In [12]:

```python
import pandas as pd

#Path to the folder where dat files exists
# Common directory path
common_path = "C:/Users/dande/Desktop/Spring2024/DMT/Assignment 1/ml-1m/ml-1m/"

# Individual file paths
movies_data = common_path + "movies.dat"
ratings_data = common_path + "ratings.dat"
user_data = common_path + "users.dat"

# Function to read .dat file with specified encoding
def read_dat_file(file_path, encoding='ISO-8859-1'):
    return pd.read_csv(file_path, sep='::', engine='python', names=['MovieID', 'Title',


# Read and process the movies file with different encoding
movies = read_dat_file(movies_data)
movies['Genres'] = movies['Genres'].apply(lambda x: x.split('|'))

# Read and process the ratings file
ratings = pd.read_csv(ratings_data, sep='::', engine='python', names=['UserID', 'MovieID

# Read and process the users file
users = pd.read_csv(user_data, sep='::', engine='python', names=['UserID', 'Gender', 'Ag
male_users_above_25 = users[(users['Gender'] == 'M') & (users['Age'] > 25)]['UserID']

# Filter ratings by male users above 25
filtered_ratings = ratings[ratings['UserID'].isin(male_users_above_25)]

# Explode the genres for aggregation
movies_exploded = movies.explode('Genres')

# Merge the datasets
merged_data = pd.merge(filtered_ratings, movies_exploded, on='MovieID')

# Aggregate ratings by genre
genre_ratings = merged_data.groupby('Genres')['Rating'].mean()

# Display the results
print("An aggregate of movie ratings by men of age above 25 for each genre:\n",genre_rat
```

```
An aggregate of movie ratings by men of age above 25 for each genre:
 Genres
Action         3.554547
Adventure      3.538637
Animation      3.721569
Children's     3.475314
Comedy         3.565456
Crime          3.764249
Documentary    3.950192
Drama          3.812309
Fantasy        3.490408
Film-Noir      4.117140
Horror         3.241089
Musical        3.700242
Mystery        3.759347
Romance        3.659748
Sci-Fi         3.509693
Thriller       3.644025
War            3.940634
Western        3.708494
Name: Rating, dtype: float64
```

New Section

In [5]:
```
# Merge the datasets
merged_data = pd.merge(ratings, movies, on='MovieID')

# Count the number of ratings for each movie and sort
top_movies = merged_data.groupby('Title').size().sort_values(ascending=False).head(5)

# Display the results
print("The top 5 ranked movies by the most number of ratings\n", top_movies)
```

```
The top 5 ranked movies by the most number of ratings
 Title
American Beauty (1999)                                3428
Star Wars: Episode IV - A New Hope (1977)             2991
Star Wars: Episode V - The Empire Strikes Back (1980)    2990
Star Wars: Episode VI - Return of the Jedi (1983)     2883
Jurassic Park (1993)                                  2672
dtype: int64
```

In [23]:
```
# Define age groups
age_bins = [0, 18, 30, 50, 70, float('inf')]
age_labels = ['<18', '18-30', '30-50', '50-70', '>70']
users['AgeGroup'] = pd.cut(users['Age'], bins=age_bins, labels=age_labels, right=False)

# Merge and group by AgeGroup
merged_data = pd.merge(ratings, users, on='UserID')
age_group_ratings = merged_data.groupby('AgeGroup')['Rating'].mean()

print(age_group_ratings)
```

```
AgeGroup
<18      3.549520
18-30    3.533299
30-50    3.624050
50-70    3.732677
>70           NaN
Name: Rating, dtype: float64
C:\Users\dande\AppData\Local\Temp\ipykernel_3724\1615720401.py:8: FutureWarning: The def
ault of observed=False is deprecated and will be changed to True in a future version of
pandas. Pass observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  age_group_ratings = merged_data.groupby('AgeGroup')['Rating'].mean()
```

In [13]:
```
# Extract year from movie titles
movies['Year'] = movies['Title'].str.extract(r'\((\d{4})\)')

# Filter movies released in the year 2000
movies_2000 = movies[movies['Year'] == '2000']

# Merge datasets
merged_data = pd.merge(pd.merge(movies_2000, ratings, on='MovieID'), users, on='UserID')

# Define age groups
age_groups = {'Under 18': [0, 18], '19 to 45': [19, 45], 'Above 45': [46, 150]}

# Count unique movies rated by each age group
unique_movies_rated = {}
for group, ages in age_groups.items():
    age_filtered = merged_data[(merged_data['Age'] >= ages[0]) & (merged_data['Age'] <=
    unique_movies_count = age_filtered['MovieID'].nunique()
    unique_movies_rated[group] = unique_movies_count
```

```
print("Unique movies rated by different age groups are:\n")
for group, count in unique_movies_rated.items():
    print(f"{group}: {count}")
```

```
Unique movies rated by different age groups are:

Under 18: 145
19 to 45: 147
Above 45: 131
```

In [14]:

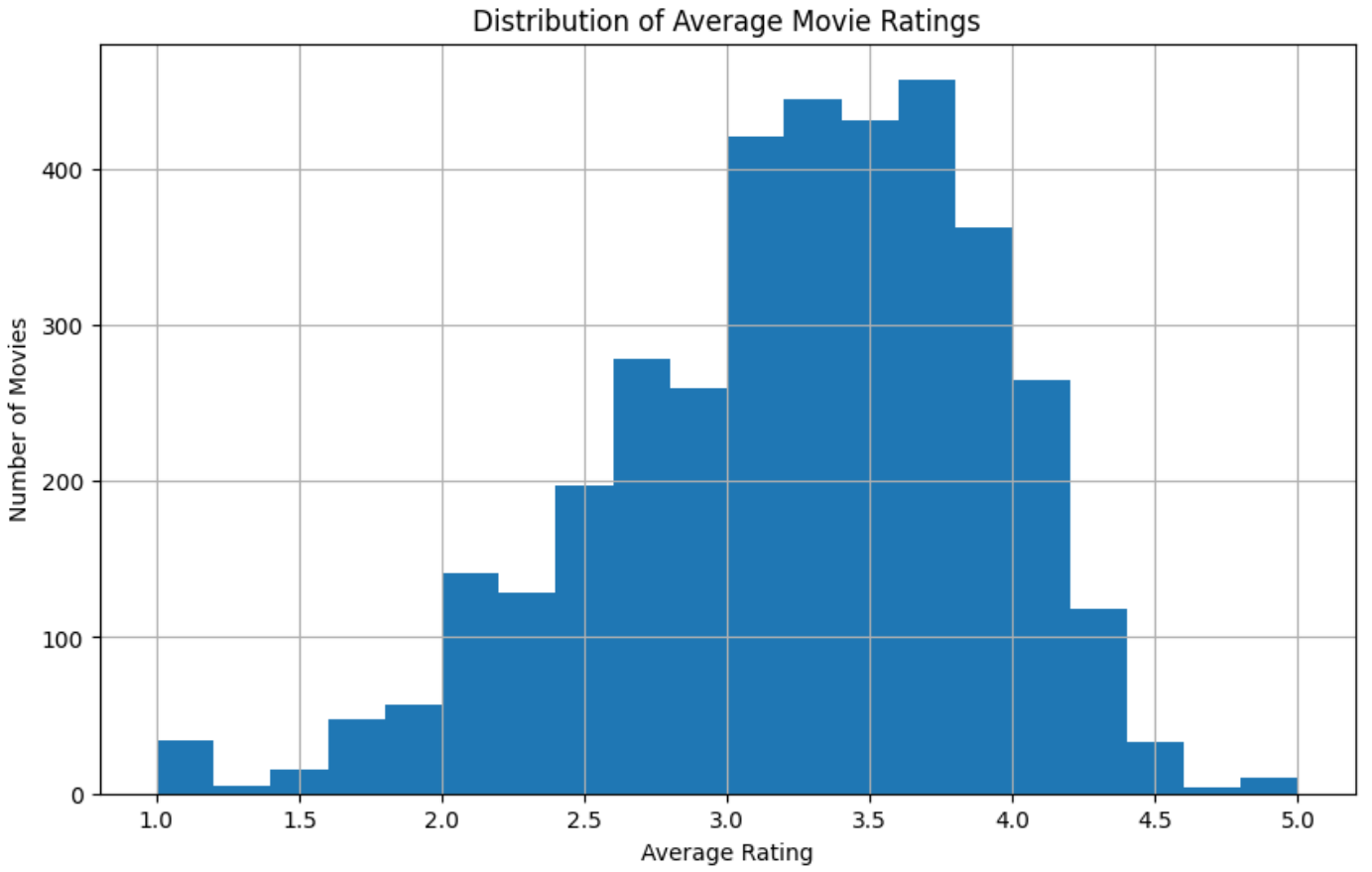
```
def find_similarly_rated_movies(user_id, movie_id):
    """Find movies rated similarly by the same user and return their IDs, titles, and ra
    try:
        user_rating = ratings[(ratings['UserID'] == user_id) & (ratings['MovieID'] == mo
        similarly_rated = ratings[(ratings['UserID'] == user_id) & (ratings['Rating'] ==
        similarly_rated_with_titles = similarly_rated.merge(movies, on='MovieID')
        return [(row['MovieID'], row['Title'], row['Rating']) for index, row in similarl
    except IndexError:
        return "Rating not found for this user and movie combination."

# Example usage
results = find_similarly_rated_movies(1, 3408)
if isinstance(results, list):
    for movie_id, title, rating in results:
        print(f"Movie ID: {movie_id}, Title: {title}, Rating: {rating}")
else:
    print(results)
```

```
Movie ID: 594, Title: Snow White and the Seven Dwarfs (1937), Rating: 4
Movie ID: 919, Title: Wizard of Oz, The (1939), Rating: 4
Movie ID: 938, Title: Gigi (1958), Rating: 4
Movie ID: 2398, Title: Miracle on 34th Street (1947), Rating: 4
Movie ID: 2918, Title: Ferris Bueller's Day Off (1986), Rating: 4
Movie ID: 2791, Title: Airplane! (1980), Rating: 4
Movie ID: 2018, Title: Bambi (1942), Rating: 4
Movie ID: 2797, Title: Big (1988), Rating: 4
Movie ID: 1097, Title: E.T. the Extra-Terrestrial (1982), Rating: 4
Movie ID: 1721, Title: Titanic (1997), Rating: 4
Movie ID: 1545, Title: Ponette (1996), Rating: 4
Movie ID: 2294, Title: Antz (1998), Rating: 4
Movie ID: 3186, Title: Girl, Interrupted (1999), Rating: 4
Movie ID: 1566, Title: Hercules (1997), Rating: 4
Movie ID: 588, Title: Aladdin (1992), Rating: 4
Movie ID: 1907, Title: Mulan (1998), Rating: 4
Movie ID: 783, Title: Hunchback of Notre Dame, The (1996), Rating: 4
Movie ID: 2762, Title: Sixth Sense, The (1999), Rating: 4
Movie ID: 1962, Title: Driving Miss Daisy (1989), Rating: 4
Movie ID: 2692, Title: Run Lola Run (Lola rennt) (1998), Rating: 4
Movie ID: 260, Title: Star Wars: Episode IV - A New Hope (1977), Rating: 4
Movie ID: 1207, Title: To Kill a Mockingbird (1962), Rating: 4
Movie ID: 531, Title: Secret Garden, The (1993), Rating: 4
Movie ID: 3114, Title: Toy Story 2 (1999), Rating: 4
Movie ID: 608, Title: Fargo (1996), Rating: 4
Movie ID: 1246, Title: Dead Poets Society (1989), Rating: 4
```

In [16]:

```
import matplotlib.pyplot as plt

# Calculate average rating for each movie
average_ratings = ratings.groupby('MovieID')['Rating'].mean()

# Convert Series to DataFrame
average_ratings_df = average_ratings.to_frame().reset_index()

# Join with movies data to get titles
average_ratings_df = average_ratings_df.join(movies.set_index('MovieID'), on='MovieID')
```

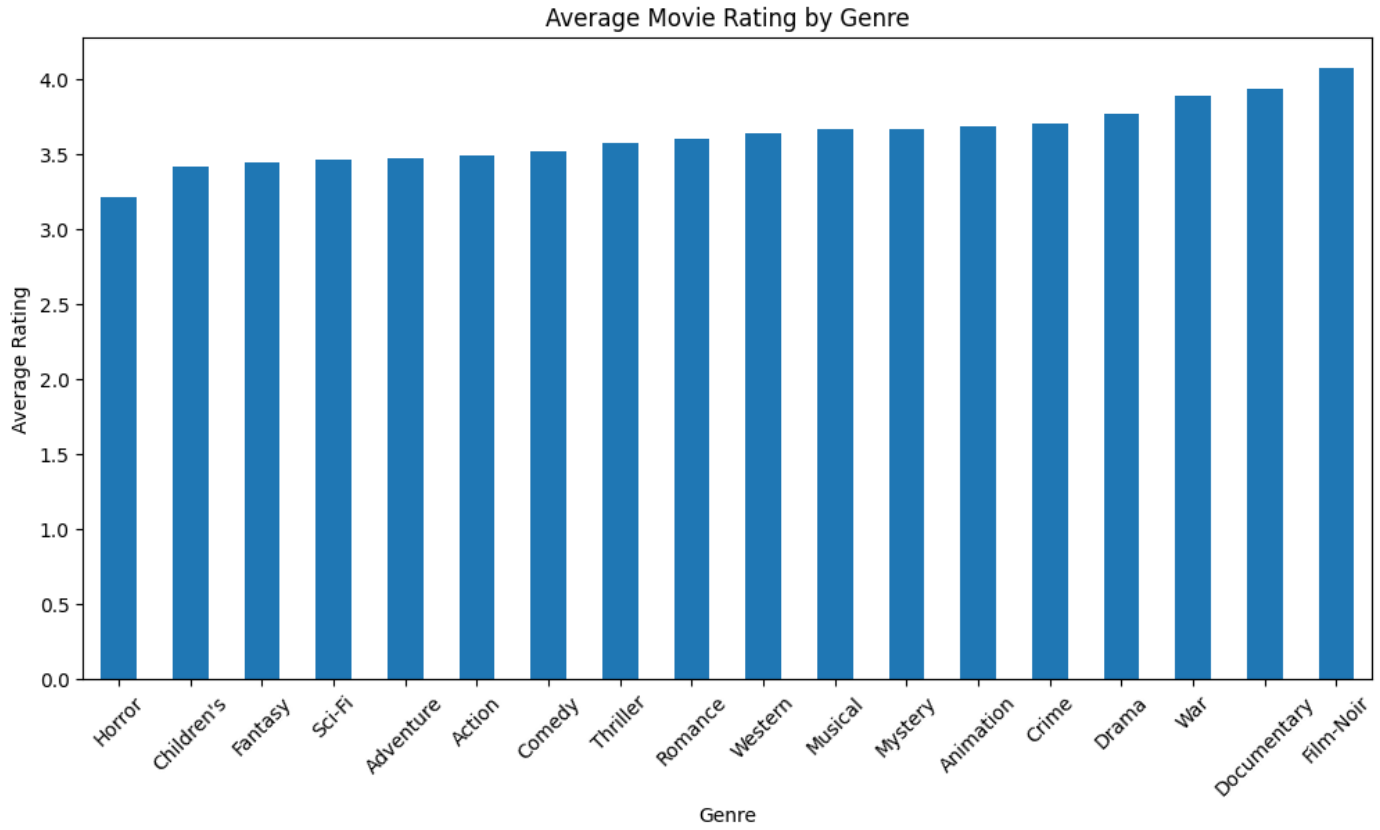
```
# Plot
plt.figure(figsize=(10, 6))
average_ratings_df['Rating'].hist(bins=20)
plt.title('Distribution of Average Movie Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Number of Movies')
plt.show()
```



In [64]:

In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt

#Path to the folder where dat files exists
# Common directory path
common_path = "C:/Users/dande/Desktop/Spring2024/DMT/Assignment 1/ml-1m/ml-1m/"

# Individual file paths
movies_data = common_path + "movies.dat"
ratings_data = common_path + "ratings.dat"

# Function to read .dat file
def read_dat_file(file_path, names, encoding='ISO-8859-1'):
    return pd.read_csv(file_path, sep='::', engine='python', names=names, encoding=encod

# Read data
movies = read_dat_file(movies_data, names=['MovieID', 'Title', 'Genres'])
ratings = read_dat_file(ratings_data, names=['UserID', 'MovieID', 'Rating', 'Timestamp']

# Convert 'Genres' to string and handle missing values
movies['Genres'] = movies['Genres'].astype(str)

# Split and explode genres
movies['Genres'] = movies['Genres'].str.split('|')
exploded_genres = movies.explode('Genres')
```

```
# Merge datasets
merged_data = pd.merge(ratings, exploded_genres, on='MovieID')

# Calculate average rating for each genre
avg_ratings_per_genre = merged_data.groupby('Genres')['Rating'].mean().sort_values()

# Plot
plt.figure(figsize=(12, 6))
avg_ratings_per_genre.plot(kind='bar')
plt.title('Average Movie Rating by Genre')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.xticks(rotation=45)
plt.show()
```



Observation from the above bar chart

The bar chart shows that people tend to rate movies from all genres pretty similarly, with no one genre standing out as really bad or really good. Horror movies got slightly lower ratings, which might mean they're not as popular, or they just get tougher reviews. Documentaries scored a bit higher, suggesting that the people who watch them really like them. Overall, most movie ratings are around the middle range, not too high or too low.