Download the dataset from: https://github.com/bellawillrise/Introduction-to-Numerical-Computing-in-Python/

Submit a pdf file, which is a rendered saved version of the jupyter notebook. Make sure to execute all the codes so the output can be viewed in the pdf.

Also include the link to the public github repository where the jupyter notebook for the assignment is uploaded.

Link to the github repository: <</insert link>>

In [52]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [53]:
```
# %matplotlib inline
```

In [54]:
```
data = pd.read_csv("data/movie_metadata_cleaned.csv")
```

In [55]:
```
data.head(2)
```

Out[55]:

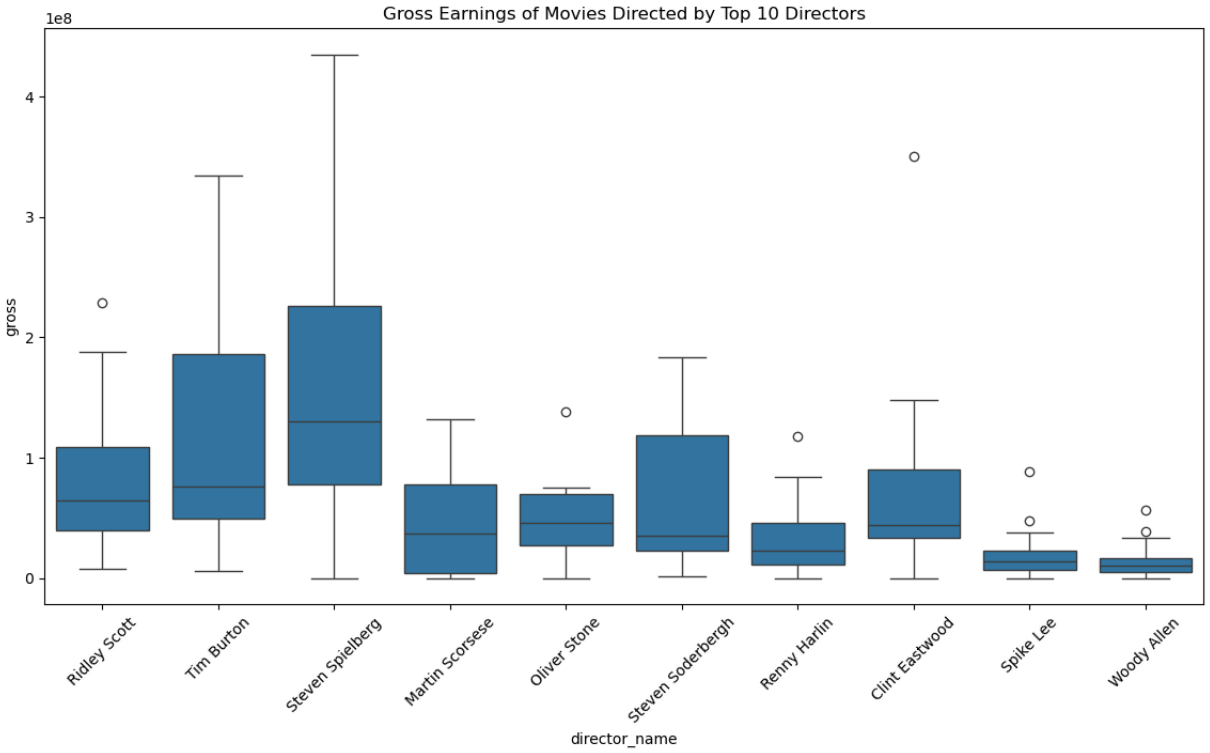
	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	

2 rows × 29 columns

Get the top 10 directors with most movies directed and use a boxplot for their gross earnings

In [87]:
```
# Get all directors name
top_directors = data['director_name']

# Filter out director with no name
top_directors = top_directors[top_directors != '0']

# Get the top 10 directors
top_directors = top_directors.value_counts().nlargest(10).index
```

```
# Get the data of the top 10 directors
top_directors_data = data[data['director_name'].isin(top_directors)]

# Create the boxplot
plt.figure(figsize=(14, 7))
sns.boxplot(x='director_name', y='gross', data=top_directors_data)
plt.title('Gross Earnings of Movies Directed by Top 10 Directors')
plt.xticks(rotation=45)
plt.show()
```

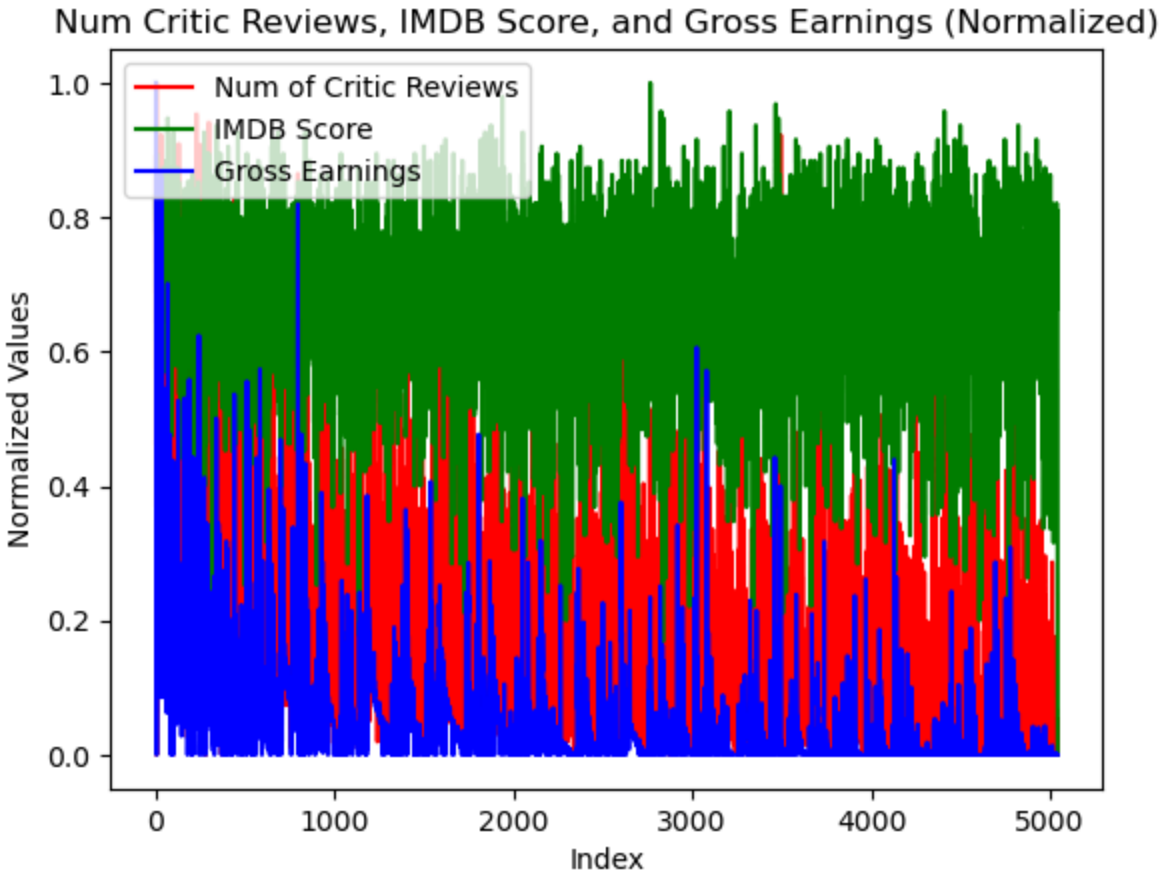


Plot the following variables in one graph:

- num_critic_for_reviews
- IMDB score
- gross

In [85]:
```
# Create an array for X-axis
X = np.arange(len(data))

# Normalize the variables (0 to 1 scale)
y = data['num_critic_for_reviews'] / data['num_critic_for_reviews'].max()
z = data['imdb_score'] / data['imdb_score'].max()
w = data['gross'] / data['gross'].max()

# Plotting
plt.plot(X, y, color='r', label='Num of Critic Reviews')
plt.plot(X, z, color='g', label='IMDB Score')
plt.plot(X, w, color='b', label='Gross Earnings')

# Naming the x-axis, y-axis, and the whole graph
```

```
plt.xlabel("Index")
plt.ylabel("Normalized Values")
plt.title("Num Critic Reviews, IMDB Score, and Gross Earnings (Normalized)")

# Legend
plt.legend()

# To load the display window
plt.show()
```



Compute Sales (Gross - Budget), add it as another column

In [83]:
```
data['sales'] = data['gross'] - data['budget']
data['sales'].head()
```

Out[83]:
```
0    523505847.0
1      9404152.0
2    -44925825.0
3    198130642.0
4            0.0
Name: sales, dtype: float64
```

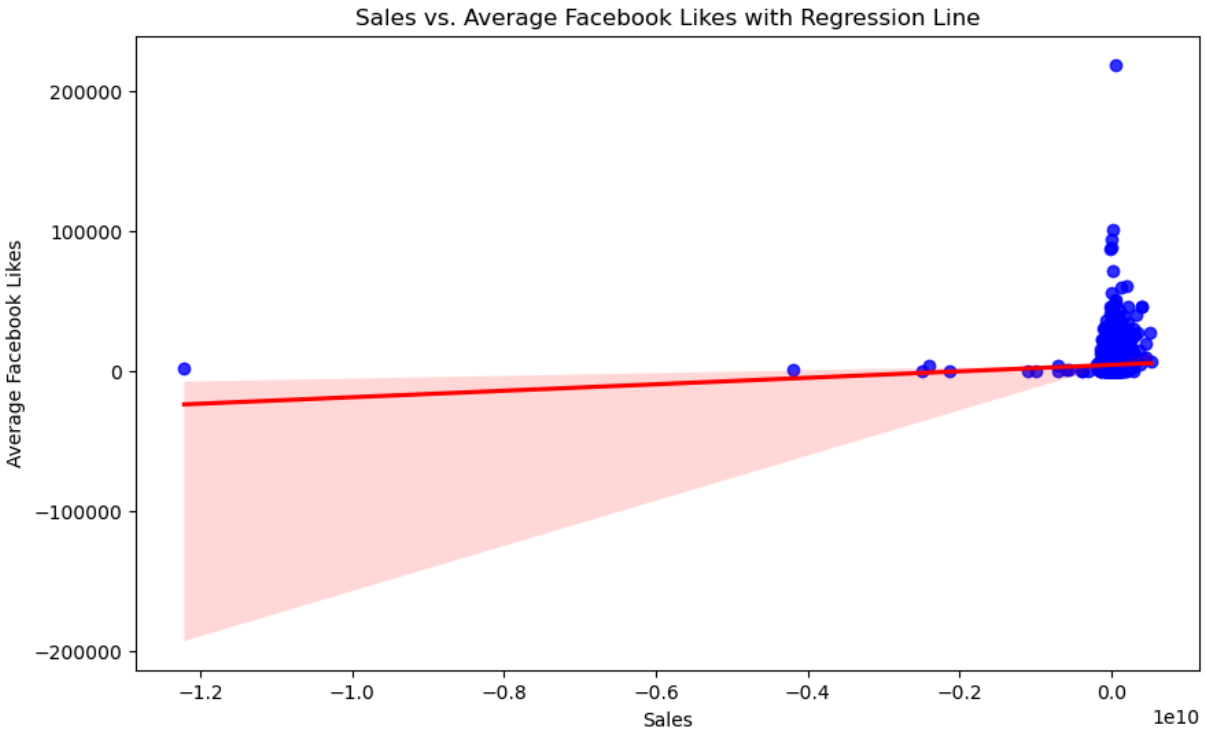
Which directors garnered the most total sales?

In [79]:
```
# Group by director and sum the sales
director_sales = data.groupby('director_name')['sales'].sum()

# Sort the results in descending order to find the directors with the most total sa
top_directors_by_sales = director_sales.sort_values(ascending=False)

top_directors_by_sales.head()
```

Out[79]:
```
director_name
Steven Spielberg    2.451332e+09
George Lucas        1.386641e+09
James Cameron       1.199626e+09
Joss Whedon         1.000887e+09
Chris Columbus      9.417076e+08
Name: sales, dtype: float64
```

Plot sales and average likes as a scatterplot. Fit it with a line.

In [81]:
```
# Calculate and plot the scatterplot with regression line directly
plt.figure(figsize=(10, 6))
sns.regplot(
    x=data['sales'],
    y=data[['director_facebook_likes',
            'actor_1_facebook_likes',
            'actor_2_facebook_likes',
            'actor_3_facebook_likes',
            'cast_total_facebook_likes',
            'movie_facebook_likes']].mean(axis=1),
    scatter_kws={'color': 'blue'},
    line_kws={'color': 'red'}
)

# Labels and title
plt.xlabel('Sales')
plt.ylabel('Average Facebook Likes')
plt.title('Sales vs. Average Facebook Likes with Regression Line')

# Show plot
plt.show()
```

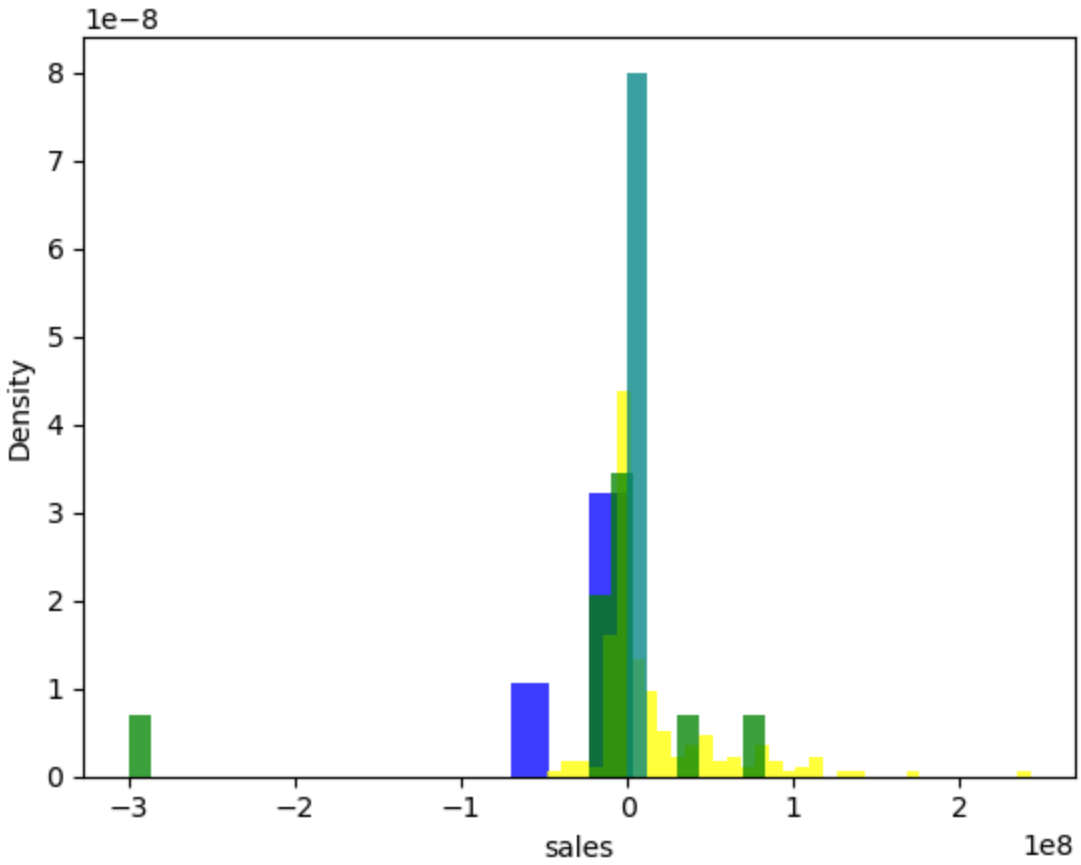


Which of these genres are the most profitable? Plot their sales using different histograms, superimposed in the same axis.

- Romance
- Comedy
- Action
- Fantasy

In [95]:
```
# Histogram for the 'Romance' genre sales
sns.histplot(data[data['genres'] == 'Romance']["sales"], color="blue", label="Roman

# Histogram for the 'Comedy' genre sales
sns.histplot(data[data['genres'] == 'Comedy']["sales"], color="yellow", label="Come

# Histogram for the 'Action' genre sales
sns.histplot(data[data['genres'] == 'Action']["sales"], color="green", label="Actio

# Histogram for the 'Fantasy' genre sales
sns.histplot(data[data['genres'] == 'Fantasy']["sales"], color="teal", label="Fanta

# Legend
ax.legend()
```

Out[95]:
```
<matplotlib.legend.Legend at 0x1e1936a3260>
```



For each of movie, compute average likes of the three actors and store it as a new variable

Read up on the mean function.

Store it as a new column, average_actor_likes.

In [77]:
```
# Compute the average likes for the three actors and store it in a new column
data['average_actor_likes'] = data[['actor_1_facebook_likes', 'actor_2_facebook_lik

data['average_actor_likes']
```

Out[77]:
```
0         930.333333
1       15333.333333
2        3851.333333
3       24333.333333
4          47.666667
             ...
5039      584.333333
5040        0.000000
5041      718.000000
5042       41.666667
5043        0.000000
Name: average_actor_likes, Length: 5044, dtype: float64
```

Copying the whole dataframe

In [75]:
```
df = data.copy()
df.head()
```

Out[75]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	

5 rows × 30 columns

Min-Max Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

The min-max approach (often called normalization) rescales the feature to a hard and fast range of [0,1] by subtracting the minimum value of the feature then dividing by the range. We can apply the min-max scaling in Pandas using the .min() and .max() methods.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Normalize each numeric column (those that have types integer or float) of the copied dataframe (df)

In [73]:
```
# Select numeric columns (excluding categorical and object types)
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Min-Max Normalization
df[numeric_columns] = (df[numeric_columns] - df[numeric_columns].min()) / (df[numer

# Display
df.head()
```

Out[73]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_
0	0.000000	b'Avatar'	Color	James Cameron	0.889299	0.941799	
1	0.000198	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	0.371464	0.894180	
2	0.000397	b'Spectre'	Color	Sam Mendes	0.740467	0.783069	
3	0.000595	b'The Dark Knight Rises'	Color	Christopher Nolan	1.000000	0.867725	
4	0.000793	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.000000	0.000000	

5 rows × 31 columns

In [ ]: