Heart Failure Dataset

Unsupervised Learning with PCA and Clustering

(AI & DS Group-6)

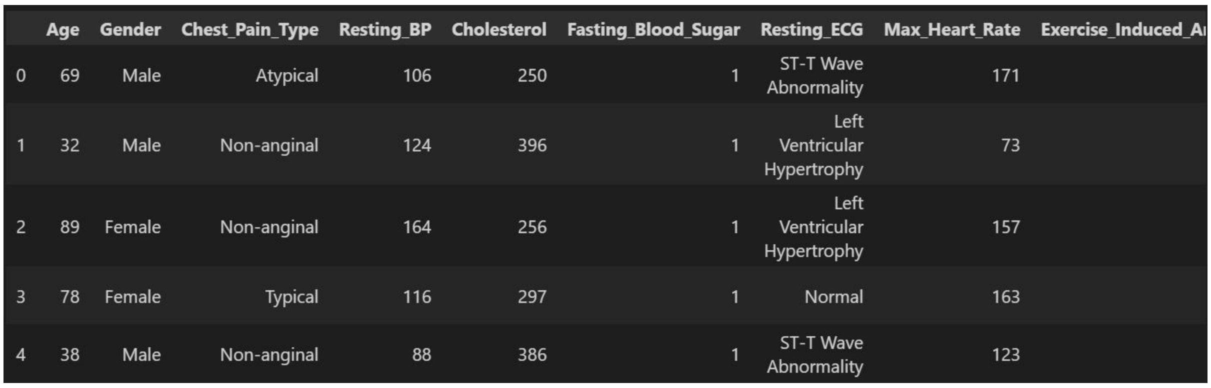
Team Members:-

NAME	REGN NO
Chittaranjan Tanty	2201020604
Angshuman Majhi	2201020686

Step 1: Data Exploration

- **Load the dataset and display the first few rows.**

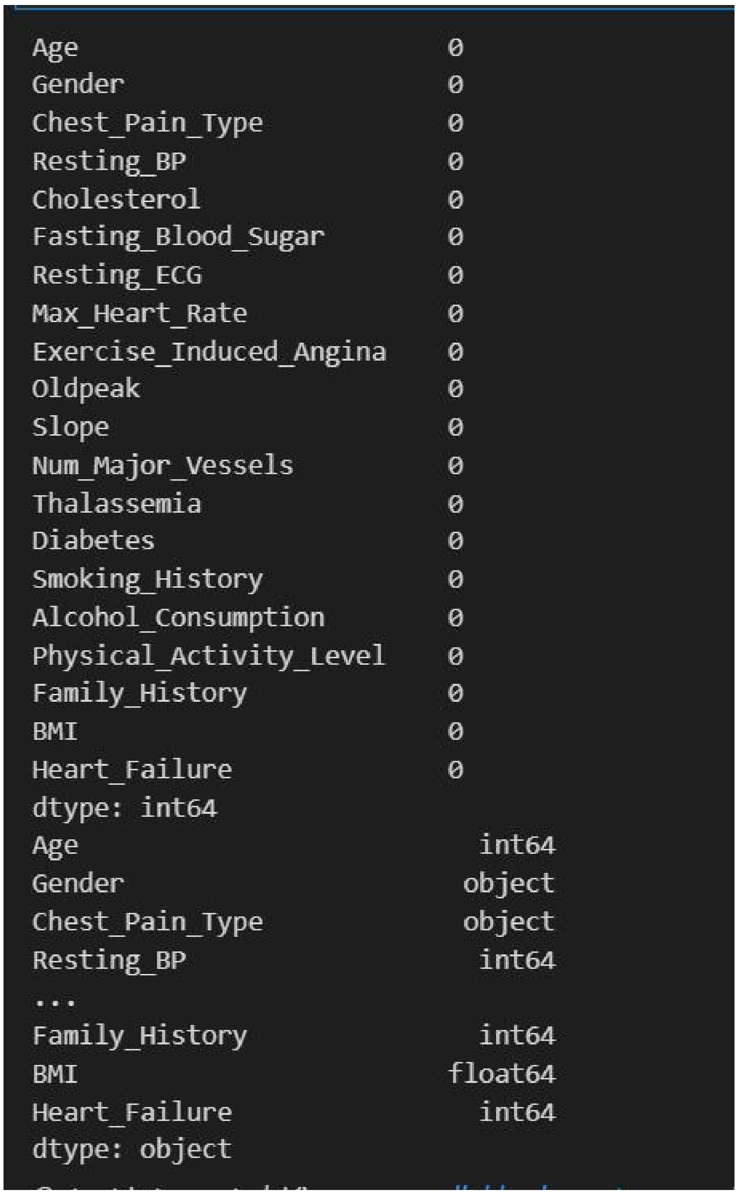
```
import pandas as pd

# Load the dataset
df = pd.read_csv('heart_failure (1).csv')

# Display the first few rows
df.head()
```



	Age	Gender	Chest_Pain_Type	Resting_BP	Cholesterol	Fasting_Blood_Sugar	Resting_ECG	Max_Heart_Rate	Exercise_Induced_A
0	69	Male	Atypical	106	250	1	ST-T Wave Abnormality	171	
1	32	Male	Non-anginal	124	396	1	Left Ventricular Hypertrophy	73	
2	89	Female	Non-anginal	164	256	1	Left Ventricular Hypertrophy	157	
3	78	Female	Typical	116	297	1	Normal	163	
4	38	Male	Non-anginal	88	386	1	ST-T Wave Abnormality	123	

- **Check for missing values and confirm data types.**

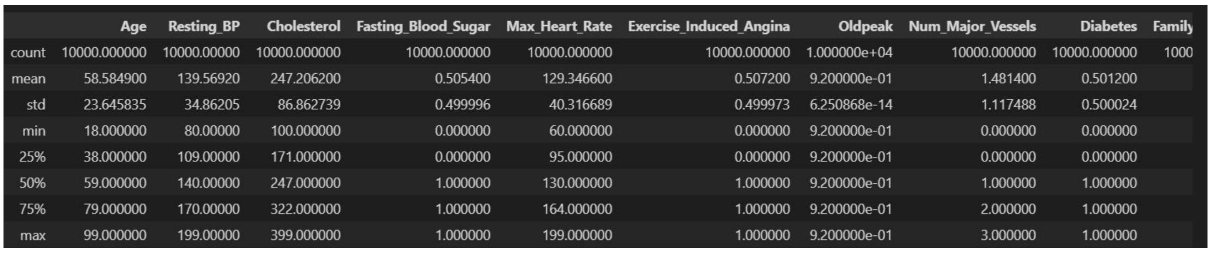
```
# Check for missing values
print(df.isnull().sum())

# Check data types
print(df.dtypes)
```



```
Age                        0
Gender                     0
Chest_Pain_Type            0
Resting_BP                 0
Cholesterol                0
Fasting_Blood_Sugar        0
Resting_ECG                0
Max_Heart_Rate             0
Exercise_Induced_Angina    0
Oldpeak                    0
Slope                      0
Num_Major_Vessels          0
Thalassemia                0
Diabetes                   0
Smoking_History            0
Alcohol_Consumption        0
Physical_Activity_Level    0
Family_History             0
BMI                        0
Heart_Failure              0
dtype: int64
Age                          int64
Gender                      object
Chest_Pain_Type             object
Resting_BP                   int64
...
Family_History               int64
BMI                        float64
Heart_Failure                int64
dtype: object
```

- **Generate summary statistics for each feature.**

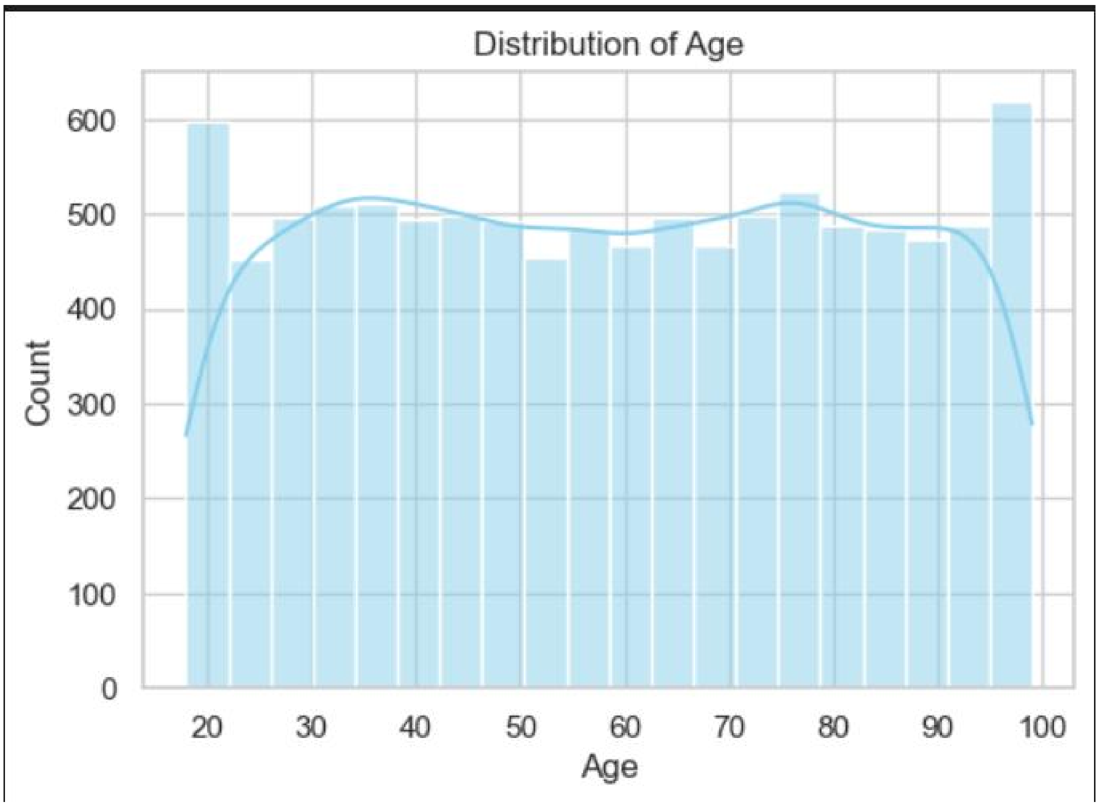
```
# Summary statistics
df.describe()
```

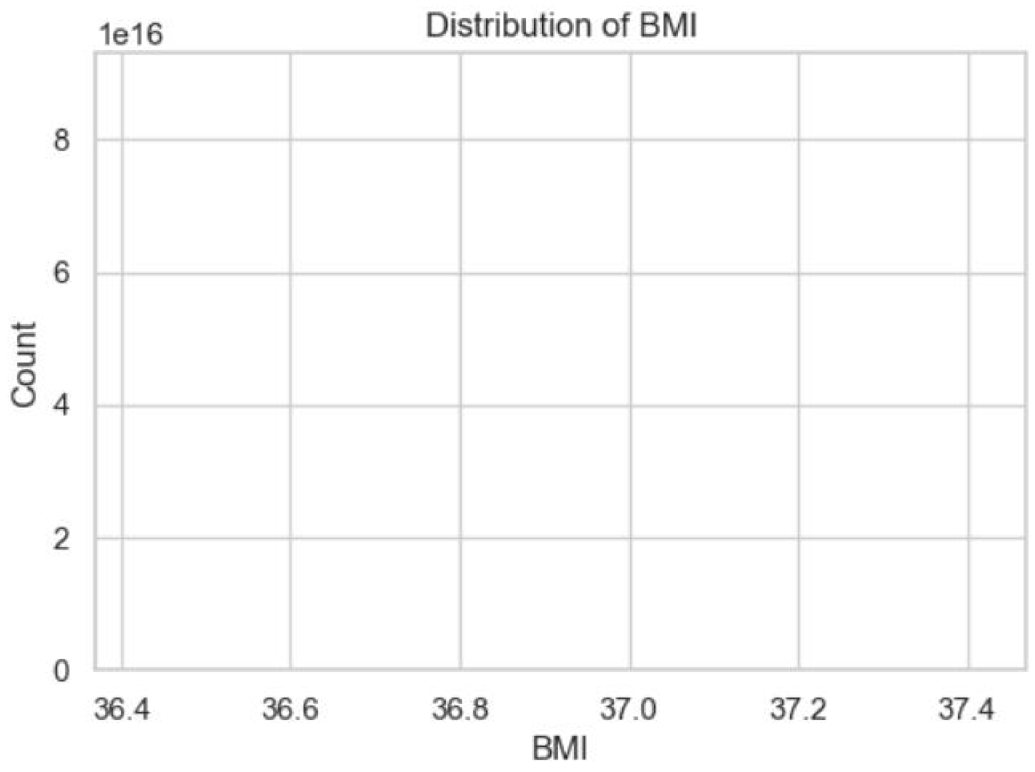


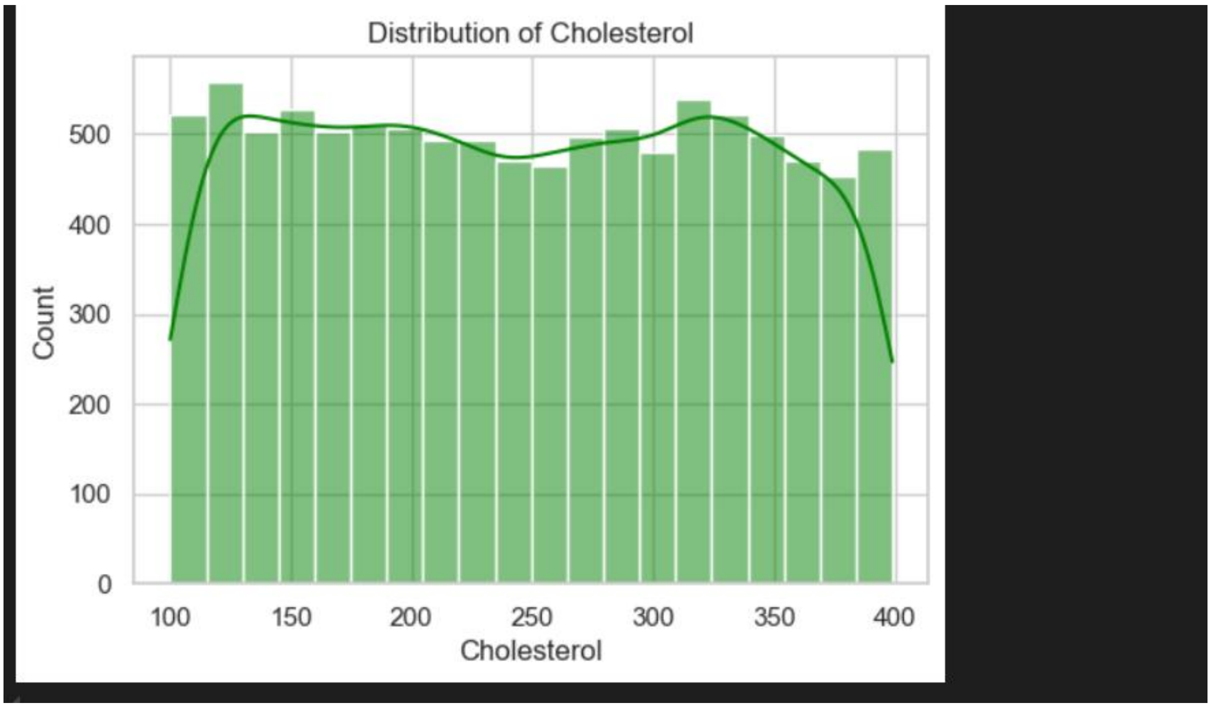
	Age	Resting_BP	Cholesterol	Fasting_Blood_Sugar	Max_Heart_Rate	Exercise_Induced_Angina	Oldpeak	Num_Major_Vessels	Diabetes	Family
count	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	1.000000e+04	10000.000000	10000.000000	1000
mean	58.584900	139.56920	247.206200	0.505400	129.346600	0.507200	9.200000e-01	1.481400	0.501200	
std	23.645835	34.86205	86.862739	0.499996	40.316689	0.499973	6.250868e-14	1.117488	0.500024	
min	18.000000	80.00000	100.000000	0.000000	60.000000	0.000000	9.200000e-01	0.000000	0.000000	
25%	38.000000	109.00000	171.000000	0.000000	95.000000	0.000000	9.200000e-01	0.000000	0.000000	
50%	59.000000	140.00000	247.000000	1.000000	130.000000	1.000000	9.200000e-01	1.000000	1.000000	
75%	79.000000	170.00000	322.000000	1.000000	164.000000	1.000000	9.200000e-01	2.000000	1.000000	
max	99.000000	199.00000	399.000000	1.000000	199.000000	1.000000	9.200000e-01	3.000000	1.000000	

- **Visualize distributions of key features (e.g., age, ejection_fraction, serum_creatinine).**

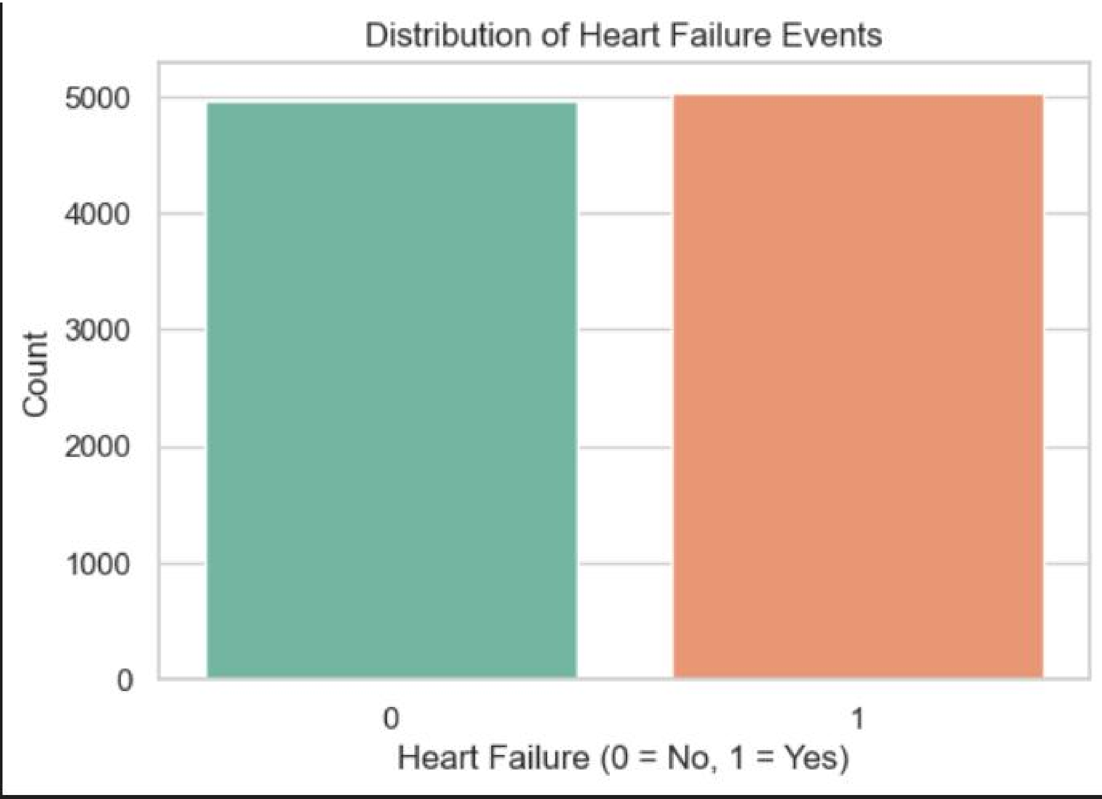
```
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of Age
plt.figure(figsize=(6, 4))
sns.histplot(df['Age'], kde=True, bins=20, color='skyblue')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

# Distribution of BMI
plt.figure(figsize=(6, 4))
sns.histplot(df['BMI'], kde=True, bins=20, color='orange')
plt.title('Distribution of BMI')
plt.xlabel('BMI')
plt.ylabel('Count')
plt.show()

# Distribution of Cholesterol
plt.figure(figsize=(6, 4))
sns.histplot(df['Cholesterol'], kde=True, bins=20, color='green')
plt.title('Distribution of Cholesterol')
plt.xlabel('Cholesterol')
plt.ylabel('Count')
plt.show()

```







```
# Countplot
plt.figure(figsize=(6, 4))
sns.countplot(x='Heart_Failure', data=df, palette='Set2')
plt.title('Distribution of Heart Failure Events')
plt.xlabel('Heart Failure (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()

# Proportions
print(df['Heart_Failure'].value_counts(normalize=True))
```

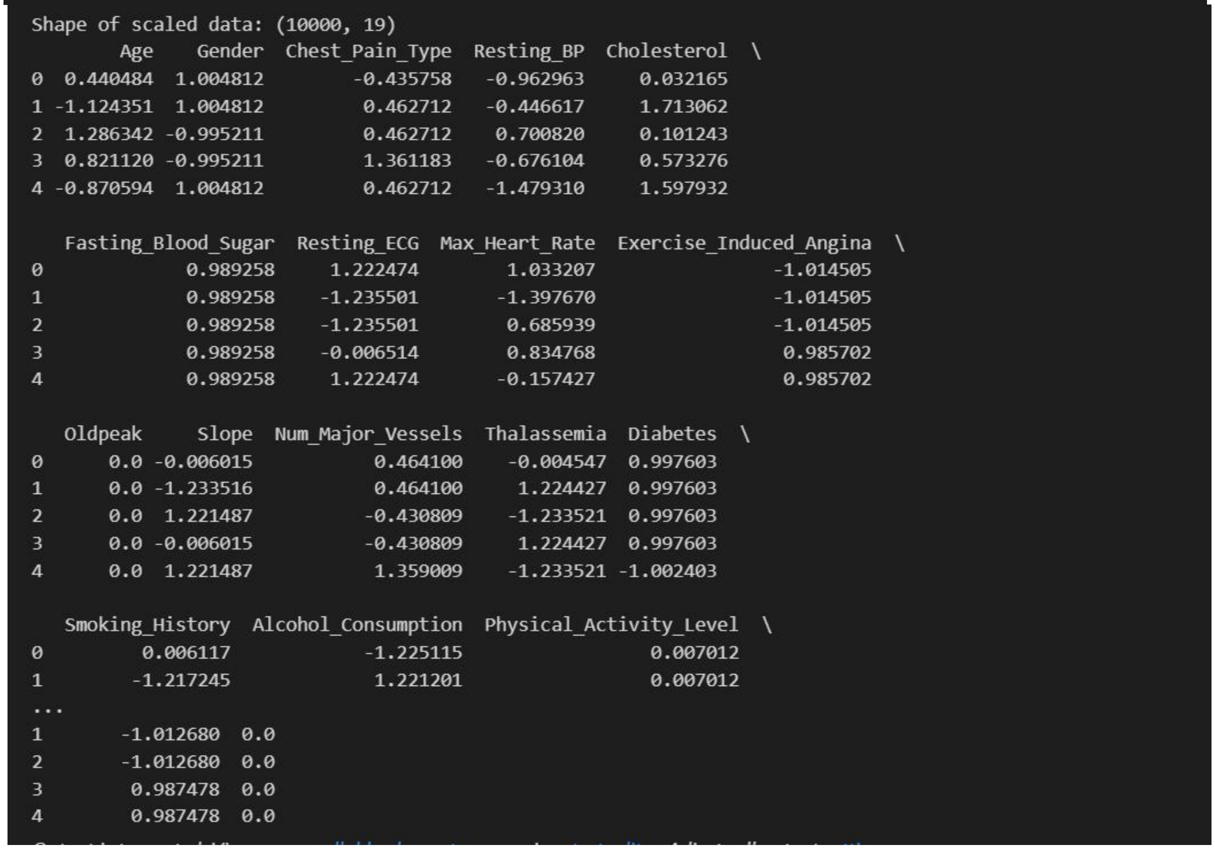


```
1    0.5036
0    0.4964
Name: Heart_Failure, dtype: float64
```

Step 2: Feature Selection and Preprocessing

```
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load the dataset
df = pd.read_csv("heart_failure (1).csv")

# Drop the target column for unsupervised learning
df_unsupervised = df.drop(columns=['Heart_Failure'])

# Identify non-numeric columns
non_numeric_columns =
df_unsupervised.select_dtypes(exclude=['number']).columns.tolist()
```

```python
# Label encode the categorical features
label_encoders = {}
for column in non_numeric_columns:
    le = LabelEncoder()
    df_unsupervised[column] = le.fit_transform(df_unsupervised[column])
    label_encoders[column] = le

# Standardize all features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_unsupervised)

# Convert to DataFrame for easier handling
scaled_df = pd.DataFrame(scaled_data, columns=df_unsupervised.columns)

# Confirm shape and preview
print("Shape of scaled data:", scaled_df.shape)
print(scaled_df.head())
```



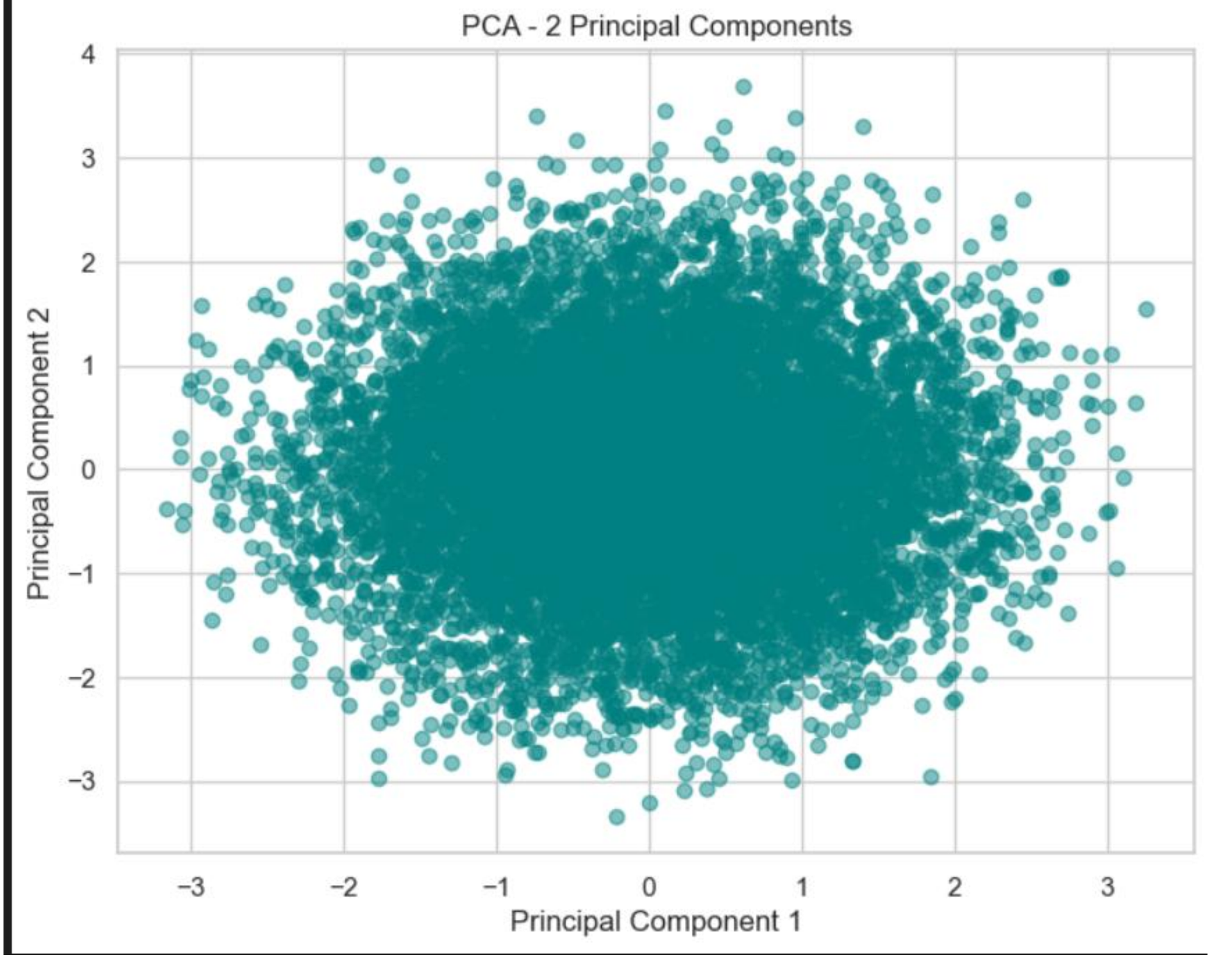
```
Shape of scaled data: (10000, 19)
        Age    Gender  Chest_Pain_Type  Resting_BP  Cholesterol  \
0  0.440484  1.004812        -0.435758   -0.962963     0.032165
1 -1.124351  1.004812         0.462712   -0.446617     1.713062
2  1.286342 -0.995211         0.462712    0.700820     0.101243
3  0.821120 -0.995211         1.361183   -0.676104     0.573276
4 -0.870594  1.004812         0.462712   -1.479310     1.597932

   Fasting_Blood_Sugar  Resting_ECG  Max_Heart_Rate  Exercise_Induced_Angina  \
0             0.989258     1.222474        1.033207                -1.014505
1             0.989258    -1.235501       -1.397670                -1.014505
2             0.989258    -1.235501        0.685939                -1.014505
3             0.989258    -0.006514        0.834768                 0.985702
4             0.989258     1.222474       -0.157427                 0.985702

   Oldpeak     Slope  Num_Major_Vessels  Thalassemia  Diabetes  \
0      0.0 -0.006015           0.464100    -0.004547  0.997603
1      0.0 -1.233516           0.464100     1.224427  0.997603
2      0.0  1.221487          -0.430809    -1.233521  0.997603
3      0.0 -0.006015          -0.430809     1.224427  0.997603
4      0.0  1.221487           1.359009    -1.233521 -1.002403

   Smoking_History  Alcohol_Consumption  Physical_Activity_Level  \
0         0.006117            -1.225115                 0.007012
1        -1.217245             1.221201                 0.007012
...
1       -1.012680  0.0
2       -1.012680  0.0
3        0.987478  0.0
4        0.987478  0.0
```

Step 3: Dimensionality Reduction with PCA

- Apply PCA to reduce the dataset to 2 principal components.
- Print the explained variance ratio for each component.

- Create a scatter plot using the two principal components

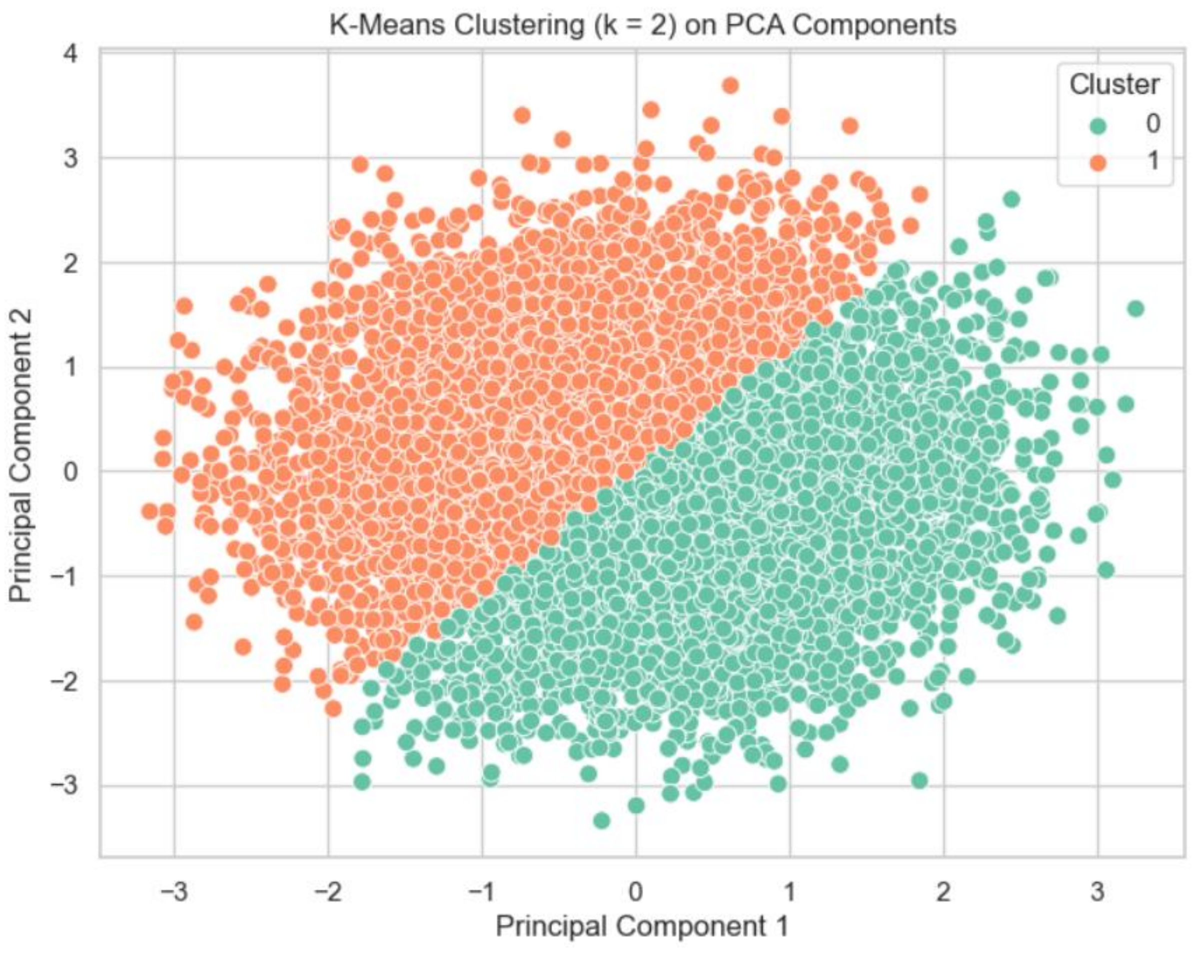
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Apply PCA to reduce to 2 components
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_df)

# Create a DataFrame with principal components
pca_df = pd.DataFrame(pca_components, columns=['PC1', 'PC2'])

# Print explained variance ratio
print("Explained variance ratio:", pca.explained_variance_ratio_)

# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], alpha=0.5, color='teal')
plt.title('PCA - 2 Principal Components')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()

```



Step 4: Clustering with K-Means

- Use the PCA-reduced data to apply K-Means clustering.
- Select an appropriate number of clusters (e.g., 2 or 3) and explain your choice.
- Assign a cluster label to each record.
- Visualize the clusters using the PCA plot with different colors for each cluster.

```python

from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt

# Choose the number of clusters
n_clusters = 2

# Apply K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(pca_df)

# Add cluster labels to the PCA DataFrame
pca_df['Cluster'] = cluster_labels

# Visualize the clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='Cluster', palette='Set2',
s=60)
plt.title(f'K-Means Clustering (k = {n_clusters}) on PCA Components')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()

```



Step 5: Interpretation

1. What does reducing a dataset to 2 principal components mean?

In simple terms:

Imagine dataset has **lots of features** (like age, cholesterol, heart rate, etc.). That's like trying to understand a story with 19 different characters talking at once.

Principal Component Analysis (PCA) finds the most important patterns and combines them into **just 2 new features** (called *principal components*) that capture as much of the original information as possible.

These two components let us **visualize** the data in 2D while keeping most of its structure.

So, PCA is like summarizing a long book into a 2-paragraph summary that still covers the key ideas.

2. How well are the clusters separated in the scatter plot?

When looking at the PCA scatter plot with colored clusters:

- If the clusters are **clearly separated** (like distinct blobs), it means the model found **strong patterns** in the data.
- If the clusters **overlap a lot**, it suggests the data points aren't easily distinguishable using the features you provided.

So this interpretation here depends on how much visual separation we see between the colored groups in the plot.

3. Do these clusters correspond to real-world patient groups or clinical patterns?

They *might*

- The clusters were made **without any labels** (unsupervised), so they're based only on feature patterns.
- If one cluster mostly includes **older patients with high cholesterol and low activity**, it might reflect a high-risk group.
- To confirm this, we can look back at the original features (e.g., average age, BMI, etc. in each cluster) or see how many patients in each cluster had heart failure.