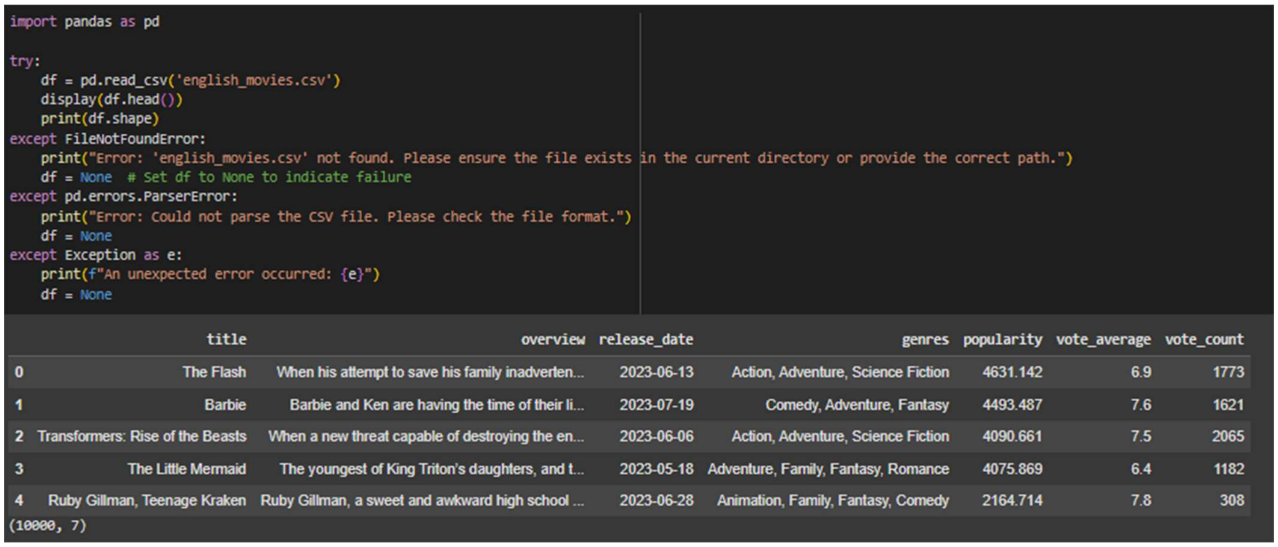
Phase 3-Source code with output

Data loading

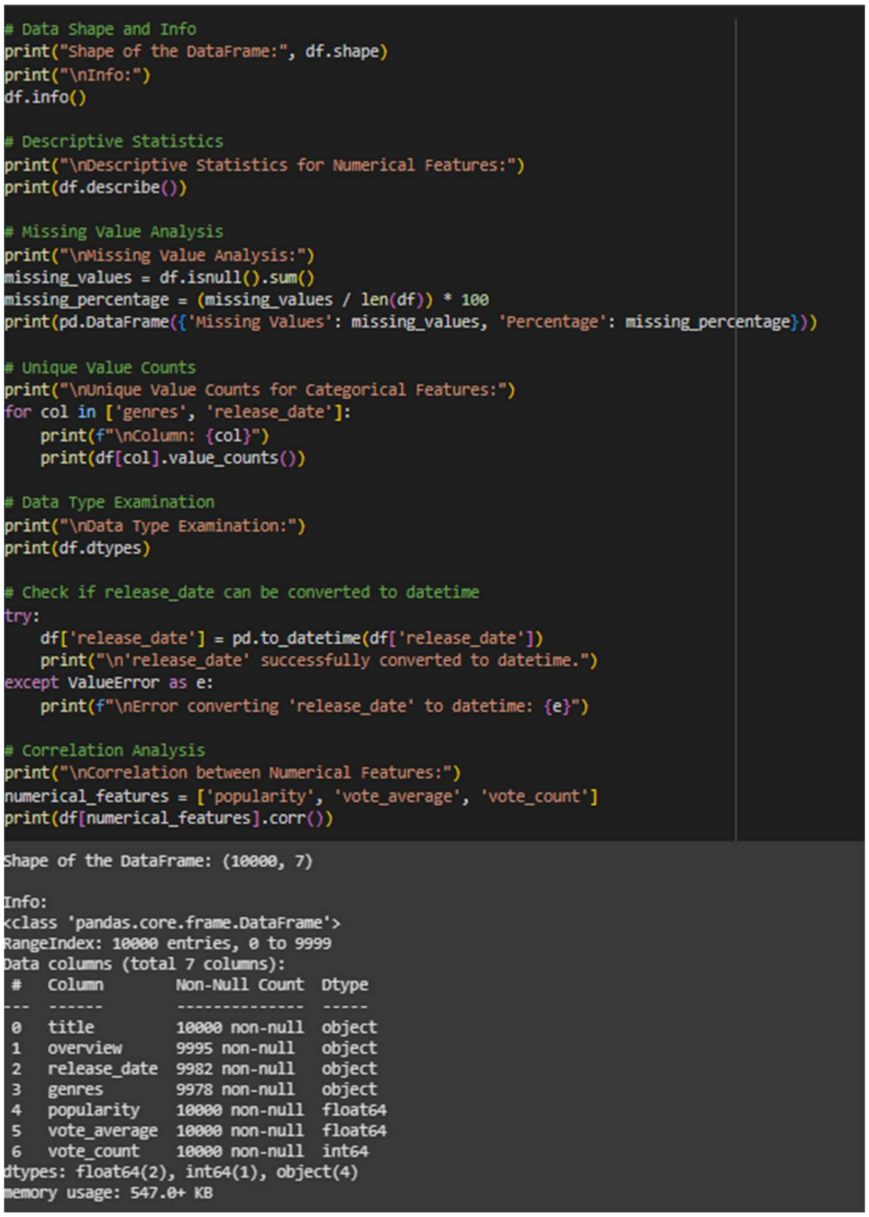


```python
import pandas as pd

try:
    df = pd.read_csv('english_movies.csv')
    display(df.head())
    print(df.shape)
except FileNotFoundError:
    print("Error: 'english_movies.csv' not found. Please ensure the file exists in the current directory or provide the correct path.")
    df = None  # Set df to None to indicate failure
except pd.errors.ParserError:
    print("Error: Could not parse the CSV file. Please check the file format.")
    df = None
except Exception as e:
    print(f"An unexpected error occurred: {e}")
    df = None
```

	title	overview	release_date	genres	popularity	vote_average	vote_count
0	The Flash	When his attempt to save his family inadverten...	2023-06-13	Action, Adventure, Science Fiction	4631.142	6.9	1773
1	Barbie	Barbie and Ken are having the time of their li...	2023-07-19	Comedy, Adventure, Fantasy	4493.487	7.6	1621
2	Transformers: Rise of the Beasts	When a new threat capable of destroying the en...	2023-06-06	Action, Adventure, Science Fiction	4090.661	7.5	2065
3	The Little Mermaid	The youngest of King Triton's daughters, and t...	2023-05-18	Adventure, Family, Fantasy, Romance	4075.869	6.4	1182
4	Ruby Gillman, Teenage Kraken	Ruby Gillman, a sweet and awkward high school ...	2023-06-28	Animation, Family, Fantasy, Comedy	2164.714	7.8	308

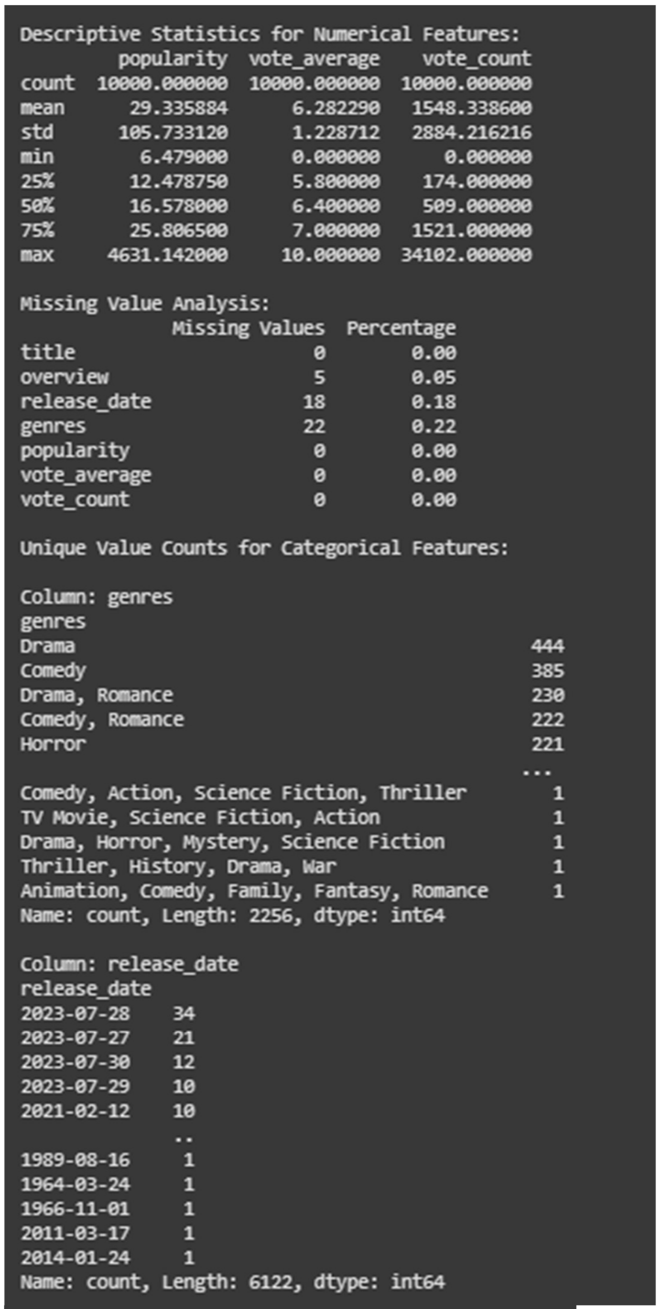
```
(10000, 7)
```

Data Exploration

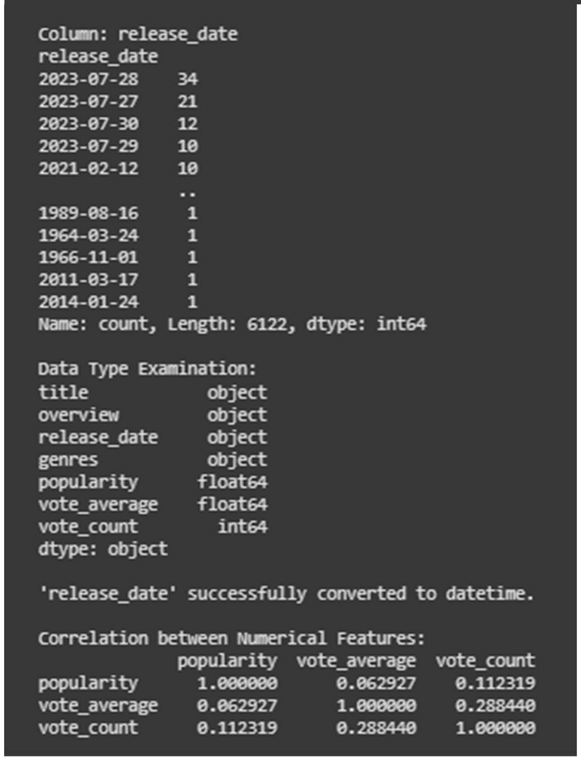


```python
# Data Shape and Info
print("Shape of the DataFrame:", df.shape)
print("\nInfo:")
df.info()

# Descriptive Statistics
print("\nDescriptive Statistics for Numerical Features:")
print(df.describe())

# Missing Value Analysis
print("\nMissing Value Analysis:")
missing_values = df.isnull().sum()
missing_percentage = (missing_values / len(df)) * 100
print(pd.DataFrame({'Missing Values': missing_values, 'Percentage': missing_percentage}))

# Unique Value Counts
print("\nUnique Value Counts for Categorical Features:")
for col in ['genres', 'release_date']:
    print(f"\nColumn: {col}")
    print(df[col].value_counts())

# Data Type Examination
print("\nData Type Examination:")
print(df.dtypes)

# Check if release_date can be converted to datetime
try:
    df['release_date'] = pd.to_datetime(df['release_date'])
    print("\n'release_date' successfully converted to datetime.")
except ValueError as e:
    print(f"\nError converting 'release_date' to datetime: {e}")

# Correlation Analysis
print("\nCorrelation between Numerical Features:")
numerical_features = ['popularity', 'vote_average', 'vote_count']
print(df[numerical_features].corr())
```

```
Shape of the DataFrame: (10000, 7)

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   title         10000 non-null  object
 1   overview      9995 non-null   object
 2   release_date  9982 non-null   object
 3   genres        9978 non-null   object
 4   popularity    10000 non-null  float64
 5   vote_average  10000 non-null  float64
 6   vote_count    10000 non-null  int64
dtypes: float64(2), int64(1), object(4)
memory usage: 547.0+ KB
```



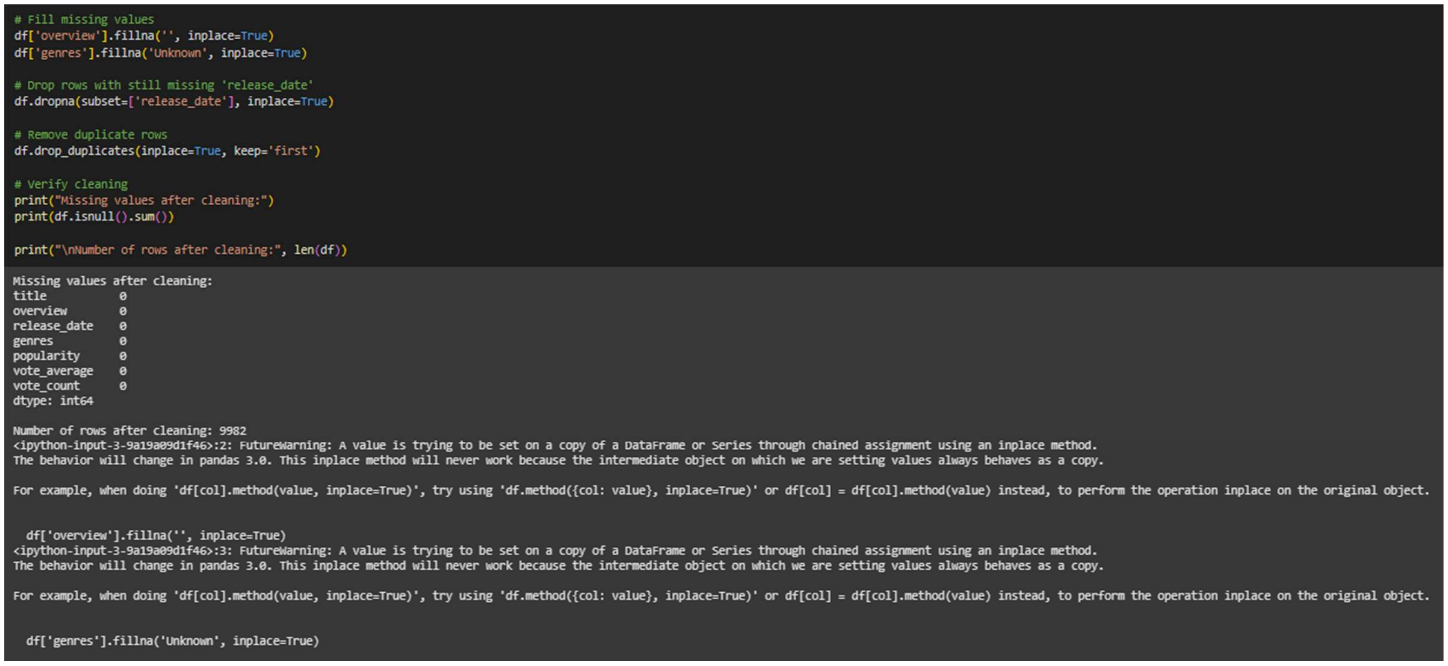
```
Descriptive Statistics for Numerical Features:
         popularity  vote_average    vote_count
count  10000.000000  10000.000000  10000.000000
mean      29.335884      6.282290   1548.338600
std      105.733120      1.228712   2884.216216
min        6.479000      0.000000      0.000000
25%       12.478750      5.800000    174.000000
50%       16.578000      6.400000    509.000000
75%       25.806500      7.000000   1521.000000
max     4631.142000     10.000000  34102.000000

Missing Value Analysis:
              Missing Values  Percentage
title                      0        0.00
overview                   5        0.05
release_date              18        0.18
genres                    22        0.22
popularity                 0        0.00
vote_average               0        0.00
vote_count                 0        0.00

Unique Value Counts for Categorical Features:

Column: genres
genres
Drama                                              444
Comedy                                             385
Drama, Romance                                     230
Comedy, Romance                                    222
Horror                                             221
                                                  ...
Comedy, Action, Science Fiction, Thriller            1
TV Movie, Science Fiction, Action                    1
Drama, Horror, Mystery, Science Fiction              1
Thriller, History, Drama, War                        1
Animation, Comedy, Family, Fantasy, Romance          1
Name: count, Length: 2256, dtype: int64

Column: release_date
release_date
2023-07-28    34
2023-07-27    21
2023-07-30    12
2023-07-29    10
2021-02-12    10
              ..
1989-08-16     1
1964-03-24     1
1966-11-01     1
2011-03-17     1
2014-01-24     1
Name: count, Length: 6122, dtype: int64
```



```
Column: release_date
release_date
2023-07-28    34
2023-07-27    21
2023-07-30    12
2023-07-29    10
2021-02-12    10
              ..
1989-08-16     1
1964-03-24     1
1966-11-01     1
2011-03-17     1
2014-01-24     1
Name: count, Length: 6122, dtype: int64
```

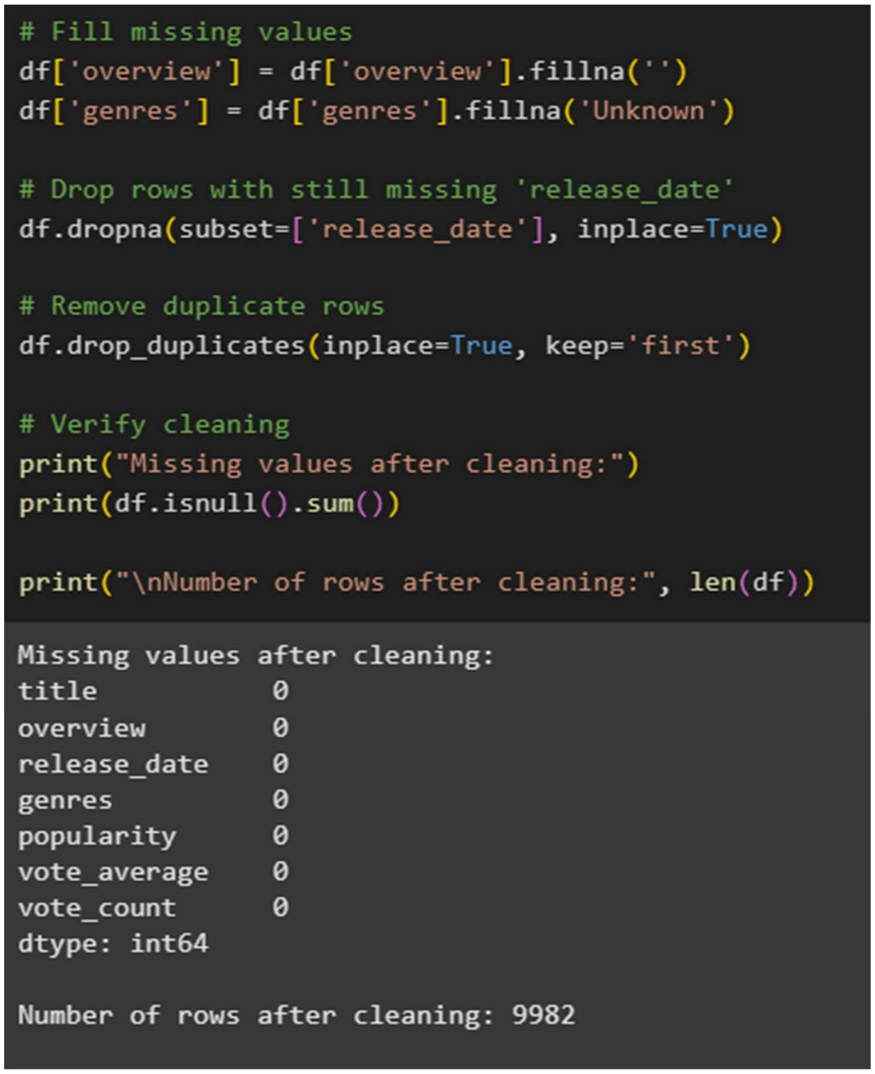
```
Data Type Examination:
title            object
overview         object
release_date     object
genres           object
popularity      float64
vote_average    float64
vote_count        int64
dtype: object

'release_date' successfully converted to datetime.

Correlation between Numerical Features:
              popularity  vote_average  vote_count
popularity      1.000000      0.062927    0.112319
vote_average    0.062927      1.000000    0.288440
vote_count      0.112319      0.288440    1.000000
```

Data Cleaning



```python
# Fill missing values
df['overview'].fillna('', inplace=True)
df['genres'].fillna('Unknown', inplace=True)

# Drop rows with still missing 'release_date'
df.dropna(subset=['release_date'], inplace=True)

# Remove duplicate rows
df.drop_duplicates(inplace=True, keep='first')

# Verify cleaning
print("Missing values after cleaning:")
print(df.isnull().sum())

print("\nNumber of rows after cleaning:", len(df))
```

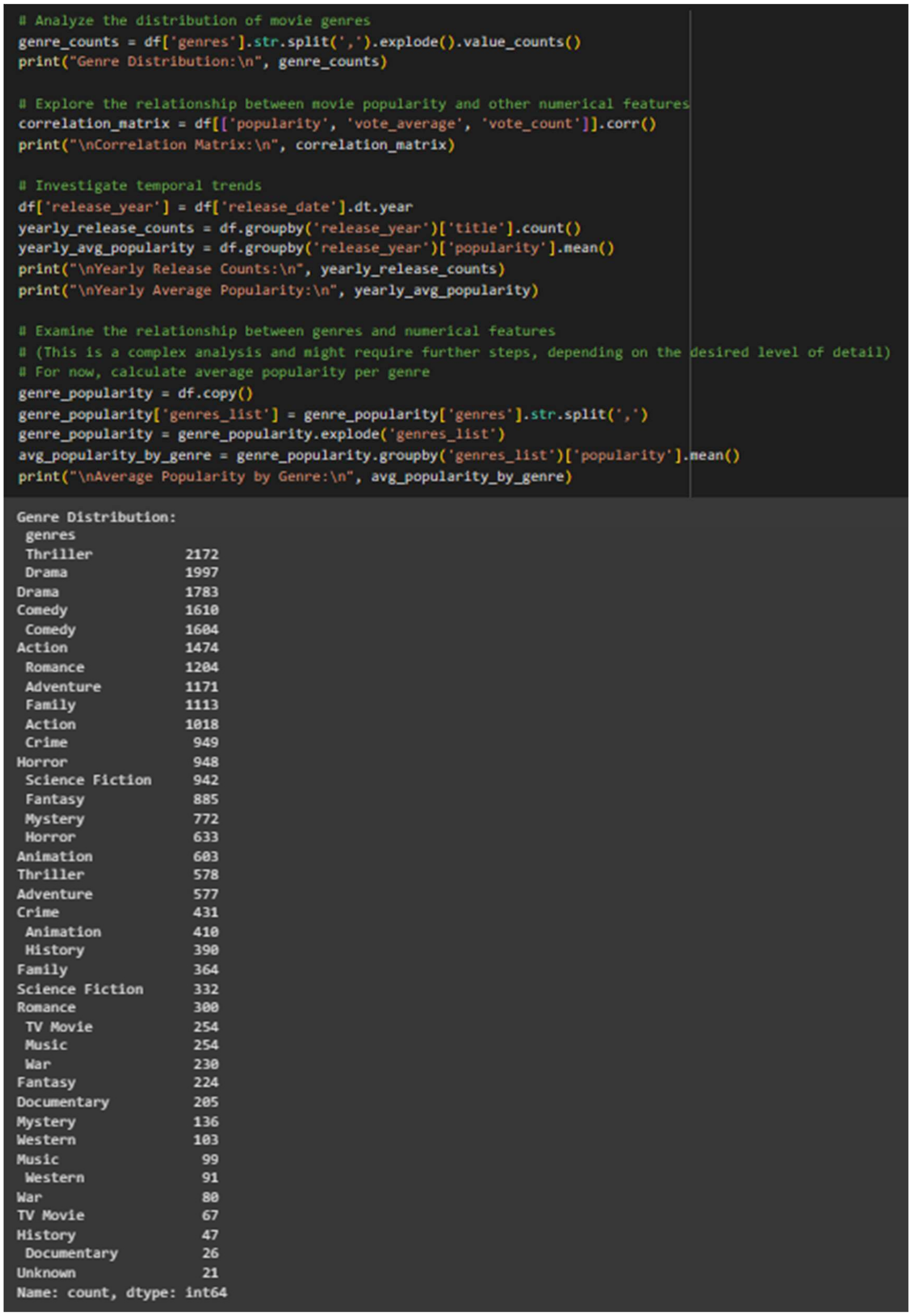
```
Missing values after cleaning:
title           0
overview        0
release_date    0
genres          0
popularity      0
vote_average    0
vote_count      0
dtype: int64

Number of rows after cleaning: 9982
<ipython-input-3-9a19a09d1f46>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['overview'].fillna('', inplace=True)
<ipython-input-3-9a19a09d1f46>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['genres'].fillna('Unknown', inplace=True)
```



```python
# Fill missing values
df['overview'] = df['overview'].fillna('')
df['genres'] = df['genres'].fillna('Unknown')

# Drop rows with still missing 'release_date'
df.dropna(subset=['release_date'], inplace=True)

# Remove duplicate rows
df.drop_duplicates(inplace=True, keep='first')

# Verify cleaning
print("Missing values after cleaning:")
print(df.isnull().sum())

print("\nNumber of rows after cleaning:", len(df))
```

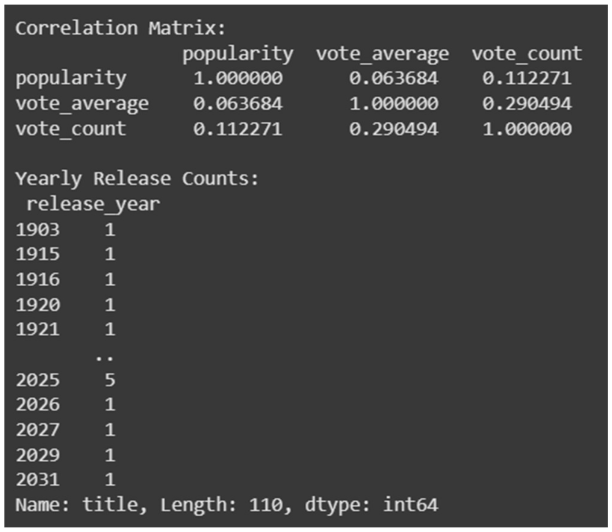
```
Missing values after cleaning:
title           0
overview        0
release_date    0
genres          0
popularity      0
vote_average    0
vote_count      0
dtype: int64

Number of rows after cleaning: 9982
```

Data Analysis:

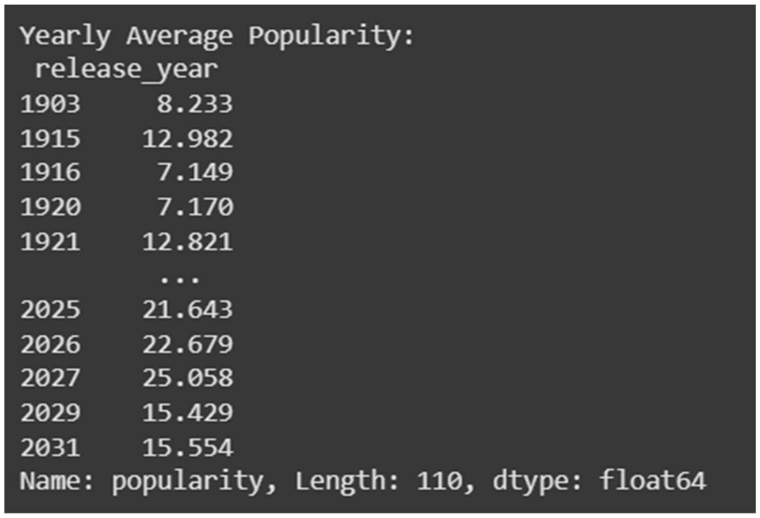


```
# Analyze the distribution of movie genres
genre_counts = df['genres'].str.split(',').explode().value_counts()
print("Genre Distribution:\n", genre_counts)

# Explore the relationship between movie popularity and other numerical features
correlation_matrix = df[['popularity', 'vote_average', 'vote_count']].corr()
print("\nCorrelation Matrix:\n", correlation_matrix)

# Investigate temporal trends
df['release_year'] = df['release_date'].dt.year
yearly_release_counts = df.groupby('release_year')['title'].count()
yearly_avg_popularity = df.groupby('release_year')['popularity'].mean()
print("\nYearly Release Counts:\n", yearly_release_counts)
print("\nYearly Average Popularity:\n", yearly_avg_popularity)

# Examine the relationship between genres and numerical features
# (This is a complex analysis and might require further steps, depending on the desired level of detail)
# For now, calculate average popularity per genre
genre_popularity = df.copy()
genre_popularity['genres_list'] = genre_popularity['genres'].str.split(',')
genre_popularity = genre_popularity.explode('genres_list')
avg_popularity_by_genre = genre_popularity.groupby('genres_list')['popularity'].mean()
print("\nAverage Popularity by Genre:\n", avg_popularity_by_genre)
```

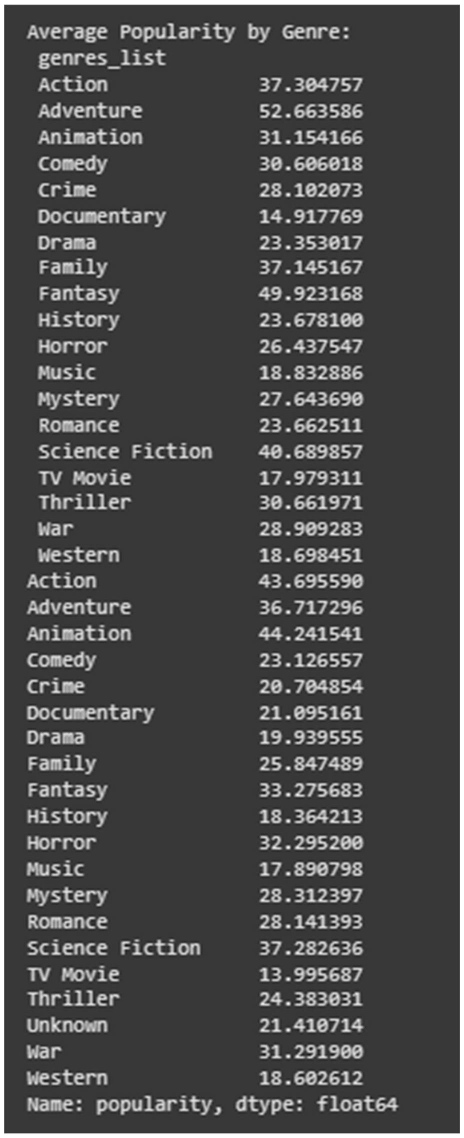
```
Genre Distribution:
 genres
 Thriller           2172
 Drama              1997
Drama               1783
Comedy              1610
 Comedy             1604
Action              1474
 Romance            1204
 Adventure          1171
 Family             1113
 Action             1018
 Crime               949
Horror               948
 Science Fiction     942
 Fantasy             885
 Mystery             772
 Horror              633
Animation            603
Thriller             578
Adventure            577
Crime                431
 Animation           410
 History             390
Family               364
Science Fiction      332
Romance              300
 TV Movie            254
 Music               254
 War                 230
Fantasy              224
Documentary          205
Mystery              136
Western              103
Music                 99
 Western              91
War                   80
TV Movie              67
History               47
 Documentary          26
Unknown               21
Name: count, dtype: int64
```



```
Correlation Matrix:
               popularity  vote_average  vote_count
popularity      1.000000      0.063684    0.112271
vote_average    0.063684      1.000000    0.290494
vote_count      0.112271      0.290494    1.000000

Yearly Release Counts:
 release_year
1903    1
1915    1
1916    1
1920    1
1921    1
       ..
2025    5
2026    1
2027    1
2029    1
2031    1
Name: title, Length: 110, dtype: int64
```

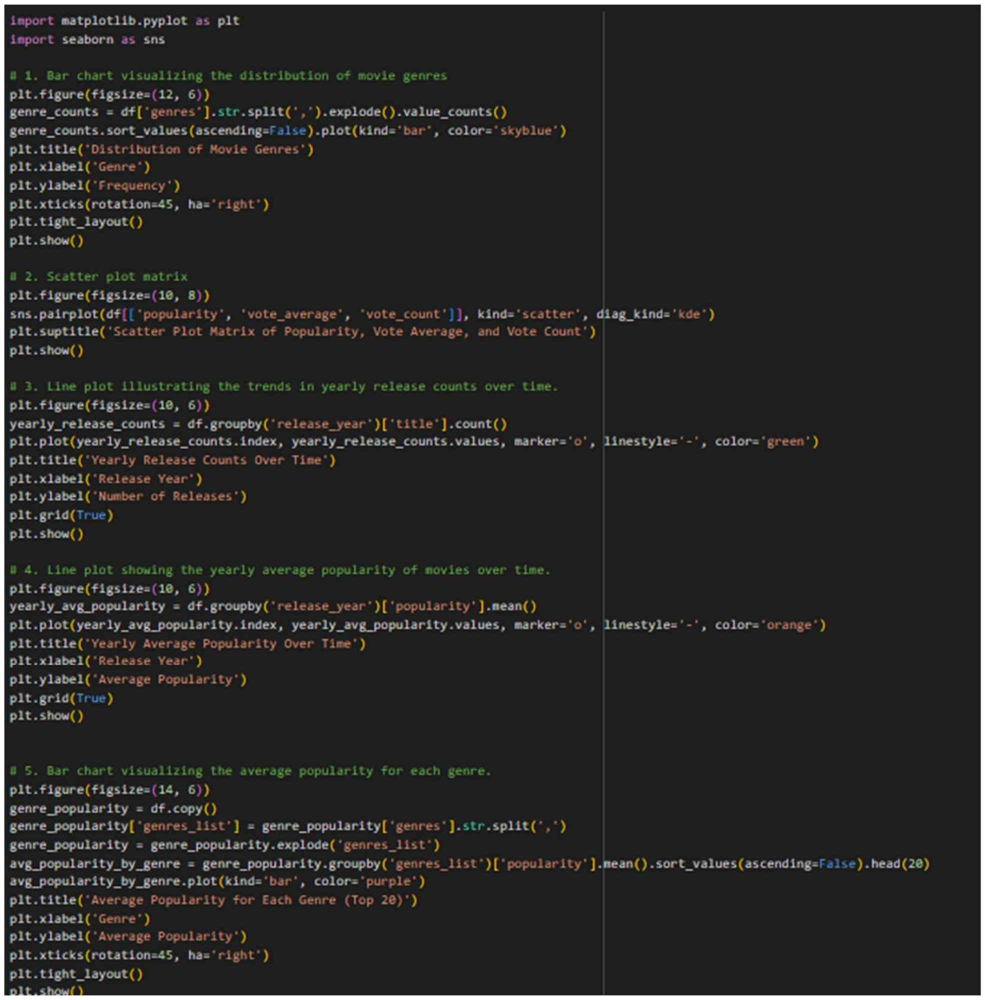


```
Yearly Average Popularity:
 release_year
1903     8.233
1915    12.982
1916     7.149
1920     7.170
1921    12.821
         ...
2025    21.643
2026    22.679
2027    25.058
2029    15.429
2031    15.554
Name: popularity, Length: 110, dtype: float64
```

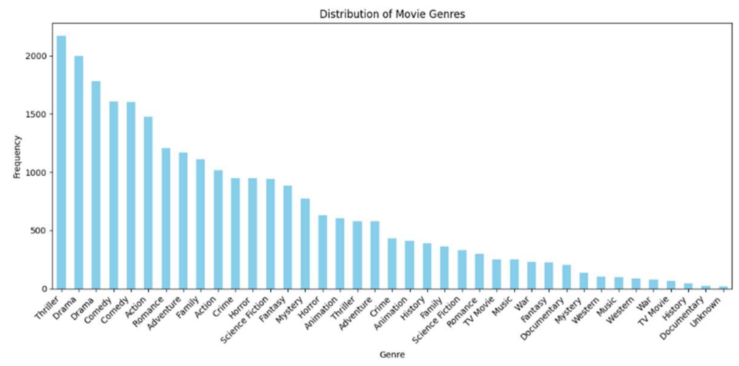


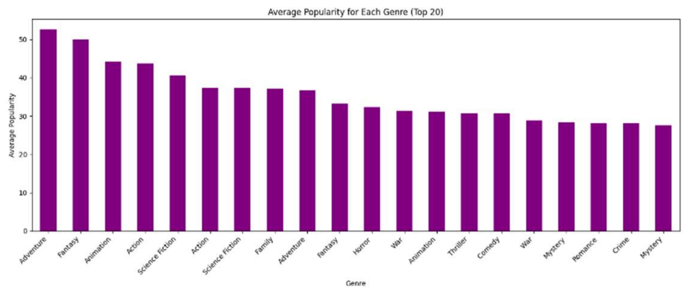
```
Average Popularity by Genre:
 genres_list
 Action              37.304757
 Adventure           52.663586
 Animation           31.154166
 Comedy              30.606018
 Crime               28.102073
 Documentary         14.917769
 Drama               23.353017
 Family              37.145167
 Fantasy             49.923168
 History             23.678100
 Horror              26.437547
 Music               18.832886
 Mystery             27.643690
 Romance             23.662511
 Science Fiction     40.689857
 TV Movie            17.979311
 Thriller            30.661971
 War                 28.909283
 Western             18.698451
Action               43.695590
Adventure            36.717296
Animation            44.241541
Comedy               23.126557
Crime                20.704854
Documentary          21.095161
Drama                19.939555
Family               25.847489
Fantasy              33.275683
History              18.364213
Horror               32.295200
Music                17.890798
Mystery              28.312397
Romance              28.141393
Science Fiction      37.282636
TV Movie             13.995687
Thriller             24.383031
Unknown              21.410714
War                  31.291900
Western              18.602612
Name: popularity, dtype: float64
```

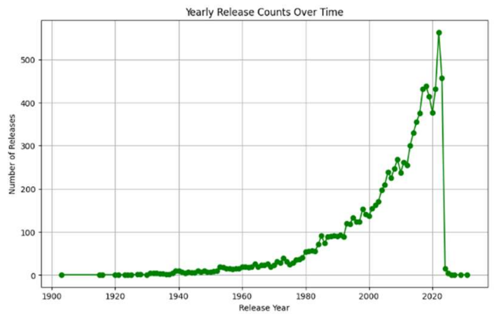
Data Visualization:

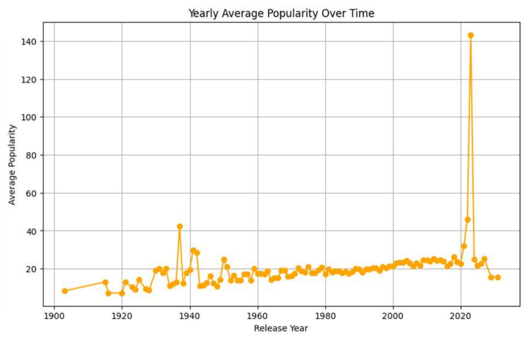


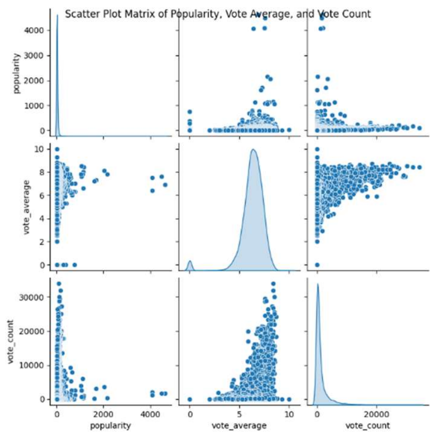
```python
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Bar chart visualizing the distribution of movie genres
plt.figure(figsize=(12, 6))
genre_counts = df['genres'].str.split(',').explode().value_counts()
genre_counts.sort_values(ascending=False).plot(kind='bar', color='skyblue')
plt.title('Distribution of Movie Genres')
plt.xlabel('Genre')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# 2. Scatter plot matrix
plt.figure(figsize=(10, 8))
sns.pairplot(df[['popularity', 'vote_average', 'vote_count']], kind='scatter', diag_kind='kde')
plt.suptitle('Scatter Plot Matrix of Popularity, Vote Average, and Vote Count')
plt.show()

# 3. Line plot illustrating the trends in yearly release counts over time.
plt.figure(figsize=(10, 6))
yearly_release_counts = df.groupby('release_year')['title'].count()
plt.plot(yearly_release_counts.index, yearly_release_counts.values, marker='o', linestyle='-', color='green')
plt.title('Yearly Release Counts Over Time')
plt.xlabel('Release Year')
plt.ylabel('Number of Releases')
plt.grid(True)
plt.show()

# 4. Line plot showing the yearly average popularity of movies over time.
plt.figure(figsize=(10, 6))
yearly_avg_popularity = df.groupby('release_year')['popularity'].mean()
plt.plot(yearly_avg_popularity.index, yearly_avg_popularity.values, marker='o', linestyle='-', color='orange')
plt.title('Yearly Average Popularity Over Time')
plt.xlabel('Release Year')
plt.ylabel('Average Popularity')
plt.grid(True)
plt.show()


# 5. Bar chart visualizing the average popularity for each genre.
plt.figure(figsize=(14, 6))
genre_popularity = df.copy()
genre_popularity['genres_list'] = genre_popularity['genres'].str.split(',')
genre_popularity = genre_popularity.explode('genres_list')
avg_popularity_by_genre = genre_popularity.groupby('genres_list')['popularity'].mean().sort_values(ascending=False).head(20)
avg_popularity_by_genre.plot(kind='bar', color='purple')
plt.title('Average Popularity for Each Genre (Top 20)')
plt.xlabel('Genre')
plt.ylabel('Average Popularity')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Distribution of Movie Genres
Frequency
Thriller
Drama
Drama
Comedy
Comedy
Action
Romance
Adventure
Family
Action
Crime
Horror
Science Fiction
Fantasy
Mystery
Horror
Animation
Thriller
Adventure
Crime
Animation
History
Family
Science Fiction
Romance
TV Movie
Music
War
Fantasy
Documentary
Mystery
Western
Music
Western
War
TV Movie
History
Documentary
Unknown
Genre


Average Popularity for Each Genre (Top 20)
Average Popularity
Adventure
Fantasy
Animation
Action
Science Fiction
Action
Science Fiction
Family
Adventure
Fantasy
Horror
War
Animation
Thriller
Comedy
War
Mystery
Romance
Crime
Mystery
Genre


Yearly Release Counts Over Time
Number of Releases
Release Year


Yearly Average Popularity Over Time
Average Popularity
Release Year


Scatter Plot Matrix of Popularity, Vote Average, and Vote Count
popularity
vote_average
vote_count
popularity
vote_average
vote_count