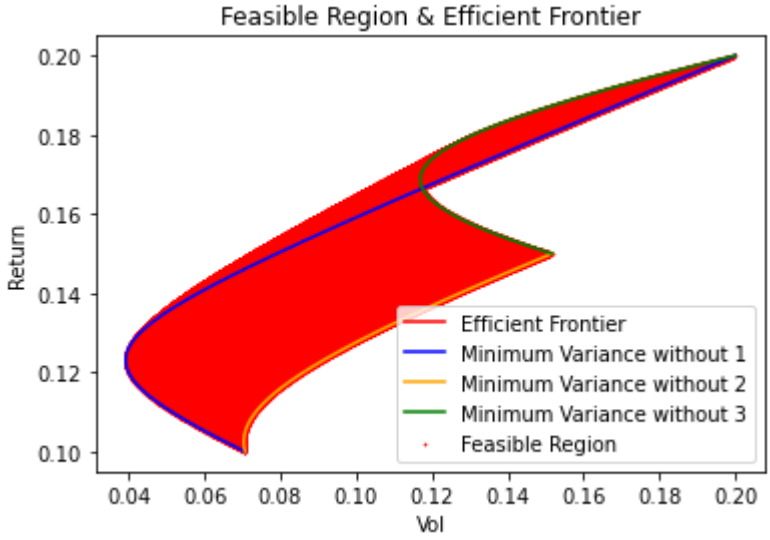
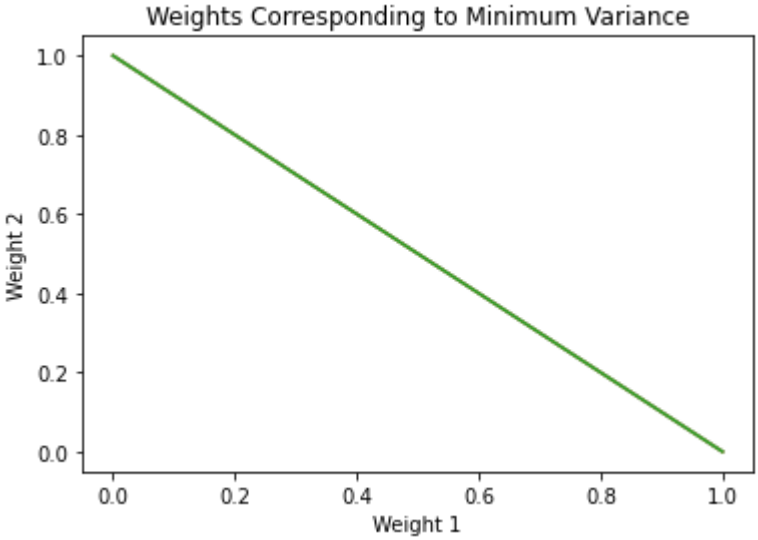
In [1]:

```
#Question No 2

import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize

mu = np.array([0.1, 0.2, 0.15])
covariance_matrix = np.array([[0.005, -0.010, 0.004],
                [-0.010, 0.040, -0.002],
                [0.004, -0.002, 0.023]])
dim = len(mu)
v = np.ones((1, dim))

def get_ret(w):
    return np.dot(w, mu)

def get_risk(w):
    return (np.matmul(np.matmul(w, covariance_matrix), np.transpose(w)))**0.5

def efficient_frontier(M):
    R = []
    for m in M:
        cons = (
            {'type': 'eq', 'fun': lambda w: np.sum(w)-1},
            {'type': 'eq', 'fun': lambda w: get_ret(w)-m}
        )
        bnds = ((0, 1), (0, 1), (0, 1))
        res = minimize(get_risk, np.array([0.2, 0.3, 0.5]), method='SLSQP', boun
ds=bnds, constraints=cons)
        R.append(res.fun)
    return R

def minimum_variance(M, i):
    R = []
    W = []
    for m in M:
        cons = (
            {'type': 'eq', 'fun': lambda w: w[i-1]},
            {'type': 'eq', 'fun': lambda w: np.sum(w)-1},
            {'type': 'eq', 'fun': lambda w: get_ret(w)-m}
        )
        bnds = ((0, 1), (0, 1), (0, 1))
        res = minimize(get_risk, np.array([0, 0, 1]), method='SLSQP', bounds=bnd
s, constraints=cons)
        R.append(res.fun)
        if i == 1:
            W.append([res.x[1], res.x[2]])
        if i == 2:
            W.append([res.x[0], res.x[2]])
        if i == 3:
            W.append([res.x[0], res.x[1]])
    return R, W

M = np.linspace(0.1, 0.2, 1000)
R = efficient_frontier(M)

M_1 = np.linspace(0.1, 0.2, 1000)
R_1, W_1 = minimum_variance(M_1, 3)
W_1 = np.transpose(W_1)
```

```
M_2 = np.linspace(0.1, 0.15, 1000)
R_2, W_2 = minimum_variance(M_2, 2)
W_2 = np.transpose(W_2)

M_3 = np.linspace(0.15, 0.2, 1000)
R_3, W_3 = minimum_variance(M_3, 1)
W_3 = np.transpose(W_3)

x_1 = 0
Y = []
X = []
while x_1 <= 1:
    x_2 = 0
    while x_2 <= 1-x_1:
        w3 = 1 - x_1 - x_2
        w = np.array([x_1, x_2, w3])
        m = np.dot(w, mu)
        r = (np.matmul(np.matmul(w, covariance_matrix), np.transpose(w)))**0.5
        Y.append(m)
        X.append(r)
        x_2 += 0.001
    x_1 += 0.001

plt.scatter(X, Y, s = 0.5, color='red', label='Feasible Region')
plt.plot(R, M, color='red', label='Efficient Frontier')
plt.plot(R_1, M_1, color='blue', label='Minimum Variance without 1')
plt.plot(R_2, M_2, color='orange', label='Minimum Variance without 2')
plt.plot(R_3, M_3, color='green', label='Minimum Variance without 3')
plt.title('Feasible Region & Efficient Frontier')
plt.xlabel('Vol')
plt.ylabel('Return')
plt.legend()
plt.show()

plt.plot(W_1[0], W_1[1])
plt.plot(W_2[0], W_2[1])
plt.plot(W_3[0], W_3[1])
plt.title('Weights Corresponding to Minimum Variance')
plt.xlabel('Weight 1')
plt.ylabel('Weight 2')
plt.show()
```

```
/srv/conda/envs/notebook/lib/python3.6/site-packages/IPython/core/py
labtools.py:132: UserWarning: Creating legend with loc="best" can be
slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```





In [ ]: