Exploratory Data Analysis (EDA) Report

1. Data Overview

Summary Statistics:

- **Number of Entries**: 10,000
- **Columns**: 10
- **Missing Values**:
  - `text_original`: 2,387 missing (24%)
  - `text_additional`: 9,997 missing (almost entirely missing)
  - `shares_count`: 5,000 missing (50%)
  - `views_count`: 4,379 missing (43.8%)

Data Types:

- **Categorical**: `platform`, `account_id`, `id`
- **Datetime (To be converted)**: `created_time`
- **Numerical**: `likes_count`, `shares_count`, `comments_count`, `views_count`

[32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   platform         10000 non-null  object 
 1   account_id       10000 non-null  object 
 2   id               10000 non-null  object 
 3   created_time     10000 non-null  object 
 4   text_original    7613 non-null   object 
 5   text_additional  3 non-null      object 
 6   likes_count      9998 non-null   float64
 7   shares_count     5000 non-null   float64
 8   comments_count   9955 non-null   float64
 9   views_count      5621 non-null   float64
dtypes: float64(4), object(6)
memory usage: 781.4+ KB
```

2. Data Cleaning & Preprocessing

Handling Missing Values:

- `text_additional` dropped due to extreme sparsity.
- `text_original` missing values left as-is (could be an indicator of non-text-based posts).
- `shares_count`, `views_count`: Missing values replaced with `0` .
- `likes_count`, `comments_count`: Missing values filled with median values.

Data Type Conversions:

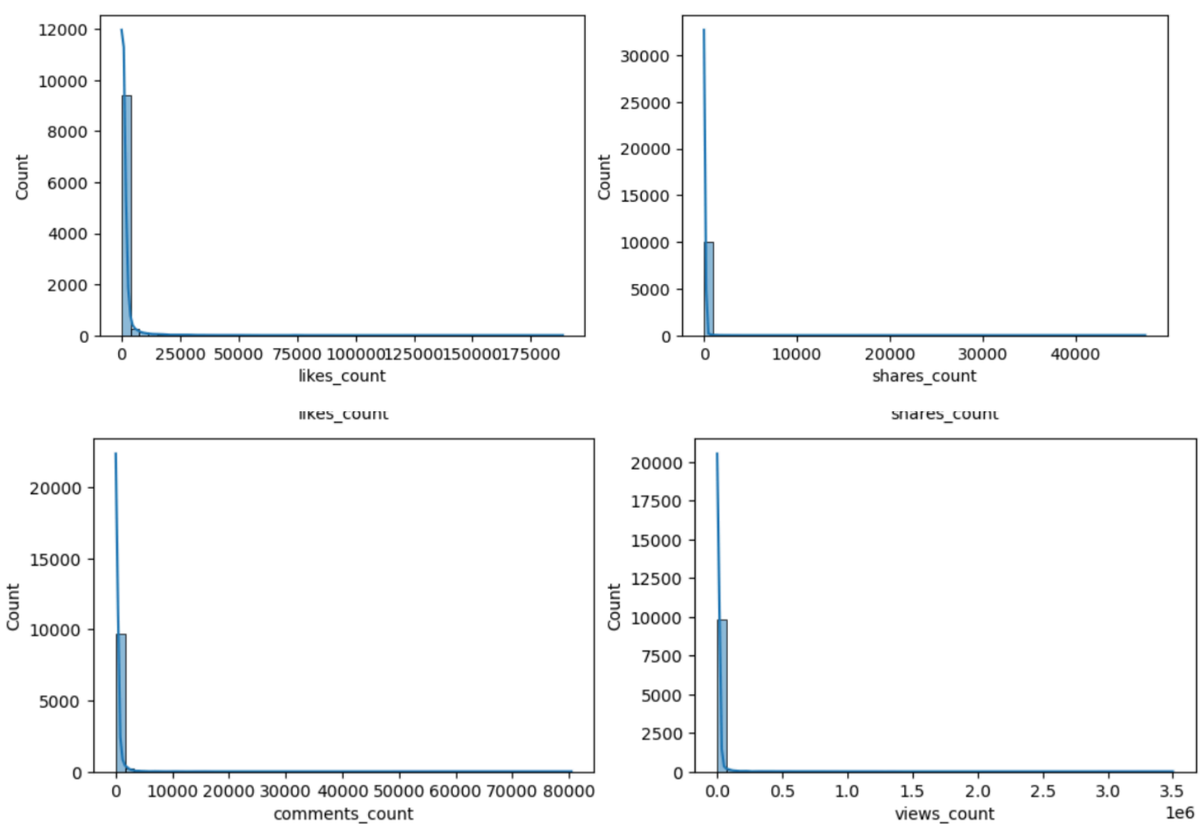
```
df['created_time'] = pd.to_datetime(df['created_time'])
```

```
df['text_additional'] = df['text_additional'].dropna()
df['likes_count'] = df['likes_count'].fillna(df['likes_count'].median())
df['shares_count'] = df['shares_count'].fillna(0)
df['comments_count'] = df['comments_count'].fillna(df['comments_count'].median())
df['views_count'] = df['views_count'].fillna(0)

df['likes_count'] = df['likes_count'].astype(int)
df['shares_count'] = df['shares_count'].astype(int)
df['comments_count'] = df['comments_count'].astype(int)
df['views_count'] = df['views_count'].astype(int)
```

3. Exploratory Analysis

Distribution of Engagement Metrics

```python
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 2, figsize=(12, 8))
sns.histplot(df['likes_count'], bins=50, kde=True, ax=axes[0, 0])
sns.histplot(df['shares_count'], bins=50, kde=True, ax=axes[0, 1])
sns.histplot(df['comments_count'], bins=50, kde=True, ax=axes[1, 0])
sns.histplot(df['views_count'], bins=50, kde=True, ax=axes[1, 1])
plt.show()
```

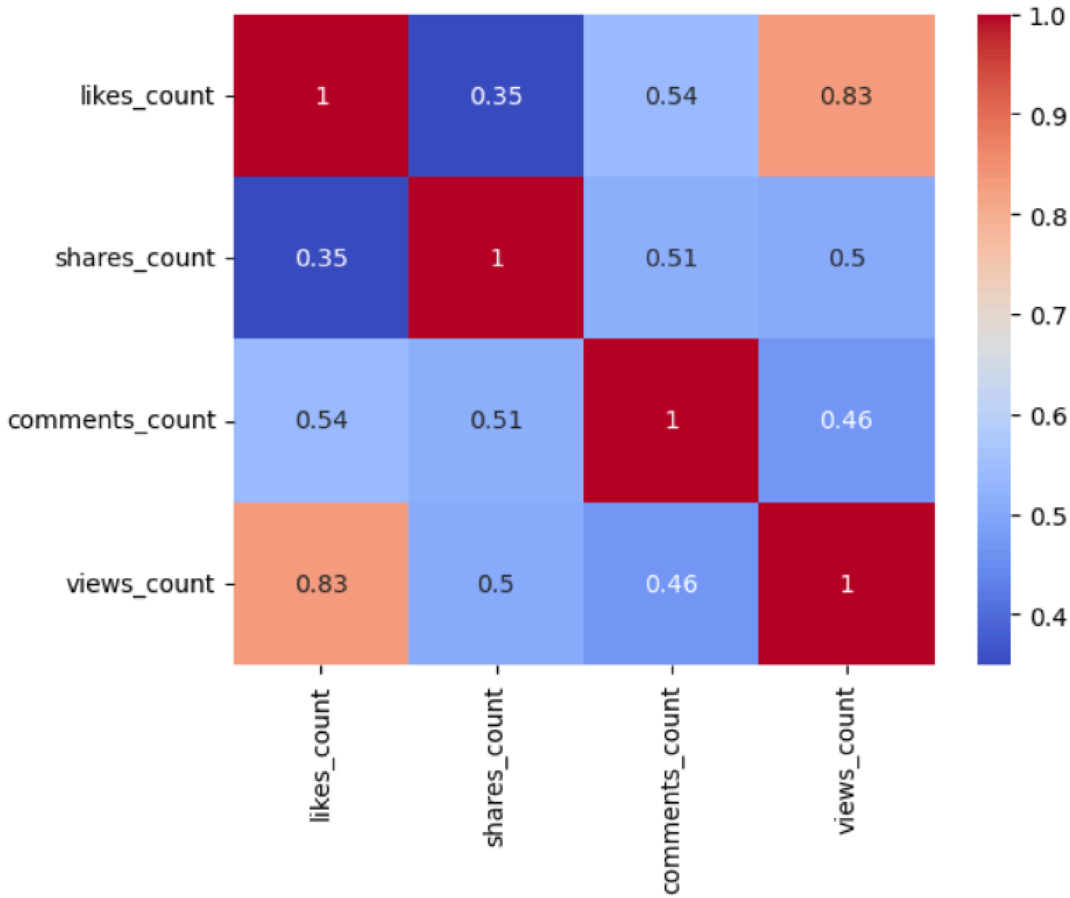


Findings:

- Likes and comments are extremely right-skewed (many low values, few high values).
- Shares are sparser than likes or comments.

Correlation Analysis

```
corr_matrix = df[['likes_count', 'shares_count', 'comments_count', 'views_count']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

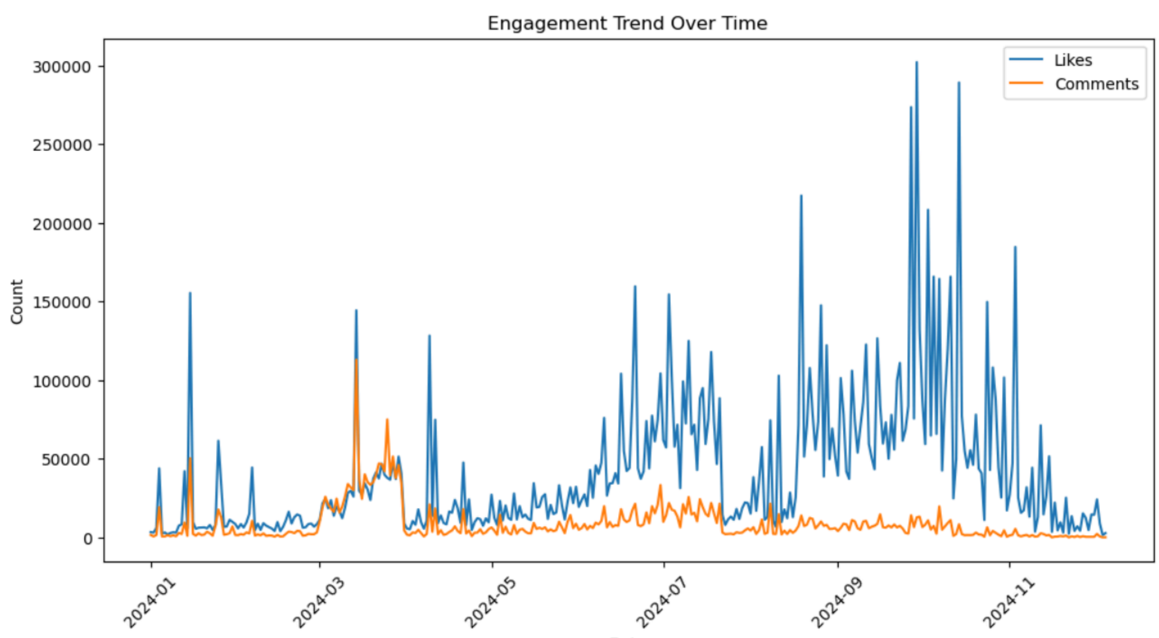


Findings:

- `views_count` and `likes_count` show a strong correlation.
- `shares_count` is weakly correlated with other engagement metrics.
- `comments_count` shows weak to moderate correlation with other metrics.

Engagement Trends Over Time

```
# Aggregating by day, excluding datetime from summation
engagement_trend = df.groupby(df['created_time'].dt.date).agg({
    'likes_count': 'sum',
    'shares_count': 'sum',
    'comments_count': 'sum',
    'views_count': 'sum'
})


plt.figure(figsize=(12, 6))
plt.plot(engagement_trend.index, engagement_trend['likes_count'], label='Likes')
plt.plot(engagement_trend.index, engagement_trend['comments_count'], label='Comments')
plt.legend()
plt.title('Engagement Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

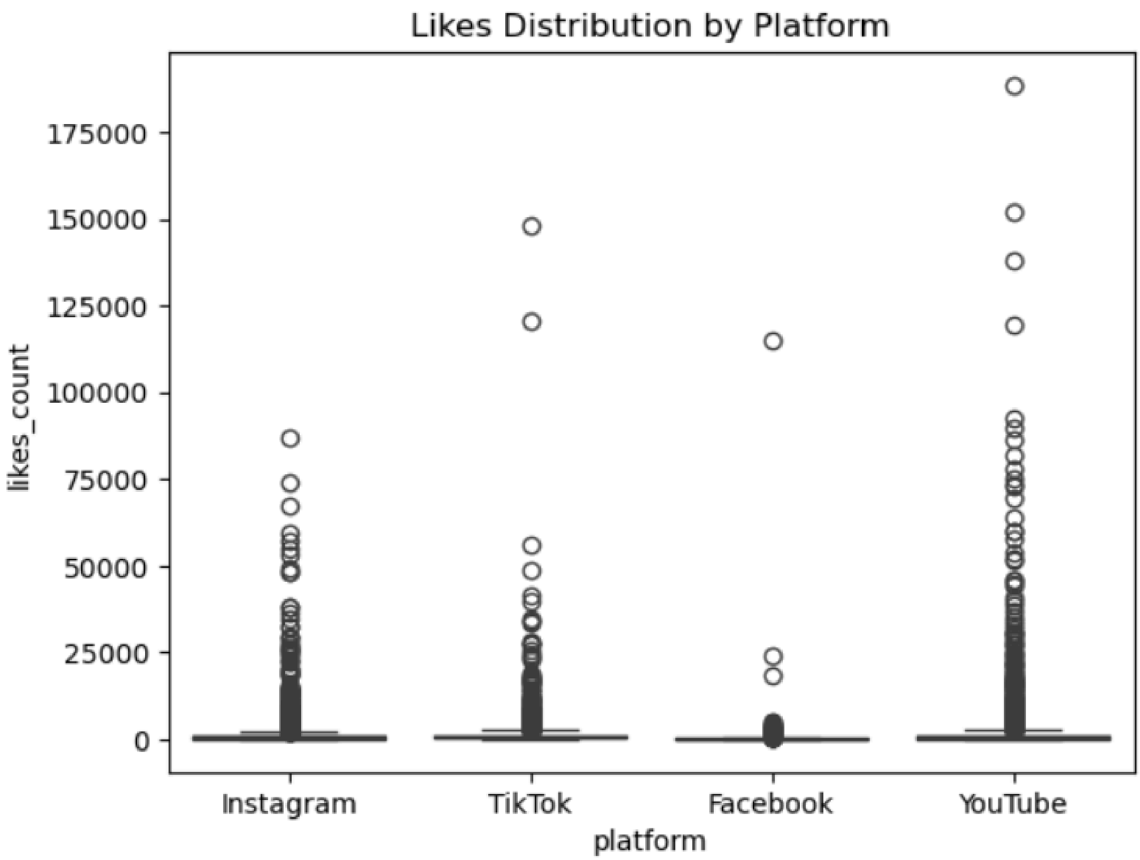


Findings:

- Engagement fluctuates over time, with noticeable peaks.
- Potential To Do: explore months with peak engagement to determine the cause of a peak or adjust content plan/adds shown according to a trend.

Platform-Wise Analysis

```
sns.boxplot(x=df['platform'], y=df['likes_count'])
plt.title('Likes Distribution by Platform')
plt.show()
```



Findings:

- Some platforms, like YouTube or Instagram have higher median engagement than others.

4. Key Insights & Summary

- **Engagement Metrics**: Likes and views are strongly correlated, while shares and comments are less related.
- **Time Trends**: Peaks in engagement suggest periodic trends.

- **Platform Differences**: More popular platforms like YouTube or Instagram have higher engagement levels than others. Short videos from TikTok show less engagement than longer content.
- **Data Issues**: Missing values in `shares_count` and `views_count` should be investigated further.

Next Steps

- Deeper analysis of engagement patterns across different times of day.
- Examine content types to see if certain formats perform better.
- Explore months with peak engagement to determine the cause of a peak or adjust content plan/adds shown according to a trend
- Use machine learning to predict post-engagement.