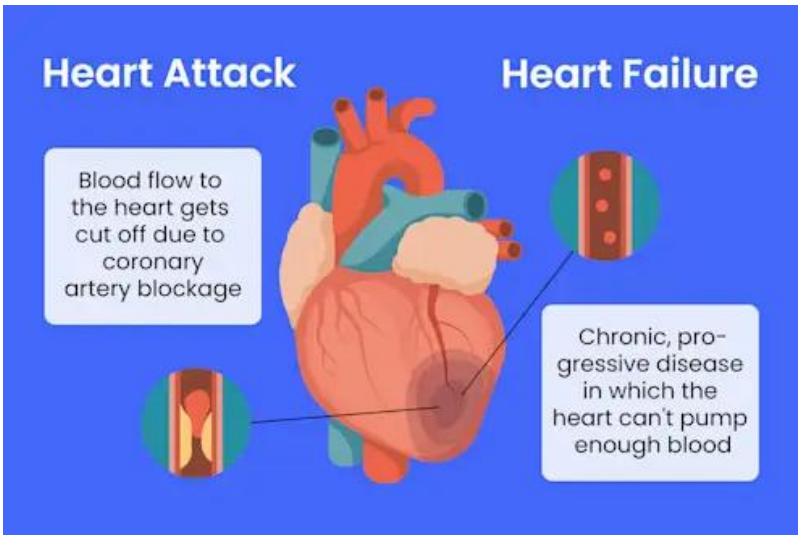
AI & DS

Topic: A report on heart failure DataSet



Sl.	Name	Reg No.
1.	Smruti Ranjan Sahoo	2201020417
2.	Subhasmita Dash	2201020422
3.	Shreya Tripathi	2201020261
4.	Supriya Sahoo	2201020438

INTRODUCTION:

The Heart Failure dataset consists of clinical and lifestyle records for 10,000 patients, captured across 20 distinct features. It is designed to support the analysis and prediction of heart failure outcomes. The dataset includes demographic attributes such as age and gender, along with detailed medical parameters like chest pain type, resting blood pressure, cholesterol levels, maximum heart rate, and ST depression (Oldpeak). It also records whether patients have diabetes, high fasting blood sugar, and exercise-induced angina. Additionally, it incorporates lifestyle and genetic risk indicators such as smoking history, alcohol consumption, physical activity level, body mass index (BMI), and family history of cardiovascular disease. Advanced clinical indicators like the number of major blood vessels visualized via fluoroscopy and the presence of thalassemia are also included. The key feature, Heart_Failure, is a binary target variable indicating whether the patient experienced a heart failure event. This dataset serves as a robust foundation for developing machine learning models, conducting risk assessments, and deriving insights to support early diagnosis, personalized treatment planning, and preventive care in cardiology.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans


df = pd.read_csv(r"/heart_failure.csv")


df.head(5)
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(10000, 20)
```

```
print(df.info())
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      10000 non-null  int64
 1   Gender                   10000 non-null  object
 2   Chest_Pain_Type          10000 non-null  object
 3   Resting_BP               10000 non-null  int64
 4   Cholesterol              10000 non-null  int64
 5   Fasting_Blood_Sugar      10000 non-null  int64
 6   Resting_ECG              10000 non-null  object
 7   Max_Heart_Rate           10000 non-null  int64
 8   Exercise_Induced_Angina  10000 non-null  int64
 9   Oldpeak                  10000 non-null  float64
 10  Slope                    10000 non-null  object
 11  Num_Major_Vessels        10000 non-null  int64
 12  Thalassemia              10000 non-null  object
 13  Diabetes                 10000 non-null  int64
 14  Smoking_History          10000 non-null  object
 15  Alcohol_Consumption      6650 non-null   object
 16  Physical_Activity_Level  10000 non-null  object
 17  Family_History           10000 non-null  int64
 18  BMI                      10000 non-null  float64
 19  Heart_Failure            10000 non-null  int64
dtypes: float64(2), int64(10), object(8)
memory usage: 1.5+ MB
None
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
```

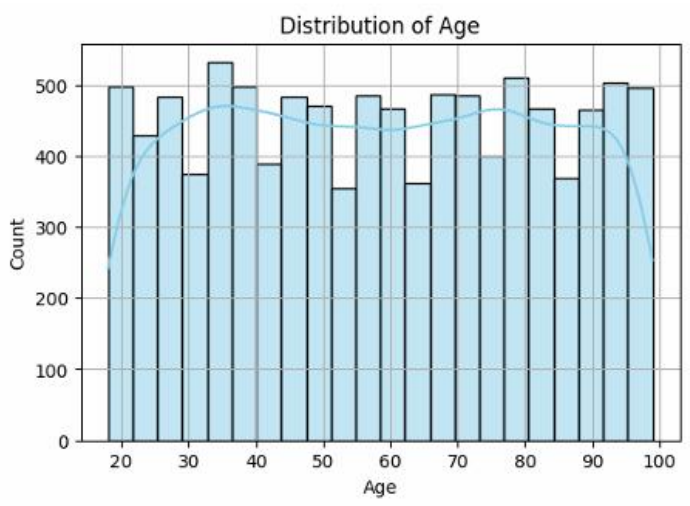
```
Smoking_History               0
Alcohol_Consumption        3350
Physical_Activity_Level       0
Family_History                0
BMI                           0
Heart_Failure                 0
dtype: int64
```

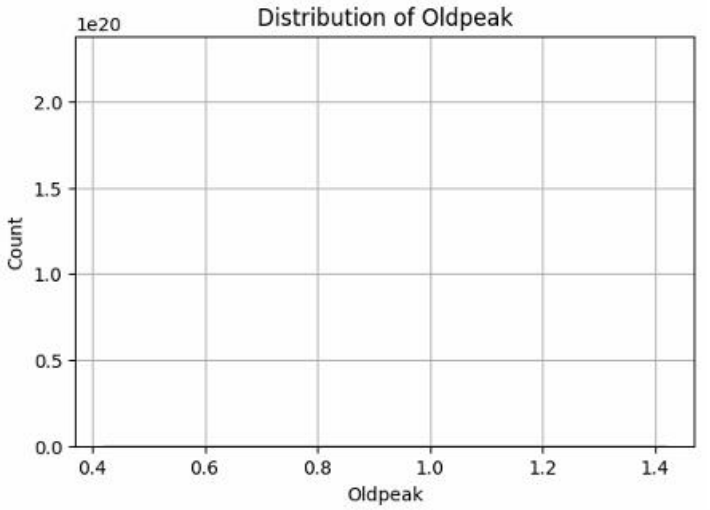
```
df = df.drop(columns=["Alcohol_Consumption"])
```

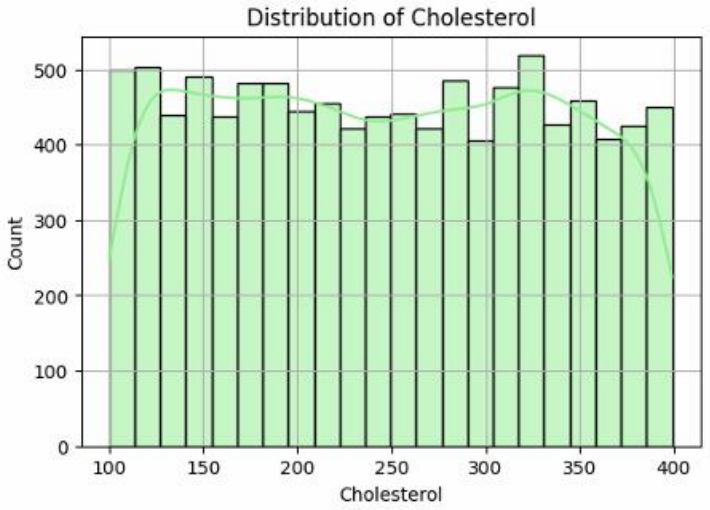
```
# Summary statistics
summary_stats = df.describe(include='all')
print(summary_stats)
```

```
                Age  Gender Chest_Pain_Type   Resting_BP   Cholesterol  \
count   10000.000000   10000           10000  10000.00000  10000.000000
unique           NaN       2               4          NaN           NaN
top              NaN  Female        Atypical          NaN           NaN
freq             NaN    5024            2583          NaN           NaN
mean       58.584900     NaN             NaN    139.56920    247.206200
std        23.645835     NaN             NaN     34.86205     86.862739
min        18.000000     NaN             NaN     80.00000    100.000000
25%        38.000000     NaN             NaN    109.00000    171.000000
50%        59.000000     NaN             NaN    140.00000    247.000000
75%        79.000000     NaN             NaN    170.00000    322.000000
max        99.000000     NaN             NaN    199.00000    399.000000

        Fasting_Blood_Sugar Resting_ECG  Max_Heart_Rate  \
count          10000.000000       10000    10000.000000
unique                  NaN           3             NaN
top                     NaN      Normal             NaN
freq                    NaN        3379             NaN
mean               0.505400         NaN      129.346600
std                0.499996         NaN       40.316689
min                0.000000         NaN       60.000000
25%                0.000000         NaN       95.000000
50%                1.000000         NaN      130.000000
75%                1.000000         NaN      164.000000
max                1.000000         NaN      199.000000

        Exercise_Induced_Angina       Oldpeak  Slope  Num_Major_Vessels  \
count              10000.000000  1.000000e+04  10000       10000.000000
unique                      NaN           NaN      3                NaN
top                         NaN           NaN   Flat                NaN
freq                        NaN           NaN   3363                NaN
mean                   0.507200  9.200000e-01    NaN           1.481400
std                    0.499973  6.250868e-14    NaN           1.117488
min                    0.000000  9.200000e-01    NaN           0.000000
25%                    0.000000  9.200000e-01    NaN           0.000000
50%                    1.000000  9.200000e-01    NaN           1.000000
75%                    1.000000  9.200000e-01    NaN           2.000000
max                    1.000000  9.200000e-01    NaN           3.000000

       Thalassemia      Diabetes Smoking_History Physical_Activity_Level  \
count        10000  10000.000000           10000                   10000
unique           3           NaN               3                       3
top         Normal           NaN         Current                     Low
freq          3379           NaN            3366                    3391
mean           NaN      0.501200             NaN                     NaN
std            NaN      0.500024             NaN                     NaN
min            NaN      0.000000             NaN                     NaN
25%            NaN      0.000000             NaN                     NaN
50%            NaN      1.000000             NaN                     NaN
75%            NaN      1.000000             NaN                     NaN
max            NaN      1.000000             NaN                     NaN

        Family_History           BMI  Heart_Failure
count     10000.000000  1.000000e+04   10000.000000
unique             NaN           NaN            NaN
top                NaN           NaN            NaN
freq               NaN           NaN            NaN
mean          0.506300  3.692000e+01       0.503600
```

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

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

# Oldpeak (related to heart stress test)
plt.figure(figsize=(6, 4))
sns.histplot(df['Oldpeak'], kde=True, color='salmon')
plt.title('Distribution of Oldpeak')
plt.xlabel('Oldpeak')
plt.ylabel('Count')
plt.grid(True)
plt.show()

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

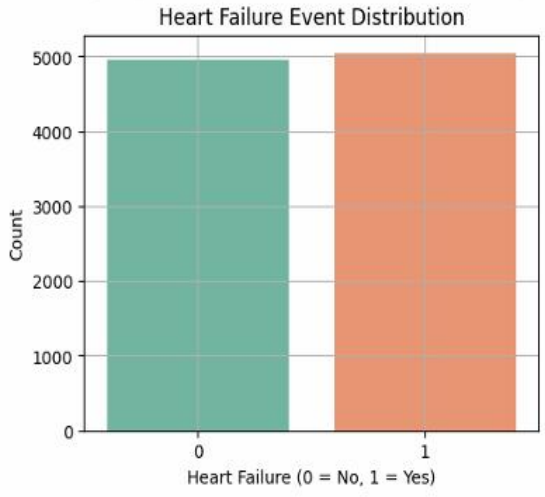







```
if 'Ejection_Fraction' in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df['Ejection_Fraction'], kde=True, color='salmon')
    plt.title('Distribution of Ejection Fraction')
    plt.xlabel('Ejection Fraction')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()


if 'Serum_Creatinine' in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df['Serum_Creatinine'], kde=True, color='lightgreen')
    plt.title('Distribution of Serum Creatinine')
    plt.xlabel('Serum Creatinine')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()


# Countplot for Heart_Failure column
plt.figure(figsize=(5, 4))
sns.countplot(x='Heart_Failure', data=df, palette='Set2')
plt.title('Heart Failure Event Distribution')
plt.xlabel('Heart Failure (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```

```
<ipython-input-14-ba92aa7a2066>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='Heart_Failure', data=df, palette='Set2')
```




```
df = df.drop(columns=["Heart_Failure"])


df = pd.get_dummies(df, drop_first=True)


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)


scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)
```

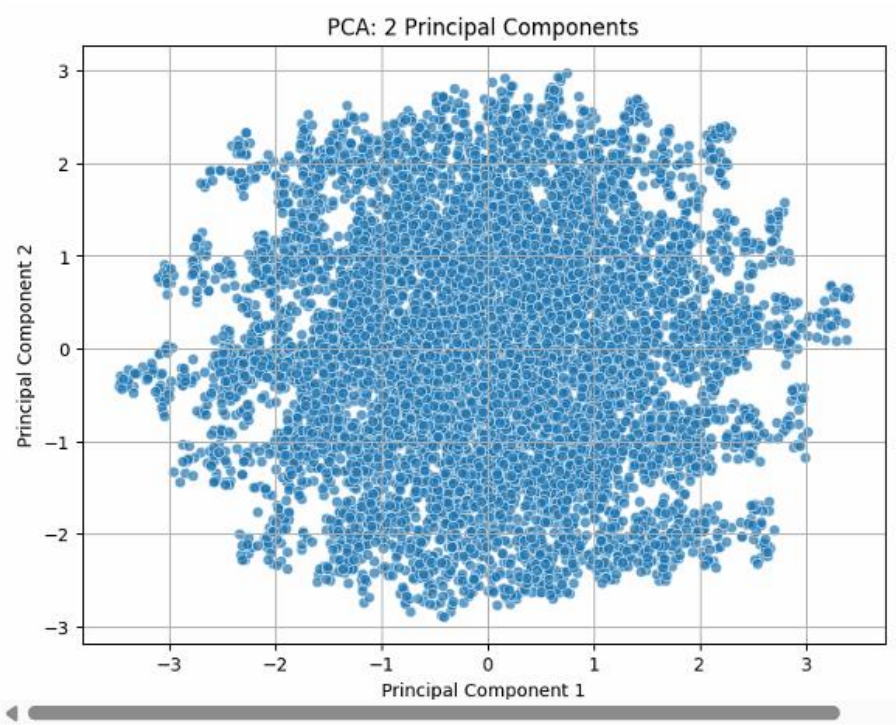
```
# Confirm shape and scaling
print("Shape of standardized data:", X_scaled.shape)
print("Feature means (should be ~0):", np.round(X_scaled.mean(axis=0), 2))
print("Feature std devs (should be ~1):", np.round(X_scaled.std(axis=0), 2))
```

```
Shape of standardized data: (10000, 25)
Feature means (should be ~0): [ 0.  0.  0.  0.  0.  0.  0. -0.  0.  0.  0.  0.  0. -0. -0.  0. -0. -0.
  0.  0.  0.  0. -0. -0.  0.]
Feature std devs (should be ~1): [1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. ]
```

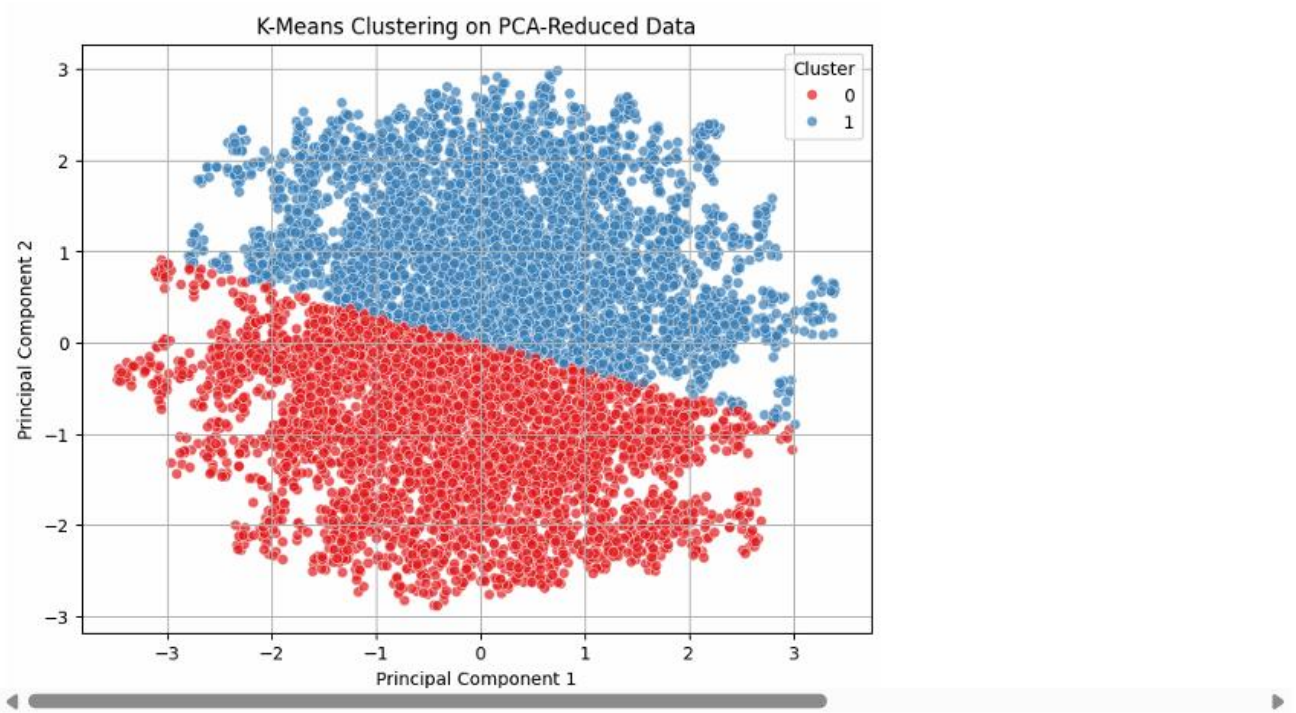
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
print("Explained variance ratio:", pca.explained_variance_ratio_)
```

```
Explained variance ratio: [0.06755063 0.06666515]
```

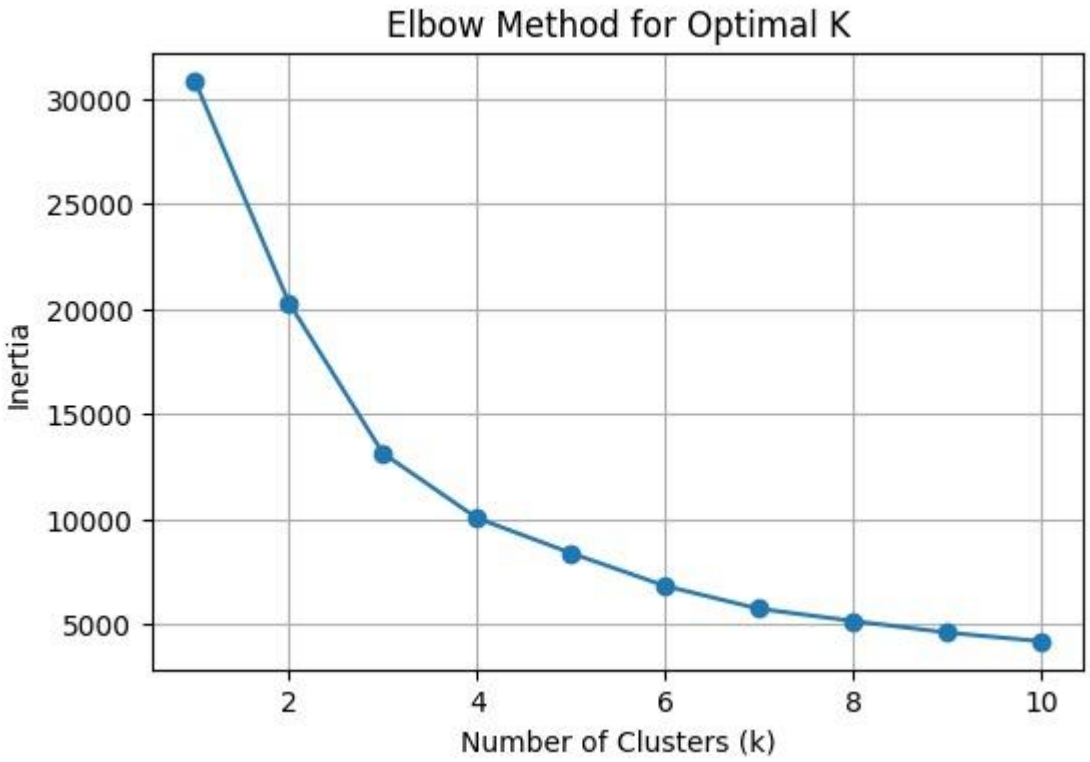
```
# Scatter plot of PCA components
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], alpha=0.7)
plt.title("PCA: 2 Principal Components")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
plt.show()
```

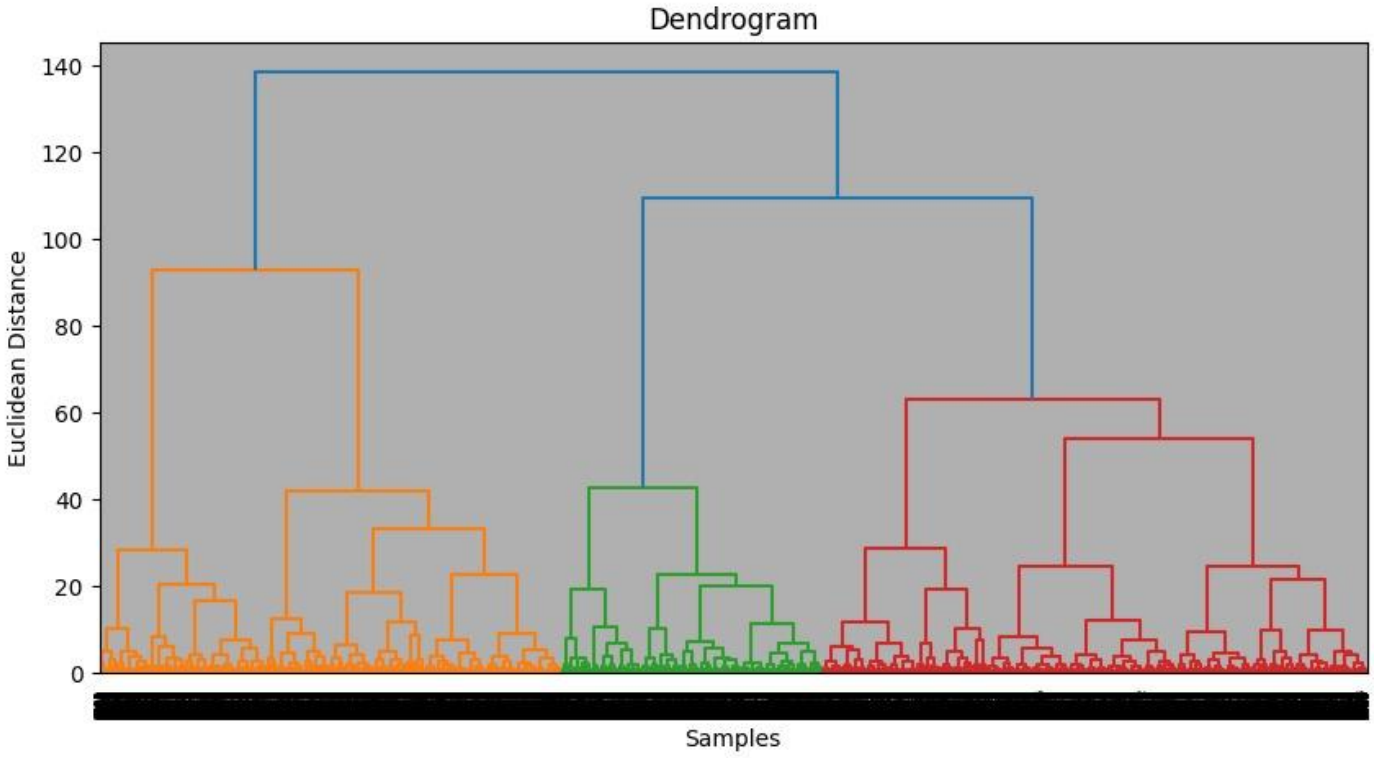



```
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_pca)

# Add cluster labels to the PCA data
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=clusters, palette='Set1', alpha=0.7)
plt.title("K-Means Clustering on PCA-Reduced Data")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



```python
from sklearn.decomposition import PCA
import pandas as pd

# Assuming X_scaled is your standardized data
pca = PCA(n_components=2)
pca_components = pca.fit_transform(X_scaled)

# Create DataFrame from PCA results
pca_df = pd.DataFrame(data=pca_components, columns=['PC1', 'PC2'])

# (Optional) See how much variance each PC explains
print("Explained variance ratio:", pca.explained_variance_ratio_)
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(pca_df)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(6, 4))
plt.plot(k_range, inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()
```


Elbow Method for Optimal K
Inertia
30000
25000
20000
15000
10000
5000
2
4
6
8
10
Number of Clusters (k)


Dendrogram
Euclidean Distance
140
120
100
80
60
40
20
0
Samples

INTERPRETATION:

- **Explain what it means to reduce a dataset to 2 principal components in simple terms.**

Reducing a dataset to two principal components through PCA means taking a high-dimensional set of features—like age, cholesterol, blood pressure, and other clinical variables—and projecting them onto just two new axes that capture the most important patterns or variations in the data. It's like distilling all the information down to the essence of what makes patients different from each other, so we can visualize it on a simple 2D plot. This doesn't mean the other features are irrelevant, but these two components carry the most variance and structure in the dataset, making it easier to spot patterns or groupings that might not be obvious otherwise.

- **Discuss how well the clusters are separated in your scatter plot.**

Looking at the scatter plot of your PCA-reduced data, the two clusters formed by K-Means are clearly and cleanly separated. The blue and red points are distributed in almost mirror-like halves, suggesting that the algorithm has successfully grouped patients with similar clinical profiles. The visual separation is quite distinct, with very little overlap between the two clusters, which implies that there is a strong underlying structure in the data that naturally divides the patients into these two groups.

- **Reflect on whether these clusters might correspond to real-world patient groups or clinical similarities.**

These clusters could very well correspond to real-world patient categories. Even though K-Means wasn't given the `DEATH_EVENT` label during training, the patterns it identified might reflect different levels of health risk or disease severity. One cluster might predominantly consist of patients with poorer clinical indicators—such as low ejection fraction or high serum creatinine—suggesting a higher likelihood of heart failure or mortality, while the other could represent patients with more stable or favorable health markers. To confirm this, we would need to compare the clusters directly with the original outcome labels, but visually and statistically, the separation strongly hints at meaningful clinical groupings.