

Final Project Submission

Please fill out:

- Student name: Andrew Levinton
- Student pace: part time
- Scheduled project review date/time: 11/28/2022
- Instructor name: Joe Comeaux
- Blog post URL: <https://github.com/andrewkoji/Phase-1-Project---Movie-Analysis.git>

Phase 1 Project

Business Problem

This project aims at finding the best movies to produce for Microsoft.

Overview

In this analysis, the following will be analyzed and used as measurements of success:

- Gross Profit
- Gross Revenue
- Net Profit
- Overall rating
- Overall reviews

To observe these measurements and make conclusions, an exploratory data analysis will be done on the datasets imported below. The analysis will help to provide key insight on the most essential aspects that make a successful movie.

Importing libraries

```
In [40]: import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import seaborn as sns
```

```
In [41]: cd zippedData
```

```
[WinError 2] The system cannot find the file specified: 'zippedData'  
C:\Users\alevi\Documents\Flatiron\dsc-data-science-env-config\Course_Folder\Phase_1\Phas  
e_1_Project\Phase-1-Project---Movie-Analysis\zippedData
```

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



```
Volume in drive C is Windows  
Volume Serial Number is A24A-7E70
```

```
Directory of C:\Users\alevi\Documents\Flatiron\dsc-data-science-env-config\Course_Fold  
e\Phase_1\Phase_1_Project\Phase-1-Project---Movie-Analysis\zippedData
```

```

11/23/2022 03:56 PM <DIR> .
11/25/2022 02:46 PM <DIR> ..
11/22/2022 11:31 PM <DIR> .ipynb_checkpoints
11/25/2022 01:57 PM 33,800 Bar Chart Rev vs Rating.jpeg
11/25/2022 02:03 PM 50,148 Bar_chart_month_profit.jpeg
11/22/2022 11:28 PM 53,544 bom.movie_gross.csv.gz
11/25/2022 02:01 PM 78,914 production_budget_vs_net_profit.jpeg
11/23/2022 03:57 PM 68,866 Profit_by_Genre.jpeg
11/25/2022 01:57 PM 85,924 Revenue by Genre.jpeg
11/25/2022 01:58 PM 56,647 Revenue by Month.jpeg
11/22/2022 11:28 PM 498,202 rt.movie_info.tsv.gz
11/22/2022 11:28 PM 3,402,194 rt.reviews.tsv.gz
11/22/2022 11:28 PM 827,840 tmdb.movies.csv.gz
11/22/2022 11:28 PM 153,218 tn.movie_budgets.csv.gz
11 File(s) 5,309,297 bytes
3 Dir(s) 72,639,352,832 bytes free

```

The Data

```
In [43]: tmdb = pd.read_csv('tmdb.movies.csv.gz')
bom = pd.read_csv('bom.movie_gross.csv.gz')
rt_reviews = pd.read_csv('rt.reviews.tsv.gz', sep='\t', encoding=('ISO-8859-1'), low_memo
rt_info = pd.read_csv('rt.movie_info.tsv.gz', sep='\t')
budgets = pd.read_csv('tn.movie_budgets.csv.gz')
```

The libraries above are the datasets that will be cleaned and analyzed for the analysis.

Getting Rating Data From Rotten Tomatoes

Cleaning the Review Data

```
In [44]: #import review data
rt_reviews = pd.read_csv('rt.reviews.tsv.gz', sep='\t', encoding=('ISO-8859-1'), low_memo

#filter by top critics
rt_reviews = rt_reviews[rt_reviews['top_critic'] == 1]

#change all "Letter Grade" ratings to a %
rt_reviews['rating'] = rt_reviews['rating'].astype(str)
rt_reviews['rating'] = rt_reviews['rating'].apply(lambda x:\n    '100' if 'A+' in x\n    else '98' if 'A' in x\n    else '95' if 'A-' in x\n    else '88' if 'B+' in x\n    else '85' if 'B' in x\n    else '80' if 'B-' in x\n    else '78' if 'C+' in x\n    else '75' if 'C' in x\n    else '70' if 'C-' in x\n    else '68' if 'D+' in x\n    else '65' if 'D' in x\n    else '60' if 'D-' in x\n    else '50' if 'F' in x\n    else '45' if 'N' in x\n    else '40' if 'R' in x else x)

#change all "fractional" reviews to %
```

```

rt_reviews['rating'] = rt_reviews['rating'].str.split('/').apply(lambda x: (float(x[0])
    if (len(x) > 1 and float(x[1]) != 0) \
    else float(x[0])))
#some reviews were over 100% because te reviews had improper fractions(eg. 2.1/2). They
rt_reviews['rating'] = rt_reviews['rating'].apply(lambda x: 100 if x > 100 else x)
rt_reviews = rt_reviews.sort_values('rating', ascending=False)

```

Merging Tomato Data

```
In [45]: rt = pd.merge(rt_reviews,rt_info,on='id')
```

```
In [46]: print(len(rt_reviews))
print(len(rt_info))
print(len(rt))
```

13096
1560
13096

Getting Genre Data

```
In [47]: # Dropping nulls from ratings and Genre
rt = rt[rt['rating_x'].notnull()]
rt = rt[rt['box_office'].notnull()]
#Changing box office revenue so it can be read numerically
rt['box_office'] = rt['box_office'].str.replace(',', '').apply(lambda x: int(x))
#split the genre(eg. Action/Drama become -> [Action, Drama])
rt['genre'] = rt['genre'].str.split('|')
```

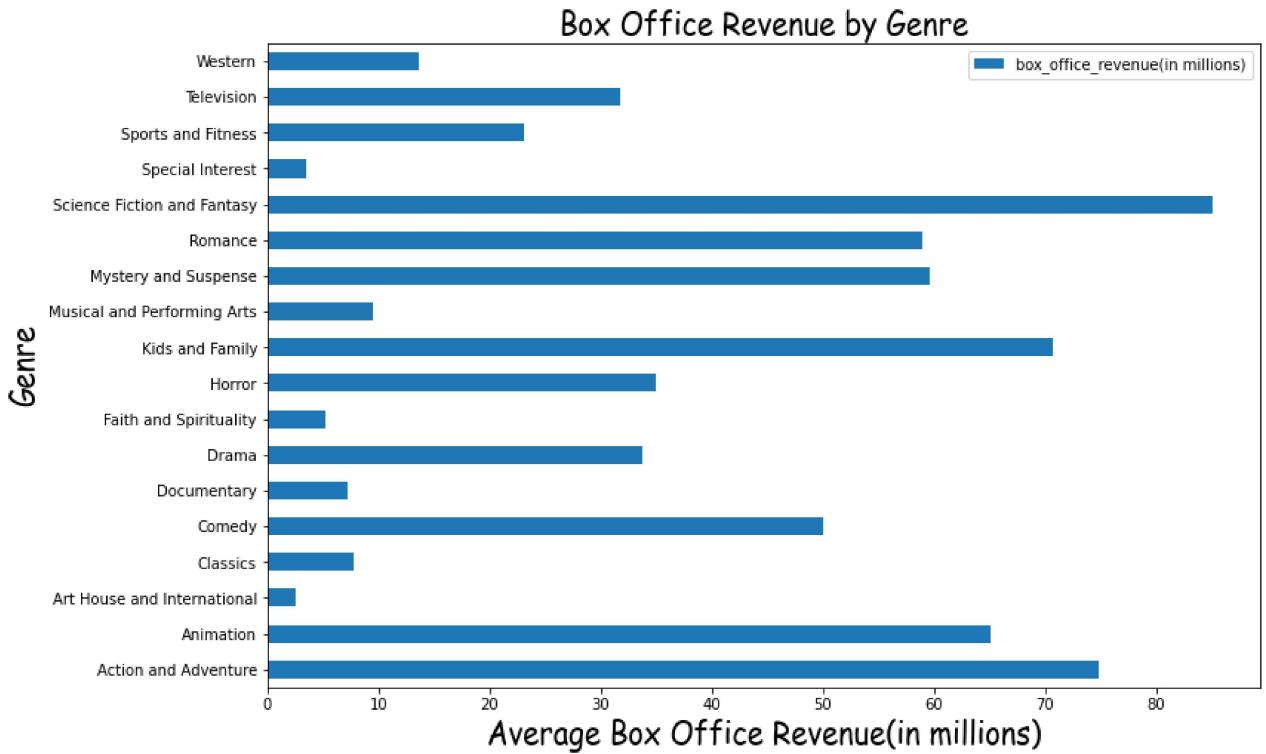
```
In [48]: #explode genre column so each value within the list has its own row. this will make it
genre = rt.explode('genre')
#rename columns
genre = genre.rename(columns={'box_office':'box_office_revenue'})
genre = genre.rename(columns={'rating_x':'rating'})
```

Box Office Revenue by Genre

```
In [49]: #format x-values
pd.set_option('display.float_format', lambda x: '%.3f' % x)
#define fonts that can be used for all the graphs
csfont = {'fontname':'Comic Sans MS'}
hfont = {'fontname':'Helvetica'}

#group by genre
genre_revenue = genre.groupby('genre')['box_office_revenue'].mean().reset_index()
#convert to millions
genre_revenue['box_office_revenue(in millions)'] = genre_revenue['box_office_revenue']

#create bar graph of box office revenue by genre
genre_revenue.plot(x='genre',y='box_office_revenue(in millions)',kind='barh',figsize=(1
plt.title('Box Office Revenue by Genre', size=20, **csfont)
plt.xlabel('Average Box Office Revenue(in millions)',size=20, **csfont)
plt.ylabel('Genre',size=20,**csfont)
#plt.show()
plt.savefig("Revenue by Genre.jpeg",bbox_inches='tight') #save as jpg
```



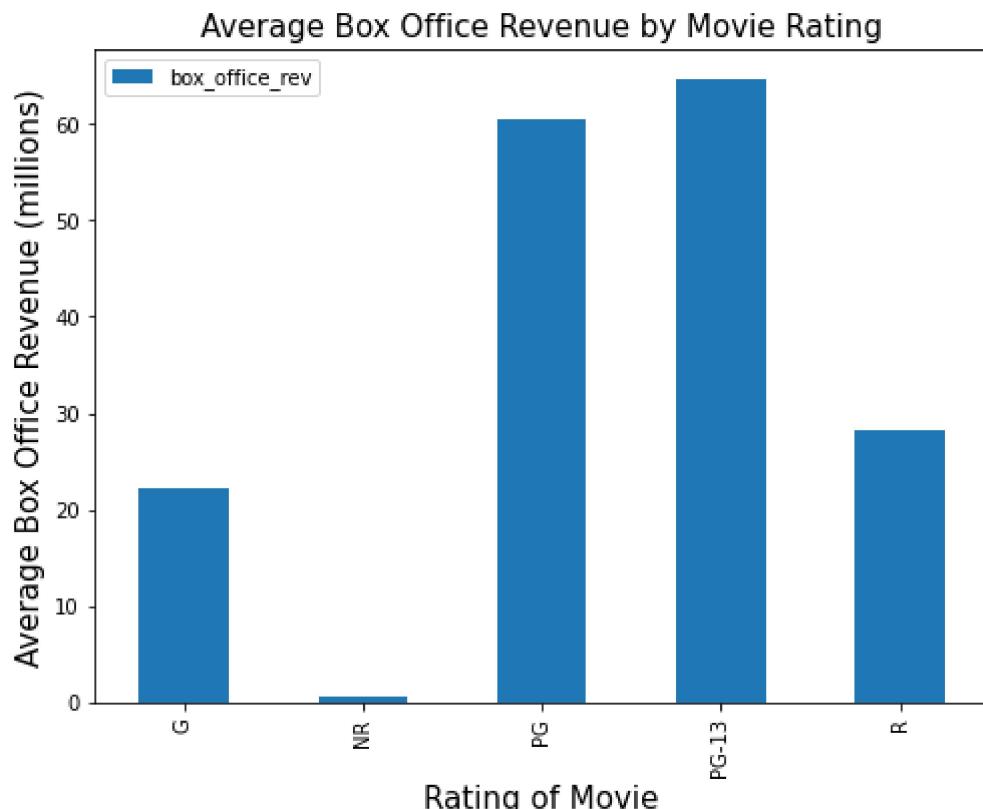
In the graph above, it is shown that Science Fiction and Fantasy along with Action and Adventure appear to yield the highest average box office revenue, with Animation, Mystery/Suspense and Romance being the next top 3 performers. As animation improves with innovative technology, it is clear that the more popular movies does lead to a higher turnout at the box office as they are the genres that apply the most to the masses.

Which Movie Rating Attracts the Most Revenue?

```
In [50]: #getting average revenue by rating
rating_movie = rt.groupby('rating_y')['box_office'].mean().reset_index()

#converting to millions
rating_movie['box_office_rev'] = rating_movie['box_office'] / 1000000

#bar graph of movie rating vs the box office revenue
rating_movie.plot(x='rating_y',y='box_office_rev',kind='bar',figsize=(8,6))
plt.ylabel('Average Box Office Revenue (millions)',size=15)
plt.xlabel('Rating of Movie',size=15)
plt.title('Average Box Office Revenue by Movie Rating',size=15)
plt.savefig('Bar Chart Rev vs Rating.jpeg')
```



What are the top 10 directors in terms of box office revenue?

```
In [51]: #using same method for genres to develop average revenue by director
directors = rt[['director','box_office']].drop_duplicates()

directors['director'] = directors['director'].str.split('|')
directors = directors.explode('director')
```

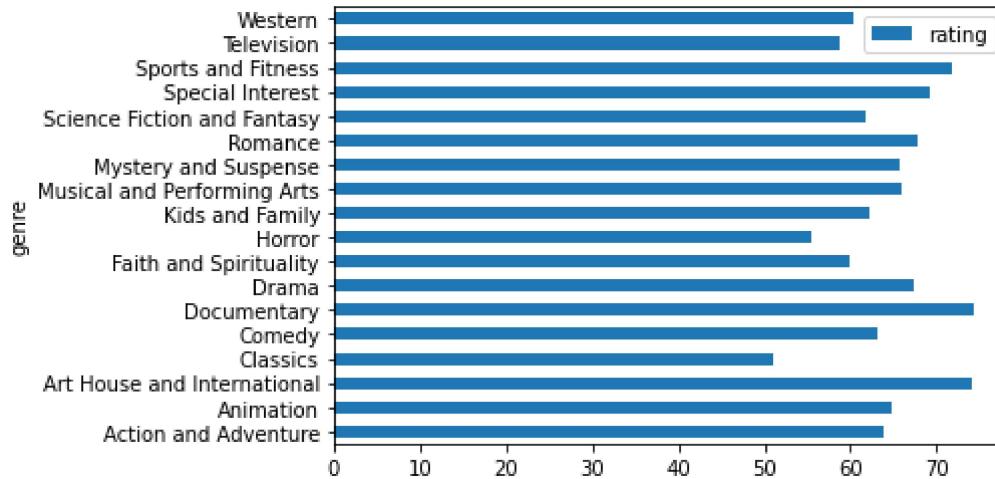
```
In [52]: #top 10 directors by box office revenue
directors.sort_values('box_office', ascending=False).head(10)
```

Out[52]:

	director	box_office
191	Mel Gibson	368000000
5441	Peter Jackson	303001229
2274	Sam Mendes	299300000
8843	Jay Roach	279167575
238	Chris Columbus	261835892
3978	Joel Zwick	241250669
0	Steven Spielberg	234141872
4075	Peter Berg	227946274
5025	Justin Lin	209805005
5676	Andy Tennant	177575142

Average Rating by Genre

```
In [53]: genre_rating = genre.groupby('genre')['rating'].mean().reset_index()
ax = genre_rating.plot(x='genre',y='rating',kind='barh')
plt.show()
```



As shown above, there is no conclusive genre that yields a significantly higher rating. My advice on this front would be to simply not consider reviews when picking a genre.

Box Office Revenue by month

```
In [54]: #stripping month number from dates
rt['release_month'] = rt['theater_date'].astype(str).apply(lambda x: x[0:3])
```

```
In [55]: #group by month
monthly = rt.groupby('release_month')['box_office'].mean().reset_index()
```

```
In [56]: monthly = monthly[0:12]
```

```
In [57]: #function to format month name
monthly['release_month'] = monthly['release_month'].apply(lambda x: 'January' if x=='Ja
else 'February' if x=='Feb'\
else 'March' if x=='Mar'\
else 'April' if x=='Apr'\
else 'May' if x=='May'\
else 'June' if x=='Jun'\
else 'July' if x=='Jul'\
else 'August' if x=='Aug'\
else 'September' if x=='Sep'\
else 'October' if x=='Oct'\
else 'November' if x=='Nov'\
else 'December' if x=='Dec'\
else x)
```

```
In [58]: #assigning number to each month to order them
monthly['month_no'] = monthly['release_month'].apply(lambda x: 1 if x=='January'\
else 2 if x=='February'\
else 3 if x=='March'\
else 4 if x=='April'\
else 5 if x=='May'\
else 6 if x=='June'\
else 7 if x=='July'\
else 8 if x=='August'\
else 9 if x=='September'\
else 10 if x=='October'\
else 11 if x=='November'\
else 12 if x=='December'\
else x)
```

```

else 5 if x=='May' \
else 6 if x=='June' \
else 7 if x=='July' \
else 8 if x=='August' \
else 9 if x=='September' \
else 10 if x=='October' \
else 11 if x=='November' \
else 12 if x=='December' \
else x)

```

```

In [59]: #sort dataframe by month
monthly = monthly.sort_values('month_no')
#convert to millions
monthly['box_office_revenue(in millions)'] = monthly['box_office'] / 1000000

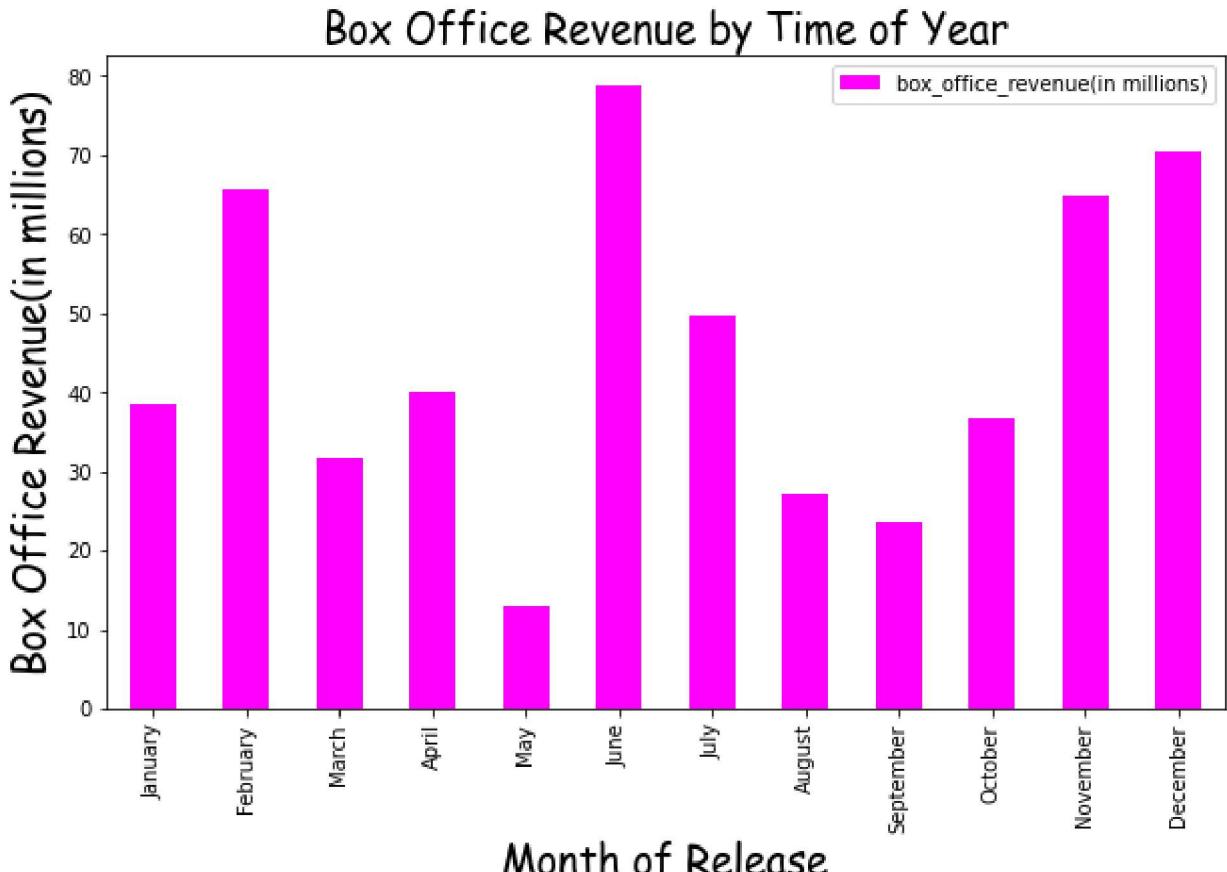
#create graph of box office revenue by month
monthly.plot(x='release_month',y='box_office_revenue(in millions)',kind='bar',color='magenta')

plt.title('Box Office Revenue by Time of Year', size=20, **csfont)
plt.ylabel('Box Office Revenue(in millions)',size=20, **csfont)
plt.xlabel('Month of Release',size=20,**csfont)
#plt.show()
plt.savefig("Revenue by Month.jpeg",bbox_inches='tight') #save as jpg

print('The chart shows more significant revenue during the months of November and December')

```

The chart shows more significant revenue during the months of November and December, as well as June and also February. What is surprising is the revenue during May, which is expected to be higher due to "Star Wars Day" on May 4th.



The chart shows more significant revenue during the months of November and December, as well as June and also February. What is surprising is the revenue during May, which is expected to be higher due to "Star Wars Day" on May 4th.

Using Budgets Dataframe to check profit

In [60]: budgets

			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
2	3	Jun 7, 2019			Dark Phoenix	\$350,000,000	\$42,762,350	\$149,762,350
3	4	May 1, 2015			Avengers: Age of Ultron	\$330,600,000	\$459,005,868	\$1,403,013,963
4	5	Dec 15, 2017			Star Wars Ep. VIII: The Last Jedi	\$317,000,000	\$620,181,382	\$1,316,721,747
...
5777	78	Dec 31, 2018			Red 11	\$7,000	\$0	\$0
5778	79	Apr 2, 1999			Following	\$6,000	\$48,482	\$240,495
5779	80	Jul 13, 2005			Return to the Land of Wonders	\$5,000	\$1,338	\$1,338
5780	81	Sep 29, 2015			A Plague So Pleasant	\$1,400	\$0	\$0
5781	82	Aug 5, 2005			My Date With Drew	\$1,100	\$181,041	\$181,041

5782 rows × 6 columns

In [61]: budgets = pd.read_csv('tn.movie_budgets.csv.gz')

```
#formatting 'gross' strings and the production budget so they can be converted to numer
budgets['domestic_gross'] = budgets['domestic_gross'].str.replace("$","").str.replace(
budgets['worldwide_gross'] = budgets['worldwide_gross'].str.replace("$","").str.replace(
budgets['production_budget'] = budgets['production_budget'].str.replace("$","").str.rep

#calculating the total gross
budgets['total_gross'] = budgets['domestic_gross'] + budgets['worldwide_gross']

#converting to billions
budgets['profit(in billions)'] = (budgets['total_gross'] - budgets['production_budget'])

#slicing month number
budgets['month'] = budgets['release_date'].apply(lambda x: x[0:3])
#slicing year number
budgets['year'] = budgets['release_date'].apply(lambda x: x[-4:])

#formatting month names
budgets['month'] = budgets['month'].apply(lambda x: 'January' if x=='Jan'\
```

```

        else 'February' if x=='Feb' \
else 'March' if x=='Mar' \
else 'April' if x=='Apr' \
else 'May' if x=='May' \
else 'June' if x=='Jun' \
else 'July' if x=='Jul' \
else 'August' if x=='Aug' \
else 'September' if x=='Sep' \
else 'October' if x=='Oct' \
else 'November' if x=='Nov' \
else 'December' if x=='Dec' \
else x)

#quick reformat to make nicer names to plot
budgets['budget'] = budgets['production_budget'] / 10000000000
budgets['profit'] = budgets['profit(in billions)']
budgets['total_gross(in billions)'] = budgets['total_gross'] / 1000000000
budgets['% of budget'] = (budgets['profit'] / budgets['budget'])*100

```

In [62]:

```

#grouping dataframe by month and average profit per month
monthly = budgets.groupby('month')['profit(in billions)'].mean().reset_index()

#assigning month number to sort the months
monthly['month_no'] = monthly['month'].apply(lambda x: 1 if x=='January' \
                                              else 2 if x=='February' \
                                              else 3 if x=='March' \
                                              else 4 if x=='April' \
                                              else 5 if x=='May' \
                                              else 6 if x=='June' \
                                              else 7 if x=='July' \
                                              else 8 if x=='August' \
                                              else 9 if x=='September' \
                                              else 10 if x=='October' \
                                              else 11 if x=='November' \
                                              else 12 if x=='December' \
                                              else x)

#sort by month number
monthly = monthly.sort_values('month_no')

```

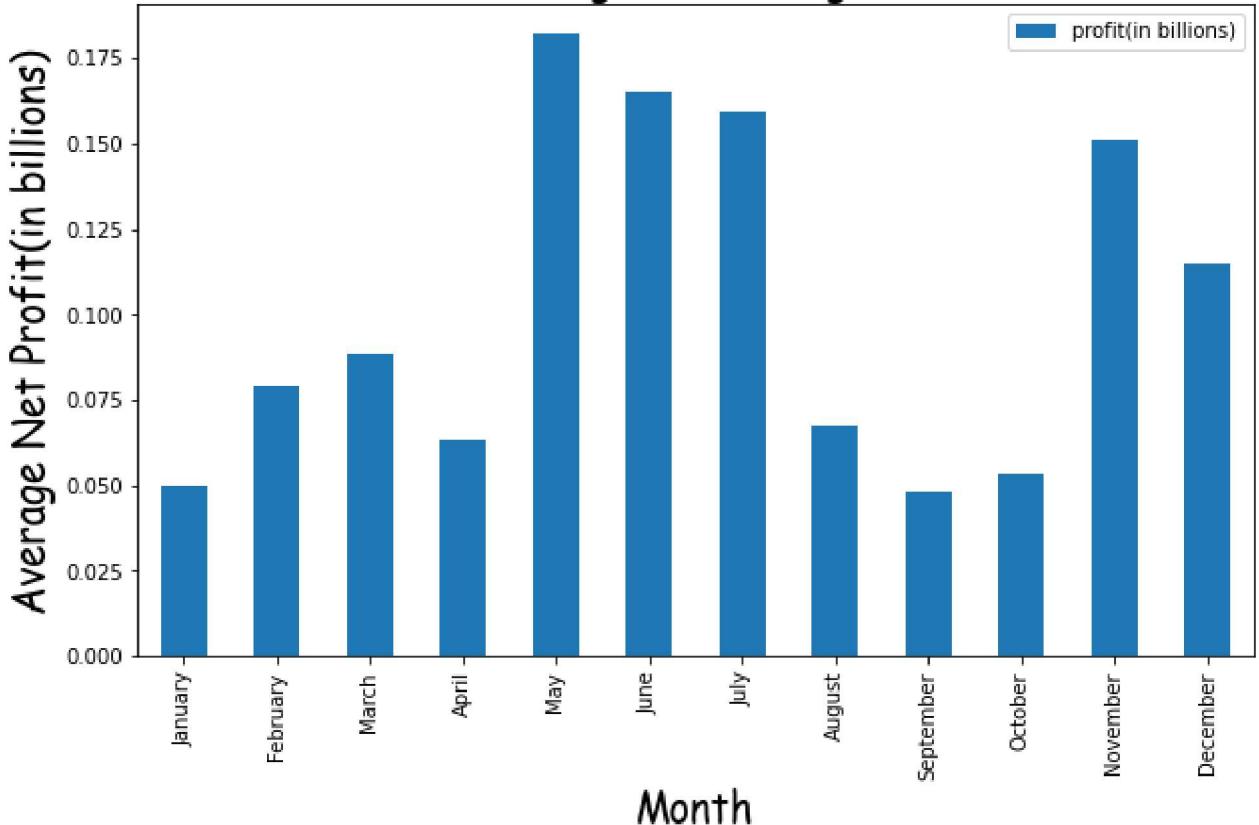
In [63]:

```

#create bar chart for average profit by month
monthly.plot(x='month',y='profit(in billions)', kind='bar', figsize=(10,6))
plt.title('Production Budget vs Average Net Profit', size=20, **csfont)
plt.ylabel('Average Net Profit(in billions)',size=20, **csfont)
plt.xlabel('Month',size=20,**csfont)
plt.savefig("Bar_chart_month_profit.jpeg",bbox_inches='tight') #save as jpg

```

Production Budget vs Average Net Profit



```
In [64]: #create bar chart for average profit by month
#assigning month number to sort the months
monthly_per = budgets.groupby('month')[ '% of budget'].mean().reset_index()

monthly_per['month_no'] = monthly_per['month'].apply(lambda x: 1 if x=='January' \
else 2 if x=='February' \
else 3 if x=='March' \
else 4 if x=='April' \
else 5 if x=='May' \
else 6 if x=='June' \
else 7 if x=='July' \
else 8 if x=='August' \
else 9 if x=='September' \
else 10 if x=='October' \
else 11 if x=='November' \
else 12 if x=='December' \
else x)

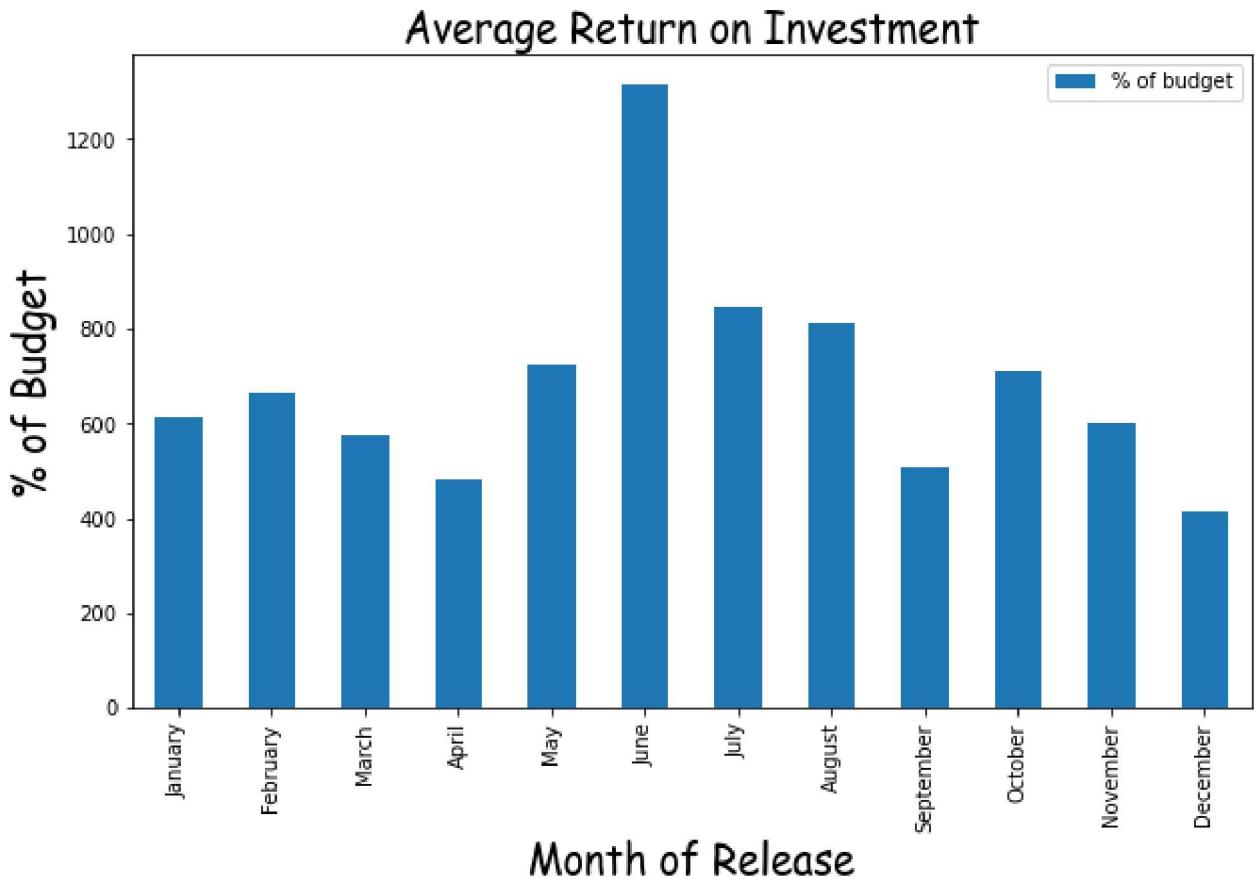
monthly_per = monthly_per.sort_values('month_no')

monthly_per.plot(x='month',y='% of budget', kind='bar', figsize=(10,6))
plt.title('Average Return on Investment', size=20, **csfont)
plt.ylabel('% of Budget',size=20, **csfont)
plt.xlabel('Month of Release',size=20,**csfont)
plt.savefig("Bar_chart_month_profit.jpeg",bbox_inches='tight') #save as jpg

print('As shown in the chart below, the average return on the movies budget by month te
```

As shown in the chart below, the average return on the movies budget by month tells a different story compared to the average profit. This metric is a normalized way to look at success of the movie. The return on budget is shown to be fairly consistent by mo

nth other than the month of June, which is a summer month(historically a good season for movies).



In []:

If you have a larger budget, will that lead to a higher profit?

In [65]:

```
#scatter plot of budget vs net profit

m = budgets['budget'].corr(budgets['profit'])
b = 1.37
net_profit_reg = m * budgets['budget'] + b
fig, ax1 = plt.subplots(figsize=(10,8))

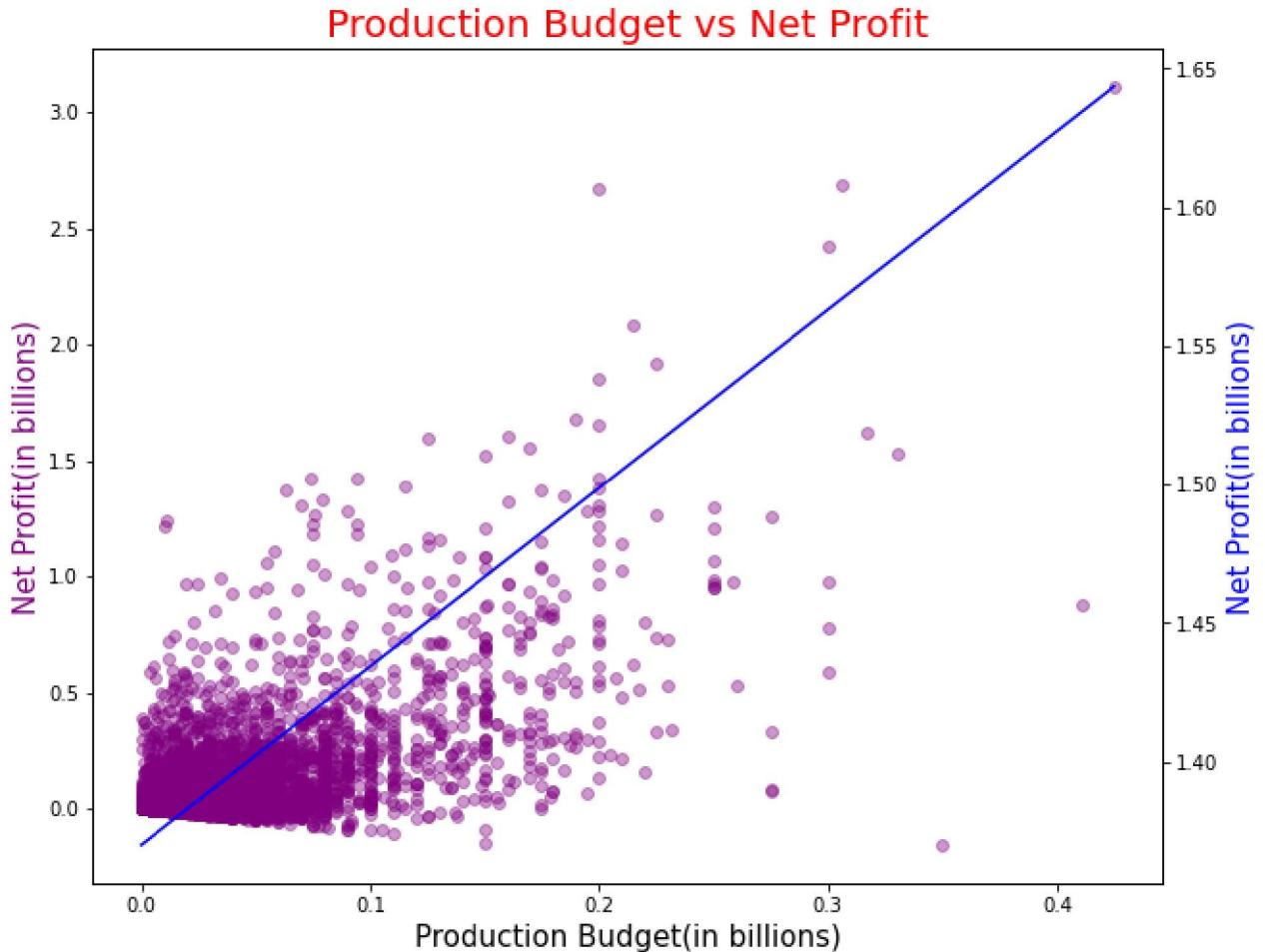
ax2 = ax1.twinx()
ax1.scatter(x=budgets['budget'], y=budgets['profit'], color='purple', alpha=0.4)
ax2.plot(budgets['budget'], net_profit_reg, 'b-')
ax1.set_title('Production Budget vs Net Profit', size=20, color='r')
ax1.set_xlabel('Production Budget(in billions)', size=15)
ax1.set_ylabel('Net Profit(in billions)', color='purple', size=15)
ax2.set_ylabel('Net Profit(in billions)', color='b', size=15)

## Correlation coefficient - line of regression
budget_profit_corr = budgets['budget'].corr(budgets['profit'])

plt.show()

print("The Correlation Coefficient is: {}. This is a sign that the budget has a positive")
```

```
weak correlation to profit of the film. With a weak correlation, it makes sense to
keeping the budget low as it is where the highest \
concentration of the data is.".format(budget_profit_corr))
```



The Correlation Coefficient is: 0.643579874967001. This is a sign that the budget has a positive, but weak correlation to profit of the film. With a weak correlation, it makes sense to try keeping the budget low as it is where the highest concentration of the data is.

```
In [66]: budgets['total gross(billions)'] = budgets['total_gross'] / 1000000000
```

```
In [67]: budget_gross_corr = budgets['total gross(billions)'].corr(budgets['budget'])
budget_gross_corr
```

```
Out[67]: 0.7399121495609374
```

```
In [68]: # Making a regression Line based on the correlation
```

```
# correlation coefficient
m = budgets['total gross(billions)'].corr(budgets['budget'])

# estimated y-intercept
b = 1.37

# regression equation
net_profit_reg = m * budgets['budget'] + b

#plot
fig, ax1 = plt.subplots(figsize=(10,8))
```

```

#create a double y-axis
ax2 = ax1.twinx()

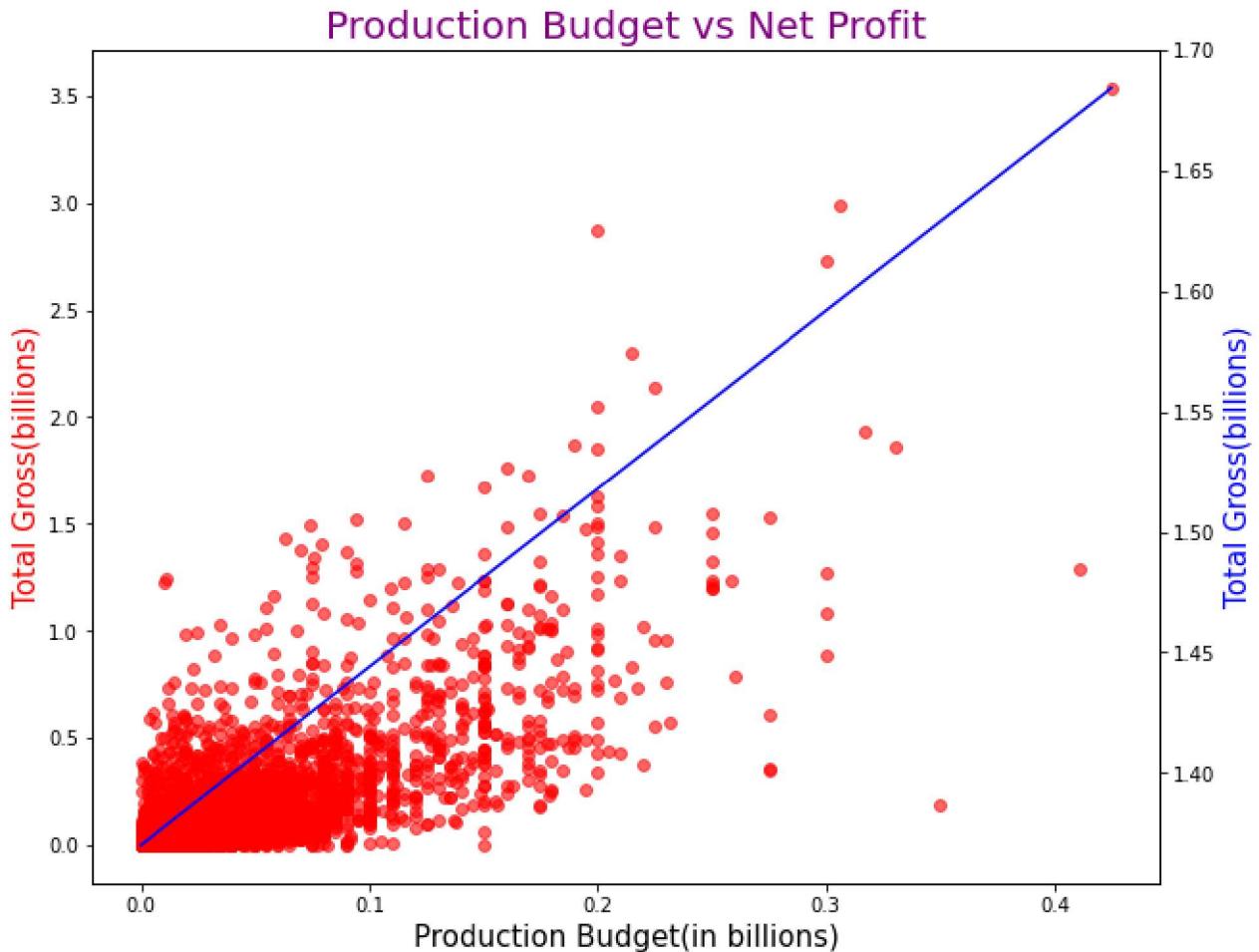
ax1.scatter(x=budgets['budget'], y=budgets['total gross(billions)'], color='r', alpha=0.5)
ax2.plot(budgets['budget'], net_profit_reg, 'b-')
ax1.set_title('Production Budget vs Net Profit', size=20, color='purple')
ax1.set_xlabel('Production Budget(in billions)', size=15)
ax1.set_ylabel('Total Gross(billions)', color='r', size=15)
ax2.set_ylabel('Total Gross(billions)', color='b', size=15)

# plt.show()
plt.savefig('production_budget_vs_net_profit.jpeg', bbox_inches='tight')

print("The Correlation Coefficient is: {}. This is a sign that the budget has a positive weak-medium correlation to gross revenue of the film. With a medium correlation, it makes sense to try keeping the budget low as it is where the highest concentration of the data is. The risk with a higher budget does appear to be too high as it grows.".format(budget_gross_corr))

```

The Correlation Coefficient is: 0.7399121495609374. This is a sign that the budget has a positive, but weak-medium correlation to gross revenue of the film. With a medium correlation, it makes sense to try keeping the budget low as it is where the highest concentration of the data is. The risk with a higher budget does appear to be too high as it grows.



In [69]: cd ..

C:\Users\alevi\Documents\Flatiron\dsc-data-science-env-config\Course_Folder\Phase_1\Phase_1_Project\Phase-1-Project---Movie-Analysis

IMDB data analysis based on genre

Querying the movie_basics and movie_ratings

```
In [70]: import sqlite3 as sql

conn = sql.connect('im.db')
df = pd.read_sql('''SELECT *
                  FROM movie_basics
                  JOIN movie_ratings
                  USING(movie_id)
                  JOIN movie_akas
                  USING(movie_id)
                  ORDER BY numvotes DESC
                  ''', conn)
```

Merge with budgets dataframe

```
In [71]: imdb_budgets = pd.merge(df,budgets, left_on='original_title',right_on='movie',how='inner')
```

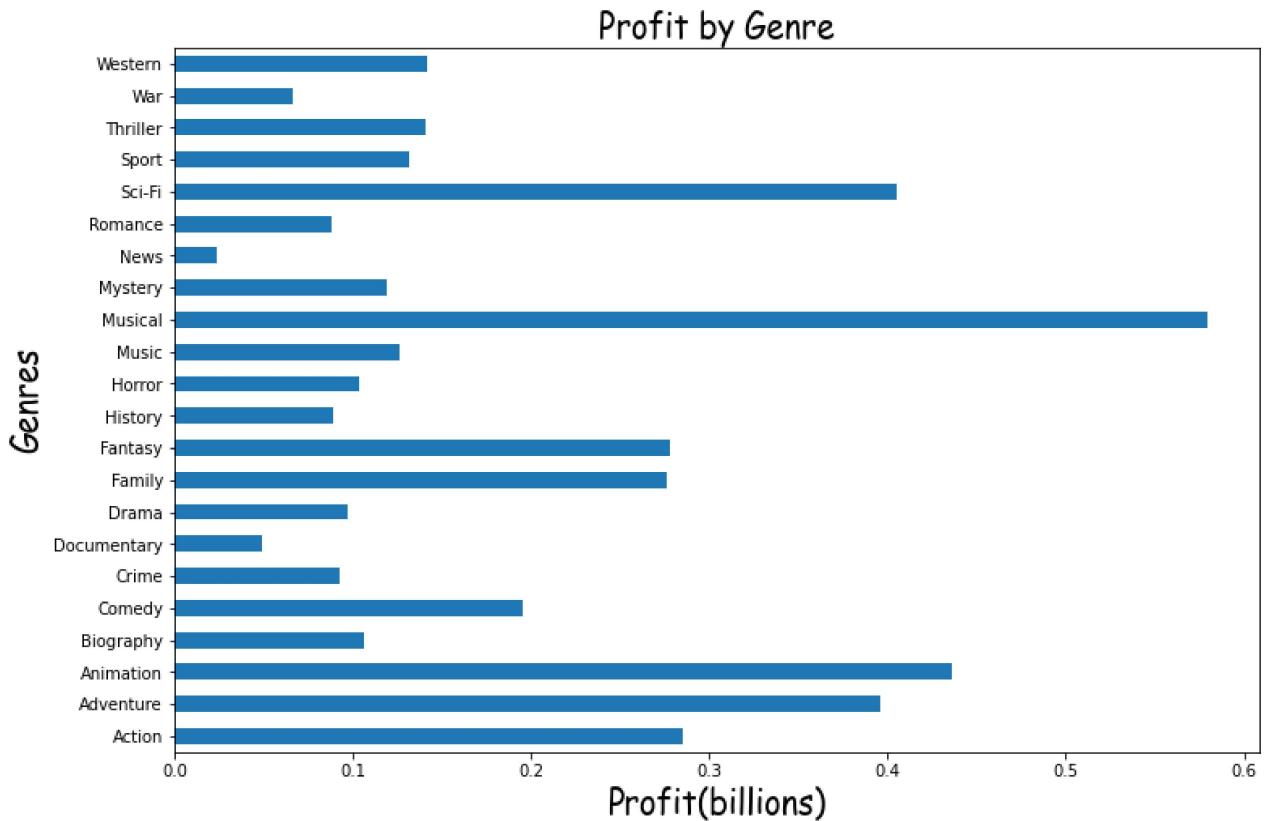
Checking Profit by Genre

```
In [72]: imdb_budgets['genres'] = imdb_budgets['genres'].str.split(',')
imdb_budgets_genres = imdb_budgets.explode('genres')
```

```
In [73]: imdb_budgets_genres_by_profit = imdb_budgets_genres.groupby('genres')['profit'].mean()
imdb_budgets_genres_by_profit.plot(x='genres',y='profit',kind='barh',figsize=(12,8))
plt.xlabel('Profit(billions)',size=20,**csfont)
plt.ylabel('Genres',size=20,**csfont)
plt.title('Profit by Genre',size=20,**csfont)
plt.savefig('Profit_by_Genre.jpeg')
```

```
print('Earlier in the analysis, it was shown that the box office revenue was heavily favored towards science fiction and fantasy as well as action and adventure. However, when looking at the data for profit the spread is slightly different. Musicals appear to be the most profitable genres along with Scifi and animation performing at a high rate. While not as strong as at the box office, action, fantasy, family and adventure also appear to be very profitable genres as well.
```

Earlier in the analysis, it was shown that the box office revenue was heavily favored towards science fiction and fantasy as well as action and adventure. However, when looking at the data for profit the spread is slightly different. Musicals appear to be the most profitable genres along with Scifi and animation performing at a high rate. While not as strong as at the box office, action, fantasy, family and adventure also appear to be very profitable genres as well.



Advice for Microsoft

- 1) Keep the budget low! For the budget, all the data points to a more reliable profit and return on investment. The correlation for budget vs profit appears to be on the weaker side, so a larger budget doesn't necessarily mean a direct return on investment.
- 2) Best Genres for revenue: Action and Adventure, Science Fiction and Fantasy Best Genres for profit: Musicals, SciFi, Animation, Adventure examples - Avatar, Avengers,
- 3) Summer time! With consideration of the summer months being the most profitable, and June showing to be the highest month for return on investment, it makes sense to try and maximize movies released during that time.