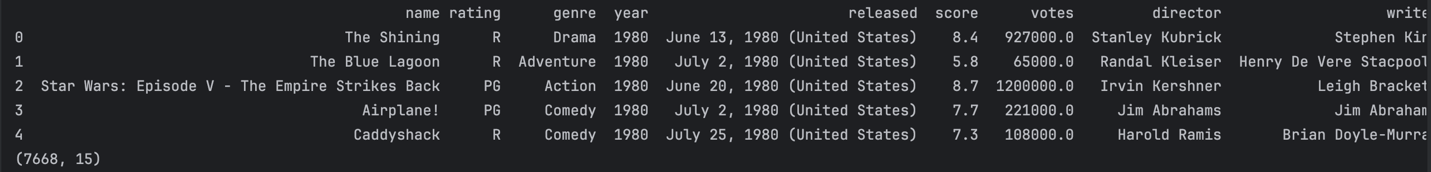
#importing libraries  
  
import pandas as pd #version 2.2.1  
import seaborn as sns #version 0.13.2  
import matplotlib.pyplot as plt #version 3.8.3  
  
plt.style.use('ggplot')  
plt.rcParams["figure.figsize"] = (10, 8)  
  
# Reading data and displaying top 5 rows and columns  
df = pd.read\_csv("/Users/lakshmipuninchittaya/Downloads/movies.csv")  
pd.options.display.width= None  
pd.options.display.max\_columns= None  
print(df.head())



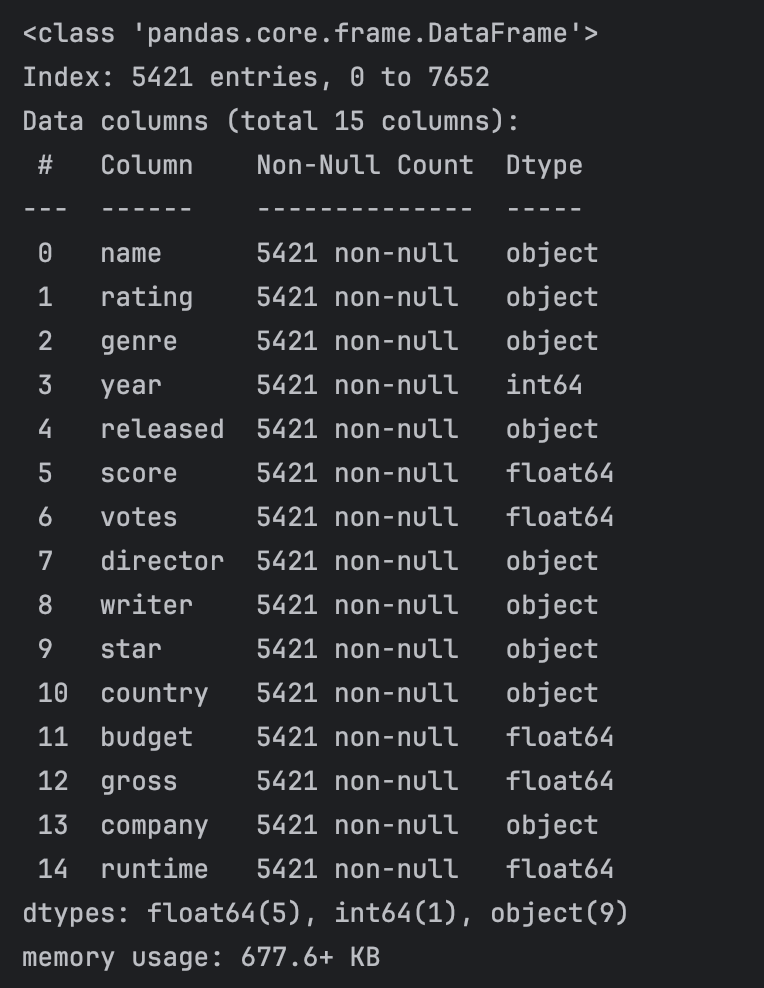
print(df.shape)

A number on a black background

Description automatically generated  
  
# looking for missing values  
for col in df.columns:  
 print(df[col].isnull().value\_counts(), "\n")

A screenshot of a computer

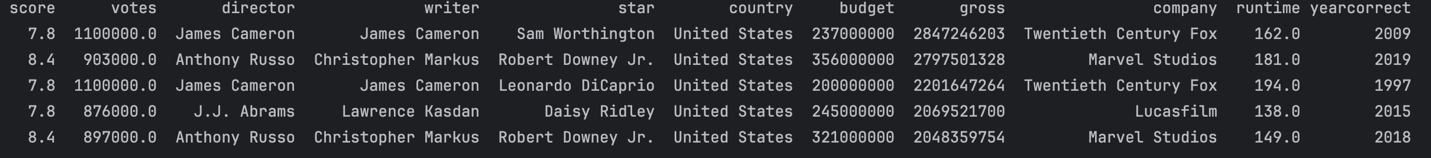
Description automatically generated  
  
#drop na rows and duplicate rows  
df = df.dropna()  
df = df.drop\_duplicates()  
  
#getting data info  
df.info()

  
  
#changing Data types of columns  
df['budget'] = df['budget'].astype(int)  
df['gross'] = df['gross'].astype(int)  
print(df.head(10))

A screenshot of a computer screen

Description automatically generated

#creating new column with release year  
df['yearcorrect'] = df['released'].astype(str).str.split().str[2]  
df1 = df.sort\_values(by=['gross'], inplace=False, ascending=False)  
print(df1.head())

  
  
#scatterplot with budget vs gross earnings  
plt.scatter(x=df['budget'], y=df['gross'])  
plt.title('Budget vs Gross earnings')  
plt.xlabel('Budget')  
plt.ylabel('Gross earnings')  
plt.show()

A graph with red dots

Description automatically generated  
  
#plot budget vs gross earnings using seaborn  
sns.regplot(x='budget',y='gross',data =df, scatter\_kws={"color":"red"}, line\_kws={"color":"blue"}).set(title ='Budget vs Gross earnings')  
plt.show()

A graph with red dots and a blue line

Description automatically generated  
  
#generating correlation matrix  
print(df.head(5))  
Matrix = df.corr(method="pearson", numeric\_only=True)  
print(Matrix)

A black screen with white numbers

Description automatically generated

#visualizing correlation matrix of numeric variables  
sns.heatmap(Matrix, annot=True)  
plt.title('Correlation matrix for numeric features')  
plt.xlabel('Movie features')  
plt.ylabel('Movie features')  
plt.show()

A screenshot of a graph

Description automatically generated

#copying df and saving it as df\_numerized  
df\_numerized = df.copy()  
  
#changing string variables into numeric  
for col\_name in df\_numerized.columns:  
 if (df\_numerized[col\_name].dtype=='object'):  
 df\_numerized[col\_name]=df\_numerized[col\_name].astype('category')  
 df\_numerized[col\_name]=df\_numerized[col\_name].cat.codes  
  
print(df\_numerized.head())

A black background with white numbers

Description automatically generated  
  
#correlation matrix for all variables  
MatrixF = df\_numerized.corr(method="pearson", numeric\_only=True)  
print(MatrixF)

A black screen with numbers

Description automatically generated

sns.heatmap(MatrixF, annot=True)  
plt.title('Correlation matrix for all features')  
plt.xlabel('Movie features')  
plt.ylabel('Movie features')  
plt.show()

A screenshot of a computer

Description automatically generated

#sorted matrix to find high correlated variables  
MatrixF = df\_numerized.corr(method="pearson", numeric\_only=True)  
Matrix\_pairs = MatrixF.unstack()  
print(Matrix\_pairs)

A screenshot of a computer

Description automatically generated

sorted\_pairs= Matrix\_pairs.sort\_values() #sorting correlation values  
print(sorted\_pairs)

A screen shot of a computer

Description automatically generated

high\_corr = sorted\_pairs[(sorted\_pairs) >0.5] #displaying only high correlated pairs  
print(high\_corr)

A screenshot of a computer screen

Description automatically generated