Hollywood movies data analysis and visualization

Import libraries

In [ ]:
```
import numpy as np
import pandas as pd
import io
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import plotly.express as px
%matplotlib inline
import seaborn as sns
import pandas_profiling
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelBinarizer
import plotly.graph_objects as go
import plotly.io as pio
pio.templates
```

Load data

In [2]:
```
df = pd.read_csv('../input/hollywood/HollywoodsMostProfitableStories.csv')
print(df)
```

```
                            Film    Genre            Lead Studio  \
0                       27 Dresses   Comedy                    Fox
1                (500) Days of Summer   Comedy                    Fox
2                  A Dangerous Method    Drama             Independent
3                       A Serious Man    Drama               Universal
4                  Across the Universe  Romance             Independent
..                                 ...     ...                    ...
69               What Happens in Vegas   Comedy                    Fox
70                       When in Rome   Comedy                 Disney
71  You Will Meet a Tall Dark Stranger   Comedy             Independent
72                   Youth in Revolt   Comedy  The Weinstein Company
73         Zack and Miri Make a Porno  Romance  The Weinstein Company

    Audience  score %  Profitability  Rotten Tomatoes %  Worldwide Gross  Year
0               71.0       5.343622               40.0       160.308654  2008
1               81.0       8.096000               87.0        60.720000  2009
2               89.0       0.448645               79.0         8.972895  2011
3               64.0       4.382857               89.0        30.680000  2009
4               84.0       0.652603               54.0        29.367143  2007
..               ...            ...                ...              ...   ...
69              72.0       6.267647               28.0       219.367646  2008
70              44.0            NaN               15.0        43.040000  2010
71              35.0       1.211818               43.0        26.660000  2010
72              52.0       1.090000               68.0        19.620000  2010
73              70.0       1.747542               64.0        41.941000  2008

[74 rows x 8 columns]
```

In [3]:
```
report = pandas_profiling.ProfileReport(df)
from IPython.display import display
display(report)
```

```
Summarize dataset:   0%|          | 0/22 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	8
Number of observations	74
Missing cells	6
Missing cells (%)	1.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	4.8 KiB
Average record size in memory	65.7 B

Variable types

NUM	5
CAT	3

Reproduction

Analysis started	2023-10-28 23:00:41.382090
Analysis finished	2023-10-28 23:00:51.677674
Duration	10.3 seconds
Version	pandas-profiling v2.8.0 (https://github.com/pandas-profiling/pandas-profiling)
Command line	`pandas_profiling --config_file config.yaml [YOUR_FILE.csv]`

In [4]:
```
print("There are {} films and {} columns in this dataset. \n".format(df.shape[0],df.shape[1]))

print("There are {} different years in this dataset {} \n".format(len(df.Year.unique()),
                                                                  ", ".join(df.Film.unique()[0:0])))

print("There are {} different genres in this dataset such as {} \n".format(len(df.Genre.unique()),
                                                                           ", ".join(df.Genre.unique()[0:6])))
```

There are 74 films and 8 columns in this dataset.

There are 5 different years in this dataset

There are 6 different genres in this dataset such as Comedy, Drama, Romance, Animation, Action, Fantasy

In [5]:
```
pd.read_csv('../input/hollywood/HollywoodsMostProfitableStories.csv')
```

Out[5]:

	Film	Genre	Lead Studio	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
0	27 Dresses	Comedy	Fox	71.0	5.343622	40.0	160.308654	2008
1	(500) Days of Summer	Comedy	Fox	81.0	8.096000	87.0	60.720000	2009
2	A Dangerous Method	Drama	Independent	89.0	0.448645	79.0	8.972895	2011
3	A Serious Man	Drama	Universal	64.0	4.382857	89.0	30.680000	2009
4	Across the Universe	Romance	Independent	84.0	0.652603	54.0	29.367143	2007
...	...	...	...	...	...	...	...	...
69	What Happens in Vegas	Comedy	Fox	72.0	6.267647	28.0	219.367646	2008
70	When in Rome	Comedy	Disney	44.0	NaN	15.0	43.040000	2010
71	You Will Meet a Tall Dark Stranger	Comedy	Independent	35.0	1.211818	43.0	26.660000	2010
72	Youth in Revolt	Comedy	The Weinstein Company	52.0	1.090000	68.0	19.620000	2010
73	Zack and Miri Make a Porno	Romance	The Weinstein Company	70.0	1.747542	64.0	41.941000	2008

74 rows × 8 columns

In [6]:
```
df.describe()
```

Out[6]:

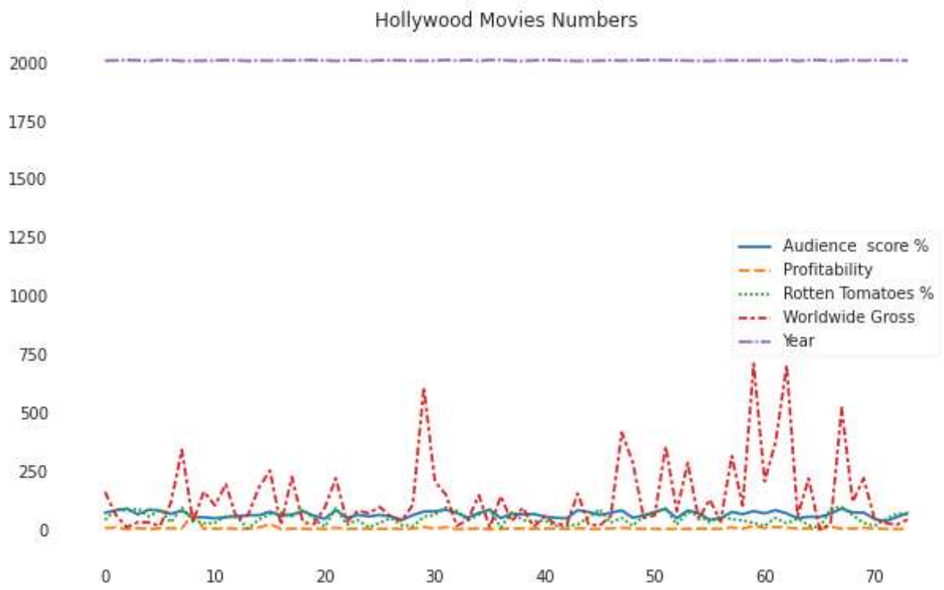
	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
count	73.000000	71.000000	73.000000	74.000000	74.000000
mean	64.136986	4.741610	47.356164	136.351979	2009.054054
std	13.647665	8.292017	26.242655	157.067561	1.353756
min	35.000000	0.005000	3.000000	0.025000	2007.000000
25%	52.000000	1.790680	27.000000	32.447500	2008.000000
50%	64.000000	2.642353	45.000000	73.198612	2009.000000
75%	76.000000	4.850958	65.000000	190.185250	2010.000000
max	89.000000	66.934000	96.000000	709.820000	2011.000000

Plot set of data

- Line plots

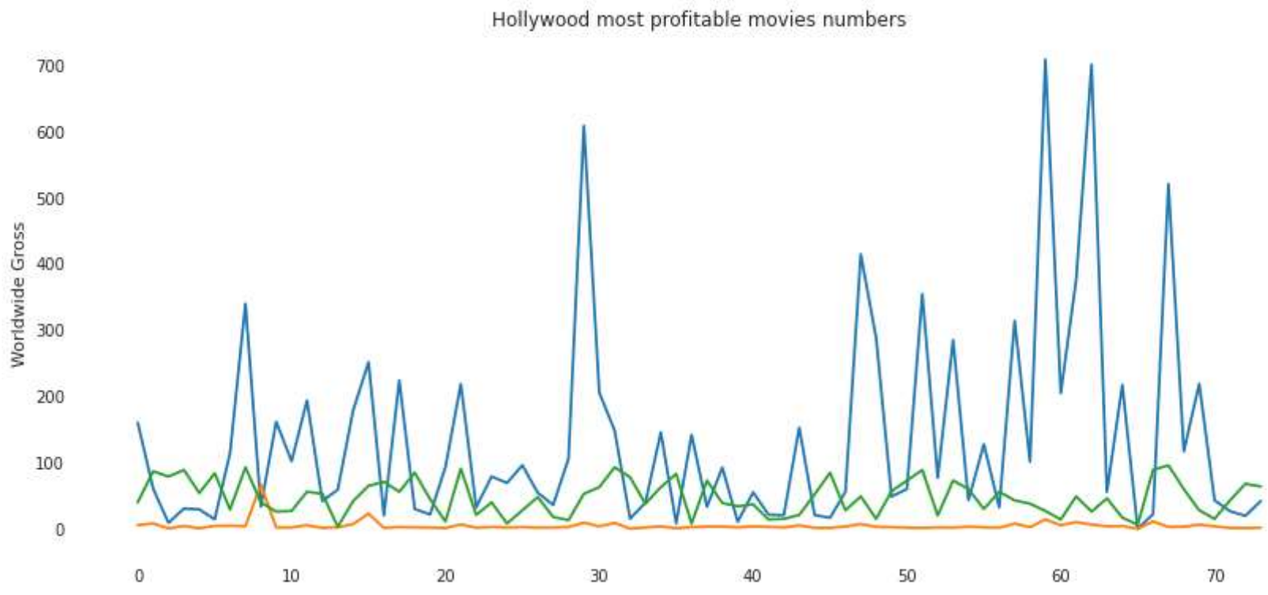
In [7]:
```
# Set the width and height of the figure
plt.figure(figsize=(10, 6))

# Add title
plt.title("Hollywood Movies Numbers")

# Line chart showing daily global streams of each song
sns.lineplot(data=df)
```

Out[7]:
```
<AxesSubplot:title={'center':'Hollywood Movies Numbers'}>
```



In [8]:
```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Line chart showing Hollywood worldwide gross profits
sns.lineplot(data=df['Worldwide Gross'])

# Line chart showing Hollywood movies audience scores%
#sns.lineplot(data=df['Audience score %'])

# Line chart showing Hollywood movies profitability
sns.lineplot(data=df['Profitability'])

# Line chart showing Hollywood movies rotten tomatoe rating
sns.lineplot(data=df['Rotten Tomatoes %'])

# Add title
plt.title("Hollywood most profitable movies numbers")
```

Out[8]:
```
Text(0.5, 1.0, 'Hollywood most profitable movies numbers')
```



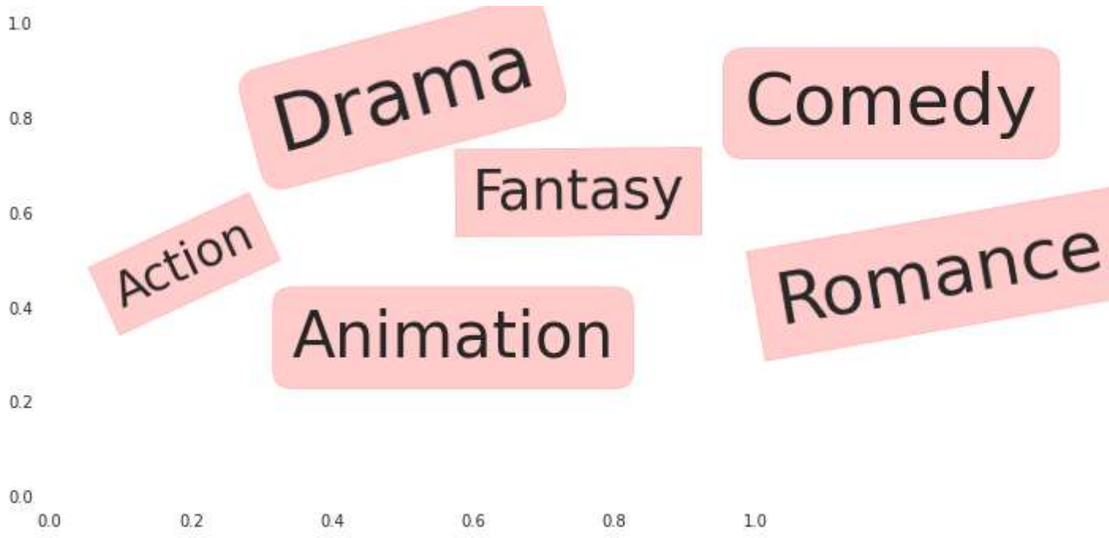
Movie Genres

This graphs which genre makes the most money compare to others. Based on the graph, it seems like drama makes the most money then romance. With comedy and animation tied. It seems like fantasy and action makes the least.

- Style Text Boxes
- Bar Charts
- Word Cloud

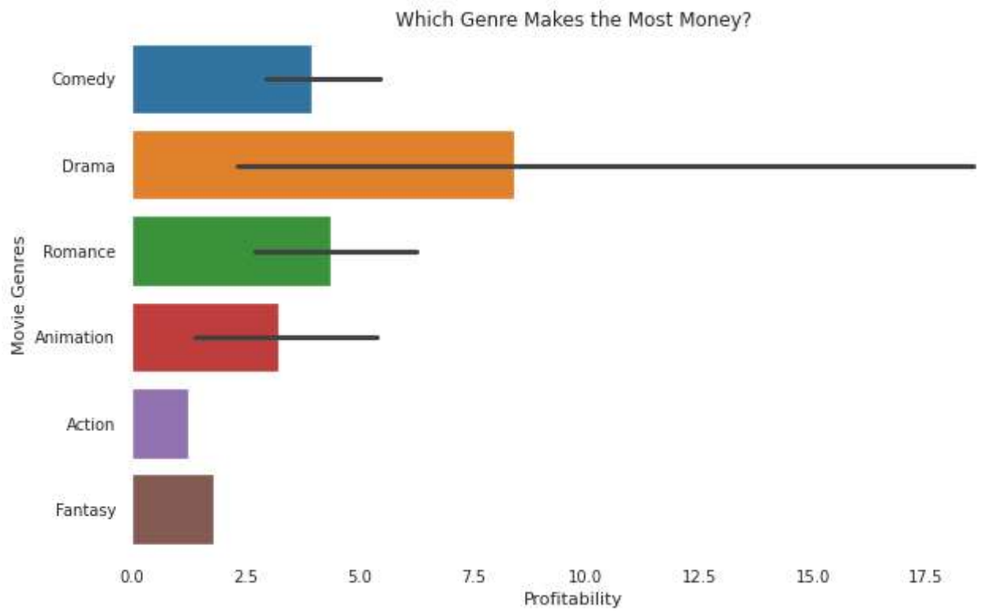
In [9]:
```
plt.text(0.9, 0.7, "Fantasy", size=35, rotation=0.5,
         ha="right", va="top",
         bbox=dict(boxstyle="square",
                   ec=(1., 0.5, 0.5),
                   fc=(1., 0.8, 0.8),
                   )
         )
plt.text(0.5, 0.85, "Drama", size=50, rotation=15.,
         ha="center", va="center",
         bbox=dict(boxstyle="round",
                   ec=(1., 0.5, 0.5),
                   fc=(1., 0.8, 0.8),
                   )
         )

plt.text(0.30, 0.6, "Action", size=30, rotation=25.,
         ha="right", va="top",
         bbox=dict(boxstyle="square",
                   ec=(1., 0.5, 0.5),
                   fc=(1., 0.8, 0.8),
                   )
         )
plt.text(1.5, 0.6, "Romance", size=45, rotation=10.,
         ha="right", va="top",
```

```
        bbox=dict(boxstyle="square",
                  ec=(1., 0.5, 0.5),
                  fc=(1., 0.8, 0.8),
                  )
        )
plt.text(1.4, 0.9, "Comedy", size=45, rotation=0.,
        ha="right", va="top",
        bbox=dict(boxstyle="round",
                  ec=(1., 0.5, 0.5),
                  fc=(1., 0.8, 0.8),
                  )
        )
plt.text(0.8, 0.4, "Animation", size=40, rotation=0.,
        ha="right", va="top",
        bbox=dict(boxstyle="round",
                  ec=(1., 0.5, 0.5),
                  fc=(1., 0.8, 0.8),
                  )
        )

plt.show()
```



In [10]:
```
# Set the width and height of the figure
plt.figure(figsize=(10,6))

# Add title
plt.title("Which Genre Makes the Most Money?")

# Bar chart showing which genre made the most money
sns.barplot(x=df.Profitability, y=df['Genre'])

# Add label for vertical axis
plt.ylabel("Movie Genres")
```

Out[10]:
```
Text(0, 0.5, 'Movie Genres')
```



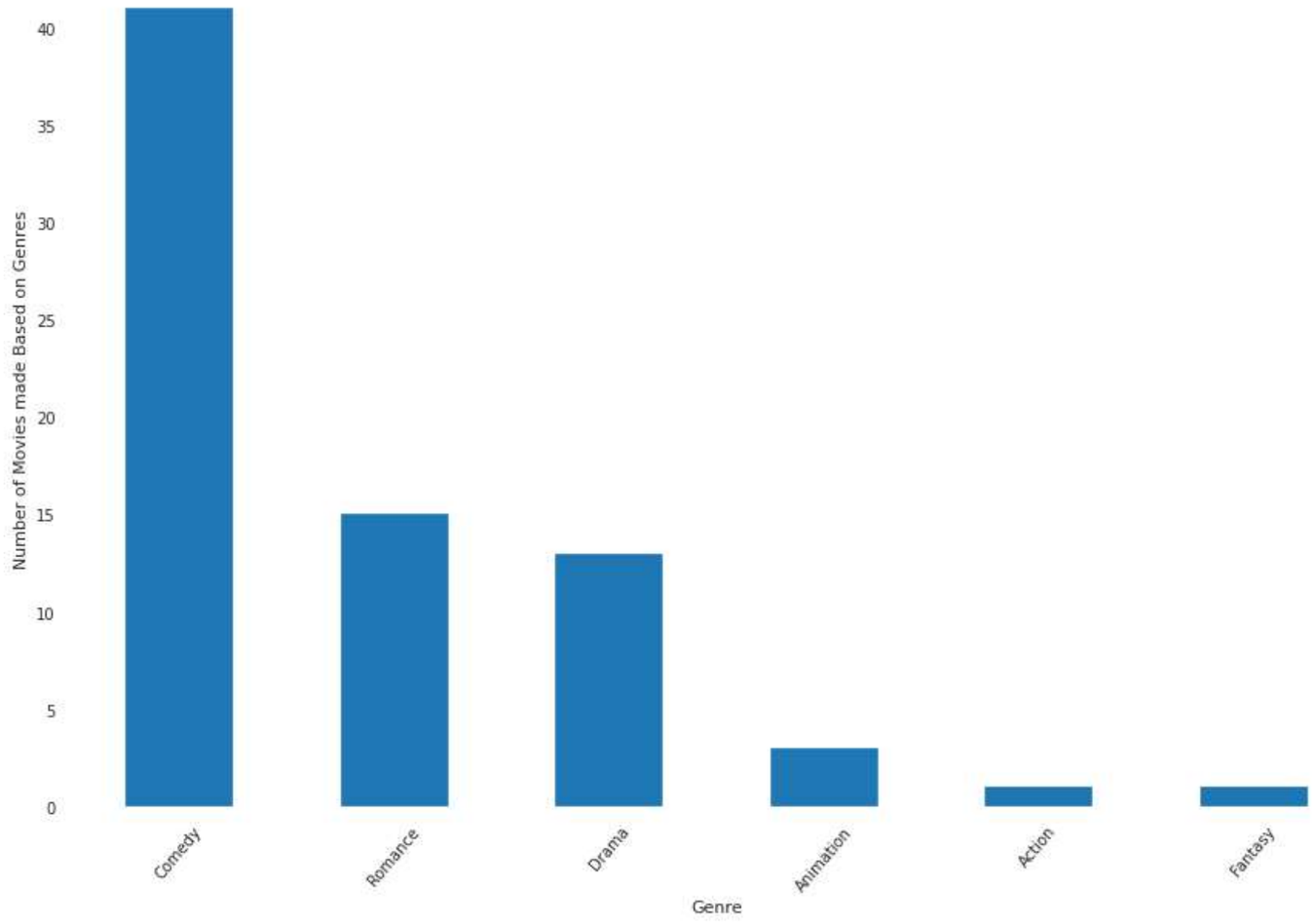
Showing the number of movies made within each genre. From the most movies made based on genre to the least.

In [11]:
```
genre = df.groupby("Genre")
genre.mean().sort_values(by="Profitability",ascending=False).head()
```

Out[11]:

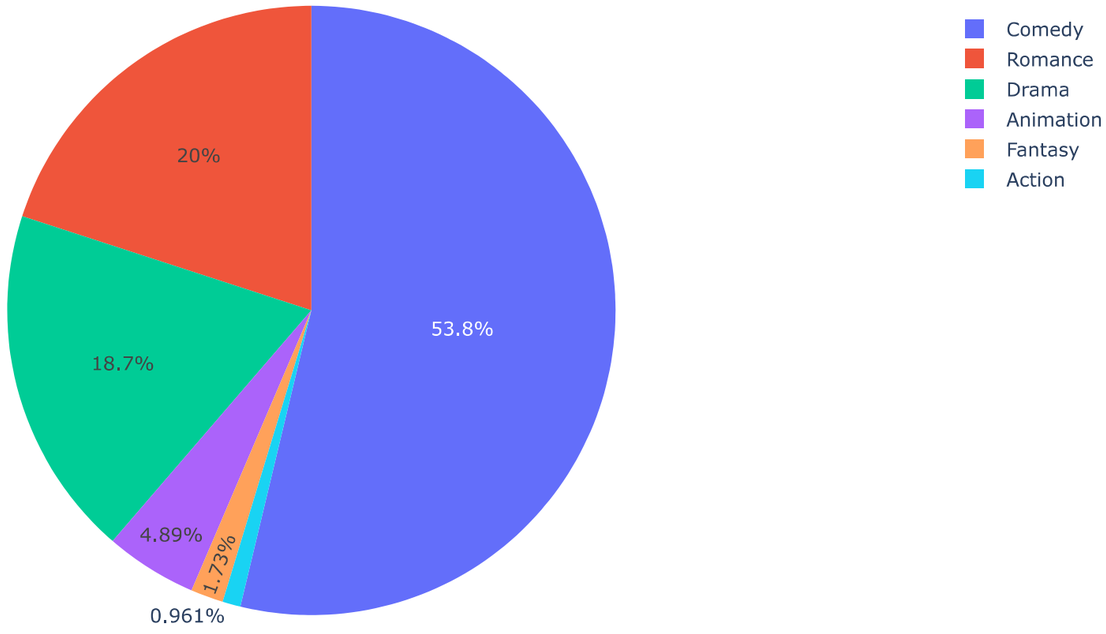
Genre	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
Drama	67.230769	8.407218	51.538462	99.011375	2009.384615
Romance	66.857143	4.371398	48.500000	133.542096	2009.266667
Comedy	61.390244	3.935434	43.487805	130.502578	2008.829268
Animation	76.333333	3.216561	80.333333	356.776811	2009.666667
Fantasy	81.000000	1.783944	73.000000	285.431000	2008.000000

In [12]:
```
plt.figure(figsize=(15,10))
genre.size().sort_values(ascending=False).plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Genre")
plt.ylabel("Number of Movies made Based on Genres")
plt.show()
```



Now we are going to see what movie genre people like the most. It seems like people like watching comedy and action the least favorite movie genre.

In [13]:
```
fig = px.pie(df, values='Audience  score %', names='Genre')
fig.show()
```



In [14]:
```
?WordCloud
```

```
Init signature:
WordCloud(
    font_path=None,
    width=400,
    height=200,
    margin=2,
    ranks_only=None,
    prefer_horizontal=0.9,
    mask=None,
    scale=1,
    color_func=None,
    max_words=200,
    min_font_size=4,
    stopwords=None,
    random_state=None,
    background_color='black',
    max_font_size=None,
    font_step=1,
    mode='RGB',
    relative_scaling='auto',
    regexp=None,
    collocations=True,
    colormap=None,
    normalize_plurals=True,
    contour_width=0,
    contour_color='black',
    repeat=False,
    include_numbers=False,
    min_word_length=0,
    collocation_threshold=30,
)
Docstring:
Word cloud object for generating and drawing.

Parameters
----------
font_path : string
    Font path to the font that will be used (OTF or TTF).
    Defaults to DroidSansMono path on a Linux machine. If you are on
    another OS or don't have this font, you need to adjust this path.

width : int (default=400)
    Width of the canvas.

height : int (default=200)
    Height of the canvas.

prefer_horizontal : float (default=0.90)
    The ratio of times to try horizontal fitting as opposed to vertical.
    If prefer_horizontal < 1, the algorithm will try rotating the word
    if it doesn't fit. (There is currently no built-in way to get only
    vertical words.)

mask : nd-array or None (default=None)
    If not None, gives a binary mask on where to draw words. If mask is not
    None, width and height will be ignored and the shape of mask will be
    used instead. All white (#FF or #FFFFFF) entries will be considerd
    "masked out" while other entries will be free to draw on. [This
    changed in the most recent version!]
```

```
contour_width: float (default=0)
    If mask is not None and contour_width > 0, draw the mask contour.

contour_color: color value (default="black")
    Mask contour color.

scale : float (default=1)
    Scaling between computation and drawing. For large word-cloud images,
    using scale instead of larger canvas size is significantly faster, but
    might lead to a coarser fit for the words.

min_font_size : int (default=4)
    Smallest font size to use. Will stop when there is no more room in this
    size.

font_step : int (default=1)
    Step size for the font. font_step > 1 might speed up computation but
    give a worse fit.

max_words : number (default=200)
    The maximum number of words.

stopwords : set of strings or None
    The words that will be eliminated. If None, the build-in STOPWORDS
    list will be used. Ignored if using generate_from_frequencies.

background_color : color value (default="black")
    Background color for the word cloud image.

max_font_size : int or None (default=None)
    Maximum font size for the largest word. If None, height of the image is
    used.

mode : string (default="RGB")
    Transparent background will be generated when mode is "RGBA" and
    background_color is None.

relative_scaling : float (default='auto')
    Importance of relative word frequencies for font-size.  With
    relative_scaling=0, only word-ranks are considered.  With
    relative_scaling=1, a word that is twice as frequent will have twice
    the size.  If you want to consider the word frequencies and not only
    their rank, relative_scaling around .5 often looks good.
    If 'auto' it will be set to 0.5 unless repeat is true, in which
    case it will be set to 0.

    .. versionchanged: 2.0
        Default is now 'auto'.

color_func : callable, default=None
    Callable with parameters word, font_size, position, orientation,
    font_path, random_state that returns a PIL color for each word.
    Overwrites "colormap".
    See colormap for specifying a matplotlib colormap instead.
    To create a word cloud with a single color, use
    ``color_func=lambda *args, **kwargs: "white"``.
    The single color can also be specified using RGB code. For example
    ``color_func=lambda *args, **kwargs: (255,0,0)`` sets color to red.

regexp : string or None (optional)
```

```
    Regular expression to split the input text into tokens in process_text.
    If None is specified, ``r"\w[\w']+"`` is used. Ignored if using
    generate_from_frequencies.

collocations : bool, default=True
    Whether to include collocations (bigrams) of two words. Ignored if using
    generate_from_frequencies.


    .. versionadded: 2.0

colormap : string or matplotlib colormap, default="viridis"
    Matplotlib colormap to randomly draw colors from for each word.
    Ignored if "color_func" is specified.

    .. versionadded: 2.0

normalize_plurals : bool, default=True
    Whether to remove trailing 's' from words. If True and a word
    appears with and without a trailing 's', the one with trailing 's'
    is removed and its counts are added to the version without
    trailing 's' -- unless the word ends with 'ss'. Ignored if using
    generate_from_frequencies.

repeat : bool, default=False
    Whether to repeat words and phrases until max_words or min_font_size
    is reached.

include_numbers : bool, default=False
    Whether to include numbers as phrases or not.

min_word_length : int, default=0
    Minimum number of letters a word must have to be included.

collocation_threshold: int, default=30
    Bigrams must have a Dunning likelihood collocation score greater than this
    parameter to be counted as bigrams. Default of 30 is arbitrary.

    See Manning, C.D., Manning, C.D. and Schütze, H., 1999. Foundations of
    Statistical Natural Language Processing. MIT press, p. 162
    https://nlp.stanford.edu/fsnlp/promo/colloc.pdf#page=22

Attributes
----------
``words_`` : dict of string to float
    Word tokens with associated frequency.

    .. versionchanged: 2.0
        ``words_`` is now a dictionary

``layout_`` : list of tuples (string, int, (int, int), int, color))
    Encodes the fitted word cloud. Encodes for each word the string, font
    size, position, orientation and color.

Notes
-----
Larger canvases with make the code significantly slower. If you need a
large word cloud, try a lower canvas size, and set the scale parameter.

The algorithm might give more weight to the ranking of the words
```

```
than their actual frequencies, depending on the ``max_font_size`` and the
scaling heuristic.
File:           /opt/conda/lib/python3.7/site-packages/wordcloud/wordcloud.py
Type:           type
Subclasses:
```

This word cloud shows which genre has the most movies made from it. Since comedy is the biggest, that means people have made more comedy movies than any other genre.

In [15]:
```
text = " ".join(review for review in df.Genre)
print ("There are {} words in the combination of all review.".format(len(text)))
```

```
There are 529 words in the combination of all review.
```

In [16]:
```
# Generate a word cloud image
wordcloud = WordCloud(background_color="white").generate(text)

# Display the generated image:
# the matplotlib way:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Creating a shape for my world cloud.

In [17]:
```
film_mask = np.array(Image.open("../input/film-image/film.png"))
film_mask
```

Out[17]:
```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       ...,
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1]], dtype=uint8)
```

In [18]:
```
def transform_format(val):
    if val == 0:
        return 255
    else:
        return val
```

In [19]:
```
transformed_film_mask = np.ndarray((film_mask.shape[0],film_mask.shape[1]), np.int32)
for i in range(len(film_mask)):
    transformed_film_mask[i] = list(map(transform_format, film_mask[i]))
```

```
# Check the expected result of your mask
transformed_film_mask
```

Out[19]:
```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       ...,
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1]], dtype=int32)
```

Image might look weird because there are only 6 words. For it to look better, try doing an image word cloud that has more words in it.

In [20]:
```
# Create a word cloud image
wc = WordCloud(background_color="white", max_words=1000, mask=transformed_film_mask,
               contour_width=3, contour_color='black')

# Generate a wordcloud
wc.generate(text)

# show
plt.figure(figsize=[10,5])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Lead Studios

This graphs which Studio makes the most money compare to others.

- Bar Charts
- Pie Charts
- Scatter plot

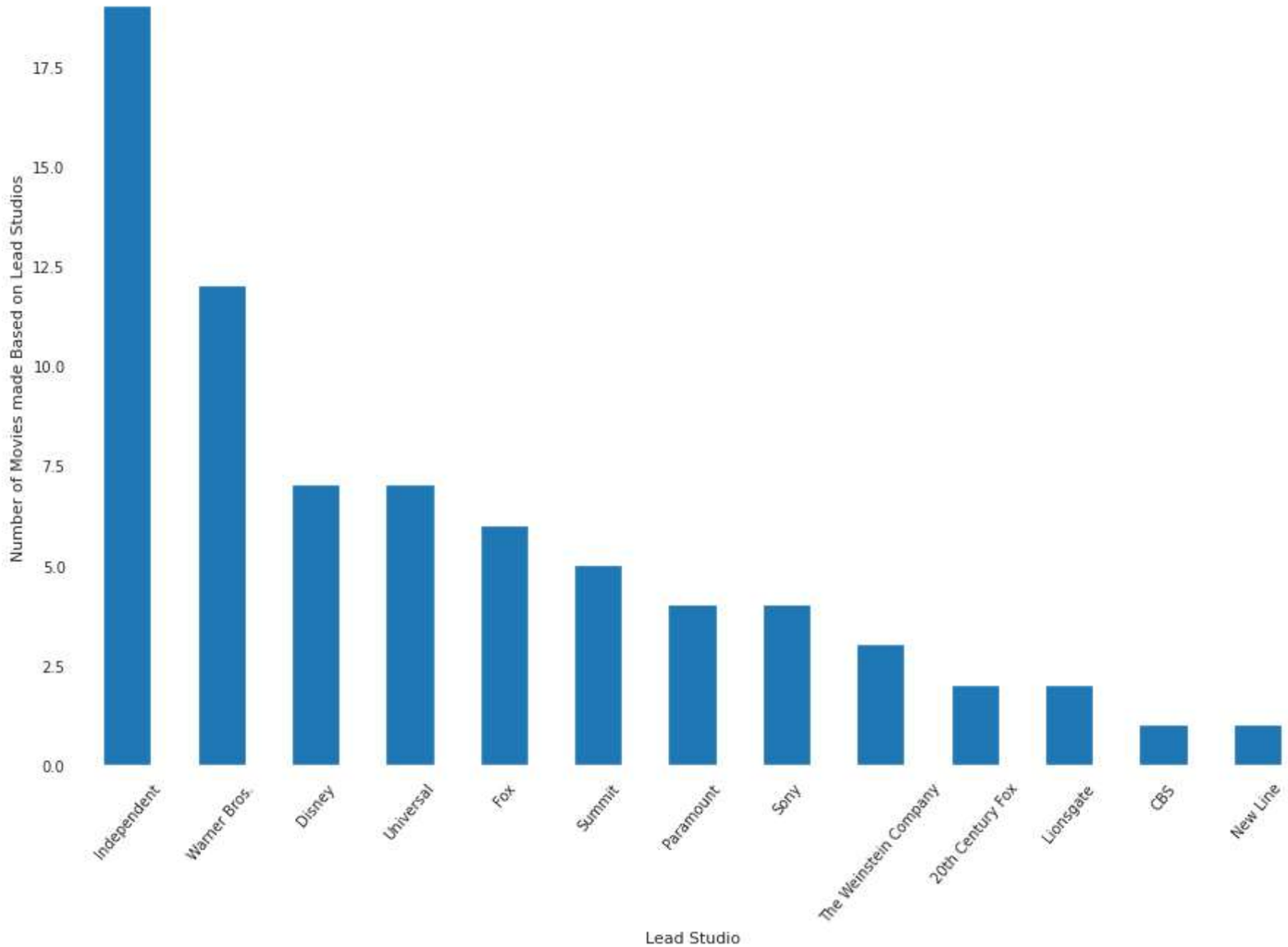
In [21]:
```
ls = df.groupby("Lead Studio")
ls.mean().sort_values(by="Profitability",ascending=False).head()
```

Out[21]:

Lead Studio	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
Disney	71.857143	7.406010	65.285714	288.657512	2009.000000
Independent	63.500000	6.582046	47.166667	81.418190	2009.263158
Summit	73.200000	6.377962	39.200000	248.452599	2009.000000
Sony	69.500000	4.836326	52.000000	100.780007	2009.250000
Fox	69.000000	4.511522	51.833333	120.428050	2008.833333

This shows Independent studio made the most movies while New Line studio made the least movies.

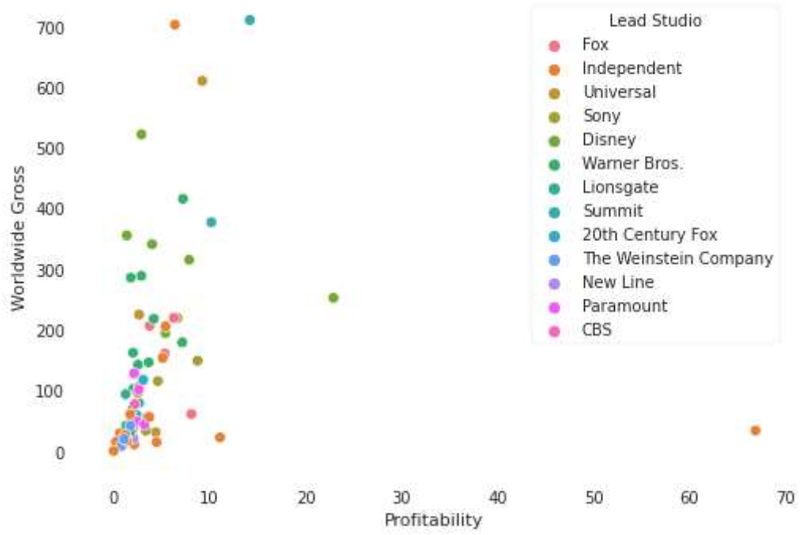
In [22]:
```
plt.figure(figsize=(15,10))
ls.size().sort_values(ascending=False).plot.bar()
plt.xticks(rotation=50)
plt.xlabel("Lead Studio")
plt.ylabel("Number of Movies made Based on Lead Studios")
plt.show()
```



This shows which studio made the most money on a film. Base on the chart it looks like Summit studio has made more money on worldwide gross and so did Independent studio. If you look at the profitability section, then you will see Indpendpent made the money profitability wise.

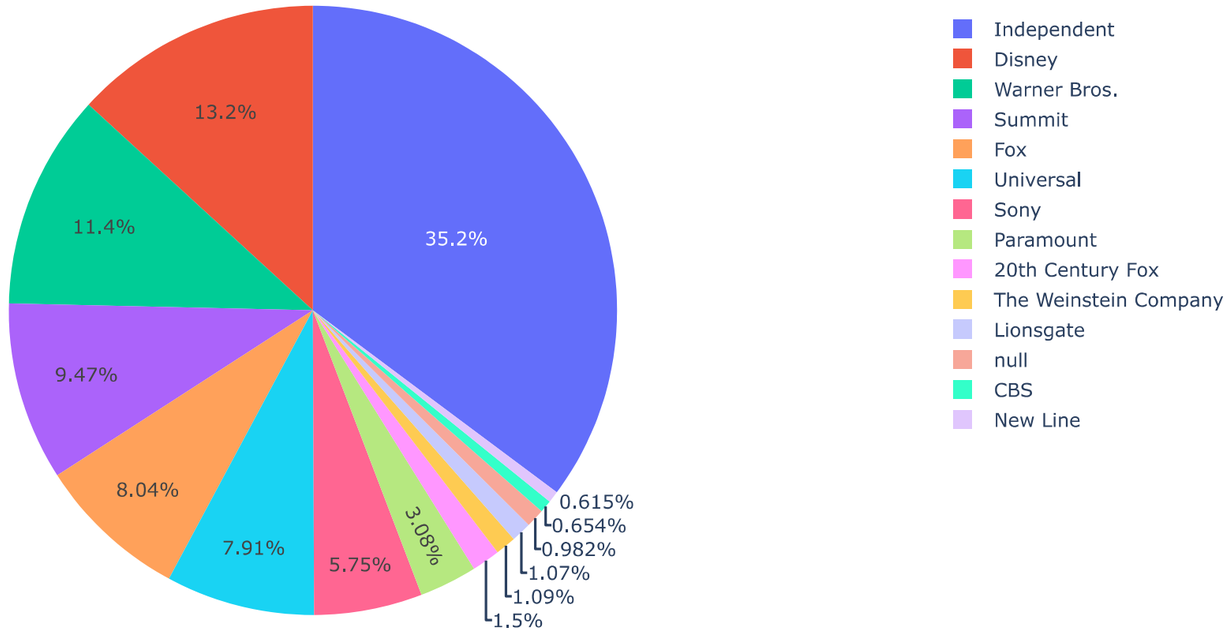
In [23]:
```
sns.scatterplot(x=df['Profitability'], y=df['Worldwide Gross'], hue=df['Lead Studio'])
```

Out[23]:
```
<AxesSubplot:xlabel='Profitability', ylabel='Worldwide Gross'>
```



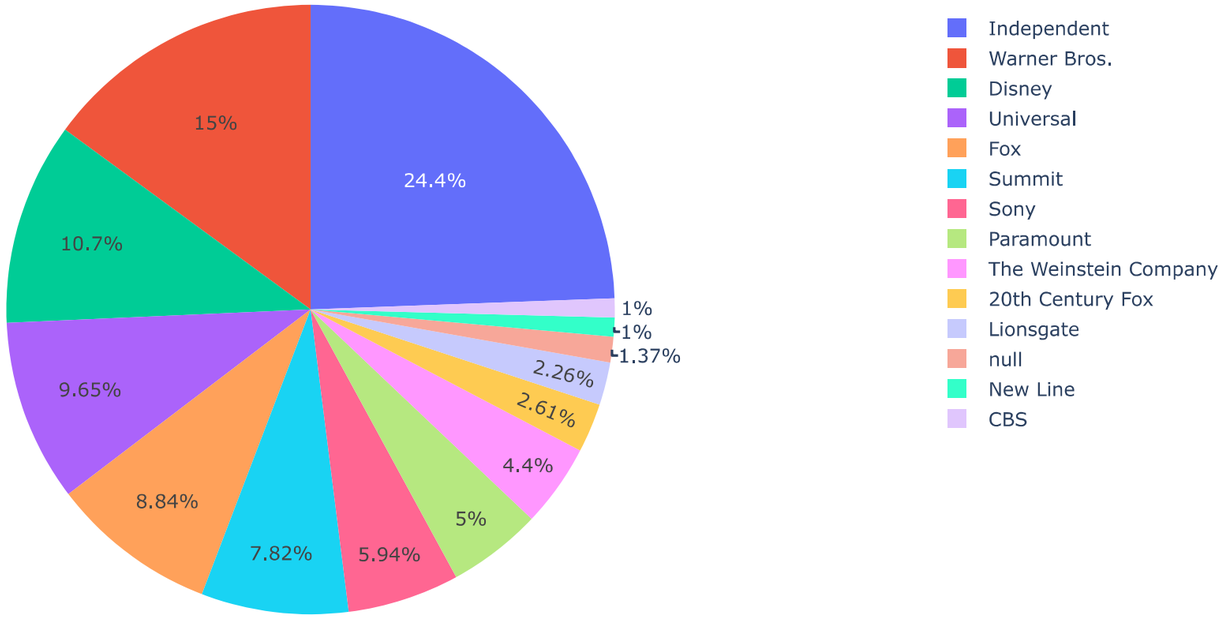
This shows which studio makes the most money. They added up the numbers of profitability fr each studio and Independent studio made the most money. This is because they also made the most movies, but Disney is the second studio to make the most money even though they are the third studio to make the most movies.

In [24]:
```
fig = px.pie(df, values='Profitability', names='Lead Studio')
fig.show()
```



Now we are going to see what studio made the most movies with the most audience score. It seems like Independent made the most movies that people enjoyed.

In [25]:
```
fig = px.pie(df, values='Audience  score %', names='Lead Studio')
fig.show()
```


24.4%
15%
10.7%
9.65%
8.84%
7.82%
5.94%
5%
4.4%
2.61%
2.26%
1.37%
1%
1%
Independent
Warner Bros.
Disney
Universal
Fox
Summit
Sony
Paramount
The Weinstein Company
20th Century Fox
Lionsgate
null
New Line
CBS