DIY Project for Data Mining and Analytics- DIY 1

Vivek Kumar Shriwas

SY BSC COMPUTER SCEINCE

SRN NO. 202100756

SEMESTER III

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
```

Q1) Clean, filter and Load data

Importing the Data

> MOVIES CSV

In [2]:
```
df = pd.read_csv('Movies.csv',encoding_errors='ignore')
```

In [3]:
```
df
```

Out[3]:

	MovieID	Title	Category
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
2944	3948	Meet the Parents (2000)	Comedy
2945	3949	Requiem for a Dream (2000)	Drama
2946	3950	Tigerland (2000)	Drama
2947	3951	Two Family House (2000)	Drama
2948	3952	Contender, The (2000)	Drama|Thriller

2949 rows × 3 columns

Data Frame

In [ ]:
```
df = pd.read_csv('Movies.csv',encoding_errors='ignore')
```

In [ ]:
```
df
```

Exploding Data set

>

> The Category Column of the Movies.csv dataset having values in Pipe seperated ('|') values in series.we need to first remove this pipe seperation and align them in proper list order.

In [8]:
```
df['Category']=df['Category'].apply(lambda x: str(x.replace('|', ',')))
df
```

Out[8]:

	MovieID	Title	Category
0	1	Toy Story (1995)	Adventure,Animation,Children,Comedy,Fantasy
1	2	Jumanji (1995)	Adventure,Children,Fantasy
2	3	Grumpier Old Men (1995)	Comedy,Romance
3	4	Waiting to Exhale (1995)	Comedy,Drama,Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
2944	3948	Meet the Parents (2000)	Comedy
2945	3949	Requiem for a Dream (2000)	Drama
2946	3950	Tigerland (2000)	Drama
2947	3951	Two Family House (2000)	Drama
2948	3952	Contender, The (2000)	Drama,Thriller

2949 rows × 3 columns

In [9]:
```
df['Category']=df.Category.str.split(',')
df
```

Out[9]:

	MovieID	Title	Category
0	1	Toy Story (1995)	[Adventure, Animation, Children, Comedy, Fantasy]
1	2	Jumanji (1995)	[Adventure, Children, Fantasy]
2	3	Grumpier Old Men (1995)	[Comedy, Romance]
3	4	Waiting to Exhale (1995)	[Comedy, Drama, Romance]
4	5	Father of the Bride Part II (1995)	[Comedy]
...	...	...	...
2944	3948	Meet the Parents (2000)	[Comedy]
2945	3949	Requiem for a Dream (2000)	[Drama]
2946	3950	Tigerland (2000)	[Drama]
2947	3951	Two Family House (2000)	[Drama]
2948	3952	Contender, The (2000)	[Drama, Thriller]

2949 rows × 3 columns

In [10]:
```
df=df.explode('Category')
df
```

Out[10]:

	MovieID	Title	Category
0	1	Toy Story (1995)	Adventure
0	1	Toy Story (1995)	Animation
0	1	Toy Story (1995)	Children
0	1	Toy Story (1995)	Comedy
0	1	Toy Story (1995)	Fantasy
...	...	...	...
2945	3949	Requiem for a Dream (2000)	Drama
2946	3950	Tigerland (2000)	Drama
2947	3951	Two Family House (2000)	Drama
2948	3952	Contender, The (2000)	Drama
2948	3952	Contender, The (2000)	Thriller

6385 rows × 3 columns

Describing DataFrame

In [11]:
```
df.describe()
```

Out[11]:

	MovieID
count	6385.000000
mean	1923.312451
std	1138.398938
min	1.000000
25%	947.000000
50%	1968.000000
75%	2885.000000
max	3952.000000

Shape of the Dataset

In [12]:
```
df.shape
```

Out[12]:
```
(6385, 3)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6385 entries, 0 to 2948
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   MovieID   6385 non-null   int64 
 1   Title     6385 non-null   object
 2   Category  6385 non-null   object
dtypes: int64(1), object(2)
memory usage: 199.5+ KB
```

checking duplicate

In [14]:
```
df.duplicated()
```

Out[14]:
```
0       False
0       False
0       False
0       False
0       False
        ...
2945    False
2946    False
2947    False
2948    False
2948    False
Length: 6385, dtype: bool
```

checking Null values

In [15]:
```
df.isnull()
```

Out[15]:

	MovieID	Title	Category
0	False	False	False
0	False	False	False
0	False	False	False
0	False	False	False
0	False	False	False
...	...	...	...
2945	False	False	False
2946	False	False	False
2947	False	False	False
2948	False	False	False
2948	False	False	False

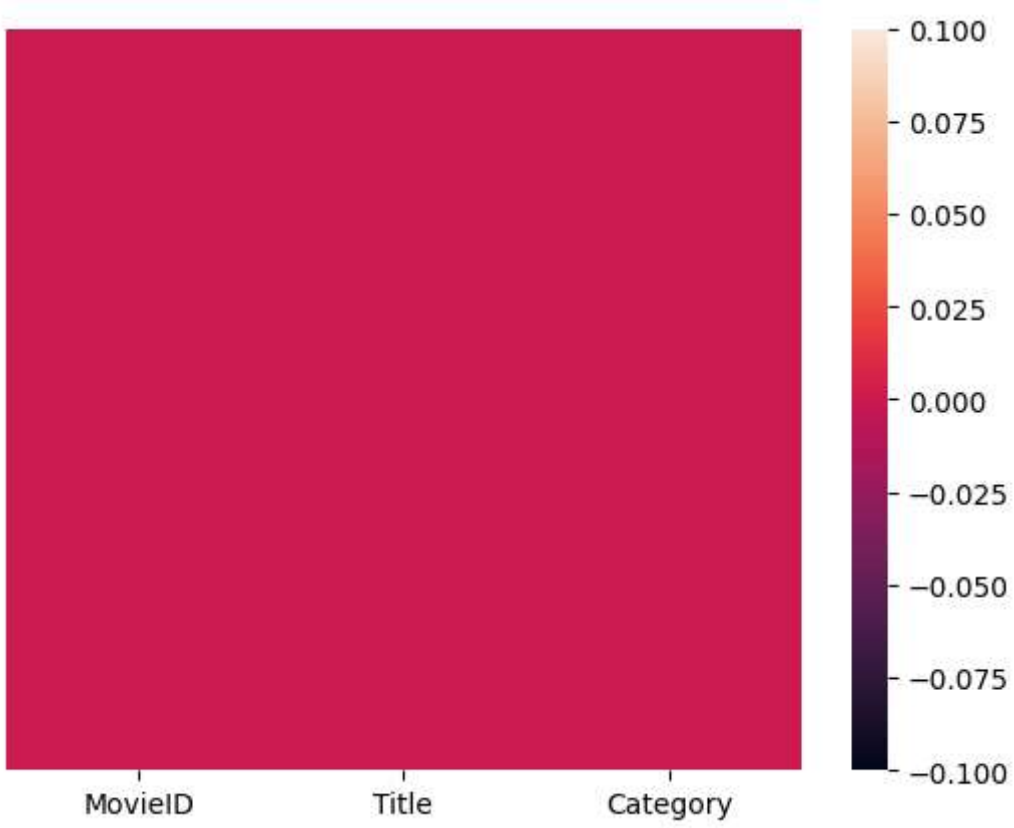
6385 rows × 3 columns

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
MovieID     0
Title       0
Category    0
dtype: int64
```

In [17]:
```
sns.heatmap(df.isnull(), yticklabels=False)
```

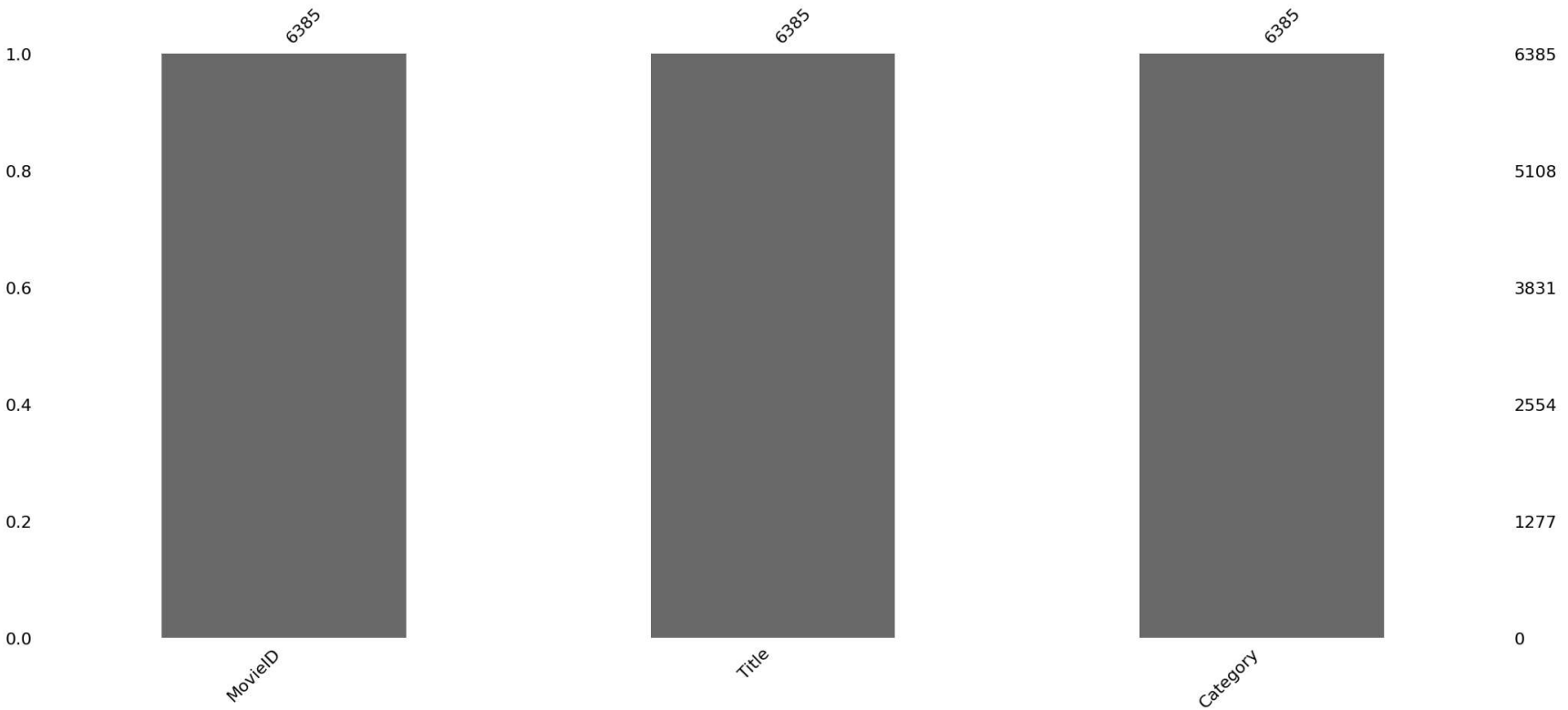
Out[17]:
```
<AxesSubplot: >
```



> Gives a bar chart of the missing values

In [18]:
```
msno.bar(df)
```

Out[18]:
```
<AxesSubplot: >
```



Filtering Data

In [20]:
```
bool_series1 = pd.isnull(df["MovieID"])
df[bool_series1]
```

Out[20]:

MovieID	Title	Category

In [21]:
```
bool_series1 = pd.isnull(df["Title"])
df[bool_series1]
```

Out[21]:

MovieID	Title	Category

In [22]:
```
bool_series1 = pd.isnull(df["Category"])
df[bool_series1]
```

Out[22]:

MovieID	Title	Category

> USER CSV

Dataframe

df1 = pd.read_csv("Users.csv")

In [25]:
```
df1
```

Out[25]:

	UserID	Gender	Age	Occupation
0	1	F	1	10
1	2	M	56	16
2	3	M	25	15
3	4	M	45	7
4	5	M	25	20
...	...	...	...	...
6035	6036	F	25	15
6036	6037	F	45	1
6037	6038	F	56	1
6038	6039	F	45	0
6039	6040	M	25	6

6040 rows × 4 columns

Describing DataFrame

In [26]:
```
df1.describe()
```

Out[26]:

	UserID	Age	Occupation
count	6040.000000	6040.000000	6040.000000
mean	3020.500000	30.639238	8.146854
std	1743.742145	12.895962	6.329511
min	1.000000	1.000000	0.000000
25%	1510.750000	25.000000	3.000000
50%	3020.500000	25.000000	7.000000
75%	4530.250000	35.000000	14.000000
max	6040.000000	56.000000	20.000000

Shape of the Dataset

In [27]:
```
df1.shape
```

Out[27]:
```
(6040, 4)
```

In [28]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 4 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 
dtypes: int64(3), object(1)
memory usage: 188.9+ KB
```

checking duplicate

In [29]:
```
df1.duplicated()
```

Out[29]:
```
0       False
1       False
2       False
3       False
4       False
        ...
6035    False
6036    False
6037    False
6038    False
6039    False
Length: 6040, dtype: bool
```

checking Null values

In [30]:
```
df1.isnull()
```

Out[30]:

	UserID	Gender	Age	Occupation
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
6035	False	False	False	False
6036	False	False	False	False
6037	False	False	False	False
6038	False	False	False	False
6039	False	False	False	False

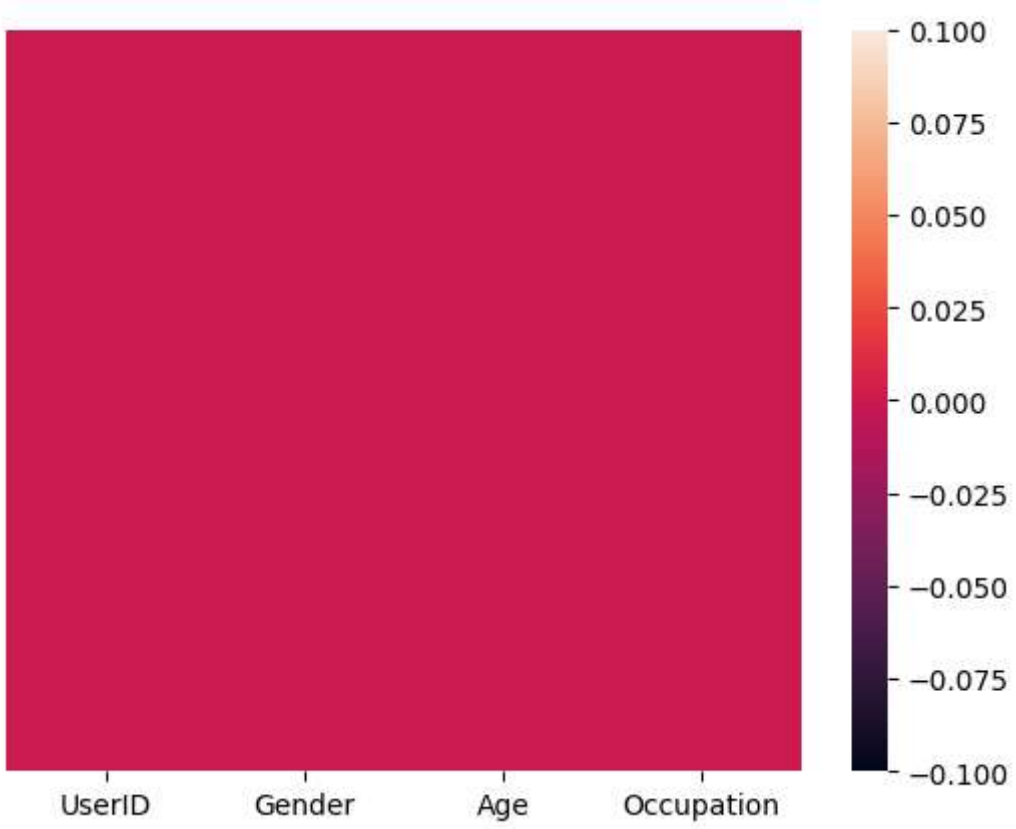
6040 rows × 4 columns

In [31]:
```
df1.isnull().sum()
```

Out[31]:
```
UserID        0
Gender        0
Age           0
Occupation    0
dtype: int64
```

In [32]:
```
sns.heatmap(df1.isnull(), yticklabels=False)
```

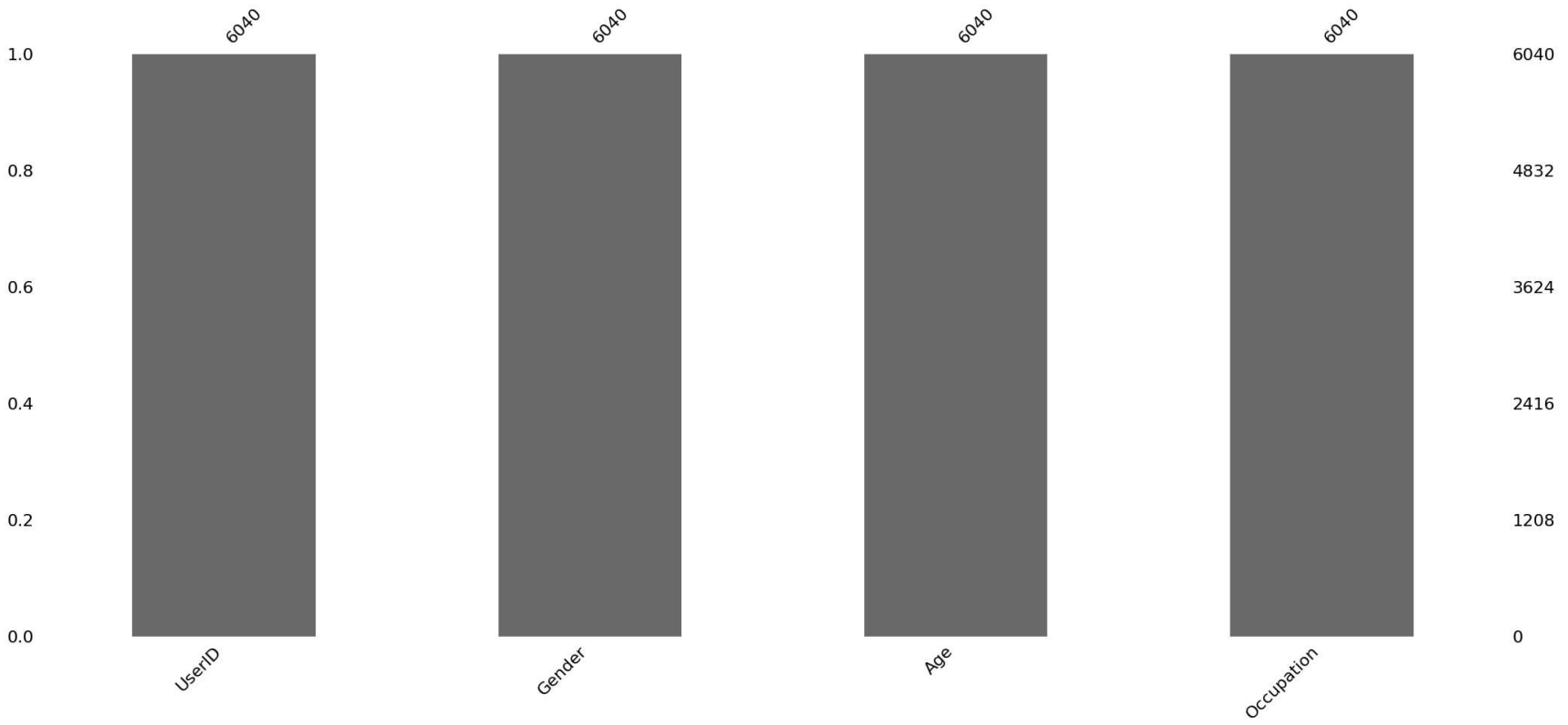
Out[32]:
```
<AxesSubplot: >
```



> Gives a bar chart of the missing values

In [33]:
```
msno.bar(df1)
```

Out[33]:
```
<AxesSubplot: >
```



In [34]:
```
bool_series1 = pd.isnull(df1["UserID"])
df1[bool_series1]
```

Out[34]:

UserID	Gender	Age	Occupation

In [35]:
```
bool_series1 = pd.isnull(df1["Gender"])
df1[bool_series1]
```

Out[35]:

UserID	Gender	Age	Occupation

In [36]:
```
bool_series1 = pd.isnull(df1["Age"])
df1[bool_series1]
```

Out[36]:

UserID	Gender	Age	Occupation

In [37]:
```
bool_series1 = pd.isnull(df1["Occupation"])
df1[bool_series1]
```

Out[37]:

	UserID	Gender	Age	Occupation

> Putting up in 'user file' age according to numbers in project file like (1-> Under 18) ,(18-> 18-24) ,(25-> 25-34) ....

In [38]:
```
Age_conditions = [
    (df1['Age'] == 1),
    (df1['Age'] == 18),
    (df1['Age'] == 25),
    (df1['Age'] == 35),
    (df1['Age'] == 45),
    (df1['Age'] == 50),
    (df1['Age'] == 56),
]

Age_values = ['Under 18','18-24','25-34','35-44','45-49','50 - 55','Above 56']

df1['Age'] = np.select(Age_conditions,Age_values)

print(df1)
```

```
      UserID Gender       Age  Occupation
0          1      F  Under 18          10
1          2      M  Above 56          16
2          3      M     25-34          15
3          4      M     45-49           7
4          5      M     25-34          20
...      ...    ...       ...         ...
6035    6036      F     25-34          15
6036    6037      F     45-49           1
6037    6038      F  Above 56           1
6038    6039      F     45-49           0
6039    6040      M     25-34           6

[6040 rows x 4 columns]
```

> Putting up in 'user file' ococcupation according to numbers in project file like 0->Not specified or other ,1->Academician ,2-> Artist ....

In [39]:

```
conditions = [
    (df1['Occupation'] == 0),
    (df1['Occupation'] == 1),
    (df1['Occupation'] == 2),
    (df1['Occupation'] == 3),
    (df1['Occupation'] == 4),
    (df1['Occupation'] == 5),
    (df1['Occupation'] == 6),
    (df1['Occupation'] == 7),
    (df1['Occupation'] == 8),
    (df1['Occupation'] == 9),
    (df1['Occupation'] == 10),
    (df1['Occupation'] == 11),
    (df1['Occupation'] == 12),
    (df1['Occupation'] == 13),
    (df1['Occupation'] == 14),
    (df1['Occupation'] == 15),
    (df1['Occupation'] == 16),
    (df1['Occupation'] == 17),
    (df1['Occupation'] == 18),
    (df1['Occupation'] == 19),
    (df1['Occupation'] == 20),
]

values = ['Not specified or other','Academician','Artist','Admin/Office  work',
         'Grad/Higher Ed student','Customer Service/Consultant','Doctor and Medical services',
          'Executive and Managerial','Farmer and Agriculture','Homemaker','K-12 Student',
          'Lawyer','Programmer','Retired','Sales and Marketing','Scientist','Self-Employed',
          'Engineer and Technician','Tradesman/Craftsman','Unemployed','Writer']

df1['Occupation'] = np.select(conditions,values)

print(df1)
```

```
      UserID Gender       Age                  Occupation
0          1      F  Under 18                K-12 Student
1          2      M  Above 56               Self-Employed
2          3      M     25-34                   Scientist
3          4      M     45-49    Executive and Managerial
4          5      M     25-34                      Writer
...      ...    ...       ...                         ...
6035    6036      F     25-34                   Scientist
6036    6037      F     45-49                 Academician
6037    6038      F  Above 56                 Academician
6038    6039      F     45-49      Not specified or other
6039    6040      M     25-34  Doctor and Medical services

[6040 rows x 4 columns]
```

> RATINGS CSV

Data Frame

In [9]:
```
df2 = pd.read_csv("Ratings.csv")
```

In [10]:
```
df2
```

Out[10]:

	UserID	MovieID	Rating
0	1	1193	5
1	1	661	3
2	1	914	3
3	1	3408	4
4	1	2355	5
...	...	...	...
1000204	6040	1091	1
1000205	6040	1094	5
1000206	6040	562	5
1000207	6040	1096	4
1000208	6040	1097	4

1000209 rows × 3 columns

In [11]:
```
df2.describe()
```

Out[11]:

	UserID	MovieID	Rating
count	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	1.865540e+03	3.581564e+00
std	1.728413e+03	1.096041e+03	1.117102e+00
min	1.000000e+00	1.000000e+00	1.000000e+00
25%	1.506000e+03	1.030000e+03	3.000000e+00
50%	3.070000e+03	1.835000e+03	4.000000e+00
75%	4.476000e+03	2.770000e+03	4.000000e+00
max	6.040000e+03	3.952000e+03	5.000000e+00

Shape of the Dataset

In [44]:
```
df2.shape
```

Out[44]:
```
(1000209, 3)
```

In [45]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 3 columns):
 #   Column   Non-Null Count    Dtype
---  ------   --------------    -----
 0   UserID   1000209 non-null  int64
 1   MovieID  1000209 non-null  int64
 2   Rating   1000209 non-null  int64
dtypes: int64(3)
memory usage: 22.9 MB
```

checking duplicate

In [46]:
```
df2.duplicated()
```

Out[46]:
```
0          False
1          False
2          False
3          False
4          False
           ...
1000204    False
1000205    False
1000206    False
1000207    False
1000208    False
Length: 1000209, dtype: bool
```

checking Null values

In [47]:
```
df2.isnull()
```

Out[47]:

	UserID	MovieID	Rating
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
1000204	False	False	False
1000205	False	False	False
1000206	False	False	False
1000207	False	False	False
1000208	False	False	False

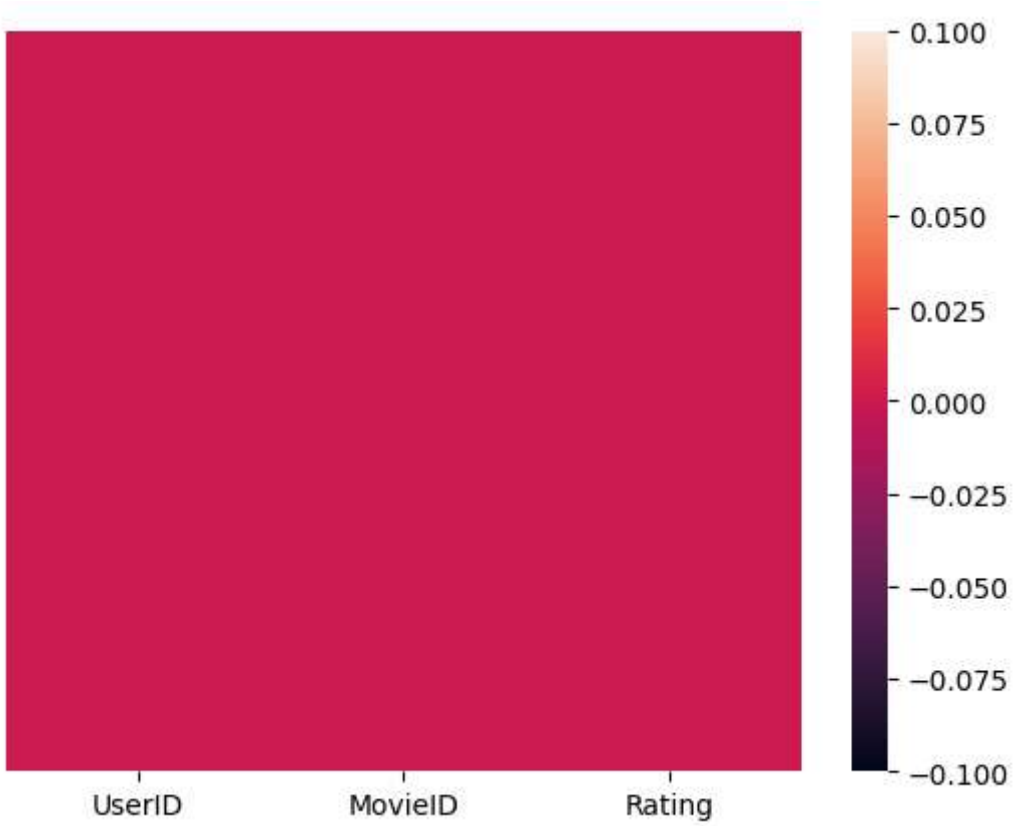
1000209 rows × 3 columns

In [48]:
```
df2.isnull().sum()
```

Out[48]:
```
UserID     0
MovieID    0
Rating     0
dtype: int64
```

In [49]:
```
sns.heatmap(df2.isnull(), yticklabels=False)
```

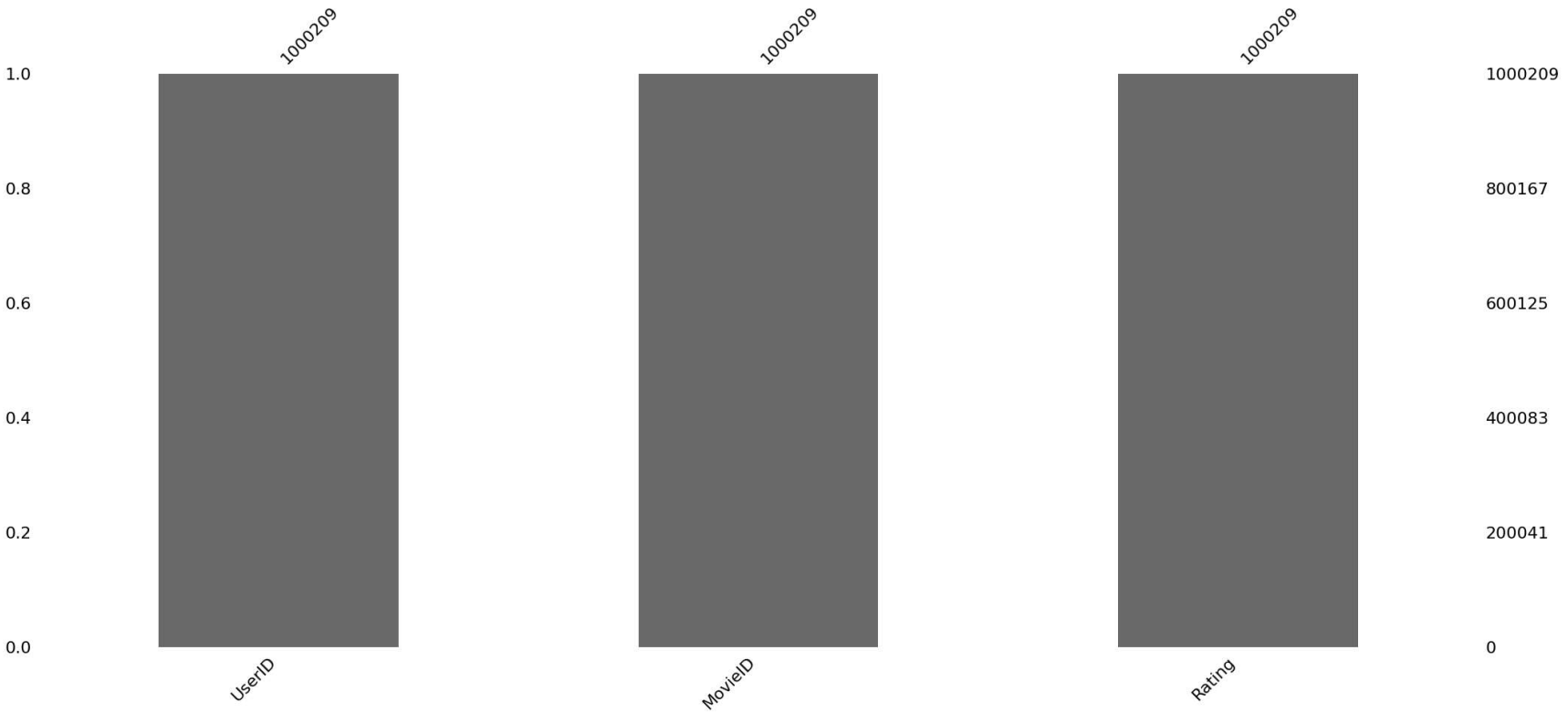
Out[49]:
```
<AxesSubplot: >
```



> Gives a bar chart of the missing values

In [50]:
```
msno.bar(df2)
```

Out[50]:
```
<AxesSubplot: >
```



Filter Data

In [52]:
```
bool_series1 = pd.isnull(df2["UserID"])
df2[bool_series1]
```

Out[52]:

UserID	MovieID	Rating

In [53]:
```
bool_series2 = pd.isnull(df2["MovieID"])
df2[bool_series2]
```

Out[53]:

UserID	MovieID	Rating

In [54]:
```
bool_series3 = pd.isnull(df2["Rating"])
df2[bool_series3]
```

Out[54]:

UserID	MovieID	Rating

In [ ]:

```
```

---------- Q2)----------

> i) Total number of movies released in each year. Display year and the count.

> Creating a new column name Year and putting up the year's of movive released

In [56]:

```
df['Year']=df["Title"].str[-5:-1]
df
```

Out[56]:

	MovieID	Title	Category	Year
0	1	Toy Story (1995)	Adventure	1995
0	1	Toy Story (1995)	Animation	1995
0	1	Toy Story (1995)	Children	1995
0	1	Toy Story (1995)	Comedy	1995
0	1	Toy Story (1995)	Fantasy	1995
...	...	...	...	...
2945	3949	Requiem for a Dream (2000)	Drama	2000
2946	3950	Tigerland (2000)	Drama	2000
2947	3951	Two Family House (2000)	Drama	2000
2948	3952	Contender, The (2000)	Drama	2000
2948	3952	Contender, The (2000)	Thriller	2000

6385 rows × 4 columns

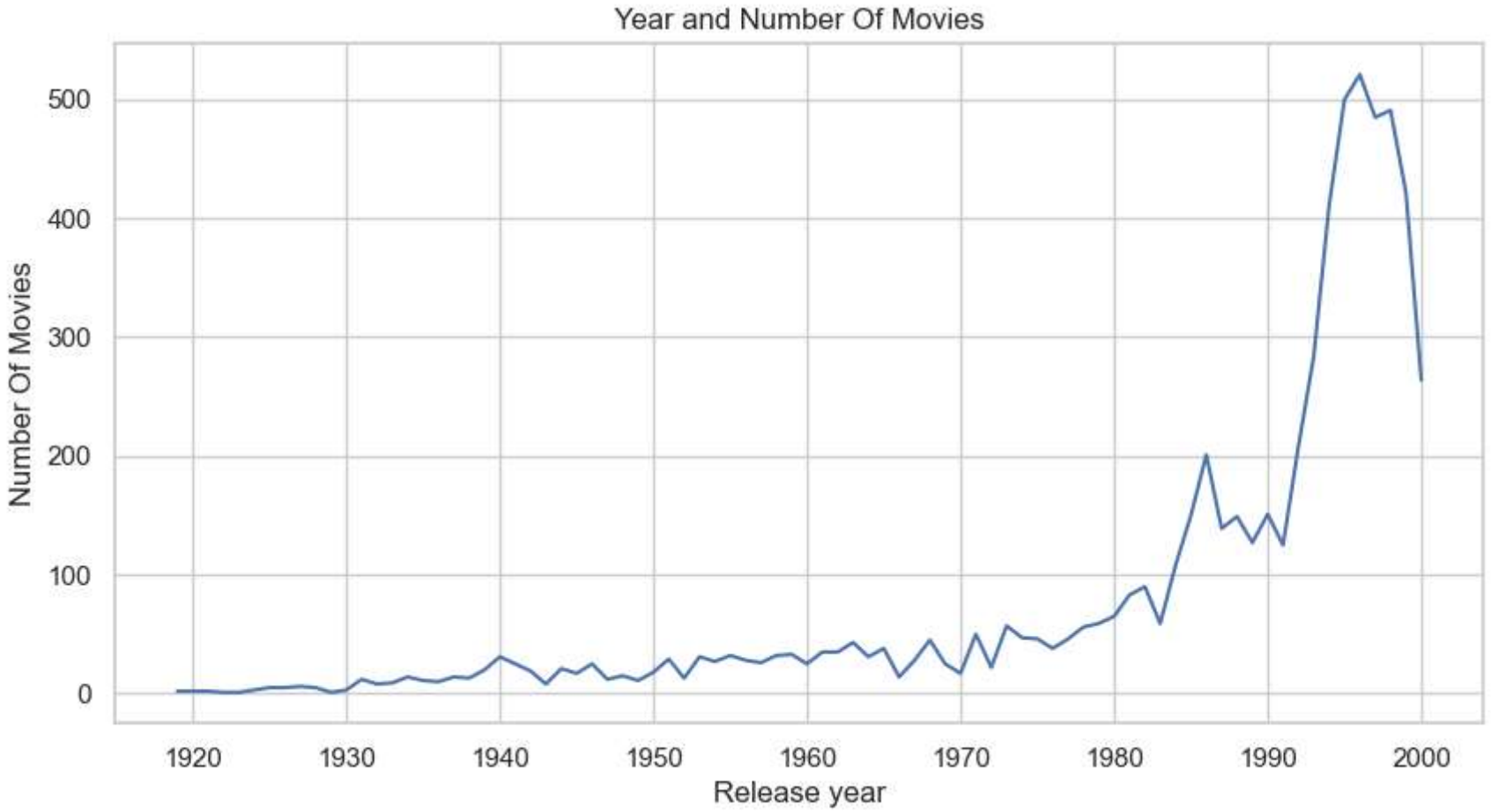
In [103…

```
data=df.groupby("Year").count()["MovieID"]
print(data.tail())

df.groupby("Year").count()["MovieID"].plot()

# sns.set(rc={'figure.figsize':(10,5)})
plt.title("Year and Number Of Movies")
plt.xlabel('Release year')
plt.ylabel('Number Of Movies')
sns.set_style("whitegrid")
```

```
Year
1996    521
1997    485
1998    491
1999    422
2000    264
Name: MovieID, dtype: int64
```



In [58]:
```
df['Year'].dtypes
```

Out[58]:
```
dtype('O')
```

In [59]:
```
df = df.astype({'Year':'int'})
```

In [60]:
```
df['Year'].dtypes
```

Out[60]:
```
dtype('int32')
```

In [ ]:
```

```

> ii) Find the movie category having highest ratings in each year. The output should comprise of
>
> YearofRelease, Category, CountOfMovies having highest rating.

In [ ]:
```
```

> iii) Find and display movie category and age group wise likings.

This may be based on the highest number of users from each age group rated for that movie category.

In [64]:
```
All_merged_df = pd.merge(df, df2, on='MovieID')
All_merged_df = pd.merge(All_merged_df, df1, on='UserID')
All_merged_df
```

Out[64]:

	MovieID	Title	Category	Year	UserID	Rating	Gender	Age	Occupation
0	1	Toy Story (1995)	Adventure	1995	1	5	F	Under 18	K-12 Student
1	1	Toy Story (1995)	Animation	1995	1	5	F	Under 18	K-12 Student
2	1	Toy Story (1995)	Children	1995	1	5	F	Under 18	K-12 Student
3	1	Toy Story (1995)	Comedy	1995	1	5	F	Under 18	K-12 Student
4	1	Toy Story (1995)	Fantasy	1995	1	5	F	Under 18	K-12 Student
...	...	...	...	...	...	...	...	...	...
2513046	3826	Hollow Man (2000)	Horror	2000	3893	1	M	25-34	Doctor and Medical services
2513047	3826	Hollow Man (2000)	Sci-Fi	2000	3893	1	M	25-34	Doctor and Medical services
2513048	3826	Hollow Man (2000)	Thriller	2000	3893	1	M	25-34	Doctor and Medical services
2513049	3831	Saving Grace (2000)	Comedy	2000	3893	4	M	25-34	Doctor and Medical services
2513050	3859	Eyes of Tammy Faye, The (2000)	Documentary	2000	3893	2	M	25-34	Doctor and Medical services

2513051 rows × 9 columns

In [65]:
```
def find_minmax(x):
    min_index = All_merged_df[x].idxmin()
    high_index = All_merged_df[x].idxmax()
```

```
    high = pd.DataFrame(All_merged_df.loc[high_index,:])
    low = pd.DataFrame(All_merged_df.loc[min_index,:])

    print("Movie Which Has Highest "+ x + " : ",All_merged_df['Category'][high_index])
    print("Movie Which Has Lowest "+ x + "  : ",All_merged_df['Category'][min_index])
```

In [66]:
```
find_minmax('Rating')
```

```
Movie Which Has Highest Rating :  Adventure
Movie Which Has Lowest Rating  :  Drama
```

In [67]:
```
def find_minmax(x):
    min_index = All_merged_df[x].idxmin()
    high_index = All_merged_df[x].idxmax()
    high = pd.DataFrame(All_merged_df.loc[high_index,:])
    low = pd.DataFrame(All_merged_df.loc[min_index,:])

    print("Age group to watch Highest Movies " + ": ",All_merged_df['Age'][high_index])
    print("Age group to watch Lowest Movies  "+ ": ",All_merged_df['Age'][min_index])
```

In [68]:
```
find_minmax('Rating')
```

```
Age group to watch Highest Movies :  Under 18
Age group to watch Lowest Movies  :  50 - 55
```

> iv) Use Cluster models to segregate movie category and age group wise likings.

In [70]:
```
Seg_df = pd.read_csv('Movies.csv',encoding_errors='ignore')
```

In [71]:
```
Seg_df
```

Out[71]:

	MovieID	Title	Category
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
2944	3948	Meet the Parents (2000)	Comedy
2945	3949	Requiem for a Dream (2000)	Drama
2946	3950	Tigerland (2000)	Drama
2947	3951	Two Family House (2000)	Drama
2948	3952	Contender, The (2000)	Drama|Thriller

2949 rows × 3 columns

In [72]:
```
test_ = Seg_df["Category"].str.split('|',n=4,expand=True)
```

In [73]:
```
test_
```

Out[73]:

	0	1	2	3	4
0	Adventure	Animation	Children	Comedy	Fantasy
1	Adventure	Children	Fantasy	None	None
2	Comedy	Romance	None	None	None
3	Comedy	Drama	Romance	None	None
4	Comedy	None	None	None	None
...	...	...	...	...	...
2944	Comedy	None	None	None	None
2945	Drama	None	None	None	None
2946	Drama	None	None	None	None
2947	Drama	None	None	None	None
2948	Drama	Thriller	None	None	None

2949 rows × 5 columns

In [74]:
```
test_.rename(columns = {0:'Category1',1:'Category2',2:'Category3',3:'Category4',4:'Category5'})
```

Out[74]:

	Category1	Category2	Category3	Category4	Category5
0	Adventure	Animation	Children	Comedy	Fantasy
1	Adventure	Children	Fantasy	None	None
2	Comedy	Romance	None	None	None
3	Comedy	Drama	Romance	None	None
4	Comedy	None	None	None	None
...	...	...	...	...	...
2944	Comedy	None	None	None	None
2945	Drama	None	None	None	None
2946	Drama	None	None	None	None
2947	Drama	None	None	None	None
2948	Drama	Thriller	None	None	None

2949 rows × 5 columns

In [75]:
```
Seg_df[['Category1','Category2','Category3','Category4','Category5']] = Seg_df["Category"].str.split('|',n=4,expand=Tru
Seg_df
```

Out[75]:

	MovieID	Title	Category	Category1	Category2	Category3	Category4	Category5
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Adventure	Animation	Children	Comedy	Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy	Adventure	Children	Fantasy	None	None
2	3	Grumpier Old Men (1995)	Comedy|Romance	Comedy	Romance	None	None	None
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	Comedy	Drama	Romance	None	None
4	5	Father of the Bride Part II (1995)	Comedy	Comedy	None	None	None	None
...	...	...	...	...	...	...	...	...
2944	3948	Meet the Parents (2000)	Comedy	Comedy	None	None	None	None
2945	3949	Requiem for a Dream (2000)	Drama	Drama	None	None	None	None
2946	3950	Tigerland (2000)	Drama	Drama	None	None	None	None
2947	3951	Two Family House (2000)	Drama	Drama	None	None	None	None
2948	3952	Contender, The (2000)	Drama|Thriller	Drama	Thriller	None	None	None

2949 rows × 8 columns

In [76]:
```
Seg_df.drop('Category', axis = 1)
```

Out[76]:

	MovieID	Title	Category1	Category2	Category3	Category4	Category5
0	1	Toy Story (1995)	Adventure	Animation	Children	Comedy	Fantasy
1	2	Jumanji (1995)	Adventure	Children	Fantasy	None	None
2	3	Grumpier Old Men (1995)	Comedy	Romance	None	None	None
3	4	Waiting to Exhale (1995)	Comedy	Drama	Romance	None	None
4	5	Father of the Bride Part II (1995)	Comedy	None	None	None	None
...	...	...	...	...	...	...	...
2944	3948	Meet the Parents (2000)	Comedy	None	None	None	None
2945	3949	Requiem for a Dream (2000)	Drama	None	None	None	None
2946	3950	Tigerland (2000)	Drama	None	None	None	None
2947	3951	Two Family House (2000)	Drama	None	None	None	None
2948	3952	Contender, The (2000)	Drama	Thriller	None	None	None

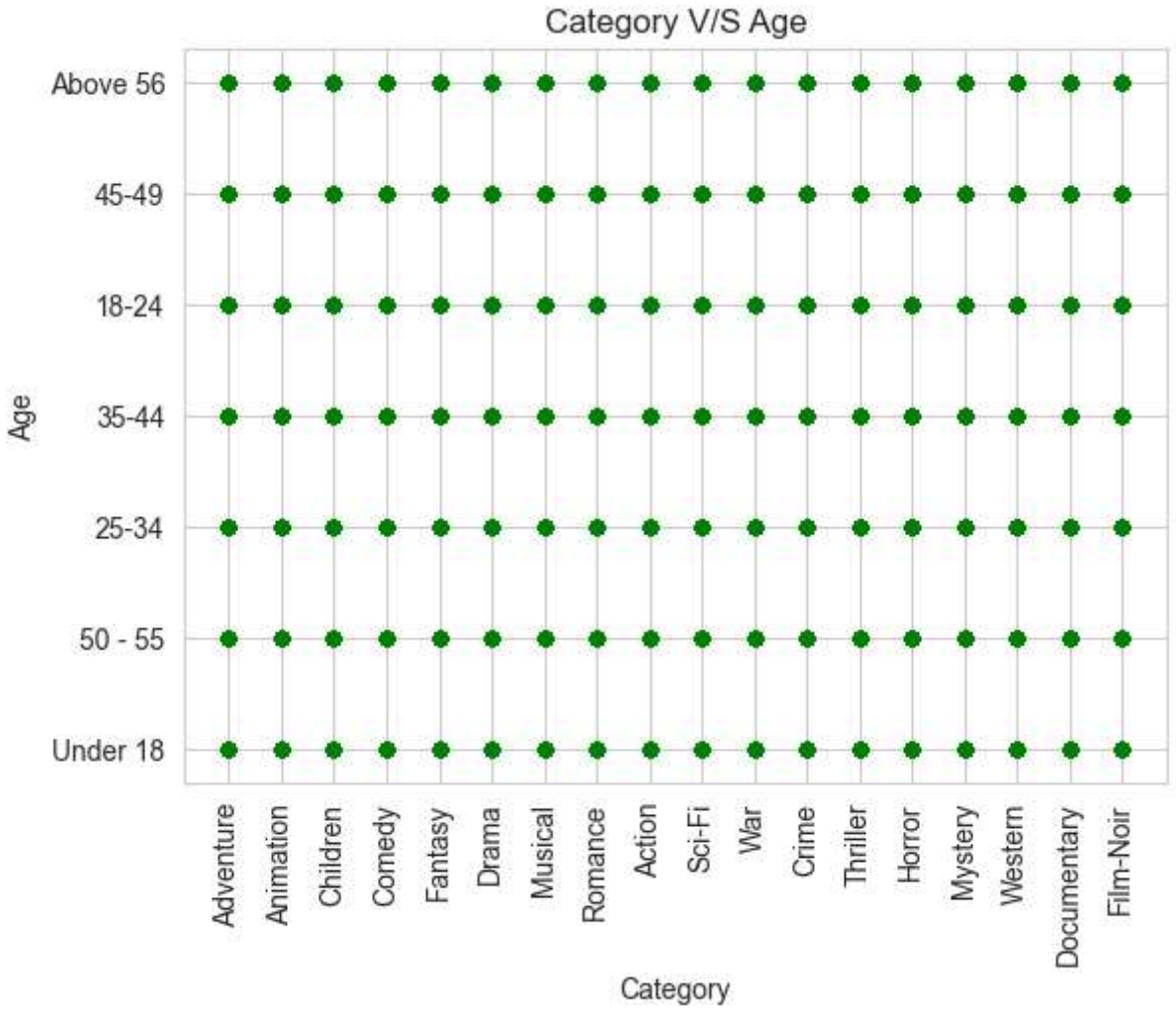
2949 rows × 7 columns

> age group wise likings

> All_merged_df.plot.scatter(x = 'Category', y = 'Occupation', s = 100);

In [78]:
```
ax2 = All_merged_df.plot.scatter(title="Category V/S Age",x='Category',y='Age',c='Green')
plt.xticks(rotation = 90)
```

Out[78]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17],
 [Text(0, 0, 'Adventure'),
  Text(1, 0, 'Animation'),
  Text(2, 0, 'Children'),
  Text(3, 0, 'Comedy'),
  Text(4, 0, 'Fantasy'),
  Text(5, 0, 'Drama'),
  Text(6, 0, 'Musical'),
  Text(7, 0, 'Romance'),
  Text(8, 0, 'Action'),
  Text(9, 0, 'Sci-Fi'),
  Text(10, 0, 'War'),
  Text(11, 0, 'Crime'),
  Text(12, 0, 'Thriller'),
  Text(13, 0, 'Horror'),
  Text(14, 0, 'Mystery'),
  Text(15, 0, 'Western'),
  Text(16, 0, 'Documentary'),
  Text(17, 0, 'Film-Noir')])
```



> v) Display year wise count of movies released.

In [80]:
```
data=df.groupby("Year").count()["MovieID"]
print(data.head(100))
```

```
Year
1919      2
1921      2
1922      1
1923      1
1925      5
        ...
1996    521
1997    485
1998    491
1999    422
2000    264
Name: MovieID, Length: 80, dtype: int64
```

In [ ]:

> vi) Display year wise, category wise count of movies released.

In [82]:
```
data1=df.groupby("Category").count()["MovieID"]
data1.head()
```

Out[82]:
```
Category
Action        470
Adventure     368
Animation     100
Children      246
Comedy       1072
Name: MovieID, dtype: int64
```

In [83]:
```
data2=df.groupby("Year").count()["MovieID"]
data2.head()
```

Out[83]:
```
Year
1919    2
1921    2
1922    1
1923    1
1925    5
Name: MovieID, dtype: int64
```

Merge

In [ ]:
```
Q = pd.DataFrame(df, columns=['Year','Category'])
```

In [86]:
```
Q
```

Out[86]:

	Year	Category
0	1995	Adventure
0	1995	Animation
0	1995	Children
0	1995	Comedy
0	1995	Fantasy
...	...	...
2945	2000	Drama
2946	2000	Drama
2947	2000	Drama
2948	2000	Drama
2948	2000	Thriller

6385 rows × 2 columns

In [87]:
```
Out=pd.merge(data2,Q, on='Year')
```

In [88]:
```
Out
```

Out[88]:

	Year	MovieID	Category
0	1919	2	Comedy
1	1919	2	Drama
2	1921	2	Comedy
3	1921	2	Drama
4	1922	1	Horror
...	...	...	...
6380	2000	264	Drama
6381	2000	264	Drama
6382	2000	264	Drama
6383	2000	264	Drama
6384	2000	264	Thriller

6385 rows × 3 columns

In [89]:
```
Out2=pd.merge(data1,Q, on='Category')
Out2
```

Out[89]:

	Category	MovieID	Year
0	Action	470	1995
1	Action	470	1995
2	Action	470	1995
3	Action	470	1995
4	Action	470	1995
...	...	...	...
6380	Western	62	1962
6381	Western	62	1946
6382	Western	62	1969
6383	Western	62	1953
6384	Western	62	1965

6385 rows × 3 columns

In [90]:
```
Out3=pd.merge(Out,Out2, on='Year')
```

In [91]:
```
Out3.rename(columns = {'MovieID_x':'Movies Per Year'}, inplace = True)
Out3.rename(columns ={'MovieID_y':'Movies Per Category'}, inplace = True)
Out3.rename(columns ={'Category_x':'Category'}, inplace = True)
Out3=Out3.drop(['Category_y'], axis=1)
```

In [92]:
```
Out3
```

Out[92]:

	Year	Movies Per Year	Category	Movies Per Category
0	1919	2	Comedy	1072
1	1919	2	Comedy	1390
2	1919	2	Drama	1072
3	1919	2	Drama	1390
4	1921	2	Comedy	1072
...	...	...	...	...
1776762	2000	264	Thriller	563
1776763	2000	264	Thriller	563
1776764	2000	264	Thriller	119
1776765	2000	264	Thriller	119
1776766	2000	264	Thriller	62

1776767 rows × 4 columns

In [ ]:

> vii) Use Clustering methods to segregate movie category and occupation of users. Display details. With the data, train the model and predict that when an occupation is entered, the model can predict the movie likings to that category of occupation.

In [94]:
```
All_merged_df["Category"]
```

Out[94]:
```
0            Adventure
1            Animation
2             Children
3               Comedy
4              Fantasy
              ...
2513046         Horror
2513047         Sci-Fi
2513048       Thriller
2513049         Comedy
2513050    Documentary
Name: Category, Length: 2513051, dtype: object
```

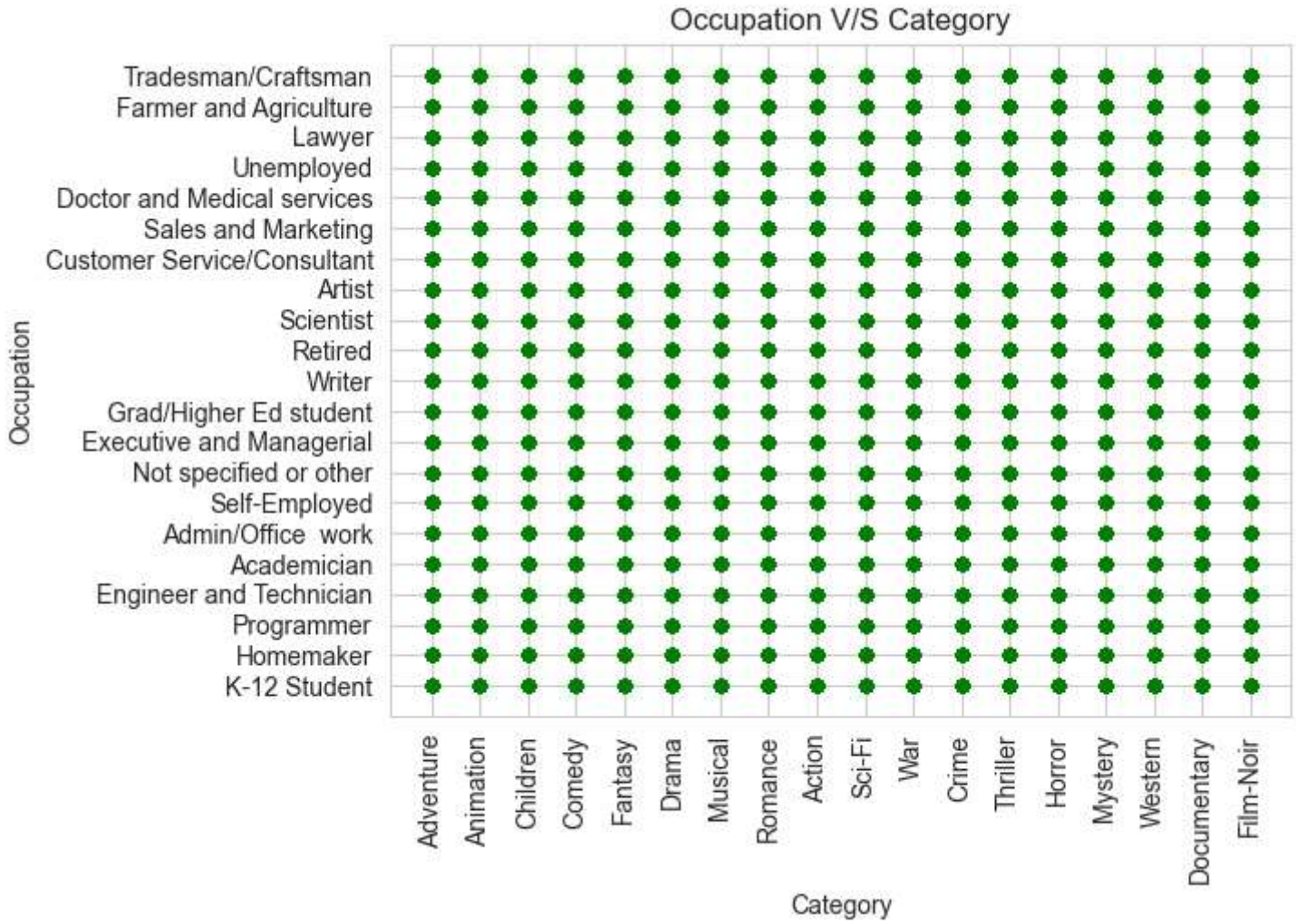
In [95]:
```
All_merged_df["Occupation"]
```

Out[95]:
```
0                        K-12 student
1                        K-12 student
2                        K-12 student
3                        K-12 student
4                        K-12 student
                      ...
2513046    Doctor and Medical services
2513047    Doctor and Medical services
2513048    Doctor and Medical services
2513049    Doctor and Medical services
2513050    Doctor and Medical services
Name: Occupation, Length: 2513051, dtype: object
```

> All_merged_df.plot.scatter(x = 'Category', y = 'Occupation', s = 100);

In [96]:
```
ax2 = All_merged_df.plot.scatter(title="Occupation V/S Category",x='Category',y='Occupation',c='Green')
plt.xticks(rotation = 90)
```

Out[96]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17],
 [Text(0, 0, 'Adventure'),
  Text(1, 0, 'Animation'),
  Text(2, 0, 'Children'),
  Text(3, 0, 'Comedy'),
  Text(4, 0, 'Fantasy'),
  Text(5, 0, 'Drama'),
  Text(6, 0, 'Musical'),
  Text(7, 0, 'Romance'),
  Text(8, 0, 'Action'),
  Text(9, 0, 'Sci-Fi'),
  Text(10, 0, 'War'),
  Text(11, 0, 'Crime'),
  Text(12, 0, 'Thriller'),
  Text(13, 0, 'Horror'),
  Text(14, 0, 'Mystery'),
  Text(15, 0, 'Western'),
  Text(16, 0, 'Documentary'),
  Text(17, 0, 'Film-Noir')])
```



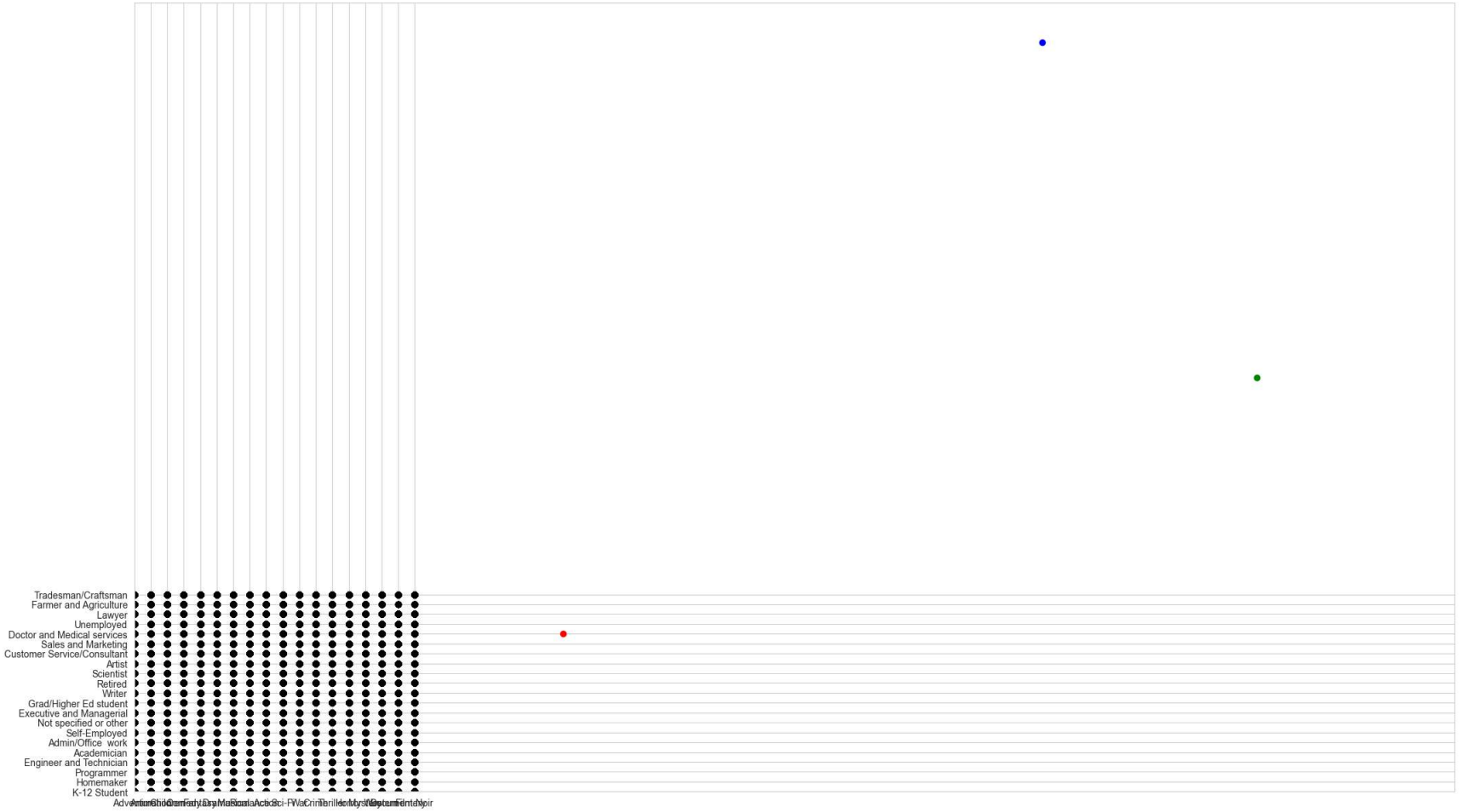
> Initialising

In [97]:
```
test_df = pd.DataFrame({
    'x':All_merged_df["Category"],
    'y':All_merged_df["Occupation"]
})

np.random.seed(200)
k = 3
# centroids[i] = [x,y]
```

```
centroids = {
    i+1:[np.random.randint(0,80),np.random.randint(0,80)]
    for i in range(k)
}

fig = plt.figure(figsize=(25,15))
plt.scatter(test_df['x'],test_df['y'], color='k')
colmap = {1:'r',2:'g',3:'b'}
for i in centroids. keys():
    plt.scatter(*centroids[i],color = colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```



In [ ]:

> viii) Now include, the age group to the above model in addition to occupation of users to refine the predictive model.

In [99]:
```
md_small = All_merged_df.iloc[:, [1,3,5,6,7,8]]
md_small.head()
md_small.dtypes
md_small['Occupation'] = md_small['Occupation'].str[:5]
pd.to_numeric(md_small['Occupation'])
md_small[md_small.columns[1:]].corr()['Rating'][:]
master_features = pd.merge(md_small, one_hot_genres, left_index=True, right_index=True)
master_features.head()
X_feature = md_small.drop(['Occupation'], axis=1)
X_feature.head()
X_feature_small = X_feature[X_feature['MovieID'] < 50]
X_feature_small_trimmed = X_feature_small.drop(['MovieID','Rating'], axis=1)
X_feature_small_trimmed.shape
X_feature_small_trimmed.head()
Y_target = master_features['Rating'][master_features['MovieID']< 50]
Y_target.shape
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_feature_small_trimmed,Y_target,random_state=1)
from sklearn.linear_model import LogisticRegression
#logreg = LogisticRegression(solver='lbfgs',class_weight='balanced', max_iter=100000)
logreg = LogisticRegression(max_iter=100000)
logreg.fit(x_train,y_train)
y_pred = logreg.predict(x_test)
from sklearn import metrics
metrics.accuracy_score(y_test,y_pred)

# print the first 30 true and predicted responses
print ('actual:    ', y_test.values[0:30])
print ('predicted: ', y_pred[0:30])
```

In [ ]:
```
```

User Interface Q3)

> Develop a simple User Interface including all the queries and processes above to make it a functional system.

In [101…

```
# from pandasgui import show
 gui = show(All_merged_df)
```

In [ ]: