In [1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('Amazon - Movies and TV Ratings.csv')
```

In [3]:
```
data1 = pd.read_csv('Amazon - Movies and TV Ratings.csv')
```

In [4]:
```
data.head()
```

Out[4]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	M
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	

5 rows × 207 columns

In [5]:
```
data.shape
```

Out[5]:
```
(4848, 207)
```

In [6]:
```
data.size
```

Out[6]:
```
1003536
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['user_id', 'Movie1', 'Movie2', 'Movie3', 'Movie4', 'Movie5', 'Movie
6',
       'Movie7', 'Movie8', 'Movie9',
       ...
       'Movie197', 'Movie198', 'Movie199', 'Movie200', 'Movie201', 'Movie20
2',
       'Movie203', 'Movie204', 'Movie205', 'Movie206'],
      dtype='object', length=207)
```

In [8]:
```
data.dtypes
```

Out[8]:
```
user_id      object
Movie1      float64
Movie2      float64
Movie3      float64
Movie4      float64
             ...
Movie202    float64
Movie203    float64
Movie204    float64
Movie205    float64
Movie206    float64
Length: 207, dtype: object
```

In [9]:
```
data.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Movie1	1.0	5.000000	NaN	5.0	5.00	5.0	5.0	5.0
Movie2	1.0	5.000000	NaN	5.0	5.00	5.0	5.0	5.0
Movie3	1.0	2.000000	NaN	2.0	2.00	2.0	2.0	2.0
Movie4	2.0	5.000000	0.000000	5.0	5.00	5.0	5.0	5.0
Movie5	29.0	4.103448	1.496301	1.0	4.00	5.0	5.0	5.0
...	...	...	...	...	...	...	...	...
Movie202	6.0	4.333333	1.632993	1.0	5.00	5.0	5.0	5.0
Movie203	1.0	3.000000	NaN	3.0	3.00	3.0	3.0	3.0
Movie204	8.0	4.375000	1.407886	1.0	4.75	5.0	5.0	5.0
Movie205	35.0	4.628571	0.910259	1.0	5.00	5.0	5.0	5.0
Movie206	13.0	4.923077	0.277350	4.0	5.00	5.0	5.0	5.0

206 rows × 8 columns

In [10]:
```
data.corr()
```

Out[10]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie1
Movie1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie5	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	.
Movie202	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie203	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie204	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie205	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Movie206	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

206 rows × 206 columns

In [11]:
```
data.count()
```

Out[11]:
```
user_id     4848
Movie1         1
Movie2         1
Movie3         1
Movie4         2
            ... 
Movie202       6
Movie203       1
Movie204       8
Movie205      35
Movie206      13
Length: 207, dtype: int64
```

In [12]:
```
data.isna().sum()
```

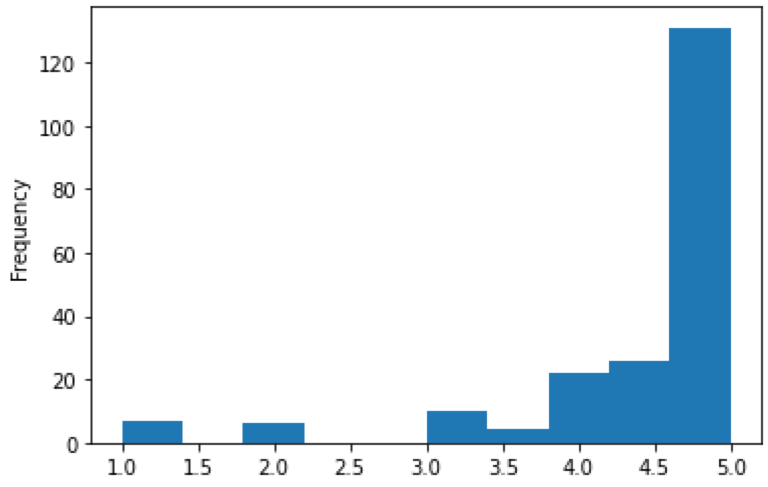
Out[12]:
```
user_id        0
Movie1      4847
Movie2      4847
Movie3      4847
Movie4      4846
            ... 
Movie202    4842
Movie203    4847
Movie204    4840
Movie205    4813
Movie206    4835
Length: 207, dtype: int64
```

In [13]:
```
data.drop_duplicates()
```

Out[13]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Mov
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	N
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	N
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	N
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	N
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	N
...	...	...	...	...	...	...	...	...	
4843	A1IMQ9WMFYKWH5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4844	A1KLIKPUF5E88I	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4845	A5HG6WFZLO10D	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4846	A3UU690TWXCG1X	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

In [14]:
```
data.describe().T['mean'].plot(kind='hist');
```



Which movies have maximum views/ratings?

In [15]:
```
data.drop('user_id', axis = 1).count().sort_values(ascending = False).head(1)
```

Out[15]:
```
Movie127    2313
dtype: int64
```

THE MOVIE WITH MAXIMUM VIEWS IS Movie127.

In [16]:
```
data.drop('user_id', axis = 1).sum().sort_values(ascending = False).head(1)
```

Out[16]:
```
Movie127    9511.0
dtype: float64
```

THE MOVIE WITH MAXIMUM RATINGS IS Movie127.

What is the average rating for each movie? Define the top 5 movies with the maximum ratings.

In [17]:
```
data.drop('user_id', axis = 1).mean()
```

Out[17]:
```
Movie1      5.000000
Movie2      5.000000
Movie3      2.000000
Movie4      5.000000
Movie5      4.103448
              ...
Movie202    4.333333
Movie203    3.000000
Movie204    4.375000
Movie205    4.628571
Movie206    4.923077
Length: 206, dtype: float64
```

In [18]:
```
data.drop('user_id', axis = 1).mean().sort_values(ascending = False).head(5)
```

Out[18]:
```
Movie1      5.0
Movie55     5.0
Movie131    5.0
Movie132    5.0
Movie133    5.0
dtype: float64
```

Define the top 5 movies with the least audience.

In [19]:
```
data.drop('user_id', axis = 1).count().sort_values(ascending = True).head(5)
```

Out[19]:
```
Movie1      1
Movie71     1
Movie145    1
Movie69     1
Movie68     1
dtype: int64
```

In [20]:
```
data
```

Out[20]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
4843	A1IMQ9WMFYKWH5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4844	A1KLIKPUF5E88I	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4845	A5HG6WFZLO10D	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4846	A3UU690TWXCG1X	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4847	AI4J762YI6S06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4848 rows × 207 columns

In [21]:
```
movie_data = data1.melt(id_vars = data1.columns[0],value_vars=data1.columns[1
movie_data
```

Out[21]:

	user_id	Movies	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	NaN
2	A3LKP6WPMP9UKX	Movie1	NaN
3	AVIY68KEPQ5ZD	Movie1	NaN
4	A1CV1WROP5KTTW	Movie1	NaN
...	...	...	...
998683	A1IMQ9WMFYKWH5	Movie206	5.0
998684	A1KLIKPUF5E88I	Movie206	5.0
998685	A5HG6WFZLO10D	Movie206	5.0
998686	A3UU690TWXCG1X	Movie206	5.0
998687	AI4J762YI6S06	Movie206	5.0

In [22]:
```
movie_data.fillna(0, inplace = True)
```

In [23]:
```
movie_data
```

Out[23]:

	user_id	Movies	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	0.0
2	A3LKP6WPMP9UKX	Movie1	0.0
3	AVIY68KEPQ5ZD	Movie1	0.0
4	A1CV1WROP5KTTW	Movie1	0.0
...	...	...	...
998683	A1IMQ9WMFYKWH5	Movie206	5.0
998684	A1KLIKPUF5E88I	Movie206	5.0
998685	A5HG6WFZLO10D	Movie206	5.0
998686	A3UU690TWXCG1X	Movie206	5.0
998687	AI4J762YI6S06	Movie206	5.0

Divide the data into training and test data

In [24]:
```
features = movie_data[['user_id', 'Movies']]
```

In [25]:
```
target = movie_data[['Rating']]
```

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(features, target, train_s
```

In [28]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[28]:
```
((749016, 2), (249672, 2), (749016, 1), (249672, 1))
```

Recommendation Model: Some of the movies hadn't been watched and therefore, are not rated by the users. Netflix would like to take this as an opportunity and build a machine learning recommendation algorithm which provides the ratings for each of the users.

In [29]:
```
data = data.drop('user_id', axis = 1)
```

In [30]:
```
data['user_id'] = np.arange(len(data))
```

In [31]:
```
data = data.set_index(data['user_id'])
```

In [32]:
```
data = data.drop(['user_id'], axis = 1)
```

In [33]:
```
data
```

Out[33]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10
user_id										
0	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
4843	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4844	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4845	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4846	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4847	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4848 rows × 206 columns

In [34]:
```
data.fillna(0, inplace = True)
```

In [35]:
```
data[data.index == 101]
```

Out[35]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10
user_id										
101	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 206 columns

In [36]:
```
data
```

Out[36]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movi
user_id										
0	5.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
4843	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4844	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4845	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Build a recommendation model

In [37]:
```
# User - User Based Collaborative Filtering
```

In [38]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [39]:
```
import operator
```

In [40]:
```
def similar_users(user_id, matrix, k=5):

    #Creating a df of just current user
    user = matrix[matrix.index == user_id]

    # Create a df for other users
    other_users = matrix[matrix.index != user_id]

    #Cal cosine similarity btw user and others
    similarities = cosine_similarity(user, other_users)[0].tolist()

    #Create list of indices of these users
    indices = other_users.index.tolist()

    #Create key/value pairs of users index and their similarity
    index_similarity = dict(zip(indices, similarities))

    #Sort by similarities
    index_similarity_sorted = sorted(index_similarity.items(), key = operator
    index_similarity_sorted.reverse()

    #Grab k users off the top
    top_users_similarities = index_similarity_sorted[:k]
    users = [u[0] for u in top_users_similarities]

    return users
```

In [41]:
```
similar_user_indices = similar_users(101, data, 10)
```

In [42]:
```
print(similar_user_indices)
```

```
[367, 366, 365, 364, 363, 362, 361, 360, 359, 358]
```

Make predictions

In [43]:
```
def recommend_movies(user_index, similar_user_indices, matrix, items = 7):

    #Load vectors for similar users
    similar_users = matrix[matrix.index.isin(similar_user_indices)]

    #Cal avg ratings across the 3 similar users
    similar_users = similar_users.mean(axis = 0)

    #Convert to dataframe so its easy to sort and filter
    similar_users_df = pd.DataFrame(similar_users, columns=['mean'])

    #Load vector for the current user
    user_df = matrix[matrix.index == user_index]

    #Transpose it so its easier to filter
    user_df_transposed = user_df.transpose()

    #Rename the column as 'rating'
    user_df_transposed.rename(columns = {user_index: 'rating'}, inplace = Tru

    #Remove any rows without a 0 value. Movies not yet watched
    movies_unseen = user_df_transposed.index.tolist()

    #Filter avg ratings of similar users for only Movies the current user has
    similar_users_df_filtered = similar_users_df[similar_users_df.index.isin(

    #Order the df
    similar_users_df_ordered = similar_users_df.sort_values(by=['mean'], asce

    #Grab the top n movies
    top_n_movies = similar_users_df_ordered.head(items)
    top_n_movies_indices = top_n_movies.index.tolist()

    #Look these books in the other df to find names
    movie_information = movie_data[movie_data['Movies'].isin(top_n_movies_inc

    return movie_information #items

recommend_movies(101, similar_user_indices, data)
```

Out[43]:

	user_id	Movies	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	0.0
2	A3LKP6WPMP9UKX	Movie1	0.0
3	AVIY68KEPQ5ZD	Movie1	0.0
4	A1CV1WROP5KTTW	Movie1	0.0
...	...	...	...
659323	A1IMQ9WMFYKWH5	Movie136	0.0
659324	A1KLIKPUF5E88I	Movie136	0.0
659325	A5HG6WFZLO10D	Movie136	0.0

	user_id	Movies	Rating
659326	A3UU690TWXCG1X	Movie136	0.0
659327	AI4J762YI6S06	Movie136	0.0

33936 rows × 3 columns

In [44]:
```
movies_recommended = recommend_movies(101, similar_user_indices, data)
```

In [45]:
```
movies_recommended
```

Out[45]:

	user_id	Movies	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	0.0
2	A3LKP6WPMP9UKX	Movie1	0.0
3	AVIY68KEPQ5ZD	Movie1	0.0
4	A1CV1WROP5KTTW	Movie1	0.0
...	...	...	...
659323	A1IMQ9WMFYKWH5	Movie136	0.0
659324	A1KLIKPUF5E88I	Movie136	0.0
659325	A5HG6WFZLO10D	Movie136	0.0
659326	A3UU690TWXCG1X	Movie136	0.0
659327	AI4J762YI6S06	Movie136	0.0

33936 rows × 3 columns

In [50]:
```
final = pd.DataFrame(movies_recommended.groupby('Movies')['Rating'].max())
```

In [47]:
```
final
```

Out[47]:

	Rating
Movies	
Movie1	5.0
Movie131	5.0
Movie133	5.0
Movie134	5.0
Movie135	5.0
Movie136	5.0
Movie16	5.0