Project: Data Analysis of High Revenue Movie Titles

Table of Contents

Introduction

>

> This data analysis uses the TMDb movie data set that has information for about ten thousand movies released from 1960 to 2015. In this project, we'll be investigating data associated with the qualities of high revenue movie titles. In particular, we'll be interested in finding trends among movies with the high revenues and how they differed from those with low revenues.
>
> Information that can be obtained from this data analysis include average runtime, top genres, and best month for high revenue movies. Some questions that can be answered through this data analysis include:
>
> 1. What are the top 3 genres associated with movies with high revenues?
> 2. What is the average runtime for movies with high revenues and those with low revenues?
> 3. What month is more probable to produce movies that will generate high revenues?

In [1]:
```
#Set up import statements for packages to use in data analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling

General Properties

In [2]:
```
#Load and view dataset
df_tmdb = pd.read_csv('tmdb-movies.csv')
df_tmdb.head()
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	htt
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	

5 rows × 21 columns

In [3]:
```
#Determine the amount of rows and columns
df_tmdb.shape
```

Out[3]: (10866, 21)

In [4]:
```
#Obtain descriptive statistics for columns
df_tmdb.describe()
```

Out[4]:

	id	popularity	budget	revenue	runtime	vote_count	vo
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

In [5]:
```
#Display a summary of the datatypes we are working with
df_tmdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

Observations

> This tmdb movie dataset has 10,866 rows and 21 columns of information. The columns give us informative data about the overview, production, and other miscellaneous details of 10,866 movies that were released from 1960 to 2015.

> From looking over this data set, I observed a few minor complications to analyzing the information provided. I first noticed that some of the structure of the data lacks an unit of measure such as those focused on popularity ratings. The vote_count column is difficult to compare from each movie as the range is over 9000, so there is a large varaibility with this. There is also no indication of how the vote_average and popularity columns were calculated and so that brings up a question of how accurate its interpretation of numbers and qualitative value would be.
>
> To be consistent throughout, I will be using the budget_adj and revenue_adj columns for my analysis instead of budget and revenue columns. Luckily, the data for budget_adj and revenue_adj has been adjusted to all be in terms of 2010 dollars to account for inflation overtime.
>
> After cleaning up the data, we will have a better data set to work with to obtain significant analytical information.

Data Cleaning

> To clean this data, I will first remove columns that are not relevant to this particular data analysis to examine the qualities of high revenue movies compared to low revenue movies. These columns include id, imdb_id, popularity, budget, revenue, cast, homepage, director, tagline, keywords, overview, production_companies, vote_count, and vote_average.
>
> Next, I would remove any duplicates among the rows, if any. The following step would be to drop rows with null values and replacing zero values with the mean value. The columns in need for this would include runtime, budget_adj, and revenue_adj. After, I would fix any orignial formatting to better analyze my data such as the release_date. Then, I will be able to perform and complete a thorough data analysis on a clean data set.

1. Remove extraneous columns

Columns to be deleted: id, imdb_id, popularity, budget, revenue, cast, homepage, director, tagline, keywords, overview, production_companies, vote_count, and vote_average.

In [6]:
```
#create a list of extraneous columns to delete
delete= ['id', 'imdb_id', 'cast', 'popularity', 'budget', 'revenue', 'homepag
e', 'director', 'tagline','keywords', 'overview', 'production_companies', 'vot
e_count', 'vote_average']

#delete columns with drop function
df_tmdb.drop(delete, axis=1, inplace=True)
#confirm changes
df_tmdb.head(5)
```

Out[6]:

	original_title	runtime	genres	release_date	release_year	budget_adj	reve
0	Jurassic World	124	Action|Adventure|Science Fiction|Thriller	6/9/15	2015	1.379999e+08	1.392
1	Mad Max: Fury Road	120	Action|Adventure|Science Fiction|Thriller	5/13/15	2015	1.379999e+08	3.481
2	Insurgent	119	Adventure|Science Fiction|Thriller	3/18/15	2015	1.012000e+08	2.716
3	Star Wars: The Force Awakens	136	Action|Adventure|Science Fiction|Fantasy	12/15/15	2015	1.839999e+08	1.902
4	Furious 7	137	Action|Crime|Thriller	4/1/15	2015	1.747999e+08	1.385

In [7]:
```
#Load new amount of columns
df_tmdb.shape
```

Out[7]: (10866, 7)

2. Remove duplicates

In [8]:
```
#drop duplicates
df_tmdb.drop_duplicates(inplace=True)

#confirm if any changes
df_tmdb.shape
```

Out[8]: (10865, 7)

After dropping duplicates, one row was removed. Now, we have 10865 rows and 7 columns to further clean with.

In [9]:
```
#Look for any columns with null values
df_tmdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 7 columns):
original_title    10865 non-null object
runtime           10865 non-null int64
genres            10842 non-null object
release_date      10865 non-null object
release_year      10865 non-null int64
budget_adj        10865 non-null float64
revenue_adj       10865 non-null float64
dtypes: float64(2), int64(2), object(3)
memory usage: 679.1+ KB
```

3. Drop null values for columns with missing data

There are null values for the genres columns. I will remove these for better analysis.

In [10]:
```
#drop rows with null values for the genre column
df_tmdb.dropna(inplace=True)
```

In [11]:
```
#confirm changes
df_tmdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10842 entries, 0 to 10865
Data columns (total 7 columns):
original_title    10842 non-null object
runtime           10842 non-null int64
genres            10842 non-null object
release_date      10842 non-null object
release_year      10842 non-null int64
budget_adj        10842 non-null float64
revenue_adj       10842 non-null float64
dtypes: float64(2), int64(2), object(3)
memory usage: 677.6+ KB
```

In [12]:
```
#confirm changes
df_tmdb.shape
```

Out[12]:
```
(10842, 7)
```

In [13]:
```
#confirm changes
df_tmdb.isnull().sum()
```

Out[13]:
```
original_title    0
runtime           0
genres            0
release_date      0
release_year      0
budget_adj        0
revenue_adj       0
dtype: int64
```

4. Replace zero values with mean

> The columns with zero values include runtime, budget_adj, and revenue_adj. I will replace the zero values in these columns with their means.

In [14]:
```
#Run calculations on the runtime column for the mean
runtime_average=df_tmdb['runtime'].mean()
runtime_average
```

Out[14]:
```
102.1384430916805
```

In [15]:
```
#Replace zero values with the mean for runtime column
df_tmdb['runtime'] = df_tmdb['runtime'].replace(0, 102.1384430916805)
```

In [16]:
```
#Run calculations on the budget_adj column for the mean
budget_adj_average=df_tmdb['budget_adj'].mean()
budget_adj_average
```

Out[16]:
```
17587121.438262936
```

In [17]:
```
#Replace zero value with the mean for budget_adj column
df_tmdb['budget_adj'] = df_tmdb['budget_adj'].replace(0, 17587121.438262936)
```

In [18]:
```
#Run calculations on the revenue_adj column for the mean
revenue_adj_average=df_tmdb['revenue_adj'].mean()
revenue_adj_average
```

Out[18]:
```
51477974.92250734
```

In [19]:
```
#Replace zero value with the mean for revenue_adj column
df_tmdb['revenue_adj'] = df_tmdb['revenue_adj'].replace(0, 51477974.92250734)
```

In [20]:
```
#confirm changes
df_tmdb.head()
```

Out[20]:

	original_title	runtime	genres	release_date	release_year	budget_adj	reve
0	Jurassic World	124.0	Action|Adventure|Science Fiction|Thriller	6/9/15	2015	1.379999e+08	1.392
1	Mad Max: Fury Road	120.0	Action|Adventure|Science Fiction|Thriller	5/13/15	2015	1.379999e+08	3.481
2	Insurgent	119.0	Adventure|Science Fiction|Thriller	3/18/15	2015	1.012000e+08	2.716
3	Star Wars: The Force Awakens	136.0	Action|Adventure|Science Fiction|Fantasy	12/15/15	2015	1.839999e+08	1.902
4	Furious 7	137.0	Action|Crime|Thriller	4/1/15	2015	1.747999e+08	1.385

In [21]:
```
#Look over columns
df_tmdb.describe()
```

Out[21]:

	runtime	release_year	budget_adj	revenue_adj
count	10842.000000	10842.000000	1.084200e+04	1.084200e+04
mean	102.421062	2001.314794	2.679108e+07	7.993283e+07
std	30.828622	12.813617	3.053264e+07	1.366907e+08
min	2.000000	1960.000000	9.210911e-01	2.370705e+00
25%	90.000000	1995.000000	1.758712e+07	5.147797e+07
50%	99.000000	2006.000000	1.758712e+07	5.147797e+07
75%	111.000000	2011.000000	2.092507e+07	5.147797e+07
max	900.000000	2015.000000	4.250000e+08	2.827124e+09

In [22]:
```
#look over columns with object datatypes
df_tmdb.describe(include = ['O'])
```

Out[22]:

	original_title	genres	release_date
count	10842	10842	10842
unique	10548	2039	5904
top	Hamlet	Drama	1/1/09
freq	4	712	28

In [23]:
```
#Check object datatypes
df_tmdb['release_date'] = pd.to_datetime(df_tmdb['release_date'])
df_tmdb.dtypes
```

Out[23]:
```
original_title             object
runtime                   float64
genres                     object
release_date       datetime64[ns]
release_year                int64
budget_adj                float64
revenue_adj               float64
dtype: object
```

In [24]:
```
#Check changes
df_tmdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10842 entries, 0 to 10865
Data columns (total 7 columns):
original_title    10842 non-null object
runtime           10842 non-null float64
genres            10842 non-null object
release_date      10842 non-null datetime64[ns]
release_year      10842 non-null int64
budget_adj        10842 non-null float64
revenue_adj       10842 non-null float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(2)
memory usage: 677.6+ KB
```

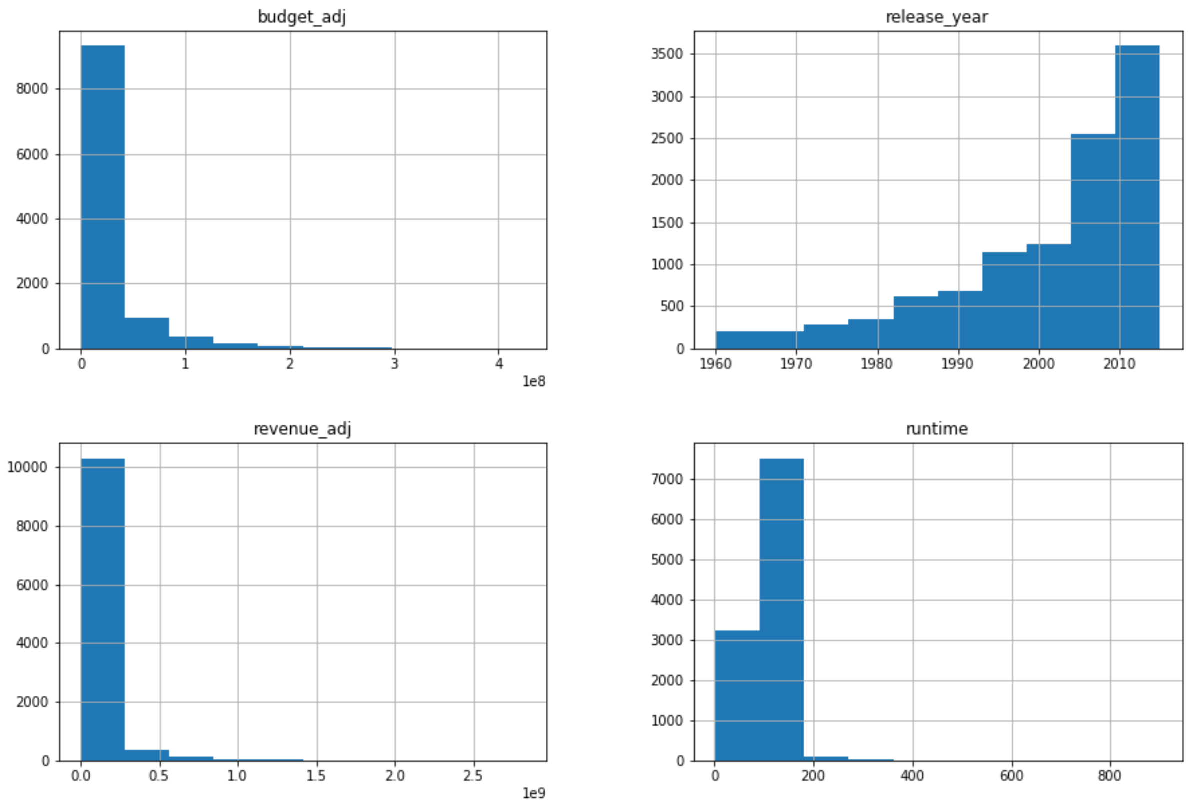
In [25]:
```
#Check new datetime format for release_date
df_tmdb.head()
```

Out[25]:

	original_title	runtime	genres	release_date	release_year	budget_adj	reve
0	Jurassic World	124.0	Action|Adventure|Science Fiction|Thriller	2015-06-09	2015	1.379999e+08	1.392
1	Mad Max: Fury Road	120.0	Action|Adventure|Science Fiction|Thriller	2015-05-13	2015	1.379999e+08	3.481
2	Insurgent	119.0	Adventure|Science Fiction|Thriller	2015-03-18	2015	1.012000e+08	2.716
3	Star Wars: The Force Awakens	136.0	Action|Adventure|Science Fiction|Fantasy	2015-12-15	2015	1.839999e+08	1.902
4	Furious 7	137.0	Action|Crime|Thriller	2015-04-01	2015	1.747999e+08	1.385

Exploratory Data Analysis

In [26]:
```
#Explore data to obtain general idea of the kind of data we are working with
 (skewness, frequency, range)
#Also see the best approach for answering research questions
df_tmdb.hist(figsize=(15,10));
```



Based on these histograms, we obtain a good general idea of the kind of data we are working with. These histograms are for our integer and float datatype columns.

For the budget_adj column and revenue_adj column: the data is right-skewed in which the median and mean is greater the mode and the median is greater than the mean. Most data values are under the same bin range as show in the histogram.

For the release_year column: there is a positive relationship that as the years progressed, more movies were released.

For the runtime column: the mode of all movies in the data set is between the range of 100-200 minutes so over 7000 of about 10,000 movies in this data set is within that range. The average runtime is likely to be in this range.

Research Question 1: What top 3 genres are associated with movies with high revenues?

In [27]:
```
#Make a copy of dataframe to have an original dataframe to answer the followin
g research questions
#To approach this question, we first must split the values in the genres colum
ns as they hold multiple values.

df_genre=df_tmdb.copy()
genres = df_genre['genres'].str.split('|', expand=True)
```

In [28]:
```
#Drop original genres column
df_genre.drop('genres', axis=1, inplace=True)
```

In [29]:
```
#Merge new genre columns to tmdb dataframe in place of the original
df_genre = pd.merge(df_genre, genres, left_index=True, right_index=True, how=
'inner')
```

In [30]:
```
#Find the top 15 movies with the highest revenues
top_15 = df_genre.nlargest(15, 'revenue_adj')
top_15
```

Out[30]:

	original_title	runtime	release_date	release_year	budget_adj	revenue_adj	0
1386	Avatar	162.0	2009-12-10	2009	2.408869e+08	2.827124e+09	Action
1329	Star Wars	121.0	1977-03-20	1977	3.957559e+07	2.789712e+09	Adventure
5231	Titanic	194.0	1997-11-18	1997	2.716921e+08	2.506406e+09	Drama
10594	The Exorcist	122.0	1973-12-26	1973	3.928928e+07	2.167325e+09	Drama
9806	Jaws	124.0	1975-06-18	1975	2.836275e+07	1.907006e+09	Horror
3	Star Wars: The Force Awakens	136.0	2015-12-15	2015	1.839999e+08	1.902723e+09	Action
8889	E.T. the Extra-Terrestrial	115.0	1982-04-03	1982	2.372625e+07	1.791694e+09	Science Fiction
8094	The Net	114.0	1995-07-28	1995	3.148127e+07	1.583050e+09	Crime
10110	One Hundred and One Dalmatians	79.0	2061-01-25	1961	2.917944e+07	1.574815e+09	Adventure
4361	The Avengers	143.0	2012-04-25	2012	2.089437e+08	1.443191e+09	Science Fiction
7309	The Empire Strikes Back	124.0	1980-01-01	1980	4.762866e+07	1.424626e+09	Adventure
0	Jurassic World	124.0	2015-06-09	2015	1.379999e+08	1.392446e+09	Action
10223	Jurassic Park	127.0	1993-06-11	1993	9.509661e+07	1.388863e+09	Adventure
4	Furious 7	137.0	2015-04-01	2015	1.747999e+08	1.385749e+09	Action
10398	The Jungle Book	78.0	2067-10-18	1967	2.614705e+07	1.345551e+09	Family

In [31]:
```
#Create a copy of the top 15 results to drop columns that are not useful to an
swer research question
#Show new table with dropped columns
copy_tmdb = top_15.copy()
copy_tmdb.drop(['original_title', 'runtime', 'release_date', 'release_year',
'budget_adj', 'revenue_adj'], axis=1, inplace=True)
copy_tmdb
```

Out[31]:

	0	1	2	3	4
1386	Action	Adventure	Fantasy	Science Fiction	None
1329	Adventure	Action	Science Fiction	None	None
5231	Drama	Romance	Thriller	None	None
10594	Drama	Horror	Thriller	None	None
9806	Horror	Thriller	Adventure	None	None
3	Action	Adventure	Science Fiction	Fantasy	None
8889	Science Fiction	Adventure	Family	Fantasy	None
8094	Crime	Drama	Mystery	Thriller	Action
10110	Adventure	Animation	Comedy	Family	None
4361	Science Fiction	Action	Adventure	None	None
7309	Adventure	Action	Science Fiction	None	None
0	Action	Adventure	Science Fiction	Thriller	None
10223	Adventure	Science Fiction	None	None	None
4	Action	Crime	Thriller	None	None
10398	Family	Animation	Adventure	None	None

In [32]:
```
#Obtain the frequency of the count value in each row of the above table
genre_count = copy_tmdb.melt()
genre_table = pd.crosstab(index=genre_count['value'], columns=genre_count['variable'])
genre_table
```

Out[32]:

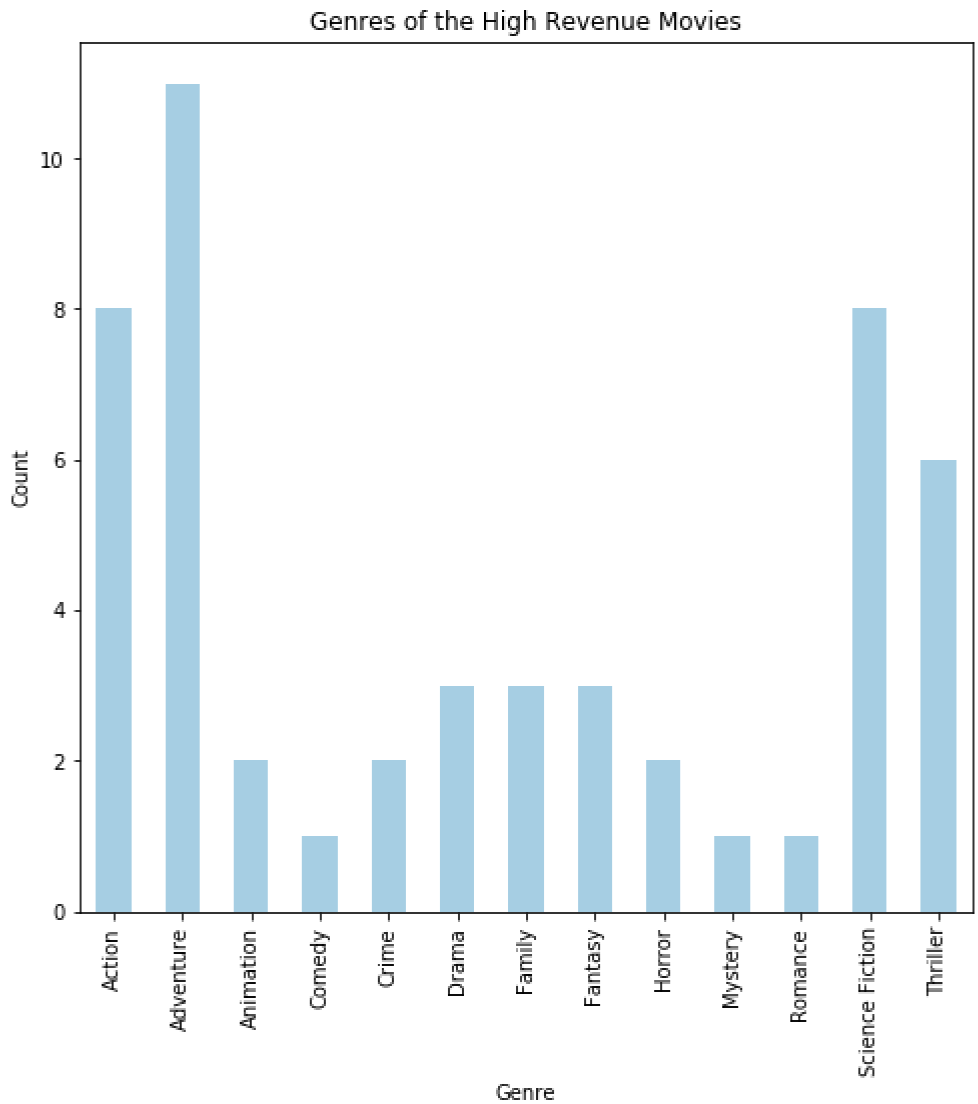
variable	0	1	2	3	4
value					
Action	4	3	0	0	1
Adventure	4	4	3	0	0
Animation	0	2	0	0	0
Comedy	0	0	1	0	0
Crime	1	1	0	0	0
Drama	2	1	0	0	0
Family	1	0	1	1	0
Fantasy	0	0	1	2	0
Horror	1	1	0	0	0
Mystery	0	0	1	0	0
Romance	0	1	0	0	0
Science Fiction	2	1	4	1	0
Thriller	0	1	3	2	0

In [33]:
```
#Create a Total column for this table to generate a bar plot and show results
genre_table['Total'] = genre_table.sum(axis=1)
genre_table
```

Out[33]:

variable value	0	1	2	3	4	Total
Action	4	3	0	0	1	8
Adventure	4	4	3	0	0	11
Animation	0	2	0	0	0	2
Comedy	0	0	1	0	0	1
Crime	1	1	0	0	0	2
Drama	2	1	0	0	0	3
Family	1	0	1	1	0	3
Fantasy	0	0	1	2	0	3
Horror	1	1	0	0	0	2
Mystery	0	0	1	0	0	1
Romance	0	1	0	0	0	1
Science Fiction	2	1	4	1	0	8
Thriller	0	1	3	2	0	6

In [34]:
```
# Plot in a bar chart and use one color for bars to provide easy reading of pl
ot
genre_table['Total'].plot(kind="bar", figsize=(8,8), fontsize=10, colormap='Pa
ired')
plt.xlabel('Genre', fontsize = 10)
plt.ylabel('Count', fontsize = 10)
plt.title('Genres of the High Revenue Movies', fontsize = 12);
```



For this bar plot, I wanted to use one color to note that there are no other variables taken into account with the use of different colors, they are all under genres. I create a table below to show descending order of the most popular genres of high revenue movies.

In [35]:
```
#Create another table to display values of count in descending order to see th
e top 3 genres of high revenue movies.
genre_table['Total'].sort_values(ascending=False)
```

Out[35]:
```
value
Adventure          11
Science Fiction     8
Action              8
Thriller            6
Fantasy             3
Family              3
Drama               3
Horror              2
Crime               2
Animation           2
Romance             1
Mystery             1
Comedy              1
Name: Total, dtype: int64
```

> From this bar graph and count table, it is shown that the top 3 genres of high revenue movies are Adventure with 11, Science Fiction with 8, and Action with 8. This analysis tells us that movie producers are more likely to generate high revenues for movies if they produce movies that can classify among these genres; Adventure, Science Fiction, or Action.
>
> I originally wanted to look at the genres associated with the highest revenues, but through my process of analysis, I found that I had to look at the top 3 since the count range is not big. Therefore, I had to change my question to 'What are the top 3 genres associated with high revenue movies?' to obtain useful information.

Research Question 2: What is the average runtime for movies with high revenues and those with low revenues?

In [36]:
```
#Create a copy dataframe to use for this research question
df_runtime=df_tmdb.copy()
```

In [37]:
```
#Find average runtime for all movies, this was completed in Data Cleaning sect
ion
runtime_average=df_tmdb['runtime'].mean()
runtime_average
```

Out[37]:
```
102.42106191595185
```

In [38]:
```
#Find the 10 top movies with highest revenue generated
top_10=df_runtime.nlargest(10, 'revenue_adj')
top_10
```

Out[38]:

	original_title	runtime	genres	release_date	release_year	bu
1386	Avatar	162.0	Action|Adventure|Fantasy|Science Fiction	2009-12-10	2009	2.408
1329	Star Wars	121.0	Adventure|Action|Science Fiction	1977-03-20	1977	3.957
5231	Titanic	194.0	Drama|Romance|Thriller	1997-11-18	1997	2.716
10594	The Exorcist	122.0	Drama|Horror|Thriller	1973-12-26	1973	3.928
9806	Jaws	124.0	Horror|Thriller|Adventure	1975-06-18	1975	2.836
3	Star Wars: The Force Awakens	136.0	Action|Adventure|Science Fiction|Fantasy	2015-12-15	2015	1.839
8889	E.T. the Extra-Terrestrial	115.0	Science Fiction|Adventure|Family|Fantasy	1982-04-03	1982	2.372
8094	The Net	114.0	Crime|Drama|Mystery|Thriller|Action	1995-07-28	1995	3.148
10110	One Hundred and One Dalmatians	79.0	Adventure|Animation|Comedy|Family	2061-01-25	1961	2.917
4361	The Avengers	143.0	Science Fiction|Action|Adventure	2012-04-25	2012	2.089

In [39]:
```
#Create a copy of the top 10 results to drop columns that are not useful to an
swer research question
#Show new table with dropped columns
copy2_tmdb = top_10.copy()
copy2_tmdb.drop(['original_title', 'release_date', 'release_year', 'budget_ad
j', 'revenue_adj', 'genres'], axis=1, inplace=True)
copy2_tmdb
```

Out[39]:

	runtime
1386	162.0
1329	121.0
5231	194.0
10594	122.0
9806	124.0
3	136.0
8889	115.0
8094	114.0
10110	79.0
4361	143.0

In [40]:
```
#Find the average runtime of the top 10 movies with high revenues
copy2_tmdb['runtime'].mean()
```

Out[40]:
```
131.0
```

In [41]:
```
#Find the 10 bottom movies with lowest revenue generated
bottom_10=df_runtime.nsmallest(10, 'revenue_adj')
bottom_10
```

Out[41]:

	original_title	runtime	genres	release_date	release_year	budg
5067	Shattered Glass	94.0	Drama|History	2003-11-14	2003	7.11211
8142	Mallrats	94.0	Romance|Comedy	1995-10-20	1995	8.58580
3239	Dr. Horrible's Sing-Along Blog	42.0	Adventure|Action|Comedy|Science Fiction|Music	2008-07-15	2008	2.02557
5162	Kid's Story	15.0	Science Fiction|Animation	2003-06-02	2003	1.18535
8523	Bordello of Blood	87.0	Horror|Comedy	1996-08-16	1996	2.08532
8226	Never Talk to Strangers	86.0	Thriller|Romance	1995-10-20	1995	9.15818
10307	The House of the Spirits	140.0	Romance|Drama	1993-10-19	1993	3.77367
3283	Parlami D'Amore	109.0	Comedy|Romance	2008-02-14	2008	1.75871
2252	Elektra Luxx	98.0	Action|Comedy|Drama	2010-03-14	2010	1.75871
5852	Hross Ã oss	85.0	Drama|Romance|Comedy	2013-08-30	2013	9.36033

In [42]:
```
#Create a copy of the bottom 10 results to drop columns that are not useful to
answer research question
#Show new table with dropped columns
copy3_tmdb = bottom_10.copy()
copy3_tmdb.drop(['original_title', 'release_date', 'release_year', 'budget_ad
j', 'revenue_adj', 'genres'], axis=1, inplace=True)
copy3_tmdb
```

Out[42]:

	runtime
5067	94.0
8142	94.0
3239	42.0
5162	15.0
8523	87.0
8226	86.0
10307	140.0
3283	109.0
2252	98.0
5852	85.0

In [43]:
```
#Find the average runtime of the bottom 10 movies with low revenues
copy3_tmdb['runtime'].mean()
```

Out[43]: 85.0

From these calculations, I found that the average runtime for the top 10 movies with high revenue to be 131 minutes and then movies that generated low revenue averaged a runtime of 85 minutes. The difference between these is 46 minutes and this may signify that movies with longer runtime up to 131 minutes, or a little over two hours are more likely to generate higher revenue. Then, movies that are close to 85 minutes or a little over one hour are more likely to generate low revenue. The average runtime for all movies was 102 minutes.

A movie with a greater runtime could the opportunity to offer more context and story development for people to enjoy while a low runtime can rush the storyline, leaving viewers confused and uninterested.

Research Question 3: What month is more probable to produce movies that will generate high revenues?

In [44]:
```
#Create a copy dataframe
df_month=df_tmdb.copy()
```

In [45]:
```
#To create a new column for month, we have to obtain this data from the releas
e_date column
df_month['month'] = df_month['release_date'].apply(lambda x: x.month)
df_month.head()
```

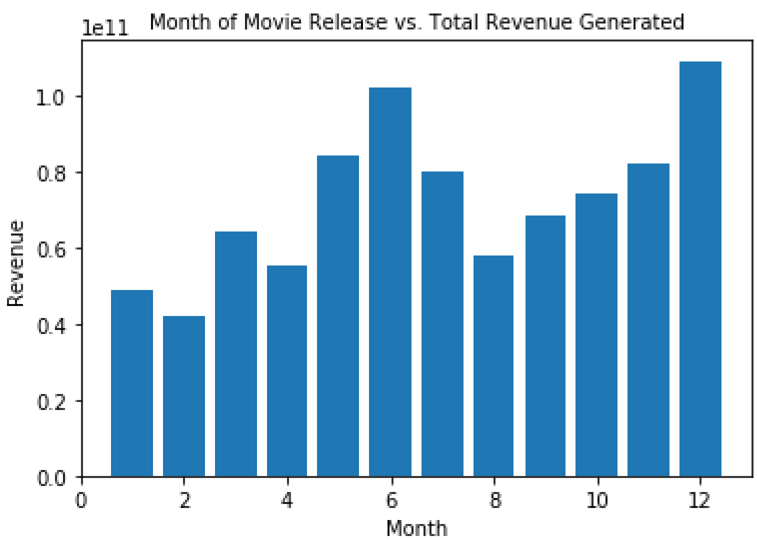
Out[45]:

	original_title	runtime	genres	release_date	release_year	budget_adj	reve
0	Jurassic World	124.0	Action|Adventure|Science Fiction|Thriller	2015-06-09	2015	1.379999e+08	1.392
1	Mad Max: Fury Road	120.0	Action|Adventure|Science Fiction|Thriller	2015-05-13	2015	1.379999e+08	3.481
2	Insurgent	119.0	Adventure|Science Fiction|Thriller	2015-03-18	2015	1.012000e+08	2.716
3	Star Wars: The Force Awakens	136.0	Action|Adventure|Science Fiction|Fantasy	2015-12-15	2015	1.839999e+08	1.902
4	Furious 7	137.0	Action|Crime|Thriller	2015-04-01	2015	1.747999e+08	1.385

In [46]:
```
#Group each month by the sum of revenue
monthly_rev = df_month.groupby('month')['revenue_adj'].sum()
monthly_rev
```

Out[46]:
```
month
1     4.902253e+10
2     4.219032e+10
3     6.385701e+10
4     5.492190e+10
5     8.423072e+10
6     1.021281e+11
7     7.987225e+10
8     5.752671e+10
9     6.807348e+10
10    7.406795e+10
11    8.176612e+10
12    1.089747e+11
Name: revenue_adj, dtype: float64
```

In [47]:
```
#Create a bar plot to show the relationship between month of movie release and total revenue generated
plt.bar([1,2,3,4,5,6,7,8,9,10,11,12], monthly_rev)
plt.title('Month of Movie Release vs. Total Revenue Generated', fontsize = 10)
plt.xlabel('Month')
plt.ylabel('Revenue');
```



In [48]:
```
#Generate a list of monthly totals to obtain exact amounts of movies released
 in a certain month
monthly_totals=df_month['month'].value_counts(ascending=False)
monthly_totals
```

Out[48]:
```
9     1330
10    1148
12     981
1      916
8      916
6      826
3      821
11     814
5      808
7      798
4      797
2      687
Name: month, dtype: int64
```

In [49]:
```
#Find the average for amount of movies released in a month.
month_avg=df_month['month'].value_counts().mean()
month_avg
```

Out[49]:
```
903.5
```

From the bar plot, the month most probable to release a movie to generate high revenue would be in December. June comes in second and November for third. This suggests that movie producers should aim to release movies during the end of the year and holiday seasons, times when people often come together in celebration. June is a month where more people have free time for leisure activities such as watching movies, so movie producers can also consider this time of year as well.

Conclusions

In summary, the qualities of high revenue movie titles include an average runtime of 131 minutes, time of release in December, and under the genres: Adventure, Science Fiction, and Action. This information is useful for movie producers to know how long their movies should run, what kind of genre to direct the movie toward, and in choosing which month or time of year is best to release the movie for public showing. This is assuming all movie producers are looking to receive high revenues from their movie production but there are limitations to my exploration that may not be reflective to true qualities of high revenue movie titles.

The limitations of this exploration include multi-categorized movie genres, personal preferences of movie runtimes, and amount of movies released the same month.

Most of the movies were categorized in multiple genres, so there may have been movies that actually generated low revenues that were taken into account to be under 'high revenue' movie titles because the movie genre was in a sub-category. For instance, there was a movie that was classified to be both under Action and Comedy, that was actually a low revenue movie. This was observed in my findings for bottom_10. An example for this limitation is found for the movie: Elektra Luxx (under Action and Comedy genre) which was the second lowest on the bottom 10. Action was tied as second highest with Science Fiction for the top 3 genres for high revenue movie titles. Comedy was one of the lowest genre in my exploration in finding the top genres based on the top 10 movies with the highest genres. Therefore, this finding is limited as movies are often under multiple genres as sub-categories.

The exploration for average movie runtimes for high revenue movie titles compared to low revenue movie titles were promising, there was a large range between the average runtimes. This means that there was a large variability between low and high revenue movie titles in regard to average runtime. High revenue movie titles are typically longer, close to two hours in runtime while low revenue movie titles are shorter, closer to about one and a half hours. This may be limited in considering the personal preferences of movie runtimes, some may prefer short runtimes for movies that are straight to the point and closes out the story as quick as possible. Others may prefer longer runtimes to see the full story development and how it unfolds throughout. The personal preferences are difficult to measure as there is no clear and effective way to understand why longer movie runtimes are generally more likely to produce high revenues. For all movies in consideration, the movie runtime average was 102 minutes which is almost right in the middle between 85 (low, 17 difference) and 131 (high, 29 difference), but slighly closer to 85 (low). This is revealing to show the kinds of limitations to this exploration.

During my exploration to finding the best month to release movies in seeking high revenues, I found that December had the highest total revenues with the movies that released that same month. The following months that also had the highest total revenues were June and November as seen in my bar plot. This finding is limited as December is the holiday season and the time where most families, friends, and loved ones gather together to spend time with each other and also when people are in the holiday spirit to spend more. One of the activities they do together to celebrate the upcoming holidays may to be go the movies. Most people watch movies as a group and so the best time of year for a movie release to generte high revenues may be during times people can easily come together. December is also the time where people have off from school or work and have more free time to go out to the movies or come together to watch movies. This can also apply to November for Thanksgiving holiday season. June is the beginning of summer and the end of school, so this can be the time where families go the movies to celebrate end of school or start of summertime. This finding is limited as there may be other factors that are associated with the best month to release movies for high revenues, such as availablity of free time and reasoning to watch movies.

Not only this consideration limits my findings in my exploration but the number of movies made in each month. As we had over 10,000 movies, each month had roughly the same amount of movies that came out that same month. The average amount of movies released each month

> was 903, in December (981 movies), and in June (826 movies). As June was ranked second as the month where high revenue movies were produced, there were 155 less movies made in this month compared to December. If June had the same amount of movies produced as December, June may have produced more revenue in comparsion. Since December had 155 more movies released, there were more opportunities for December movies to generate a higher movie revenue total, simply because more movies were made. Therefore, it is diffcult to properly reason why December was the best month to release movies for high revenues in this dataset. For the best month of movie release exploration, 'best month' can be defined as a highly suggestive month to consider releasing a movie that could possibly generate higher revenues than releasing the movie in other months such as November or June.

Resources used for this data analysis:

https://stackoverflow.com/questions/20804673/appending-column-totals-to-a-pandas-dataframe (https://stackoverflow.com/questions/20804673/appending-column-totals-to-a-pandas-dataframe)

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.sort_values.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.sort_values.html)

https://stackoverflow.com/questions/44493417/pandas-dataframe-bar-plot-plot-bars-different-colors-from-specific-colormap (https://stackoverflow.com/questions/44493417/pandas-dataframe-bar-plot-plot-bars-different-colors-from-specific-colormap)

https://stackoverflow.com/questions/52323435/how-to-get-the-frequency-of-a-specific-value-in-each-row-of-pandas-dataframe (https://stackoverflow.com/questions/52323435/how-to-get-the-frequency-of-a-specific-value-in-each-row-of-pandas-dataframe)

https://stackoverflow.com/questions/34378059/update-in-pandas-on-specific-columns (https://stackoverflow.com/questions/34378059/update-in-pandas-on-specific-columns)

https://stackoverflow.com/questions/35364601/group-by-and-find-top-n-value-counts-pandas (https://stackoverflow.com/questions/35364601/group-by-and-find-top-n-value-counts-pandas)