ONLINE STREAMING PLATFORMS

NETFLIX prime video hulu Disney+

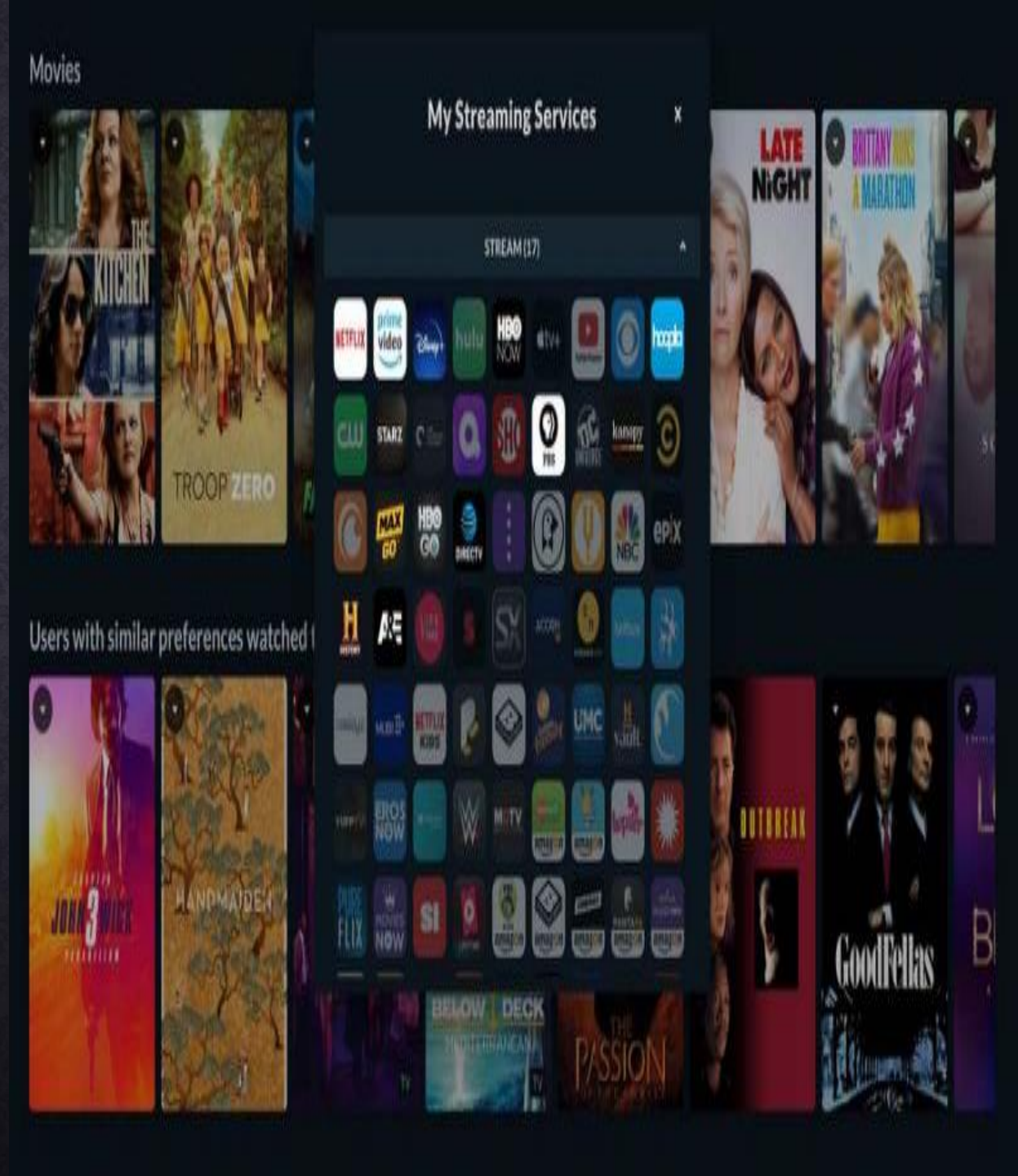
Project Review -2

Done by,

- Duddela Siva Priya
  19MIA1016
- Yuvasree R
  19MIA1053
- Madasu Deepika
  19MIA1066
- Harinisri G
  19MIA1069

Recap of the Objective

- No of Streaming shows by Age Groups
- Age Groups and No of Streaming Shows - stacked by Streaming Platforms
- Age Groups and No of Streaming Shows - faceted by Streaming Platforms
- Streaming Platforms by IMDB Ratings - Density Plot Ridges
- Streaming Platforms by Rotten Tomatoes Ratings - Density Plot Ridges
- High IMDB Rated Shows - Streaming Platforms
- Year Wise Progressing of Number of Shows by Streaming Platforms



NETFLIX hulu

VS

prime video hotstar

Problem description

As we are going to analyze and visualize, we can find :

- the best app to see the TV shows
- Most watched movies or shows
- The movies and shows a particular age group can watch.
- We can find whether the movie is present or not in a particular app.
- We can find the most used app by the customers.
- We can also find the maximum and minimum rated IMDb and rotten tomatoes.

Finding max and min IMDb ratings

```
In [6]: #finding the minimum and maximum IMDb ratings
        new_df[new_df.IMDb==new_df.IMDb.min()]
```

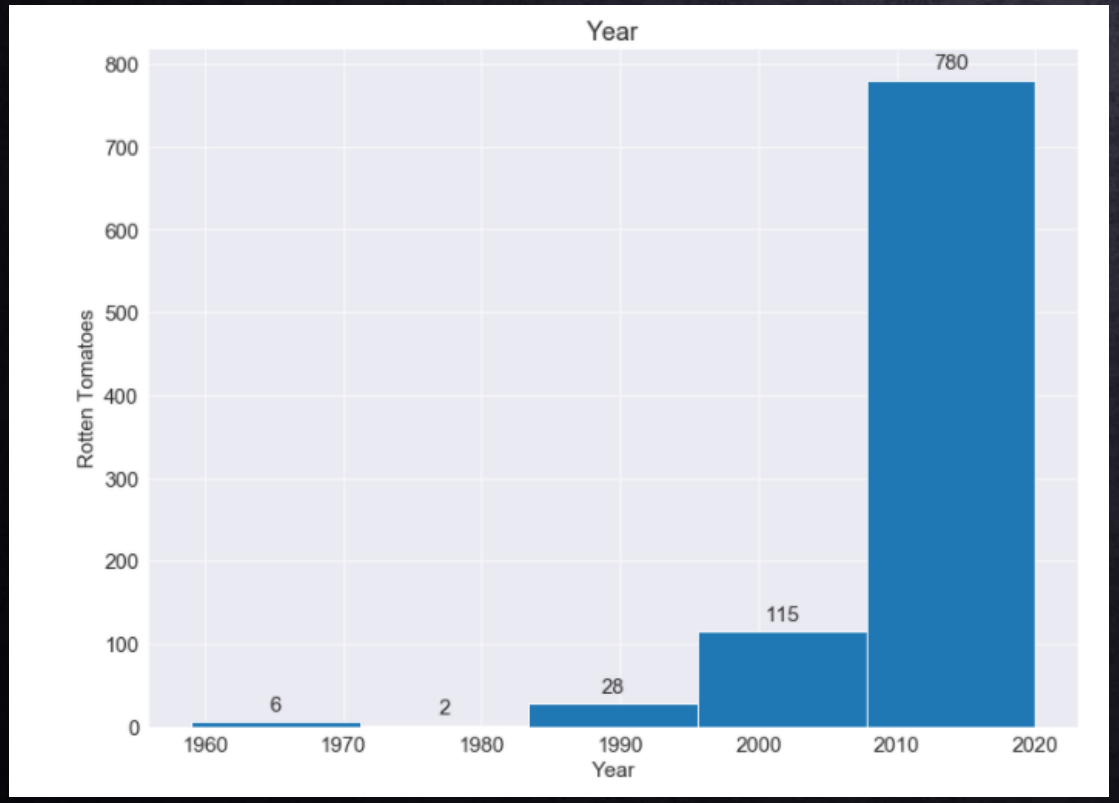
Out[6]:

	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+
732	A Little Late with Lilly Singh	2019	16+	1.7	90.0	0	1	0	0

```
In [7]: new_df[new_df.IMDb==new_df.IMDb.max()]
```

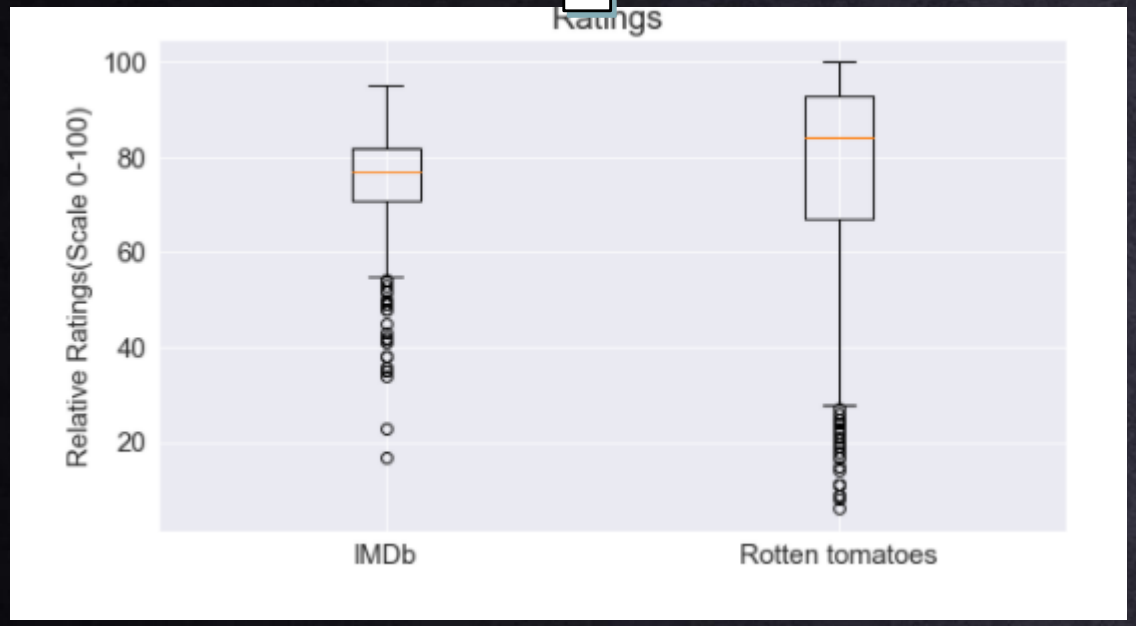
Out[7]:

	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+
0	Breaking Bad	2008	18+	9.5	96.0	1	0	0	0

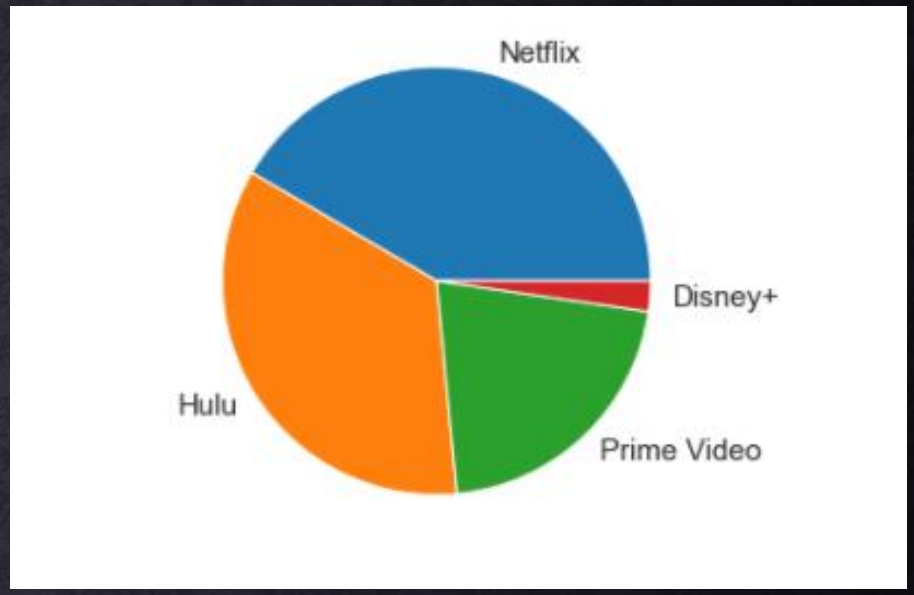


Histogram-rotten tomatoes vs year

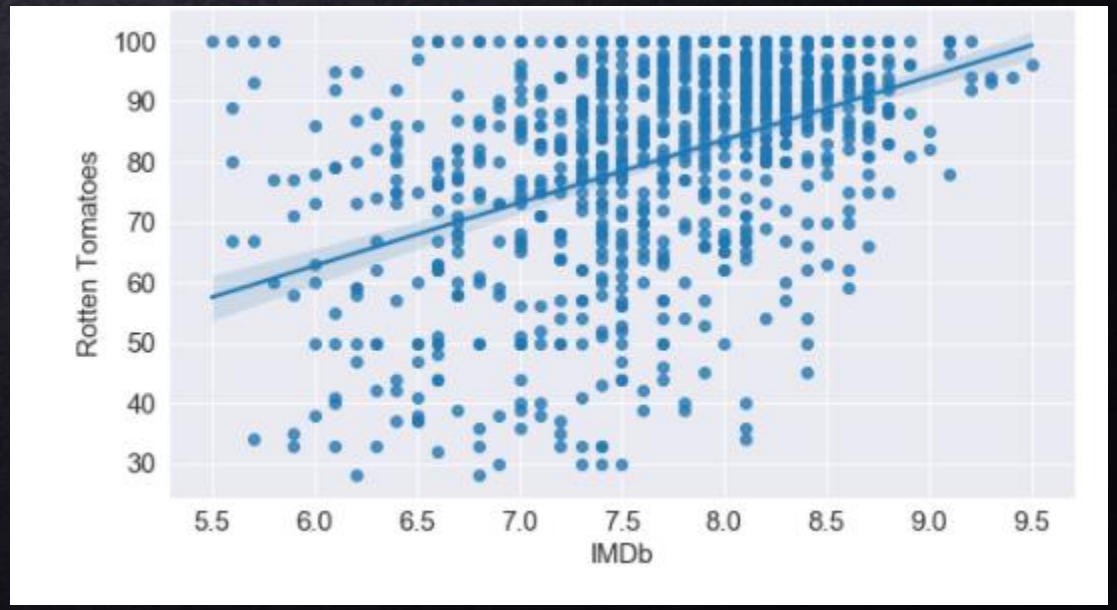
boxplot



piechart

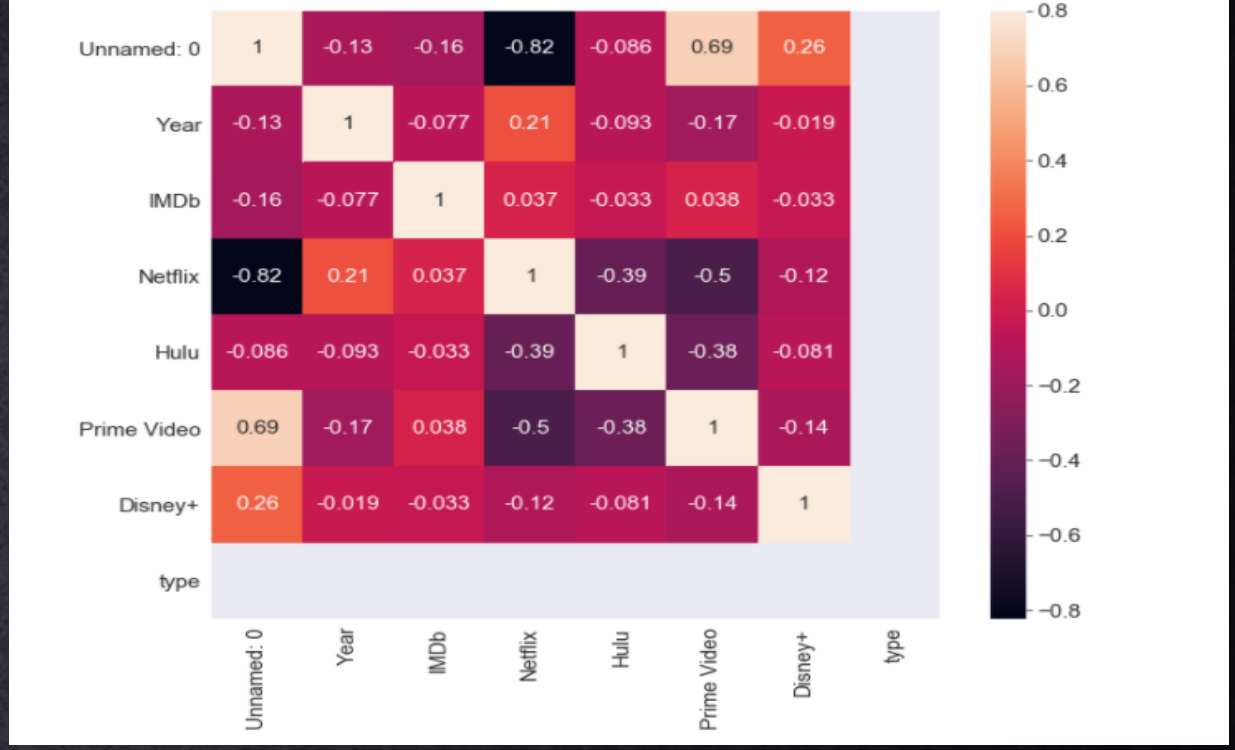


Graph after removing the outliers



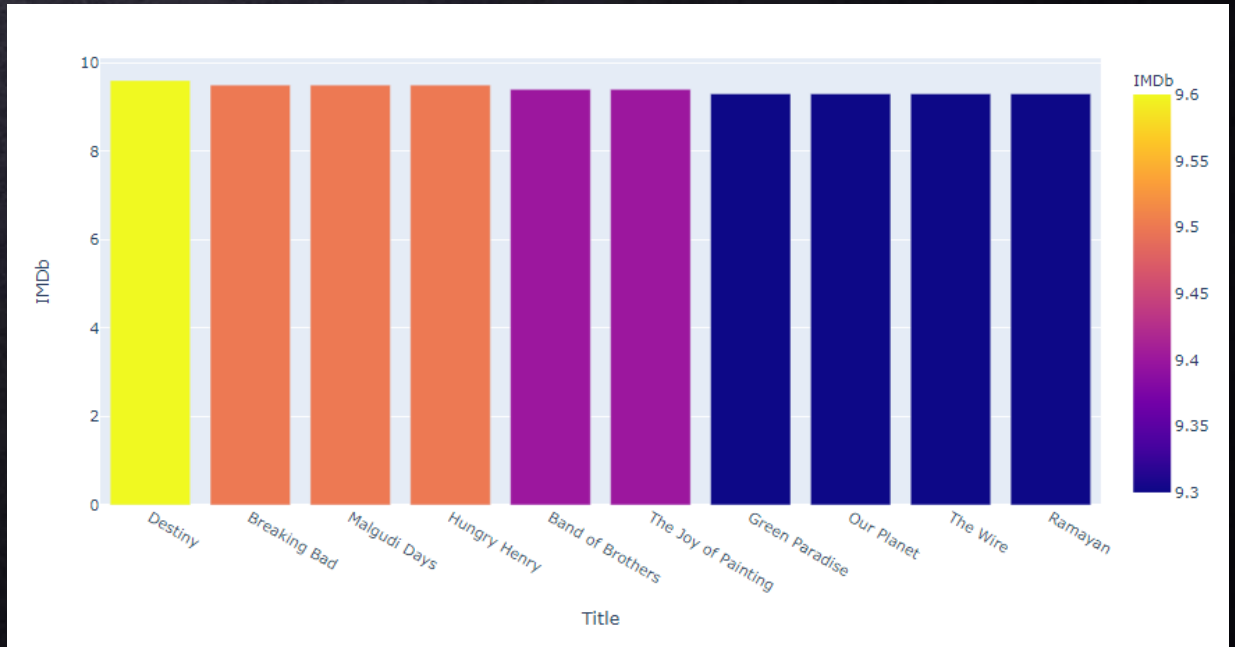
#Correlation matrix

```
import seaborn as sns
import matplotlib.pyplot as plt
corrmat = df.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True, annot = True)
plt.show()
```



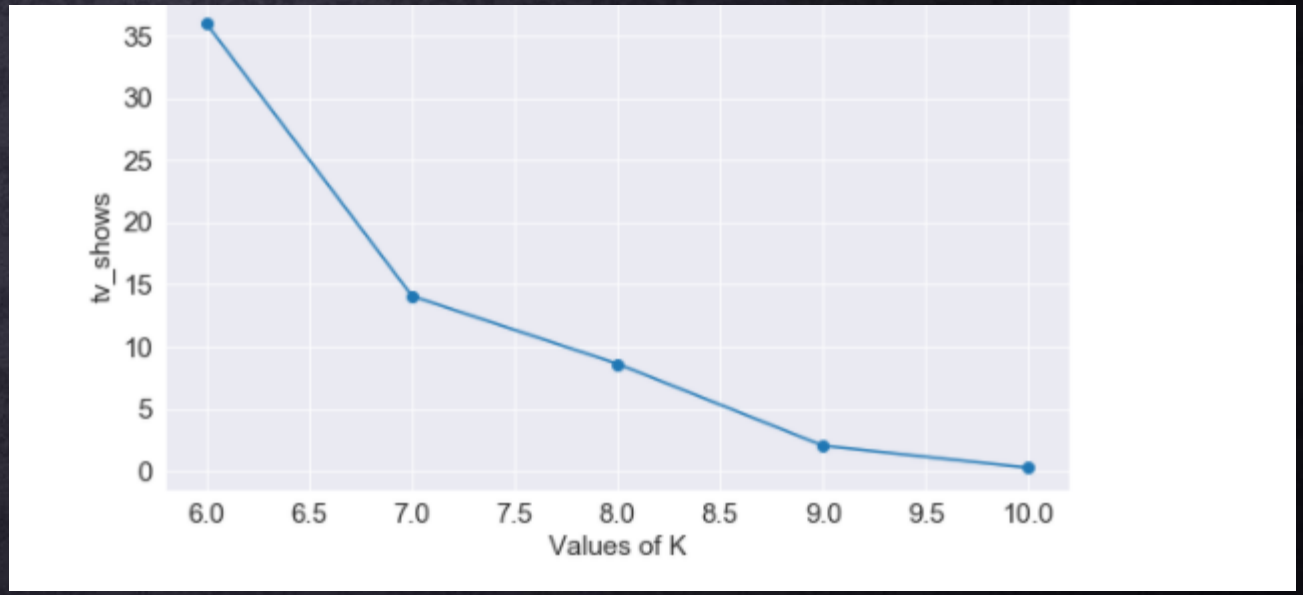
#top10IMDb

```
import plotly.express as px
fig = px.bar(top10imdb, y="IMDb", x="Title",
color='IMDb')
fig.show()
```



K-means clustering

```
import matplotlib.pyplot as plt

# Amount of values to be tested for K
Ks = range(6, 11)

# List to hold on the metrics for each value of K
results = []

# Executing the loop
for K in Ks:

    model = KMeans(n_clusters = K)
    model.fit(DF_NORM)

    results.append(model.inertia_)

# Plotting the final result
plt.plot(Ks, results, 'o-')
plt.xlabel("Values of K")
plt.ylabel("tv_shows")
plt.show()
```

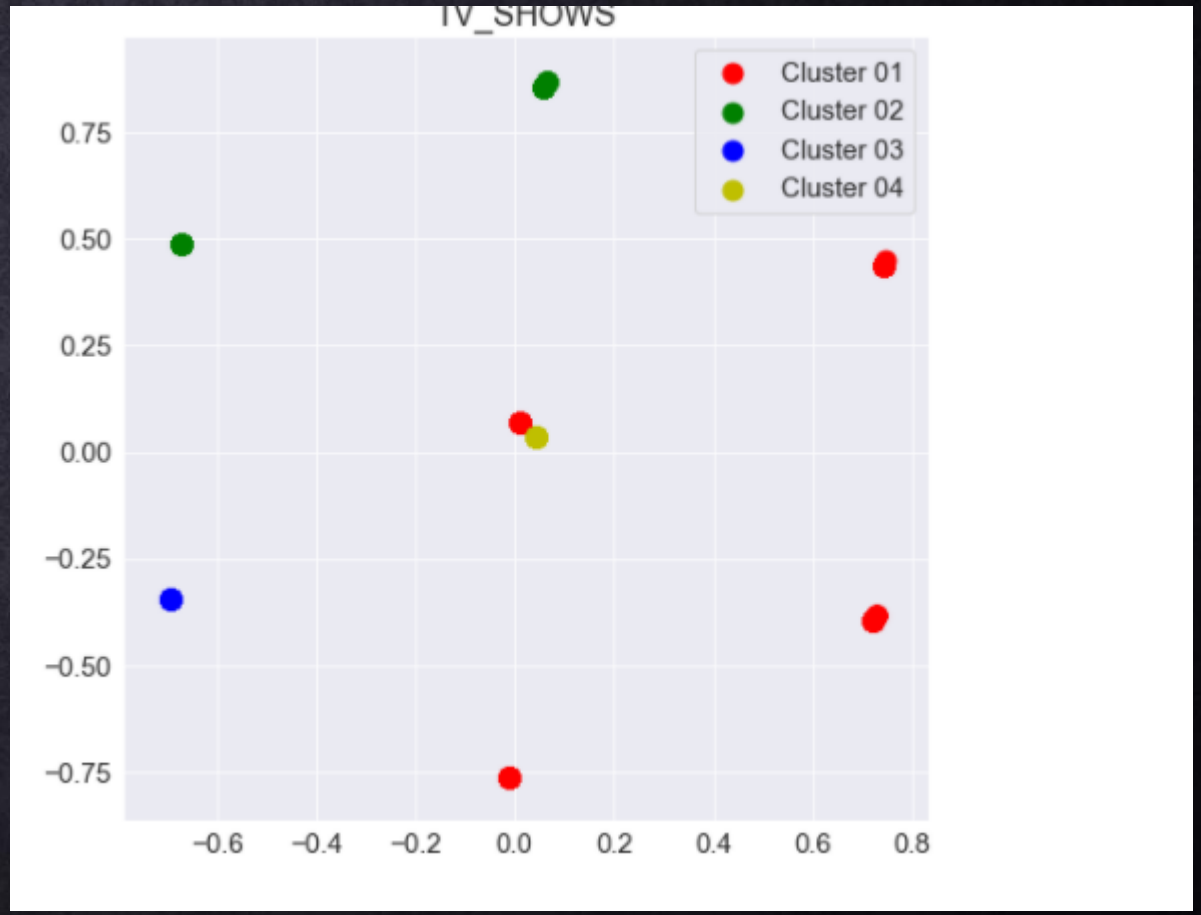


In [43]:
```
from sklearn.cluster import KMeans

# Creating our Model
kmeans = KMeans(n_clusters = 4)

# Training our model
kmeans.fit(DF_NORM)

# You can see the labels (clusters) assigned for each data point with the function labels_
kmeans.labels_

# Assigning the labels to the initial dataset
DF['cluster'] = kmeans.labels_
```

In [44]:
```
from matplotlib import pylab
from pylab import *
```

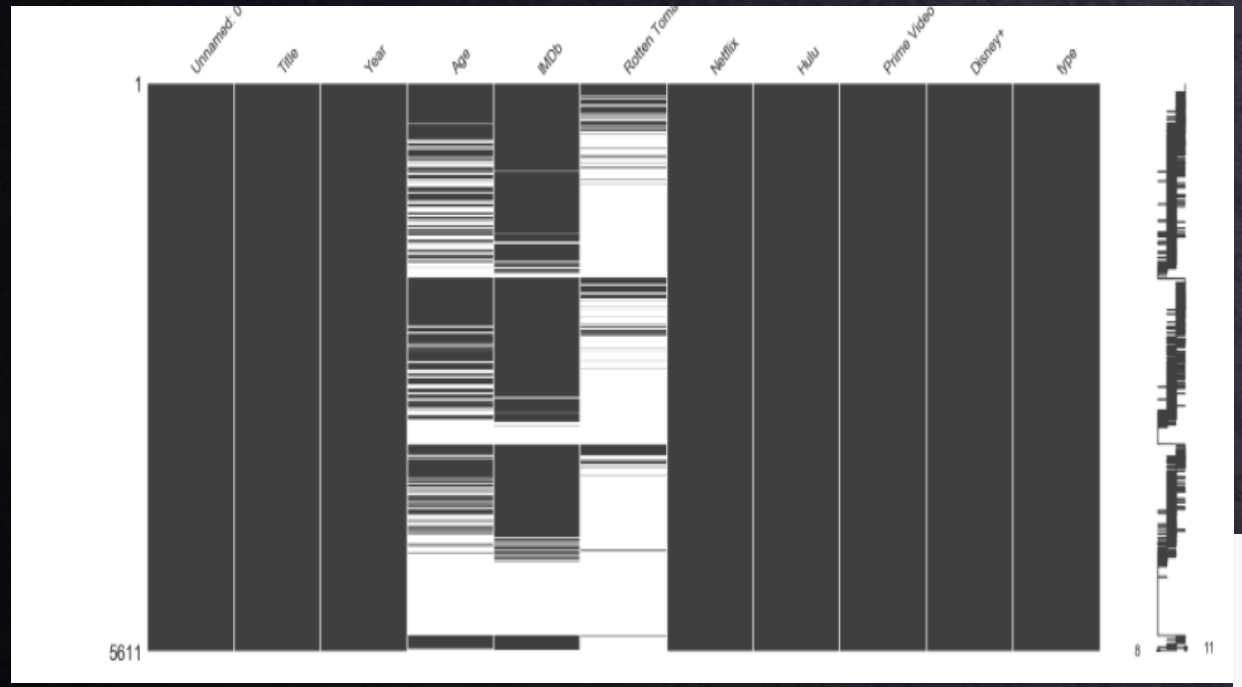
In [45]:
```
from sklearn.decomposition import PCA
import pylab as pl

# Reducing data dimensions
PCA_ = PCA(n_components = 2).fit(DF_ARRAY)

# Applying the PCA
PCA_2 = PCA_.transform(DF_ARRAY)

#--------------------------
# Plotting the cluster
#--------------------------

# Plot size
pylab.rcParams['figure.figsize'] = (8.0, 8.0)

# Plotting each point individually depending on their cluster
for i in range(0, PCA_2.shape[0]):
```

```
PCA_2 = PCA_.transform(DF_ARRAY)

#--------------------------
# Plotting the cluster
#--------------------------

# Plot size
pylab.rcParams['figure.figsize'] = (8.0, 8.0)

# Plotting each point individually depending on their cluster
for i in range(0, PCA_2.shape[0]):

    # If the 'i' data point be in cluster 0, it will be plotted as the formatting inside the if functions
    # And so on...
    if kmeans.labels_[i] == 0:
        CLUSTER_01 = pl.scatter(PCA_2[i,0], PCA_2[i,1], c ='r', marker = 'o', s = 120)

    elif kmeans.labels_[i] == 1:
        CLUSTER_02 = pl.scatter(PCA_2[i,0], PCA_2[i,1], c ='g', marker = 'o', s = 120)

    elif kmeans.labels_[i] == 2:
        CLUSTER_03 = pl.scatter(PCA_2[i,0], PCA_2[i,1], c ='b', marker = 'o', s = 120)

    elif kmeans.labels_[i] == 3:
        CLUSTER_04 = pl.scatter(PCA_2[i,0], PCA_2[i,1], c ='y', marker = 'o', s = 120)


        # Formatting the Plot
        pl.legend([CLUSTER_01, CLUSTER_02, CLUSTER_03, CLUSTER_04],
                  ['Cluster 01', 'Cluster 02', 'Cluster 03', 'Cluster 04'])

        pl.title('TV_SHOWS')

pl.show()
```

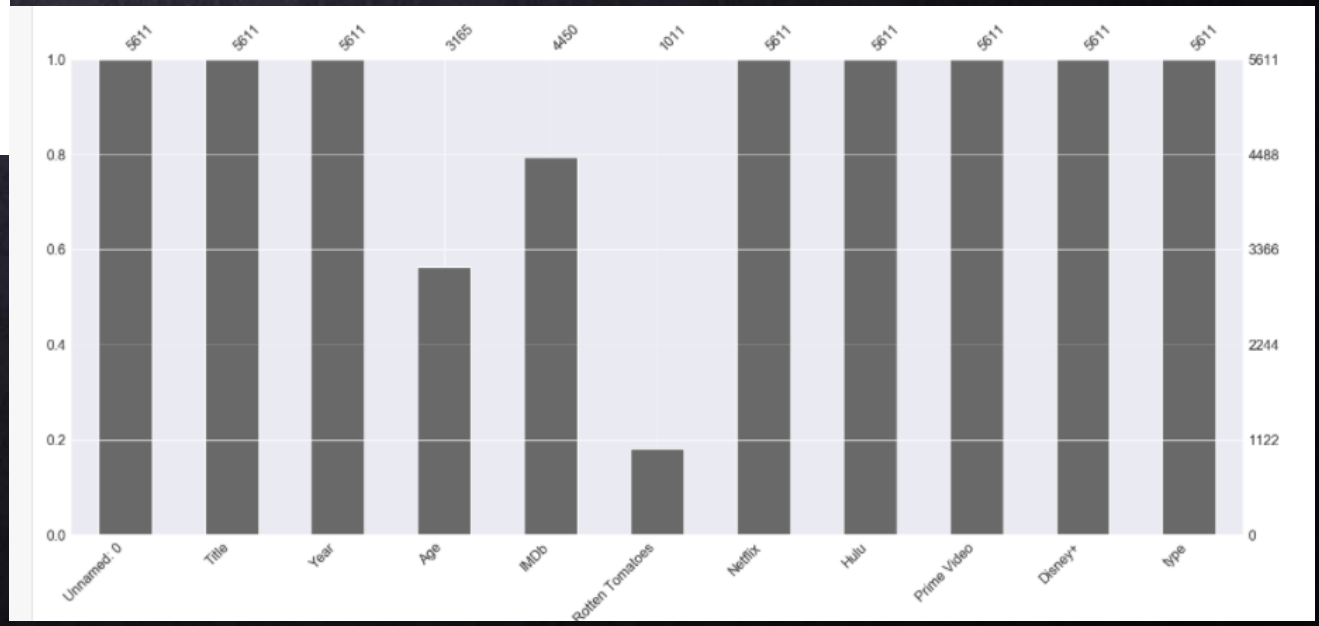


Visualizing the missing data

Matrix of missing data

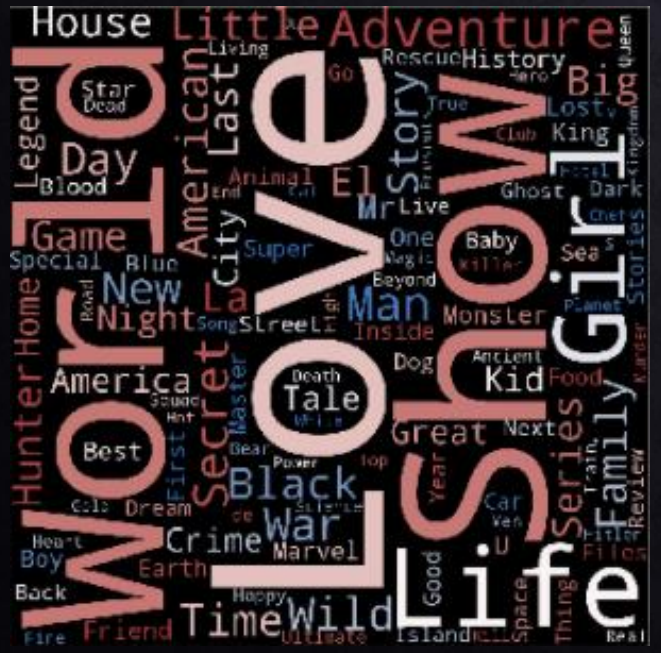


Bar graph of missing data



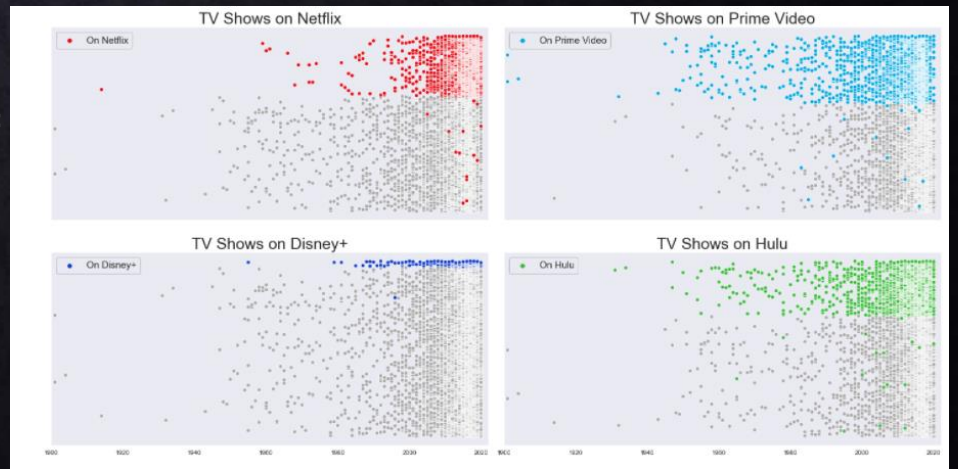
Word cloud

```
#wordcloud
from wordcloud import WordCloud, STOPWORDS

text = ' '.join(df['Title'])

plt.rcParams['figure.figsize'] = (6,12)
wordcloud = WordCloud(background_color =
'black',colormap='vlag', width = 1200,  height = 1200,
max_words = 121).generate(text)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



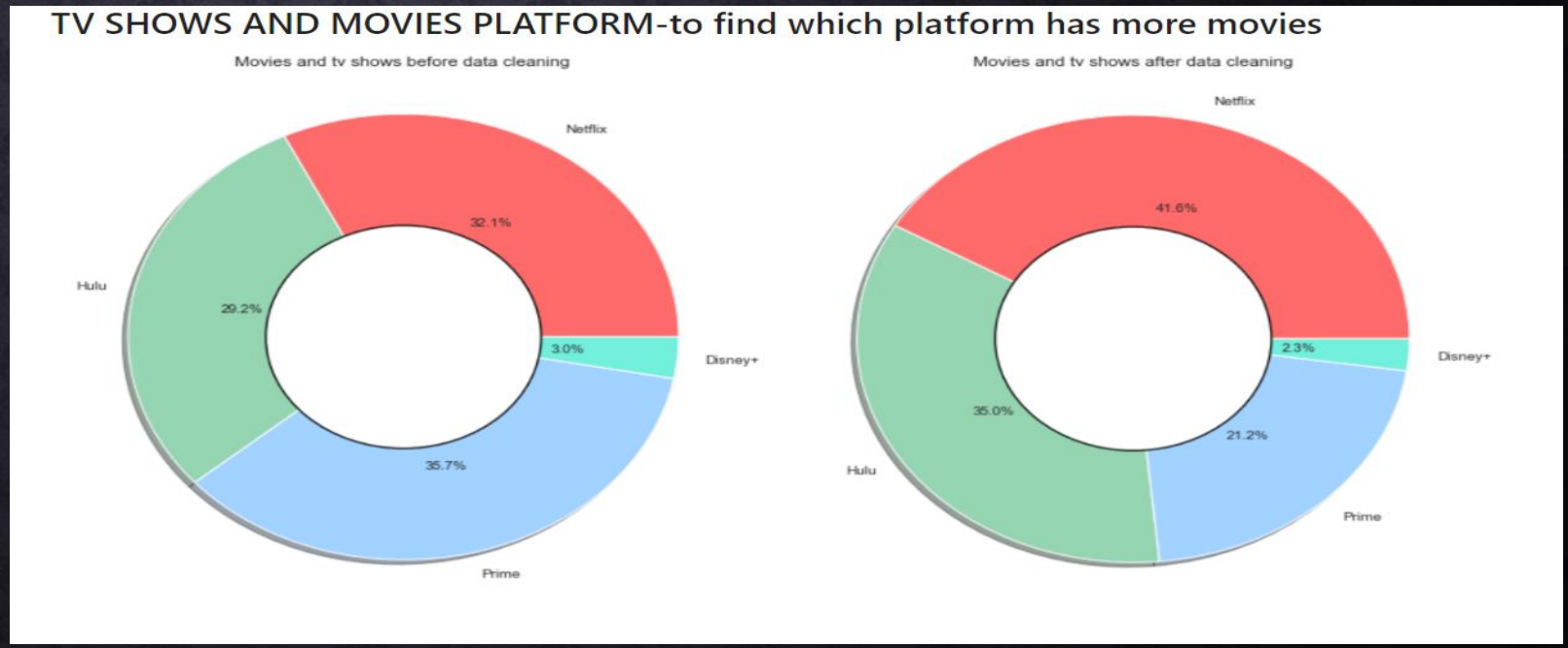
Scatter plot

```
fig = plt.figure(figsize=[20,10])
sns.set_style('dark')

for col_name, num in zip(col_names_online_media,
                         range(1,len(col_names_online_media) + 1)):
    ax = fig.add_subplot(2, 2, num)
    sns.scatterplot(x="Year", y="Title",
                    palette = pallete_dict[col_name],
                    hue=col_name,
                    data=df.sort_values(by=[col_name, 'Title']),
                    ax=ax)
    sns.despine
    ax.set_title('TV Shows on ' + col_name, fontsize=25)
    handles, labels = ax.get_legend_handles_labels()
    ax.legend(loc='upper left',
              frameon=None,
              edgecolor='black',
              fontsize=15,
              framealpha=0.2,
              handles=[handles[2]],
              labels=['On ' + col_name])
    ax.set_xlim(1900, 2022)
    ax.set(yticklabels=[])
    ax.set(ylabel=None, xlabel=None)

fig.tight_layout()

for ax in fig.get_axes():
    ax.label_outer()

plt.show()
```

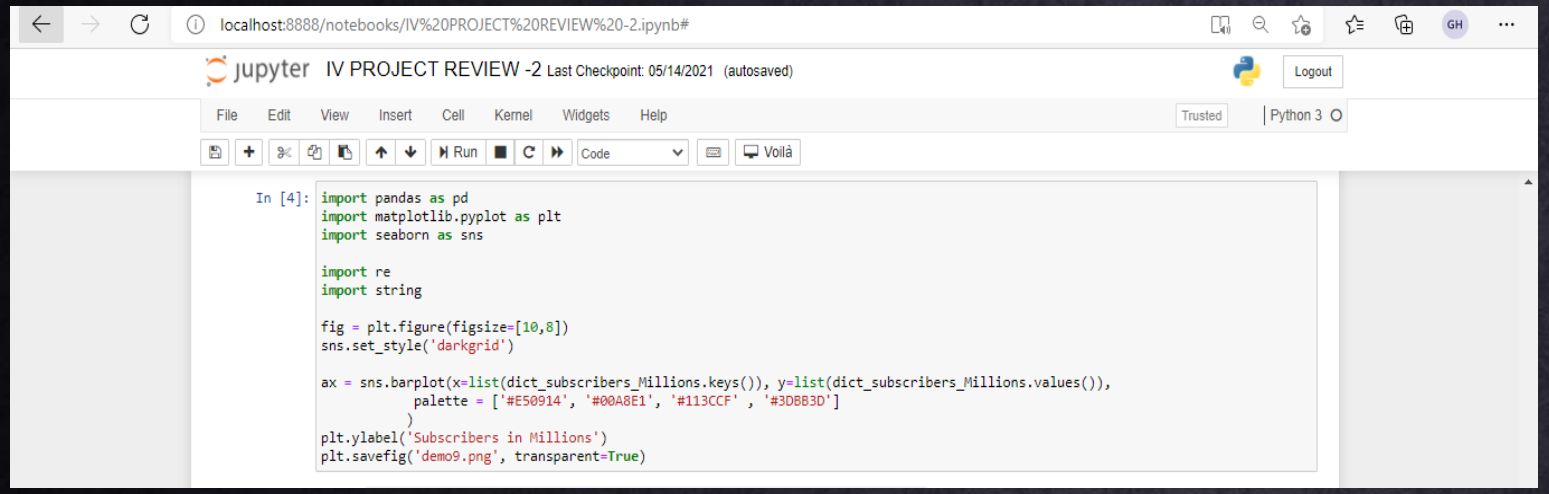


```
def donut(i,df,sizes,title):
    plt.subplot(i)
    plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
            autopct='%1.1f%%', shadow=True)

    centre_circle = plt.Circle((0,0),0.5,color='black',
fc='white',linewidth=1.25)
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)
    plt.title(title)
    plt.axis('equal')
fig = plt.subplots(figsize=(16, 8))
labels = 'Netflix', 'Hulu','Prime','Disney+'
sizes1 = [val_counts[0],
val_counts[1],val_counts[2],val_counts[3]]
sizes2 = [val_counts[4],
val_counts[5],val_counts[6],val_counts[7]]
custom_colors =
["#ff6b6b","#95d5b2","#a2d2ff","#72efdd"]
colors = custom_colors
explode = (0, 0, 0, 0)

donut(121,tvshows_df,sizes1,'Movies and tv shows before
data cleaning')
donut(122,new_df,sizes2,'Movies and tv shows after data
cleaning')
plt.show()
```

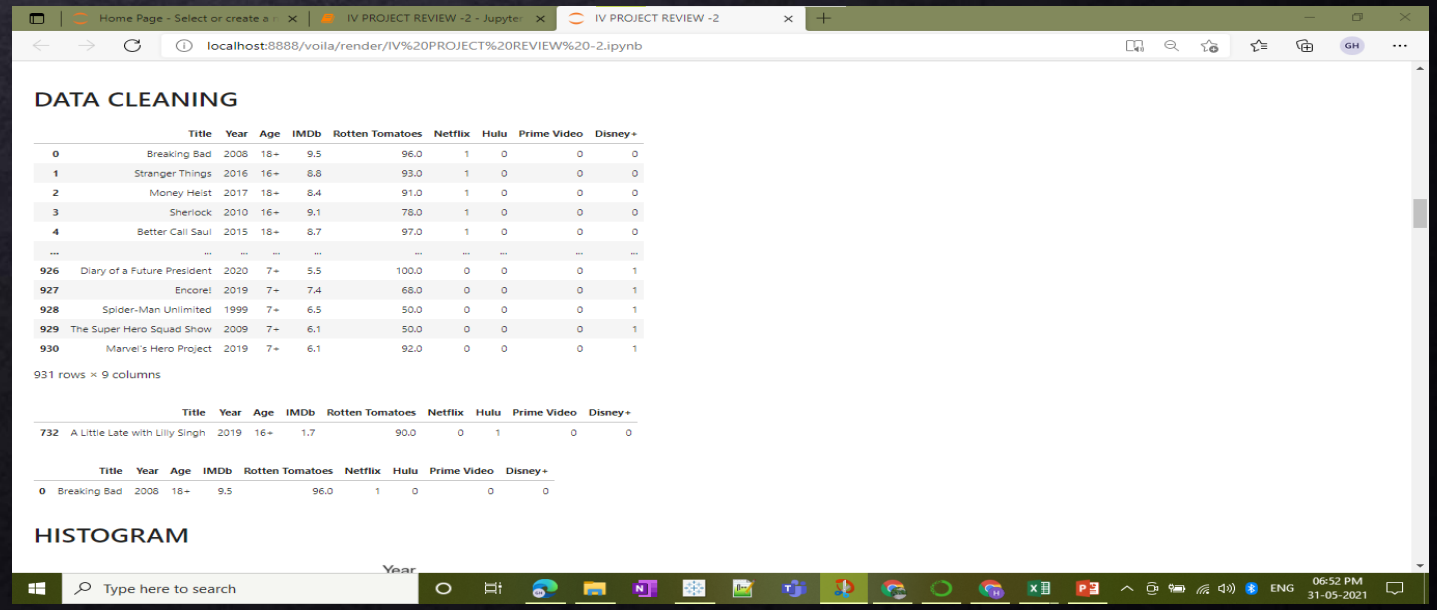
Donut piechart

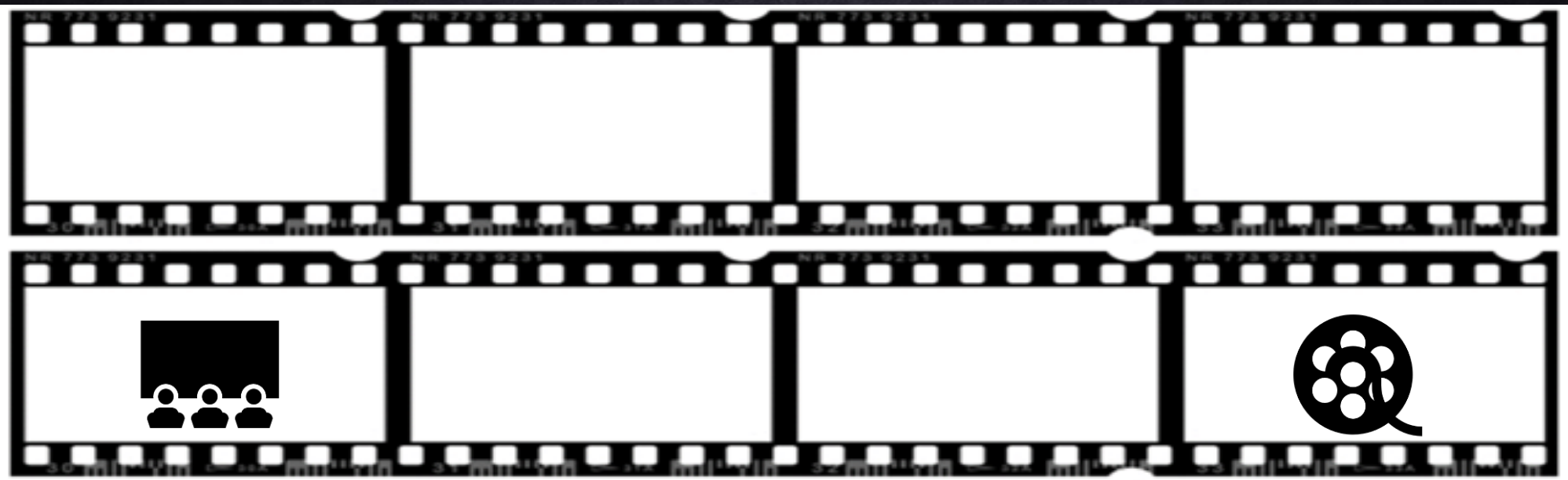


Voila package



This package turns the whole notebook
To web application , and supports to create Interactive dashboards,reports .
It runs all code one by one.





Thank You