In [3]:
```
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  #to ignore files are not original and are copy slice

df=pd.read_csv(r'D:\Analyst\Projects Portfolio\Python\Correlation Movies\archive (1)\movies.csv')
```

In [16]:
```
df.head()
```

Out[16]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	compan
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warn Bro
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columb Picture
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	Mark Hamill	United States	18000000.0	538375067.0	Lucasfil
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramou Picture
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Oric Picture

In [9]:
```
#data missing
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col,pct_missing))
```

```
name - 0.0%
rating - 0.010041731872717789%
genre - 0.0%
year - 0.0%
released - 0.00026082420448611763%
score - 0.00039123630672926445%
votes - 0.00039123630672926445%
director - 0.0%
writer - 0.00039123630672926445%
star - 0.00013041210224308815%
country - 0.00039123630672926445%
budget - 0.2831246739697444%
gross - 0.02464788732394366%
company - 0.002217005738132499%
runtime - 0.00052164840897235265%
```

In [18]:
```
#data types for columns
df.dtypes
```

Out[18]:
```
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 [4]:
```
df.sort_values(by=['gross'],inplace=False,ascending=False)
```

Out[4]:

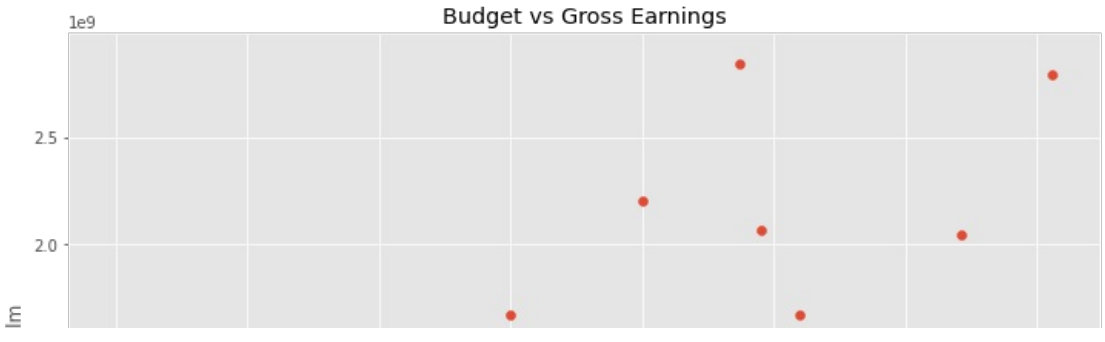
	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	
5445	Avatar	PG-13	Action	2009	December 18, 2009 (United States)	7.8	1100000.0	James Cameron	James Cameron	Sam Worthington	United States	237000000.0	2.847246e+09	
7445	Avengers: Endgame	PG-13	Action	2019	April 26, 2019 (United States)	8.4	903000.0	Anthony Russo	Christopher Markus	Robert Downey Jr.	United States	356000000.0	2.797501e+09	
3045	Titanic	PG-13	Drama	1997	December 19, 1997 (United States)	7.8	1100000.0	James Cameron	James Cameron	Leonardo DiCaprio	United States	200000000.0	2.201647e+09	
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 Ridley	United States	245000000.0	2.069522e+09	
7244	Avengers: Infinity War	PG-13	Action	2018	April 27, 2018 (United States)	8.4	897000.0	Anthony Russo	Christopher Markus	Robert Downey Jr.	United States	321000000.0	2.048360e+09	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7663	More to Life	NaN	Drama	2020	October 23, 2020 (United States)	3.1	18.0	Joseph Ebanks	Joseph Ebanks	Shannon Bond	United States	7000.0	NaN	
7664	Dream Round	NaN	Comedy	2020	February 7, 2020 (United States)	4.7	36.0	Dusty Dukatz	Lisa Huston	Michael Saquella	United States	NaN	NaN	
7665	Saving Mbango	NaN	Drama	2020	April 27, 2020 (Cameroon)	5.7	29.0	Nkanya Nkwai	Lynno Lovert	Onyama Laura	United States	58750.0	NaN	
7666	It's Just Us	NaN	Drama	2020	October 1, 2020 (United States)	NaN	NaN	James Randall	James Randall	Christina Roz	United States	15000.0	NaN	
7667	Tee em el	NaN	Horror	2020	August 19, 2020 (United States)	5.7	7.0	Pereko Mosia	Pereko Mosia	Siyabonga Mabaso	South Africa	NaN	NaN	

7668 rows × 15 columns

In [1]:
```
pd.set_option('display.max_rows',None) #for displaying all the table at the above code
```

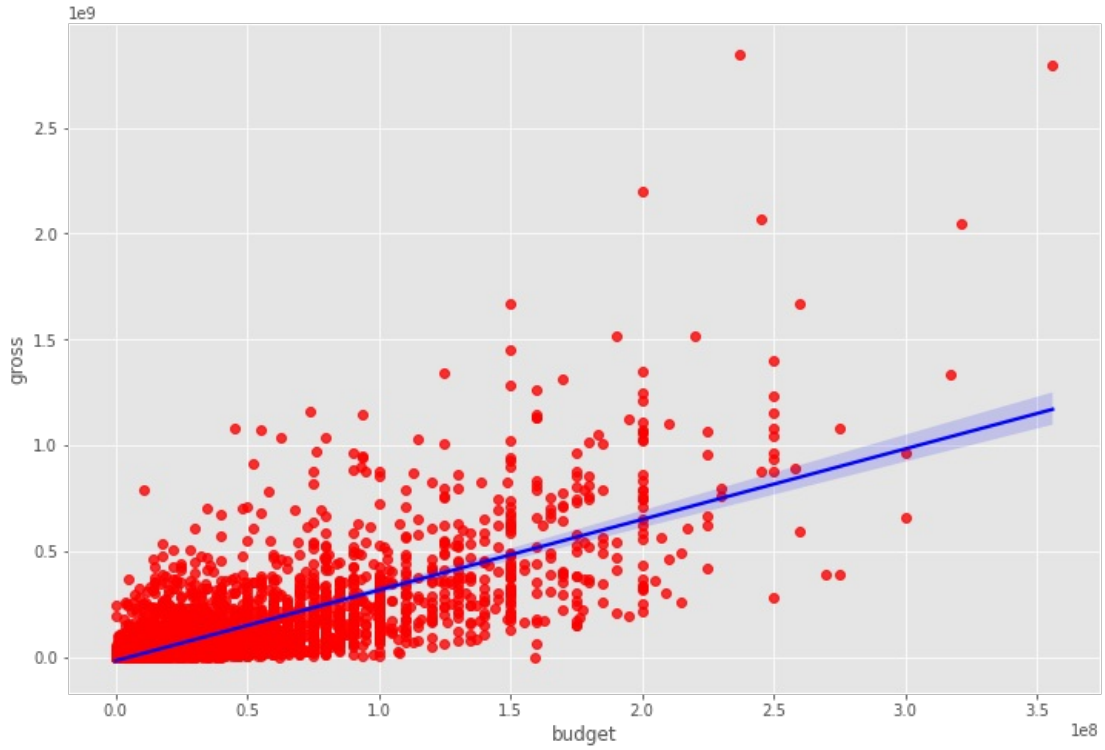
In [14]:
```
#Scatter plot with budget vs gross
plt.scatter(x=df['budget'], y=df['gross'])
plt.title('Budget vs Gross Earnings')
plt.xlabel('Gross Earnings')
plt.ylabel('Buget for Film')
```

Out[14]:
```
Text(0, 0.5, 'Buget for Film')
```



In [13]:
```
#Plot the budget vs Gross using seadorn
sns.regplot(x='budget', y='gross', data=df, scatter_kws={'color':'red'},line_kws={'color':'blue'})
```

Out[13]:
```
<AxesSubplot:xlabel='budget', ylabel='gross'>
```



In [14]:
```
#correlation is only between numerical data not strings
#by default is Pearson method
df.corr()
```

Out[14]:

	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 [18]:
```
df.corr(method='kendall')
```

Out[18]:

	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 [19]:
```
df.corr(method='spearman')
```

Out[19]:

	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

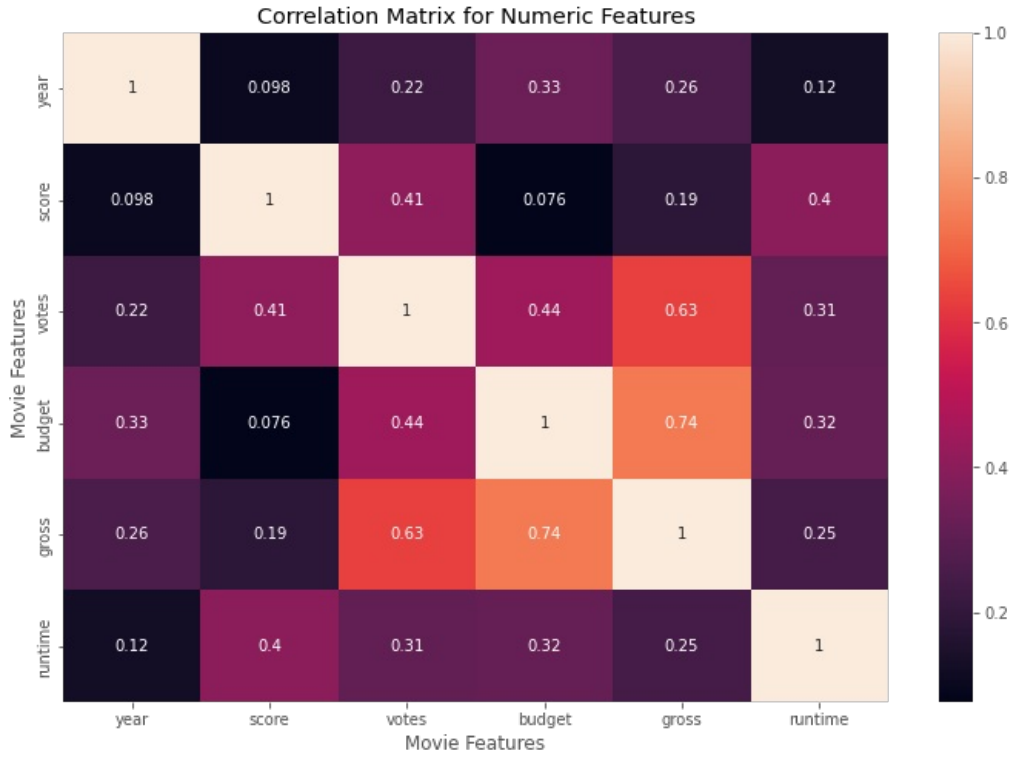
In [21]:
```
df.corr(method='pearson')
```

Out[21]:

	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 [23]:
```
correlation_matrix=df.corr(method='pearson')
sns.heatmap(correlation_matrix,annot=True)
plt.title('Correlation Matrix for Numeric Features')
plt.xlabel('Movie Features')
plt.ylabel('Movie Features')
```

Out[23]:
```
Text(87.0, 0.5, 'Movie Features')
```



In [21]:
```
# Using factorize - this assigns a random numeric value for each unique categorical value
```

In [20]:
```
df.apply(lambda x: x.factorize()[0]).corr(method='pearson')
```

Out[20]:

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	g
name	1.000000	0.143938	0.036367	0.965761	0.959015	-0.046733	0.287776	0.745905	0.805211	0.731565	0.142828	0.277488	0.94
rating	0.143938	1.000000	-0.086723	0.156713	0.146606	0.012595	0.099972	0.085520	0.103623	0.093116	0.000494	0.193353	0.15
genre	0.036367	-0.086723	1.000000	0.037184	0.035940	-0.002437	0.023285	0.047288	0.033688	0.038649	-0.015795	0.073008	0.03
year	0.965761	0.156713	0.037184	1.000000	0.993190	-0.044981	0.312401	0.770497	0.824770	0.756400	0.140216	0.300621	0.98
released	0.959015	0.146606	0.035940	0.993190	1.000000	-0.045761	0.299905	0.770876	0.819617	0.754468	0.148468	0.285691	0.97
score	-0.046733	0.012595	-0.002437	-0.044981	-0.045761	1.000000	-0.009749	-0.022687	-0.034685	-0.009896	0.023097	-0.012642	-0.04
votes	0.287776	0.099972	0.023285	0.312401	0.299905	-0.009749	1.000000	0.192220	0.224122	0.179601	-0.045914	0.398519	0.28
director	0.745905	0.085520	0.047288	0.770497	0.770876	-0.022687	0.192220	1.000000	0.748340	0.682385	0.155471	0.106617	0.75
writer	0.805211	0.103623	0.033688	0.824770	0.819617	-0.034685	0.224122	0.748340	1.000000	0.675685	0.157202	0.187238	0.80
star	0.731565	0.093116	0.038649	0.756400	0.754468	-0.009896	0.179601	0.682385	0.675685	1.000000	0.182045	0.107991	0.73
country	0.142828	0.000494	-0.015795	0.140216	0.148468	0.023097	-0.045914	0.155471	0.157202	0.182045	1.000000	-0.082082	0.13
budget	0.277488	0.193353	0.073008	0.300621	0.285691	-0.012642	0.398519	0.106617	0.187238	0.107991	-0.082082	1.000000	0.28
gross	0.947324	0.158582	0.038616	0.980873	0.976423	-0.047041	0.286180	0.750911	0.805576	0.735680	0.133982	0.285832	1.00
company	0.591667	-0.028035	0.009566	0.601571	0.607954	-0.028432	0.008900	0.552258	0.546151	0.527116	0.226346	-0.092249	0.58
runtime	0.048955	0.032741	0.001462	0.050647	0.048235	0.026436	0.106024	-0.011070	0.032264	0.035392	0.124154	0.112097	0.04

In [22]:
```
correlation_mat = df.apply(lambda x: x.factorize()[0]).corr() #factorize() function can be used to encode strings

corr_pairs = correlation_mat.unstack() #unstack to reshape a data frame

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 [23]:
```
sorted_pairs = corr_pairs.sort_values(kind="quicksort") #quicksort in its general form is an in-place sort (i.e.

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 [26]:
```
#take a look at the ones that have a high correlation (> 0.5)

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
company   director    0.552258
gross     company     0.588156
company   gross       0.588156
          name        0.591667
name      company     0.591667
year      company     0.601571
company   year        0.601571
released  company     0.607954
company   released    0.607954
writer    star        0.675685
star      writer      0.675685
director  star        0.682385
star      director    0.682385
name      star        0.731565
star      name        0.731565
gross     star        0.735680
star      gross       0.735680
director  name        0.745905
name      director    0.745905
writer    director    0.748340
director  writer      0.748340
gross     director    0.750911
director  gross       0.750911
released  star        0.754468
star      released    0.754468
year      star        0.756400
star      year        0.756400
year      director    0.770497
director  year        0.770497
released  director    0.770876
director  released    0.770876
writer    name        0.805211
name      writer      0.805211
gross     writer      0.805576
writer    gross       0.805576
          released    0.819617
released  writer      0.819617
year      writer      0.824770
writer    year        0.824770
name      gross       0.947324
gross     name        0.947324
name      released    0.959015
released  name        0.959015
name      year        0.965761
year      name        0.965761
released  gross       0.976423
gross     released    0.976423
year      gross       0.980873
gross     year        0.980873
released  year        0.993190
year      released    0.993190
name      name        1.000000
director  director    1.000000
gross     gross       1.000000
budget    budget      1.000000
country   country     1.000000
star      star        1.000000
writer    writer      1.000000
votes     votes       1.000000
score     score       1.000000
released  released    1.000000
year      year        1.000000
genre     genre       1.000000
rating    rating      1.000000
company   company     1.000000
runtime   runtime     1.000000
dtype: float64
```

In [25]:
```
pd.set_option('display.max_rows',None) #for the above list
```

In [ ]:
```
# company has the highest correlation to gross earnings
```