TASK 2- CODE CLAUSE PROJECT " MOVIE RECOMMENDATION SYSTEM"

In [3]:
```
import numpy as np  #numpy library
import pandas as pd  #pandas library
import matplotlib.pyplot as plt #pyplot
import seaborn as sns #seaborn
import plotly.express as px #for visualization
```

In [4]:
```
df = pd.read_csv("Dataset (1).csv")
df
```

Out[4]:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742
...	...	...	...	...
99998	880	476	3	880175444
99999	716	204	5	879795543
100000	276	1090	1	874795795
100001	13	225	2	882399156
100002	12	203	3	879959583

100003 rows × 4 columns

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
user_id      0
item_id      0
rating       0
timestamp    0
dtype: int64
```

In [6]:
```
df.shape # before removing null values
```

Out[6]:
```
(100003, 4)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100003 entries, 0 to 100002
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   user_id    100003 non-null  int64
 1   item_id    100003 non-null  int64
 2   rating     100003 non-null  int64
 3   timestamp  100003 non-null  int64
dtypes: int64(4)
memory usage: 3.1 MB
```

In [8]:
```
df.head(5)
```

Out[8]:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

In [9]:
```
df.dropna(how='any',inplace=True)
print(df)
```

```
        user_id  item_id  rating  timestamp
0             0       50       5  881250949
1             0      172       5  881250949
2             0      133       1  881250949
3           196      242       3  881250949
4           186      302       3  891717742
...         ...      ...     ...        ...
99998       880      476       3  880175444
99999       716      204       5  879795543
100000      276     1090       1  874795795
100001       13      225       2  882399156
100002       12      203       3  879959583

[100003 rows x 4 columns]
```

In [10]:
```
df.describe()
```

Out[10]:

	user_id	item_id	rating	timestamp
count	100003.000000	100003.000000	100003.000000	1.000030e+05
mean	462.470876	425.520914	3.529864	8.835288e+08
std	266.622454	330.797791	1.125704	5.343791e+06
min	0.000000	1.000000	1.000000	8.747247e+08
25%	254.000000	175.000000	3.000000	8.794487e+08
50%	447.000000	322.000000	4.000000	8.828269e+08
75%	682.000000	631.000000	4.000000	8.882600e+08
max	943.000000	1682.000000	5.000000	8.932866e+08

In [11]:
```
df.dtypes
```

Out[11]:
```
user_id      int64
item_id      int64
rating       int64
timestamp    int64
dtype: object
```

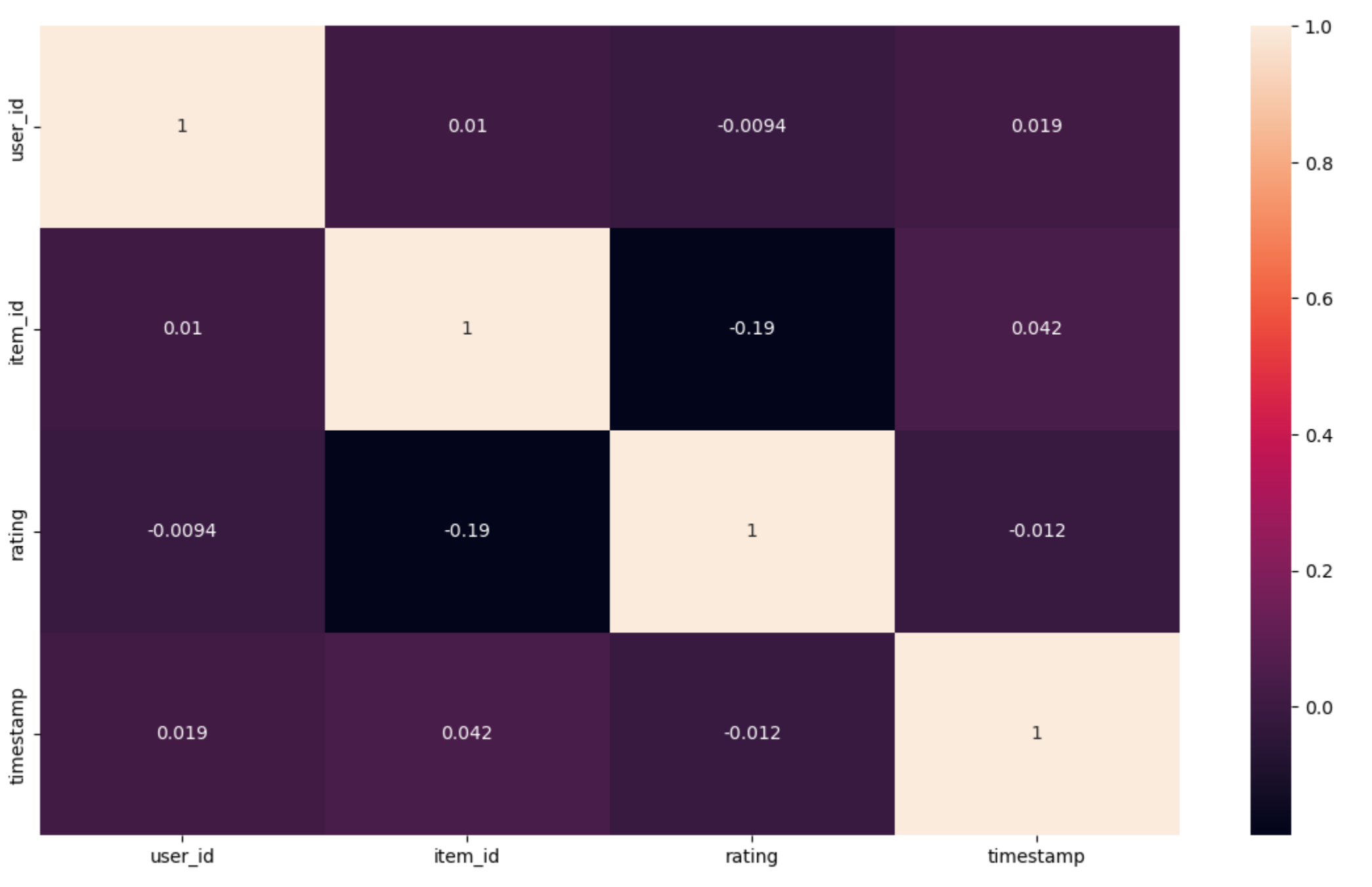
In [12]:
```
df.nunique()
```

Out[12]:
```
user_id        944
item_id       1682
rating           5
timestamp    49282
dtype: int64
```

In [13]:
```
df.index
```

Out[13]:
```
RangeIndex(start=0, stop=100003, step=1)
```

In [14]:
```
import seaborn
correlation = df.corr ()
fig=plt.figure(figsize=(14,8))
seaborn.heatmap(correlation,annot=True)
plt.show()
```



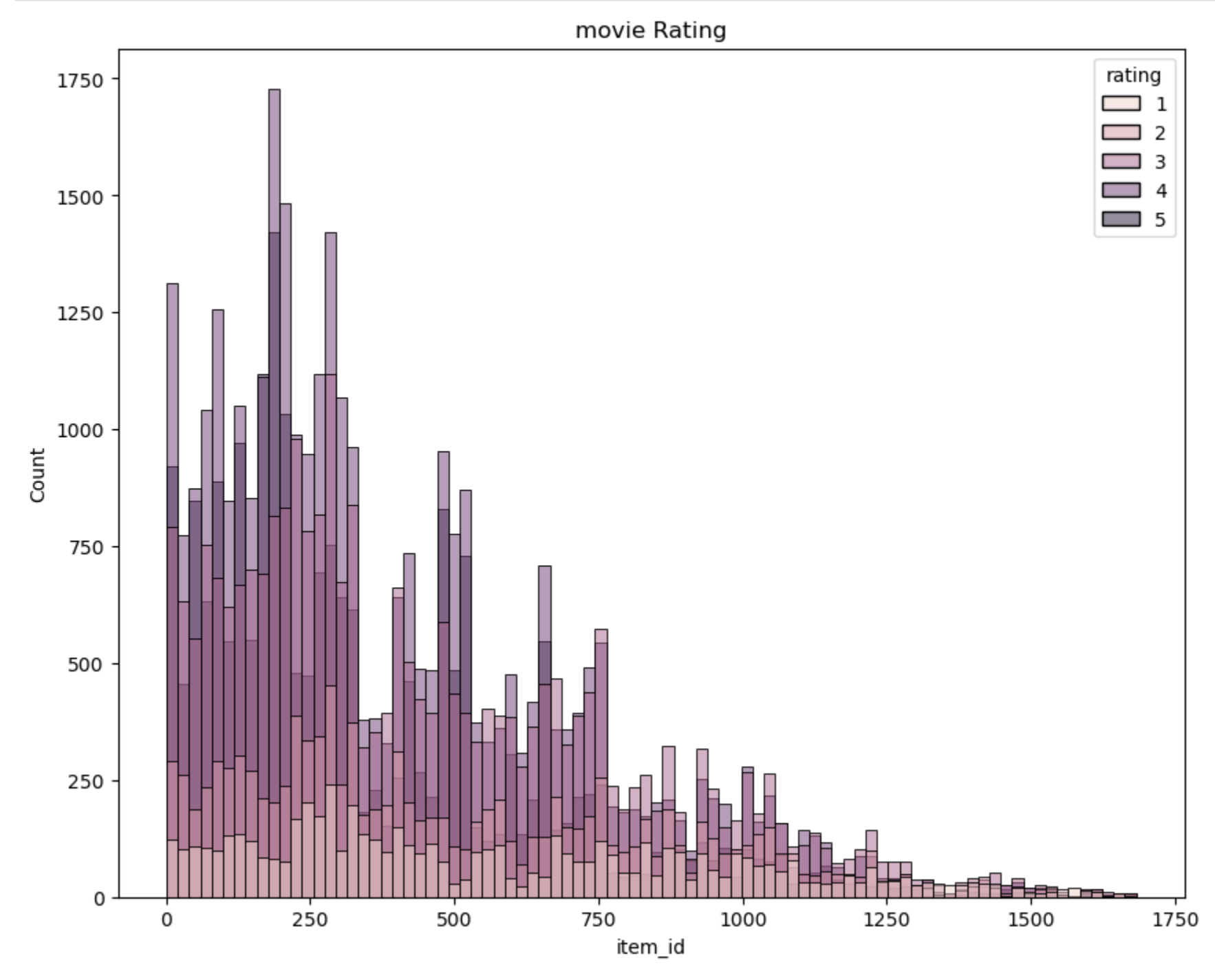
In [15]:
```
df.corr()
```

Out[15]:

	user_id	item_id	rating	timestamp
user_id	1.000000	0.010425	-0.009377	0.019103
item_id	0.010425	1.000000	-0.189119	0.041878
rating	-0.009377	-0.189119	1.000000	-0.012004
timestamp	0.019103	0.041878	-0.012004	1.000000

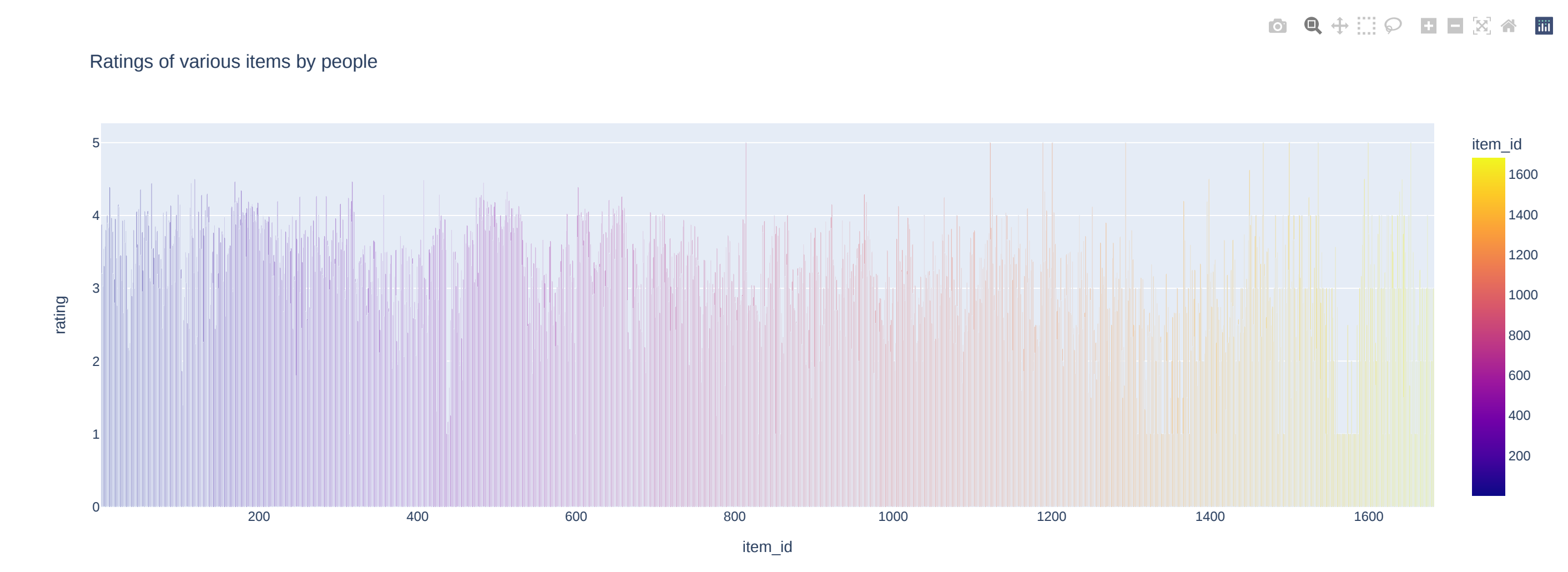
In [16]:
```
df.columns= ["user_id","item_id","rating","timestamp"]

plt.figure(figsize=(10, 8))
plt.title("movie Rating")
sns.histplot(x="item_id", hue="rating", data=df)
plt.show()
```



In [17]:
```
region = df.groupby(["item_id"])[["timestamp","user_id","rating"]].mean()
region = pd.DataFrame(region).reset_index()
fig = px.bar(region, x="item_id", y="rating", color="item_id", title="Ratings of various items by people")
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()

# barchart
```



In [18]:
```
x= df.drop(['rating'], axis=1)

y = df['rating']
print("okay")
```

```
okay
```

In [19]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)
```

In [20]:
```
x_train.shape, x_test.shape
```

Out[20]:
```
((67002, 3), (33001, 3))
```

In [21]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
print(df.shape)
```

```
(67002, 3)
(67002,)
(33001, 3)
(33001,)
(100003, 4)
```

In [22]:
```
x_train.dtypes
```

Out[22]:
```
user_id      int64
item_id      int64
timestamp    int64
dtype: object
```

In [23]:
```
#Import Libraries file

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split #Train Test Split

from sklearn.naive_bayes import GaussianNB   # Naive Bayes Classifier

from sklearn import preprocessing            # Label Encoder

from sklearn.neighbors import KNeighborsClassifier  # KNN Classsifiers
```

In [24]:
```
#Train Test split

x = df[['item_id','timestamp','user_id']]
y = df['rating']

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30, random_state=0)

x_train.shape
```

Out[24]:
```
(70002, 3)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

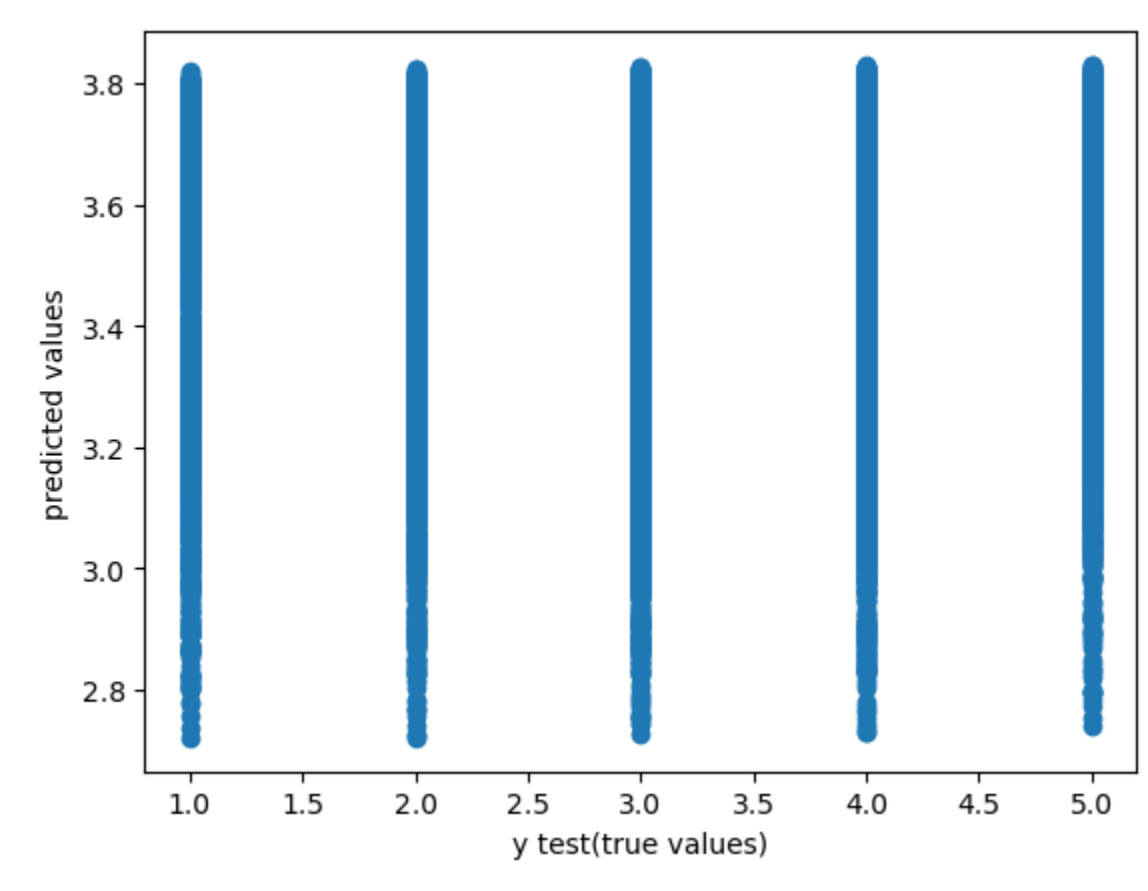
In [26]:
```
model.fit(x_train,y_train)
```

Out[26]:
```
▾ LinearRegression
LinearRegression()
```

In [27]:
```
predictions=model.predict(x_test)
```

In [28]:
```
plt.scatter(y_test,predictions)
plt.xlabel('y test(true values)')
plt.ylabel('predicted values')
```

Out[28]:
```
Text(0, 0.5, 'predicted values')
```



In [29]:
```
model.score(x_test,y_test)
```

Out[29]:
```
0.037920234505267825
```

In [30]:
```
print("ACCURACY IS:",model.score(x_test,y_test)*100)
```

```
ACCURACY IS: 3.7920234505267825
```

In [31]:
```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(x_train, y_train)
```

Out[31]:
```
▾ GaussianNB
GaussianNB()
```

In [32]:
```
#Predict the response for test dataset
y_pred = gnb.predict(x_test)
```

In [33]:
```
# Evaluating model
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.3461551281623946
```

In [34]:
```
# Evaluating model
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.classification_report(y_test, y_pred))
```

```
Accuracy:               precision    recall  f1-score   support

           1       0.29      0.01      0.02      1869
           2       0.00      0.00      0.00      3513
           3       0.31      0.14      0.19      8087
           4       0.35      0.90      0.51     10217
           5       0.00      0.00      0.00      6315

    accuracy                           0.35     30001
   macro avg       0.19      0.21      0.14     30001
weighted avg       0.22      0.35      0.23     30001
```

```
C:\Users\Sutharsahana\.ms-ad\lib\site-packages\sklearn\metrics\_classification.py:1344: 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.

C:\Users\Sutharsahana\.ms-ad\lib\site-packages\sklearn\metrics\_classification.py:1344: 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.

C:\Users\Sutharsahana\.ms-ad\lib\site-packages\sklearn\metrics\_classification.py:1344: 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.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: