example

Microsoft Studios 2022 Strategy Recommendations

Authors: Robert Cauvy

Table of Contents

Overview

This project leverages tools from base Python and pandas to provide exploratory data analysis that helps the head of Microsoft's fledgling movie studio decide what type of films to create. The actionable insights have been distilled from data sets provided by IMDB and The Numbers.

Business Problem

After making the business decision to compete at the Box Office with original content. The new head of the studio must determine which projects should be prioritized for allocating resources. Since this is a brand new movie studio, decision making will rely on public and readily available historical data of past movie releases. Because optimal financial performance is the ultimate objective of this new business division, the analysis will use gross profits and rate of return as the primary metrics for benchmarking variables involving past films' genres, release dates and directors. Using these measurables to provide an unbiased, rational baseline for the studio head to begin with the early planning of the content and release strategy. Once these key decisions have been approved, further analysis can be conducted.

Data Understanding

The data used in this project comes from sources, IMDB and The Numbers database. Specifically the data stored in the files; imdb.title.basics.csv, tn.movie_budgets.csv, imdb.title.crew.csv, imdb.name.basics.csv. After combining data from these sources, financial performance could be pulled from each individual title and used to measure variables surrounding the genre, release and director.

In [1]:
```
# Importing standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
from matplotlib.ticker import FuncFormatter
from collections import Counter
pd.set_option('display.max_columns',0)
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
# Code to explore the data
# Sourced from jirvingphd at Flatiron School
# https://www.youtube.com/watch?v=BKpSVE0VF0U&ab_channel=JamesIrving

import glob, os
fpath = 'zippedData/'
os.listdir(fpath)
```

Out[2]:
```
['imdb.title.crew.csv.gz',
 'tmdb.movies.csv.gz',
 'imdb.title.akas.csv.gz',
 'imdb.title.ratings.csv.gz',
 'imdb.name.basics.csv.gz',
 'rt.reviews.tsv.gz',
 'imdb.title.basics.csv.gz',
 'rt.movie_info.tsv.gz',
 'tn.movie_budgets.csv.gz',
 'bom.movie_gross.csv.gz',
 'imdb.title.principals.csv.gz']
```

In [3]:
```
# Sourced from jirvingphd at Flatiron School
# https://www.youtube.com/watch?v=BKpSVE0VF0U&ab_channel=JamesIrving
query = fpath+'*gz'

file_list = glob.glob(query)
```

In [4]:
```
# Sourced from jirvingphd at Flatiron School
# https://www.youtube.com/watch?v=BKpSVE0VF0U&ab_channel=JamesIrving
tables = {}
for file in file_list:
#     print('---'*20)
```

```
    file_name = file.replace('zippedData/','').replace('.','_')

#     print(file_name)

    if '.tsv.gz' in file:
        temp_df = pd.read_csv(file, sep='\t', encoding='latin-1')
    else:
        temp_df = pd.read_csv(file)
#     display(temp_df.head(),temp_df.tail())


    tables[file_name] = temp_df
```

Data Preparation

Now that we has selected the data tables that contain the information we need for our analysis, it must be prepared in a way that help answer questions stemming from the business problem. First, the IMDB Title Basics which contains information about individual films, such as titles and genres is merged with The Numbers Budgets data tables, at the at the movie title column to bring in release dates, production budgets and gross revenues form the box office. After removing duplicates, new columns are added to benchmark financial performances of each title. The two metrics used for this analysis are gross profits and return on investment. Before creating new columns to the dataframe, the financial data must be converted into integers so that operations can be applied. The gross profit column was created by subtracting the worldwide gross revenues from the production budget and ROI is a ratio obtained by dividing worldwide gross profits by production budget and multiplying by 100.

Since it can be assumed that the new studio head will be judged by how profitable his business division becomes gross profits will be the more important metric. And ROI be strategically used to optimize the studios resources and corporate overhead such as sound stages, utilities and human capital. Next, the release date column will be seperated into new colummns also from strings to integers. This allows to filter out much older titles and suggest when releases should be scheduled. Later in the analysis, we'll breakout each titles genre from a single string, seperated by commas into new row. SO that we can apply the financial metrics to genre categories. And lastly, the dataframe will be merged with IMDB's Film Crew and Basic Names data tables to identify and recomment which directors should be hired to work on films that fall into the priority genres.

In [5]:
```
tn_movie_budgets = tables['tn_movie_budgets_csv_gz'].copy()
tn_movie_budgets
```

Out[5]:

	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

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
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 [6]:
```
imdb_title_basics = tables['imdb_title_basics_csv_gz'].copy()
imdb_title_basics
```

Out[6]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genre
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Dram
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantas
...	...	...	...	...	...	.
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Dram
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentar
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comed
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	Nal

	tconst	primary_title	original_title	start_year	runtime_minutes	genre
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentar

146144 rows × 6 columns

In [7]:
```
# Merging IMDB basics table with The Numbers budget data
df = pd.merge(tn_movie_budgets,imdb_title_basics,left_on='movie',
              right_on='original_title')
```

In [8]:
```
df[df.duplicated(keep=False,
                 subset=['movie','original_title','domestic_gross'])]
```

Out[8]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
27	39	May 14, 2010	Robin Hood	$210,000,000	$105,487,148	$322,459,006	tt0955308
28	39	May 14, 2010	Robin Hood	$210,000,000	$105,487,148	$322,459,006	tt2363363
29	39	May 14, 2010	Robin Hood	$210,000,000	$105,487,148	$322,459,006	tt4532826
30	39	May 14, 2010	Robin Hood	$210,000,000	$105,487,148	$322,459,006	tt6858500
31	39	May 14, 2010	Robin Hood	$210,000,000	$105,487,148	$322,459,006	tt8558276
...	...	...	...	...	...	...	...
3521	51	Apr 21, 2015	Ten	$25,000	$0	$0	tt2309562
3522	51	Apr 21, 2015	Ten	$25,000	$0	$0	tt2496400
3523	51	Apr 21, 2015	Ten	$25,000	$0	$0	tt6415838
3531	68	Jul 6, 2001	Cure	$10,000	$94,596	$94,596	tt1872026
3532	68	Jul 6, 2001	Cure	$10,000	$94,596	$94,596	tt5936960

1735 rows × 12 columns

In [9]:
```
df = df.drop_duplicates(keep='first',
                        subset=['movie','original_title','domestic_gross'])
df[df.duplicated(keep=False,
                 subset=['movie','original_title','domestic_gross'])]
df
```

Out[9]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
0	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875	tt1298

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
1	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	tt656
2	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	tt239
3	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200	tt415
4	9	Nov 17, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209	tt097
...	...	...	...	...	...	...	
3531	68	Jul 6, 2001	Cure	$10,000	$94,596	$94,596	tt187
3533	70	Apr 1, 1996	Bang	$10,000	$527	$527	tt661
3534	73	Jan 13, 2012	Newlyweds	$9,000	$4,584	$4,584	tt188
3535	78	Dec 31, 2018	Red 11	$7,000	$0	$0	tt783
3536	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0	tt210

2330 rows × 12 columns

In [10]:
```
#Change financial data from strings to integers

cols_to_ints  = ['worldwide_gross', 'production_budget', 'domestic_gross']

for col in cols_to_ints:


    df[col] = df[col].str.replace('$','').str.replace(',','')
    df[col] = df[col].astype(int)
```

In [11]:
```
#Create the a column for the 1st financial metric, Gross Profit
df['profit'] = df['worldwide_gross'] - df['production_budget']
```

In [12]:
```
#Create the a column for the 2nd financial metric, ROI
df['ROI'] = (df['profit'] / df['production_budget']) * 100
df
```

Out[12]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
0	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt129
1	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	tt656

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
2	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	tt239!
3	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	2048134200	tt415
4	9	Nov 17, 2017	Justice League	300000000	229024295	655945209	tt097
...	...	...	...	...	...	...	
3531	68	Jul 6, 2001	Cure	10000	94596	94596	tt187
3533	70	Apr 1, 1996	Bang	10000	527	527	tt661
3534	73	Jan 13, 2012	Newlyweds	9000	4584	4584	tt188
3535	78	Dec 31, 2018	Red 11	7000	0	0	tt783
3536	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	tt210

2330 rows × 14 columns

In [13]:

```
## Add release month and years columns - convert to pd.datetime
df['release_date'] = pd.to_datetime(df['release_date'])
df['release_month'] = df['release_date'].dt.month
df['release_year'] = df['release_date'].dt.year
df
```

Out[13]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
0	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt129
1	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt656
2	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt239!
3	7	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt415
4	9	2017-11-17	Justice League	300000000	229024295	655945209	tt097
...	...	...	...	...	...	...	
3531	68	2001-07-06	Cure	10000	94596	94596	tt187
3533	70	1996-04-01	Bang	10000	527	527	tt661
3534	73	2012-01-13	Newlyweds	9000	4584	4584	tt188
3535	78	2018-12-31	Red 11	7000	0	0	tt783

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	tc
3536	81	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107

2330 rows × 16 columns

In [14]:
```
#Removing unnecessary columns
df.drop(['id','primary_title','original_title','start_year'],axis=1,
        inplace=True)
```

In [15]:
```
#How far back does the data go?
df[df.release_year == df.release_year.min()]
```

Out[15]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst	runt
2217	1915-02-08	The Birth of a Nation	110000	10000000	11000000	tt4196450	

In [16]:
```
#1915! Let's remove some of the older data points for a more modern representati
#Remove Movies Older Than 50 Years - drops 31 records and leaves 2299
df_modern = df[df['release_year'] > 1970]
df_modern
```

Out[16]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
2	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756
4	2017-11-17	Justice League	300000000	229024295	655945209	tt0974015
...	...	...	...	...	...	...
3531	2001-07-06	Cure	10000	94596	94596	tt1872026
3533	1996-04-01	Bang	10000	527	527	tt6616538
3534	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
3535	2018-12-31	Red 11	7000	0	0	tt7837402
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2299 rows × 12 columns

In [17]:
```
# Adding a column to give 'Big Budget' and 'Profitable' attributes
df_modern['big_budget'] = df_modern['production_budget'] >= 75000000
df_modern['profitable'] = df_modern['profit'] > 0
df_modern
```

Out[17]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
2	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756
4	2017-11-17	Justice League	300000000	229024295	655945209	tt0974015
...	...	...	...	...	...	...
3531	2001-07-06	Cure	10000	94596	94596	tt1872026
3533	1996-04-01	Bang	10000	527	527	tt6616538
3534	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
3535	2018-12-31	Red 11	7000	0	0	tt7837402
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2299 rows × 14 columns

In [18]:
```
# Format to replace scientific notation
pd.options.display.float_format = '{:,.0f}'.format
```

In [19]:
```
df_modern.describe()
```

Out[19]:

	production_budget	domestic_gross	worldwide_gross	runtime_minutes	profit	
count	2,299	2,299	2,299	2,134	2,299	2,
mean	35,839,223	44,338,651	106,821,877	102	70,982,654	
std	48,886,948	74,352,560	201,509,441	22	166,685,029	1,
min	1,400	0	0	1	-200,237,650	-
25%	5,000,000	553,436	2,433,423	90	-2,000,000	
50%	18,000,000	17,686,929	30,628,981	101	10,369,708	

	production_budget	domestic_gross	worldwide_gross	runtime_minutes	profit	
75%	43,500,000	54,286,573	108,816,294	114	68,488,334	:
max	410,600,000	700,059,566	2,208,208,395	189	2,008,208,395	41,!

In [20]:

```
# Creating a new DF for measure performances by genre
df_genres = df_modern.copy()
df_genres['genres_list'] = df_genres['genres'].str.split(pat=',')
df_genres.head()
```

Out[20]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst	rur
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650	
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702	
2	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427	
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756	
4	2017-11-17	Justice League	300000000	229024295	655945209	tt0974015	

In [21]:

```
#  Inster comment
df_genres = df_genres.explode('genres_list')
df_genres
```

Out[21]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
...	...	...	...	...	...	...
3535	2018-12-31	Red 11	7000	0	0	tt7837402
3535	2018-12-31	Red 11	7000	0	0	tt7837402
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

5345 rows × 15 columns

In [22]:

```
# Merge principals table with df_modern to merge mconst
imdb_crew = tables['imdb_title_crew_csv_gz']
df_crew = pd.merge(df_modern,imdb_crew,on='tconst')
df_crew
```

Out[22]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
2	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756
4	2017-11-17	Justice League	300000000	229024295	655945209	tt0974015
...	...	...	...	...	...	...
2294	2001-07-06	Cure	10000	94596	94596	tt1872026
2295	1996-04-01	Bang	10000	527	527	tt6616538
2296	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
2297	2018-12-31	Red 11	7000	0	0	tt7837402

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
2298	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2299 rows × 16 columns

In [23]:
```
df_crew['director_list'] = df_crew['directors'].str.split(pat=',')
```

In [24]:
```
df_directors = df_crew.explode('director_list')
df_directors
```

Out[24]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
2	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756
3	2018-04-27	Avengers: Infinity War	300000000	678815482	2048134200	tt4154756
...	...	...	...	...	...	...
2295	1996-04-01	Bang	10000	527	527	tt6616538
2296	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
2297	2018-12-31	Red 11	7000	0	0	tt7837402
2298	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644
2298	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2551 rows × 17 columns

In [25]:
```
#Merge nconst with wirter and director names
imdb_names_basics = tables['imdb_name_basics_csv_gz']
df_directors = pd.merge(df_directors,imdb_names_basics,
                        left_on='director_list',right_on='nconst')

df_directors
```

Out[25]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2018-12-19	Mary Poppins Returns	130000000	171958438	341528518	tt5028340
2	2014-12-25	Into the Woods	56200000	128002372	213116401	tt2180411
3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
...	...	...	...	...	...	...
2517	1996-04-01	Bang	10000	527	527	tt6616538
2518	1996-04-01	Bang	10000	527	527	tt6616538
2519	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
2520	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644
2521	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2522 rows × 23 columns

Data Analysis

First thing to look at is which attributes are most closely correlated with financial performance.

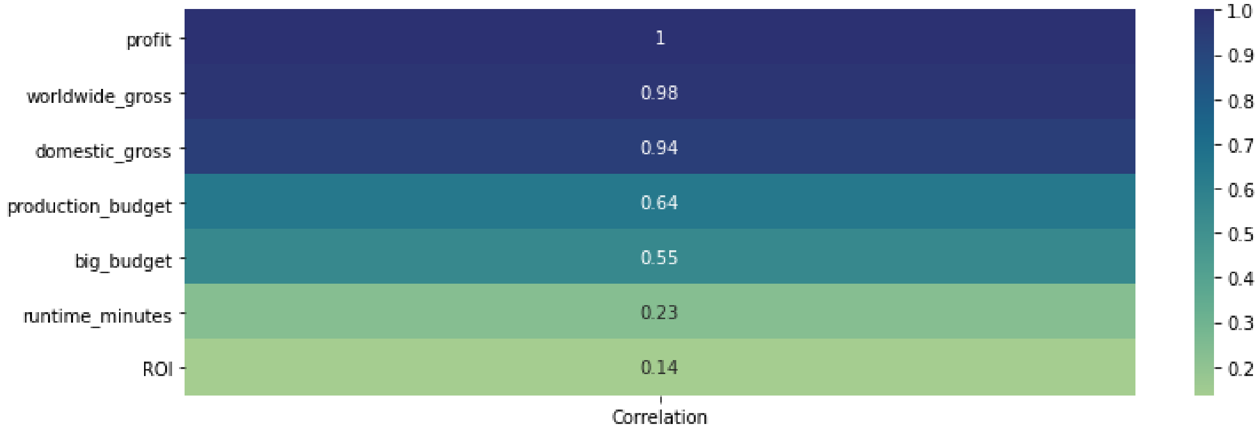
Q. What attributes are the most correlated with gross profit?

In [26]:
```
# Measuring how attributes in the DF are correlated to Gross profits
features = df_modern.drop(columns=['release_year','profitable',
                                    'release_month'])
ft_corr = features.corrwith(df_modern['profit']).to_frame('Correlation')
```

In [27]:
```
# Sorting values in descending order
ft_corr.sort_values('Correlation',ascending=False,inplace=True)
```

In [28]:
```
# Plotting the values to a heatmap
```

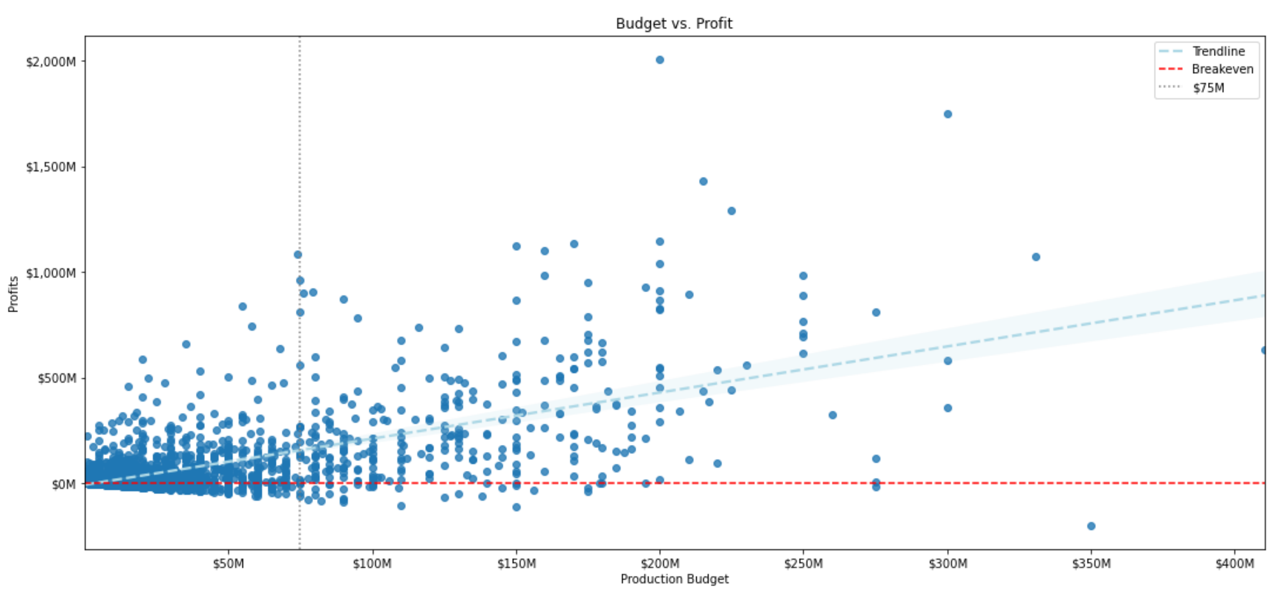
```
fig, ax = plt.subplots(figsize=(12, 4))

sns.heatmap(ft_corr,cmap='crest',annot=True);
```



What does the relationship between production budget and profit look like?

In [29]:
```
# Defining a function to format numbers from scientific notation to be used
# later in visualizations and analysis
# Sourced code from https://matplotlib.org/3.1.1/gallery/ticks_and_spines/tick-f
def millions(x,pos):
    return f"${int(x*1e-6):,}M"

price_fmt_mill = FuncFormatter(millions)
```

In [32]:
```
fig, ax = plt.subplots(figsize=(18, 8))
sns.regplot(data=df_modern,ax=ax,x='production_budget',y='profit',
            line_kws={'color':'lightblue','ls':'--','label':'Trendline'})
ax.set(xlabel="Production Budget",ylabel="Profits",
       title="Budget vs. Profit")
ax.axhline(0,color='red',linestyle='--',label="Breakeven")
ax.axvline(75000000,color='grey',ls=':',label='$75M')

ax.legend()
ax.yaxis.set_major_formatter(price_fmt_mill)
ax.xaxis.set_major_formatter(price_fmt_mill);
```



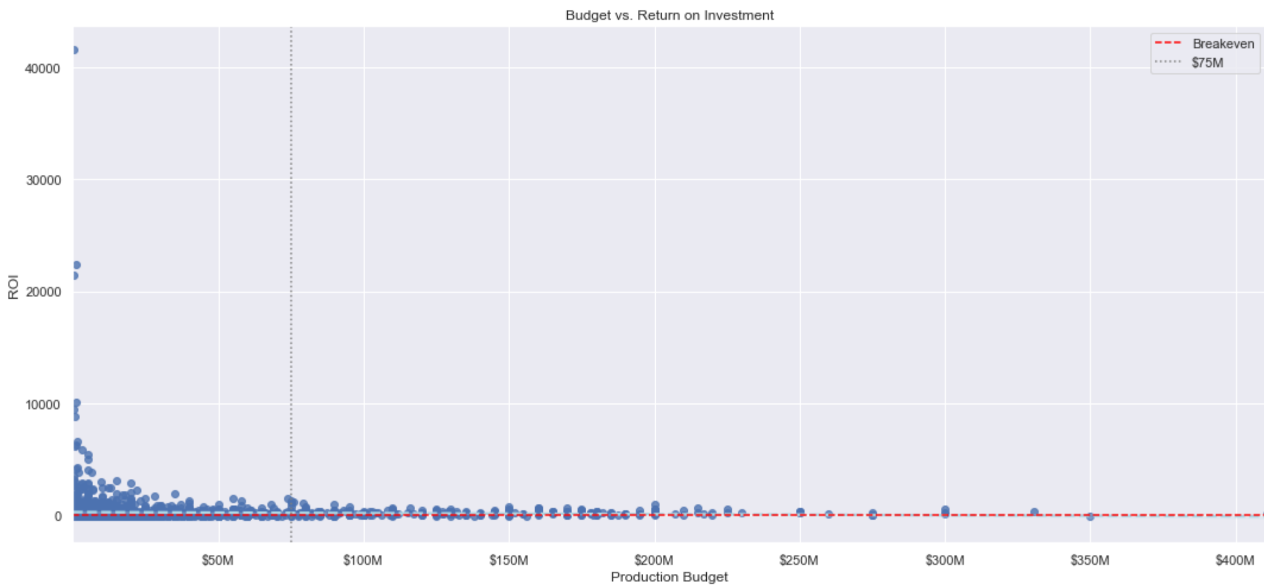
There is clearly a positive correlation and a number of outliers.

91% of all movies with production budgets over $75 million are commercially successful.

While those with budgets under $50M are only 56% profitable.

How about budgets and ROI?

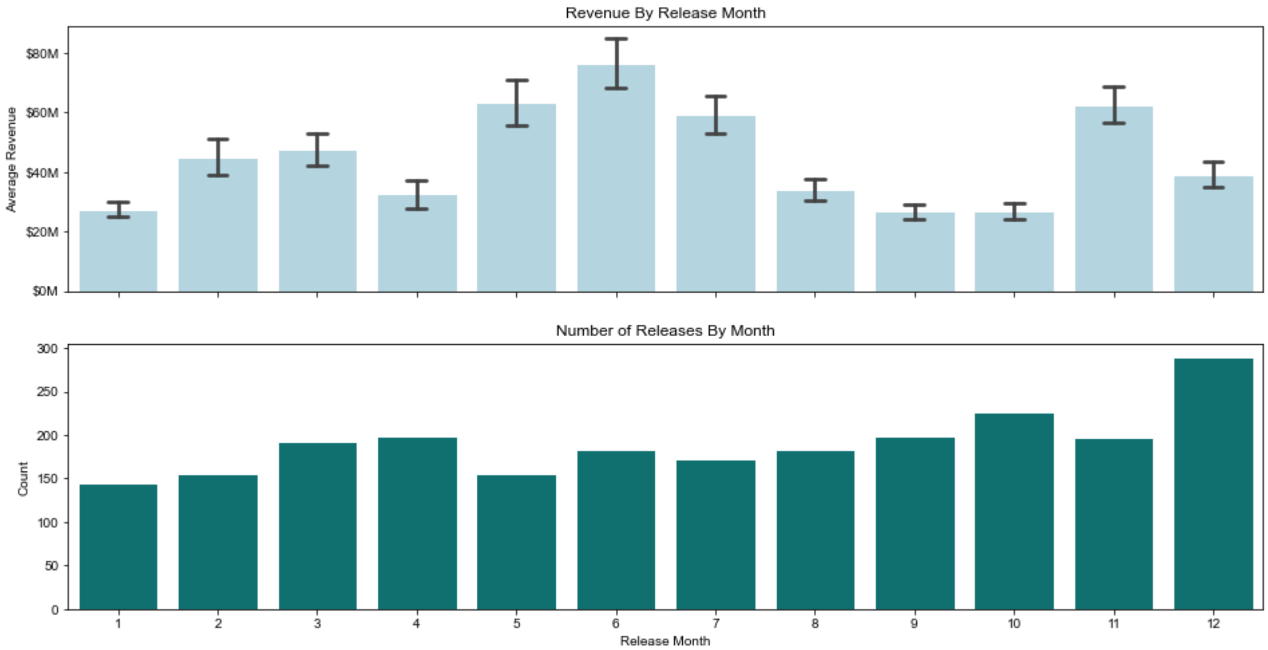
In [73]:
```
fig, ax = plt.subplots(figsize=(18, 8))
sns.regplot(data=df_modern,ax=ax,x='production_budget',y='ROI',
            line_kws={'color':'lightblue','ls':'--'})
ax.set(xlabel="Production Budget",ylabel="ROI",
       title="Budget vs. Return on Investment")
ax.axhline(0,color='red',linestyle='--',label='Breakeven')
ax.axvline(75000000,color='grey',ls=':',label='$75M')
ax.legend()
ax.xaxis.set_major_formatter(price_fmt_mill);
```



Again, there are quite a few significant outliers. And only small budget films have enormous ROIs.

Q. Are Profits Impacted By Release Month?

In [34]:
```
fig, axes = plt.subplots(figsize=(16, 8),nrows=2, sharex=True)

sns.set(style="darkgrid")
sns.barplot(data=df_modern,x='release_month',y='domestic_gross',capsize=.2,
            ci=68,color='lightblue',ax=axes[0])
sns.countplot(data=df_modern,x='release_month',color='teal',ax=axes[1])
axes[0].set(title='Revenue By Release Month', xlabel=None,
            ylabel='Average Revenue')
axes[1].set(title='Number of Releases By Month',ylabel='Count',
            xlabel='Release Month')
axes[0].yaxis.set_major_formatter(price_fmt_mill);
```



Yes! On average, movies opening in May, June, July and November are the most successful.

Should any release months be avoided?

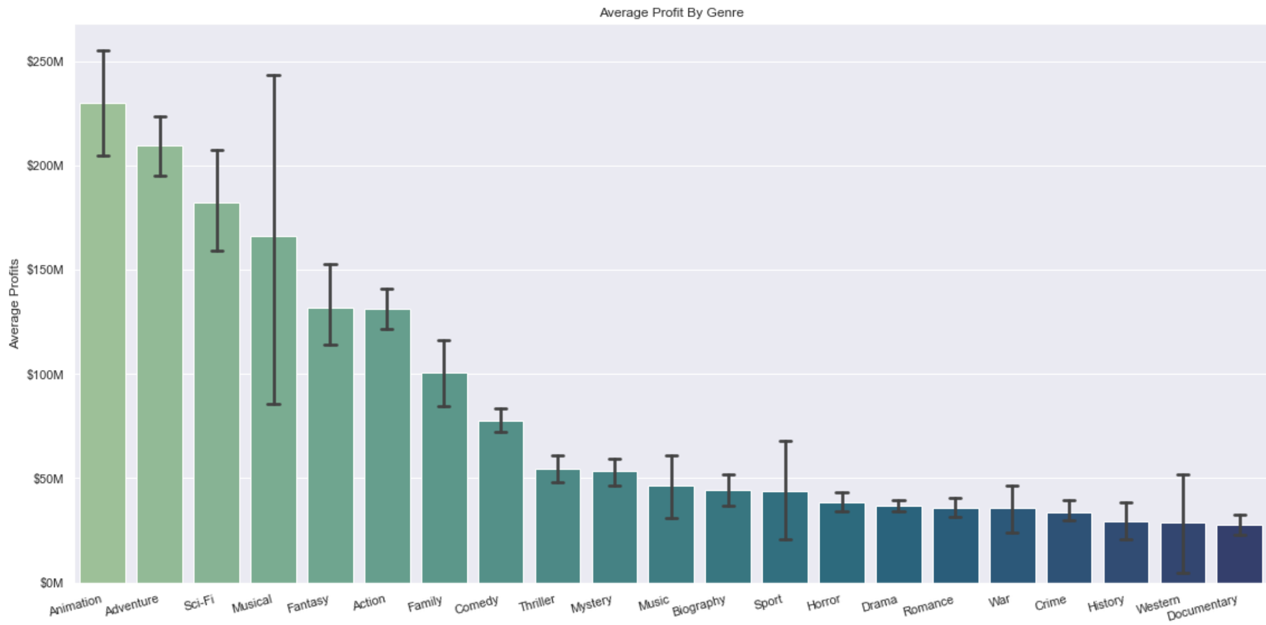
Yes, April, September, October and December have the most releases.

And as we saw above, those months also average low revenues.

Q. What genres should be prioritized?

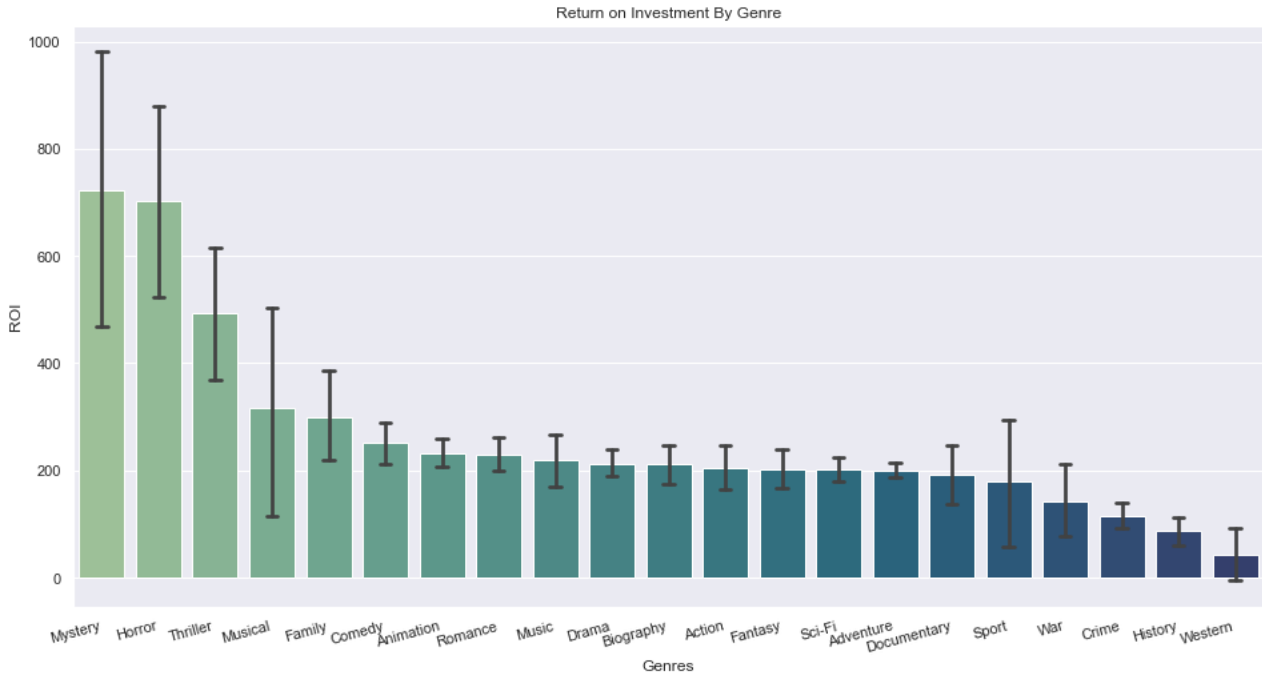
In [35]:
```
# Creating a list in order of descending average Gross Profit by genre
# Using pandas groupby and aggregate functions to obtain an genre means


genre_order = list(df_genres.groupby(['genres_list']).
                   mean()['profit'].sort_values(ascending=False).index)
```

In [36]:
```
# Plotting Bar Charts by genre by average Profit

fig, ax = plt.subplots(figsize=(16, 8))


palette = sns.color_palette("crest",len(genre_order))
sns.barplot(data=df_genres,y='profit',x='genres_list', ax=ax,
            order=genre_order,capsize=.2, ci=68, palette=palette)


ax.set(xlabel=None, ylabel='Average Profits',
            title='Average Profit By Genre')

ax.set_xticklabels(ax.get_xticklabels(),rotation=15,ha='right')
ax.yaxis.set_major_formatter(price_fmt_mill)
plt.tight_layout();
```



In [38]:
```
# Creating a list in order of descending average ROI by genre
# Using pandas groupby and aggregate functions to obtain an genre means


genre_order_roi = list(df_genres.groupby(['genres_list']).
                   mean()['ROI'].sort_values(ascending=False).index)
```

In [39]:
```
# Plotting Bar Charts by genre by average ROI

fig, ax = plt.subplots(figsize=(16, 8))


palette = sns.color_palette("crest",len(genre_order))
sns.barplot(data=df_genres,y='ROI',x='genres_list', ax=ax,
            order=genre_order_roi,capsize=.2, ci=68, palette=palette)

ax.set(xlabel='Genres', ylabel='ROI',
            title='Return on Investment By Genre')

ax.set_xticklabels(ax.get_xticklabels(),rotation=15,ha='right');
```



In [40]:

```
# Creating a new DF filtering out data for only top performing genres
# This new DF will be used later when directors are pulled in

top_genres = ['Animation','Family','Comedy']
top_genres_df = df_genres[df_genres['genres_list'].isin(top_genres)]
top_genres_df
```

Out[40]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
10	2010-11-24	Tangled	260000000	200821936	586477240	tt0398286
10	2010-11-24	Tangled	260000000	200821936	586477240	tt0398286
13	2012-12-14	The Hobbit: An Unexpected Journey	250000000	303003568	1017003568	tt0903624
25	2012-05-25	Men in Black 3	215000000	179020854	654213485	tt1409024
43	2018-06-15	Incredibles 2	200000000	608581744	1242520711	tt3606756
...	...	...	...	...	...	..
3512	2013-12-31	Paraphobia	30000	0	0	tt3123250
3525	2014-12-31	Dry Spell	22000	0	0	tt2375036
3527	2015-04-21	The Front Man	20000	0	0	tt2357398
3530	2006-04-28	Clean	10000	138711	138711	tt6619196
3534	2012-01-13	Newlyweds	9000	4584	4584	tt1880418

898 rows × 15 columns

In [41]:

```
# Same as above but for top genres by Profit
```

```
top_genres_profit = ['Action','Adventure','Sci-Fi','Fantasy']
top_genres_profit_df = df_genres[df_genres['genres_list'].isin(top_genres_profit
top_genres_profit_df
```

Out[41]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tcons
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt129865(
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt129865(
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt129865(
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt656570:
1	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt656570:
...	...	...	...	...	...	..
3491	2015-07-17	Dawn of the Crescent Moon	75000	8799	8799	tt315731ε
3492	2015-09-29	Queen Crab	75000	0	0	tt231945(
3492	2015-09-29	Queen Crab	75000	0	0	tt231945(
3526	2013-01-04	All Superheroes Must Die	20000	0	0	tt183621:
3535	2018-12-31	Red 11	7000	0	0	tt783740:

1305 rows × 15 columns

In [42]:

```
# Same as above but for top genres by ROI

top_genres_roi = ['Mystery','Horror','Thriller','Romance']
top_genres_roi_df = df_genres[df_genres['genres_list'].isin(top_genres_roi)]
top_genres_roi_df
```

Out[42]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst	r
5	2015-11-06	Spectre	300000000	200074175	879620923	tt2379713	
6	2012-07-20	The Dark Knight Rises	275000000	448139099	1084439099	tt1345836	

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst	r
16	2017-04-14	The Fate of the Furious	250000000	225764765	1234846267	tt4630562	
52	2012-11-08	Skyfall	200000000	304360277	1110526981	tt1074638	
65	2013-06-21	World War Z	190000000	202359711	531514650	tt0816711	
...	...	...	...	...	...	...	
3530	2006-04-28	Clean	10000	138711	138711	tt6619196	
3535	2018-12-31	Red 11	7000	0	0	tt7837402	
3535	2018-12-31	Red 11	7000	0	0	tt7837402	
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644	
3536	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644	

1096 rows × 15 columns

In [43]:

```
# Creating a Multi-index df to display releases by release month by genre
genre_months = top_genres_df.groupby(['release_month',
                                      'genres_list']).size().unstack()
genre_months
```

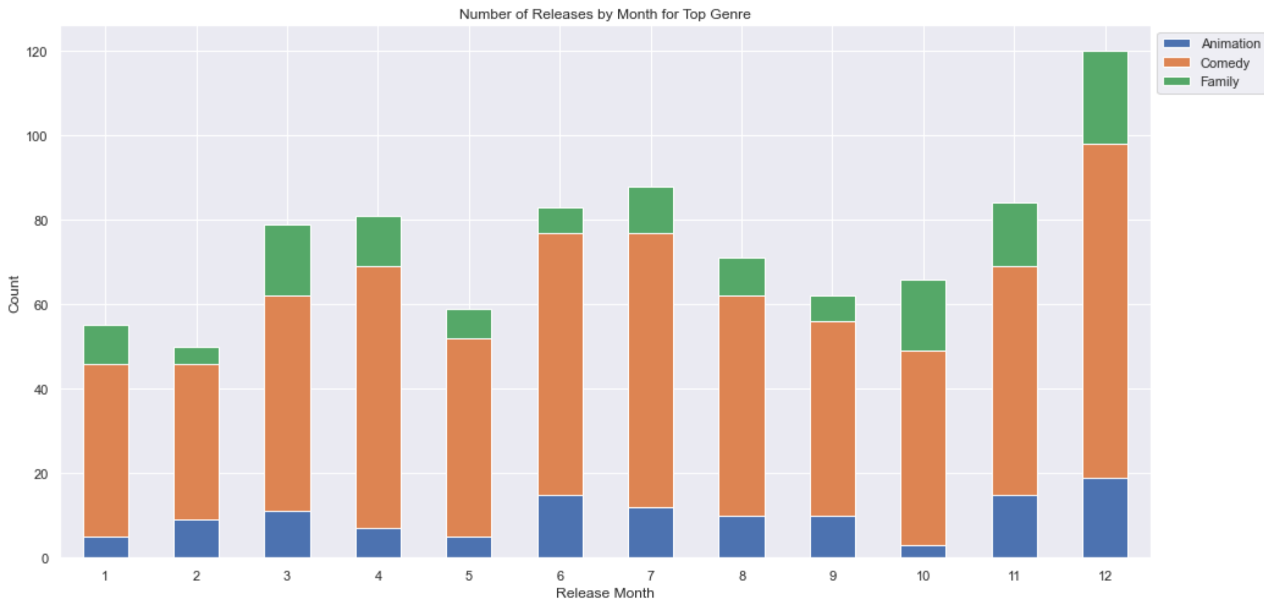
Out[43]:

genres_list	Animation	Comedy	Family
release_month			
1	5	41	9
2	9	37	4
3	11	51	17
4	7	62	12
5	5	47	7
6	15	62	6
7	12	65	11
8	10	52	9
9	10	46	6
10	3	46	17
11	15	54	15
12	19	79	22

Should certain genres be released at different times of the year?

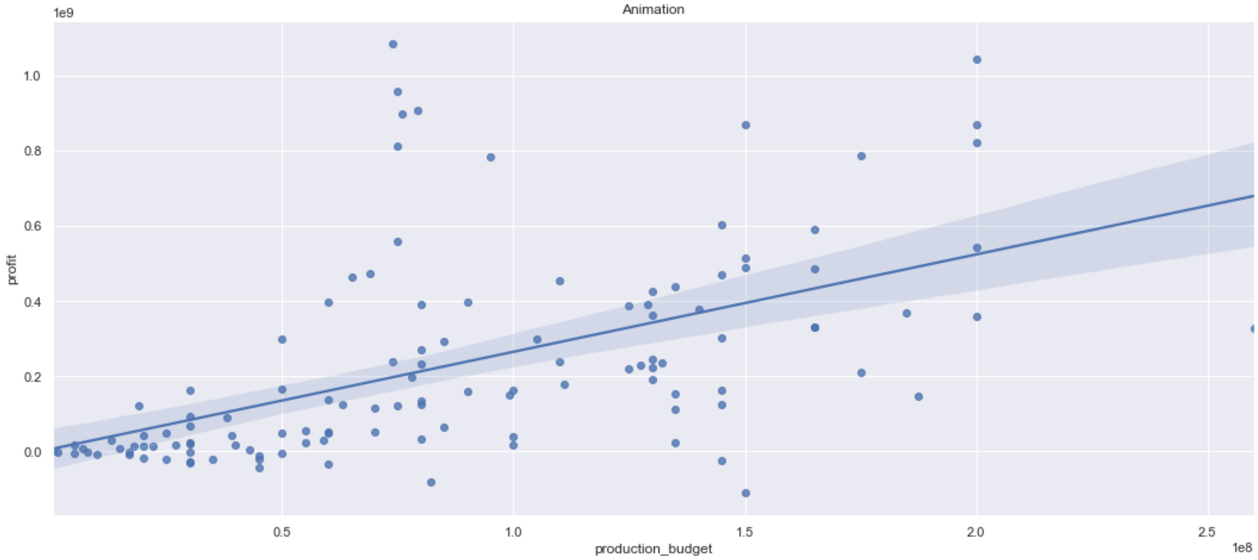
In [69]:

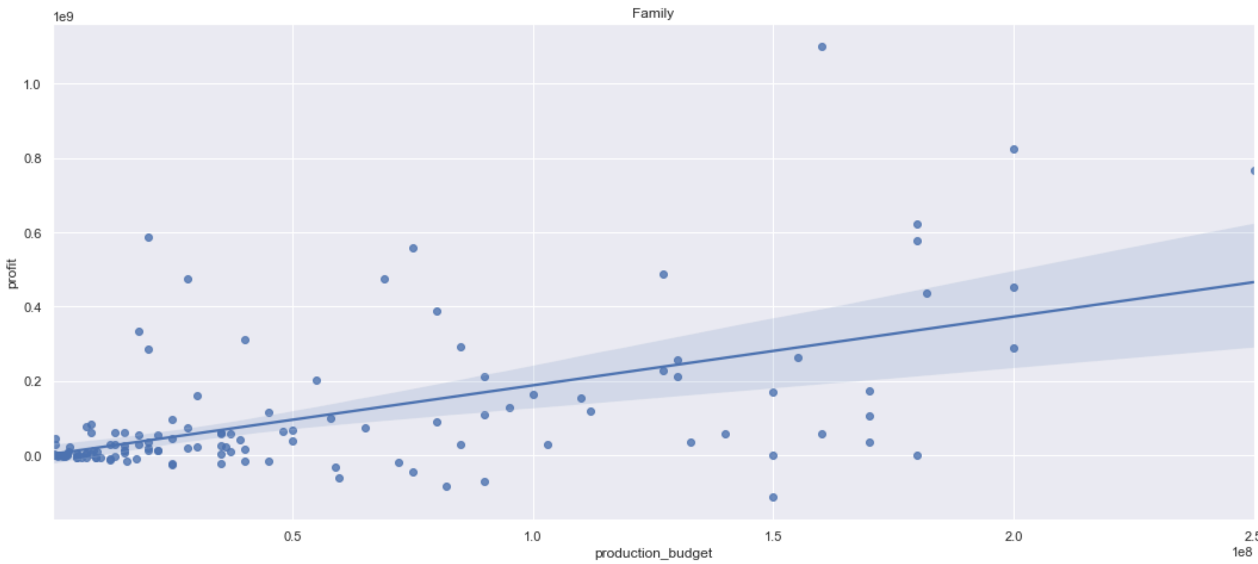
```
fig, ax = plt.subplots(figsize=(16, 8))

ax = genre_months.plot.bar(stacked=True,ax=ax,rot=360,)
ax.set(xlabel='Release Month',ylabel='Count',
       title='Number of Releases by Month for Top Genre')
ax.legend(bbox_to_anchor=[1,1]);
```

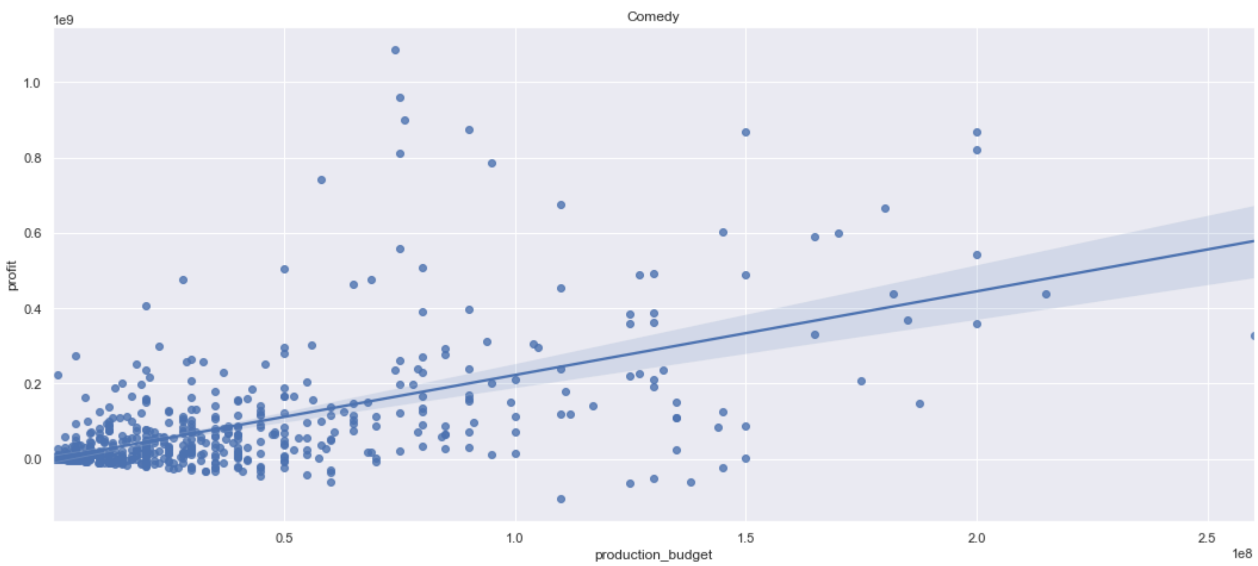


In [46]:

```
fig, ax = plt.subplots(figsize=(15, 20),nrows=3,ncols=1)
ax = ax.flatten()

for i, genre in enumerate(top_genres):
    plot_df = df_genres[df_genres['genres_list'] == genre]
    sns.regplot(data=plot_df, y='profit',x='production_budget',
                ax=ax[i]);
    ax[i].set_title(genre)
plt.tight_layout()
```







In [49]:

```
import plotly.express as px
df = df_genres
fig = px.scatter(top_genres_df,x='production_budget',y='profit',
                 height=700,width=1000,trendline='ols',
                 title='Profit Against Budget By Top Genres',
                 color='genres_list',hover_name='movie');
```

```
fig.show()
```

Q. Who are the key film crew members behind the top box office hits?

In [50]:
```
# Previewing the DF we prepared earlier to analyze financial performances by fil
df_directors
```

Out[50]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst

	release_date	movie	production_budget	domestic_gross	worldwide_gross	tconst
0	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650
1	2018-12-19	Mary Poppins Returns	130000000	171958438	341528518	tt5028340
2	2014-12-25	Into the Woods	56200000	128002372	213116401	tt2180411
3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	tt6565702
4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427
...	...	...	...	...	...	...
2517	1996-04-01	Bang	10000	527	527	tt6616538
2518	1996-04-01	Bang	10000	527	527	tt6616538
2519	2012-01-13	Newlyweds	9000	4584	4584	tt1880418
2520	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644
2521	2015-09-29	A Plague So Pleasant	1400	0	0	tt2107644

2522 rows × 23 columns

In [51]:

```
# Removing excess columns
df_directors.drop(['release_date','runtime_minutes',
                    'birth_year','death_year','directors',
                   'writers','director_list','primary_profession',
                    'known_for_titles'],axis=1,inplace=True)
df_directors
```

Out[51]:

	movie	production_budget	domestic_gross	worldwide_gross	tconst	
0	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650	Action,Advent
1	Mary Poppins Returns	130000000	171958438	341528518	tt5028340	Comedy,Fa

	movie	production_budget	domestic_gross	worldwide_gross	tconst	
2	Into the Woods	56200000	128002372	213116401	tt2180411	Adventure,Cor
3	Dark Phoenix	350000000	42762350	149762350	tt6565702	Action,Adve
4	Avengers: Age of Ultron	330600000	459005868	1403013963	tt2395427	Action,Adve
...	...	...	...	...	...	
2517	Bang	10000	527	527	tt6616538	
2518	Bang	10000	527	527	tt6616538	
2519	Newlyweds	9000	4584	4584	tt1880418	Cor
2520	A Plague So Pleasant	1400	0	0	tt2107644	Drama,H(
2521	A Plague So Pleasant	1400	0	0	tt2107644	Drama,H(

2522 rows × 14 columns

In [52]:
```
# Top Directors By Total Profit
dir_names = df_directors.groupby(['primary_name']).sum().round()
top_dirs_profit = dir_names.sort_values('profit',ascending=False).head(15)
top_dirs_profit
```

Out[52]:

primary_name	production_budget	domestic_gross	worldwide_gross	profit	ROI	release
Pierre Coffin	294000000	1220249440	3713745331	3419745331	4,618	
Joe Russo	720000000	1346646789	3902605502	3182605502	1,259	
Anthony Russo	720000000	1346646789	3902605502	3182605502	1,259	
Joss Whedon	615600000	1105670831	2969535276	2353935276	880	
Christopher Nolan	750000000	1118801468	3086180484	2336180484	1,254	
Michael Bay	648000000	777873593	2911998250	2263998250	1,292	
Chris Renaud	300000000	1051759355	2518783438	2218783438	2,995	
Peter Jackson	750000000	816490211	2922948044	2172948044	869	
Kyle Balda	149000000	600670070	2195063923	2046063923	2,748	
Francis Lawrence	522000000	1149112056	2543191543	2021191543	1,703	
Kevin Lincoln	200000000	659363944	2208208395	2008208395	1,004	

primary_name	production_budget	domestic_gross	worldwide_gross	profit	ROI	release
Pete Meads	200000000	659363944	2208208395	2008208395	1,004	
Bryan Singer	628000000	670854965	2383073266	1755073266	2,007	
Bill Condon	206000000	883325603	1873785010	1667785010	3,553	
Sam Mendes	500000000	504434452	1990147904	1490147904	648	

In [53]:

```
# Top Directors By Average ROI
dir_names = df_directors.groupby(['primary_name']).mean()
top_dirs_roi = dir_names.sort_values('ROI',ascending=False).head(15)
top_dirs_roi
```

Out[53]:

primary_name	production_budget	domestic_gross	worldwide_gross	profit	ROI	releas
Chris Lofing	100,000	22,764,410	41,656,474	41,556,474	41,556	
Travis Cluff	100,000	22,764,410	41,656,474	41,556,474	41,556	
Sujit Mondal	1,000,000	117,235,147	225,000,000	224,000,000	22,400	
Joaquin Perea	1,000,000	53,262,945	101,759,490	100,759,490	10,076	
Levan Gabriadze	1,000,000	32,789,645	64,364,198	63,364,198	6,336	
Brandon Camp	500,000	31,559,560	31,559,560	31,059,560	6,212	
Tod Williams	3,000,000	84,752,907	177,512,032	174,512,032	5,817	
Jamie Buckner	5,000,000	138,141,585	278,964,806	273,964,806	5,479	
Jordan Peele	5,000,000	176,040,665	255,367,951	250,367,951	5,007	
Chris Stokes	2,125,000	11,947,000	11,947,000	9,822,000	4,679	
Bradley Parker	1,000,000	18,119,640	42,411,721	41,411,721	4,141	
Chris Kaye	250,000	489,220	8,969,065	8,719,065	3,488	
Henry Joost	5,000,000	78,964,571	174,928,918	169,928,918	3,399	
Ariel Schulman	5,000,000	78,964,571	174,928,918	169,928,918	3,399	
David Gordon Green	17,290,625	40,183,370	59,655,460	42,364,834	3,009	

In [54]:

```
# Creating a heatmap of top directors to visualize ranking of attributes
s = top_dirs_profit.style.background_gradient();
s
```

Out[54]:

primary_name	production_budget	domestic_gross	worldwide_gross	profit	ROI

primary_name	production_budget	domestic_gross	worldwide_gross	profit	ROI
Pierre Coffin	294000000	1220249440	3713745331	3419745331	4618.000000
Joe Russo	720000000	1346646789	3902605502	3182605502	1259.000000
Anthony Russo	720000000	1346646789	3902605502	3182605502	1259.000000
Joss Whedon	615600000	1105670831	2969535276	2353935276	880.000000
Christopher Nolan	750000000	1118801468	3086180484	2336180484	1254.000000
Michael Bay	648000000	777873593	2911998250	2263998250	1292.000000
Chris Renaud	300000000	1051759355	2518783438	2218783438	2995.000000
Peter Jackson	750000000	816490211	2922948044	2172948044	869.000000
Kyle Balda	149000000	600670070	2195063923	2046063923	2748.000000
Francis Lawrence	522000000	1149112056	2543191543	2021191543	1703.000000
Kevin Lincoln	200000000	659363944	2208208395	2008208395	1004.000000
Pete Meads	200000000	659363944	2208208395	2008208395	1004.000000
Bryan Singer	628000000	670854965	2383073266	1755073266	2007.000000
Bill Condon	206000000	883325603	1873785010	1667785010	3553.000000
Sam Mendes	500000000	504434452	1990147904	1490147904	648.000000

In [55]:

```python
# Same as above but for ROI
s = top_dirs_roi.style.background_gradient();
s
```

Out[55]:

primary_name	production_budget	domestic_gross	worldwide_gross	profit	
Chris Lofing	100000.000000	22764410.000000	41656474.000000	41556474.000000	415
Travis Cluff	100000.000000	22764410.000000	41656474.000000	41556474.000000	415
Sujit Mondal	1000000.000000	117235147.000000	225000000.000000	224000000.000000	224
Joaquin Perea	1000000.000000	53262945.000000	101759490.000000	100759490.000000	10(
Levan Gabriadze	1000000.000000	32789645.000000	64364198.000000	63364198.000000	63
Brandon Camp	500000.000000	31559560.000000	31559560.000000	31059560.000000	6
Tod Williams	3000000.000000	84752907.000000	177512032.000000	174512032.000000	5
Jamie Buckner	5000000.000000	138141585.000000	278964806.000000	273964806.000000	54
Jordan Peele	5000000.000000	176040665.000000	255367951.000000	250367951.000000	5(
Chris Stokes	2125000.000000	11947000.000000	11947000.000000	9822000.000000	46

primary_name	production_budget	domestic_gross	worldwide_gross	profit	
Bradley Parker	1000000.000000	18119640.000000	42411721.000000	41411721.000000	4
Chris Kaye	250000.000000	489220.000000	8969065.000000	8719065.000000	34
Henry Joost	5000000.000000	78964571.000000	174928918.000000	169928918.000000	33
Ariel Schulman	5000000.000000	78964571.000000	174928918.000000	169928918.000000	33
David Gordon Green	17290625.000000	40183370.125000	59655459.500000	42364834.500000	3

In [56]:
```
# Breaking out genres from the Directors DF to sort filmmakers by genre
df_directors['genre_list'] = df_directors['genres'].str.split(pat=',')
```

In [57]:
```
df_genre_directors = df_directors.explode('genre_list')
df_genre_directors
```

Out[57]:

	movie	production_budget	domestic_gross	worldwide_gross	tconst	
0	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650	Action,Adventu
0	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650	Action,Adventu
0	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	tt1298650	Action,Adventu
1	Mary Poppins Returns	130000000	171958438	341528518	tt5028340	Comedy,Fam
1	Mary Poppins Returns	130000000	171958438	341528518	tt5028340	Comedy,Fam
...	...	...	...	...	...	
2520	A Plague So Pleasant	1400	0	0	tt2107644	Drama,Ho
2520	A Plague So Pleasant	1400	0	0	tt2107644	Drama,Ho

	movie	production_budget	domestic_gross	worldwide_gross	tconst	
2521	A Plague So Pleasant	1400	0	0	tt2107644	Drama,Hoı
2521	A Plague So Pleasant	1400	0	0	tt2107644	Drama,Hoı
2521	A Plague So Pleasant	1400	0	0	tt2107644	Drama,Hoı

5836 rows × 15 columns

In [58]:
```
# Creating a list of our strategic genres
top_genres_df['genres_list'].unique()
```

Out[58]:
```
array(['Animation', 'Comedy', 'Family'], dtype=object)
```

In [60]:
```
results = {}
for genre in top_genres_df['genres_list'].unique():
    group_df = df_genre_directors.groupby('genre_list').get_group(genre)
    directors = group_df.groupby('primary_name').mean()['profit'].sort_values(as

    results[genre] = directors.head(10).to_frame().reset_index()
results.keys()
```

Out[60]:
```
dict_keys(['Animation', 'Comedy', 'Family'])
```

In [61]:
```
# Creating a multi-indexed DF to display Top 10 Directors from our
# strategic genres and their respective average profits

results_df = pd.concat(results,axis=1)
results_df
```

Out[61]:

	Animation		Comedy		Family	
	primary_name	**profit**	**primary_name**	**profit**	**primary_name**	**profit**
0	Brad Bird	1,042,520,711	Kyle Balda	1,023,031,962	Bill Condon	843,815,419
1	Kyle Balda	1,023,031,962	Eric Guillon	959,727,750	Peter Jackson	767,003,568
2	Eric Guillon	959,727,750	Jared Bush	869,429,616	Robert Stromberg	578,536,735
3	Jon Favreau	906,914,868	Lee Unkrich	868,879,522	Christophe Lourdelet	559,454,789
4	Jared Bush	869,429,616	Pierre Coffin	854,936,333	Garth Jennings	559,454,789
5	Lee Unkrich	868,879,522	Andrew Stanton	821,215,193	David Yates	537,311,470
6	Pierre Coffin	854,936,333	Angus MacLane	821,215,193	Caleb Doyle	488,461,394
7	Andrew Stanton	821,215,193	Yarrow Cheney	811,750,534	Kyle Lawrence	488,461,394

	Animation		Comedy		Family	
	primary_name	profit	primary_name	profit	primary_name	profit
8	Angus MacLane	821,215,193	Tim Miller	743,025,593	Pierre Coffin	474,464,573
9	Yarrow Cheney	811,750,534	David Leitch	676,680,557	Chris Renaud	474,464,573

In [62]:
```
top_genres_profit_df['genres_list'].unique()
```

Out[62]:
```
array(['Action', 'Adventure', 'Fantasy', 'Sci-Fi'], dtype=object)
```

In [63]:
```
results2 = {}
for genre in top_genres_profit_df['genres_list'].unique():
    group_df = df_genre_directors.groupby('genre_list').get_group(genre)
    directors = group_df.groupby('primary_name').mean()['profit'].sort_values(as

    results2[genre] = directors.head(10).to_frame().reset_index()
results2.keys()
```

Out[63]:
```
dict_keys(['Action', 'Adventure', 'Fantasy', 'Sci-Fi'])
```

In [64]:
```
# Creating a similar mutli-index DF as above but for average ROI
results2_df = pd.concat(results2,axis=1)
results2_df
```

Out[64]:

	Action		Adventure		Fantasy		
	primary_name	profit	primary_name	profit	primary_name	profit	prima
0	Colin Trevorrow	1,433,854,864	Kevin Lincoln	2,008,208,395	James Wan	986,894,640	T
1	Ryan Coogler	1,148,258,224	Pete Meads	2,008,208,395	Bill Condon	843,815,419	Ryaı
2	J.A. Bayona	1,135,772,799	Colin Trevorrow	1,433,854,864	Peter Jackson	724,316,015	J.A
3	Anthony Russo	1,060,868,501	Ryan Coogler	1,148,258,224	Patty Jenkins	671,133,378	J
4	Joe Russo	1,060,868,501	J.A. Bayona	1,135,772,799	David Slade	638,102,828	Antho
5	James Wan	986,894,640	Adam Green	1,122,469,910	Joachim Rønning	558,241,137	R
6	Ryan Fleck	948,061,550	Joe Russo	1,060,868,501	Espen Sandberg	558,241,137	An
7	Anna Boden	948,061,550	Anthony Russo	1,060,868,501	David Yates	537,311,470	Joss
8	Jake Kasdan	874,496,193	Kyle Balda	1,023,031,962	Scott Derrickson	511,404,566	Mi
9	Joss Whedon	784,645,092	James Wan	986,894,640	Seth MacFarlane	506,016,627	J

In [65]:
```
top_genres_roi_df['genres_list'].unique()
```

Out[65]:
```
array(['Thriller', 'Horror', 'Romance', 'Mystery'], dtype=object)
```

In [66]:
```
results3 = {}
for genre in top_genres_roi_df['genres_list'].unique():
    group_df = df_genre_directors.groupby('genre_list').get_group(genre)
    directors = group_df.groupby('primary_name').mean()['ROI'].sort_values(ascen

    results3[genre] = directors.head(10).to_frame().reset_index()
results3.keys()
```

Out[66]: dict_keys(['Thriller', 'Horror', 'Romance', 'Mystery'])

In [67]:
```
# Creating a similar mutli-index DF as above but for average ROI
results3_df = pd.concat(results3,axis=1)
results3_df
```

Out[67]:

	Thriller		Horror		Romance		Mystery	
	primary_name	**ROI**	**primary_name**	**ROI**	**primary_name**	**ROI**	**primary_name**	**ROI**
0	Chris Lofing	41,556	Travis Cluff	41,556	Jamie Buckner	5,479	Travis Cluff	41,556
1	Travis Cluff	41,556	Chris Lofing	41,556	Josh Boone	2,460	Chris Lofing	41,556
2	Joaquin Perea	10,076	Joaquin Perea	10,076	Richard Dailey	1,874	Levan Gabriadze	6,336
3	Chris Stokes	9,458	David Gordon Green	8,101	Ryan Coogler	1,850	Jordan Peele	5,007
4	David Gordon Green	8,101	Levan Gabriadze	6,336	Drake Doremus	1,391	Bradley Parker	4,141
5	Levan Gabriadze	6,336	Tod Williams	5,817	Sam Taylor-Johnson	1,327	James Wan	4,024
6	Jordan Peele	5,007	Jordan Peele	5,007	David Lowery	1,296	John R. Leonetti	3,852
7	Bradley Parker	4,141	Bradley Parker	4,141	John Madden	1,246	Henry Joost	3,399
8	James Wan	4,024	James Wan	4,024	Troy Murray	1,146	Ariel Schulman	3,399
9	John R. Leonetti	3,852	Ariel Schulman	3,399	Justin Baldoni	1,050	Robert Heath	2,618

Conclusions

After understanding the business problem, the data required to answer those questions was selected and prepared for analysis. This was done by merging tables, dropping duplicated entries, altering object types and creating new columns.

After performing analysis the first recommendation for the business stakeholders are to plan releases for May-July and November. The second recommendation is to prioritize Animation, Comedy, and Family genres. And then plan big budget Adventure, Sci-Fi, Fantasy and Action projects and lastly supplement with small budget Mystery, Horror, Thriller and Romance that have potential to return great value. The third recommendation is to bring in proven filmmakers that have had success with projects in our targeted genres.

Of course this project is an exploratory data analysis and a much deeper review is required. There are a handful of outliers in the dataset that significantly skews the outcomes. It would be advisable to take a detailed look as to what separated those titles. Other important limitations of this analysis are Motion Picture Association ratings and actors. And perhaps the greatest limitation is the lack of marketing data. Since we discovered the strongest correlated attribute to gross profit is worldwide gross revenue. So looking into what promotes the greatest sales is certainly a worthwhile endeavor.