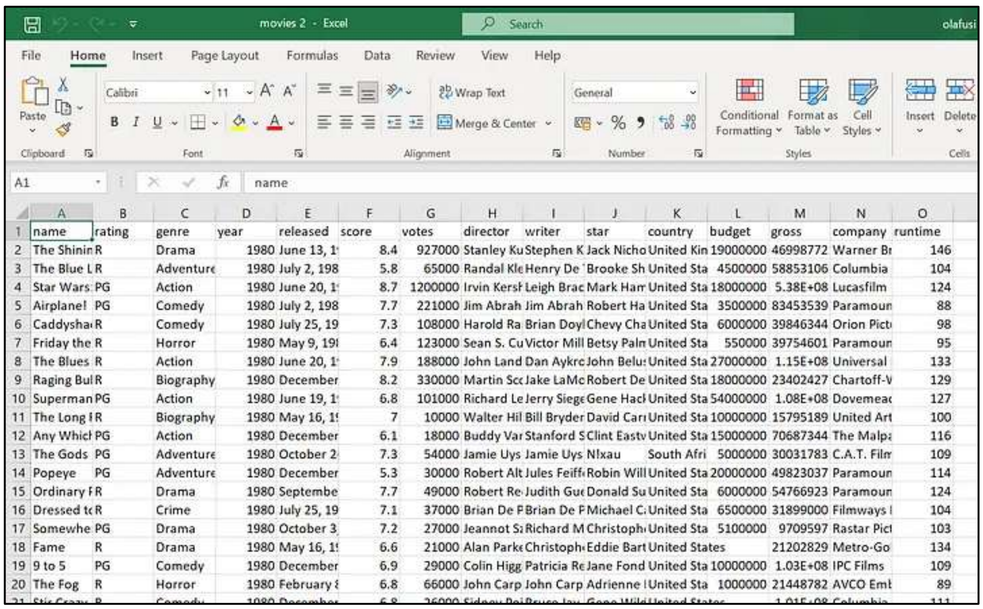
<u>Correlation Project on Python Using a Movie Data Set</u>

"Correlation is a statistical technique used to determine the degree to which two variables are related."

A) SUMMARY OF THE DATA SET

The Data set is titled "movies" and was taken from Kaggle.

1. There are 15 columns in the Data set.
2. Column Names [name, rating genre, year, released, score, votes, director, writer, star, country, budget, gross, company, runtime]
3. 7,669 Rows of Data.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime
2	The Shinin	R	Drama	1980	June 13, 1	8.4	927000	Stanley Ku	Stephen K	Jack Nicho	United Kin	19000000	46998772	Warner Br	146
3	The Blue L	R	Adventure	1980	July 2, 198	5.8	65000	Randal Kle	Henry De	Brooke Sh	United Sta	4500000	58853106	Columbia	104
4	Star Wars:	PG	Action	1980	June 20, 1	8.7	1200000	Irvin Kersh	Leigh Brac	Mark Ham	United Sta	18000000	5.38E+08	Lucasfilm	124
5	Airplane!	PG	Comedy	1980	July 2, 198	7.7	221000	Jim Abrah	Jim Abrah	Robert Ha	United Sta	3500000	83453539	Paramoun	88
6	Caddysha	R	Comedy	1980	July 25, 19	7.3	108000	Harold Ra	Brian Doyl	Chevy Cha	United Sta	6000000	39846344	Orion Pict	98
7	Friday the	R	Horror	1980	May 9, 198	6.4	123000	Sean S. Cu	Victor Mill	Betsy Paln	United Sta	550000	39754601	Paramoun	95
8	The Blues	R	Action	1980	June 20, 1	7.9	188000	John Land	Dan Aykro	John Belu	United Sta	27000000	1.15E+08	Universal	133
9	Raging Bul	R	Biography	1980	December	8.2	330000	Martin Sco	Jake LaMo	Robert De	United Sta	18000000	23402427	Chartoff-W	129
10	Superman	PG	Action	1980	June 19, 1	6.8	101000	Richard Le	Jerry Siege	Gene Hack	United Sta	54000000	1.08E+08	Dovemead	127
11	The Long	R	Biography	1980	May 16, 19	7	10000	Walter Hil	Bill Bryden	David Carr	United Sta	10000000	15795189	United Art	100
12	Any Which	PG	Action	1980	December	6.1	18000	Buddy Var	Stanford S	Clint Eastv	United Sta	15000000	70687344	The Malpa	116
13	The Gods	PG	Adventure	1980	October 2	7.3	54000	Jamie Uys	Jamie Uys	N!xau	South Afri	5000000	30031783	C.A.T. Film	109
14	Popeye	PG	Adventure	1980	December	5.3	30000	Robert Alt	Jules Feiff	Robin Will	United Sta	20000000	49823037	Paramoun	114
15	Ordinary F	R	Drama	1980	Septembe	7.7	49000	Robert Re	Judith Gue	Donald Su	United Sta	6000000	54766923	Paramoun	124
16	Dressed to	R	Crime	1980	July 25, 19	7.1	37000	Brian De P	Brian De P	Michael C	United Sta	6500000	31899000	Filmways	104
17	Somewhe	PG	Drama	1980	October 3	7.2	27000	Jeannot Sz	Richard M	Christoph	United Sta	5100000	9709597	Rastar Pict	103
18	Fame	R	Drama	1980	May 16, 19	6.6	21000	Alan Parke	Christoph	Eddie Bart	United States		21202829	Metro-Go	134
19	9 to 5	PG	Comedy	1980	December	6.9	29000	Colin Higg	Patricia Re	Jane Fond	United Sta	10000000	1.03E+08	IPC Films	109
20	The Fog	R	Horror	1980	February	6.8	66000	John Carp	John Carp	Adrienne	United Sta	1000000	21448782	AVCO Emb	89

B) STEP BY STEP ANALYSIS AND RESULTS

1. **Importing Libraries such as Matplotlib, Seaborn, Pandas and NumPy.**

a) Importing Libraries ¶

```
# Importing the packages
import pandas as pd # analyzing, cleaning, exploring, and manipulat
import numpy as np # mathematical operations
import seaborn as sns # making statistical graphics

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

%matplotlib inline # the plots are displayed inline within the note
matplotlib.rcParams['figure.figsize'] = (12,8)

pd.options.mode.chained_assignment = None
```

2. **Reading the Data. CSV import to Jupyter Notebook.**

```
# Reading the data
df = pd.read_csv(r'C:\Users\DELL\Desktop\Project\Project Movie Correlation - Python\movies.csv')
```

3. **Having a quick look at the Data, the top 5 rows (head) to be precise.**

```
# Looking at the data
df.head()
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	146.0
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	104.0
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	124.0
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures	88.0
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures	98.0

C) CLEANING THE DATA

1. **Searching for Missing Data which might affect our Visualization. Using *foreloop* to loop through every column to search for missing Data.**

b) Handling Missing data

```
# Checking if we have any missing data

for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
name - 0%
rating - 1%
genre - 0%
year - 0%
released - 0%
score - 0%
votes - 0%
director - 0%
writer - 0%
star - 0%
country - 0%
budget - 28%
gross - 2%
company - 0%
runtime - 0%
```

The result above shows that there are missing (Null) values in certain columns such as **Ratings, Budget, Gross.** We have this percentage but we also need to be sure what is the actual number.

```python
# Checking missing data in actual numbers

print(df.isnull().sum())
```

```
name           0
rating        77
genre          0
year           0
released       2
score          3
votes          3
director       0
writer         3
star           1
country        3
budget      2171
gross        189
company       17
runtime        4
dtype: int64
```

The actual numbers show that there are missing values in rating, released, score, votes, writer, star, country, budget, gross, company and runtime.

2. Looking at the Data Types for each Columns.

```python
# Data Types for our columns

print(df.dtypes)
```

```
name         object
rating       object
genre        object
year          int64
released     object
score       float64
votes       float64
director     object
writer       object
star         object
country      object
budget      float64
gross       float64
company      object
runtime     float64
dtype: object
```

D) HANDLING THE MISSING DATA

I proposed 2 approaches to handle missing values

```
Missing values Handling
Option 1: Delete those rows (Not recommended)
Option 2: Impute the values
```

Option 1: Deleting the rows

```
# Option 1: Delete those rows

# Dropping rows where 'budget' column has missing values
df_cleaned = df.dropna(subset=['budget'])

# Display DataFrame after dropping rows
df_cleaned.head()
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	146.0
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	104.0
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	124.0
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures	88.0
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures	98.0

```
# Checking missing data in actual numbers
print(df_cleaned.isnull().sum())


# Checking if we have any missing data
for col in df_cleaned.columns:
    pct_missing = np.mean(df_cleaned[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
name         0
rating      20
genre        0
year         0
released     0
score        3
votes        3
director     0
writer       0
star         1
country      1
budget       0
gross       61
company      6
runtime      2
dtype: int64
name - 0%
rating - 0%
genre - 0%
year - 0%
released - 0%
score - 0%
votes - 0%
director - 0%
writer - 0%
star - 0%
country - 0%
budget - 0%
gross - 1%
```

After deleting the rows with missing values in the column budget there seems to be negligible number of missing values.

Option 2: Impute the values (Using KNN)

```python
# Option 2: Impute the values

from sklearn.impute import KNNImputer

# Initialize the KNN Imputer
imputer = KNNImputer(n_neighbors=10)

# Copy the original DataFrame (for further comparision bw the two)
df_imputed = df.copy()

# Select the columns to be used for imputation
columns_to_impute = ['budget', 'gross', 'score']

# Apply KNN imputation
df_imputed[columns_to_impute] = imputer.fit_transform(df_imputed[columns_to_impute])

# Display the DataFrame with imputed values
df_imputed.head()
```

- **Why KNN?**

Estimates values

KNN imputation uses the concept of nearest neighbors to consider multiple variables and estimate values for missing data points.

Preserves data

KNN imputation maintains the value and variability of datasets, unlike other methods that can waste data or reduce variability.

```python
# Checking if we have any missing data

for col in df_imputed.columns:
    pct_missing = np.mean(df_imputed[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))

# Checking missing data in actual numbers
print(df_imputed.isnull().sum())
```

```
name - 0%
rating - 1%
genre - 0%
year - 0%
released - 0%
score - 0%
votes - 0%
director - 0%
writer - 0%
star - 0%
country - 0%
budget - 0%
gross - 0%
company - 0%
runtime - 0%
name         0
rating      77
genre        0
year         0
released     2
score        0
votes        3
director     0
writer       3
star         1
country      3
budget       0
gross        0
company     17
```

After imputing the rows in the column budget, gross and score there seems to be negligible number of missing values.

```
# Comparing rows with only imputed values

# a) Identify rows where 'budget' was imputed
imputed_rows = df['budget'].isnull()

# b) Creating a DataFrame for imputed rows with certain columns
comparison_df = pd.DataFrame({
    'movie_name': df.loc[imputed_rows, 'name'],
    'score_original': df.loc[imputed_rows, 'score'],
    'year': df.loc[imputed_rows, 'year'],
    'budget_original': df.loc[imputed_rows, 'budget'],
    'budget_imputed': df_imputed.loc[imputed_rows, 'budget'],
    'gross_original': df.loc[imputed_rows, 'gross'],
    'gross_imputed': df_imputed.loc[imputed_rows, 'gross']
})

# c) Display the comparison DataFrame
comparison_df.head()
```

	movie_name	score_original	year	budget_original	budget_imputed	gross_original	gross_imputed
16	Fame	6.6	1980	NaN	1987800.0	21202829.0	21202829.0
19	Stir Crazy	6.8	1980	NaN	5252500.0	101300000.0	101300000.0
24	Urban Cowboy	6.4	1980	NaN	2470500.0	46918287.0	46918287.0
25	Altered States	6.9	1980	NaN	5922800.0	19853892.0	19853892.0
26	Little Darlings	6.5	1980	NaN	5969300.0	34326249.0	34326249.0

After comparison of the original columns in budget and gross, it seems to have been imputed using KNN.

E) CLEANING THE DATA AFTER IMPUTATION

1. Extracting the year from released year. Correcting the Year column, since some numbers in year are wrong and show a different number. A new column was made using year from released column as YearCorrected.

c) Correcting Year Column

```
# Extract year and convert to float
df['YearCorrect'] = df['released'].str.extract(r'(\d{4})').astype(float)
df.head()
```

name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime	YearCorrect
Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	146.0	1980.0
he Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	104.0	1980.0
ar Wars: sode V - e Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	124.0	1980.0
irplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures	88.0	1980.0
dyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures	98.0	1980.0

2. Creating a new temp column named as Match to verify if the number in year and released year is different.

```
# Check if the two columns match
df['Match'] = df['year'] == df['YearCorrect']

# Display the DataFrame with the match column
df.head()
```

rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime	YearCorrect	Match
R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.	146.0	1980.0	True
R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures	104.0	1980.0	True
PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm	124.0	1980.0	True
PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures	88.0	1980.0	True
R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures	98.0	1980.0	True

3. Verifying if the numbers were actually different and yes, they were.

```
# Filter rows where Match is False
false_matches = df[df['Match'] == False][['name', 'year', 'YearCorrect']]

false_matches.head()
```

	name	year	YearCorrect
8	Superman II	1980	1981.0
11	The Gods Must Be Crazy	1980	1984.0
21	Heaven's Gate	1980	1981.0
33	Cattle Annie and Little Britches	1980	1981.0
40	The Watcher in the Woods	1980	1981.0

4. Proving the results.

Superman II (Subtitled) / Release date

June 19, 1981

USA

5. Ordering it by the Gross Revenue Column from descending to ascending order.

d) Ordering data and dropping duplicates

```
# Ordering our data
df.sort_values(by=['gross'], inplace = False, ascending = False)
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	r
5445	Avatar	PG-13	Action	2009	December 18, 2009 (United States)	7.8	1100000.0	James Cameron	James Cameron	Sam Worthington	United States	237000000.0	2.847246e+09	Twentieth Century Fox	
7445	Avengers: Endgame	PG-13	Action	2019	April 26, 2019 (United States)	8.4	903000.0	Anthony Russo	Christopher Markus	Robert Downey Jr.	United States	356000000.0	2.797501e+09	Marvel Studios	
3045	Titanic	PG-13	Drama	1997	December 19, 1997 (United States)	7.8	1100000.0	James Cameron	James Cameron	Leonardo DiCaprio	United States	200000000.0	2.201647e+09	Twentieth Century Fox	
6663	Star Wars: Episode VII - The Force Awakens	PG-13	Action	2015	December 18, 2015 (United States)	7.8	876000.0	J.J. Abrams	Lawrence Kasdan	Daisy Ridley	United States	245000000.0	2.069522e+09	Lucasfilm	
7244	Avengers: Infinity War	PG-13	Action	2018	April 27, 2018 (United States)	8.4	897000.0	Anthony Russo	Christopher Markus	Robert Downey Jr.	United States	321000000.0	2.048360e+09	Marvel Studios	

The result shows the highest grossing movie as Avatar and continues to descend from there.

6. Checking for duplicates and dropping them.

```
# Dropping Duplicates
df.drop_duplicates()
```

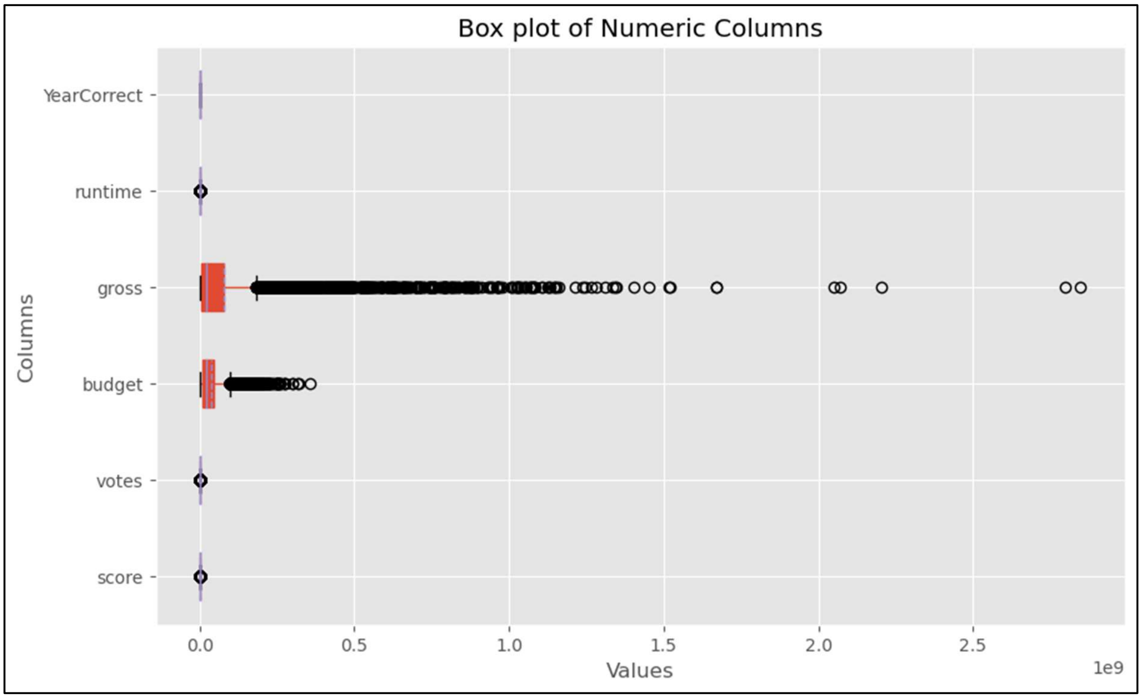
	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick	Stephen King	Jack Nicholson	United Kingdom	19000000.0	46998772.0	Warner Bros.
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser	Henry De Vere Stacpoole	Brooke Shields	United States	4500000.0	58853106.0	Columbia Pictures
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980	June 20, 1980 (United States)	8.7	1200000.0	Irvin Kershner	Leigh Brackett	Mark Hamill	United States	18000000.0	538375067.0	Lucasfilm
3	Airplane!	PG	Comedy	1980	July 2, 1980 (United States)	7.7	221000.0	Jim Abrahams	Jim Abrahams	Robert Hays	United States	3500000.0	83453539.0	Paramount Pictures
4	Caddyshack	R	Comedy	1980	July 25, 1980 (United States)	7.3	108000.0	Harold Ramis	Brian Doyle-Murray	Chevy Chase	United States	6000000.0	39846344.0	Orion Pictures

No duplicated were found and dropped as this is a distinct count of the values in the column.

7. Checking for outliers.

```
# Any Outliers?

# Plotting a box plot for numeric columns
numerical_columns = ['score', 'votes', 'budget', 'gross', 'runtime', 'YearCorrec
df_box = df[numerical_columns]

# Creating the box plot
plt.figure(figsize=(10, 6))
df_box.boxplot(vert=False, patch_artist=True, meanline=True, showmeans=True)
plt.title('Box plot of Numeric Columns')
plt.xlabel('Values')
plt.ylabel('Columns')
plt.show()
```

No outliers were found. The values in the gross section which is above 2 are high grossing movies such as Avatar, Avengers, Titanic and more. These movies are crucial in our analysis and should not be removed.



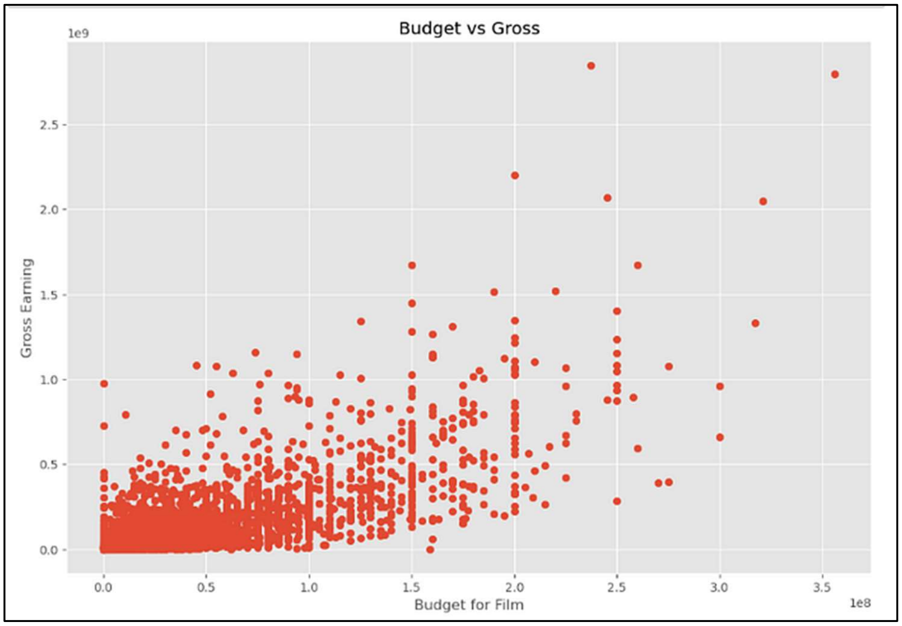
8. Looking at all of the Data instead of just snippets of it.

```
# Displaying max numbers of rows
pd.set_option('display.max_rows', None)
```

F) STATISTICAL ANALYSIS AND VISUALIZING THE DATA

1. Normal scatter plot

```
In [11]: #Correlation

         #Build a scatter plot with budget vs gross
         #notice the x and y axis

         plt.scatter(x=df['budget'], y=df['gross'])

         plt.title('Budget vs Gross')  # Labeling the scatter plot

         plt.xlabel('Budget for Film')  # Labeling the X axis (corrected method name)

         plt.ylabel('Gross Earning')  # Labeling the Y axis (corrected method name)
         plt.show()
```

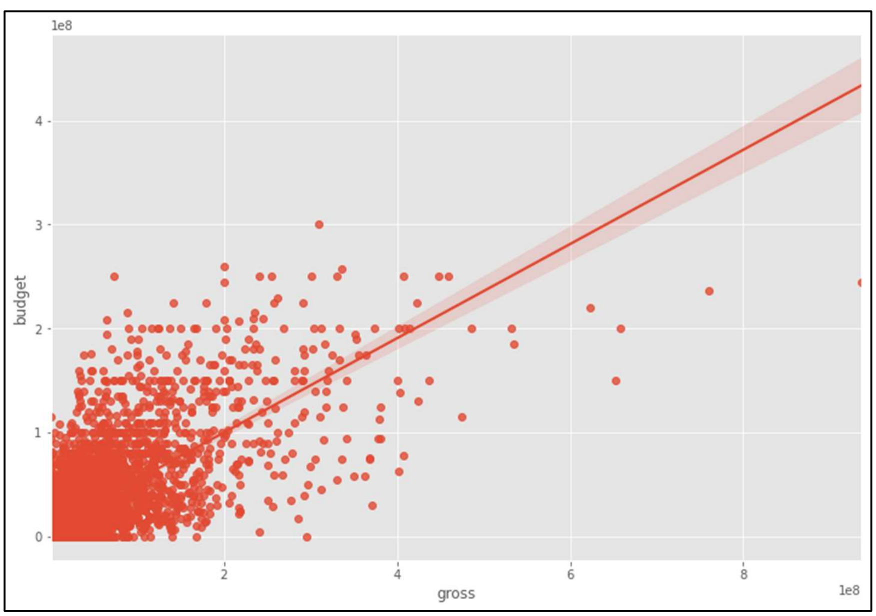


It is quite hard to see if they are corelated using this, next we use SEABORN.

2. Budget Vs Gross using SEABORN showing R*egression Plot.*

e) Statistical Analysis

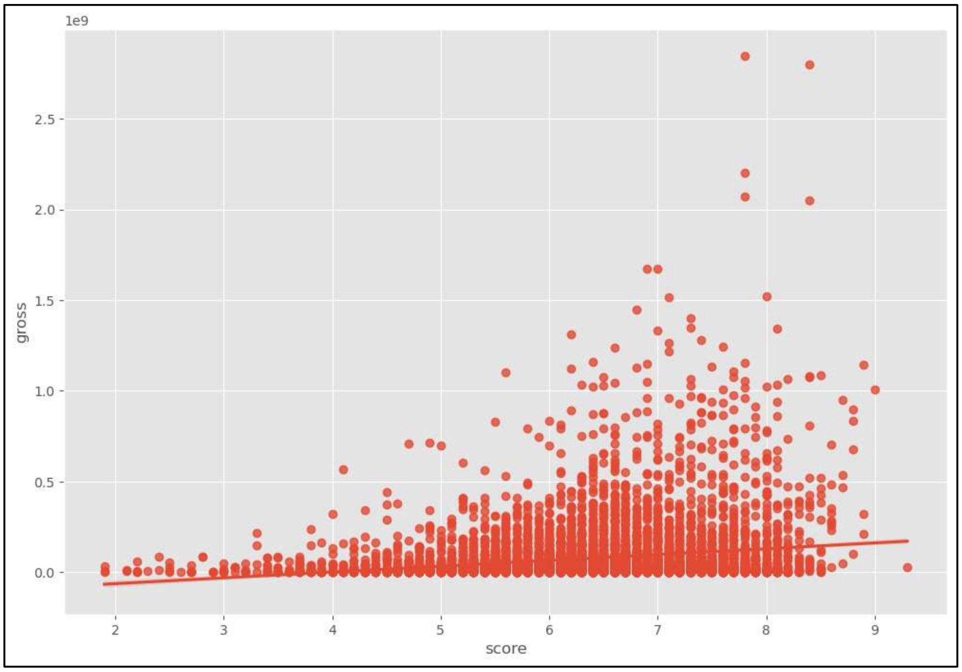
```
# a) Scatter chart with Regression Line (Gross and Budget)
sns.regplot(x="gross", y="budget", data=df)
```



This shows the degree of correlation but it is still unclear due to the cluster at the low end of the Budget and gross scatter plot table. But there is a positive correlation so this means that as the independent variable increases, the dependent variable also tends to increase or as budget increases the gross value increases.

3. Score Vs Gross using SEABORN showing R*egression Plot.*

```python
# b) Scatter chart with Regression Line (score and Gross)
sns.regplot(x="score", y="gross", data=df)
```

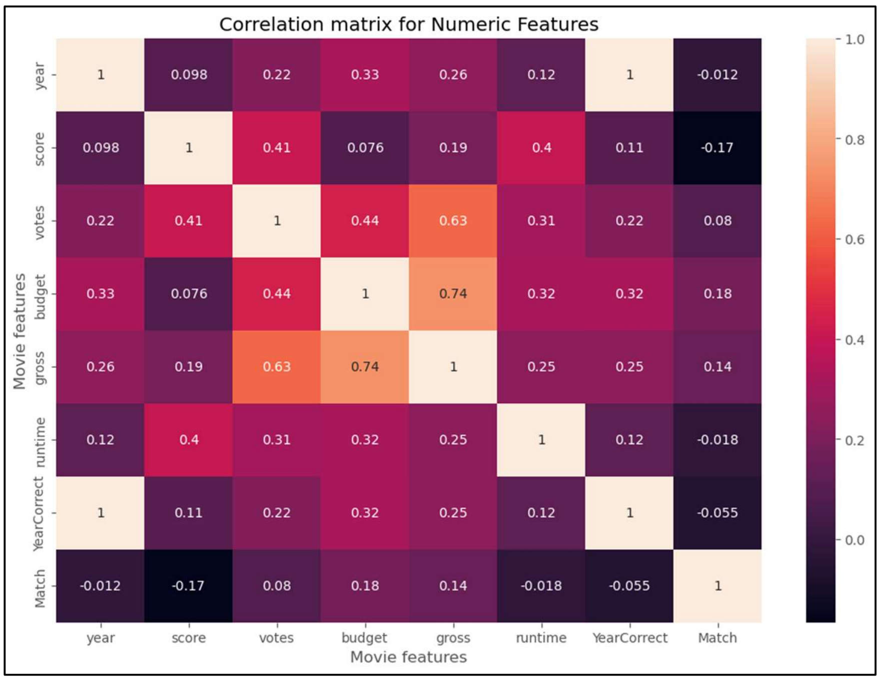


This shows the degree of correlation but it is still unclear due to the cluster at the medium end of the score plot table. But there is a positive correlation so this means that as the independent variable increases, the dependent variable also tends to increase or as score increases the gross value increases.

4. Determining correlation using Pearson method

```
# c) Correlation Matrix between all numeric columns (pearson)

df.corr(method ='pearson') # research into these
```

	year	score	votes	budget	gross	runtime	YearCorrect	Match
year	1.000000	0.097995	0.222945	0.329321	0.257486	0.120811	0.997415	-0.012163
score	0.097995	1.000000	0.409182	0.076254	0.186258	0.399451	0.105994	-0.165669
votes	0.222945	0.409182	1.000000	0.442429	0.630757	0.309212	0.218429	0.080450
budget	0.329321	0.076254	0.442429	1.000000	0.740395	0.320447	0.321918	0.180592
gross	0.257486	0.186258	0.630757	0.740395	1.000000	0.245216	0.250514	0.144928
runtime	0.120811	0.399451	0.309212	0.320447	0.245216	1.000000	0.120636	-0.018272
YearCorrect	0.997415	0.105994	0.218429	0.321918	0.250514	0.120636	1.000000	-0.055025
Match	-0.012163	-0.165669	0.080450	0.180592	0.144928	-0.018272	-0.055025	1.000000



BLACK = Super Low Correlation

BRIGHT COLOURS = High Correlation

The **Pearson correlation coefficient** measures the strength and direction of the linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation). A value of 0 indicates no correlation.

The correlation between year and score is 0.097995, which indicates a very **weak positive correlation**. The correlation between votes and gross is 0.630757, which indicates a **strong positive correlation**. There is a **very strong positive correlation** between budget and gross at 0.740395. This suggests that with higher budget, and higher gross revenue tend to be related.

There are strong positive correlations between votes, budget, and gross. This suggests that movies with more votes, higher budgets, and higher gross revenue tend to be related.

5. Determining correlation using Kendall method

```
# d) Correlation Matrix between all numeric columns (kendall)
df.corr(method ='kendall')
```

	year	score	votes	budget	gross	runtime	YearCorrect	Match
year	1.000000	0.067652	0.331465	0.224120	0.200618	0.097184	0.987642	-0.009845
score	0.067652	1.000000	0.300115	-0.000566	0.086046	0.283611	0.073436	-0.150789
votes	0.331465	0.300115	1.000000	0.353702	0.548899	0.198240	0.325814	0.075703
budget	0.224120	-0.000566	0.353702	1.000000	0.512637	0.235483	0.216871	0.190249
gross	0.200618	0.086046	0.548899	0.512637	1.000000	0.168933	0.190789	0.217702
runtime	0.097184	0.283611	0.198240	0.235483	0.168933	1.000000	0.096999	-0.014291
YearCorrect	0.987642	0.073436	0.325814	0.216871	0.190789	0.096999	1.000000	-0.044701
Match	-0.009845	-0.150789	0.075703	0.190249	0.217702	-0.014291	-0.044701	1.000000

Kendall correlation coefficient measures the ordinal association between two variables. Unlike Pearson correlation, which assesses linear relationships, Kendall correlation is more robust to outliers and non-linear patterns.

Observations:

Strong Positive Correlation:

- No relatively significantly important strong positive correlation.

Moderate Positive Correlation:

- votes and gross: A moderate positive correlation (0.548899) suggest that movies with more votes tend to have higher gross revenue.
- votes and budget: A moderate positive correlation (0.353702) implies that movies with higher budgets often receive more votes.
- budget and gross: A moderate positive correlation (0.512637) suggests that movies with higher budgets tend to have higher gross revenue.

Weak or No Correlation:

- Most other correlations are relatively weak, suggesting limited ordinal relationships between those variables.

6. Determining correlation using Spearman method

```
# e) Correlation Matrix between all numeric columns (spearman)
df.corr(method ='spearman')
```

	year	score	votes	budget	gross	runtime	YearCorrect	Match
year	1.000000	0.099045	0.469829	0.317336	0.293084	0.142977	0.997407	-0.011908
score	0.099045	1.000000	0.428138	-0.001403	0.126116	0.399857	0.107602	-0.181927
votes	0.469829	0.428138	1.000000	0.502466	0.742050	0.290159	0.462767	0.092504
budget	0.317336	-0.001403	0.502466	1.000000	0.693670	0.336370	0.306901	0.231051
gross	0.293084	0.126116	0.742050	0.693670	1.000000	0.246243	0.278701	0.266611
runtime	0.142977	0.399857	0.290159	0.336370	0.246243	1.000000	0.142915	-0.017339
YearCorrect	0.997407	0.107602	0.462767	0.306901	0.278701	0.142915	1.000000	-0.054066
Match	-0.011908	-0.181927	0.092504	0.231051	0.266611	-0.017339	-0.054066	1.000000

Spearman correlation measures the strength and direction of the monotonic relationship between two variables. However, unlike Kendall correlation, Spearman correlation assigns ranks to the data points before calculating the correlation coefficient.

<u>**Observations:**</u>

Strong Positive Correlation:

- No relatively significantly important strong positive correlation.

Moderate Positive Correlation:

- votes and gross: A moderate positive correlation (0.742050) suggest that movies with more votes tend to have higher gross revenue.
- votes and budget: A moderate positive correlation (0.502466) implies that movies with higher budgets often receive more votes.
- budget and gross: A moderate positive correlation (0.693670) suggests that movies with higher budgets tend to have higher gross revenue.

Weak or No Correlation:

- Most other correlations are relatively weak, suggesting limited monotonic relationships between those variables.

7. Comparing categorical variables by factorizing method.

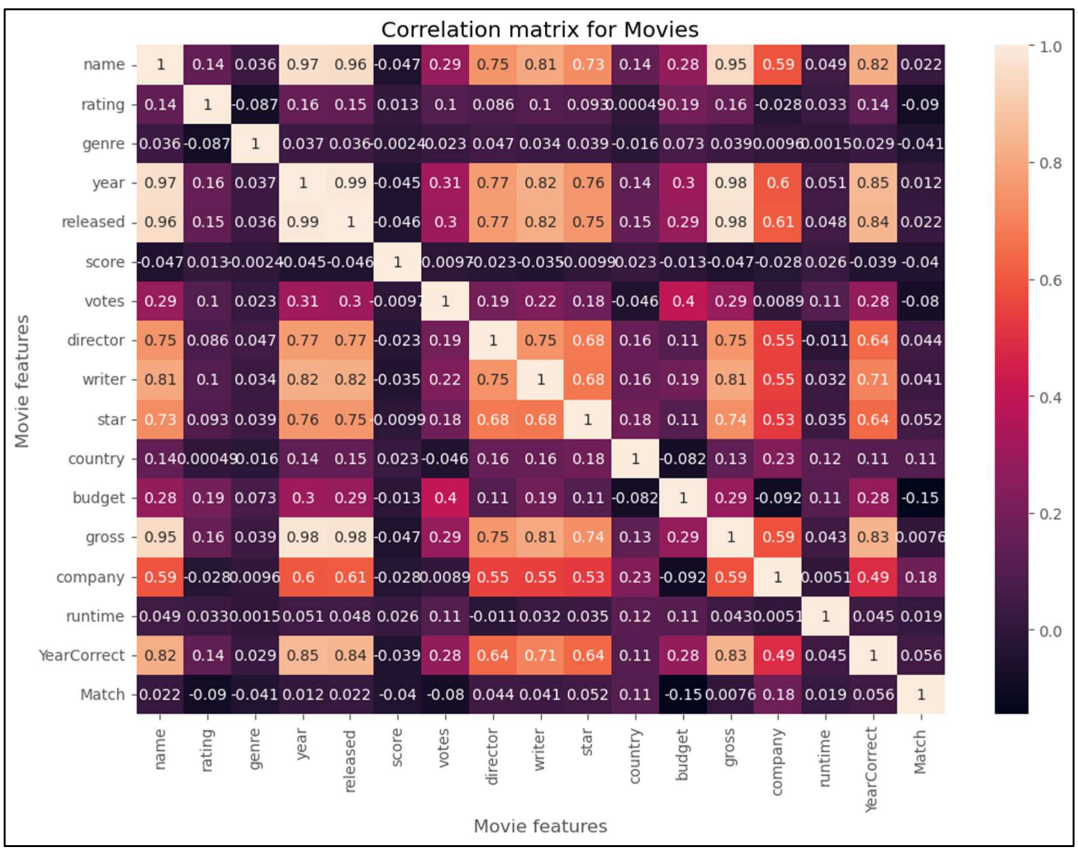
This correlation matrix compares all the columns by factorizing it and assigning a random numeric value for each categorical value.

Factor analysis is a statistical method that can be used to define the underlying structure of a group of related variables, including their correlations. It does this by identifying a set of common underlying dimensions, known as factors. This technique can be used to analyse the structure of correlations among a large number of variables

```
# g) Comparing categorical variables
#  factorize - (assigns a random numeric value for each unique categorical value)

df.apply(lambda x: x.factorize()[0]).corr(method='pearson')
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	compar
name	1.000000	0.143938	0.036367	0.965761	0.959015	-0.046733	0.287776	0.745905	0.805211	0.731565	0.142828	0.277488	0.947324	0.5916
rating	0.143938	1.000000	-0.086723	0.156713	0.146606	0.012595	0.099972	0.085520	0.103623	0.093116	0.000494	0.193353	0.158582	-0.0280
genre	0.036367	-0.086723	1.000000	0.037184	0.035940	-0.002437	0.023285	0.047288	0.033688	0.038649	-0.015795	0.073008	0.038616	0.0095
year	0.965761	0.156713	0.037184	1.000000	0.993190	-0.044981	0.312401	0.770497	0.824770	0.756400	0.140216	0.300621	0.980873	0.6015
released	0.959015	0.146606	0.035940	0.993190	1.000000	-0.045761	0.299905	0.770876	0.819617	0.754468	0.148468	0.285691	0.976423	0.6079
score	-0.046733	0.012595	-0.002437	-0.044981	-0.045761	1.000000	-0.009749	-0.022687	-0.034685	-0.009896	0.023097	-0.012642	-0.047041	-0.0284
votes	0.287776	0.099972	0.023285	0.312401	0.299905	-0.009749	1.000000	0.192220	0.224122	0.179601	-0.045914	0.398519	0.286180	0.0089
director	0.745905	0.085520	0.047288	0.770497	0.770876	-0.022687	0.192220	1.000000	0.748340	0.682385	0.155471	0.106617	0.750911	0.5522
writer	0.805211	0.103623	0.033688	0.824770	0.819617	-0.034685	0.224122	0.748340	1.000000	0.675685	0.157202	0.187238	0.805576	0.5461
star	0.731565	0.093116	0.038649	0.756400	0.754468	-0.009896	0.179601	0.682385	0.675685	1.000000	0.182045	0.107991	0.735680	0.5271
country	0.142828	0.000494	-0.015795	0.140216	0.148468	0.023097	-0.045914	0.155471	0.157202	0.182045	1.000000	-0.082082	0.133982	0.2263
budget	0.277488	0.193353	0.073008	0.300621	0.285691	-0.012642	0.398519	0.106617	0.187238	0.107991	-0.082082	1.000000	0.285832	-0.0922
gross	0.947324	0.158582	0.038616	0.980873	0.976423	-0.047041	0.286180	0.750911	0.805576	0.735680	0.133982	0.285832	1.000000	0.5881
company	0.591667	-0.028035	0.009566	0.601571	0.607954	-0.028432	0.008900	0.552258	0.546151	0.527116	0.226346	-0.092249	0.588156	1.0000
runtime	0.048955	0.032741	0.001462	0.050647	0.048235	0.026436	0.106024	-0.011070	0.032264	0.035392	0.124154	0.112097	0.042978	0.0051
YearCorrect	0.819948	0.137371	0.028570	0.848638	0.840589	-0.038776	0.276969	0.641912	0.706832	0.638927	0.112782	0.284229	0.831610	0.4941
Match	0.021866	-0.089590	-0.041163	0.012163	0.022037	-0.040074	-0.079588	0.043826	0.040508	0.052350	0.109184	-0.145122	0.007593	0.1753

```
# h) Plotting a full correlation heatmap with categorical data as well.
correlation_matrix = df.apply(lambda x: x.factorize()[0]).corr(method='pearson')
sns.heatmap(correlation_matrix, annot = True)
plt.title("Correlation matrix for Movies")
plt.xlabel("Movie features")
plt.ylabel("Movie features")
plt.show()
```



Observations:

Strong Positive Correlations:

- Gross and Budget: A moderate to strong positive correlation (0.29) suggests that higher budget movies tend to have higher gross revenue.
- YearCorrect and Year: A very strong positive correlation (0.85) indicates that the YearCorrect feature is likely a refined or corrected version of the Year feature.
- Director, Writer, and Star with Gross: Moderate positive correlations (0.75, 0.81, 0.73 respectively) suggest that movies with well-known directors, writers, or stars tend to have higher gross revenue.

Strong Negative Correlations:

- Score and Votes: A weak negative correlation (-0.04) suggests a minimal relationship between movie score and the number of votes.
- Match and several other features: The "Match" feature shows very low correlations with most other features, indicating it is a unique or unrelated variable (which we created).

8. Unstacking to see the ones with the highest correlation and better understanding of a particular variable with others. This is useful when we have to analyse a particular variable with all others.

```
# i) Unstacking and one to one view of correlation.
correlation_mat = df.apply(lambda x: x.factorize()[0]).corr()
corr_pairs = correlation_mat.unstack()
corr_pairs.head()
#print(corr_pairs)
```

```
name  name        1.000000
      rating      0.143938
      genre       0.036367
      year        0.965761
      released    0.959015
dtype: float64
```

9. Pairing and sorting correlation pairs one to one.

```
#j) Sorting correlation pairs one to one.
sorted_pairs = corr_pairs.sort_values(kind="quicksort")
sorted_pairs.head()
#print(sorted_pairs)
```

```
budget   Match     -0.145122
Match    budget    -0.145122
company  budget    -0.092249
budget   company   -0.092249
rating   Match     -0.089590
dtype: float64
```

10. Sorting correlation pairs which are significantly correlated (more than 0.5)

```
# k) Sorting correlation pairs one to one that have a high correlation (> 0.5)

strong_pairs = sorted_pairs[abs(sorted_pairs) > 0.5]
strong_pairs.head()
#print(strong_pairs)
```

```
star     company     0.527116
company  star        0.527116
         writer      0.546151
writer   company     0.546151
company  director    0.552258
dtype: float64
```

11. Looking at top 15 companies by gross revenue.

```
# l) Looking at the top 15 companies by gross revenue (company)

CompanyGrossSum = df.groupby('company')[["gross"]].sum()
CompanyGrossSumSorted = CompanyGrossSum.sort_values('gross', ascending = False)[:15]
CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')
CompanyGrossSumSorted
```

```
company
Warner Bros.                 56491421806
Universal Pictures           52514188890
Columbia Pictures            43008941346
Paramount Pictures           40493607415
Twentieth Century Fox        40257053857
Walt Disney Pictures         36327887792
New Line Cinema              19883797684
Marvel Studios               15065592411
DreamWorks Animation         11873612858
Touchstone Pictures          11795832638
Dreamworks Pictures          11635441081
Metro-Goldwyn-Mayer (MGM)     9230230105
Summit Entertainment          8373718838
Pixar Animation Studios       7886344526
Fox 2000 Pictures             7443502667
Name: gross, dtype: int64
```

12. Looking at top 15 companies by gross revenue (Company and Year).

```
# m) Looking at the top 15 companies by gross revenue (company and year)
CompanyGrossSum = df.groupby(['company', 'year'])[["gross"]].sum()
CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company','year'], ascending = False)[:15]
CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')
CompanyGrossSumSorted
```

```
company               year
Walt Disney Pictures  2019    5773131804
Marvel Studios        2018    4018631866
Universal Pictures    2015    3834354888
Twentieth Century Fox 2009    3793491246
Walt Disney Pictures  2017    3789382071
Paramount Pictures    2011    3565705182
Warner Bros.          2010    3300479986
                      2011    3223799224
Walt Disney Pictures  2010    3104474158
Paramount Pictures    2014    3071298586
Columbia Pictures     2006    2934631933
                      2019    2932757449
Marvel Studios        2019    2797501328
Warner Bros.          2018    2774168962
Columbia Pictures     2011    2738363306
Name: gross, dtype: int64
```

14. Creating a numeric representation of Company for further interpretations. So, converting it from a string to a numeric data type to input it into the correlation. Instead of numerizing just company alone, all columns are Numerized. Foreloop used.

```
# p) Encoding categorical variables in df_numerized df

df_numerized = df

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

df_numerized.head()
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	company	runtime	YearCorrect	Match
0	6587	6	6	1980	1705	8.4	927000.0	2589	4014	1047	54	19000000.0	46998772.0	2319	146.0	1980.0	True
1	5573	6	1	1980	1492	5.8	65000.0	2269	1632	327	55	4500000.0	58853106.0	731	104.0	1980.0	True
2	5142	4	0	1980	1771	8.7	1200000.0	1111	2567	1745	55	18000000.0	538375067.0	1540	124.0	1980.0	True
3	286	4	4	1980	1492	7.7	221000.0	1301	2000	2246	55	3500000.0	83453539.0	1812	88.0	1980.0	True
4	1027	6	4	1980	1543	7.3	108000.0	1054	521	410	55	6000000.0	39846344.0	1777	98.0	1980.0	True

It left already Numerized Data types and focused more on string values.

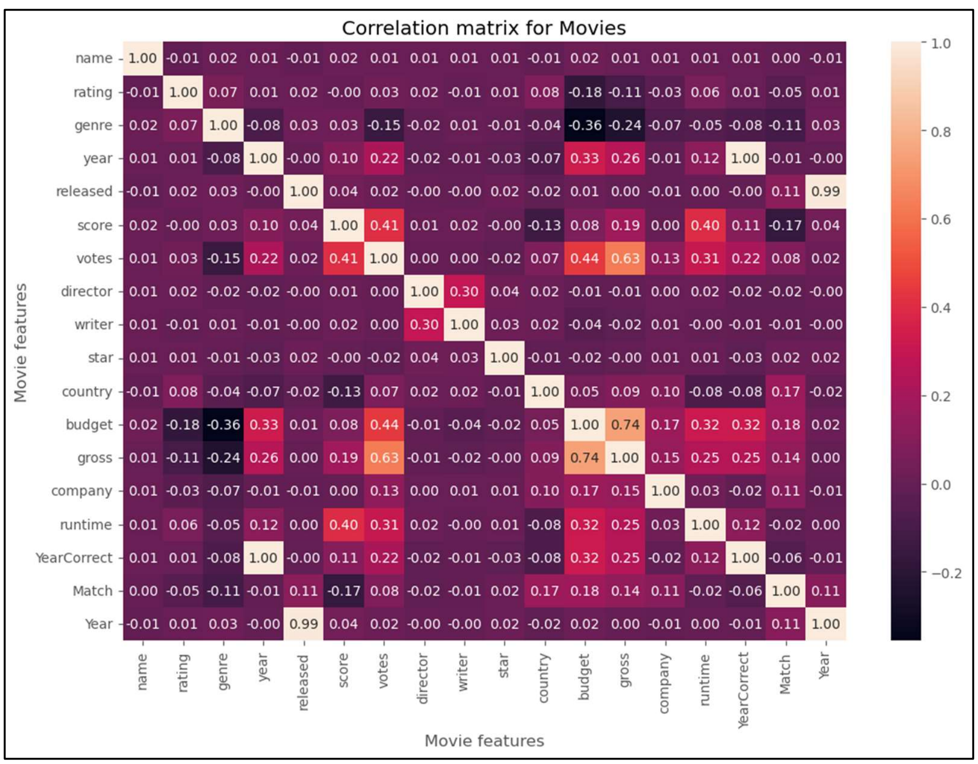
15. Creating a df.numerized correlation matrix

```python
# q) Correlation in df numerized

df_numerized.corr(method='pearson')
```

	name	rating	genre	year	released	score	votes	director	writer	star	country	budget	gross	compar
name	1.000000	-0.008069	0.016355	0.011453	-0.011311	0.017097	0.013088	0.009079	0.009081	0.006472	-0.010737	0.023970	0.005533	0.0092
rating	-0.008069	1.000000	0.072423	0.008779	0.016613	-0.001314	0.033225	0.019483	-0.005921	0.013405	0.081244	-0.176002	-0.107339	-0.0329
genre	0.016355	0.072423	1.000000	-0.081261	0.029822	0.027965	-0.145307	-0.015258	0.006567	-0.005477	-0.037615	-0.356564	-0.235650	-0.0710
year	0.011453	0.008779	-0.081261	1.000000	-0.000695	0.097995	0.222945	-0.020795	-0.008656	-0.027242	-0.070938	0.329321	0.257486	-0.0104
released	-0.011311	0.016613	0.029822	-0.000695	1.000000	0.042788	0.016097	-0.001478	-0.002404	0.015777	-0.020427	0.014683	0.001659	-0.0104
score	0.017097	-0.001314	0.027965	0.097995	0.042788	1.000000	0.409182	0.009559	0.019416	-0.001609	-0.133348	0.076254	0.186258	0.0010
votes	0.013088	0.033225	-0.145307	0.222945	0.016097	0.409182	1.000000	0.000260	0.000892	-0.019282	0.073625	0.442429	0.630757	0.1332
director	0.009079	0.019483	-0.015258	-0.020795	-0.001478	0.009559	0.000260	1.000000	0.299067	0.039234	0.017490	-0.012272	-0.014441	0.0044
writer	0.009081	-0.005921	0.006567	-0.008656	-0.002404	0.019416	0.000892	0.299067	1.000000	0.027245	0.015343	-0.039451	-0.023519	0.0056
star	0.006472	0.013405	-0.005477	-0.027242	0.015777	-0.001609	-0.019282	0.039234	0.027245	1.000000	-0.012998	-0.019589	-0.002717	0.0124
country	-0.010737	0.081244	-0.037615	-0.070938	-0.020427	-0.133348	0.073625	0.017490	0.015343	-0.012998	1.000000	0.054063	0.092129	0.0955
budget	0.023970	-0.176002	-0.356564	0.329321	0.014683	0.076254	0.442429	-0.012272	-0.039451	-0.019589	0.054063	1.000000	0.740395	0.1732
gross	0.005533	-0.107339	-0.235650	0.257486	0.001659	0.186258	0.630757	-0.014441	-0.023519	-0.002717	0.092129	0.740395	1.000000	0.1548
company	0.009211	-0.032943	-0.071067	-0.010431	-0.010474	0.001030	0.133204	0.004404	0.005646	0.012442	0.095548	0.173214	0.154840	1.0000
runtime	0.010392	0.062145	-0.052711	0.120811	0.000868	0.399451	0.309212	0.017624	-0.003511	0.010174	-0.078412	0.320447	0.245216	0.0344
YearCorrect	0.010699	0.006741	-0.077911	0.997415	-0.004644	0.105994	0.218429	-0.020422	-0.008611	-0.027611	-0.080844	0.321918	0.250514	-0.0151
Match	0.003614	-0.046430	-0.110644	-0.012163	0.110840	-0.165669	0.080450	-0.019467	-0.008093	0.019237	0.170358	0.180592	0.144928	0.1057
Year	-0.011725	0.013475	0.028397	-0.001562	0.993694	0.040993	0.017337	-0.000105	-0.002892	0.015406	-0.022277	0.015682	0.002946	-0.0107

16. Correlation plot of df.numerized.

```python
# r) Correlation matrix of df_numerized

correlation_matrix = df_numerized.corr(method='pearson')
sns.heatmap(correlation_matrix, annot = True, fmt=".2f")
plt.title("Correlation matrix for Movies")
plt.xlabel("Movie features")
plt.ylabel("Movie features")
plt.show()
```



Observations:

Strong Positive Correlations:

- Gross and Budget: A moderate positive correlation (0.74) suggests that higher budget movies tend to have higher gross revenue.
- Votes and Score: A moderate positive correlation (0.44) indicates that movies with more votes tend to have higher scores.
- Runtime and YearCorrect: A weak positive correlation (0.12) suggest a slight tendency for movies to be longer in recent years.
- Gross and votes: A moderate positive correlation (0.64) suggests that higher voted movies tend to have higher gross revenue.

Strong Negative Correlations:

- Genre and Runtime: A moderate negative correlation (-0.36) suggests that certain genres tend to have shorter runtimes.

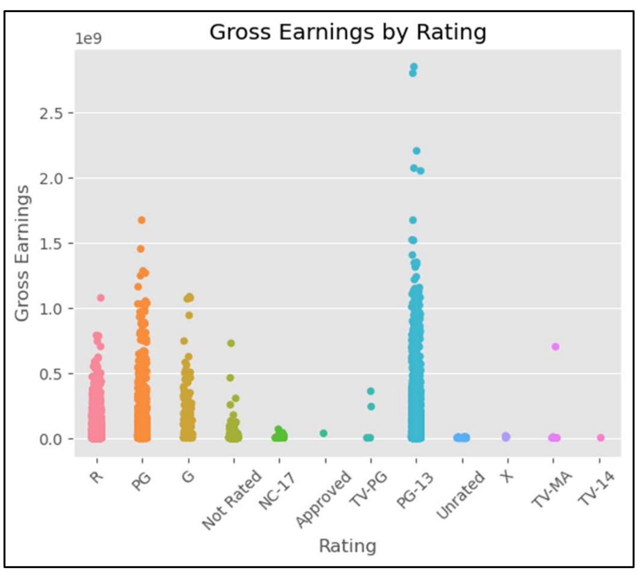
Other Correlations:

- Director, Writer, and Star with Gross: Moderate positive correlations (0.3, 0.3, 0.25 respectively) suggest that movies with well-known directors, writers, or stars tend to have higher gross revenue.
- Country and Budget: A weak positive correlation (0.05) suggests a slight tendency for movies from certain countries to have higher budgets.

Potential Insights and Further Analysis:

- Feature Importance for Predicting Gross Revenue: Budget, director, writer, and star seem to be important predictors of gross revenue.

17. Strip plot visualizing the distribution of gross earnings across different movie ratings.



Observations:

1. Variation in Gross Earnings: There is significant variation in gross earnings within each rating category. Some movies earn significantly more than others within the same rating.
2. Dominant Ratings: The ratings with the highest number of movies and the widest range of gross earnings are PG-13, PG, and R.
3. High-Grossing Ratings: PG-13 and PG-rated movies tend to have the highest gross earnings overall, with some films reaching over $2 billion.
4. Lower-Grossing Ratings: Ratings like G, Not Rated, and NC-17 have fewer movies and generally lower gross earnings.
5. Outliers: There are a few outliers in some rating categories, such as very high-grossing movies in the PG-13 category and extremely low-grossing movies in the R category.