In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from tqdm import tqdm
from ast import literal_eval
import numpy as np
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.externals import joblib
```

In [2]:

```
#reading train data
train= pd.read_csv('train.csv')

train.describe()
```

Out[2]:

	id	budget	popularity	runtime	revenue
count	3000.000000	3.000000e+03	3000.000000	2998.000000	3.000000e+03
mean	1500.500000	2.253133e+07	8.463274	107.856571	6.672585e+07
std	866.169729	3.702609e+07	12.104000	22.086434	1.375323e+08
min	1.000000	0.000000e+00	0.000001	0.000000	1.000000e+00
25%	750.750000	0.000000e+00	4.018053	94.000000	2.379808e+06
50%	1500.500000	8.000000e+06	7.374861	104.000000	1.680707e+07
75%	2250.250000	2.900000e+07	10.890983	118.000000	6.891920e+07
max	3000.000000	3.800000e+08	294.337037	338.000000	1.519558e+09

In [3]:

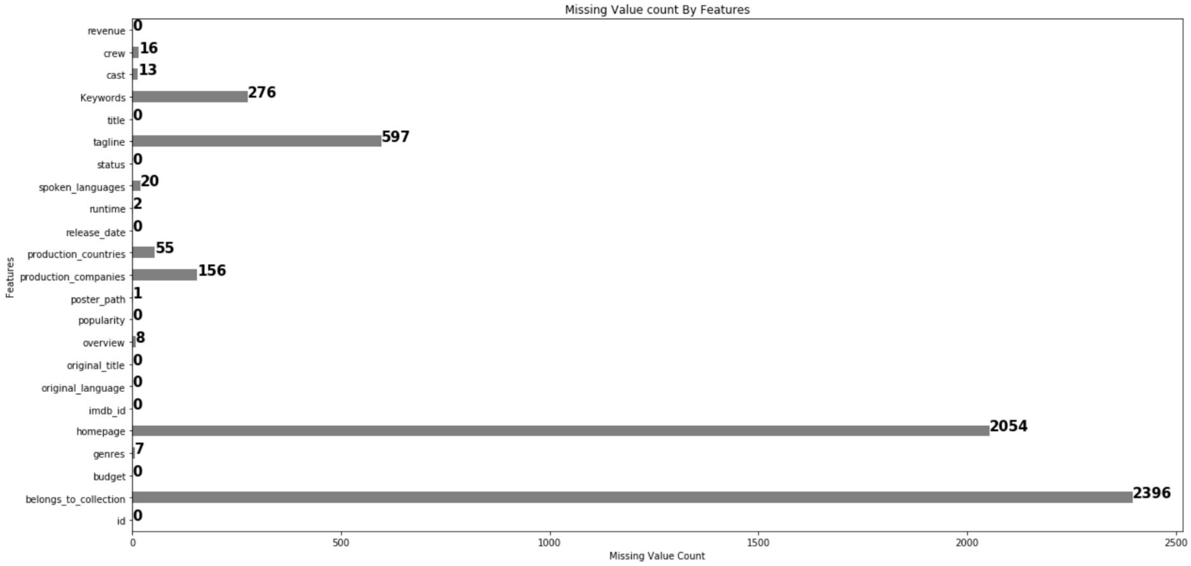
```
#reading test data
test= pd.read_csv('test.csv')

test.describe()
```

Out[3]:

	id	budget	popularity	runtime
count	4398.000000	4.398000e+03	4398.000000	4394.000000
mean	5199.500000	2.264929e+07	8.550230	107.622212
std	1269.737571	3.689991e+07	12.209014	21.058290
min	3001.000000	0.000000e+00	0.000001	0.000000
25%	4100.250000	0.000000e+00	3.895186	94.000000
50%	5199.500000	7.450000e+06	7.482241	104.000000
75%	6298.750000	2.800000e+07	10.938524	118.000000
max	7398.000000	2.600000e+08	547.488298	320.000000

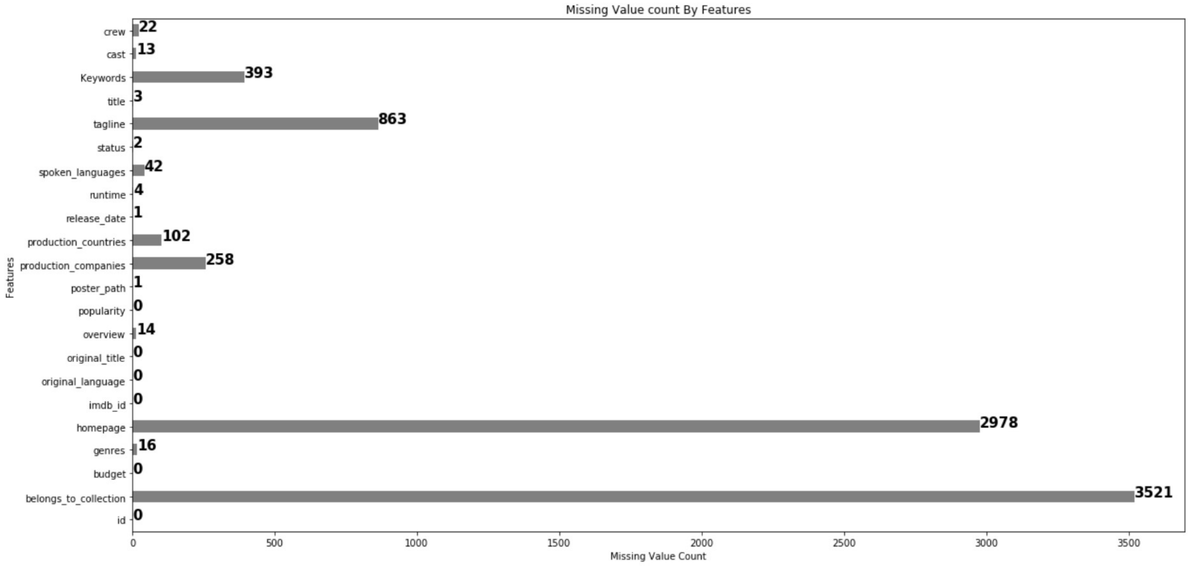
In [9]:
```python
#Counting Missing Value By Features
train.isna().sum().plot(kind="barh", figsize=(20,10),color='grey')
for i, v in enumerate(train.isna().sum()):
    plt.text(v, i, str(v), fontweight='bold', fontsize = 15)
plt.xlabel("Missing Value Count")
plt.ylabel("Features")
plt.title("Missing Value count By Features")
```

Out[9]:
```
Text(0.5, 1.0, 'Missing Value count By Features')
```



In [11]:
```python
#Counting Missing Value By Features
test.isna().sum().plot(kind="barh", figsize=(20,10),color='grey')
for i, v in enumerate(test.isna().sum()):
    plt.text(v, i, str(v), fontweight='bold', fontsize = 15)
plt.xlabel("Missing Value Count")
plt.ylabel("Features")
plt.title("Missing Value count By Features")
```

Out[11]:
```
Text(0.5, 1.0, 'Missing Value count By Features')
```



In [12]:
```
train.head()
```

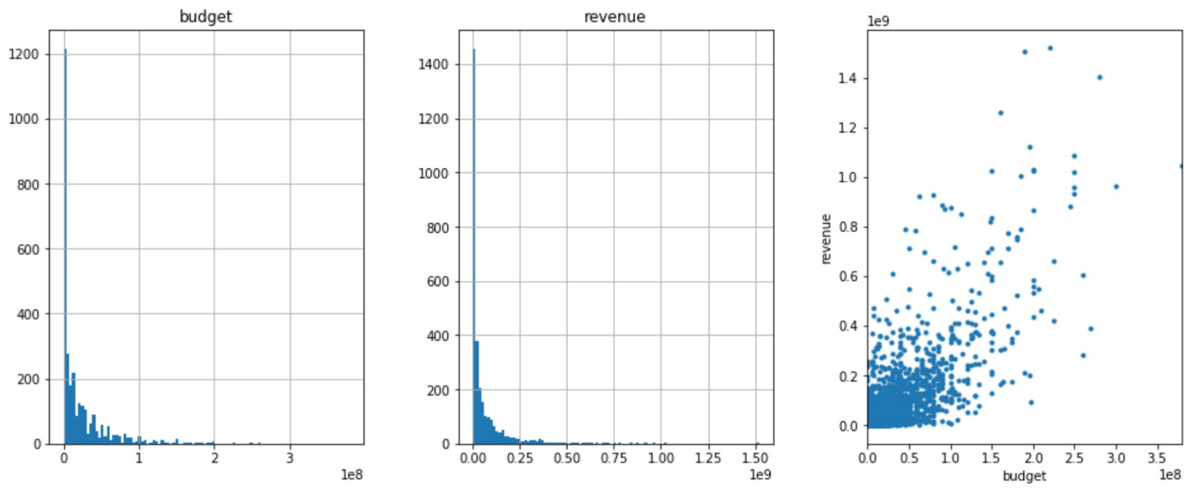
Out[12]:

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	origina
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	en	H Mac
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	en	The Pr Dia Engag
2	3	NaN	3300000	[{'id': 18, 'name': 'Drama'}]	http://sonyclassics.com/whiplash/	tt2582802	en	Wh
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com/	tt1821480	hi	Ka
4	5	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...	NaN	tt1380152	ko	마

5 rows × 23 columns

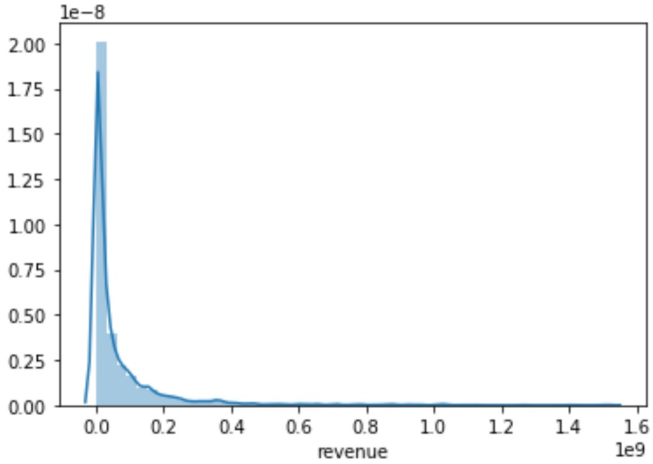
In [14]:
```
plt.figure(figsize=(16,6));
ax1 = plt.subplot(131)
train.hist('budget',bins=100,ax=ax1)
ax2 = plt.subplot(132)
train.hist('revenue',bins=100,ax=ax2)
ax3 = plt.subplot(133)
train.plot(x='budget',y='revenue',style='.',ax=ax3,legend=False)
plt.ylabel('revenue')
```

Out[14]:
```
Text(0, 0.5, 'revenue')
```



```
In [15]: sns.distplot(train['revenue'])
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x22b30567ac8>
```

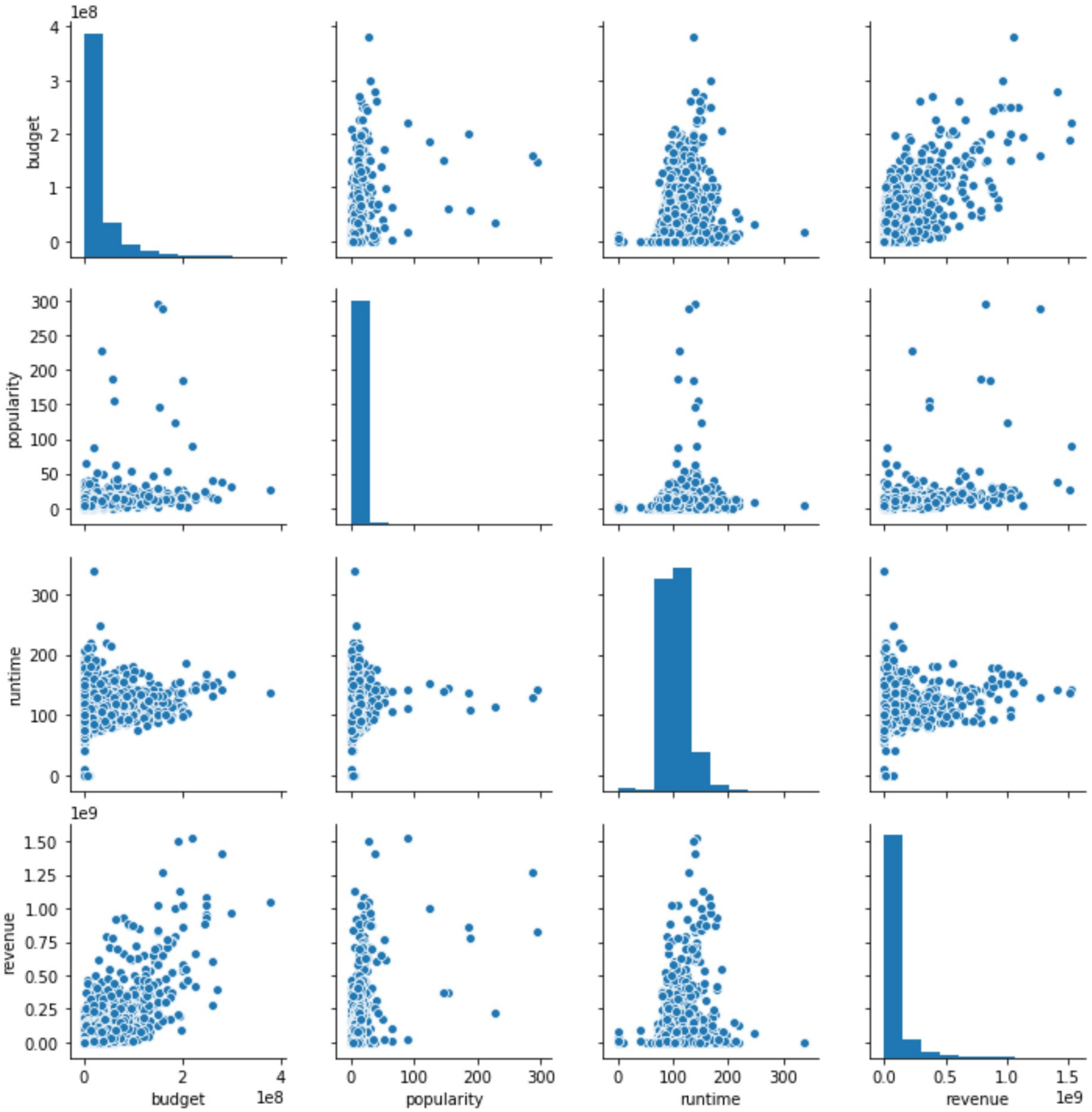


In [16]:

```
#pairplots
train_numer = train.select_dtypes(['number']).drop(['id'], axis=1).fillna(0)
sns.pairplot(train_numer)
```

Out[16]:

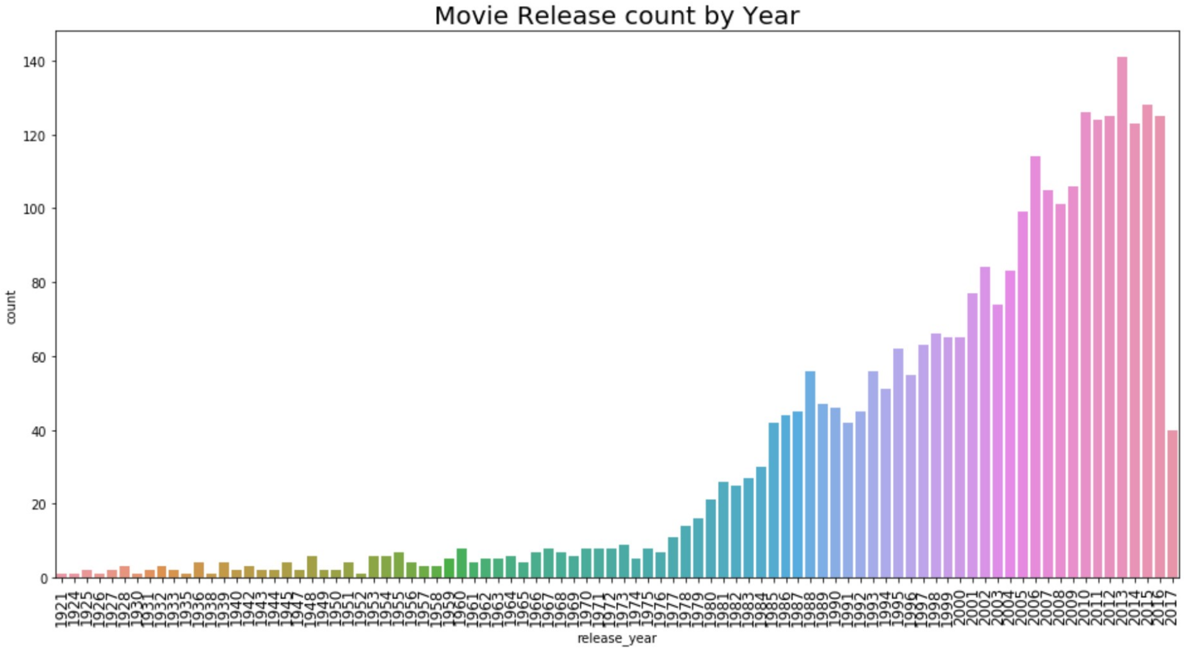
```
<seaborn.axisgrid.PairGrid at 0x22b33114908>
```



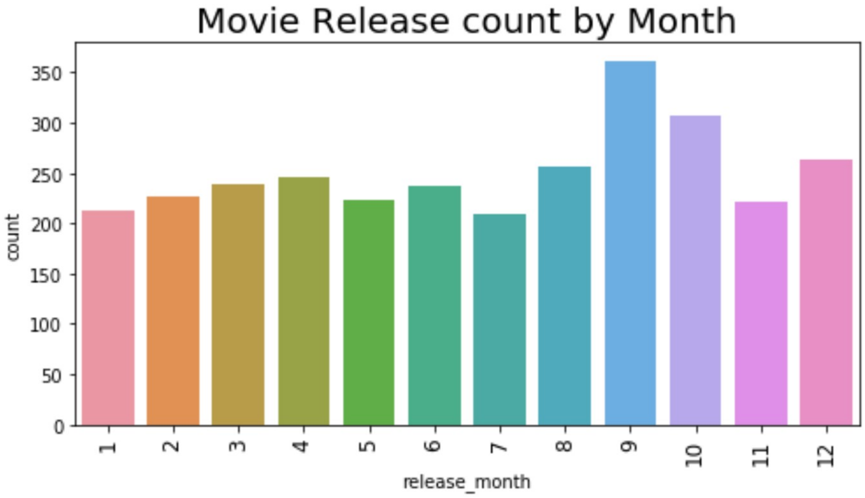
In [17]:

```
train['release_month'] = train.release_date.str.extract('(\S+)/\S+/\S+', expand=Fal
se).astype(np.int16)
train['release_year'] = train.release_date.str.extract('\S+/\S+/(\S+)', expand=Fals
e).astype(np.int16)
train['release_day'] = train.release_date.str.extract('\S+/(\S+)/\S+', expand=False
).astype(np.int16)
train.loc[(21 <= train.release_year) & (train.release_year <= 99), 'release_year']
+= 1900
train.loc[train.release_year < 21, 'release_year'] += 2000

train['release_date'] = pd.to_datetime(train.release_day.astype(str) + '-' +
                                       train.release_month.astype(str) + '-' +
                                       train.release_year.astype(str))

train['release_weekday'] = train.release_date.dt.weekday_name.str.slice(0, 3)
```

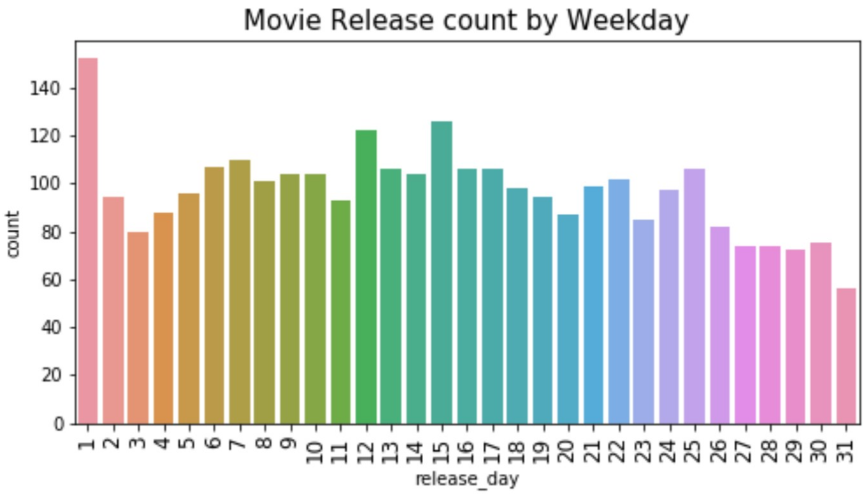
```
In [19]: #counting Movie Releases by  Year
         plt.figure(figsize=(16,8))
         sns.countplot(train['release_year'].sort_values())
         plt.title("Movie Release count by Year", fontsize=20)
         loc, labels = plt.xticks()
         plt.xticks(fontsize=12, rotation=90)
         plt.show()
```



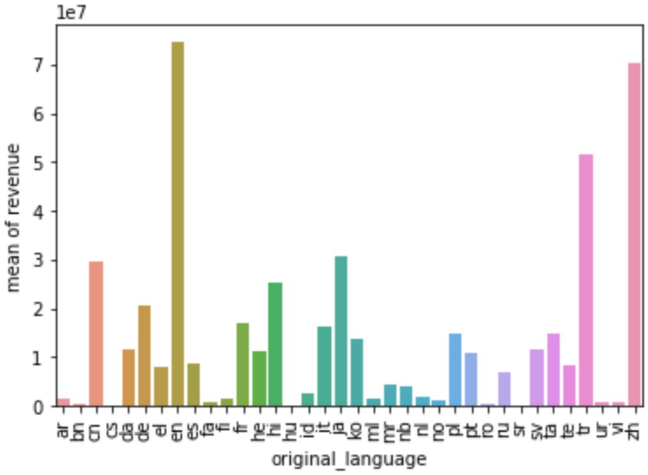
```
In [20]: #counting Movie Releases by  month
         plt.figure(figsize=(8,4))
         sns.countplot(train['release_month'].sort_values())
         plt.title("Movie Release count by Month", fontsize=20)
         loc, labels = plt.xticks()
         plt.xticks(fontsize=12, rotation=90)
         plt.show()
```



In [21]:
```
#counting Movie Releases by Weekday
plt.figure(figsize=(8,4))
sns.countplot(train['release_day'].sort_values())
plt.title("Movie Release count by Weekday", fontsize=15)
loc, labels = plt.xticks()
plt.xticks(fontsize=12, rotation=90)
plt.show()
```



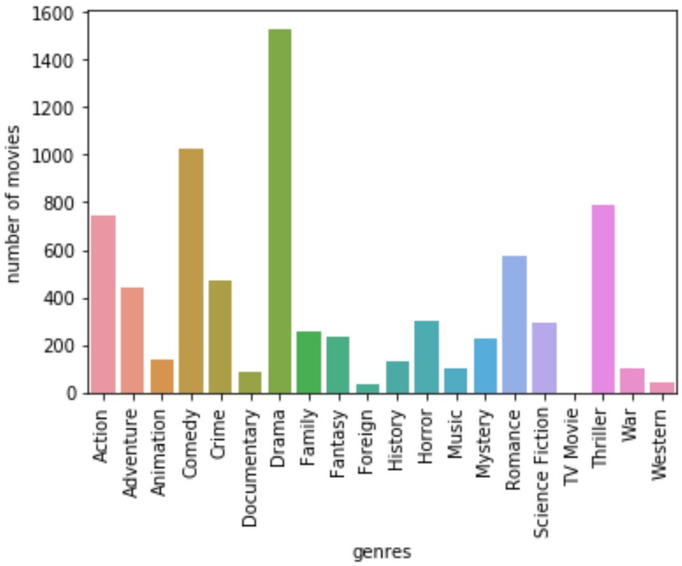
In [25]:
```
#mean of revenue across languages
revenue_by_lang = train.groupby('original_language')['revenue'].aggregate([np.mean]
)
revenue_by_lang.reset_index(inplace=True)
fig = sns.barplot(x='original_language', y='mean', data=revenue_by_lang)
fig.set(ylabel='mean of revenue')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
```



In [26]:
```
train.loc[train.genres.isnull(), 'genres'] = "{}"
train['genres'] = train.genres.apply(lambda x: sorted([d['name'] for d in eval(x)])
).apply(lambda x: ','.join(map(str, x)))
genres = train.genres.str.get_dummies(sep=',')
```

In [27]:

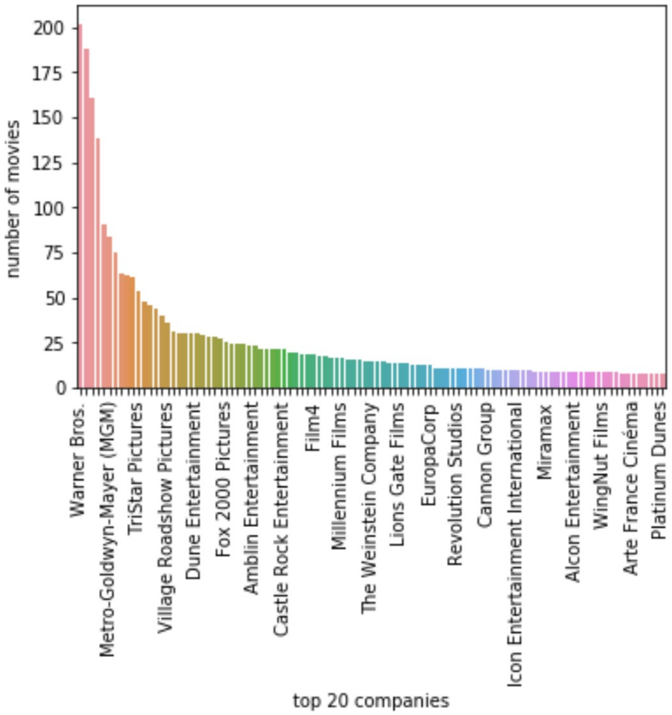
```
#number of movies across genres
movies_by_genre = pd.DataFrame(genres.sum(axis=0)).reset_index()
movies_by_genre.columns = ['genres', 'movies']
fig = sns.barplot(x='genres', y='movies', data=movies_by_genre)
fig.set(ylabel='number of movies')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
```



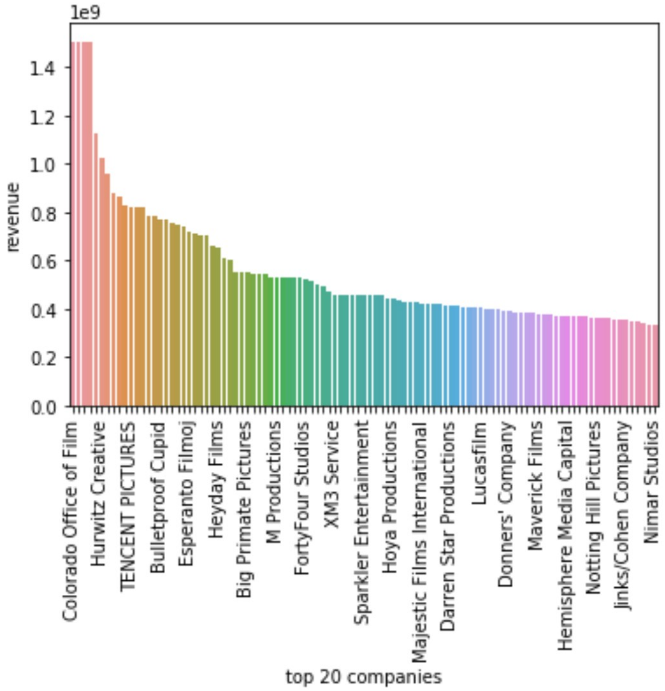
In [28]:

```
train.loc[train.production_companies.isnull(), 'production_companies'] = "{}"
train['production_companies'] = train.production_companies.apply(lambda x: sorted([
d['name'] for d in eval(x)])).apply(lambda x: ','.join(map(str, x)))
companies = train.production_companies.str.get_dummies(sep=',')
```

In [29]:
```
#number of movies by production companies
movies_by_companies = pd.DataFrame(companies.sum(axis=0)).reset_index()
movies_by_companies.columns = ['company', 'movies']
top_100_companies = movies_by_companies.sort_values(by='movies', ascending=False).r
eset_index().loc[0:100]
fig = sns.barplot(x='company', y='movies', data=top_100_companies)
fig.set(ylabel='number of movies', xlabel='top 20 companies')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
for index, label in enumerate(fig.xaxis.get_ticklabels()):
    if index % 5 != 0:
        label.set_visible(False)
```

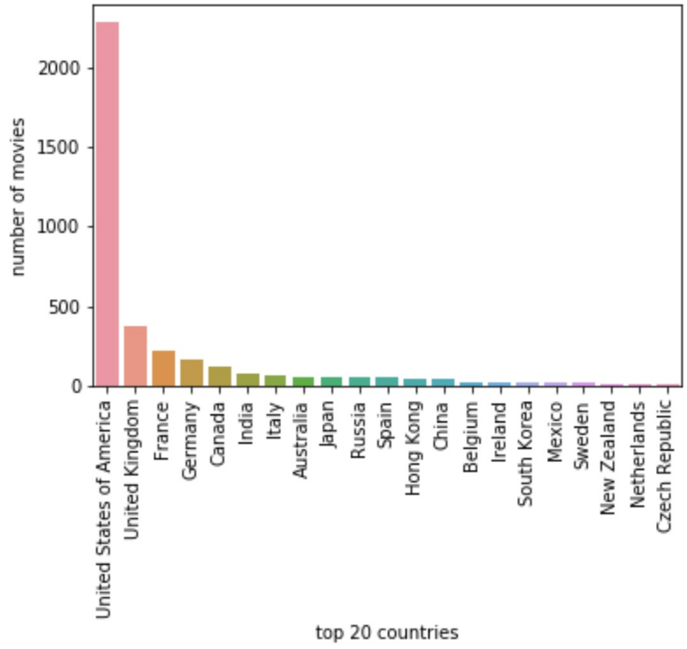


In [30]:
```
#revenue of movies by companies
revenue_by_companies = list()
for col in companies.columns:
    revenue_by_companies.append([col, train.loc[companies[col]==1, 'revenue'].median()])
revenue_by_companies = pd.DataFrame(revenue_by_companies, columns=['company', 'revenue'])
top_100_companies = revenue_by_companies.sort_values(by='revenue', ascending=False).reset_index().loc[0:100]
fig = sns.barplot(x='company', y='revenue', data=top_100_companies)
fig.set(xlabel='top 20 companies')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
for index, label in enumerate(fig.xaxis.get_ticklabels()):
    if index % 5 != 0:
        label.set_visible(False)
```



In [31]:
```
train.loc[train.production_countries.isnull(), 'production_countries'] = "{}"
train['production_countries'] = train.production_countries.apply(lambda x: sorted([d['name'] for d in eval(x)])).apply(lambda x: ','.join(map(str, x)))
countries = train.production_countries.str.get_dummies(sep=',')
```

In [32]:
```
#top production names by no. of movies produced
movies_by_countries = pd.DataFrame(countries.sum(axis=0)).reset_index()
movies_by_countries.columns = ['countries', 'movies']
top_20_countries = movies_by_countries.sort_values(by='movies', ascending=False).re
set_index().loc[0:20]
fig = sns.barplot(x='countries', y='movies', data=top_20_countries)
fig.set(ylabel='number of movies', xlabel='top 20 countries')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
```

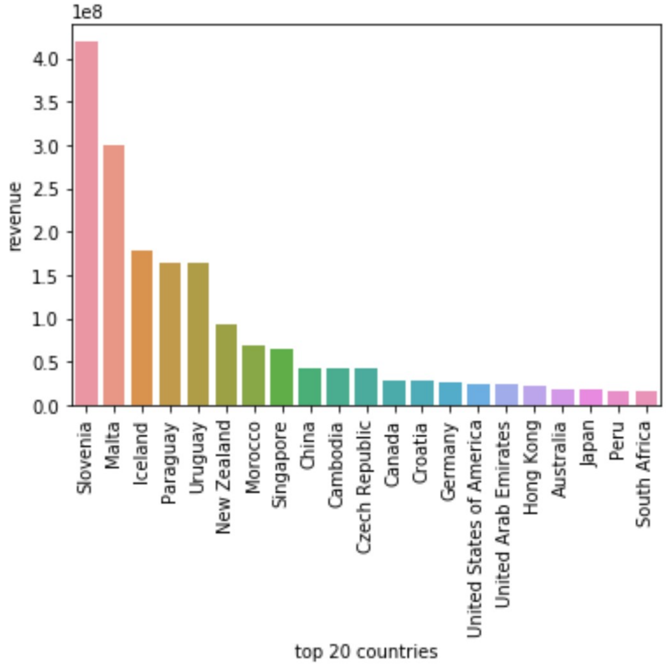


In [33]:

```
#top 20 countries by revenue
revenue_by_countries = list()
for col in countries.columns:
    revenue_by_countries.append([col, train.loc[countries[col]==1, 'revenue'].median()])
revenue_by_countries = pd.DataFrame(revenue_by_countries, columns=['country', 'revenue'])
top_20_countries = revenue_by_countries.sort_values(by='revenue', ascending=False).reset_index().loc[0:20]
fig = sns.barplot(x='country', y='revenue', data=top_20_countries)
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
fig.set(xlabel='top 20 countries')
```

Out[33]:

```
[Text(0.5, 0, 'top 20 countries')]
```

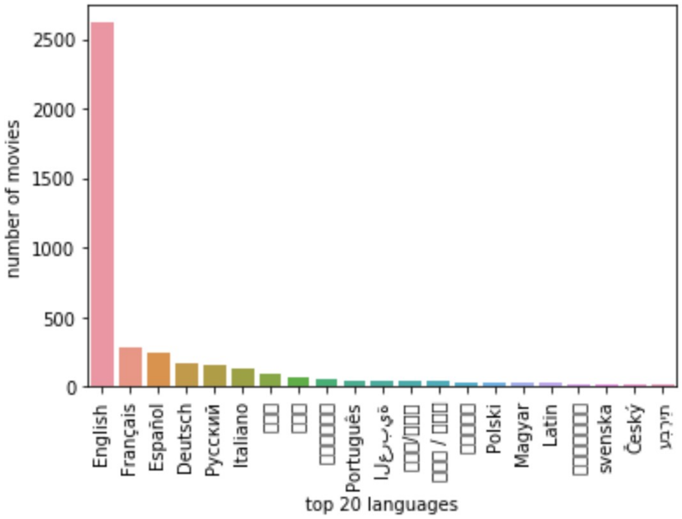


In [34]:

```
train.loc[train.spoken_languages.isnull(), 'spoken_languages'] = "{}"
train['spoken_languages'] = train.spoken_languages.apply(lambda x: sorted([d['name'] for d in eval(x)])).apply(lambda x: ','.join(map(str, x)))
languages = train.spoken_languages.str.get_dummies(sep=',')
```

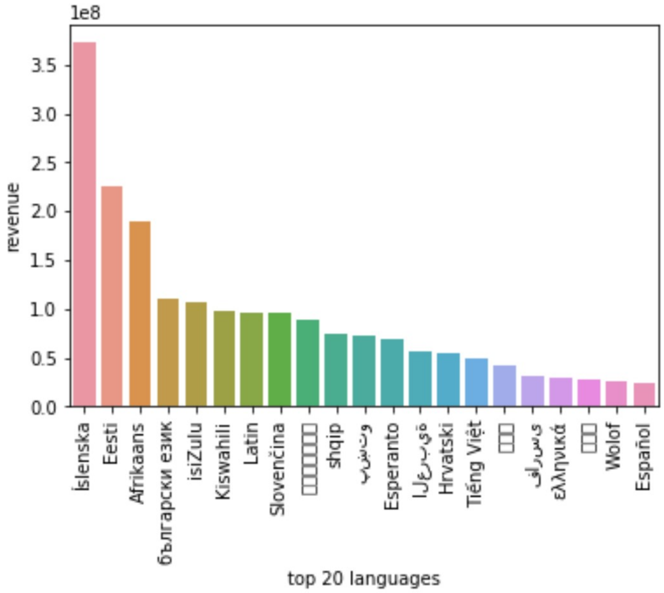
In [35]:

```
#Top 20 languages with highest revenue of movies
movies_by_languages = pd.DataFrame(languages.sum(axis=0)).reset_index()
movies_by_languages.columns = ['language', 'movies']
top_20_languages = movies_by_languages.sort_values(by='movies', ascending=False).re
set_index().loc[0:20]
fig = sns.barplot(x='language', y='movies', data=top_20_languages)
fig.set(ylabel='number of movies', xlabel='top 20 languages')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
```



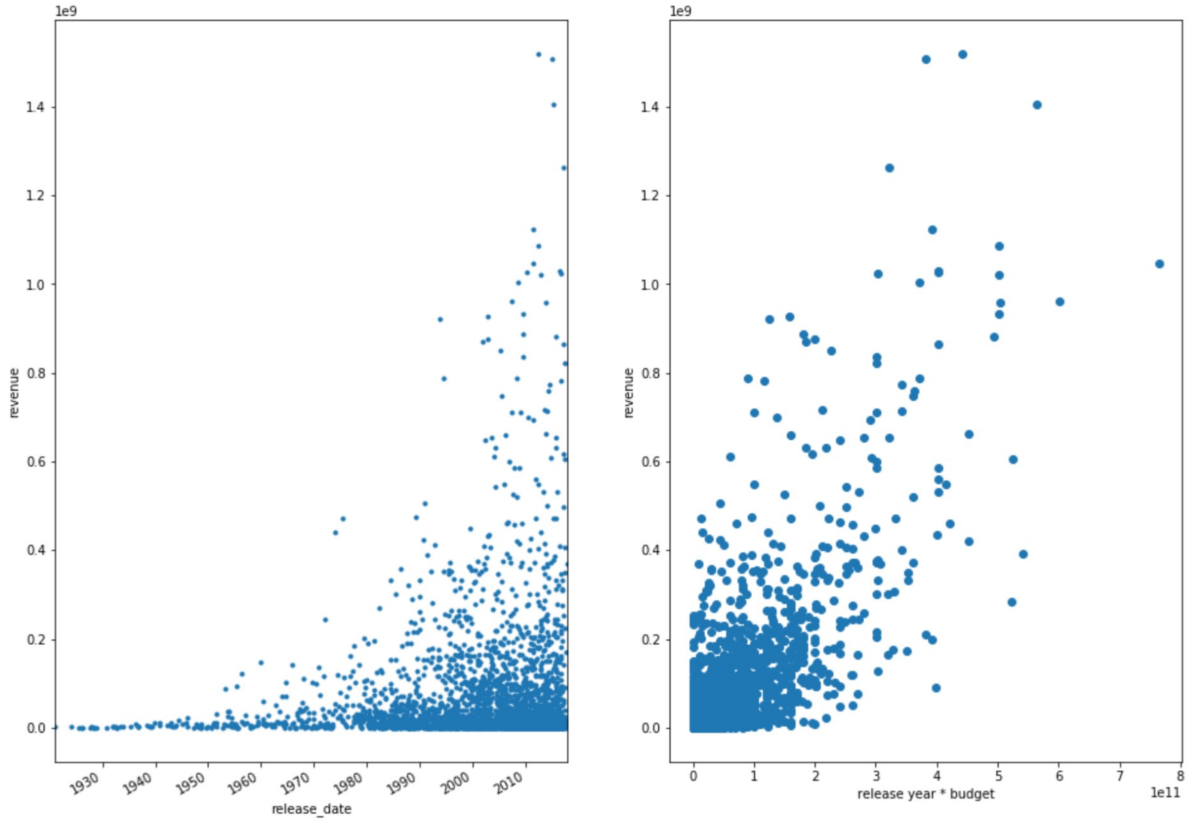
In [36]:

```
#Top 20 languages with lowest revenue of movies
revenue_by_languages = list()
for col in languages.columns:
    revenue_by_languages.append([col, train.loc[languages[col]==1, 'revenue'].media
n()])
revenue_by_languages = pd.DataFrame(revenue_by_languages, columns=['language', 'rev
enue'])
top_20_languages = revenue_by_languages.sort_values(by='revenue', ascending=False).
reset_index().loc[0:20]
fig = sns.barplot(x='language', y='revenue', data=top_20_languages)
fig.set(xlabel='top 20 languages')
_ = fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
```



```
In [39]: #scatter plot revenue vs release_date and release year*budget
         plt.figure(figsize=(16,12))
         ax1 = plt.subplot(121)
         train.plot('release_date','revenue',style='.',ax=ax1,legend=False)
         plt.ylabel('revenue')
         ax2 = plt.subplot(122)
         plt.scatter(x=train['release_year']*train['budget'],y=train['revenue'])
         plt.xlabel('release year * budget')
         plt.ylabel('revenue')
```

```
Out[39]: Text(0, 0.5, 'revenue')
```

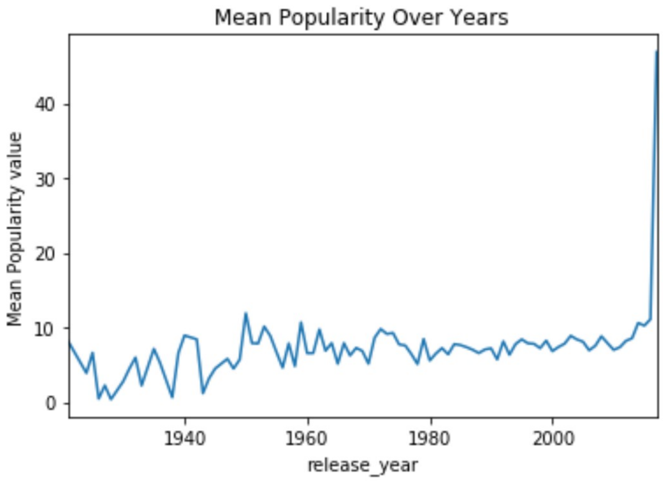


In [40]:

```
#Mean Popularity Over Years
release_year_mean_data=train.groupby(['release_year'])['budget','popularity','revenue'].mean()
release_year_mean_data.head()

fig = plt.figure(figsize=(6, 4))
release_year_mean_data['popularity'].plot(kind='line')
plt.ylabel('Mean Popularity value')
plt.title('Mean Popularity Over Years')
```

Out[40]:

```
Text(0.5, 1.0, 'Mean Popularity Over Years')
```



In [ ]: