Project - MovieLens Data Analysis

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. The data is widely used for collaborative filtering and other filtering solutions. However, we will be using this data to act as a means to demonstrate our skill in using Python to “play” with data.

Objective:

- To implement the techniques learnt as a part of the course.

Datasets Information:

rating.csv: It contains information on ratings given by the users to a particular movie.

- user id: id assigned to every user
- movie id: id assigned to every movie
- rating: rating given by the user
- timestamp: Time recorded when the user gave a rating

movie.csv: File contains information related to the movies and their genre.

- movie id: id assigned to every movie
- movie title: Title of the movie
- release date: Date of release of the movie
- Action: Genre containing binary values (1 - for action 0 - not action)
- Adventure: Genre containing binary values (1 - for adventure 0 - not adventure)
- Animation: Genre containing binary values (1 - for animation 0 - not animation)
- Children’s: Genre containing binary values (1 - for children's 0 - not children's)
- Comedy: Genre containing binary values (1 - for comedy 0 - not comedy)
- Crime: Genre containing binary values (1 - for crime 0 - not crime)
- Documentary: Genre containing binary values (1 - for documentary 0 - not documentary)
- Drama: Genre containing binary values (1 - for drama 0 - not drama)
- Fantasy: Genre containing binary values (1 - for fantasy 0 - not fantasy)
- Film-Noir: Genre containing binary values (1 - for film-noir 0 - not film-noir)
- Horror: Genre containing binary values (1 - for horror 0 - not horror)
- Musical: Genre containing binary values (1 - for musical 0 - not musical)
- Mystery: Genre containing binary values (1 - for mystery 0 - not mystery)
- Romance: Genre containing binary values (1 - for romance 0 - not romance)
- Sci-Fi: Genre containing binary values (1 - for sci-fi 0 - not sci-fi)
- Thriller: Genre containing binary values (1 - for thriller 0 - not thriller)
- War: Genre containing binary values (1 - for war 0 - not war)
- Western: Genre containing binary values (1 - for western - not western)

user.csv: It contains information of the users who have rated the movies.

- user id: id assigned to every user
- age: Age of the user
- gender: Gender of the user
- occupation: Occupation of the user
- zip code: Zip code of the use

Please provide your insights wherever necessary.

Learning Outcomes:

- Exploratory Data Analysis
- Visualization using Python
- Pandas – groupby, merging

Domain

- Internet and Entertainment

Note that the project will need you to apply the concepts of groupby and merging extensively.

1. Import the necessary packages - 2.5 marks

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

2. Read the 3 datasets into dataframes - 2.5 marks

In [79]:
```
ratings = pd.read_csv('rating.csv') #Data.csv renamed to rating.csv
movies = pd.read_csv('movie.csv') #item.csv renamed to movie.csv
users = pd.read_csv('user.csv')
```

3. Apply info, shape, describe, and find the number of missing values in the data. Present at least 3 observations from these operations - 2.5 marks

- Note that you will need to do it for all the three datasets seperately

In [4]:
```
print('Ratings Data Information:\n') #prints title for below output, likewise
ratings.info()
print('\n')
print('Ratings Data Shape:')
ratings.shape
```

```
Ratings Data Information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
```

```
 0   user id    100000 non-null  int64
 1   movie id   100000 non-null  int64
 2   rating     100000 non-null  int64
 3   timestamp  100000 non-null  int64
dtypes: int64(4)
memory usage: 3.1 MB


Ratings Data Shape:
```

Out[4]:
```
(100000, 4)
```

In [5]:
```
print('Ratings Data Description:')
ratings.describe()
```

```
Ratings Data Description:
```

Out[5]:

	user id	movie id	rating	timestamp
count	100000.00000	100000.000000	100000.000000	1.000000e+05
mean	462.48475	425.530130	3.529860	8.835289e+08
std	266.61442	330.798356	1.125674	5.343856e+06
min	1.00000	1.000000	1.000000	8.747247e+08
25%	254.00000	175.000000	3.000000	8.794487e+08
50%	447.00000	322.000000	4.000000	8.828269e+08
75%	682.00000	631.000000	4.000000	8.882600e+08
max	943.00000	1682.000000	5.000000	8.932866e+08

In [6]:
```
print('Ratings No. of missing values:')
ratings.isnull().sum()
```

```
Ratings No. of missing values:
```

Out[6]:
```
user id      0
movie id     0
rating       0
timestamp    0
dtype: int64
```

In [7]:
```
print('Movies Data Information:\n')
movies.info()
print('\n')
print('Movies Data Shape:')
movies.shape
```

```
Movies Data Information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1681 entries, 0 to 1680
Data columns (total 22 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   movie id      1681 non-null   int64
 1   movie title   1681 non-null   object
 2   release date  1681 non-null   object
 3   unknown       1681 non-null   int64
 4   Action        1681 non-null   int64
 5   Adventure     1681 non-null   int64
 6   Animation     1681 non-null   int64
 7   Childrens     1681 non-null   int64
 8   Comedy        1681 non-null   int64
 9   Crime         1681 non-null   int64
 10  Documentary   1681 non-null   int64
 11  Drama         1681 non-null   int64
```

```
 12  Fantasy       1681 non-null   int64
 13  Film-Noir     1681 non-null   int64
 14  Horror        1681 non-null   int64
 15  Musical       1681 non-null   int64
 16  Mystery       1681 non-null   int64
 17  Romance       1681 non-null   int64
 18  Sci-Fi        1681 non-null   int64
 19  Thriller      1681 non-null   int64
 20  War           1681 non-null   int64
 21  Western       1681 non-null   int64
dtypes: int64(20), object(2)
memory usage: 289.0+ KB


Movies Data Shape:
```

Out[7]:

```
(1681, 22)
```

In [8]:

```
print('Movies Data Description:')
movies.describe()
```

```
Movies Data Description:
```

Out[8]:

	movie id	unknown	Action	Adventure	Animation	Childrens	Co
count	1681.000000	1681.000000	1681.000000	1681.000000	1681.000000	1681.000000	1681.0(
mean	841.841761	0.000595	0.149316	0.080309	0.024985	0.072576	0.3
std	485.638077	0.024390	0.356506	0.271852	0.156126	0.259516	0.4
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	422.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
50%	842.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
75%	1262.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.0(
max	1682.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0(

In [9]:

```
print('Movies No. of missing values:')
movies.isnull().sum()
```

```
Movies No. of missing values:
```

Out[9]:

```
movie id        0
movie title     0
release date    0
unknown         0
Action          0
Adventure       0
Animation       0
Childrens       0
Comedy          0
Crime           0
Documentary     0
Drama           0
Fantasy         0
Film-Noir       0
Horror          0
Musical         0
Mystery         0
Romance         0
Sci-Fi          0
Thriller        0
War             0
Western         0
dtype: int64
```

In [10]:
```
print('Users Data Information:\n')
users.info()
print('\n')
print('Users Data Shape:')
users.shape
```

```
Users Data Information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 943 entries, 0 to 942
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   user id     943 non-null    int64 
 1   age         943 non-null    int64 
 2   gender      943 non-null    object
 3   occupation  943 non-null    object
 4   zip code    943 non-null    object
dtypes: int64(2), object(3)
memory usage: 37.0+ KB


Users Data Shape:
```

Out[10]:
```
(943, 5)
```

In [11]:
```
print('Users Data Description:')
users.describe()
```

```
Users Data Description:
```

Out[11]:

	user id	age
count	943.000000	943.000000
mean	472.000000	34.051962
std	272.364951	12.192740
min	1.000000	7.000000
25%	236.500000	25.000000
50%	472.000000	31.000000
75%	707.500000	43.000000
max	943.000000	73.000000

In [12]:
```
print('Users No. of missing values:')
users.isnull().sum()
```

```
Users No. of missing values:
```

Out[12]:
```
user id       0
age           0
gender        0
occupation    0
zip code      0
dtype: int64
```

Observations: 1- There is no missing values in any of the datasets 2- With the movie ratings out of a 1-5 scale, 3.5/5 seems to be the average for a user rating 3- Regarding the users age data, with the mean (34) being higher than the 50% value, it is likely that the data is slightly positively skewed

4. Find the number of movies per genre using the item data - 2.5 marks

In [80]:
```
#for each genre, there is either a 1 or 0 as input for each movie, we only ne
print('unknown:',movies['unknown'].value_counts()[1])
print('Action:',movies['Action'].value_counts()[1])
print('Adventure:',movies['Adventure'].value_counts()[1])
print('Animation:',movies['Animation'].value_counts()[1])
print('Childrens:',movies['Childrens'].value_counts()[1])
print('Comedy:',movies['Comedy'].value_counts()[1])
print('Crime:',movies['Crime'].value_counts()[1])
print('Documentary:',movies['Documentary'].value_counts()[1])
print('Drama:',movies['Drama'].value_counts()[1])
print('Fantasy:',movies['Fantasy'].value_counts()[1])
print('Film-Noir:',movies['Film-Noir'].value_counts()[1])
print('Horror:',movies['Horror'].value_counts()[1])
print('Musical:',movies['Musical'].value_counts()[1])
print('Mystery:',movies['Mystery'].value_counts()[1])
print('Romance:',movies['Romance'].value_counts()[1])
print('Sci-Fi:',movies['Sci-Fi'].value_counts()[1])
print('Thriller:',movies['Thriller'].value_counts()[1])
print('War:',movies['War'].value_counts()[1])
print('Western:',movies['Western'].value_counts()[1])
```

```
unknown: 1
Action: 251
Adventure: 135
Animation: 42
Childrens: 122
Comedy: 505
Crime: 109
Documentary: 50
Drama: 725
Fantasy: 22
Film-Noir: 24
Horror: 92
Musical: 56
Mystery: 61
Romance: 247
Sci-Fi: 101
Thriller: 251
War: 71
Western: 27
```

Insights: It seems 'Drama' is the most popular genre from our dataset with 725 total movies. There is 1 movie with an unknown genre. 'Fantasy' is the least popular from the dataset.

5. Drop the movie where the genre is unknown - 2.5 marks

In [14]:
```
movies = movies[movies['unknown']!=1]
```

Insights: Since we dropped the movie with the unknown genre, now we have 1680 entries instead of 1681. This is done by feeding the original dataframe with the dataframe that only has '0' values under the 'unknown' column.

6. Find the movies that have more than one genre - 5 marks

hint: use sum on the axis = 1

Display movie name, number of genres for the movie in dataframe

and also print(total number of movies which have more than one genres)

In [15]:
```
column_list = list(movies) #creates a list that stores the columns
column_list.remove("movie id") #removes 'movie id' from list so it does not m
```

```
movies["no. of genres"] = movies[column_list].sum(axis = 1) #creates new colu
movies_updated = movies["no. of genres"] #our new dataframe in new variable
movies[movies_updated > 1] # returns movies with genres > 1 in that column
```

Out[15]:

	movie id	movie title	release date	unknown	Action	Adventure	Animation	Childrens	Comedy
0	1	Toy Story	01-Jan-1995	0	0	0	1	1	1
1	2	GoldenEye	01-Jan-1995	0	1	1	0	0	0
3	4	Get Shorty	01-Jan-1995	0	1	0	0	0	1
4	5	Copycat	01-Jan-1995	0	0	0	0	0	0
6	7	Twelve Monkeys	01-Jan-1995	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
1667	1669	MURDER and murder	20-Jun-1997	0	0	0	0	0	0
1668	1670	Tainted	01-Feb-1998	0	0	0	0	0	1
1671	1673	Mirage	01-Jan-1995	0	1	0	0	0	0
1677	1679	B. Monkey	06-Feb-1998	0	0	0	0	0	0
1678	1680	Sliding Doors	01-Jan-1998	0	0	0	0	0	0

849 rows × 23 columns

In [16]:

```
movies_updatedv2 = movies[movies_updated > 1] #dataframe with genres > 1 in n
print('Total no. of movies with more than one more genre is: ',movies_updated
#no. of rows in shape corresponds to number of movies in this dataframe
```

```
Total no. of movies with more than one more genre is:  849
```

Insights:

7. Univariate plots of columns: 'rating', 'Age', 'release year', 'Gender' and 'Occupation' - 10 marks

HINT: Use distplot for age and countplot for release year, ratings,

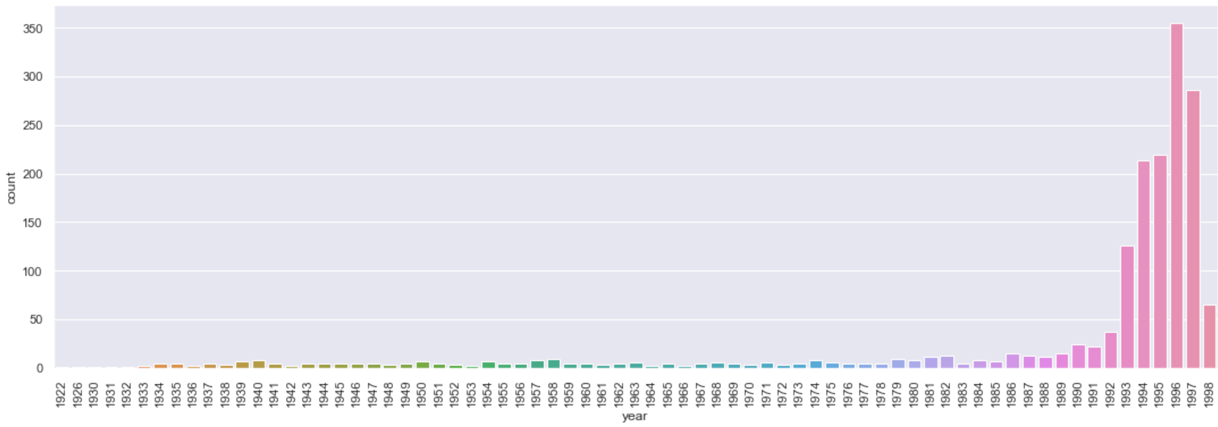
HINT: Please refer to the below snippet to understand how to get to release year from release date. You can use str.split() as depicted below or you could convert it to pandas datetime format and extract year (.dt.year)

In [ ]:
```
a = 'My*cat*is*brown'
print(a.split('*')[3])

#similarly, the release year needs to be taken out from release date

#also you can simply slice existing string to get the desired data, if we wan

print(a[10:])
print(a[-5:])
```

In [17]:
```
#starting with the 'release year' plot
movies['release date'] = pd.to_datetime(movies['release date']) #converting d
movies['year'] = movies['release date'].dt.year #extracting the year and putt
```

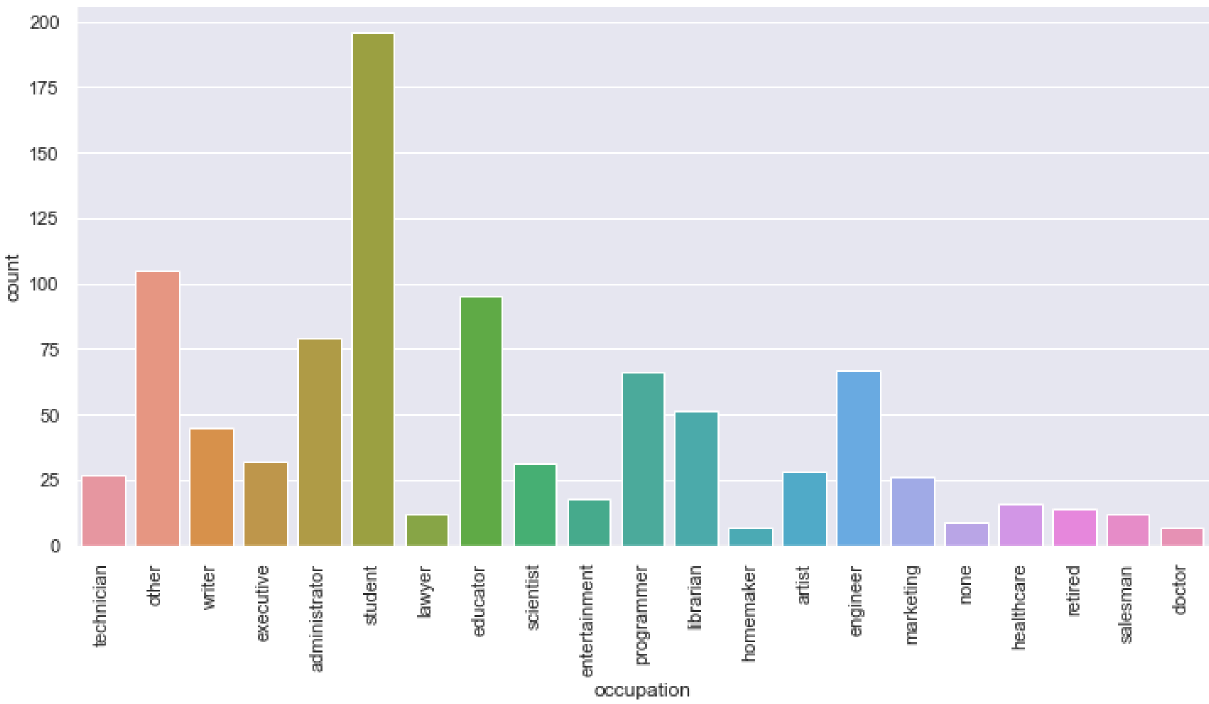
In [18]:
```
year_plot = sns.catplot(x='year',data=movies,kind="count",aspect=3)
year_plot.set_xticklabels(rotation=90) #xticklabels enables us to rotate catp
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x1ef0cd01a30>



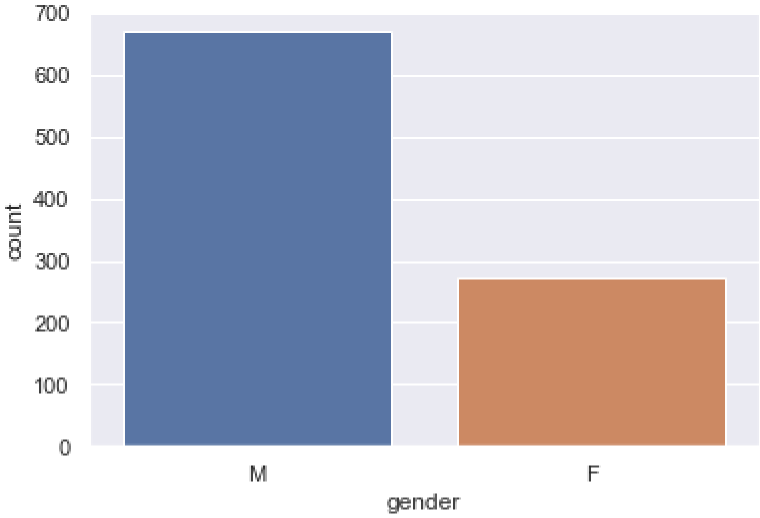
In [19]:
```
occupation_plot = sns.catplot(x='occupation',data=users,kind="count",aspect=2
occupation_plot.set_xticklabels(rotation=90)
```

Out[19]: <seaborn.axisgrid.FacetGrid at 0x1ef0e12c9d0>



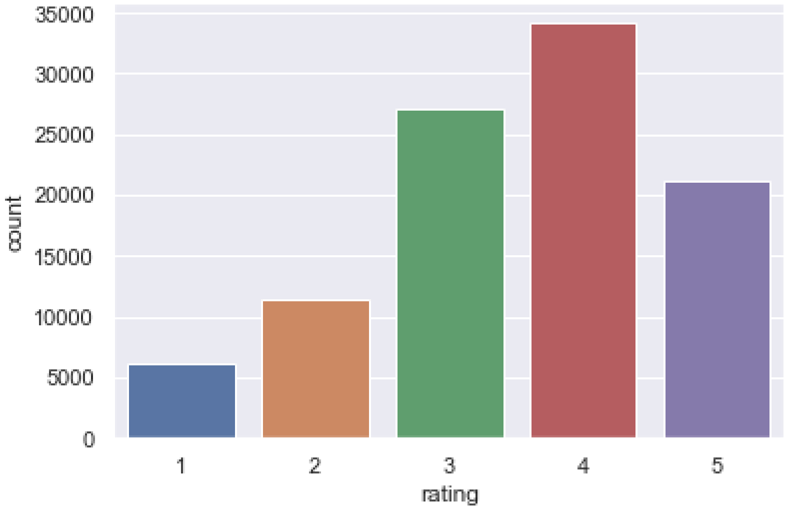
In [20]:
```
sns.countplot(x=users['gender'])
```

Out[20]: <AxesSubplot:xlabel='gender', ylabel='count'>



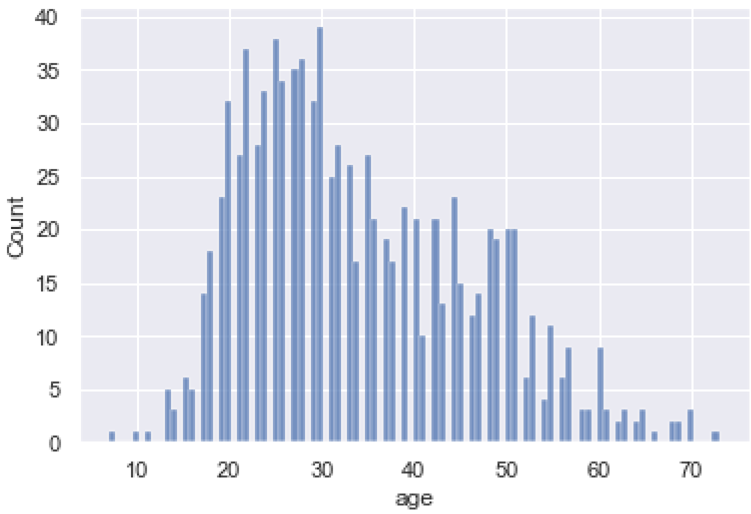
In [21]:
```
sns.countplot(x=ratings['rating'])
```

Out[21]: <AxesSubplot:xlabel='rating', ylabel='count'>



In [22]:
```
sns.histplot(users['age'],bins = 100)
```

Out[22]: <AxesSubplot:xlabel='age', ylabel='Count'>



8. Visualize how popularity of genres has changed over the years - 10 marks

Note that you need to use the **percent of number of releases in a year** as a parameter of popularity of a genre

Hint 1: You need to reach to a data frame where the release year is the index and the genre is the column names (one cell shows the number of release in a year in one genre) or vice versa. (Drop unnecessary column if there are any)

Hint 2: Find the total number of movies release in a year(use item dataset to get count of movies released in a particular year, store that value in a new column as 'total'). Now divide the value of each genre in that year by total to get percentage number of release in a particular year. `(df.div(df['total'], axis= 0) * 100)`

Once that is achieved, you can either use univariate plots or can use the heatmap to visualise all the changes over the years in one go.

Hint 3: Use groupby on the relevant column and use sum() on the same to find out the number of releases in a year/genre.

In [25]:
```
movies['release date'] = pd.to_datetime(movies['release date'])
movies_per_year = movies['release date'].groupby(movies['release date'].dt.ye
movies_per_year #need to store in column 'total' in dataframe that we will cr
```

Out[25]:
```
release date
1922      1
1926      1
1930      1
1931      1
1932      1
       ...
1994    214
1995    219
1996    355
1997    286
1998     65
Name: release date, Length: 71, dtype: int64
```

In [23]:
```
genre_list = list(movies) #this list stores the columns in movie dataset
genre_list.remove("movie id") #removing non genre columns from list
genre_list.remove("movie title")
genre_list.remove("release date")
genre_list.remove("no. of genres")
genre_list.remove("year")
```

In [26]:
```
df = movies.groupby(by=['year'])[genre_list].sum()
df["Total"]=movies_per_year
df
```

Out[26]:

	unknown	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drar
year									
1922	0	0	0	0	0	0	0	0	
1926	0	0	0	0	0	0	0	0	
1930	0	0	0	0	0	0	0	0	
1931	0	0	0	0	0	0	1	0	
1932	0	0	0	0	0	0	0	0	

	unknown	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drar
year									
...	...	...	...	...	...	...	...	...	
1994	0	30	13	4	15	82	8	9	
1995	0	40	22	6	21	63	11	5	[illegible]
1996	0	44	24	9	21	108	21	18	1
1997	0	46	20	3	22	87	30	6	1
1998	0	12	3	0	1	13	7	3	[illegible]

71 rows × 20 columns

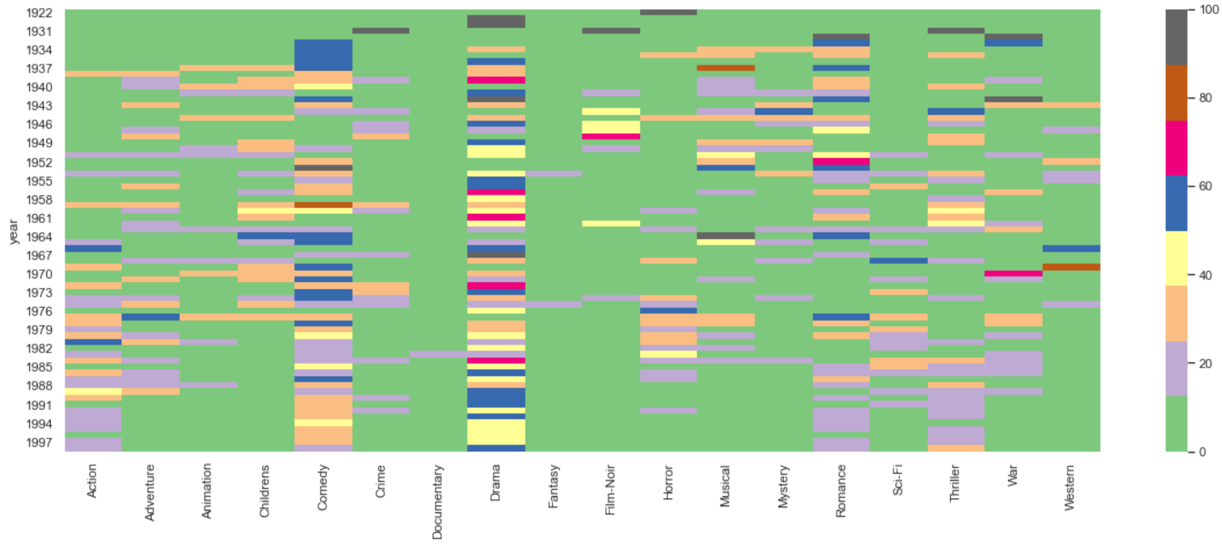
In [27]:
```python
df2 = (df.div(df['Total'], axis=0)*100) #converting to percentages based on c
df2.drop(['unknown','Total'],axis=1,inplace=True)
df2
```

Out[27]:

	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	
year								
1922	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
1926	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.
1930	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.
1931	0.000000	0.000000	0.000000	0.000000	0.000000	100.000000	0.000000	0.
1932	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
...	...	...	...	...	...	...	...	
1994	14.018692	6.074766	1.869159	7.009346	38.317757	3.738318	4.205607	45
1995	18.264840	10.045662	2.739726	9.589041	28.767123	5.022831	2.283105	40.
1996	12.394366	6.760563	2.535211	5.915493	30.422535	5.915493	5.070423	47.
1997	16.083916	6.993007	1.048951	7.692308	30.419580	10.489510	2.097902	39.
1998	18.461538	4.615385	0.000000	1.538462	20.000000	10.769231	4.615385	50

71 rows × 18 columns

In [28]:
```python
plt.figure(figsize=(20,7))
sns.heatmap(df2, cmap='Accent')
plt.show()
```



Insights:

- Based on the plot, 'Drama' seemed to be the most popular over the years, usually exceeding 40% out of all movies per year.
- The least popular genre in the dataset would be 'Documentary' only showing up in 1983 at about 20%.
- One insight about 'Horror' it was most popular during 1976 more than any previous or following year.

9. Find the top 25 movies in terms of average ratings for movies that have been rated more than 100 times - 10 marks

Hints :

1. Find the count of ratings and average ratings for every movie.
2. Slice the movies which have ratings more than 100.
3. Sort values according to average rating such that movie which highest rating is on top.
4. Select top 25 movies.
5. You will have to use the .merge() function to get the movie titles.

Note: This question will need you to research about groupby and apply your findings. You can find more on groupby on https://realpython.com/pandas-groupby/.

In [29]:
```
average_ratings = ratings.groupby('movie id')['rating'].mean()
average_ratings
```

Out[29]:
```
movie id
1       3.878319
2       3.206107
3       3.033333
4       3.550239
5       3.302326
          ...
1678    1.000000
1679    3.000000
1680    2.000000
1681    3.000000
1682    3.000000
Name: rating, Length: 1682, dtype: float64
```

In [30]:
```
count_group = ratings.groupby("movie id").count()["rating"] #number of rating
count_group
```

Out[30]:
```
movie id
1       452
2       131
3        90
4       209
5        86
       ...
1678      1
1679      1
1680      1
1681      1
1682      1
Name: rating, Length: 1682, dtype: int64
```

In [31]:
```
sorted_ratings = pd.merge(average_ratings, count_group,how='outer',on='movie
sorted_ratings
```

Out[31]:

movie id	rating_x	rating_y
1	3.878319	452
2	3.206107	131
3	3.033333	90
4	3.550239	209
5	3.302326	86
...	...	...
1678	1.000000	1
1679	3.000000	1
1680	2.000000	1
1681	3.000000	1
1682	3.000000	1

1682 rows × 2 columns

In [32]:
```
more_than_100 = sorted_ratings[count_group > 100] #contains movies that have
more_than_100
```

Out[32]:

movie id	rating_x	rating_y
1	3.878319	452
2	3.206107	131
4	3.550239	209
7	3.798469	392
8	3.995434	219
...	...	...
926	2.702970	101
928	3.115385	104
1016	3.459854	137

	rating_x	rating_y
movie id		
1028	3.040541	148
1047	2.835821	134

334 rows × 2 columns

In [33]:
```
more_than_100.columns =['avg. rating', 'no. ratings']
more_than_100
```

Out[33]:

	avg. rating	no. ratings
movie id		
1	3.878319	452
2	3.206107	131
4	3.550239	209
7	3.798469	392
8	3.995434	219
...	...	...
926	2.702970	101
928	3.115385	104
1016	3.459854	137
1028	3.040541	148
1047	2.835821	134

334 rows × 2 columns

In [34]:
```
highest_to_lowest=more_than_100.sort_values('avg. rating',ascending=False)
highest_to_lowest
```

Out[34]:

	avg. rating	no. ratings
movie id		
408	4.491071	112
318	4.466443	298
169	4.466102	118
483	4.456790	243
64	4.445230	283
...	...	...
358	2.615385	143
260	2.574803	127
325	2.546875	128
243	2.439394	132
122	2.339623	106

334 rows × 2 columns

In [35]:
```
top_25 = highest_to_lowest.head(25) #now we have the top 25 of movies rated m
top_25
```

Out[35]:

movie id	avg. rating	no. ratings
408	4.491071	112
318	4.466443	298
169	4.466102	118
483	4.456790	243
64	4.445230	283
603	4.387560	209
12	4.385768	267
50	4.358491	583
178	4.344000	125
134	4.292929	198
427	4.292237	219
357	4.291667	264
98	4.289744	390
480	4.284916	179
127	4.283293	413
285	4.265432	162
272	4.262626	198
657	4.259542	131
474	4.252577	194
174	4.252381	420
479	4.251397	179
313	4.245714	350
511	4.231214	173
484	4.210145	138
172	4.204360	367

In [40]:
```
top25_with_titles = top_25.merge(movies, left_on='movie id', right_on='movie
print('Top 25 movies in terms of average ratings that are rated more than 100
top25_with_titles
```

```
Top 25 movies in terms of average ratings that are rated more than 100 times:
```

Out[40]:

	movie id	avg. rating	no. ratings	movie title	release date	unknown	Action	Adventure	Animation	(
0	408	4.491071	112	Close Shave, A	1996-04-28	0	0	0	1	

	movie id	avg. rating	no. ratings	movie title	release date	unknown	Action	Adventure	Animation
1	318	4.466443	298	Schindler's List	1993-01-01	0	0	0	0
2	169	4.466102	118	Wrong Trousers, The	1993-01-01	0	0	0	1
3	483	4.456790	243	Casablanca	1942-01-01	0	0	0	0
4	64	4.445230	283	Shawshank Redemption, The	1994-01-01	0	0	0	0
5	603	4.387560	209	Rear Window	1954-01-01	0	0	0	0
6	12	4.385768	267	Usual Suspects, The	1995-08-14	0	0	0	0
7	50	4.358491	583	Star Wars	1977-01-01	0	1	1	0
8	178	4.344000	125	12 Angry Men	1957-01-01	0	0	0	0
9	134	4.292929	198	Citizen Kane	1941-01-01	0	0	0	0
10	427	4.292237	219	To Kill a Mockingbird	1962-01-01	0	0	0	0
11	357	4.291667	264	One Flew Over the Cuckoo's Nest	1975-01-01	0	0	0	0
12	98	4.289744	390	Silence of the Lambs, The	1991-01-01	0	0	0	0
13	480	4.284916	179	North by Northwest	1959-01-01	0	0	0	0
14	127	4.283293	413	Godfather, The	1972-01-01	0	1	0	0
15	285	4.265432	162	Secrets & Lies	1996-10-04	0	0	0	0
16	272	4.262626	198	Good Will Hunting	1997-01-01	0	0	0	0
17	657	4.259542	131	Manchurian Candidate, The	1962-01-01	0	0	0	0
18	474	4.252577	194	Dr. Strangelove or: How I Learned to Stop Worr...	1963-01-01	0	0	0	0
19	174	4.252381	420	Raiders of the Lost Ark	1981-01-01	0	1	1	0
20	479	4.251397	179	Vertigo	1958-01-01	0	0	0	0

	movie id	avg. rating	no. ratings	movie title	release date	unknown	Action	Adventure	Animation
21	313	4.245714	350	Titanic	1997-01-01	0	1	0	0
22	511	4.231214	173	Lawrence of Arabia	1962-01-01	0	0	1	0
23	484	4.210145	138	Maltese Falcon, The	1941-01-01	0	0	0	0
24	172	4.204360	367	Empire Strikes Back, The	1980-01-01	0	1	1	0

25 rows × 26 columns

10. Check for the validity of the below statements with respect to the data provided - 10 marks

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

compare the percentages

Please pay attention to what should be the denominator while calculating percentages

1. Merge all the datasets
2. There is no need to conduct statistical tests around this. Just **compare the percentages** and comment on the validity of the above statements.
3. you might want ot use the .sum(), .div() function here.
4. Use number of ratings to validate the numbers. For example, if out of 4000 ratings received by women, 3000 are for drama, we will assume that 75% of the women watch drama.

In [51]:
```
dataframe = users.merge(ratings, left_on='user id', right_on='user id', how='
dataframe
```

Out[51]:

	user id	age	gender	occupation	zip code	movie id	rating	timestamp
0	1	24	M	technician	85711	61	4	878542420
1	1	24	M	technician	85711	189	3	888732928
2	1	24	M	technician	85711	33	4	878542699
3	1	24	M	technician	85711	160	4	875072547
4	1	24	M	technician	85711	20	4	887431883
...	...	...	...	...	...	...	...	...
99995	943	22	M	student	77841	415	1	888640027
99996	943	22	M	student	77841	219	4	888639575
99997	943	22	M	student	77841	796	3	888640311

	user id	age	gender	occupation	zip code	movie id	rating	timestamp
99998	943	22	M	student	77841	739	4	888639929
99999	943	22	M	student	77841	391	2	888640291

100000 rows × 8 columns

In [59]:
```
dataframe2 = dataframe.merge(movies, left_on='movie id', right_on='movie id',
dataframe2
```

Out[59]:

	user id	age	gender	occupation	zip code	movie id	rating	timestamp	movie title	release date
0	1	24	M	technician	85711	61	4	878542420	Three Colors: White	1994-01-01
1	1	24	M	technician	85711	189	3	888732928	Grand Day Out, A	1992-01-01
2	1	24	M	technician	85711	33	4	878542699	Desperado	1995-01-01
3	1	24	M	technician	85711	160	4	875072547	Glengarry Glen Ross	1992-01-01
4	1	24	M	technician	85711	20	4	887431883	Angels and Insects	1995-01-01
...	...	...	...	...	...	...	...	...	...	...
99995	943	22	M	student	77841	415	1	888640027	Apple Dumpling Gang, The	1975-01-01
99996	943	22	M	student	77841	219	4	888639575	Nightmare on Elm Street, A	1984-01-01
99997	943	22	M	student	77841	796	3	888640311	Speechless	1994-01-01
99998	943	22	M	student	77841	739	4	888639929	Pretty Woman	1990-01-01
99999	943	22	M	student	77841	391	2	888640291	Last Action Hero	1993-01-01

100000 rows × 31 columns

In [64]:
```
df3 = dataframe2.groupby(by=['gender'])[genre_list].sum() #using genre list c
df3
```

Out[64]:

	unknown	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	
gender									
F	0.0	5442.0	3141.0	995.0	2232.0	8068.0	1794.0	187.0	1
M	0.0	20147.0	10612.0	2610.0	4950.0	21764.0	6261.0	571.0	2

In [66]:
```
Total_Ratings = df3.sum(axis=0) #total for both Female and Male in each genre
Total_Ratings
```

Out[66]:
```
unknown            0.0
Action         25589.0
Adventure      13753.0
Animation       3605.0
Childrens       7182.0
Comedy         29832.0
Crime           8055.0
Documentary      758.0
Drama          39895.0
Fantasy         1352.0
Film-Noir       1733.0
Horror          5317.0
Musical         4954.0
Mystery         5245.0
Romance        19461.0
Sci-Fi         12730.0
Thriller       21872.0
War             9398.0
Western         1854.0
dtype: float64
```

In [75]:
```python
df4 = (df3.div(Total_Ratings, axis=1)*100) #convert gender dataframe into per
df4.drop(columns="unknown",inplace=True)
df4
```

Out[75]:

ıma	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller
243	26.849112	22.215811	22.512695	29.107792	25.052431	30.101228	20.652003	23.253475
757	73.150888	77.784189	77.487305	70.892208	74.947569	69.898772	79.347997	76.746525

Conclusion:

```
- Men watch more Drama than Women = True, with
Men(72.4%)>Women(27.59%)
- Women do not watch more Sci-Fi than Men, with
Men(79.34%)>Women(20.65%)
- Men watch more Romance than Women = True, with
Men(69.89%)>Women(30.1%)
Although these are percentages on a specific user database with
a limited number of movies, these numbers do not reflect on
the general viewers outside of this dataset.
```