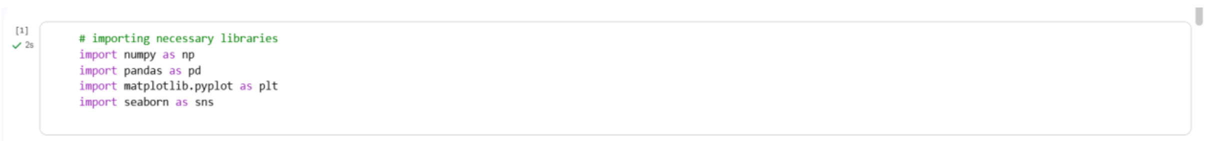
Netflix movie data analysis project code explanation and results

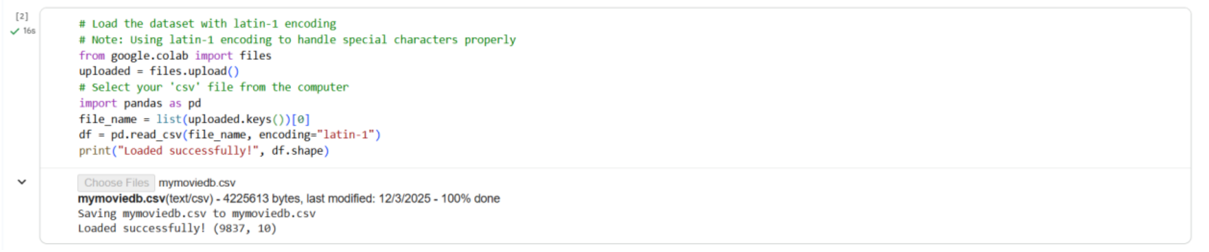
Problem Statement:

Netflix is known for its work in data science, AI, and ML, particularly for building strong recommendation models and algorithms that understand customer behavior and patterns. Suppose you are working in a data-driven job role, and you have a dataset of more than 9,000 movies. You need to solve the following questions to help the company make informed business decisions accordingly.

1) What is the most frequent genre of movies released on Netflix?
2) Which has highest votes in vote avg column?
3) What movie got the highest popularity? what's its genre?
4) What movie got the lowest popularity? what's its genre?
5) Which year has the most filmmed movies?

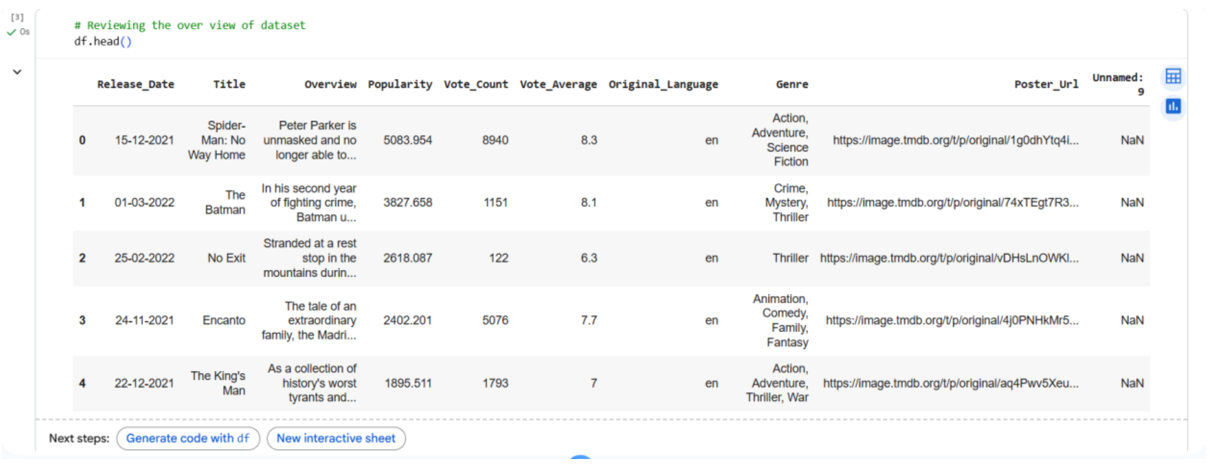


```
[1]
# importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```



```
[2]
# Load the dataset with latin-1 encoding
# Note: Using latin-1 encoding to handle special characters properly
from google.colab import files
uploaded = files.upload()
# Select your 'csv' file from the computer
import pandas as pd
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name, encoding="latin-1")
print("Loaded successfully!", df.shape)
```

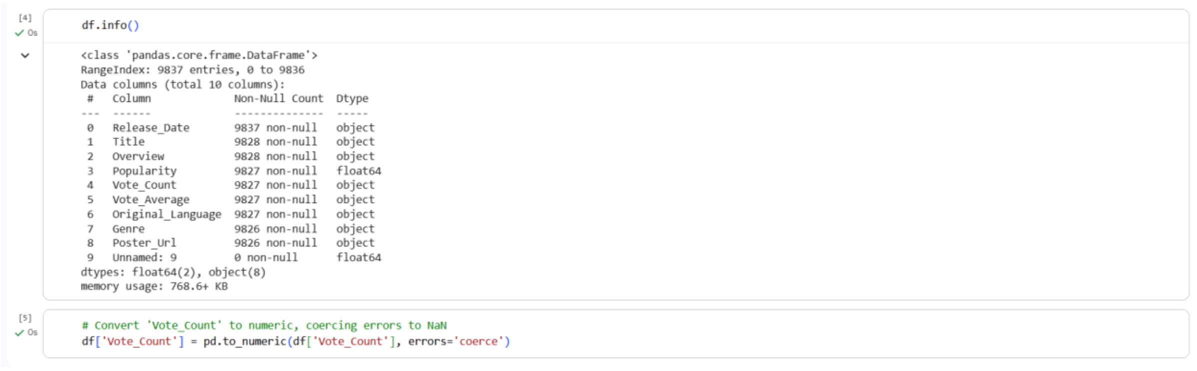
```
Choose Files  mymoviedb.csv
mymoviedb.csv(text/csv) - 4225613 bytes, last modified: 12/3/2025 - 100% done
Saving mymoviedb.csv to mymoviedb.csv
Loaded successfully! (9837, 10)
```



```
[3]
# Reviewing the over view of dataset
df.head()
```

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	Poster_Url	Unnamed: 9
0	15-12-2021	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.org/t/p/original/1g0dhYtq4i...	NaN
1	01-03-2022	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151	8.1	en	Crime, Mystery, Thriller	https://image.tmdb.org/t/p/original/74xTEgt7R3...	NaN
2	25-02-2022	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...	NaN
3	24-11-2021	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...	NaN
4	22-12-2021	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793	7	en	Action, Adventure, Thriller, War	https://image.tmdb.org/t/p/original/aq4Pwv5Xeu...	NaN

Next steps: Generate code with df New interactive sheet



```
[4]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9837 entries, 0 to 9836
Data columns (total 10 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
 9   Unnamed: 9         0 non-null      float64
dtypes: float64(2), object(8)
memory usage: 768.6+ KB
```

```
[5]
# Convert 'Vote_Count' to numeric, coercing errors to NaN
df['Vote_Count'] = pd.to_numeric(df['Vote_Count'], errors='coerce')
```

[6]

```
# Verify the data type and non-null counts after conversion
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9837 entries, 0 to 9836
Data columns (total 10 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         9826 non-null   float64
 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 
 9   Unnamed: 9         0 non-null      float64
dtypes: float64(3), object(7)
memory usage: 768.6+ KB
```

looks like our dataset has no NaNs!

Overview, Original_Language , Poster-Url, and Unnamed: 9 wouldn't be so useful during analysis, and Release_Date column needs to be casted into date time and to extract only the year value

[7]

```
#exploring genres column
df['Genre'].head()
```

	Genre
0	Action, Adventure, Science Fiction
1	Crime, Mystery, Thriller
2	Thriller
3	Animation, Comedy, Family, Fantasy
4	Action, Adventure, Thriller, War

dtype: object

[8]

```
#checking for duplicated sum
df.duplicated().sum()
```

```
np.int64(0)
```

our dataset has no duplicated rows either.

[9]

```
#Exploring summary statistics
df.describe()
```

	Popularity	Vote_Count	Unnamed: 9
count	9827.000000	9826.000000	0.0
mean	40.320570	1392.943721	NaN
std	108.874308	2611.303856	NaN
min	7.100000	0.000000	NaN
25%	16.127500	146.000000	NaN
50%	21.191000	444.000000	NaN
75%	35.174500	1376.000000	NaN
max	5083.954000	31077.000000	NaN

Exploration Summary

we have a dataframe consisting of 9837 rows and 9 columns.

our dataset looks a bit tidy with no NaNs nor duplicated values.

Release_Date column needs to be casted into date time and to extract only the year

Overview, Original_Languege, Poster-Url, and unnamed :9 wouldn't be so useful during analys, so we are going to remove them.

Vote_Average better be categorised for proper analysis.

Genre column has comma saperated values and white spaces that needs to be handled

Data Cleaning

Casting Release_Date column and extracing year values

[10]

```
df.head()
```

	Release_Date	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	Poster_Url	Unnamed: 9
0	15-12-2021	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940.0	8.3	en	Action, Adventure, Science Fiction	https://image.tmdb.org/t/p/original/1g0dhYtq4i...	NaN
1	01-03-2022	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151.0	8.1	en	Crime, Mystery, Thriller	https://image.tmdb.org/t/p/original/74xTEgt7R3...	NaN
2	25-02-2022	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122.0	6.3	en	Thriller	https://image.tmdb.org/t/p/original/vDHsLnOWKl...	NaN
3	24-11-2021	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076.0	7.7	en	Animation, Comedy, Family, Fantasy	https://image.tmdb.org/t/p/original/4j0PNHkMr5...	NaN
		The King's	As a collection of					Action,		

Creating a new column as Relese_Year and converting it's data from datetime64 to int64.

This Relese_Year column helps to find that which year has netflix relesed the most of the movies.

[11]

```
df['Release_Date'] = pd.to_datetime(df['Release_Date'], errors='coerce')
# confirming changes
print(df['Release_Date'].dtypes)

# Extracting the year from 'Release_Date'
df['Release_Year'] = df['Release_Date'].dt.year

# Fill any NaN values in 'Release_Year' (resulting from NaT in Release_Date) with 0
df['Release_Year'].fillna(0, inplace=True)

# Convert the 'Release_Year' column to integer type
df['Release_Year'] = df['Release_Year'].astype('int')

# Dropping the original 'Release_Date' column as 'Release_Year' is now available
df.drop('Release_Date', axis=1, inplace=True)

# Displaying the first few rows with the new 'Release_Year' column
print(df[['Title', 'Release_Year']].head())

# Confirming the data type of the new 'Release_Year' column
print(df['Release_Year'].dtypes)
```

```
datetime64[ns]
                   Title  Release_Year
0  Spider-Man: No Way Home          2021
1               The Batman          2022
2                  No Exit          2022
3                  Encanto          2021
4            The King's Man          2021
int64
/tmp/ipython-input-4279685774.py:1: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pass `dayfirst=True` or specify a format t
  df['Release_Date'] = pd.to_datetime(df['Release_Date'], errors='coerce')
/tmp/ipython-input-4279685774.py:9: 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 o


  df['Release_Year'].fillna(0, inplace=True)
```

[12]

```
print(df['Release_Year'].dtypes)
```

```
int64
```

[13]

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9837 entries, 0 to 9836
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Title              9828 non-null   object
 1   Overview           9828 non-null   object
 2   Popularity         9827 non-null   float64
 3   Vote_Count         9826 non-null   float64
 4   Vote_Average       9827 non-null   object
 5   Original_Language  9827 non-null   object
 6   Genre              9826 non-null   object
 7   Poster_Url         9826 non-null   object
 8   Unnamed: 9         0 non-null      float64
 9   Release_Year       9837 non-null   int64
dtypes: float64(3), int64(1), object(6)
memory usage: 768.6+ KB
```

[14]

```
df.head()
```

	Title	Overview	Popularity	Vote_Count	Vote_Average	Original_Language	Genre	Poster_Url	Unnamed: 9	Release_Year
0	Spider-Man: No Way Home	Peter Parker is unmasked and no longer able to...	5083.954	8940.0	8.3	en	Action, Adventure, Science Fiction	https://image.tmdb.org/t/p/original/1g0dhYtq4i...	NaN	2021
1	The Batman	In his second year of fighting crime, Batman u...	3827.658	1151.0	8.1	en	Crime, Mystery, Thriller	https://image.tmdb.org/t/p/original/74xTEgt7R3...	NaN	2022
2	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122.0	6.3	en	Thriller	https://image.tmdb.org/t/p/original/vDHsLnOWKl...	NaN	2022
3	Encanto	The tale of an extraordinary family, the Madri...	2402.201	5076.0	7.7	en	Animation, Comedy, Family, Fantasy	https://image.tmdb.org/t/p/original/4j0PNHkMr5...	NaN	2021
4	The King's Man	As a collection of history's worst tyrants and...	1895.511	1793.0	7	en	Action, Adventure, Thriller, War	https://image.tmdb.org/t/p/original/aq4Pwv5Xeu...	NaN	2021

Next steps: Generate code with df New interactive sheet

Dropping Overview, Original_Languege, Poster-Url, and Unnamed :9 cloumns

[15]

```
 # making list of column to be dropped
cols = ['Overview', 'Original_Language', 'Poster_Url','Unnamed: 9' ]
# dropping columns and confirming changes
df.drop(cols, axis = 1, inplace = True)
df.columns
```

```
Index(['Title', 'Popularity', 'Vote_Count', 'Vote_Average', 'Genre',
       'Release_Year'],
      dtype='object')
```

[16]

```
df.head()
```

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
0	Spider-Man: No Way Home	5083.954	8940.0	8.3	Action, Adventure, Science Fiction	2021
1	The Batman	3827.658	1151.0	8.1	Crime, Mystery, Thriller	2022
2	No Exit	2618.087	122.0	6.3	Thriller	2022
3	Encanto	2402.201	5076.0	7.7	Animation, Comedy, Family, Fantasy	2021
4	The King's Man	1895.511	1793.0	7	Action, Adventure, Thriller, War	2021

[17]

```
# Converting 'Vote_Average' from object to float64
df['Vote_Average'] = pd.to_numeric(df['Vote_Average'], errors='coerce')
```

[18]

```
# Verify the data type and non-null counts after conversion
print("DataFrame Info after 'Vote_Average' conversion:")
df.info()
```

```
DataFrame Info after 'Vote_Average' conversion:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9837 entries, 0 to 9836
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Title         9828 non-null   object
 1   Popularity    9827 non-null   float64
 2   Vote_Count    9826 non-null   float64
 3   Vote_Average  9826 non-null   float64
 4   Genre         9826 non-null   object
 5   Release_Year  9837 non-null   int64
dtypes: float64(3), int64(1), object(2)
memory usage: 461.2+ KB
```

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.

[19]

```
def catigorize_col (df, col, labels):
    """
    catigorizes a certain column based on its quartiles

    Args:
        (df)     df   - dataframe we are proccesing
        (col)    str  - to be catigorized column's name
        (labels) list - list of labels from min to max

    Returns:
        (df)     df   - dataframe with the categorized col
    """

    # setting the edges to cut the column accordingly
    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
```

[20] ✓ 0s
```
# define labels for edges
labels = ['not_popular', 'below_avg', 'average', 'popular']
 # categorize column based on labels and edges
catigorize_col(df, 'Vote_Average', labels)
 # confirming changes
df['Vote_Average'].unique()
```

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

[21] ✓ 0s
```
df.head()
```

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
0	Spider-Man: No Way Home	5083.954	8940.0	popular	Action, Adventure, Science Fiction	2021
1	The Batman	3827.658	1151.0	popular	Crime, Mystery, Thriller	2022
2	No Exit	2618.087	122.0	below_avg	Thriller	2022
3	Encanto	2402.201	5076.0	popular	Animation, Comedy, Family, Fantasy	2021
4	The King's Man	1895.511	1793.0	average	Action, Adventure, Thriller, War	2021

Next steps: Generate code with df New interactive sheet

[22] ✓ 0s
```
# exploring column
df['Vote_Average'].value_counts()
```

	count
Vote_Average	
not_popular	2467
popular	2450
average	2411
below_avg	2398

dtype: int64

[23] ✓ 0s
```
# dropping NaNs
df.dropna(inplace = True)
 # confirming
df.isna().sum()
```

	0
Title	0
Popularity	0
Vote_Count	0
Vote_Average	0
Genre	0
Release_Year	0

dtype: int64

[24] ✓ 0s
```
df.head()
```

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
0	Spider-Man: No Way Home	5083.954	8940.0	popular	Action, Adventure, Science Fiction	2021
1	The Batman	3827.658	1151.0	popular	Crime, Mystery, Thriller	2022
2	No Exit	2618.087	122.0	below_avg	Thriller	2022
3	Encanto	2402.201	5076.0	popular	Animation, Comedy, Family, Fantasy	2021
4	The King's Man	1895.511	1793.0	average	Action, Adventure, Thriller, War	2021

Next steps: Generate code with df New interactive sheet

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

[25] ✓ 0s
```
 # split the strings into lists
df['Genre'] = df['Genre'].str.split(', ')
 # explode the lists
df = df.explode('Genre').reset_index(drop=True)
df.head()
```

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
0	Spider-Man: No Way Home	5083.954	8940.0	popular	Action	2021
1	Spider-Man: No Way Home	5083.954	8940.0	popular	Adventure	2021
2	Spider-Man: No Way Home	5083.954	8940.0	popular	Science Fiction	2021
3	The Batman	3827.658	1151.0	popular	Crime	2022
4	The Batman	3827.658	1151.0	popular	Mystery	2022

Next steps: Generate code with df New interactive sheet

[26] ✓ 0s
```
# casting column into category
df['Genre'] = df['Genre'].astype('category')
 # confirming changes
df['Genre'].dtypes
```

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

[27]

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25551 entries, 0 to 25550
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Title         25551 non-null  object
 1   Popularity    25551 non-null  float64
 2   Vote_Count    25551 non-null  float64
 3   Vote_Average  25551 non-null  category
 4   Genre         25551 non-null  category
 5   Release_Year  25551 non-null  int64
dtypes: category(2), float64(2), int64(1), object(1)
memory usage: 849.4+ KB
```

[28]

```
df.nunique()
```

	0
Title	9414
Popularity	8087
Vote_Count	3265
Vote_Average	4
Genre	19
Release_Year	100

dtype: int64

Now that our dataset is clean and tidy, we are left with a total of 6 columns and 25551 rows to dig into during our analysis

Data Visualization

Here, we'd use Matplotlib and seaborn for making some informative visuals to gain insights abut our data.

[29]

```
# setting up seaborn configurations
sns.set_style('whitegrid')
```

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

[30]

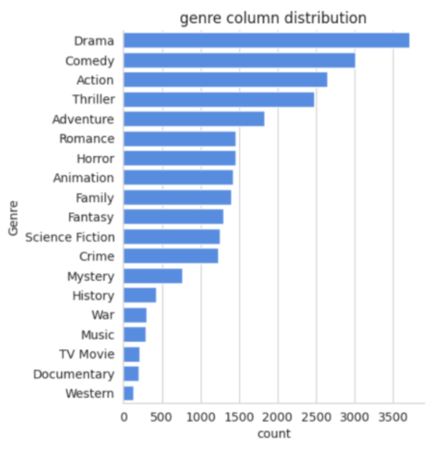
```
# showing stats. on genre column
df['Genre'].describe()
```

	Genre
count	25551
unique	19
top	Drama
freq	3715

dtype: object

[31]

```
# 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()
```

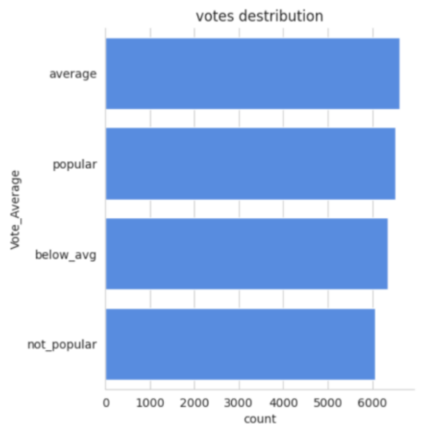


we can notice from the above visual that Drama genre is the most frequent genre in our dataset and has appeared more than **14%** of the times among 19 other genres.

Q2: What genres has highest votes ?

[32]

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



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

[33]

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

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
0	Spider-Man: No Way Home	5083.954	8940.0	popular	Action	2021
1	Spider-Man: No Way Home	5083.954	8940.0	popular	Adventure	2021
2	Spider-Man: No Way Home	5083.954	8940.0	popular	Science Fiction	2021

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

[34]

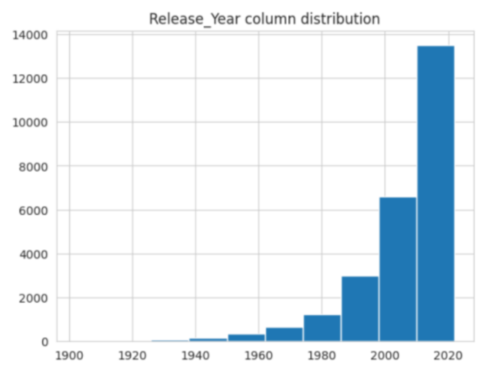
```
# checking max popularity in dataset
df[df['Popularity'] == df['Popularity'].min()]
```

	Title	Popularity	Vote_Count	Vote_Average	Genre	Release_Year
25545	The United States vs. Billie Holiday	13.354	152.0	average	Music	2021
25546	The United States vs. Billie Holiday	13.354	152.0	average	Drama	2021
25547	The United States vs. Billie Holiday	13.354	152.0	average	History	2021
25548	Threads	13.354	186.0	popular	War	1984
25549	Threads	13.354	186.0	popular	Drama	1984
25550	Threads	13.354	186.0	popular	Science Fiction	1984

Q5: Which year has the most filmmed movies?

[35]

```
df['Release_Year'].hist()
plt.title('Release_Year column distribution')
plt.show()
```



Conclusion

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

Ans: **Drama** genre is the most frequent genre in our dataset and has appeared more than 14% of the times among 19 other genres.

Q2: What genres has highest votes ?

Ans: we have 25.5% of our dataset with popular vote (6520 rows). **Drama** again gets the highest popularity among fans by being having more than 18.5% of movies popularities.

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

Ans: **Spider-Man: No Way Home**has the highest popularity rate in our dataset and it has genres of **Action Adventure** and **Sience Fiction.**

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

Ans: **The united states vs Billie Holiday** and **Threads** has the highest lowest popularity in our dataset and it has genres of **music, drama, History, War and Science Fiction.**

Q5: Which year has the most filmmed movies?

Ans: In the year **2020** netflix has the most flimmed movies.