

student

August 21, 2021

0.1 Final Project Submission

Please fill out: * Student name: Huseyin Caglar * Student pace: Part time * Scheduled project review date/time:

* Instructor name: Claude Fried * Blog post URL:

```
[1]: import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

0.2 Looking inside Data Frames

```
[2]: # Loading Data
df_movie_gross=pd.read_csv('zippedData/bom.movie_gross.csv.gz')
df_names=pd.read_csv('zippedData/imdb.name.basics.csv.gz')
df_title_akas=pd.read_csv('zippedData/imdb.title.akas.csv.gz')
df_titles=pd.read_csv('zippedData/imdb.title.basics.csv.gz')
df_crew=pd.read_csv('zippedData/imdb.title.crew.csv.gz')
df_principals=pd.read_csv('zippedData/imdb.title.principals.csv.gz')
df_ratings=pd.read_csv('zippedData/imdb.title.ratings.csv.gz')
df_movie_info=pd.read_csv('zippedData/rt.movie_info.tsv.
    ↪gz',sep='\t',encoding='latin1')
df_reviews=pd.read_csv('zippedData/rt.reviews.tsv.
    ↪gz',sep='\t',encoding='latin1')
df_movies=pd.read_csv('zippedData/tmdb.movies.csv.gz')
df_budgets=pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
```

0.2.1 Movie Gross

```
[3]: df_movie_gross.head()
```

```
[3]:
```

	title	studio	domestic_gross	\
0	Toy Story 3	BV	415000000.0	
1	Alice in Wonderland (2010)	BV	334200000.0	
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	
3	Inception	WB	292600000.0	

4	Shrek Forever After	P/DW	238700000.0
---	---------------------	------	-------------

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

0.2.2 Names

```
[4]: df_names.head()
```

```
[4]:      nconst      primary_name  birth_year  death_year  \
0  nm0061671  Mary Ellen Bauder          NaN          NaN
1  nm0061865    Joseph Bauer          NaN          NaN
2  nm0062070    Bruce Baum          NaN          NaN
3  nm0062195    Axel Baumann          NaN          NaN
4  nm0062798    Pete Baxter          NaN          NaN

      primary_profession  \
0  miscellaneous,production_manager,producer
1  composer,music_department,sound_department
2  miscellaneous,actor,writer
3  camera_department,cinematographer,art_department
4  production_designer,art_department,set_decorator

      known_for_titles
0  tt0837562,tt2398241,tt0844471,tt0118553
1  tt0896534,tt6791238,tt0287072,tt1682940
2  tt1470654,tt0363631,tt0104030,tt0102898
3  tt0114371,tt2004304,tt1618448,tt1224387
4  tt0452644,tt0452692,tt3458030,tt2178256
```

0.2.3 Title Akas

```
[5]: df_title_akas.head()
```

```
[5]:      title_id  ordering      title region  \
0  tt0369610      10          BG
1  tt0369610      11  Jurashikku warudo  JP
2  tt0369610      12  Jurassic World: 0 Mundo dos Dinossauros  BR
3  tt0369610      13          0 Mundo dos Dinossauros  BR
4  tt0369610      14          Jurassic World  FR

      language      types  attributes  is_original_title
0          bg          NaN          NaN              0.0
```

1	NaN	imdbDisplay	NaN	0.0
2	NaN	imdbDisplay	NaN	0.0
3	NaN	NaN	short title	0.0
4	NaN	imdbDisplay	NaN	0.0

0.2.4 Titles

```
[6]: df_titles.head()
```

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

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

0.2.5 Crew

```
[7]: df_crew.head()
```

```
[7]:      tconst      directors      writers
0  tt0285252      nm0899854      nm0899854
1  tt0438973          NaN  nm0175726, nm1802864
2  tt0462036      nm1940585      nm1940585
3  tt0835418      nm0151540  nm0310087, nm0841532
4  tt0878654  nm0089502, nm2291498, nm2292011      nm0284943
```

0.2.6 Principals

```
[8]: df_principals.head()
```

```
[8]:      tconst  ordering  nconst  category  job      characters
0  tt0111414         1  nm0246005    actor    NaN  ["The Man"]
1  tt0111414         2  nm0398271  director    NaN          NaN
2  tt0111414         3  nm3739909  producer  producer          NaN
3  tt0323808        10  nm0059247    editor    NaN          NaN
4  tt0323808         1  nm3579312  actress    NaN  ["Beth Boothby"]
```

0.2.7 Ratings

```
[9]: df_ratings.head()
```

```
[9]:      tconst  averagerating  numvotes
0  tt10356526           8.3         31
1  tt10384606           8.9        559
2   tt1042974           6.4         20
3  tt1043726           4.2       50352
4   tt1060240           6.5         21
```

0.2.8 Movie Info

```
[10]: df_movie_info.head()
```

```
[10]:      id      synopsis rating \
0    1  This gritty, fast-paced, and innovative police...      R
1    3  New York City, not-too-distant-future: Eric Pa...      R
2    5  Illeana Douglas delivers a superb performance ...      R
3    6  Michael Douglas runs afoul of a treacherous su...      R
4    7                                     NaN      NR

      genre      director \
0  Action and Adventure|Classics|Drama  William Friedkin
1    Drama|Science Fiction and Fantasy  David Cronenberg
2    Drama|Musical and Performing Arts   Allison Anders
3      Drama|Mystery and Suspense     Barry Levinson
4      Drama|Romance       Rodney Bennett

      writer  theater_date  dvd_date  currency \
0   Ernest Tidyman   Oct 9, 1971  Sep 25, 2001     NaN
1  David Cronenberg|Don DeLillo  Aug 17, 2012   Jan 1, 2013      $
2   Allison Anders   Sep 13, 1996  Apr 18, 2000     NaN
3  Paul Attanasio|Michael Crichton  Dec 9, 1994  Aug 27, 1997     NaN
4      Giles Cooper           NaN           NaN     NaN

      box_office  runtime      studio
0           NaN  104 minutes           NaN
1    600,000  108 minutes  Entertainment One
2           NaN  116 minutes           NaN
3           NaN  128 minutes           NaN
4           NaN  200 minutes           NaN
```

0.2.9 Reviews

```
[11]: df_reviews.head()
```

```
[11]: id review rating fresh \
0 3 A distinctly gallows take on contemporary fina... 3/5 fresh
1 3 It's an allegory in search of a meaning that n... NaN rotten
2 3 ... life lived in a bubble in financial dealin... NaN fresh
3 3 Continuing along a line introduced in last yea... NaN fresh
4 3 ... a perverse twist on neorealism... NaN fresh

critic top_critic publisher date
0 PJ Nabarro 0 Patrick Nabarro November 10, 2018
1 Annalee Newitz 0 io9.com May 23, 2018
2 Sean Axmaker 0 Stream on Demand January 4, 2018
3 Daniel Kasman 0 MUBI November 16, 2017
4 NaN 0 Cinema Scope October 12, 2017
```

0.2.10 Movies

```
[12]: df_movies.head()
```

```
[12]: Unnamed: 0 genre_ids id original_language \
0 0 [12, 14, 10751] 12444 en
1 1 [14, 12, 16, 10751] 10191 en
2 2 [12, 28, 878] 10138 en
3 3 [16, 35, 10751] 862 en
4 4 [28, 878, 12] 27205 en

original_title popularity release_date \
0 Harry Potter and the Deathly Hallows: Part 1 33.533 2010-11-19
1 How to Train Your Dragon 28.734 2010-03-26
2 Iron Man 2 28.515 2010-05-07
3 Toy Story 28.005 1995-11-22
4 Inception 27.920 2010-07-16

title vote_average vote_count
0 Harry Potter and the Deathly Hallows: Part 1 7.7 10788
1 How to Train Your Dragon 7.7 7610
2 Iron Man 2 6.8 12368
3 Toy Story 7.9 10174
4 Inception 8.3 22186
```

0.2.11 Budgets

```
[13]: df_budgets.head()
```

```
[13]: id release_date movie \
0 1 Dec 18, 2009 Avatar
1 2 May 20, 2011 Pirates of the Caribbean: On Stranger Tides
2 3 Jun 7, 2019 Dark Phoenix
```

3	4	May 1, 2015	Avengers: Age of Ultron
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi

	production_budget	domestic_gross	worldwide_gross
0	\$425,000,000	\$760,507,625	\$2,776,345,279
1	\$410,600,000	\$241,063,875	\$1,045,663,875
2	\$350,000,000	\$42,762,350	\$149,762,350
3	\$330,600,000	\$459,005,868	\$1,403,013,963
4	\$317,000,000	\$620,181,382	\$1,316,721,747

```
[14]: ## tight_layout()
```

0.3 Getting Inside of Business Problem

1. Which studios has the most revenue movies?
2. What type of movies most made?
3. Is there a good relation with movie budget and revenue?
4. Which directors made most revenue and why?

0.3.1 Studios

```
[15]: #Remembering first dataframe for what is data look like
df_movie_gross.head()
```

```
[15]:
```

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

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

```
[16]: #Looking for missing data if there is any.
df_movie_gross.isna().sum()
```

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

```
[17]: # Cleaning proccess of 'foreign_gross' column.
df_movie_gross['foreign_gross'] = df_movie_gross['foreign_gross'].str.
      ↪replace(',','').astype(float)

[18]: # Creating new column for total gross and sorting.
df_movie_gross['total_gross'] = df_movie_gross['domestic_gross']+_
      ↪df_movie_gross['foreign_gross']

[19]: #Grouping and sorting dataframe by studio to look studios individually.
grouped_studio = df_movie_gross.groupby('studio').sum().reset_index()
grouped_studio.sort_values(by='total_gross',ascending=False,inplace=True)

[20]: #Creating variable for top 20 studios with most revenue.
most_gross_20_studio = grouped_studio[:20]

[21]: most_gross_20_studio
```

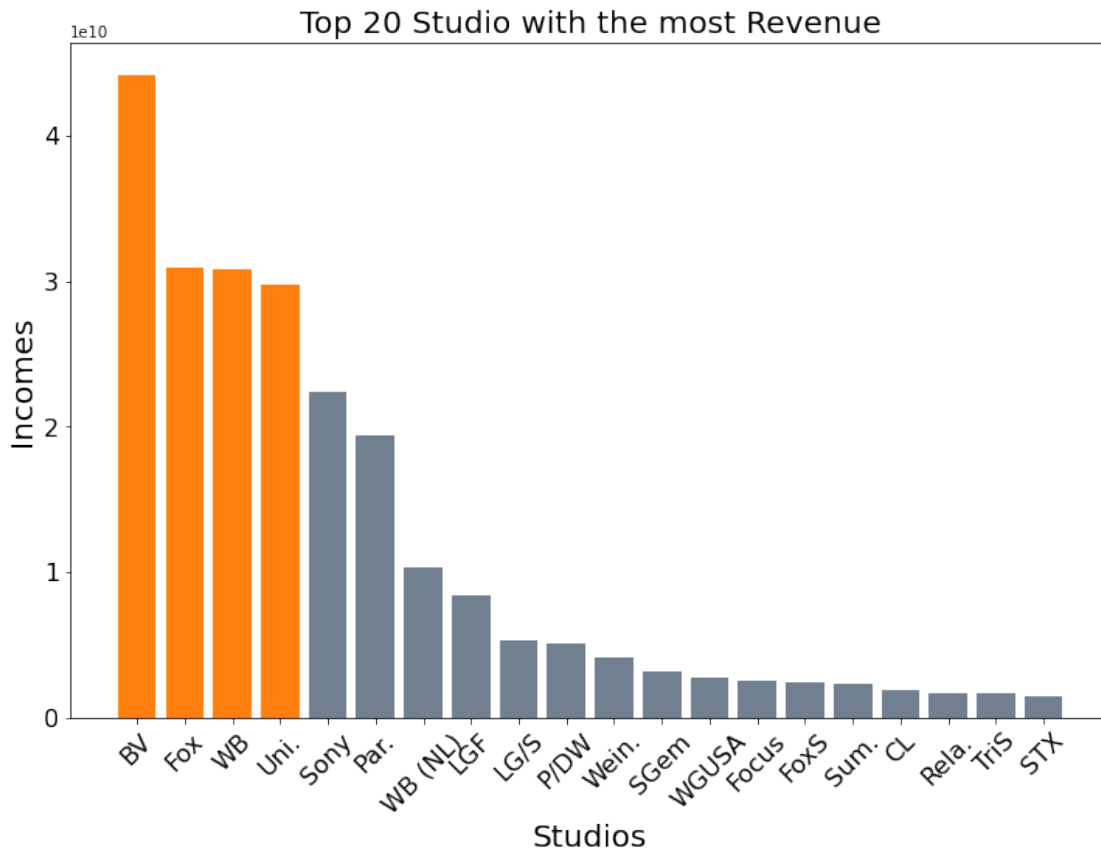
	studio	domestic_gross	foreign_gross	year	total_gross
36	BV	1.841903e+10	2.579385e+10	213451	4.419038e+10
93	Fox	1.094950e+10	2.005587e+10	273882	3.098037e+10
246	WB	1.216805e+10	1.866790e+10	281941	3.079150e+10
238	Uni.	1.290239e+10	1.685477e+10	296082	2.974681e+10
215	Sony	8.459683e+09	1.394535e+10	221575	2.240472e+10
185	Par.	7.685871e+09	1.186338e+10	203417	1.944420e+10
247	WB (NL)	3.995700e+09	6.339000e+09	90644	1.031410e+10
134	LGF	4.118963e+09	4.482619e+09	207437	8.467471e+09
133	LG/S	2.078200e+09	3.353724e+09	82599	5.318924e+09
171	P/DW	1.682900e+09	3.393600e+09	20109	5.076500e+09
251	Wein.	1.540550e+09	2.624086e+09	155022	4.095903e+09
205	SGem	1.526400e+09	1.624062e+09	70462	3.140162e+09
248	WGUSA	2.539460e+07	2.761447e+09	116902	2.778054e+09
92	Focus	1.172041e+09	1.369969e+09	120844	2.496769e+09
94	FoxS	1.061832e+09	1.497388e+09	134904	2.474688e+09
219	Sum.	9.318710e+08	1.354900e+09	30158	2.284971e+09
48	CL	1.820020e+07	2.005700e+09	149049	1.898686e+09
196	Rela.	9.432940e+08	8.228780e+08	70454	1.715417e+09
229	TriS	9.709000e+08	8.849550e+08	46320	1.713055e+09
210	STX	7.521000e+08	7.462000e+08	48406	1.474200e+09

```
[22]: #Creating for loop to highlight first three value of graph.
colors=[]
for studio in most_gross_20_studio['studio']:
    if ((studio=='BV') or (studio=='WB') or (studio=='Uni.') or (studio=='Fox')):
        colors.append('tab:orange')
    else:
        colors.append('slategray')
```

```

fig , ax = plt.subplots(figsize=(12,8))
ax.bar(most_gross_20_studio['studio'],most_gross_20_studio['total_gross'],color_
    ↪= colors)
ax.set_title('Top 20 Studio with the most Revenue',fontsize=20)
ax.tick_params(axis='x', labels=16,rotation=45)
ax.tick_params(axis='y', labels=16)
plt.xlabel('Studios',fontsize=20)
plt.ylabel('Incomes',fontsize=20)
plt.style.use('ggplot')

```



```

[23]: #Merging to dataframe to look studio and popularity and sorting by popularity.
df_popularity = df_movie_gross.merge(df_movies)
popularity_grouped = df_popularity.groupby('studio').sum().reset_index()
popularity_sorted = popularity_grouped.
    ↪sort_values(by='popularity',ascending=False)
#Creating colors to highlight first three popular ones.
colors=[]
for studio in popularity_sorted[:20]['studio']:
    if ((studio == 'Uni.') or (studio == 'Fox') or (studio == 'BV') or (studio_
    ↪== 'WB')):

```



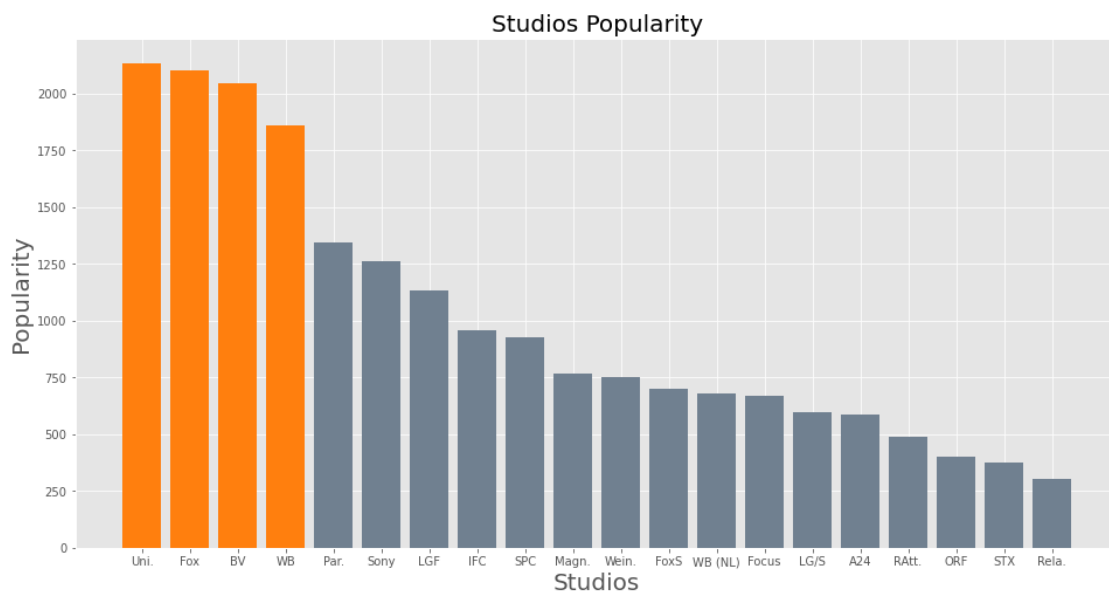
```

        colors.append('tab:orange')
    else:
        colors.append('slategray')

fig , ax = plt.subplots(figsize=(16,8))
ax.bar(popularity_sorted[:20]['studio'],popularity_sorted[:
    ↳20]['popularity'],color=colors)
ax.set_title('Studios Popularity', fontsize=20)
ax.set_xlabel('Studios', fontsize=20)
ax.set_ylabel('Popularity', fontsize=20)

```

[23]: Text(0, 0.5, 'Popularity')



0.3.2 Movie types

[24]: *# Need to some changing on column name. So both dataframes could have same_*
↳column name to merge on(`title`column).
`df_titles.rename(columns={'primary_title':'title'},inplace=True)`
`df_gross_and_titles = df_movie_gross.merge(df_titles)`

[25]: *#Cleaning and sorting process of genres column.*
`clean_genres = df_gross_and_titles['genres'].str.get_dummies(',').sum()`
`sorted_genres = clean_genres.sort_values(ascending=False)`

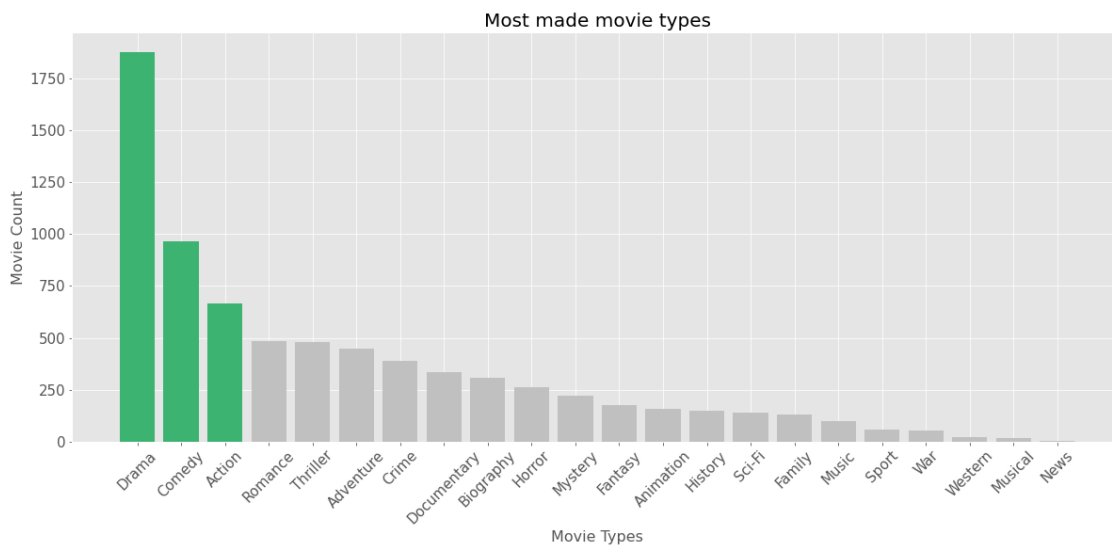
`colors=[]`
`[colors.append('mediumseagreen') if((i_`
↳=='Drama')or(i=='Action')or(i=='Comedy'))

```

else colors.append('silver')for i in sorted_genres.index]

fig , ax = plt.subplots(figsize=(20,8))
ax.bar(sorted_genres.index,sorted_genres,color=colors)
ax.set_title('Most made movie types',fontsize=20)
ax.set_xlabel('Movie Types',fontsize=16)
ax.set_ylabel('Movie Count',fontsize=16)
plt.tick_params(labelsize = 15)
plt.xticks(rotation=45)
plt.show()

```



```

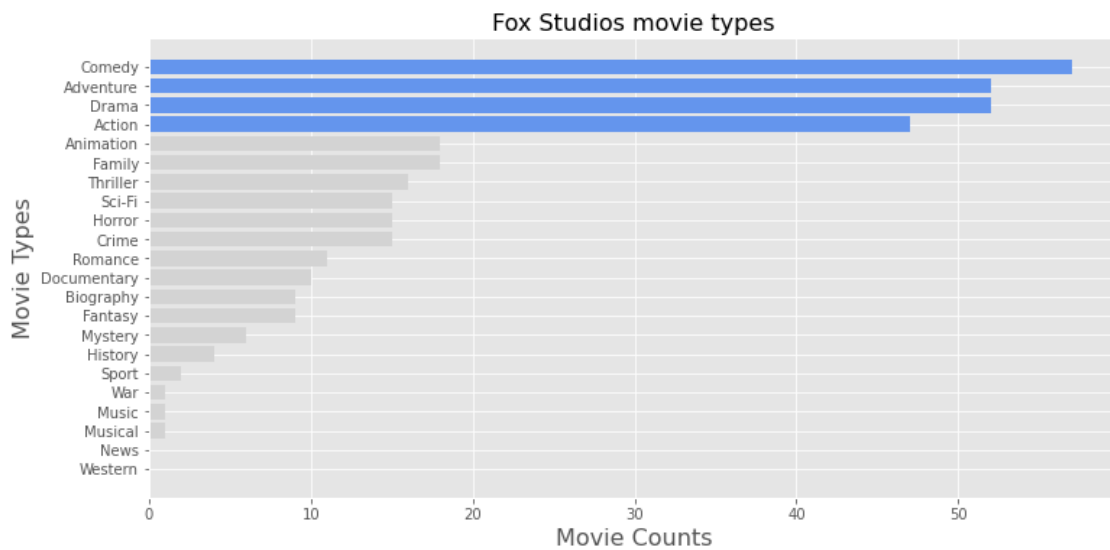
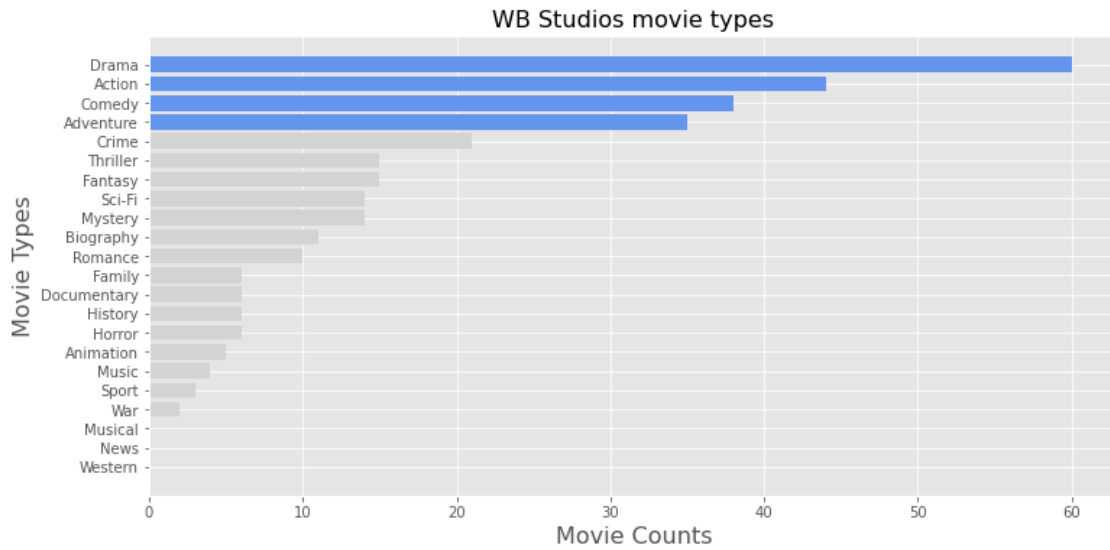
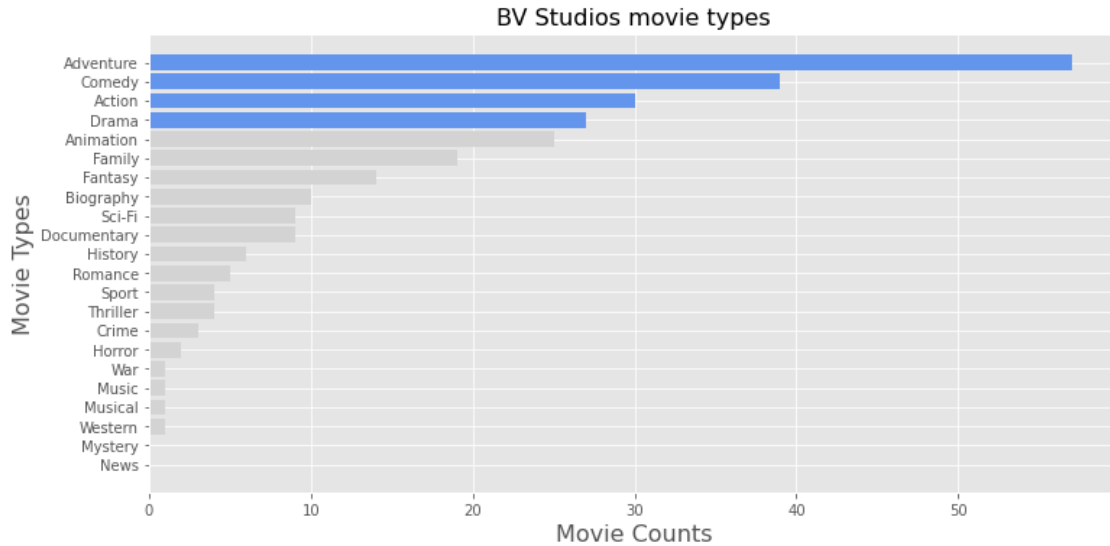
[26]: # Need to seperate multiple value of genres column as down below line and sort
      ↪ them to visualize.
grouped_genres = df_gross_and_titles['genres'].str.get_dummies(sep=',').
      ↪groupby(df_gross_and_titles['studio']).sum()
sorted_values_BV = grouped_genres.T['BV'].sort_values()
sorted_values_WB = grouped_genres.T['WB'].sort_values()
sorted_values_Fox = grouped_genres.T['Fox'].sort_values()

#Creating colors to highlight most made movie types.
colors=[]
[colors.append('lightgray') if i<=29 else colors.append('cornflowerblue')for i
  ↪in sorted_values_WB]

fig , axs = plt.subplots(3,figsize=(12,8))
axs[0].barh(sorted_values_BV.index,sorted_values_BV,color=colors)
axs[0].set_title('BV Studios movie types',fontsize=16)
axs[0].set_xlabel('Movie Counts',fontsize=16)

```

```
axs[0].set_ylabel('Movie Types',fontsize=16)
axs[1].barh(sorted_values_WB.index,sorted_values_WB,color=colors)
axs[1].set_title('WB Studios movie types',fontsize=16)
axs[1].set_xlabel('Movie Counts',fontsize=16)
axs[1].set_ylabel('Movie Types',fontsize=16)
axs[2].barh(sorted_values_Fox.index,sorted_values_Fox,color=colors)
axs[2].set_title('Fox Studios movie types',fontsize=16)
axs[2].set_xlabel('Movie Counts',fontsize=16)
axs[2].set_ylabel('Movie Types',fontsize=16)
fig.subplots_adjust(top=2)
```



0.3.3 Budget Relations

```
[27]: #Cleaning process of production_budget column.
df_budgets['production_budget'] = df_budgets['production_budget'].str.
      ↪replace(',', '')
df_budgets['production_budget'] = df_budgets['production_budget'].str.
      ↪replace('$', '').astype(float)

[28]: #Cleaning process of worldwide_gross column.
df_budgets['worldwide_gross'] = df_budgets['worldwide_gross'].str.
      ↪replace(',', '')
df_budgets['worldwide_gross'] = df_budgets['worldwide_gross'].str.
      ↪replace('$', '').astype(float)

[29]: #Sorted dataframe by descending worldwide_gross column.
df_budgets.sort_values(by='worldwide_gross',ascending=False,inplace=True)

[30]: #Creating function to create 3 category.
def split_budget(budget):
    if budget < 1e8:
        return 'Under 100 million.'
    if budget < 2e8:
        return 'Under 200 million.'
    return 'Over 200 million.'

[31]: #Using function into the dataframe to create new column and categorize values.
df_budgets['split_budget'] = df_budgets['production_budget'].apply(split_budget)

[32]: #Looking again data.
df_budgets
```



```
[32]:
```

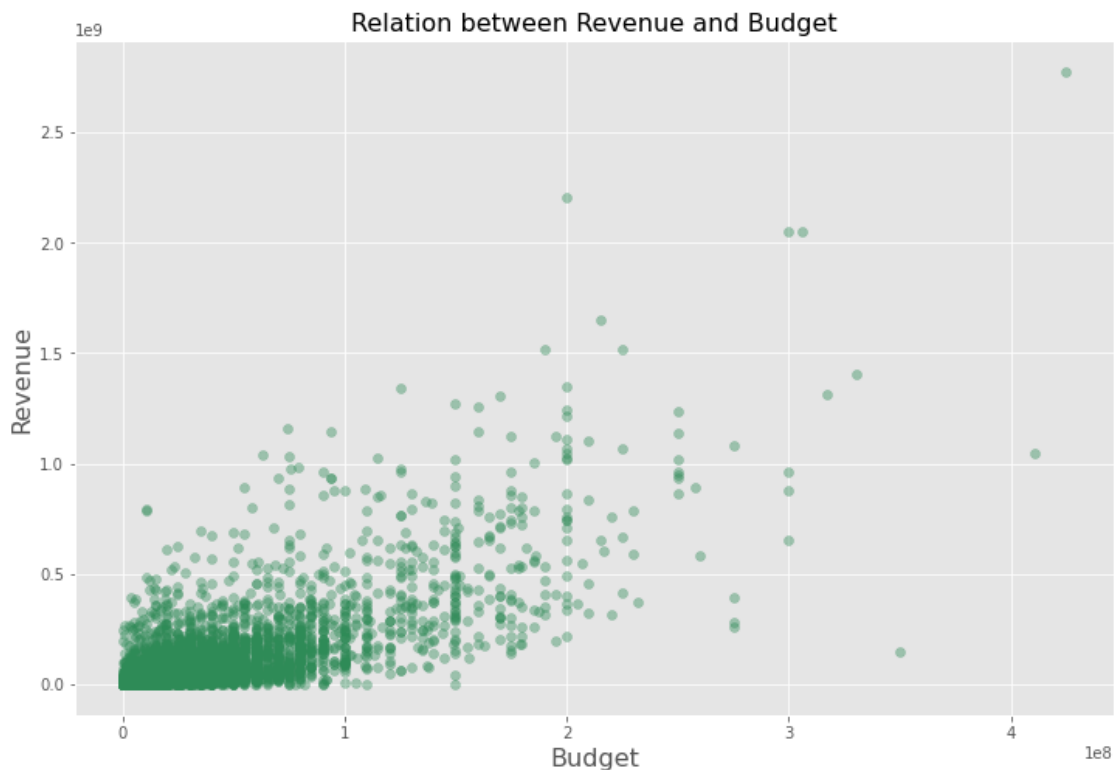
	id	release_date	movie	\
0	1	Dec 18, 2009	Avatar	
42	43	Dec 19, 1997	Titanic	
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	
6	7	Apr 27, 2018	Avengers: Infinity War	
33	34	Jun 12, 2015	Jurassic World	
...	
5474	75	Dec 31, 2005	Insomnia Manica	
5473	74	Jul 17, 2012	Girls Gone Dead	
5472	73	Apr 3, 2012	Enter Nowhere	
5471	72	Dec 31, 2010	Drones	
4068	69	Dec 12, 2008	The Kings of Appletown	

	production_budget	domestic_gross	worldwide_gross	split_budget
0	425000000.0	\$760,507,625	2.776345e+09	Over 200 million.
42	200000000.0	\$659,363,944	2.208208e+09	Over 200 million.
5	306000000.0	\$936,662,225	2.053311e+09	Over 200 million.
6	300000000.0	\$678,815,482	2.048134e+09	Over 200 million.
33	215000000.0	\$652,270,625	1.648855e+09	Over 200 million.
...
5474	500000.0	\$0	0.000000e+00	Under 100 million.
5473	500000.0	\$0	0.000000e+00	Under 100 million.
5472	500000.0	\$0	0.000000e+00	Under 100 million.
5471	500000.0	\$0	0.000000e+00	Under 100 million.
4068	7000000.0	\$0	0.000000e+00	Under 100 million.

[5782 rows x 7 columns]

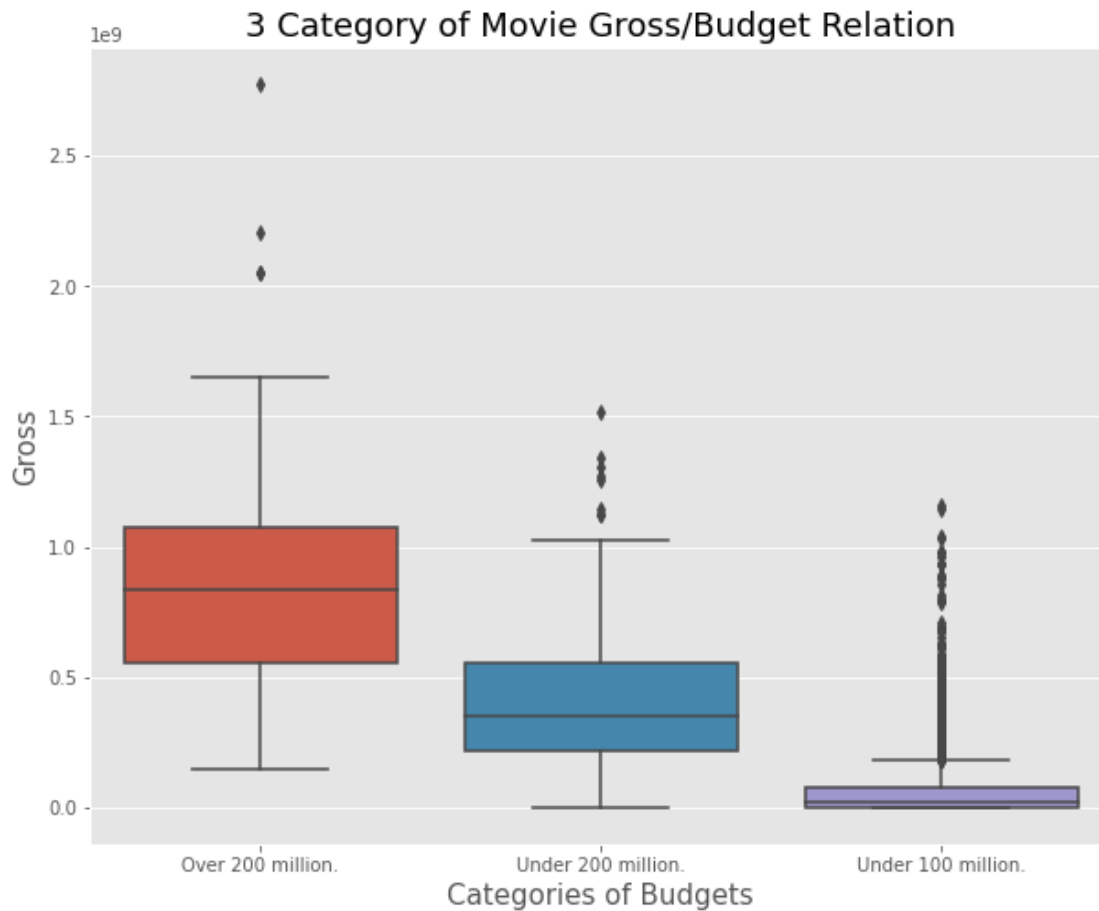
[33]: *#Creating scatterplot to see relationship between budget and gross.*

```
fig , ax = plt.subplots(figsize=(12,8))
ax.scatter(df_budgets['production_budget'],df_budgets['worldwide_gross'],
↪,c='seagreen',alpha=0.4)
ax.set_title('Relation between Revenue and Budget',fontsize=16)
ax.set_xlabel('Budget',fontsize=16)
ax.set_ylabel('Revenue',fontsize=16)
plt.show()
```



```
[34]: #Creating boxplot for 3 different budget category.
```

```
fig, ax = plt.subplots(figsize = (10,8))
ax = sns.boxplot(x='split_budget',y='worldwide_gross',data=df_budgets)
ax.set_xlabel('Categories of Budgets', fontsize = 15)
ax.set_ylabel ('Gross', fontsize = 15)
ax.set_title('3 Category of Movie Gross/Budget Relation', fontsize = 18)
plt.show()
```



0.3.4 Directors

```
[35]: #Merging names and principals dataframes.
```

```
df_names_principals = df_names.merge(df_principals)
df_names_principals
```

```

[35]:          nconst          primary_name  birth_year  death_year  \
0      nm0061671      Mary Ellen Bauder          NaN          NaN
1      nm0061865          Joseph Bauer          NaN          NaN
2      nm0061865          Joseph Bauer          NaN          NaN
3      nm0061865          Joseph Bauer          NaN          NaN
4      nm0061865          Joseph Bauer          NaN          NaN
...
1027907  nm9990381          Susan Grobes          NaN          NaN
1027908  nm9990690          Joo Yeon So          NaN          NaN
1027909  nm9991320          Madeline Smith          NaN          NaN
1027910  nm9991786  Michelle Modigliani          NaN          NaN
1027911  nm9993380          Pegasus Envoyé          NaN          NaN

          primary_profession  \
0      miscellaneous,production_manager,producer
1      composer,music_department,sound_department
2      composer,music_department,sound_department
3      composer,music_department,sound_department
4      composer,music_department,sound_department
...
1027907          actress
1027908          actress
1027909          actress
1027910          producer
1027911      director,actor,writer

          known_for_titles          tconst  ordering  \
0      tt0837562,tt2398241,tt0844471,tt0118553  tt2398241          9
1      tt0896534,tt6791238,tt0287072,tt1682940  tt0433397          7
2      tt0896534,tt6791238,tt0287072,tt1682940  tt1681372          8
3      tt0896534,tt6791238,tt0287072,tt1682940  tt2387710          8
4      tt0896534,tt6791238,tt0287072,tt1682940  tt2281215          7
...
1027907          NaN  tt6527982          2
1027908          tt9090932,tt8737130  tt8737130          4
1027909          tt8734436,tt9615610  tt8734436          3
1027910          NaN  tt8739240          9
1027911          tt8743182  tt8743182          5

          category          job          characters
0      producer  producer          NaN
1      composer          NaN          NaN
2      composer          NaN          NaN
3      composer          NaN          NaN
4      composer          NaN          NaN
...
1027907  actress          NaN  ["Cheryl","Gypsy"]

```


1027908	actress	NaN	NaN
1027909	actress	NaN	["Anna"]
1027910	producer	producer	NaN
1027911	director	NaN	NaN

[1027912 rows x 11 columns]

```
[36]: #Dropping unnecessary and missing columns.
df_names_principals.
    →drop(['primary_profession','birth_year','death_year','known_for_titles','job','characters'],axis=1,inplace=True)
```

```
[37]: #Creating directors dataframe.
df_directors = df_names_principals.
    →loc[df_names_principals['category']=='director']
```

```
[38]: df_directors
```

```
[38]:
```

	nconst	primary_name	tconst	ordering	category
12	nm0062879	Ruel S. Bayani	tt2057445	5	director
13	nm0062879	Ruel S. Bayani	tt1592569	5	director
14	nm0062879	Ruel S. Bayani	tt2590280	5	director
15	nm0062879	Ruel S. Bayani	tt8421806	5	director
48	nm0064023	Bryan Beasley	tt3501180	2	director
...
1027881	nm9971456	Zheng Wei	tt8697720	1	director
1027889	nm9980896	Rama Narayanan	tt8715016	5	director
1027890	nm9980896	Rama Narayanan	tt8919136	5	director
1027891	nm9981679	Samir Eshra	tt8717234	1	director
1027911	nm9993380	Pegasus Envoyé	tt8743182	5	director

[146393 rows x 5 columns]

```
[39]: #Merging to dataframes to merge another one.
titles_directors = df_directors.merge(df_titles)

#Renaming column name to merge on df_budgets dataframe.
titles_directors.rename(columns={'title':'movie'},inplace=True)

#Merging third dataframe.
budget_and_directors = titles_directors.merge(df_budgets)
```

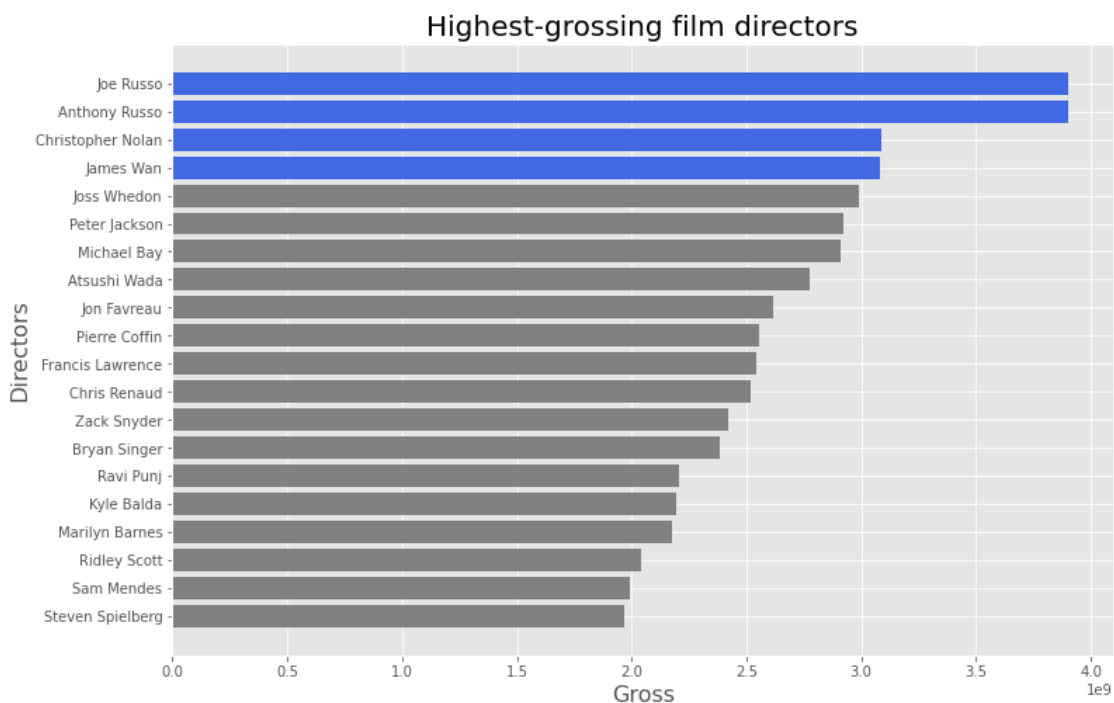
```
[40]: #Sorting dataframe by descending worldwide gross.
budget_and_directors.
    →sort_values(by='worldwide_gross',ascending=False,inplace=True)
```

```
[48]: grouped_directors = budget_and_directors.groupby('primary_name').sum()
sorted_directors = grouped_directors.
    ↳sort_values(by='worldwide_gross',ascending=True)

colors=[]
[colors.append('royalblue') if ((i == 'Anthony Russo')or (i=='Joe_
    ↳Russo')or(i=='Christopher Nolan')or(i=='James Wan'))
else colors.append('gray')for i in sorted_directors.tail(20).index]

fig , ax = plt.subplots(figsize=(12,8))
ax.barh(sorted_directors.tail(20).index,sorted_directors.
    ↳tail(20)['worldwide_gross'] ,color=colors)
ax.set_title('Highest-grossing film directors',fontsize=20)
ax.set_xlabel('Gross',fontsize=16)
ax.set_ylabel('Directors',fontsize=16)
```

```
[48]: Text(0, 0.5, 'Directors')
```



0.4 Conclusion :

As seems in first visualization BV, Fox, WB, Uni. made most revenue movies at all time. And at

At the section 3.2 shows us most made types of movies as Drama,Comedy and Action as general. I.

At the section 3.3 we can see there is positive relationship between budget and revenue. Also :

Lastly it seems Russo brothers(Avengers: Endgame), Christopher Nolan(The Dark Knight Rises) and

We can recommend to new studio to make movie with one of these directors, on action,adventure,

[]: