Merged Jupyter Notebook

from file: Aseasonal_profits_tables

Seasonal Profits Tables

Import programs so all functions and methods work

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Read in datadrame containing budget and print to see what the dataframe contains

In [2]:
```
budget = pd.read_csv('data/zippedData/tn.movie_budgets.csv.gz')
budget
```

Out[2]:

	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

Write a function to get all the months and create a column with their names for easier

access when making boxplot. Assign new seasons column as 'Seasons'

In [3]:
```
def getSeason(release_date):
    if (release_date[0:3] == "Dec") or  (release_date[0:3] == "Jan") or (release
        return "Winter"
    elif(release_date[0:3] == "Mar") or (release_date[0:3] == "Apr") or (release
        return "Spring"
    elif(release_date[0:3] == "Jun") or (release_date[0:3] == "Jul") or (release
        return "Summer"
    else:
        return "Fall"

season = budget.release_date.apply(getSeason)
budget['Season'] = season
budget
```

Out[3]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	Season
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279	Winter
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875	Spring
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	Summer
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	Spring
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747	Winter
...	...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0	Winter
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495	Spring
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338	Summer
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0	Fall
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041	Summer

5782 rows × 7 columns

Grab the last four digits of release_date column to retrieve the year. By grabing the last for indexes, the function is guaranteed to only grab the string of numbers pertaining to years

```
In [4]: budget["year"] = budget.release_date.apply(lambda x: x[-4:])
```

Make sure the years are cast as ints instead of strings for future callback.

```
In [5]: budget.year = budget.year.astype(int)
```

Obtain all years within budget df that is greater than or equal to 2015 because we want our data to include the most recent years

```
In [6]: budget = budget[budget.year>=2015]
```

Grab the first three digits in release_date column to obtain the month -- declare it as variable 'month'. This creates easier access for the months boxplot

```
In [7]: budget["month"] = budget.release_date.apply(lambda x: x[0:3])
        budget
```

Out[7]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	Season
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	Summer
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	Spring
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747	Winter
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,053,311,220	Winter
6	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200	Spring
...	...	...	...	...	...	...	...
5751	52	Dec 1, 2015	Dutch Kills	$25,000	$0	$0	Winter
5756	57	Apr 21, 2015	The Front Man	$20,000	$0	$0	Spring
5771	72	May 19, 2015	Family Motocross	$10,000	$0	$0	Spring
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0	Winter
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0	Fall

938 rows × 9 columns

Take out all commas and dollar signs so dataframe is more accessible

```
In [8]: budget['production_budget'] = budget['production_budget'].str.replace('$', '')
```

```
budget['domestic_gross'] = budget['domestic_gross'].str.replace('$', '')
budget['production_budget'] = budget['production_budget'].str.replace(',', '')
budget['domestic_gross'] = budget['domestic_gross'].str.replace(',', '')

budget
```

Out[8]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	Season
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	$149,762,350	Summer
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	$1,403,013,963	Spring
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	$1,316,721,747	Winter
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	$2,053,311,220	Winter
6	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	$2,048,134,200	Spring
...	...	...	...	...	...	...	...
5751	52	Dec 1, 2015	Dutch Kills	25000	0	$0	Winter
5756	57	Apr 21, 2015	The Front Man	20000	0	$0	Spring
5771	72	May 19, 2015	Family Motocross	10000	0	$0	Spring
5777	78	Dec 31, 2018	Red 11	7000	0	$0	Winter
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	$0	Fall

938 rows × 9 columns

Make domestic_gross and production_budget numerical & drop 'worldwide_gross' column(not needed for our research). Print to ensure everything worked

In [9]:
```
budget[['domestic_gross', 'production_budget']] = budget[['domestic_gross', 'pro
budget = budget.drop(columns="worldwide_gross")
budget
```

Out[9]:

	id	release_date	movie	production_budget	domestic_gross	Season	year	month
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	Summer	2019	Jun
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	Spring	2015	May

	id	release_date	movie	production_budget	domestic_gross	Season	year	month
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	Winter	2017	Dec
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	Winter	2015	Dec
6	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	Spring	2018	Apr
...	...	...	...	...	...	...	...	...
5751	52	Dec 1, 2015	Dutch Kills	25000	0	Winter	2015	Dec
5756	57	Apr 21, 2015	The Front Man	20000	0	Spring	2015	Apr
5771	72	May 19, 2015	Family Motocross	10000	0	Spring	2015	May
5777	78	Dec 31, 2018	Red 11	7000	0	Winter	2018	Dec
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	Fall	2015	Sep

938 rows × 8 columns

Create new column 'profit' which is 'domestic_gross' minus 'production_budget

In [10]:
```python
budget["profit"] = budget["domestic_gross"] - budget["production_budget"]
budget
```

Out[10]:

	id	release_date	movie	production_budget	domestic_gross	Season	year	month	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	Summer	2019	Jun	-3
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	Spring	2015	May	1
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	Winter	2017	Dec	3
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	Winter	2015	Dec	6
6	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	Spring	2018	Apr	3
...	...	...	...	...	...	...	...	...	

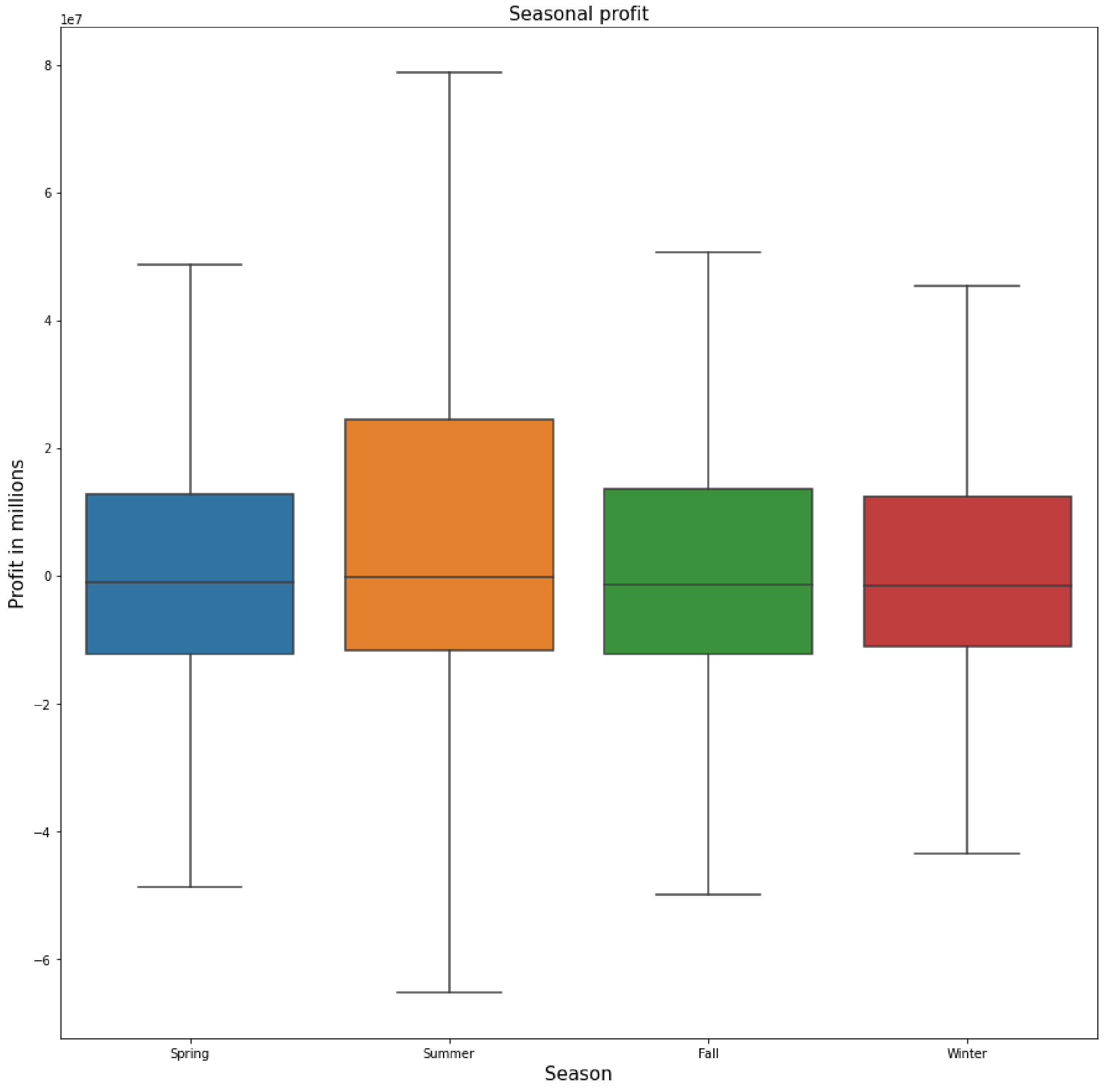
	id	release_date	movie	production_budget	domestic_gross	Season	year	month
5751	52	Dec 1, 2015	Dutch Kills	25000	0	Winter	2015	Dec
5756	57	Apr 21, 2015	The Front Man	20000	0	Spring	2015	Apr
5771	72	May 19, 2015	Family Motocross	10000	0	Spring	2015	May
5777	78	Dec 31, 2018	Red 11	7000	0	Winter	2018	Dec
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	Fall	2015	Sep

938 rows × 9 columns

Make a boxplot with the information above. Create a list with the seasons in order that you want them to be on the xlabels.

```
In [11]: order_list = ['Spring', 'Summer', 'Fall', 'Winter']
```

```
In [12]: # Your code here
         plot_five_fig, plot_five_ax = plt.subplots(figsize=(15,15))

         sns.boxplot(x='Season', y='profit', data=budget, ax=plot_five_ax, showfliers=Fal

         plot_five_ax.set_xlabel('Season', fontsize = 15)

         plot_five_ax.set_ylabel('Profit in millions', fontsize = 15)

         plot_five_ax.set_title('Seasonal profit', fontsize = 15)


         plt.savefig('images/seasonalprofits.png')
```



In [13]:
```
#Summer has the largest range which indicates a high risk of whether or not the
```

Find the mean for profit, budget, and gross

In [14]:
```
profit_mean = budget.profit.mean()
profit_mean
```

Out[14]:
```
9372304.52238806
```

There is a mean profit of ~ 10 million dollars. To be successful, your movies should have a profit around that number

In [15]:
```
budget_mean = budget.production_budget.mean()
budget_mean
```

Out[15]:
```
39360287.20682303
```

The mean budget is ~ 31 million dollars. Your movie should average a budget around that number to ensure a promising profit.

```
In [16]: gross_mean = budget.domestic_gross.mean()
         gross_mean
```

```
Out[16]: 48732591.729211085
```

The average gross for american movies is around 42 million. You should be projecting your sales towards that number to be successful.

Find the mean for each column in years 2015-2019. Exclude 2020 because of the skewed data.

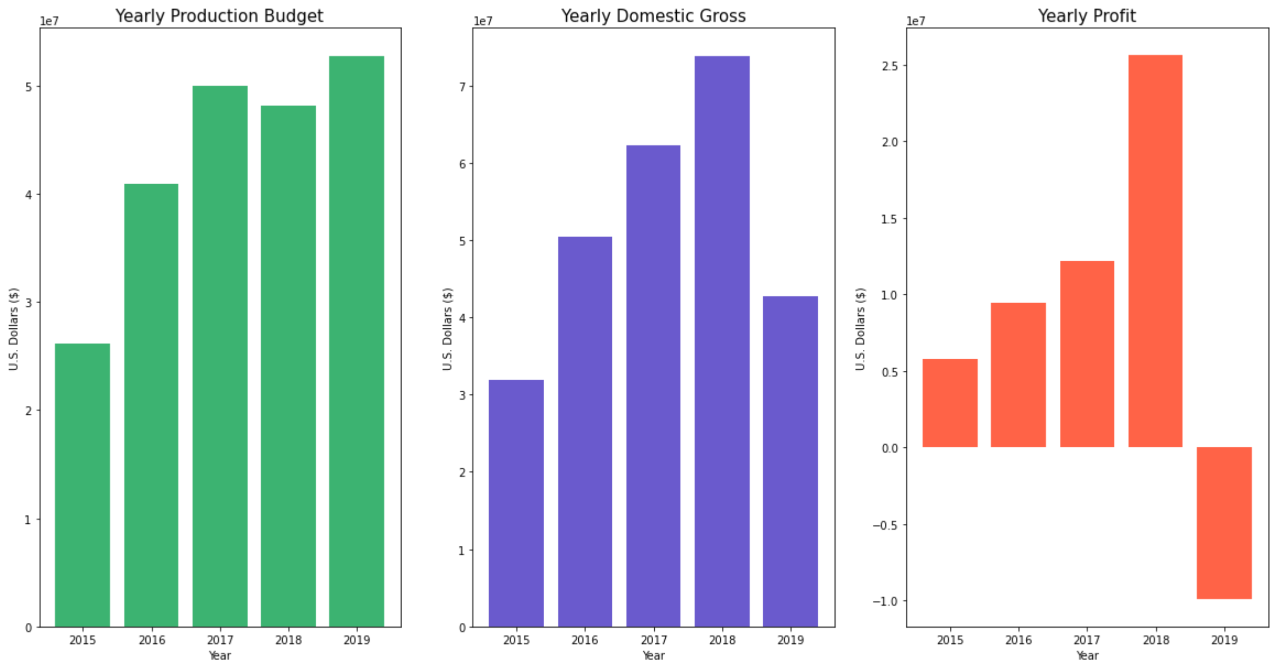
```
In [17]: budget_groups = budget.groupby('year').mean().reset_index().astype(float)
         budget_groups2 = budget_groups[budget_groups.year<2020]

         budget_groups2
```

Out[17]:

	year	id	production_budget	domestic_gross	profit
0	2015.0	51.260355	2.616029e+07	3.193948e+07	5.779185e+06
1	2016.0	49.643836	4.097370e+07	5.042387e+07	9.450175e+06
2	2017.0	51.422619	5.003073e+07	6.222259e+07	1.219186e+07
3	2018.0	53.286713	4.813886e+07	7.378870e+07	2.564984e+07
4	2019.0	51.791045	5.273896e+07	4.280029e+07	-9.938666e+06

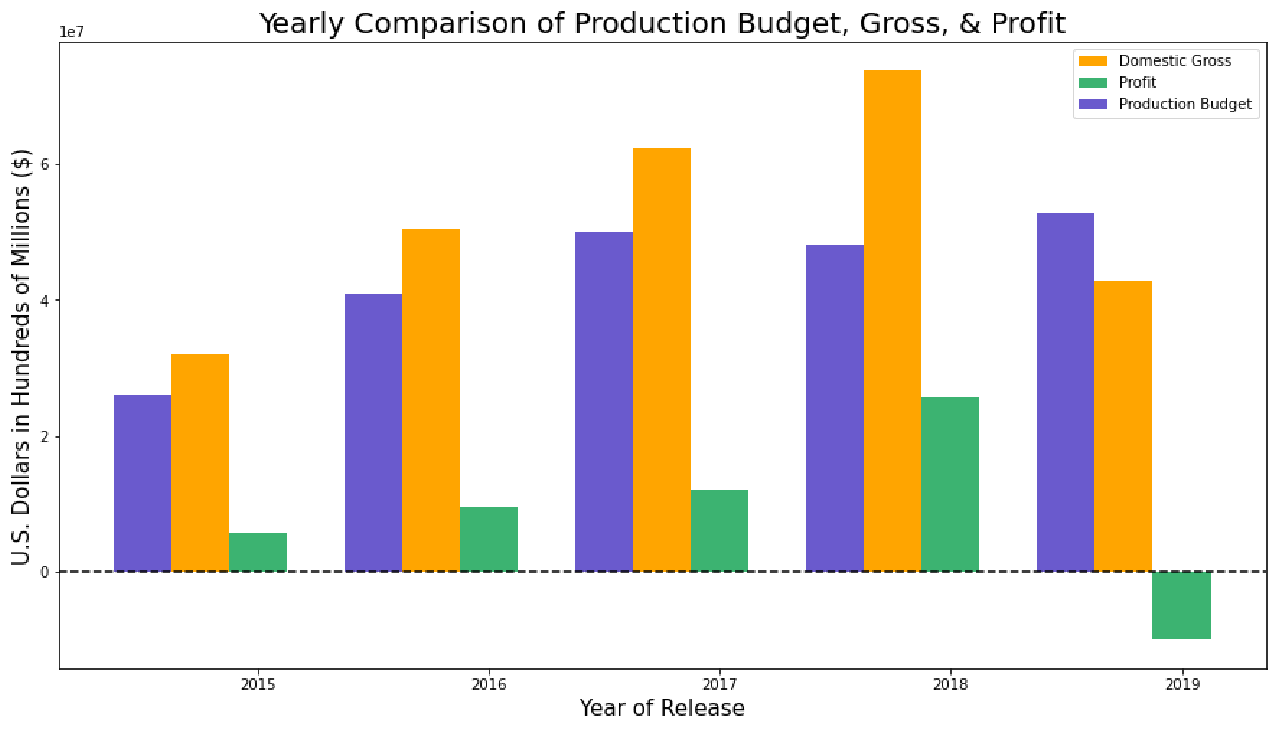
Create bar plots for yearly budget, domestic gross, and profit to examine. Use info to create a combined barplot.

```
In [18]: fig, ax = plt.subplots(figsize=(20,10), ncols=3)

         ax[0].bar(budget_groups2.year, budget_groups2.production_budget, color='mediumse
         ax[0].set_xlabel('Year')
         ax[0].set_ylabel('U.S. Dollars ($)')
         ax[0].set_title('Yearly Production Budget', fontsize=15)

         ax[1].bar(budget_groups2.year, budget_groups2.domestic_gross, color='slateblue')
         ax[1].set_xlabel('Year')
         ax[1].set_ylabel('U.S. Dollars ($)')
         ax[1].set_title('Yearly Domestic Gross', fontsize=15)


         ax[2].bar(budget_groups2.year, budget_groups2.profit, color='tomato')
         ax[2].set_xlabel('Year')
         ax[2].set_ylabel('U.S. Dollars ($)')
         ax[2].set_title('Yearly Profit', fontsize=15);

         plt.savefig('images/gross.png')
```



Combine three graphs from above for reassurance that the more budget put in, the more profit is produced (excluding 2019, but need for further explanation)

In [19]:
```
fig, ax = plt.subplots(figsize=(15,8))
N = 5
ind = np.arange(N)
width = 0.25

xvals = budget_groups2['domestic_gross']
bar1 = plt.bar(ind, xvals, width, color = 'orange')

yvals2 = budget_groups2['profit']
bar2 = plt.bar(ind+width, yvals2, width, color = 'mediumseagreen')

zvals3 = budget_groups2['production_budget']
bar3 = plt.bar(ind-width, zvals3, width, color = 'slateblue')


plt.xlabel("Year of Release", fontsize=15)
plt.ylabel('U.S. Dollars in Hundreds of Millions ($)', fontsize=15)
plt.title("Yearly Comparison of Production Budget, Gross, & Profit", fontsize=20

plt.axhline(y=0.5, color='black', linestyle='--')


plt.xticks(ind+width, ['2015','2016','2017','2018','2019'])
plt.legend( (bar1, bar2, bar3), ('Domestic Gross', 'Profit', 'Production Budget'

plt.show()
plt.savefig('images/Yearly_comparison_profit.png')
```



```
<Figure size 432x288 with 0 Axes>
```

Create a boxplot for monthly profits to help visualize the difference between each month. Explore their ranges, max, and mins.

```
In [20]: budget
```

Out[20]:

	id	release_date	movie	production_budget	domestic_gross	Season	year	month	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	Summer	2019	Jun	-3
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	Spring	2015	May	1
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	Winter	2017	Dec	3
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	Winter	2015	Dec	6
6	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	Spring	2018	Apr	3
...	...	...	...	...	...	...	...	...	
5751	52	Dec 1, 2015	Dutch Kills	25000	0	Winter	2015	Dec	
5756	57	Apr 21, 2015	The Front Man	20000	0	Spring	2015	Apr	
5771	72	May 19, 2015	Family Motocross	10000	0	Spring	2015	May	

	id	release_date	movie	production_budget	domestic_gross	Season	year	month
5777	78	Dec 31, 2018	Red 11	7000	0	Winter	2018	Dec
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	Fall	2015	Sep

938 rows × 9 columns

Make the months into ints which will help with xticklabels. Setting them to ints will make it easier to sort them and make sure they are in numerical order on the x axis.

```
In [21]: month_map = {'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'Jun': 6,'Jul': 7
```

Apply month_map to months ; switches to numbers ; variable name = 'month_num'

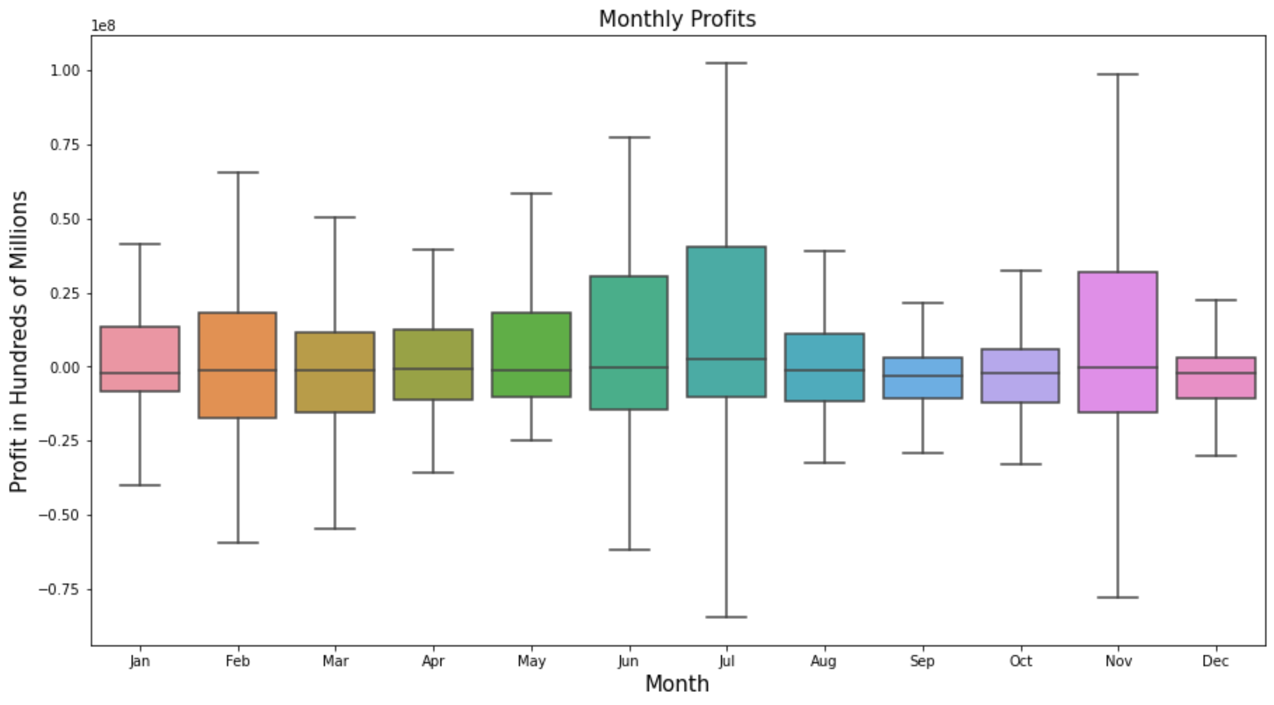
```
In [22]: budget['month_num'] = budget['month'].apply(lambda x: month_map.get(x))
```

Sort month_num values so they are in ascending order

```
In [23]: sorted_budget = budget.sort_values('month_num')
```

Make a box plot of all the months and their profits

```
In [24]: plot_one_fig, plot_one_ax = plt.subplots(figsize=(15,8))

         sns.boxplot(x='month', y='profit', data=sorted_budget, ax=plot_one_ax, showflier
         plot_one_ax.set_xlabel('Month', fontsize=15)
         plot_one_ax.set_ylabel('Profit in Hundreds of Millions', fontsize=15)
         plot_one_ax.set_title('Monthly Profits', fontsize=15);
         plt.savefig('images/MonthlyProfitsReal.png')
```



from file: BAudience Data.ipyn

Importing the different programs used

In [25]:
```
import numpy as np
```

In [26]:
```
import pandas as pd
```

In [27]:
```
import seaborn as sns
```

In [28]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Imorting the Rotten Tomatoes data set

In [29]:
```
df = pd.read_csv('data/zippedData/rotten_tomatoes_movies.csv.gz')
df.head()
```

Out[29]:

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	gen
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	PG	Actio Adventu Come Dra Scie F
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	R	Com
2	m/10	10	A successful, middle-aged Hollywood songwriter...	Blake Edwards' bawdy comedy may not score a pe...	R	Come Roma
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	Following the closing arguments in a murder tr...	Sidney Lumet's feature debut is a superbly wri...	NR	Class Dra
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	In 1866, Professor Pierre M. Aronnax (Paul Luk...	One of Disney's finest live-action adventures,...	G	Actio Adventu Dra Kid Far

5 rows × 22 columns

Filtering the relevent columns from the Rotten Tomates data set

In [30]:
```
df = df[['movie_title', 'content_rating', 'audience_count']]
```

```
df
```

Out[30]:

	movie_title	content_rating	audience_count
0	Percy Jackson & the Olympians: The Lightning T...	PG	254421.0
1	Please Give	R	11574.0
2	10	R	14684.0
3	12 Angry Men (Twelve Angry Men)	NR	105386.0
4	20,000 Leagues Under The Sea	G	68918.0
...	...	...	...
17707	Zoot Suit	R	1195.0
17708	Zootopia	PG	101511.0
17709	Zorba the Greek	NR	7146.0
17710	Zulu	PG	30193.0
17711	Zulu Dawn	PG	4469.0

17712 rows × 3 columns

Sorting the movies by audience_count by greatest to least from the Rotten Tomatoes data set

In [31]:
```
df = df.sort_values('audience_count', ascending=False)
df
```

Out[31]:

	movie_title	content_rating	audience_count
16297	Titanic	PG-13	35797635.0
15410	The Lord of the Rings: The Return of the King	PG-13	34679773.0
13694	Spider-Man	PG-13	34297354.0
13276	Shrek 2	PG	34232524.0
7526	Harry Potter and the Goblet of Fire	PG-13	34153607.0
...	...	...	...
17522	Working Girls	NR	NaN
17607	Yom Yom	NR	NaN
17609	Yosemite	R	NaN
17618	You Don't Need Feet to Dance	NR	NaN
17671	Z	NR	NaN

17712 rows × 3 columns

Deleting null values from audience_count values

In [32]:
```
df = df[df['audience_count'].notna()]
df
#audience_count column ordered from greatest to least, and NaN values dropped
```

Out[32]:

	movie_title	content_rating	audience_count
16297	Titanic	PG-13	35797635.0
15410	The Lord of the Rings: The Return of the King	PG-13	34679773.0
13694	Spider-Man	PG-13	34297354.0
13276	Shrek 2	PG	34232524.0
7526	Harry Potter and the Goblet of Fire	PG-13	34153607.0
...	...	...	...
6157	Evil Little Things	NR	5.0
1951	2 in the Bush: A Love Story	NR	5.0
10243	Measure for Measure	NR	5.0
7354	Guest Artist	NR	5.0
17119	The Wedding Banquet (Xi yan)	R	5.0

17415 rows × 3 columns

Finding the mean for audience_count

In [33]:
```
df['audience_count'].mean()
```

Out[33]:
```
143940.06833189778
```

Amount of movies in each content_rating

In [34]:
```
df.content_rating.value_counts()
```

Out[34]:
```
R         6348
NR        5220
PG-13     2970
PG        2163
G          676
NC17        38
Name: content_rating, dtype: int64
```

DataFrame with movies that have a value of R in content_rating

In [35]:
```
df_R = df.loc[df['content_rating'] == 'R']
df_R
```

Out[35]:

	movie_title	content_rating	audience_count
7052	Gladiator	R	34128168.0
2662	American Pie	R	33781574.0
10183	The Matrix	R	33324202.0
16106	There's Something About Mary	R	33121539.0
17120	Wedding Crashers	R	32961772.0
...	...	...	...
7549	Haunt	R	8.0

	movie_title	content_rating	audience_count
4794	Come to Daddy	R	8.0
14807	The Death of Dick Long	R	7.0
7376	Guns Akimbo	R	5.0
17119	The Wedding Banquet (Xi yan)	R	5.0

6348 rows × 3 columns

Mean audience_count for movies with a content_rating of R

```
In [36]: df_R.mean()
```

```
Out[36]: audience_count    117463.646345
         dtype: float64
```

DataFrame with movies that have a value of PG-13 in content_rating

```
In [37]: df_PG13 = df.loc[df['content_rating'] == 'PG-13']
         df_PG13
```

Out[37]:

	movie_title	content_rating	audience_count
16297	Titanic	PG-13	35797635.0
15410	The Lord of the Rings: The Return of the King	PG-13	34679773.0
13694	Spider-Man	PG-13	34297354.0
7526	Harry Potter and the Goblet of Fire	PG-13	34153607.0
9014	King Kong	PG-13	33766734.0
...	...	...	...
14937	The Fight	PG-13	27.0
14352	Tesla	PG-13	14.0
12364	Red Penguins	PG-13	7.0
13943	Strange But True	PG-13	6.0
11425	Ophelia	PG-13	5.0

2970 rows × 3 columns

Mean audience_count for movies with a content_rating of Pg-13

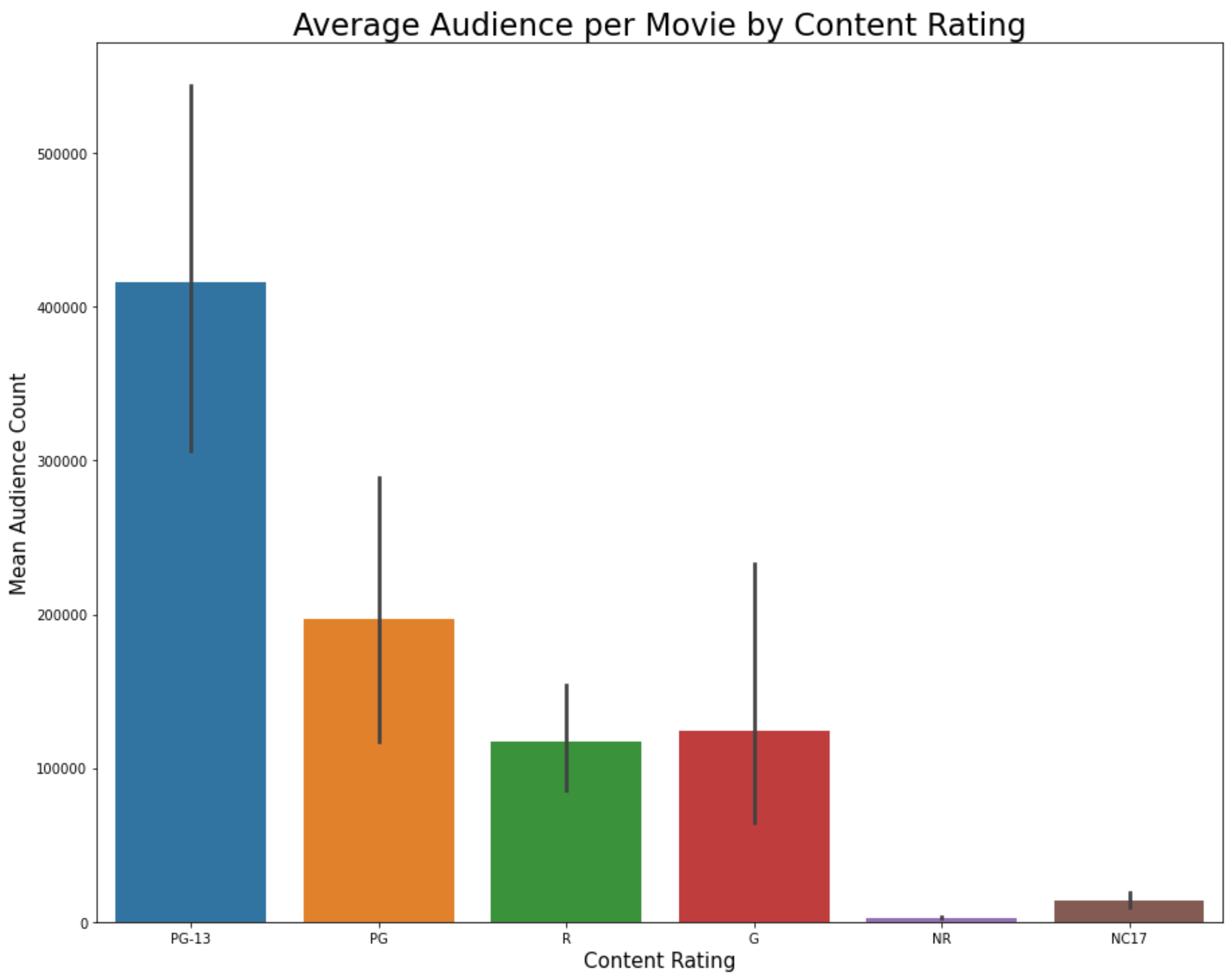
```
In [38]: df_PG13.mean()
```

```
Out[38]: audience_count    415981.044781
         dtype: float64
```

Seaborn barplot for Average Audience per Movie by Content rating

```
In [39]: plot_fig, plot_ax = plt.subplots(figsize=(15, 12))

         sns.barplot(x="content_rating", y="audience_count", data=df)
         plot_ax.set_xlabel('Content Rating', fontsize=15)
         plot_ax.set_ylabel('Mean Audience Count', fontsize=15);
         plot_ax.set_title('Average Audience per Movie by Content Rating', fontsize=23)
```

Out[39]:
```
Text(0.5, 1.0, 'Average Audience per Movie by Content Rating')
```



In [ ]:

---

from file: CMovie_analysis_genres_ross

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

Reads in the Rotten Tomatoes csv data and prints the first few rows

In [41]:
```
df = pd.read_csv('data/zippedData/rotten_tomatoes_movies.csv.gz')
df.head(2)
```

Out[41]:

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	genres	di

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	genres	di
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Cc
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	R	Comedy	Hol

2 rows × 22 columns

Reads in the movie gross csv data and prints the first few rows

In [42]:
```
df_gross = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')
df_gross
```

Out[42]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

Merges the two DataFrames together on the movie title and creates a DataFrame with only the data we want to work with

In [43]:
```
merged = pd.merge(df, df_gross, left_on='movie_title', right_on='title')
df_rating_genres = merged[['movie_title','genres', 'tomatometer_rating', 'domest
```

In [44]:
```
df_rating_genres
```

Out[44]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0

	movie_title	genres	tomatometer_rating	domestic_gross
1	Going the Distance	Comedy	0.0	17800000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
3	The Silence	Action & Adventure, Drama, Mystery & Suspense,...	50.0	100000.0
4	The Silence	Art House & International, Drama, Mystery & Su...	88.0	100000.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	92.0	3100000.0
2145	Zombeavers	Action & Adventure, Comedy, Horror	69.0	14900.0
2146	Zookeeper	Comedy, Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure, Animation, Comedy	98.0	341300000.0

2149 rows × 4 columns

Checks .info() to see how many null values are in the DataFrame rows

In [45]:
```
df_rating_genres.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2149 entries, 0 to 2148
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   movie_title         2149 non-null   object 
 1   genres              2149 non-null   object 
 2   tomatometer_rating  2148 non-null   float64
 3   domestic_gross      2137 non-null   float64
dtypes: float64(2), object(2)
memory usage: 83.9+ KB
```

Removes any rows with null values

In [46]:
```
df_rating_genres.dropna(inplace=True)
df_rating_genres.head(12)
```

Out[46]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
1	Going the Distance	Comedy	0.0	17800000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
3	The Silence	Action & Adventure, Drama, Mystery & Suspense,...	50.0	100000.0

	movie_title	genres	tomatometer_rating	domestic_gross
4	The Silence	Art House & International, Drama, Mystery & Su...	88.0	100000.0
5	The Silence	Horror, Mystery & Suspense	30.0	100000.0
6	Gone	Horror, Mystery & Suspense	54.0	11700000.0
7	Gone	Mystery & Suspense	12.0	11700000.0
8	Fireflies in the Garden	Drama	22.0	70600.0
9	Priest	Action & Adventure, Horror, Mystery & Suspense...	15.0	29100000.0
10	Red	Drama, Mystery & Suspense	70.0	90400000.0
11	Red	Action & Adventure, Comedy, Mystery & Suspense	72.0	90400000.0

Drops the duplicate values and keeps the last instance

```
In [47]: df_rating_genres.drop_duplicates(subset='movie_title', keep='last' , inplace=Tru
         df_rating_genres
```

Out[47]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
5	The Silence	Horror, Mystery & Suspense	30.0	100000.0
7	Gone	Mystery & Suspense	12.0	11700000.0
8	Fireflies in the Garden	Drama	22.0	70600.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	92.0	3100000.0
2145	Zombeavers	Action & Adventure, Comedy, Horror	69.0	14900.0
2146	Zookeeper	Comedy, Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure, Animation, Comedy	98.0	341300000.0

2063 rows × 4 columns

Checks for any more missing values and how many rows were removed

```
In [48]: df_rating_genres.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2063 entries, 0 to 2148
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype
 ---  ------              --------------  ------
 0    movie_title         2063 non-null   object
 1    genres              2063 non-null   object
 2    tomatometer_rating  2063 non-null   float64
 3    domestic_gross      2063 non-null   float64
dtypes: float64(2), object(2)
memory usage: 80.6+ KB
```

Looks at the highest and lowest gross to get a feel for which genres do best

In [49]:
```
df_sorted = df_rating_genres.sort_values('domestic_gross', ascending=False)
df_sorted
```

Out[49]:

	movie_title	genres	tomatometer_rating	domestic_gross
343	Black Panther	Action & Adventure, Drama, Science Fiction & F...	96.0	700100000.0
283	Avengers: Infinity War	Action & Adventure, Science Fiction & Fantasy	85.0	678800000.0
923	Jurassic World	Action & Adventure, Mystery & Suspense, Scienc...	70.0	652300000.0
1096	Marvel's The Avengers	Action & Adventure, Science Fiction & Fantasy	92.0	623400000.0
1496	Star Wars: The Last Jedi	Action & Adventure, Drama, Science Fiction & F...	90.0	620200000.0
...	...	...	...	...
267	Jackpot	Action & Adventure	59.0	800.0
238	Amityville: The Awakening	Horror, Mystery & Suspense	30.0	700.0
863	Into the White	Action & Adventure, Art House & International,...	45.0	700.0
120	2:22	Drama, Mystery & Suspense	22.0	400.0
1507	Storage 24	Horror, Science Fiction & Fantasy	42.0	100.0

2063 rows × 4 columns

Splits the genres in the genres row into a list containing the genres

In [50]:
```
df_rating_genres['genres'] = df_rating_genres['genres'].str.split(',', expand =

df_rating_genres
```

Out[50]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	[Comedy]	87.0	4000000.0
2	Going the Distance	[Comedy, Romance]	54.0	17800000.0
5	The Silence	[Horror, Mystery & Suspense]	30.0	100000.0
7	Gone	[Mystery & Suspense]	12.0	11700000.0
8	Fireflies in the Garden	[Drama]	22.0	70600.0
...	...	...	...	...

	movie_title	genres	tomatometer_rating	domestic_gross
2144	Zindagi Na Milegi Dobara	[Art House & International, Comedy, Drama]	92.0	3100000.0
2145	Zombeavers	[Action & Adventure, Comedy, Horror]	69.0	14900.0
2146	Zookeeper	[Comedy, Romance]	14.0	80400000.0
2147	Zoolander 2	[Comedy]	22.0	28800000.0
2148	Zootopia	[Action & Adventure, Animation, Comedy]	98.0	341300000.0

2063 rows × 4 columns

Explodes the genres column which separates all the genres for each movie into its own column

In [51]:
```
df_rating_genres = df_rating_genres.explode('genres')
df_rating_genres
```

Out[51]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
2	Going the Distance	Comedy	54.0	17800000.0
2	Going the Distance	Romance	54.0	17800000.0
5	The Silence	Horror	30.0	100000.0
5	The Silence	Mystery & Suspense	30.0	100000.0
...	...	...	...	...
2146	Zookeeper	Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure	98.0	341300000.0
2148	Zootopia	Animation	98.0	341300000.0
2148	Zootopia	Comedy	98.0	341300000.0

4118 rows × 4 columns

Checks the data types of the columns

In [52]:
```
df_rating_genres.dtypes
```

Out[52]:
```
movie_title              object
genres                   object
tomatometer_rating      float64
domestic_gross          float64
dtype: object
```

Strips any whitespace from the genres in the genres column so that we can easily pull the genres

In [53]:
```
df_rating_genres['genres'] = df_rating_genres['genres'].str.strip()
```

I only want to focus on genres with enough data points to generate sufficient means so I will only look at genres above 150 data values

In [54]:
```
df_rating_genres['genres'].value_counts()
```

Out[54]:
```
Drama                          1145
Comedy                          632
Action & Adventure              496
Mystery & Suspense              403
Art House & International       249
Science Fiction & Fantasy       244
Romance                         175
Horror                          171
Documentary                     140
Kids & Family                   126
Animation                       116
Special Interest                 89
Musical & Performing Arts        61
Sports & Fitness                 21
Western                          19
Classics                         12
Television                       10
Faith & Spirituality              6
Anime & Manga                     1
Cult Movies                       1
Gay & Lesbian                     1
Name: genres, dtype: int64
```

Creates a new DataFrame of all the movies classified as the specified genre

In [55]:
```
df_comedy = df_rating_genres.loc[df_rating_genres['genres'] == 'Comedy']
df_romance = df_rating_genres.loc[df_rating_genres['genres'] == 'Romance']
df_horror = df_rating_genres.loc[df_rating_genres['genres'] == 'Horror']
df_mystery_suspence = df_rating_genres.loc[df_rating_genres['genres'] == 'Myster
df_action_adventure = df_rating_genres.loc[df_rating_genres['genres'] == 'Action
df_animation = df_rating_genres.loc[df_rating_genres['genres'] == 'Animation']
df_art = df_rating_genres.loc[df_rating_genres['genres'] == 'Art House & Interna
df_drama = df_rating_genres.loc[df_rating_genres['genres'] == 'Drama']
```

Finds the mean of the domestic gross profit for all the relevent DataFrames

In [56]:
```
#mean of the comedy gross profit
y_comedy = round(df_comedy['domestic_gross'].mean())
y_drama = round(df_drama['domestic_gross'].mean())
y_romance = round(df_romance['domestic_gross'].mean())
y_horror = round(df_horror['domestic_gross'].mean())
y_mystery = round(df_mystery_suspence['domestic_gross'].mean())
y_action_adventure = round(df_action_adventure['domestic_gross'].mean())
y_animation = round(df_animation['domestic_gross'].mean())
y_art = round(df_art['domestic_gross'].mean())
```

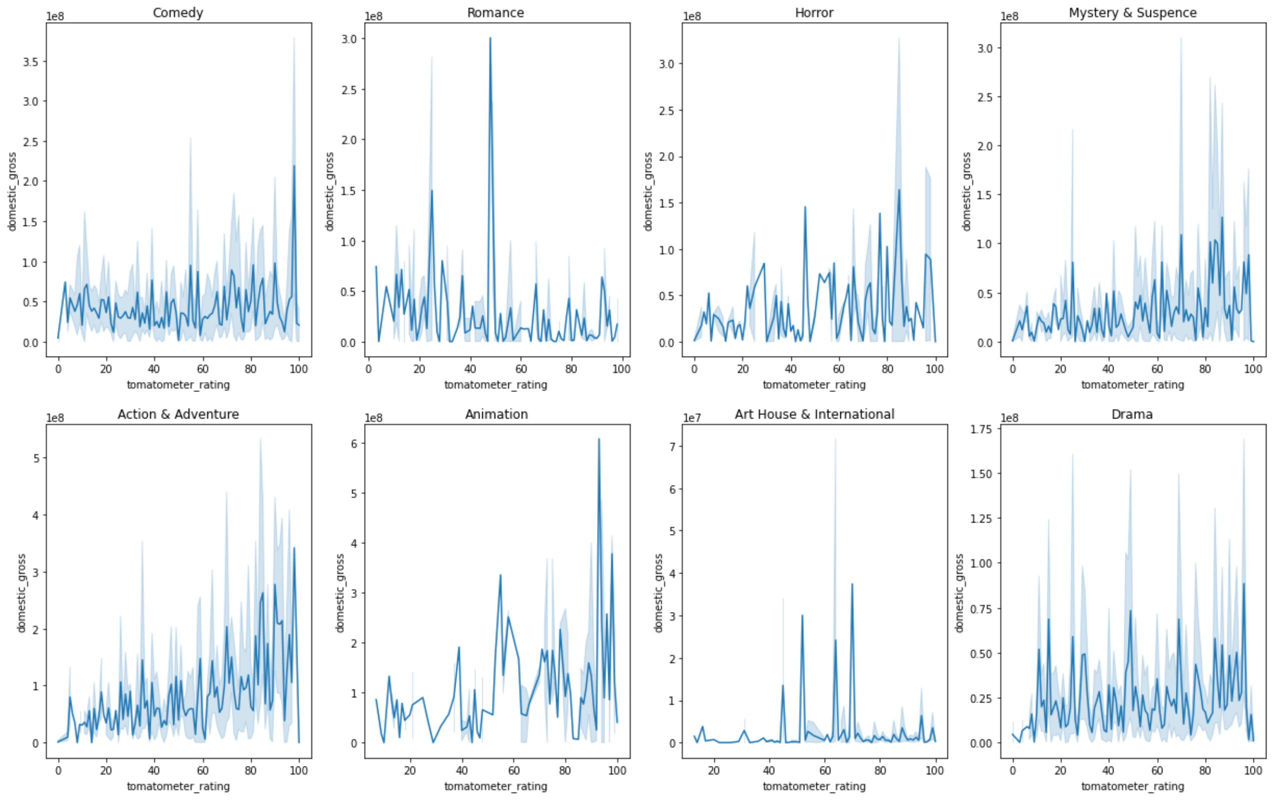
Finds the mean of the tomatometer rating for all the relevent genre DataFrames

In [57]:
```
#mean of the comedy genre tomato ratings
x_comedy = round(df_comedy['tomatometer_rating'].mean())
x_romance = round(df_romance['tomatometer_rating'].mean())
x_horror = round(df_horror['tomatometer_rating'].mean())
x_mystery = round(df_mystery_suspence['tomatometer_rating'].mean())
x_action_adventure = round(df_action_adventure['tomatometer_rating'].mean())
x_animation = round(df_animation['tomatometer_rating'].mean())
```

```
x_art = round(df_art['tomatometer_rating'].mean())
x_drama = round(df_drama['tomatometer_rating'].mean())
```

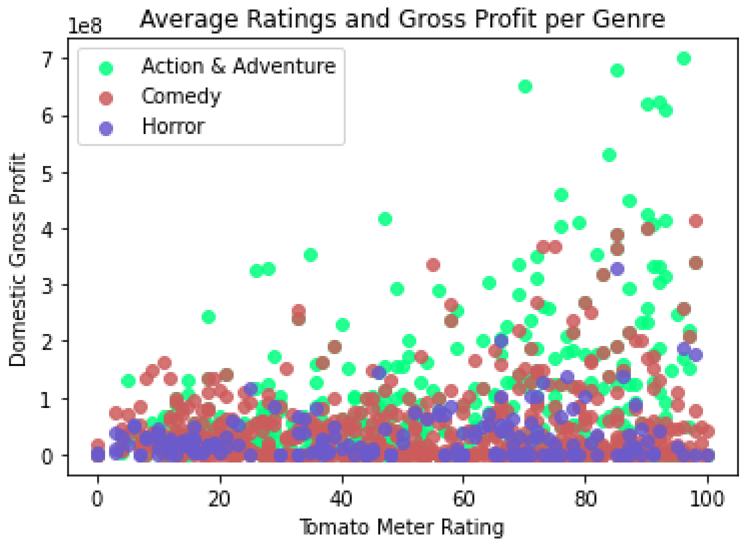
Graphs all the genre DataFrames to get an idea of which genre has the highest ratings and brought the most profit

In [58]:
```
fig, axes =plt.subplots(nrows=2,ncols=4, figsize =(21,13))
sns.lineplot(data=df_comedy, x="tomatometer_rating", y="domestic_gross", ax=axes
sns.lineplot(data=df_romance, x="tomatometer_rating", y="domestic_gross", ax=axe
sns.lineplot(data=df_horror, x="tomatometer_rating", y="domestic_gross", ax=axes
sns.lineplot(data=df_mystery_suspence, x="tomatometer_rating", y="domestic_gross
sns.lineplot(data=df_action_adventure, x="tomatometer_rating", y="domestic_gross
sns.lineplot(data=df_animation, x="tomatometer_rating", y="domestic_gross", ax=a
sns.lineplot(data=df_art, x="tomatometer_rating", y="domestic_gross", ax=axes[1]
sns.lineplot(data=df_drama, x="tomatometer_rating", y="domestic_gross", ax=axes[
plt.savefig('images/GenresGrossRatingScatters.png');
```

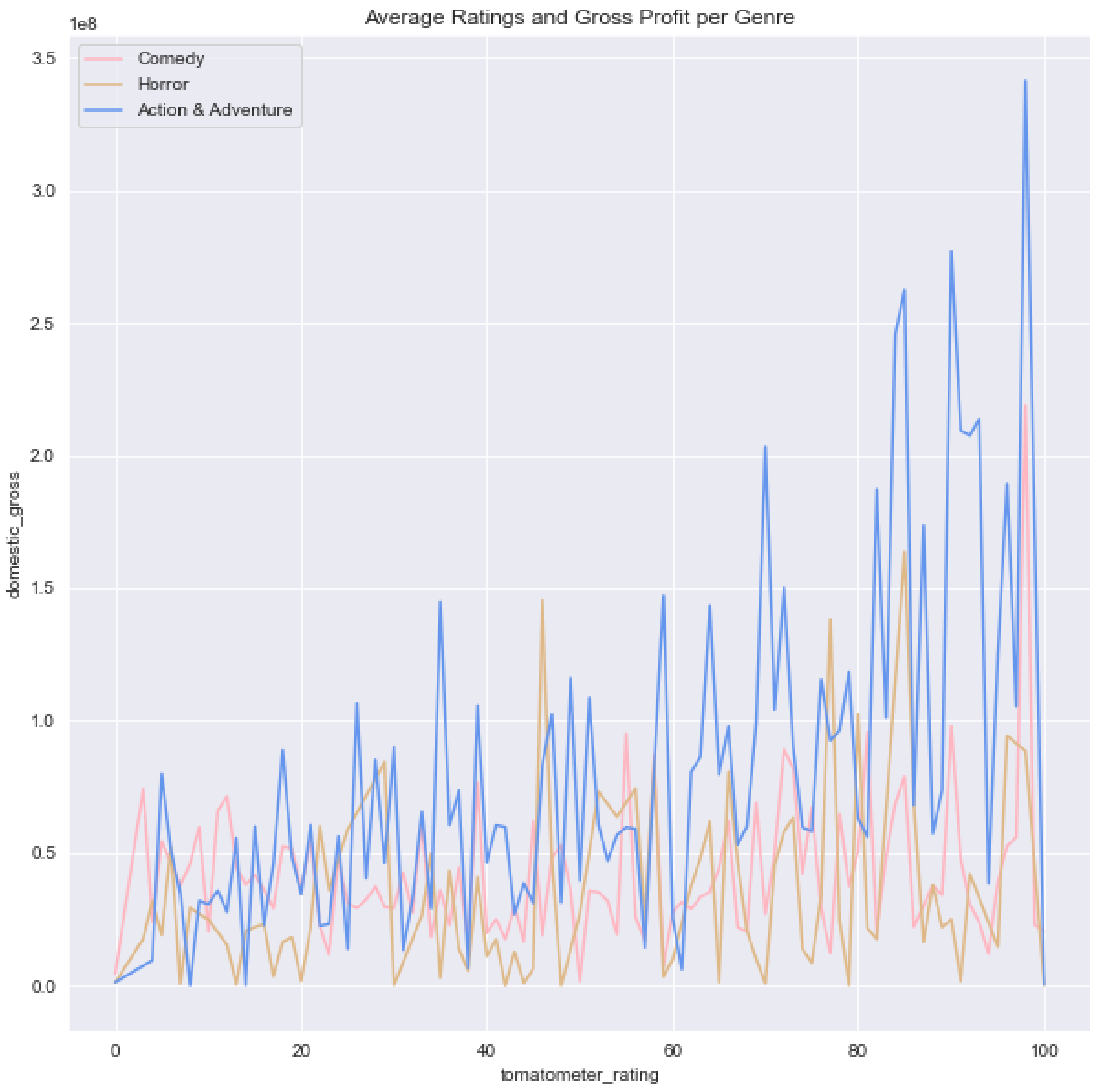


Comedy, Horror, and Action & Adventure appears to be the highest so this is a graph of them together

In [59]:
```
fig, ax = plt.subplots(1, 1)

ax.scatter(df_action_adventure['tomatometer_rating'], df_action_adventure["domes
ax.scatter(df_comedy['tomatometer_rating'], df_comedy["domestic_gross"], color="
ax.scatter(df_horror['tomatometer_rating'], df_horror["domestic_gross"], color="
#ax.scatter(df_drama['tomatometer_rating'], df_drama["domestic_gross"], color="o

ax.set_title('Average Ratings and Gross Profit per Genre')

ax.set_ylabel('Domestic Gross Profit')
ax.set_xlabel('Tomato Meter Rating')
ax.legend();
```

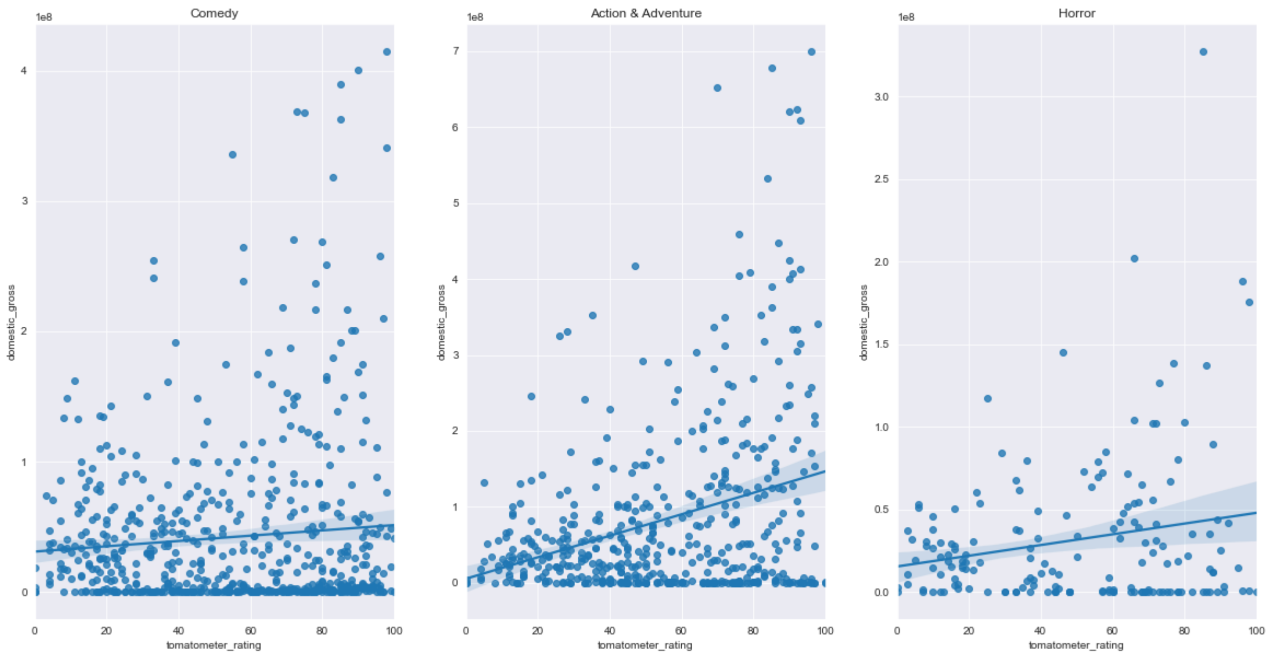


In [60]:
```
sns.set_style('darkgrid')
fig, ax = plt.subplots(figsize=(10,10))
sns.lineplot(data=df_comedy, x="tomatometer_rating", y="domestic_gross", ax=ax,
sns.lineplot(data=df_horror, x="tomatometer_rating", y="domestic_gross", ax=ax,
sns.lineplot(data=df_action_adventure, x="tomatometer_rating", y="domestic_gross
ax.legend(['Comedy', 'Horror', 'Action & Adventure']);
```



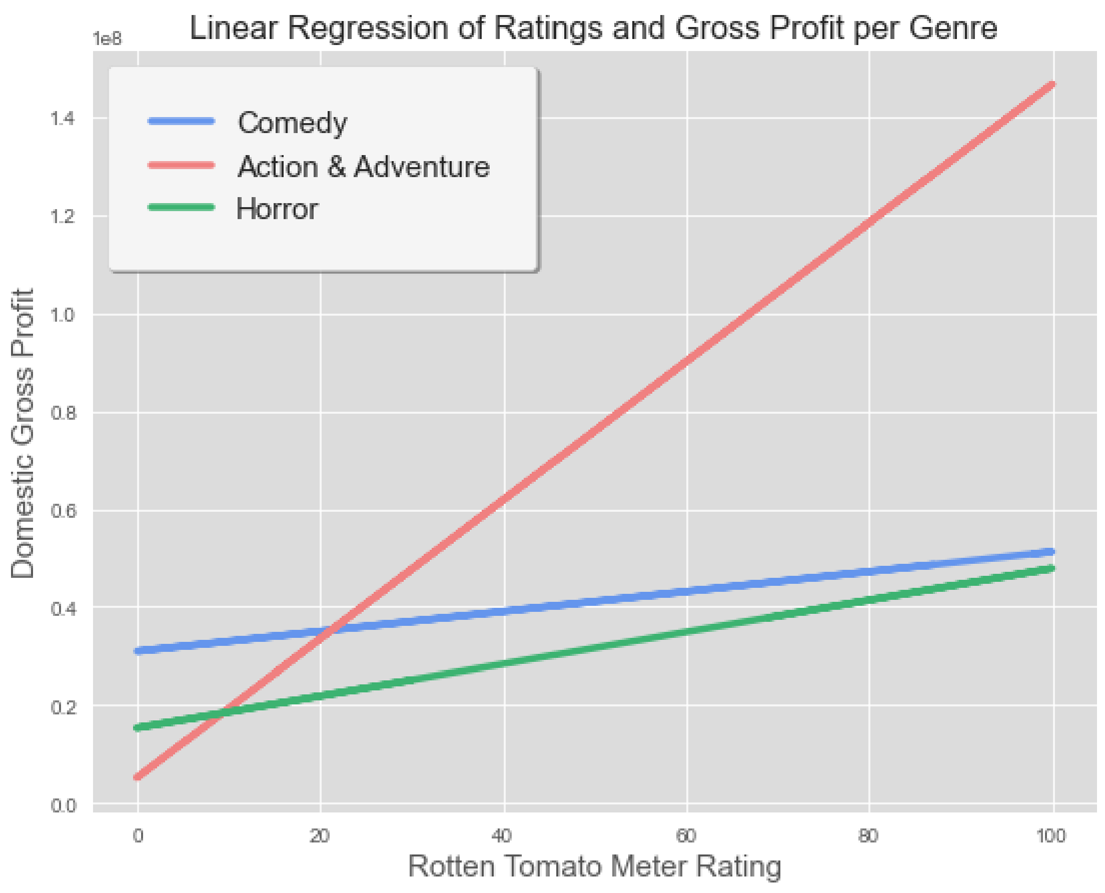
These are a little busy lets split them up and show them in a linear regression plot

In [61]:
```
fig, ax = plt.subplots(figsize=(20,10), ncols=3)
#sns.set(font_scale = 1.5)
sns.regplot(data=df_comedy, x="tomatometer_rating", y="domestic_gross",ax=ax[0])
sns.regplot(data=df_action_adventure, x="tomatometer_rating", y="domestic_gross"
sns.regplot(data=df_horror, x="tomatometer_rating", y="domestic_gross", ax=ax[2]

plt.savefig('images/LinearRegressionRatingsGrossGenreSplit.png');
```



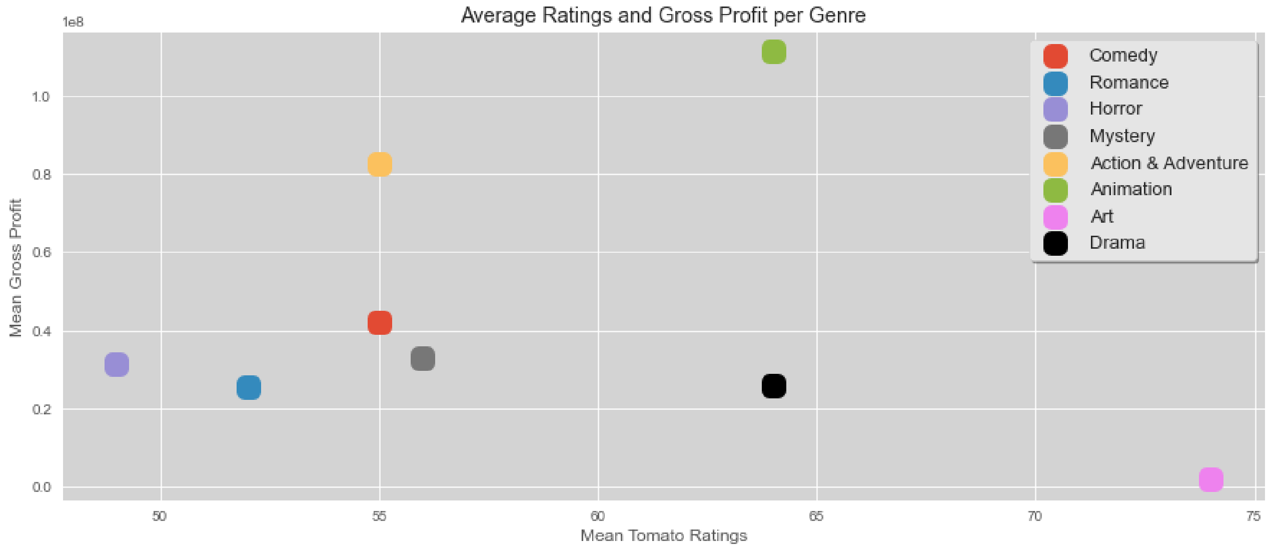
Lets simplify this to just show the linear regression lines for these top three genres and put them back together

In [62]:
```
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(9,7))

ax.set_facecolor('gainsboro')

x = np.array(df_comedy['tomatometer_rating'])
y = np.array(df_comedy['domestic_gross'])

m, b = np.polyfit(x, y, 1)
ax.plot(x, m*x + b, color='cornflowerblue', linewidth=3.5)

x = np.array(df_action_adventure['tomatometer_rating'])
y = np.array(df_action_adventure['domestic_gross'])

m, b = np.polyfit(x, y, 1)
ax.plot(x, m*x + b, color='lightcoral', linewidth=3.5)

x = np.array(df_horror['tomatometer_rating'])
y = np.array(df_horror['domestic_gross'])

m, b = np.polyfit(x, y, 1)
ax.plot(x, m*x + b, color='mediumseagreen', linewidth=3.5)

ax.set_title('Linear Regression of Ratings and Gross Profit per Genre', fontsize

ax.set_ylabel('Domestic Gross Profit', fontsize=15)
ax.set_xlabel('Rotten Tomato Meter Rating',fontsize=15)

plt.legend(['Comedy','Action & Adventure', 'Horror'], fancybox=True, framealpha=
           borderpad=1.5, prop={"size":15}, facecolor='whitesmoke')
plt.savefig('images/LinearRegressionRatingsGrossGenre.png');
```



This is a scatter plot of the means of the ratings and gross profit columns for all the genres graphed together on a scatter plot

```
In [63]: fig, ax = plt.subplots(figsize=(15,6))
         ax.scatter(x_comedy,y_comedy, marker='s', linewidth=11)
         ax.scatter(x_romance,y_romance, marker='s', linewidth=11)
         ax.scatter(x_horror,y_horror, marker='s', linewidth=11)
         ax.scatter(x_mystery,y_mystery, marker='s', linewidth=11)
         ax.scatter(x_action_adventure,y_action_adventure, marker='s', linewidth=11)
         ax.scatter(x_animation,y_animation, marker='s', linewidth=11)
         ax.scatter(x_art,y_art, marker='s', linewidth=11, color='violet')
         ax.scatter(x_drama,y_drama, marker='s', linewidth=11, color='black')


         ax.legend(['Comedy', 'Romance', 'Horror', 'Mystery', 'Action & Adventure', 'Anim
                    'Art', 'Drama'], prop={"size":13}, fancybox=True, framealpha=1, shadow

         ax.set_xlabel('Mean Tomato Ratings')
         ax.set_ylabel('Mean Gross Profit')
         ax.set_title('Average Ratings and Gross Profit per Genre')
         ax.set_facecolor('lightgrey')
         plt.savefig('images/AverageRatingsGrossperGenre.png')
```



It appears that the Action & Adventure genre will bring the most domestic profit and have the highest rating

---

Now lets put the months data together with our genre info to get the best months to release each genre

In [64]:
```
seasons_merged = pd.merge(df, df_gross, left_on='movie_title', right_on='title')
df_rating_seasons = merged[['movie_title','genres', 'domestic_gross', 'original_
```

In [65]:
```
df_rating_seasons
```

Out[65]:

	movie_title	genres	domestic_gross	original_release_date
0	Please Give	Comedy	4000000.0	2010-04-30
1	Going the Distance	Comedy	17800000.0	2004-08-20
2	Going the Distance	Comedy, Romance	17800000.0	2010-09-03
3	The Silence	Action & Adventure, Drama, Mystery & Suspense,...	100000.0	NaN
4	The Silence	Art House & International, Drama, Mystery & Su...	100000.0	2013-03-08
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	3100000.0	2011-05-27
2145	Zombeavers	Action & Adventure, Comedy, Horror	14900.0	2015-03-20
2146	Zookeeper	Comedy, Romance	80400000.0	2011-07-08
2147	Zoolander 2	Comedy	28800000.0	2016-02-12
2148	Zootopia	Action & Adventure, Animation, Comedy	341300000.0	2016-03-04

2149 rows × 4 columns

Checks the info to check that they DataFrames merged correctly

```
In [66]: df_rating_seasons.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2149 entries, 0 to 2148
Data columns (total 4 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   movie_title            2149 non-null   object 
 1   genres                 2149 non-null   object 
 2   domestic_gross         2137 non-null   float64
 3   original_release_date  2128 non-null   object 
dtypes: float64(1), object(3)
memory usage: 83.9+ KB
```

Drop the null values

```
In [67]: df_rating_seasons.dropna(inplace=True)
```

Drops duplicates from the merge

```
In [68]: df_rating_seasons.drop_duplicates(subset='movie_title', keep='last' , inplace=Tr
         df_rating_seasons
```

Out[68]:

	movie_title	genres	domestic_gross	original_release_date
0	Please Give	Comedy	4000000.0	2010-04-30
2	Going the Distance	Comedy, Romance	17800000.0	2010-09-03
5	The Silence	Horror, Mystery & Suspense	100000.0	2019-12-07
7	Gone	Mystery & Suspense	11700000.0	2012-02-24
8	Fireflies in the Garden	Drama	70600.0	2011-10-14
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	3100000.0	2011-05-27
2145	Zombeavers	Action & Adventure, Comedy, Horror	14900.0	2015-03-20
2146	Zookeeper	Comedy, Romance	80400000.0	2011-07-08
2147	Zoolander 2	Comedy	28800000.0	2016-02-12
2148	Zootopia	Action & Adventure, Animation, Comedy	341300000.0	2016-03-04

2049 rows × 4 columns

Splits the genres at the commas and puts them in a list

```
In [69]: df_rating_seasons['genres'] = df_rating_seasons['genres'].str.split(',', expand
```

Explodes the new lists to new rows

```
In [70]: df_rating_seasons = df_rating_seasons.explode('genres')
```

Strips the whitespace

```
In [71]: df_rating_seasons['genres'] = df_rating_seasons['genres'].str.strip()
```

Grabs the month from the release date column

```
In [72]: df_rating_seasons['month'] = df_rating_seasons['original_release_date'].apply(la
         df_rating_seasons
```

Out[72]:

	movie_title	genres	domestic_gross	original_release_date	month
0	Please Give	Comedy	4000000.0	2010-04-30	04
2	Going the Distance	Comedy	17800000.0	2010-09-03	09
2	Going the Distance	Romance	17800000.0	2010-09-03	09
5	The Silence	Horror	100000.0	2019-12-07	12
5	The Silence	Mystery & Suspense	100000.0	2019-12-07	12
...	...	...	...	...	...
2146	Zookeeper	Romance	80400000.0	2011-07-08	07
2147	Zoolander 2	Comedy	28800000.0	2016-02-12	02
2148	Zootopia	Action & Adventure	341300000.0	2016-03-04	03
2148	Zootopia	Animation	341300000.0	2016-03-04	03
2148	Zootopia	Comedy	341300000.0	2016-03-04	03

4085 rows × 5 columns

Makes a new column of the month names so that we can easily group by months

```
In [73]: df_rating_seasons['month_name'] = df_rating_seasons['month'].map({'01':'January'
         '02':'February',
         '03':'March',
         '04':'April',
         '05':'May',
         '06':'June',
         '07':'July',
         '08':'August',
         '09':'September',
         '10':'October',
         '11':'November',
         '12':'December'})
         df_rating_seasons
```

Out[73]:

	movie_title	genres	domestic_gross	original_release_date	month	month_name
0	Please Give	Comedy	4000000.0	2010-04-30	04	April
2	Going the Distance	Comedy	17800000.0	2010-09-03	09	September
2	Going the Distance	Romance	17800000.0	2010-09-03	09	September
5	The Silence	Horror	100000.0	2019-12-07	12	December

	movie_title	genres	domestic_gross	original_release_date	month	month_name
5	The Silence	Mystery & Suspense	100000.0	2019-12-07	12	December
...	...	...	...	...	...	...
2146	Zookeeper	Romance	80400000.0	2011-07-08	07	July
2147	Zoolander 2	Comedy	28800000.0	2016-02-12	02	February
2148	Zootopia	Action & Adventure	341300000.0	2016-03-04	03	March
2148	Zootopia	Animation	341300000.0	2016-03-04	03	March
2148	Zootopia	Comedy	341300000.0	2016-03-04	03	March

4085 rows × 6 columns

Groups the genres and month_name columns with the gross mean

In [74]:
```
real_df = df_rating_seasons.groupby(['genres', 'month_name']).mean().reset_index
real_df
```

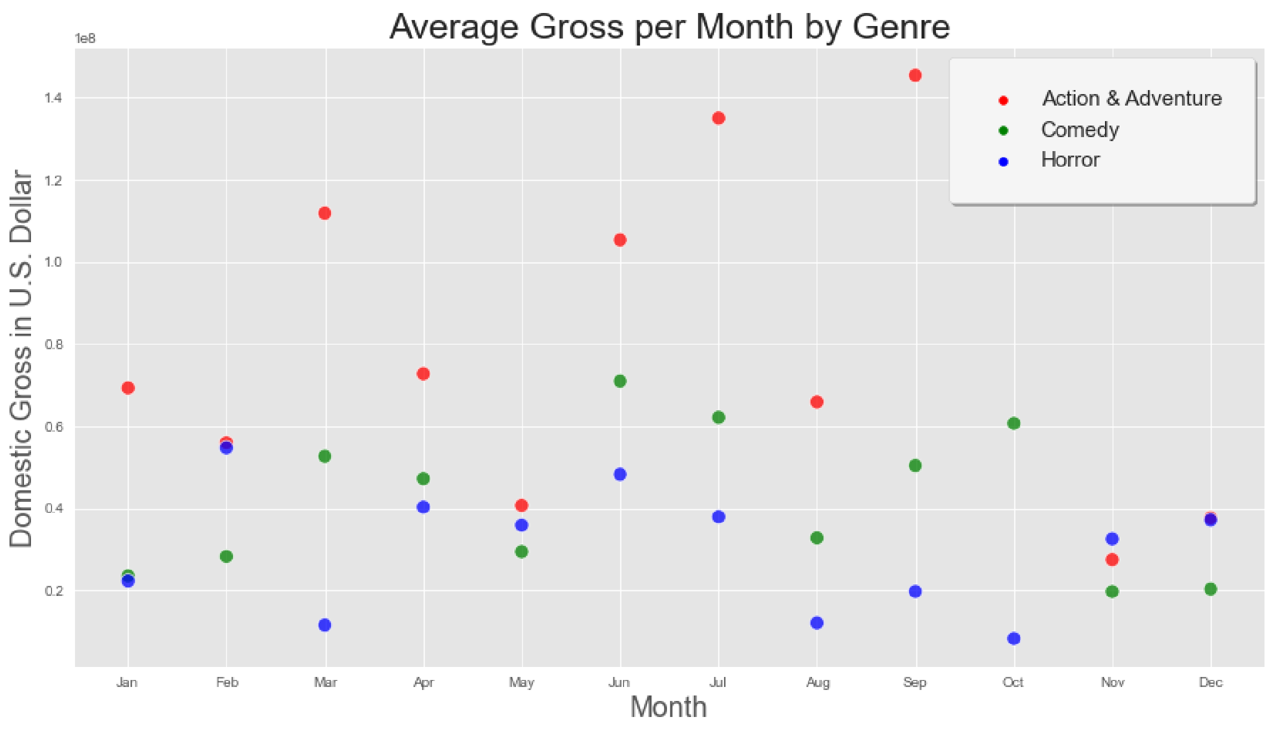
Out[74]:

	genres	month_name	domestic_gross
0	Action & Adventure	April	6.927307e+07
1	Action & Adventure	August	5.593980e+07
2	Action & Adventure	December	1.118571e+08
3	Action & Adventure	February	7.275662e+07
4	Action & Adventure	January	4.067424e+07
...	...	...	...
195	Western	July	9.475000e+07
196	Western	June	3.939000e+06
197	Western	May	3.610000e+07
198	Western	November	1.210550e+06
199	Western	October	2.010000e+05

200 rows × 3 columns

Graphs the Average Gross per Month by Genre

In [75]:
```
warnings.filterwarnings('ignore')
fig, ax = plt.subplots(figsize=(15, 8))

filtered = real_df.loc[(real_df['genres'] == 'Comedy')|(real_df['genres'] == 'Ho
x = sns.scatterplot(data = filtered, x = "month_name", y = "domestic_gross", hue

ax.legend(prop={"size":15}, loc=1, fancybox=True, framealpha=1, shadow=True,
             borderpad=1.5, facecolor='whitesmoke')
x.set_xticklabels(['Jan', 'Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct',
ax.set_xlabel('Month', fontsize=20)
ax.set_ylabel('Domestic Gross in U.S. Dollar', fontsize=20)
ax.set_title('Average Gross per Month by Genre', fontsize=25)

plt.savefig('images/AvgGrossPerMonthByGenre.png');
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

---

from file: movie_analysis_tables

Importing the different programs used

```
In [76]: import numpy as np
```

```
In [77]: import pandas as pd
```

```
In [78]: import seaborn as sns
```

```
In [79]: import matplotlib.pyplot as plt
         %matplotlib inline
```

Importing the Rotten Tomatoes data set

```
In [80]: df = pd.read_csv('data/zippedData/rotten_tomatoes_movies.csv.gz')
         df
```

Out[80]:

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	PG	S
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	R	
2	m/10	10	A successful, middle-aged Hollywood songwriter...	Blake Edwards' bawdy comedy may not score a pe...	R	
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	Following the closing arguments in a murder tr...	Sidney Lumet's feature debut is a superbly wri...	NR	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	In 1866, Professor Pierre M. Aronnax (Paul Luk...	One of Disney's finest live-action adventures,...	G	D
...	...	...	...	...	...	
17707	m/zoot_suit	Zoot Suit	Mexican-American gangster Henry Reyna (Daniel ...	NaN	R	
17708	m/zootopia	Zootopia	From the largest elephant to the smallest shre...	The brilliantly well-rounded Zootopia offers a...	PG	
17709	m/zorba_the_greek	Zorba the Greek	Traveling to inspect an abandoned mine his fat...	NaN	NR	Ac Int

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	
17710	m/zulu	Zulu	In 1879, the Zulu nation hands colonial Britis...	Zulu patiently establishes a cast of colorful ...	PG	
17711	m/zulu_dawn	Zulu Dawn	Sir Henry Bartle Frere's (John Mills) vastly o...	NaN	PG	Ac Int

17712 rows × 22 columns

Importing the Bom movie gross data set

In [81]:
```
df_gross = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')
df_gross
```

Out[81]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

Importing the movie budgets data set

In [82]:
```
df_budgets = pd.read_csv('data/zippedData/tn.movie_budgets.csv.gz')
df_budgets
```

Out[82]:

	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

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
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

Replacing the '$' and ',' in the movie budgets dat set, so that they can be numeric values.

```
In [83]: df_budgets['production_budget'] = df_budgets['production_budget'].str.replace('$
         df_budgets['domestic_gross'] = df_budgets['domestic_gross'].str.replace('$', '')
         df_budgets['production_budget'] = df_budgets['production_budget'].str.replace(',
         df_budgets['domestic_gross'] = df_budgets['domestic_gross'].str.replace(',', '')
         df_budgets
```

Out[83]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	425000000	760507625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0	$0
5778	79	Apr 2, 1999	Following	6000	48482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	$0

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	$181,041

5782 rows × 6 columns

Converting domestic_gross and production_budget columns to numbers, and dropping the worldwide_gross column. Dropped the worldwide gross because we are not utilizing it, and converting to numbers so we can use math with the values in the other two columns.

```
In [84]: df_budgets[['domestic_gross', 'production_budget']] = df_budgets[['domestic_gros
         df_budgets = df_budgets.drop(columns="worldwide_gross")
         df_budgets
```

Out[84]:

	id	release_date	movie	production_budget	domestic_gross
0	1	Dec 18, 2009	Avatar	425000000	760507625
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382
...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0
5778	79	Apr 2, 1999	Following	6000	48482
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0
5781	82	Aug 5, 2005	My Date With Drew	1100	181041

5782 rows × 5 columns

Creating a 'profit' column by subtracting production_budget from domestic_gross in order to show how much a movie is making.

```
In [85]: df_budgets["profit"] = df_budgets["domestic_gross"] - df_budgets["production_bud
         df_budgets
```

Out[85]:

	id	release_date	movie	production_budget	domestic_gross	profit
0	1	Dec 18, 2009	Avatar	425000000	760507625	335507625
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	-169536125
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	-307237650
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	128405868

	id	release_date	movie	production_budget	domestic_gross	profit
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	303181382
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0	-7000
5778	79	Apr 2, 1999	Following	6000	48482	42482
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	-3662
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	-1400
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	179941

5782 rows × 6 columns

Merged the Rotten Tomatoes data set with the Bom movie data set to create the rating_genres dataframe in order to have domestic gross with the genres.

In [86]:
```
merged = pd.merge(df, df_gross, left_on='movie_title', right_on='title')
df_rating_genres = merged[['movie_title','genres', 'tomatometer_rating', 'domest
df_rating_genres
```

Out[86]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
1	Going the Distance	Comedy	0.0	17800000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
3	The Silence	Action & Adventure, Drama, Mystery & Suspense,...	50.0	100000.0
4	The Silence	Art House & International, Drama, Mystery & Su...	88.0	100000.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	92.0	3100000.0
2145	Zombeavers	Action & Adventure, Comedy, Horror	69.0	14900.0
2146	Zookeeper	Comedy, Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure, Animation, Comedy	98.0	341300000.0

2149 rows × 4 columns

Dropping any null values from all columns in the rating_genres dataframe that we just merged so that there is no outliers without any value.

```
In [87]: df_rating_genres.dropna(inplace=True)
         df_rating_genres
```

Out[87]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
1	Going the Distance	Comedy	0.0	17800000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
3	The Silence	Action & Adventure, Drama, Mystery & Suspense,...	50.0	100000.0
4	The Silence	Art House & International, Drama, Mystery & Su...	88.0	100000.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	92.0	3100000.0
2145	Zombeavers	Action & Adventure, Comedy, Horror	69.0	14900.0
2146	Zookeeper	Comedy, Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure, Animation, Comedy	98.0	341300000.0

2136 rows × 4 columns

Dropping duplicates in the movie_title column so the data does not get skewed by the duplicates.

```
In [88]: df_rating_genres.drop_duplicates(subset='movie_title', keep='last' , inplace=Tru
         df_rating_genres
```

Out[88]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
2	Going the Distance	Comedy, Romance	54.0	17800000.0
5	The Silence	Horror, Mystery & Suspense	30.0	100000.0
7	Gone	Mystery & Suspense	12.0	11700000.0
8	Fireflies in the Garden	Drama	22.0	70600.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	Art House & International, Comedy, Drama	92.0	3100000.0
2145	Zombeavers	Action & Adventure, Comedy, Horror	69.0	14900.0
2146	Zookeeper	Comedy, Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0

	movie_title	genres	tomatometer_rating	domestic_gross
2148	Zootopia	Action & Adventure, Animation, Comedy	98.0	341300000.0

2063 rows × 4 columns

Making the genres column values into a list to be able to explode the genre column.

```
In [89]: df_rating_genres['genres'] = df_rating_genres['genres'].str.split(',', expand =
         df_rating_genres
```

Out[89]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	[Comedy]	87.0	4000000.0
2	Going the Distance	[Comedy, Romance]	54.0	17800000.0
5	The Silence	[Horror, Mystery & Suspense]	30.0	100000.0
7	Gone	[Mystery & Suspense]	12.0	11700000.0
8	Fireflies in the Garden	[Drama]	22.0	70600.0
...	...	...	...	...
2144	Zindagi Na Milegi Dobara	[Art House & International, Comedy, Drama]	92.0	3100000.0
2145	Zombeavers	[Action & Adventure, Comedy, Horror]	69.0	14900.0
2146	Zookeeper	[Comedy, Romance]	14.0	80400000.0
2147	Zoolander 2	[Comedy]	22.0	28800000.0
2148	Zootopia	[Action & Adventure, Animation, Comedy]	98.0	341300000.0

2063 rows × 4 columns

Exploding the genre column so that all the genres are seperated.

```
In [90]: df_rating_genres = df_rating_genres.explode('genres')
         df_rating_genres
```

Out[90]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
2	Going the Distance	Comedy	54.0	17800000.0
2	Going the Distance	Romance	54.0	17800000.0
5	The Silence	Horror	30.0	100000.0
5	The Silence	Mystery & Suspense	30.0	100000.0
...	...	...	...	...
2146	Zookeeper	Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0

	movie_title	genres	tomatometer_rating	domestic_gross
2148	Zootopia	Action & Adventure	98.0	341300000.0
2148	Zootopia	Animation	98.0	341300000.0
2148	Zootopia	Comedy	98.0	341300000.0

4118 rows × 4 columns

Now every movie has its own row with each of its genre.

```
In [91]: df_rating_genres['genres'] = df_rating_genres['genres'].str.strip()
         df_rating_genres
```

Out[91]:

	movie_title	genres	tomatometer_rating	domestic_gross
0	Please Give	Comedy	87.0	4000000.0
2	Going the Distance	Comedy	54.0	17800000.0
2	Going the Distance	Romance	54.0	17800000.0
5	The Silence	Horror	30.0	100000.0
5	The Silence	Mystery & Suspense	30.0	100000.0
...	...	...	...	...
2146	Zookeeper	Romance	14.0	80400000.0
2147	Zoolander 2	Comedy	22.0	28800000.0
2148	Zootopia	Action & Adventure	98.0	341300000.0
2148	Zootopia	Animation	98.0	341300000.0
2148	Zootopia	Comedy	98.0	341300000.0

4118 rows × 4 columns

Merging the df_rating_genres dataframe with the df_budgets dataframe, in order to get production_budget into the dataframe

```
In [92]: merged = pd.merge(df_rating_genres, df_budgets, left_on='movie_title', right_on=
         df_production = merged[['movie', 'genres', 'production_budget']].copy()
         df_production
```

Out[92]:

	movie	genres	production_budget
0	Please Give	Comedy	3000000
1	Going the Distance	Comedy	32000000
2	Going the Distance	Romance	32000000
3	Fireflies in the Garden	Drama	8000000
4	Priest	Action & Adventure	60000000
...	...	...	...
2319	Zookeeper	Romance	80000000

	movie	genres	production_budget
2320	Zoolander 2	Comedy	50000000
2321	Zootopia	Action & Adventure	150000000
2322	Zootopia	Animation	150000000
2323	Zootopia	Comedy	150000000

2324 rows × 3 columns

Sorting the production_budget column from greatest to least to see what the highest values are.

```
In [93]: df_sorted = df_production.sort_values('production_budget', ascending=False)
         df_sorted
```

Out[93]:

	movie	genres	production_budget
1347	Pirates of the Caribbean: On Stranger Tides	Comedy	410600000
1346	Pirates of the Caribbean: On Stranger Tides	Action & Adventure	410600000
1348	Pirates of the Caribbean: On Stranger Tides	Science Fiction & Fantasy	410600000
320	Avengers: Age of Ultron	Science Fiction & Fantasy	330600000
319	Avengers: Age of Ultron	Action & Adventure	330600000
...	...	...	...
2307	Your Sister's Sister	Comedy	120000
172	A Ghost Story	Drama	100000
1853	The Gallows	Mystery & Suspense	100000
1852	The Gallows	Horror	100000
2131	Tiny Furniture	Comedy	50000

2324 rows × 3 columns

Making the production budget in hundreds of thousands so it is easier to read on the graph.

```
In [94]: df_sorted['new_x'] = df_sorted['production_budget'] / 100000
```

```
In [95]: df_sorted['new_x'] = df_sorted['new_x'].sort_values(ascending=False)
         df_sorted['new_x']
```

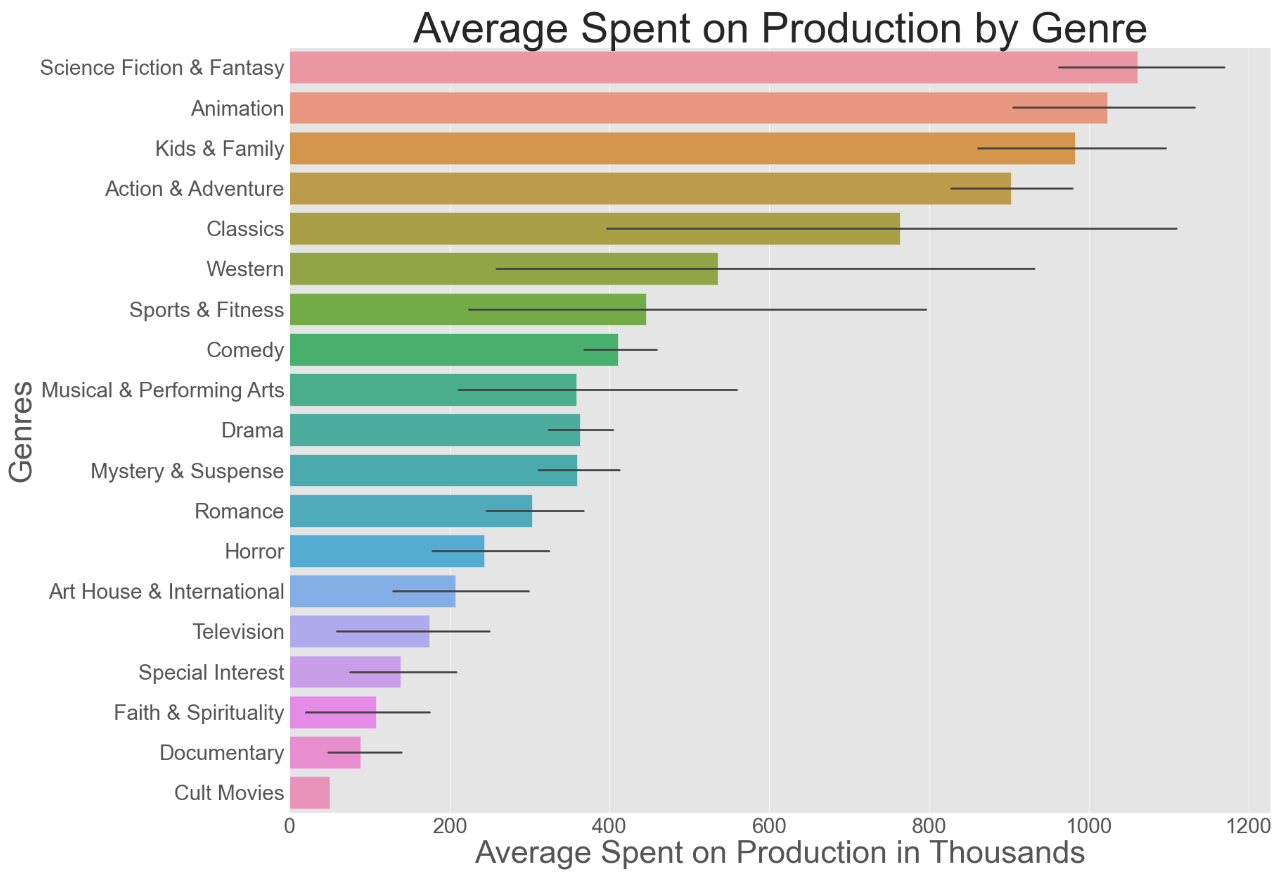
```
Out[95]: 1347    4106.0
         1346    4106.0
         1348    4106.0
         320     3306.0
         319     3306.0
                  ...
         2307       1.2
         172        1.0
         1853       1.0
         1852       1.0
         2131       0.5
         Name: new_x, Length: 2324, dtype: float64
```

Ordered where we want the genres to be on the y-axis so that its greatest to least.

In [96]:
```
order_list = ['Science Fiction & Fantasy', 'Animation', 'Kids & Family', 'Action
```

Seaborn barplot to show average spent on production by genre.

In [97]:
```
plot_fig_two, plot_two_ax = plt.subplots(figsize=(25, 20))

sns.barplot(x='new_x', y='genres', data=df_sorted, order=order_list)

plot_two_ax.set_xlabel('Average Spent on Production in Thousands', fontsize=45)
plot_two_ax.set_ylabel('Genres', fontsize=45)
plot_two_ax.set_title('Average Spent on Production by Genre', fontsize=60)


locs, labels = plt.xticks(fontsize=30)
locs, labels = plt.yticks(fontsize=30)
#plt.setp(labels, rotation=90, fontsize=30)
plt.savefig('average_production.png');
```



In [ ]:

In [ ]:

In [ ]: