Principal Component Analysis(PCA)

In [12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

In [4]:
```
# Creating a sample dataset with features: Height, Weight, Age, and Gender
data = {
    'Height': [170, 165, 180, 175, 160, 172, 168, 177, 162, 158],
    'Weight': [65, 59, 75, 68, 55, 70, 62, 74, 58, 54],
    'Age':    [30, 25, 35, 28, 22, 32, 27, 33, 24, 21],
    'Gender': [1, 0, 1, 1, 0, 1, 0, 1, 0, 0]  # 1 = Male, 0 = Female
}
df = pd.DataFrame(data)
df
```

Out[4]:

	Height	Weight	Age	Gender
0	170	65	30	1
1	165	59	25	0
2	180	75	35	1
3	175	68	28	1
4	160	55	22	0
5	172	70	32	1
6	168	62	27	0
7	177	74	33	1
8	162	58	24	0
9	158	54	21	0

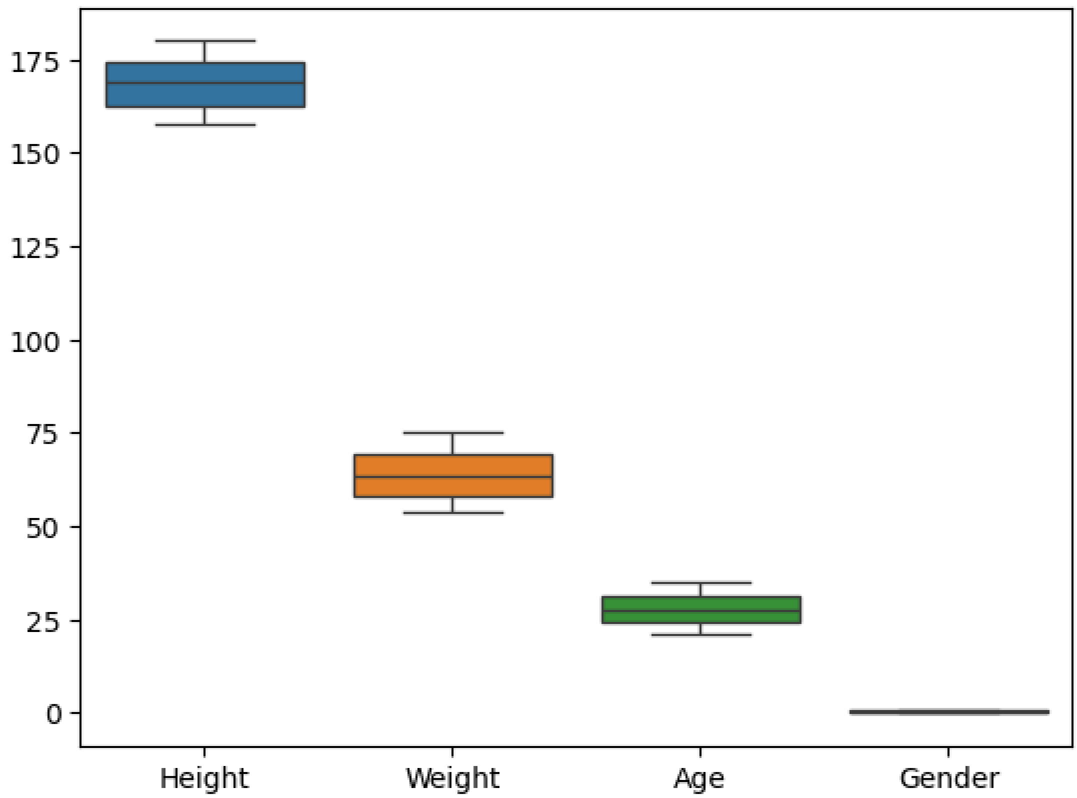
- # Missing value and outlier detection

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Height    0
Weight    0
Age       0
Gender    0
dtype: int64
```

In [6]:
```
sns.boxplot(data = df)
```

Out[6]: <Axes: >



Interpretation: The boxplot represents, there are **no outliers** in these features.

- # Standardizing and applying PCA

In [9]:
```
# Separate the features (X) and target (y)
X = df.drop('Gender', axis=1)
y = df['Gender']
X
```

Out[9]:

	Height	Weight	Age
0	170	65	30
1	165	59	25
2	180	75	35
3	175	68	28
4	160	55	22
5	172	70	32
6	168	62	27
7	177	74	33
8	162	58	24
9	158	54	21

In [11]:
```
# Using Standard Scaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[11]:
```
array([[ 0.18419807,  0.13867505,  0.50910379],
       [-0.52425605, -0.69337525, -0.59764358],
       [ 1.60110632,  1.52542554,  1.61585117],
       [ 0.8926522 ,  0.5547002 ,  0.06640484],
       [-1.23271018, -1.24807544, -1.26169201],
       [ 0.46757972,  0.83205029,  0.95180275],
       [-0.09918358, -0.2773501 , -0.15494463],
       [ 1.17603385,  1.38675049,  1.17315222],
       [-0.94932853, -0.83205029, -0.81899306],
       [-1.51609183, -1.38675049, -1.48304149]])
```

In [14]:
```
# Apply PCA to reduce dimensions to 2 components
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Split the dataset into training (70%) and testing (30%)
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random
# Fit logistic regression on the PCA-transformed data
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred
```

Out[14]:
```
array([0, 0, 1])
```

Interpretation: The ouput predicts 2 female (0) and 1 male(1)

• Model accuracy

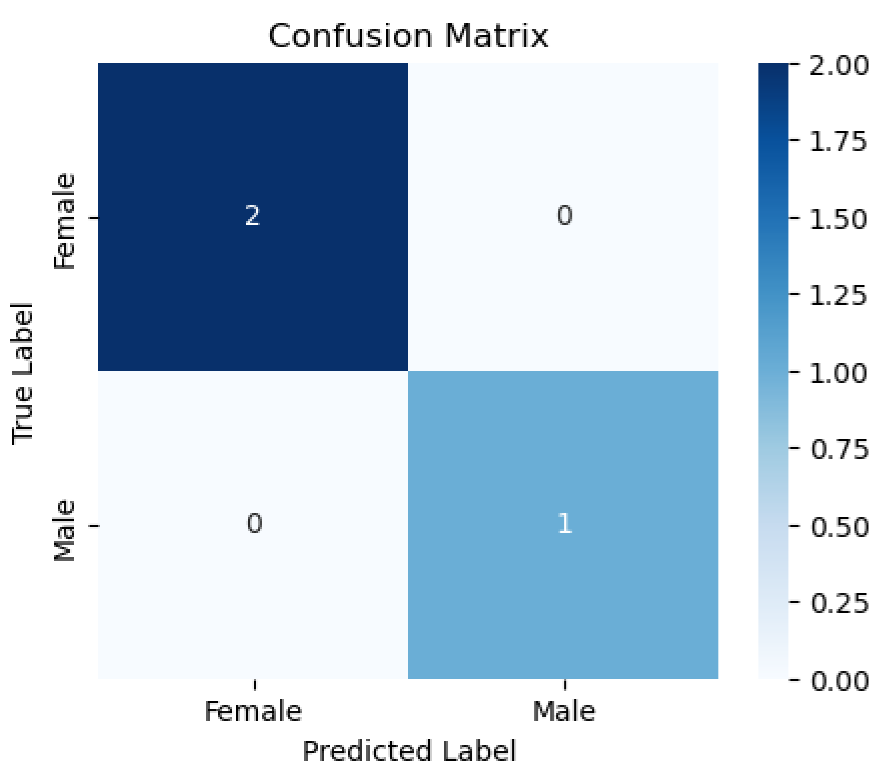
In [16]:
```
# Evaluate the model
score = model.score(X_test, y_test)
print(f"Model Accuracy: {score:.2f}")
```

```
Model Accuracy: 1.00
```

Interpretation: This model works properly and give 100% accuracy because of small dataset.

• Visualization

In [18]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Create confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Plot confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Female', 'Male'],
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Interpretation: The confusion matrix shows that the model achieved perfect classification performance for this dataset. It correctly predicted all cases without any misclassification. Specifically, the model identified both actual Female instances correctly (2 true positives) and correctly classified the single Male instance (1 true negative). There were no false positives or false negatives, meaning the model did not confuse one class for the other. As a result, the accuracy, precision, and recall for both classes are all 100%, indicating flawless prediction on the given data.

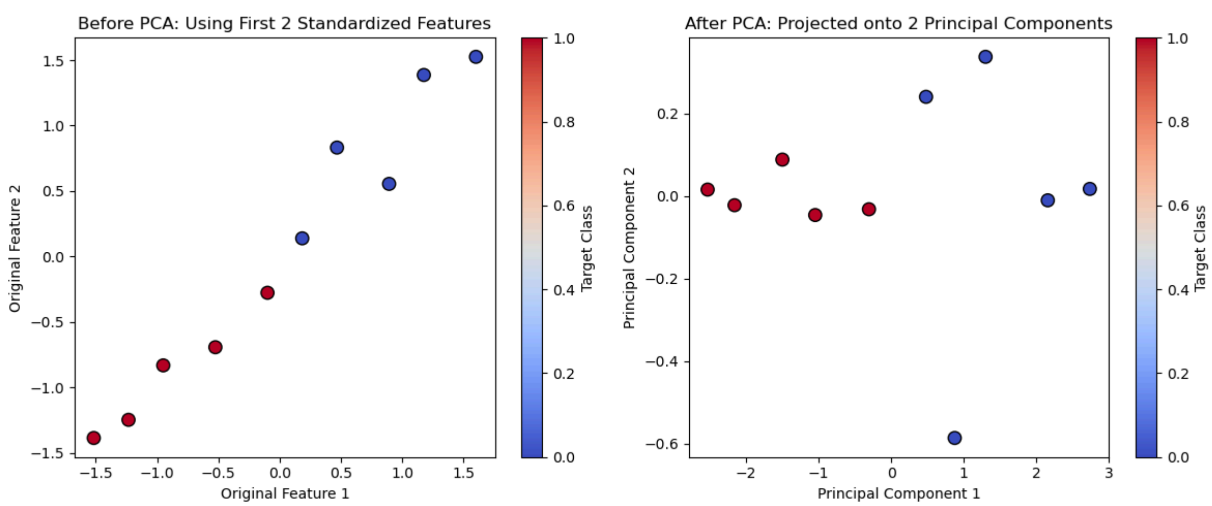
In [20]:
```
# Factorize gender labels for color mapping (0 = Female, 1 = Male)
y_numeric = pd.factorize(y)[0]

plt.figure(figsize=(12, 5))

# Plot original standardized features before PCA
plt.subplot(1, 2, 1)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_numeric, cmap='coolwarm', edgecolor
plt.xlabel('Original Feature 1')
plt.ylabel('Original Feature 2')
plt.title('Before PCA: Using First 2 Standardized Features')
plt.colorbar(label='Target Class')

# Plot PCA-reduced features
plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_numeric, cmap='coolwarm', edgecolor='k',
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('After PCA: Projected onto 2 Principal Components')
plt.colorbar(label='Target Class')

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



Interpretation: The left plot shows the dataset in its original standardized feature space, where the two features are highly correlated, as indicated by the strong diagonal trend. In this form, both axes contain overlapping information, resulting in redundancy. On the other hand, the right plot, represents the same data after applying Principal Component Analysis

(PCA), where the axes are rotated to form new variables called principal components that do simplify the data, remove correlations, and focus on the most important patterns without losing much information.

```
In [ ]:
```