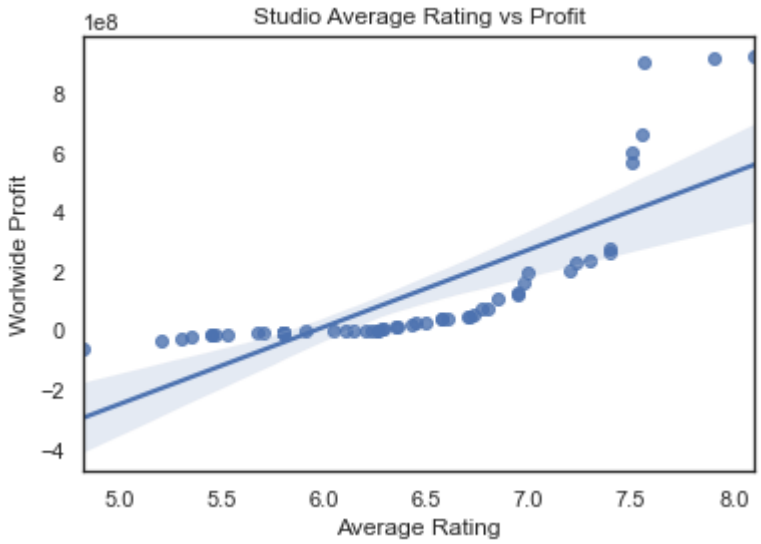
In [440…
```
studio_2014_df['averagerating'].sort_values(ascending=False).plot.bar(figsize=(15,8))
```

Out[440…
```
<AxesSubplot:xlabel='studio'>
```



In [625…
```
g = sns.regplot(studios2014_outliers_removed_mean['averagerating'].sort_values(ascendin
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

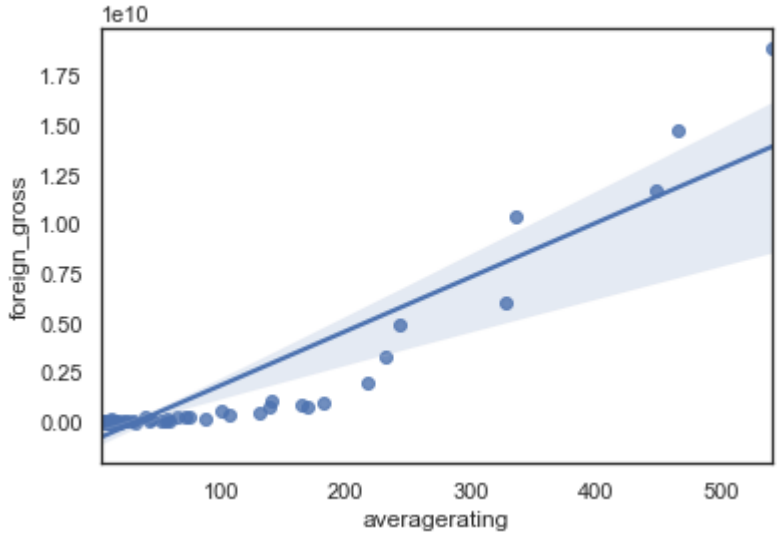


In [626…
```
studios2014_outliers_removed_mean['averagerating'].corr(studios2014_outliers_removed['w
```

Out[626…
```
-0.05224139074524369
```

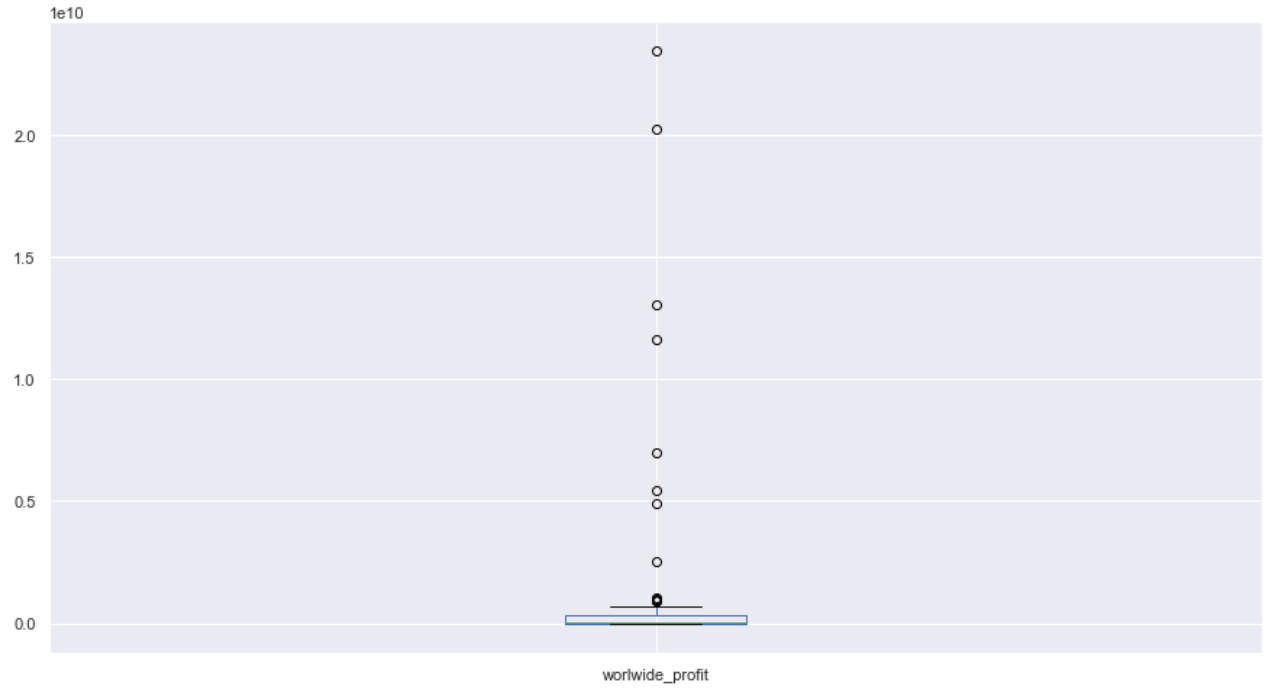
In [587…
```
sns.regplot(studio_2014_df['averagerating'].sort_values(ascending = False), studio_2014
```

Out[587…
```
<AxesSubplot:xlabel='averagerating', ylabel='foreign_gross'>
```



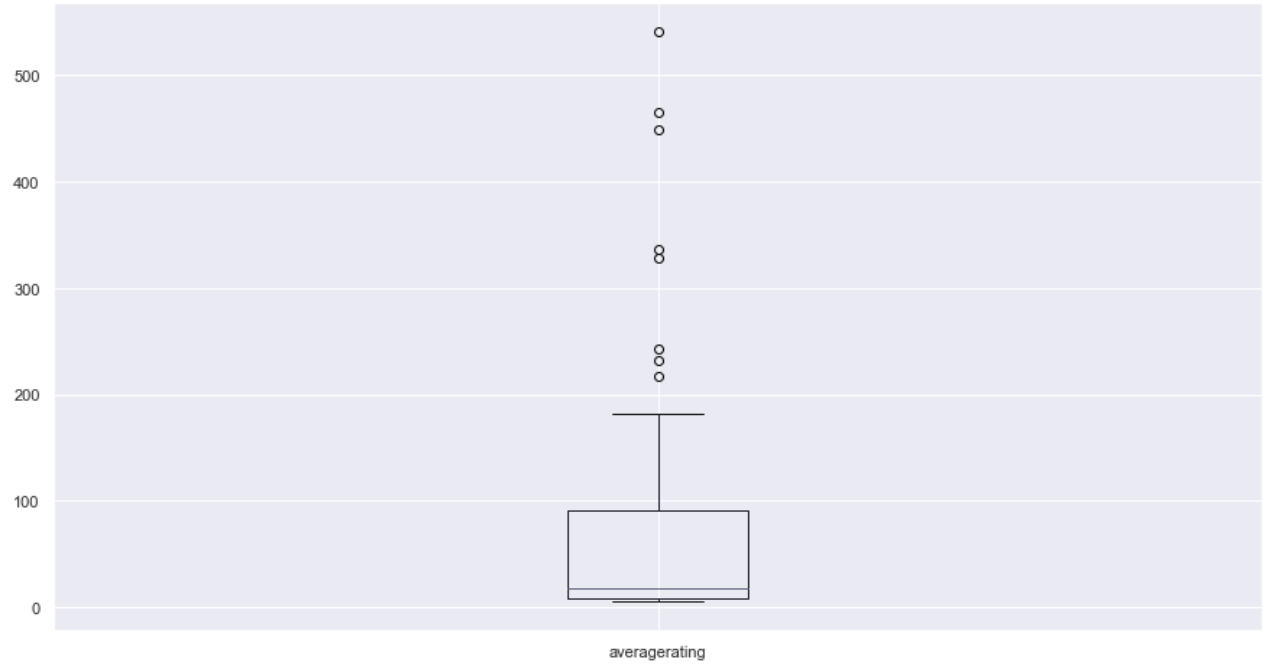
In [419…
```
studio_2014_df['worlwide_profit'].plot.box(figsize = (15,8))
```

Out[419…
```
<AxesSubplot:>
```



In [432…
```
studio_2014_df['averagerating'].plot.box(figsize = (15,8))
```

Out[432…
```
<AxesSubplot:>
```



In [413…
```
studio_2014_df['averagerating'].corr(studio_2014_df['worlwide_profit'])
```

Out[413…
```
0.8507391169433233
```

In [465…
```
studio_2014_df['averagerating'].corr(studio_2014_df['foreign_gross'])
```

Out[465…
```
0.8594084943481347
```

In [580…
```
studio_2014_df['averagerating'][10:].corr(studio_2014_df['domestic_gross'])
```

Out[580…
```
0.7360949675582893
```

In [468…
```
master2014_df['averagerating'].corr(master2014_df['domestic_gross'])
```

Out[468…
```
0.19467543823920735
```

In [ ]:
```
studio_2014_df[studio_2014_df['averagerating']]
```

In [426…
```
sns.regplot(master2014_df['production_budget'],master2014_df['worlwide_profit'])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-426-ab77908789e5> in <module>
----> 1 sns.regplot(master2014_df['production_budget'],master2014_df['worlwide_profit']
, palette='Dark2')

~\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **
kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

TypeError: regplot() got an unexpected keyword argument 'palette'
```

In [424…
```
master2014_df['production_budget'].corr(master2014_df['worlwide_profit'])
```

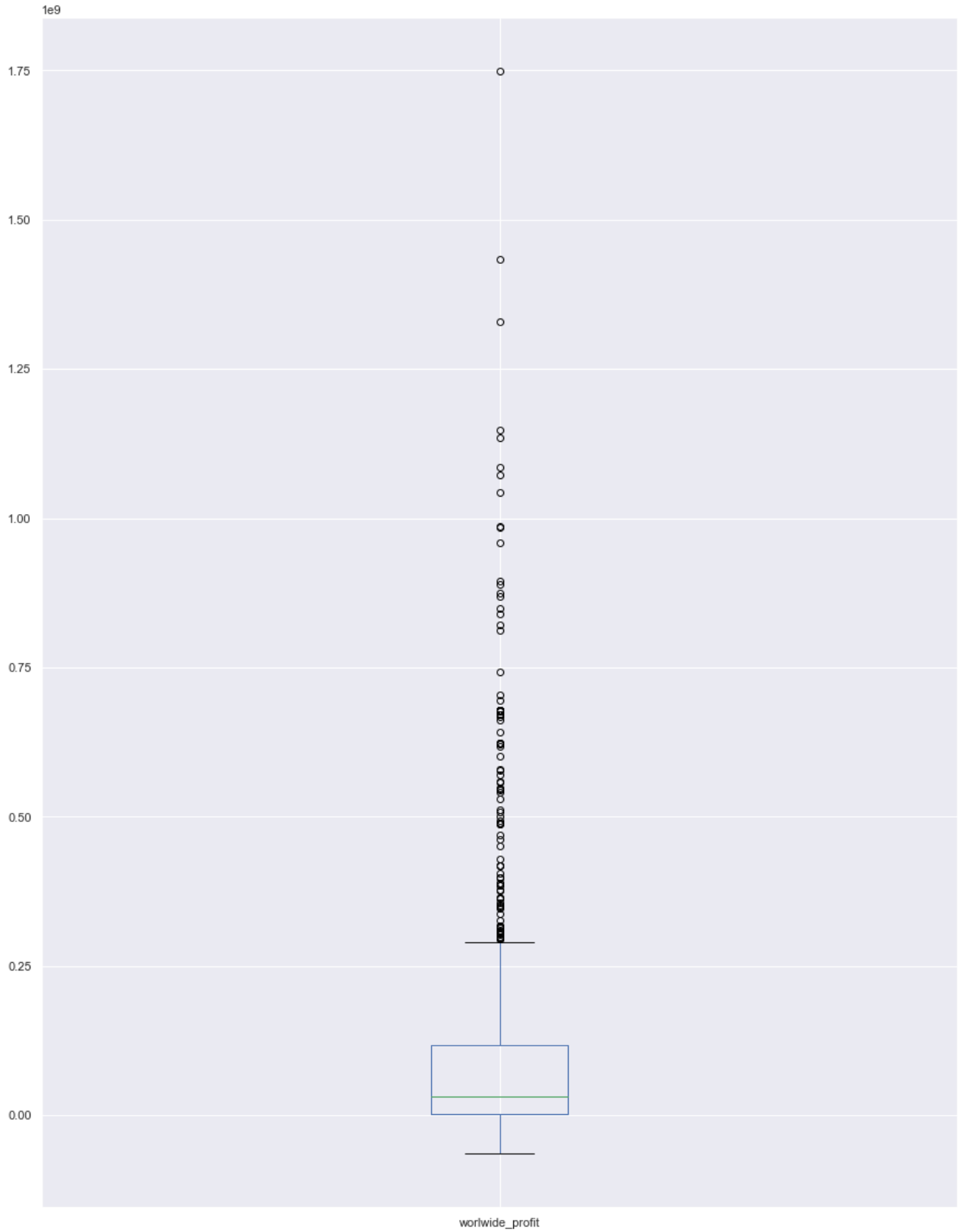
Out[424…
```
0.7013841018838237
```

In [442…
```
sns.set()
```

In [ ]:
```
no_outliers2014_df = master2014_df[]
```

In [445…
```
master2014_df['worlwide_profit'].plot.box(figsize = (15,20))
```

Out[445…
```
<AxesSubplot:>
```



In [459…
```
master2014_df['worlwide_profit'].sort_values(ascending = False)[:88].sum()
```

Out[459…
```
55728977108
```

In [460…
```
master2014_df['worlwide_profit'].sort_values(ascending = False)[88:].sum()
```

Out[460…
```
41505803959
```

In [454…
```
len(master2014_df['worlwide_profit'].sort_values(ascending = False)[60:])
```

Out[454…
```
822
```

In [461…
```
# % total profit of top 10% movies
55728977108/(55728977108 + 41505803959)
```

Out[461…
```
0.5731382998600031
```

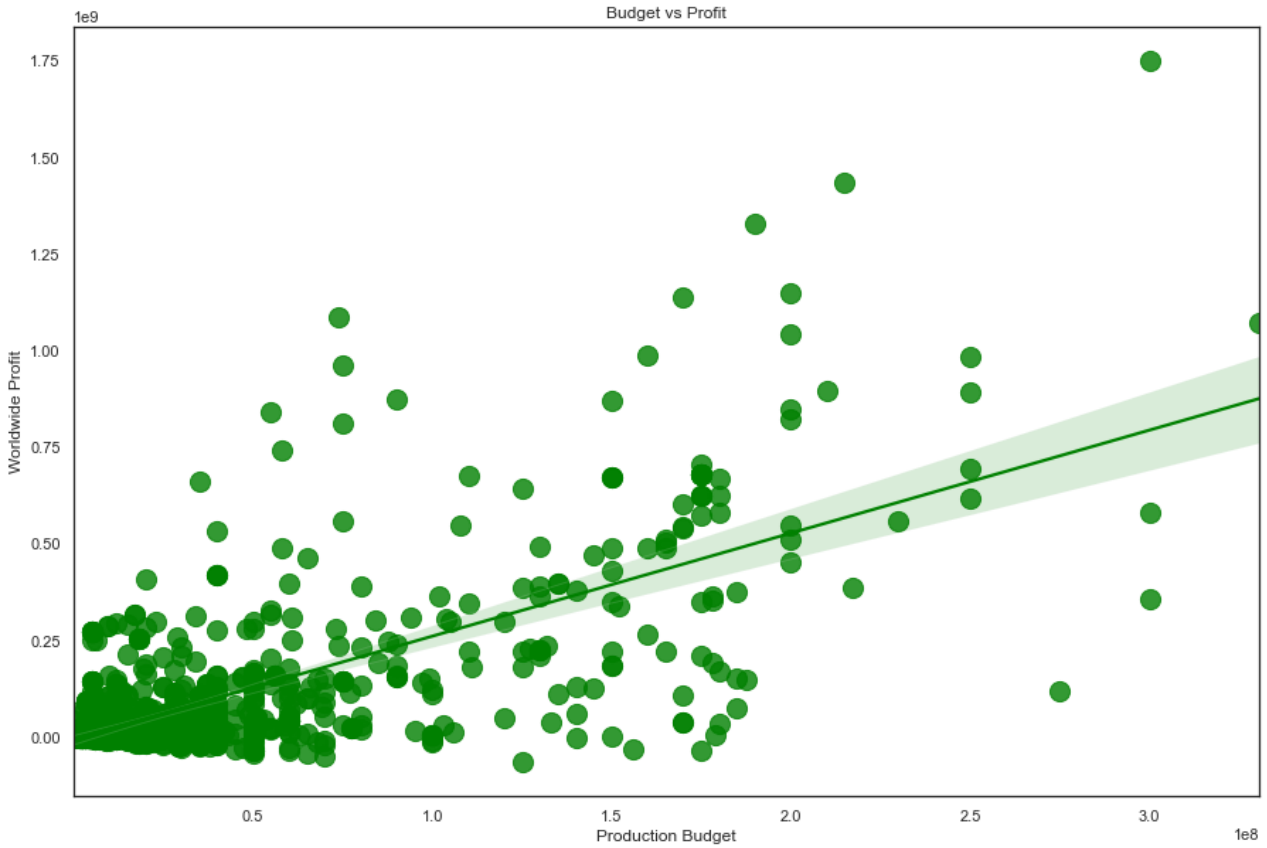
In [502…
```
# correlation of budget and global profit
master2014_df['production_budget'].corr(master2014_df['worlwide_profit'])
```

Out[502…
```
0.7013841018838237
```

In [685…
```
gr = sns.regplot(master2014_df['production_budget'],master2014_df['worlwide_profit'], c
sns.set(rc = {'figure.figsize':(15,6)})
sns.set_style('white')

;
```

```
C:\Users\Nick\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[685…
```
''
```



In [565…
```
master2014_df['averagerating']
```

Out[565…
```
6.5
```