

## DATA PREPARATION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Double-click (or enter) to edit

```
Ratings = pd.read_csv('/ratings.csv') # columns: userId,movieId,rating,timestamp
movies = pd.read_csv('/movies.csv') # columns: movieId,title,genres
tags = pd.read_csv('/tags.csv') # columns: userId,movieId,tag,timestamp
links = pd.read_csv('/links.csv') # columns: movieId,imdbId,tmdbId
```

```
Ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

```
movies.head()
```

	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

```
tags.head()
```

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

```
links.head()
```

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

```
df = Ratings.merge(movies, on='movieId', how='left')
```

```
tags_count = tags.groupby('movieId').agg(
    n_tags = ('tag','count'),
    unique_tags = ('tag', lambda s: ','.join(sorted(set(s)))) )
).reset_index()
```

```
df = df.merge(tags_count[['movieId','n_tags']], on='movieId', how='left')
df = df.merge(links, on='movieId', how='left')
```

To erase the duplicates

```
print("Duplicates in ratings (full rows):", df.duplicated().sum())
df = df.drop_duplicates()
print("Missing per column:\n", df.isna().sum())

Duplicates in ratings (full rows): 0
Missing per column:
userId          0
movieId         0
rating          0
timestamp       0
title           0
genres          0
n_tags      52549
imdbId         0
tmdbId        13
dtype: int64
```

If n tags is missing

```
df['n_tags'] = df['n_tags'].fillna(0).astype(int)
```

```
print(df)

      userId  movieId  rating  timestamp               title \
0            1        1     4.0  964982703  Toy Story (1995)
1            1        3     4.0  964981247  Grumpier Old Men (1995)
2            1        6     4.0  964982224             Heat (1995)
3            1       47     5.0  964983815  Seven (a.k.a. Se7en) (1995)
4            1       50     5.0  964982931  Usual Suspects, The (1995)
...
100831      610    166534     4.0  1493848402           Split (2017)
100832      610    168248     5.0  1493850091  John Wick: Chapter Two (2017)
100833      610    168250     5.0  1494273047           Get Out (2017)
100834      610    168252     5.0  1493846352           Logan (2017)
100835      610    170875     3.0  1493846415  The Fate of the Furious (2017)

      genres  n_tags  imdbId  tmdbId
0  Adventure|Animation|Children|Comedy|Fantasy      3   114709   862.0
1          Comedy|Romance                  2   113228  15602.0
2      Action|Crime|Thriller                 0   113277   949.0
3      Mystery|Thriller                  3   114369   807.0
4  Crime|Mystery|Thriller                 6   114814   629.0
...
100831      Drama|Horror|Thriller                0   4972582  381288.0
100832      Action|Crime|Thriller                8   4425200  324552.0
100833          Horror                   0   5052448  419430.0
100834      Action|Sci-Fi                  5   3315342  263115.0
100835  Action|Crime|Drama|Thriller                0   4630562  337339.0

[100836 rows x 9 columns]
```

Start coding or [generate](#) with AI.

To convert the timestamp

```
df['date'] = pd.to_datetime(df['timestamp'], unit='s')
```

```
df['month_name'] = df['date'].dt.strftime('%B')
```

Double-click (or enter) to edit

```
# timestamps in ratings and tags are usually unix epoch seconds
df['rating_ts'] = pd.to_datetime(df['timestamp'], unit='s')
# if you merged tags' timestamp or there is a 'timestamp' column conflict, rename properly.
# create convenient temporal columns
df['rating_year'] = df['rating_ts'].dt.year
df['rating_month'] = df['rating_ts'].dt.month_name
df['rating_dayofweek'] = df['rating_ts'].dt.day_name()
df['rating_hour'] = df['rating_ts'].dt.hour
```

```
print(df)
```

	userId	movieId	rating	timestamp	title	\
0	1	1	4.0	964982703	Toy Story (1995)	
1	1	3	4.0	964981247	Grumpier Old Men (1995)	
2	1	6	4.0	964982224	Heat (1995)	
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	
4	1	50	5.0	964982931	Usual Suspects, The (1995)	
...	...	...	...	...	...	...
100831	610	166534	4.0	1493848402	Split (2017)	
100832	610	168248	5.0	1493850091	John Wick: Chapter Two (2017)	
100833	610	168250	5.0	1494273047	Get Out (2017)	
100834	610	168252	5.0	1493846352	Logan (2017)	
100835	610	170875	3.0	1493846415	The Fate of the Furious (2017)	
				genres	n_tags	imdbId
0		Adventure Animation Children Comedy Fantasy			3	114709
1		Comedy Romance			2	113228
2		Action Crime Thriller			0	113277
3		Mystery Thriller			3	114369
4		Crime Mystery Thriller			6	114814
...		...		...	...	...
100831		Drama Horror Thriller			0	4972582
100832		Action Crime Thriller			8	4425200
100833		Horror			0	5052448
100834		Action Sci-Fi			5	3315342
100835		Action Crime Drama Thriller			0	4630562
	tmdbId	rating_ts	rating_year	\		
0	862.0	2000-07-30 18:45:03	2000			
1	15602.0	2000-07-30 18:20:47	2000			
2	949.0	2000-07-30 18:37:04	2000			
3	807.0	2000-07-30 19:03:35	2000			
4	629.0	2000-07-30 18:48:51	2000			
...	...	...	...	...		
100831	381288.0	2017-05-03 21:53:22	2017			
100832	324552.0	2017-05-03 22:21:31	2017			
100833	419430.0	2017-05-08 19:50:47	2017			
100834	263115.0	2017-05-03 21:19:12	2017			
100835	337339.0	2017-05-03 21:20:15	2017			
		rating_month	rating_dayofweek	\		
0	<bound method PandasDelegate._add_delegate_acc...		Sunday			
1	<bound method PandasDelegate._add_delegate_acc...		Sunday			
2	<bound method PandasDelegate._add_delegate_acc...		Sunday			
3	<bound method PandasDelegate._add_delegate_acc...		Sunday			
4	<bound method PandasDelegate._add_delegate_acc...		Sunday			
...	...	...	...	...		
100831	<bound method PandasDelegate._add_delegate_acc...		Wednesday			
100832	<bound method PandasDelegate._add_delegate_acc...		Wednesday			
100833	<bound method PandasDelegate._add_delegate_acc...		Monday			
100834	<bound method PandasDelegate._add_delegate_acc...		Wednesday			
100835	<bound method PandasDelegate._add_delegate_acc...		Wednesday			
	rating_hour					
0	18					
1	18					
2	18					
3	19					
4	18					

```
# Extract specific parts of the date
df['year'] = df['date'].dt.year # e.g., 2020
df['month'] = df['date'].dt.month # e.g., 7
df['day'] = df['date'].dt.day # e.g., 30

# Optional: Extract month name and day name
df['month_name'] = df['date'].dt.strftime('%B') # July, August...
df['day_name'] = df['date'].dt.strftime('%A') # Monday, Tuesday...
```

```
# Preview result
print(df[['date', 'year', 'month', 'day', 'month_name', 'day_name']].head())
```

	date	year	month	day	month_name	day_name
0	2000-07-30 18:45:03	2000	7	30	July	Sunday
1	2000-07-30 18:20:47	2000	7	30	July	Sunday
2	2000-07-30 18:37:04	2000	7	30	July	Sunday
3	2000-07-30 19:03:35	2000	7	30	July	Sunday
4	2000-07-30 18:48:51	2000	7	30	July	Sunday

```
df['timestamp'] = pd.to_datetime(df['timestamp'], unit='s')
```

```
# Preview the changes
print(df.head())
```

```

    userId  movieId  rating      timestamp          title \
0        1        1    4.0 2000-07-30 18:45:03      Toy Story (1995)
1        1        3    4.0 2000-07-30 18:20:47  Grumpier Old Men (1995)
2        1        6    4.0 2000-07-30 18:37:04           Heat (1995)
3        1       47    5.0 2000-07-30 19:03:35  Seven (a.k.a. Se7en) (1995)
4        1       50    5.0 2000-07-30 18:48:51  Usual Suspects, The (1995)

                           genres  n_tags  imdbId  tmdbId \
0  Adventure|Animation|Children|Comedy|Fantasy            3   114709   862.0
1                  Comedy|Romance                2   113228  15602.0
2             Action|Crime|Thriller                0   113277   949.0
3            Mystery|Thriller                 3   114369   807.0
4  Crime|Mystery|Thriller                6   114814   629.0

      rating_ts  rating_year \
0 2000-07-30 18:45:03      2000
1 2000-07-30 18:20:47      2000
2 2000-07-30 18:37:04      2000
3 2000-07-30 19:03:35      2000
4 2000-07-30 18:48:51      2000

                           rating_month  rating_dayofweek \
0 <bound method PandasDelegate._add_delegate_acc...           Sunday
1 <bound method PandasDelegate._add_delegate_acc...           Sunday
2 <bound method PandasDelegate._add_delegate_acc...           Sunday
3 <bound method PandasDelegate._add_delegate_acc...           Sunday
4 <bound method PandasDelegate._add_delegate_acc...           Sunday

      rating_hour          date month_name  year  month  day day_name
0          18 2000-07-30 18:45:03     July  2000     7   30 Sunday
1          18 2000-07-30 18:20:47     July  2000     7   30 Sunday
2          18 2000-07-30 18:37:04     July  2000     7   30 Sunday
3          19 2000-07-30 19:03:35     July  2000     7   30 Sunday
4          18 2000-07-30 18:48:51     July  2000     7   30 Sunday

```

## FEATURING ENGINEERING

```

import re
def extract_year(title):
    m = re.search(r'\((\d{4})\)\s*$', str(title))
    return int(m.group(1)) if m else np.nan
df['release_year'] = df['title'].apply(extract_year).astype('Int64')

```

```
print(df)
```

```

    userId  movieId  rating      timestamp          title \
0        1        1    4.0 2000-07-30 18:45:03      Toy Story (1995)
1        1        3    4.0 2000-07-30 18:20:47  Grumpier Old Men (1995)
2        1        6    4.0 2000-07-30 18:37:04           Heat (1995)
3        1       47    5.0 2000-07-30 19:03:35  Seven (a.k.a. Se7en) (1995)
4        1       50    5.0 2000-07-30 18:48:51  Usual Suspects, The (1995)
...
...
100831     610    166534    4.0 2017-05-03 21:53:22
100832     610    168248    5.0 2017-05-03 22:21:31
100833     610    168250    5.0 2017-05-08 19:50:47
100834     610    168252    5.0 2017-05-03 21:19:12
100835     610    170875    3.0 2017-05-03 21:20:15

                           title \
0                  Toy Story (1995)
1            Grumpier Old Men (1995)
2                  Heat (1995)
3            Seven (a.k.a. Se7en) (1995)
4  Usual Suspects, The (1995)
...
...
100831            Split (2017)
100832  John Wick: Chapter Two (2017)
100833            Get Out (2017)
100834            Logan (2017)
100835  The Fate of the Furious (2017)

                           genres  n_tags  imdbId \
0  Adventure|Animation|Children|Comedy|Fantasy            3   114709
1                  Comedy|Romance                2   113228
2             Action|Crime|Thriller                0   113277
3            Mystery|Thriller                 3   114369
4  Crime|Mystery|Thriller                6   114814
...
...
100831  Drama|Horror|Thriller                0   4972582
100832  Action|Crime|Thriller                8   4425200
100833            Horror                 0   5052448
100834  Action|Sci-Fi                  5   3315342
100835  Action|Crime|Drama|Thriller                0   4630562

      tmdbId  rating_ts ...

```

```

0      862.0 2000-07-30 18:45:03 ...
1     15602.0 2000-07-30 18:20:47 ...
2      949.0 2000-07-30 18:37:04 ...
3      807.0 2000-07-30 19:03:35 ...
4      629.0 2000-07-30 18:48:51 ...
...
100831  381288.0 2017-05-03 21:53:22 ...
100832  324552.0 2017-05-03 22:21:31 ...
100833  419430.0 2017-05-08 19:50:47 ...
100834  263115.0 2017-05-03 21:19:12 ...
100835  337339.0 2017-05-03 21:20:15 ...

                           rating_month rating_dayofweek \
0           <bound method PandasDelegate._add_delegate_acc... Sunday
1           <bound method PandasDelegate._add_delegate_acc... Sunday
2           <bound method PandasDelegate._add_delegate_acc... Sunday
3           <bound method PandasDelegate._add_delegate_acc... Sunday
4           <bound method PandasDelegate._add_delegate_acc... Sunday

```

```
# 2) num_genres
df['num_genres'] = df['genres'].fillna('').apply(lambda s: 0 if s == '(no genres listed)' or s=='' else len(s.split('|')))
```

```
print(df['num_genres'])
```

```

0      5
1      2
2      3
3      2
4      3
...
100831  3
100832  3
100833  1
100834  2
100835  4
Name: num_genres, Length: 100836, dtype: int64

```

```
# 3) movie_popularity (total number of ratings for the movie)
movie_rating_counts = df.groupby('movieId')['rating'].count().rename('movie_rating_count')
df = df.merge(movie_rating_counts, on='movieId', how='left')
```

```
print(movie_rating_counts)
```

```

movieId
1      215
2      110
3      52
4       7
5      49
...
193581   1
193583   1
193585   1
193587   1
193609   1
Name: movie_rating_count, Length: 9724, dtype: int64

```

```
# 4) movie_avg_rating (mean rating per movie)
movie_avg = df.groupby('movieId')['rating'].mean().rename('movie_avg_rating')
df = df.merge(movie_avg, on='movieId', how='left')
```

```
print(movie_avg)
```

```

movieId
1      3.920930
2      3.431818
3      3.259615
4      2.357143
5      3.071429
...
193581  4.000000
193583  3.500000
193585  3.500000
193587  3.500000
193609  4.000000
Name: movie_avg_rating, Length: 9724, dtype: float64

```

```
# 5) rating_age (years between release and rating)
# if release_year missing, leave NaN
df['rating_age_years'] = (df['rating_year'] - df['release_year']).astype('Float64')
```

```
print(df['rating_age_years'])
```

```

0      5.0
1      5.0
2      5.0
3      5.0
4      5.0
...
100831   0.0
100832   0.0
100833   0.0
100834   0.0
100835   0.0
Name: rating_age_years, Length: 100836, dtype: Float64

```

```
# 6) is_recent_release (binary): movie released within 5 years of rating
df['is_recent_release'] = (df['rating_age_years'] <= 5).astype('Int64').fillna(0)
```

```
print(df['is_recent_release'])
```

```

0      1
1      1
2      1
3      1
4      1
...
100831   1
100832   1
100833   1
100834   1
100835   1
Name: is_recent_release, Length: 100836, dtype: Int64

```

```
# 7) user_activity (number of ratings a user made) -> useful for user bias
user_counts = df.groupby('userId')['rating'].count().rename('user_rating_count')
df = df.merge(user_counts, on='userId', how='left')
```

```
print(user_counts)
```

```

userId
1      232
2      29
3      39
4      216
5      44
...
606    1115
607    187
608    831
609    37
610    1302
Name: user_rating_count, Length: 610, dtype: int64

```

```
# Reorder or select columns
cols_keep = ['userId', 'movieId', 'title', 'genres', 'release_year', 'num_genres',
             'rating', 'rating_ts', 'rating_year', 'rating_dayofweek',
             'rating_hour', 'movie_rating_count', 'movie_avg_rating', 'user_rating_count',
             'rating_age_years', 'is_recent_release', 'n_tags', 'imdbId', 'tmdbId', 'date', 'year', 'month_name', 'day_name']
df_clean = df[cols_keep].copy()
```

```
print(df_clean)
```

	userId	movieId	title	genres	release_year	num_genres
0	1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	1995	5
1	1	3	Grumpier Old Men (1995)	Comedy Romance	1995	2
2	1	6	Heat (1995)	Action Crime Thriller	1995	3
3	1	47	Seven (a.k.a. Se7en) (1995)	Mystery Thriller	1995	2
4	1	50	The Usual Suspects, The (1995)	Crime Mystery Thriller	1995	3
...	...	...	...	...	...	...
100831	610	166534	Split (2017)	Drama Horror Thriller	2017	3
100832	610	168248	John Wick: Chapter Two (2017)	Action Crime Thriller	2017	3
100833	610	168250	Get Out (2017)	Horror	2017	1
100834	610	168252	Logan (2017)			
100835	610	170875	The Fate of the Furious (2017)			

100834		Action Sci-Fi	2017	2		
100835		Action Crime Drama Thriller	2017	4		
<hr/>						
0	rating	rating_ts	rating_year	rating_dayofweek		
0	4.0	2000-07-30 18:45:03	2000	Sunday		
1	4.0	2000-07-30 18:20:47	2000	Sunday		
2	4.0	2000-07-30 18:37:04	2000	Sunday		
3	5.0	2000-07-30 19:03:35	2000	Sunday		
4	5.0	2000-07-30 18:48:51	2000	Sunday		
...	...	...	...	...		
100831	4.0	2017-05-03 21:53:22	2017	Wednesday		
100832	5.0	2017-05-03 22:21:31	2017	Wednesday		
100833	5.0	2017-05-08 19:50:47	2017	Monday		
100834	5.0	2017-05-03 21:19:12	2017	Wednesday		
100835	3.0	2017-05-03 21:20:15	2017	Wednesday		
<hr/>						
0	user_rating_count	rating_age_years	is_recent_release	n_tags		
0	232	5.0	1	3		
1	232	5.0	1	2		
2	232	5.0	1	0		
3	232	5.0	1	3		
4	232	5.0	1	6		
...	...	...	...	...		
100831	1302	0.0	1	0		
100832	1302	0.0	1	8		
100833	1302	0.0	1	0		
100834	1302	0.0	1	5		
100835	1302	0.0	1	0		
<hr/>						
0	imdbId	tmdbId	date	year	month_name	day_name
0	114709	862.0	2000-07-30 18:45:03	2000	July	Sunday
1	113228	15602.0	2000-07-30 18:20:47	2000	July	Sunday
2	113277	949.0	2000-07-30 18:37:04	2000	July	Sunday
3	114369	807.0	2000-07-30 19:03:35	2000	July	Sunday
4	114814	629.0	2000-07-30 18:48:51	2000	July	Sunday

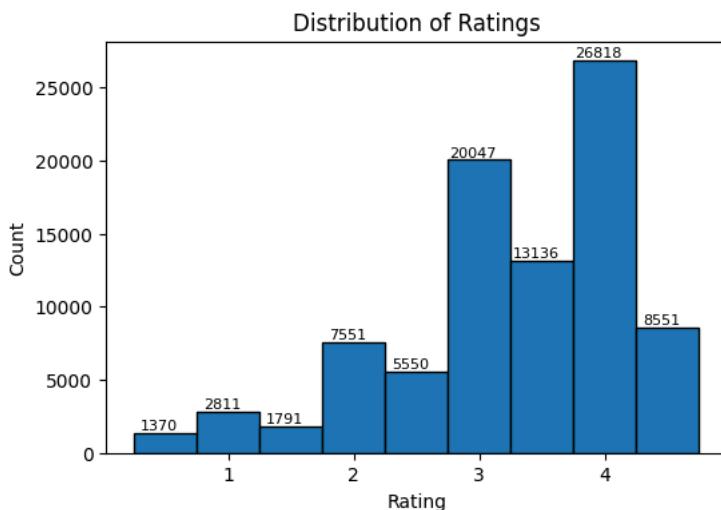
## EXPLANATORY DATA ANALYSIS

### Distribution Rating

```
plt.figure(figsize=(6,4))
counts, bins, patches = plt.hist(df_clean['rating'], bins=np.arange(0.25,5.25,0.5), edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.title('Distribution of Ratings')

# Add labels on top of each bar
for count, x in zip(counts, bins):
    if count > 0:
        plt.text(x + 0.2, count + 50, int(count), ha='center', va='bottom', fontsize=8)

plt.show()
```



```
# Make sure your dataset has 'genres' and 'rating' columns
# Step 1: Split multiple genres into individual rows
genre_exploded = df_clean.assign(genre=df_clean['genres'].str.split('|')).explode('genre')
```

```
# Step 2: Compute aggregated statistics per genre
genre_stats = genre_exploded.groupby('genre').agg(
    avg_rating=('rating','mean'),
    count_ratings=('rating','count'),
    num_movies=('movieId','nunique')
).sort_values('count_ratings', ascending=False)

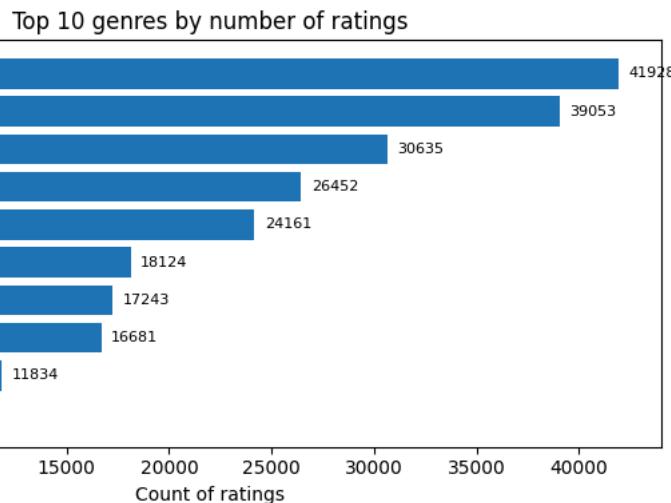
# Step 3: Select the top 10 genres
top10 = genre_stats.head(10)
```

## Top 10 genre by number of rating

```
plt.figure(figsize=(8,4))
plt.barh(top10.index[::-1], top10['count_ratings'][::-1])
plt.xlabel('Count of ratings')
plt.title('Top 10 genres by number of ratings')

# Add labels on bars
for index, value in enumerate(top10['count_ratings'][::-1]):
    plt.text(value + 500, index, str(int(value)), va='center', fontsize=8)

plt.tight_layout()
plt.show()
```



Double-click (or enter) to edit

## Top 5 Genres by Ratings share

## Recent vs Old Releases

```
# Group by recency flag and compute average rating
recency_stats = df_clean.groupby('is_recent_release').agg(
    avg_rating=('rating','mean'),
    count_ratings=('rating','count')
).reset_index()

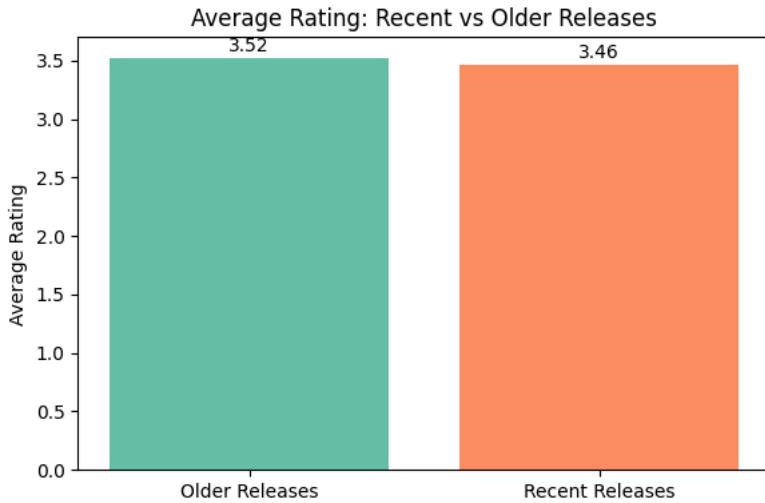
# Map values for readability
recency_stats['label'] = recency_stats['is_recent_release'].map({0: 'Older Releases', 1: 'Recent Releases'})

# Plot average rating comparison
fig, ax = plt.subplots(figsize=(6,4))
bars = ax.bar(recency_stats['label'], recency_stats['avg_rating'], color=['#66c2a5','#fc8d62'])
ax.set_ylabel('Average Rating')
ax.set_title('Average Rating: Recent vs Older Releases')

# Add labels above bars
for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 0.02, f'{height:.2f}', ha='center', va='bottom', fontsize=10)

plt.tight_layout()
```

plt.show()



## Genre Popularity Over Time

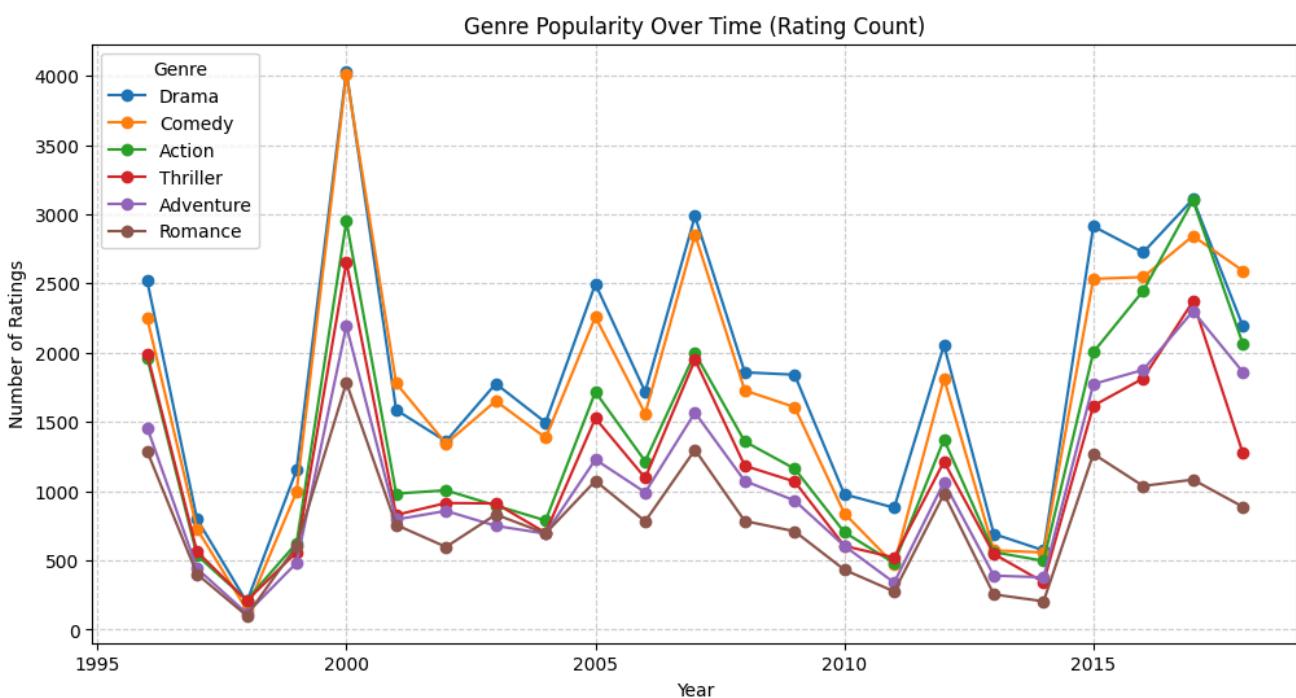
```
# Split genres
df_genre = df_clean.assign(genre=df_clean['genres'].str.split('|')).explode('genre')

# Count ratings per genre per year
genre_year = df_genre.groupby(['year', 'genre']).size().reset_index(name='rating_count')

# Select top 6 genres overall
top_genres = genre_year.groupby('genre')['rating_count'].sum().nlargest(6).index
subset = genre_year[genre_year['genre'].isin(top_genres)]

plt.figure(figsize=(12,6))
for genre in top_genres:
    data = subset[subset['genre']==genre]
    plt.plot(data['year'], data['rating_count'], marker='o', label=genre)

plt.title('Genre Popularity Over Time (Rating Count)')
plt.xlabel('Year')
plt.ylabel('Number of Ratings')
plt.legend(title='Genre')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

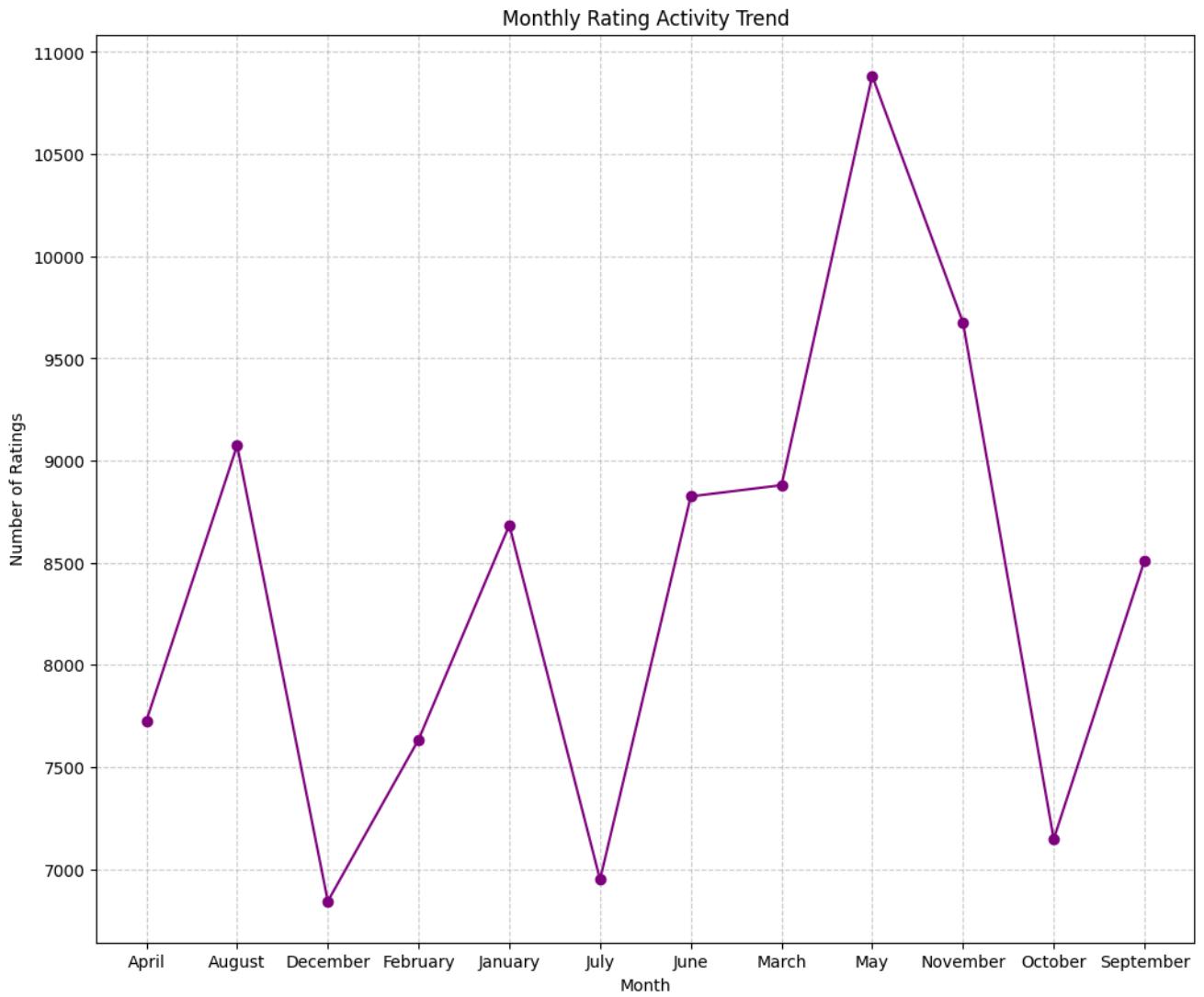


Double-click (or enter) to edit

```

monthly_stats = df_clean.groupby('month_name')['rating'].count().reset_index()
plt.figure(figsize=(12,10))
plt.plot(monthly_stats['month_name'], monthly_stats['rating'], marker='o', color='purple')
plt.title('Monthly Rating Activity Trend')
plt.xlabel('Month')
plt.ylabel('Number of Ratings')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

```



## ▼ Average Rating by Year

```

# Extract year from timestamp
df_clean['rating_year'] = df_clean['rating_ts'].dt.year

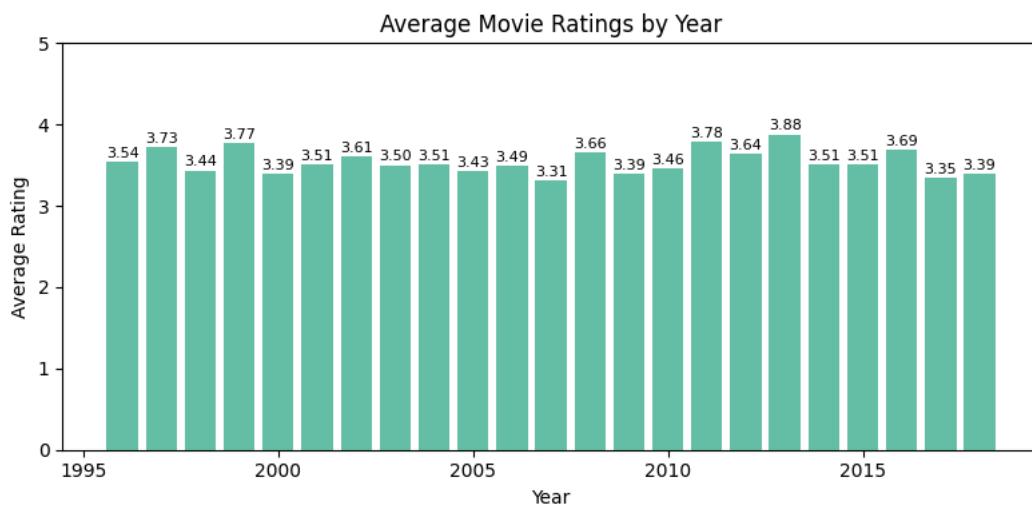
# Group by year and compute average rating
yearly_trend = df_clean.groupby('rating_year')['rating'].mean()

plt.figure(figsize=(8,4))
bars = plt.bar(yearly_trend.index, yearly_trend.values, color='#66c2a5')
plt.title('Average Movie Ratings by Year')
plt.xlabel('Year')
plt.ylabel('Average Rating')
plt.ylim(0, 5)

# Add data labels above each bar
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.02, f'{height:.2f}',
             ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()

```



Start coding or [generate](#) with AI.