Shentai Tao 01/31/23

Overview

Film industry is big. And some films have big box offices, such as Avatar. Some films may not seems big but with small investment, yielded significant profits comparing to investment, like The Gallows. There are many factors can affect investment outcomes, such as politics, time of year, economy. More in films themselfs, there are *factors* like **Budget, Director/Actor, Genre**.

from upsplash.com

Business Understanding

Microsoft wants to invest in film industry. Here I use **Return on Investment**(or ROI) as measurment to determine if the film is worth to invest. **Budget, Director/Actor, Genre** were investigated here to see how different value of them can affect ROI. At end of the project, recommendations are provided on how to choose between these 3 selections in order to yield high **ROI**. However, these recommendations are just recommendations, final decisions have to be made based on real situation.

Data Understanding

There are 5 *csv files*(including tsv files) and 1 *sqlite database*.

movie_budgets.csv contains information about **budgets** and **worldwide_gross**, which were used to calculate **ROI**.

In sqlite3 database, there are four tables *movie_basics*,*movie_akas*,*persons*,*principals*. These tables contains information about directors, actors, actresses, including their name. Also movie titles have primary titles, original title and aka titles, which are used to connected to *movie_budgets.csv*, so we can get the ROI for each **category(directors, actors, actresses)**. At mean time, we also can get the ROI for each **genres**.

Data Preparation

Data can be aquired here: [https://github.com/learn-co-curriculum/dsc-phase-1-project-v2-4/tree/master/zippedData (https://github.com/learn-co-curriculum/dsc-phase-1-project-v2-4/tree/master/zippedData)](https://github.com/learn-co-curriculum/dsc-phase-1-project-v2-4/tree/master/zippedData)

Copy files to `data` folder in project

Table previews

Table **previews** are showned below to give a brief peep into datas that will be worked on.

In [145]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3

%matplotlib inline
```

In [146]:
```
df_tmdb =pd.read_csv('./Data/tmdb.movies.csv')
df_tmdb
```

Out[146]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_da
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-
...	...	...	...	...	...	...	
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-
26516	26516	[53, 27]	309885	en	The Church	0.600	2018-10-

26517 rows × 10 columns

In [147]:
```
df_rt_reviews= pd.read_csv('./data/rt.reviews.tsv',sep='\t', encoding
df_rt_reviews
```

Out[147]:

	id	review	rating	fresh	critic	top_critic	publisher	date
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

In [148]:
```
df_rt_movie_info = pd.read_csv('./data/rt.movie_info.tsv', sep='\t', e
df_rt_movie_info
```

1556	1997	Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner	
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	NaN	
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	
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure|Art House and Internation...	NaN	Luc Besson	S

1560 rows × 12 columns

In [149]:
```python
df_tn_movie_budgets = pd.read_csv('./data/tn.movie_budgets.csv')
df_tn_movie_budgets
```

Out[149]:

	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

This table is only table that contain infomation on **budget**, which is used to calculate **ROI**.

In [150]:
```python
df_bom_movie_gross = pd.read_csv('./data/bom.movie_gross.csv')
df_bom_movie_gross
```

Out[150]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

This table contains less records of **gross** and doesn't contain **budget** info.

In [151]:
```
conn = sqlite3.connect('./data/im.db')
cur = conn.cursor()
cur.execute("""SELECT name FROM sqlite_master WHERE type = 'table';"""
table_names = cur.fetchall()
table_names
```

Out[151]:
```
[('movie_basics',),
 ('directors',),
 ('known_for',),
 ('movie_akas',),
 ('movie_ratings',),
 ('persons',),
 ('principals',),
 ('writers',)]
```

In [152]:
```
def sql_q1(table_name):
    q = f"""
    SELECT *
    FROM {table_name};"""
    return q

df_movie_basics = pd.read_sql(sql_q1('movie_basics'),conn)
```

In [153]:
```

df_movie_basics
```

0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Dram
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantas
...	...	...	...	...	...	.
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Dram
		Rodolpho	Rodolpho			

This table contains **primary_title, original_title** and **genres**.

In [154]:
```
df_directors = pd.read_sql(sql_q1('directors'),conn)
```

In [155]:
```
df_known_for = pd.read_sql(sql_q1('known_for'),conn)
```

In [156]:
```
df_movie_akas = pd.read_sql(sql_q1('movie_akas'),conn)
df_movie_akas
```

Out[156]:

	movie_id	ordering	title	region	language	types	attributes	is_original_t
0	tt0369610	10	Джурасик свят	BG	bg	None	None	
1	tt0369610	11	Jurashikku warudo	JP	None	imdbDisplay	None	
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	None	imdbDisplay	None	
3	tt0369610	13	O Mundo dos Dinossauros	BR	None	None	short title	
4	tt0369610	14	Jurassic World	FR	None	imdbDisplay	None	
...	...	...	...	...	...	...	...	
331698	tt9827784	2	Sayonara	None	None	original	None	

This table contains more **title**.

In [157]:
```
df_movie_ratings = pd.read_sql(sql_q1('movie_ratings'),conn)
```

In [158]:
```
df_persons = pd.read_sql(sql_q1('persons'),conn)
df_persons
```

Out[158]:

	person_id	primary_name	birth_year	death_year	primar
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_mana
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sounc
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneou
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,ar
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,s
...	...	...	...	...	
606643	nm9990381	Susan Grobes	NaN	NaN	
606644	nm9990690	Joo Yeon So	NaN	NaN	
606645	nm9991320	Madeline Smith	NaN	NaN	
606646	nm9991786	Michelle Modigliani	NaN	NaN	
606647	nm9993380	Pegasus Envoyé	NaN	NaN	directo

606648 rows × 5 columns

This table contains **person** name.

In [159]:
```
df_principals = pd.read_sql(sql_q1('principals'),conn)
df_principals
```

Out[159]:

	movie_id	ordering	person_id	category	job	characters
0	tt0111414	1	nm0246005	actor	None	["The Man"]
1	tt0111414	2	nm0398271	director	None	None
2	tt0111414	3	nm3739909	producer	producer	None
3	tt0323808	10	nm0059247	editor	None	None
4	tt0323808	1	nm3579312	actress	None	["Beth Boothby"]
...	...	...	...	...	...	...
1028181	tt9692684	1	nm0186469	actor	None	["Ebenezer Scrooge"]
1028182	tt9692684	2	nm4929530	self	None	["Herself","Regan"]
1028183	tt9692684	3	nm10441594	director	None	None
1028184	tt9692684	4	nm6009913	writer	writer	None
1028185	tt9692684	5	nm10441595	producer	producer	None

1028186 rows × 6 columns

This table contains **category**. It also contain movie_id and person_id, which can be used as **keys** to connect other tables.

In [160]:
```
df_writers = pd.read_sql(sql_q1('writers'),conn)
```

In [161]:
```
df_principals.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028186 entries, 0 to 1028185
Data columns (total 6 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   movie_id    1028186 non-null  object
 1   ordering    1028186 non-null  int64 
 2   person_id   1028186 non-null  object
 3   category    1028186 non-null  object
 4   job         177684 non-null   object
 5   characters  393360 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.1+ MB
```

In [162]:
```
df_principals.movie_id.value_counts()
```

Out[162]:
```
tt3407878     10
tt7061094     10
tt9837530     10
tt2770324     10
tt7916276     10
              ..
tt6543072      1
tt6645470      1
tt10457064     1
tt10278270     1
tt6639498      1
Name: movie_id, Length: 143454, dtype: int64
```

EDA

Budgets

In [163]:
```
df_tn_movie_budgets
```

Out[163]:

	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

In [164]:
```
!ls
```

```
Microsoft_Movie_Analysis.key     code
Microsoft_Movie_Analysis.pdf     data
Notebook - Jupyter Notebook.pdf  images
Notebook.ipynb                   presentation.pdf
Notebook_copy.ipynb              readme.md
ROI_by_budget.png
```

In [165]:
```
# trying to import module but failed.
import sys
sys.path.insert(0,'./code')

import data_preparation as dp
```

In [166]:
```python



# formate currency from str '$4,000' to int '4000'
df_tn_movie_budgets['production_budget'] = df_tn_movie_budgets.product
```

In [167]:
```python
# formate currency from str '$4,000' to int '4000'
df_tn_movie_budgets['domestic_gross'] = df_tn_movie_budgets.domestic_g
```

In [168]:
```python
# formate currency from str '$4,000' to int '4000'
df_tn_movie_budgets['worldwide_gross'] = df_tn_movie_budgets.worldwide
```

In [169]:
```python
# insanity check
df_tn_movie_budgets['production_budget'][1]
```

Out[169]:
```
410600000
```

In [170]:
```python
df_tn_movie_budgets
```

Out[170]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0	0
5778	79	Apr 2, 1999	Following	6000	48482	240495
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041

5782 rows × 6 columns

In [171]:
```
# add column ROI
df_tn_movie_budgets['ROI'] = (df_tn_movie_budgets['worldwide_gross'] -
```

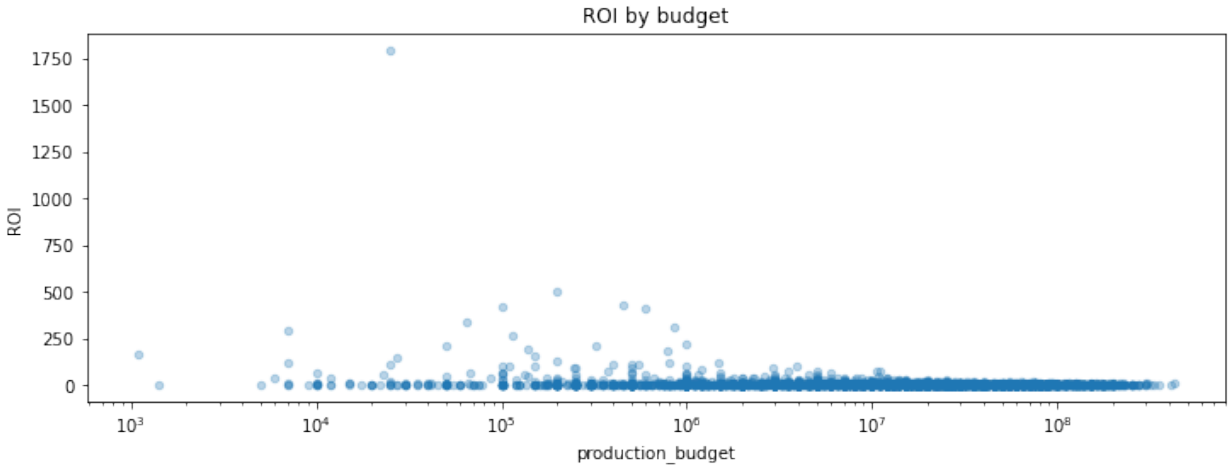
In [172]:
```
# insanity check
df_tn_movie_budgets
```

Out[172]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	4.5
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	0.5
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	-1.5
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	2.2
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	2.1
...	...	...	...	...	...	...	
5777	78	Dec 31, 2018	Red 11	7000	0	0	-2.0
5778	79	Apr 2, 1999	Following	6000	48482	240495	38.0
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338	-1.7
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	-2.0
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041	162.5

5782 rows × 7 columns

In [173]:
```
 use scatter plot to visulize the relationship between ROI and budget
f_tn_movie_budgets.plot(kind='scatter', x='production_budget',y='ROI',
```



In [174]:
```
# sort by ROI
df_tn_movie_budgets = df_tn_movie_budgets.sort_values(by='ROI',ascendi
```

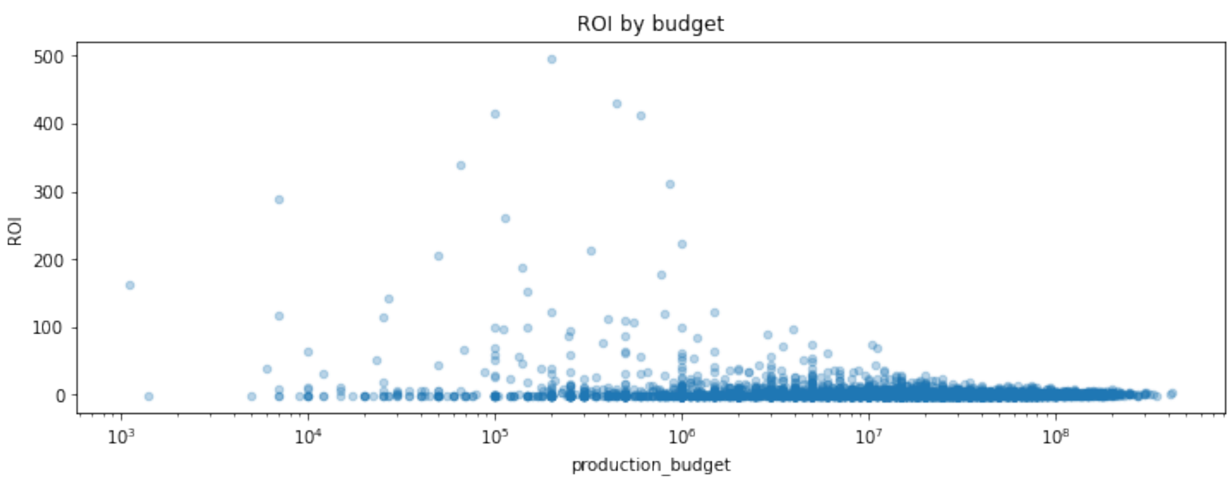
There is a outlier, it's 'Deep Throat', which has 1750 as return on investment.

To continue analysis, let's delete it.

In [175]:
```
df_tn_movie_budgets.reset_index(inplace=True)
```

In [176]:
```
# drop outlier
df_tn_movie_budgets.drop(0,inplace=True)
```

In [177]:
```
df_tn_movie_budgets.plot(kind='scatter', x='production_budget',y='ROI'
plt.savefig('ROI_by_budget')
```



1. Most films have ROI below 50.
2. When **budget** is *between* 10K to 1M, films have *higher* chance to achieve higher ROI, especially when budget are *between* 100K to 1M.
3. There are many films that has **budget** *between* 1M to 10M, but they can't achieve ROI more than 100. Then budgets *higher* than 10M lead to *much lower* ROI overall.
4. There are few films that has **budget** *between* 1K to 10K, but ROI in this **budget** range can be very high or low, means having big variance.

My **recommendation** is to have **budget** set *between* 100K to 200K to have better chance to have high return on investment.

Director/Actor

In [178]:
```
# recall what related tables look like
df_movie_basics
```

0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Dram
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantas
...	...	...	...	...	...	.
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Dram
		Rodolpho	Rodolpho			

In [179]:
```
# recall what related tables look like
df_principals
```

Out[179]:

	movie_id	ordering	person_id	category	job	characters
0	tt0111414	1	nm0246005	actor	None	["The Man"]
1	tt0111414	2	nm0398271	director	None	None
2	tt0111414	3	nm3739909	producer	producer	None
3	tt0323808	10	nm0059247	editor	None	None
4	tt0323808	1	nm3579312	actress	None	["Beth Boothby"]
...	...	...	...	...	...	...
1028181	tt9692684	1	nm0186469	actor	None	["Ebenezer Scrooge"]
1028182	tt9692684	2	nm4929530	self	None	["Herself","Regan"]
1028183	tt9692684	3	nm10441594	director	None	None
1028184	tt9692684	4	nm6009913	writer	writer	None
1028185	tt9692684	5	nm10441595	producer	producer	None

1028186 rows × 6 columns

In [180]:
```
ovie_akas to get tables that contains category and titles.
s.merge(df_movie_basics,on='movie_id').merge(df_persons,on='person_id')
```

Out[180]:

	movie_id	ordering_x	person_id	category	job	characters	primary_title	or
0	tt0111414	1	nm0246005	actor	None	["The Man"]	A Thin Life	
1	tt0111414	2	nm0398271	director	None	None	A Thin Life	
2	tt5573596	5	nm0398271	director	None	None	Remembering Nigel	Rer
3	tt0111414	3	nm3739909	producer	producer	None	A Thin Life	
4	tt0323808	10	nm0059247	editor	None	None	The Wicker Tree	-
...	...	...	...	...	...	...	...	
2972478	tt9692684	4	nm6009913	writer	writer	None	Disnatured	
2972479	tt9692684	4	nm6009913	writer	writer	None	Disnatured	
2972480	tt9692684	5	nm10441595	producer	producer	None	Disnatured	
2972481	tt9692684	5	nm10441595	producer	producer	None	Disnatured	
2972482	tt9692684	5	nm10441595	producer	producer	None	Disnatured	

2972483 rows × 22 columns

In [181]:
```
df_tn_movie_budgets
```

Out[181]:

	index	id	release_date	movie	production_budget	domestic_gross	worldwide_gros
1	5613	14	Mar 21, 1980	Mad Max	200000	8750000	9975000
2	5492	93	Sep 25, 2009	Paranormal Activity	450000	107918810	19418303
3	5679	80	Jul 10, 2015	The Gallows	100000	22764410	4165647
4	5406	7	Jul 14, 1999	The Blair Witch Project	600000	140539099	24830000
5	5709	10	May 7, 2004	Super Size Me	65000	11529368	2223380
...	...	...	...	...	...	...	..
5777	5522	23	Dec 31, 2014	Pancakes	400000	0	(
5778	5521	22	Nov 4, 2005	Show Me	400000	0	(
5779	5520	21	Apr 1, 1986	My Beautiful Laundrette	400000	0	(
5780	5116	17	Sep 8, 2015	Checkmate	1500000	0	(
5781	4982	83	Oct 14, 2008	No Man's Land: The Rise of Reeker	2000000	0	(

5781 rows × 8 columns

In [182]:
```
# merge direct_actor and movie_budgets to get table using primary_titl
df_on_primary_title_ROI = df_tn_movie_budgets.merge(df_director_actor,
```

In [183]:
```
df_on_primary_title_ROI
```

Out[183]:

	index	id	release_date	movie	production_budget	domestic_gross	worldwide_gr
0	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
1	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
2	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
3	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
4	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
...	...	...	...	...	...	...	
457179	5116	17	Sep 8, 2015	Checkmate	1500000	0	
457180	5116	17	Sep 8, 2015	Checkmate	1500000	0	
457181	5116	17	Sep 8, 2015	Checkmate	1500000	0	
457182	5116	17	Sep 8, 2015	Checkmate	1500000	0	
457183	5116	17	Sep 8, 2015	Checkmate	1500000	0	

457184 rows × 30 columns

In [184]:
```
# merge direct_actor and movie_budgets to get table using original_tit
df_on_original_title_ROI = df_tn_movie_budgets.merge(df_director_actor
```

In [185]:
```
df_on_original_title_ROI
```

Out[185]:

	index	id	release_date	movie	production_budget	domestic_gross	worldwide_gr
0	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
1	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
2	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
3	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
4	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656
...	...	...	...	...	...	...	
444874	5116	17	Sep 8, 2015	Checkmate	1500000	0	
444875	5116	17	Sep 8, 2015	Checkmate	1500000	0	

In [186]:
```
# merge direct_actor and movie_budgets to get table using aka_title
df_on_aka_title_ROI = df_tn_movie_budgets.merge(df_director_actor,left
```

In [187]:
```
df_final = pd.concat([df_on_original_title_ROI,df_on_primary_title_ROI
```

In [188]:
```
# drop duplicated records led by merge
df_final.drop_duplicates(inplace=True)
```

In [189]:
```
pd.set_option('max_columns', None)
```

In [190]:
```python
# insanity check
df_final.iloc[180:240,:-1]
```

232	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
233	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
234	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
235	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
236	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
237	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
238	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4
239	5679	80	Jul 10, 2015	The Gallows	100000	22764410	41656474	4

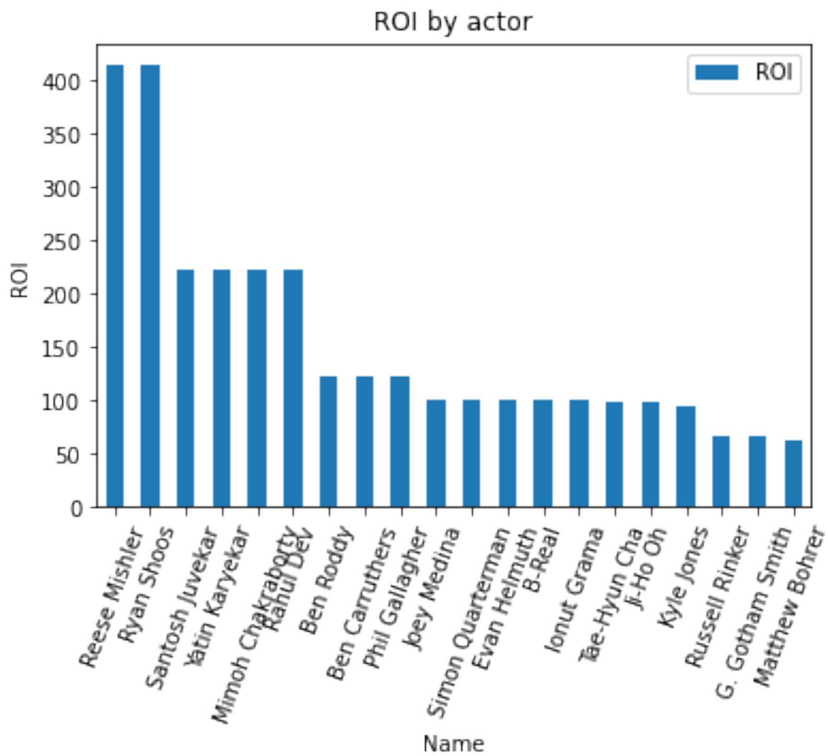
In [191]:
```python
df_final.category.value_counts()
```

Out[191]:
```
actor                  121454
writer                  93398
producer                90592
actress                 69613
director                52064
composer                23501
cinematographer         14954
editor                   7702
production_designer      2276
self                     2003
archive_footage           164
archive_sound              55
Name: category, dtype: int64
```

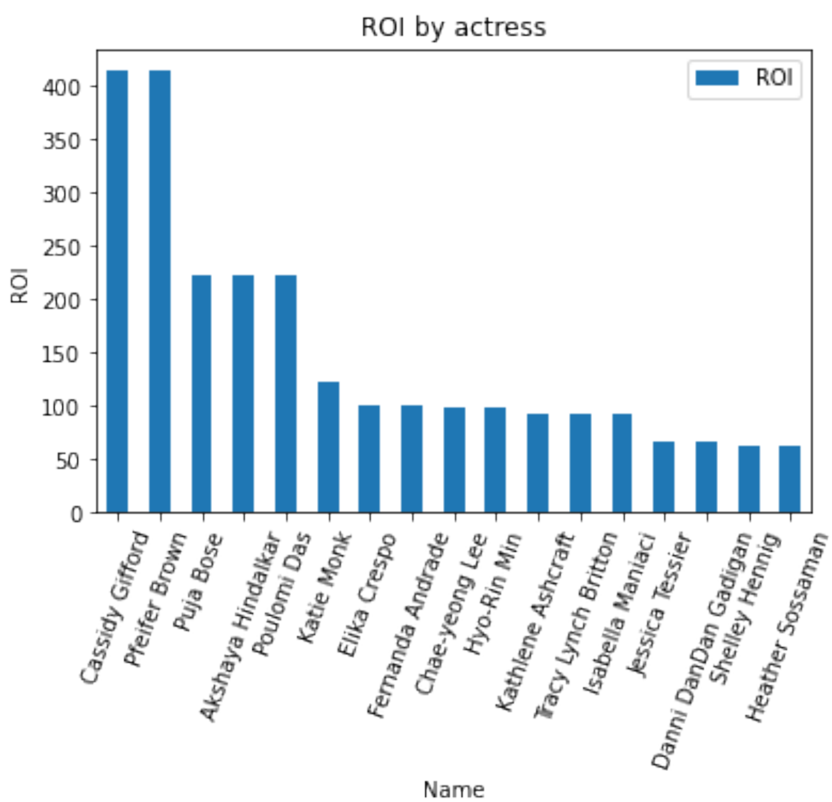
In [192]:
```python
df_final_aftergroupby = df_final.groupby(by=['category','person_id']).
```

In [193]:
```python
def get_name_bar(x):
    df_top_20_x = df_final_aftergroupby[df_final_aftergroupby.category=
    df_top_20_x_name = df_top_20_x.merge(df_final)
    df_top_20_x_name[['primary_name','ROI']].drop_duplicates().plot(kin
```

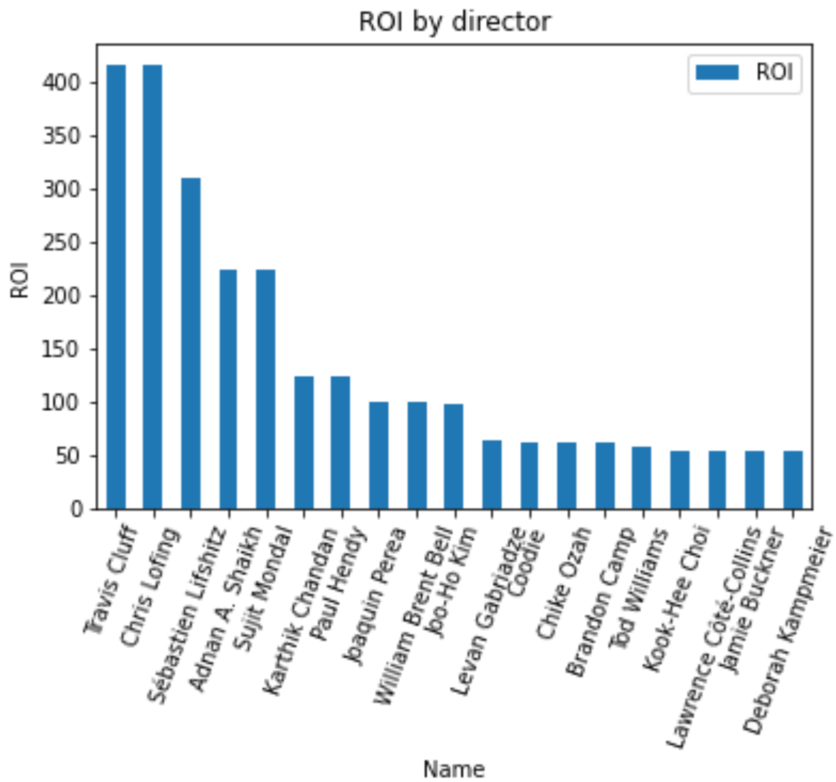
In [194]:
```
get_name_bar('actor')
plt.savefig('ROI_by_actor',bbox_inches='tight')
```



In [195]:
```
get_name_bar('actress')
plt.savefig('ROI_by_actress',bbox_inches='tight')
```



In [196]:
```
get_name_bar('director')
plt.savefig('ROI_by_director',bbox_inches='tight')
```

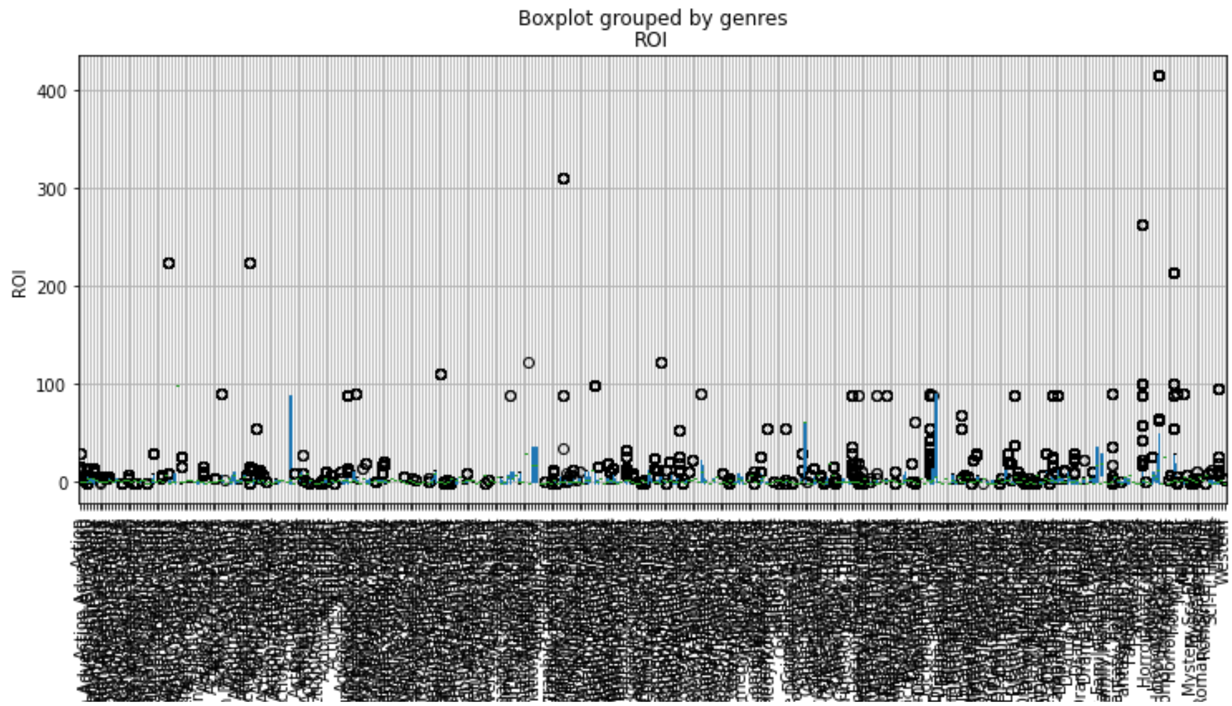


These show *top 20* **actors, directors, actresses** from above graphs. However this result is strongly *correlated* the specific **films** they are in. As top 1 and top 2 in each category, Reese Mishler, Cassiidy Gifford, Travis Cluff, Chris Lofing, Pfeifer Brown, and Ryan Shoos all are part of in *same* film.

I recommend pick top 10 personas from each category, since these persons have significant higher ROI than rest of person.

Genre

In [197]:
```
df_final.boxplot(by='genres',column='ROI',rot=90,figsize=(12,5))
plt.ylabel('ROI');
plt.savefig('ROI_by_genres',bbox_inches='tight')
```



Genres can be Drama, Documentary, Comedy, Horror, Family, Mystery, Adventure, Animation, Crime, Fatasy, War, Sci-Fi, News.........

There are so many!

I filtered that has top 10 records amount to compare **genres** that only has signifiant records size.

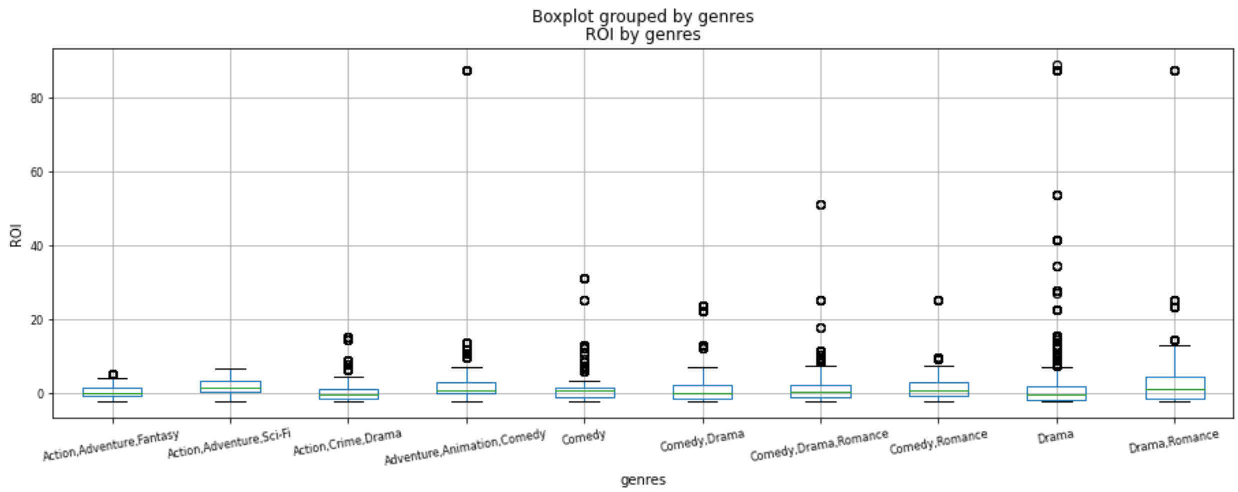
In [198]:
```
# get geners that has top 10 records amount.
df_final_genre_aftergroupby_top10 = df_final.groupby('genres').count()
```

In [199]:
```
df_final_genre_aftergroupby_top10
```

Out[199]:
```
genres
Adventure,Animation,Comedy    26779
Drama                         20713
Action,Adventure,Sci-Fi       20497
Comedy,Drama,Romance          14417
Comedy,Drama                  14351
Comedy                        13966
Action,Adventure,Fantasy      13924
Drama,Romance                 13209
Action,Crime,Drama            10648
Comedy,Romance                10230
Name: id, dtype: int64
```

In [200]:
```
# prepare final table to draw plot. Here only choose records whose gen
df_final_genre_aftergroupby_top10_1 = df_final[df_final.genres.isin(df
```

In [201]:
```
df_final_genre_aftergroupby_top10_1.boxplot(by='genres',column='ROI',r
plt.title('ROI by genres')
plt.ylabel('ROI');
plt.savefig('ROI_by_genres_top_10',bbox_inches='tight')
```



1. '*Adventure, Animation, Comedy*', '*Action, Adventrue, Sci-Fi*','*Action, Adventure, Fantasy*' are there **genres** that generates *most* **ROI** comparing to other **genres** or genres conbinations.
2. However overall each **genres** tends to have smilar **ROI** based on median comparison.
3. *Drama* has many *high* **ROI** films, however it could be the results of *high records volumns*. And majority of dramas films have *negative* **ROI**.
4. *Comedy* and *Documentary* have many films that has *high* **ROI**.

I would *recommend* setting genre as adventure and action or comedy, since films yeild high **ROI** are in these genre or commbination of these genres.

Results

This Analysis generates 3 recommendations:

1. Have **budget** set *between* 100K to 200K to have better chance to have high return on investment.
2. Pick top 10 personas from each category such as Reese Mishler, Cassiidy Gifford, Travis Cluff, Chris Lofing, Pfeifer Brown, and Ryan Shoos, since these persons have significant higher **ROI** than rest of person.
3. Set genre as adventure and action or comedy, since films yeild high **ROI** are in these genre or commbination of these genres.

More

There are some more related finds:

1. Another *measurement* here can be films **rating**, while it would provide perspective about how people like the film, ratings might not give as many insights about profit as **ROI** does.
2. There are many **Generes** that doesn't have many records. However for those not-well-produced **generes** have some films that yield high ROI. This films can be investigated further and out of this project's scope.
3. The impact of persons on films'ROI are so correlated to films themselves. If there a group of person are in same film and that film has high **ROI** then those person all have high **ROI**. So alternatively, we can investigate **ROI** for each **title**, and then choose person from top 10 films that have highest ROI.
4. Avatar has big box office, however it's budget is so high that makes it not high in ROI.

In [ ]: