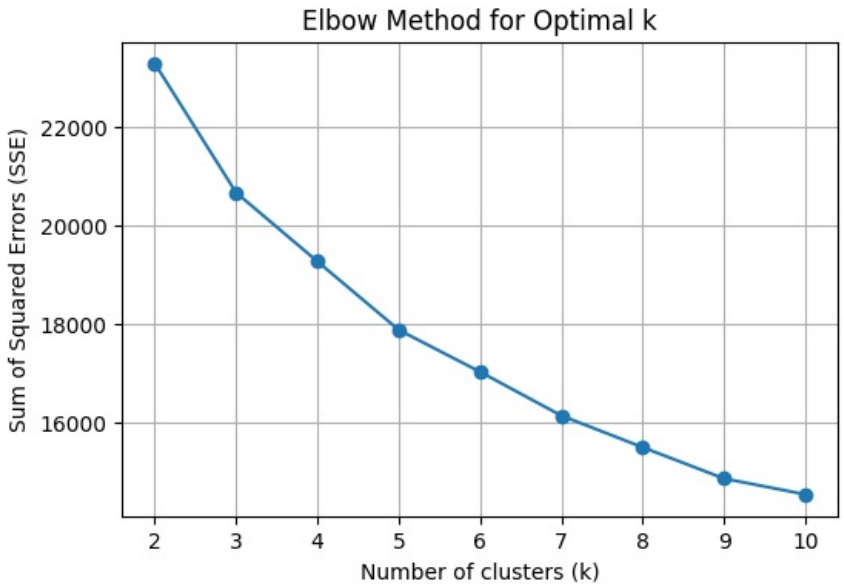
In [3]:

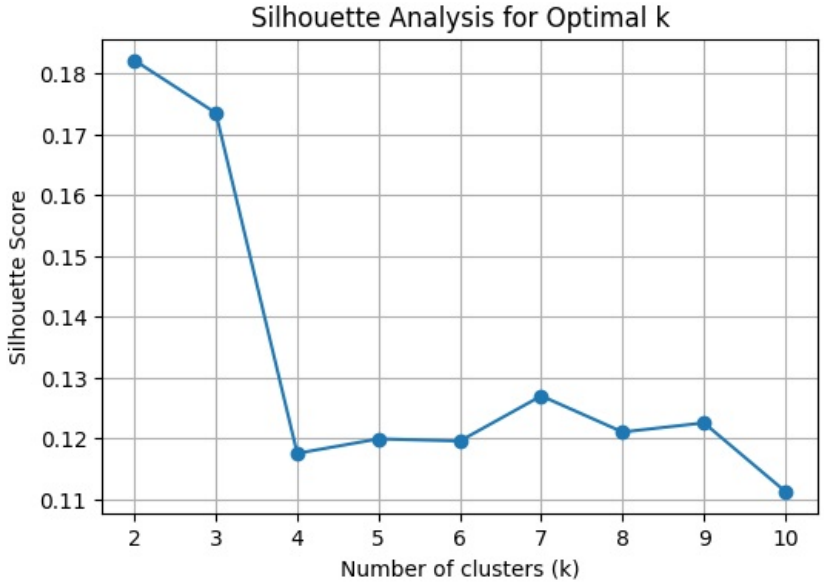
```
!pip install matplotlib seaborn scikit-learn
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = r'C:\Users\smita\Downloads\Healthcare-Diabetes.csv'
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())

# Step 1: Handle missing values
# Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())

# You can choose to drop rows with missing values or fill them.
# Here, we'll fill missing values with the mean of the respective column
df.fillna(df.mean(), inplace=True)

# Step 2: Normalize the features
# Select numeric columns for scaling (assuming all columns are relevant for clustering)
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Standardize the numeric columns using StandardScaler
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Display the first few rows after preprocessing
print("\nFirst few rows after preprocessing:")
print(df.head())

# The dataset is now preprocessed and ready for clustering
```

```
Requirement already satisfied: matplotlib in c:\users\smita\appdata\local\programs\python\python312\lib\site-pac
kages (3.9.2)
Requirement already satisfied: seaborn in c:\users\smita\appdata\local\programs\python\python312\lib\site-packag
es (0.13.2)
Requirement already satisfied: scikit-learn in c:\users\smita\appdata\local\programs\python\python312\lib\site-p
ackages (1.5.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\smita\appdata\local\programs\python\python312\lib\si
te-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\smita\appdata\local\programs\python\python312\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\smita\appdata\local\programs\python\python312\lib\s
ite-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\smita\appdata\local\programs\python\python312\lib\s
ite-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.23 in c:\users\smita\appdata\local\programs\python\python312\lib\site-pa
ckages (from matplotlib) (2.1.0)
Requirement already satisfied: packaging>=20.0 in c:\users\smita\appdata\local\programs\python\python312\lib\sit
e-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\smita\appdata\local\programs\python\python312\lib\site-pack
ages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\smita\appdata\local\programs\python\python312\lib\si
te-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\smita\appdata\local\programs\python\python312\li
b\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\users\smita\appdata\local\programs\python\python312\lib\site-pa
ckages (from seaborn) (2.2.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\smita\appdata\local\programs\python\python312\lib\site-p
ackages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\smita\appdata\local\programs\python\python312\lib\site-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\smita\appdata\local\programs\python\python312\li
b\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\smita\appdata\local\programs\python\python312\lib\site-p
ackages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\smita\appdata\local\programs\python\python312\lib\site
-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\smita\appdata\local\programs\python\python312\lib\site-packa
ges (from python-dateutil>=2.7->matplotlib) (1.16.0)
First few rows of the dataset:
   Id  Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0   1            6      148             72             35        0  33.6
1   2            1       85             66             29        0  26.6
2   3            8      183             64              0        0  23.3
3   4            1       89             66             23       94  28.1
4   5            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Missing values in each column:
Id                          0
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

First few rows after preprocessing:
         Id  Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin  \
0 -1.731425     0.679232  0.839738       0.149033       0.882845 -0.713633
1 -1.730174    -0.825341 -1.127124      -0.163012       0.509169 -0.713633
2 -1.728922     1.281062  1.932439      -0.267027      -1.296931 -0.713633
3 -1.727671    -0.825341 -1.002244      -0.163012       0.135494  0.123547
4 -1.726419    -1.126256  0.496317      -1.515209       0.882845  0.782604

        BMI  DiabetesPedigreeFunction       Age   Outcome
0  0.181135                  0.478509  1.432495  1.381146
1 -0.685773                 -0.369130 -0.181079 -0.724037
2 -1.094459                  0.616712 -0.096154  1.381146
3 -0.500007                 -0.934224 -1.030329 -0.724037
4  1.357654                  5.579704 -0.011229  1.381146
```

In [4]:
```
# Import necessary libraries
import pandas as pd
```

```python
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = r'C:\Users\smita\Downloads\Healthcare-Diabetes.csv'
df = pd.read_csv(file_path)

# Data Preprocessing
# Fill missing values with column means
df.fillna(df.mean(), inplace=True)

# Normalize numeric columns
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Step 1: Elbow Method to determine optimal k
sse = []  # List to hold the sum of squared errors (SSE) for each k
k_range = range(2, 11)  # We'll check k from 2 to 10

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df[numeric_cols])
    sse.append(kmeans.inertia_)  # Inertia represents SSE

# Plot the Elbow Method
plt.figure(figsize=(6, 4))
plt.plot(k_range, sse, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Sum of Squared Errors (SSE)')
plt.title('Elbow Method for Optimal k')
plt.grid(True)
plt.show()

# Step 2: Silhouette Analysis to evaluate k
silhouette_scores = []  # List to hold silhouette scores for each k

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(df[numeric_cols])
    silhouette_scores.append(silhouette_score(df[numeric_cols], cluster_labels))

# Plot Silhouette Scores
plt.figure(figsize=(6, 4))
plt.plot(k_range, silhouette_scores, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis for Optimal k')
plt.grid(True)
plt.show()

# Optimal k based on silhouette score
optimal_k = k_range[silhouette_scores.index(max(silhouette_scores))]
print(f"The optimal number of clusters based on silhouette analysis is: {optimal_k}")
```



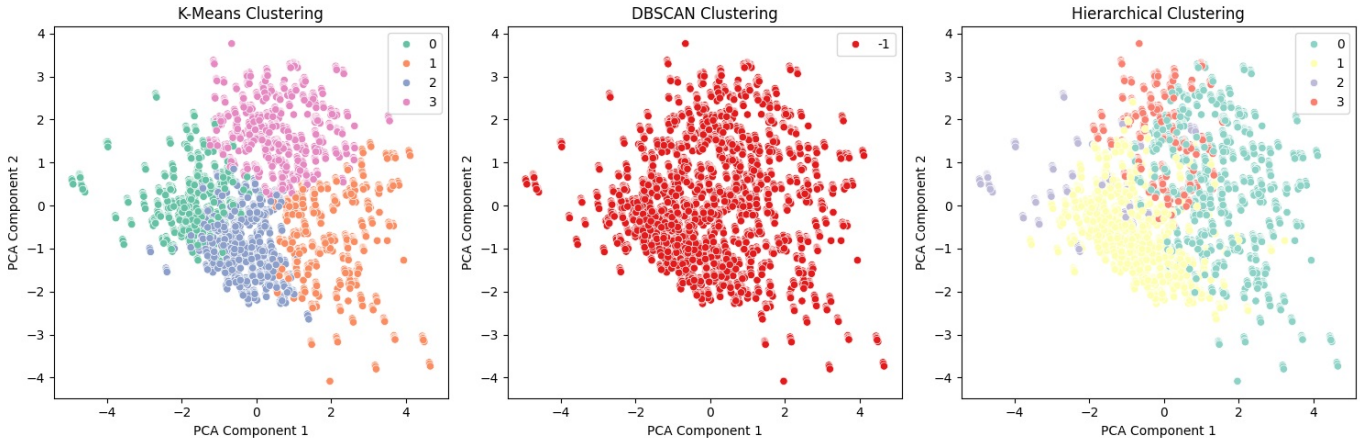



```
The optimal number of clusters based on silhouette analysis is: 2
```

In [5]:

```python
# Import necessary libraries for clustering
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA

# Load the dataset
file_path = r'C:\Users\smita\Downloads\Healthcare-Diabetes.csv'
df = pd.read_csv(file_path)

# Data Preprocessing (fill missing values and normalize features)
df.fillna(df.mean(), inplace=True)
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# For visualization purposes, let's reduce the dimensionality using PCA
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df[numeric_cols])

# Apply K-Means Clustering
optimal_k = 4  # Example: set based on previous analysis (you can change this)
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_labels = kmeans.fit_predict(df[numeric_cols])

# Apply DBSCAN Clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(df[numeric_cols])

# Apply Hierarchical Clustering
agglo_clustering = AgglomerativeClustering(n_clusters=optimal_k)
agglo_labels = agglo_clustering.fit_predict(df[numeric_cols])

# Add cluster labels to the original dataframe
df['KMeans_Cluster'] = kmeans_labels
df['DBSCAN_Cluster'] = dbscan_labels
df['Agglo_Cluster'] = agglo_labels

# Visualization of clusters using PCA-reduced 2D data
plt.figure(figsize=(15, 5))

# K-Means clusters
plt.subplot(1, 3, 1)
sns.scatterplot(x=df_pca[:, 0], y=df_pca[:, 1], hue=kmeans_labels, palette='Set2')
plt.title('K-Means Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')

# DBSCAN clusters
plt.subplot(1, 3, 2)
sns.scatterplot(x=df_pca[:, 0], y=df_pca[:, 1], hue=dbscan_labels, palette='Set1')
plt.title('DBSCAN Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
```

```
# Hierarchical Clustering
plt.subplot(1, 3, 3)
sns.scatterplot(x=df_pca[:, 0], y=df_pca[:, 1], hue=agglo_labels, palette='Set3')
plt.title('Hierarchical Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')

plt.tight_layout()
plt.show()

# Display the first few rows of the dataframe with the assigned clusters
print("Data with assigned clusters:")
print(df.head())
```



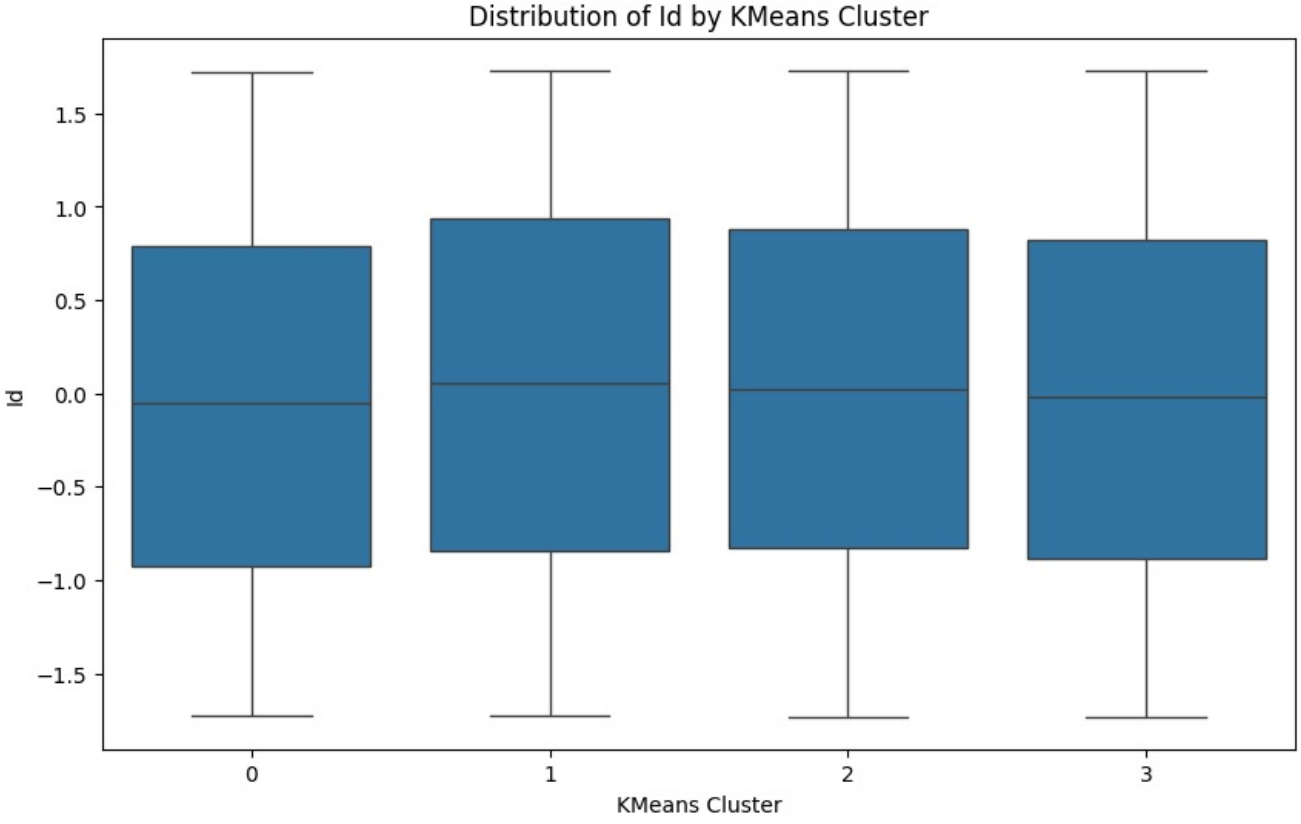

```
Data with assigned clusters:
         Id  Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin  \
0 -1.731425     0.679232  0.839738       0.149033       0.882845 -0.713633
1 -1.730174    -0.825341 -1.127124      -0.163012       0.509169 -0.713633
2 -1.728922     1.281062  1.932439      -0.267027      -1.296931 -0.713633
3 -1.727671    -0.825341 -1.002244      -0.163012       0.135494  0.123547
4 -1.726419    -1.126256  0.496317      -1.515209       0.882845  0.782604

        BMI  DiabetesPedigreeFunction       Age   Outcome  KMeans_Cluster  \
0  0.181135                  0.478509  1.432495  1.381146               3
1 -0.685773                 -0.369130 -0.181079 -0.724037               2
2 -1.094459                  0.616712 -0.096154  1.381146               3
3 -0.500007                 -0.934224 -1.030329 -0.724037               2
4  1.357654                  5.579704 -0.011229  1.381146               1

   DBSCAN_Cluster  Agglo_Cluster
0              -1              0
1              -1              1
2              -1              0
3              -1              1
4              -1              0
```

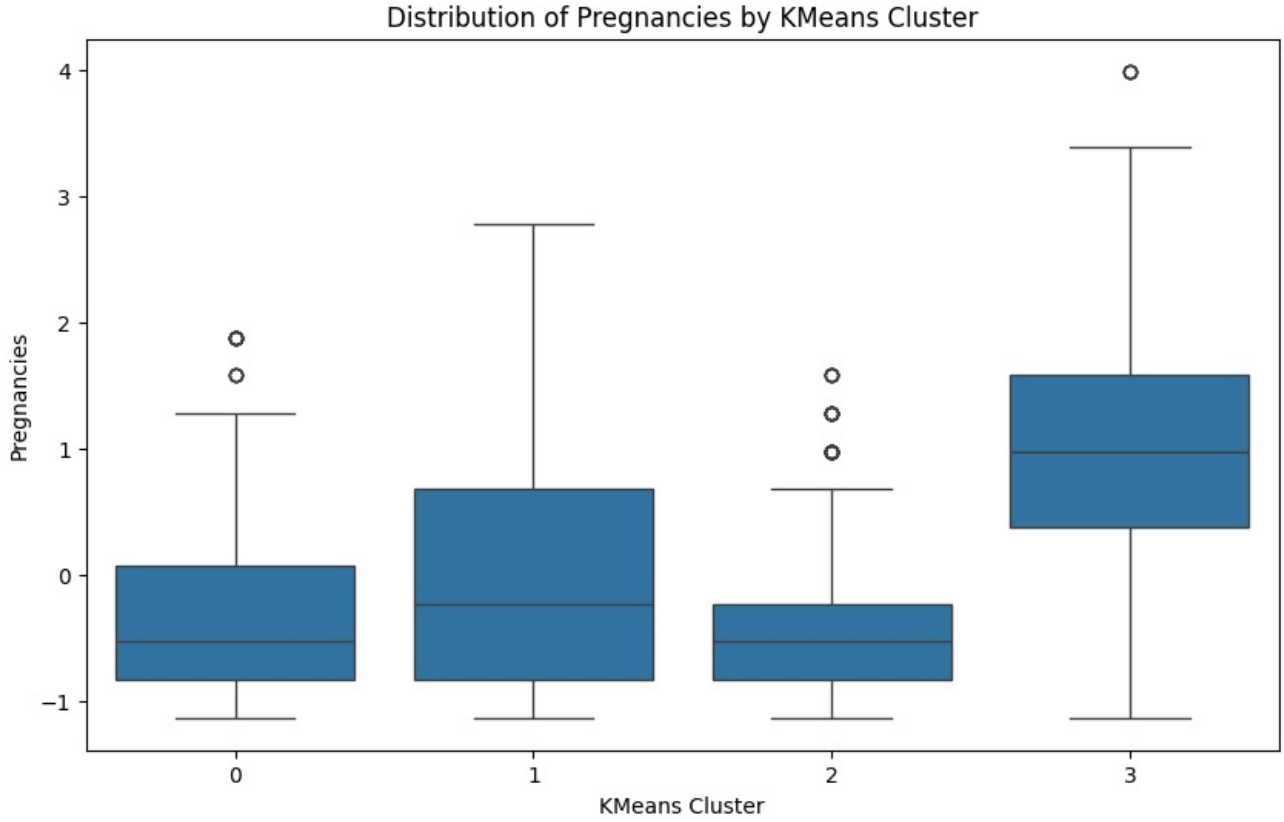
In [8]:

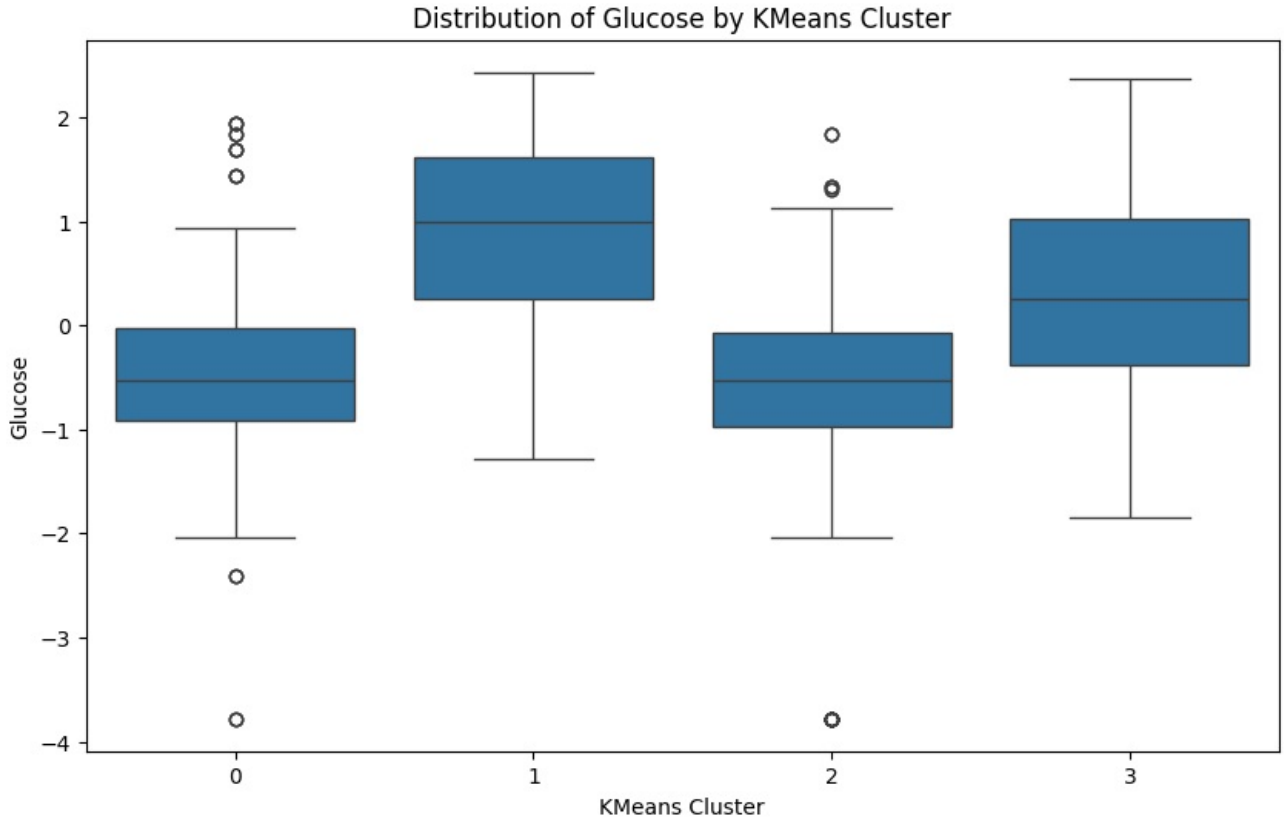
```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Load the dataset
file_path = r'C:\Users\smita\Downloads\Healthcare-Diabetes.csv'
df = pd.read_csv(file_path)

# Step 1: Data Preprocessing
# Handle missing values
df.fillna(df.mean(), inplace=True)

# Normalize features
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Step 2: Apply K-Means Clustering
optimal_k = 4  # Set based on previous analysis (Elbow/Silhouette)
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['KMeans_Cluster'] = kmeans.fit_predict(df[numeric_cols])

# Step 3: Analyze characteristics of each KMeans cluster
cluster_summary = df.groupby('KMeans_Cluster')[numeric_cols].mean()
print("K-Means Cluster Characteristics (Mean of Features):")
```

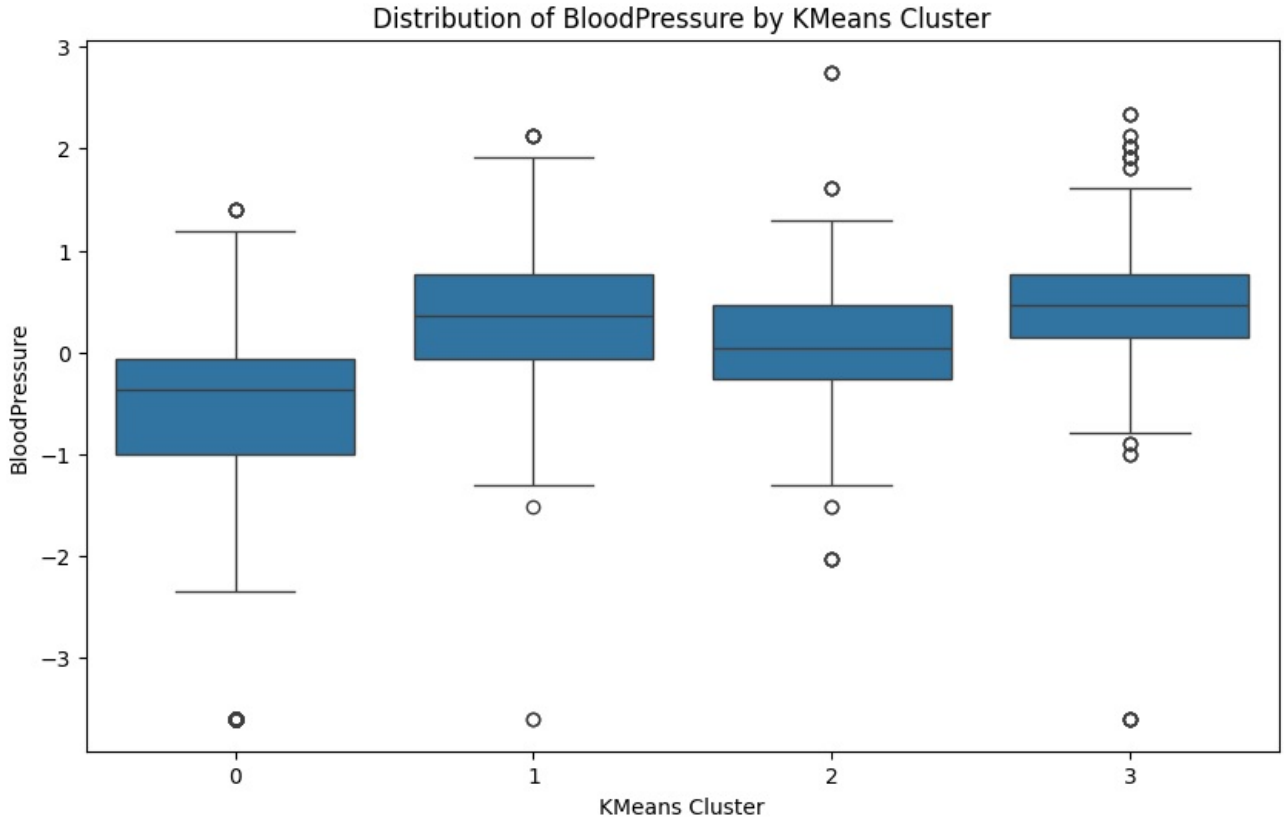
```
print(cluster_summary)

# Step 4: Visualize the distribution of key features for each cluster
for col in numeric_cols[:4]:  # Example with first four numeric columns
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='KMeans_Cluster', y=col, data=df)
    plt.title(f'Distribution of {col} by KMeans Cluster')
    plt.xlabel('KMeans Cluster')
    plt.ylabel(col)
    plt.show()

# Step 5: Pairplot for Multi-Feature Visualization
sns.pairplot(df, hue='KMeans_Cluster', vars=numeric_cols[:4])  # Visualizing first 4 numeric columns
plt.suptitle('Pairplot of Clusters', y=1.02)
plt.show()

# Step 6: Suggest Marketing Strategies Based on Cluster Characteristics
# Example strategies based on characteristics of clusters
cluster_strategies = {
    0: "Target high-value customers with loyalty programs and exclusive offers.",
    1: "Engage frequent buyers with personalized recommendations and bundle discounts.",
    2: "Focus on educating less active customers through email campaigns and product demos.",
    3: "Offer seasonal promotions to attract budget-conscious customers."
}

print("\nSuggested Marketing Strategies for Each Cluster:")
for cluster, strategy in cluster_strategies.items():
    print(f"Cluster {cluster}: {strategy}")
```

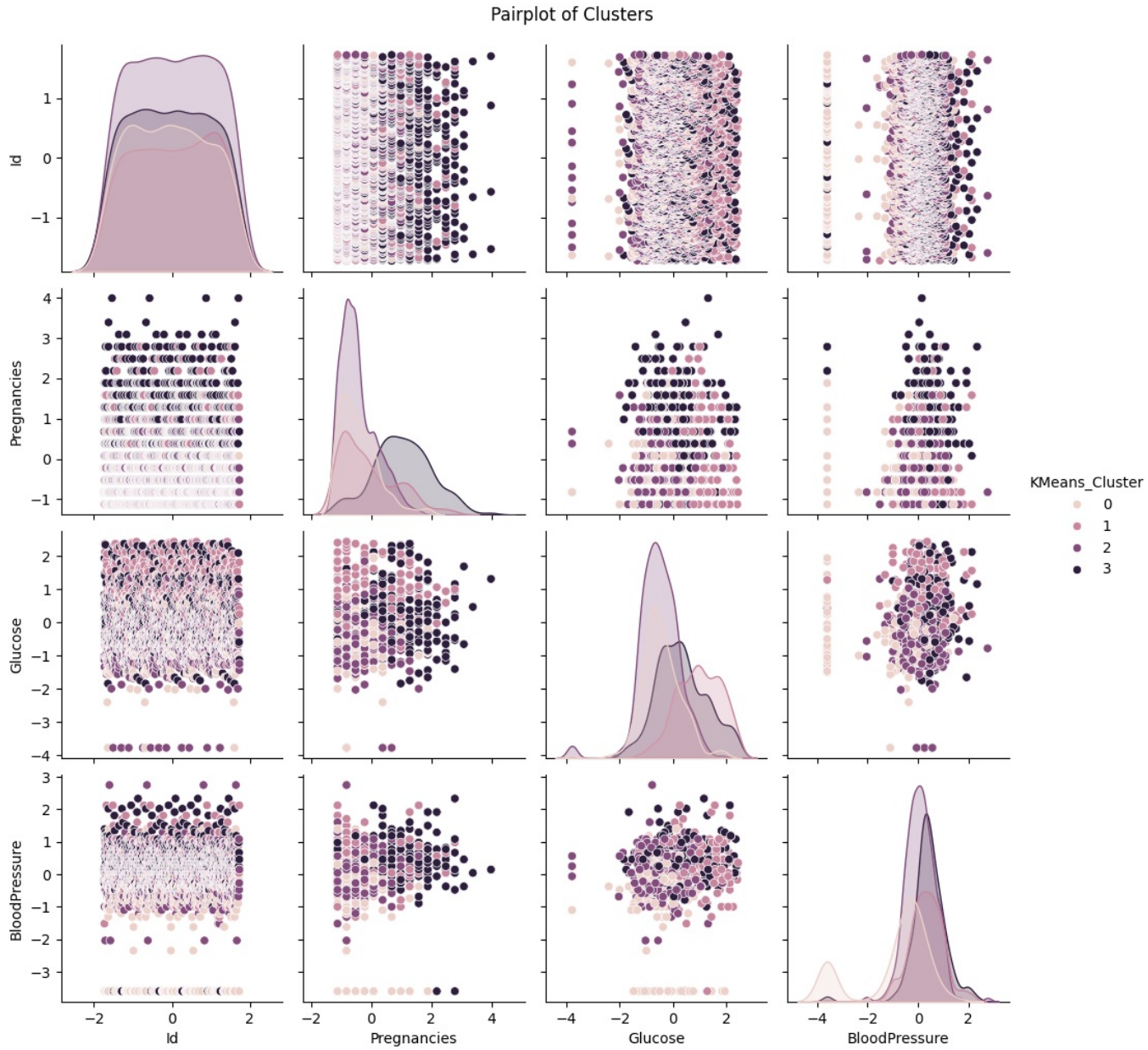
```
K-Means Cluster Characteristics (Mean of Features):
                      Id  Pregnancies   Glucose  BloodPressure  SkinThickness  \
KMeans_Cluster
0              -0.038766    -0.426118 -0.443287      -0.896897      -0.843521
1               0.032607    -0.093997  0.959929       0.312962       0.835052
2               0.027510    -0.442064 -0.518334       0.085761       0.471990
3              -0.028881     1.039807  0.324449       0.420210      -0.557600

                 Insulin       BMI  DiabetesPedigreeFunction       Age  \
KMeans_Cluster
0              -0.516545 -0.879036                 -0.330570 -0.535285
1               1.200379  0.657686                  0.620773  0.085672
2               0.015291  0.176651                 -0.076642 -0.520230
3              -0.518264  0.010173                 -0.099108  1.098080

                 Outcome
KMeans_Cluster
0              -0.437760
1               0.968592
2              -0.615148
3               0.442801
```




Distribution of Pregnancies by KMeans Cluster
Pregnancies
KMeans Cluster
4
3
2
1
0
−1
0
1
2
3


Distribution of Glucose by KMeans Cluster
Glucose
KMeans Cluster
2
1
0
−1
−2
−3
−4
0
1
2
3


Distribution of BloodPressure by KMeans Cluster
BloodPressure
KMeans Cluster
3
2
1
0
−1
−2
−3
0
1
2
3


Pairplot of Clusters
Id
Pregnancies
Glucose
BloodPressure
KMeans_Cluster
0
1
2
3

```
Suggested Marketing Strategies for Each Cluster:

Cluster 0: Target high-value customers with loyalty programs and exclusive offers.
Cluster 1: Engage frequent buyers with personalized recommendations and bundle discounts.
Cluster 2: Focus on educating less active customers through email campaigns and product demos.
Cluster 3: Offer seasonal promotions to attract budget-conscious customers.
```