> **Tip**: Welcome to the Investigate a Dataset project! You will find tips in quoted sections like this to help organize your approach to your investigation. Once you complete this project, remove these **Tip** sections from your report before submission. First things first, you might want to double-click this Markdown cell and change the title so that it reflects your dataset and investigation.

Project: Investigate a Dataset - [IMDB MOVIE - DATA SET]

Table of Contents

Introduction

Dataset Description

> **Tip**: In this section of the report, provide a brief introduction to the dataset you've selected/downloaded for analysis. Read through the description available on the homepage-links present here (https://docs.google.com/document/d/e/2PACX-1vTlVmknRRnfy_4eTrjw5hYGaiQim5ctr9naaRd4V9du2B5bxpd8FEH3KtDgp8qVekw7Cj1GLk1IXdZi/pub?embedded=True). List all column names in each table, and their significance. In case of multiple tables, describe the relationship between tables.

Question(s) for Analysis

> **Tip**: Clearly state one or more questions that you plan on exploring over the course of the report. You will address these questions in the **data analysis** and **conclusion** sections. Try to build your report around the analysis of at least one dependent variable and three independent variables. If you're not sure what questions to ask, then make sure you familiarize yourself with the dataset, its variables and the dataset context for ideas of what to explore.

> **Tip**: Once you start coding, use NumPy arrays, Pandas Series, and DataFrames where appropriate rather than Python lists and dictionaries. Also, **use good coding practices**, such as, define and use functions to avoid repetitive code. Use appropriate comments within the code cells, explanation in the mark-down cells, and meaningful variable names.

TMDB MOVIE DATA ANALYSIS

> This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue. Certain columns, like ‘cast’ and ‘genres’, contain multiple values separated by pipe (|) characters. There are some odd characters in the ‘cast’ column. Don’t worry about cleaning them. You can leave them as is. The final two columns ending with “_adj” show the budget and revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

In [106]:
```
# Use this cell to set up import statements for all of the packages that you
#   plan to use.

# Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html


# Importing important packages

import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling

> **Tip**: In this section of the report, you will load in the data, check for cleanliness, and then trim and clean your dataset for analysis. Make sure that you **document your data cleaning steps in mark-down cells precisely and justify your cleaning decisions.**

General Properties

> **Tip**: You should *not* perform too many operations in each cell. Create cells freely to explore your data. One option that you can take with this project is to do a lot of explorations in an initial notebook. These don't have to be organized, but make sure you use enough comments to understand the purpose of each code cell. Then, after you're done with your analysis, create a duplicate notebook where you will trim the excess and organize your steps so that you have a flowing, cohesive report.

In [107]:
```
# Load your data and print out a few lines. Perform operations to inspect data
# types and look for instances of missing or possibly errant data.

df_movies = pd.read_csv('tmdb-movies.csv')
```

In [108]:
```
df_movies.shape
```

Out[108]:
```
(10866, 21)
```

Shape of the Data

> It can be seen that the TMDB movie data has initail rows of 10866 and colums of 21.

In [109]:
```
df_movies.head()
```

Out[109]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force	Harrison Ford|Mark Hamill|Carrie	http://www.starwars.com/films/star-wars-episod	J.J. Abrams	Every generation has a	...

In [110]:
```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
```

> A list all column names in each table, and their significance with the relationship between tables are explained in the table below ::

Serial	Field	Importance
1	id	primary key for data frame
2	imdb_id	key reference in imdb database
3	popularity	how well the movie was known by public
4	budget	amount spent on the movie
5	revenue	amount of money made from the movie - reflects current figures
6	original_title	movie title
7	cast	Actor and actresses listed in the order they appear in the credits and separated by
8	homepage	movie's website
9	director	movie director
10	tagline	popular catch theme for the movie
11	keywords	search, words

Serial	Field	Importance
12	overview	summary of plot
13	runtime	lenghth of movie
14	genres	type of movie
15	production_companies	Companies involved in production
16	release_date	date released
17	vote_count	No of votes for the movie
18	vote_average	movie ratings
19	release_year	Year of release
20	budget_adj	Adjusted budget to account for inflation
21	revenue_adj	adjusted revenue to account for inflation

Null Data Fields

It can be observed that tData fram has some fields with null values. The Fields are :

1. imdb_id
2. cast
3. homepage
4. director
5. tagline
6. keywords
7. overview
8. genres
9. production_companies

In [111]:
```
#determine total number of each of empty fields across columns

result = df_movies[['imdb_id', 'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview', 'genres', 'productio
]].isnull().sum();

print(result);
```

```
imdb_id                 10
cast                    76
homepage              7930
director                44
tagline               2824
keywords              1493
overview                 4
genres                  23
production_companies  1030
dtype: int64
```

In [112]:
```
#identification of data types

df_movies.dtypes
```

Out[112]:
```
id                      int64
imdb_id                object
popularity            float64
budget                  int64
revenue                 int64
original_title         object
cast                   object
homepage               object
director               object
tagline                object
keywords               object
overview               object
runtime                 int64
genres                 object
production_companies   object
release_date           object
vote_count              int64
vote_average          float64
release_year            int64
budget_adj            float64
revenue_adj           float64
dtype: object
```

Data Cleaning

Tip: Make sure that you keep your reader informed on the steps that you are taking in your investigation. Follow every code cell, or every set of related code cells, with a markdown cell to describe to the reader what was found in the preceding cell(s). Try to make it so that the reader can then understand what they will be seeing in the following cell(s).

In [113]:
```
# After discussing the structure of the data and any problems that need to be
# cleaned, perform those cleaning steps in the second part of this section.
```

Missing Data

The following fields have missing data which will be corrected as shown in the table below::

Serial	Field	Count of Missing Values	Data type	Filling Values
1	imdb_id	10	object	drop rows with NaN
2	cast	76	object	replace NaN with "No_dets"
3	homepage	7930	object	replace NaN with "No_dets"
4	director	44	object	replace NaN with "No_dets"
5	tagline	2824	object	replace NaN with "No_dets"
6	keywords	1493	object	replace NaN with "No_dets"
7	overview	4	object	replace NaN with "No_dets"
8	genres	23	object	replace NaN with "No_dets"
9	production_companies	1030	object	replace NaN with "No_dets"

Handling Missing Data

In [114]:
```
#drop all rows with NaN imdb_id

df_movies.dropna(subset=['imdb_id'], inplace=True)
```

In [115]:
```
#all  the 10 identified imdb_id NaN rows deleted
df_movies.shape
```

Out[115]:
```
(10856, 21)
```

In [116]:
```
#replace all other NaN values with No_dets

df_movies.fillna('no_dets', inplace=True)
```

In [117]:
```
df_movies.shape
```

Out[117]:
```
(10856, 21)
```

Duplicate Rows

In [118]:
```
#test for duplicate rows

print(df_movies.duplicated().value_counts())
```

```
False    10855
True         1
dtype: int64
```

In [119]:
```
#display duplicated rows

df_dups = df_movies[df_movies.duplicated(keep=False)]

df_dups.head()
```

Out[119]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...	overview	runtime	
2089	42194	tt0411951	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	no_dets	Dwight H. Little	Survival is no game	...	In the year of 2039, after World Wars destroy ...	92	Crime|Drama|Action|Thril
2090	42194	tt0411951	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	no_dets	Dwight H. Little	Survival is no game	...	In the year of 2039, after World Wars destroy ...	92	Crime|Drama|Action|Thril

2 rows × 21 columns

Handling Duplicate Rows

In [120]:
```
#drop duplicated rows

df_movies.drop_duplicates(inplace=True)
```

In [121]:
```
#duplicated row removed
df_movies.shape
```

Out[121]:
```
(10855, 21)
```

Fixing Data Structure & Types

In [122]:
```
#First observation is to make **imdb** id into string by deleting preceeding "tt" for performance efficiency.

df_movies['imdb_id'] = df_movies['imdb_id'].str.replace('tt', '')
```

In [123]:
```
#Also convert release_date to date format

df_movies['release_date'] = pd.to_datetime(df_movies['release_date'])
```

In [124]:
```
df_movies.dtypes
```

Out[124]:
```
id                             int64
imdb_id                       object
popularity                   float64
budget                         int64
revenue                        int64
original_title                object
cast                          object
homepage                      object
director                      object
tagline                       object
keywords                      object
overview                      object
runtime                        int64
genres                        object
production_companies          object
release_date          datetime64[ns]
vote_count                     int64
vote_average                 float64
release_year                   int64
budget_adj                   float64
revenue_adj                  float64
dtype: object
```

In [125]:
```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10855 entries, 0 to 10865
Data columns (total 21 columns):
id                    10855 non-null int64
imdb_id               10855 non-null object
popularity            10855 non-null float64
budget                10855 non-null int64
revenue               10855 non-null int64
original_title        10855 non-null object
cast                  10855 non-null object
homepage              10855 non-null object
director              10855 non-null object
tagline               10855 non-null object
keywords              10855 non-null object
overview              10855 non-null object
runtime               10855 non-null int64
genres                10855 non-null object
production_companies  10855 non-null object
release_date          10855 non-null datetime64[ns]
vote_count            10855 non-null int64
vote_average          10855 non-null float64
release_year          10855 non-null int64
budget_adj            10855 non-null float64
revenue_adj           10855 non-null float64
dtypes: datetime64[ns](1), float64(4), int64(6), object(10)
memory usage: 1.8+ MB
```

Cleaned Data

In [126]:
```
df_movies.head(20)
```

Out[126]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	
0	135397	0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	Th
1	76341	1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	
2	262500	2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	
3	140607	2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams	ge

End of Data Cleaning

Exploratory Data Analysis

Tip: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. **Compute statistics** and **create visualizations** with the goal of addressing the research questions that you posed in the Introduction section. You should compute the relevant statistics throughout the analysis when an inference is made about the data. Note that at least two or more kinds of plots should be created as part of the exploration, and you must compare and show trends in the varied visualizations.

Tip: - Investigate the stated question(s) from multiple angles. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables. You should explore at least three variables in relation to the primary question. This can be an exploratory relationship between three variables of interest, or looking at how two independent variables relate to a single dependent variable of interest. Lastly, you should perform both single-variable (1d) and multiple-variable (2d) explorations.

Brief overview of the data

In [127]:
```
df_movies.describe()
```

Out[127]:

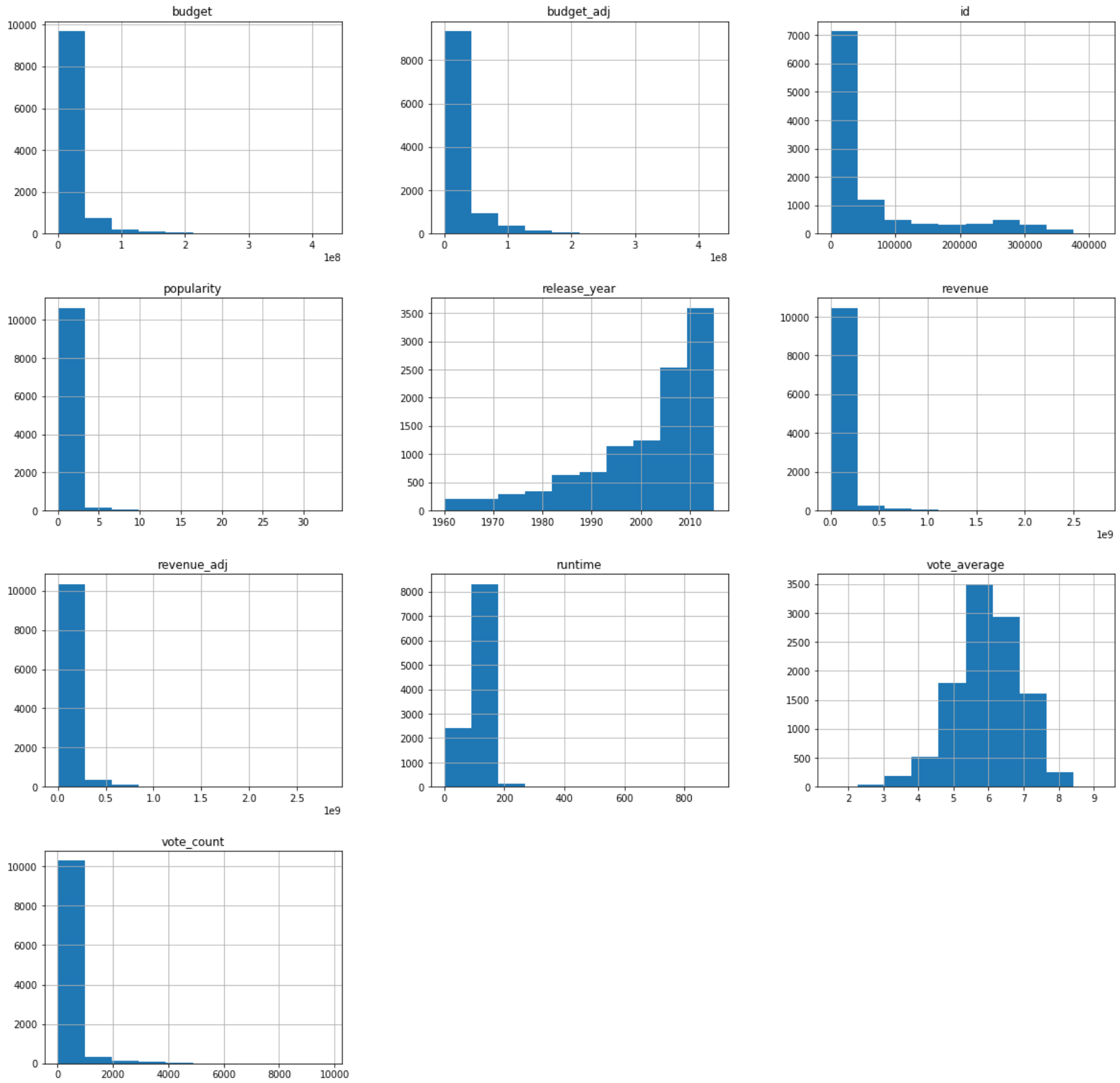
	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10855.000000	10855.000000	1.085500e+04	1.085500e+04	10855.000000	10855.000000	10855.000000	10855.000000	1.085500e+04	1.085500e+04
mean	65959.191617	0.646832	1.463776e+07	3.986359e+07	102.105205	217.584155	5.973865	2001.313128	1.756606e+07	5.141632e+07
std	92018.246342	1.000591	3.092533e+07	1.170559e+08	31.348734	575.877532	0.934604	12.815672	3.431919e+07	1.446965e+08
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+00
25%	10591.500000	0.207733	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+00
50%	20618.000000	0.383998	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+00
75%	75393.500000	0.714446	1.500000e+07	2.404727e+07	111.000000	146.000000	6.600000	2011.000000	2.085325e+07	3.374346e+07
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

In [128]:
```
#Using top 500 rated  movies

df_movies.sort_values(by=['vote_average'], ascending=[False]).head(500)
```

Out[128]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director
3894	125336	2044056	0.006925	0	0	The Story of Film: An Odyssey	Mark Cousins|Jean-Michel Frodon|Cari Beauchamp...	http://www.channel4.com/programmes/the-story-o...	Mark Cousins
538	224972	3983674	0.114264	0	0	The Mask You Live In	no_dets	http://themaskyoulivein.org	Jennifer Siebel Newsom
1200	374430	3973198	0.129531	0	0	Black Mirror: White Christmas	Jon Hamm|Rafe Spall|Oona Chaplin|Natalia Tena|...	no_dets	Carl Tibbetts
2269	51299	1828232	0.222293	0	0	Life Cycles	no_dets	http://www.lifecyclesfilm.com/	Derek Frankowski
6911	24970	0110758	0.212010	0	0	Pink Floyd: Pulse	David Gilmour|Nick Mason|Richard Wright|Sam Br...	no_dets	David Mallet

In [129]:
```
df_movies.hist(figsize = (20, 20));
```



I seek to answer the following questions after reviewing the table ::

Research Question 1 - What are the top 10 most popular movies?

Research Question 2 - What are the Highly Rated Movies?

Research Question 3 - Correlation between popularity vs High-rating?

Research Question 4 - What are the most profitable movies?

Research Question 5 - What Directors produce most movies?

Research Question 6 - What Production Companies produce most movies?

Research Question 6 - What Production Companies produce most movies?

Research Question 7 - Movie release by year?

Research Question 8 - Profitable Production Companies?

Research Question 9 - High Rated Directors Production Companies?

Research Question 1 - What are the top 10 most popular movies?

In [130]:
```
df_movies['popularity'].describe()
```

Out[130]:
```
count    10855.000000
mean         0.646832
std          1.000591
min          0.000065
25%          0.207733
50%          0.383998
75%          0.714446
max         32.985763
Name: popularity, dtype: float64
```

In [131]:
```
#show top 10 movies in lot

df_popular = df_movies[["original_title","popularity"]]

df_popular.head()
```

Out[131]:

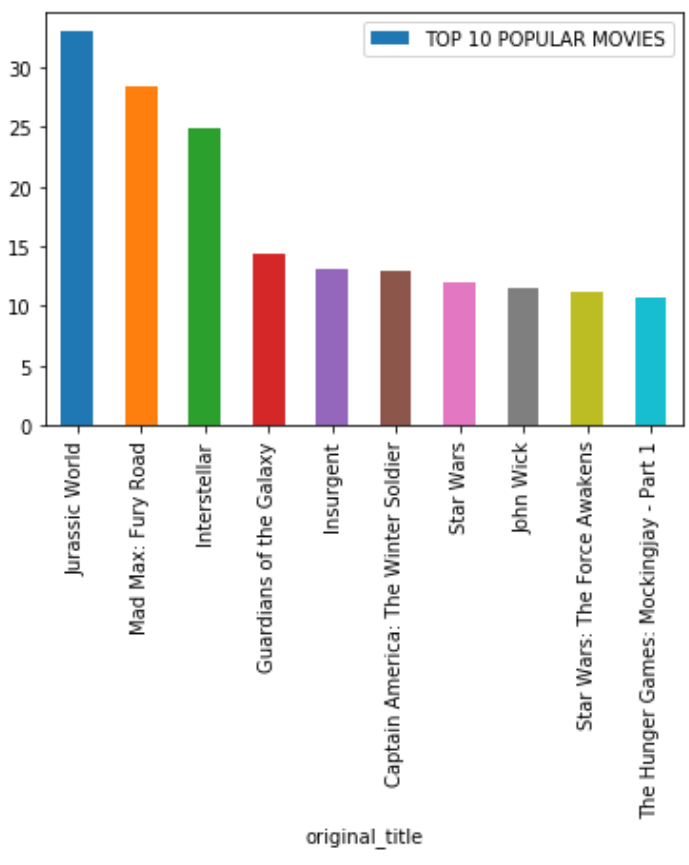
	original_title	popularity
0	Jurassic World	32.985763
1	Mad Max: Fury Road	28.419936
2	Insurgent	13.112507
3	Star Wars: The Force Awakens	11.173104
4	Furious 7	9.335014

In [132]:
```
df_pop10 = df_popular.sort_values(by=['popularity'], ascending=False).head(10)

df_pop10.head(10)
```

Out[132]:

	original_title	popularity
0	Jurassic World	32.985763
1	Mad Max: Fury Road	28.419936
629	Interstellar	24.949134
630	Guardians of the Galaxy	14.311205
2	Insurgent	13.112507
631	Captain America: The Winter Soldier	12.971027
1329	Star Wars	12.037933
632	John Wick	11.422751
3	Star Wars: The Force Awakens	11.173104
633	The Hunger Games: Mockingjay - Part 1	10.739009

In [133]:
```
df_pop10.plot(x='original_title', y='popularity', kind = 'bar', label='TOP 10 POPULAR MOVIES');
```



Research Question 2 - What are the Highly Rated Movies?

In [134]:
```
df_movies['vote_average'].describe()
```

Out[134]:
```
count    10855.000000
mean         5.973865
std          0.934604
min          1.500000
25%          5.400000
50%          6.000000
75%          6.600000
max          9.200000
Name: vote_average, dtype: float64
```

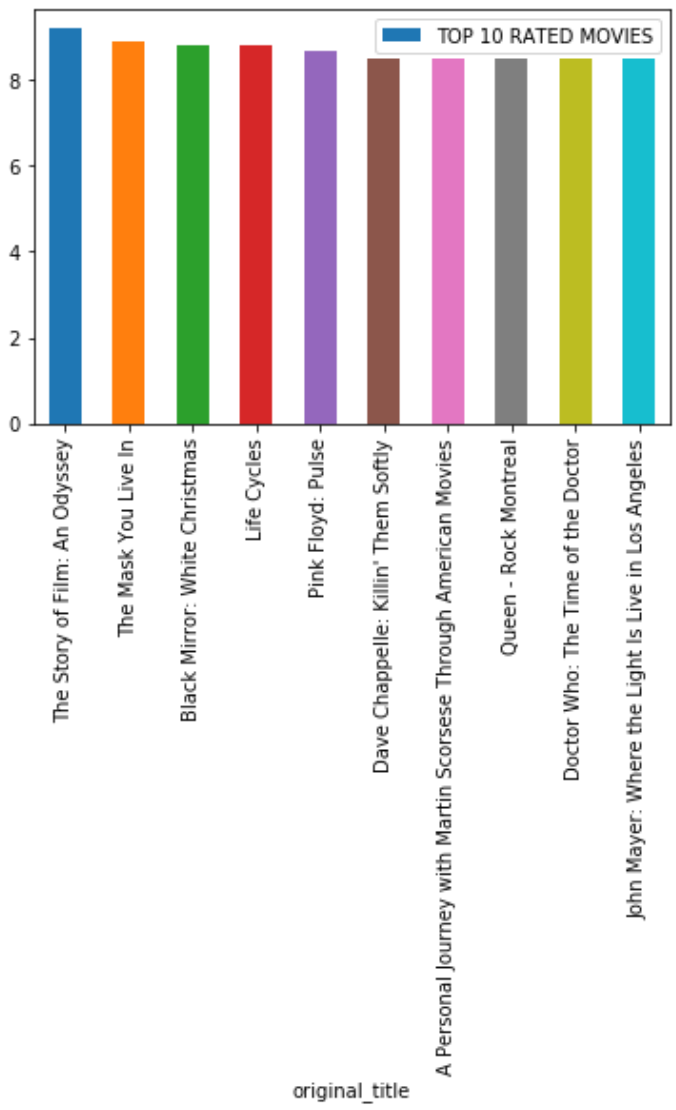
In [135]:
```
#show top 10 rated  movies in lot

df_toprated = df_movies[["original_title","vote_average"]]
```

In [136]:
```
df_top10 = df_toprated.sort_values(by=['vote_average'], ascending=[False]).head(10)

df_top10.head(10)
```

Out[136]:

	original_title	vote_average
3894	The Story of Film: An Odyssey	9.2
538	The Mask You Live In	8.9
1200	Black Mirror: White Christmas	8.8
2269	Life Cycles	8.8
6911	Pink Floyd: Pulse	8.7
8839	Dave Chappelle: Killin' Them Softly	8.5
8221	A Personal Journey with Martin Scorsese Throug...	8.5
8411	Queen - Rock Montreal	8.5
5830	Doctor Who: The Time of the Doctor	8.5
3224	John Mayer: Where the Light Is Live in Los Ang...	8.5

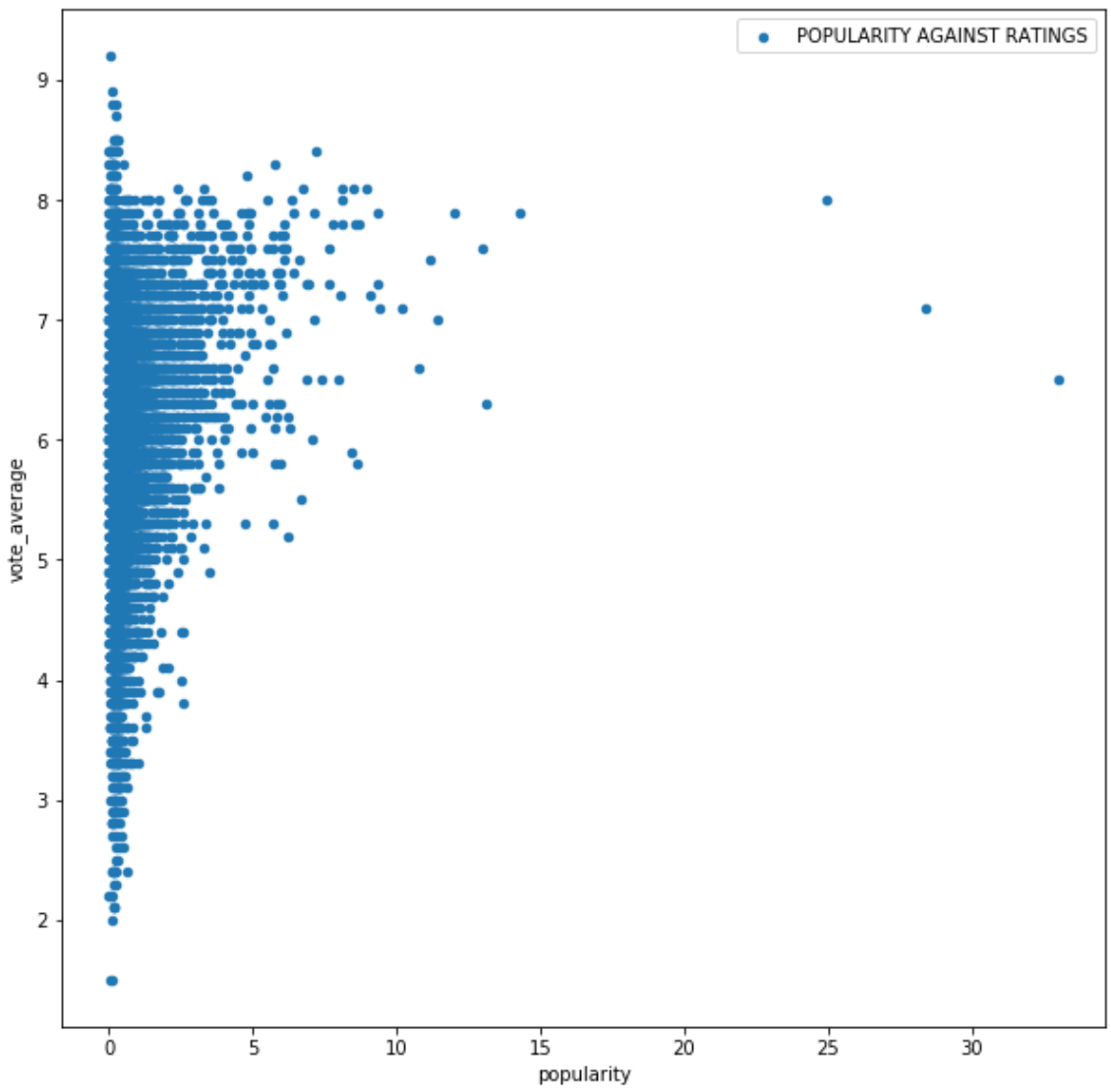
In [137]:
```
df_top10.plot(x='original_title', y='vote_average', kind = 'bar', label='TOP 10 RATED MOVIES');
```



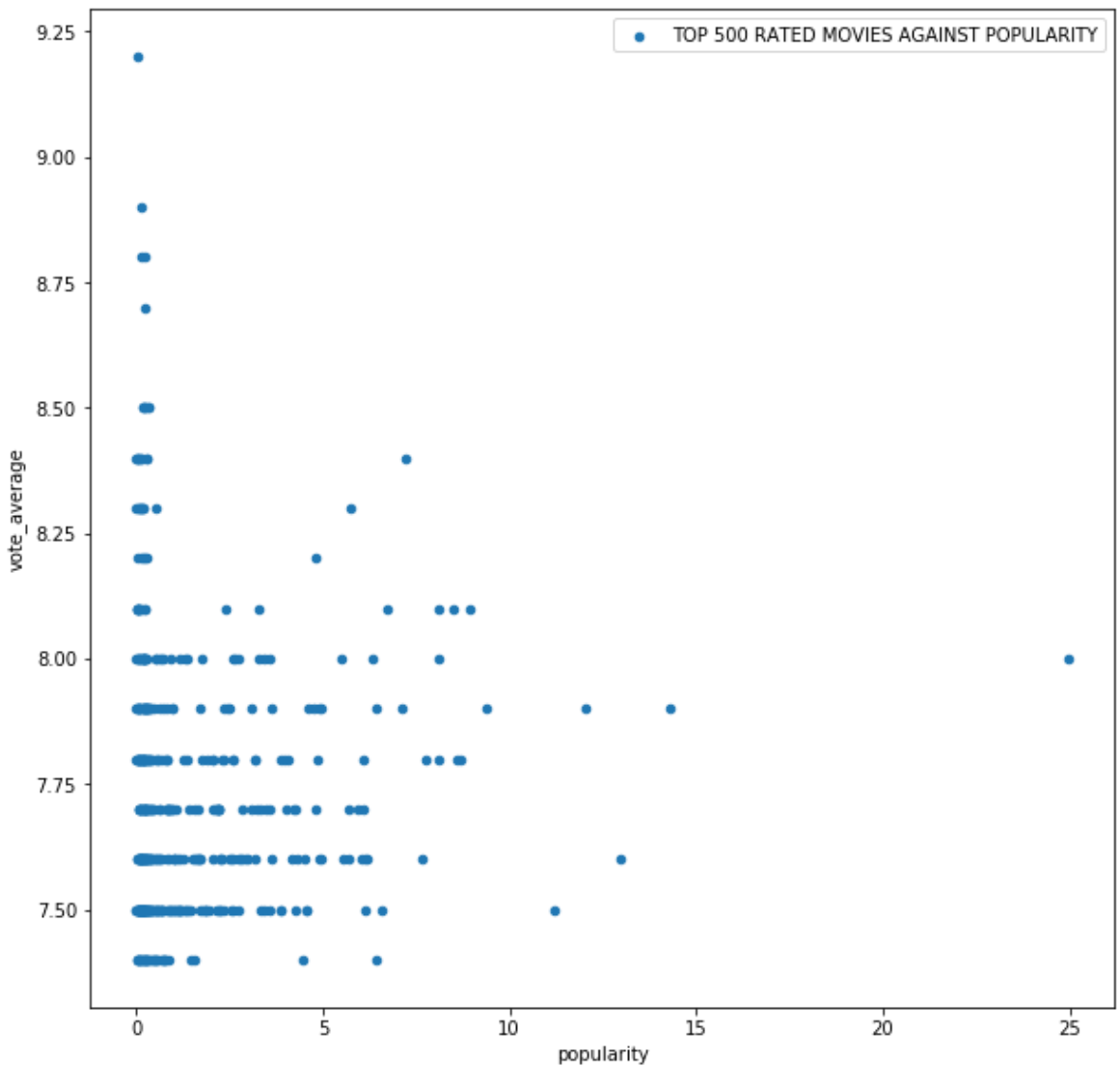
Research Question 3 - Correlation between popularity vs High-rating?

Using Scatter Plot to determine correlation

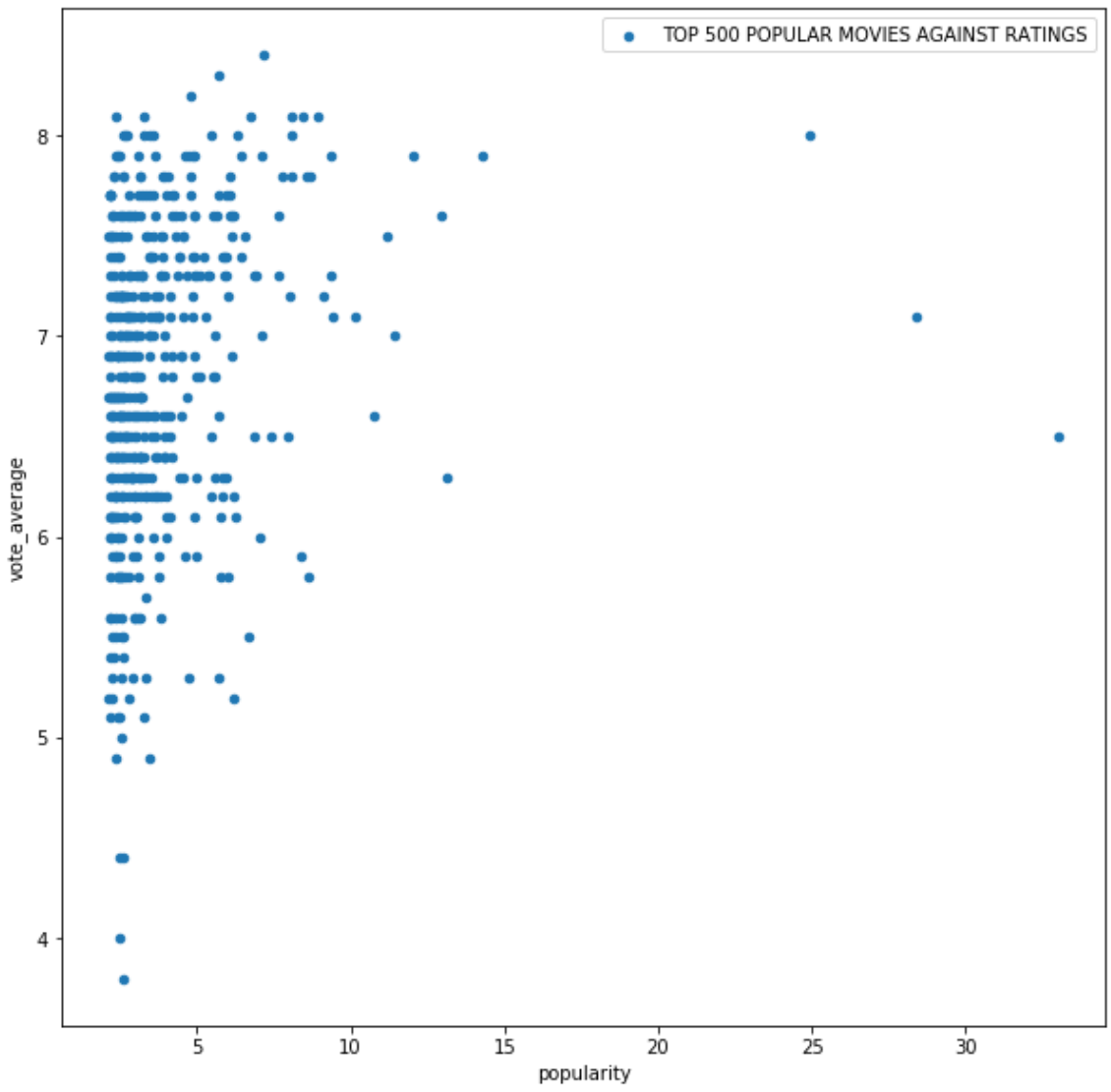
In [138]:
```
#General ScatterPlot
df_movies.plot.scatter(x='popularity',y='vote_average', figsize=(10,10), label='POPULARITY AGAINST RATINGS');
```



In [139]:
```
#Using top 500 rated  movies

df_poprated1 = df_movies.sort_values(by=['vote_average'], ascending=[False]).head(500)

df_poprated1.plot.scatter(x='popularity',y='vote_average', figsize=(10,10), label='TOP 500 RATED MOVIES AGAINST POPULA
```



In [140]:
```
#Using top 500 rated  movies

df_poprated1 = df_movies.sort_values(by=['popularity'], ascending=[False]).head(500)

df_poprated1.plot.scatter(x='popularity',y='vote_average', figsize=(10,10), label='TOP 500 POPULAR MOVIES AGAINST RATI
```



In [ ]:
```

```

Profit Calculations

In [141]:
```
#Calculating Profits

df_diff = df_movies['revenue_adj'] - df_movies['budget_adj']

df_diff.colums = ["Profit"]

df_det = df_movies[["original_title", "production_companies","director", "budget_adj", "revenue_adj" ]]

df_profit = pd.concat([df_det, df_diff], axis=1,  join='inner')

df_profit.columns = ['title', 'Company', 'director', 'budget', 'revenue', 'profit']

df_profit.head()
```

Out[141]:

	title	Company	director	budget	revenue	profit
0	Jurassic World	Universal Studios|Amblin Entertainment|Legenda...	Colin Trevorrow	1.379999e+08	1.392446e+09	1.254446e+09
1	Mad Max: Fury Road	Village Roadshow Pictures|Kennedy Miller Produ...	George Miller	1.379999e+08	3.481613e+08	2.101614e+08
2	Insurgent	Summit Entertainment|Mandeville Films|Red Wago...	Robert Schwentke	1.012000e+08	2.716190e+08	1.704191e+08
3	Star Wars: The Force Awakens	Lucasfilm|Truenorth Productions|Bad Robot	J.J. Abrams	1.839999e+08	1.902723e+09	1.718723e+09
4	Furious 7	Universal Pictures|Original Film|Media Rights ...	James Wan	1.747999e+08	1.385749e+09	1.210949e+09

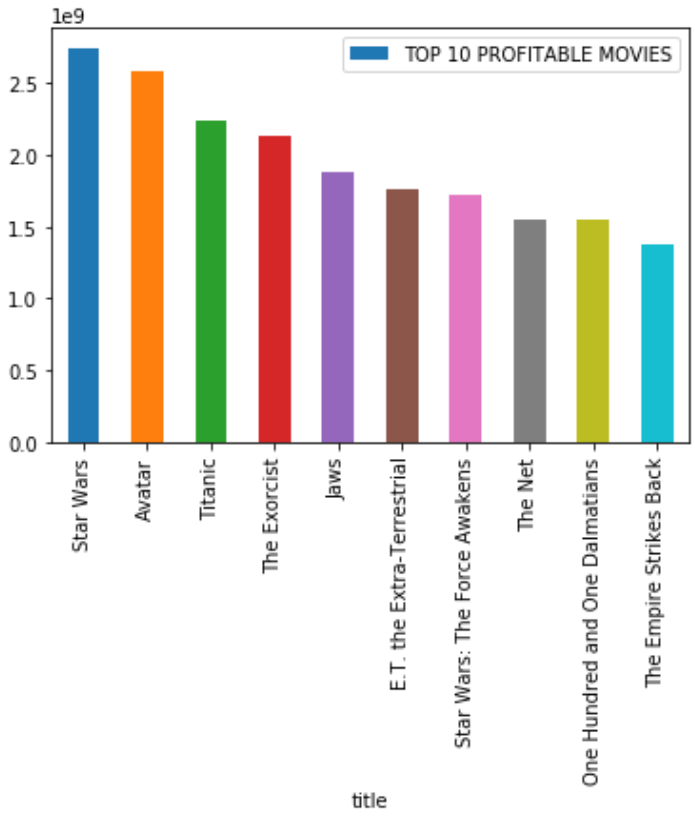
Research Question 4 - What are the most profitable movies?

In [142]:
```
df_profitmovies = df_profit.sort_values(by=['profit'], ascending=[False]).head(10)

df_profitmovies.head(10)
```

Out[142]:

	title	Company	director	budget	revenue	profit
1329	Star Wars	Lucasfilm|Twentieth Century Fox Film Corporation	George Lucas	3.957559e+07	2.789712e+09	2.750137e+09
1386	Avatar	Ingenious Film Partners|Twentieth Century Fox ...	James Cameron	2.408869e+08	2.827124e+09	2.586237e+09
5231	Titanic	Paramount Pictures|Twentieth Century Fox Film ...	James Cameron	2.716921e+08	2.506406e+09	2.234714e+09
10594	The Exorcist	Warner Bros.|Hoya Productions	William Friedkin	3.928928e+07	2.167325e+09	2.128036e+09
9806	Jaws	Universal Pictures|Zanuck/Brown Productions	Steven Spielberg	2.836275e+07	1.907006e+09	1.878643e+09
8889	E.T. the Extra-Terrestrial	Universal Pictures|Amblin Entertainment	Steven Spielberg	2.372625e+07	1.791694e+09	1.767968e+09
3	Star Wars: The Force Awakens	Lucasfilm|Truenorth Productions|Bad Robot	J.J. Abrams	1.839999e+08	1.902723e+09	1.718723e+09
8094	The Net	Columbia Pictures	Irwin Winkler	3.148127e+07	1.583050e+09	1.551568e+09
10110	One Hundred and One Dalmatians	Walt Disney Productions	Clyde Geronimi|Hamilton Luske|Wolfgang Reitherman	2.917944e+07	1.574815e+09	1.545635e+09
7309	The Empire Strikes Back	Lucasfilm|Twentieth Century Fox Film Corporation	Irvin Kershner	4.762866e+07	1.424626e+09	1.376998e+09

In [143]:
```
df_profitmovies.plot(x='title', y='profit', kind = 'bar', label='TOP 10 PROFITABLE MOVIES');
```



Research Question 5 - What Directors produce most movies?

In [144]:
```
#No of movies per director

df_movies['director'].value_counts()
```

Out[144]:
```
Woody Allen                                              45
no_dets                                                  40
Clint Eastwood                                           34
Steven Spielberg                                         29
Martin Scorsese                                          29
Ridley Scott                                             23
Ron Howard                                               22
Steven Soderbergh                                        22
Joel Schumacher                                          21
Brian De Palma                                           20
Barry Levinson                                           19
Tim Burton                                               19
Wes Craven                                               19
David Cronenberg                                         18
John Carpenter                                           18
Mike Nichols                                             18
Rob Reiner                                               18
Oliver Stone                                             17
Walter Hill                                              17
Sidney Lumet                                             17
Spike Lee                                                17
Stephen Frears                                           17
Peter Hyams                                              17
Francis Ford Coppola                                     17
Robert Zemeckis                                          17
Norman Jewison                                           17
Renny Harlin                                             17
Tyler Perry                                              17
Blake Edwards                                            16
John Landis                                              16
                                                         ..
Dominic Harari|Teresa Pelegri                             1
Tom DeCerchio                                             1
Robbie Pickering                                          1
YÄ±lmaz ErdoÄŸan                                          1
K.C. Bascombe                                             1
Gavin Wiesen                                              1
Graeme Clifford                                           1
Dan Gilroy                                                1
Cathy Malkasian|Jeff McGrath                              1
Jean-Paul Lilienfeld                                      1
Michael Dudok de Wit                                      1
James Clavell                                             1
Troy Beyer                                                1
Charles Winkler                                           1
NicolÃ¡s Goldbart                                         1
Craig Johnson                                             1
Sajid Khan                                                1
Michael Schroeder                                         1
Kat Candler                                               1
Michael Curtiz                                            1
Jonathan Judge                                            1
William A. Graham                                         1
FranÃ§ois Alaux|HervÃ© de CrÃ©cy|Ludovic Houplain          1
Demian Lichtenstein                                       1
Bonner Bellew                                             1
Didier Bourdon                                            1
John Lounsbery|Wolfgang Reitherman|Art Stevens            1
George Mihalka                                            1
Tom Kalin                                                 1
Y.K. Kim|Woo-sang Park                                    1
Name: director, Length: 5065, dtype: int64
```

Research Question 6 - What Production Companies produce most movies?

In [145]:
```
df_movies['production_companies'].value_counts(sort = 'ascending')
```

Out[145]:
```
no_dets                                                                                                          10
25
Paramount Pictures                                                                                                1
56
Universal Pictures                                                                                                1
33
Warner Bros.
84
Walt Disney Pictures
76
Columbia Pictures
72
Metro-Goldwyn-Mayer (MGM)
72
New Line Cinema
61
Touchstone Pictures
51
20th Century Fox
50
Twentieth Century Fox Film Corporation
49
TriStar Pictures
45
Orion Pictures
42
Miramax Films
32
Columbia Pictures Corporation
31
DreamWorks Animation
31
Pixar Animation Studios
30
Walt Disney Productions
29
Dimension Films
28
United Artists
23
Imagine Entertainment|Universal Pictures
22
Lions Gate Films
21
The Asylum
21
Marvel Studios
20
Walt Disney Pictures|Pixar Animation Studios
17
New World Pictures
17
American International Pictures (AIP)
14
Disney Channel
14
Hammer Film Productions
13
Walt Disney Pictures|Walt Disney Animation Studios
12

...
Laughlin Park Pictures|IFC Productions
1
3 Arts Entertainment|Irwin Entertainment
1
The Made Bed Productions|Nomadic Independence Pictures|Jagjaguwar
1
NGN Productions|Caliber Media Company|Hangar 14 Films
1
VH1 Rock Docs
1
United Artists|Furst Films|SLS Video Productions|Cinerenta Feature Films|Cinegreen
1
David Foster Productions|Turman-Foster Company|Metro-Goldwyn-Mayer (MGM)
1
Seven Arts Productions|Metro-Goldwyn-Mayer (MGM)
1
Sepia Films
1
Columbia Pictures Corporation|Renaissance Pictures|Embassy Film Associates
1
Channel Four Films|Pro-ject Filmproduktion|Argos Films|Westdeutscher Rundfunk (WDR)|Road Movies Filmproduktion
1
Columbia Pictures|Hemdale Film|Robert Stigwood Organization (RSO)
1
Channel Four Films|JVC Entertainment|Le Studio Canal+|Locus Solus Entertainment|Pandora Cinema
1
Screen Gems, Inc.|20th Century Fox Home Entertainment
1
```

```
Bad Hat Harry Productions|Jeff Rice Films|Casadelic Pictures|Guerin-Adler-Scott Pictures
1
Cinema Center Films|Waterbury Films
1
BBC Wales
1
Twentieth Century Fox Film Corporation|Dino de Laurentiis Cinematografica|International Classics
1
Onset Films|Cliffbrook Films
1
Maguire Entertainment|Screen Gems|Sony Pictures Home Entertainment
1
Brookwell-McNamara Entertainment|Pantelion Film|Televisa|Circle of Confusion
1
Castel Film Romania|Steamroller Productions
1
FilmColony|Unique New York Productions
1
Legendary Pictures|GK Films|Thunder Road Pictures|Warner Bros.
1
Twentieth Century Fox Film Corporation|Regency Enterprises
1
EuropaCorp|Mandarin Films
1
Likely Story|Random Films|Focus Features
1
20th Century Fox|Davis Entertainment
1
Fox Searchlight Pictures|Cowboy Films|DNA Films|Scottish Screen|UK Film Council
1
Cue the Dog Productions|A+E Studios|Fries Film Company
1
Name: production_companies, Length: 7444, dtype: int64
```

Research Question 7 - Movie release by year?

In [146]:
```
df_movies['release_year'].value_counts()
```

Out[146]:
```
2014    699
2013    658
2015    628
2012    586
2011    540
2009    531
2008    496
2010    488
2007    436
2006    408
2005    364
2004    307
2003    281
2002    266
2001    242
2000    227
1999    224
1998    210
1996    204
1997    192
1994    184
1993    178
1995    175
1988    145
1989    137
1992    133
1991    133
1990    132
1987    125
1986    121
1985    109
1984    105
1981     82
1982     81
1983     80
1980     78
1978     65
1979     57
1977     57
1971     55
1973     55
1974     47
1976     47
1966     46
1975     44
1964     42
1970     41
1967     40
1972     40
1968     39
1965     35
1963     34
1962     32
1960     32
1969     31
1961     31
Name: release_year, dtype: int64
```

Research Question 8 - Profitable Production Companies?

In [147]:
```
df_profitmovies = df_profit.sort_values(by=['profit'], ascending=[False]).head(10)

df_profitcomp = df_profitmovies[['Company', 'profit']]
df_profitcomp.head()
```

Out[147]:

	Company	profit
1329	Lucasfilm|Twentieth Century Fox Film Corporation	2.750137e+09
1386	Ingenious Film Partners|Twentieth Century Fox ...	2.586237e+09
5231	Paramount Pictures|Twentieth Century Fox Film ...	2.234714e+09
10594	Warner Bros.|Hoya Productions	2.128036e+09
9806	Universal Pictures|Zanuck/Brown Productions	1.878643e+09

In [148]:
```
df_profitcomps = df_profitcomp.groupby('Company').sum()

df_profitcomps.sort_values(by=['profit'], ascending=[False]).head(10)
```

Out[148]:

Company	profit
Lucasfilm|Twentieth Century Fox Film Corporation	4.127134e+09
Ingenious Film Partners|Twentieth Century Fox Film Corporation|Dune Entertainment|Lightstorm Entertainment	2.586237e+09
Paramount Pictures|Twentieth Century Fox Film Corporation|Lightstorm Entertainment	2.234714e+09
Warner Bros.|Hoya Productions	2.128036e+09
Universal Pictures|Zanuck/Brown Productions	1.878643e+09
Universal Pictures|Amblin Entertainment	1.767968e+09
Lucasfilm|Truenorth Productions|Bad Robot	1.718723e+09
Columbia Pictures	1.551568e+09
Walt Disney Productions	1.545635e+09

Research Question 9 - High Rated Directors Production Companies?

In [149]:
```
df_directorratings = df_movies[['director', 'vote_average']]
```

In [150]:
```
df_directorrated = df_directorratings.groupby('director').sum()

df_directorrated.sort_values(by=['vote_average'], ascending=[False]).head(10)
```

Out[150]:

director	vote_average
Woody Allen	290.0
no_dets	270.5
Clint Eastwood	221.3
Martin Scorsese	201.8
Steven Spielberg	197.9
Ridley Scott	149.0
Ron Howard	140.3
Steven Soderbergh	135.8
Brian De Palma	127.1
Tim Burton	126.2

Limitations

Most Fields were missing and can greatly affect certain ranking and ratings for the analysis made.

Summary

Most popular movie is "Jurrasic World"

Most Highly rated movie is "The Story of Film: An Odyssey"

No correlation between popular movie and highly rated movies

Profitable Movie was "Star Wars" produced by Lucasfilm|Twentieth Century Fox Film Corporation.

Lucasfilm|Twentieth Century Fox Film Corporation is the most profitable production company within the period.

Paramount Pictures produced highest number of movies totalling 156.

DDirecor Woody Allen produced the most movies

Director Woody Allen can be concluded with the Director with highest aggregate movie ratings.

2014 recorded highest number of movies totalling 699 movies

Conclusions

Tip: Finally, summarize your findings and the results that have been performed in relation to the question(s) provided at the beginning of the analysis. Summarize the results accurately, and point out where additional research can be done or where additional information could be useful.

Tip: Make sure that you are clear with regards to the limitations of your exploration. You should have at least 1 limitation explained clearly.

Tip: If you haven't done any statistical tests, do not imply any statistical conclusions. And make sure you avoid implying causation from correlation!

Tip: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

Submitting your Project

Tip: Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Tip: Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Tip: Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

In [151]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

Out[151]: 0

In [ ]: