

```
In [26]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: df = pd.read_csv('mymoviedb.csv', lineterminator = '\n')
```

```
In [5]: df.head(5)
```

```
Out[5]:    Release_Date      Title      Overview  Popularity  Vote_Count  Vote_Average  Original_Lan
```

|   | Release_Date | Title                   | Overview  | Popularity | Vote_Count | Vote_Average | Original_Lan |
|---|--------------|-------------------------|---|------------|------------|--------------|--------------|
| 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 [6]: # viewing 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(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

```
In [7]: # exploring genres column  
df['Genre'].head(5)
```

```
Out[7]: 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
```

- genres are separated by commas followed by whitespaces.

```
In [8]: # check for duplicated rows  
df.duplicated().sum()
```

```
Out[8]: np.int64(0)
```

- our dataset has no duplicated rows either.

```
In [11]: # exploring summary statistics  
df.describe()
```

Out[11]:

|              | Popularity  | Vote_Count   | Vote_Average |
|--------------|-------------|--------------|--------------|
| <b>count</b> | 9827.000000 | 9827.000000  | 9827.000000  |
| <b>mean</b>  | 40.326088   | 1392.805536  | 6.439534     |
| <b>std</b>   | 108.873998  | 2611.206907  | 1.129759     |
| <b>min</b>   | 13.354000   | 0.000000     | 0.000000     |
| <b>25%</b>   | 16.128500   | 146.000000   | 5.900000     |
| <b>50%</b>   | 21.199000   | 444.000000   | 6.500000     |
| <b>75%</b>   | 35.191500   | 1376.000000  | 7.100000     |
| <b>max</b>   | 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\_Language and Poster-Url wouldn't be so useful during analysis
- there is noticeable outliers in Popularity column
- Vote\_Average better be categorised for proper analysis.
- Genre column has comma separated values and white spaces that needs to be handled

In [12]: df.head(5)

Out[12]:

|   | Release_Date | Title                   | Overview  | Popularity | Vote_Count | Vote_Average | Original_Lan |
|---|--------------|-------------------------|---|------------|------------|--------------|--------------|
| 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          |              |

## Data Cleaning

- Casting Release\_Date column and extracting year values

```
In [18]: # casting column a
df['Release_Date'] = pd.to_datetime(df['Release_Date'])

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

datetime64[ns]

```
In [19]: df['Release_Date'] = df['Release_Date'].dt.year
```

```
In [21]: df['Release_Date'].dtypes
```

```
Out[21]: dtype('int32')
```

```
In [22]: df.head(5)
```

Out[22]:

|   | Release_Date | Title                   | Overview  | Popularity | Vote_Count | Vote_Average | Original_Lan |
|---|--------------|-------------------------|---|------------|------------|--------------|--------------|
| 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          |              |

## Dropping Overview, Original\_Language and Poster-Url

```
In [23]: cols = ['Overview', 'Original_Language', 'Poster_Url']
```

```
In [24]: df.drop(cols, axis = 1, inplace = True)
```

```
In [25]: df.head(5)
```

Out[25]:

|   | 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 [27]:

```
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
```

In [28]:

```
# 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()
```

```
Out[28]: ['popular', 'below_avg', 'average', 'not_popular', NaN]
Categories (4, object): ['not_popular' < 'below_avg' < 'average' < 'popular']
```

```
In [29]: df.head(5)
```

|   | 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 [30]: # exploring column
df['Vote_Average'].value_counts()
```

```
Out[30]: Vote_Average
not_popular    2467
popular        2450
average         2412
below_avg      2398
Name: count, dtype: int64
```

```
In [31]: # dropping Nans
df.dropna(inplace = True)
# confirming
df.isna().sum()
```

```
Out[31]: Release_Date    0
Title          0
Popularity     0
Vote_Count     0
Vote_Average   0
Genre          0
dtype: int64
```

```
In [32]: df.head(5)
```

Out[32]:

|   | 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   |

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

In [33]:

```
# 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[33]:

|   | 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 [34]:

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

```
Out[34]: 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 [35]: `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 [36]: `df.nunique()`

```
Out[36]: 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 about our data.

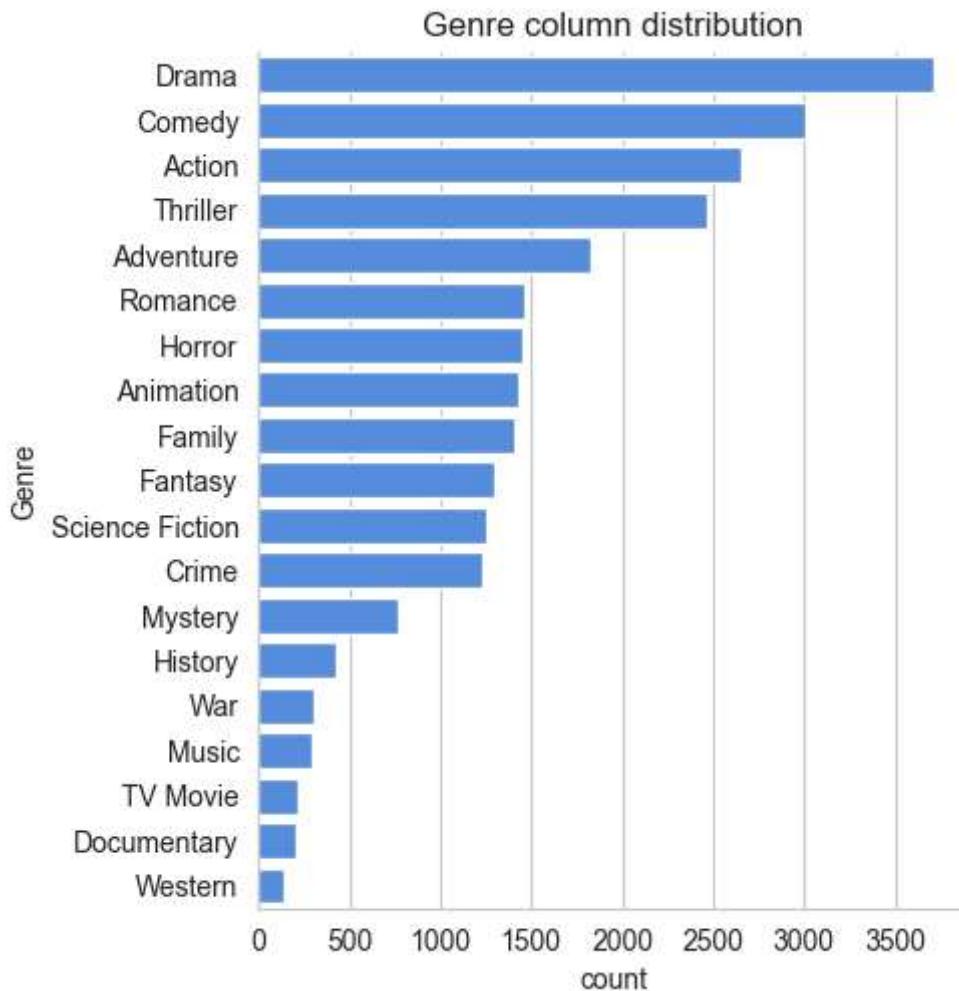
In [37]: `# setting up seaborn configurations  
sns.set_style('whitegrid')`

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

In [38]: `# showing stats. on genre column  
df['Genre'].describe()`

```
Out[38]: count    25552
unique      19
top        Drama
freq      3715
Name: Genre, dtype: object
```

```
In [40]: # 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()
```

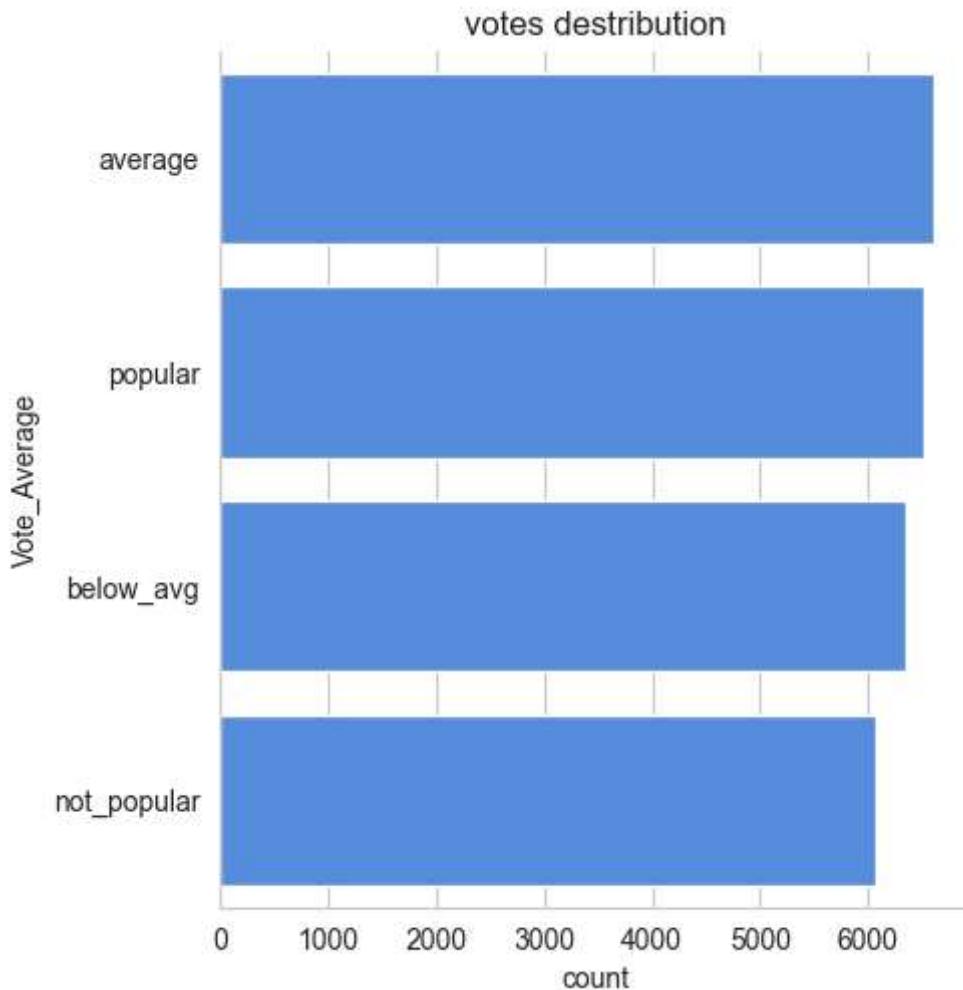


- we can notice from the above visual that Drama genre is the most frequent genre in our dataset and has appeared more than 14% of the times among 19 other genres.

## Q2: What genres has highest votes ?

```
In [41]: # 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 distribution')
plt.show()
```



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

In [42]:

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

Out[42]:

|   | 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?

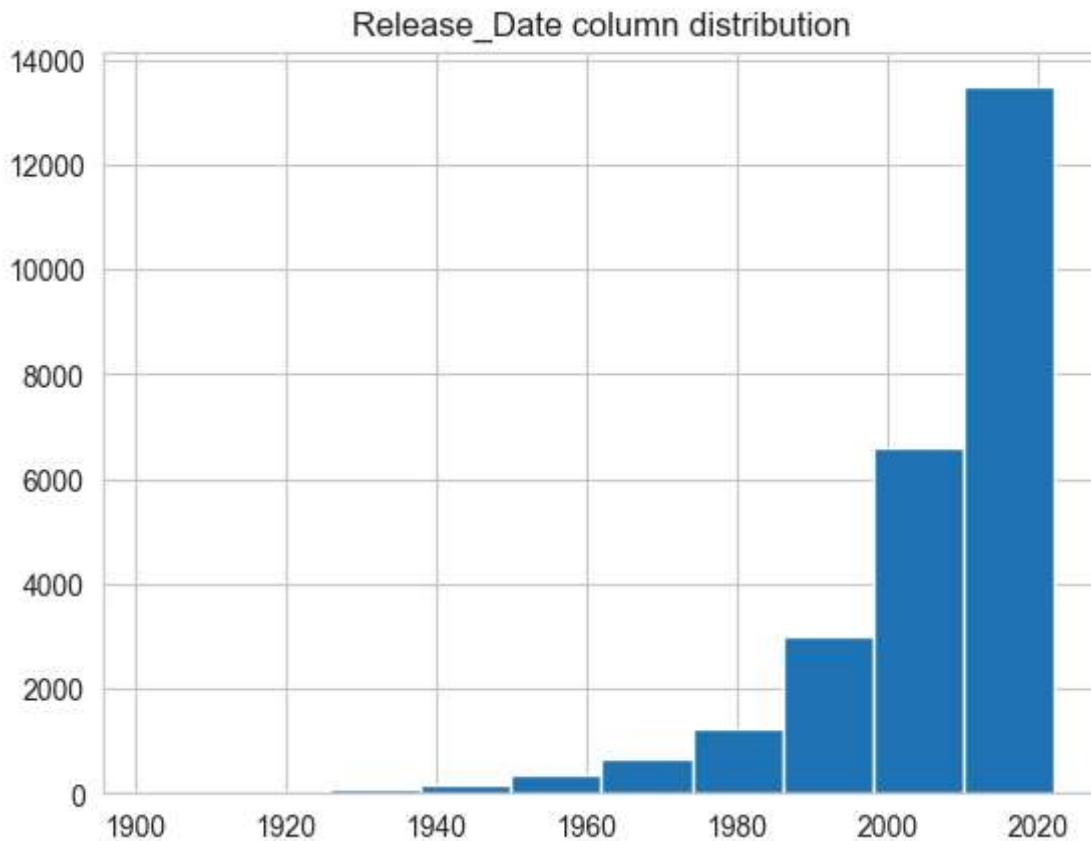
```
In [43]: # checking max popularity in dataset
df[df['Popularity'] == df['Popularity'].min()]
```

Out[43]:

|       | 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 |

## Q5: Which year has the most filmmmed movies?

```
In [44]: df['Release_Date'].hist()
plt.title('Release_Date column distribution')
plt.show()
```



In [ ]:

## Conclusion

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

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 ?

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 ?

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 ?

The united states, thread' has the highest lowest rate in our dataset and it has genres of music , drama , 'war', 'sci-fi' and history'.

### Q5: Which year has the most filammed movies?

year 2020 has the highest filmming rate in our dataset.

In [ ]: