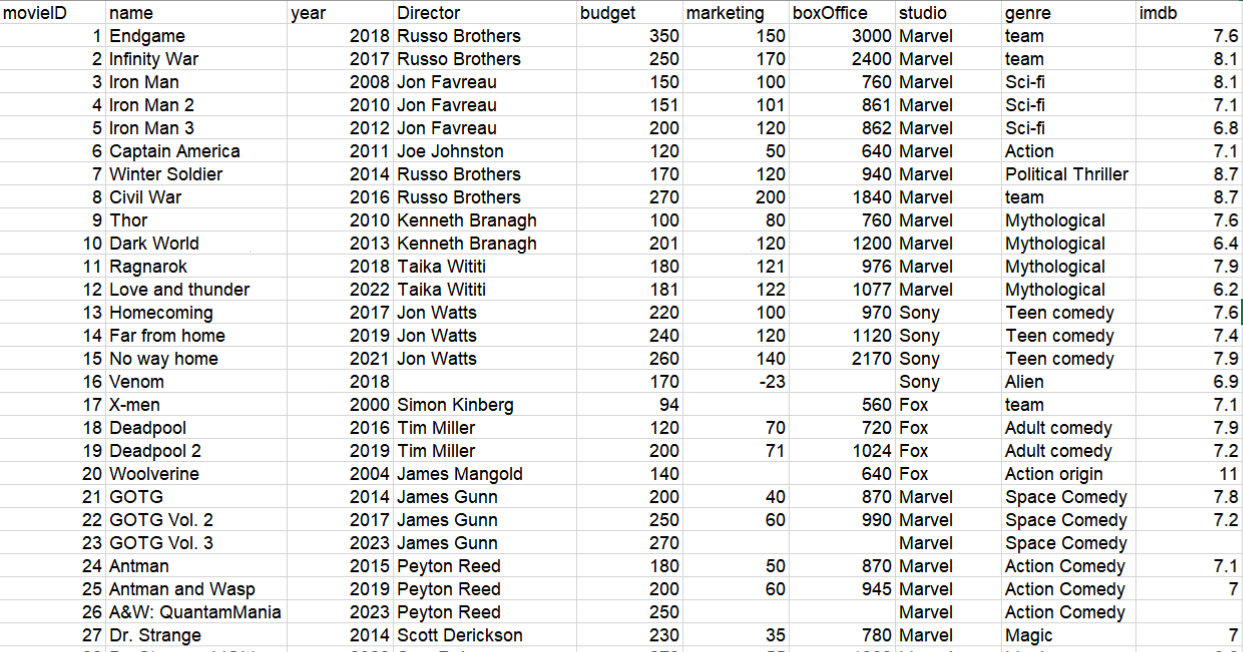
**Vivian Joseph**

20BCE0777

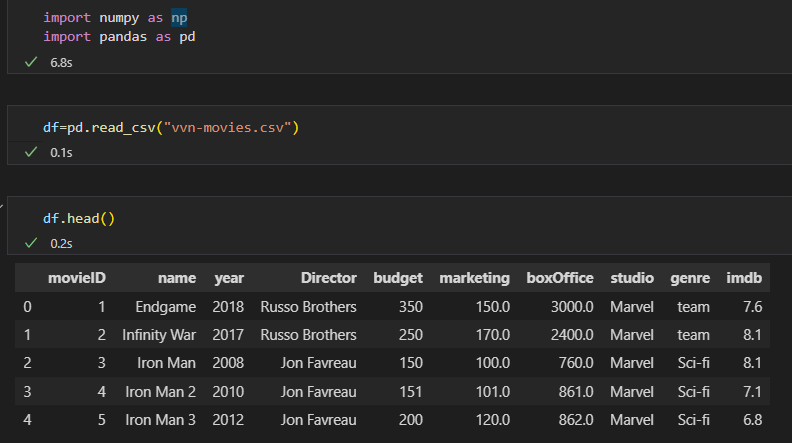
**Machine Learning LAB DA-1**

**Pre-processing of Data**

**Q1) Create your own dataset - 9 column/ features and minimum 30 samples/rows.**



Importing the data into the notebook



**Q2) Apply Data Manipulation Operations like:**

1. **Insert new sample**

to insert new row:

df=df.append({'movieID':30,'name':'Black Panther','year':2018,'Director':'Ryan Coogler','imdb':8.1},ignore\_index=True)

output:

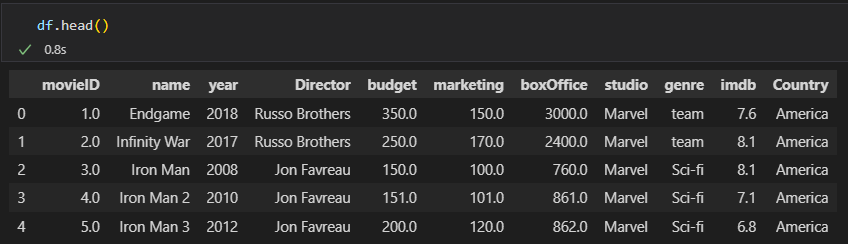




to insert a column:

df.insert(10,"Country","America")

output:

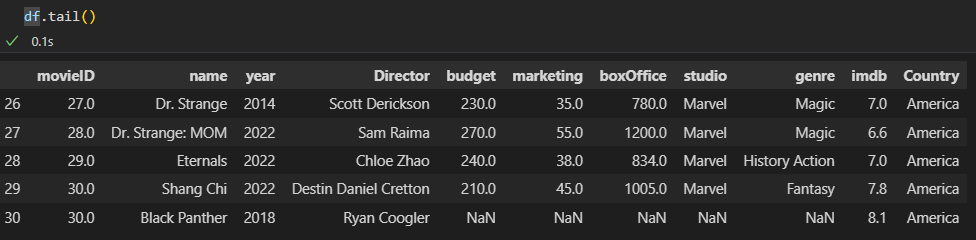


1. **Delete particular sample**

dropping row by row id:

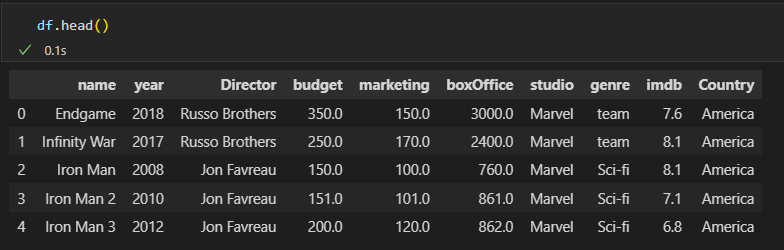
df.drop(29, axis=0, inplace=True)

output:



dropping column:

df.drop('movieID', inplace=True, axis=1)

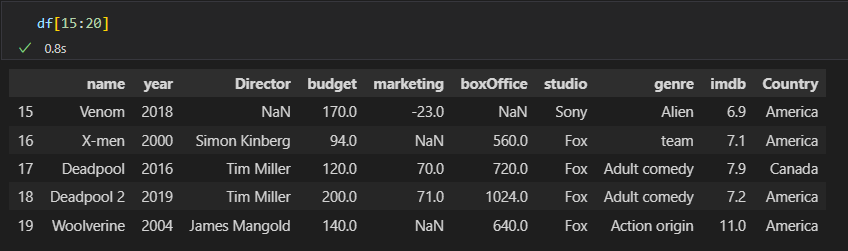


1. **Update particular sample**

To update values at specific sample

df.at[17,'Country']="Canada"

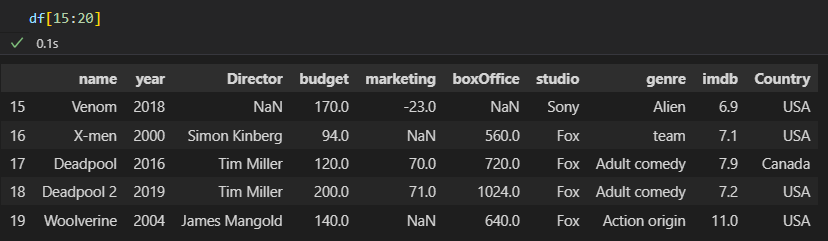
output:



Replacing given values across the data

df.replace("America", "USA", inplace=True)

output:



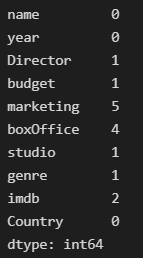
**Q3) Apply Data pre-processing like:**

1. **Find no. of missing values**

Find nan values in each column:

df.isna().sum()

output:



Find total nan values in df

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

output:



1. **Replace missing values by mean, median and mode operations.**

mean\_value=df['marketing'].mean()

df['marketing'].fillna(value=mean\_value, inplace=True)

median\_value=df['boxOffice'].median()

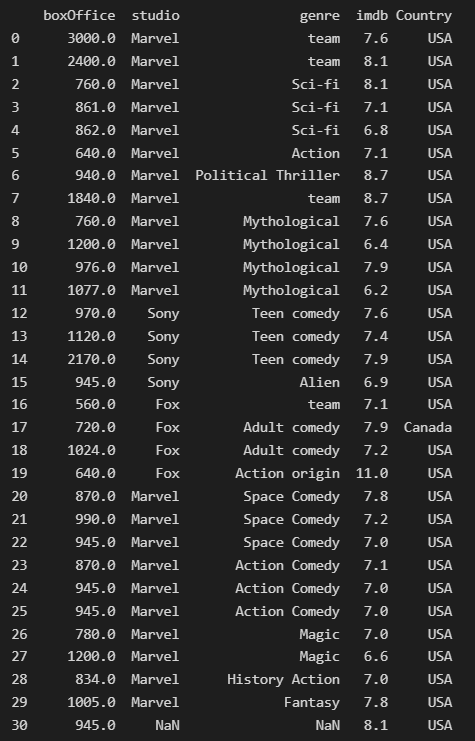
df['boxOffice'].fillna(value=median\_value, inplace=True)

mode\_value=df['imdb'].mode()

df['imdb'].fillna(value=int(mode\_value), inplace=True)

print(df)

output:



replaced by median replaced by mode



replaced by mean



**c. Apply encoding techniques like**

i. ordinal to categorical

values=['fresh','rotten']

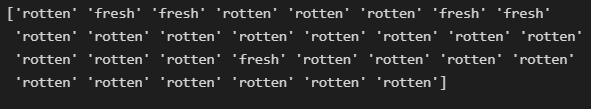
conditions=[(df["imdb"]>8),(df["imdb"]<=8)]

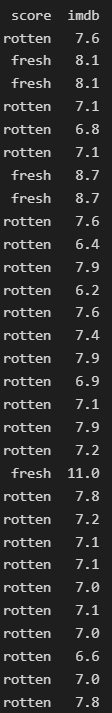
arr=np.select(conditions,values)

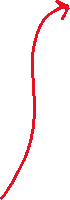
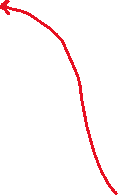
print(arr)

df.insert(9,"score",arr)

output:



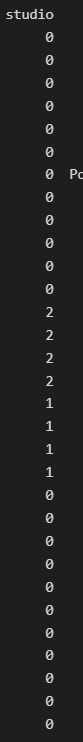
Categorical data  Original numeric data (no categorical data available in dataset)

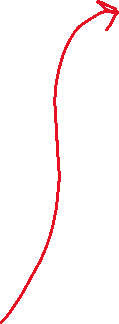
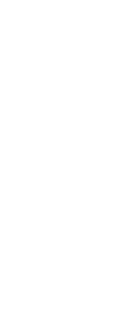


ii. categorical (text) to categorical (numeric)

df['Country'].replace([0,1], ['USA', 'Canada'], inplace=True)

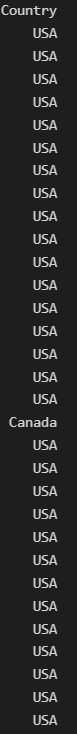
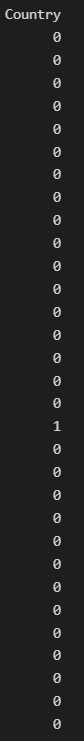
Output:

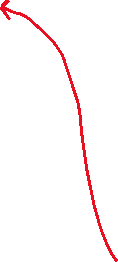
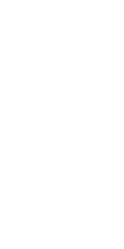
Categorical (text) Data   Categorical (numeric) Data



iii. label encoding to binary encoding

df['studio'].replace(['Marvel', 'Fox', 'Sony'], [0,1,2], inplace=True)

label encoding   binary encoding



**Q4) Apply Normalization techniques**

**a.Minmax**

df\_min\_max\_scaled = df[['imdb','boxOffice','marketing','budget']]

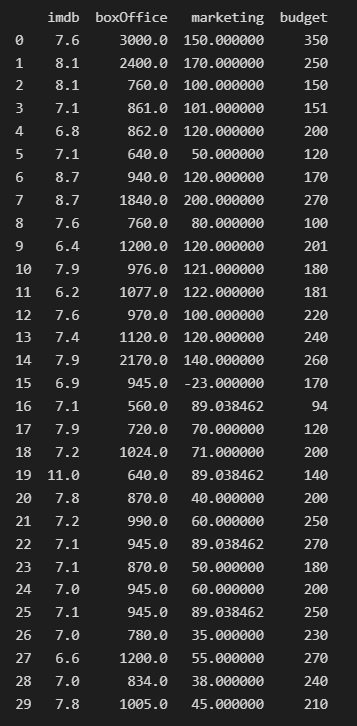
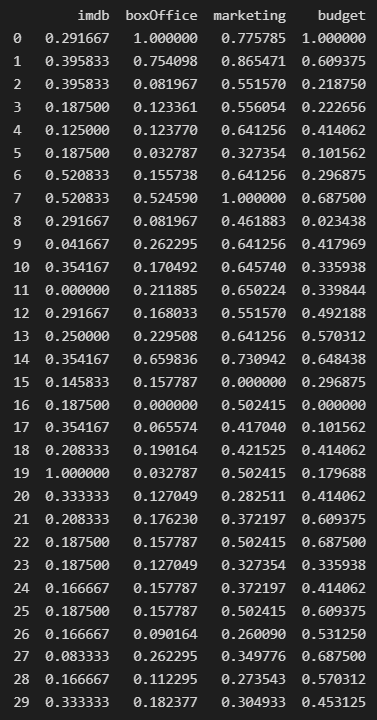
for column in df\_min\_max\_scaled.columns:

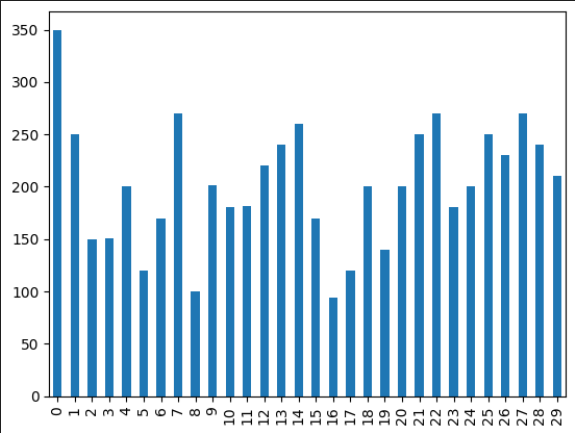
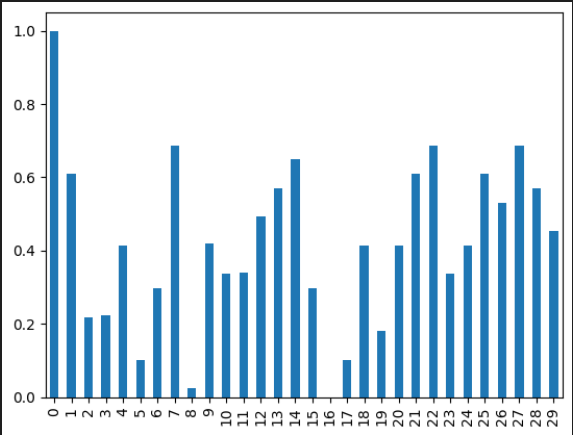
    df\_min\_max\_scaled[column] = (df\_min\_max\_scaled[column] - df\_min\_max\_scaled[column].min()) / (df\_min\_max\_scaled[column].max() - df\_min\_max\_scaled[column].min())

print(df\_min\_max\_scaled)

Output:

Before After MinMax Normalization

Budget column bar graph (before and after)

b. Standard scaling (according to mean)

df\_standard\_scaled = df[['imdb','boxOffice','marketing','budget']]

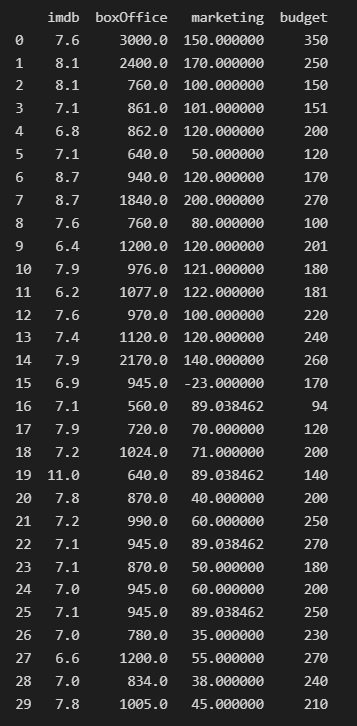
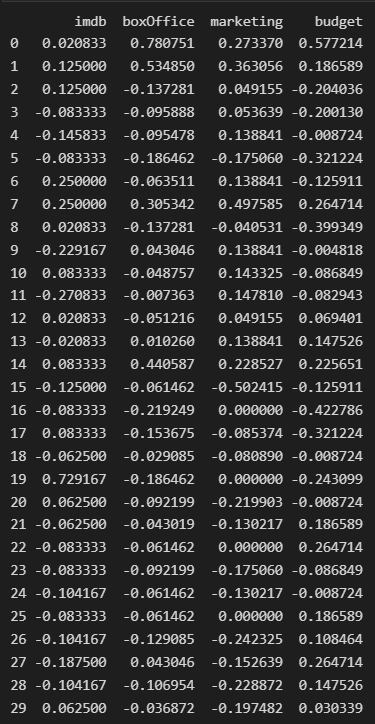
for column in df\_standard\_scaled.columns:

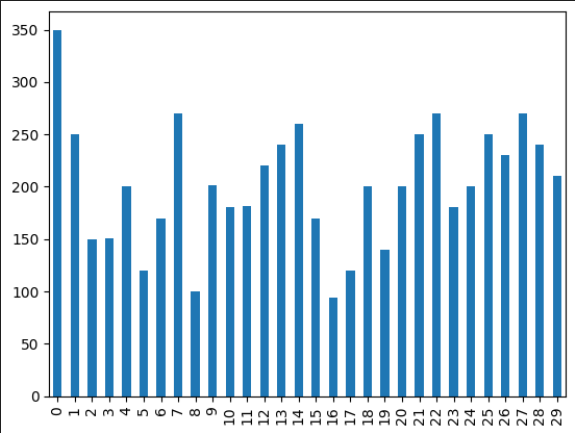
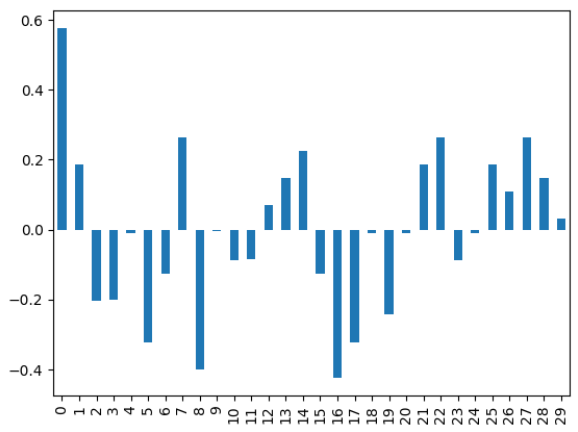
    df\_standard\_scaled[column] = (df\_standard\_scaled[column] - df\_standard\_scaled[column].mean()) / (df\_standard\_scaled[column].max() - df\_standard\_scaled[column].min())

print(df\_standard\_scaled)

Output:

Before After

Budget column bar graph (before and after)