IMDB DATASET Student

Details:

Student Name and ID: **Jayachandra Jarajapu (1001964536)**

Loading the dataset:

- The IMDB dataset is loaded into a data frame using pandas pd
- We use head function to retrieve the number of rows in the data frame

```
In [109]: # Read the file
          df_data = pd.read_csv('IMDB_movie_reviews_details.csv', encoding='latin1')
          # the data from the CSV is read into df_data dataframe using pandas pd function
```

```
In [110]: #return the first 5 rows of the dataset
          df_data.head(5)
          #the first five rows of the dataframe are retrived and we use head function in Python
```

Out[110]:

	Unnamed: 0	name	year	runtime	genre	rating	metascore	timeline	votes	gross
0	0	The Shawshank Redemption	1994	142	Drama	9.3	80.0	Two imprisoned men bond over a number of years...	2,394,059	$28.34M
1	1	The Godfather	1972	175	Crime, Drama	9.2	100.0	An organized crime dynasty's aging patriarch t...	1,658,439	$134.97M
2	2	Soorarai Pottru	2020	153	Drama	9.1	NaN	Nedumaaran Rajangam "Maara" sets out to make t...	78,266	NaN
3	3	The Dark Knight	2008	152	Action, Crime, Drama	9.0	84.0	When the menace known as the Joker wreaks havo...	2,355,907	$534.86M
4	4	The Godfather: Part II	1974	202	Crime, Drama	9.0	90.0	The early life and career of Vito Corleone in ...	1,152,912	$57.30M

TASK 1 - Statistical Exploratory Data Analysis

Task1-a: Printing the details of the data frame

- We use option_context from pandas to print all the details of the data frame in python

```
In [235]: #1-a Print the details of dataframe
          #to print all the details of the dataframe we use option_context in pandas
          #this will show all the results
          with pd.option_context('display.max_rows', None, 'display.max_columns', None):
              print (">>Task 1-a: Details of df_data data frame are: \n", df_data)
```

```
>>Task 1-a: Details of df_data data frame are:
      Unnamed: 0                                                name      year  \
0              0                            The Shawshank Redemption      1994
1              1                                       The Godfather      1972
2              2                                     Soorarai Pottru      2020
3              3                                     The Dark Knight      2008
4              4                                The Godfather: Part II      1974
5              5                                        12 Angry Men      1957
6              6      The Lord of the Rings: The Return of the King      2003
7              7                                         Pulp Fiction      1994
8              8                                    Schindler's List      1993
9              9                                           Inception      2010
10            10                                         Fight Club      1999
11            11  The Lord of the Rings: The Fellowship of the Ring      2001
12            12                                      Forrest Gump      1994
13            13                The Good, the Bad and the Ugly          1966
14            14          The Lord of the Rings: The Two Towers       2002
15            15                                        The Matrix      1999
16            16                                        Goodfellas      1990
```

Task 1-b: Finding the number of rows and columns in dataset

- To find the length of the rows and columns we use len in python

In [112]:
```
#1-b Find the number of rows and columns in dataset
#We have a function in pandas which will count the length of the rows and columns
#the length function will calculate the value through the axes
num_rows = len(df_data.axes[0])
#axes[0] is considered as rows
num_cols = len(df_data.axes[1])
#axes[1] is considered as columns
print ("\n\n>>Task 1-b: Number of rows:%s and number of columns:%s" % (num_rows, num_cols))
```

```
>>Task 1-b: Number of rows:1000 and number of columns:10
```

Task 1-c: descriptive detail of a 'metascore' and ‘rating’ column in dataset

- We have describe function in python which will provide all the descriptive details of the particular column in a data frame

In [113]:
```
#1-c Print descriptive detail of a 'metascore' and ‘rating’ column in dataset
#In pandas we have a special tool which will provide the description of the dataset
print ("\n\n>>Task 1-c: Descriptive details of 'metascore' and ‘rating’ column are\n", df_data[['metascore', 'rating']].describe
```

```
>>Task 1-c: Descriptive details of 'metascore' and ‘rating’ column are
         metascore       rating
count  841.000000  1000.000000
mean    78.158145     7.954000
std     12.289270     0.276008
min     28.000000     7.600000
25%     71.000000     7.700000
50%     79.000000     7.900000
75%     87.000000     8.100000
max    100.000000     9.300000
```

Task 1-d: Finding all the unique values for a column year and it’s respective length.

- We have unique function in python which will provide all the unique values in python
- To calculate the respective length we use length function

In [291]:
```
#1-d Find all the unique values for a column year and it’s respective length.
#to find the unique values we can use unique function in python pandas
num_uniq_year= df_data.year.unique()
#the above above is stored to a variable num_uniq_year
print ("\n\n >>Task 1-d:")
print(num_uniq_year)
print("##################################################")
print("The respective length of year column :")
print(len(num_uniq_year))
#the length of the required unique years are printed accordingly
```

```
 >>Task 1-d:
['1994' '1972' '2020' '2008' '1974' '1957' '2003' '1993' '2010' '1999'
 '2001' '1966' '2002' '1990' '1980' '1975' '2019' '2014' '1998' '1997'
 '1995' '1991' '1977' '1962' '1954' '1946' '2018' '2011' '2006' '2000'
 '1988' '1985' '1968' '1960' '1942' '1936' '1931' '2017' '2016' 'I 2017'
 '2012' '2009' '2007' '1984' '1981' '1979' '1963' '1964' '1950' '1940'
 '2013' 'I 2020' '2005' '2004' '1992' '1987' '1986' '1983' '1976' '1973'
 '1971' '1959' '1958' '1952' '1948' '1944' '1941' '1927' '1921' '2015'
 '2021' '1996' '1989' '1978' '1965' '1961' '1953' '1925' '1924' 'III 2016'
 'I 2014' 'I 2015' 'I 2013' '1982' '1967' '1955' '1951' '1949' '1939'
 '1937' '1934' '1930' '1928' '1926' '1920' 'I 2004' '1970' '1969' '1956'
 '1947' '1945' '1943' 'II 2016' 'I 2011' 'I 2001' '1938' '1935' '1933'
 '1932' '1922' 'I 2010' 'I 2008' 'I 2007' 'I 1985' 'III 2018' 'II 2015'
 'I 2016' 'I 1995']
##################################################
The respective length of year column :
118
```

Task 2-a: Data whose rating is greater than 9

- We have used arithmetic operation to calculate the movies with rating greater than 9

In [115]:
```
#Task 2-a: Filter out the data by rating whose rating is greater than 9
Rating_g = df_data[df_data['rating'] > 9]
# the above code will filter data with rating more than 9
print (">>Task 2-a: Filter out the data by rating whose rating is greater than 9 \n %s"
% (Rating_g))
```

```
>>Task 2-a: Filter out the data by rating whose rating is greater than 9
    Unnamed: 0                     name  year  runtime         genre  rating  \
0           0  The Shawshank Redemption  1994      142         Drama     9.3
1           1             The Godfather  1972      175  Crime, Drama     9.2
2           2           Soorarai Pottru  2020      153         Drama     9.1

   metascore                                           timeline      votes  \
0       80.0  Two imprisoned men bond over a number of years...  2,394,059
1      100.0  An organized crime dynasty's aging patriarch t...  1,658,439
2        NaN  Nedumaaran Rajangam "Maara" sets out to make t...     78,266

      gross
0   $28.34M
1  $134.97M
2       NaN
```

Task 2-b: Number of movies released between 1990 and 2000

- We have created a new dataframe from the existing data frame because there are string values in the year column
- Later we have split the column and converted it to a float value
- Doing this will help us print the values between year 1990 and 2000

```
2       NaN
```

In [283]:
```
#Task 2-b: Total number of movies released between 1990 in 2000
df_data = pd.read_csv('IMDB_movie_reviews_details.csv', encoding='latin1')
df_data1 = df_data.copy()

df_data1['year'] = df_data1.year.str.replace(r"[a-zA-Z]",'')
df_data1['year'] = df_data1.year.str.replace(r" ",'')

df_data1['year'] = df_data1['year'].astype(float)
num_movies = df_data1[((df_data1['year']) >= 1990) & ((df_data1['year']) <= 2000)]
print(len(num_movies))
```

```
169
```

Task 2C: Top 10 Movies with the highest rating

- To find the top 10 movies with highest rating we can sort the dataframe with the highest rating to the descending order
- This will retrive the top 10 values of the data frame
- We have used Head function to retrive the top 10 movies with highest rating

In [159]:
```
#Task 2-c: Find out the top 10 movies with the highest rating.
top10_movies = df_data.sort_values(["rating"], ascending=False)
#we use sort_values function in order to ascend the values in the dataframe
print ("\n\n>>Task 2-c: top 10 movies with the highest rating: \n" , top10_movies.head(10))
# we use head function to print the top 10 movies with highest rating
```

```
>>Task 2-c: top 10 movies with the highest rating:
     Unnamed: 0                                                name  year  \
0             0                          The Shawshank Redemption  1994
1             1                                     The Godfather  1972
2             2                                   Soorarai Pottru  2020
3             3                                   The Dark Knight  2008
4             4                             The Godfather: Part II  1974
5             5                                      12 Angry Men  1957
6             6      The Lord of the Rings: The Return of the King  2003
7             7                                      Pulp Fiction  1994
8             8                                  Schindler's List  1993
11           11  The Lord of the Rings: The Fellowship of the Ring  2001

    runtime                       genre  rating  metascore  \
0       142                       Drama     9.3       80.0
1       175               Crime, Drama     9.2      100.0
2       153                       Drama     9.1        NaN
3       152       Action, Crime, Drama     9.0       84.0
4       202               Crime, Drama     9.0       90.0
5        96               Crime, Drama     9.0       96.0
6       201   Action, Adventure, Drama     8.9       94.0
7       154               Crime, Drama     8.9       94.0
8       195  Biography, Drama, History     8.9       94.0
11      178   Action, Adventure, Drama     8.8       92.0
```

##TASK 3: VISUALIZATION

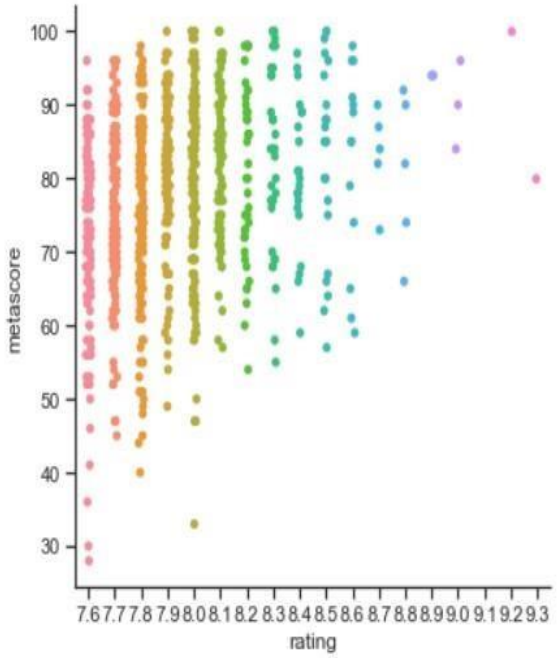
Task 3-a: Plotting the comparision of ratings with metascore

- Seaborn is a package in python which helps to plot the data in the dataframe through the axes
- We use catplot function to plot the comparision of ratings with metascore

In [292]:
```
#Task 3-a: Plot the comparison of ratings with metascore.

sns.set_theme(style="ticks")
# to compare two columns we use seaborn in python and it is imported as sns and the theme set is ticks
sns.catplot(data=df_data, x="rating", y="metascore")
#we use catplot function in seaborn between x axis and y axis and the graph is plotted
```

Out[292]: <seaborn.axisgrid.FacetGrid at 0x2548dfa85b0>



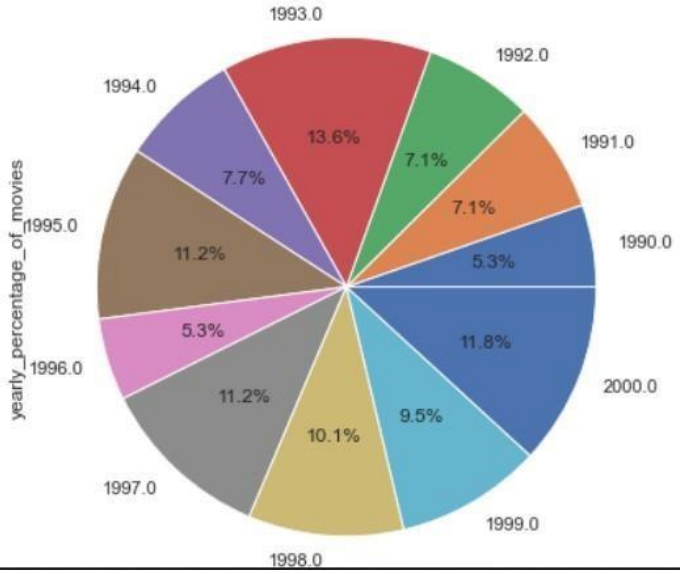
Task 3-b: Pie chart that shows the number of movies between 1990 and 2000.

- The custom dataframe df_data1 has the data between 1990 and 2000
- We use the column ‘Year’ to count the values using the count function
- The title is changed using Rename function
- We use matlibplot to create a pie chart with size using figsize and rcparams

```
In [286]: # task 3b: Draw a pie chart that shows the number of movies between 1990 and 2000.

num_movies1 = df_data1[((df_data1['year']) >= 1990) & ((df_data1['year']) <= 2000)]
#We have loaded the data between 1990 and 2000 to a variable

count_movies1 = num_movies1.groupby(['year']).count()
#we have counted the values of the year column in the variable
count_movies1.rename(columns={"name": "yearly_percentage_of_movies"}, inplace=True)
#the title for the pie chart is set to title name
count_movies1.yearly_percentage_of_movies.plot.pie(y ='year', figsize=(7, 7), autopct = "%0.1f%%")
#pie is a function which will plot the dataframe and we use figsize for the size of the figure
colors = plt.rcParams['axes.prop_cycle']
plt.show()
```

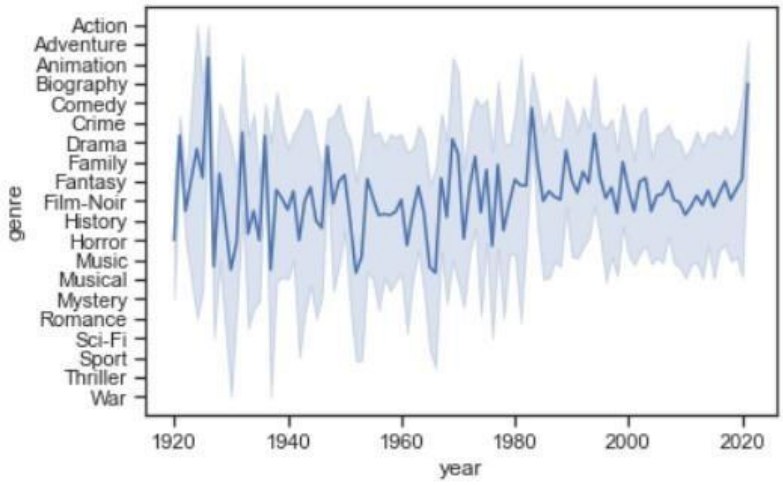


Task 4: Pattern Visualization

- To find an interesting pattern we have used two columns ‘genre’ and ‘year’ and we implemented loc function to plot the data
- Since many movies have multiple genres we have used split function to split the genre for the respective movies
- Then used groupby function to group the data for genres and year
- Seaborn sns is used to plot the data in the graph
- From the graph we could see the count of the movies released in a particular year for the genres

In [288]:
```
#Task4 finding an interesting pattern
df_data1.sort_values(by=['year'], inplace=True)
#we used a customized dataframe to find the pattern
plot_df = df_data1.loc[:, ['year']]
#we have use loc function to locate the data to a graph
plot_df['genre'] = df_data1['genre'].str.split(', ')
#we used split function to split the genre as some movies have multiple genres
plot_df = plot_df.explode('genre').reset_index(drop=True)
#we used explode function for genre to an index
testo = plot_df.groupby(['genre','year']).count()
#Genre is grouped by year and the count of the year and genre is calculated
sns.lineplot(data=testo, x='year', y='genre')
#we use seaborn lineplot for line graph between year and genre
```

Out[288]:
```
<AxesSubplot:xlabel='year', ylabel='genre'>
```



REFERENCES:

https://seaborn.pydata.org/generated/seaborn.lineplot.html

https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html