Project: The Movie Analysis

Table of Contents

Introduction

Dataset Description

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

The columns in the movie dataset are:

- **id:** primary key, unique identifier
- **imdb_id:** the id for each title on IMDb.
- **popularity:** A complex metric that is built from various factors, such as number of votes, number of views, release date, and number of users adding the title to their "watchlist." More information can be found on:https://developers.themoviedb.org/3/getting-started/popularity (https://developers.themoviedb.org/3/getting-started/popularity)
- **budget:** the recorded budget of the film at time of release
- **revenue:** the cumulative revenue made by the film by the dataset's release in 2015
- **original_title:** the title of the film at release
- **cast:** a list of prominent cast-members in the film
- **homepage:** contains a link to the homepage of the movie's website
- **director:** contains either the lead director, or a list of directors associated with the film.
- **tagline:** the one or two sentence tagline accompanying the film's title for promotion
- **keywords:** a list of SEO keywords for searching and indexing
- **overview:** a brief description of the movie's plot
- **runtime:** the length of the movie in minutes
- **genres:** a list of genres that the film falls under production_companies: a list of companies involved in the production of the film
- **release_date:** the day the film was released
- **vote_count:** the number of unique votes that have been submitted for the movie on TMDB
- **vote_average:** the mean score calculated from all votes
- **release_year:** the year the title was released
- **budget_adj:** the film's budget, adjusted for inflation to 2015 dollars.
- **revenue_adj:** the film's revenue, adjusted for inflation to 2015 dollars.

Questions for Analysis

In our analysis we explore and answer the following questions:

1. What are titles the most profitable movies?
2. Which genres are most popular from year to year?

In [1]:
```
# import the packages.

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

%matplotlib inline
```

1. Reading and Inspection

Import and read

Import and read the movie database and store it in variable called **movies**

In [2]:
```
# Load the data and print out a few lines

movies = pd.read_csv("tmdb-movies.csv")
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://\
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://w\

2 rows × 21 columns

Inspect the dataframe

Inspect the dataframe columns, shape, variable, types e.t.c

In [3]:

Out[3]: (10866, 21)

In [4]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

2. Data Cleaning

In respect to the research questions, i will drop some columns that are not necessary.

Drop unecessary column

In [5]:

```
# get the columns in the dataset

cols = movies.columns
```

Out[5]:

```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title
',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

I selected the following data for the statistical modeling:

- Movie title -> original_title
- Genre of the movie -> genres
- Movie duration (in minutes) -> runtime
- Release year of the film -> release_year
- Number of votes -> vote_count
- Number of average vote -> vote_average
- Movie cast -> cast
- Movie Budget -> budjet_adj
- Movie Revenue-> revenue_adj

I will be dropping the tagline, homepage, keywords, cast, director, production companies, budget, revenue and overview columns from the dataset.

In [6]:
```
# Drop the unnecessary columns

movies = movies.drop(['imdb_id', 'popularity', 'budget', 'revenue', 'homepa
```

In [7]:
```
# confirm the remaining columns after droppng the unnecessary columns.
```

Out[7]:
```
Index(['id', 'original_title', 'cast', 'director', 'runtime', 'genres',
       'release_date', 'vote_count', 'vote_average', 'release_year',
       'budget_adj', 'revenue_adj'],
      dtype='object')
```

Identify the missing values within the dataset.

In [8]:

Out[8]:
```
cast              76
director          44
genres            23
id                 0
original_title     0
runtime            0
release_date       0
vote_count         0
vote_average       0
release_year       0
budget_adj         0
revenue_adj        0
dtype: int64
```

In [9]:
```
# Check for any duplicate
```

Out[9]:
```
1
```

In [10]:
```
# drop the duplicate
movies.drop_duplicates(inplace = True)
```

Percentage of null values

We check for the percentage of the null values, to know if dropping them may affect the dataset.

In [15]:
```
# percentage of null values in the columns
```

Out[15]:
```
cast              0.699494
director          0.404970
genres            0.211689
id                0.000000
original_title    0.000000
runtime           0.000000
release_date      0.000000
vote_count        0.000000
vote_average      0.000000
release_year      0.000000
budget_adj        0.000000
revenue_adj       0.000000
dtype: float64
```

We can therefore drop the null values since they are less than 5%

Dropping null data

In [12]:
```
# Dropping null data from remaining data to keep a clean dataset.
```

In [13]:
```

```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10731 entries, 0 to 10865
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              10731 non-null  int64
 1   original_title  10731 non-null  object
 2   cast            10731 non-null  object
 3   director        10731 non-null  object
 4   runtime         10731 non-null  int64
 5   genres          10731 non-null  object
 6   release_date    10731 non-null  object
 7   vote_count      10731 non-null  int64
 8   vote_average    10731 non-null  float64
 9   release_year    10731 non-null  int64
 10  budget_adj      10731 non-null  float64
 11  revenue_adj     10731 non-null  float64
dtypes: float64(3), int64(4), object(5)
memory usage: 1.1+ MB
```

In [14]:

Out[14]:

	id	runtime	vote_count	vote_average	release_year	budget_adj
count	10731.000000	10731.000000	10731.000000	10731.000000	10731.000000	1.073100e+04
mean	65201.741869	102.468829	219.812972	5.964710	2001.259622	1.776530e+07
std	91470.508056	30.493873	578.815324	0.930283	12.820151	3.446630e+07
min	5.000000	0.000000	10.000000	1.500000	1960.000000	0.000000e+00
25%	10547.500000	90.000000	17.000000	5.400000	1995.000000	0.000000e+00
50%	20323.000000	99.000000	39.000000	6.000000	2006.000000	0.000000e+00
75%	73948.500000	112.000000	148.000000	6.600000	2011.000000	2.110885e+07
max	417859.000000	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08

Note: The minimum values for runtime, budget_adj, and revenue_adj are zero, which implies that the titles have incomplete data. We will just set them to null values to prevent them from interfering with later data visualization.

In [15]:
```
movies['runtime'] = movies['runtime'].replace(0,np.NaN)
movies['budget_adj'] = movies['budget_adj'].replace(0, np.NaN)
movies['revenue_adj'] = movies['revenue_adj'].replace(0, np.NaN)
```

In [16]:

Out[16]:

	id	runtime	vote_count	vote_average	release_year	budget_adj
count	10731.000000	10703.000000	10731.000000	10731.000000	10731.000000	5.153000e+03
mean	65201.741869	102.736896	219.812972	5.964710	2001.259622	3.699582e+07
std	91470.508056	30.079331	578.815324	0.930283	12.820151	4.198202e+07
min	5.000000	3.000000	10.000000	1.500000	1960.000000	9.210911e-01
25%	10547.500000	90.000000	17.000000	5.400000	1995.000000	8.142944e+06
50%	20323.000000	99.000000	39.000000	6.000000	2006.000000	2.287867e+07
75%	73948.500000	112.000000	148.000000	6.600000	2011.000000	5.024535e+07
max	417859.000000	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08

Now that i've trimmed and cleaned the data, then ready to move on to exploration.

Exploratory Data Analysis

> we will begin our exploratory analysis to see what information we can find to help us answer our questions.

Research Question 1 : What are titles the most profitable

movies?

Convert the unit of the budget_adj and revenue_adj column from dollar to million dollar

In [17]:
```
movies['budget_adj']=movies['budget_adj']/1000000
```

Movies title with the highest profit.

1. Create a new column called **profit** which contains the differnce between revenue_adj and budget_adj
2. Sort the dataframe using profit column as refernce
3. Extract the title and director of the top 10 profiting movies and store it to a new dataframe named **Top_10**

In [18]:
```
```

In [19]:
```
Top_10 = movies.sort_values(by='profit', ascending=False).head(10)
Top_10_movies = Top_10.loc[:, ['original_title','profit']]
```

Out[19]:

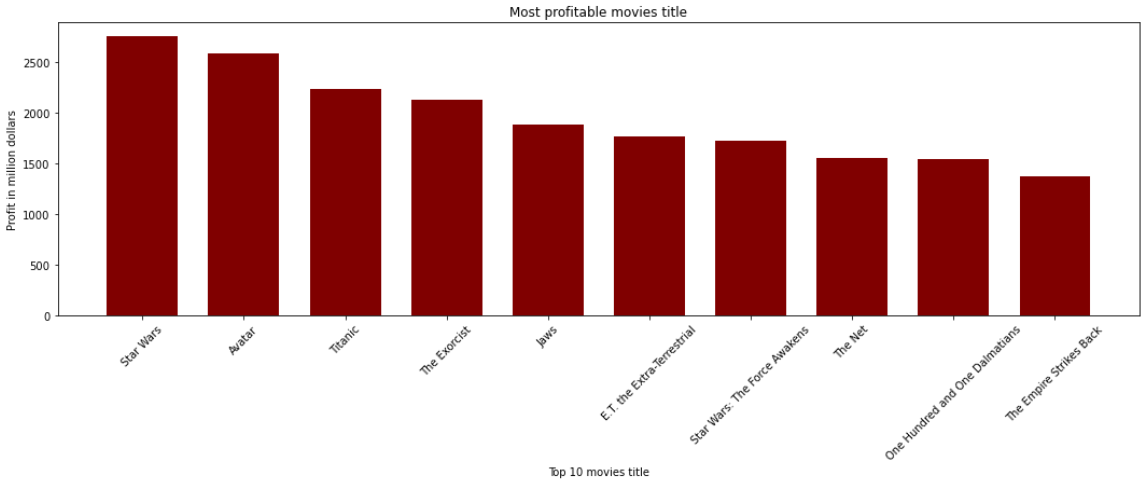
	original_title	profit
1329	Star Wars	2750.136651
1386	Avatar	2586.236848
5231	Titanic	2234.713671
10594	The Exorcist	2128.035625
9806	Jaws	1878.643094
8889	E.T. the Extra-Terrestrial	1767.968064
3	Star Wars: The Force Awakens	1718.723211
8094	The Net	1551.568265
10110	One Hundred and One Dalmatians	1545.635295
7309	The Empire Strikes Back	1376.997526

In [20]:
```
title = Top_10_movies['original_title']
values = Top_10_movies ['profit']

fig = plt.figure(figsize = (18, 5))

# creating the bar plot
plt.bar(title, values, color ='maroon', width = 0.7)

plt.xlabel("Top 10 movies title")
plt.ylabel("Profit in million dollars")
plt.title("Most profitable movies title")
plt.xticks(rotation = 45)
```



. We can clearly see that the movie titled **Star wars** earned the most profit

Research Question 2 : Which genres are most popular from year to year?

I noticed that the genre column in the dataframe have the genre of the movies seperated by a pipe (|). Out of all the genres, the first two are most significant for any of the movies.

1. Extract the genres from the genre columns. As we can see a movie might belong to many genres which are separated by | in our dataset. We ‘ll count each genre for a movie as a separate record and split them to create those records.
2. One way of doing that could be creating dummies for each possible genre, such as Sci-Fi or Drama, and having a single column for each. Creating dummies means creating 0s and 1s just like you can see in the example below:

In [21]:
```
# Get list of genres
genre_list = movies['genres'].str.split('|', expand=True)
```

In [22]:
```
genre_list = genre_list.apply(pd.Series.value_counts).index.tolist()
```

In [23]:
```
genre_list = np.array(genre_list)
```

In [24]:

Out[24]:
```
array(['Action', 'Adventure', 'Animation', 'Comedy', 'Crime',
       'Documentary', 'Drama', 'Family', 'Fantasy', 'Foreign', 'History',
       'Horror', 'Music', 'Mystery', 'Romance', 'Science Fiction',
       'TV Movie', 'Thriller', 'War', 'Western'], dtype='<U15')
```

In [25]:
```
# Get list of years and sort from oldest to newest
year_list = movies['release_year'].value_counts().index.tolist()
year_list.sort()
```

In [26]:

Out[26]:
```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
       2015])
```

Defining a function to aid visualisation

In [27]:
```
def figure_resize(f, w, h):
#     Sets a figure's width and height to (w, h)

#        Parameters:
#        f (matplotlib.pyplot.figure): plt figure to modify
#        w : desired width
#        h : desired height

    f.set_figwidth(w)
    f.set_figheight(h)

def set_color_palette(palette, n_colors, axis):
#        Sets the seaborn color palette for plotting.

#        Parameters:
#        palette (string): seaborn palette to use
#        n_colors (int): number of colors to cycle in palette
#        axis (matplotlib.pyplot.axis): axis to pass from subplots

    col = sns.color_palette(palette, n_colors)
    axis.set_prop_cycle('color', col)


def next_line_style():
#        Returns the next line style. Call as line style argument in plt.p

#        Parameters:
#        N/A

#        Returns: next line style as char ( line_styles[line_style_iterator


    global line_style_iterator
    line_style_iterator = line_style_iterator + 1 if line_style_iterator < 
    return line_styles[line_style_iterator]

line_styles = ['-', '--', '-.', ':']  # List of line styles to use for plt.
line_style_iterator = -1              # Used to iterate line styles in next_
```

Comparing mean adjusted budget for each genre

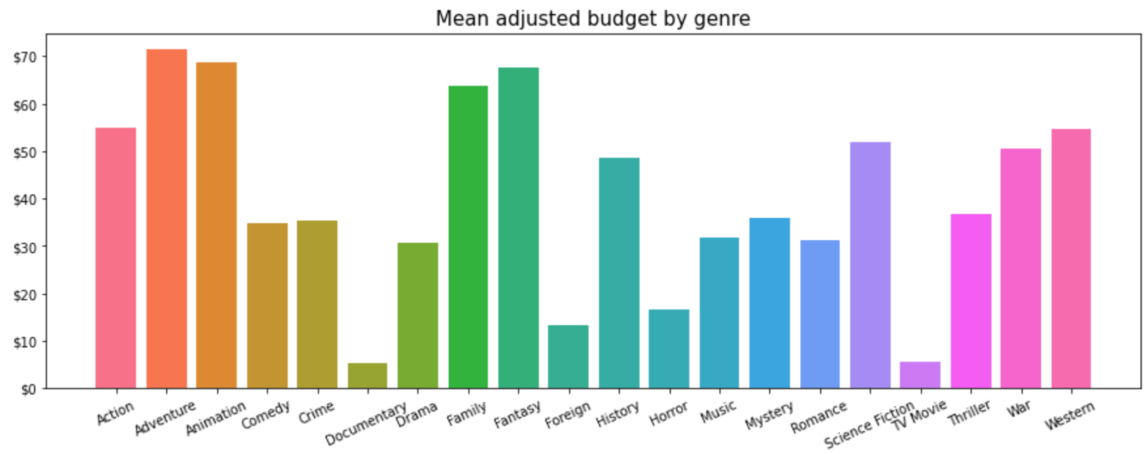
In [28]:

```
fig, ax = plt.subplots()

#formatting y-axis as dollar figures
ax.yaxis.set_major_formatter('${x:1,.0f}')


#plot formatting
figure_resize(fig, 15, 5)
set_color_palette('husl', len(genre_list), ax)
plt.title('Mean adjusted budget by genre', fontsize=15)
plt.xticks(rotation = 25)

for genre in genre_list:
    plt.bar(genre,movies.loc[movies['genres'].str.contains(genre)]['budget_a
plt.show()
```

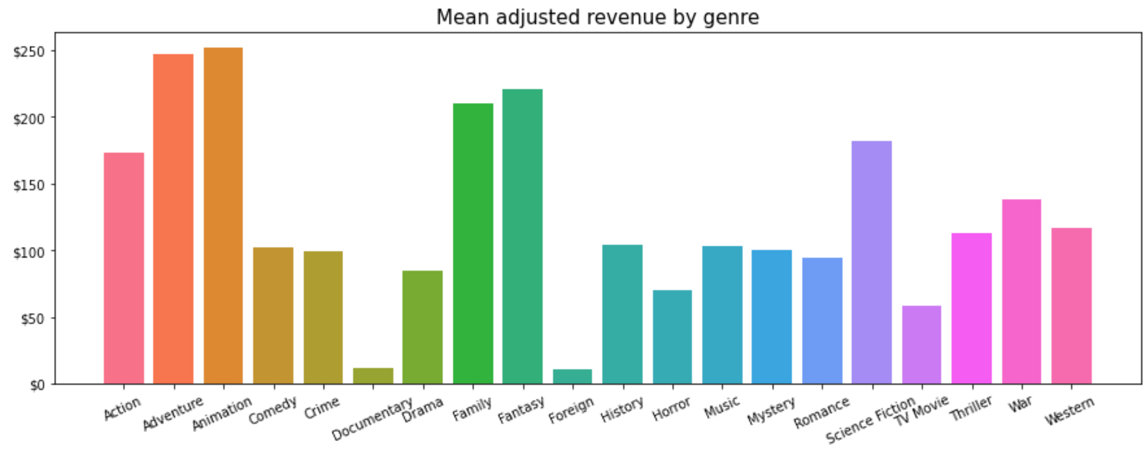


In [29]:
```
fig, ax = plt.subplots()

#formatting y-axis as dollar figures
ax.yaxis.set_major_formatter('${x:1,.0f}')


#plot formatting
figure_resize(fig, 15, 5)
set_color_palette('husl', len(genre_list), ax)
plt.title('Mean adjusted revenue by genre', fontsize=15)
plt.xticks(rotation = 25)

for genre in genre_list:
    plt.bar(genre,movies.loc[movies['genres'].str.contains(genre)]['revenue_
plt.show()
```



In [30]:
```
genre = movies['genres'].str.get_dummies('|')
```

Out[30]:

	Action	Adventure	Animation	Comedy	Crime	Documentary	Drama	Family	Fantasy
0	1	1	0	0	0	0	0	0	C
1	1	1	0	0	0	0	0	0	C
2	0	1	0	0	0	0	0	0	C
3	1	1	0	0	0	0	0	0	1
4	1	0	0	0	1	0	0	0	C
...	...	...	...	...	...	...	...	...	..
10861	0	0	0	0	0	1	0	0	C
10862	1	1	0	0	0	0	1	0	C
10863	0	0	0	1	0	0	0	0	C
10864	1	0	0	1	0	0	0	0	C
10865	0	0	0	0	0	0	0	0	C

10731 rows × 20 columns

In [31]:
```
# concatenate these dummies to the original movies data frame
movies = pd.concat([movies, genre], axis =1)
```

In [32]:
```
# Get the top genre
top_genre = (movies.iloc[:, 13:-1].sum().sort_values(ascending = False))
```

Out[32]:
```
Drama              4746
Comedy             3775
Thriller           2902
Action             2376
Romance            1708
Horror             1636
Adventure          1465
Crime              1353
Science Fiction    1221
Family             1214
Fantasy             908
Mystery             808
Animation           664
Documentary         470
Music               399
History             330
War                 268
Foreign             184
TV Movie            162
dtype: int64
```

let’s focus on the genres with a high volume of movies. You are going to identify the top 6 genres with the highest number of movies in them, and filter them out to produce the next chart:

In [33]:
```
# top 5 genres by the total number of movies
top5_genre = (movies.iloc[:, 13:-1] # get the genre columns only
              .sum() # sum them up
              .sort_values(ascending=False) # sort descending
              .head(5)# get the first 5
              .index.values # get the genre names
              )
```

Out[33]:
```
array(['Drama', 'Comedy', 'Thriller', 'Action', 'Romance'], dtype=object)
```

In [34]:

Out[34]:
```
[2014,
 2013,
 2015,
 2012,
 2011,
 2009,
 2008,
 2010,
 2007,
 2006,
 2005,
 2004,
 2003,
 2002,
 2001,
 2000,
 1999,
 1998,
 1996,
 1997,
 1994,
 1993,
 1995,
 1988,
 1989,
 1991,
 1990,
 1992,
 1987,
 1986,
 1985,
 1984,
 1981,
 1982,
 1983,
 1980,
 1978,
 1979,
 1977,
 1973,
 1971,
 1976,
 1974,
 1966,
 1975,
 1964,
 1970,
 1972,
 1967,
 1968,
 1965,
 1963,
 1960,
 1962,
 1961,
 1969]
```

In [35]:

Out[35]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000027E36971CA0
>
```

In [36]:
```
movies['release_years']= movies['release_year']
```

In [37]:

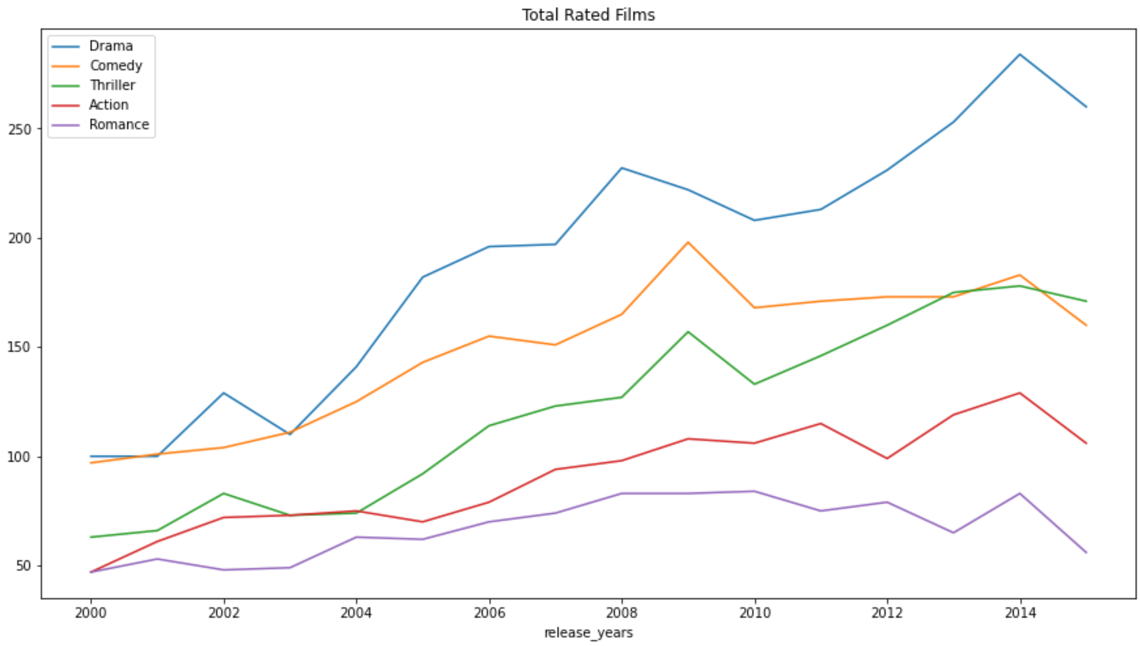
Out[37]:

	id	original_title	cast	director	runtime	genr
0	135397	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124.0	Action|Adventure|Scien Fiction|Thril
1	76341	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	120.0	Action|Adventure|Scien Fiction|Thril
2	262500	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	119.0	Adventure|Scien Fiction|Thril
3	140607	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	136.0	Action|Adventure|Scien Fiction|Fanta
4	168259	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	137.0	Action|Crime|Thril
...	...	...	...	...	...	
10861	21	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	Bruce Brown	95.0	Documenta
10862	20379	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	John Frankenheimer	176.0	Action|Adventure|Drar
10863	39768	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	Eldar Ryazanov	94.0	Mystery|Come
10864	21449	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	Woody Allen	80.0	Action|Come
10865	22293	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	Harold P. Warren	74.0	Hor

10731 rows × 33 columns

In [38]:

In [39]:
```
genre_groups.rolling(1).mean().plot(figsize=(15,8),
                                    title="Total Rated Films");
```



This gives a nice visual representation and helps you to interpret the data to answer the question you posed before. Here are the take-aways that I took from it:

- Drama and Comedy are the winner genres
- Seems that Action and Romance are not as popular

Conclusion

The preparation of the data, the modeling of these data, then the visualization of these data with a wide variety of graphs, and finally the interpretation of these graphs made it possible to conduct an analysis and a global view of movies released in the cinema between 2000 and 2015.

This study through a large volume of data, allowed me to determine the following points for movies between 2000 and 2015 only but we couldn't show recent trends from the dataset to show if the genres (adventure film) will still have highest overall revenue in recent time.

In this report, we have explored movies title that made the highest profit and how the genre increased from year to year.

- We then found that, as a percent of total films released in a given year, most genres maintained a relatively stable position in number of titles released for every year from the 1960s to the present date. We then looked specifically at drama titles to see if we could find any information that would be useful in identifying changes in tastes over time. We found that, while there are overall more drama movies being made, those movies are making less money on average.
- We examined factors that may affect a film's profit and revenue. We started by gathering data about basic financial information. Interestingly, we discovered that many films either lost money, or profited very little.

We found that adventure film had the highest overall revenue, while documentaries had a very high profit margin when compared to their small budgets.

Limitation

While trying to perform the data cleaning, we checked the percentage of null values from the required column needed for the research qestions. It was observed that less than 2% of it were missing which will have no effect on the dataset. Therefore, they were dropped.

No limitation was discovered for this project.

In [40]:
```
from subprocess import call
```

Out[40]: 1

In [ ]:

In [ ]: