Questions:

1. Which genres are the most common (Number of movies made)?
2. Which movie geners have high avg budget and revenue?
3. Which movie genres have high profit?
4. Which generes have high avg popularity?
5. Which geners have highest number of movies with an voting average >= 8?

Research Hypothesis (H):

1. The best movies according to vote average return high profit and revenue.
2. The best movies according to popularity return high profit and revenue.
3. Highly budgeted movies return high revenue and profit.
4. Highly budgeted movies have a high popularity.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
movies = pd.read_csv(r'E:\Data Analytics\Pandas\pandas mastery\13.real world projec
movies
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...
...	...	...	...	...	...	...	...
10861	21	tt0060371	0.080598	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...
10862	20379	tt0060472	0.065543	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...
10863	39768	tt0060161	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...
10864	21449	tt0061177	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...
10865	22293	tt0060666	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...

	id	imdb_id	popularity	budget	revenue	original_title	cast

In [3]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

In [4]:
```
movies.head()
```

Out[4]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://ww
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	htt
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	

5 rows × 21 columns

In [5]:
```
# Check for duplicates
movies[movies.duplicated()]
```

Out[5]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	di
2090	42194	tt0411951	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	NaN	I H

1 rows × 21 columns

In [6]:
```
# Removing duplicates
movies.drop_duplicates(inplace = True)
```

In [7]:
```
# Checking the geners which have null
movies.dropna(subset = ['genres'], inplace = True)
```

In [8]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10842 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10842 non-null  int64  
 1   imdb_id               10834 non-null  object 
 2   popularity            10842 non-null  float64
 3   budget                10842 non-null  int64  
 4   revenue               10842 non-null  int64  
 5   original_title        10842 non-null  object 
 6   cast                  10767 non-null  object 
 7   homepage              2931 non-null   object 
 8   director              10800 non-null  object 
 9   tagline               8036 non-null   object 
 10  keywords              9367 non-null   object 
 11  overview              10839 non-null  object 
 12  runtime               10842 non-null  int64  
 13  genres                10842 non-null  object 
 14  production_companies  9826 non-null   object 
 15  release_date          10842 non-null  object 
 16  vote_count            10842 non-null  int64  
 17  vote_average          10842 non-null  float64
 18  release_year          10842 non-null  int64  
 19  budget_adj            10842 non-null  float64
 20  revenue_adj           10842 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.8+ MB
```

In [9]:
```
# Calucalting the profit(Question 3)
movies['profit'] = movies['revenue'] - movies['budget']
```

In [36]:
```
# Keeping only columns that are needed
movies_genres = movies[['popularity', 'budget', 'revenue', 'original_title', 'runti
                        'vote_count', 'profit']]
```

In [37]:
```
# movies_genres['genres'].str.split('|') => This part of the code divides the dataf
# .apply(Series,1) => Applying the data to individual series with axis = 1
# .stack() => Keeps all the data to the individual index
from pandas import Series, DataFrame
movies_genres['genres'].str.split('|').apply(Series,1).stack()
```

Out[37]:
```
0      0             Action
       1          Adventure
       2    Science Fiction
       3           Thriller
1      0             Action
                  ...       
10863  0            Mystery
       1             Comedy
10864  0             Action
       1             Comedy
10865  0             Horror
Length: 26955, dtype: object
```

In [46]:
```
# Getting rid of the second index
split = movies_genres['genres'].str.split('|').apply(Series,1).stack()
# The line of code didn't work but worked for alex the analyst
split_index = split.index.droplevel(-1)
split
```

Out[46]:
```
0      0              Action
       1           Adventure
       2     Science Fiction
       3            Thriller
1      0              Action
                       ...
10863  0             Mystery
       1              Comedy
10864  0              Action
       1              Comedy
10865  0              Horror
Length: 26955, dtype: object
```

In [47]:
```
movies_genres
```

Out[47]:

	popularity	budget	revenue	original_title	runtime	genres	release_
0	32.985763	150000000	1513528810	Jurassic World	124	Action|Adventure|Science Fiction|Thriller	6
1	28.419936	150000000	378436354	Mad Max: Fury Road	120	Action|Adventure|Science Fiction|Thriller	5/
2	13.112507	110000000	295238201	Insurgent	119	Adventure|Science Fiction|Thriller	3/
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	136	Action|Adventure|Science Fiction|Fantasy	12/
4	9.335014	190000000	1506249360	Furious 7	137	Action|Crime|Thriller	4
...	...	...	...	...	...	...	
10861	0.080598	0	0	The Endless Summer	95	Documentary	6/
10862	0.065543	0	0	Grand Prix	176	Action|Adventure|Drama	12/
10863	0.065141	0	0	Beregis Avtomobilya	94	Mystery|Comedy	1
10864	0.064317	0	0	What's Up, Tiger Lily?	80	Action|Comedy	11
10865	0.035919	19000	0	Manos: The Hands of Fate	74	Horror	11/

10842 rows × 10 columns

In [48]:
```
split.reset_index(level = 1, drop = True, inplace = True)
```

In [49]:
```
split
```

Out[49]:
```
0                 Action
0              Adventure
0        Science Fiction
0               Thriller
1                 Action
               ...
10863            Mystery
10863             Comedy
10864             Action
10864             Comedy
10865             Horror
Length: 26955, dtype: object
```

In [50]:
```
split.name = 'genres_split'
del movies_genres['genres']
movies_genres = movies_genres.join(split)
```

In [51]:
```
movies_genres
```

Out[51]:

	popularity	budget	revenue	original_title	runtime	release_date	vote_average	vot
0	32.985763	150000000	1513528810	Jurassic World	124	6/9/15	6.5	
0	32.985763	150000000	1513528810	Jurassic World	124	6/9/15	6.5	
0	32.985763	150000000	1513528810	Jurassic World	124	6/9/15	6.5	
0	32.985763	150000000	1513528810	Jurassic World	124	6/9/15	6.5	
1	28.419936	150000000	378436354	Mad Max: Fury Road	120	5/13/15	7.1	
...	...	...	...	...	...	...	...	
10863	0.065141	0	0	Beregis Avtomobilya	94	1/1/66	6.5	
10863	0.065141	0	0	Beregis Avtomobilya	94	1/1/66	6.5	
10864	0.064317	0	0	What's Up, Tiger Lily?	80	11/2/66	5.4	
10864	0.064317	0	0	What's Up, Tiger Lily?	80	11/2/66	5.4	
10865	0.035919	19000	0	Manos: The Hands of Fate	74	11/15/66	1.5	

26955 rows × 10 columns

1. Which genres are the most common (Number of movies made)?

In [55]:
```
# Counting the values and making it into a data frame
pd.DataFrame(movies_genres.groupby('genres_split').original_title.nunique()).sort_v
```

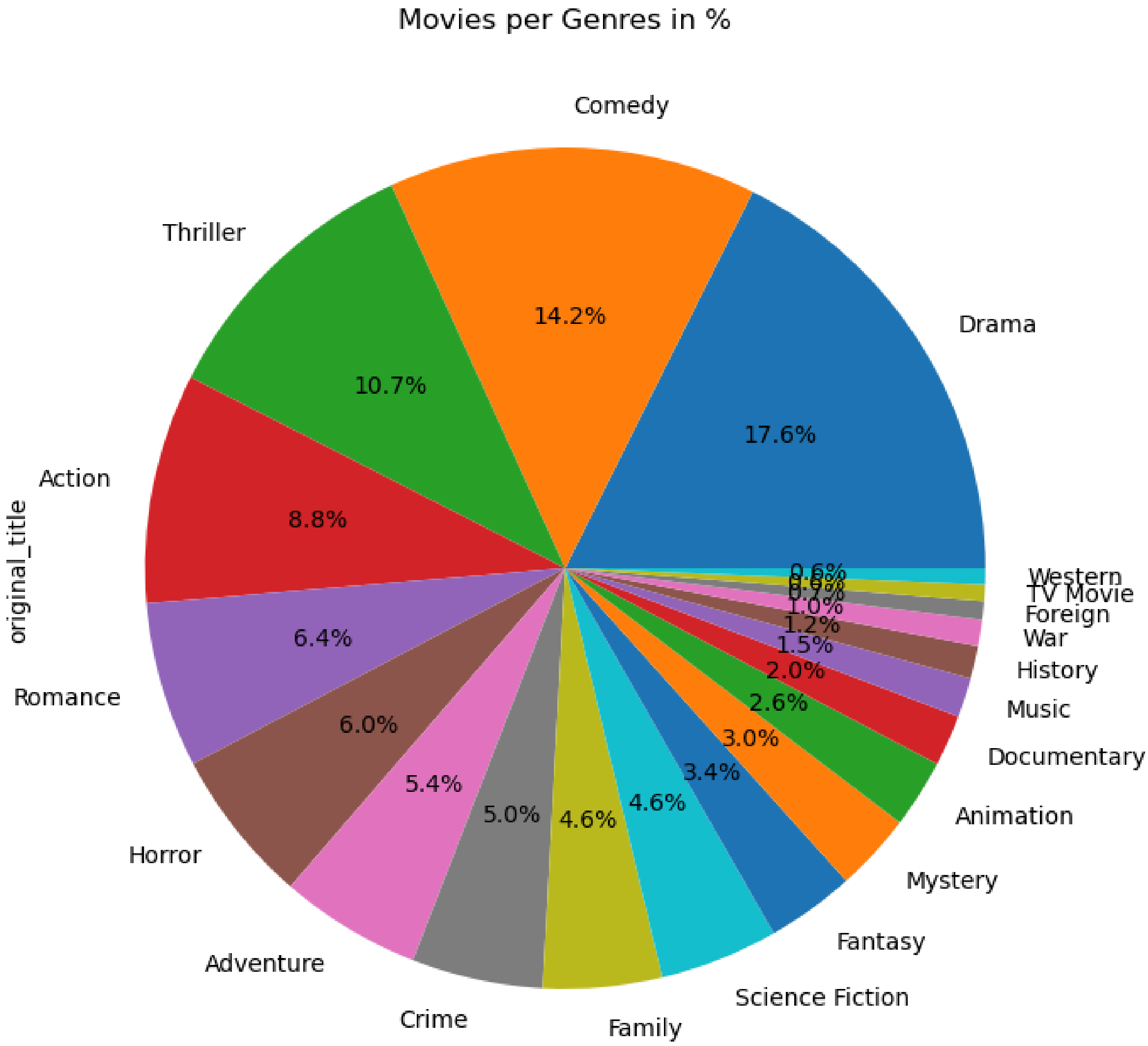
Out[55]:

genres_split	original_title
Drama	4672
Comedy	3750
Thriller	2841
Action	2339
Romance	1686
Horror	1580
Adventure	1442
Crime	1337
Family	1211
Science Fiction	1207
Fantasy	899
Mystery	796
Animation	697
Documentary	520
Music	403
History	331
War	269
Foreign	188
TV Movie	167
Western	163

In [56]:
```
genres_count = pd.DataFrame(movies_genres.groupby('genres_split').original_title.nu
                          sort_values('original_title',ascending = False)
```

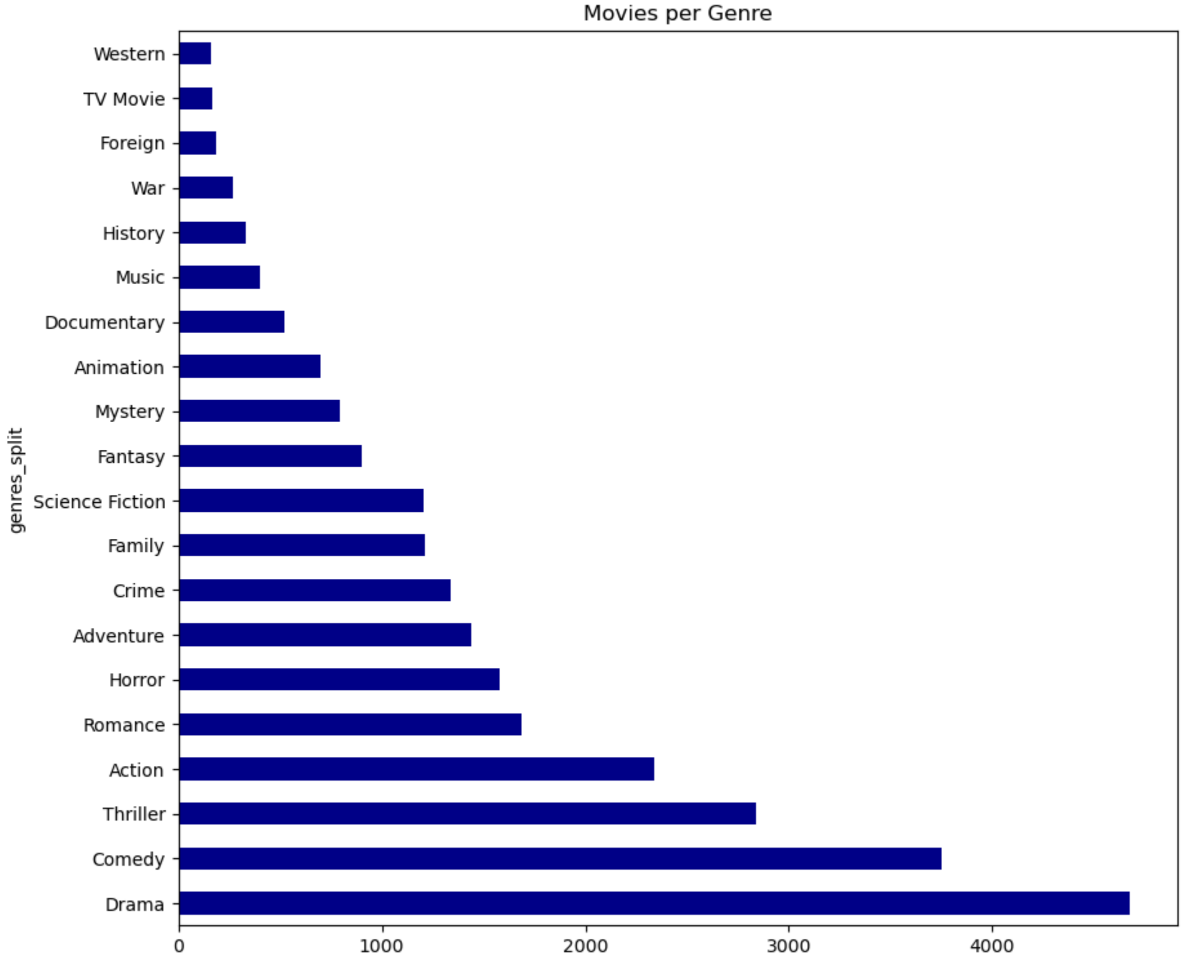
In [66]:
```
genres_count['original_title'].plot.pie(title = 'Movies per Genres in %',autopct =
```

Out[66]:
```
<Axes: title={'center': 'Movies per Genres in %'}, ylabel='original_title'>
```



In [68]:
```
# Making horizantal bar chart
# If we want the bar chart in a reverse way, then we have to change in the genres_c
genres_count['original_title'].plot.barh(title = 'Movies per Genre', color = 'DarkB
```

Out[68]:
```
<Axes: title={'center': 'Movies per Genre'}, ylabel='genres_split'>
```



1. Which movie geners have high avg budget and revenue?

In [162…

```
# Here we need to extract the numerical data from the movies_genres
numerical_data = movies_genres.select_dtypes(include = ['int','float'])
numerical_data
```

Out[162]:

	popularity	budget	revenue	runtime	vote_average	vote_count	profit
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810
1	28.419936	150000000	378436354	120	7.100000	6185	228436354
...	...	...	...	...	...	...	...
10863	0.065141	0	0	94	6.500000	11	0
10863	0.065141	0	0	94	6.500000	11	0
10864	0.064317	0	0	80	5.400000	22	0
10864	0.064317	0	0	80	5.400000	22	0
10865	0.035919	19000	0	74	1.500000	15	-19000

26955 rows × 7 columns

In [163…

```
# Since, we need to group the data using the genres_split column, we need to add th
numerical_data['genres_split'] = movies_genres['genres_split'].copy()
numerical_data
```

Out[163]:

	popularity	budget	revenue	runtime	vote_average	vote_count	profit	genre
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	Adv
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	S
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	
1	28.419936	150000000	378436354	120	7.100000	6185	228436354	
...	...	...	...	...	...	...	...	
10863	0.065141	0	0	94	6.500000	11	0	M
10863	0.065141	0	0	94	6.500000	11	0	Co
10864	0.064317	0	0	80	5.400000	22	0	
10864	0.064317	0	0	80	5.400000	22	0	Co
10865	0.035919	19000	0	74	1.500000	15	-19000	

26955 rows × 8 columns

In [77]:

```
# Calucalting the mean of the data
genres_avg = numerical_data.groupby('genres_split').mean()
```

In [78]:

```
genres_avg
```

Out[78]:

genres_split	popularity	budget	revenue	runtime	vote_average	vote_count	
Action	0.926274	2.772782e+07	7.279473e+07	104.917785	5.787752	392.993708	4.5
Adventure	1.154259	3.754369e+07	1.131379e+08	106.173351	5.940585	513.125085	7.5
Animation	0.852182	2.315978e+07	7.525606e+07	68.181688	6.403147	303.000000	5.2
Comedy	0.592607	1.329792e+07	3.752624e+07	96.745057	5.905167	176.436330	2.4
Crime	0.744930	1.766380e+07	4.236866e+07	106.917282	6.124889	278.805022	2.4
Documentary	0.181432	5.771491e+05	2.041107e+06	102.651923	6.908462	35.105769	1.4
Drama	0.591495	1.188072e+07	2.923226e+07	110.478151	6.165546	182.544538	1.7
Family	0.786668	2.335934e+07	7.243318e+07	89.603574	5.997563	272.320877	4.9
Fantasy	0.992840	3.261259e+07	9.631366e+07	100.736900	5.863537	420.741266	6.3
Foreign	0.191496	1.451435e+06	1.520460e+06	107.228723	5.981383	16.627660	6.9
History	0.575936	1.859492e+07	3.201179e+07	136.206587	6.410479	183.772455	1.3
Horror	0.465357	6.226529e+06	1.682281e+07	94.424557	5.337447	120.059866	1.0
Music	0.487321	9.438628e+06	2.857177e+07	105.137255	6.480392	124.340686	1.9
Mystery	0.690012	1.611927e+07	4.021757e+07	105.928395	5.946790	236.998765	2.4
Romance	0.592082	1.253127e+07	3.569197e+07	106.891355	6.042874	166.070678	2.3
Science Fiction	1.001548	2.497268e+07	7.014056e+07	99.419854	5.665582	437.096013	4.5
TV Movie	0.270896	2.676647e+05	2.514970e+05	91.982036	5.788024	34.365269	-1.6
Thriller	0.741563	1.720769e+07	4.172842e+07	103.247678	5.750671	255.484348	2.4
War	0.727683	2.089189e+07	4.760518e+07	127.625926	6.297778	270.733333	2.6
Western	0.590615	1.897411e+07	2.856871e+07	117.575758	6.083030	205.739394	9.5

In [79]:

```python
# Changing the data from scientific format to noraml format
pd.options.display.float_format = '{:2f}'.format
genres_avg
```

Out[79]:

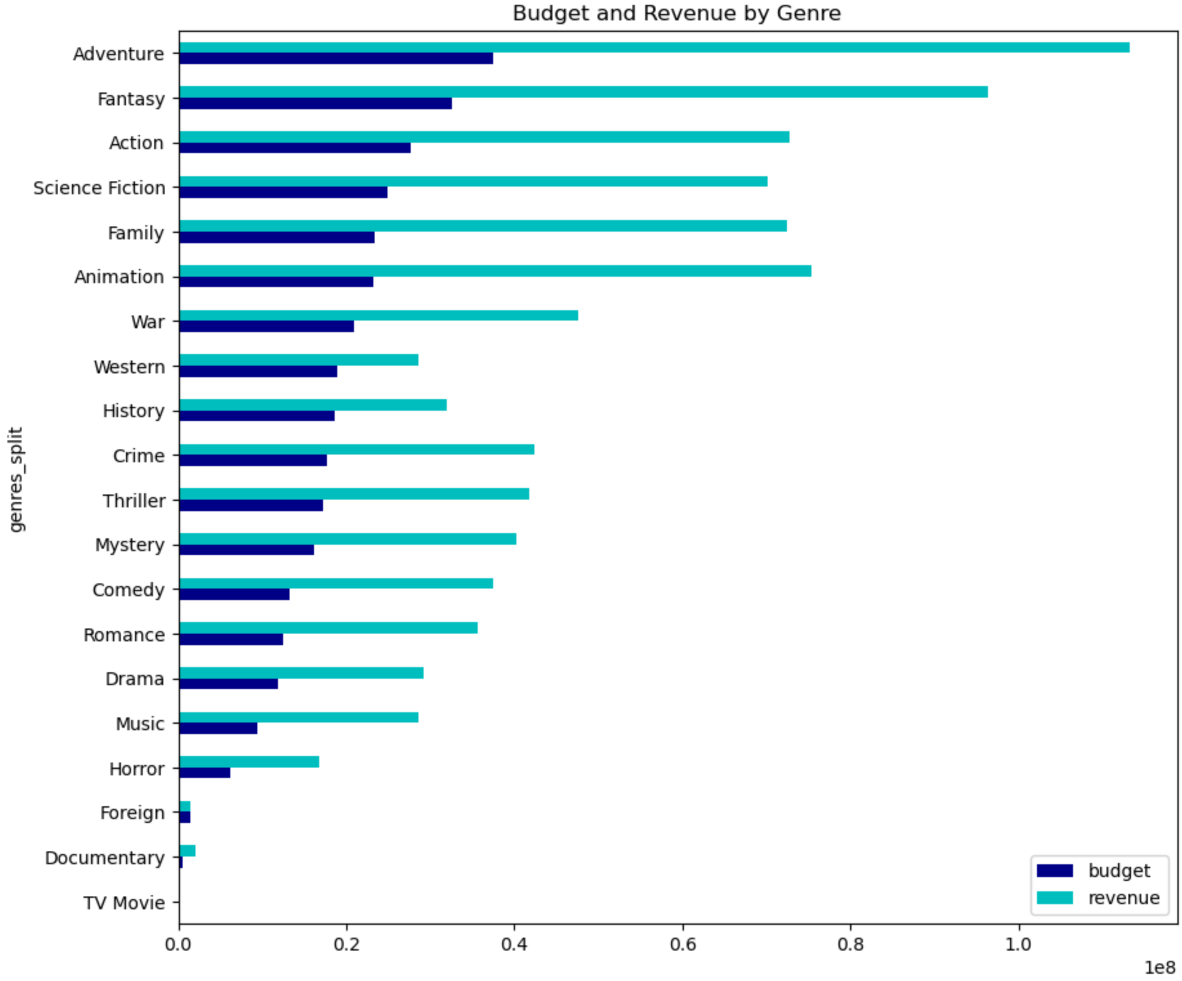
genres_split	popularity	budget	revenue	runtime	vote_average	vote_coun
Action	0.926274	27727820.331376	72794732.002517	104.917785	5.787752	392.99370
Adventure	1.154259	37543694.530931	113137861.069341	106.173351	5.940585	513.12508
Animation	0.852182	23159781.606581	75256062.223176	68.181688	6.403147	303.00000
Comedy	0.592607	13297915.618244	37526242.072238	96.745057	5.905167	176.43633
Crime	0.744930	17663801.124815	42368661.645495	106.917282	6.124889	278.80502
Documentary	0.181432	577149.148077	2041106.994231	102.651923	6.908462	35.10576
Drama	0.591495	11880717.773529	29232255.725840	110.478151	6.165546	182.54453
Family	0.786668	23359337.420796	72433176.373680	89.603574	5.997563	272.32087
Fantasy	0.992840	32612585.348253	96313657.081878	100.736900	5.863537	420.74126
Foreign	0.191496	1451434.925532	1520459.835106	107.228723	5.981383	16.62766
History	0.575936	18594919.302395	32011793.215569	136.206587	6.410479	183.77245
Horror	0.465357	6226529.210751	16822808.624313	94.424557	5.337447	120.05986
Music	0.487321	9438627.549020	28571768.691176	105.137255	6.480392	124.34068
Mystery	0.690012	16119270.062963	40217566.661728	105.928395	5.946790	236.99876
Romance	0.592082	12531271.847547	35691972.327103	106.891355	6.042874	166.07067
Science Fiction	1.001548	24972680.524003	70140558.034174	99.419854	5.665582	437.09601
TV Movie	0.270896	267664.670659	251497.005988	91.982036	5.788024	34.36526
Thriller	0.741563	17207693.769178	41728417.543860	103.247678	5.750671	255.48434
War	0.727683	20891886.103704	47605183.300000	127.625926	6.297778	270.73333
Western	0.590615	18974107.975758	28568709.284848	117.575758	6.083030	205.73939

In [82]:

```
genres_avg.sort_values('budget', ascending = True, inplace = True)
genres_avg[['budget', 'revenue']].plot.barh(title = 'Budget and Revenue by Genre',
```

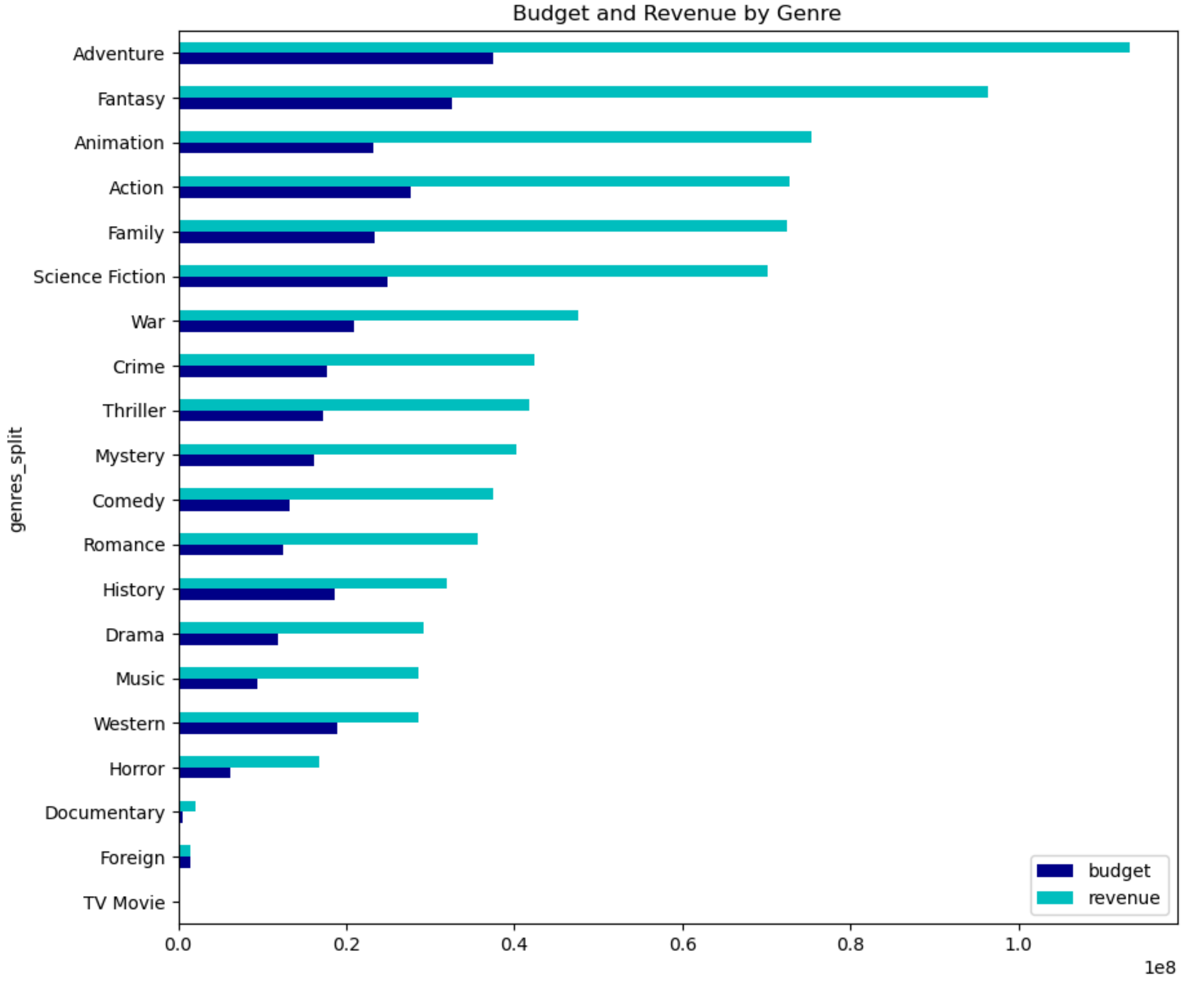
Out[82]:

```
<Axes: title={'center': 'Budget and Revenue by Genre'}, ylabel='genres_split'>
```



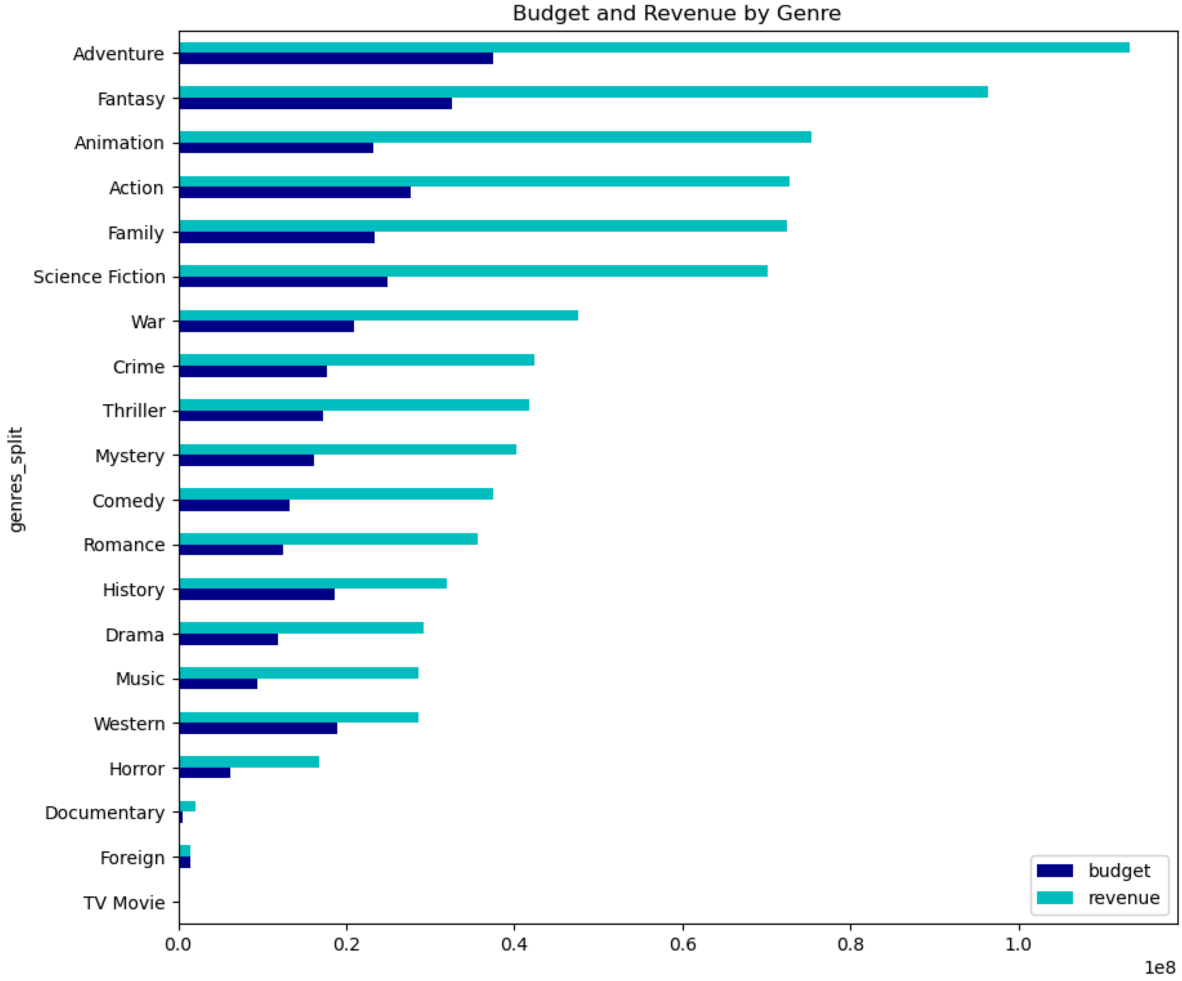
In [83]:
```
genres_avg.sort_values('revenue', ascending = True, inplace = True)
genres_avg[['budget', 'revenue']].plot.barh(title = 'Budget and Revenue by Genre',
```

Out[83]:
```
<Axes: title={'center': 'Budget and Revenue by Genre'}, ylabel='genres_split'>
```



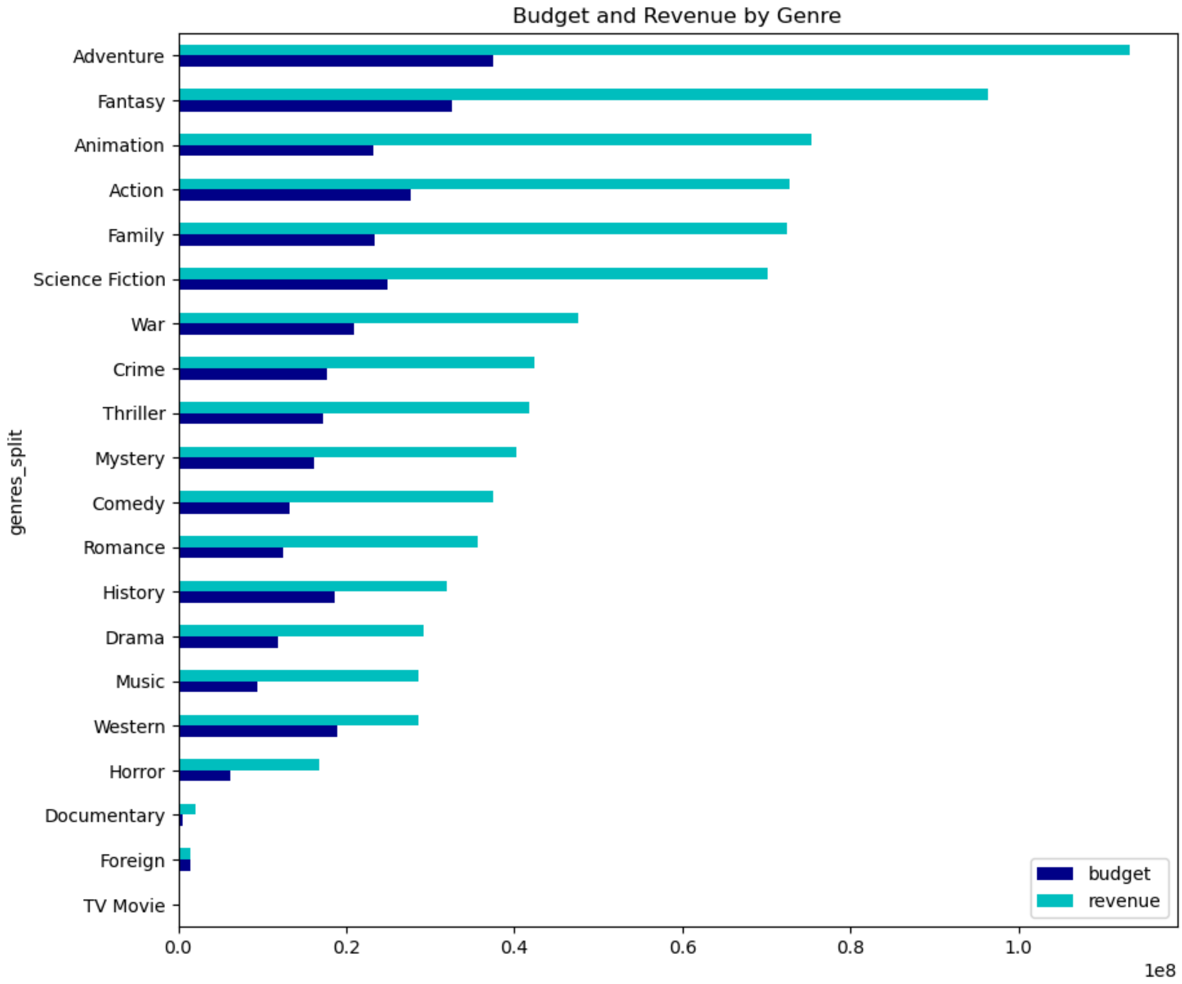
In [87]:
```
genres_avg.sort_values(by = ['revenue','budget'], ascending = True, inplace = True)
genres_avg[['budget', 'revenue']].plot.barh(title = 'Budget and Revenue by Genre',
```

Out[87]:
```
<Axes: title={'center': 'Budget and Revenue by Genre'}, ylabel='genres_split'>
```



In [89]:
```
# Sorting the both columns budget and the revenue at the same time
genres_avg.sort_values('budget').astype('str') + genres_avg.sort_values('revenue').
genres_avg[['budget', 'revenue']].plot.barh(title = 'Budget and Revenue by Genre',
```

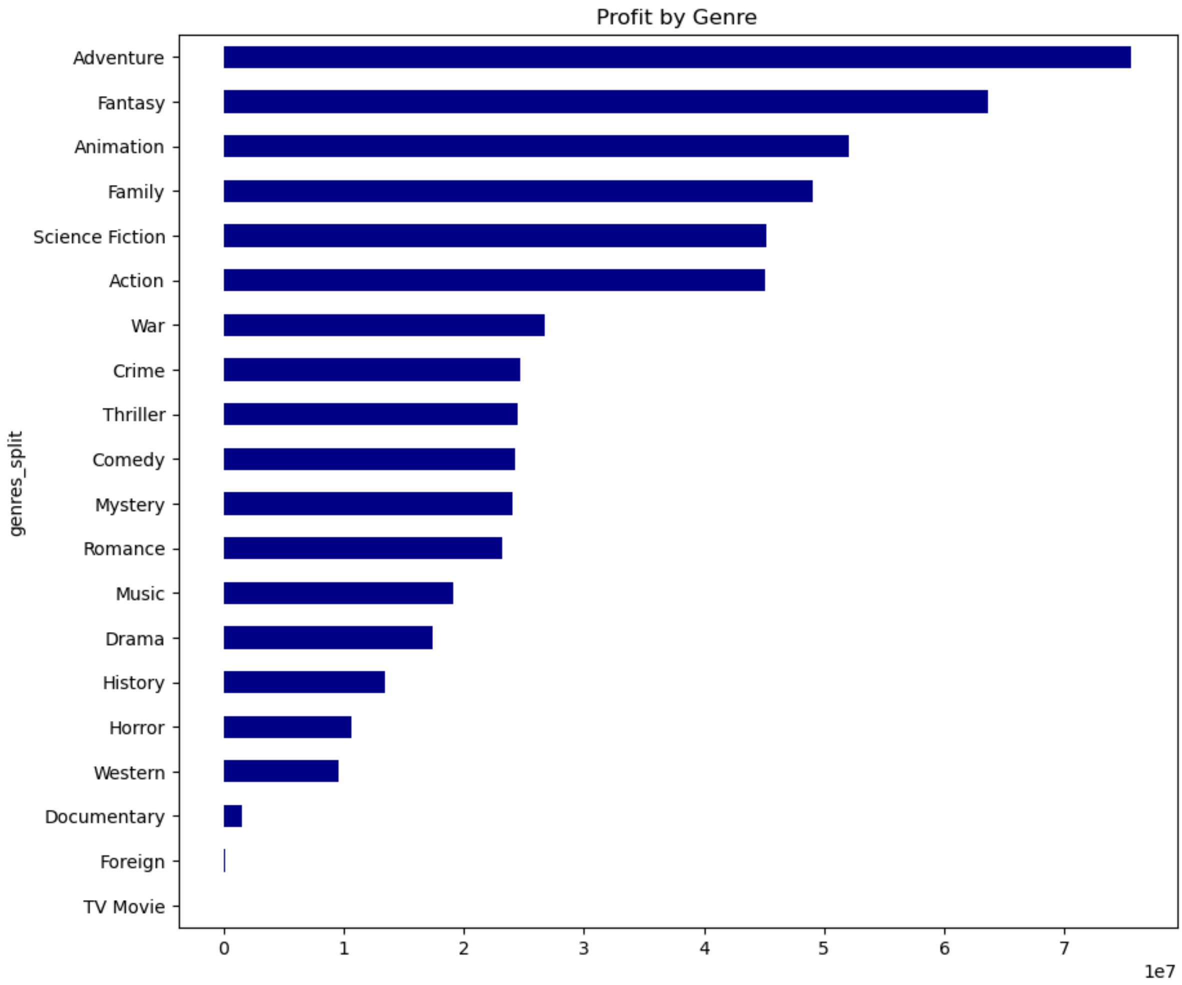
Out[89]:
```
<Axes: title={'center': 'Budget and Revenue by Genre'}, ylabel='genres_split'>
```



1. Which genres have highest profit?

In [90]:
```
genres_avg.sort_values('profit', ascending = True, inplace = True)
genres_avg['profit'].plot.barh(title = 'Profit by Genre', color = 'DarkBlue', figsi
```

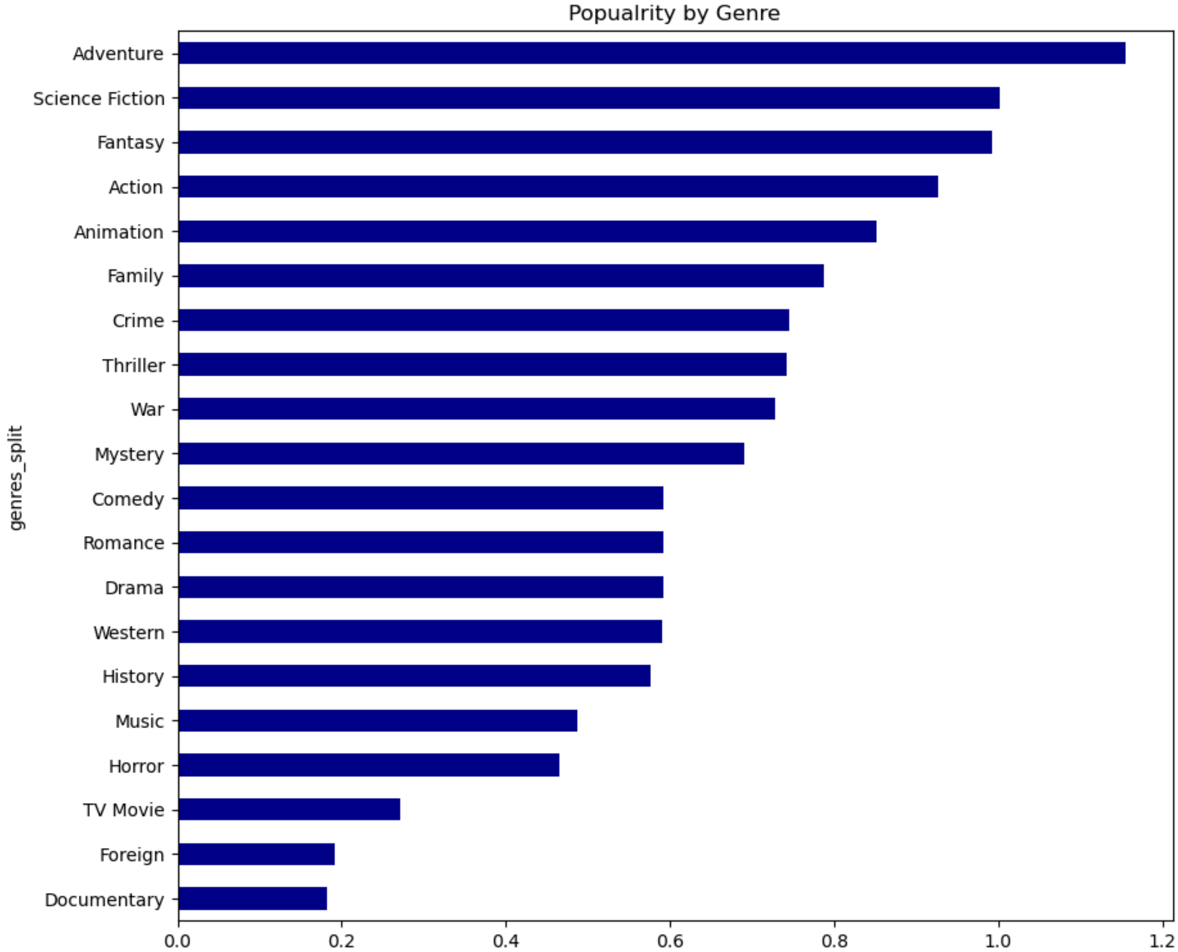
Out[90]:
```
<Axes: title={'center': 'Profit by Genre'}, ylabel='genres_split'>
```



1. Which generes have high avg popularity?

In [92]:
```
genres_avg.sort_values('popularity', ascending = True, inplace = True)
genres_avg['popularity'].plot.barh(title = 'Popualrity by Genre', color = 'DarkBlue
```

Out[92]:
```
<Axes: title={'center': 'Popualrity by Genre'}, ylabel='genres_split'>
```



1. Which geners have highest number of movies with an voting average >= 8?

The solution that is below is my own solution. What I have done is:

1. We need to make sure that the voting average is greater than or equal to 8.
2. That need to be grouped by geners and the taken has to be with the original_title

In [115…

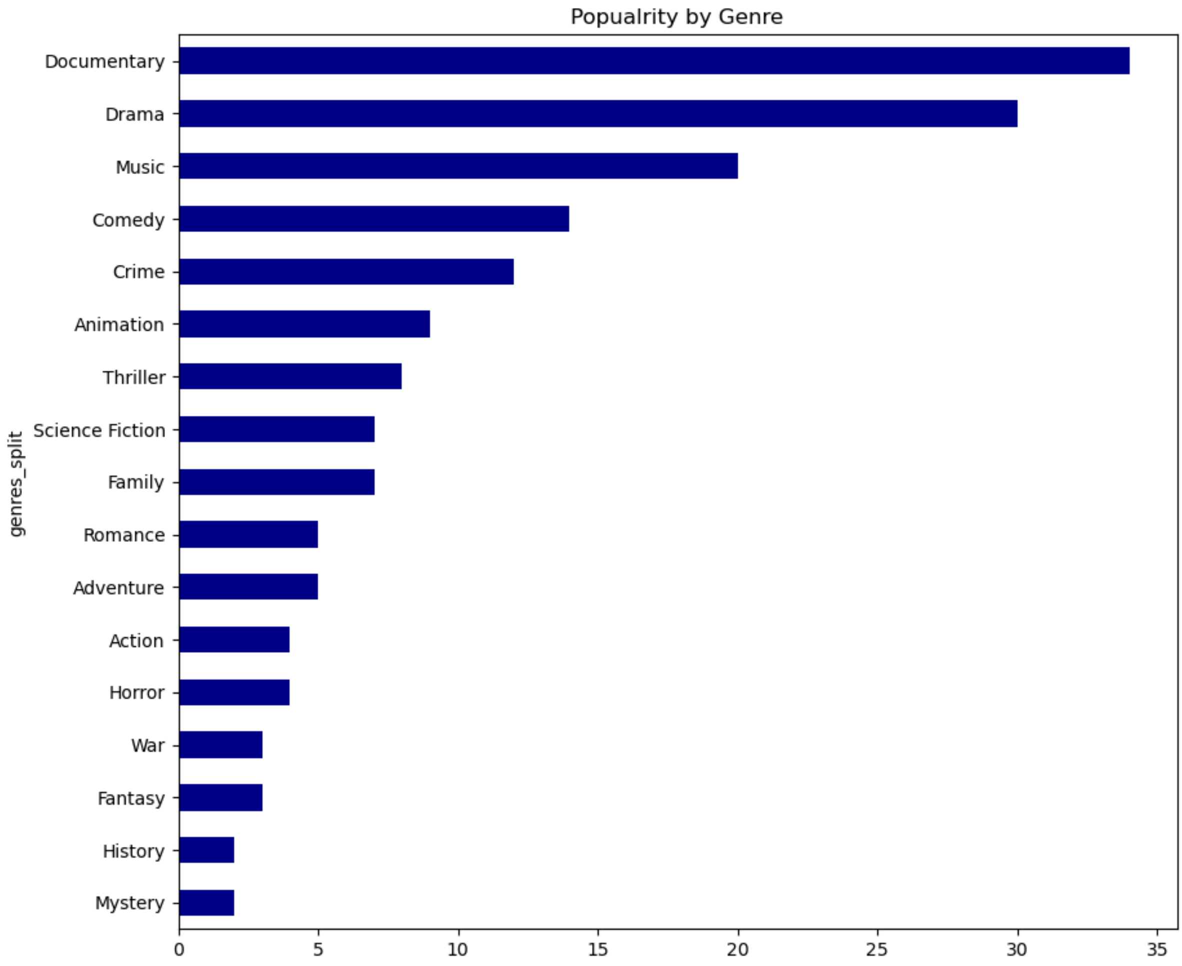
```
avg_greater_than_8 = numerical_data[numerical_data['vote_average'] >= 8].groupby('g
avg_greater_than_8['vote_average'].sort_values(ascending = False)
```

Out[115]:

```
genres_split
Documentary        34
Drama              30
Music              20
Comedy             14
Crime              12
Animation           9
Thriller            8
Family              7
Science Fiction     7
Romance             5
Adventure           5
Action              4
Horror              4
Fantasy             3
War                 3
Mystery             2
History             2
Name: vote_average, dtype: int64
```

In [117…
```
avg_greater_than_8['vote_average'].plot.barh(title = 'Popualrity by Genre', color =
```

Out[117]:
```
<Axes: title={'center': 'Popualrity by Genre'}, ylabel='genres_split'>
```



The solution below is the solution by Alex the Analyst

In [118…
```
vote_fifty = movies_genres[(movies_genres['vote_count'] >= 50) & (movies_genres['vo
vote_zero  = movies_genres[movies_genres['vote_average'] >= 8]
```

In [119…
```
genres_vote = pd.DataFrame(vote_fifty.groupby('genres_split').vote_average.nunique(
               .sort_values('vote_average', ascending = True)
```

In [120…
```
genres_vote
```

Out[120]:

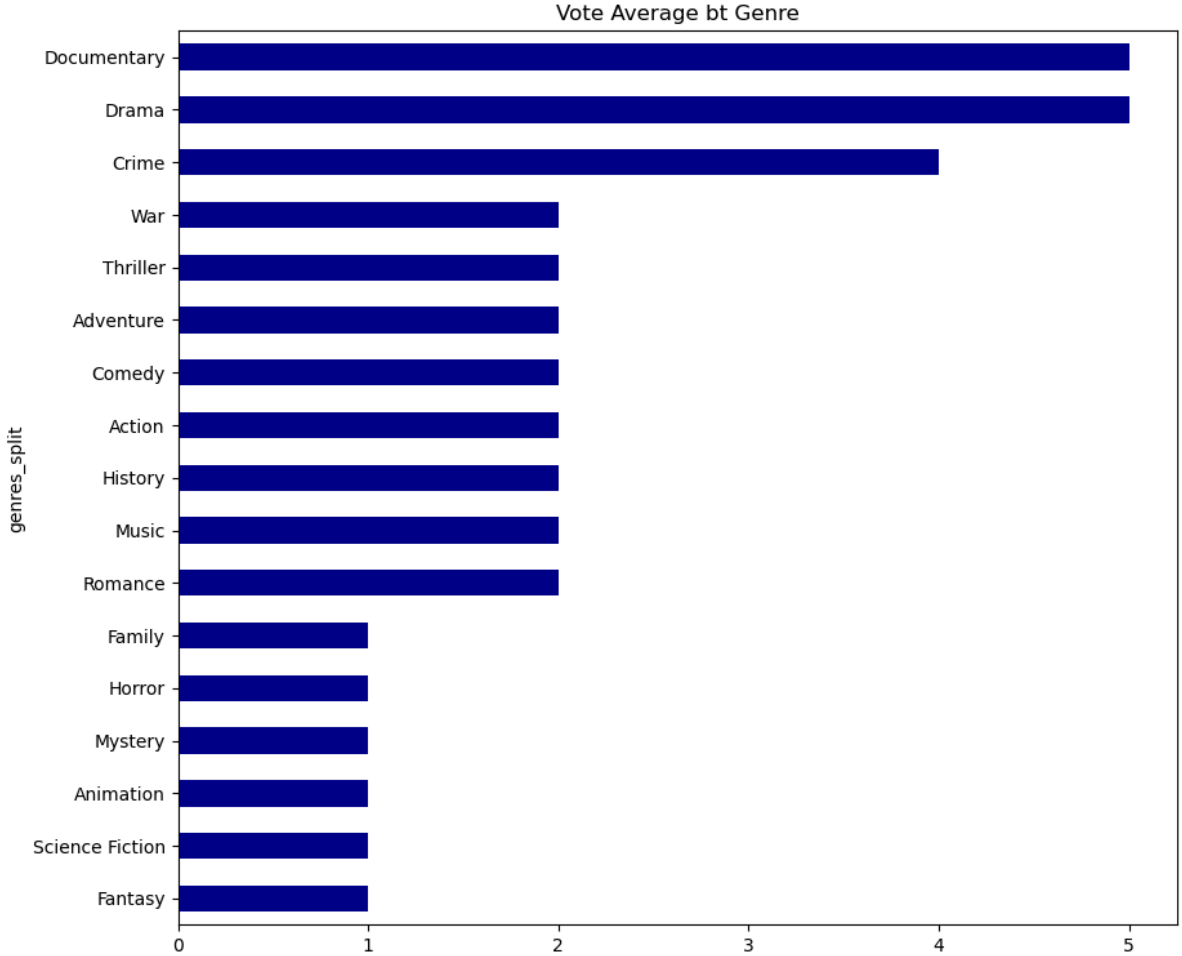
genres_split	vote_average
Fantasy	1
Science Fiction	1
Animation	1
Mystery	1
Horror	1
Family	1
Romance	2
Music	2
History	2
Action	2
Comedy	2
Adventure	2
Thriller	2
War	2
Crime	4
Drama	5
Documentary	5

In [122…

```
genres_vote['vote_average'].plot.barh(title = 'Vote Average bt Genre', color = 'Dar
```

Out[122]:

```
<Axes: title={'center': 'Vote Average bt Genre'}, ylabel='genres_split'>
```



In [124…

```
genres_vote = pd.DataFrame(vote_zero.groupby('genres_split').vote_average.nunique()
              .sort_values('vote_average', ascending = True)
```

In [125…

```
genres_vote
```

Out[125]:

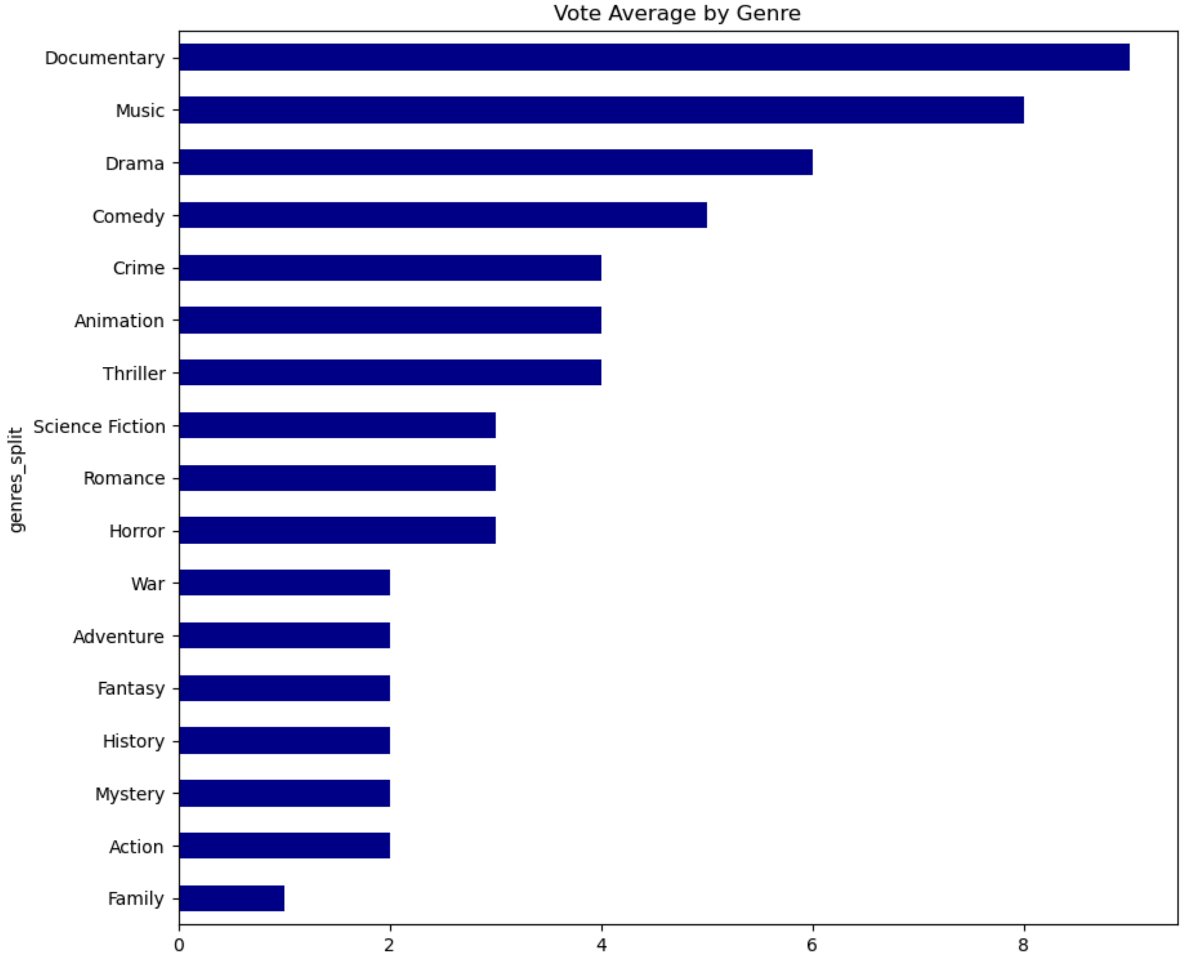
	vote_average
genres_split	
Family	1
Action	2
Mystery	2
History	2
Fantasy	2
Adventure	2
War	2
Horror	3
Romance	3
Science Fiction	3
Thriller	4
Animation	4
Crime	4
Comedy	5
Drama	6
Music	8
Documentary	9

In [126…

```
genres_vote['vote_average'].plot.barh(title = 'Vote Average by Genre', color = 'Dar
```

Out[126]:

```
<Axes: title={'center': 'Vote Average by Genre'}, ylabel='genres_split'>
```



1. The best movies according to vote average return high profit and revenue.

In [127…

```
movies = pd.read_csv(r'E:\Data Analytics\Pandas\pandas mastery\13.real world projec
```

In [158…

```
movies.drop_duplicates(inplace = True)
movies['profit'] = movies['revenue'] - movies['budget']
movies_genre = movies[['popularity', 'budget', 'revenue', 'original_title', 'runtim
                       'vote_count', 'profit']]
```

In [159…

```
movies_genre.head()
```

Out[159]:

	popularity	budget	revenue	original_title	runtime	genres	release_date
0	32.985763	150000000	1513528810	Jurassic World	124	Action|Adventure|Science Fiction|Thriller	6/9/15
1	28.419936	150000000	378436354	Mad Max: Fury Road	120	Action|Adventure|Science Fiction|Thriller	5/13/15
2	13.112507	110000000	295238201	Insurgent	119	Adventure|Science Fiction|Thriller	3/18/15
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	136	Action|Adventure|Science Fiction|Fantasy	12/15/15
4	9.335014	190000000	1506249360	Furious 7	137	Action|Crime|Thriller	4/1/15

In [160…

```
# Checking the correlation between the vote count and (revenue, budget)
# Checking the correlation is because we have a doubt that there are some outliers
```

```
numerical_data = movies_genre.select_dtypes(include = ['int', 'float'])
movies_counted = numerical_data[numerical_data['vote_count'] >= 50]
movies_counted.corr(method = 'spearman')
```

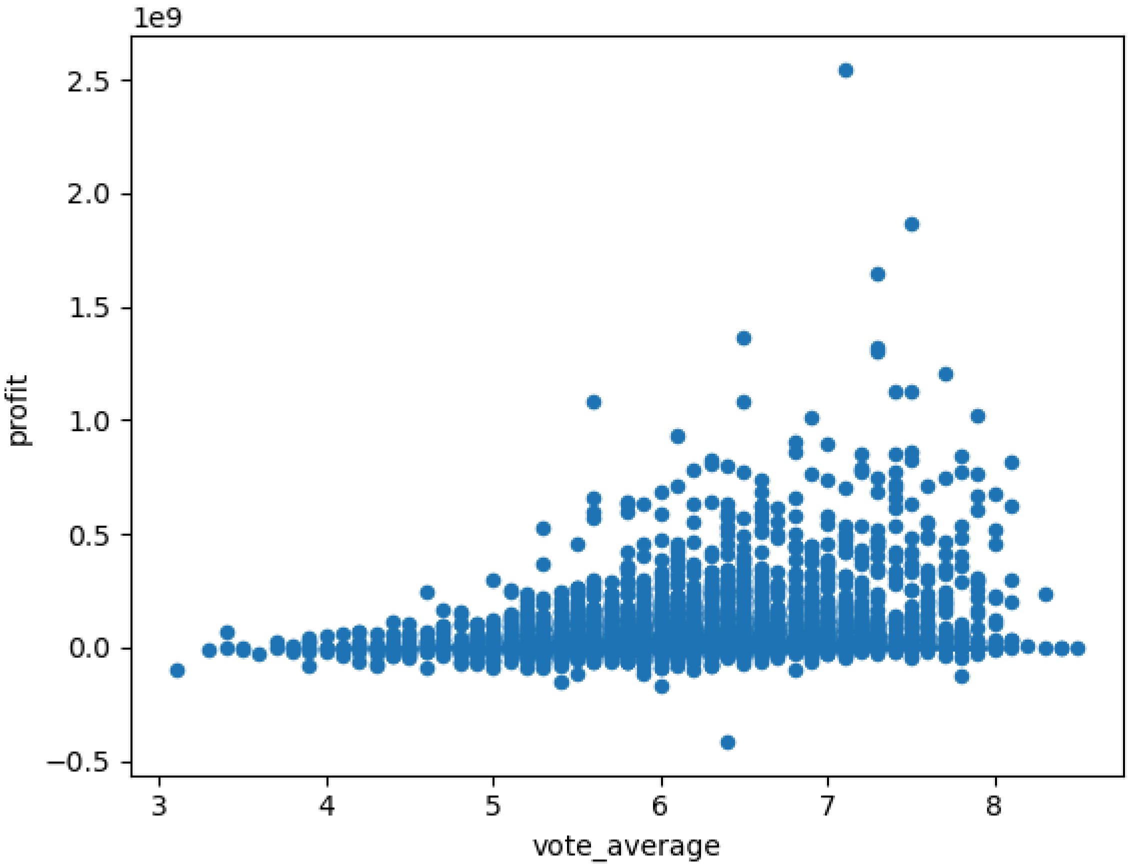
Out[160]:

	popularity	budget	revenue	runtime	vote_average	vote_count	profit
popularity	1.000000	0.485149	0.588739	0.230518	0.188670	0.768966	0.498117
budget	0.485149	1.000000	0.714828	0.360230	-0.044440	0.554586	0.334390
revenue	0.588739	0.714828	1.000000	0.341707	0.111731	0.682656	0.842221
runtime	0.230518	0.360230	0.341707	1.000000	0.285514	0.263460	0.212060
vote_average	0.188670	-0.044440	0.111731	0.285514	1.000000	0.284470	0.198308
vote_count	0.768966	0.554586	0.682656	0.263460	0.284470	1.000000	0.583602
profit	0.498117	0.334390	0.842221	0.212060	0.198308	0.583602	1.000000

In [137…

```
movies_counted.plot.scatter(x = 'vote_average', y = 'profit')
```

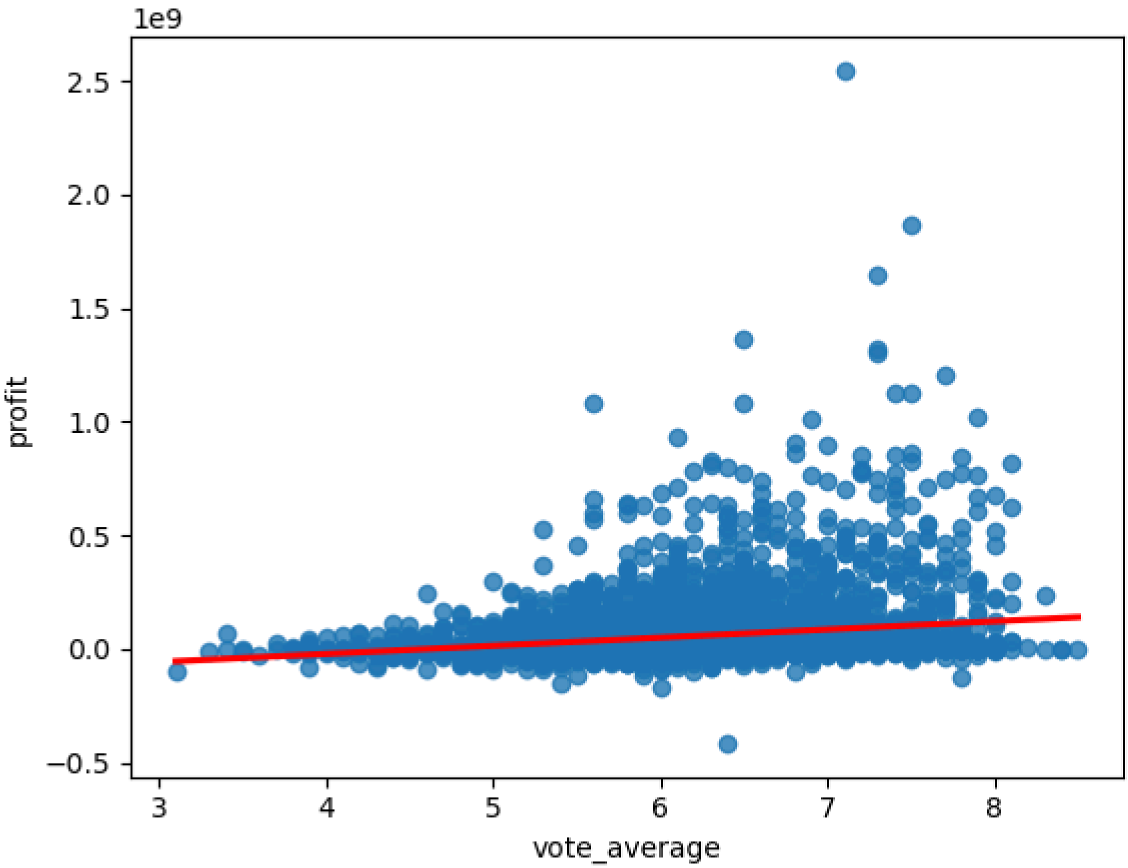
Out[137]: <Axes: xlabel='vote_average', ylabel='profit'>



In [138…

```
# Through this reg plot we know that if there is an increase in the vote_average th
# According to the above graph, it will be the smallest
sns.regplot(x = 'vote_average', y = 'profit', data = movies_counted, line_kws = {'c
```

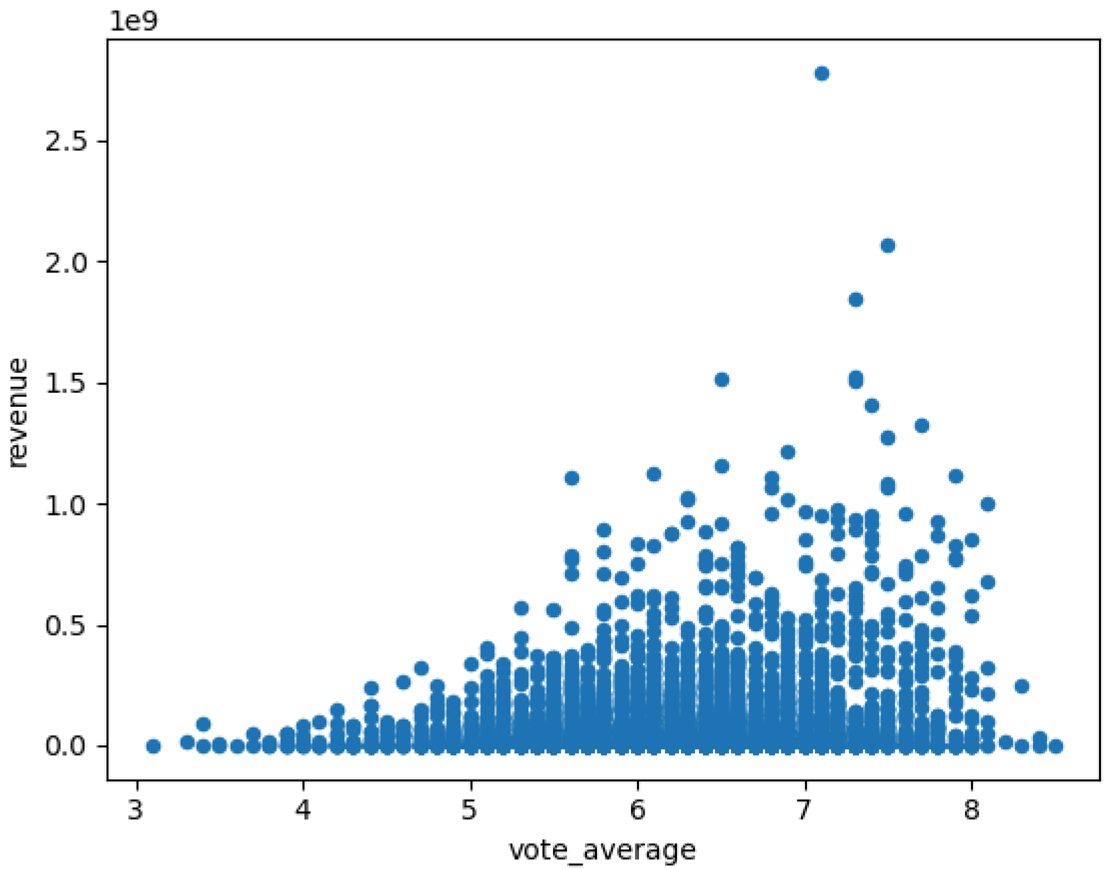
Out[138]: <Axes: xlabel='vote_average', ylabel='profit'>



In [139…

```
movies_counted.plot.scatter(x = 'vote_average', y = 'revenue')
```

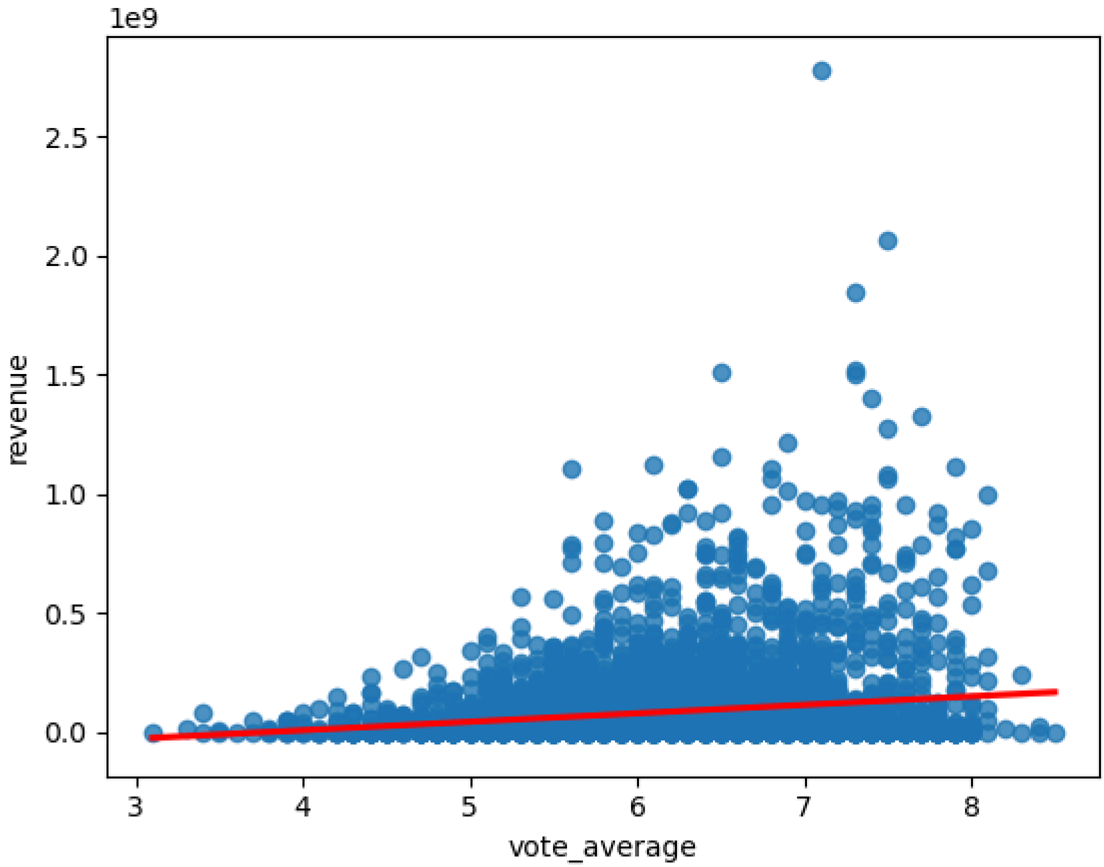
Out[139]: <Axes: xlabel='vote_average', ylabel='revenue'>



In [140…

```
sns.regplot(x = 'vote_average', y = 'revenue', data = movies_counted, line_kws = {'
```

Out[140]: <Axes: xlabel='vote_average', ylabel='revenue'>



1. The best movies according to popularity return high profit and revenue.

In [141…

```
# Checking the correlation between the data
movies_counted.corr()
```

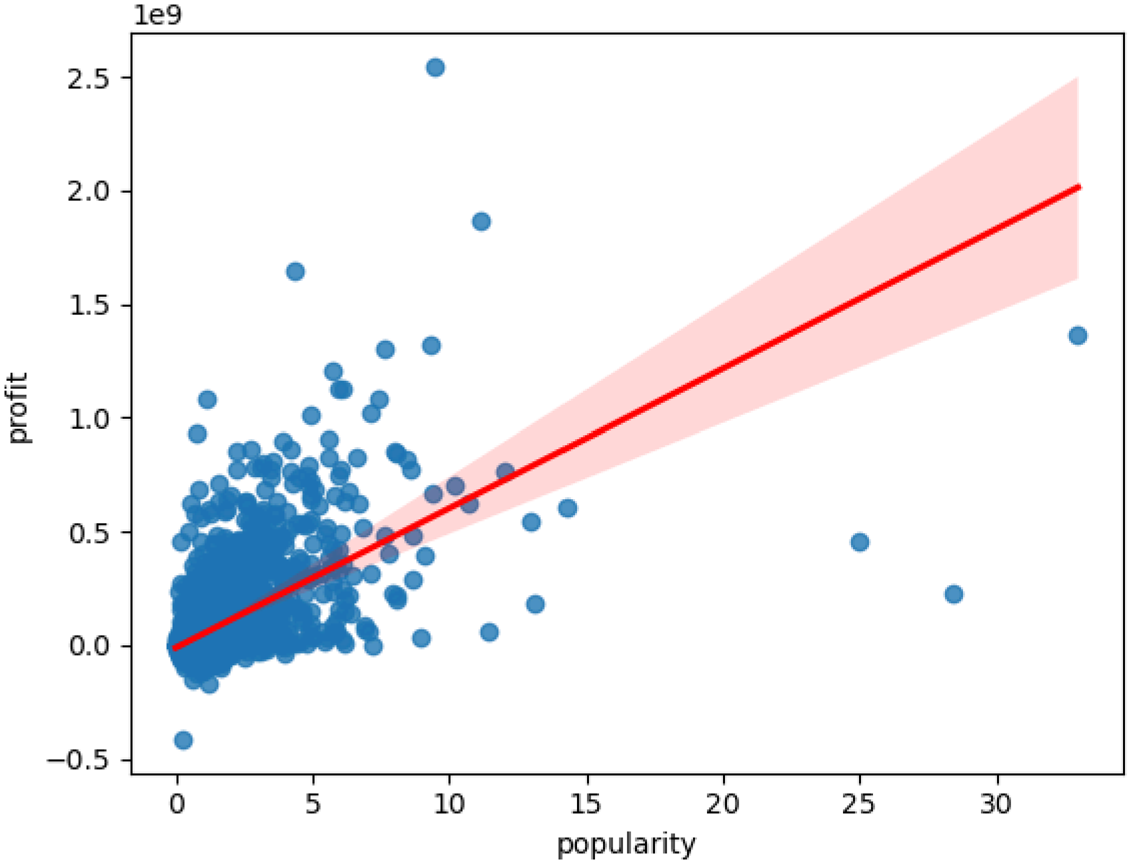
Out[141]:

	popularity	budget	revenue	runtime	vote_average	vote_count	profit
popularity	1.000000	0.458926	0.616012	0.186892	0.246604	0.776101	0.594608
budget	0.458926	1.000000	0.703506	0.263744	-0.004655	0.575313	0.537979
revenue	0.616012	0.703506	1.000000	0.233610	0.171162	0.760077	0.977553
runtime	0.186892	0.263744	0.233610	1.000000	0.210541	0.237413	0.198899
vote_average	0.246604	-0.004655	0.171162	0.210541	1.000000	0.320899	0.204397
vote_count	0.776101	0.575313	0.760077	0.237413	0.320899	1.000000	0.730980
profit	0.594608	0.537979	0.977553	0.198899	0.204397	0.730980	1.000000

In [142…

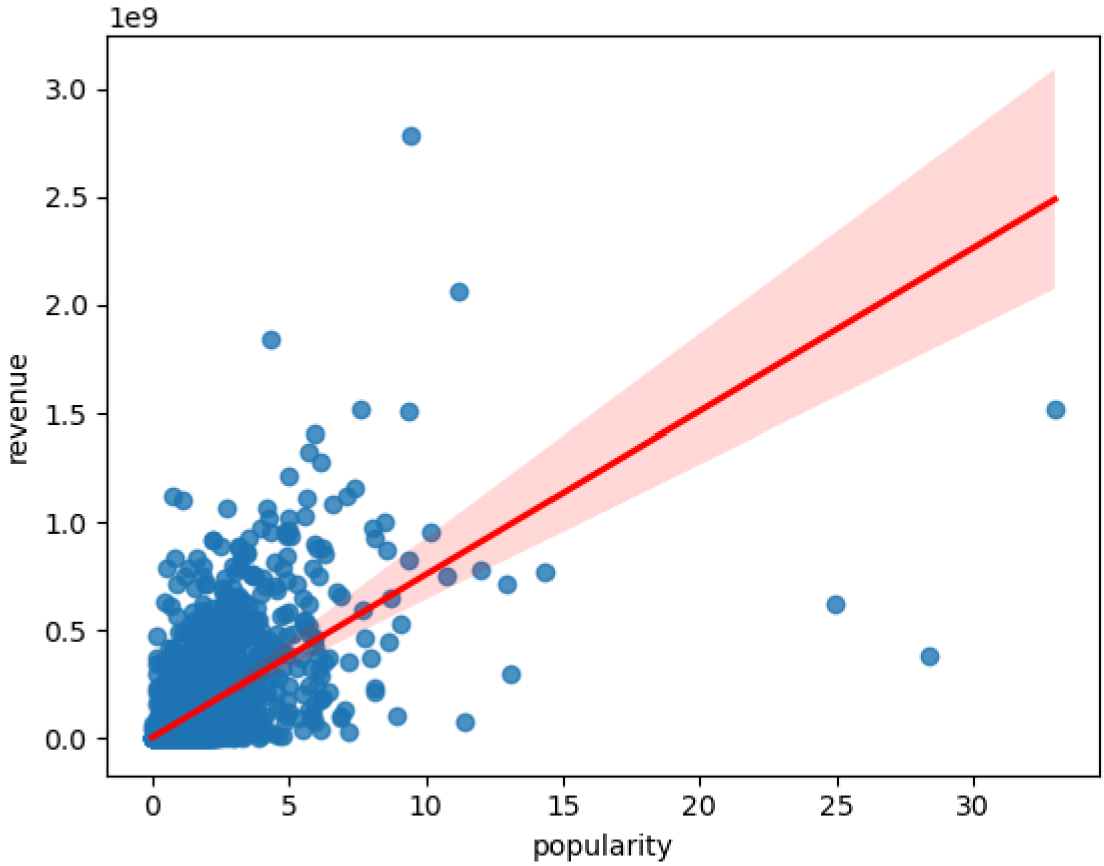
```
sns.regplot(x = 'popularity', y = 'profit', data = movies_counted, line_kws = {'col
```

Out[142]: <Axes: xlabel='popularity', ylabel='profit'>



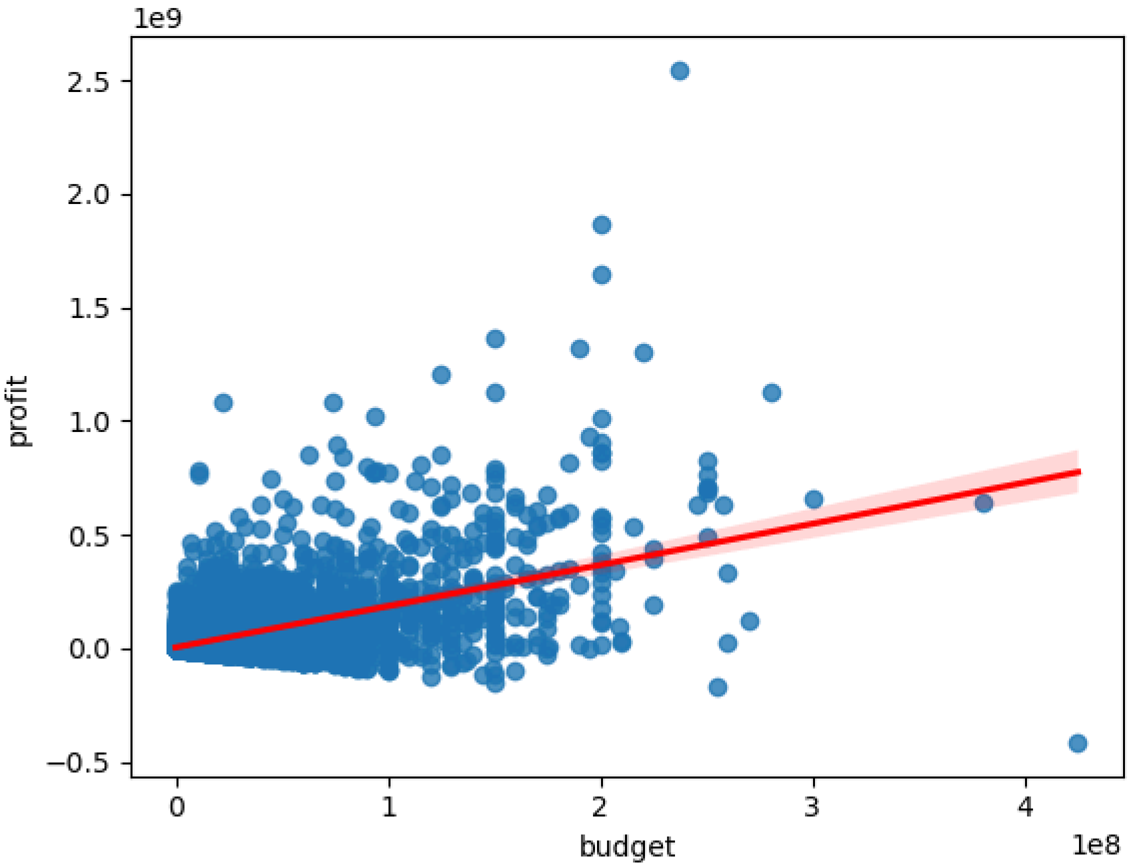
In [144…
```
sns.regplot(x = 'popularity', y = 'revenue', data = movies_counted, line_kws = {'co
```

Out[144]: <Axes: xlabel='popularity', ylabel='revenue'>

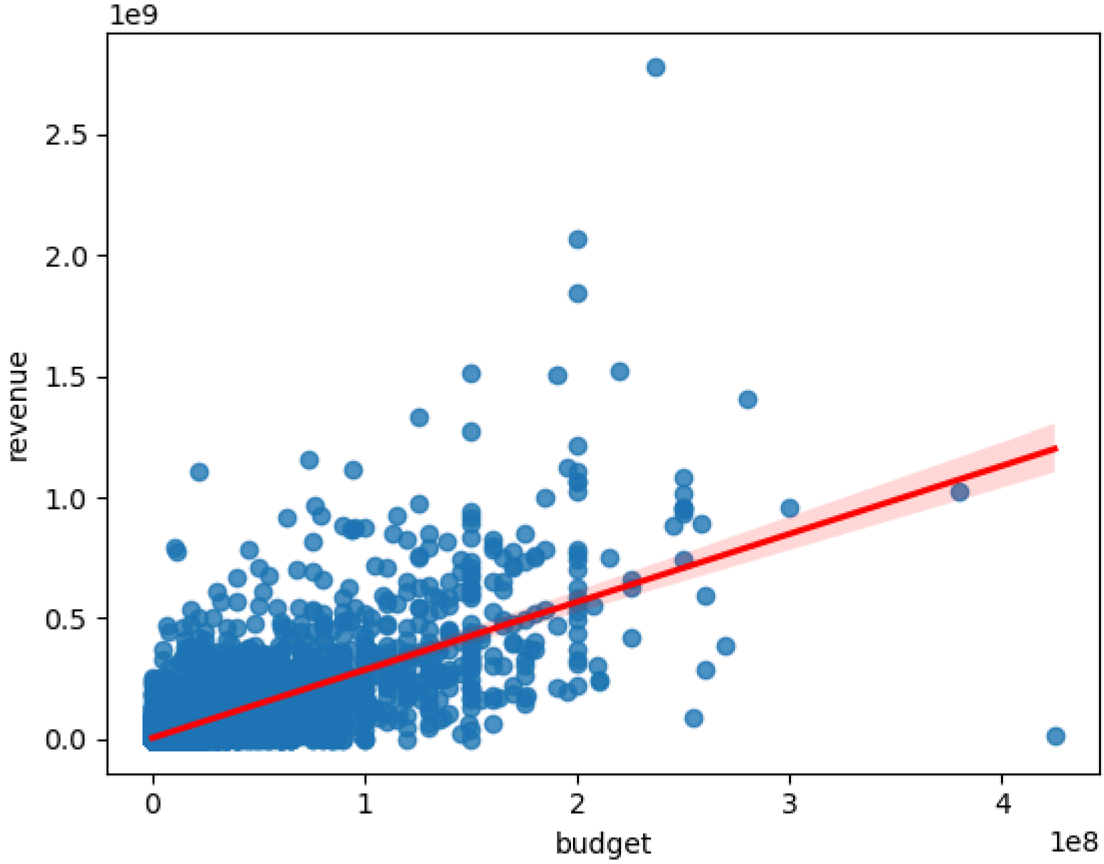


1. Highly budgeted movies return high revenue and profit.

In [145…
```
sns.regplot(x = 'budget', y = 'profit', data = movies_counted, line_kws = {'color':
```
Out[145]:
```
<Axes: xlabel='budget', ylabel='profit'>
```



In [146…
```
sns.regplot(x = 'budget', y = 'revenue', data = movies_counted, line_kws = {'color'
```
Out[146]:
```
<Axes: xlabel='budget', ylabel='revenue'>
```



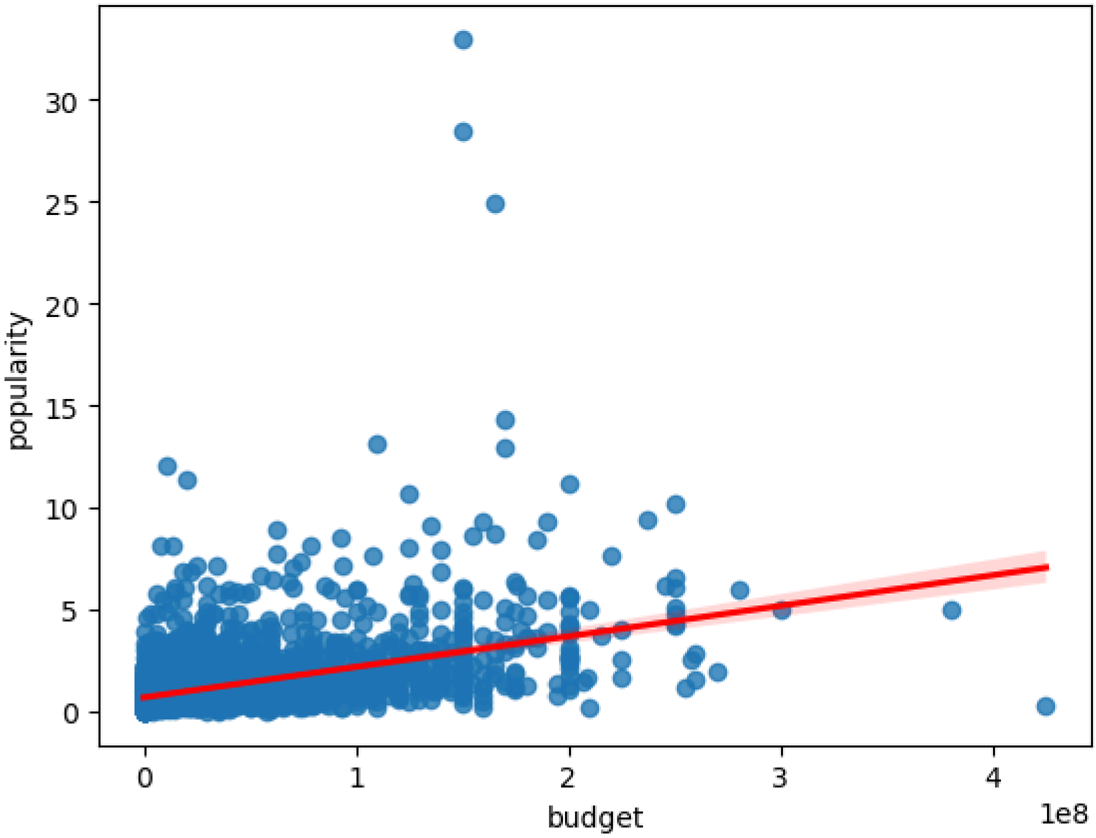
1. Highly budgeted movies have a high popularity.

In [147…

```
sns.regplot(x = 'budget', y = 'popularity', data = movies_counted, line_kws = {'col
```

Out[147]:

```
<Axes: xlabel='budget', ylabel='popularity'>
```



1. Profit per genre per Year

In [173…
```
movies['profit'] = movies['revenue'] - movies['budget']
movies_genre = movies[['popularity', 'budget', 'revenue', 'original_title', 'runtim
                        'vote_count', 'profit']]
release_year = movies['release_year']
release_year
split.name = 'genres_split'
del movies_genres['genres']
movies_genres = movies_genres.join(split)
```

Out[173]:
```
0        2015
1        2015
2        2015
3        2015
4        2015
         ...
10861    1966
10862    1966
10863    1966
10864    1966
10865    1966
Name: release_year, Length: 10865, dtype: int64
```

In [174…
```
numerical_data['original_title'] = movies_genre['original_title'].copy()
release_year.name = 'release_year'
numerical_data = numerical_data.join(release_year)
```

In [175…
```
numerical_data
```

Out[175]:

	popularity	budget	revenue	runtime	vote_average	vote_count	profit	genre
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	Adv
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	S
0	32.985763	150000000	1513528810	124	6.500000	5562	1363528810	
1	28.419936	150000000	378436354	120	7.100000	6185	228436354	
...	...	...	...	...	...	...	...	
10863	0.065141	0	0	94	6.500000	11	0	M
10863	0.065141	0	0	94	6.500000	11	0	Co
10864	0.064317	0	0	80	5.400000	22	0	
10864	0.064317	0	0	80	5.400000	22	0	Co
10865	0.035919	19000	0	74	1.500000	15	-19000	

26955 rows × 10 columns

In [178…

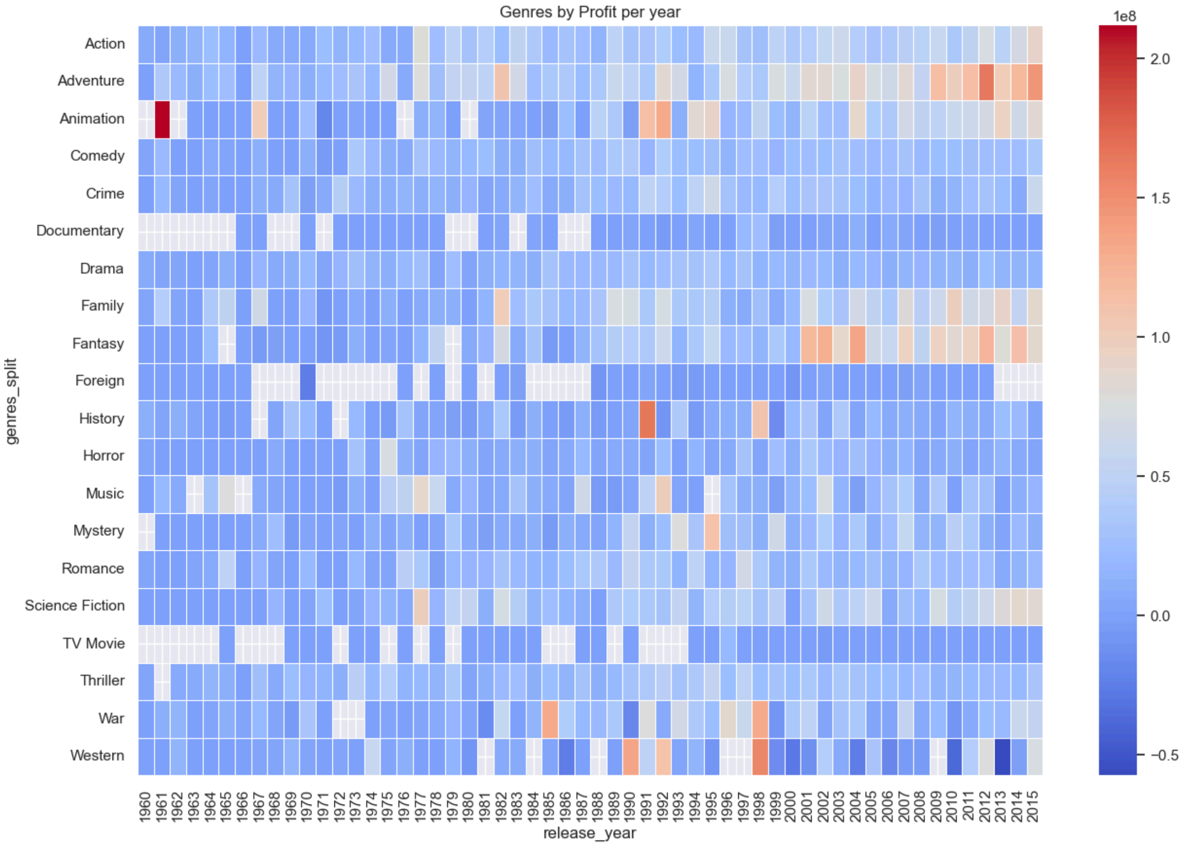
```
time_genre = pd.DataFrame(numerical_data.groupby(['release_year','genres_split'])['
```

In [180…

```
final_genre = pd.pivot_table(time_genre, values = 'profit', index = ['genres_split'
```

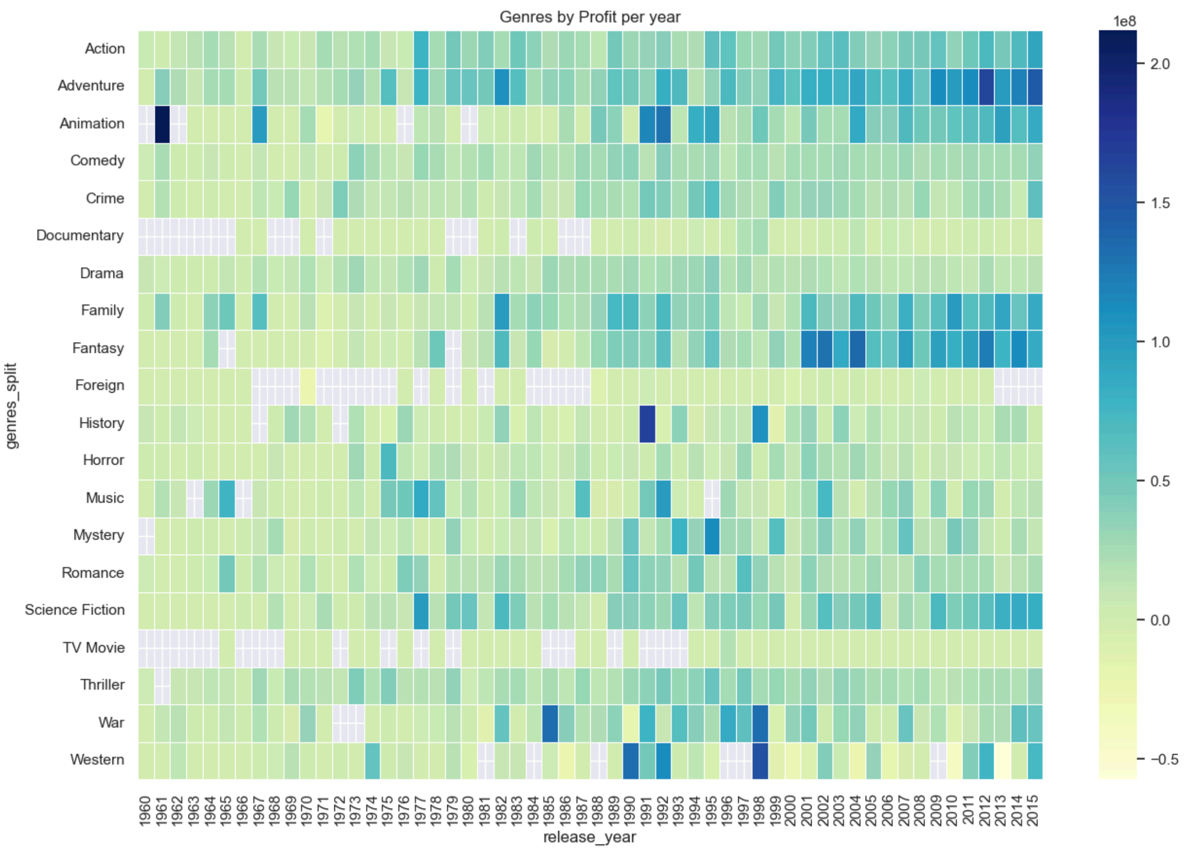
In [183…

```
sns.set(rc = {'figure.figsize' : (15,10)})
sns.heatmap(final_genre, cmap = 'coolwarm', linewidths = 0.5)
plt.title('Genres by Profit per year')
plt.show()
```



In [184…

```
sns.set(rc = {'figure.figsize' : (15,10)})
sns.heatmap(final_genre, cmap = 'YlGnBu', linewidths = 0.5)
plt.title('Genres by Profit per year')
plt.show()
```



In [ ]: