```
# importing lib.
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
```

```
df = pd.read_csv('/content/drive/MyDrive/mymoviedb.csv', lineterminator='\n')
df.head() # Remove any leading spaces or tabs before this line
```

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

```
# viewing dataset info
df.info() # Removed the extra space before 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
```

• looks like our dataset has no NaNs! • Overview, Original_Language and Poster-Url wouldn't be so useful during analysis • Release_Date column needs to be casted into date time and to extract only the year value

```
# exploring genres column
df['Genre'].head() # Removed the extra space before 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

```
# check for duplicated rows
df.duplicated().sum()
```

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

• our dataset has no duplicated rows either

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

	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 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 • Overview, Original_Languege and Poster-Url wouldn't be so useful during analys • there is noticable outliers in Popularity column • Vote_Average bettter be categorised for proper analysis.

Data Cleaning

• Genre column has comma saperated values and white spaces that needs to be hand Casting Release_Date column and extracing year values

```
df.head()
```

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

```
# casting column a
df['Release_Date'] = pd.to_datetime(df['Release_Date'])
# confirming changes
print(df['Release_Date'].dtypes)
```

```
datetime64[ns]
```

```
df['Release_Date'] = df['Release_Date'].dt.year
df['Release_Date'].dtypes # Removed the extra space before df['Release_Date'].dtypes
```

```
dtype('int32')
```

```
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   int32
 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), int32(1), int64(1), object(5)
memory usage: 652.7+ KB
```

```
df.head()
```

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

Dropping Overview, Original_Languege and Poster-Url

```
# making list of column to be dropped
cols = ['Overview', 'Original_Language', 'Poster_Url']
```

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

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

```
df.head()
```

	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,

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.

```
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']]
    # The following line was incorrectly indented, causing the error.
    # It should be indented by 4 spaces, not 8.
    df[col] = pd.cut(df[col], edges, labels=labels, duplicates='drop')
    return df # Also, make sure to return 'df' instead of 'd'


# 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']
```

```
df.head()
```

	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,

```
# exploring column
df['Vote_Average'].value_counts()
```

Vote_Average	count
not_popular	2467
popular	2450
average	2412
below_avg	2398

dtype: int64

```
# dropping NaNs
df.dropna(inplace = True)
# confirming
df.isna().sum()
```

	0
Release_Date	0
Title	0
Popularity	0
Vote_Count	0
Vote_Average	0
Genre	0

dtype: int64

```
df.head()
```

	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,

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

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

	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

```
# 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',
                  'TV Movie', 'Thriller', 'War', 'Western'],
, ordered=False, categories_dtype=object)
```

```
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
```

```
df.nunique()
```

	0
Release_Date	100
Title	9415
Popularity	8088
Vote_Count	3265
Vote_Average	4
Genre	19

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.

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

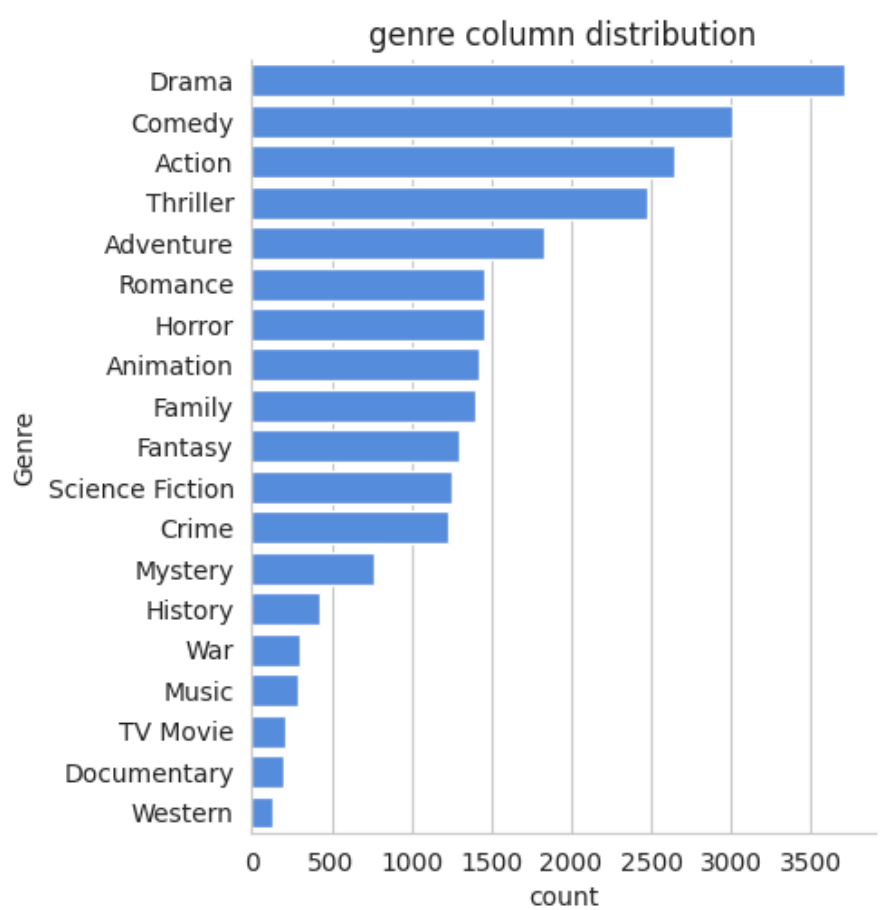
Q1: What is the most frequent genre in the dataset?

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

	Genre
count	25552
unique	19
top	Drama
freq	3715

dtype: object

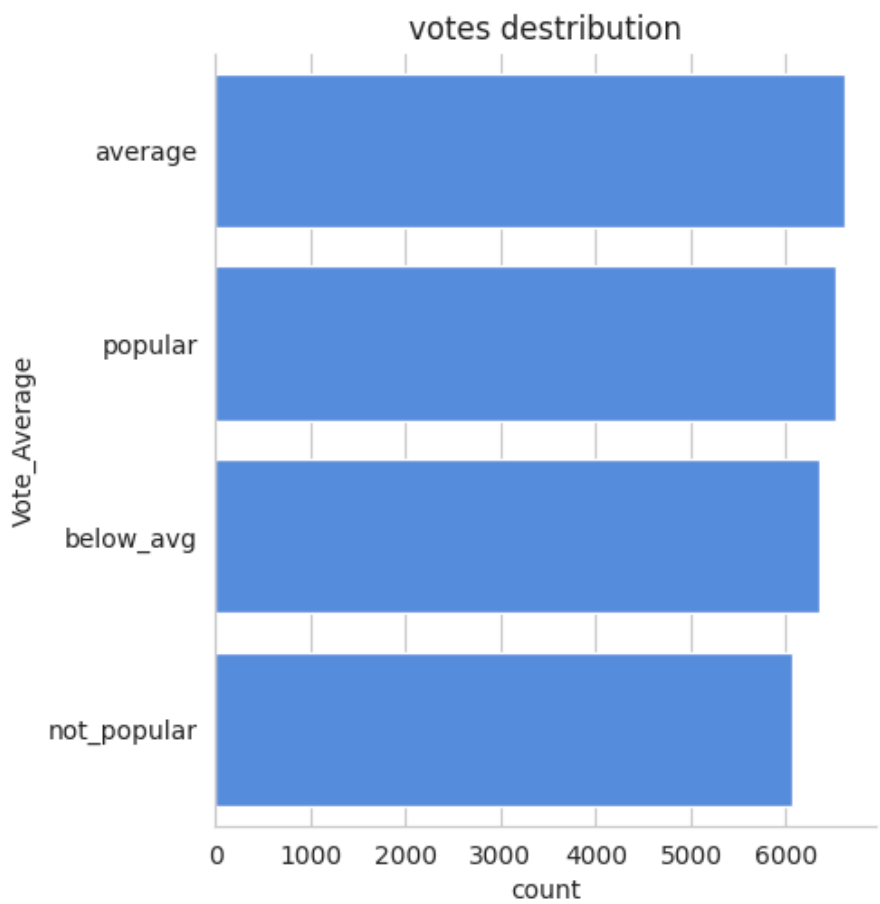
```
import matplotlib as plt


import matplotlib.pyplot as plt # Import matplotlib.pyplot as plt
import seaborn as sns
# setting up seaborn configurations
sns.set_style('whitegrid')

# ... your other code ...

# visualizing genre column
sns.catplot(y = 'Genre', data = df, kind = 'count',
            order = df['Genre'].value_counts().index,
            color = '#4287f5')
plt.title('genre column distribution') # Now plt.title should work
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 In [71]: genres has highest votes ?

```
# 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 ?

```
df[df['Popularity'] == df['Popularity'].max()]
```

	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

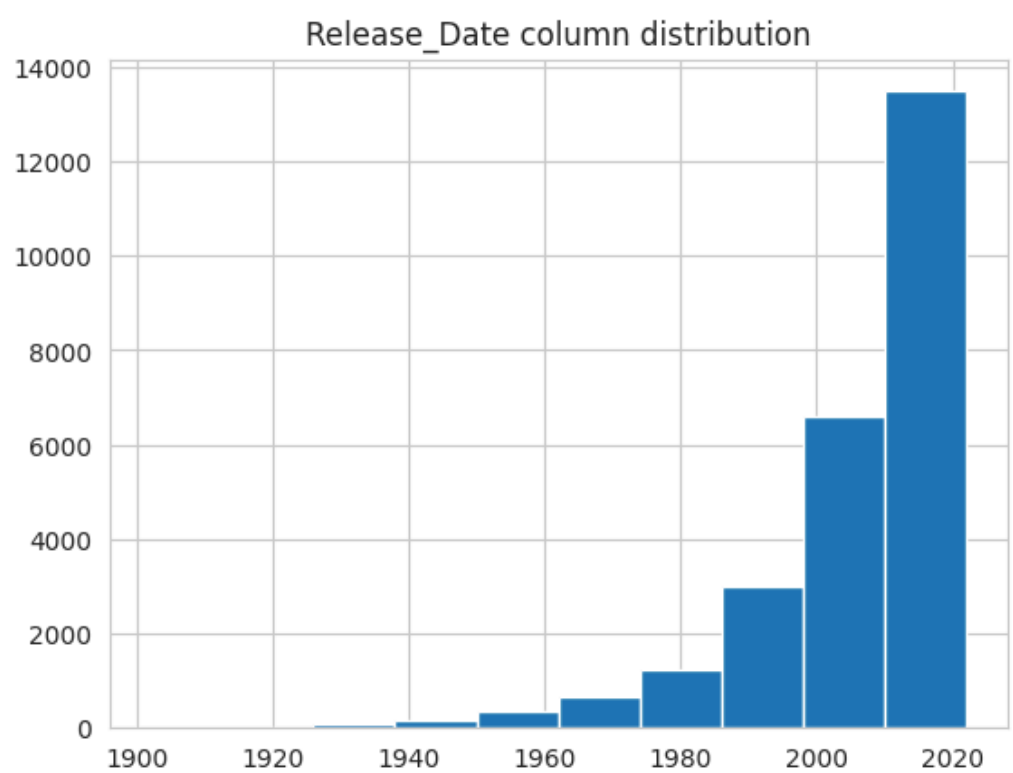
Q4: What movie got the lowest popularity? what's its genre?

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

	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

Q5: Which year has the most filmmed movies?

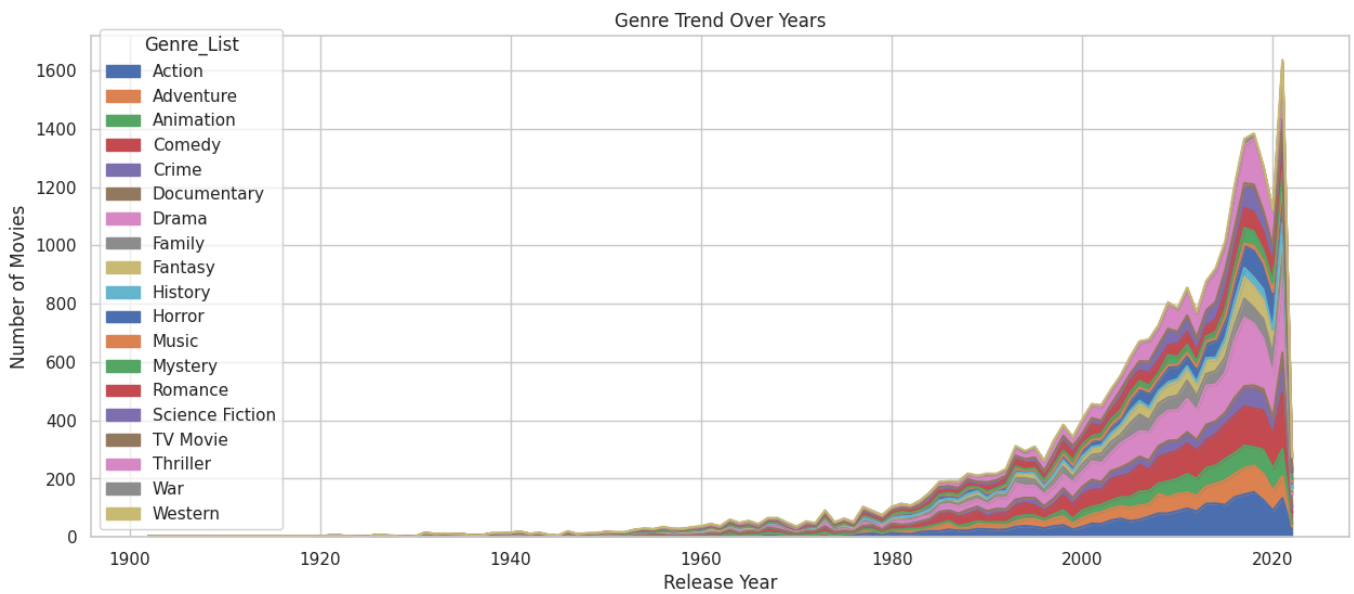
```
df['Release_Date'].hist()
plt.title('Release_Date column distribution')
plt.show()
```




```
from collections import Counter
from itertools import chain

# Expand genres per movie
df['Genre_List'] = df['Genre'].str.split(', ')
genre_year_data = df.explode('Genre_List').groupby(['Release_Date', 'Genre_List']).size().unstack
```

```
genre_year_data = df.explode('Genre_List').groupby(['Release_Date', 'Genre_List']).size().unstack.
```

```
# Plot trend
genre_year_data.plot(kind='area', stacked=True, figsize=(15, 6))
plt.title('Genre Trend Over Years')
plt.xlabel('Release Year')
plt.ylabel('Number of Movies')
plt.show()
```




```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

tfidf = TfidfVectorizer(stop_words='english')
# Use 'Title' column instead of 'Overview'
tfidf_matrix = tfidf.fit_transform(df['Title'])

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

def recommend_movie(title):
    idx = df[df['Title'].str.lower() == title.lower()].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[1:6]
    movie_indices = [i[0] for i in sim_scores]
    return df['Title'].iloc[movie_indices]

print(recommend_movie("The Batman"))
```

```
4       The Batman
5       The Batman
199         Batman
200         Batman
4434        Batman
Name: Title, dtype: object
```

```
from ipywidgets import interact
```

```
from ipywidgets import interact

@interact(genre=df['Genre_List'].explode().unique(), year=sorted(df['Release_Date'].unique()))
def filter_movies(genre='Action', year=2022):
    filtered = df[df['Genre'].str.contains(genre, na=False) & (df['Release_Date'] == year)]
    display(filtered[['Title', 'Popularity', 'Vote_Average']].sort_values(by='Popularity', ascenc
```

genre TV Movie

year 2022

	Title	Popularity	Vote_Average
302	Laura y el misterio del asesino inesperado	266.754	popular
3513	Ray Donovan: The Movie	56.863	average
15514	The Wedding Veil Legacy	18.948	not_popular
17436	Aurora Teagarden Mysteries: Haunted By Murder	17.536	not_popular
18964	Stolen by Their Father	16.475	not_popular

Start coding or generate with AI.