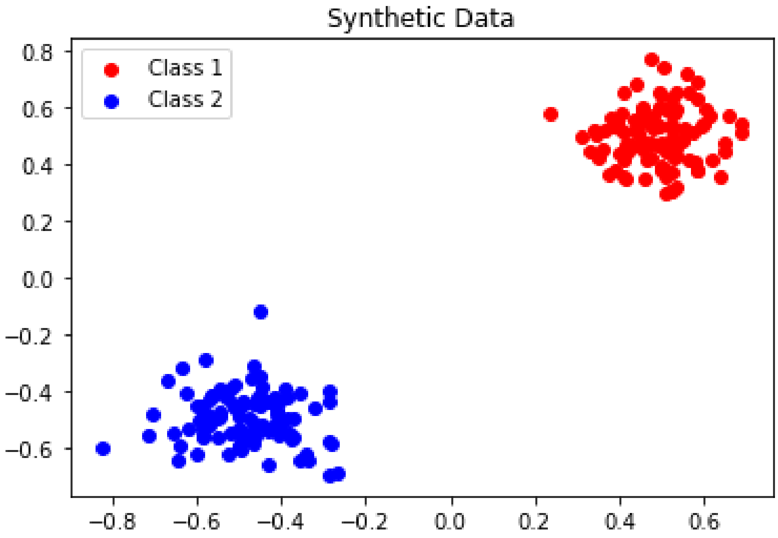
In [1]:

```
# 1st Part
```

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)

mean1 = [0.5, 0.5]
cov1 = [[0.01, 0], [0, 0.01]]
class1 = np.random.multivariate_normal(mean1, cov1, 100)

mean2 = [-0.5, -0.5]
cov2 = [[0.01, 0], [0, 0.01]]
class2 = np.random.multivariate_normal(mean2, cov2, 100)

X = np.vstack((class1, class2))
y = np.hstack((np.ones(100), -np.ones(100)))

plt.scatter(class1[:, 0], class1[:, 1], c='r', label='Class 1')
plt.scatter(class2[:, 0], class2[:, 1], c='b', label='Class 2')
plt.legend()
plt.title('Synthetic Data')
plt.show()
```



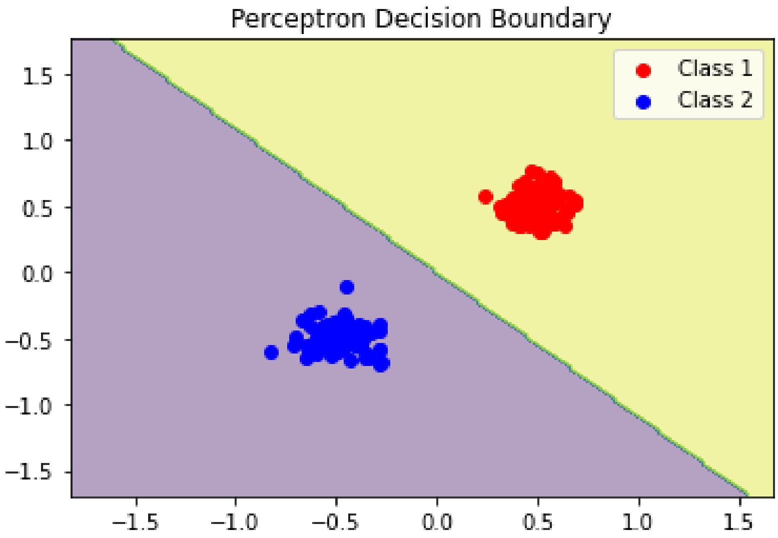
In [3]:

```
# 2nd Part
```

In [4]:

```
def perceptron(X, y, max_epochs=100, learning_rate=1):
    n_samples, n_features = X.shape
    weights = np.zeros(n_features)
    bias = 0

    for epoch in range(max_epochs):
        misclassified = 0
        for i in range(n_samples):
            prediction = np.dot(X[i], weights) + bias
            if y[i] * prediction <= 0:
                weights += learning_rate * y[i] * X[i]
                bias += learning_rate * y[i]
                misclassified += 1

        if misclassified == 0:
            print(f"Converged after {epoch + 1} epochs")
            return weights, bias

    print("Did not converge within the maximum number of epochs")
    return weights, bias


weights, bias = perceptron(X, y)


x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], weights) + bias)
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(class1[:, 0], class1[:, 1], c='r', label='Class 1')
plt.scatter(class2[:, 0], class2[:, 1], c='b', label='Class 2')
plt.legend()
plt.title('Perceptron Decision Boundary')
plt.show()
```

```
Converged after 2 epochs
```



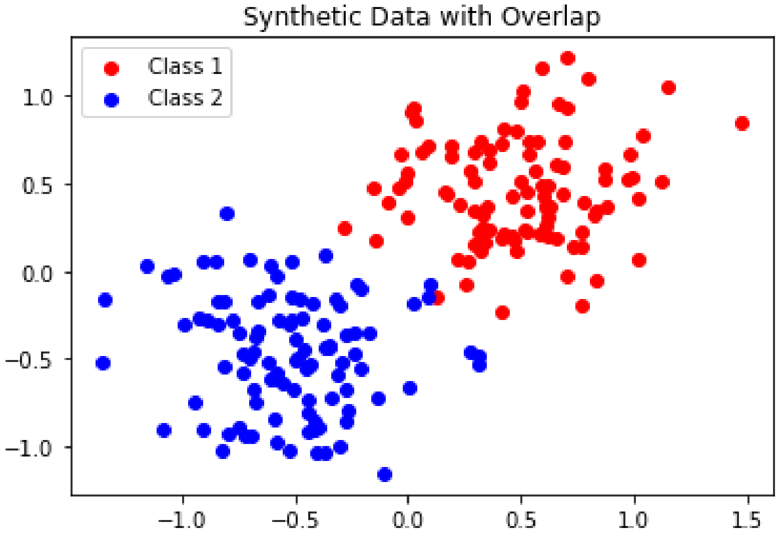
In [5]:

```
# 3rd Part
```

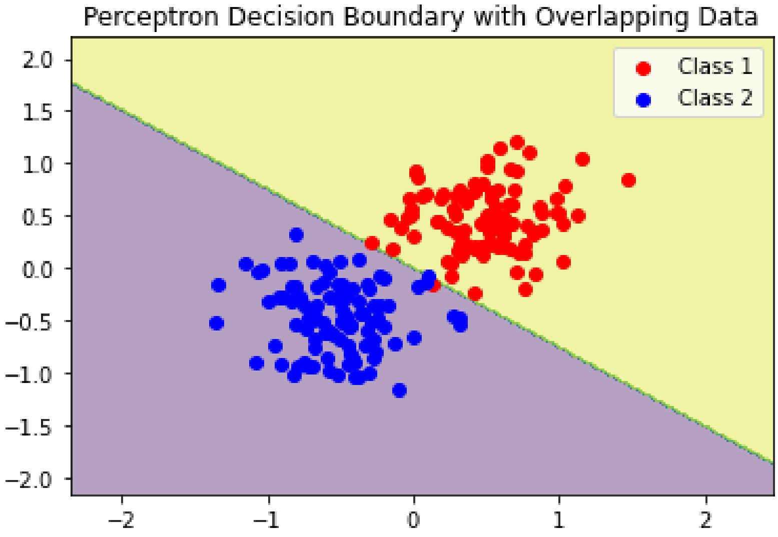
In [6]:

```
cov1 = [[0.1, 0], [0, 0.1]]
cov2 = [[0.1, 0], [0, 0.1]]

class1 = np.random.multivariate_normal(mean1, cov1, 100)
class2 = np.random.multivariate_normal(mean2, cov2, 100)

X = np.vstack((class1, class2))
y = np.hstack((np.ones(100), -np.ones(100)))

plt.scatter(class1[:, 0], class1[:, 1], c='r', label='Class 1')
plt.scatter(class2[:, 0], class2[:, 1], c='b', label='Class 2')
plt.legend()
```

```python
plt.title('Synthetic Data with Overlap')
plt.show()


weights, bias = perceptron(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], weights) + bias)
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(class1[:, 0], class1[:, 1], c='r', label='Class 1')
plt.scatter(class2[:, 0], class2[:, 1], c='b', label='Class 2')
plt.legend()
plt.title('Perceptron Decision Boundary with Overlapping Data')
plt.show()
```



```
Did not converge within the maximum number of epochs
```


Perceptron Decision Boundary with Overlapping Data
● Class 1
● Class 2