```python
from IPython.display import Image
image_path="dataset-cover.jpg"
Image(filename=image_path)
```

Library Imports and Loading Dataset

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import os

warnings.simplefilter(action="ignore")

print(os.listdir())
```

```
['.config', 'Music', 'CAD_processed_dataset.csv', 'COMPLETED_IPL
Analysis 2008-2022.ipynb', '.DS_Store',
'MELBOURNE_HOUSE_PRICES_LESS.csv', '.CFUserTextEncoding',
'Diabetes.ipynb', 'Completed_Flipkart_Mobile_Data_Analysis.ipynb',
'Flipkart_Mobiles.csv', 'DataSet', 'Analysis Help Kit.ipynb',
'Pictures', '.zprofile', 'Clearing doubts.ipynb',
'college_job_placement_analysis.ipynb', 'tmdb_5000_credits.csv',
'.zsh_history', '.ipython', 'spotify_top_songs_audio_features.csv',
'Desktop', 'Library', '.matplotlib', 'Handling Missing Value.ipynb',
'job_placement.csv', 'Hacker Rank Problems.ipynb', 'PycharmProjects',
'Public', 'IPL_Matches_2008_2022.csv', '.idlerc', 'dataset-cover.jpg',
'flixpatrol.csv', 'Movies', 'Applications',
'Flipkart_mobile_brands_scraped_data.csv', '.Trash', 'Most Watched
Movies and TV Shows.ipynb', '.ipynb_checkpoints', '.jupyter',
```

```
'.keras', 'diabetes.csv', 'Documents', '.vscode', 'Downloads',
'.zsh_sessions', 'Print Not Repeated elements.ipynb']

data=pd.read_csv(r"flixpatrol.csv")
```

Copy of data

```
dataset=data.copy()
```

Display First 5 rows

```
dataset.head()

   Rank                                Title     Type  Premiere
Genre  \
0   1.0                      The Night Agent  TV Show    2023.0
Action
1   2.0                      Ginny & Georgia  TV Show    2021.0
Drama
2   3.0                            The Glory  TV Show    2022.0
Thriller
3   4.0                            Wednesday  TV Show    2022.0
Fantasy
4   5.0  Queen Charlotte: A Bridgerton Story  TV Show    2023.0
Drama

     Watchtime Watchtime in Million
0  812,100,000               812.1M
1  665,100,000               665.1M
2  622,800,000               622.8M
3  507,700,000               507.7M
4  503,000,000               503.0M
```

Display Last 5 rows

```
dataset.tail()

          Rank                  Title     Type  Premiere      Genre
Watchtime  \
18159  18210.0        Spiritual House  TV Show    2017.0  Talk Show
100,000
18160  18211.0       Suite Francaise    Movie    2014.0        War
100,000
18161  18212.0  The Bishop's Bedroom    Movie    1977.0     Comedy
100,000
18162  18213.0       30 Chưa Phải Tết    Movie    2020.0     Comedy
100,000
18163  18214.0      The Promised Land    Movie    2019.0      Crime
100,000
```

```
      Watchtime in Million
18159                 0.1M
18160                 0.1M
18161                 0.1M
18162                 0.1M
18163                 0.1M
```

Number of rows and columns in dataset

```
dataset.shape
```

```
(18164, 7)
```

Let's dive deep into the dataset

```
dataset.columns
```

```
Index(['Rank', 'Title', 'Type', 'Premiere', 'Genre', 'Watchtime',
       'Watchtime in Million'],
      dtype='object')
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18164 entries, 0 to 18163
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Rank                  18164 non-null  float64
 1   Title                 18164 non-null  object 
 2   Type                  18164 non-null  object 
 3   Premiere              18030 non-null  float64
 4   Genre                 17984 non-null  object 
 5   Watchtime             18164 non-null  object 
 6   Watchtime in Million  18164 non-null  object 
dtypes: float64(2), object(5)
memory usage: 993.5+ KB
```

```
dataset.describe()
```

```
               Rank      Premiere
count  18164.000000  18030.000000
mean    9126.719335   2014.188297
std     5252.511432      8.844017
min        1.000000   1940.000000
25%     4591.750000   2012.000000
50%     9132.500000   2017.000000
75%    13673.250000   2020.000000
max    18214.000000   2023.000000
```

Check Missing Value

```
dataset.isnull().sum()
```

```
Rank                      0
Title                     0
Type                      0
Premiere                134
Genre                   180
Watchtime                 0
Watchtime in Million      0
dtype: int64
```

To determine unique values

```
for i in dataset.columns:
    print(i,":- \n",dataset[i].unique())
```

```
Rank :-
 [1.0000e+00 2.0000e+00 3.0000e+00 ... 1.8212e+04 1.8213e+04
1.8214e+04]
Title :-
 ['The Night Agent' 'Ginny & Georgia' 'The Glory' ...
 "The Bishop's Bedroom" '30 Chưa Phải Tết' 'The Promised Land']
Type :-
 ['TV Show' 'Movie']
Premiere :-
 [2023. 2021. 2022. 2018. 2011. 2020. 2012. 2013. 2010. 2016. 2003.
2019.
 2008. 2017. 2000. 2004. 2015. 2014. 2009. 1996. 2005. 1994. 2007.
2001.
 1989. 1997. 2006. 2002. 1993.   nan 1999. 1995. 1972. 1983. 1978.
1998.
 1974. 1986. 1988. 1991. 1976. 1985. 1987. 1992. 1977. 1990. 1979.
1973.
 1982. 1966. 1984. 1980. 1975. 1940. 1963. 1970. 1981. 1964. 1960.
1971.
 1968. 1969. 1962. 1954. 1961. 1953. 1957. 1956. 1958. 1965. 1951.
1955.
 1967. 1952.]
Genre :-
 ['Action' 'Drama' 'Thriller' 'Fantasy' 'Crime' 'Reality-Show'
'Comedy'
 'History' 'Superhero' 'Animation' 'Science Fiction' 'Horror'
'Adventure'
 'Documentary' 'War' 'Musical' nan 'Romance' 'Family' 'Stand-Up'
'Western'
 'Sports' 'Biography' 'Talk Show' 'Game-Show' 'Broadcast' 'Concerts'
 'News' 'Fairy Tale']
Watchtime :-
```

```
 ['812,100,000' '665,100,000' '622,800,000' '507,700,000'
'503,000,000'
 '440,600,000' '429,600,000' '402,500,000' '302,100,000' '266,200,000'
 '262,600,000' '252,500,000' '251,500,000' '249,900,000' '235,000,000'
 '234,800,000' '229,700,000' '221,100,000' '214,100,000' '209,700,000'
 '206,500,000' '205,500,000' '201,800,000' '200,700,000' '194,700,000'
 '192,900,000' '184,000,000' '182,300,000' '181,800,000' '176,800,000'
 '175,500,000' '174,300,000' '173,600,000' '172,400,000' '170,100,000'
 '168,300,000' '163,000,000' '162,000,000' '161,100,000' '157,600,000'
 '155,300,000' '153,900,000' '153,000,000' '152,100,000' '151,500,000'
 '151,400,000' '150,200,000' '149,300,000' '148,600,000' '146,900,000'
 '146,700,000' '142,900,000' '140,100,000' '139,900,000' '139,300,000'
 '136,800,000' '136,600,000' '136,200,000' '135,900,000' '134,800,000'
 '133,600,000' '133,500,000' '133,400,000' '132,100,000' '130,700,000'
 '129,200,000' '129,100,000' '126,400,000' '124,400,000' '123,500,000'
 '120,700,000' '120,500,000' '120,300,000' '120,000,000' '118,900,000'
 '118,600,000' '116,500,000' '116,200,000' '115,800,000' '113,600,000'
 '107,200,000' '107,000,000' '106,600,000' '104,600,000' '104,500,000'
 '104,300,000' '102,800,000' '102,300,000' '101,700,000' '99,900,000'
 '99,500,000' '99,000,000' '98,500,000' '97,800,000' '97,600,000'
 '96,400,000' '95,800,000' '95,700,000' '95,100,000' '95,000,000'
 '94,700,000' '94,600,000' '94,400,000' '94,300,000' '94,200,000'
 '92,900,000' '92,500,000' '92,300,000' '92,200,000' '91,400,000'
 '91,200,000' '90,800,000' '90,200,000' '89,000,000' '88,600,000'
 '87,900,000' '87,300,000' '87,200,000' '86,200,000' '86,100,000'
 '86,000,000' '85,400,000' '85,000,000' '84,600,000' '84,400,000'
 '83,600,000' '83,200,000' '82,800,000' '82,500,000' '82,400,000'
 '82,100,000' '81,800,000' '81,700,000' '81,000,000' '80,800,000'
 '80,500,000' '80,300,000' '80,000,000' '79,700,000' '78,200,000'
 '77,800,000' '77,200,000' '76,600,000' '76,300,000' '75,700,000'
 '75,200,000' '75,100,000' '74,300,000' '73,400,000' '73,300,000'
 '73,100,000' '72,800,000' '72,200,000' '71,600,000' '71,300,000'
 '71,100,000' '71,000,000' '70,600,000' '69,900,000' '69,800,000'
 '69,700,000' '69,500,000' '69,200,000' '69,000,000' '68,900,000'
 '68,500,000' '68,200,000' '68,100,000' '67,800,000' '67,700,000'
 '67,500,000' '67,200,000' '67,100,000' '67,000,000' '66,700,000'
 '66,500,000' '66,000,000' '65,900,000' '65,300,000' '65,200,000'
 '64,400,000' '64,300,000' '64,200,000' '63,900,000' '63,700,000'
 '63,500,000' '63,100,000' '62,800,000' '62,700,000' '62,300,000'
 '62,000,000' '61,800,000' '61,600,000' '61,500,000' '61,300,000'
 '61,100,000' '61,000,000' '60,800,000' '60,600,000' '60,100,000'
 '59,900,000' '59,800,000' '59,600,000' '59,300,000' '58,500,000'
 '58,300,000' '57,900,000' '57,800,000' '57,500,000' '57,400,000'
 '57,000,000' '56,700,000' '56,600,000' '56,500,000' '56,400,000'
 '56,300,000' '56,000,000' '55,700,000' '55,500,000' '55,200,000'
 '55,100,000' '55,000,000' '54,800,000' '54,400,000' '53,800,000'
 '53,700,000' '53,600,000' '53,500,000' '53,300,000' '53,100,000'
 '53,000,000' '52,800,000' '52,500,000' '52,400,000' '52,300,000'
 '52,200,000' '52,000,000' '51,800,000' '51,700,000' '51,600,000'
```

```
 '51,300,000' '51,200,000' '51,000,000' '50,900,000' '50,800,000'
 '50,600,000' '50,400,000' '50,300,000' '50,100,000' '50,000,000'
 '49,700,000' '49,600,000' '49,400,000' '49,300,000' '49,200,000'
 '48,900,000' '48,800,000' '48,600,000' '48,500,000' '48,400,000'
 '48,300,000' '48,200,000' '48,100,000' '47,900,000' '47,800,000'
 '47,500,000' '47,100,000' '46,900,000' '46,600,000' '46,500,000'
 '46,400,000' '46,300,000' '46,200,000' '46,100,000' '46,000,000'
 '45,900,000' '45,800,000' '45,700,000' '45,600,000' '45,500,000'
 '45,400,000' '45,300,000' '45,200,000' '45,100,000' '44,900,000'
 '44,800,000' '44,700,000' '44,600,000' '44,300,000' '44,200,000'
 '44,000,000' '43,900,000' '43,800,000' '43,700,000' '43,600,000'
 '43,500,000' '43,400,000' '43,300,000' '43,200,000' '43,100,000'
 '42,900,000' '42,800,000' '42,700,000' '42,600,000' '42,500,000'
 '42,400,000' '42,300,000' '42,100,000' '41,800,000' '41,700,000'
 '41,500,000' '41,400,000' '41,300,000' '41,200,000' '41,100,000'
 '41,000,000' '40,900,000' '40,600,000' '40,500,000' '40,400,000'
 '40,300,000' '40,200,000' '40,100,000' '40,000,000' '39,900,000'
 '39,800,000' '39,700,000' '39,500,000' '39,400,000' '39,300,000'
 '39,200,000' '39,100,000' '38,900,000' '38,700,000' '38,600,000'
 '38,400,000' '38,300,000' '38,200,000' '38,100,000' '38,000,000'
 '37,800,000' '37,700,000' '37,600,000' '37,500,000' '37,400,000'
 '37,300,000' '37,100,000' '37,000,000' '36,900,000' '36,700,000'
 '36,600,000' '36,500,000' '36,400,000' '36,300,000' '36,200,000'
 '36,100,000' '36,000,000' '35,900,000' '35,800,000' '35,700,000'
 '35,600,000' '35,500,000' '35,400,000' '35,300,000' '35,200,000'
 '35,000,000' '34,900,000' '34,800,000' '34,600,000' '34,500,000'
 '34,400,000' '34,300,000' '34,200,000' '34,100,000' '34,000,000'
 '33,900,000' '33,800,000' '33,700,000' '33,500,000' '33,400,000'
 '33,300,000' '33,200,000' '33,100,000' '33,000,000' '32,900,000'
 '32,800,000' '32,700,000' '32,600,000' '32,500,000' '32,400,000'
 '32,300,000' '32,200,000' '32,100,000' '32,000,000' '31,900,000'
 '31,800,000' '31,700,000' '31,600,000' '31,500,000' '31,400,000'
 '31,300,000' '31,200,000' '31,100,000' '31,000,000' '30,900,000'
 '30,800,000' '30,700,000' '30,600,000' '30,500,000' '30,400,000'
 '30,300,000' '30,200,000' '30,100,000' '30,000,000' '29,900,000'
 '29,800,000' '29,700,000' '29,600,000' '29,500,000' '29,400,000'
 '29,300,000' '29,200,000' '29,100,000' '29,000,000' '28,900,000'
 '28,800,000' '28,700,000' '28,600,000' '28,500,000' '28,400,000'
 '28,300,000' '28,200,000' '28,100,000' '28,000,000' '27,900,000'
 '27,800,000' '27,700,000' '27,600,000' '27,500,000' '27,400,000'
 '27,300,000' '27,200,000' '27,100,000' '27,000,000' '26,900,000'
 '26,800,000' '26,700,000' '26,600,000' '26,500,000' '26,400,000'
 '26,300,000' '26,200,000' '26,100,000' '26,000,000' '25,900,000'
 '25,800,000' '25,700,000' '25,600,000' '25,500,000' '25,400,000'
 '25,300,000' '25,200,000' '25,100,000' '25,000,000' '24,900,000'
 '24,800,000' '24,700,000' '24,500,000' '24,400,000' '24,300,000'
 '24,200,000' '24,100,000' '24,000,000' '23,900,000' '23,800,000'
 '23,700,000' '23,600,000' '23,500,000' '23,400,000' '23,300,000'
 '23,200,000' '23,100,000' '23,000,000' '22,900,000' '22,800,000'
```

```
 '22,700,000' '22,600,000' '22,500,000' '22,400,000' '22,300,000'
 '22,200,000' '22,100,000' '22,000,000' '21,900,000' '21,800,000'
 '21,700,000' '21,600,000' '21,500,000' '21,400,000' '21,300,000'
 '21,200,000' '21,100,000' '21,000,000' '20,900,000' '20,800,000'
 '20,700,000' '20,600,000' '20,500,000' '20,400,000' '20,300,000'
 '20,200,000' '20,100,000' '20,000,000' '19,900,000' '19,800,000'
 '19,700,000' '19,600,000' '19,500,000' '19,400,000' '19,300,000'
 '19,200,000' '19,100,000' '19,000,000' '18,900,000' '18,800,000'
 '18,700,000' '18,600,000' '18,500,000' '18,400,000' '18,300,000'
 '18,200,000' '18,100,000' '18,000,000' '17,900,000' '17,800,000'
 '17,700,000' '17,600,000' '17,500,000' '17,400,000' '17,300,000'
 '17,200,000' '17,100,000' '17,000,000' '16,900,000' '16,800,000'
 '16,700,000' '16,600,000' '16,500,000' '16,400,000' '16,300,000'
 '16,200,000' '16,100,000' '16,000,000' '15,900,000' '15,800,000'
 '15,700,000' '15,600,000' '15,500,000' '15,400,000' '15,300,000'
 '15,200,000' '15,100,000' '15,000,000' '14,900,000' '14,800,000'
 '14,700,000' '14,600,000' '14,500,000' '14,400,000' '14,300,000'
 '14,200,000' '14,100,000' '14,000,000' '13,900,000' '13,800,000'
 '13,700,000' '13,600,000' '13,500,000' '13,400,000' '13,300,000'
 '13,200,000' '13,100,000' '13,000,000' '12,900,000' '12,800,000'
 '12,700,000' '12,600,000' '12,500,000' '12,400,000' '12,300,000'
 '12,200,000' '12,100,000' '12,000,000' '11,900,000' '11,800,000'
 '11,700,000' '11,600,000' '11,500,000' '11,400,000' '11,300,000'
 '11,200,000' '11,100,000' '11,000,000' '10,900,000' '10,800,000'
 '10,700,000' '10,600,000' '10,500,000' '10,400,000' '10,300,000'
 '10,200,000' '10,100,000' '10,000,000' '9,900,000' '9,800,000'
 '9,700,000' '9,400,000' '9,300,000' '9,200,000' '9,100,000'
'9,000,000'
 '8,900,000' '8,800,000' '8,700,000' '8,600,000' '8,500,000'
'8,400,000'
 '8,300,000' '8,200,000' '8,100,000' '8,000,000' '7,900,000'
'7,800,000'
 '7,700,000' '7,600,000' '7,500,000' '7,400,000' '7,300,000'
'7,200,000'
 '7,100,000' '7,000,000' '6,900,000' '6,800,000' '6,700,000'
'6,600,000'
 '6,500,000' '6,400,000' '6,300,000' '6,200,000' '6,100,000'
'6,000,000'
 '5,900,000' '5,800,000' '5,700,000' '5,600,000' '5,500,000'
'5,400,000'
 '5,300,000' '5,200,000' '5,100,000' '5,000,000' '4,900,000'
'4,800,000'
 '4,700,000' '4,600,000' '4,500,000' '4,400,000' '4,300,000'
'4,200,000'
 '4,100,000' '4,000,000' '3,900,000' '3,800,000' '3,700,000'
'3,600,000'
 '3,500,000' '3,400,000' '3,300,000' '3,200,000' '3,100,000'
'3,000,000'
 '2,900,000' '2,800,000' '2,700,000' '2,600,000' '2,500,000'
```

```
'2,400,000'
 '2,300,000' '2,200,000' '2,100,000' '2,000,000' '1,900,000'
'1,800,000'
 '1,700,000' '1,600,000' '1,500,000' '1,400,000' '1,300,000'
'1,200,000'
 '1,100,000' '1,000,000' '900,000' '800,000' '700,000' '600,000'
'500,000'
 '400,000' '300,000' '200,000' '100,000']
Watchtime in Million :-
 ['812.1M' '665.1M' '622.8M' '507.7M' '503.0M' '440.6M' '429.6M'
'402.5M'
 '302.1M' '266.2M' '262.6M' '252.5M' '251.5M' '249.9M' '235.0M'
'234.8M'
 '229.7M' '221.1M' '214.1M' '209.7M' '206.5M' '205.5M' '201.8M'
'200.7M'
 '194.7M' '192.9M' '184.0M' '182.3M' '181.8M' '176.8M' '175.5M'
'174.3M'
 '173.6M' '172.4M' '170.1M' '168.3M' '163.0M' '162.0M' '161.1M'
'157.6M'
 '155.3M' '153.9M' '153.0M' '152.1M' '151.5M' '151.4M' '150.2M'
'149.3M'
 '148.6M' '146.9M' '146.7M' '142.9M' '140.1M' '139.9M' '139.3M'
'136.8M'
 '136.6M' '136.2M' '135.9M' '134.8M' '133.6M' '133.5M' '133.4M'
'132.1M'
 '130.7M' '129.2M' '129.1M' '126.4M' '124.4M' '123.5M' '120.7M'
'120.5M'
 '120.3M' '120.0M' '118.9M' '118.6M' '116.5M' '116.2M' '115.8M'
'113.6M'
 '107.2M' '107.0M' '106.6M' '104.6M' '104.5M' '104.3M' '102.8M'
'102.3M'
 '101.7M' '99.9M' '99.5M' '99.0M' '98.5M' '97.8M' '97.6M' '96.4M'
'95.8M'
 '95.7M' '95.1M' '95.0M' '94.7M' '94.6M' '94.4M' '94.3M' '94.2M'
'92.9M'
 '92.5M' '92.3M' '92.2M' '91.4M' '91.2M' '90.8M' '90.2M' '89.0M'
'88.6M'
 '87.9M' '87.3M' '87.2M' '86.2M' '86.1M' '86.0M' '85.4M' '85.0M'
'84.6M'
 '84.4M' '83.6M' '83.2M' '82.8M' '82.5M' '82.4M' '82.1M' '81.8M'
'81.7M'
 '81.0M' '80.8M' '80.5M' '80.3M' '80.0M' '79.7M' '78.2M' '77.8M'
'77.2M'
 '76.6M' '76.3M' '75.7M' '75.2M' '75.1M' '74.3M' '73.4M' '73.3M'
'73.1M'
 '72.8M' '72.2M' '71.6M' '71.3M' '71.1M' '71.0M' '70.6M' '69.9M'
'69.8M'
 '69.7M' '69.5M' '69.2M' '69.0M' '68.9M' '68.5M' '68.2M' '68.1M'
'67.8M'
```

```
 '67.7M' '67.5M' '67.2M' '67.1M' '67.0M' '66.7M' '66.5M' '66.0M'
'65.9M'
 '65.3M' '65.2M' '64.4M' '64.3M' '64.2M' '63.9M' '63.7M' '63.5M'
'63.1M'
 '62.8M' '62.7M' '62.3M' '62.0M' '61.8M' '61.6M' '61.5M' '61.3M'
'61.1M'
 '61.0M' '60.8M' '60.6M' '60.1M' '59.9M' '59.8M' '59.6M' '59.3M'
'58.5M'
 '58.3M' '57.9M' '57.8M' '57.5M' '57.4M' '57.0M' '56.7M' '56.6M'
'56.5M'
 '56.4M' '56.3M' '56.0M' '55.7M' '55.5M' '55.2M' '55.1M' '55.0M'
'54.8M'
 '54.4M' '53.8M' '53.7M' '53.6M' '53.5M' '53.3M' '53.1M' '53.0M'
'52.8M'
 '52.5M' '52.4M' '52.3M' '52.2M' '52.0M' '51.8M' '51.7M' '51.6M'
'51.3M'
 '51.2M' '51.0M' '50.9M' '50.8M' '50.6M' '50.4M' '50.3M' '50.1M'
'50.0M'
 '49.7M' '49.6M' '49.4M' '49.3M' '49.2M' '48.9M' '48.8M' '48.6M'
'48.5M'
 '48.4M' '48.3M' '48.2M' '48.1M' '47.9M' '47.8M' '47.5M' '47.1M'
'46.9M'
 '46.6M' '46.5M' '46.4M' '46.3M' '46.2M' '46.1M' '46.0M' '45.9M'
'45.8M'
 '45.7M' '45.6M' '45.5M' '45.4M' '45.3M' '45.2M' '45.1M' '44.9M'
'44.8M'
 '44.7M' '44.6M' '44.3M' '44.2M' '44.0M' '43.9M' '43.8M' '43.7M'
'43.6M'
 '43.5M' '43.4M' '43.3M' '43.2M' '43.1M' '42.9M' '42.8M' '42.7M'
'42.6M'
 '42.5M' '42.4M' '42.3M' '42.1M' '41.8M' '41.7M' '41.5M' '41.4M'
'41.3M'
 '41.2M' '41.1M' '41.0M' '40.9M' '40.6M' '40.5M' '40.4M' '40.3M'
'40.2M'
 '40.1M' '40.0M' '39.9M' '39.8M' '39.7M' '39.5M' '39.4M' '39.3M'
'39.2M'
 '39.1M' '38.9M' '38.7M' '38.6M' '38.4M' '38.3M' '38.2M' '38.1M'
'38.0M'
 '37.8M' '37.7M' '37.6M' '37.5M' '37.4M' '37.3M' '37.1M' '37.0M'
'36.9M'
 '36.7M' '36.6M' '36.5M' '36.4M' '36.3M' '36.2M' '36.1M' '36.0M'
'35.9M'
 '35.8M' '35.7M' '35.6M' '35.5M' '35.4M' '35.3M' '35.2M' '35.0M'
'34.9M'
 '34.8M' '34.6M' '34.5M' '34.4M' '34.3M' '34.2M' '34.1M' '34.0M'
'33.9M'
 '33.8M' '33.7M' '33.5M' '33.4M' '33.3M' '33.2M' '33.1M' '33.0M'
'32.9M'
 '32.8M' '32.7M' '32.6M' '32.5M' '32.4M' '32.3M' '32.2M' '32.1M'
```

```
'32.0M'
 '31.9M' '31.8M' '31.7M' '31.6M' '31.5M' '31.4M' '31.3M' '31.2M'
'31.1M'
 '31.0M' '30.9M' '30.8M' '30.7M' '30.6M' '30.5M' '30.4M' '30.3M'
'30.2M'
 '30.1M' '30.0M' '29.9M' '29.8M' '29.7M' '29.6M' '29.5M' '29.4M'
'29.3M'
 '29.2M' '29.1M' '29.0M' '28.9M' '28.8M' '28.7M' '28.6M' '28.5M'
'28.4M'
 '28.3M' '28.2M' '28.1M' '28.0M' '27.9M' '27.8M' '27.7M' '27.6M'
'27.5M'
 '27.4M' '27.3M' '27.2M' '27.1M' '27.0M' '26.9M' '26.8M' '26.7M'
'26.6M'
 '26.5M' '26.4M' '26.3M' '26.2M' '26.1M' '26.0M' '25.9M' '25.8M'
'25.7M'
 '25.6M' '25.5M' '25.4M' '25.3M' '25.2M' '25.1M' '25.0M' '24.9M'
'24.8M'
 '24.7M' '24.5M' '24.4M' '24.3M' '24.2M' '24.1M' '24.0M' '23.9M'
'23.8M'
 '23.7M' '23.6M' '23.5M' '23.4M' '23.3M' '23.2M' '23.1M' '23.0M'
'22.9M'
 '22.8M' '22.7M' '22.6M' '22.5M' '22.4M' '22.3M' '22.2M' '22.1M'
'22.0M'
 '21.9M' '21.8M' '21.7M' '21.6M' '21.5M' '21.4M' '21.3M' '21.2M'
'21.1M'
 '21.0M' '20.9M' '20.8M' '20.7M' '20.6M' '20.5M' '20.4M' '20.3M'
'20.2M'
 '20.1M' '20.0M' '19.9M' '19.8M' '19.7M' '19.6M' '19.5M' '19.4M'
'19.3M'
 '19.2M' '19.1M' '19.0M' '18.9M' '18.8M' '18.7M' '18.6M' '18.5M'
'18.4M'
 '18.3M' '18.2M' '18.1M' '18.0M' '17.9M' '17.8M' '17.7M' '17.6M'
'17.5M'
 '17.4M' '17.3M' '17.2M' '17.1M' '17.0M' '16.9M' '16.8M' '16.7M'
'16.6M'
 '16.5M' '16.4M' '16.3M' '16.2M' '16.1M' '16.0M' '15.9M' '15.8M'
'15.7M'
 '15.6M' '15.5M' '15.4M' '15.3M' '15.2M' '15.1M' '15.0M' '14.9M'
'14.8M'
 '14.7M' '14.6M' '14.5M' '14.4M' '14.3M' '14.2M' '14.1M' '14.0M'
'13.9M'
 '13.8M' '13.7M' '13.6M' '13.5M' '13.4M' '13.3M' '13.2M' '13.1M'
'13.0M'
 '12.9M' '12.8M' '12.7M' '12.6M' '12.5M' '12.4M' '12.3M' '12.2M'
'12.1M'
 '12.0M' '11.9M' '11.8M' '11.7M' '11.6M' '11.5M' '11.4M' '11.3M'
'11.2M'
 '11.1M' '11.0M' '10.9M' '10.8M' '10.7M' '10.6M' '10.5M' '10.4M'
'10.3M'
```

```
 '10.2M' '10.1M' '10.0M' '9.9M' '9.8M' '9.7M' '9.4M' '9.3M' '9.2M'
'9.1M'
 '9.0M' '8.9M' '8.8M' '8.7M' '8.6M' '8.5M' '8.4M' '8.3M' '8.2M' '8.1M'
 '8.0M' '7.9M' '7.8M' '7.7M' '7.6M' '7.5M' '7.4M' '7.3M' '7.2M' '7.1M'
 '7.0M' '6.9M' '6.8M' '6.7M' '6.6M' '6.5M' '6.4M' '6.3M' '6.2M' '6.1M'
 '6.0M' '5.9M' '5.8M' '5.7M' '5.6M' '5.5M' '5.4M' '5.3M' '5.2M' '5.1M'
 '5.0M' '4.9M' '4.8M' '4.7M' '4.6M' '4.5M' '4.4M' '4.3M' '4.2M' '4.1M'
 '4.0M' '3.9M' '3.8M' '3.7M' '3.6M' '3.5M' '3.4M' '3.3M' '3.2M' '3.1M'
 '3.0M' '2.9M' '2.8M' '2.7M' '2.6M' '2.5M' '2.4M' '2.3M' '2.2M' '2.1M'
 '2.0M' '1.9M' '1.8M' '1.7M' '1.6M' '1.5M' '1.4M' '1.3M' '1.2M' '1.1M'
 '1.0M' '0.9M' '0.8M' '0.7M' '0.6M' '0.5M' '0.4M' '0.3M' '0.2M'
'0.1M']
```

Handling Missing Value

Handling Genre Missing Data

```
dataset['Genre'].fillna("Nan", inplace=True)
```

Handling Premiere Missing Data

```
dataset.dropna(subset=["Premiere"],inplace=True)

dataset.isnull().sum()

Rank                    0
Title                   0
Type                    0
Premiere                0
Genre                   0
Watchtime               0
Watchtime in Million    0
dtype: int64
```

Data Cleaning

```
dataset['Watchtime']=pd.to_numeric(dataset['Watchtime'].str.replace(",
",""), errors="coerce")
dataset.head()

   Rank                                Title     Type  Premiere
Genre  \
0   1.0                      The Night Agent  TV Show    2023.0
Action
1   2.0                      Ginny & Georgia  TV Show    2021.0
Drama
2   3.0                            The Glory  TV Show    2022.0
Thriller
3   4.0                            Wednesday  TV Show    2022.0
```

```
Fantasy
4   5.0  Queen Charlotte: A Bridgerton Story  TV Show    2023.0
Drama

   Watchtime Watchtime in Million
0  812100000                812.1M
1  665100000                665.1M
2  622800000                622.8M
3  507700000                507.7M
4  503000000                503.0M
```

This part filters the dataset to include only the rows where the stripped 'Genre' values are not equal to an empty string.

```
dataset=dataset[dataset["Genre"].str.strip()!=""]

#change the datatype of Premiere column float to integer
dataset['Premiere']=dataset['Premiere'].astype(int)

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18030 entries, 0 to 18163
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Rank                  18030 non-null  float64
 1   Title                 18030 non-null  object
 2   Type                  18030 non-null  object
 3   Premiere              18030 non-null  int64
 4   Genre                 18030 non-null  object
 5   Watchtime             18030 non-null  int64
 6   Watchtime in Million  18030 non-null  object
dtypes: float64(1), int64(2), object(4)
memory usage: 1.1+ MB
```

```
#Check for duplicates "Title"
column_name="Title"
duplicates=dataset.duplicated(subset=[column_name], keep=False)
duplicates_rows=dataset[duplicates].sort_values(by=column_name)
num_duplicates=duplicates_rows.shape[0]
print("Number of Duplicates",num_duplicates)
```

```
Number of Duplicates 5259
```

```
duplicates_rows.head(10)
```

```
          Rank                               Title     Type  Premiere
\
1569    1570.0           100 Dias Para Enamorarnos   TV Show      2020
```

```
7230    7281.0          100 Dias Para Enamorarnos  TV Show      2020

1008    1009.0                      13 Reasons Why  TV Show      2017

1297    1298.0                      13 Reasons Why  TV Show      2017

584      585.0                      13 Reasons Why  TV Show      2017

2058    2059.0                      13 Reasons Why  TV Show      2017

16518  16569.0  13 Reasons Why: Beyond the Reasons  TV Show      2017

16439  16490.0  13 Reasons Why: Beyond the Reasons  TV Show      2017

6774    6825.0                               19-2  TV Show      2014

6769    6820.0                               19-2  TV Show      2014


             Genre  Watchtime Watchtime in Million
1569        Comedy   13400000                13.4M
7230        Comedy    1300000                 1.3M
1008         Drama   21100000                21.1M
1297         Drama   16500000                16.5M
584          Drama   31700000                31.7M
2058         Drama    9900000                 9.9M
16518  Documentary     100000                 0.1M
16439  Documentary     100000                 0.1M
6774         Crime    1500000                 1.5M
6769         Crime    1500000                 1.5M
```

Exploratory Data Analysis

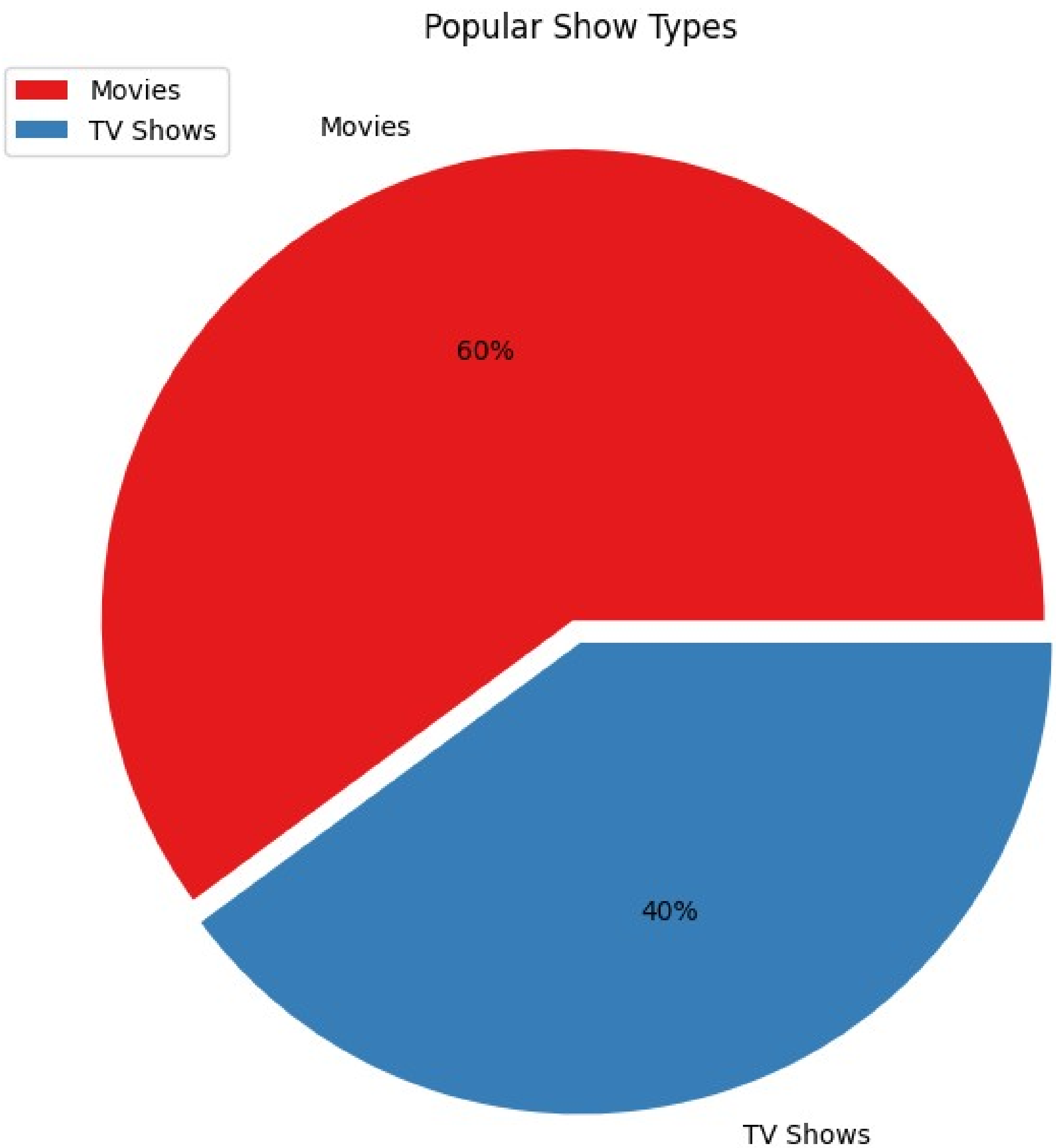
```
dataset.Type.value_counts()

Type
Movie      10837
TV Show     7193
Name: count, dtype: int64
```

Finding the most common types in the dataset

```
type_counts=dataset.Type.value_counts()
values=type_counts*100/len(dataset)
labels=["Movies", "TV Shows"]
explode=[0.05,0]
pallete_color=sns.color_palette("Set1")
plt.figure(figsize=(10,8))
plt.pie(values, labels=labels, colors=pallete_color, explode=explode,
autopct="%0.0f%%")
```

```
plt.title("Popular Show Types")
plt.legend(loc='upper left')
```

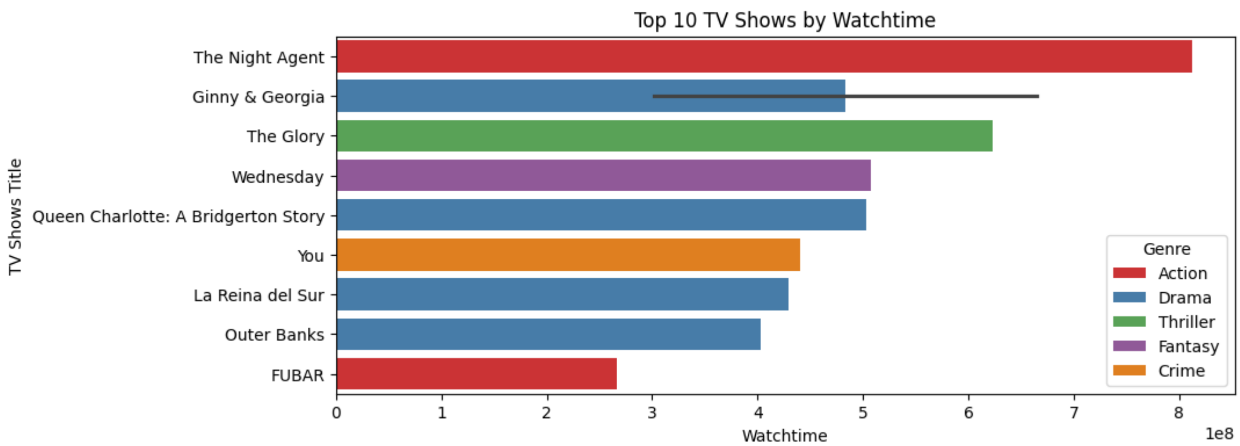
```
<matplotlib.legend.Legend at 0x13f4e9c90>
```



TV Shows Watchtime Analysis

```
sorted_tv_shows=dataset[dataset["Type"]=='TV
Show'].sort_values(by="Watchtime",ascending=False)
top_10_tv_shows=sorted_tv_shows.head(10)
```

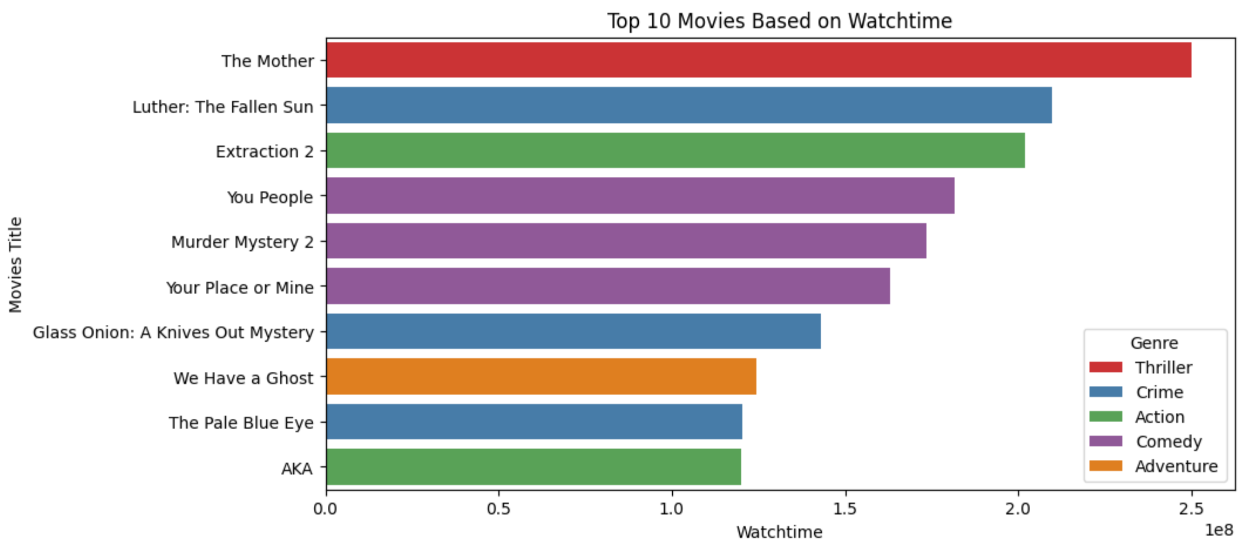
```
#create horizontal bar plot using seaborn and matplotlib
plt.figure(figsize=(10,4))
sns.barplot(data=top_10_tv_shows, x="Watchtime", y="Title",
hue="Genre", palette="Set1")
plt.xlabel("Watchtime")
plt.ylabel("TV Shows Title")
plt.legend(loc="lower right", title="Genre")
plt.title("Top 10 TV Shows by Watchtime")
plt.show()
```



Analysis by Watchtime for TV Shows When examining the watchtime data for TV Shows, we observe significant viewership for the top performers. Include- The Night Agent: Accumulating an impressive watchtime of 8M. Ginny & Georgia: Garnering a substantial watchtime of 6. M The Glory: Attracting viewers with a watchtime of 6.2M Wednesday: Captivating audiences with a watchtime of 5M. Queen Charlotte: A Bridgerton Story: Amassing a significant watchtime of 5M

Movies Watchtime Analysis

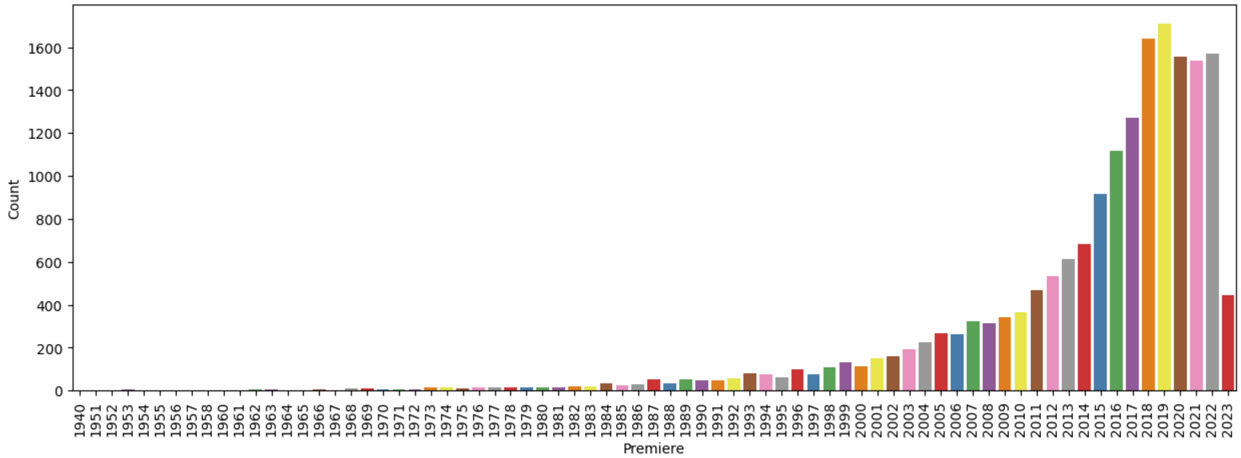
```
sorted_movies=dataset[dataset["Type"]=="Movie"].sort_values(by="Watcht
ime", ascending=False)
top_10_movies=sorted_movies.head(10)

#create horizontal bar plot using seaborn and %matplotlib

plt.figure(figsize=(10,5))
sns.barplot(data=top_10_movies, x="Watchtime",
y="Title",palette="Set1", hue="Genre")
plt.xlabel("Watchtime")
plt.ylabel("Movies Title")
plt.legend(loc="lower right", title="Genre")
plt.title("Top 10 Movies Based on Watchtime")
plt.show()
```



Similarly, Top 5 movies by watchtime engagement levels: The Mother: Garnering considerable attention with a watchtime of 2.4M. Luther: The Fallen Sun: Captivating viewers with a watchtime of 2.09M Extraction 2: Attracting substantial viewership with a watchtime of 2.01M You People: Engaging audiences with a watchtime of 1.8M. Murder Mystery 2: Enticing viewers with a watchtime of 1.7M

Values counts for premiere

```
dataset.Premiere.value_counts()

Premiere
2019    1712
2018    1641
2022    1569
2020    1554
2021    1536
        ...
1965       1
1951       1
1955       1
1967       1
1952       1
Name: count, Length: 73, dtype: int64

plt.figure(figsize=(15,5))
sns.countplot(data=dataset, x="Premiere", hue="Premiere",
palette="Set1", legend=False)
plt.xlabel("Premiere")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()
```



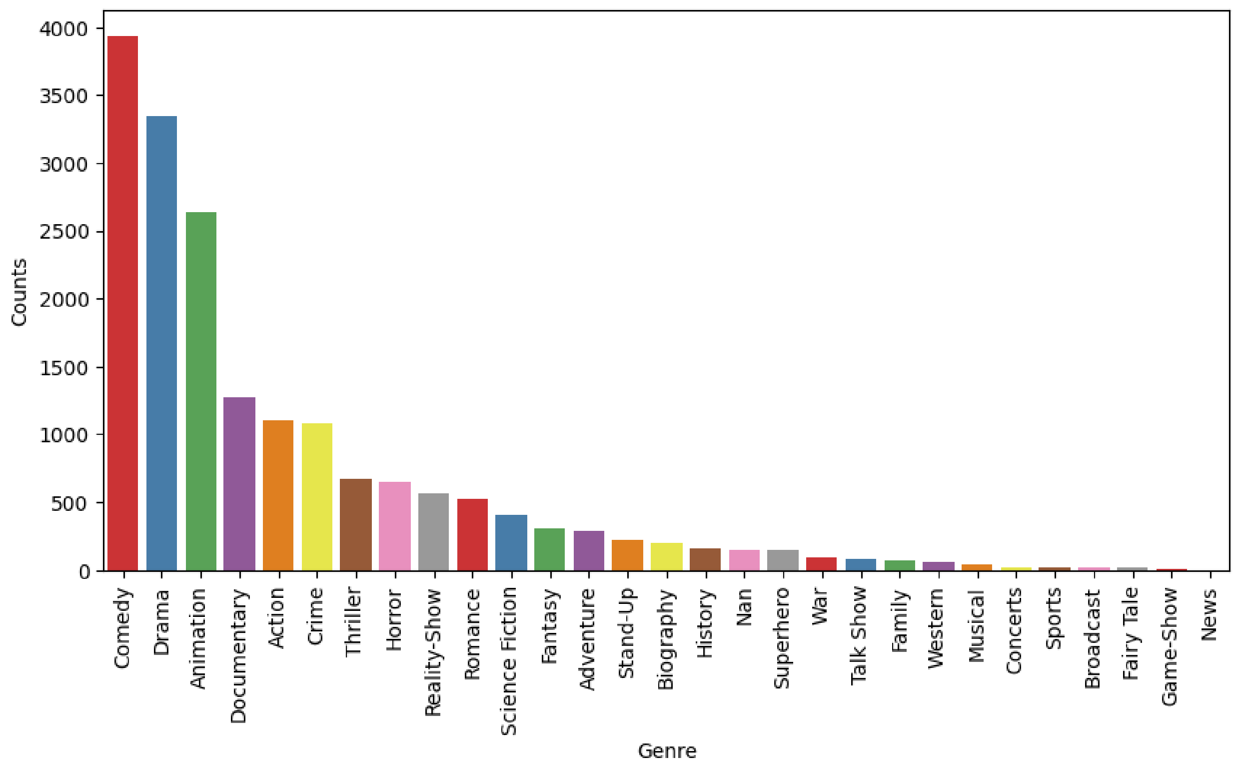
As we can see, 2019 had more premieres than others years. From 2015, the number of premieres has increased compared to previous years.

Values counts for Genre

```
dataset.Genre.value_counts()

Genre
Comedy              3933
Drama               3340
Animation           2638
Documentary         1275
Action              1099
Crime               1080
Thriller             674
Horror               654
Reality-Show         562
Romance              519
Science Fiction      398
Fantasy              303
Adventure            281
Stand-Up             224
Biography            198
History              154
Nan                  151
Superhero            143
War                   90
Talk Show             81
Family                71
Western               55
Musical               39
Concerts              18
Sports                17
Broadcast             12
Fairy Tale            12
Game-Show              8
```

```
News                        1
Name: count, dtype: int64
```

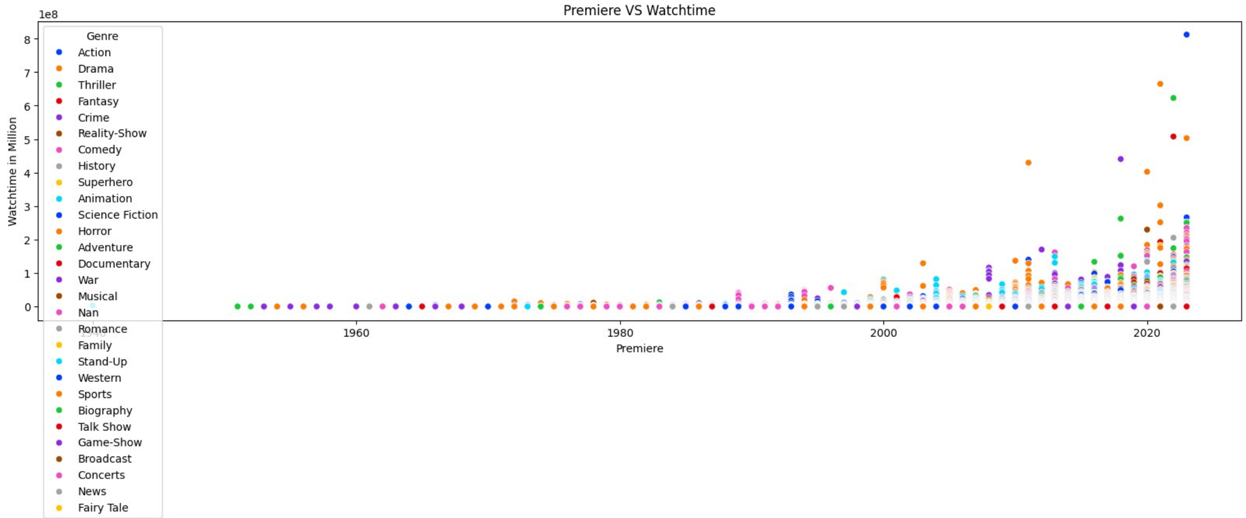
```
plt.figure(figsize=(10,5))
sns.countplot(data=dataset, x="Genre", palette="Set1", legend=False,
order=dataset.Genre.value_counts().index)
plt.xlabel("Genre")
plt.ylabel("Counts")
plt.xticks(rotation=90)
plt.show()
```



People were more interested in watching comedy, drama, animation, action, crime, and thriller genres

Premiere Year Vs Watchtime

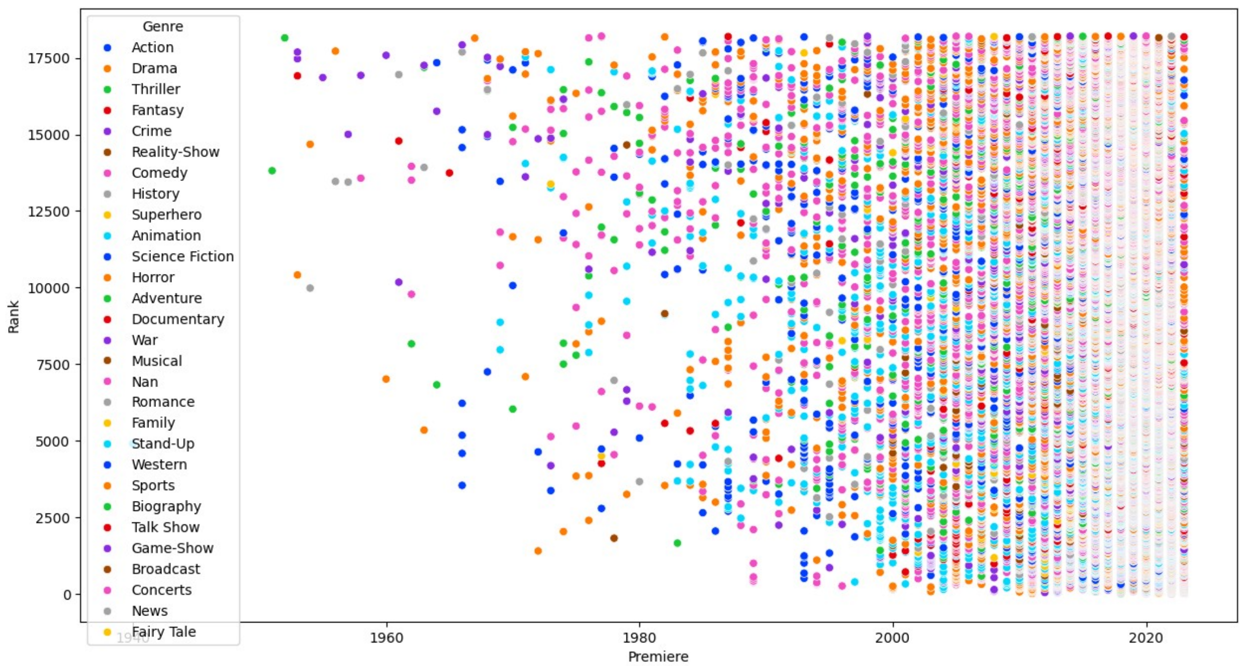
```
plt.figure(figsize=(20,5))
sns.scatterplot(data=dataset, x="Premiere", y="Watchtime",
palette="bright", hue="Genre")
plt.title("Premiere VS Watchtime")
plt.xlabel("Premiere")
plt.ylabel("Watchtime in Million")
plt.show()
```



There appears to be a weak positive correlation between watchtime and premiere year. This means that movies released in recent years(2015-2023) tend to have higher watchtimes. This could be due to a number of factors, such as changes in viewing habits (e.g., the rise of streaming services) or the fact that more recent movies are simply more popular.

Premiere Year Vs Rank

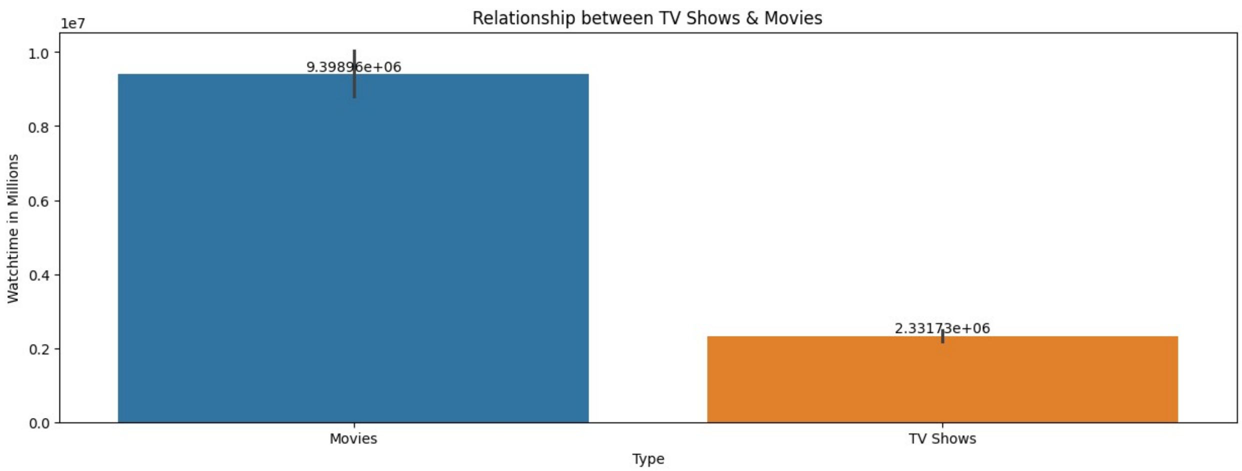
```
plt.figure(figsize=(15,8))
sns.scatterplot(data=dataset, x="Premiere", y="Rank",
palette="bright", hue="Genre")
plt.xlabel("Premiere")
plt.ylabel("Rank")
plt.show()
```



Weak Positive Correlation: There appears to be a weak positive correlation between watchtime and rank Shows and movies released after 2000 and with longer watch times tend to have higher ranks in the dataset.

Relationship Between Watchtime and Type

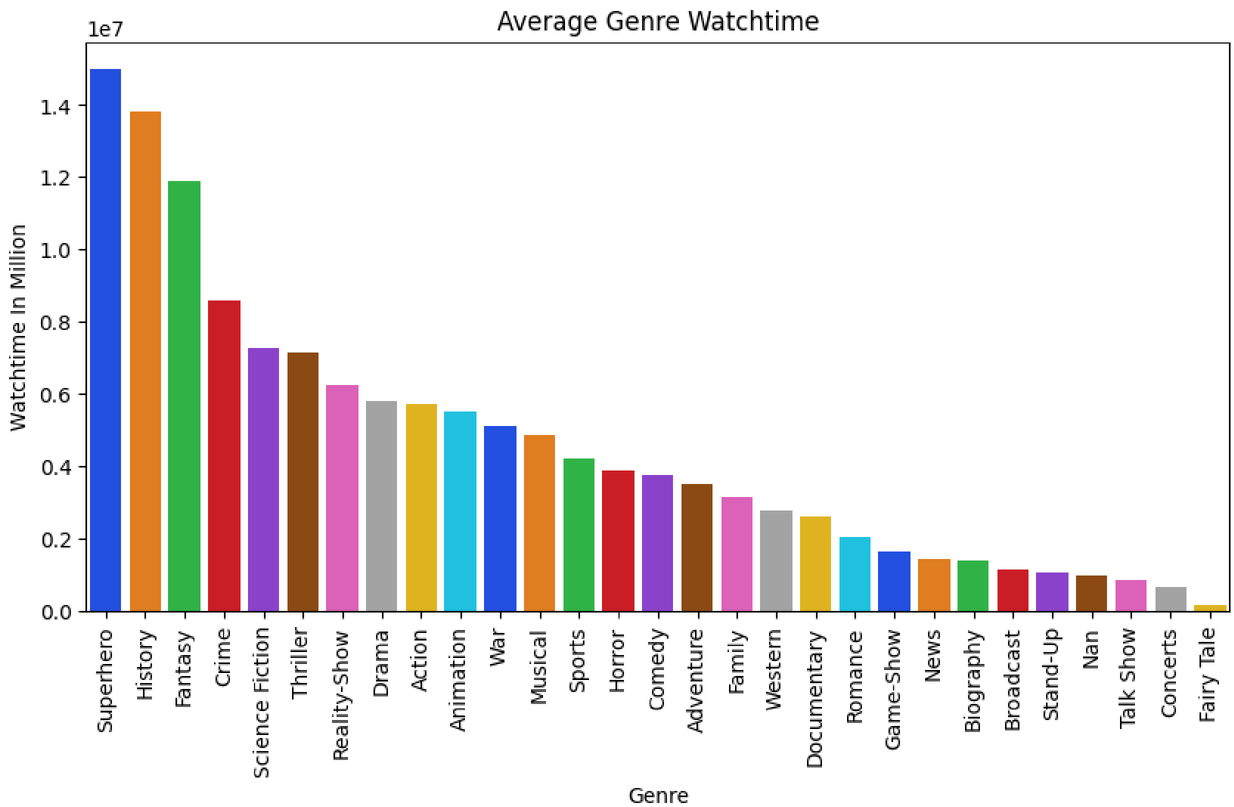
```
plt.figure(figsize=(15,5))
bar_plot=sns.barplot(data=dataset, x="Type", y="Watchtime",
hue="Type", legend=False)
for bar in bar_plot.containers:
    bar_plot.bar_label(bar)
plt.title("Relationship between TV Shows & Movies")
plt.xlabel("Type")
plt.ylabel("Watchtime in Millions")
plt.xticks(ticks=[0,1], labels=["Movies", "TV Shows"])
plt.show()
```



On average, Tv shows have lower watchtime compared to movies Viewers spend more time on watching Movies than TV Shows

Average Genre Watchtime

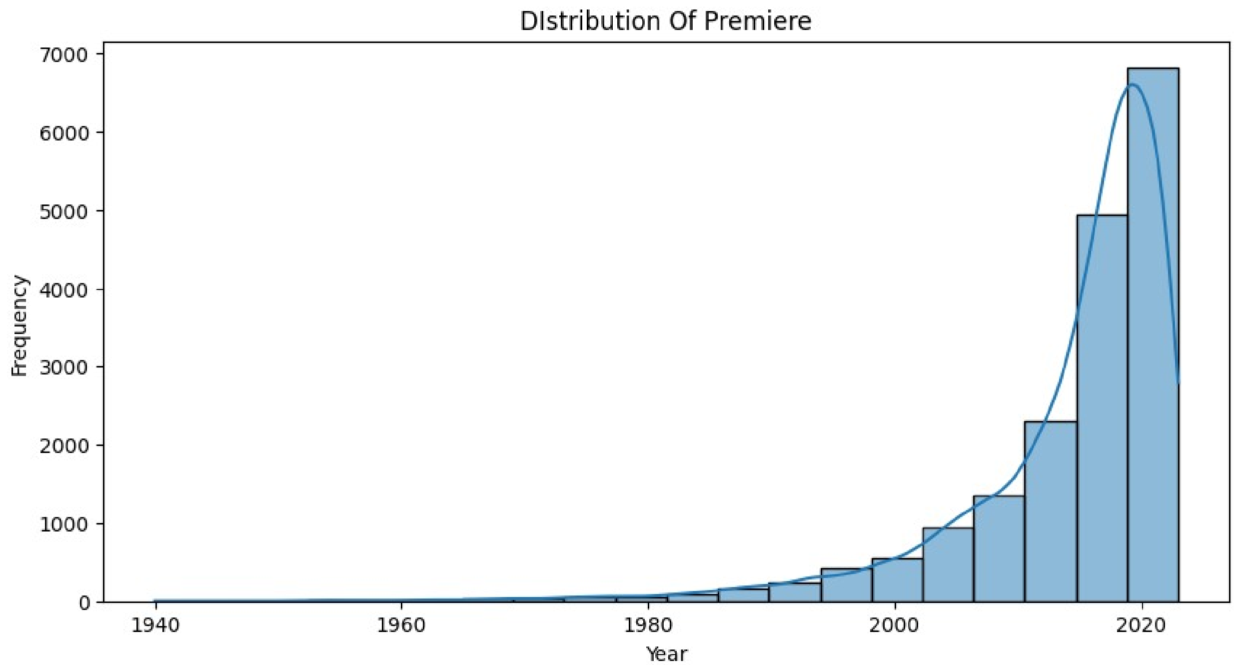
```
avg_genre=dataset.groupby("Genre")["Watchtime"].mean().reset_index()
avg_genre_sorted=avg_genre.sort_values(by="Watchtime",
ascending=False)

plt.figure(figsize=(10,5))
sns.barplot(data=avg_genre_sorted, x="Genre", y="Watchtime",
palette="bright")
plt.xlabel("Genre")
plt.ylabel("Watchtime In Million")
plt.xticks(rotation=90)
plt.title("Average Genre Watchtime")
plt.show()
```



People like different types of shows, so they watch some more than others, which affects how long they spend watching them.

Distribution of Premieres Years

```
plt.figure(figsize=(10,5))
sns.histplot(dataset.Premiere,bins=20,kde=True)
plt.xlabel("Year")
plt.ylabel("Frequency")
plt.title("DIstribution Of Premiere")
plt.show()
```



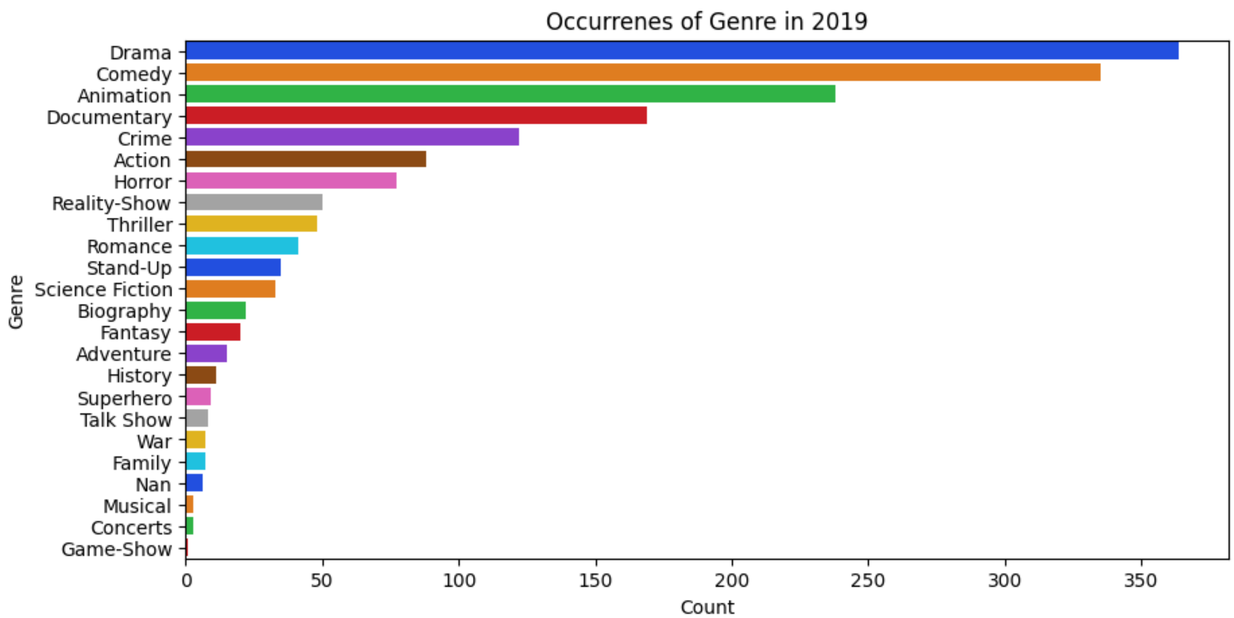
Most TV shows/Movies in the dataset premiered in recent years, with a peak around 2018-2023.

Exploratory Analysis on Most Productive Year/ Most Premiere Year

```
dataset_2019=dataset[dataset["Premiere"]==2019]
dataset_2019.head()

      Rank                           Title     Type  Premiere
Genre  \
72    73.0             Crash Landing on You  TV Show      2019
Comedy
97    98.0  Demon Slayer: Kimetsu no Yaiba  TV Show      2019
Animation
114  115.0     Formula 1: Drive to Survive  TV Show      2019
Documentary
117  118.0                   Murder Mystery    Movie      2019
Comedy
132  133.0                   Selling Sunset  TV Show      2019
Reality-Show

     Watchtime Watchtime in Million
72   120300000               120.3M
97    95800000                95.8M
114   90200000                90.2M
117   87900000                87.9M
132   82800000                82.8M
```

Occurrenes of Genres in 2019

```
genres_2019=dataset_2019["Genre"].value_counts().reset_index()
genres_2019.columns=["Genre", "Count"]
plt.figure(figsize=(10,5))
sns.barplot(data=genres_2019,x="Count", y="Genre", palette="bright")
plt.xlabel("Count")
plt.ylabel("Genre")
plt.title("Occurrenes of Genre in 2019")
plt.show()
```

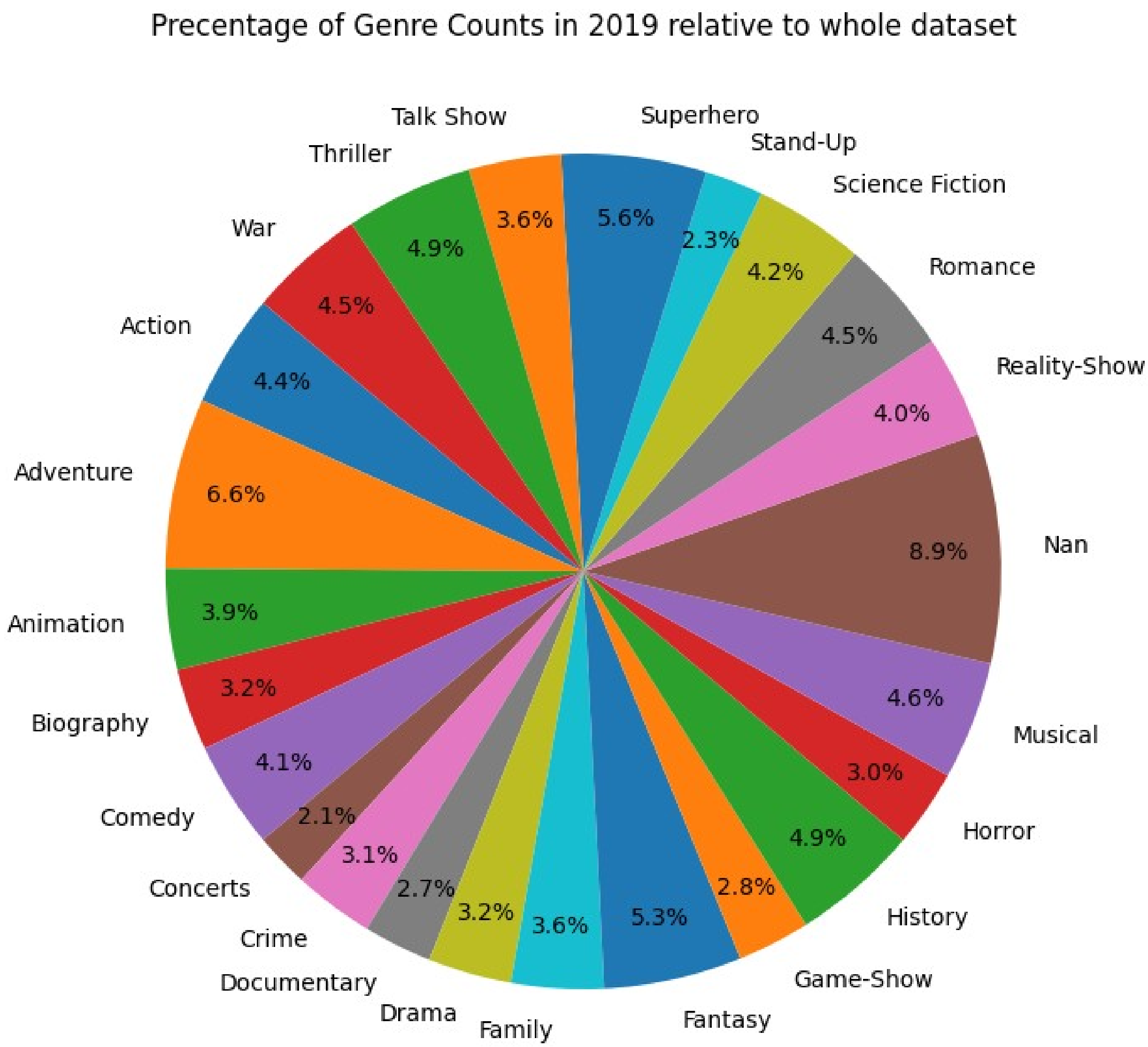


While top genres across the entire dataset and specifically in the year 2019 remain consistent as Comedy, Drama and Animation.

Precentage of Genre Counts in 2019 relative to Whole Dataset

```
#calculating total count of each genre in whole dataset

dataset_whole_count=dataset.Genre.value_counts()

#calculting total count of each genre in 2019 dataset

count_2019_genre=dataset_2019.Genre.value_counts()

#calculating percetage for each genre in 2019 relative to whole
dataset

genre_percentage_2019=(dataset_whole_count/count_2019_genre)*100
genre_percentage_2019=genre_percentage_2019.dropna()

#plot pie chart
plt.figure(figsize=(8,10))
plt.pie(genre_percentage_2019, labels=genre_percentage_2019.index,
```

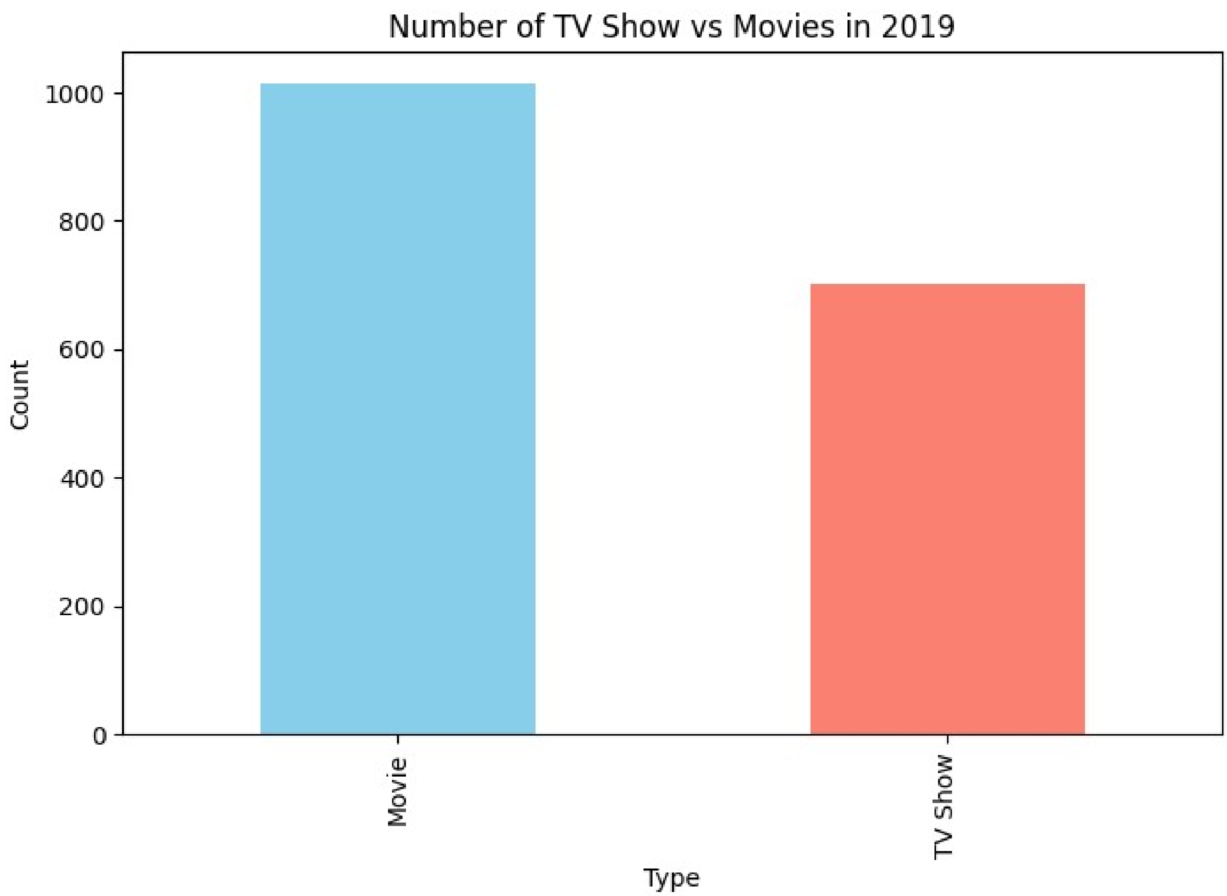
```
autopct="%1.1f%%", startangle=140, pctdistance=0.85)
plt.title("Precentage of Genre Counts in 2019 relative to whole
dataset")
plt.show()
```



Number of TV Show vs Movies in 2019

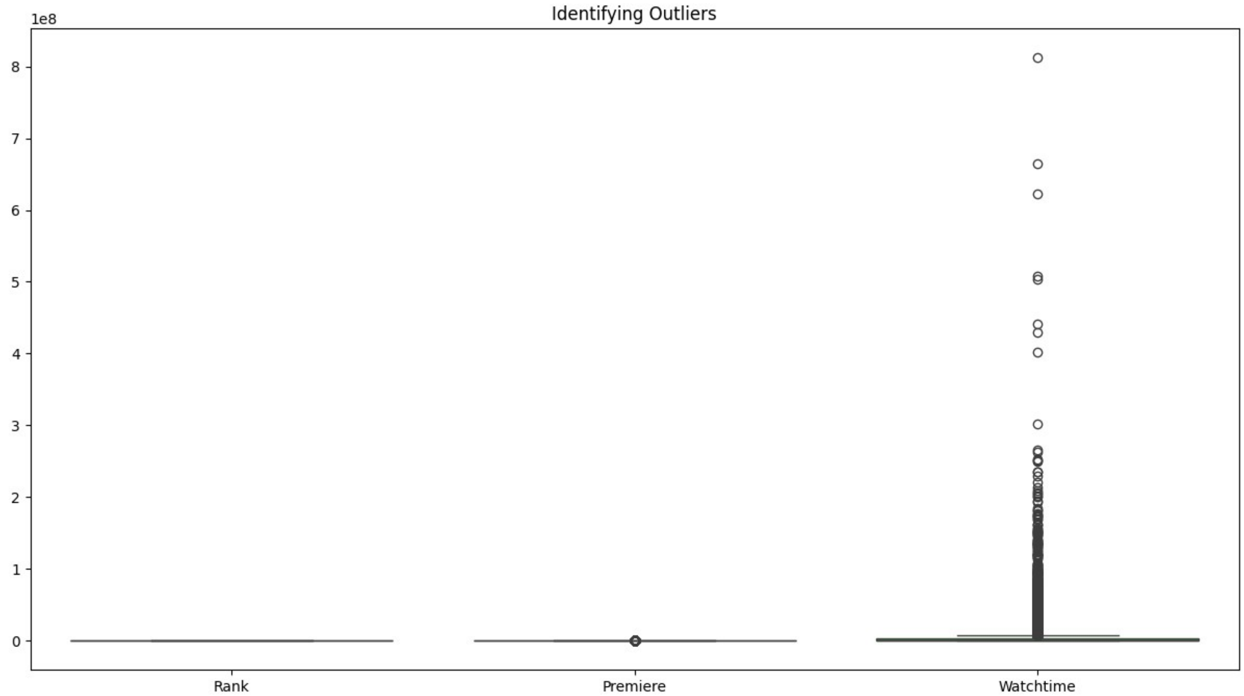
```
#calculte total counts of type
counts_type_2019=dataset_2019.Type.value_counts()
#plot in bar
plt.figure(figsize=(8,5))
# sns.barplot(data=counts_type_2019)# we can plot using sns also
counts_type_2019.plot(kind="bar", color=["skyblue", "salmon"])
```

```
plt.xlabel("Type")
plt.ylabel("Count")
plt.title("Number of TV Show vs Movies in 2019")
plt.show()
```



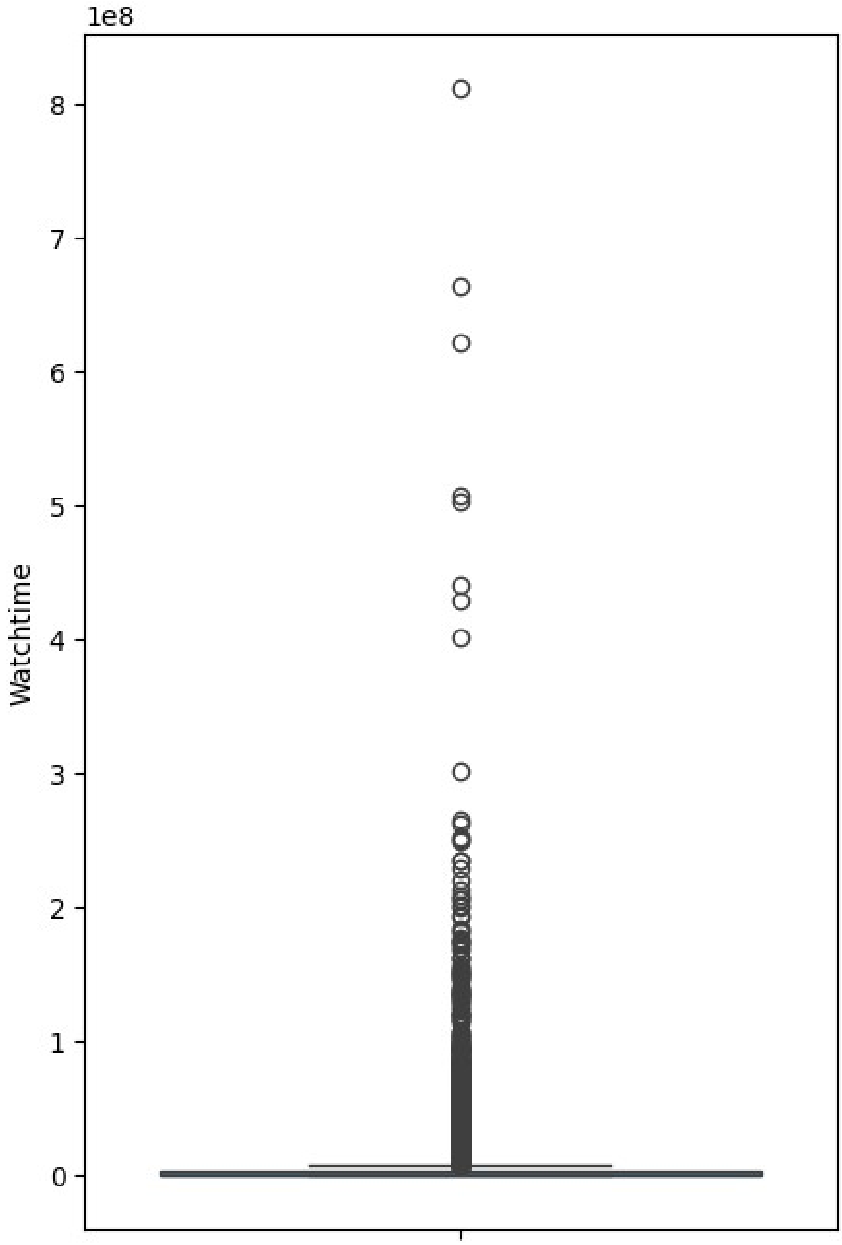
Identifying and removing Outliers using Boxplot

```
plt.figure(figsize=(15,8))
sns.boxplot(data=dataset)
plt.title("Identifying Outliers")
plt.show()
```



As we see, Watchtime has the most outliers

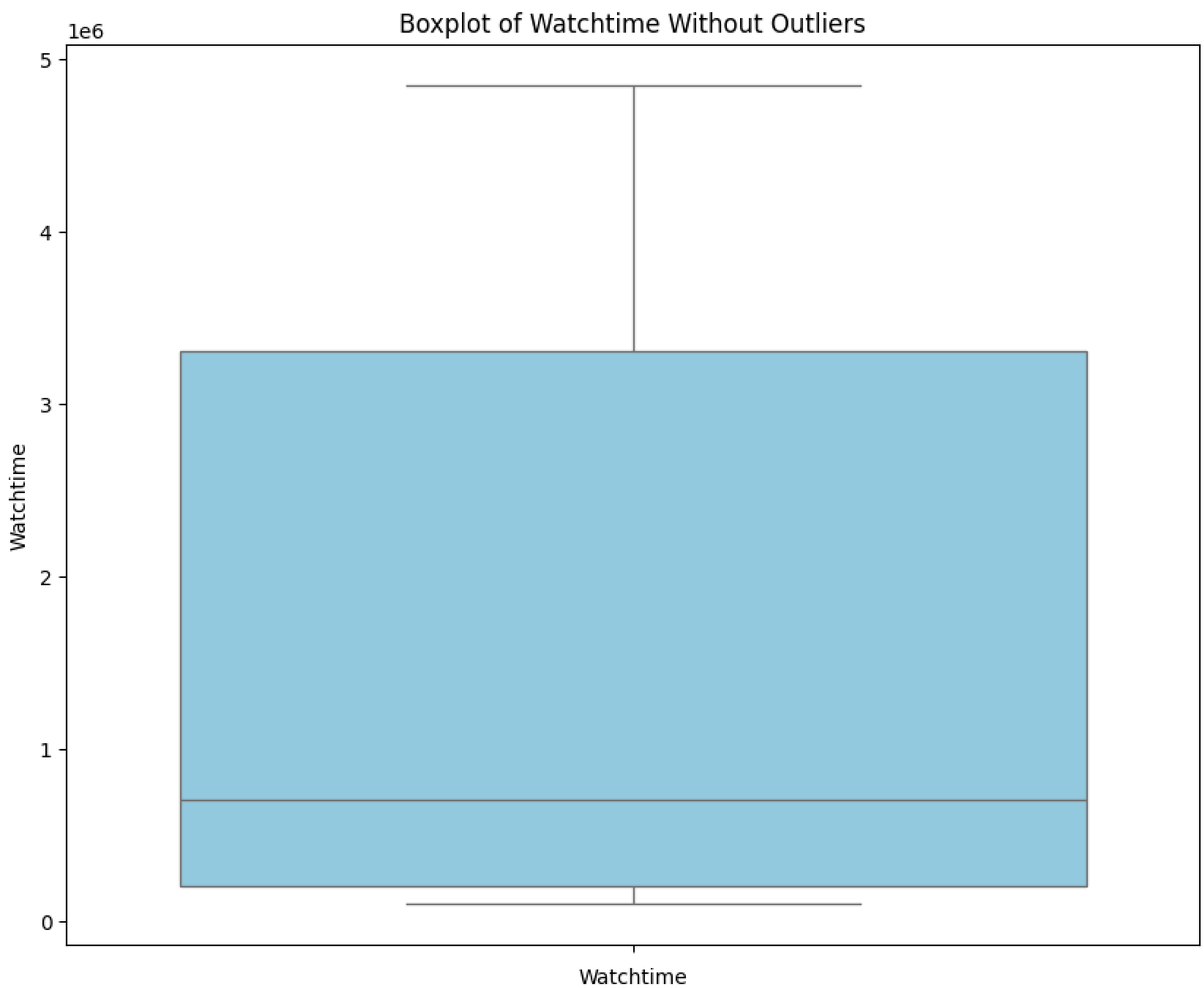
```
plt.figure(figsize=(5,8))
sns.boxplot(data=dataset, y="Watchtime")
plt.ylabel("Watchtime")
plt.show()
```



-The interquartile range (IQR), or the middle 50% of the data, for watchtime is between 1 million and 4 million. This means that half of the users have watched content on this dataset that falls within this range. -There are outliers at both ends of the watchtime spectrum. There seems to be a small number of users who have watched a very high amount of content (over 7 million watchtime) and a small number of users who have watched very little content (under 1 million watchtime). -It is difficult to say definitively what the most popular watchtime is on this dataset because the scale starts at 0 and goes to 100 million. However, we can say that the most common watchtime falls somewhere between 1 and 4 million watchtime.

```
q1=dataset.Watchtime.quantile(0.25)
q3=dataset.Watchtime.quantile(0.75)
```

```
IQR=q1-q3 #Interquantile Range

#calculate IQR in this way also
# q1,q3=np.percentile(dataset["Watchtime"],[25,75])
# IQR=q1-q3

#set lower and upper bound for outliers  (1.5 IQR for quartiles)
upper_bound=q3+1.5*IQR
lower_bound=q1-1.5*IQR

# Cap outliers to bounds using numpy.clip
dataset["Watchtime"]=np.clip(dataset["Watchtime"], lower_bound,
upper_bound)

plt.figure(figsize=(10,8))
sns.boxplot(data=dataset['Watchtime'], color="skyblue")
plt.title("Boxplot of Watchtime Without Outliers")
plt.xlabel("Watchtime")
plt.show()
```



Labeling Type and Genre Columns

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
dataset["Type"]=LE.fit_transform(dataset["Type"])
dataset["Genre"]=LE.fit_transform(dataset["Genre"])
```

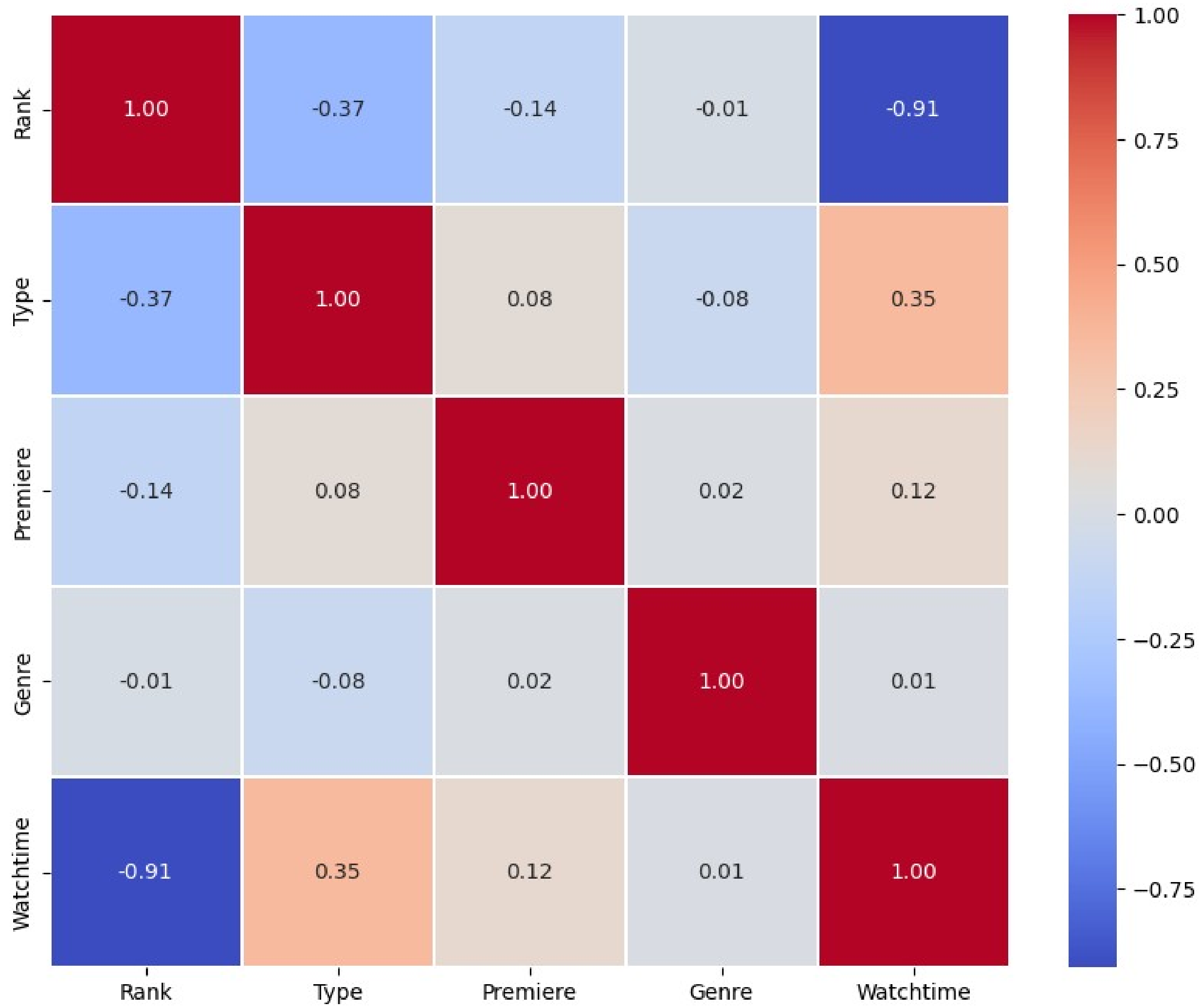
Co-relation and Heatmap

```
select_numeric_data=dataset.select_dtypes(include=("float64",
"int64"))
#calculate co-relation matrix
corealtion=select_numeric_data.corr()
corealtion
```

	Rank	Type	Premiere	Genre	Watchtime
Rank	1.000000	-0.374728	-0.136033	-0.005965	-0.907673
Type	-0.374728	1.000000	0.084599	-0.080830	0.345643
Premiere	-0.136033	0.084599	1.000000	0.023229	0.116940
Genre	-0.005965	-0.080830	0.023229	1.000000	0.011694
Watchtime	-0.907673	0.345643	0.116940	0.011694	1.000000

Create Heatmap

```
plt.figure(figsize=(10,8))
sns.heatmap(corealtion, annot=True, fmt=".2f",cmap="coolwarm",
linewidths=0.2)
plt.show()
```



Strong Positive Correlations:

Watchtime - Rank: There's a strong positive correlation (around 0.7) between watchtime and rank. This suggests that shows/movies with longer watchtimes tend to have higher ranks in the dataset. This aligns with the idea that viewers spend more time watching content they enjoy.

Moderate Positive Correlations:

Rank - Type: There's a moderate positive correlation (around 0.5) between rank and type. This could indicate that certain types of shows/movies (e.g., Drama, Action) are generally ranked higher than others. However, it's important to explore this further to see if the trend holds within each type.

Weak Correlations:

Premiere - Watchtime/Rank: The correlations between premiere year and watchtime/rank are weak (around -0.2 for both). This suggests that the release year doesn't have a strong influence on how long viewers watch a show/movie or how it's ranked.

Genre - Watchtime/Rank: The correlations between genre and watchtime/rank are also weak. This indicates that genre might not be a significant factor in determining watchtime or rank.

Other Observations:

There appears to be a weak negative correlation between watchtime and type (around -0.3). This is interesting and might require further investigation. It could be that certain long shows/movies (e.g., documentaries) tend to fall under specific types.

Overall Insights: Watchtime is the strongest indicator of rank in this dataset, suggesting viewers tend to spend more time watching higher-ranked shows/movies. The type of show/movie might also play a role in rank, but more investigation is needed to understand genre-specific trends. Premiere year and genre seem to have weaker influences on watchtime and rank based on the correlation matrix.

Feature Selection- Feature selection is the process in machine learning that involves choosing the most relevant features from the dataset. It plays a vital role in improving model performance by reducing overfitting, simplifying models, and enhancing interpretability. By selecting the most informative features, unnecessary noise and dimensionality are reduced, leading to more efficient and accurate predictions.

```
from sklearn.feature_selection import f_regression
from scipy.stats import f

X=dataset[["Watchtime", "Genre", "Type", "Premiere"]] #Features
Y=dataset[["Rank"]] #Target

#perform Annova Test

f_values, p_values=f_regression(X,Y)

#Check Significant Features Based on P_Values
significant_features=X.columns[p_values<0.05]
significant_features

Index(['Watchtime', 'Type', 'Premiere'], dtype='object')
```

Model Selection and Model Training

```
#split the data into tarining and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train,
Y_test=train_test_split(X[significant_features],Y,test_size=0.30,
random_state=42)
print(f" Train shape:{X_train.shape} and Test Shape:{X_test.shape}")

 Train shape:(12621, 3) and Test Shape:(5409, 3)

#Define Models
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor,
GradientBoostingRegressor
from sklearn.metrics import r2_score

models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor()
}

#Define hyperpyrameters of each models
parameters={
    "Linear Regression":{},
    "Decision Tree":{
        'max_depth':[3, 5, 7, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
    },

    "Random Forest": {
        'n_estimators': [50, 100],
        'max_depth': [3, 5, 7, 10],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
    },

    "Gradient Boosting": {
        'n_estimators': [50, 100],
        'learning_rate': [0.01, 0.1, 1],
        'max_depth': [3, 5, 7]
    }
}
```

Model Tunning and Model Evalution

```python
#Hyperparameter tuning and training
best_params={}
for model_name, model in models.items():
    print(f"Tunning hyperparameteres for {model_name}... ")
    grid_search=GridSearchCV(model, parameters[model_name],
cv=5,scoring='r2')
    grid_search.fit(X_train, Y_train)
    best_params[model_name]=grid_search.best_params_
    print("Best Parameters:",best_params[model_name])
```

```
Tunning hyperparameteres for Linear Regression...
Best Parameters: {}
```

```
Tunning hyperparameteres for Decision Tree...
Best Parameters: {'max_depth': 7, 'min_samples_leaf': 4,
'min_samples_split': 10}
Tunning hyperparameteres for Random Forest...
Best Parameters: {'max_depth': 7, 'min_samples_leaf': 1,
'min_samples_split': 5, 'n_estimators': 100}
Tunning hyperparameteres for Gradient Boosting...
Best Parameters: {'learning_rate': 0.1, 'max_depth': 3,
'n_estimators': 100}
```

Modelling with Linear Regression

```
from sklearn.linear_model import LinearRegression
model_1_lr=LinearRegression()
model_1_lr.fit(X_train,Y_train)
y_pred1=model_1_lr.predict(X_test)
print(y_pred1)
```

```
[[13349.35417274]
 [10944.85307441]
 [10635.34201324]
 ...
 [ 9737.18644579]
 [ 1582.05066405]
 [ 1524.3967821 ]]
```

Modelling with Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
model_2_dt=DecisionTreeRegressor(criterion="squared_error",
max_depth=7, min_samples_leaf=4, min_samples_split=10)
model_2_dt.fit(X_train,Y_train)
y_pred2=model_2_dt.predict(X_test)
print(y_pred2)
```

```
[16300.07526882  7696.96551724  7954.8        ...  7192.22807018
  2111.76684882  2111.76684882]
```

Modelling with Random Forest

```
from sklearn.ensemble import RandomForestRegressor
model_3_rf=RandomForestRegressor(max_depth=7, min_samples_leaf=1,
min_samples_split=2, n_estimators=100,bootstrap=True)
model_3_rf.fit(X_train, Y_train)
y_pred3=model_3_rf.predict(X_test)
print(y_pred3)
```

```
[16289.67544505  7693.96392669  7956.46373344 ...  7200.43394875
  2107.00712449  2085.32823692]
```

Modelling with AdaBoost

```
from sklearn.ensemble import AdaBoostRegressor
model_4_ada=AdaBoostRegressor(n_estimators=50, random_state=42,
learning_rate=1)
model_4_ada.fit(X_train,Y_train)
y_pred4=model_4_ada.predict(X_test)
print(y_pred4)
```

```
[16304.41986755  7779.40798226  7779.40798226 ...  7779.40798226
  1832.12357955  1832.12357955]
```

Modelling with Gradient Boosting

```
from sklearn.ensemble import GradientBoostingRegressor
model_5_gb=GradientBoostingRegressor(learning_rate=0.1, max_depth=3,
n_estimators=100, random_state=42)
model_5_gb.fit(X_train, Y_train)
y_pred5=model_5_gb.predict(X_test)
print(y_pred5)
```

```
[16312.4077996   7713.06472339  7953.19751506 ...  7135.18581102
  2052.72679766  2113.87608126]
```

```
model=("Linear Regression", "Decision Tree Regression", "Random Forest
Regressor", "AdaBoost Regressor", "Gradient Boost Regressor")
models=[LinearRegression, DecisionTreeRegressor,
RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor]

from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error, mean_absolute_percentage_error
last_model=[model_1_lr, model_2_dt, model_3_rf, model_4_ada,
model_5_gb]
r2_list=[]

for i in last_model:
    print("Model Name is:- ", i)
    i.fit(X_train, Y_train)
    y_pred=i.predict(X_test)
    r2=r2_score(Y_test, y_pred)
    r2_list.append(r2)
    print("R2 Score:-",r2_score(Y_test, y_pred))
    print("MAE Score:-", mean_absolute_error(Y_test, y_pred))
    print("MSE Score:-", mean_squared_error(Y_test, y_pred))
    print("MAPE Score:-", mean_absolute_percentage_error(Y_test,
y_pred))
    print("="*20)
```

```
Model Name is:-  LinearRegression()
R2 Score:- 0.8252251861924147
```

```
MAE Score:- 1838.8470246216764
MSE Score:- 4743878.805321169
MAPE Score:- 0.6146084098986512
====================
Model Name is:-  DecisionTreeRegressor(max_depth=7,
min_samples_leaf=4, min_samples_split=10)
R2 Score:- 0.9812997752686046
MAE Score:- 477.53946012857347
MSE Score:- 507576.565669667
MAPE Score:- 0.800141922475259
====================
Model Name is:-  RandomForestRegressor(max_depth=7)
R2 Score:- 0.98139574258343
MAE Score:- 475.7901511550486
MSE Score:- 504971.74349371495
MAPE Score:- 0.7968911553292825
====================
Model Name is:-  AdaBoostRegressor(learning_rate=1, random_state=42)
R2 Score:- 0.9723549335245445
MAE Score:- 720.8416513531167
MSE Score:- 750364.6667819631
MAPE Score:- 1.040077735728387
====================
Model Name is:-  GradientBoostingRegressor(random_state=42)
R2 Score:- 0.9814134262407102
MAE Score:- 480.23992091671465
MSE Score:- 504491.7594208119
MAPE Score:- 0.8151141432283489
====================
```

```
r2_list
```

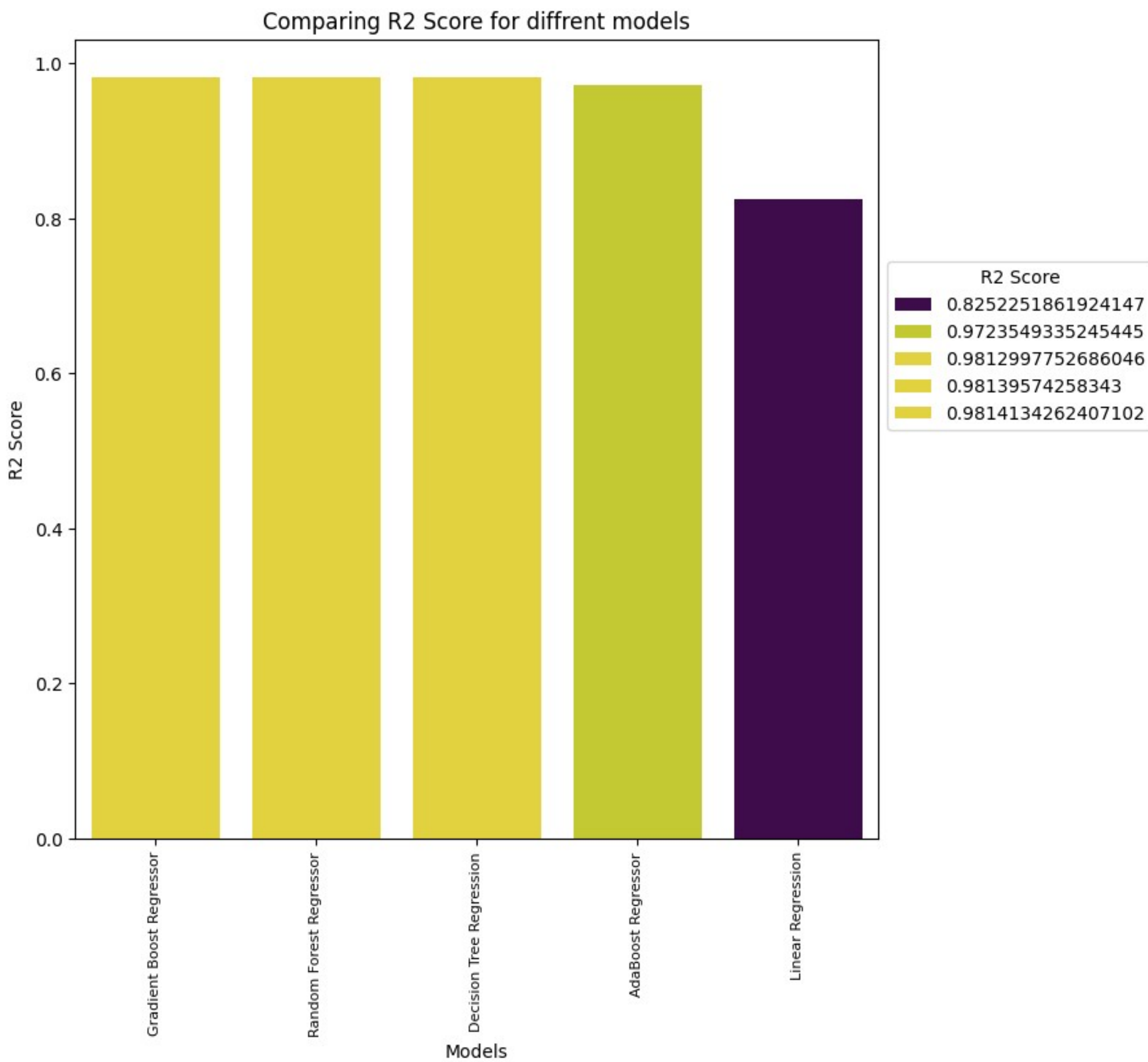
```
[0.8252251861924147,
 0.9812997752686046,
 0.98139574258343,
 0.9723549335245445,
 0.9814134262407102]
```

```
model_best=pd.DataFrame({"model":model, "R2 Score":r2_list})
best_model=model_best.sort_values(by="R2 Score", ascending=False)
best_model
```

```
                      model  R2 Score
4  Gradient Boost Regressor  0.981413
2   Random Forest Regressor  0.981396
1  Decision Tree Regression  0.981300
3        AdaBoost Regressor  0.972355
0         Linear Regression  0.825225
```

Comparing R2 Score Diagram

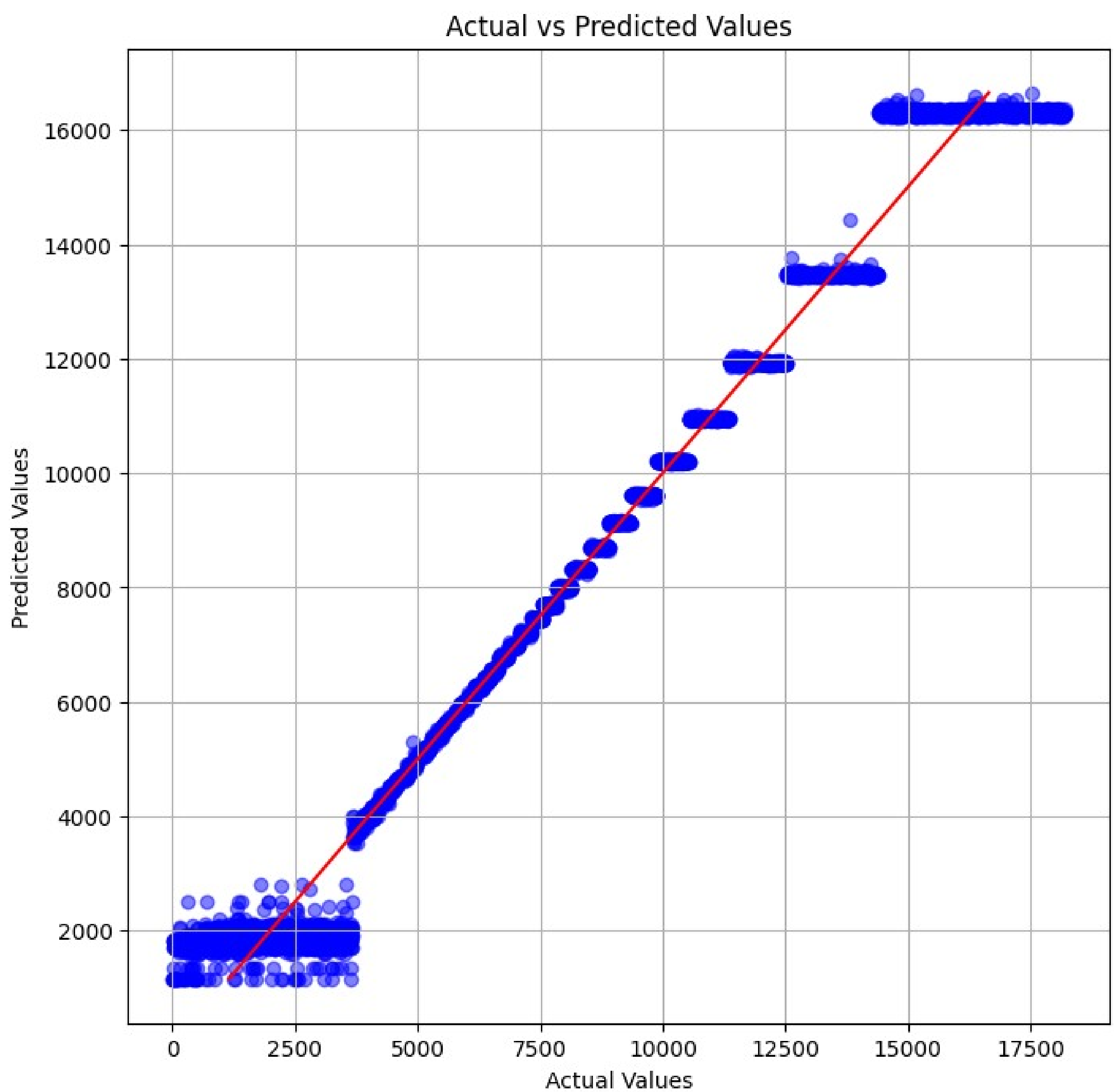
```
plt.figure(figsize=(8,8))
plt.xticks(rotation=90, fontsize=8)
sns.barplot(data=best_model, x=best_model['model'], y=best_model['R2 Score'], hue=best_model['R2 Score'], palette="viridis")
plt.xlabel("Models")
plt.ylabel("R2 Score")
plt.title("Comparing R2 Score for diffrent models")
plt.legend(title="R2 Score", loc="lower left", bbox_to_anchor=(1, 0.50))
plt.show()
```



Actual vs Predicted

```
plt.figure(figsize=(8,8))
# plt.scatter(Y_test, y_pred5, color='blue', alpha=0.5)
```

```python
# plt.plot([min(Y_test), max(y_pred5)], [min(Y_test), max(y_pred5)], color='red')
plt.scatter(Y_test, y_pred5, color='blue', alpha=0.5)
plt.plot([min(y_pred5), max(y_pred5)], [min(y_pred5), max(y_pred5)], color='red')
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Values")
plt.grid(True)
plt.show()
```



BEST MODEL IS GradientBoostingRegressor R2 Score is 0.981413.