Final Project Submission

Please fill out:

- Student name: Ryan Keats
- Student pace: self paced / part time / full time
- Scheduled project review date/time: September 11, 11:59pm.
- Instructor name: Hardik Idnani

In [42]:
```
# Firstly, I have imported pandas and numpy to help me work on my d
import pandas as pd
import numpy as np

# I have imported matplotlib to be able to peform my graphs.
import matplotlib.pyplot as plt

# I have added this to clean up the number formatting of the data s
import pandas as pd
pd.set_option('display.float_format', lambda x: '${:,.2f}'.format(x
```

In [43]:
```
# This is the first data set I have used and defined to start my pr
movie_info = pd.read_csv("zippedData/bom.movie_gross.csv.gz")
movie_info.head()
```

Out[43]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	$415,000,000.00	652000000	2010
1	Alice in Wonderland (2010)	BV	$334,200,000.00	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	$296,000,000.00	664300000	2010
3	Inception	WB	$292,600,000.00	535700000	2010
4	Shrek Forever After	P/DW	$238,700,000.00	513900000	2010

In [44]:
```
# I made column names easier to use.
movie_info.columns = movie_info.columns.str.lower().str.replace(' '

# I dropped unnecessary columns.
movie_info.drop(columns = ['studio', 'foreign_gross'], inplace=True
In [45]: # I have used .iloc to display another way I searched the data fram
         movie_info.iloc[0:20,:]
```

Out[45]:

	title	domestic_gross	year
0	Toy Story 3	$415,000,000.00	2010
1	Alice in Wonderland (2010)	$334,200,000.00	2010
2	Harry Potter and the Deathly Hallows Part 1	$296,000,000.00	2010
3	Inception	$292,600,000.00	2010
4	Shrek Forever After	$238,700,000.00	2010
5	The Twilight Saga: Eclipse	$300,500,000.00	2010
6	Iron Man 2	$312,400,000.00	2010
7	Tangled	$200,800,000.00	2010
8	Despicable Me	$251,500,000.00	2010
9	How to Train Your Dragon	$217,600,000.00	2010
10	Clash of the Titans (2010)	$163,200,000.00	2010
11	The Chronicles of Narnia: The Voyage of the Da...	$104,400,000.00	2010
12	The King's Speech	$135,500,000.00	2010
13	Tron Legacy	$172,100,000.00	2010
14	The Karate Kid	$176,600,000.00	2010
15	Prince of Persia: The Sands of Time	$90,800,000.00	2010
16	Black Swan	$107,000,000.00	2010
17	Megamind	$148,400,000.00	2010
18	Robin Hood	$105,300,000.00	2010
19	The Last Airbender	$131,800,000.00	2010

```
In [12]: # I have used .columns to outline the columns of the data set.
         movie_info.columns

Out[12]: Index(['title', 'domestic_gross', 'year'], dtype='object')
```

```
In [13]: # I have chosen .shape to show the size of the first data set I am
         movie_info.shape

Out[13]: (3387, 3)
```

In [46]:
```
# I have sorted the coloumn domestic_gross here, displaying from hi
movie_info = movie_info.sort_values('domestic_gross', ascending = F
movie_info.head(20)
```

Out[46]:

	title	domestic_gross	year
1872	Star Wars: The Force Awakens	$936,700,000.00	2015
3080	Black Panther	$700,100,000.00	2018
3079	Avengers: Infinity War	$678,800,000.00	2018
1873	Jurassic World	$652,300,000.00	2015
727	Marvel's The Avengers	$623,400,000.00	2012
2758	Star Wars: The Last Jedi	$620,200,000.00	2017
3082	Incredibles 2	$608,600,000.00	2018
2323	Rogue One: A Star Wars Story	$532,200,000.00	2016
2759	Beauty and the Beast (2017)	$504,000,000.00	2017
2324	Finding Dory	$486,300,000.00	2016
1875	Avengers: Age of Ultron	$459,000,000.00	2015
729	The Dark Knight Rises	$448,100,000.00	2012
1131	The Hunger Games: Catching Fire	$424,700,000.00	2013
3081	Jurassic World: Fallen Kingdom	$417,700,000.00	2018
0	Toy Story 3	$415,000,000.00	2010
2767	Wonder Woman	$412,600,000.00	2017
1128	Iron Man 3	$409,000,000.00	2013
2322	Captain America: Civil War	$408,100,000.00	2016
735	The Hunger Games	$408,000,000.00	2012
2762	Jumanji: Welcome to the Jungle	$404,500,000.00	2017

In [15]:
```
# I wanted to show the annual domestic_gross average, by using .mea
movie_info.groupby('year').mean()
```

Out[15]:

	domestic_gross
year	
2010	$31,445,592.57
2011	$25,350,524.43
2012	$27,675,842.23
2013	$31,282,115.64
2014	$26,439,228.90
2015	$24,613,375.04
2016	$25,989,960.96
2017	$34,166,456.87
2018	$36,010,421.75

In [16]:
```
# In addition to showing the average annual domestic_gross, I wante
movie_info.groupby('year').sum()
```

Out[16]:

	domestic_gross
year	
2010	$10,156,926,399.00
2011	$10,064,158,200.00
2012	$10,876,605,997.00
2013	$10,792,329,897.00
2014	$10,337,738,499.00
2015	$11,051,405,394.00
2016	$11,253,653,097.00
2017	$10,933,266,198.00
2018	$11,091,209,899.00

In [17]:
```
# I have added this to clean up the number formatting of the data s
import pandas as pd
pd.set_option('display.float_format', lambda x: '{:,.2f}'.format(x)
```

In [18]:
```
# I wanted to get a description of the domestic_gross and the Inner
movie_info['domestic_gross'].describe()
```

Out[18]:
```
count         3,359.00
mean     28,745,845.07
std      66,982,498.24
min             100.00
25%         120,000.00
50%       1,400,000.00
75%      27,900,000.00
max     936,700,000.00
Name: domestic_gross, dtype: float64
```

In [19]:
```
# I have grouped the movies by year, to illustrate their yearly pro
movie_info.groupby('year').count().describe()
```

Out[19]:

	title	domestic_gross
count	9.00	9.00
mean	376.33	373.22
std	51.44	51.23
min	308.00	308.00
25%	328.00	323.00
50%	395.00	391.00
75%	400.00	397.00
max	450.00	449.00

In [20]:
```
# I have added the value_counts() before .mean() to get the average
# Which is also displayed with a different code in the above cell. 
movie_info['year'].value_counts().mean()
```

Out[20]:
```
376.3333333333333
```

In [21]:
```
# I have shown how many movies are made each with .value_counts() a
movie_info['year'].value_counts().sort_values()
```

Out[21]:
```
2018    308
2017    321
2010    328
2013    350
2014    395
2011    399
2012    400
2016    436
2015    450
Name: year, dtype: int64
```

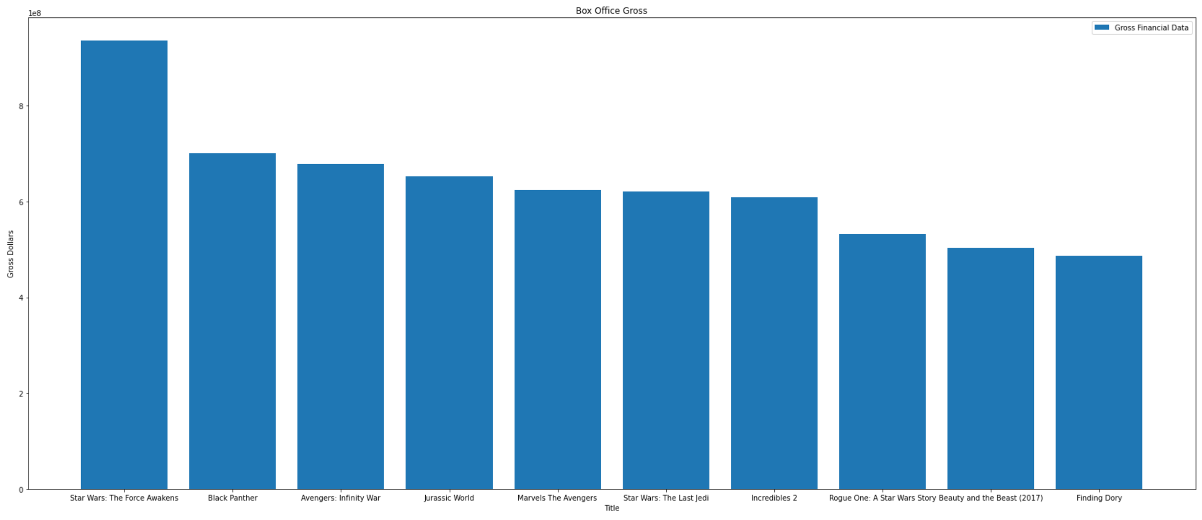
In [14]:
```
# I have chosen to use bar graphs as I feel they are the most simpl
y = [936700000.0, 700100000.0, 678800000.0, 652300000.0, 623400000.
    608600000.0, 532200000.0, 504000000.0, 486300000.0 ]
x = range(10)
labels = ['Star Wars: The Force Awakens', 'Black Panther', 'Avenger
          'Marvels The Avengers', 'Star Wars: The Last Jedi', 'Incr
        'Beauty and the Beast (2017)', 'Finding Dory' ]

# Create the plot
fig, ax = plt.subplots(figsize=(29,12))


ax.bar(x, y, tick_label = labels)

ax.set_title('Box Office Gross')
ax.set_ylabel('Gross Dollars')
ax.set_xlabel('Title');

ax.legend(['Gross Financial Data'], loc=1);
```



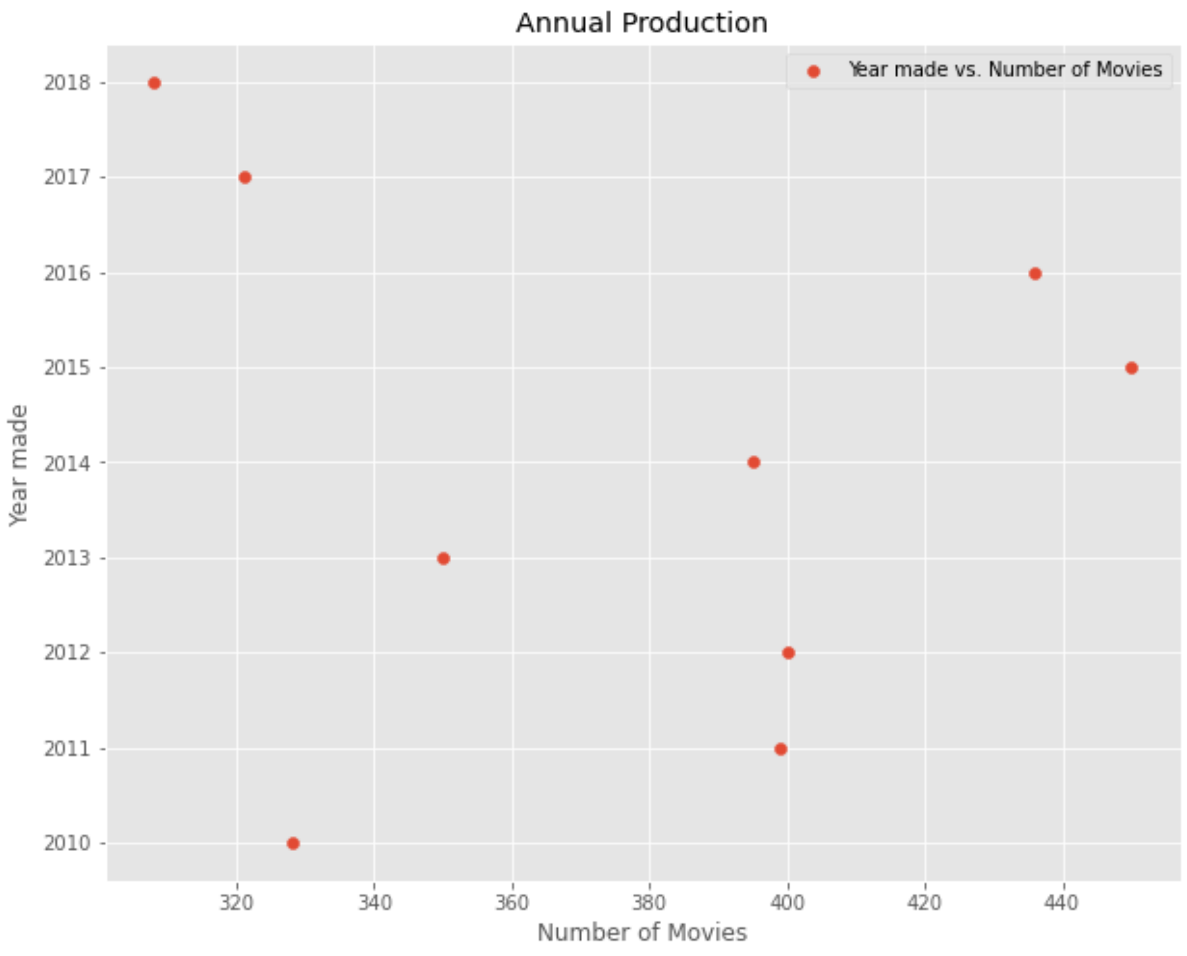
In [17]:
```
# I have used a scatter plot for the same data as above, for a cont
# I have also showed this format to display that there is no correl
domestic_gross = [328, 399, 400, 350, 395, 450, 436, 321, 308]
year = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018]

# Create the plot
fig, ax = plt.subplots(figsize=(10,8))


ax.scatter(domestic_gross, year)

ax.set_xlabel('Number of Movies')
ax.set_ylabel('Year made')
ax.legend(['Year made' ' vs. ' 'Number of Movies'])
ax.set_title('Annual Production');

plt.style.use('ggplot')
```



In [18]:
```
# I have added the .value_counts() of the studio's production infor
movie_info['studio'].value_counts().head(10)
```

Out[18]:
```
IFC      166
Uni.     147
WB       140
Fox      136
Magn.    136
SPC      123
Sony     110
BV       106
LGF      103
Par.     101
Name: studio, dtype: int64
```

In [19]:
```
# I have also added .describe() to show general information of the
movie_info['studio'].describe()
```

Out[19]:
```
count      3382
unique      257
top         IFC
freq        166
Name: studio, dtype: object
```

In [20]:
```
# I wanted to add a command/comments cell gap between each of my da
# Also, for clarity and uniformity to display that I am moving onto
```

In [48]:
```
#I have imported pandas and numpy to help me work on my data sets.
import pandas as pd
import numpy as np

# I have imported matplotlib to be able to peform my graphs.
import matplotlib.pyplot as plt

# I have added this to clean up the number formatting of the data s
import pandas as pd
pd.set_option('display.float_format', lambda x: '{:,.2f}'.format(x)
```

In [56]:
```
#This is the second data set I have used to get my findings from.
movie_info2=pd.read_csv("zippedData/imdb.title.basics.csv.gz")
movie_info2.head().sort_values('runtime_minutes')
```

Out[56]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fantasy
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Drama
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama

In [57]:
```
# I made column names easier to use.
movie_info2.columns = movie_info2.columns.str.lower().str.replace('

# I dropped unnecessary columns.
movie_info2.drop(columns = ['tconst', 'original_title'], inplace=Tr
```

In [58]:
```
# I have used .columns to outline the columns of the data set.
movie_info2.columns
```

Out[58]:
```
Index(['primary_title', 'start_year', 'runtime_minutes', 'genres']
, dtype='object')
```

In [59]:
```
# I have chosen .shape to show the size of the first data set I am
movie_info2.shape
```

Out[59]:
```
(146144, 4)
```

In [60]:
```
#I wanted to use the .value_count() feature to display the top genr
movie_info2['genres'].value_counts().head(10)
```

Out[60]:
```
Documentary                 32185
Drama                       21486
Comedy                       9177
Horror                       4372
Comedy,Drama                 3519
Thriller                     3046
Action                       2219
Biography,Documentary        2115
Drama,Romance                2079
Comedy,Drama,Romance         1558
Name: genres, dtype: int64
```

In [61]:
```
#I have used .describe() on the runtime to outline the (IQR).
movie_info2['runtime_minutes'].describe()
```

Out[61]:
```
count   114,405.00
mean         86.19
std         166.36
min           1.00
25%          70.00
50%          87.00
75%          99.00
max      51,420.00
Name: runtime_minutes, dtype: float64
```

In [62]:
```
# I have displayed the above information in a different format of c
# I prefer this outcome. But, thought it was good to display multip
movie_info2 = movie_info2.sort_values('runtime_minutes', ascending
movie_info2.describe()
```

Out[62]:

	start_year	runtime_minutes
count	146,144.00	114,405.00
mean	2,014.62	86.19
std	2.73	166.36
min	2,010.00	1.00
25%	2,012.00	70.00
50%	2,015.00	87.00
75%	2,017.00	99.00
max	2,115.00	51,420.00

In [63]:
```
# I wanted to add a command/comments cell gap between each of my da
# Also, for clarity and uniformity to display that I am moving onto
```

In [64]:
```
#I have imported pandas and numpy to help me work on my data sets.
import pandas as pd
import numpy as np

# I have imported matplotlib to be able to peform my graphs.
import matplotlib.pyplot as plt

# I have added this to clean up the number formatting of the data s
import pandas as pd
pd.set_option('display.float_format', lambda x: '{:,.2f}'.format(x))
```

In [65]:
```
#This is the third data set I have used to get my findings from.
movie_info3=pd.read_csv("zippedData/imdb.title.ratings.csv.gz")
movie_info3.head()
```

Out[65]:

	tconst	averagerating	numvotes
0	tt10356526	8.30	31
1	tt10384606	8.90	559
2	tt1042974	6.40	20
3	tt1043726	4.20	50352
4	tt1060240	6.50	21

In [66]:
```
# I made column names easier to use.
movie_info3.columns = movie_info3.columns.str.lower().str.replace('

# I dropped unnecessary columns.
movie_info3.drop(columns = ['tconst'], inplace=True)
```

In [67]:
```
# I have used .columns to outline the columns of the data set.
movie_info3.columns
```

Out[67]:
```
Index(['averagerating', 'numvotes'], dtype='object')
```

In [68]:
```
# I have chosen .shape to show the size of the third data set I am
movie_info3.shape
```

Out[68]:
```
(73856, 2)
```

In [69]:
```
# I have sorted the coloumn numvotes here, displaying from high to
movie_info3 = movie_info3.sort_values('numvotes', ascending = False
movie_info3.head(20)
```

Out[69]:

	averagerating	numvotes
63498	8.80	1841066
8738	8.40	1387769
24920	8.60	1299334
38058	8.40	1211405
48221	8.10	1183655
39356	8.20	1035358
3140	8.10	1005960
25777	8.10	948394
60518	8.00	820847
63506	7.20	795227
36852	8.00	784780
3195	8.10	780910
35125	8.10	761592
2237	7.90	719629
52520	7.70	710018
24980	7.20	692794
758	8.10	691835
13579	7.00	683264
51135	8.30	682218
29797	8.00	680116

In [70]:
```
# I have displayed a different way to produce the above information
movie_info3 = movie_info3.sort_values(by=['numvotes','averagerating
movie_info3.tail(20)
```

Out[70]:

	averagerating	numvotes
29797	8.00	680116
51135	8.30	682218
13579	7.00	683264
758	8.10	691835
24980	7.20	692794
52520	7.70	710018
2237	7.90	719629
35125	8.10	761592
3195	8.10	780910
36852	8.00	784780
63506	7.20	795227
60518	8.00	820847
25777	8.10	948394
3140	8.10	1005960
39356	8.20	1035358
48221	8.10	1183655
38058	8.40	1211405
24920	8.60	1299334
8738	8.40	1387769
63498	8.80	1841066

In [71]:
```
# I have displayed the IQR with this code.
movie_info3 = movie_info3.sort_values('averagerating', ascending =
movie_info3.describe()
```

Out[71]:

	averagerating	numvotes
count	73,856.00	73,856.00
mean	6.33	3,523.66
std	1.47	30,294.02
min	1.00	5.00
25%	5.50	14.00
50%	6.50	49.00
75%	7.40	282.00
max	10.00	1,841,066.00