

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('/content/drive/MyDrive/mdata/youtube_video.csv')
try:
    df
except NameError:
    raise RuntimeError("DataFrame 'df' غير موجود - ضع df = pd.read_csv('data.csv') الكود قبل تنفيذ الكود")
for col in df.columns:
    if df[col].dtype == 'object':
        sample = df[col].dropna().astype(str).head(30)
        convertible = True
        for v in sample:
            try:
                float(v.replace(',', ''))
            except:
                convertible = False
                break
        if convertible:
            df[col] = pd.to_numeric(df[col].astype(str).str.replace(',', ''), errors='coerce')
            print(f"Converted column to numeric: {col}")
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print("Numeric columns detected:", num_cols)
if not num_cols:
    raise RuntimeError("لا توجد أعمدة عددية بعد المحاولة. تأكد من بياناتك أو حدد أسماء الأعمدة العددية يدوياً")
plt.figure(figsize=(8,4))
df[num_cols].boxplot()
plt.title('Boxplot before outlier handling')
plt.tight_layout()
plt.savefig('boxplot_before.png')
print("Saved boxplot before -> boxplot_before.png")
plt.close()
mask_any_outlier = pd.Series(False, index=df.index)

outlier_summary = []
for col in num_cols:
    col_series = df[col]
    if col_series.dropna().empty:
        print(f"Column {col} has only NaN values; skipped.")
        continue
    Q1 = col_series.quantile(0.25)
    Q3 = col_series.quantile(0.75)
    IQR = Q3 - Q1
    if pd.isna(IQR) or IQR == 0:
        print(f"Column {col} has IQR zero or NaN (Q1={Q1}, Q3={Q3}); skipping IQR outlier detection for this column.")
        continue
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    col_mask = (col_series < lower) | (col_series > upper)
    n_out = int(col_mask.sum())
    outlier_summary.append((col, n_out, lower, upper))
    print(f"{col}: outliers={n_out}, lower={lower:.3f}, upper={upper:.3f}")
    # اجمع الأفضنة
    mask_any_outlier = mask_any_outlier | col_mask.fillna(False)
total_outlier_rows = int(mask_any_outlier.sum())
print(f"Total rows with >=1 outlier (any numeric col): {total_outlier_rows} / {df.shape[0]}")
df_dropped = df.loc[~mask_any_outlier].copy()
print("Rows before:", df.shape[0], "Rows after dropping outliers:", df_dropped.shape[0])
plt.figure(figsize=(8,4))
if not df_dropped[num_cols].dropna(how='all').empty:
    df_dropped[num_cols].boxplot()
    plt.title('Boxplot after dropping outliers')
    plt.tight_layout()
    plt.savefig('boxplot_after_dropped.png')
    print("Saved boxplot after dropping -> boxplot_after_dropped.png")
else:
    print("No numeric data remaining after dropping outliers; skipped saving boxplot.")

plt.close()
df_capped = df.copy()
for col in num_cols:
    lower_p = df_capped[col].quantile(0.01)
    upper_p = df_capped[col].quantile(0.99)
    if pd.isna(lower_p) or pd.isna(upper_p):
        continue
    df_capped[col] = np.clip(df_capped[col], lower_p, upper_p)

plt.figure(figsize=(8,4))
df_capped[num_cols].boxplot()
plt.title('Boxplot after capping (1%-99%)')

```

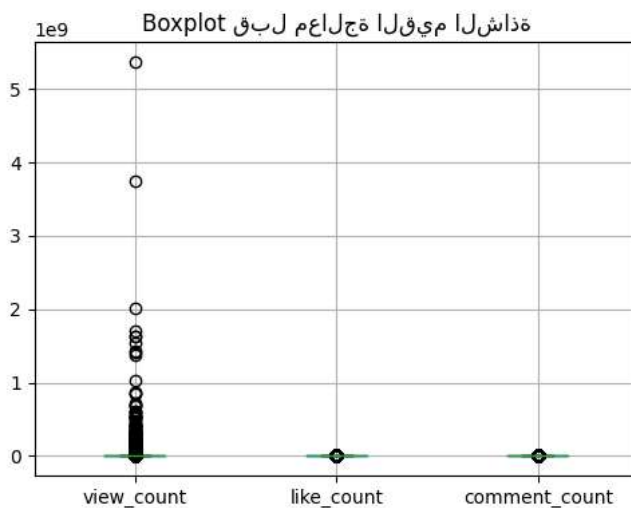
```
plt.title('Boxplot after capping (1% 99%)')
plt.tight_layout()
plt.savefig('boxplot_after_capped.png')
print("Saved boxplot after capping -> boxplot_after_capped.png")
plt.close()
print("\nOutlier details per numeric column:")
for col, n_out, lower, upper in outlier_summary:
    print(f"- {col}: {n_out} outliers; lower={lower:.3f}, upper={upper:.3f}")
```

```
Numeric columns detected: ['view_count', 'like_count', 'comment_count']
Saved boxplot before -> boxplot_before.png
view_count: outliers=24971, lower=-324042.000, upper=558222.000
like_count: outliers=24615, lower=-7986.000, upper=13630.000
comment_count: outliers=23895, lower=-356.000, upper=604.000
Total rows with >=1 outlier (any numeric col): 34598 / 155669
Rows before: 155669 Rows after dropping outliers: 121071
Saved boxplot after dropping -> boxplot_after_dropped.png
Saved boxplot after capping -> boxplot_after_capped.png
```

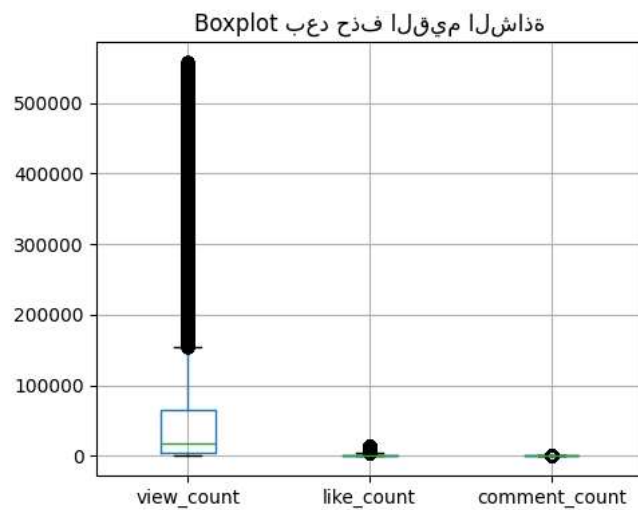
```
Outlier details per numeric column:
- view_count: 24971 outliers; lower=-324042.000, upper=558222.000
- like_count: 24615 outliers; lower=-7986.000, upper=13630.000
- comment_count: 23895 outliers; lower=-356.000, upper=604.000
```

```
import matplotlib.pyplot as plt

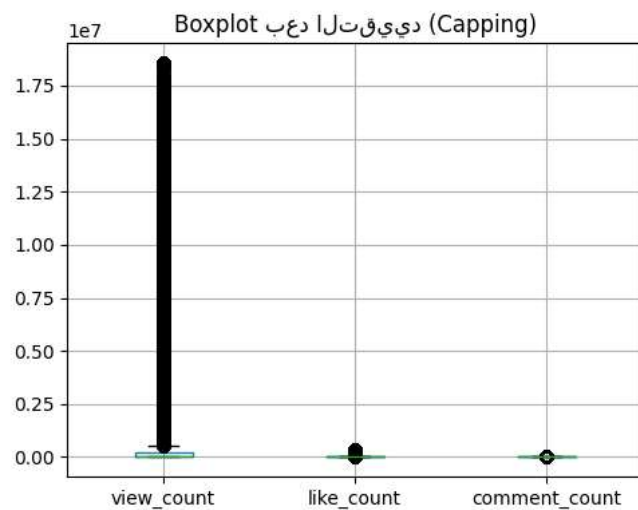
plt.figure(figsize=(5,4))
df[num_cols].boxplot()
plt.title('Boxplot معالجة القيم الشاذة')
plt.tight_layout()
plt.show()
```



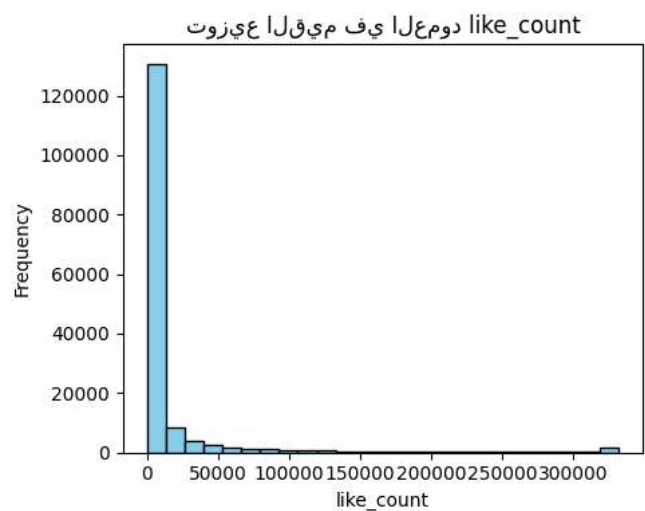
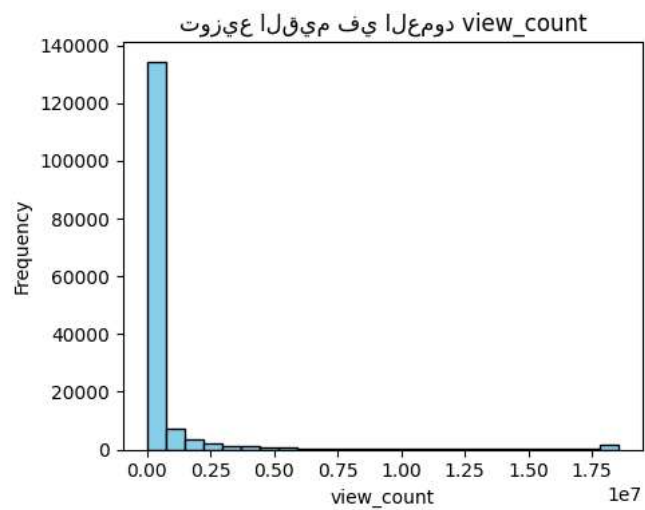
```
plt.figure(figsize=(5,4))
df_dropped[num_cols].boxplot()
plt.title('Boxplot حذف القيم الشاذة')
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(5,4))
df_capped[num_cols].boxplot()
plt.title('Boxplot بعد التقنين (Capping)')
plt.tight_layout()
plt.show()
```



```
for col in num_cols:
    plt.figure(figsize=(5,4))
    plt.hist(df_capped[col], bins=25, color='skyblue', edgecolor='black')
    plt.title(f'توزيع القيم في العمود {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```



```
corr = df_capped[num_cols].corr()

plt.figure(figsize=(5,4))
plt.imshow(corr, cmap='coolwarm', interpolation='nearest')
plt.colorbar(label='Correlation')
plt.xticks(range(len(corr.columns)), corr.columns, rotation=45)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.title('خريطة الارتباط بين المتغيرات العددية')
plt.tight_layout()
plt.show()
```

