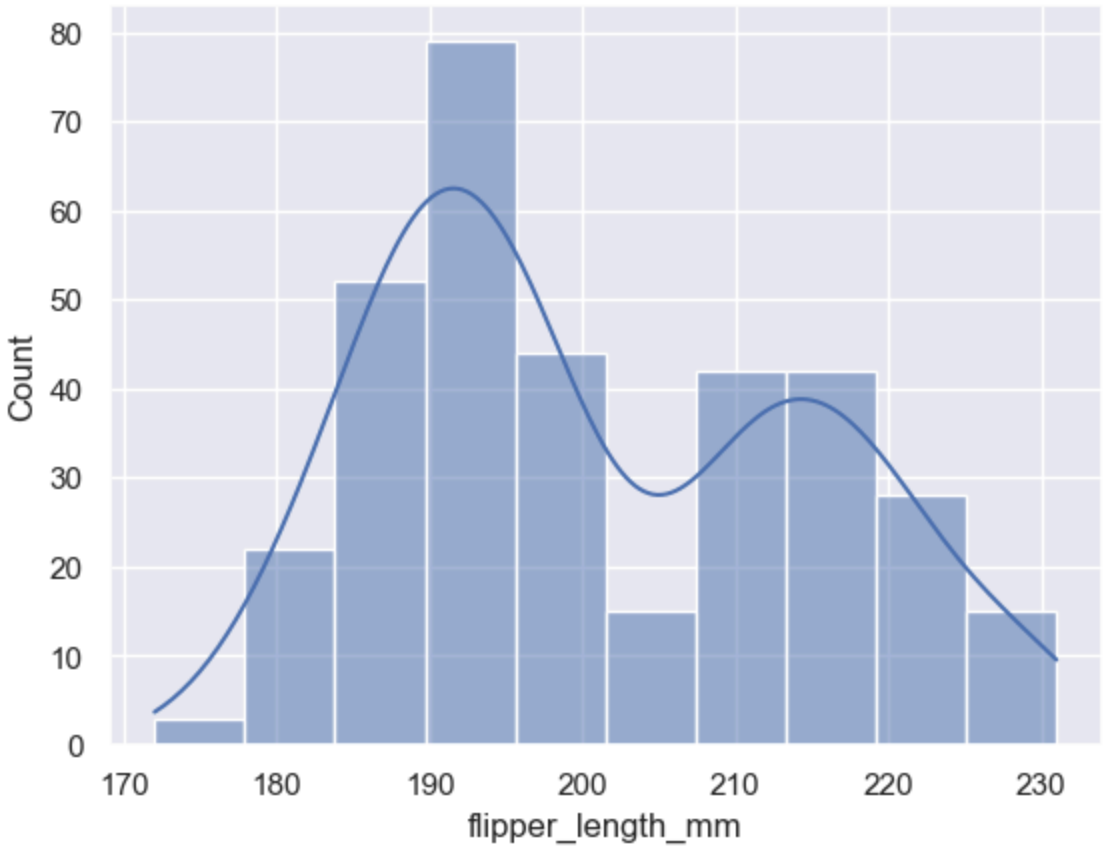
In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(color_codes=True)
```

In [3]:
```
import seaborn as sns
sns.set_theme()
df = sns.load_dataset("penguins")
```

In [4]:
```
sns.histplot(df["flipper_length_mm"], kde=True)
```

Out[4]: <Axes: xlabel='flipper_length_mm', ylabel='Count'>



In [5]:
```
u_cols = ['Userid','Age','Gender','Occupation','Zip code']
users = pd.read_csv('C:/Users/asus/u.user', names=u_cols, sep='|')
users.head()
```

Out[5]:

	Userid	Age	Gender	Occupation	Zip code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

In [6]:
```
users.isna().apply(pd.value_counts) #missing value check
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\2030029336.py:1: FutureWarning: pand
as.value_counts is deprecated and will be removed in a future version. Use pd.Series
(obj).value_counts() instead.
  users.isna().apply(pd.value_counts) #missing value check
```

Out[6]:

	Userid	Age	Gender	Occupation	Zip code
False	943	943	943	943	943

In [7]:
```
#read u.data file from the folder and name the columns after referring to the Readm
# , | \t
ratings = pd.read_csv('C:/Users/asus/u.data',
                      sep = '\t',names= ['UseID', 'ItemID', 'rating', 'Timestamp' ]
print(ratings.shape)
ratings.head(20)
```

```
(100000, 4)
```

Out[7]:

	UseID	ItemID	rating	Timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
5	298	474	4	884182806
6	115	265	2	881171488
7	253	465	5	891628467
8	305	451	3	886324817
9	6	86	3	883603013
10	62	257	2	879372434
11	286	1014	5	879781125
12	200	222	5	876042340
13	210	40	3	891035994
14	224	29	3	888104457
15	303	785	3	879485318
16	122	387	5	879270459
17	194	274	2	879539794
18	291	1042	4	874834944
19	234	1184	2	892079237

In [8]:
```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   UseID      100000 non-null  int64
 1   ItemID     100000 non-null  int64
 2   rating     100000 non-null  int64
 3   Timestamp  100000 non-null  int64
dtypes: int64(4)
memory usage: 3.1 MB
```

In [9]:
```
ratings.isna().apply(pd.value_counts)   #missing value check
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\2061556625.py:1: FutureWarning: pand
as.value_counts is deprecated and will be removed in a future version. Use pd.Series
(obj).value_counts() instead.
  ratings.isna().apply(pd.value_counts)   #missing value check
```

Out[9]:

	UseID	ItemID	rating	Timestamp
False	100000	100000	100000	100000

In [10]:
```
# Create a list of column names by copy pasting from the Readme file and read the u
col_n = ['movie id' , 'movie title' , 'release date' , 'video release date' ,
              'IMDb URL' , 'unknown' , 'Action' , 'Adventure' , 'Animation' ,
              "Children's" , 'Comedy' , 'Crime' , 'Documentary' , 'Drama' , 'Fantas
              'Film-Noir' , 'Horror' , 'Musical' , 'Mystery' , 'Romance' , 'Sci-Fi'
              'Thriller' , 'War' , 'Western']


movies = pd.read_csv('C:/Users/asus/u.item',
                     sep = '|',encoding = 'latin-1',names=col_n )
print(movies.shape)
movies.head()
```

```
(1682, 24)
```

Out[10]:

	movie id	movie title	release date	video release date	IMDb URL	unknown	Action	Adven
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	

5 rows × 24 columns

In [11]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1682 entries, 0 to 1681
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   movie id            1682 non-null   int64  
 1   movie title         1682 non-null   object 
 2   release date        1681 non-null   object 
 3   video release date  0 non-null      float64
 4   IMDb URL            1679 non-null   object 
 5   unknown             1682 non-null   int64  
 6   Action              1682 non-null   int64  
 7   Adventure           1682 non-null   int64  
 8   Animation           1682 non-null   int64  
 9   Children's          1682 non-null   int64  
 10  Comedy              1682 non-null   int64  
 11  Crime               1682 non-null   int64  
 12  Documentary         1682 non-null   int64  
 13  Drama               1682 non-null   int64  
 14  Fantasy             1682 non-null   int64  
 15  Film-Noir           1682 non-null   int64  
 16  Horror              1682 non-null   int64  
 17  Musical             1682 non-null   int64  
 18  Mystery             1682 non-null   int64  
 19  Romance             1682 non-null   int64  
 20  Sci-Fi              1682 non-null   int64  
 21  Thriller            1682 non-null   int64  
 22  War                 1682 non-null   int64  
 23  Western             1682 non-null   int64  
dtypes: float64(1), int64(20), object(3)
memory usage: 315.5+ KB
```

In [12]:
```
# Dropping irrelevant columns
movies.drop(columns= ['video release date', 'IMDb URL'], inplace=True)  #drop unnec
```

In [13]:
```
# looking at the counts of individual genres
l = []

for i in movies.loc[:,'unknown' : 'Western'].columns:
    b = movies[i].value_counts()[1]
    l.append(b)


#print(movies.loc[:,'unknown' : 'Western'].columns)
#print(l)

# Create a new Dataframe and assign the genre values
genre_df = pd.DataFrame()
genre_df['Genre'] = movies.loc[:,'unknown' : 'Western'].columns
genre_df['Counts'] = l
genre_df
```

Out[13]:

	Genre	Counts
0	unknown	2
1	Action	251
2	Adventure	135
3	Animation	42
4	Children's	122
5	Comedy	505
6	Crime	109
7	Documentary	50
8	Drama	725
9	Fantasy	22
10	Film-Noir	24
11	Horror	92
12	Musical	56
13	Mystery	61
14	Romance	247
15	Sci-Fi	101
16	Thriller	251
17	War	71
18	Western	27

In [14]:
```
#the unknown column has only 2 entries. let's look at them
# filter in pandas to choose only movies for which unknown is 1 (or True)
# execution happens from right to left
movies[movies['unknown']== 1]
```

Out[14]:

	movie id	movie title	release date	unknown	Action	Adventure	Animation	Children's	Com
266	267	unknown	NaN	1	0	0	0	0	
1372	1373	Good Morning (1971)	4-Feb-1971	1	0	0	0	0	

2 rows × 22 columns

In [15]:
```
#let's see if there is information about the movieid 1373
ratings[ratings.ItemID == 1373]
```

Out[15]:

	UseID	ItemID	rating	Timestamp
8567	181	1373	1	878962052

In [16]:
```
# Since there is unknown column doesnt have a lot of info and the only movie associ
movies.drop(movies[movies['unknown'] == 1].index, axis=0, inplace=True)
movies.drop(columns= 'unknown',inplace=True)
```

In [17]:
```
movies.isna().apply(pd.value_counts)  #missing value check
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\2504064662.py:1: FutureWarning: pand
as.value_counts is deprecated and will be removed in a future version. Use pd.Series
(obj).value_counts() instead.
  movies.isna().apply(pd.value_counts)  #missing value check
```

Out[17]:

	movie id	movie title	release date	Action	Adventure	Animation	Children's	Comedy	Crime
False	1680	1680	1680	1680	1680	1680	1680	1680	1680

1 rows × 21 columns

In [18]:
```
# read the u.user data and name the columns as per the Readme document in the folde
users = pd.read_csv('C:/Users/asus/u.user',
                    sep = '|', names= ['UserID',  'Age', 'Gender', 'Occupation','Zi
print(users.shape)
users.head()
```

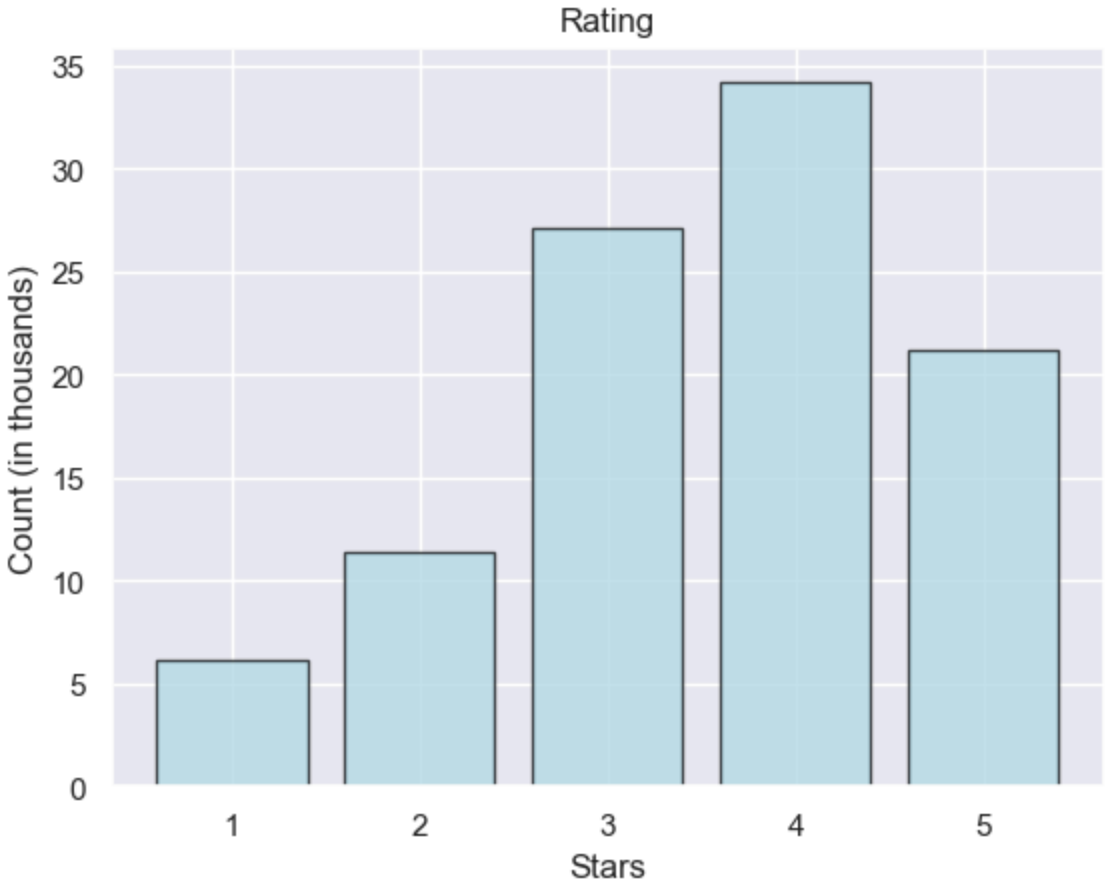
```
(943, 5)
```

Out[18]:

	UserID	Age	Gender	Occupation	Zip-code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

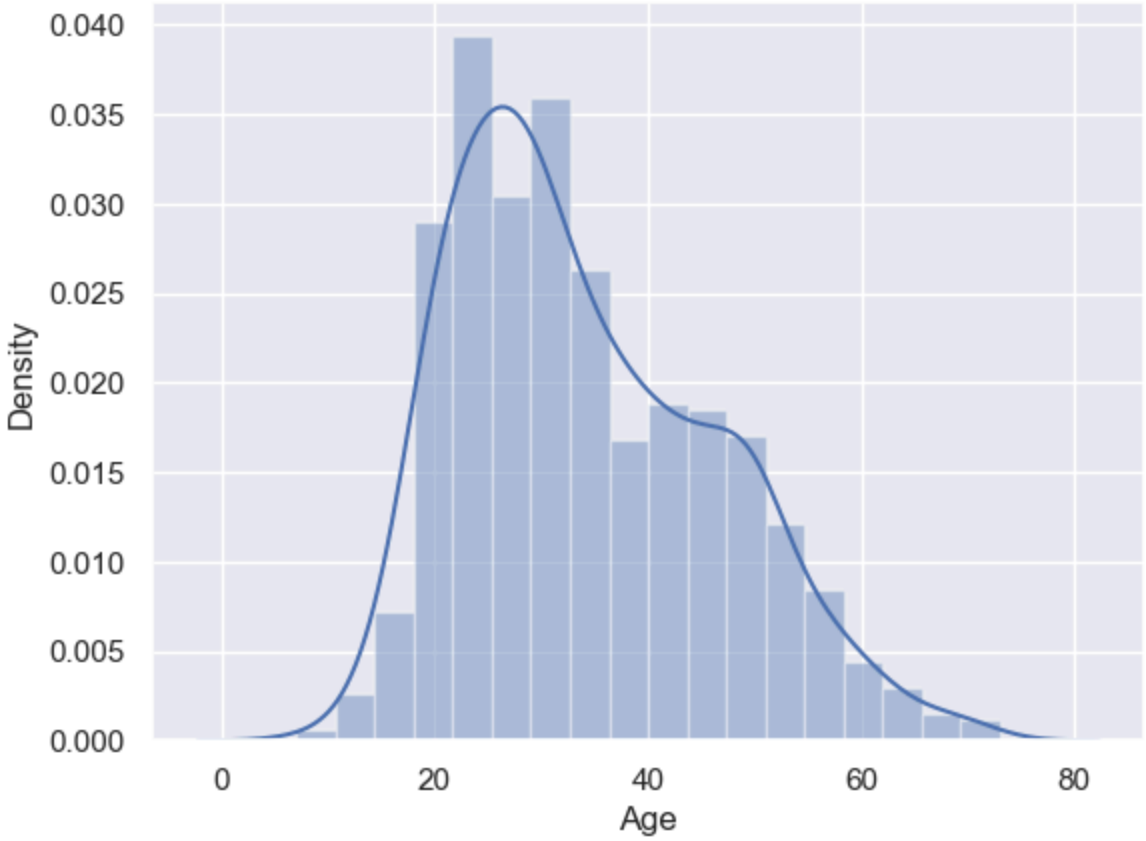
In [19]:
```
# Bar chart to view distribution of ratings accross all movies
x = ratings.rating.value_counts().index    #Values for x-axis
y = [ratings['rating'].value_counts()[i]/1000 for i in x]   #count(in thousands) on

# Other plot customizations
# For more use the link - https://matplotlib.org/contents.html
plt.bar(x,y, align='center',color = 'lightblue',edgecolor = 'black', alpha = 0.7)
plt.xlabel('Stars')
```

```
plt.ylabel('Count (in thousands)')
plt.title('Rating')
plt.savefig("plot01.png", dpi=300)
plt.show()
```



In [20]:
```
# Distribution of age
sns.set()    #setting seaborn style to default
sns.distplot(users.Age)
plt.show()
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\876738852.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(users.Age)
```

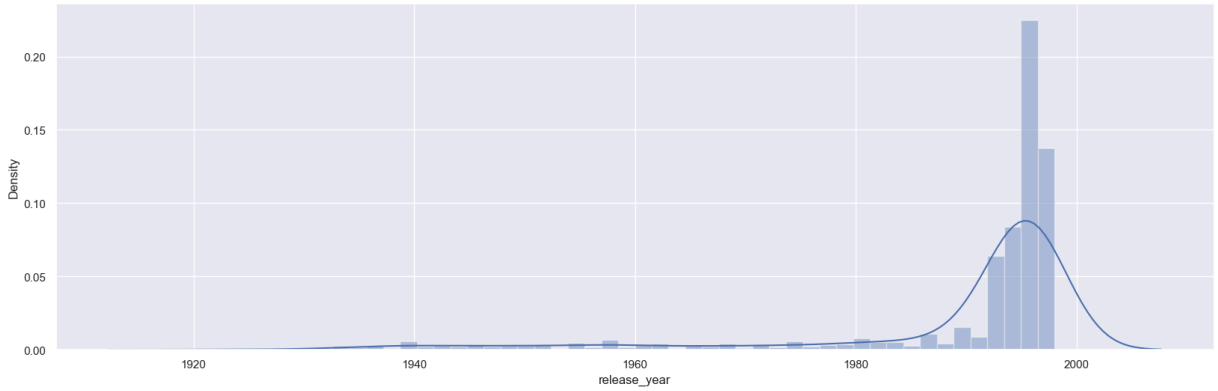


In [21]:
```
# Example line of what will happen below.
# For every value in release date, we convert to a string and split by -. 1 st valu
movies['release date'].head(1).str.split("-")[0][2]
```

Out[21]: '1995'

In [22]:
```
# Distribution of movies w.r.t release year
movies['release_year'] = movies['release date'].str.split('-', expand = True)[2]  #
movies['release_year'] = movies.release_year.astype(int)  # changing the type to in
plt.figure(figsize=(20,6))    #increasing the figure size
sns.distplot(movies.release_year)
plt.show()
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\4223481109.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(movies.release_year)
```

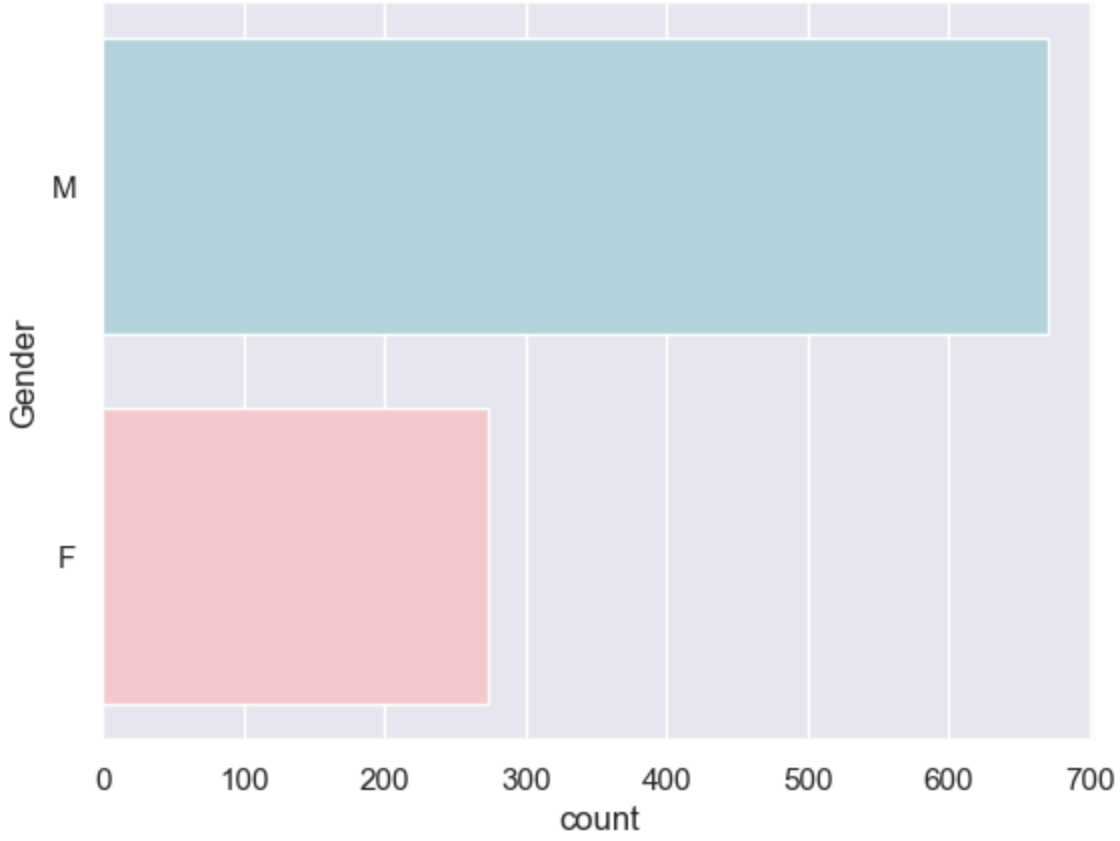


In [23]:
```
plt.figure(figsize=(15,6))
sns.countplot(users.Age)
```

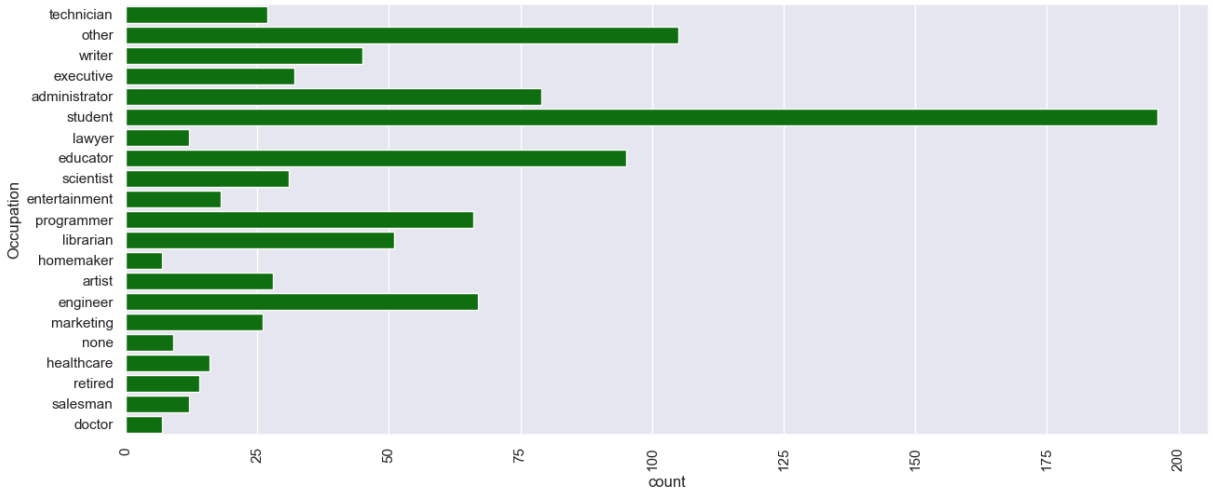
Out[23]: <Axes: ylabel='count'>

In [24]:
```
#Gender distribution

sns.countplot(users.Gender, palette=['lightblue','pink'])
plt.show()
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\536086039.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(users.Gender, palette=['lightblue','pink'])
```



In [25]:
```
#Distribution of users w.r.t population

plt.figure(figsize=(15,6)) # Customizing to show the plot neatly

sns.countplot(users.Occupation, color= 'green')
plt.yticks(rotation=0)
plt.xticks(rotation=90)
plt.show()
```



Visualize how popularity of genres has changed over the years

In [26]:
```
genre_by_year = movies.groupby('release_year').sum()
#genre_by_year
```

In [57]:
```
genre_by_year = movies.groupby('release_year').sum()
genre_by_year = genre_by_year.drop(columns = 'movie id').T
genre_by_year
```

Out[57]:

release_year	1922	1926	1930	1931	1932	1933	1934	
movie title	Nosferatu (Nosferatu, eine Symphonie des Graue...	Scarlet Letter, The (1926)	Blue Angel, The (Blaue Engel, Der) (1930)	M (1931)	Farewell to Arms, A (1932)	Duck Soup (1933)Liebelei (1933)	Thin Man, The (1934)It Happened One Night (193...	39 S (1935) Franken (
release date	01-Jan-1922	01-Jan-1926	01-Jan-1930	01-Jan-1931	01-Jan-1932	01-Jan-193301-Jan-1933	01-Jan-193401-Jan-193401-Jan-193401-Jan-1934	01 193501 193501 193501
Action	0	0	0	0	0	0	0	
Adventure	0	0	0	0	0	0	0	
Animation	0	0	0	0	0	0	0	
Children's	0	0	0	0	0	0	0	
Comedy	0	0	0	0	0	1	2	
Crime	0	0	0	1	0	0	0	
Documentary	0	0	0	0	0	0	0	
Drama	0	1	1	0	0	0	1	
Fantasy	0	0	0	0	0	0	0	
Film-Noir	0	0	0	1	0	0	0	
Horror	1	0	0	0	0	0	0	
Musical	0	0	0	0	0	0	1	
Mystery	0	0	0	0	0	0	1	
Romance	0	0	0	0	1	1	1	
Sci-Fi	0	0	0	0	0	0	0	
Thriller	0	0	0	1	0	0	0	
War	0	0	0	0	1	1	0	
Western	0	0	0	0	0	0	0	

20 rows × 71 columns

In [58]:
```
print(genre_by_year.info())
print(genre_by_year.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 20 entries, movie title to Western
Data columns (total 71 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   1922    20 non-null     object
 1   1926    20 non-null     object
 2   1930    20 non-null     object
 3   1931    20 non-null     object
 4   1932    20 non-null     object
 5   1933    20 non-null     object
 6   1934    20 non-null     object
 7   1935    20 non-null     object
 8   1936    20 non-null     object
 9   1937    20 non-null     object
 10  1938    20 non-null     object
 11  1939    20 non-null     object
 12  1940    20 non-null     object
 13  1941    20 non-null     object
 14  1942    20 non-null     object
 15  1943    20 non-null     object
 16  1944    20 non-null     object
 17  1945    20 non-null     object
 18  1946    20 non-null     object
 19  1947    20 non-null     object
 20  1948    20 non-null     object
 21  1949    20 non-null     object
 22  1950    20 non-null     object
 23  1951    20 non-null     object
 24  1952    20 non-null     object
 25  1953    20 non-null     object
 26  1954    20 non-null     object
 27  1955    20 non-null     object
 28  1956    20 non-null     object
 29  1957    20 non-null     object
 30  1958    20 non-null     object
 31  1959    20 non-null     object
 32  1960    20 non-null     object
 33  1961    20 non-null     object
 34  1962    20 non-null     object
 35  1963    20 non-null     object
 36  1964    20 non-null     object
 37  1965    20 non-null     object
 38  1966    20 non-null     object
 39  1967    20 non-null     object
 40  1968    20 non-null     object
 41  1969    20 non-null     object
 42  1970    20 non-null     object
 43  1971    20 non-null     object
 44  1972    20 non-null     object
 45  1973    20 non-null     object
 46  1974    20 non-null     object
 47  1975    20 non-null     object
 48  1976    20 non-null     object
 49  1977    20 non-null     object
 50  1978    20 non-null     object
```

```
 51  1979    20 non-null     object
 52  1980    20 non-null     object
 53  1981    20 non-null     object
 54  1982    20 non-null     object
 55  1983    20 non-null     object
 56  1984    20 non-null     object
 57  1985    20 non-null     object
 58  1986    20 non-null     object
 59  1987    20 non-null     object
 60  1988    20 non-null     object
 61  1989    20 non-null     object
 62  1990    20 non-null     object
 63  1991    20 non-null     object
 64  1992    20 non-null     object
 65  1993    20 non-null     object
 66  1994    20 non-null     object
 67  1995    20 non-null     object
 68  1996    20 non-null     object
 69  1997    20 non-null     object
 70  1998    20 non-null     object
dtypes: object(71)
memory usage: 11.2+ KB
None
release_year                                            1922  \
movie title   Nosferatu (Nosferatu, eine Symphonie des Graue...
release date                                       01-Jan-1922
Action                                                       0
Adventure                                                    0
Animation                                                    0

release_year                        1926  \
movie title   Scarlet Letter, The (1926)
release date                01-Jan-1926
Action                                0
Adventure                             0
Animation                             0

release_year                                      1930         1931  \
movie title   Blue Angel, The (Blaue Engel, Der) (1930)     M (1931)
release date                                01-Jan-1930  01-Jan-1931
Action                                                0            0
Adventure                                             0            0
Animation                                             0            0

release_year                        1932                             1933  \
movie title   Farewell to Arms, A (1932)  Duck Soup (1933)Liebelei (1933)
release date                01-Jan-1932          01-Jan-193301-Jan-1933
Action                                0                               0
Adventure                             0                               0
Animation                             0                               0

release_year                                            1934  \
movie title   Thin Man, The (1934)It Happened One Night (193...
release date      01-Jan-193401-Jan-193401-Jan-193401-Jan-1934
Action                                                       0
Adventure                                                    0
```

```
Animation                                                    0

release_year                                              1935  \
movie title   39 Steps, The (1935)Bride of Frankenstein (193...
release date      01-Jan-193501-Jan-193501-Jan-193501-Jan-1935
Action                                                       0
Adventure                                                    0
Animation                                                    0

release_year                                              1936  \
movie title   My Man Godfrey (1936)Little Lord Fauntleroy (1...
release date                            01-Jan-193601-Jan-1936
Action                                                       0
Adventure                                                    0
Animation                                                    0

release_year                                              1937  ...  \
movie title   Snow White and the Seven Dwarfs (1937)Lost Hor...  ...
release date      01-Jan-193701-Jan-193701-Jan-193701-Jan-1937  ...
Action                                                       0  ...
Adventure                                                    0  ...
Animation                                                    1  ...

release_year                                              1989  \
movie title   Weekend at Bernie's (1989)Abyss, The (1989)Hen...
release date  01-Jan-198901-Jan-198901-Jan-198901-Jan-198901...
Action                                                       6
Adventure                                                    4
Animation                                                    0

release_year                                              1990  \
movie title   Home Alone (1990)Dances with Wolves (1990)Good...
release date  01-Jan-199001-Jan-199001-Jan-199001-Jan-199001...
Action                                                       7
Adventure                                                    1
Animation                                                    0

release_year                                              1991  \
movie title   Terminator 2: Judgment Day (1991)Silence of th...
release date  01-Jan-199101-Jan-199101-Jan-199101-Jan-199101...
Action                                                       2
Adventure                                                    1
Animation                                                    1

release_year                                              1992  \
movie title   Aladdin (1992)Lawnmower Man, The (1992)Reservo...
release date  01-Jan-199201-Jan-199201-Jan-199201-Jan-199201...
Action                                                       8
Adventure                                                    1
Animation                                                    2

release_year                                              1993  \
movie title   Three Colors: Blue (1993)What's Eating Gilbert...
release date  01-Jan-199301-Jan-199301-Jan-199301-Jan-199301...
Action                                                      20
Adventure                                                    9
```

```
Animation                                                     1

release_year                                               1994  \
movie title   Postino, Il (1994)Crumb (1994)Nadja (1994)Cler...
release date  01-Jan-199401-Jan-199401-Jan-199401-Jan-199401...
Action                                                       30
Adventure                                                    13
Animation                                                     4

release_year                                               1995  \
movie title   Toy Story (1995)GoldenEye (1995)Four Rooms (19...
release date  01-Jan-199501-Jan-199501-Jan-199501-Jan-199501...
Action                                                       40
Adventure                                                    22
Animation                                                     6

release_year                                               1996  \
movie title   Richard III (1995)Mr. Holland's Opus (1995)Fro...
release date  22-Jan-199629-Jan-199605-Feb-199616-Feb-199616...
Action                                                       44
Adventure                                                    24
Animation                                                     9

release_year                                               1997  \
movie title   Fargo (1996)Return of the Jedi (1983)Kolya (19...
release date  14-Feb-199714-Mar-199724-Jan-199707-Mar-199714...
Action                                                       46
Adventure                                                    20
Animation                                                     3

release_year                                               1998
movie title   Apt Pupil (1998)Desperate Measures (1998)Wag t...
release date  23-Oct-199830-Jan-199809-Jan-199830-Jan-199816...
Action                                                       12
Adventure                                                     3
Animation                                                     0

[5 rows x 71 columns]
```

In [59]:
```
print(genre_by_year.columns)
```

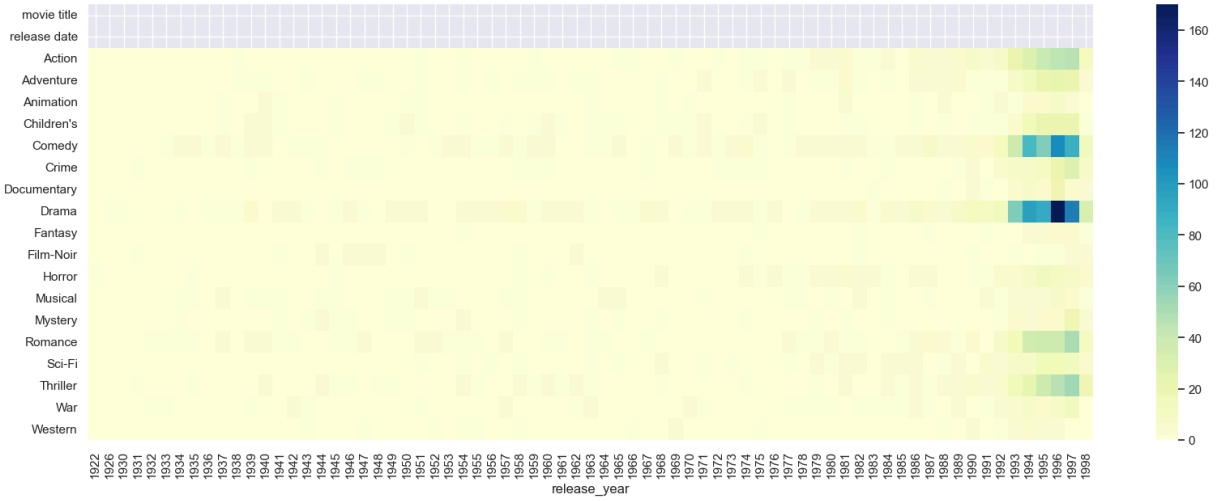
```
Index([1922, 1926, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939,
       1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951,
       1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963,
       1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975,
       1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987,
       1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998],
      dtype='int32', name='release_year')
```

In [60]:
```
# Attempt conversion to ensure numeric data if it's in string format
genre_by_year = genre_by_year.apply(pd.to_numeric, errors='coerce')

# there are numeric columns, but the select_dtypes function isn’t working as expect
#so I isolating numeric columns manually by specifying column names.
#For example, if genre_by_year should have columns for years and counts of genres,
#I convert them explicitly to numeric:
```

In [61]:
```
# Group by 'release_year' and sum the values
#genre_by_year = movies.groupby('release_year').sum()

# Filter to include only rows where 'release_year' is between 1980 and 1998
#genre_by_year = genre_by_year[(genre_by_year.index >= 1985) & (genre_by_year.index

# Drop the 'movie id' column, if it exists
#genre_by_year = genre_by_year.drop(columns='movie id', errors='ignore').T

# Display the DataFrame
#genre_by_year
```

In [62]:
```
# Attempt conversion to ensure numeric data if it's in string format
genre_by_year = genre_by_year.apply(pd.to_numeric, errors='coerce')

# there are numeric columns, but the select_dtypes function isn't working as expect
#so I isolating numeric columns manually by specifying column names.
#For example, if genre_by_year should have columns for years and counts of genres,
#I convert them explicitly to numeric:
```

In [63]:
```
# Filter only the numeric columns and plot
genre_by_year_numeric = genre_by_year.select_dtypes(include=['number'])


plt.figure(figsize=(20,7))  #setting the figure size
sns.heatmap(genre_by_year_numeric, cmap='YlGnBu')
plt.savefig("plot02.png", dpi=300)
plt.show()
```



In [64]:
```
print(genre_by_year_numeric.isnull().sum())  # Check if all values are NaN
print(genre_by_year_numeric.shape)           # Check dimensions
```

```
release_year
1922    2
1926    2
1930    2
1931    2
1932    2
       ..
1994    2
1995    2
1996    2
1997    2
1998    2
Length: 71, dtype: int64
(20, 71)
```

Find the top 25 movies, each having a minimum of 100 ratings

In [65]:
```
items = ratings.groupby('ItemID').count()
items.head()
```

Out[65]:

ItemID	UseID	rating	Timestamp
1	452	452	452
2	131	131	131
3	90	90	90
4	209	209	209
5	86	86	86

In [73]:
```
items = ratings.groupby('ItemID').count()

items = items[ratings.groupby('ItemID').count().UseID > 100].index

items = ratings.loc[ratings.ItemID.isin(items)]

items = items.groupby('ItemID').mean()

items = items.sort_values('rating',ascending = False)

order = items.index
```

In [74]:
```
rating_list = items.rating[0:25]
```

In [75]:
```
movies.columns
items = movies.loc[movies['movie id'].isin(order)]
```

In [76]:
```
top_25_movies = items.set_index('movie id').loc[order]

top_25_movies = top_25_movies.iloc[0:25, 0]
```

In [77]:
```
top_25_movies = top_25_movies.reset_index()
```

In [78]:
```
top_25_movies['avg_rating'] =  rating_list.values
```

In [79]:
```
top_25_movies
```

Out[79]:

	ItemID	movie title	avg_rating
0	408	Close Shave, A (1995)	4.491071
1	318	Schindler's List (1993)	4.466443
2	169	Wrong Trousers, The (1993)	4.466102
3	483	Casablanca (1942)	4.456790
4	64	Shawshank Redemption, The (1994)	4.445230
5	603	Rear Window (1954)	4.387560
6	12	Usual Suspects, The (1995)	4.385768
7	50	Star Wars (1977)	4.358491
8	178	12 Angry Men (1957)	4.344000
9	134	Citizen Kane (1941)	4.292929
10	427	To Kill a Mockingbird (1962)	4.292237
11	357	One Flew Over the Cuckoo's Nest (1975)	4.291667
12	98	Silence of the Lambs, The (1991)	4.289744
13	480	North by Northwest (1959)	4.284916
14	127	Godfather, The (1972)	4.283293
15	285	Secrets & Lies (1996)	4.265432
16	272	Good Will Hunting (1997)	4.262626
17	657	Manchurian Candidate, The (1962)	4.259542
18	474	Dr. Strangelove or: How I Learned to Stop Worr...	4.252577
19	174	Raiders of the Lost Ark (1981)	4.252381
20	479	Vertigo (1958)	4.251397
21	313	Titanic (1997)	4.245714
22	511	Lawrence of Arabia (1962)	4.231214
23	484	Maltese Falcon, The (1941)	4.210145
24	172	Empire Strikes Back, The (1980)	4.204360

In [80]:
```
top_25_movies.to_excel("table2.xlsx", index=False)
```

See gender distribution across different genres Verify the following:

Men watch more drama than women Women watch more Sci-Fi than men Men watch more Romance than women

In [81]:
```
movies.columns  # columns of movies dataframe
users.columns   #columns of users dataframe
ratings.columns  # columns of ratings dataframe
```

Out[81]:
```
Index(['UseID', 'ItemID', 'rating', 'Timestamp'], dtype='object')
```

In [82]:
```
ratings.rename(columns= {'UseID' : 'UserID'}, inplace=True)
#Renaming the column to have the same column name so that we can perform join opera
```

In [129…
```
result = pd.merge(ratings, users, how='inner', on='UserID')  #merging dataframes wi

movies.rename(columns= {'movie id' : 'ItemID'}, inplace= True)
#Renaming the column to have the same column name so that we can perform join opera

result = pd.merge(result, movies, how='inner', on='ItemID')  # merging dataframes w
```

In [130…
```
result.head()
```

Out[130…

	UserID	ItemID	rating	Timestamp	Age	Gender	Occupation	Zip-code	movie title	rel
0	196	242	3	881250949	49	M	writer	55105	Kolya (1996)	
1	186	302	3	891717742	39	F	executive	00000	L.A. Confidential (1997)	
2	22	377	1	878887116	25	M	writer	40206	Heavyweights (1994)	
3	244	51	2	880606923	28	M	technician	80525	Legends of the Fall (1994)	
4	166	346	1	886397596	47	M	educator	55113	Jackie Brown (1997)	

5 rows × 29 columns

In [85]:
```
Genre_by_gender = result.groupby('Gender').sum().loc[:,'Action':'Western']  #Group

Genre_by_gender['total'] = Genre_by_gender.sum(axis = 1)  #Row total of the datafra
```

```
Genre_by_gender
```

Out[85]:

Gender	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama
F	5442	3141	995	2232	8068	1794	187	11008
M	20147	10612	2610	4950	21764	6261	571	28887

In [61]:
```
"""In absolute numbers,

Men watch more drama than women - True, 28887 men watch more drama as compared to 1
Men watch more Romance than women - True, 13603 men watch more romance than 5858 wo
Women watch more Sci-Fi than men - False, 2629 woman watch less Sci-Fi as compared
```

Out[61]:
```
'In absolute numbers,\n\nMen watch more drama than women - True, 28887 men watch m
ore drama as compared to 11008 woman\nMen watch more Romance than women - True, 13
603 men watch more romance than 5858 woman\nWomen watch more Sci-Fi than men - Fal
se, 2629 woman watch less Sci-Fi as compared to 10101 men'
```

Let's try percentages

In [86]:
```
Genre_by_gender.div(Genre_by_gender.total, axis= 0) * 100  #dividing each cell with
```

Out[86]:

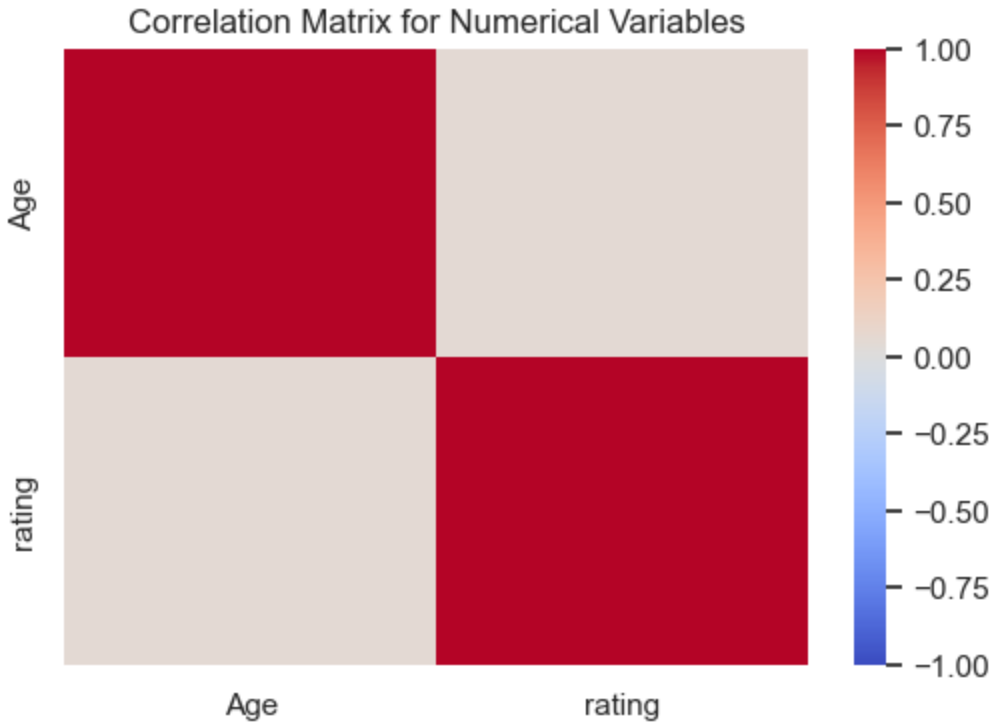
Gender	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary
F	10.133889	5.849053	1.852852	4.156347	15.023929	3.340720	0.348224
M	12.680320	6.679087	1.642708	3.115480	13.698044	3.940611	0.359382

In [ ]:
```
"""Conclusion: In Percentages

Of all the women, 20% watched Drama and of all the men, 18% watched Drama
Of all men, 8% watched romance whereas nearly 10% of woman have watched romance.
Of all the women, 4.8% watched Sci-Fi and of all the men, 6.3% watched Sci-Fi"""
```

In [125…
```
numerical_df = result[['Age', 'rating']]

# Calculate correlation matrix
correlation_matrix = numerical_df.corr()
print("Correlation matrix:\n", correlation_matrix)

# Visualize the correlation matrix as a heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Correlation Matrix for Numerical Variables")
```

```
plt.savefig("plot03.png", dpi=300)
plt.show()
```

```
Correlation matrix:
              Age    rating
Age      1.000000  0.054462
rating   0.054462  1.000000
```



In [126…

```
print(result.columns)
```

```
Index(['UserID', 'ItemID', 'rating', 'Timestamp', 'Age', 'Gender',
       'Occupation', 'Zip-code', 'movie title', 'release date', 'Action',
       'Adventure', 'Animation', 'Children's', 'Comedy', 'Crime',
       'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
       'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western',
       'release_year'],
      dtype='object')
```

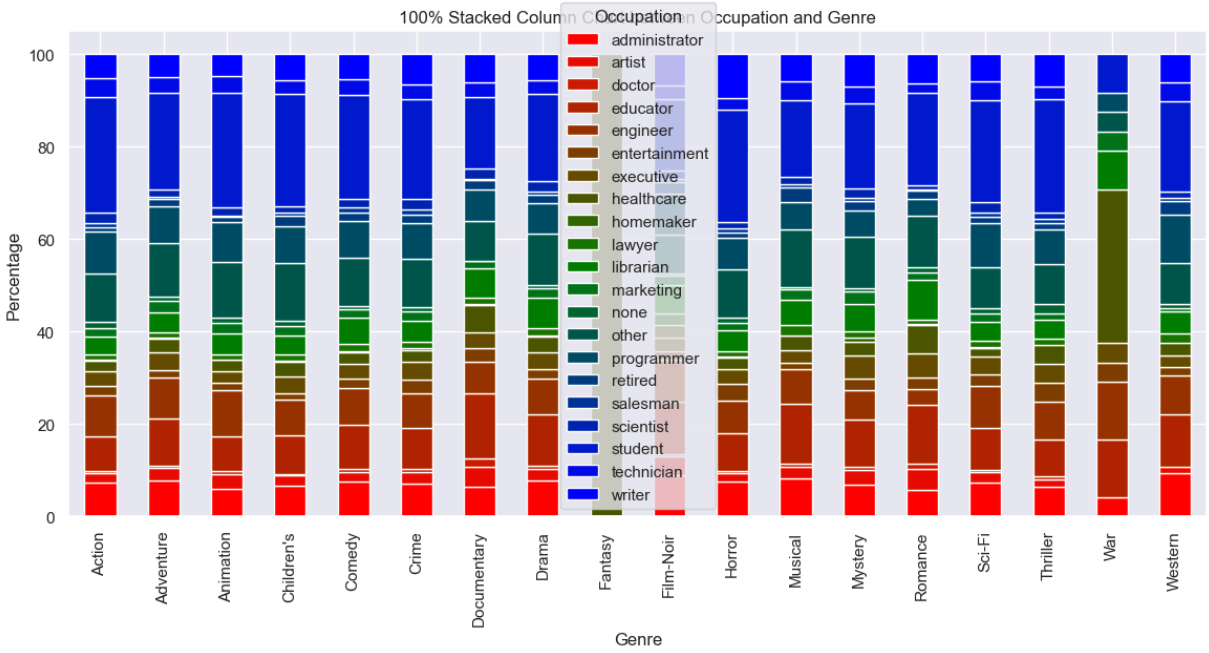
In [143…

```
from matplotlib.colors import LinearSegmentedColormap

# Extract genre columns and sum them row-wise to create a 'Genre' column
genre_df = result.loc[:, 'Action':'Western']
result['Genre'] = genre_df.idxmax(axis=1)  # Finds the genre with the highest count

# Now create a contingency table with Gender and computed Genre
contingency_table = pd.crosstab(result['Occupation'], result['Genre'])
print("Contingency table between Occupation and Genre:\n", contingency_table)

# Optional: Visualize the contingency table as a heatmap
import seaborn as sns
import matplotlib.pyplot as plt


# Assuming `contingency_table` is your original table
# Convert each column to a percentage by dividing by the column sum
```

```
contingency_table_percentage = contingency_table.div(contingency_table.sum(axis=0),

custom_cmap = LinearSegmentedColormap.from_list("custom_cmap", ["red", "green", "bl

# Plot a 100% stacked bar chart
contingency_table_percentage.T.plot(kind='bar', stacked=True, figsize=(14, 6), cmap

# Add plot labels
plt.title("100% Stacked Column Chart between Occupation and Genre")
plt.ylabel("Percentage")
plt.xlabel("Genre")
plt.legend(title="Occupation")
plt.savefig("plot05.png", dpi=300)
plt.show()
```

```
Contingency table between Occupation and Genre:
 Genre          Action  Adventure  Animation  Children's  Comedy  Crime  \
Occupation
administrator    1858        266        191         170    1742    354
artist            528         94        108          56     461    132
doctor            110         16         20           6     138     26
educator         1962        353        243         217    2166    450
engineer         2277        307        324         197    1816    371
entertainment     499         60         53          34     462    144
executive         808        130         80          94     724    194
healthcare        577         98         76          80     574    128
homemaker          92         10          7          10      71     19
lawyer            283         42         32          34     358     65
librarian         988        146        146         102    1288    229
marketing         477         88         73          53     410    106
none              301         29         38          31     181     47
other            2672        398        390         319    2386    514
programmer       2322        274        281         204    1787    390
retired           278         58         34          55     422     90
salesman          222         15         13          19     251     52
scientist         566         59         56          34     413    118
student          6398        716        804         623    5155   1080
technician       1079        119        114          77     781    155
writer           1292        170        153         142    1210    324

Genre          Documentary  Drama  Fantasy  Film-Noir  Horror  Musical  \
Occupation
administrator           48   2130        0         91     222       61
artist                  34    654        0         41      58       18
doctor                  13    165        0          4      10        4
educator               107   3012        0        115     242       95
engineer                51   2118        0         92     205       55
entertainment           23    544        0         19     111        9
executive               26   1000        0         30      91       21
healthcare              44    932        1         28      77       22
homemaker                2     68        0          0       4        1
lawyer                  10    375        0         25      34       16
librarian               48   1838        0         61     134       40
marketing               12    556        0         21      47       17
none                     1    175        0          5      34        3
other                   65   2963        0         85     305       91
programmer              51   1794        0         91     200       43
retired                 16    485        0         24      35       23
salesman                 2    206        0          7      29        5
scientist               17    643        0         18      37       11
student                117   5050        0        157     716      122
technician              24    852        0         29      75       29
writer                  46   1490        0         69     280       43

Genre          Mystery  Romance  Sci-Fi  Thriller  War  Western
Occupation
administrator      111       24      86        70    1       54
artist              48       19      29        19    0        9
doctor              12        5       5         6    0        0
educator           163       53     107        87    3       66
engineer           101       14     106        89    3       48
```

entertainment	39	10	30	45	1	11
executive	77	22	45	45	1	14
healthcare	50	25	22	45	8	17
homemaker	11	1	3	0	0	0
lawyer	25	4	15	16	0	11
librarian	93	36	50	44	2	28
marketing	42	7	21	15	1	4
none	11	4	13	22	0	5
other	176	47	104	94	1	52
programmer	89	15	115	82	1	61
retired	34	7	16	15	0	17
salesman	10	1	11	10	0	3
scientist	33	4	25	16	0	8
student	291	83	261	267	2	114
technician	59	9	47	32	0	24
writer	110	26	70	75	0	35



In [144…

```
from matplotlib.colors import LinearSegmentedColormap

# Extract genre columns and sum them row-wise to create a 'Genre' column
genre_df = result.loc[:, 'Action':'Western']
result['Genre'] = genre_df.idxmax(axis=1)  # Finds the genre with the highest count

# Now create a contingency table with Gender and computed Genre
contingency_table = pd.crosstab(result['Occupation'], result['Genre'])
print("Contingency table between Occupation and Genre:\n", contingency_table)

# Optional: Visualize the contingency table as a heatmap
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(10, 10))
sns.heatmap(contingency_table, annot=False, cmap="YlGnBu", fmt="d")
plt.title("Contingency Table: Occupation vs. Genre")
plt.xlabel("Genre")
```

```
plt.ylabel("Occupation")
plt.savefig("plot06.png", dpi=300)
plt.show()
```

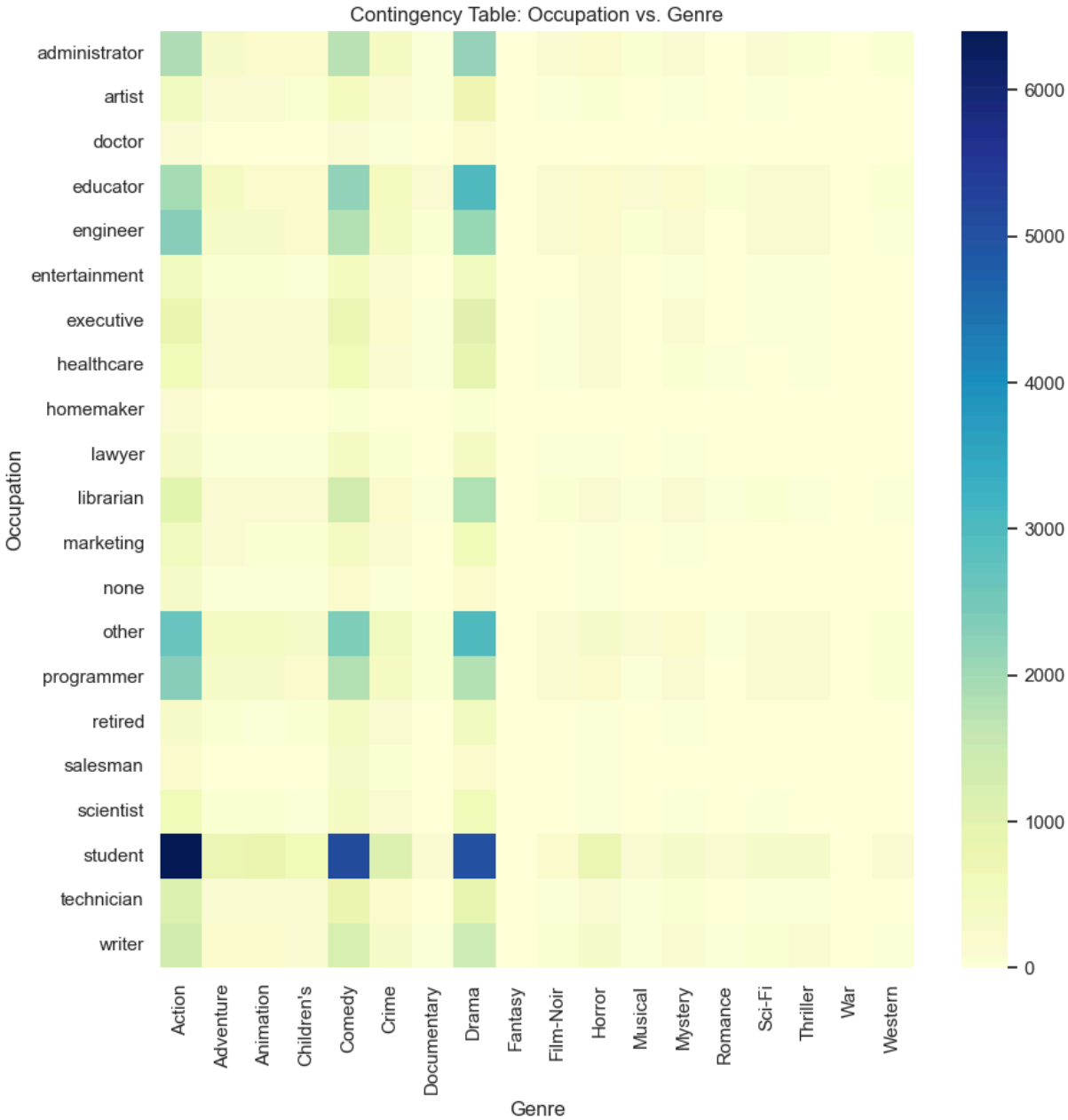
Contingency table between Occupation and Genre:

Occupation	Action	Adventure	Animation	Children's	Comedy	Crime
administrator	1858	266	191	170	1742	354
artist	528	94	108	56	461	132
doctor	110	16	20	6	138	26
educator	1962	353	243	217	2166	450
engineer	2277	307	324	197	1816	371
entertainment	499	60	53	34	462	144
executive	808	130	80	94	724	194
healthcare	577	98	76	80	574	128
homemaker	92	10	7	10	71	19
lawyer	283	42	32	34	358	65
librarian	988	146	146	102	1288	229
marketing	477	88	73	53	410	106
none	301	29	38	31	181	47
other	2672	398	390	319	2386	514
programmer	2322	274	281	204	1787	390
retired	278	58	34	55	422	90
salesman	222	15	13	19	251	52
scientist	566	59	56	34	413	118
student	6398	716	804	623	5155	1080
technician	1079	119	114	77	781	155
writer	1292	170	153	142	1210	324

Occupation	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical
administrator	48	2130	0	91	222	61
artist	34	654	0	41	58	18
doctor	13	165	0	4	10	4
educator	107	3012	0	115	242	95
engineer	51	2118	0	92	205	55
entertainment	23	544	0	19	111	9
executive	26	1000	0	30	91	21
healthcare	44	932	1	28	77	22
homemaker	2	68	0	0	4	1
lawyer	10	375	0	25	34	16
librarian	48	1838	0	61	134	40
marketing	12	556	0	21	47	17
none	1	175	0	5	34	3
other	65	2963	0	85	305	91
programmer	51	1794	0	91	200	43
retired	16	485	0	24	35	23
salesman	2	206	0	7	29	5
scientist	17	643	0	18	37	11
student	117	5050	0	157	716	122
technician	24	852	0	29	75	29
writer	46	1490	0	69	280	43

Occupation	Mystery	Romance	Sci-Fi	Thriller	War	Western
administrator	111	24	86	70	1	54
artist	48	19	29	19	0	9
doctor	12	5	5	6	0	0
educator	163	53	107	87	3	66
engineer	101	14	106	89	3	48

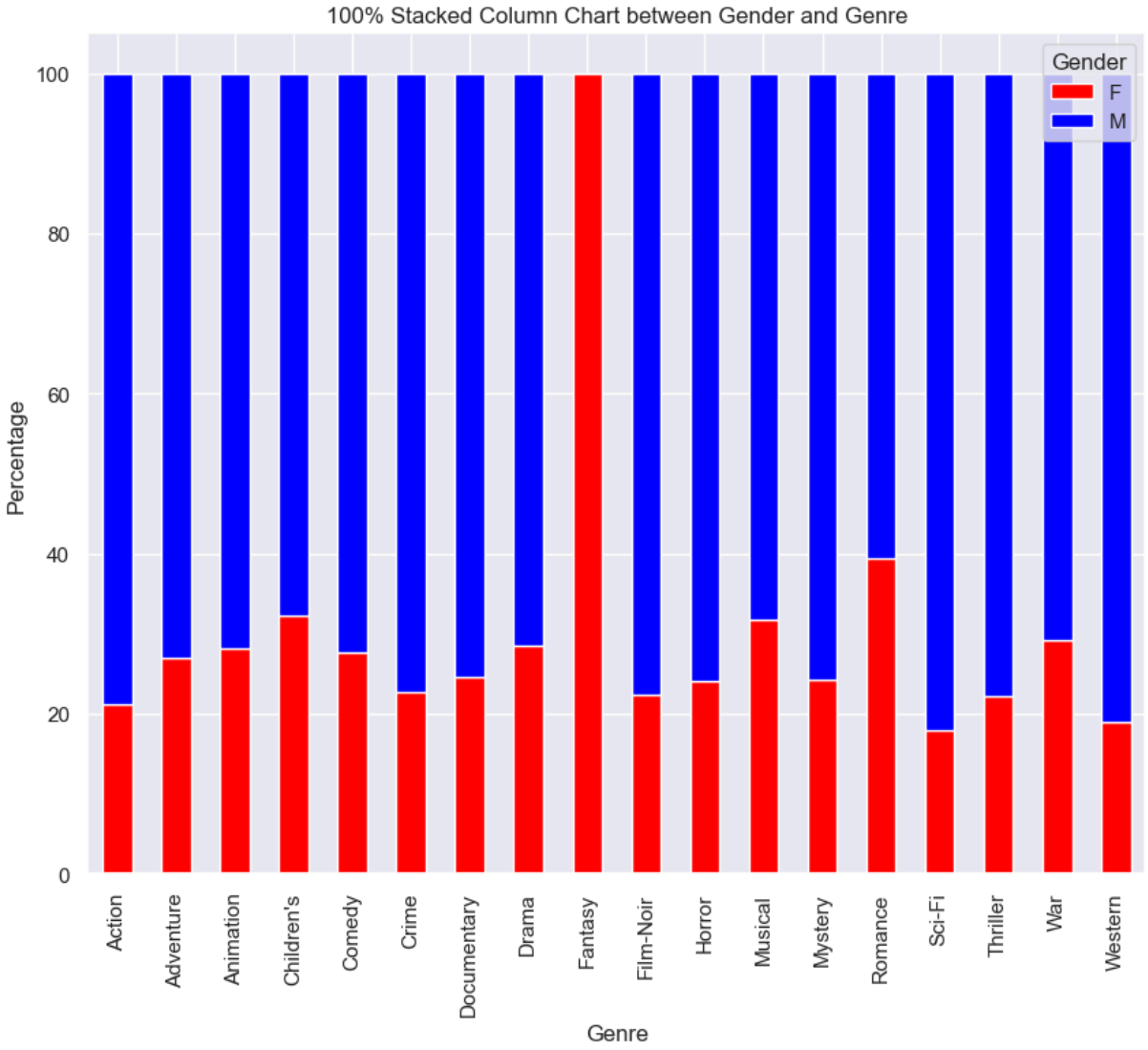
entertainment	39	10	30	45	1	11
executive	77	22	45	45	1	14
healthcare	50	25	22	45	8	17
homemaker	11	1	3	0	0	0
lawyer	25	4	15	16	0	11
librarian	93	36	50	44	2	28
marketing	42	7	21	15	1	4
none	11	4	13	22	0	5
other	176	47	104	94	1	52
programmer	89	15	115	82	1	61
retired	34	7	16	15	0	17
salesman	10	1	11	10	0	3
scientist	33	4	25	16	0	8
student	291	83	261	267	2	114
technician	59	9	47	32	0	24
writer	110	26	70	75	0	35



In [145…

```
from matplotlib.colors import LinearSegmentedColormap

# Extract genre columns and sum them row-wise to create a 'Genre' column
genre_df = result.loc[:, 'Action':'Western']
result['Genre'] = genre_df.idxmax(axis=1)  # Finds the genre with the highest count

# Now create a contingency table with Gender and computed Genre
contingency_table = pd.crosstab(result['Gender'], result['Genre'])
print("Contingency table between Gender and Genre:\n", contingency_table)

# Optional: Visualize the contingency table as a heatmap
import seaborn as sns
import matplotlib.pyplot as plt

#plt.figure(figsize=(10, 8))
#sns.heatmap(contingency_table, annot=False, fmt="d", cmap="YlGnBu")
#plt.title("Contingency Table between Gender and Genre")
#plt.ylabel("Gender")
#plt.xlabel("Genre")
#plt.show()




# Assuming `contingency_table` is your original table
# Convert each column to a percentage by dividing by the column sum
contingency_table_percentage = contingency_table.div(contingency_table.sum(axis=0),

custom_cmap = LinearSegmentedColormap.from_list("custom_cmap", ["red", "green", "bl

# Plot a 100% stacked bar chart
contingency_table_percentage.T.plot(kind='bar', stacked=True, figsize=(10, 8), cmap

# Add plot labels
plt.title("100% Stacked Column Chart between Gender and Genre")
plt.ylabel("Percentage")
plt.xlabel("Genre")
plt.legend(title="Gender")
plt.savefig("plot07.png", dpi=300)
plt.show()
```

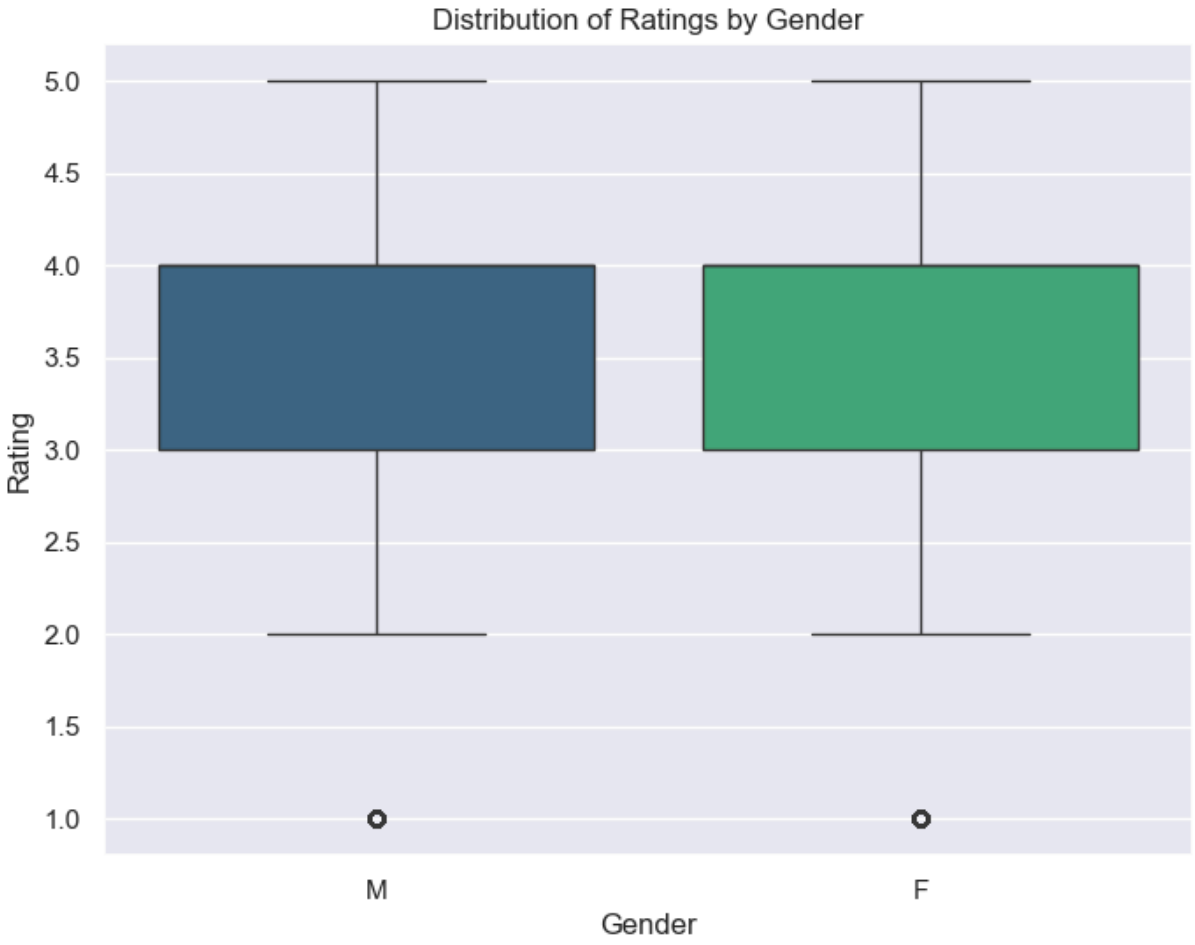
```
Contingency table between Gender and Genre:
 Genre   Action  Adventure  Animation  Children's  Comedy  Crime  Documentary  \
Gender
F         5442        928        910         824    6301   1135          186
M        20147       2520       2326        1733   16495   3853          571

Genre   Drama  Fantasy  Film-Noir  Horror  Musical  Mystery  Romance  Sci-Fi  \
Gender
F        7724        1        226     708      231      384      164     213
M       19326        0        786    2238      498     1201      252     968

Genre   Thriller  War  Western
Gender
F            244    7      110
M            850   17      471
```



In [146…

```
# Box plot to compare rating by Gender
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='rating', data=result, palette="viridis")
plt.title("Distribution of Ratings by Gender")
plt.xlabel("Gender")
plt.ylabel("Rating")
plt.savefig("plot08.png", dpi=300)
plt.show()
```

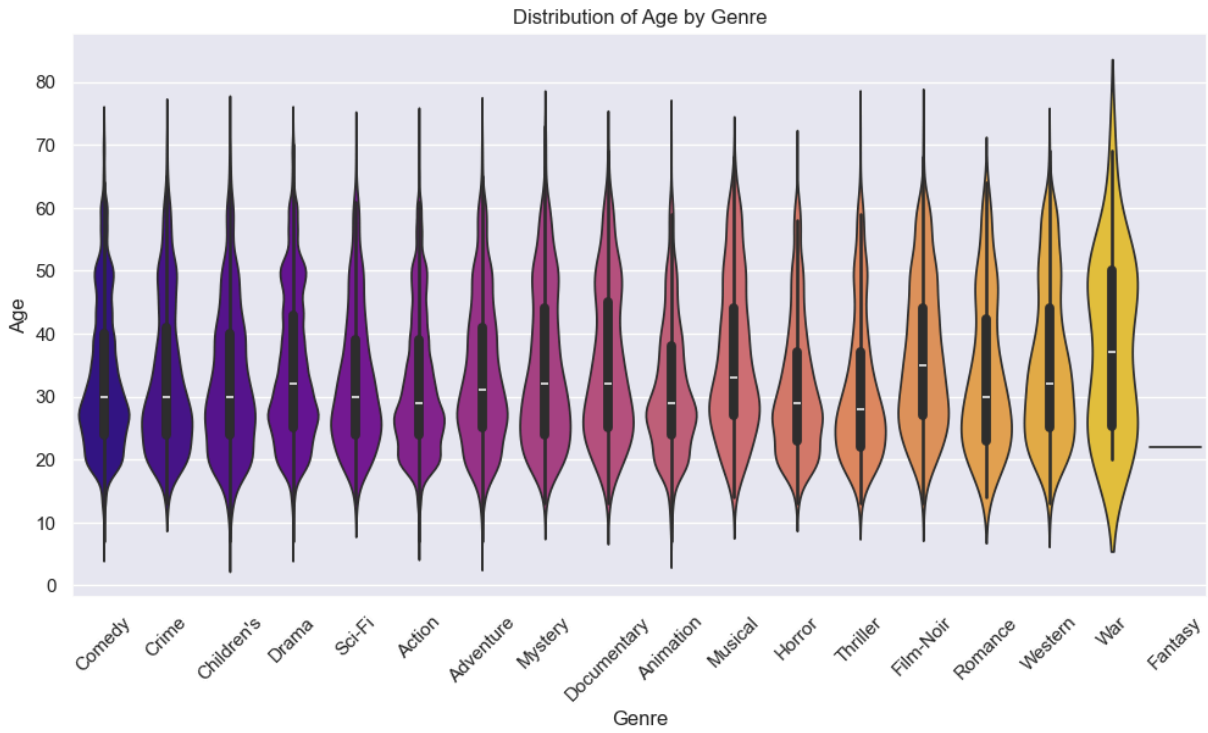
```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\19688611.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Gender', y='rating', data=result, palette="viridis")
```



In [147…

```
# Violin plot to compare Age distributions across different Genre categories
plt.figure(figsize=(12, 6))
sns.violinplot(x='Genre', y='Age', data=result, palette="plasma")
plt.title("Distribution of Age by Genre")
plt.xlabel("Genre")
plt.ylabel("Age")
plt.xticks(rotation=45)
plt.savefig("plot09.png", dpi=300)
plt.show()
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_2584\3372776160.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(x='Genre', y='Age', data=result, palette="plasma")
```



In [149…

```
# Mean and Standard Deviation of rating by Gender
rating_stats_by_gender = result.groupby('Gender')['rating'].agg(['mean', 'std', 'me
print("Rating statistics by Gender:\n", rating_stats_by_gender)

# Mean and Standard Deviation of Age by Genre
age_stats_by_genre = result.groupby('Genre')['Age'].agg(['mean', 'std', 'median', '
print("Age statistics by Genre:\n", age_stats_by_genre)
```

```
Rating statistics by Gender:
             mean       std  median  min  max
Gender
F       3.531510  1.170989     4.0    1    5
M       3.529333  1.109517     4.0    1    5
Age statistics by Genre:
                   mean        std  median  min  max
Genre
Action        31.709524  11.012644    29.0    7   73
Adventure     33.564675  11.572022    31.0    7   73
Animation     31.159147  10.405994    29.0    7   73
Children's    32.319906  11.532975    30.0    7   73
Comedy        32.819968  11.523951    30.0    7   73
Crime         32.830393  11.880500    30.0   13   73
Documentary   34.960370  12.072138    32.0   13   69
Drama         34.402699  11.959617    32.0    7   73
Fantasy       22.000000        NaN    22.0   22   22
Film-Noir     36.116601  11.738337    35.0   13   73
Horror        31.189409  10.711343    29.0   13   68
Musical       35.854595  12.147479    33.0   14   68
Mystery       34.659937  12.224068    32.0   13   73
Romance       33.218750  12.120191    30.0   14   64
Sci-Fi        32.475021  10.819416    30.0   13   70
Thriller      31.217550  11.409705    28.0   13   73
War           38.125000  13.762939    37.0   20   69
Western       35.013769  12.232651    32.0   13   69
```

In [150…
```
rating_stats_by_gender.to_excel("table2.xlsx", index=False)
```

In [151…
```
age_stats_by_genre.to_excel("table3.xlsx", index=False)
```

In [ ]:
```
numericcoloumns = data_processed.select_dtypes(include= ['number'])
numericcoloumns.head()
correlationmatrix = numericcoloumns.corr()
plt.figure(figsize=(8, 5))
sns.heatmap(data = data_processed , annot=True, cmap='coolwarm', fmt='.2f', cbar_kw
plt.title('Correlation Matrix of Data Set')
plt.show()
```