In [100…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [101…
```
#It is important here to use engine = python because file had long/messy lines t
df = pd.read_csv("mymoviedb.csv", engine="python")
df.head()
```

Out[101…

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_
0	2021-12-15	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940	8.3	
1	2022-03-01	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	
2	2022-02-25	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	
3	2021-11-24	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076	7.7	
4	2021-12-22	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793	7.0	

In [102…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9837 entries, 0 to 9836
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Release_Date       9837 non-null   object 
 1   Title              9828 non-null   object 
 2   Overview           9828 non-null   object 
 3   Popularity         9827 non-null   float64
 4   Vote_Count         9827 non-null   object 
 5   Vote_Average       9827 non-null   object 
 6   Original_Language  9827 non-null   object 
 7   Genre              9826 non-null   object 
 8   Poster_Url         9826 non-null   object 
dtypes: float64(1), object(8)
memory usage: 691.8+ KB
```

In [103…
```
df['Genre'].unique()
```

Out[103…
```
array(['Action, Adventure, Science Fiction', 'Crime, Mystery, Thriller',
       'Thriller', ..., 'Comedy, TV Movie, Romance',
       'Science Fiction, Fantasy, Family, Music',
       'War, Drama, Science Fiction'], dtype=object)
```

In [104…
```
#this is how you check for duplicate rows
df.duplicated().sum()
```

Out[104…
```
0
```

Data Cleaning

Right now, Release_Date is just text. If you try to sort it, pandas will treat "2025-09-24" like a string, not a real date.

In [107…
```
#At row 1106 in your Release_Date column, the value is " - Just Desserts", which
#Because of that, pd.to_datetime() fails when trying to parse everything as a da
df['Release_Date'] = pd.to_datetime(df['Release_Date'], errors="coerce")
```

In [108…
```
#Now OrderDate is a datetime64[ns] type. If some rows had invalid dates they bec
#NaN is a float in pandas, so the whole column becomes float64.So we fill nan as
df['Release_Date'] = df['Release_Date'].dt.year.fillna(0).astype(int)
df['Release_Date'].dtypes
```

Out[108…
```
dtype('int32')
```

In [109…
```
df.head()
```

Out[109…

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_
0	2021	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940	8.3	
1	2022	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	
2	2022	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	
3	2021	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076	7.7	
4	2021	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793	7.0	

Now we drop unnecessary columns

In [111…
```
not_needed_cols=['Overview','Original_Language','Poster_Url']
```

In [112…
```
new_df=df.drop(not_needed_cols,axis=1)
new_df.columns
```

Out[112…
```
Index(['Release_Date', 'Title', 'Popularity', 'Vote_Count', 'Vote_Average',
       'Genre'],
      dtype='object')
```

In [113…
```
new_df.head()
```

Out[113…

	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

In [117…
```
# Convert Vote_Average column to numeric (force errors to NaN if non-numeric exi
new_df['Vote_Average'] = pd.to_numeric(new_df['Vote_Average'], errors='coerce')
```

In [119…
```
#We have vote_average column which has float points but we have to categorize th
#1-Flop,2-Below Avg,3-Avg,4-Popular
def categorize_col(new_df, col, labels):
    edges = [
        new_df[col].min(),
        new_df[col].quantile(0.25),
        new_df[col].quantile(0.50),
        new_df[col].quantile(0.75),
        new_df[col].max()
    ]
    new_df[col] = pd.cut(new_df[col], edges, labels=labels, duplicates='drop')
    return new_df
```

In [121…
```
# define labels for edges
labels = ['flop','below_avg', 'average', 'popular']
 # categorize column based on labels and edges
categorize_col(new_df, 'Vote_Average', labels)
 # confirming changes
new_df['Vote_Average'].unique()
```

Out[121…
```
['popular', 'below_avg', 'average', 'flop', NaN]
Categories (4, object): ['flop' < 'below_avg' < 'average' < 'popular']
```

In [123…
```
new_df.isna().sum()
```

Out[123…
```
Release_Date      0
Title             9
Popularity       10
Vote_Count       10
Vote_Average    111
Genre            11
dtype: int64
```

In [125…
```
new_df.dropna(inplace = True)
```

In [127…
```
new_df.isna().sum()
```

Out[127…
```
Release_Date    0
Title           0
Popularity      0
Vote_Count      0
Vote_Average    0
Genre           0
dtype: int64
```

In [131…
```
#the Genre column contains values like:"Action, Adventure, Science Fiction"
# split the strings into lists
new_df['Genre'] = new_df['Genre'].str.split(', ')
#this turns it into a Python list:["Action", "Adventure", "Science Fiction"]
# explode the lists
new_df = new_df.explode('Genre').reset_index(drop=True)
#.explode('Genre') splits each list element into a separate row.
#So instead of keeping all genres in one row, it duplicates the other column val
```

In [133…
```
new_df.head()
```

Out[133…

	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 [135…
```
 # casting column into category
new_df['Genre'] = new_df['Genre'].astype('category')
# confirming changes
new_df['Genre'].dtypes
```

Out[135…
```
CategoricalDtype(categories=['Action', 'Adventure', 'Animation', 'Comedy', 'Cri
me',
                  'Documentary', 'Drama', 'Family', 'Fantasy', 'History',
                  'Horror', 'Music', 'Mystery', 'Romance', 'Science Fiction',
                  'TV Movie', 'Thriller', 'War', 'Western'],
, ordered=False, categories_dtype=object)
```

In [137…
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25551 entries, 0 to 25550
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype   
---  ------        --------------  -----   
 0   Release_Date  25551 non-null  int32   
 1   Title         25551 non-null  object  
 2   Popularity    25551 non-null  float64 
 3   Vote_Count    25551 non-null  object  
 4   Vote_Average  25551 non-null  category
 5   Genre         25551 non-null  category
dtypes: category(2), float64(1), int32(1), object(2)
memory usage: 749.6+ KB
```

In [147…
```
new_df.nunique()
```

Out[147…
```
Release_Date     100
Title           9414
Popularity      8087
Vote_Count      3265
Vote_Average       4
Genre             19
dtype: int64
```

Data Visualisation

In [152…
```
# setting up seaborn configurations
sns.set_style('darkgrid')
```

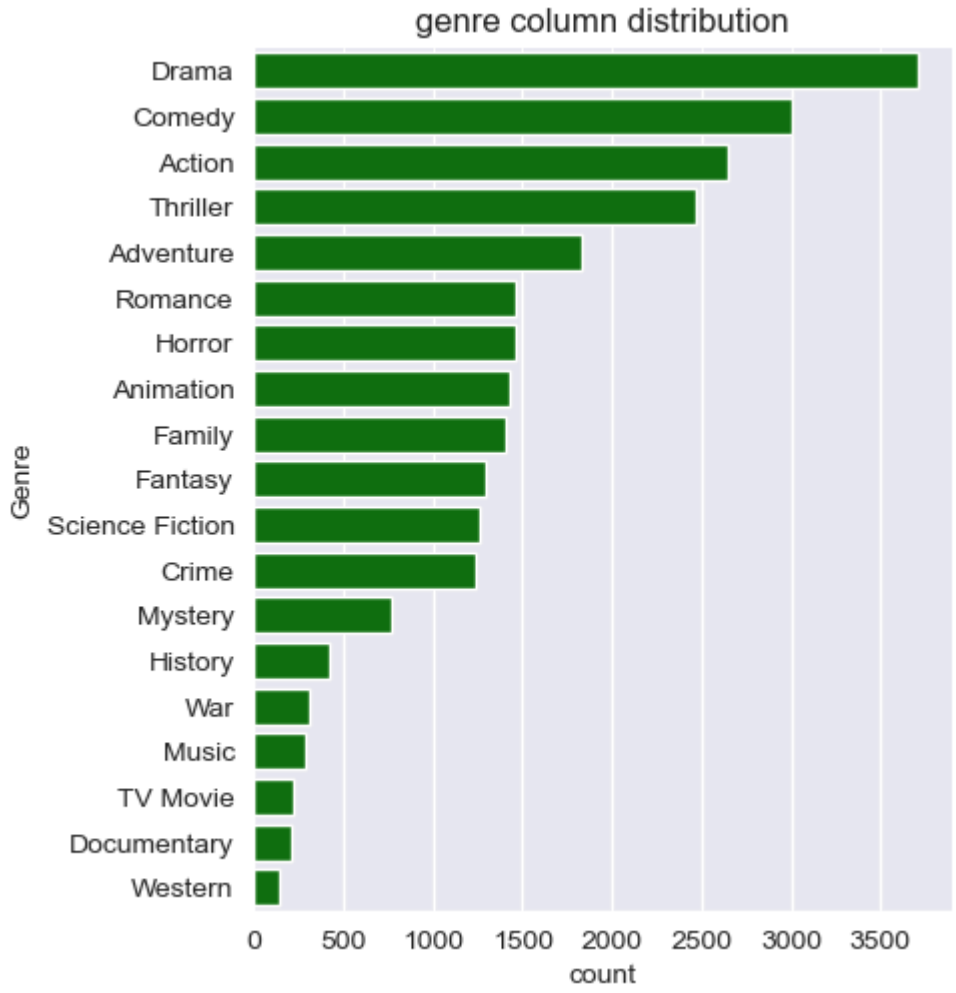
In [154…
```
new_df['Genre'].describe()
```

Out[154…
```
count     25551
unique       19
top       Drama
freq       3715
Name: Genre, dtype: object
```

What is the most frequent genre in the dataset

In [162…
```
# visualizing genre column
sns.catplot(y = 'Genre', data =new_df, kind = 'count',
order = new_df['Genre'].value_counts().index,color = 'green')
plt.title('genre column distribution')
plt.show()
```

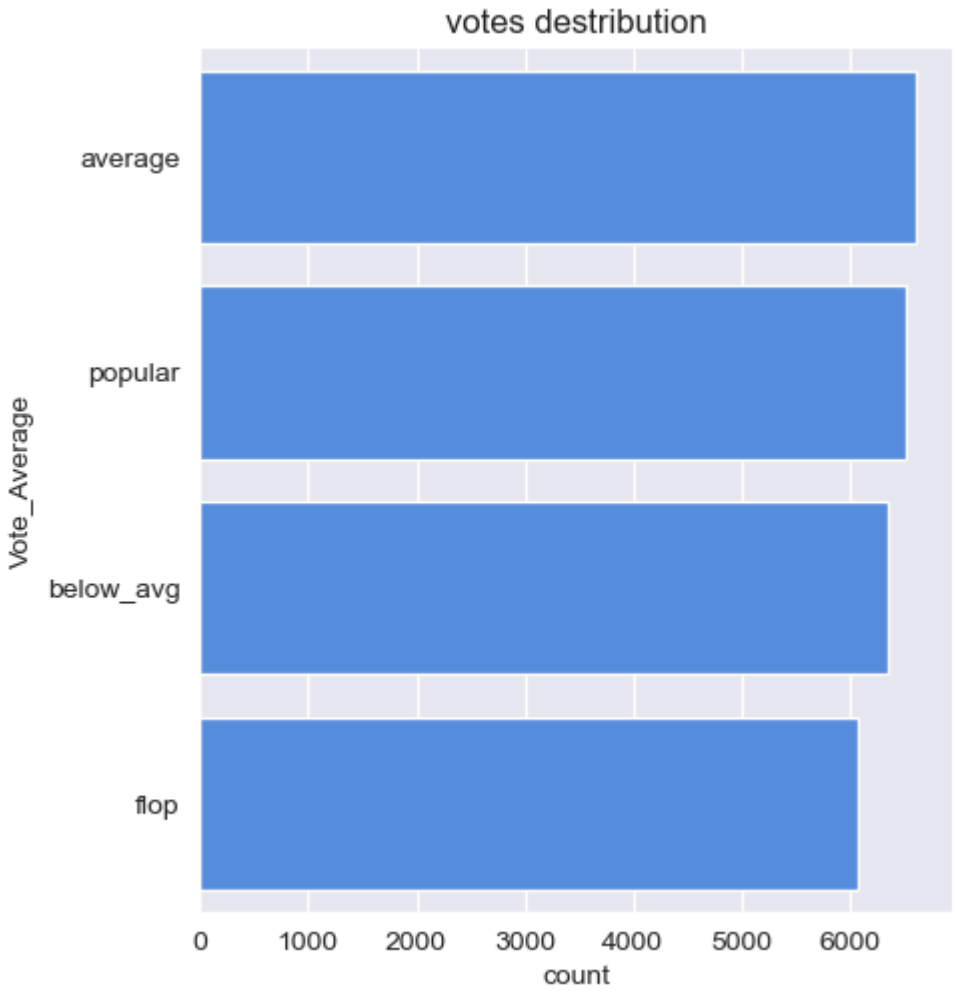
```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarni
ng: The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or obser
ved=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
C:\Users\user\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarni
ng: The default 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 obser
ved=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



In [171…

```
 # visualizing vote_average column
sns.catplot(y = 'Vote_Average', data =new_df, kind = 'count', order = new_df['Vc
color = '#4287f5')
plt.title('votes destribution')
plt.show()
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarni
ng: The default 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 obser
ved=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
C:\Users\user\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarni
ng: The default 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 obser
ved=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



Most Popular movie

In [173…
```
# checking max popularity in dataset
new_df[new_df['Popularity'] == new_df['Popularity'].max()]
```

Out[173…

	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

Least Popular movie

In [183…
```
# checking min popularity in dataset
new_df[new_df['Popularity'] == new_df['Popularity'].min()]
```

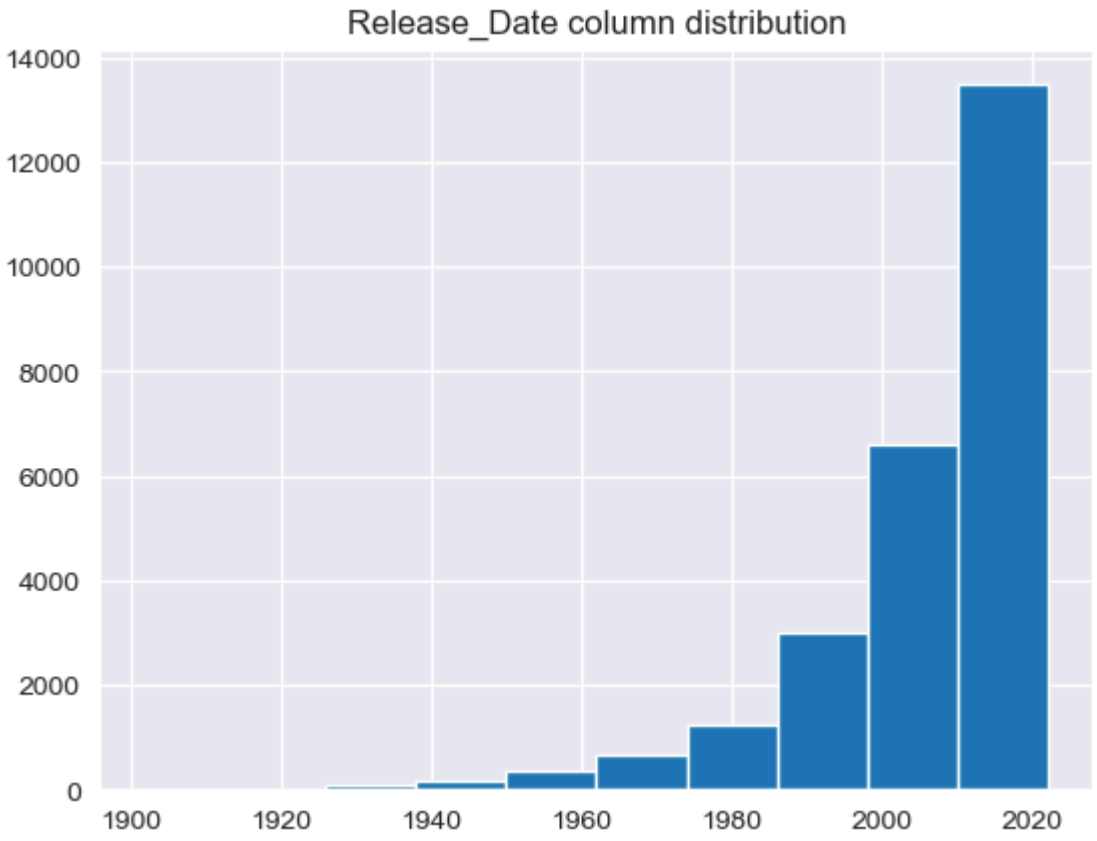
Out[183…

	Release_Date	Title	Popularity	Vote_Count	Vote_Average	Genre
25545	2021	The United States vs. Billie Holiday	13.354	152	average	Music
25546	2021	The United States vs. Billie Holiday	13.354	152	average	Drama
25547	2021	The United States vs. Billie Holiday	13.354	152	average	History
25548	1984	Threads	13.354	186	popular	War
25549	1984	Threads	13.354	186	popular	Drama
25550	1984	Threads	13.354	186	popular	Science Fiction

Which year has the most filmmed movies

In [186…

```
new_df['Release_Date'].hist()
plt.title('Release_Date column distribution')
plt.show()
```



In [ ]: