In [3]:
```
import pandas as pd
import numpy as np
import numpy as pandas
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
#Code and Kaggle References: https://www.kaggle.com/code/codersaurabh/movies-recommendation-system/notebook
#Import libraries required for recommender project
```

In [4]:
```
df_movies =pd.read_csv('movies.csv')
print('Movies_shape=',df_movies.shape)
df_links =pd.read_csv('links.csv')
print('Links_shape=',df_links.shape)
df_rating =pd.read_csv('ratings.csv')
print('Rating_shape=',df_rating.shape)
df_tags=pd.read_csv('tags.csv')
print('Tags_shape=',df_tags.shape)

Movies_shape= (9742, 3)
Links_shape= (9742, 3)
Rating_shape= (100836, 4)
Tags_shape= (3683, 4)
```

In [5]:
```
df_movies.head(5)#View df_movies for five rows.
```

Out[5]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [6]:
```
df_links.head(5)#View df_links header information for 5 rows.
```

Out[6]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

In [7]:
```
df_rating.head(5)#View df_rating header information for 5 rows.
```

Out[7]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [8]:
```
df_tags.head(5)#View df_tags header information for 5 rows.
```

Out[8]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

In [9]:
```
final_data =df_rating.pivot(index='movieId',columns='userId',values='rating')#Pivot on index columns and ratings.
final_data.head()#View df for them.
```

Out[9]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	NaN	NaN	NaN	4.0	NaN	4.5	NaN	NaN	NaN	...	4.0	NaN	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	...	NaN	4.0	NaN	NaN	5.0	3.5	NaN	NaN	2.0	NaN
3	4.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN

5 rows × 610 columns

In [10]:
```
final_data =final_data.fillna(0)#
final_data.shape#View final shape.
```

Out[10]:
```
(9724, 610)
```

In [11]:
```
final_data.head()#View final data headers.
```

Out[11]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	...	0.0	4.0	0.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0

5 rows × 610 columns

In [12]:
```
df_rate_merge =pd.merge(df_rating,df_movies,on='movieId')#Merge the various dataframes.
df_rate_merge.head()#View the merged dataframes.
```

Out[12]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

In [13]:
```
df_rate_merge.groupby('title')['rating'].mean().sort_values(ascending=False).head()#Group by title and rating and view first five rows.
```

Out[13]:
```
title
Gena the Crocodile (1969)                 5.0
True Stories (1986)                       5.0
Cosmic Scrat-tastrophe (2015)             5.0
Love and Pigeons (1985)                   5.0
Red Sorghum (Hong gao liang) (1987)       5.0
Name: rating, dtype: float64
```

In [15]:
```
rating =pd.DataFrame(df_rate_merge.groupby('title')['rating'].mean())
rating.head()
```

Out[15]:

	rating
title	
'71 (2014)	4.0
'Hellboy': The Seeds of Creation (2004)	4.0
'Round Midnight (1986)	3.5
'Salem's Lot (2004)	5.0
'Til There Was You (1997)	4.0

In [16]:
```
rating['count_rating']=pd.DataFrame(df_rate_merge.groupby('title')['rating'].count())#Groupby title and rating and create new dataframe.
```

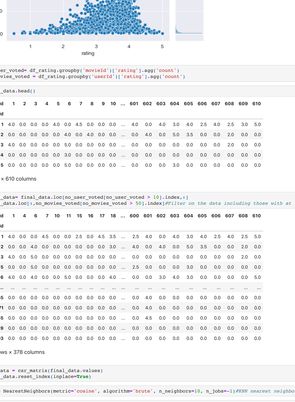
In [17]:
```
rating.head()#View the header information for first five rows.
```

Out[17]:

	rating	count_rating
title		
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

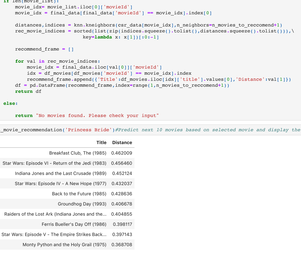
In [18]:
```
sns.set_style('darkgrid')
plt.figure(figsize=(10,8))
rating['count_rating'].hist(bins=70)#View a histograph with counts and ratings.
```

Out[18]:
```
<AxesSubplot:>
```



In [19]:
```
plt.figure(figsize=(10,8))
rating['rating'].hist(bins=100)#View a histogram with ratings.
```

Out[19]:
```
<AxesSubplot:>
```



In [20]:
```
sns.jointplot(x='rating',y='count_rating',data=rating)#Joint plot and scatterplot visualization with counts and ratings.
```

Out[20]:
```
<seaborn.axisgrid.JointGrid at 0x7f90da8dfad0>
```


In [21]:
```
no_user_voted= df_rating.groupby('movieId')['rating'].agg('count')
no_movies_voted = df_rating.groupby('userId')['rating'].agg('count')
```

In [22]:
```
final_data.head()
```

Out[22]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	...	0.0	4.0	0.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0

5 rows × 610 columns

In [23]:
```
final_data= final_data.loc[no_user_voted[no_user_voted > 10].index,:]
final_data=final_data.loc[:,no_movies_voted[no_movies_voted > 50].index]#Filter on the data including those with at least with 50 movies and 50 votes.
```

Out[23]:

userId	1	4	6	7	10	11	15	16	17	18	...	600	601	602	603	604	605	606	607	608	610
movieId																					
1	4.0	0.0	4.0	4.5	0.0	0.0	2.5	0.0	4.5	3.5	...	2.5	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	5.0
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	4.0	0.0	4.0	0.0	0.0	5.0	3.5	0.0	0.0	0.0
3	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0
6	4.0	0.0	4.0	0.0	5.0	0.0	0.0	0.0	4.0	...	0.0	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
176371	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
187593	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2121 rows × 378 columns

In [24]:
```
csr_data = csr_matrix(final_data.values)
final_data.reset_index(inplace=True)
```

In [25]:
```
knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=10, n_jobs=-1)#KNN nearest neighbors set up.
```

In [26]:
```
knn.fit(csr_data)#Fit the data and the KNN model.
```

Out[26]:
```
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=10)
```

In [27]:
```
def get_movie_recommendation(movie_name):#Python code to recommend the user movies based on movie selected.
    n_movies_to_reccomend = 10
    movie_list = df_movies[df_movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_data[final_data['movieId'] == movie_idx].index[0]

        distances,indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
        rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),\
                                   key=lambda x: x[1])[:0:-1]

        recommend_frame = []

        for val in rec_movie_indices:
            movie_idx = final_data.iloc[val[0]]['movieId']
            idx = df_movies[df_movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':df_movies.iloc[idx]['title'].values[0],'Distance':val[1]})
        df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return df

    else:

        return "No movies found. Please check your input"
```

In [31]:
```
get_movie_recommendation('Princess Bride')#Predict next 10 movies based on selected movie and display the distance between the movie selected and the movie predicted.
```

Out[31]:

	Title	Distance
1	Breakfast Club, The (1985)	0.462009
2	Star Wars: Episode VI - Return of the Jedi (1983)	0.456460
3	Indiana Jones and the Last Crusade (1989)	0.452124
4	Star Wars: Episode IV - A New Hope (1977)	0.432037
5	Back to the Future (1985)	0.428536
6	Groundhog Day (1993)	0.409678
7	Raiders of the Lost Ark (Indiana Jones and the...	0.404855
8	Ferris Bueller's Day Off (1986)	0.398117
9	Star Wars: Episode V - The Empire Strikes Back...	0.397143
10	Monty Python and the Holy Grail (1975)	0.368708