Import Necessary Libraries

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

Load the dataset

In [2]:
```
movie = pd.read_csv('D://CodSoft Task//Data Science Internships//Movie Rat
movie.head()
```

Out[2]:

	movie_id	movie_name	Category
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

In [3]:
```
rating = pd.read_csv("D://CodSoft Task//Data Science Internships//Movie Ra
rating.head()
```

Out[3]:

	user_id	movie_id	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

In [4]:
```
user_data = pd.read_csv("D://CodSoft Task//Data Science Internships//Movie
user_data.head()
```

Out[4]:

	user_id	gender	age	Occupation	zipcode
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

Join the dataset as one dataset

In [5]:
```
data = pd.merge(movie,rating,on = "movie_id")
data
```

Out[5]:

	movie_id	movie_name	Category	user_id	rating	timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474
...	...	...	...	...	...	...
1000204	3952	Contender, The (2000)	Drama|Thriller	5812	4	992072099
1000205	3952	Contender, The (2000)	Drama|Thriller	5831	3	986223125
1000206	3952	Contender, The (2000)	Drama|Thriller	5837	4	1011902656
1000207	3952	Contender, The (2000)	Drama|Thriller	5927	1	979852537
1000208	3952	Contender, The (2000)	Drama|Thriller	5998	4	1001781044

1000209 rows × 6 columns

In [6]:
```
data = pd.merge(data,user_data, on ="user_id")
data.head()
```

Out[6]:

	movie_id	movie_name	Category	user_id	rating	timestamp	gende
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F

Checking the statistics of the dataset

In [7]:
```
1 data.describe()
```

Out[7]:

	movie_id	user_id	rating	timestamp	age	Occupation
count	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06
mean	1.865540e+03	3.024512e+03	3.581564e+00	9.722437e+08	2.973831e+01	8.036138e+00
std	1.096041e+03	1.728413e+03	1.117102e+00	1.215256e+07	1.175198e+01	6.531336e+00
min	1.000000e+00	1.000000e+00	1.000000e+00	9.567039e+08	1.000000e+00	0.000000e+00
25%	1.030000e+03	1.506000e+03	3.000000e+00	9.653026e+08	2.500000e+01	2.000000e+00
50%	1.835000e+03	3.070000e+03	4.000000e+00	9.730180e+08	2.500000e+01	7.000000e+00
75%	2.770000e+03	4.476000e+03	4.000000e+00	9.752209e+08	3.500000e+01	1.400000e+01
max	3.952000e+03	6.040000e+03	5.000000e+00	1.046455e+09	5.600000e+01	2.000000e+01

Checking the description of the dataset

In [8]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   movie_id    1000209 non-null  int64 
 1   movie_name  1000209 non-null  object
 2   Category    1000209 non-null  object
 3   user_id     1000209 non-null  int64 
 4   rating      1000209 non-null  int64 
 5   timestamp   1000209 non-null  int64 
 6   gender      1000209 non-null  object
 7   age         1000209 non-null  int64 
 8   Occupation  1000209 non-null  int64 
 9   zipcode     1000209 non-null  object
dtypes: int64(6), object(4)
memory usage: 83.9+ MB
```

Checking the presence of Null values

In [9]:
```
1 data.isnull().sum()
```

Out[9]:
```
movie_id      0
movie_name    0
Category      0
user_id       0
rating        0
timestamp     0
gender        0
age           0
Occupation    0
zipcode       0
dtype: int64
```

As we can there is no null value so there is no need to deal with missing values.

Checking the Presence of Duplicate Values

In [10]:
```
1 data.duplicated().value_counts()
```

Out[10]:
```
False    1000209
dtype: int64
```

There is no duplicate value

In [11]:
```
1 data.head()
```

Out[11]:

	movie_id	movie_name	Category	user_id	rating	timestamp	gende
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F

Determining the number of rating to each movie

In [12]:
```
1 rat = pd.DataFrame(data["rating"].groupby(data["movie_name"]).count())
```

In [13]:
```
rat.rename(columns = {"rating":"No. of Rating"},inplace = True)
rat.head()
```

Out[13]:

movie_name	No. of Rating
$1,000,000 Duck (1971)	37
'Night Mother (1986)	70
'Til There Was You (1997)	52
'burbs, The (1989)	303
...And Justice for All (1979)	199

In [14]:
```
rat["rating"] = pd.DataFrame(data["rating"].groupby(data["movie_name"]).me
rat.head(1)
```

Out[14]:

movie_name	No. of Rating	rating
$1,000,000 Duck (1971)	37	3.027027

In [15]:
```
rat = rat.reindex(columns = ["rating","No. of Rating"])
rat.head(1)
```

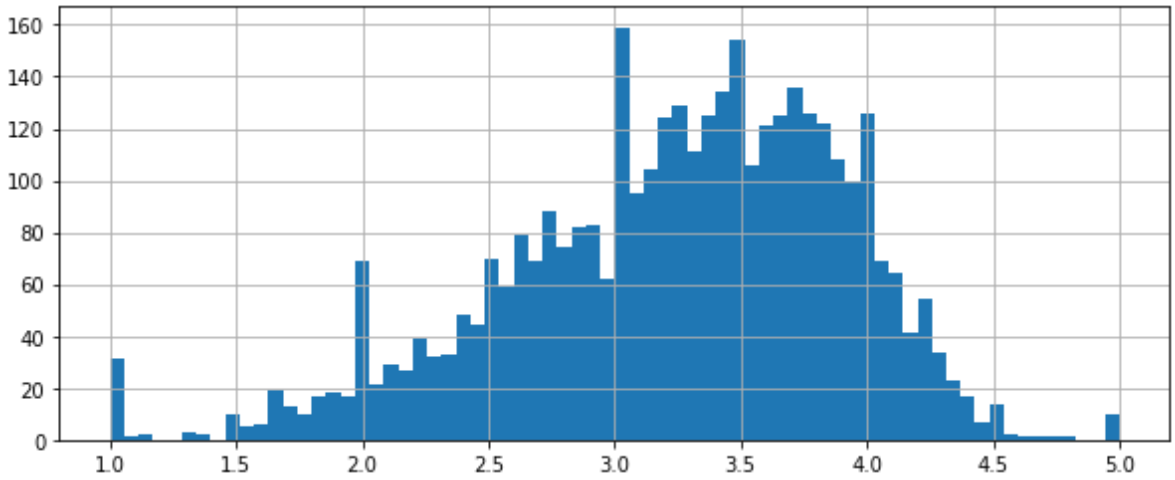
Out[15]:

movie_name	rating	No. of Rating
$1,000,000 Duck (1971)	3.027027	37

plot graph of 'ratings' column

In [16]:
```
1 plt.figure(figsize =(10, 4))
2
3 rat['rating'].hist(bins = 70)
```

Out[16]: <AxesSubplot:>

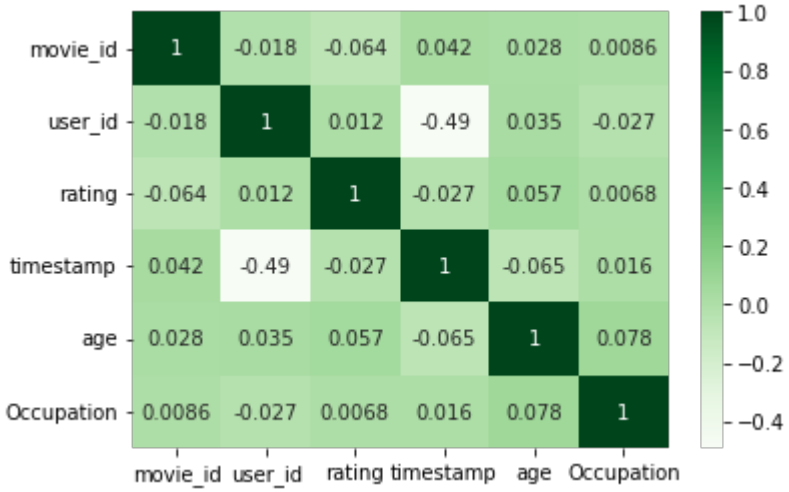


From this graph we can see that most of the people give rating 3.0 to the movies.

Check the correlation of the dataset

In [17]:
```
1 corr = data.corr()
2 sns.heatmap(corr,annot = True,cmap = "Greens")
```

Out[17]: <AxesSubplot:>



Seperate the independent and dependent dataset

In [18]:
```
x = data[["movie_id","user_id","timestamp","Occupation","age"]]
y = data["rating"]
```

Split the Dataset into training and testing set

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 =0.3,random
```

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
reg_lr =LinearRegression()
reg_lr.fit(x_train,y_train)
y_pred_train = reg_lr.predict(x_train)
y_pred_test = reg_lr.predict(x_test)
print("Training Accuracy = ",r2_score(y_train,y_pred_train))
print("Training Mean Square Error = ",mean_squared_error(y_train,y_pred_tr
print("Testing Accuracy = ",r2_score(y_test,y_pred_test))
print("Testing Mean Square Error = ",mean_squared_error(y_test,y_pred_test
```

```
Training Accuracy =  0.007832766100009492
Training Mean Square Error =  1.23770297097172
Testing Accuracy =  0.008266672259429564
Testing Mean Square Error =  1.238619984994157
```

In [ ]:
```

```