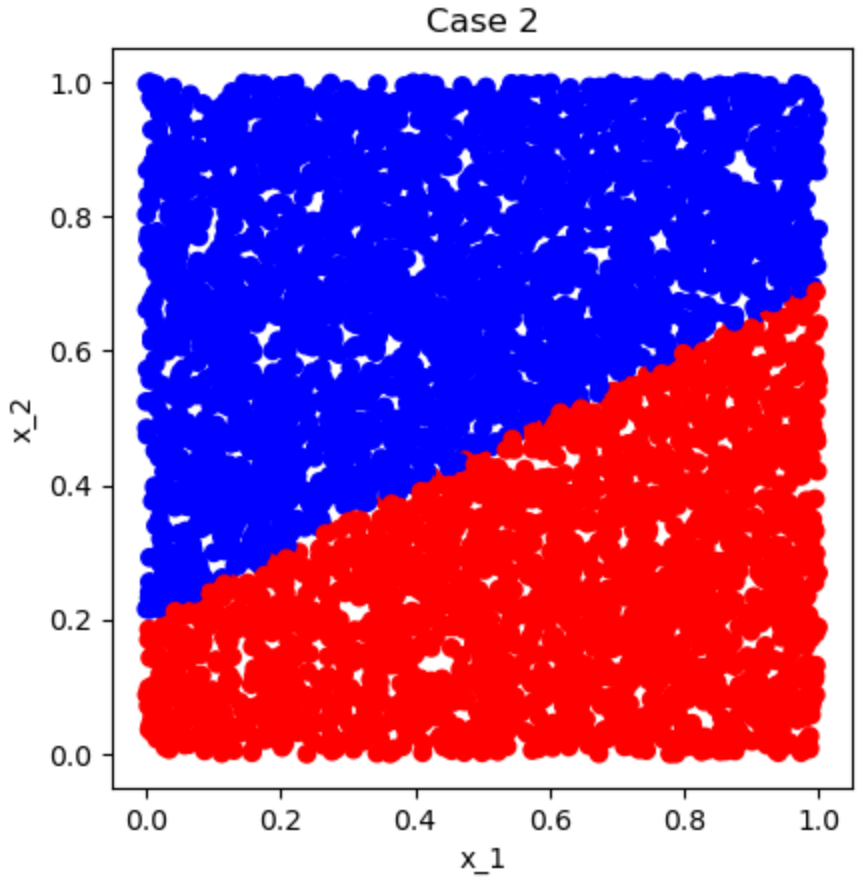
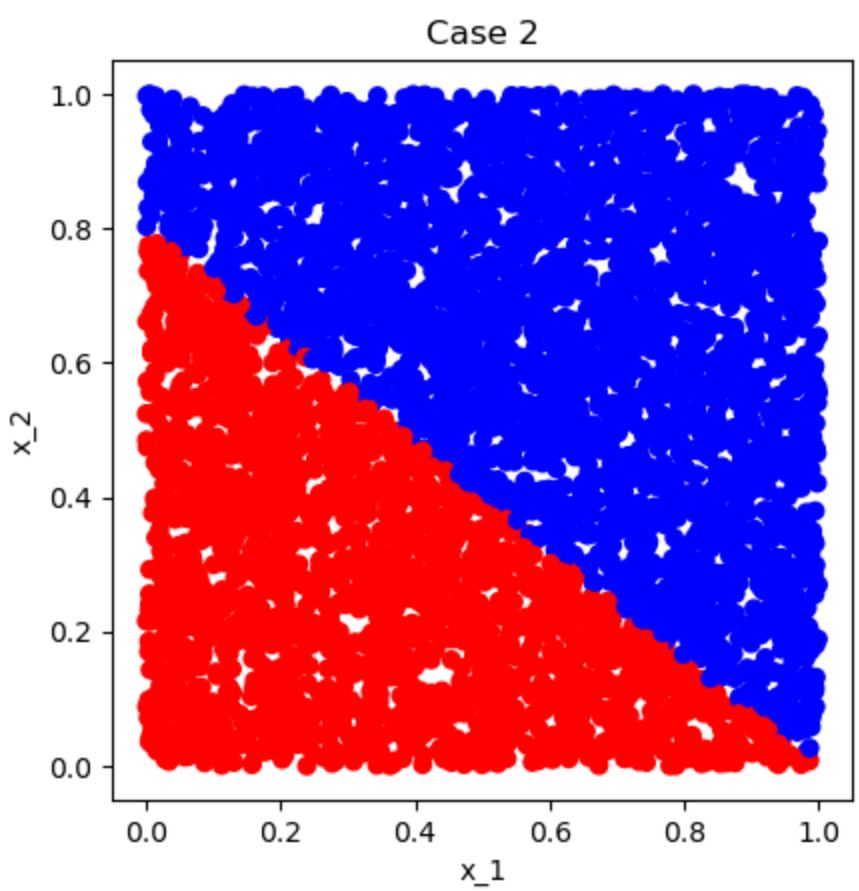
In [16]:

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

# number of features
p = 2

# number of examples
n = 5000

# generate matrix of n (random) examples of p features with a column of all ones
X0 = np.random.rand(n,p)
onevec = np.ones(shape = (n,1))
X = np.concatenate((X0,onevec),axis=1)

# Classifier weights
w = [[-1], [2], [-0.4]]

# Multiply feature matrix with weights  yhat = X*w
yhat = X@w

# Decide which class based on whether yhat is >< 0
# sign function returns +1 when yhat(i)>0 and -1 when yhat(i)<0
pred_label = np.sign(yhat);

plt.scatter(X[:,[0]],X[:,[1]], color = ['r' if i==-1 else 'b' for i in pred_label])

plt.xlabel("x_1")
plt.ylabel("x_2")
plt.title("Case 2")
plt.axis('square')
plt.show()


# Problem 1.e comment: The boundary is a line with y-intercept 0.2 and slope 0.5
```



In [18]:

```
# Classifier weights
w = [[1.6], [2], [-1.6]]

# Multiply feature matrix with weights  yhat = X*w
yhat = X@w

# Decide which class based on whether yhat is >< 0
# sign function returns +1 when yhat(i)>0 and -1 when yhat(i)<0
pred_label = np.sign(yhat);

plt.scatter(X[:,[0]],X[:,[1]], color = ['r' if i==-1 else 'b' for i in pred_label])

plt.xlabel("x_1")
plt.ylabel("x_2")
plt.title("Case 2")
plt.axis('square')
plt.show()

# Problem 1.f comment: The new weights adjusted the slope and y-intercept of the
# decision boundary. Now, the boundary has y-intercept 0.8 and slope -0.8
```



In [ ]: