

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

```
# Display kidney_disease csv
df=pd.read_csv("/content/sample_data/kidney_disease.csv")
df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no

5 rows × 25 columns

Getting Features type

```
df.dtypes
```

	0
age	float64
bp	float64
sg	float64
al	float64
su	float64
rbc	object
pc	object
pcc	object
ba	object
bgr	float64
bu	float64
sc	float64
sod	float64
pot	float64
hemo	float64
pcv	object
wc	object
rc	object
htn	object
dm	object
cad	object
appet	object
pe	object
ane	object
classification	object

```
dtype: object
```

Print columns

```
df.columns
```

```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
      'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
      'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

Getting the count of missing values

```
df.isnull().sum()
```

	0
age	9
bp	12
sg	46
al	45
su	48
rbc	151
pc	64
pcc	4
ba	4
bgr	44
bu	19
sc	17
sod	87
pot	88
hemo	51
pcv	69
wc	105
rc	129
htn	2
dm	2
cad	2
appet	1
pe	1
ane	1
classification	0

```
dtype: int64
```

Deleting columns that have 30% missing values

```
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage >
threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)

# Display the resulting dataframe after removing columns with too many missing values

print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: ['rbc', 'rc']
Remaining columns: Index(['age', 'bp', 'sg', 'al', 'su', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc',
      'sod', 'pot', 'hemo', 'pcv', 'wc', 'htn', 'dm', 'cad', 'appet', 'pe',
      'ane', 'classification'],
      dtype='object')
```

Pre-processing on dataset

```

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])
df.head()

```

	age	bp	sg	al	su	pc	pcc	ba	bgr	bu	...	hemo	pcv	wc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	1	0	0	121.0	36.0	...	15.4	32	72	1	4	1	0	0	0	0
1	7.0	50.0	1.020	4.0	0.0	1	0	0	NaN	18.0	...	11.3	26	56	0	3	1	0	0	0	0
2	62.0	80.0	1.010	2.0	3.0	1	0	0	423.0	53.0	...	9.6	19	70	0	4	1	1	0	1	0
3	48.0	70.0	1.005	4.0	0.0	0	1	0	117.0	56.0	...	11.2	20	62	1	3	1	1	1	1	0
4	51.0	80.0	1.010	2.0	0.0	1	0	0	106.0	26.0	...	11.6	23	68	0	3	1	0	0	0	0

5 rows × 23 columns

Filling missing values based on feature type

```

# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean()
# Get the mode (most frequent value)
df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0]
# Get the mode (most frequent value)
df[column] = df[column].fillna(mode_value)
df.isnull().sum()

```

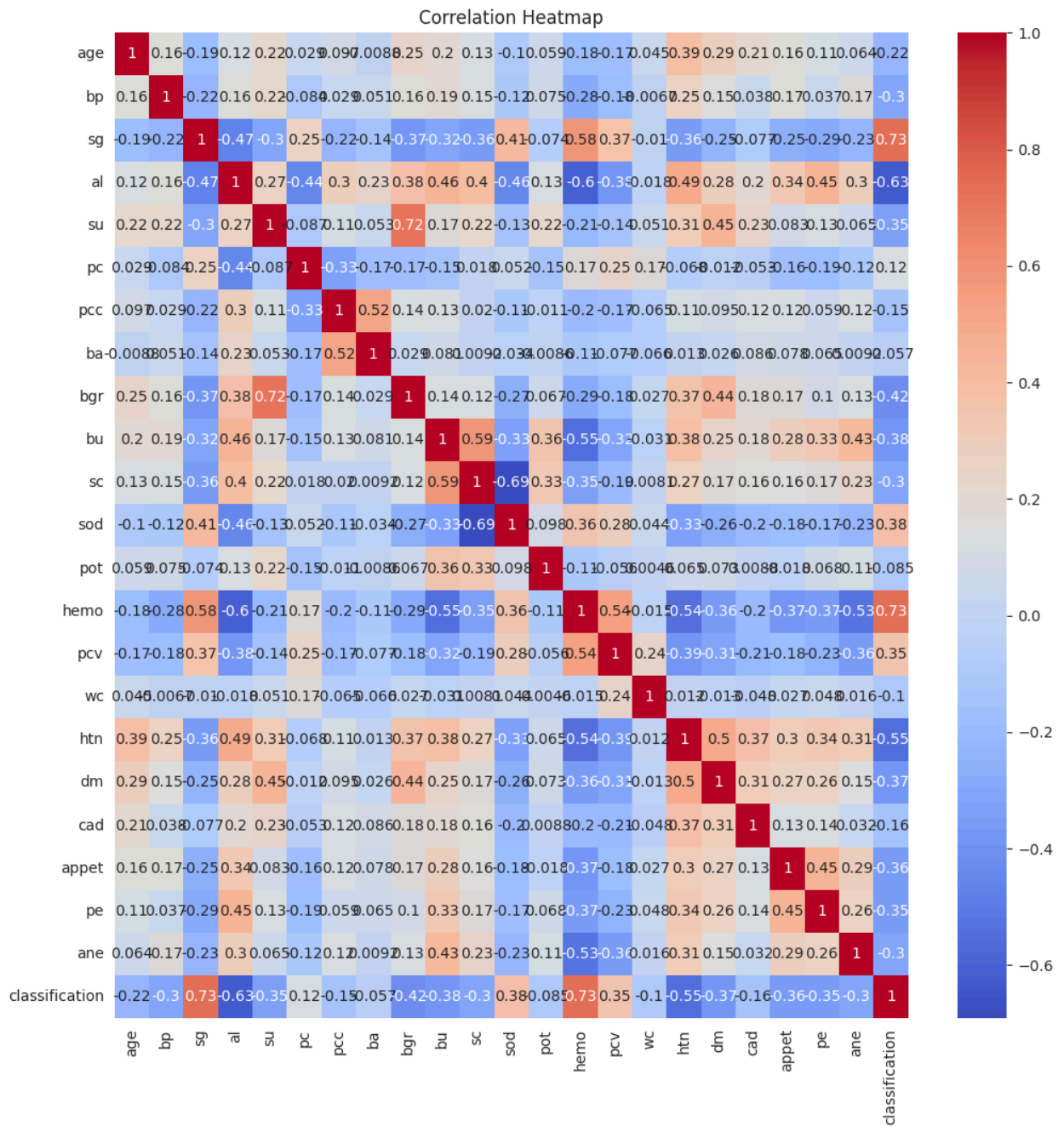
	0
age	9
bp	12
sg	46
al	45
su	48
pc	0
pcc	0
ba	0
bgr	44
bu	19
sc	17
sod	87
pot	88
hemo	0
pcv	0
wc	0
htn	0
dm	0
cad	0
appet	0
pe	0
ane	0
classification	0

dtype: int64

▼ Data Visulization

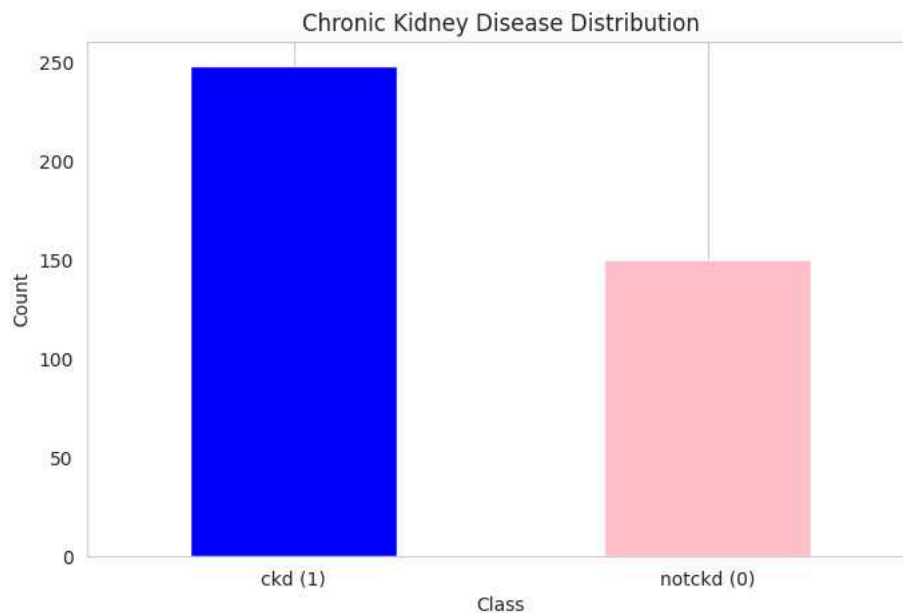
Correlation heatmap

```
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
df['classification'].value_counts()
```

```
# Create a bar chart
plt.figure(figsize=(8, 5))
df['classification'].value_counts().plot(kind='bar', color=['blue', 'pink'])
plt.xticks([0, 1], ['ckd (1)', 'notckd (0)'], rotation=0)
plt.title('Chronic Kidney Disease Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```

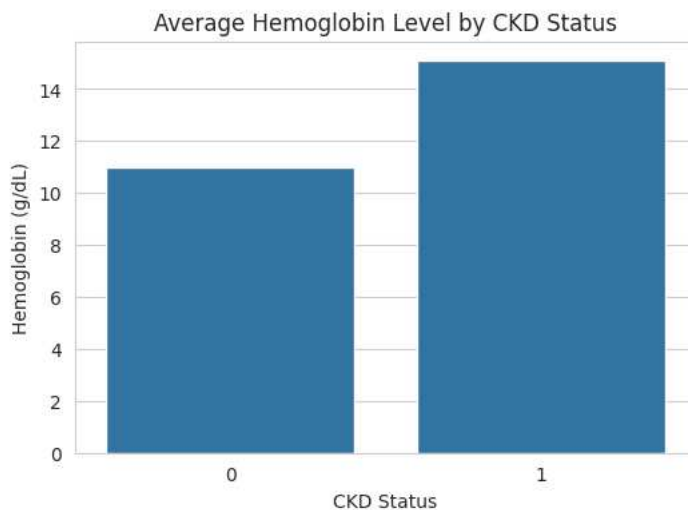


```
# Example 2: Average Hemoglobin by CKD Status
import seaborn as sns
plt.figure(figsize=(6,4))
sns.barplot(x='classification', y='hemo', data=df, estimator='mean',
ci=None)
plt.title("Average Hemoglobin Level by CKD Status")
plt.xlabel("CKD Status")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```

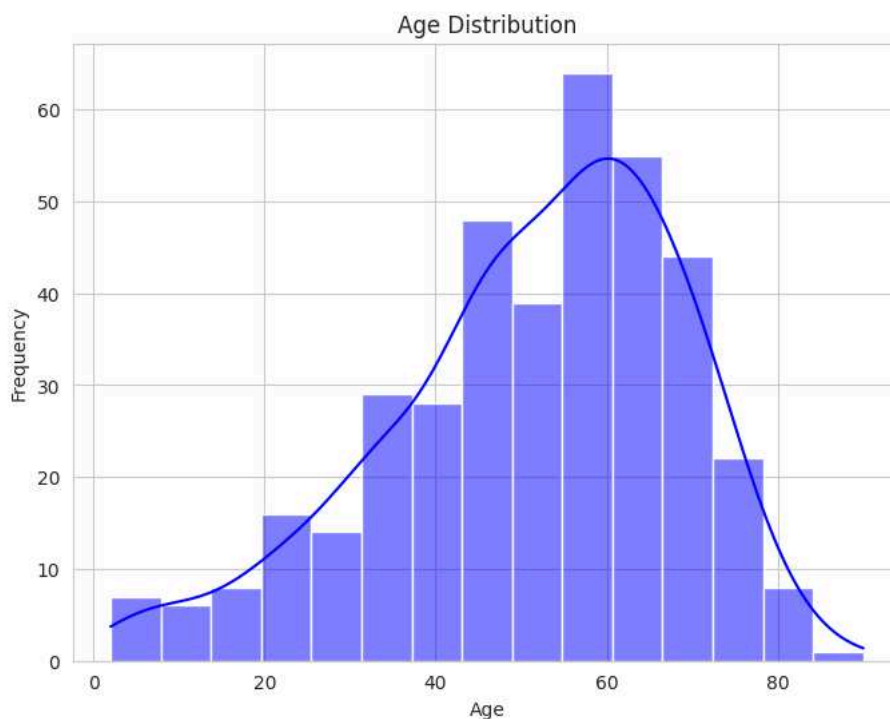
/tmp/ipython-input-1309407608.py:4: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

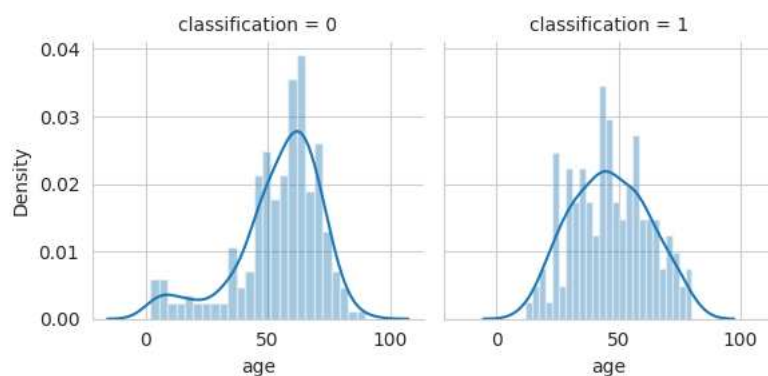
```
sns.barplot(x='classification', y='hemo', data=df, estimator='mean',
```



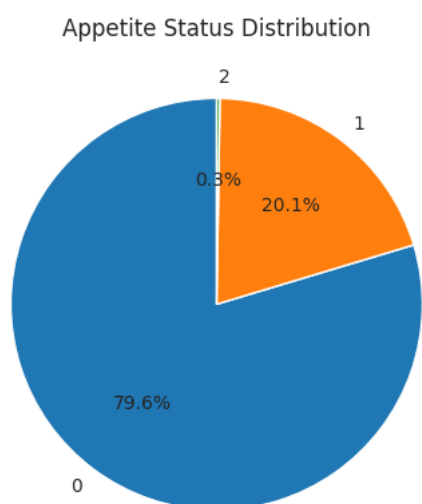
```
cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=15, kde=True, color='blue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



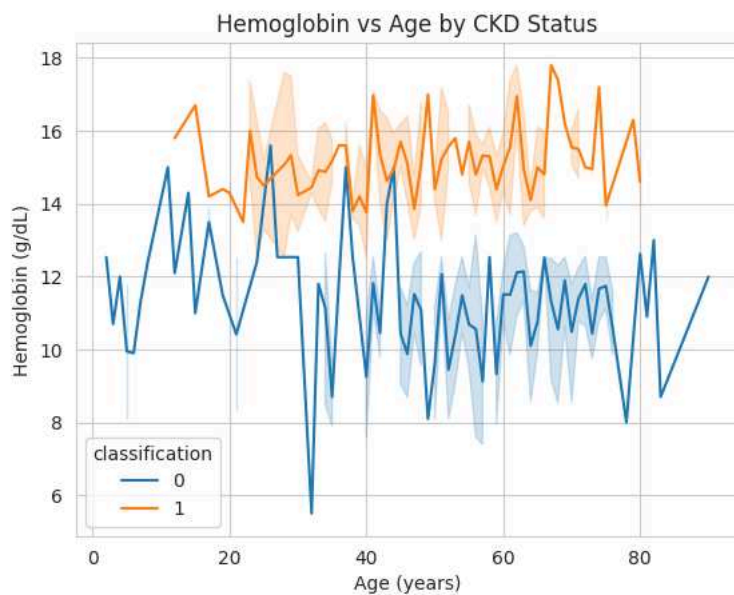
```
g = sns.FacetGrid(df,col="classification")
g.map(sns.distplot,"age", bins=25)
plt.show()
```



```
appet_counts = df['appet'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title("Appetite Status Distribution")
plt.show()
```

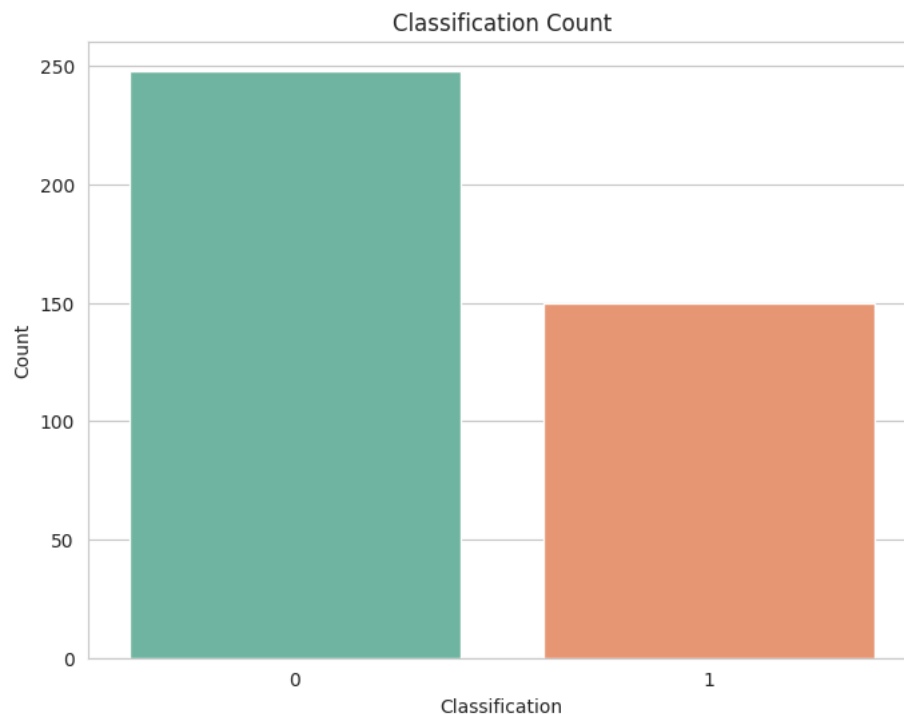


```
sns.lineplot(x='age', y='hemo', hue='classification', data=df)
plt.title("Hemoglobin vs Age by CKD Status")
plt.xlabel("Age (years)")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```

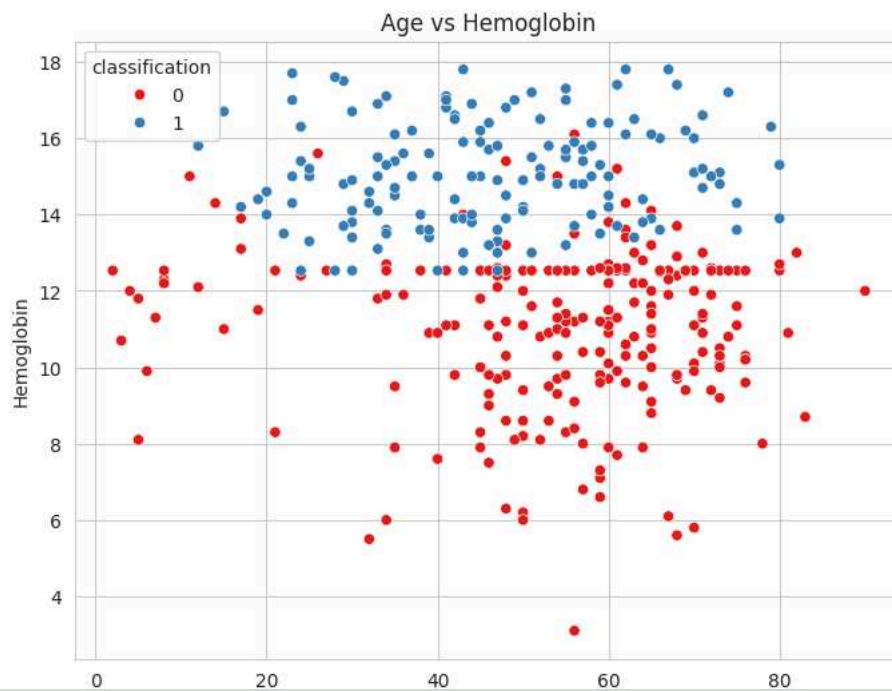


Observations

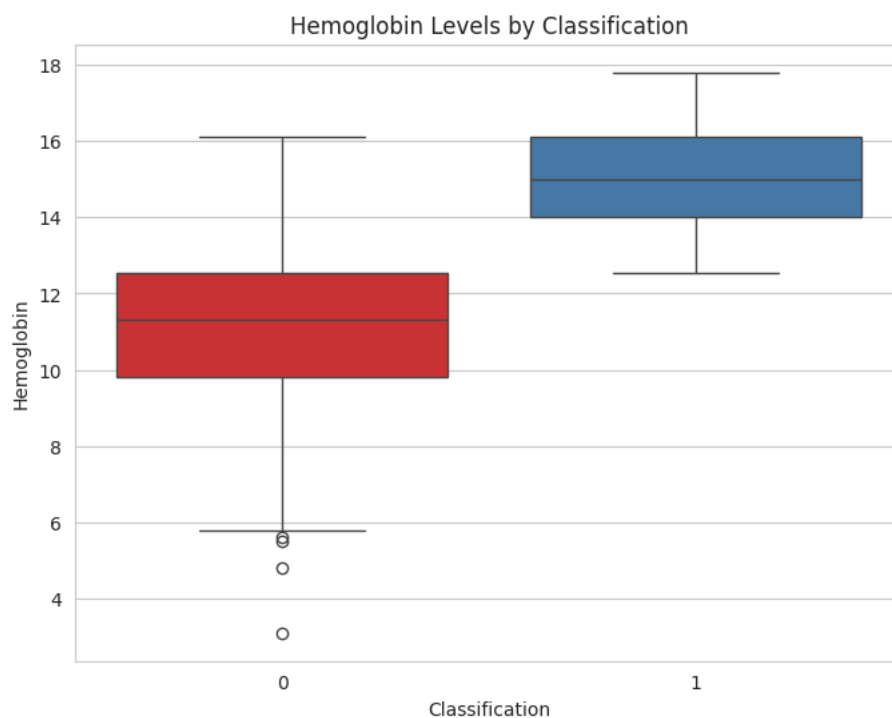
```
# 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='classification', data=df, palette='Set2')
plt.title('Classification Count')
plt.xlabel('Classification')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='hemo', data=df, hue='classification',
               palette='Set1')
plt.title('Age vs Hemoglobin')
plt.xlabel('Age')
plt.ylabel('Hemoglobin')
plt.show()
```



```
# Boxplot for Hemoglobin by Classification
plt.figure(figsize=(8, 6))
sns.boxplot(x='classification', y='hemo', data=df, palette='Set1')
plt.title('Hemoglobin Levels by Classification')
plt.xlabel('Classification')
plt.ylabel('Hemoglobin')
plt.show()
```



✓ Heart disease project

```
df = pd.read_csv('/content/sample_data/heart.csv')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

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

```
numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
                    'ca']
df.describe()
```

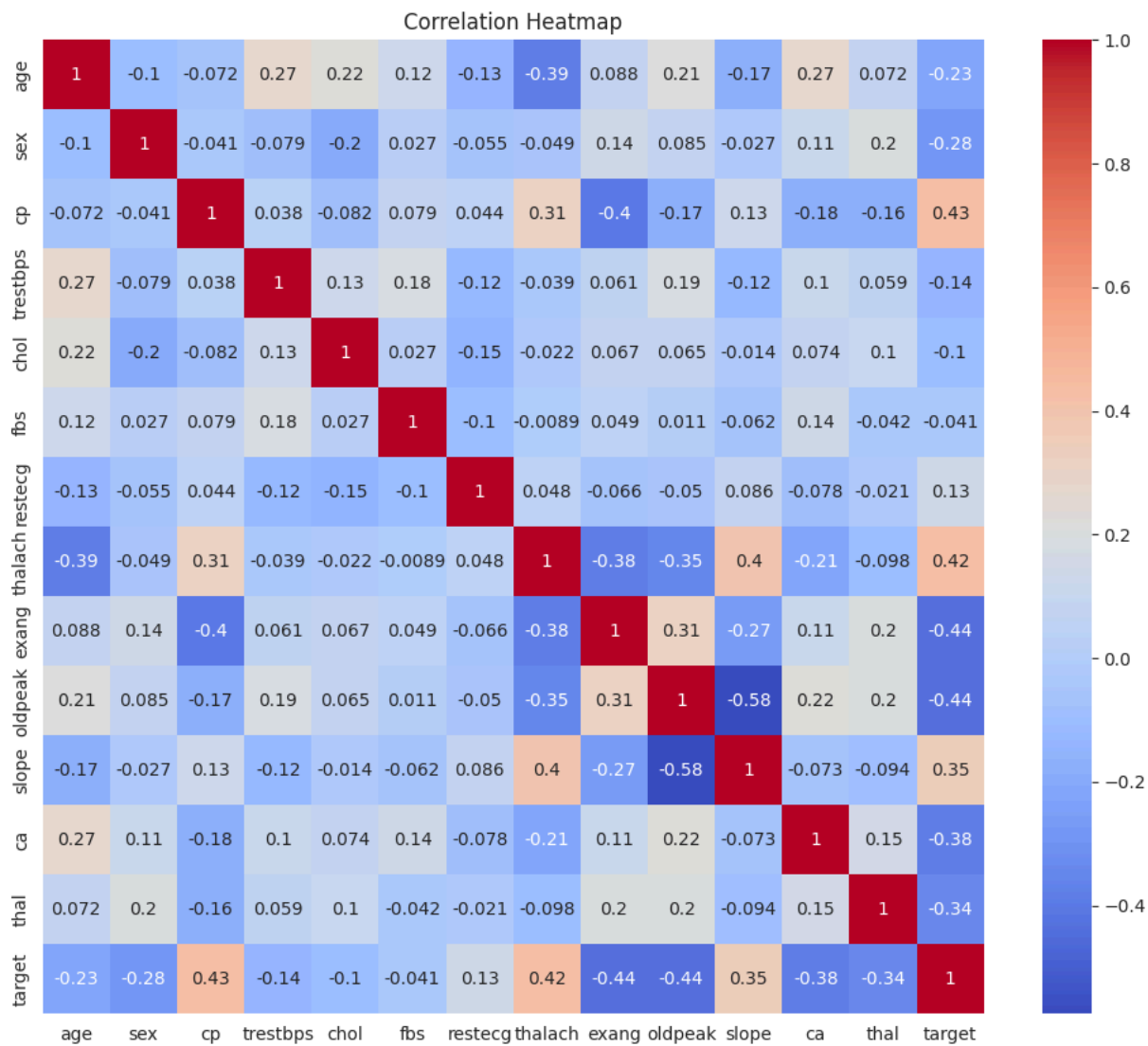
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	o
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.000000	0.149268	0.529756	149.114146	0.336585	1.000000
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.000000
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	132.000000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	152.000000	0.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000	1.000000	166.000000	1.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.000000

```
df.shape
(1025, 14)
df.isnull().sum()
```

	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

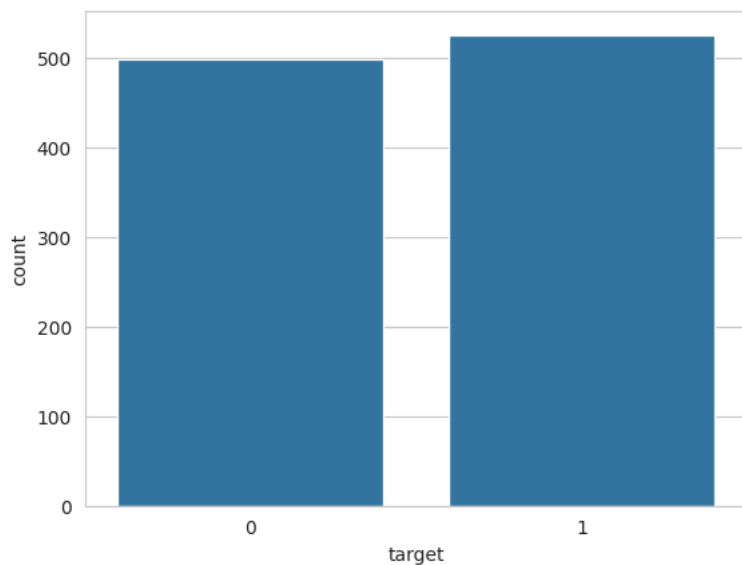
dtype: int64

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
sns.countplot(x='target', data=df)
```

<Axes: xlabel='target', ylabel='count'>



```
plt.figure(figsize=(20,35))
plotnumber =1
for column in df.columns[1:]:
    ax = plt.subplot(10,3,plotnumber)
    sns.boxplot(data = df, x = column, palette='pastel')
    plt.xlabel(column)
    plotnumber+=1
plt.show()
```

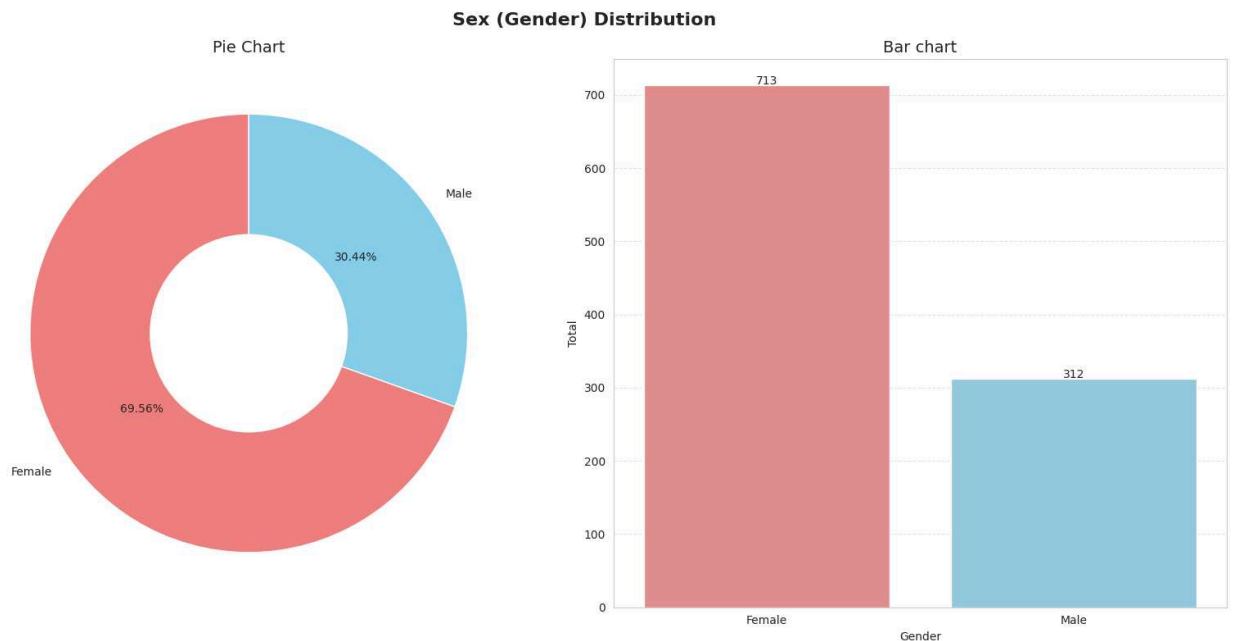


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Example colors and data (you can replace with your actual values)
colors = ['lightcoral', 'skyblue']
labels = ['Female', 'Male']
```

```

# Set order based on actual data
order = df['sex'].value_counts().index
# --- Set up the figure ---
plt.figure(figsize=(16, 8))
plt.suptitle('Sex (Gender) Distribution', fontsize=16,
fontweight='bold')
# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['sex'].value_counts(), labels=labels, colors=colors,
autopct='%0.2f%%', startangle=90)
# Add white circle in the middle for donut effect
centre = plt.Circle((0, 0), 0.45, color='white')
plt.gca().add_artist(centre)
# --- Histogram (Count Plot) ---
plt.subplot(1, 2, 2)
plt.title('Bar chart', fontsize=14)
ax = sns.countplot(x='sex', data=df, palette=colors, order=order)
# Add counts on top of bars
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 1, int(height),
    ha='center', fontsize=10)
plt.xlabel('Gender')
plt.ylabel('Total')
plt.xticks([0, 1], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```



```

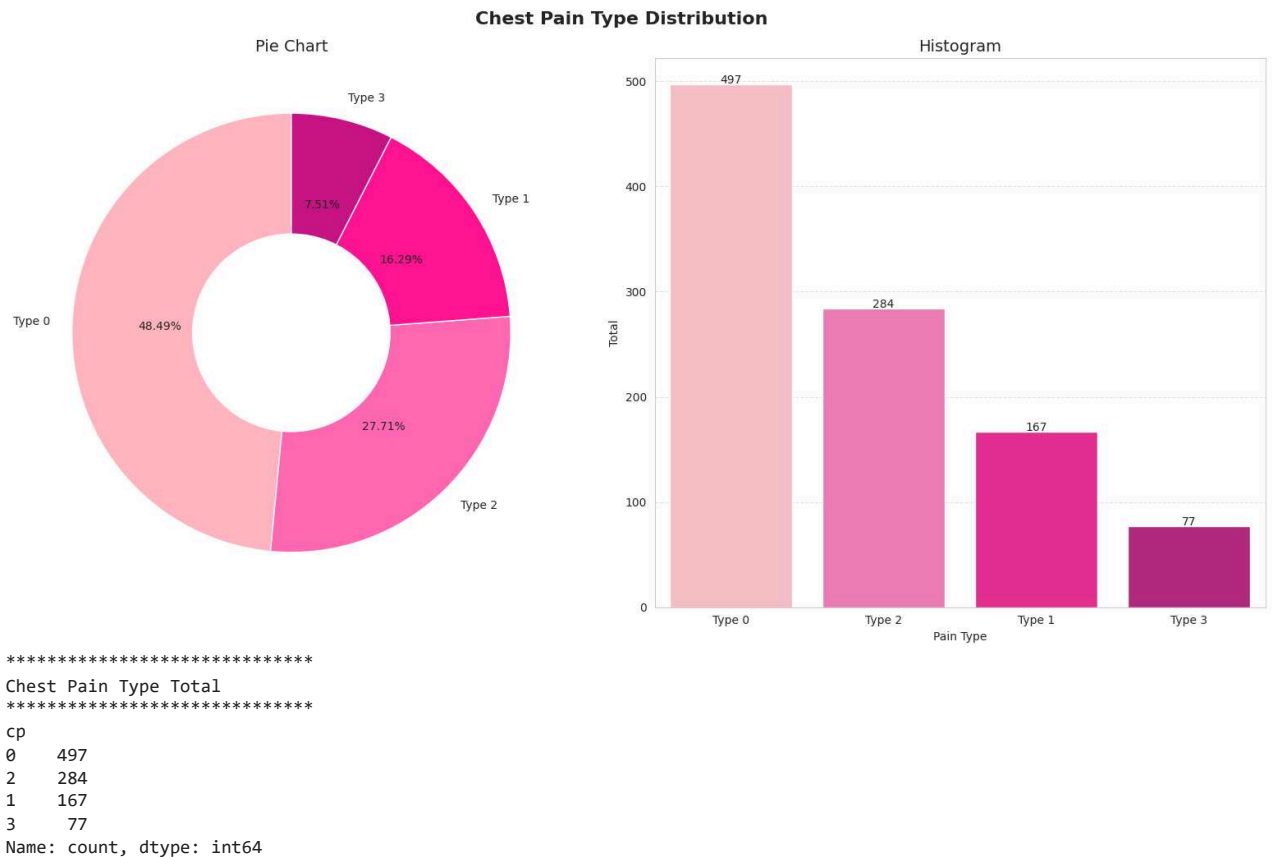
# Example setup
colors = ['lightpink', 'hotpink', 'deeppink', 'mediumvioletred']
labels = ['Type 0', 'Type 2', 'Type 1', 'Type 3']
order = df['cp'].value_counts().index
# Ensure bars follow actual data order
# --- Create Figure ---
plt.figure(figsize=(16, 8))
plt.suptitle('Chest Pain Type Distribution', fontsize=16,
fontweight='bold')
# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['cp'].value_counts(), labels=labels, colors=colors,
autopct='%0.2f%%', startangle=90)

```

```

autopct = '%.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white') # Donut-style center
plt.gca().add_artist(centre)
# --- Histogram (Count Plot) ---
plt.subplot(1, 2, 2)
plt.title('Histogram', fontsize=14)
ax = sns.countplot(x='cp', data=df, palette=colors, order=order)
# Add count labels on top of bars
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height + 1,
            int(height),
            ha='center', fontsize=10)
plt.xlabel('Pain Type')
plt.ylabel('Total')
plt.xticks([0, 1, 2, 3], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
# --- Display value counts ---
print('*' * 30)
print('Chest Pain Type Total')
print('*' * 30)
print(df['cp'].value_counts(dropna=False))

```



```

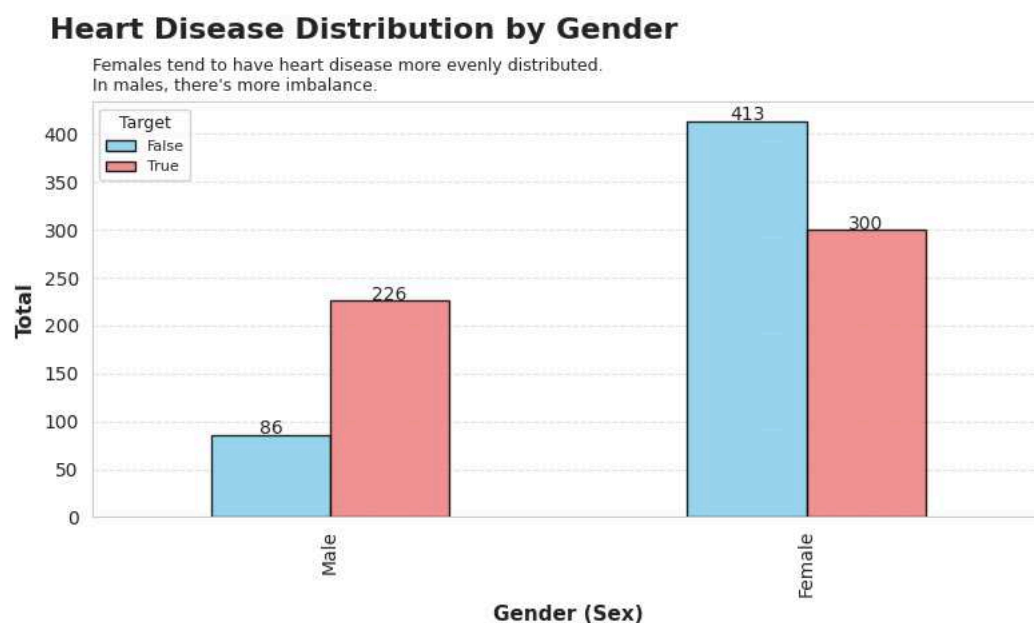
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# --- Label Settings ---
labels = ['False', 'True'] # For target: 0 = no disease, 1 = disease
label_gender = [0, 1]
label_gender2 = ['Male', 'Female']
# --- Create Bar Chart ---
ax = pd.crosstab(df.sex, df.target).plot(
    kind='bar',

```

```

figsize=(8, 5),
color=['skyblue', 'lightcoral'],
edgecolor='black',
alpha=0.85
)
# --- Annotate Bars ---
for rect in ax.patches:
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height + 1,
            int(height), ha='center', fontsize=10)
# --- Titles and Labels ---
plt.suptitle('Heart Disease Distribution by Gender', fontsize=16,
fontweight='bold', x=0.05, ha='left')
plt.title("Females tend to have heart disease more evenly distributed.\nIn males, there's more imbalance.", fontsize=9, loc=
plt.xlabel('Gender (Sex)', fontsize=11, fontweight='bold')
plt.ylabel('Total', fontsize=11, fontweight='bold')
plt.xticks(label_gender, label_gender2)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.grid(axis='x', alpha=0)
# --- Legend ---
plt.legend(labels=labels, title='Target', fontsize=8,
title_fontsize=9, loc='upper left', frameon=True)
plt.tight_layout(rect=[0, 0.04, 1, 1.02])
plt.show()

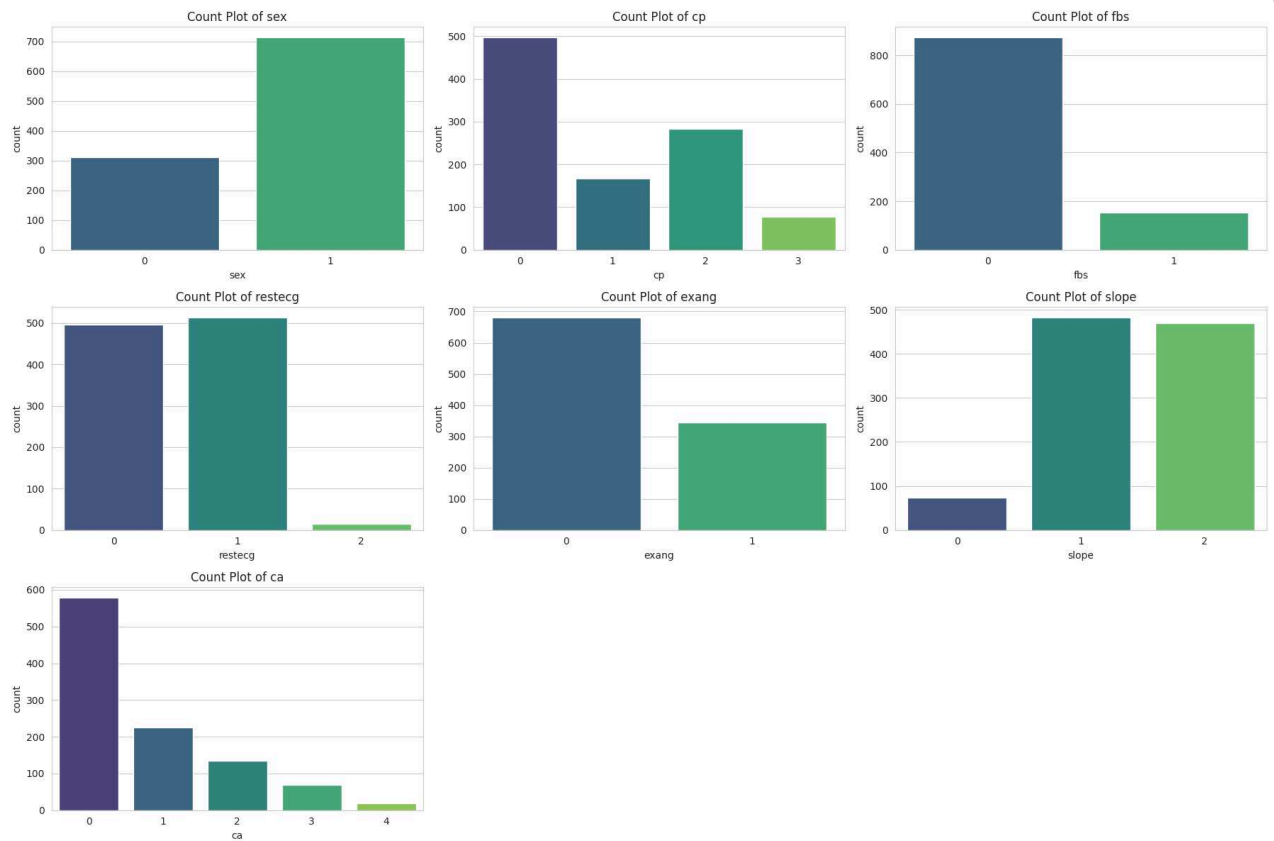
```



```

plt.figure(figsize=(18, 12))
for i, col in enumerate(categorical_cols):
    plt.subplot(3, 3, i + 1)
    sns.countplot(x=col, data=df, palette='viridis')
    plt.title(f'Count Plot of {col}')
plt.tight_layout()
plt.show()

```



```
# Number of categorical columns
n = len(categorical_cols)
nrows, ncols = (3, 3) # adjust as needed

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 12))
axes = axes.flatten()

for i, col in enumerate(categorical_cols):
    ct = pd.crosstab(df[col], df['target'], normalize='index')

    # Create a combined pie chart for all categories in the feature
    # Each pie slice = proportion of samples in each category having target=0 or 1
    for idx, category in enumerate(ct.index):
        ax = axes[i]
        sizes = ct.loc[category]
        labels = [f'{label} ({sizes[label] * 100:.1f}%)' for label in sizes.index]

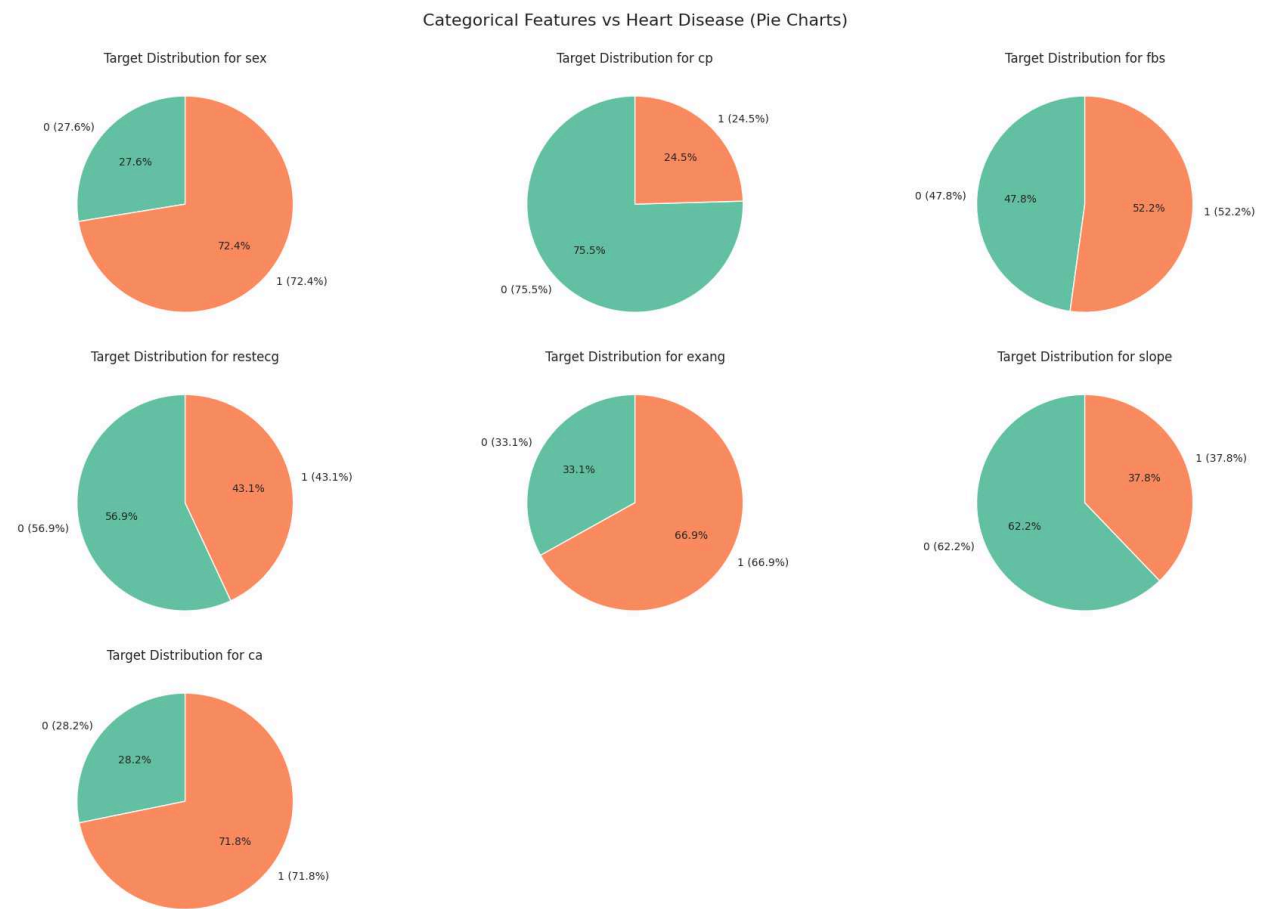
        ax.pie(
            sizes,
            labels=labels,
            autopct='%1.1f%%',
            startangle=90,
            colors=plt.cm.Set2.colors
        )
        ax.set_title(f'{col}: {category}')
        break # Comment this line if you want separate subplots per category

    axes[i].set_title(f'Target Distribution for {col}')

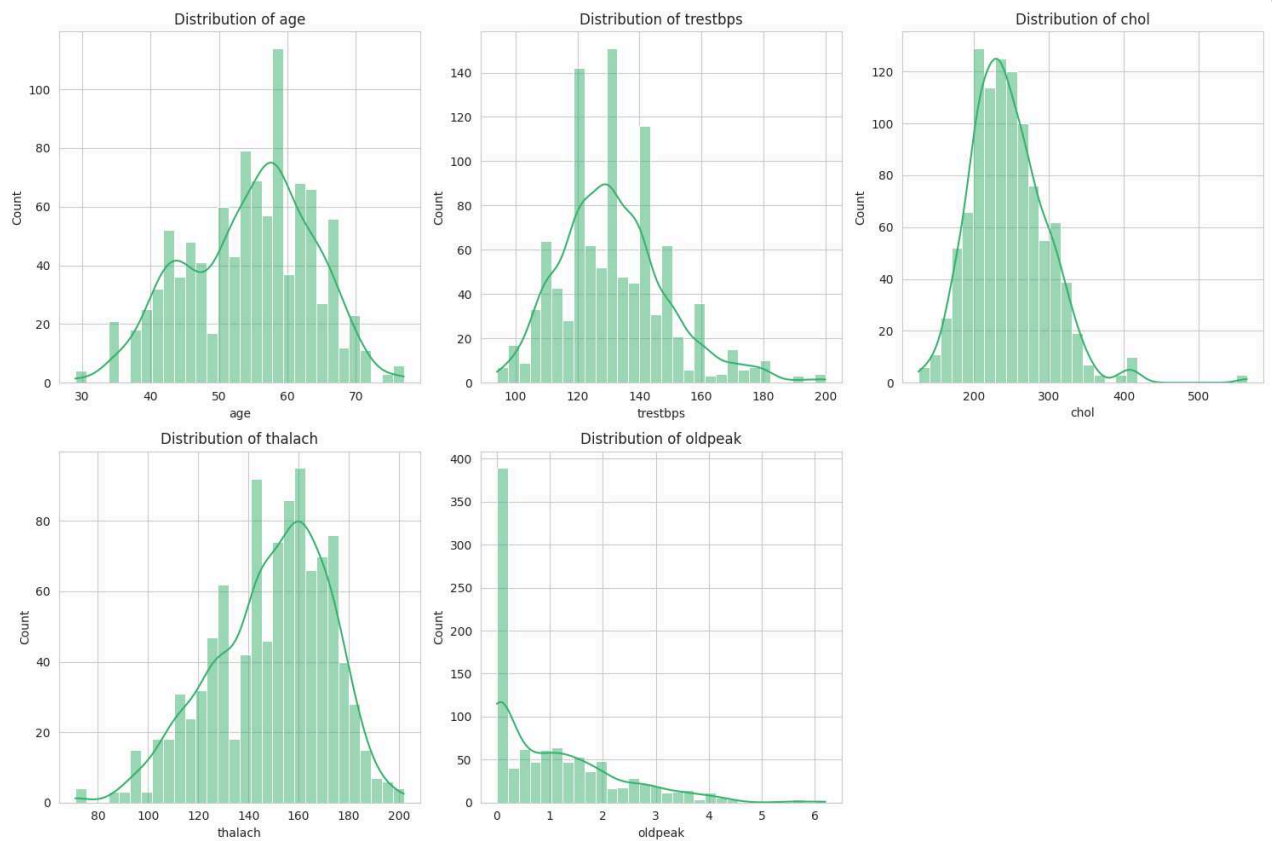
# Remove empty axes if there are fewer categorical columns
for j in range(len(categorical_cols), len(axes)):
```

```
fig.delaxes(axes[j])

plt.tight_layout()
plt.suptitle('Categorical Features vs Heart Disease (Pie Charts)', fontsize=16, y=1.03)
plt.show()
```

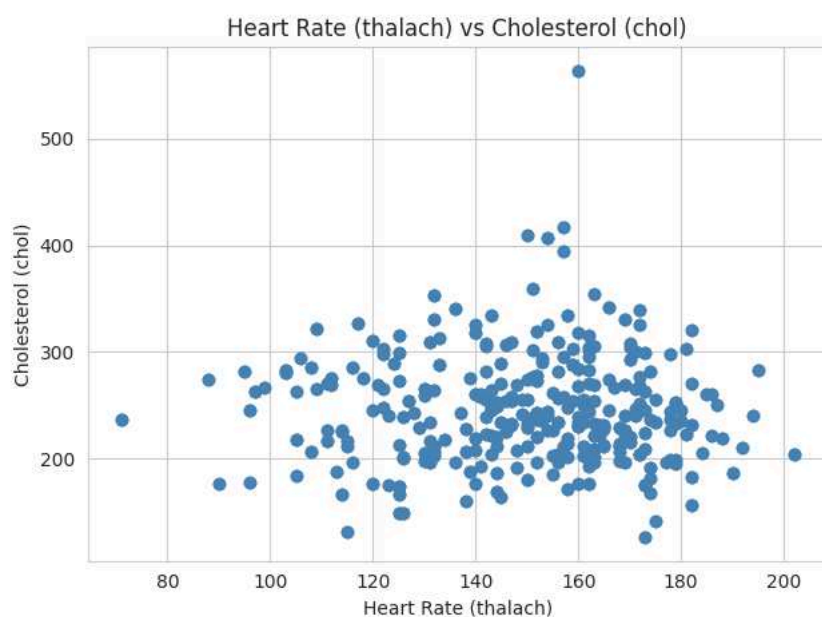


```
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(2, 3, i + 1)
    sns.histplot(df[col], kde=True, bins=30, color='mediumseagreen')
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```



Heart Rate (thalach) vs Cholesterol (chol)

```
plt.figure()
plt.scatter(df["thalach"], df["chol"], color='steelblue', alpha=0.7)
plt.title("Heart Rate (thalach) vs Cholesterol (chol)")
plt.xlabel("Heart Rate (thalach)")
plt.ylabel("Cholesterol (chol)")
plt.tight_layout()
plt.show()
```

**Interpretation:**

People with higher or lower cholesterol can have a wide range of heart rates. A few outliers exist — for example, some patients with

Interpretation:

People with higher or lower cholesterol can have a wide range of heart rates. A few outliers exist — for example, some patients with very high cholesterol (>500 mg/dL) but moderate heart rates (~140 bpm). Overall, the data suggests weak or no correlation between cholesterol and heart rate in this sample.

```
# 1b. Boxplot of age grouped by target
plt.figure()
data_for_box = [neg.values, pos.values]
plt.boxplot(data_for_box, labels=["No Disease (0)", "Disease (1)"])
plt.title("Age by Outcome (Boxplot)")
plt.ylabel("Age")
plt.tight_layout()
plt.show()
```

