Assignment 1

Title: Exploring the MovieLens 1M Dataset

Name: Pranjal Rane NUID: 002756852

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Importing Data and Exploratory Analysis

In [2]:

```
unames = ['user_id', 'gender', 'age', 'occupation', 'zip']
users = pd.read_table('ml-1m/users.dat', sep='::', header=None, names=unames, engine='py
thon', encoding='latin-1')

rnames = ['user_id', 'movie_id', 'rating', 'timestamp']
ratings = pd.read_table('ml-1m/ratings.dat', sep='::', header=None, names=rnames, engine
='python',encoding='latin-1')

mnames = ['movie_id', 'title', 'genres']
movies = pd.read_table('ml-1m/movies.dat', sep='::', header=None, names=mnames, engine='
python',encoding='latin-1')
```

In [3]:

```
users[:5]
```

Out[3]:

	user_id	gender	age	occupation	zip
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

In [4]:

```
ratings[:5]
```

Out[4]:

	user_id	movie_id	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

In [5]:

```
movies[:5]
```

Out[5]:

	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

Merging the three tables

In [6]:

```
data = pd.merge(pd.merge(ratings, users), movies)
```

In [7]:

```
data.head(1)
```

Out[7]:

	user_id	movie_id	rating	timestamp	gender	age	occupation	zip	title	genres
0	1	1193	5	978300760	F	1	10	48067	One Flew Over the Cuckoo's Nest (1975)	Drama

Objective 1

An aggregate of movie ratings by men of age above 25 for each particular genre

In [8]:

```
filtered_data = data[(data['gender'] == 'M') & (data['age'] > 25)]
filtered_data = filtered_data.assign(genres=filtered_data['genres'].str.split('|')).explode('genres')
genre_ratings = filtered_data.groupby('genres')['rating'].mean().reset_index()
genre_ratings_sorted = genre_ratings.sort_values(by='rating', ascending=False)

genre_ratings_sorted
```

Out[8]:

	genres	rating
9	Film-Noir	4.117140
6	Documentary	3.950192
16	War	3.940634
7	Drama	3.812309
5	Crime	3.764249
12	Mystery	3.759347
2	Animation	3.721569
17	Western	3.708494
11	Musical	3.700242
13	Romance	3.659748
15	Thriller	3.644025
4	Comedy	3.565456

	genres	rating
0	Action	3.5[illegible]
1	Adventure	3.538637
14	Sci-Fi	3.509693
8	Fantasy	3.490408
3	Children's	3.475314
10	Horror	3.241089

Objective 2

The top 5 ranked movies by the most number of ratings (not the highest rating)

In [9]:

```
most_rated = data.groupby('title').size().sort_values(ascending=False)[:5]
most_rated
```

Out[9]:

```
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
```

Objective 3

Average movie ratings between users of different age groups (<18, 18-30, 30-50, 50-70, 70>)

In [10]:

```
bins = [0, 18, 30, 50, 70, float('inf')]
labels = ['<18', '18-30', '30-50', '50-70', '70>']
data['age_group'] = pd.cut(data['age'], bins=bins, labels=labels, right=False)
age_group_ratings = data.groupby('age_group')['rating'].mean().reset_index()

age_group_ratings
```

Out[10]:

	age_group	rating
0	<18	3.549520
1	18-30	3.533299
2	30-50	3.624050
3	50-70	3.732677
4	70>	NaN

Objective 4

Pick a movie of your choice and for all movies of the same year, provide a breakdown of the number of unique movies rated by 3 ranges of age of reviewers (a) under 18 (b) 19 to 45 (c) Above 45.

In [11]:

```
chosen_movie = "Dumb & Dumber (1994)"

data['release_year'] = data['title'].str.extract(r'\((\d{4})\)').astype(int)

chosen_movie_year = data[data['title'] == chosen_movie]['release_year'].values[0]
```

```
same_year_movies = data[data['release_year'] == chosen_movie_year]

bins = [0, 18, 45, float('inf')]
labels = ['Under 18', '19 to 45', 'Above 45']
same_year_movies['age_group'] = pd.cut(same_year_movies['age'], bins=bins, labels=labels, right=False)

age_group_movie_counts = same_year_movies.groupby('age_group')['movie_id'].nunique().reset_index(name='unique_movies_count')

age_group_movie_counts
```

Out[11]:

	age_group	unique_movies_count
0	Under 18	154
1	19 to 45	241
2	Above 45	226

Objective 5

A function that takes in a user_id and a movie_id, and returns a list of all the other movies that the user rated similarly to the given movie, i.e. with the same rating. Demonstrate that your function works.

In [12]:

```
def find_similarly_rated_movies(user_id, movie_id, data):
    user_rating = data[(data['user_id'] == user_id) & (data['movie_id'] == movie_id)]['rating'].values[0]

    user_data = data[data['user_id'] == user_id]

    similarly_rated_movies = user_data[user_data['rating'] == user_rating]['movie_id'].unique()

    similarly_rated_movies = similarly_rated_movies[similarly_rated_movies != movie_id]

    return similarly_rated_movies


user_id_example = 12
movie_id_example = 1193

similar_movies = find_similarly_rated_movies(user_id_example, movie_id_example, data)
print(f"User {user_id_example} rated movies {similar_movies} similarly to movie {movie_id_example}.")

selected = movies[movies["movie_id"].isin(similar_movies)][['movie_id','title']].value_counts().reset_index()[['movie_id','title']]
selected
```

```
User 12 rated movies [3265 3897 3658 1303  999] similarly to movie 1193.
```

Out[12]:

	movie_id	title
0	999	2 Days in the Valley (1996)
1	1303	Man Who Would Be King, The (1975)
2	3265	Hard-Boiled (Lashou shentan) (1992)
3	3658	Quatermass and the Pit (1967)
4	3897	Almost Famous (2000)

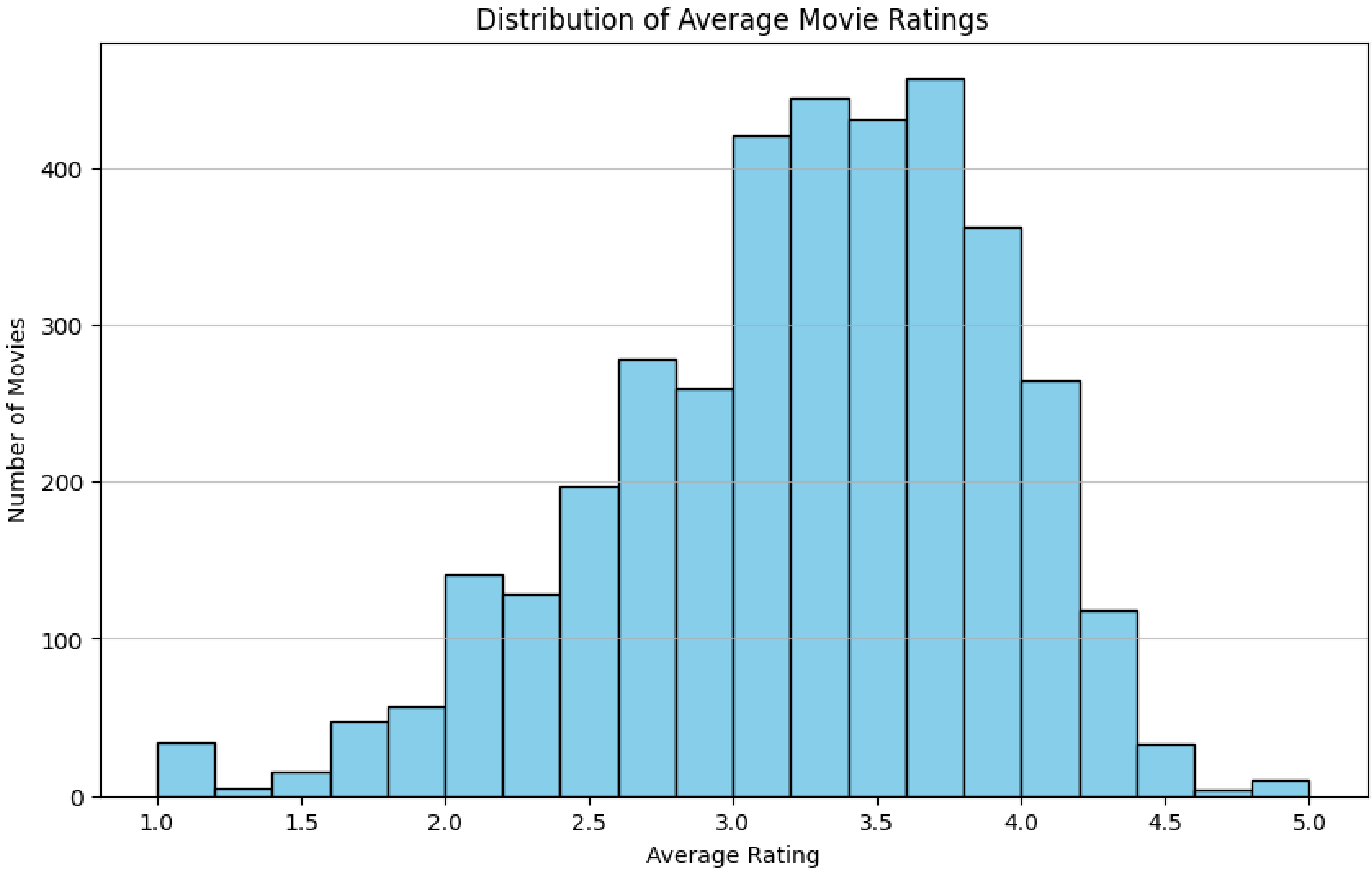
Objective 6

Some other statistic, figure, aggregate, or plot that you created using this dataset, along with a short description of what interesting observations you derived from it.

In [13]:

```
average_ratings = data.groupby('movie_id')['rating'].mean()

plt.figure(figsize=(10, 6))
plt.hist(average_ratings, bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Average Movie Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Number of Movies')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



From the above histogram we can infer that:

1. **A peak at a certain rating range (3.0-3.5). This peak indicates the most common average rating that movies receive.**
2. **The distribution is slightly right-skewed, indicating ratings above 2.5 are more common.**

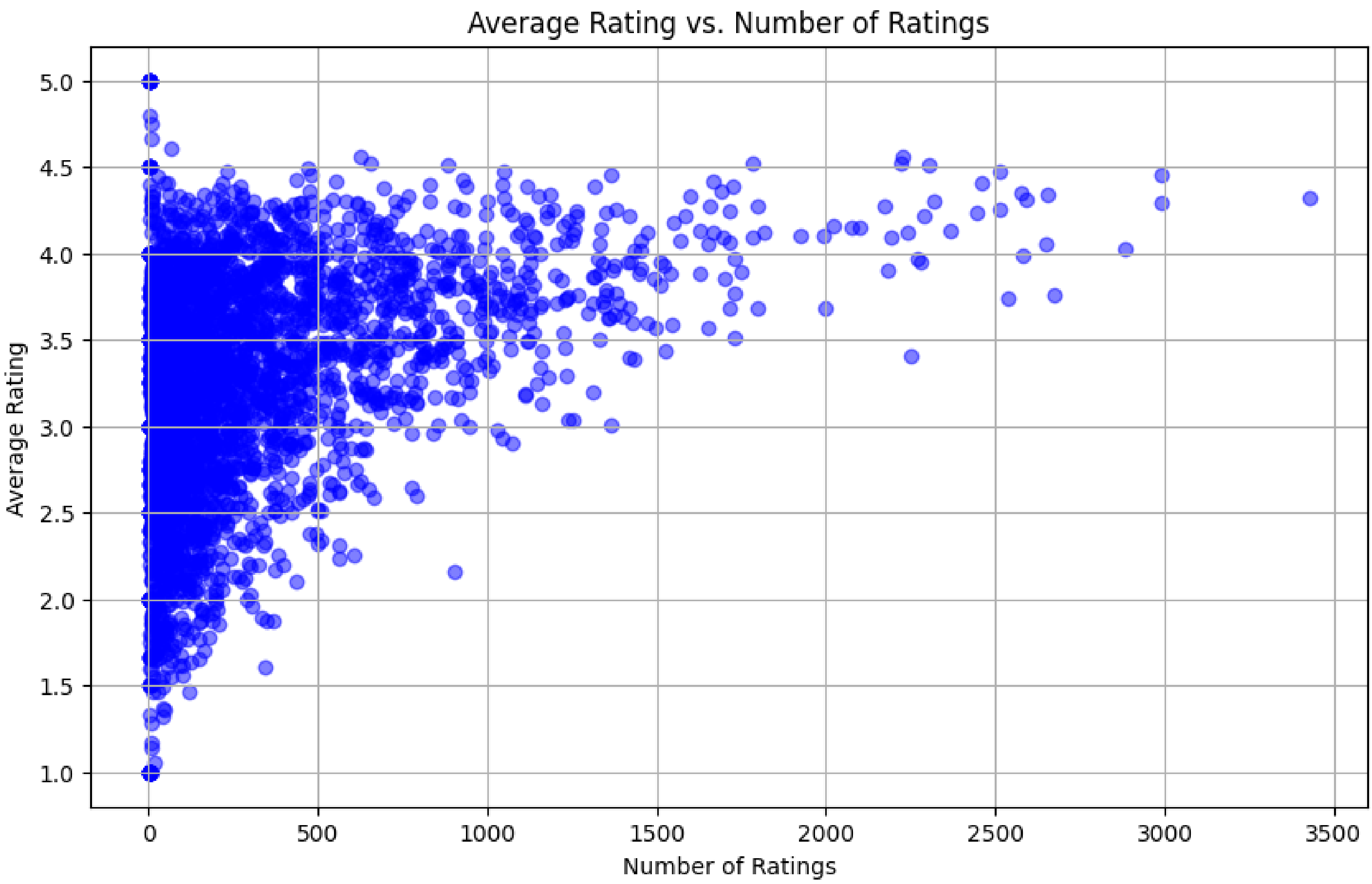
In [14]:

```
average_ratings = data.groupby('movie_id')['rating'].mean()

ratings_count = data.groupby('movie_id')['rating'].count()

movie_stats = pd.DataFrame({
    'average_rating': average_ratings,
    'number_of_ratings': ratings_count
})

plt.figure(figsize=(10, 6))
plt.scatter(movie_stats['number_of_ratings'], movie_stats['average_rating'], alpha=0.5,
color='blue')
plt.title('Average Rating vs. Number of Ratings')
plt.xlabel('Number of Ratings')
plt.ylabel('Average Rating')
plt.grid(True)
plt.show()
```



From the above scatter plot we can infer that:

1. **Areas with more points (3.0 - 4.0) indicate a common number of ratings and average rating scores**
2. **Movies with ratings on extreme end of the spectrum has considerably less number of ratings than those with ratings which are part of the bell curve**