In [3]:

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

In [6]:

```
df = pd.read_csv('https://raw.githubusercontent.com/satishgunjal/datasets/master/univar
df.head() # To get first n rows from the dataset default value of n is 5
M=len(df)
M
```

Out[6]: 97

In [7]:

```
X = df.values[:, 0]  # get input values from first column
y = df.values[:, 1]  # get output values from second column
m = len(y) # Number of training examples
print('X = ', X[: 5]) # Show only first 5 records
print('y = ', y[: 5])
print('m = ', m)
```

```
X =  [6.1101 5.5277 8.5186 7.0032 5.8598]
y =  [17.592   9.1302 13.662  11.854   6.8233]
m =  97
```

In [8]:

```
X = df.values[:, 0]  # get input values from first column
y = df.values[:, 1]  # get output values from second column
m = len(y) # Number of training examples
print('X = ', X[: 97]) # Show only first 5 records
print('y = ', y[: 97])
print('m = ', m)
```

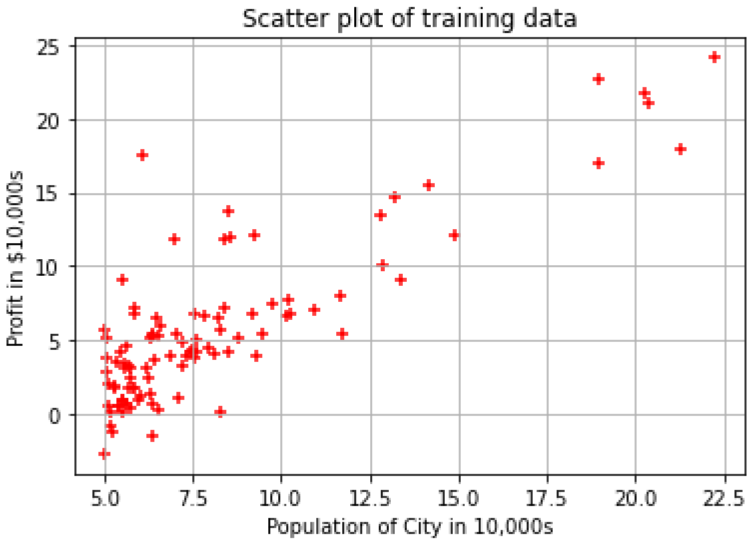
```
X =  [ 6.1101  5.5277  8.5186  7.0032  5.8598  8.3829  7.4764  8.5781  6.4862
  5.0546  5.7107 14.164   5.734   8.4084  5.6407  5.3794  6.3654  5.1301
  6.4296  7.0708  6.1891 20.27    5.4901  6.3261  5.5649 18.945  12.828
 10.957  13.176  22.203   5.2524  6.5894  9.2482  5.8918  8.2111  7.9334
  8.0959  5.6063 12.836   6.3534  5.4069  6.8825 11.708   5.7737  7.8247
  7.0931  5.0702  5.8014 11.7     5.5416  7.5402  5.3077  7.4239  7.6031
  6.3328  6.3589  6.2742  5.6397  9.3102  9.4536  8.8254  5.1793 21.279
 14.908  18.959   7.2182  8.2951 10.236   5.4994 20.341  10.136   7.3345
  6.0062  7.2259  5.0269  6.5479  7.5386  5.0365 10.274   5.1077  5.7292
  5.1884  6.3557  9.7687  6.5159  8.5172  9.1802  6.002   5.5204  5.0594
  5.7077  7.6366  5.8707  5.3054  8.2934 13.394   5.4369]
y =  [17.592    9.1302  13.662   11.854    6.8233  11.886    4.3483  12.
  6.5987   3.8166   3.2522  15.505    3.1551   7.2258   0.71618  3.5129
  5.3048   0.56077  3.6518   5.3893   3.1386  21.767    4.263    5.1875
  3.0825  22.638   13.501    7.0467  14.692   24.147   -1.22     5.9966
 12.134    1.8495   6.5426   4.5623   4.1164   3.3928  10.117    5.4974
  0.55657  3.9115   5.3854   2.4406   6.7318   1.0463   5.1337   1.844
  8.0043   1.0179   6.7504   1.8396   4.2885   4.9981   1.4233  -1.4211
  2.4756   4.6042   3.9624   5.4141   5.1694  -0.74279 17.929   12.054
 17.054    4.8852   5.7442   7.7754   1.0173  20.992    6.6799   4.0259
  1.2784   3.3411  -2.6807   0.29678  3.8845   5.7014   6.7526   2.0576
  0.47953  0.20421  0.67861  7.5435   5.3436   4.2415   6.7981   0.92695
  0.152    2.8214   1.8451   4.2959   7.2029   1.9869   0.14454  9.0551
  0.61705]
m =  97
```

In [9]:

```
plt.scatter(X,y, color='red',marker= '+')
```

```
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Population of City in 10,000s')
plt.ylabel('Profit in $10,000s')
plt.title('Scatter plot of training data')
```

Out[9]: Text(0.5, 1.0, 'Scatter plot of training data')



In [10]:

```
#Lets create a matrix with single column of ones
X_0 = np.ones((m, 1))
X_0[:5]
```

Out[10]:

```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

In [11]:

```
# Using reshape function convert X 1D array to 2D array of dimension 97x1
X_1 = X.reshape(m, 1)
X_1[:10]
```

Out[11]:

```
array([[6.1101],
       [5.5277],
       [8.5186],
       [7.0032],
       [5.8598],
       [8.3829],
       [7.4764],
       [8.5781],
       [6.4862],
       [5.0546]])
```

In [12]:

```
# Lets use hstack() function from numpy to stack X_0 and X_1 horizontally (i.e. column
# This will be our final X matrix (feature matrix)
X = np.hstack((X_0, X_1))
X[:5]
```

Out[12]:

```
array([[1.    , 6.1101],
       [1.    , 5.5277],
```

```
       [1.    , 8.5186],
       [1.    , 7.0032],
       [1.    , 5.8598]])
```

In [13]:

```
theta = np.zeros(2)
theta
```

Out[13]:

```
array([0., 0.])
```

In [14]:

```
def compute_cost(X, y, theta):
  """
  Compute cost for linear regression.

  Input Parameters
  ----------------
  X : 2D array where each row represent the training example and each column represent
      m= number of training examples
      n= number of features (including X_0 column of ones)
  y : 1D array of labels/target value for each traing example. dimension(1 x m)

  theta : 1D array of fitting parameters or weights. Dimension (1 x n)

  Output Parameters
  -----------------
  J : Scalar value.
  """
  predictions = X.dot(theta)
  errors = np.subtract(predictions, y)
  sqrErrors = np.square(errors)
  J = 1 / (2 * m) * np.sum(sqrErrors)

  return J
```

In [15]:

```
# Lets compute the cost for theta values
cost = compute_cost(X, y, theta)
print('The cost for given values of theta_0 and theta_1 =', cost)
```

```
The cost for given values of theta_0 and theta_1 = 32.072733877455676
```

In [16]:

```
def gradient_descent(X, y, theta, alpha, iterations):
  """
  Compute cost for linear regression.

  Input Parameters
  ----------------
  X : 2D array where each row represent the training example and each column represent
      m= number of training examples
      n= number of features (including X_0 column of ones)
  y : 1D array of labels/target value for each traing example. dimension(m x 1)
  theta : 1D array of fitting parameters or weights. Dimension (1 x n)
  alpha : Learning rate. Scalar value
  iterations: No of iterations. Scalar value.

  Output Parameters
  -----------------
  theta : Final Value. 1D array of fitting parameters or weights. Dimension (1 x n)
  cost_history: Conatins value of cost for each iteration. 1D array. Dimansion(m x 1)
```

```
    """
    cost_history = np.zeros(iterations)

    for i in range(iterations):
      predictions = X.dot(theta)
      errors = np.subtract(predictions, y)
      sum_delta = (alpha / m) * X.transpose().dot(errors);
      theta = theta - sum_delta;
      cost_history[i] = compute_cost(X, y, theta)

    return theta, cost_history
```

In [17]:

```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

In [18]:

```
theta, cost_history = gradient_descent(X, y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [-3.63029144  1.16636235]
cost_history = [6.73719046 5.93159357 5.90115471 ... 4.48343473 4.48341145 4.48338826]
```

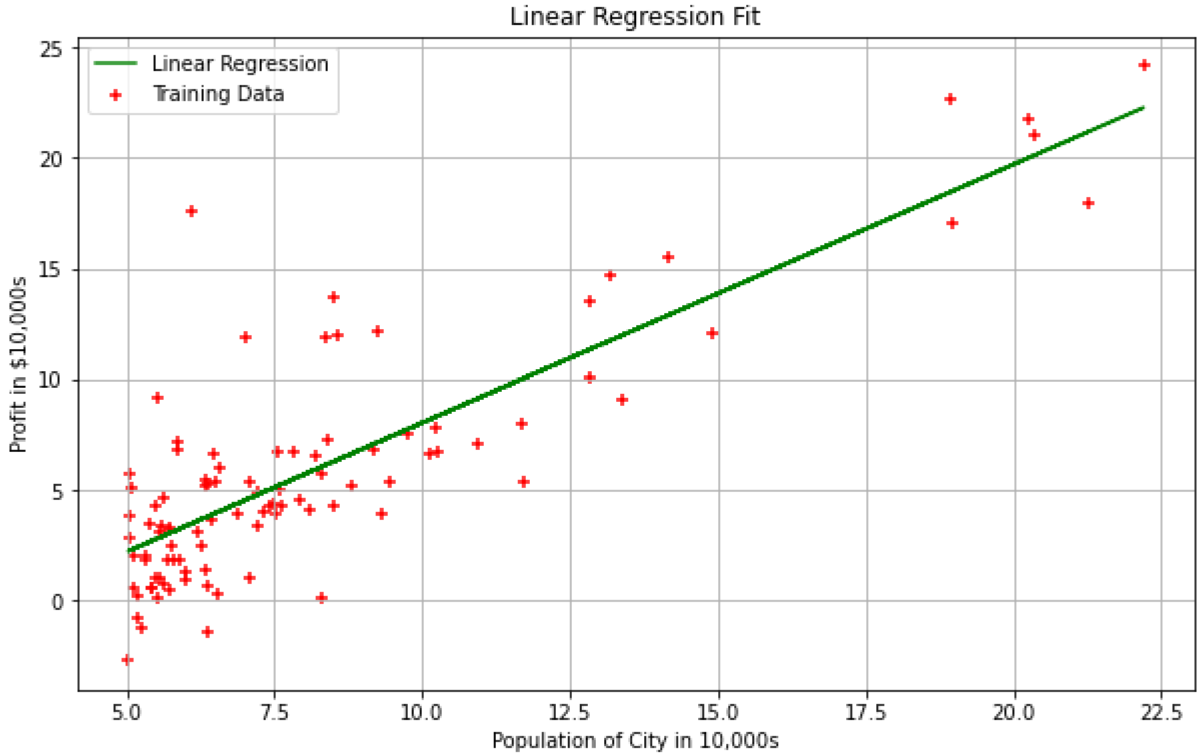
In [19]:

```
# Since X is list of list (feature matrix) lets take values of column of index 1 only
plt.scatter(X[:,1], y, color='red', marker= '+', label= 'Training Data')
plt.plot(X[:,1],X.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Population of City in 10,000s')
plt.ylabel('Profit in $10,000s')
plt.title('Linear Regression Fit')
plt.legend()
```

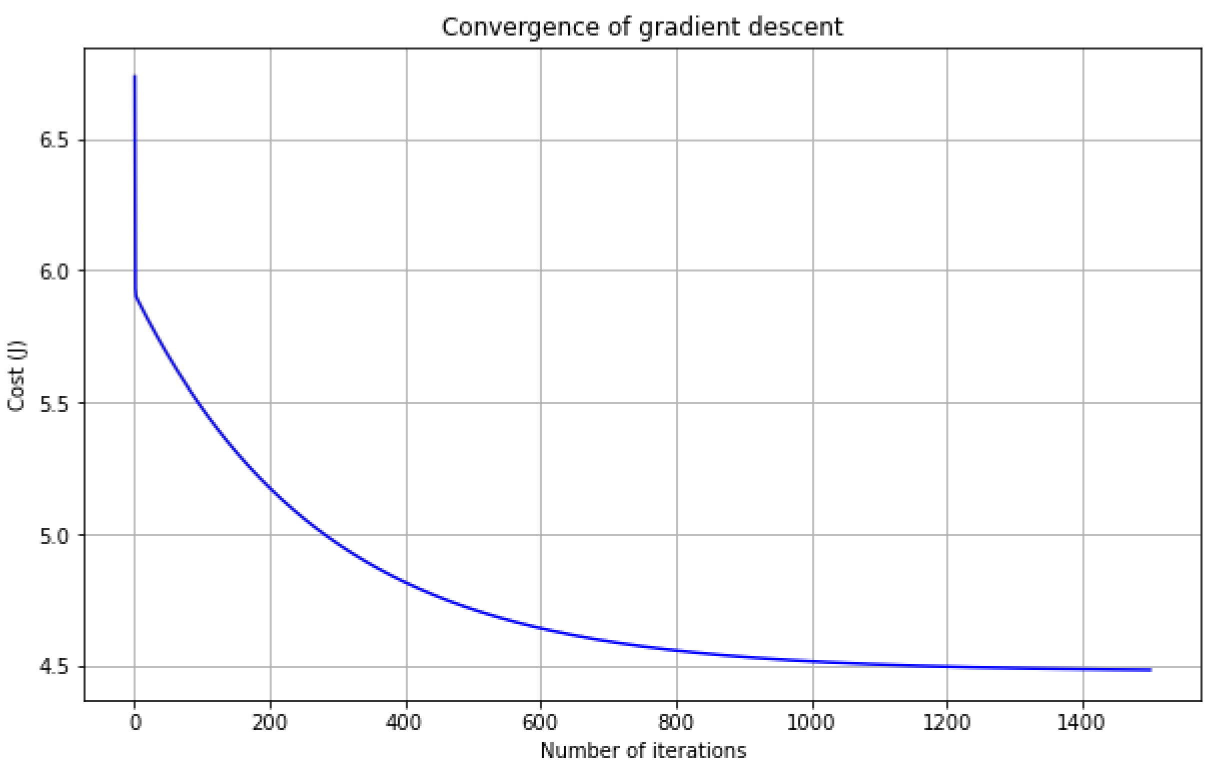
Out[19]:

```
<matplotlib.legend.Legend at 0x28719cd9fa0>
```



In [20]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent')
```

Out[20]: `Text(0.5, 1.0, 'Convergence of gradient descent')`



In [ ]: