Data Description: Amazon Reviews data. The repository has several datasets. For this case study, we are using the Electronics dataset.

Domain: E-commerce

Context: Online E-commerce websites like Amazon, Flipkart uses different recommendation models to provide different suggestions to different users. Amazon currently uses item-to-item collaborative filtering, which scales to massive data sets and produces high-quality recommendations in real-time.

Attribute Information: ● userId : Every user identified with a unique id ● productId : Every product identified with a unique id ● Rating : Rating of the corresponding product by the corresponding user ● timestamp : Time of the rating

Objective: Build a recommendation system to recommend products to customers based on the their previous ratings for other products.

1. Read and explore the given dataset. (Rename column/add headers, plot histograms, find data characteristics)

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

%matplotlib inline
```

Load the Dataset and Add headers

In [2]:
```
electronics_data = pd.read_csv("ratings_Electronics.csv")
electronics_data.columns = ['userId','productId','Rating','timestamp']
electronics_data.head()
```

Out[2]:

	userId	productId	Rating	timestamp
0	A2CX7LUOHB2NDG	321732944	5	1341100800
1	A2NWSAGRHCP8N5	439886341	1	1367193600
2	A2WNBOD3WNDNKT	439886341	3	1374451200
3	A1GI0U4ZRJA8WN	439886341	1	1334707200
4	A1QGNMC6O1VW39	511189877	5	1397433600

In [3]:
```
# Display the data

electronics_data.head()
```

Out[3]:

	userId	productId	Rating	timestamp
0	A2CX7LUOHB2NDG	321732944	5	1341100800
1	A2NWSAGRHCP8N5	439886341	1	1367193600
2	A2WNBOD3WNDNKT	439886341	3	1374451200
3	A1GI0U4ZRJA8WN	439886341	1	1334707200
4	A1QGNMC6O1VW39	511189877	5	1397433600

In [4]:
```
#Shape of the data
electronics_data.shape
```

Out[4]:
```
(1048575, 4)
```

In [5]:
```
#Taking subset of the dataset
electronics_data=electronics_data.iloc[:1048576,0:]
```

In [6]:
```
#Check the datatypes
electronics_data.dtypes
```

Out[6]:
```
userId       object
productId    object
Rating        int64
timestamp     int64
dtype: object
```

In [7]:
```
electronics_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 4 columns):
 #   Column     Non-Null Count    Dtype 
---  ------     --------------    ----- 
 0   userId     1048575 non-null  object
 1   productId  1048575 non-null  object
 2   Rating     1048575 non-null  int64 
 3   timestamp  1048575 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 32.0+ MB
```

In [8]:
```
#Five point summary

electronics_data.describe()['Rating'].T
```

Out[8]:
```
count    1.048575e+06
mean     3.973379e+00
std      1.399329e+00
min      1.000000e+00
25%      3.000000e+00
50%      5.000000e+00
75%      5.000000e+00
max      5.000000e+00
Name: Rating, dtype: float64
```

In [9]:
```
#Find the minimum and maximum ratings
print('Minimum rating is: %d' %(electronics_data.Rating.min()))
print('Maximum rating is: %d' %(electronics_data.Rating.max()))
```

```
Minimum rating is: 1
Maximum rating is: 5
```

The rating of the product range from 0 to 1

Handling Missing values

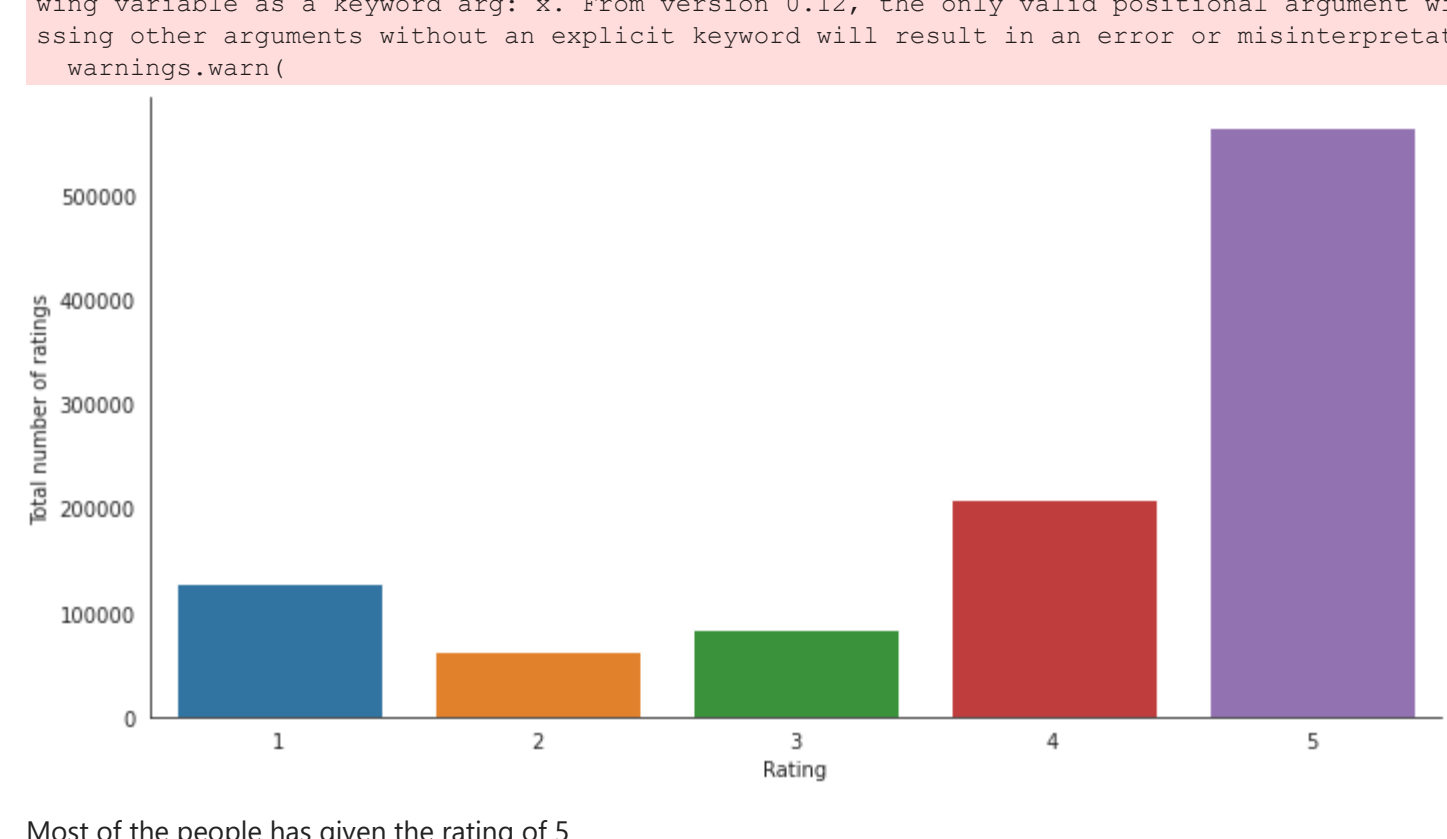
In [10]:
```
#Check for missing values
print('Number of missing values across columns: \n',electronics_data.isnull().sum())
```

```
Number of missing values across columns:
 userId       0
productId    0
Rating       0
timestamp    0
dtype: int64
```

Ratings

In [11]:
```
# Check the distribution of the rating
with sns.axes_style('white'):
    g = sns.factorplot("Rating", data=electronics_data, aspect=2.0,kind='count')
    g.set_ylabels("Total number of ratings")
```

```
C:\Users\Prashant Varshney\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\Prashant Varshney\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Most of the people has given the rating of 5

Unique Users and products

In [12]:
```
print("Total data ")
print("-"*50)
print("\nTotal no of ratings :",electronics_data.shape[0])
print("Total No of Users   :", len(np.unique(electronics_data.userId)))
print("Total No of products  :", len(np.unique(electronics_data.productId)))
```

```
Total data 
--------------------------------------------------

Total no of ratings : 1048575
Total No of Users   : 786329
Total No of products  : 61893
```

Dropping the TimeStamp Column

In [13]:
```
#Dropping the Timestamp column

electronics_data.drop(['timestamp'], axis=1,inplace=True)
```

Analyzing the rating

In [14]:
```
#Analysis of rating given by the user

no_of_rated_products_per_user = electronics_data.groupby(by='userId')['Rating'].count().sort_values(ascending=F

no_of_rated_products_per_user.head()
```

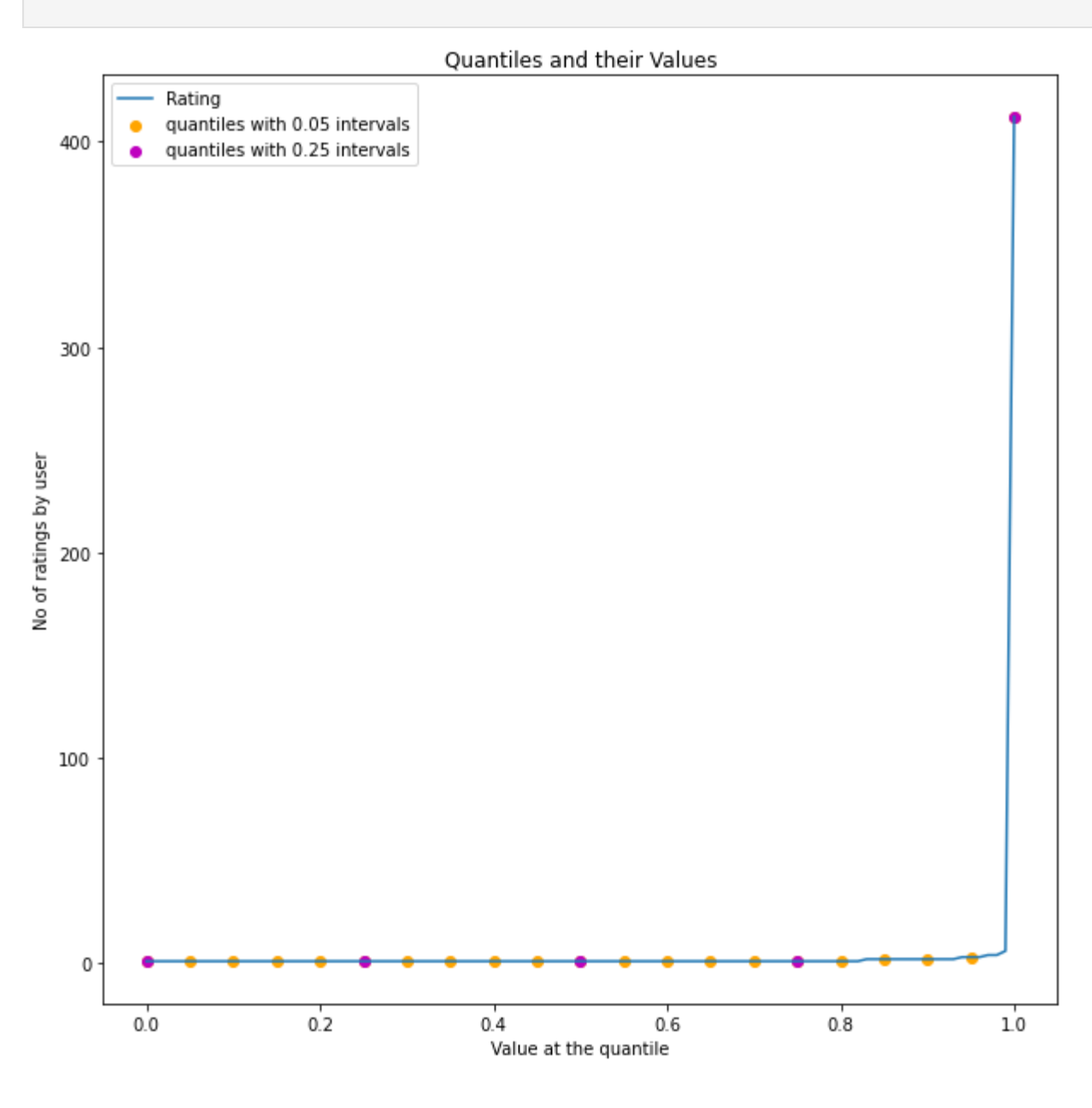
Out[14]:
```
userId
A5JLAU2ARJ0BO     412
A231WM2Z2JL0U3    249
A25HBO5V8S8SEA    164
A6FIAB28IS79      146
AT6CZDCP4TRGA     128
Name: Rating, dtype: int64
```

In [15]:
```
no_of_rated_products_per_user.describe()
```

Out[15]:
```
count    786329.000000
mean          1.333507
std           1.385613
min           1.000000
25%           1.000000
50%           1.000000
75%           1.000000
max         412.000000
Name: Rating, dtype: float64
```

In [16]:
```
quantiles = no_of_rated_products_per_user.quantile(np.arange(0,1.01,0.01), interpolation='higher')
```

In [17]:
```
plt.figure(figsize=(10,10))
plt.title("Quantiles and their Values")
quantiles.plot()
# quantiles with 0.05 difference
plt.scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange', label="quantiles with 0.05 intervals")
# quantiles with 0.25 difference
plt.scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m', label = "quantiles with 0.25 intervals")
plt.ylabel('No of ratings by user')
plt.xlabel('Value at the quantile')
plt.legend(loc='best')
plt.show()
```



2. Take a subset of the dataset to make it less sparse/ denser. (For example, keep the users only who has given 50 or more number of ratings)

In [18]:
```
print('\n No of rated product more than 50 per user : {}\n'.format(sum(no_of_rated_products_per_user >= 50)) )
```

```
 No of rated product more than 50 per user : 38
```

In [19]:
```
#Getting the new dataframe which contains users who has given 50 or more ratings

new_df=electronics_data.groupby("productId").filter(lambda x:x['Rating'].count() >=50)
```

3. Split the data randomly into train and test dataset. (For example, split it in 70/30 ratio)

In [21]:
```
from surprise import KNNWithMeans
from surprise import Dataset
from surprise import accuracy
from surprise import Reader
import os
from surprise.model_selection import train_test_split
```

In [ ]:

In [22]:
```
#Reading the dataset
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(new_df,reader)
```

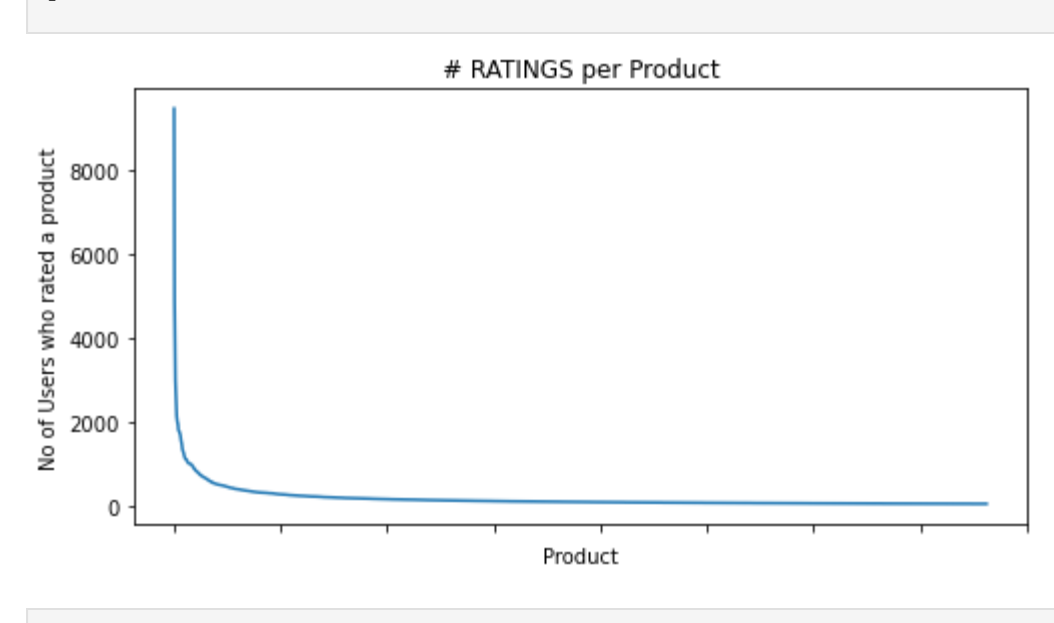
In [23]:
```
#Splitting the dataset
trainset, testset = train_test_split(data, test_size=0.3,random_state=10)
```

4. Build Popularity Recommender model.

In [24]:
```
no_of_ratings_per_movie = new_df.groupby(by='productId')['Rating'].count().sort_values(ascending=False)

fig = plt.figure(figsize=plt.figaspect(.5))
ax = plt.gca()
plt.plot(no_of_ratings_per_movie.values)
plt.title('# RATINGS per Product')
plt.xlabel('Product')
plt.ylabel('No of Users who rated a product')
ax.set_xticklabels([])

plt.show()
```



In [25]:
```
#Average rating of the product

new_df.groupby('productId')['Rating'].mean().head()
```

Out[25]:
```
productId
1400501466    3.560000
1400501520    4.243902
1400501776    3.884892
1400532620    3.684211
1400532655    3.727273
Name: Rating, dtype: float64
```

In [26]:
```
new_df.groupby('productId')['Rating'].mean().sort_values(ascending=False).head()
```

Out[26]:
```
productId
B0000DYV9H    4.947368
B000053HC5    4.945783
B00009R96C    4.885714
B00005LE76    4.879310
B000I1X3W8    4.869565
Name: Rating, dtype: float64
```

In [27]:
```
#Total no of rating for product

new_df.groupby('productId')['Rating'].count().sort_values(ascending=False).head()
```

Out[27]:
```
productId
B0002L5R78    9487
B0001FTVEK    5345
B000I68BD4    4903
B000BQ7GW8    4275
B00007E7JU    3523
Name: Rating, dtype: int64
```

In [28]:
```
ratings_mean_count = pd.DataFrame(new_df.groupby('productId')['Rating'].mean())
```

In [29]:
```
ratings_mean_count['rating_counts'] = pd.DataFrame(new_df.groupby('productId')['Rating'].count())
```

In [30]:
```
ratings_mean_count.head()
```

Out[30]:

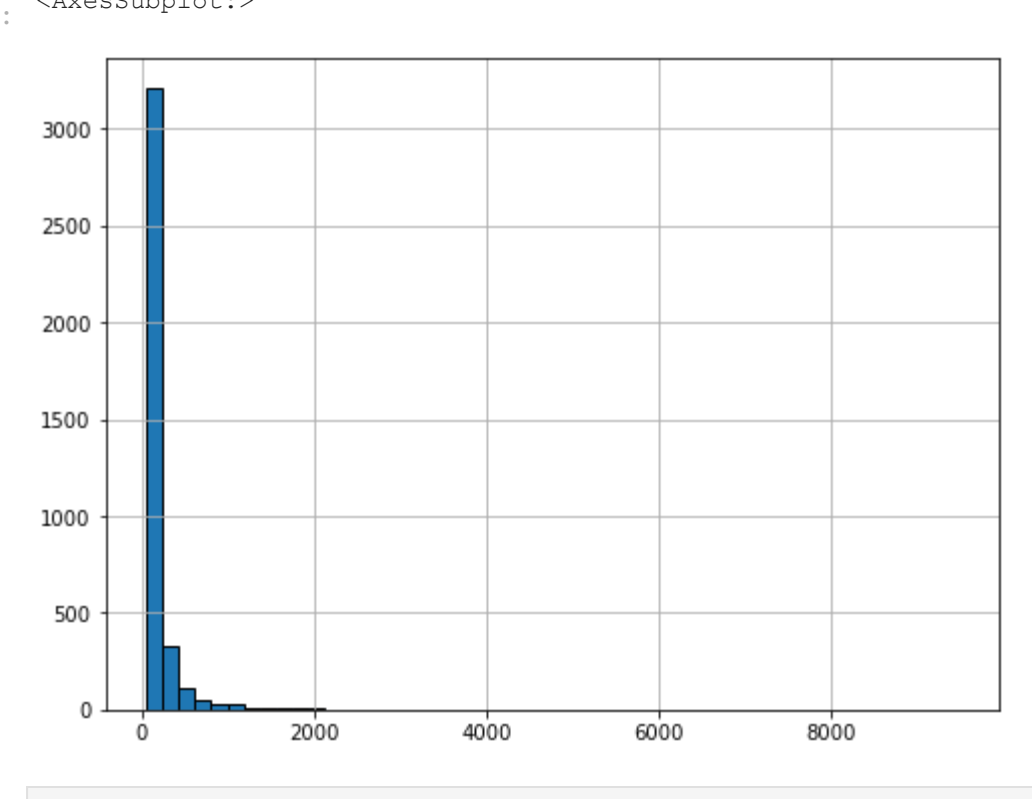
productId	Rating	rating_counts
1400501466	3.560000	250
1400501520	4.243902	82
1400501776	3.884892	139
1400532620	3.684211	171
1400532655	3.727273	484

In [31]:
```
ratings_mean_count['rating_counts'].max()
```

Out[31]:
```
9487
```

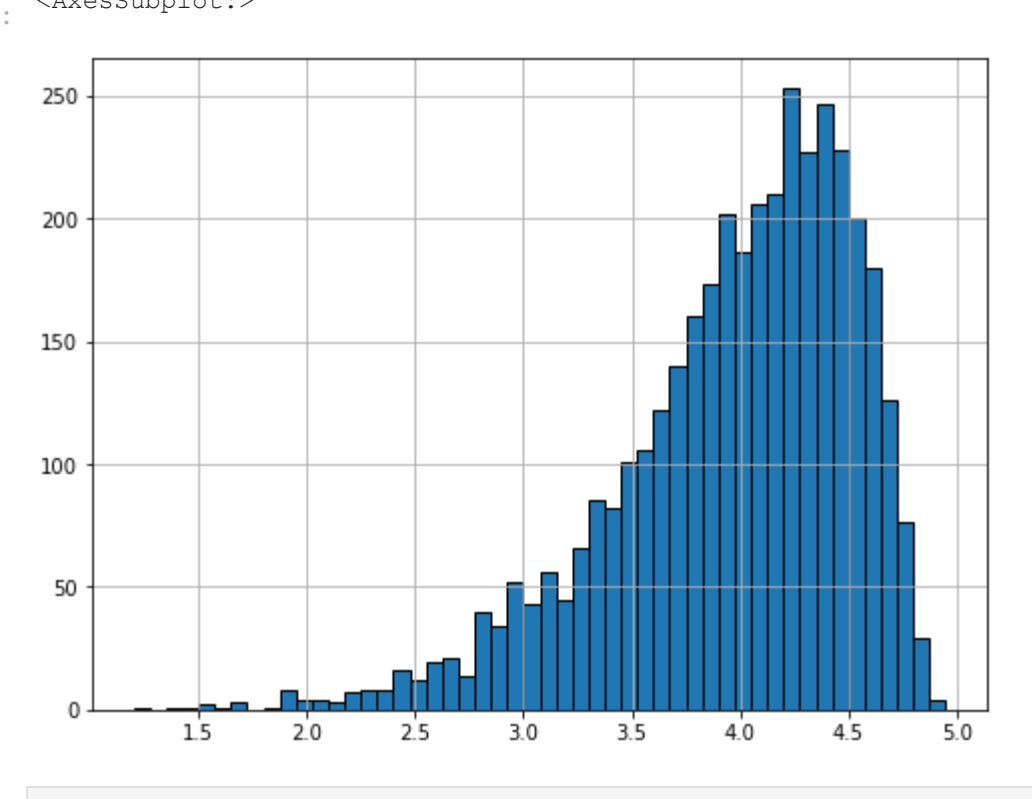
In [32]:
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['rating_counts'].hist(bins=50)
```

Out[32]:
```
<AxesSubplot:>
```



In [33]:
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['Rating'].hist(bins=50)
```

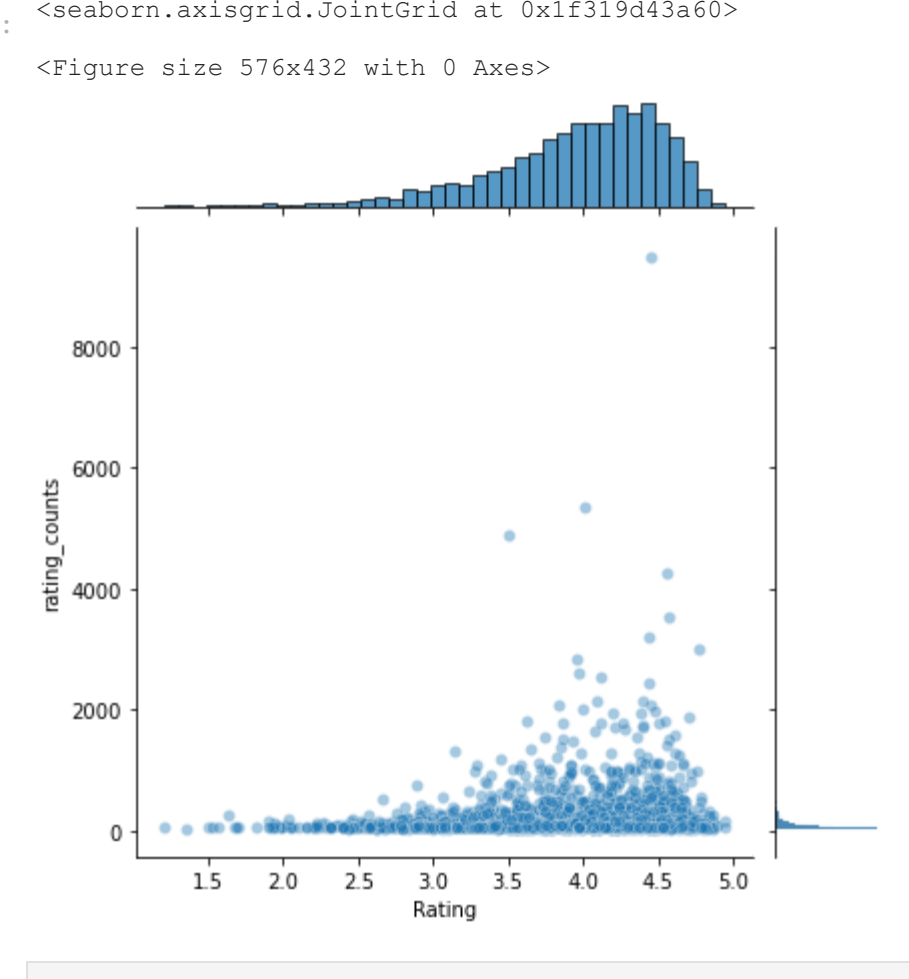
Out[33]:
```
<AxesSubplot:>
```



In [34]:
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
sns.jointplot(x='Rating', y='rating_counts', data=ratings_mean_count, alpha=0.4)
```

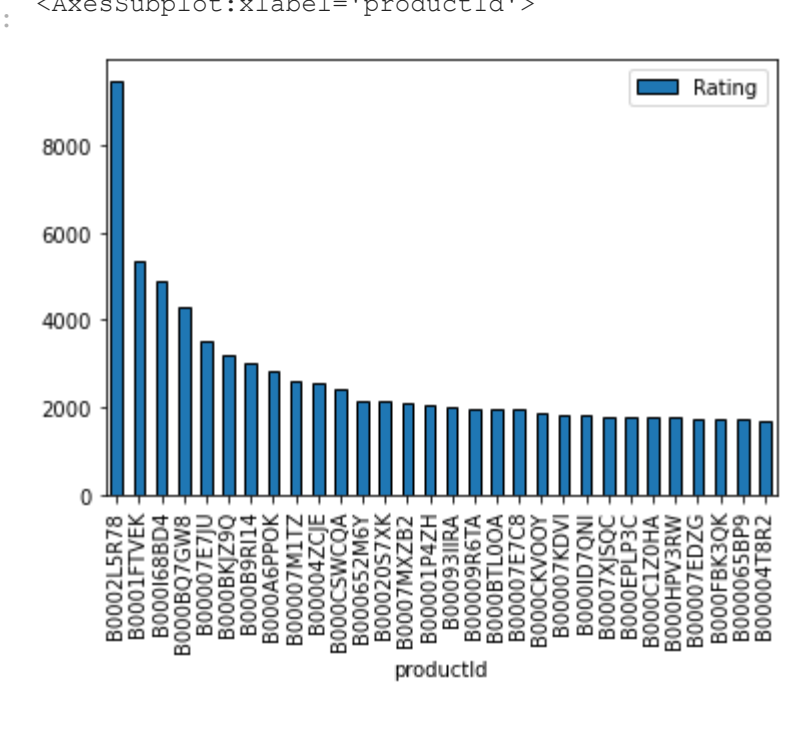
Out[34]:
```
<seaborn.axisgrid.JointGrid at 0x1f319d43a60>
```

```
<Figure size 576x432 with 0 Axes>
```



In [35]:
```
popular_products = pd.DataFrame(new_df.groupby('productId')['Rating'].count())
most_popular = popular_products.sort_values('Rating', ascending=False)
most_popular.head(30).plot(kind = "bar")
```

Out[35]:
```
<AxesSubplot:xlabel='productId'>
```



5. Build Collaborative Filtering model.

In [36]:
```
# Use user_based true/false to switch between user-based or item-based collaborative filtering
algo = KNNWithMeans(k=5, sim_options={'name': 'pearson_baseline', 'user_based': False})
algo.fit(trainset)
```

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.
```

Out[36]:
```
<surprise.prediction_algorithms.knns.KNNWithMeans at 0x1f327047100>
```

In [37]:
```
# run the trained model against the testset
test_pred = algo.test(testset)
```

In [38]:
```
test_pred
```

```
 Prediction(uid='A1HZ31834BDTK7', iid='B000085BD8', r_ui=2.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1NIFUKZLPH0O1', iid='B0002KVQBA', r_ui=5.0, est=4.575304022450888, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A32WJ62MN3J0HC', iid='B00008KWWF', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1XMAR5V7YWMY4', iid='B00009EHJV', r_ui=2.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A11NNJSM6OVXV5', iid='B000E0DU7G', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2TUB7T3RS28JM', iid='B000BRO3PA', r_ui=2.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3IBUF3JFW2KG5', iid='B0002VO7HO', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3GW639VY3QR2M', iid='B00000JDHV', r_ui=5.0, est=3.914285714285714, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A160NEMUNL9RYV', iid='B000FOYMKU', r_ui=3.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='ALCAUHCUP0OH7', iid='B000BAX50G', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2TWNR1G92O51V', iid='B00029U1DK', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2CX79FL19TP46', iid='B00012OJ22', r_ui=5.0, est=4.044444444444444, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A2Z78I2972Y9IX', iid='B0007QN18U', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AFH8E68EY1XRK', iid='B00067PWFU', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1USO1P7JLSUHE', iid='B00004WCID', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3M43RY36HNZW8', iid='B0002K6Y1C', r_ui=2.0, est=4.5, details={'actual_k': 0, 'was_impossibl
e': False}),
 Prediction(uid='ALNOAS6HQ5573', iid='B000HZDF8W', r_ui=3.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2L0YVT50ZTVZM', iid='B00028ONIA', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A28KVV4QM8C17E', iid='B0002L5R78', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3B825H43ZJBCY', iid='B000BI2180', r_ui=4.0, est=3.8329979879275653, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A2TJNYT5O6BLGH', iid='B00008EM7U', r_ui=5.0, est=3.4649122807017543, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A2O4D9K0DT0DPA', iid='B00005T380', r_ui=4.0, est=4.6513409961685825, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A27UW0BS75YSBW', iid='B000CKVOOY', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2FPV4MDWR2UVV', iid='B000096B01', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AB5HWD6RXNHVV', iid='B000ARAPQW', r_ui=5.0, est=3.7248322147651005, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A1VTIPWVHERCEE', iid='B0009OAFJ8', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3QK7K1CHPCTQT', iid='B0009JZSKO', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A13D742MWRLM4X', iid='B000I4RMOU', r_ui=2.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3SD3KC3C9MXGL', iid='B00025742A', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2BDHYJD60N1AO', iid='B0001A3N6C', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2UMYQ6BAIV9DS', iid='B0002F3G7M', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1YECTLG5TG01K', iid='B000928KII', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1IYKN7YNTKGFI', iid='B000BQ7GW8', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3G8R3CAHTF9SV', iid='B0003NN83U', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='ANTR66FRJH7CE', iid='B00006B9HC', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AXIPWBZ0B1B60', iid='B0002D03ZW', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='ANBEYL4U22V25', iid='B00009EHJV', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2AM2NUXWIYRDM', iid='B000CS1TLE', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3135K3EQ60AA3', iid='B00009X6PT', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2KGIDFMFX03HM', iid='B000A1FFOK', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3EZJ2PDAMEAFI', iid='B000GU88CQ', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3N3EKLMQMNFHN', iid='B00011KLOI', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2FT47QVQ47F0U', iid='B000CRT07U', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1RYZRYXB6X60C', iid='B00006I53W', r_ui=5.0, est=4.875968992248062, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A3EDMRG9TK6L66', iid='B0002DG0OA', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1AO1HGGQ7F3BL', iid='B000A6NV0U', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AMZTVXGJ0OMXO', iid='B0002U1TJY', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1D5URI0CZY6WX', iid='B000C1Z0HA', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AWKGVBDPO4YXS', iid='B0002BEUZ8', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1NEL661KNJR4A', iid='B000CKVOOY', r_ui=5.0, est=4.710244648318043, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A2F6S6RODPDJJL', iid='B0007XJSQC', r_ui=3.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2WM1KL9DQSQ18', iid='B000085ZKV', r_ui=4.0, est=2.356164383561644, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='AS6HY92W5YQH', iid='B000F7JCRA', r_ui=1.0, est=4.25, details={'actual_k': 0, 'was_impossible':
 False}),
 Prediction(uid='A2A57RXJ3MMT9J', iid='B00007EDZG', r_ui=4.0, est=4.430693069306931, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A354G6DF1931AQ', iid='B00066IJPQ', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1PQ3FIGSFTMN7', iid='B00006B82N', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2W5Y9BTIA9ASA', iid='B000BP50AY', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3777T16LVJEUS', iid='B0007WK8LC', r_ui=3.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3B0TX0QRN7WQ9', iid='B000EVM5DU', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A207LJRAH17N7I', iid='B00069BVIK', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='ANTUN5NU3UP80', iid='B000HZEX1K', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A252F712DXWRSB', iid='B000CSWHCY', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A37WS5LGXSHKVT', iid='B000BQ7GW8', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A29WKHA2KHB4NE', iid='B000IBLMHQ', r_ui=5.0, est=4.375, details={'actual_k': 0, 'was_impossibl
e': False}),
 Prediction(uid='A28P96T5G5EL8W', iid='B0007VMROE', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1AXJTN6GPQ0I', iid='B00009R98A', r_ui=4.0, est=4.43859649122807, details={'actual_k': 0, 'wa
s_impossible': False}),
 Prediction(uid='A3BB0VCA1RV7VF', iid='B000EPLP3C', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2FDV397XG60OQ', iid='B000067RC4', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1W08VMZJMBB13', iid='B000EPLP3C', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3RIYAAACTYD3S', iid='B00005N6KG', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A162BMXVYNP58A', iid='B00008ZPJU', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AQR3OG0KHIQ6R', iid='B00064O1P0', r_ui=5.0, est=4.131868131868132, details={'actual_k': 0, 'wa
s_impossible': False}),
 Prediction(uid='A1DBFOH9EC2U5E', iid='B0007WZLDC', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3T3OSRNXHRA0D', iid='B000EJVWGS', r_ui=4.0, est=4.5131578947368425, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A19PUPLQ30WRHX', iid='B000117DF8', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A34KMXMZTKPF45', iid='B000B9O83A', r_ui=5.0, est=4.414893617021277, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A1M6KQM4W2G7JK', iid='B000ANGNN0', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2VW4OJZ72UYMV', iid='B000IBNXX2', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1KIOMFS41P2FI', iid='B000BSLTIS', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3SSDQFBBVDGRS', iid='B000DZDP9K', r_ui=5.0, est=3.6444444444444444, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A1SJ4E4I684N6M', iid='972683275', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1X4PMDX0HSIIU', iid='B00080BOKQ', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1GJ35HP82C0P', iid='B000BYCKUI', r_ui=5.0, est=4.095238095238095, details={'actual_k': 0, 'wa
s_impossible': False}),
 Prediction(uid='A2WSOU3H2ESOLY', iid='B00005ARK3', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2XY732R0AEV2A', iid='B0007U0IQE', r_ui=3.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A27SG07M1HW2XN', iid='B0000A98AC', r_ui=2.0, est=3.757575757575758, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='AIBYKX9DVNYRK', iid='B000BTH36E', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A32X30YGYWDTC2', iid='B000H7GFWS', r_ui=2.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1S80XSY3I47OQ', iid='B00004SABB', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='ACP6WVC7URYG4', iid='B000BWACX2', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='AZXLIYCZHA742', iid='B0009Y7APU', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1MX4N5OTYH0GF', iid='B0007QU6WY', r_ui=3.0, est=4.563380281690141, details={'actual_k': 0, 'w
as_impossible': False}),
 Prediction(uid='A25L5PKR14GXAI', iid='B0000642RX', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A2R9DK7JEEQD2D', iid='B000067RVL', r_ui=5.0, est=4.753353666436509, details={'actual_k': 1, 'w
as_impossible': False}),
 Prediction(uid='A1T5PHQ5R6G8LG', iid='B000HZDF8W', r_ui=1.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1YT4DXJ4CMFKG', iid='B0006TI9WG', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A1FMZHUO378RO8', iid='B00004Z5M1', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A3OMCYZ9JARADX', iid='B00005AXHW', r_ui=5.0, est=4.5141242937853105, details={'actual_k': 0,
 'was_impossible': False}),
 Prediction(uid='A2CMUPW76Z8C3M', iid='B000FOYMKU', r_ui=5.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 Prediction(uid='A25WGPSWTK9EP1', iid='B000CSOXTO', r_ui=4.0, est=4.050870065449826, details={'was_impossible':
 True, 'reason': 'User and/or item is unknown.'}),
 ...]
```

6. Evaluate the models.

(Once the model is trained on the training data, it can be used to compute the error (RMSE) on predictions made on the test data).

In [39]:

```python
# get RMSE
print("Item-based Model : Test Set")
accuracy.rmse(test_pred, verbose=True)
```

```
Item-based Model : Test Set
RMSE: 1.3436
```

Out[39]:

```
1.343641161111319
```

7. Get top - K (K = 5) recommendations. Since our goal is to recommend new products for each user based on his/her habits, we will recommend 5 new products

In [40]:

```python
new_df1=new_df.head(10000)
ratings_matrix = new_df1.pivot_table(values='Rating', index='userId', columns='productId', fill_value=0)
ratings_matrix.head()
```

Out[40]:

productId	1400501466	1400501520	1400501776	1400532620	1400532655	140053271X	1400532736	1400599997	140069
userId									
A01852072Z7B68UHLI5UG	0	0	0	0	0	0	0	0	
A0266076X6KPZ6CCHGVS	0	0	0	0	0	0	0	0	
A0293130VTX2ZXA70JQS	0	0	0	0	0	0	0	0	
A030530627MK66BD8V4LN	0	0	0	0	0	0	0	0	
A0571176384K8RBNKGF8O	0	0	0	0	0	0	0	0	

5 rows × 76 columns

As expected, the utility matrix obtaned above is sparce, I have filled up the unknown values wth 0.

In [41]:

```python
ratings_matrix.shape
```

Out[41]:

```
(9832, 76)
```

Transposing the matrix

In [42]:

```python
X = ratings_matrix.T
X.head()
```

Out[42]:

userId	A01852072Z7B68UHLI5UG	A0266076X6KPZ6CCHGVS	A0293130VTX2ZXA70JQS	A030530627MK66BD8V4LN	A0571176384K8RBNKG
productId					
1400501466	0	0	0	0	
1400501520	0	0	0	0	
1400501776	0	0	0	0	
1400532620	0	0	0	0	
1400532655	0	0	0	0	

5 rows × 9832 columns

In [43]:

```python
X.shape
```

Out[43]:

```
(76, 9832)
```

Unique products in subset of data

In [44]:

```python
X1 = X
```

In [45]:

```python
#Decomposing the Matrix
from sklearn.decomposition import TruncatedSVD
SVD = TruncatedSVD(n_components=10)
decomposed_matrix = SVD.fit_transform(X)
decomposed_matrix.shape
```

Out[45]:

```
(76, 10)
```

In [46]:

```python
#Correlation Matrix

correlation_matrix = np.corrcoef(decomposed_matrix)
correlation_matrix.shape
```

Out[46]:

```
(76, 76)
```

In [47]:

```python
X.index[75]
```

Out[47]:

```
'B00000K135'
```

Index # of product ID purchased by customer

In [48]:

```python
i = "B00000K135"

product_names = list(X.index)
product_ID = product_names.index(i)
product_ID
```

Out[48]:

```
75
```

Correlation for all items with the item purchased by this customer based on items rated by other customers people who bought the same product

In [49]:

```python
correlation_product_ID = correlation_matrix[product_ID]
correlation_product_ID.shape
```

Out[49]:

```
(76,)
```

Recommending top 5 highly correlated products in sequence

In [50]:

```python
Recommend = list(X.index[correlation_product_ID > 0.90])

# Removes the item already bought by the customer
Recommend.remove(i)

Recommend[0:4]
```

Out[50]:

```
['1400599997', '9575871979', '9625993428', '9985511476']
```

Here are the top 05 products to be displayed by the recommendation system to the above customer based on the purchase history of other customers in the website. '1400501776', '8862935293', 'B00000JBAT', 'B00000JBHP' are the top recommended products for the customer who purchased a product 'B00000K135'. The most popular product recommended to the user is : B0002L5R78

In [ ]: