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

In [46]:
```
booksdf = pd.read_csv('C:/Users/User/Desktop/Data Science Notes/Project/PC DS - M
userdf = pd.read_csv('C:/Users/User/Desktop/Data Science Notes/Project/PC DS - Ma
rating = pd.read_csv('C:/Users/User/Desktop/Data Science Notes/Project/PC DS - Ma
```

In [47]:
```
booksdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   isbn                 10000 non-null  object
 1   book_title           10000 non-null  object
 2   book_author          10000 non-null  object
 3   year_of_publication  10000 non-null  object
 4   publisher            10000 non-null  object
dtypes: object(5)
memory usage: 390.8+ KB
```

In [48]:
```
rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   user_id  10000 non-null  int64 
 1   isbn     10000 non-null  object
 2   rating   10000 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

In [49]:
```
rating.isbn.value_counts()
```

Out[49]:
```
971880107     17
316666343      9
440225701      9
380731851      7
804106304      7
              ..
60002492       1
60958022       1
887926396X     1
8804491523     1
425164403      1
Name: isbn, Length: 9335, dtype: int64
```

In [50]:
```
rating.user_id.value_counts()
```

Out[50]:
```
278418    4533
277427     497
277639     270
277478     214
278188     197
          ...
277838       1
277840       1
277841       1
277842       1
276725       1
Name: user_id, Length: 941, dtype: int64
```

In [51]:
```
rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   user_id  10000 non-null  int64
 1   isbn     10000 non-null  object
 2   rating   10000 non-null  int64
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

In [52]:
```
booksdf['isbn']
```

Out[52]:
```
0        195153448
1          2005018
2         60973129
3        374157065
4        393045218
           ...
9995     140283404
9996     380730774
9997     862418879
9998    340414645X
9999    3442730988
Name: isbn, Length: 10000, dtype: object
```

In [53]:
```
rating_dict = {'itemID' : rating['isbn'], 'userID' : rating['user_id'], 'rating'
```

In [54]:
```
df = pd.DataFrame(rating_dict)
```

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   itemID  10000 non-null  object
 1   userID  10000 non-null  int64 
 2   rating  10000 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

In [56]:
```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate
from surprise.reader import Reader
reader = Reader(rating_scale=(1, 5))
```

In [57]:
```
reader
```

Out[57]:
```
<surprise.reader.Reader at 0x9607667250>
```

In [58]:
```
data = Dataset.load_from_df(df[['userID', 'itemID', 'rating']], reader)
algo = SVD()        # Singular Value Decomposition
cross_validate(algo, data, measures=['RMSE', 'MSE', 'MAE'], cv=2)
```

Out[58]:
```
{'test_rmse': array([3.06599906, 2.91693236]),
 'test_mse': array([9.40035023, 8.50849441]),
 'test_mae': array([2.36334652, 2.28852521]),
 'fit_time': (0.9477307796478271, 0.8466024398803711),
 'test_time': (0.0840599536895752, 0.07455277442932129)}
```

In [59]:
```
# Finding Null Values
df.isna().sum()
```

Out[59]:
```
itemID    0
userID    0
rating    0
dtype: int64
```

In [60]:
```
# Read the data where ratings are given by users
df[['userID','rating']].head(50)
```

Out[60]:

	userID	rating
0	276725	0
1	276726	5
2	276727	0
3	276729	3
4	276729	6
5	276733	0
6	276736	8
7	276737	6
8	276744	7
9	276745	10
10	276746	0
11	276746	0
12	276746	0
13	276746	0
14	276746	0
15	276746	0
16	276747	9
17	276747	0
18	276747	0
19	276747	9
20	276747	8
21	276747	7
22	276747	0
23	276747	7
24	276748	6
25	276748	0
26	276751	0
27	276751	8
28	276754	8
29	276755	5
30	276760	10
31	276762	0
32	276762	0
33	276762	5

	userID	rating
34	276762	0
35	276762	0
36	276762	0
37	276762	0
38	276762	0
39	276762	0
40	276762	0
41	276762	0
42	276762	0
43	276762	0
44	276762	8
45	276762	0
46	276762	0
47	276762	0
48	276762	0
49	276762	3

In [61]:
```
# Take a quick look at the number of unique users and books
df[['itemID','userID']].nunique()

print('Number of Unique Books : ', df['itemID'].nunique())
print('Number of Unique Users : ', df['userID'].nunique())
```

```
Number of Unique Books :  9335
Number of Unique Users :  941
```

In [62]:
```
df_zero_rating = df[df['rating']==0]
```

In [63]:
```
df_zero_rating.rating.unique()
```

Out[63]:
```
array([0], dtype=int64)
```

In [64]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   itemID  10000 non-null  object
 1   userID  10000 non-null  int64 
 2   rating  10000 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

In [65]:
```
# Remove the Items whose rating is 0 for "Data reduction"
df_sorted = df[df['rating']!=0]
df_sorted.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2646 entries, 1 to 9994
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   itemID  2646 non-null   object
 1   userID  2646 non-null   int64 
 2   rating  2646 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 82.7+ KB
```

In [66]:
```
## Convert ISBN variables to numeric numbers in the correct order
# Here df['itemID'] = rating['isbn']

item_list = df.itemID.unique()
print('Length of Item List : ',len(item_list))
def get_item_numeric_id(item):
    itemindex = np.where(item_list==item)
    return itemindex[0][0]
```

```
Length of Item List :  9335
```

In [67]:
```
## Convert the user_id variable to numeric numbers in the correct order

user_list = df.userID.unique()
print('Length of User List : ', len(user_list))
def get_user_id_numeric_id(user_id):
    itemindex = np.where(user_list==user_id)
    return itemindex[0][0]
```

```
Length of User List :  941
```

In [73]:
```
# Converting user_id variable to numeric number in correct order
df['user_ID'] = df['userID'].apply(get_user_id_numeric_id)
df.head()
```

Out[73]:

	itemID	userID	rating	user_ID	item_id
0	034545104X	276725	0	0	0
1	155061224	276726	5	1	1
2	446520802	276727	0	2	2
3	052165615X	276729	3	3	3
4	521795028	276729	6	3	4

In [74]:
```
# Converting ISBN/item_id variable to numeric number in correct order
df['item_id'] = df['itemID'].apply(get_item_numeric_id)
df.head()
```

Out[74]:

	itemID	userID	rating	user_ID	item_id
0	034545104X	276725	0	0	0
1	155061224	276726	5	1	1
2	446520802	276727	0	2	2
3	052165615X	276729	3	3	3
4	521795028	276729	6	3	4

In [76]:
```
# Re-index the columns to build a matrix
new_col_order = ['itemID','userID','user_ID', 'item_id', 'rating']
df = df.reindex(columns= new_col_order)
df.head()
```

Out[76]:

	itemID	userID	user_ID	item_id	rating
0	034545104X	276725	0	0	0
1	155061224	276726	1	1	5
2	446520802	276727	2	2	0
3	052165615X	276729	3	3	3
4	521795028	276729	3	4	6

In [78]:
```
# Split your data into two sets (training and testing)
from sklearn.model_selection import train_test_split

x = df[['user_ID','item_id']]
y = df['rating']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, train_s
```

In [79]:
```
# Make predictions based on user and item variables
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

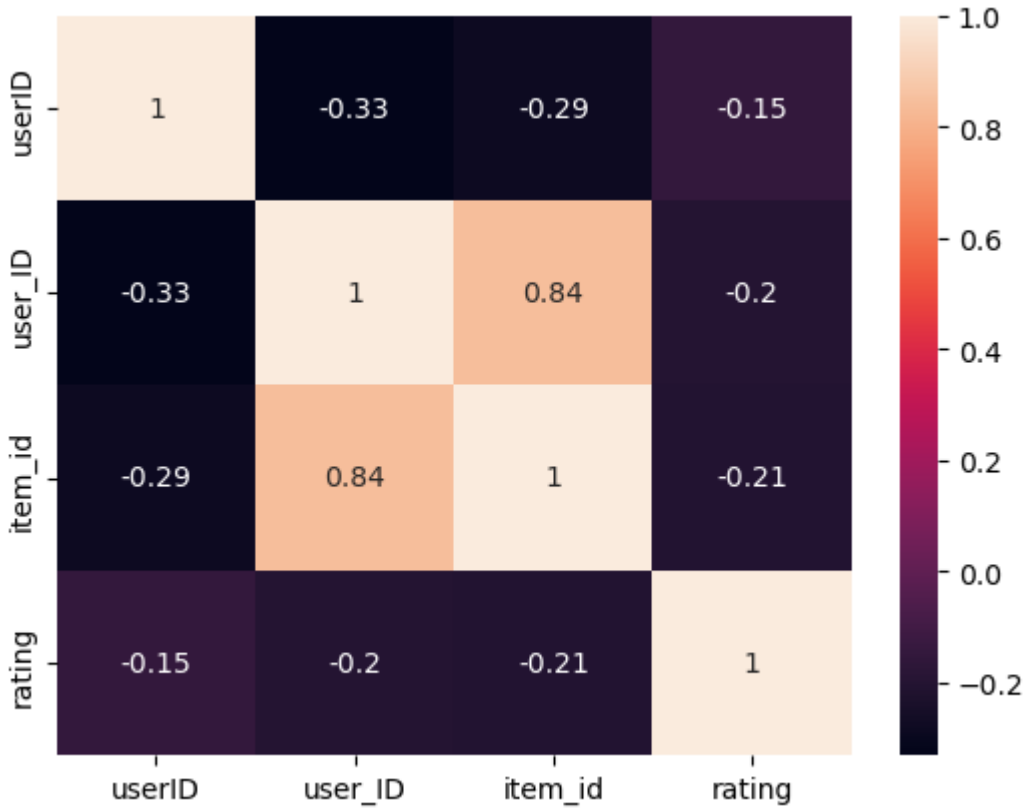
In [96]:
```
from sklearn.metrics import mean_squared_error, accuracy_score, r2_score, confusi
from math import sqrt
print('MSE : ', mean_squared_error(y_test, y_pred))
print('RMSE : ', sqrt(mean_squared_error(y_test, y_pred)))
print('Accuracy Score : ', accuracy_score(y_test, y_pred))
print('R Squared Score : ', r2_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
MSE :  15.314
RMSE :  3.913310618900575
Accuracy Score :  0.736
R Squared Score :  -0.3303218520609825
[[2208    0    0    0    0    0    0    0    0    0    0]
 [   3    0    0    0    0    0    0    0    0    0    0]
 [   6    0    0    0    0    0    0    0    0    0    0]
 [  17    0    0    0    0    0    0    0    0    0    0]
 [  21    0    0    0    0    0    0    0    0    0    0]
 [  93    0    0    0    0    0    0    0    0    0    0]
 [  83    0    0    0    0    0    0    0    0    0    0]
 [ 159    0    0    0    0    0    0    0    0    0    0]
 [ 183    0    0    0    0    0    0    0    0    0    0]
 [ 110    0    0    0    0    0    0    0    0    0    0]
 [ 117    0    0    0    0    0    0    0    0    0    0]]
              precision    recall  f1-score   support

           0       0.74      1.00      0.85      2208
           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         6
           3       0.00      0.00      0.00        17
           4       0.00      0.00      0.00        21
           5       0.00      0.00      0.00        93
           6       0.00      0.00      0.00        83
           7       0.00      0.00      0.00       159
           8       0.00      0.00      0.00       183
           9       0.00      0.00      0.00       110
          10       0.00      0.00      0.00       117

    accuracy                           0.74      3000
   macro avg       0.07      0.09      0.08      3000
weighted avg       0.54      0.74      0.62      3000


C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
```

```
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [83]:
```
sns.heatmap(df.corr(), annot=True)
```

Out[83]: <AxesSubplot:>



In [84]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

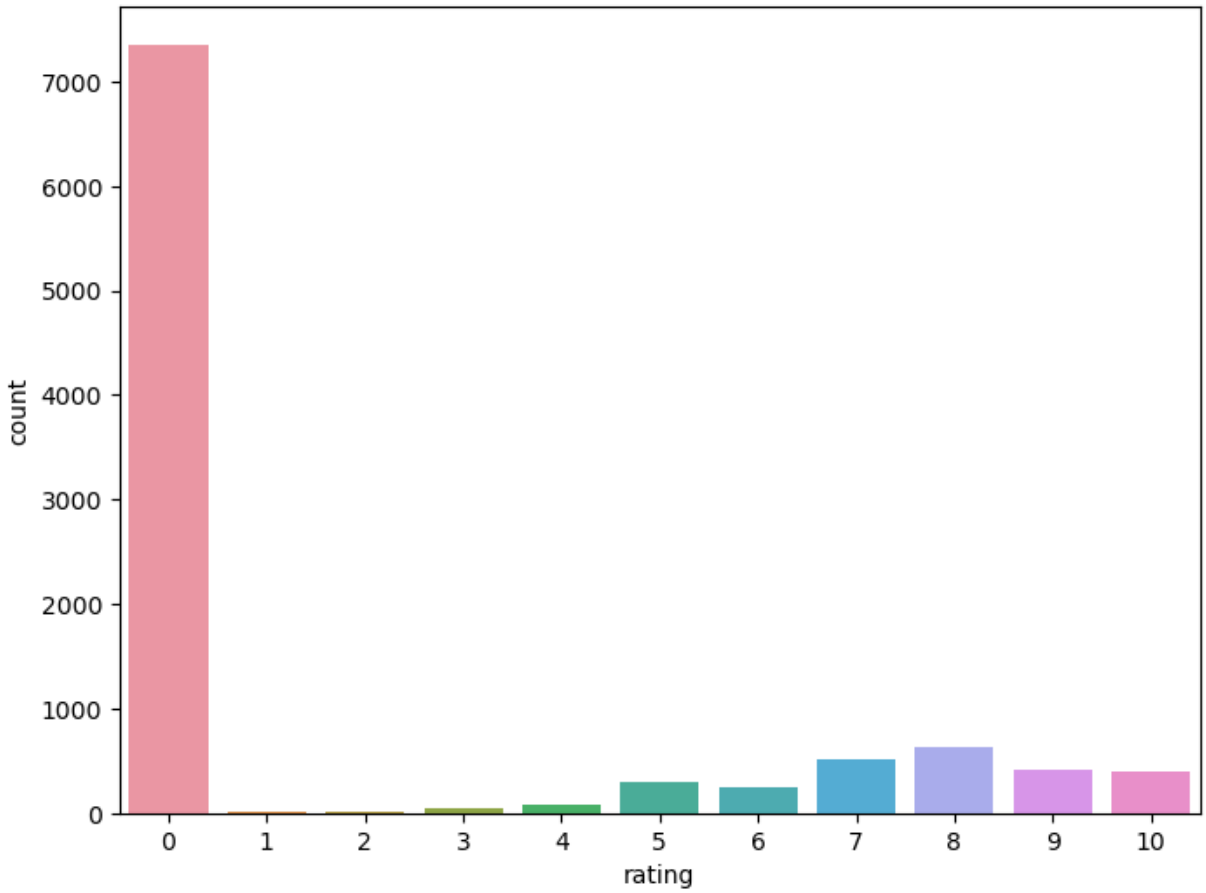
```
C:\Users\User\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:81
4: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

In [98]:
```
print('MSE : ', mean_squared_error(y_test, y_pred))
print('RMSE : ', sqrt(mean_squared_error(y_test, y_pred)))
print('Accuracy Score : ', accuracy_score(y_test, y_pred))
print('R Squared Score : ', r2_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
MSE :  15.314
RMSE :  3.913310618900575
Accuracy Score :  0.736
R Squared Score :  -0.3303218520609825
[[2208    0    0    0    0    0    0    0    0    0    0]
 [   3    0    0    0    0    0    0    0    0    0    0]
 [   6    0    0    0    0    0    0    0    0    0    0]
 [  17    0    0    0    0    0    0    0    0    0    0]
 [  21    0    0    0    0    0    0    0    0    0    0]
 [  93    0    0    0    0    0    0    0    0    0    0]
 [  83    0    0    0    0    0    0    0    0    0    0]
 [ 159    0    0    0    0    0    0    0    0    0    0]
 [ 183    0    0    0    0    0    0    0    0    0    0]
 [ 110    0    0    0    0    0    0    0    0    0    0]
 [ 117    0    0    0    0    0    0    0    0    0    0]]
              precision    recall  f1-score   support

           0       0.74      1.00      0.85      2208
           1       0.00      0.00      0.00         3
```

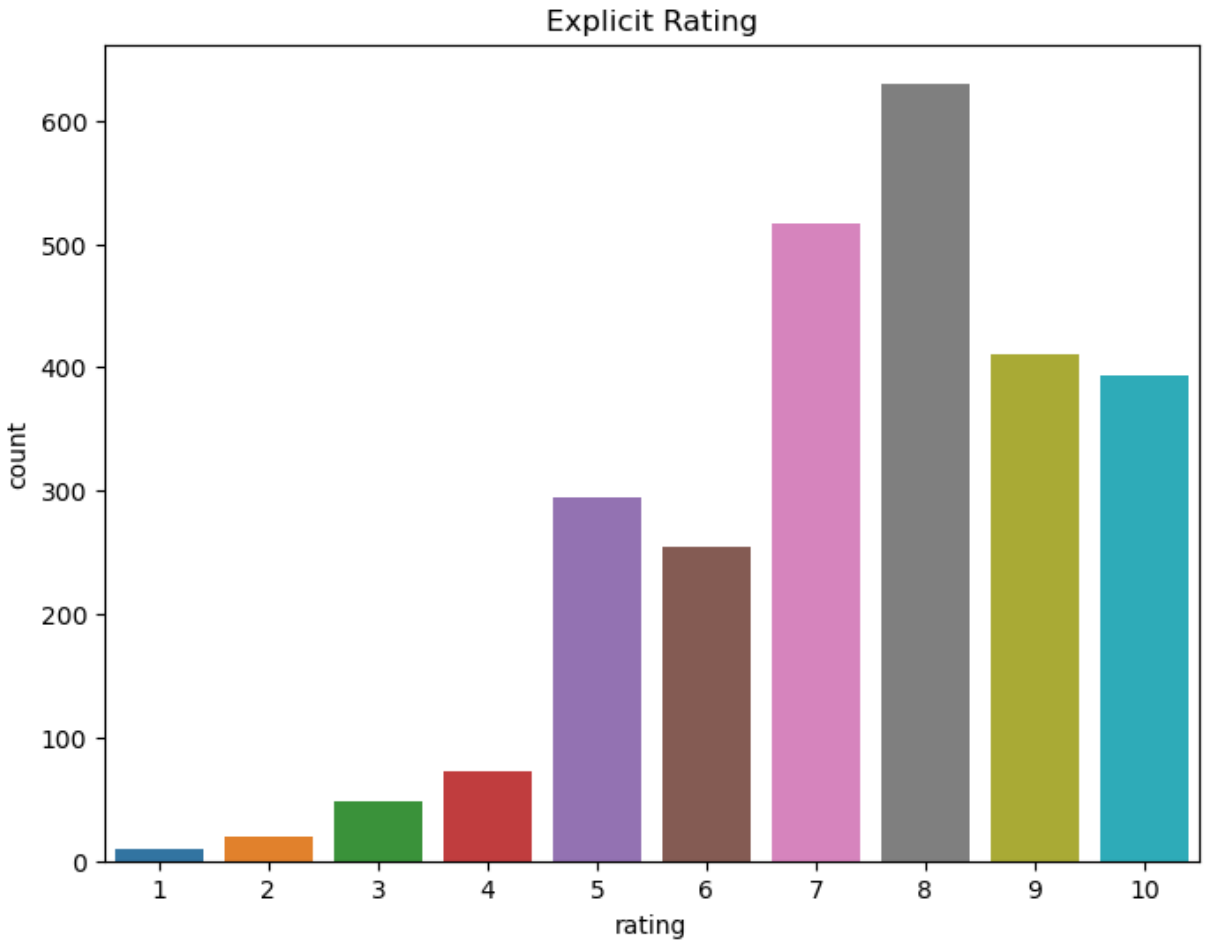
In [100]:
```
plt.figure(figsize=(8,6))
sns.countplot(x='rating', data=df)
```

Out[100]: <AxesSubplot:xlabel='rating', ylabel='count'>



In [101]:
```
plt.figure(figsize=(8,6))
data = df[df['rating']!=0]
sns.countplot(x='rating', data=data)
plt.title('Explicit Rating')
```

Out[101]: Text(0.5, 1.0, 'Explicit Rating')



In [ ]: