

```

In [1]: # imports all my libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import re
from datetime import datetime
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

pd.set_option("display.max_rows", None, "display.max_columns", None)

In [2]: #load the datasets
movies = pd.read_csv("movies.csv")
ratings = pd.read_csv("ratings.csv")
links = pd.read_csv("links.csv")
tags = pd.read_csv("tags.csv")

In [3]: # Inspecting the dataframe
print("MoviesShape: ", movies.shape)
movies.head()

MoviesShape: (9742, 3)
Out[3]:
  movieId          title      genres
0       1   Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy
1       2        Jumanji (1995) Adventure|Children|Fantasy
2       3    Grumpier Old Men (1995) Comedy|Romance
3       4     Waiting to Exhale (1995) Comedy|Drama|Romance
4       5 Father of the Bride Part II (1995) Comedy

In [4]: print("RatingsShape: ", ratings.shape)
movies.head(4)

RatingsShape: (100836, 4)
Out[4]:
  movieId          title      genres
0       1   Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy
1       2        Jumanji (1995) Adventure|Children|Fantasy
2       3    Grumpier Old Men (1995) Comedy|Romance
3       4     Waiting to Exhale (1995) Comedy|Drama|Romance
4       5 Father of the Bride Part II (1995) Comedy
5       6         Heat (1995) Action|Crime|Thriller

In [5]: #show structure and data type
movies.info()
ratings.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 # Column   Non-Null Count  Dtype  
--- 
 0 movieId    9742 non-null  int64  
 1 title      9742 non-null  object  
 2 genres     9742 non-null  object  
dtypes: int64(1), object(2)
memory usage: 228.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 # Column   Non-Null Count  Dtype  
--- 
 0 userId     100836 non-null  int64  
 1 movieId    100836 non-null  int64  
 2 rating     100836 non-null  float64 
 3 timestamp  100836 non-null  int64  
dtypes: float64(1), int64(3)
memory usage: 3.1 MB

In [6]: #check for missing values
movies.isna().sum()
ratings.isna().sum()

Out[6]:
  userId      0
  movieId     0
  rating      0
  timestamp   0
  dtype: int64

In [7]: ratings_movies = pd.merge(ratings, movies, on="movieId", how="left")

In [8]: duplicates = ratings_movies[ratings_movies.duplicated()]

In [9]: print(duplicates)

Empty DataFrame
Columns: [userId, movieId, rating, timestamp, title, genres]
Index: []

In [10]: if np.issubdtype(ratings_movies['timestamp'], np.number):
    ratings_movies['rating_ts'] = pd.to_datetime(ratings_movies['timestamp'], unit='s')
else:
    ratings_movies['rating_ts'] = pd.to_datetime(ratings_movies['timestamp'], errors='coerce')

ratings_movies[['timestamp', 'rating_ts']].head()

Out[10]:
  timestamp      rating_ts
0  964982703  2000-07-30 18:45:03
1  964981247  2000-07-30 18:20:47
2  964982224  2000-07-30 18:37:04
3  964983815  2000-07-30 19:03:35
4  964982931  2000-07-30 18:48:51

In [11]: df = ratings.merge(movies, on="movieId", how="left")
df.head(20)

Out[11]:
  userId movieId rating timestamp          title      genres rating_ts
0       1      1  4.0  964982703  Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy 2000-07-30 18:45:03
1       1      3  4.0  964981247  Grumpier Old Men (1995) Comedy|Romance 2000-07-30 18:20:47
2       1      6  4.0  964982224           Heat (1995) Action|Crime|Thriller 2000-07-30 18:37:04
3       1     47  5.0  964983815  Seven (a.k.a. Se7en) (1995) Mystery|Thriller 2000-07-30 19:03:35
4       1     50  5.0  964982931  Usual Suspects, The (1995) Crime|Mystery|Thriller 2000-07-30 18:48:51
5       1     70  3.0  964982400  From Dusk Till Dawn (1996) Action|Comedy|Horror|Thriller 2000-07-30 18:40:00
6       1     101  5.0  964980868           Bottie Rocket (1996) Adventure|Comedy|Crime|Romance 2000-07-30 18:14:28
7       1    110  4.0  964982176           Braveheart (1995) Action|Drama|War 2000-07-30 18:36:16
8       1    151  5.0  964984041           Rob Roy (1995) Action|Drama|Romance|War 2000-07-30 19:07:21
9       1    157  5.0  964984100           Canadian Bacon (1995) Comedy|War 2000-07-30 19:08:20
10      1    163  5.0  964983650           Desperado (1995) Action|Romance|Western 2000-07-30 19:00:50
11      1    216  5.0  964981208           Billy Madison (1995) Comedy 2000-07-30 18:20:08
12      1    223  3.0  964980985           Clerks (1994) Comedy 2000-07-30 18:16:25
13      1    231  5.0  964981179  Dumb & Dumber (Dumb and Dumber) (1994) Adventure|Comedy 2000-07-30 18:19:39
14      1    235  4.0  964980908           Ed Wood (1994) Comedy|Drama 2000-07-30 18:15:08
15      1    260  5.0  964981680  Star Wars: Episode IV - A New Hope (1977) Action|Adventure|Sci-Fi 2000-07-30 18:28:00
16      1    296  3.0  964982967           Pulp Fiction (1994) Comedy|Crime|Drama|Thriller 2000-07-30 18:49:27
17      1    316  3.0  964982310           Stargate (1994) Action|Adventure|Sci-Fi 2000-07-30 18:38:30
18      1    333  5.0  964981179           Tommy Boy (1995) Comedy 2000-07-30 18:19:39
19      1    349  4.0  964982563  Clear and Present Danger (1994) Action|Crime|Drama|Thriller 2000-07-30 18:42:43

In [12]: #extract movie year
def extract_year(title):
    if isinstance(title, str):
        m = re.search(r'(\d{4})(\d{2})', title)
        return int(m.group(1)) if m else np.nan
    return np.nan

In [13]: #duplicate the merged dataset and create a new column called release year
ratings_movies_feat = ratings_movies.copy()

# extract the release year from the movie title
ratings_movies_feat['release_year'] = ratings_movies_feat['title'].str.extract(r'(\d{4})').astype('int64')

In [14]: #fills empty cells with NO genre listed
ratings_movies_feat['genre'] = ratings_movies_feat['genres'].fillna('no genres listed')

In [15]: #Creates a new column called genres list
ratings_movies_feat['genre_list'] = ratings_movies_feat['genres'].apply(lambda s: s.split(',') if s != 'no genres listed' else [])

In [16]: #counts the number of items in the genres list
ratings_movies_feat['genre_count'] = ratings_movies_feat['genre_list'].apply(len)

In [17]: # 3. Primary genre (first listed)
ratings_movies_feat['primary_genre'] = ratings_movies_feat['genre_list'].apply(lambda L: L[0] if len(L)>0 else 'Unknown')

In [18]: # 4. Movie title length (helps detect complexity or naming pattern)
ratings_movies_feat['title_length'] = ratings_movies_feat['title'].apply(lambda x: len(str(x)))

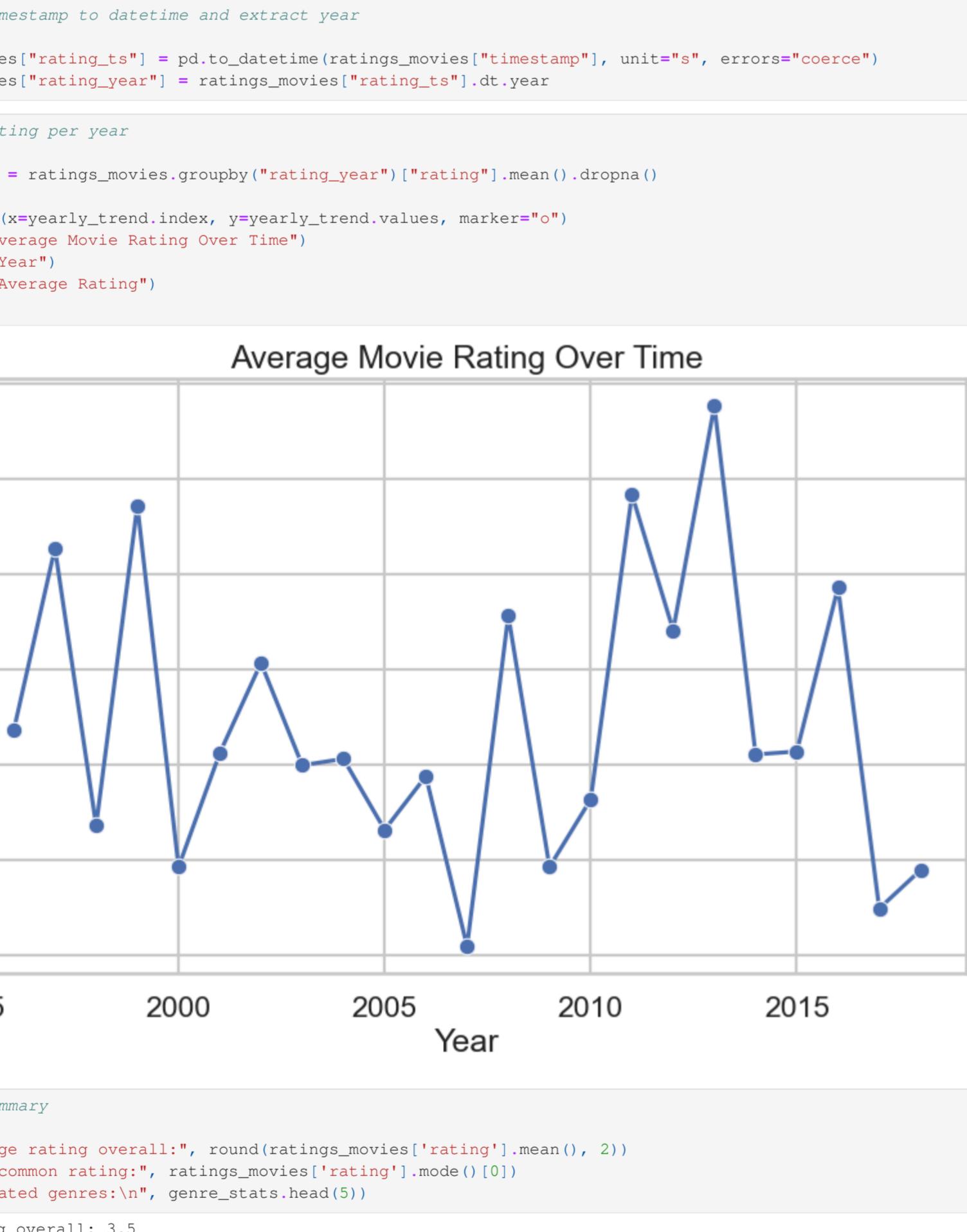
In [19]: # 5. Release decade (group movies by era)
ratings_movies_feat['release_decade'] = (ratings_movies_feat['release_year'] // 10*10).astype('int64').astype(str)*'s'

In [20]: # 6. Has sequel (detects if movie title contains a sequel number)
ratings_movies_feat['has_sequel'] = ratings_movies_feat['title'].apply(lambda x: 1 if re.search(r'\b(2-9)\b', str(x)) else 0)

In [21]: # Exploratory Data Analysis.
sns.set(style="whitegrid", context="talk")
plt.rcParams["figure.figsize"] = (10, 6)

In [22]: # Rating Distribution
print(ratings_movies['rating'].describe())
sns.histplot(ratings_movies['rating'], bins=10, kde=True)
plt.title("Distribution of Movie Ratings")
plt.xlabel("Rating")
plt.ylabel("Count")
plt.show()

count: 100836
mean: 3.503557
std: 1.042529
min: 0.500000
25%: 3.000000
50%: 3.500000
75%: 4.000000
max: 5.000000
Name: rating, dtype: float64

```

```
In [31]: # Merged ratings_movies to get my new sheet  
ratings_movies_feat.to_csv("merged_ratings_movies.csv", index=False)
```

```
In [ ]:
```