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

```
rating_header = "UserID::MovieID::Rating::Timestamp".split("::")
users_header = "UserID::Gender::Age::Occupation::Zip-Code".split("::")
movies_header = "MovieID::Title::Genres".split("::")
```

```
movies=pd.read_csv("movies.dat" , sep="::" , names=movies_header)
users=pd.read_csv("users.dat" , sep="::" , names=users_header)
ratings=pd.read_csv("ratings.dat" , sep="::" , names=rating_header , parse_dates=[])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: ParserWarning: Falling back to the 'python' engine because the 'c'
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: ParserWarning: Falling back to the 'python' engine because the 'c'

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: ParserWarning: Falling back to the 'python' engine because the 'c'
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
movies
```

	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

```
users.head()
```

	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

```
ratings.head()
```

	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

```
data_1 = pd.merge(movies,ratings, on='MovieID')
```

```
data_1.head()
```

	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

```
FinalData = pd.merge(data_1,users, on='UserID')
FinalData.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupa
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	

```
FinalData['Title'].unique()
```

```
array(['Toy Story (1995)', 'Pocahontas (1995)', 'Apollo 13 (1995)', ...,
       'Voyage to the Beginning of the World (1997)',
       'Project Moon Base (1953)', "Heaven's Burning (1997)"],
      dtype=object)
```

```
FinalData['Title'].head(20)
```

```
0                                    Toy Story (1995)
1                                  Pocahontas (1995)
2                                   Apollo 13 (1995)
3             Star Wars: Episode IV - A New Hope (1977)
4                              Schindler's List (1993)
5                            Secret Garden, The (1993)
6                                      Aladdin (1992)
7                Snow White and the Seven Dwarfs (1937)
8                           Beauty and the Beast (1991)
9                                        Fargo (1996)
10                     James and the Giant Peach (1996)
11    Wallace & Gromit: The Best of Aardman Animatio...
12                                 Close Shave, A (1995)
13                Hunchback of Notre Dame, The (1996)
14                                 My Fair Lady (1964)
15                             Wizard of Oz, The (1939)
16                                         Gigi (1958)
17                                  Cinderella (1950)
18                                Mary Poppins (1964)
19                                       Dumbo (1941)
Name: Title, dtype: object
```

```
FinalData['Age'].value_counts()
```

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

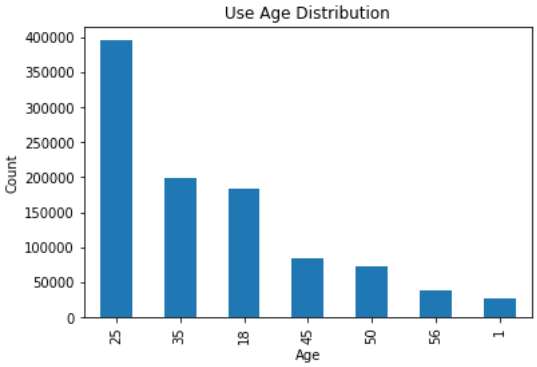
User Age Distribution

```
#plt.bar(FinalData['Age','Title']) #getting key value error
```

```
plt.hist(users['Age'])
```

```
(array([ 222.,    0.,    0., 1103., 2096.,    0., 1193.,    0., 1046.,
         380.]),
 array([ 1. ,  6.5, 12. , 17.5, 23. , 28.5, 34. , 39.5, 45. , 50.5, 56. ]),
 <a list of 10 Patch objects>)
```


```
FinalData['Age'].value_counts().plot(kind='bar')
plt.xlabel("Age")
plt.ylabel('Count')
plt.title("Use Age Distribution")
plt.show()
```



User rating of the movie “Toy Story”

```
FinalData.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupa
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	

```
FinalData[FinalData['Title'].str.contains('Toy Story')== True]
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	4	978302174	F	1	10
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	9
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	12
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	17
...	...	...	...	...	...	...	...	...	...

```
ToyStory_ratings = FinalData[FinalData['Title'].str.contains('Toy Story')== True]
```

```
ToyStory_ratings
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	4	978302174	F	1	10
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	9
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	12
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	17
...	...	...	...	...	...	...	...	...	...

```
ToyStory_ratings.groupby(["Title","Rating"]).size()
```

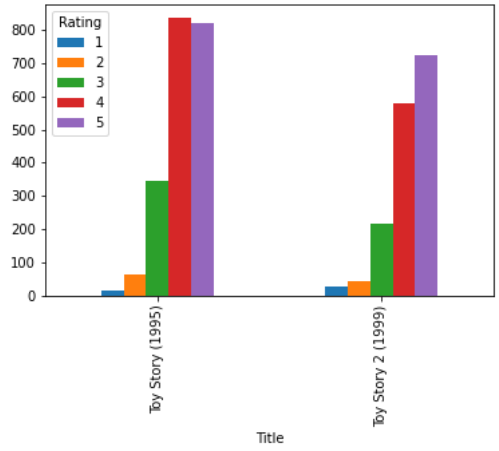
```
Title               Rating
Toy Story (1995)    1          16
                    2          61
                    3         345
                    4         835
                    5         820
Toy Story 2 (1999)  1          25
                    2          44
                    3         214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

```
ToyStory_ratings_plot=ToyStory_ratings.groupby(["Title","Rating"]).size()
ToyStory_ratings_plot
```

```
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
```

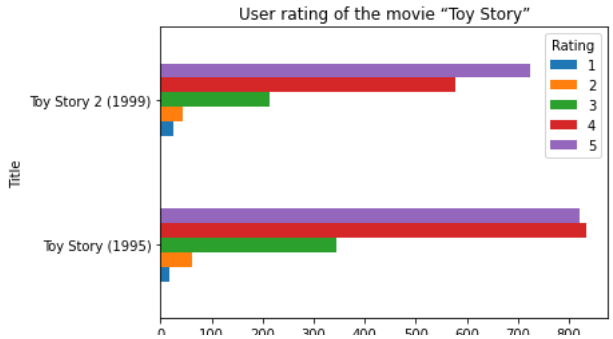
```
ToyStory_ratings_plot.unstack().plot(kind='bar' , legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b09f6e150>
```



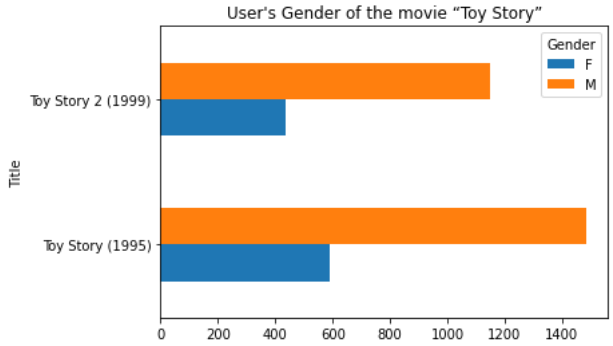

```
ToyStory_ratings_plot.unstack().plot(kind='barh', stacked=False, legend=True)
plt.title("User rating of the movie “Toy Story”")
```

```
Text(0.5, 1.0, 'User rating of the movie “Toy Story”')
```




```
ToyStory_ratings.groupby(["Title","Gender"]).size().unstack().plot(kind='barh', stacked=False, legend=True)
plt.title("User's Gender of the movie “Toy Story”")
```

```
Text(0.5, 1.0, "User's Gender of the movie “Toy Story”")
```



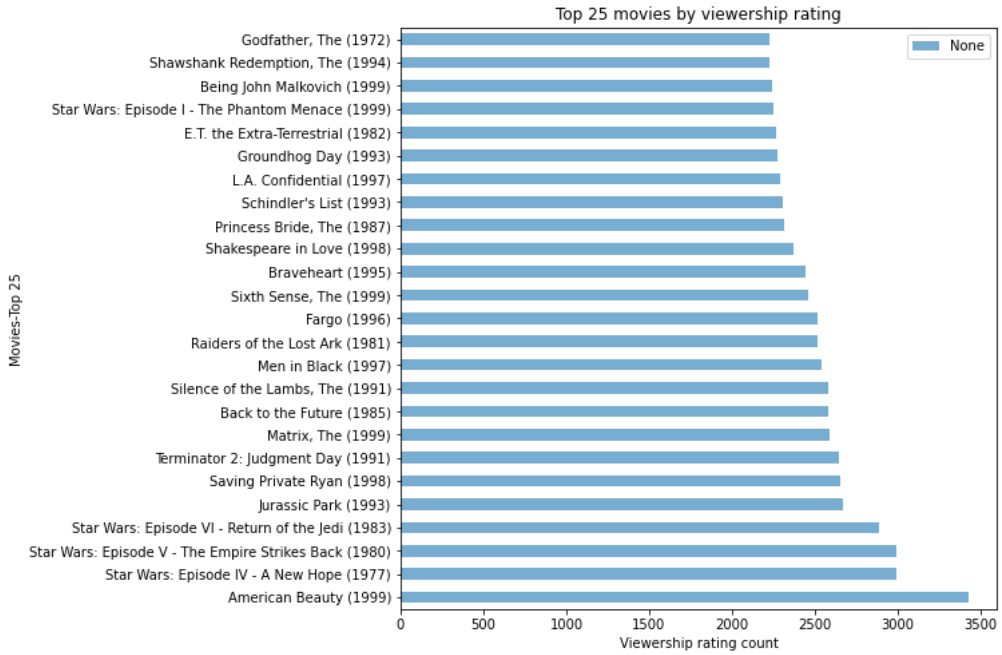
Top 25 movies by viewership rating

```
#FinalData.to_excel("FinalData.xlsx")
#df1=pd.read_excel('FinalData.xlsx')
#df1
```

```
Top25 = FinalData.groupby('Title').size()
Top25.sort_values(ascending=False).head(25)
```

```
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
```

```
Top25.sort_values(ascending=False)[:25].plot(kind='barh', legend=True , alpha = 0.6 , figsize=(8,8))
plt.xlabel("Viewership rating count")
plt.ylabel("Movies-Top 25")
plt.title("Top 25 movies by viewership rating")
plt.show()
```



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

```
FinalData
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1
...	...	...	...	...	...	...	...	...
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	M	25

```
userId = 2696
```

```
FinalData[FinalData["UserID"] == 2696]
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupati
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	
		Shining							

```
UserId2696 = FinalData[FinalData["UserID"] == userId]
UserId2696
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupati
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	25	
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	25	
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	25	
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	25	
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	25	
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	25	

** FEATURE ENGINEERING**

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)

```
#make new dataset for genres, by spliting the Movies according to different genres category.

Genres_Df= FinalData["Genres"].str.split("|")
Genres_Df
```

```
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
```

```
Genres = set()                    # taking the uniques genres from the list of genres.
for genre in Genres_Df:
   Genres =Genres.union(set(genre))
```

```
Genres # Unique genres list
```

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

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.

```
OneHotencoding = FinalData["Genres"].str.get_dummies("|")
```

1. Pandas str.get_dummies() is used to separate each string in the caller series at the passed separator. A data frame is returned with all the possible values after splitting every string.
2. If the text value in original data frame at same index contains the string (Column name/ Splited values) then the value at that position is 1 otherwise, 0.

cikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

```
OneHotencoding
```

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

1000209 rows × 18 columns

```
FinalData1=FinalData
```

```
FinalData1
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1
...	...	...	...	...	...	...	...	...
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	M	25

```
FinalData1 = pd.concat([FinalData,OneHotencoding],axis=1) #Conactinating the finalData and OneHotencoding DFs
```

```
FinalData1.head()
```

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

```
FinalData1.columns
```

```
Index(['MovieID', 'Title', 'Genres', 'UserID', 'Rating', 'Timestamp', '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')
```

Determine the features affecting the ratings of any particular movie.

Series.str can be used to access the values of the series as strings and apply several methods to it. Pandas Series.str.extract() function is used to extract capture groups in the regex pat as columns in a DataFrame. For each subject string in the Series, extract groups from the first match of regular expression pat.

1. Syntax: Series.str.extract(pat, flags=0, expand=True)
2. Parameter : pat : Regular expression pattern with capturing groups. flags : int, default 0 (no flags) expand : If True, return DataFrame with one column per capture group.

Returns : DataFrame or Series or Index

(https://www.geeksforgeeks.org/python-pandas-series-str-extract/)

```
Df2=FinalData[["title","Years"]] = FinalData1.Title.str.extract("(.)\s\((.\d+)",expand=True)
```

```
Df2
```

	0	1
0	y	1995
1	s	1995
2	3	1995
3	e	1977
4	t	1993
...	...	...
1000204	t	2000
1000205	o	2000
1000206	h	2000
1000207	1	2000
1000208	r	2000

1000209 rows × 2 columns

```
FinalData.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupa
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	

```
FinalData1[["title","Years"]] = FinalData1.Title.str.extract("(.)\s\((.\d+)",expand=True)
FinalData1
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1
...	...	...	...	...	...	...	...	...
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	M	25
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	958489970	M	25
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	958489902	M	25
1000207	3555	U-571 (2000)	Action|Thriller	5727	3	958490699	M	25
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	958490171	M	25

1000209 rows × 30 columns

```
FinalData2 = FinalData1.drop(columns=["title"])
FinalData2.head()
```

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

```
FinalData2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 29 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   MovieID      1000209 non-null  int64
 1   Title        1000209 non-null  object
 2   Genres       1000209 non-null  object
 3   UserID       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   Zip-Code     1000209 non-null  object
 10  Action       1000209 non-null  int64
 11  Adventure    1000209 non-null  int64
 12  Animation    1000209 non-null  int64
 13  Children's   1000209 non-null  int64
 14  Comedy       1000209 non-null  int64
 15  Crime        1000209 non-null  int64
 16  Documentary  1000209 non-null  int64
 17  Drama        1000209 non-null  int64
 18  Fantasy      1000209 non-null  int64
 19  Film-Noir    1000209 non-null  int64
 20  Horror       1000209 non-null  int64
 21  Musical      1000209 non-null  int64
 22  Mystery      1000209 non-null  int64
 23  Romance      1000209 non-null  int64
 24  Sci-Fi       1000209 non-null  int64
 25  Thriller     1000209 non-null  int64
 26  War          1000209 non-null  int64
 27  Western      1000209 non-null  int64
 28  Years        1000209 non-null  object
dtypes: int64(24), object(5)
memory usage: 228.9+ MB
```

```
df4=FinalData2['Years'] = FinalData2.Years.astype(int)
```

```
df4.head()
```

```
0    1995
1    1995
2    1995
3    1977
4    1993
Name: Years, dtype: int64
```

```
FinalData2['Years'] = FinalData2.Years.astype(int) # As we can see thar the Years is object type so converting it into int type
```

```
FinalData2['Movie_Age'] = 2000 - FinalData2.Years  # ADDING NEW COLUMN
```

```
FinalData2.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupa
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	

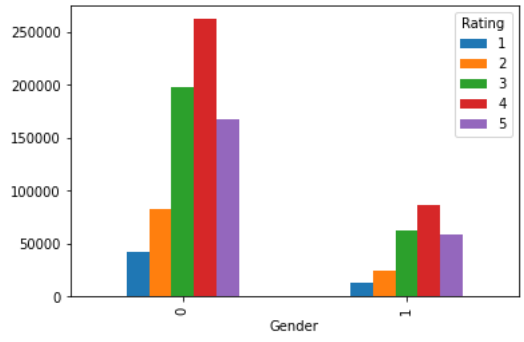
```
FinalData2['Gender'] = FinalData2.Gender.str.replace('F','1')

FinalData2['Gender'] = FinalData2.Gender.str.replace('M','0')

FinalData2['Gender'] = FinalData2.Gender.astype(int)

FinalData2.head()
```

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

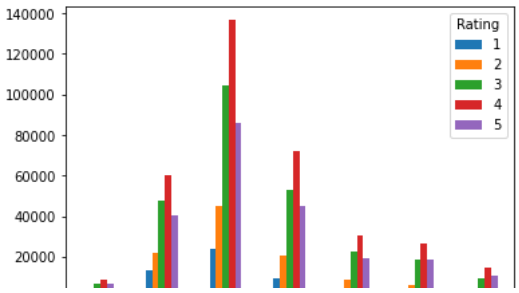
```
GenderAffecting = FinalData2.groupby('Gender').size().sort_values(ascending=False)[:25]

GenderAffecting
```

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

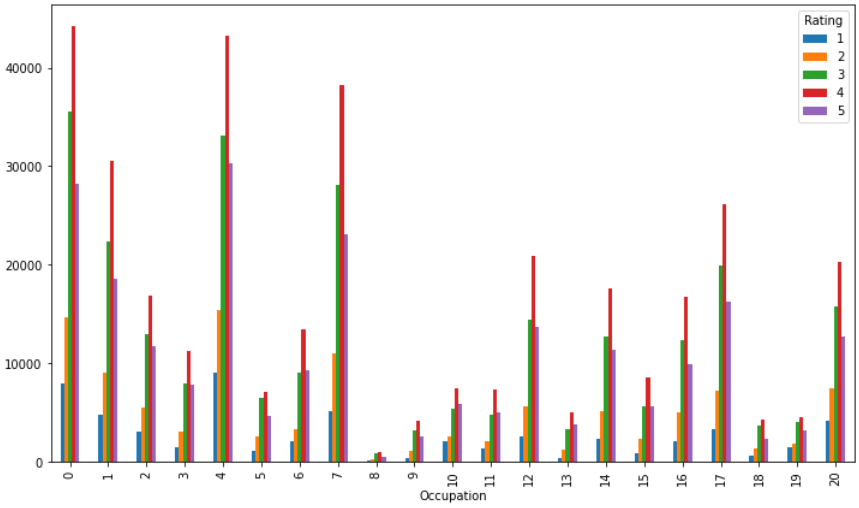
```
FinalData2.groupby(["Gender","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
plt.show()
```



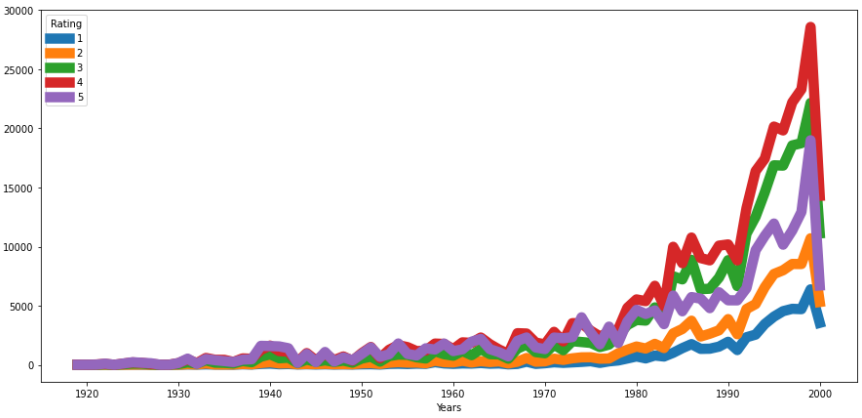

```
FinalData2.groupby(["Age","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
plt.show()
```



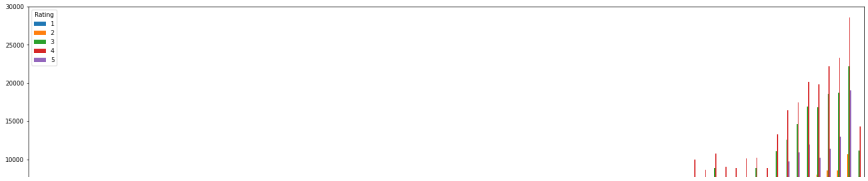

```
FinalData2.groupby(["Occupation","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True , linewidth = '20.5' , figsize=(12
plt.show()
```



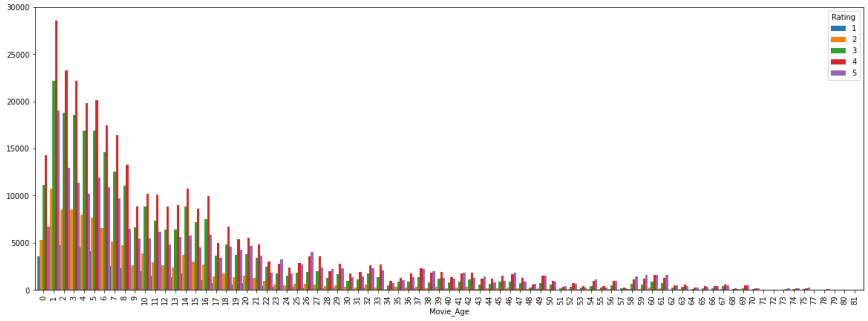

```
FinalData2.groupby(["Years","Rating"]).size().unstack().plot(kind='line',stacked=False,legend=True , linewidth = '10.5', figsize=(15,7))
plt.show()
```




```
FinalData2.groupby(["Years","Rating"]).size().unstack().plot(kind='bar',stacked=False ,legend=True , figsize=(25,7))
plt.show()
```



```
FinalData2.groupby(["Movie_Age","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True , width=1.2, figsize=(20,7))
plt.show()
```




```
first_500 = FinalData2[:1000]
```

```
first_500
```

	MovieID	Title	Genres	UserID	Rating	Timestamp
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
...	...	...	...	...	...	...
995	2384	Babe: Pig in the City (1998)	Children's|Comedy	18	2	978155233
996	2391	Simple Plan, A (1998)	Crime|Thriller	18	1	978155685
997	2394	Prince of Egypt, The (1998)	Animation|Musical	18	4	978154907
998	2402	Rambo: First Blood Part II (1985)	Action|War	18	2	978153894
999	2404	Rambo III (1988)	Action|War	18	2	978153977

1000 rows × 30 columns

```
FinalData2
```

	MovieID	Title	Genres	UserID	Rating	Times
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	97882
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	97882
2	150	Apollo 13 (1995)	Drama	1	5	97830
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	97830
4	527	Schindler's List (1993)	Drama|War	1	5	97882
...	...	...	...	...	...	
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	95848
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	95848
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	95848
1000207	3555	U-571 (2000)	Action|Thriller	5727	3	95849
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	95849

1000209 rows × 30 columns

```
gender_effect = FinalData1['Gender'].str.get_dummies()
```

```
gender_effect
```

	F	M
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
1000204	0	1
1000205	0	1
1000206	0	1
1000207	0	1
1000208	0	1

1000209 rows × 2 columns

```
FinalData3 = FinalData1
```

```
FinalData3 = pd.merge(FinalData1, gender_effect, how='inner', left_index=True, right_index=True)
```

```
FinalData3.head(5)
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	(
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	

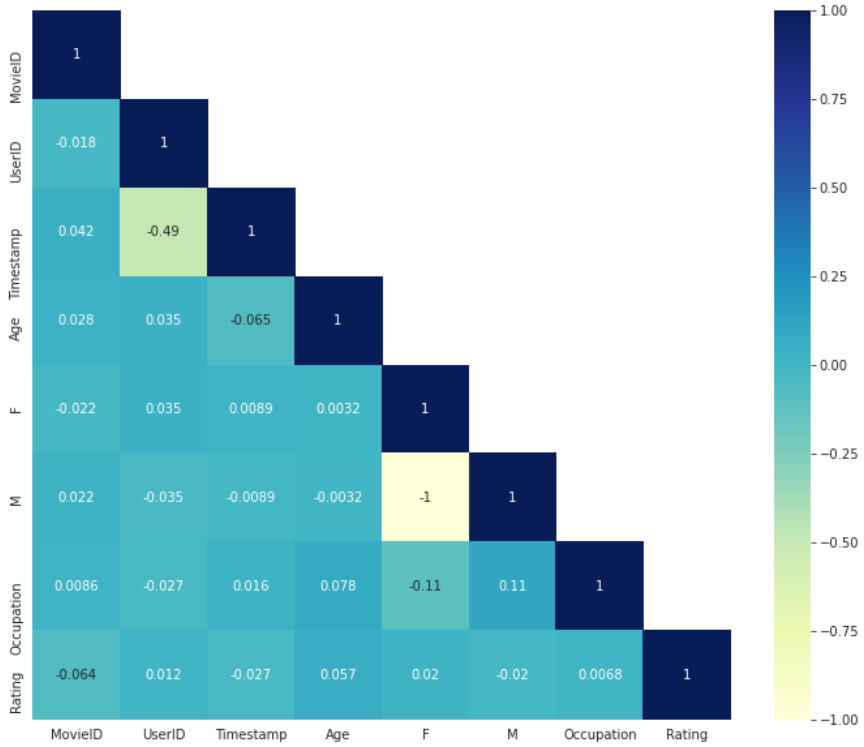
```
# Creating Heatmap feature affecting the Movies
Heat_Map = np.ones_like(FinalData3[["MovieID", "UserID", "Timestamp", "Age" ,"F", "M", "Occupation", "Rating"]].corr())
Heat_Map
```

```
array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

```
Heat_Map[np.tril_indices_from(Heat_Map)] = 0
```

```
sns.set_style("white")
plt.figure(figsize=(12,10))
sns.heatmap(FinalData3[["MovieID", "UserID", "Timestamp", "Age" ,"F", "M", "Occupation", "Rating"]].corr(),
annot=True, cmap='YlGnBu', mask=Heat_Map)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b09ff9450>
```




```
FinalData3
```

	MovieID	Title	Genres	UserID	Rating	Times
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	97882
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	97882
2	150	Apollo 13 (1995)	Drama	1	5	97830
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	97830
4	527	Schindler's List (1993)	Drama|War	1	5	97882
...	...	...	...	...	...	
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	95848
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	95848

```
features = FinalData3[['MovieID', 'UserID', 'F', 'M', 'Timestamp', 'Age', 'Occupation', 'Action', 'Adventure', 'Animation',
"Children's", "Comedy", 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
'Sci-Fi', 'Thriller', 'War', 'Western', 'Rating']]
```

```
features
```

	MovieID	UserID	F	M	Timestamp	Age	Occupation	Action	Adventure	Animation	Children's	C
0	1	1	1	0	978824268	1	10	0	0	1	1	
1	48	1	1	0	978824351	1	10	0	0	1	1	
2	150	1	1	0	978301777	1	10	0	0	0	0	
3	260	1	1	0	978300760	1	10	1	1	0	0	
4	527	1	1	0	978824195	1	10	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
1000204	3513	5727	0	1	958489970	25	4	0	0	0	0	
1000205	3535	5727	0	1	958489970	25	4	0	0	0	0	
1000206	3536	5727	0	1	958489902	25	4	0	0	0	0	
1000207	3555	5727	0	1	958490699	25	4	1	0	0	0	
1000208	3578	5727	0	1	958490171	25	4	1	0	0	0	

1000209 rows × 26 columns

```
x = features.drop(['Rating'], axis=1)
y = features['Rating']
x
y
```

```
0          5
1          5
2          5
3          4
4          5
          ..
1000204    4
1000205    2
1000206    5
1000207    3
1000208    5
Name: Rating, Length: 1000209, dtype: int64
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

class sklearn.feature_selection.SelectKBest (score_func=, *, k=10)

```
# selecting top 5 features affecting the Movies
Best_Features = SelectKBest(score_func=chi2, k=10)
fit = Best_Features.fit(x,y)
scores_df = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,scores_df], axis=1)
featureScores.columns = ['Features', 'Score']
print(featureScores.nlargest(10, 'Score'))
```

```
      Features         Score
4   Timestamp  1.638876e+08
0     MovieID  3.341344e+06
1      UserID  1.720993e+05
5         Age  1.865494e+04
14      Drama  9.705111e+03
17     Horror  9.192491e+03
23        War  6.405101e+03
16  Film-Noir  3.826022e+03
21     Sci-Fi  1.845628e+03
7      Action  1.734519e+03
```

```
Features_df = features[['Timestamp', 'MovieID', 'UserID', 'Age', 'Drama', 'Horror', 'War', 'Film-Noir', 'Sci-Fi', 'Action']]
Features_df.head(10)
```

	Timestamp	MovieID	UserID	Age	Drama	Horror	War	Film-Noir	Sci-Fi	Action
0	978824268	1	1	1	0	0	0	0	0	0
1	978824351	48	1	1	0	0	0	0	0	0
2	978301777	150	1	1	1	0	0	0	0	0
3	978300760	260	1	1	0	0	0	0	1	1
4	978824195	527	1	1	1	0	1	0	0	0
5	978302149	531	1	1	1	0	0	0	0	0
6	978824268	588	1	1	0	0	0	0	0	0
7	978302268	594	1	1	0	0	0	0	0	0
8	978824268	595	1	1	0	0	0	0	0	0
9	978301398	608	1	1	1	0	0	0	0	0

sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)

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

```
# split the dataset into tain and test dataset
X_train, X_test, y_train, y_test = train_test_split(Features_df,y,test_size=0.20, random_state=1)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((800167, 10), (200042, 10), (800167,), (200042,))
```

This is a classification type problem, so we can use following method to make model - Naive Bayes K-NearestNeghbors Decision Tree Random Forest

Using K-Nearest Neighbors

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, , *weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None,* *kwargs)

```
Knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)
Knnaccuracy = Knn.score(X_test, y_test)
Knn_predictions = Knn.predict(X_test)
print("KNN test accuracy: {:.4f}".format(Knnaccuracy))
print("KNN train accuracy: {:.4f}".format(Knn.score(X_train, y_train)))
```

```
KNN test accuracy: 0.3568
KNN train accuracy: 0.5238
```

```
from sklearn.neighbors import KNeighborsClassifier
```