

**TASK**

**Exploratory Data Analysis on Imdb Movies Set**

[](https://www.hyperiondev.com/)

**Introduction**

### Context

Dataset Source: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

## About Dataset

What can we say about the success of a movie before it is released? Are there certain companies (Pixar?) that have found a consistent formula? Given that major films costing over $100 million to produce can still flop, this question is more important than ever to the industry. Film aficionados might have different interests. Can we predict which films will be highly rated, whether or not they are a commercial success?

This is a great place to start digging in to those questions, with data on the plot, cast, crew, budget, and revenues of several thousand films.

The below information is present in the dataset

* Budget
* genres
* homepage
* id
* original title
* overview
* popularity
* production companies
* production countries
* release date
* revenue
* runtime
* spoken languages
* status
* tagline
* title
* vote average
* vote count

**DATA CLEANING**

### Reading Dataset

Uploaded the dataset into the Jupyter notebook as “df” dataset utilizing Pandas pd.read\_csv method. After uploading, utilized the below methods to preview and understand the dataset:

#Load the movie dataset and create their dataframes

df = pd.read\_csv('movies.csv')

df.head()

Data Cleansing:

Clean the data. Identify columns that are redundant or unnecessary.

It is always easier to make decisions based on data which is relevant and concise. Removing the following columns ['keywords', 'homepage', 'status', 'tagline', 'original\_language', 'overview', 'production\_companies', 'original\_title'] from the data set as they will not be used in the analysis.

df.drop(['keywords','homepage','status','tagline','original\_language','overview','production\_companies','original\_title'], axis = 1, inplace = True)

Remove any duplicate rows:

df = df.drop\_duplicates(keep='first')

df.head()

Remove zero budget/revenue movies:

Some movies in the database have zero budget or zero revenue which implies that their values have not been recorded or some information is missing. Discarding such entries from the dataframe.

df.drop(df[df['budget'] == 0].index, inplace=True)

df.drop(df[df['revenue'] == 0].index, inplace=True)

Changing Date formats and data types:

To manipulate the columns easily, it is important that we make use of the python objects. Changing the release date column into Date format and extract the year from the date. This will help us in analysing yearly data.

# Change the release\_date column to DateTime column

df['release\_date'] = pd.to\_datetime(df['release\_date'], format='%Y-%m-%d')

# Extract the release year from every release date

df['release\_year'] = pd.DatetimeIndex(df['release\_date']).year

df.release\_year

Change budget and revenue columns format to integer using numpy’s int64 method.

df['budget'] = df['budget'].apply(np.int64)

On checking the dataset, we see that genre, keywords, production\_companies, production\_countries, spoken\_languages are in the JSON format which will make it difficult to manipulate the dataframe. Now let’s flatten these columns into a format that can be easily interpreted. Writing a generic function to parse JSON columns.

def parse\_col\_json(column, key):

    """

    Args:

        column: string

            name of the column to be processed.

        key: string

            name of the dictionary key which needs to be extracted

    """

    for index,i in zip(df.index,df[column].apply(json.loads)):

        list1=[]

        for j in range(len(i)):

            list1.append((i[j][key]))# the key 'name' contains the name of the genre

        df.loc[index,column]=str(list1)

parse\_col\_json('genres', 'name')

parse\_col\_json('spoken\_languages', 'name')

parse\_col\_json('production\_countries', 'name')

**MISSING DATA**

After the above data cleaning, checking for any missing values:

missing\_data = df.isnull().sum()

missing\_data

No missing values observed

**DATA STORIES AND VISUALISATIONS**

Identify relationships between variables / features:

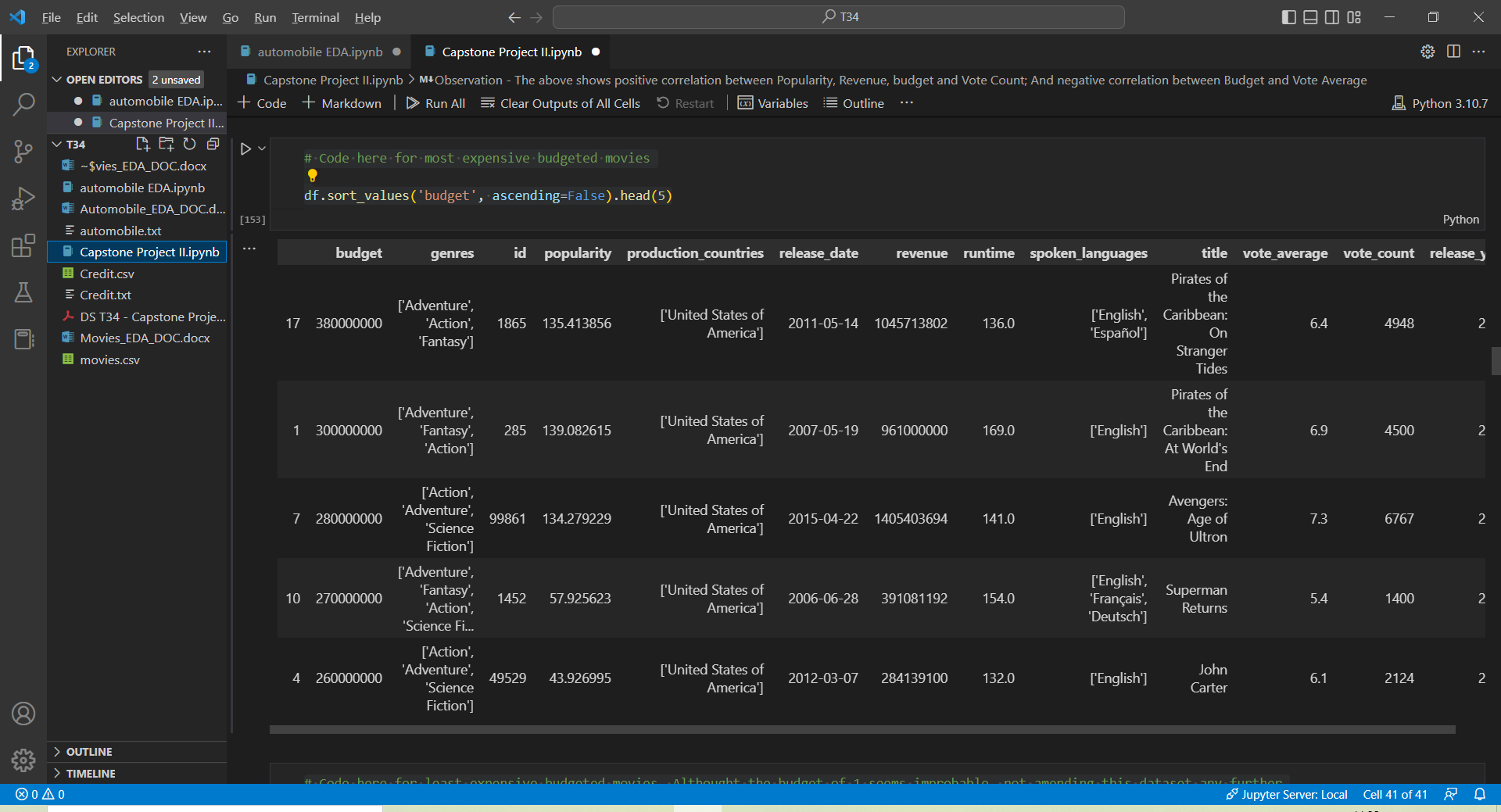
The main goal here is to identify and create relationships which can help to build ideas. Defined few questions which can help identify some relationships to explore.

Question - Which are the 5 most expensive movies? How do the most expensive and cheapest movies compare? Exploring the most expensive movies help you explore if some movies are worth the money spent on them based on their performance and revenue generated.

# Code here for most expensive budgeted movies

df.sort\_values('budget', ascending=False).head(5)

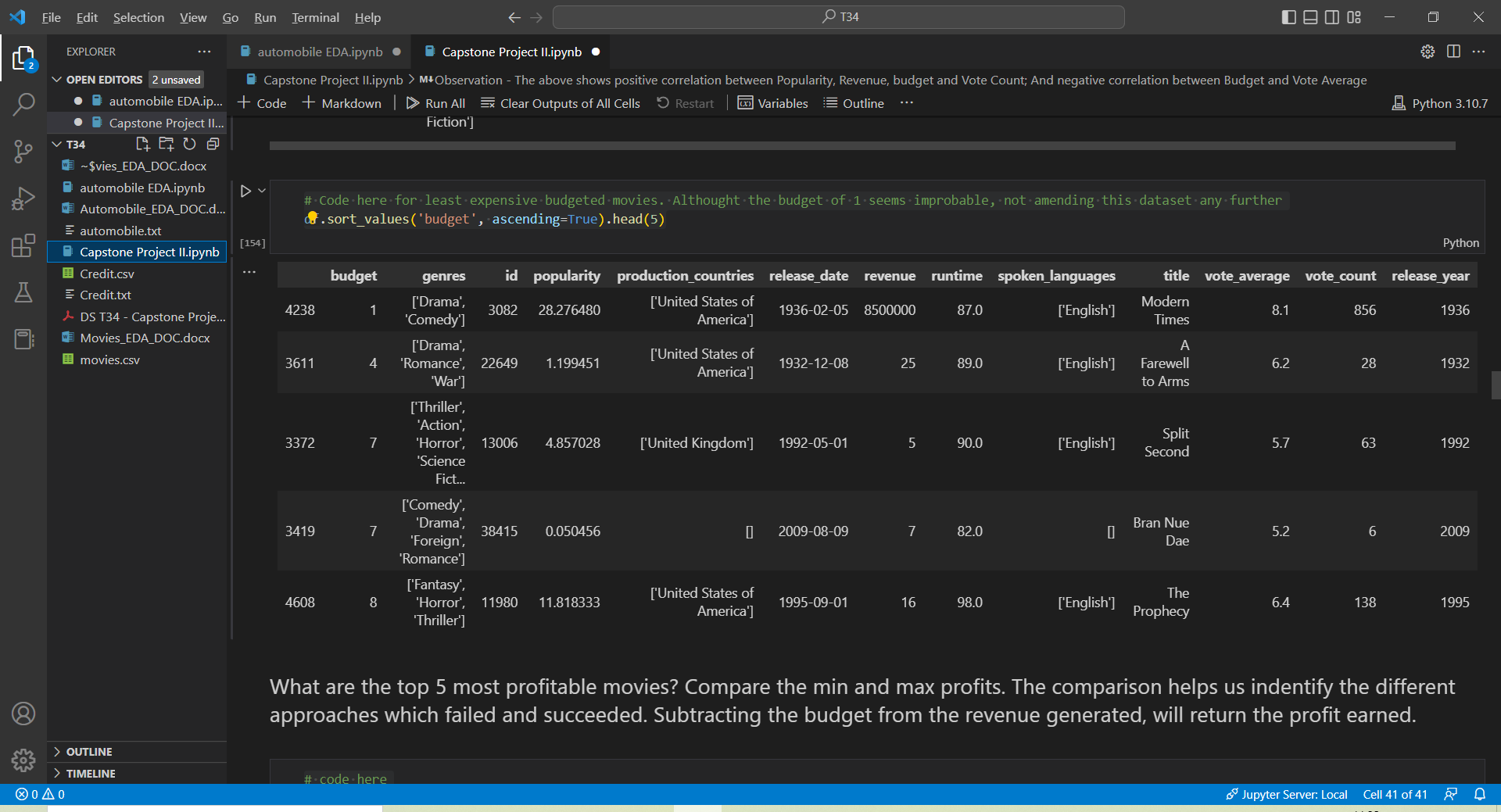
**Table showing the most Expensive Movies:**



# Code here for least expensive budgeted movies. Although the budget of 1 seems improbable, not amending this dataset any further

df.sort\_values('budget', ascending=True).head(5)

**Table showing the least Expensive Movies:**



**Observations**: 1) The most expensive movies are all produced in USA, while UK had a place in least expensive table. Although the budget of 1 seems improbable, not amending this dataset any further

1. Most expensive movies are all from ‘Adventure’ genre and from the recent decades (2000’s)

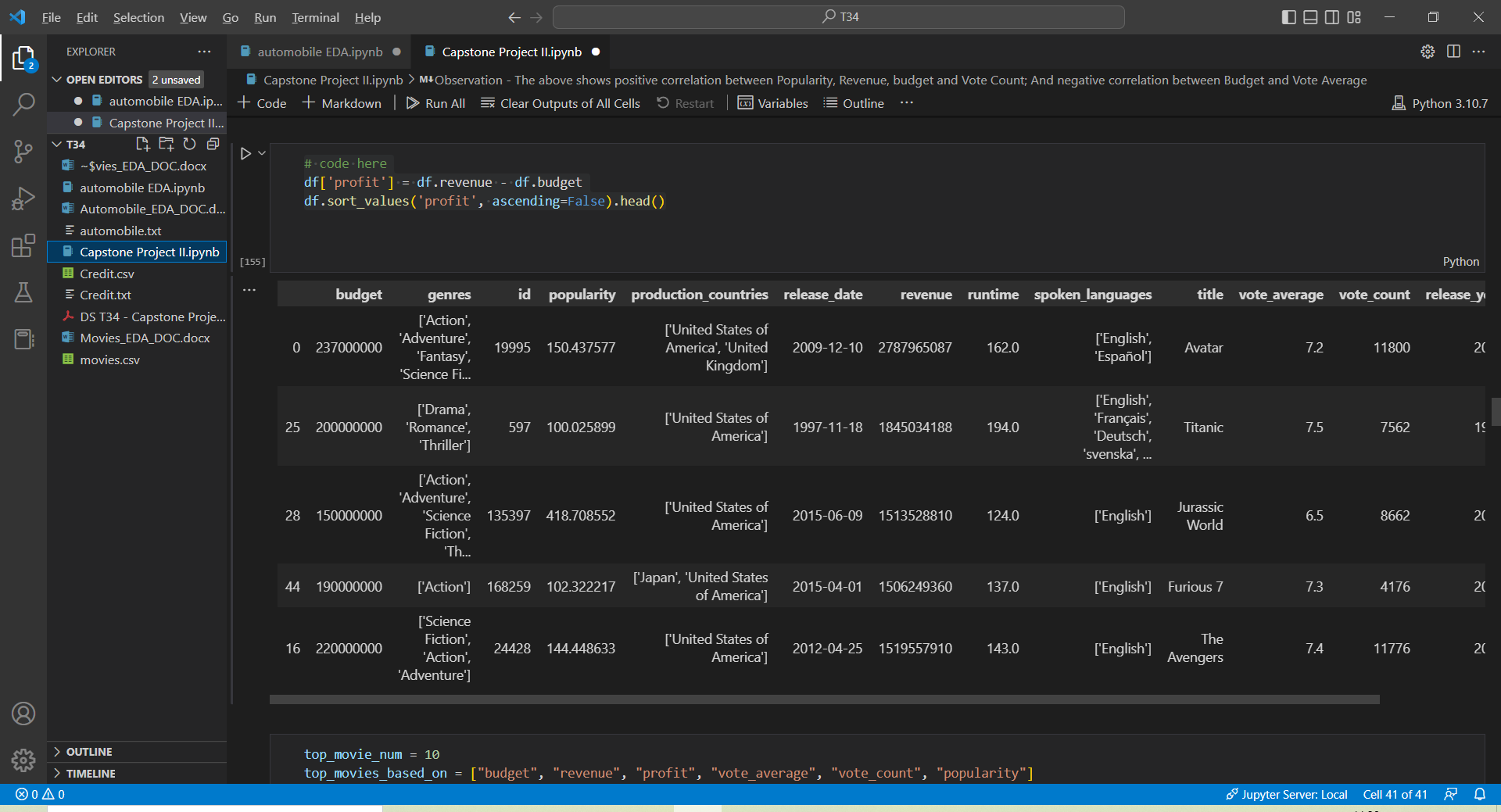
**Question** - What are the top 5 most profitable movies? Compare the min and max profits. The comparison helps us identify the different approaches which failed and succeeded. Subtracting the budget from the revenue generated, will return the profit earned

# code here

df['profit'] = df.revenue - df.budget

df.sort\_values('profit', ascending=False).head()

**Table showing the top 5 Profitable Movies:**



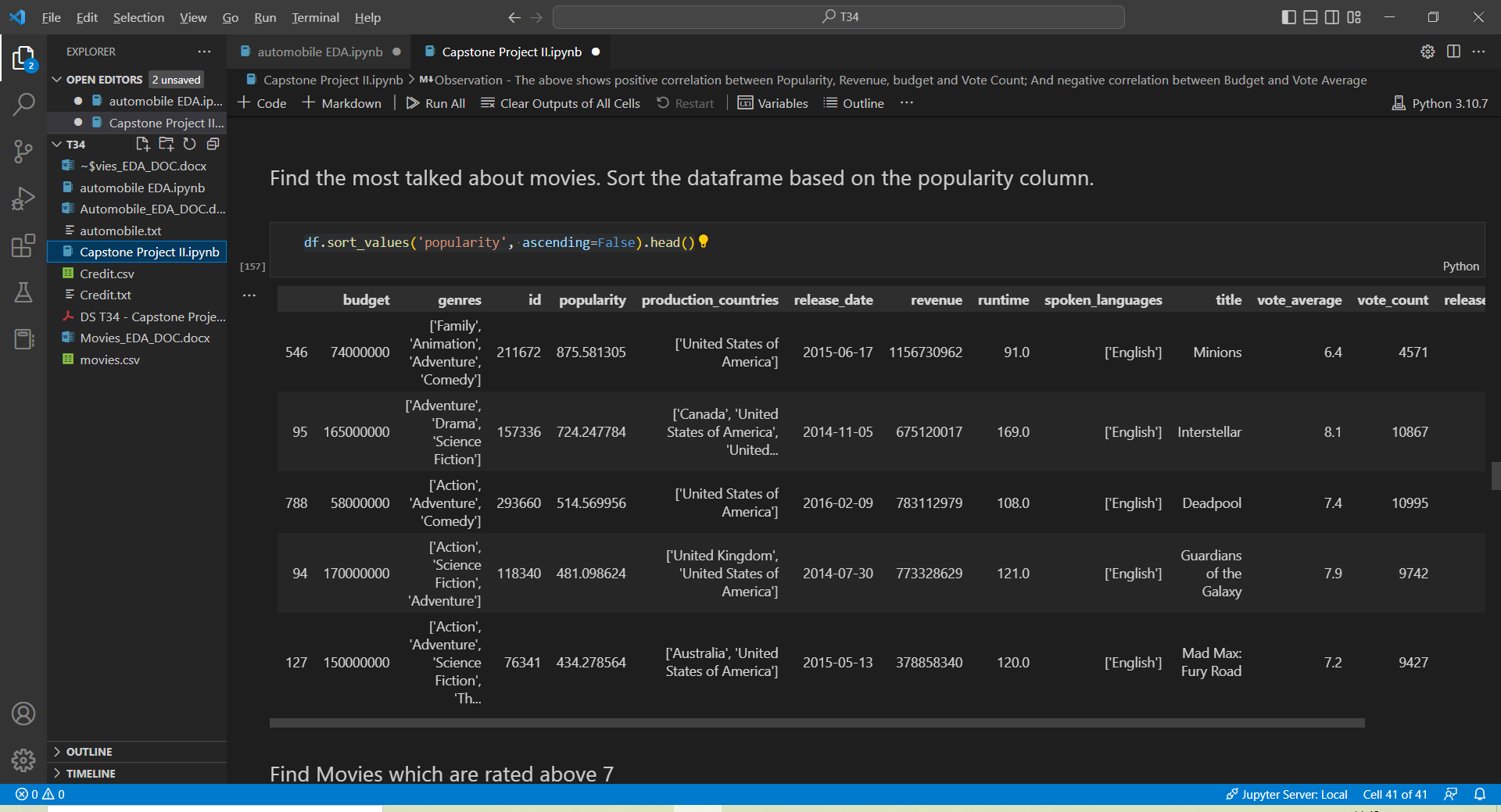
**Observations**:

1. The most profitable movies are all produced in USA and mostly from recent decades
2. Action and Adventure is most recurrent genre on profitable movies

**Question**: Find the most talked about movies. Sort the dataframe based on the popularity column.

df.sort\_values('popularity', ascending=False).head()

**Table showing the most talked about Movies:**



Observations - 1) All of the above are from last 8 years. Did we get start talking more about movies only recently - a good thought to ponder!

2) Interestingly, Canada and Australia make an appearance on these top tables for the first time

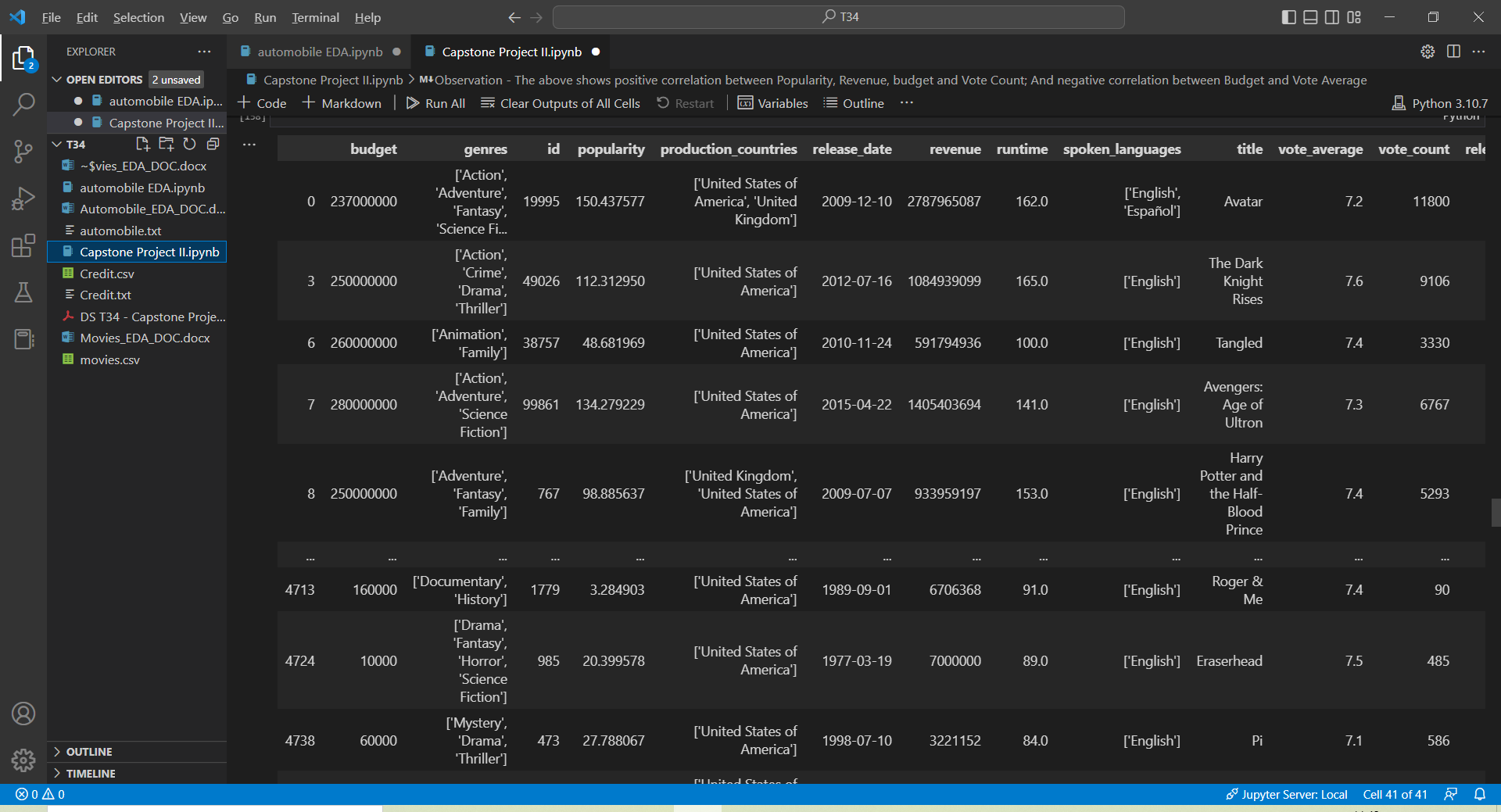
3) All English language movies in the list

**Question**: Find Movies which are rated above 7

# Code here

above\_7 = df[(df.vote\_average >7)]

**Table showing the movies rated above 7:**



Observation - 1) Movies from the 70's, 80's and 90's are appearing in our analysis on top movies for the first time

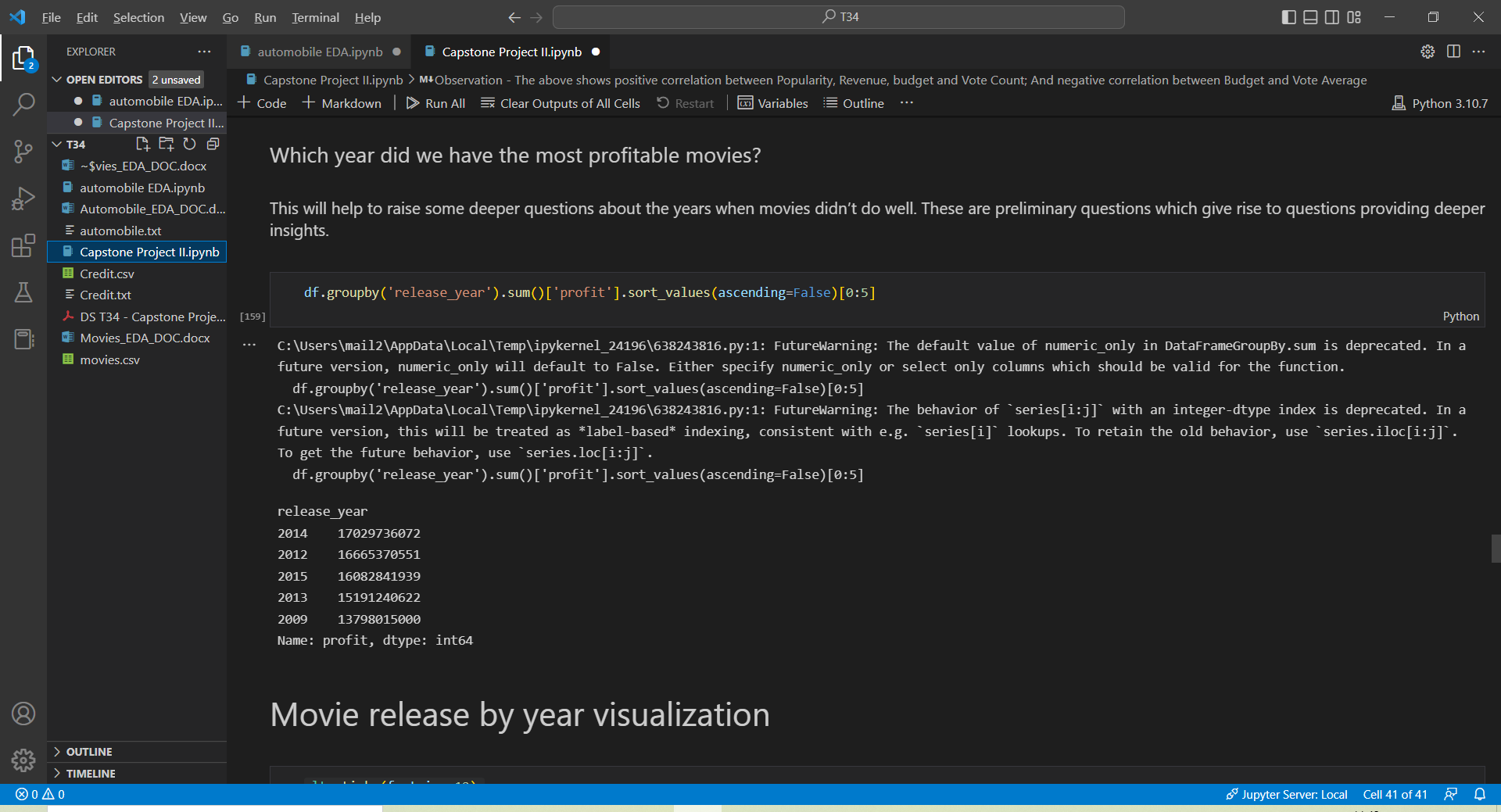
2) Non-English versions - Espanol and Japanese - appearing in the lists for the first time

3) The above two observations show that voting average can give a comprehensive world and time view of movies

**Question:** Which year did we have the most profitable movies? This will help to raise some deeper questions about the years when movies didn’t do well. These are preliminary questions which give rise to questions providing deeper insights.

df.groupby('release\_year').sum()['profit'].sort\_values(ascending=False)[0:5]

**Table showing the year with most profitable movies:**



Observation – All from recent decades in 2000’s

Creating visualizations based on profit, revenue, budget and other features:

top\_movie\_num = 10

top\_movies\_based\_on = ["budget", "revenue", "profit", "vote\_average", "vote\_count", "popularity"]

fig, ax = plt.subplots(len(top\_movies\_based\_on)//3, 3, figsize=(30,14))

colors = plt.cm.get\_cmap('viridis', top\_movie\_num)

for i, col in enumerate(top\_movies\_based\_on):

    top\_movies\_by = df.sort\_values(by=[col], ascending=False).head(top\_movie\_num)

    r, c = i//3, i%3

    ax[r][c].barh(top\_movies\_by["title"], top\_movies\_by[col], color=colors.colors)

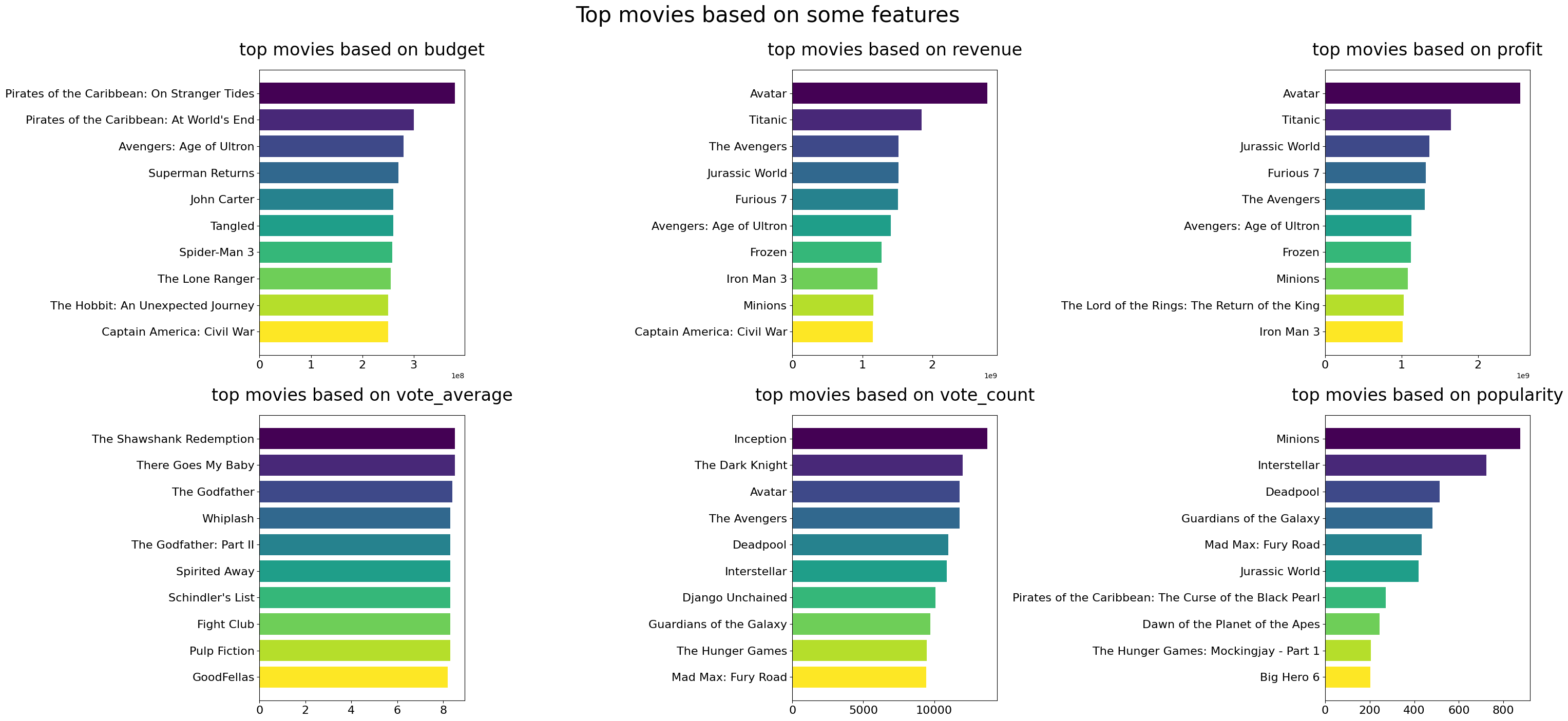
    ax[r][c].set\_title(f"top movies based on {col}", fontsize=24, pad=20)

    ax[r][c].tick\_params(axis='both', which='major', labelsize=16)

    ax[r][c].invert\_yaxis()

plt.suptitle('Top movies based on some features',fontsize=30, y=1)

fig.tight\_layout()



Observation - Avatar movie, although is most profitable and revenue generating, it is not in the list of most popular movie or has highest vote average

Most successful genres - Visualize the frequency of movies in each genre.

from collections import Counter

import squarify as sq

genres\_flatten = sum(df["genres"].values, []) # genres are list of lists

genres\_info = Counter(genres\_flatten)

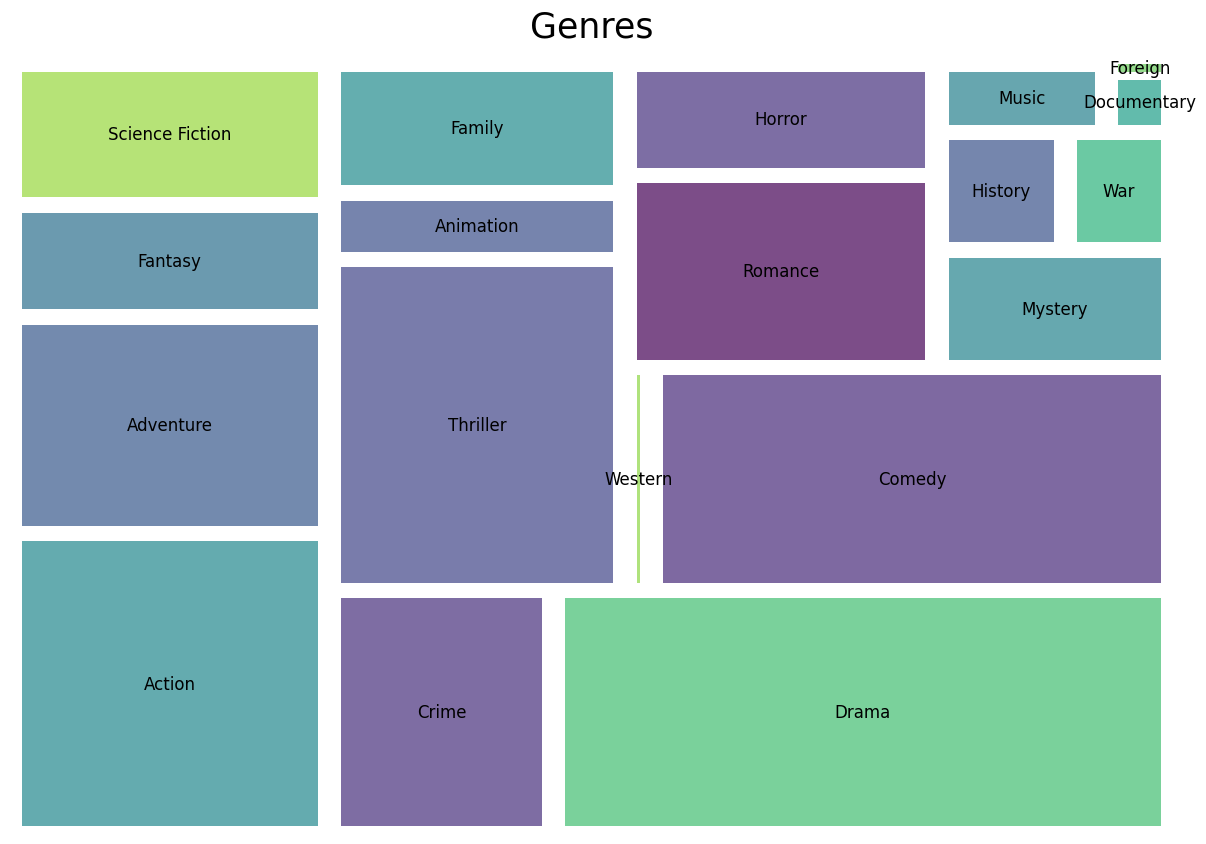
genres\_info

plt.figure(figsize=(15, 10))

plt.title("Genres", fontsize=25, pad=20)

sq.plot(genres\_info.values(), label=genres\_info.keys(), text\_kwargs={'fontsize':12}, bar\_kwargs={'alpha':.7}, pad=True)

plt.axis("off");

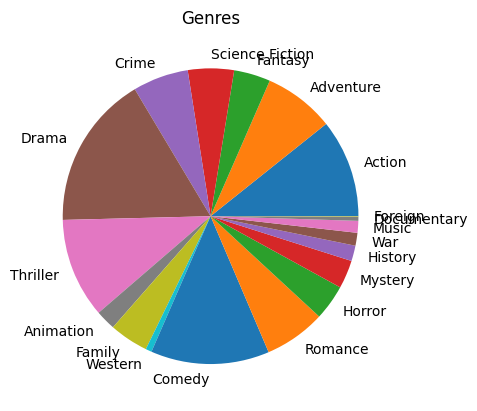


fig, ax = plt.subplots()

ax.pie(genres\_info.values(), labels = genres\_info.keys())

ax.set\_title("Genres")

plt.show()



 # Code here

genres\_df=df.explode('genres')

df["genres"] = df["genres"].apply(ast.literal\_eval)

df.genres

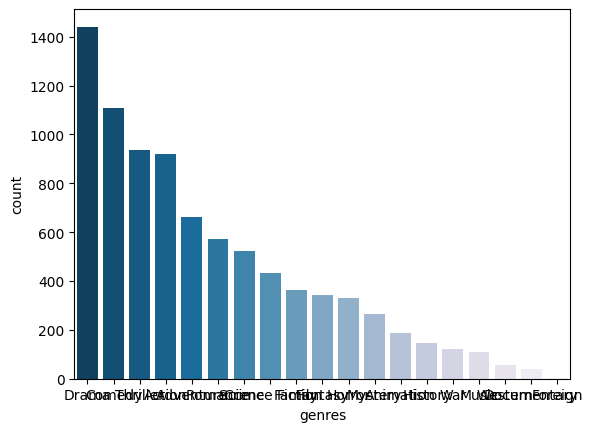
sns.countplot(

data=df,

x=df.genres.explode(), # Explode each list of genres to new row and plot

order=df.genres.explode().value\_counts().index, # Order by largest frequency of genre

palette="PuBu\_r")

****

Observation - Drama, Comedy and Action seems to be the most popular genres

Visualize movie per year:

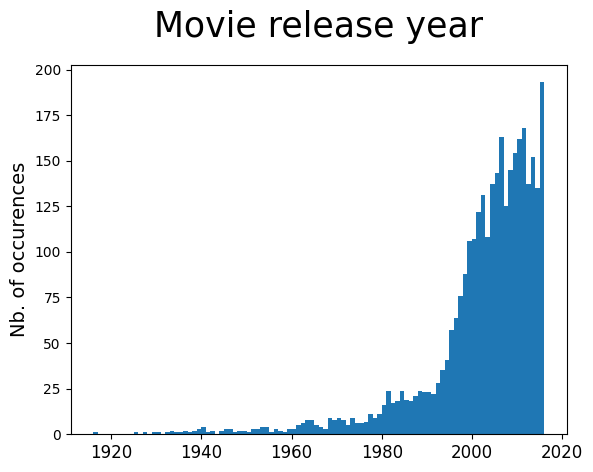
plt.xticks(fontsize=12)

plt.ylabel("Nb. of occurences", fontsize=14)

plt.title("Movie release year", fontsize=25, pad=20)

plt.hist(df["release\_year"], bins=100)

plt.figure(figsize=(15,5))

****

Observation - Production of Movies Increased from 1990’s exponentially

Visualize correlation between features

matrix = df[["budget", "popularity", "revenue", "runtime", "vote\_average", "vote\_count"]].corr()

f, ax = plt.subplots(figsize=(12, 10))

plt.title("correlation between features", fontsize=20, pad=20)

plt.xticks(range(len(matrix.index)), matrix.index, fontsize=12)

plt.yticks(range(len(matrix.index)), matrix.index, fontsize=12)

# adding values

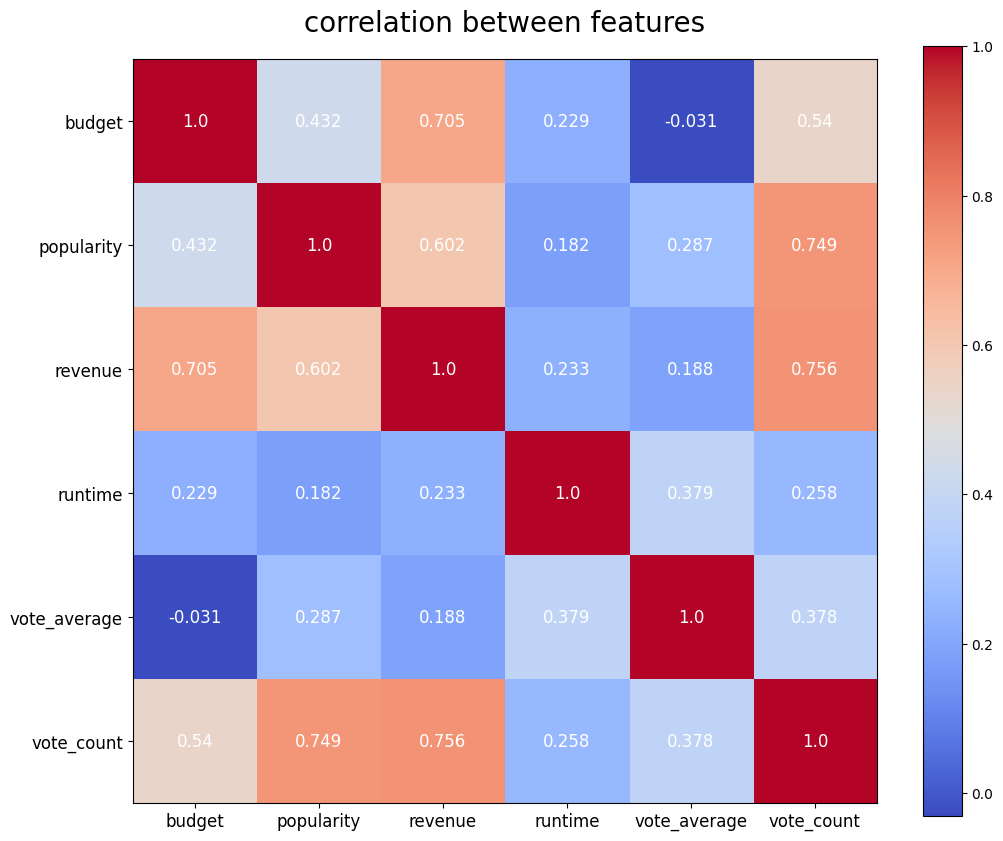
for i in range(len(matrix.index)):

    for j in range(len(matrix.index)):

        text = ax.text(j, i, round(matrix.iloc[i, j],3), ha="center", va="center", color="w", fontsize=12)

plt.imshow(matrix, cmap='coolwarm', interpolation='nearest')

plt.colorbar();



Observation - The above shows positive correlation between Popularity, Revenue, budget and Vote Count; And negative correlation between Budget and Vote Average

**THIS REPORT WAS WRITTEN BY: JENNINGS BALAVARI**

**DATE: 19TH JAN 2023**

