1. Import SQL, NumPy, pandas, Matplotlib for Python

In [1]:

```
import sqlite3
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

- **Connect to the database file `movies.db` and use the file path `data/`.**

In [2]:

```
conn = sqlite3.connect('data\movies.db')
```

2. Open the database file as a dataframe called `query`.

In addition, you will `JOIN` tables and `SELECT` all the rows below to get the information you need.

Since this is a `db` file, you will need to use `pd.read_sql` and input the SQL code to get the output below.

In [3]:

```
#Print the query dataframe and filter the rows needed
query = pd.read_sql('''SELECT DISTINCT tconst, g.title, i.genres, g.studio,  g.year,
                       t.production_budget, t.worldwide_gross
                       FROM bom_movie_gross as g
                       JOIN imdb_title_basics as i
                       ON g.title = i.primary_title
                       AND g.year = i.start_year
                       JOIN tn_movie_budgets as t
                       ON t.movie = i.primary_title''', conn)
query.head()
```

Out[3]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross
0	tt0435761	Toy Story 3	Adventure,Animation,Comedy	BV	2010	$200,000,000	$1,068,879,522
1	tt1375666	Inception	Action,Adventure,Sci-Fi	WB	2010	$160,000,000	$835,524,642
2	tt0892791	Shrek Forever After	Adventure,Animation,Comedy	P/DW	2010	$165,000,000	$756,244,673
3	tt1325004	The Twilight Saga: Eclipse	Adventure,Drama,Fantasy	Sum.	2010	$68,000,000	$706,102,828
4	tt1228705	Iron Man 2	Action,Adventure,Sci-Fi	Par.	2010	$170,000,000	$621,156,389

I've joined the `bom_movie_gross` table and the `imdb_title_basics` table to get the data I need for Microsoft. All columns selected are `DISTINCT` so we receive different values for each movie. In addition, we used the `ON` clause to `JOIN` multiple columns. First, we connected the tables based on movie title - `g.title = i.primary_title`. Then we connected the tables based on year - `g.year = i.start_year` - because we only need movies under the `title` column listed once. Otherwise, we would have "Frozen" listed three times.

I've also joined the `tn_movie_budgets` table with the `imdb_basiccs_title` table based on movie title - `t.movie = i.primary_title`. This will allow us to add `production_budget` and `worldwide_gross` columns in the next steps.

3. Convert `production_budget` and `worldwide_gross` columns to floats in the dataframe.

First, we need to remove the commas (,) and the $ from the dataframe.

In [4]:

```
#Eliminate all $ and commas from the columns
query['production_budget'] = query['production_budget'].map(lambda x: x.replace(',', '')
)
query['production_budget'] = query['production_budget'].map(lambda x: x.replace('$', '')
)
query['worldwide_gross'] = query['worldwide_gross'].map(lambda x: x.replace(',', ''))
query['worldwide_gross'] = query['worldwide_gross'].map(lambda x: x.replace('$', ''))
```

In [5]:

```
query
```

Out[5]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross
0	tt0435761	Toy Story 3	Adventure,Animation,Comedy	BV	2010	200000000	1068879522
1	tt1375666	Inception	Action,Adventure,Sci-Fi	WB	2010	160000000	835524642
2	tt0892791	Shrek Forever After	Adventure,Animation,Comedy	P/DW	2010	165000000	756244673
3	tt1325004	The Twilight Saga: Eclipse	Adventure,Drama,Fantasy	Sum.	2010	68000000	706102828
4	tt1228705	Iron Man 2	Action,Adventure,Sci-Fi	Par.	2010	170000000	621156389
...	...	...	...	...	...	...	...
1070	tt1034415	Suspiria	Fantasy,Horror,Mystery	Amazon	2018	20000000	7034615
1071	tt5360952	The Hurricane Heist	Action,Adventure,Crime	ENTMP	2018	40000000	30963684
1072	tt7137380	Destroyer	Action,Crime,Drama	Annapurna	2018	9000000	3681096
1073	tt1801552	Gotti	Biography,Crime,Drama	VE	2018	10000000	6089100
1074	tt6998518	Mandy	Action,Fantasy,Horror	RLJ	2018	6000000	1427656

1075 rows × 7 columns

Next, the `production_budget` column and the `worldwide_gross` column will have to be converted from a string to a float. Check the dtypes for all columns and you'll see both columns are currently strings - `object`.

In [6]:

```
query.dtypes
```

Out[6]:

```
tconst               object
title                object
genres               object
studio               object
year                  int64
production_budget    object
worldwide_gross      object
dtype: object
```

In [7]:

```
#Convert columns from strings to floats
query.production_budget = query.production_budget.astype(float)
query.worldwide_gross = query.worldwide_gross.astype(float)
query
```

Out[7]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross
0	tt0435761	Toy Story 3	Adventure,Animation,Comedy	BV	2010	200000000.0	1.068880e+09
1	tt1375666	Inception	Action,Adventure,Sci-Fi	WB	2010	160000000.0	8.355246e+08
2	tt0892791	Shrek Forever After	Adventure,Animation,Comedy	P/DW	2010	165000000.0	7.562447e+08
3	tt1325004	The Twilight Saga: Eclipse	Adventure,Drama,Fantasy	Sum.	2010	68000000.0	7.061028e+08
4	tt1228705	Iron Man 2	Action,Adventure,Sci-Fi	Par.	2010	170000000.0	6.211564e+08
...	...	...	...	...	...	...	...
1070	tt1034415	Suspiria	Fantasy,Horror,Mystery	Amazon	2018	20000000.0	7.034615e+06
1071	tt5360952	The Hurricane Heist	Action,Adventure,Crime	ENTMP	2018	40000000.0	3.096368e+07
1072	tt7137380	Destroyer	Action,Crime,Drama	Annapurna	2018	9000000.0	3.681096e+06
1073	tt1801552	Gotti	Biography,Crime,Drama	VE	2018	10000000.0	6.089100e+06
1074	tt6998518	Mandy	Action,Fantasy,Horror	RLJ	2018	6000000.0	1.427656e+06

1075 rows × 7 columns

Run the dtypes again and you'll see both columns are now floats - `float64`.

In [8]:

```
#Verify change
query.dtypes
```

Out[8]:

```
tconst                object
title                 object
genres                object
studio                object
year                   int64
production_budget    float64
worldwide_gross      float64
dtype: object
```

4. Add a new column called `profit` to the dataframe.

We'll use the newly created `profit` column as our primary metric for our data!

In [9]:

```
#Add 'profit' column to the dataframe
query['profit'] = query['worldwide_gross'] - query['production_budget']
query
```

Out[9]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit
0	tt0435761	Toy Story 3	Adventure,Animation,Comedy	BV	2010	200000000.0	1.068880e+09	868879522.0
1	tt1375666	Inception	Action,Adventure,Sci-Fi	WB	2010	160000000.0	8.355246e+08	675524642.0
2	tt0892791	Shrek Forever After	Adventure,Animation,Comedy	P/DW	2010	165000000.0	7.562447e+08	591244673.0
3	tt1325004	The Twilight Saga: Eclipse	Adventure,Drama,Fantasy	Sum.	2010	68000000.0	7.061028e+08	638102828.0

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit
4	tt1228705	Iron Man 2	Action,Adventure,Sci-Fi	Par.	2010	170000000.0	6.211564e+08	451156389.0
...	...	...	...	...	...	...	...	...
1070	tt1034415	Suspiria	Fantasy,Horror,Mystery	Amazon	2018	20000000.0	7.034615e+06	-12965385.0
1071	tt5360952	The Hurricane Heist	Action,Adventure,Crime	ENTMP	2018	40000000.0	3.096368e+07	-9036316.0
1072	tt7137380	Destroyer	Action,Crime,Drama	Annapurna	2018	9000000.0	3.681096e+06	-5318904.0
1073	tt1801552	Gotti	Biography,Crime,Drama	VE	2018	10000000.0	6.089100e+06	-3910900.0
1074	tt6998518	Mandy	Action,Fantasy,Horror	RLJ	2018	6000000.0	1.427656e+06	-4572344.0

1075 rows × 8 columns

5. Find the Top 10 Most Profitable Films using the `profit` column.

Sort the dataframe in descending order and then reduce the list down to the top 10 profitable films.

In [10]:

```
#Top 10 Profitable Films
query = query.sort_values('profit', ascending=False).iloc[:10]
```

In [11]:

```
query
```

Out[11]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09

6. Sort the `query` database to display only the movie titles and their profits.

I want to use this information for a bar plot I'll create in the next step.

Sort the values to display only movie titles and their profits.

In [12]:

```
movie_profits = query.set_index('title')['profit'].sort_values(ascending=False)
```

In [13]:

```
movie_profits
```

Out[13]:

```
title
Avengers: Infinity War              1.748134e+09
Jurassic World                      1.433855e+09
Furious 7                           1.328723e+09
Black Panther                       1.148258e+09
Jurassic World: Fallen Kingdom      1.135773e+09
Frozen                              1.122470e+09
Minions                             1.086336e+09
Avengers: Age of Ultron             1.072414e+09
Incredibles 2                       1.042521e+09
Iron Man 3                          1.015392e+09
Name: profit, dtype: float64
```

7. Generate a bar plot

In the cell below, create a sorted bar chart displaying the Top 10 Films with the highest `profit`.

Use `fig` and `ax` as your variables.

Your chart should have the following:

1. **A figsize set to** `(15,8)`
2. **A title set to** `Top 10 Worldwide Profitable Movies`
3. **A ylabel set to** `Total Profit (in billions $)`
4. **An xlabel set to** `Movies`

In [14]:

```
movie_profits.index
```

Out[14]:

```
Index(['Avengers: Infinity War', 'Jurassic World', 'Furious 7',
       'Black Panther', 'Jurassic World: Fallen Kingdom', 'Frozen', 'Minions',
       'Avengers: Age of Ultron', 'Incredibles 2', 'Iron Man 3'],
      dtype='object', name='title')
```

In [15]:

```
movie_profits.values
```

Out[15]:

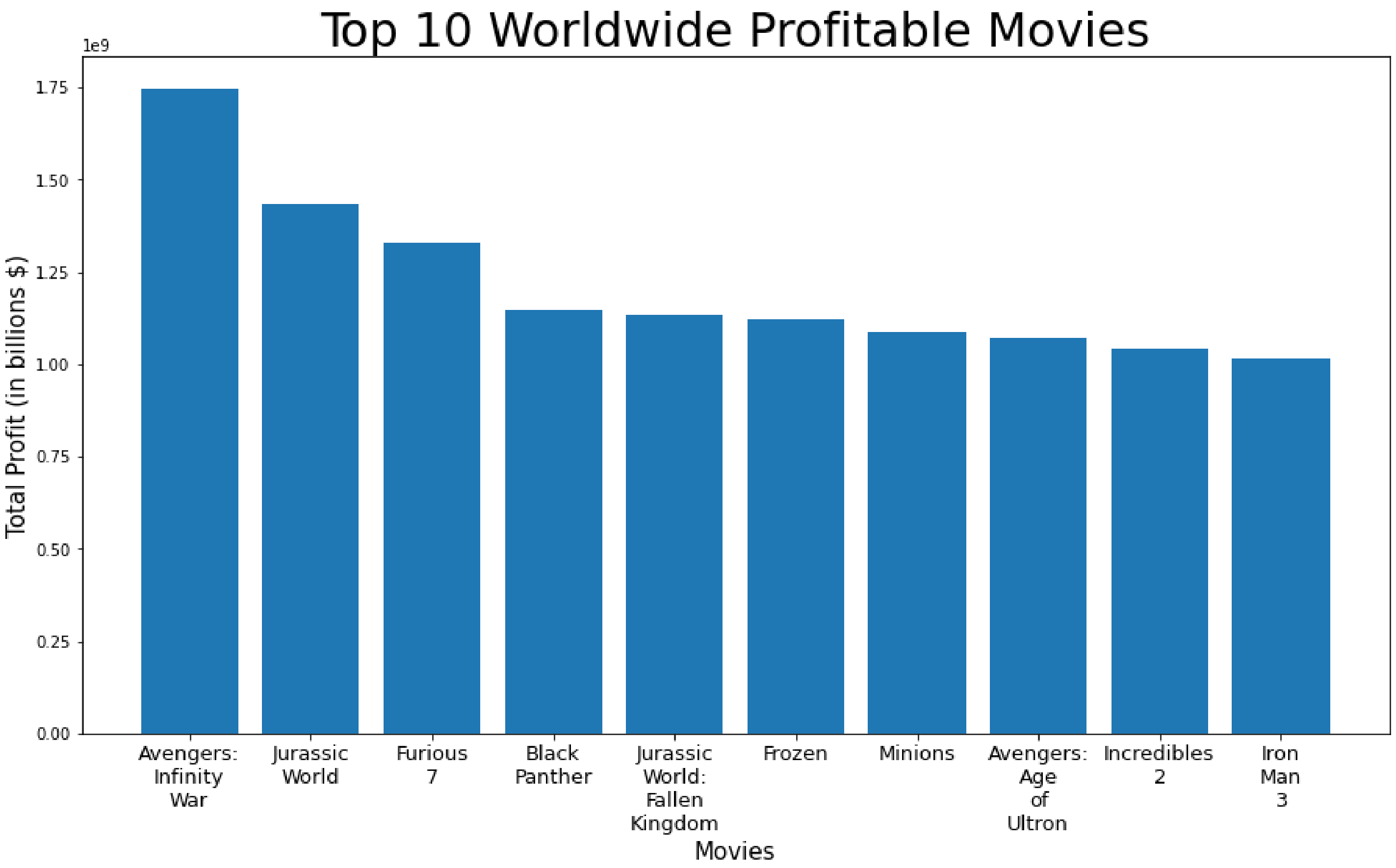
```
array([1.74813420e+09, 1.43385486e+09, 1.32872279e+09, 1.14825822e+09,
       1.13577280e+09, 1.12246991e+09, 1.08633617e+09, 1.07241396e+09,
       1.04252071e+09, 1.01539227e+09])
```

In [16]:

```
#Create bar chart showing Top 10 Worldwide Profitable Movies
x = movie_profits.index
y = movie_profits.values

fig, ax = plt.subplots(figsize=(15,8))
```

```
ax.bar(x,y)

ax.set_title('Top 10 Worldwide Profitable Movies', fontsize=30)
ax.set_xlabel('Movies', fontsize=15)
ax.set_ylabel('Total Profit (in billions $)', fontsize=15)

ax.set_xticklabels(x.str.replace(' ', '\n'))
ax.tick_params(axis='x', labelsize=13)
```

```
<ipython-input-16-30e8a7f3da02>:13: UserWarning: FixedFormatter should only be used toget
her with FixedLocator
  ax.set_xticklabels(x.str.replace(' ', '\n'))
```



The data above lets us know which movies brought in the highest profits around the globe.

8. Create a new column called `new_genres` and then split each genre into its own row.

We're doing this because we want to count the number of times each genre appears for our top 10 profitable movies, and we want to know which genre is the most profitable in the top 10.

To do this, you will need to use the `genres` column and split each genre from each other.

In [17]:

```
query
```

Out[17]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09

644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09

In [18]:

```
query['new_genres'] = query['genres'].str.split(',',3)
```

In [19]:

```
query
```

Out[19]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit	ne
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	/
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	/
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	/
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	/
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	[/ /
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	[/ /
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	/
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	/ /
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	/

Next, update the dataframe using the `explode` function so that each genre under `new_genres` is listed in its own row.

In [20]:

```
query = query.explode('new_genres')
```

In [21]:

```
query
```

Out[21]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit	ne
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	
645	tt2395427	Avengers: Age of	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	

	tconst	Ultron title	genres	studio	year	production_budget	worldwide_gross	profit	ne
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	

9. Group and sort in descending order all the genres associated with the Top 10 Worldwide Profitable Movies.

You will use `new_genres` column to find this.

In [22]:

```
query_genres = query.groupby('new_genres')['profit'].count()
query_genres
```

Out[22]:

```
new_genres
Action       8
Adventure    9
Animation    3
Comedy       2
Crime        1
Sci-Fi       6
Thriller     1
Name: profit, dtype: int64
```

In [23]:

```
genres_sorted = query_genres.sort_values(ascending=False)
genres_sorted
```

Out[23]:

```
new_genres
Adventure    9
Action       8
Sci-Fi       6
Animation    3
Comedy       2
Thriller     1
Crime        1
Name: profit, dtype: int64
```

10. Generate a bar plot

In the cell below, create a sorted bar chart displaying the most popular genres within the Top 10 Worldwide Profitable Movies.

Use `fig` and `ax` as your variables.

Your chart should have the following:

1. **A figsize set to** `(15,8)`
2. **A title set to** `Most Popular Genres From Most Profitable Movies`
3. **A ylabel set to** `Total Genre Count`
4. **An xlabel set to** `Genres`

In [24]:

```
genres_sorted.index
```

Out[24]:

```
Index(['Adventure', 'Action', 'Sci-Fi', 'Animation', 'Comedy', 'Thriller',
       'Crime'],
      dtype='object', name='new_genres')
```

In [25]:

```
genres_sorted.values
```

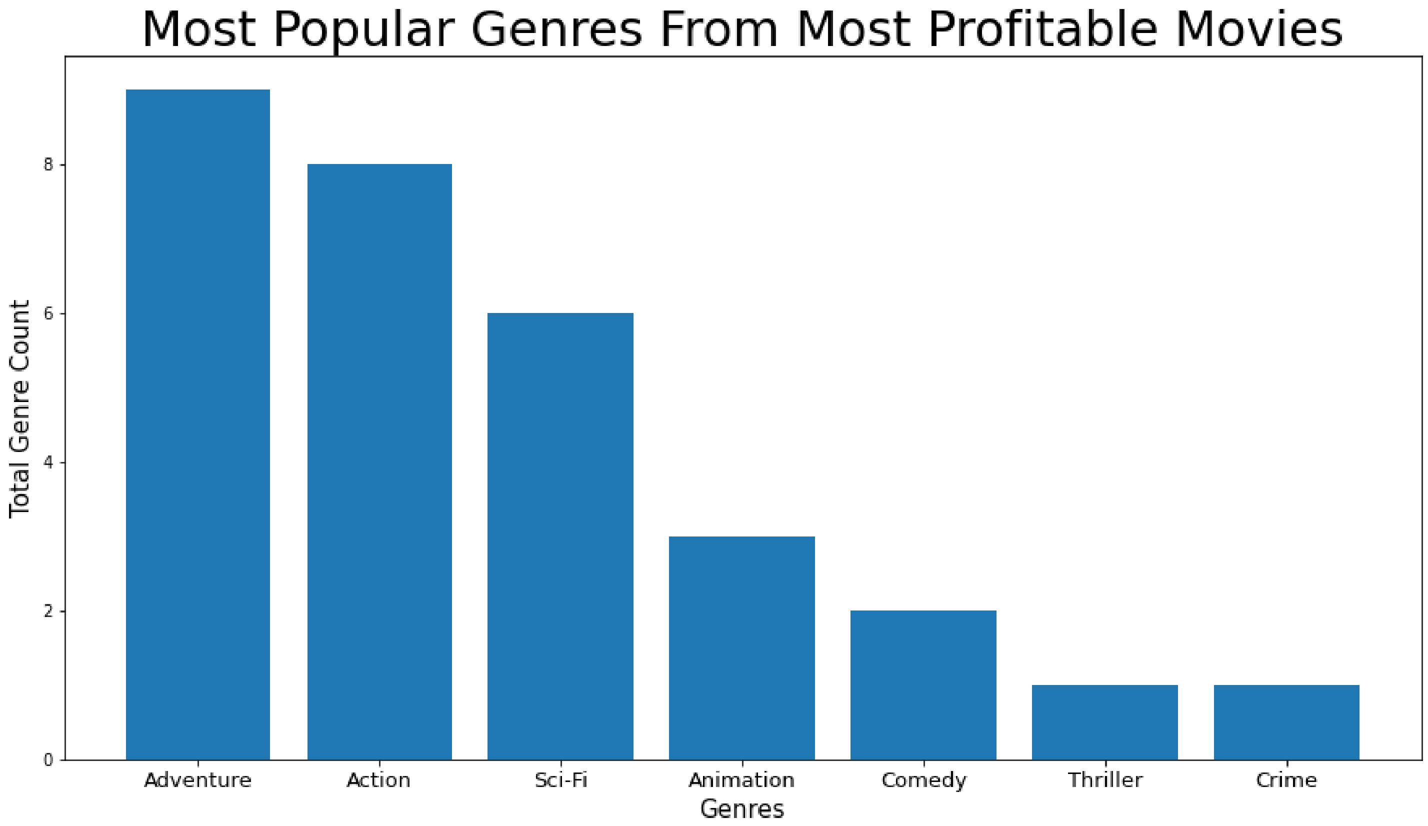
Out[25]:

```
array([9, 8, 6, 3, 2, 1, 1], dtype=int64)
```

In [26]:

```
#Create bar chart showing the most popular genres from the top 10 most profitable movies
x = genres_sorted.index
y = genres_sorted.values

fig, ax = plt.subplots(figsize=(15,8))

ax.bar(x,y)

ax.set_title('Most Popular Genres From Most Profitable Movies', fontsize=30)
ax.set_xlabel('Genres', fontsize=15)
ax.set_ylabel('Total Genre Count', fontsize=15)

ax.set_xticklabels(x.str.replace(',', '\n'))
ax.tick_params(axis='x', labelsize=13)
```

```
<ipython-input-26-5a3094562b83>:13: UserWarning: FixedFormatter should only be used toget
her with FixedLocator
  ax.set_xticklabels(x.str.replace(',', '\n'))
```



The data above lets us know how frequent each genre appears in our Top 10 Worldwide Profitable Movies bar chart. This is important because it lets us know which genre is most popular with audiences.

11. Find out which genre is making the most profit from the Top 10 Most Profitable Movies.

Once again, you will use the `new_genres` column to do this.

In [27]:

```
query
```

Out[27]:

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit	ne
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
987	tt4154756	Avengers: Infinity War	Action,Adventure,Sci-Fi	BV	2018	300000000.0	2.048134e+09	1.748134e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
643	tt0369610	Jurassic World	Action,Adventure,Sci-Fi	Uni.	2015	215000000.0	1.648855e+09	1.433855e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
644	tt2820852	Furious 7	Action,Crime,Thriller	Uni.	2015	190000000.0	1.518723e+09	1.328723e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
988	tt1825683	Black Panther	Action,Adventure,Sci-Fi	BV	2018	200000000.0	1.348258e+09	1.148258e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
989	tt4881806	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	Uni.	2018	170000000.0	1.305773e+09	1.135773e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
407	tt2294629	Frozen	Adventure,Animation,Comedy	BV	2013	150000000.0	1.272470e+09	1.122470e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	

	tconst	title	genres	studio	year	production_budget	worldwide_gross	profit	ne
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
646	tt2293640	Minions	Adventure,Animation,Comedy	Uni.	2015	74000000.0	1.160336e+09	1.086336e+09	
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	
645	tt2395427	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	BV	2015	330600000.0	1.403014e+09	1.072414e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
990	tt3606756	Incredibles 2	Action,Adventure,Animation	BV	2018	200000000.0	1.242521e+09	1.042521e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	
408	tt1300854	Iron Man 3	Action,Adventure,Sci-Fi	BV	2013	200000000.0	1.215392e+09	1.015392e+09	

In [28]:

```
genre_profits =query.groupby('new_genres')['profit'].sum()
genre_profits
```

Out[28]:

```
new_genres
Action          9.925070e+09
Adventure       1.080515e+10
Animation       3.251327e+09
Comedy          2.208806e+09
Crime           1.328723e+09
Sci-Fi          7.553826e+09
Thriller        1.328723e+09
Name: profit, dtype: float64
```

In [29]:

```
genre_profits_sorted = genre_profits.sort_values(ascending=False)
genre_profits_sorted
```

Out[29]:

```
new_genres
Adventure       1.080515e+10
Action          9.925070e+09
Sci-Fi          7.553826e+09
Animation       3.251327e+09
Comedy          2.208806e+09
Thriller        1.328723e+09
Crime           1.328723e+09
Name: profit, dtype: float64
```

12. Generate a bar plot

In the cell below, create a sorted bar chart displaying the most profitable genres within the Top 10 Worldwide Profitable Movies.

Use `fig` and `ax` as your variables.

Your chart should have the following:

1. **A figsize set to** `(15,8)`
2. **A title set to** `Most Profitable Genres`
3. **A ylabel set to** `Profit(in billions $)`
4. **An xlabel set to** `Genres`

In [30]:

```
genre_profits_sorted.index
```

Out[30]:

```
Index(['Adventure', 'Action', 'Sci-Fi', 'Animation', 'Comedy', 'Thriller',
       'Crime'],
      dtype='object', name='new_genres')
```

In [31]:

```
genre_profits_sorted.values
```

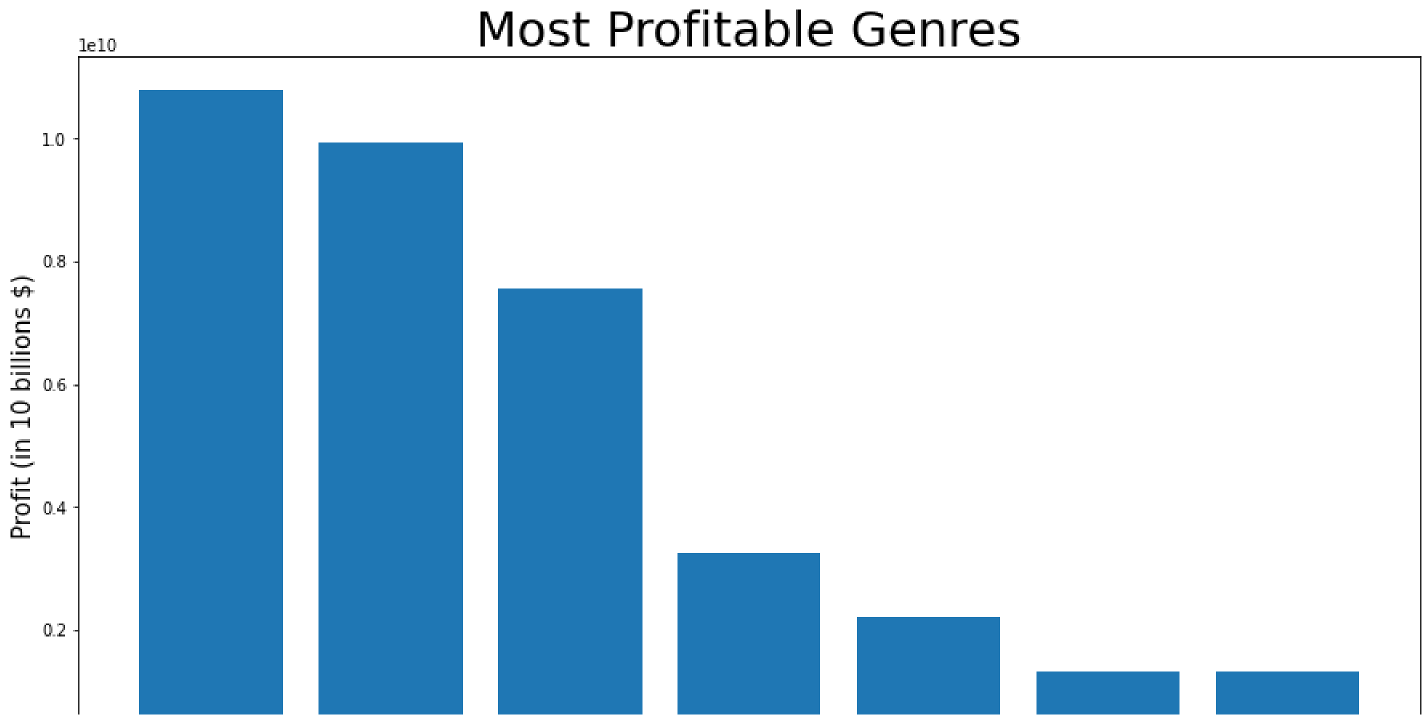
Out[31]:

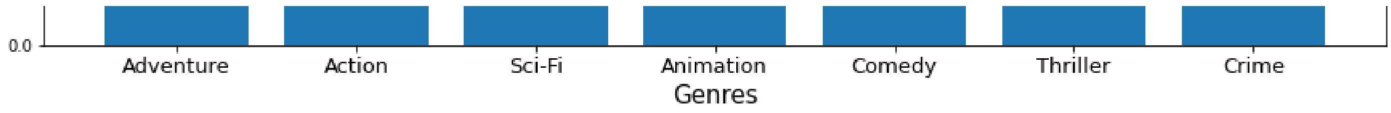
```
array([1.08051531e+10, 9.92506983e+09, 7.55382632e+09, 3.25132679e+09,
       2.20880608e+09, 1.32872279e+09, 1.32872279e+09])
```

In [32]:

```
#Create bar chart showing the most profitable genres from the top 10 most profitable movi
es
x = genre_profits_sorted.index
y = genre_profits_sorted.values

fig, ax = plt.subplots(figsize=(15,8))

ax.bar(x,y)

ax.set_title('Most Profitable Genres', fontsize=30)
ax.set_xlabel('Genres', fontsize=15)
ax.set_ylabel('Profit (in 10 billions $)', fontsize=15)

ax.set_xticklabels(x.str.replace(',', '\n'))
ax.tick_params(axis='x', labelsize=13)
```

```
<ipython-input-32-1368149a0d9d>:13: UserWarning: FixedFormatter should only be used toget
her with FixedLocator
  ax.set_xticklabels(x.str.replace(',', '\n'))
```





The data above tells us how profitable each genre is under the Top 10 Worldwide Profitable Movies bar chart. This is important because it tells us which genres have the best chance at making a profit for movie studios.

13. Find the Top 10 Actors/Actresses

Import Data

In [33]:

```
import sqlite3
import pandas as pd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [34]:

```
bom_movie_gross = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')
tmbd = pd.read_csv('data/zippedData/tmdb.movies.csv.gz')
title_crew = pd.read_csv('data/zippedData/imdb.title.crew.csv.gz')
name_basics = pd.read_csv('data/zippedData/imdb.name.basics.csv.gz')
title_p =pd.read_csv('data/zippedData/imdb.title.principals.csv.gz')
title_b =pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
movies =pd.read_csv('data/zippedData/tmdb.movies.csv.gz')
ratings = pd.read_csv('data/zippedData/imdb.title.ratings.csv.gz')
```

Merging title principles and name basics into a single column.

In [35]:

```
principals_and_names = pd.merge(
    title_p,
    name_basics,
    how='inner',
    on ='nconst')
principals_and_names.head(30)
```

Out[35]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]	Tommy Dysart	NaN	NaN
1	tt0111414	2	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN
2	tt5573596	5	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN
3	tt0111414	3	nm3739909	producer	producer	NaN	Barry Porter-Robinson	NaN	NaN
4	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN
5	tt2081348	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN
6	tt1414378	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN
7	tt2712990	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN
8	tt2395207	9	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	
9	tt4566480	9	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN	
10	tt8556530	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.0	NaN	
11	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]	Brittania Nicol	NaN	NaN	
12	tt0323808	2	nm2694680	actor	NaN	["Steve Thomson"]	Henry Garrett	NaN	NaN	
13	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.0	NaN	
14	tt1680140	2	nm0574615	actor	NaN	["Bully"]	Graham McTavish	1961.0	NaN	
15	tt3072876	4	nm0574615	self	NaN	["Himself"]	Graham McTavish	1961.0	NaN	
16	tt7168262	1	nm0574615	actor	NaN	["Ibrahim Kozlov"]	Graham McTavish	1961.0	NaN	
17	tt9013026	1	nm0574615	actor	NaN	["Mallory"]	Graham McTavish	1961.0	NaN	
18	tt0323808	4	nm0502652	actress	NaN	["Lady Delia Morrison"]	Jacqueline Leonard	1967.0	NaN	
19	tt0323808	5	nm0362736	director	NaN	NaN	Robin Hardy	1929.0	2016.0	
20	tt0323808	6	nm0811056	producer	producer	NaN	Peter Snell	1938.0	NaN	
21	tt10334148	10	nm0811056	producer	producer	NaN	Peter Snell	1938.0	NaN	
22	tt0323808	7	nm0914939	producer	producer	NaN	Peter Watson-Wood	NaN	NaN	
23	tt0323808	8	nm0779346	composer	NaN	NaN	John Scott	1930.0	NaN	
24	tt2298564	9	nm0779346	self	NaN	["Himself"]	John Scott	1930.0	NaN	
25	tt4663524	10	nm0779346	composer	NaN	NaN	John Scott	1930.0	NaN	
26	tt6711128	2	nm0779346	self	NaN	["Himself"]	John Scott	1930.0	NaN	
27	tt0323808	9	nm0676104	cinematographer	NaN	NaN	Jan Pester	NaN	NaN	came
28	tt2379402	8	nm0676104	cinematographer	NaN	NaN	Jan Pester	NaN	NaN	came
29	tt6263324	9	nm0676104	cinematographer	director of photography	NaN	Jan Pester	NaN	NaN	came

Filter out the professions that are not actors.

In [36]:

```
actors = principals_and_names[principals_and_names["primary_profession"].str.contains('ac
tor|actress', na=False)]
```

In [37]:

```
actors.head()
```

Out[37]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	primary_profession
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]	Tommy Dysart	NaN	NaN	actor
1	tt0111414	2	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN	actor,writer,producer
2	tt5573596	5	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN	actor,writer,producer
11	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]	Brittania Nicol	NaN	NaN	actress,soundtrack

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	primary_profession
12	tt0323808	2	nm2694680	actor	NaN	["Steve Thomson"]	Henry Garrett	NaN	NaN	actor

Filter out birth years greater than 1939

In [38]:

```
actors = actors[actors['birth_year'] > 1939]
actors.sort_values(by=['birth_year'])
```

Out[38]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	prim
108183	tt2375005	9	nm0000448	self	NaN	["Himself"]	Lance Henriksen	1940.0	NaN	actor
108179	tt1935072	3	nm0000448	self	NaN	["Himself"]	Lance Henriksen	1940.0	NaN	actor
108178	tt1384961	2	nm0000448	actor	NaN	["Mulciber"]	Lance Henriksen	1940.0	NaN	actor
108177	tt1528813	1	nm0000448	actor	NaN	["Mr. Darnell"]	Lance Henriksen	1940.0	NaN	actor
108176	tt1558258	4	nm0000448	actor	NaN	["Father Reed"]	Lance Henriksen	1940.0	NaN	actor
...	...	...	...	...	...	...	...	...	...	
656421	tt4100182	2	nm6446418	actor	NaN	["Guyus"]	Gaius Lee DuPree	2013.0	NaN	actor,ci
106804	tt10360096	3	nm10708650	actress	NaN	["Katie"]	Nayana Niter	2014.0	NaN	
942469	tt9396174	1	nm10306475	actress	NaN	["Monica"]	Natalye Archiles	2014.0	NaN	
942468	tt9392402	1	nm10306475	actress	NaN	["Buny"]	Natalye Archiles	2014.0	NaN	
876196	tt6023560	3	nm8405397	archive_footage	NaN	["Herself"]	Indiana Feek	2014.0	NaN	

154769 rows × 11 columns

Merge actors and their individual ratings.

In [39]:

```
actors_and_ratings = pd.merge(
    actors,
    ratings,
    how='inner',
    on ='tconst')
actors_and_ratings
```

Out[39]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	prin
0	tt5573596	5	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN	actor,
1	tt5573596	4	nm0000476	actress	NaN	NaN	Sally Kirkland	1941.0	NaN	actress,produce
2	tt5573596	3	nm0121007	actor	NaN	NaN	Eric Burdon	1941.0	NaN	soundtrack,
3	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.0	NaN	actor,sou
4	tt0323808	4	nm0502652	actress	NaN	["Lady Delia Morrison"]	Jacqueline Leonard	1967.0	NaN	

...	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	prin
121526	tt6798460	4	nm8214951	actress	NaN	["Verka"]	Elizabet Nenova	1994.0	NaN	
121527	tt6798460	8	nm7094029	actor	NaN	["Commissar Yagoda"]	Soufiane El Khalidy	1987.0	NaN	acto
121528	tt7808528	1	nm1190444	self	NaN	["Himself"]	Dikembe Mutombo	1966.0	NaN	
121529	tt7808528	2	nm0645927	self	NaN	["Himself"]	Hakeem Olajuwon	1963.0	NaN	
121530	tt9670776	4	nm1625509	actor	NaN	["Petras Klimas"]	Rolandas Kazlas	1969.0	NaN	

121531 rows × 13 columns

Clean the actors and ratings columns.

In [40]:

```
actors_and_ratings.columns.str.replace(' ', '')
actors_and_ratings.head()
```

Out[40]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	primary_pro
0	tt5573596	5	nm0398271	director	NaN	NaN	Frank Howson	1952.0	NaN	actor,writer,pr
1	tt5573596	4	nm0000476	actress	NaN	NaN	Sally Kirkland	1941.0	NaN	actress,producer,miscell
2	tt5573596	3	nm0121007	actor	NaN	NaN	Eric Burdon	1941.0	NaN	soundtrack,actor,co
3	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.0	NaN	actor,soundtrack,
4	tt0323808	4	nm0502652	actress	NaN	["Lady Delia Morrison"]	Jacqueline Leonard	1967.0	NaN	

Drop the columns that are not needed.

In [41]:

```
actors_and_ratings = actors_and_ratings.drop(['ordering', 'job', 'characters', 'death_yea
r', 'known_for_titles'], axis=1)
actors_and_ratings
```

Out[41]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating	numvotes
0	tt5573596	nm0398271	director	Frank Howson	1952.0	actor,writer,producer	7.8	6
1	tt5573596	nm0000476	actress	Sally Kirkland	1941.0	actress,producer,miscellaneous	7.8	6
2	tt5573596	nm0121007	actor	Eric Burdon	1941.0	soundtrack,actor,composer	7.8	6
3	tt0323808	nm0574615	actor	Graham McTavish	1961.0	actor,soundtrack,director	3.9	2328
4	tt0323808	nm0502652	actress	Jacqueline Leonard	1967.0	actress	3.9	2328
...	...	...	...	...	...	...	...	...
121526	tt6798460	nm8214951	actress	Elizabet Nenova	1994.0	actress	7.9	14

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating	numvotes
121527	tt6798460	nm7094029	actor	Soufiane El Khalidy	1987.0	actor,writer,director	7.9	14
121528	tt7808528	nm1190444	self	Dikembe Mutombo	1966.0	actor	9.2	26
121529	tt7808528	nm0645927	self	Hakeem Olajuwon	1963.0	actor	9.2	26
121530	tt9670776	nm1625509	actor	Rolandas Kazlas	1969.0	actor	8.0	9

121531 rows × 8 columns

Filter out rows that are do not have "actor" or "actress" in "category" column.

In [42]:

```
actors_and_ratings = actors_and_ratings[actors_and_ratings["category"].str.contains('actor|actress', na=False)]
actors_and_ratings
```

Out[42]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating	numvotes
1	tt5573596	nm0000476	actress	Sally Kirkland	1941.0	actress,producer,miscellaneous	7.8	6
2	tt5573596	nm0121007	actor	Eric Burdon	1941.0	soundtrack,actor,composer	7.8	6
3	tt0323808	nm0574615	actor	Graham McTavish	1961.0	actor,soundtrack,director	3.9	2328
4	tt0323808	nm0502652	actress	Jacqueline Leonard	1967.0	actress	3.9	2328
5	tt1680140	nm0574615	actor	Graham McTavish	1961.0	actor,soundtrack,director	5.1	777
...	...	...	...	...	...	...	...	...
121523	tt7935646	nm7487478	actress	Zeynep Tugçe Bayat	1990.0	actress,soundtrack	5.0	473
121524	tt8178850	nm0596670	actress	Claudia Molina	1985.0	actress,soundtrack	3.5	30
121526	tt6798460	nm8214951	actress	Elizabet Nenova	1994.0	actress	7.9	14
121527	tt6798460	nm7094029	actor	Soufiane El Khalidy	1987.0	actor,writer,director	7.9	14
121530	tt9670776	nm1625509	actor	Rolandas Kazlas	1969.0	actor	8.0	9

97817 rows × 8 columns

1. **Making a new variable that stores average rating mutiplied by their votes.**
2. **Sort values by ascending.**

In [43]:

```
actors_and_ratings['popularity_&_rating'] = actors_and_ratings['averagerating'] * actors_and_ratings['numvotes']
actors_and_ratings.sort_values(by=['popularity_&_rating'],ascending=False)
```

```
<ipython-input-43-b3585c45e334>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  actors_and_ratings['popularity_&_rating'] = actors_and_ratings['averagerating'] * actors_and_ratings['numvotes']
```

Out[43]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating	numvote
52206	tt1375666	nm0330687	actor	Joseph Gordon-Levitt	1981.0	actor,producer,soundtrack	8.8	184106
52203	tt1375666	nm0913822	actor	Ken Watanabe	1959.0	actor,producer,director	8.8	184106
52204	tt1375666	nm0000138	actor	Leonardo DiCaprio	1974.0	actor,producer,writer	8.8	184106
52205	tt1375666	nm0680983	actress	Ellen Page	1987.0	actress,producer,soundtrack	8.8	184106
4677	tt1345836	nm0000198	actor	Gary Oldman	1958.0	actor,soundtrack,producer	8.4	138776
...	...	...	...	...	...	...	...	..
110611	tt8446392	nm0129648	actress	Nathalie Caldonazzo	1969.0	actress	1.0	
112502	tt6792126	nm2511348	actress	Anna Kulovaná	1981.0	actress	1.0	
112499	tt6792126	nm0603184	actress	Dana Morávková	1971.0	actress,soundtrack,miscellaneous	1.0	
112501	tt6792126	nm1551077	actor	Ladislav Ondrej	1991.0	actor	1.0	
112500	tt6792126	nm4153825	actor	Petr Batek	1972.0	actor	1.0	

97817 rows × 9 columns

Drop columns that are not needed.

In [44]:

```
simple_actors_and_ratings = actors_and_ratings.drop(['tconst','nconst','category','birth_
year','primary_profession','averagerating','numvotes'], axis=1)
simple_actors_and_ratings.sort_values(by=['popularity_&_rating'],ascending=False)
```

Out[44]:

	primary_name	popularity_&_rating
52206	Joseph Gordon-Levitt	16201380.8
52203	Ken Watanabe	16201380.8
52204	Leonardo DiCaprio	16201380.8
52205	Ellen Page	16201380.8
4677	Gary Oldman	11657259.6
...	...	...
110611	Nathalie Caldonazzo	5.0
112502	Anna Kulovaná	5.0
112499	Dana Morávková	5.0
112501	Ladislav Ondrej	5.0
112500	Petr Batek	5.0

97817 rows × 2 columns

1. **Group primary name,mean, and count.**
2. **Find the mean of the popularity and reviews.**

In [45]:

```
actors_ranking = simple_actors_and_ratings.groupby(by=['primary_name']).agg(['mean','count'],as_index=False)
actors_ranking = actors_ranking.rename(columns={'mean':'pop_and_reviews','count':'count'})
actors_ranking.columns = actors_ranking.columns.droplevel(0)
actors_ranking = actors_ranking.sort_values(by=['pop_and_reviews'],ascending=False)
actors_ranking
```

Out[45]:

	pop_and_reviews	count
primary_name		
Leonardo DiCaprio	5.199703e+06	10
Idina Menzel	3.877485e+06	1
Robert Downey Jr.	3.751984e+06	13
Lucy Davis	3.656452e+06	1
Carrie Fisher	3.286611e+06	1
...	...	...
Yûichirô Hirose	6.000000e+00	1
Ladislav Ondrej	5.000000e+00	1
Anna Kulovaná	5.000000e+00	1
Petr Batek	5.000000e+00	1
Nathalie Caldonazzo	5.000000e+00	1

34127 rows × 2 columns

In [46]:

```
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

List out the top 10

In [47]:

```
top_10 = actors_ranking.head(10)
top_10
```

Out[47]:

	pop_and_reviews	count
primary_name		
Leonardo DiCaprio	5199703.07	10
Idina Menzel	3877485.00	1
Robert Downey Jr.	3751983.99	13
Lucy Davis	3656452.50	1
Carrie Fisher	3286611.30	1
Daniel Kaluuya	3083649.80	1
Tom Hardy	2985223.97	13
Donna Murphy	2857654.80	1
Ben Hardy	2763728.00	1
Orto Ignatiussen	2737371.35	2

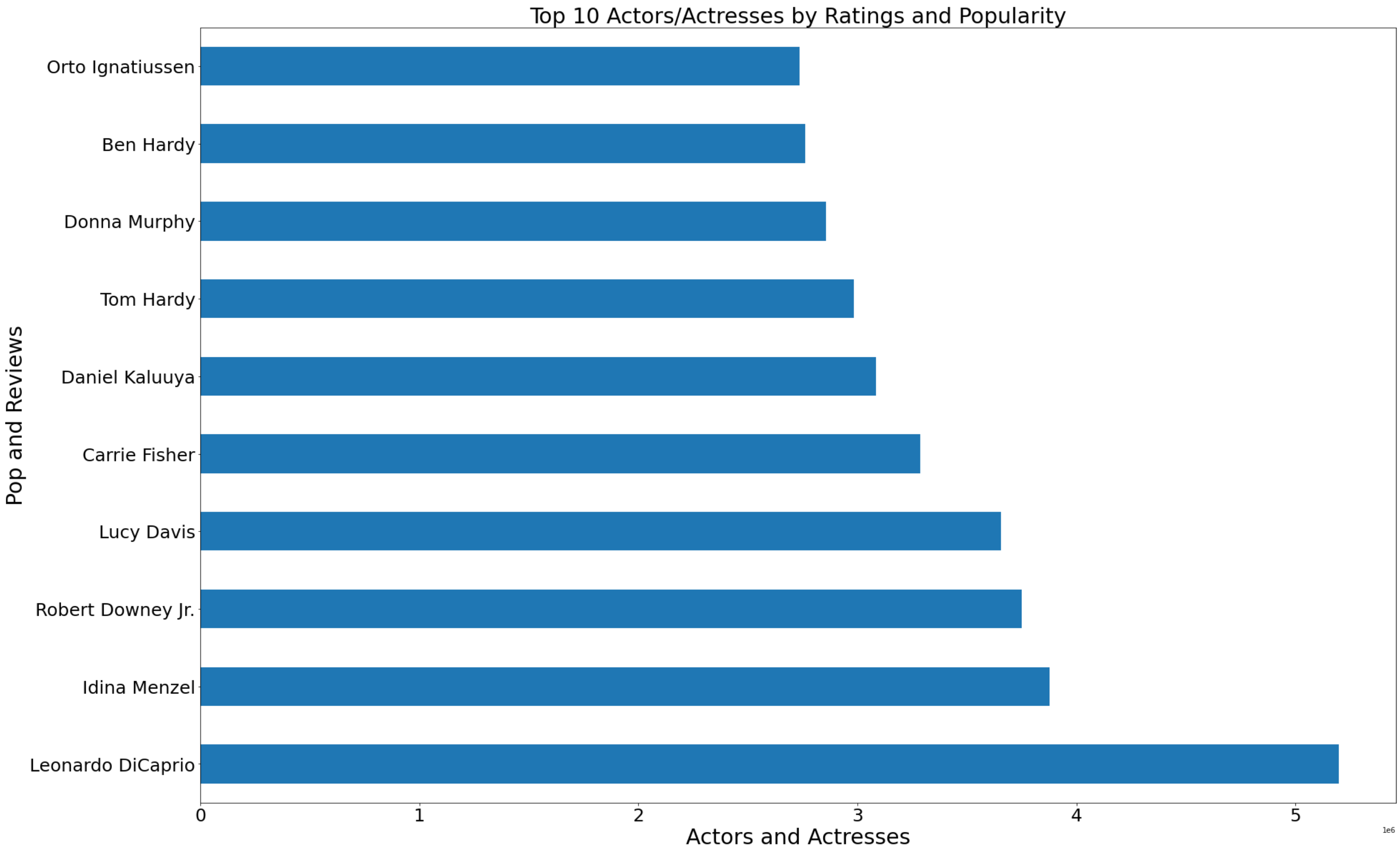
Plot Data

In [48]:

```
from matplotlib import pyplot as plt
top_10['pop_and_reviews'].plot(kind="barh",figsize=(30, 20),fontsize=25)
plt.title("Top 10 Actors/Actresses by Ratings and Popularity",fontsize=30)
plt.ylabel("Pop and Reviews",fontsize=30)
plt.xlabel("Actors and Actresses",fontsize=30)

plt.show
```

Out[48]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



14. Find the Top 10 Directors

Import Data

In [49]:

```
#import data then merge to dataframes
bom_movie_gross = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')
tmbd = pd.read_csv('data/zippedData/tmdb.movies.csv.gz')
title_crew = pd.read_csv('data/zippedData/imdb.title.crew.csv.gz')
name_basics = pd.read_csv('data/zippedData/imdb.name.basics.csv.gz')
princi_title=pd.read_csv('data/zippedData/imdb.title.principals.csv.gz')
title_b =pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
movies =pd.read_csv('data/zippedData/tmdb.movies.csv.gz')
ratings = pd.read_csv('data/zippedData/imdb.title.ratings.csv.gz')
```

Merge title principles and name basics.

In [50]:

```
names_descript = pd.merge(
    princi_title,
    name_basics,
    how='inner',
    on ='nconst')
names_descript.head(30)
```

Out[50]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]	Tommy Dysart	nan	nan	
1	tt0111414	2	nm0398271	director	NaN	NaN	Frank Howson	1952.00	nan	
2	tt5573596	5	nm0398271	director	NaN	NaN	Frank Howson	1952.00	nan	
3	tt0111414	3	nm3739909	producer	producer	NaN	Barry Porter-Robinson	nan	nan	
4	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
5	tt2081348	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
6	tt1414378	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
7	tt2712990	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
8	tt2395207	9	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
9	tt4566480	9	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
10	tt8556530	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	
11	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]	Brittania Nicol	nan	nan	
12	tt0323808	2	nm2694680	actor	NaN	["Steve Thomson"]	Henry Garrett	nan	nan	
13	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.00	nan	
14	tt1680140	2	nm0574615	actor	NaN	["Bully"]	Graham McTavish	1961.00	nan	
15	tt3072876	4	nm0574615	self	NaN	["Himself"]	Graham McTavish	1961.00	nan	
16	tt7168262	1	nm0574615	actor	NaN	["Ibrahim Kozlov"]	Graham McTavish	1961.00	nan	
17	tt9013026	1	nm0574615	actor	NaN	["Mallory"]	Graham McTavish	1961.00	nan	
18	tt0323808	4	nm0502652	actress	NaN	["Lady Delia Morrison"]	Jacqueline Leonard	1967.00	nan	
19	tt0323808	5	nm0362736	director	NaN	NaN	Robin Hardy	1929.00	2016.00	
20	tt0323808	6	nm0811056	producer	producer	NaN	Peter Snell	1938.00	nan	
21	tt10334148	10	nm0811056	producer	producer	NaN	Peter Snell	1938.00	nan	
22	tt0323808	7	nm0914939	producer	producer	NaN	Peter Watson-Wood	nan	nan	
23	tt0323808	8	nm0779346	composer	NaN	NaN	John Scott	1930.00	nan	
24	tt2298564	9	nm0779346	self	NaN	["Himself"]	John Scott	1930.00	nan	
25	tt4663524	10	nm0779346	composer	NaN	NaN	John Scott	1930.00	nan	
26	tt6711128	2	nm0779346	self	NaN	["Himself"]	John Scott	1930.00	nan	
27	tt0323808	9	nm0676104	cinematographer	NaN	NaN	Jan Pester	nan	nan	came
28	tt2379402	8	nm0676104	cinematographer	NaN	NaN	Jan Pester	nan	nan	came
29	tt6263324	9	nm0676104	cinematographer	director of photography	NaN	Jan Pester	nan	nan	came

Filter the data so that it shows the primary profession and only director.

In [51]:

```
directors = names_descript[names_descript["primary_profession"].str.contains('director|di
rector', na=False)]
```

In [52]:

```
directors.head()
```

Out[52]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	
4	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
5	tt2081348	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
6	tt1414378	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
7	tt2712990	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
8	tt2395207	9	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme

Filter the directors by birthday greater than 1939.

In [53]:

```
directors = directors[directors['birth_year'] > 1939]
directors.sort_values(by=['birth_year'])
```

Out[53]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	primary
47365	tt1706417	2	nm0000783	self	NaN	["Himself"]	Dario Argento	1940.00	nan	writer,direct
5208	tt3315342	2	nm0001772	actor	NaN	["Charles"]	Patrick Stewart	1940.00	nan	actor,produ
5209	tt6108612	1	nm0001772	actor	NaN	["Narrator"]	Patrick Stewart	1940.00	nan	actor,produ
5210	tt5066056	3	nm0001772	actor	NaN	["Harold"]	Patrick Stewart	1940.00	nan	actor,produ
5211	tt5610626	1	nm0001772	actor	NaN	["Drago"]	Patrick Stewart	1940.00	nan	actor,produ
...	...	...	...	...	...	...	...	...	...	
750831	tt5709892	5	nm8395992	producer	producer	NaN	Andrey Gromov	2004.00	nan	producer,ac
750833	tt6017238	3	nm8395992	actor	NaN	["Alexandr Zlovredniy"]	Andrey Gromov	2004.00	nan	producer,ac
750834	tt4980576	3	nm8395992	actor	NaN	["Andrey"]	Andrey Gromov	2004.00	nan	producer,ac
750832	tt6018006	2	nm8395992	actor	NaN	["Andrey"]	Andrey Gromov	2004.00	nan	producer,ac
849144	tt5278522	1	nm2171994	actor	NaN	["The Kid"]	Alec Coulouris	2005.00	nan	actor,assist

68969 rows × 11 columns

Merge the directors and ratings.

In [54]:

```
names_descript = pd.merge(
    directors,
    ratings,
    how='inner',
    on ='tconst')
names_descript
```

Out[54]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	
0	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_c
1	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.00	nan	
2	tt2081348	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_c
3	tt2081348	2	nm0803397	actor	NaN	["Paul"]	Jamie Sives	1973.00	nan	
4	tt1414378	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_c
...	...	...	...	...	...	...	...	...	...	
52154	tt8037610	10	nm0787509	actress	NaN	NaN	Renuka Shahane	1966.00	nan	
52155	tt6798460	8	nm7094029	actor	NaN	["Commissar Yagoda"]	Soufiane El Khalidy	1987.00	nan	
52156	tt8328740	5	nm0466901	director	NaN	NaN	Kitarô Kôsaka	1962.00	nan	art_department,a
52157	tt8861786	3	nm1035160	actor	NaN	["Dog Gnarly"]	Tyreese Burnett	1975.00	nan	
52158	tt9557190	3	nm1048560	director	NaN	NaN	Marcus Raboy	1965.00	nan	direct

52159 rows × 13 columns

Remove the columns in the rows.

In [55]:

```
names_descript.columns.str.replace(', ', '')
names_descript.head()
```

Out[55]:

	tconst	ordering	nconst	category	job	characters	primary_name	birth_year	death_year	
0	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
1	tt0323808	3	nm0574615	actor	NaN	["Sir Lachlan Morrison"]	Graham McTavish	1961.00	nan	acto
2	tt2081348	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme
3	tt2081348	2	nm0803397	actor	NaN	["Paul"]	Jamie Sives	1973.00	nan	ac
4	tt1414378	10	nm0059247	editor	NaN	NaN	Sean Barton	1944.00	nan	editor,editorial_departme

Drop all rows that are not needed.

In [56]:

```
names_descript= names_descript.drop(['ordering', 'job', 'characters', 'death_year', 'kno
wn_for_titles'], axis=1)
names_descript
```

Out[56]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating
0	tt0323808	nm0059247	editor	Sean Barton	1944.00	editor,editorial_department,assistant_director	3.9
1	tt0323808	nm0574615	actor	Graham McTavish	1961.00	actor,soundtrack,director	3.9
2	tt2081348	nm0059247	editor	Sean Barton	1944.00	editor,editorial_department,assistant_director	4.1
3	tt2081348	nm0803397	actor	Jamie Sives	1973.00	actor,producer,director	4.1
4	tt1414378	nm0059247	editor	Sean Barton	1944.00	editor,editorial_department,assistant_director	6.5
...	...	...	...	...	...	...	..
52154	tt8037610	nm0787509	actress	Renuka Shahane	1966.00	actress,writer,director	8.3
52155	tt6798460	nm7094029	actor	Soufiane El Khalidy	1987.00	actor,writer,director	7.9
52156	tt8328740	nm0466901	director	Kitarô Kôsaka	1962.00	art_department,animation_department,director	6.9
52157	tt8861786	nm1035160	actor	Tyreese Burnett	1975.00	actor,producer,art_director	4.4
52158	tt9557190	nm1048560	director	Marcus Raboy	1965.00	director,producer,cinematographer	6.2

52159 rows × 8 columns

Pull only the director profession from the category colunmn.

In [57]:

```
names_descript = names_descript[names_descript["category"].str.contains('director|directo
r', na=False)]
names_descript
```

Out[57]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating
5	tt1414378	nm0789054	director	Ian Sharp	1946.00	director,assistant_director,writer	6.5
11	tt1680140	nm0425894	director	Niall Johnson	1965.00	writer,director,producer	5.1
15	tt0417610	nm1145057	director	Alejandro Chomski	1968.00	director,writer,producer	6.4
17	tt1563675	nm0563760	director	Laura Mañá	1968.00	actress,director,writer	6.2
20	tt0426566	nm1163513	director	Julio Bove	1954.00	producer,director,actor	6.0
...	...	...	...	...	...	...	..
52147	tt9260454	nm1887409	director	Dani Rosenberg	1979.00	writer,director,producer	8.6
52150	tt7001792	nm0088392	director	Michael Blieden	1971.00	writer,director,actor	6.0
52153	tt7836394	nm9535026	director	Nico Baumbach	1986.00	director,cinematographer,editor	8.1
52156	tt8328740	nm0466901	director	Kitarô Kôsaka	1962.00	art_department,animation_department,director	6.9
52158	tt9557190	nm1048560	director	Marcus Raboy	1965.00	director,producer,cinematographer	6.2

19605 rows × 8 columns

1. **Making a new variable that stores average rating mutiplied by their votes.**
2. **Sort values by ascending.**

In [58]:

```
names_descript['Popularity_Ratings'] = names_descript['averagerating'] * names_descript['
numvotes']
```

```
names_descript.sort_values(by=['Popularity_Ratings'],ascending=False)
```

```
<ipython-input-58-52fb35f770a8>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  names_descript['Popularity_Ratings'] = names_descript['averagerating'] * names_descript
['numvotes']
```

Out[58]:

	tconst	nconst	category	primary_name	birth_year	primary_profession	averagerating	numvotes	Popularit
15350	tt1375666	nm0634240	director	Christopher Nolan	1970.00	writer,producer,director	8.80	1841066	16
30654	tt1345836	nm0634240	director	Christopher Nolan	1970.00	writer,producer,director	8.40	1387769	11
30659	tt0816692	nm0634240	director	Christopher Nolan	1970.00	writer,producer,director	8.60	1299334	11
2401	tt0848228	nm0923736	director	Joss Whedon	1964.00	writer,producer,director	8.10	1183655	9
11986	tt0993846	nm0000217	director	Martin Scorsese	1942.00	producer,director,actor	8.20	1035358	8
...	...	...	...	...	...	...	...	...	
44039	tt5364390	nm0158427	director	Derek Chiu	1961.00	director,writer,producer	2.00	5	
12417	tt5935758	nm3062127	director	Geno McGahee	1974.00	writer,producer,director	2.00	5	
45700	tt8476266	nm2508880	director	Ján Novák	1966.00	director,writer,editor	1.50	6	
46010	tt6153116	nm1512437	director	Maria Ramos	1964.00	director,writer,producer	1.00	7	
45857	tt6792126	nm6008960	director	Eva Toulová	1990.00	director,writer,producer	1.00	5	

19605 rows × 9 columns

Drop columns that are not needed then sort the popularity and rating to ascending.

In [59]:

```
director_ratings = names_descript.drop(['tconst','nconst','category','birth_year','primar
y_profession','averagerating','numvotes',], axis=1)
director_ratings.sort_values(by=['Popularity_Ratings'],ascending=False)
```

Out[59]:

	primary_name	Popularity_Ratings
15350	Christopher Nolan	16201380.80
30654	Christopher Nolan	11657259.60
30659	Christopher Nolan	11174272.40
2401	Joss Whedon	9587605.50
11986	Martin Scorsese	8489935.60
...	...	...
44039	Derek Chiu	10.00
12417	Geno McGahee	10.00
45700	Ján Novák	9.00
46010	Maria Ramos	7.00
45857	Eva Toulová	5.00

19605 rows × 2 columns

1. **Group primary name,mean, and count.**
2. **Find the mean of the popularity and reviews.**

In [60]:

```
directors_rank = director_ratings.groupby(by=['primary_name']).agg(['mean','count'],as_i
ndex=False)
directors_rank = directors_rank.rename(columns={'mean':'Popularity_Ratings','count':'cou
nt'})
directors_rank.columns = directors_rank.columns.droplevel(0)
directors_rank = directors_rank.sort_values(by=['Popularity_Ratings'],ascending=False)
directors_rank
```

Out[60]:

	Popularity_Ratings	count
primary_name		
Christopher Nolan	10679723.70	4
Joss Whedon	4852476.57	3
Anthony Russo	4833244.80	4
Joe Russo	4833244.80	4
David Fincher	4523356.60	3
...	...	...
Daisuke Yamanouchi	16.60	4
Henrik Normann	16.00	1
Ping Ho	13.20	1
Vlastimil Simunek	12.00	1
Grzegorz Lewandowski	11.20	1

10161 rows × 2 columns

In [61]:

```
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

List out the top 10 directors inside of variable called top_10_dir

In [62]:

```
top_10_dir = directors_rank.head(10)
top_10_dir
```

Out[62]:

	Popularity_Ratings	count
primary_name		
Christopher Nolan	10679723.70	4
Joss Whedon	4852476.57	3
Anthony Russo	4833244.80	4
Joe Russo	4833244.80	4
David Fincher	4523356.60	3
James Gunn	3960186.50	3
Chris Buck	3877485.00	1
Patty Jenkins	3656452.50	1

primary_name	Popularity_Ratings	count
Patty Jenkins	[illegible]	[illegible]
Matthew Vaughn	3557716.60	4
Sam Mendes	3508175.50	2

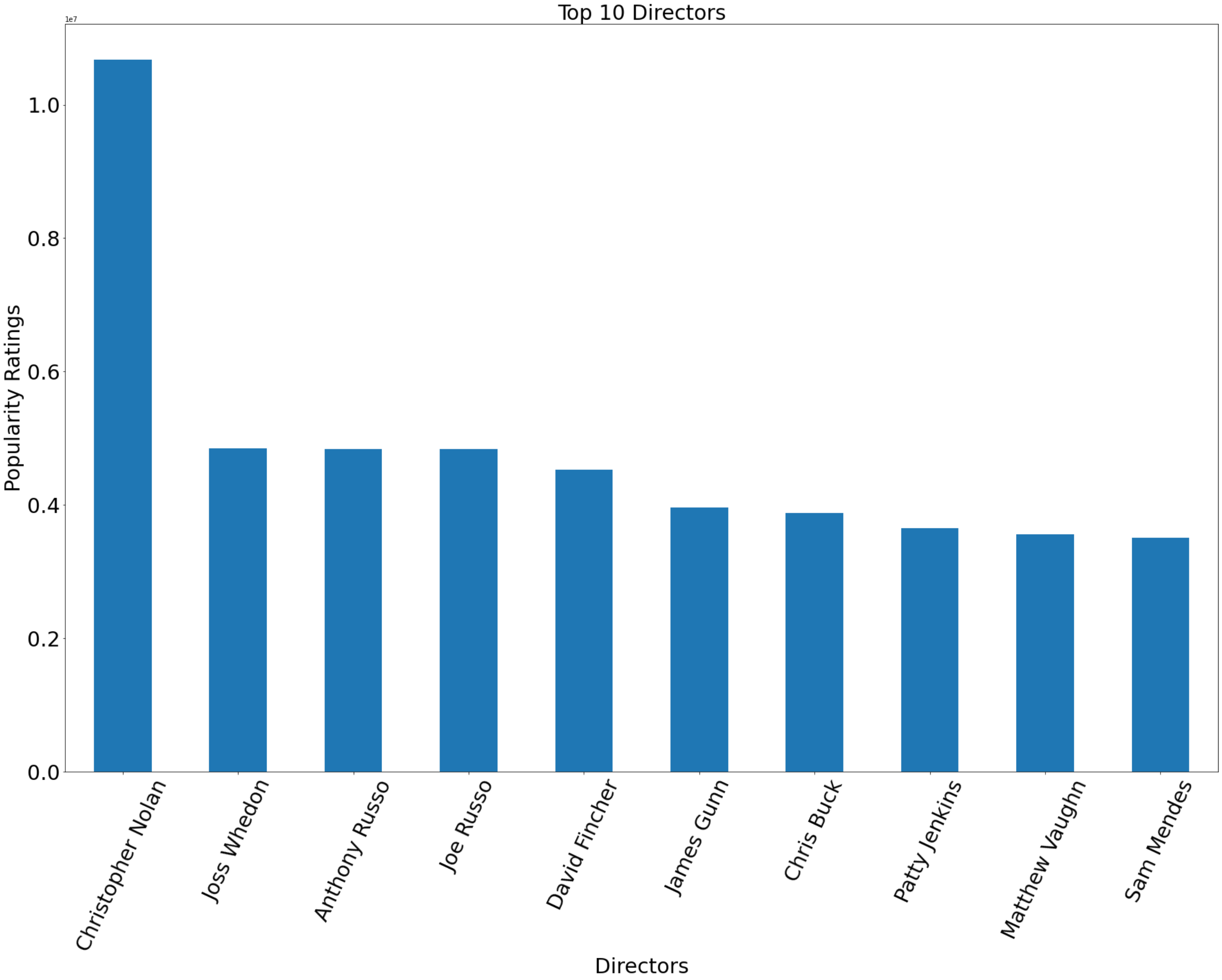
Plot Data

In [63]:

```
from matplotlib import pyplot as plt
top_10_dir['Popularity_Ratings'].plot(kind="bar",figsize=(30, 20),fontsize=30)

plt.title("Top 10 Directors",fontsize=30)
plt.ylabel("Popularity Ratings",fontsize=30)
plt.xlabel("Directors",fontsize=30)
plt.xticks(rotation = 65)
plt.show
```

Out[63]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



15. Actors and Actresses in the Most Profitable Movies

This is a subplot with two plots. The first shows actors and the second shows actresses.

I looked at each table to make note of all available columns. I noticed the tconst and nconst columns that I counld use to merge all of the data together into one dataframe. After merging, I dropped many columns that I did not need. These included columns that I though contained duplicate information or information that would not be useful. Next, I dropped rows with null values in a subset of all columns. I used outer merges so there

where lots of rows with missing data. I wanted to be sure of the data that I had before getting rid of any. I also did not want to drop rows that were only missing data in either domestic or worldwide gross because I only needed one.

I then got just the rows where the profession was either actor or actress and made them their own dataframes. Then, I grouped them by the name of the person and summed the profits of all the movies they were in. Next, I sorted the people by the profit and selected the top 10. After that, I just plotted the top actors and actresses in the most profitable movies.

In [64]:

```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
```

In [65]:

```
#load data
nb = pd.read_csv('data/zippedData/imdb.name.basics.csv.gz')
takas = pd.read_csv('data/zippedData/imdb.title.akas.csv.gz')
tbase = pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
tbase = pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
tcrew = pd.read_csv('data/zippedData/imdb.title.crew.csv.gz')
tprinc = pd.read_csv('data/zippedData/imdb.title.principals.csv.gz')
trat = pd.read_csv('data/zippedData/imdb.title.ratings.csv.gz')
tn = pd.read_csv('data/zippedData/tn.movie_budgets.csv.gz')
```

In [66]:

```
#rename takas['title_id'] to takas['tconst'] to that we can use it to merge on
takas.rename(columns={'title_id':'tconst'},inplace=True)
#merge data
tcombo = trat.merge(tcrew,how='outer',on='tconst')
tcombo = tcombo.merge(tbase,how='outer',on='tconst')
tcombo = tcombo.merge(takas,how='outer',on='tconst')
tcombo = tcombo.merge(tprinc,how='outer',on='tconst')
tcombo = tcombo.merge(nb,how='outer',on='nconst')

#change tcombo['primary_title'] to movie
tcombo.rename(columns={'primary_title':'movie'},inplace=True)
#finish merging
tcombo = tcombo.merge(tn,how='outer',on='movie')
#drop some columns
tcombo.drop(['language','types','attributes','is_original_title','ordering_x','ordering_
y','original_title','numvotes','birth_year','death_year'],axis=1,inplace=True)
#drop null values from this subset of columns
tcombo.dropna(subset=['tconst','averagerating','directors','writers','movie','start_year
','runtime_minutes','genres','title','region','id','release_date','production_budget'],in
place=True)

#I had lots of duplicates so I dropped the dupes in tconst and movie columns
tcombo.drop_duplicates(subset=['tconst','movie'],inplace=True)

#change budget and gross types to int
tcombo['production_budget'] = tcombo['production_budget'].map(lambda x: x.replace('$',""
))
tcombo['production_budget'] = tcombo['production_budget'].map(lambda x: x.replace(',',""
))
tcombo['domestic_gross'] = tcombo['domestic_gross'].map(lambda x: x.replace('$',""))
tcombo['domestic_gross'] = tcombo['domestic_gross'].map(lambda x: x.replace(',',""))
tcombo['worldwide_gross'] = tcombo['worldwide_gross'].map(lambda x: x.replace('$',""))
tcombo['worldwide_gross'] = tcombo['worldwide_gross'].map(lambda x: x.replace(',',""))
tcombo.production_budget = tcombo.production_budget.astype(float)
tcombo.domestic_gross = tcombo.domestic_gross.astype(float)
tcombo.worldwide_gross = tcombo.worldwide_gross.astype(float)
```

```
#making a new column called release month
tcombo['release_month'] = tcombo['release_date'].map(lambda x: x[:3])

#make profit column
tcombo['profit'] = (tcombo['worldwide_gross'] - tcombo['production_budget'])

#drop rows in gross and budget that are zero
tcombo = tcombo.loc[((tcombo['domestic_gross'] != 0) | (tcombo['worldwide_gross'] != 0))
& (tcombo['production_budget'] != 0)]
tcombo.head()
```

Out[66]:

	tconst	averagerating	directors	writers	movie	start_year	runtime_minutes	g
470	tt4080386	6.50	nm1819881	nm6369175,nm3057599	Sardaar Ji	2015.00	141.00	Comedy,Fantasy,I
1782	tt3431016	5.10	nm0004541	nm1183861	The Messenger	2015.00	101.00	Drama,Mystery,T
1787	tt4984930	7.70	nm0753393	nm7556970,nm0753393	The Messenger	2015.00	89.00	Documentary,I
1788	tt8706988	5.80	nm0664582	nm0664582,nm2561798	The Messenger	2019.00	100.00	Drama,Thrille
1843	tt2176244	8.00	nm2613589	nm2613589	The Messenger	2012.00	74.00	I

5 rows × 24 columns

Above is a peak at the data I am using. Below I am getting only the data I need for professions, names, and profit.

In [67]:

```
#get a subset of the data
tcombo_profession_profit = tcombo[['directors','profit','primary_name','primary_professio
n']]
#split directors on the comma
tcombo_profession_profit[:]['directors'] = tcombo_profession_profit['directors'].str.spli
t(',')
#split the professions on the comma
tcombo_profession_profit[:]['primary_profession'] = tcombo_profession_profit['primary_pro
fession'].str.split(',')
```

In [68]:

```
#use .explode
tcombo_profession_profit = tcombo_profession_profit.explode('directors')

#replace spaces with new line so that the names look better on the graphs
tcombo_profession_profit.primary_name = tcombo_profession_profit.primary_name.map(lambda
x: x.replace(' ','\n'))

tcombo_profession_profit = tcombo_profession_profit.explode('primary_profession')
```

This is where I get all of the actors and actresses.

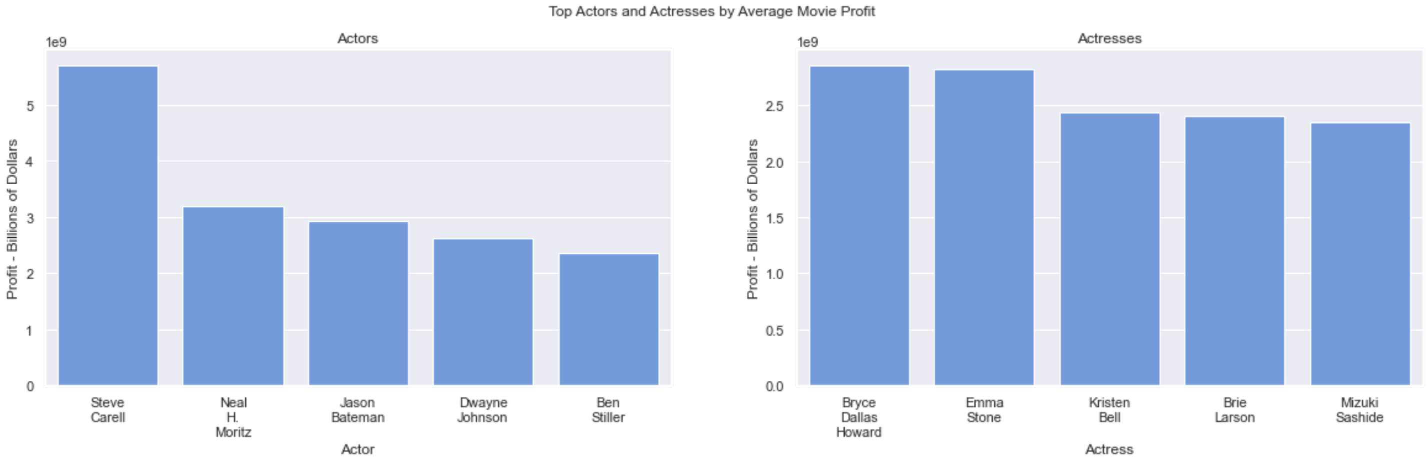
In [69]:

```
#find all the rows where the profession is actor
actor_group_sum = tcombo_profession_profit.loc[tcombo_profession_profit['primary_professi
on'] == 'actor']
actor_group_sum = actor_group_sum.groupby(['primary_name']).sum().sort_values(by='profit
',ascending=False)[:5]
#find all the rows where the proffession is actress
actress_group_sum = tcombo_profession_profit.loc[tcombo_profession_profit['primary_profes
sion'] == 'actress']
actress_group_sum = actress_group_sum.groupby(['primary_name']).sum().sort_values(by='profit',ascending=False)[:5]
```

Making the figure

In [70]:

```
fig, axes = plt.subplots(nrows=1,ncols=2,figsize=(20,5))
fig.suptitle('Top Actors and Actresses by Average Movie Profit')

sns.barplot(ax=axes[0],x=actor_group_sum.index,y='profit',data=actor_group_sum,color='cornflowerblue')
axes[0].set(xlabel='Actor',ylabel='Profit - Billions of Dollars',title='Actors')

sns.barplot(ax=axes[1],x=actress_group_sum.index,y='profit',data=actress_group_sum,color='cornflowerblue')
axes[1].set(xlabel='Actress',ylabel='Profit - Billions of Dollars',title='Actresses');
```

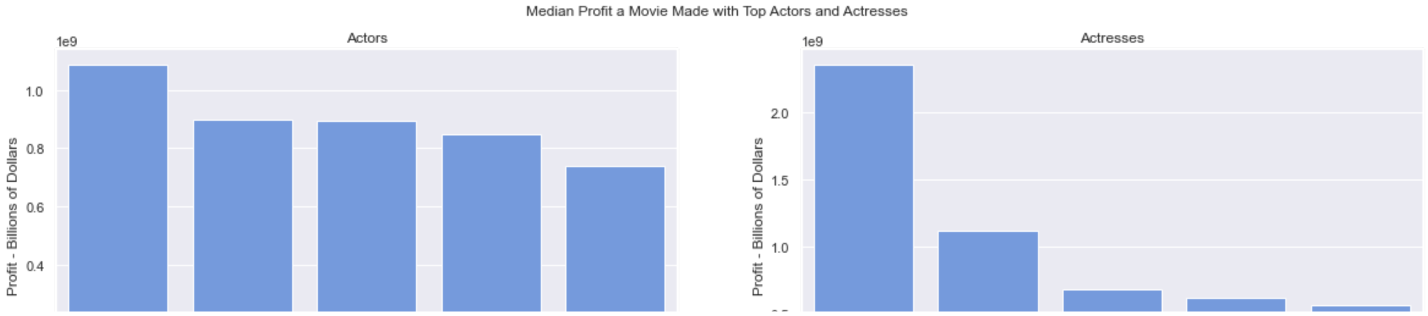


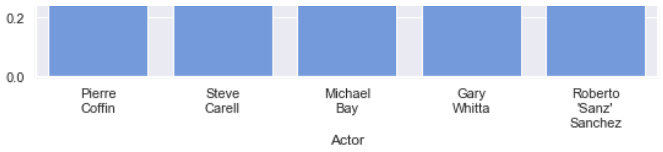
In [71]:

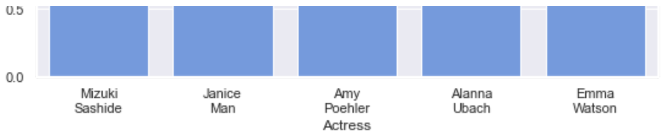
```
#find all the rows where the profession is actor
actor_group_median = tcombo_profession_profit.loc[tcombo_profession_profit['primary_profession'] == 'actor']
actor_group_median = actor_group_median.groupby(['primary_name']).median().sort_values(by='profit',ascending=False)[:5]
#find all the rows where the proffession is actress
actress_group_median = tcombo_profession_profit.loc[tcombo_profession_profit['primary_profession'] == 'actress']
actress_group_median = actress_group_median.groupby(['primary_name']).median().sort_values(by='profit',ascending=False)[:5]
```

In [72]:

```
fig, axes = plt.subplots(nrows=1,ncols=2,figsize=(20,5))
fig.suptitle('Median Profit a Movie Made with Top Actors and Actresses')

sns.barplot(ax=axes[0],x=actor_group_median.index,y='profit',data=actor_group_median,color='cornflowerblue')
axes[0].set(xlabel='Actor',ylabel='Profit - Billions of Dollars',title='Actors')

sns.barplot(ax=axes[1],x=actress_group_median.index,y='profit',data=actress_group_median,color='cornflowerblue')
axes[1].set(xlabel='Actress',ylabel='Profit - Billions of Dollars',title='Actresses');
```







16. Profit vs. Director

This code makes a barplot of the top directors for the most profitable movies. The first three cells will be the same for each plot because I used the same data for all of them.

I looked at each table to make note of all available columns. I noticed the tconst and nconst columns that I counld use to merge all of the data together into one dataframe. After merging, I dropped many columns that I did not need. These included columns that I though contained duplicate information or information that would not be useful. Next, I dropped rows with null values in a subset of all columns. I used outer merges so there where lots of rows with missing data. I wanted to be sure of the data that I had before getting rid of any. I also did not want to drop rows that were only missing data in either domestic or worldwide gross because I only needed one.

I'm finding the necessary columns.

In [73]:

```
#get a subset of the data
tcombo_profession_profit = tcombo[['directors','profit','primary_name','primary_profession']]
#split directors on the comma
tcombo_profession_profit[:]['directors'] = tcombo_profession_profit['directors'].str.split(',')
#split the professions on the comma
tcombo_profession_profit[:]['primary_profession'] = tcombo_profession_profit['primary_profession'].str.split(',')
```

Now I need to make sure that there is only one entry per row.

In [74]:

```
#use .explode
tcombo_profession_profit = tcombo_profession_profit.explode('directors')

#replace spaces with new line so that the names look better on the graphs
tcombo_profession_profit.primary_name = tcombo_profession_profit.primary_name.map(lambda x: x.replace(' ','\n'))

tcombo_profession_profit = tcombo_profession_profit.explode('primary_profession')
```

Get only the directors

In [75]:

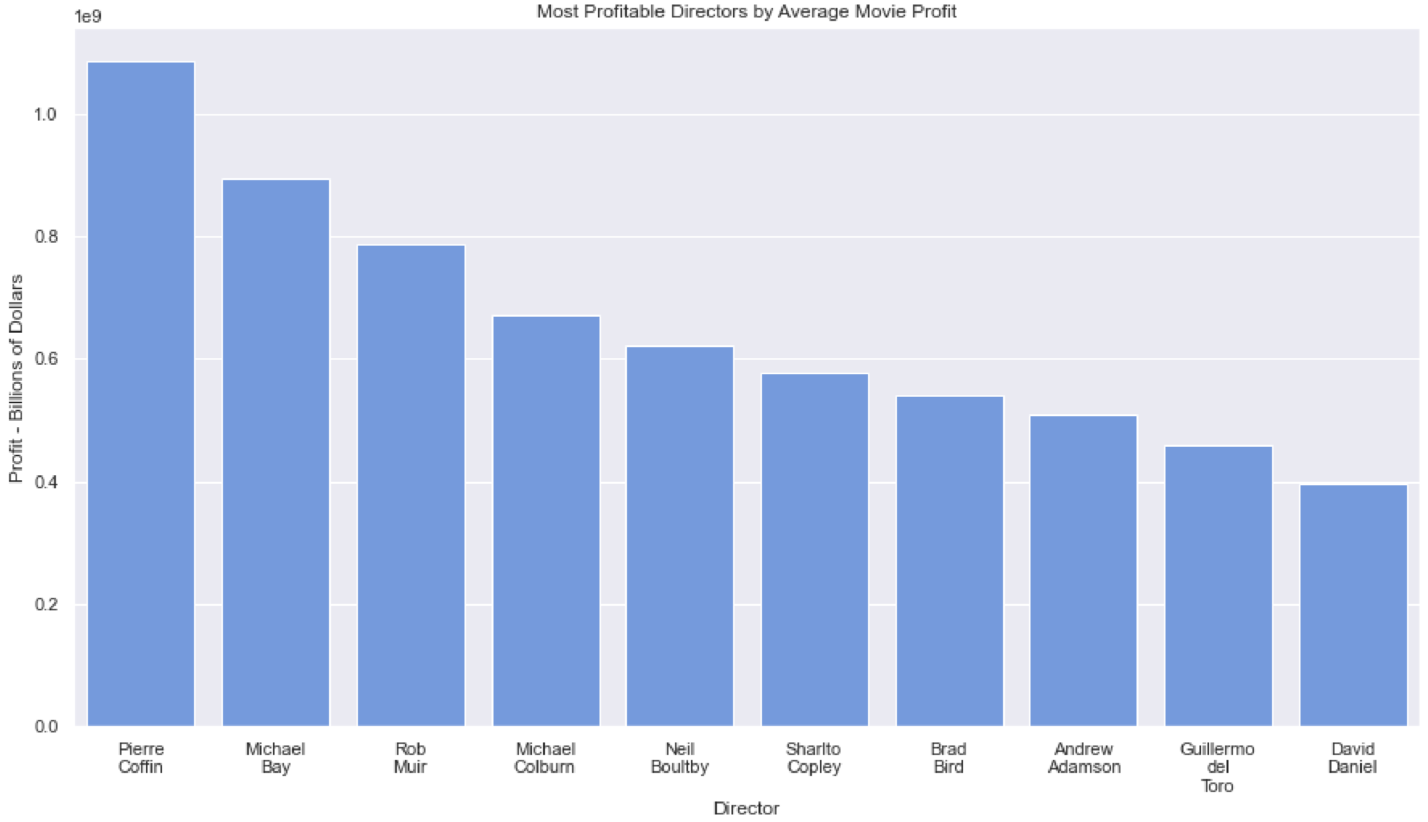
```
#find all rows where the primary profession is director
director_group_median = tcombo_profession_profit.loc[tcombo_profession_profit['primary_profession'] == 'director']
director_group_median = director_group_median.groupby(['primary_name']).median().sort_values(by='profit',ascending=False)[:10]
```

Make the plot

In [76]:

```
#director and profit
sns.set(rc={'figure.figsize':(15,8)})
ax = sns.barplot(x=director_group_median.index,y='profit',data=director_group_median,color='cornflowerblue')
```

```
#these are the directors who bring in the most money
ax.set(xlabel='Director',ylabel='Profit - Billions of Dollars',title='Most Profitable Directors by Average Movie Profit');
```



17. Profit vs Region

This is a bar plot of the regions where movies made the most money.

I looked at each table to make note of all available columns. I noticed the tconst and nconst columns that I counld use to merge all of the data together into one dataframe. After merging, I dropped many columns that I did not need. These included columns that I though contained duplicate information or information that would not be useful. Next, I dropped rows with null values in a subset of all columns. I used outer merges so there where lots of rows with missing data. I wanted to be sure of the data that I had before getting rid of any. I also did not want to drop rows that were only missing data in either domestic or worldwide gross because I only needed one.

I found a dictionary of country codes and their country names. I used it to map the country names in place of the codes in the region column. This makes the plot labels more meaningful.

Country Code Dictionary

I found a dictionary of country codes so I can map them and use the actual country name on the graph.

In [77]:

```
CC = {
    "AF": "AFGHANISTAN",
    "AX": "ÅLAND ISLANDS",
    "AL": "ALBANIA",
    "DZ": "ALGERIA",
    "AS": "AMERICAN SAMOA",
    "AD": "ANDORRA",
    "AO": "ANGOLA",
    "AI": "ANGUILLA",
    "AQ": "ANTARCTICA",
    "AG": "ANTIGUA AND BARBUDA",
    "AR": "ARGENTINA",
    "AM": "ARMENIA",
    "AW": "ARUBA",
    "AU": "AUSTRALIA",
    "AT": "AUSTRIA",
    "AZ": "AZERBAIJAN",
    "BS": "BAHAMAS",
    "BH": "BAHRAIN",
    "BD": "BANGLADESH",
    "BB": "BARBADOS",
    "BY": "BELARUS",
    "BE": "BELGIUM",
    "BZ": "BELIZE",
    "BJ": "BENIN",
    "BM": "BERMUDA",
    "BT": "BHUTAN",
    "BO": "BOLIVIA, PLURINATIONAL STATE OF",
    "BQ": "BONAIRE, SINT EUSTATIUS AND SABA",
    "BA": "BOSNIA AND HERZEGOVINA",
    "BW": "BOTSWANA",
    "BV": "BOUVET ISLAND",
    "BR": "BRAZIL",
    "IO": "BRITISH INDIAN OCEAN TERRITORY",
    "BN": "BRUNEI DARUSSALAM",
    "BG": "BULGARIA",
    "BF": "BURKINA FASO",
    "BI": "BURUNDI",
    "KH": "CAMBODIA",
    "CM": "CAMEROON",
    "CA": "CANADA",
    "CV": "CAPE VERDE",
    "KY": "CAYMAN ISLANDS",
    "CF": "CENTRAL AFRICAN REPUBLIC",
    "TD": "CHAD",
    "CL": "CHILE",
    "CN": "CHINA",
    "CX": "CHRISTMAS ISLAND",
    "CC": "COCOS (KEELING) ISLANDS",
    "CO": "COLOMBIA",
    "KM": "COMOROS",
    "CG": "CONGO",
    "CD": "CONGO, THE DEMOCRATIC REPUBLIC OF THE",
    "CK": "COOK ISLANDS",
    "CR": "COSTA RICA",
    "CI": "CÔTE D'IVOIRE",
    "HR": "CROATIA",
    "CU": "CUBA",
    "CW": "CURAÇAO",
    "CY": "CYPRUS",
    "CZ": "CZECH REPUBLIC",
    "DK": "DENMARK",
    "DJ": "DJIBOUTI",
    "DM": "DOMINICA",
    "DO": "DOMINICAN REPUBLIC",
    "EC": "ECUADOR",
    "EG": "EGYPT",
    "SV": "EL SALVADOR",
    "GQ": "EQUATORIAL GUINEA",
    "ER": "ERITREA",
    "EE": "ESTONIA",
    "ET": "ETHIOPIA",
    "FK": "FALKLAND ISLANDS (MALVINAS)",
    "FO": "FAROE ISLANDS",
    "FJ": "FIJI",
    "FI": "FINLAND",
    "FR": "FRANCE",
    "GF": "FRENCH GUIANA",
    "PF": "FRENCH POLYNESIA",
    "TF": "FRENCH SOUTHERN TERRITORIES",
    "GA": "GABON",
    "GM": "GAMBIA",
    "GE": "GEORGIA",
    "DE": "GERMANY",
    "GH": "GHANA",
    "GI": "GIBRALTAR",
    "GR": "GREECE",
    "GL": "GREENLAND",
    "GD": "GRENADA",
    "GP": "GUADELOUPE",
    "GU": "GUAM",
    "GT": "GUATEMALA",
    "GG": "GUERNSEY",
    "GN": "GUINEA",
    "GW": "GUINEA-BISSAU",
    "GY": "GUYANA",
    "HT": "HAITI",
    "HM": "HEARD ISLAND AND MCDONALD ISLANDS",
    "VA": "HOLY SEE (VATICAN CITY STATE)",
    "HN": "HONDURAS",
    "HK": "HONG KONG",
    "HU": "HUNGARY",
    "IS": "ICELAND",
    "IN": "INDIA",
    "ID": "INDONESIA",
    "IR": "IRAN, ISLAMIC REPUBLIC OF",
    "IQ": "IRAQ",
    "IE": "IRELAND",
    "IM": "ISLE OF MAN",
    "IL": "ISRAEL",
    "IT": "ITALY",
    "JM": "JAMAICA",
    "JP": "JAPAN",
    "JE": "JERSEY",
    "JO": "JORDAN",
    "KZ": "KAZAKHSTAN",
    "KE": "KENYA",
    "KI": "KIRIBATI",
    "KP": "KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF",
    "KR": "KOREA, REPUBLIC OF",
    "KW": "KUWAIT",
    "KG": "KYRGYZSTAN",
    "LA": "LAO PEOPLE'S DEMOCRATIC REPUBLIC",
    "LV": "LATVIA",
    "LB": "LEBANON",
    "LS": "LESOTHO",
    "LR": "LIBERIA",
    "LY": "LIBYA",
    "LI": "LIECHTENSTEIN",
    "LT": "LITHUANIA",
    "LU": "LUXEMBOURG",
    "MO": "MACAO",
    "MK": "MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF",
    "MG": "MADAGASCAR",
    "MW": "MALAWI",
    "MY": "MALAYSIA",
    "MV": "MALDIVES",
    "ML": "MALI",
    "MT": "MALTA",
    "MH": "MARSHALL ISLANDS",
    "MQ": "MARTINIQUE",
    "MR": "MAURITANIA",
    "MU": "MAURITIUS",
    "YT": "MAYOTTE",
    "MX": "MEXICO",
    "FM": "MICRONESIA, FEDERATED STATES OF",
    "MD": "MOLDOVA, REPUBLIC OF",
    "MC": "MONACO",
    "MN": "MONGOLIA",
    "ME": "MONTENEGRO",
    "MS": "MONTSERRAT",
    "MA": "MOROCCO",
    "MZ": "MOZAMBIQUE",
    "MM": "MYANMAR",
    "NA": "NAMIBIA",
    "NR": "NAURU",
    "NP": "NEPAL",
    "NL": "NETHERLANDS",
    "NC": "NEW CALEDONIA",
    "NZ": "NEW ZEALAND",
    "NI": "NICARAGUA",
    "NE": "NIGER",
    "NG": "NIGERIA",
    "NU": "NIUE",
    "NF": "NORFOLK ISLAND",
    "MP": "NORTHERN MARIANA ISLANDS",
    "NO": "NORWAY",
    "OM": "OMAN",
    "PK": "PAKISTAN",
    "PW": "PALAU",
    "PS": "PALESTINE, STATE OF",
    "PA": "PANAMA",
    "PG": "PAPUA NEW GUINEA",
    "PY": "PARAGUAY",
    "PE": "PERU",
    "PH": "PHILIPPINES",
    "PN": "PITCAIRN",
    "PL": "POLAND",
    "PT": "PORTUGAL",
    "PR": "PUERTO RICO",
    "QA": "QATAR",
    "RE": "RÉUNION",
    "RO": "ROMANIA",
    "RU": "RUSSIAN FEDERATION",
    "RW": "RWANDA",
    "BL": "SAINT BARTHÉLEMY",
    "SH": "SAINT HELENA, ASCENSION AND TRISTAN DA CUNHA",
    "KN": "SAINT KITTS AND NEVIS",
    "LC": "SAINT LUCIA",
    "MF": "SAINT MARTIN (FRENCH PART)",
    "PM": "SAINT PIERRE AND MIQUELON",
    "VC": "SAINT VINCENT AND THE GRENADINES",
    "WS": "SAMOA",
    "SM": "SAN MARINO",
    "ST": "SAO TOME AND PRINCIPE",
    "SA": "SAUDI ARABIA",
    "SN": "SENEGAL",
    "RS": "SERBIA",
    "SC": "SEYCHELLES",
    "SL": "SIERRA LEONE",
    "SG": "SINGAPORE",
    "SX": "SINT MAARTEN (DUTCH PART)",
    "SK": "SLOVAKIA",
    "SI": "SLOVENIA",
    "SB": "SOLOMON ISLANDS",
    "SO": "SOMALIA",
    "ZA": "SOUTH AFRICA",
    "GS": "SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS",
    "SS": "SOUTH SUDAN",
    "ES": "SPAIN",
    "LK": "SRI LANKA",
    "SD": "SUDAN",
    "SR": "SURINAME",
    "SJ": "SVALBARD AND JAN MAYEN",
    "SZ": "SWAZILAND",
    "SE": "SWEDEN",
    "CH": "SWITZERLAND",
    "SY": "SYRIAN ARAB REPUBLIC",
    "TW": "TAIWAN, PROVINCE OF CHINA",
    "TJ": "TAJIKISTAN",
    "TZ": "TANZANIA, UNITED REPUBLIC OF",
    "TH": "THAILAND",
    "TL": "TIMOR-LESTE",
    "TG": "TOGO",
    "TK": "TOKELAU",
    "TO": "TONGA",
    "TT": "TRINIDAD AND TOBAGO",
    "TN": "TUNISIA",
    "TR": "TURKEY",
    "TM": "TURKMENISTAN",
    "TC": "TURKS AND CAICOS ISLANDS",
    "TV": "TUVALU",
    "UG": "UGANDA",
    "UA": "UKRAINE",
    "AE": "UNITED ARAB EMIRATES",
    "GB": "UNITED KINGDOM",
    "US": "UNITED STATES",
    "UM": "UNITED STATES MINOR OUTLYING ISLANDS",
    "UY": "URUGUAY",
    "UZ": "UZBEKISTAN",
    "VU": "VANUATU",
    "VE": "VENEZUELA, BOLIVARIAN REPUBLIC OF",
    "VN": "VIET NAM",
    "VG": "VIRGIN ISLANDS, BRITISH",
    "VI": "VIRGIN ISLANDS, U.S.",
    "WF": "WALLIS AND FUTUNA",
    "EH": "WESTERN SAHARA",
    "YE": "YEMEN",
    "ZM": "ZAMBIA",
    "ZW": "ZIMBABWE",
}
```

Region Plot

A barplot of the profitable regions (countries)

Below is where I do the actual mapping with a lambda function and get only the data that I need.

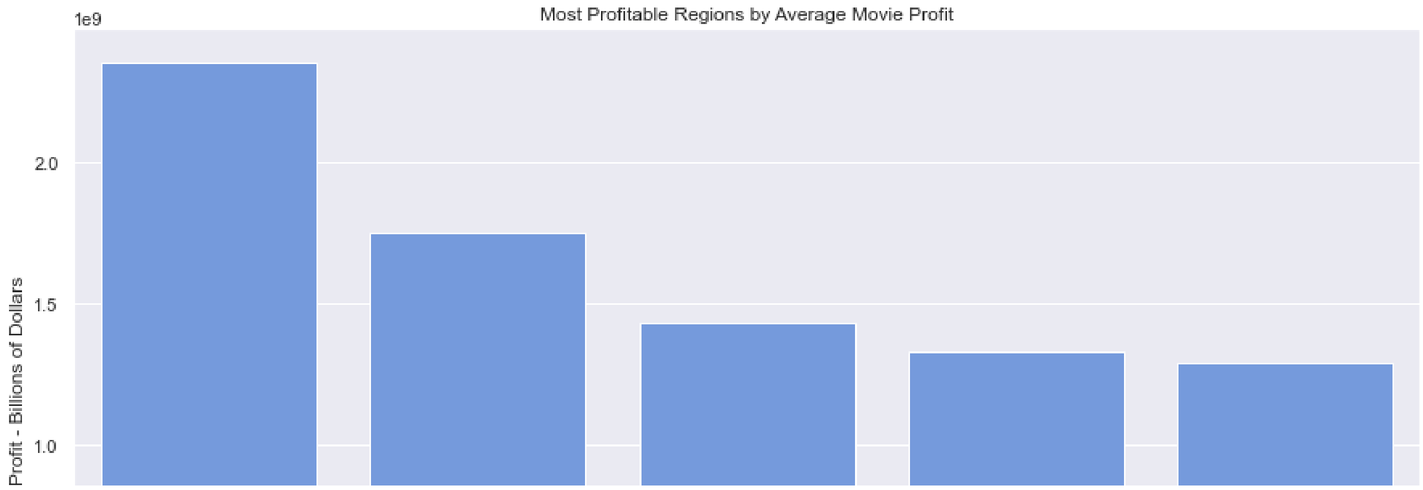
In [78]:

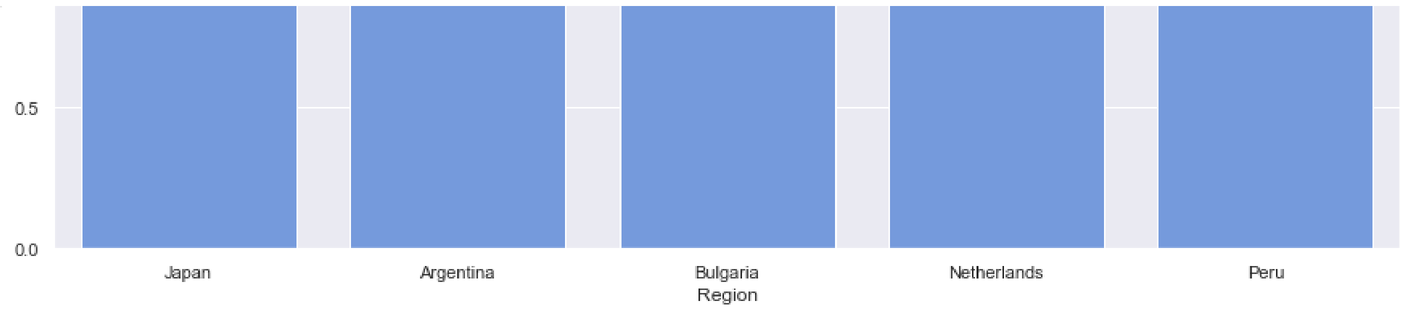
```
tcombo_profit_region_median = tcombo[['profit','region']].sort_values(by='profit',ascending=False)[:12]

#map cCodes to tcombo_profit_region
tcombo_profit_region_median['region'] = tcombo_profit_region_median['region'].map(lambda x: CC[x].title())

tcombo_profit_region_median = tcombo_profit_region_median.groupby(['region']).median().sort_values(by='profit',ascending=False)[:5]
```

Making the plot

In [79]:

```
sns.set_theme(context='notebook',style="darkgrid")

sns.set(rc={'figure.figsize':(15,8)})

ax = sns.barplot(x=tcombo_profit_region_median.index,y='profit',data=tcombo_profit_region_median,color='cornflowerblue')

ax.set(xlabel='Region',ylabel='Profit - Billions of Dollars',title='Most Profitable Regions by Average Movie Profit');
```





18. Profit vs Writer

This code makes a barplot of the top writers for the most profitable movies. The first three cells will be the same for each plot because I used the same data for all of them.

I looked at each table to make note of all available columns. I noticed the tconst and nconst columns that I counld use to merge all of the data together into one dataframe. After merging, I dropped many columns that I did not need. These included columns that I though contained duplicate information or information that would not be useful. Next, I dropped rows with null values in a subset of all columns. I used outer merges so there where lots of rows with missing data. I wanted to be sure of the data that I had before getting rid of any. I also did not want to drop rows that were only missing data in either domestic or worldwide gross because I only needed one.

Next, I made a new dataframe with only the rows where the profession was a writer. I grouped that dataframe my the median profit and the name of the person.

Get only the data that I need

In [80]:

```
#get a subset of the data
tcombo_profession_profit = tcombo[['directors','profit','primary_name','primary_professio
n']]
#split directors on the comma
tcombo_profession_profit[:]['directors'] = tcombo_profession_profit['directors'].str.spli
t(',')
#split the professions on the comma
tcombo_profession_profit[:]['primary_profession'] = tcombo_profession_profit['primary_pro
fession'].str.split(',')
```

Now I need to make sure that there is only one entry per row.

In [81]:

```
#use .explode to put one director per row
tcombo_profession_profit = tcombo_profession_profit.explode('directors')

#replace spaces with new line so that the names look better on the graphs
tcombo_profession_profit.primary_name = tcombo_profession_profit.primary_name.map(lambda
x: x.replace(' ','\n'))

#use .explode again to put each profession on one row
tcombo_profession_profit = tcombo_profession_profit.explode('primary_profession')
```

Select only the writers

In [82]:

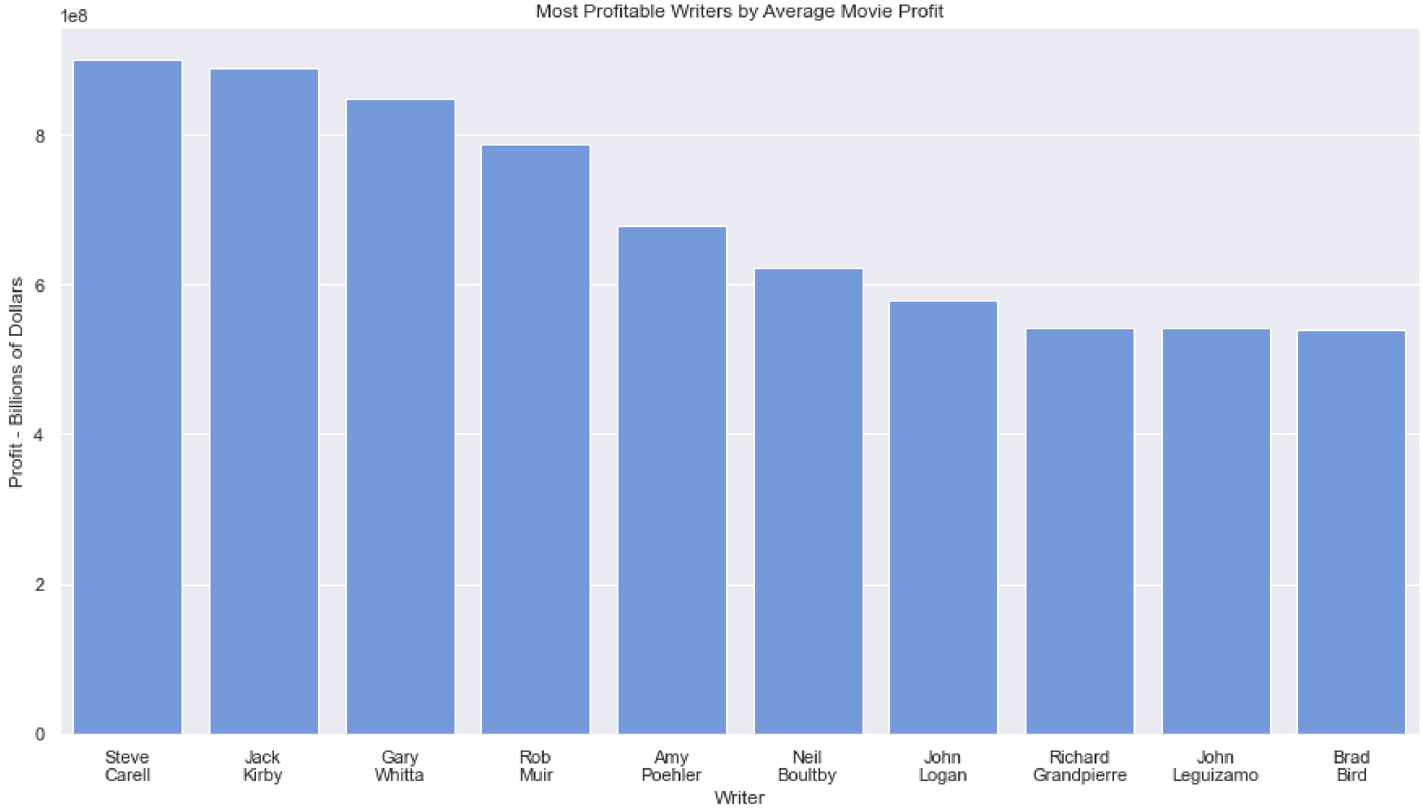
```
#find all rows where the profession is writer
#add atleast number of movies
writer_group_median = tcombo_profession_profit.loc[tcombo_profession_profit['primary_prof
ession'] == 'writer']
#groupby name and average of profits for all movies that the writer had a part in
writer_group_median = writer_group_median.groupby(['primary_name']).median().sort_values
```

```
(by='profit',ascending=False)[:10]
```

Make the plot

In [83]:

```
sns.set(rc={'figure.figsize':(15,8)})

ax = sns.barplot(x=writer_group_median.index,y='profit',data=writer_group_median,color='
cornflowerblue')

ax.set(xlabel='Writer',ylabel='Profit - Billions of Dollars',title='Most Profitable Write
rs by Average Movie Profit');
```



19. Release month vs profit

I looked at each table to make note of all available columns. I noticed the tconst and nconst columns that I counld use to merge all of the data together into one dataframe. After merging, I dropped many columns that I did not need. These included columns that I though contained duplicate information or information that would not be useful. Next, I dropped rows with null values in a subset of all columns. I used outer merges so there where lots of rows with missing data. I wanted to be sure of the data that I had before getting rid of any. I also did not want to drop rows that were only missing data in either domestic or worldwide gross because I only needed one.

The first plot shows the total profit made per release month. The second shows the total number of movies released per month. I suspected that the beginning of the summer would be where the most money was made and when most movies were released. It looks like June movies made the most money, December has the higher number of movies released.

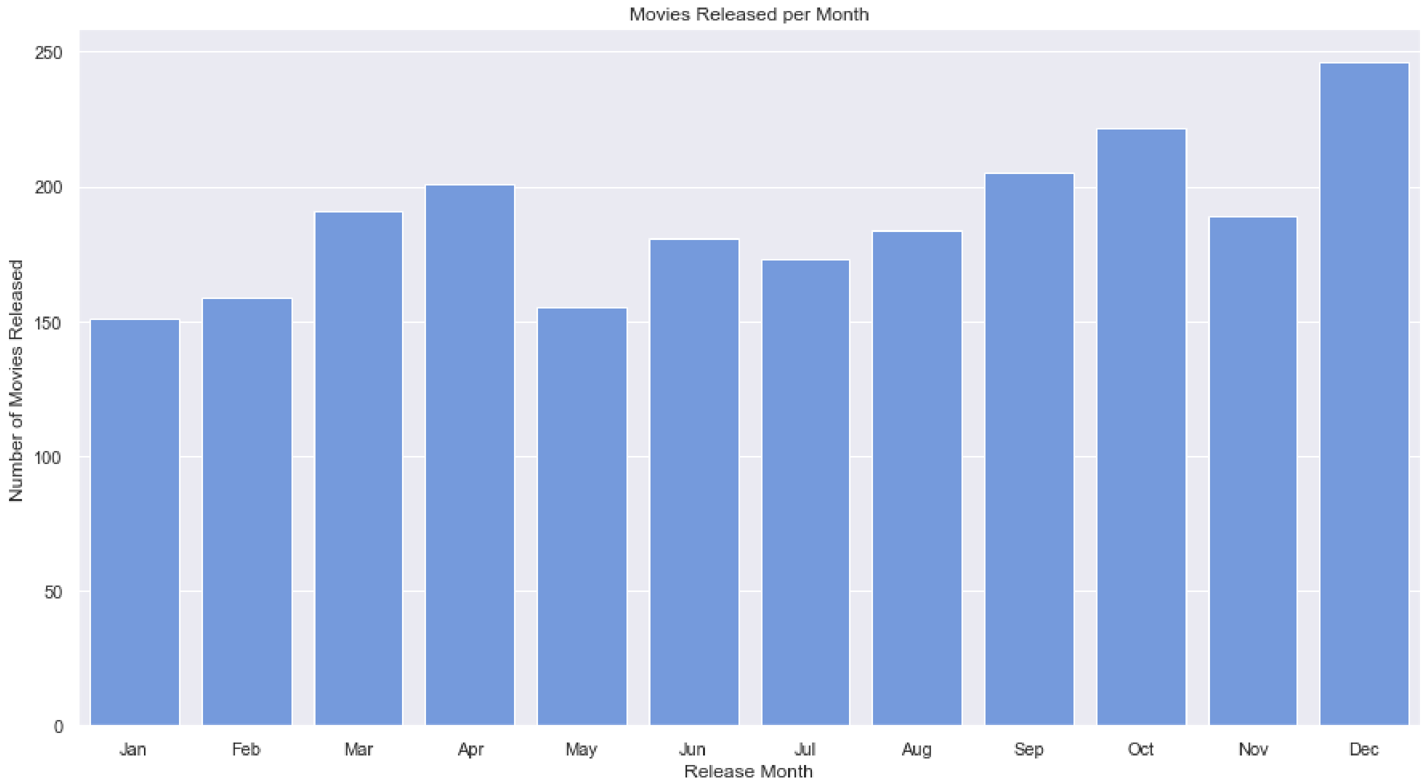
This plot suggests the best month to release a movie of a given genre.

I define an order that I want the months to show up in and then, make the column values categorical and pass in the sort order. Next, I sort by the release month and make a histogram showing the number of movies released per month.

In [84]:

```
#define an order for the months
```

```
month_order = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
#make the months categorical and pass the order that they should appear
tcombo['release_month'] = pd.Categorical(tcombo['release_month'], month_order)
#sort by release month
tcombo = tcombo.sort_values(by='release_month',ascending=False)
#make the plot
sns.set(rc={'figure.figsize':(15,8)})

ax = sns.countplot(x='release_month',data=tcombo,color='cornflowerblue')
#how many movies are released each month
ax.set(xlabel='Release Month',ylabel='Number of Movies Released',title='Movies Released
per Month');
```



Best Movie Genres to Release per Month

First, I make a new dataframe with the genres, release months, and profits. The genres are in a comma separated string so I split them on the comma and then use .explode on the lists so that each genre is in its own row.

In [85]:

```
release_genre = tcombo[['genres','release_month','profit']]
release_genre[:]['genres'] = release_genre['genres'].str.split(',')
release_genre = release_genre.explode('genres')

release_genre.dropna(axis=0,how='any',subset=['profit'],inplace=True)
```

I wrote this function to return the total profit per genre of movies in a given month.

In [86]:

```
#returns the profit of each genre sorted descending for the given month
def month_profit_sum(month,release_genre):
    month_profit = release_genre.loc[release_genre['release_month'] == month]

    #removed this line so that it is not grouped by release month anymore
    #    month_profit = month_profit.groupby(['release_month','genres']).sum()

    month_profit_sum = month_profit.groupby(['genres']).sum()
    month_profit_sum.dropna(axis=0,how='any',subset=['profit'],inplace=True)

    month_profit_sum = month_profit_sum.sort_values(by='profit',ascending=False)
    return month_profit_sum
```

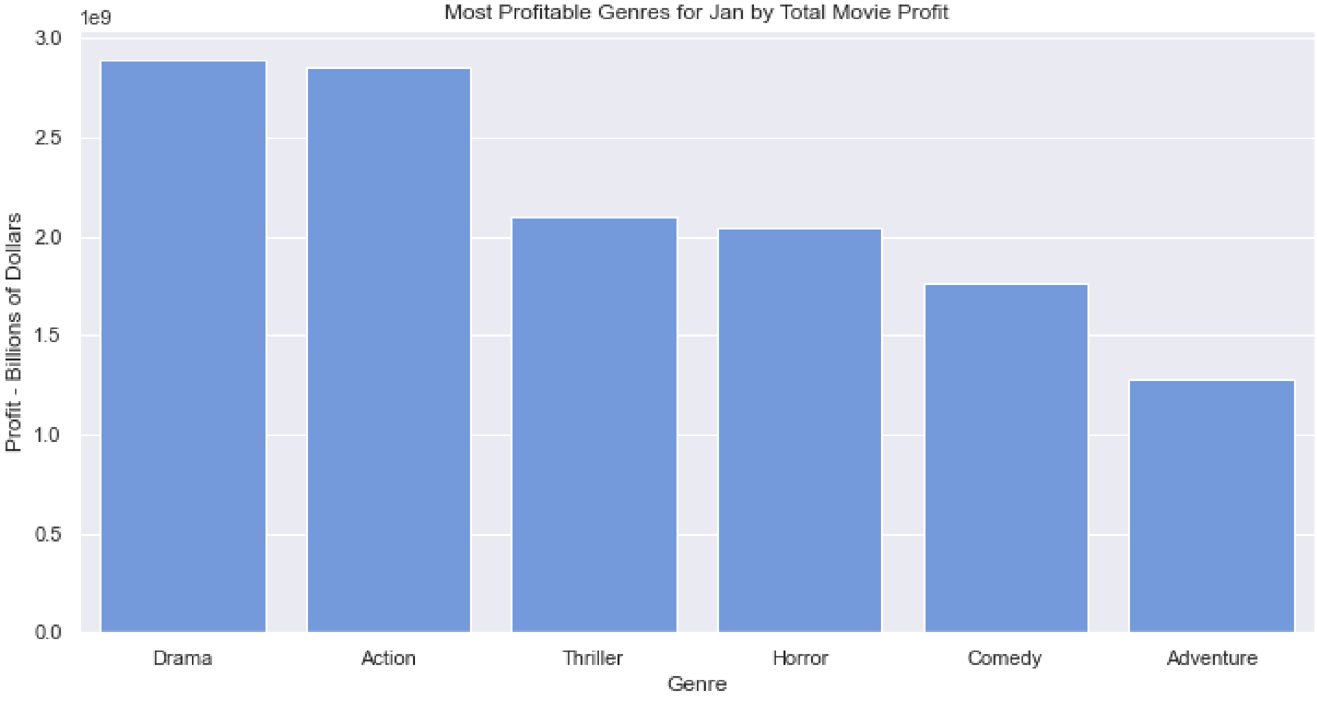
It is interesting to see which genres are most profitable per month. For example: October-Thriller ; February-Romance is 5th

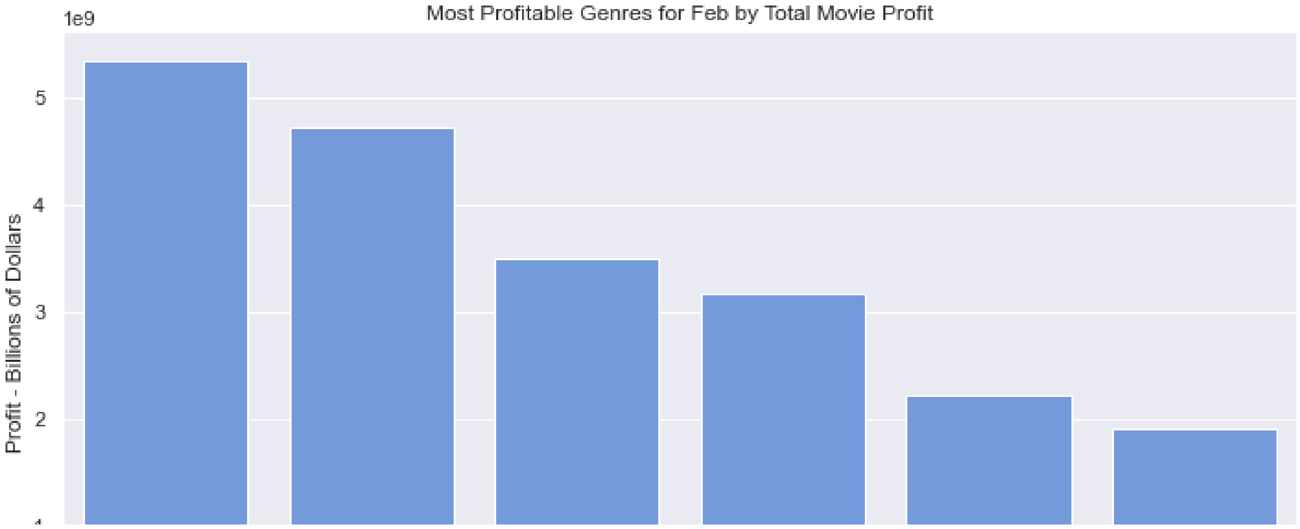
Maybe October is spooky month, February is romance month, and everything else is action/adventure.

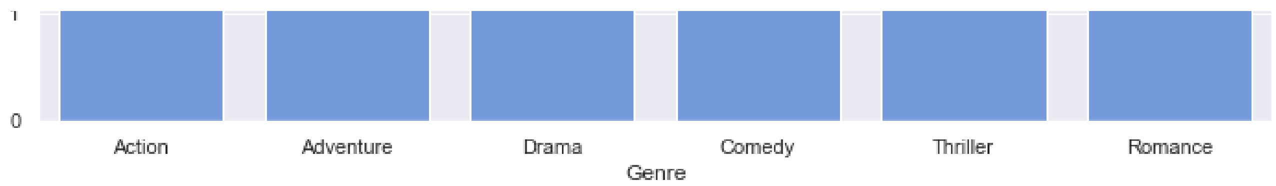
I wrote a for loop to make a bar plot showing the profitable genres by month.

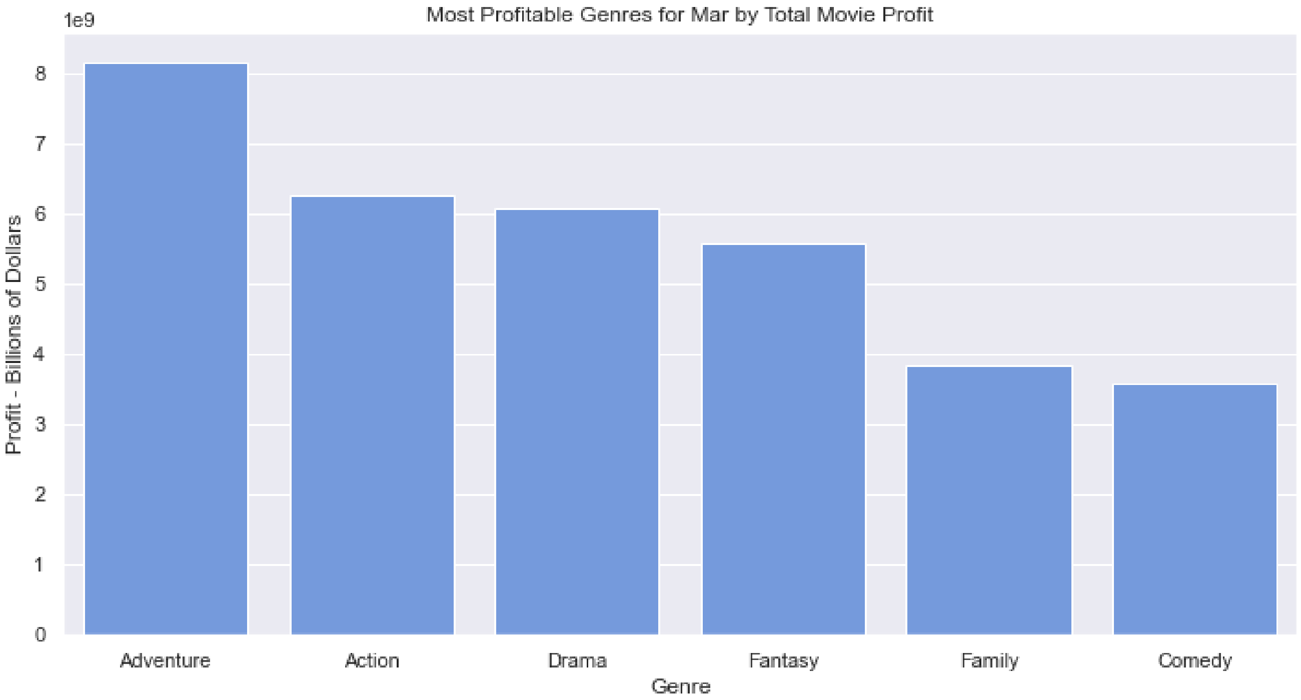
In [87]:

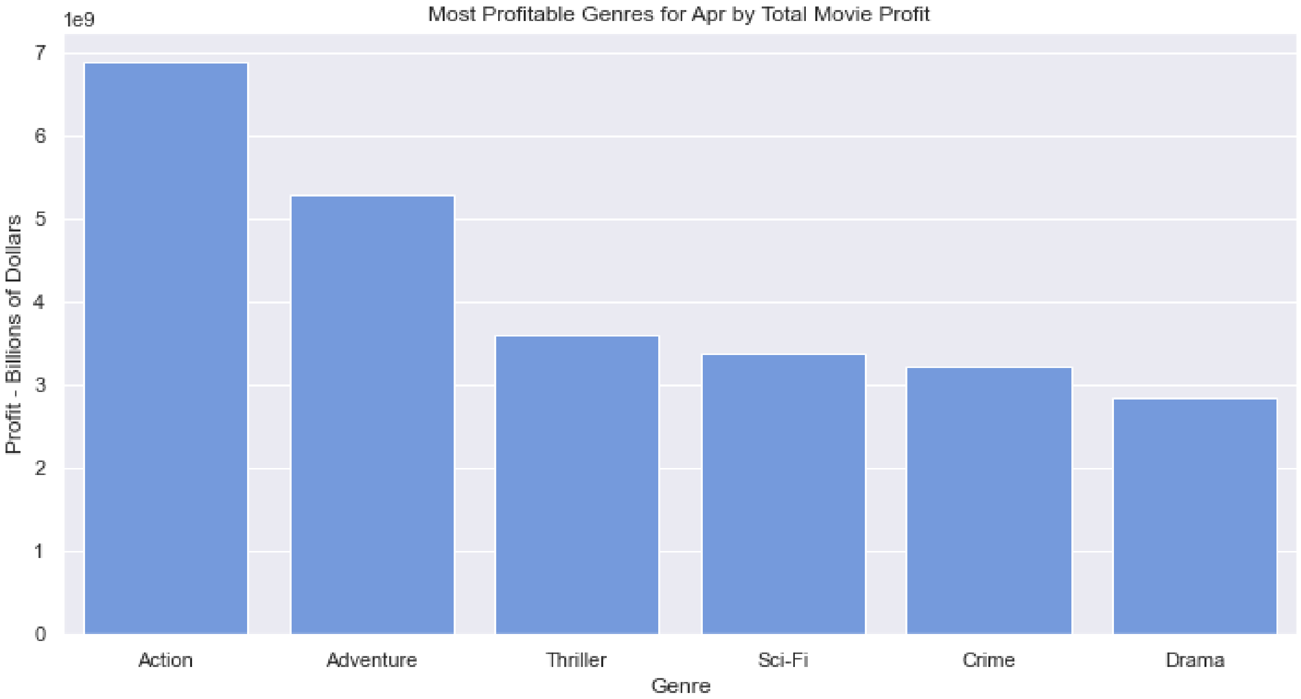
```
month_order = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']

months_to_get = ['Feb','Oct']
#makes a plot of top genres for each month passed to it
for month in month_order:

    fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,6))


    df=month_profit_sum(month,release_genre)
    df = df[:6]
    #print(df)
    #print(df.index)

    ax = sns.barplot(x=df.index,y='profit',data=df,color='cornflowerblue')
    ax.set(xlabel='Genre',ylabel='Profit - Billions of Dollars',title=f'Most Profitable
Genres for {month.title()} by Total Movie Profit')
```

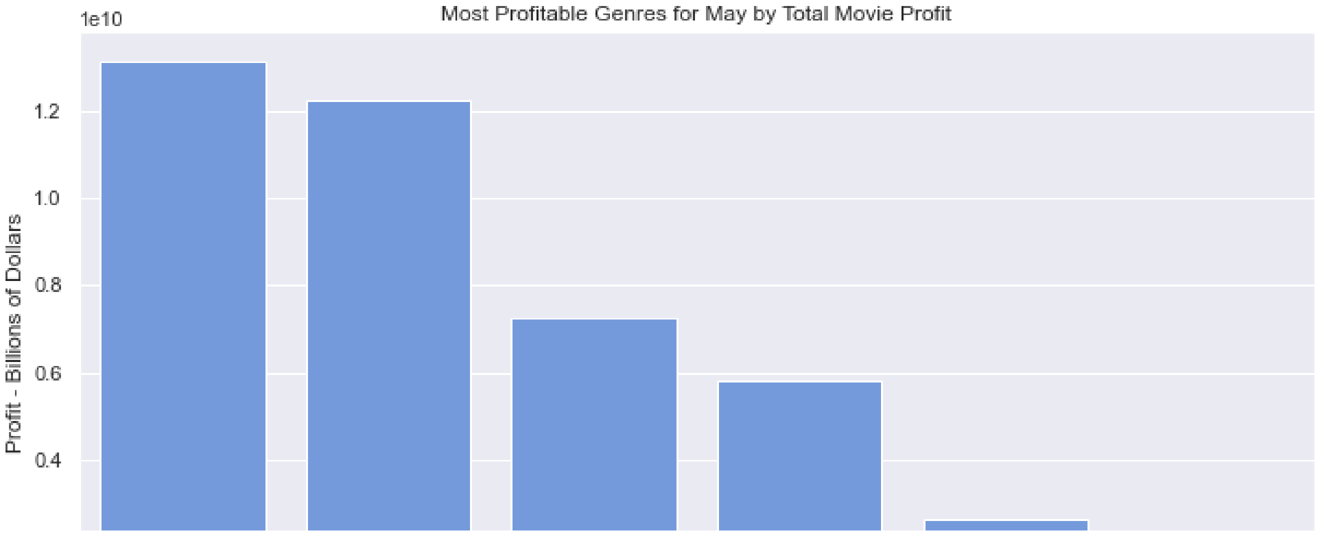


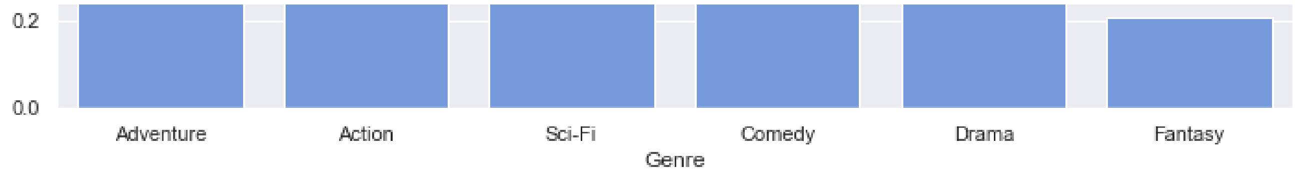


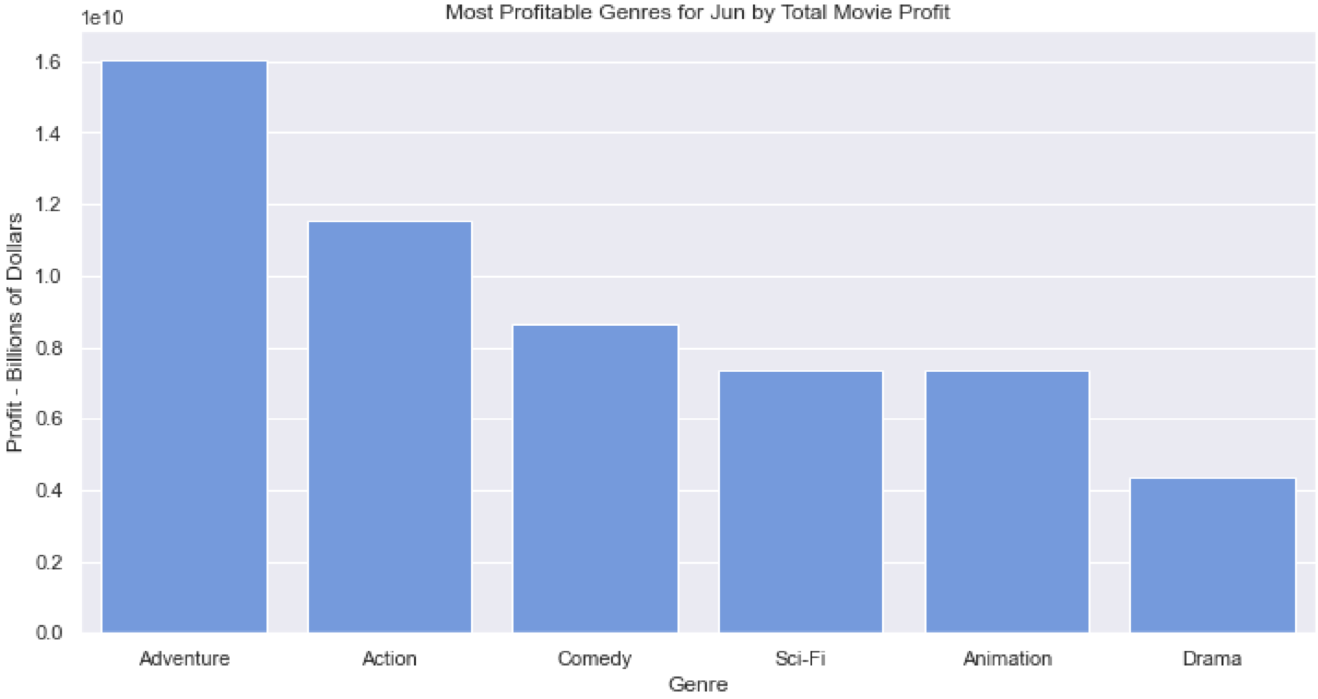

1
0
Action
Adventure
Drama
Comedy
Thriller
Romance
Genre

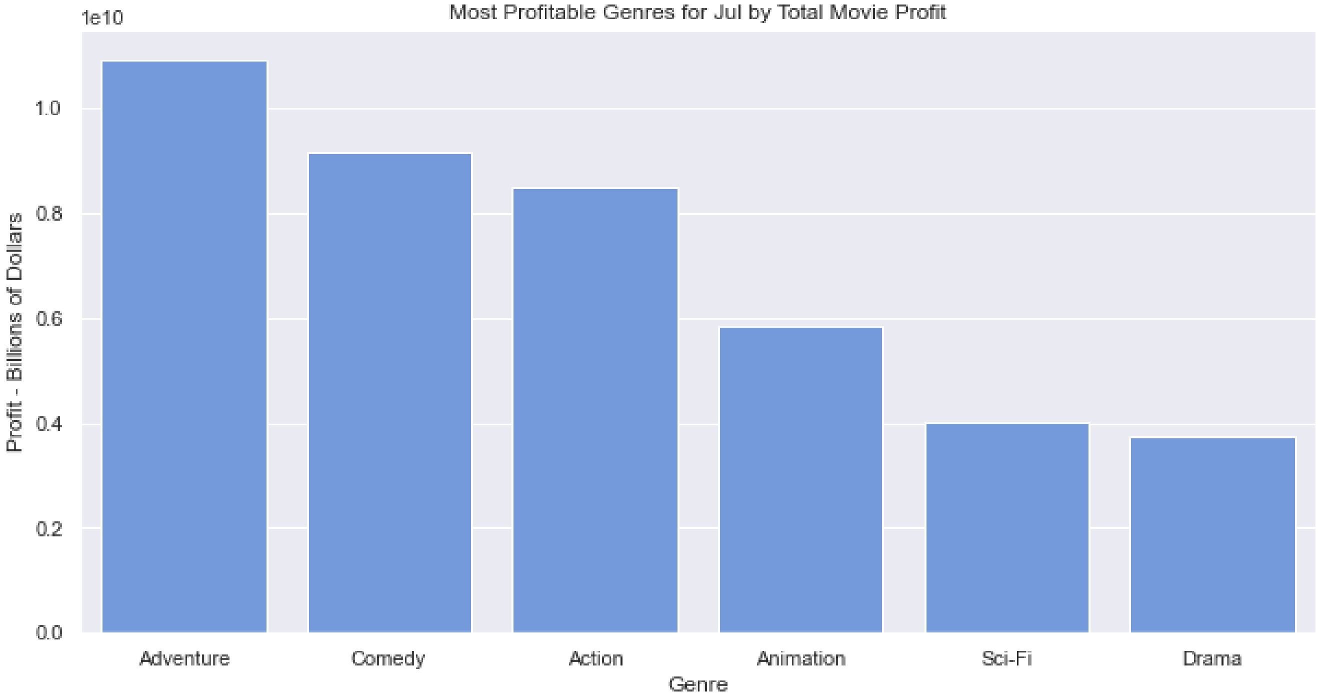

Most Profitable Genres for Mar by Total Movie Profit
1e9
Profit - Billions of Dollars
8
7
6
5
4
3
2
1
0
Adventure
Action
Drama
Fantasy
Family
Comedy
Genre

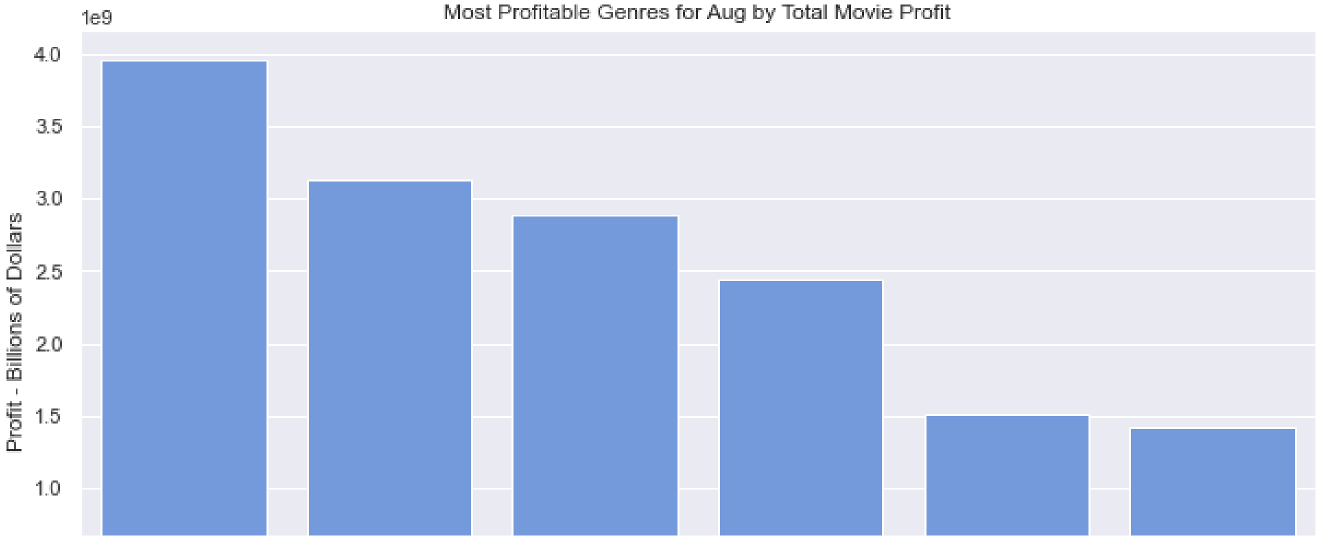

Most Profitable Genres for Apr by Total Movie Profit
1e9
Profit - Billions of Dollars
7
6
5
4
3
2
1
0
Action
Adventure
Thriller
Sci-Fi
Crime
Drama
Genre

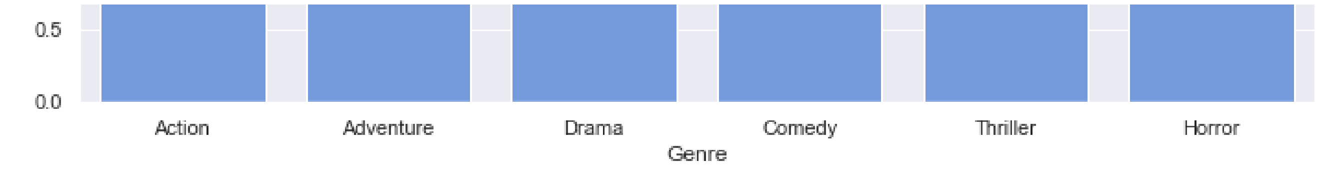

Most Profitable Genres for May by Total Movie Profit
1e10
Profit - Billions of Dollars
1.2
1.0
0.8
0.6
0.4

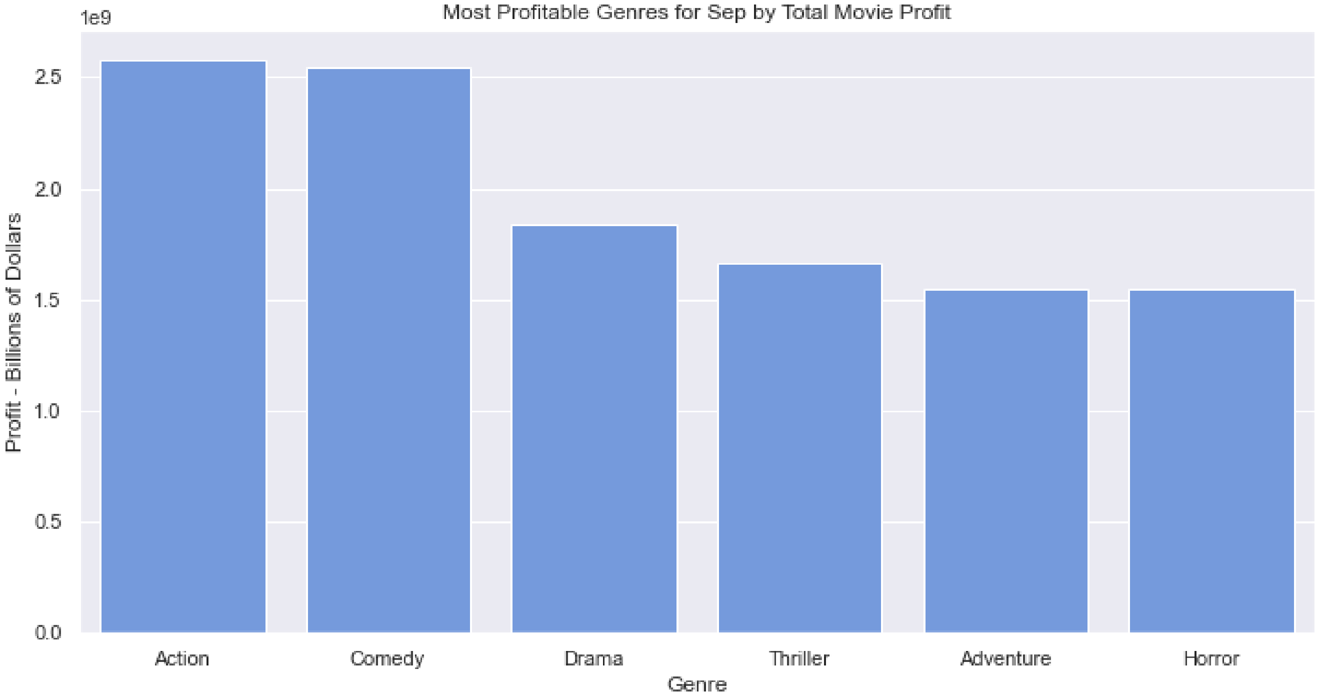


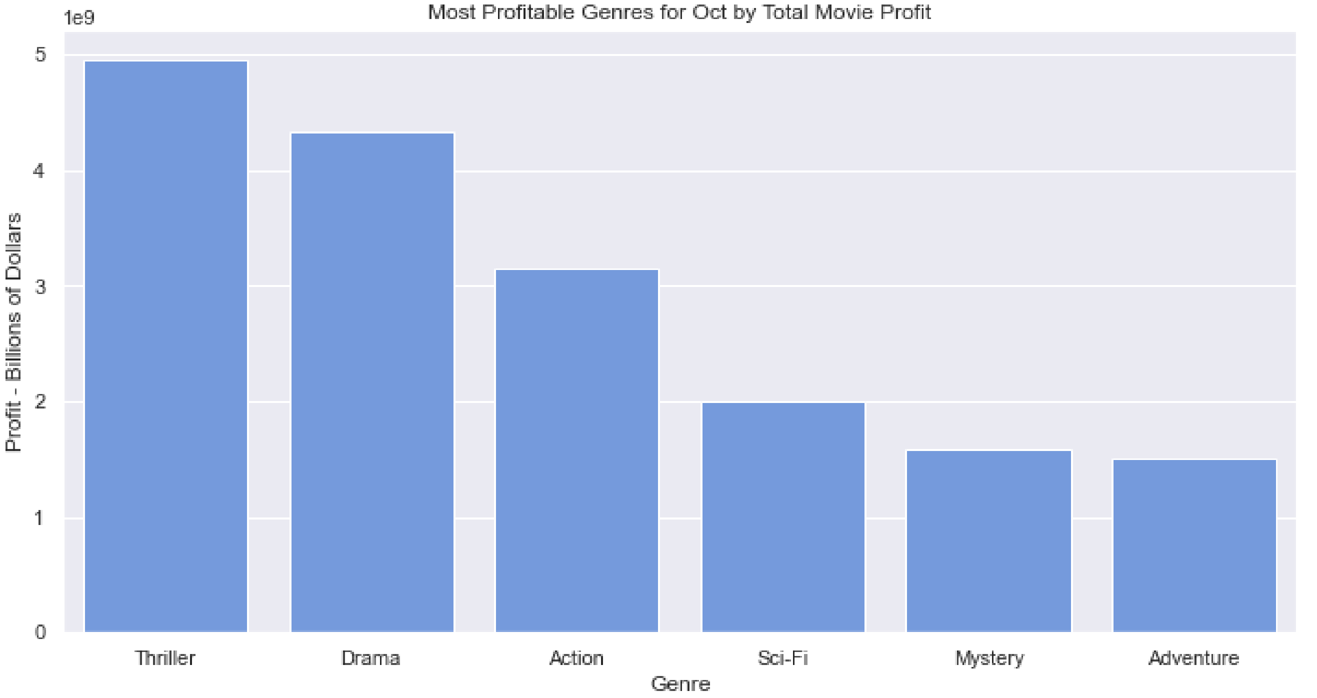

Most Profitable Genres for Jun by Total Movie Profit

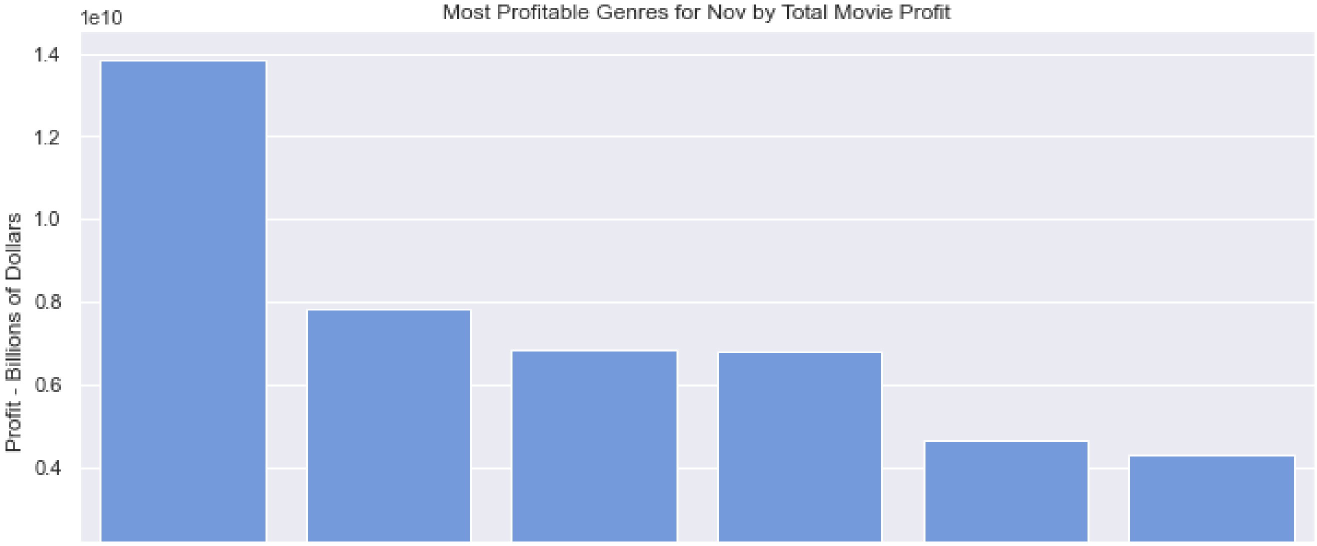

Most Profitable Genres for Jul by Total Movie Profit

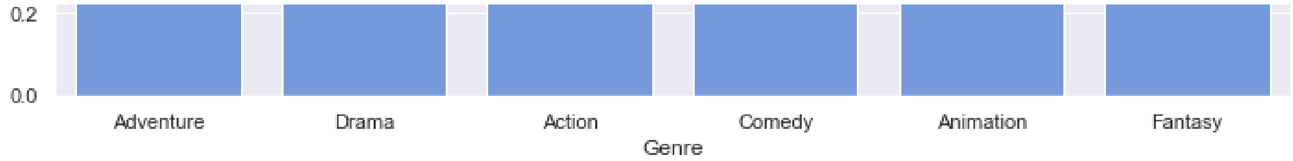

Most Profitable Genres for Aug by Total Movie Profit

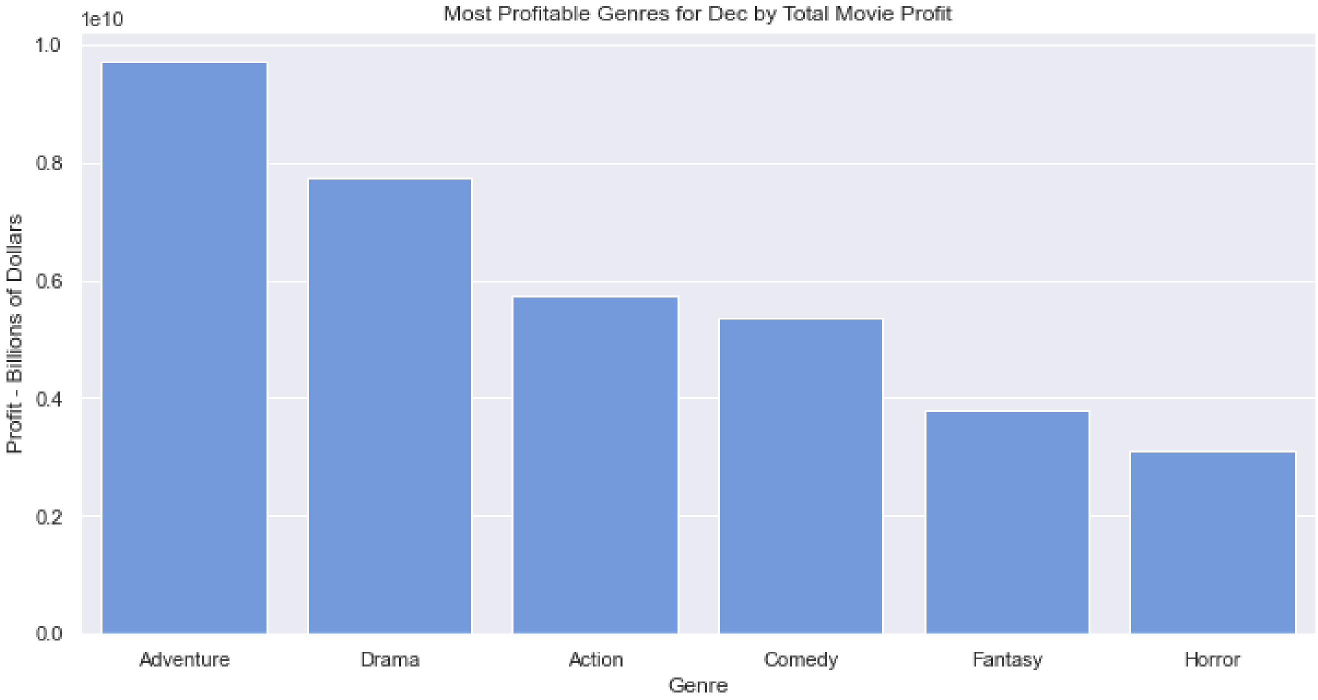

0.5
0.0
Action
Adventure
Drama
Comedy
Thriller
Horror
Genre


Most Profitable Genres for Sep by Total Movie Profit
1e9
2.5
2.0
1.5
1.0
0.5
0.0
Profit - Billions of Dollars
Action
Comedy
Drama
Thriller
Adventure
Horror
Genre


Most Profitable Genres for Oct by Total Movie Profit
1e9
5
4
3
2
1
0
Profit - Billions of Dollars
Thriller
Drama
Action
Sci-Fi
Mystery
Adventure
Genre


Most Profitable Genres for Nov by Total Movie Profit
1e10
1.4
1.2
1.0
0.8
0.6
0.4
Profit - Billions of Dollars


0.2
0.0
Adventure
Drama
Action
Comedy
Animation
Fantasy
Genre


Most Profitable Genres for Dec by Total Movie Profit
1e10
1.0
0.8
0.6
0.4
0.2
0.0
Profit - Billions of Dollars
Adventure
Drama
Action
Comedy
Fantasy
Horror
Genre

```
In [ ]:
```