

```
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Hepatitis C disease dataset <https://www.kaggle.com/datasets/fedesoriano/hepatitis-c-dataset>

```
# =====
# التهاب الكبد C (Hepatitis C) مشروع
# =====

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings, re
warnings.filterwarnings("ignore")
```

```
# -----
# تحميل البيانات
# -----
df = pd.read_csv("/content/drive/MyDrive/Project1/Hepatitis-C-dataset.csv")
```

```
# -----
# معاينة سريعة للبيانات
# -----
df.head()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT	
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0	
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5	
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3	
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7	
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
# -----
# أنواع الأعمدة (dtypes)
# -----
df.dtypes
```

	0
Unnamed: 0	int64
Category	object
Age	int64
Sex	object
ALB	float64
ALP	float64
ALT	float64
AST	float64
BIL	float64
CHE	float64
CHOL	float64
CREA	float64
GGT	float64
PROT	float64

dtype: object

```
# -----
# أسماء الأعمدة (columns)
# -----
df.columns

Index(['Unnamed: 0', 'Category', 'Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST',
       'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT'],
      dtype='object')
```

```
# -----
# حساب القيم المفقودة لكل عمود
# -----
df.isnull().sum()
```

	0
Unnamed: 0	0
Category	0
Age	0
Sex	0
ALB	1
ALP	18
ALT	1
AST	0
BIL	0
CHE	0
CHOL	10
CREA	0
GGT	0
PROT	1

dtype: int64

```
# -----
# حذف الأعمدة ذات المفقودات التي تتجاوز 30%
# -----
missing_percentage = df.isnull().mean() * 100
threshold = 30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=columns_to_drop)
print(f"Columns dropped (>30% missing): {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns.tolist()}")

Columns dropped (>30% missing): []
Remaining columns: ['Unnamed: 0', 'Category', 'Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']
```

```
# -----
# ترميز الأعمدة النصية (Label Encoding)
# -----
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column].astype(str))
```

```
# -----
# تعبئة القيم المفقودة
# عددية: بالمتوسط
# نصية (إن وجدت لاحقاً): بالنمط
# -----
for column in df.select_dtypes(include=['float64', 'int64']).columns:
    if df[column].isnull().any():
        df[column] = df[column].fillna(df[column].mean())

for column in df.select_dtypes(include=['object']).columns:
    if df[column].isnull().any():
        df[column] = df[column].fillna(df[column].mode()[0])
```

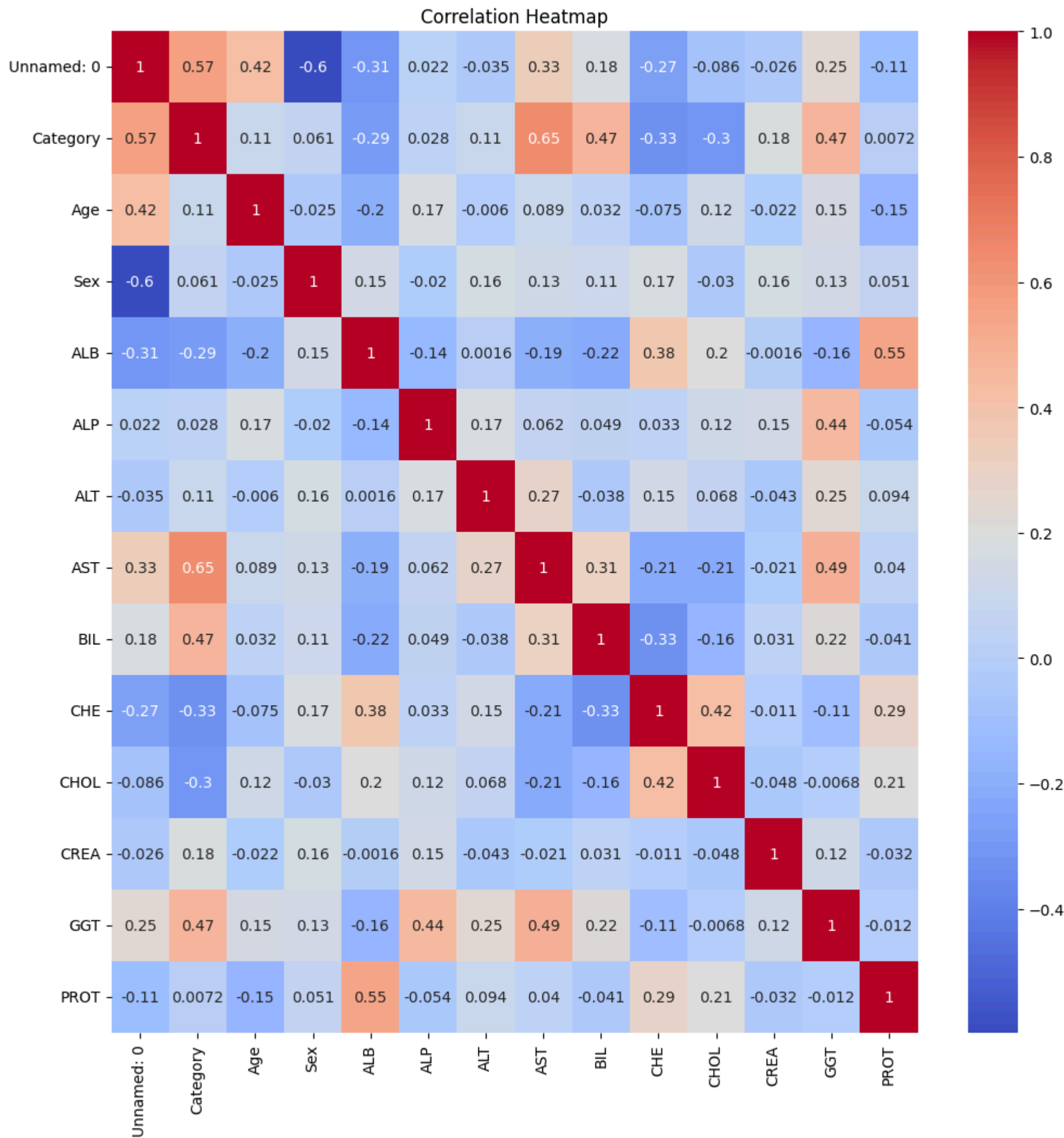
```
# التحقق النهائي للمفقودات
df.isnull().sum()
```

```
0
Unnamed: 0    0
Category      0
Age           0
Sex           0
ALB           0
ALP           0
ALT           0
AST           0
BIL           0
CHE           0
CHOL          0
CREA          0
GGT           0
PROT          0
```

dtype: int64

```
# -----
# تحديد عمود الهدف (التصنيف)
# -----
TARGET_CANDIDATES = [
    "Category", "category", "target", "label", "class", "classification",
    "status", "diagnosis", "outcome", "disease", "y"
]
target_col = next((c for c in TARGET_CANDIDATES if c in df.columns), None)
if target_col is None:
    target_col = df.columns[-1]
```

```
# -----
# خريطة الارتباط (Correlation Heatmap)
# -----
num_df = df.select_dtypes(include=['float64', 'int64'])
if num_df.shape[1] > 1:
    plt.figure(figsize=(12,12))
    sns.heatmap(num_df.corr(), annot=True, cmap='coolwarm')
    plt.title('Correlation Heatmap')
    plt.show()
else:
    print("Not enough numeric columns for heatmap.")
```



```
# =====
# بطريقة (Outliers) كشف ومعالجة القيم الشاذة
# =====
```

```
# تحديد الأعمدة العددية فقط
num_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
```

```
# لكل عمود عددي (Q3) والربع الأعلى (Q1) حساب الربع الأدنى
Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1
```

```
# تحديد حدود القيم المسموح بها
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
# جديد لتحديد الصفوف التي تحتوي على قيم شاذة DataFrame إنشاء
outliers_mask = ((df[num_cols] < lower_bound) | (df[num_cols] > upper_bound))
```

```
# Outliers عرض الأعمدة التي تحتوي على
outlier_columns = outliers_mask.any()
print("Outliers الأعمدة التي تحتوي على:")
print(outlier_columns[outlier_columns].index.tolist())
```

```
Outliers الأعمدة التي تحتوي على:
['Category', 'Age', 'ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']
```

```
# عرض عدد القيم الشاذة في كل عمود
print("\nعدد القيم الشاذة في كل عمود:")
print(outliers_mask.sum())
```

```
عدد القيم الشاذة في كل عمود:
Unnamed: 0      0
Category      82
Age            1
Sex            0
ALB           27
ALP           14
ALT           36
AST           64
BIL           47
CHE           24
CHOL          12
CREA          12
GGT           65
PROT          20
dtype: int64
```

```
# -----
# Outliers معالجة ال (NaN استبدالها بـ)
```

```
# -----
df_cleaned = df.copy()
for col in num_cols:
    Q1 = df_cleaned[col].quantile(0.25)
    Q3 = df_cleaned[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df_cleaned[col] = df_cleaned[col].where((df_cleaned[col] >= lower) & (df_cleaned[col] <= upper), np.nan)
```

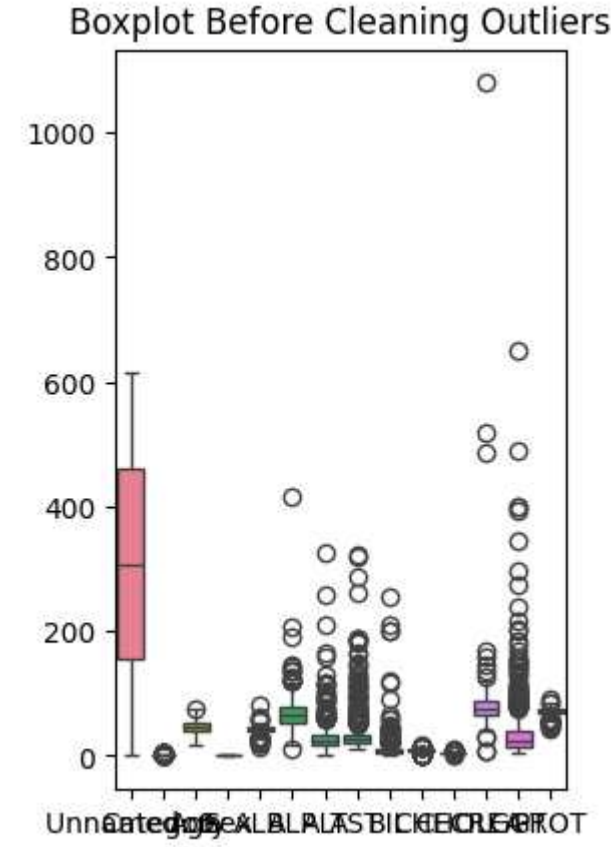
```
# -----
# تعينة القيم المفردة الناتجة عن استبدال القيم الشاذة
# -----
for col in num_cols:
    if df_cleaned[col].isnull().any():
        df_cleaned[col] = df_cleaned[col].fillna(df_cleaned[col].mean())
```

```
# -----
# مشاهدة القيم الشاذة قبل وبعد التنظيف Boxplot رسم
# -----
plt.figure(figsize=(14, 6))
```

<Figure size 1400x600 with 0 Axes>
<Figure size 1400x600 with 0 Axes>

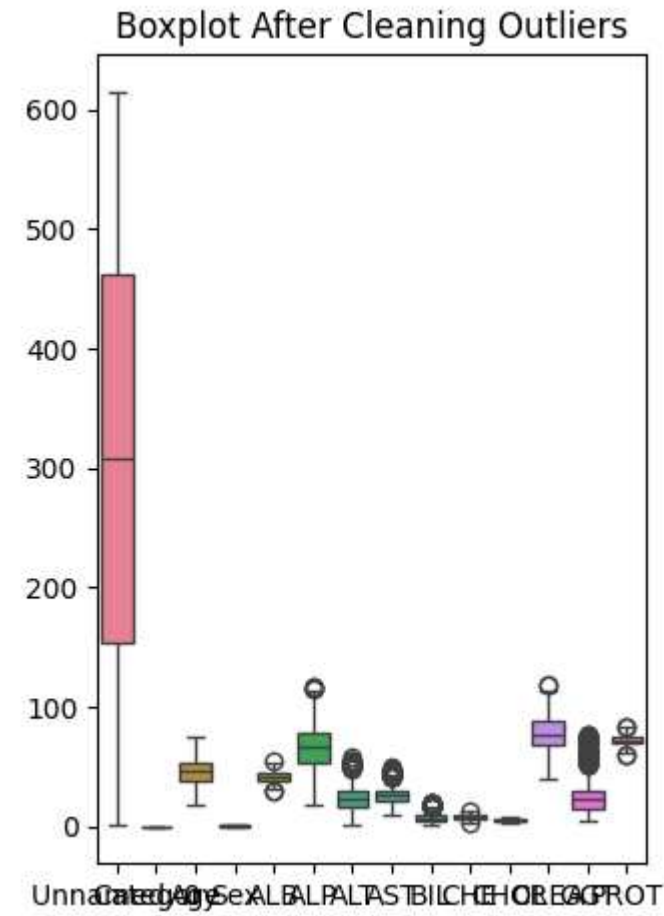
```
# قبل التنظيف
plt.subplot(1, 2, 1)
sns.boxplot(data=df[num_cols])
plt.title('Boxplot Before Cleaning Outliers')
```

Text(0.5, 1.0, 'Boxplot Before Cleaning Outliers')

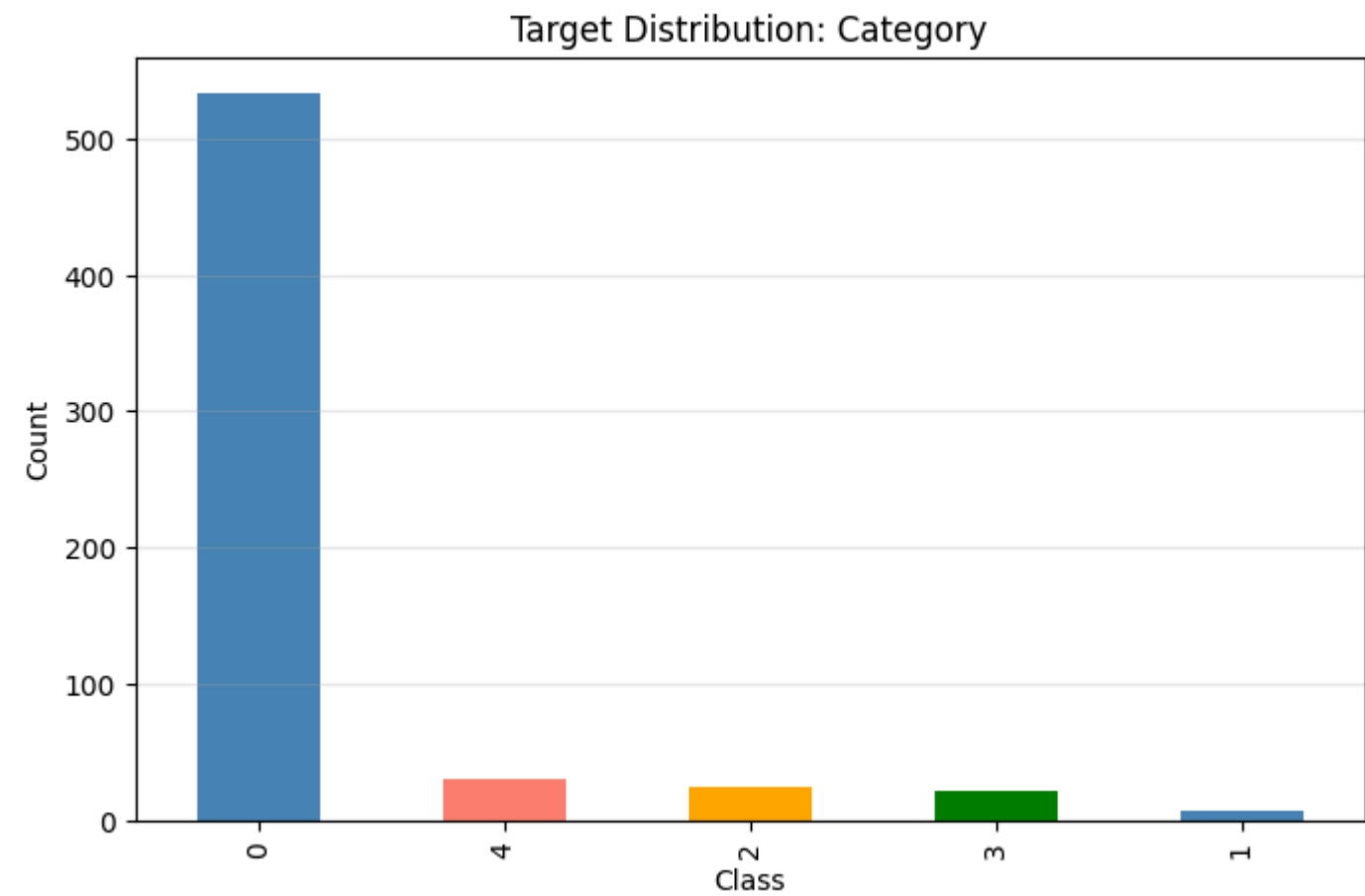


```
# بعد التنظيف
plt.subplot(1, 2, 2)
sns.boxplot(data=df_cleaned[num_cols])
plt.title('Boxplot After Cleaning Outliers')

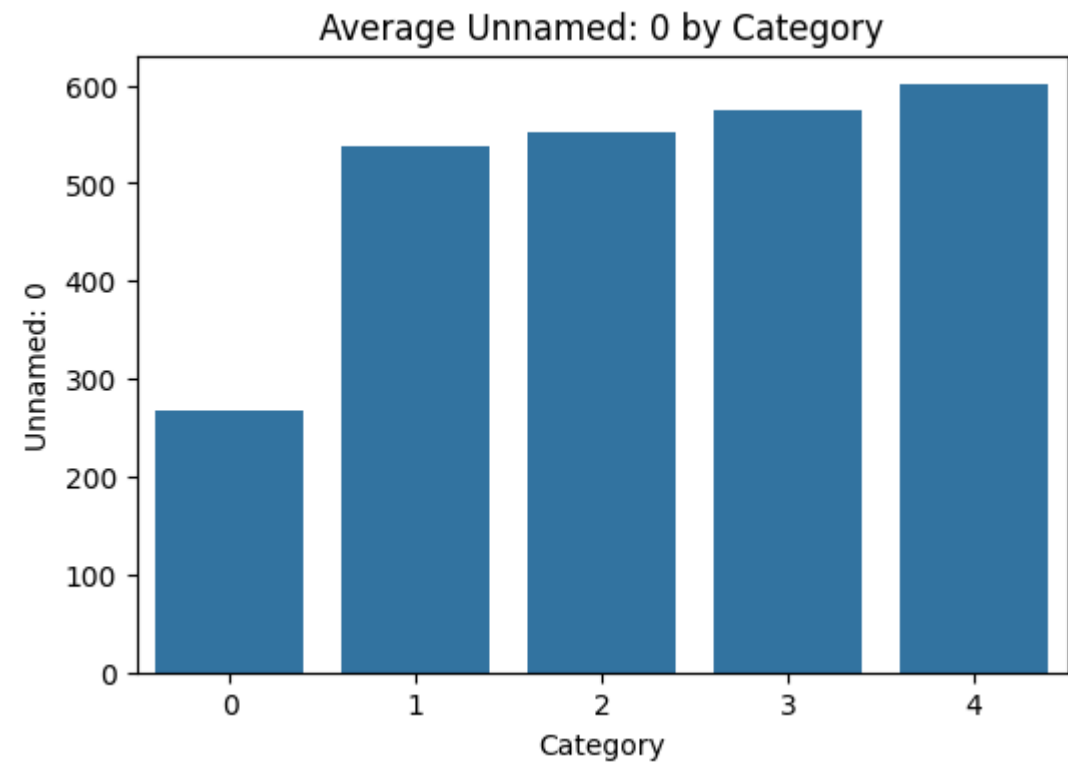
plt.tight_layout()
plt.show()
```



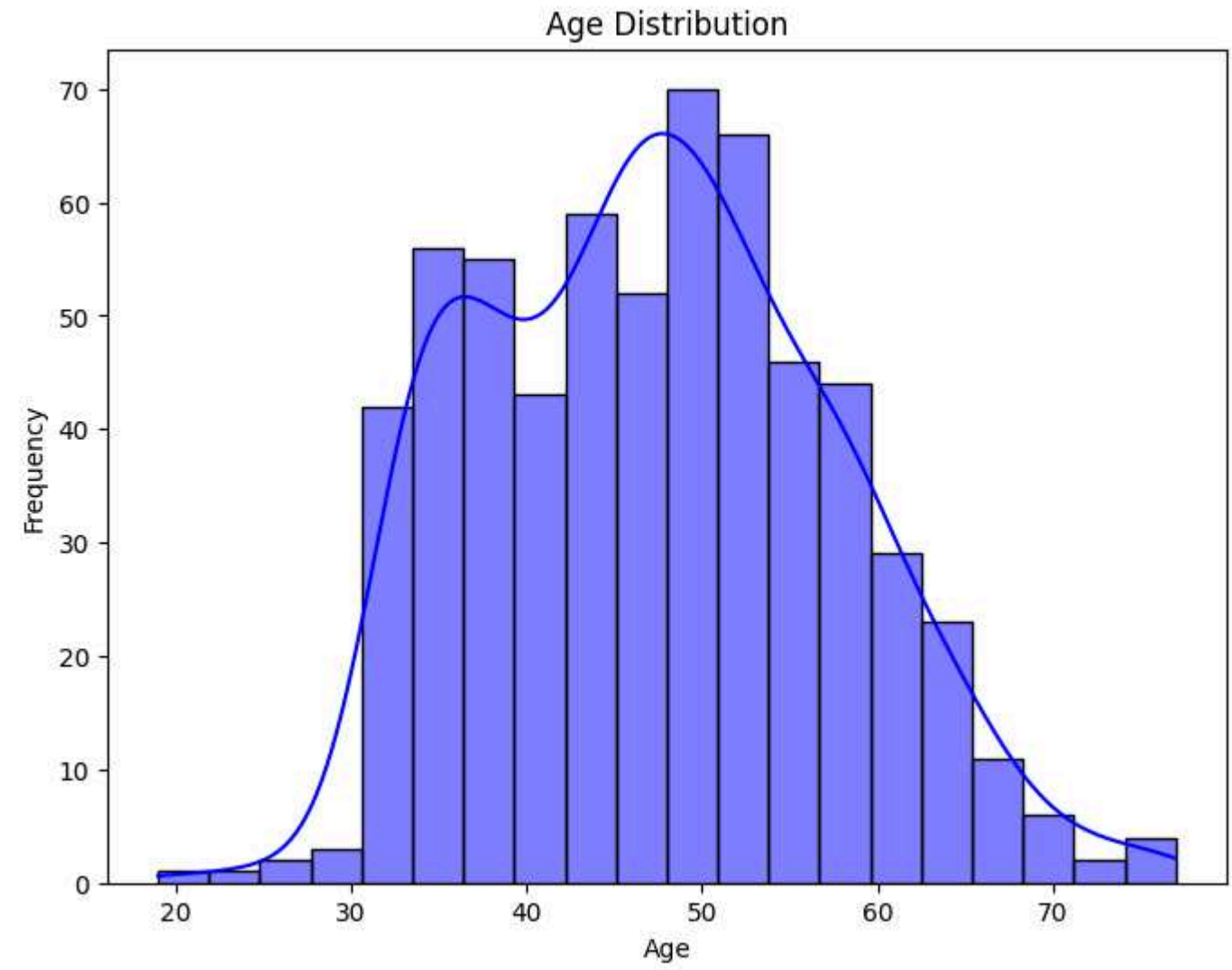
```
# -----
# توزيع الفئة المستهدفة (Target Distribution)
# -----
if target_col in df.columns:
    plt.figure(figsize=(8,5))
    df[target_col].value_counts().plot(kind='bar', color=['steelblue','salmon','orange','green'])
    plt.title(f'Target Distribution: {target_col}')
    plt.xlabel('Class'); plt.ylabel('Count')
    plt.grid(axis='y', alpha=0.3)
    plt.show()
```



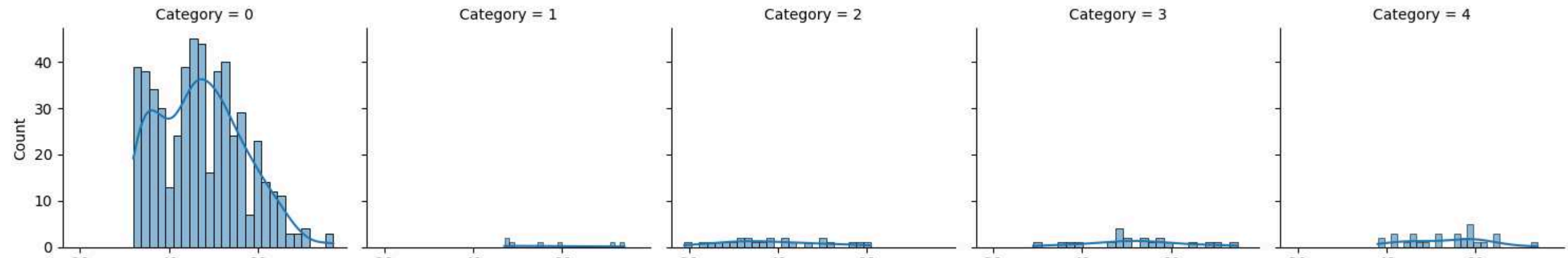
```
# -----
# متوسط أول عمود عددي حسب الفئة (Mean by Category)
# -----
num_cols = [c for c in df.columns if df[c].dtype in ['float64','int64'] and c != target_col]
if len(num_cols) > 0 and target_col in df.columns:
    first_num = num_cols[0]
    plt.figure(figsize=(6,4))
    sns.barplot(x=target_col, y=first_num, data=df, estimator='mean', ci=None)
    plt.title(f"Average {first_num} by {target_col}")
    plt.xlabel(target_col); plt.ylabel(first_num)
    plt.show()
```



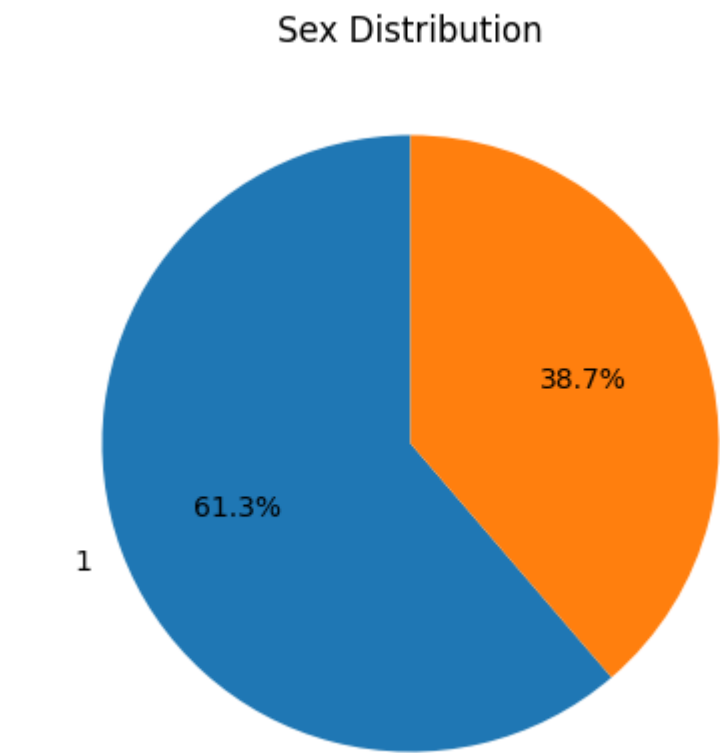
```
# -----
# توزيع عمود عددي (Numeric Distribution)
# -----
plot_num = 'Age' if 'Age' in df.columns and df['Age'].dtype != 'object' else (num_cols[0] if num_cols else None)
if plot_num is not None:
    plt.figure(figsize=(8,6))
    sns.histplot(df[plot_num], bins=20, kde=True, color='blue')
    plt.title(f'{plot_num} Distribution')
    plt.xlabel(plot_num); plt.ylabel('Frequency')
    plt.show()
```



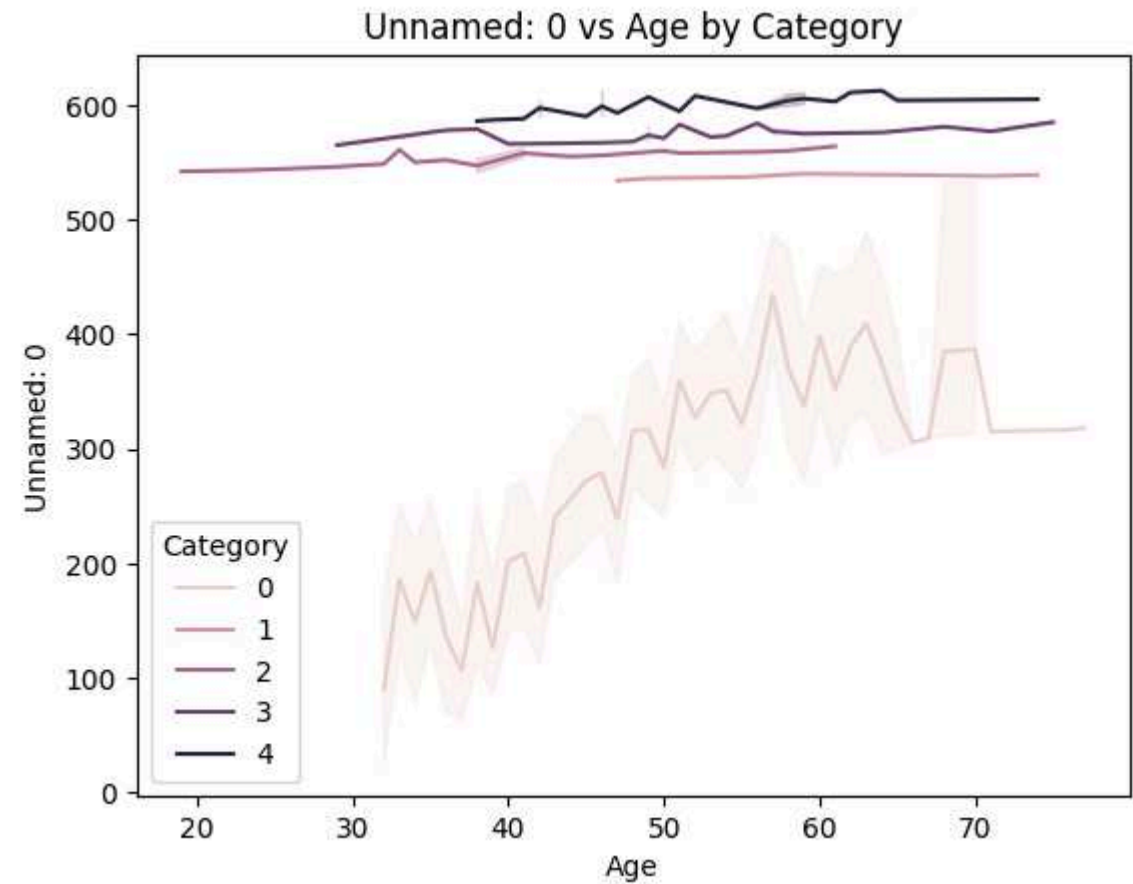
```
# -----
# توزيع عمود عددي حسب الفئة (FacetGrid)
# -----
if plot_num is not None and target_col in df.columns and df[target_col].nunique() <= 10:
    g = sns.FacetGrid(df, col=target_col)
    g.map(sns.histplot, plot_num, bins=25, kde=True)
    plt.show()
```



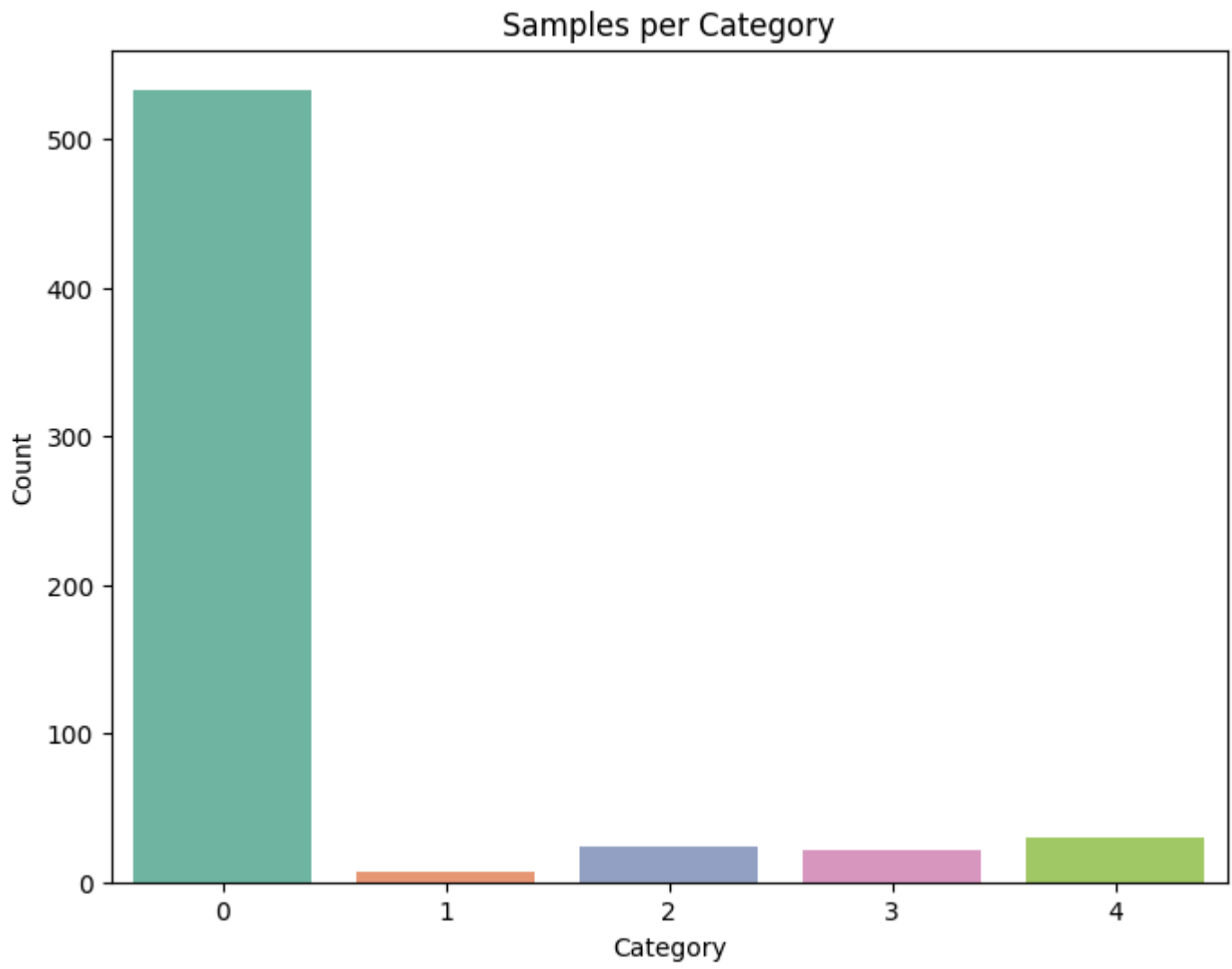
```
# -----
# رسم دائري/عذي لعمود تصنيفي قليل القيم (Pie Chart)
# -----
cat_like = [c for c in df.columns
             if c != target_col and df[c].dtype in ['int64','int32'] and df[c].nunique() <= 10]
if len(cat_like) > 0:
    feature_cat = cat_like[0]
    counts = df[feature_cat].value_counts()
    plt.figure(figsize=(5,5))
    plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90)
    plt.title(f'{feature_cat} Distribution')
    plt.show()
```



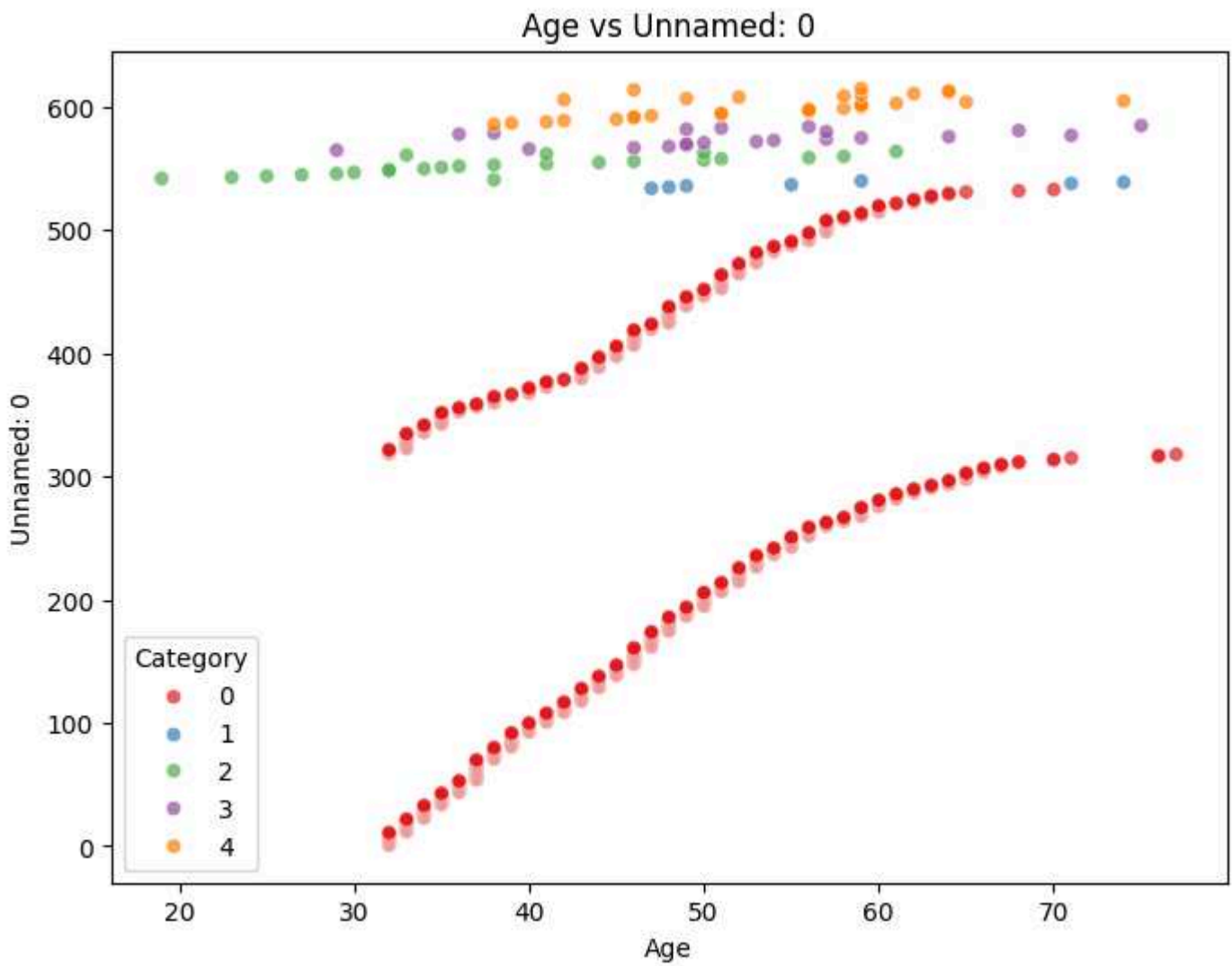
```
# -----
# مخطط خطي: عمود عددي مقابل آخر ملون بالفئة (Lineplot)
# -----
if len(num_cols) >= 2 and target_col in df.columns and df[target_col].nunique() <= 10:
    xcol = plot_num if plot_num is not None else num_cols[0]
    ycol = num_cols[1] if num_cols[0] == xcol and len(num_cols) > 1 else (num_cols[0] if xcol != num_cols[0] else (num_cols[2] if len(num_cols) > 2 else None))
    if ycol is not None and xcol != ycol:
        sns.lineplot(x=xcol, y=ycol, hue=target_col, data=df)
        plt.title(f'{ycol} vs {xcol} by {target_col}')
        plt.xlabel(xcol); plt.ylabel(ycol)
        plt.show()
```



```
# -----
# Countplot لعمود الهدف (Countplot)
# -----
if target_col in df.columns:
    plt.figure(figsize=(8,6))
    sns.countplot(x=target_col, data=df, palette='Set2')
    plt.title('Samples per Category')
    plt.xlabel(target_col); plt.ylabel('Count')
    plt.show()
```



```
# -----  
# Scatterplot: عمود عددي مقابل آخر ملون بال فئة (Scatterplot)  
# -----  
if plot_num is not None and len(num_cols) >= 2:  
    yscatter = None  
    for c in num_cols:  
        if c != plot_num:  
            yscatter = c  
            break  
if yscatter is not None:  
    plt.figure(figsize=(8,6))  
    if target_col in df.columns and df[target_col].nunique() <= 10:  
        sns.scatterplot(x=plot_num, y=yscatter, data=df, hue=target_col, palette='Set1', alpha=0.7)  
    else:  
        sns.scatterplot(x=plot_num, y=yscatter, data=df, color='steelblue', alpha=0.7)  
    plt.title(f'{plot_num} vs {yscatter}')  
    plt.xlabel(plot_num); plt.ylabel(yscatter)  
    plt.show()
```



```
# -----  
# Boxplot: توزيع أول عمود عددي حسب الفئة (Boxplot)  
# -----  
if target_col in df.columns and len(num_cols) > 0:  
    plt.figure(figsize=(8,6))  
    sns.boxplot(x=target_col, y=first_num, data=df, palette='Set1')  
    plt.title(f'{first_num} by {target_col}')  
    plt.xlabel(target_col); plt.ylabel(first_num)  
    plt.show()
```

