Worldwide Box Office Data Analysis

Libraries

In [1]:

```
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.style.use('ggplot')
import datetime
from scipy import stats
from scipy.sparse import hstack, csr_matrix
from sklearn.model_selection import train_test_split, KFold
from wordcloud import WordCloud
from collections import Counter
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import StandardScaler
import nltk
nltk.download('stopwords')
stop = set(stopwords.words('english'))
import os
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import json
import ast
from urllib.request import urlopen
from PIL import Image
print("Packages imported successfully")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\vedan\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Packages imported successfully
```

Task: Data Loading and Exploration

In [2]:

```
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
```

In [3]:

```
train.head()
```

Out[3]:

	id	budget	homepage	imdb_id	original_language	origina
0	1	14000000	NaN	tt2637294	en	Hot Tu Mac
1	2	40000000	NaN	tt0368933	en	The Pr Dia Engag
2	3	3300000	http://sonyclassics.com/whiplash/	tt2582802	en	Wh
3	4	1200000	http://kahaanithefilm.com/	tt1821480	hi	K
4	5	0	NaN	tt1380152	ko	마

In [4]:

```
train.shape
```

Out[4]:

```
(3000, 203)
```

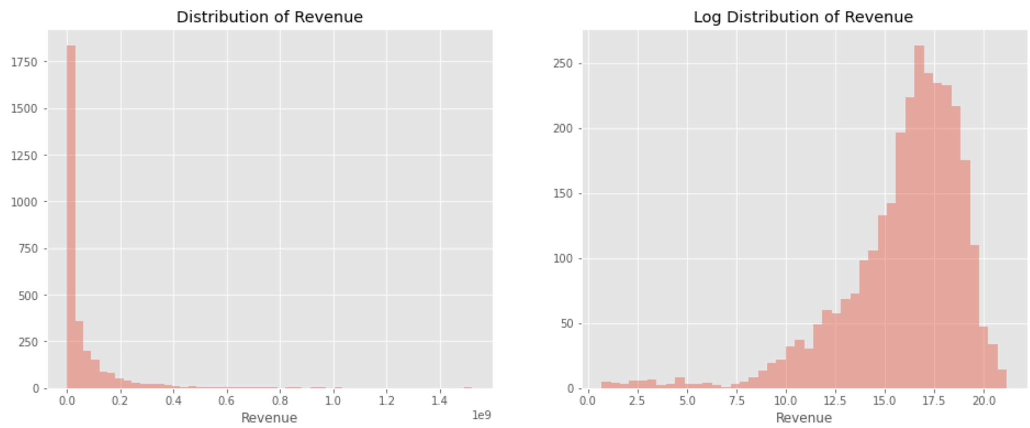
Task: Visualizing the Target Distribution

In [5]:

```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
sns.distplot(train['revenue'], kde = False)
plt.title("Distribution of Revenue")
plt.xlabel("Revenue")
plt.subplot(1, 2, 2)
sns.distplot(np.log1p(train['revenue']), kde = False)
plt.title("Log Distribution of Revenue")
plt.xlabel("Revenue")
```

Out[5]:

```
Text(0.5, 0, 'Revenue')
```



In [6]:

```
train['log_revenue'] = np.log1p(train['revenue'])
```

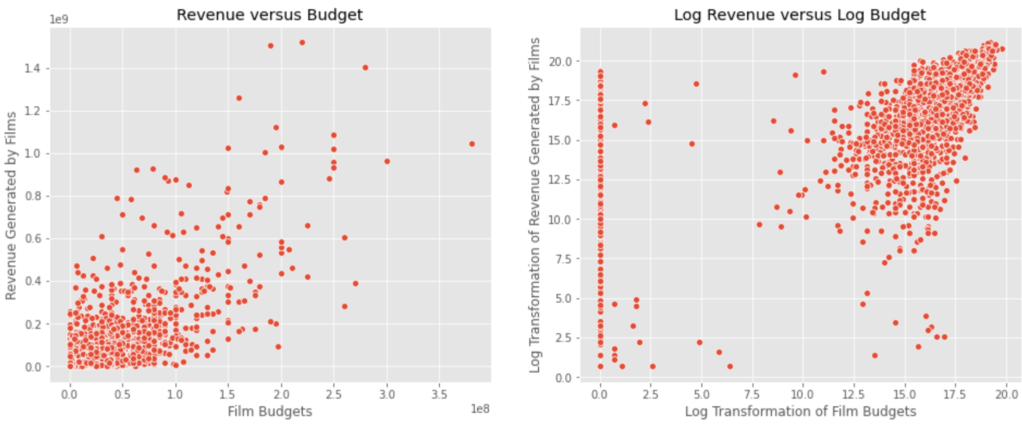
Task: Relationship between Film Revenue and Budget

In [7]:

```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
sns.scatterplot(train['budget'], train['revenue'])
plt.title("Revenue versus Budget")
plt.xlabel("Film Budgets")
plt.ylabel("Revenue Generated by Films")
plt.subplot(1, 2, 2)
sns.scatterplot(np.log1p(train['budget']), np.log1p(train['revenue']))
plt.title("Log Revenue versus Log Budget")
plt.xlabel("Log Transformation of Film Budgets")
plt.ylabel("Log Transformation of Revenue Generated by Films")
```

Out[7]:

```
Text(0, 0.5, 'Log Transformation of Revenue Generated by Films')
```



In [8]:

```
train['log_budget'] = np.log1p(train['budget'])
test['log_budget'] = np.log1p(test['budget'])
```

Task: Does having an Official Homepage Affect Revenue?

In [9]:

```
train['homepage'].value_counts().head(10)
```

Out[9]:

```
http://www.transformersmovie.com/                                   4
http://www.lordoftherings.net/                                      2
http://www.thehobbit.com/                                           2
http://phoenixforgotten.com/                                        1
http://www.the-scorpion-king.com/                                   1
http://disneydvd.disney.go.com/tarzanr-special-edition.html         1
http://constantinemovie.warnerbros.com/                             1
http://www.paranormalactivity-movie.com/                            1
http://www.moanmovie.com/                                           1
http://www.imdb.com/title/tt0095169/                                1
Name: homepage, dtype: int64
```

In [10]:

```
train['has_homepage'] = 0
train.loc[train['homepage'].isnull() == False, 'has_homepage'] = 1
test['has_homepage'] = 0
test.loc[test['homepage'].isnull() == False, 'has_homepage'] = 1
```

In [11]:

```
fig = plt.figure(figsize = (10, 8))
sns.catplot(x = "has_homepage", y = "revenue", data = train)
plt.title("Revenue for Films with & without Websites")
plt.xlabel("HomePages")
plt.ylabel("Revenue Generated")
```

Out[11]:

```
Text(6.799999999999997, 0.5, 'Revenue Generated')
```

```
<Figure size 720x576 with 0 Axes>
```

Task: Distribution of Languages in Film

In [12]:

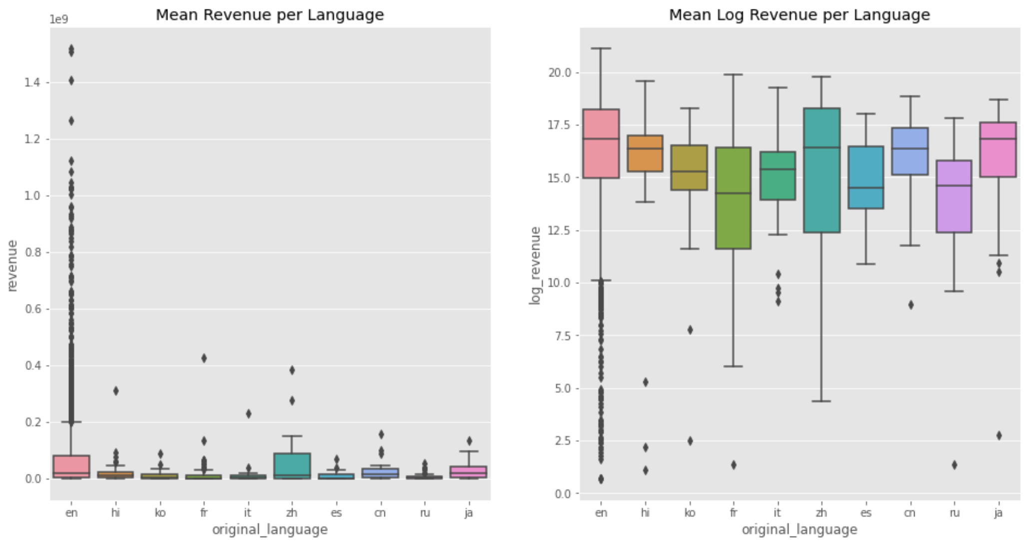
```
language_data = train.loc[train['original_language'].isin(train['original_langu
age'].value_counts().head(10).index)]
```

In [13]:

```
plt.figure(figsize = (16, 8))
plt.subplot(1, 2, 1)
sns.boxplot(x = "original_language", y = "revenue", data = language_data)
plt.title("Mean Revenue per Language")
plt.subplot(1, 2, 2)
sns.boxplot(x = "original_language", y = "log_revenue", data = language_data)
plt.title("Mean Log Revenue per Language")
```

Out[13]:

```
Text(0.5, 1.0, 'Mean Log Revenue per Language')
```

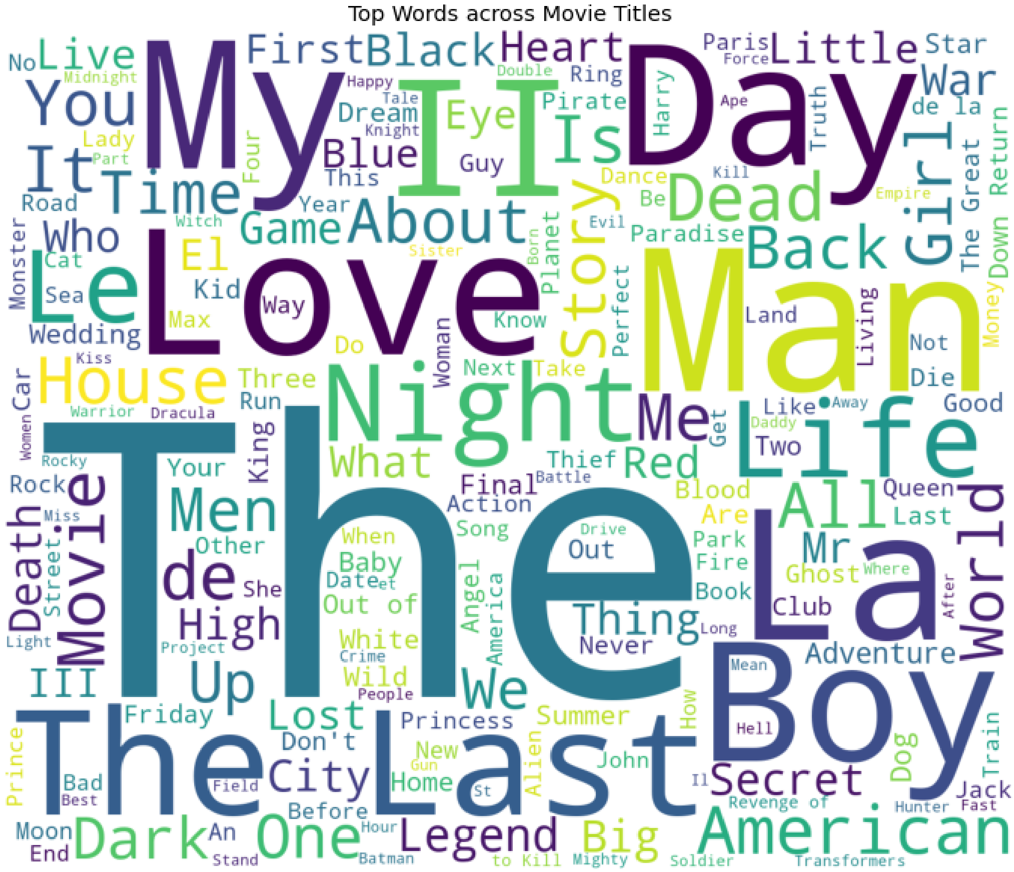


Task: Frequent Words in Film Titles and Descriptions

In [14]:

```
plt.figure(figsize = (12,12))
text = ' '.join(train['original_title'].values)
wordcloud = WordCloud(max_font_size=None,
                      background_color="white",
                      width = 1200, height = 1000).generate(text)
plt.imshow(wordcloud)
plt.title("Top Words across Movie Titles")
plt.axis('off')
plt.show()
```

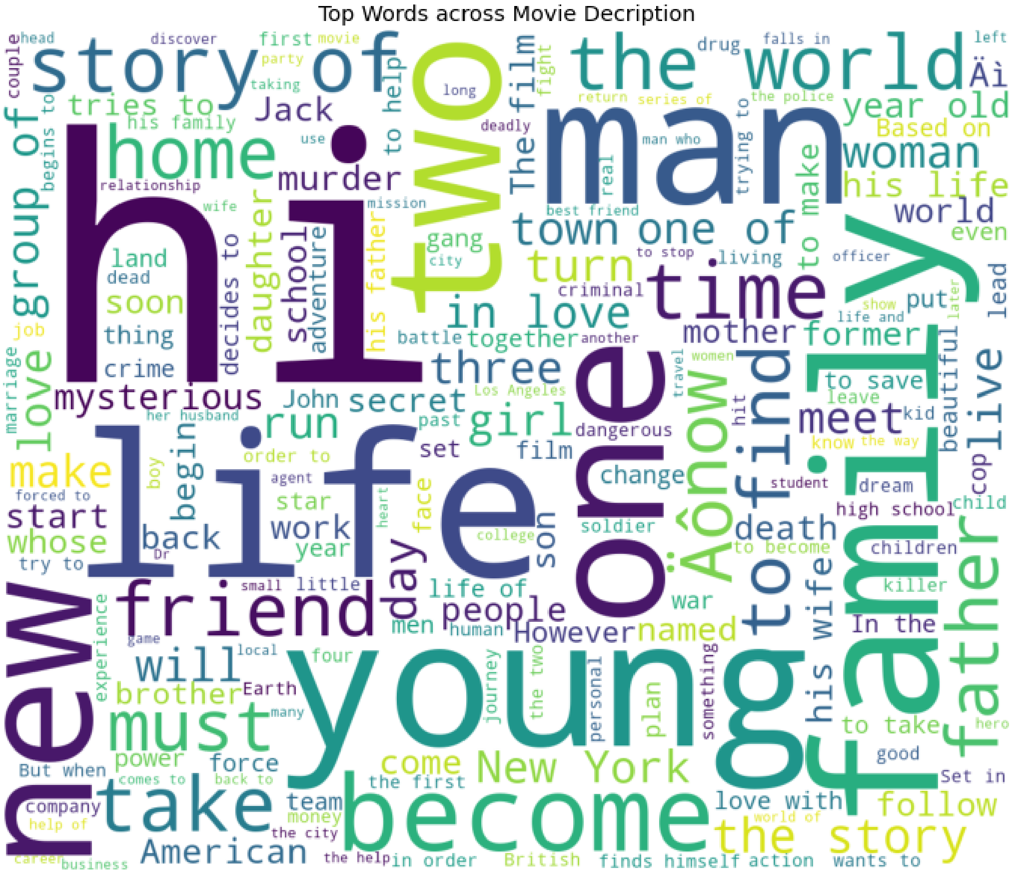
Top Words across Movie Titles



In [15]:

```
plt.figure(figsize = (12,12))
text = ' '.join(train['overview'].fillna('').values)
wordcloud = WordCloud(max_font_size=None,
                      background_color="white",
                      width = 1200, height = 1000).generate(text)
plt.imshow(wordcloud)
plt.title("Top Words across Movie Decription")
plt.axis('off')
plt.show()
```

Top Words across Movie Decription



Task: Do Film Descriptions Impact Revenue?

In [16]:

```
import eli5
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LinearRegression
```

```
C:\Users\vedan\anaconda3\lib\site-packages\sklearn\utils\deprecati
on.py:143: FutureWarning:

The sklearn.metrics.scorer module is  deprecated in version 0.22 a
nd will be removed in version 0.24. The corresponding classes / fu
nctions should instead be imported from sklearn.metrics. Anything
that cannot be imported from sklearn.metrics is now part of the pr
ivate API.

C:\Users\vedan\anaconda3\lib\site-packages\sklearn\utils\deprecati
on.py:143: FutureWarning:

The sklearn.feature_selection.base module is  deprecated in versio
n 0.22 and will be removed in version 0.24. The corresponding clas
ses / functions should instead be imported from sklearn.feature_se
lection. Anything that cannot be imported from sklearn.feature_sel
ection is now part of the private API.
```

In [17]:

```
vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    analyzer='word',
    token_pattern=r'\w{1,}',
    ngram_range=(1, 2),
    min_df=5
)
```

In [18]:

```
overview_text = vectorizer.fit_transform(train['overview'].fillna(''))
linreg = LinearRegression()
linreg.fit(overview_text, train['log_revenue'])
```

Out[18]:

```
LinearRegression()
```

In [19]:

```
eli5.show_weights(linreg, vec = vectorizer,
                  feature_filter = lambda x: x != '<BIAS>')
```

Out[19]:

y top features

Weight?	Feature
+13.074	to
+10.131	bombing
+9.981	the
+9.777	complications
… 3858 more positive …	
… 3315 more negative …	
-9.281	politicians
-9.391	18
-9.481	violence
-9.628	escape and
-9.716	life they
-10.021	ones
-10.111	sally
-10.291	attracted to
-10.321	who also
-10.421	casino
-10.614	receiving
-10.759	kept
-12.139	and be
-12.939	campaign
-13.858	mike
-15.273	woman from

Task: Analyzing Movie Release Dates

In [20]:

```
test.loc[test['release_date'].isnull() == False, 'release_date'].head()
```

Out[20]:

```
0    7/14/07
1    5/19/58
2    5/23/97
3     9/4/10
4    2/11/05
Name: release_date, dtype: object
```

Task: Preprocessing Features

In [21]:

```
def fix_date(x) :
    year = x.split('/')[2]
    if int(year) <= 19 :
        return x[:-2] + '20' + year
    else :
        return x[:-2] + '19' + year
```

In [22]:

```
test.loc[test['release_date'].isnull() == True].head()
```

Out[22]:

	id	budget	homepage	imdb_id	original_language	original_title	overview
828	3829	0	NaN	tt0210130	en	Jails, Hospitals & Hip-Hop	Jails, Hospitals & Hip-Hop is a cinematic ...

In [23]:

```
test.loc[test['release_date'].isnull() == True, 'release_date'] = '05/01/00'
```

In [24]:

```
train['release_date'] = train['release_date'].apply(lambda x: fix_date(x))
test['release_date'] = test['release_date'].apply(lambda x: fix_date(x))
```

Task: Creating Features Based on Release Date

In [25]:

```
train['release_date'] = pd.to_datetime(train['release_date'])
test['release_date'] = pd.to_datetime(test['release_date'])
```

In [26]:

```
def process_date(df) :
    date_parts = ['year', 'weekday', 'month', 'weekofyear', 'day', 'quarter']
    for part in date_parts :
        part_col = 'release_date' + '_' + part
        df[part_col] = getattr(df['release_date'].dt, part).astype(int)
    return df

train = process_date(train)
test = process_date(test)
```

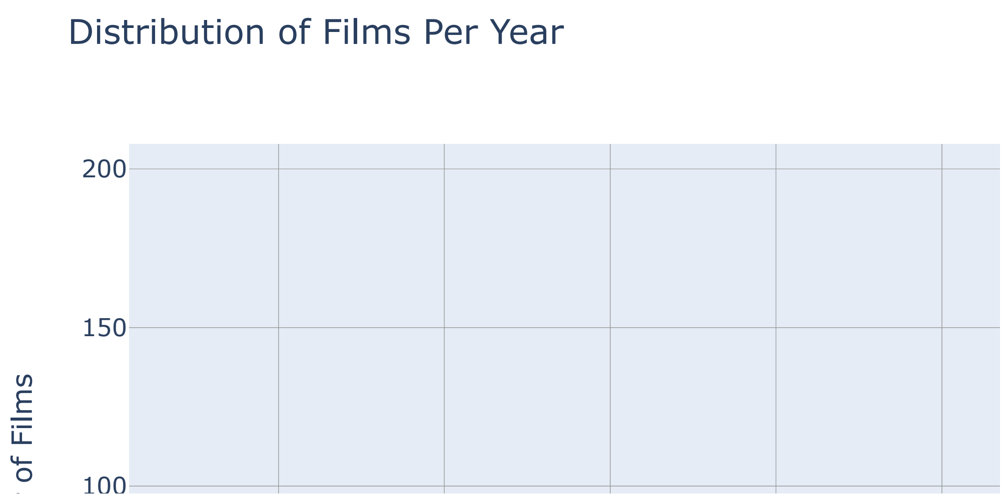
Task: Using Plotly to Visualize the Number of Films Per Year

In [30]:

```
d1 = train['release_date_year'].value_counts().sort_index()
d2 = test['release_date_year'].value_counts().sort_index()
```

In [32]:

```
data = [go.Scatter(x = d1.index, y = d1.values, name = 'train'),
        go.Scatter(x = d2.index, y = d2.values, name = 'test')]
layout = go.Layout(dict(title = "Distribution of Films Per Year",
                       xaxis = dict(title = "Year"),
                       yaxis = dict(title = "Number of Films")),
                   legend = dict(orientation = 'v'))
py.iplot(dict(data = data, layout = layout))
```



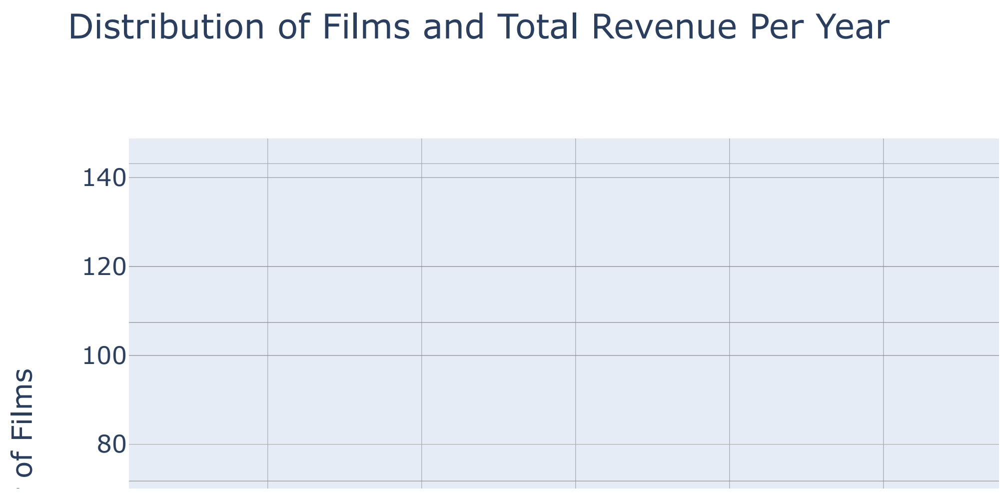
Task: Number of Films and Revenue Per Year

In [ ]:

```
d1 = train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].sum()
```

In [33]:

```
data = [go.Scatter(x = d1.index, y = d1.values, name = 'Film Count'),
        go.Scatter(x = d2.index, y = d2.values, name = 'Total Revenue', yaxis =
'y2')]
layout = go.Layout(dict(title = "Distribution of Films and Total Revenue Per Ye
ar",
                        xaxis = dict(title = "Year"),
                        yaxis = dict(title = "Number of Films"),
                        yaxis2 = dict(title = "Total Revenue", overlaying = 'y'
, side = 'right')),
                   legend = dict(orientation = 'v'))
py.iplot(dict(data = data, layout = layout))
```

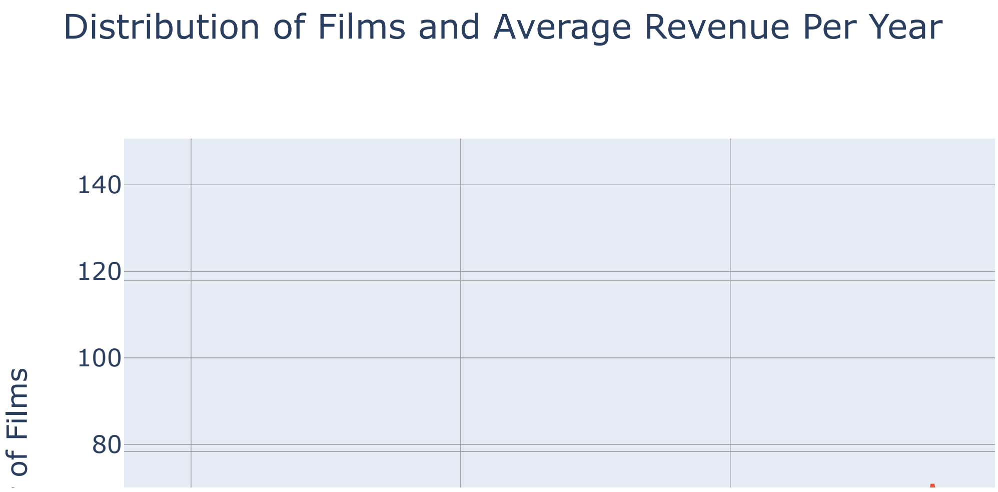


In [35]:

```
d1 = train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].mean()
```

In [41]:

```
data = [go.Scatter(x = d1.index, y = d1.values, name = 'Film Count', mode = "li
nes+markers"),
        go.Scatter(x = d2.index, y = d2.values, name = 'Average Revenue', yaxis
= 'y2')]
layout = go.Layout(dict(title = "Distribution of Films and Average Revenue Per
 Year",
                        xaxis = dict(title = "Year"),
                        yaxis = dict(title = "Number of Films"),
                        yaxis2 = dict(title = "Average Revenue", overlaying =
'y', side = 'right')),
                   legend = dict(orientation = 'v'))
py.iplot(dict(data = data, layout = layout))
```



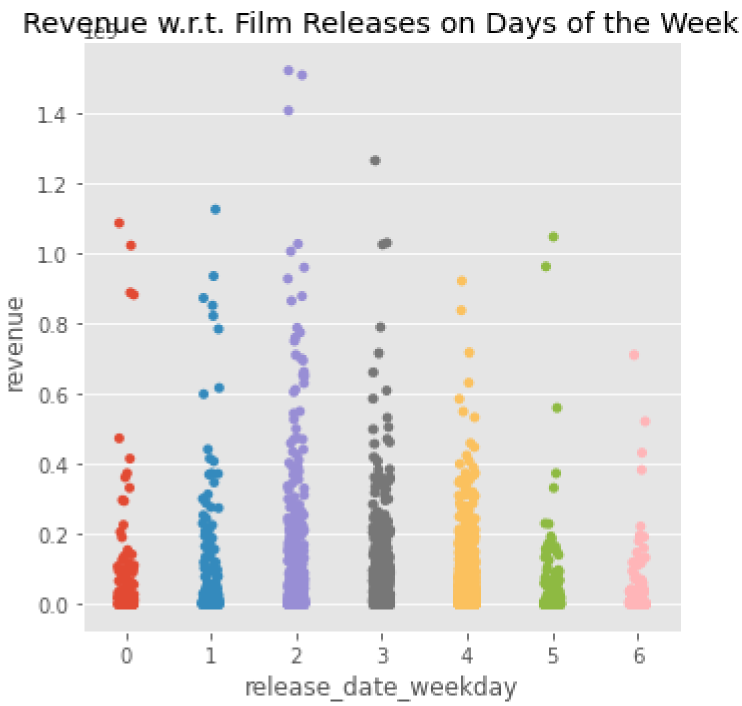
Task: Do Release Days Impact Revenue

In [37]:

```
sns.catplot(x = "release_date_weekday", y = "revenue", data = train)
plt.title("Revenue w.r.t. Film Releases on Days of the Week")
```

Out[37]:

```
Text(0.5, 1.0, 'Revenue w.r.t. Film Releases on Days of the Week')
```



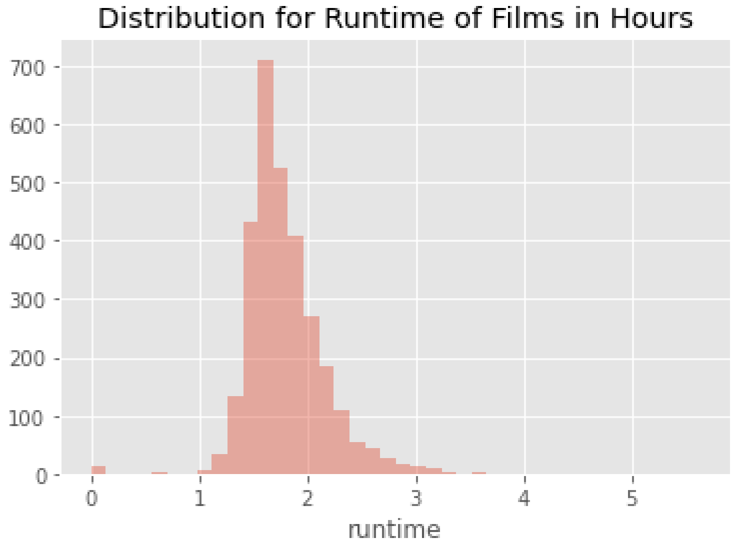
Task: Relationship between Runtime and Revenue

In [39]:

```
sns.distplot(train['runtime'].fillna(0) / 60, bins = 40, kde = False)
plt.title('Distribution for Runtime of Films in Hours')
```

Out[39]:

```
Text(0.5, 1.0, 'Distribution for Runtime of Films in Hours')
```

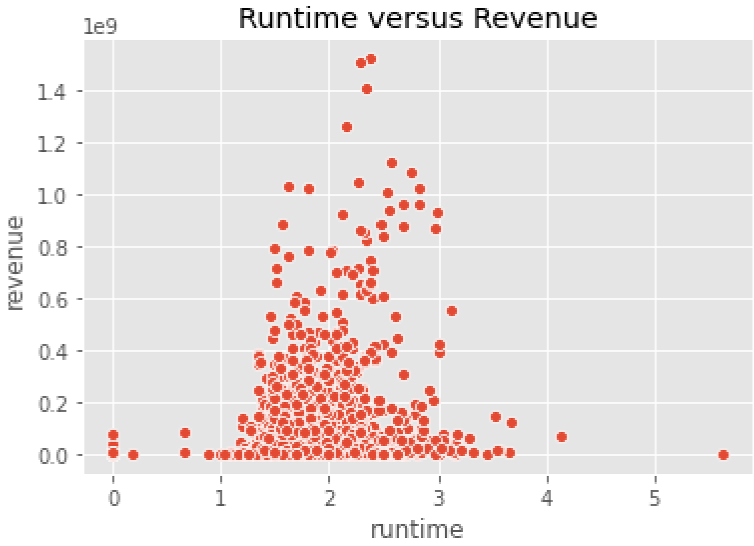


In [40]:

```
sns.scatterplot(x = train['runtime'].fillna(0) / 60, y = train['revenue'])
plt.title("Runtime versus Revenue")
```

Out[40]:

```
Text(0.5, 1.0, 'Runtime versus Revenue')
```



In [ ]: