In [74]:
```
import pandas as pd
```

In [75]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [235]:
```
filename = "ex1data1.txt"
```

In [236]:
```
data = pd.read_csv(filename, names = ['Population', 'Profit'])
```

In [237]:
```
array = data.values
```

In [238]:
```
print(data.shape)
```

```
(97, 2)
```

In [239]:
```
print(data.describe())
```

```
       Population     Profit
count   97.000000  97.000000
mean     8.159800   5.839135
std      3.869884   5.510262
min      5.026900  -2.680700
25%      5.707700   1.986900
50%      6.589400   4.562300
75%      8.578100   7.046700
max     22.203000  24.147000
```

In [240]:
```
print(data.head(20))
```

```
    Population    Profit
0       6.1101  17.59200
1       5.5277   9.13020
2       8.5186  13.66200
3       7.0032  11.85400
4       5.8598   6.82330
5       8.3829  11.88600
6       7.4764   4.34830
7       8.5781  12.00000
8       6.4862   6.59870
9       5.0546   3.81660
10      5.7107   3.25220
11     14.1640  15.50500
12      5.7340   3.15510
13      8.4084   7.22580
14      5.6407   0.71618
15      5.3794   3.51290
16      6.3654   5.30480
17      5.1301   0.56077
18      6.4296   3.65180
19      7.0708   5.38930
```

In [241]:
```
print(data.head(20))
```

```
    Population   Profit
0       6.1101 17.59200
1       5.5277  9.13020
2       8.5186 13.66200
3       7.0032 11.85400
4       5.8598  6.82330
5       8.3829 11.88600
6       7.4764  4.34830
7       8.5781 12.00000
8       6.4862  6.59870
9       5.0546  3.81660
10      5.7107  3.25220
11     14.1640 15.50500
12      5.7340  3.15510
13      8.4084  7.22580
14      5.6407  0.71618
15      5.3794  3.51290
16      6.3654  5.30480
17      5.1301  0.56077
18      6.4296  3.65180
19      7.0708  5.38930
```

In [242]:
```
print(data.head(20))
```

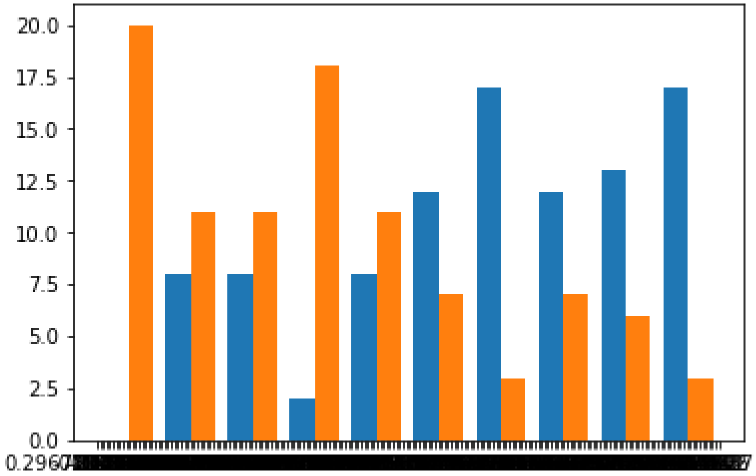
```
    Population    Profit
0       6.1101  17.59200
1       5.5277   9.13020
2       8.5186  13.66200
3       7.0032  11.85400
4       5.8598   6.82330
5       8.3829  11.88600
6       7.4764   4.34830
7       8.5781  12.00000
8       6.4862   6.59870
9       5.0546   3.81660
10      5.7107   3.25220
11     14.1640  15.50500
12      5.7340   3.15510
13      8.4084   7.22580
14      5.6407   0.71618
15      5.3794   3.51290
16      6.3654   5.30480
17      5.1301   0.56077
18      6.4296   3.65180
19      7.0708   5.38930
```

In [243]:
```
print(data.head(20))
```

```
    Population   Profit
0       6.1101 17.59200
1       5.5277  9.13020
2       8.5186 13.66200
3       7.0032 11.85400
4       5.8598  6.82330
5       8.3829 11.88600
6       7.4764  4.34830
7       8.5781 12.00000
8       6.4862  6.59870
9       5.0546  3.81660
10      5.7107  3.25220
11     14.1640 15.50500
12      5.7340  3.15510
13      8.4084  7.22580
14      5.6407  0.71618
15      5.3794  3.51290
16      6.3654  5.30480
17      5.1301  0.56077
18      6.4296  3.65180
19      7.0708  5.38930
```

In [244]:
```
plt.hist(data)
```

Out[244]:
```
([array([ 0.,  8.,  8.,  2.,  8., 12., 17., 12., 13., 17.]),
  array([20., 11., 11., 18., 11.,  7.,  3.,  7.,  6.,  3.])],
 array([  0. ,  19.3,  38.6,  57.9,  77.2,  96.5, 115.8, 135.1, 154.4,
        173.7, 193. ]),
 <a list of 2 Lists of Patches objects>)
```




In [245]:
```
data.head()
```

Out[245]:

| | Population | Profit |
|---|---|---|
| **0** | 6.1101 | 17.5920 |
| **1** | 5.5277 | 9.1302 |
| **2** | 8.5186 | 13.6620 |
| **3** | 7.0032 | 11.8540 |
| **4** | 5.8598 | 6.8233 |

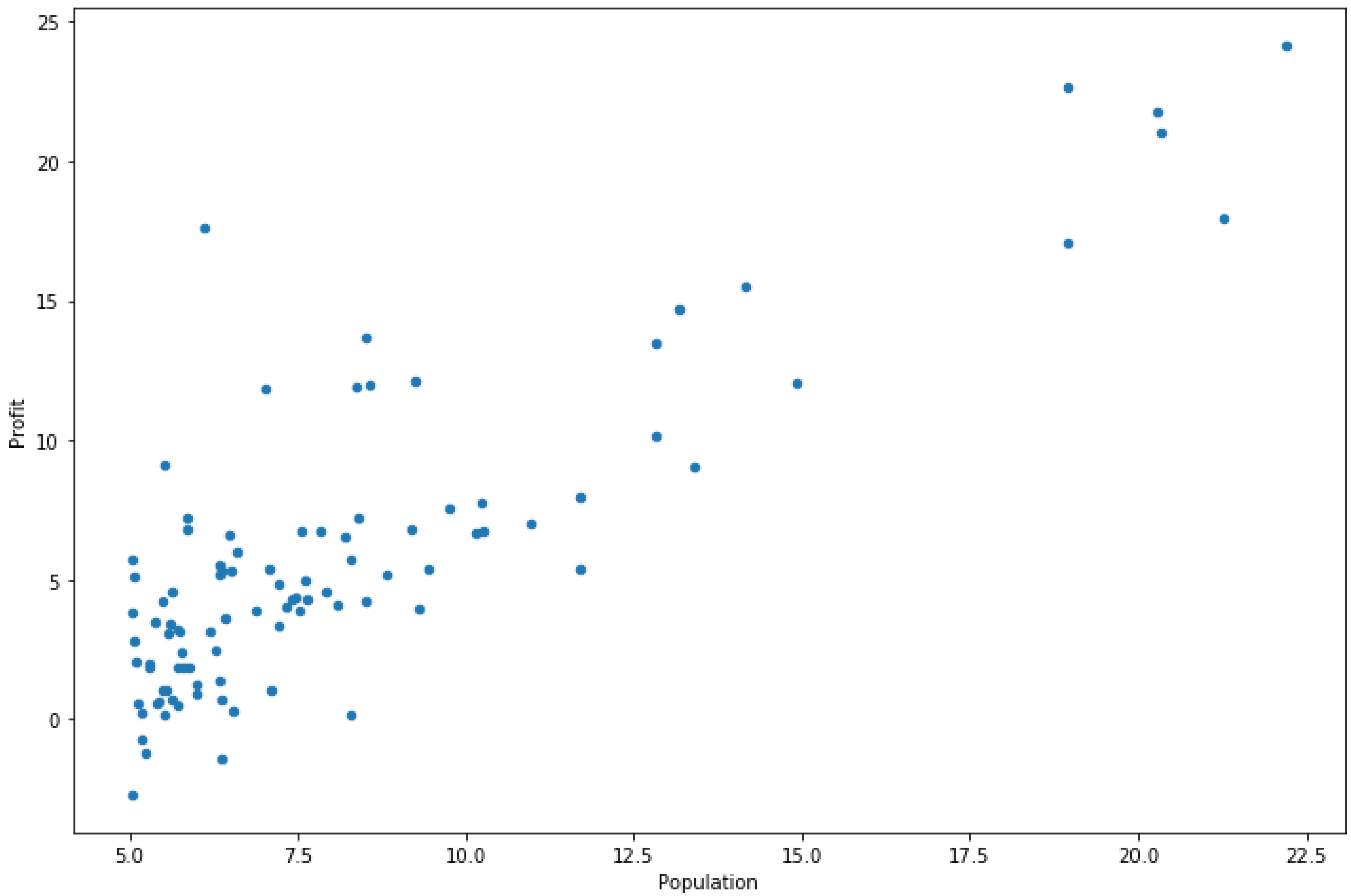
In [246]:
```
data.describe()
```

Out[246]:

| | Population | Profit |
|---|---|---|
| **count** | 97.000000 | 97.000000 |
| **mean** | 8.159800 | 5.839135 |
| **std** | 3.869884 | 5.510262 |
| **min** | 5.026900 | -2.680700 |
| **25%** | 5.707700 | 1.986900 |
| **50%** | 6.589400 | 4.562300 |
| **75%** | 8.578100 | 7.046700 |
| **max** | 22.203000 | 24.147000 |

In [247]:
```
data.plot(kind='scatter', x='Population', y='Profit', figsize=(12,8))
```

Out[247]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5922be9810>




In [248]:
```
def computeCost(X, y, theta):
    inner = np.power(((X * theta.T)-y), 2)
    return np.sum(inner) / (2*len(X))
```

In [249]:
```
data.insert(0, 'Ones', 1)
```

In [250]:
```
print(data.shape)
```

```
(97, 3)
```

In [251]:
```
print(data.head(20))
```

```
    Ones  Population    Profit
0      1      6.1101  17.59200
1      1      5.5277   9.13020
2      1      8.5186  13.66200
3      1      7.0032  11.85400
4      1      5.8598   6.82330
5      1      8.3829  11.88600
6      1      7.4764   4.34830
7      1      8.5781  12.00000
8      1      6.4862   6.59870
9      1      5.0546   3.81660
10     1      5.7107   3.25220
11     1     14.1640  15.50500
12     1      5.7340   3.15510
13     1      8.4084   7.22580
14     1      5.6407   0.71618
15     1      5.3794   3.51290
16     1      6.3654   5.30480
17     1      5.1301   0.56077
18     1      6.4296   3.65180
19     1      7.0708   5.38930
```

In [252]:
```
# set X (training data) and y (target variable)
cols = data.shape[1]
X = data.iloc[:,0:cols-1]
y = data.iloc[:,cols-1:cols]
```

In [253]:
```
X.head()
```

Out[253]:

| | Ones | Population |
|---|---|---|
| **0** | 1 | 6.1101 |
| **1** | 1 | 5.5277 |
| **2** | 1 | 8.5186 |
| **3** | 1 | 7.0032 |
| **4** | 1 | 5.8598 |

In [254]:
```
y.head()
```

Out[254]:

| | Profit |
|---|---|
| **0** | 17.5920 |
| **1** | 9.1302 |
| **2** | 13.6620 |
| **3** | 11.8540 |
| **4** | 6.8233 |

In [255]:
```
X = np.matrix(X.values)
y = np.matrix(y.values)
theta = np.matrix(np.array([0,0]))
```

In [256]:
```
theta
```

Out[256]:
```
matrix([[0, 0]])
```

In [257]:
```
X.shape, theta.shape, y.shape
```

Out[257]:
```
((97, 2), (1, 2), (97, 1))
```

In [258]:
```
computeCost(X, y, theta)
```

Out[258]:
```
32.072733877455676
```

In [259]:
```
def gradientDescent(X, y, theta, alpha, iters):
    temp = np.matrix(np.zeros(theta.shape))
    parameters = int(theta.ravel().shape[1])
    cost = np.zeros(iters)

    for i in range(iters):
        error = (X * theta.T) - y

        for j in range(parameters):
            term = np.multiply(error, X[:,j])
            temp[0,j] = theta[0,j] - ((alpha / len(X)) * np.sum(term))

        theta = temp
        cost[i] = computeCost(X, y, theta)

    return theta, cost
```

In [260]:
```
alpha = 0.01
iters = 1000
```

In [261]:
```
g, cost = gradientDescent(X, y, theta, alpha, iters)
g
```

Out[261]:
```
matrix([[-3.24140214,  1.1272942 ]])
```

In [262]:
```
computeCost(X, y, g)
```

Out[262]:
```
4.515955503078912
```

In [263]:
```
x = np.linspace(data.Population.min(), data.Population.max(), 100)
f = g[0, 0] + (g[0, 1] * x)

fig, ax = plt.subplots(figsize=(12,8))
ax.plot(x, f, 'r', label='Prediction')
ax.scatter(data.Population, data.Profit, label='Traning Data')
ax.legend(loc=2)
ax.set_xlabel('Population')
ax.set_ylabel('Profit')
ax.set_title('Predicted Profit vs. Population Size')
```

Out[263]: Text(0.5,1,u'Predicted Profit vs. Population Size')