In [33]:
```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)

pd.options.mode.chained_assignment = None
df = pd.read_csv(r'C:\Users\Sima\Desktop\movies.csv')
```

In [72]:
```
df.head()
```

Out[72]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budge
0	6587	6	6	1980	1705	8.4	927000.0	2589	4014	1047	54	19000000.
1	5573	6	1	1980	1492	5.8	65000.0	2269	1632	327	55	4500000.
2	5142	4	0	1980	1771	8.7	1200000.0	1111	2567	1745	55	18000000.
3	286	4	4	1980	1492	7.7	221000.0	1301	2000	2246	55	3500000.
4	1027	6	4	1980	1543	7.3	108000.0	1054	521	410	55	6000000.

In [35]:
```
#missing data
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

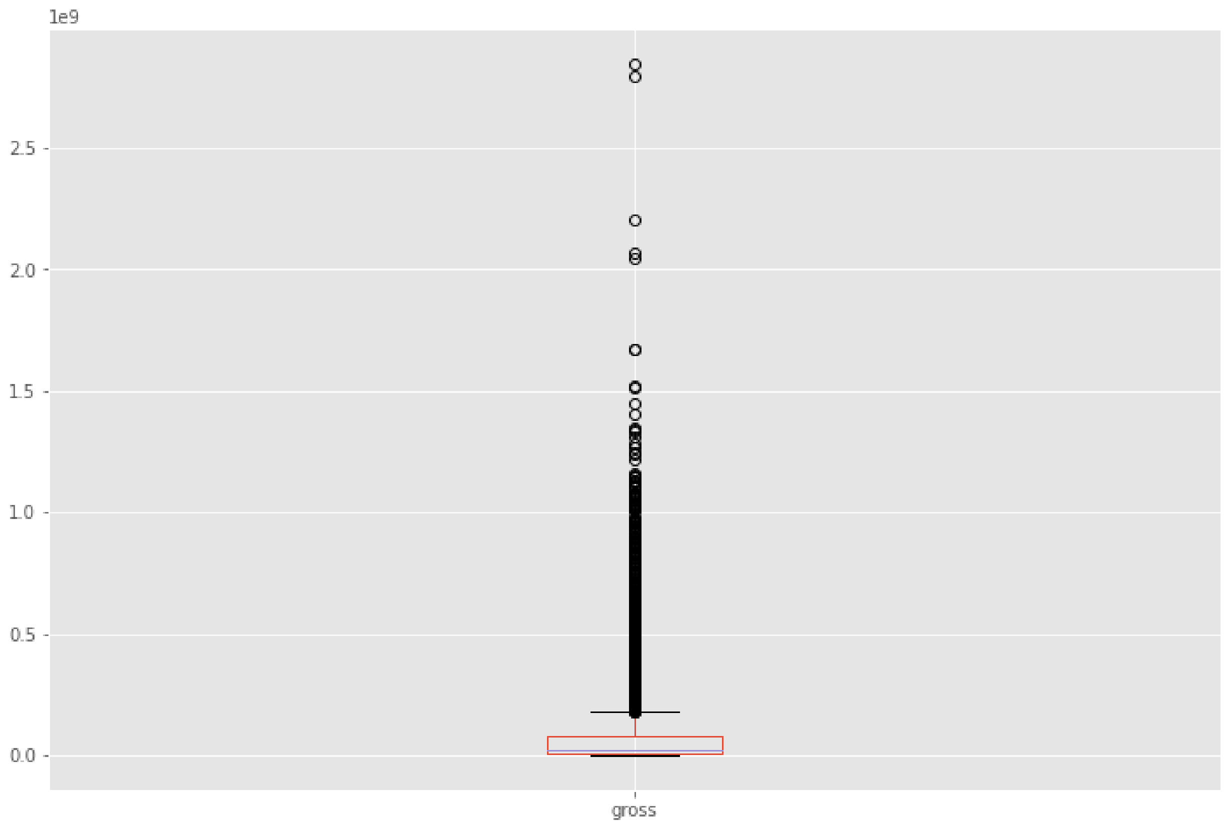
```
name - 0%
rating - 1%
genre - 0%
year - 0%
released - 0%
score - 0%
votes - 0%
director - 0%
writer - 0%
star - 0%
country - 0%
budget - 28%
gross - 2%
company - 0%
runtime - 0%
```

In [36]:
```
print(df.dtypes)
```

```
name         object
rating       object
genre        object
year          int64
released     object
score       float64
votes       float64
director     object
writer       object
star         object
country      object
budget      float64
gross       float64
company      object
runtime     float64
dtype: object
```

In [37]:
```
df.boxplot(column=['gross'])
```

Out[37]:
```
<AxesSubplot:>
```



In [38]:
```
df.drop_duplicates()
```

Out[38]:

	name	rating	genre	year	released	score	votes	director	writer	
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Nich
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Br Sh
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	H
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	R
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	C C
...	...	...	...	...	...	...	...	...	...	
7663	More to Life	NaN	Drama	2020	October 23, 2020 (United States)	3.1	18.0	Joseph Ebanks	Joseph Ebanks	Sha
7664	Dream Round	NaN	Comedy	2020	February 7, 2020 (United States)	4.7	36.0	Dusty Dukatz	Lisa Huston	Mi Sac
7665	Saving Mbango	NaN	Drama	2020	April 27, 2020 (Cameroon)	5.7	29.0	Nkanya Nkwai	Lynno Lovert	On I
7666	It's Just Us	NaN	Drama	2020	October 1, 2020 (United States)	NaN	NaN	James Randall	James Randall	Chr
7667	Tee em el	NaN	Horror	2020	August 19, 2020 (United States)	5.7	7.0	Pereko Mosia	Pereko Mosia	Siyab Ma

7668 rows × 15 columns

In [39]:
```
df.sort_values(by=['gross'], inplace=False, ascending=False)
```

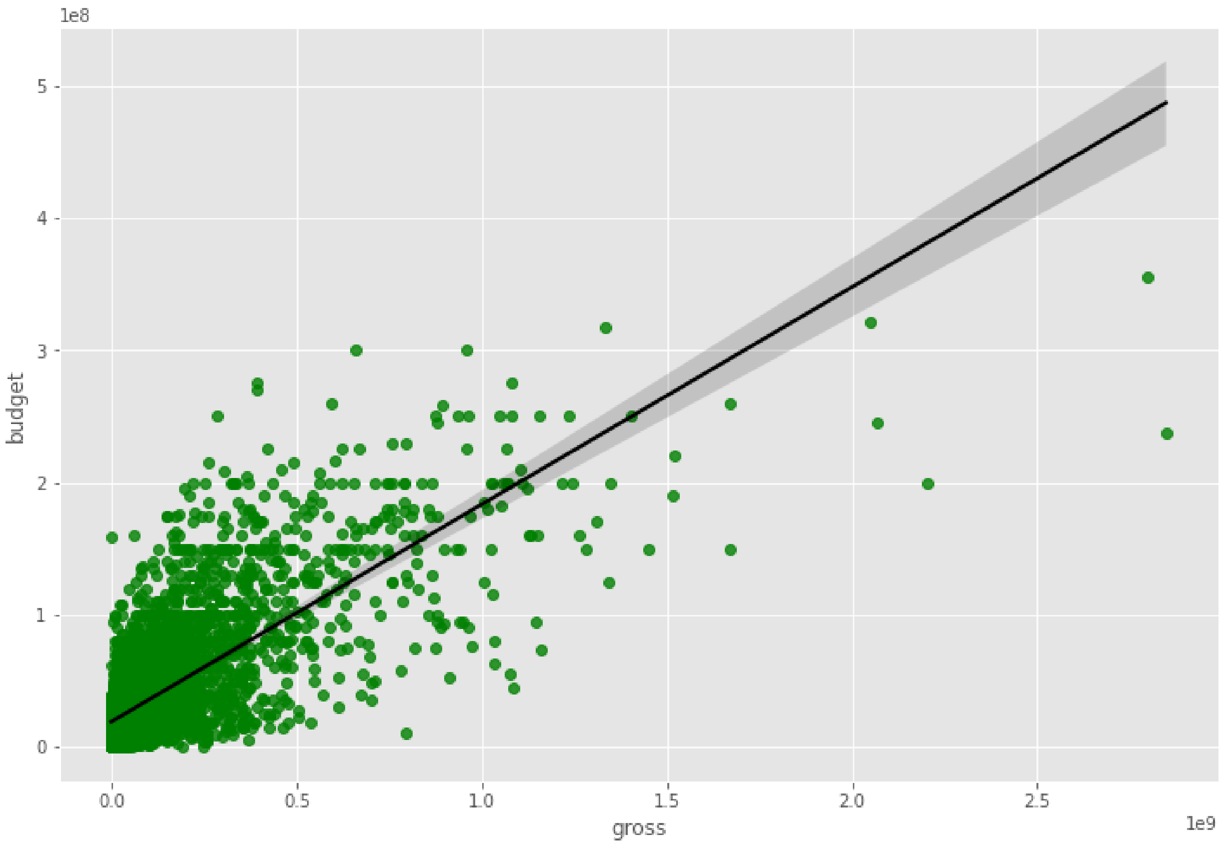
Out[39]:

	name	rating	genre	year	released	score	votes	director	writer	
5445	Avatar	PG-13	Action	2009	December 18, 2009 (United States)	7.8	1100000.0	James Cameron	James Cameron	S Worthing
7445	Avengers: Endgame	PG-13	Action	2019	April 26, 2019 (United States)	8.4	903000.0	Anthony Russo	Christopher Markus	Rol Downe
3045	Titanic	PG-13	Drama	1997	December 19, 1997 (United States)	7.8	1100000.0	James Cameron	James Cameron	Leona DiCa
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	2015	December 18, 2015 (United States)	7.8	876000.0	J.J. Abrams	Lawrence Kasdan	Daisy Ri
7244	Avengers: Infinity War	PG-13	Action	2018	April 27, 2018 (United States)	8.4	897000.0	Anthony Russo	Christopher Markus	Rol Downe
...	...	...	...	...	...	...	...	...	...	
7663	More to Life	NaN	Drama	2020	October 23, 2020 (United States)	3.1	18.0	Joseph Ebanks	Joseph Ebanks	Shan B
7664	Dream Round	NaN	Comedy	2020	February 7, 2020 (United States)	4.7	36.0	Dusty Dukatz	Lisa Huston	Micl Saqu
7665	Saving Mbango	NaN	Drama	2020	April 27, 2020 (Cameroon)	5.7	29.0	Nkanya Nkwai	Lynno Lovert	Onya La
7666	It's Just Us	NaN	Drama	2020	October 1, 2020 (United States)	NaN	NaN	James Randall	James Randall	Chris
7667	Tee em el	NaN	Horror	2020	August 19, 2020 (United States)	5.7	7.0	Pereko Mosia	Pereko Mosia	Siyabo Mab

7668 rows × 15 columns

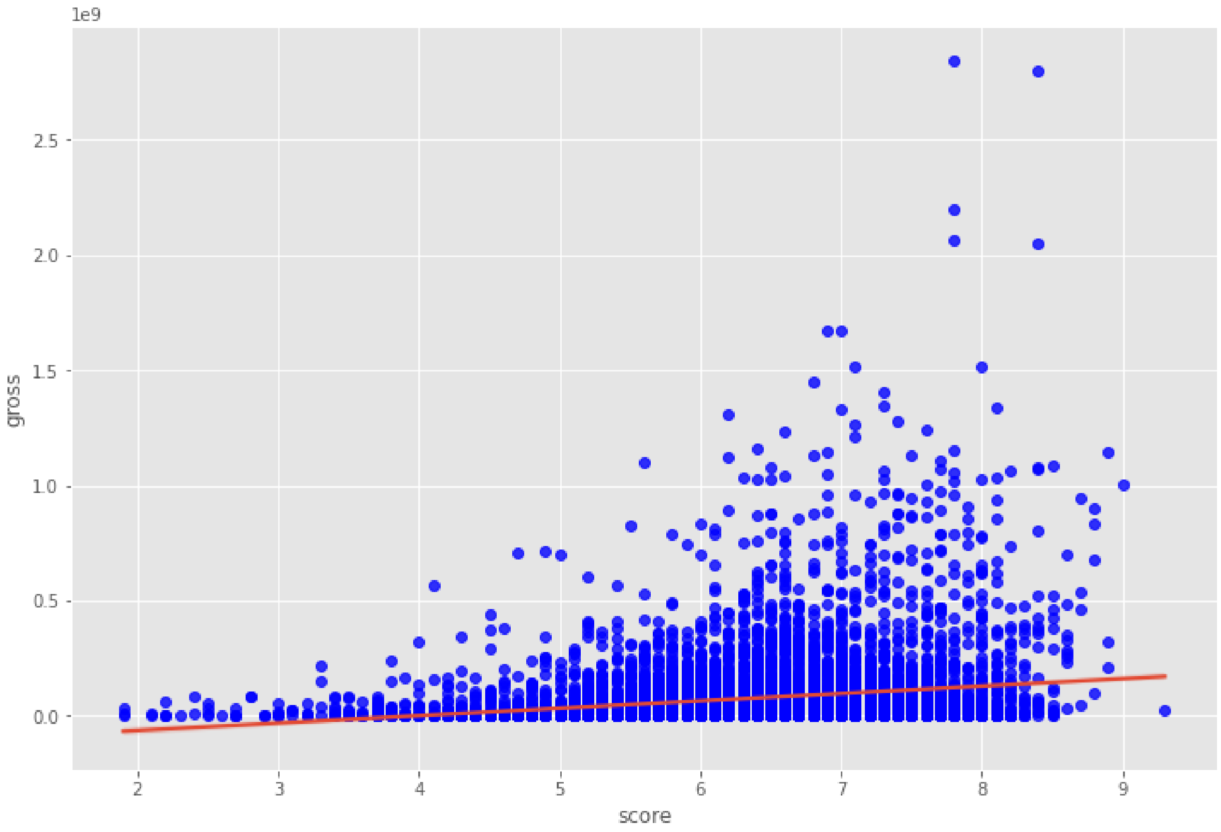
In [70]:
```
sns.regplot(x="gross", y="budget", data=df, scatter_kws={"color": "green"}, line_kws={
```

Out[70]:
```
<AxesSubplot:xlabel='gross', ylabel='budget'>
```



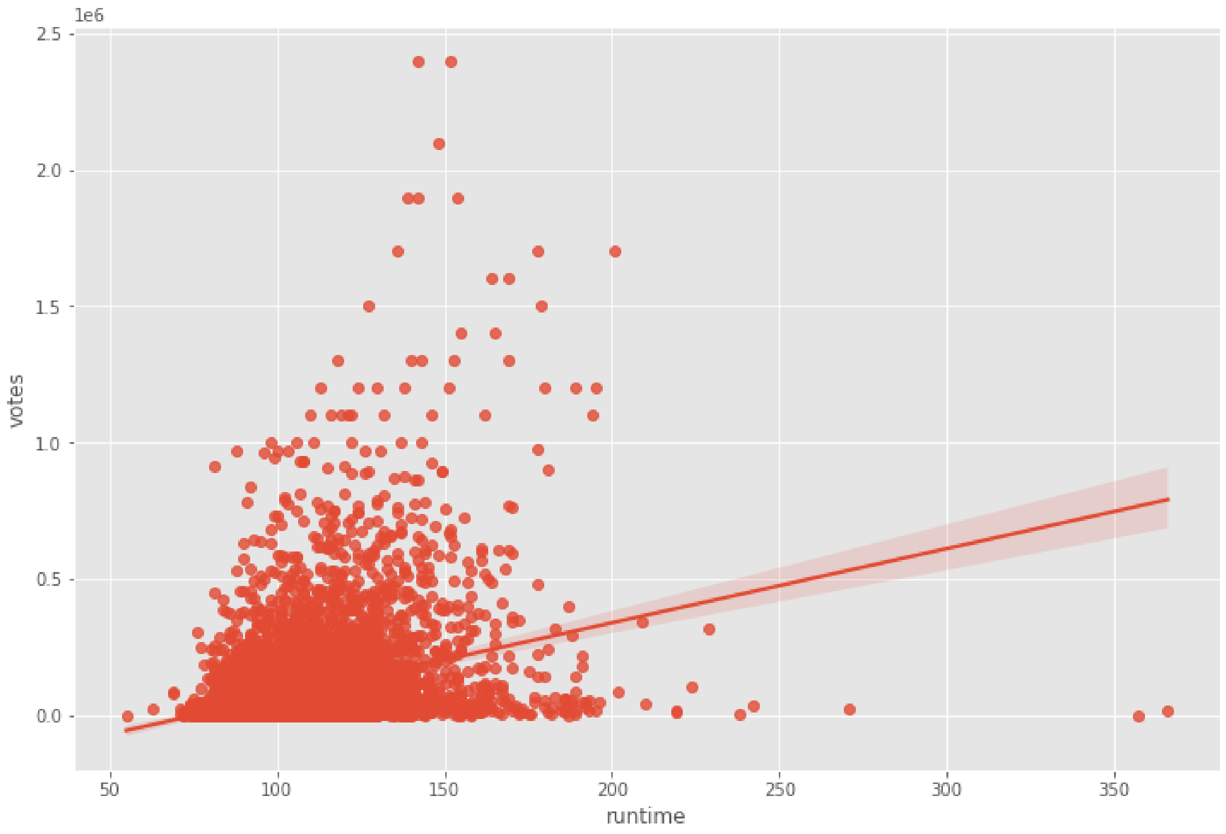
In [71]:
```
sns.regplot(x="score", y="gross", data=df,scatter_kws={"color": "blue"})
```

Out[71]:
```
<AxesSubplot:xlabel='score', ylabel='gross'>
```



In [44]:
```
sns.regplot(x="runtime", y="votes", data=df)
```

Out[44]:
```
<AxesSubplot:xlabel='runtime', ylabel='votes'>
```



In [45]:
```python
# Correlation Matrix between all numeric columns

df.corr(method ='pearson')
```

Out[45]:

	year	score	votes	budget	gross	runtime
year	1.000000	0.097995	0.222945	0.329321	0.257486	0.120811
score	0.097995	1.000000	0.409182	0.076254	0.186258	0.399451
votes	0.222945	0.409182	1.000000	0.442429	0.630757	0.309212
budget	0.329321	0.076254	0.442429	1.000000	0.740395	0.320447
gross	0.257486	0.186258	0.630757	0.740395	1.000000	0.245216
runtime	0.120811	0.399451	0.309212	0.320447	0.245216	1.000000

In [46]:
```python
df.corr(method ='kendall')
```

Out[46]:

	year	score	votes	budget	gross	runtime
year	1.000000	0.067652	0.331465	0.224120	0.200618	0.097184
score	0.067652	1.000000	0.300115	-0.000566	0.086046	0.283611
votes	0.331465	0.300115	1.000000	0.353702	0.548899	0.198240
budget	0.224120	-0.000566	0.353702	1.000000	0.512637	0.235483
gross	0.200618	0.086046	0.548899	0.512637	1.000000	0.168933
runtime	0.097184	0.283611	0.198240	0.235483	0.168933	1.000000

In [47]:

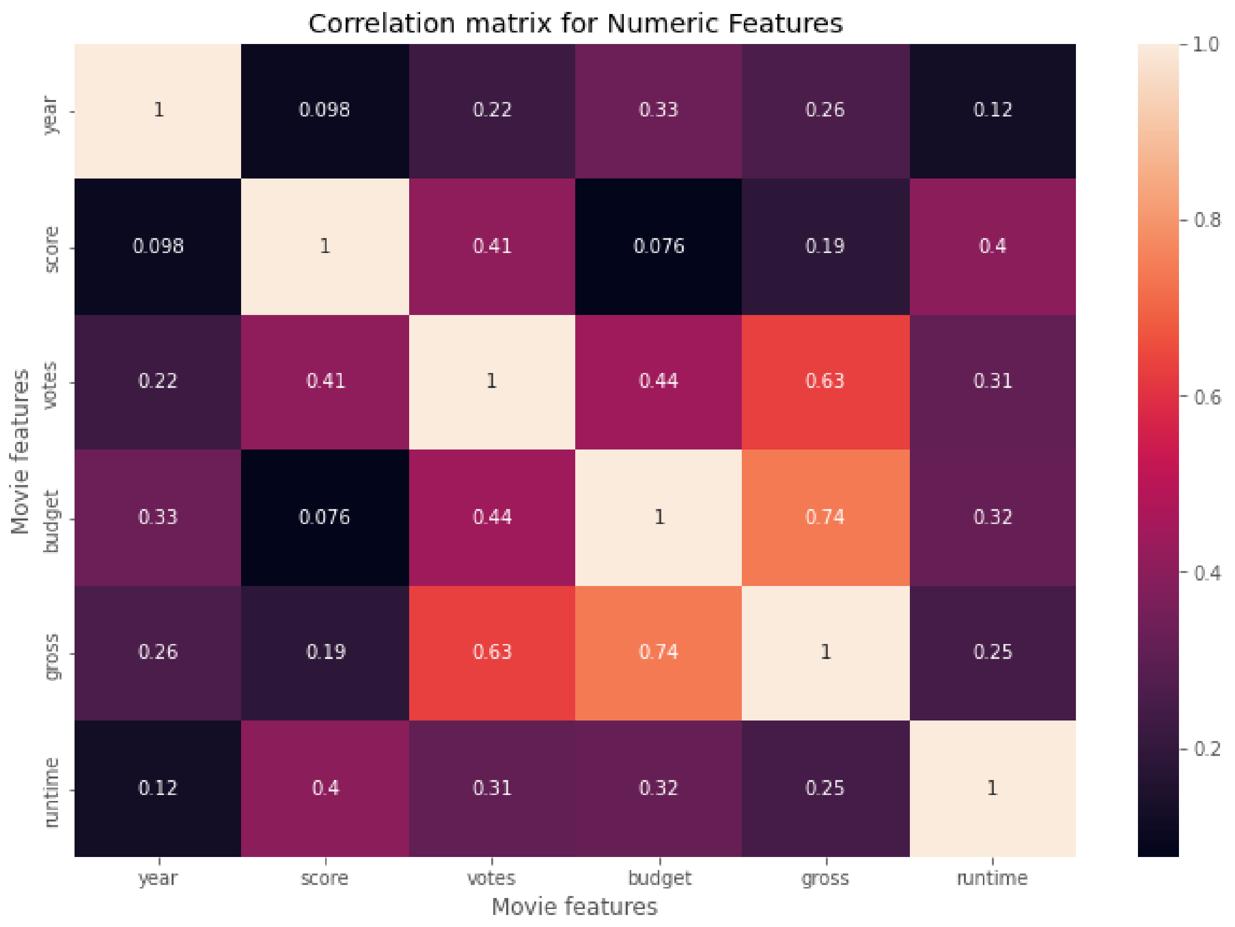
```
df.corr(method ='spearman')
```

Out[47]:

	year	score	votes	budget	gross	runtime
year	1.000000	0.099045	0.469829	0.317336	0.293084	0.142977
score	0.099045	1.000000	0.428138	-0.001403	0.126116	0.399857
votes	0.469829	0.428138	1.000000	0.502466	0.742050	0.290159
budget	0.317336	-0.001403	0.502466	1.000000	0.693670	0.336370
gross	0.293084	0.126116	0.742050	0.693670	1.000000	0.246243
runtime	0.142977	0.399857	0.290159	0.336370	0.246243	1.000000

In [48]:

```
correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Numeric Features")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()
```

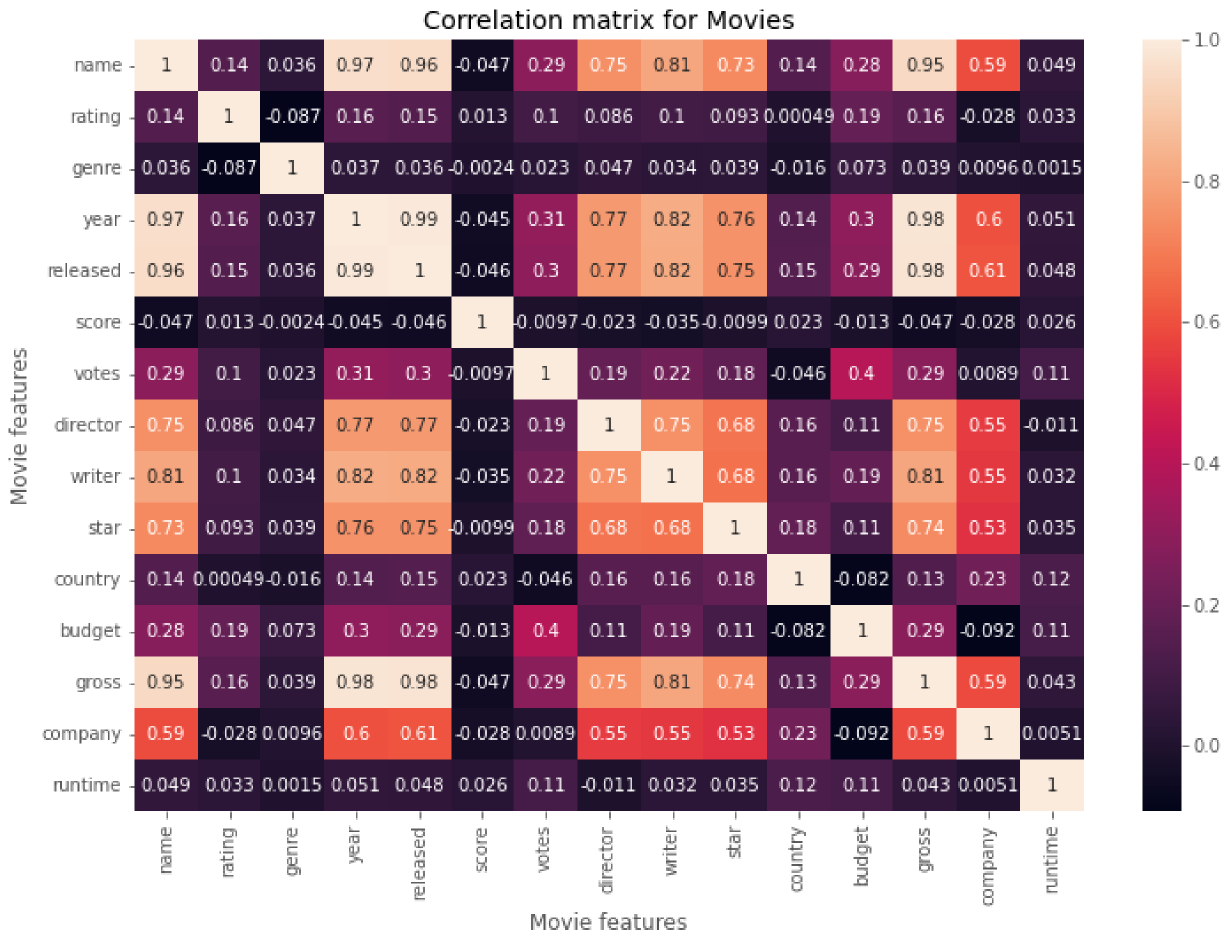


```
In [49]: df.apply(lambda x: x.factorize()[0]).corr(method='pearson')
```

Out[49]:

	name	rating	genre	year	released	score	votes	director	w
name	1.000000	0.143938	0.036367	0.965761	0.959015	-0.046733	0.287776	0.745905	0.80
rating	0.143938	1.000000	-0.086723	0.156713	0.146606	0.012595	0.099972	0.085520	0.10
genre	0.036367	-0.086723	1.000000	0.037184	0.035940	-0.002437	0.023285	0.047288	0.03
year	0.965761	0.156713	0.037184	1.000000	0.993190	-0.044981	0.312401	0.770497	0.82
released	0.959015	0.146606	0.035940	0.993190	1.000000	-0.045761	0.299905	0.770876	0.81
score	-0.046733	0.012595	-0.002437	-0.044981	-0.045761	1.000000	-0.009749	-0.022687	-0.03
votes	0.287776	0.099972	0.023285	0.312401	0.299905	-0.009749	1.000000	0.192220	0.22
director	0.745905	0.085520	0.047288	0.770497	0.770876	-0.022687	0.192220	1.000000	0.74
writer	0.805211	0.103623	0.033688	0.824770	0.819617	-0.034685	0.224122	0.748340	1.00
star	0.731565	0.093116	0.038649	0.756400	0.754468	-0.009896	0.179601	0.682385	0.67
country	0.142828	0.000494	-0.015795	0.140216	0.148468	0.023097	-0.045914	0.155471	0.15
budget	0.277488	0.193353	0.073008	0.300621	0.285691	-0.012642	0.398519	0.106617	0.18
gross	0.947324	0.158582	0.038616	0.980873	0.976423	-0.047041	0.286180	0.750911	0.80
company	0.591667	-0.028035	0.009566	0.601571	0.607954	-0.028432	0.008900	0.552258	0.54
runtime	0.048955	0.032741	0.001462	0.050647	0.048235	0.026436	0.106024	-0.011070	0.03

In [50]:
```
correlation_matrix = df.apply(lambda x: x.factorize()[0]).corr(method='pearson')

sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Movies")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()
```



In [51]:
```
correlation_mat = df.apply(lambda x: x.factorize()[0]).corr()

corr_pairs = correlation_mat.unstack()

print(corr_pairs)
```

```
name      name        1.000000
          rating      0.143938
          genre       0.036367
          year        0.965761
          released    0.959015
                        ...
runtime   country     0.124154
          budget      0.112097
          gross       0.042978
          company     0.005137
          runtime     1.000000
Length: 225, dtype: float64
```

In [52]:
```
sorted_pairs = corr_pairs.sort_values(kind="quicksort")

print(sorted_pairs)
```

```
budget    company    -0.092249
company   budget     -0.092249
genre     rating     -0.086723
rating    genre      -0.086723
budget    country    -0.082082
                        ...
year      year        1.000000
genre     genre       1.000000
rating    rating      1.000000
company   company     1.000000
runtime   runtime     1.000000
Length: 225, dtype: float64
```

In [53]:
```
strong_pairs = sorted_pairs[abs(sorted_pairs) > 0.5]

print(strong_pairs)
```

```
star      company     0.527116
company   star        0.527116
          writer      0.546151
writer    company     0.546151
director  company     0.552258
                        ...
year      year        1.000000
genre     genre       1.000000
rating    rating      1.000000
company   company     1.000000
runtime   runtime     1.000000
Length: 71, dtype: float64
```

In [54]:
```
CompanyGrossSum = df.groupby('company')[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values('gross', ascending = False)[:15]

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')

CompanyGrossSumSorted
```

Out[54]:
```
company
Warner Bros.                 56491421806
Universal Pictures           52514188890
Columbia Pictures            43008941346
Paramount Pictures           40493607415
Twentieth Century Fox        40257053857
Walt Disney Pictures         36327887792
New Line Cinema              19883797684
Marvel Studios               15065592411
DreamWorks Animation         11873612858
Touchstone Pictures          11795832638
Dreamworks Pictures          11635441081
Metro-Goldwyn-Mayer (MGM)     9230230105
Summit Entertainment          8373718838
Pixar Animation Studios       7886344526
Fox 2000 Pictures             7443502667
Name: gross, dtype: int64
```

In [55]:
```
df.groupby(['company', 'year'])[["gross"]].sum()
```

Out[55]:

company	year	gross
"DIA" Productions GmbH & Co. KG	2003	44350926.0
"Weathering With You" Film Partners	2019	193457467.0
.406 Production	1996	10580.0
1+2 Seisaku Iinkai	2000	1196218.0
10 West Studios	2010	814906.0
...	...	...
i am OTHER	2015	17986781.0
i5 Films	2001	10031529.0
iDeal Partners Film Fund	2013	506303.0
micro_scope	2010	7099598.0
thefyzz	2017	62198461.0

4536 rows × 1 columns

In [56]:
```
CompanyGrossSum = df.groupby(['company', 'year'])[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company','year'], ascend

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')

CompanyGrossSumSorted
```

Out[56]:
```
company                 year
Walt Disney Pictures    2019    5773131804
Marvel Studios          2018    4018631866
Universal Pictures      2015    3834354888
Twentieth Century Fox   2009    3793491246
Walt Disney Pictures    2017    3789382071
Paramount Pictures      2011    3565705182
Warner Bros.            2010    3300479986
                        2011    3223799224
Walt Disney Pictures    2010    3104474158
Paramount Pictures      2014    3071298586
Columbia Pictures       2006    2934631933
                        2019    2932757449
Marvel Studios          2019    2797501328
Warner Bros.            2018    2774168962
Columbia Pictures       2011    2738363306
Name: gross, dtype: int64
```

In [57]:
```
CompanyGrossSum = df.groupby(['company'])[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company'], ascending = F

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')

CompanyGrossSumSorted
```

Out[57]:
```
company
Warner Bros.                 56491421806
Universal Pictures           52514188890
Columbia Pictures            43008941346
Paramount Pictures           40493607415
Twentieth Century Fox        40257053857
Walt Disney Pictures         36327887792
New Line Cinema              19883797684
Marvel Studios               15065592411
DreamWorks Animation         11873612858
Touchstone Pictures          11795832638
Dreamworks Pictures          11635441081
Metro-Goldwyn-Mayer (MGM)     9230230105
Summit Entertainment          8373718838
Pixar Animation Studios       7886344526
Fox 2000 Pictures             7443502667
Name: gross, dtype: int64
```

In [ ]:
```
plt.scatter(x=df['budget'], y=df['gross'], alpha=0.8)
plt.title('Budget vs Gross Earnings')
plt.xlabel('Gross Earnings')
plt.ylabel('Budget for Film')
plt.show()
```

In [59]:
```
df_numerized = df


for col_name in df_numerized.columns:
    if(df_numerized[col_name].dtype == 'object'):
        df_numerized[col_name]= df_numerized[col_name].astype('category')
        df_numerized[col_name] = df_numerized[col_name].cat.codes

df_numerized
```

Out[59]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	bu
0	6587	6	6	1980	1705	8.4	927000.0	2589	4014	1047	54	19000
1	5573	6	1	1980	1492	5.8	65000.0	2269	1632	327	55	4500
2	5142	4	0	1980	1771	8.7	1200000.0	1111	2567	1745	55	18000
3	286	4	4	1980	1492	7.7	221000.0	1301	2000	2246	55	3500
4	1027	6	4	1980	1543	7.3	108000.0	1054	521	410	55	6000
...	...	...	...	...	...	...	...	...	...	...	...	
7663	3705	-1	6	2020	2964	3.1	18.0	1500	2289	2421	55	7
7664	1678	-1	4	2020	1107	4.7	36.0	774	2614	1886	55	
7665	4717	-1	6	2020	193	5.7	29.0	2061	2683	2040	55	58
7666	2843	-1	6	2020	2817	NaN	NaN	1184	1824	450	55	15
7667	5394	-1	10	2020	391	5.7	7.0	2165	3344	2463	44	

7668 rows × 15 columns

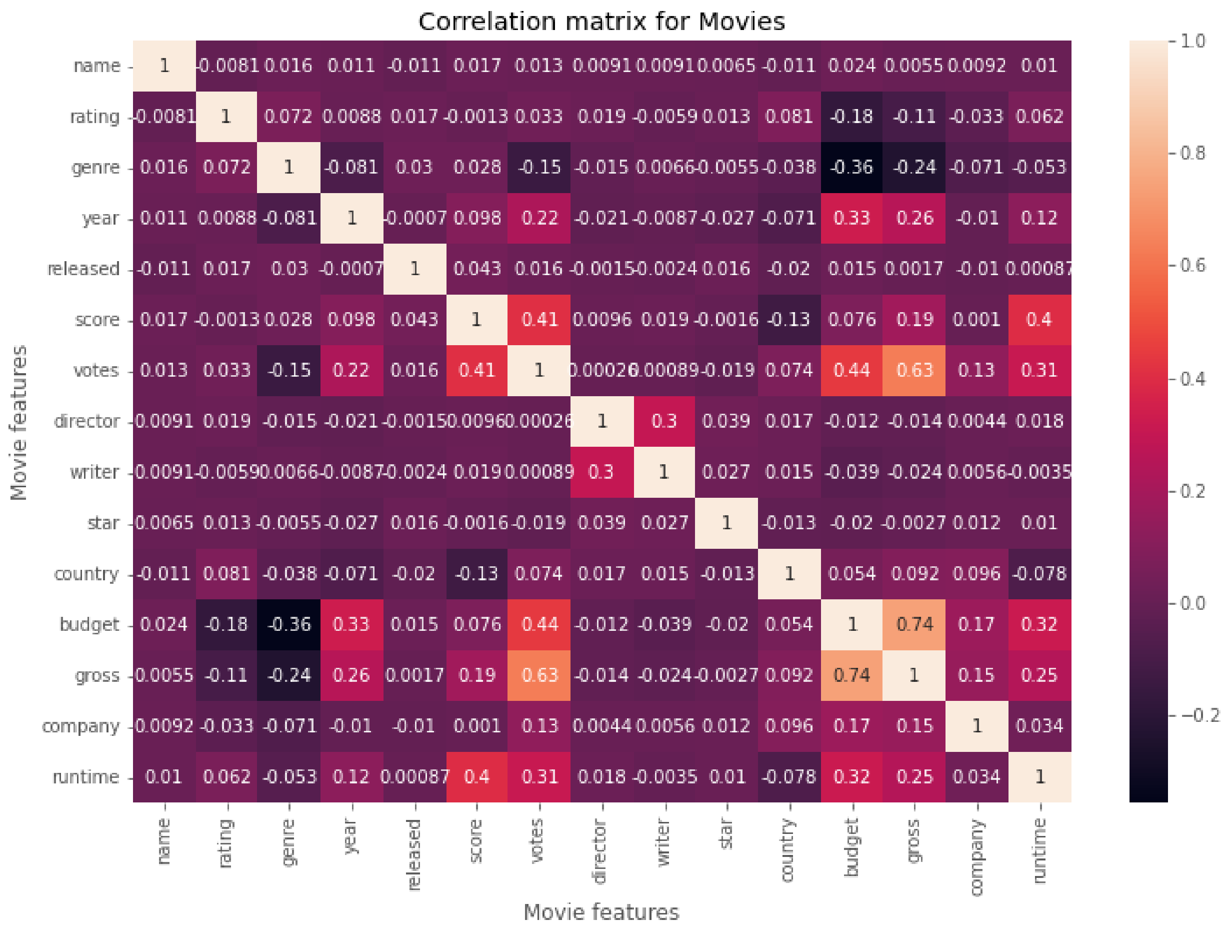
In [60]:
```python
df_numerized.corr(method='pearson')
```

Out[60]:

	name	rating	genre	year	released	score	votes	director	w
name	1.000000	-0.008069	0.016355	0.011453	-0.011311	0.017097	0.013088	0.009079	0.00
rating	-0.008069	1.000000	0.072423	0.008779	0.016613	-0.001314	0.033225	0.019483	-0.00
genre	0.016355	0.072423	1.000000	-0.081261	0.029822	0.027965	-0.145307	-0.015258	0.00
year	0.011453	0.008779	-0.081261	1.000000	-0.000695	0.097995	0.222945	-0.020795	-0.00
released	-0.011311	0.016613	0.029822	-0.000695	1.000000	0.042788	0.016097	-0.001478	-0.00
score	0.017097	-0.001314	0.027965	0.097995	0.042788	1.000000	0.409182	0.009559	0.01
votes	0.013088	0.033225	-0.145307	0.222945	0.016097	0.409182	1.000000	0.000260	0.00
director	0.009079	0.019483	-0.015258	-0.020795	-0.001478	0.009559	0.000260	1.000000	0.29
writer	0.009081	-0.005921	0.006567	-0.008656	-0.002404	0.019416	0.000892	0.299067	1.00
star	0.006472	0.013405	-0.005477	-0.027242	0.015777	-0.001609	-0.019282	0.039234	0.02
country	-0.010737	0.081244	-0.037615	-0.070938	-0.020427	-0.133348	0.073625	0.017490	0.01
budget	0.023970	-0.176002	-0.356564	0.329321	0.014683	0.076254	0.442429	-0.012272	-0.03
gross	0.005533	-0.107339	-0.235650	0.257486	0.001659	0.186258	0.630757	-0.014441	-0.02
company	0.009211	-0.032943	-0.071067	-0.010431	-0.010474	0.001030	0.133204	0.004404	0.00
runtime	0.010392	0.062145	-0.052711	0.120811	0.000868	0.399451	0.309212	0.017624	-0.00

In [61]:
```python
correlation_matrix = df_numerized.corr(method='pearson')
```

```
sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Movies")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()
```



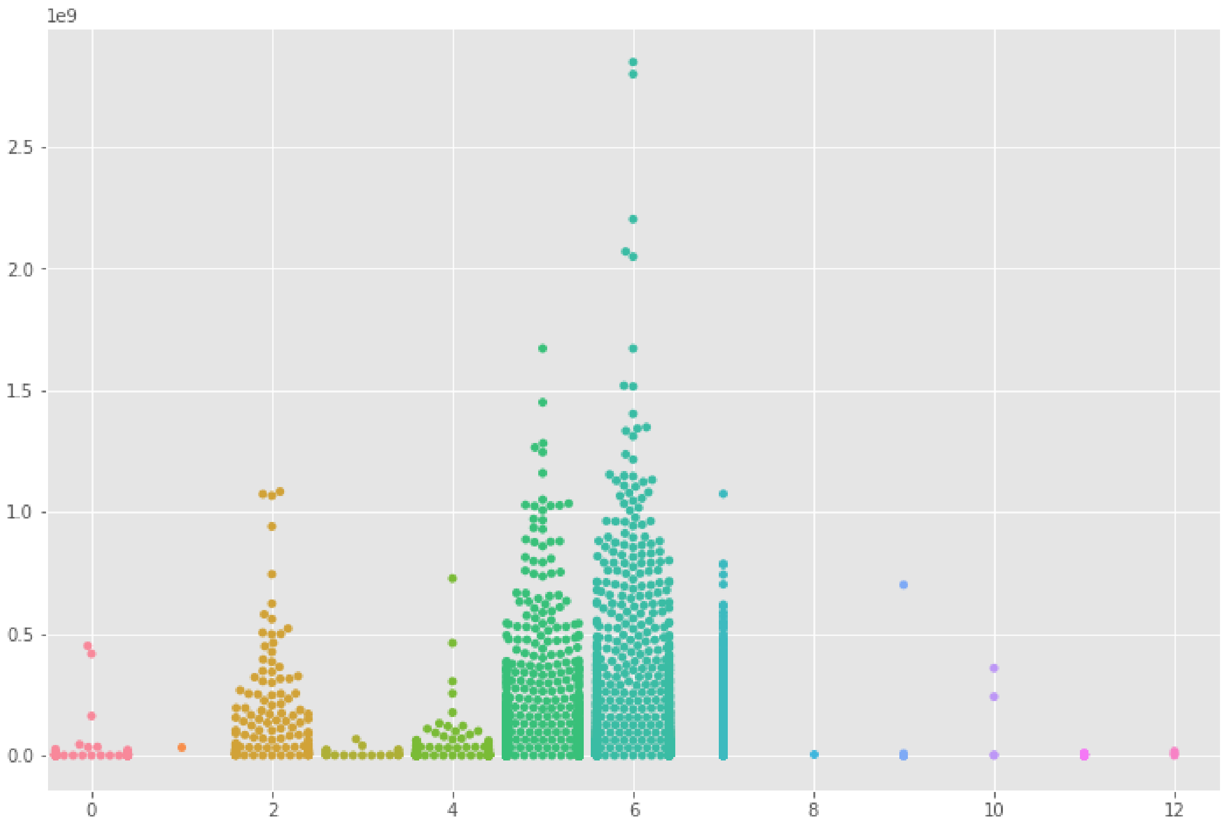
```
In [62]: for col_name in df.columns:
             if(df[col_name].dtype == 'object'):
                 df[col_name]= df[col_name].astype('category')
                 df[col_name] = df[col_name].cat.codes
```

```
In [66]: sns.swarmplot(x="rating", y="gross", data=df)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
53.2% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
48.4% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
60.9% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
80.6% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
84.4% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning:
88.2% of the points cannot be placed; you may want to decrease the size of the marker
s or use stripplot.
  warnings.warn(msg, UserWarning)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [66], in <cell line: 1>()
----> 1 sns.swarmplot(x="rating", y="gross", data=df)

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:46, in _deprec
ate_positional_args.<locals>.inner_f(*args, **kwargs)
     36     warnings.warn(
     37         "Pass the following variable{} as {}keyword arg{}: {}. "
     38         "From version 0.12, the only valid positional argument "
   (...)
     43         FutureWarning
     44     )
     45 kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46 return f(**kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3019, in swarm
plot(x, y, hue, data, order, hue_order, dodge, orient, color, palette, size, edgecolo
r, linewidth, ax, **kwargs)
   3014     edgecolor = plotter.gray
   3015 kwargs.update(dict(s=size ** 2,
   3016                    edgecolor=edgecolor,
   3017                    linewidth=linewidth))
-> 3019 plotter.plot(ax, kwargs)
   3020 return ax

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1420, in _Swar
mPlotter.plot(self, ax, kws)
   1418 def plot(self, ax, kws):
   1419     """Make the full plot."""
-> 1420     self.draw_swarmplot(ax, kws)
   1421     self.add_legend_data(ax)
   1422     self.annotate_axes(ax)

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1416, in _Swar
mPlotter.draw_swarmplot(self, ax, kws)
   1414 for center, swarm in zip(centers, swarms):
   1415     if swarm.get_offsets().size:
-> 1416         self.swarm_points(ax, swarm, center, width, s, **kws)

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1318, in _Swar
mPlotter.swarm_points(self, ax, points, center, width, s, **kws)
   1315     orig_xy = orig_xy[:, [1, 0]]
   1317 # Do the beeswarm in point coordinates
-> 1318 new_xy = self.beeswarm(orig_xy, d)
   1320 # Transform the point coordinates back to data coordinates
   1321 if self.orient == "h":

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1270, in _Swar
mPlotter.beeswarm(self, orig_xy, d)
   1267 candidates = candidates[np.argsort(offsets)]
   1269 # Find the first candidate that does not overlap any neighbours
-> 1270 new_xy_i = self.first_non_overlapping_candidate(candidates,
   1271                                                 neighbors, d)
   1273 # Place it into the swarm
   1274 swarm.append(new_xy_i)

File C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1229, in _Swar
mPlotter.first_non_overlapping_candidate(self, candidates, neighbors, d)
   1226 dx = neighbors_x - x_i
```

```
    1227 dy = neighbors_y - y_i
-> 1229 sq_distances = np.power(dx, 2.0) + np.power(dy, 2.0)
    1231 # good candidate does not overlap any of neighbors
    1232 # which means that squared distance between candidate
    1233 # and any of the neighbours has to be at least
    1234 # square of the diameter
    1235 good_candidate = np.all(sq_distances >= d_square)

KeyboardInterrupt:
```



In [67]:
```
sorted_pairs = corr_pairs.sort_values()
sorted_pairs
```

Out[67]:
```
budget   company   -0.092249
company  budget    -0.092249
genre    rating    -0.086723
rating   genre     -0.086723
budget   country   -0.082082
                      ...
year     year       1.000000
genre    genre      1.000000
rating   rating     1.000000
company  company    1.000000
runtime  runtime    1.000000
Length: 225, dtype: float64
```

In [69]:
```
high_corr = sorted_pairs[(sorted_pairs) > 0.5]
high_corr
```

Out[69]:
```
star      company    0.527116
company   star       0.527116
          writer     0.546151
writer    company    0.546151
director  company    0.552258
                       ...
year      year       1.000000
genre     genre      1.000000
rating    rating     1.000000
company   company    1.000000
runtime   runtime    1.000000
Length: 71, dtype: float64
```

In [ ]:
```
# company has low correlation
#votes and budget have the highest correlation
```