In [436]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [437]:
```
df=pd.read_csv("imdb_top_1000.csv")
```

In [438]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   int64  
 3   Certificate    899 non-null    object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          831 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 125.1+ KB
```

In [439]:
```
df.head()
```

Out[439]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime, Drama

In [440]:
```
df["Genre"]
```

Out[440]:
```
0                       Drama
1                 Crime, Drama
2          Action, Crime, Drama
3                 Crime, Drama
4                 Crime, Drama
                  ...
995      Comedy, Drama, Romance
996              Drama, Western
997         Drama, Romance, War
998                 Drama, War
999    Crime, Mystery, Thriller
Name: Genre, Length: 1000, dtype: object
```

In [441]:
```
genre=df.Genre.str.split(",",expand=True)
```

In [442]:
```
d1=pd.get_dummies(genre[0],drop_first=False)
```

In [443]:
```
d2=pd.get_dummies(genre[1],drop_first=False)
```

In [444]:
```
dfnew=pd.concat([df,d2,d1['Animation']],axis=1)
```

In [445]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 37 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   int64  
 3   Certificate    899 non-null    object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          831 non-null    object 
 16   Action        1000 non-null   uint8  
 17   Adventure     1000 non-null   uint8  
 18   Biography     1000 non-null   uint8  
 19   Comedy        1000 non-null   uint8  
 20   Crime         1000 non-null   uint8  
 21   Drama         1000 non-null   uint8  
 22   Family        1000 non-null   uint8  
 23   Fantasy       1000 non-null   uint8  
 24   Film-Noir     1000 non-null   uint8  
 25   History       1000 non-null   uint8  
 26   Horror        1000 non-null   uint8  
 27   Music         1000 non-null   uint8  
 28   Musical       1000 non-null   uint8  
 29   Mystery       1000 non-null   uint8  
 30   Romance       1000 non-null   uint8  
 31   Sci-Fi        1000 non-null   uint8  
 32   Sport         1000 non-null   uint8  
 33   Thriller      1000 non-null   uint8  
 34   War           1000 non-null   uint8  
 35   Western       1000 non-null   uint8  
 36  Animation      1000 non-null   uint8  
dtypes: float64(2), int64(2), object(12), uint8(21)
memory usage: 145.6+ KB
```

In [446]:
```
dfnew["Runtime"]=dfnew["Runtime"].str.strip(' min').astype(int)
```

In [447]:
```
def comma(col):
    if type(col) is str:
        col=int(col.replace(",",""))
        return col
    else:
        return 56536878
```

In [448]:
```
dfnew["Certificate"].unique()
```

Out[448]:
```
array(['A', 'UA', 'U', 'PG-13', 'R', nan, 'PG', 'G', 'Passed', 'TV-14',
       '16', 'TV-MA', 'Unrated', 'GP', 'Approved', 'TV-PG', 'U/A'],
      dtype=object)
```

In [ ]:
```

```

In [449]:
```
pd.get_dummies(dfnew["Certificate"],drop_first=False)
```

Out[449]:

	16	A	Approved	G	GP	PG	PG-13	Passed	R	TV-14	TV-MA	TV-PG	U	U/A	UA	Unrated
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
996	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1000 rows × 16 columns

In [450]:
```
dfnew["Gross"]=dfnew["Gross"].apply(comma)
```

In [451]:
```
dfnew["Gross"].mean() #please check
```

Out[451]:
```
66091610.358
```

In [ ]:
```

```

In [452]:
```
dfnew["Meta_score"].mean()
```

Out[452]:
```
77.97153024911032
```

In [453]:
```
dfnew["Meta_score"].fillna(78, inplace=True)
```

In [454]:
```
dfnew["Released_Year"].astype(int)
```

Out[454]:
```
0      1994
1      1972
2      2008
3      1974
4      1957
       ...
995    1961
996    1956
997    1953
998    1944
999    1935
Name: Released_Year, Length: 1000, dtype: int32
```

In [455]:
```
duni=dfnew["Director"].unique()
duni
```

```
       'Sidney Lumet', 'Peter Jackson', 'Quentin Tarantino',
       'Steven Spielberg', 'David Fincher', 'Robert Zemeckis',
       'Sergio Leone', 'Lana Wachowski', 'Martin Scorsese',
       'Irvin Kershner', 'Milos Forman', 'Thomas Kail', 'Bong Joon Ho',
       'Sudha Kongara', 'Fernando Meirelles', 'Hayao Miyazaki',
       'Roberto Benigni', 'Jonathan Demme', 'George Lucas',
       'Masaki Kobayashi', 'Akira Kurosawa', 'Frank Capra',
       'Todd Phillips', 'Damien Chazelle', 'Olivier Nakache',
       'Roman Polanski', 'Ridley Scott', 'Tony Kaye', 'Bryan Singer',
       'Luc Besson', 'Roger Allers', 'James Cameron',
       'Giuseppe Tornatore', 'Isao Takahata', 'Alfred Hitchcock',
       'Michael Curtiz', 'Charles Chaplin', 'Nadine Labaki', 'Can Ulkay',
       'Gayatri', 'Makoto Shinkai', 'Nitesh Tiwari', 'Bob Persichetti',
       'Anthony Russo', 'Lee Unkrich', 'Rajkumar Hirani', 'Aamir Khan',
       'Andrew Stanton', 'Florian Henckel von Donnersmarck',
       'Chan-wook Park', 'Stanley Kubrick', 'Hrishikesh Mukherjee',
       'Billy Wilder', 'Sam Mendes', 'Rahi Anil Barve', 'Sriram Raghavan',
       'Jeethu Joseph', 'Thomas Vinterberg', 'Asghar Farhadi',
       'Denis Villeneuve', 'Mehmet Ada Öztekin', 'Çagan Irmak',
       'Michel Gondry', 'Jean-Pierre Jeunet', 'Guy Ritchie',
```

In [456]:
```
df["Director"].unique()
```

Out[456]:
```
array(['Frank Darabont', 'Francis Ford Coppola', 'Christopher Nolan',
       'Sidney Lumet', 'Peter Jackson', 'Quentin Tarantino',
       'Steven Spielberg', 'David Fincher', 'Robert Zemeckis',
       'Sergio Leone', 'Lana Wachowski', 'Martin Scorsese',
       'Irvin Kershner', 'Milos Forman', 'Thomas Kail', 'Bong Joon Ho',
       'Sudha Kongara', 'Fernando Meirelles', 'Hayao Miyazaki',
       'Roberto Benigni', 'Jonathan Demme', 'George Lucas',
       'Masaki Kobayashi', 'Akira Kurosawa', 'Frank Capra',
       'Todd Phillips', 'Damien Chazelle', 'Olivier Nakache',
       'Roman Polanski', 'Ridley Scott', 'Tony Kaye', 'Bryan Singer',
       'Luc Besson', 'Roger Allers', 'James Cameron',
       'Giuseppe Tornatore', 'Isao Takahata', 'Alfred Hitchcock',
       'Michael Curtiz', 'Charles Chaplin', 'Nadine Labaki', 'Can Ulkay',
       'Gayatri', 'Makoto Shinkai', 'Nitesh Tiwari', 'Bob Persichetti',
       'Anthony Russo', 'Lee Unkrich', 'Rajkumar Hirani', 'Aamir Khan',
       'Andrew Stanton', 'Florian Henckel von Donnersmarck',
       'Chan-wook Park', 'Stanley Kubrick', 'Hrishikesh Mukherjee',
       'Billy Wilder', 'Sam Mendes', 'Rahi Anil Barve', 'Sriram Raghavan',
       'Jeethu Joseph', 'Thomas Vinterberg', 'Asghar Farhadi',
```

In [457]:
```
result = np. where(duni == 'Bradley Cooper')
if len(result) > 0 and len(result[0]) > 0:
    print(result[0][0])
```

```
510
```

In [458]:
```
#director mehtod
def Dir(col):
    result = np. where(duni==col)
    if len(result) > 0 and len(result[0]) > 0:
        return (result[0][0])
```

In [459]:
```
dfnew["Director"]=dfnew["Director"].apply(Dir)
```

In [460]:
```
dfnew["Certificate"].fillna("No Rated",inplace=True)
```

In [461]:
```
ceuni=dfnew["Certificate"].unique()
ceuni
```

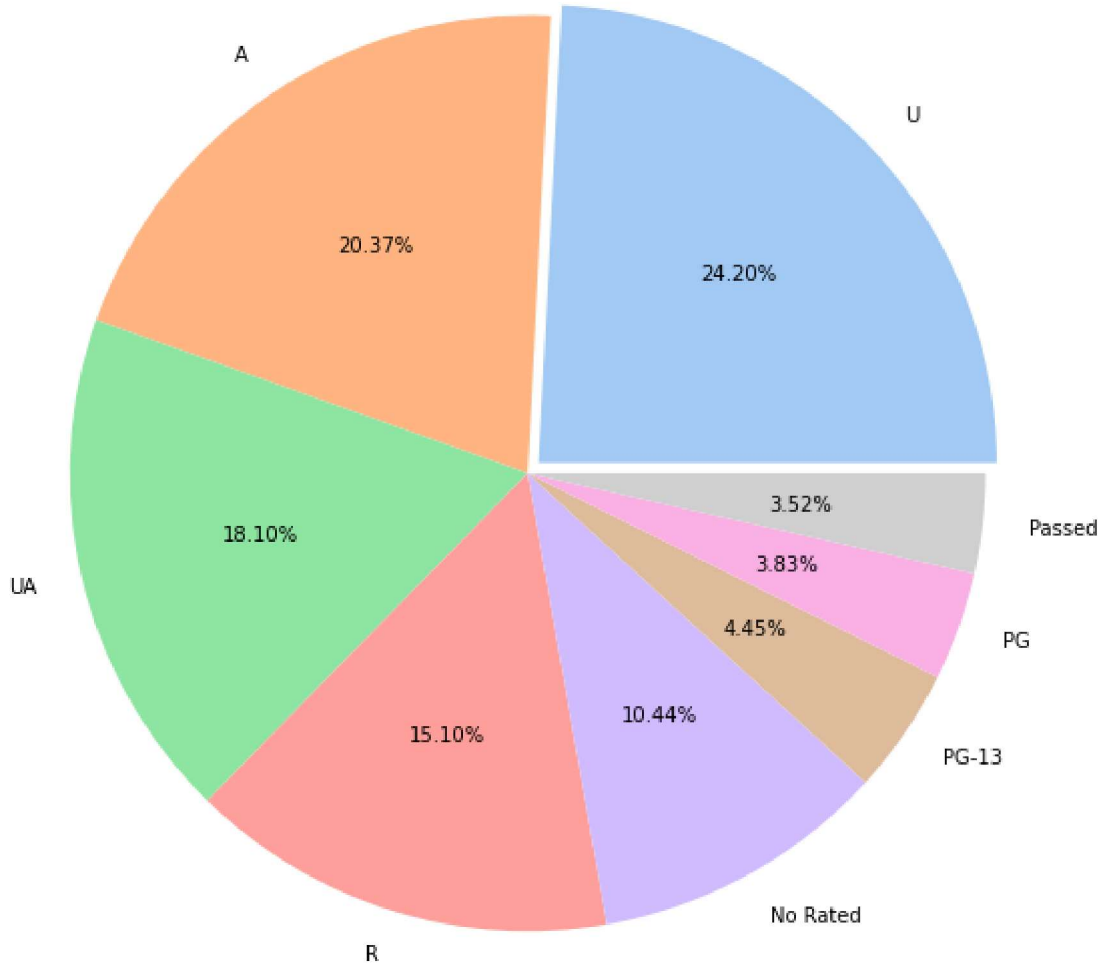
Out[461]:
```
array(['A', 'UA', 'U', 'PG-13', 'R', 'No Rated', 'PG', 'G', 'Passed',
       'TV-14', '16', 'TV-MA', 'Unrated', 'GP', 'Approved', 'TV-PG',
       'U/A'], dtype=object)
```

In [462]:
```
dfnew["Certificate"].value_counts()
```

Out[462]:
```
U            234
A            197
UA           175
R            146
No Rated     101
PG-13         43
PG            37
Passed        34
G             12
Approved      11
TV-PG          3
GP             2
U/A            1
Unrated        1
16             1
TV-MA          1
TV-14          1
Name: Certificate, dtype: int64
```

In [463]:
```
val=[234,197,175,146,101,43,37,34]
colors = sns.color_palette('pastel')[0:len(val)]
label=['U','A','UA','R','No Rated','PG-13','PG','Passed']
plt.pie(val,labels=label,autopct='%1.2f%%',colors=colors,radius=3,explode=(0.1,0,
plt.show()
```

```
C:\Users\Test\anaconda3\lib\site-packages\IPython\core\pylabtools.py:132: UserW
arning: Tight layout not applied. The bottom and top margins cannot be made lar
ge enough to accommodate all axes decorations.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [464]:
```
result1 = np. where(ceuni == 'U/A')
if len(result1) > 0 and len(result1[0]) > 0:
    print(result1[0][0])
```

```
16
```

In [465]:
```
#certificate mehtod
def Cer(col):
    result1 = np. where(ceuni==col)
    if len(result1) > 0 and len(result1[0]) > 0:
        return (result1[0][0])
```

In [466]:
```
dfnew["Certificate"]=dfnew["Certificate"].apply(Cer)
```

In [467]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 37 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   int64  
 3   Certificate    1000 non-null   int64  
 4   Runtime        1000 non-null   int32  
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     1000 non-null   float64
 9   Director       1000 non-null   int64  
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          1000 non-null   int64  
 16   Action        1000 non-null   uint8  
 17   Adventure     1000 non-null   uint8  
 18   Biography     1000 non-null   uint8  
 19   Comedy        1000 non-null   uint8  
 20   Crime         1000 non-null   uint8  
 21   Drama         1000 non-null   uint8  
 22   Family        1000 non-null   uint8  
 23   Fantasy       1000 non-null   uint8  
 24   Film-Noir     1000 non-null   uint8  
 25   History       1000 non-null   uint8  
 26   Horror        1000 non-null   uint8  
 27   Music         1000 non-null   uint8  
 28   Musical       1000 non-null   uint8  
 29   Mystery       1000 non-null   uint8  
 30   Romance       1000 non-null   uint8  
 31   Sci-Fi        1000 non-null   uint8  
 32   Sport         1000 non-null   uint8  
 33   Thriller      1000 non-null   uint8  
 34   War           1000 non-null   uint8  
 35   Western       1000 non-null   uint8  
 36  Animation      1000 non-null   uint8  
dtypes: float64(2), int32(1), int64(5), object(8), uint8(21)
memory usage: 141.7+ KB
```

In [468]:
```
dfnew.head()
```

Out[468]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	0	142	Drama
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	0	175	Crime, Drama
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	1	152	Action, Crime, Drama
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	0	202	Crime, Drama
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	2	96	Crime, Drama

5 rows × 37 columns

In [469]:
```
dfnew.describe()
```

Out[469]:

	Released_Year	Certificate	Runtime	IMDB_Rating	Meta_score	Director	No_of_V
count	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000
mean	1991.18100	2.722000	122.891000	7.949300	77.976000	214.443000	2.736929
std	23.31351	2.631281	28.093671	0.275491	11.362065	157.325733	3.273727
min	1920.00000	0.000000	45.000000	7.600000	28.000000	0.000000	2.508800
25%	1976.00000	1.000000	103.000000	7.700000	72.000000	71.000000	5.552625
50%	1999.00000	2.000000	119.000000	7.900000	78.000000	190.500000	1.385485
75%	2009.00000	4.000000	137.000000	8.100000	85.250000	330.250000	3.741612
max	2020.00000	16.000000	321.000000	9.300000	100.000000	547.000000	2.343110

8 rows × 29 columns

In [470]:
```
dfnew.columns
```

Out[470]:
```
Index(['Poster_Link', 'Series_Title', 'Released_Year', 'Certificate',
       'Runtime', 'Genre', 'IMDB_Rating', 'Overview', 'Meta_score', 'Director',
       'Star1', 'Star2', 'Star3', 'Star4', 'No_of_Votes', 'Gross', ' Action',
       ' Adventure', ' Biography', ' Comedy', ' Crime', ' Drama', ' Family',
       ' Fantasy', ' Film-Noir', ' History', ' Horror', ' Music', ' Musical',
       ' Mystery', ' Romance', ' Sci-Fi', ' Sport', ' Thriller', ' War',
       ' Western', 'Animation'],
      dtype='object')
```

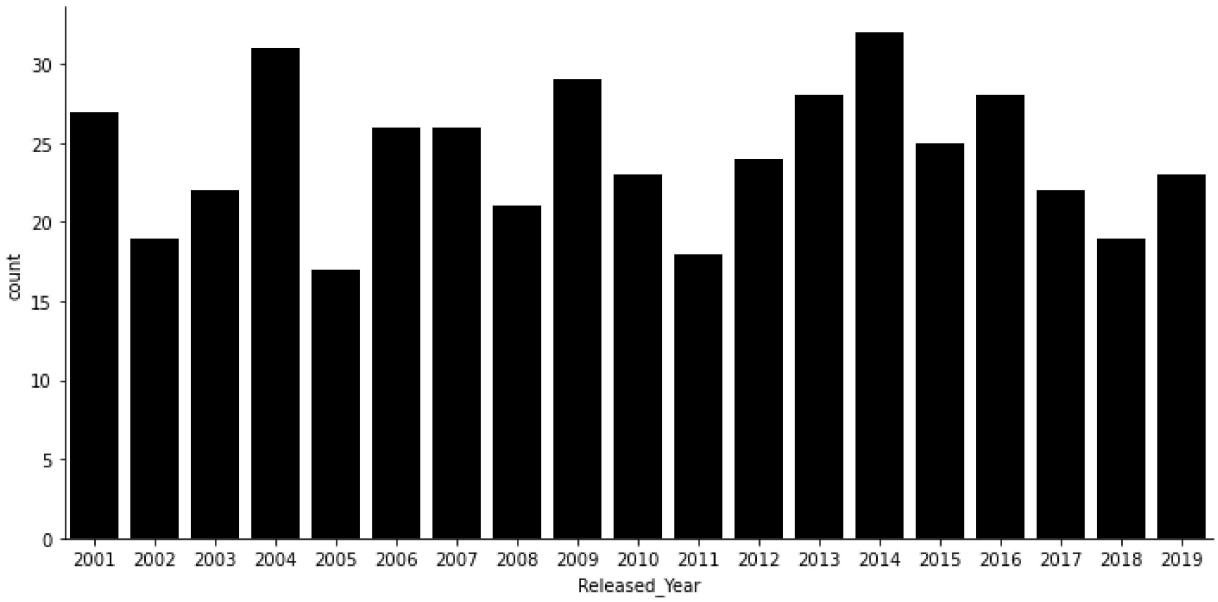
In [471]:
```
dfyear=dfnew[(dfnew["Released_Year"]>2000) & (dfnew["Released_Year"]<2020) ]
```

In [472]:
```
#count of movies in each year
plt.figure(figsize=(100,100))
sns.factorplot("Released_Year",aspect=2, data=dfyear,kind="count",color='black')
```

```
C:\Users\Test\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserWarn
ing: The `factorplot` function has been renamed to `catplot`. The original name
will be removed in a future release. Please update your code. Note that the def
ault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\Test\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

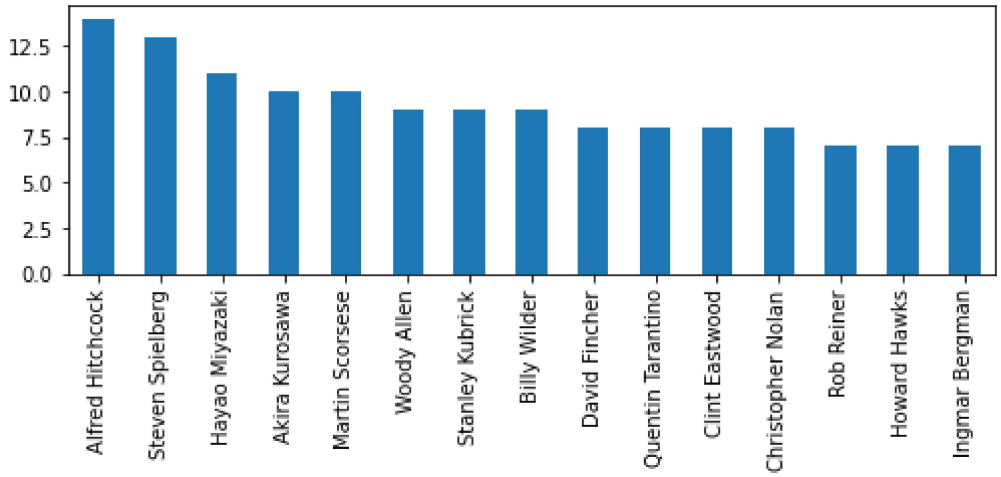
Out[472]:
```
<seaborn.axisgrid.FacetGrid at 0x29446113f10>

<Figure size 7200x7200 with 0 Axes>
```

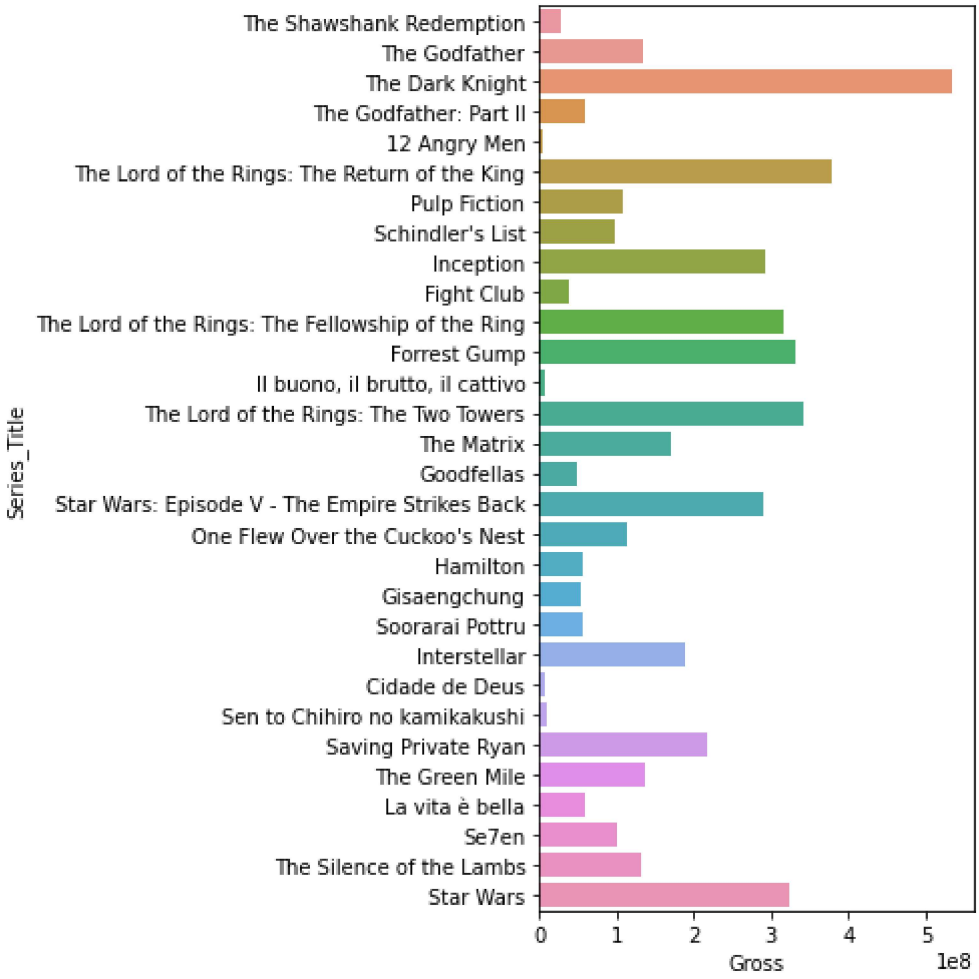


In [473]:
```
#counts of movies for each director
df["Director"].value_counts().head(15).plot.bar()
```

Out[473]: <AxesSubplot:>

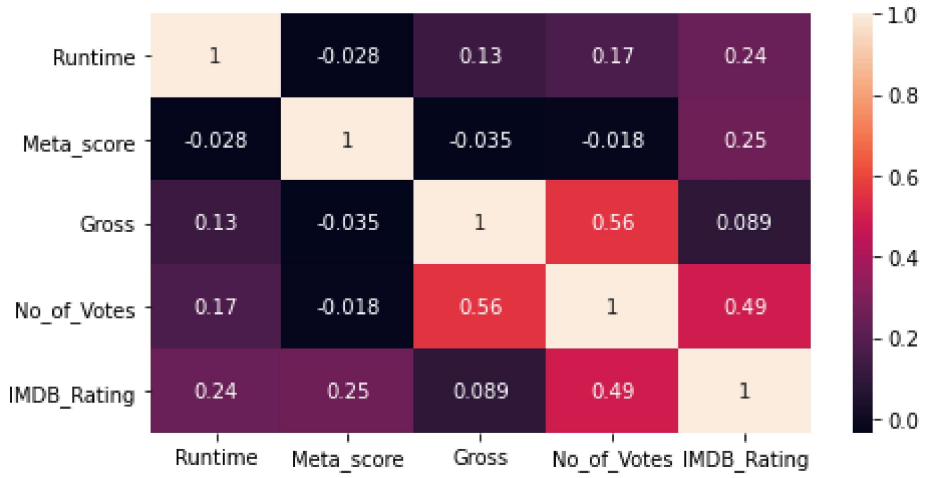


In [474]:
```
moviegross=dfnew.head(30)
plt.figure(figsize=(7,7))
sns.barplot(y="Series_Title", x="Gross", data=moviegross, ci= None)
plt.show()
```



In [475]:
```
a=dfnew[["Runtime","Meta_score","Gross","No_of_Votes","IMDB_Rating"]].corr()
sns.heatmap(a,annot=True)
```

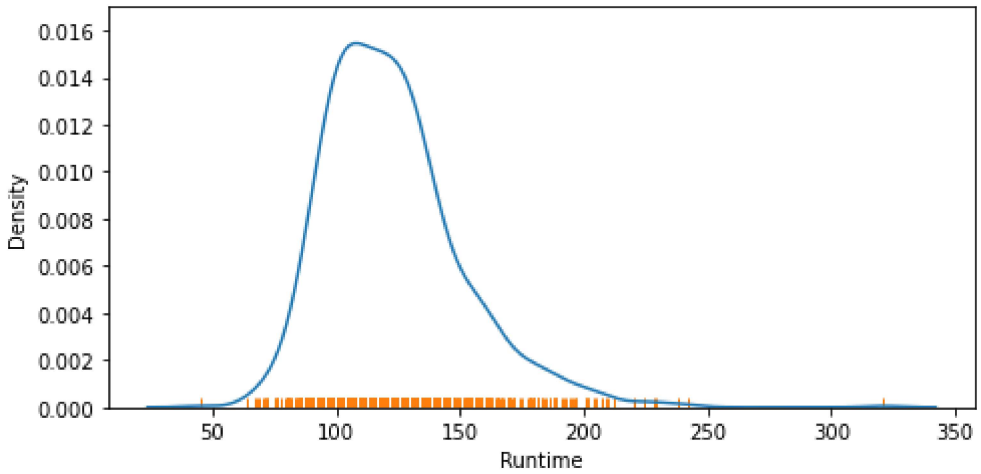
Out[475]: <AxesSubplot:>



In [476]:
```
#sns.jointplot("Genre",)
```

In [477]:
```
sns.kdeplot(dfnew['Runtime'])
sns.rugplot(dfnew['Runtime'])
```

Out[477]: <AxesSubplot:xlabel='Runtime', ylabel='Density'>



In [478]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 37 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   int64  
 3   Certificate    1000 non-null   int64  
 4   Runtime        1000 non-null   int32  
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     1000 non-null   float64
 9   Director       1000 non-null   int64  
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          1000 non-null   int64  
 16   Action        1000 non-null   uint8  
 17   Adventure     1000 non-null   uint8  
 18   Biography     1000 non-null   uint8  
 19   Comedy        1000 non-null   uint8  
 20   Crime         1000 non-null   uint8  
 21   Drama         1000 non-null   uint8  
 22   Family        1000 non-null   uint8  
 23   Fantasy       1000 non-null   uint8  
 24   Film-Noir     1000 non-null   uint8  
 25   History       1000 non-null   uint8  
 26   Horror        1000 non-null   uint8  
 27   Music         1000 non-null   uint8  
 28   Musical       1000 non-null   uint8  
 29   Mystery       1000 non-null   uint8  
 30   Romance       1000 non-null   uint8  
 31   Sci-Fi        1000 non-null   uint8  
 32   Sport         1000 non-null   uint8  
 33   Thriller      1000 non-null   uint8  
 34   War           1000 non-null   uint8  
 35   Western       1000 non-null   uint8  
 36  Animation      1000 non-null   uint8  
dtypes: float64(2), int32(1), int64(5), object(8), uint8(21)
memory usage: 141.7+ KB
```

In [479]:
```
dfnew=dfnew.drop(['Poster_Link','Series_Title','Overview','Genre','Star1','Star2'
```

In [480]:
```
dfscale=dfnew[["No_of_Votes","Gross"]]
dfscale
```

Out[480]:

	No_of_Votes	Gross
0	2343110	28341469
1	1620367	134966411
2	2303232	534858444
3	1129952	57300000
4	689845	4360000
...	...	...
995	166544	56536878
996	34075	56536878
997	43374	30500000
998	26471	56536878
999	51853	56536878

1000 rows × 2 columns

In [481]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
scaler=StandardScaler()
scaler=scaler.fit_transform(dfscale)
tempscale=pd.DataFrame(scaler)
tempscale
```

Out[481]:

	0	1
0	6.324451	-0.377200
1	4.115639	0.688198
2	6.202578	4.683928
3	2.616857	-0.087846
4	1.271824	-0.616824
...	...	...
995	-0.327463	-0.095471
996	-0.732308	-0.095471
997	-0.703889	-0.355632
998	-0.755547	-0.095471
999	-0.677976	-0.095471

1000 rows × 2 columns

In [482]:
```
#dfnew.to_excel("testnew.xlsx")
dftemp=dfnew
```

In [483]:
```
dftemp["No_of_Votes"]=tempscale[0]
dftemp["Gross"]=tempscale[1]
```

In [484]:
```
dfnew
```

Out[484]:

	Released_Year	Certificate	Runtime	IMDB_Rating	Meta_score	Director	No_of_Votes	Gro
0	1994	0	142	9.3	80.0	0	6.324451	-0.3772
1	1972	0	175	9.2	100.0	1	4.115639	0.6881
2	2008	1	152	9.0	84.0	2	6.202578	4.6839
3	1974	0	202	9.0	90.0	1	2.616857	-0.0878
4	1957	2	96	9.0	96.0	3	1.271824	-0.6168
...	...	...	...	...	...	...	...	
995	1961	0	115	7.6	76.0	546	-0.327463	-0.0954
996	1956	7	201	7.6	84.0	547	-0.732308	-0.0954
997	1953	8	118	7.6	85.0	336	-0.703889	-0.3556
998	1944	5	97	7.6	78.0	37	-0.755547	-0.0954
999	1935	5	86	7.6	93.0	37	-0.677976	-0.0954

1000 rows × 29 columns

In [485]:
```
from sklearn.cluster import KMeans
```

In [486]:
```
kmeans = KMeans(n_clusters=40)
```

In [487]:
```
kmeans.fit(dftemp)
```

Out[487]:
```
KMeans(n_clusters=40)
```

In [488]:
```
-df["Cluster"]=kmeans.labels_
```

In [489]:
```
df[df["Cluster"]==1]
```

Out[489]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Ge
690	https://m.media-amazon.com/images/M/MV5BZjMyZm...	Days of Heaven	1978	PG	94 min	Dra Roma
694	https://m.media-amazon.com/images/M/MV5BYjhhMD...	La planète sauvage	1973	U	72 min	Animati Sc
696	https://m.media-amazon.com/images/M/MV5BMDcxNj...	Badlands	1973	PG	94 min	Acti Crir Dra
698	https://m.media-amazon.com/images/M/MV5BZTIlND...	Willy Wonka & the Chocolate Factory	1971	U	100 min	Fan Fanta Mus
699	https://m.media-amazon.com/images/M/MV5BNTgwZm...	Midnight Cowboy	1969	A	113 min	Dra
700	https://m.media-amazon.com/images/M/MV5BMTQyNT...	Wait Until Dark	1967	NaN	108 min	Thri
702	https://m.media-amazon.com/images/M/MV5BOTViZm...	Bonnie and Clyde	1967	A	111 min	Acti Biograp Cri
707	https://m.media-amazon.com/images/M/MV5BNGQyNj...	The Innocents	1961	A	100 min	Hor
716	https://m.media-amazon.com/images/M/MV5BOTUzMz...	Bride of Frankenstein	1935	NaN	75 min	Dra Hor Sc
717	https://m.media-amazon.com/images/M/MV5BYmYxZG...	Duck Soup	1933	NaN	69 min	Come Musi V
719	https://m.media-amazon.com/images/M/MV5BMTQ0Nj...	Frankenstein	1931	Passed	70 min	Dra Hor Sc
876	https://m.media-amazon.com/images/M/MV5BYjlIMm...	The Invisible Man	1933	TV-PG	71 min	Hor Sc

In [490]:
```
user_movie=input("Your Favourite Movie : ")
movie_list=list(df["Series_Title"])
cluster_list=list(df["Cluster"])

for i in range(0,1000):
    if user_movie==movie_list[i]:
        cluster=cluster_list[i]

dfs=df[df["Cluster"]==cluster]

suggestion_movielist=list(dfs["Series_Title"])

print("")
print("*************************")
print("Movie Suggestion : ")
print("")

for movie in suggestion_movielist:
    if movie == user_movie :
        pass
    else:
        print(movie)
```

```
Your Favourite Movie : Evil Dead II

*************************
Movie Suggestion :

Ondskan
The Notebook
Lilja 4-ever
The Count of Monte Cristo
Remember the Titans
The Boondock Saints
October Sky
Shrek
Hana-bi
Gattaca
Tombstone
The Sandlot
The Remains of the Day
The Fugitive
A Bronx Tale
Batman: Mask of the Phantasm
Lat sau san taam
Night on Earth
Boyz n the Hood
Awakenings
Dip huet seung hung
Ferris Bueller's Day Off
Down by Law
The Goonies
The Breakfast Club
Ghostbusters
Kramer vs. Kramer
Lord of War
```

```
Dead Man
Planes, Trains & Automobiles
Lethal Weapon
```

In [492]:
```
df.to_excel("projectfinal.xlsx")
```

In [ ]:
```

```