Movie Data Exploration

This is an indepth walk through of my data cleaning and exploration process

Importing and exploring the data

First, importing all the data set that were provided. Running the bellow cell will importpandas that we'll use to import that data sets as well as numpy, matplotlib and the data sets themselves.

In [1]:

```
# Running this cell will take a while but will import all of the data we have in one go.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook

bom = pd.read_csv('zippedData/bom.movie_gross.csv.gz', thousands=',')
imdbName = pd.read_csv('zippedData/new_imdb/name.basics.tsv.gz', sep='\t')
imdbTitleAkas = pd.read_csv('zippedData/new_imdb/title.akas.tsv.gz', sep='\t', compression='gzip', delimi
ter='\t', encoding='iso-8859-1')
imdbTitleBasics = pd.read_csv('zippedData/new_imdb/title.basics.tsv.gz', sep='\t', compression='gzip', de
limiter='\t', encoding='iso-8859-1')
imdbTitleCrew = pd.read_csv('zippedData/new_imdb/title.crew.tsv.gz', sep='\t')
imdbTitleRatings = pd.read_csv('zippedData/new_imdb/title.ratings.tsv.gz', sep='\t')
imdbTitlePrin = pd.read_csv('zippedData/new_imdb/title.principals.tsv.gz', sep='\t')
rtMovie = pd.read_csv('zippedData/rt.movie_info.tsv.gz', sep='\t')
rtReviews = pd.read_csv('zippedData/new_imdb/name.basics.tsv.gz', sep='\t', compression='gzip', delimiter
='\t', encoding='iso-8859-1')
tmdb = pd.read_csv('zippedData/tmdb.movies.csv.gz')
tn = pd.read_csv('zippedData/tn.movie_budgets.csv.gz')
```

```
C:\Users\Sweet Deals\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshell.py:3072: Dty
peWarning: Columns (7) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
C:\Users\Sweet Deals\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshell.py:3072: Dty
peWarning: Columns (5) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

First look at our data frames

Now that all the data is imported, let's check that they imported correctly.

In [2]:

```
# This data frame looks like it's only info on boxoffice gross
bom
```

Out[2]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010
3	Inception	WB	292600000.0	535700000.0	2010
4	Shrek Forever After	P/DW	238700000.0	513900000.0	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

In [3]:

```python
# Uneccessary and becomes more trouble than it's worth to combine
imdbName
```

Out[3]:

	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
0	nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0053137,tt0072308,tt0031983
1	nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0038355,tt0071877,tt0117057,tt0037382
2	nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,music_department	tt0057345,tt0059956,tt0049189,tt0054452
3	nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0080455,tt0078723,tt0077975,tt0072562
4	nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050986,tt0050976,tt0083922,tt0060827
...	...	...	...	...	...	...
10505047	nm9993714	Romeo del Rosario	\N	\N	animation_department,art_department	tt2455546
10505048	nm9993716	Essias Loberg	\N	\N	NaN	\N
10505049	nm9993717	Harikrishnan Rajan	\N	\N	cinematographer	tt8736744
10505050	nm9993718	Aayush Nair	\N	\N	cinematographer	\N
10505051	nm9993719	Andre Hill	\N	\N	NaN	\N

10505052 rows × 6 columns

In [4]:

```python
# It appears that most of this data is either corrupted
# repeated with a different name or incomplete.
# This data is probably not worth making sense of.
imdbTitleAkas
```

Out[4]:

	titleId	ordering	title	region	language	types	attributes	isOriginalTitle
0	tt0000001	1	ÐÐ°ÑÐ¼ÐµÐ½ÑÑÑÐ°	UA	\N	imdbDisplay	\N	0
1	tt0000001	2	Carmencita	DE	\N	\N	literal title	0
2	tt0000001	3	Carmencita - spanyol tÃ¡nc	HU	\N	imdbDisplay	\N	0
3	tt0000001	4	ÎÎ±ÏÎ¼ÎµÎ½ÏÎ¯ÏÎ±	GR	\N	imdbDisplay	\N	0
4	tt0000001	5	ÐÐ°ÑÐ¼ÐµÐ½ÑÐ¸ÑÐ°	RU	\N	imdbDisplay	\N	0
...	...	...	...	...	...	...	...	...
24169281	tt9916852	4	ã¨ãã½ã¼ã #3.20	JP	ja	\N	\N	0
24169282	tt9916852	5	EpisÃ³dio #3.20	PT	pt	\N	\N	0
24169283	tt9916852	6	Episodio #3.20	IT	it	\N	\N	0
24169284	tt9916852	7	à¤à¤ªà¤¿à¤¸à¥à¤¡ #3.20	IN	hi	\N	\N	0
24169285	tt9916856	1	The Wind	DE	\N	\N	\N	0

24169286 rows × 8 columns

In [5]:

```python
# Information about movie titles and their respoective genres
imdbTitleBasics
```

Out[5]:

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
0	tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
1	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
2	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
3	tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	12	Animation,Short
4	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short
...	...	...	...	...	...	...	...	...	...

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
7342349	tt9916848	tvEpisode	Episode #3.17	Episode #3.17	0	2010	\N	\N	Action,Drama,Family
7342350	tt9916850	tvEpisode	Episode #3.19	Episode #3.19	0	2010	\N	\N	Action,Drama,Family
7342351	tt9916852	tvEpisode	Episode #3.20	Episode #3.20	0	2010	\N	\N	Action,Drama,Family
7342352	tt9916856	short	The Wind	The Wind	0	2015	\N	27	Short
7342353	tt9916880	tvEpisode	Horrid Henry Knows It All	Horrid Henry Knows It All	0	2014	\N	10	Animation,Comedy,Family

7342354 rows × 9 columns

In [6]:

```
# information about writing and directing staff.
# Usful if we need to know about hiring staff but perhaps irrelevant to monitary gain and reputation.
imdbTitleCrew
```

Out[6]:

	tconst	directors	writers
0	tt0000001	nm0005690	\N
1	tt0000002	nm0721526	\N
2	tt0000003	nm0721526	\N
3	tt0000004	nm0721526	\N
4	tt0000005	nm0005690	\N
...	...	...	...
7342349	tt9916848	nm5519454,nm5519375	nm6182221,nm1628284,nm2921377
7342350	tt9916850	nm5519454,nm5519375	nm6182221,nm1628284,nm2921377
7342351	tt9916852	nm5519454,nm5519375	nm6182221,nm1628284,nm2921377
7342352	tt9916856	nm10538645	nm6951431
7342353	tt9916880	nm0996406	nm1482639,nm2586970

7342354 rows × 3 columns

In [7]:

```
# Information about populatiry and ratings.
imdbTitleRatings
```

Out[7]:

	tconst	averageRating	numVotes
0	tt0000001	5.6	1660
1	tt0000002	6.1	203
2	tt0000003	6.5	1373
3	tt0000004	6.2	123
4	tt0000005	6.2	2161
...	...	...	...
1091113	tt9916580	7.2	5
1091114	tt9916690	6.6	5
1091115	tt9916720	6.0	66
1091116	tt9916766	6.9	15
1091117	tt9916778	7.3	24

1091118 rows × 3 columns

In [8]:

```
# Staffing information and chracter names (if applicable) to actors/actresses. Will probably cause repeart row creation
# when combining cells data frames
imdbTitlePrin
```

Out[8]:

	tconst	ordering	nconst	category	job	characters

	tconst	ordering	nconst	category	job	characters
0	tt0000001	1	nm1588970	self	\N	["Self"]
1	tt0000001	2	nm0005690	director	\N	\N
2	tt0000001	3	nm0374658	cinematographer	director of photography	\N
3	tt0000002	1	nm0721526	director	\N	\N
4	tt0000002	2	nm1335271	composer	\N	\N
...	...	...	...	...	...	...
41887285	tt9916880	5	nm0996406	director	principal director	\N
41887286	tt9916880	6	nm1482639	writer	\N	\N
41887287	tt9916880	7	nm2586970	writer	books	\N
41887288	tt9916880	8	nm1594058	producer	producer	\N
41887289	tt9916880	9	nm1052583	actress	\N	["Mum","Tidy Ted","Fang"]

41887290 rows × 6 columns

In [9]:

```
# information on box office sales and genres. Lacks movie title.
rtMovie
```

Out[9]:

<table>
<tr><th></th><th>id</th><th>synopsis</th><th>rating</th><th>genre</th><th>director</th><th>writer</th><th>theater_date</th><th>dvd_date</th><th>currency</th><th>box_office</th></tr>
<tr><td>0</td><td>1</td><td>This gritty, fast-paced, and innovative police...</td><td>R</td><td>Action and Adventure|Classics|Drama</td><td>William Friedkin</td><td>Ernest Tidyman</td><td>Oct 9, 1971</td><td>Sep 25, 2001</td><td>NaN</td><td>NaN</td></tr>
<tr><td>1</td><td>3</td><td>New York City, not-too-distant-future: Eric Pa...</td><td>R</td><td>Drama|Science Fiction and Fantasy</td><td>David Cronenberg</td><td>David Cronenberg|Don DeLillo</td><td>Aug 17, 2012</td><td>Jan 1, 2013</td><td>$</td><td>600,000</td></tr>
<tr><td>2</td><td>5</td><td>Illeana Douglas delivers a superb performance ...</td><td>R</td><td>Drama|Musical and Performing Arts</td><td>Allison Anders</td><td>Allison Anders</td><td>Sep 13, 1996</td><td>Apr 18, 2000</td><td>NaN</td><td>NaN</td></tr>
<tr><td>3</td><td>6</td><td>Michael Douglas runs afoul of a treacherous su...</td><td>R</td><td>Drama|Mystery and Suspense</td><td>Barry Levinson</td><td>Paul Attanasio|Michael Crichton</td><td>Dec 9, 1994</td><td>Aug 27, 1997</td><td>NaN</td><td>NaN</td></tr>
<tr><td>4</td><td>7</td><td>NaN</td><td>NR</td><td>Drama|Romance</td><td>Rodney Bennett</td><td>Giles Cooper</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>1555</td><td>1996</td><td>Forget terrorists or hijackers -- there's a ha...</td><td>R</td><td>Action and Adventure|Horror|Mystery and Suspense</td><td>NaN</td><td>NaN</td><td>Aug 18, 2006</td><td>Jan 2, 2007</td><td>$</td><td>33,886,034</td></tr>
<tr><td>1556</td><td>1997</td><td>The popular Saturday Night Live sketch was exp...</td><td>PG</td><td>Comedy|Science Fiction and Fantasy</td><td>Steve Barron</td><td>Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner</td><td>Jul 23, 1993</td><td>Apr 17, 2001</td><td>NaN</td><td>NaN</td></tr>
<tr><td>1557</td><td>1998</td><td>Based on a novel by Richard Powell, when the l...</td><td>G</td><td>Classics|Comedy|Drama|Musical and Performing Arts</td><td>Gordon Douglas</td><td>NaN</td><td>Jan 1, 1962</td><td>May 11, 2004</td><td>NaN</td><td>NaN</td></tr>
<tr><td>1558</td><td>1999</td><td>The Sandlot is a coming-of-age story about a g...</td><td>PG</td><td>Comedy|Drama|Kids and Family|Sports and Fitness</td><td>David Mickey Evans</td><td>David Mickey Evans|Robert Gunter</td><td>Apr 1, 1993</td><td>Jan 29, 2002</td><td>NaN</td><td>NaN</td></tr>
<tr><td>1559</td><td>2000</td><td>Suspended from the force, Paris</td><td>R</td><td>Action and Adventure|Art House and Internation</td><td>NaN</td><td>Luc Besson</td><td>Sep 27, 2001</td><td>Feb 11, 2003</td><td>NaN</td><td>NaN</td></tr>
</table>

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency	box_office

1560 rows × 12 columns

In [10]:

```
# Staffing information about previously worked on titles and brith/death years
rtReviews
```

Out[10]:

	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
0	nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0053137,tt0072308,tt0031983
1	nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0038355,tt0071877,tt0117057,tt0037382
2	nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,music_department	tt0057345,tt0059956,tt0049189,tt0054452
3	nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0080455,tt0078723,tt0077975,tt0072562
4	nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050986,tt0050976,tt0083922,tt0060827
...	...	...	...	...	...	...
10505047	nm9993714	Romeo del Rosario	\N	\N	animation_department,art_department	tt2455546
10505048	nm9993716	Essias Loberg	\N	\N	NaN	\N
10505049	nm9993717	Harikrishnan Rajan	\N	\N	cinematographer	tt8736744
10505050	nm9993718	Aayush Nair	\N	\N	cinematographer	\N
10505051	nm9993719	Andre Hill	\N	\N	NaN	\N

10505052 rows × 6 columns

In [11]:

```
# information about movie ratings based on movie title
tmdb
```

Out[11]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_coun
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	1078
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	761
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	1236
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	1017
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	2218
...	...	...	...	...	...	...	...	...	...	.
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0	
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	
26516	26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	The Church	0.0	

26517 rows × 10 columns

In [12]:

```
# information about box offic gross based on movie title.
```

```
# Information about box office gross based on movie title.
tn
```

Out[12]:

	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

Based on what we can see, as well as making some assumptions

- **Based on where the data come from, we can conclude that some of these data frames will have consistent naming/labeling schemes with one another, such as those that came from imdb.**
- **Some of the data will be usable while some will not. And that's perfecetly fine. The right information is more imporatnt than the amount of information.**
- **The column 'tconst' seems to appear quite a bit, like a unique identifier for movie titles. This is important, especially considering that some movies could be named exactly the same but are totally different.**

Let's try to join all imdb data frame on like values into one concatenated data frame.

First we need check what columns each data frame for the imdb's have in common.

In [13]:

```
# This code give a printed and organized list of columns per data frame.

print(
    '-imdbName: ', '\n',list(imdbName.columns),'\n',
    '-imdbTitleAkas: ','\n',list(imdbTitleAkas.columns),'\n',
    '-imdbTitleBasics', '\n',list(imdbTitleBasics.columns),'\n',
    '-imdbTitleCrew: ', '\n',list(imdbTitleCrew.columns),'\n',
    '-imdbTitleRatings: ', '\n',list(imdbTitleRatings.columns),'\n',
    '-imdbTitlePrin: ', '\n',list(imdbTitlePrin.columns)
)
```

```
-imdbName: 
 ['nconst', 'primaryName', 'birthYear', 'deathYear', 'primaryProfession', 'knownForTitles']
 -imdbTitleAkas: 
 ['titleId', 'ordering', 'title', 'region', 'language', 'types', 'attributes', 'isOriginalTitle']
 -imdbTitleBasics 
 ['tconst', 'titleType', 'primaryTitle', 'originalTitle', 'isAdult', 'startYear', 'endYear', 'runtimeMinut
es', 'genres']
 -imdbTitleCrew: 
 ['tconst', 'directors', 'writers']
 -imdbTitleRatings: 
 ['tconst', 'averageRating', 'numVotes']
 -imdbTitlePrin: 
 ['tconst', 'ordering', 'nconst', 'category', 'job', 'characters']
```

It looks like imdbTitleAkas doesn't have any columns in common with the rest of the data from the other imdb data frames. Not only that but imdbTitleAkas was the data frame we saw with the corrupted data. It's probably best we ommit this data frame, entirely, rather than force it to fit in.

imdbName and imdbTitlePrin also poses some problems. Combining them will result in duplicate rows as they a list of the movie staff and not the movies them selves. And since almost every movie will have more than one person on staff, we'll get a list that has movies listed for each instance of a staff member for that movie. It's best we just manually explore these later if needed rather than including it now..

As for the rest, it seems that imdbTitleBasics, imdbTitleCrew, and imdbTitleRatings have 'tconst' in common. Let's combine them on 'tconst' for one concatenated list.

In [14]:

```python
# Running this cell will combin all 3 Data frames into 1 data frame called 'imdb'.
imdbBasicsCrew = pd.merge(imdbTitleBasics, imdbTitleCrew, on=['tconst'])
imdb = pd.merge(imdbBasicsCrew, imdbTitleRatings, on=['tconst'])
```

In [15]:

```python
# Let's run imdb to check for success.
imdb
```

Out[15]:

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	directo
0	tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short	nm00056
1	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short	nm07215
2	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance	nm07215
3	tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	12	Animation,Short	nm07215
4	tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short	nm00056
...	...	...	...	...	...	...	...	...	...	
1091113	tt9916580	tvEpisode	Horrid Henry Horrid Boy?	Horrid Henry Horrid Boy?	0	2012	\N	10	Animation,Comedy,Family	nm09964
1091114	tt9916690	tvEpisode	Horrid Henry Delivers the Milk	Horrid Henry Delivers the Milk	0	2012	\N	\N	Animation,Comedy,Family	nm09964
1091115	tt9916720	short	The Nun 2	The Nun 2	0	2019	\N	10	Comedy,Horror,Mystery	nm105386
1091116	tt9916766	tvEpisode	Episode #10.15	Episode #10.15	0	2019	\N	43	Family,Reality-TV	
1091117	tt9916778	tvEpisode	Escape	Escape	0	2019	\N	\N	Drama	

1091118 rows × 13 columns

Success!...kinda. It seems our new data frame seems to have some extrenuous data (namely most values in the column 'titleType'). For simplicity's sake, we only care about box office movies. Let's check to see how many titleTypes we need to filter out.

In [16]:

```python
pd.unique(imdb['titleType'])
```

Out[16]:

```
array(['short', 'movie', 'tvShort', 'tvSeries', 'tvMovie', 'tvEpisode',
       'tvMiniSeries', 'tvSpecial', 'video', 'videoGame'], dtype=object)
```

Out of these, we only TRUELY care about 'movie'. Let's filter everything else out. And while were at it let's get rid of all rows whos column value for 'isAdult' is equal to 1...for obvious reasons...

In [17]:

```python
imdb.drop(imdb.loc[imdb['titleType']=='tvEpisode'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='tvMiniSeries'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='tvMovie'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='tvShort'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='tvSpecial'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='tvSeries'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='video'].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='short'].index, inplace=True)
imdb.drop(imdb.loc[imdb['isAdult']==1].index, inplace=True)
imdb.drop(imdb.loc[imdb['titleType']=='videoGame'].index, inplace=True)
```

While we're at it, let's remove some columns that don't provide any usful information/aren't needed anymore.

In [18]:

```
imdb = imdb.drop(columns = ['tconst', 'titleType', 'isAdult', 'endYear'], axis = 1)
```

In [19]:

```
# Let's check for' success!
imdb
```

Out[19]:

	primaryTitle	originalTitle	startYear	runtimeMinutes	genres	directors	
8	Miss Jerry	Miss Jerry	1894	45	Romance	nm0085156	nm0
259	Soldiers of the Cross	Soldiers of the Cross	1900	\N	Biography,Drama	nm0095714,nm0675140	
337	Bohemios	Bohemios	1905	100	\N	nm0063413	nm0063413,nm0657268,nm0
371	The Story of the Kelly Gang	The Story of the Kelly Gang	1906	70	Biography,Crime,Drama	nm0846879	nm0
391	Robbery Under Arms	Robbery Under Arms	1907	\N	Drama	nm0533958	nm0092809,nm0
...	...	...	...	...	...	...	
1091096	DrÃ¸mmeland	DrÃ¸mmeland	2019	72	Documentary	nm5684093	
1091097	Safeguard	Safeguard	2020	90	Action,Adventure,Thriller	nm7308376	nm7
1091105	Coven	Akelarre	2020	90	Drama,History,Horror	nm1893148	nm1893148,nm3
1091108	The Secret of China	Hong xing zhao yao Zhong guo	2019	\N	Adventure,History,War	nm0910951	
1091109	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123	Drama	nm4457074	nm4843252,nm4900525,nm2

251368 rows × 9 columns

Success! We've cut down our data frame size from 1 mill+ to just over 250k results. This 'imdb' Data frame becomes one of data frame we use for querries about genre, reviews, and release years

Now, just like the imdb data frames, we also have rt data frames that are similar in name. Let's check to see if there is any consistance between these.

In [20]:

```
print(
    '-rtMovie: ', '\n',list(rtMovie.columns),'\n',
    '-rtReviews: ','\n',list(rtReviews.columns),'\n'
)
```

```
-rtMovie: 
 ['id', 'synopsis', 'rating', 'genre', 'director', 'writer', 'theater_date', 'dvd_date', 'currency', 'box_
office', 'runtime', 'studio'] 
 -rtReviews: 
 ['nconst', 'primaryName', 'birthYear', 'deathYear', 'primaryProfession', 'knownForTitles'] 
```

Hm, unfortunate. It looks like these two data frames don't have any joinable columns in common. rtMovie has some usful info about box office gross but no assocciated movie title. Vice versa, rtReviews has no usful info but presumably the assocciated movie titles for rtMovies. With no way to reliably join the two, we'll need to ommit them both.

Next, let's clean up some of the other data frames that are still usful to us. Tmdb has a some extranious info that we just don't need. 'genre_id' & 'id' are some columns we don't need. We need to make a decision about the data here as well. Since Microsoft is american company new to the movie scene it would be safest to start with english movies so let's filter out movies that are originally english.

In [21]:

```
# This code filters and removes all movies that aren't english (column value that isn't 'en').
tmdb.drop(tmdb.loc[tmdb['original_language']!='en'].index, inplace=True)
# This code removes columns that we don't need. Since we already filtered by languages, we dont need 'ori
ginal-language' anymore.
tmdb = tmdb.drop(columns = ['genre_ids', 'id', 'original_language'])
# Check for success!
tmdb
```

Out[21]:

	Unnamed: 0	original_title	popularity	release_date	title	vote_average	vote_count
0	0	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	1	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	2	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	3	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	4	Inception	27.920	2010-07-16	Inception	8.3	22186
...	...	...	...	...	...	...	...
26512	26512	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	1
26513	26513	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	1
26514	26514	The Last One	0.600	2018-10-01	The Last One	0.0	1
26515	26515	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	1
26516	26516	The Church	0.600	2018-10-05	The Church	0.0	1

23291 rows × 7 columns

Now, lets take at our tn data frame it looks like our columns with currency values are wirtten as stings instead of a numeric. This will cause a sorting problem so lets write a function to fix that.

In [22]:

```
# this code removes non-numeric symbols, leaving only the numbers and replacing the value in its place as
a float
def dirtyMoney(x):
    if isinstance(x, str):
        return(x.replace('$', '').replace(',', ''))
    return(x)

# This code will apply the new function to the colums that represent money.
tn.worldwide_gross = tn.worldwide_gross.apply(dirtyMoney).astype(np.int64)
tn.domestic_gross = tn.domestic_gross.apply(dirtyMoney).astype(np.int64)
tn.production_budget = tn.production_budget.apply(dirtyMoney).astype(np.int64)

# This code will attempt to sort one of the columns so if our new function worked.

tn.sort_values('domestic_gross', ascending = False)
```

Out[22]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279
41	42	Feb 16, 2018	Black Panther	200000000	700059566	1348258224
6	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	2048134200
42	43	Dec 19, 1997	Titanic	200000000	659363944	2208208395
...	...	...	...	...	...	...
2709	10	Mar 31, 2004	The Touch	20000000	0	5918742
2708	9	Apr 13, 2010	Three Kingdoms: Resurrection of the Dragon	20000000	0	22139590
2707	8	Dec 31, 2012	Zambezia	20000000	0	34454336
2706	7	Dec 31, 2008	Admiral	20000000	0	38585047
5317	18	Jun 24, 2014	A Fine Step	1000000	0	0

5782 rows × 6 columns

Success! Lastely, data frame bom has a column 'studio' doesn't provide any significant info. Let's remove that as well as filling some of the NaN values with 0.

In [23]:

```
bom = bom.fillna(0)
bom = bom.drop(columns = 'studio')
bom
```

Out[23]:

Out[23]:

	title	domestic_gross	foreign_gross	year
0	Toy Story 3	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	334200000.0	691300000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	296000000.0	664300000.0	2010
3	Inception	292600000.0	535700000.0	2010
4	Shrek Forever After	238700000.0	513900000.0	2010
...	...	...	...	...
3382	The Quake	6200.0	0.0	2018
3383	Edward II (2018 re-release)	4800.0	0.0	2018
3384	El Pacto	2500.0	0.0	2018
3385	The Swan	2400.0	0.0	2018
3386	An Actor Prepares	1700.0	0.0	2018

3387 rows × 4 columns

Success!

Let's recall what we have so far and take a look at what we have once more.

- imdb - info about genre and rating per movie
- tmdb - info about genre and rating per movie
- tn - info bout box office gross per movie
- bom - info bout box office gross per movie

In [24]:

```
imdb.head()
```

Out[24]:

	primaryTitle	originalTitle	startYear	runtimeMinutes	genres	directors	writers
8	Miss Jerry	Miss Jerry	1894	45	Romance	nm0085156	nm0085156
259	Soldiers of the Cross	Soldiers of the Cross	1900	\N	Biography,Drama	nm0095714,nm0675140	\N
337	Bohemios	Bohemios	1905	100	\N	nm0063413	nm0063413,nm0657268,nm0675388
371	The Story of the Kelly Gang	The Story of the Kelly Gang	1906	70	Biography,Crime,Drama	nm0846879	nm0846879
391	Robbery Under Arms	Robbery Under Arms	1907	\N	Drama	nm0533958	nm0092809,nm0533958

In [25]:

```
tmdb.head()
```

Out[25]:

	Unnamed: 0	original_title	popularity	release_date	title	vote_average	vote_count
0	0	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	1	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	2	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	3	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	4	Inception	27.920	2010-07-16	Inception	8.3	22186

In [26]:

```
tn.head()
```

Out[26]:

	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

In [27]:

```
bom.head()
```

Out[27]:

	title	domestic_gross	foreign_gross	year
0	Toy Story 3	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	334200000.0	691300000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	296000000.0	664300000.0	2010
3	Inception	292600000.0	535700000.0	2010
4	Shrek Forever After	238700000.0	513900000.0	2010

Between all 4 of these data frames we've cleaned and made it seems like each have something new to offer. So, we may need to combine them to find a correlation between separate information that we may later deem pertinent. To make it easier on outselves later, we take the name of the movies as our join point. Though it may not be the most reliable, it's the one column every data frame has in common.

In [28]:

```
imdb = imdb.rename(columns = {'primaryTitle': 'movieTitle'})
bom = bom.rename(columns = {'title': 'movieTitle'})
tn = tn.rename(columns = {'movie': 'movieTitle'})
tmdb = tmdb.rename(columns = {'title': 'movieTitle'})
```

In [29]:

```
tmdb
```

Out[29]:

	Unnamed: 0	original_title	popularity	release_date	movieTitle	vote_average	vote_count
0	0	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	1	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	2	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	3	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	4	Inception	27.920	2010-07-16	Inception	8.3	22186
...	...	...	...	...	...	...	...
26512	26512	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	1
26513	26513	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	1
26514	26514	The Last One	0.600	2018-10-01	The Last One	0.0	1
26515	26515	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	1
26516	26516	The Church	0.600	2018-10-05	The Church	0.0	1

23291 rows × 7 columns

Second exploration and application for findind desired results

Our first step here is to define what result we want by the questions we have.

- **1.) What movies made the most profit? Are there any traits in common with those?**
- **2.) What movies have the highest Rating?**
- **3.) How should Microsoft take this information and act upon it.**

Now, we can finally ask our data some questions and get usful infomation. Since were looking for exacutable information for a large corperation money will be our driving factor.

1.) What movies made the most profit? Are there any traits in common with those?

Let's start by sorting the values of oclumn 'domestic_gross' from our **bom** data frame and return only the top 50 results. We'll save this sortying as it's own variable so that we don't mess with the original data.

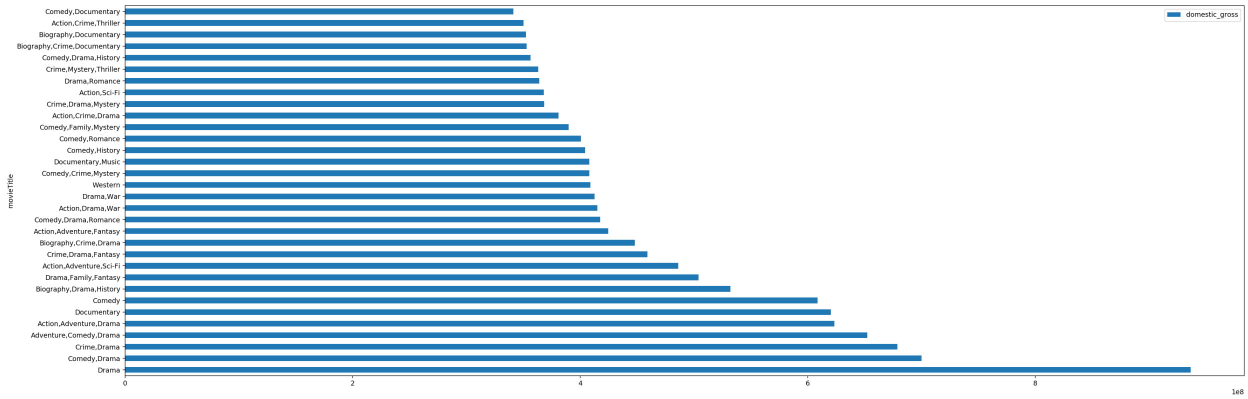
In [30]:

```
bomDom = bom.sort_values('domestic_gross', ascending=False).head(50)
```

Next, lets use matplotlib to help create a nice horizontal bar graph for data visualization. We'll be sure to save this data visualization for later viewing.

In [31]:

```
# This data take in our data frame 'bomDom' and returns a bar graph with the results of
# displyed per movie title
bomDom.plot('movieTitle', 'domestic_gross', kind='barh', figsize = (30, 10))
plt.savefig('plots.pdf')
```



So, we have a nice and neat histogram of what some of the most profitable movies have been for a domestic audience. But that doesn't help us unless we know the properties of these movies. We need to know what *about* the movie made it desireable. It's not all lost, however. What it does let us know is that, the group we've grabbed are highest earners. That means that any data within the defined group is all relevant and we already filtered out data that could be considered invalid. Microsoft, the big corperation that they are, would be interested in making as much money as they can. Let's merge on 'title' with one of our other data frames to take a look at some other information about each movie.

In [32]:

```
imdbBom = pd.merge(imdb, bom, on=['movieTitle'])
imdbBom
```

Out[32]:

	movieTitle	originalTitle	startYear	runtimeMinutes	genres	directors	write
0	Passion	Madame DuBarry	1919	85	Biography,Drama,Romance	nm0523932	nm0266183,nm04731
1	Passion	Passion	1954	84	Adventure,Western	nm0245385	nm0237532,nm0655755,nm0500837,nm02614
2	Passion	Passion	1982	88	Comedy,Drama	nm0000419	nm0140643,nm00004
3	Passion	Zui ai	1986	95	Drama,Romance	nm0151827	nm01518
4	Passion	Ishq	1997	161	Action,Comedy,Drama	nm0409791	nm0442396,nm1007381,nm1065687,nm06614
...	...	...	...	...	...	...	
4191	How Long Will I Love U	Chao shi kong tong ju	2018	101	Comedy,Fantasy,Romance	nm6050764	nm60507
4192	Helicopter Eela	Helicopter Eela	2018	135	Drama	nm1224879	nm3784276,nm16281
4193	Last Letter	Last Letter	2020	120	Romance	nm0412517	nm04125
	Last	Ni hao					

	movieTitle	originalTitle	startYear	runtimeMinutes	genres	directors	writers
4194	Last Letter	Ni hao, Zhihua	2018	114	Drama,Romance	nm0412517	nm0412517,nm56054
4195	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84	Documentary,Music	nm10201503	

4196 rows × 12 columns

Next, we sort the value of 'domestic_gross' again to make sure we're working with the right group. Let's start with the top 1,000 highest earning movie entries.

The code bellow returns a count of every instance of the genre combinations in our data frame, just the same as it did the first time.

In [33]:

```
imdbBomDom = imdbBom.sort_values('domestic_gross', ascending=False).head(1000)
imdbBomDom['genres'].value_counts().head(50)
```

Out[33]:

```
Drama                          82
Adventure,Animation,Comedy     59
Action,Adventure,Sci-Fi        53
Comedy                         39
Comedy,Drama                   33
Documentary                    29
Comedy,Romance                 24
Action,Comedy,Crime            23
Comedy,Drama,Romance           22
Action,Adventure,Drama         21
Action,Adventure,Fantasy       20
Action,Thriller                18
Horror,Mystery,Thriller        18
\N                             17
Action,Adventure,Comedy        16
Action,Adventure,Thriller      15
Drama,Romance                  15
Drama,Thriller                 14
Action,Adventure,Animation     14
Horror                         12
Thriller                       11
Crime,Drama                    11
Biography,Drama,History        10
Action,Crime,Drama             10
Horror,Thriller                 9
Action,Crime,Thriller           9
Adventure,Family,Fantasy        9
Adventure,Comedy,Family         9
Crime,Drama,Thriller            8
Action,Drama,Thriller           8
Action,Drama                    7
Comedy,Family                   7
Crime,Drama,Mystery             7
Comedy,Crime                    6
Action,Drama,History            6
Horror,Mystery                  6
Drama,Mystery,Thriller          6
Biography,Drama                 6
Biography,Comedy,Drama          6
Action                          5
Drama,Horror,Mystery            5
Comedy,Crime,Drama              5
Action,Drama,Fantasy            5
Action,Adventure,Family         5
Adventure,Comedy,Drama          5
Action,Horror,Sci-Fi            5
Action,Adventure,Crime          5
Animation,Comedy,Family         5
Action,Sci-Fi,Thriller          5
Comedy,Family,Fantasy           4
Name: genres, dtype: int64
```

Here, we can see that the 'Drama' genre is by far the most popular. However, at a glace it would seem that 'Action', 'Adventure', 'Comedy' and 'Sci-Fi' come up more often as attributes of other genre combinations. We can see this is true by the very next two entreis right bellow Drama as show by "Action, Animation, Comdey' and 'Action, Adventure, Sci-Fi'. We can see this is true still if we narrow back in on our data.

In [34]:

```
imdbBomDom2 = imdbBom.sort_values('domestic_gross', ascending=False).head(100)
imdbBomDom2['genres'].value_counts().head(50)
```

Out[34]:

```
Action,Adventure,Sci-Fi          22
Adventure,Animation,Comedy       14
Action,Adventure,Comedy           7
Drama                             6
Action,Adventure,Fantasy          6
Action,Adventure,Thriller         4
Action,Adventure,Animation        3
Comedy                            3
Adventure,Fantasy                 3
Documentary                       2
Crime,Drama                       2
Drama,Fantasy,Romance             1
Horror                            1
Comedy,Romance                    1
Action,Adventure,Family           1
Biography,Documentary,History     1
Action,Drama,Romance              1
Action,Biography,Drama            1
Action,Adventure,Mystery          1
Documentary,Drama,Sport           1
Drama,Romance                     1
Drama,Music                       1
Thriller                          1
Animation,Comedy,Crime            1
Drama,Mystery,Thriller            1
Comedy,Crime,Thriller             1
Action,Adventure,Crime            1
Drama,Family,Fantasy              1
Family                            1
Drama,Sci-Fi,Thriller             1
Animation,Comedy,Family           1
Adventure,Drama,Thriller          1
Fantasy,Romance                   1
Adventure,Drama,Sci-Fi            1
Action,Adventure                  1
Action,Adventure,Drama            1
Sci-Fi                            1
Action                            1
Comedy,Drama                      1
Name: genres, dtype: int64
```

Drama, though still close to the top, has now moved down to 3rd place. Let's run another experiment to cover our bases. This time we'll filter out any movies that grossed less that $10,000,000 and then get a value count of genres from that new list. This will be our broadest check but remember that anything in this list made at least (and often much more) and 10 millino dollars. That is still a large enough sum of money to at least be considered valid.

In [35]:

```
imdbBomHighDoll = imdbBom.loc[imdbBom['domestic_gross'] >= 10000000]
imdbBomHighDoll['genres'].value_counts().head(50)
```

Out[35]:

```
Drama                          142
Adventure,Animation,Comedy      68
Comedy,Drama                    55
Comedy                          54
Action,Adventure,Sci-Fi         53
Drama,Romance                   51
Documentary                     41
Comedy,Romance                  40
Comedy,Drama,Romance            40
Horror,Mystery,Thriller         32
Action,Adventure,Fantasy        31
Action,Comedy,Crime             28
\N                              27
Action,Adventure,Drama          27
Action,Thriller                 25
Action,Crime,Thriller           25
Action,Crime,Drama              22
Drama,Thriller                  21
Crime,Drama,Thriller            19
Biography,Drama,History         17
Action,Adventure,Comedy         17
Thriller                        17
Biography,Drama                 16
Action,Adventure,Thriller       16
Crime,Drama                     15
```

```
Crime,Drama
Action,Adventure,Animation     15
Horror                         15
Horror,Thriller                15
Biography,Comedy,Drama         14
Action,Drama,Thriller          13
Horror,Mystery                 12
Crime,Drama,Mystery            11
Adventure,Comedy,Family        10
Biography,Crime,Drama          10
Drama,Horror,Mystery           10
Comedy,Family                  10
Action                          9
Adventure,Family,Fantasy        9
Drama,Mystery,Thriller          9
Action,Drama                    9
Action,Sci-Fi,Thriller          9
Comedy,Crime,Drama              9
Action,Comedy                   9
Drama,Horror,Thriller           8
Comedy,Crime                    8
Drama,Fantasy,Horror            8
Comedy,Drama,Music              8
Action,Drama,History            7
Adventure,Comedy,Drama          7
Drama,Fantasy                   7
Name: genres, dtype: int64
```

Inteseting! Our data now has change slightly, once again. We see that Drama commands the lead with the most common genre. But now we've see comedy has been take the 2nd, 3rd, and 4th place spot for most common. If we take a look back, we can see that comedy has been very present across every check in all the top 5 spots. People like to laugh!

Let's make a nice and neat visual horizontal bar graph for out new findings while we're still thinking about it.

In [36]:

```
imdbBomDom3 = imdbBom.sort_values('domestic_gross', ascending=False).head(50)
imdbBomDom3['genres'].value_counts().plot(kind='barh')
plt.savefig('imdbBom-highest-earners.pdf')
```

1.) What movie's made the most profit? Are there any traits in common with those?

A: The data would suggest that

- **The movies that see the most box office success, both as one of the most common and as one of the highest earners, is 'Action, Animation, & Comedy'.**
- **Drama is the most common movie type**
- **Action, Adventure, & Sci-Fi is the single highest earner genre combination**

2.) What movies have the highest Rating?

First, we'll need to take a look at our data again but sort it on numVotes. numVotes is the number of times a person has rated a particular movie. We can take a way from this that the higher the numVotes value, the more talked about a movie is. However, 'most popular' doesn't always mean 'best'. It would be more accurate to say that higher NumVotes only means more attention and attention is one thing we need more of for building a brand image.

The code bellow filters out any rows bellow 10,000 votes and then sorts them by the highest rated movies first. We've invalidating those movies who didn't set a good enough example for gaining traction.

In [37]:

```
imdbNumV = imdb.loc[imdb['numVotes'] >= 10000]
imdbNumV.sort_values('numVotes', ascending=False)
```

Out[37]:

	movieTitle	originalTitle	startYear	runtimeMinutes	genres	directors	
81555	The Shawshank Redemption	The Shawshank Redemption	1994	142	Drama	nm0001104	nm0000175,nm0
245359	The Dark Knight	The Dark Knight	2008	152	Action,Crime,Drama	nm0634240	nm0634300,nm0634240,nm0333060,nm0
571614	Inception	Inception	2010	148	Action,Adventure,Sci-Fi	nm0634240	nm0
97802	Fight Club	Fight Club	1999	139	Drama	nm0000399	nm0657333,nm0

	movieTitle	originalTitle	startYear	runtimeMinutes	genres	directors	
81339	Pulp Fiction	Pulp Fiction	1994	154	Crime,Drama	nm0000233	nm0000233,nm0(
...	...	...	...	...	...	...	
652948	Queen of the Desert	Queen of the Desert	2015	128	Adventure,Biography,Drama	nm0001348	nm0(
77013	Trespass	Trespass	1992	101	Action,Thriller	nm0001353	nm0301826,nm0(
953827	Badrinath Ki Dulhania	Badrinath Ki Dulhania	2017	139	Comedy,Drama,Romance	nm4264671	nm42
450113	Gulabo Sitabo	Gulabo Sitabo	2020	124	Comedy,Drama	nm1999473	nm4!
101676	Two Hands	Two Hands	1999	103	Comedy,Crime,Thriller	nm0429964	nm0

8787 rows × 9 columns

Next, we can use the new filtered data frame to sort out our averageRating to find what genres were the most liked. As we're trying to only validate movies that had a rating betetr than a D, anything less that a 7.0 rating will be ommitted. Just like last time, we'll take the top 100 ratings, top 1,000, and create a cut-off for totally invalidadted rows (voted at least 7.0).

In [38]:

```
imdbAvRat = imdbNumV.sort_values('averageRating', ascending = False).head(100)
imdbAvRat2 = imdbNumV.sort_values('averageRating', ascending = False).head(1000)
imdbAvRat3 = imdbNumV.loc[imdb['averageRating'] >= 7.0]
```

In [39]:

```
imdbAvRat['genres'].value_counts().head(50)
```

Out[39]:

```
Drama                               11
Comedy,Drama                         6
Crime,Drama                          6
Action,Crime,Drama                   5
Action,Adventure,Drama               5
Drama,Romance                        4
Adventure,Comedy,Drama               3
Crime,Drama,Thriller                 3
Documentary                          3
Comedy,Romance                       3
Action,Adventure,Fantasy             2
Comedy                               2
Action,Sci-Fi                        2
Biography,Drama,History              2
Biography,Crime,Documentary          2
Western                              2
Comedy,Drama,Romance                 2
Biography,Documentary,Sport          1
Adventure,Animation,Drama            1
Action,Drama,Mystery                 1
Comedy,Crime,Drama                   1
Comedy,Family,Mystery                1
Adventure,Comedy,Sci-Fi              1
Drama,Thriller                       1
Comedy,History                       1
Comedy,Drama,History                 1
Biography,Documentary                1
Comedy,Drama,Family                  1
Drama,Music                          1
Crime,Mystery,Thriller               1
Biography,Crime,Drama                1
Biography,Documentary,History        1
Crime,Drama,Fantasy                  1
Comedy,Drama,Thriller                1
Documentary,Music                    1
Comedy,Drama,Western                 1
Animation,Comedy,Family              1
Drama,Mystery,Sci-Fi                 1
Comedy,Documentary                   1
Horror,Mystery,Thriller              1
Adventure,Comedy,Crime               1
Adventure,Drama,Sci-Fi               1
Action,Drama,War                     1
Biography,Drama,Music                1
Action,Crime,Thriller                1
Comedy,Crime,Mystery                 1
Crime,Drama,Mystery                  1
Drama,Romance,War                    1
```

```
Drama,Romance,War                  1
Drama,War                          1
Comedy,Crime,Romance               1
Name: genres, dtype: int64
```

In [40]:

```
imdbAvRat2['genres'].value_counts().head(50)
```

Out[40]:

```
Drama                            102
Drama,Romance                     45
Comedy,Drama                      42
Comedy,Drama,Romance              30
Crime,Drama                       25
Action,Crime,Drama                25
Biography,Drama,History           23
Crime,Drama,Thriller              22
Drama,War                         21
Crime,Drama,Mystery               19
Documentary                       19
Comedy,Romance                    16
Adventure,Animation,Comedy        15
Biography,Drama                   15
Action,Adventure,Sci-Fi           14
Action,Adventure,Animation        12
Comedy,Crime,Drama                12
Crime,Drama,Film-Noir             11
Biography,Crime,Drama             11
Drama,Thriller                    11
Comedy                            11
Adventure,Comedy,Drama             9
Action,Adventure,Drama             8
Adventure,Biography,Drama          8
Drama,Family                       7
Documentary,Music                  7
Comedy,Drama,Family                7
Action,Adventure,Comedy            7
Mystery,Thriller                   7
Drama,Music                        7
Action,Thriller                    7
Action,Biography,Drama             6
Drama,Music,Romance                6
Drama,Mystery,Thriller             6
Action,Crime,Thriller              6
Biography,Drama,Sport              6
Comedy,Drama,War                   6
Biography,Comedy,Drama             6
Action,Drama                       6
Comedy,Drama,Fantasy               5
Biography,Documentary,Music        5
Adventure,Drama,Fantasy            5
Drama,Romance,War                  5
Comedy,Drama,Music                 5
Action,Drama,Thriller              5
Action,Adventure,Fantasy           5
Drama,Sport                        5
Drama,Fantasy,Romance              5
Biography,Crime,Documentary        5
Action,Comedy,Crime                5
Name: genres, dtype: int64
```

In [41]:

```
imdbAvRat3['genres'].value_counts().head(50)
```

Out[41]:

```
Drama                            266
Comedy,Drama                     189
Drama,Romance                    169
Comedy,Drama,Romance             169
Crime,Drama,Thriller             109
Action,Crime,Drama               106
Biography,Drama,History           80
Crime,Drama                       78
Crime,Drama,Mystery               70
Comedy                            66
Adventure,Animation,Comedy        61
Biography,Drama                   57
Biography,Crime,Drama             51
Comedy,Crime,Drama                47
Drama,Thriller                    46
Action,Adventure,Sci-Fi           46
```

```
Action,Adventure,Sci-Fi         46
Documentary                     45
Drama,War                       42
Comedy,Romance                  42
Action,Crime,Thriller           39
Action,Adventure,Drama          38
Action,Adventure,Comedy         37
Drama,Mystery,Thriller          35
Biography,Comedy,Drama          35
Comedy,Drama,Music              34
Action,Adventure,Animation      34
Action,Comedy,Crime             31
Comedy,Drama,Fantasy            30
Crime,Drama,Romance             28
Adventure,Comedy,Drama          27
Biography,Drama,Sport           27
Action,Drama,History            27
Action,Biography,Drama          25
Biography,Drama,Romance         22
Action,Drama,Thriller           22
Horror                          21
Comedy,Crime                    21
Crime,Drama,Film-Noir           21
Action,Thriller                 20
Action,Adventure,Thriller       20
Biography,Drama,Music           20
Adventure,Biography,Drama       19
Drama,History,War               18
Adventure,Animation,Drama       18
Drama,Mystery,Romance           18
Drama,Western                   17
Action,Adventure,Fantasy        17
Comedy,Drama,Family             17
Drama,Romance,War               17
Drama,Music                     16
Name: genres, dtype: int64
```

So, here is something interesting that we didn't see in the our domestic_gross check. Here, we see that Drama is the undisputed champion of all with not only the most common grenre type for all three checks but also holds the very top spot for being the highest rated genre at 9.3 Average Rating

However, once again wee see that comedy has come in second again as 'Comedy, Drama' for the most common data type for 2/3 checks

Once again, let's make a nice and neat Horizontal Bar Graph

In [42]:

```python
imdbAvRat4 = imdbNumV.sort_values('averageRating', ascending = False).head(50)
imdbAvRat4['genres'].value_counts().plot(kind='barh')
plt.savefig('imdbBom-highest-rated-genres.pdf')
```

So, here is something interesting that we didn't see in the our domestic_gross check. Here, we see that Drama is the undisputed champion of all with not only the most common grenre type for all three checks but also holds the very top spot for being the highest rated genre at 9.3 Average Rating

However, once again wee see that comedy has come in second again as 'Comedy, Drama' for the most common data type for 2/3 checks

2.) What movies have the highest Rating?

The data shows that

- **Purely drama movies have the highest ratings as well as being the most common genre type.**
- **The second most common genre combination is Comedy & Drama**

3.) How should Microsoft take this information and act upon it.

As an observation, I'd say that the two results couldn't be more different from each other. But it's important to hit on both Top Earner genres and Top Rated genres. Building up brand image will be the best way to start on the right foot. A pure drama or 'drama, comedy' is statistically the safest way to build up a likable reputaion without sacraficing of monitary gain completely. Once a customer trust is foraged and you brand is known for making likable movies, it would then be smart to move into 'Action, Adventure, Sci-Fi' or 'Action, Animation, Comedy' for a prefrence of monetary gain to build up bigger budgets for next projects.