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: https://github.com/ainzzcutie/CMSC-197---Machine-Learning.git

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
# %matplotlib inline
```

In [9]:
```
data = pd.read_csv("movie_metadata_cleaned.csv")
```

In [11]:
```
data.head(2)
```

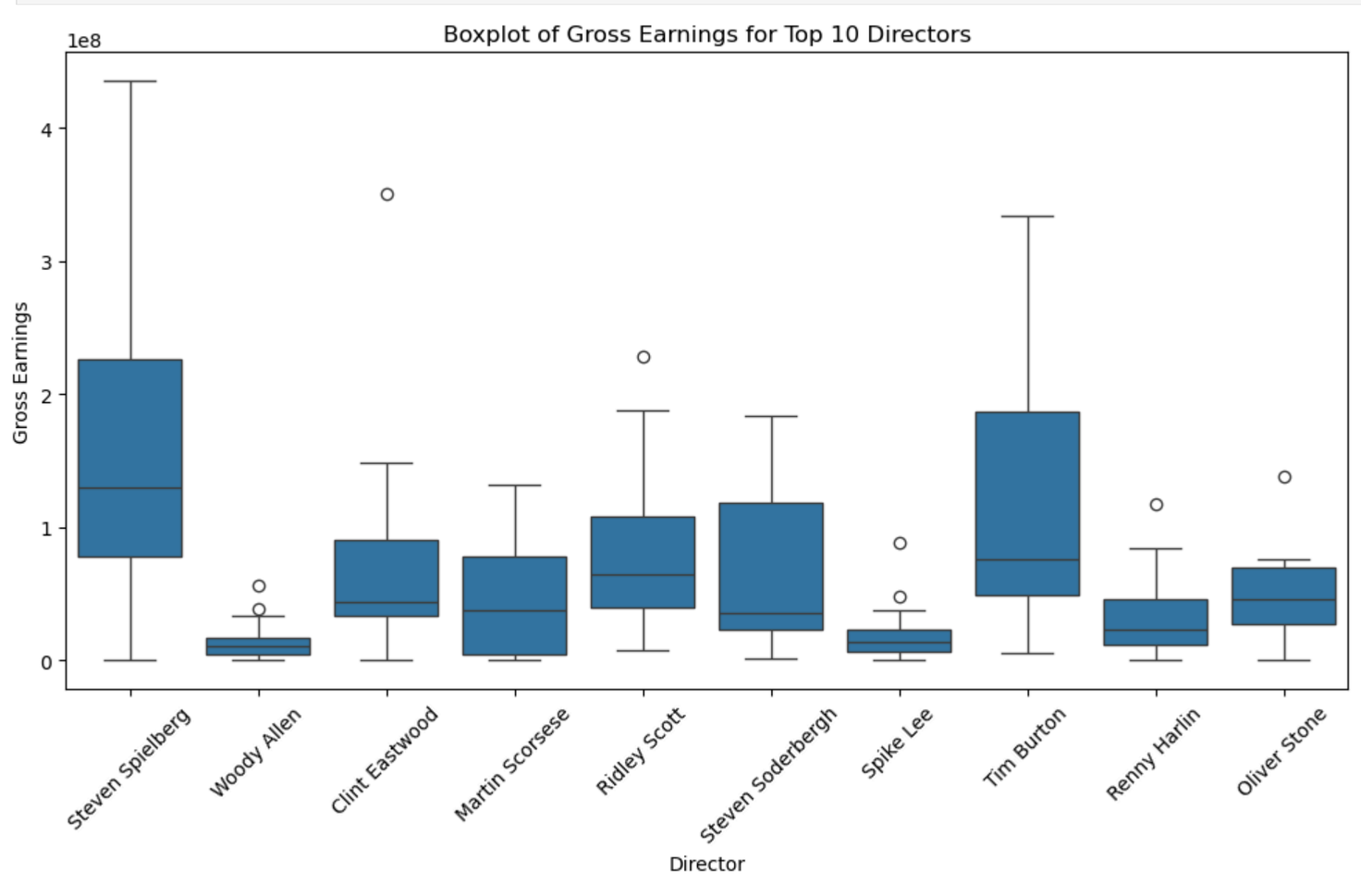
Out[11]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	num_user_for_reviews	language	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	3054.0	English	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000.0
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	...	1238.0	English	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0.0

2 rows × 29 columns

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

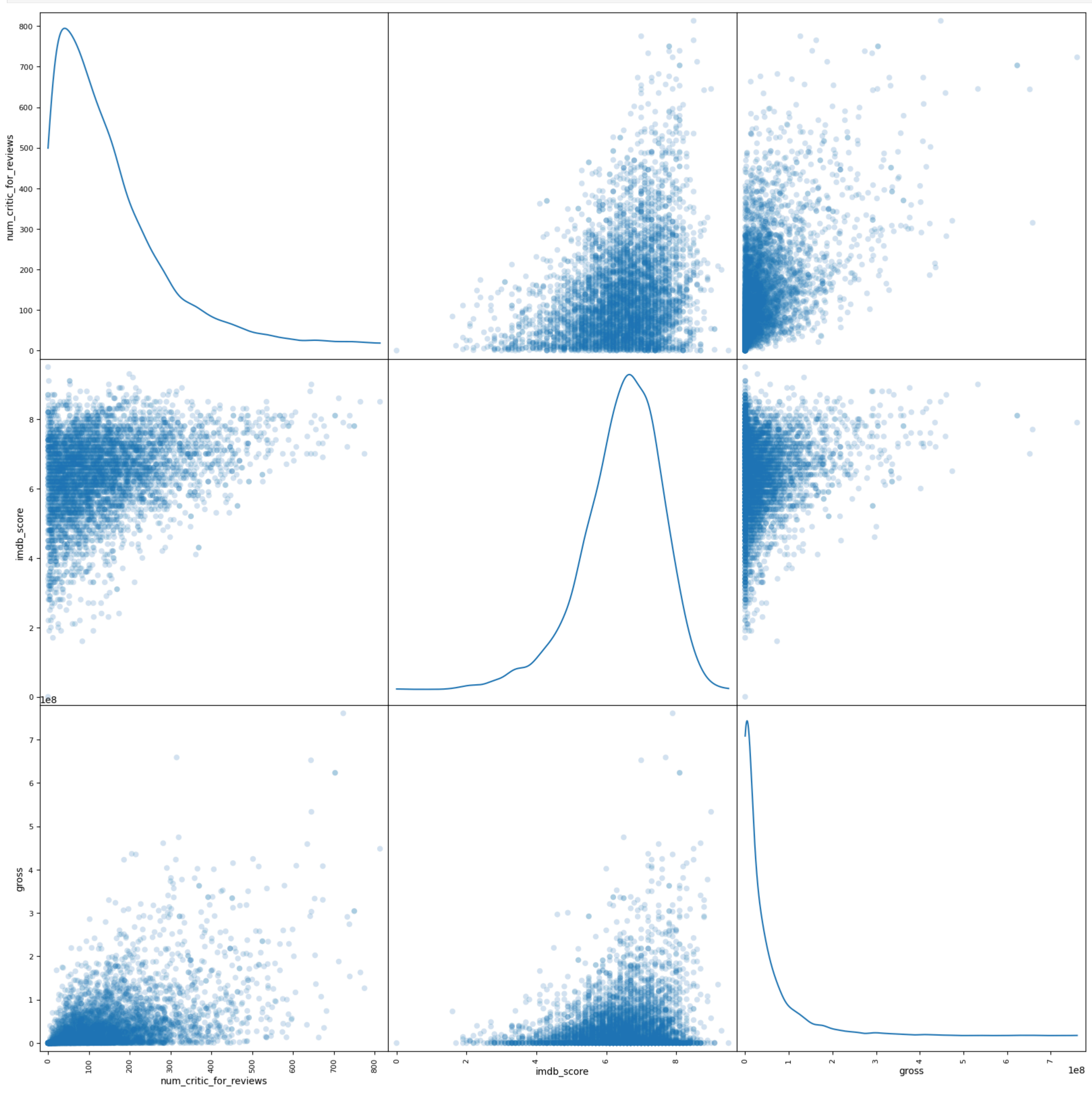
In [15]:
```
director_counts = data.groupby('director_name').size().reset_index(name='movie_count')
director_counts = director_counts.sort_values(by='movie_count', ascending=False)

# Filter to exclude director_name = '0'
director_counts = director_counts[director_counts['director_name'] != '0']
top_10_directors = director_counts[:10]

# Filter to only include the movies made by the top 10 directors
top_directors_movies = data[data['director_name'].isin(top_10_directors['director_name'])]

# boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(x='director_name', y='gross', data=top_directors_movies, order=top_10_directors['director_name'])
plt.xticks(rotation=45)
plt.title('Boxplot of Gross Earnings for Top 10 Directors')
plt.xlabel('Director')
plt.ylabel('Gross Earnings')
plt.show()
```



Plot the following variables in one graph:

- num_critic_for_reviews
- IMDB score
- gross

In [17]:
```
from pandas.plotting import scatter_matrix

# Define the columns to be included in the scatter matrix
cols = ["num_critic_for_reviews", "imdb_score", "gross"]

# Create a scatter matrix for the specified columns in the data
scatter_matrix(data[cols], alpha=0.2, figsize=(20, 20),
               diagonal='kde', marker='o')

# Display the scatter matrix plot
plt.show()
```



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

In [19]:
```
# Calculate the 'sales' by subtracting 'budget' from 'gross'
data['sales'] = data['gross'] - data['budget']

# Display the data frame
data
```

Out[19]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	language	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	sales
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	English	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000.0	523505847.0
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	...	English	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0.0	9404152.0
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	English	UK	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000.0	-44925825.0
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	English	USA	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000.0	198130642.0
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	131.0	0.0	Rob Walker	131.0	...	0	0	0	0.0	0.0	12.0	7.1	0.00	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5039	5039	b'The Following '	Color	0	43.0	43.0	0.0	319.0	Valorie Curry	841.0	...	English	USA	TV-14	0.0	0.0	593.0	7.5	16.00	32000.0	0.0
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	...	English	USA	0	1400.0	2013.0	0.0	6.3	0.00	16.0	-1400.0
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	...	English	USA	PG-13	0.0	2012.0	719.0	6.3	2.35	660.0	10443.0
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	...	English	USA	PG	1100.0	2004.0	23.0	6.6	1.85	456.0	84122.0
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	0.0	0.0	Toni Gonzaga	0.0	...	0	Philippines	PG	0.0	2014.0	0.0	0.0	0.00	0.0	0.0

5044 rows × 30 columns

Which directors garnered the most total sales?

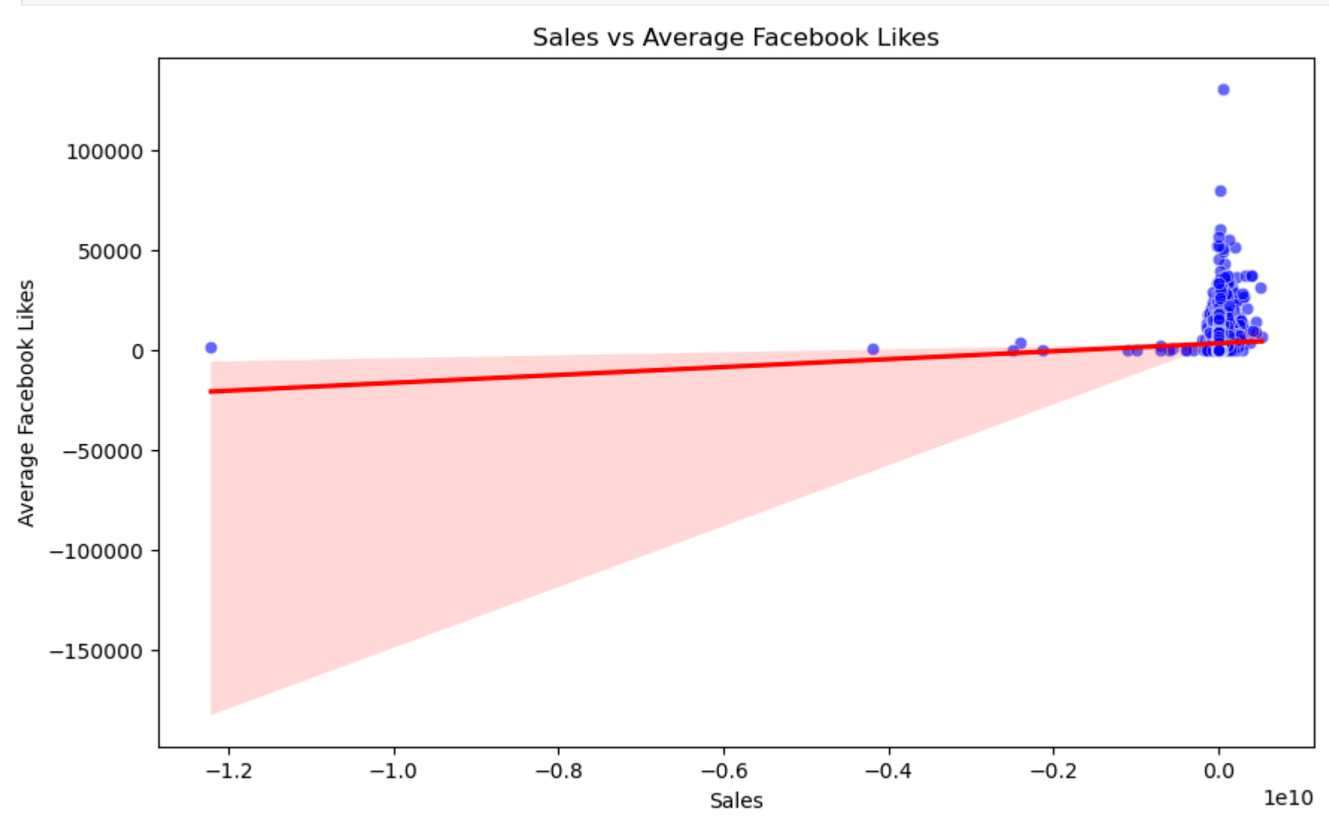
In [21]:
```
# Group the data by 'director_name' and calculate the total sales for each director
total_sales = data.groupby('director_name')['sales'].sum().reset_index()

# Sort in descending order
total_sales = total_sales.sort_values(by='sales', ascending=False)

# slice to get top 10 directors' sales data
top_directors_sales = total_sales[:10]

# Display the top directors' sales data
top_directors_sales
```

Out[21]:

	director_name	sales
2159	Steven Spielberg	2.451332e+09
765	George Lucas	1.386641e+09
923	James Cameron	1.199626e+09
1219	Joss Whedon	1.000887e+09
335	Chris Columbus	9.417076e+08
1787	Peter Jackson	9.009693e+08
2221	Tim Burton	8.242755e+08
374	Christopher Nolan	8.082276e+08
1158	Jon Favreau	7.693815e+08
695	Francis Lawrence	7.555020e+08

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

In [23]:
```
# Create a scatter plot of sales vs average facebook likes
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sales',
                y=data[['director_facebook_likes',
                        'actor_1_facebook_likes',
                        'actor_2_facebook_likes',
                        'actor_3_facebook_likes',
                        'movie_facebook_likes']].mean(axis=1),
                data=data, color='blue', alpha=0.6)

# Fit a line to the scatter plot
sns.regplot(x='sales',
            y=data[['director_facebook_likes',
                    'actor_1_facebook_likes',
                    'actor_2_facebook_likes',
                    'actor_3_facebook_likes',
                    'movie_facebook_likes']].mean(axis=1),
            data=data, scatter=False, color='red')

# Adding titles and labels
plt.title('Sales vs Average Facebook Likes')
plt.xlabel('Sales')
plt.ylabel('Average Facebook Likes')

# Show the plot
plt.show()
```

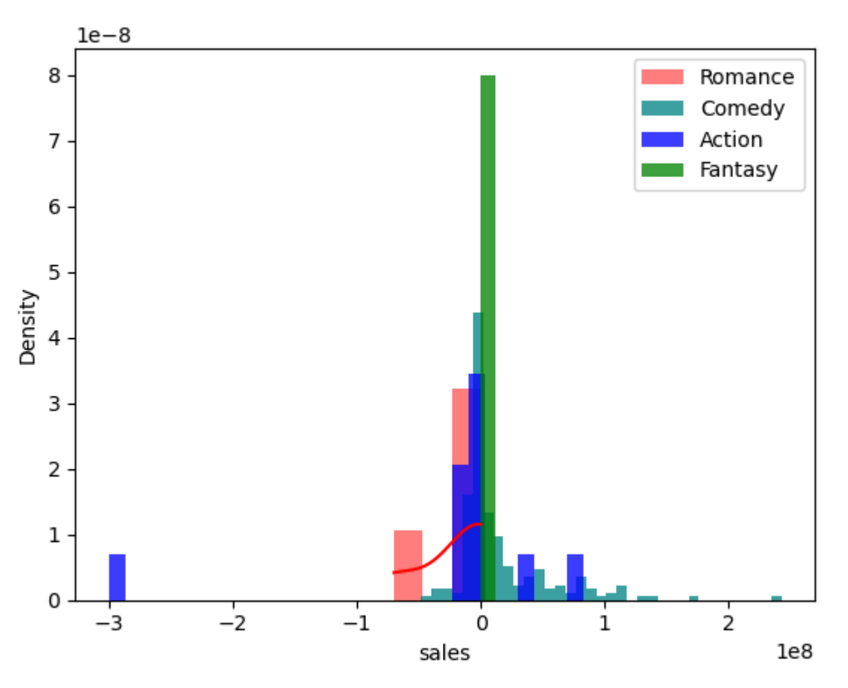


Which of these genres are the most profitable? Plot their sales using different histograms, superimposed in the same axis.

- Romance
- Comedy
- Action
- Fantasy

In [25]:
```
# Create a histogram for the 'Romance' genre sales
ax = sns.histplot(data[data['genres'] == 'Romance']["sales"], color="red", label="Romance", kde=True, stat="density", linewidth=0)

# Create a histogram for the 'Comedy' genre sales
sns.histplot(data[data['genres'] == 'Comedy']["sales"], color="teal", label="Comedy", kde=False, stat="density", linewidth=0)

# Create a histogram for the 'Action' genre sales
sns.histplot(data[data['genres'] == 'Action']["sales"], color="blue", label="Action", kde=False, stat="density", linewidth=0)

# Create a histogram for the 'Fantasy' genre sales
sns.histplot(data[data['genres'] == 'Fantasy']["sales"], color="green", label="Fantasy", kde=False, stat="density", linewidth=0)

# Add a legend to the plot to identify the genres
ax.legend()
```

Out[25]: <matplotlib.legend.Legend at 0x279ad8b4320>



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 [27]:
```
# Calculate the average Facebook likes for the three actors in each movie
data['average_actor_likes'] = data[['actor_1_facebook_likes', 'actor_2_facebook_likes', 'actor_3_facebook_likes']].mean(axis=1)

# Display the updated data frame with the new 'average_actor_likes' column
data
```

Out[27]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	sales	average_actor_likes
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000.0	523505847.0	930.333333
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	...	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0.0	9404152.0	15333.333333
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	UK	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000.0	-44925825.0	3851.333333
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	USA	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000.0	198130642.0	24333.333333
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	131.0	0.0	Rob Walker	131.0	...	0	0	0.0	0.0	12.0	7.1	0.00	0.0	0.0	47.666667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5039	5039	b'The Following '	Color	0	43.0	43.0	0.0	319.0	Valorie Curry	841.0	...	USA	TV-14	0.0	0.0	593.0	7.5	16.00	32000.0	0.0	584.333333
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	...	USA	0	1400.0	2013.0	0.0	6.3	0.00	16.0	-1400.0	0.000000
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	...	USA	PG-13	0.0	2012.0	719.0	6.3	2.35	660.0	10443.0	718.000000
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	...	USA	PG	1100.0	2004.0	23.0	6.6	1.85	456.0	84122.0	41.666667
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	0.0	0.0	Toni Gonzaga	0.0	...	Philippines	PG	0.0	2014.0	0.0	0.0	0.00	0.0	0.0	0.000000

5044 rows × 31 columns

Copying the whole dataframe

In [31]:
```
df = data.copy()
df.head()
```

Out[31]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	sales	average_actor_likes
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000.0	523505847.0	930.333333
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	...	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0.0	9404152.0	15333.333333
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	UK	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000.0	-44925825.0	3851.333333
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	USA	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000.0	198130642.0	24333.333333
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	131.0	0.0	Rob Walker	131.0	...	0	0	0.0	0.0	12.0	7.1	0.00	0.0	0.0	47.666667

5 rows × 31 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 [33]:
```
# Identify numeric columns (integer or float)
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

# Initialize a data frame to hold the normalized values
normalized_df = df.copy()

# Loop through each numeric column
for column in numeric_columns:
    # Calculate the minimum and maximum values for the current column
    min_value = df[column].min()
    max_value = df[column].max()

    # Calculate the range of the current column
    range_value = max_value - min_value

    # Normalize the column using min-max approach
    normalized_df[column] = (df[column] - min_value) / range_value

# Display the data frame with normalized values
normalized_df
```

Out[33]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	sales	average_actor_likes
0	0.000000	b'Avatar'	Color	James Cameron	0.889299	0.941799	0.000000	0.037174	Joel David Moore	0.001563	...	USA	PG-13	1.940158e-02	0.996528	0.006832	0.831579	0.111250	0.094556	1.000000	0.004261
1	0.000198	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	0.371464	0.894180	0.024478	0.043478	Orlando Bloom	0.062500	...	USA	PG-13	2.455896e-02	0.995536	0.036496	0.747368	0.146875	0.000000	0.959637	0.070229
2	0.000397	b'Spectre'	Color	Sam Mendes	0.740467	0.783069	0.000000	0.007000	Rory Kinnear	0.017188	...	UK	PG-13	2.005649e-02	0.999504	0.002869	0.715789	0.146875	0.243553	0.955371	0.017640
3	0.000595	b'The Dark Knight Rises'	Color	Christopher Nolan	1.000000	0.867725	0.956522	1.000000	Christian Bale	0.042188	...	USA	PG-13	2.046580e-02	0.998016	0.167883	0.894737	0.146875	0.469914	0.974454	0.111450
4	0.000793	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.000000	0.000000	0.005696	0.000000	Rob Walker	0.000205	...	0	0	0.000000e+00	0.000000	0.000088	0.747368	0.000000	0.000000	0.958898	0.000218
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5039	0.999207	b'The Following '	Color	0	0.052891	0.227513	0.000000	0.013870	Valorie Curry	0.001314	...	USA	TV-14	0.000000e+00	0.000000	0.004328	0.789474	1.000000	0.091691	0.958898	0.002676
5040	0.999405	b'A Plague So Pleasant'	Color	Benjamin Roberds	0.015990	0.402116	0.000000	0.000000	Maxwell Moody	0.000000	...	USA	0	1.146085e-07	0.998512	0.000000	0.663158	0.000000	0.000046	0.958898	0.000000
5041	0.999603	b'Shanghai Calling'	Color	Daniel Hsia	0.017220	0.529101	0.000000	0.021261	Daniel Henney	0.001478	...	USA	PG-13	0.000000e+00	0.998016	0.005248	0.663158	0.146875	0.001891	0.958899	0.003289
5042	0.999802	b'My Date with Drew'	Color	Jon Gunn	0.052891	0.476190	0.000696	0.000696	Brian Herzlinger	0.000134	...	USA	PG	9.004953e-08	0.994048	0.000168	0.694737	0.115625	0.001307	0.958905	0.000191
5043	1.000000	b'Starting Over Again'	0	Olivia Lamasan	0.000000	0.000000	0.000000	0.000000	Toni Gonzaga	0.000000	...	Philippines	PG	0.000000e+00	0.999008	0.000000	0.000000	0.000000	0.000000	0.958898	0.000000

5044 rows × 31 columns

In [ ]: