In [2]:
```
# Filtering out the warnings

import warnings

warnings.filterwarnings('ignore')
```

In [3]:
```
# Importing the required libraries

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

IMDb Movie Assignment

You have the data for the 100 top-rated movies from the past decade along with various pieces of information about the movie, its actors, and the voters who have rated these movies online. In this assignment, you will try to find some interesting insights into these movies and their voters, using Python.

Task 1: Reading the data

- ### Subtask 1.1: Read the Movies Data.

Read the movies data file provided and store it in a dataframe `movies`.

In [37]:
```
# Read the csv file using 'read_csv'. Please write your dataset locatio
n here.
movies = pd.read_csv("Movie+Assignment+Data.csv")
movies
```

Out[37]:

Out[37]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	ac
0	La La Land	2016	30000000	151101803	Ryan Gosling	Emma Stone	Amiée Conn	
1	Zootopia	2016	150000000	341268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	
2	Lion	2016	12000000	51738905	Dev Patel	Nicole Kidman	Rooney Mara	
3	Arrival	2016	47000000	100546139	Amy Adams	Jeremy Renner	Forest Whitaker	
4	Manchester by the Sea	2016	9000000	47695371	Casey Affleck	Michelle Williams	Kyle Chandler	
...	...	...	...	...	...	...	...	
95	Whiplash	2014	3300000	13092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	
96	Before Midnight	2013	3000000	8114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	
97	Star Wars: Episode VII - The Force Awakens	2015	245000000	936662225	Doug Walker	Rob Walker	0	
98	Harry Potter and the Deathly Hallows: Part I	2010	150000000	296347721	Rupert Grint	Toby Jones	Alfred Enoch	
99	Tucker and Dale vs Evil	2010	5000000	223838	Katrina Bowden	Tyler Labine	Chelan Simmons	

100 rows × 62 columns

- ### Subtask 1.2: Inspect the Dataframe

Inspect the dataframe for dimensions, null-values, and summary of different numeric columns.

In [7]:
```python
# Check the number of rows and columns in the dataframe
movies.shape
```

Out[7]: (100, 62)

In [8]:
```python
# Check the column-wise info of the dataframe

movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 62 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Title                   100 non-null    object 
 1   title_year              100 non-null    int64  
 2   budget                  100 non-null    int64  
 3   Gross                   100 non-null    int64  
 4   actor_1_name            100 non-null    object 
 5   actor_2_name            100 non-null    object 
 6   actor_3_name            100 non-null    object 
 7   actor_1_facebook_likes  100 non-null    int64  
 8   actor_2_facebook_likes  99 non-null     float64
 9   actor_3_facebook_likes  98 non-null     float64
 10  IMDb_rating             100 non-null    float64
 11  genre_1                 100 non-null    object 
 12  genre_2                 97 non-null     object 
 13  genre_3                 74 non-null     object 
 14  MetaCritic              95 non-null     float64
 15  Runtime                 100 non-null    int64  
 16  CVotes10                100 non-null    int64  
 17  CVotes09                100 non-null    int64  
 18  CVotes08                100 non-null    int64  
 19  CVotes07                100 non-null    int64  
 20  CVotes06                100 non-null    int64  
 21  CVotes05                100 non-null    int64  
 22  CVotes04                100 non-null    int64  
 23  CVotes03                100 non-null    int64  
 24  CVotes02                100 non-null    int64  
```

```
 25  CVotes01                100 non-null    int64
 26  CVotesMale              100 non-null    int64
 27  CVotesFemale            100 non-null    int64
 28  CVotesU18               100 non-null    int64
 29  CVotesU18M              100 non-null    int64
 30  CVotesU18F              100 non-null    int64
 31  CVotes1829              100 non-null    int64
 32  CVotes1829M             100 non-null    int64
 33  CVotes1829F             100 non-null    int64
 34  CVotes3044              100 non-null    int64
 35  CVotes3044M             100 non-null    int64
 36  CVotes3044F             100 non-null    int64
 37  CVotes45A               100 non-null    int64
 38  CVotes45AM              100 non-null    int64
 39  CVotes45AF              100 non-null    int64
 40  CVotes1000              100 non-null    int64
 41  CVotesUS                100 non-null    int64
 42  CVotesnUS               100 non-null    int64
 43  VotesM                  100 non-null    float64
 44  VotesF                  100 non-null    float64
 45  VotesU18                100 non-null    float64
 46  VotesU18M               100 non-null    float64
 47  VotesU18F               100 non-null    float64
 48  Votes1829               100 non-null    float64
 49  Votes1829M              100 non-null    float64
 50  Votes1829F              100 non-null    float64
 51  Votes3044               100 non-null    float64
 52  Votes3044M              100 non-null    float64
 53  Votes3044F              100 non-null    float64
 54  Votes45A                100 non-null    float64
 55  Votes45AM               100 non-null    float64
 56  Votes45AF               100 non-null    float64
 57  Votes1000               100 non-null    float64
 58  VotesUS                 100 non-null    float64
 59  VotesnUS                100 non-null    float64
 60  content_rating          100 non-null    object
 61  Country                 100 non-null    object
dtypes: float64(21), int64(32), object(9)
memory usage: 48.6+ KB
```

In [9]:
```
# Check the summary for the numeric columns

movies.describe()
```

Out[9]:

	title_year	budget	Gross	actor_1_facebook_likes	actor_2_facebook_likes
count	100.000000	1.000000e+02	1.000000e+02	100.000000	99.000000
mean	2012.820000	7.838400e+07	1.468679e+08	13407.270000	7377.303030
std	1.919491	7.445295e+07	1.454004e+08	10649.037862	13471.568216
min	2010.000000	3.000000e+06	2.238380e+05	39.000000	12.000000
25%	2011.000000	1.575000e+07	4.199752e+07	1000.000000	580.000000
50%	2013.000000	4.225000e+07	1.070266e+08	13000.000000	1000.000000
75%	2014.000000	1.500000e+08	2.107548e+08	20000.000000	11000.000000
max	2016.000000	2.600000e+08	9.366622e+08	35000.000000	96000.000000

8 rows × 53 columns

Task 2: Data Analysis

Now that we have loaded the dataset and inspected it, we see that most of the data is in place. As of now, no data cleaning is required, so let's start with some data manipulation, analysis, and visualisation to get various insights about the data.

- ### Subtask 2.1: Reduce those Digits!

These numbers in the `budget` and `gross` are too big, compromising its readability. Let's convert the unit of the `budget` and `gross` columns from `$` to `million $` first.

In [10]:
```
# Divide the 'gross' and 'budget' columns by 1000000 to convert '$' to
 'million $'
movies['budget'] = movies['budget'].div(1000000)
```

```
movies['Gross'] = movies['Gross'].div(1000000)
movies
```

Out[10]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor
0	La La Land	2016	30.0	151.101803	Ryan Gosling	Emma Stone	Amiée Conn	
1	Zootopia	2016	150.0	341.268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	
2	Lion	2016	12.0	51.738905	Dev Patel	Nicole Kidman	Rooney Mara	
3	Arrival	2016	47.0	100.546139	Amy Adams	Jeremy Renner	Forest Whitaker	
4	Manchester by the Sea	2016	9.0	47.695371	Casey Affleck	Michelle Williams	Kyle Chandler	
...	...	...	...	...	...	...	...	
95	Whiplash	2014	3.3	13.092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	
96	Before Midnight	2013	3.0	8.114507	Seamus Davey-Fitzpatrick	Ariane Labed	Athina Rachel Tsangari	
97	Star Wars: Episode VII - The Force Awakens	2015	245.0	936.662225	Doug Walker	Rob Walker	0	
98	Harry Potter and the Deathly Hallows: Part I	2010	150.0	296.347721	Rupert Grint	Toby Jones	Alfred Enoch	
99	Tucker and Dale vs Evil	2010	5.0	0.223838	Katrina Bowden	Tyler Labine	Chelan Simmons	

100 rows × 62 columns

- ### Subtask 2.2: Let's Talk Profit!

1. Create a new column called `profit` which contains the difference of the two columns: `gross` and `budget`.
2. Sort the dataframe using the `profit` column as reference.
3. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`.
4. Plot a scatter or a joint plot between the columns `budget` and `profit` and write a few words on what you observed.
5. Extract the movies with a negative profit and store them in a new dataframe - `neg_profit`

In [11]:
```
# Create the new column named 'profit' by subtracting the 'budget' colu
mn from the 'gross' column
movies["profit"] = movies["Gross"] - movies["budget"]
movies["profit"]
```

Out[11]:
```
0     121.101803
1     191.268248
2      39.738905
3      53.546139
4      38.695371
         ...
95      9.792000
96      5.114507
97    691.662225
98    146.347721
99     -4.776162
Name: profit, Length: 100, dtype: float64
```

In [12]:
```
# Sort the dataframe with the 'profit' column as reference using the 's
ort_values' function. Make sure to set the argument
#'ascending' to 'False'

movies = movies.sort_values(by='profit', ascending=False)
movies
```

Out[12]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1
97	Star Wars: Episode VII - The Force Awakens	2015	245.0	936.662225	Doug Walker	Rob Walker	0	
11	The Avengers	2012	220.0	623.279547	Chris Hemsworth	Robert Downey Jr.	Scarlett Johansson	
47	Deadpool	2016	58.0	363.024263	Ryan Reynolds	Ed Skrein	Stefan Kapicic	
32	The Hunger Games: Catching Fire	2013	130.0	424.645577	Jennifer Lawrence	Josh Hutcherson	Sandra Ellis Lafferty	
12	Toy Story 3	2010	200.0	414.984497	Tom Hanks	John Ratzenberger	Don Rickles	
...	...	...	...	...	...	...	...	
46	Scott Pilgrim vs. the World	2010	60.0	31.494270	Anna Kendrick	Kieran Culkin	Ellen Wong	
7	Tangled	2010	260.0	200.807262	Brad Garrett	Donna Murphy	M.C. Gainey	
17	Edge of Tomorrow	2014	178.0	100.189501	Tom Cruise	Lara Pulver	Noah Taylor	
39	The Little Prince	2015	81.2	1.339152	Jeff Bridges	James Franco	Mackenzie Foy	
22	Hugo	2011	170.0	73.820094	ChloÃ« Grace Moretz	Christopher Lee	Ray Winstone	

100 rows × 63 columns

In [13]:
```
# Get the top 10 profitable movies by using position based indexing. Sp
ecify the rows till 10 (0-9)
```

```
movies.head(10)
```

Out[13]:

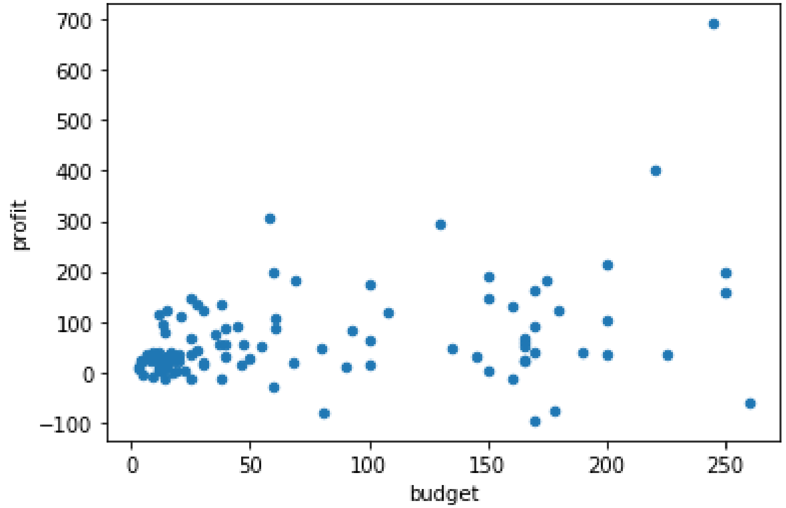
	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_
97	Star Wars: Episode VII - The Force Awakens	2015	245.0	936.662225	Doug Walker	Rob Walker	0	
11	The Avengers	2012	220.0	623.279547	Chris Hemsworth	Robert Downey Jr.	Scarlett Johansson	
47	Deadpool	2016	58.0	363.024263	Ryan Reynolds	Ed Skrein	Stefan Kapicic	
32	The Hunger Games: Catching Fire	2013	130.0	424.645577	Jennifer Lawrence	Josh Hutcherson	Sandra Ellis Lafferty	
12	Toy Story 3	2010	200.0	414.984497	Tom Hanks	John Ratzenberger	Don Rickles	
8	The Dark Knight Rises	2012	250.0	448.130642	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	
45	The Lego Movie	2014	60.0	257.756197	Morgan Freeman	Will Ferrell	Alison Brie	
1	Zootopia	2016	150.0	341.268248	Ginnifer Goodwin	Jason Bateman	Idris Elba	
41	Despicable Me	2010	69.0	251.501645	Steve Carell	Miranda Cosgrove	Jack McBrayer	
18	Inside Out	2015	175.0	356.454367	Amy Poehler	Mindy Kaling	Phyllis Smith	

10 rows × 63 columns

In [14]:

```
#Plot profit vs budget
```

```
movies.plot(x ='budget', y='profit', kind = 'scatter')
plt.show()
```



The dataset contains the 100 best performing movies from the year 2010 to 2016. However scatter plot tells a different story. You can notice that there are some movies with negative profit. Although good movies do incur losses, but there appear to be quite a few movie with losses. What can be the reason behind this? Lets have a closer look at this by finding the movies with negative profit.

In [15]:

```
#Find the movies with negative profit

movies[(movies['profit'] < 0)]
```

Out[15]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1
99	Tucker and Dale vs Evil	2010	5.0	0.223838	Katrina Bowden	Tyler Labine	Chelan Simmons	
89	Amour	2012	8.9	0.225377	Isabelle Huppert	Emmanuelle Riva	Jean-Louis Trintignant	

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1
56	Rush	2013	38.0	26.903709	Chris Hemsworth	Olivia Wilde	Alexandra Maria Lara	
66	Warrior	2011	25.0	13.651662	Tom Hardy	Frank Grillo	Kevin Dunn	
82	Flipped	2010	14.0	1.752214	Madeline Carroll	Rebecca De Mornay	Aidan Quinn	
28	X-Men: First Class	2011	160.0	146.405371	Jennifer Lawrence	Michael Fassbender	Oliver Platt	
46	Scott Pilgrim vs. the World	2010	60.0	31.494270	Anna Kendrick	Kieran Culkin	Ellen Wong	
7	Tangled	2010	260.0	200.807262	Brad Garrett	Donna Murphy	M.C. Gainey	
17	Edge of Tomorrow	2014	178.0	100.189501	Tom Cruise	Lara Pulver	Noah Taylor	
39	The Little Prince	2015	81.2	1.339152	Jeff Bridges	James Franco	Mackenzie Foy	
22	Hugo	2011	170.0	73.820094	ChloÃ« Grace Moretz	Christopher Lee	Ray Winstone	

11 rows × 63 columns

`Checkpoint 1:` Can you spot the movie `Tangled` in the dataset? You may be aware of the movie 'Tangled'. Although its one of the highest grossing movies of all time, it has negative profit as per this result. If you cross check the gross values of this movie (link: https://www.imdb.com/title/tt0398286/), you can see that the gross in the dataset accounts only for the domestic gross and not the worldwide gross. This is true for may other movies also in the list.

- ### Subtask 2.3: The General Audience and the Critics

You might have noticed the column `MetaCritic` in this dataset. This is a very popular website where an average score is determined through the scores given by the top-rated critics. Second, you also have another column `IMDb_rating` which tells you the IMDb rating of a movie. This rating is determined by taking the average of hundred-thousands of ratings from the general audience.

As a part of this subtask, you are required to find out the highest rated movies which have been liked by critics and audiences alike.

1. Firstly you will notice that the `MetaCritic` score is on a scale of `100` whereas the `IMDb_rating` is on a scale of 10. First convert the `MetaCritic` column to a scale of 10.
2. Now, to find out the movies which have been liked by both critics and audiences alike and also have a high rating overall, you need to -
   - Create a new column `Avg_rating` which will have the average of the `MetaCritic` and `Rating` columns
   - Retain only the movies in which the absolute difference(using abs() function) between the `IMDb_rating` and `Metacritic` columns is less than 0.5. Refer to this link to know how abs() funtion works - https://www.geeksforgeeks.org/abs-in-python/ .
   - Sort these values in a descending order of `Avg_rating` and retain only the movies with a rating equal to higher than `8` and store these movies in a new dataframe `UniversalAcclaim`.

In [16]:
```
# Change the scale of MetaCritic

movies["MetaCritic"].div(10)
```

Out[16]:
```
97     8.1
11     6.9
47     6.5
32     7.6
12     9.2
      ... 
46     6.9
7      7.1
```

```
17     7.1
39     7.0
22     8.3
Name: MetaCritic, Length: 100, dtype: float64
```

In [17]:

```
# Find the average ratings

movies["IMDb_rating"].mean(axis = 0)
```

Out[17]:

```
7.8830000000000044
```

In [18]:

```
#Sort in descending order of average rating
movies = movies.sort_values(by='IMDb_rating', ascending=False)
movies
```

Out[18]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_
27	Inception	2010	160.0	292.568851	Leonardo DiCaprio	Tom Hardy	Joseph Gordon-Levitt	
26	Interstellar	2014	165.0	187.991439	Matthew McConaughey	Anne Hathaway	Mackenzie Foy	
95	Whiplash	2014	3.3	13.092000	J.K. Simmons	Melissa Benoist	Chris Mulkey	
35	Django Unchained	2012	100.0	162.804648	Leonardo DiCaprio	Christoph Waltz	Ato Essandoh	
8	The Dark Knight Rises	2012	250.0	448.130642	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	
...	...	...	...	...	...	...	...	
55	True Grit	2010	38.0	171.031347	Matt Damon	Jeff Bridges	Bruce Green	
32	The Hunger Games: Catching Fire	2013	130.0	424.645577	Jennifer Lawrence	Josh Hutcherson	Sandra Ellis Lafferty	

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_
52	Lone Survivor	2013	40.0	125.069696	Jerry Ferrara	Scott Elrod	Dan Bilzerian	
46	Scott Pilgrim vs. the World	2010	60.0	31.494270	Anna Kendrick	Kieran Culkin	Ellen Wong	
22	Hugo	2011	170.0	73.820094	ChloÃ« Grace Moretz	Christopher Lee	Ray Winstone	

100 rows × 63 columns

In [19]:

```
# Find the movies with metacritic-rating < 0.5 and also with the averag
e rating of >8

movies[(movies['MetaCritic'] < 0.5) & (movies['IMDb_rating'] > 8)]
```

Out[19]:

Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1_facebook_l

0 rows × 63 columns

`Checkpoint 2:` Can you spot a `Star Wars` movie in your final dataset?

- ### Subtask 2.4: Find the Most Popular Trios - I

You're a producer looking to make a blockbuster movie. There will primarily be three lead roles in your movie and you wish to cast the most popular actors for it. Now, since you don't want to take a risk, you will cast a trio which has already acted in together in a movie before. The metric that you've chosen to check the popularity is the Facebook likes of each of these actors.

The dataframe has three columns to help you out for the same, viz. `actor_1_facebook_likes`, `actor_2_facebook_likes`, and `actor_3_facebook_likes`. Your objective is to find the trios which has the most number of

Facebook likes combined. That is, the sum of `actor_1_facebook_likes`, `actor_2_facebook_likes` and `actor_3_facebook_likes` should be maximum. Find out the top 5 popular trios, and output their names in a list.

In [39]:
```
# Write your code here

movies["trios1"] = movies["actor_1_facebook_likes"] + movies["actor_2_f
acebook_likes"] + movies["actor_3_facebook_likes"]
movies = movies.sort_values(by='trios1', ascending=False)
movies.head(5)
```

Out[39]:

	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor
2	Lion	2016	12.0	51.738905	Dev Patel	Nicole Kidman	Rooney Mara	
27	Inception	2010	160.0	292.568851	Leonardo DiCaprio	Tom Hardy	Joseph Gordon-Levitt	
14	X-Men: Days of Future Past	2014	200.0	233.914986	Jennifer Lawrence	Peter Dinklage	Hugh Jackman	
4	Manchester by the Sea	2016	9.0	47.695371	Casey Affleck	Michelle Williams	Kyle Chandler	
8	The Dark Knight Rises	2012	250.0	448.130642	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	

5 rows × 64 columns

- ### Subtask 2.5: Find the Most Popular Trios - II

In the previous subtask you found the popular trio based on the total number of facebook likes. Let's add a small condition to it and make sure that all three actors are popular. The condition is **none of the three actors' Facebook likes should be less than half of the other two**. For example, the following is a valid combo:

- actor_1_facebook_likes: 70000

- actor_2_facebook_likes: 40000
- actor_3_facebook_likes: 50000

But the below one is not:

- actor_1_facebook_likes: 70000
- actor_2_facebook_likes: 40000
- actor_3_facebook_likes: 30000

since in this case, `actor_3_facebook_likes` is 30000, which is less than half of `actor_1_facebook_likes`.

Having this condition ensures that you aren't getting any unpopular actor in your trio (since the total likes calculated in the previous question doesn't tell anything about the individual popularities of each actor in the trio.).

You can do a manual inspection of the top 5 popular trios you have found in the previous subtask and check how many of those trios satisfy this condition. Also, which is the most popular trio after applying the condition above?

Write your answers below.

- **`No. of trios that satisfy the above condition:`**
- **`Most popular trio after applying the condition:`**

`Optional:` Even though you are finding this out by a natural inspection of the dataframe, can you also achieve this through some *if-else* statements to incorporate this. You can try this out on your own time after you are done with the assignment.

In [40]:
```
# Your answer here (optional)
# run code 1.1.1 before running this code
movies[((movies["actor_1_facebook_likes"] > (movies["actor_2_facebook_l
ikes"].div(2))) & (movies["actor_1_facebook_likes"] > (movies["actor_3_
facebook_likes"].div(2))))&
((movies["actor_2_facebook_likes"] > (movies["actor_1_facebook_likes"].
```

```
div(2))) & (movies["actor_2_facebook_likes"] > (movies["actor_3_faceboo
k_likes"].div(2))))&
((movies["actor_3_facebook_likes"] > (movies["actor_1_facebook_likes"].
div(2))) & (movies["actor_3_facebook_likes"] > (movies["actor_2_faceboo
k_likes"].div(2))))].head(5)
```

Out[40]:

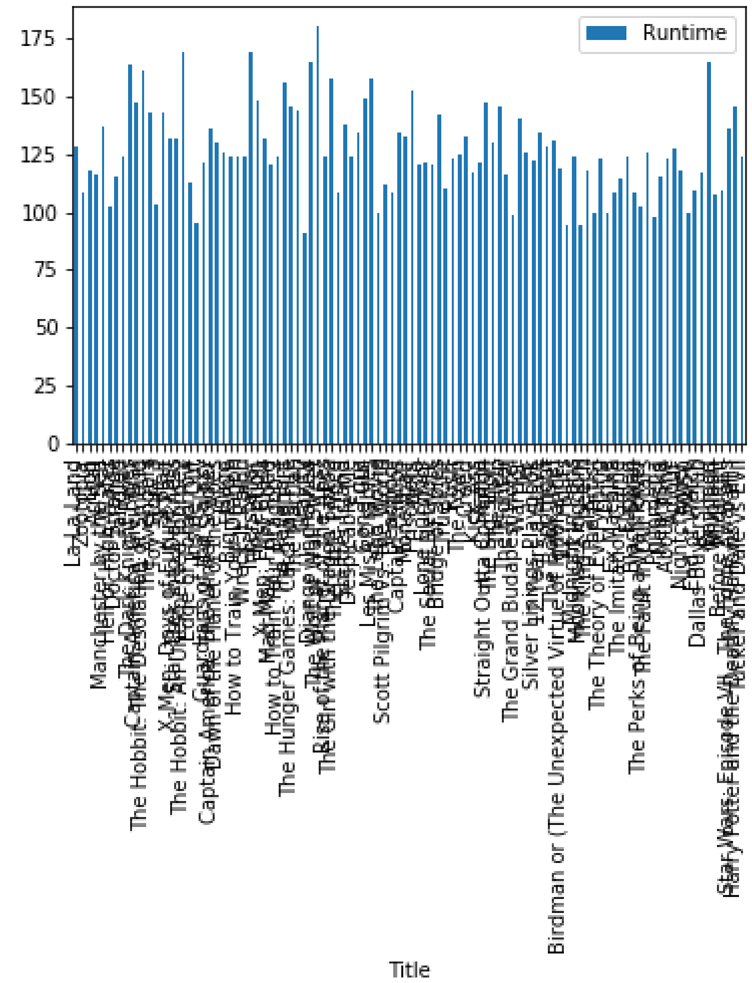
	Title	title_year	budget	Gross	actor_1_name	actor_2_name	actor_3_name	actor_1
27	Inception	2010	160.0	292.568851	Leonardo DiCaprio	Tom Hardy	Joseph Gordon-Levitt	
14	X-Men: Days of Future Past	2014	200.0	233.914986	Jennifer Lawrence	Peter Dinklage	Hugh Jackman	
8	The Dark Knight Rises	2012	250.0	448.130642	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	
11	The Avengers	2012	220.0	623.279547	Chris Hemsworth	Robert Downey Jr.	Scarlett Johansson	
9	Captain America: Civil War	2016	250.0	407.197282	Robert Downey Jr.	Scarlett Johansson	Chris Evans	

5 rows × 64 columns

- ### Subtask 2.6: Runtime Analysis

There is a column named `Runtime` in the dataframe which primarily shows the length of the movie. It might be intersting to see how this variable this distributed. Plot a `histogram` or `distplot` of seaborn to find the `Runtime` range most of the movies fall into.

In [21]:
```
# Runtime histogram/density plot
movies.plot(kind='bar',x='Title',y='Runtime')
plt.show()
```



`Checkpoint 3:` Most of the movies appear to be sharply 2 hour-long.

- ### Subtask 2.7: R-Rated Movies

Although R rated movies are restricted movies for the under 18 age group, still there are vote counts from that age group. Among all the R rated movies that have been voted by the under-18 age group, find the top 10 movies that have the highest number of votes i.e. `CVotesU18` from the `movies` dataframe. Store these in a dataframe named `PopularR`.

In [22]:
```
# Write your code here
PopularR = movies[['Title','CVotesU18','content_rating']]
PopularR = PopularR.sort_values(by='CVotesU18', ascending=False)
PopularR.loc[PopularR['content_rating'] == 'R'].head(10)
```

Out[22]:

	Title	CVotesU18	content_rating
47	Deadpool	4598	R
36	The Wolf of Wall Street	3622	R
35	Django Unchained	3250	R
29	Mad Max: Fury Road	3159	R
95	Whiplash	2878	R
31	The Revenant	2619	R
40	Shutter Island	2321	R
43	Gone Girl	2286	R
65	The Grand Budapest Hotel	2083	R
72	Birdman or (The Unexpected Virtue of Ignorance)	1891	R

`Checkpoint 4:` Are these kids watching `Deadpool` a lot?

Task 3 : Demographic analysis

If you take a look at the last columns in the dataframe, most of these are related to demographics of the voters (in the last subtask, i.e., 2.8, you made use one of these columns - CVotesU18). We also have three genre columns indicating the genres of a particular movie. We will extensively use these columns for the third and the final stage of our assignment wherein we will analyse the voters across all demographics and also see how these vary across various genres. So without further ado, let's get started with `demographic analysis`.

- ### Subtask 3.1 Combine the Dataframe by Genres

There are 3 columns in the dataframe - `genre_1`, `genre_2`, and `genre_3`. As a part of this subtask, you need to aggregate a few values over these 3 columns.

1. First create a new dataframe `df_by_genre` that contains `genre_1`, `genre_2`, and `genre_3` and all the columns related to **CVotes/Votes** from the `movies` data frame. There are 47 columns to be extracted in total.
2. Now, Add a column called `cnt` to the dataframe `df_by_genre` and initialize it to one. You will realise the use of this column by the end of this subtask.
3. First group the dataframe `df_by_genre` by `genre_1` and find the sum of all the numeric columns such as `cnt`, columns related to CVotes and Votes columns and store it in a dataframe `df_by_g1`.
4. Perform the same operation for `genre_2` and `genre_3` and store it dataframes `df_by_g2` and `df_by_g3` respectively.
5. Now that you have 3 dataframes performed by grouping over `genre_1`, `genre_2`, and `genre_3` separately, it's time to combine them. For this, add the three dataframes and store it in a new dataframe `df_add`, so that the corresponding values of Votes/CVotes get added for each genre.There is a function called `add()` in pandas which lets you do this. You can refer to this link to see how this function works. https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.add.html
6. The column `cnt` on aggregation has basically kept the track of the number of occurences of each genre.Subset the genres that have atleast 10 movies into a new dataframe `genre_top10` based on the `cnt` column value.
7. Now, take the mean of all the numeric columns by dividing them with the column value `cnt` and store it back to the same dataframe. We will be using this dataframe for further analysis in this task unless it is explicitly mentioned to use the dataframe `movies`.
8. Since the number of votes can't be a fraction, type cast all the CVotes related columns to integers. Also, round off all the Votes related columns upto two digits after the decimal point.

In [23]:
```
# Create the dataframe df_by_genre
d1 = movies.loc[:,"CVotes10":"VotesnUS"]
df_by_genre = movies.loc[:,"genre_1":"genre_3"]
df_by_genre = df_by_genre.join(d1)
df_by_genre
```

Out[23]:

	genre_1	genre_2	genre_3	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05
0	Comedy	Drama	Music	74245	71191	64640	38831	17377	8044
1	Animation	Adventure	Comedy	53626	70912	102352	57261	16719	4539
2	Biography	Drama	NaN	23325	29830	40564	20296	5842	1669
3	Drama	Mystery	Sci-Fi	55533	87850	109536	65440	26913	10556
4	Drama	NaN	NaN	18191	33532	46596	29626	11879	4539
...	...	...	...	...	...	...	...	...	...
95	Drama	Music	NaN	110404	161864	132656	56007	16577	6031
96	Drama	Romance	NaN	16953	22109	31439	19251	8142	3412
97	Action	Adventure	Fantasy	155391	161810	166378	99402	40734	18060
98	Adventure	Family	Fantasy	68937	54947	102488	80465	31205	11792
99	Comedy	Horror	NaN	16572	19818	44460	35863	13456	4588

100 rows × 47 columns

In [24]:
```
# Create a column cnt and initialize it to 1
df_by_genre['cnt'] = 1
df_by_genre
```

Out[24]:

	genre_1	genre_2	genre_3	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05
0	Comedy	Drama	Music	74245	71191	64640	38831	17377	8044
1	Animation	Adventure	Comedy	53626	70912	102352	57261	16719	4539
2	Biography	Drama	NaN	23325	29830	40564	20296	5842	1669

	genre_1	genre_2	genre_3	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05
3	Drama	Mystery	Sci-Fi	55533	87850	109536	65440	26913	10556
4	Drama	NaN	NaN	18191	33532	46596	29626	11879	4539
...	...	...	...	...	...	...	...	...	...
95	Drama	Music	NaN	110404	161864	132656	56007	16577	6031
96	Drama	Romance	NaN	16953	22109	31439	19251	8142	3412
97	Action	Adventure	Fantasy	155391	161810	166378	99402	40734	18060
98	Adventure	Family	Fantasy	68937	54947	102488	80465	31205	11792
99	Comedy	Horror	NaN	16572	19818	44460	35863	13456	4588

100 rows × 48 columns

In [25]:
```python
# Group the movies by individual genres
df_by_g1 = df_by_genre.groupby('genre_1').sum()
print(df_by_g1)
df_by_g2 = df_by_genre.groupby('genre_2').sum()
print(df_by_g2)
df_by_g3 = df_by_genre.groupby('genre_3').sum()
print(df_by_g3)
```

```
           CVotes10  CVotes09  CVotes08  CVotes07  CVotes06  CVotes05  
\
genre_1

Action      2928407   3261919   4247693   2662020    986774    364234

Adventure   1058779   1179818   1560541    966275    365486    136985

Animation    681562    798227   1153214    722782    251076     83069

Biography    666831   1088430   1654704    962977    306247    100005

Comedy       371217    496905    770395    518566    205434     81933
```

```
Crime          383290    690221   1083469    627593    206756     71460

Drama         1080725   1494053   1827363   1078966    417205    163874

Mystery        150405    230844    278844    132349     45167     15615


           CVotes04  CVotes03  CVotes02  CVotes01  ...  Votes3044  Vote
s3044M  \
genre_1                                            ...

Action       156150     89483     61975    162426  ...      209.1
208.8
Adventure     58559     33174     22018     48100  ...       92.7
  92.6
Animation     30718     15733     10026     25193  ...       85.4
  84.9
Biography     38874     21536     15365     37469  ...      100.8
100.7
Comedy        35788     20965     15286     33241  ...       68.6
  68.7
Crime         30336     17190     11757     25839  ...       69.4
  69.7
Drama         75525     45846     32068     71464  ...      139.3
139.0
Mystery        7061      3780      2662      4703  ...        7.9
   7.9

           Votes3044F  Votes45A  Votes45AM  Votes45AF  Votes1000  Votes
US  \
genre_1

Action          210.0     206.5      206.0      209.0      197.2     21
5.8
Adventure        93.5      92.0       91.6       93.8       88.9      9
5.3
Animation        87.8      84.5       84.1       86.7       80.0      8
7.6
Biography       101.3     100.5      100.0      102.9       94.7     10
```

```
3.3
Comedy          68.9       67.7        67.5        68.7        62.7      7
0.9
Crime           68.8       68.7        68.6        69.6        66.3      7
1.9
Drama          139.7      137.7       137.2       138.7       130.0     14
3.2
Mystery          8.0        7.5         7.4         7.6         7.6
7.8

           VotesnUS  cnt
genre_1
Action        209.5   27
Adventure      93.5   12
Animation      86.1   11
Biography     101.5   13
Comedy         69.4    9
Crime          70.1    9
Drama         141.1   18
Mystery         8.1    1

[8 rows x 45 columns]
           CVotes10  CVotes09  CVotes08  CVotes07  CVotes06  CVotes05
\
genre_2

Action       238060    285510    430062    260106     88580     29250

Adventure   2297820   2548864   3271725   2055600    758009    272735

Biography    185172    313178    576374    370003    119348     38643

Comedy       428995    624720    854162    512668    193916     76752

Crime         19576     40247     85359     64633     24920      8548

Drama       1923492   2761237   4112363   2492241    853434    300100

Family        68937     54947    102488     80465     31205     11792
```

```
Fantasy       270616    290831    447307    291071    120920     47215

History        15757     32840     83322     63800     19183      5178

Horror         16572     19818     44460     35863     13456      4588

Music         110404    161864    132656     56007     16577      6031

Musical        54268     47750     63323     45160     22393     10744

Mystery        85313    152619    233474    148176     55575     20135

Romance       230425    270702    353326    215355     87701     35226

Sci-Fi        350243    361819    433983    284879    125143     53350

Sport          21364     28964     58237     45563     16432      5118

Thriller      624792    709424    734770    365134    135892     52714

War            15911     17607     32570     24461     10274      3848

Western       234824    339329    286911    121445     38251     14227


           CVotes04  CVotes03  CVotes02  CVotes01  ...  Votes3044  Vote
s3044M  \
genre_2                                            ...

Action        10820      5521      3598      8821  ...       30.9
  30.7
Adventure    113691     64623     44121    116937  ...      171.0
170.4
Biography     14844      7974      5248     13828  ...       38.3
  38.2
Comedy        35193     20995     14798     30509  ...       54.0
  54.1
Crime          3261      1669       970      1689  ...        7.5
```

```
   7.6
Drama        124511     70205     49642    112896  ...        270.2
270.4
Family         4808      2454      1617      4522  ...          7.4
   7.3
Fantasy       19848     10871      6885     14702  ...         22.9
  22.9
History        1657       735       419       878  ...          7.5
   7.5
Horror         1684       855       479       848  ...          7.5
   7.5
Music          2937      1859      1263      2723  ...          8.3
   8.3
Musical        5551      3484      2490      5020  ...          7.3
   7.2
Mystery        8863      5034      3359      6916  ...         15.4
  15.4
Romance       16298      9448      6708     14889  ...         38.7
  38.7
Sci-Fi        24462     15069     10747     28001  ...         15.8
  15.8
Sport          1655       848       464      1151  ...          7.4
   7.4
Thriller      25734     15558     10716     25417  ...         24.1
  24.1
War            1387       726       342       755  ...          7.3
   7.2
Western        6469      4149      3181      8065  ...          8.3
   8.3

           Votes3044F  Votes45A  Votes45AM  Votes45AF  Votes1000  Votes
US  \
genre_2

Action           31.8      30.5       30.4       31.4       29.0      3
1.8
Adventure       173.9     169.2      168.4      172.8      162.7     17
6.4
Biography        38.5      38.0       37.9       38.8       35.4      3
```

```
9.4
Comedy          53.4      53.0       53.1       53.1       51.0     5
5.5
Crime            7.2       7.6        7.6        7.4        7.2
7.8
Drama          270.1     267.9      267.1      271.8      253.4    27
8.2
Family           8.1       7.4        7.3        8.0        6.7
7.9
Fantasy         23.3      23.0       22.8       23.7       22.1     2
3.4
History          7.5       7.7        7.6        7.9        7.4
7.7
Horror           7.7       7.5        7.4        7.7        7.1
7.7
Music            8.2       8.1        8.1        8.2        8.0
8.6
Musical          7.6       7.4        7.3        7.7        6.6
7.6
Mystery         15.2      15.2       15.2       15.1       14.8     1
5.9
Romance         38.7      37.9       37.8       37.7       35.3     3
9.7
Sci-Fi          15.6      15.6       15.6       15.6       14.9     1
6.2
Sport            7.4       7.3        7.3        7.5        7.1
7.9
Thriller        24.2      22.9       22.8       22.9       23.0     2
4.2
War              7.7       7.6        7.5        8.0        6.6
7.6
Western          8.3       8.0        8.0        8.1        7.8
8.4

           VotesnUS  cnt
genre_2
Action         31.1    4
Adventure     171.6   22
Biography      38.4    5
```

```
Comedy         54.3    7
Crime           7.6    1
Drama         272.6   35
Family          7.5    1
Fantasy        23.2    3
History         7.5    1
Horror          7.5    1
Music           8.4    1
Musical         7.5    1
Mystery        15.6    2
Romance        39.2    5
Sci-Fi         15.7    2
Sport           7.5    1
Thriller       24.5    3
War             7.5    1
Western         8.4    1

[19 rows x 45 columns]
           CVotes10  CVotes09  CVotes08  CVotes07  CVotes06  CVotes05
\
genre_3

Adventure    238060    285510    430062    260106     88580     29250

Comedy       583404    653362    882294    559835    200937     68167

Crime        171660    236650    250667    129164     46715     18682

Drama        400221    680085   1167327    748493    258717     88338

Family        29228     40728     77893     62936     27932     11179

Fantasy      301836    311392    442460    308676    120911     46269

History      135504    227547    311209    159262     48678     16055

Music         74245     71191     64640     38831     17377      8044

Mystery      274446    443661    654167    375087    128131     44818
```

```
Romance       319534    418790    715954    497486    193588     75675

Sci-Fi       1975041   2169036   2569011   1517219    546668    200825

Sport          95717    140282    214806    138600     44470     13821

Thriller      456909    756067   1258608    810665    280154     97239

War            36753     54703    111271     82505     30231     10553

Western        21094     40901     91825     67175     23055      7191


           CVotes04  CVotes03  CVotes02  CVotes01  ...  Votes3044  Vote
s3044M  \
genre_3                                            ...

Adventure     10820      5521      3598      8821  ...       30.9
  30.7
Comedy        26488     14258      9307     24617  ...       54.8
  54.6
Crime          8674      5854      4258      9689  ...        8.0
   8.1
Drama         35439     19075     12475     26948  ...       91.8
  91.7
Family         4664      2674      1700      3023  ...        7.4
   7.4
Fantasy       19555     11362      7808     24139  ...       30.4
  30.2
History        6307      3649      2729      8413  ...       23.7
  23.6
Music          3998      2839      2407      6802  ...        7.9
   7.9
Mystery       18755     10578      7149     17825  ...       30.8
  30.8
Romance       32615     18250     12492     25186  ...       60.2
  60.2
Sci-Fi        87463     50835     35424     86434  ...      117.8
```

```
117.7
Sport          5155      2509      1828      4083  ...       15.5
  15.5
Thriller      39547     22382     15051     32213  ...       76.5
  76.6
War            4303      2388      1629      3246  ...        7.4
   7.4
Western        2678      1305       779      1672  ...        7.6
   7.6

           Votes3044F  Votes45A  Votes45AM  Votes45AF  Votes1000  Votes
US  \
genre_3

Adventure        31.8      30.5       30.4       31.4       29.0      3
1.8
Comedy           56.0      54.3       54.1       55.3       51.7      5
6.2
Crime             7.7       7.6        7.6        7.5        7.8
8.1
Drama            92.0      91.2       91.0       92.7       86.1      9
4.5
Family            7.4       7.5        7.5        7.6        7.4
7.7
Fantasy          31.7      30.4       30.0       31.8       28.4      3
1.5
History          23.8      23.3       23.1       24.2       22.0      2
4.5
Music             7.8       7.6        7.6        7.5        7.1
8.3
Mystery          31.5      30.4       30.3       31.3       29.3      3
1.9
Romance          60.9      59.9       59.7       61.2       54.6      6
2.1
Sci-Fi          117.6     115.5      115.2      115.9      113.0     12
1.3
Sport            15.2      15.2       15.2       15.0       14.2      1
6.0
Thriller         75.9      76.7       76.5       77.8       73.2      7
```

```
8.9
War                  7.4        7.4         7.4         7.4         6.8
7.6
Western              7.5        7.7         7.7         7.7         7.3
7.9

           VotesnUS  cnt
genre_3
Adventure      31.1    4
Comedy         55.2    7
Crime           8.1    1
Drama          92.3   12
Family          7.5    1
Fantasy        30.5    4
History        23.7    3
Music           8.1    1
Mystery        31.3    4
Romance        60.9    8
Sci-Fi        118.3   15
Sport          15.6    2
Thriller       77.0   10
War             7.5    1
Western         7.6    1

[15 rows x 45 columns]
```

In [26]:

```
# Add the grouped data frames and store it in a new data frame

df_add = df_by_g1 + df_by_g2 +df_by_g3
df_add
```

Out[26]:

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Action	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Adventure	3594659.0	4014192.0	5262328.0	3281981.0	1212075.0	438970.0	183070.0	103318.0
Animation	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Biography	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Comedy	1383616.0	1774987.0	2506851.0	1591069.0	600287.0	226852.0	97469.0	56218.0
Crime	574526.0	967118.0	1419495.0	821390.0	278391.0	98690.0	42271.0	24713.0
Drama	3404438.0	4935375.0	7107053.0	4319700.0	1529356.0	552312.0	235475.0	135126.0
Family	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fantasy	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
History	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Horror	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Music	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Musical	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Mystery	510164.0	827124.0	1166485.0	655612.0	228873.0	80568.0	34679.0	19392.0
Romance	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Sci-Fi	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Sport	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Thriller	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
War	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Western	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

20 rows × 45 columns

In [31]:

```
# Extract genres with atleast 10 occurences
genre_top10 = df_add.loc[(df_add['cnt'] != 'NAN') & (df_add['cnt'] > 10
)]
genre_top10
```

Out[31]:

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Adventure	3594659.0	4014192.0	5262328.0	3281981.0	1212075.0	438970.0	183070.0	103318.0
Comedy	1383616.0	1774987.0	2506851.0	1591069.0	600287.0	226852.0	97469.0	56218.0

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Crime	574526.0	967118.0	1419495.0	821390.0	278391.0	98690.0	42271.0	24713.0
Drama	3404438.0	4935375.0	7107053.0	4319700.0	1529356.0	552312.0	235475.0	135126.0

4 rows × 45 columns

In [28]:

```
# Take the mean for every column by dividing with cnt
genre_top10 = genre_top10.div(genre_top10.cnt, axis='index')
genre_top10
```

Out[28]:

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes0
Adventure	94596.289474	105636.631579	138482.315789	86367.921053	31896.710526	11551.84210
Comedy	60157.217391	77173.347826	108993.521739	69176.913043	26099.434783	9863.13043
Crime	52229.636364	87919.818182	129045.000000	74671.818182	25308.272727	8971.81818
Drama	52375.969231	75928.846154	109339.276923	66456.923077	23528.553846	8497.10769

4 rows × 45 columns

In [29]:

```
# Rounding off the columns of Votes to two decimals
genre_top10 = genre_top10.round(decimals = 2)
genre_top10
```

Out[29]:

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Adventure	94596.29	105636.63	138482.32	86367.92	31896.71	11551.84	4817.63	2718.89
Comedy	60157.22	77173.35	108993.52	69176.91	26099.43	9863.13	4237.78	2444.26
Crime	52229.64	87919.82	129045.00	74671.82	25308.27	8971.82	3842.82	2246.64
Drama	52375.97	75928.85	109339.28	66456.92	23528.55	8497.11	3622.69	2078.86

4 rows × 45 columns

In [30]:
```
# Converting CVotes to int type
genre_top10 = genre_top10.loc[:, 'CVotes10':'CVotesnUS'].astype('int')
genre_top10
```

Out[30]:

	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03
Adventure	94596	105636	138482	86367	31896	11551	4817	2718
Comedy	60157	77173	108993	69176	26099	9863	4237	2444
Crime	52229	87919	129045	74671	25308	8971	3842	2246
Drama	52375	75928	109339	66456	23528	8497	3622	2078

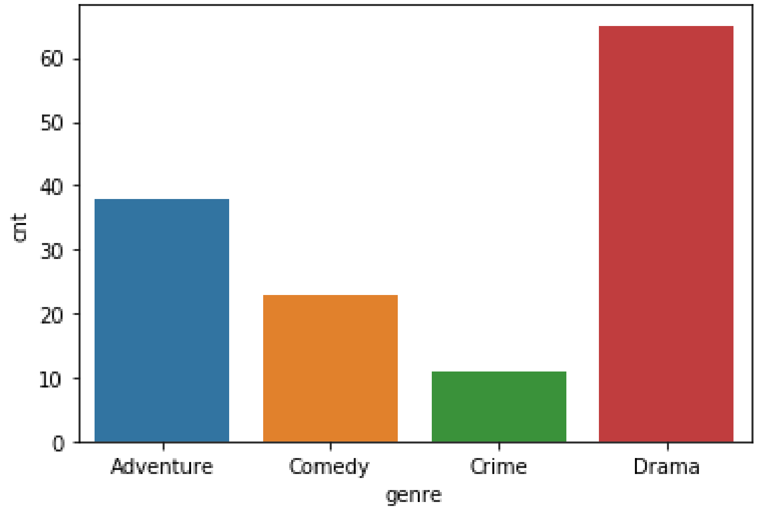
4 rows × 27 columns

If you take a look at the final dataframe that you have gotten, you will see that you now have the complete information about all the demographic (Votes- and CVotes-related) columns across the top 10 genres. We can use this dataset to extract exciting insights about the voters!

- ### Subtask 3.2: Genre Counts!

Now let's derive some insights from this data frame. Make a bar chart plotting different genres vs cnt using seaborn.

In [32]:
```
# Countplot for genres
# Run 3.1.5 before running this code
genre_top10 = genre_top10.reset_index(drop=True)
lst = ['Adventure', 'Comedy', 'Crime', 'Drama']
df = pd.DataFrame(lst,columns =['genre'],dtype='object')
genre_top10 = genre_top10.join(df)
sns.barplot(data = genre_top10, x= 'genre', y = 'cnt')
plt.show()
```



`Checkpoint 5:` Is the bar for `Drama` the tallest?

- ### Subtask 3.3: Gender and Genre

If you have closely looked at the Votes- and CVotes-related columns, you might have noticed the suffixes `F` and `M` indicating Female and Male. Since we have the vote counts for both males and females, across various age groups, let's now see how the popularity of genres vary between the two genders in the dataframe.

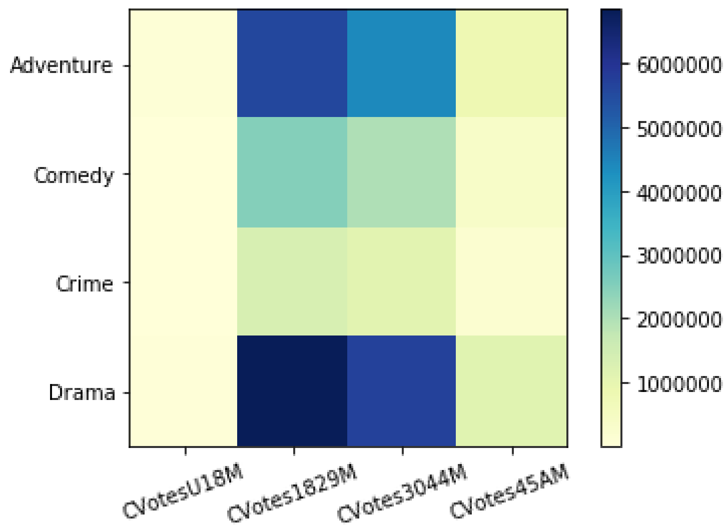
1. Make the first heatmap to see how the average number of votes of males is varying across the genres. Use seaborn heatmap for this analysis. The X-axis should contain the four age-groups for males, i.e., `CVotesU18M`, `CVotes1829M`, `CVotes3044M`, and `CVotes45AM`. The Y-axis will have the genres and the annotation in the heatmap tell the average number of votes for that age-male group.
2. Make the second heatmap to see how the average number of votes of females is varying across the genres. Use seaborn heatmap for this analysis. The X-axis should contain the four age-groups for females, i.e., `CVotesU18F`, `CVotes1829F`, `CVotes3044F`, and `CVotes45AF`. The Y-axis will have the genres and the annotation in the heatmap tell the average number of votes for that age-female group.

3. Make sure that you plot these heatmaps side by side using `subplots` so that you can easily compare the two genders and derive insights.
4. Write your any three inferences from this plot. You can make use of the previous bar plot also here for better insights. Refer to this link- https://seaborn.pydata.org/generated/seaborn.heatmap.html. You might have to plot something similar to the fifth chart in this page (You have to plot two such heatmaps side by side).
5. Repeat subtasks 1 to 4, but now instead of taking the CVotes-related columns, you need to do the same process for the Votes-related columns. These heatmaps will show you how the two genders have rated movies across various genres.

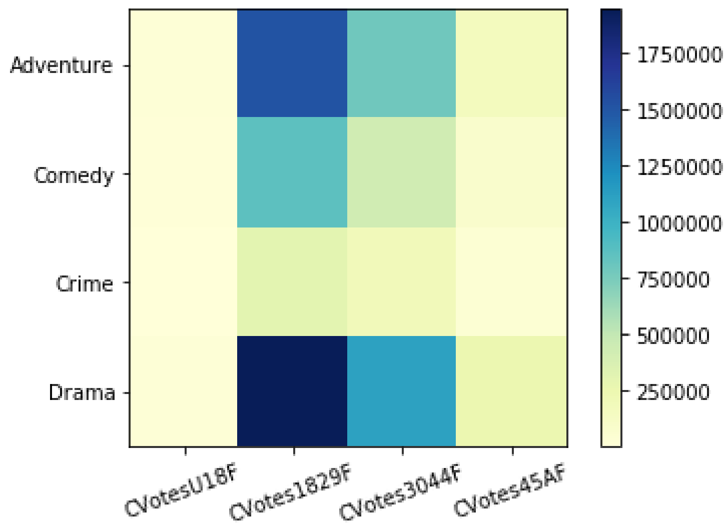
You might need the below link for formatting your heatmap. https://stackoverflow.com/questions/56942670/matplotlib-seaborn-first-and-last-row-cut-in-half-of-heatmap-plot

- Note : Use `genre_top10` dataframe for this subtask

In [33]:
```
# 1st set of heat maps for CVotes-related columns
dfM = genre_top10[["genre","CVotesU18M","CVotes1829M","CVotes3044M","CVotes45AM"]]
dfM = dfM.set_index("genre")
plt.imshow(dfM, cmap="YlGnBu")
plt.colorbar()
plt.xticks(range(len(dfM)),dfM.columns, rotation=20)
plt.yticks(range(len(dfM)),dfM.index)
plt.show()
```



In [34]:
```
dfF = genre_top10[["genre","CVotesU18F","CVotes1829F","CVotes3044F","CV
otes45AF"]]
dfF = dfF.set_index("genre")
plt.imshow(dfF, cmap="YlGnBu")
plt.colorbar()
plt.xticks(range(len(dfF)),dfF.columns, rotation=20)
plt.yticks(range(len(dfF)),dfF.index)
plt.show()
```

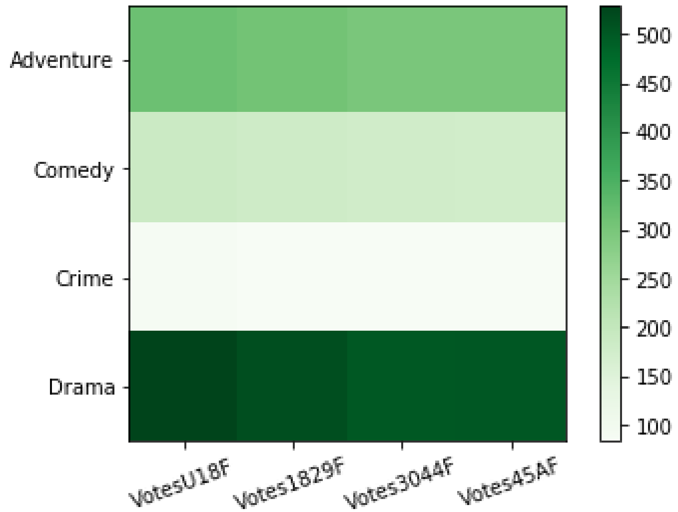


`Inferences:` A few inferences that can be seen from the heatmap above is that males have voted more than females, and Sci-Fi appears to be most popular among the 18-29 age group irrespective of their gender. What more can you infer from the two heatmaps that you have plotted? Write your three inferences/observations below:

- Inference 1:
- Inference 2:
- Inference 3:

In [35]:

```
# 2nd set of heat maps for Votes-related columns
dfF1 = genre_top10[["genre","VotesU18F","Votes1829F","Votes3044F","Vote
s45AF"]]
dfF1 = dfF1.set_index("genre")
plt.imshow(dfF1, cmap="Greens")
plt.colorbar()
plt.xticks(range(len(dfF1)),dfF1.columns, rotation=20)
plt.yticks(range(len(dfF1)),dfF1.index)
plt.show()
```

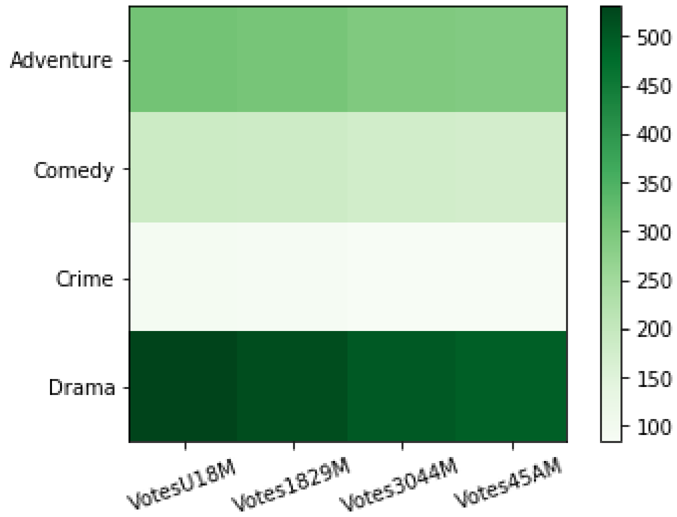


`Inferences:` Sci-Fi appears to be the highest rated genre in the age group of U18 for both males and females. Also, females in this age group have rated it a bit higher than the males in the same age group. What more can you infer from the two heatmaps that you have plotted? Write your three inferences/observations below:

- Inference 1:
- Inference 2:
- Inference 3:

In [36]:

```
dfM1 = genre_top10[["genre","VotesU18M","Votes1829M","Votes3044M","Vote
s45AM"]]
dfM1 = dfM1.set_index("genre")
plt.imshow(dfM1, cmap="Greens")
plt.colorbar()
plt.xticks(range(len(dfM1)),dfM1.columns, rotation=20)
plt.yticks(range(len(dfM1)),dfM1.index)
plt.show()
```



- ### Subtask 3.4: US vs non-US Cross Analysis

The dataset contains both the US and non-US movies. Let's analyse how both the US and the non-US voters have responded to the US and the non-US movies.

1. Create a column `IFUS` in the dataframe `movies`. The column `IFUS` should contain the value "USA" if the `Country` of the movie is "USA". For all other countries other than the USA, `IFUS` should contain the value `non-USA`.

1. Now make a boxplot that shows how the number of votes from the US people i.e. `CVotesUS` is varying for the US and non-US movies. Make use of the column `IFUS` to make this plot. Similarly, make another subplot that shows how non US voters have voted for the US and non-US movies by plotting `CVotesnUS` for both the US and non-US movies. Write any of your two inferences/observations from these plots.

1. Again do a similar analysis but with the ratings. Make a boxplot that shows how the ratings from the US people i.e. `VotesUS` is varying for the US and non-US movies. Similarly, make another subplot that shows how `VotesnUS` is varying for the US and non-US movies. Write any of your two inferences/observations from these plots.

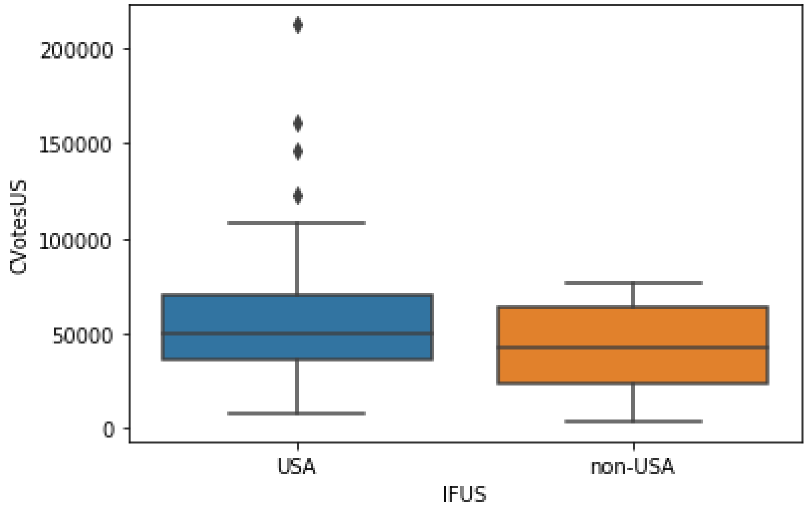
Note : Use `movies` dataframe for this subtask. Make use of this documention to format your boxplot - https://seaborn.pydata.org/generated/seaborn.boxplot.html

In [38]:
```
# Creating IFUS column
# run 1.1.1 before running this code
movies['IFUS'] ='USA'
movies.loc[movies.Country != 'USA', 'IFUS'] = 'non-USA'
movies['IFUS']
```

Out[38]:
```
0         USA
1         USA
2     non-USA
3         USA
4         USA
       ...
95        USA
96        USA
97        USA
98    non-USA
99    non-USA
Name: IFUS, Length: 100, dtype: object
```

In [39]:
```
# Box plot - 1: CVotesUS(y) vs IFUS(x)
sns.boxplot(y='CVotesUS', x='IFUS', data=movies)
```

Out[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23069115e08>
```

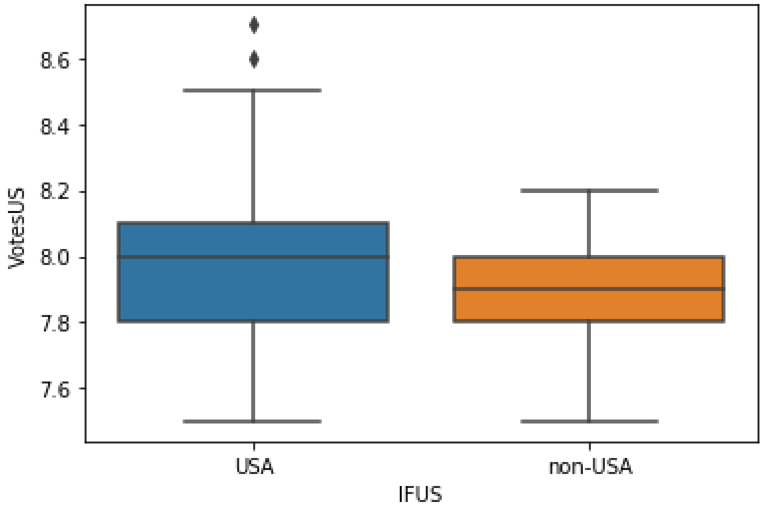


Inferences: Write your two inferences/observations below:

- Inference 1:
- Inference 2:

In [40]:
```
# Box plot - 2: VotesUS(y) vs IFUS(x)
sns.boxplot(y='VotesUS', x='IFUS', data=movies)
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2306908fa48>
```



`Inferences:` Write your two inferences/observations below:

- Inference 1:
- Inference 2:

- ### Subtask 3.5: Top 1000 Voters Vs Genres

You might have also observed the column `CVotes1000`. This column represents the top 1000 voters on IMDb and gives the count for the number of these voters who have voted for a particular movie. Let's see how these top 1000 voters have voted across the genres.

1. Sort the dataframe genre_top10 based on the value of `CVotes1000` in a descending order.
2. Make a seaborn barplot for `genre` vs `CVotes1000`.
3. Write your inferences. You can also try to relate it with the heatmaps you did in the previous subtasks.

In [41]:
```
# Sorting by CVotes1000
genre_top10.sort_values(by='CVotes1000', ascending=False)
```

Out[41]:

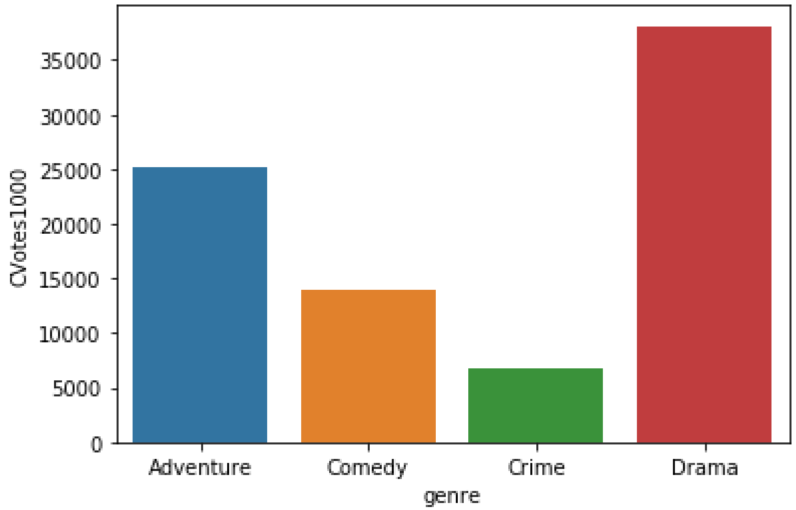
	CVotes10	CVotes09	CVotes08	CVotes07	CVotes06	CVotes05	CVotes04	CVotes03	CVotes
3	3404438.0	4935375.0	7107053.0	4319700.0	1529356.0	552312.0	235475.0	135126.0	94185
0	3594659.0	4014192.0	5262328.0	3281981.0	1212075.0	438970.0	183070.0	103318.0	69737
1	1383616.0	1774987.0	2506851.0	1591069.0	600287.0	226852.0	97469.0	56218.0	39391
2	574526.0	967118.0	1419495.0	821390.0	278391.0	98690.0	42271.0	24713.0	16985

4 rows × 46 columns

In [42]:

```
# Bar plot
sns.barplot(data = genre_top10, x = 'genre', y = 'CVotes1000')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x23069550a48>



`Inferences:` Write your inferences/observations here.

`Checkpoint 6:` The genre `Romance` seems to be most unpopular among the top 1000 voters.

With the above subtask, your assignment is over. In your free time, do explore the dataset further on your own and see what kind of other insights you can get across various other columns.