In [40]:

```
import pandas as pd
import numpy as np
import seaborn as sn
import warnings
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [41]:

```
warnings.filterwarnings(action='ignore')
```

In [42]:

```
# Q1: Import the three datasets
movies_colnames = ['MovieID','Title','Genere']
movies_df = pd.read_csv('C:/Users/Gautham/movies.dat ',sep='::',names=movies_colnames)
#movies_df.head()
```

In [43]:

```
rating_colnames =['UserID','MovieID','Rating','Timestamp']
rating_df = pd.read_csv('C:/Users/Gautham/ratings.dat',sep='::',names=rating_colnames)
#rating_df.head()
```

In [44]:

```
#UserID::Gender::Age::Occupation::Zip-code
user_colnames = ['UserID','Gender','Age','Occupation','Zip-code']
user_df = pd.read_csv('C:/Users/Gautham/users.dat',sep='::',names=user_colnames)
#user_df.head()
```

In [45]:

```
#MovieID Title UserID Age Gender Occupation Rating.
```

In [46]:

```
merge1 = pd.merge(movies_df,rating_df,on='MovieID')
merge1.head()
Master_data = pd.merge(merge1,user_df,on='UserID')
Master_data.head()
#Master_data.isna().any()
```

Out[46]:

	MovieID	Title	Genere	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

In [47]:

```
Master_data.drop(['Timestamp'],axis=1,inplace=True)
```

In [48]:

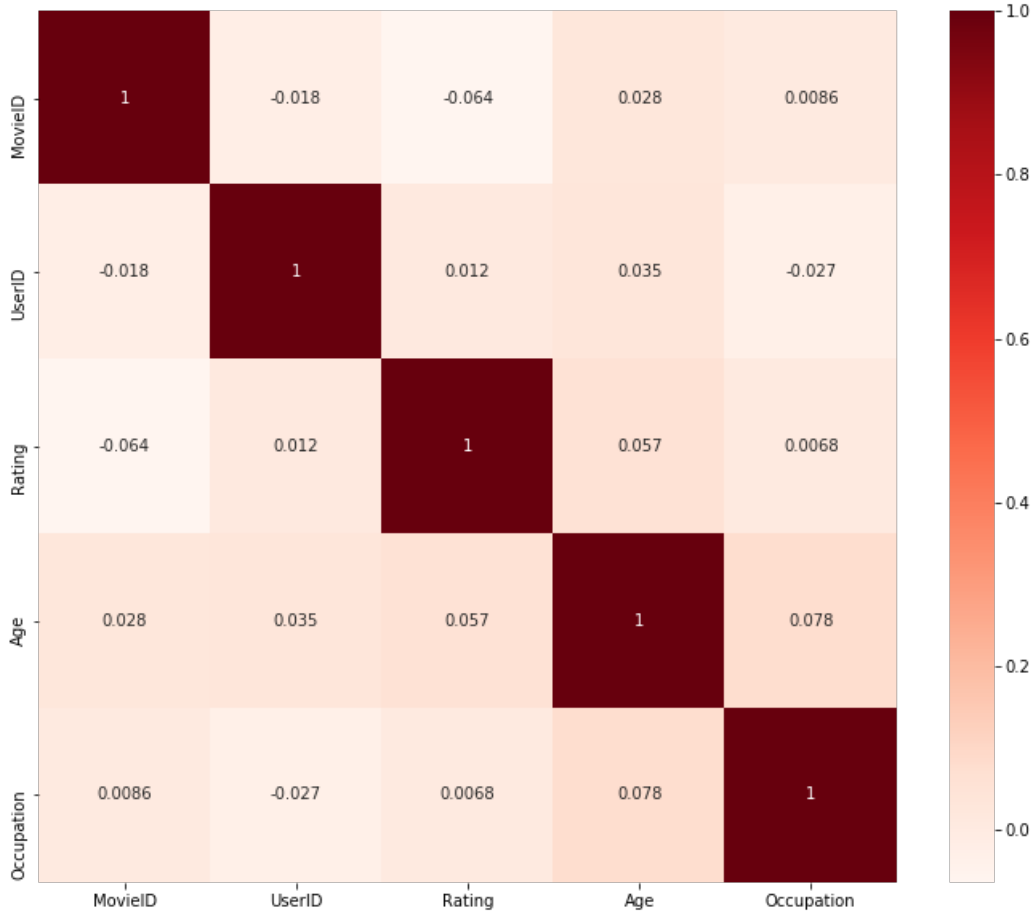
```
Master_data.kurtosis()
```

Out[48]:

```
MovieID      -1.111021
UserID       -1.200995
Rating       -0.351971
Age           0.019044
Occupation   -1.217006
dtype: float64
```

In [49]:

```
corr = Master_data.corr()
plt.figure(figsize=(12,10))
sn.heatmap(corr, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [50]:

```
Master_data.skew()
```

Out[50]:

```
MovieID       0.092436
UserID        0.005735
Rating       -0.553610
Age           0.398471
Occupation    0.404363
dtype: float64
```

In [51]:

```
# kurtosis and skewness are fine. Above graph is normally distributed.
#25-30 Age group is the age group who  provides more rating for movies
```

In [52]:

```
Master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 9 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   MovieID     1000209 non-null  int64
 1   Title       1000209 non-null  object
 2   Genere      1000209 non-null  object
 3   UserID      1000209 non-null  int64
 4   Rating      1000209 non-null  int64
 5   Gender      1000209 non-null  object
 6   Age         1000209 non-null  int64
 7   Occupation  1000209 non-null  int64
 8   Zip-code    1000209 non-null  object
dtypes: int64(5), object(4)
memory usage: 76.3+ MB
```

In [53]:

```
#User rating of the movie "Toy Story"
Master_data['Rating'][Master_data['Title']=="Toy Story (1995)"].value_counts()
```

Out[53]:

```
4    835
5    820
3    345
2     61
1     16
Name: Rating, dtype: int64
```

In [54]:

```
#Top 25 movies by viewership rating
TopMovies = Master_data['Title'][Master_data['Rating']==5].value_counts()
TopMovies.head(25)
```

Out[54]:

```
American Beauty (1999)                                1963
Star Wars: Episode IV - A New Hope (1977)             1826
Raiders of the Lost Ark (1981)                        1500
Star Wars: Episode V - The Empire Strikes Back (1980) 1483
Schindler's List (1993)                               1475
Godfather, The (1972)                                 1475
Shawshank Redemption, The (1994)                      1457
Matrix, The (1999)                                    1430
Saving Private Ryan (1998)                            1405
Sixth Sense, The (1999)                               1385
Silence of the Lambs, The (1991)                      1350
Fargo (1996)                                          1278
Braveheart (1995)                                     1206
Pulp Fiction (1994)                                   1193
Princess Bride, The (1987)                            1186
Usual Suspects, The (1995)                            1144
Star Wars: Episode VI - Return of the Jedi (1983)     1028
L.A. Confidential (1997)                              1009
Being John Malkovich (1999)                           1007
Shakespeare in Love (1998)                             987
Casablanca (1942)                                      984
Forrest Gump (1994)                                    945
Terminator 2: Judgment Day (1991)                      942
Godfather: Part II, The (1974)                         941
One Flew Over the Cuckoo's Nest (1975)                 937
Name: Title, dtype: int64
```

In [55]:

```
#Find the ratings for all the movies reviewed by for a particular user of user id = 2696
Master_data['Rating'][Master_data['UserID']==2696].value_counts()
```

Out[55]:

```
4    11
3     3
2     3
1     2
5     1
Name: Rating, dtype: int64
```

In [18]:

```
#Uisng filters
filter = Master_data['UserID']==2696
User_2696 = Master_data[filter]
User_2696[['Title','Rating']]
```

Out[18]:

	Title	Rating
991035	Client, The (1994)	3
991036	Lone Star (1996)	5
991037	Basic Instinct (1992)	4
991038	E.T. the Extra-Terrestrial (1982)	3
991039	Shining, The (1980)	4
991040	Back to the Future (1985)	2
991041	Cop Land (1997)	3
991042	L.A. Confidential (1997)	4
991043	Game, The (1997)	4
991044	I Know What You Did Last Summer (1997)	2
991045	Devil's Advocate, The (1997)	4
991046	Midnight in the Garden of Good and Evil (1997)	4
991047	Palmetto (1998)	4
991048	Wild Things (1998)	4
991049	Perfect Murder, A (1998)	4
991050	I Still Know What You Did Last Summer (1998)	2
991051	Psycho (1998)	4
991052	Lake Placid (1999)	1
991053	Talented Mr. Ripley, The (1999)	4
991054	JFK (1991)	1

In [56]:

```
geners_split = Master_data['Genere'].str.split("|")
```

In [57]:

```
#Find out all the unique genres (Hint: split the data in column genre making a list and then proce
ss the data to find out only the unique categories of genres)
listGenres = set()
for genre in geners_split:
    listGenres = listGenres.union(set(genre))
listGenres
```

Out[57]:

```
{'Action',
 'Adventure',
```

```
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

In [58]:

```
generes_encoding = Master_data['Genere'].str.get_dummies("|")
```

In [59]:

```
generes_encoding.head()
```

Out[59]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romanc
0	0	0	1	1	1	0	0	0	0	0	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	1	0	
2	0	0	0	0	0	0	0	1	0	0	0	0	0	
3	1	1	0	0	0	0	0	0	1	0	0	0	0	
4	0	0	0	0	0	0	0	1	0	0	0	0	0	

In [61]:

```
Master_data =  pd.concat([Master_data,generes_encoding],axis=1)
Master_data.head()
```

Out[61]:

	MovieID	Title	Genere	UserID	Rating	Gender	Age	Occupation	Zip-code	Action	...	Fantasy	Fil N
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	F	1	10	48067	0	...	0	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	F	1	10	48067	0	...	0	
2	150	Apollo 13 (1995)	Drama	1	5	F	1	10	48067	0	...	0	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	F	1	10	48067	1	...	1	
4	527	Schindler's List (1993)	Drama|War	1	5	F	1	10	48067	0	...	0	

5 rows × 27 columns

In [62]:

```
Master_data.columns
```

Out[62]:

```
Index(['MovieID', 'Title', 'Genere', 'UserID', 'Rating', 'Gender', 'Age',
       'Occupation', 'Zip-code', 'Action', 'Adventure', 'Animation',
       'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
       'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
       'Thriller', 'War', 'Western'],
      dtype='object')
```

In [65]:

```
Master_data.drop(['Genere'],inplace=True,axis=1)
```

In [66]:

```
Master_data.head()
```

Out[66]:

	MovieID	Title	UserID	Rating	Gender	Age	Occupation	Zip-code	Action	Adventure	...	Fantasy	Film-Noir	Horror	Musical	Mys
0	1	Toy Story (1995)	1	5	F	1	10	48067	0	0	...	0	0	0	0	
1	48	Pocahontas (1995)	1	5	F	1	10	48067	0	0	...	0	0	0	1	
2	150	Apollo 13 (1995)	1	5	F	1	10	48067	0	0	...	0	0	0	0	
3	260	Star Wars: Episode IV - A New Hope (1977)	1	4	F	1	10	48067	1	1	...	1	0	0	0	
4	527	Schindler's List (1993)	1	5	F	1	10	48067	0	0	...	0	0	0	0	

5 rows × 26 columns

In [70]:

```
Master_data['Gender'] =Master_data['Gender'].str.replace('F','0')
Master_data['Gender'] =Master_data['Gender'].str.replace('M','1')
Master_data['Gender'] = Master_data['Gender'].astype(int)
```

In [72]:

```
Master_data['Gender'].value_counts()
```

Out[72]:

```
1    753769
0    246440
Name: Gender, dtype: int64
```

In [74]:

```
mean_age = round(geners_encoded['Age'].mean())
Master_data['Age'] = Master_data['Age'].replace([1],mean_age)
```

In [66]:

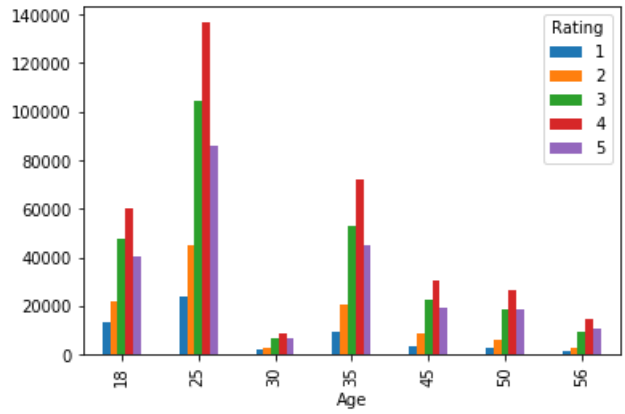
```
#geners_encoded = geners_encoded.drop(['Genere'],axis = 1)
#geners_encoded = geners_encoded.drop(['Zip-code'],axis= 1)
```

In [76]:

```
#Features affecting the ratings of any particular movie.
Master_data.groupby(["Age","Rating"]).size().unstack().plot(kind='bar',legend=True)
```

Out[76]:

<matplotlib.axes._subplots.AxesSubplot at 0x25fa22e2388>

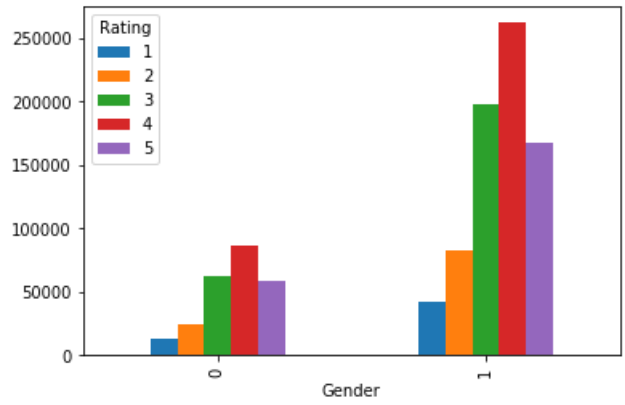


In [79]:

```
Master_data.groupby(["Gender","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
```

Out[79]:

<matplotlib.axes._subplots.AxesSubplot at 0x25ff8068388>



In [82]:

```
first_500 = Master_data[:1000]
```

In [84]:

```
features = first_500[['MovieID','Age','Occupation']].values
labels = first_500[['Rating']].values
```

In [85]:

```
from sklearn.model_selection import train_test_split
```

In [86]:

```
train, test, train_labels, test_labels = train_test_split(features,labels,test_size=0.33,random_state=42)
```

In [88]:

```
from sklearn.linear_model import LogisticRegression #Classification for multiclass
logreg = LogisticRegression()
logreg.fit(train, train_labels)
Y_pred = logreg.predict(test)
acc_log = round(logreg.score(train, train_labels) * 100, 2)
```

```
acc_log
```

Out[88]:

```
36.42
```

In [90]:

```
svc = SVC()
svc.fit(train, train_labels)
Y_pred = svc.predict(test)
acc_svc = round(svc.score(train, train_labels) * 100, 2)
acc_svc
```

Out[90]:

```
38.81
```

In [91]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(train, train_labels)
Y_pred = knn.predict(test)
acc_knn = round(knn.score(train, train_labels) * 100, 2)
acc_knn
```

Out[91]:

```
58.51
```

In [95]:

```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(train, train_labels)
Y_pred = decision_tree.predict(test)
acc_decision_tree = round(decision_tree.score(train, train_labels) * 100, 2)
acc_decision_tree
```

Out[95]:

```
100.0
```

In [93]:

```
random_forest = RandomForestClassifier(n_estimators=50)
random_forest.fit(train, train_labels)
Y_pred = random_forest.predict(test)
random_forest.score(train, train_labels)
acc_random_forest = round(random_forest.score(train, train_labels) * 100, 2)
acc_random_forest
```

Out[93]:

```
100.0
```