In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from bs4 import BeautifulSoup
import requests

%matplotlib inline
```

1 Imported & Uploading Databases

In [2]:
```
df_titlebasics      = pd.read_csv('data/zippeddata/imdb.title.basics.csv')
df_budget           = pd.read_csv('data/zippeddata/tn.movie_budgets.csv')
```

In [3]:
```
df1= pd.read_csv('/Users/jonax/Documents/opus_df.csv')
```

2 Exploring the databases & cleaning them

In [4]:
```
#looking at the databases
df_titlebasics.head(2)
```

Out[4]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama

In [5]:
```
#understanding information
df_titlebasics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [6]:
```
#renaming column
df_titlebasics.rename(columns={'primary_title': 'movie'})
```

Out[6]:

	tconst	movie	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [7]:
```
#looking at the head
df_titlebasics.head(2)
```

Out[7]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama

In [8]:
```
#renaming columns
df_titlebasics.rename(columns={'primary_title': 'movie'}, inplace=True)
```

In [9]:
```
#looking at the head
df_titlebasics.head()
```

Out[9]:

	tconst	movie	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [10]:
```
#dropping columns
df_titlebasics.drop(['original_title','start_year','genres'], axis=1, inpla
```

In [11]:
```
#dropping additional column
df_titlebasics.drop(['tconst'], axis=1, inplace=True)
```

In [12]:
```
#setting an index
df_titlebasics.set_index('movie')
```

Out[12]:

	runtime_minutes
movie	
Sunghursh	175.0
One Day Before the Rainy Season	114.0
The Other Side of the Wind	122.0
Sabse Bada Sukh	NaN
The Wandering Soap Opera	80.0
...	...
Kuambil Lagi Hatiku	123.0
Rodolpho Teóphilo - O Legado de um Pioneiro	NaN
Dankyavar Danka	NaN
6 Gunn	116.0
Chico Albuquerque - Revelações	NaN

146144 rows × 1 columns

In [13]:
```
#looking at the head
df_budget.head(2)
```

Out[13]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875

In [14]:
```
#dropping columns
df_budget.drop(['id','release_date','domestic_gross'], axis=1, inplace=True
```

In [15]:
```
# removing dollar signs and commas from dollar amounts
# converting dollar amounts from strings into integers
df_budget['production_budget'] = df_budget['production_budget'].str.replace
df_budget['worldwide_gross'] = df_budget['worldwide_gross'].str.replace(','
```

In [16]:
```
df_budget.head(2)
```

Out[16]:

	movie	production_budget	worldwide_gross
0	Avatar	425000000	2776345279
1	Pirates of the Caribbean: On Stranger Tides	410600000	1045663875

In [17]:
```
#setting the index
df_budget.set_index('movie')
```

Out[17]:

movie	production_budget	worldwide_gross
Avatar	425000000	2776345279
Pirates of the Caribbean: On Stranger Tides	410600000	1045663875
Dark Phoenix	350000000	149762350
Avengers: Age of Ultron	330600000	1403013963
Star Wars Ep. VIII: The Last Jedi	317000000	1316721747
...	...	...
Red 11	7000	0
Following	6000	240495
Return to the Land of Wonders	5000	1338
A Plague So Pleasant	1400	0
My Date With Drew	1100	181041

5782 rows × 2 columns

In [18]:
```
#seeing the information
df_budget.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   movie              5782 non-null   object
 1   production_budget  5782 non-null   int64 
 2   worldwide_gross    5782 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 135.6+ KB
```

In [19]:
```
#taking a look at the head
df1.head(2)
```

Out[19]:

	Unnamed: 0	title	production_year	budget	dom_gross	int_gross	creative_type	prod_meth
0	0	Madea's Family Reunion	2006	10000000	63257940	62581	Contemporary Fiction	Live Acti
1	1	Krrish	2006	10000000	1430721	31000000	Science Fiction	Live Acti

In [20]:
```
#dropping columns
df1.drop(['production_year', 'budget', 'dom_gross', 'int_gross', 'creative_
```

In [21]:
```
#dropping columns
df1.head(2)
```

Out[21]:

	Unnamed: 0	title	genre
0	0	Madea's Family Reunion	Comedy
1	1	Krrish	Action

In [22]:
```
#dropping columns
df1.rename(columns={'title':'movie'}, inplace=True)
```

In [23]:
```
#dropping columns
df1.head(2)
```

Out[23]:

	Unnamed: 0	movie	genre
0	0	Madea's Family Reunion	Comedy
1	1	Krrish	Action

In [24]:
```
#dropping columns
df1.drop(['Unnamed: 0'], axis=1, inplace=True)
```

df1.head(2)

In [25]:
```
#looking at it
df1
```

Out[25]:

	movie	genre
0	Madea's Family Reunion	Comedy
1	Krrish	Action
2	End of the Spear	Drama
3	A Prairie Home Companion	Comedy
4	Saw III	Horror
...	...	...
1931	The Nutcracker and the Four Realms	Adventure
1932	Aquaman	Action
1933	Ralph Breaks The Internet	Adventure
1934	Mission: Impossible—Fallout	Action
1935	Fantastic Beasts: The Crimes of Grindelwald	Adventure

1936 rows × 2 columns

In [26]:
```
#setting the index
df1.set_index('movie')
```

Out[26]:

	genre
movie	
Madea's Family Reunion	Comedy
Krrish	Action
End of the Spear	Drama
A Prairie Home Companion	Comedy
Saw III	Horror
...	...
The Nutcracker and the Four Realms	Adventure
Aquaman	Action
Ralph Breaks The Internet	Adventure
Mission: Impossible—Fallout	Action
Fantastic Beasts: The Crimes of Grindelwald	Adventure

1936 rows × 1 columns

3 Merging Cleaned Databases & Creating a New Database

In [27]:
```
#merging and creating a new database
df_super=df_budget.merge(df1, on='movie', how='left')
```

In [28]:
```
df_super.head(3)
```

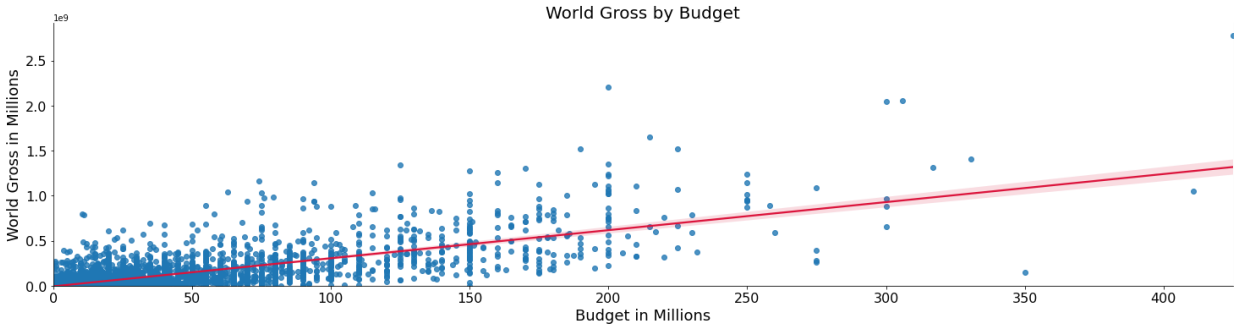
Out[28]:

	movie	production_budget	worldwide_gross	genre
0	Avatar	425000000	2776345279	Action
1	Pirates of the Caribbean: On Stranger Tides	410600000	1045663875	Adventure
2	Dark Phoenix	350000000	149762350	NaN

In [29]:
```
#making the numbers more maneageable of these columns
df_super['production_budget'] = df_super['production_budget'] / 1000000
```

In [30]:
```
#making the numbers maneageable for these columns
df_super['world_gross_mil'] = df_super['worldwide_gross']/1000000
```

In [35]:
```
# plotting world gross by budget in terms of genre with line of best fit
sns.lmplot(x='production_budget', y='worldwide_gross', data=df_super, aspec
plt.title('World Gross by Budget', fontsize=20)
plt.xlabel('Budget in Millions', fontsize=18)
plt.ylabel('World Gross in Millions', fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.ylim(0,None)
plt.xlim(0, None);
```



In [ ]:
```
```

3.1 Analysis

The regression line in this plot shows the general increase in movie gross as the budget increases. While this varies greatly from movie to movie, it was a good to find and and form my recommendations. This plot does confirm that a higher production budget typically leads to a higher box office gross. and that, world gross

3.2 Recommendation 5

Microsoft should know there is a positive relation between the money spent and the growth. To make more money they should focus their effort in budgeting for movies that may cost to make between 150 million and 270 million, approximately. It is in this range where they can expect to make the most money according to the chart above.

Next Recommendation

In [32]:
```
#creating a new database by joining/merging to add a new factor "Run Times"
df_run= df_super.merge(df_titlebasics, on= 'movie', how='left')
```

In [33]:
```
#taking a look at its head
df_run.head()
```

Out[33]:

	movie	production_budget	worldwide_gross	genre	world_gross_mil	runtime_minutes
0	Avatar	425.0	2776345279	Action	2776.345279	93.0
1	Pirates of the Caribbean: On Stranger Tides	410.6	1045663875	Adventure	1045.663875	136.0
2	Dark Phoenix	350.0	149762350	NaN	149.762350	113.0
3	Avengers: Age of Ultron	330.6	1403013963	Action	1403.013963	141.0
4	Star Wars Ep. VIII: The Last Jedi	317.0	1316721747	Adventure	1316.721747	NaN

In [34]:
```
#making a function to group the run time
def coolgroup (row):
    if row["runtime_minutes"] <= 90:
        gp=1
    elif row["runtime_minutes"] >90 and row["runtime_minutes"] <= 120:
        gp=2
    elif row["runtime_minutes"] >120 and row["runtime_minutes"] <= 200:
        gp=3
    else:
        gp=4
    return gp

df_run["runtime_group"] = df_run.apply(coolgroup, axis=1)


df_run.groupby("runtime_group").mean()
```

Out[34]:

runtime_group	production_budget	worldwide_gross	world_gross_mil	runtime_minutes
1	25.010818	6.524879e+07	65.248790	69.991763
2	32.488174	9.279350e+07	92.793502	103.631304
3	60.770889	2.049589e+08	204.958904	135.563410
4	28.535530	7.807857e+07	78.078572	277.200000

3.3 Analysis

Creating four groups for run times in minutes to have an average of these groups per run time and world gross. The discovery is that higher gross movies run about two hours and fifteen minutes, as seen in Group 3. The world gross in this group is higher than in any of the other groups. Group number 1 with an average run time of a little over an hour makes the less money world wide. Group 4 on the other hand with 277minutes of runtime is the second worst time.

3.4 Recommendation 6

Microsoft should clearly not make movies that are only one hour long or more than three hours long. The best runtime for production is two hours and fifteen minutes approximately. Is in this run times where the highest expected world movie gross occurs.

Type *Markdown* and LaTeX: α^2