Investigate_a_Dataset

June 7, 2022

1 Project: Investigate a Dataset - Tmdb-Movies Dataset

1.1 Table of Contents

Introduction

1.1.1 Dataset Description

: This is a dataset containing 10,000 movies, collated from 1960 - 2015. Collected from The Movie Database, it contains 21 columns . They are, * the ID column - containing unique identifiers for each individual movie entry, * Imdb Id - containing identifiers for each movie's result on Imdb, * popularity - of each movie given a numerical value, * budget - the budget for each movie, * revenue - the revenue for each movie upon release, * original_title - the title of each individual movie, * cast - the main cast of each movie, * homepage - the link to each movie's release website, * director - the director of each movie, * tagline - the tagline or slogan given to each movie on release, * keywords - the combination of keywords found in the movie, * overview - an overview of the movie's description in string literal, * runtime - the entire length of each movie in minutes, * genres - the genre or genres under which each movie was released, * production_companies - the company behind each movie, * release_date - the date each movie was released, * vote_count, * vote_average, * release_year- the year in which each movie was released, * budget_adj - the budget of each movie adjusted for inflation in 2010 dollars, * revenue_adj - the revenue of each movie adjusted for inflation in 2010 dollars

1.1.2 Question(s) for Analysis

Which Genres are the most popular from year to year? What kind of properties are associated with movies that have high revenue?

In [1]:
```
# We import modules for use
import pandas as pd
import numpy as np
import ast
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [1]:
```
# Upgrade pandas to use dataframe.explode() function.
!pip install --upgrade pandas==0.25.0
```

```
Collecting pandas==0.25.0
  Downloading https://files.pythonhosted.org/packages/1d/9a/7eb9952f4b4d73fbd75ad1d5d6112f407e69
    100% || 10.5MB 3.2MB/s eta 0:00:01  4% |                              | 450kB 8.6MB/s eta 0:
Collecting numpy>=1.13.3 (from pandas==0.25.0)
  Downloading https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d9473
    100% || 13.4MB 2.8MB/s eta 0:00:01   12% |                            | 1.6MB 24.3MB/s eta 0
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in /opt/conda/lib/python
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /opt/conda/lib/python3.6/site-p
Requirement already satisfied, skipping upgrade: six>=1.5 in /opt/conda/lib/python3.6/site-packa
tensorflow 1.3.0 requires tensorflow-tensorboard<0.2.0,>=0.1.0, which is not installed.
Installing collected packages: numpy, pandas
  Found existing installation: numpy 1.12.1
    Uninstalling numpy-1.12.1:
      Successfully uninstalled numpy-1.12.1
  Found existing installation: pandas 0.23.3
    Uninstalling pandas-0.23.3:
      Successfully uninstalled pandas-0.23.3
Successfully installed numpy-1.19.5 pandas-0.25.0
```

Data Wrangling

1.1.3 General Properties

Here, we need to read in our data from a csv file containing the table for analysis. Our file is separated by a comma delimiter, so we do not need to specify a separator type. Afterwards, we inspect the data that has been read in by getting some info, some summary statistics and determining the shape and size of our data. This information will be useful in cleaning our data.

In [2]:
```
# 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 = pd.read_csv('tmdb-movies.csv')
df.head()
```

Out[2]:
```
       id    imdb_id  popularity     budget     revenue  \
0  135397  tt0369610   32.985763  150000000  1513528810
1   76341  tt1392190   28.419936  150000000   378436354
2  262500  tt2908446   13.112507  110000000   295238201
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360
```

```
             original_title  \
0            Jurassic World
1          Mad Max: Fury Road
2                 Insurgent
3  Star Wars: The Force Awakens
4                 Furious 7

                                                 cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...

                                           homepage          director  \
0                     http://www.jurassicworld.com/   Colin Trevorrow
1                       http://www.madmaxmovie.com/     George Miller
2     http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod...       J.J. Abrams
4                          http://www.furious7.com/         James Wan

                          tagline  ...  \
0              The park is open.  ...
1             What a Lovely Day.  ...
2        One Choice Can Destroy You  ...
3  Every generation has a story.  ...
4             Vengeance Hits Home  ...

                                           overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137

                                      genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947
```

```
   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

In [3]:
```python
# Check through to determine how many empty values there are in the dataset
df.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
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

In [4]:
```python
# Determine number of rows and columns
df.shape
```

Out[4]: (10866, 21)

In [5]:
```python
# Summary statistics
df.describe()
```

```
Out[5]:                   id    popularity        budget       revenue       runtime  \
        count   10866.000000  10866.000000  1.086600e+04  1.086600e+04  10866.000000
        mean    66064.177434      0.646441  1.462570e+07  3.982332e+07    102.070863
        std     92130.136561      1.000185  3.091321e+07  1.170035e+08     31.381405
        min         5.000000      0.000065  0.000000e+00  0.000000e+00      0.000000
        25%     10596.250000      0.207583  0.000000e+00  0.000000e+00     90.000000
        50%     20669.000000      0.383856  0.000000e+00  0.000000e+00     99.000000
        75%     75610.000000      0.713817  1.500000e+07  2.400000e+07    111.000000
        max    417859.000000     32.985763  4.250000e+08  2.781506e+09    900.000000

                 vote_count  vote_average  release_year    budget_adj   revenue_adj
        count  10866.000000  10866.000000  10866.000000  1.086600e+04  1.086600e+04
        mean     217.389748      5.974922   2001.322658  1.755104e+07  5.136436e+07
        std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
        min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
        25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
        50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
        75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
        max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

To begin our data cleaning process, we first take a look at how many empty rows there are in our data, then the number of duplicates, and finally the number of unique values.

```
In [6]: # Return sum of null values in dataset
        df.isna().sum()
```

```
Out[6]: id                         0
        imdb_id                   10
        popularity                 0
        budget                     0
        revenue                    0
        original_title             0
        cast                      76
        homepage                7930
        director                  44
        tagline                 2824
        keywords                1493
        overview                   4
        runtime                    0
        genres                    23
        production_companies    1030
        release_date               0
        vote_count                 0
        vote_average               0
        release_year               0
        budget_adj                 0
        revenue_adj                0
        dtype: int64
```

In [7]:
```
# Sum of duplicates
df.duplicated().sum()
```

Out[7]: 1

In [8]:
```
df.nunique()
```

Out[8]:
```
id                      10865
imdb_id                 10855
popularity              10814
budget                    557
revenue                  4702
original_title          10571
cast                    10719
homepage                 2896
director                 5067
tagline                  7997
keywords                 8804
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj               2614
revenue_adj              4840
dtype: int64
```

1.1.4 Data Cleaning

After determining the columns useful to our analysis, those that are not integral to our research questions are dropped. Afterwards, missing values and duplicates are dropped from the rows.

1.1.5 Drop Extraneous Columns

In [9]:
```
# Dropping columns not to be used for analysis
df.drop(['cast', 'homepage', 'imdb_id', 'keywords', 'overview', 'tagline', 'vote_count',
```

In [10]:
```
# Checking to see if columns have been dropped
df.head()
```

Out[10]:
```
       id  popularity          director  runtime  \
0  135397   32.985763   Colin Trevorrow      124
1   76341   28.419936     George Miller      120
2  262500   13.112507  Robert Schwentke      119
3  140607   11.173104      J.J. Abrams       136
```

```
4  168259    9.335014         James Wan      137

                                           genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                               production_companies  release_year  \
0  Universal Studios|Amblin Entertainment|Legenda...          2015
1  Village Roadshow Pictures|Kennedy Miller Produ...          2015
2  Summit Entertainment|Mandeville Films|Red Wago...          2015
3          Lucasfilm|Truenorth Productions|Bad Robot          2015
4  Universal Pictures|Original Film|Media Rights ...          2015

     budget_adj   revenue_adj
0  1.379999e+08  1.392446e+09
1  1.379999e+08  3.481613e+08
2  1.012000e+08  2.716190e+08
3  1.839999e+08  1.902723e+09
4  1.747999e+08  1.385749e+09
```

Then we have to further remove columns that contain null values and that can't be replaced with mean because they're strings

In [11]:
```
# Drop empty rows
df.dropna(inplace = True)
```

In [12]:
```
# Check to see if empty rows have been trimmed
df.shape
```

Out[12]:
```
(9807, 9)
```

In [13]:
```
# Drop duplicate values in dataset
df.drop_duplicates(inplace = True)
```

In [14]:
```
df.head()
```

Out[14]:
```
       id  popularity          director  runtime  \
0  135397   32.985763   Colin Trevorrow      124
1   76341   28.419936     George Miller      120
2  262500   13.112507  Robert Schwentke      119
3  140607   11.173104       J.J. Abrams      136
4  168259    9.335014         James Wan      137

                                           genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
```

```
2        Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                                        production_companies  release_year  \
0  Universal Studios|Amblin Entertainment|Legenda...          2015
1  Village Roadshow Pictures|Kennedy Miller Produ...          2015
2  Summit Entertainment|Mandeville Films|Red Wago...          2015
3          Lucasfilm|Truenorth Productions|Bad Robot          2015
4  Universal Pictures|Original Film|Media Rights ...          2015

     budget_adj   revenue_adj
0  1.379999e+08  1.392446e+09
1  1.379999e+08  3.481613e+08
2  1.012000e+08  2.716190e+08
3  1.839999e+08  1.902723e+09
4  1.747999e+08  1.385749e+09
```

Our Genres column contains multiple genres for each movie, we can then split the genres column, and convert the column from a string to a list containing the genres pertaining to each movie, in order for analysis to be easier.

In [15]:
```python
# columns to split by "/"
split_columns = ['genres']

# apply split function to each column of each dataframe copy
for c in split_columns:
    df[c] = df[c].apply(lambda x: x.split("|"))
```

In [16]:
```python
df.head()
```

Out[16]:
```
       id  popularity         director  runtime  \
0  135397   32.985763  Colin Trevorrow      124
1   76341   28.419936    George Miller      120
2  262500   13.112507  Robert Schwentke     119
3  140607   11.173104      J.J. Abrams      136
4  168259    9.335014        James Wan      137

                                        genres  \
0  [Action, Adventure, Science Fiction, Thriller]
1  [Action, Adventure, Science Fiction, Thriller]
2          [Adventure, Science Fiction, Thriller]
3   [Action, Adventure, Science Fiction, Fantasy]
4                      [Action, Crime, Thriller]

                                        production_companies  release_year  \
0  Universal Studios|Amblin Entertainment|Legenda...          2015
1  Village Roadshow Pictures|Kennedy Miller Produ...          2015
2  Summit Entertainment|Mandeville Films|Red Wago...          2015
```

```
3          Lucasfilm|Truenorth Productions|Bad Robot          2015
4  Universal Pictures|Original Film|Media Rights ...          2015

     budget_adj   revenue_adj
0  1.379999e+08  1.392446e+09
1  1.379999e+08  3.481613e+08
2  1.012000e+08  2.716190e+08
3  1.839999e+08  1.902723e+09
4  1.747999e+08  1.385749e+09
```

Now that our data has been properly cleaned, and unused columns have been dropped, missing values and duplicate values have been removed, it's time to anwer the research questions.

Exploratory Data Analysis

1.1.6 Research Question 1 (Which genres are most popular from year to year?)

In [17]:
```
# To begin exploring our data, we groupby release year first
popularity_mean = df.groupby(['release_year']).popularity.mean()
popularity_mean.head()
```

Out[17]:
```
release_year
1960    0.458932
1961    0.430438
1962    0.465245
1963    0.502706
1964    0.421091
Name: popularity, dtype: float64
```

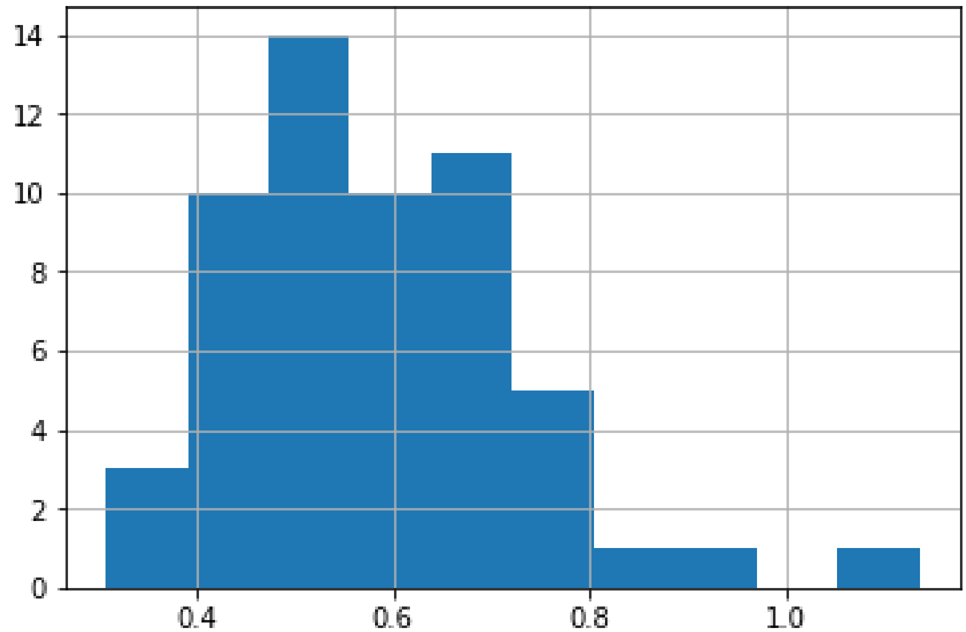
In [18]:
```
# We then look at the summary statistics
popularity_mean.describe()
```

Out[18]:
```
count    56.000000
mean      0.588379
std       0.147449
min       0.308457
25%       0.486578
50%       0.572578
75%       0.674149
max       1.135148
Name: popularity, dtype: float64
```

From our summary statistics, we can see that the max mean popularity is quite high compared to other values, this could be due to improperly inputted values. To continue our analysis, we'll look at a visualization of our data.

In [19]:
```
# Popularity visualized
popularity_mean.hist()
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc076d40128>
```



We can see here that the average or mean popularity of all genres is skewed to the right.

```
In [20]: from collections import Counter

         # Because the genres for every movie are basically sets
         # We can use a "flattened list" approach for this question
         flattened_genres = [elem for sublist in df['genres'] for elem in sublist]
         fav_genre = Counter(flattened_genres).most_common()

         # retrieving top 5 genres
         fav_genre[:5]
```

```
Out[20]: [('Drama', 4369),
          ('Comedy', 3438),
          ('Thriller', 2747),
          ('Action', 2235),
          ('Romance', 1570)]
```

Now we can retrieve the top 5 genres over the years from 1960 -2015, and we can see that Drama comes out as the most popular movie over this entire period. This means that over this period of years, drama had the most consistently high popularity.

1.1.7 Research Question 2 (What kind of properties are associated with movies that have high revenues?)

In [21]:
```
#Return a query of all properties
df. describe
```

Out[21]:
```
<bound method NDFrame.describe of            id  popularity           director  runtim
0      135397   32.985763     Colin Trevorrow      124
1       76341   28.419936      George Miller      120
2      262500   13.112507    Robert Schwentke      119
3      140607   11.173104         J.J. Abrams      136
4      168259    9.335014           James Wan      137
...       ...         ...                ...      ...
10861      21    0.080598         Bruce Brown       95
10862   20379    0.065543  John Frankenheimer      176
10863   39768    0.065141      Eldar Ryazanov       94
10864   21449    0.064317         Woody Allen       80
10865   22293    0.035919     Harold P. Warren       74

                                          genres  \
0      [Action, Adventure, Science Fiction, Thriller]
1      [Action, Adventure, Science Fiction, Thriller]
2              [Adventure, Science Fiction, Thriller]
3       [Action, Adventure, Science Fiction, Fantasy]
4                           [Action, Crime, Thriller]
...                                              ...
10861                                  [Documentary]
10862                     [Action, Adventure, Drama]
10863                              [Mystery, Comedy]
10864                               [Action, Comedy]
10865                                       [Horror]

                                    production_companies  release_year  \
0      Universal Studios|Amblin Entertainment|Legenda...          2015
1      Village Roadshow Pictures|Kennedy Miller Produ...          2015
2      Summit Entertainment|Mandeville Films|Red Wago...          2015
3              Lucasfilm|Truenorth Productions|Bad Robot          2015
4      Universal Pictures|Original Film|Media Rights ...          2015
...                                                  ...           ...
10861                                  Bruce Brown Films          1966
10862  Cherokee Productions|Joel Productions|Douglas ...          1966
10863                                            Mosfilm          1966
10864                             Benedict Pictures Corp.          1966
10865                                         Norm-Iris          1966

         budget_adj   revenue_adj
0      1.379999e+08  1.392446e+09
1      1.379999e+08  3.481613e+08
2      1.012000e+08  2.716190e+08
```

```
3      1.839999e+08  1.902723e+09
4      1.747999e+08  1.385749e+09
...             ...           ...
10861  0.000000e+00  0.000000e+00
10862  0.000000e+00  0.000000e+00
10863  0.000000e+00  0.000000e+00
10864  0.000000e+00  0.000000e+00
10865  1.276423e+05  0.000000e+00

[9806 rows x 9 columns]>
```

In [23]:
```
# We return a query showing only top 50% revenues
top_movies = df.query('revenue_adj > revenue_adj.mean()')

# Determine how many unique entries we have in data
top_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2166 entries, 0 to 10848
Data columns (total 9 columns):
id                      2166 non-null int64
popularity              2166 non-null float64
director                2166 non-null object
runtime                 2166 non-null int64
genres                  2166 non-null object
production_companies    2166 non-null object
release_year            2166 non-null int64
budget_adj              2166 non-null float64
revenue_adj             2166 non-null float64
dtypes: float64(3), int64(3), object(3)
memory usage: 169.2+ KB
```

Here we can just confirm how many entries out of the entire data frame fall into the higher revenue range.

In [24]:
```
top_movies.describe()
```

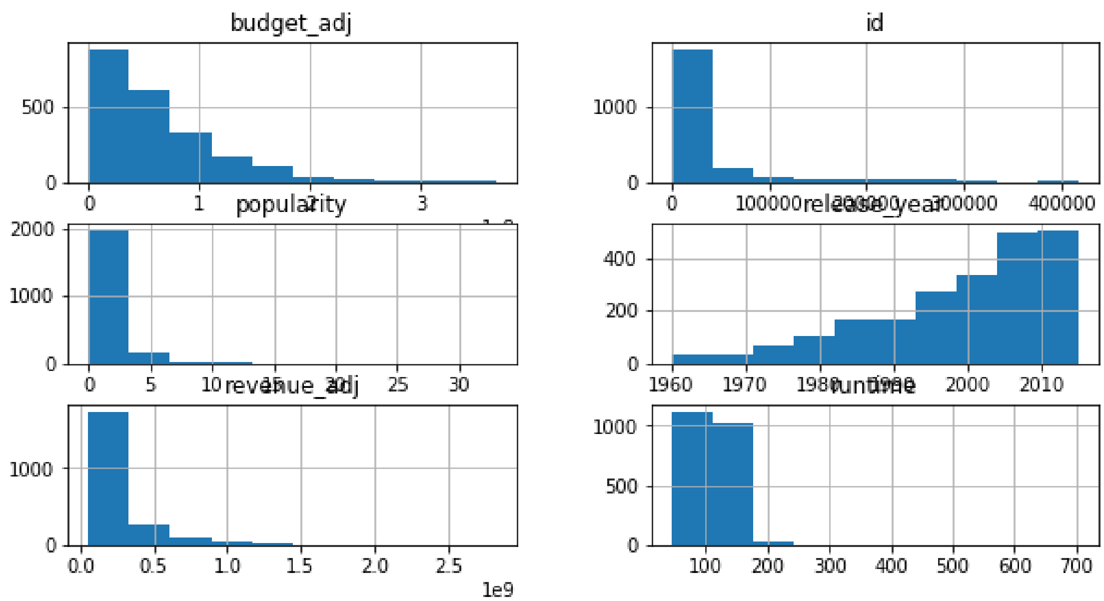
Out[24]:
```
                  id   popularity      runtime  release_year    budget_adj  \
count    2166.000000  2166.000000  2166.000000   2166.000000  2.166000e+03
mean    31583.251616     1.592239   113.243767   1999.242382  5.969112e+07
std     59323.470090     1.795896    24.243409     12.464485  5.088598e+07
min        11.000000     0.010335    44.000000   1960.000000  0.000000e+00
25%      2112.500000     0.683560    98.000000   1992.000000  2.308905e+07
50%      9881.000000     1.120763   110.000000   2002.000000  4.607896e+07
75%     21817.750000     1.862250   124.000000   2009.000000  8.374198e+07
max    417859.000000    32.985763   705.000000   2015.000000  3.683713e+08

         revenue_adj
```

```
count    2.166000e+03
mean     2.358479e+08
std      2.486913e+08
min      5.691199e+07
25%      8.974871e+07
50%      1.487309e+08
75%      2.779126e+08
max      2.827124e+09
```

We can see from this table and comparing it to the table description of all properties the variables which have a lower mean value against revenue, and those which have a higher mean value.

```
In [25]: # histogram to show variables
         top_movies.hist(figsize= (10, 5))
```

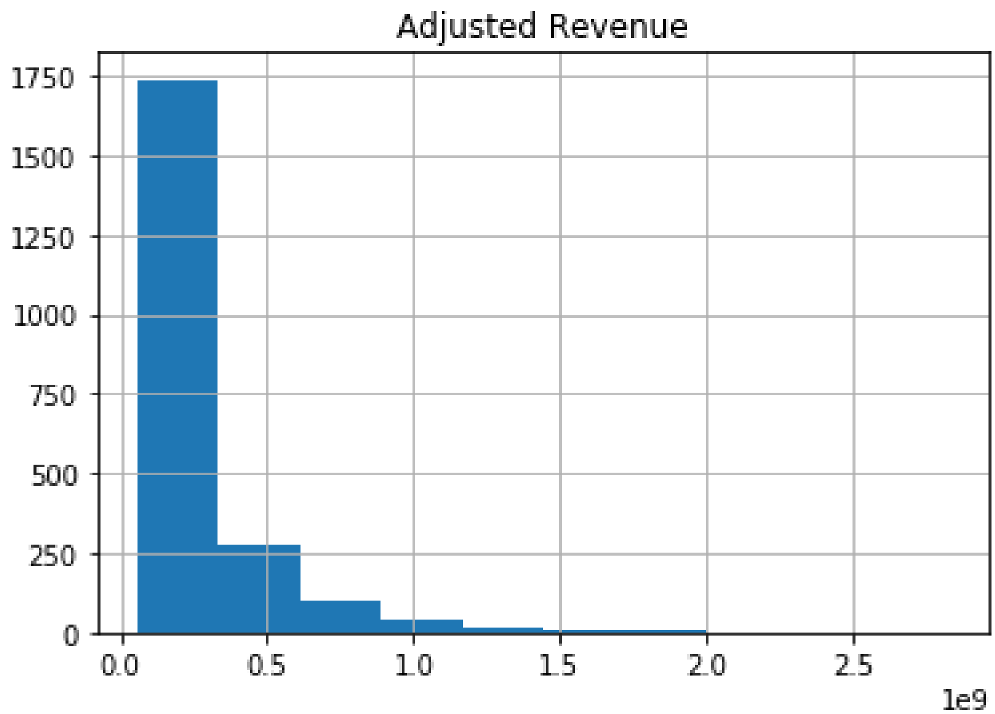
```
Out[25]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fc0766e3278>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7fc07668ee48>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc07663fe48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7fc076653588>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc07662ddd8>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7fc07662de10>]],
               dtype=object)
```



We can now take a look at our top earning movies. Then we create scatter plots to show the relationship between revenue and other variables to see if there is positive or negative correlation between the variables.

In [26]:
```
top_movies.revenue_adj.hist()
plt.title("Adjusted Revenue")
```

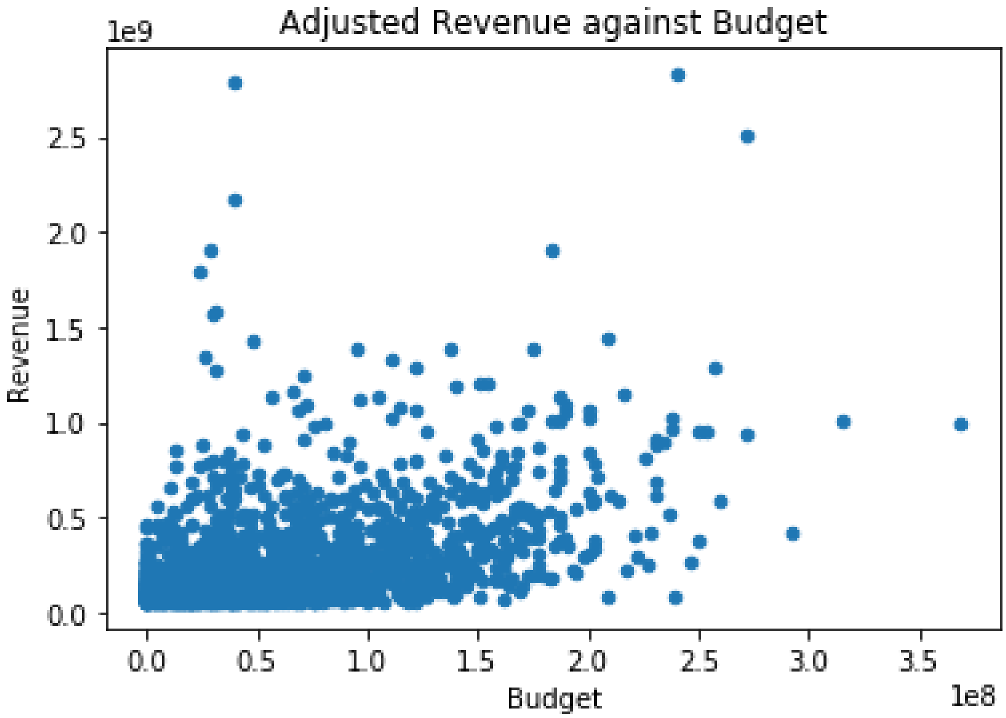
Out[26]: Text(0.5,1,'Adjusted Revenue')



Now, we create scatter plots to visualize the relationship that exists between our dependent variable, the Revenue on the y axis, and the independent variables on the x axis.

In [27]:
```
# Revenue against Budget
top_movies.plot(x = 'budget_adj', y = 'revenue_adj', kind = 'scatter')
plt.title("Adjusted Revenue against Budget")
plt.xlabel('Budget')
plt.ylabel('Revenue')
```

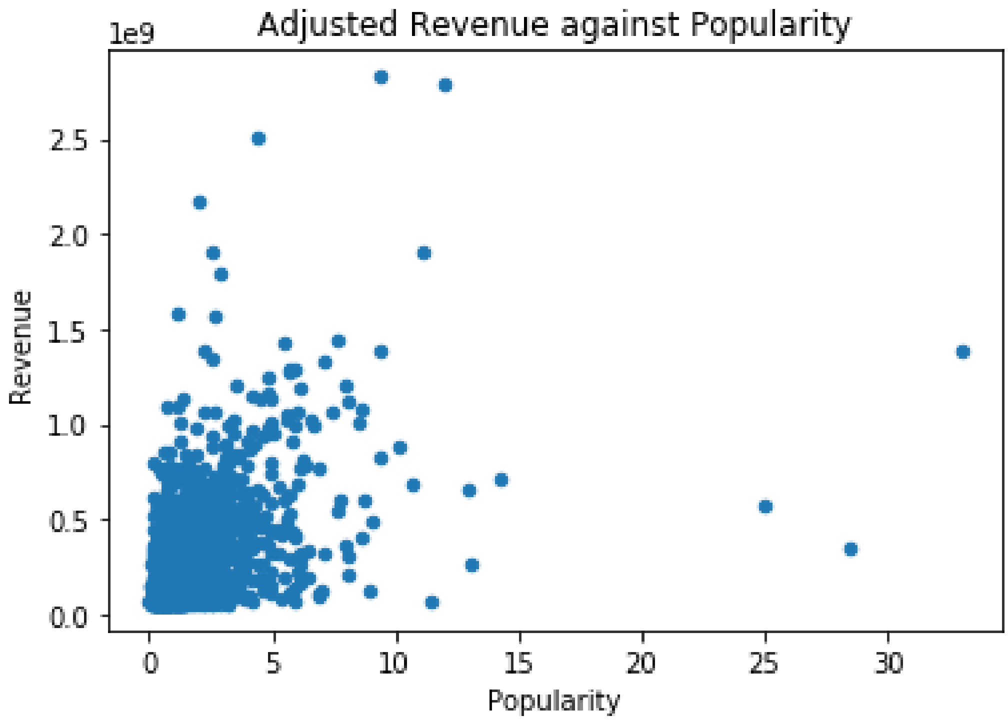
Out[27]: Text(0,0.5,'Revenue')



We can see that our adjusted revenue against budget seems to have minimal to no correlation.

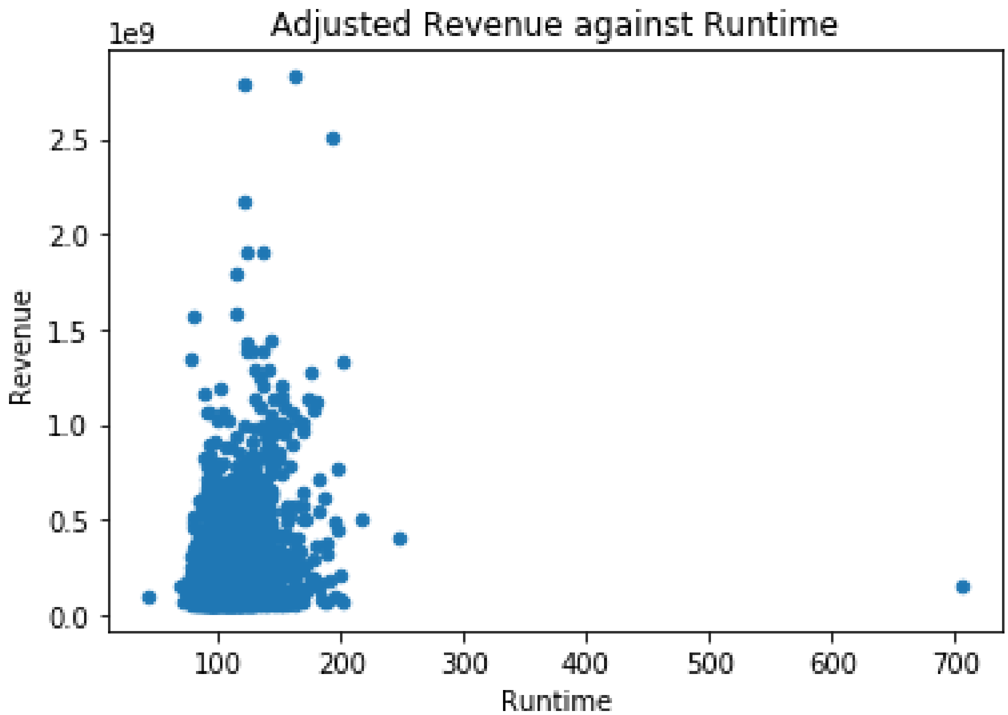
In [28]:
```
# Revenue against popularity
top_movies.plot(x = 'popularity', y = 'revenue_adj', kind = 'scatter')
plt.title("Adjusted Revenue against Popularity")
plt.xlabel('Popularity')
plt.ylabel('Revenue')
```

Out[28]:
```
Text(0,0.5,'Revenue')
```



Likewise, we can see that Adjusted revenue has little to no correlation to popularity.

```
In [29]: # Revenue against runtime
         top_movies.plot(x = 'runtime', y = 'revenue_adj', kind = 'scatter')
         plt.title("Adjusted Revenue against Runtime")
         plt.xlabel('Runtime')
         plt.ylabel('Revenue')

Out[29]: Text(0,0.5,'Revenue')
```



After comparing our adjusted revenue against the three major quantitative independent variables, we find little to no correlation among them. This leads us to the conclusion that with the data we have, we're inconclusive on determining which variables have an impact, or are associated with high revenue movies.

Conclusions

Our dataset was collected from Kaggle.com and is a compilation of movies of different genres from 1960 - 2015. The dataset contains a sizable number of missing values. To clean our data, we trimmed the dataset to only contain the necessary columns for our exploratory analysis, removed all duplicates found in our dataset and likewise dropped all null values from our dataset. In summary, in response to our first research question, we found that
In response to our second research question, we could not find any correlation, be it positive or negative between high revenue and any of the independent variables in the dataset. This might be due to the presence of outliers within said variables. Our statistical analysis showed that the high revenue generated by well-performing movies could not be traced back to any of our quantitative variables usch as the budget, the popularity etc.

1.2 Limitations

A limitation in exploration results or findings in our dataset is that, the revenue variable for all movies might not be in US dollars. This is because data was collected from user input, and can't be fully verified.

Reference: Some part of our data analysis process was sourced and derived from Kaggle.com

```
In [30]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])

Out[30]: 0
```