

```
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)

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

```
df = pd.read_csv('/content/drive/MyDrive/cod1/project1/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](#)

```
df.dtypes
```

	0
age	int64
sex	int64
cp	int64
trestbps	int64
chol	int64
fbs	int64
restecg	int64
thalach	int64
exang	int64
oldpeak	float64
slope	int64
ca	int64
thal	int64
target	int64

dtype: object

df.columns

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',  
      'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],  
      dtype='object')
```

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

```
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing
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
```

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

```
Columns dropped: []
Remaining columns: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
                        'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
                        dtype='object')
```

```
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	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](#)

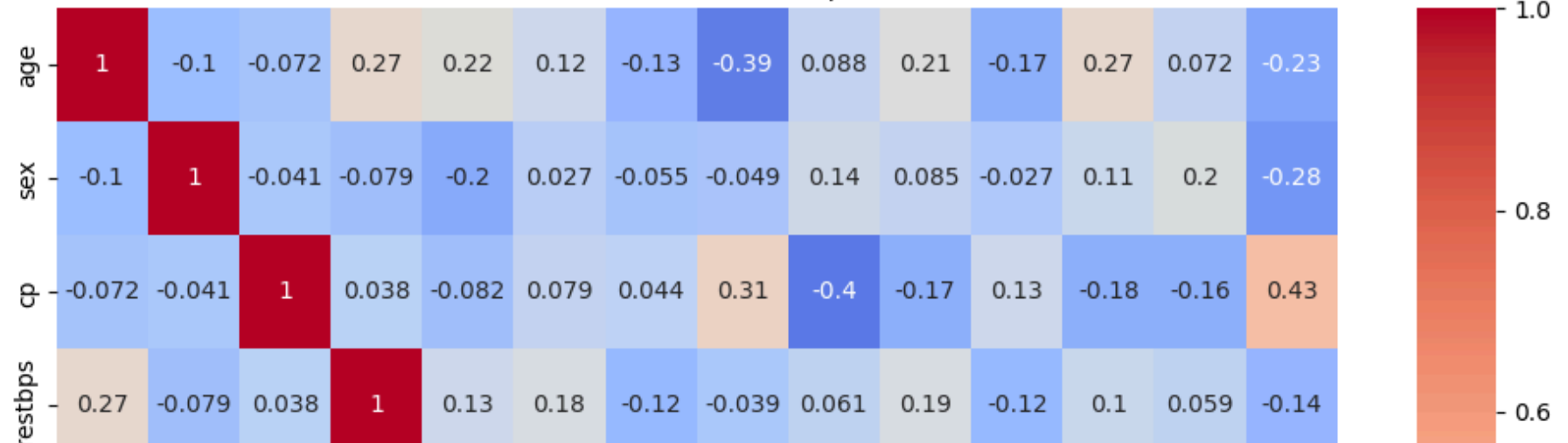
```
# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent)
    df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent)
    df[column] = df[column].fillna(mode_value)
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, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

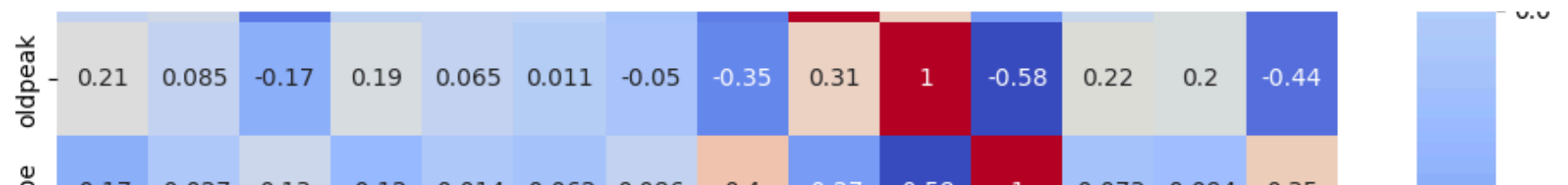

Correlation Heatmap

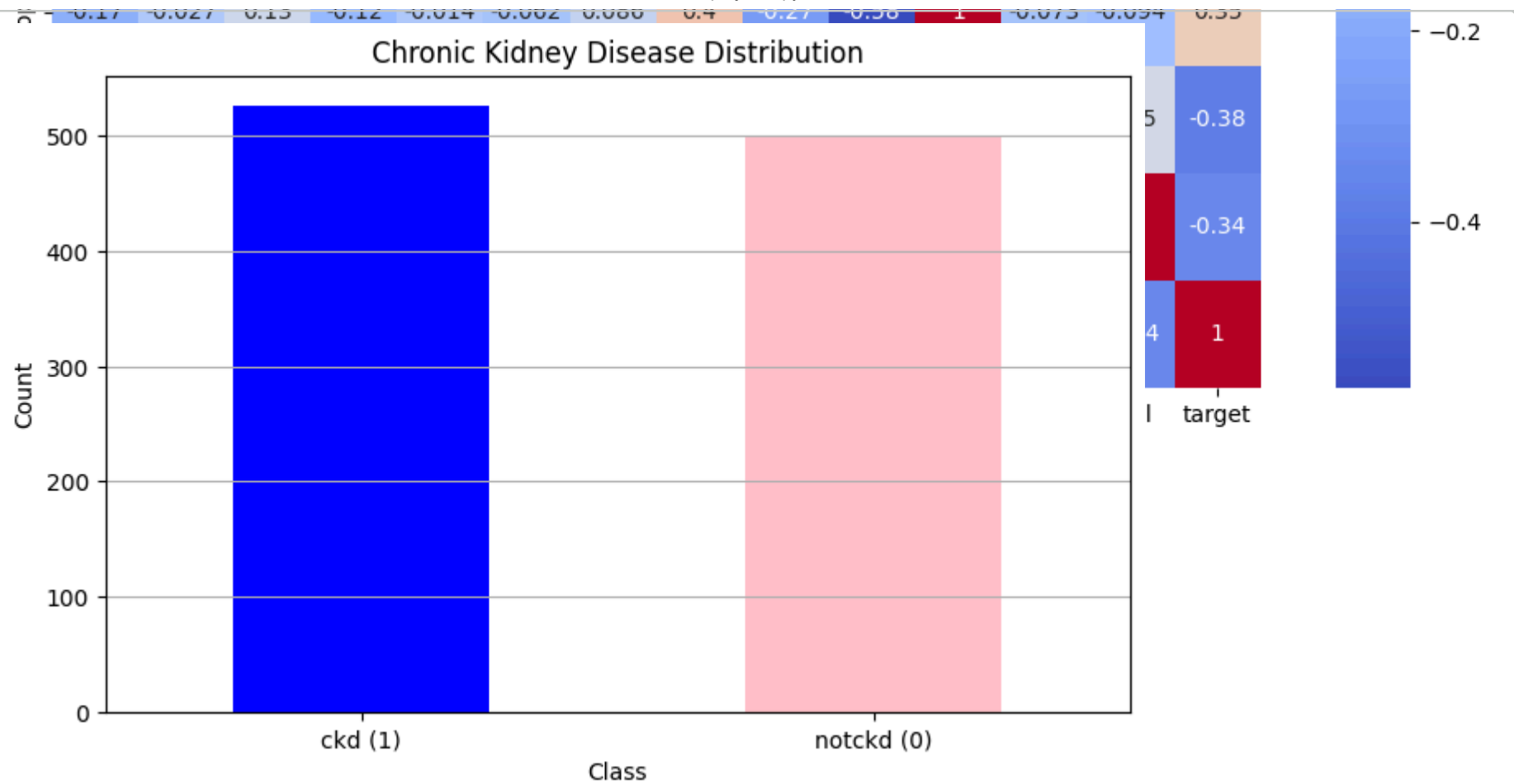


```
df.columns
```

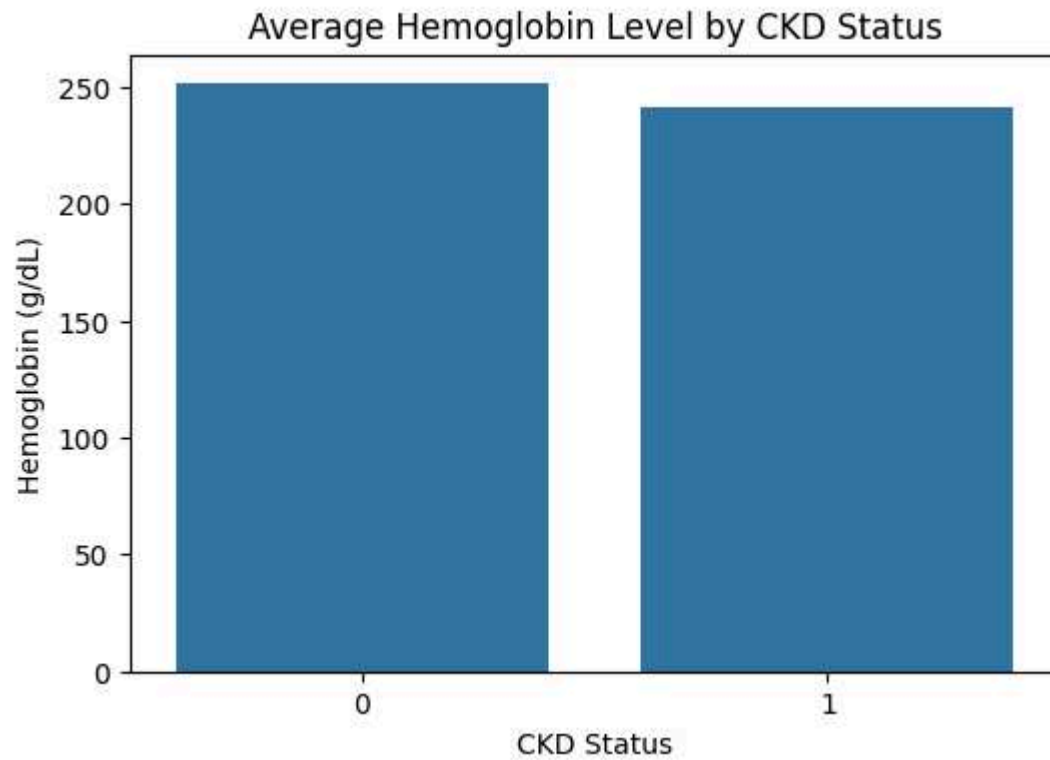
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
      'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
df['target'].value_counts()
# Create a bar chart
plt.figure(figsize=(8, 5))
df['target'].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()
```

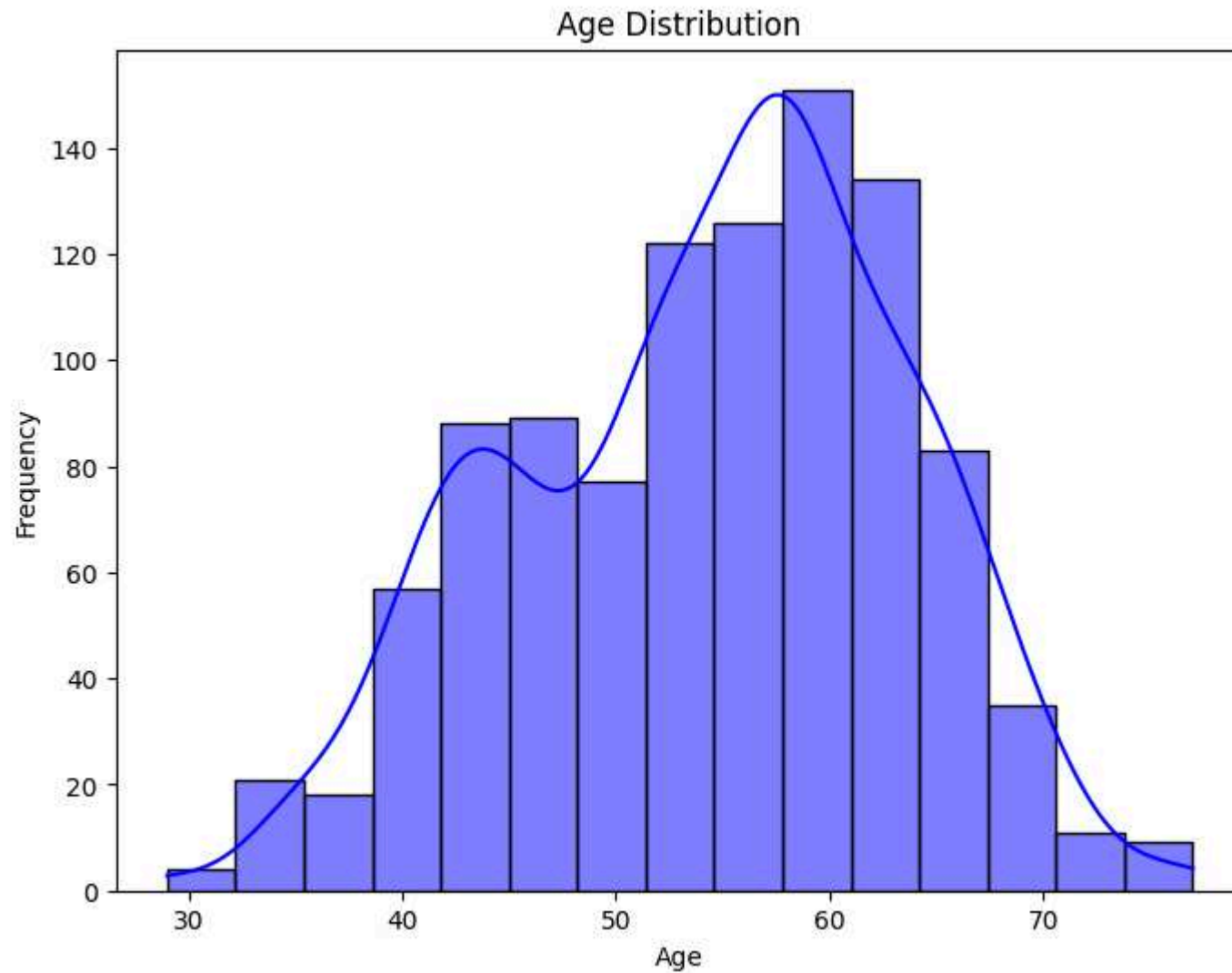




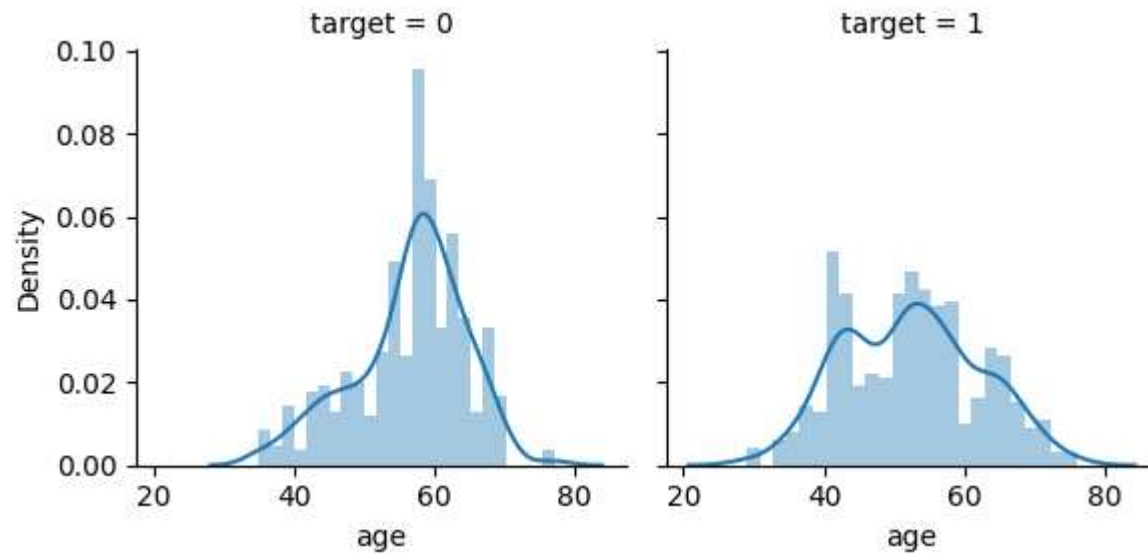
```
plt.figure(figsize=(6,4))
sns.barplot(x='target', y='chol', data=df, estimator='mean', ci=None)
plt.title("Average Hemoglobin Level by CKD Status")
plt.xlabel("CKD Status")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```



```
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()
```

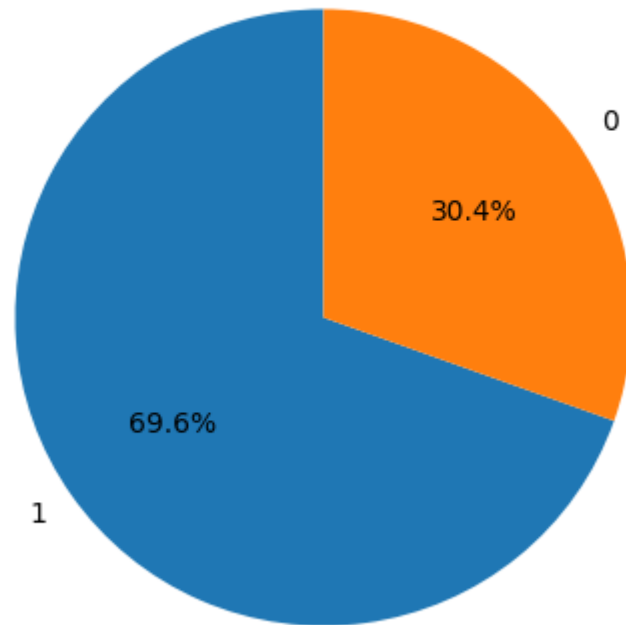


```
g = sns.FacetGrid(df,col="target")
g.map(sns.distplot,"age", bins=25)
plt.show()
```

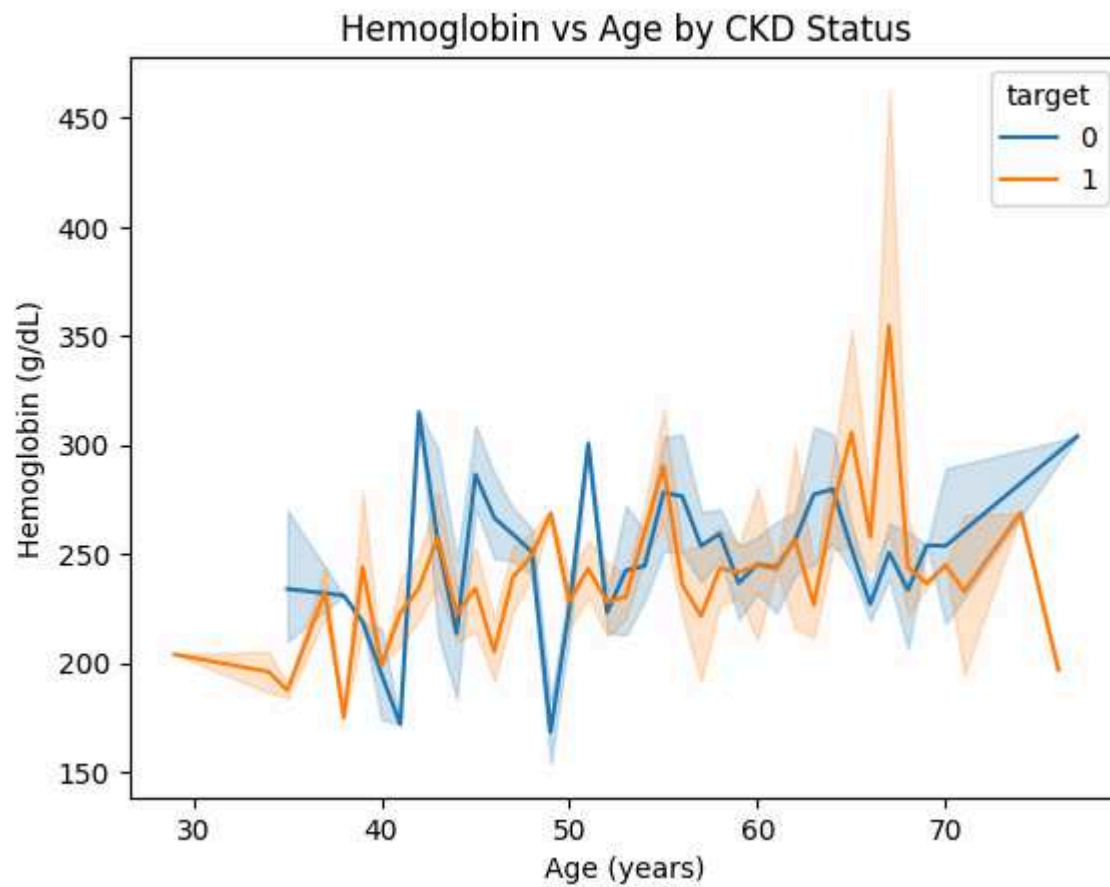


```
appet_counts = df['sex'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title("Appetite Status Distribution")
plt.show()
```

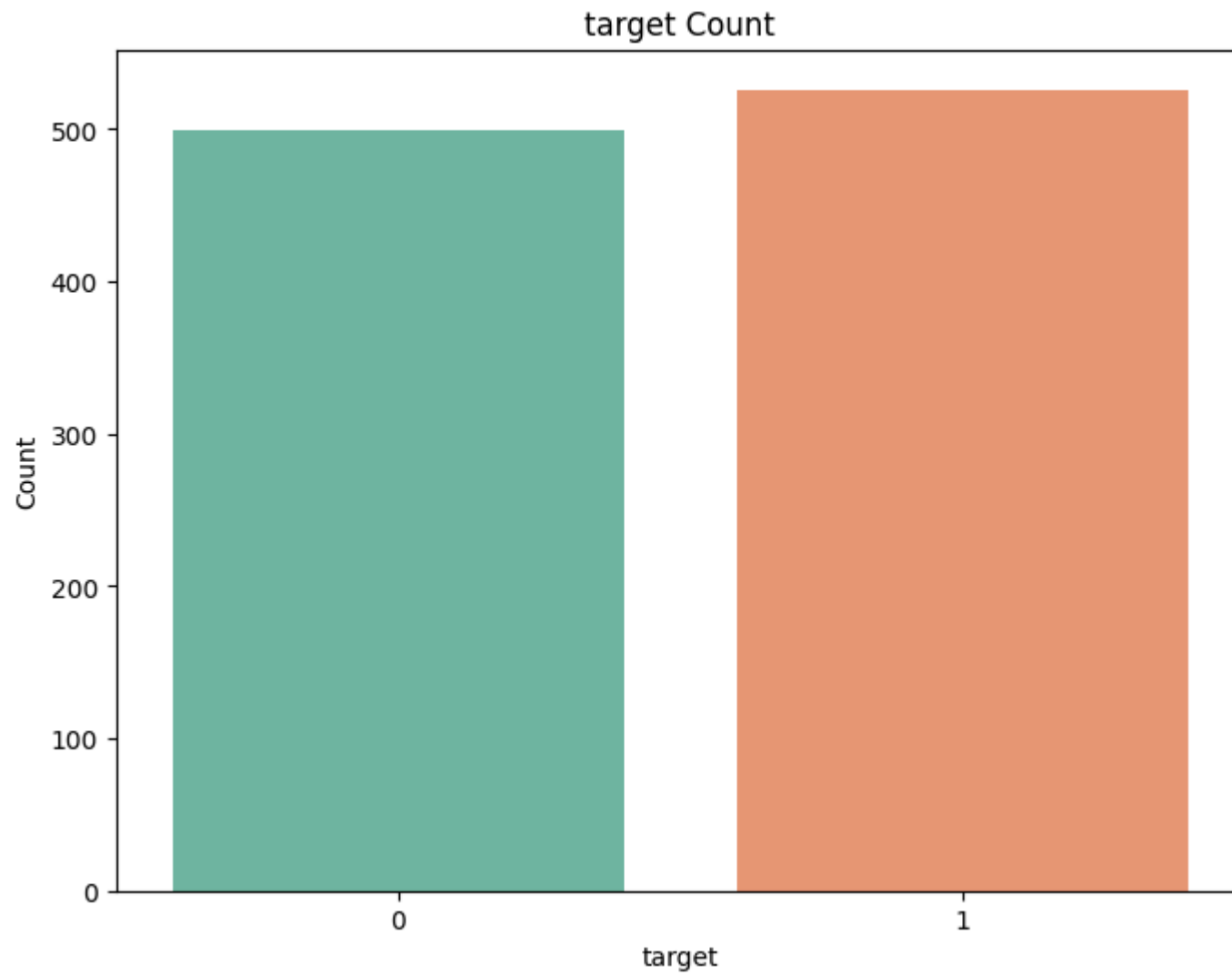
Appetite Status Distribution



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



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



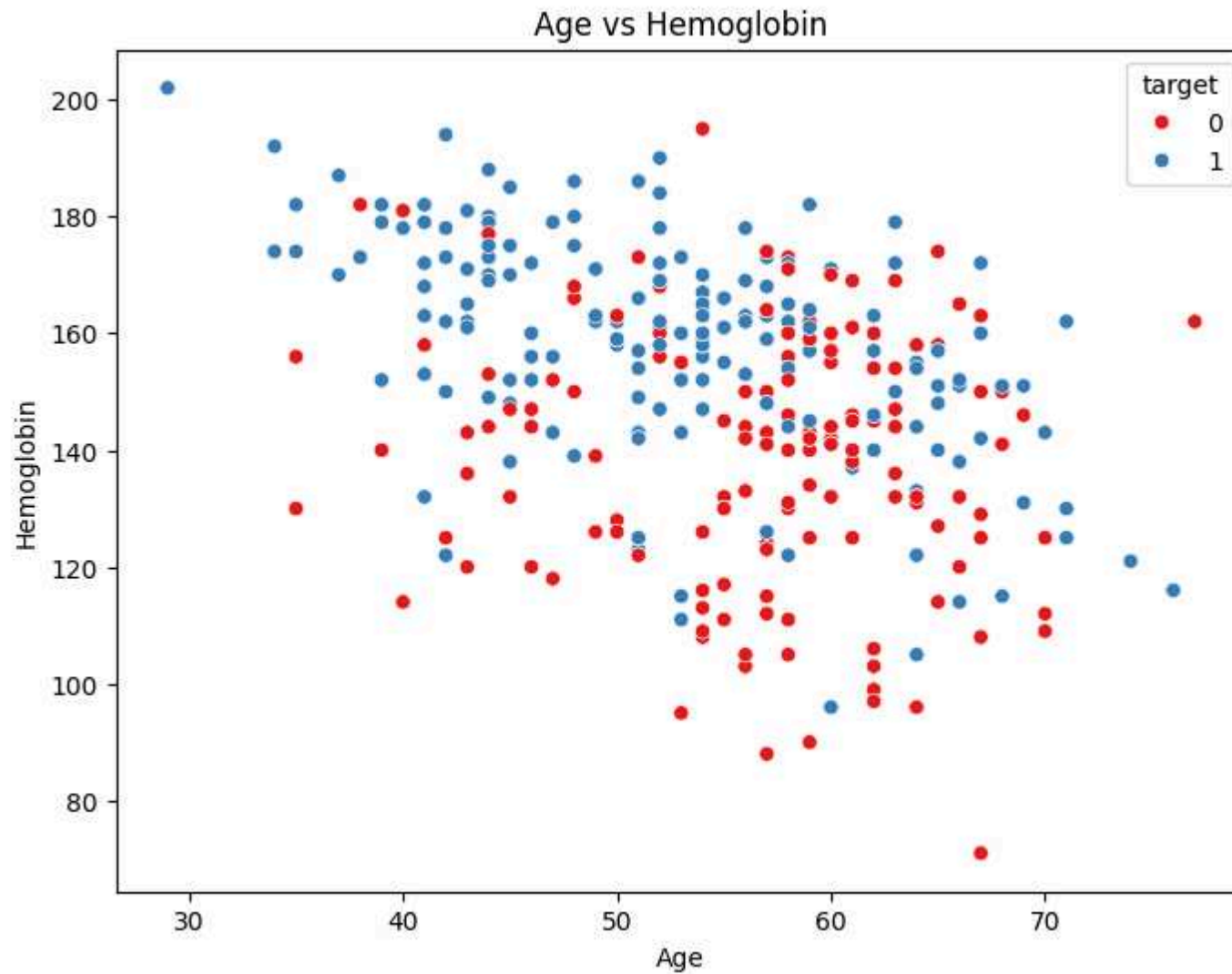
```
df.head()
```

	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	

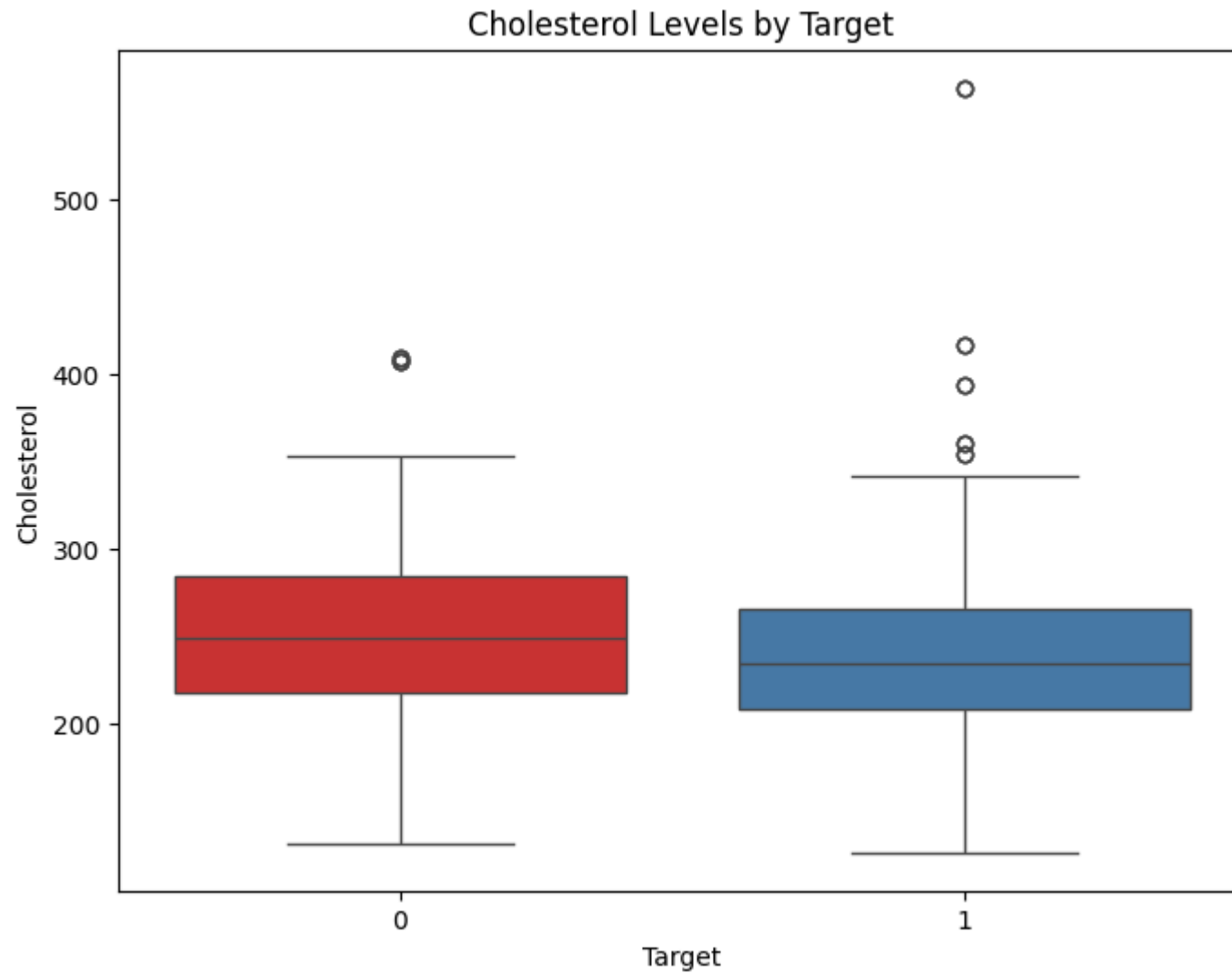
Next steps:

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

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



```
plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='chol', data=df, palette='Set1')
plt.title('Cholesterol Levels by Target')
plt.xlabel('Target')
plt.ylabel('Cholesterol')
plt.show()
```



```
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)`

```
df = pd.read_csv('/content/drive/MyDrive/cod1/project1/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()
```

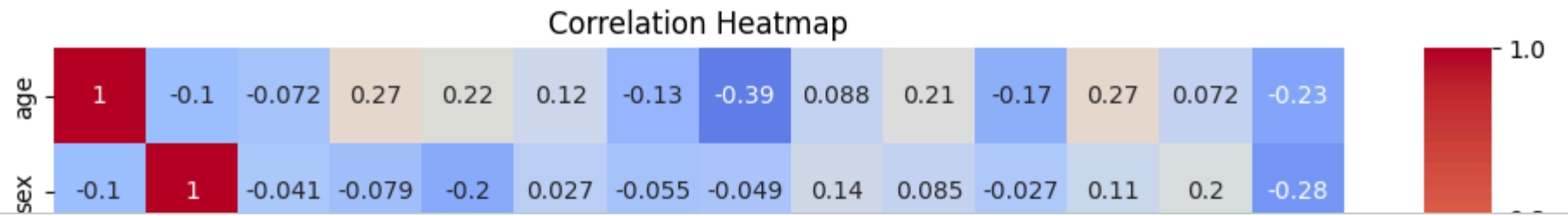
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1
mean	54.434146	0.695610	0.942439	131.611707	246.000000	0.149268	0.529756	149.114146	0.336585	
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	132.000000	0.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	152.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000	1.000000	166.000000	1.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	

```
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()
```

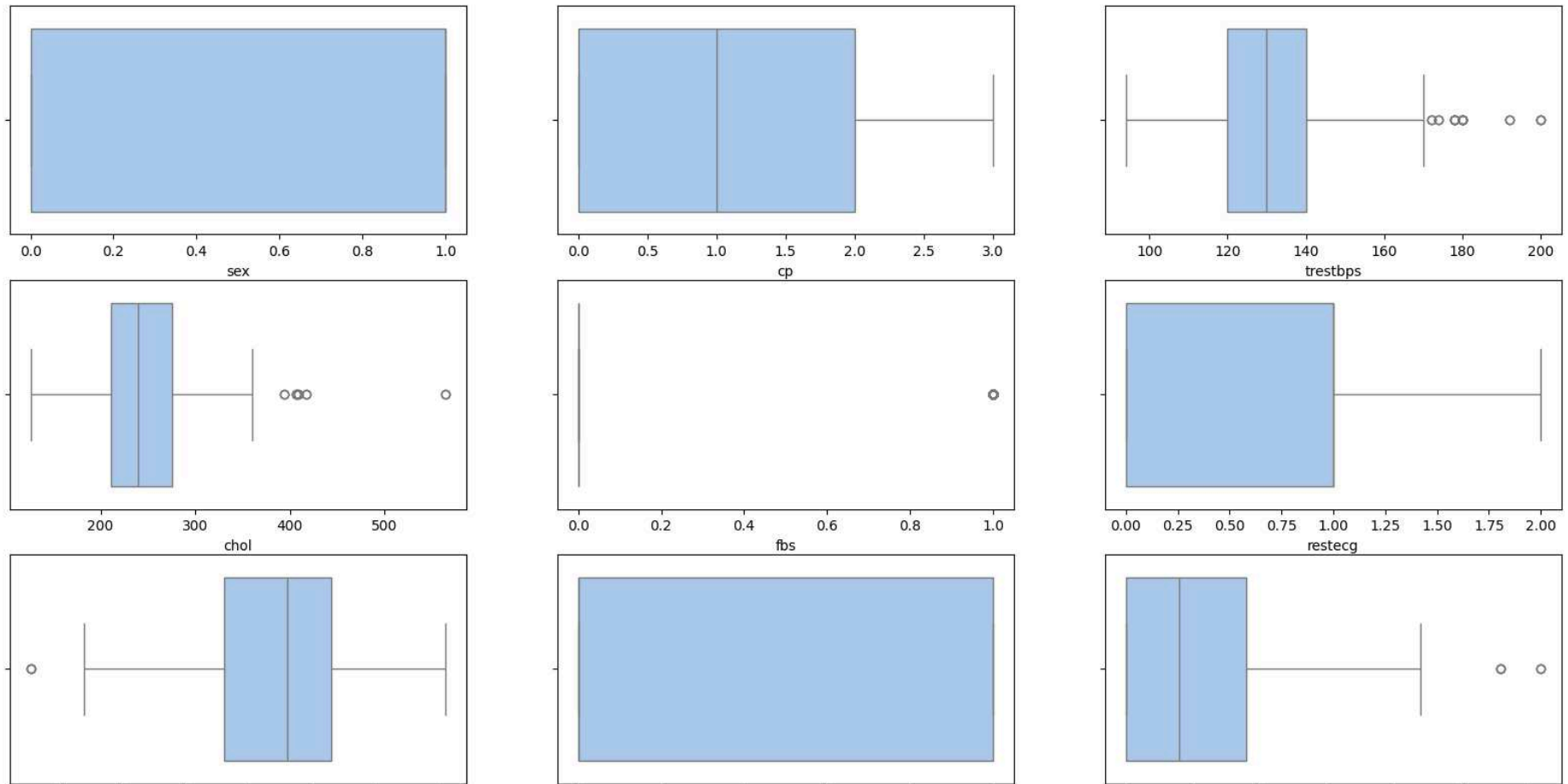



```
sns.countplot(x='target',data=df)
```



```
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()
```





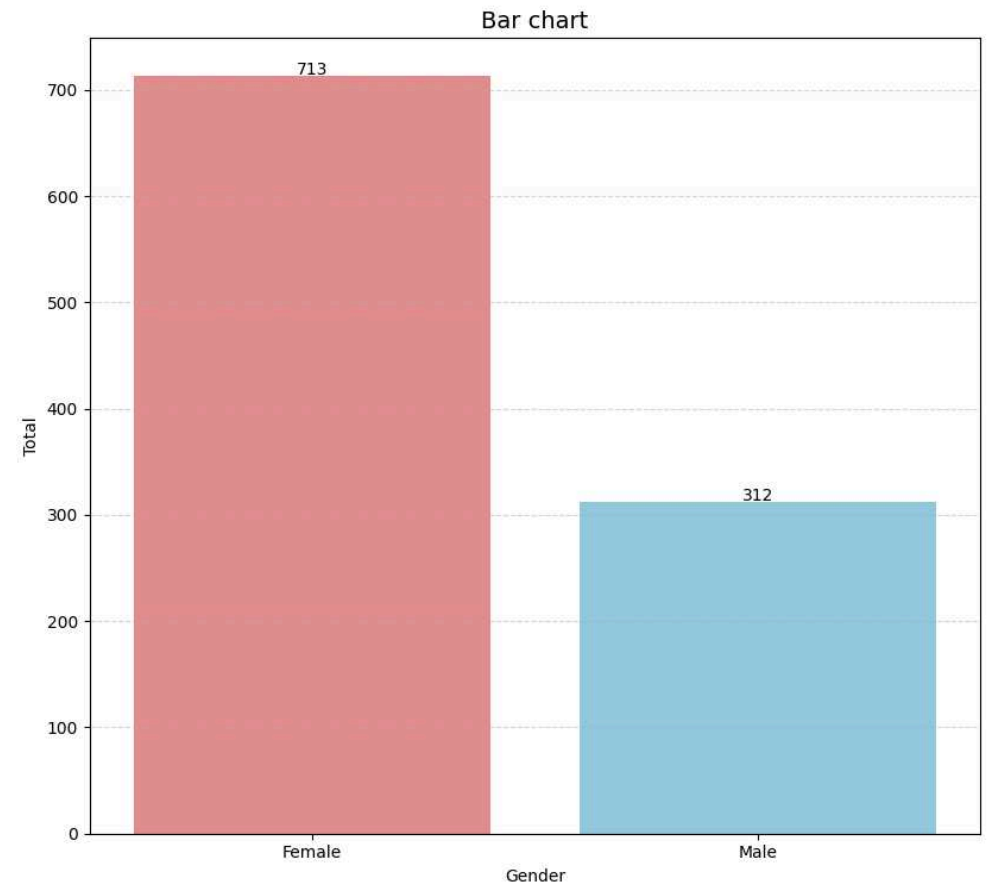
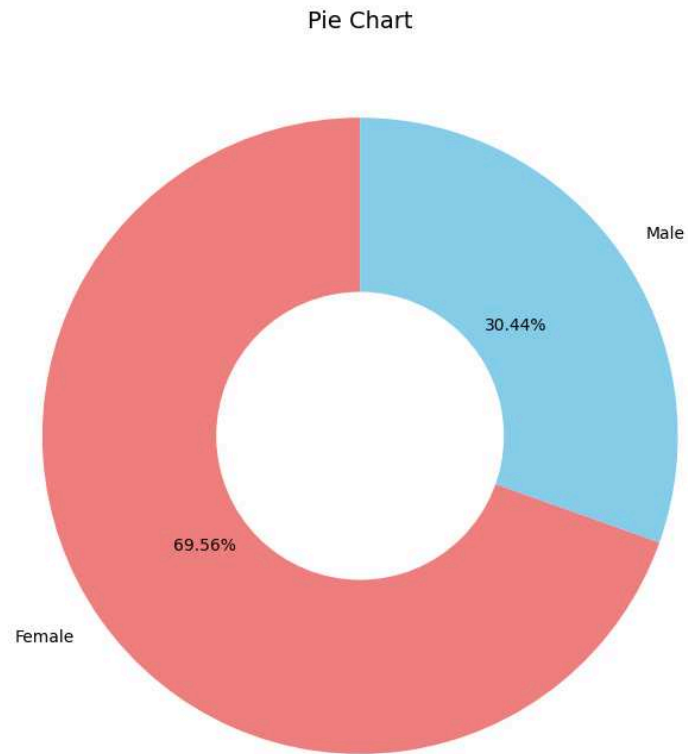
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Example colors and data (you can replace with your actual values)
colors = ['lightcoral', 'skyblue']
labels = ['Female', 'Male']
# Set order based on actual data
order = df['sex'].value_counts().index
# --- Set up the figure ---
plt.figure(figsize=(16, 8))
plt.suptitle('Sex (Gender) Distribution', fontsize=16,
```

```
fontweight='bold')
# --- Pie Chart ---

Text(0.5, 0.98, 'Sex (Gender) Distribution')
<Figure Size 1600x800 with 0 Axes>
```

```
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='%.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()
```

Sex (Gender) Distribution

```
colors = ['lightpink', 'hotpink', 'deeppink', 'mediumvioletred']
labels = ['Type 0', 'Type 2', 'Type 1', 'Type 3']
order = df['cp'].value_counts().index

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='%.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white')
plt.gca().add_artist(centre)

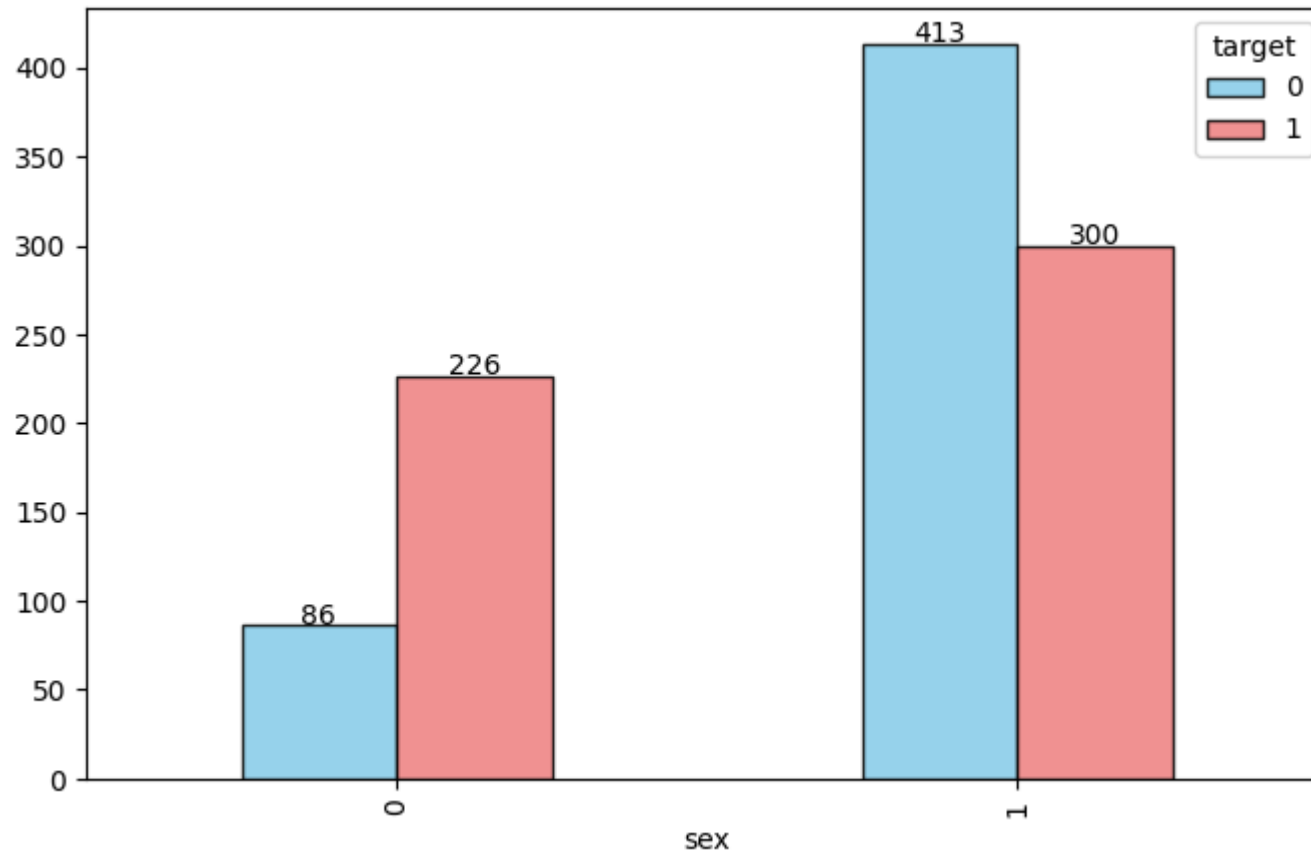
# Histogram
plt.subplot(1, 2, 2)
plt.title('Histogram', fontsize=14)
ax = sns.countplot(x='cp', data=df, palette=colors, order=order)

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)

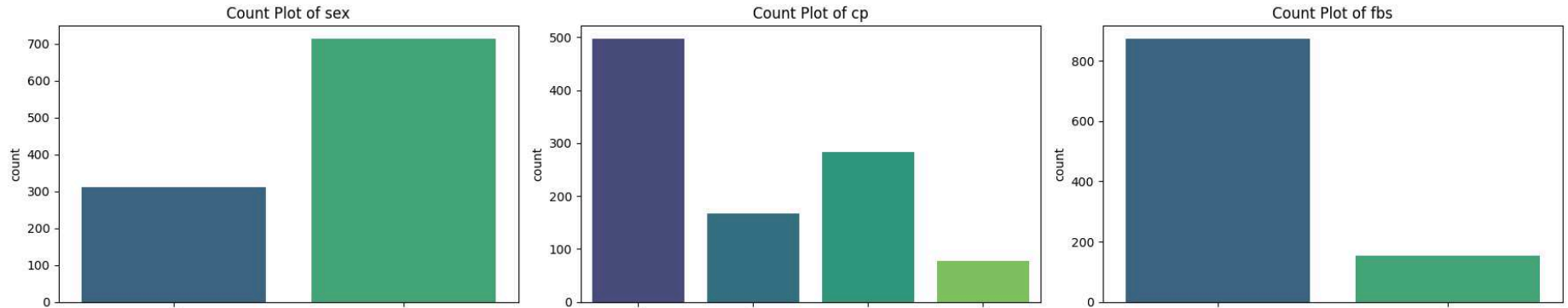
plt.show()
```


Chest Pain Type Distribution

```
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
    )
```



```
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')

    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 # هذا ينهي اللوب الداخلية ←

    # هذا السطر لازم يكون هنا خارج اللوب الداخلية ←
    axes[i].set_title(f'Target Distribution for {col}')
```