

Phase 2 Final Project Submission

- Student Name: Tai Ngoc Bui
- Student Pace: Flex
- Scheduled Project Review: April 1st, 2024
- Instructor Name: Morgan Jones

1. Business Understanding

This data exploratory project focuses on analyzing characteristics of successful movies at the box office to support the company's strategic investment in the movie industry. The goal is to gain valuable insights contributing to movie production success and provide meaningful recommendations to major stakeholders of the investments.

2. Data Understanding

The dataset used in this analysis extracted from various different movie sources including IMDb, the Numbers, and TMDB. As these datasets were collected from different sources, they have different formats. While TMDB and the Numbers datasets are compressed as CVS files, the IMDb, the largest dataset among the three, is stored within a SQLite database. These datasets not only contain movies' information on their cast, production crew, budget, and revenues, etc. but also the public opinions regarding the movies' success such as ratings and votes.

3. Data Preparation

```
In [1]: 1 #import appropriate libraries for analysis and visualization
        2 import pandas as pd
        3 import sqlite3
        4 import datetime
        5 import matplotlib.pyplot as plt
        6 import seaborn as sns
        7 import numpy as np
        8 import re
        9
        10 %matplotlib inline
```

a. Load Data to Dataframes

```
In [2]: 1 conn = sqlite3.connect('data\im.db')
2 df = pd.read_sql('''SELECT title, start_year, runtime_minutes, averagerati
3             primary_name, category FROM
4             (SELECT * FROM movie_basics LEFT JOIN
5             (SELECT * FROM movie_akas LEFT JOIN movie_rat
6             USING(movie_id)) LEFT JOIN
7             (SELECT movie_id, primary_name, category FROM
8             persons USING(person_id) WHERE death_year IS
9             ''', conn)
10
```

```
In [3]: 1 #Load all csv and tsc files into dataframe
2 df1 = pd.read_csv('Data/tmdb.movies.csv', parse_dates = ['release_date'],
3 df2 = pd.read_csv('Data/tn.movie_budgets.csv', parse_dates = ['release_dat
4
```

b. Cleaning Dataframes

IMDb Dataset

```
In [4]: 1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2928123 entries, 0 to 2928122
Data columns (total 7 columns):
#   Column          Dtype
---  -
0   title           object
1   start_year      int64
2   runtime_minutes float64
3   averagerating   float64
4   numvotes        float64
5   primary_name    object
6   category        object
dtypes: float64(3), int64(1), object(3)
memory usage: 156.4+ MB
```

```
In [5]: 1 df.drop_duplicates(inplace = True)
2 df.isna().sum()
```

```
Out[5]: title           128692
start_year             0
runtime_minutes       243179
averagerating         494112
numvotes              494112
primary_name          1697
category              1313
dtype: int64
```

```
In [6]: 1 # As I will use this dataset to analyze directors' success, it is best to
2 # have Nan director values
3 df.dropna(subset=['title','primary_name'], inplace = True)
4 # fill column runtime_minutes' Nan values with its mode,
5 # meanwhile, averagerating and numvotes' Nan values will be filled with 0
6 runtime = df['runtime_minutes'].mode().values[0]
7 df.fillna(value = {'runtime_minutes':runtime, 'averagerating':0, 'numvotes'
```

```
In [7]: 1 df.isna().sum()
```

```
Out[7]: title                0
start_year                0
runtime_minutes          0
averagerating            0
numvotes                 0
primary_name             0
category                 0
dtype: int64
```

TMDB Dataset

```
In [8]: 1 #Let's have a overview of the TMDB dataset
2 df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            26517 non-null  int64
1   genre_ids             26517 non-null  object
2   id                    26517 non-null  int64
3   original_language    26517 non-null  object
4   original_title       26517 non-null  object
5   popularity           26517 non-null  float64
6   release_date         26517 non-null  datetime64[ns]
7   title                 26517 non-null  object
8   vote_average         26517 non-null  float64
9   vote_count           26517 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(3), object(4)
memory usage: 2.0+ MB
```

```
In [9]: 1 #Check the number of Nan values in TMDB dataset
        2 df1.isna().sum()
```

```
Out[9]: Unnamed: 0      0
        genre_ids      0
        id             0
        original_language  0
        original_title  0
        popularity     0
        release_date    0
        title           0
        vote_average    0
        vote_count      0
        dtype: int64
```

I realize the genres in TMDB dataset are all recorded as id code. I found the actual name for each of genre id from TMDB website and store them in dictionary d. I will split the data stored in column genre_ids into individual columns. If a movie is assigned as that particular genre, the cell in that genre column will show 1, else it will show 0.

```

In [10]: 1 # Split the genre_ids column into multiple columns and change the genre_id
2         d = {28: 'Action',
3              12: 'Adventure',
4              16: 'Animation',
5              35: 'Comedy',
6              80: 'Crime',
7              99: 'Documentary',
8              18: 'Drama',
9              10751: 'Family',
10             14: 'Fantasy',
11             36: 'History',
12             27: 'Horror',
13             10402: 'Music',
14             9648: 'Mystery',
15             10749: 'Romance',
16             878: 'Science Fiction',
17             10770: 'TV Movie',
18             53: 'Thriller',
19             10752: 'War',
20             37: 'Western'}
21 # Split the genre_ids column which is a string type into multiple columns
22 # if movie contains a certain genre, the movie's genre column will show 1,
23 genres = np.unique(np.concatenate(np.array(df1['genre_ids'])))
24 genres = [int(x) for x in genres]
25 df1[genres] = 0
26 for i, row in df1.iterrows():
27     for g in row['genre_ids']:
28         df1.at[i,g] = 1
29
30 #Change the genre_ids into genres' identical names
31 df1.rename(columns = d, inplace = True)
32 # the genres array now contains the genres identical names, not ids
33 genres = list(d.values())
34
35 # create a new column containing the number of genres associated with the
36 df1['genre_len'] = [len(x) for x in df1['genre_ids']]

```

The Numbers Dataset

```

In [11]: 1 df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                     5782 non-null   int64
1   release_date           5782 non-null   datetime64[ns]
2   movie                  5782 non-null   object
3   production_budget      5782 non-null   object
4   domestic_gross         5782 non-null   object
5   worldwide_gross        5782 non-null   object
dtypes: datetime64[ns](1), int64(1), object(4)
memory usage: 271.2+ KB

```

```
In [12]: 1 #Check the number of Nan values in The Numbers dataset
        2 df2.isna().sum()
```

```
Out[12]: id                0
         release_date      0
         movie             0
         production_budget  0
         domestic_gross    0
         worldwide_gross   0
         dtype: int64
```

I realize all data stored in columns `production_budget`, `domestic_gross` and `worldwide_gross` are under string type. To convert these columns to float type, I need to strip the special character \$ out of the string before converting. I also create a new column ROI which stands for return on investment. ROI is calculated as the difference between `worldwide_gross` and `production_budget` divided by `production_budget`.

```
In [13]: 1 #convert string types columns into float types
        2 df2['production_budget'] = df2['production_budget'].replace('\$', '', r
        3 df2['domestic_gross'] = df2['domestic_gross'].replace('\$', '', regex=T
        4 df2['worldwide_gross'] = df2['worldwide_gross'].replace('\$', '', regex
        5
        6 #Create ROI (Return on Investment) column: ROI = (earnings - budget)/budge
        7 df2['ROI'] = (df2['worldwide_gross']-df2['production_budget'])/df2['produc
        8 df2['year'] = pd.DatetimeIndex(df2['release_date']).year
```

Joining TMDb and The Numbers dataset

While TMDb dataset has over 25000 data points of different movies with great features such as genres, popularity, vote average; it is missing gross revenue - one of the crucial features determining the success of a movie. Such gross revenue information is only carried by the Numbers' dataset. As a result, I joined the two dataset together using their title columns to better understand how different movies' features contributing to their success.

```
In [14]: 1 #join dataset from TMDB and The Numbers using their titles. As The Numbers
2 # have a greater number of data points, I use right join to maximize the d
3 df12 = pd.merge(df1, df2, left_on='original_title', right_on='movie', how
4 df12.dropna(inplace = True)
5 # Keep only important columns
6 keep = ['title', 'vote_average', 'vote_count', 'release_date_x', 'popularity'
7 df12 = df12[keep]
8
9 #reset index so that the index shows continuous values rather than the joi
10 df12.reset_index(inplace=True)
11 df12.drop(['index'], axis = 1, inplace = True)
12 df12.drop_duplicates(inplace = True)
13
14 #create new column "released_month" which is the month part of "released_d
15 df12['release_month'] = pd.DatetimeIndex(df12['release_date_x']).month
```

Joining IMDb and The Numbers dataset

I plan to analyze what are the best directors in terms of both ROI and overall ratings later. While IMDb's dataset contains detail information regarding movies and directors, it does not have any gross revenue or production budget information. Such data can only be found under The Numbers' dataset. Thus, I decide to merge IMDb's and The Numbers' dataset together.

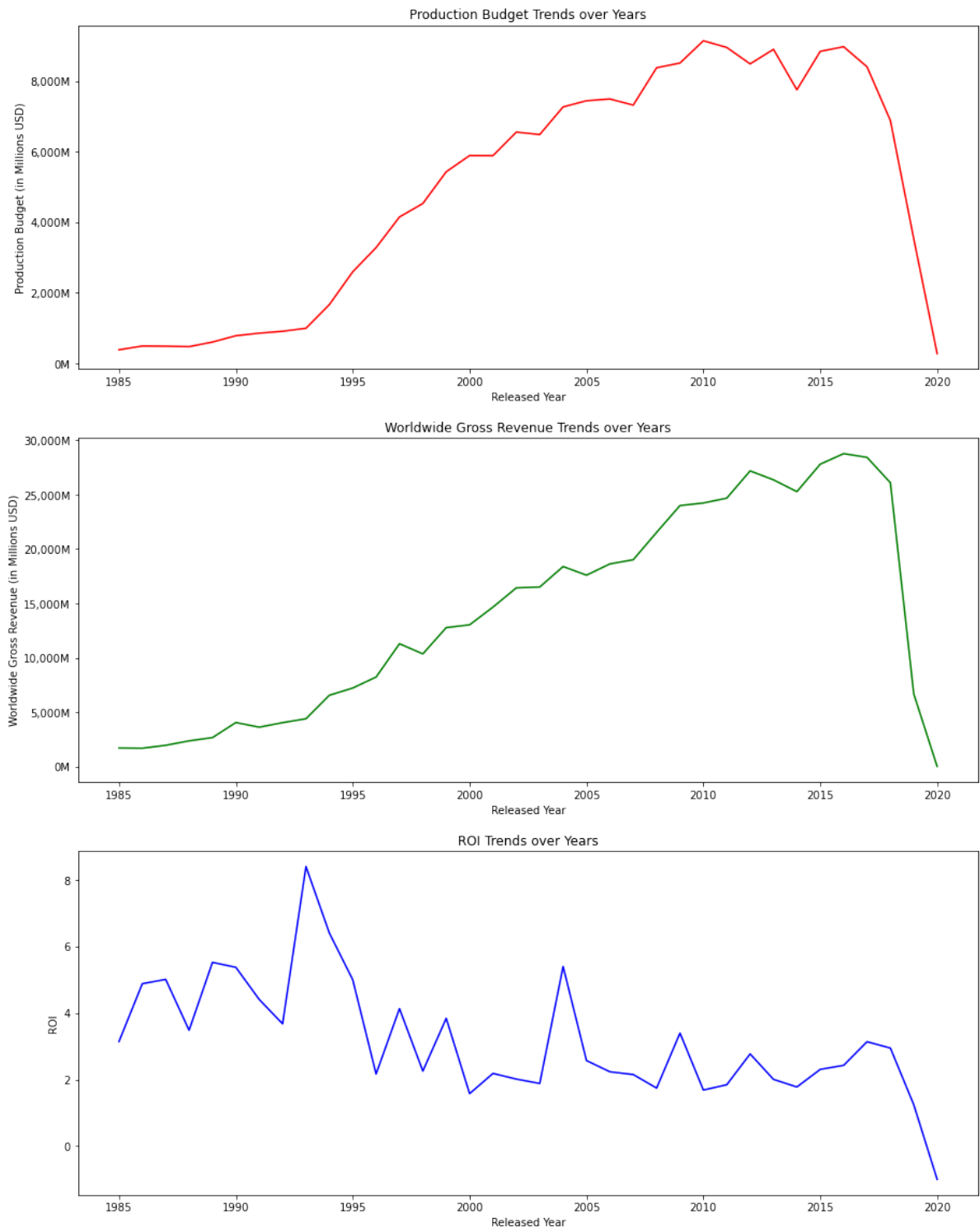
```
In [15]: 1 # Here I am only interested in movies that have directors' information
2 df_dir = df[df['category'] == 'director']
3
4 # Merging data with a right join
5 df_dir = pd.merge(df_dir, df2, left_on=['title', 'start_year'], right_on=[
6 df_dir = df_dir[['title', 'averagerating', 'numvotes', 'primary_name', 'catego
7 'production_budget', 'domestic_gross', 'worldwide_gross', 'R
8 # Create a new column ROI in the joined dataframe
9 df_dir['ROI'] = (df_dir['worldwide_gross'] - df_dir['production_budget'])
10
11 df_dir.dropna(inplace = True)
```

Exploratory Data Analysis

Overview Trends

```
In [16]: 1 df2['year'] = pd.DatetimeIndex(df2['release_date']).year
2 df_trends = df2[df2['year'] >= 1985]
3 d_year = df_trends.groupby('year').aggregate({'production_budget':'sum',
4                                               'worldwide_gross':'sum','ROI':'mean'})
5 fig, ((ax1), (ax2), (ax3)) = plt.subplots(figsize = (15,20), nrows = 3)
6 ax1.plot(d_year.index, d_year.production_budget, color = 'red')
7 ax1.set_xlabel('Released Year')
8 ax1.set_ylabel('Production Budget (in Millions USD)')
9 ax1.set_title('Production Budget Trends over Years')
10
11 ax2.plot(d_year.index, d_year.worldwide_gross, color = 'green')
12 ax2.set_xlabel('Released Year')
13 ax2.set_ylabel('Worldwide Gross Revenue (in Millions USD)')
14 ax2.set_title('Worldwide Gross Revenue Trends over Years')
15
16 ax3.plot(d_year.index, d_year.ROI, color = 'blue')
17 ax3.set_xlabel('Released Year')
18 ax3.set_ylabel('ROI')
19 ax3.set_title('ROI Trends over Years')
20
21 # suppress scientific notation by setting yticklabels
22 for ax in [ax1, ax2]:
23     ylabels = ['{:, .0f}'.format(y) + 'M' for y in ax.get_yticks()/1000000]
24     ax.set_yticklabels(ylabels)
25 plt.show();
```

<ipython-input-16-0a7f579d11db>:24: UserWarning: FixedFormatter should only be used together with FixedLocator
ax.set_yticklabels(ylabels)

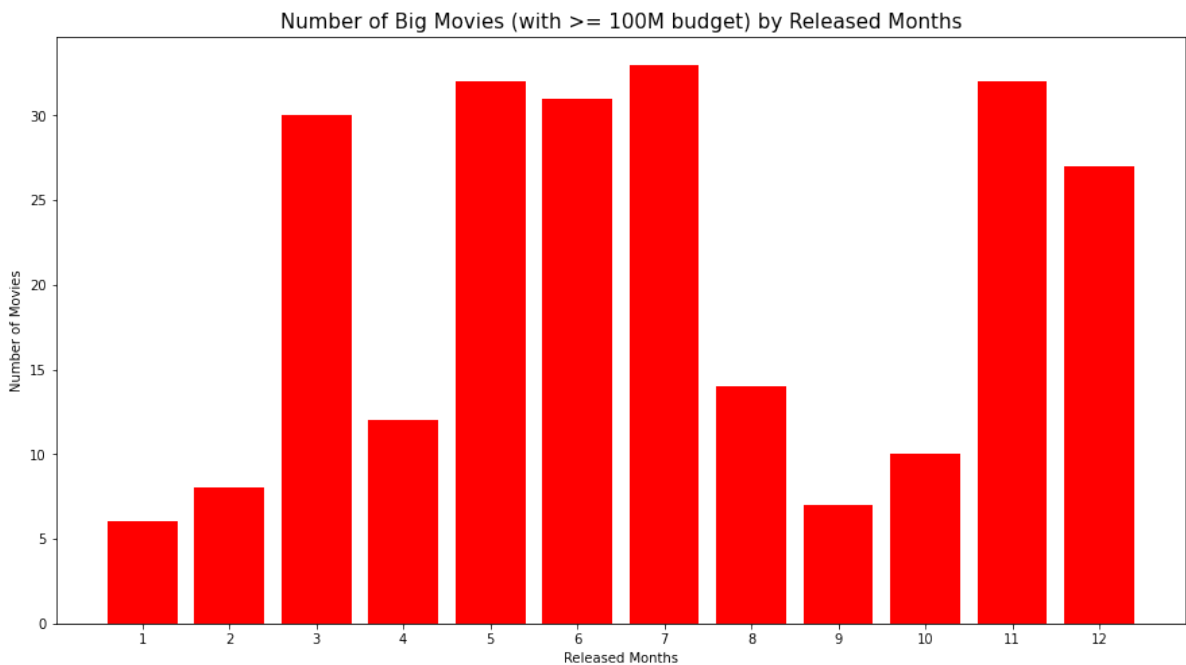


The above charts show that the investment in the movie industry has consistently increased from 1985 - 2018. This consistent investment trend in movie industry also leads to growing worldwide gross revenue for movie production. However, the Covid-19 incident in late 2019 - early 2020 had caused a deep dive in both movie budget production and worldwide gross revenue. An incentive for investors to pour their investment in the movie industry is due to its high ROI. Though ROI is somewhat on a down trend from 1985 - 2020, the yearly average return had been consistently stayed above 200% from 1985 - 2018.

First Recommendation

a. In which month big movies (production budget over 100M) released?

```
In [17]: 1 # group the joined dataset by release month and filter only those moview w  
2 # more than 100M  
3 big_movies = df12[df12['production_budget'] >= 100000000].groupby('release  
4  
5 #Graphing the data  
6 fig, ax = plt.subplots(figsize = (15,8))  
7 plt.bar(x = big_movies.index, height = big_movies['production_budget'], co  
8 plt.xticks(range(1,13))  
9 plt.xlabel('Released Months')  
10 plt.ylabel('Number of Movies')  
11 plt.title('Number of Big Movies (with >= 100M budget) by Released Months',  
12 plt.show();
```



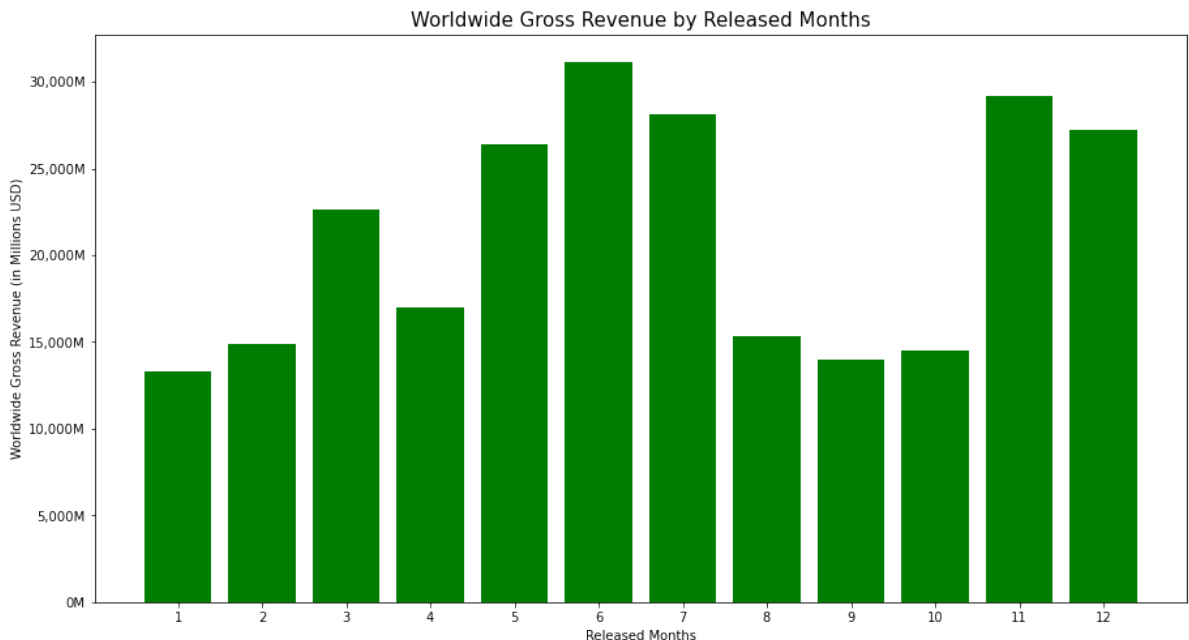
b. In which month do movies hit best gross revenue?

```

In [18]: 1 # Group the joined dataset by released_month but this time, we want to view
2 # distributed by released month
3 release_month = df12.groupby('release_month').sum()
4
5 # Graphing the data
6 fig, ax = plt.subplots(figsize = (15,8))
7 ax.bar(x = release_month.index, height = release_month['worldwide_gross'],
8 ax.set_xticks(range(1,13))
9 ax.set_xlabel('Released Months')
10 ax.set_ylabel('Worldwide Gross Revenue (in Millions USD)')
11 ax.set_title('Worldwide Gross Revenue by Released Months', fontsize = 15)
12 ylabels = ['{:, .0f}'.format(y) + 'M' for y in ax.get_yticks()/1000000]
13 ax.set_yticklabels(ylabels)
14 plt.show();

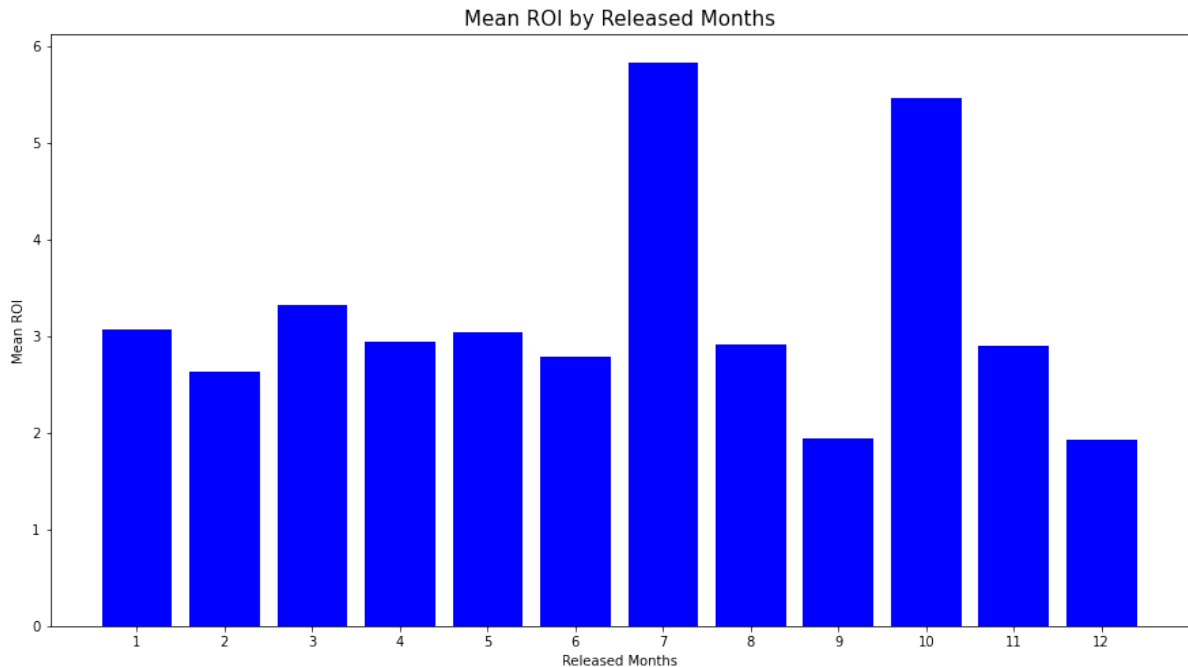
```

<ipython-input-18-1a363bddecbb>:13: UserWarning: FixedFormatter should only be used together with FixedLocator
 ax.set_yticklabels(ylabels)



c. In which month do movies achieve best ROI?

```
In [19]: 1 # Group the joined dataset by released_month, but this time, we want to co
2 # of the movies' ROI rate
3 roi_month = df12.groupby('release_month').mean()
4
5 # Graphing the data
6 fig, ax = plt.subplots(figsize = (15,8))
7 plt.bar(x = roi_month.index, height = roi_month['ROI'], color = 'blue')
8 plt.xticks(range(1,13))
9 plt.xlabel('Released Months')
10 plt.ylabel('Mean ROI')
11 plt.title('Mean ROI by Released Months', fontsize = 15)
12 plt.show();
```



With our 12 months analysis, we can realize July and October are the 2 best months in which movies can achieve great ROI for studio producers. On the other hand, we also want to avoid compete directly with the major movies in the box office (assumed with over 100M budget production). The first and second chart show that most big movies are often released during summer (Jun - August) and the last two months of the year (November and December). As a result, my first recommendation to the stakeholders is to prioritize releasing movie in October of the year. Not only we can avoid compete directly with major movies, but also have a higher chance of hitting higher ROI rates.

Second Recommendation

a. Which movie genres have the largest production budget?

```

In [20]: 1 #create a new column rev_per_genre
2 df12['rev_per_genre'] = df12['worldwide_gross'] / df12['genre_len']
3
4 # create a new column budget_per_genre
5 df12['budget_per_genre'] = df12['production_budget'] / df12['genre_len']
6
7 # distribute budget per genre and revenue per genre into each of genre col
8 gross_rev_genre = df12[genres].multiply(df12['rev_per_genre'], axis = 0)
9 budget_genre = df12[genres].multiply(df12['budget_per_genre'], axis = 0)
10
11 #get the total revenue and budget production associated with each genres
12 sum_rev_genres = gross_rev_genre.sum(axis = 0)
13 sum_budget_genres = budget_genre.sum(axis = 0)
14 # obtain return on investment associated with each genres
15 roi_genres = round(sum_rev_genres/sum_budget_genres - 1,2)

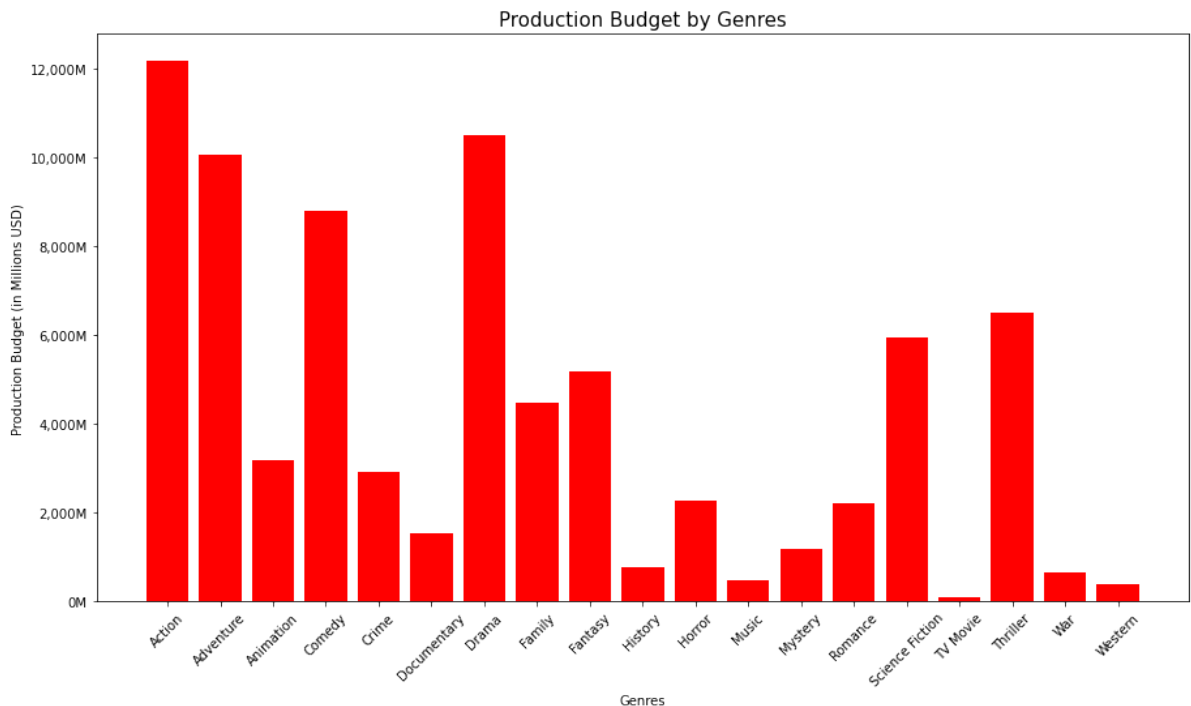
```

```

In [21]: 1 # Chart production budget by genres
2 fig, ax = plt.subplots(figsize = (15,8))
3 ax.bar(x = sum_budget_genres.index, height = sum_budget_genres.values, col
4 ax.tick_params(axis='x', labelrotation=45)
5 ax.set_xlabel('Genres')
6 ax.set_ylabel('Production Budget (in Millions USD)')
7 ax.set_title('Production Budget by Genres', fontsize = 15)
8 ylabels = ['{:.0f}'.format(y) + 'M' for y in ax.get_yticks()/1000000]
9 ax.set_yticklabels(ylabels)
10 plt.show();

```

<ipython-input-21-b66f968ec4fd>:9: UserWarning: FixedFormatter should only be used together with FixedLocator
 ax.set_yticklabels(ylabels)

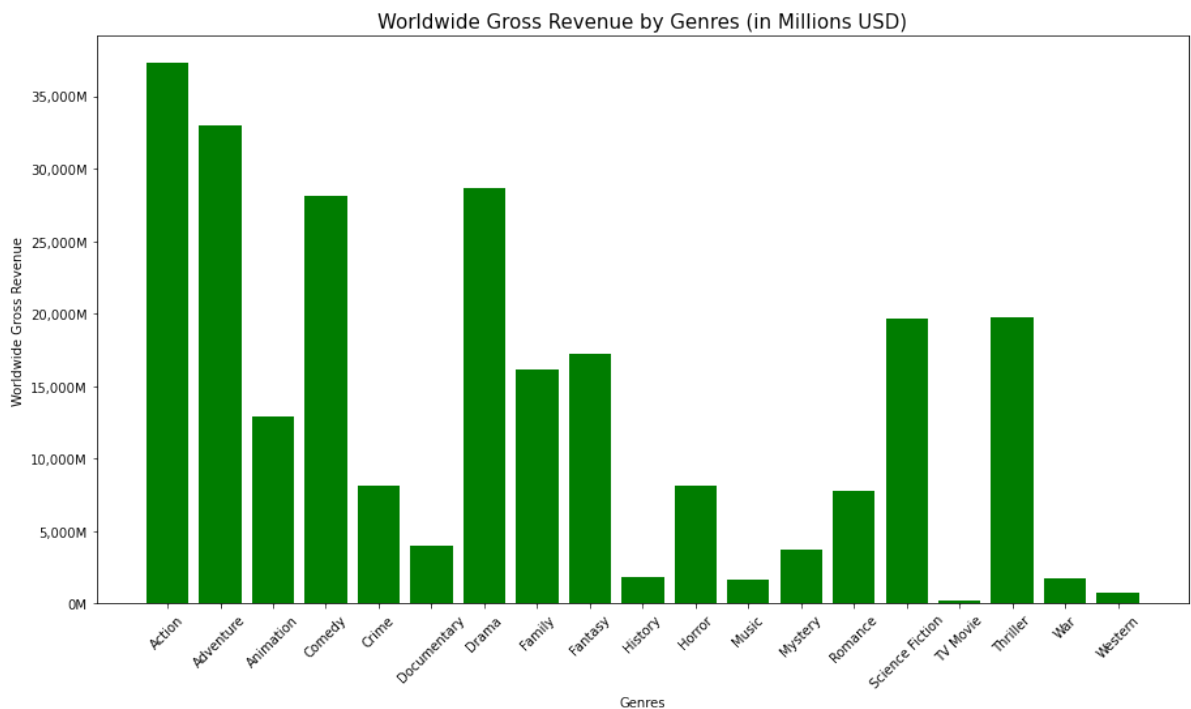


b. Which movie genres have the highest gross revenue?

```
In [22]: 1 # Chart worldwide gross revenue by genres
2 fig, ax = plt.subplots(figsize = (15,8))
3 ax.bar(x = sum_rev_genres.index, height = sum_rev_genres.values, color = 'green')
4 ax.tick_params(axis='x', labelrotation=45)
5 ax.set_xlabel('Genres')
6 ax.set_ylabel('Worldwide Gross Revenue')
7 ax.set_title('Worldwide Gross Revenue by Genres (in Millions USD)', fontsize=12)
8 ylabels = ['{:.0f}'.format(y) + 'M' for y in ax.get_yticks()/1000000]
9 ax.set_yticklabels(ylabels)
10 plt.show();
```

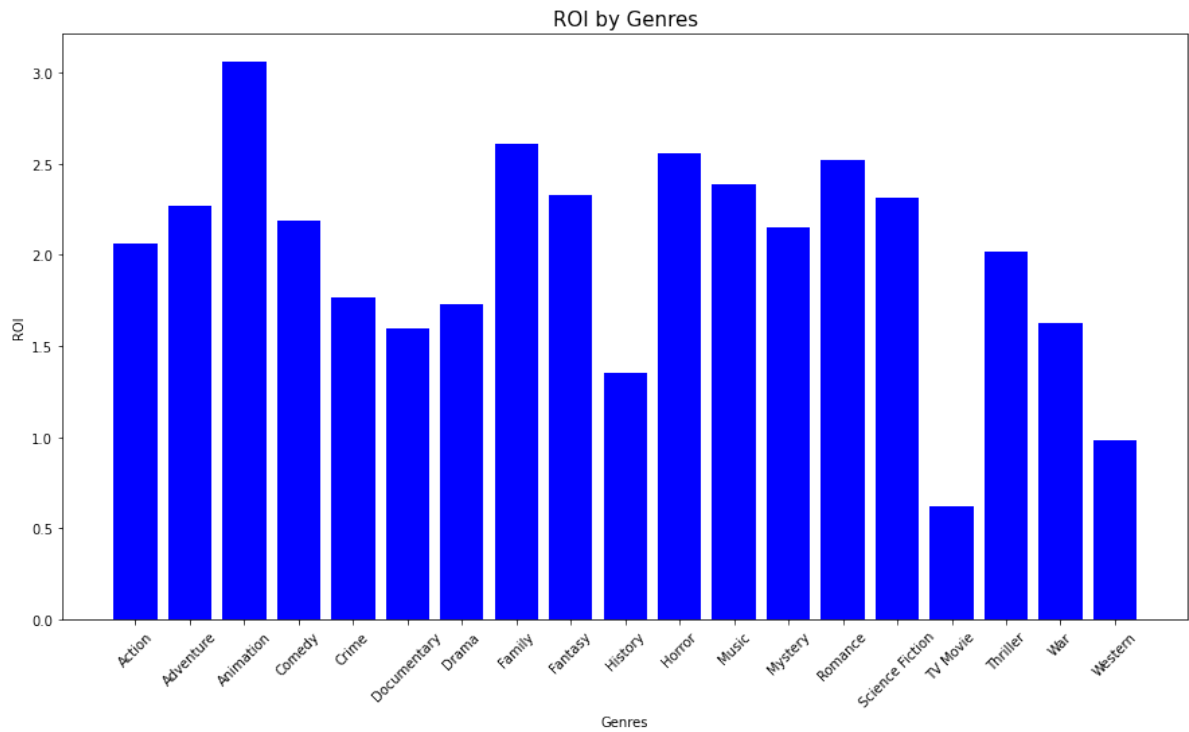
<ipython-input-22-011072ee5840>:9: UserWarning: FixedFormatter should only be used together with FixedLocator

```
ax.set_yticklabels(ylabels)
```



c. Which movie genres have the highest return on investment (ROI)?

```
In [23]: 1 # Chart ROI by genres
2 fig, ax = plt.subplots(figsize = (15,8))
3 ax.bar(x = roi_genres.index, height = roi_genres.values, color = 'blue')
4 ax.tick_params(axis='x', labelrotation=45)
5 ax.set_xlabel('Genres')
6 ax.set_ylabel('ROI')
7 ax.set_title('ROI by Genres', fontsize = 15)
8 plt.show();
```



d. Which movie genres have the highest ROI and lowest ratio between budget and revenue?

```

In [27]: 1 #Suppressing scientific notations
2 pd.options.display.float_format = '{:.0f}'.format
3
4 #combine the aggregate genres together and turn budget, revenue and ROI in
5 g = pd.concat([sum_budget_genres, sum_rev_genres, roi_genres], axis = 1)
6 g.columns = ['budget', 'revenue', 'ROI']
7 #calculate to see what percentage of budget needed to achieve gross revenue
8 g['budget/rev'] = g['budget']/g['revenue']*100
9 # sort values with ROI and revenue as descending while budget as ascending
10 g.sort_values(by = ['ROI', 'budget/rev'], ascending = [False, True])[:10]

```

```

Out[27]:

```

	budget	revenue	ROI	budget/rev
Animation	3173761905	12887223169	3	25
Family	4478295238	16153244216	3	28
Horror	2270333873	8090762134	3	28
Romance	2208368238	7783494817	3	28
Music	472733333	1602741319	2	29
Fantasy	5177499405	17252211819	2	30
Science Fiction	5929398750	19613437008	2	30
Adventure	10062379821	32951920652	2	31
Comedy	8805747744	28091860428	2	31
Mystery	1180781121	3724501544	2	32

My second recommendation is to focus on producing animation, horror, romance, and family movies. The reason I recommend these genres is due to their general low production cost relative to gross revenue and high ROI rate. As a result, the risk of loss due to high production cost in movie industry will be minimized if stakeholders' studio only invest in the four genres mentioned above, particularly animation.

Third Recommendation

Which directors have the highest ROI and average ratings? (great with both values and movie qualities)


```

In [25]: 1 # Group the movies' stats by directors
2 directors_stats = df_dir.groupby(['primary_name']).aggregate({'production_
3                                     'worldwide_g
4                                     'averagerati
5                                     'title': 'coun
6 # Only include directors with at least 3 movies (average in this database)
7 # purpose is to exclude one-hit directors and look for consistent performe
8 directors_stats = directors_stats[directors_stats.title >= 3]
9
10 # # Only include directors with at least 6 average rating (quality movies)
11 directors_stats = directors_stats[directors_stats.averagerating >= 6]
12
13 # # Rank directors by their ROI, average rating, and number of votes recei
14 # # only keep top 10
15 df_d = directors_stats.sort_values(by = ['ROI', 'averagerating'], ascending
16 df_d

```

```

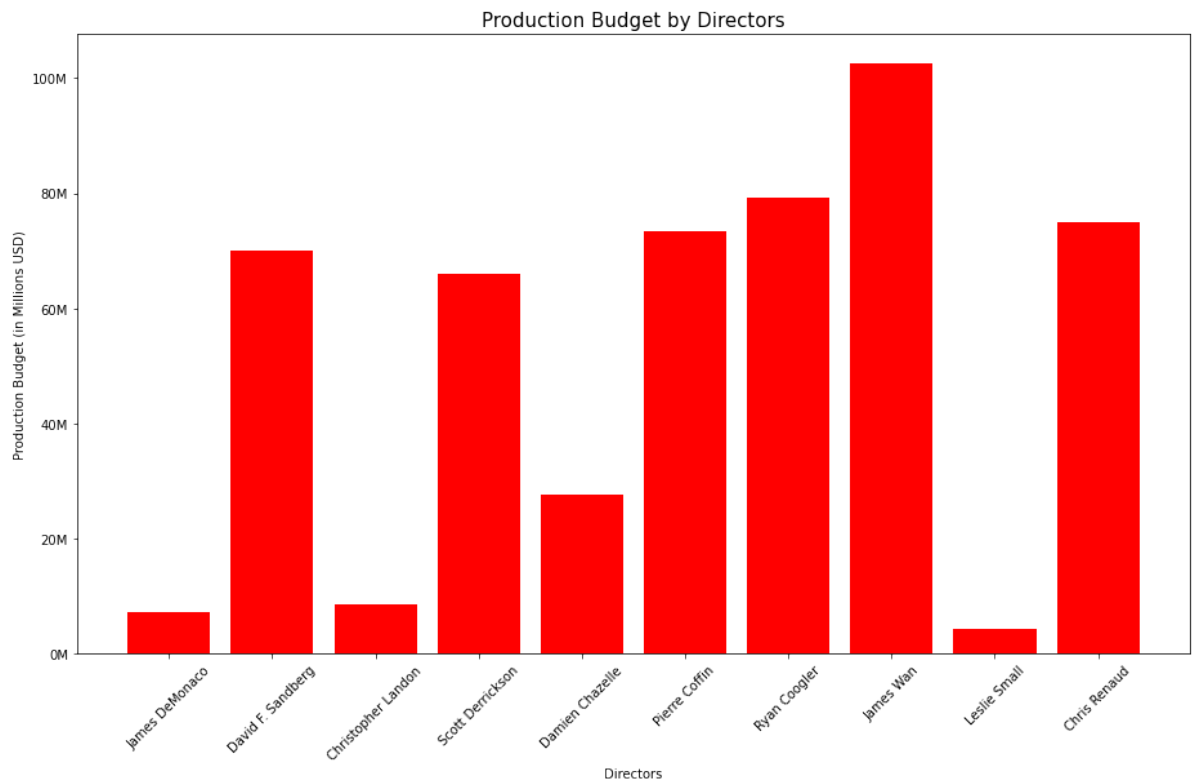
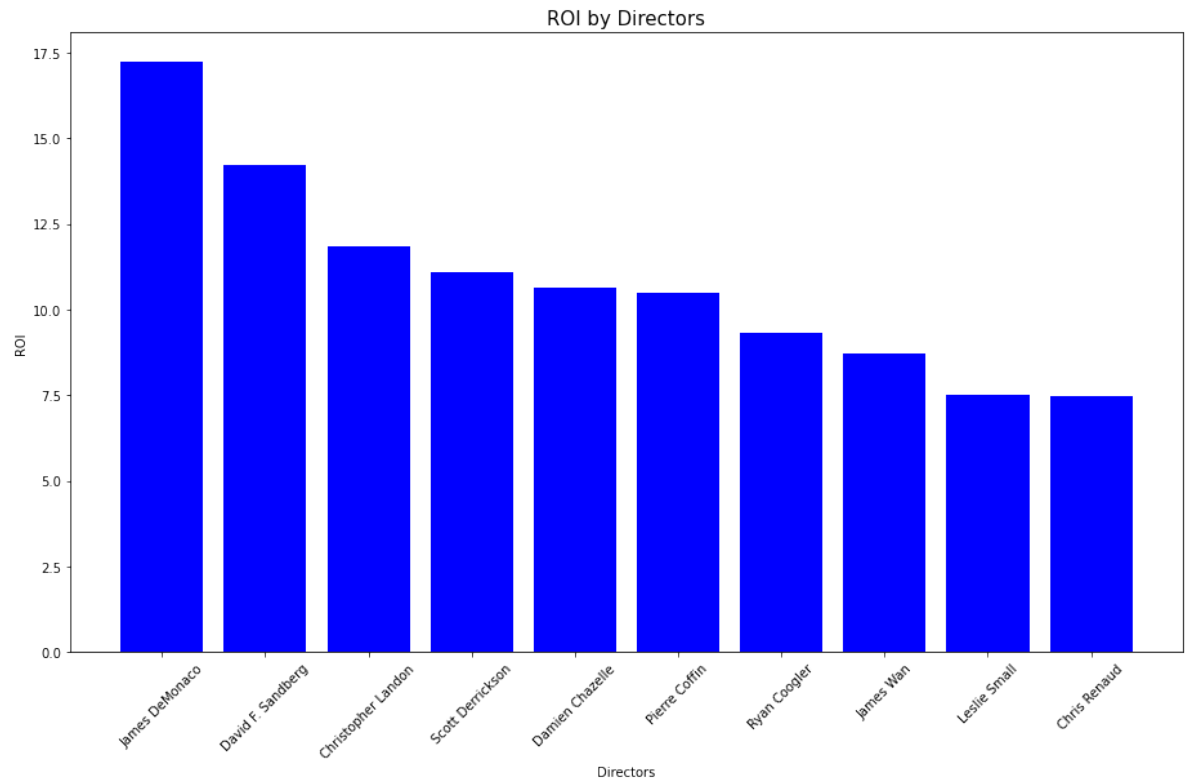
Out[25]:

```

	production_budget	worldwide_gross	ROI	averagerating	numvotes	title
primary_name						
James DeMonaco	7.333333e+06	1.071054e+08	17.222144	6.066667	390006.0	3
David F. Sandberg	7.000000e+07	4.850382e+08	14.201799	6.900000	409154.0	4
Christopher Landon	8.500000e+06	7.391237e+07	11.837767	6.025000	192220.0	4
Scott Derrickson	6.600000e+07	2.840297e+08	11.091304	6.833333	776648.0	3
Damien Chazelle	2.776667e+07	1.901747e+08	10.626585	7.966667	1172995.0	3
Pierre Coffin	7.333333e+07	8.511364e+08	10.501490	7.133333	901321.0	3
Ryan Coogler	7.930000e+07	5.131252e+08	9.310637	7.466667	809006.0	3
James Wan	1.025000e+08	8.237199e+08	8.710732	7.300000	1190178.0	4
Leslie Small	4.416667e+06	2.121024e+07	7.524418	6.700000	21095.0	3
Chris Renaud	7.500000e+07	6.296959e+08	7.487083	7.050000	974205.0	4

```
In [26]: 1 # Chart production budget
2 fig, ((ax1), (ax2)) = plt.subplots(figsize = (15,20), nrows = 2)
3
4 ax1.bar(x = df_d.index, height = df_d.ROI, color = 'blue')
5 ax1.tick_params(axis='x', labelrotation=45)
6 ax1.set_ylabel('ROI')
7 ax1.set_xlabel('Directors')
8 ax1.set_title('ROI by Directors', fontsize = 15)
9
10 ax2.bar(x = df_d.index, height = df_d.production_budget, color = 'red')
11 ax2.tick_params(axis='x', labelrotation=45)
12 ax2.set_ylabel('Production Budget (in Millions USD)')
13 ax2.set_xlabel('Directors')
14 ax2.set_title('Production Budget by Directors', fontsize = 15)
15 ylabels = ['{:.0f}'.format(y) + 'M' for y in ax2.get_yticks()/1000000]
16 ax2.set_yticklabels(ylabels)
17
18 # set the spacing between subplots
19 plt.subplots_adjust(left=0.1,
20                     bottom=0.1,
21                     right=0.9,
22                     top=0.9,
23                     wspace=0.4,
24                     hspace=0.4)
25 plt.show();
```

```
<ipython-input-26-9a5514519ca3>:16: UserWarning: FixedFormatter should only be used together with FixedLocator
ax2.set_yticklabels(ylabels)
```



My third recommendation to stakeholder is to select the 3 directors James DeMonaco, Christopher Landon, and Damien Chazelle for their studio investment. According to my analysis, these 3 directors not only achieve consistent good ROI rates with their movies but also awarded

Summary

Based on our exploratory analysis, there are three recommendations I have for our stakeholders include:

- Prioritize releasing movie in October of the year. Not only we can avoid compete directly with major movies, but also have a higher chance of hitting higher ROI rates.
- Prioritize producing animation, horror, romance, and family movies due to their general low production cost relative to gross revenue and high ROI rate.
- Prioritize inviting James DeMonaco, Christopher Landon, and Damien Chazelle to be the directors of the company's movies. They not only have great experience in the movie industry but also good track record of turning small budget movies into great returns.