In [1]:
```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Import the numpy and pandas packages

import numpy as np
import pandas as pd
```

Task 1: Reading and Inspection

- ## Subtask 1.1: Import and read

Import and read the movie database. Store it in a variable called `movies`.

In [4]:
```
movies = pd.read_csv("D:/Envision_DataAnalyst/Projects/Python/IMDB/Movie_data.csv",
movies
```

Out[4]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	
...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	
5039	Color	NaN	43.0	43.0	NaN	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	
5041	Color	Daniel Hsia	14.0	100.0	0.0	
5042	Color	Jon Gunn	43.0	90.0	16.0	

5043 rows × 28 columns

- ## Subtask 1.2: Inspect the dataframe

Inspect the dataframe's columns, shapes, variable types etc.

In [6]:
```
print('Shape of the dataframe:',movies.shape)
print('Number of rows:', len(movies))
print('Number of columns:',len(movies.columns))
print('------------------------')
print("The column-wise data types are as follows,\n")
print(movies.dtypes)
```

```
Shape of the dataframe: (5043, 28)
Number of rows: 5043
Number of columns: 28
------------------------
The column-wise data types are as follows,

color                         object
director_name                 object
num_critic_for_reviews       float64
duration                     float64
director_facebook_likes      float64
actor_3_facebook_likes       float64
actor_2_name                  object
actor_1_facebook_likes       float64
gross                        float64
genres                        object
actor_1_name                  object
movie_title                   object
num_voted_users                int64
cast_total_facebook_likes      int64
actor_3_name                  object
facenumber_in_poster         float64
plot_keywords                 object
movie_imdb_link               object
num_user_for_reviews         float64
language                      object
country                       object
content_rating                object
budget                       float64
title_year                   float64
actor_2_facebook_likes       float64
imdb_score                   float64
aspect_ratio                 float64
movie_facebook_likes           int64
dtype: object
```

Task 2: Cleaning the Data

- ## Subtask 2.1: Inspect Null values

Find out the number of Null values in all the columns and rows. Also, find the percentage of Null values in each column. Round off the percentages upto two decimal places.

In [8]:
```
# Write your code for column-wise null count here
print(movies.isnull().sum())
```

```
color                        19
director_name               104
num_critic_for_reviews       50
duration                     15
director_facebook_likes     104
actor_3_facebook_likes       23
actor_2_name                 13
actor_1_facebook_likes        7
gross                       884
genres                        0
actor_1_name                  7
movie_title                   0
num_voted_users               0
cast_total_facebook_likes     0
actor_3_name                 23
facenumber_in_poster         13
plot_keywords               153
movie_imdb_link               0
num_user_for_reviews         21
language                     14
country                       5
content_rating              303
budget                      492
title_year                  108
actor_2_facebook_likes       13
imdb_score                    0
aspect_ratio                329
movie_facebook_likes          0
dtype: int64
```

In [9]:
```
# Write your code for row-wise null count here
print("Null in rows:\n",movies.isnull().sum(axis=1))
```

```
Null in rows:
 0        0
1        0
2        0
3        0
4       14
        ..
5038     4
5039     5
5040     4
5041     2
5042     0
Length: 5043, dtype: int64
```

In [10]:
```
# Write your code for column-wise null percentages here
print((movies.isnull().sum()/(movies.isnull().sum() + movies.notnull().sum()))* 100
```

```
color                          0.376760
director_name                  2.062265
num_critic_for_reviews         0.991473
duration                       0.297442
director_facebook_likes        2.062265
actor_3_facebook_likes         0.456078
actor_2_name                   0.257783
actor_1_facebook_likes         0.138806
gross                         17.529248
genres                         0.000000
actor_1_name                   0.138806
movie_title                    0.000000
num_voted_users                0.000000
cast_total_facebook_likes      0.000000
actor_3_name                   0.456078
facenumber_in_poster           0.257783
plot_keywords                  3.033908
movie_imdb_link                0.000000
num_user_for_reviews           0.416419
language                       0.277613
country                        0.099147
content_rating                 6.008328
budget                         9.756098
title_year                     2.141582
actor_2_facebook_likes         0.257783
imdb_score                     0.000000
aspect_ratio                   6.523895
movie_facebook_likes           0.000000
dtype: float64
```

- ## Subtask 2.2: Drop unecessary columns

For this assignment, you will mostly be analyzing the movies with respect to the ratings, gross collection, popularity of movies, etc. So many of the columns in this dataframe are not required. So it is advised to drop the following columns.

- color
- director_facebook_likes
- actor_1_facebook_likes
- actor_2_facebook_likes
- actor_3_facebook_likes
- actor_2_name
- cast_total_facebook_likes
- actor_3_name
- duration
- facenumber_in_poster
- content_rating
- country
- movie_imdb_link
- aspect_ratio

- plot_keywords

In [12]:
```
# Write your code for dropping the columns here. It is advised to keep inspecting t
movies1 = movies.drop(['color','director_facebook_likes','actor_1_facebook_likes',
                       'cast_total_facebook_likes','actor_3_name','duration','facenum
                       'aspect_ratio','plot_keywords'],axis='columns')
print('Number of columns in updated dataframe:',len(movies1.columns))
print("The updated dataframe is as follows:\n")
movies1
```

```
Number of columns in updated dataframe: 13
The updated dataframe is as follows:
```

Out[12]:

	director_name	num_critic_for_reviews	gross	genres	actor
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Joh
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	T
4	Doug Walker	NaN	NaN	Documentary	Dou
...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Er
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	N
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	
5042	Jon Gunn	43.0	85222.0	Documentary	Joh

5043 rows × 13 columns

- ## Subtask 2.3: Drop unecessary rows using columns with high Null percentages

Now, on inspection you might notice that some columns have large percentage (greater than 5%) of Null values. Drop all the rows which have Null values for such columns.

In [14]:

```
# movies_2 = movies_1.dropna(axis=0, subset=('gross','budget'))
# movies_2

# Write your code for dropping the rows here
```

```python
col=(movies1.isnull().sum() / (movies1.isnull().sum()+movies1.notnull().sum()))
col_null=col>0.05
print(col_null)

# Columns: gross and budget have null values greater than 5%
movies2 = movies1.dropna(axis=0, subset=('gross','budget'))

print('Number of rows after cleaning null rows:', len(movies2))
movies2
```

```
director_name             False
num_critic_for_reviews    False
gross                      True
genres                    False
actor_1_name              False
movie_title               False
num_voted_users           False
num_user_for_reviews      False
language                  False
budget                     True
title_year                False
imdb_score                False
movie_facebook_likes      False
dtype: bool
Number of rows after cleaning null rows: 3891
```

Out[14]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-F
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller
3	Christopher Nolan	813.0	448130642.0	Action|Thriller
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-F
...	...	...	...	..
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller
5034	Neill Dela Llana	35.0	70071.0	Thriller
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller
5037	Edward Burns	14.0	4584.0	Comedy|Drama
5042	Jon Gunn	43.0	85222.0	Documentary

3891 rows × 13 columns

- ## Subtask 2.4: Fill NaN values

You might notice that the `language` column has some NaN values. Here, on inspection, you will see that it is safe to replace all the missing values with `'English'`.

In [16]:

```
# Write your code for filling the NaN values in the 'language' column here
movies2['language'].fillna('English',inplace=True)
movies2
```

Out[16]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-F
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller
3	Christopher Nolan	813.0	448130642.0	Action|Thriller
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-F
...	...	...	...	..
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller
5034	Neill Dela Llana	35.0	70071.0	Thriller
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller
5037	Edward Burns	14.0	4584.0	Comedy|Drama
5042	Jon Gunn	43.0	85222.0	Documentary

3891 rows × 13 columns

- ## Subtask 2.5: Check the number of retained rows

You might notice that two of the columns viz. `num_critic_for_reviews` and `actor_1_name` have small percentages of NaN values left. You can let these columns as it is for now. Check the number and percentage of the rows retained after completing all the tasks above.

In [18]:

```
# Write your code for checking number of retained rows here
#print((movies2.isnull().sum() / (movies2.isnull().sum()+movies2.notnull().sum())*1

print("No of retained rows :", len(movies2))
print("No of total rows in source dataframe:", len(movies))
print("Percent of retained rows :", (len(movies2)/len(movies))*100,"%")
```

```
No of retained rows : 3891
No of total rows in source dataframe: 5043
Percent of retained rows : 77.15645449137418 %
```

Checkpoint 1: You might have noticed that we still have around `77%` of the rows!

Task 3: Data Analysis

- ## Subtask 3.1: Change the unit of columns

Convert the unit of the `budget` and `gross` columns from `$` to `million $`.

In [21]:
```
# Write your code for unit conversion here
movies3 = movies2
movies3['budget'] = movies3['budget'].apply(lambda x: x / 1000000)
movies3['gross'] = movies3['gross'].apply(lambda x: x / 1000000)
movies3
```

Out[21]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller
3	Christopher Nolan	813.0	448.130642	Action|Thriller
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi
...	...	...	...	...
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller
5034	Neill Dela Llana	35.0	0.070071	Thriller
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller
5037	Edward Burns	14.0	0.004584	Comedy|Drama
5042	Jon Gunn	43.0	0.085222	Documentary

3891 rows × 13 columns

- ## Subtask 3.2: Find the movies with highest 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. Plot `profit` (y-axis) vs `budget` (x- axis) and observe the outliers using the appropriate chart type.
  4. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`

In [23]:

```
# Write your code for creating the profit column here
movies3['profit'] = movies3['gross'] - movies3['budget']
movies3
```

Out[23]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller
3	Christopher Nolan	813.0	448.130642	Action|Thriller
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi
...	...	...	...	...
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller
5034	Neill Dela Llana	35.0	0.070071	Thriller
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller
5037	Edward Burns	14.0	0.004584	Comedy|Drama
5042	Jon Gunn	43.0	0.085222	Documentary

3891 rows × 14 columns

In [24]:

```
# Write your code for sorting the dataframe here
movies3 = movies3.sort_values(by='profit', ascending=False)
movies3
```

Out[24]:

	director_name	num_critic_for_reviews	gross	genre
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-I
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thrille
26	James Cameron	315.0	658.672302	Drama|Romanc
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-I
3080	Steven Spielberg	215.0	434.949459	Family|Sci-I
...	...	...	...	.
2334	Katsuhiro Ã□tomo	105.0	0.410388	Action|Adventure|Animation|Family|Sc Fi|Thrille
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantas
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|Wa
3859	Chan-wook Park	202.0	0.211667	Crime|Dram
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-I

3891 rows × 14 columns

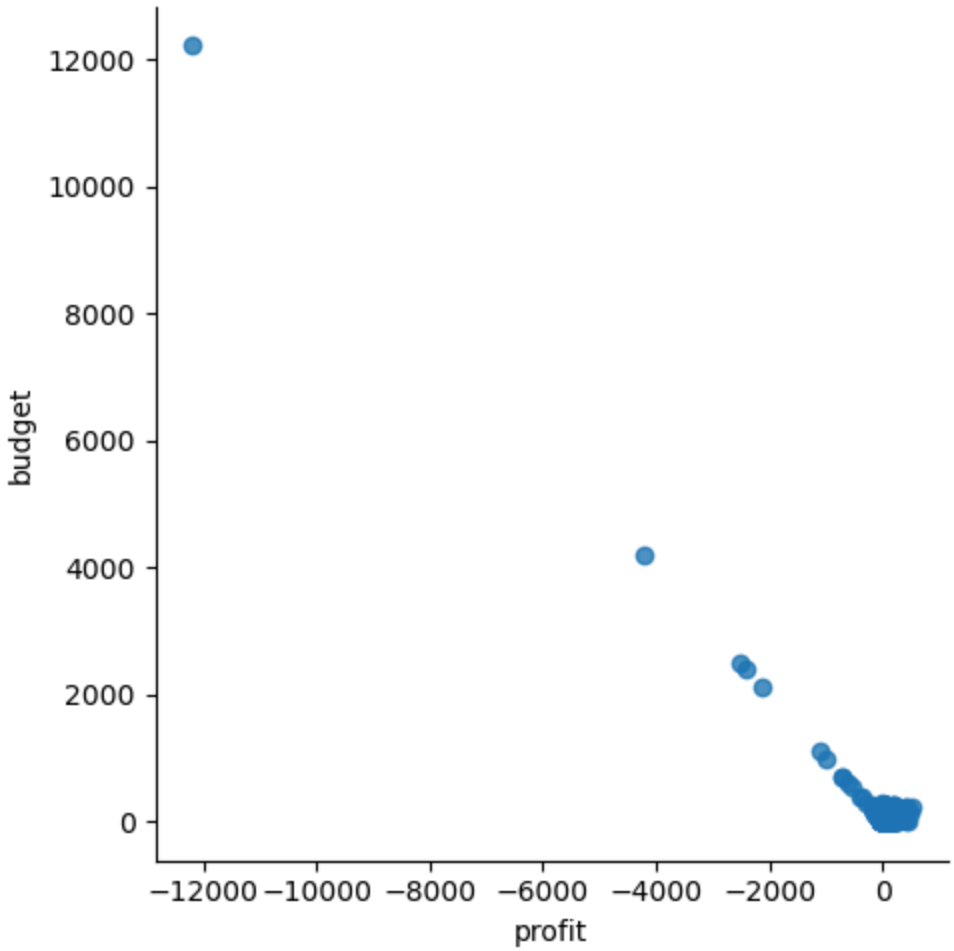
In [25]:

```
# Write code for profit vs budget plot here
import matplotlib.pyplot as plt
import seaborn as sns

sns.lmplot(x='profit',y='budget',data=movies3,fit_reg=False)
```

Out[25]:

```
<seaborn.axisgrid.FacetGrid at 0x2671c7c19d0>
```



In [26]:
```
# Write your code to get the top 10 profiting movies here
top10 = movies3.nlargest(10, 'profit')
top10
```

Out[26]:

	director_name	num_critic_for_reviews	gross	g
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|T
26	James Cameron	315.0	658.672302	Drama|Ron
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|
3080	Steven Spielberg	215.0	434.949459	Family|
794	Joss Whedon	703.0	623.279547	Action|Adventure|
17	Joss Whedon	703.0	623.279547	Action|Adventure|
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|M
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|T

- ## Subtask 3.3: Drop duplicate values

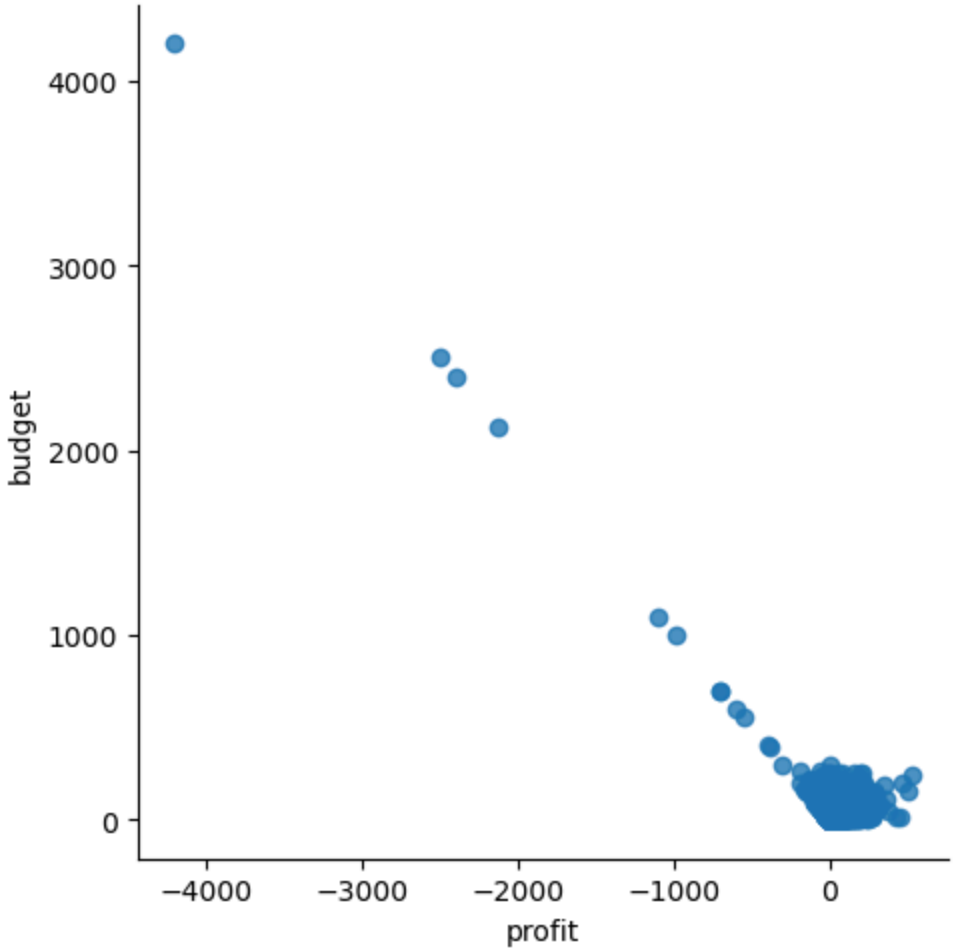
After you found out the top 10 profiting movies, you might have noticed a duplicate value. So, it seems like the dataframe has duplicate values as well. Drop the duplicate values from the dataframe and repeat `Subtask 3.2`. Note that the same `movie_title` can be there in different languages.

In [28]:

```
# Write your code for dropping duplicate values here
movies4 = movies2
movies4["language"].fillna("English",inplace=True)
movies4.drop_duplicates(subset ="movie_title", keep = False, inplace = True)
sns.lmplot(x='profit',y='budget',data=movies4,fit_reg=False)
movies4
top10_1 = movies4.nlargest(10, 'profit')
top10_1
```

Out[28]:

	director_name	num_critic_for_reviews	gross	g
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|T
26	James Cameron	315.0	658.672302	Drama|Ro
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|
3080	Steven Spielberg	215.0	434.949459	Family|
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|M
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|T
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|T
812	Tim Miller	579.0	363.024263	Action|Adventure|Comedy|Romance|



In [29]:
```
# Write code for repeating subtask 2 here
movies4['profit'] = movies4['gross'] - movies4['budget']
movies4 = movies4.sort_values(by='profit', ascending=False)
movies4
```

Out[29]:

	director_name	num_critic_for_reviews	gross	genre
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-I
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thrille
26	James Cameron	315.0	658.672302	Drama|Romanc
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-I
3080	Steven Spielberg	215.0	434.949459	Family|Sci-I
...	...	...	...	.
3423	Katsuhiro Ã□tomo	150.0	0.439162	Action|Animation|Sci-I
2334	Katsuhiro Ã□tomo	105.0	0.410388	Action|Adventure|Animation|Family|Sc Fi|Thrille
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantas
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|Wa
3859	Chan-wook Park	202.0	0.211667	Crime|Dram

3693 rows × 14 columns

Checkpoint 2: You might spot two movies directed by `James Cameron` in the list.

- ## Subtask 3.4: Find IMDb Top 250

  1. Create a new dataframe `IMDb_Top_250` and store the top 250 movies with the highest IMDb Rating (corresponding to the column: `imdb_score`). Also make sure that for all of these movies, the `num_voted_users` is greater than 25,000.

Also add a `Rank` column containing the values 1 to 250 indicating the ranks of the corresponding films. 2. Extract all the movies in the `IMDb_Top_250` dataframe which are not in the English language and store them in a new dataframe named `Top_Foreign_Lang_Film`.

In [32]:
```
# Write your code for extracting the top 250 movies as per the IMDb score here. Mak
# and name that dataframe as 'IMDb_Top_250'
IMDb_Top_250 = movies4[movies4['num_voted_users'] > 25000]
IMDb_Top_250 = IMDb_Top_250.sort_values(by='imdb_score', ascending=False).head(250)
IMDb_Top_250['rank'] = IMDb_Top_250['imdb_score'].rank(method='min', ascending=Fals
IMDb_Top_250
```

Out[32]:

	director_name	num_critic_for_reviews	gross	
1937	Frank Darabont	199.0	28.341469	
3466	Francis Ford Coppola	208.0	134.821952	
2837	Francis Ford Coppola	149.0	57.300000	
66	Christopher Nolan	645.0	533.316061	Action|Crime|
339	Peter Jackson	328.0	377.019252	Action|Adventure|D
...	...	...	...	
639	Michael Mann	209.0	28.965197	Biography|
2547	Wes Anderson	487.0	45.507053	Adventure|Comedy|Dr
7	Nathan Greno	324.0	200.807262	Adventure|Animation|Comedy|Family|F
1892	George P. Cosmatos	84.0	56.505065	Action|Biography|Drama|History|Rom
2133	Trey Parker	163.0	52.008288	Animation|Comedy|Fa

250 rows × 15 columns

In [33]:
```
# Write your code to extract top foreign language films from 'IMDb_Top_250' here
Top_Foreign_Lang_Film = IMDb_Top_250[IMDb_Top_250['language'] != 'English']
Top_Foreign_Lang_Film
```

Out[33]:

	director_name	num_critic_for_reviews	gross	
4498	Sergio Leone	181.0	6.100000	
4747	Akira Kurosawa	153.0	0.269061	Action|A
4029	Fernando Meirelles	214.0	7.563397	
2373	Hayao Miyazaki	246.0	10.049886	Adventure|Animatio
4921	Majid Majidi	46.0	0.925402	
4259	Florian Henckel von Donnersmarck	215.0	11.284657	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Ar
1329	S.S. Rajamouli	44.0	6.498000	Action|Adventure|Dra
4659	Asghar Farhadi	354.0	7.098492	
4105	Chan-wook Park	305.0	2.181290	Drama
1298	Jean-Pierre Jeunet	242.0	33.201661	C
2970	Wolfgang Petersen	96.0	11.433134	Adventure|Dr
2829	Oliver Hirschbiegel	192.0	5.501940	Biography|Dr
2734	Fritz Lang	260.0	0.026435	
4033	Thomas Vinterberg	349.0	0.610968	
2551	Guillermo del Toro	406.0	37.623143	Dra
2047	Hayao Miyazaki	212.0	4.710455	Adventure|Animatio
4000	Juan JosÃ© Campanella	262.0	20.167424	Drama

	director_name	num_critic_for_reviews	gross	
3550	Denis Villeneuve	226.0	6.857096	Dra
4461	Thomas Vinterberg	98.0	1.647780	
4267	Alejandro G. IÃ±Ã¡rritu	157.0	5.383834	
2914	Je-kyu Kang	86.0	1.110186	A
2830	Alejandro AmenÃ¡bar	157.0	2.086345	Biography|
3423	Katsuhiro Ã□tomo	150.0	0.439162	Action|
3553	JosÃ© Padilha	142.0	0.008060	Action|Crim
4144	Walter Salles	71.0	5.595428	
4897	Sergio Leone	122.0	3.500000	Actior
3344	Karan Johar	210.0	4.018695	Adventu
3456	Vincent Paronnaud	242.0	4.443403	Animation|Biogr
4284	Ari Folman	231.0	2.283276	Animation|Biography|Documentary|Dr
4640	Cristian Mungiu	233.0	1.185783	
3264	Michael Haneke	447.0	0.225377	
2605	Ang Lee	287.0	128.067808	Action|
4415	FabiÃ¡n Bielinsky	94.0	1.221261	Crim
2863	Clint Eastwood	251.0	13.753931	Dr

	director_name	num_critic_for_reviews	gross	
3510	Yash Chopra	29.0	2.921738	Drama|
3677	Christophe Barratier	112.0	3.629758	

Checkpoint 3: Can you spot `Veer-Zaara` in the dataframe?

- ## Subtask 3.5: Find the best directors

  1. Group the dataframe using the `director_name` column.
  2. Find out the top 10 directors for whom the mean of `imdb_score` is the highest and store them in a new dataframe `top10director`. Incase of a tie in IMDb score between two directors, sort them alphabetically.

In [36]:
```
# Write your code for extracting the top 10 directors here
#groupedset = movies4.groupby('director_name').mean()
# List the columns you want to calculate the mean for
columns_to_group = [
    'num_critic_for_reviews', 'gross', 'num_voted_users',
    'num_user_for_reviews', 'budget', 'title_year', 'imdb_score',
    'movie_facebook_likes', 'profit'
]
# Group by 'director_name' and calculate the mean for the specified columns
groupedset = movies4.groupby('director_name')[columns_to_group].mean()
groupedset

top10director = groupedset.sort_values(by=['imdb_score','director_name'], ascending
top10director
```

Out[36]:

director_name	num_critic_for_reviews	gross	num_voted_users	num_user_for_reviews
Charles Chaplin	120.00	0.163245	1.430860e+05	211.000000
Tony Kaye	162.00	6.712241	7.824370e+05	1420.000000
Alfred Hitchcock	290.00	32.000000	4.224320e+05	1040.000000
Damien Chazelle	535.00	13.092000	3.991380e+05	731.000000
Florian Henckel von Donnersmarck	215.00	11.284657	2.593790e+05	407.000000
Majid Majidi	46.00	0.925402	2.788200e+04	130.000000
Ron Fricke	115.00	2.601847	2.245700e+04	69.000000
Sergio Leone	138.00	4.966667	2.906917e+05	503.333333
Christopher Nolan	511.25	226.653447	1.013285e+06	2424.875000
Asghar Farhadi	354.00	7.098492	1.518120e+05	264.000000

Checkpoint 4: No surprises that `Damien Chazelle` (director of Whiplash and La La Land) is in this list.

- ## Subtask 3.6: Find popular genres

You might have noticed the `genres` column in the dataframe with all the genres of the movies seperated by a pipe (`|`). Out of all the movie genres, the first two are most significant for any film.

1. Extract the first two genres from the `genres` column and store them in two new columns: `genre_1` and `genre_2` . Some of the movies might have only one genre. In such cases, extract the single genre into both the columns, i.e. for such movies the `genre_2` will be the same as `genre_1` .
2. Group the dataframe using `genre_1` as the primary column and `genre_2` as the secondary column.
3. Find out the 5 most popular combo of genres by finding the mean of the gross values using the `gross` column and store them in a new dataframe named `PopGenre` .

In [39]:

```
# Write your code for extracting the first two genres of each movie here
p_genres=movies4
```

```
p_genres[['genre_1', 'genre_2','genres3']] = p_genres['genres'].str.split('|', expa
# For rows where there is only one genre, set 'genre_2' equal to 'genre_1'
p_genres['genre_2'] = p_genres['genre_2'].fillna(p_genres['genre_1'])
p_genres
```

Out[39]:

	director_name	num_critic_for_reviews	gross	genre
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-I
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thrille
26	James Cameron	315.0	658.672302	Drama|Romanc
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-I
3080	Steven Spielberg	215.0	434.949459	Family|Sci-I
...	...	...	...	
3423	Katsuhiro Ã□tomo	150.0	0.439162	Action|Animation|Sci-I
2334	Katsuhiro Ã□tomo	105.0	0.410388	Action|Adventure|Animation|Family|Sc Fi|Thrille
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantas
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|Wa
3859	Chan-wook Park	202.0	0.211667	Crime|Dram

3693 rows × 17 columns

In [40]:

```
# Write your code for grouping the dataframe here
movies_by_segment = movies4.groupby(['genre_1', 'genre_2']).agg(
    num_critic_for_reviews=('num_critic_for_reviews', 'mean'),
    gross=('gross', 'mean'),
    num_voted_users=('num_voted_users', 'mean'),
    num_user_for_reviews=('num_user_for_reviews', 'mean'),
    budget=('budget', 'mean'),
    title_year=('title_year', 'mean'),
    imdb_score=('imdb_score', 'mean'),
    movie_facebook_likes=('movie_facebook_likes', 'mean'),
    profit=('profit', 'mean')
).reset_index()
```

```
movies_by_segment
```

Out[40]:

	genre_1	genre_2	num_critic_for_reviews	gross	num_voted_users	num_user_f
0	Action	Action	171.166667	59.520907	203309.500000	
1	Action	Adventure	215.543307	105.644808	175651.440945	
2	Action	Animation	184.625000	92.680515	78015.375000	
3	Action	Biography	165.857143	44.355422	86330.928571	
4	Action	Comedy	136.370079	52.307861	80718.448819	
...	...	...	...	...	...	
97	Romance	Sci-Fi	413.000000	62.453315	200035.000000	
98	Sci-Fi	Sci-Fi	8.000000	0.018195	336.000000	
99	Sci-Fi	Thriller	200.000000	29.793790	169014.428571	
100	Thriller	Thriller	16.333333	0.040513	784.666667	
101	Western	Western	74.000000	15.914589	181034.333333	

102 rows × 11 columns

In [41]:

```
# Write your code for getting the 5 most popular combo of genres here
PopGenre = movies_by_segment.sort_values('gross', ascending=False).head(5)
PopGenre
```

Out[41]:

	genre_1	genre_2	num_critic_for_reviews	gross	num_voted_users	num_user_f
82	Family	Sci-Fi	215.000000	434.949459	281842.000000	
27	Adventure	Sci-Fi	344.375000	228.627758	290322.500000	
17	Adventure	Animation	170.061404	115.410520	137771.675439	
1	Action	Adventure	215.543307	105.644808	175651.440945	
24	Adventure	Fantasy	284.555556	102.406985	197128.222222	

Checkpoint 5: Well, as it turns out. `Family + Sci-Fi` is the most popular combo of genres out there!

- ## Subtask 3.7: Find the critic-favorite and audience-favorite actors

  1. Create three new dataframes namely, `Meryl_Streep`, `Leo_Caprio`, and `Brad_Pitt` which contain the movies in which the actors: 'Meryl Streep',

'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the `actor_1_name` column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.

2. Append the rows of all these dataframes and store them in a new dataframe named `Combined`.
3. Group the combined dataframe using the `actor_1_name` column.
4. Find the mean of the `num_critic_for_reviews` and `num_users_for_review` and identify the actors which have the highest mean.
5. Observe the change in number of voted users over decades using a bar chart. Create a column called `decade` which represents the decade to which every movie belongs to. For example, the `title_year` year 1923, 1925 should be stored as 1920s. Sort the dataframe based on the column `decade`, group it by `decade` and find the sum of users voted in each decade. Store this in a new data frame called `df_by_decade`.

In [44]:
```
# Write your code for creating three new dataframes here
# Include all movies in which Meryl_Streep is the lead
Meryl_Streep =  movies4[movies4['actor_1_name'] == 'Meryl Streep']
Meryl_Streep
```

Out[44]:

	director_name	num_critic_for_reviews	gross	genres	acto
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Me
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Me
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Me
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Me
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Me
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Me
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Me
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Me
1106	Curtis Hanson	42.0	46.815748	Action|Adventure|Crime|Thriller	Me
1674	Carl Franklin	64.0	23.209440	Drama	Me
1483	Robert Redford	227.0	14.998070	Drama|Thriller|War	Me

In [46]:

```
# Include all movies in which Leo_Caprio is the lead
Leo_Caprio = movies4[movies4['actor_1_name'] == 'Leonardo DiCaprio']
Leo_Caprio
```

Out[46]:

	director_name	num_critic_for_reviews	gross	genres	ac
26	James Cameron	315.0	658.672302	Drama|Romance	
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	
911	Steven Spielberg	194.0	164.435221	Biography|Crime|Drama	
296	Quentin Tarantino	765.0	162.804648	Drama|Western	
179	Alejandro G. IÃ±Ã¡rritu	556.0	183.635922	Adventure|Drama|Thriller|Western	
452	Martin Scorsese	490.0	127.968405	Mystery|Thriller	
361	Martin Scorsese	352.0	132.373442	Crime|Drama|Thriller	
2757	Baz Luhrmann	106.0	46.338728	Drama|Romance	
1422	Randall Wallace	83.0	56.876365	Action|Adventure	
308	Martin Scorsese	606.0	116.866727	Biography|Comedy|Crime|Drama	
1453	Clint Eastwood	392.0	37.304950	Biography|Crime|Drama	
257	Martin Scorsese	267.0	102.608827	Biography|Drama	
2067	Jerry Zaks	45.0	12.782508	Drama	
990	Danny Boyle	118.0	39.778599	Adventure|Drama|Thriller	
1114	Sam Mendes	323.0	22.877808	Drama|Romance	
1560	Sam Raimi	63.0	18.636537	Action|Thriller|Western	
326	Martin Scorsese	233.0	77.679638	Crime|Drama	
641	Ridley Scott	238.0	39.380442	Action|Drama|Thriller	
307	Edward Zwick	166.0	57.366262	Adventure|Drama|Thriller	

In [52]:
```
# Include all movies in which Brad_Pitt is the lead
Brad_Pitt = movies4[movies4['actor_1_name'] == 'Brad Pitt']
Brad_Pitt
```

Out[52]:

	director_name	num_critic_for_reviews	gross	
400	Steven Soderbergh	186.0	183.405771	
255	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Ro
940	Neil Jordan	120.0	105.264608	Drama|
470	David Ayer	406.0	85.707116	Acti
254	Steven Soderbergh	198.0	125.531634	
2204	Alejandro G. IÃ±Ã¡rritu	285.0	34.300771	
2682	Andrew Dominik	414.0	14.938570	
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Ro
2333	Angelina Jolie Pitt	131.0	0.531009	Dı
1490	Terrence Malick	584.0	13.303319	
101	David Fincher	362.0	127.490802	Drama|Far
683	David Fincher	315.0	37.023395	
1722	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|H
611	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Dran
792	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|
147	Wolfgang Petersen	220.0	133.228348	
382	Tony Scott	142.0	0.026871	Actior

In [54]:
```
# Write your code for combining the three dataframes here
Combined = pd.concat([Meryl_Streep, Leo_Caprio, Brad_Pitt], ignore_index=True)
Combined
```

Out[54]:

	director_name	num_critic_for_reviews	gross	
0	David Frankel	208.0	124.732962	Comedy|Dran
1	Sydney Pollack	66.0	87.100000	Biography|Dran
2	Nora Ephron	252.0	94.125426	Biography|Dran
3	David Frankel	234.0	63.536011	Comedy|Dran
4	Nancy Meyers	187.0	112.703470	Comedy|Dran
5	Phyllida Lloyd	331.0	29.959436	Biography|Dr
6	Stephen Daldry	174.0	41.597830	Dran
7	Robert Altman	211.0	20.338609	Comedy|D
8	Curtis Hanson	42.0	46.815748	Action|Adventure|C
9	Carl Franklin	64.0	23.209440	
10	Robert Redford	227.0	14.998070	Drama
11	James Cameron	315.0	658.672302	Dran
12	Christopher Nolan	642.0	292.568851	Action|Adventure|S
13	Steven Spielberg	194.0	164.435221	Biography|C
14	Quentin Tarantino	765.0	162.804648	Dra
15	Alejandro G. IÃ±Ã¡rritu	556.0	183.635922	Adventure|Drama|Thri
16	Martin Scorsese	490.0	127.968405	My
17	Martin Scorsese	352.0	132.373442	Crime|Dr
18	Baz Luhrmann	106.0	46.338728	Dran

	director_name	num_critic_for_reviews	gross	
19	Randall Wallace	83.0	56.876365	Actio
20	Martin Scorsese	606.0	116.866727	Biography|Comedy|C
21	Clint Eastwood	392.0	37.304950	Biography|C
22	Martin Scorsese	267.0	102.608827	Biogr
23	Jerry Zaks	45.0	12.782508	
24	Danny Boyle	118.0	39.778599	Adventure|Dr
25	Sam Mendes	323.0	22.877808	Dran
26	Sam Raimi	63.0	18.636537	Action|Thri
27	Martin Scorsese	233.0	77.679638	C
28	Ridley Scott	238.0	39.380442	Action|Dr
29	Edward Zwick	166.0	57.366262	Adventure|Dr
30	Steven Soderbergh	186.0	183.405771	C
31	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Rom
32	Neil Jordan	120.0	105.264608	Drama|Far
33	David Ayer	406.0	85.707116	Action
34	Steven Soderbergh	198.0	125.531634	C
35	Alejandro G. IÃ±Ã¡rritu	285.0	34.300771	
36	Andrew Dominik	414.0	14.938570	C

	director_name	num_critic_for_reviews	gross	
37	Tony Scott	122.0	12.281500	Action|Crime|Drama|Rom
38	Angelina Jolie Pitt	131.0	0.531009	Dran
39	Terrence Malick	584.0	13.303319	Dr
40	David Fincher	362.0	127.490802	Drama|Fanta
41	David Fincher	315.0	37.023395	
42	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|Hist
43	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Drama|
44	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|Far
45	Wolfgang Petersen	220.0	133.228348	
46	Tony Scott	142.0	0.026871	Action|C

In [66]:

```
# Write your code for grouping the combined dataframe here
group_columns =['num_critic_for_reviews', 'gross', 'num_voted_users',
    'num_user_for_reviews', 'budget', 'title_year', 'imdb_score',
    'movie_facebook_likes', 'profit'
]
grouped_df = Combined.groupby('actor_1_name')[group_columns].mean()
grouped_df
```

Out[66]:

actor_1_name	num_critic_for_reviews	gross	num_voted_users	num_user_for_reviews
Brad Pitt	245.000000	66.321449	283583.823529	742.352941
Leonardo DiCaprio	313.368421	123.734536	450278.473684	931.473684
Meryl Streep	181.454545	59.919727	73545.545455	297.181818

In [80]:
```
# Write the code for finding the mean of critic reviews and audience reviews here
# grouped_df = Combined.groupby('actor_1_name')[group_columns].mean()
mean_critic_reviews = grouped_df.sort_values('num_critic_for_reviews', ascending=Fa
mean_audience_reviews =grouped_df.sort_values('num_user_for_reviews',ascending=Fals
mean_critic_reviews
mean_audience_reviews
```

Out[80]:

actor_1_name	num_critic_for_reviews	gross	num_voted_users	num_user_for_reviews
Leonardo DiCaprio	313.368421	123.734536	450278.473684	931.473684

Checkpoint 6: `Leonardo` has aced both the lists!

In [84]:
```
# Write the code for calculating decade here
# Divide the year by 10, then multiply by 10 to get the decade
movies5 = movies4
movies5['decade'] = (movies5['title_year'] // 10) * 10
movies5['decade'] = movies5['decade'].astype(str) + 's'
movies5
```

Out[84]:

	director_name	num_critic_for_reviews	gross	genre
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-I
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thrille
26	James Cameron	315.0	658.672302	Drama|Romanc
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-I
3080	Steven Spielberg	215.0	434.949459	Family|Sci-I
...	...	...	...	
3423	Katsuhiro Ã□tomo	150.0	0.439162	Action|Animation|Sci-I
2334	Katsuhiro Ã□tomo	105.0	0.410388	Action|Adventure|Animation|Family|Sc Fi|Thrille
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantas
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|Wa
3859	Chan-wook Park	202.0	0.211667	Crime|Dram

3693 rows × 18 columns

In [96]:

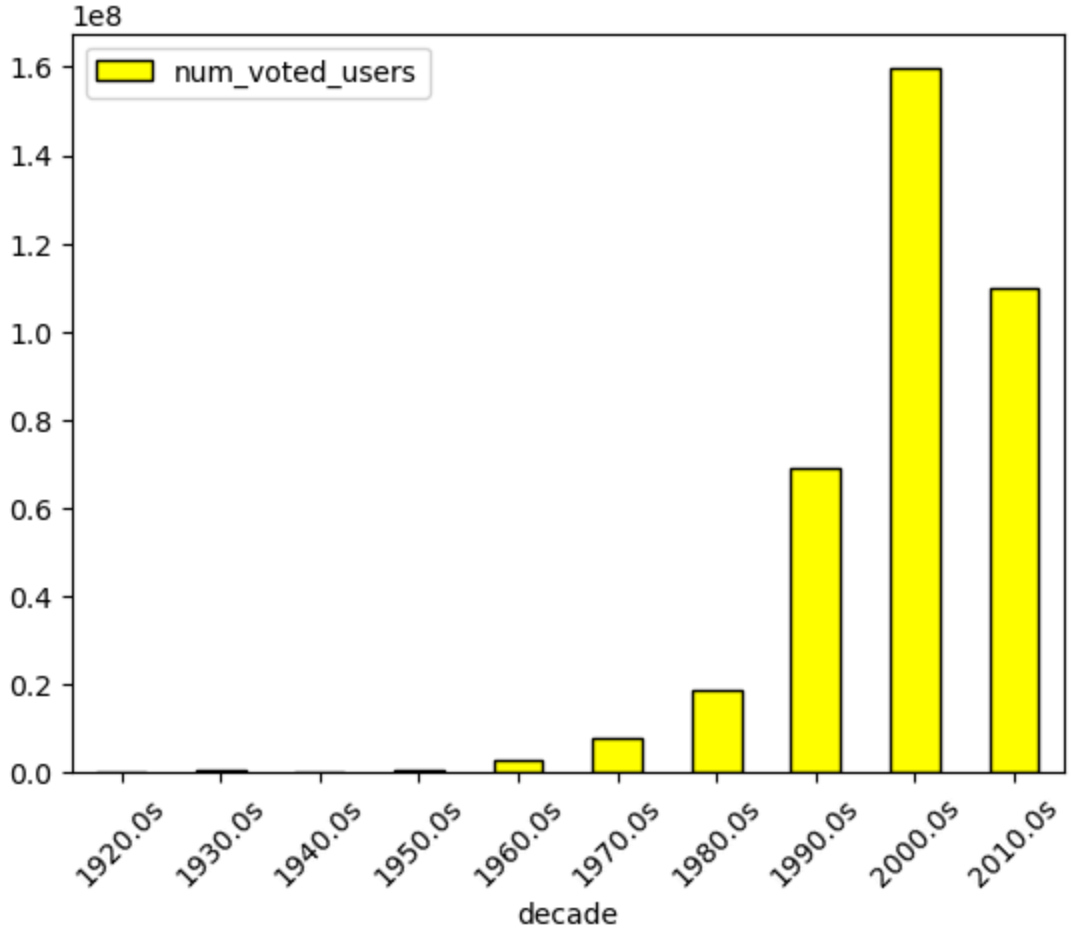
```
# Write your code for creating the data frame df_by_decade here
group_columns =['num_critic_for_reviews', 'gross', 'num_voted_users',
    'num_user_for_reviews', 'budget', 'title_year', 'imdb_score',
    'movie_facebook_likes', 'profit'
]
df_by_decade = movies5.groupby('decade')[group_columns].sum()
df_by_decade
```

Out[96]:

decade	num_critic_for_reviews	gross	num_voted_users	num_user_for_reviews	
1920.0s	297.0	5.834435	116392	485.0	
1930.0s	766.0	411.246620	804839	1849.0	1
1940.0s	464.0	207.713927	230838	928.0	2
1950.0s	764.0	147.069061	678336	1854.0	2
1960.0s	3094.0	1050.396801	2983442	9424.0	19
1970.0s	5344.0	3326.041414	8015561	17540.0	48
1980.0s	18483.0	10331.503185	18596981	47508.0	427
1990.0s	60205.0	34169.119186	69182961	205243.0	2767
2000.0s	252307.0	76746.891175	159582998	608673.0	7393
2010.0s	242181.0	56005.187672	110078938	273252.0	4478

In [130…

```
# Write your code for plotting number of voted users vs decade
barplot = df_by_decade.plot.bar( y='num_voted_users', rot=45, ec='black', color='ye
```



In [ ]: