###Drive Mounting

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

###Importing

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
movies = pd.read_csv('/content/drive/MyDrive/STUDY2/DATA ANALYSIS LAB/LABCYCLE/DATASETS/movies _9thQuestion.csv')
movies
```

	Title	US Gross	Worldwide Gross	US DVD Sales	Production Budget	Release Date	MPAA Rating	Running Time (min)	Distributor	Source	Major Genre	Creative Type	Director	Rotten Tomatoes Rating	IMDB Rating
0	The Land Girls	146083	146083	NaN	8000000.0	12-Jun-98	R	NaN	Gramercy	NaN	NaN	NaN	NaN	NaN	6.1
1	First Love, Last Rites	10876	10876	NaN	300000.0	7-Aug-98	R	NaN	Strand	NaN	Drama	NaN	NaN	NaN	6.9
2	I Married a Strange Person	203134	203134	NaN	250000.0	28-Aug-98	NaN	NaN	Lionsgate	NaN	Comedy	NaN	NaN	NaN	6.8
3	Let's Talk About Sex	373615	373615	NaN	300000.0	11-Sep-98	NaN	NaN	Fine Line	NaN	Comedy	NaN	NaN	13.0	NaN
4	Slam	1009819	1087521	NaN	1000000.0	9-Oct-98	R	NaN	Trimark	Original Screenplay	Drama	Contemporary Fiction	NaN	62.0	3.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3196	Zack and Miri Make a Porno	31452765	36851125	21240321.0	24000000.0	31-Oct-08	R	101.0	Weinstein Co.	Original Screenplay	Comedy	Contemporary Fiction	Kevin Smith	65.0	7.0
3197	Zodiac	33080084	83080084	20983030.0	85000000.0	2-Mar-07	R	157.0	Paramount Pictures	Based on Book/Short Story	Thriller/Suspense	Dramatization	David Fincher	89.0	NaN
3198	Zoom	11989328	12506188	6679409.0	35000000.0	11-Aug-06	PG	NaN	Sony Pictures	Based on Comic/Graphic Novel	Adventure	Super Hero	Peter Hewitt	3.0	3.4
3199	The Legend of Zorro	45575336	141475336	NaN	80000000.0	28-Oct-05	PG	129.0	Sony Pictures	Remake	Adventure	Historical Fiction	Martin Campbell	26.0	5.7
3200	The Mask of Zorro	93828745	233700000	NaN	65000000.0	17-Jul-98	PG-13	136.0	Sony Pictures	Remake	Adventure	Historical Fiction	Martin Campbell	82.0	6.7

3201 rows × 16 columns

###Cleaning

```
movies = movies.dropna(subset=['Title', 'US Gross', 'Worldwide Gross', 'Release Date', 'MPAA Rating','Distributor', 'Source', 'Major Genre',
                               'Creative Type', 'Director',])
```

```
def modify_release_year(year):
    if year > 23:
        return 1900 + year
    else:
        return 2000 + year

movies['Release Year'] = movies['Release Date'].apply(lambda x: x.split('-')[2].strip())

num_cols=['US DVD Sales','US Gross','Production Budget','Running Time (min)', 'Rotten Tomatoes Rating', 'IMDB Rating','IMDB Votes','Worldwide Gro
for num in num_cols:
    movies[num].fillna(movies[num].mean(),inplace=True)
    movies[num] = movies[num].astype(int)

movies['Release Year'] = movies['Release Year'].apply(modify_release_year)
movies
```

]:
```
<ipython-input-5-6ef33899a93e>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['Release Year'] = movies['Release Date'].apply(lambda x: x.split('-')[2].strip())
<ipython-input-5-6ef33899a93e>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies[num].fillna(movies[num].mean(),inplace=True)
<ipython-input-5-6ef33899a93e>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies[num] = movies[num].astype(int)
<ipython-input-5-6ef33899a93e>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['Release Year'] = movies['Release Year'].apply(modify_release_year)
```

]:

	Title	US Gross	Worldwide Gross	US DVD Sales	Production Budget	Release Date	MPAA Rating	Running Time (min)	Distributor	Source	Major Genre	Creative Type	Director	Rotten Tomatoes Rating	IMDB Rating
20	Twelve Monkeys	57141459	168841459	42195480	29000000	27-Dec-95	R	114	Universal	Based on Short Film	Drama	Science Fiction	Terry Gilliam	55	8
28	Twin Falls Idaho	985341	1027228	42195480	500000	30-Jul-99	R	114	Sony Pictures Classics	Original Screenplay	Drama	Contemporary Fiction	Michael Polish	77	7
34	Four Rooms	4301000	4301000	42195480	4000000	25-Dec-95	R	114	Miramax	Original Screenplay	Comedy	Contemporary Fiction	Robert Rodriguez	14	6
36	Four Weddings and a Funeral	52700832	242895809	42195480	4500000	9-Mar-94	R	114	Gramercy	Original Screenplay	Romantic Comedy	Contemporary Fiction	Mike Newell	96	7
41	The Abyss	54243125	54243125	42195480	70000000	9-Aug-89	PG-13	114	20th Century Fox	Original Screenplay	Action	Science Fiction	James Cameron	88	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3196	Zack and Miri Make a Porno	31452765	36851125	21240321	24000000	31-Oct-08	R	101	Weinstein Co.	Original Screenplay	Comedy	Contemporary Fiction	Kevin Smith	65	7
3197	Zodiac	33080084	83080084	20983030	85000000	2-Mar-07	R	157	Paramount Pictures	Based on Book/Short Story	Thriller/Suspense	Dramatization	David Fincher	89	6
3198	Zoom	11989328	12506188	6679409	35000000	11-Aug-06	PG	114	Sony Pictures	Based on Comic/Graphic Novel	Adventure	Super Hero	Peter Hewitt	3	3
3199	The Legend of Zorro	45575336	141475336	42195480	80000000	28-Oct-05	PG	129	Sony Pictures	Remake	Adventure	Historical Fiction	Martin Campbell	26	5
3200	The Mask of Zorro	93828745	233700000	42195480	65000000	17-Jul-98	PG-13	136	Sony Pictures	Remake	Adventure	Historical Fiction	Martin Campbell	82	6

1455 rows × 17 columns

###a. Find number of movies released under each genre in each year.

]:
```
movies_in_genre = movies.groupby(['Release Year','Major Genre'])['Title'].nunique()
movies_in_genre.reset_index()
```

]:

	Release Year	Major Genre	Title
0	1939	Drama	1
1	1939	Musical	1
2	1960	Western	1
3	1965	Drama	1
4	1969	Western	1
...	...	...	...
199	2010	Comedy	5
200	2010	Drama	8
201	2010	Horror	2
202	2010	Romantic Comedy	1
203	2010	Thriller/Suspense	5

204 rows × 3 columns

###b. Find movies with loss every year for each distributor.

]:
```
loss = movies[movies['Production Budget'] > movies['Worldwide Gross']]
loss
film_loss = loss.groupby(['Release Year','Distributor','Title']).agg(list).reset_index()
film_loss.pivot_table(index=['Release Year','Distributor','Title'])
```

]:
```
<ipython-input-12-d9d7667805c1>:4: FutureWarning: pivot_table dropped a column because it failed to aggregate. This behavior is deprecated and
will raise in a future version of pandas. Select only the columns that can be aggregated.
  film_loss.pivot_table(index=['Release Year','Distributor','Title'])
```

]:

Release Year	Distributor	Title
1960	United Artists	The Alamo
1980	Universal	The Island
1985	Universal	Brazil
1986	20th Century Fox	Highlander
1989	20th Century Fox	The Abyss
...	...	...
2010	Universal	Green Zone
		Scott Pilgrim vs. The World
		The Wolf Man
	Warner Bros.	The Losers
		The Town

398 rows × 0 columns

###c. Find the Directors who directed for each creative type with IMDB rating above 6.

]:
```
imdb_6 = movies[movies['IMDB Rating']>6]
#imdb_6
dir_6 = movies.groupby(['Director','Creative Type'])['IMDB Rating'].mean().reset_index()
dir_6 = dir_6.rename(columns={'IMDB Rating':'AVG_IMDB_RATING'})
dir_6.pivot_table(index=['Director','Creative Type'])
```

]:

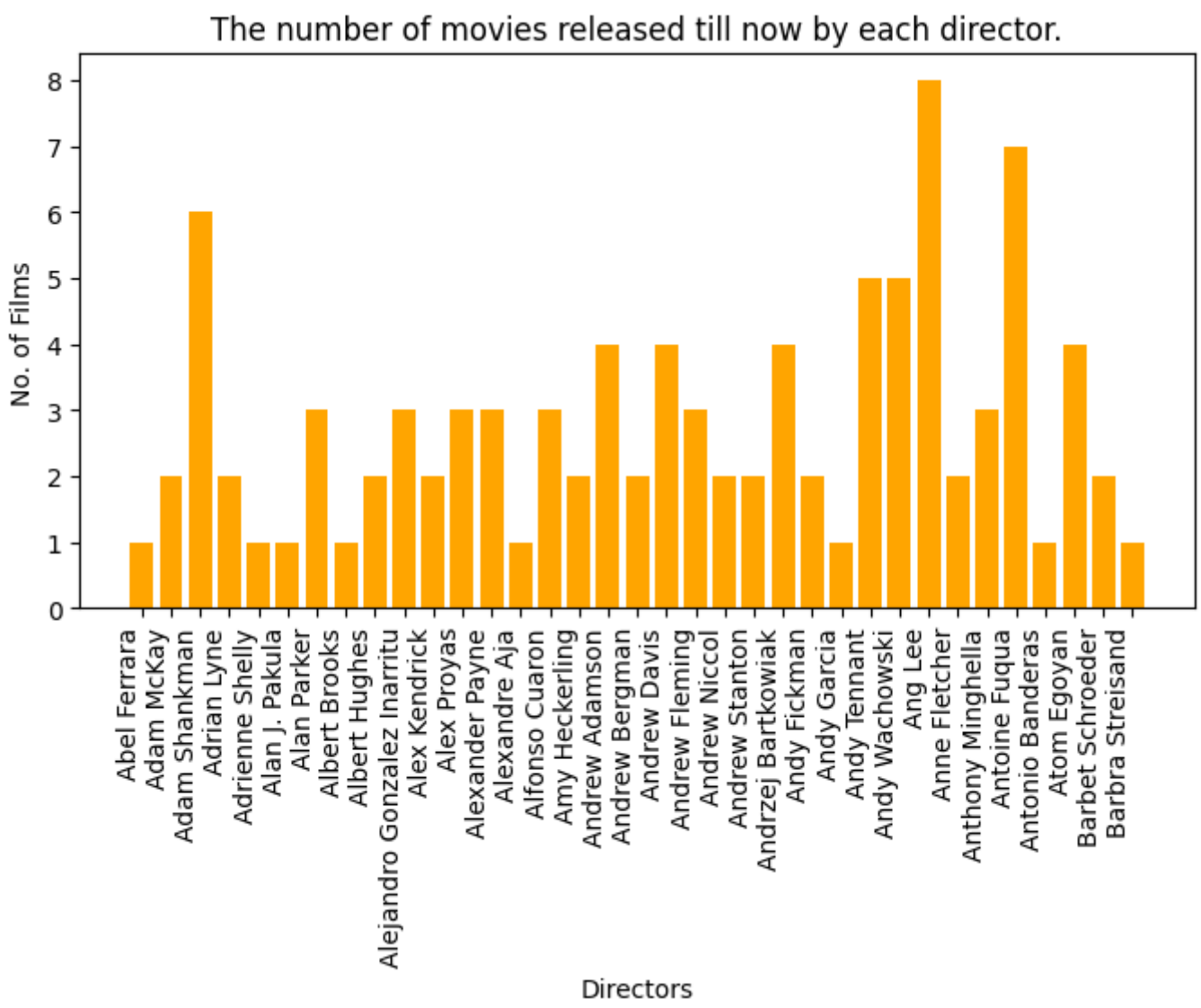
		AVG_IMDB_RATING
Director	Creative Type	
Abel Ferrara	Historical Fiction	6.0
Adam McKay	Contemporary Fiction	6.0
	Historical Fiction	7.0
Adam Shankman	Contemporary Fiction	5.2
	Historical Fiction	7.0
...	...	...
Zach Braff	Contemporary Fiction	7.0
Zack Snyder	Historical Fiction	7.0
	Science Fiction	7.0
	Super Hero	7.0
Zak Penn	Contemporary Fiction	6.0

828 rows × 1 columns

###d. Draw the plot to compare the number of movies released till now by each director.

]:
```
no_of_films = movies.groupby(['Director'])['Title'].nunique().reset_index()
no_of_films = no_of_films.head(35)
plt.figure(figsize=(8,4))
plt.bar(no_of_films['Director'],no_of_films['Title'],color='orange')
plt.xlabel('Directors')
plt.ylabel('No. of Films')
plt.title('The number of movies released till now by each director.')
plt.xticks(rotation=90, ha='right')
plt.show()
```

]:



###e. Find the genres of the movies released in each year in the ascending order.

]:
```
sorted_movies = movies.sort_values(by=['Release Year','Major Genre'])
movie_genres = sorted_movies.groupby(['Release Year','Major Genre']).agg(list).reset_index()
#movie_genres[['Release Year','Major Genre']]
pivot_table = movie_genres.pivot_table(index=['Release Year','Major Genre'])
pivot_table
```

]:
```
<ipython-input-269-db6305ed14a4>:4: FutureWarning: pivot_table dropped a column because it failed to aggregate. This behavior is deprecated and
will raise in a future version of pandas. Select only the columns that can be aggregated.
  pivot_table = movie_genres.pivot_table(index=['Release Year','Major Genre'])
```

]:

Release Year	Major Genre
1939	Drama
	Musical
1960	Western
1965	Drama
1969	Western
...	...
2010	Comedy
	Drama
	Horror
	Romantic Comedy
	Thriller/Suspense

204 rows × 0 columns

###f. Find the budgets of the movies released by each distributor along with movie names.

]:
```
movie_budget = movies.groupby(['Distributor','Title','Production Budget']).agg(list).reset_index()
movie_budget.pivot_table(index=['Distributor','Title', 'Production Budget'])
```

]:
```
<ipython-input-270-1da8bcc3ee6b>:2: FutureWarning: pivot_table dropped a column because it failed to aggregate. This behavior is deprecated and
will raise in a future version of pandas. Select only the columns that can be aggregated.
```

```
movie_budget.pivot_table(index=['Distributor','Title', 'Production Budget'])
```

Distributor	Title	Production Budget
20th Century Fox	12 Rounds	20000000
	A Good Year	35000000
	AVP: Alien Vs. Predator	70000000
	Alien	9000000
	Alien: Resurrection	60000000
...	...	...
Weinstein Co.	Zack and Miri Make a Porno	24000000
Weinstein/Dimension	DOA: Dead or Alive	30000000
	Grindhouse	53000000
	Scary Movie 4	40000000
WellSpring	The Brown Bunny	10000000

1455 rows × 0 columns

###g. Find the movies with the same IMDB rating but with different no.of IMDB votes.

```
ratings = movies.groupby(['IMDB Rating','IMDB Votes','Title']).agg(list).reset_index()
ratings.pivot_table(index=['IMDB Rating','IMDB Votes','Title'])
```

```
<ipython-input-271-e717066823c6>:2: FutureWarning: pivot_table dropped a column because it failed to aggregate. This behavior is deprecated and
will raise in a future version of pandas. Select only the columns that can be aggregated.
  ratings.pivot_table(index=['IMDB Rating','IMDB Votes','Title'])
```

IMDB Rating	IMDB Votes	Title
1	34928	Disaster Movie
2	212	Closer
	217	Scream
	3449	Marci X
	3458	Furry Vengeance
...	...	...
8	387438	The Lord of the Rings: The Fellowship of the Ring
	417703	Pulp Fiction
	465000	The Dark Knight
9	188247	Inception
	519541	The Shawshank Redemption

1452 rows × 0 columns

###h. Write a Pandas program to get those movies whose revenue more than 2 million and spent less than 1 million.

```
movies['revenue'] = movies['US Gross'] + movies['Worldwide Gross']
revenue = movies[(movies['Production Budget']<1000000) & (movies['revenue']>2000000)]
revenue[['Title','revenue','Production Budget']].reset_index()
```

```
<ipython-input-272-fb99deab1f71>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['revenue'] = movies['US Gross'] + movies['Worldwide Gross']
```

	index	Title	revenue	Production Budget
0	28	Twin Falls Idaho	2012569	500000
1	132	The Brothers McMullen	20853012	25000
2	184	Clerks	6146856	27000
3	463	In the Company of Men	5767322	25000
4	708	Pi	7899665	68000
5	913	Saints and Soldiers	2620940	780000
6	923	Swingers	11048559	200000
7	1039	Welcome to the Dollhouse	8924869	800000
8	1298	Better Luck Tomorrow	7611616	250000
9	1438	Chasing Amy	27161609	250000
10	1455	Chuck&Buck	2213343	250000
11	1727	Facing the Giants	20356662	100000
12	1793	Fireproof	66902958	500000
13	2432	Napoleon Dynamite	90681912	400000
14	2526	Paranormal Activity	301689263	15000
15	2758	Super Size Me	41058736	65000
16	2914	Tadpole	6091529	150000
17	3067	Raising Victor Vargas	4885423	800000

###i. Find the no. of movies in each genre under each source.

]:
```
genre_source = movies.groupby(['Source','Major Genre'])['Title'].nunique().reset_index()
genre_source = genre_source.rename(columns={'Title':'No. of Movies'})
genre_source.pivot_table(index=['Source','Major Genre','No. of Movies'])
```

]:

Source	Major Genre	No. of Movies
Based on Book/Short Story	Action	37
	Adventure	51
	Black Comedy	8
	Comedy	40
	Drama	158
...	...	...
Spin-Off	Horror	2
Traditional/Legend/Fairytale	Action	1
	Adventure	4
	Comedy	1
	Drama	2

74 rows × 0 columns

###j. Find the no. of movies released in each decade.

]:
```
m2 = movies
m2['Decade'] = (m2['Release Year']//10) % 10 * 10
decade = m2.groupby(['Decade'])['Title'].nunique().reset_index()
decade
```

]:
```
<ipython-input-282-16ca15abdcd0>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  m2['Decade'] = (m2['Release Year']//10) % 10 * 10
```

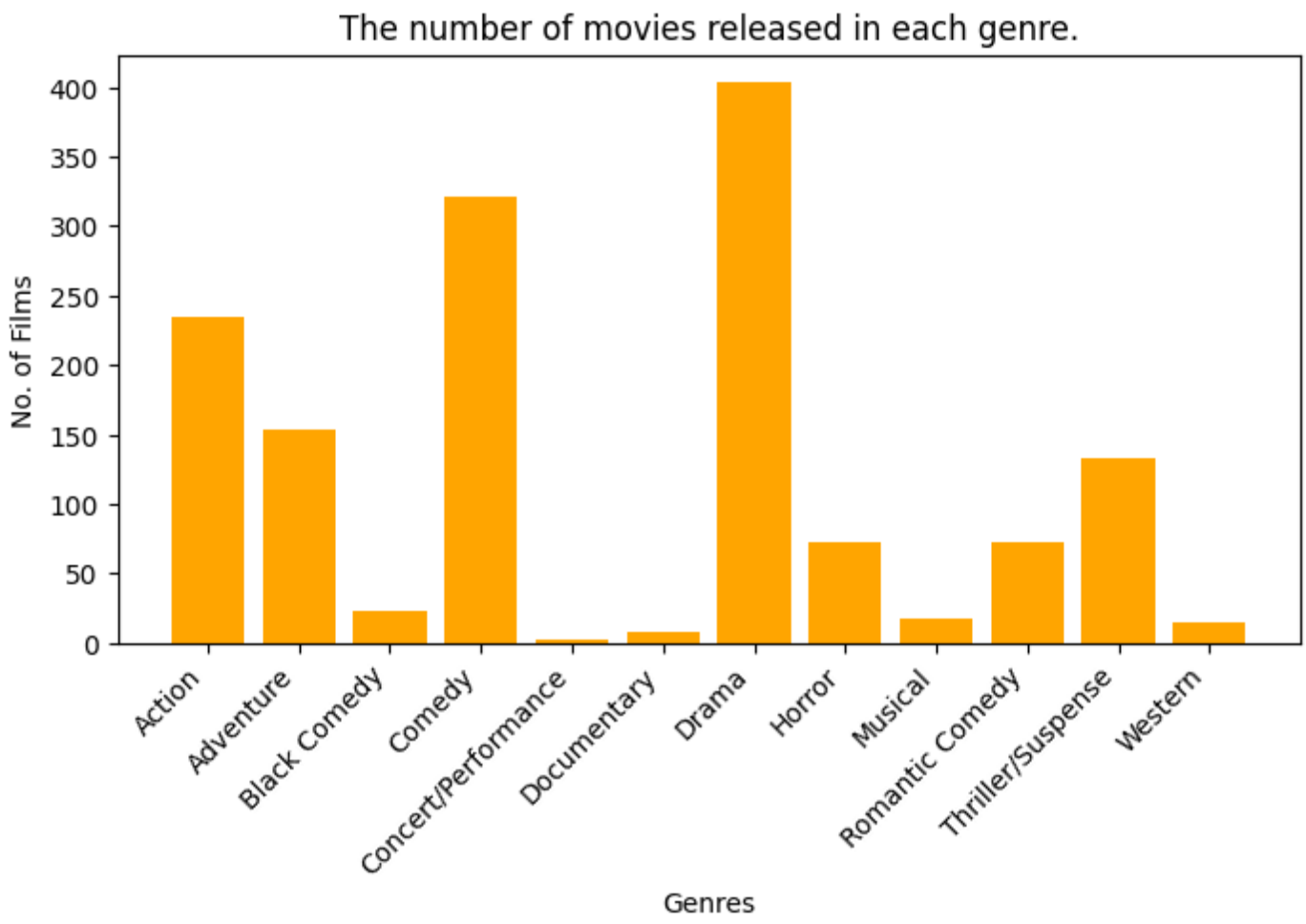
]:

	Decade	Title
0	0	953
1	10	29
2	30	2
3	60	3
4	70	8
5	80	22
6	90	438

###k. Draw the plot showing the no. of movies released in each genre.

]:
```
mov_genre = movies.groupby(['Major Genre'])['Title'].nunique().reset_index()
plt.figure(figsize=(8,4))
plt.bar(mov_genre['Major Genre'],mov_genre['Title'],color='orange')
plt.xlabel('Genres')
plt.ylabel('No. of Films')
plt.title('The number of movies released in each genre.')
plt.xticks(rotation=45, ha='right')
plt.show()
```

]:



###I. Show the no.of movies not rated under each genre in each fiction.

]:

```
not_rated = movies[(movies['MPAA Rating']=='Not Rated') & movies['Creative Type'].str.contains('Fiction')]
not_rated = not_rated.groupby(['Major Genre','Creative Type'])['Title'].nunique().reset_index()
pd.pivot_table(not_rated,index=['Major Genre','Creative Type'],values='Title')
```

]:

		Title
Major Genre	Creative Type	
Comedy	Contemporary Fiction	2
Drama	Contemporary Fiction	1
	Historical Fiction	2