**Project 2: Predicting IMDb Scores**

**TEAM MEMBER: THIRISHA P**

**Phase 1: Problem Definition and Design Thinking:**

In this part you will need to understand the problem statement and create a document on what have you understood and how will you proceed ahead with solving the problem. Please think on a design and present in form of a document.

**Problem Definition:**

The problem is to develop a machine learning model that predicts IMDb scores of movies available on Films based on features like genre, premiere date, runtime, and language. The objective is to create a model that accurately estimates the popularity of movies, helping users discover highly rated films that match their preferences. This project involves data preprocessing, feature engineering, model selection, training, and evaluation .

**Design Thinking:**

1. **Data Source:** Utilize a dataset containing information about movies, including features like genre, premiere date, runtime, language, and IMDb scores.
2. **Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.
3. **Feature Engineering:** Extract relevant features from the available data that could contribute to predicting IMDb scores.
4. **Model Selection:** Choose appropriate regression algorithms (e.g., Linear Regression, Random Forest Regressor) for predicting IMDb scores.
5. **Model Training:** Train the selected model using the preprocessed data.
6. **Evaluation:** Evaluate the model's performance using regression metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

**Working Methodology:**

The working method for this work involves few steps. The methodology is shown in figure 1. The steps are described below.

• Data Extraction

• Data Preprocessing

• Applying Machine Learning Techniques

• Comparing the results of different algorithms

DATA EXTRACTION

DATA PREPROCESSING

MACHINE LEARNING TECHNIQUES

**Algorithm :**  Algorithm for developing the model

1: Prepare data set

2: Check Minority

3: If needed apply SMOTE algorithm until the minority class becomes equal to the size of it’s closest class 4: Classification

5: Accuracy ←− 0

6: while True do

7: Resample Data

8: Call (Classifier)

9: if % of correctly classified Instance >Previous Accuracy Measure then

10: Accuracy ←− % of correctly classif ied Instance

11: else

12: Break

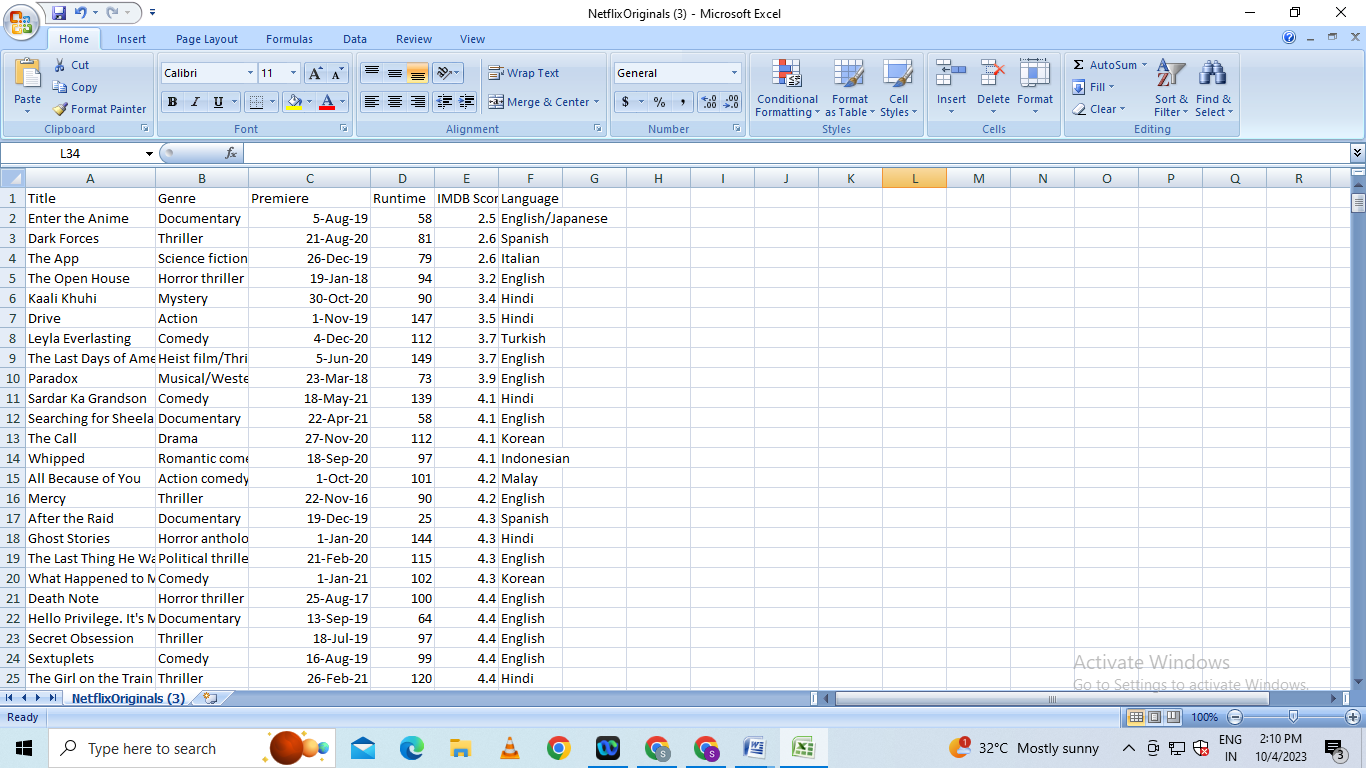
13: end if

14: end while=0

**Data Source:**

A Good Data for Predicting IMDb Scores using machine learning model should be Accurate , complete , covering the geographic area of interest , accessible

Dataset Link :[**https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores**](https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores)



**Data Preprocessing:**

Data preprocessing is the critical first step in any machine learning project.It involves cleaning the data,removing outliers and handling missing values to prepare the dataset for model training. In the context of the predicting the IMDB scores project , let’s elaborate on the specific steps:

1. **Duplicate Removal:**

Duplicate rows can introduce bias into model.We will identify and remove duplicates,typically by sorting the dataset based on unique identifier and then eliminating consecutive rows with same identifiers

**b)Handling Missing Values:**

Missing data is common and needs to be addressed . We will utilize suitable methods such as :

* **Mean Imputation**
* **Median Imputation**

## Imports

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv

In [2]:

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from datetime import datetime,timedelta

**Dataset**

In [3]:

ds = pd.read\_csv("/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv",encoding = "ISO-8859-1")

ds\_date = ds.copy()

ds.head(5)

Out[3]:

|  | Title | Genre | Premiere | Runtime | IMDB Score | Language |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | Enter the Anime | Documentary | August 5, 2019 | 58 | 2.5 | English/Japanese |
| 1 | Dark Forces | Thriller | August 21, 2020 | 81 | 2.6 | Spanish |
| 2 | The App | Science fiction/Drama | December 26, 2019 | 79 | 2.6 | Italian |
| 3 | The Open House | Horror thriller | January 19, 2018 | 94 | 3.2 | English |
| 4 | Kaali Khuhi | Mystery | October 30, 2020 | 90 | 3.4 | Hindi |

In [4]:

ds.describe().T

Out[4]:

|  | count | mean | std | min | 25% | 50% | 75% | max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Runtime | 584.0 | 93.577055 | 27.761683 | 4.0 | 86.0 | 97.00 | 108.0 | 209.0 |
| IMDB Score | 584.0 | 6.271747 | 0.979256 | 2.5 | 5.7 | 6.35 | 7.0 | 9.0 |

insights: categorical of IMDB Score 5.7 > rendah 6.35 > sedang 7.0 > tinggi 9.0 > sangat tinggi

In [5]:

ds.info(verbose=True,show\_counts=True)

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 584 entries, 0 to 583

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Title 584 non-null object

1 Genre 584 non-null object

2 Premiere 584 non-null object

3 Runtime 584 non-null int64

4 IMDB Score 584 non-null float64

5 Language 584 non-null object

dtypes: float64(1), int64(1), object(4)

memory usage: 27.5+ KB

In [6]:

ds.isna().sum()

Out[6]:

Title 0

Genre 0

Premiere 0

Runtime 0

IMDB Score 0

Language 0

dtype: int64

In [7]:

ds['Title'].value\_counts()

Out[7]:

Enter the Anime 1

Have a Good Trip: Adventures in Psychedelics 1

Tallulah 1

The Old Guard 1

Tony Robbins: I Am Not Your Guru 1

..

Cam 1

Earthquake Bird 1

Frankenstein's Monster's Monster, Frankenstein 1

Horse Girl 1

David Attenborough: A Life on Our Planet 1

Name: Title, Length: 584, dtype: int64

In [8]:

ds['Genre'].value\_counts()

Out[8]:

Documentary 159

Drama 77

Comedy 49

Romantic comedy 39

Thriller 33

...

Romantic comedy-drama 1

Heist film/Thriller 1

Musical/Western/Fantasy 1

Horror anthology 1

Animation/Christmas/Comedy/Adventure 1

Name: Genre, Length: 115, dtype: int64

In [9]:

ds['Premiere'].value\_counts()

Out[9]:

October 2, 2020 6

November 1, 2019 5

October 18, 2019 5

November 2, 2018 4

June 19, 2020 4

..

September 20, 2019 1

March 10, 2017 1

March 17, 2017 1

May 29, 2015 1

October 4, 2020 1

Name: Premiere, Length: 390, dtype: int64

In [10]:

ds\_date["Premiere"] = ds\_date["Premiere"].apply(lambda x: "".join(x for x **in** x.replace(".",",")))

ds\_date["PremiereDate"] = ds\_date["Premiere"].apply(lambda x: datetime.strptime(x, "%B **%d**, %Y").date())

ds\_date["Year"] = ds\_date["Premiere"].apply(lambda x: "".join(x for x **in** x.replace(",","").split()[-1]))

*#Convert object to date*

ds\_date["PremiereDate"] = pd.to\_datetime(ds\_date["PremiereDate"])

ds\_date

Out[10]:

|  | Title | Genre | Premiere | Run time | IMDB Score | Language | PremiereDate | Year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Enter the Anime | Documentary | August 5, 2019 | 58 | 2.5 | English/Japanese | 2019-08-05 | 2019 |
| 1 | Dark Forces | Thriller | August 21, 2020 | 81 | 2.6 | Spanish | 2020-08-21 | 2020 |
| 2 | The App | Science fiction/Drama | December 26, 2019 | 79 | 2.6 | Italian | 2019-12-26 | 2019 |
| 3 | The Open House | Horror thriller | January 19, 2018 | 94 | 3.2 | English | 2018-01-19 | 2018 |
| 4 | Kaali Khuhi | Mystery | October 30, 2020 | 90 | 3.4 | Hindi | 2020-10-30 | 2020 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 579 | Taylor Swift: Reputation Stadium Tour | Concert Film | December 31, 2018 | 125 | 8.4 | English | 2018-12-31 | 2018 |
| 580 | Winter on Fire: Ukraine's Fight for Freedom | Documentary | October 9, 2015 | 91 | 8.4 | English/Ukranian/Russian | 2015-10-09 | 2015 |
| 581 | Springsteen on Broadway | One-man show | December 16, 2018 | 153 | 8.5 | English | 2018-12-16 | 2018 |
| 582 | Emicida: AmarElo - It's All For Yesterday | Documentary | December 8, 2020 | 89 | 8.6 | Portuguese | 2020-12-08 | 2020 |
| 583 | David Attenborough: A Life on Our Planet | Documentary | October 4, 2020 | 83 | 9.0 | English | 2020-10-04 | 2020 |

584 rows × 8 columns

In [11]:

ds\_date.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 584 entries, 0 to 583

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Title 584 non-null object

1 Genre 584 non-null object

2 Premiere 584 non-null object

3 Runtime 584 non-null int64

4 IMDB Score 584 non-null float64

5 Language 584 non-null object

6 PremiereDate 584 non-null datetime64[ns]

7 Year 584 non-null object

dtypes: datetime64[ns](1), float64(1), int64(1), object(5)

memory usage: 36.6+ KB

In [12]:

ds['Language'].value\_counts()

Out[12]:

English 401

Hindi 33

Spanish 31

French 20

Italian 14

Portuguese 12

Indonesian 9

Japanese 6

Korean 6

German 5

Turkish 5

English/Spanish 5

Polish 3

Dutch 3

Marathi 3

English/Hindi 2

Thai 2

English/Mandarin 2

English/Japanese 2

Filipino 2

English/Russian 1

Bengali 1

English/Arabic 1

English/Korean 1

Spanish/English 1

Tamil 1

English/Akan 1

Khmer/English/French 1

Swedish 1

Georgian 1

Thia/English 1

English/Taiwanese/Mandarin 1

English/Swedish 1

Spanish/Catalan 1

Spanish/Basque 1

Norwegian 1

Malay 1

English/Ukranian/Russian 1

Name: Language, dtype: int64

EDA

In [13]:

ds['Genre'].value\_counts()

genre = ds['Genre'].value\_counts()

genre.head()

Out[13]:

Documentary 159

Drama 77

Comedy 49

Romantic comedy 39

Thriller 33

Name: Genre, dtype: int64

In [14]:

plt.figure(figsize=(16, 5))

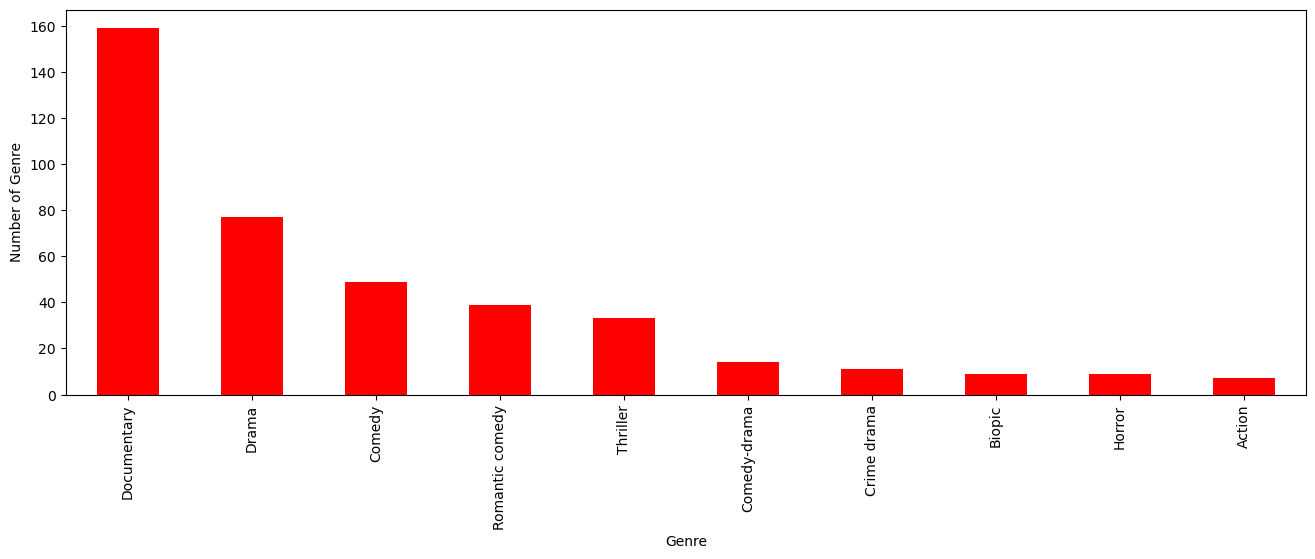
ds['Genre'].value\_counts().head(10).plot(kind='bar', color='red')

plt.xlabel('Genre')

plt.ylabel('Number of Genre')

plt.xticks(rotation=90)

plt.show(block=True)



insights: the most popular movies from genre is documentary

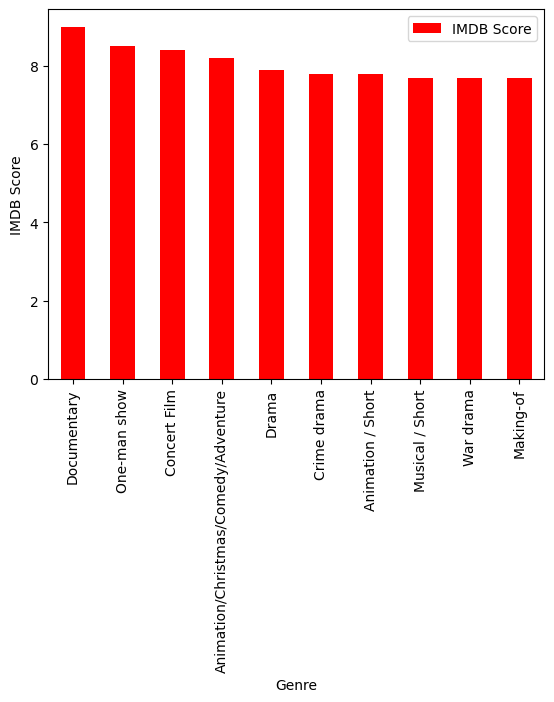
In [15]:

ds[['Genre', 'IMDB Score']].sort\_values('IMDB Score', ascending=False).drop\_duplicates('Genre').head(10).plot(x='Genre', y='IMDB Score', kind='bar', color='red')

plt.xlabel('Genre')

plt.ylabel('IMDB Score')

plt.show(block=True)



In [16]:

ds['Language'].value\_counts()

Out[16]:

English 401

Hindi 33

Spanish 31

French 20

Italian 14

Portuguese 12

Indonesian 9

Japanese 6

Korean 6

German 5

Turkish 5

English/Spanish 5

Polish 3

Dutch 3

Marathi 3

English/Hindi 2

Thai 2

English/Mandarin 2

English/Japanese 2

Filipino 2

English/Russian 1

Bengali 1

English/Arabic 1

English/Korean 1

Spanish/English 1

Tamil 1

English/Akan 1

Khmer/English/French 1

Swedish 1

Georgian 1

Thia/English 1

English/Taiwanese/Mandarin 1

English/Swedish 1

Spanish/Catalan 1

Spanish/Basque 1

Norwegian 1

Malay 1

English/Ukranian/Russian 1

Name: Language, dtype: int64

In [17]:

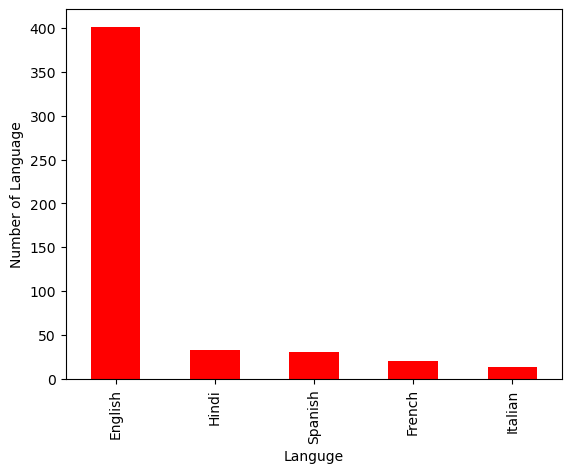
ds\_lang = ds['Language'].value\_counts()

ds\_lang.head(5).plot(kind='bar', color='red')

plt.xlabel('Languge')

plt.ylabel('Number of Language')

plt.show(block=True)



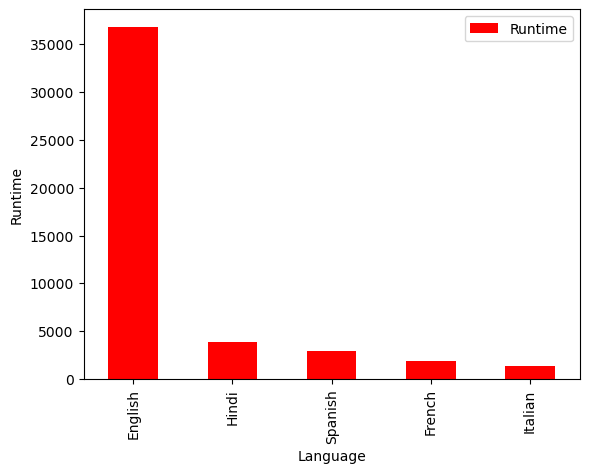
In [18]:

ds.groupby('Language').agg({'Runtime': 'sum'}).sort\_values('Runtime', ascending=False).head(5).plot(kind='bar',color='red')

plt.xlabel('Language')

plt.ylabel('Runtime')

plt.show(block=True)



In [19]:

ds\_english = ds[ds['Language'] == 'English'].sort\_values('IMDB Score', ascending=False)

ds\_english.head()

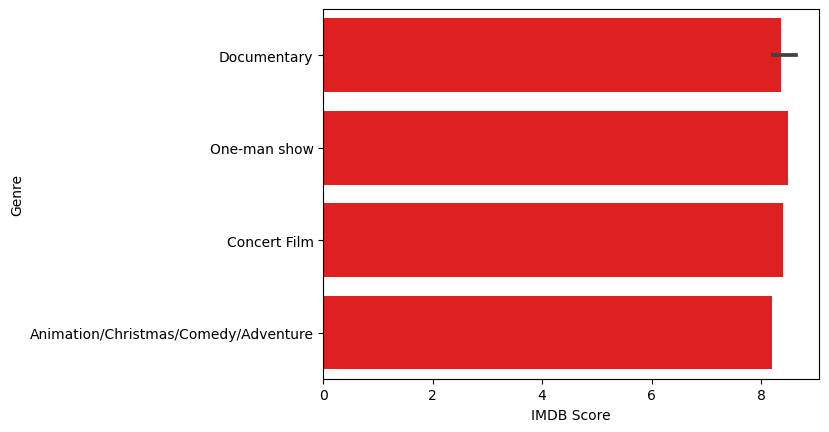
Out[19]:

|  | Title | Genre | Premiere | Runtime | IMDB Score | Language |
| --- | --- | --- | --- | --- | --- | --- |
| 583 | David Attenborough: A Life on Our Planet | Documentary | October 4, 2020 | 83 | 9.0 | English |
| 581 | Springsteen on Broadway | One-man show | December 16, 2018 | 153 | 8.5 | English |
| 579 | Taylor Swift: Reputation Stadium Tour | Concert Film | December 31, 2018 | 125 | 8.4 | English |
| 578 | Ben Platt: Live from Radio City Music Hall | Concert Film | May 20, 2020 | 85 | 8.4 | English |
| 577 | Dancing with the Birds | Documentary | October 23, 2019 | 51 | 8.3 | English |

In [20]:

sns.barplot(y=ds\_english['Genre'].head(10), x=ds\_english['IMDB Score'], color='red')

plt.show(block=True)

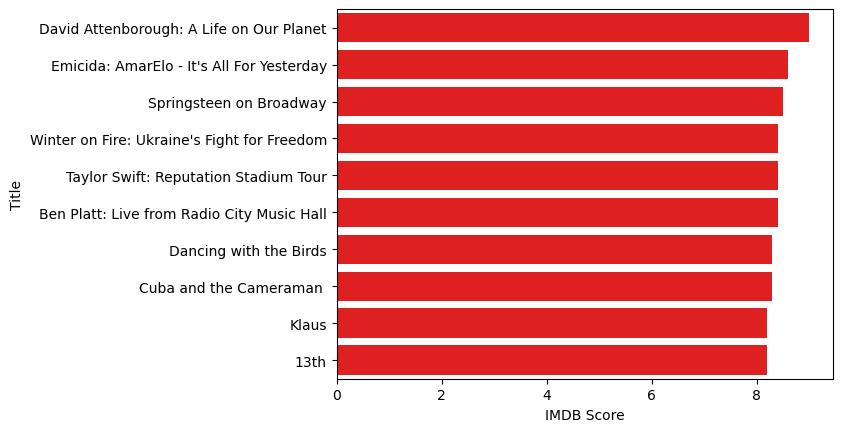


In [21]:

ds\_movie = ds[['Title', 'IMDB Score']].sort\_values('IMDB Score', ascending=False).head(10)

sns.barplot(y='Title', x='IMDB Score', data=ds\_movie, color='red')

plt.show(block=True)



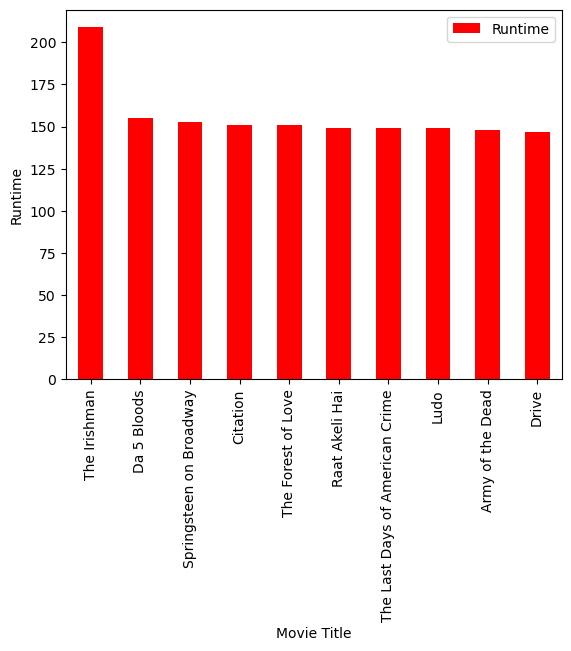
In [22]:

ds[['Title', 'Runtime']].sort\_values('Runtime', ascending=False).head(10).plot(x='Title', y='Runtime', kind='bar', color='red')

plt.xlabel('Movie Title')

plt.ylabel('Runtime')

plt.show(block=True)



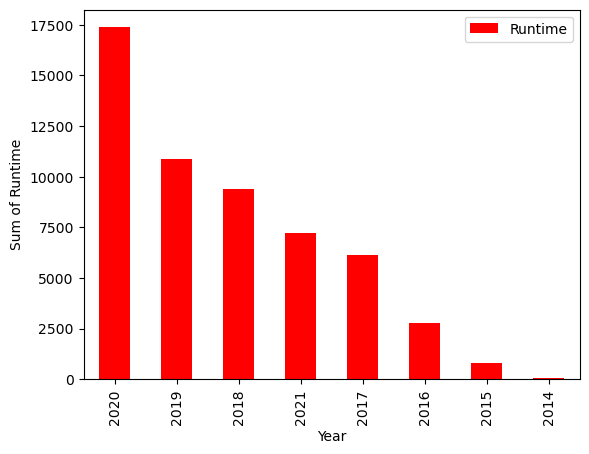
In [23]:

ds\_date.groupby('Year').agg({'Runtime': 'sum'}).sort\_values('Runtime', ascending=False).plot(kind='bar', color='red')

plt.xlabel('Year')

plt.ylabel('Sum of Runtime')

plt.show(block=True)



In [24]:

ds\_date.groupby('Year').agg({'Title': 'count'}).sort\_values('Title', ascending=False).plot(kind='bar', color='red')

plt.xlabel('Year')

plt.ylabel('Number of Film')

plt.show(block=True)

