In [2]:

```
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')  #lineterminator is used to show all the rows in every nex
df
```

Out[3]:

	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.t
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.tn
2	2022-02-25	No Exit	Stranded at a rest stop in the mountains durin...	2618.087	122	6.3	en	Thriller	https://image.tmc
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.tn
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.tm
...	...	...	...	...	...	...	...	...	
9822	1973-10-15	Badlands	A dramatization of the Starkweather-Fugate kil...	13.357	896	7.6	en	Drama, Crime	https://image.tr
9823	2020-10-01	Violent Delights	A female vampire falls in love with a man she ...	13.356	8	3.5	es	Horror	https://image.tn
9824	2016-05-06	The Offering	When young and successful reporter Jamie finds...	13.355	94	5.0	en	Mystery, Thriller, Horror	https://image.tmd
9825	2021-03-31	The United States vs. Billie Holiday	Billie Holiday spent much of her career being ...	13.354	152	6.7	en	Music, Drama, History	https://image.tn
9826	1984-09-23	Threads	Documentary style account of a nuclear holocau...	13.354	186	7.8	en	War, Drama, Science Fiction	https://image.tm

9827 rows × 9 columns

In [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.info()  #info() is used to show the datatypes of the columns and if there is any missing or null value then
```

```
<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 [6]:

```
# We will change the datatype of the Release_Date column because it is in string datatype(Object). And we will
```

In [7]:

```
df['Genre'].head() #Show the first 5 movies Genre.
```

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
```

In [8]:

```
df.duplicated().sum()  #It counts the number of duplicate rows in a DataFrame and value 0 means there is no dup
```

Out[8]:

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

In [9]:

```
df.describe()  #describe() function is used to do some statistical problems on those columns whose datatype is
```

Out[9]:

	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.
- Genre column has comma saperated values and white spaces that needs to be hand

"Data Cleaning":-->

In [10]:
```
df['Release_Date']=pd.to_datetime(df['Release_Date'])
print(df['Release_Date'].dtype)
```

```
datetime64[ns]
```

In [11]:
```
df['Release_Date']=df['Release_Date'].dt.year ### By using this we only show the year not the full date.
```

In [12]:
```
df.head()
```

Out[12]:

	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

Delete the unwanted columns

In [13]:
```
df.drop(['Overview', 'Original_Language', 'Poster_Url'], axis="columns", inplace=True, errors='ignore')
```

In [14]:
```
df.head()
```

Out[14]:

	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 [15]:

```python
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
     """

    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') #edges=[min-25,25-50,50-75,75-100], labels=['
    return df
```

In [16]:

```python
#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[16]:

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

In [17]:

```python
df.head()
```

Out[17]:

	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 [18]:

```python
df['Vote_Average'].value_counts() #check how many movies fall under the 'popular' and other categories
```

Out[18]:

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

In [19]:

```python
df.dropna(inplace=True) #drop the NaN values

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

Out[19]:

```
Release_Date    0
Title           0
Popularity      0
Vote_Count      0
Vote_Average    0
Genre           0
dtype: int64
```

In [20]:

```python
df.head()
```

Out[20]:

	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 want to split every genre category into different lines for every movie.

In [21]:

```
# 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[21]:

	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 [22]:

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

Out[22]:

```
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 [23]:

```
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 [24]:

```
df.nunique()
```

Out[24]:

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

In [25]:

```
df.head()
```

Out[25]:

	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

Solve questions with Data Visualization:-

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

In [26]:

```
df['Genre'].describe()
```

Out[26]:

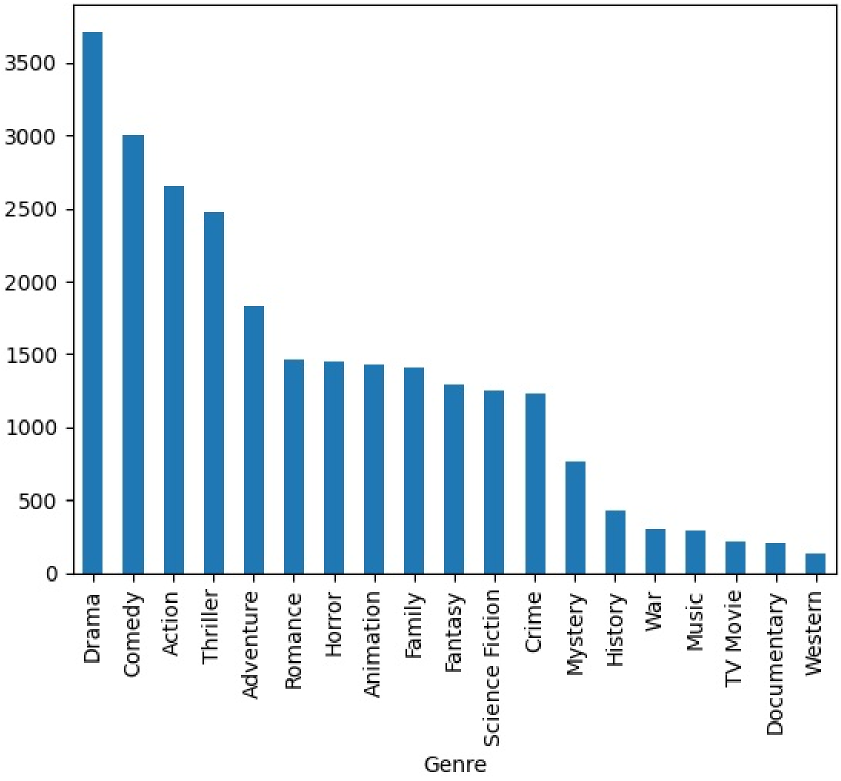
```
count     25552
unique       19
top       Drama
freq       3715
Name: Genre, dtype: object
```

In [27]:

```
df['Genre'].value_counts().plot(kind='bar')
```

Out[27]:

```
<Axes: xlabel='Genre'>
```



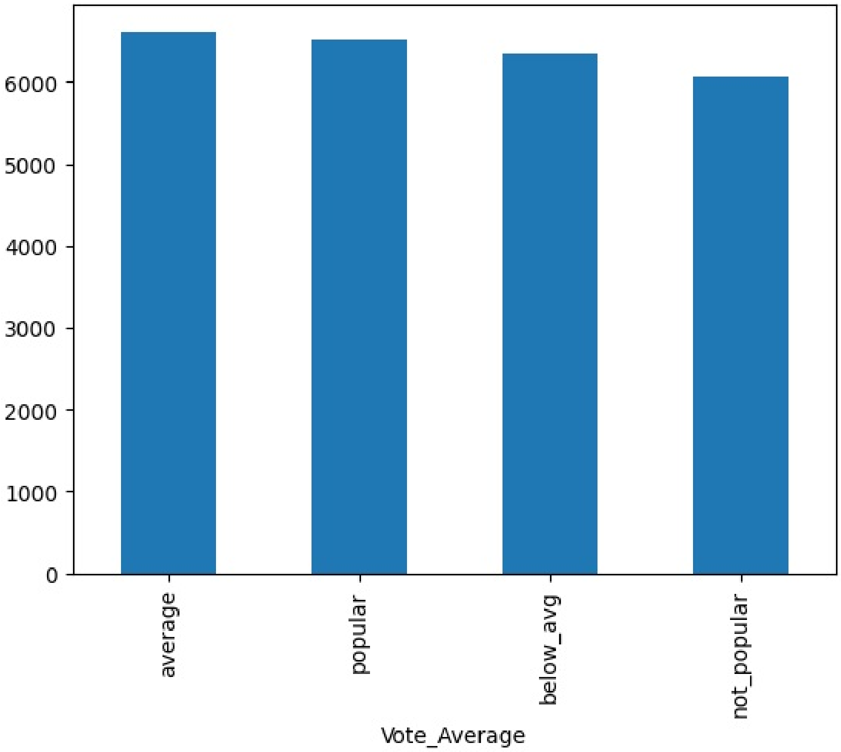
Q2: What Vote_Average has highest votes ?

In [28]:

```
df['Vote_Average'].value_counts().plot(kind='bar')
```

Out[28]:

```
<Axes: xlabel='Vote_Average'>
```



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

```
In [29]: df[df['Popularity']==df['Popularity'].max()]
```

Out[29]:

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

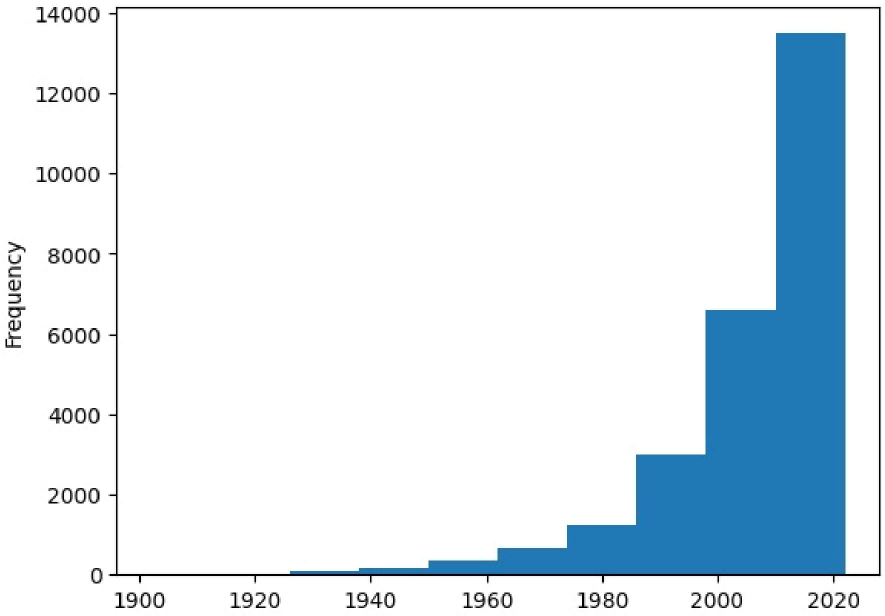
Out[30]:

	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 filmmed movies?

```
In [34]: df['Release_Date'].plot(kind='hist')
```

```
Out[34]: <Axes: ylabel='Frequency'>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js