In [28]:
```
# import libraries

import pandas as pd
import numpy as np
import seaborn as sns

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

%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8) #Adjust the configuration of the plo

# Read in the data

df= pd.read_csv(r'C:\Users\gaura\Downloads\movies.csv')
```

In [29]:
```
# Let's look at the data

df.head()
```

Out[29]:

	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
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole
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
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray

In [30]:
```
# Let's see if there is any 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 [31]:
```
# Dropping the rows with missing data

df = df.dropna()
```

In [32]:
```
# Clean Data

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

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

In [33]:
```
df.dtypes
```

Out[33]:
```
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 [34]:
```
# Changing the Data Types

df['budget'] = df['budget'].astype('int64')
df['gross'] = df['gross'].astype('int64')
df['votes'] = df['votes'].astype('int64')
```

In [35]:
```
df.head()
```

Out[35]:

	name	rating	genre	year	released	score	votes	director	writer	
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000	Stanley Kubrick	Stephen King	Ni
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000	Randal Kleiser	Henry De Vere Stacpoole	
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000	Irvin Kershner	Leigh Brackett	
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000	Jim Abrahams	Jim Abrahams	
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000	Harold Ramis	Brian Doyle-Murray	

In [36]:
```
# create correct year column, we can see some years that does not match with releas
```

```
df['yearcorrect'] = df['released'].str.extract(pat = '([0-9]{4})').astype(int)
df.head()
```

Out[36]:

	name	rating	genre	year	released	score	votes	director	writer	
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000	Stanley Kubrick	Stephen King	Ni
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000	Randal Kleiser	Henry De Vere Stacpoole	
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000	Irvin Kershner	Leigh Brackett	
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000	Jim Abrahams	Jim Abrahams	
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000	Harold Ramis	Brian Doyle-Murray	

In [37]:
```
df.drop(columns=['year'], inplace=True)
```

In [38]:
```
df= df.sort_values(by=['gross'], inplace=False, ascending=False)
```

In [39]:
```
# If you want to see the entire DATA

# pd.set_option('display.max_rows', None)
```

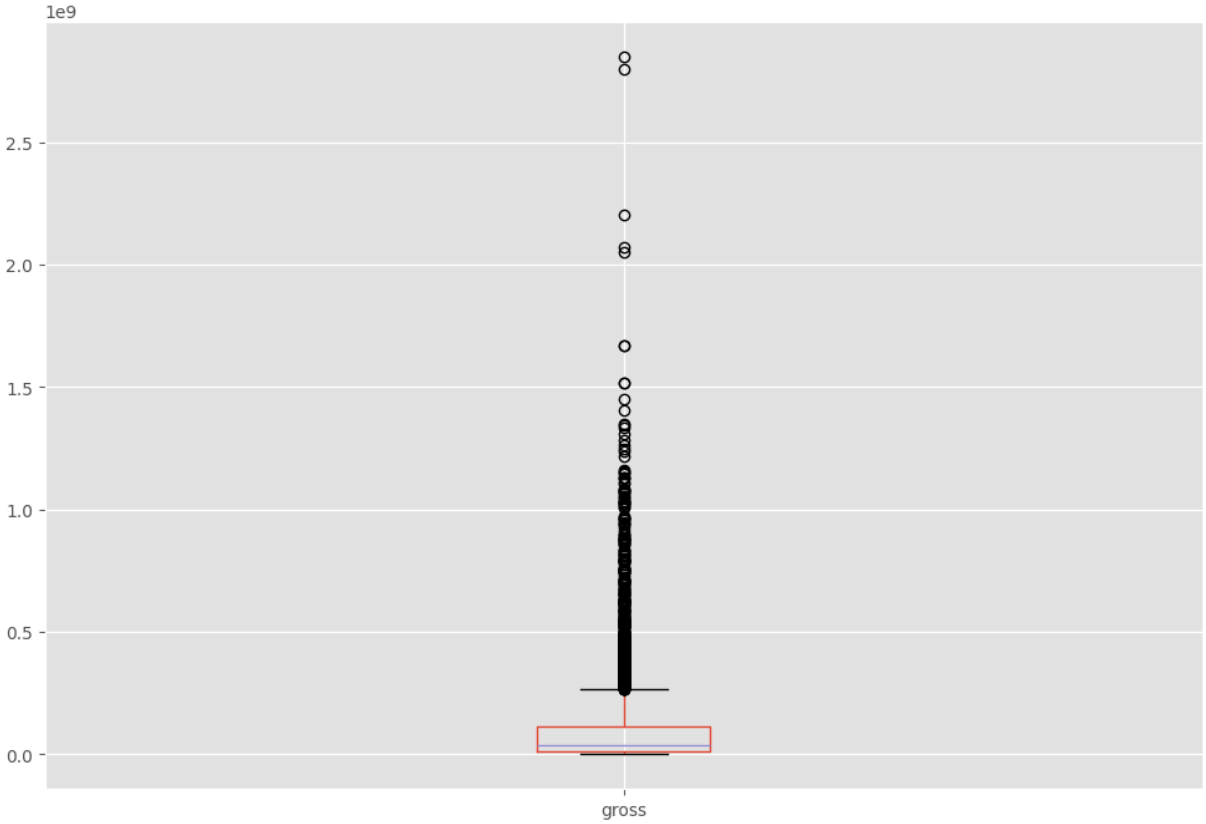
In [40]:
```
# Drop any duplicates

df.drop_duplicates()
df.head()
```

Out[40]:

	name	rating	genre	released	score	votes	director	writer	s
5445	Avatar	PG-13	Action	December 18, 2009 (United States)	7.8	1100000	James Cameron	James Cameron	S Worthing
7445	Avengers: Endgame	PG-13	Action	April 26, 2019 (United States)	8.4	903000	Anthony Russo	Christopher Markus	Rob Downey
3045	Titanic	PG-13	Drama	December 19, 1997 (United States)	7.8	1100000	James Cameron	James Cameron	Leonar DiCap
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	December 18, 2015 (United States)	7.8	876000	J.J. Abrams	Lawrence Kasdan	Daisy Rid
7244	Avengers: Infinity War	PG-13	Action	April 27, 2018 (United States)	8.4	897000	Anthony Russo	Christopher Markus	Rob Downey

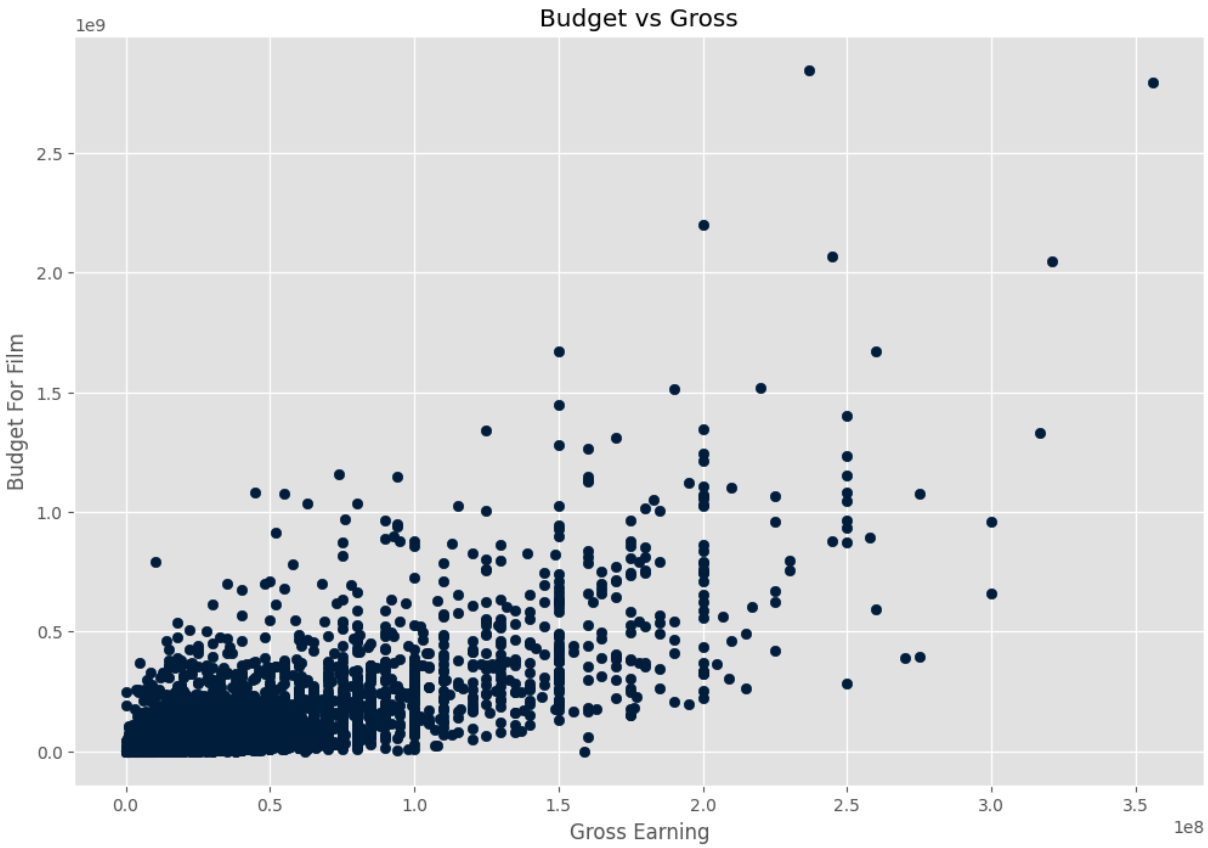
In [41]:
```
# Now let's box plot the Gross

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

Out[41]:
```
<Axes: >
```



In [42]:
```
# Let's build a scatter plot to compare buget and gross

plt.scatter(x=df['budget'], y=df['gross'], color='#001f3f')

plt.title('Budget vs Gross')
plt.xlabel('Gross Earning')
plt.ylabel('Budget For Film')
plt.show
```

Out[42]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```


Budget vs Gross
1e9
2.5
2.0
1.5
1.0
0.5
0.0
Budget For Film
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e8
Gross Earning

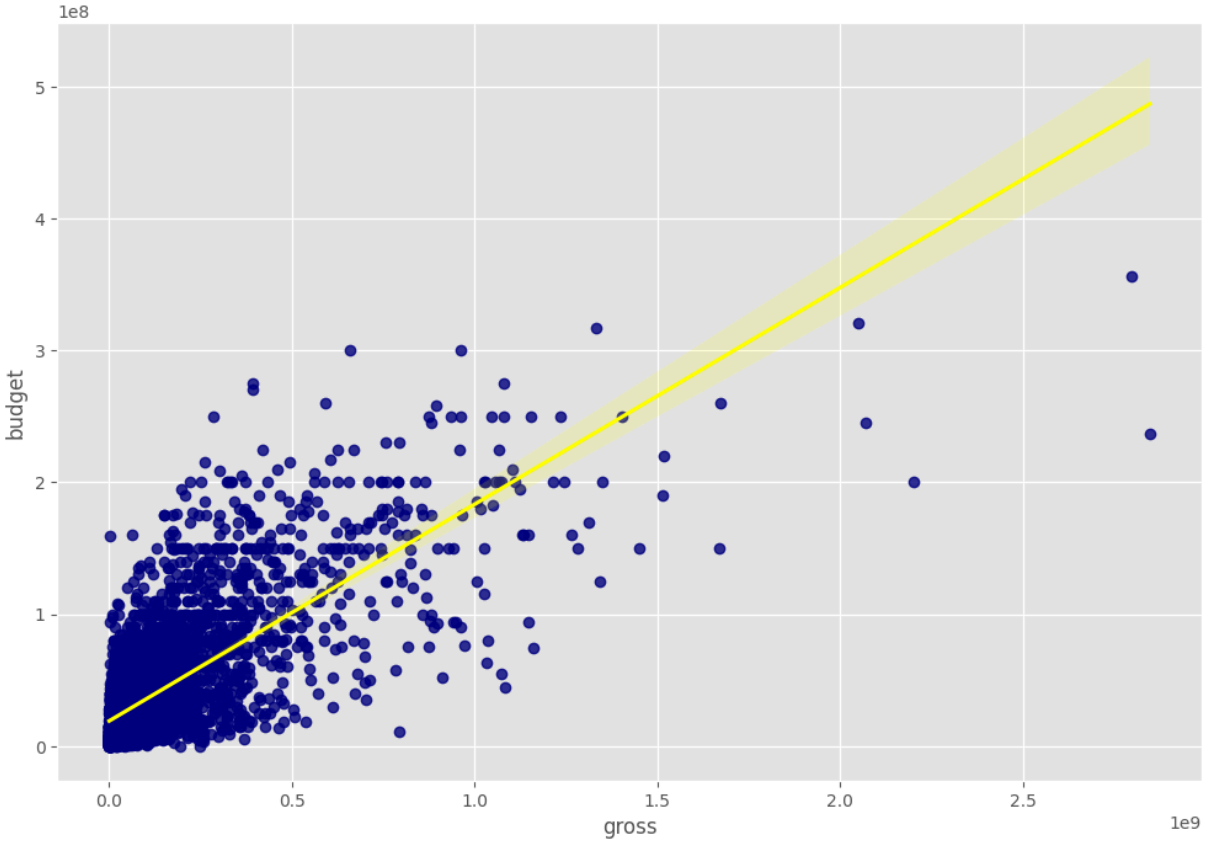
In [43]:
```
df.head()
```

Out[43]:

	name	rating	genre	released	score	votes	director	writer	s
5445	Avatar	PG-13	Action	December 18, 2009 (United States)	7.8	1100000	James Cameron	James Cameron	S Worthing
7445	Avengers: Endgame	PG-13	Action	April 26, 2019 (United States)	8.4	903000	Anthony Russo	Christopher Markus	Rob Downey
3045	Titanic	PG-13	Drama	December 19, 1997 (United States)	7.8	1100000	James Cameron	James Cameron	Leonar DiCap
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	December 18, 2015 (United States)	7.8	876000	J.J. Abrams	Lawrence Kasdan	Daisy Rid
7244	Avengers: Infinity War	PG-13	Action	April 27, 2018 (United States)	8.4	897000	Anthony Russo	Christopher Markus	Rob Downey

In [44]:
```
# Plot the budget vs gross using regression from seaborn library

sns.regplot(x='gross', y='budget', data=df, scatter_kws={"color": "#000080"}, line_
plt.show
```

Out[44]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

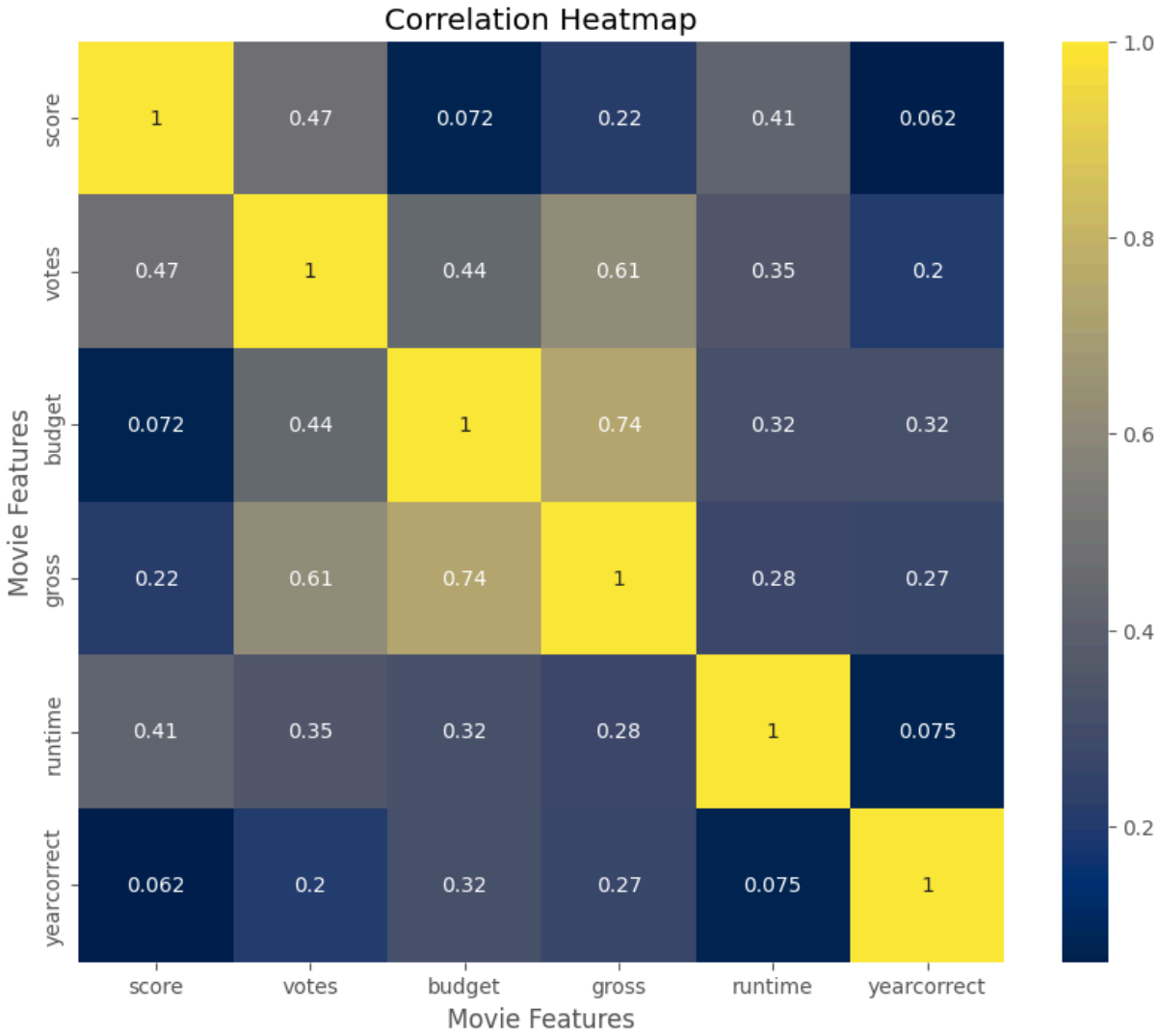


In [45]:
```
# Exploring the different methods to calculate the correlation

df.corr(numeric_only=True, method='pearson')
```

Out[45]:

	score	votes	budget	gross	runtime	yearcorrect
score	1.000000	0.474256	0.072001	0.222556	0.414068	0.061923
votes	0.474256	1.000000	0.439675	0.614751	0.352303	0.203098
budget	0.072001	0.439675	1.000000	0.740247	0.318695	0.320312
gross	0.222556	0.614751	0.740247	1.000000	0.275796	0.268721
runtime	0.414068	0.352303	0.318695	0.275796	1.000000	0.075294
yearcorrect	0.061923	0.203098	0.320312	0.268721	0.075294	1.000000

In [46]:
```
correlation_matrix = df.corr(numeric_only=True, method='pearson')

# Create heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='cividis')
plt.title('Correlation Heatmap')
plt.xlabel('Movie Features')
plt.ylabel('Movie Features')
plt.show()
```



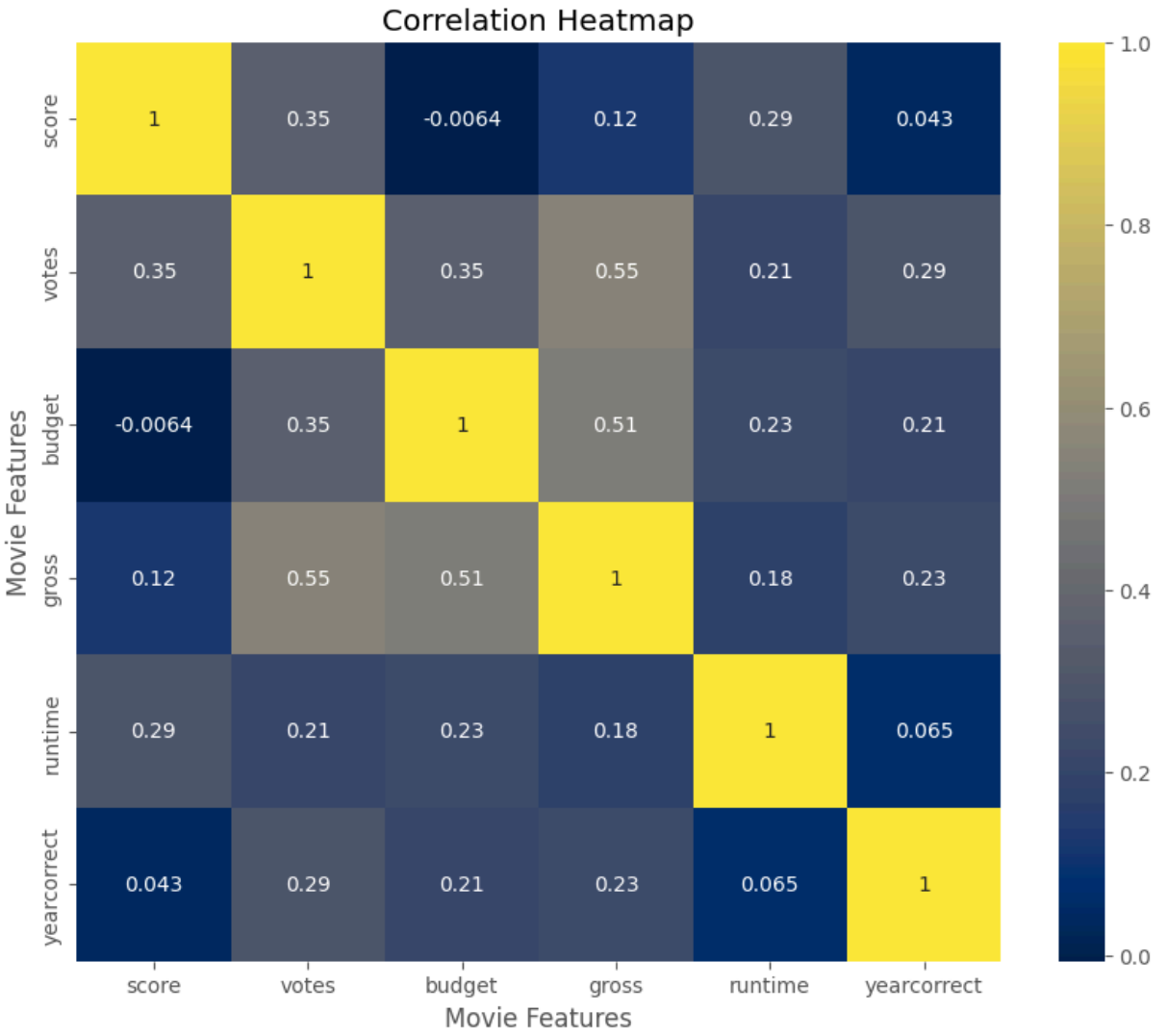
In [47]:
```
df.corr(numeric_only=True, method='kendall')
```

Out[47]:

	score	votes	budget	gross	runtime	yearcorrect
score	1.000000	0.350185	-0.006406	0.124943	0.292254	0.043400
votes	0.350185	1.000000	0.346274	0.553625	0.205344	0.293044
budget	-0.006406	0.346274	1.000000	0.512057	0.231278	0.213719
gross	0.124943	0.553625	0.512057	1.000000	0.176979	0.232372
runtime	0.292254	0.205344	0.231278	0.176979	1.000000	0.064793
yearcorrect	0.043400	0.293044	0.213719	0.232372	0.064793	1.000000

In [48]:
```
correlation_matrix = df.corr(numeric_only=True, method='kendall')

# Create heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='cividis')
plt.title('Correlation Heatmap')
plt.xlabel('Movie Features')
```

```
plt.ylabel('Movie Features')
plt.show()
```



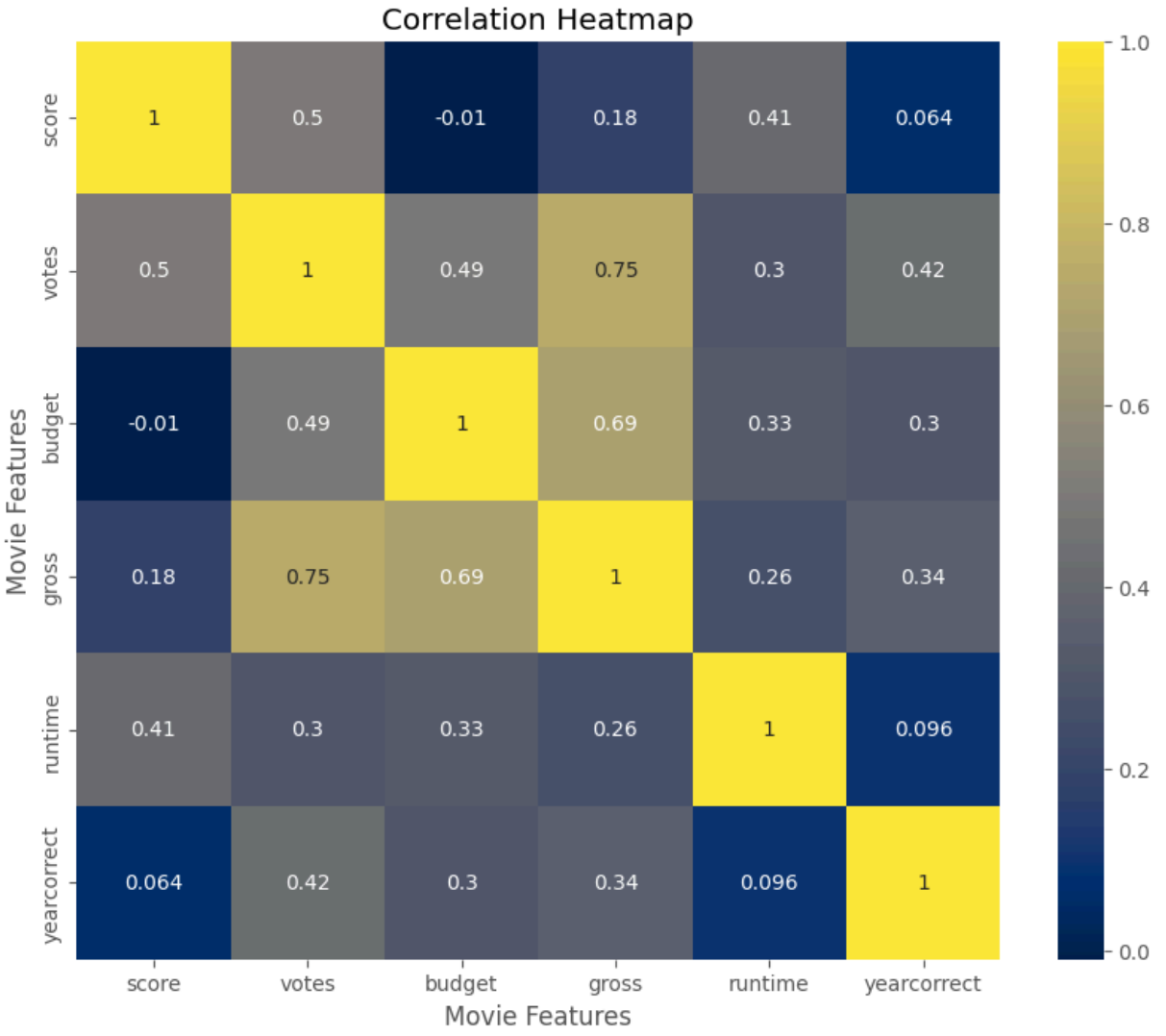
In [49]:
```
df.corr(numeric_only=True, method='spearman')
```

Out[49]:

	score	votes	budget	gross	runtime	yearcorrect
score	1.000000	0.495409	-0.009971	0.183192	0.412155	0.063674
votes	0.495409	1.000000	0.493461	0.745793	0.300621	0.422988
budget	-0.009971	0.493461	1.000000	0.692958	0.330794	0.302535
gross	0.183192	0.745793	0.692958	1.000000	0.257400	0.340529
runtime	0.412155	0.300621	0.330794	0.257400	1.000000	0.095507
yearcorrect	0.063674	0.422988	0.302535	0.340529	0.095507	1.000000

In [50]:
```
correlation_matrix = df.corr(numeric_only=True, method='spearman')

# Create heatmap
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='cividis')
plt.title('Correlation Heatmap')
plt.xlabel('Movie Features')
plt.ylabel('Movie Features')
plt.show()
```



In [51]:
```
df.head()
```

Out[51]:

	name	rating	genre	released	score	votes	director	writer	s
5445	Avatar	PG-13	Action	December 18, 2009 (United States)	7.8	1100000	James Cameron	James Cameron	S Worthing
7445	Avengers: Endgame	PG-13	Action	April 26, 2019 (United States)	8.4	903000	Anthony Russo	Christopher Markus	Rob Downey
3045	Titanic	PG-13	Drama	December 19, 1997 (United States)	7.8	1100000	James Cameron	James Cameron	Leonar DiCap
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	December 18, 2015 (United States)	7.8	876000	J.J. Abrams	Lawrence Kasdan	Daisy Rid
7244	Avengers: Infinity War	PG-13	Action	April 27, 2018 (United States)	8.4	897000	Anthony Russo	Christopher Markus	Rob Downey

In [52]:

```
# Converting Categorical Variables to Numerical Representation

df_numerized = df.copy()  # Make a copy of the DataFrame

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
```

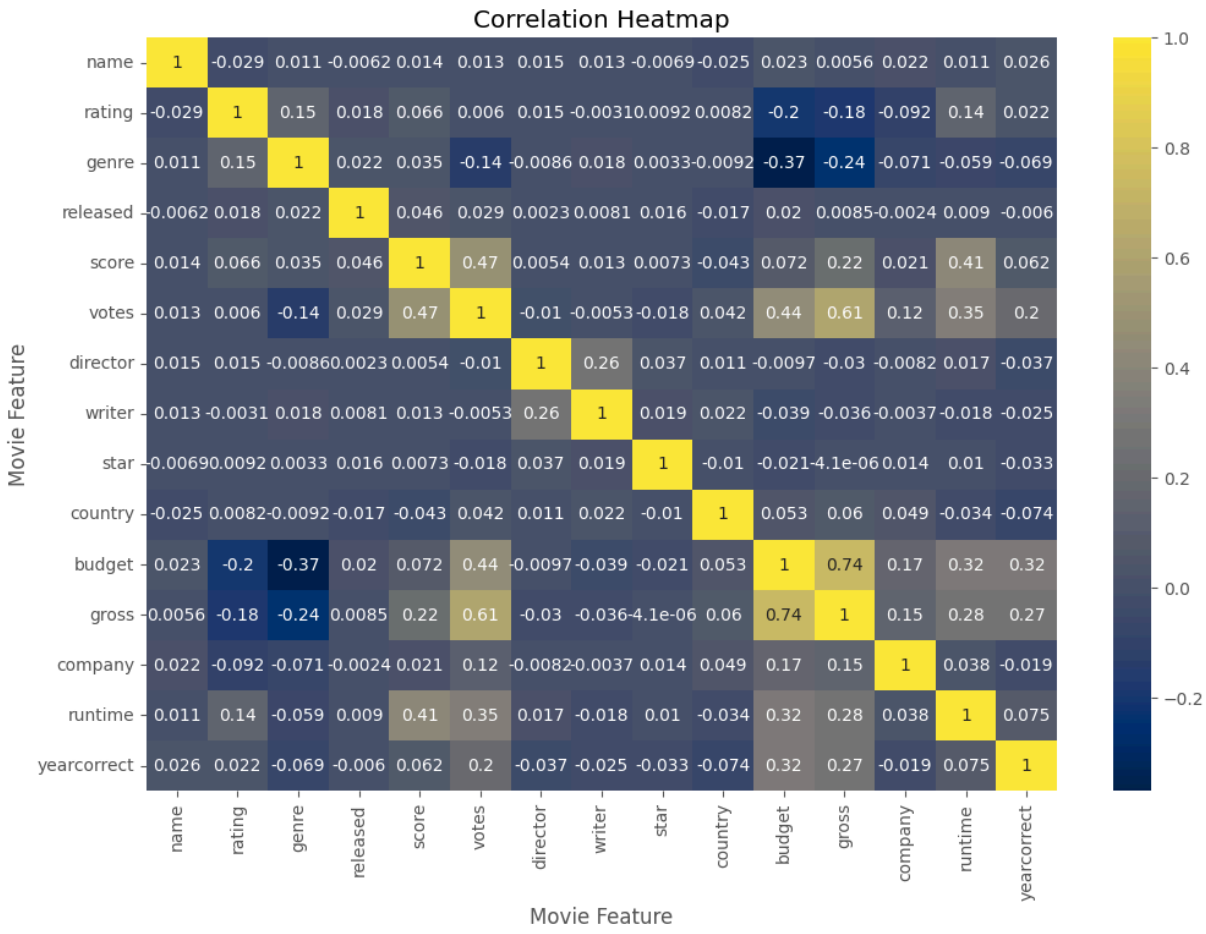
Out[52]:

	name	rating	genre	released	score	votes	director	writer	star	country	
5445	386	5	0	527	7.8	1100000	785	1263	1534	47	237
7445	388	5	0	137	8.4	903000	105	513	1470	47	356
3045	4909	5	6	534	7.8	1100000	785	1263	1073	47	200
6663	3643	5	0	529	7.8	876000	768	1806	356	47	245
7244	389	5	0	145	8.4	897000	105	513	1470	47	321
...	...	...	...	...	...	...	...	...	...	...	
5640	3794	6	6	890	5.8	3500	585	2924	1498	47	3
2434	2969	5	0	1467	4.5	1900	1805	3102	186	47	5
3681	1595	3	6	1721	6.8	43000	952	1683	527	6	5
272	2909	6	9	1525	3.9	2300	261	55	1473	47	
3203	4966	5	4	2152	5.7	5800	651	161	1811	47	15

5421 rows × 15 columns

In [53]:

```
# Correlation Heatmap of Movie Features

correlation_matrix = df_numerized.corr()


sns.heatmap(correlation_matrix, annot=True, cmap='cividis')
plt.title('Correlation Heatmap')
plt.xlabel('Movie Feature')
plt.ylabel('Movie Feature')
plt.show()
```



In [54]:
```
df_numerized.corr()
```

Out[54]:

	name	rating	genre	released	score	votes	director	
name	1.000000	-0.029234	0.010996	-0.006152	0.014450	0.012615	0.015246	0.0
rating	-0.029234	1.000000	0.147796	0.018083	0.065983	0.006031	0.014656	-0.0
genre	0.010996	0.147796	1.000000	0.022142	0.035106	-0.135990	-0.008553	0.0
released	-0.006152	0.018083	0.022142	1.000000	0.045874	0.028833	0.002308	0.0
score	0.014450	0.065983	0.035106	0.045874	1.000000	0.474256	0.005413	0.0
votes	0.012615	0.006031	-0.135990	0.028833	0.474256	1.000000	-0.010376	-0.0
director	0.015246	0.014656	-0.008553	0.002308	0.005413	-0.010376	1.000000	0.2
writer	0.012880	-0.003149	0.017578	0.008072	0.012843	-0.005316	0.261735	1.0
star	-0.006882	0.009196	0.003341	0.015706	0.007296	-0.017638	0.036593	0.0
country	-0.025490	0.008230	-0.009164	-0.017228	-0.043051	0.041551	0.011133	0.0
budget	0.023392	-0.203946	-0.368523	0.019952	0.072001	0.439675	-0.009662	-0.0
gross	0.005639	-0.181906	-0.244101	0.008501	0.222556	0.614751	-0.029560	-0.0
company	0.021697	-0.092357	-0.071334	-0.002407	0.020656	0.118470	-0.008223	-0.0
runtime	0.010850	0.140792	-0.059237	0.008975	0.414068	0.352303	0.017433	-0.0
yearcorrect	0.025542	0.022021	-0.069147	-0.005989	0.061923	0.203098	-0.037371	-0.0

In [55]:
```
# Let's do Pairwise Correlation between Numerical Variables

correlation_mat = df_numerized.corr()
corr_pairs = correlation_mat.unstack()
corr_pairs
```

Out[55]:
```
name         name           1.000000
             rating        -0.029234
             genre          0.010996
             released      -0.006152
             score          0.014450
                             ...
yearcorrect  budget         0.320312
             gross          0.268721
             company       -0.018806
             runtime        0.075294
             yearcorrect    1.000000
Length: 225, dtype: float64
```

In [56]:
```
# Sorted Pairwise Correlation between Numerical Variables

sorted_pairs = corr_pairs.sort_values()

print(sorted_pairs)
```

```
budget       genre         -0.368523
genre        budget        -0.368523
             gross         -0.244101
gross        genre         -0.244101
rating       budget        -0.203946
                              ...
released     released       1.000000
genre        genre          1.000000
rating       rating         1.000000
runtime      runtime        1.000000
yearcorrect  yearcorrect    1.000000
Length: 225, dtype: float64
```

In [57]:
```
# Strong pair correlation

strong_pairs = sorted_pairs[sorted_pairs > 0.5]

strong_pairs
```

Out[57]:
```
votes        gross          0.614751
gross        votes          0.614751
budget       gross          0.740247
gross        budget         0.740247
name         name           1.000000
writer       writer         1.000000
company      company        1.000000
gross        gross          1.000000
budget       budget         1.000000
country      country        1.000000
star         star           1.000000
director     director       1.000000
votes        votes          1.000000
score        score          1.000000
released     released       1.000000
genre        genre          1.000000
rating       rating         1.000000
runtime      runtime        1.000000
yearcorrect  yearcorrect    1.000000
dtype: float64
```

In [58]:
```
# Gross and Votes have the highest correlation to gross earning
# Company has the lowest correlation
```

In [59]:
```
# Now, Let's look at the top 15 companies by gross revenue

CompanyGrossSum = df.groupby('company')[["gross"]].sum()
CompanyGrossSumSorted = CompanyGrossSum.sort_values('gross', ascending = False)[:15
CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')
CompanyGrossSumSorted
```

Out[59]:
```
company
Warner Bros.                 54610959970
Universal Pictures           51241105418
Columbia Pictures            42356430218
Paramount Pictures           40021704691
Twentieth Century Fox        39542573303
Walt Disney Pictures         35833650748
New Line Cinema              19612851164
Marvel Studios               15065592411
DreamWorks Animation         11873612858
Dreamworks Pictures          11593807697
Touchstone Pictures          10664679494
Metro-Goldwyn-Mayer (MGM)     8937010092
Summit Entertainment          8318570396
Pixar Animation Studios       7886344526
Fox 2000 Pictures             7243673721
Name: gross, dtype: int64
```

In [60]:
```
df['yearcorrected'] = df['released'].astype(str).str[:4]
df
```

Out[60]:

	name	rating	genre	released	score	votes	director	writer	
5445	Avatar	PG-13	Action	December 18, 2009 (United States)	7.8	1100000	James Cameron	James Cameron	Wort
7445	Avengers: Endgame	PG-13	Action	April 26, 2019 (United States)	8.4	903000	Anthony Russo	Christopher Markus	Do
3045	Titanic	PG-13	Drama	December 19, 1997 (United States)	7.8	1100000	James Cameron	James Cameron	L
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	December 18, 2015 (United States)	7.8	876000	J.J. Abrams	Lawrence Kasdan	Dais
7244	Avengers: Infinity War	PG-13	Action	April 27, 2018 (United States)	8.4	897000	Anthony Russo	Christopher Markus	Do
...	...	...	...	...	...	...	...	...	
5640	Tanner Hall	R	Drama	January 15, 2015 (Sweden)	5.8	3500	Francesca Gregorini	Tatiana von Fürstenberg	
2434	Philadelphia Experiment II	PG-13	Action	June 4, 1994 (South Korea)	4.5	1900	Stephen Cornwell	Wallace C. Bennett	
3681	Ginger Snaps	Not Rated	Drama	May 11, 2001 (Canada)	6.8	43000	John Fawcett	Karen Walton	
272	Parasite	R	Horror	March 12, 1982 (United States)	3.9	2300	Charles Band	Alan J. Adler	
3203	Trojan War	PG-13	Comedy	October 1, 1997 (Brazil)	5.7	5800	George Huang	Andy Burg	Wi

5421 rows × 16 columns

In [61]:
```
df.groupby(['company', 'yearcorrected'])[["gross"]].sum()
```

Out[61]:

		gross
company	yearcorrected	
"DIA" Productions GmbH & Co. KG	Apri	44350926
"Weathering With You" Film Partners	Janu	193457467
.406 Production	Apri	10580
1492 Pictures	Dece	87423861
	Nove	129832389
...	...	...
erbp	Augu	587174
i am OTHER	June	17986781
i5 Films	Augu	10031529
micro_scope	Janu	7099598
thefyzz	June	62198461

2613 rows × 1 columns

In [62]:
```
# Let's look at the top 15 top 15 combinations of company and year
# based on the sum of 'gross' earnings

CompanyGrossSum = df.groupby(['company', 'yearcorrected'])[["gross"]].sum()
CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company','yearcorrect
CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')
CompanyGrossSumSorted
```

Out[62]:
```
company               yearcorrected
Universal Pictures    June            10039199978
Columbia Pictures     July             9763253735
Warner Bros.          July             9240468686
Twentieth Century Fox Dece             8845278237
Warner Bros.          June             8384558528
Universal Pictures    July             8356021321
Paramount Pictures    May              8226992512
Walt Disney Pictures  June             7758208901
Warner Bros.          Dece             7465401978
Paramount Pictures    June             7003735344
Columbia Pictures     Dece             6278736410
Walt Disney Pictures  Nove             6181450855
Marvel Studios        May              6153529854
Twentieth Century Fox July             6105415541
Warner Bros.          Marc             6073696584
Name: gross, dtype: int64
```

In [63]:
```
# Let's take the top 15 companies with the highest sum of 'gross' earnings,
# and converts the earnings to a specific data type
```

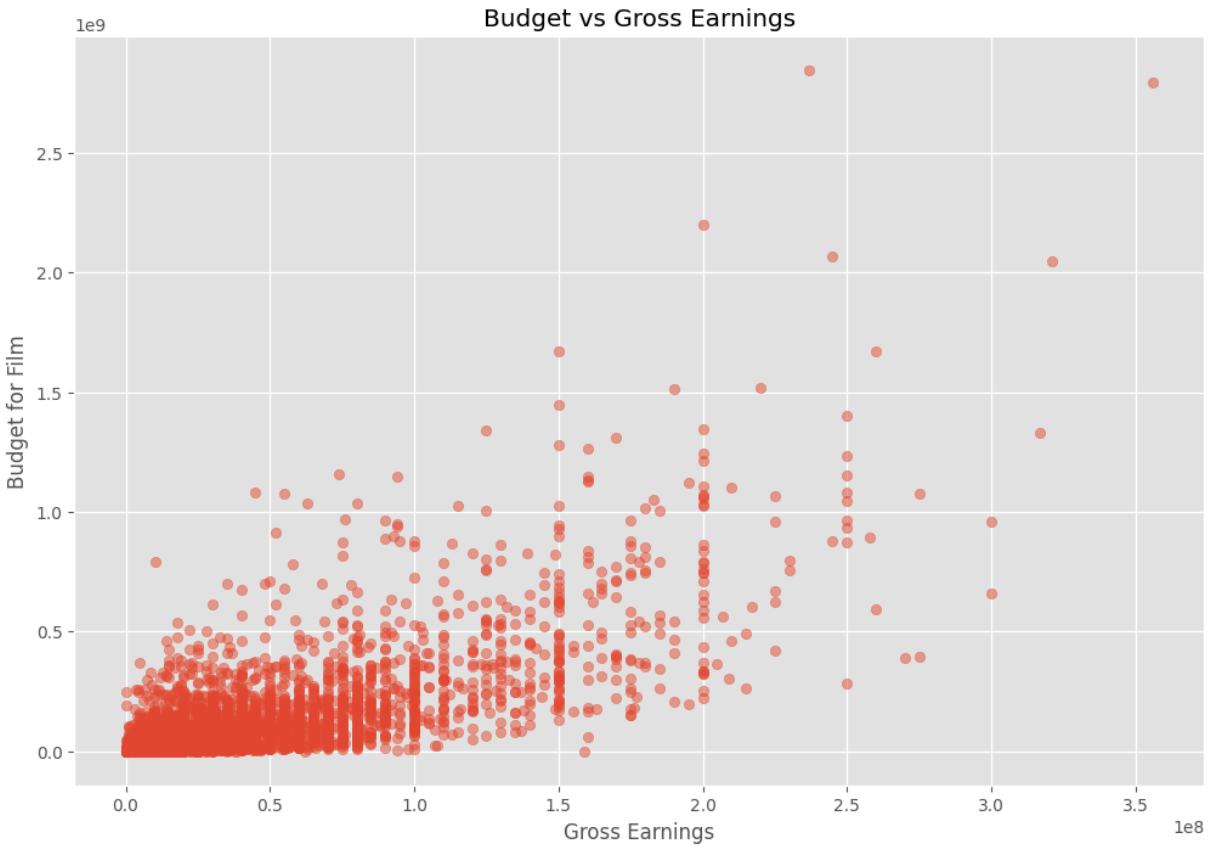
```
CompanyGrossSum = df.groupby(['company'])[["gross"]].sum()
CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company'], ascending
CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')
CompanyGrossSumSorted
```

Out[63]:

```
company
Warner Bros.                 54610959970
Universal Pictures           51241105418
Columbia Pictures            42356430218
Paramount Pictures           40021704691
Twentieth Century Fox        39542573303
Walt Disney Pictures         35833650748
New Line Cinema              19612851164
Marvel Studios               15065592411
DreamWorks Animation         11873612858
Dreamworks Pictures          11593807697
Touchstone Pictures          10664679494
Metro-Goldwyn-Mayer (MGM)     8937010092
Summit Entertainment          8318570396
Pixar Animation Studios       7886344526
Fox 2000 Pictures             7243673721
Name: gross, dtype: int64
```

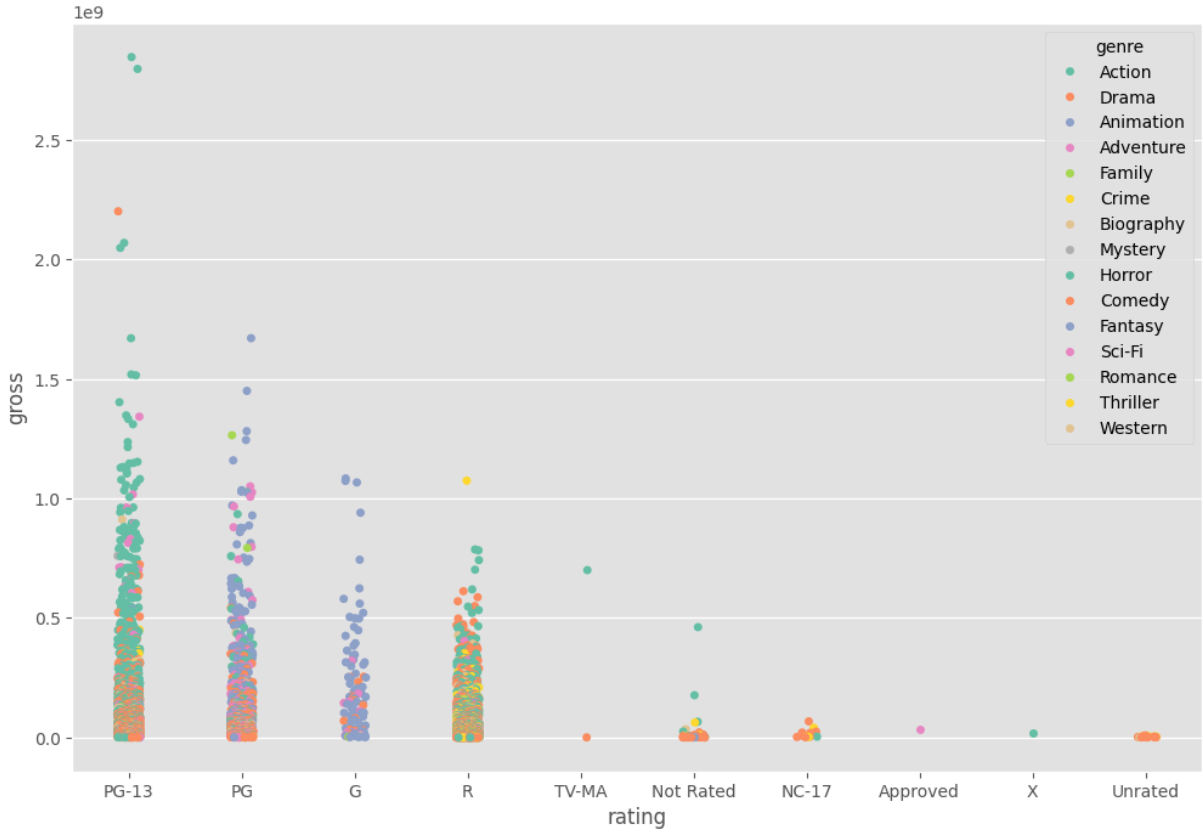
In [64]:

```
# Generating a scatter plot to explore the relationship
#between the budget allocated for a film and its gross earnings.

plt.scatter(x=df['budget'], y=df['gross'], alpha=0.5)
plt.title('Budget vs Gross Earnings')
plt.xlabel('Gross Earnings')
plt.ylabel('Budget for Film')
plt.show()
```



In [65]:
```
# Let's Explore and create a strip plot to visualize the relationship between the r

sns.stripplot(x="rating", y="gross", hue= "genre", palette="Set2", data=df)
```

Out[65]:
```
<Axes: xlabel='rating', ylabel='gross'>
```



In [66]:
```
# Yipeee, I did it!! You can explore more combinations.... Go give it a try...
```