Problem Statement

Create a Recommender System to show personalized movie recommendations based on ratings given by a user and other users similar to them in order to improve user experience.

Importing libraries

```
# Import Libraries
import numpy as np
import pandas as pd
from datetime import datetime
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

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

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

```
# Formatting the data files to bring them into a workable format
```

```
users_data = "/content/drive/MyDrive/Scaler Case studies/Zee RS/zee-users.dat"
users = pd.read_csv(users_data,sep='\::')
```

```
users.shape
```

```
(6040, 5)
```

```
ratings_data = "/content/drive/MyDrive/Scaler Case studies/Zee RS/zee-ratings.dat"
ratings = pd.read_csv(ratings_data,sep='\::')
```

```
ratings.shape
```

```
(1000209, 4)
```

```
movies_data = "/content/drive/MyDrive/Scaler Case studies/Zee RS/zee-movies.dat"
movies = pd.read_csv(movies_data,encoding="ISO-8859-1",sep='\::')
```

```
movies.shape
```

```
(3883, 3)
```

Data Pre-processing

```
# Understanding users and performing necessary data type conversions as suggested
```

```
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

```
users1 = users.copy()
```

```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   UserID      6040 non-null   int64 
 1   Gender      6040 non-null   object
 2   Age         6040 non-null   int64 
 3   Occupation  6040 non-null   int64 
 4   Zip-code    6040 non-null   object
dtypes: int64(3), object(2)
memory usage: 236.1+ KB
```

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

```
M    4331
F    1709
Name: Gender, dtype: int64
```

```
users['Age'] = users['Age'].astype(str)
```

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

```
users.head()
```

	UserID	Gender	Age	Occupation	Zip-code
0	1	F	Under 18	10	48067
1	2	M	56+	16	70072

```
users['Occupation'] = users['Occupation'].astype(str)
```

```
users.replace({"Occupation" : {'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"}} , inplace = True)
```

```
users.head()
```

	UserID	Gender	Age	Occupation	Zip-code
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

```
users.shape
```

```
(6040, 5)
```

```
users.duplicated().sum()
```

```
0
```

```
users.isnull().sum()
```

```
UserID        0
Gender        0
Age           0
Occupation    0
```

```
Zip-code      0
dtype: int64
```

```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   UserID      6040 non-null   int64 
 1   Gender      6040 non-null   object
 2   Age         6040 non-null   object
 3   Occupation  6040 non-null   object
 4   Zip-code    6040 non-null   object
dtypes: int64(1), object(4)
memory usage: 236.1+ KB
```

```
# Understanding movies and performing necessary data type conversions as suggested
```

```
movies.shape
```

```
(3883, 3)
```

```
movies.duplicated().sum()
```

```
0
```

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Movie ID  3883 non-null   int64 
 1   Title     3883 non-null   object
 2   Genres    3883 non-null   object
dtypes: int64(1), object(2)
memory usage: 91.1+ KB
```

```
movies.isnull().sum()
```

```
Movie ID    0
Title       0
Genres      0
dtype: int64
```

```
# Understanding ratings and performing necessary data type conversions as suggested
```

```
ratings.shape
```

```
(1000209, 4)
```

```
ratings.duplicated().sum()
```

```
0
```

```
ratings.isnull().sum()
```

```
UserID       0
MovieID      0
Rating       0
Timestamp    0
dtype: int64
```

```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 4 columns):
 #   Column     Non-Null Count    Dtype
---  ------     --------------    -----
 0   UserID     1000209 non-null  int64
 1   MovieID    1000209 non-null  int64
 2   Rating     1000209 non-null  int64
 3   Timestamp  1000209 non-null  int64
dtypes: int64(4)
memory usage: 30.5 MB
```

```
ratings['Rating'].value_counts()
```

```
4    348971
3    261197
5    226310
2    107557
1     56174
Name: Rating, dtype: int64
```

```
ratings['Rating'] = ratings['Rating'].astype(str)
```

```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 4 columns):
 #   Column     Non-Null Count    Dtype
---  ------     --------------    -----
 0   UserID     1000209 non-null  int64
 1   MovieID    1000209 non-null  int64
 2   Rating     1000209 non-null  object
 3   Timestamp  1000209 non-null  int64
dtypes: int64(3), object(1)
memory usage: 30.5+ MB
```

Feature Engineering

```
movies["Release_year"] = movies["Title"].str.findall('\((\d{4})\)').str.get(0)
```

```
movies["Release_Decade"] = (((movies["Release_year"].astype("int64") % 100)//10)*10).astype("objec
```

```
movies.head()
```

	Movie ID	Title	Genres	Release_year	Release_Decade
0	1	Toy Story (1995)	Animation|Children's|Comedy	1995	90
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	1995	90
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	90
3	4	Waiting to Exhale (1995)	Comedy|Drama	1995	90

```
movies.columns
```

```
Index(['Movie ID', 'Title', 'Genres', 'Release_year', 'Release_Decade'], dtype='object')
```

```
users.columns
```

```
Index(['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code'], dtype='object')
```

```
movies.rename(columns = {'Movie ID': 'MovieID'}, inplace=True)
```

Merging Dataframe

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

```
len(df)
```

```
1000209
```

```
df.duplicated(subset = ["UserID" , "MovieID"]).sum()
```

```
0
```

```
df.isnull().sum()
```

```
UserID            0
MovieID           0
Rating            0
Timestamp         0
Title             0
Genres            0
Release_year      0
Release_Decade    0
Gender            0
Age               0
Occupation        0
Zip-code          0
dtype: int64
```

```
df.shape
```

```
(1000209, 12)
```

```
df.dtypes
```

```
UserID             int64
MovieID            int64
Rating            object
Timestamp          int64
Title             object
Genres            object
Release_year      object
Release_Decade    object
Gender            object
Age               object
Occupation        object
Zip-code          object
dtype: object
```

```
df['Zip-code'].value_counts()
```

```
94110    3802
60640    3430
98103    3204
95616    3079
02138    3019
         ...
33547      20
46556      20
46350      20
21015      20
48146      20
Name: Zip-code, Length: 3439, dtype: int64
```

```
#Categorical variables and numerical variables
numeric_df = df.select_dtypes(include=[np.number])
categorical_df = df.select_dtypes(exclude=[np.number])
```

EDA

Univariate analysis

```
from matplotlib import rcParams
rcParams['figure.figsize'] = 10, 10
import seaborn as sns
import matplotlib.pyplot as plt
```

For categorical variable(s): countplot, piechart, barplot

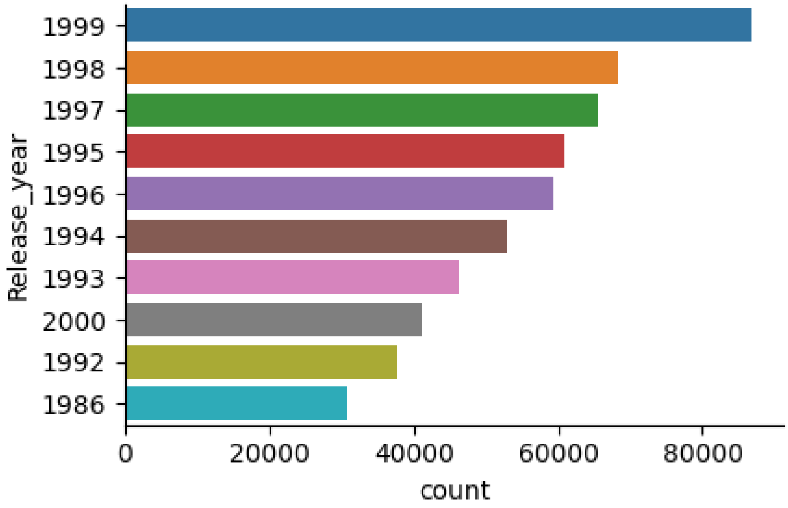
```
# categorical variales
categorical_df.columns
```

```
Index(['Rating', 'Title', 'Genres', 'Release_year', 'Release_Decade', 'Gender',
       'Age', 'Occupation', 'Zip-code'],
      dtype='object')
```

Barplot with counts

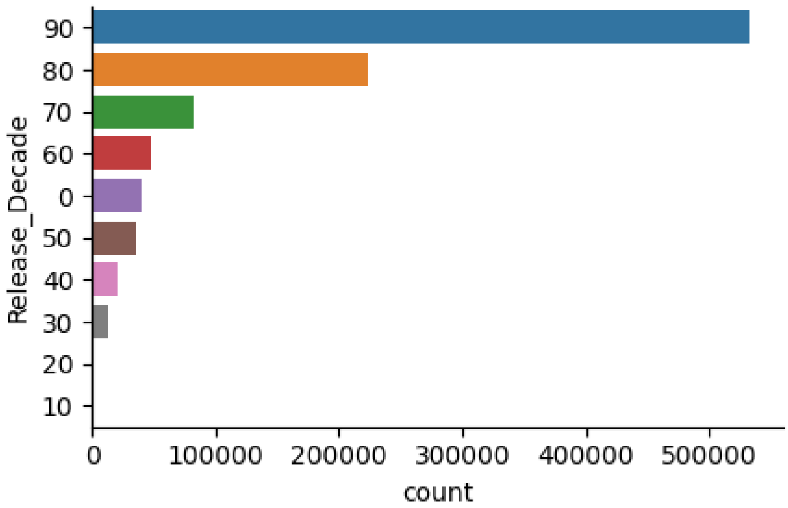
```
sns.catplot(y='Release_year', kind='count', height=3, aspect=1.5, order = categorical_df['Release_
```

```
<seaborn.axisgrid.FacetGrid at 0x79203d663010>
```



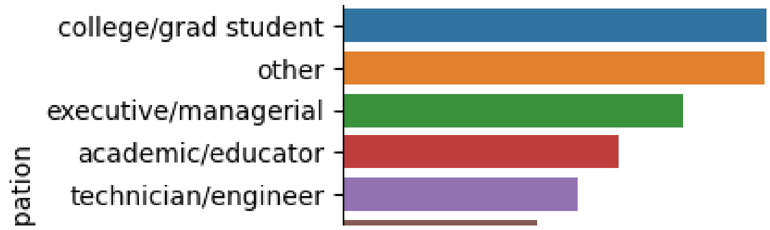

```
sns.catplot(y='Release_Decade', kind='count', height=3, aspect=1.5, order = categorical_df['Releas
```

```
<seaborn.axisgrid.FacetGrid at 0x79203d7837f0>
```



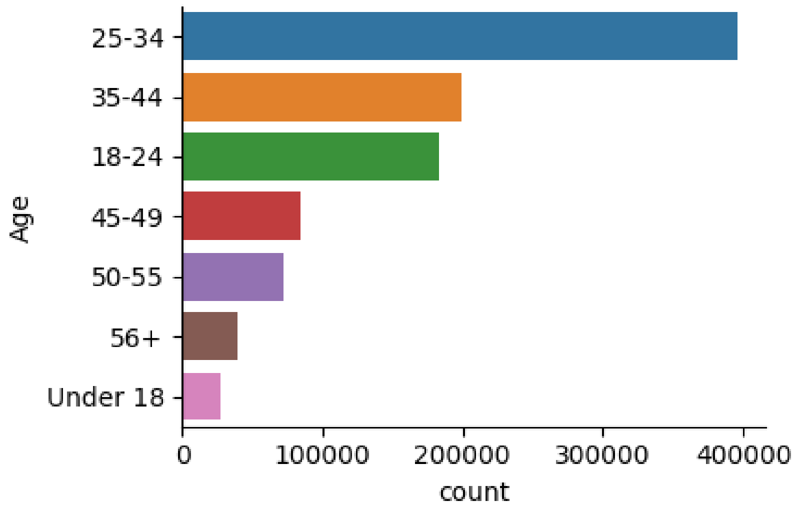

```
sns.catplot(y='Occupation', kind='count', height=3, aspect=1.5, order = categorical_df['Occupation
```

```
<seaborn.axisgrid.FacetGrid at 0x792036d4a530>
```



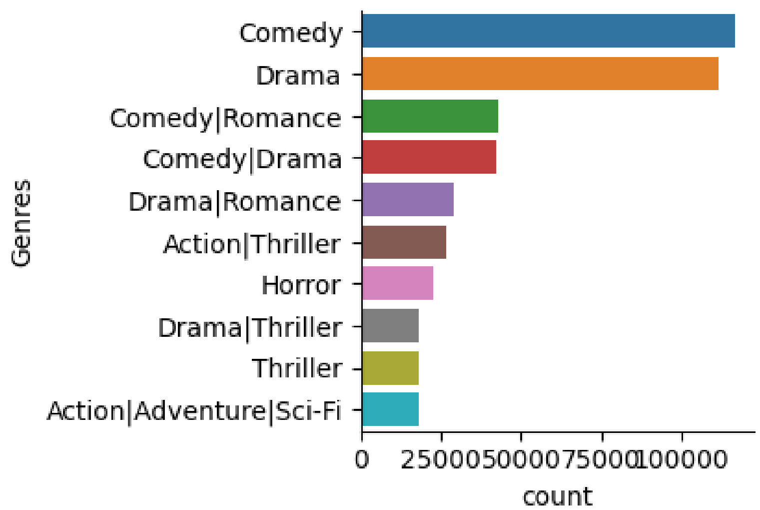

```
sns.catplot(y='Age', kind='count', height=3, aspect=1.5, order = categorical_df['Age'].value_count
```

```
<seaborn.axisgrid.FacetGrid at 0x7920370f83a0>
```



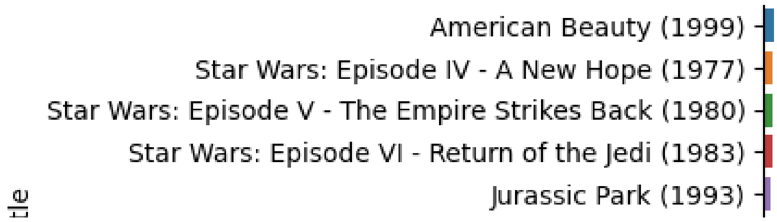

```
sns.catplot(y='Genres', kind='count', height=3, aspect=1.5, order = categorical_df['Genres'].value
```

```
<seaborn.axisgrid.FacetGrid at 0x792060851150>
```




```
sns.catplot(y='Title', kind='count', height=3, aspect=1.5, order = categorical_df['Title'].value_c
```

```
<seaborn.axisgrid.FacetGrid at 0x792036d677c0>
```

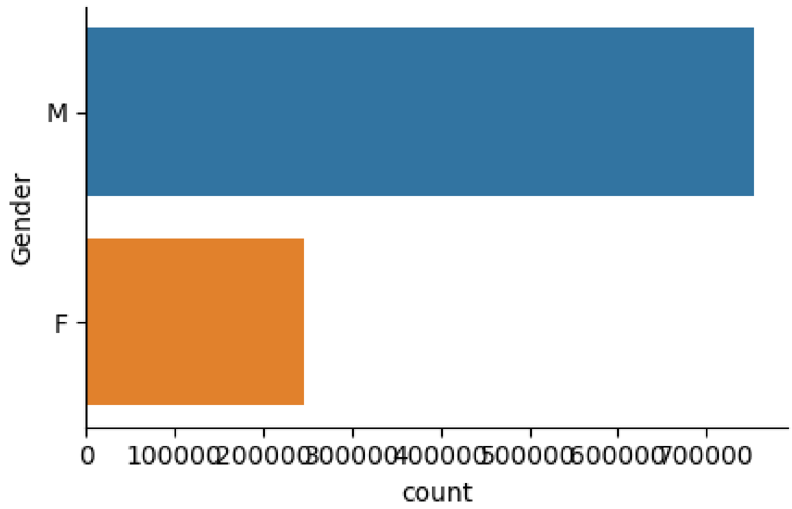



```
categorical_df['Title'].value_counts()
```

```
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
                                                         ...
Waiting Game, The (2000)                                    1
Shadows (Cienie) (1988)                                     1
Juno and Paycock (1930)                                     1
Resurrection Man (1998)                                     1
Windows (1980)                                              1
Name: Title, Length: 3706, dtype: int64
```

```
sns.catplot(y='Gender', kind='count', height=3, aspect=1.5, order = categorical_df['Gender'].value
```

```
<seaborn.axisgrid.FacetGrid at 0x7920369827a0>
```



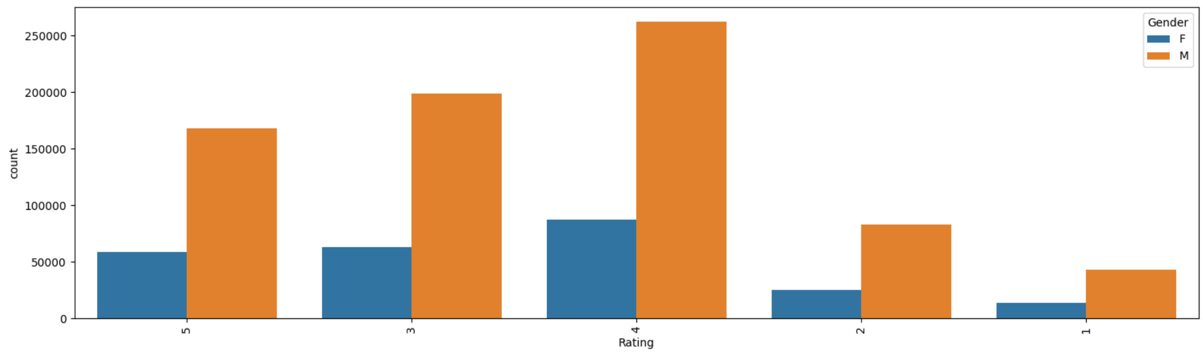

```
sns.catplot(y='Rating', kind='count', height=3, aspect=1.5, order = categorical_df['Rating'].value
```

```
<seaborn.axisgrid.FacetGrid at 0x79203684bca0>
```

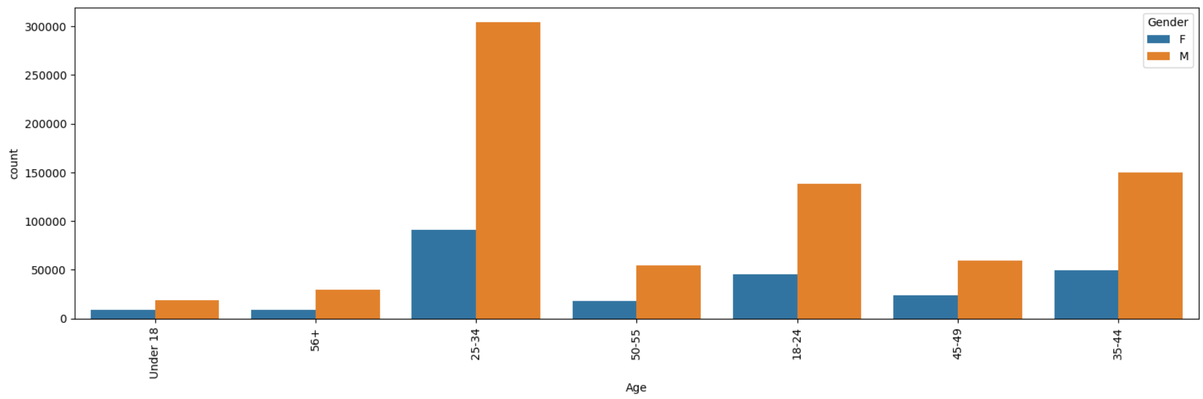
Bivariate analysis

```
## Categorical variables relationship with respect to Gender and Age
```

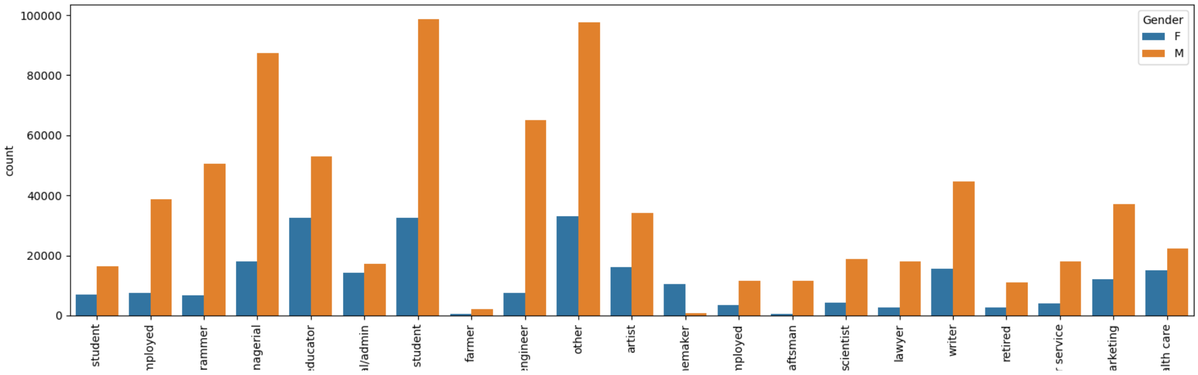
```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Rating" , hue = "Gender" )
plt.xticks(rotation = "vertical")
plt.show()
```



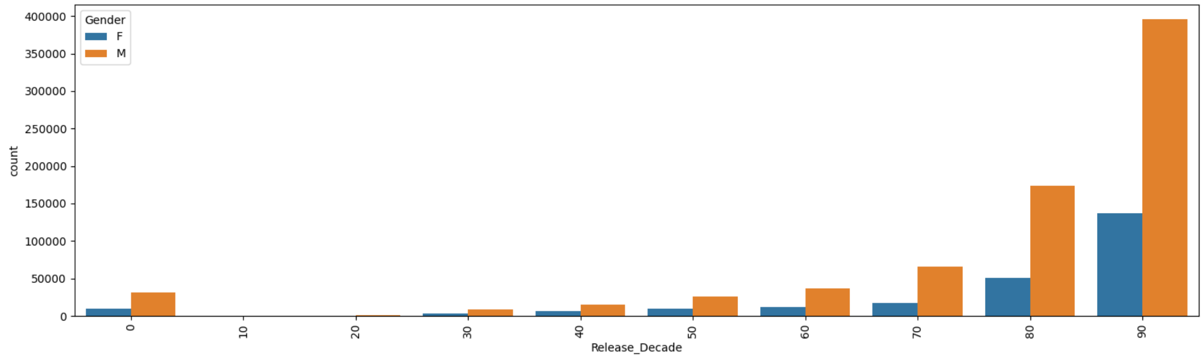

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Age" , hue = "Gender" )
plt.xticks(rotation = "vertical")
plt.show()
```



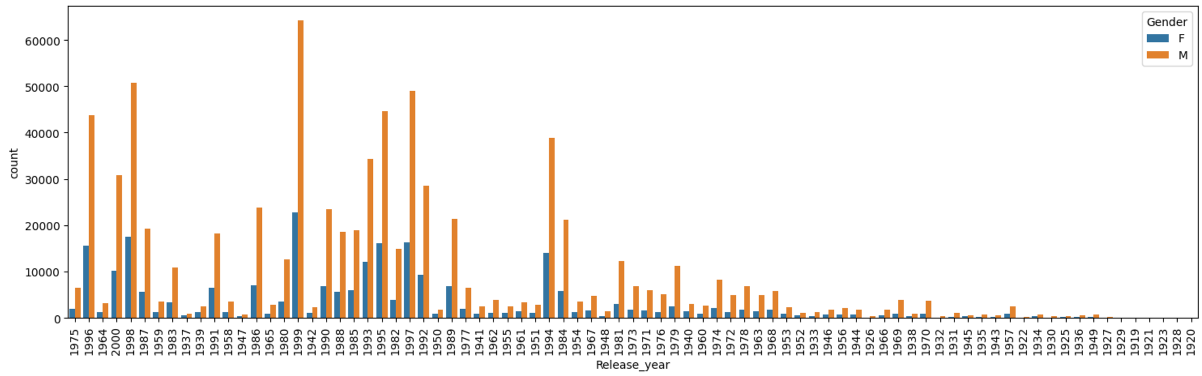

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Occupation" , hue = "Gender" )
plt.xticks(rotation = "vertical")
plt.show()
```



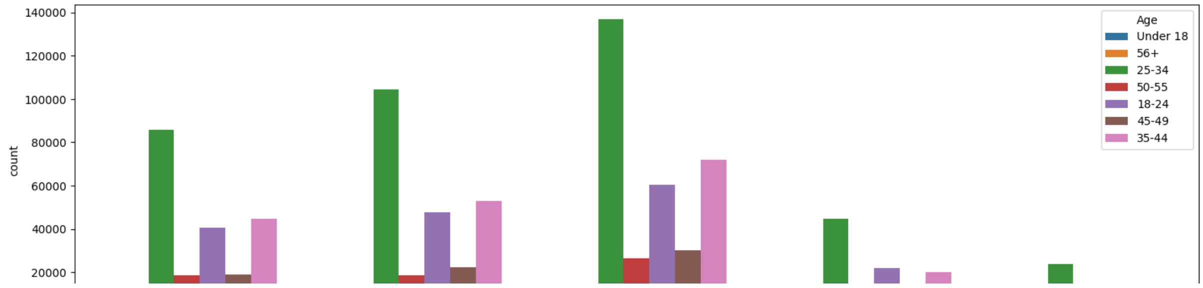

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Release_Decade" , hue = "Gender" )
plt.xticks(rotation = "vertical")
plt.show()
```



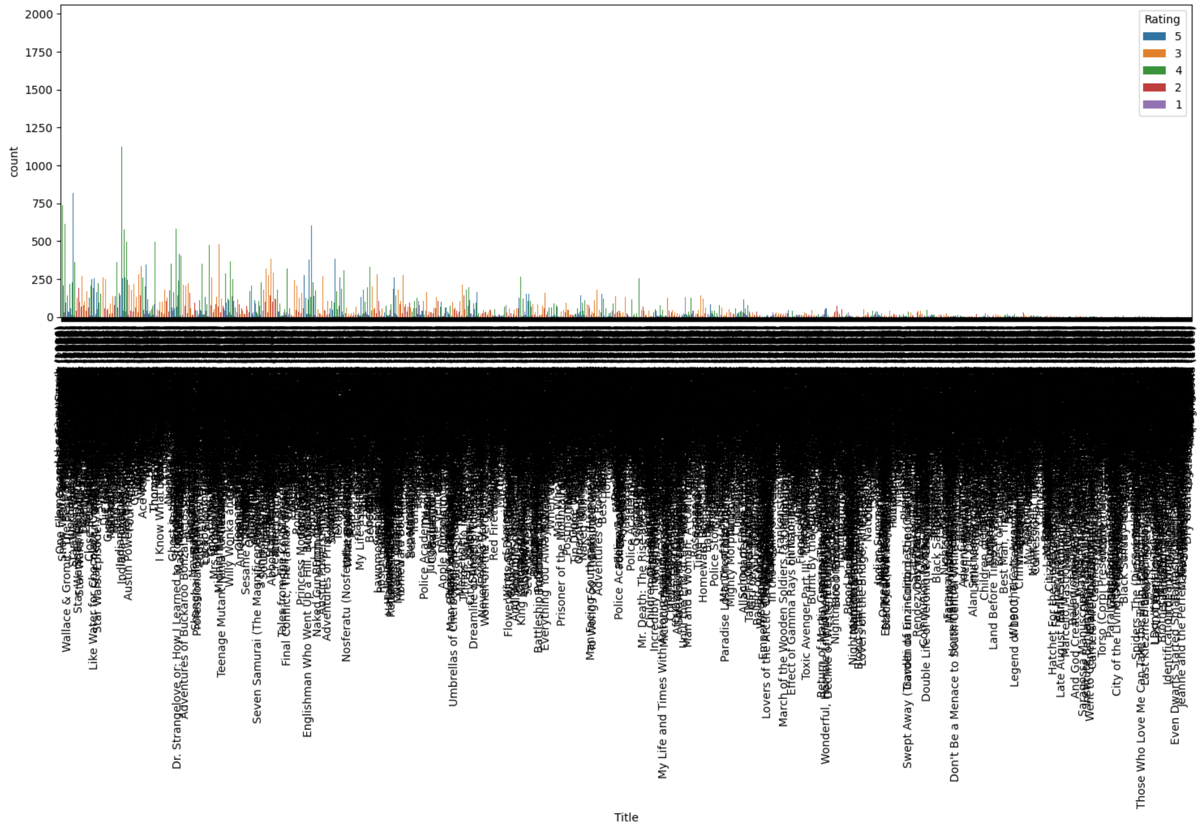

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Release_year" , hue = "Gender" )
plt.xticks(rotation = "vertical")
plt.show()
```




```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Rating" , hue = "Age" )
plt.xticks(rotation = "vertical")
plt.show()
```

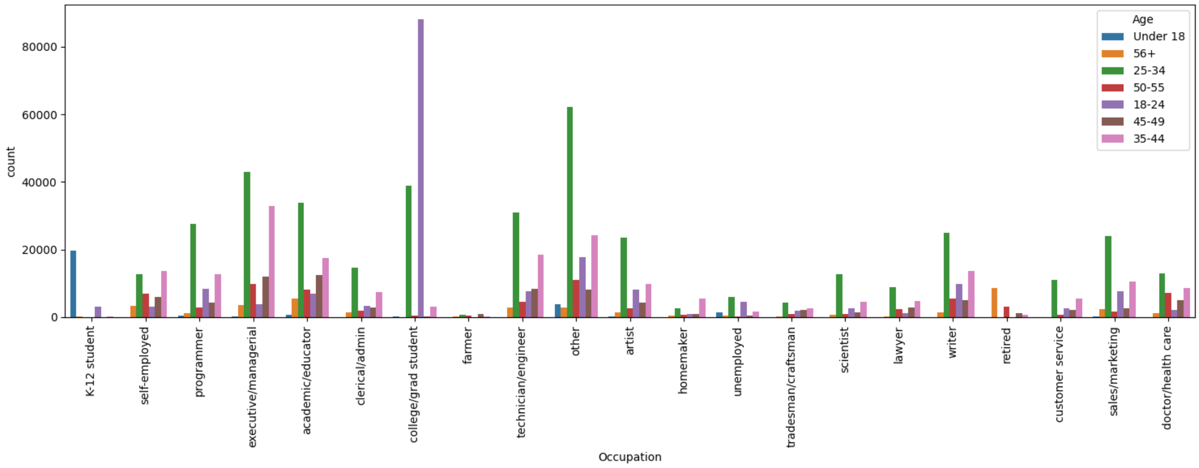



```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Title" , hue = "Rating" )
plt.xticks(rotation = "vertical")
plt.show()
```

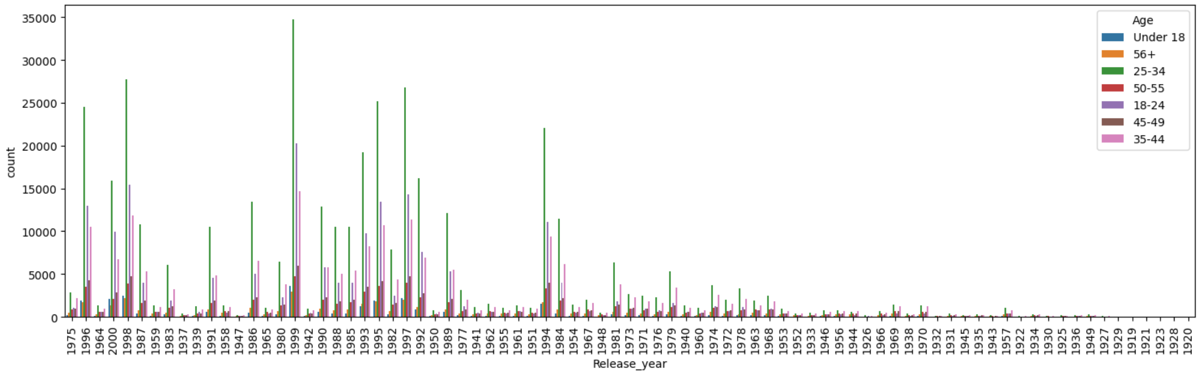



```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Release_Decade" , hue = "Age" )
plt.xticks(rotation = "vertical")
plt.show()
```


```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Occupation" , hue = "Age" )
plt.xticks(rotation = "vertical")
plt.show()
```




```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Release_year" , hue = "Age" )
plt.xticks(rotation = "vertical")
plt.show()
```



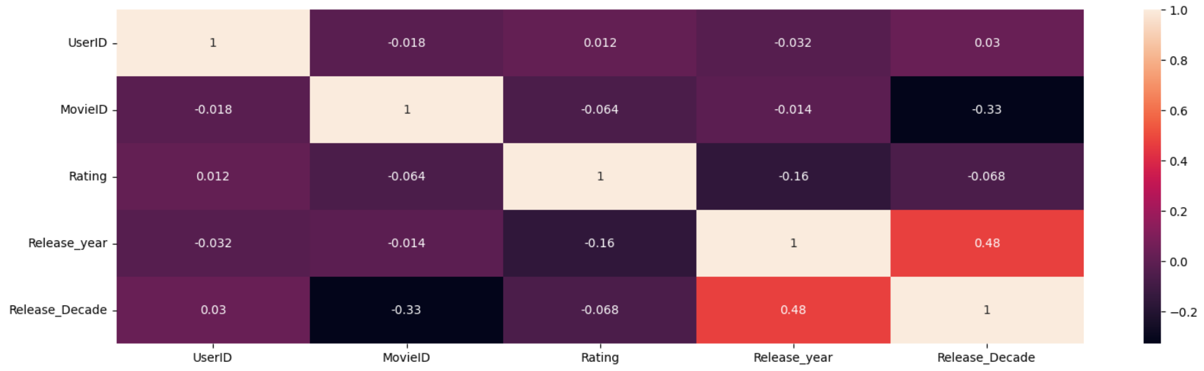
Heatmap - Correlation analysis

```
df["Timestamp"] = pd.to_datetime(df['Timestamp'], unit='s')
df["Release_year"] = df["Release_year"].astype("int64")
df["Rating"] = df["Rating"].astype("int64")
df["Release_Decade"] = df["Release_Decade"].astype("int64")
```

```
df.dtypes
```

```
UserID                    int64
MovieID                   int64
Rating                    int64
Timestamp        datetime64[ns]
Title                    object
Genres                   object
Release_year              int64
Release_Decade            int64
Gender                   object
Age                      object
Occupation               object
Zip-code                 object
dtype: object
```

```
plt.figure(figsize = (18,5))
sns.heatmap(data = df.corr() , annot=True)
plt.show()
```



Genres Preprocessing

```
movies.Genres.value_counts()
```

```
Drama                                  843
Comedy                                 521
Horror                                 178
Comedy|Drama                           162
Comedy|Romance                         142
                                      ...
Action|Comedy|Crime|Horror|Thriller      1
Action|Drama|Thriller|War                1
Action|Adventure|Children's              1
Action|Adventure|Children's|Fantasy      1
Adventure|Crime|Sci-Fi|Thriller          1
Name: Genres, Length: 301, dtype: int64
```

```
dfmov = movies.copy()
dfmov.dropna(inplace=True)
dfmov.Genres = dfmov.Genres.str.split('|')
dfmov['Genres'] = dfmov['Genres'].apply(lambda x: [i for i in x if i!='A' and i!='D' and i!= 'F' a
for i in dfmov['Genres']:
    for j in range(len(i)):
        if i[j] == 'Ro' or i[j] == 'Rom' or i[j] == 'Roman' or i[j] == 'R' or i[j] == 'Roma':
            i[j] = 'Romance'
        elif i[j] == 'Chil' or i[j] == 'Childre' or i[j] == 'Childr' or i[j] == "Children'" or i[j
```

```
            i[j] = "Children's"
        elif i[j] == 'Fantas' or i[j] == 'Fant':
            i[j] = 'Fantasy'
        elif i[j] == 'Dr' or i[j] == 'Dram':
            i[j] = 'Drama'
        elif i[j] == 'Documenta'or i[j] == 'Docu' or i[j] == 'Document' or i[j] == 'Documen':
            i[j] = 'Documentary'
        elif i[j] == 'Wester'or i[j] == 'We':
            i[j] = 'Western'
        elif i[j] == 'Animati':
            i[j] = 'Animation'
        elif i[j] == 'Come'or i[j] == 'Comed' or i[j] == 'Com':
            i[j] = 'Comedy'
        elif i[j] == 'Sci-F'or i[j] == 'S' or i[j] == 'Sci-' or i[j] == 'Sci':
            i[j] = 'Sci-Fi'
        elif i[j] == 'Adv'or i[j] == 'Adventu' or i[j] == 'Adventur' or i[j] == 'Advent':
            i[j] = 'Adventure'
        elif i[j] == 'Horro'or i[j] == 'Horr':
            i[j] = 'Horror'
        elif i[j] == 'Th'or i[j] == 'Thri' or i[j] == 'Thrille':
            i[j] = 'Thriller'
        elif i[j] == 'Acti':
            i[j] = 'Action'
        elif i[j] == 'Wa':
            i[j] = 'War'
        elif i[j] == 'Music':
            i[j] = 'Musical'
```

```
dfmov.head()
```

	MovieID	Title	Genres	Release_year	Release_Decade
0	1	Toy Story (1995)	[Animation, Children's, Comedy]	1995	90
1	2	Jumanji (1995)	[Adventure, Children's, Fantasy]	1995	90
2	3	Grumpier Old Men (1995)	[Comedy, Romance]	1995	90
3	4	Waiting to Exhale (1995)	[Comedy, Drama]	1995	90
4	5	Father of the Bride	[Comedy]	1995	90

```
movies.head()
```

	MovieID	Title	Genres	Release_year	Release_Decade
0	1	Toy Story (1995)	Animation|Children's|Comedy	1995	90
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	1995	90
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	90
		Waiting to			

```
genres_df = pd.get_dummies(dfmov['Genres'].apply(pd.Series).stack()).sum(level=0)
genres_df.head()
```

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fanta:
0	0	0	1	1	1	0	0	0	
1	0	1	0	1	0	0	0	0	
2	0	0	0	0	1	0	0	0	
3	0	0	0	0	1	0	0	1	
4	0	0	0	0	1	0	0	0	

```
test = genres_df.iloc[:,0:].sum()
test=test.iloc[1:]
print(test)
```

```
Adventure       283
Animation       105
Children's      251
Comedy         1200
Crime           211
Documentary     127
Drama          1603
Fantasy          68
Film-Noir        44
Horror          343
Musical         114
Mystery         106
Romance         471
Sci-Fi          276
Thriller        492
War             143
Western          68
dtype: int64
```

```
len(test)
```

```
17
```

```
print(type(pd.to_numeric(test)))
print(type(test.to_numpy().reshape(17,)[0]))
# genre_sum = np.hstack((np.asarray(genre_list).reshape(18,1), test.to_numpy().reshape(18,)))
# genre_sum[:,1] = genre_sum[:,1].astype('int64')
test2 = test.to_numpy().reshape(17,)
```

```
<class 'pandas.core.series.Series'>
<class 'numpy.int64'>
```

```
test2
```

```
array([ 283,  105,  251, 1200,  211,  127, 1603,   68,   44,  343,  114,
        106,  471,  276,  492,  143,   68])
```

```
movies.isnull().sum()
```

```
MovieID           0
Title             0
Genres            0
Release_year      0
Release_Decade    0
dtype: int64
```

```
df_new = ratings.merge(movies , on = "MovieID").merge(users , on = "UserID")
```

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

Build a Recommender System based on Pearson Correlation

User-Interaction Matrix

```
matrix = pd.pivot_table(df, index='UserID', columns='Title', values='Rating', aggfunc='mean')
```

```
matrix.fillna(0, inplace=True) # Imputing 'NaN' values with Zero rating

print(matrix.shape)

matrix.head(10)
```

```
(6040, 3706)
```

Title	$1,000,000 Duck (1971)	'Night Mother (1986)	'Til There Was You (1997)	'burbs, The (1989)	...And Justice for All (1979)	1-900 (1994)	10 Things I Hate About You (1999)	101 Dalmatians (1961)	Dal
UserID									
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	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	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9	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	

10 rows × 3706 columns

```
n_users = df['UserID'].nunique()
n_movies = df['MovieID'].nunique()
sparsity = round(1.0 - df.shape[0] / float( n_users * n_movies), 3)
print('The sparsity level of dataset is ' +  str(sparsity * 100) + '%')
```

```
The sparsity level of dataset is 95.5%
```

Pearson Correlation

Item based approach

```
df[df['Title']=='Jumanji (1995)']
```

	UserID	MovieID	Rating	Timestamp	Title	Genres	Releas
758	18	2	2	978152541	Jumanji (1995)	Adventure|Children's|Fantasy	
2197	44	2	4	1004410663	Jumanji (1995)	Adventure|Children's|Fantasy	
2567	48	2	3	978064964	Jumanji (1995)	Adventure|Children's|Fantasy	
3281	53	2	5	977981548	Jumanji (1995)	Adventure|Children's|Fantasy	
4571	62	2	4	977904756	Jumanji (1995)	Adventure|Children's|Fantasy	
...	...	...	...	...	...	...	
986720	4242	2	4	965312117	Jumanji (1995)	Adventure|Children's|Fantasy	
987439	2152	2	3	974620151	Jumanji (1995)	Adventure|Children's|Fantasy	
991246	4020	2	3	965524602	Jumanji (1995)	Adventure|Children's|Fantasy	
995173	6019	2	4	956761170	Jumanji (1995)	Adventure|Children's|Fantasy	
995895	1529	2	3	974744877	Jumanji (1995)	Adventure|Children's|Fantasy	

701 rows × 12 columns

```
movies_name = 'Jumanji (1995)'
movie_rating = matrix[movies_name]
print(movie_rating)
```

```
UserID
1       0.0
2       0.0
```

```
3       0.0
4       0.0
5       0.0
       ...
6036    0.0
6037    0.0
6038    0.0
6039    0.0
6040    0.0
Name: Jumanji (1995), Length: 6040, dtype: float64
```

```
sim_movies = matrix.corrwith(movie_rating)
```

```
sim_df = pd.DataFrame(sim_movies, columns=['Cor-relation'])
sim_df.sort_values('Cor-relation', ascending=False, inplace=True)
sim_df.iloc[1: , :].head(10)
```

	Cor-relation
Title	
Hook (1991)	0.520853
Dragonheart (1996)	0.446999
Indian in the Cupboard, The (1995)	0.439029
Santa Clause, The (1994)	0.416383
NeverEnding Story, The (1984)	0.414332
Honey, I Shrunk the Kids (1989)	0.402690
Space Jam (1996)	0.396840
Teenage Mutant Ninja Turtles (1990)	0.392595
Willow (1988)	0.389206
Mask, The (1994)	0.388248

Cosine Similarity

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

```
item_similarity = cosine_similarity(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.        ]])
```

```
item_similarity.shape
```

```
(3706, 3706)
```

Item-based similarity

```
item_similarity_matrix = pd.DataFrame(item_similarity, index=matrix.columns, columns=matrix.column
item_similarity_matrix.head()
```

Title	$1,000,000 Duck (1971)	'Night Mother (1986)	'Til There Was You (1997)	'burbs, The (1989)	...And Justice for All (1979)	1-900 (1994)	10 Things I Hate About You (1999)	Dalmat (1
Title								
$1,000,000 Duck (1971)	1.000000	0.072357	0.037011	0.079291	0.060838	0.00000	0.058619	0.18
'Night Mother (1986)	0.072357	1.000000	0.115290	0.115545	0.159526	0.00000	0.076798	0.14
'Til There Was You (1997)	0.037011	0.115290	1.000000	0.098756	0.066301	0.08025	0.127895	0.11
'burbs, The (1989)	0.079291	0.115545	0.098756	1.000000	0.143620	0.00000	0.192191	0.24
...And Justice for All (1979)	0.060838	0.159526	0.066301	0.143620	1.000000	0.00000	0.075093	0.19

5 rows × 3706 columns

User-based similarity

```
user_similarity = cosine_similarity(matrix)
user_similarity
```

```
array([[1.        , 0.09638153, 0.12060981, ..., 0.        , 0.17460369,
        0.13359025],
       [0.09638153, 1.        , 0.1514786 , ..., 0.06611767, 0.0664575 ,
        0.21827563],
       [0.12060981, 0.1514786 , 1.        , ..., 0.12023352, 0.09467506,
        0.13314404],
       ...,
       [0.        , 0.06611767, 0.12023352, ..., 1.        , 0.16171426,
```

```
         0.09930008],
        [0.17460369, 0.0664575 , 0.09467506, ..., 0.16171426, 1.        ,
         0.22833237],
        [0.13359025, 0.21827563, 0.13314404, ..., 0.09930008, 0.22833237,
         1.        ]])
```

```
user_similarity_matrix = pd.DataFrame(user_similarity, index=matrix.index, columns=matrix.index)
user_similarity_matrix.head()
```

UserID	1	2	3	4	5	6	7	8
UserID								
1	1.000000	0.096382	0.120610	0.132455	0.090158	0.179222	0.059678	0.138241
2	0.096382	1.000000	0.151479	0.171176	0.114394	0.100865	0.305787	0.203337
3	0.120610	0.151479	1.000000	0.151227	0.062907	0.074603	0.138332	0.077656
4	0.132455	0.171176	0.151227	1.000000	0.045094	0.013529	0.130339	0.100856
5	0.090158	0.114394	0.062907	0.045094	1.000000	0.047449	0.126257	0.220817

5 rows × 6040 columns

Matrix Factorization

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

```
df.head()
```

	UserID	MovieID	Rating	Timestamp	Title	Genres	Release_y
0	1	1193	5	978300760	One Flew Over the Cuckoo's Nest (1975)	Drama	1
1	1	661	3	978302109	James and the Giant Peach (1996)	Animation|Children's|Musical	1
2	1	914	3	978301968	My Fair Lady (1964)	Musical|Romance	1
3	1	3408	4	978300275	Erin Brockovich (2000)	Drama	2
4	1	2355	5	978824291	Bug's Life, A (1998)	Animation|Children's|Comedy	1

```
rating_movie = df.pivot(index = 'UserID', columns ='MovieID', values = 'Rating').fillna(0)
rating_movie.head()
```

MovieID	1	2	3	4	5	6	7	8	9	10	...	3943	3944	3945	3946	3947	3948	3949
UserID																		
1	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	0	0	0	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	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
5	0	0	0	0	0	2	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 3706 columns

```
rating_movie = rating_movie.astype(int)
```

```
rating_movie.dtypes
```

```
MovieID
1       int64
2       int64
3       int64
4       int64
5       int64
        ...
3948    int64
3949    int64
3950    int64
3951    int64
3952    int64
Length: 3706, dtype: object
```

CMFREC library

```
!pip install cmfrec
```

```
Collecting cmfrec
  Downloading cmfrec-3.5.1.post6.tar.gz (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.4/268.4 kB 3.7 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: cython in /usr/local/lib/python3.10/dist-packages (from cmfre
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from cmfrec
Requirement already satisfied: pandas>=0.25.0 in /usr/local/lib/python3.10/dist-packages (fr
Collecting findblas (from cmfrec)
  Using cached findblas-0.1.21-py3-none-any.whl
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pyt
Building wheels for collected packages: cmfrec
  Building wheel for cmfrec (pyproject.toml) ... done
  Created wheel for cmfrec: filename=cmfrec-3.5.1.post6-cp310-cp310-linux_x86_64.whl size=58
```

```
  Stored in directory: /root/.cache/pip/wheels/7d/ef/82/425eeca860fd20527d2a9159fc5920b59114
Successfully built cmfrec
Installing collected packages: findblas, cmfrec
Successfully installed cmfrec-3.5.1.post6 findblas-0.1.21
```

```
from cmfrec import CMF
```

```
user_item = df_new[['UserID', 'MovieID', 'Rating']].copy()
user_item.head(2)
```

	UserID	MovieID	Rating
0	1	1193	5
1	1	661	3

```
user_item.columns = ['UserId', 'ItemId', 'Rating']
```

```
print(user_item.shape)
print("No.of Users:",len(user_item['UserId'].unique()))
print("No.of Items:",len(user_item['ItemId'].unique()))
```

```
(1000209, 3)
No.of Users: 6040
No.of Items: 3706
```

```
model = CMF(method="als", k=4, lambda_=0.1, user_bias=False, item_bias=False, verbose=False)
```

```
model.fit(user_item)
```

```
Collective matrix factorization model
(explicit-feedback variant)
```

```
model.A_.shape
```

```
(6040, 4)
```

```
model.B_.shape
```

```
(3706, 4)
```

```
user_item['Rating'].mean()
```

```
inf
```

```
model.glob_mean_
```

```
3.581564426422119
```

```
rm_ = np.dot(model.A_, model.B_.T) + model.glob_mean_
rm_
```

```
array([[4.070904 , 3.98917  , 4.2135158, ..., 3.5903254, 3.6889405,
        3.6445234],
       [4.19682  , 2.9005492, 4.716052 , ..., 3.222189 , 3.2738314,
        4.4364557],
       [4.396451 , 2.7396243, 3.0979629, ..., 3.5120726, 3.4605935,
        2.6689832],
       ...,
       [4.612237 , 3.8701787, 3.9580421, ..., 3.5483565, 3.192472 ,
        4.444893 ],
       [3.8280816, 2.8445845, 3.2666838, ..., 3.4400403, 3.2254841,
        3.79869  ],
       [3.715859 , 2.4141521, 3.4052198, ..., 3.3756154, 3.5127134,
        2.9444382]], dtype=float32)
```

```
top_items = model.topN(user=100, n=10) # Acoording to matrix factorization recommendation user no.
movies.loc[movies.MovieID.isin(top_items)]
```

	MovieID	Title	Genres	Release_year	Release_Decade
52	53	Lamerica (1994)	Drama	1994	90
820	831	Stonewall (1995)	Drama	1995	90
1649	1696	Bent (1997)	Drama|War	1997	90
2128	2197	Firelight (1997)	Drama	1997	90
2691	2760	Gambler, The (A Játékos) (1997)	Drama	1997	90
3167	3236	Zachariah (1971)	Western	1971	70
3176	3245	I Am Cuba (Soy Cuba/Ya Kuba) (1964)	Drama	1964	60
3237	3306	Circus, The (1928)	Comedy	1928	20
3241	3310	Kid, The (1921)	Action	1921	20

```
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_percentage_error as mape
from sklearn.metrics import r2_score
```

```
rmse = mse(rating_movie.values[rating_movie > 0], rm_[rating_movie > 0], squared=False)
print('Root Mean Squared Error: {:.3f}'.format(rmse))
```

```
Root Mean Squared Error: 1.468
```

```
mape_ =  mape(rating_movie.values[rating_movie > 0], rm_[rating_movie > 0])
print('Mean Absolute Percentage Error: {:.3f}'.format(mape_))
```

```
Mean Absolute Percentage Error: 0.418
```

Embeddings - User-User similarity

```
user = cosine_similarity(model.A_)
```

```
user_similarity_matrix = pd.DataFrame(user, index=matrix.index, columns=matrix.index)
user_similarity_matrix.head()
```

UserID	1	2	3	4	5	6	7	
UserID								
1	1.000000	-0.131566	-0.087433	-0.372021	0.688001	0.385648	-0.015955	0.25975
2	-0.131566	1.000000	0.174572	0.792994	0.049807	0.623963	0.457949	0.38960
3	-0.087433	0.174572	1.000000	0.651531	0.172597	0.252347	0.949201	0.21447
4	-0.372021	0.792994	0.651531	1.000000	-0.184389	0.331698	0.802272	0.13207
5	0.688001	0.049807	0.172597	-0.184389	1.000000	0.804996	0.232143	0.85203

5 rows × 6040 columns

```
item = cosine_similarity(model.B_)

item_similarity_matrix = pd.DataFrame(item, index=user_item['ItemId'].unique(), columns=user_item[
item_similarity_matrix.head()
```

	1193	661	914	3408	2355	1197	1287	2804	
1193	1.000000	0.347248	0.649961	0.453908	0.761645	0.950435	0.810734	0.969964	0.6
661	0.347248	1.000000	0.490303	0.327722	0.516373	0.251613	0.208057	0.312304	0.6
914	0.649961	0.490303	1.000000	0.906059	0.879536	0.739796	0.760508	0.713627	0.9
3408	0.453908	0.327722	0.906059	1.000000	0.884974	0.658708	0.805612	0.611715	0.8
2355	0.761645	0.516373	0.879536	0.884974	1.000000	0.877524	0.934120	0.868945	0.8

5 rows × 3706 columns

```
item_similarity_matrix[594]
```

```
1193    0.607032
661     0.658896
914     0.977164
3408    0.873714
2355    0.877063
          ...
3280   -0.859901
3647   -0.372651
402    -0.342810
872    -0.082815
684     0.737391
Name: 594, Length: 3706, dtype: float32
```

```
item_similarity_matrix
```

	1193	661	914	3408	2355	1197	1287	2804
1193	1.000000	0.347248	0.649961	0.453908	0.761645	0.950435	0.810734	0.969964
661	0.347248	1.000000	0.490303	0.327722	0.516373	0.251613	0.208057	0.312304
914	0.649961	0.490303	1.000000	0.906059	0.879536	0.739796	0.760508	0.713627
3408	0.453908	0.327722	0.906059	1.000000	0.884974	0.658708	0.805612	0.611715
2355	0.761645	0.516373	0.879536	0.884974	1.000000	0.877524	0.934120	0.868945
...	...	...	...	...	...	...	...	..
3280	-0.206711	-0.469756	-0.851340	-0.752857	-0.544189	-0.282161	-0.329258	-0.247486
3647	0.152397	0.319503	-0.487988	-0.644975	-0.238129	-0.083366	-0.260667	0.010277
402	-0.227801	0.465608	-0.524140	-0.547634	-0.289542	-0.388621	-0.435592	-0.301848
872	-0.494833	0.304455	-0.223537	0.071396	0.009163	-0.373649	-0.146024	-0.357539

```
movie_name=594
movie_rating = item_similarity_matrix[movie_name]
print(movie_rating)
```

```
1193    0.607032
661     0.658896
914     0.977164
3408    0.873714
2355    0.877063
          ...
3280   -0.859901
3647   -0.372651
402    -0.342810
872    -0.082815
684     0.737391
Name: 594, Length: 3706, dtype: float32
```

```
similar_movies = item_similarity_matrix.corrwith(movie_rating)

similar_df = pd.DataFrame(similar_movies, columns=['Correlation'])
similar_df.sort_values('Correlation', ascending=False, inplace=True)

similar_df.iloc[1: , :].head()
```

	Correlation
3668	0.999137
2137	0.997830
596	0.996587
897	0.995555
915	0.995191

```
item_movie = df[['MovieID', 'Title']].copy()
item_movie.drop_duplicates(inplace=True)
item_movie.reset_index(drop=True,inplace=True)
```

```
similar_df1= similar_df.copy()
similar_df1.reset_index(inplace=True)
similar_df1.rename(columns = {'index':'MovieID'}, inplace = True)
similar_mov = pd.merge(similar_df1,item_movie,on='MovieID',how='inner')
similar_mov.head(6)
```

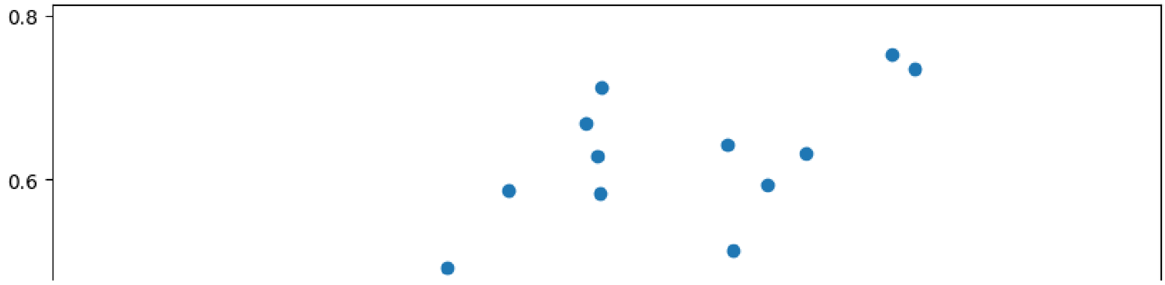
	MovieID	Correlation	Title
0	594	1.000000	Snow White and the Seven Dwarfs (1937)
1	3668	0.999137	Romeo and Juliet (1968)
2	2137	0.997830	Charlotte's Web (1973)
3	596	0.996587	Pinocchio (1940)
4	897	0.995555	For Whom the Bell Tolls (1943)
5	915	0.995191	Sabrina (1954)

```
model1 = CMF(method="als", k=2, lambda_=0.1, user_bias=False, item_bias=False, verbose=False)
model1.fit(user_item)
```

```
Collective matrix factorization model
(explicit-feedback variant)
```

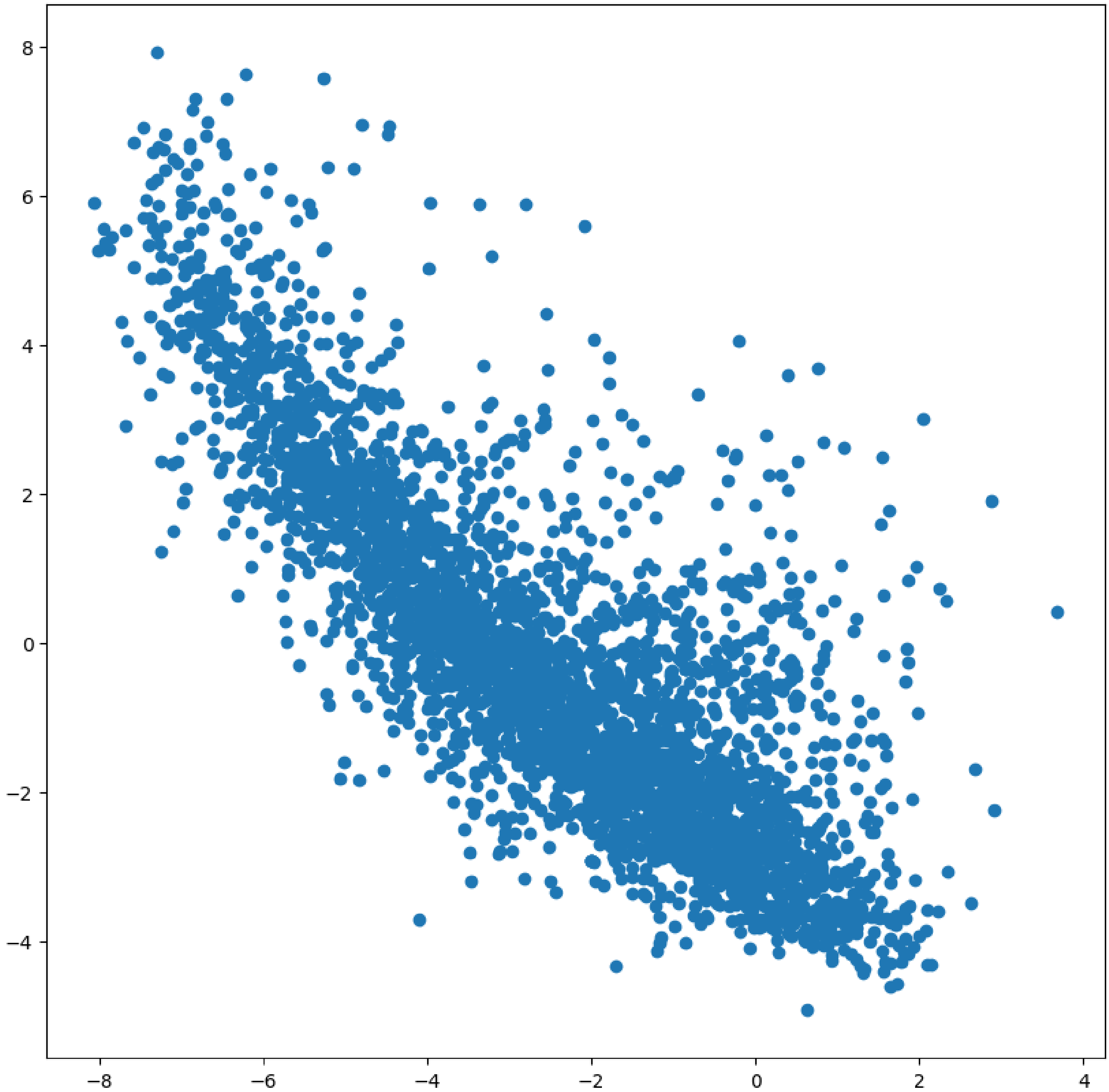
```
plt.scatter(model1.A_[:, 0], model1.A_[:, 1], cmap = 'hot')
```

```
<matplotlib.collections.PathCollection at 0x792030bc7490>
```




```
plt.scatter(model1.B_[:, 0], model1.B_[:, 1], cmap='hot')
```

```
<matplotlib.collections.PathCollection at 0x792033dd40a0>
```



Model Building using Regression

```
from sklearn.preprocessing import StandardScaler
```

```
m = movies.copy()
m.head()
```

	MovieID	Title	Genres	Release_year	Release_Decade
0	1	Toy Story (1995)	Animation|Children's|Comedy	1995	90
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	1995	90
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	90
		Waiting to			

```
u = users.copy()
u.head()
```

	UserID	Gender	Age	Occupation	Zip-code
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

```
r = ratings.copy()
r.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

```
m = pd.concat([movies['MovieID'],genres_df.iloc[:,1:]],axis=1)
m.head()
```

	MovieID	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fant
0	1	0	1	1	1	0	0	0	

```
r['Timestamp']=r['Timestamp'].astype('int32')
r['Rating']=r['Rating'].astype('int32')
r['hour'] = r['Timestamp'].apply(lambda x: datetime.fromtimestamp(x).hour)
r.head()
```

	UserID	MovieID	Rating	Timestamp	hour
0	1	1193	5	978300760	22
1	1	661	3	978302109	22
2	1	914	3	978301968	22
3	1	3408	4	978300275	22
4	1	2355	5	978824291	23

```
u_new = u.merge(r.groupby('UserID')["Rating"].mean().reset_index(), on='UserID')
u_new = u_new.merge(r.groupby('UserID').hour.mean().reset_index(), on='UserID')
u_new.head()
```

	UserID	Gender	Age	Occupation	Zip-code	Rating	hour
0	1	F	Under 18	K-12 student	48067	4.188679	22.245283
1	2	M	56+	self-employed	70072	3.713178	21.155039
2	3	M	25-34	scientist	55117	3.901961	21.000000
3	4	M	45-49	executive/managerial	02460	4.190476	20.000000
4	5	M	25-34	writer	55455	3.146465	6.015152

```
u_df = u_new[['UserID', 'Rating', 'hour']].copy()
u_df = u_df.set_index('UserID')
u_df.columns = ['User_avg_rating', 'hour']
```

```
scaler = StandardScaler()
u_df = pd.DataFrame(scaler.fit_transform(u_df), columns=u_df.columns, index=u_df.index)
u_df.head()
```

	User_avg_rating	hour
UserID		
1	1.131261	1.414540
2	0.024380	1.261846
3	0.463832	1.240132
4	1.135444	1.100078
5	-1.294827	-0.858566

```
df_cat = u_new[['Gender','Occupation']]
df_cat['Gender'] = pd.get_dummies(df_cat['Gender'], columns=['Gender'] ,drop_first=True)
df_cat = pd.concat([users['UserID'],df_cat],axis=1)
df_cat.head()
```

	UserID	Gender	Occupation
0	1	0	K-12 student
1	2	1	self-employed
2	3	1	scientist
3	4	1	executive/managerial
4	5	1	writer

```
X = ratings[['MovieID', 'UserID', 'Rating']].copy()
X = X.merge(u.reset_index(), on='UserID', how='right')
X = X.merge(m.reset_index(), on='MovieID', how='right')
X = X.merge(df_cat, on='UserID', how='right')

X.dropna(inplace=True)
X.reset_index(inplace=True,drop=True)
X1=X.copy()
X.head()
```

	MovieID	UserID	Rating	index_x	Gender_x	Age	Occupation_x	Zip-code	index_y	A(
0	1	1.0	5	0.0	F	Under 18	K-12 student	48067	0	
1	48	1.0	5	0.0	F	Under 18	K-12 student	48067	47	
2	150	1.0	5	0.0	F	Under 18	K-12 student	48067	148	
3	260	1.0	4	0.0	F	Under 18	K-12 student	48067	257	
4	527	1.0	5	0.0	F	Under 18	K-12 student	48067	523	

5 rows × 28 columns

```
X = X.drop(columns = ['MovieID', 'UserID'])
y = X.pop('Rating')
```

```
X
```

	index_x	Gender_x	Age	Occupation_x	Zip-code	index_y	Adventure	Animation
0	0.0	F	Under 18	K-12 student	48067	0	0	1
1	0.0	F	Under 18	K-12 student	48067	47	0	1
2	0.0	F	Under 18	K-12 student	48067	148	0	0
3	0.0	F	Under 18	K-12 student	48067	257	1	0
4	0.0	F	Under 18	K-12 student	48067	523	0	0
...	...	...	...	...	...	...	...	...
1000204	6039.0	M	25-34	doctor/health care	11106	3614	0	0
1000205	6039.0	M	25-34	doctor/health care	11106	3634	0	0
1000206	6039.0	M	25-34	doctor/health care	11106	3666	0	0
				doctor/health				

```
y
```

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

```
X = X.drop(columns = ['Gender_x', 'Age', 'Occupation_x', 'Zip-code','Occupation_y' ])
```

```
X_train.dtypes
```

```
index_x          float64
Gender_x          object
Age               object
Occupation_x      object
Zip-code          object
index_y            int64
Adventure          uint8
Animation          uint8
Children's         uint8
Comedy             uint8
Crime              uint8
Documentary        uint8
Drama              uint8
Fantasy            uint8
Film-Noir          uint8
```

```
Horror          uint8
Musical         uint8
Mystery         uint8
Romance         uint8
Sci-Fi          uint8
Thriller        uint8
War             uint8
Western         uint8
Gender_y        uint8
Occupation_y   object
dtype: object
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
```

```
from sklearn.ensemble import GradientBoostingRegressor

model = GradientBoostingRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
rmse = mse(y_test, y_pred, squared=False)
print('Root Mean Squared Error: {:.3f}'.format(rmse))
```

```
Root Mean Squared Error: 1.060
```

```
mape_ =  mape(y_test, y_pred) #calculating mape value
print('Mean Absolute Percentage Error: {:.3f}'.format(mape_))
```

```
Mean Absolute Percentage Error: 0.354
```

Questionnaire

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

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Age" , hue = "Rating")
plt.show()
```

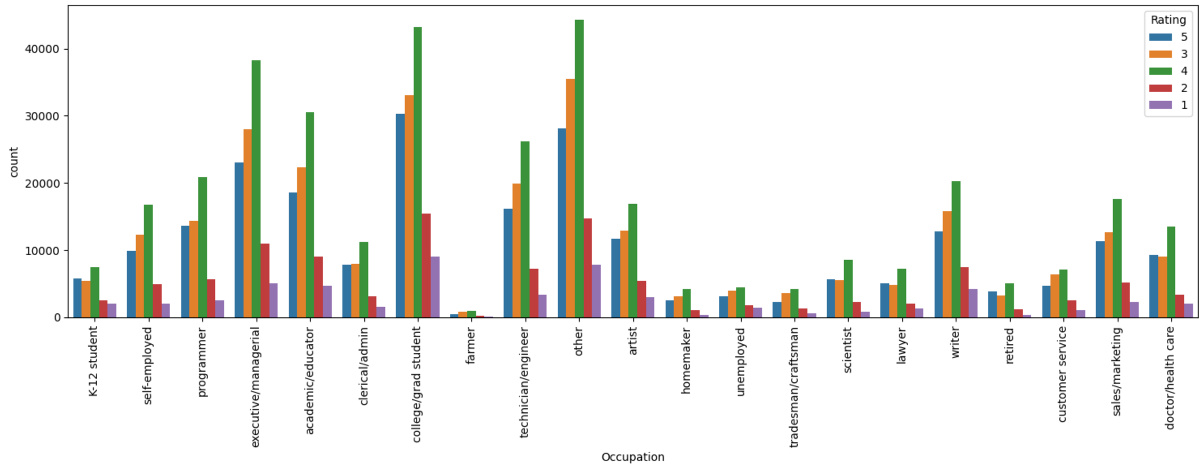

```
df["Age"].value_counts()
```

```
25-34       395556
35-44       199003
18-24       183536
45-49        83633
50-55        72490
56+          38780
Under 18     27211
Name: Age, dtype: int64
```

Answer : (25-34) age group have watched and rated the most number of movies

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

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Occupation" , hue = "Rating")
plt.xticks(rotation = 90)
plt.show()
```



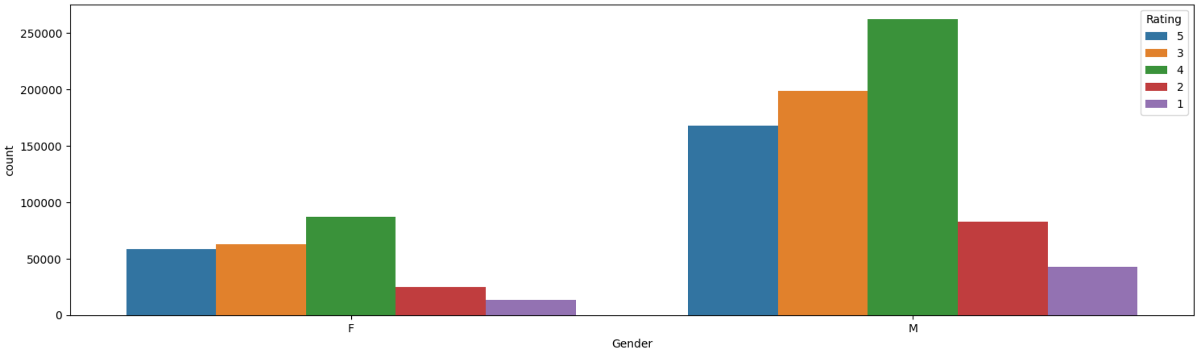

```
df["Occupation"].value_counts().head()
```

```
college/grad student    131032
other                   130499
executive/managerial    105425
academic/educator        85351
technician/engineer      72816
Name: Occupation, dtype: int64
```

Answer : "college/grad student" have watched and rated the most movies

3. Most of the users in our dataset who've rated the movies are Male. (T/F)

```
plt.figure(figsize = (18,5))
sns.countplot(data = df , x = "Gender" , hue = "Rating")
plt.show()
```




```
df["Gender"].value_counts()
```

```
M    753769
F    246440
Name: Gender, dtype: int64
```

Answer : True

4. Most of the movies present in our dataset were released in which decade?

a. 70s b. 90s c. 50s d.80s

```
df["Release_Decade"].value_counts()
```

```
90    532843
80    224056
70     82552
60     48555
0      41000
50     35232
40     21501
30     12729
20      1696
10        45
Name: Release_Decade, dtype: int64
```

Answer : (b) i.e Most of the movies present in our dataset were released in 90s decade

5. The movie with maximum no. of ratings is ___.

```
df['Title'].value_counts()
```

```
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
```

```
                                                  ...
Waiting Game, The (2000)                            1
Shadows (Cienie) (1988)                             1
Juno and Paycock (1930)                             1
Resurrection Man (1998)                             1
Windows (1980)                                      1
Name: Title, Length: 3706, dtype: int64
```

Answer : American Beauty

6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.

Answer :

1) Ace Ventura: Pet,

2) Mrs. Doubtfire,

3) Detective Dumb & Dumber

7. On the basis of approach, Collaborative Filtering methods can be classified into **_-based and _**-based.

Answer : On the basis of approach, Collaborative Filtering methods can be classified into Memory-based and Model-based.

8. Pearson Correlation ranges between **_ to _** whereas, Cosine Similarity belongs to the interval between **_ to _**.

Answer : Pearson Correlation ranges between -1 to 1 whereas, Cosine Similarity belongs to the interval between -1 to 1.

9. Mention the RMSE and MAPE that you got while evaluating the Matrix Factorization model.

Answer :

The RMSE is 1.05

The MAPE is 0.33

10. Give the sparse 'row' matrix representation for the following dense matrix -

[[1 0] [3 7]]

```
from numpy import array

from scipy.sparse import csr_matrix


A = array([[1, 0], [3,7]])
S = csr_matrix(A)
print(S)
```

```
  (0, 0)	1
  (1, 0)	3
  (1, 1)	7
```

Insights & Recommendations:-

- Based on the analysis, the target audience for *ZEE* entertainment is the occupation of category college/grade students followed by executives/managers, and others.
- Male gives more rating compared to female which shows zee get maximum data from male gender. This idea is applicable to build a robust recommendation model for new all users.
- As the female audience is less, to increase female audience ZEE can provide some offers like 30 day free trial in order to boost female ratings and views.
- Most ZEE audience age limit is from 25-34. So advertising can be done based on their interests and preferences to retain them.