

# Final Project Submission

Please fill out:

- Student name: Tim Fuger
- Student pace: Part time
- Scheduled project review date/time: TBD
- Instructor name: Morgan Jones
- Blog post URL: TBD

## Project Goal

The goal of this project is to research/analyze movie data from multiple sources in order to make a recommendation to a new movie studio company (Microsoft) on what types of movies they should pursue.

The primary ask from the company is that the type of movies they pursue are similar to the ones "doing best at the box office." From a business perspective, I believe that they are looking to receive the greatest return on the investment that is placed in the movie. To understand what has given the best ROI in the past, we must dig into the data on previous movies and analyze the metrics. We will also analyze the budgets of these movies to determine what the range of spending will be for a movie project.

## Data Source and Data Exploration

The data shown in the following notebook comes from five main sources.

1. Box Office Mojo
2. IMDB
3. Rotten Tomatoes
4. TheMovieDB
5. The Numbers

We start by importing the proper packages required

```
In [1]: import sqlite3
conn = sqlite3.connect("zippedData/im.db/im.db")
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
```

## Create Dataframes

In the next segment, we create dataframes for each data source available

```
In [2]: BOM_df = pd.read_csv('zippedData/bom.movie_gross.csv.gz')
# Combine movie ratings and movie basics from the IMDB dataset to create one dataframe
imdb_movie_df = pd.read_sql("""
SELECT *
FROM movie_basics
JOIN movie_ratings USING(movie_id)
;
""", conn)
rtmovieinfo_df = pd.read_csv('zippedData/rt.movie_info.tsv.gz', sep='\t')
rtreviews_df = pd.read_csv('zippedData/rt.reviews.tsv.gz', sep='\t', encoding='latin-1')
tmdb_df = pd.read_csv('zippedData/tmdb.movies.csv.gz')
tnmoviebudget_df = pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
```

```
In [3]: imdb_movie_df
```

Out[3]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	average
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	
...	...	...	...	...	...	...	...
73851	tt9913084	Diabolik sono io	Diabolik sono io	2019	75.0	Documentary	
73852	tt9914286	Sokagin Çocuklari	Sokagin Çocuklari	2019	98.0	Drama,Family	
73853	tt9914642	Albatross	Albatross	2017	NaN	Documentary	
73854	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	NaN	None	
73855	tt9916160	Drømmeland	Drømmeland	2019	72.0	Documentary	

73856 rows × 8 columns

```
In [4]: rtmovieinfo_df
```

Out[4]:

	id	synopsis	rating	genre	director	writer	theater_date
--	----	----------	--------	-------	----------	--------	--------------

	<b>id</b>	<b>synopsis</b>	<b>rating</b>		<b>genre</b>	<b>director</b>	<b>writer</b>	<b>theater_da</b>
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure Classics Drama	William Friedkin	Ernest Tidyman	Oct 9, 19...	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama Science Fiction and Fantasy	David Cronenberg	David Cronenberg Don DeLillo	Aug 17, 20...	
2	5	Illeana Douglas delivers a superb performance ...	R	Drama Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 19...	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama Mystery and Suspense	Barry Levinson	Paul Attanasio Michael Crichton	Dec 9, 19...	
4	7	NaN	NR	Drama Romance	Rodney Bennett	Giles Cooper	NaN	
...	...	...	...	...	...	...	...	
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure Horror Mystery and Suspense	NaN	NaN	Aug 18, 20...	
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy Science Fiction and Fantasy	Steve Barron	Terry Turner Tom Davis Dan Aykroyd Bonnie Turner	Jul 23, 19...	
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics Comedy Drama Musical and Performing Arts	Gordon Douglas	NaN	Jan 1, 19...	
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy Drama Kids and Family Sports and Fitness	David Mickey Evans	David Mickey Evans Robert Gunter	Apr 1, 19...	
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure Art House and Internation...	NaN	Luc Besson	Sep 27, 20...	

1560 rows × 12 columns

In [5]: `rtrreviews_df`

Out[5]:

	<b>id</b>	<b>review</b>	<b>rating</b>	<b>fresh</b>	<b>critic</b>	<b>top_critic</b>	<b>publisher</b>	<b>date</b>
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	November 10, 2018
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	Annalee Newitz	0	io9.com	May 23, 2018
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	Sean Axmaker	0	Stream on Demand	January 4, 2018
3	3	Continuing along a line introduced in last yea...	NaN	fresh	Daniel Kasman	0	MUBI	November 16, 2017
4	3	... a perverse twist on neorealism...	NaN	fresh	NaN	0	Cinema Scope	October 12, 2017
...	...	...	...	...	...	...	...	...
54427	2000	The real charm of this trifle is the deadpan c...	NaN	fresh	Laura Sinagra	1	Village Voice	September 24, 2002
54428	2000	NaN	1/5	rotten	Michael Szymanski	0	Zap2it.com	September 21, 2005
54429	2000	NaN	2/5	rotten	Emanuel Levy	0	EmanuelLevy.Com	July 17, 2005
54430	2000	NaN	2.5/5	rotten	Christopher Null	0	Filmcritic.com	September 7, 2003
54431	2000	NaN	3/5	fresh	Nicolas Lacroix	0	Showbizz.net	November 12, 2002

54432 rows × 8 columns

In [6]: `tmdb_df`

Out[6]:

	<b>Unnamed: 0</b>	<b>genre_ids</b>	<b>id</b>	<b>original_language</b>	<b>original_title</b>	<b>popularity</b>	<b>release_date</b>	
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	H and t Hall
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	Ho Yc
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	I

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16
...	...	...	...	...	...	...	...
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22
26516	26516	[53, 27]	309885	en	The Church	0.600	2018-10-05

26517 rows × 10 columns

◀	▶
---	---

In [7]: tnmoviebudget\_df

	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
2	3	Jun 7, 2019	Dark Phoenix	\$350,000,000	\$42,762,350	\$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	\$330,600,000	\$459,005,868	\$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	\$317,000,000	\$620,181,382	\$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	\$7,000	\$0	\$0
5778	79	Apr 2, 1999	Following	\$6,000	\$48,482	\$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	\$5,000	\$1,338	\$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	\$1,400	\$0	\$0
5781	82	Aug 5, 2005	My Date With Drew	\$1,100	\$181,041	\$181,041

5782 rows × 6 columns

After getting a glimpse of all of the data, we have the following categorical data to work with:

## 1. Box Office Mojo

- A. title
  - B. studio
  - C. domestic gross
  - D. foreign gross
  - E. year
2. IMDB
- A. movie id
  - B. primary title
  - C. original title
  - D. start year
  - E. runtime
  - F. genre
  - G. average rating
  - H. number votes
3. Rotten Tomatoes - movie info
- A. id
  - B. synopsis
  - C. rating
  - D. genre
  - E. director
  - F. writer
  - G. theatre date
  - H. dvd date
  - I. currency
  - J. box office
  - K. runtime
  - L. studio
4. Rotten Tomatoes - review info
- A. id
  - B. review
  - C. rating
  - D. fresh
  - E. critic
  - F. top critic
  - G. publisher
  - H. date
5. TheMovieDB
- A. Unnamed 0
  - B. genre id
  - C. id
  - D. original language
  - E. original title
  - F. popularity

- G. release date
  - H. title
  - I. vote average
  - J. vote count
6. The Numbers
- A. id
  - B. release date
  - C. movie
  - D. production budget
  - E. domestic gross
  - F. worldwide gross

From the above categorical data, we can better understand the variables we have to analyze the movie datasets. As a result, we can better formulate our questions.

The first question to answer (and the most important) is "What are the movies that have the highest return on investment?" Without having dug into the data yet, we should be able to determine this by first finding the highest profiting movies (difference between budget and gross revenue) from datasets where we have budget and revenue available to us. Then calculate the ROI (return on investment) percentage for each of those movies by taking the profit over the budget (initial investment).

Once we determine which movies provide the highest ROI (above a percentage threshold), then we can analyze those values in the organized set for both the average/range of production budget values, as well as analyze some matching characteristics within that range. Therefore our second question to answer will be "What is the average production budget the company will have to spend on a movie?" The third question will be "What are the matching characteristics between the movies with the highest return on investment?" Based on the categorical data we have to work with, we can look at genre, and runtime as they pertain to ROI.

We also have data in the rotten tomatoes datasets that can inform our recommendation. These datasets do not have movie titles unfortunately, so we will need to infer the success of these movies using the average viewer ratings of the movies. We can then analyze the categorical data from the higher end of those ratings to make a soft recommendation on what the best rating and movie director is for the success of a movie.

We will start with determining the highest profiting movie by taking the difference between the production budget and domestic gross from the "The Numbers" dataset.

First we clean the dataframe by converting the budget, and gross columns from strings to floats. Take out the commas, remove the \$ sign and convert the object to float

In [8]:

```
# remove commas
tnmoviebudget_df = tnmoviebudget_df.replace(',', '', regex=True)
tnmoviebudget_df = tnmoviebudget_df.set_index('movie')

# take out $ sign and convert numbers from string to float
tnmoviebudget_df['production_budget'] = tnmoviebudget_df['production_budget'].str[1:].astype(float)
```

```
tnmoviebudget_df['domestic_gross'] = tnmoviebudget_df['domestic_gross'].str[1:].astype(int)
tnmoviebudget_df['worldwide_gross'] = tnmoviebudget_df['worldwide_gross'].str[1:].astype(int)
```

Next we will calculate the difference between the worldwide gross and budget, which should give us an idea of profit for the movies. Then we will drop any rows with negative values for profit.

```
In [9]: # calculate difference between gross and production budget
tnmoviebudget_df['Profit'] = tnmoviebudget_df['worldwide_gross'] - tnmoviebudget_df['production_budget']

# any negative values converted to zero
tnmoviebudget_df[tnmoviebudget_df['Profit'] < 0] = 0
```

```
In [10]: # drop all zero values
tnmoviebudget_df = tnmoviebudget_df.loc[tnmoviebudget_df['Profit'] != 0]
```

```
In [11]: # sort the values by the highest difference between worldwide gross and production budget
tnmovieBudget_df = tnmoviebudget_df.sort_values(by=['Profit'], ascending=False, inplace=True)
```

```
In [12]: # How many movies we have in the dataframe
# notice how this compares to the count of movies in adjusted dataframes as we continue
len(tnmoviebudget_df.index)
```

Out[12]: 3657

```
In [13]: tnmoviebudget_df.head()
```

	<b>id</b>	<b>release_date</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>Profit</b>
	<b>movie</b>					
<b>Avatar</b>	1	Dec 18 2009	425000000.0	760507625.0	2.776345e+09	2.351345e+09
<b>Titanic</b>	43	Dec 19 1997	200000000.0	659363944.0	2.208208e+09	2.008208e+09
<b>Avengers: Infinity War</b>	7	Apr 27 2018	300000000.0	678815482.0	2.048134e+09	1.748134e+09
<b>Star Wars Ep. VII: The Force Awakens</b>	6	Dec 18 2015	306000000.0	936662225.0	2.053311e+09	1.747311e+09
<b>Jurassic World</b>	34	Jun 12 2015	215000000.0	652270625.0	1.648855e+09	1.433855e+09

Now we will take a look at the highest return on investment for the movies

```
In [14]: # calculate the return on investment using the difference between the domestic gross and worldwide gross
tnmoviebudget_df['ROI'] = (tnmoviebudget_df['Profit'] / tnmoviebudget_df['production_budget']) * 100
tnmoviebudget_df['ROI'] = tnmoviebudget_df['ROI'].astype(int)
tnmovieBudget_df = tnmoviebudget_df.sort_values(by=['ROI'], ascending=False, inplace=True)
```

```
In [15]: tnmoviebudget_df.head(20)
```

	<b>id</b>	<b>release_date</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>Profit</b>	<b>ROI</b>
	<b>movie</b>						
<b>Deep Throat</b>	46	Jun 30 1972	25000.0	45000000.0	45000000.0	44975000.0	1799%

	<b>id</b>	<b>release_date</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>Profit</b>	<b>R</b>
<b>movie</b>							
<b>Mad Max</b>	14	Mar 21 1980	2000000.0	8750000.0	99750000.0	99550000.0	497
<b>Paranormal Activity</b>	93	Sep 25 2009	450000.0	107918810.0	194183034.0	193733034.0	430
<b>The Gallows</b>	80	Jul 10 2015	100000.0	22764410.0	41656474.0	41556474.0	415
<b>The Blair Witch Project</b>	7	Jul 14 1999	600000.0	140539099.0	248300000.0	247700000.0	412
<b>Super Size Me</b>	10	May 7 2004	65000.0	11529368.0	22233808.0	22168808.0	341
<b>Bambi</b>	47	Aug 13 1942	858000.0	102797000.0	268000000.0	267142000.0	311
<b>El Mariachi</b>	74	Feb 26 1993	7000.0	2040920.0	2041928.0	2034928.0	290
<b>Night of the Living Dead</b>	77	Oct 1 1968	114000.0	12087064.0	30087064.0	29973064.0	262
<b>Rocky</b>	11	Nov 21 1976	1000000.0	117235147.0	225000000.0	224000000.0	224
<b>Halloween</b>	37	Oct 17 1978	325000.0	47000000.0	70000000.0	69675000.0	214
<b>The Brothers McMullen</b>	16	Aug 9 1995	50000.0	10426506.0	10426506.0	10376506.0	207
<b>The Texas Chainsaw Massacre</b>	66	Oct 18 1974	140000.0	26572439.0	26572439.0	26432439.0	188
<b>American Graffiti</b>	73	Aug 11 1973	777000.0	115000000.0	140000000.0	139223000.0	179
<b>My Date With Drew</b>	82	Aug 5 2005	1100.0	181041.0	181041.0	179941.0	163
<b>Once</b>	57	May 16 2007	150000.0	9445857.0	23323631.0	23173631.0	154
<b>Clerks</b>	43	Oct 19 1994	27000.0	3073428.0	3894240.0	3867240.0	143
<b>The Stewardesses</b>	13	Jul 25 1969	200000.0	13500000.0	25000000.0	24800000.0	124
<b>Snow White and the Seven Dwarfs</b>	18	Dec 21 1937	1488000.0	184925486.0	184925486.0	183437486.0	123
<b>Billy Jack</b>	58	Jan 1 1971	800000.0	98000000.0	98000000.0	97200000.0	121

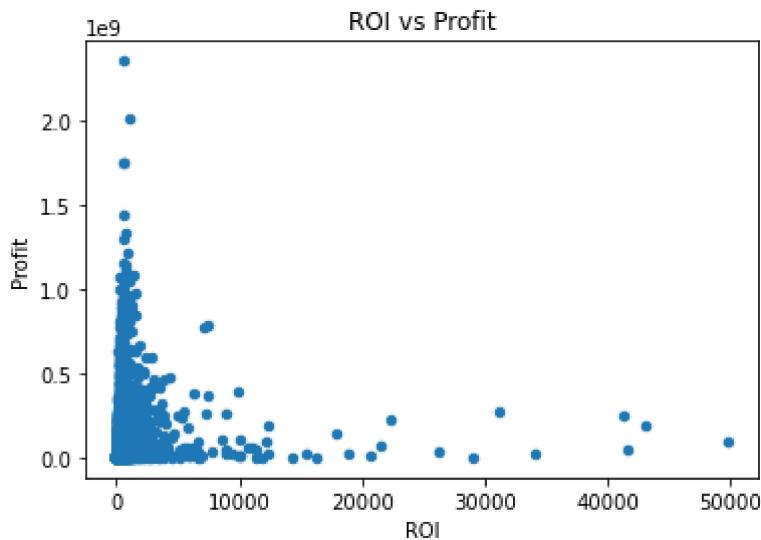
◀ ▶

The first movie in the list is an anomaly as Deep throat was a start of a popularization of a new movie genre. So we can remove that right away

In [16]: `# removing the first row  
tnmoviebudget_df = tnmoviebudget_df.iloc[1:, :]`

Lets take a quick look at ROI vs Profit

```
In [17]: ROIvsWP1 = tnmoviebudget_df.plot.scatter(x='ROI', y='Profit')
ROIvsWP1.set(xlabel="ROI", ylabel="Profit", title="ROI vs Profit");
```



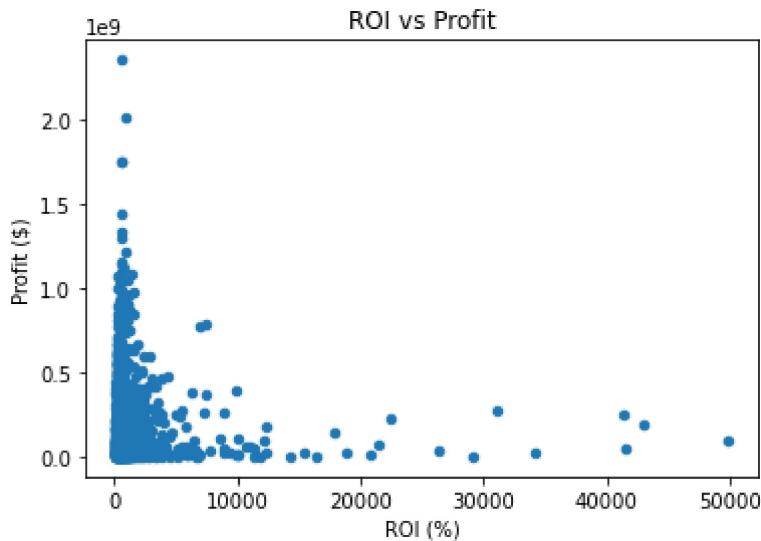
Based on the scatter plot above we can see that most of the movies we now have in the dataset fall closer to the 0,0 mark which means the gross of the movie was the same as the budget. Since we are looking for the highest rate of return we can just take all of movies that fall above 200% ROI and analyze those.

```
In [18]: # Remove all movies that fall below 200% ROI
tnmoviebudgetROI_df = tnmoviebudget_df.loc[tnmoviebudget_df['ROI'] > 200]
```

```
In [19]: # suppress c argument error for color
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
# plot roi vs profit
ROIvsWP2 = tnmoviebudgetROI_df.plot.scatter(x='ROI', y='Profit')
ROIvsWP2.set(xlabel="ROI (%)", ylabel="Profit ($)", title="ROI vs Profit");

# print the amount of movies we have to work with
len(tnmoviebudgetROI_df.index)
```

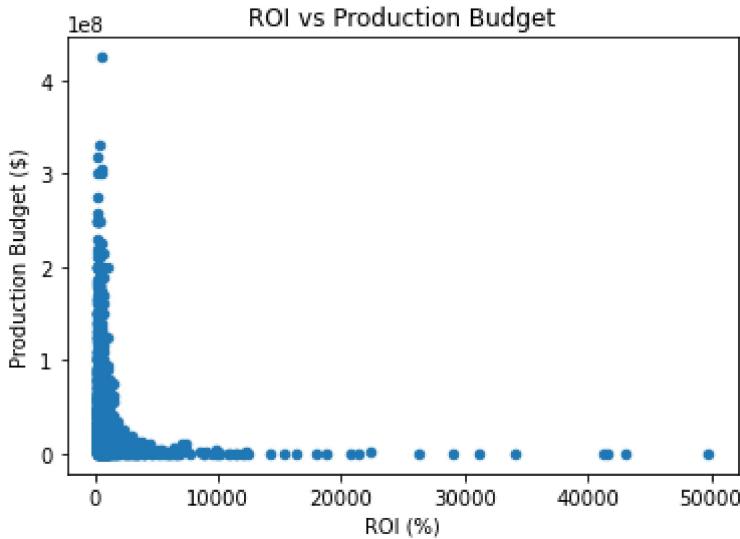
Out[19]: 1846



The plot above looks the same, but the points are so close together that it is not clear that we have removed any movies. But from printing the amount of movies we have in the dataframe, it is clear that there are about half the amount of movies than what we started with.

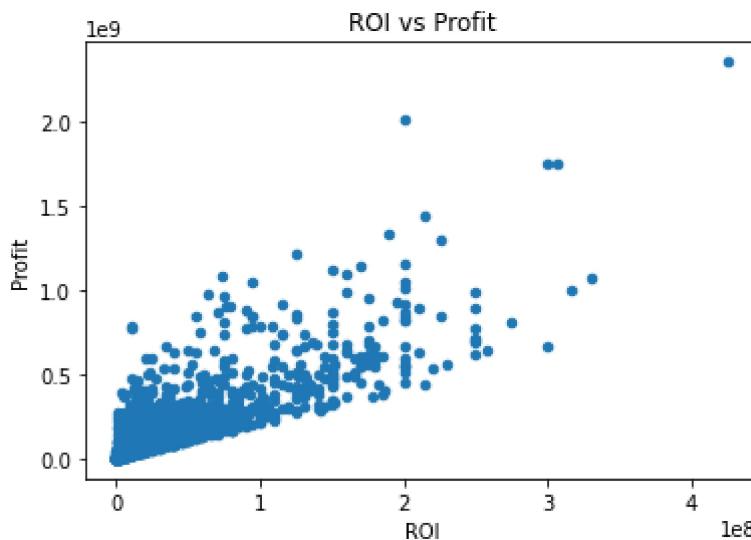
Lets take a look at how ROI compares with profit and budget

```
In [20]: ROIvsPB = tnmoviebudgetROI_df.plot.scatter(x='ROI', y='production_budget')
ROIvsPB.set(xlabel="ROI (%)", ylabel="Production Budget ($)", title="ROI vs Production Budget")
```



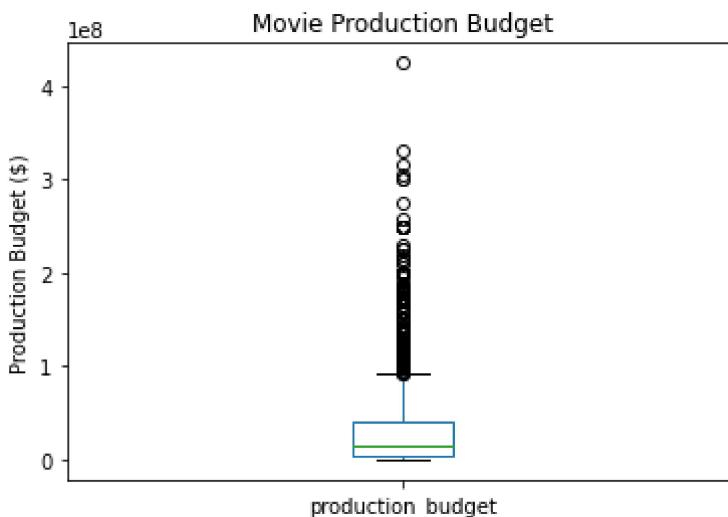
Looks like the greater the production budget, the far less ROI there is in the movie. That doesn't necessarily mean the studio didn't make money, as ROI is calculated as a percentage. But we should probably take the midrange of movies as I don't see a new Microsoft pursuing something at the high end of the spectrum, but they also probably have enough money that they do not want to spend at the low end of the spectrum either.

```
In [21]: ROIvsProfit = tnmoviebudgetROI_df.plot.scatter(x='production_budget', y='Profit');
ROIvsProfit.set(xlabel="ROI", ylabel="Profit", title="ROI vs Profit");
```



Lets also get a look at where most of these fall within these variables as a standard budget that it spent, and the average profit that is earned from that spending category

```
In [22]: PBROI = tnmoviebudgetROI_df['production_budget'].plot.box();
PBROI.set(xlabel=None, ylabel="Production Budget ($)", title="Movie Production Budget")
```



```
In [23]: tnmoviebudgetROI_df['production_budget'].describe()
```

```
Out[23]: count    1.846000e+03
mean      3.449113e+07
std       4.879925e+07
min       1.100000e+03
25%       5.000000e+06
50%       1.500000e+07
75%       4.000000e+07
max       4.250000e+08
Name: production_budget, dtype: float64
```

So the mean (average) production budget for movies that receive over 200% ROI is around 35,000,000 dollars. The cost of the majority of these movies ranges from the 25% quartile at 5,000,000 dollars to the 75% quartile at 40,000,000 dollars. We do not know how much Microsoft would like to spend, but a good recommendation would be to suggest starting in the average

quartile range from 5 million to 40 million dollars. Given that, we should then isolate these movies to analyze going forward.

```
In [24]: tnmoviebudgetROI2_df = tnmoviebudgetROI_df[tnmoviebudgetROI_df['production_budget'] > 5]
tnmoviebudgetROI2_df = tnmoviebudgetROI2_df[tnmoviebudgetROI2_df['production_budget'] <
tnmoviebudgetROI2_df
```

	<b>id</b>	<b>release_date</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>Profit</b>	<b>ROI</b>
	<b>movie</b>						
<b>ET: The Extra-Terrestrial</b>	26	Jun 11 1982	10500000.0	435110554.0	792965326.0	782465326.0	7452
<b>Star Wars Ep. IV: A New Hope</b>	65	May 25 1977	11000000.0	460998007.0	786598007.0	775598007.0	7050
<b>Grease</b>	13	Jun 16 1978	6000000.0	181813770.0	387510179.0	381510179.0	6358
<b>Les Intouchables</b>	18	May 25 2012	10800000.0	13182281.0	484873045.0	474073045.0	4389
<b>Annabelle</b>	84	Oct 3 2014	6500000.0	84273813.0	256862920.0	250362920.0	3851
...	...	...	...	...	...	...	...
<b>Why Did I Get Married Too?</b>	16	Apr 2 2010	20000000.0	60095852.0	60831067.0	40831067.0	204
<b>Kabhi Alvida Naa Kehna</b>	20	Aug 11 2006	10750000.0	3275443.0	32575443.0	21825443.0	203
<b>Daddy Day Camp</b>	39	Aug 8 2007	6000000.0	13235267.0	18209872.0	12209872.0	203
<b>Unplanned</b>	33	Mar 29 2019	6000000.0	18107621.0	18107621.0	12107621.0	201
<b>Jane Austen's Mafia</b>	6	Jul 24 1998	10000000.0	19843795.0	30143795.0	20143795.0	201

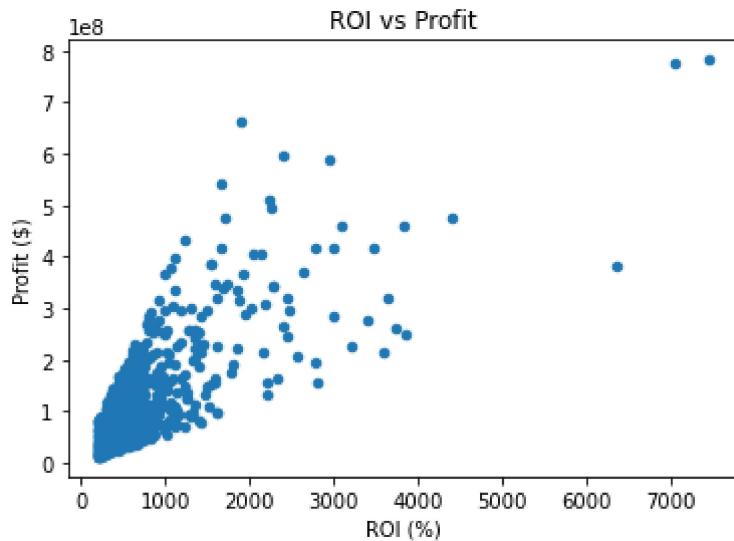
834 rows × 7 columns



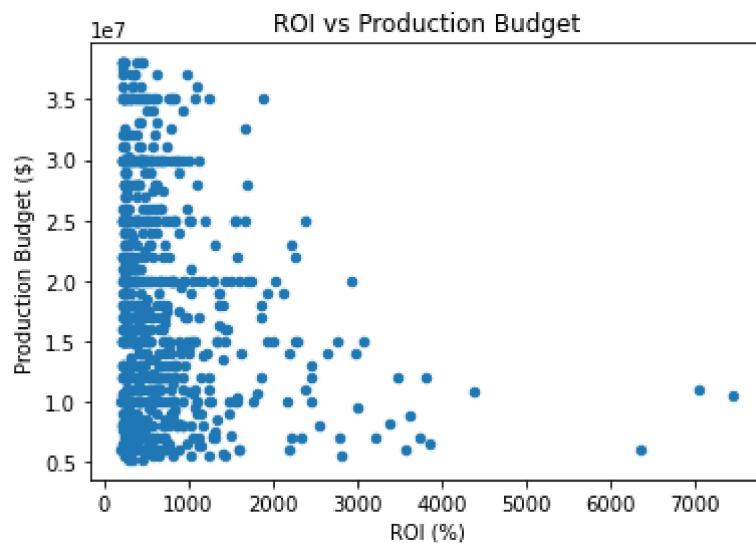
```
In [25]: # plot roi vs profit
ROIvsWP3 = tnmoviebudgetROI2_df.plot.scatter(x='ROI', y='Profit')
ROIvsWP3.set(xlabel="ROI (%)", ylabel="Profit ($)", title="ROI vs Profit");

# print the amount of movies we have to work with
len(tnmoviebudgetROI2_df.index)
```

Out[25]: 834

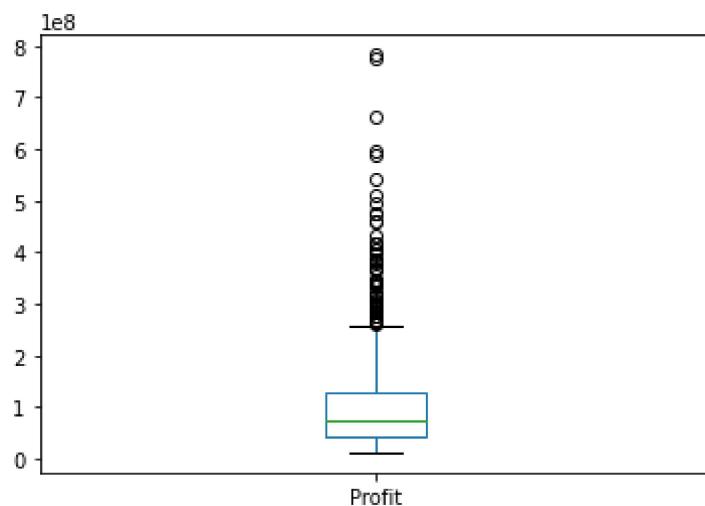


```
In [26]: ROIvsPB2 = tnmoviebudgetROI2_df.plot.scatter(x='ROI', y='production_budget')
ROIvsPB2.set(xlabel="ROI (%)", ylabel="Production Budget ($)", title="ROI vs Production Budget")
```



Lets take a look at how much Microsoft can make with this range of movies

```
In [27]: tnmoviebudgetROI2_df['Profit'].plot.box();
```



In [28]: tnmoviebudgetROI2\_df['Profit'].describe()

Out[28]:

	count	mean	std	min	25%	50%	75%	max
Name: Profit, dtype: float64	8.340000e+02	1.050169e+08	9.828770e+07	1.210762e+07	4.414060e+07	7.339307e+07	1.303245e+08	7.824653e+08

Looks like with these recommendations they can make anywhere from 12,000,000 to 130,000,000 dollars in profit.

Now that we have determined the movies to use, we will analyze their characteristics to make recommendations. We will need to combine the datasets in order to relate the data to one another. For the genre and the runtime we can use the IMDB dataset, for the director we can use the Rotten tomatoes movie info dataset, and for the rating we can use the Rotten tomatoes review info dataset.

First we will take the tnmovie dataset that we have been working with and find those equivalent movies in the IMDB dataset to use as reference.

In [29]:

```
# Set the index to original title in order to use as our key to join IMDB and the number of
imdb_movie_df = imdb_movie_df.set_index('original_title')
imdb_movie_df
```

Out[29]:

	movie_id	primary_title	start_year	runtime_minutes	genres	averagerating
original_title						
<b>Sunghursh</b>	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0
<b>Ashad Ka Ek Din</b>	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography,Drama	7.2
<b>The Other Side of the Wind</b>	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9
<b>Sabse Bada Sukh</b>	tt0069204	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1
<b>La Telenovela Errante</b>	tt0100275	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5
...	...	...	...	...	...	...
<b>Diabolik sono io</b>	tt9913084	Diabolik sono io	2019	75.0	Documentary	6.2
<b>Sokagin Çocukları</b>	tt9914286	Sokagin Çocukları	2019	98.0	Drama,Family	8.7
<b>Albatross</b>	tt9914642	Albatross	2017	NaN	Documentary	8.5
<b>La vida sense la Sara Amat</b>	tt9914942	La vida sense la Sara Amat	2019	NaN	None	6.6
<b>Drømmeland</b>	tt9916160	Drømmeland	2019	72.0	Documentary	6.5

73856 rows × 7 columns

In [30]:

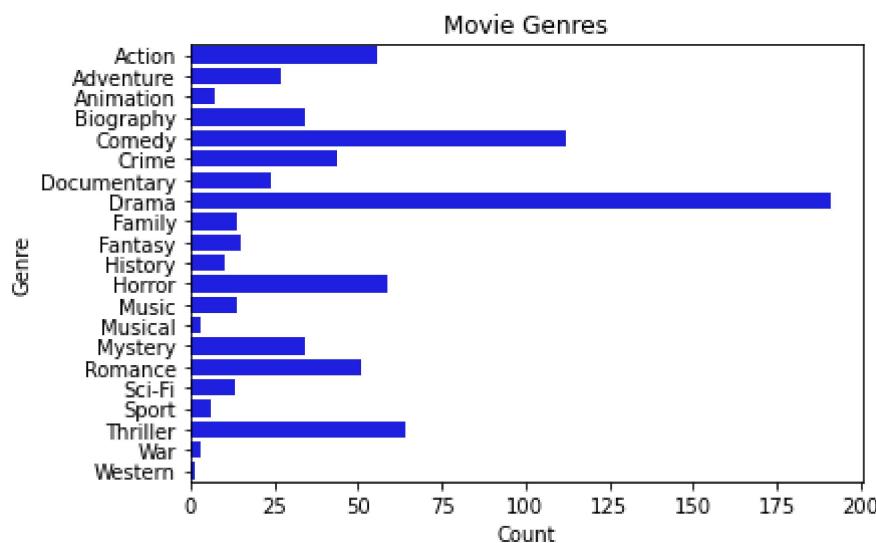
```
# combine the tn movie and imdb dataset
moviescombined_df = tnmoviebudgetROI2_df.join(imdb_movie_df, how='inner')
```

Now that we have combined the movie data, we can analyze any categories that fall in this list to the ROI. Lets start with genre

In [31]:

```
# Split genres into a combined list of all of the genres
unique_genres = [val.strip() for sublist in moviescombined_df.genres.dropna().str.split(',')
# Gets a unique list of all of the genres
genre, counts = np.unique(unique_genres, return_counts=True)

barplot = sns.barplot(y=genre, x=counts, color='b')
barplot.set(xlabel="Count", ylabel="Genre", title="Movie Genres");
```

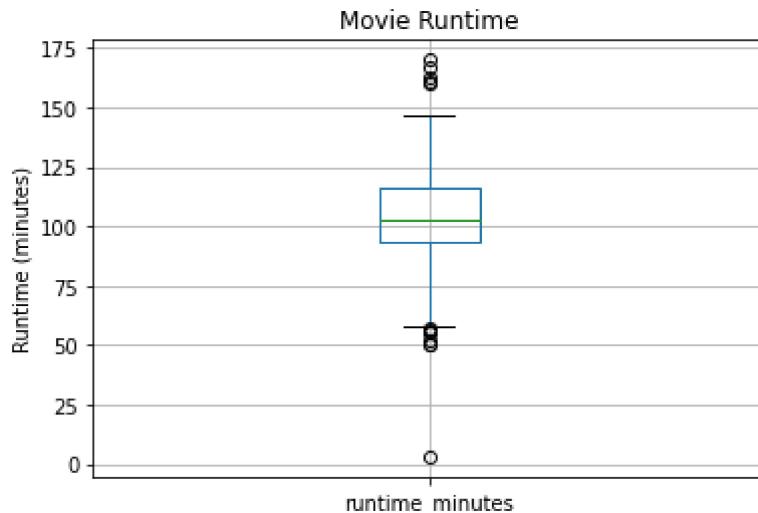


It looks like the most successful genre categories are Drama, Comedy, and Thriller. With Drama being almost double in frequency as compared to the next sucessful genre.

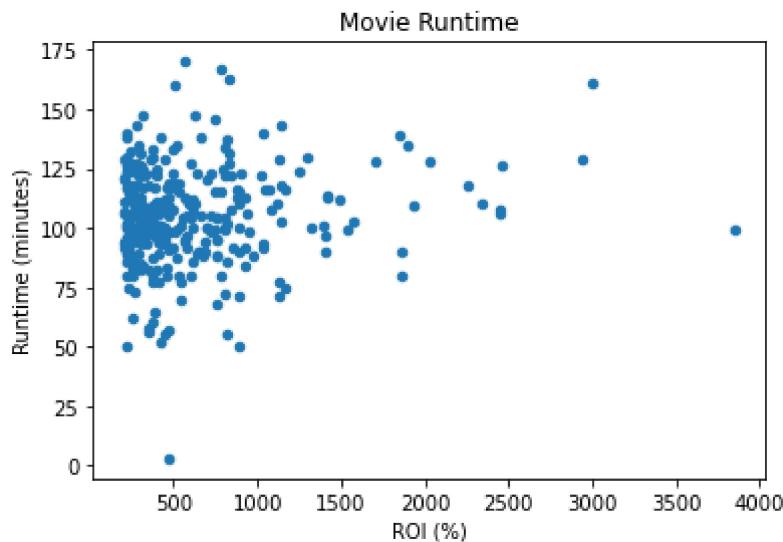
Lets look at runtime next

In [32]:

```
boxplot = moviescombined_df.boxplot(column=['runtime_minutes'])
boxplot.set(xlabel =None, ylabel="Runtime (minutes)", title="Movie Runtime");
```



```
In [33]: scatter = moviescombined_df.plot.scatter(x='ROI', y='runtime_minutes')
scatter.set(xlabel = 'ROI (%)', ylabel="Runtime (minutes)", title="Movie Runtime");
```



```
In [34]: moviescombined_df['runtime_minutes'].describe()
```

```
Out[34]: count    331.000000
mean     104.323263
std      20.262849
min      3.000000
25%     93.000000
50%    103.000000
75%    116.500000
max    170.000000
Name: runtime_minutes, dtype: float64
```

Looks like the average runtime is at around 104 minutes, and the times can vary from the 93 minutes at the 25% quartile to 116.5 minutes at the 75% quartile

```
In [35]: rtreviews_df.dtypes
```

```
Out[35]: id          int64
review       object
rating       object
fresh        object
critic       object
```

```
top_critic      int64  
publisher      object  
date          object  
dtype: object
```

```
In [36]: # Converts string overall rating (example. 3/5) to just a number
rtreviews df['rating cleaned'] = rtreviews df['rating'].str.split('/').str[0]
```

```
In [37]: # removes any rows that contain letter ratings
rtreviews_df = rtreviews_df[rtreviews_df["rating_cleaned"].str.contains("NaN|N|T|R|A")]
```

```
In [38]: # show all values in column to confirm we did not miss any values
rtreviews_df.rating_cleaned.unique()
```

```
Out[38]: array(['3', '2', '4', '6', '1', '8', '2.5', '2.0', '7', '5', '3.5', '1.5',
   '9', '7.5', '5.5', '0', '0.5', '3.1', '4.5', '4.0', '9.5', '2.1',
   '6.5', '3.7', '8.4', '7.2', '2.2', '7.7', '5.0', '8.5', '3.0',
   '2.3', '8.6', '2.7', '4.2', '5.8', '7.1', '5.9', '2.4', '1.9',
   '4.9', '7.4', '8.8', '3.8', '6.8', '7.3', '7.0', '3.2', '6.3',
   '7.6', '8.1', '3.6', '1.8', '8.9', '8.2', '8.3', '2.6', '4.1',
   '6.0', '1.0', '7.9', '8.7', '4.3', '9.6', '9.0', '1.7', '6.7',
   '8.0', '9.2', '5.2', '4.7', '6.2', '3.4', '9.7', '3.3', '4.8',
   '1.6', '7.8', '4.4', '6.9', '9.8'], dtype=object)
```

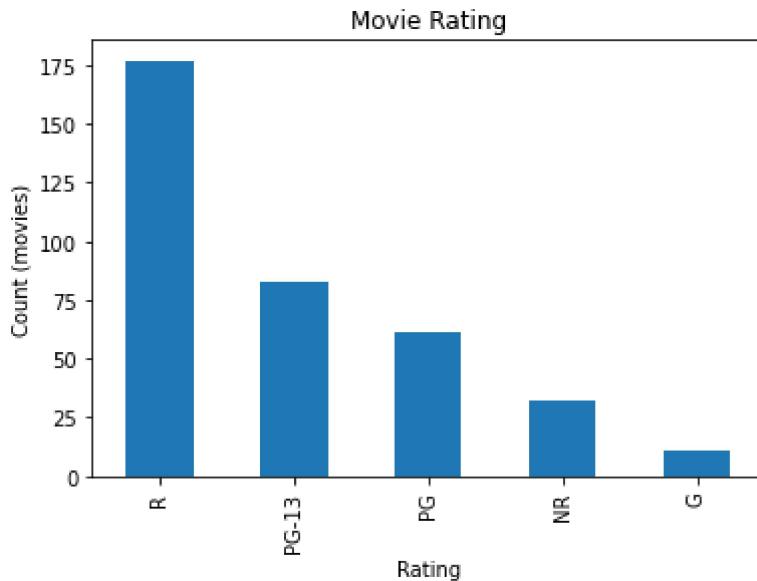
```
In [39]: # Pandas was throwing a "setting with copy warning" for the second line in this code. N  
# copies the column then convert it from string to float values  
rtreviews_df = rtreviews_df[rtreviews_df['rating_cleaned'].notnull()].copy()  
rtreviews_df['rating_cleaned'] = rtreviews_df['rating_cleaned'].astype(float)
```

```
In [40]: # keeps all ratings above 7
rtreviews_df = rtreviews_df.loc[rtreviews_df['rating_cleaned'] > 7.0]
```

```
In [41]: # sets index to the movie id  
rtmovieinfo_df = rtmovieinfo_df.set_index('id')
```

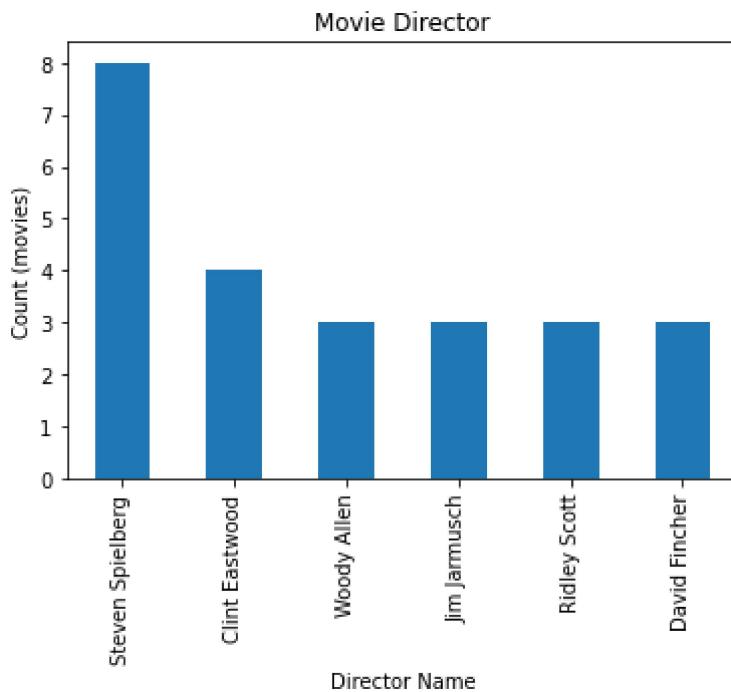
```
In [42]: # gets all unique ids and gets those ids in the rtmovieinfo dataframe  
rtmovieid = rtreviews_df.id.unique()  
rtmovieinfo_df = rtmovieinfo_df.loc[rtmovieid]
```

```
In [43]: # plots movie ratings  
barplot2 = rtmovieinfo_df['rating'].value_counts().plot( kind='bar');  
barplot2.set(xlabel = 'Rating', ylabel="Count (movies)", title="Movie Rating");
```



Looks like the most popular movie ratings are R, PG-13, and PG. The R rating beats the next movie rating by about double in frequency.

```
In [44]: rtdirectors = rtmovieinfo_df['director'].value_counts()
rtdirectorsbool = rtdirectors > 2
rtdirectorsprimary = rtdirectorsbool[rtdirectorsbool].index
rtdirectorsprimary_df = rtdirectors[rtdirectorsprimary]
barplot3 = rtdirectorsprimary_df.plot( kind='bar');
barplot3.set(xlabel = 'Director Name', ylabel="Count (movies)", title="Movie Director")
```



In experimenting with the number above it looks like there are over 30 directors who have done 2 or more movies, but there are 6 directors who have only done 3 or more movies. One of the directors, Woody Allen, is actually no longer active in directing. So we will use the other 5 as our list of recommendations for the directors.

Our results have revealed a few findings on Microsoft pursuing a movie studio. Here are the

takeaways in analyzing movies that have at least 200% ROI:

- The average cost of production for a movie with the range being the 25% to 75% quartile range, can be 5 million to 40 million dollars.
- The profit for a movie with the range being the 25% to 75% quartile range, can be 12 million to 130 million dollars.

Analyzing movies that have at least 200% ROI and a range of production cost being from 5 million to 40 million:

- The genre ordered by highest frequency in the dataset are Drama, Comedy, and Thriller
- The runtime for a movie with the range being the 25% to 75% quartile range, can be from 93 minutes to 116.5 minutes long

Analyzing viewer reviews that are over 7 out of 10:

- The directors listed by highest frequency in the dataset are Steven Spielberg, Clint Eastwood, Ridley Scott, Jim Jarmusch, and David Fincher (these exclude Woody Allen who is no longer active as a director).
- The rating listed by highest frequency in the dataset are R, PG-13, and PG.

In [ ]: