INTRODUCTION

For this project we have chosen the data set, TMDb movie data, which contains data from approximately 10,0000 movies with information such as movie name, duration, director, cast and more.

IMDb receives more than 200 million visitors per month and its database contains information regarding more than 3 million movies and television shows, in addition to having information on more than 6 million actors and people on the production team.
In addition to getting information about movies and TV series, you can check the billboard (subject to availability in your city), see information about upcoming releases, watch movie shorts, watch scenes from TV shows and watch TV channel programming (also subject to availability in your city).
An interesting function is that you can consult a movie, see how people who have gone to see it rated it and, furthermore, you can see how that rating is broken down by age and gender (male or female), in such a way that you can know what sectors of the public preferred the film; Perhaps what teens didn't like was over 40, for example.

To start, we load all the necessary tools to carry out the analysis:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Once the data has been accessed in Python, we look at what the data set looks like. That is, what columns we have, and their interpretation to be able to ask the questions that might interest us:

```
# Access CSV

df0= pd.read_csv('/Users/user01/Downloads/tmdb-movies.csv')
df0.head(5)
list(df0)  # column names
```

```
>>> df0.head(5)
      id    imdb_id  popularity  ...  release_year    budget_adj   revenue_adj
0  135397  tt0369610   32.985763  ...          2015  1.379999e+08  1.392446e+09
1   76341  tt1392190   28.419936  ...          2015  1.379999e+08  3.481613e+08
2  262500  tt2908446   13.112507  ...          2015  1.012000e+08  2.716190e+08
3  140607  tt2488496   11.173104  ...          2015  1.839999e+08  1.902723e+09
4  168259  tt2820852    9.335014  ...          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
>>>
```

Questions

1) Is there a relationship between the genre and the popularity of a movie?
2) How do movie incomes compare between 1985 and 2015?
3) Which genres raise the most money?

Cleaning, Data transformation

Then we can discard any columns that are not relevant to the analysis. In addition, we must determine if any column contains values that may affect the analysis, such as duplicate and missing values. In such a case, we will have to take the pertinent steps to process those data.

Using the .info () method we can obtain information to know if any column contains null values.

```
>>>df0.info()
```

```
---  ------                --------------  -----
 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)
```

For the case of our data set, we observe that the columns
'Imdb', 'genres' among others, contain null values, since in total we have 10,866 lines and for these fields, the number of non-null lines is less than 10,866.

```
df1 =
df0.drop(['imdb_id','budget','original_title','cast','homepage','director','
tagline','keywords','overview','runtime','production_companies','release_dat
e','budget_adj','revenue_adj'],axis=1)df1.head(5)
df1.info()
df1.describe
```

Furthermore, the existence of outliers may also affect the analysis. To check for outliers we use the .describe () method on our dataframe.

An outlier is a value in your data that is either extremely high or low in comparison with the other data.

	id	popularity	budget	...	release_year	budget_adj	revenue_adj
count	10866.000000	10866.000000	1.086600e+04	...	10866.000000	1.086600e+04	1.086600e+04
mean	66064.177434	0.646441	1.462570e+07	...	2001.322658	1.755104e+07	5.136436e+07
std	92130.136561	1.000185	3.091321e+07	...	12.812941	3.430616e+07	1.446325e+08
min	5.000000	0.000065	0.000000e+00	...	1960.000000	0.000000e+00	0.000000e+00
25%	10596.250000	0.207583	0.000000e+00	...	1995.000000	0.000000e+00	0.000000e+00
50%	20669.000000	0.383856	0.000000e+00	...	2006.000000	0.000000e+00	0.000000e+00
75%	75610.000000	0.713817	1.500000e+07	...	2011.000000	2.085325e+07	3.369710e+07
max	417859.000000	32.985763	4.250000e+08	...	2015.000000	4.250000e+08	2.827124e+09

[8 rows x 10 columns]

We note that there are outliers in the **'popularity'** column

To find values 0 we look at the information about the minimum value of each column, in order to detect zeros.

Thus we note that the columns: **'budget'**, **'revenue'**, **'runtime'**, **'budget_adj'**, **'revenue_adj'** have values 0.

To start cleaning, we discard all the columns that will not be useful for analysis.

```
df1 =
df0.drop(['imdb_id','budget','original_title','cast','homepage','director','
tagline','keywords','overview','runtime','production_companies','release_dat
e','budget_adj','revenue_adj'],axis=1)
```

Since for our analysis we only consider the period from 1985 - 2015, the information is extreme.

```
idx0=df1['release_year']>=1985
df2=df1[idx0]
```

After applying the above methods again. We observe that there are only null values in **'genres'**. Values 0 in the column **'revenue'** which is what interests us.

```
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            9570 non-null   int64
 1   popularity    9570 non-null   float64
 2   revenue       9570 non-null   int64
 3   genres        9549 non-null   object
 4   vote_count    9570 non-null   int64
 5   vote_average  9570 non-null   float64
 6   release_year  9570 non-null   int64
dtypes: float64(2), int64(4), object(1)
memory usage: 598.1+ KB
```

	id	popularity	revenue	vote_count	vote_average	release_year
count	9570.000000	9570.000000	9.570000e+03	9570.000000	9570.000000	9570.000000
mean	72518.605747	0.668774	4.266137e+07	232.612539	5.938140	2004.973772
std	96133.841489	1.035179	1.226776e+08	600.224732	0.943646	8.218718
min	5.000000	0.000065	0.000000e+00	10.000000	1.500000	1985.000000
25%	10708.500000	0.213632	0.000000e+00	17.000000	5.400000	1999.000000
50%	22804.500000	0.395980	0.000000e+00	42.000000	6.000000	2007.000000
75%	87823.250000	0.744108	2.596620e+07	163.000000	6.600000	2012.000000
max	417859.000000	32.985763	2.781506e+09	9767.000000	9.200000	2015.000000

In the next step, we filter the 0 values:

```
df_0=df2.query('revenue == 0')
```

	id	popularity	revenue	vote_count	vote_average	release_year
count	5203.000000	5203.000000	5203.0	5203.000000	5203.000000	5203.000000
mean	92851.751682	0.331482	0.0	42.971170	5.789679	2006.121276
std	106483.542069	0.313523	0.0	70.492167	1.031229	7.845790
min	17.000000	0.000065	0.0	10.000000	1.500000	1985.000000
25%	14792.000000	0.148910	0.0	14.000000	5.100000	2002.000000
50%	36234.000000	0.266572	0.0	21.000000	5.800000	2008.000000
75%	152740.500000	0.425224	0.0	42.000000	6.500000	2012.000000
max	414419.000000	8.411577	0.0	1143.000000	9.200000	2015.000000

Now, since there are a large number of lines (5203) whose value for the column **'revenue'** is 0. There are several ways to deal with these values, one would be to assign the average value of income and another simply discard them. But in this case, to avoid entering values that could affect the analysis, we discard all these lines.

```
df2['revenue'] = df2['revenue'].replace(0, np.NaN)
df2.info() # verify null values
col = ['revenue']
df2.dropna(subset = col, how = 'any', inplace = True)
```

As a next step, we check for duplicate values, and if there are, we discard these values as well.

```
dups = df2[df2.duplicated()]
print(dups)
df2.drop_duplicates(inplace=True)
```

Thus, our dataset remains like this after these steps.

```
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            4366 non-null   int64
 1   popularity    4366 non-null   float64
 2   revenue       4366 non-null   float64
 3   genres        4366 non-null   object
 4   vote_count    4366 non-null   int64
 5   vote_average  4366 non-null   float64
 6   release_year  4366 non-null   int64
```

	id	popularity	revenue	vote_count	vote_average	release_year
count	4366.000000	4366.000000	4.366000e+03	4366.000000	4366.000000	4366.000000
mean	48294.365323	1.070745	9.351085e+07	458.637884	6.115277	2003.605131
std	75278.429035	1.390968	1.680365e+08	830.598781	0.791760	8.442098
min	5.000000	0.001117	2.000000e+00	10.000000	2.100000	1985.000000
25%	8836.750000	0.401545	7.097993e+06	50.000000	5.600000	1997.000000
50%	12906.500000	0.703144	3.335307e+07	161.000000	6.100000	2005.000000
75%	50213.750000	1.235331	1.051772e+08	468.750000	6.700000	2011.000000
max	417859.000000	32.985763	2.781506e+09	9767.000000	8.400000	2015.000000

VISUALIZATION

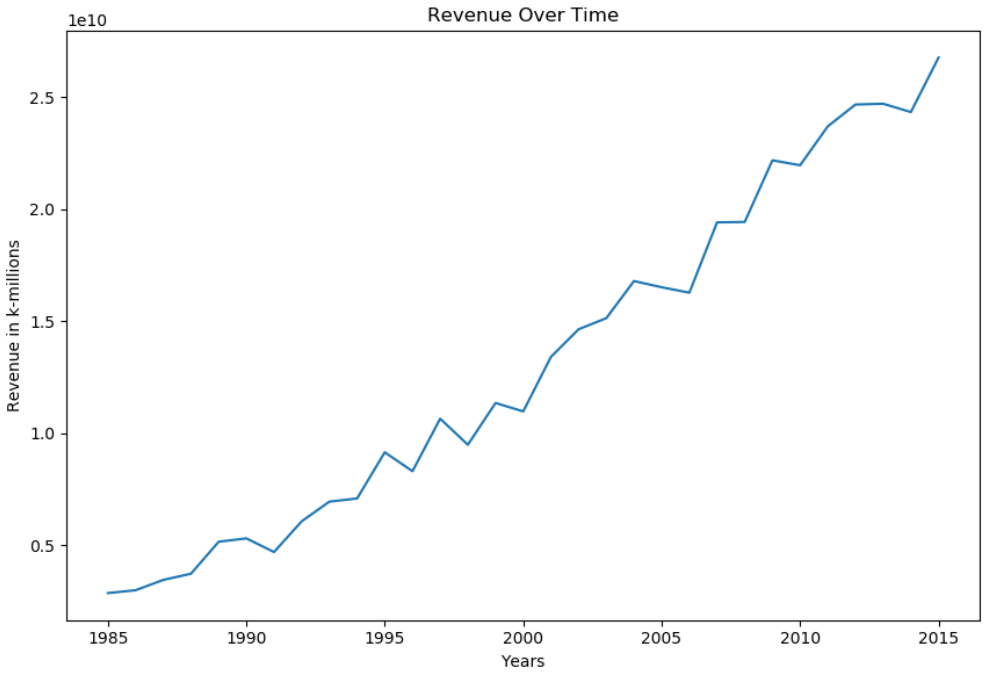
Once we have our dataset ready, we move on, and start doing a visual analysis that allows us to answer our questions.

First, to do the analysis on the income of all the films over time, we grouped all the films by year. Then we add up all the income per year.

```
df_revenue_by_year = df2.groupby(['release_year'])['revenue'].sum()
```

And we plot these results using the matplotlib library:

```
plt.title("Revenue Over Time")
plt.xlabel("Years")
plt.ylabel("Revenue in k-millions")
```



We note here that the film industry has had a very pronounced growth over time.

Now, to do the analysis considering the genres, remember that the genres column had several values delimited by '|', so it is necessary to perform a transformation to handle this data.

```
df3 = df2
df3['genres'] = df3['genres'].str.split('|', expand = True)
df3['genres'].head(5)
list(df3)
```

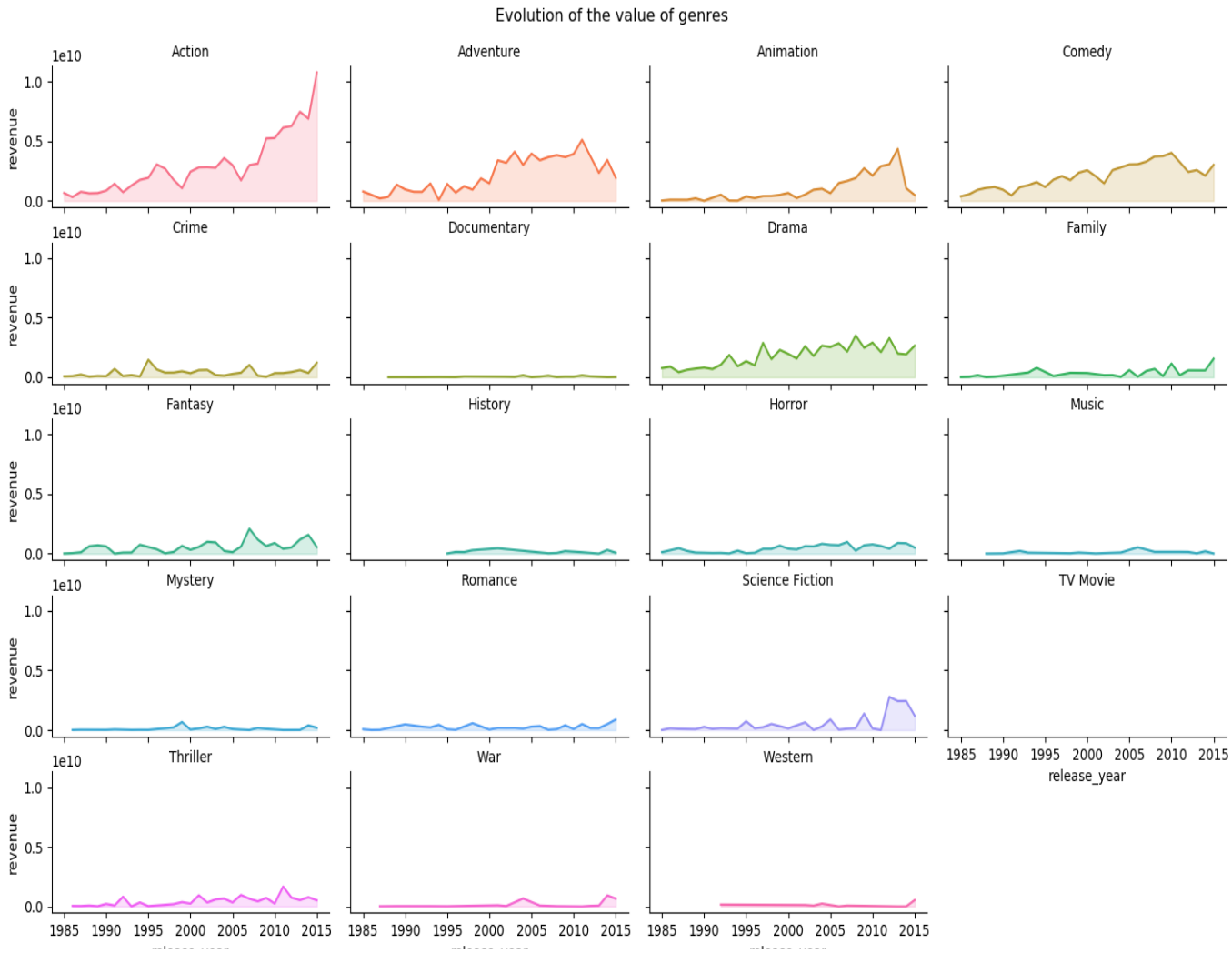
For the same reason that this column had different values in more than 1 row, we only consider the first value, as the main gender and it is the one on which we will base ourselves to make the analysis.

Then, we group the data by gender, by year of release, and add all the revenue:

```
df_genres = df3
df_genres_revenue = df_genres.groupby(['genres',
'release_year'])['revenue'].sum()
df_genres_revenue = df_genres_revenue.to_frame().reset_index()
```

We use seaborn to build the graph:

```
g = sns.FacetGrid(df_genres_revenue, col='genres', hue='genres', col_wrap=4,
)
g = g.map(plt.plot, 'release_year', 'revenue')
g = g.map(plt.fill_between, 'release_year',
'revenue',alpha=0.2).set_titles("{col_name} Genres")
g = g.set_titles("{col_name}")
plt.subplots_adjust(top=0.92)
g = g.fig.suptitle('Evolution of the value of genres')
plt.show()
```



After looking at how the income of the films is over the years, by genre, we note that 5 genres stand out:

Action, comedy, animation, drama and adventure.

Particularly, the action genre is the one that has income with the most significant growth over the years. And from our results we see that it is the genre that generates the most income.

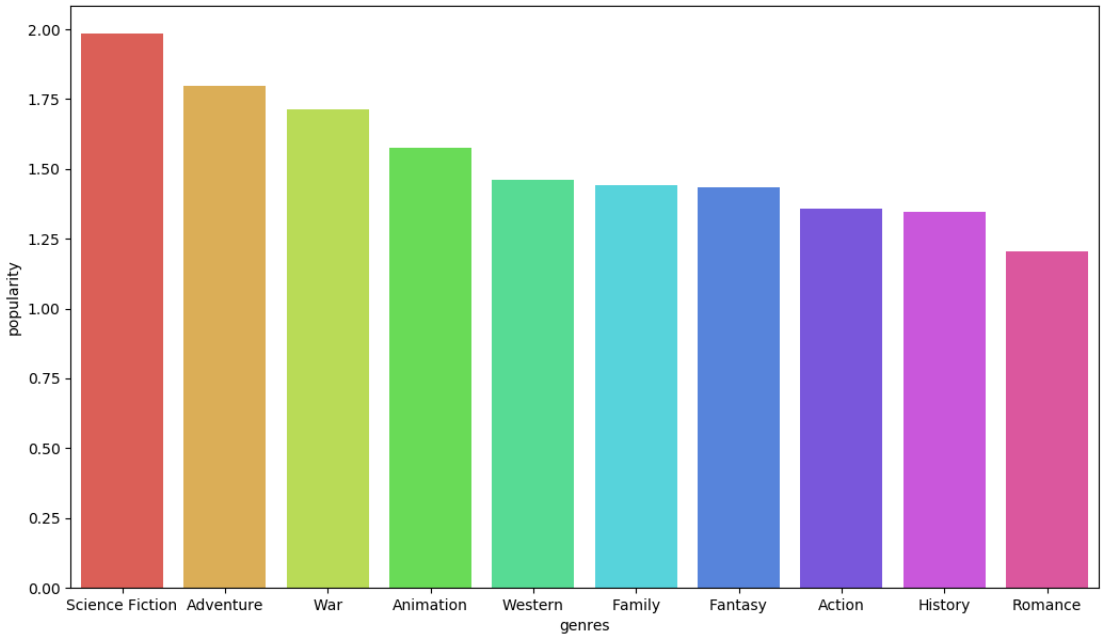
Now, we do the analysis that involves popularity by gender. For this, we group our data by gender and then obtain the average popularity for these. We extract subsets of the 10 most popular and after the 10 least popular.

```
df4 = df3
df_pop = df4.groupby(['genres'])['popularity'].mean()
df_pop = df_pop.to_frame().reset_index()
most_pop = df_pop.sort_values(by = ['popularity'], ascending = False)
most10_pop = most_pop.head(10)
less_pop = df_pop.sort_values(by = ['popularity'])
less10_pop = less_pop.head(10)
```

Then we build the graphs for these two sets:

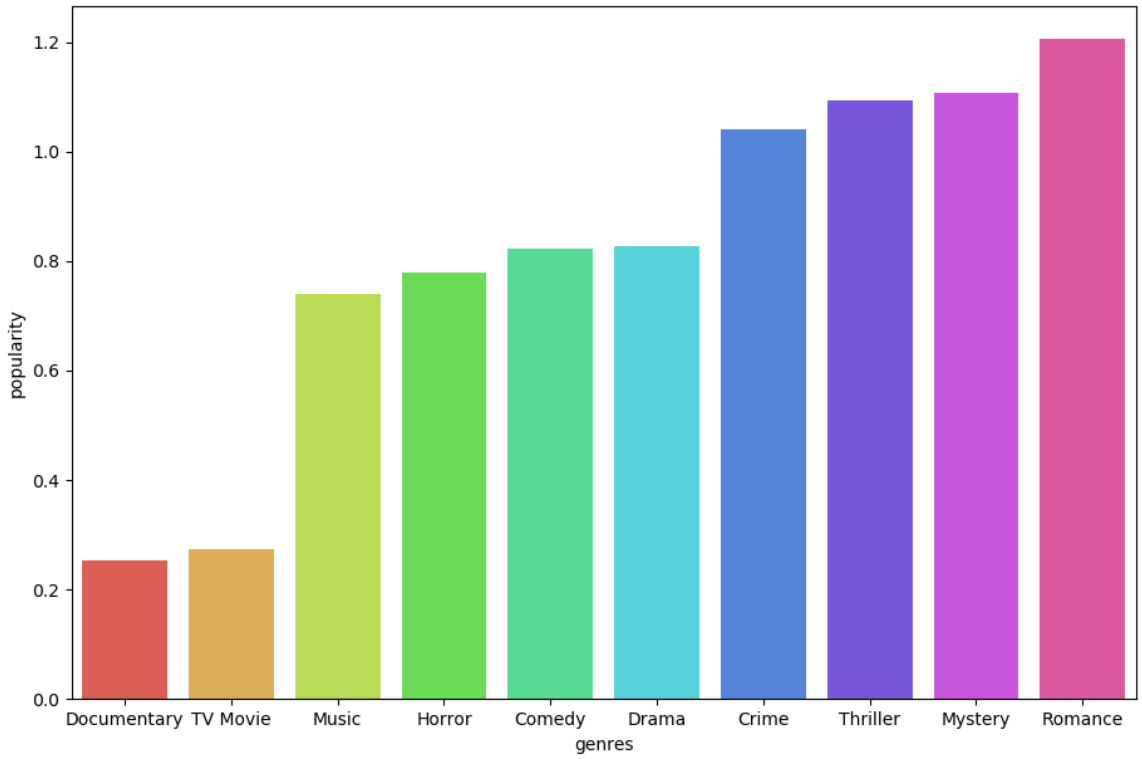
1) Most popular genres:

```
sns.barplot(x = "genres", y = "popularity", data=most10_pop, palette =
"hls", capsize = 0.05, saturation = 8, errcolor = "gray", errwidth = 2, ci =
"sd")
plt.show()
```



2) Less popular genres:

```
sns.barplot(x = "genres", y = "popularity", data=less10_pop, palette =
"hls", capsize = 0.05, saturation = 8, errcolor = "gray", errwidth = 2, ci =
"sd")
plt.show()
```



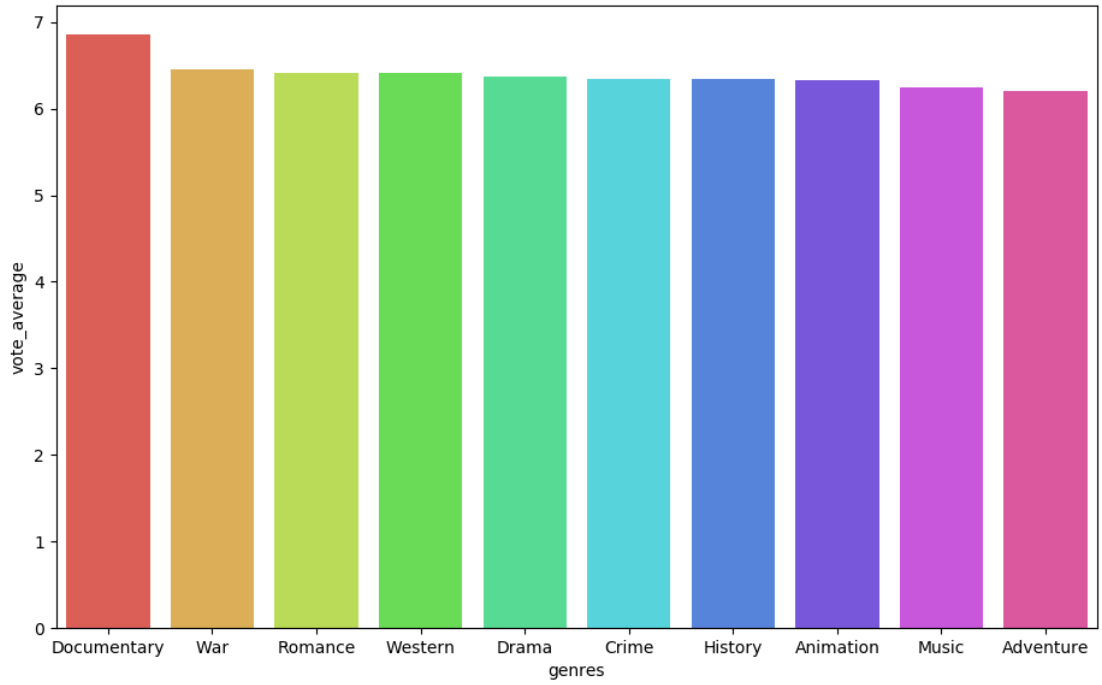
When analyzing the results of popularity by gender, we see that taking into account the popularity average, things are a bit different, since some genres whose income is not as great as those of the aforementioned genres but are very popular, as is the science or war genre. So there is not a very close relationship between popularity and income.

Additionally, we also provide charts for vote averages by gender.

```
df5 = df3
df_vote = df4.groupby(['genres'])['vote_average'].mean()
df_vote = df_vote.to_frame().reset_index()
most_vote = df_vote.sort_values(by = ['vote_average'], ascending = False)
most10_vote = most_vote.head(10)
less_vote = df_vote.sort_values(by = ['vote_average'])
less10_vote = less_vote.head(10)
```

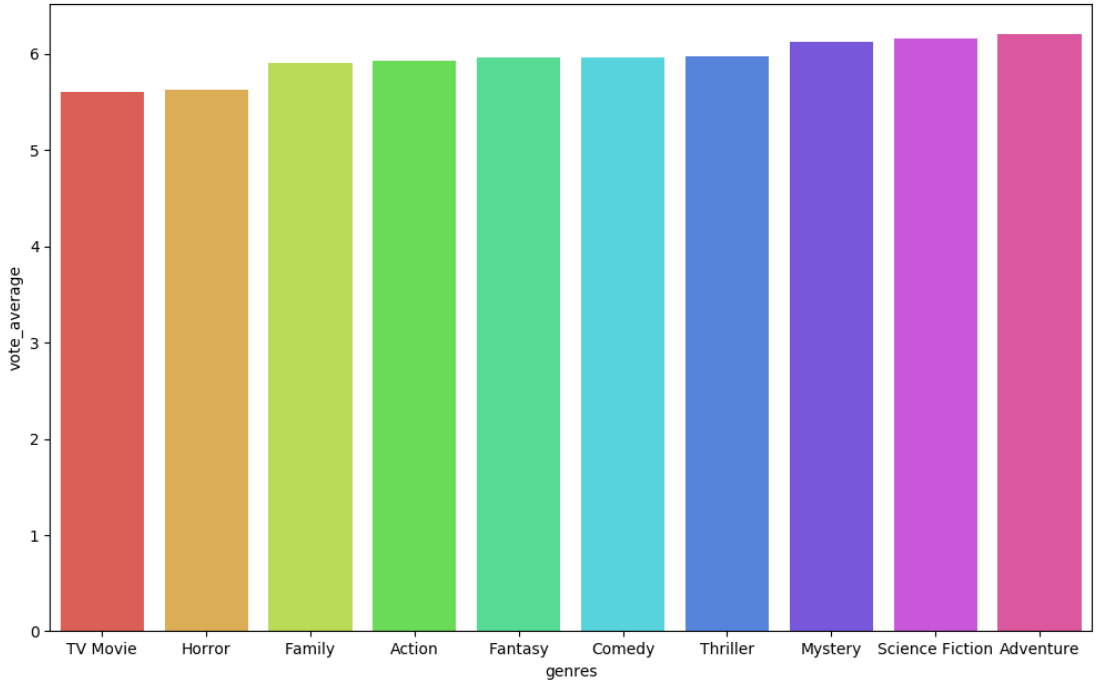
1) Most voted

```
sns.barplot(x = "genres", y = "vote_average", data=most10_vote, palette =
"hls", capsize = 0.05, saturation = 8, errcolor = "gray", errwidth = 2, ci =
"sd")
plt.show()
```



2) Less voted

```
sns.barplot(x = "genres", y = "vote_average", data=less10_vote, palette =
"hls", capsize = 0.05, saturation = 8, errcolor = "gray", errwidth = 2, ci =
"sd")
plt.show()
```



CONCLUSIONS

From our analysis we have discovered that the income of the films does not depend entirely on popularity, since from all genres, given our results, we conclude that the most popular genre is science fiction, which does not generate as much income as per example the genre 'Action'.

In addition, there has been a great growth in this industry, which seems reasonable, since over time, this industry has become accessible to everyone. In addition to that the new technology facilitates access to these services.