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

In [2]:
```
df = pd.read_csv(r"C:\Users\santo\OneDrive\Desktop\Data science\Dec2025\14th dec EDA\social_media_engagement1.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	post_id	platform	post_type	post_time	likes	comments	shares	post_day	sentiment_score
0	1	Facebook	image	8/17/2023 14:45	2121	474	628	Thursday	positive
1	2	Facebook	carousel	5/14/2023 0:45	3660	432	694	Sunday	neutral
2	3	Instagram	poll	2/21/2023 16:15	4955	408	688	Tuesday	negative
3	4	Twitter	image	11/16/2023 0:45	1183	90	187	Thursday	negative
4	5	Twitter	video	5/23/2023 0:30	3499	247	286	Tuesday	positive

In [6]:
```
("\nData Info:\n", df.info)
```

Out[6]:
```
('\nData Info:\n',
 <bound method DataFrame.info of     post_id   platform post_type         post_time  likes  comments  shares  \
 0         1   Facebook     image   8/17/2023 14:45   2121       474     628
 1         2   Facebook  carousel    5/14/2023 0:45   3660       432     694
 2         3  Instagram      poll   2/21/2023 16:15   4955       408     688
 3         4    Twitter     image   11/16/2023 0:45   1183        90     187
 4         5    Twitter     video    5/23/2023 0:30   3499       247     286
 ..      ...        ...       ...               ...    ...       ...     ...
 95       96  Instagram  carousel   7/12/2023 17:45     36       294     911
 96       97    Twitter     video  10/27/2023 23:45    314       108     458
 97       98    Twitter      text     8/5/2023 8:45    229       179      38
 98       99  Instagram      poll  12/29/2023 12:15   5000       500     204
 99      100  Instagram     image    6/6/2023 21:00   4483       357      25

      post_day sentiment_score
 0    Thursday        positive
 1      Sunday         neutral
 2     Tuesday        negative
 3    Thursday        negative
 4     Tuesday        positive
 ..        ...             ...
 95  Wednesday        positive
 96     Friday         neutral
 97   Saturday        positive
 98     Friday        positive
 99    Tuesday         neutral

 [100 rows x 9 columns]>)
```

In [7]:
```
df.describe()
```

Out[7]:

	post_id	likes	comments	shares
count	100.000000	100.000000	100.00000	100.000000
mean	50.500000	2381.810000	202.66000	415.650000
std	29.011492	1632.573284	138.84067	283.877601
min	1.000000	15.000000	10.00000	16.000000
25%	25.750000	895.750000	89.75000	183.000000
50%	50.500000	2220.000000	171.00000	356.500000
75%	75.250000	3593.250000	299.00000	689.500000
max	100.000000	5000.000000	500.00000	993.000000

Data Scrubbing

In [9]:
```
print(df.isnull().sum())
```

```
post_id           0
platform          0
post_type         0
post_time         0
likes             0
comments          0
shares            0
post_day          0
sentiment_score   0
dtype: int64
```

In [10]:
```
# Check missing values
print(df.isnull().sum())

# Option 1: Drop rows with missing values
df = df.dropna()
```

```
# Option 2: Fill missing values (example: fill numeric with 0, categorical with mode)
df["likes"] = df["likes"].fillna(0)
df["comments"] = df["comments"].fillna(0)
df["shares"] = df["shares"].fillna(0)

for col in ["platform", "post_type", "post_day", "sentiment_score"]:
    df[col] = df[col].fillna(df[col].mode()[0])
```

```
post_id            0
platform           0
post_type          0
post_time          0
likes              0
comments           0
shares             0
post_day           0
sentiment_score    0
dtype: int64
```

checking Duplicate Values:

In [11]:
```
print("Duplicate rows:", df.duplicated().sum())
df = df.drop_duplicates()
```

```
Duplicate rows: 0
```

Data Exploration

Summary of the data

In [13]:
```
summary_platform = df.groupby("platform")[["likes", "comments", "shares"]].mean().round(3) #mean is rounded with three decimal

summary_platform
```

Out[13]:

platform	likes	comments	shares
Facebook	2699.750	248.906	474.375
Instagram	2999.833	232.444	525.167
Twitter	1368.594	122.906	233.719

In [14]:
```
#Summary of engagement by post type:

summary_post_type = df.groupby("post_type")[["likes", "comments", "shares"]].mean().round(3) #mean is rounded with three decim


summary_post_type
```

Out[14]:

post_type	likes	comments	shares
carousel	2263.577	236.000	531.154
image	2032.765	197.176	371.529
poll	3061.077	214.538	470.769
text	1815.286	138.381	274.952
video	2906.783	221.000	415.000

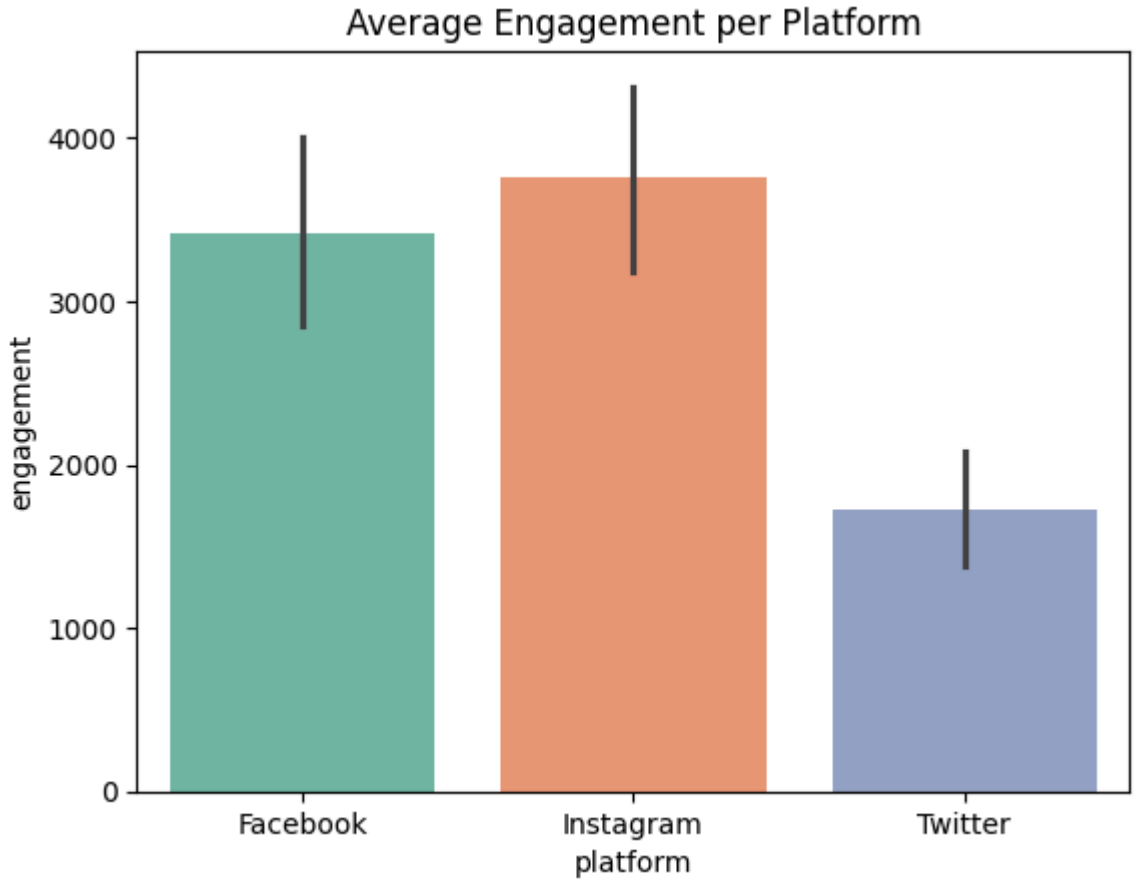
In [15]:
```
# Create engagement column
df["engagement"] = df["likes"] + df["comments"] + df["shares"]

# Average engagement per platform
engagement_platform = df.groupby("platform")["engagement"].mean().round(2) #mean is rounded with two decimal points

engagement_platform
```

Out[15]:
```
platform
Facebook     3423.03
Instagram    3757.44
Twitter      1725.22
Name: engagement, dtype: float64
```

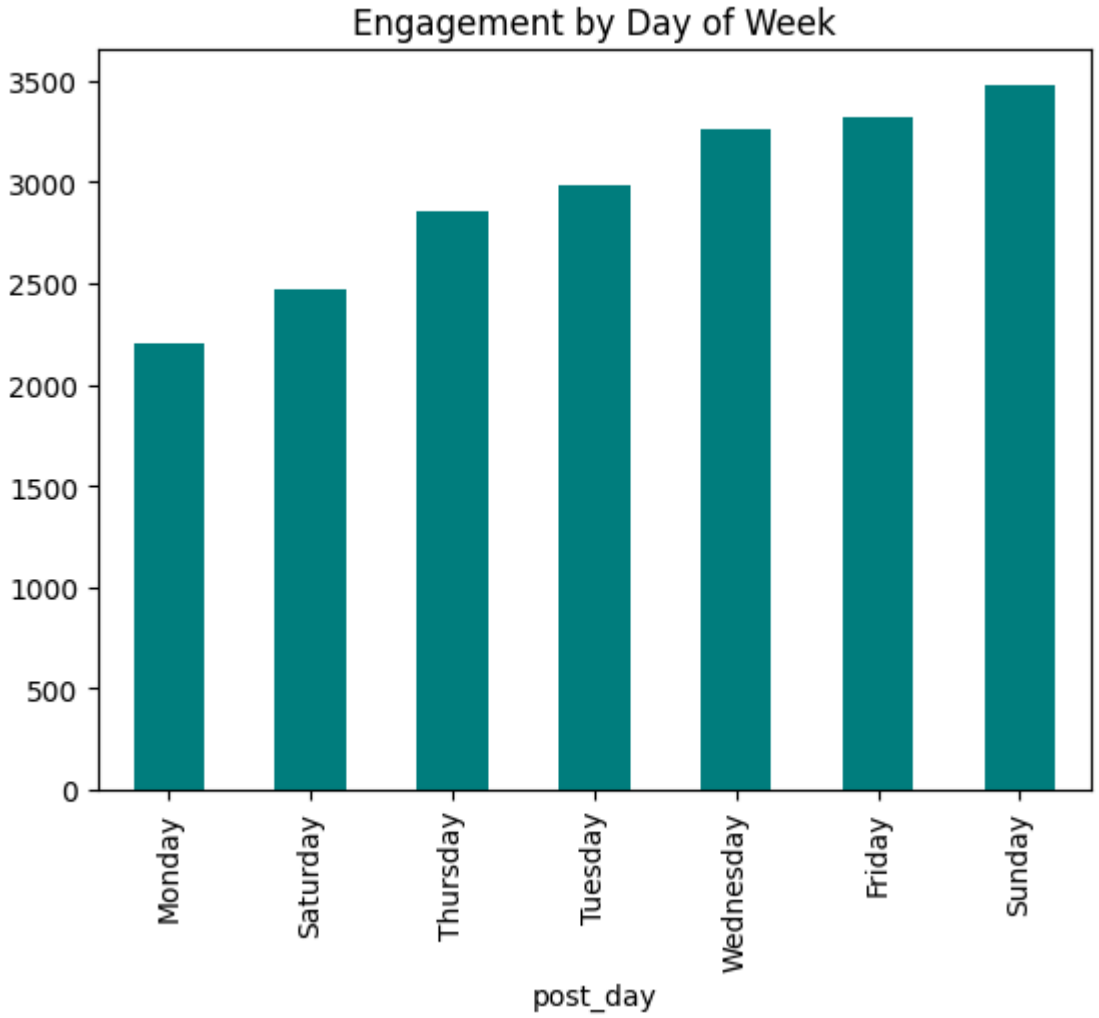
iNterpretation (Visualization of the data)

In [18]:
```
sns.barplot(
    data=df,
    x="platform",
    y="engagement",
    estimator="mean",
    palette="Set2"   # color palette
)

plt.title("Average Engagement per Platform")
plt.show()
```

```
C:\Users\santo\AppData\Local\Temp\ipykernel_17880\1142403526.py:1: FutureWarning:

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

  sns.barplot(
```

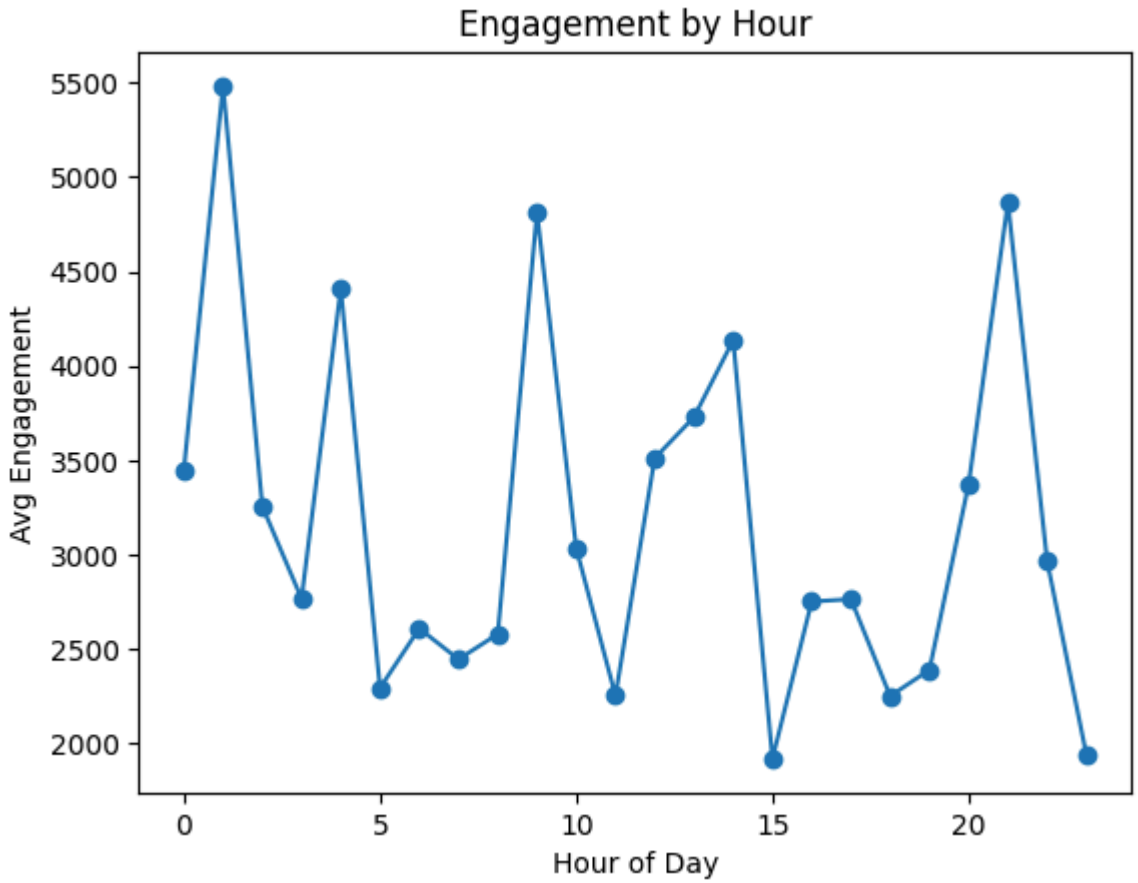


In [19]:
```
engagement_day = df.groupby("post_day")["engagement"].mean().sort_values()
engagement_day.plot(kind="bar", color="teal", title="Engagement by Day of Week")
plt.show()
```



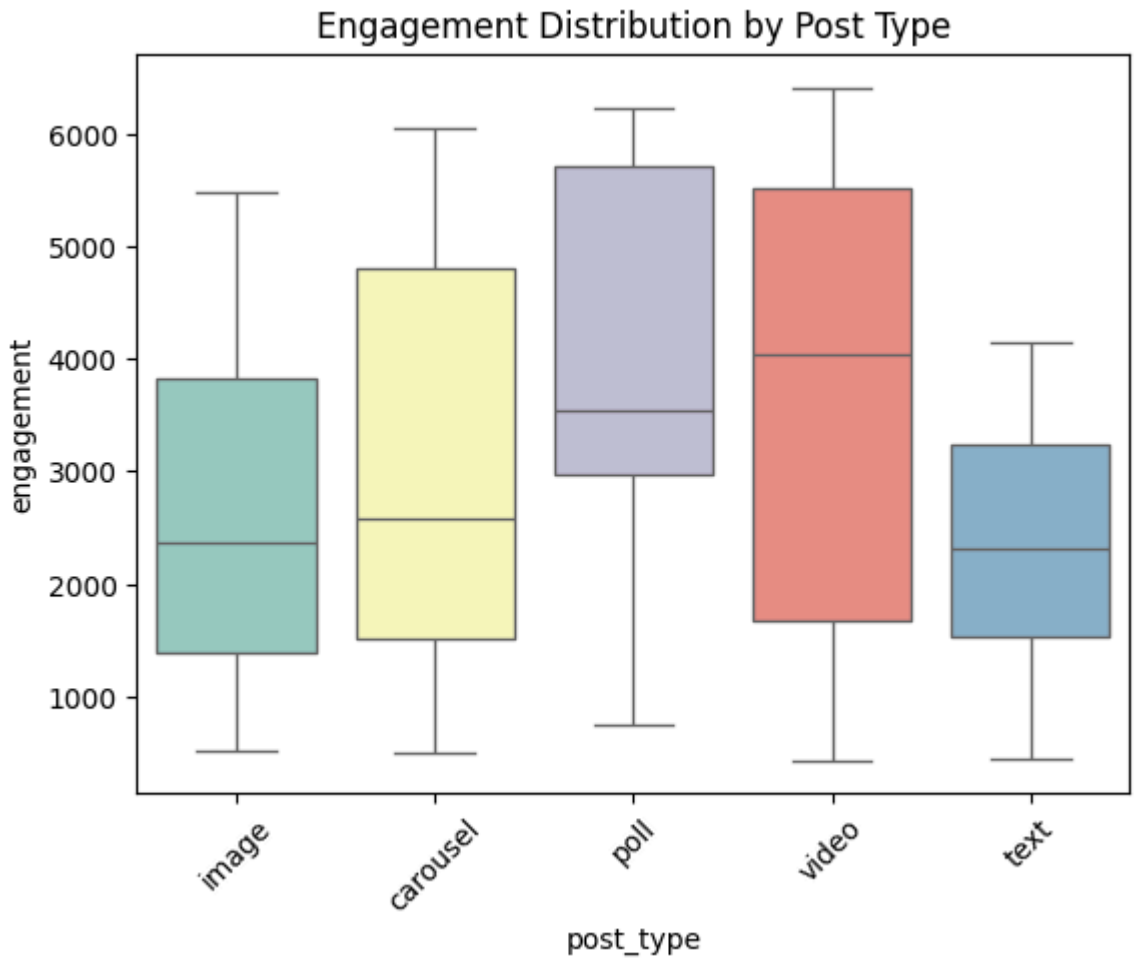
In [20]:
```
# Extract hour from post_time
df["post_time"] = pd.to_datetime(df["post_time"])
df["hour"] = df["post_time"].dt.hour

# Engagement by hour
engagement_hour = df.groupby("hour")["engagement"].mean()
engagement_hour.plot(kind="line", marker="o", title="Engagement by Hour")
plt.xlabel("Hour of Day")
```

```
plt.ylabel("Avg Engagement")
plt.show()
```



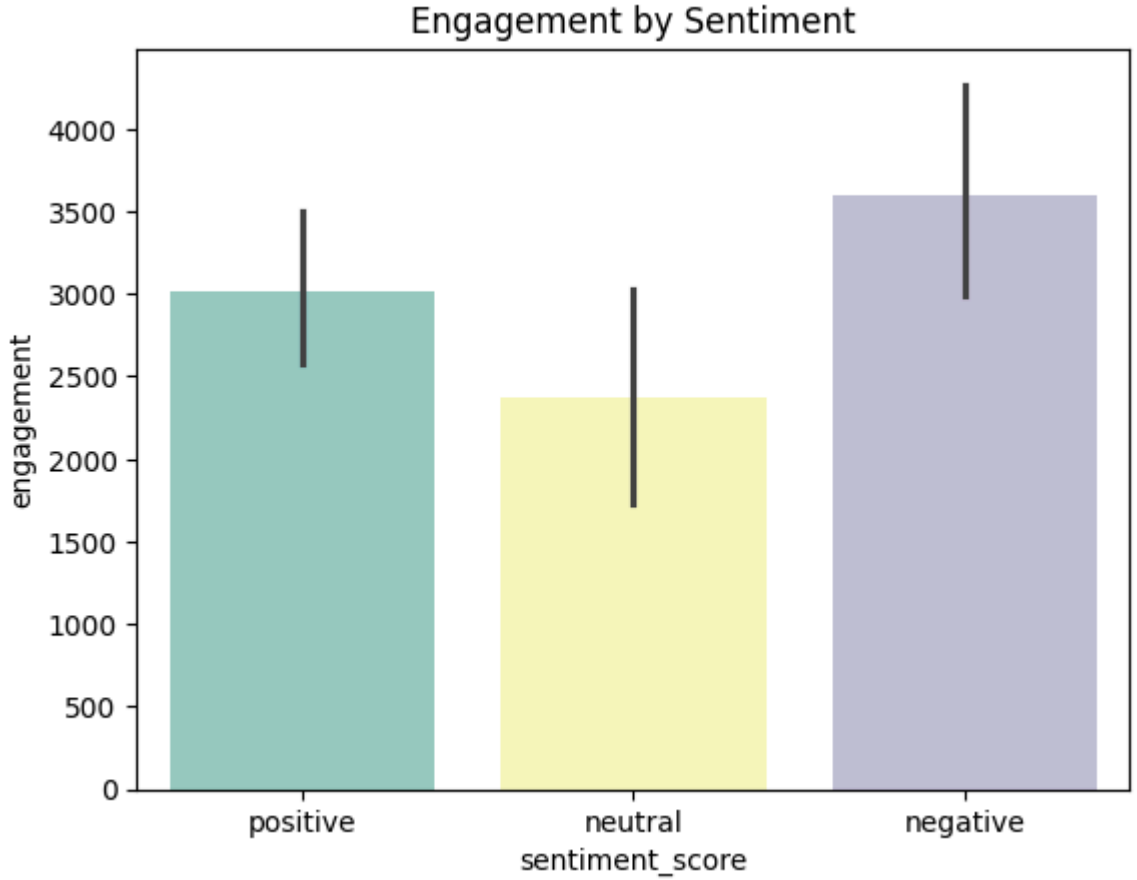
In [22]:
```
sns.boxplot(
    data=df,
    x="post_type",
    y="engagement",
    palette="Set3"
)

plt.title("Engagement Distribution by Post Type")
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\santo\AppData\Local\Temp\ipykernel_17880\2858543308.py:1: FutureWarning:

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

  sns.boxplot(
```

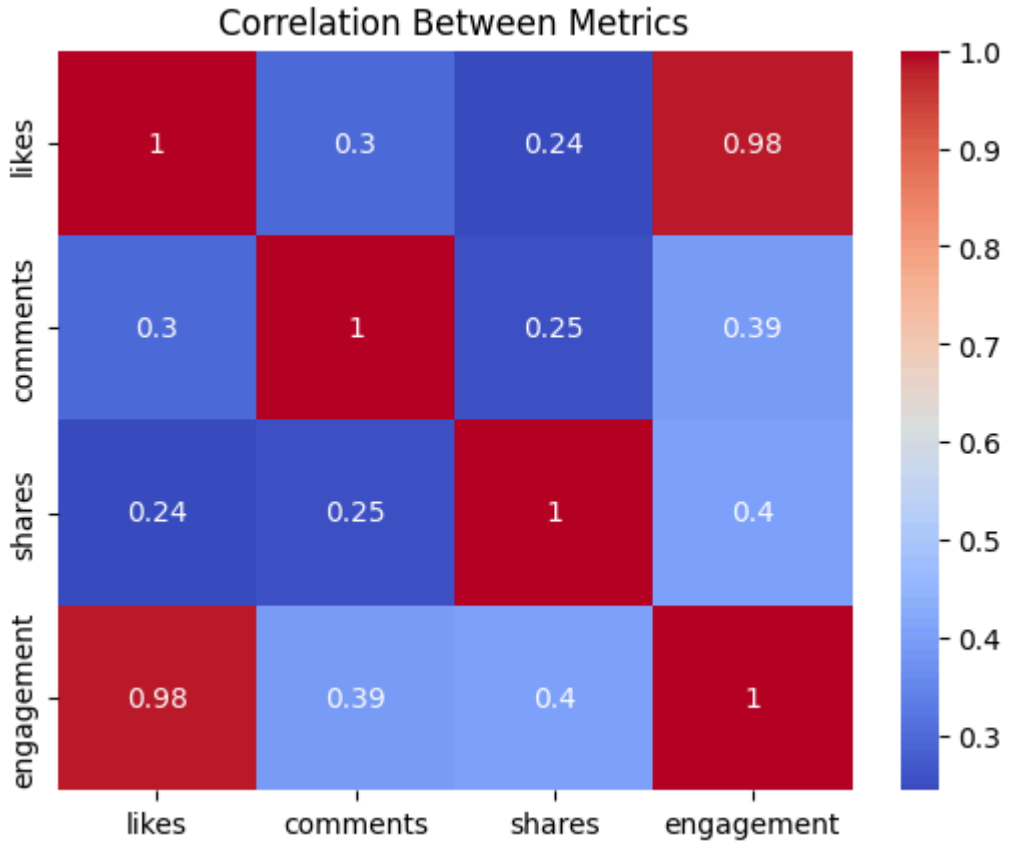


In [25]:
```
sns.barplot(data=df, x="sentiment_score", y="engagement", estimator="mean", palette="Set3")
plt.title("Engagement by Sentiment")
plt.show()
```

```
C:\Users\santo\AppData\Local\Temp\ipykernel_17880\3590280743.py:1: FutureWarning:

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

  sns.barplot(data=df, x="sentiment_score", y="engagement", estimator="mean", palette="Set3")
```



In [26]:
```
sns.heatmap(df[["likes","comments","shares","engagement"]].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Between Metrics")
plt.show()
```

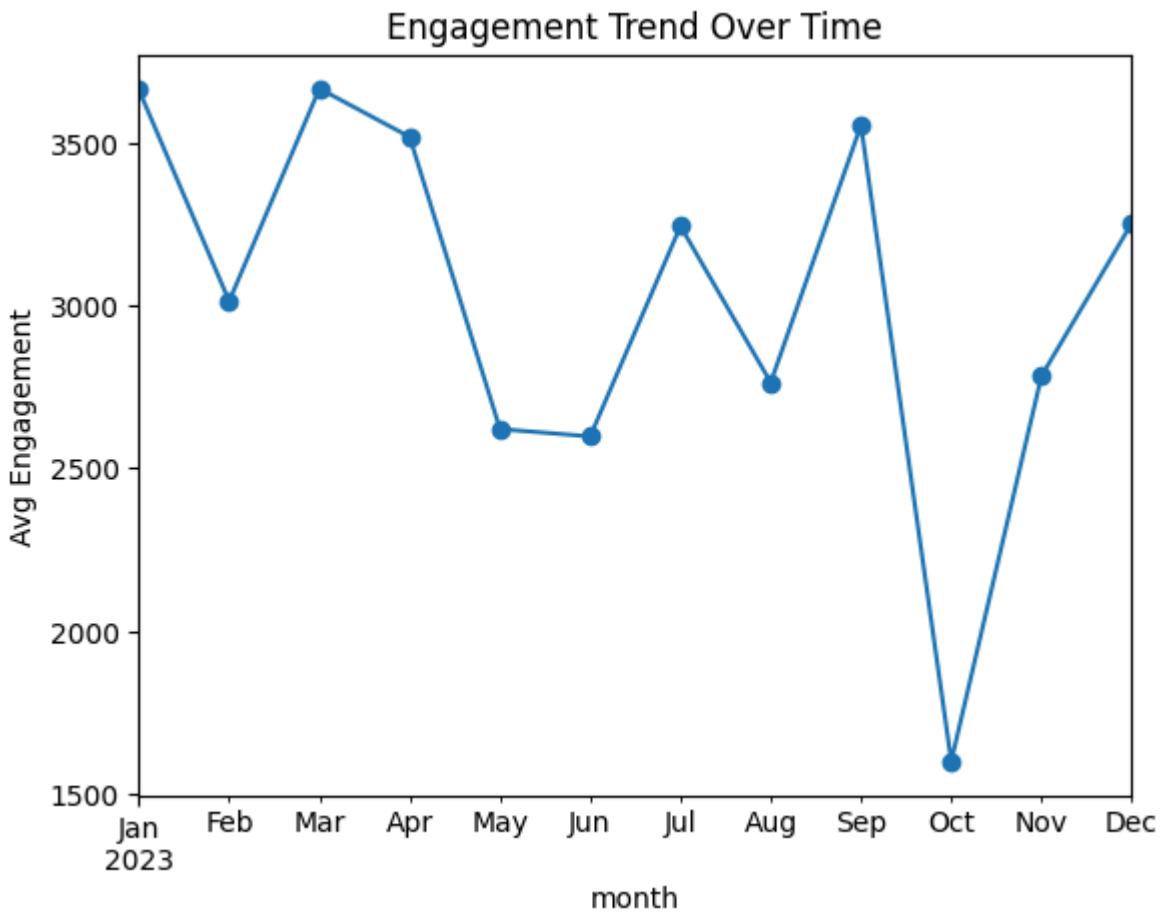


In [28]:
```
top_posts = df.nlargest(5, "engagement")[["post_id","platform","post_type","engagement"]]
print(top_posts)
```

```
    post_id   platform post_type  engagement
38       39   Facebook     video        6410
30       31   Facebook      poll        6222
73       74  Instagram     video        6198
2         3  Instagram      poll        6051
59       60  Instagram  carousel        6051
```

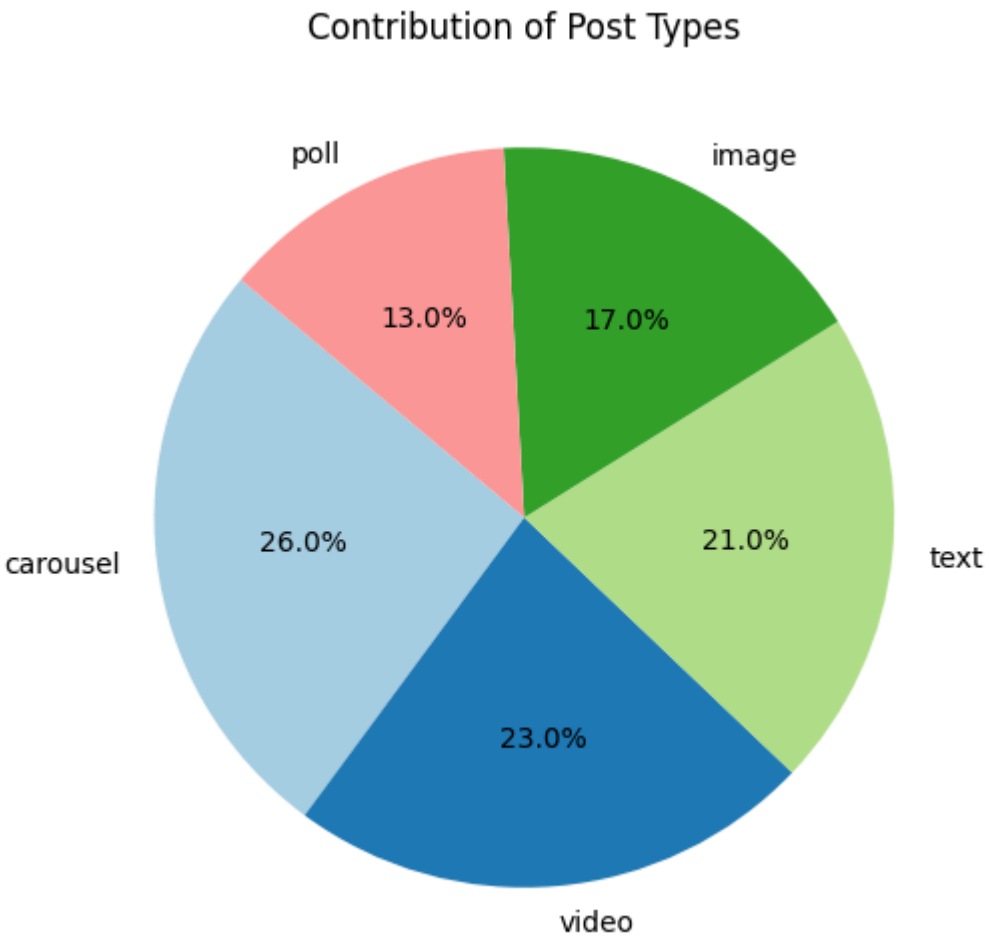
In [29]:
```
df["month"] = df["post_time"].dt.to_period("M")

engagement_month = df.groupby("month")["engagement"].mean()
engagement_month.plot(kind="line", marker="o", title="Engagement Trend Over Time")
```

```
plt.ylabel("Avg Engagement")
plt.show()
```



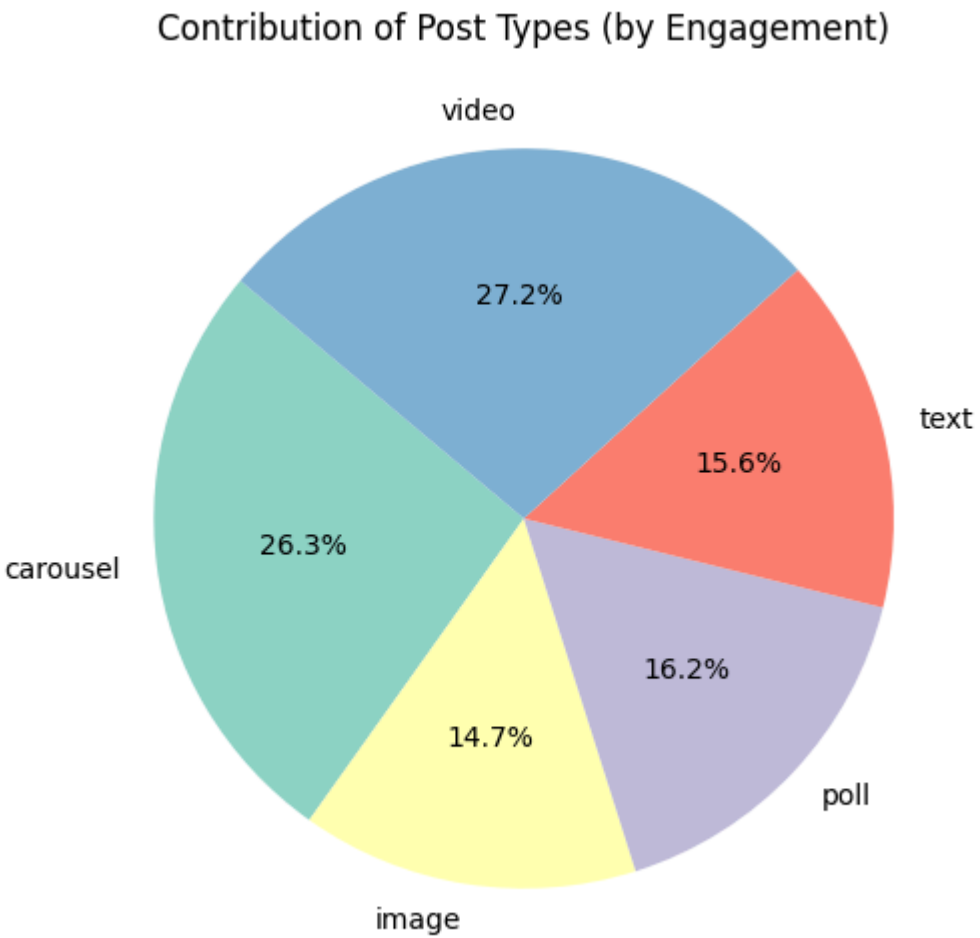
In [30]:
```
import matplotlib.pyplot as plt

# Count each post_type
post_type_counts = df["post_type"].value_counts()

# Pie chart
plt.figure(figsize=(6,6))
plt.pie(
    post_type_counts,
```

```
    labels=post_type_counts.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=plt.cm.Paired.colors
)
plt.title("Contribution of Post Types")
plt.show()
```



In [31]:
```
post_type_engagement = df.groupby("post_type")["engagement"].sum()
```

```
plt.figure(figsize=(6,6))
plt.pie(
    post_type_engagement,
    labels=post_type_engagement.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=plt.cm.Set3.colors
)
plt.title("Contribution of Post Types (by Engagement)")
plt.show()
```



Summary of the data :

from IPython.display import Markdown, display

summary_md = """

Key Insights

1. Videos and polls are the best-performing formats across platforms.

2. Instagram drives the highest overall engagement, making it the strongest platform for visual and interactive content.

3. Facebook polls and videos also generate high engagement, showing that interactive content resonates strongly there.

4. Carousels are particularly valuable for shares, amplifying content reach.

5. Twitter's lower performance suggests it may not be the best platform for maximizing engagement with this content mix.

6. Best posting times: 1 AM, 10 AM, and 9 PM are optimal for maximizing engagement.

In [ ]: