In [1]:

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

In [2]:

```python
# Load the dataset

data = pd.read_csv("fitness_tracker_data.csv")

data
```

Out[2]:

	user_id	age	gender	steps_per_day	active_minutes	calories_burned	heart_rate_avg	sleep_hours	stress_level	goal_achieved
0	8270	46	Male	8304	254	2717	109	8.4	Moderate	False
1	1860	56	Male	9345	12	2810	113	7.3	High	False
2	6390	20	Male	17352	72	2270	73	7.9	Moderate	False
3	6191	49	Female	2597	58	2257	109	5.6	Low	True
4	6734	27	Female	4336	85	1635	74	7.6	High	False
...	...	...	...	...	...	...	...	...	...	...
995	6232	30	Male	5664	217	2840	96	8.6	Low	False
996	6797	21	Female	15050	229	1914	62	9.6	Low	True
997	5926	43	Female	2680	199	3308	68	6.9	Low	False
998	7016	42	Male	6357	293	2547	112	5.9	Moderate	True
999	4335	47	Male	3363	246	3373	77	8.7	Low	False

1000 rows × 10 columns

In [3]:

```python
# Select numerical columns for dimensionality reduction
numerical_cols = ["age", "steps_per_day", "active_minutes", "calories_burned", "heart_rate_avg", "sleep_hours"]
X = data[numerical_cols]
```

In [4]:

```python
# Standardize numerical data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

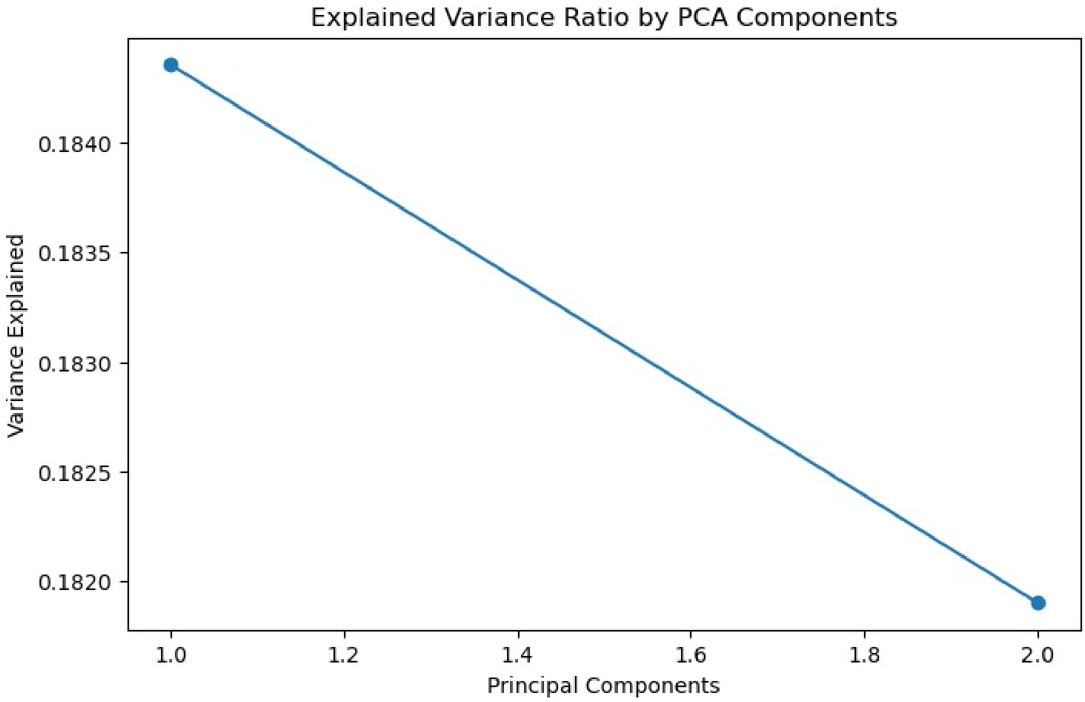
In [5]:

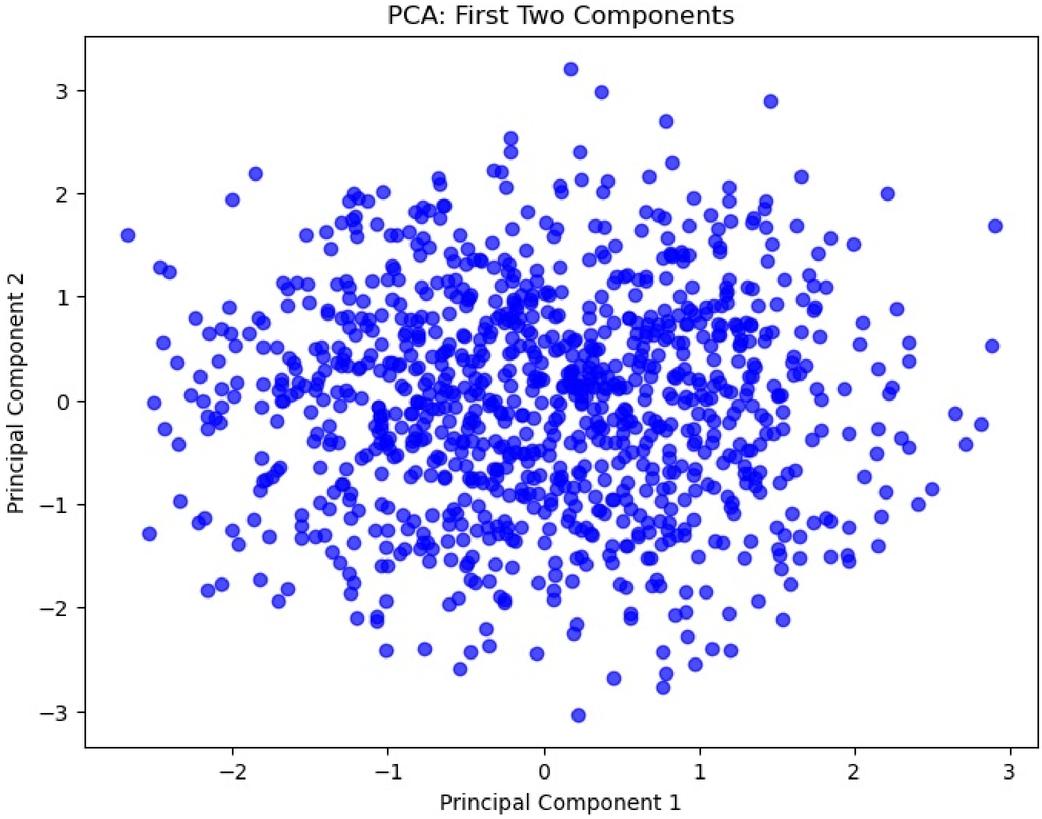
```python
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Apply PCA to reduce dimensions to 2
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Plot explained variance ratio
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_, marker='o')
plt.title("Explained Variance Ratio by PCA Components")
plt.xlabel("Principal Components")
plt.ylabel("Variance Explained")
plt.show()

# Scatter plot of first two principal components
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.7, c='blue')
plt.title("PCA: First Two Components")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```



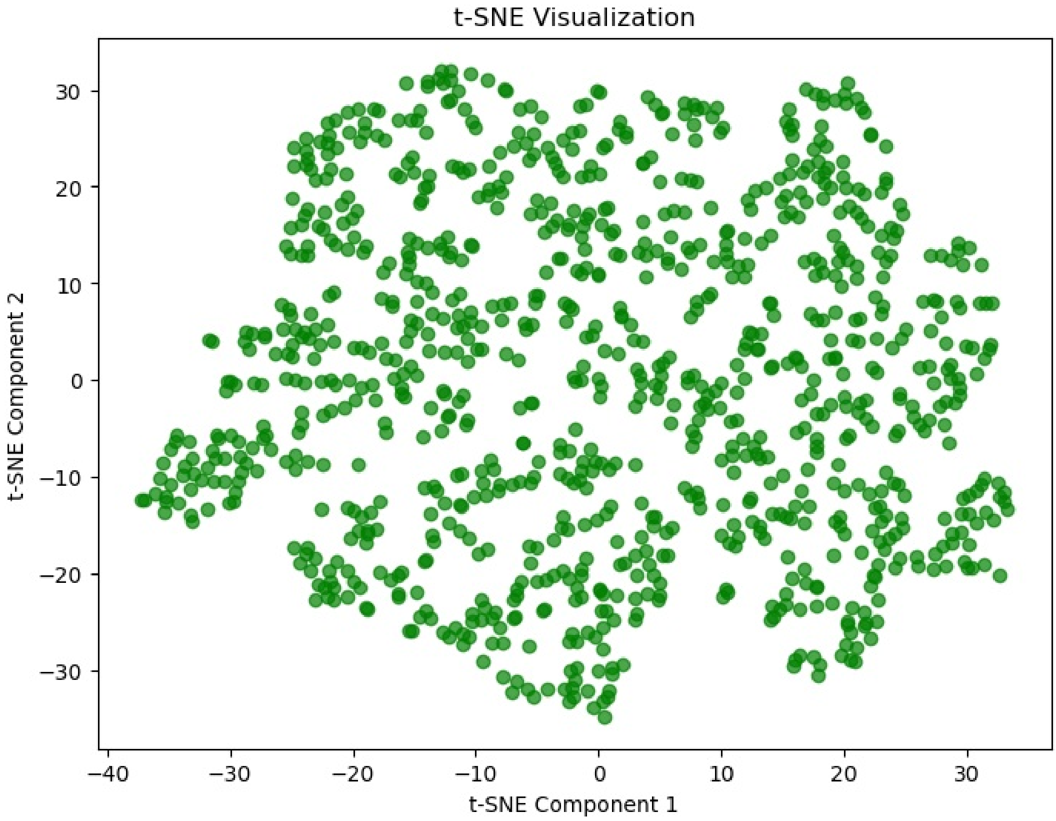


In [7]:

```
from sklearn.manifold import TSNE

# Apply t-SNE to reduce dimensions to 2
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

# Scatter plot of t-SNE
plt.figure(figsize=(8, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], alpha=0.7, c='green')
plt.title("t-SNE Visualization")
plt.xlabel("t-SNE Component 1")
plt.ylabel("t-SNE Component 2")
plt.show()
```



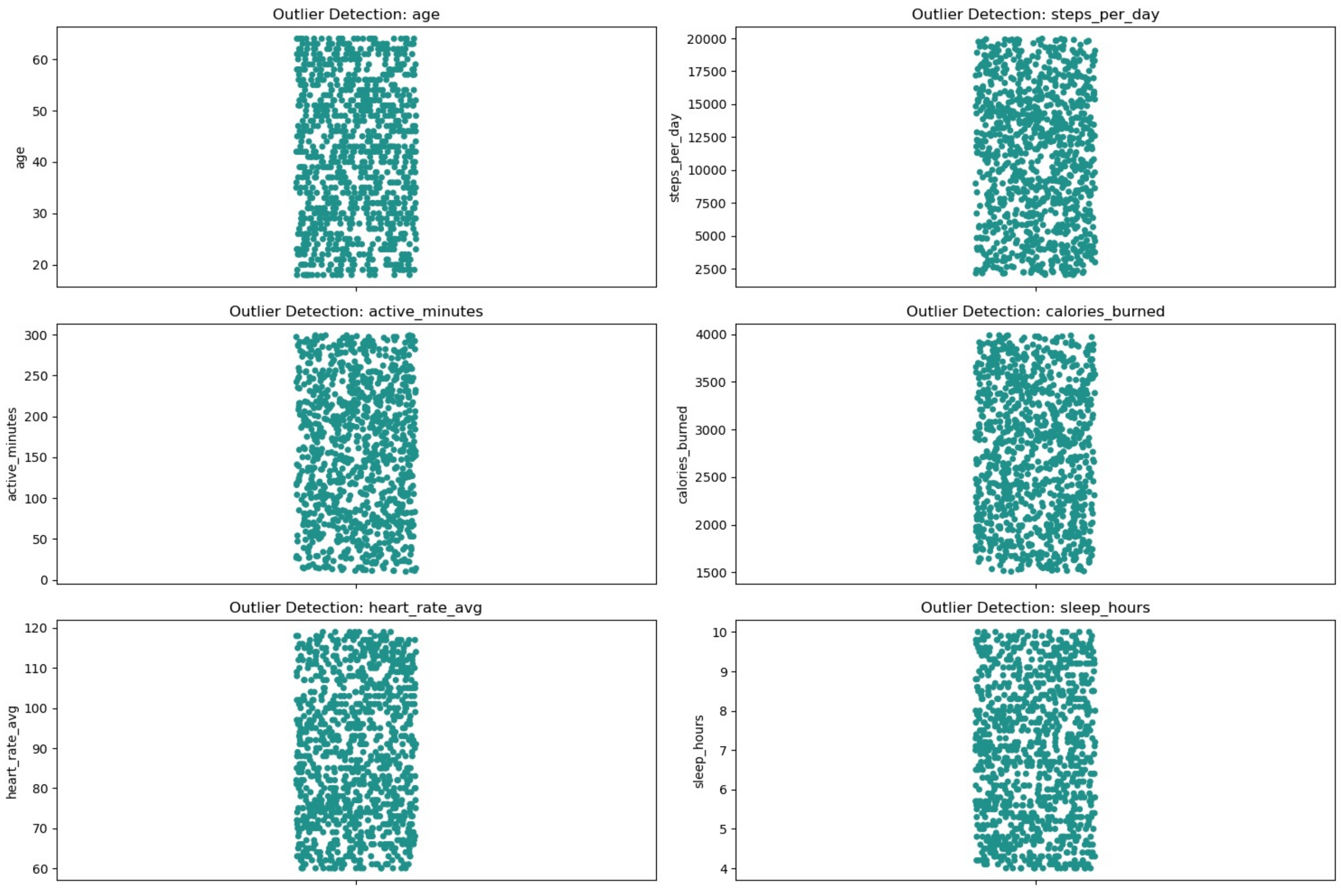
In [9]:

```
import seaborn as sns

# Create a dot plot for numerical features
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(3, 2, i + 1)
    sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
    plt.title(f"Outlier Detection: {col}")
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```

```
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_11276\97432049.py:7: FutureWarning: Passing `palette` without
assigning `hue` is deprecated.
  sns.stripplot(y=col, data=data, jitter=True, palette="viridis")
```



In [10]:

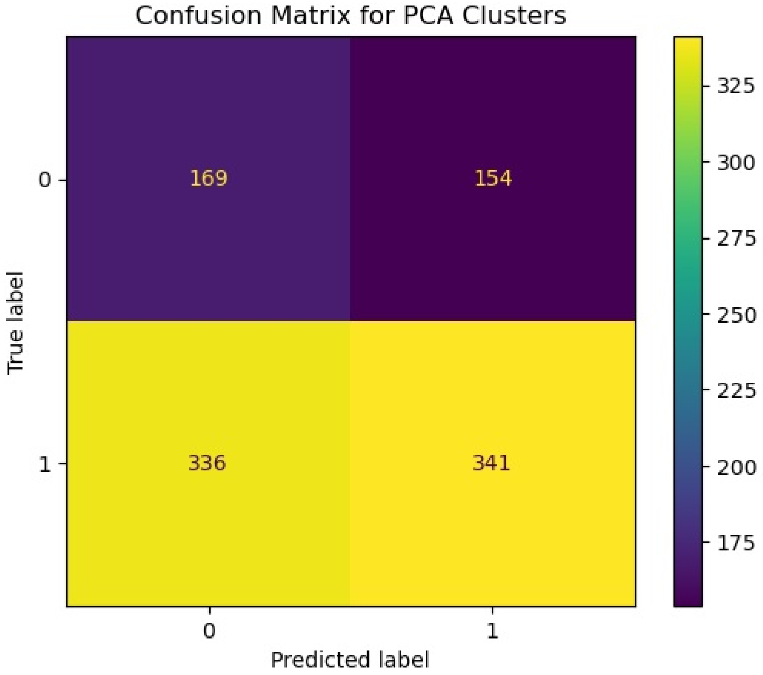
```python
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Apply K-Means on PCA-reduced data
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_pca)

# Compare clusters with the target variable (if available)
if "goal_achieved" in data.columns:
    y_true = data["goal_achieved"]
    cm = confusion_matrix(y_true, clusters)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])

    # Plot confusion matrix
    plt.figure(figsize=(8, 6))
    disp.plot(cmap="viridis", values_format="d")
    plt.title("Confusion Matrix for PCA Clusters")
    plt.show()
```

```
C:\Users\KEERTHANA.R\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default val
ue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warni
ng
  warnings.warn(
C:\Users\KEERTHANA.R\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it
by setting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
<Figure size 800x600 with 0 Axes>
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js