Import libraries

In [21]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
electronics_data=pd.read_csv("./ratings_Electronics (1).csv",names=['userId', 'p
```

In [3]:
```
#display the data
electronics_data.head()
```

Out[3]:

	userId	productId	Rating	timestamp
0	AKM1MP6P0OYPR	0132793040	5.0	1365811200
1	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
2	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
3	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
4	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200

In [5]:
```
#Shape of the data
electronics_data.shape
```

Out[5]:
```
(7824482, 4)
```

In [6]:
```
#Taking subset of the dataset
electronics_data=electronics_data.iloc[:1048576,0:]
```

In [7]:
```
#Check the datatypes
electronics_data.dtypes
```

Out[7]:
```
userId        object
productId     object
Rating       float64
timestamp      int64
dtype: object
```

In [8]:
```
electronics_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048576 entries, 0 to 1048575
Data columns (total 4 columns):
userId       1048576 non-null object
productId    1048576 non-null object
Rating       1048576 non-null float64
timestamp    1048576 non-null int64
dtypes: float64(1), int64(1), object(2)
memory usage: 32.0+ MB
```

In [9]:
```
#Five point summary
electronics_data.describe()['Rating'].T
```

Out[9]:
```
count    1.048576e+06
mean     3.973380e+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 [10]:
```
#Find the minimum and maximum ratings
print('Minimum rating is: %d' %(electronics_data.Rating.min()))
print('Maximum rating is: %d' %(electronics_data.Rating.max()))
```

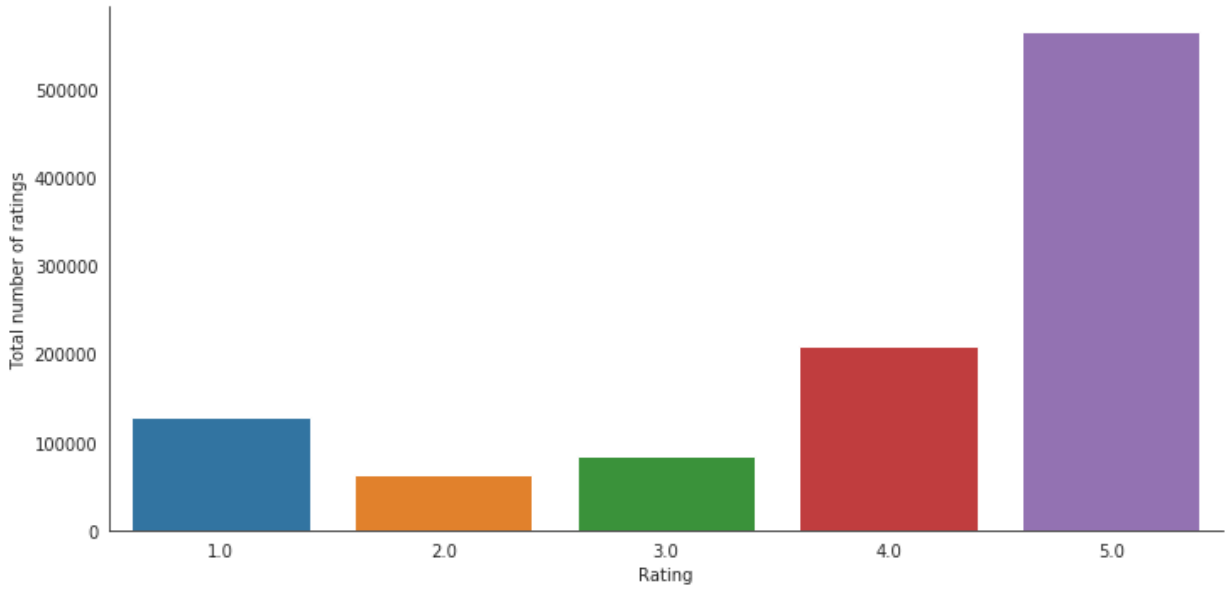
```
Minimum rating is: 1
Maximum rating is: 5
```

In [11]:
```
#Check for missing values
print('Number of missing values across columns: \n',electronics_data.isnull().su
```

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

In [13]:
```
# 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\tejaswini.buddha\AppData\Local\Continuum\anaconda3\lib\site-packages\seaborn\categorical.py:3669: 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)
```



Unique Users and products

In [16]:
```
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 : 1048576
Total No of Users   : 786330
Total No of products  : 61894
```

Dropping the TimeStamp Column

In [17]:
```
#Dropping the Timestamp column
electronics_data.drop(['timestamp'], axis=1,inplace=True)
```

Analyzing the rating

In [18]:
```
#Analysis of rating given by the user
no_of_rated_products_per_user = electronics_data.groupby(by='userId')['Rating'].
no_of_rated_products_per_user.head()
```

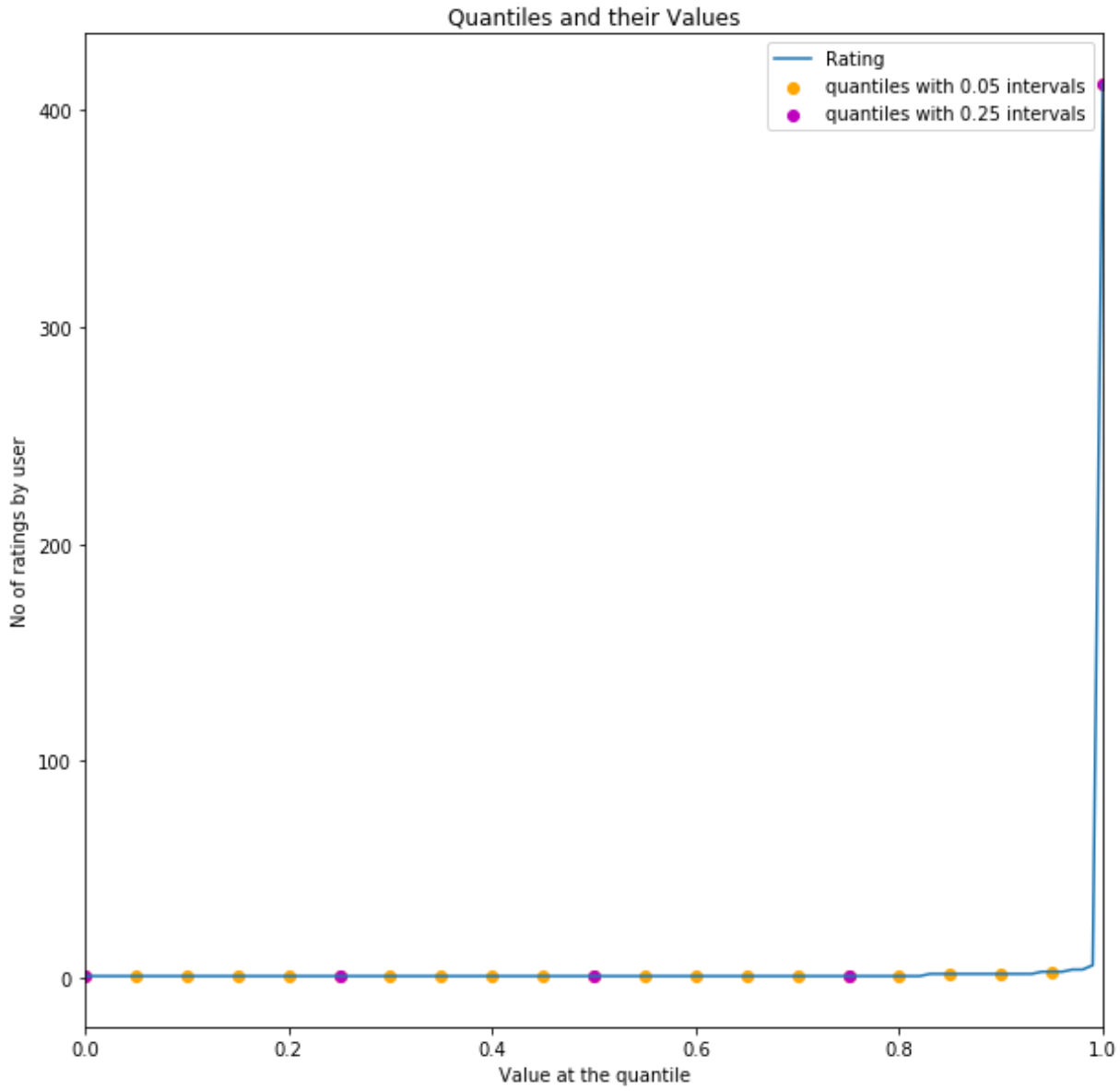
Out[18]:
```
userId
A5JLAU2ARJ0BO     412
A231WM2Z2JL0U3    249
A25HBO5V8S8SEA    164
A6FIAB28IS79      146
AT6CZDCP4TRGA     128
Name: Rating, dtype: int64
```

In [19]:
```
no_of_rated_products_per_user.describe()
```

Out[19]:
```
count    786330.000000
mean          1.333506
std           1.385612
min           1.000000
25%           1.000000
50%           1.000000
75%           1.000000
max         412.000000
Name: Rating, dtype: float64
```

In [20]:
```
quantiles = no_of_rated_products_per_user.quantile(np.arange(0,1.01,0.01), interp
```

In [22]:
```
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 0.25 difference
plt.scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m', label = "q
plt.ylabel('No of ratings by user')
plt.xlabel('Value at the quantile')
plt.legend(loc='best')
plt.show()
```



In [23]:
```
print('\n No of rated product more than 50 per user : {}\n'.format(sum(no_of_rate
```

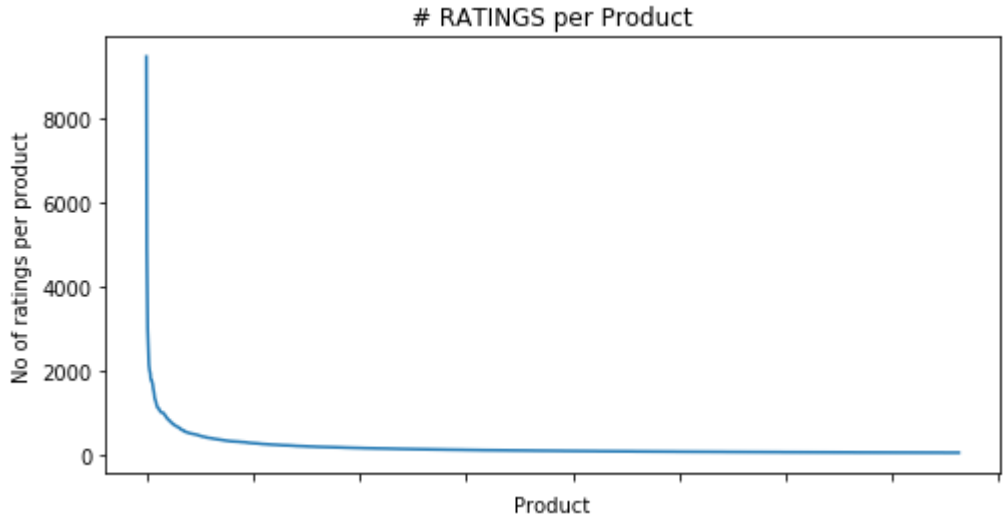
```
 No of rated product more than 50 per user : 38
```

In [24]:
```
#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()
```

In [25]:
```
no_of_ratings_per_product = new_df.groupby(by='productId')['Rating'].count().sor

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

plt.show()
```



In [26]:
```
#Average rating of the product
new_df.groupby('productId')['Rating'].mean().head()
```

Out[26]:
```
productId
0972683275    4.470980
1400501466    3.560000
1400501520    4.243902
1400501776    3.884892
1400532620    3.684211
Name: Rating, dtype: float64
```

In [27]:
```
new_df.groupby('productId')['Rating'].mean().sort_values(ascending=False).head()
```

Out[27]:
```
productId
B0000DYV9H    4.947368
B000053HC5    4.945783
B00009R96C    4.885714
B00005LE76    4.879310
B000I1X3W8    4.869565
Name: Rating, dtype: float64
```

In [28]:
```
#Total no of rating for product
new_df.groupby('productId')['Rating'].count().sort_values(ascending=False).head(
```

Out[28]:
```
productId
B0002L5R78    9487
B0001FTVEK    5345
B000I68BD4    4903
B000BQ7GW8    4275
B00007E7JU    3523
Name: Rating, dtype: int64
```

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

In [30]:
```
ratings_mean_count['rating_counts'] = pd.DataFrame(new_df.groupby('productId')['
```

In [31]:
```
ratings_mean_count.head()
```

Out[31]:

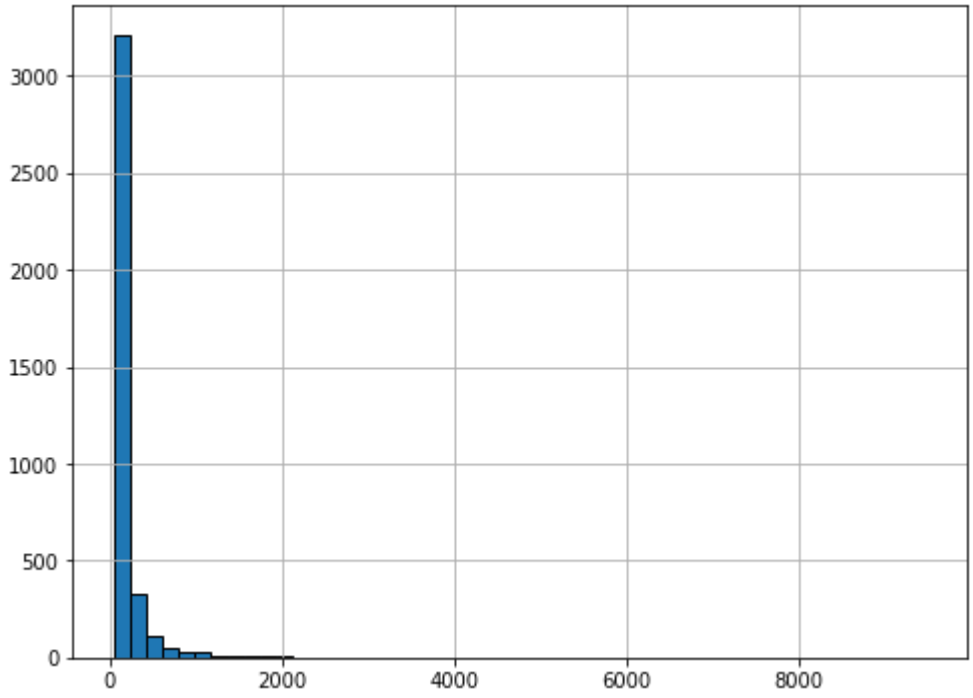
productId	Rating	rating_counts
0972683275	4.470980	1051
1400501466	3.560000	250
1400501520	4.243902	82
1400501776	3.884892	139
1400532620	3.684211	171

In [32]:
```
ratings_mean_count['rating_counts'].max()
```

Out[32]:
```
9487
```

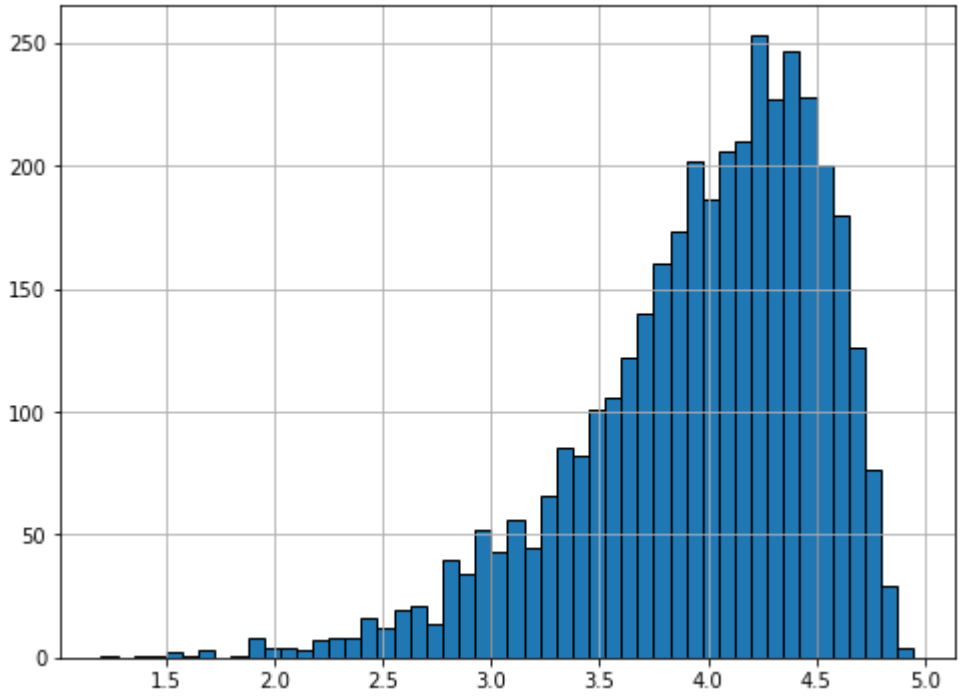
In [33]:
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['rating_counts'].hist(bins=50)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c40324d828>
```



In [34]:
```
plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['Rating'].hist(bins=50)
```

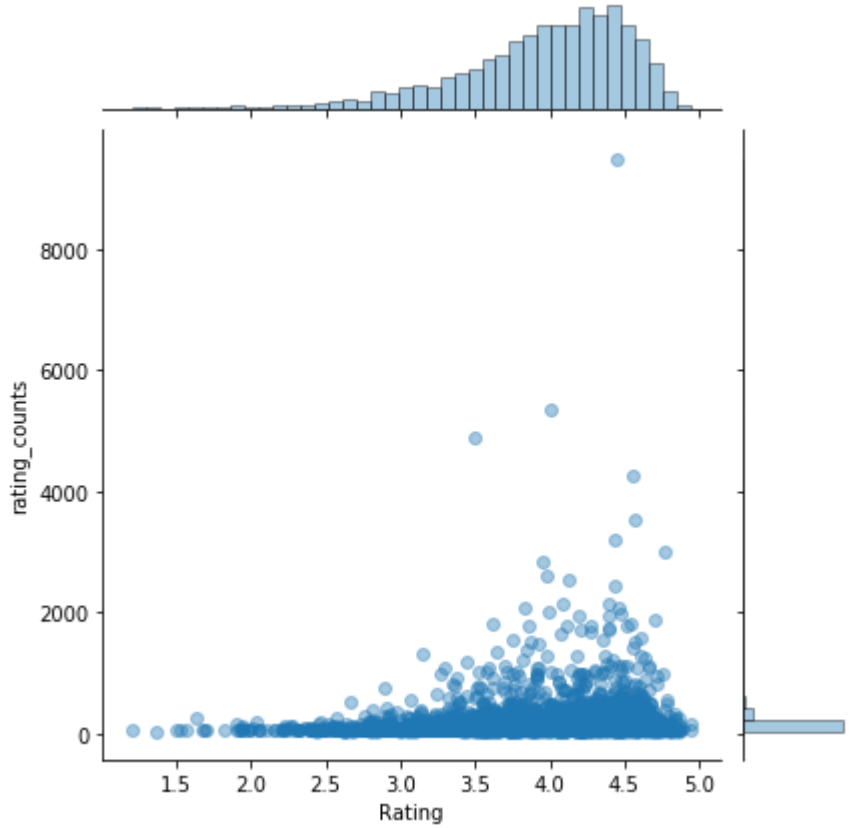
Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c402ac9b00>
```



In [35]:
```
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)
```

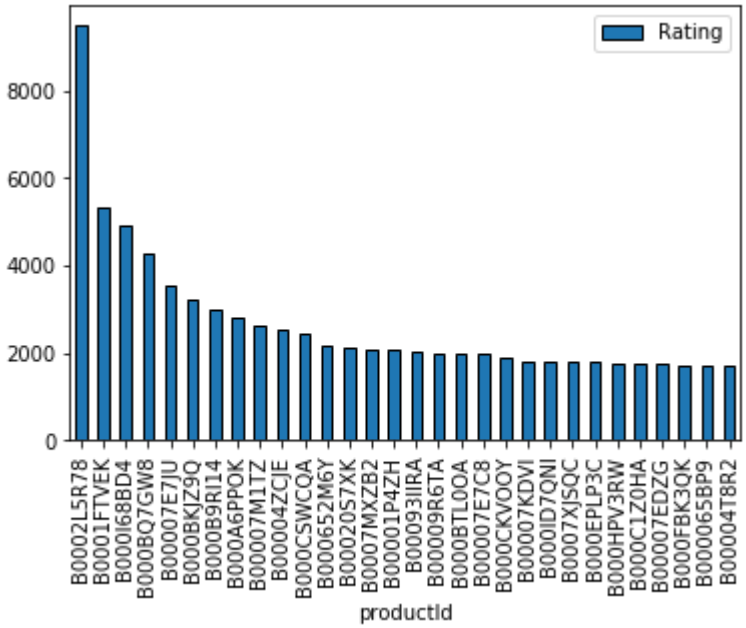
Out[35]:
```
<seaborn.axisgrid.JointGrid at 0x1c402b81470>

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



In [36]:
```
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[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1c402c6e128>



Collaberative filtering

In [39]:
```
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 [40]:
```
#Reading the dataset
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(new_df,reader)
```

In [44]:
```
#Splitting the dataset
trainset, testset = train_test_split(data, test_size=0.3,random_state=10)
```

In [45]:
```
# Use user_based true/false to switch between user-based or item-based collaborat
algo = KNNWithMeans(k=5, sim_options={'name': 'pearson_baseline', 'user_based': F
algo.fit(trainset)
```

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

Out[45]:
```
<surprise.prediction_algorithms.knns.KNNWithMeans at 0x1c402bad748>
```

In [46]:
```
# run the trained model against the testset
test_pred = algo.test(testset)
```

In [47]:
```
test_pred
```

```
own.'}),
 Prediction(uid='A1F9Z42CFF9IAY', iid='B00007E7QS', r_ui=4.0, est=4.56756756
75675675, details={'actual_k': 0, 'was_impossible': False}),
 Prediction(uid='A127XYGM306P84', iid='B00004TVSP', r_ui=5.0, est=4.05087006
5449826, details={'was_impossible': True, 'reason': 'User and/or item is unk
own.'}),
 Prediction(uid='A3RIBQ1ATEFVI0', iid='B000FBK3QK', r_ui=2.0, est=4.05087006
5449826, details={'was_impossible': True, 'reason': 'User and/or item is unk
own.'}),
 Prediction(uid='A3JBAGW91MOW56', iid='B000GYU9IS', r_ui=5.0, est=3.39036685
68547927, details={'actual_k': 1, 'was_impossible': False}),
 Prediction(uid='A35G1B3GVJQ253', iid='B00005T6GZ', r_ui=5.0, est=4.24074074
07407405, details={'actual_k': 0, 'was_impossible': False}),
 Prediction(uid='A2NG92YJ5B9T0G', iid='B000BOCUUI', r_ui=1.0, est=4.05087006
5449826, details={'was_impossible': True, 'reason': 'User and/or item is unk
own.'}),
 Prediction(uid='AQIFC6JHGOVM5', iid='B000BVB2FK', r_ui=4.0, est=4.629032258
064516, details={'actual_k': 0, 'was_impossible': False}),
 Prediction(uid='A31CT320VGN2GX', iid='B0001FV30K', r_ui=4.0, est=4.05087006
5449826, details={'was_impossible': True, 'reason': 'User and/or item is unk
```

In [48]:
```
# get RMSE
print("Item-based Model : Test Set")
accuracy.rmse(test_pred, verbose=True)
```

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

Out[48]:
```
1.343641161111319
```

Model based collabarative filtering system

In [49]:
```
new_df1=new_df.head(10000)
ratings_matrix = new_df1.pivot_table(values='Rating', index='userId', columns='pr
ratings_matrix.head()
```

Out[49]:

productId	0972683275	1400501466	1400501520	1400501776	1400532620	1400532(
userId						
A01852072Z7B68UHLI5UG	0	0	0	0	0	
A0266076X6KPZ6CCHGVS	0	0	0	0	0	
A0293130VTX2ZXA70JQS	5	0	0	0	0	
A030530627MK66BD8V4LN	4	0	0	0	0	
A0571176384K8RBNKGF8O	0	0	0	0	0	

5 rows × 76 columns

In [50]:
```
ratings_matrix.shape
```

Out[50]:
```
(9832, 76)
```

In [51]:
```
X = ratings_matrix.T
X.head()
```

Out[51]:

userId	A01852072Z7B68UHLI5UG	A0266076X6KPZ6CCHGVS	A0293130VTX2ZXA70JQS	A030530
productId				
0972683275	0	0	5	
1400501466	0	0	0	
1400501520	0	0	0	
1400501776	0	0	0	
1400532620	0	0	0	

5 rows × 9832 columns

In [52]:
```
X.shape
```

Out[52]:
```
(76, 9832)
```

In [53]:
```
X1 = X
```

In [54]:
```
#Decomposing the Matrix
from sklearn.decomposition import TruncatedSVD
SVD = TruncatedSVD(n_components=10)
decomposed_matrix = SVD.fit_transform(X)
decomposed_matrix.shape
```

Out[54]: (76, 10)

In [55]:
```
#Correlation Matrix
correlation_matrix = np.corrcoef(decomposed_matrix)
correlation_matrix.shape
```

Out[55]: (76, 76)

Index # of product ID purchased by customer

In [56]:
```
X.index[75]
```

Out[56]: 'B00000K135'

In [57]:
```
i = "B00000K135"

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

Out[57]: 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 [58]:
```
correlation_product_ID = correlation_matrix[product_ID]
correlation_product_ID.shape
```

Out[58]: (76,)

In [59]:
```
Recommend = list(X.index[correlation_product_ID > 0.65])
# Removes the item already bought by the customer
Recommend.remove(i)
Recommend[0:24]
```

Out[59]:
```
['8862935293',
 '9573212919',
 '9888002198',
 '9966694544',
 '9983891212',
 '9985511476',
 'B000001OM4',
 'B00000J08Q',
 'B00000J0D5',
 'B00000J1EQ',
 'B00000J1F3',
 'B00000J1SC',
 'B00000J3UJ',
 'B00000JCTO',
 'B00000JFE3',
 'B00000JMUG']
```

Here are the top 10 products to be displayed by the recommendation system to the above customer based on the purchase history of other customers in the website.