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 [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

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

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("data/movie_metadata_cleaned.csv")
data.head(3)
```

Out[2]:

	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
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	3054.0	English
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
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	994.0	English

3 rows × 29 columns

In [7]:
```
data = pd.read_csv("data/movie_metadata_cleaned.csv")
data.head(2)
```

Out[7]:

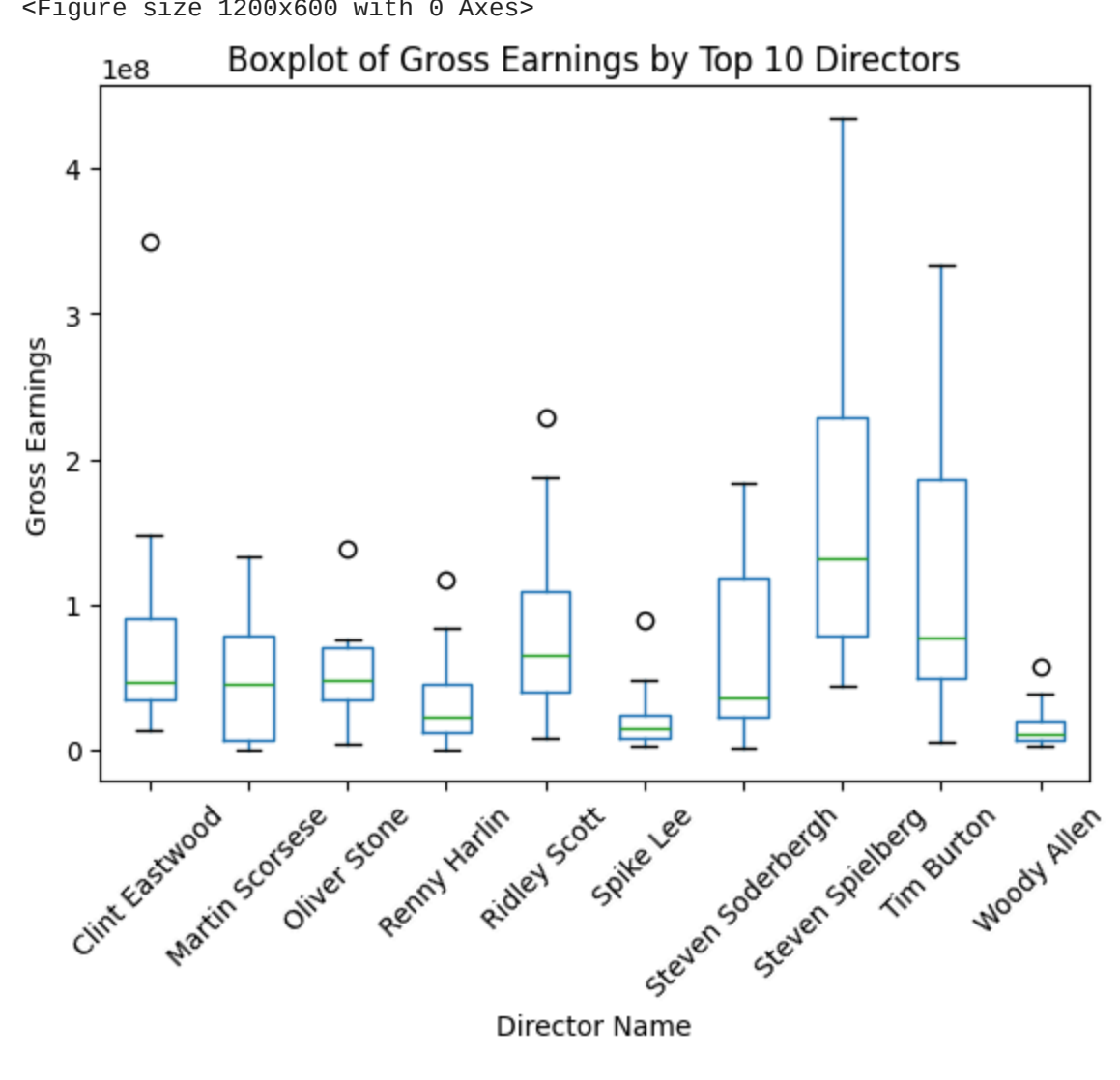
	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	num_user_for_reviews	language
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	3054.0	English
1	1	b"Pirates 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

2 rows × 29 columns

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

In [8]:
```
# Filter where the director_name is 0
data = data[data['director_name'] != '0']

# Top 10 directors with the most movies directed
top_directors = data['director_name'].value_counts().nlargest(10).index

# Filter top 10 directors
top_directors_data = data[data['director_name'].isin(top_directors)]

# Filter out rows where the gross is 0 or NaN
top_directors_data_without0 = top_directors_data[top_directors_data['gross'].notnull() & (top_directors_data['gross'] != 0)]

# Create a boxplot for the gross earnings of movies by these top 10 directors using Matplotlib
plt.figure(figsize=(12, 6))
top_directors_data_without0.boxplot(column='gross', by='director_name', grid=False)
plt.title('Boxplot of Gross Earnings by Top 10 Directors')
plt.suptitle('')  # Remove the automatic 'Boxplot grouped by' title
plt.xlabel('Director Name')
plt.ylabel('Gross Earnings')
plt.xticks(rotation=45)
plt.show()
```

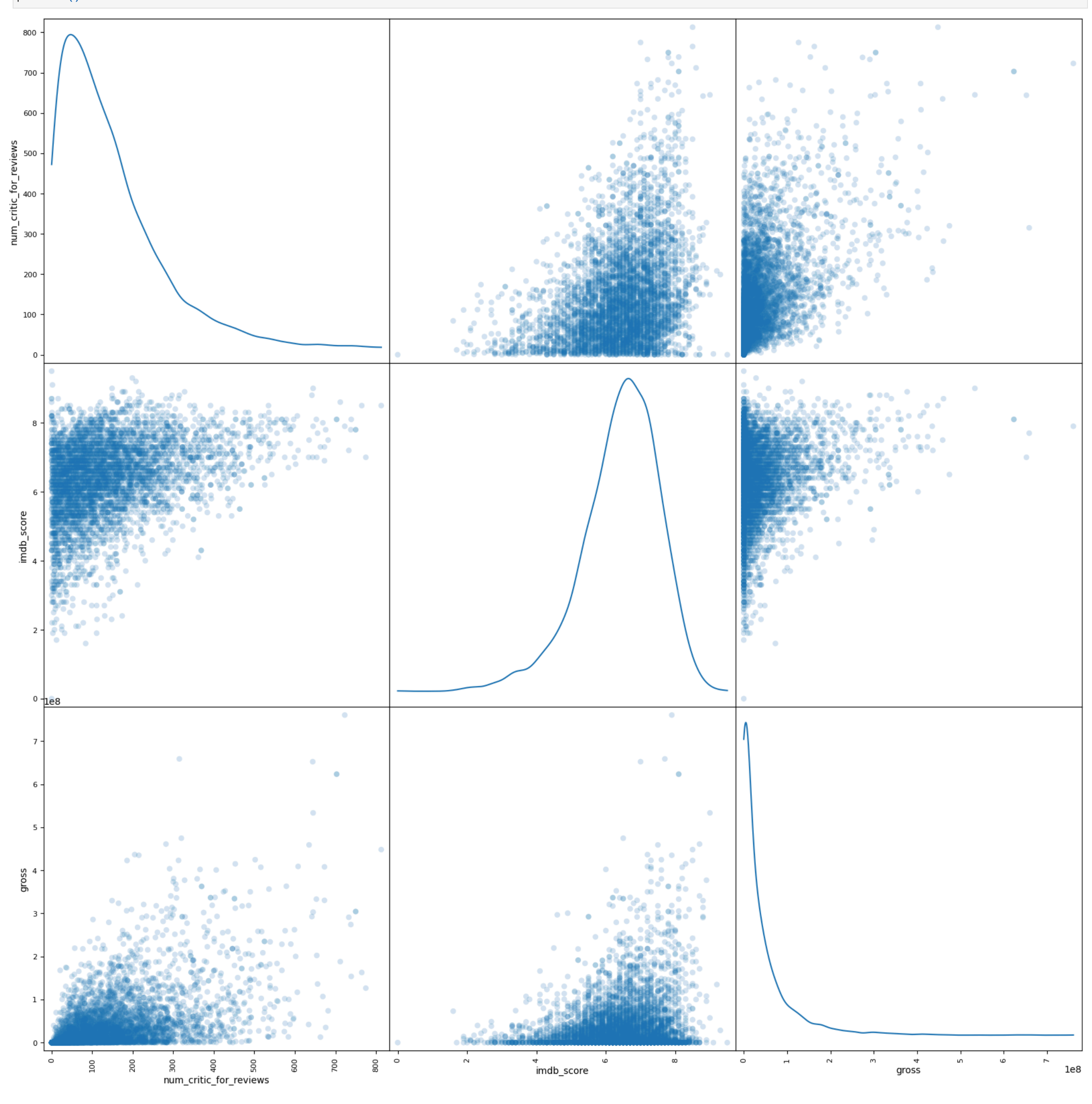
<Figure size 1200x600 with 0 Axes>



Plot the following variables in one graph:

- num_critic_for_reviews
- IMDB score
- gross

In [9]:
```
from pandas.plotting import scatter_matrix

# DInitailize needed columns
cols = ["num_critic_for_reviews", "imdb_score", "gross"]

# Add scatter_matrix on included data in columns
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 [12]:
```
# Filter out rows where director_name is 0, Null, or empty
data = data[(data['director_name'] != '0') & (data['director_name'].notna()) & (data['director_name'] != '')]

# Ensure there are no missing values in the gross and budget columns
data = data.dropna(subset=['gross', 'budget'])

# Compute sales (gross - budget) and add it as a new column
data['sales'] = data['gross'] - data['budget']

# Display the first few rows to verify the new column
data.head()
```

Out[12]:

	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
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	English	USA	PG-13
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
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	English	UK	PG-13
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
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

5 rows × 30 columns

Which directors garnered the most total sales?

In [15]:
```
# Ensure there are no missing values in the gross and budget columns
data = data.dropna(subset=['gross', 'budget'])

# Compute sales (gross - budget) and add it as a new column
data['sales'] = data['gross'] - data['budget']

# Display the first few rows to verify the new column
data
```

Out[15]:

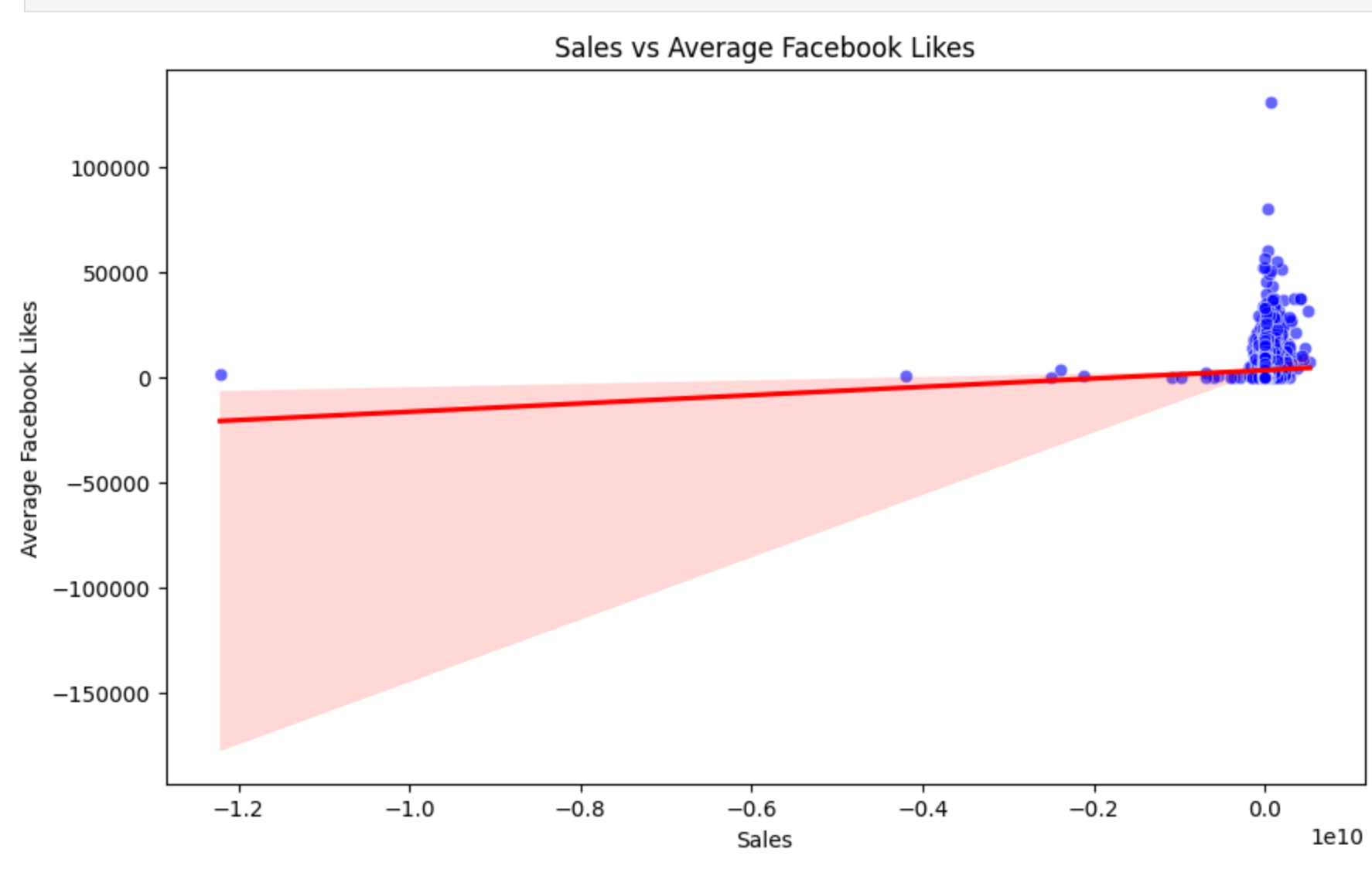
	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	...	language	country
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	English	USA
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	...	English	USA
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	English	UK
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	English	USA
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5038	5038	b'Signed Sealed Delivered'	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	...	English	Canada
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	...	English	USA
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	...	English	USA
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	...	English	USA
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	0.0	0.0	Toni Gonzaga	0.0	...	0	Philippines

4940 rows × 30 columns

In [ ]:

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

In [23]:
```
# Scatterplot for sales and 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)

# Add fit line regression
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')

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

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

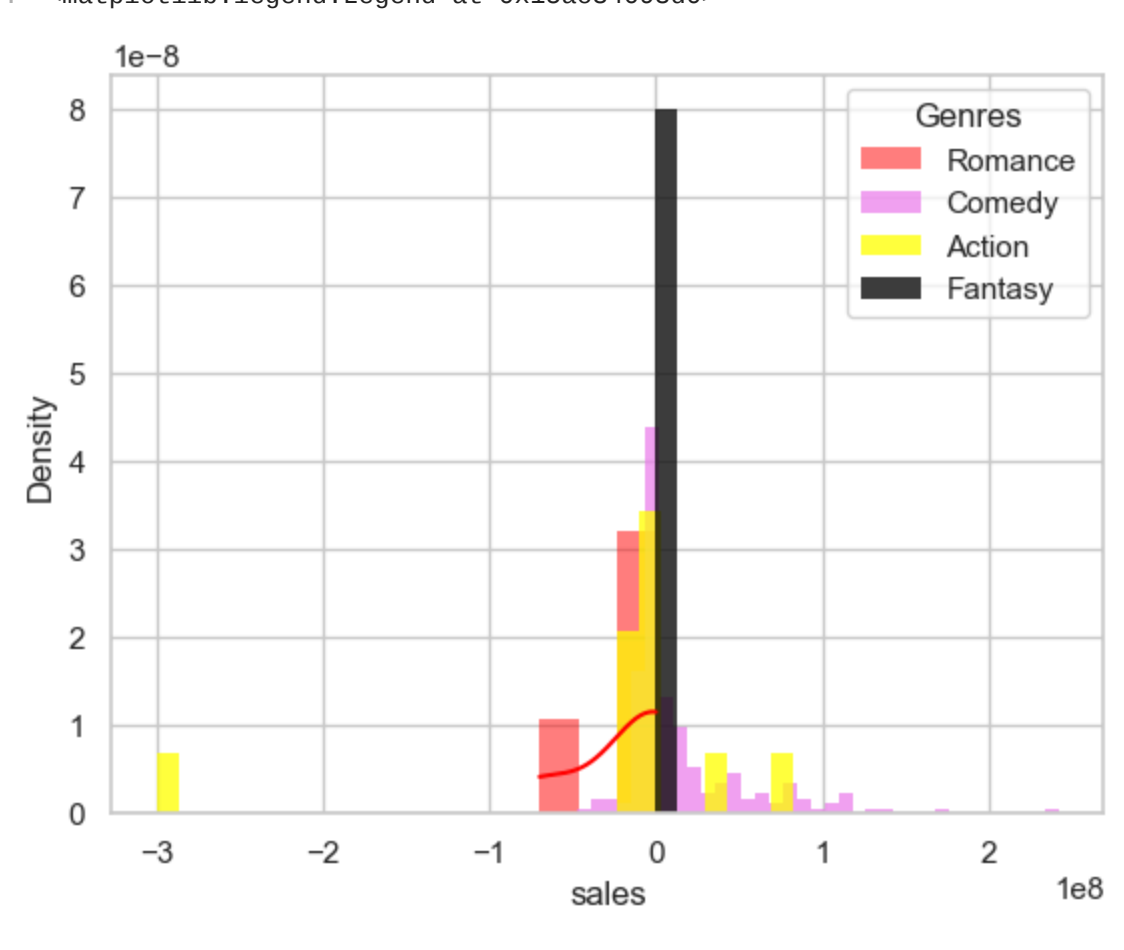


Which of these genres are the most profitable? Plot their sales using different histograms, superimposed in the same axis.

- Romance
- Comedy
- Action
- Fantasy

In [28]:
```
# Histogram for the 'Romance' genre sales with a Kernel Density Estimate (KDE)
ax = sns.histplot(data[data['genres'] == 'Romance']["sales"], color="red", label="Romance", kde=True, stat="density", linewidth=0)

# Histogram for the 'Comedy' genre sales without KDE
sns.histplot(data[data['genres'] == 'Comedy']["sales"], color="violet", label="Comedy", kde=False, stat="density", linewidth=0)

# Histogram for the 'Action' genre sales without KDE
sns.histplot(data[data['genres'] == 'Action']["sales"], color="yellow", label="Action", kde=False, stat="density", linewidth=0)

# Histogram for the 'Fantasy' genre sales without KDE
sns.histplot(data[data['genres'] == 'Fantasy']["sales"], color="black", label="Fantasy", kde=False, stat="density", linewidth=0)

# Legend title for lists of genre
ax.legend(title="Genres")
```

Out[28]: <matplotlib.legend.Legend at 0x15ac34093d0>



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 [22]:
```
#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[22]:

	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	...	actor_2_facebook_likes
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	936.0
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	...	5000.0
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	393.0
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	23000.0
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	...	12.0
...	...	...	...	...	...	...	...	...	...	...	...	...
5039	5039	b'The Following '	Color	0	43.0	43.0	0.0	319.0	Valorie Curry	841.0	...	593.0
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	...	0.0
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	...	719.0
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	...	23.0
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	0.0	0.0	Toni Gonzaga	0.0	...	0.0

5044 rows × 35 columns

Copying the whole dataframe

In [35]:
```
df = data.copy()
df.head()
```

Out[35]:

	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	...	content_rating	budget
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	...	PG-13	237000000.0
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	...	PG-13	300000000.0
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	...	PG-13	245000000.0
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	...	PG-13	250000000.0
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

5 rows × 32 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 [40]:
```
# Select columns with numeric data types (integers or floats)
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

# Create a copy of the original DataFrame to store the normalized values
normalized_df = df.copy()

# Iterate in numeric column
for column in numeric_columns:
    # Compute the minimum and maximum values of the current column
    min_value = df[column].min()
    max_value = df[column].max()

    # Apply min-max normalization
    normalized_df[column] = (df[column] - min_value) / (max_value - min_value)

# Output the normalized DataFrame
normalized_df
```

Out[40]:

	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	...	content_rating	budget
0	0.000000	b'Avatar'	Color	James Cameron	0.889299	0.941799	0.000000	0.037174	Joel David Moore	0.001563	...	PG-13	1.940158e-02
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	...	PG-13	2.455896e-02
2	0.000397	b'Spectre'	Color	Sam Mendes	0.740467	0.783069	0.000000	0.007000	Rory Kinnear	0.017188	...	PG-13	2.005649e-02
3	0.000595	b'The Dark Knight Rises'	Color	Christopher Nolan	1.000000	0.867725	0.956522	1.000000	Christian Bale	0.042188	...	PG-13	2.046580e-02
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	0.000000e+00
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5039	0.999207	b'The Following '	Color	0	0.052891	0.227513	0.000000	0.013870	Valorie Curry	0.001314	...	TV-14	0.000000e+00
5040	0.999405	b'A Plague So Pleasant'	Color	Benjamin Roberds	0.015990	0.402116	0.000000	0.000000	Maxwell Moody	0.000000	...	0	1.146085e-07
5041	0.999603	b'Shanghai Calling'	Color	Daniel Hsia	0.017220	0.529101	0.000000	0.021261	Daniel Henney	0.001478	...	PG-13	0.000000e+00
5042	0.999802	b'My Date with Drew'	Color	Jon Gunn	0.052891	0.476190	0.000696	0.000696	Brian Herzlinger	0.000134	...	PG	9.004953e-08
5043	1.000000	b'Starting Over Again'	0	Olivia Lamasan	0.000000	0.000000	0.000000	0.000000	Toni Gonzaga	0.000000	...	PG	0.000000e+00

5044 rows × 32 columns