In [1]:
```
# importing lib.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [43]:
```
#line terminator is used her to get the data in a structured format
df = pd.read_csv(r"D:\data analyst\portfolio project\netflix eda\mymoviedb.csv", lineterminator = '\n')
```

In [44]:
```
df.head()
```

Out[44]:

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	
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.o
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
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/
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
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

In [45]:
```
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(2), int64(1), object(6)
memory usage: 691.1+ KB
```

In [46]:
```
df['Genre'].head()
#we may have to remove the spaces after comma
```

Out[46]:
```
0    Action, Adventure, Science Fiction
1              Crime, Mystery, Thriller
2                              Thriller
3    Animation, Comedy, Family, Fantasy
4      Action, Adventure, Thriller, War
Name: Genre, dtype: object
```

In [47]:
```
#checking if there are any duplicate values
df.duplicated().sum()
```

Out[47]:
```
np.int64(0)
```

In [48]:
```
df.describe()
```

Out[48]:

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

- Exploration Summary
- we have a dataframe consisting of 9827 rows and 9 columns
- our dataset doesnt contain NaNs nor duplicated values.
- Release_Date column needs to be casted into date time and to extract only the year value.
- Overview, Original_Languege and Poster-Url wouldn't be so useful during analysis, so we will drop them.
- there is noticable outliers in Popularity column
- Vote_Average bettter be categorised for proper analysis.
- Genre column has comma saperated values and white spaces that needs to be handled and casted into category.Exploartion summary

In [49]:
```
#converted to datetime format
df['Release_Date'] = pd.to_datetime(df['Release_Date'])

print(df['Release_Date'].dtypes)
```
```
datetime64[ns]
```

In [50]:
```
#we have to eliminate everything and just keep the release year
df['Release_Date'] = df['Release_Date'].dt.year

df['Release_Date'].dtypes
```
Out[50]:
```
dtype('int32')
```

In [51]:
```
df.head()
```
Out[51]:

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	
0	2021	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.o
1	2022	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	en	Crime, Mystery, Thriller	https://image.tmdb.org
2	2022	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	en	Thriller	https://image.tmdb.org/
3	2021	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076	7.7	en	Animation, Comedy, Family, Fantasy	https://image.tmdb.org
4	2021	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

Dropping the columns

In [52]:
```
cols = ['Overview', 'Original_Language', 'Poster_Url']
```

In [53]:
```
df.drop(cols, axis = 1, inplace = True)
df.columns
```

Out[53]:
```
Index(['Release_Date', 'Title', 'Popularity', 'Vote_Count', 'Vote_Average',
       'Genre'],
      dtype='object')
```

In [54]:
```
df.head()
```

Out[54]:

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021	Spider-Man: No Way Home	5083.954	8940	8.3	Action, Adventure, Science Fiction
1	2022	The Batman	3827.658	1151	8.1	Crime, Mystery, Thriller
2	2022	No Exit	2618.087	122	6.3	Thriller
3	2021	Encanto	2402.201	5076	7.7	Animation, Comedy, Family, Fantasy
4	2021	The King's Man	1895.511	1793	7.0	Action, Adventure, Thriller, War

categorizing Vote_Average column We would cut the Vote_Average values and make 4 categories: popular average below_avg not_popular to describe it more using catigorize_col() function provided above.

In [55]:
```
#creating an user defined function to convert numerical value to lables
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']]
    df[col] = pd.cut(df[col], edges, labels = labels, duplicates = 'drop')
    return df
```

In [56]:
```
# define labels for edges
labels = ['not_popular', 'below_avg', 'average', 'popular']
 # categorize column based on labels and edges
categorize_col(df, 'Vote_Average', labels)
 # confirming changes
df['Vote_Average'].unique()
```

Out[56]:
```
['popular', 'below_avg', 'average', 'not_popular', NaN]
Categories (4, object): ['not_popular' < 'below_avg' < 'average' < 'popular']
```

In [57]:
```
df.head()
```

Out[57]:

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021	Spider-Man: No Way Home	5083.954	8940	popular	Action, Adventure, Science Fiction
1	2022	The Batman	3827.658	1151	popular	Crime, Mystery, Thriller
2	2022	No Exit	2618.087	122	below_avg	Thriller
3	2021	Encanto	2402.201	5076	popular	Animation, Comedy, Family, Fantasy
4	2021	The King's Man	1895.511	1793	average	Action, Adventure, Thriller, War

In [58]:
```
df.Vote_Average.value_counts()
#you can write it as well df['Vote_Average'].value_counts()
```

Out[58]:
```
Vote_Average
not_popular    2467
popular        2450
average        2412
below_avg      2398
Name: count, dtype: int64
```

In [59]:
```
df.dropna(inplace = True)

df.isna().sum()
```

Out[59]:
```
Release_Date    0
Title           0
Popularity      0
Vote_Count      0
Vote_Average    0
Genre           0
dtype: int64
```

we'd split genres into a list and then explore our dataframe to have only one genre per row for each movie

In [60]:
```
df.head()
```

Out[60]:

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
0	2021	Spider-Man: No Way Home	5083.954	8940	popular	Action, Adventure, Science Fiction
1	2022	The Batman	3827.658	1151	popular	Crime, Mystery, Thriller
2	2022	No Exit	2618.087	122	below_avg	Thriller
3	2021	Encanto	2402.201	5076	popular	Animation, Comedy, Family, Fantasy
4	2021	The King's Man	1895.511	1793	average	Action, Adventure, Thriller, War

In [63]:
```
# split the strings into lists
df['Genre'] = df['Genre'].astype(str).str.split(', ')

# explode the lists
df = df.explode('Genre').reset_index(drop=True)

#Replace 'nan' strings back to actual NaN
df['Genre'] = df['Genre'].replace('nan', pd.NA)
df.head()
```

Out[63]:

	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 [65]:
```
#casting column into category

df['Genre'] = df['Genre'].astype('category')

df['Genre'].dtypes
```

Out[65]:
```
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)
```

In [66]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25552 entries, 0 to 25551
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Release_Date  25552 non-null  int32
 1   Title         25552 non-null  object
 2   Popularity    25552 non-null  float64
 3   Vote_Count    25552 non-null  int64
 4   Vote_Average  25552 non-null  category
 5   Genre         25552 non-null  category
dtypes: category(2), float64(1), int32(1), int64(1), object(1)
memory usage: 749.6+ KB
```

In [67]:
```
df.nunique()
```

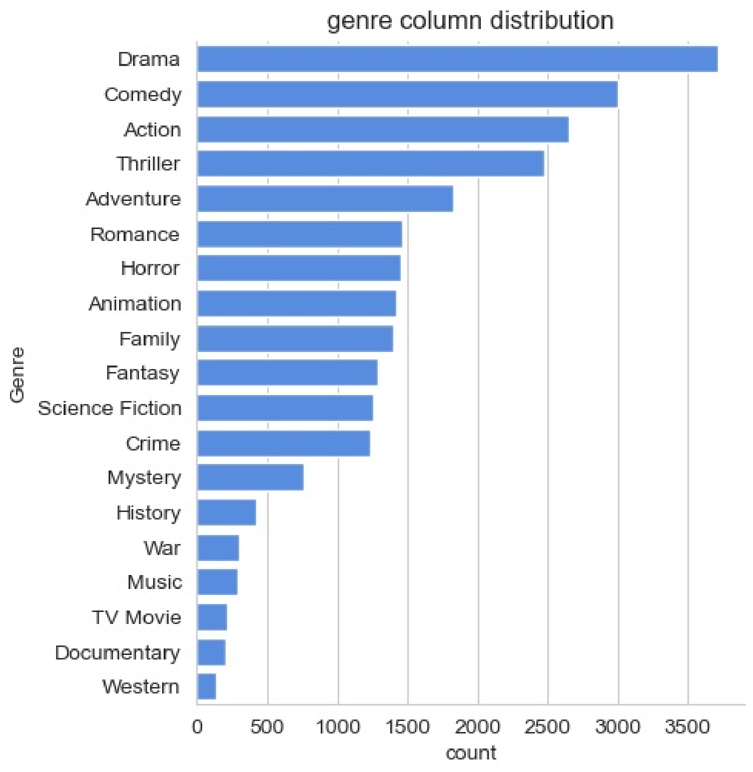
Out[67]:
```
Release_Date     100
Title           9415
Popularity      8088
Vote_Count      3265
Vote_Average       4
Genre             19
dtype: int64
```

Data Visualization

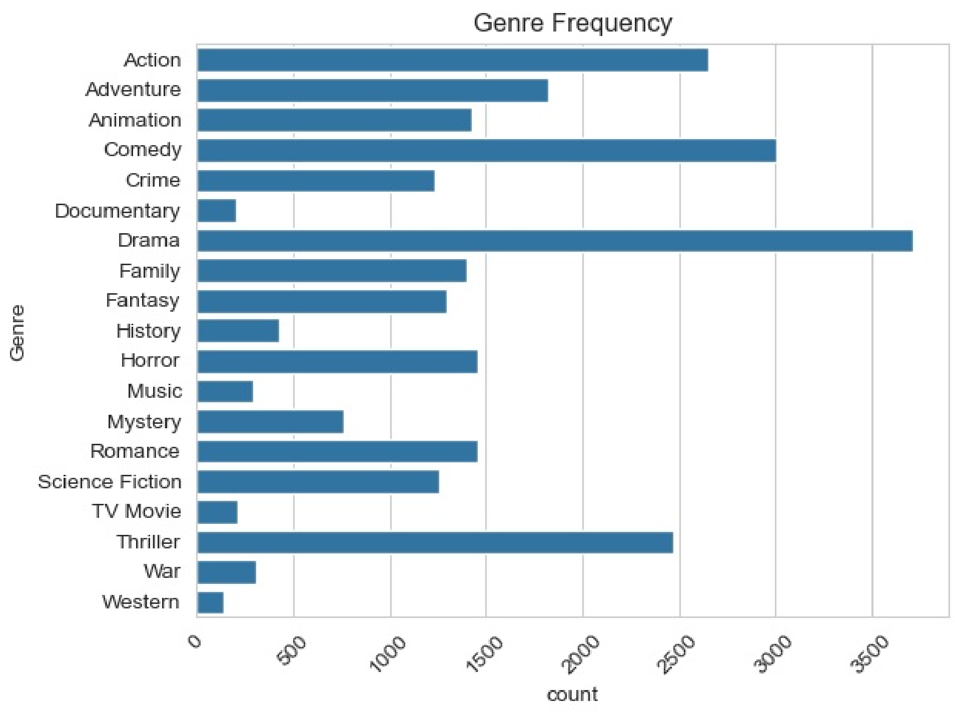
In [69]:
```
# setting up seaborn configurations
sns.set_style('whitegrid')
```

the most frequent genre in the dataset

In [70]:
```
# visualizing genre column
sns.catplot(y = 'Genre', data = df, kind = 'count',
order = df['Genre'].value_counts().index,
color = '#4287f5')
plt.title('genre column distribution')
plt.show()
```

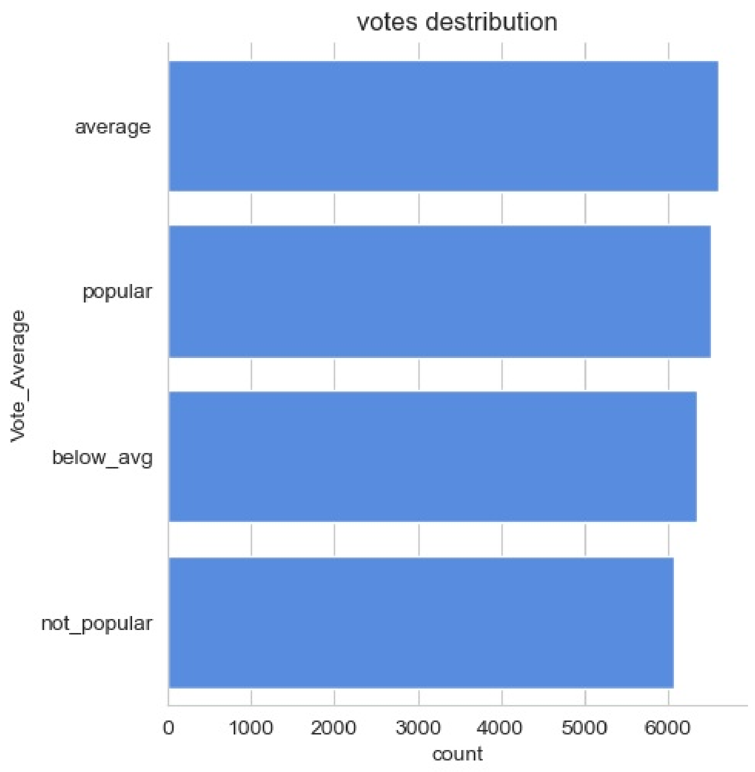


In [74]:
```
# code generated using ai
genre_counts = df['Genre'].value_counts().reset_index().sort_values(by='Genre', ascending=False)
sns.barplot(x='count', y='Genre', data=genre_counts)
plt.xticks(rotation=45)
plt.title('Genre Frequency')
plt.tight_layout()
plt.show()
```



What genres has highest votes ?

In [85]:
```
sns.catplot(y = 'Vote_Average', data = df, kind = 'count',
order = df['Vote_Average'].value_counts().index,
color = '#4287f5')
plt.title('votes destribution')
plt.show()
```



What movie got the highest genre ?

```
In [89]: df[df['Popularity'] == df['Popularity'].max()]
```

Out[89]:

	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

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

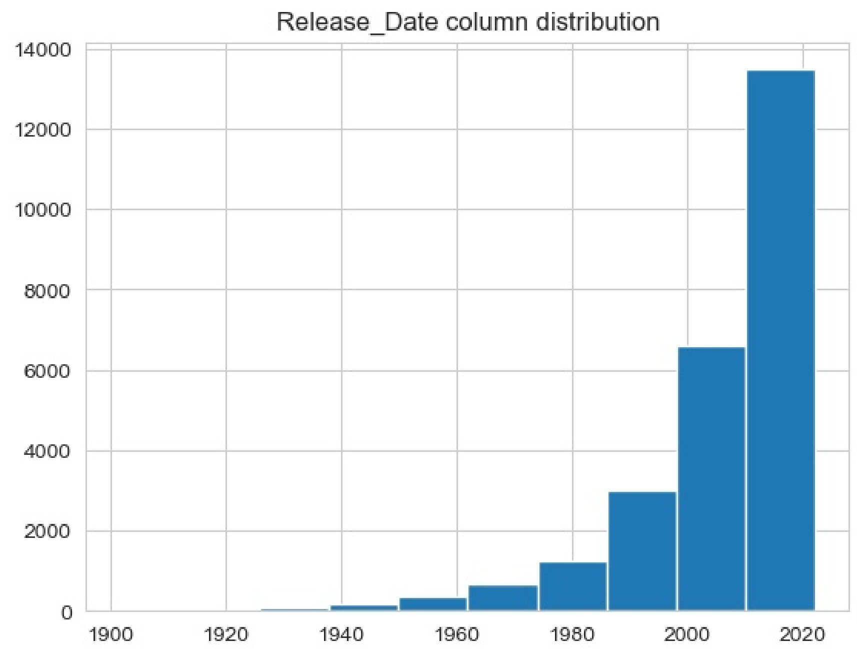
```
In [90]: df[df['Popularity'] == df['Popularity'].min()]
```

Out[90]:

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
25546	2021	The United States vs. Billie Holiday	13.354	152	average	Music
25547	2021	The United States vs. Billie Holiday	13.354	152	average	Drama
25548	2021	The United States vs. Billie Holiday	13.354	152	average	History
25549	1984	Threads	13.354	186	popular	War
25550	1984	Threads	13.354	186	popular	Drama
25551	1984	Threads	13.354	186	popular	Science Fiction

Which year has the most filmmed movies?

```
In [94]: df['Release_Date'].hist()
         plt.title('Release_Date column distribution')
         plt.show()
```


Release_Date column distribution
14000
12000
10000
8000
6000
4000
2000
0
1900
1920
1940
1960
1980
2000
2020

```
In [ ]:
```