Importing the necesssary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
```

Annual Ticket Sales analysis

```
AnnualTicketSales = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/AnnualTicketSales.csv')
```

```
AnnualTicketSales.head()
```

```
   YEAR    TICKETS SOLD TOTAL BOX OFFICE TOTAL INFLATION ADJUSTED BOX
OFFICE  \
0  2021    42,37,74,881   $3,881,777,912
$3,881,777,912
1  2020    22,36,38,958   $2,048,534,616
$2,048,534,616
2  2019  1,22,85,41,629  $11,253,443,955
$11,253,444,050
3  2018  1,31,15,36,128  $11,948,096,650
$12,013,670,952
4  2017  1,22,56,39,761  $10,993,991,460
$11,226,860,216

  AVERAGE TICKET PRICE  Unnamed: 5
0                $9.16         NaN
1                $9.16         NaN
2                $9.16         NaN
3                $9.11         NaN
4                $8.97         NaN
```

```
#The number of of null values
AnnualTicketSales.isnull().sum()
```

```
YEAR                                  0
TICKETS SOLD                          0
TOTAL BOX OFFICE                      0
TOTAL INFLATION ADJUSTED BOX OFFICE   0
AVERAGE TICKET PRICE                  0
Unnamed: 5                           27
dtype: int64
```

```
# Removing Unnecessay Unnamed: 5 column
AnnualTicketSales = AnnualTicketSales.drop('Unnamed: 5', axis=1)

AnnualTicketSales.head()
```

```
   YEAR    TICKETS SOLD TOTAL BOX OFFICE TOTAL INFLATION ADJUSTED BOX
OFFICE  \
0  2021    42,37,74,881   $3,881,777,912
$3,881,777,912
1  2020    22,36,38,958   $2,048,534,616
$2,048,534,616
2  2019  1,22,85,41,629  $11,253,443,955
$11,253,444,050
3  2018  1,31,15,36,128  $11,948,096,650
$12,013,670,952
4  2017  1,22,56,39,761  $10,993,991,460
$11,226,860,216

  AVERAGE TICKET PRICE
0                $9.16
1                $9.16
2                $9.16
3                $9.11
4                $8.97
```

```
AnnualTicketSales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 5 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   YEAR                                  27 non-null     int64
 1   TICKETS SOLD                          27 non-null     object
 2   TOTAL BOX OFFICE                      27 non-null     object
 3   TOTAL INFLATION ADJUSTED BOX OFFICE  27 non-null     object
 4   AVERAGE TICKET PRICE                  27 non-null     object
dtypes: int64(1), object(4)
memory usage: 1.2+ KB
```

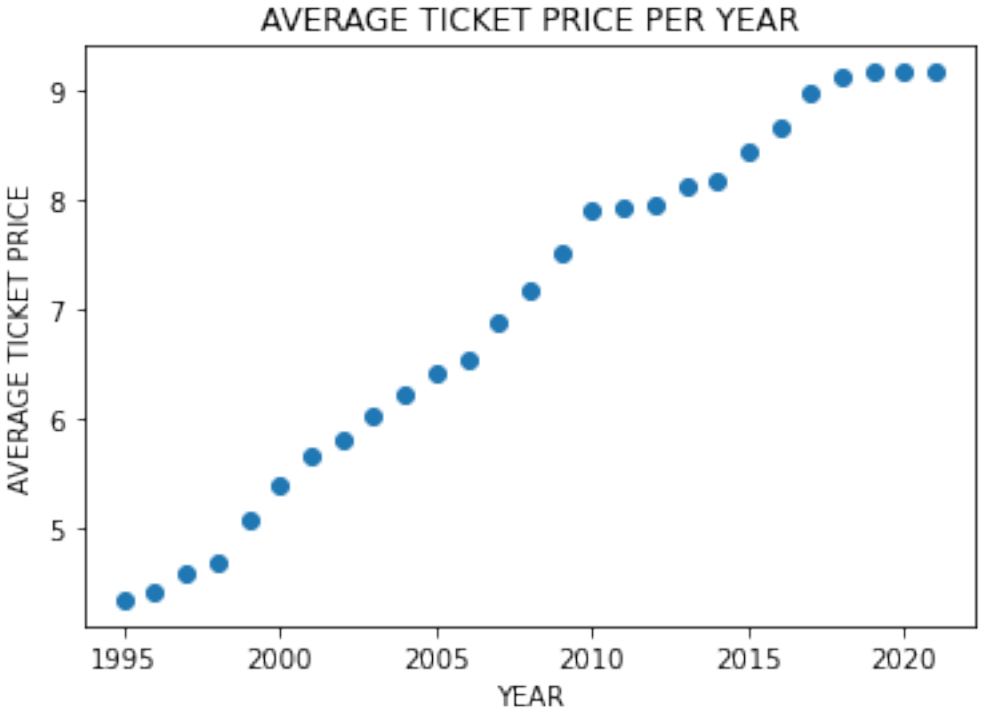
```
#Converting TICKET SOLD from object to int
AnnualTicketSales['TICKETS SOLD']=AnnualTicketSales['TICKETS
SOLD'].replace(',','', regex=True)
AnnualTicketSales['TICKETS
SOLD']=pd.to_numeric(AnnualTicketSales['TICKETS SOLD'])

#Converting TOTAL BOX OFFICE from object to int
AnnualTicketSales['TOTAL BOX OFFICE']=AnnualTicketSales['TOTAL BOX
OFFICE'].replace(',','', regex=True)
AnnualTicketSales['TOTAL BOX OFFICE']=AnnualTicketSales['TOTAL BOX
OFFICE'].str.replace('$','', regex=True)
```

```
AnnualTicketSales['TOTAL BOX
OFFICE']=pd.to_numeric(AnnualTicketSales['TOTAL BOX OFFICE'])

#Converting TOTAL INFLATION ADJUSTED BOX OFFICE from object to int
AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE']=AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE'].replace(',','', regex=True)
AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE']=AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE'].str.replace('$','', regex=True)
AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE']=pd.to_numeric(AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE'])


#Converting AVERAGE TICKET PRICE from object to int
AnnualTicketSales['AVERAGE TICKET PRICE']=AnnualTicketSales['AVERAGE
TICKET PRICE'].replace(',','', regex=True)
AnnualTicketSales['AVERAGE TICKET PRICE']=AnnualTicketSales['AVERAGE
TICKET PRICE'].str.replace('$','', regex=True)
AnnualTicketSales['AVERAGE TICKET
PRICE']=pd.to_numeric(AnnualTicketSales['AVERAGE TICKET PRICE'])

#Final DataFrame
AnnualTicketSales.head()
```

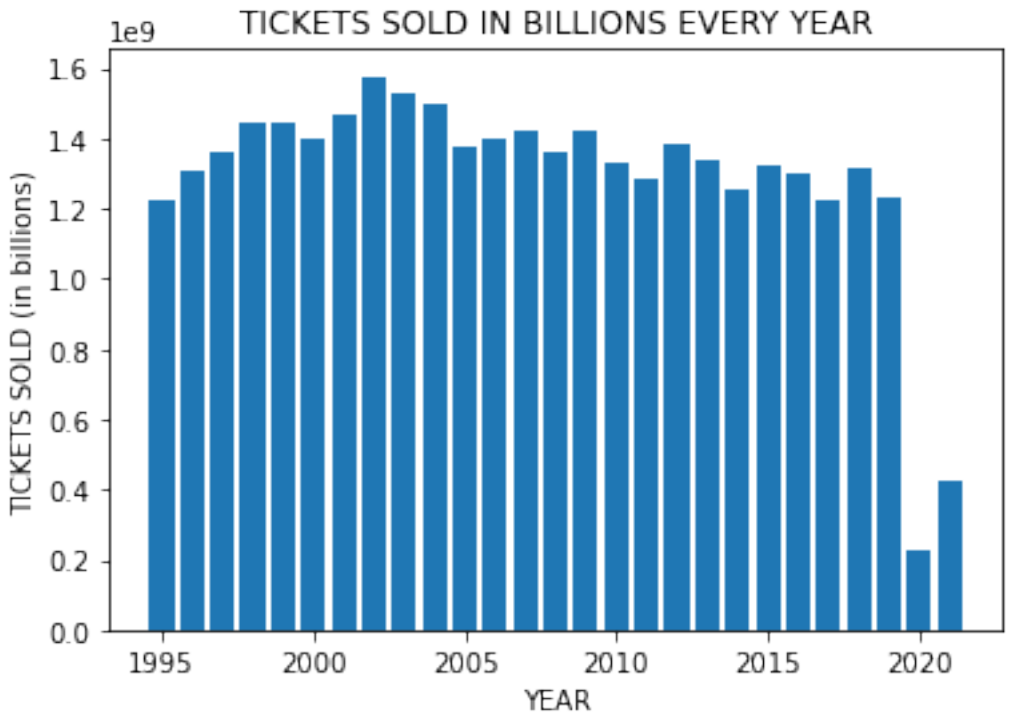
```
   YEAR  TICKETS SOLD  TOTAL BOX OFFICE  TOTAL INFLATION ADJUSTED BOX
OFFICE  \
0  2021     423774881        3881777912
3881777912
1  2020     223638958        2048534616
2048534616
2  2019    1228541629       11253443955
11253444050
3  2018    1311536128       11948096650
12013670952
4  2017    1225639761       10993991460
11226860216

   AVERAGE TICKET PRICE
0                  9.16
1                  9.16
2                  9.16
3                  9.11
4                  8.97
```

```
# Average Ticket Price change yearly
figure=plt.Figure()
plt.scatter(AnnualTicketSales['YEAR'], AnnualTicketSales['AVERAGE
TICKET PRICE'])
```

```
plt.ylabel('AVERAGE TICKET PRICE')
plt.xlabel('YEAR')
plt.title('AVERAGE TICKET PRICE PER YEAR')
plt.show()
```




```
# Total tickets sold every year
plt.bar(AnnualTicketSales['YEAR'], AnnualTicketSales['TICKETS SOLD'])
plt.xlabel("YEAR")
plt.ylabel("TICKETS SOLD (in billions)")
plt.title("TICKETS SOLD IN BILLIONS EVERY YEAR")
plt.show()
```




```
#Total Box Office and Inflation from 1995 to 2020
fig = go.Figure()

# Add graph object
fig.add_trace(go.Scatter(x=AnnualTicketSales['YEAR'],
                         y=AnnualTicketSales['TOTAL BOX OFFICE'],
                         mode='markers+lines',
                         name='TOTAL BOX OFFICE',
                         marker=dict(color='rgba(200, 148, 237,.7)')))

fig.add_trace(go.Scatter(x=AnnualTicketSales['YEAR'],
                         y=AnnualTicketSales['TOTAL INFLATION ADJUSTED
BOX OFFICE'],
                         mode='markers+lines',
                         name='TOTAL INFLATION ADJUSTED BOX OFFICE',
                         marker=dict(color='rgba(205, 210, 250,.7)')))

# Update layout
fig.update_layout(title='TOTAL BOX OFFICE AND INFLATION EACH YEAR',
                  yaxis=dict(showgrid=False, showline=True,
linecolor='rgb(0,0,0)', title='USD'),
                  xaxis=dict(showgrid=False, showline=True,
linecolor='rgb(0,0,0)', title='YEAR'),
                  paper_bgcolor='rgb(255,255,255)',
                  plot_bgcolor='rgb(255,255,255)',
                  hovermode='x unified')
```

```
# Add annotation with arrow
fig.add_annotation(x=AnnualTicketSales.iloc[AnnualTicketSales['TOTAL
INFLATION ADJUSTED BOX OFFICE'].argmax()].YEAR,
                   y=AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX
OFFICE']
[AnnualTicketSales.YEAR==AnnualTicketSales.iloc[AnnualTicketSales['TOT
AL INFLATION ADJUSTED BOX OFFICE'].argmax()].YEAR].values[0],
                   text='Highest TOTAL INFLATION BOX OFFICE')

fig.add_annotation(x=AnnualTicketSales.iloc[AnnualTicketSales['TOTAL
BOX OFFICE'].argmax()].YEAR,
                   y=AnnualTicketSales['TOTAL BOX OFFICE']
[AnnualTicketSales.YEAR==AnnualTicketSales.iloc[AnnualTicketSales['TOT
AL BOX OFFICE'].argmax()].YEAR].values[0],
                   text='Highest TOTAL BOX OFFICE')
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":[{"marker":
{"color":"rgba(200, 148,
237,.7)"},"mode":"markers+lines","name":"TOTAL BOX
OFFICE","type":"scatter","x":
[2021,2020,2019,2018,2017,2016,2015,2014,2013,2012,2011,2010,2009,2008
,2007,2006,2005,2004,2003,2002,2001,2000,1999,1998,1997,1996,1995],"y"
:
[3881777912,2048534616,11253443955,11948096650,10993991460,11267115924
,11155900636,10272985008,10887446341,10992141616,10173519704,104822540
25,10639257284,9750744148,9769854914,9161738221,8800805718,9287996519,
9193277289,9155147215,8296849636,7532311479,7338894852,6771575283,6230
235770,5769078886,5314421390]},{"marker":{"color":"rgba(205, 210,
250,.7)"},"mode":"markers+lines","name":"TOTAL INFLATION ADJUSTED BOX
OFFICE","type":"scatter","x":
[2021,2020,2019,2018,2017,2016,2015,2014,2013,2012,2011,2010,2009,2008
,2007,2006,2005,2004,2003,2002,2001,2000,1999,1998,1997,1996,1995],"y"
:
[3881777912,2048534616,11253444050,12013670952,11226860216,11931416424
,12121948075,11517810744,12266787382,12649244986,11751502955,121695090
32,12994051137,12439665380,13007535993,12812442671,12576499367,1370016
5883,13965240914,14433929789,13427407722,12800734319,13233123027,13225
505439,12433322785,11955781912,11190826105]}],"layout":{"annotations":
[{"text":"Highest TOTAL INFLATION BOX
OFFICE","x":2002,"y":14433929789},{"text":"Highest TOTAL BOX
OFFICE","x":2018,"y":11948096650}],"hovermode":"x
unified","paper_bgcolor":"rgb(255,255,255)","plot_bgcolor":"rgb(255,25
5,255)","template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5}},"type":"bar"}],"barpolar":[{"marker":
{"line":{"color":"#E5ECF6","width":0.5}},"type":"barpolar"}],"carpet":
[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
```

{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"histogram"}],"histogram2d":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":**true**,"type":"pie"}],"scatter":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter"}],"scatter3d":
[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":

```
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
```

```
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"TOTAL BOX OFFICE AND INFLATION EACH YEAR"},"xaxis":
{"linecolor":"rgb(0,0,0)","showgrid":false,"showline":true,"title":
{"text":"YEAR"}},"yaxis":
{"linecolor":"rgb(0,0,0)","showgrid":false,"showline":true,"title":
{"text":"USD"}}}}
```


```
#Correlation matrix
cormat=AnnualTicketSales.corr()
round(cormat,2)
```

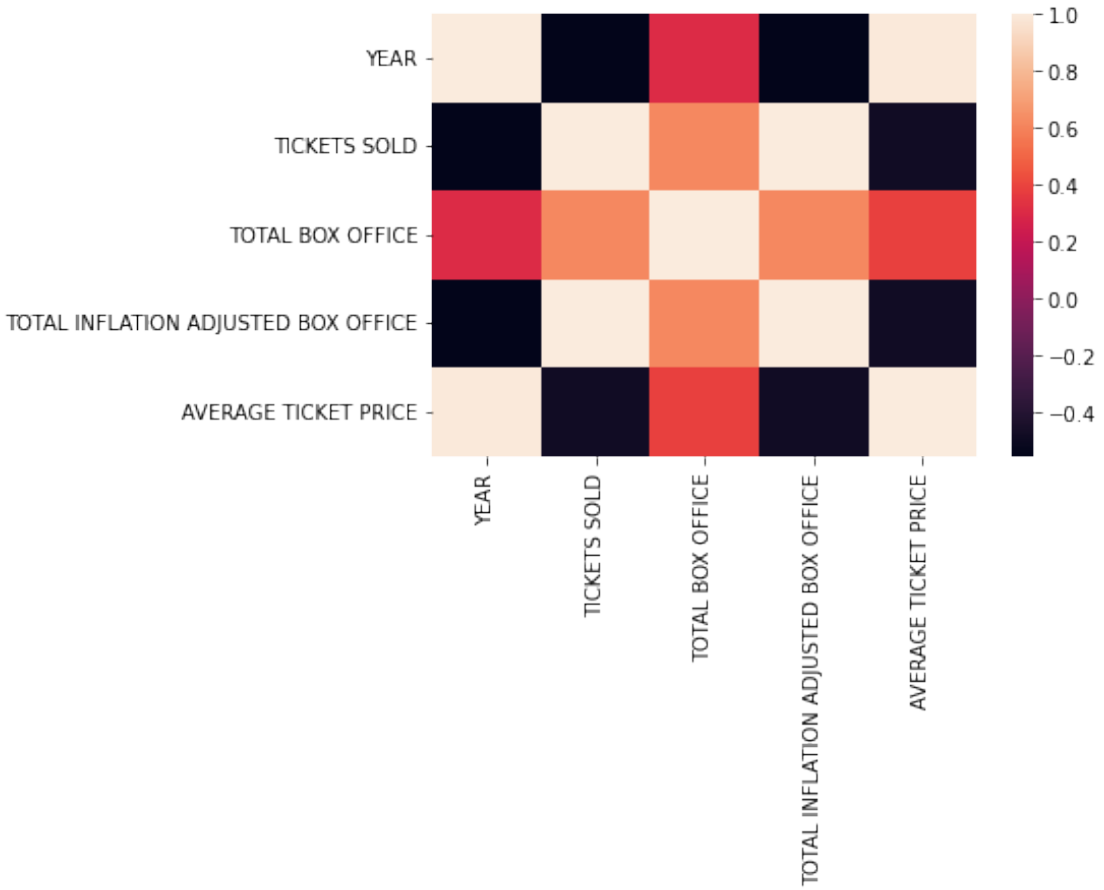
```
                                    YEAR  TICKETS SOLD  TOTAL BOX
OFFICE  \
YEAR                                1.00         -0.55
0.30
TICKETS SOLD                       -0.55          1.00
0.62
TOTAL BOX OFFICE                    0.30          0.62
1.00
TOTAL INFLATION ADJUSTED BOX OFFICE -0.55          1.00
0.62
AVERAGE TICKET PRICE                0.99         -0.48
0.39

                                    TOTAL INFLATION ADJUSTED BOX
OFFICE  \
YEAR                                                          -
0.55
TICKETS SOLD
1.00
TOTAL BOX OFFICE
0.62
```

```
TOTAL INFLATION ADJUSTED BOX OFFICE
1.00
AVERAGE TICKET PRICE                                                    -
0.48
```

	AVERAGE TICKET PRICE
YEAR	0.99
TICKETS SOLD	-0.48
TOTAL BOX OFFICE	0.39
TOTAL INFLATION ADJUSTED BOX OFFICE	-0.48
AVERAGE TICKET PRICE	1.00

```
# plotting the correlation matrix
sns.heatmap(cormat)
```

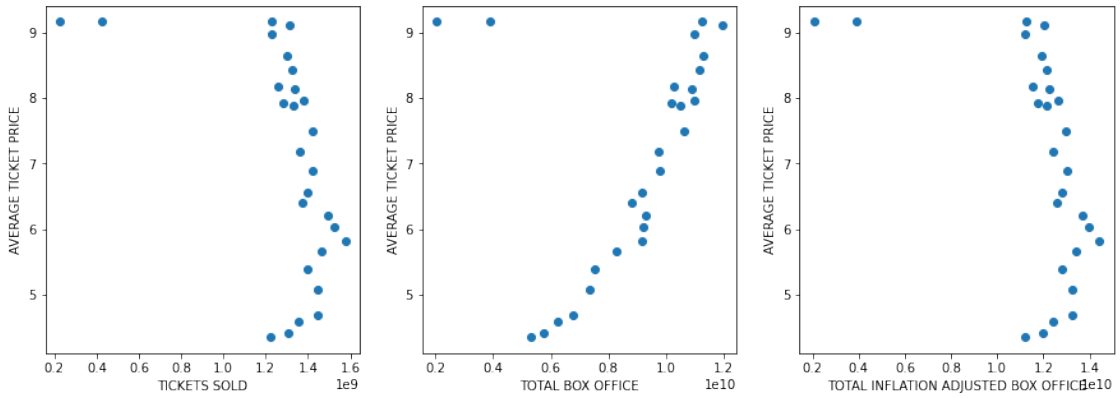
```
<AxesSubplot:>
```




```
#plotting AVERAGE TICKET PRICE AND other variables in subplots to
observe the correlation

plt_1=plt.figure(figsize=[15,5])
#plot 1
plt.subplot(1,3,1)
```

```
plt.scatter(y=AnnualTicketSales['AVERAGE TICKET PRICE'],
x=AnnualTicketSales['TICKETS SOLD'])
plt.ylabel('AVERAGE TICKET PRICE')
plt.xlabel('TICKETS SOLD')

#plot 2
plt.subplot(1,3,2)
plt.scatter(y=AnnualTicketSales['AVERAGE TICKET PRICE'],
x=AnnualTicketSales['TOTAL BOX OFFICE'])
plt.ylabel('AVERAGE TICKET PRICE')
plt.xlabel('TOTAL BOX OFFICE')

#plot 3
plt.subplot(1,3,3)
plt.scatter(y=AnnualTicketSales['AVERAGE TICKET PRICE'],
x=AnnualTicketSales['TOTAL INFLATION ADJUSTED BOX OFFICE'])
plt.ylabel('AVERAGE TICKET PRICE')
plt.xlabel('TOTAL INFLATION ADJUSTED BOX OFFICE')
```

```
Text(0.5, 0, 'TOTAL INFLATION ADJUSTED BOX OFFICE')
```



Highest Grossers Analysis

```
HighestGrossers = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/HighestGrossers.csv')

HighestGrossers.head()
```

```
   YEAR                                MOVIE      GENRE MPAA RATING  \
0  1995                      Batman Forever      Drama       PG-13
1  1996                   Independence Day  Adventure       PG-13
2  1997                       Men in Black  Adventure       PG-13
3  1998                            Titanic  Adventure       PG-13
4  1999  Star Wars Ep. I: The Phantom Menace  Adventure          PG

          DISTRIBUTOR TOTAL FOR YEAR TOTAL IN 2019 DOLLARS TICKETS
SOLD
```

```
0        Warner Bros.   $184,031,112          $387,522,978
4,23,06,002
1    20th Century Fox   $306,169,255          $634,504,608
6,92,69,062
2       Sony Pictures   $250,650,052          $500,207,943
5,46,07,854
3  Paramount Pictures   $443,319,081          $865,842,808
9,45,24,324
4    20th Century Fox   $430,443,350          $776,153,749
8,47,32,942
```

```
## Data cleaning  for TOTAL FOR YEAR, TOTAL IN 2019 DOLLARS AND
TICKETS SOLD columns
#Converting TICKETS SOLD from object to int
HighestGrossers['TICKETS SOLD']=HighestGrossers['TICKETS
SOLD'].replace(',','', regex=True)
HighestGrossers['TICKETS SOLD']=pd.to_numeric(HighestGrossers['TICKETS
SOLD'])

#Converting TOTAL FOR YEAR from object to int
HighestGrossers['TOTAL FOR YEAR']=HighestGrossers['TOTAL FOR
YEAR'].replace(',','', regex=True)
HighestGrossers['TOTAL FOR YEAR']=HighestGrossers['TOTAL FOR
YEAR'].str.replace('$','', regex=True)
HighestGrossers['TOTAL FOR YEAR']=pd.to_numeric(HighestGrossers['TOTAL
FOR YEAR'])

#Converting TOTAL IN 2019 DOLLARS from object to int
HighestGrossers['TOTAL IN 2019 DOLLARS']=HighestGrossers['TOTAL IN
2019 DOLLARS'].replace(',','', regex=True)
HighestGrossers['TOTAL IN 2019 DOLLARS']=HighestGrossers['TOTAL IN
2019 DOLLARS'].str.replace('$','', regex=True)
HighestGrossers['TOTAL IN 2019
DOLLARS']=pd.to_numeric(HighestGrossers['TOTAL IN 2019 DOLLARS'])

HighestGrossers.head()
```

```
   YEAR                                MOVIE      GENRE MPAA RATING  \
0  1995                        Batman Forever      Drama       PG-13
1  1996                     Independence Day  Adventure       PG-13
2  1997                         Men in Black  Adventure       PG-13
3  1998                              Titanic  Adventure       PG-13
4  1999  Star Wars Ep. I: The Phantom Menace  Adventure          PG

          DISTRIBUTOR  TOTAL FOR YEAR  TOTAL IN 2019 DOLLARS  TICKETS
SOLD
0        Warner Bros.       184031112              387522978
42306002
1    20th Century Fox       306169255              634504608
69269062
```

```
2       Sony Pictures       250650052              500207943
54607854
3  Paramount Pictures       443319081              865842808
94524324
4    20th Century Fox       430443350              776153749
84732942
```

```
HighestGrossers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   YEAR                   27 non-null     int64
 1   MOVIE                  27 non-null     object
 2   GENRE                  24 non-null     object
 3   MPAA RATING            27 non-null     object
 4   DISTRIBUTOR            27 non-null     object
 5   TOTAL FOR YEAR         27 non-null     int64
 6   TOTAL IN 2019 DOLLARS  27 non-null     int64
 7   TICKETS SOLD           27 non-null     int64
dtypes: int64(4), object(4)
memory usage: 1.8+ KB
```

```
HighestGrossers.isnull().sum()
```

```
YEAR                    0
MOVIE                   0
GENRE                   3
MPAA RATING             0
DISTRIBUTOR             0
TOTAL FOR YEAR          0
TOTAL IN 2019 DOLLARS   0
TICKETS SOLD            0
dtype: int64
```

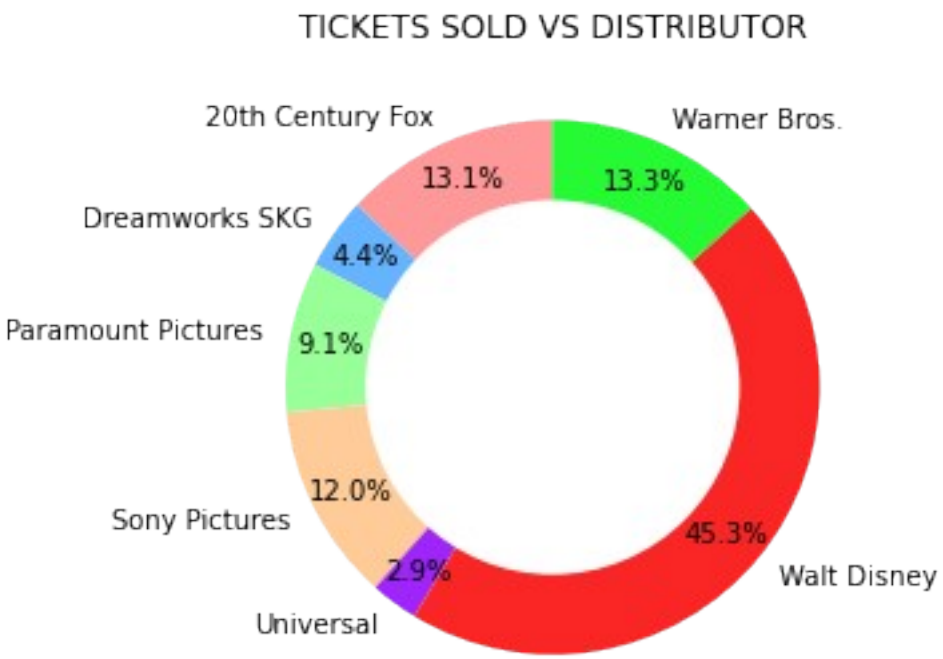
Analyze for Distributors

```
# group by for distributors
group_by_distributor = HighestGrossers.groupby(['DISTRIBUTOR'],
as_index=False).sum()
group_by_distributor=group_by_distributor.drop(labels='YEAR', axis=1)
group_by_distributor
```

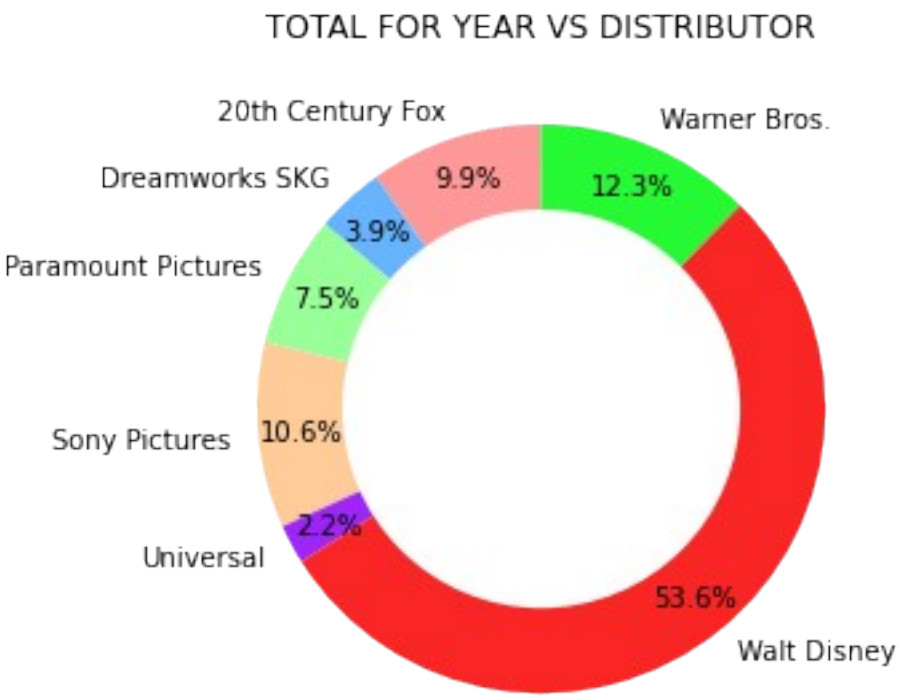
```
          DISTRIBUTOR  TOTAL FOR YEAR  TOTAL IN 2019 DOLLARS  TICKETS
SOLD
0    20th Century Fox      1116883182             1954071528
213326586
1      Dreamworks SKG       441226247              650826473
71050925
```

```
2  Paramount Pictures       845430951             1356955439
148139240
3       Sony Pictures      1195304585             1789160942
195323247
4           Universal       253367455              430583644
47006948
5         Walt Disney      6071822640             6777527993
739904802
6        Warner Bros.      1396448343             1991232438
217383454
```

```
# Total Ticket Sold vs Distributor

#colors
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99', '#9F25FA',
'#FA2525', '#25FA35']

plt.pie(group_by_distributor['TICKETS SOLD'], colors = colors,
labels=group_by_distributor['DISTRIBUTOR'], autopct='%1.1f%%',
startangle=90, pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
# ax1.axis('equal')
plt.title('TICKETS SOLD VS DISTRIBUTOR')
plt.tight_layout()
plt.show()
```




```
# Total For Year vs Distributor

#colors
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99', '#9F25FA',
'#FA2525', '#25FA35']

plt.pie(group_by_distributor['TOTAL FOR YEAR'], colors = colors,
labels=group_by_distributor['DISTRIBUTOR'], autopct='%1.1f%%',
startangle=90, pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
# ax1.axis('equal')
plt.tight_layout()
plt.title('TOTAL FOR YEAR VS DISTRIBUTOR')
plt.show()
```



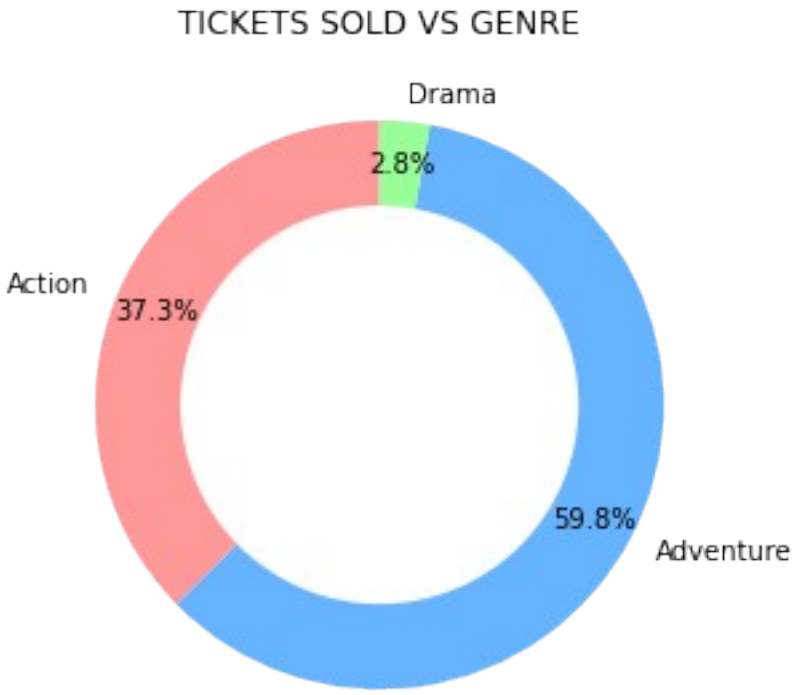
Analyze for Genre

```
group_by_genre = HighestGrossers.groupby(['GENRE'],
as_index=False).sum()
group_by_genre=group_by_genre.drop(labels='YEAR', axis=1)
group_by_genre
```

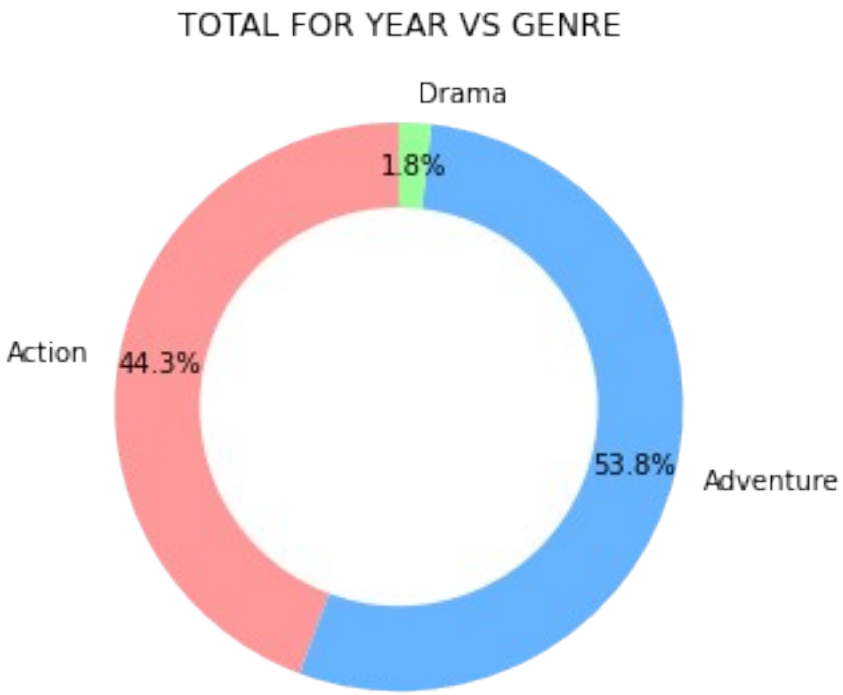
	GENRE	TOTAL FOR YEAR	TOTAL IN 2019 DOLLARS	TICKETS SOLD
0	Action	4447496795	5101959828	556982514
1	Adventure	5401937937	8173858097	892342587
2	Drama	184031112	387522978	42306002

```
# TICKETS SOLD VS GENRE

#colors
colors = ['#ff9999','#66b3ff','#99ff99']

plt.pie(group_by_genre['TICKETS SOLD'], colors = colors,
labels=group_by_genre['GENRE'], autopct='%1.1f%%', startangle=90,
pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
# ax1.axis('equal')
```

```
plt.tight_layout()
plt.title('TICKETS SOLD VS GENRE')
plt.show()
```




```
# TOTAL FOR YEAR VS GENRE

#colors
colors = ['#ff9999','#66b3ff','#99ff99']

plt.pie(group_by_genre['TOTAL FOR YEAR'], colors = colors,
labels=group_by_genre['GENRE'], autopct='%1.1f%%', startangle=90,
pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
# ax1.axis('equal')
plt.tight_layout()
plt.title('TOTAL FOR YEAR VS GENRE')
plt.show()
```



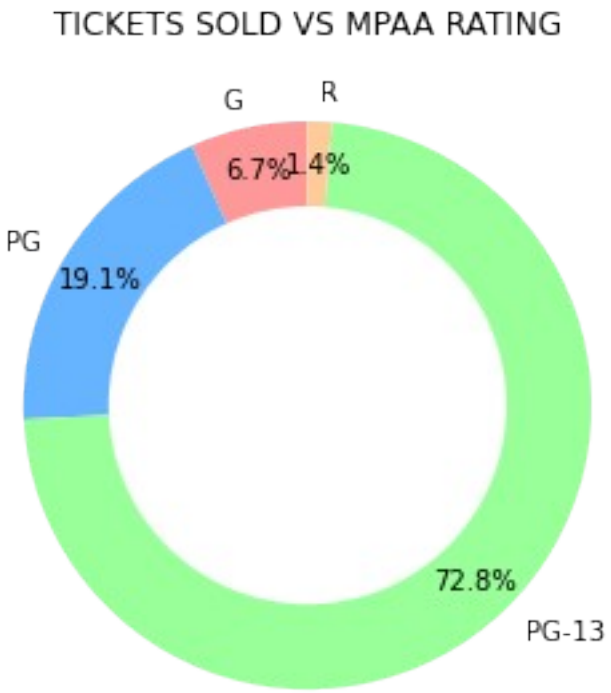
Analyze for MPAA rating

```
group_by_mpaa = HighestGrossers.groupby(['MPAA RATING'],
as_index=False).sum()
group_by_mpaa=group_by_mpaa.drop(labels='YEAR', axis=1)
group_by_mpaa
```

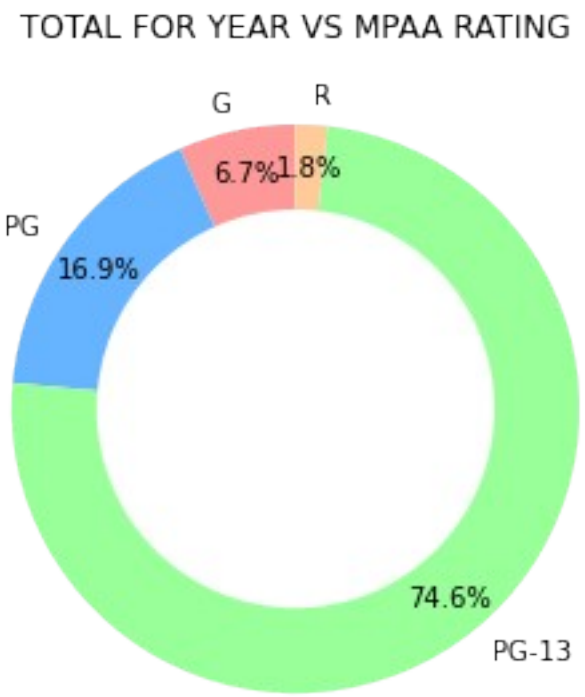
	MPAA RATING	TOTAL FOR YEAR	TOTAL IN 2019 DOLLARS	TICKETS SOLD
0	G	754719247	997855757	108936218
1	PG	1911737047	2858698078	312084943
2	PG-13	8449609254	10889386774	1188797682
3	R	204417855	204417848	22316359

```
# TICKETS SOLD VS MPAA RATING

#colors
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']

plt.pie(group_by_mpaa['TICKETS SOLD'], colors = colors,
labels=group_by_mpaa['MPAA RATING'], autopct='%1.1f%%', startangle=90,
pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
```

```
# ax1.axis('equal')
plt.tight_layout()
plt.title('TICKETS SOLD VS MPAA RATING')
plt.show()
```




```
# TOTAL FOR YEAR VS MPAA RATING

#colors
colors = ['#ff9999','#66b3ff','#99ff99', '#ffcc99']

plt.pie(group_by_mpaa['TOTAL FOR YEAR'], colors = colors,
labels=group_by_mpaa['MPAA RATING'], autopct='%1.1f%%', startangle=90,
pctdistance=0.85)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
# ax1.axis('equal')
plt.tight_layout()
plt.title('TOTAL FOR YEAR VS MPAA RATING')
plt.show()
```



HighestGrossers

	YEAR	MOVIE	GENRE
0	1995	Batman Forever	Drama
1	1996	Independence Day	Adventure
2	1997	Men in Black	Adventure
3	1998	Titanic	Adventure
4	1999	Star Wars Ep. I: The Phantom Menace	Adventure
5	2000	How the Grinch Stole Christmas	Adventure
6	2001	Harry Potter and the Sorcerer’s Stone	Adventure
7	2002	Spider-Man	Adventure
8	2003	Finding Nemo	Adventure
9	2004	Shrek 2	Adventure
10	2005	Star Wars Ep. III: Revenge of the Sith	Action
11	2006	Pirates of the Caribbean: Dead Man’s Chest	Action
12	2007	Spider-Man 3	Adventure
13	2008	The Dark Knight	Adventure
14	2009	Transformers: Revenge of the Fallen	Action
15	2010	Toy Story 3	Action
16	2011	Harry Potter and the Deathly Hallows: Part II	Action
17	2012	The Avengers	Adventure
18	2013	Iron Man 3	Adventure
19	2014	Guardians of the Galaxy	Adventure
20	2015	Star Wars Ep. VII: The Force Awakens	Action
21	2016	Finding Dory	Action
22	2017	Star Wars Ep. VIII: The Last Jedi	Action
23	2018	Black Panther	Action

```
24  2019                                Avengers: Endgame        NaN
25  2020                                Bad Boys For Life        NaN
26  2021      Shang-Chi and the Legend of the Ten Rings        NaN

   MPAA RATING         DISTRIBUTOR  TOTAL FOR YEAR  TOTAL IN 2019
DOLLARS  \
0        PG-13        Warner Bros.       184031112
387522978
1        PG-13    20th Century Fox       306169255
634504608
2        PG-13       Sony Pictures       250650052
500207943
3        PG-13  Paramount Pictures       443319081
865842808
4           PG    20th Century Fox       430443350
776153749
5           PG           Universal       253367455
430583644
6           PG        Warner Bros.       300404434
486166890
7        PG-13       Sony Pictures       403706375
636480273
8            G         Walt Disney       339714367
516050346
9           PG      Dreamworks SKG       441226247
650826473
10       PG-13    20th Century Fox       380270577
543413171
11       PG-13         Walt Disney       423315812
591995851
12       PG-13       Sony Pictures       336530303
448054878
13       PG-13        Warner Bros.       531001578
677433772
14       PG-13  Paramount Pictures       402111870
491112631
15           G         Walt Disney       415004880
481805411
16       PG-13        Warner Bros.       381011219
440108798
17       PG-13         Walt Disney       623357910
717331462
18       PG-13         Walt Disney       408992272
460808016
19       PG-13         Walt Disney       333055258
373413235
20       PG-13         Walt Disney       742208942
806480887
21          PG         Walt Disney       486295561
514967322
```

```
22       PG-13         Walt Disney       517218368
528173936
23       PG-13         Walt Disney       700059566
703901821
24       PG-13         Walt Disney       858373000
858373002
25           R       Sony Pictures       204417855
204417848
26       PG-13         Walt Disney       224226704
224226704

    TICKETS SOLD
0       42306002
1       69269062
2       54607854
3       94524324
4       84732942
5       47006948
6       53074988
7       69484746
8       56337374
9       71050925
10      59324582
11      64628368
12      48914288
13      73955652
14      53614916
15      52598844
16      48046812
17      78311295
18      50306552
19      40765637
20      88043765
21      56219140
22      57660910
23      76845177
24      93708843
25      22316359
26      24478897
```

Top 10 and Least 10 movies based on Tickets Sold

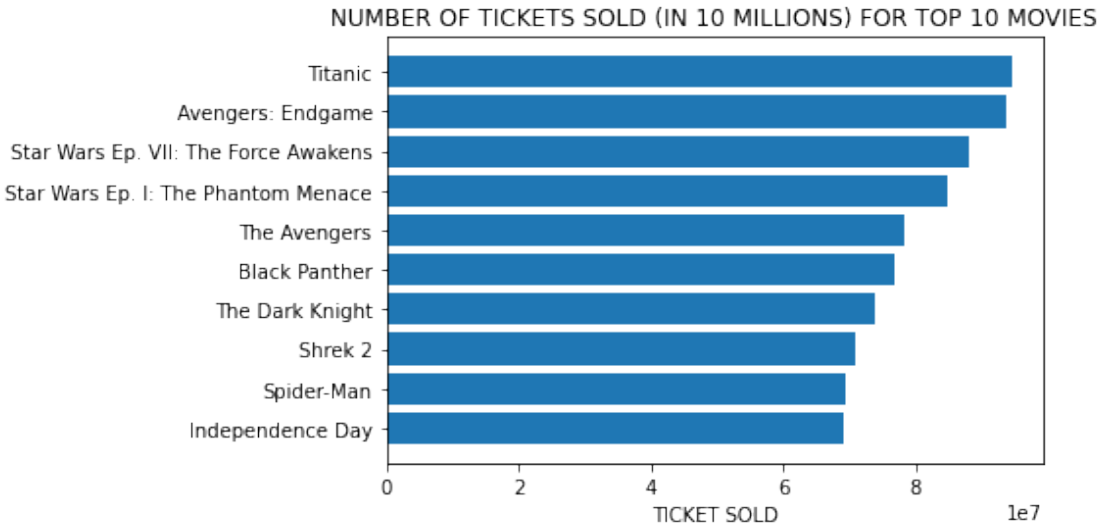
```
top_10_movies = HighestGrossers.nlargest(n=10, columns=['TICKETS
SOLD'])
top_10_movies
```

```
    YEAR                                 MOVIE      GENRE MPAA RATING
\
3   1998                               Titanic  Adventure       PG-13
```

```
24  2019                     Avengers: Endgame        NaN       PG-13

20  2015  Star Wars Ep. VII: The Force Awakens     Action       PG-13

4   1999   Star Wars Ep. I: The Phantom Menace  Adventure          PG

17  2012                          The Avengers  Adventure       PG-13

23  2018                         Black Panther     Action       PG-13

13  2008                       The Dark Knight  Adventure       PG-13

9   2004                              Shrek 2  Adventure          PG

7   2002                           Spider-Man  Adventure       PG-13

1   1996                       Independence Day  Adventure       PG-13


           DISTRIBUTOR  TOTAL FOR YEAR  TOTAL IN 2019 DOLLARS  TICKETS
SOLD
3   Paramount Pictures       443319081              865842808
94524324
24         Walt Disney       858373000              858373002
93708843
20         Walt Disney       742208942              806480887
88043765
4     20th Century Fox       430443350              776153749
84732942
17         Walt Disney       623357910              717331462
78311295
23         Walt Disney       700059566              703901821
76845177
13        Warner Bros.       531001578              677433772
73955652
9       Dreamworks SKG       441226247              650826473
71050925
7         Sony Pictures       403706375              636480273
69484746
1     20th Century Fox       306169255              634504608
69269062
```

```
# bar plot
fig, ax = plt.subplots()
ax.barh(top_10_movies['MOVIE'], top_10_movies['TICKETS SOLD'],
align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('TICKET SOLD')
ax.set_title('NUMBER OF TICKETS SOLD (IN 10 MILLIONS) FOR TOP 10
```

```
MOVIES')

plt.show()
```



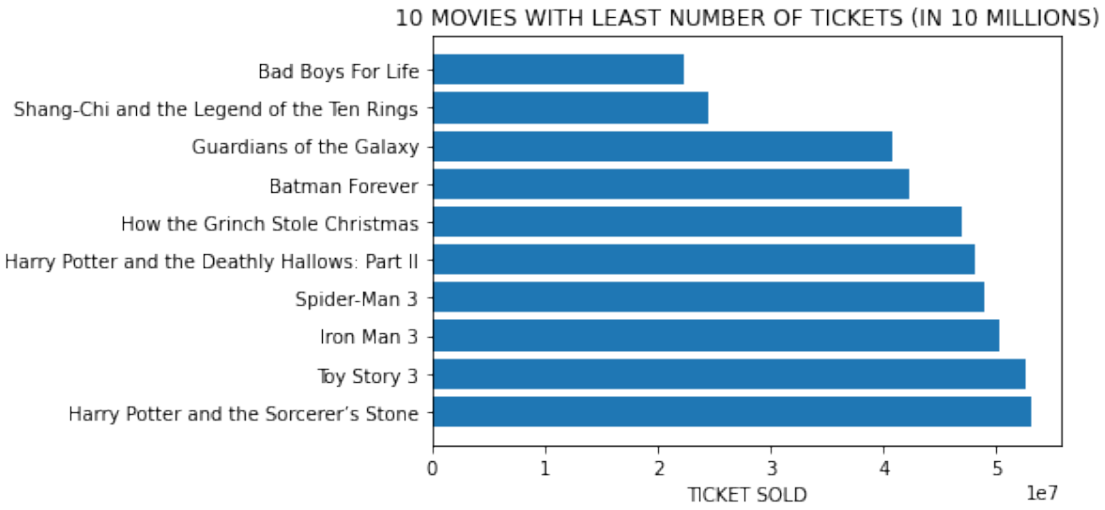

```
least_10_movies = HighestGrossers.nsmallest(n=10, columns=['TICKETS
SOLD'])
least_10_movies
```

```
    YEAR                                              MOVIE      GENRE  \
25  2020                                Bad Boys For Life        NaN
26  2021      Shang-Chi and the Legend of the Ten Rings        NaN
19  2014                          Guardians of the Galaxy  Adventure
0   1995                                   Batman Forever      Drama
5   2000                  How the Grinch Stole Christmas  Adventure
16  2011  Harry Potter and the Deathly Hallows: Part II     Action
12  2007                                     Spider-Man 3  Adventure
18  2013                                       Iron Man 3  Adventure
15  2010                                      Toy Story 3     Action
6   2001          Harry Potter and the Sorcerer's Stone  Adventure

   MPAA RATING    DISTRIBUTOR  TOTAL FOR YEAR  TOTAL IN 2019
DOLLARS  \
25           R  Sony Pictures       204417855              204417848

26       PG-13    Walt Disney       224226704              224226704

19       PG-13    Walt Disney       333055258              373413235

0        PG-13   Warner Bros.       184031112              387522978

5           PG      Universal       253367455              430583644

16       PG-13   Warner Bros.       381011219              440108798
```

```
12       PG-13  Sony Pictures       336530303              448054878

18       PG-13    Walt Disney       408992272              460808016

15           G    Walt Disney       415004880              481805411

6           PG   Warner Bros.       300404434              486166890
```

```
    TICKETS SOLD
25      22316359
26      24478897
19      40765637
0       42306002
5       47006948
16      48046812
12      48914288
18      50306552
15      52598844
6       53074988
```

```
# bar plot
fig, ax = plt.subplots()
ax.barh(least_10_movies['MOVIE'], least_10_movies['TICKETS SOLD'],
align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('TICKET SOLD')
ax.set_title('10 MOVIES WITH LEAST NUMBER OF TICKETS (IN 10
MILLIONS)')

plt.show()
```



Popular Creative Types Analysis

```
PopularCreativeTypes = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/PopularCreativeTypes.csv')
PopularCreativeTypes
```

```
   RANK           CREATIVE TYPES MOVIES      TOTAL GROSS AVERAGE GROSS
\
0   1.0     Contemporary Fiction  7,442  $96,203,727,036   $12,927,133

1   2.0             Kids Fiction    564  $32,035,539,746   $56,800,602

2   3.0          Science Fiction    724  $29,922,660,857   $41,329,642

3   4.0                  Fantasy    759  $21,724,062,575   $28,621,953

4   5.0               Super Hero    129  $20,273,157,911  $157,156,263

5   6.0       Historical Fiction  1,487  $18,521,260,744   $12,455,454

6   7.0            Dramatization  1,175  $15,715,191,699   $13,374,631

7   8.0                  Factual  2,467   $2,960,327,207    $1,199,970

8   9.0  Multiple Creative Types     42     $117,574,526    $2,799,393

9   NaN                      NaN    NaN              NaN           NaN


  MARKET SHARE
0       40.46%
1       13.47%
2       12.59%
3        9.14%
4        8.53%
5        7.79%
6        6.61%
7        1.25%
8        0.05%
9          NaN
```

```
#drop null values
PopularCreativeTypes=PopularCreativeTypes.dropna()
PopularCreativeTypes
```

```
   RANK           CREATIVE TYPES MOVIES      TOTAL GROSS AVERAGE GROSS
\
0   1.0     Contemporary Fiction  7,442  $96,203,727,036   $12,927,133

1   2.0             Kids Fiction    564  $32,035,539,746   $56,800,602
```

```
2   3.0          Science Fiction    724  $29,922,660,857   $41,329,642

3   4.0                  Fantasy    759  $21,724,062,575   $28,621,953

4   5.0               Super Hero    129  $20,273,157,911  $157,156,263

5   6.0       Historical Fiction  1,487  $18,521,260,744   $12,455,454

6   7.0            Dramatization  1,175  $15,715,191,699   $13,374,631

7   8.0                  Factual  2,467   $2,960,327,207    $1,199,970

8   9.0  Multiple Creative Types     42     $117,574,526    $2,799,393
```

```
  MARKET SHARE
0       40.46%
1       13.47%
2       12.59%
3        9.14%
4        8.53%
5        7.79%
6        6.61%
7        1.25%
8        0.05%
```

```
PopularCreativeTypes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9 entries, 0 to 8
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   RANK            9 non-null      float64
 1   CREATIVE TYPES  9 non-null      object
 2   MOVIES          9 non-null      object
 3   TOTAL GROSS     9 non-null      object
 4   AVERAGE GROSS   9 non-null      object
 5   MARKET SHARE    9 non-null      object
dtypes: float64(1), object(5)
memory usage: 504.0+ bytes
```

```
# data cleaning
#Converting MOVIES from object to int
PopularCreativeTypes['MOVIES']=PopularCreativeTypes['MOVIES'].replace(
',','', regex=True)
PopularCreativeTypes['MOVIES']=pd.to_numeric(PopularCreativeTypes['MOV
IES'])
```

```
#Converting TOTAL GROSS from object to float
PopularCreativeTypes['TOTAL GROSS']=PopularCreativeTypes['TOTAL
GROSS'].replace(',','', regex=True)
PopularCreativeTypes['TOTAL GROSS']=PopularCreativeTypes['TOTAL
GROSS'].str.replace('$','', regex=True)
PopularCreativeTypes['TOTAL
GROSS']=pd.to_numeric(PopularCreativeTypes['TOTAL GROSS'])

#Converting AVERAGE GROSS from object to float
PopularCreativeTypes['AVERAGE GROSS']=PopularCreativeTypes['AVERAGE
GROSS'].replace(',','', regex=True)
PopularCreativeTypes['AVERAGE GROSS']=PopularCreativeTypes['AVERAGE
GROSS'].str.replace('$','', regex=True)
PopularCreativeTypes['AVERAGE
GROSS']=pd.to_numeric(PopularCreativeTypes['AVERAGE GROSS'])

PopularCreativeTypes
```

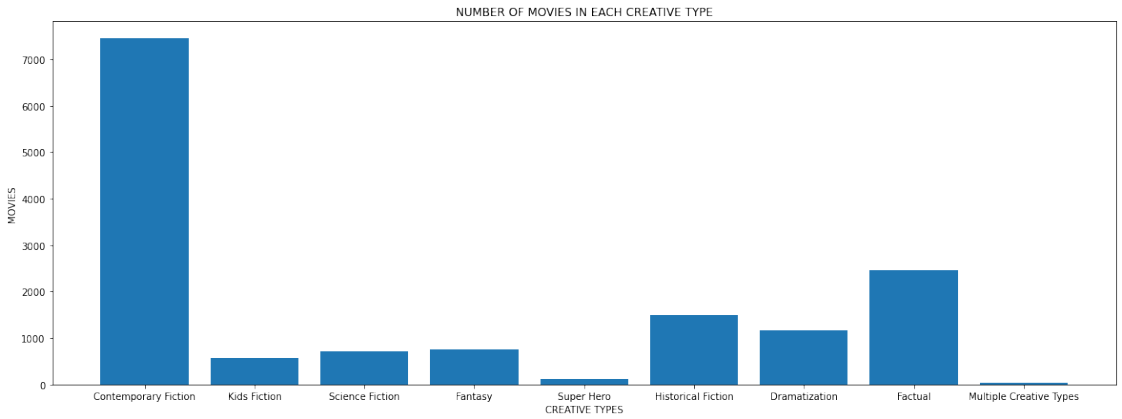
```
C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:3:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:4:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:7:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
```

```
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:8:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:9:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:12:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:13:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

C:\Users\Yusuf\AppData\Local\Temp/ipykernel_4608/3044506155.py:14:
SettingWithCopyWarning:
```

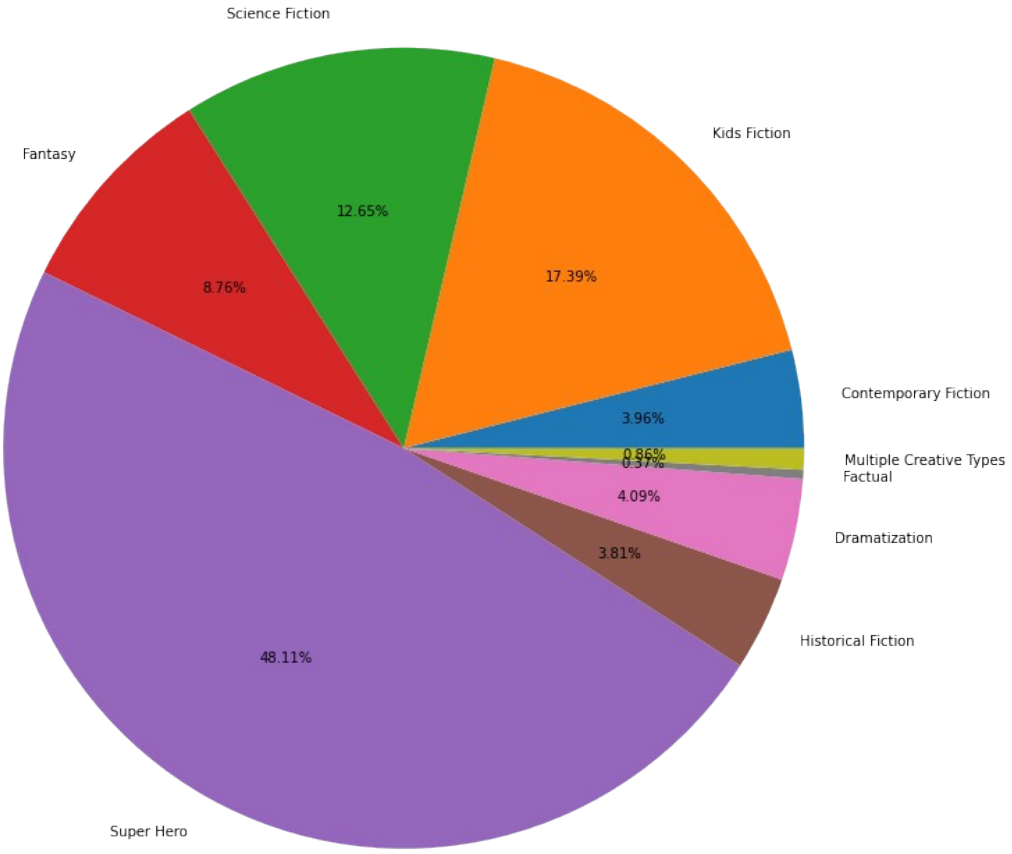
```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy


   RANK           CREATIVE TYPES  MOVIES  TOTAL GROSS  AVERAGE
GROSS  \
0   1.0     Contemporary Fiction    7442  96203727036       12927133

1   2.0              Kids Fiction     564  32035539746       56800602

2   3.0          Science Fiction     724  29922660857       41329642

3   4.0                  Fantasy     759  21724062575       28621953

4   5.0               Super Hero     129  20273157911      157156263

5   6.0       Historical Fiction    1487  18521260744       12455454

6   7.0            Dramatization    1175  15715191699       13374631

7   8.0                  Factual    2467   2960327207        1199970

8   9.0  Multiple Creative Types      42    117574526        2799393


  MARKET SHARE
0       40.46%
1       13.47%
2       12.59%
3        9.14%
4        8.53%
5        7.79%
6        6.61%
7        1.25%
8        0.05%

PopularCreativeTypes.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 9 entries, 0 to 8
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   RANK            9 non-null      float64
 1   CREATIVE TYPES  9 non-null      object
 2   MOVIES          9 non-null      int64
 3   TOTAL GROSS     9 non-null      int64
 4   AVERAGE GROSS   9 non-null      int64
 5   MARKET SHARE    9 non-null      object
dtypes: float64(1), int64(3), object(2)
memory usage: 504.0+ bytes
```

```
# Bar plot creative types and movies in each type
# Total tickets sold every year
figure=plt.figure(figsize=(20,7))
plt.bar(PopularCreativeTypes['CREATIVE TYPES'],
PopularCreativeTypes['MOVIES'])
plt.xlabel("CREATIVE TYPES")
plt.ylabel("MOVIES")
plt.title("NUMBER OF MOVIES IN EACH CREATIVE TYPE")
plt.show()
```




```
# Pie chart of Creative types and Average Gross
fig = plt.figure(figsize=[10,10])
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
ax.pie(PopularCreativeTypes['AVERAGE GROSS'], labels =
PopularCreativeTypes['CREATIVE TYPES'],autopct='%1.2f%%')
plt.show()
```



Top Distributors

```
# importing the dataset
TopDistributors = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/TopDistributors.csv')
TopDistributors
```

	RANK	DISTRIBUTORS	MOVIES	TOTAL GROSS	AVERAGE GROSS	\
0	1	Walt Disney	588	$40,472,424,278	$68,830,654	
1	2	Warner Bros.	824	$36,269,425,479	$44,016,293	
2	3	Sony Pictures	747	$29,113,002,302	$38,973,229	
3	4	Universal	535	$28,089,932,569	$52,504,547	
4	5	20th Century Fox	525	$25,857,839,756	$49,253,028	
5	6	Paramount Pictures	493	$24,361,425,304	$49,414,656	
6	7	Lionsgate	426	$9,631,837,781	$22,609,948	
7	8	New Line	209	$6,195,268,024	$29,642,431	
8	9	Dreamworks SKG	77	$4,278,649,271	$55,566,874	
9	10	Miramax	385	$3,836,019,208	$9,963,686	

```
  MARKET SHARE
0       17.02%
1       15.25%
2       12.24%
3       11.81%
4       10.88%
5       10.25%
6        4.05%
7        2.61%
8        1.80%
9        1.61%
```

```
#data cleaning

#Converting TOTAL GROSS from object to int
TopDistributors['TOTAL GROSS']=TopDistributors['TOTAL
GROSS'].replace(',','', regex=True)
TopDistributors['TOTAL GROSS']=TopDistributors['TOTAL
GROSS'].str.replace('$','', regex=True)
TopDistributors['TOTAL GROSS']=pd.to_numeric(TopDistributors['TOTAL
GROSS'])

#Converting TOTAL INFLATION ADJUSTED BOX OFFICE from object to int
TopDistributors['AVERAGE GROSS']=TopDistributors['AVERAGE
GROSS'].replace(',','', regex=True)
TopDistributors['AVERAGE GROSS']=TopDistributors['AVERAGE
GROSS'].str.replace('$','', regex=True)
TopDistributors['AVERAGE
GROSS']=pd.to_numeric(TopDistributors['AVERAGE GROSS'])

TopDistributors
```

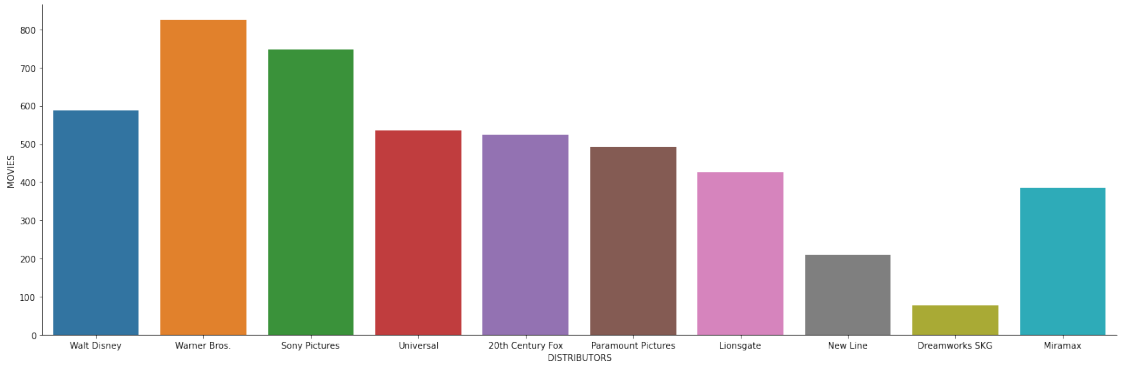
```
   RANK        DISTRIBUTORS  MOVIES  TOTAL GROSS  AVERAGE GROSS MARKET
SHARE
0     1         Walt Disney     588  40472424278       68830654
17.02%
1     2        Warner Bros.     824  36269425479       44016293
15.25%
2     3       Sony Pictures     747  29113002302       38973229
12.24%
3     4           Universal     535  28089932569       52504547
11.81%
4     5    20th Century Fox     525  25857839756       49253028
10.88%
5     6  Paramount Pictures     493  24361425304       49414656
10.25%
6     7           Lionsgate     426   9631837781       22609948
4.05%
7     8            New Line     209   6195268024       29642431
2.61%
```

```
8     9      Dreamworks SKG      77   4278649271       55566874
1.80%
9    10             Miramax     385   3836019208        9963686
1.61%
```

```
#Distributors vs Number of movies they released
fig=plt.figure(figsize=(5,10))
ax = sns.catplot(y='MOVIES', x='DISTRIBUTORS',kind='bar',
data=TopDistributors, height=6, aspect=3)
plt.ylabel('MOVIES')
plt.xlabel('DISTRIBUTORS')
```

```
Text(0.5, 6.800000000000011, 'DISTRIBUTORS')
```

```
<Figure size 360x720 with 0 Axes>
```



Top Genres analysis

```
TopGenres = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood Theatrical
Market Synopsis 1995 to 2021/TopGenres.csv')
TopGenres
```

```
   RANK             GENRES MOVIES      TOTAL GROSS AVERAGE GROSS
MARKET SHARE
0     1          Adventure  1,102  $64,529,536,530   $58,556,748
27.14%
1     2             Action  1,098  $49,339,974,493   $44,936,224
20.75%
2     3              Drama  5,479  $35,586,177,269    $6,495,013
14.97%
3     4             Comedy  2,418  $33,687,992,318   $13,932,172
14.17%
4     5  Thriller/Suspense  1,186  $19,810,201,102   $16,703,374
8.33%
5     6             Horror    716  $13,430,378,699   $18,757,512
5.65%
6     7    Romantic Comedy    630  $10,480,124,374   $16,635,118
4.41%
7     8            Musical    201   $4,293,988,317   $21,363,126
```

```
1.81%
8     9        Documentary  2,415   $2,519,513,142    $1,043,277
1.06%
9    10       Black Comedy    213   $2,185,433,323   $10,260,250
0.92%
```

```
TopGenres.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   RANK            10 non-null     int64
 1   GENRES          10 non-null     object
 2   MOVIES          10 non-null     object
 3   TOTAL GROSS     10 non-null     object
 4   AVERAGE GROSS   10 non-null     object
 5   MARKET SHARE    10 non-null     object
dtypes: int64(1), object(5)
memory usage: 608.0+ bytes
```

```
#data cleaning
#Converting TOTAL GROSS from object to int
TopGenres['TOTAL GROSS']=TopGenres['TOTAL GROSS'].replace(',','',
regex=True)
TopGenres['TOTAL GROSS']=TopGenres['TOTAL GROSS'].str.replace('$','',
regex=True)
TopGenres['TOTAL GROSS']=pd.to_numeric(TopGenres['TOTAL GROSS'])

#Converting AVERAGE GROSS from object to int
TopGenres['AVERAGE GROSS']=TopGenres['AVERAGE GROSS'].replace(',','',
regex=True)
TopGenres['AVERAGE GROSS']=TopGenres['AVERAGE
GROSS'].str.replace('$','', regex=True)
TopGenres['AVERAGE GROSS']=pd.to_numeric(TopGenres['AVERAGE GROSS'])

#Converting AVERAGE GROSS from object to int
TopGenres['MOVIES']=TopGenres['MOVIES'].replace(',','', regex=True)
TopGenres['MOVIES']=TopGenres['MOVIES'].str.replace('$','',
regex=True)
TopGenres['MOVIES']=pd.to_numeric(TopGenres['MOVIES'])
```

```
TopGenres
```

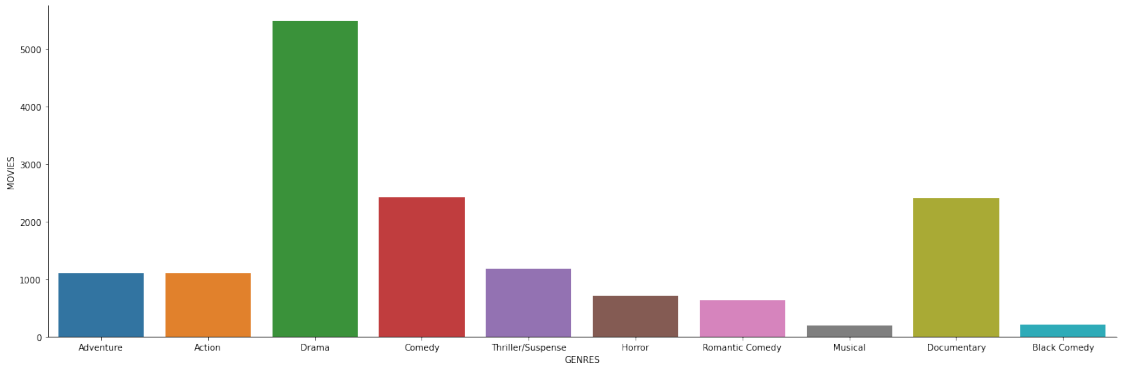
```
   RANK             GENRES  MOVIES  TOTAL GROSS  AVERAGE GROSS MARKET
SHARE
0     1          Adventure    1102  64529536530       58556748
27.14%
1     2             Action    1098  49339974493       44936224
20.75%
```

```
2     3              Drama     5479  35586177269        6495013
14.97%
3     4             Comedy     2418  33687992318       13932172
14.17%
4     5  Thriller/Suspense     1186  19810201102       16703374
8.33%
5     6             Horror      716  13430378699       18757512
5.65%
6     7    Romantic Comedy      630  10480124374       16635118
4.41%
7     8            Musical      201   4293988317       21363126
1.81%
8     9        Documentary     2415   2519513142        1043277
1.06%
9    10       Black Comedy      213   2185433323       10260250
0.92%
```

```
#Genres vs Number of movies released
fig=plt.figure(figsize=(5,10))
ax = sns.catplot(y='MOVIES', x='GENRES',kind='bar', data=TopGenres,
height=6, aspect=3)
plt.ylabel('MOVIES')
plt.xlabel('GENRES')
```

```
Text(0.5, 6.800000000000011, 'GENRES')
```

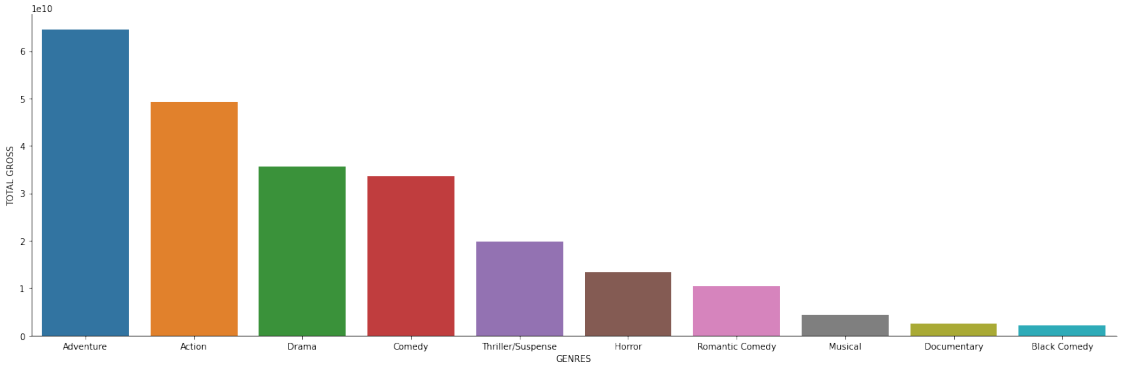
```
<Figure size 360x720 with 0 Axes>
```




```
#Genres vs Total Gross
fig=plt.figure(figsize=(5,10))
ax = sns.catplot(y='TOTAL GROSS', x='GENRES',kind='bar',
data=TopGenres, height=6, aspect=3)
plt.ylabel('TOTAL GROSS')
plt.xlabel('GENRES')
```

```
Text(0.5, 6.800000000000011, 'GENRES')
```

```
<Figure size 360x720 with 0 Axes>
```



Top Grossing Ratings Analysis

```
TopGrossingRatings = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/TopGrossingRatings.csv')
TopGrossingRatings
```

```
   RANK MPAA RATINGS MOVIES       TOTAL GROSS AVERAGE GROSS MARKET
SHARE
0     1        PG-13  3,243  $113,524,789,243   $35,006,102
47.75%
1     2            R  5,480   $63,497,164,978   $11,587,074
26.71%
2     3           PG  1,535   $49,124,317,794   $32,002,813
20.66%
3     4            G    395    $9,572,240,391   $24,233,520
4.03%
4     5    Not Rated  5,820    $1,918,358,283      $329,615
0.81%
5     6        NC-17     24       $44,850,139    $1,868,756
0.02%
6     7         Open      5        $5,489,687    $1,097,937
0.00%
7     8           GP      7          $552,618       $78,945
0.00%
```

```
TopGrossingRatings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   RANK            8 non-null      int64
 1   MPAA RATINGS    8 non-null      object
 2   MOVIES          8 non-null      object
 3   TOTAL GROSS     8 non-null      object
 4   AVERAGE GROSS   8 non-null      object
 5   MARKET SHARE    8 non-null      object
```

```
dtypes: int64(1), object(5)
memory usage: 512.0+ bytes
```

```
# data cleaning
#Converting TOTAL GROSS from object to int
TopGrossingRatings['TOTAL GROSS']=TopGrossingRatings['TOTAL
GROSS'].replace(',','', regex=True)
TopGrossingRatings['TOTAL GROSS']=TopGrossingRatings['TOTAL
GROSS'].str.replace('$','', regex=True)
TopGrossingRatings['TOTAL
GROSS']=pd.to_numeric(TopGrossingRatings['TOTAL GROSS'])

#Converting AVERAGE GROSS from object to int
TopGrossingRatings['AVERAGE GROSS']=TopGrossingRatings['AVERAGE
GROSS'].replace(',','', regex=True)
TopGrossingRatings['AVERAGE GROSS']=TopGrossingRatings['AVERAGE
GROSS'].str.replace('$','', regex=True)
TopGrossingRatings['AVERAGE
GROSS']=pd.to_numeric(TopGrossingRatings['AVERAGE GROSS'])

#Converting AVERAGE GROSS from object to int
TopGrossingRatings['MOVIES']=TopGrossingRatings['MOVIES'].replace(',',
'', regex=True)
TopGrossingRatings['MOVIES']=TopGrossingRatings['MOVIES'].str.replace(
'$','', regex=True)
TopGrossingRatings['MOVIES']=pd.to_numeric(TopGrossingRatings['MOVIES'
])
```

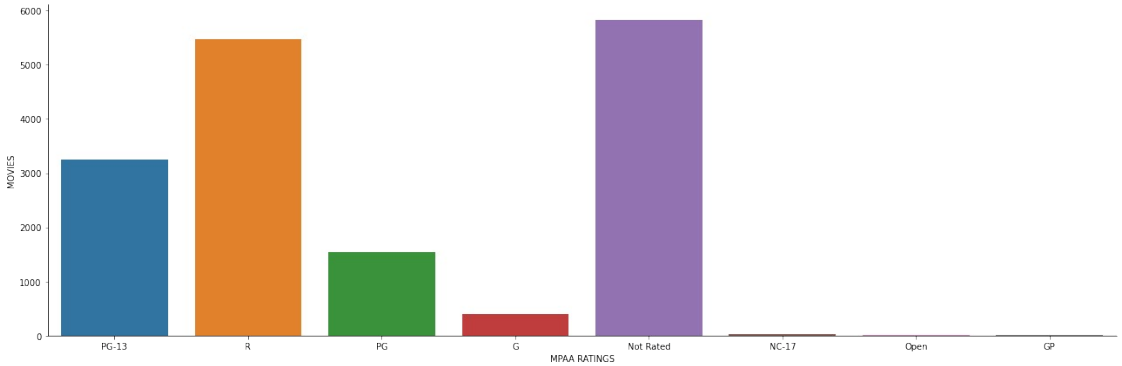
```
TopGrossingRatings
```

	RANK	MPAA RATINGS	MOVIES	TOTAL GROSS	AVERAGE GROSS	MARKET SHARE
0	1	PG-13	3243	113524789243	35006102	47.75%
1	2	R	5480	63497164978	11587074	26.71%
2	3	PG	1535	49124317794	32002813	20.66%
3	4	G	395	9572240391	24233520	4.03%
4	5	Not Rated	5820	1918358283	329615	0.81%
5	6	NC-17	24	44850139	1868756	0.02%
6	7	Open	5	5489687	1097937	0.00%
7	8	GP	7	552618	78945	0.00%

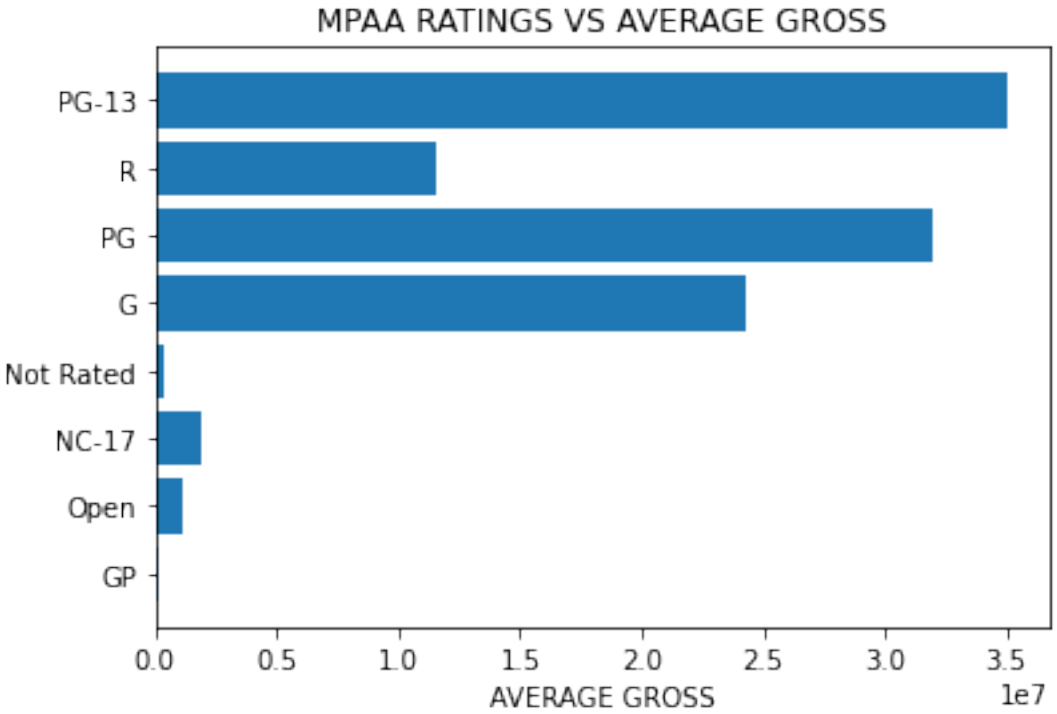
```
#Genres vs Number of movies released
fig=plt.figure(figsize=(5,10))
ax = sns.catplot(y='MOVIES', x='MPAA RATINGS',kind='bar',
data=TopGrossingRatings, height=6, aspect=3)
plt.ylabel('MOVIES')
plt.xlabel('MPAA RATINGS')
```

```
Text(0.5, 6.800000000000011, 'MPAA RATINGS')
```

```
<Figure size 360x720 with 0 Axes>
```




```
# Bar plot for MPAA Rating and Average Gross
fig, ax = plt.subplots()
ax.barh(TopGrossingRatings['MPAA RATINGS'],
TopGrossingRatings['AVERAGE GROSS'], align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('AVERAGE GROSS')
ax.set_title('MPAA RATINGS VS AVERAGE GROSS')
plt.show()
```



Top Grossing Sources Analysis

```
TopGrossingSources = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/TopGrossingSources.csv')
TopGrossingSources
```

```
   RANK                              SOURCES MOVIES       TOTAL GROSS
\
0     1                   Original Screenplay  7,946  $106,375,196,782

1     2    Based on Fiction Book/Short Story  2,150   $47,005,613,207

2     3         Based on Comic/Graphic Novel    249   $23,369,989,130

3     4                               Remake    328   $12,832,659,970

4     5            Based on Real Life Events  3,225   $11,398,356,297

5     6                          Based on TV    231   $11,305,006,312

6     7        Based on Factual Book/Article    295    $7,443,681,990

7     8                             Spin-Off     41    $3,833,128,331

8     9  Based on Folk Tale/Legend/Fairytale     78    $3,406,118,495

9    10                        Based on Play    271    $2,111,190,923


  AVERAGE GROSS MARKET SHARE
0   $13,387,264       44.74%
1   $21,863,076       19.77%
2   $93,855,378        9.83%
3   $39,123,963        5.40%
4    $3,534,374        4.79%
5   $48,939,421        4.75%
6   $25,232,820        3.13%
7   $93,490,935        1.61%
8   $43,668,186        1.43%
9    $7,790,372        0.89%

TopGrossingRatings.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   RANK           8 non-null      int64
 1   MPAA RATINGS   8 non-null      object
 2   MOVIES         8 non-null      int64
 3   TOTAL GROSS    8 non-null      int64
 4   AVERAGE GROSS  8 non-null      int64
 5   MARKET SHARE   8 non-null      object
```

```
dtypes: int64(4), object(2)
memory usage: 512.0+ bytes
```

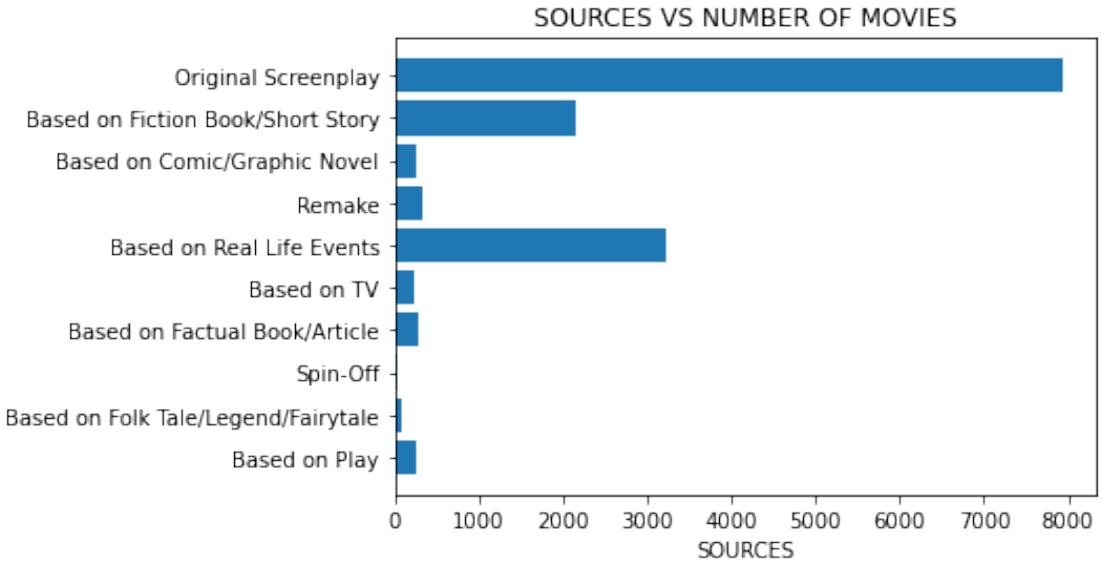
```
# data cleaning
#Converting TOTAL GROSS from object to int
TopGrossingSources['TOTAL GROSS']=TopGrossingSources['TOTAL
GROSS'].replace(',','', regex=True)
TopGrossingSources['TOTAL GROSS']=TopGrossingSources['TOTAL
GROSS'].str.replace('$','', regex=True)
TopGrossingSources['TOTAL
GROSS']=pd.to_numeric(TopGrossingSources['TOTAL GROSS'])

#Converting AVERAGE GROSS from object to int
TopGrossingSources['AVERAGE GROSS']=TopGrossingSources['AVERAGE
GROSS'].replace(',','', regex=True)
TopGrossingSources['AVERAGE GROSS']=TopGrossingSources['AVERAGE
GROSS'].str.replace('$','', regex=True)
TopGrossingSources['AVERAGE
GROSS']=pd.to_numeric(TopGrossingSources['AVERAGE GROSS'])

#Converting AVERAGE GROSS from object to int
TopGrossingSources['MOVIES']=TopGrossingSources['MOVIES'].replace(',',
'', regex=True)
TopGrossingSources['MOVIES']=TopGrossingSources['MOVIES'].str.replace(
'$','', regex=True)
TopGrossingSources['MOVIES']=pd.to_numeric(TopGrossingSources['MOVIES'
])
```

```
TopGrossingSources
```

```
   RANK                              SOURCES  MOVIES   TOTAL GROSS  \
0     1                   Original Screenplay    7946  106375196782
1     2    Based on Fiction Book/Short Story    2150   47005613207
2     3         Based on Comic/Graphic Novel     249   23369989130
3     4                               Remake     328   12832659970
4     5            Based on Real Life Events    3225   11398356297
5     6                              Based on TV     231   11305006312
6     7        Based on Factual Book/Article     295    7443681990
7     8                             Spin-Off      41    3833128331
8     9  Based on Folk Tale/Legend/Fairytale      78    3406118495
9    10                          Based on Play     271    2111190923

   AVERAGE GROSS MARKET SHARE
0       13387264       44.74%
1       21863076       19.77%
2       93855378        9.83%
3       39123963        5.40%
4        3534374        4.79%
5       48939421        4.75%
6       25232820        3.13%
```

```
7       93490935        1.61%
8       43668186        1.43%
9        7790372        0.89%
```

```
# Bar plot for Sources and Number of Movies
fig, ax = plt.subplots()
ax.barh(TopGrossingSources['SOURCES'], TopGrossingSources['MOVIES'],
align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('SOURCES')
ax.set_title('SOURCES VS NUMBER OF MOVIES')
plt.show()
```



Top Production Methods Analysis

```
 TopProductionMethods = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/TopProductionMethods.csv')
 TopProductionMethods
```

```
   RANK           PRODUCTION METHODS  MOVIES       TOTAL GROSS AVERAGE
GROSS  \
0     1                  Live Action  14,613  $179,637,201,848
$12,292,972
1     2        Animation/Live Action     264   $30,346,622,254
$114,949,327
2     3            Digital Animation     365   $23,920,180,508
$65,534,741
3     4               Hand Animation     164    $2,960,497,487
$18,051,814
4     5        Stop-Motion Animation      37      $676,490,120
$18,283,517
5     6  Multiple Production Methods      26       $43,728,300
```

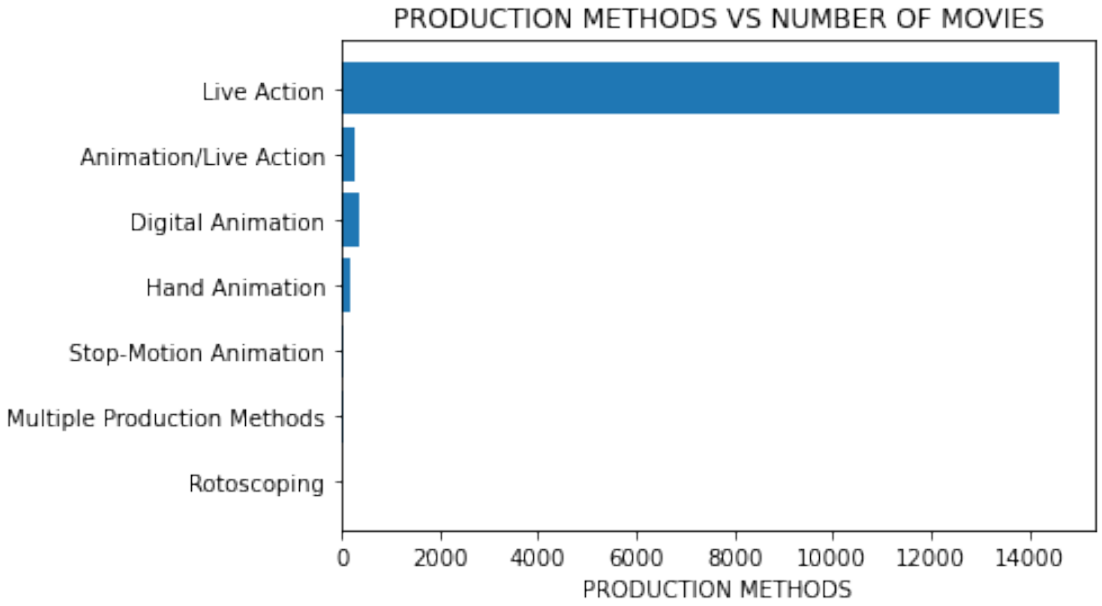
```
$1,681,858
6     7                  Rotoscoping       4        $8,468,385
$2,117,096

  MARKET SHARE
0       75.56%
1       12.76%
2       10.06%
3        1.25%
4        0.28%
5        0.02%
6        0.00%
```

```
# data cleaning
#Converting TOTAL GROSS from object to int
TopProductionMethods['TOTAL GROSS']=TopProductionMethods['TOTAL
GROSS'].replace(',','', regex=True)
TopProductionMethods['TOTAL GROSS']=TopProductionMethods['TOTAL
GROSS'].str.replace('$','', regex=True)
TopProductionMethods['TOTAL
GROSS']=pd.to_numeric(TopProductionMethods['TOTAL GROSS'])

#Converting AVERAGE GROSS from object to int
TopProductionMethods['AVERAGE GROSS']=TopProductionMethods['AVERAGE
GROSS'].replace(',','', regex=True)
TopProductionMethods['AVERAGE GROSS']=TopProductionMethods['AVERAGE
GROSS'].str.replace('$','', regex=True)
TopProductionMethods['AVERAGE
GROSS']=pd.to_numeric(TopProductionMethods['AVERAGE GROSS'])

#Converting AVERAGE GROSS from object to int
TopProductionMethods['MOVIES']=TopProductionMethods['MOVIES'].replace(
',','', regex=True)
TopProductionMethods['MOVIES']=TopProductionMethods['MOVIES'].str.repl
ace('$','', regex=True)
TopProductionMethods['MOVIES']=pd.to_numeric(TopProductionMethods['MOV
IES'])

TopProductionMethods
```

```
   RANK           PRODUCTION METHODS  MOVIES   TOTAL GROSS  AVERAGE
GROSS  \
0     1                  Live Action   14613  179637201848
12292972
1     2        Animation/Live Action     264   30346622254
114949327
2     3            Digital Animation     365   23920180508
65534741
3     4               Hand Animation     164    2960497487
18051814
```

```
4     5        Stop-Motion Animation      37     676490120
18283517
5     6  Multiple Production Methods      26      43728300
1681858
6     7                   Rotoscoping       4       8468385
2117096

  MARKET SHARE
0       75.56%
1       12.76%
2       10.06%
3        1.25%
4        0.28%
5        0.02%
6        0.00%
```

```
# Bar plot for Production Methods and Number of Movies
fig, ax = plt.subplots()
ax.barh(TopProductionMethods['PRODUCTION METHODS'],
TopProductionMethods['MOVIES'], align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('PRODUCTION METHODS')
ax.set_title('PRODUCTION METHODS VS NUMBER OF MOVIES')
plt.show()
```



Wide Releases Count Analysis

```
WideReleasesCount = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/WideReleasesCount.csv')
WideReleasesCount
```

	YEAR	WARNER BROS	WALT DISNEY	20TH CENTURY FOX	PARAMOUNT PICTURES
0	2021	17	7	0	4
1	2020	5	3	1	3
2	2019	18	10	11	9
3	2018	19	10	11	10
4	2017	18	8	14	10
5	2016	17	12	16	12
6	2015	22	11	18	9
7	2014	17	12	17	10
8	2013	17	8	15	8
9	2012	16	11	15	13
10	2011	20	13	15	13
11	2010	20	12	18	12
12	2009	25	14	20	10
13	2008	19	11	22	14
14	2007	30	13	17	16
15	2006	26	17	25	13
16	2005	20	20	19	12
17	2004	27	25	18	14
18	2003	28	19	13	14
19	2002	32	23	15	16
20	2001	30	16	16	14
21	2000	29	22	13	12
22	1999	27	20	15	13
23	1998	27	21	11	11

```
24  1997           31           22               12
16
25  1996           31           23               13
16
26  1995           27           22               11
12

    SONY PICTURES  UNIVERSAL  TOTAL MAJOR 6  TOTAL OTHER STUDIOS
Unnamed: 9
0               16         17             61                   38
NaN
1                9         13             34                   23
NaN
2               18         21             87                   44
NaN
3               16         20             86                   58
NaN
4               16         15             81                   50
NaN
5               16         22             95                   46
NaN
6               13         20             93                   33
NaN
7               17         15             88                   37
NaN
8               14         16             78                   42
NaN
9               18         17             90                   42
NaN
10              21         19            101                   35
NaN
11              17         17             96                   30
NaN
12              21         21            111                   30
NaN
13              19         19            104                   48
NaN
14              22         20            118                   50
NaN
15              26         21            128                   31
NaN
16              19         17            107                   30
NaN
17              15         14            113                   25
NaN
18              19         13            106                   23
NaN
19              20         13            119                   21
NaN
20              17         10            103                   25
```

```
NaN
21              15          13              104                     27
NaN
22              22          16              113                     19
NaN
23              20          16              106                     20
NaN
24              22          11              114                     22
NaN
25              24          13              120                     22
NaN
26              20          17              109                     27
NaN
```

```
# drop unnecassary column
WideReleasesCount = WideReleasesCount.drop(['Unnamed: 9'], axis=1)
WideReleasesCount
```

```
    YEAR  WARNER BROS  WALT DISNEY  20TH CENTURY FOX  PARAMOUNT
PICTURES  \
0   2021           17             7                  0
4
1   2020            5             3                  1
3
2   2019           18            10                 11
9
3   2018           19            10                 11
10
4   2017           18             8                 14
10
5   2016           17            12                 16
12
6   2015           22            11                 18
9
7   2014           17            12                 17
10
8   2013           17             8                 15
8
9   2012           16            11                 15
13
10  2011           20            13                 15
13
11  2010           20            12                 18
12
12  2009           25            14                 20
10
13  2008           19            11                 22
14
14  2007           30            13                 17
16
15  2006           26            17                 25
```

13
16 2005   20   20   19
12
17 2004   27   25   18
14
18 2003   28   19   13
14
19 2002   32   23   15
16
20 2001   30   16   16
14
21 2000   29   22   13
12
22 1999   27   20   15
13
23 1998   27   21   11
11
24 1997   31   22   12
16
25 1996   31   23   13
16
26 1995   27   22   11
12

	SONY PICTURES	UNIVERSAL	TOTAL MAJOR 6	TOTAL OTHER STUDIOS
0	16	17	61	38
1	9	13	34	23
2	18	21	87	44
3	16	20	86	58
4	16	15	81	50
5	16	22	95	46
6	13	20	93	33
7	17	15	88	37
8	14	16	78	42
9	18	17	90	42
10	21	19	101	35
11	17	17	96	30
12	21	21	111	30
13	19	19	104	48
14	22	20	118	50
15	26	21	128	31
16	19	17	107	30
17	15	14	113	25
18	19	13	106	23
19	20	13	119	21
20	17	10	103	25
21	15	13	104	27
22	22	16	113	19
23	20	16	106	20
24	22	11	114	22

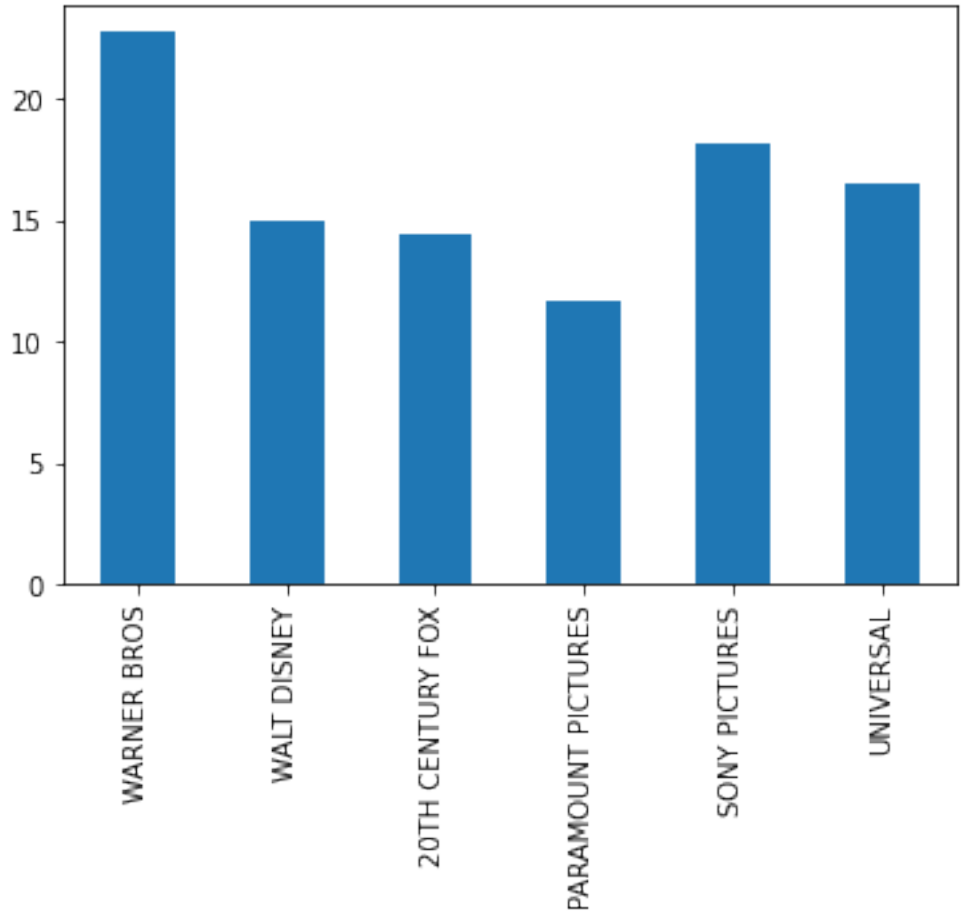
```
25              24          13             120                     22
26              20          17             109                     27

WideReleasesCount.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   YEAR                  27 non-null     int64
 1   WARNER BROS           27 non-null     int64
 2   WALT DISNEY           27 non-null     int64
 3   20TH CENTURY FOX      27 non-null     int64
 4   PARAMOUNT PICTURES    27 non-null     int64
 5   SONY PICTURES         27 non-null     int64
 6   UNIVERSAL             27 non-null     int64
 7   TOTAL MAJOR 6         27 non-null     int64
 8   TOTAL OTHER STUDIOS   27 non-null     int64
dtypes: int64(9)
memory usage: 2.0 KB

#Comparison of Total movies released from 6 major production from 1995
to 2021

# drop  unnecessary columns for now
WideReleasesCount = WideReleasesCount.drop(['TOTAL MAJOR 6',  'TOTAL
OTHER STUDIOS', 'YEAR'], axis=1)
WideReleasesCount

    WARNER BROS  WALT DISNEY  20TH CENTURY FOX  PARAMOUNT PICTURES  \
0            17            7                 0                    4
1             5            3                 1                    3
2            18           10                11                    9
3            19           10                11                   10
4            18            8                14                   10
5            17           12                16                   12
6            22           11                18                    9
7            17           12                17                   10
8            17            8                15                    8
9            16           11                15                   13
10           20           13                15                   13
11           20           12                18                   12
12           25           14                20                   10
13           19           11                22                   14
14           30           13                17                   16
15           26           17                25                   13
16           20           20                19                   12
17           27           25                18                   14
18           28           19                13                   14
19           32           23                15                   16
```

```
20           30           16                16                14
21           29           22                13                12
22           27           20                15                13
23           27           21                11                11
24           31           22                12                16
25           31           23                13                16
26           27           22                11                12

    SONY PICTURES  UNIVERSAL
0              16         17
1               9         13
2              18         21
3              16         20
4              16         15
5              16         22
6              13         20
7              17         15
8              14         16
9              18         17
10             21         19
11             17         17
12             21         21
13             19         19
14             22         20
15             26         21
16             19         17
17             15         14
18             19         13
19             20         13
20             17         10
21             15         13
22             22         16
23             20         16
24             22         11
25             24         13
26             20         17

WideReleasesCount.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   WARNER BROS         27 non-null     int64
 1   WALT DISNEY         27 non-null     int64
 2   20TH CENTURY FOX    27 non-null     int64
 3   PARAMOUNT PICTURES  27 non-null     int64
 4   SONY PICTURES       27 non-null     int64
 5   UNIVERSAL           27 non-null     int64
```

```
dtypes: int64(6)
memory usage: 1.4 KB
```

```
k = WideReleasesCount.sum()/len(WideReleasesCount)
k.plot.bar()
```

```
<AxesSubplot:>
```




```
# Reset our dataframe
WideReleasesCount = pd.read_csv('C:/Users/Yusuf/Desktop/Hollywood
Theatrical Market Synopsis 1995 to 2021/WideReleasesCount.csv')
WideReleasesCount = WideReleasesCount.drop(['Unnamed: 9'], axis=1)
WideReleasesCount
```

```
    YEAR  WARNER BROS  WALT DISNEY  20TH CENTURY FOX  PARAMOUNT
PICTURES  \
0   2021           17            7                 0
4
1   2020            5            3                 1
3
2   2019           18           10                11
9
3   2018           19           10                11
```

10 4	2017	18	8	14	
10 5	2016	17	12	16	
12 6	2015	22	11	18	
9 7	2014	17	12	17	
10 8	2013	17	8	15	
8 9	2012	16	11	15	
13 10	2011	20	13	15	
13 11	2010	20	12	18	
12 12	2009	25	14	20	
10 13	2008	19	11	22	
14 14	2007	30	13	17	
16 15	2006	26	17	25	
13 16	2005	20	20	19	
12 17	2004	27	25	18	
14 18	2003	28	19	13	
14 19	2002	32	23	15	
16 20	2001	30	16	16	
14 21	2000	29	22	13	
12 22	1999	27	20	15	
13 23	1998	27	21	11	
11 24	1997	31	22	12	
16 25	1996	31	23	13	
16 26	1995	27	22	11	
12					

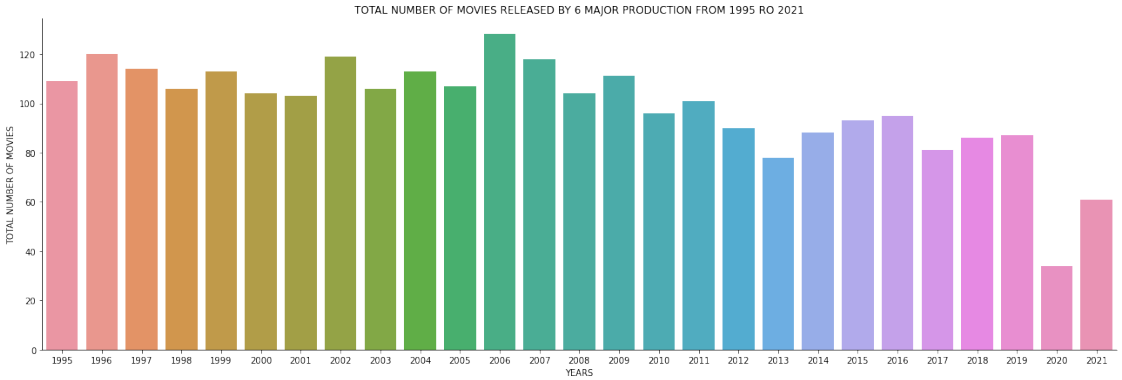
	SONY PICTURES	UNIVERSAL	TOTAL MAJOR 6	TOTAL OTHER STUDIOS
0	16	17	61	38

1	9	13	34	23
2	18	21	87	44
3	16	20	86	58
4	16	15	81	50
5	16	22	95	46
6	13	20	93	33
7	17	15	88	37
8	14	16	78	42
9	18	17	90	42
10	21	19	101	35
11	17	17	96	30
12	21	21	111	30
13	19	19	104	48
14	22	20	118	50
15	26	21	128	31
16	19	17	107	30
17	15	14	113	25
18	19	13	106	23
19	20	13	119	21
20	17	10	103	25
21	15	13	104	27
22	22	16	113	19
23	20	16	106	20
24	22	11	114	22
25	24	13	120	22
26	20	17	109	27

```
#Let see the trendline of total movies released by 6 major production
from 1995 to 2021
fig=plt.figure(figsize=(5,10))
ax = sns.catplot(y='TOTAL MAJOR 6', x='YEAR',kind='bar',
data=WideReleasesCount, height=6, aspect=3)
plt.ylabel('TOTAL NUMBER OF MOVIES')
plt.xlabel('YEARS')
plt.title('TOTAL NUMBER OF MOVIES RELEASED BY 6 MAJOR PRODUCTION FROM
1995 RO 2021')
```

```
Text(0.5, 1.0, 'TOTAL NUMBER OF MOVIES RELEASED BY 6 MAJOR PRODUCTION
FROM 1995 RO 2021')
```

```
<Figure size 360x720 with 0 Axes>
```


TOTAL NUMBER OF MOVIES RELEASED BY 6 MAJOR PRODUCTION FROM 1995 RO 2021
TOTAL NUMBER OF MOVIES
0
20
40
60
80
100
120
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
YEARS