In [34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [35]:
```
df_user=pd.read_csv('users.dat',sep="::",names=['UserID','Gender','Age','Occupatior
#'engine' is used to make file compatible with python
```

In [36]:
```
df_user
```

Out[36]:

	UserID	Gender	Age	Occupation	Zip code
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
...	...	...	...	...	...
6035	6036	F	25	15	32603
6036	6037	F	45	1	76006
6037	6038	F	56	1	14706
6038	6039	F	45	0	01060
6039	6040	M	25	6	11106

6040 rows × 5 columns

In [37]:
```
df_movies=pd.read_csv('movies.dat',sep="::",names=['MovieID','Title','Genres'],engi
#encoding is done to convert it to python3 from UTF-8 as the data was not opening i
```

In [38]:
```
df_movies
```

Out[38]:

	MovieID	Title	Genres
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
...	...	...	...
3878	3948	Meet the Parents (2000)	Comedy
3879	3949	Requiem for a Dream (2000)	Drama
3880	3950	Tigerland (2000)	Drama
3881	3951	Two Family House (2000)	Drama
3882	3952	Contender, The (2000)	Drama|Thriller

3883 rows × 3 columns

In [39]:
```
df_ratings=pd.read_csv('ratings.dat',sep="::",names=['UserID','MovieID','Rating','T
```

In [40]:
```
df_ratings
```

Out[40]:

	UserID	MovieID	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
...	...	...	...	...
1000204	6040	1091	1	956716541
1000205	6040	1094	5	956704887
1000206	6040	562	5	956704746
1000207	6040	1096	4	956715648
1000208	6040	1097	4	956715569

1000209 rows × 4 columns

In [41]:
```
df_ratings.shape
```

Out[41]:
```
(1000209, 4)
```

In [42]:
```
df_movies.shape
```

Out[42]:
```
(3883, 3)
```

In [43]:
```
df_user.shape
```

Out[43]:
```
(6040, 5)
```

Create a new dataset [Master_Data] with the following columns

MovieID, Title, UserID, Age, Gender, Occupation, Rating.

(i) Merge two tables at a time.

(ii) Merge the tables using two primary keys MovieID & UserId

In [44]:
```
dfMovieRatings=df_movies.merge(df_ratings,on='MovieID',how='inner')
```

In [45]:
```
dfMovieRatings
```

Out[45]:

	MovieID	Title	Genres	UserID	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 [46]:
```
dfMaster=dfMovieRatings.merge(df_user,on='UserID',how='inner')
```

In [47]:
```
dfMaster
```

Out[47]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gend
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	
...	...	...	...	...	...	...	
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	958489970	
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	958489902	
1000207	3555	U-571 (2000)	Action|Thriller	5727	3	958490699	
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	958490171	

1000209 rows × 10 columns

In [48]:
```
dfMaster.to_csv('Master data.csv')
```

In [49]:
```
dfMaster.isna().sum()
```

Out[49]:
```
MovieID       0
Title         0
Genres        0
UserID        0
Rating        0
Timestamp     0
Gender        0
Age           0
Occupation    0
Zip code      0
dtype: int64
```

Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

1.User Age Distribution

2.User rating of the movie “Toy Story”

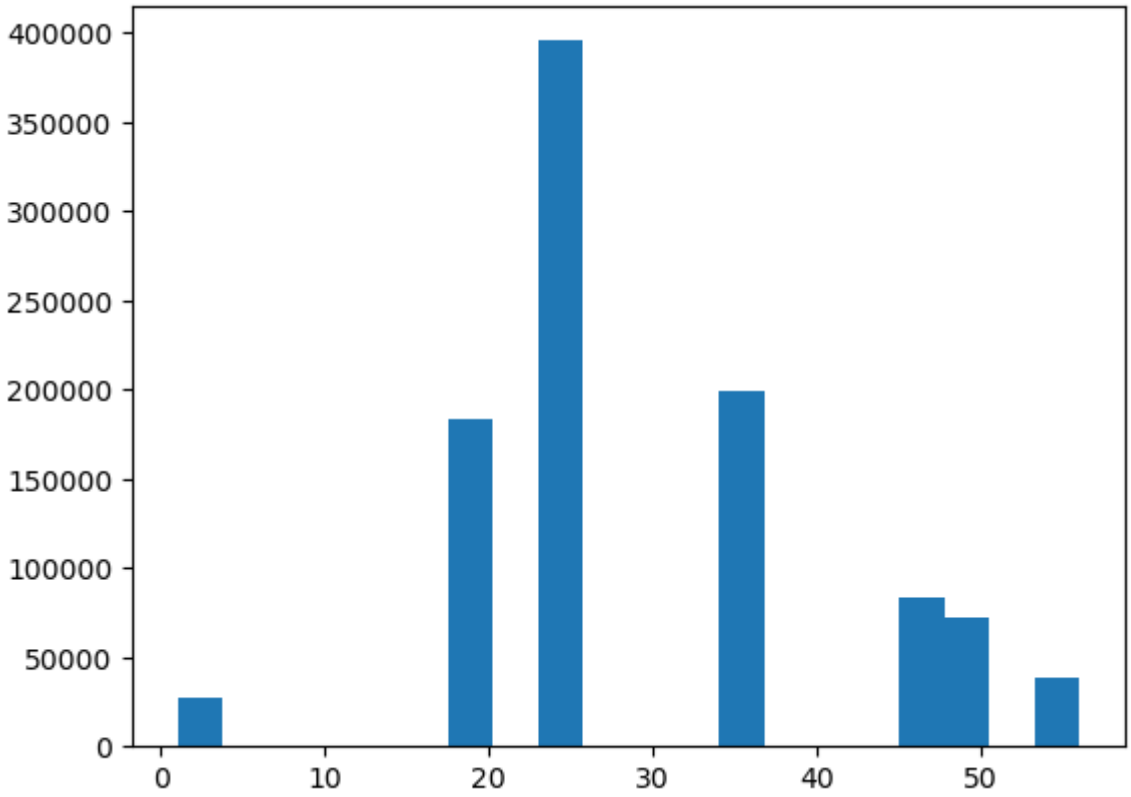
3.Top 25 movies by viewership rating

4.Find the ratings for all the movies reviewed by for a particular user of user id = 2696

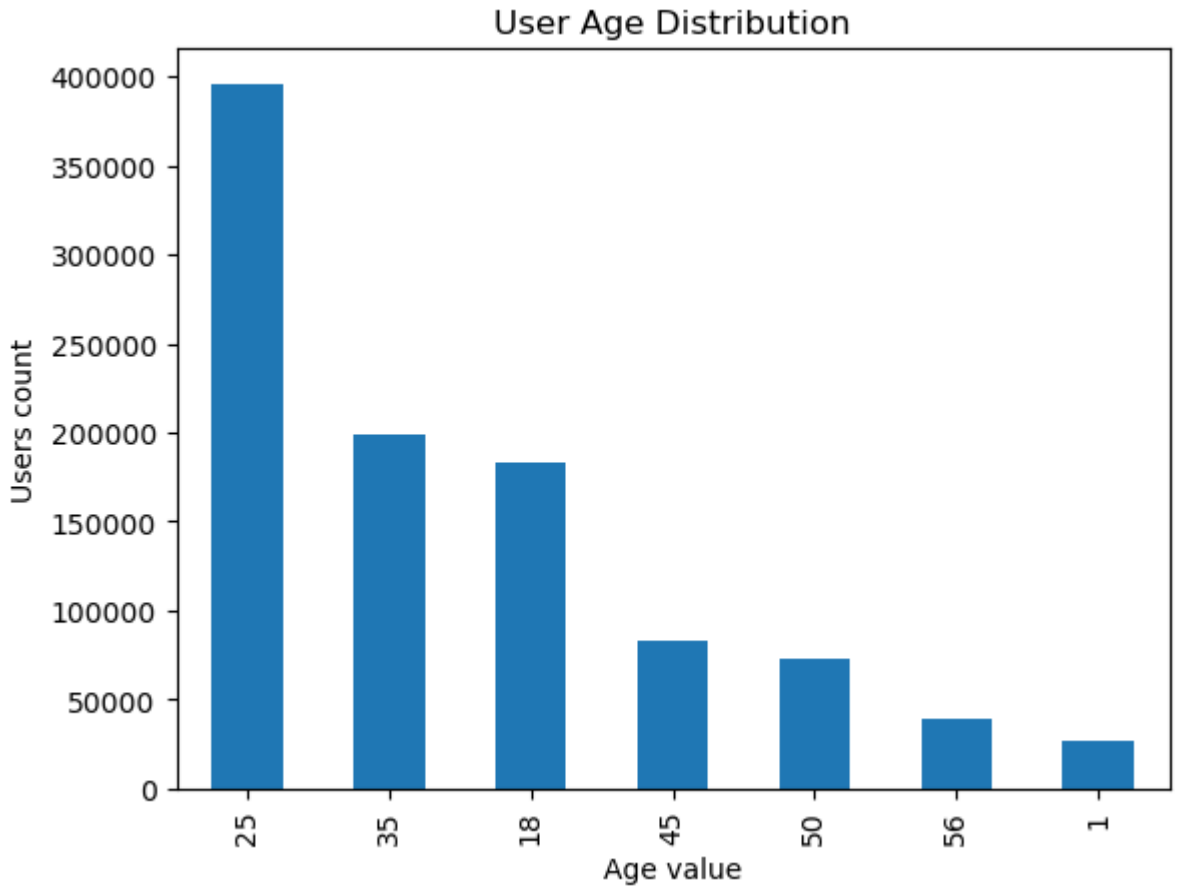
In [50]:
```
dfMaster['Age'].value_counts()
```

Out[50]:
```
25    395556
35    199003
18    183536
45     83633
50     72490
56     38780
1      27211
Name: Age, dtype: int64
```

In [51]:
```
plt.hist(dfMaster['Age'],bins=20)
plt.show()
```



In [57]:
```
dfMaster['Age'].value_counts().plot(kind='bar')
plt.xlabel('Age value')
plt.ylabel('Users count')
plt.title('User Age Distribution')
plt.show()
```



Comment - From the above bar chart we can note that, The age value 25 i.e. the age group between 25-34 years have given the maximun number of ratings

In [63]:
```
# To get user rating of the movie Toy story , Extract toy story movies
toystory=dfMaster[dfMaster['Title'].str.contains('Toy Story')==True]
toystory
```

Out[63]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occu
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	4	978302174	F	1	
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	
...	...	...	...	...	...	...	...	...	
998988	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	3023	4	970471948	F	25	
999027	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5800	5	958015250	M	35	
999486	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	2189	4	974607816	M	1	
999869	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	159	4	989966944	F	45	
1000192	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5727	5	958492554	M	25	

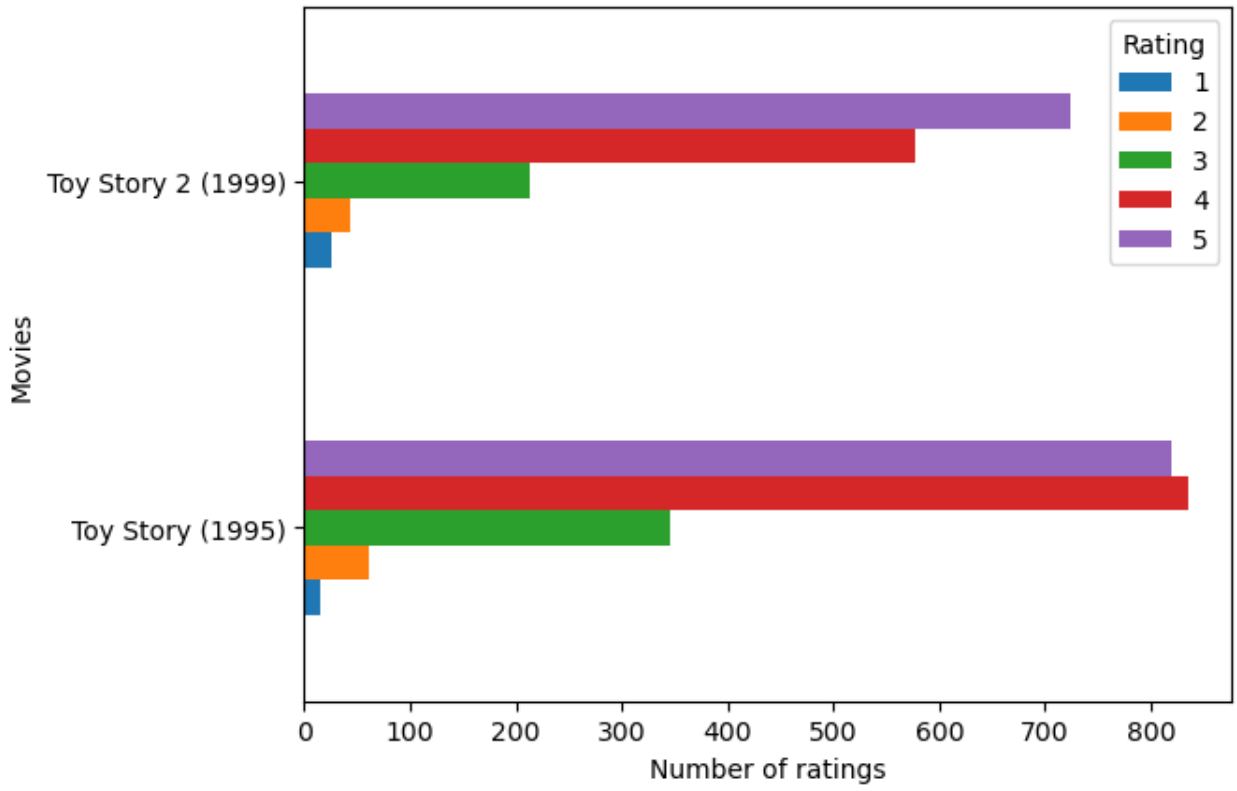
3662 rows × 10 columns

In [64]:
```
toystory.groupby(['Title','Rating']).size()
```

Out[64]:
```
Title                Rating
Toy Story (1995)     1          16
                     2          61
                     3         345
                     4         835
                     5         820
Toy Story 2 (1999)   1          25
                     2          44
                     3         214
                     4         578
                     5         724
dtype: int64
```

In [67]:
```
toystory.groupby(['Title','Rating']).size().unstack().plot(kind='barh',legend=True)
plt.ylabel('Movies')
plt.xlabel('Number of ratings')
```

Out[67]:
```
Text(0.5, 0, 'Number of ratings')
```



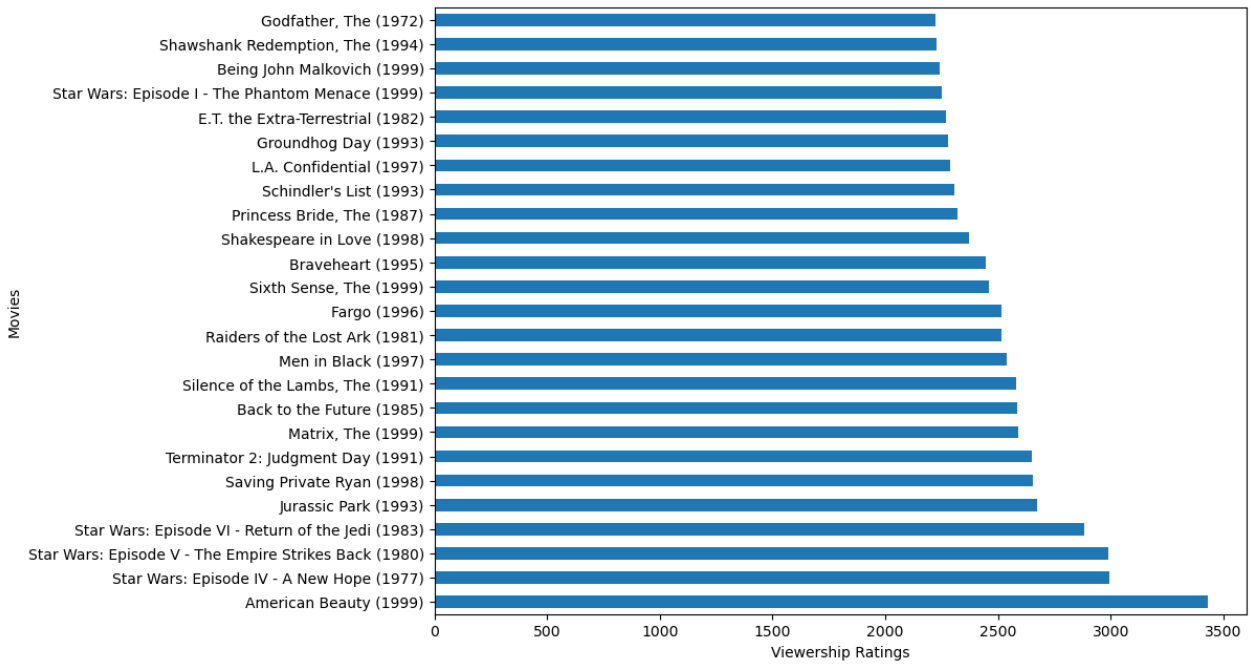
Coment - Movie Toy Story(1995) has got the highest ratings ranging from 4&5 when compared to Toy Story(1999). Thus we can say that, Toy Story(1995) was more popular movie than the second part.

In [78]:
```
# Top 25 movies by viewership rating
dfTop25=dfMaster.groupby('Title').size().sort_values(ascending=False)[:25]
dfTop25
```

Out[78]:
```
Title
American Beauty (1999)                                3428
Star Wars: Episode IV - A New Hope (1977)             2991
Star Wars: Episode V - The Empire Strikes Back (1980) 2990
Star Wars: Episode VI - Return of the Jedi (1983)     2883
Jurassic Park (1993)                                  2672
Saving Private Ryan (1998)                            2653
Terminator 2: Judgment Day (1991)                     2649
Matrix, The (1999)                                    2590
Back to the Future (1985)                             2583
Silence of the Lambs, The (1991)                      2578
Men in Black (1997)                                   2538
Raiders of the Lost Ark (1981)                        2514
Fargo (1996)                                          2513
Sixth Sense, The (1999)                               2459
Braveheart (1995)                                     2443
Shakespeare in Love (1998)                            2369
Princess Bride, The (1987)                            2318
Schindler's List (1993)                               2304
L.A. Confidential (1997)                              2288
Groundhog Day (1993)                                  2278
E.T. the Extra-Terrestrial (1982)                     2269
Star Wars: Episode I - The Phantom Menace (1999)      2250
Being John Malkovich (1999)                           2241
Shawshank Redemption, The (1994)                      2227
Godfather, The (1972)                                 2223
dtype: int64
```

In [82]:
```
plt.figure(figsize=(10,7.5))
dfTop25.plot(kind='barh')
plt.xlabel('Viewership Ratings')
plt.ylabel('Movies')
```

Out[82]:
```
Text(0, 0.5, 'Movies')
```



Comment - By looking at the bar chart, we can see that the movie American Beauty(1999) has been rated by highest number of people amongst all the given movies.

In [85]:
```
#Find the ratings for all the movies reviewed by for a particular user of user id =
df_user2696=dfMaster[dfMaster['UserID']==2696]
df_user2696
```

Out[85]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	A
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	A
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M	
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M	
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M	
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M	

In [87]:
```
df_user2696.shape
```

Out[87]: (20, 10)

Comment - The User with UserID - 2696 has reviewed total 20 movies in which he/she has given 5 star rating to only one movie i.e. Lone star(1996)

Feature Engineering:

Use column genres:

1. Find out all the unique genres (Hint: split the data in column genre making a list and then process the data to find out only the unique categories of genres)

2. Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

3. Determine the features affecting the ratings of any particular movie.

4. Develop an appropriate model to predict the movie ratings

In [92]:
```
#Find out all the unique genres
dfMaster['Genres'].unique()
```

```
'Action|Adventure|Comedy|War', 'Mystery', 'Drama|Western',
'Action|Adventure|Crime|Thriller',
'Action|Mystery|Sci-Fi|Thriller',
"Adventure|Children's|Comedy|Fantasy|Romance",
"Adventure|Children's|Romance",
"Action|Adventure|Animation|Children's|Fantasy",
"Action|Adventure|Children's", "Adventure|Animation|Children's",
'Musical|War', 'Action|Crime|Mystery',
"Adventure|Animation|Children's|Fantasy", 'Comedy|Horror|Thriller',
'Film-Noir', 'Crime|Film-Noir|Mystery', 'Drama|Film-Noir|Thriller',
'Drama|Film-Noir', 'Action|Adventure|War', 'Crime|Drama|Romance',
'Documentary|War', 'Sci-Fi|Thriller|War', 'Action|Comedy|Crime',
'Crime|Horror', 'Drama|Romance|Sci-Fi', 'Crime|Mystery',
'Comedy|Drama|Thriller', 'Crime|Horror|Thriller', 'Horror|Mystery',
'Documentary|Drama', 'Drama|Horror|Thriller',
'Comedy|Horror|Sci-Fi', "Action|Adventure|Children's|Fantasy",
'Animation|Mystery', 'Comedy|Romance|Sci-Fi', 'Romance|Western',
'Drama|Romance|Western', 'Comedy|Film-Noir|Thriller',
'Film-Noir|Horror', 'Fantasy'], dtype=object)
```

In [93]:
```
dfGenres=dfMaster['Genres'].str.split('|')
dfGenres
#str.split- converts string to list
```

Out[93]:
```
0                    [Animation, Children's, Comedy]
1          [Animation, Children's, Musical, Romance]
2                                             [Drama]
3               [Action, Adventure, Fantasy, Sci-Fi]
4                                       [Drama, War]
                              ...
1000204                           [Drama, Thriller]
1000205                  [Comedy, Horror, Thriller]
1000206                           [Comedy, Romance]
1000207                          [Action, Thriller]
1000208                             [Action, Drama]
Name: Genres, Length: 1000209, dtype: object
```

In [94]:
```
listgenres=set() #this is constructor method
for genre in dfGenres:
    listgenres=listgenres.union(set(genre))
```

In [95]:
```
listgenres #List of all the Unique Genres
```

Out[95]:
```
{'Action',
 'Adventure',
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

In [96]:
```
len(listgenres)
```

Out[96]:
```
18
```

In [97]:
```
dfMaster['Genres']
```

Out[97]:
```
0                    Animation|Children's|Comedy
1          Animation|Children's|Musical|Romance
2                                         Drama
3               Action|Adventure|Fantasy|Sci-Fi
4                                     Drama|War
                          ...
1000204                        Drama|Thriller
1000205               Comedy|Horror|Thriller
1000206                       Comedy|Romance
1000207                      Action|Thriller
1000208                         Action|Drama
Name: Genres, Length: 1000209, dtype: object
```

In [98]:
```
#Creating a separate column for each genre category with a one-hot encoding ( 1 and
GenresOnehot=dfMaster['Genres'].str.get_dummies('|')
```

In [99]:
```
GenresOnehot
```

Out[99]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy
0	0	0	1	1	1	0	0	0	0
1	0	0	1	1	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0
3	1	1	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...
1000204	0	0	0	0	0	0	0	1	0
1000205	0	0	0	0	1	0	0	0	0
1000206	0	0	0	0	1	0	0	0	0
1000207	1	0	0	0	0	0	0	0	0
1000208	1	0	0	0	0	0	0	1	0

1000209 rows × 18 columns

In [100]:
```
dfMaster=pd.concat([dfMaster,GenresOnehot],axis=1)
dfMaster
```

Out[100]:

g	Timestamp	Gender	Age	Occupation	Zip code	...	Fantasy	Film-Noir	Horror	Musical	Mystery	Romanc
5	978824268	F	1	10	48067	...	0	0	0	0	0	
5	978824351	F	1	10	48067	...	0	0	0	1	0	
5	978301777	F	1	10	48067	...	0	0	0	0	0	
4	978300760	F	1	10	48067	...	1	0	0	0	0	
5	978824195	F	1	10	48067	...	0	0	0	0	0	
.	...	...	...	...	...	...	...	...	...	...	...	
4	958489970	M	25	4	92843	...	0	0	0	0	0	
2	958489970	M	25	4	92843	...	0	0	1	0	0	
5	958489902	M	25	4	92843	...	0	0	0	0	0	
3	958490699	M	25	4	92843	...	0	0	0	0	0	
5	958490171	M	25	4	92843	...	0	0	0	0	0	

In [101]:
```
dfMaster.dtypes
```

Out[101]:
```
MovieID         int64
Title          object
Genres         object
UserID          int64
Rating          int64
Timestamp       int64
Gender         object
Age             int64
Occupation      int64
Zip code       object
Action          int64
Adventure       int64
Animation       int64
Children's      int64
Comedy          int64
Crime           int64
Documentary     int64
Drama           int64
Fantasy         int64
Film-Noir       int64
Horror          int64
Musical         int64
Mystery         int64
Romance         int64
Sci-Fi          int64
Thriller        int64
War             int64
Western         int64
dtype: object
```

In [102]:
```
dfMaster['Gender']=dfMaster['Gender'].replace('M','0')
dfMaster['Gender']=dfMaster['Gender'].replace('F','1')
```

In [103]:
```
dfMaster['Gender']
```

Out[103]:
```
0          1
1          1
2          1
3          1
4          1
          ..
1000204    0
1000205    0
1000206    0
1000207    0
1000208    0
Name: Gender, Length: 1000209, dtype: object
```

In [104]:
```
dfMaster['Gender']=dfMaster['Gender'].astype('int') #to convert object into integer
dfMaster['Gender'].dtype
```

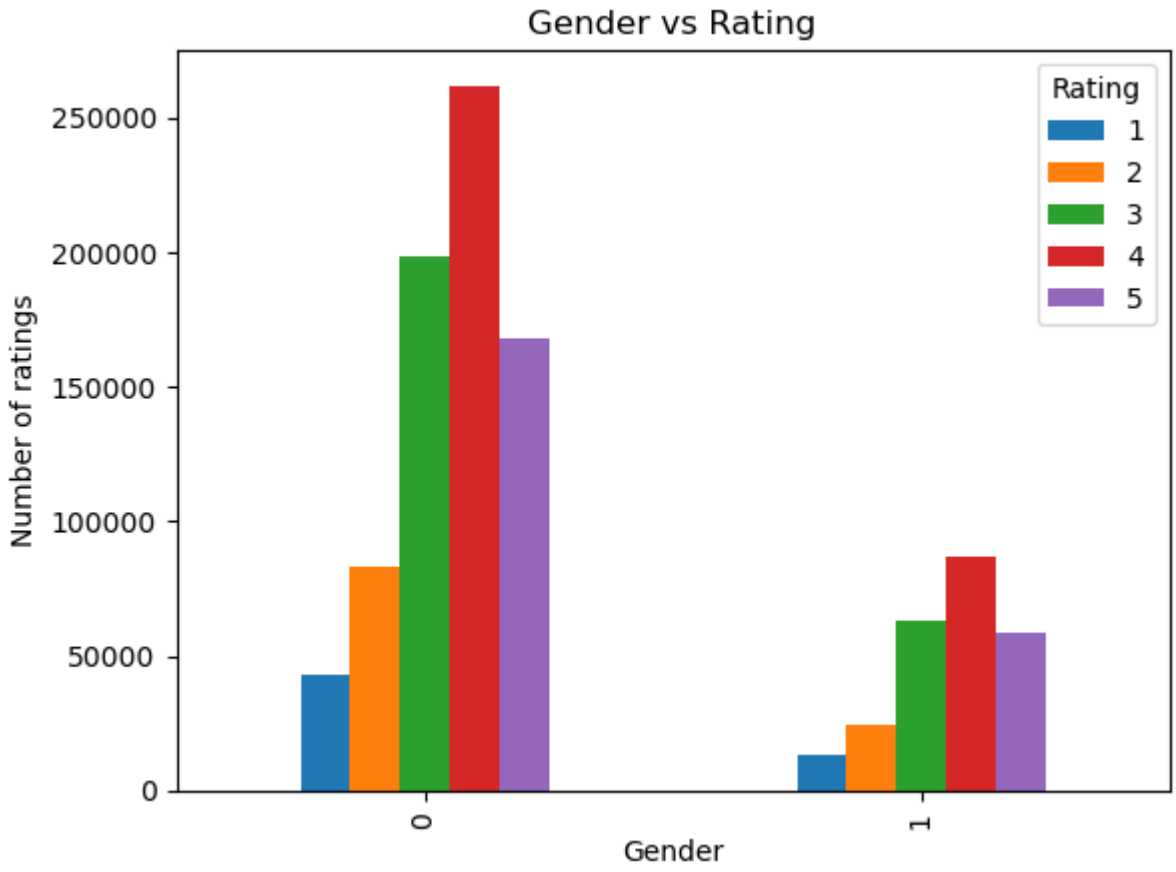
Out[104]:
```
dtype('int32')
```

In [108]:
```
# Gender vs rating
GenderAffecting=dfMaster.groupby('Gender').size()
GenderAffecting
```

Out[108]:
```
Gender
0    753769
1    246440
dtype: int64
```

In [112]:
```
dfMaster.groupby(['Gender','Rating']).size().unstack().plot(kind='bar',legend=True)
plt.ylabel('Number of ratings')
plt.title('Gender vs Rating')
```

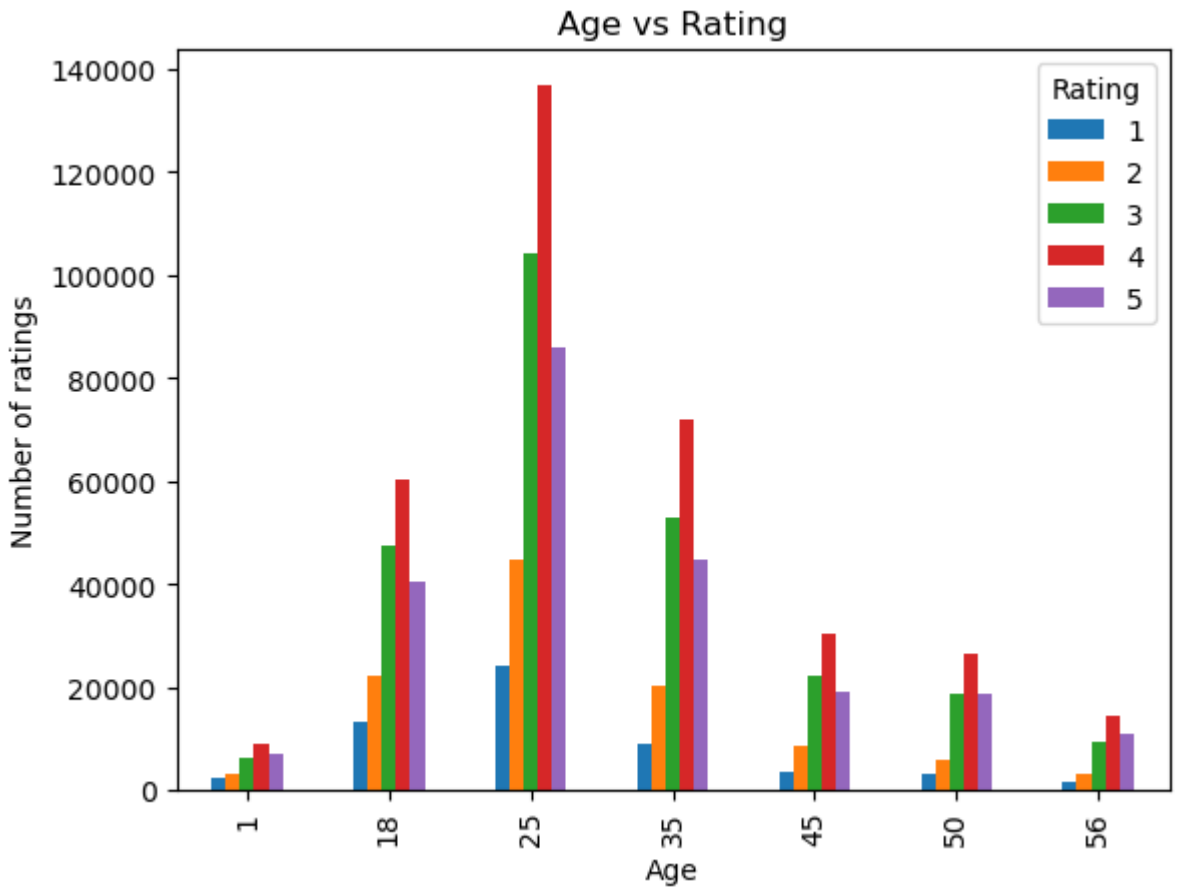
Out[112]:
```
Text(0.5, 1.0, 'Gender vs Rating')
```



Comment - It is evident that, Males have shown more interest in rating the movies compared to females whose number of ratings stand mere infront of their counterparts.

In [115]:
```
dfMaster.groupby(['Age','Rating']).size().unstack().plot(kind='bar',legend=True)
plt.ylabel('Number of ratings')
plt.title('Age vs Rating')
```

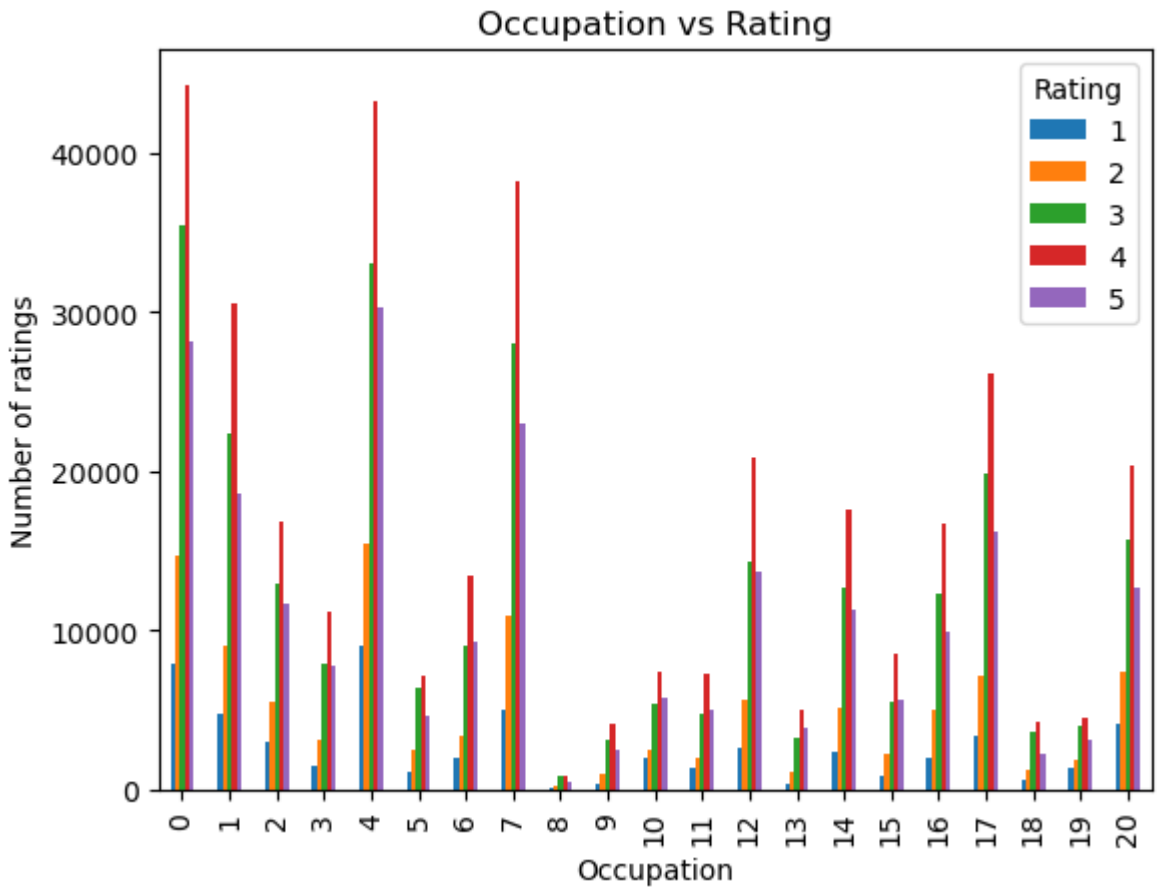
Out[115]: Text(0.5, 1.0, 'Age vs Rating')



Comment - People of the age group between 25-34 years have been more prone to rating the movies than people of other age groups.

In [116]:
```
dfMaster.groupby(['Occupation','Rating']).size().unstack().plot(kind='bar',legend=T
plt.ylabel('Number of ratings')
plt.title('Occupation vs Rating')
```

Out[116]:
```
Text(0.5, 1.0, 'Occupation vs Rating')
```



Comment - People who are in managerial positions,students and other undescribed professionals have participated in the reviewing of the movies more than people of other professions.

Developing an appropriate model to predict the movie ratings

In [117]:
```
# first 500 records
new_data=dfMaster[:500]
features=new_data[['MovieID','Age','Occupation','Gender']].values
features
```

Out[117]:
```
array([[   1,    1,   10,    1],
       [  48,    1,   10,    1],
       [ 150,    1,   10,    1],
       ...,
       [1200,   35,    1,    1],
       [1201,   35,    1,    1],
       [1203,   35,    1,    1]], dtype=int64)
```

In [120]:
```
label=new_data[['Rating']].values
```

In [122]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(features,label,test_size=0.20,random
```

In [123]:
```
X_train.shape
```

Out[123]:
```
(400, 4)
```

In [124]:
```
X_test.shape
```

Out[124]:
```
(100, 4)
```

In [126]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(features,label,test_size=0.20,random
```

In [127]:
```
from sklearn.linear_model import LinearRegression # y=bo+b1X1+b2X2..
lr=LinearRegression()
```

In [128]:
```
lr.fit(X_train,y_train)
lr.predict(X_test)
```

Out[128]:
```
array([[3.46596348],
       [3.96471588],
       [4.31047093],
       [3.99788954],
       [3.68849118],
       [4.27254115],
       [4.30172875],
       [4.05316268],
       [4.02252707],
       [4.18751639],
       [4.2936504 ],
       [3.81923886],
       [4.26506799],
       [3.90352061],
       [4.31625205],
       [4.28057831],
       [4.14845859],
       [4.13671417],
       [3.75346841],
       [4.24955767],
       [4.15804678],
       [4.15720077],
       [3.90518754],
       [3.49627911],
       [3.96076779],
       [4.18032524],
       [4.14803558],
       [3.7680548 ],
       [3.75335251],
       [3.65335635],
       [4.17130106],
       [3.64912626],
       [4.19047745],
       [3.92527304],
       [4.35822974],
       [3.65561239],
       [4.24391756],
       [3.97811115],
       [3.78314211],
       [4.05270169],
       [4.28791046],
       [3.82230294],
       [4.16199486],
       [3.84659342],
       [4.10287347],
       [3.95963977],
       [4.37923917],
       [4.27785808],
       [4.23954647],
       [4.2111637 ],
       [3.97698313],
       [3.86196274],
       [4.04822758],
       [4.24293053],
       [3.82022588],
       [4.10191171],
       [4.01354086],
```

```
[4.23442918],
[3.80866364],
[3.95329464],
[4.14859959],
[3.55584542],
[3.73415101],
[4.16340489],
[3.9514616 ],
[4.04943336],
[4.18709338],
[3.98499503],
[3.6421392 ],
[4.27747625],
[3.51785255],
[3.58736699],
[3.91646778],
[3.86711676],
[3.74881532],
[3.80296043],
[3.95566659],
[3.98854537],
[4.18497834],
[4.17398011],
[4.00014559],
[3.92396604],
[4.16791699],
[3.94882053],
[3.73965013],
[3.83217314],
[3.87366598],
[4.24067449],
[3.77292682],
[3.91237869],
[4.29185854],
[4.17200607],
[3.45270921],
[3.72625485],
[4.2121919 ],
[3.53308087],
[3.88547224],
[3.96147281],
[4.19625857],
[4.2497985 ]])
```

In [132]:
```
y_pred=lr.predict(X_test)
y_pred
```

Out[132]:
```
array([[3.46596348],
       [3.96471588],
       [4.31047093],
       [3.99788954],
       [3.68849118],
       [4.27254115],
       [4.30172875],
       [4.05316268],
       [4.02252707],
       [4.18751639],
       [4.2936504 ],
       [3.81923886],
       [4.26506799],
       [3.90352061],
       [4.31625205],
       [4.28057831],
       [4.14845859],
       [4.13671417],
       [3.75346841],
       [4.24955767],
       [4.15804678],
       [4.15720077],
       [3.90518754],
       [3.49627911],
       [3.96076779],
       [4.18032524],
       [4.14803558],
       [3.7680548 ],
       [3.75335251],
       [3.65335635],
       [4.17130106],
       [3.64912626],
       [4.19047745],
       [3.92527304],
       [4.35822974],
       [3.65561239],
       [4.24391756],
       [3.97811115],
       [3.78314211],
       [4.05270169],
       [4.28791046],
       [3.82230294],
       [4.16199486],
       [3.84659342],
       [4.10287347],
       [3.95963977],
       [4.37923917],
       [4.27785808],
       [4.23954647],
       [4.2111637 ],
       [3.97698313],
       [3.86196274],
       [4.04822758],
       [4.24293053],
       [3.82022588],
       [4.10191171],
       [4.01354086],
```

```
       [4.23442918],
       [3.80866364],
       [3.95329464],
       [4.14859959],
       [3.55584542],
       [3.73415101],
       [4.16340489],
       [3.9514616 ],
       [4.04943336],
       [4.18709338],
       [3.98499503],
       [3.6421392 ],
       [4.27747625],
       [3.51785255],
       [3.58736699],
       [3.91646778],
       [3.86711676],
       [3.74881532],
       [3.80296043],
       [3.95566659],
       [3.98854537],
       [4.18497834],
       [4.17398011],
       [4.00014559],
       [3.92396604],
       [4.16791699],
       [3.94882053],
       [3.73965013],
       [3.83217314],
       [3.87366598],
       [4.24067449],
       [3.77292682],
       [3.91237869],
       [4.29185854],
       [4.17200607],
       [3.45270921],
       [3.72625485],
       [4.2121919 ],
       [3.53308087],
       [3.88547224],
       [3.96147281],
       [4.19625857],
       [4.2497985 ]])
```

In [130]:
```
# error
from sklearn.metrics import mean_squared_error
print('Mean Squared Error',mean_squared_error(y_test,y_pred))
```

```
Mean Squared Error 0.6489142338657047
```

Finding the co-efficient of determination

In [136]:
```
from sklearn.metrics import r2_score
print('R2 score',r2_score(y_test,y_pred))
```

```
R2 score -0.07240825295935327
```