```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,8)
import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
Start coding or generate with AI.
```

```
movies = pd.read_csv("/content/zee-movies.csv")
ratings =pd.read_csv("/content/zee-ratings.csv")
users = pd.read_csv("/content/zee-users.csv" )
```

```
delimiter =":::"

users = users["UserID::Gender::Age::Occupation::Zip-code"].str.split(delimiter,expand = True)
users.columns = ["UserID","Gender","Age","Occupation","Zipcode"]

users["Age"].replace({"1": "Under 18","18": "18-24","25": "25-34",
                          "35": "35-44","45": "45-49","50": "50-55","56": "56+"},inplace=True)

users["Occupation"] = users["Occupation"].astype(int).replace({0: "other",1: "academic/educator",2: "artist",
                                                               3: "clerical/admin",4: "college/grad student",
                                             5: "customer service",6: "doctor/health care",7: "executive/managerial",
                                             8: "farmer" ,9: "homemaker",10: "K-12 student",11: "lawyer",
                                             12: "programmer",13: "retired",14: "sales/marketing",15: "scientist",
                                             16: "self-employed",17: "technician/engineer",
                                             18: "tradesman/craftsman",19: "unemployed",20: "writer"},
                                            )

delimiter =":::"

ratings = ratings["UserID::MovieID::Rating::Timestamp"].str.split(delimiter,expand = True)
ratings.columns = ["UserID","MovieID","Rating","Timestamp"]


movies.drop(["Unnamed: 1","Unnamed: 2"],axis = 1,inplace=True)



delimiter =":::"

movies = movies["Movie ID::Title::Genres"].str.split(delimiter,expand = True)
movies.columns = ["MovieID","Title","Genres"]


movies.shape,ratings.shape,users.shape
```

```
((3883, 3), (1000209, 4), (6040, 5))
```

```
movies # need to take care of Genres .
```

	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

```
ratings # need to convert timestamp to hrs.
```

	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

```
users
```

	UserID	Gender	Age	Occupation	Zipcode
0	1	F	Under 18	K-12 student	48067
1	2	M	56+	self-employed	70072
2	3	M	25-34	scientist	55117
3	4	M	45-49	executive/managerial	02460
4	5	M	25-34	writer	55455
...	...	...	...	...	...
6035	6036	F	25-34	scientist	32603
6036	6037	F	45-49	academic/educator	76006
6037	6038	F	56+	academic/educator	14706
6038	6039	F	45-49	other	01060
6039	6040	M	25-34	doctor/health care	11106

6040 rows × 5 columns

```
# taking out the release year from the title column from movie table :

movies["Release_year"] = movies["Title"].str.extract('^(.+)\s\(([0-9]*)\)$',expand = True)[1]
movies["Title"] = movies["Title"].str.split("(").apply(lambda x:x[0])


# Converting timestamp to hours

from datetime import datetime
ratings["Watch_Hour"] =ratings["Timestamp"].apply(lambda x:datetime.fromtimestamp(int(x)).hour)
ratings.drop(["Timestamp"],axis = 1,inplace=True)
```

```
Start coding or generate with AI.
```

```
movies.shape,ratings.shape,users.shape
```

```
((3883, 4), (1000209, 4), (6040, 5))
```

Merging all the tables into one data frame :

```
df = users.merge(movies.merge(ratings,on="MovieID",how="outer"),on="UserID",how="outer")
```

```
df.shape
```

```
(1000386, 11)
```

```
df
```

```
df
```

	UserID	Gender	Age	Occupation	Zipcode	MovieID	Title	Genres	Release_year	Rating	Watch_Hour
0	1	F	Under 18	K-12 student	48067	1	Toy Story	Animation|Children's|Comedy	1995	5	23.0
1	1	F	Under 18	K-12 student	48067	48	Pocahontas	Animation|Children's|Musical|Romance	1995	5	23.0
2	1	F	Under 18	K-12 student	48067	150	Apollo 13	Drama	1995	5	22.0
3	1	F	Under 18	K-12 student	48067	260	Star Wars: Episode IV - A New Hope	Action|Adventure|Fantas	1977	4	22.0
4	1	F	Under 18	K-12 student	48067	527	Schindler's List	Drama|War	1993	5	23.0
...	...	...	...	...	...	...	...	...	...	...	...
1000381	NaN	NaN	NaN	NaN	NaN	3650	Anguish	Horror	1986	NaN	NaN
1000382	NaN	NaN	NaN	NaN	NaN	3750	Boricua's Bond	Drama	2000	NaN	NaN
1000383	NaN	NaN	NaN	NaN	NaN	3829	Mad About Mambo	Comedy|Romance	2000	NaN	NaN
1000384	NaN	NaN	NaN	NaN	NaN	3856	Autumn Heart	Drama	1999	NaN	NaN
1000385	NaN	NaN	NaN	NaN	NaN	3907	Prince of Central Park, The	Drama	1999	NaN	NaN

1000386 rows × 11 columns

```
df_ = df.copy()
```

```
df_.dropna(inplace=True)
```

```
df_.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 996144 entries, 0 to 1000208
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   UserID        996144 non-null  object 
 1   Gender        996144 non-null  object 
 2   Age           996144 non-null  object 
 3   Occupation    996144 non-null  object 
 4   Zipcode       996144 non-null  object 
 5   MovieID       996144 non-null  object 
 6   Title         996144 non-null  object 
 7   Genres        996144 non-null  object 
 8   Release_year  996144 non-null  object 
 9   Rating        996144 non-null  object 
 10  Watch_Hour    996144 non-null  float64
dtypes: float64(1), object(10)
memory usage: 91.2+ MB
```

```
df_['Release_year']=df_['Release_year'].astype('int32')
df_['Rating']=df_['Rating'].astype('int32')
```

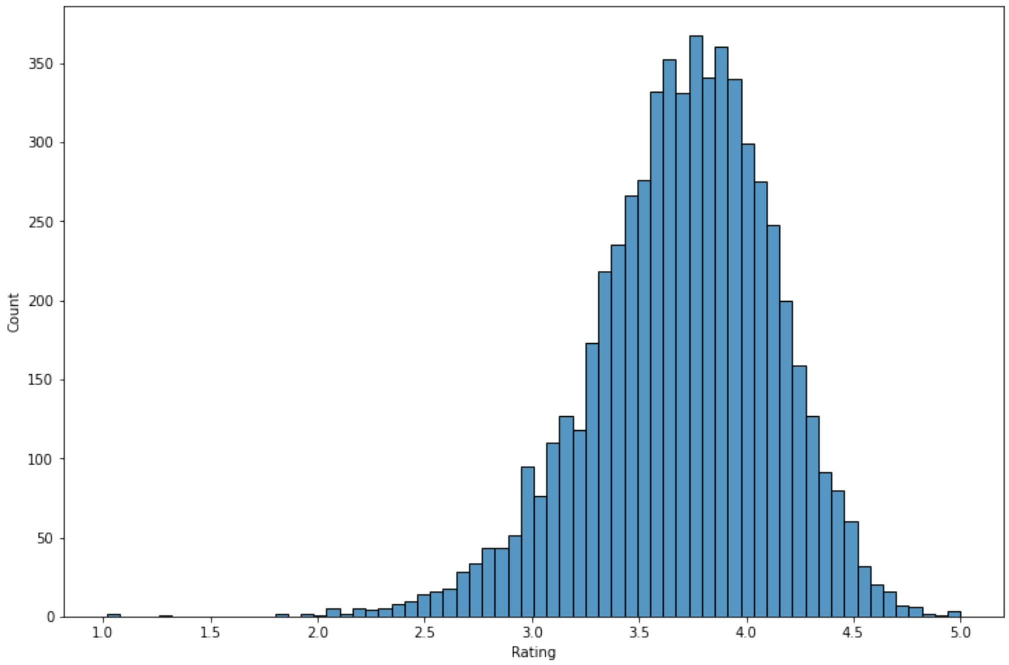
```
bins = [1919, 1929, 1939, 1949, 1959, 1969, 1979, 1989, 2000]
labels = ['20s', '30s', '40s', '50s', '60s', '70s', '80s', '90s']

df_["Released_In"] =  pd.cut(df_['Release_year'], bins=bins, labels=labels)
```

```
import seaborn as sns
```

Average user rating distribution :

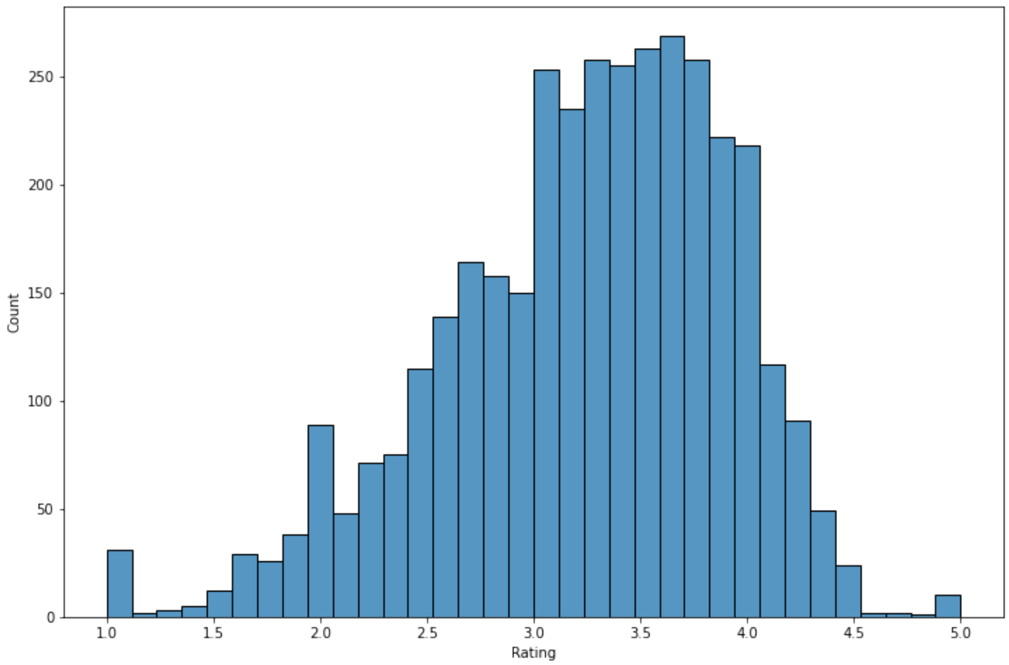
```
sns.histplot(df_[['UserID','Rating']].groupby('UserID').mean()["Rating"])

# average ratings given by each user distribution
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f082ee1d130>
```



```
sns.histplot(df_[['MovieID','Rating']].groupby('MovieID').mean()["Rating"])

# average rating , that each movie has receieved by users .
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f082eb12490>
```



```
df_["MovieID"].nunique()
```

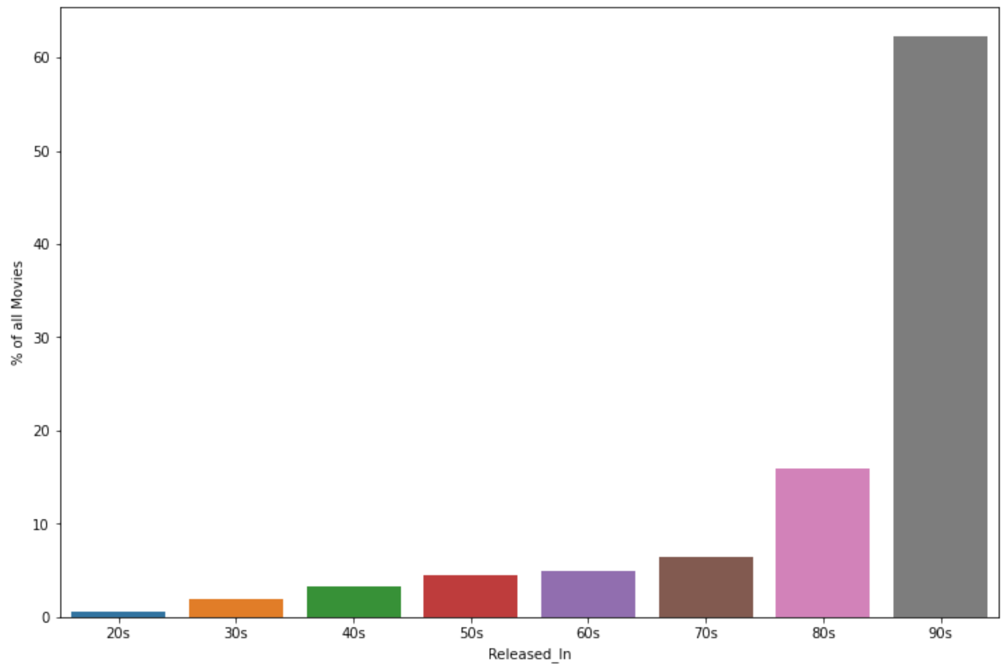
```
3682
```

```
movies_per_decade = df_[['MovieID','Released_In']].groupby('Released_In').nunique()
movies_per_decade["% of all Movies"] = (movies_per_decade["MovieID"]/(df_["MovieID"].nunique())) * 100
movies_per_decade
```

	MovieID	% of all Movies
Released_In		
20s	23	0.624661
30s	71	1.928300
40s	120	3.259098
50s	164	4.454101
60s	184	4.997284
70s	237	6.436719
80s	586	15.915263
90s	2294	62.303096

```
sns.barplot(movies_per_decade.index,
            movies_per_decade["% of all Movies"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f082e91ed90>
```



Start coding or generate with AI.

Start coding or generate with AI.

```
m = movies[["MovieID","Title","Genres"]]
m["Genres"] = m["Genres"].str.split("|")
m = m.explode("Genres")
m["Genres"] = m["Genres"].replace({"":"Other","Horro":"Horror","Sci-":"Sci-Fi","Sci":"Sci-Fi","Sci-F":"Sci-Fi","Dr":"Drama","Documenta":"Documentary",
                     "Wester":"Western","Fant":"Fantasy","Chil":"Children's","R":"Romance","D":"Drama","Rom":"Romance","Animati":"Animation","Childr":"Children's","Childre":"Chi
                     "Fantas":"Fantasy","Come":"Comedy","Dram":"Drama","S":"Sci-Fi","Roma":"Romance","A":"Adventure","Children":"Children's","Adventu":"Adventure","Adv":"Adventu
                      "Wa":"War","Thrille"  :"Thriller","Com":"Comedy","Comed":"Comedy","Acti":"Action","Advent":"Adventure","Adventur":"Adventure","Thri":"Thriller",
                        "Chi":"Children's","Ro":"Romance","F":"Fantasy","We":"Western","Documen":"Documentary","Music":"Musical","Children":"Children's" ,"Horr":"Horror",
                     "Children'":"Children's","Roman":"Romance","Docu":"Documentary","Th":"Thriller","Document":"Documentary"
                    })

m = m.pivot_table(values="Title", index="MovieID", columns="Genres", aggfunc= np.size,).fillna(0)


def apply(x):
  if x >= 1:
    return 1
  else:
    return 0

m["Adventure"] = m["Adventure"].apply(apply)
m = m.astype(int)
```

```
m
```

Genres	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Other	Romance	Sci-Fi	Thriller	War	Western
MovieID																			
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
100	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
1000	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1001	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
994	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
996	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
997	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
998	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

3858 rows × 19 columns

```
final_data = df.merge(m,on="MovieID",how="left").drop(["Genres"],axis = 1)
```

```
final_data
```

	UserID	Gender	Age	Occupation	Zipcode	MovieID	Title	Release_year	Rating	Watch_Hour	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary
0	1	F	Under 18	K-12 student	48067	1	Toy Story	1995	5	23.0	0.0	0.0	1.0	1.0	1.0	0.0	0.(
1	1	F	Under 18	K-12 student	48067	48	Pocahontas	1995	5	23.0	0.0	0.0	1.0	1.0	0.0	0.0	0.(
2	1	F	Under 18	K-12 student	48067	150	Apollo 13	1995	5	22.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
3	1	F	Under 18	K-12 student	48067	260	Star Wars: Episode IV - A New Hope	1977	4	22.0	1.0	1.0	0.0	0.0	0.0	0.0	0.(
4	1	F	Under 18	K-12 student	48067	527	Schindler's List	1993	5	23.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
1000381	NaN	NaN	NaN	NaN	NaN	3650	Anguish	1986	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.(
1000382	NaN	NaN	NaN	NaN	NaN	3750	Boricua's Bond	2000	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.(
1000383	NaN	NaN	NaN	NaN	NaN	3829	Mad About Mambo	2000	NaN	NaN	0.0	0.0	0.0	0.0	1.0	0.0	0.(
1000384	NaN	NaN	NaN	NaN	NaN	3856	Autumn Heart	1999	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.(
1000385	NaN	NaN	NaN	NaN	NaN	3907	Prince of Central Park, The	1999	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.(

1000386 rows × 29 columns

```
Start coding or generate with AI.
```

```
final_data.MovieID = final_data.MovieID.astype(int)
final_data.UserID = final_data.UserID.astype(float)
final_data.Release_year = final_data.Release_year.astype(float)
```

```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000386 entries, 0 to 1000385
Data columns (total 29 columns):
 #   Column        Non-Null Count    Dtype
---  ------        --------------    -----
 0   UserID        1000209 non-null  float64
 1   Gender        1000209 non-null  object
 2   Age           1000209 non-null  object
 3   Occupation    1000209 non-null  object
 4   Zipcode       1000209 non-null  object
 5   MovieID       1000386 non-null  int64
 6   Title         1000386 non-null  object
 7   Release_year  996606 non-null   float64
 8   Rating        1000209 non-null  object
 9   Watch_Hour    1000209 non-null  float64
 10  Action        996320 non-null   float64
 11  Adventure     996320 non-null   float64
 12  Animation     996320 non-null   float64
 13  Children's    996320 non-null   float64
 14  Comedy        996320 non-null   float64
 15  Crime         996320 non-null   float64
 16  Documentary   996320 non-null   float64
 17  Drama         996320 non-null   float64
 18  Fantasy       996320 non-null   float64
 19  Film-Noir     996320 non-null   float64
 20  Horror        996320 non-null   float64
 21  Musical       996320 non-null   float64
 22  Mystery       996320 non-null   float64
 23  Other         996320 non-null   float64
 24  Romance       996320 non-null   float64
 25  Sci-Fi        996320 non-null   float64
 26  Thriller      996320 non-null   float64
 27  War           996320 non-null   float64
 28  Western       996320 non-null   float64
dtypes: float64(22), int64(1), object(6)
memory usage: 229.0+ MB
```

```
final_data.describe()
```

	UserID	MovieID	Release_year	Watch_Hour	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Dr
count	1.000209e+06	1.000386e+06	996606.000000	1.000209e+06	996320.000000	996320.000000	996320.000000	996320.000000	996320.000000	996320.000000	996320.000000	996320.00
mean	3.024512e+03	1.865526e+03	1986.758010	1.191620e+01	0.257534	0.134088	0.043107	0.072154	0.354897	0.079690	0.007845	0.35
std	1.728413e+03	1.096030e+03	14.314345	7.894465e+00	0.437276	0.340747	0.203097	0.258742	0.478482	0.270813	0.088223	0.47
min	1.000000e+00	1.000000e+00	1919.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	1.506000e+03	1.030000e+03	1982.000000	4.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	3.070000e+03	1.835000e+03	1992.000000	1.400000e+01	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	4.476000e+03	2.770000e+03	1997.000000	1.900000e+01	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.00
max	6.040000e+03	3.952000e+03	2000.000000	2.300000e+01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

```
final_data.describe(include="object")
```

	Gender	Age	Occupation	Zipcode	Title	Rating
count	1000209	1000209	1000209	1000209	1000386	1000209
unique	2	7	21	3439	3833	5
top	M	25-34	college/grad student	94110	American Beauty	4
freq	753769	395556	131032	3802	3428	348971

```
final_data.nunique()
```

```
UserID          6040
Gender             2
Age                7
Occupation        21
Zipcode         3439
MovieID         3883
Title           3833
Release_year      81
Rating             5
Watch_Hour        24
Action             2
Adventure          2
Animation          2
Children's         2
Comedy             2
Crime              2
Documentary        2
Drama              2
Fantasy            2
Film-Noir          2
Horror             2
Musical            2
Mystery            2
Other              2
Romance            2
Sci-Fi             2
Thriller           2
War                2
Western            2
dtype: int64
```

Unique values present in data

- 6040 unique UserID
- 7 different age groups
- 21 occupations
- 3439 different locations of users
- 3883 unique movies

- There are movies available in database , which were never been watched by any user before .
- Thats is the reason we have lots of NaN values in our final dataset.

```
final_data.shape
```

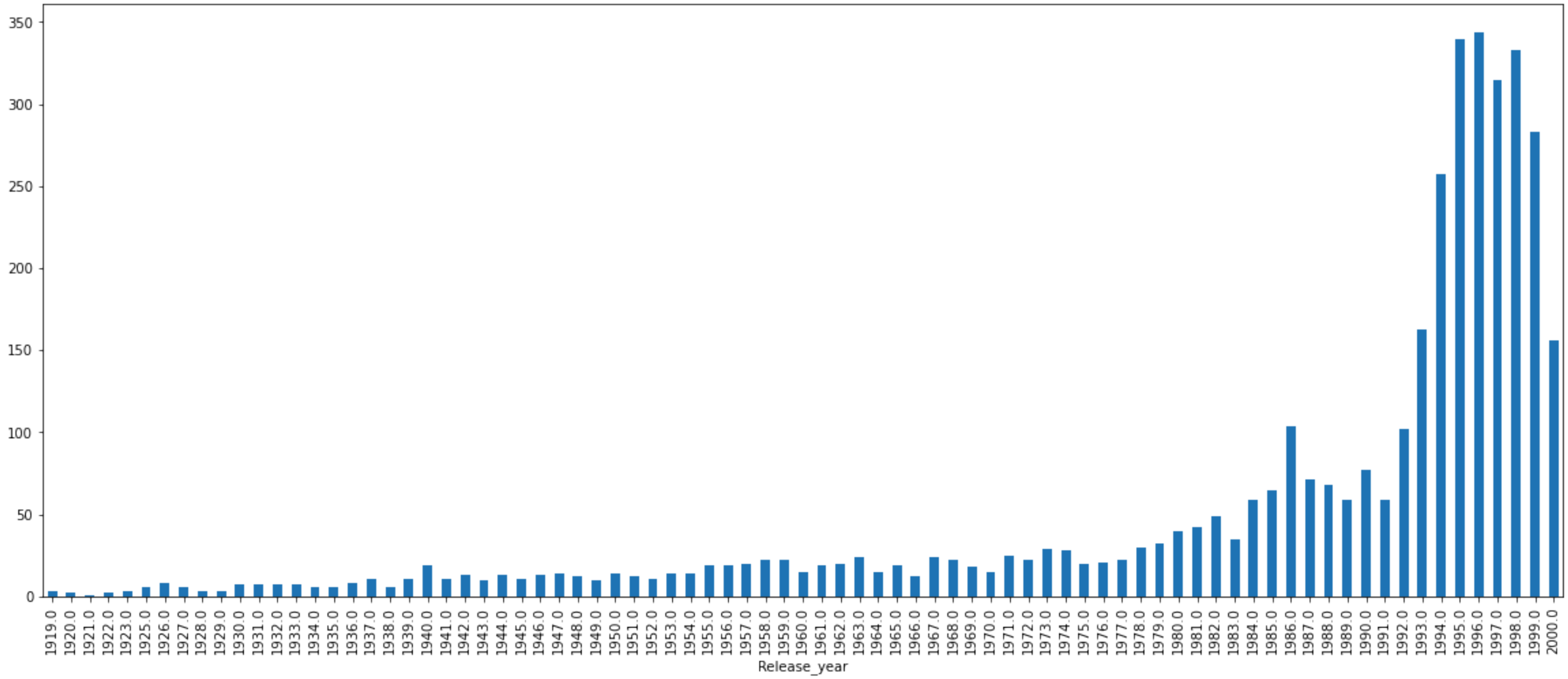
```
(1000386, 29)
```

```
plt.rcParams["figure.figsize"] = (20,8)
```

Most of the movies present in our dataset were released in year:

```
final_data.groupby("Release_year")["Title"].nunique().plot(kind="bar")
```

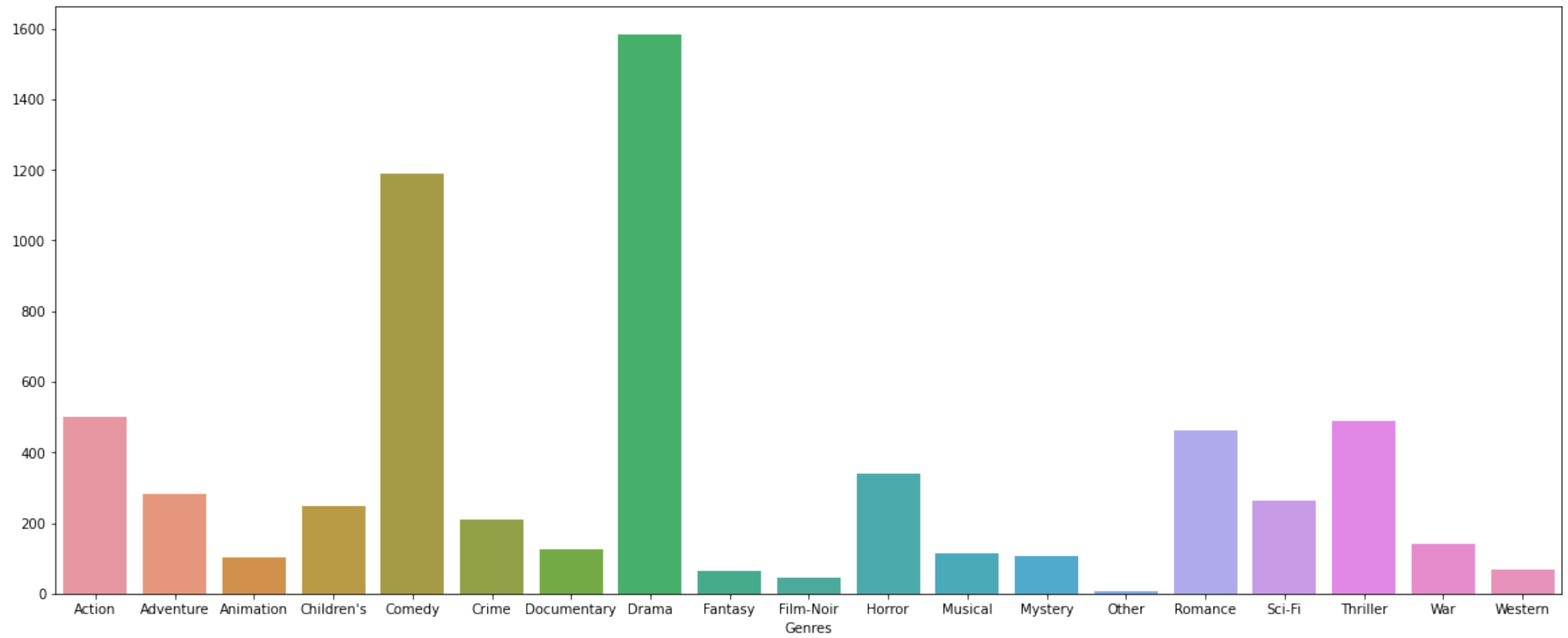
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f082e81b0d0>
```



```
Start coding or generate with AI.
```

```
# Number of Movies per Genres:

sns.barplot(m.sum(axis= 0).index,
            m.sum(axis= 0))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f082e740cd0>
```



```
m.sum(axis= 0)
```

```
Genres
Action          501
Adventure       283
Animation       104
Children's      249
Comedy         1189
Crime           210
Documentary     124
Drama          1585
Fantasy          63
Film-Noir        44
Horror          340
Musical         113
Mystery         105
Other             8
Romance         462
Sci-Fi          265
Thriller        488
War             139
Western          68
dtype: int64
```

```
final_data["Rating"].count()
```

```
1000209
```

Number of movies Rated by each Gender type :

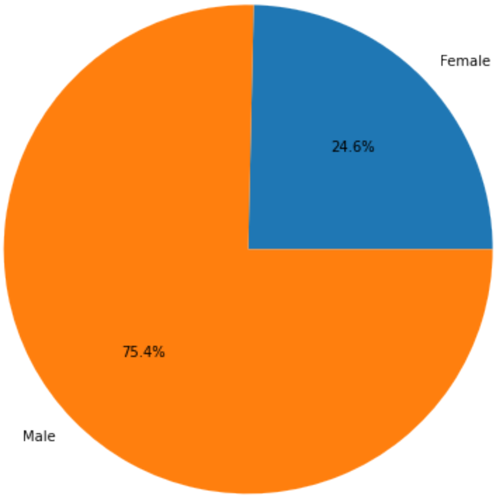
```
# Gender

asd = final_data.groupby("Gender")["Rating"].count() / final_data["Rating"].count() * 100
asd
```

```
Gender
F    24.63885
M    75.36115
Name: Rating, dtype: float64
```

```
Start coding or generate with AI.
```

```
plt.pie(asd, labels = ["Female", "Male"],autopct='%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x7f0847853e20>,
  <matplotlib.patches.Wedge at 0x7f084782b310>],
 [Text(0.7865921924460241, 0.7689426004474957, 'Female'),
  Text(-0.7865921564492223, -0.7689426372705329, 'Male')],
 [Text(0.4290502867887404, 0.4194232366077249, '24.6%'),
  Text(-0.4290502671541212, -0.4194232566930179, '75.4%')])
```

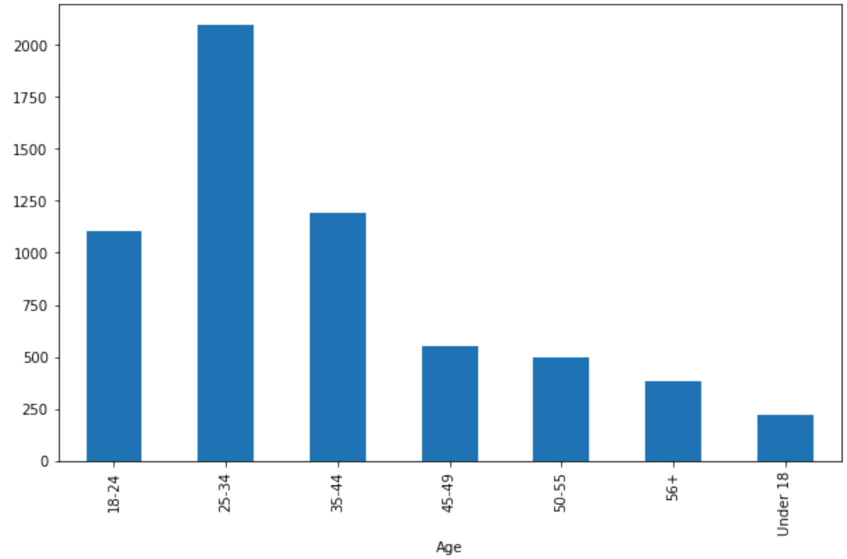


```
Start coding or generate with AI.
```

Users of which age group have watched and rated the most number of movies?

```
plt.rcParams["figure.figsize"] = (10,6)
final_data.groupby("Age")["UserID"].nunique().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08478ed100>
```



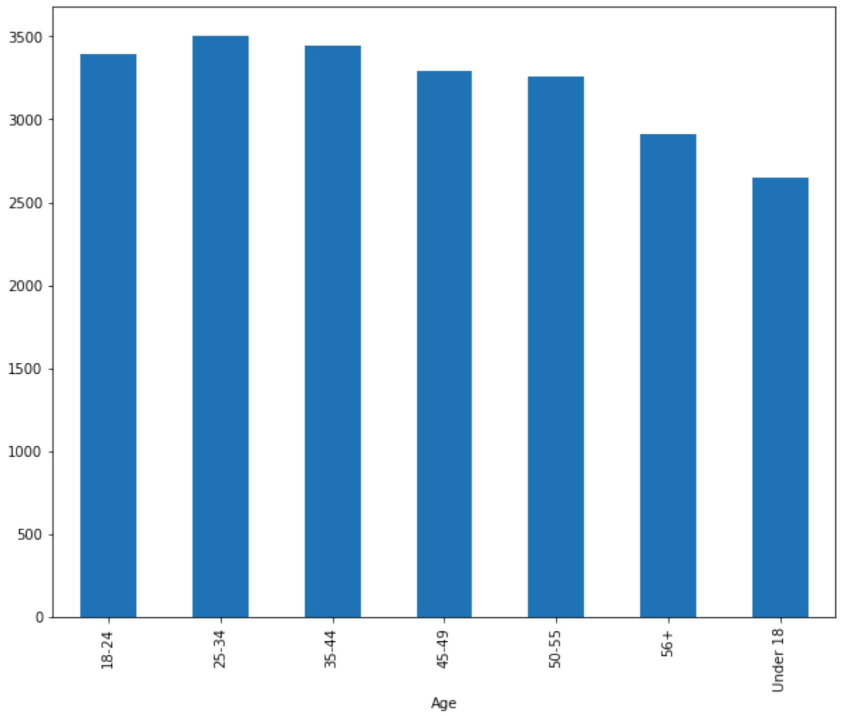
- in DataSet : majority of the viewers are in age group of 25-34
- out of all , 25-34 age group have rated and watched the maximum number of movies.
- for other age groups data are as below:

```
final_data.groupby("Age")["MovieID"].nunique()
```

```
Age
18-24       3393
25-34       3508
35-44       3447
45-49       3288
50-55       3258
56+         2913
Under 18    2650
Name: MovieID, dtype: int64
```

```
plt.rcParams["figure.figsize"] = (10,8)
final_data.groupby("Age")["MovieID"].nunique().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0848d3b4c0>
```

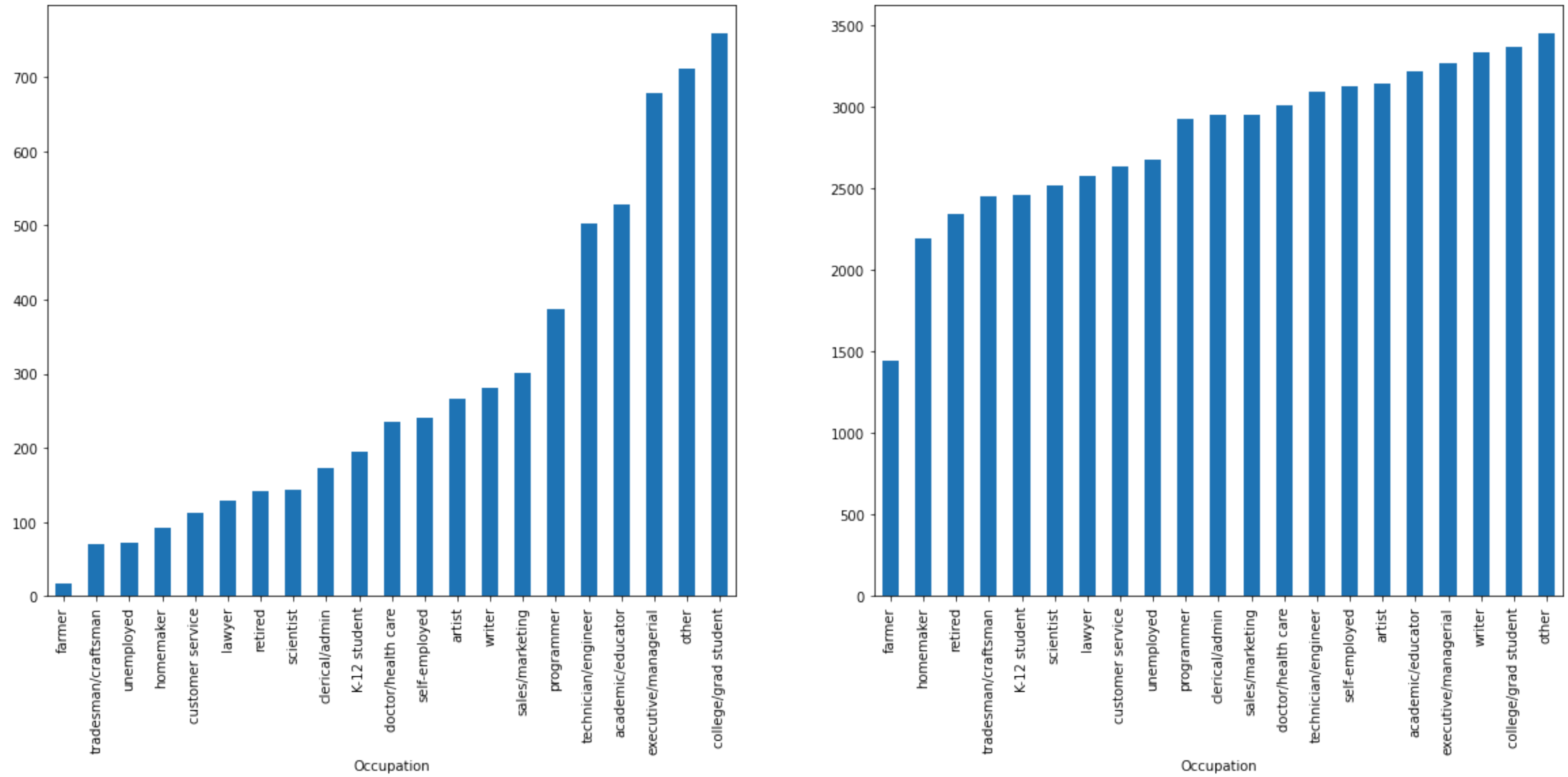


```
Start coding or generate with AI.
```

Users belonging to which profession have watched and rated the most movies?

```
plt.rcParams["figure.figsize"] = (20,8)

plt.subplot(121)
final_data.groupby("Occupation")["UserID"].nunique().sort_values().plot(kind="bar")
plt.subplot(122)
final_data.groupby("Occupation")["MovieID"].nunique().sort_values().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0849c5a160>
```



- Majority of the Users are College Graduates and Students , followed by Executives, educators and engineers. y of the Users are College Graduates and Students , followed by Executives, educators and engineers.
- Maximum movies are watched and rated by user's occupations are College graduate students , writers , executives, educator and artists.

```
final_data.groupby("Occupation")["MovieID"].nunique().sort_values(ascending = False).head(6)
```

```
Occupation
other                   3448
college/grad student    3363
writer                  3330
executive/managerial    3269
academic/educator       3218
artist                  3145
Name: MovieID, dtype: int64
```

```
final_data.columns
```

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

Movie Recommendation based on Genres as per Majority Users occupation :

- below table shows the rank preference of each occupation users:
- higher the number more prefered .

```
## Movie Recommendation based on Genre as per Majority Users :
```

```
np.argsort((final_data.groupby("Occupation")['Action', 'Adventure', 'Animation', "Children's",
                                             'Comedy', 'Crime','Documentary', 'Drama', 'Fantasy',
                                             'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Other',
                                             'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western'].mean())   *100,axis = 1).loc[["writer","artist","academic/educator","executive/ma
```

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Other	Romance	Sci-Fi	Thriller	War
Occupation																		
writer	13	6	18	9	8	12	2	11	17	3	10	5	1	15	14	16	0	4
artist	13	6	18	9	8	12	2	11	17	3	10	5	1	15	14	16	0	4
academic/educator	13	6	18	9	8	2	12	11	10	17	3	5	1	15	14	16	0	4
executive/managerial	13	6	9	18	8	2	11	12	3	17	10	5	1	15	14	16	0	4
college/grad student	13	6	9	18	8	12	11	2	17	10	5	3	1	14	15	16	0	7

- Writers , artists and educator most preferes to watch Animation, Fantasy and Science Fiction movies, followed by Romance , Action and rest of the genres.
- COllege Students most prefer to watch Children's , Science Fiction, Romance and Fantasy movies.
- Film-Noir is more prefered by the educators and Executive occupation users.

```
Start coding or generate with AI.
```

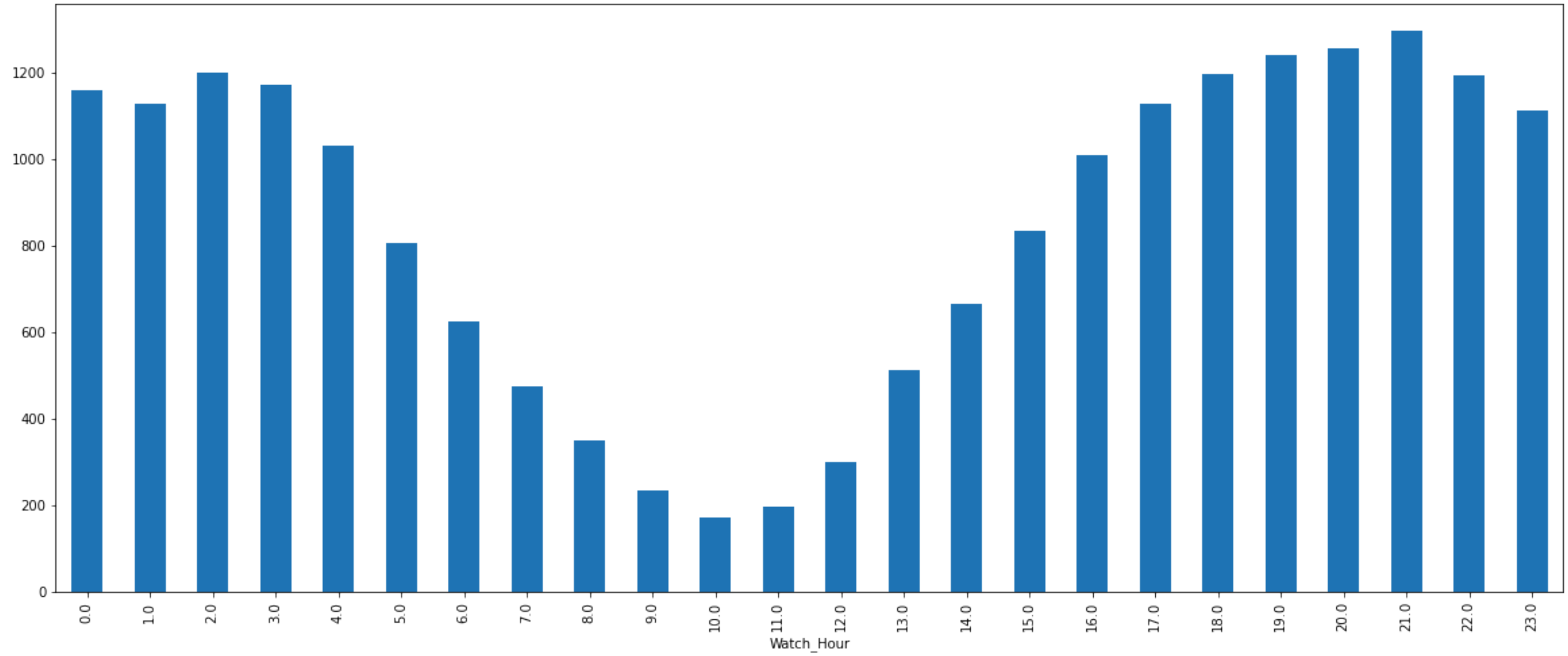
```
Start coding or generate with AI.
```

what is the traffic on OTT, based on watch hour :

```
Start coding or generate with AI.
```

```
final_data.groupby("Watch_Hour")["UserID"].nunique().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f084b993820>
```



```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Top 10 Movies have got the most number of ratings :

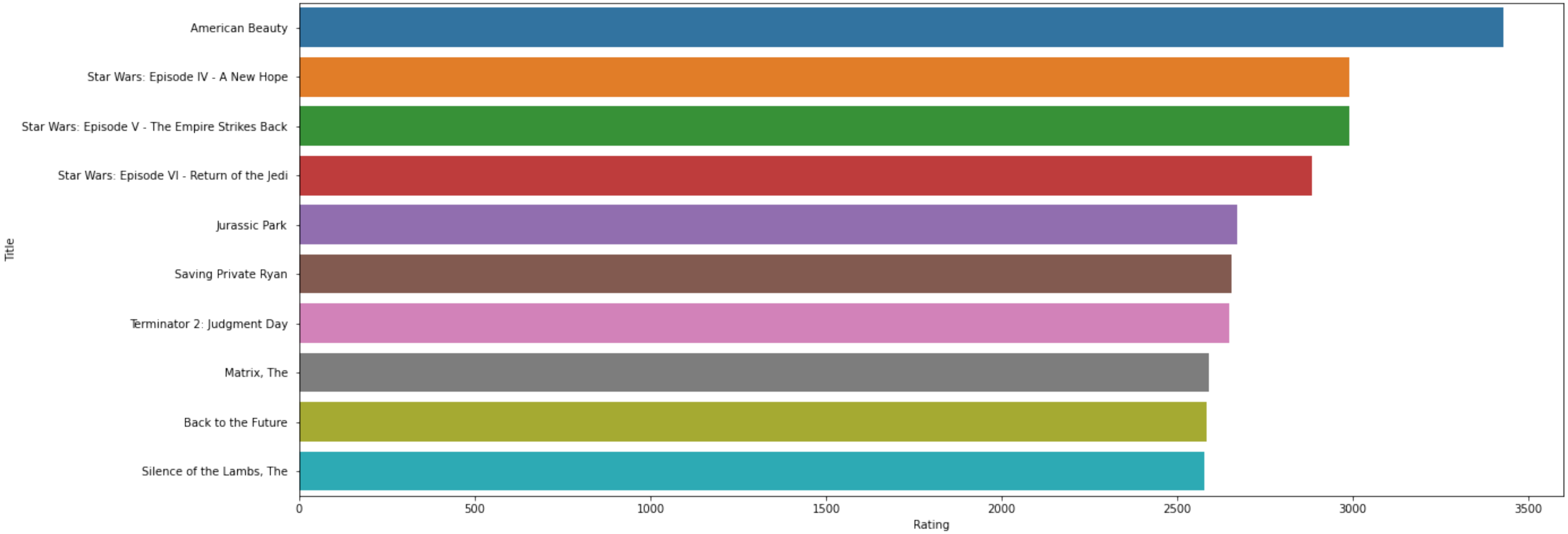
```
top10_movies = final_data.groupby("Title")["Rating"].count().reset_index().sort_values(by="Rating",ascending=False).head(10)
```

```
top10_movies
```

	Title	Rating
127	American Beauty	3428
3261	Star Wars: Episode IV - A New Hope	2991
3262	Star Wars: Episode V - The Empire Strikes Back	2990
3263	Star Wars: Episode VI - Return of the Jedi	2883
1846	Jurassic Park	2672
2994	Saving Private Ryan	2653
3405	Terminator 2: Judgment Day	2649
2186	Matrix, The	2590
262	Back to the Future	2583
3090	Silence of the Lambs, The	2578

```
sns.barplot(y = top10_movies["Title"],
            x = top10_movies["Rating"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08454caf70>
```



```
Start coding or generate with AI.
```

5 Top rated Recommended Movies per each genre :

```
Genres = ['Action', 'Adventure', 'Animation', "Children's", 'Comedy', 'Crime','Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Other','Romance',

for G in Genres:
  print(G)
  print("----------------------")
  print(final_data[final_data[G] == 1].groupby("Title")["Rating"].count().sort_values(ascending=False).head(5))
  print()
  print()
  print()
```

```
Action
----------------------
Title
Star Wars: Episode IV - A New Hope                2991
Star Wars: Episode V - The Empire Strikes Back    2990
Star Wars: Episode VI - Return of the Jedi        2883
Jurassic Park                                     2672
Saving Private Ryan                               2653
Name: Rating, dtype: int64



Adventure
----------------------
Title
Star Wars: Episode IV - A New Hope                2991
Star Wars: Episode V - The Empire Strikes Back    2990
Star Wars: Episode VI - Return of the Jedi        2883
Jurassic Park                                     2672
Men in Black                                      2538
Name: Rating, dtype: int64



Animation
----------------------
Title
Toy Story                      2077
Who Framed Roger Rabbit?       1799
Bug's Life, A                  1703
Toy Story 2                    1585
Aladdin                        1351
Name: Rating, dtype: int64



Children's
----------------------
Title
E.T. the Extra-Terrestrial     2269
Toy Story                      2077
Babe                           1751
Wizard of Oz, The              1718
Bug's Life, A                  1703
Name: Rating, dtype: int64
```

```
Comedy
----------------------
Title
American Beauty         3428
Back to the Future      2583
Men in Black            2538
Shakespeare in Love     2369
Princess Bride, The     2318
Name: Rating, dtype: int64
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Top 5 movie recommended as per age_Group based on ratings each age group provided

```
age_groups = final_data.Age.unique()
```

```
for age_ in age_groups:
  print(age_)
  print("------")
  print(final_data[final_data.Age == age_].groupby("Title")["Rating"].count().sort_values(ascending=False).head())
  print()
  print()
  print()
```

```
Under 18
------
Title
Toy Story                                   112
Sixth Sense, The                            109
Star Wars: Episode IV - A New Hope          101
Men in Black                                100
Star Wars: Episode VI - Return of the Jedi  100
Name: Rating, dtype: int64



56+
------
Title
American Beauty        184
Schindler's List       137
Shakespeare in Love    136
Godfather, The         122
Saving Private Ryan    121
Name: Rating, dtype: int64
```

```
25-34
------
Title
American Beauty                                    1334
Star Wars: Episode V - The Empire Strikes Back     1176
Star Wars: Episode VI - Return of the Jedi         1134
Star Wars: Episode IV - A New Hope                 1128
Terminator 2: Judgment Day                         1087
Name: Rating, dtype: int64



45-49
------
Title
American Beauty                                    258
Star Wars: Episode IV - A New Hope                 243
Star Wars: Episode V - The Empire Strikes Back     226
Jurassic Park                                      218
Shakespeare in Love                                217
Name: Rating, dtype: int64



50-55
------
Title
American Beauty                                    248
Star Wars: Episode IV - A New Hope                 215
Star Wars: Episode V - The Empire Strikes Back     206
Fargo                                              199
Godfather, The                                     198
Name: Rating, dtype: int64
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Creating a user Movie average rating Matrix :

```
df_.columns
```

```
Index(['UserID', 'Gender', 'Age', 'Occupation', 'Zipcode', 'MovieID', 'Title', 'Genres', 'Release_year', 'Rating', 'Watch_Hour', 'Released_In'], dtype='object')
```

```
user_movie_rating_matrix = pd.pivot_table(df_,index = "UserID",
               columns = "Title",
               values = "Rating",
               aggfunc = "mean").fillna(0)
user_movie_rating_matrix.shape
```

```
(6040, 3633)
```

```
user_movie_rating_matrix
```

Title	$1,000,000 Duck	'Night Mother	'Til There Was You	'burbs, The	...And Justice for All	1-900	10 Things I Hate About You	101 Dalmatians	12 Angry Men	13th Warrior, The	187	2 Days in the Valley	20 Dates	20,000 Leagues Under the Sea	200 Cigarettes	2001: A Space Odyssey	2010	24 7: Twenty Four Seven	24-hour Woman	28 Days	Nin Noo Moun
UserID																					
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	3.0	4.0	0.0	0.0	0.0	4.0	0.0	3.0	0.0	0.0	0.0	0.0	
100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	3.0	0.0	0.0	0.0	4.0	4.0	0.0	0.0	0.0	
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
999	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	2.0	0.0	0.0	0.0	0.0	

6040 rows × 3633 columns

```
Start coding or generate with AI.
```

item item similarity(hamming distance) based recommendation :

```
m = movies[["MovieID","Title","Genres"]]
m["Genres"] = m["Genres"].str.split("|")
m = m.explode("Genres")
m["Genres"] = m["Genres"].replace({"":"Other","Horro":"Horror","Sci-":"Sci-Fi","Sci":"Sci-Fi","Sci-F":"Sci-Fi","Dr":"Drama","Documenta":"Documentary",
                     "Wester":"Western","Fant":"Fantasy","Chil":"Children's","R":"Romance","D":"Drama","Rom":"Romance","Animati":"Animation","Childr":"Children's","Childre":"Chi
                     "Fantas":"Fantasy","Come":"Comedy","Dram":"Drama","S":"Sci-Fi","Roma":"Romance","A":"Adventure","Children":"Children's","Adventu":"Adventure","Adv":"Adventu
                      "Wa":"War","Thrille"  :"Thriller","Com":"Comedy","Comed":"Comedy","Acti":"Action","Advent":"Adventure","Adventur":"Adventure","Thri":"Thriller",
                        "Chi":"Children's","Ro":"Romance","F":"Fantasy","We":"Western","Documen":"Documentary","Music":"Musical","Children":"Children's" ,"Horr":"Horror",
                     "Children'":"Children's","Roman":"Romance","Docu":"Documentary","Th":"Thriller","Document":"Documentary"
                    })

m = m.pivot_table(values="Title", index="MovieID", columns="Genres", aggfunc= np.size,).fillna(0)


def apply(x):
  if x >= 1:
    return 1
  else:
    return 0

m["Adventure"] = m["Adventure"].apply(apply)
m = m.astype(int)
```

```
m
```

Genres	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Other	Romance	Sci-Fi	Thriller	War	Western
MovieID																			
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
100	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
1000	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1001	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
994	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
996	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
997	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
998	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

3858 rows × 19 columns

```
def Hamming_distance(x1,x2):
  return np.sum(abs(x1-x2))

Ranks = []
Query = "1"
for candidate in m.index:
  if candidate == Query:
    continue
  Ranks.append([Query,candidate,Hamming_distance(m.loc[Query],m.loc[candidate])])

Ranks = pd.DataFrame(Ranks,columns=["Query","Candidate","Hamming_distance"])
Ranks = Ranks.merge(movies[['MovieID', 'Title']], left_on='Query', right_on='MovieID').rename(columns={'Title': 'query_tittle'}).drop(columns=['MovieID'])
Ranks = Ranks.merge(movies[['MovieID', 'Title']], left_on='Candidate', right_on='MovieID').rename(columns={'Title': 'candidate_tittle'}).drop(columns=['MovieID'])
Ranks = Ranks.sort_values(by=['Query', 'Hamming_distance'])
```

```
Ranks.head(10)
```

	Query	Candidate	Hamming_distance	query_tittle	candidate_tittle
71	1	1064	0	Toy Story	Aladdin and the King of Thieves
1208	1	2141	0	Toy Story	American Tail, An
1442	1	2354	0	Toy Story	Rugrats Movie, The
1443	1	2355	0	Toy Story	Bug's Life, A
2281	1	3114	0	Toy Story	Toy Story 2
2831	1	3611	0	Toy Story	Saludos Amigos
2981	1	3751	0	Toy Story	Chicken Run
7	1	1005	1	Toy Story	D3: The Mighty Ducks
13	1	1010	1	Toy Story	Love Bug, The
19	1	1016	1	Toy Story	Shaggy Dog, The

```
def Hamming_distance(x1,x2):
  return np.sum(abs(x1-x2))

Ranks = []
Query = "1485"
for candidate in m.index:
  if candidate == Query:
    continue
  Ranks.append([Query,candidate,Hamming_distance(m.loc[Query],m.loc[candidate])])

Ranks = pd.DataFrame(Ranks,columns=["Query","Candidate","Hamming_distance"])
Ranks = Ranks.merge(movies[['MovieID', 'Title']], left_on='Query', right_on='MovieID').rename(columns={'Title': 'query_tittle'}).drop(columns=['MovieID'])
Ranks = Ranks.merge(movies[['MovieID', 'Title']], left_on='Candidate', right_on='MovieID').rename(columns={'Title': 'candidate_tittle'}).drop(columns=['MovieID'])
Ranks = Ranks.sort_values(by=['Query', 'Hamming_distance'])
```

```
Ranks.head(10)
```

	Query	Candidate	Hamming_distance	query_tittle	candidate_tittle
4	1485	1001	0	Liar Liar	Associate, The
5	1485	1002	0	Liar Liar	Ed's Next Move
13	1485	101	0	Liar Liar	Bottle Rocket
24	1485	102	0	Liar Liar	Mr. Wrong
25	1485	1020	0	Liar Liar	Cool Runnings
46	1485	104	0	Liar Liar	Happy Gilmore
49	1485	1042	0	Liar Liar	That Thing You Do!
82	1485	1075	0	Liar Liar	Sexual Life of the Belgians, The
86	1485	1079	0	Liar Liar	Fish Called Wanda, A
88	1485	1080	0	Liar Liar	Monty Python's Life of Brian

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
movies = pd.read_fwf("/content/drive/Othercomputers/My Laptop/Data Science Studies/GitHub_Desktop/BusinessCase_Data_Exploration-/Recommender System for OTT /zee-movies.dat",enco
ratings =pd.read_fwf("/content/drive/Othercomputers/My Laptop/Data Science Studies/GitHub_Desktop/BusinessCase_Data_Exploration-/Recommender System for OTT /zee-ratings.dat",enc
users = pd.read_fwf("/content/drive/Othercomputers/My Laptop/Data Science Studies/GitHub_Desktop/BusinessCase_Data_Exploration-/Recommender System for OTT /zee-users.dat",encodi

delimiter =":："

users = users["UserID::Gender::Age::Occupation::Zip-code"].str.split(delimiter,expand = True)
users.columns = ["UserID","Gender","Age","Occupation","Zipcode"]

users["Age"].replace({"1": "Under 18","18": "18-24","25": "25-34",
                          "35": "35-44","45": "45-49","50": "50-55","56": "56+"},inplace=True)

users["Occupation"] = users["Occupation"].astype(int).replace({0: "other",1: "academic/educator",2: "artist",
                                                               3: "clerical/admin",4: "college/grad student",
                                             5: "customer service",6: "doctor/health care",7: "executive/managerial",
                                             8: "farmer" ,9: "homemaker",10: "K-12 student",11: "lawyer",
                                             12: "programmer",13: "retired",14: "sales/marketing",15: "scientist",
                                             16: "self-employed",17: "technician/engineer",
                                             18: "tradesman/craftsman",19: "unemployed",20: "writer"},
                                            )

delimiter =":："

ratings = ratings["UserID::MovieID::Rating::Timestamp"].str.split(delimiter,expand = True)
ratings.columns = ["UserID","MovieID","Rating","Timestamp"]


movies.drop(["Unnamed: 1","Unnamed: 2"],axis = 1,inplace=True)

delimiter =":："

movies = movies["Movie ID::Title::Genres"].str.split(delimiter,expand = True)
movies.columns = ["MovieID","Title","Genres"]

movies.shape,ratings.shape,users.shape

movies["Release_year"] = movies["Title"].str.extract('^(.+)\s\(([0-9]*)\)$',expand = True)[1]
movies["Title"] = movies["Title"].str.split("(").apply(lambda x:x[0])

from datetime import datetime
ratings["Watch_Hour"] =ratings["Timestamp"].apply(lambda x:datetime.fromtimestamp(int(x)).hour)
ratings.drop(["Timestamp"],axis = 1,inplace=True)

df = users.merge(movies.merge(ratings,on="MovieID",how="outer"),on="UserID",how="outer")
df["Genres"] = df["Genres"].str.split("|")
df = df.explode('Genres')

df["Genres"] = df["Genres"].replace({"":"Other","Horro":"Horror","Sci-":"Sci-Fi","Sci":"Sci-Fi","Sci-F":"Sci-Fi","Dr":"Drama","Documenta":"Documentary",
                     "Wester":"Western","Fant":"Fantasy","Chil":"Children's","R":"Romance","D":"Drama","Rom":"Romance","Animati":"Animation","Childr":"Children's","Childre":"Chi
                     "Fantas":"Fantasy","Come":"Comedy","Dram":"Drama","S":"Sci-Fi","Roma":"Romance","A":"Adventure","Children":"Children's","Adventu":"Adventure","Adv":"Adventu
                      "Wa":"War","Thrille"  :"Thriller","Com":"Comedy","Comed":"Comedy","Acti":"Action","Advent":"Adventure","Adventur":"Adventure","Thri":"Thriller",
                        "Chi":"Children's","Ro":"Romance","F":"Fantasy","We":"Western","Documen":"Documentary","Music":"Musical","Children":"Children's" ,"Horr":"Horror",
                     "Children'":"Children's","Roman":"Romance","Docu":"Documentary","Th":"Thriller","Document":"Documentary"
                    })
m = df.groupby(['MovieID','Genres'])['Title'].unique().str[0].unstack().reset_index().set_index('MovieID')
```

```
m = ~m.isna()
m = m.astype(int)
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Cosine Similarity :

Item and User : -Cosine similarity Matrix :

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
Item_similarity = cosine_similarity(user_movie_rating_matrix.T)
Item_similarity
```

```
array([[1.        , 0.07235746, 0.03701053, ..., 0.        , 0.12024178,
        0.02700277],
       [0.07235746, 1.        , 0.11528952, ..., 0.        , 0.        ,
        0.07780705],
       [0.03701053, 0.11528952, 1.        , ..., 0.        , 0.04752635,
        0.0632837 ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.04564448],
       [0.12024178, 0.        , 0.04752635, ..., 0.        , 1.        ,
        0.04433508],
       [0.02700277, 0.07780705, 0.0632837 , ..., 0.04564448, 0.04433508,
        1.        ]])
```

```
user_similarity.shape
```

```
(3633, 3633)
```

```
Item_similarty_matrix = pd.DataFrame(Item_similarity,
             index = user_movie_rating_matrix.columns,
             columns = user_movie_rating_matrix.columns)
Item_similarty_matrix
```

Title	$1,000,000 Duck	'Night Mother	'Til There Was You	'burbs, The	...And Justice for All	1-900	10 Things I Hate About You	101 Dalmatians	12 Angry Men	13th Warrior, The	187	2 Days in the Valley	20 Dates	20,000 Leagues Under the Sea	200 Cigarettes	2001: A Space Odyssey	
Title																	
$1,000,000 Duck	1.000000	0.072357	0.037011	0.079291	0.060838	0.000000	0.058619	0.189843	0.094785	0.058418	0.028171	0.021295	0.016918	0.141379	0.089850	0.068224	0.03
'Night Mother	0.072357	1.000000	0.115290	0.115545	0.159526	0.000000	0.076798	0.137135	0.111413	0.046135	0.060254	0.108613	0.038041	0.072367	0.139323	0.102570	0.07
'Til There Was You	0.037011	0.115290	1.000000	0.098756	0.066301	0.080250	0.127895	0.128523	0.079115	0.066598	0.019914	0.067742	0.091690	0.034371	0.197216	0.056820	0.03
'burbs, The	0.079291	0.115545	0.098756	1.000000	0.143620	0.000000	0.192191	0.250140	0.170719	0.197808	0.103273	0.183970	0.049312	0.186721	0.243211	0.201256	0.19
...And Justice for All	0.060838	0.159526	0.066301	0.143620	1.000000	0.000000	0.075093	0.178928	0.205486	0.122431	0.114231	0.195255	0.039933	0.171536	0.114865	0.219975	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Zed & Two Noughts, A	0.045280	0.091150	0.022594	0.055704	0.086080	0.000000	0.012702	0.042295	0.039344	0.041324	0.021497	0.083057	0.000000	0.063932	0.033813	0.094548	0.08
Zero Effect	0.039395	0.074787	0.079261	0.161174	0.110867	0.000000	0.175771	0.157313	0.133061	0.156505	0.097833	0.273992	0.063650	0.110569	0.120041	0.213441	0.16
Zero Kelvin	0.000000	0.000000	0.000000	0.000000	0.074317	0.000000	0.000000	0.033120	0.036867	0.034797	0.000000	0.041621	0.000000	0.035089	0.000000	0.033336	0.04
Zeus and Roxanne	0.120242	0.000000	0.047526	0.033567	0.000000	0.000000	0.058708	0.089840	0.058692	0.034623	0.000000	0.000000	0.031335	0.043226	0.074772	0.044879	0.03
eXistenZ	0.027003	0.077807	0.063284	0.110525	0.111040	0.039561	0.162060	0.120762	0.098731	0.230799	0.036650	0.165736	0.120666	0.116836	0.149607	0.246847	0.19

3633 rows × 3633 columns

```
Start coding or generate with AI.
```

User Based Similartiy :

```
Start coding or generate with AI.
```

```
User_similarity = cosine_similarity(user_movie_rating_matrix)
User_similarity.shape
```

```
(6040, 6040)
```

```
User_similarity
```

```
array([[1.        , 0.25531859, 0.12396703, ..., 0.15926709, 0.11935626,
        0.12239079],
       [0.25531859, 1.        , 0.25964457, ..., 0.16569953, 0.13332665,
        0.24845029],
       [0.12396703, 0.25964457, 1.        , ..., 0.20430203, 0.11352239,
        0.30693676],
       ...,
       [0.15926709, 0.16569953, 0.20430203, ..., 1.        , 0.18657496,
        0.18563871],
       [0.11935626, 0.13332665, 0.11352239, ..., 0.18657496, 1.        ,
        0.10827118],
       [0.12239079, 0.24845029, 0.30693676, ..., 0.18563871, 0.10827118,
        1.        ]])
```

```
User_similarity_matrix = pd.DataFrame(User_similarity,
             index = user_movie_rating_matrix.index,
             columns = user_movie_rating_matrix.index)
User_similarity_matrix
```

UserID	1	10	100	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	101	1010	1011	1012	1
UserID																		
1	1.000000	0.255319	0.123967	0.207800	0.139317	0.110320	0.121384	0.180226	0.103896	0.052816	0.060032	0.102675	0.049839	0.029990	0.159501	0.078094	0.080898	0.047
10	0.255319	1.000000	0.259645	0.280479	0.158703	0.112917	0.141985	0.432536	0.194915	0.102487	0.161729	0.220798	0.118062	0.205438	0.352071	0.199990	0.153074	0.156
100	0.123967	0.259645	1.000000	0.306067	0.075736	0.110450	0.358686	0.237492	0.172872	0.099147	0.060103	0.043367	0.061238	0.347549	0.264241	0.139793	0.085678	0.024
1000	0.207800	0.280479	0.306067	1.000000	0.099117	0.047677	0.201722	0.355920	0.325966	0.130702	0.042828	0.077724	0.123638	0.282423	0.249464	0.124237	0.117258	0.050
1001	0.139317	0.158703	0.075736	0.099117	1.000000	0.164854	0.053887	0.152057	0.138602	0.134710	0.019576	0.083651	0.200411	0.067900	0.254713	0.072502	0.037462	0.073
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	0.035731	0.146552	0.033754	0.044404	0.109700	0.072578	0.031406	0.088838	0.061450	0.032265	0.000000	0.041055	0.019928	0.029313	0.087898	0.104274	0.100905	0.131
996	0.170184	0.304806	0.344290	0.330748	0.222119	0.224779	0.185226	0.352014	0.287965	0.164045	0.078759	0.117937	0.151984	0.272145	0.471598	0.182751	0.130688	0.090
997	0.159267	0.165700	0.204302	0.172803	0.103255	0.068980	0.170771	0.175488	0.106303	0.049536	0.037536	0.041037	0.031871	0.089073	0.170519	0.070629	0.033974	0.025
998	0.119356	0.133327	0.113522	0.098456	0.269952	0.218905	0.141829	0.075538	0.112029	0.052900	0.012658	0.056094	0.169102	0.036257	0.230255	0.046755	0.078672	0.091
999	0.122391	0.248450	0.306937	0.250564	0.178399	0.178474	0.198656	0.334470	0.164777	0.143866	0.054761	0.057473	0.107457	0.338316	0.426651	0.118305	0.137676	0.023

6040 rows × 6040 columns

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

	Query	Candidate	cosine_similarity	query_tittle	candidate_tittle
0	1	10	0.0	Toy Story	GoldenEye
1	1	100	0.0	Toy Story	City Hall
2	1	1000	0.0	Toy Story	Curdled
5	1	1003	0.0	Toy Story	Extreme Measures
6	1	1004	0.0	Toy Story	Glimmer Man, The
8	1	1006	0.0	Toy Story	Chamber, The
10	1	1008	0.0	Toy Story	Davy Crockett, King of the Wild Frontier
31	1	1027	0.0	Toy Story	Robin Hood: Prince of Thieves
34	1	103	0.0	Toy Story	Unforgettable
39	1	1034	0.0	Toy Story	Freeway

```
Start coding or generate with AI.
```

```
m
```

Genres	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Other	Romance	Sci-Fi	Thriller	War	Western
MovieID																			
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
100	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
1000	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1001	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
994	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
996	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
997	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
998	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

3858 rows × 19 columns

Pearson Correlation

```
correlated_movie_matrix = m.T.corr()
```

```
correlated_movie_matrix
```

MovieID	1	10	100	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	101	1010	1011	
MovieID																	
1	1.000000	-0.187500	-0.148522	-0.102062	0.544331	0.544331	-0.148522	-0.148522	0.792118	-0.102062	0.604167	-0.102062	0.208333	0.544331	0.792118	0.604167	0.3
10	-0.187500	1.000000	0.321798	-0.102062	-0.102062	-0.102062	0.321798	0.792118	-0.148522	-0.102062	-0.187500	-0.102062	0.208333	-0.102062	-0.148522	0.208333	-0.1
100	-0.148522	0.321798	1.000000	-0.080845	-0.080845	-0.080845	1.000000	0.441176	-0.117647	0.687184	-0.148522	-0.080845	-0.148522	-0.080845	-0.117647	-0.148522	0.4
1000	-0.102062	-0.102062	-0.080845	1.000000	-0.055556	-0.055556	-0.080845	-0.080845	-0.080845	-0.055556	-0.102062	-0.055556	-0.102062	-0.055556	-0.080845	-0.102062	-0.0
1001	0.544331	-0.102062	-0.080845	-0.055556	1.000000	1.000000	-0.080845	-0.080845	0.687184	-0.055556	0.544331	-0.055556	-0.102062	1.000000	0.687184	0.544331	-0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
994	-0.102062	-0.102062	0.687184	-0.055556	-0.055556	-0.055556	0.687184	-0.080845	-0.080845	1.000000	-0.102062	-0.055556	-0.102062	-0.055556	-0.080845	-0.102062	0.6
996	-0.187500	0.208333	0.321798	-0.102062	-0.102062	-0.102062	0.321798	0.321798	-0.148522	0.544331	0.208333	0.544331	-0.187500	-0.102062	-0.148522	-0.187500	0.3
997	-0.148522	0.321798	1.000000	-0.080845	-0.080845	-0.080845	1.000000	0.441176	-0.117647	0.687184	-0.148522	-0.080845	-0.148522	-0.080845	-0.117647	-0.148522	0.4
998	-0.148522	0.321798	-0.117647	0.687184	-0.080845	-0.080845	-0.117647	0.441176	-0.117647	-0.080845	-0.148522	-0.080845	-0.148522	-0.080845	-0.117647	-0.148522	-0.1
999	-0.102062	-0.102062	-0.080845	1.000000	-0.055556	-0.055556	-0.080845	-0.080845	-0.080845	-0.055556	-0.102062	-0.055556	-0.102062	-0.055556	-0.080845	-0.102062	-0.0

3858 rows × 3858 columns

```
movies[movies.MovieID == "1"]["Title"][0]
```

```
'Toy Story '
```

```
movies[movies.Title.str.contains("Toy Story")].iloc[0].MovieID
```

```
'1'
```

```
def recommend_movie_based_on_correlation(movie):
    TITLE = movies[movies.Title.str.contains(movie)].iloc[0]["Title"]

    INDEX = movies[movies.Title.str.contains(movie)].iloc[0].MovieID

    print(TITLE)
    print(INDEX)

    print(movies[movies.MovieID.isin(correlated_movie_matrix[INDEX].sort_values(ascending=False).head(10).index.to_list())]["Title"])
```

```
recommend_movie_based_on_correlation("Toy Story")
```

```
Toy Story 
1
0                             Toy Story 
```

```
584                              Aladdin
1050    Aladdin and the King of Thieves
2009                   Jungle Book, The
2072                  American Tail, An
2285                 Rugrats Movie, The
2286                      Bug's Life, A
3045                        Toy Story 2
3542                     Saludos Amigos
3682                        Chicken Run
Name: Title, dtype: object
```

```
recommend_movie_based_on_correlation("Shawshank")
```

```
Shawshank Redemption, The
318
35             Dead Man Walking
384                   Boys Life
631                       Frisk
1555               Career Girls
2443    Ballad of Narayama, The
2451                    Airport
2452               Airport 1975
2453                Airport '77
3525               Center Stage
3529                     Hamlet
Name: Title, dtype: object
```

```
recommend_movie_based_on_correlation("Titanic")
```

```
Titanic
1721
200                  Total Eclipse
357         It Could Happen to You
1372                 Jerry Maguire
1466         Inventing the Abbotts
1951            Dangerous Liaisons
2106                       Déjà Vu
2247               Practical Magic
2850    Year of Living Dangerously
3086             Anna and the King
3599              Romeo and Juliet
Name: Title, dtype: object
```

```
recommend_movie_based_on_correlation("Braveheart")
```

```
Braveheart
110
461            Heaven & Earth
1204        Full Metal Jacket
1214                Boat, The
1222                    Glory
1959      Saving Private Ryan
2358       Thin Red Line, The
2993         Longest Day, The
3559            Flying Tigers
3574    Fighting Seabees, The
3585    Guns of Navarone, The
```

```
    Name: Title, dtype: object
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

k - Nearest Neighbours

```
from sklearn.neighbors import NearestNeighbors
```

```
kNN_model = NearestNeighbors(metric='cosine')
kNN_model.fit(user_movie_rating_matrix.T)
```

```
    NearestNeighbors(metric='cosine')
```

```
distances, indices = kNN_model.kneighbors(user_movie_rating_matrix.T, n_neighbors= 5)
```

```
result = pd.DataFrame(indices)
result
```

	0	1	2	3	4
0	0	731	414	285	582
1	1	803	72	2162	3029
2	2	1622	2524	3313	2583
3	3	1452	2164	1304	1043
4	4	26	723	890	493
...	...	...	...	...	...
3628	3628	2548	750	1582	2439
3629	3629	382	1699	482	1578
3630	3630	1328	1687	3393	2922
3631	3631	1609	1176	3225	2093
3632	3632	839	3126	2519	1278