INTRODUCTION

The dataset we would be exploring in this project is tmdb-movies which consists of 21 features and 10866 observations, during the course of the work we would be answering the following questions

- 1. Which year had the highest number of movies released?
- 1. Which month has the highest number of released movies?
- 1. Which month generated more revenue for the movie industry?
- 1. Does the number of movies released translate to more revenue?
- 1. Does vote_count affect revenue generated from a movie?
- 1. Which day accounts for more movies released?
- 1. Which movie title generated the highest revenue?
- 1. Which movie title has the longest runtime?
- 1. Does a longer runtime indicate more revenue?
- 1. Which movie has the highest runtime?

IMPORTING LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
```

LOADING DATASET

In [2]:
```
df = pd.read_csv('tmdb-movies.csv')
df.head()
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan

5 rows × 21 columns

DATA WRANGLING

In [3]:
```
#Checking to see the number of features and observations
df.shape
```

Out[3]:
```
(10866, 21)
```

In [4]:
```
#Checking to see the names of all the columns
df.columns
```

Out[4]:
```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

In [5]:
```
#Checking to see the data type of the dataset
df.dtypes
```

Out[5]:
```
id                        int64
imdb_id                  object
popularity              float64
budget                    int64
revenue                   int64
original_title           object
cast                     object
homepage                 object
director                 object
tagline                  object
keywords                 object
overview                 object
runtime                   int64
genres                   object
production_companies     object
release_date             object
vote_count                int64
vote_average            float64
release_year              int64
budget_adj              float64
revenue_adj             float64
dtype: object
```

In [6]:
```
#Checking to see a brief information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10866 non-null  int64
 1   imdb_id               10856 non-null  object
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64
 4   revenue               10866 non-null  int64
 5   original_title        10866 non-null  object
 6   cast                  10790 non-null  object
 7   homepage              2936 non-null   object
 8   director              10822 non-null  object
 9   tagline               8042 non-null   object
 10  keywords              9373 non-null   object
 11  overview              10862 non-null  object
 12  runtime               10866 non-null  int64
 13  genres                10843 non-null  object
 14  production_companies  9836 non-null   object
 15  release_date          10866 non-null  object
 16  vote_count            10866 non-null  int64
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

In [7]:
```
#Checking for null values
df.isnull().sum()
```

Out[7]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

In [8]:
```
#Checking to see a summary statistics of the dataset
df.describe()
```

Out[8]:

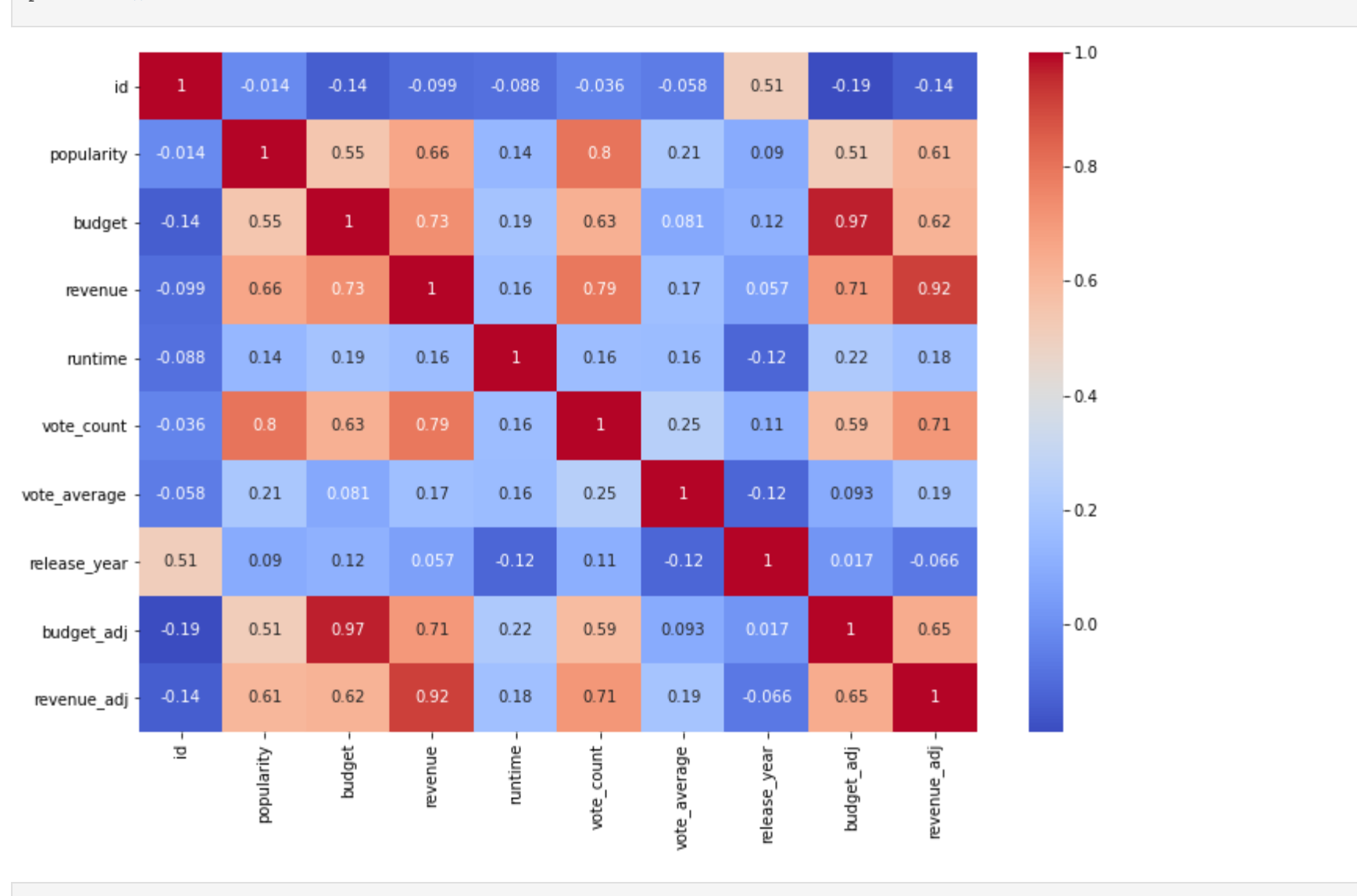
	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	rever
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.0866
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658	1.755104e+07	5.1364
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941	3.430616e+07	1.4463
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.0000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.0000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.0000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000	2.085325e+07	3.3697
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.8271

In [9]:
```
#Checking to see correlation between features
df.corr()
```

Out[9]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
id	1.000000	-0.014350	-0.141351	-0.099227	-0.088360	-0.035551	-0.058363	0.511364	-0.189015	-0.138477
popularity	-0.014350	1.000000	0.545472	0.663358	0.139033	0.800828	0.209511	0.089801	0.513550	0.609083
budget	-0.141351	0.545472	1.000000	0.734901	0.191283	0.632702	0.081014	0.115931	0.968963	0.622505
revenue	-0.099227	0.663358	0.734901	1.000000	0.162838	0.791175	0.172564	0.057048	0.706427	0.919110
runtime	-0.088360	0.139033	0.191283	0.162838	1.000000	0.163278	0.156835	-0.117204	0.221114	0.175676
vote_count	-0.035551	0.800828	0.632702	0.791175	0.163278	1.000000	0.253823	0.107948	0.587051	0.707942
vote_average	-0.058363	0.209511	0.081014	0.172564	0.156835	0.253823	1.000000	-0.117632	0.093039	0.193085
release_year	0.511364	0.089801	0.115931	0.057048	-0.117204	0.107948	-0.117632	1.000000	0.016793	-0.066256
budget_adj	-0.189015	0.513550	0.968963	0.706427	0.221114	0.587051	0.093039	0.016793	1.000000	0.646607
revenue_adj	-0.138477	0.609083	0.622505	0.919110	0.175676	0.707942	0.193085	-0.066256	0.646607	1.000000

In [10]:
```
#Visualizing correlation between features
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), cmap = 'coolwarm', annot = True)
plt.show()
```



In [11]:
```
#Checking the number of unique features
df.nunique()
```

Out[11]:
```
id                      10865
imdb_id                 10855
popularity              10814
budget                    557
revenue                  4702
original_title          10571
cast                    10719
homepage                 2896
director                 5067
tagline                  7997
keywords                 8804
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj               2614
revenue_adj              4840
dtype: int64
```

DATA CLEANING

In [12]:
```
null_val = df.isnull().sum()
```

In [13]:
```
#Determining percentage null values in dataset
for item in null_val:
    print((item/10866)*100)
```

```
0.0
0.09203018590097553
0.0
0.0
0.0
0.0
0.6994294128474139
72.97993741947359
0.40493281796429226
25.989324498435483
13.740106755015645
0.03681207436039021
0.0
0.21166942757224366
9.479109147800479
0.0
0.0
0.0
0.0
0.0
0.0
```

In [14]:
```
#features with 70% null values, the entire columns will be dropped
df.drop(columns=['homepage'], inplace=True)
```

In [15]:
```
#features with null values between 5% and less than 70% will be filled with unknown
df['tagline'].fillna('Unknown',inplace=True)
df['keywords'].fillna('Unknown',inplace=True)
df['production_companies'].fillna('Unknown',inplace=True)
```

In [16]:
```
#features with less than 5% null values, the observations with the null values will be dropped since the percen
df.dropna(inplace=True)
```

In [17]:
```
#Checking to confirm if the dataset is clean
df.isnull().sum()
```

Out[17]:
```
id                      0
imdb_id                 0
popularity              0
budget                  0
revenue                 0
original_title          0
cast                    0
director                0
tagline                 0
keywords                0
overview                0
runtime                 0
genres                  0
production_companies    0
release_date            0
vote_count              0
vote_average            0
release_year            0
budget_adj              0
revenue_adj             0
dtype: int64
```

In [18]:
```
#Changing the data type of release_date to datetime
df['release_date'] = pd.to_datetime(df['release_date'])
```

In [19]:
```
#Confirming the change in release_date data type
df.dtypes
```

Out[19]:
```
id                               int64
imdb_id                         object
popularity                     float64
budget                           int64
revenue                          int64
original_title                  object
cast                            object
director                        object
tagline                         object
keywords                        object
overview                        object
runtime                          int64
genres                          object
production_companies            object
release_date            datetime64[ns]
vote_count                       int64
vote_average                   float64
release_year                     int64
budget_adj                     float64
revenue_adj                    float64
dtype: object
```

In [20]:
```
#creating new feature release_day) from release_date
df['release_day'] = df.release_date.dt.weekday
```

In [21]:
```
#Converting the numeric variables in release_day to string
mapper = {0:'Monday',1:'Tuesday',2:'Wednesday',3:'Thursday',4:'Friday',5:'Saturday'}
df['release_day'] = df['release_day'].map(mapper)
```

In [22]:
```
#Confirming the change
df.head()
```

Out[22]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	tagline	keywo
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	monster|dna|tyrannosau rex|velociraptor|isla
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	future|chase|po apocalyptic|dystopia|austr
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	One Choice Can Destroy You	based novel|revolution|dystopia|sequel|dy
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Every generation has a story.	android|spaceship|jedi|space opera
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Vengeance Hits Home	car race|speed|revenge|suspense|

5 rows × 21 columns

In [23]:
```
#creating new feature release_month) from release_date
df['release_month'] = df.release_date.dt.month
```

In [24]:
```
#Converting the numeric variables in release_day to string
mapper = {1:'January',2:'February',3:'March',4:'April',5:'May',6:'June',7:'July',8:'August',9:'September',10:'O
df['release_month'] = df['release_month'].map(mapper)
```

In [25]:
```
#Confirming the change in release_month
df.head()
```

Out[25]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	tagline	keywo
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	monster|dna|tyrannosau rex|velociraptor|isla
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	future|chase|po apocalyptic|dystopia|austr
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	One Choice Can Destroy You	based novel|revolution|dystopia|sequel|dy
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Every generation has a story.	android|spaceship|jedi|space opera
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Vengeance Hits Home	car race|speed|revenge|suspense|

5 rows × 22 columns

In [26]:
```
#Checking for duplicates in the dataset
sum(df.duplicated())
```

Out[26]:
```
1
```

In [27]:
```
#Removing duplicate values
df.drop_duplicates(inplace=True)
sum(df.duplicated())
```

Out[27]:
```
0
```

EXPLORATORY DATA ANALYSIS

Which year had the most number of movies released?

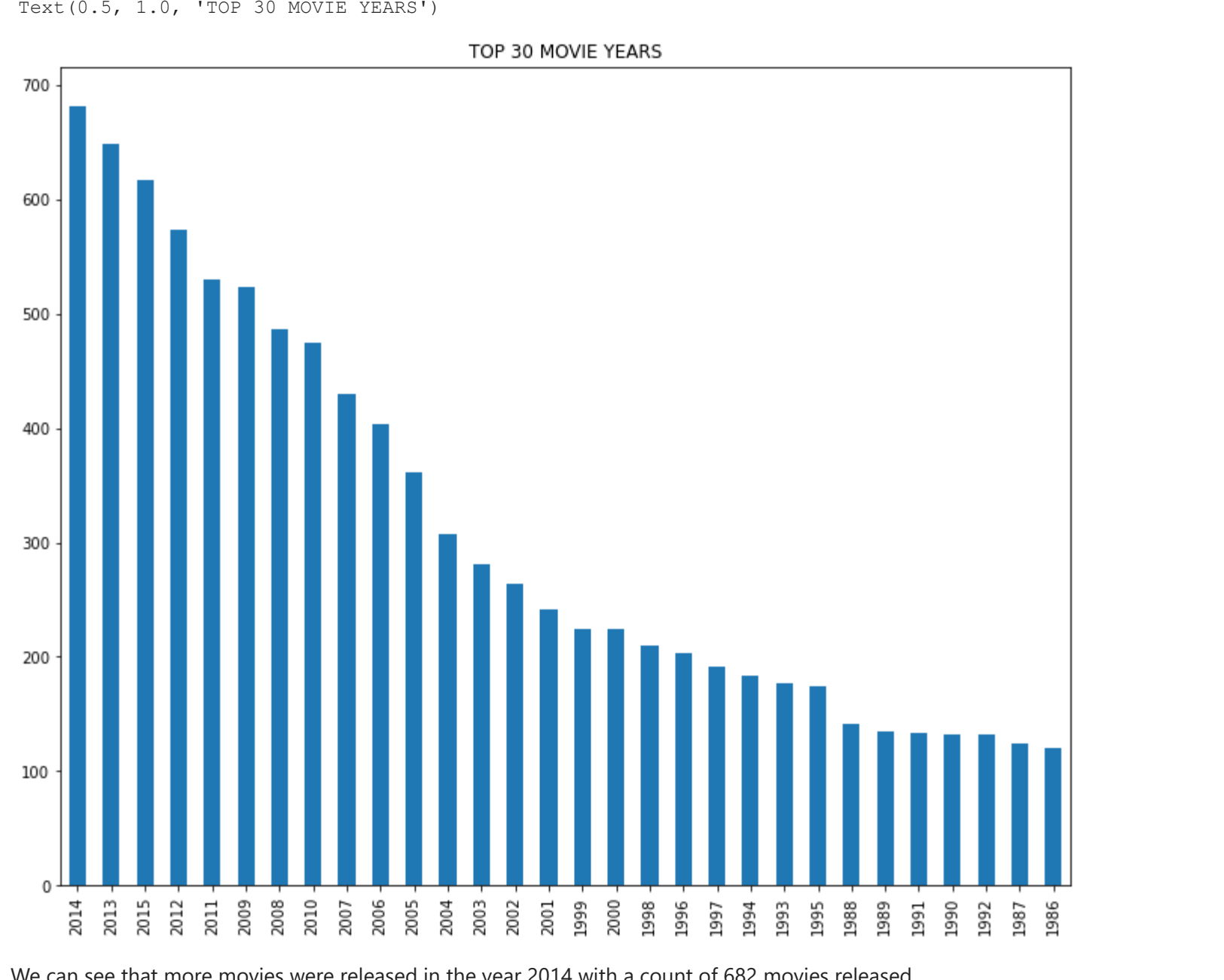
In [28]:
```
pd.DataFrame(df['release_year'].value_counts(ascending=False)[0:16])
```

Out[28]:

	release_year
2014	682
2013	648
2015	617
2012	574
2011	530
2009	523
2008	486
2010	475
2007	430
2006	404
2005	361
2004	307
2003	281
2002	264
2001	241
1999	224

In [29]:
```
#Visualizing the result
plt.figure(figsize=(12,10))
df['release_year'].value_counts()[0:30].plot(kind='bar')
plt.title('TOP 30 MOVIE YEARS')
```

Out[29]:
```
Text(0.5, 1.0, 'TOP 30 MOVIE YEARS')
```



We can see that more movies were released in the year 2014 with a count of 682 movies released

Which month had the highest number of released movies?

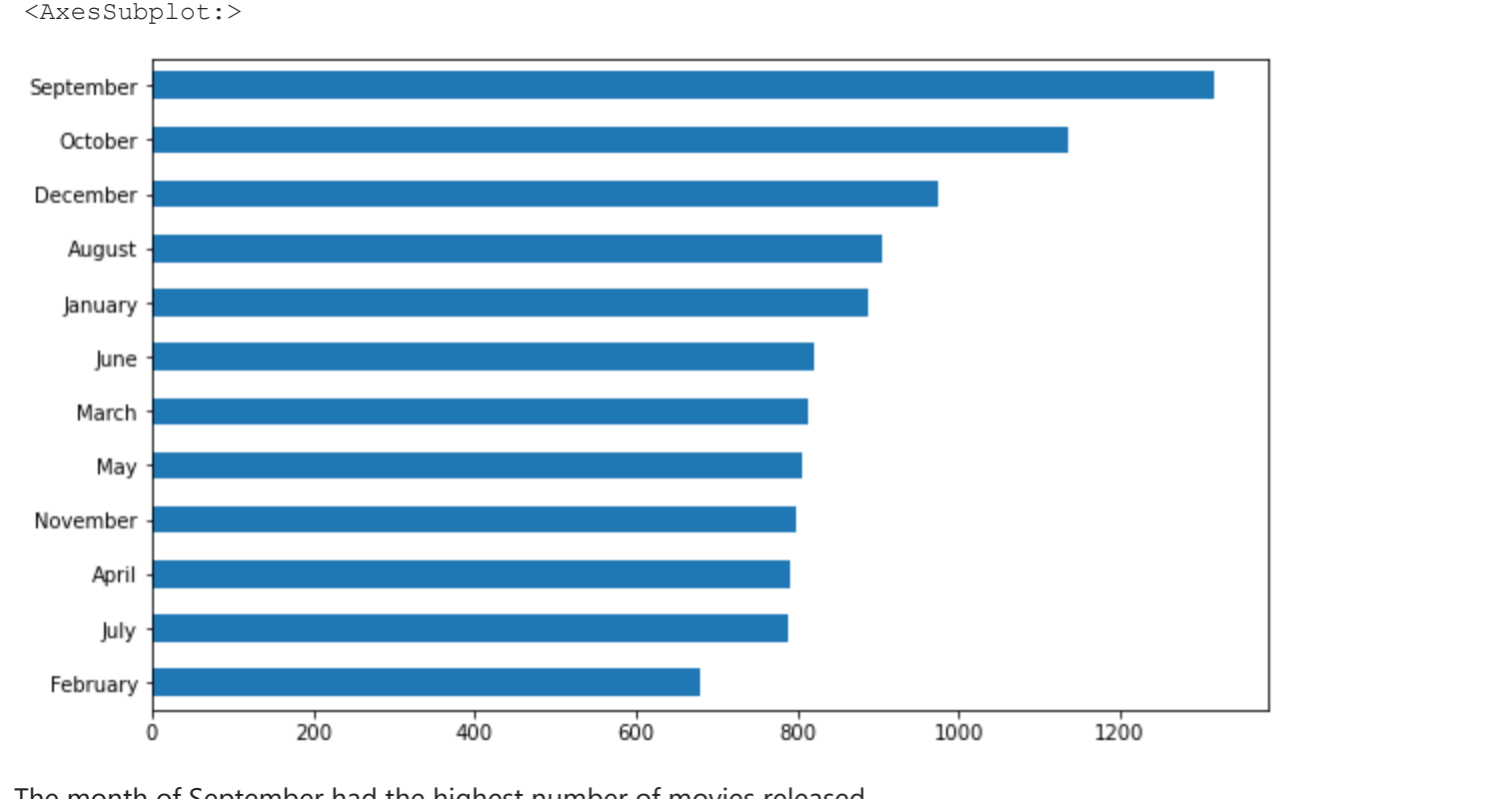
In [30]:
```
pd.DataFrame(df['release_month'].value_counts())
```

Out[30]:

	release_month
September	1318
October	1137
December	974
August	906
January	888
June	822
March	813
May	807
November	798
April	791
July	790
February	680

In [31]:
```
df['release_month'].value_counts(ascending=True).plot(kind='barh', figsize=(10,6))
```

Out[31]:
```
<AxesSubplot:>
```



The month of September had the highest number of movies released

Which day accounts for more movie release?

In [32]:
```
df['release_day'].value_counts()
```

Out[32]:
```
Friday       4225
Thursday     1674
Wednesday    1513
Tuesday      1189
Saturday      782
Monday        653
Name: release_day, dtype: int64
```

In [33]:
```
pd.DataFrame(df['release_day'].value_counts(ascending=False, normalize=True)*100)
```

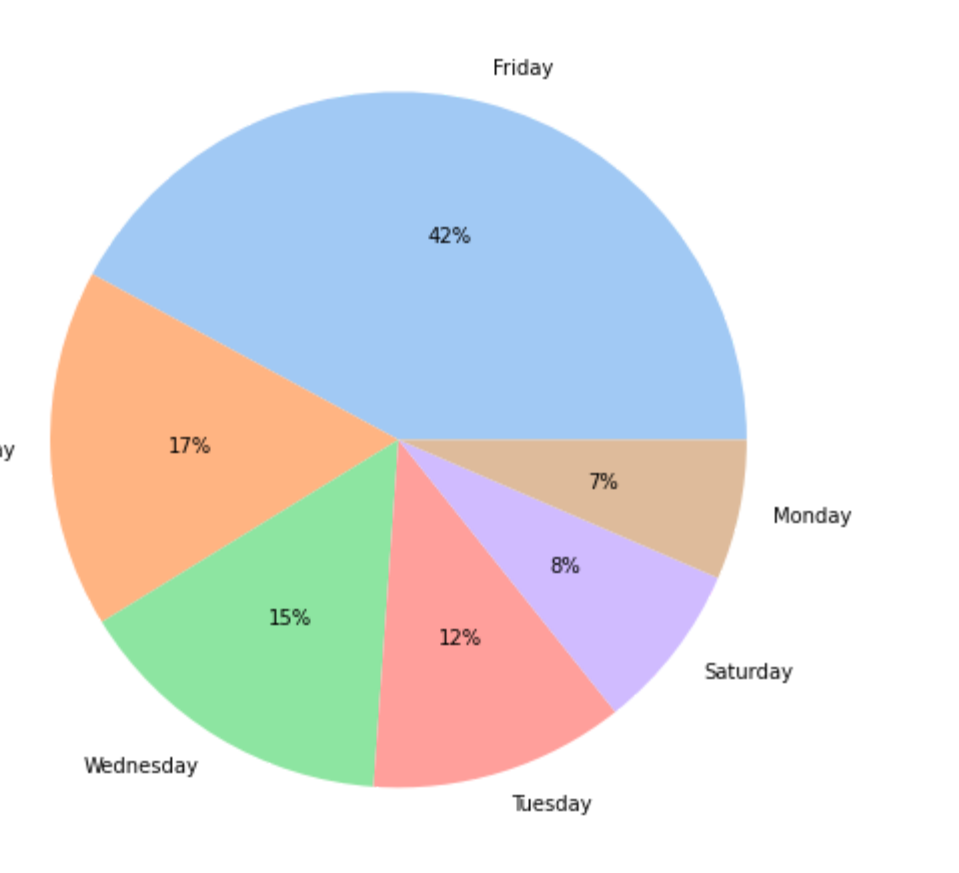
Out[33]:

	release_day
Friday	42.098446
Thursday	16.679952
Wednesday	15.075727
Tuesday	11.847350
Saturday	7.791949
Monday	6.506576

In [34]:
```
#Visualizing the result
release_day = pd.DataFrame(df.groupby('release_day')['release_day'].size().sort_values(ascending=False))
release_day.columns = ['Count']

labels = list(release_day.index)
#Define Seaborn color palette to use
colors = sns.color_palette('pastel')[0:6]

#Create pie chart
plt.figure(figsize=(8,8))
plt.pie(release_day['Count'], labels = labels, colors = colors, autopct='%.0f%%')
plt.show()
```



More movies were released on Friday which account for about 42%

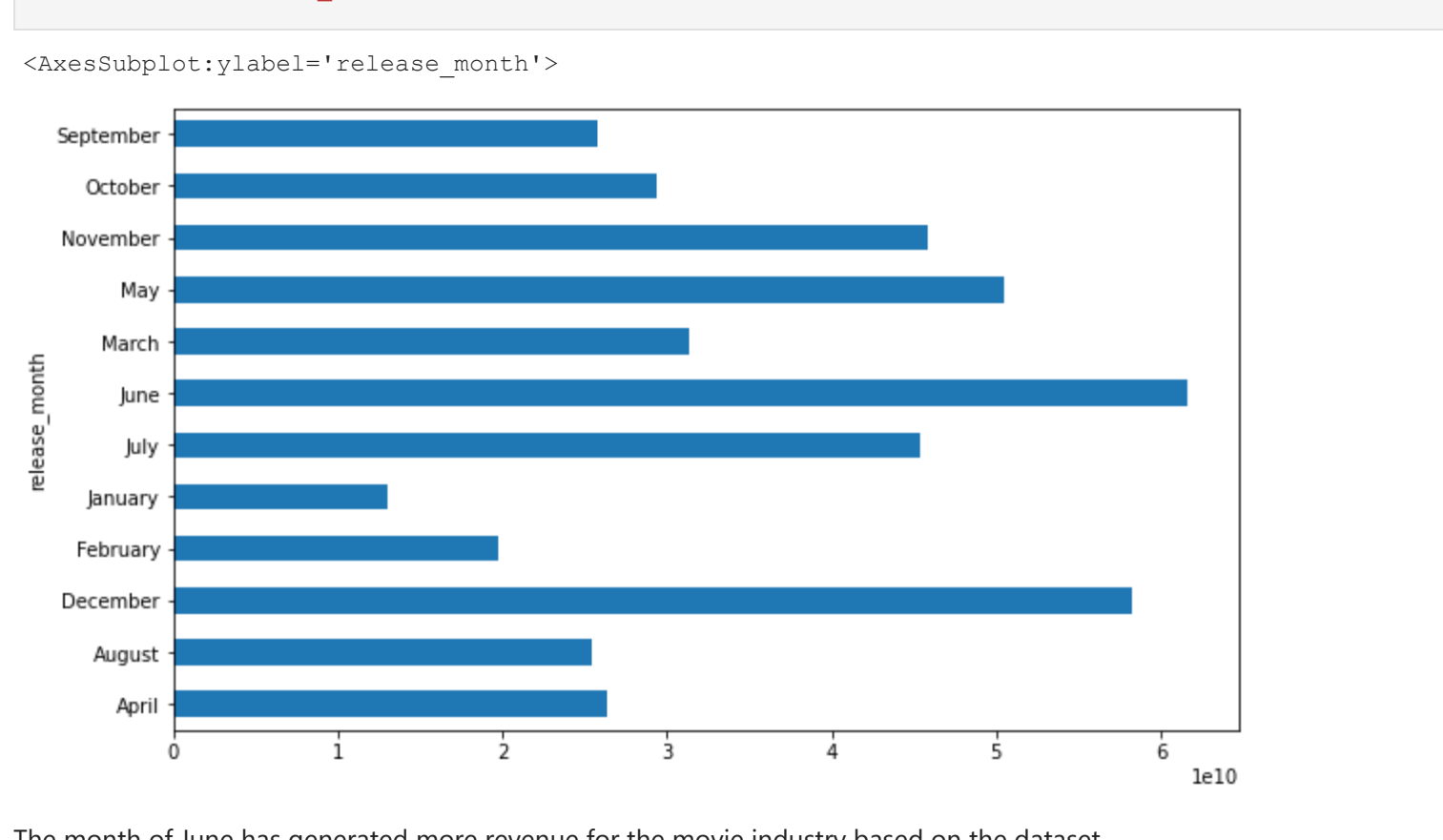
Which month has generated more revenue in the movie industry?

In [35]:
```
df.groupby('release_month')['revenue'].sum()
```

Out[35]:
```
release_month
April        26392671428
August       25477766900
December     58211863204
February     19793785507
January      12968479332
July         45337825481
June         61660585217
March        31393443375
May          50454865815
November     45896250786
October      29353709677
September    25731466985
Name: revenue, dtype: int64
```

In [36]:
```
df.groupby('release_month')['revenue'].sum().plot(kind='barh', figsize=(10,6))
```

Out[36]:
```
<AxesSubplot:ylabel='release_month'>
```



The month of June has generated more revenue for the movie industry based on the dataset

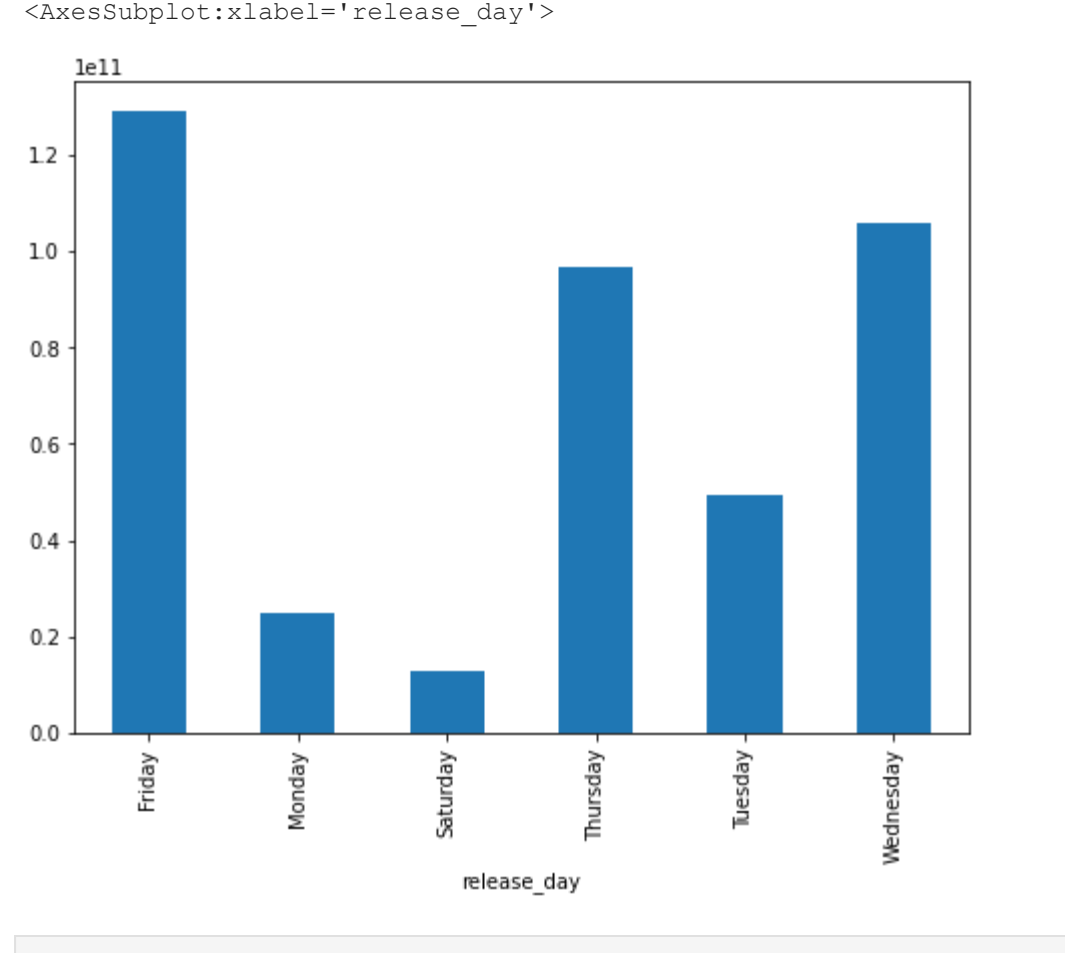
Does the number of movies released translate to more revenue?

In [37]:
```
df.groupby('release_day')['revenue'].sum()
```

Out[37]:
```
release_day
Friday       128852378314
Monday        24959065456
Saturday      13060306270
Thursday      96627259140
Tuesday       49507764714
Wednesday    105661470111
Name: revenue, dtype: int64
```

In [38]:
```
df.groupby('release_day')['revenue'].sum().plot(kind='bar', figsize=(8,6))
```

Out[38]:
```
<AxesSubplot:xlabel='release_day'>
```

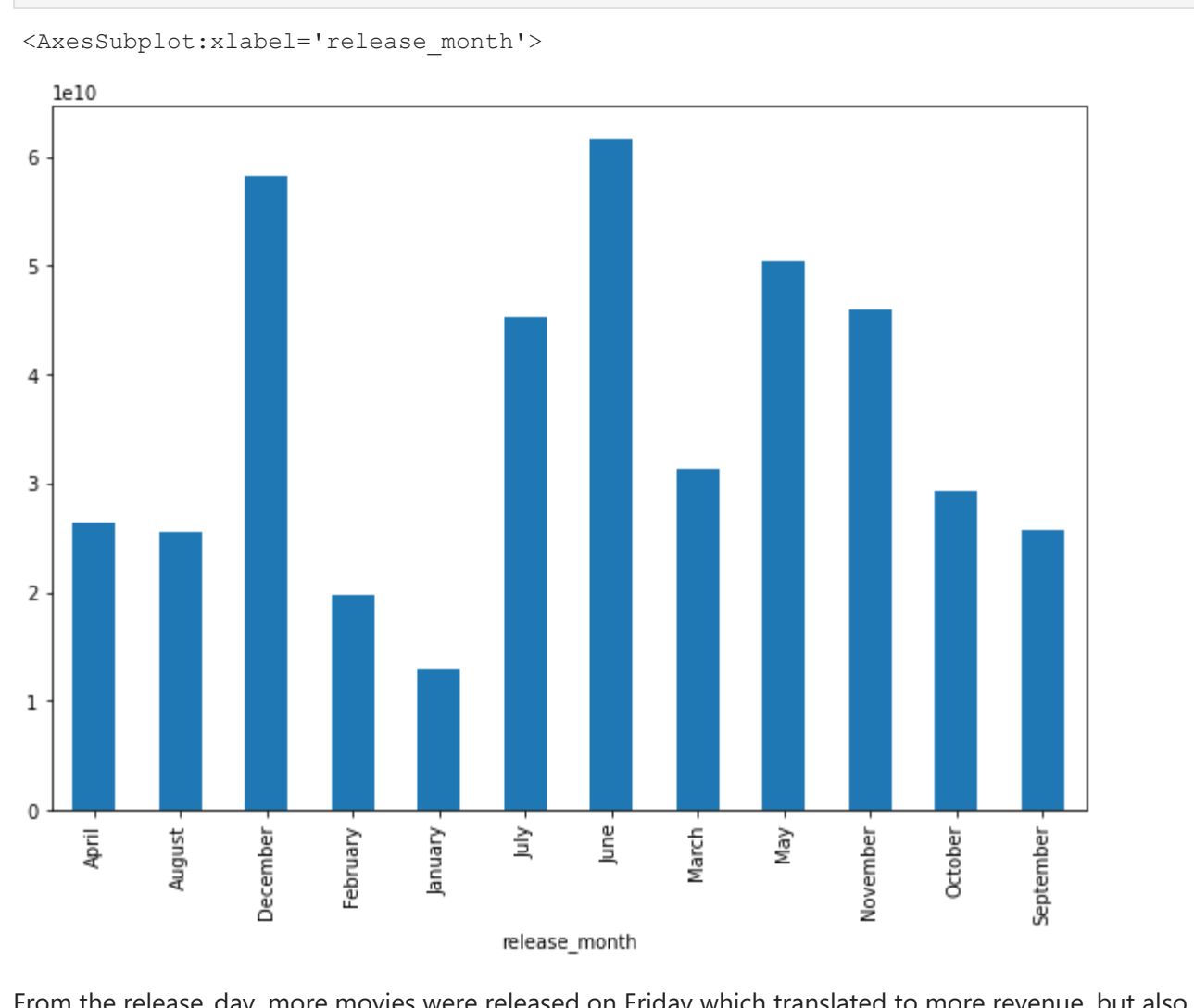


In [39]:
```
df.groupby('release_month')['revenue'].sum()
```

Out[39]:
```
release_month
April        26392671428
August       25477766900
December     58211863204
February     19793785507
January      12968479332
July         45337825481
June         61660585217
March        31393443375
May          50454865815
November     45896250786
October      29353709677
September    25731466985
Name: revenue, dtype: int64
```

In [40]:
```
df.groupby('release_month')['revenue'].sum().plot(kind='bar', figsize=(10,7))
```

Out[40]:
```
<AxesSubplot:xlabel='release_month'>
```



From the release_day, more movies were released on Friday which translated to more revenue, but also more movies were released in the month of September but this didn't translate to more revenue hence we can't say from the analysis if more movies released can translate to more revenue.

Does the vote_count affect revenue

In [41]:
```
df['revenue'].max()
```

Out[41]:
```
2781505847
```

In [42]:
```
df['revenue'].min()
```

Out[42]:
```
0
```

In [43]:
```
df['vote_count'].max()
```
Out[43]: 9767

In [44]:
```
df['vote_count'].min()
```
Out[44]: 10

In [45]:
```
df[df['vote_count'] ==9767]
```
Out[45]:

<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th>keyw</th></tr>
<tr><td>1919</td><td>27205</td><td>tt1375666</td><td>9.363643</td><td>160000000</td><td>825500000</td><td>Inception</td><td>Leonardo DiCaprio|Joseph Gordon-Levitt|Ellen P...</td><td>Christopher Nolan</td><td>Your mind is the scene of the crime.</td><td>lo lover|dream|sleep|subconsciousness|</td></tr>
</table>

1 rows × 22 columns

In [46]:
```
df[df['revenue'] == 2781505847]
```
Out[46]:

<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th>keywords</th><th>...</th><th></th></tr>
<tr><td>1386</td><td>19995</td><td>tt0499549</td><td>9.432768</td><td>237000000</td><td>2781505847</td><td>Avatar</td><td>Sam Worthington|Zoe Saldana|Sigourney Weaver|S...</td><td>James Cameron</td><td>Enter the World of Pandora.</td><td>culture clash|future|space war|space colony|so...</td><td>...</td><td>Action|Adv</td></tr>
</table>

1 rows × 22 columns

In [47]:
```
df[df['vote_count'] == 10]
```
Out[47]:

<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th>keywords</th></tr>
<tr><td>240</td><td>363689</td><td>tt4229298</td><td>0.532700</td><td>0</td><td>0</td><td>The Unspoken</td><td>Jodelle Ferland|Sunny Suljic|Neal McDonough|Ma...</td><td>Sheldon Wilson</td><td>Unknown</td><td>Unknown</td></tr>
<tr><td>259</td><td>360603</td><td>tt5133572</td><td>0.476341</td><td>0</td><td>0</td><td>Crown for Christmas</td><td>Danica McKellar|Rupert Penry-Jones|Ellie Botte...</td><td>Alex Zamm</td><td>Unknown</td><td>Unknown</td></tr>
<tr><td>298</td><td>354220</td><td>tt3826866</td><td>0.370258</td><td>0</td><td>0</td><td>The Girl in the Photographs</td><td>Kal Penn|Claudia Lee|Kenny Wormald|Toby Heming...</td><td>Nick Simon</td><td>Unknown</td><td>serial killer|tied feet|tied up while barefoot</td></tr>
<tr><td>345</td><td>361263</td><td>tt4505830</td><td>0.342530</td><td>0</td><td>0</td><td>Jesse Stone: Lost in Paradise</td><td>Tom Selleck|William Devane|Gloria Reuben|Luke ...</td><td>Robert Harmon</td><td>Unknown</td><td>Unknown</td></tr>
<tr><td>386</td><td>316885</td><td>tt4180576</td><td>0.291244</td><td>0</td><td>0</td><td>Bridgend</td><td>Hannah Murray|Steven Waddington|Adrian Rawlins...</td><td>Jeppe RÃ¸nde</td><td>Unknown</td><td>suicide|adolescence|based on true story|magic ...</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>10802</td><td>54000</td><td>tt0077539</td><td>0.227338</td><td>6727000</td><td>0</td><td>Fedora</td><td>William Holden|Marthe Keller|Hildegard Knef|Jo...</td><td>Billy Wilder</td><td>Youth had been a habit of hers for so long tha...</td><td>hollywood|actor|down on his luck</td></tr>
<tr><td>10808</td><td>40060</td><td>tt0077904</td><td>0.129123</td><td>0</td><td>0</td><td>The Manitou</td><td>Tony Curtis|Michael Ansara|Susan Strasberg|Ste...</td><td>William Girdler</td><td>Evil does not dieâ€¦ It waits to be re-born!</td><td>female nudity|based on novel|nudity|tarot card...</td></tr>
<tr><td>10809</td><td>31428</td><td>tt0078295</td><td>0.128231</td><td>0</td><td>0</td><td>Someone's Watching Me!</td><td>Lauren Hutton|David Birney|Adrienne Barbeau|Le...</td><td>John Carpenter</td><td>Unknown</td><td>stalker|suspense|stalking|made for television</td></tr>
<tr><td>10812</td><td>31948</td><td>tt0077629</td><td>0.318883</td><td>0</td><td>5438927</td><td>Gray Lady Down</td><td>Charlton Heston|David Carradine|Stacy Keach|Ne...</td><td>David Greene</td><td>Trapped underwater... with time running out.</td><td>submarine|drowning|based on novel|rescue|disaster</td></tr>
<tr><td>10857</td><td>5921</td><td>tt0060748</td><td>0.131378</td><td>0</td><td>0</td><td>Nevada Smith</td><td>Steve McQueen|Karl Malden|Brian Keith|Arthur K...</td><td>Henry Hathaway</td><td>Some called him savage- and some called him sa...</td><td>repayment|revenge|native american|wild west|ha...</td></tr>
</table>

489 rows × 22 columns

In [48]:
```
df[df['revenue'] == 0]
```
Out[48]:

<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th></th></tr>
<tr><td>48</td><td>265208</td><td>tt2231253</td><td>2.932340</td><td>30000000</td><td>0</td><td>Wild Card</td><td>Jason Statham|Michael Angarano|Milo Ventimigli...</td><td>Simon West</td><td>Never bet against a man with a killer hand.</td><td>gambling|bo</td></tr>
<tr><td>67</td><td>334074</td><td>tt3247714</td><td>2.331636</td><td>20000000</td><td>0</td><td>Survivor</td><td>Pierce Brosnan|Milla Jovovich|Dylan McDermott|...</td><td>James McTeigue</td><td>His Next Target is Now Hunting Him</td><td>eve|fire|showdown|t</td></tr>
<tr><td>74</td><td>347096</td><td>tt3478232</td><td>2.165433</td><td>0</td><td>0</td><td>Mythica: The Darkspore</td><td>Melanie Stone|Kevin Sorbo|Adam Johnson|Jake St...</td><td>Anne K. Black</td><td>Unknown</td><td>sword|magic|sorcery|necr</td></tr>
<tr><td>75</td><td>308369</td><td>tt2582496</td><td>2.141506</td><td>0</td><td>0</td><td>Me and Earl and the Dying Girl</td><td>Thomas Mann|RJ Cyler|Olivia Cooke|Connie Britt...</td><td>Alfonso Gomez-Rejon</td><td>A Little Friendship Never Killed Anyone.</td><td>school|leukemi</td></tr>
<tr><td>92</td><td>370687</td><td>tt3608646</td><td>1.876037</td><td>0</td><td>0</td><td>Mythica: The Necromancer</td><td>Melanie Stone|Adam Johnson|Kevin Sorbo|Nicola ...</td><td>A. Todd Smith</td><td>Unknown</td><td>sword|magic|sorce</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td></td></tr>
<tr><td>10861</td><td>21</td><td>tt0060371</td><td>0.080598</td><td>0</td><td>0</td><td>The Endless Summer</td><td>Michael Hynson|Robert August|Lord 'Tally Ho' B...</td><td>Bruce Brown</td><td>Unknown</td><td>surfer|s</td></tr>
<tr><td>10862</td><td>20379</td><td>tt0060472</td><td>0.065543</td><td>0</td><td>0</td><td>Grand Prix</td><td>James Garner|Eva Marie Saint|Yves Montand|Tosh...</td><td>John Frankenheimer</td><td>Cinerama sweeps YOU into a drama of speed and ...</td><td>car race|</td></tr>
<tr><td>10863</td><td>39768</td><td>tt0060161</td><td>0.065141</td><td>0</td><td>0</td><td>Beregis Avtomobilya</td><td>Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...</td><td>Eldar Ryazanov</td><td>Unknown</td><td>car|tr</td></tr>
<tr><td>10864</td><td>21449</td><td>tt0061177</td><td>0.064317</td><td>0</td><td>0</td><td>What's Up, Tiger Lily?</td><td>Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...</td><td>Woody Allen</td><td>WOODY ALLEN STRIKES BACK!</td><td></td></tr>
<tr><td>10865</td><td>22293</td><td>tt0060666</td><td>0.035919</td><td>19000</td><td>0</td><td>Manos: The Hands of Fate</td><td>Harold P. Warren|Tom Neyman|John Reynolds|Dian...</td><td>Harold P. Warren</td><td>It's Shocking! It's Beyond Your Imagination!</td><td>fire|gun|drive|sa</td></tr>
</table>

5881 rows × 22 columns

The vote_count does not affect the revenue as we can see that the minimum revenue is 0 which has quite a large number of vote_count while the minimum value for vote_count is 10 some having revenue of 0 and some having a large amount as revenue

Which movie generated the highest amount of revenue?

In [49]:
```
df['revenue'].max()
```
Out[49]: 2781505847

In [50]:
```
df[df['revenue'] == 2781505847]
```
Out[50]:

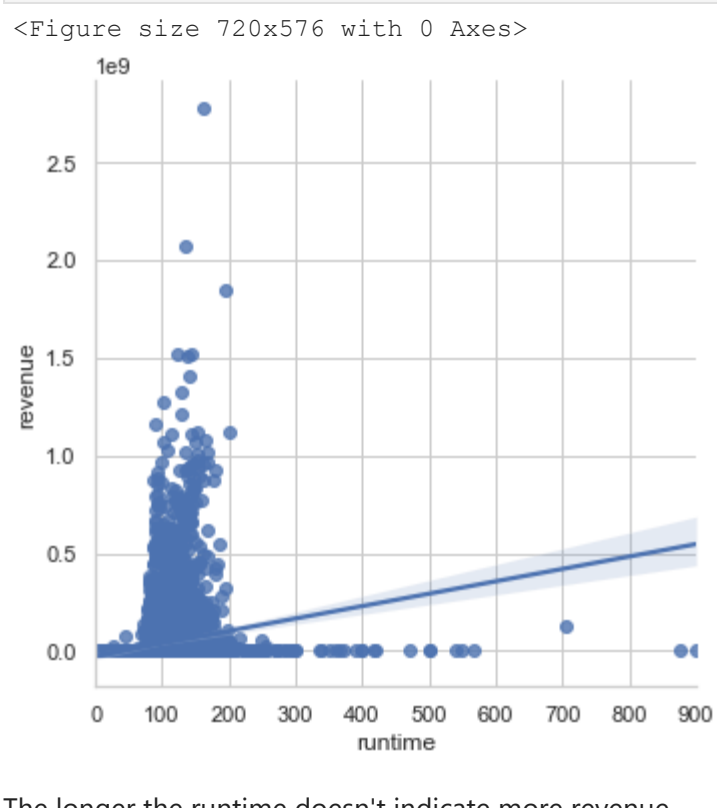
<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th>keywords</th><th>...</th><th></th></tr>
<tr><td>1386</td><td>19995</td><td>tt0499549</td><td>9.432768</td><td>237000000</td><td>2781505847</td><td>Avatar</td><td>Sam Worthington|Zoe Saldana|Sigourney Weaver|S...</td><td>James Cameron</td><td>Enter the World of Pandora.</td><td>culture clash|future|space war|space colony|so...</td><td>...</td><td>Action|Adv</td></tr>
</table>

1 rows × 22 columns

The movie Avatar generated the highest amount of revenue with the amount 2781505847

Does longer runtime indicate more revenue?

In [51]:
```
plt.figure(figsize=(10,8))
sns.set(style="whitegrid")
ax = sns.lmplot(x='runtime', y='revenue', data=df)
```
```
<Figure size 720x576 with 0 Axes>
```



The longer the runtime doesn't indicate more revenue

What is the longest runtime of a movie?

In [52]:
```
df['runtime'].max()
```
Out[52]: 900

The longest runtime for a movie is 900 minutes

Which movie has the longest runtime?

In [53]:
```
df[df['runtime'] == 900]
```
Out[53]:

<table>
<tr><th></th><th>id</th><th>imdb_id</th><th>popularity</th><th>budget</th><th>revenue</th><th>original_title</th><th>cast</th><th>director</th><th>tagline</th><th>keywords</th><th>...</th><th>genres</th><th>produ</th></tr>
<tr><td>3894</td><td>125336</td><td>tt2044056</td><td>0.006925</td><td>0</td><td>0</td><td>The Story of Film: An Odyssey</td><td>Mark Cousins|Jean-Michel Frodon|Cari Beauchamp...</td><td>Mark Cousins</td><td>Unknown</td><td>cinema|nouvelle vague|hindi cinema|cinema novo...</td><td>...</td><td>Documentary</td><td></td></tr>
</table>

1 rows × 22 columns

The Story of Film:An Odyssey has the longest runtime

SUMMARY OF FINDINGS AND CONCLUSION

- More movies were released in the year 2014.
- The month of September recorded the highest number of movies released.
- 42% of the movies were released on Friday.
- The month of June generated more revenue for the movie industry based on the dataset.
- From the release_day, more movies were released on Friday which translated to more revenue, but also more movies were released in the month of September but this didn't translate to more revenue hence we can't say from the analysis if more movies released can translate to more revenue.
- The vote_count does not affect the revenue.
- The longer the runtime doesn't indicate more revenue.
- The longest runtime for a movie is 900 minutes.
- The Story of Film:An Odyssey had the longest runtime.
- The movie Avatar generated the highest amount of revenue.

Conclusion: Basic questions were just answered in this project based on the scope of the work, machine learning model can be built to better predict how more features can affect the target variable.