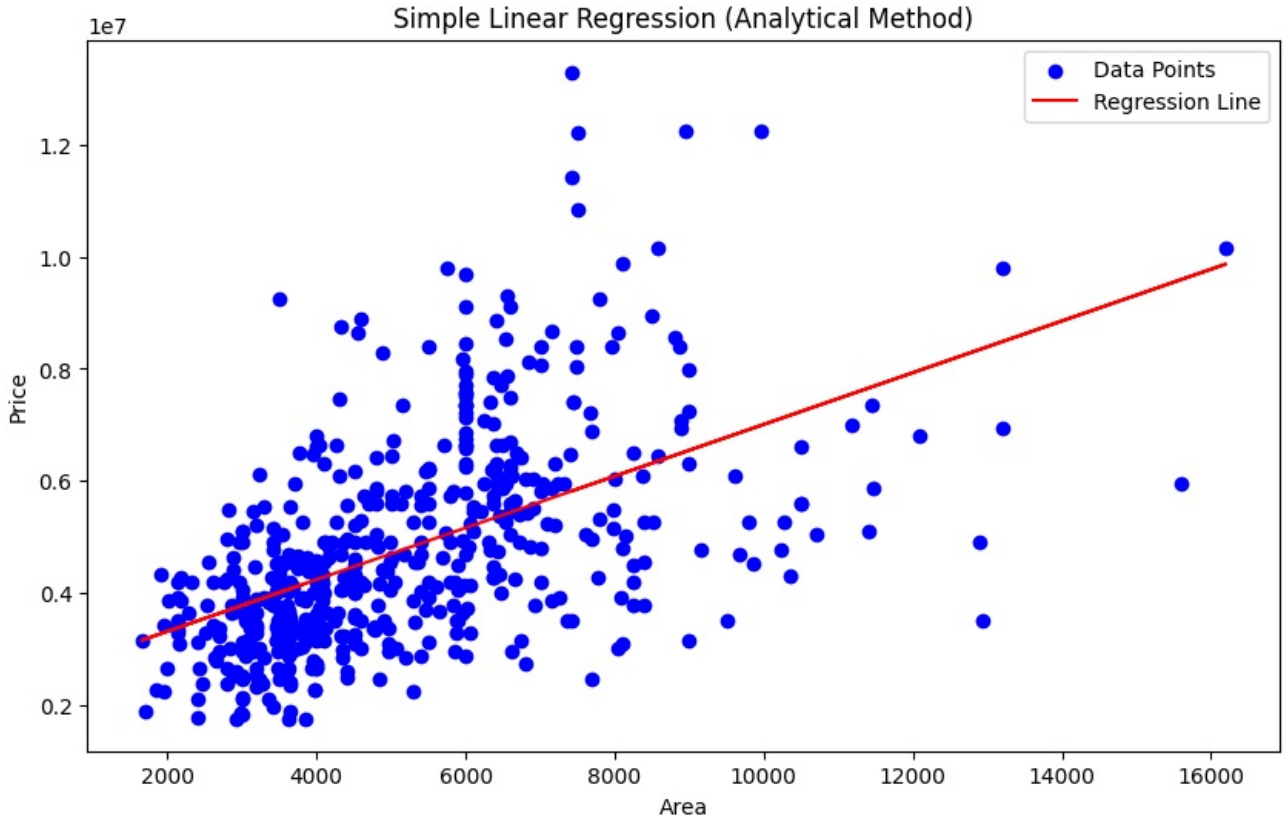
# EXPERIMENT 3

## Analytical Method

In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv(r'C:\Users\hp\OneDrive\Desktop\housing.csv')  # Update the path to your dataset

# Select relevant columns (Area as the independent variable and Price as the dependent variable)
X = df['area'].values
y = df['price'].values

# Add a column of ones to include the intercept (bias) in the model
X_b = np.c_[np.ones((len(X), 1)), X]

# Analytical method: theta = (X^T * X)^-1 * X^T * y
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

print("Optimal parameters (theta):", theta_best)

# Make predictions
y_pred = X_b.dot(theta_best)

# Plotting the data points and the regression line
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Data Points')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.xlabel('Area')
plt.ylabel('Price')
plt.title('Simple Linear Regression (Analytical Method)')
plt.legend()
plt.show()

# Print steps of analytical solution
print("\nSteps of Analytical Solution:")
print("1. Compute X^T * X:")
X_T_X = X_b.T.dot(X_b)
print(X_T_X)
print("\n2. Compute (X^T * X)^-1:")
X_T_X_inv = np.linalg.inv(X_T_X)
print(X_T_X_inv)
print("\n3. Compute X^T * y:")
X_T_y = X_b.T.dot(y)
print(X_T_y)
print("\n4. Compute theta = (X^T * X)^-1 * X^T * y:")
print(theta_best)
```

```
Optimal parameters (theta): [2.38730848e+06 4.61974894e+02]
```




```
Steps of Analytical Solution:
1. Compute X^T * X:
[[5.45000000e+02 2.80704500e+06]
 [2.80704500e+06 1.70197757e+10]]

2. Compute (X^T * X)^-1:
[[ 1.21894053e-02 -2.01037956e-06]
 [-2.01037956e-06  3.90323938e-10]]

3. Compute X^T * y:
[2.59786744e+09 1.45639914e+13]

4. Compute theta = (X^T * X)^-1 * X^T * y:
[2.38730848e+06 4.61974894e+02]
```

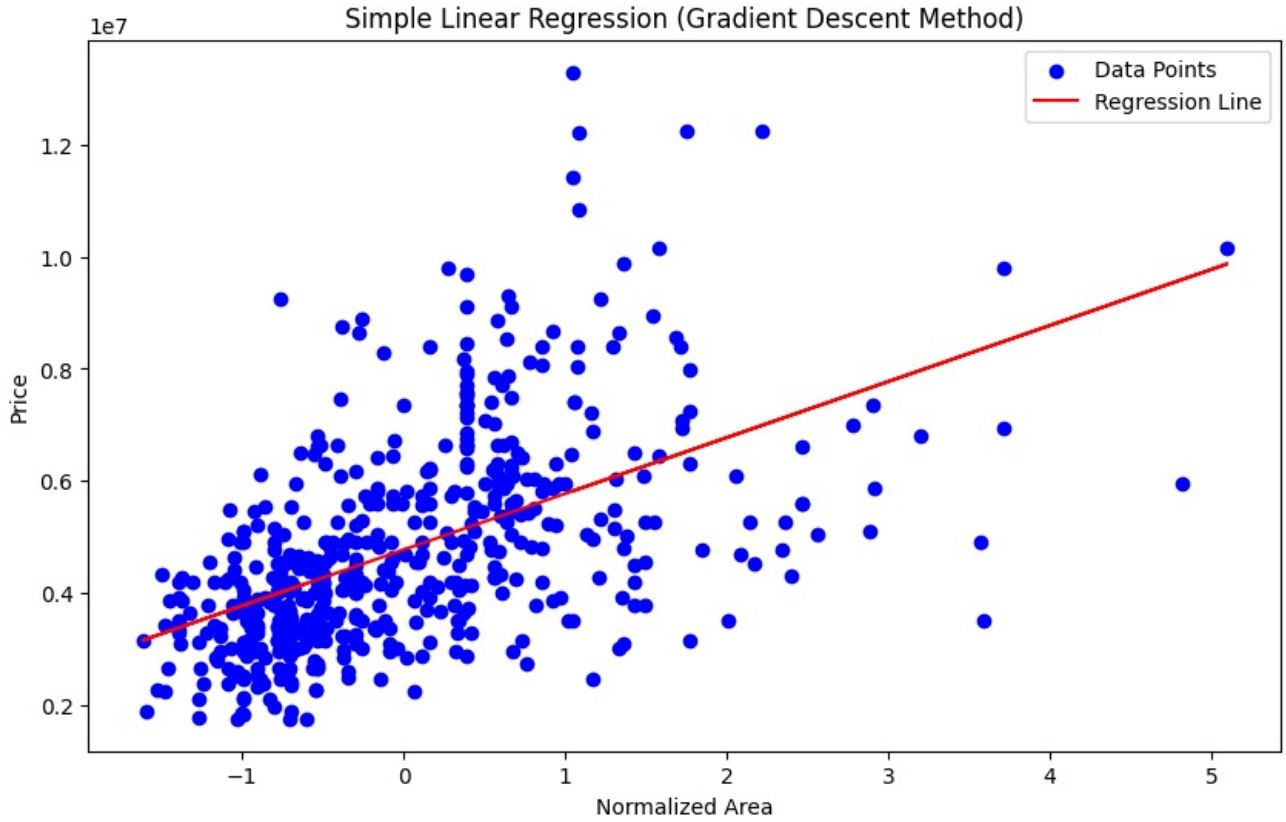
# Machine learning Method

In [9]:

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

# Load the dataset
df = pd.read_csv(r'C:\Users\hp\OneDrive\Desktop\housing.csv')  # Update the path to your dataset
```

```python
# Select relevant columns (Area as the independent variable and Price as the dependent variable)
X = df['area'].values
y = df['price'].values

# Normalize the features for better performance
X = (X - X.mean()) / X.std()

# Add a column of ones to include the intercept (bias) in the model
X_b = np.c_[np.ones((len(X), 1)), X]

# Gradient Descent parameters
learning_rate = 0.01
n_iterations = 1000
m = len(X)

# Initialize theta (weights)
theta = np.random.randn(2)

# Gradient Descent Loop
print(f"{'Iteration':<10}{'Theta0':<15}{'Theta1':<15}{'Cost Function'}")
print("-" * 45)

for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - learning_rate * gradients
    cost = np.sum((X_b.dot(theta) - y) ** 2) / (2 * m)

    if iteration % 100 == 0:  # Print every 100 iterations
        print(f"{iteration:<10}{theta[0]:<15.4f}{theta[1]:<15.4f}{cost:<20.4f}")

print("\nOptimal parameters (theta):", theta)

# Make predictions
y_pred = X_b.dot(theta)

# Plotting the data points and the regression line
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Data Points')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.xlabel('Normalized Area')
plt.ylabel('Price')
plt.title('Simple Linear Regression (Gradient Descent Method)')
plt.legend()
plt.show()
```

```
Iteration Theta0         Theta1         Cost Function
---------------------------------------------
0         95334.3807     20031.7703     12637163710153.3242
100       4147210.9350   871451.2915    1444805478982.1462
200       4684569.0042   984366.1703    1247954880020.1990
300       4755833.1927   999340.8913    1244492682196.3521
400       4765284.2177   1001326.8322   1244431789245.8879
500       4766537.6085   1001590.2068   1244430718263.9263
600       4766703.8326   1001625.1354   1244430699427.5530
700       4766725.8772   1001629.7676   1244430699096.2595
800       4766728.8007   1001630.3819   1244430699090.4329
900       4766729.1884   1001630.4634   1244430699090.3306

Optimal parameters (theta): [4766729.23968417 1001630.47418752]
```




In [ ]: