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

In [3]:
```
df = pd.read_csv(r'C:\Users\abhiw\Downloads\USvideos.csv')
```

In [9]:
```
df.head(2)
```

Out[9]:

	video_id	trending_date	title	channel_title	category_id	publish_time	
0	2kyS6SvSYSE	17.14.11	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	SH
1	1ZAPwfrtAFY	17.14.11	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	las tonight presiden

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   video_id                40949 non-null  object
 1   trending_date           40949 non-null  object
 2   title                   40949 non-null  object
 3   channel_title           40949 non-null  object
 4   category_id             40949 non-null  int64 
 5   publish_time            40949 non-null  object
 6   tags                    40949 non-null  object
 7   views                   40949 non-null  int64 
 8   likes                   40949 non-null  int64 
 9   dislikes                40949 non-null  int64 
 10  comment_count           40949 non-null  int64 
 11  thumbnail_link          40949 non-null  object
 12  comments_disabled       40949 non-null  bool  
 13  ratings_disabled        40949 non-null  bool  
 14  video_error_or_removed  40949 non-null  bool  
 15  description             40379 non-null  object
dtypes: bool(3), int64(5), object(8)
memory usage: 4.2+ MB
```

In [13]:
```
df.describe()
```

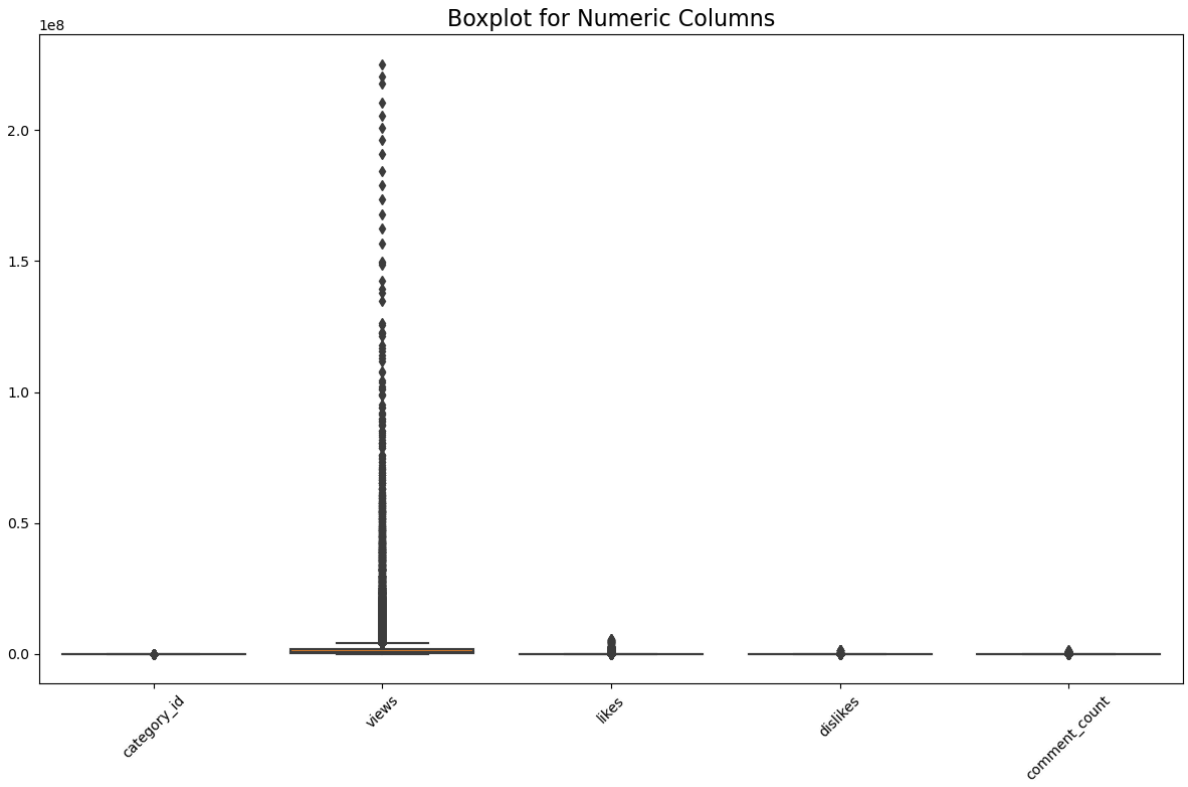
Out[13]:

	category_id	views	likes	dislikes	comment_count
count	40949.000000	4.094900e+04	4.094900e+04	4.094900e+04	4.094900e+04
mean	19.972429	2.360785e+06	7.426670e+04	3.711401e+03	8.446804e+03
std	7.568327	7.394114e+06	2.288853e+05	2.902971e+04	3.743049e+04
min	1.000000	5.490000e+02	0.000000e+00	0.000000e+00	0.000000e+00
25%	17.000000	2.423290e+05	5.424000e+03	2.020000e+02	6.140000e+02
50%	24.000000	6.818610e+05	1.809100e+04	6.310000e+02	1.856000e+03
75%	25.000000	1.823157e+06	5.541700e+04	1.938000e+03	5.755000e+03
max	43.000000	2.252119e+08	5.613827e+06	1.674420e+06	1.361580e+06

In [17]:
```
df.isnull().sum()
```

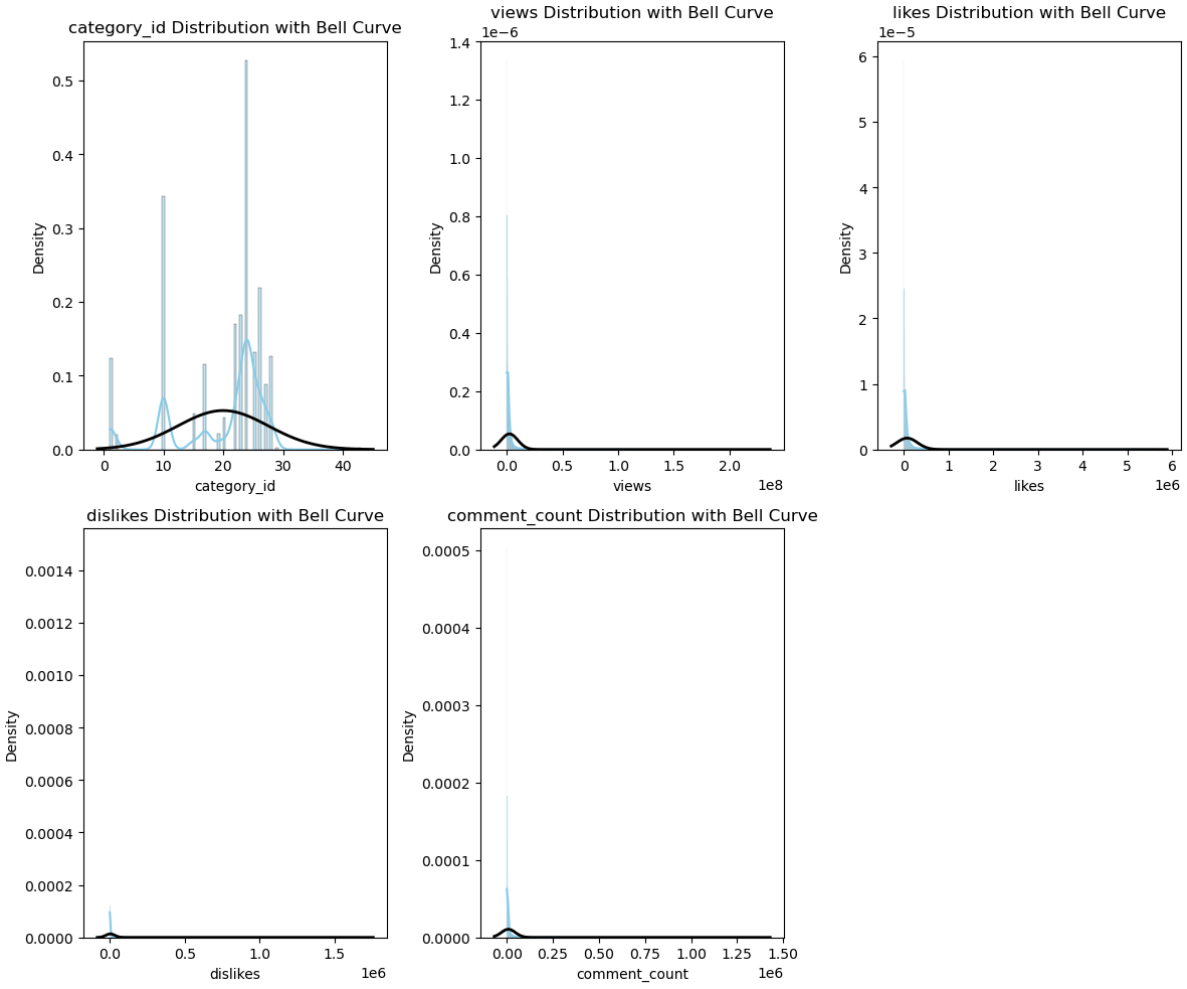
Out[17]:
```
video_id                  0
trending_date             0
title                     0
channel_title             0
category_id               0
publish_time              0
tags                      0
views                     0
likes                     0
dislikes                  0
comment_count             0
thumbnail_link            0
comments_disabled         0
ratings_disabled          0
video_error_or_removed    0
description             570
dtype: int64
```

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# List of numeric columns
numeric_columns = ['category_id', 'views', 'likes', 'dislikes', 'comment_count']

# Create boxplot for all numeric columns
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[numeric_columns])

# Set title and labels
plt.title('Boxplot for Numeric Columns', fontsize=16)
plt.xticks(rotation=45)
plt.tight_layout()

# Display plot
plt.show()
```



In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# List of numeric columns
numeric_columns = ['category_id', 'views', 'likes', 'dislikes', 'comment_count']

# Set the size of the plot
plt.figure(figsize=(12, 10))

# Loop through each numeric column to plot histograms with a bell curve
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[col], kde=True, stat="density", color='skyblue')

    # Fit and plot a normal distribution curve (bell curve)
    mu, std = norm.fit(df[col].dropna())  # Fit the data to a normal distribution
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mu, std)  # Probability density function
    plt.plot(x, p, 'k', linewidth=2)  # Plot the bell curve

    plt.title(f'{col} Distribution with Bell Curve', fontsize=12)
    plt.tight_layout()

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

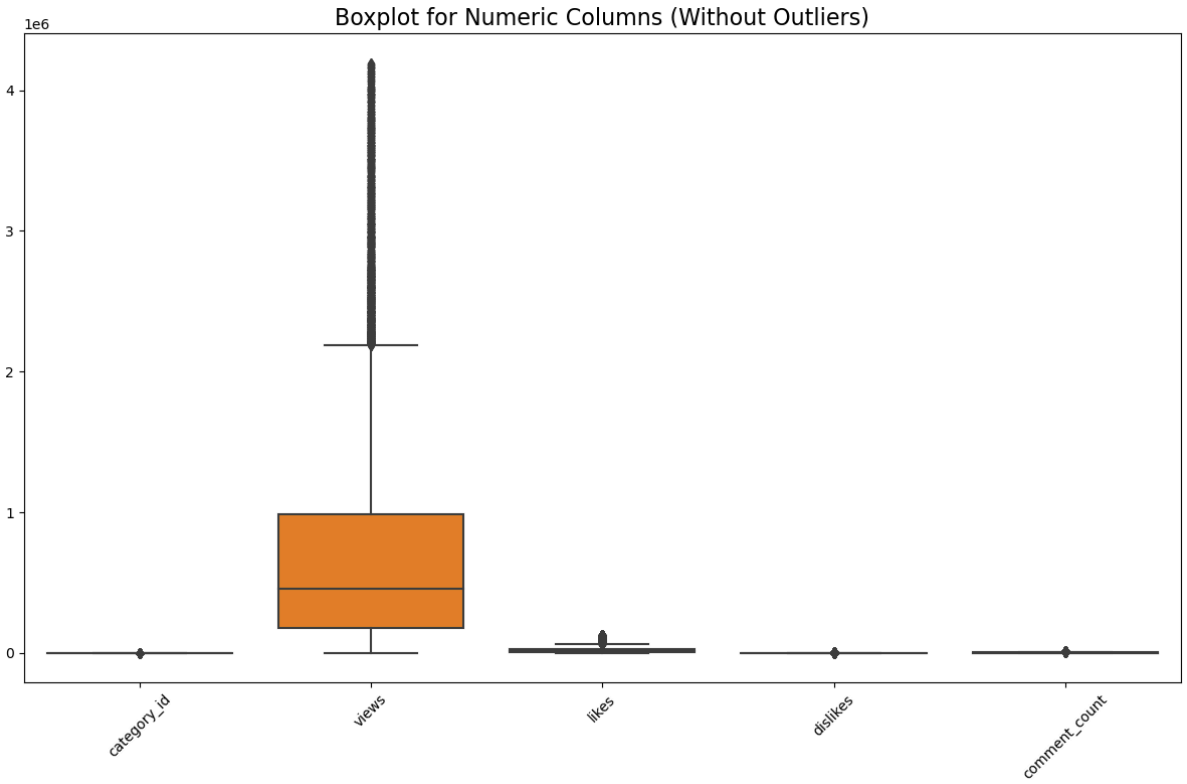
```
C:\Users\abhiw\AppData\Local\Temp\ipykernel_26328\3076963516.py:25: UserWarning: T
he figure layout has changed to tight
  plt.tight_layout()
C:\Users\abhiw\AppData\Local\Temp\ipykernel_26328\3076963516.py:25: UserWarning: T
he figure layout has changed to tight
  plt.tight_layout()
C:\Users\abhiw\AppData\Local\Temp\ipykernel_26328\3076963516.py:25: UserWarning: T
he figure layout has changed to tight
  plt.tight_layout()
C:\Users\abhiw\AppData\Local\Temp\ipykernel_26328\3076963516.py:25: UserWarning: T
he figure layout has changed to tight
  plt.tight_layout()
```



In [23]:
```
# Calculate Q1 (25th percentile) and Q3 (75th percentile) for each column
Q1 = df[numeric_columns].quantile(0.25)
Q3 = df[numeric_columns].quantile(0.75)

# Calculate the IQR (Interquartile Range)
IQR = Q3 - Q1

# Define the lower and upper bounds for each column
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the DataFrame to remove outliers
df_no_outliers = df[~((df[numeric_columns] < lower_bound) | (df[numeric_columns] >

# Check the shape of the new DataFrame
print(f"Original shape: {df.shape}")
print(f"Shape after removing outliers: {df_no_outliers.shape}")
```

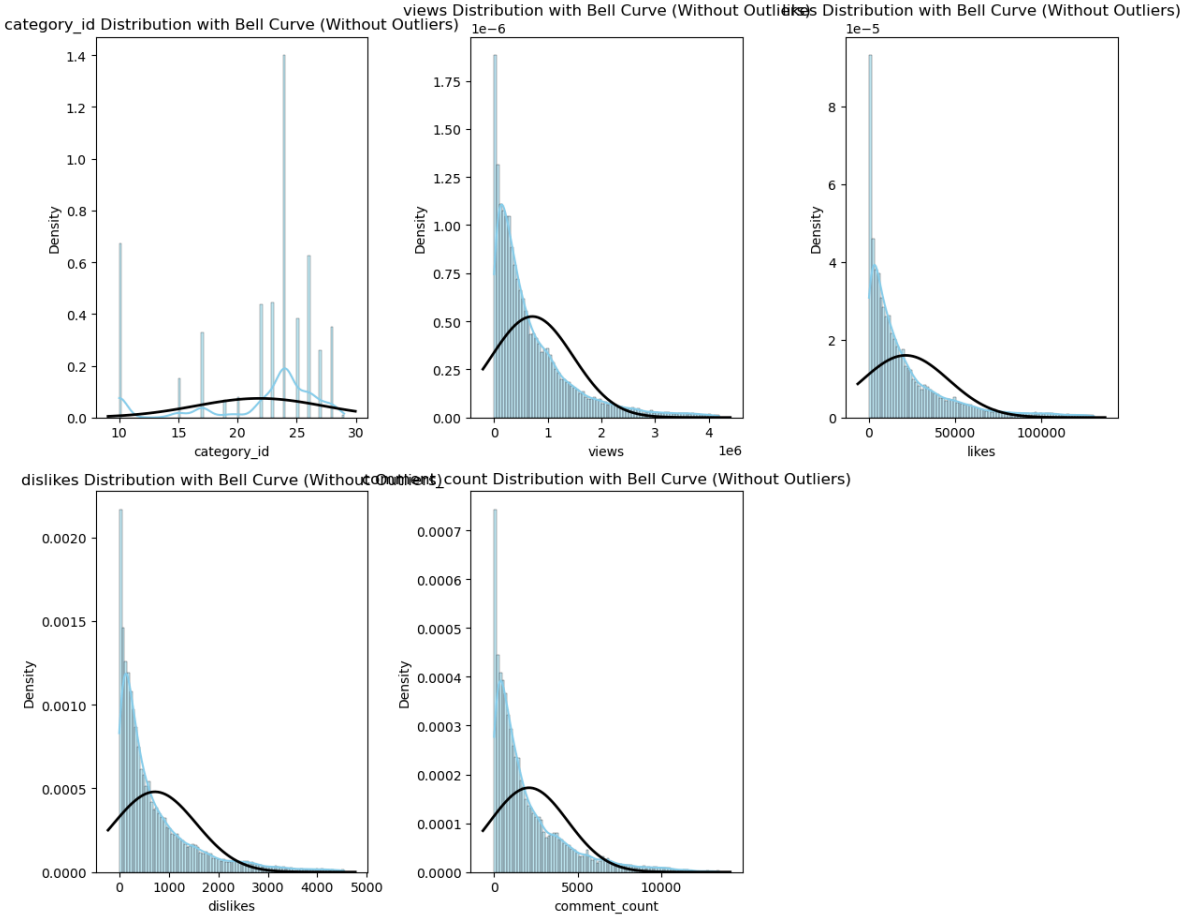
```
Original shape: (40949, 16)
Shape after removing outliers: (30299, 16)
```

In [25]:
```
# Recreate the DataFrame after removing outliers using IQR
Q1 = df[numeric_columns].quantile(0.25)
Q3 = df[numeric_columns].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for each column
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the DataFrame to remove outliers
df_no_outliers = df[~((df[numeric_columns] < lower_bound) | (df[numeric_columns] >

# Create boxplot for all numeric columns after removing outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=df_no_outliers[numeric_columns])

# Set title and labels
plt.title('Boxplot for Numeric Columns (Without Outliers)', fontsize=16)
plt.xticks(rotation=45)
plt.tight_layout()

# Display plot
plt.show()
```



In [27]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# Set the size of the plot
plt.figure(figsize=(12, 10))

# Loop through each numeric column to plot histograms with a bell curve after remov
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df_no_outliers[col], kde=True, stat="density", color='skyblue')

    # Fit and plot a normal distribution curve (bell curve)
```

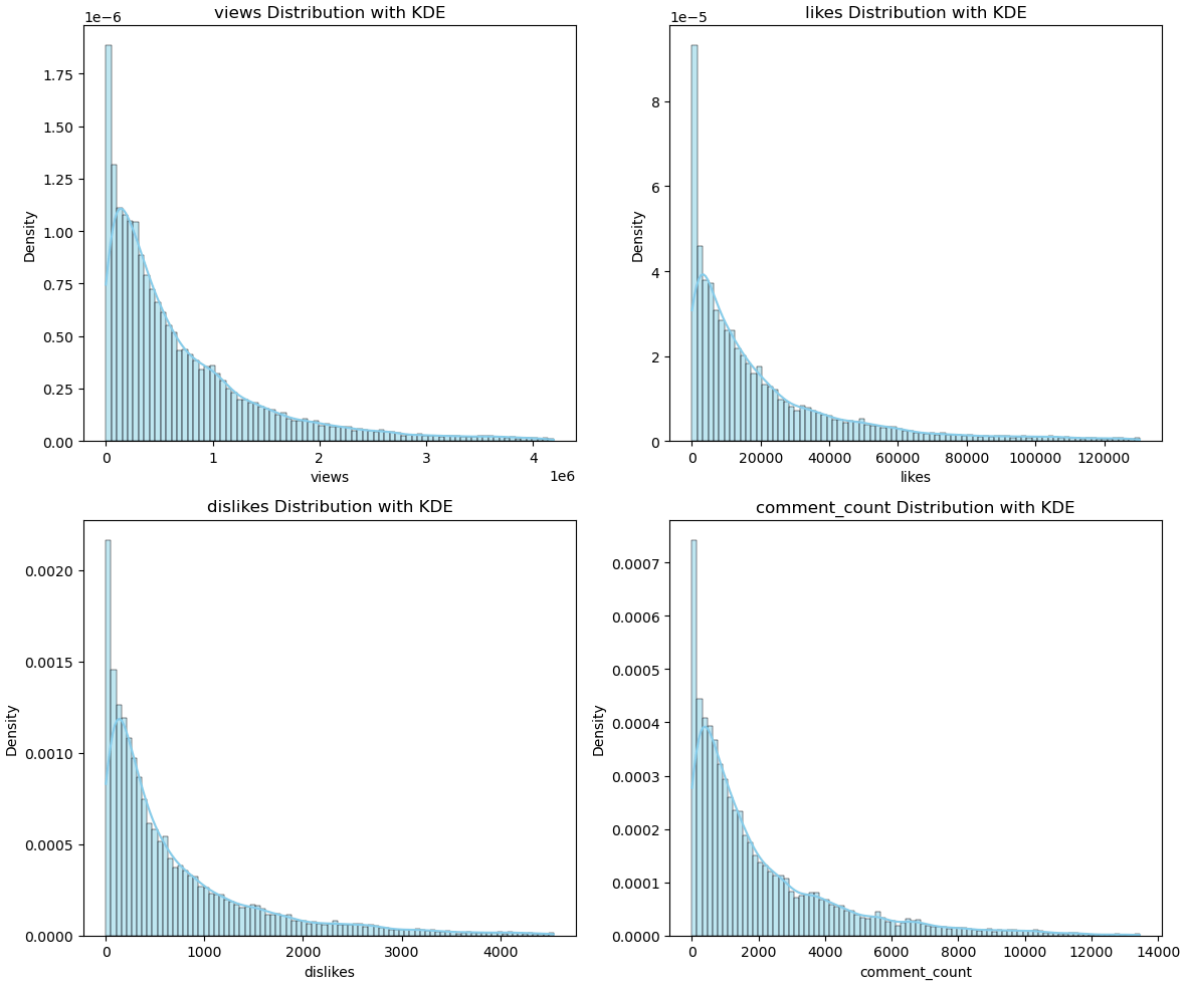
```
    mu, std = norm.fit(df_no_outliers[col].dropna())  # Fit the data to a normal di
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mu, std)  # Probability density function
    plt.plot(x, p, 'k', linewidth=2)  # Plot the bell curve

    plt.title(f'{col} Distribution with Bell Curve (Without Outliers)', fontsize=12

# Adjust layout
plt.tight_layout()

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



In [37]:
```
#Distribution of Views, Likes, Dislikes, and Comment Counts
```

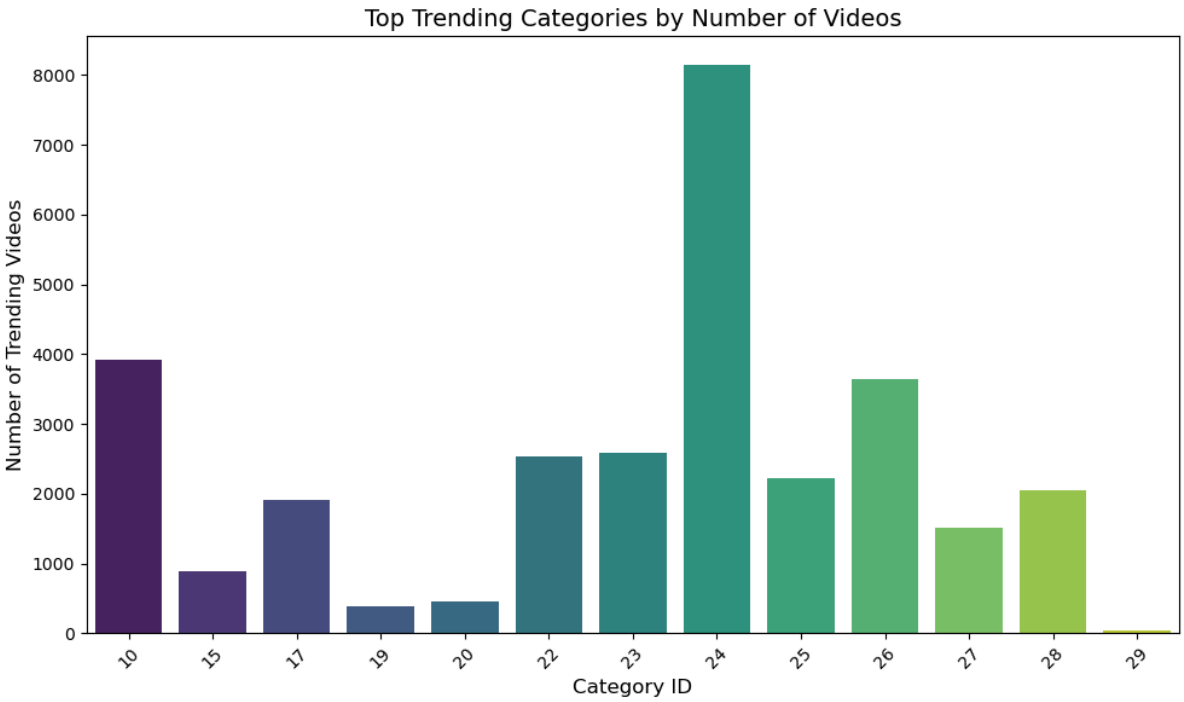
In [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the size of the plot
plt.figure(figsize=(12, 10))

# Loop through each numeric column to plot histogram and KDE
numeric_columns_to_plot = ['views', 'likes', 'dislikes', 'comment_count']

for i, col in enumerate(numeric_columns_to_plot, 1):
    plt.subplot(2, 2, i)
    sns.histplot(df_no_outliers[col], kde=True, stat="density", color='skyblue')
    plt.title(f'{col} Distribution with KDE', fontsize=12)

# Adjust layout
plt.tight_layout()
```

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



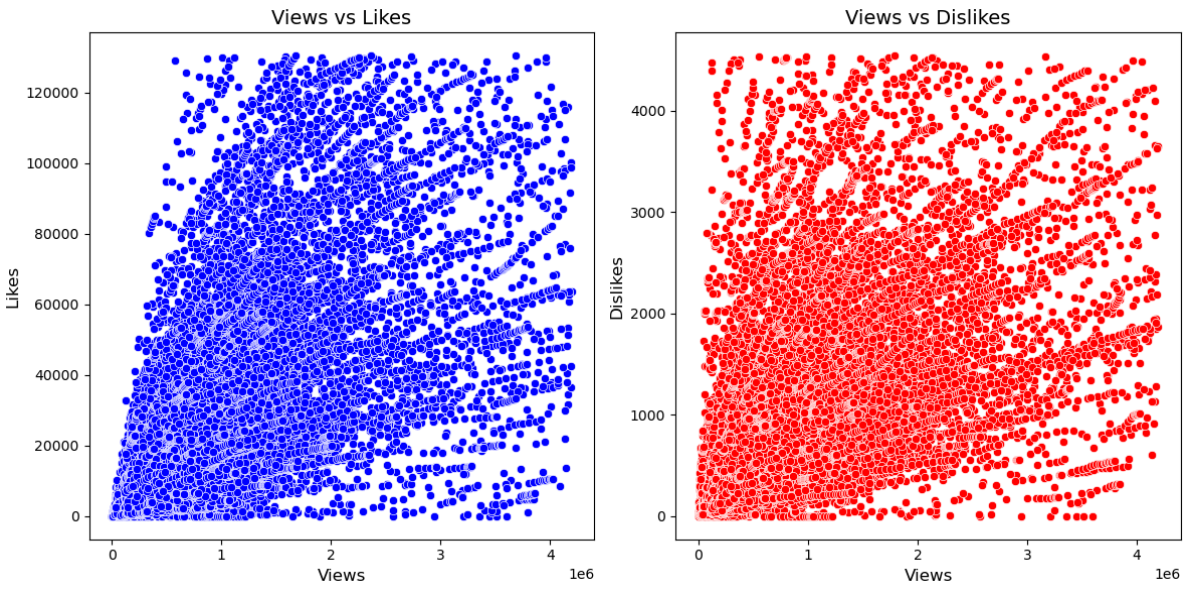
In [41]:
```
#2. Top Trending Categories
```

In [43]:
```
# Count the number of trending videos per category
category_counts = df_no_outliers['category_id'].value_counts()

# Create a bar plot to show the top categories by number of trending videos
plt.figure(figsize=(10, 6))
sns.barplot(x=category_counts.index, y=category_counts.values, palette='viridis')

# Set title and labels
plt.title('Top Trending Categories by Number of Videos', fontsize=14)
plt.xlabel('Category ID', fontsize=12)
plt.ylabel('Number of Trending Videos', fontsize=12)
plt.xticks(rotation=45)

# Display plot
plt.tight_layout()
plt.show()
```



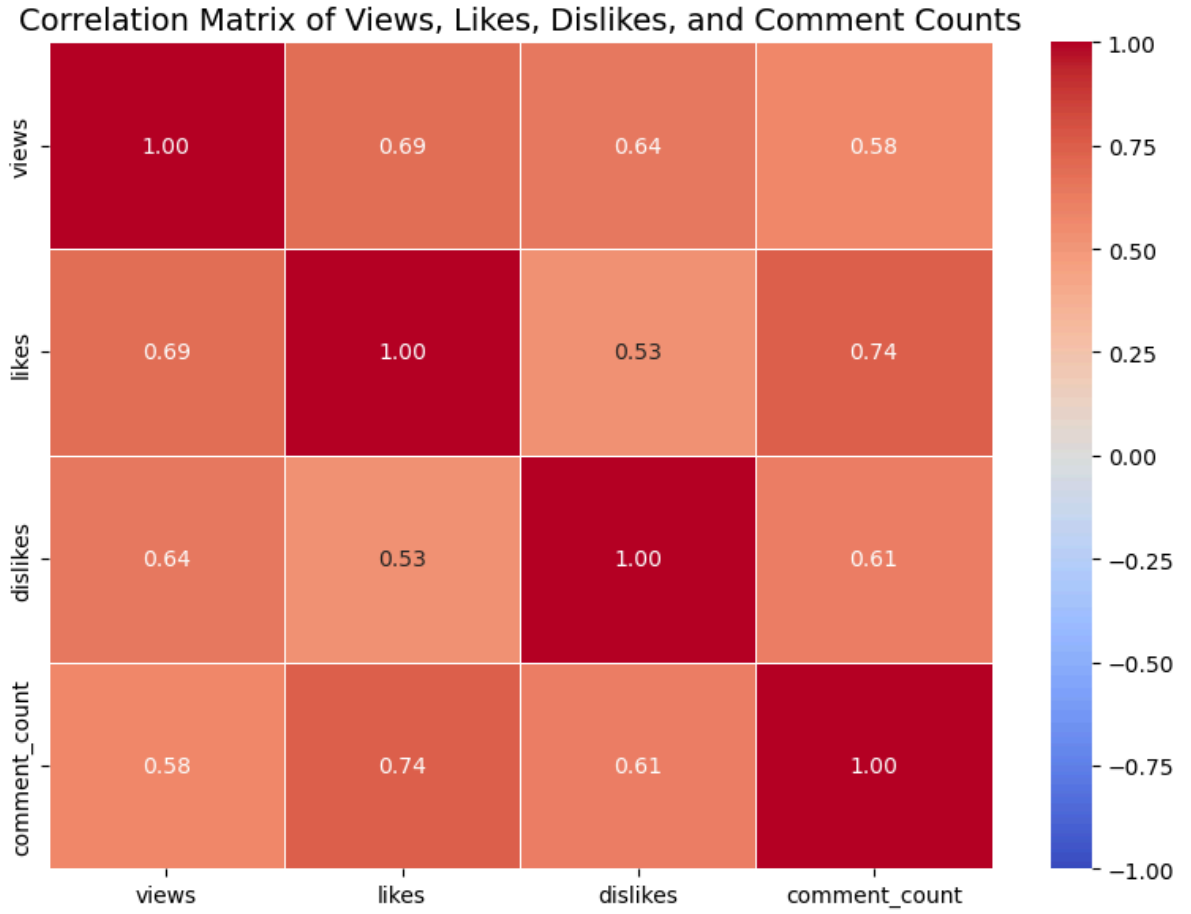
In [45]:
```
#3. Views vs Likes vs Dislikes
```

In [47]:
```
# Scatter plot of Views vs Likes
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(data=df_no_outliers, x='views', y='likes', color='blue')
plt.title('Views vs Likes', fontsize=14)
plt.xlabel('Views', fontsize=12)
plt.ylabel('Likes', fontsize=12)

# Scatter plot of Views vs Dislikes
plt.subplot(1, 2, 2)
sns.scatterplot(data=df_no_outliers, x='views', y='dislikes', color='red')
plt.title('Views vs Dislikes', fontsize=14)
plt.xlabel('Views', fontsize=12)
plt.ylabel('Dislikes', fontsize=12)

# Display plot
plt.tight_layout()
plt.show()
```



In [53]:
```
#4. Correlation Matrix
```

In [55]:
```
# Calculate the correlation matrix
corr_matrix = df_no_outliers[['views', 'likes', 'dislikes', 'comment_count']].corr(

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5, vm

# Set title
plt.title('Correlation Matrix of Views, Likes, Dislikes, and Comment Counts', fonts

# Display plot
plt.tight_layout()
plt.show()
```

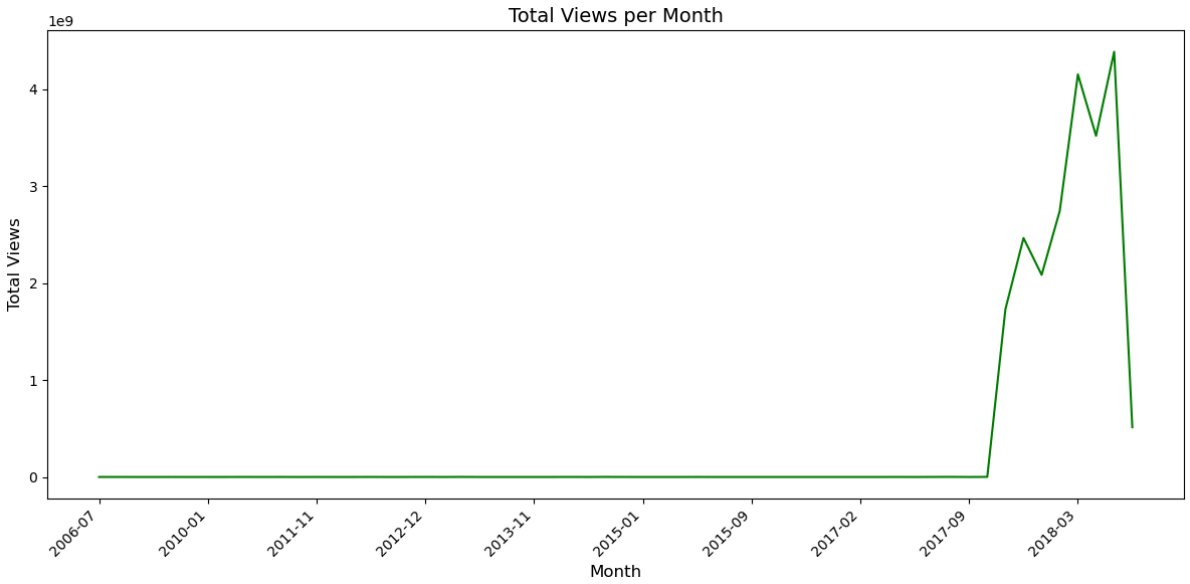


In [57]:
```
#Time-based Trends (Monthly or Yearly Trends)
```

In [61]:
```
# Use .loc to avoid SettingWithCopyWarning
df_no_outliers.loc[:, 'publish_time'] = pd.to_datetime(df_no_outliers['publish_time

# Convert 'publish_time' to Period and handle any timezone issues
df_no_outliers.loc[:, 'year_month'] = df_no_outliers['publish_time'].dt.to_period('

# Calculate the total views per month
monthly_views = df_no_outliers.groupby('year_month')['views'].sum()

# Plot the views over time
plt.figure(figsize=(12, 6))
sns.lineplot(x=monthly_views.index.astype(str), y=monthly_views.values, color='gree

# Set title and labels
plt.title('Total Views per Month', fontsize=14)
```

```
plt.xlabel('Month', fontsize=12)
plt.ylabel('Total Views', fontsize=12)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Reduce the number of ticks on the x-axis (optional)
plt.xticks(ticks=range(0, len(monthly_views), 6), labels=monthly_views.index[::6])

# Display plot
plt.tight_layout()
plt.show()
```

```
C:\Users\abhiw\AppData\Local\Temp\ipykernel_26328\4134611870.py:5: UserWarning: Co
nverting to PeriodArray/Index representation will drop timezone information.
  df_no_outliers.loc[:, 'year_month'] = df_no_outliers['publish_time'].dt.to_perio
d('M')
```



In [ ]: