---------- Netflix Movie Streaming Analysis ----------

Importing Libraries

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Dataset Load

```
In [2]:
df = pd.read_csv("Netflix Movies and TV Shows_clustering.csv", lineterminator = '\n')
```

```
In [3]:
df.head()
```

```
Out[3]:
```

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	Poster_Url
0	2021-12-15	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940	8.3	en	Action, Adventure, Science Fiction	https://image.tmdb.org/t/p/original/1g0dhYtq4...
1	2022-03-01	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	en	Crime, Mystery, Thriller	https://image.tmdb.org/t/p/original/74xTEgt7R3...
2	2022-02-25	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	en	Thriller	https://image.tmdb.org/t/p/original/vDHsLnOWKl...
3	2021-11-24	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076	7.7	en	Animation, Comedy, Family, Fantasy	https://image.tmdb.org/t/p/original/4j0PNHkMr5...
4	2021-12-22	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793	7.0	en	Action, Adventure, Thriller, War	https://image.tmdb.org/t/p/original/aq4Pwv5Xeu...

Exploratory Data Analysis (EDA)

```
In [4]:
# Checking Dataset info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9827 entries, 0 to 9826
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Release_Date     9827 non-null   object
 1   Title            9827 non-null   object
 2   Overview         9827 non-null   object
 3   Popularity       9827 non-null   float64
 4   Vote_Count       9827 non-null   int64
 5   Vote_Average     9827 non-null   float64
 6   Original_Language  9827 non-null   object
 7   Genre            9827 non-null   object
 8   Poster_Url       9827 non-null   object
dtypes: float64(3), int64(1), object(5)
memory usage: 691.1+ KB
```

```
In [5]:
df.isnull().sum()
```

```
Out[5]:
Release_Date       0
Title              0
Popularity         0
Vote_Count         0
Vote_Average       0
Genre              0
dtype: int64
```

```
In [6]:
# Checking duplicate values

df.duplicated().sum()
```

```
Out[6]:
np.int64(0)
```

```
In [7]:
df.head(1)
```

```
Out[7]:
```

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	Poster_Url
0	2021-12-15	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940	8.3	en	Action, Adventure, Science Fiction	https://image.tmdb.org/t/p/original/1g0dhYtq4...

Dropping Overview, Original_Language and Poster-Url

```
In [8]:
# Making list of colums to be dropped
cols = ["Overview", "Original_Language", "Poster_Url"]

# dropping columns and confirming changes
df.drop(cols, axis = 1, inplace = True)
df.head(1)
```

```
Out[8]:
```

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021-12-15	Spider-Man: No Way Home	5083.954	8940	8.3	Action, Adventure, Science Fiction

Relase_Date column convert to date time

```
In [9]:
df["Release_Date"] = pd.to_datetime(df["Release_Date"])

print(df["Release_Date"].dtypes)
```

```
datetime64[ns]
```

```
In [10]:
# cover release date to year

df["Release_Date"] = df["Release_Date"].dt.year

df["Release_Date"].dtypes
```

```
Out[10]:
dtype('int32')
```

Remove white space and split genre

```
In [11]:
# split the strings into lists
df["Genre"] = df["Genre"].str.split(', ')
# explode the lists
df = df.explode("Genre").reset_index(drop=True)
df.head()
```

```
Out[11]:
```

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021	Spider-Man: No Way Home	5083.954	8940	popular	Action
1	2021	Spider-Man: No Way Home	5083.954	8940	popular	Adventure
2	2021	Spider-Man: No Way Home	5083.954	8940	popular	Science Fiction
3	2022	The Batman	3827.658	1151	popular	Crime
4	2022	The Batman	3827.658	1151	popular	Mystery

```
In [12]:
# Casting column into category
df["Genre"] = df["Genre"].astype("category")
df["Genre"].unique()
df["Genre"].dtypes
```

```
Out[12]:
CategoricalDtype(categories=['Action', 'Adventure', 'Animation', 'Comedy', 'Crime',
                  'Documentary', 'Drama', 'Family', 'Fantasy', 'History',
                  'Horror', 'Music', 'Mystery', 'Romance', 'Science Fiction',
                  'TV Movie', 'Thriller', 'War', 'Western'],
, ordered=False, categories_dtype=object)
```

Statical Summery all dataset

```
In [13]:
df.describe()
```

```
Out[13]:
```

	Release_Date	Popularity	Vote_Count	Vote_Average
count	9827.000000	9827.000000	9827.000000	9827.000000
mean	2006.283623	40.326088	1392.805536	6.439534
std	15.685554	108.873998	2611.206907	1.129759
min	1902.000000	13.354000	0.000000	0.000000
25%	2000.000000	16.128500	146.000000	5.900000
50%	2011.000000	21.199000	444.000000	6.500000
75%	2017.000000	35.191500	1376.000000	7.100000
max	2024.000000	5083.954000	31077.000000	10.000000

Create user define function

We will convert the vote_average column into 4 categorise: Popular, Average, Below_Avg, Not_Popular

```
In [14]:
# Create function
def categorize_col(df, col, labels):
    edges = [
        df[col].describe()["min"],
        df[col].describe()["25%"],
        df[col].describe()["50%"],
        df[col].describe()["75%"],
        df[col].describe()["max"]
    ]

    # Use pd.cut to categorize the column based on edges and labels
    df[col] = pd.cut(df[col], edges, labels=labels, duplicates='drop')
    return df

# Define labels for edges
labels = ["not_popular", "below_avg", "average", "popular"]

# Categorize column based on labels and edges
df = categorize_col(df, "Vote_Average", labels)

# Confirming changes
print(df["Vote_Average"].unique())

['popular', 'below_avg', 'average', 'not_popular', NaN]
Categories (4, object): ['not_popular' < 'below_avg' < 'average' < 'popular']
```

```
In [15]:
df["Vote_Average"].value_counts()
```

```
Out[15]:
Vote_Average
not_popular    2487
average        2481
popular        2412
below_avg      2398
Name: count, dtype: int64
```

```
In [16]:
df.nunique()
```

```
Out[16]:
Release_Date      102
Title            9513
Popularity       8160
Vote_Count       3266
Vote_Average        4
Genre              19
dtype: int64
```

Description Analysis

Q1. What is the most frequent genre of movies released on Netflix?

```
In [17]:
df["Genre"].describe()
```

```
Out[17]:
count     25793
unique       19
top       Drama
freq       3715
Name: Genre, dtype: object
```

```
In [18]:
# visualizing genre column
sns.set_style("whitegrid")
plt.figure(figsize=(11, 6))

sns.catplot( y = "Genre", data = df, kind = "count",
            order = df["Genre"].value_counts().index,
            palette = "Paired")

plt.title("Genre Column Distribution")
plt.show()
```

```
C:\Users\smart view\AppData\Local\Temp\ipykernel_20124\3413327206.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.catplot( y = "Genre", data = df, kind = "count",
<Figure size 1250x600 with 0 Axes>
```

Q2. Which has highest votes in vote avg column?

```
In [19]:
# visualizing genre column
sns.set_style("whitegrid")
plt.figure(figsize=(11, 6))

sns.catplot( y = "Vote_Average", data = df, kind = "count",
            order = df["Vote_Average"].value_counts().index,
            palette = "Paired", )

plt.title("Vote Distribution")
plt.show()
```

```
C:\Users\smart view\AppData\Local\Temp\ipykernel_20124\3414332295.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.catplot( y = "Vote_Average", data = df, kind = "count",
<Figure size 1250x600 with 0 Axes>
```

Q3. What movie got the highest popularity? what's its genre?

```
In [20]:
# checking max popularity in dataset
df[df['Popularity'] == df['Popularity'].max()]
```

```
Out[20]:
```

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021	Spider-Man: No Way Home	5083.954	8940	popular	Action
1	2021	Spider-Man: No Way Home	5083.954	8940	popular	Adventure
2	2021	Spider-Man: No Way Home	5083.954	8940	popular	Science Fiction

Q4. What movie got the lowest popularity? what's its genre?

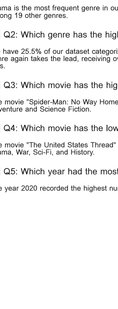
```
In [21]:
# checking min popularity in dataset
df[df['Popularity'] == df['Popularity'].min()]
```

```
Out[21]:
```

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
25787	2021	The United States vs. Billie Holiday	13.354	152	average	Music
25788	2021	The United States vs. Billie Holiday	13.354	152	average	Drama
25789	2021	The United States vs. Billie Holiday	13.354	152	average	History
25790	1984	Threads	13.354	186	popular	War
25791	1984	Threads	13.354	186	popular	Drama
25792	1984	Threads	13.354	186	popular	Science Fiction

Q5. Which year has the most filmmed movies?

```
In [22]:
df["Release_Date"].hist()
plt.title("Release Date Column Distribution")
plt.show()
```



✅ Q1: What is the most frequent genre in the dataset?

Drama is the most frequent genre in our dataset. It appears in over 14% of the total records, standing out among 19 other genres.

✅ Q2: Which genre has the highest votes?

We have 25.5% of our dataset categorized as having a popular vote (6,520 rows). Among these, the Drama genre again takes the lead, receiving over 18.5% of the highest votes, making it the most popular genre among fans.

✅ Q3: Which movie has the highest popularity? What is its genre?

The movie "Spider-Man: No Way Home" holds the highest popularity score in our dataset. Its genres are Adventure and Science Fiction.

✅ Q4: Which movie has the lowest popularity? What is its genre?

The movie "The United States Thread" has the lowest popularity score in the dataset. Its genres are Music, Drama, War, Sci-Fi, and History.

✅ Q5: Which year had the most films produced?

The year 2020 recorded the highest number of films produced in our dataset.