In [1]:
```python
# import libraries
import pymysql
import pandas as pd

# Database Connection Details
db_config = {
    "host": "localhost",
    "user": "root",
    "password": "123456",
    "database": "socialmedia_db"
}

try:
    # Establish connection
    connection = pymysql.connect(**db_config)
    cursor = connection.cursor()
    # Create database if not exists
    cursor.execute("CREATE DATABASE IF NOT EXISTS socialmedia_db")
    cursor.execute("USE socialmedia_db")

    # Create a table to store social media data
    create_table_query = """
    CREATE TABLE IF NOT EXISTS social_media_sentiments (
        Post_ID INT AUTO_INCREMENT PRIMARY KEY,
        User_Name VARCHAR(255),
        Post_Content TEXT,
        Platform VARCHAR(50),
        Post_Date DATETIME,
        Sentiment_Score FLOAT,
        Sentiment_Label VARCHAR(20)
    );
    """
    cursor.execute(create_table_query)
    print("Table 'social_media_sentiments' successfully created in socialmedia_db database")

except pymysql.MySQLError as err:
    print(f"Error: {err}")
finally:
    if connection:
        cursor.close()
        connection.close()
        print("MySQL connection closed")
```

```
Table 'social_media_sentiments' successfully created in socialmedia_db database
MySQL connection closed
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_2664\4208718895.py:2: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

In [2]:
```python
import pymysql
import pandas as pd

# Load dataset
data = pd.read_csv('social_media_sentiments.csv')

# Database Connection Details
db_config = {
    "host": "localhost",
    "user": "root",
    "password": "123456",
    "database": "socialmedia_db"
}

# Establish the connection
connection = pymysql.connect(**db_config)
cursor = connection.cursor()

# Prepare the insert query
insert_query = """
INSERT INTO social_media_sentiments (
    User_Name, Post_Content, Platform, Post_Date, Sentiment_Score, Sentiment_Label
) VALUES (%s, %s, %s, %s, %s, %s);
"""

# Inserting each row into the MySQL DB
for row in data.itertuples(index=False, name=None):
    try:
        cursor.execute(insert_query, row)
    except pymysql.MySQLError as err:
        print(f"Error inserting row {row}: {err}")
        continue

# Commit transaction
connection.commit()
print("Social media data inserted successfully")

# Close the connection
cursor.close()
connection.close()
```

```
Social media data inserted successfully
```

Understanding Data

In [3]:
```python
# to see first five rows
data.head(5)
```

Out[3]:

	User_Name	Post_Content	Platform	Post_Date	Sentiment_Score	Sentiment_Label
0	John Richardson	Upon never science.	LinkedIn	2024-11-13 05:03:27	-0.24	Neutral
1	Raymond Anderson	Federal if let.	Instagram	2024-10-29 04:58:33	0.95	Positive
2	Jason Hoffman	Draw TV site such.	Instagram	2025-02-02 00:33:02	-0.73	Negative
3	Melissa Davis	Sing work else compare sure then east.	Twitter	2023-07-15 21:46:05	-0.38	Neutral
4	Jamie Summers	Green level majority such.	Twitter	2025-02-06 15:26:27	-0.66	Negative

In [4]:
```python
# Information about dataset

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User_Name        500 non-null    object
 1   Post_Content     500 non-null    object
 2   Platform         500 non-null    object
 3   Post_Date        500 non-null    object
 4   Sentiment_Score  500 non-null    float64
 5   Sentiment_Label  500 non-null    object
dtypes: float64(1), object(5)
memory usage: 23.6+ KB
```

In [5]:
```python
# to see basic descriptive statistics
data.describe()
```

Out[5]:

	Sentiment_Score
count	500.000000
mean	0.012020
std	0.628103
min	-1.000000
25%	-0.572500
50%	0.005000
75%	0.620000
max	1.000000

In [6]:
```python
# to see datatypes
data.dtypes
```

Out[6]:
```
User_Name          object
Post_Content       object
Platform           object
Post_Date          object
Sentiment_Score    float64
Sentiment_Label    object
dtype: object
```

Data Cleaning & Preprocessing

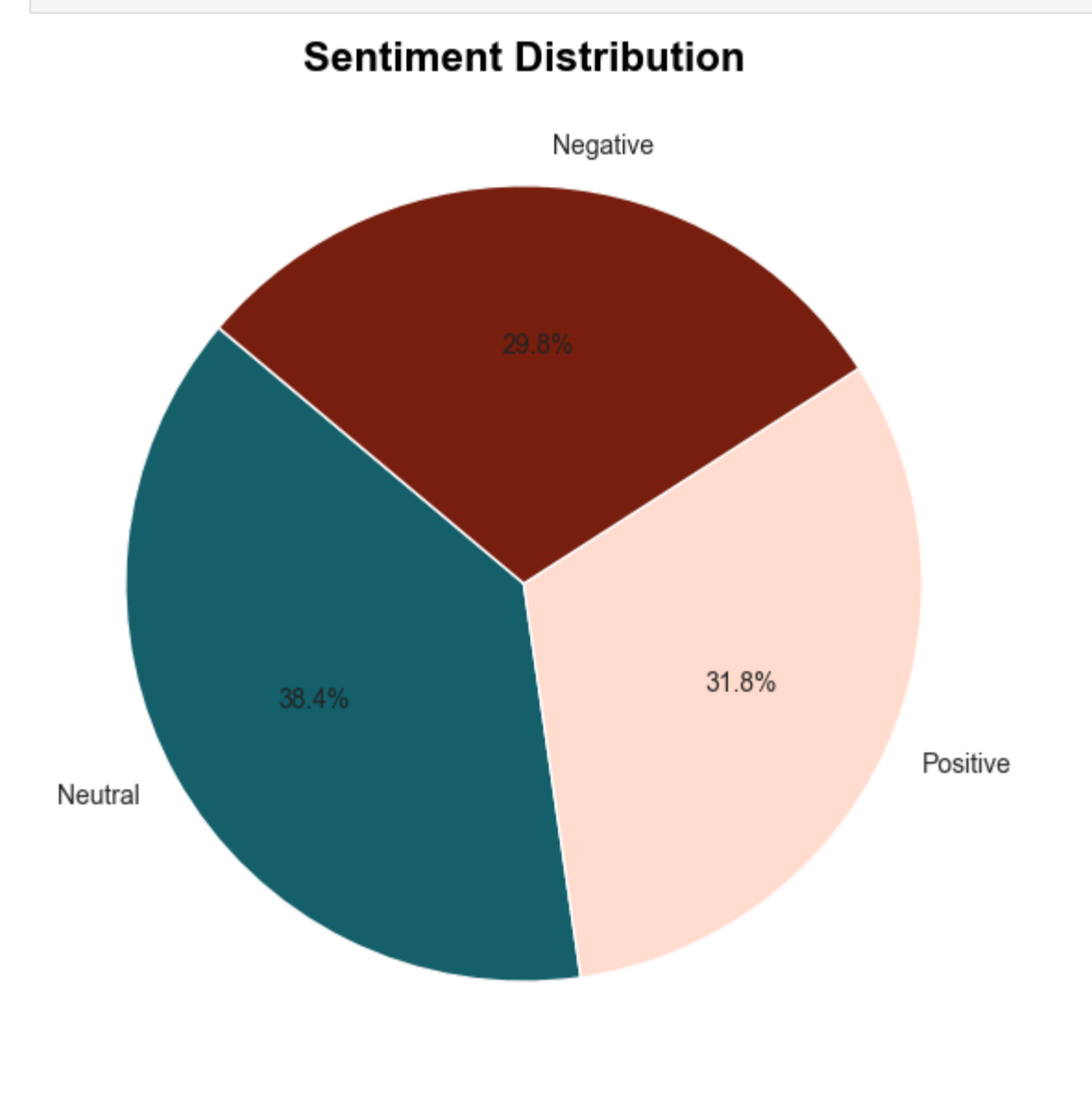
In [7]:
```python
# check duplicated values
data.duplicated().sum()
```

Out[7]: 0

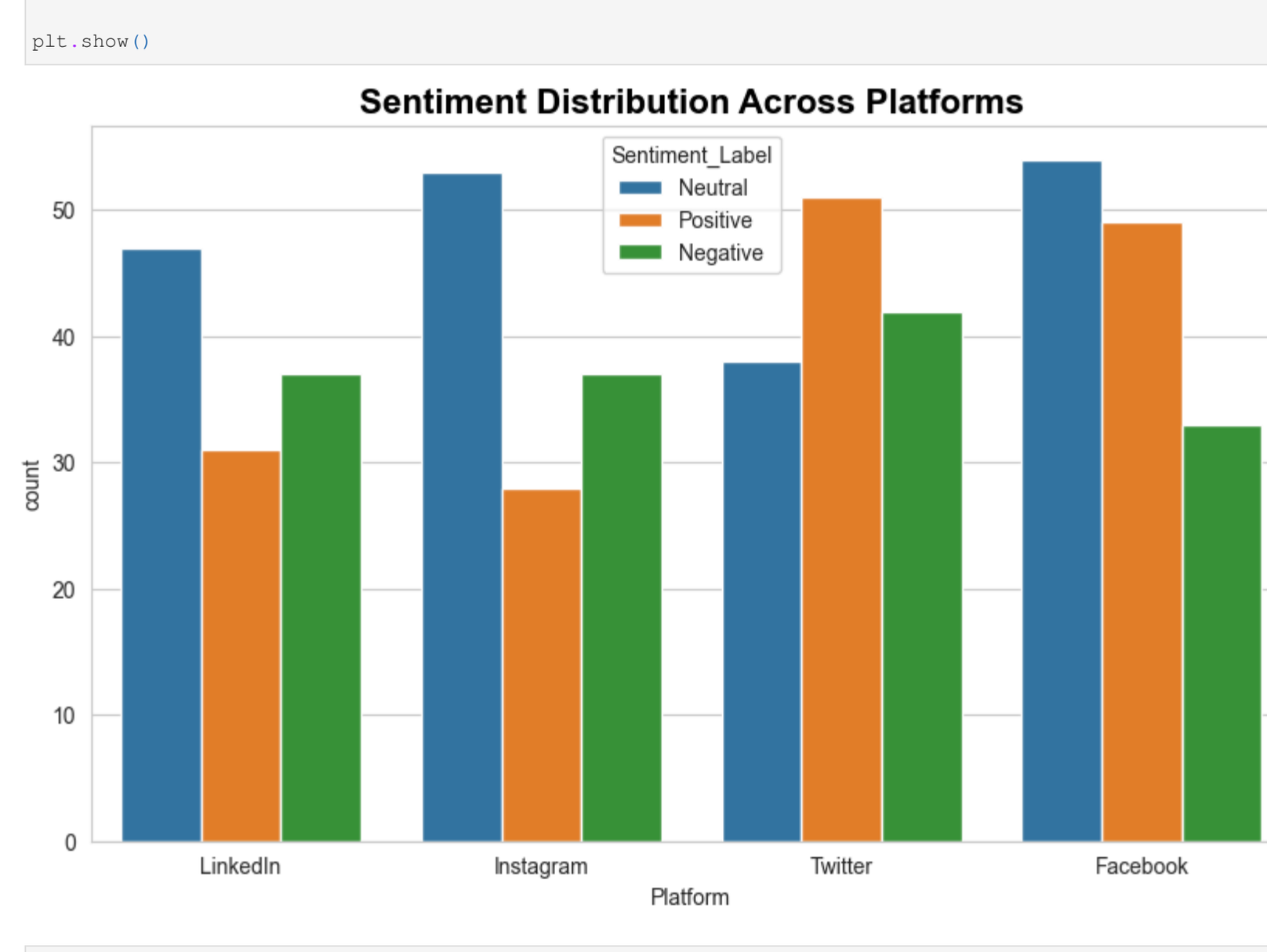
In [8]:
```python
# to check null values
data.isnull().sum()
```

Out[8]:
```
User_Name          0
Post_Content       0
Platform           0
Post_Date          0
Sentiment_Score    0
Sentiment_Label    0
dtype: int64
```

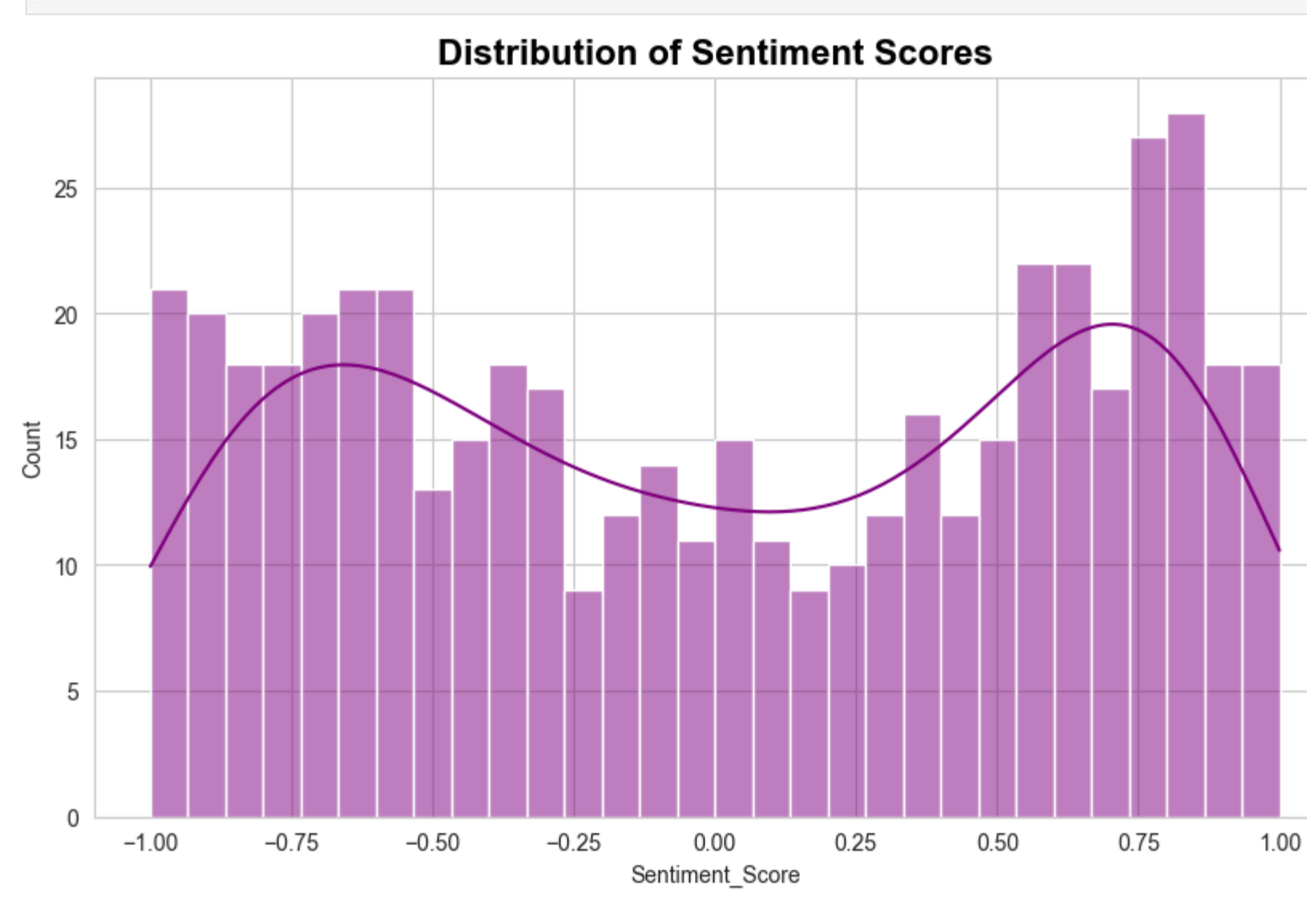
In [75]:
```python
import matplotlib.pyplot as plt

# Sentiment Distribution (Pie Chart)

plt.figure(figsize=(7, 7))
data["Sentiment_Label"].value_counts().plot.pie(autopct='%1.1f%%', startangle=140, colors=["#15616D", "#FFDCD1", "#78200F"])
# Customize the chart
plt.title("Sentiment Distribution",fontsize=16, fontweight='bold', color="black")
plt.ylabel("")
plt.show()
```



In [76]:
```python
# Platform-wise Sentiment Distribution

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.countplot(x="Platform", hue="Sentiment_Label", data=data)
plt.title("Sentiment Distribution Across Platforms",fontsize=16, fontweight='bold', color="black")

plt.show()
```

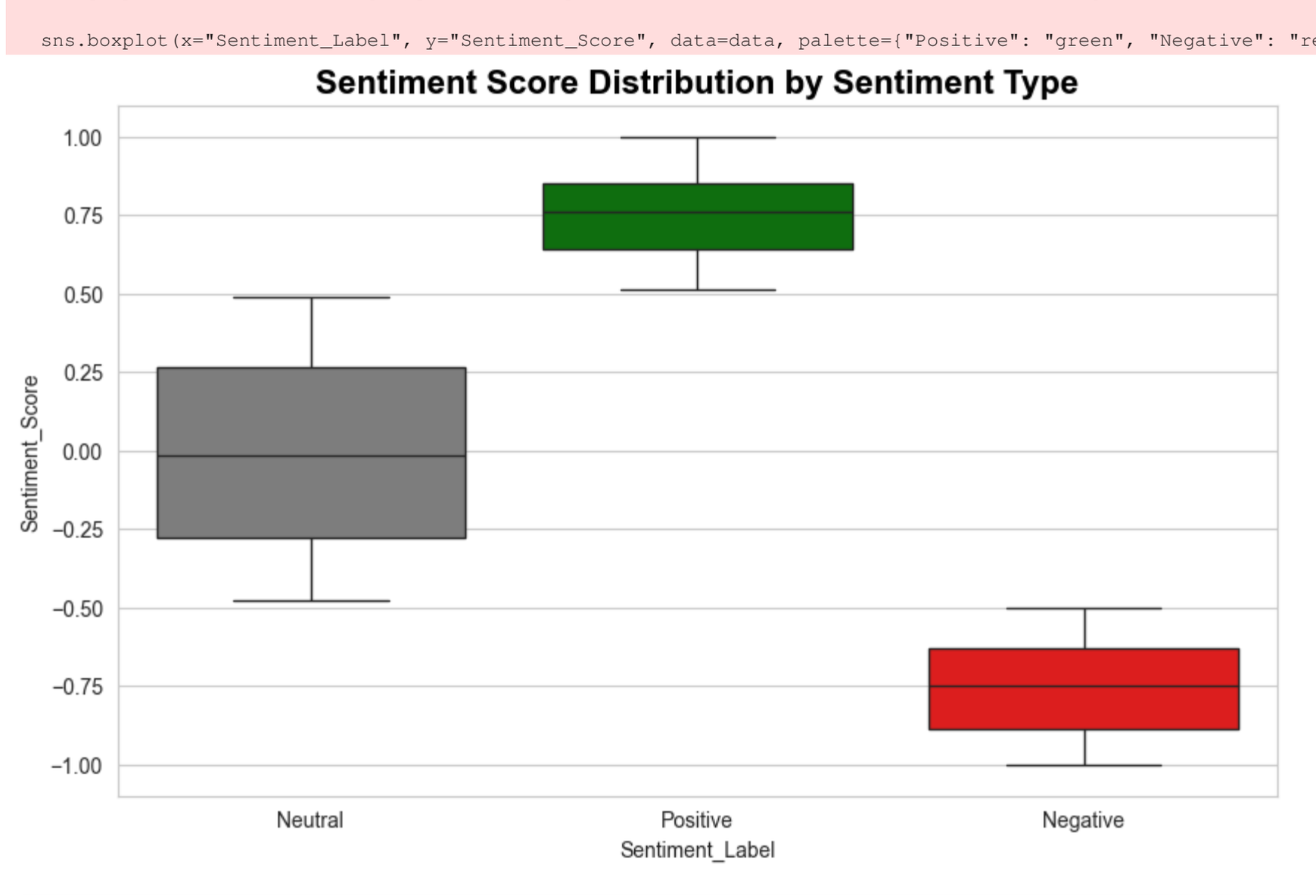


In [16]:
```python
# Sentiment Score Distribution (Histogram)

plt.figure(figsize=(10, 6))
sns.histplot(data["Sentiment_Score"], bins=30, kde=True, color="purple")
plt.title("Distribution of Sentiment Scores",fontsize=16, fontweight='bold', color="black")
plt.show()
```

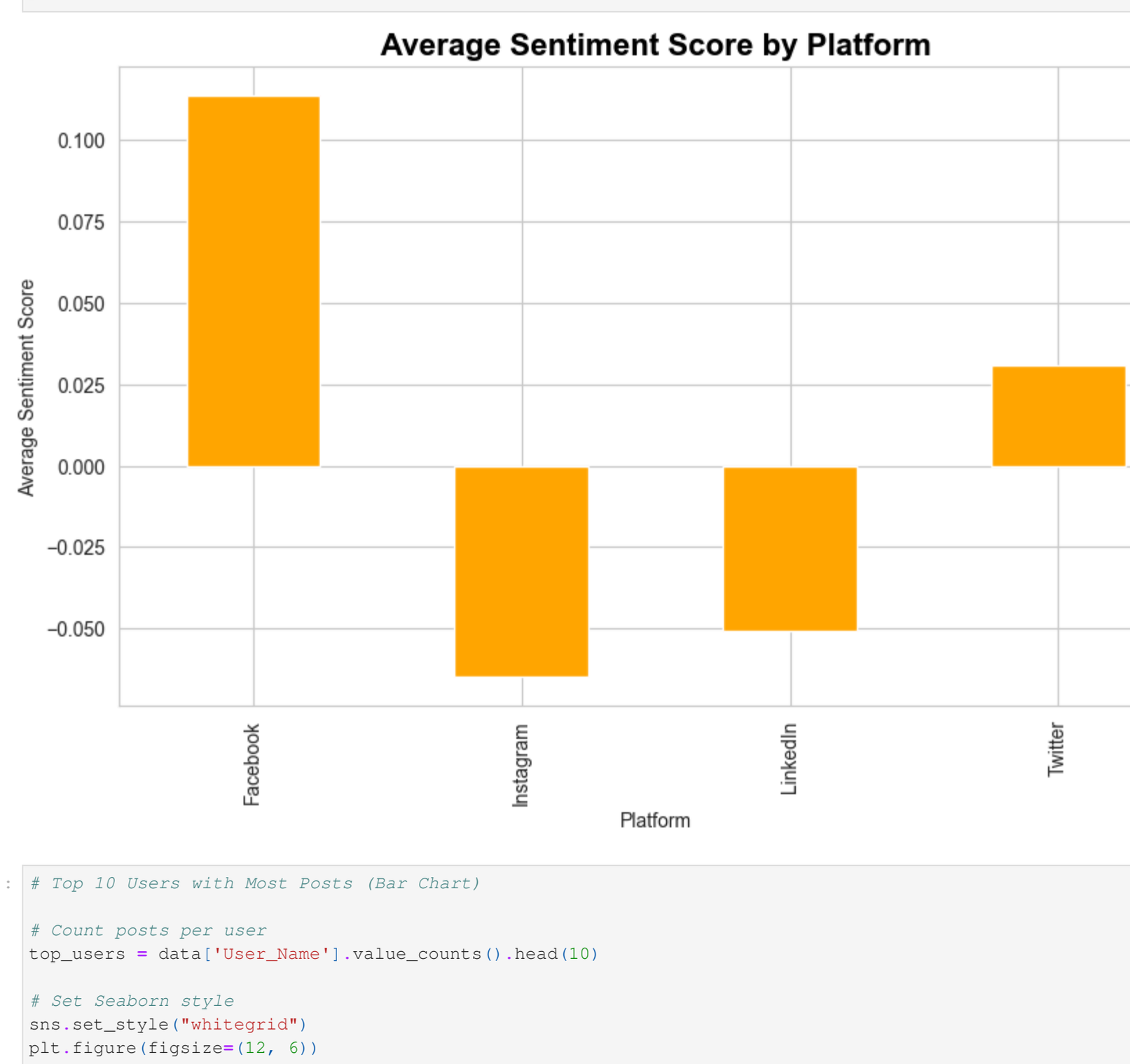


In [20]:
```python
# Sentiment Score Distribution by Sentiment type (Boxplot)

plt.figure(figsize=(10, 6))
sns.boxplot(x="Sentiment_Label", y="Sentiment_Score", data=data, palette={"Positive": "green", "Negative": "red", "Neutral": "gray"})
plt.title("Sentiment Score Distribution by Sentiment Type",fontsize=16, fontweight='bold', color="black")
plt.show()
```

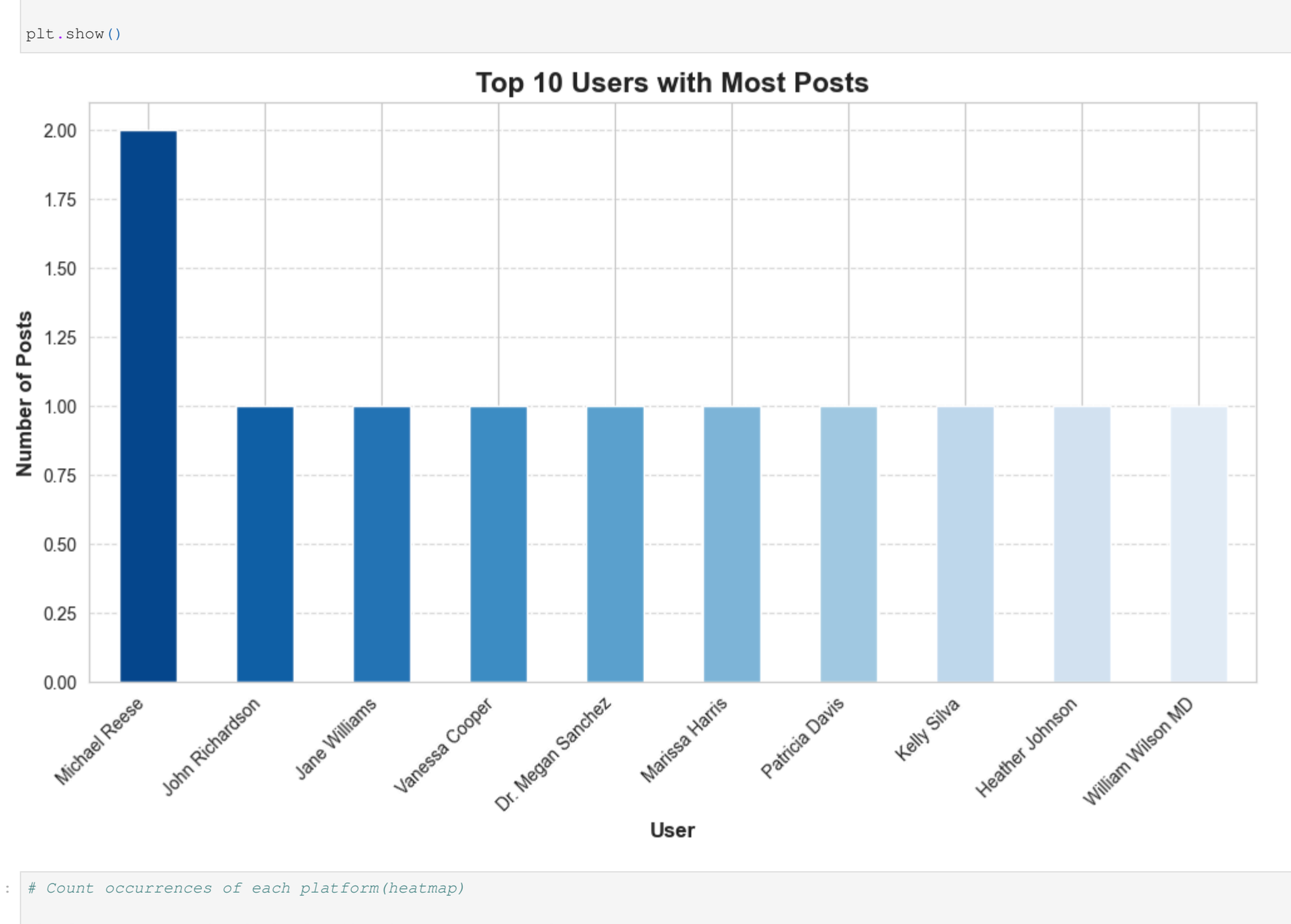
```
C:\Users\dell\AppData\Local\Temp\ipykernel_2664\2865760163.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sam

  sns.boxplot(x="Sentiment_Label", y="Sentiment_Score", data=data, palette={"Positive": "green", "Negative": "red", "Neutral": "gray"})
```



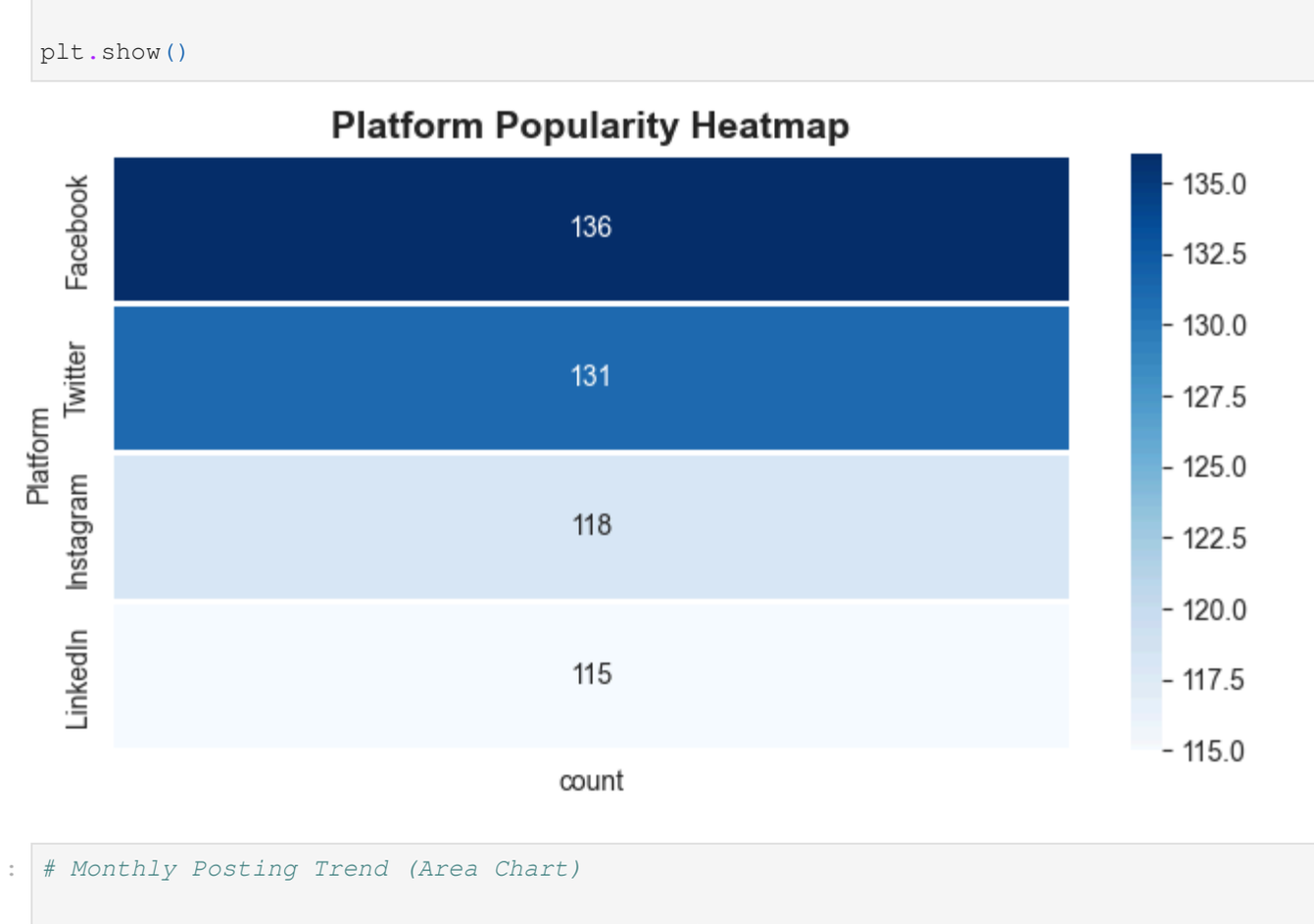
In [21]:
```python
# Average Sentiment Score by Platform (Bar Chart)

platform_sentiments = data.groupby("Platform")["Sentiment_Score"].mean()
plt.figure(figsize=(10, 6))

# Create a bar plot
platform_sentiments.plot(kind="bar", color="orange")

# Customize the chart
plt.title("Average Sentiment Score by Platform",fontsize=16, fontweight='bold', color="black")
plt.xlabel("Platform")
plt.ylabel("Average Sentiment Score")
plt.show()
```



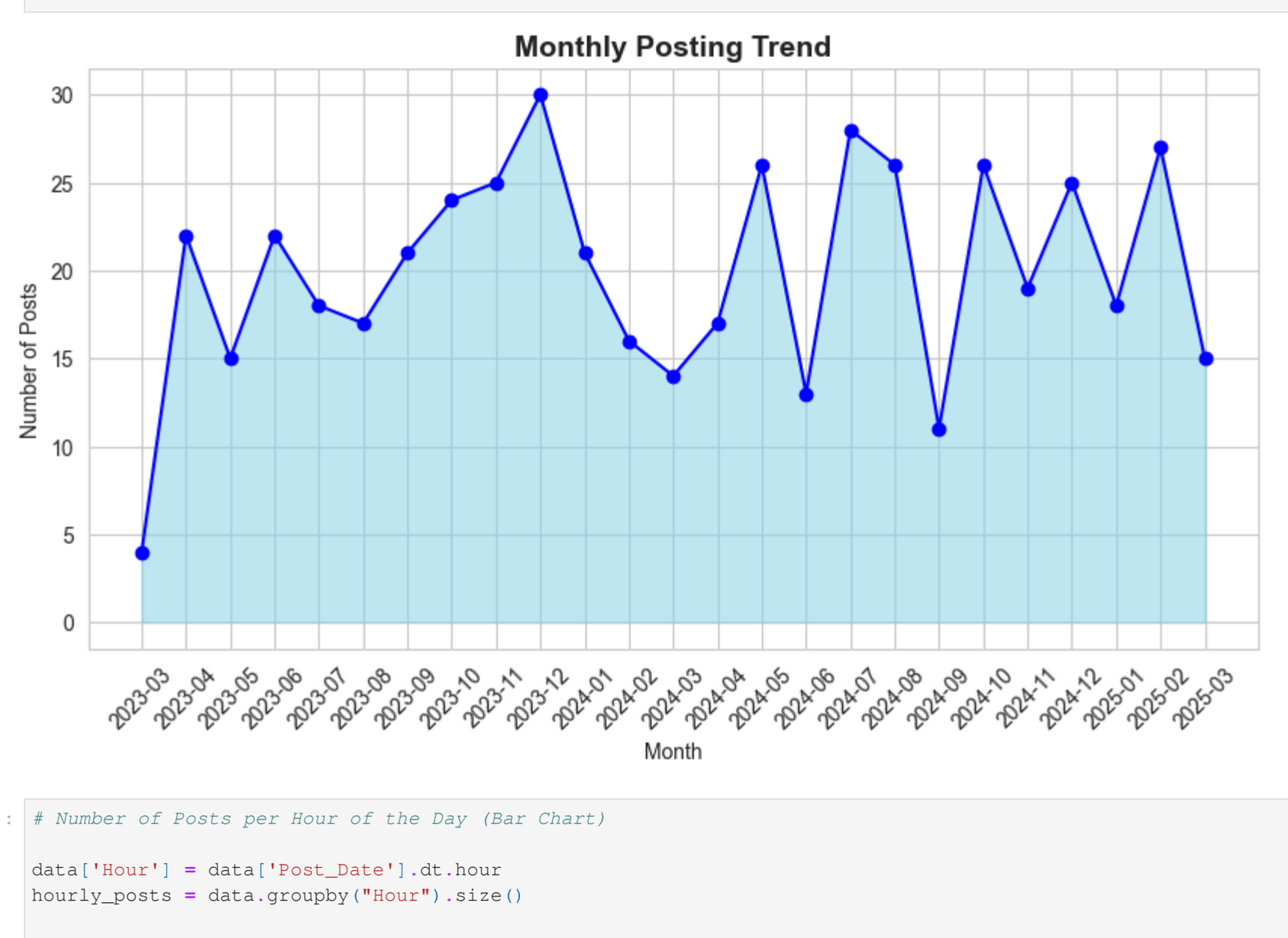
In [33]:
```python
# Top 10 Users with Most Posts (Bar Chart)

# Count posts per user
top_users = data['User_Name'].value_counts().head(10)

# Set Seaborn style
sns.set_style("whitegrid")
plt.figure(figsize=(12, 6))

# Create a bar plot
colors = sns.color_palette("Blues_r", len(top_users))
top_users.plot(kind="bar", color=colors)

# Customize the chart
plt.title("Top 10 Users with Most Posts", fontsize=16, fontweight='bold')
plt.xlabel("User", fontsize=12, fontweight='bold')
plt.ylabel("Number of Posts", fontsize=12, fontweight='bold')
plt.xticks(rotation=45, ha="right", fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()
```



In [77]:
```python
# Count occurrences of each platform(heatmap)

platform_counts = data['Platform'].value_counts().to_frame()

# Create heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(platform_counts, annot=True, cmap="Blues", fmt="d", linewidths=1)

# Title and labels
plt.title("Platform Popularity Heatmap", fontsize=14, fontweight="bold")
plt.ylabel("Platform")
# plt.ylabel("Count")

plt.show()
```

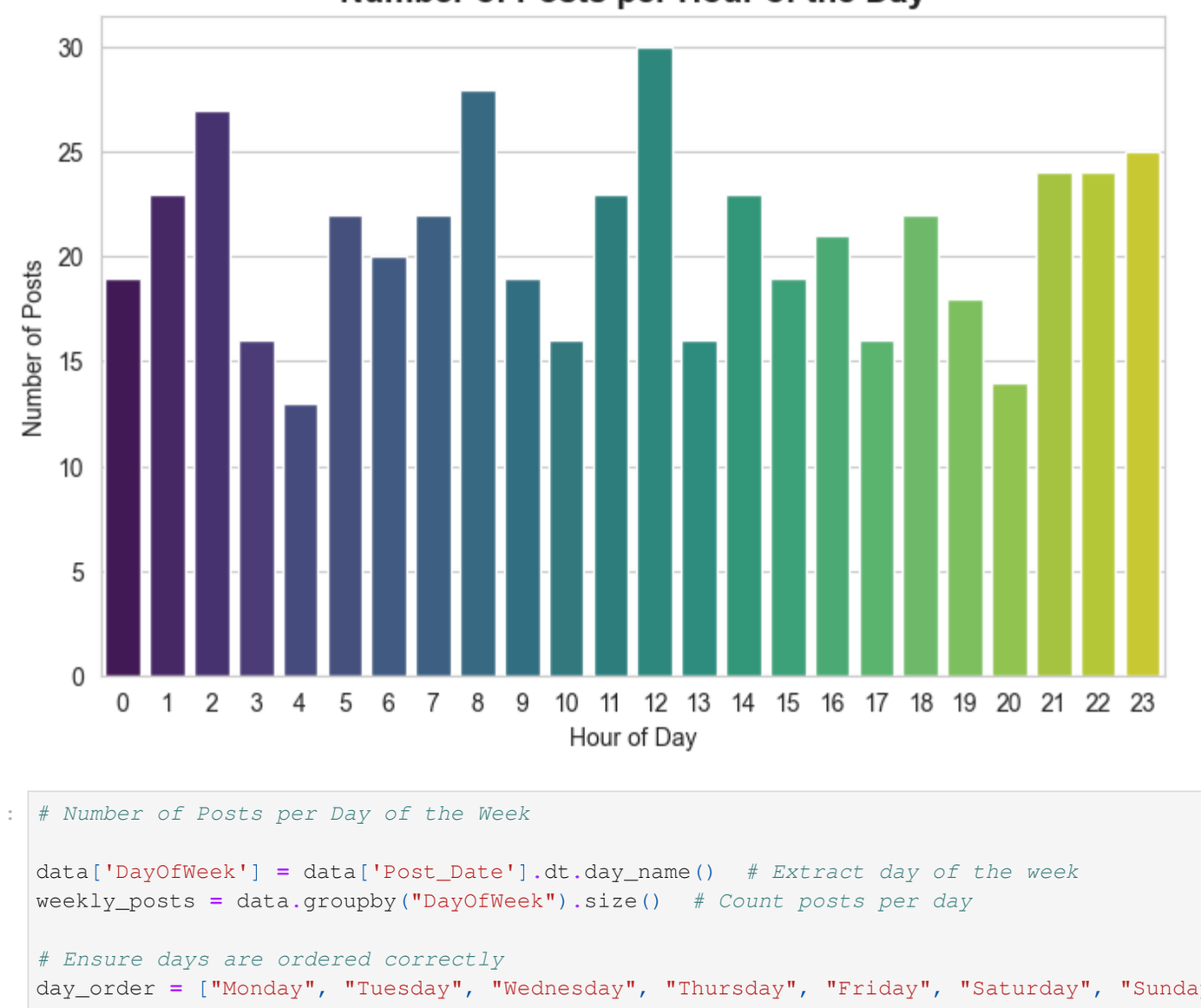


In [59]:
```python
# Monthly Posting Trend (Area Chart)

data["Month"] = data["Post_Date"].dt.to_period("M")
posts_per_month = data.groupby("Month").size()

plt.figure(figsize=(10, 5))
plt.fill_between(posts_per_month.index.astype(str), posts_per_month, color="skyblue", alpha=0.5)
plt.plot(posts_per_month.index.astype(str), posts_per_month, marker="o", color="blue")
# customize chart
plt.xticks(rotation=45)
plt.title("Monthly Posting Trend", fontsize=14, fontweight="bold")
plt.xlabel("Month")
plt.ylabel("Number of Posts")
plt.show()
```



In [72]:
```python
# Number of Posts per Hour of the Day (Bar Chart)

data['Hour'] = data['Post_Date'].dt.hour
hourly_posts = data.groupby("Hour").size()

plt.figure(figsize=(8, 5))
sns.barplot(x=hourly_posts.index, y=hourly_posts.values, palette="viridis")
plt.title("Number of Posts per Hour of the Day", fontsize=14, fontweight="bold")
plt.xlabel("Hour of Day")
plt.ylabel("Number of Posts")
plt.xticks(range(0, 24))
plt.show()
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_2664\2021508293.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sam

  sns.barplot(x=hourly_posts.index, y=hourly_posts.values, palette="viridis")
```



In [83]:
```python
# Number of Posts per Day of the Week

data['DayOfWeek'] = data['Post_Date'].dt.day_name()  # Extract day of the week
weekly_posts = data.groupby("DayOfWeek").size()  # Count posts per day

# Ensure days are ordered correctly
day_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

plt.figure(figsize=(8, 5))
sns.barplot(x=weekly_posts.index, y=weekly_posts.values, order=day_order, palette="cool")
plt.title("Number of Posts per Day of the Week", fontsize=14, fontweight="bold")
plt.xlabel("Day of the Week")
plt.ylabel("Number of Posts")
plt.show()
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_2664\950804386.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sam

  sns.barplot(x=weekly_posts.index, y=weekly_posts.values, order=day_order, palette="cool")
```