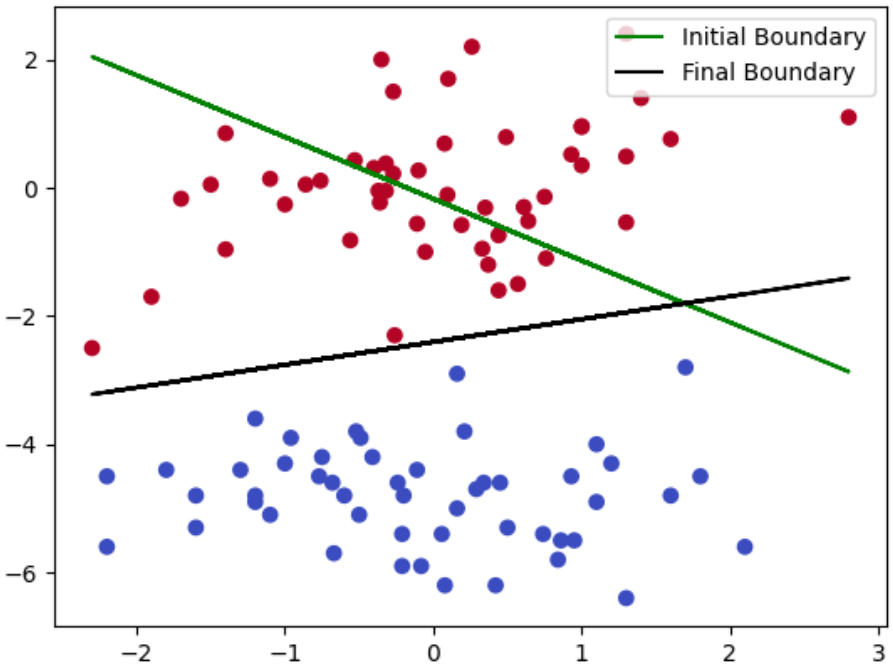
Exercise 2

$Z = 8.61 - 1.27x_1 + 3.58x_2$

Alpha = 0.1

Epochs = 100



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


def h_theta(t, x1, x2):
    """
    Calculate the hypothesis function

    :param t: array of thetas
    :param x1: float x1
    :param x2: float x2
    :return: hypothesis
    """
    z = t[0] + t[1] * x1 + t[2] * x2
    return 1 / (1 + np.exp(-z))


def logistic_regression(d, t, lr, ep):
    """
    Calculate the logistic regression

    :param d: array of data points
    :param t: array of thetas
    :param lr: learning rate
    :param ep: number of epochs
    :return: array of thetas
    """
    for _ in range(ep):
        for s in d:
            # Update thetas for each data point
            x1 = s[0]
            x2 = s[1]
            y = s[2]
            t[0] += lr * (y - h_theta(t, x1, x2))
            t[1] += lr * (y - h_theta(t, x1, x2)) * x1
            t[2] += lr * (y - h_theta(t, x1, x2)) * x2
    return t


def x_2(t, x):
    """
    Calculate the x_2 value

    :param t: array of thetas
    :param x: float x
    :return: float x_2
    """
    return (t[0] + t[1] * x) * (-1 / t[2])
```

```
if __name__ == '__main__':
    # Load data
    data = np.loadtxt('data.txt', delimiter=' ')
    # Plot data points
    plt.scatter(data[:, 0], data[:, 1], c=data[:, 2], cmap='coolwarm')
    # Set learning rate, number of epochs and initialize thetas
    alpha = 1e-1
    epochs = 100
    thetas = np.random.uniform(-.01, .01, (3, 1))
    # Plot initial boundary
    plt.plot(data[:, 0], [x_2(thetas, x) for x in data[:, 0]], 'g',
label='Initial Boundary')
    # Get updated thetas by calling logistic regression
    thetas = logistic_regression(data, thetas, alpha, epochs)
    # Plot final boundary
    plt.plot(data[:, 0], [x_2(thetas, x) for x in data[:, 0]], 'k',
label='Final Boundary')
    plt.legend()
    plt.show()
```