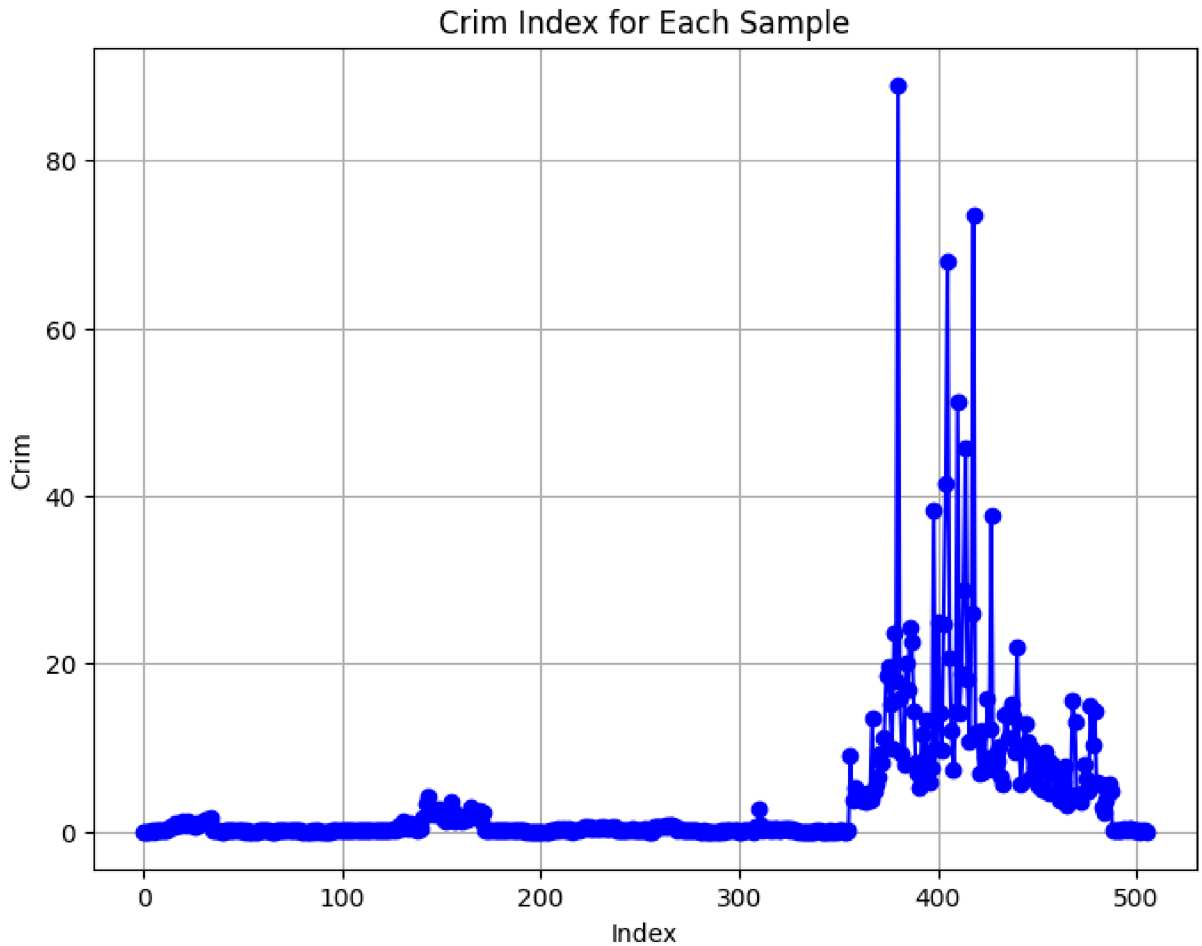
Question 8

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [7]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Task 1: Read the dataset and report instances and features
# Reading the dataset from the 'housing.header.txt' file
df = pd.read_csv('housing.header.txt')

# Report the number of instances (rows) and features (columns)
print(f"Number of instances: {df.shape[0]}")
print(f"Number of features: {df.shape[1]}")

# Plot all samples with respect to the 'Crim' index
plt.figure(figsize=(8, 6))
plt.plot(df.index, df['Crim'], marker='o', linestyle='-', color='b')
plt.xlabel('Index')
plt.ylabel('Crim')
plt.title('Crim Index for Each Sample')
plt.grid(True)
plt.show()
```

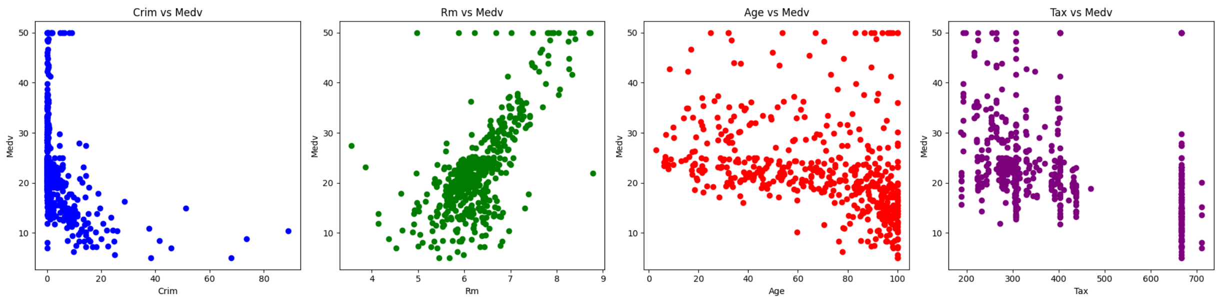
```
Number of instances: 506
Number of features: 14
```



In [8]:

```
# Task 2: Show 1x4 scatter plots
plt.figure(figsize=(20, 5))

# Crim vs Medv
plt.subplot(1, 4, 1)
plt.scatter(df['Crim'], df['Medv'], color='b')
plt.xlabel('Crim')
plt.ylabel('Medv')
plt.title('Crim vs Medv')

# Rm vs Medv
plt.subplot(1, 4, 2)
plt.scatter(df['Rm'], df['Medv'], color='g')
plt.xlabel('Rm')
plt.ylabel('Medv')
plt.title('Rm vs Medv')

# Age vs Medv
plt.subplot(1, 4, 3)
plt.scatter(df['Age'], df['Medv'], color='r')
plt.xlabel('Age')
plt.ylabel('Medv')
plt.title('Age vs Medv')

# Tax vs Medv
plt.subplot(1, 4, 4)
```

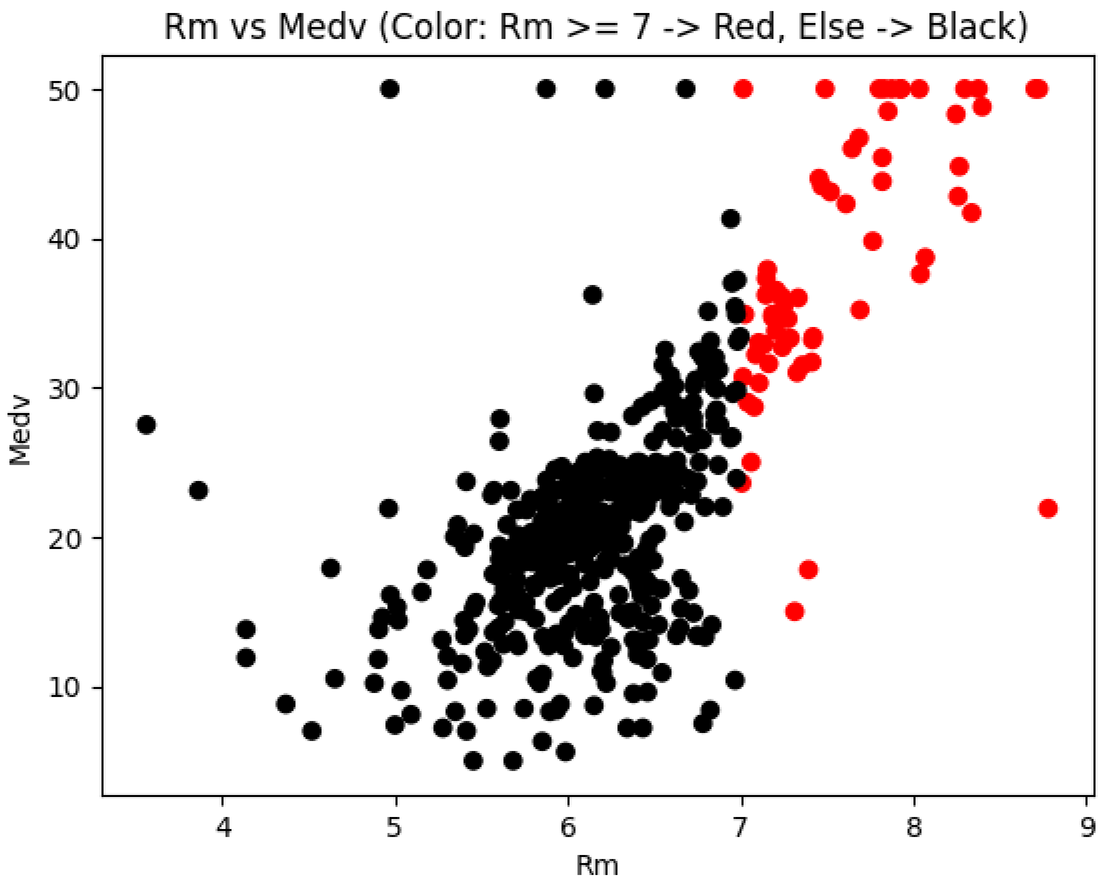
```
plt.scatter(df['Tax'], df['Medv'], color='purple')
plt.xlabel('Tax')
plt.ylabel('Medv')
plt.title('Tax vs Medv')

plt.tight_layout()
plt.show()

# Correlation explanation (to be manually added based on the visual inspection)
# Typically, Crim (crime rate) and Tax are negatively correlated with Medv,
# while Rm (number of rooms) is positively correlated. Age might show a weaker or mix
```



In [11]:
```
# 3. Create a subset where Crim <= 1 and Rm >= 6
subset_df = df[(df['Crim'] <= 1) & (df['Rm'] >= 6)]
print(f'Subset size: {subset_df.shape}')
```

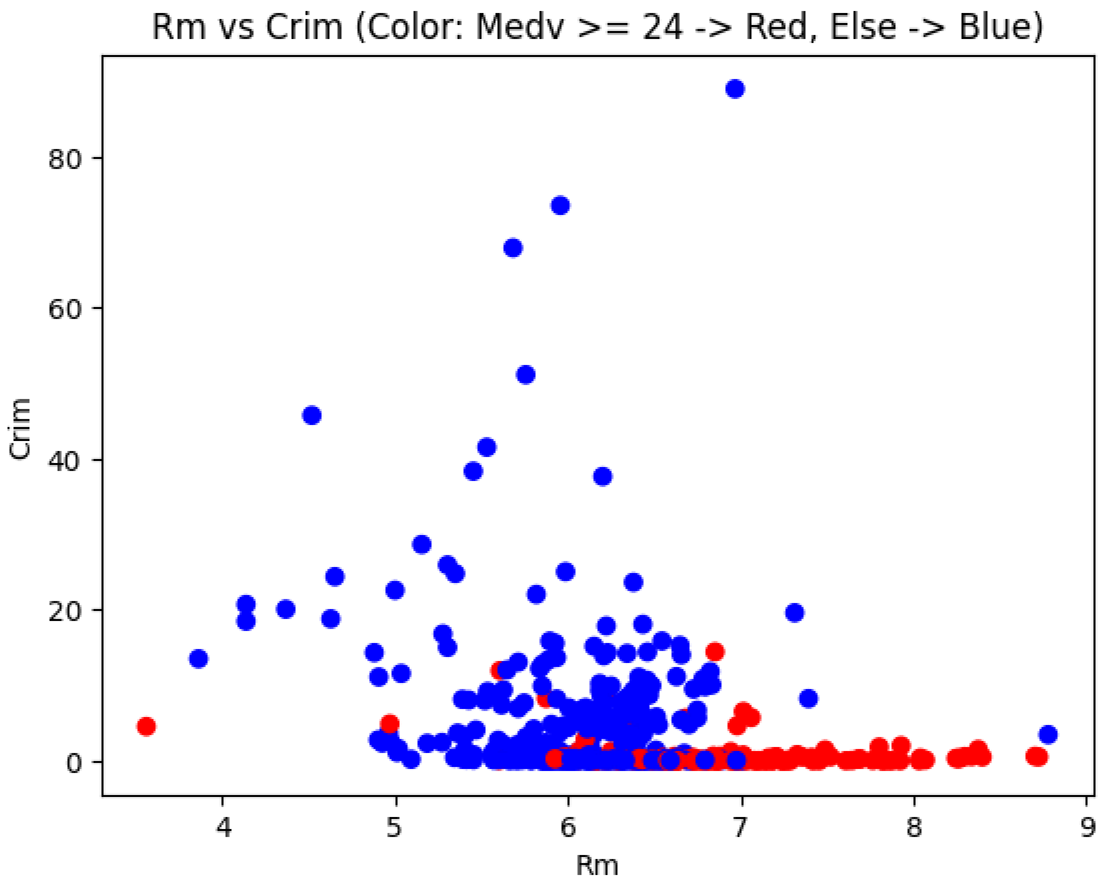
```
Subset size: (236, 14)
```

In [12]:
```
# 4. Scatter plot between Rm and Medv, color Rm >= 7 as red, rest as black
colors = np.where(df['Rm'] >= 7, 'red', 'black')
plt.scatter(df['Rm'], df['Medv'], c=colors)
plt.xlabel('Rm')
plt.ylabel('Medv')
plt.title('Rm vs Medv (Color: Rm >= 7 -> Red, Else -> Black)')
plt.show()
```

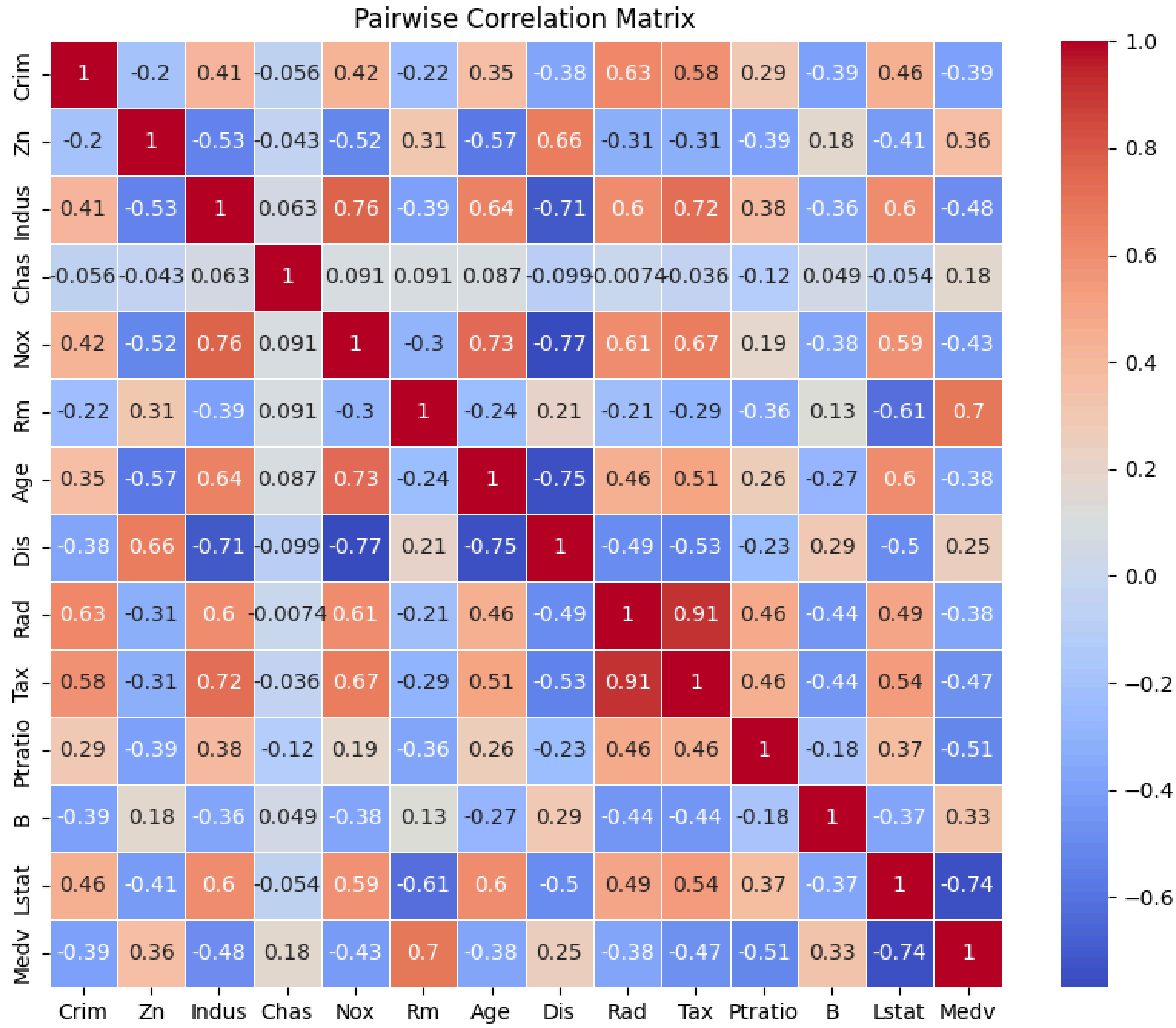


In [13]:

```
# 5. Scatter plot between Rm and Crim, color Medv >= 24 as red, else blue
colors = np.where(df['Medv'] >= 24, 'red', 'blue')
plt.scatter(df['Rm'], df['Crim'], c=colors)
plt.xlabel('Rm')
plt.ylabel('Crim')
plt.title('Rm vs Crim (Color: Medv >= 24 -> Red, Else -> Blue)')
plt.show()
```



In [14]:
```
# 6. Report the pairwise correlation matrix
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Pairwise Correlation Matrix')
plt.show()
```



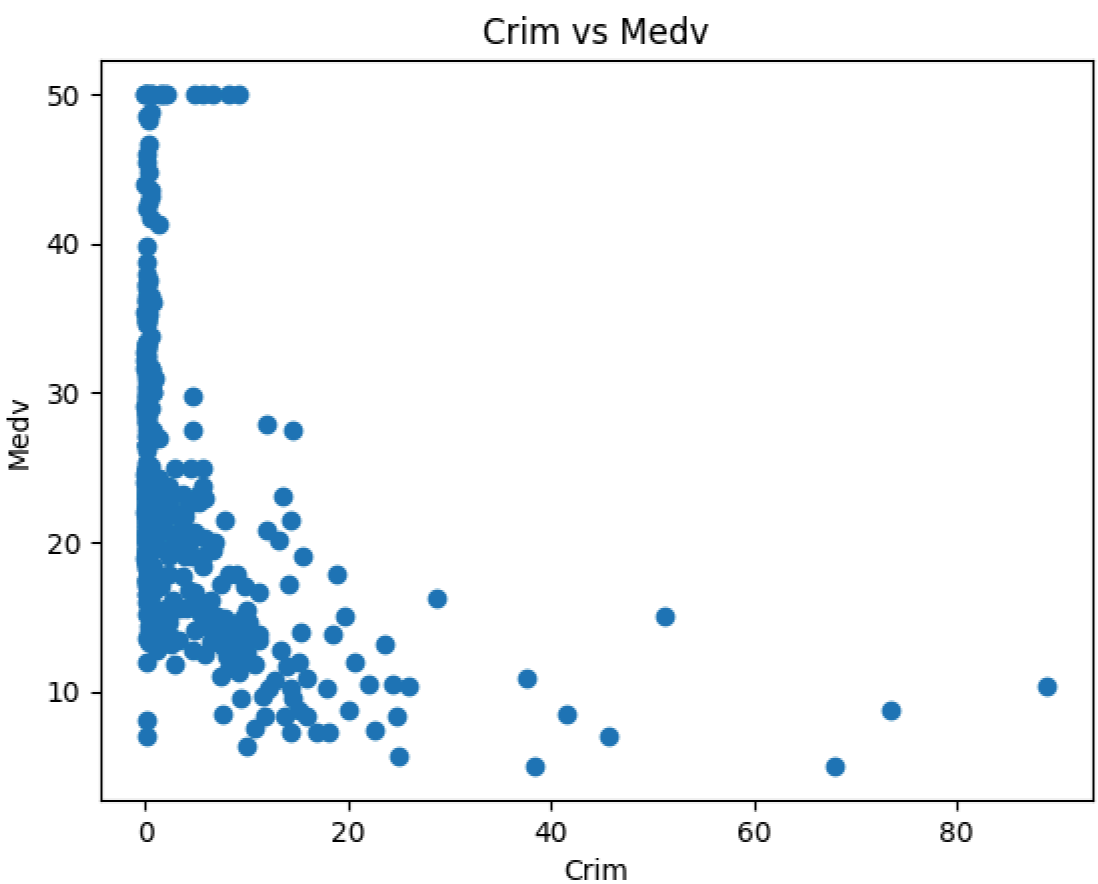
7. Explain the most positively and negatively correlated variables to Medv

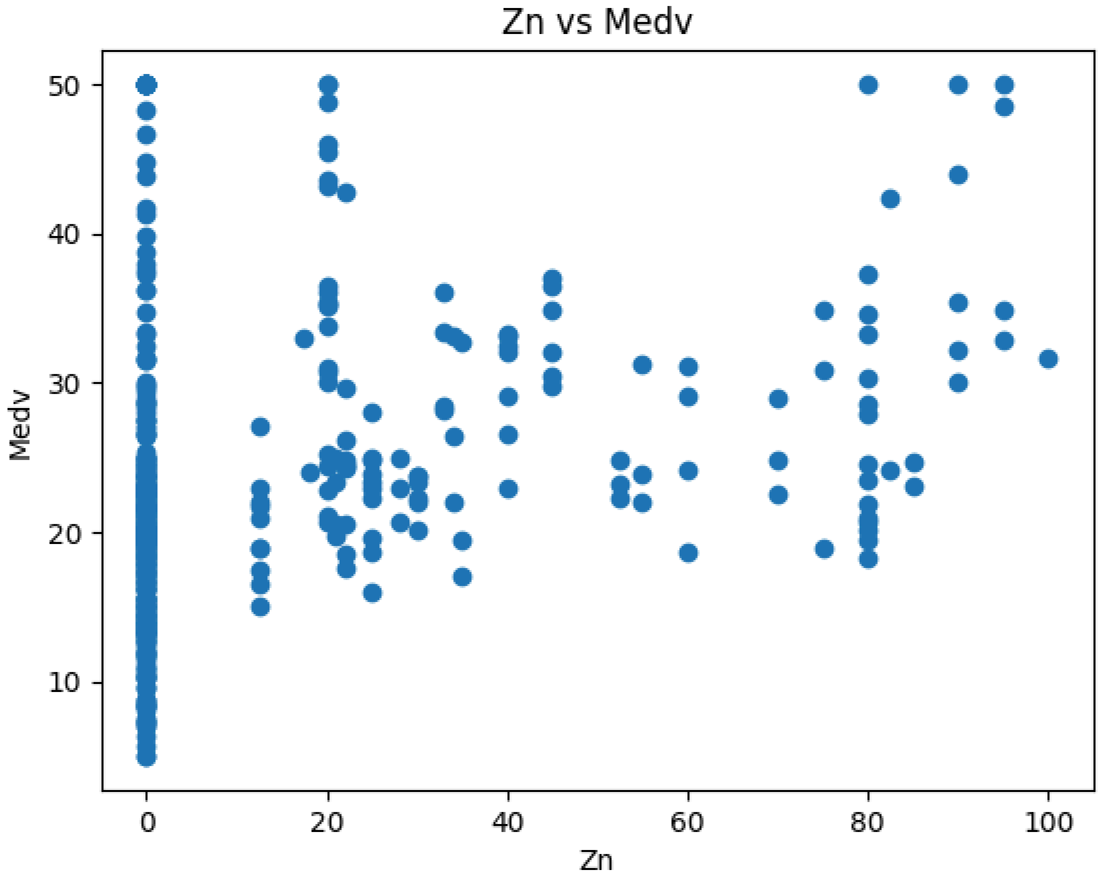
The variable most positively correlated to Medv is Rm, which (number of rooms) bears the highest positive correlation (around 0.7) with this method. This signifies that houses having several rooms are likely to have higher median values.

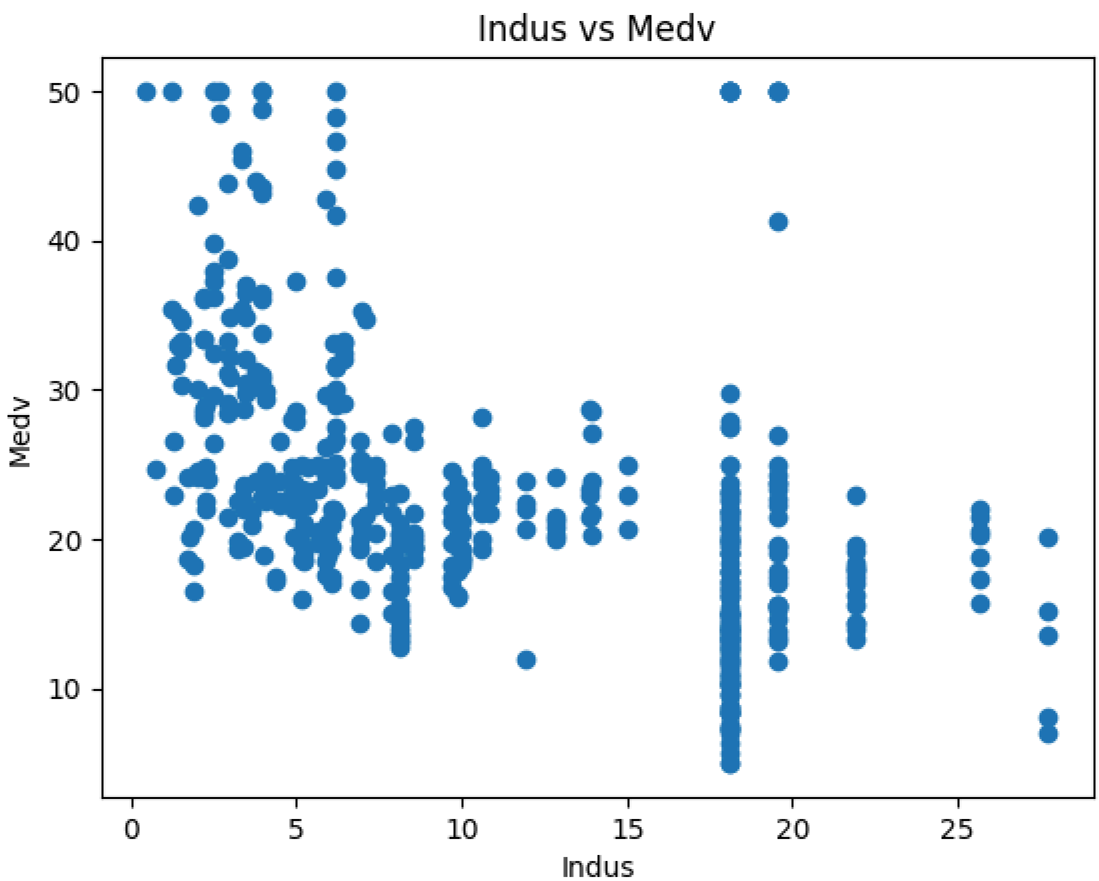
The variable that is most negatively correlated to Medv is Lstat (percentage of lower status of the population). With a

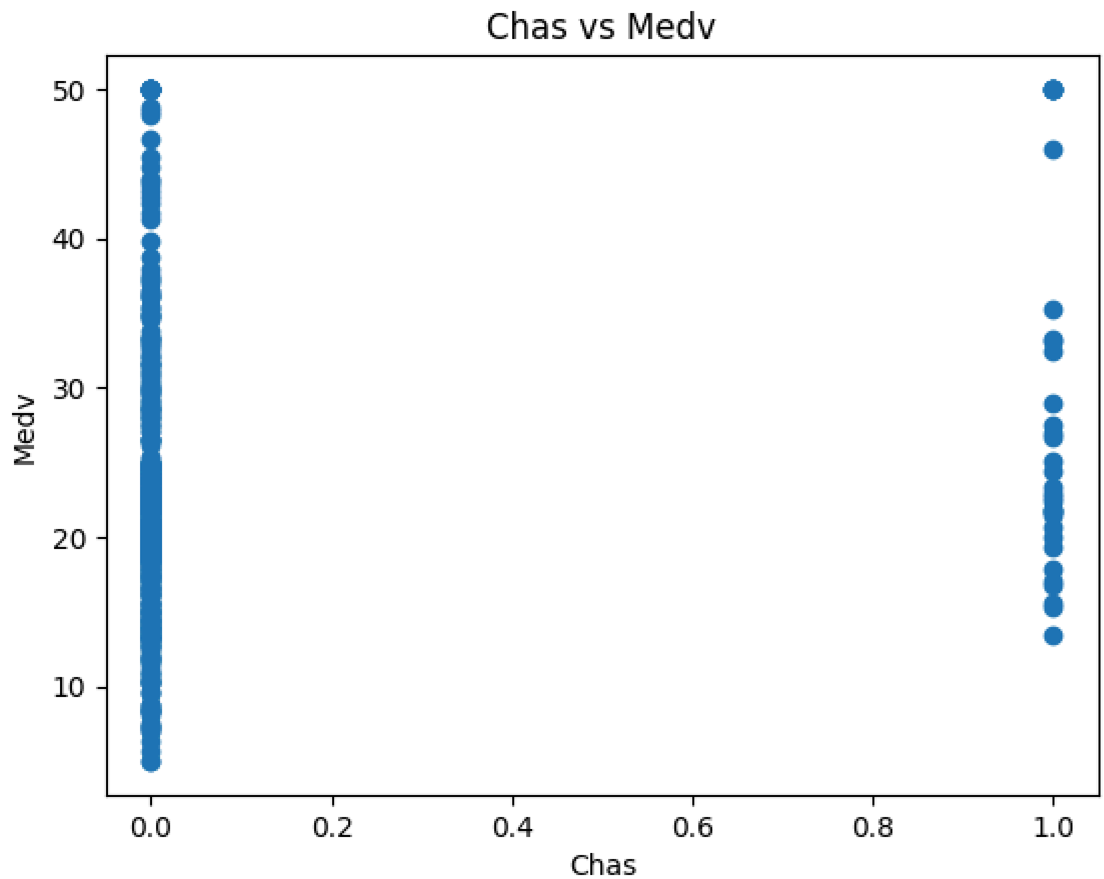
correlation value of about -0.74, it has the strongest negative correlation with Medv. This suggests that an increased number of people from lower-status neighborhoods contributes to low median house prices in given areas.

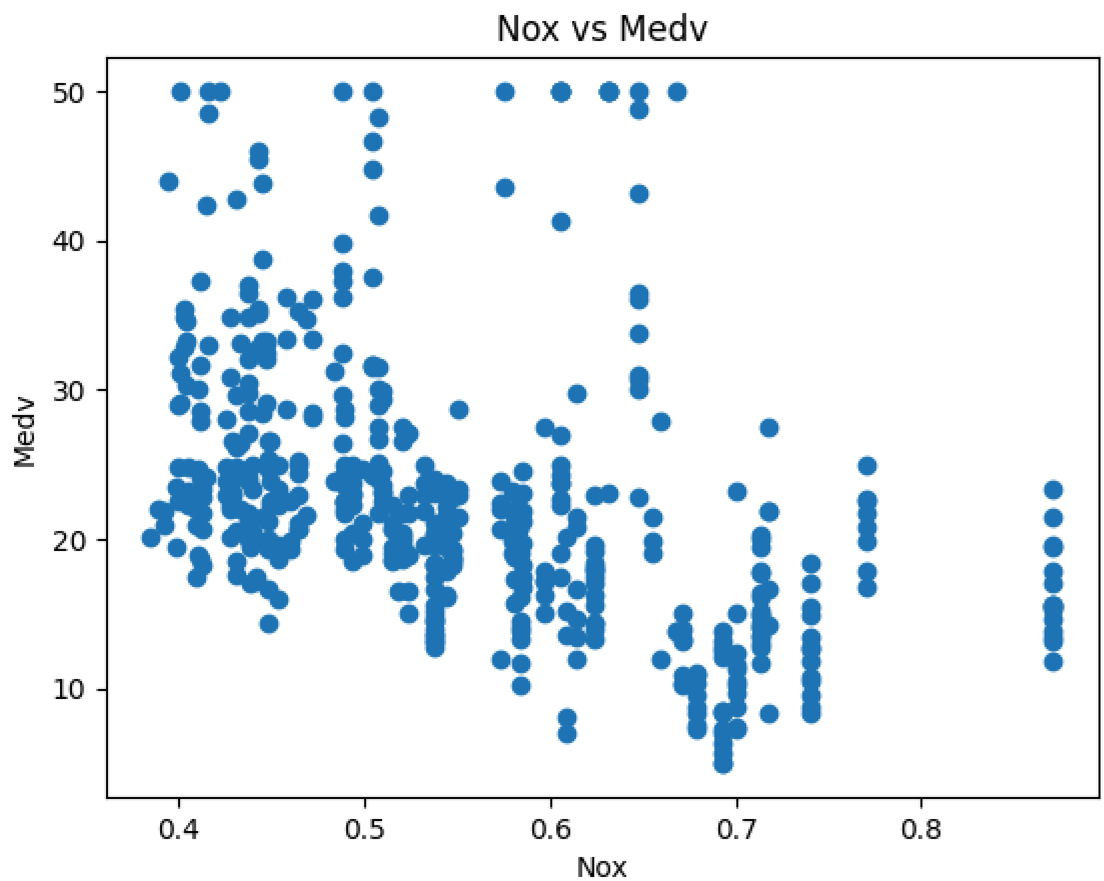
In [15]:
```
# 8. Draw scatterplots to show relationship between each attribute and Medv
for column in df.columns:
    if column != 'Medv':
        plt.figure()
        plt.scatter(df[column], df['Medv'])
        plt.xlabel(column)
        plt.ylabel('Medv')
        plt.title(f'{column} vs Medv')
        plt.show()
```

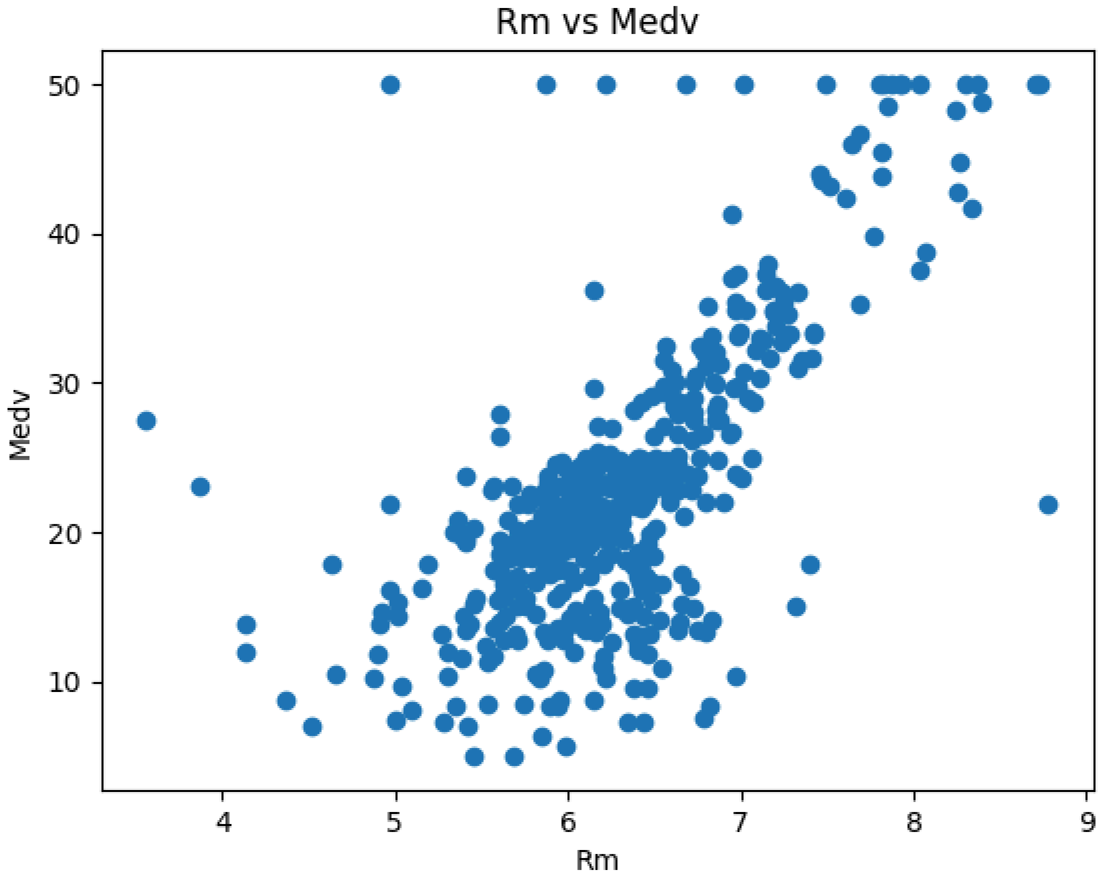


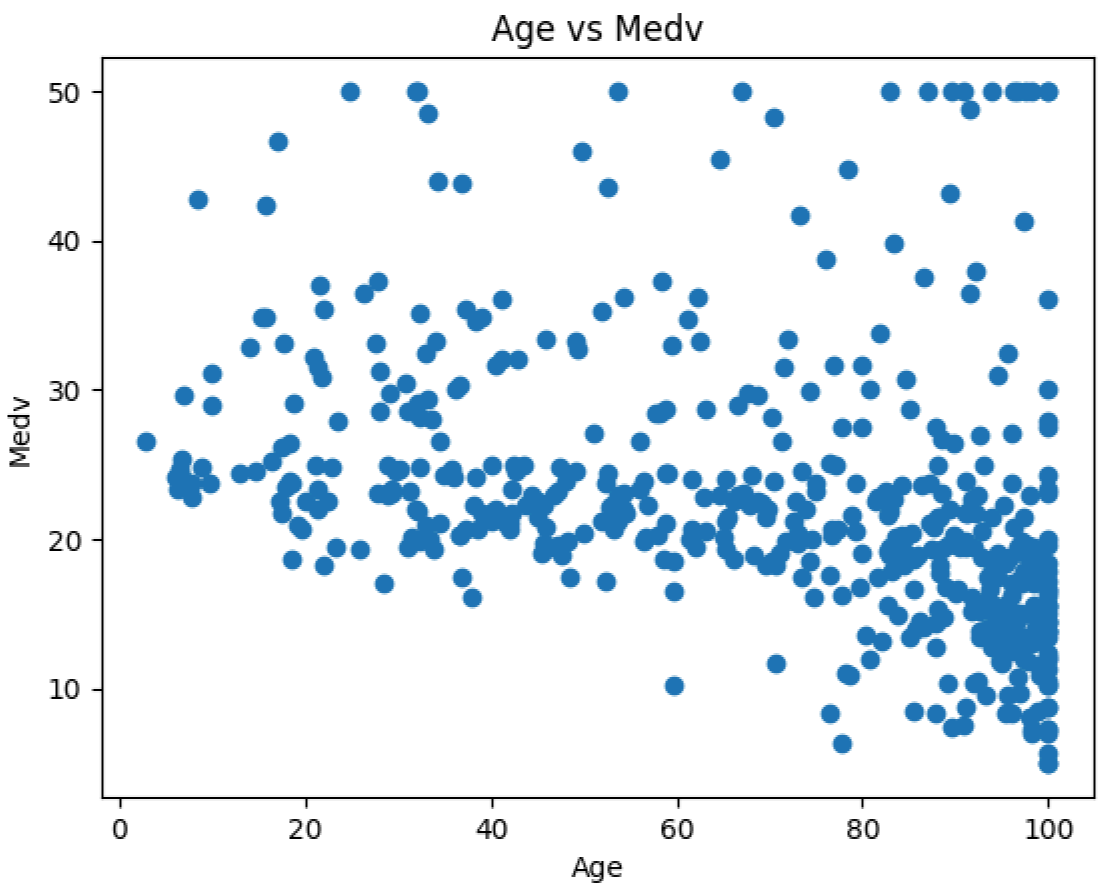

Zn vs Medv
50
40
30
20
10
Medv
0
20
40
60
80
100
Zn

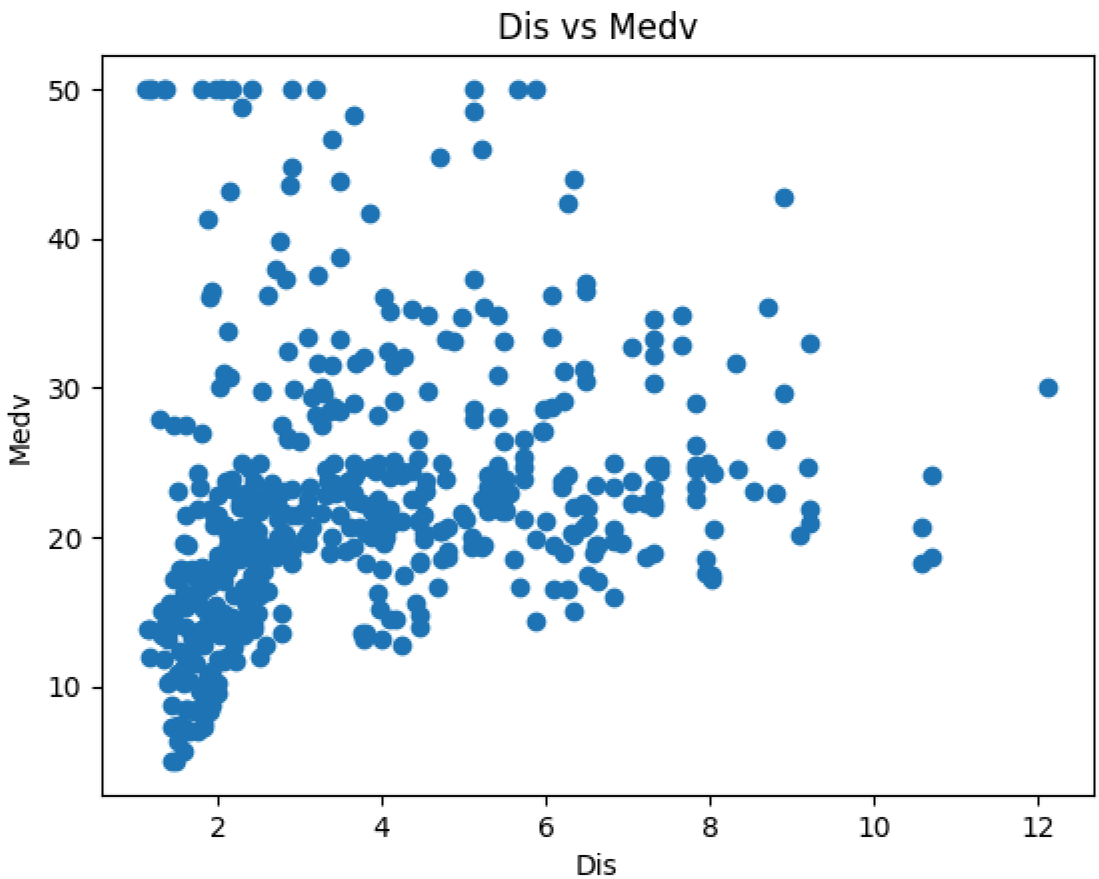

Indus vs Medv
50
40
30
20
10
Medv
0
5
10
15
20
25
Indus

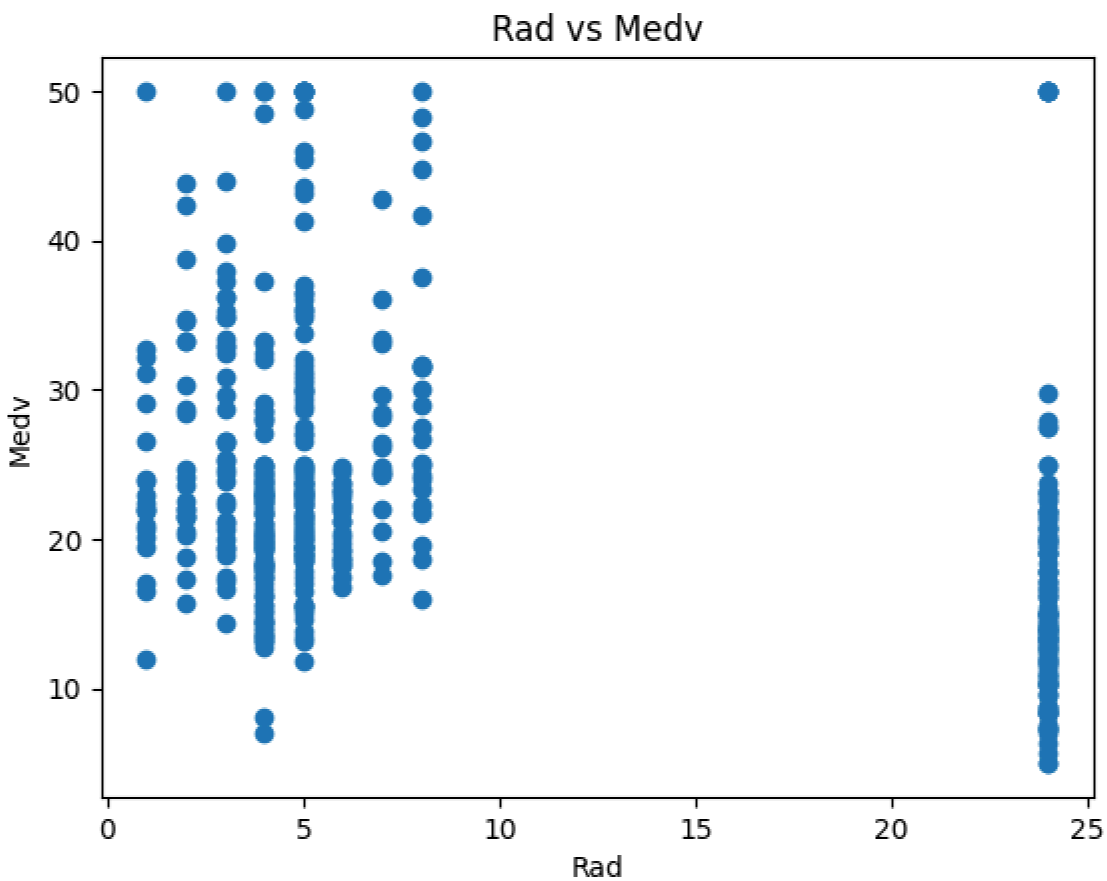

Chas vs Medv
Medv
Chas
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0

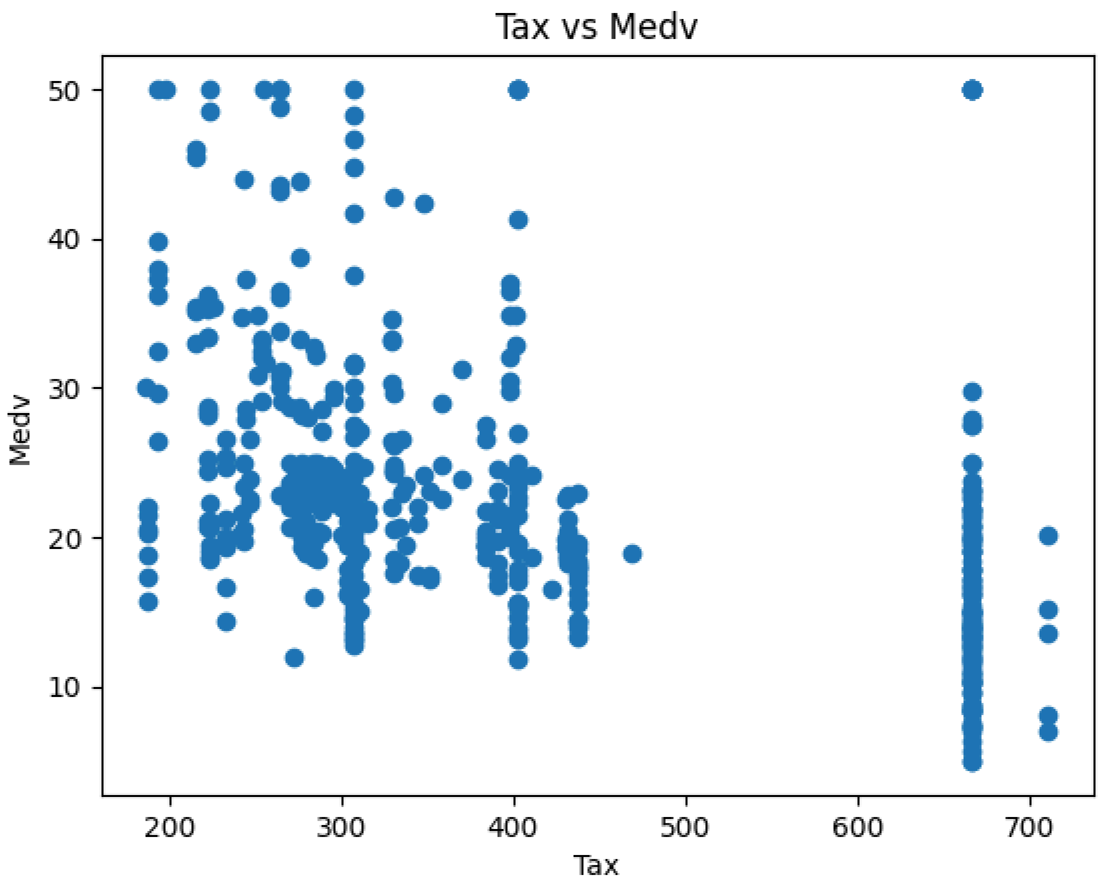

Nox vs Medv
Medv
Nox
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8

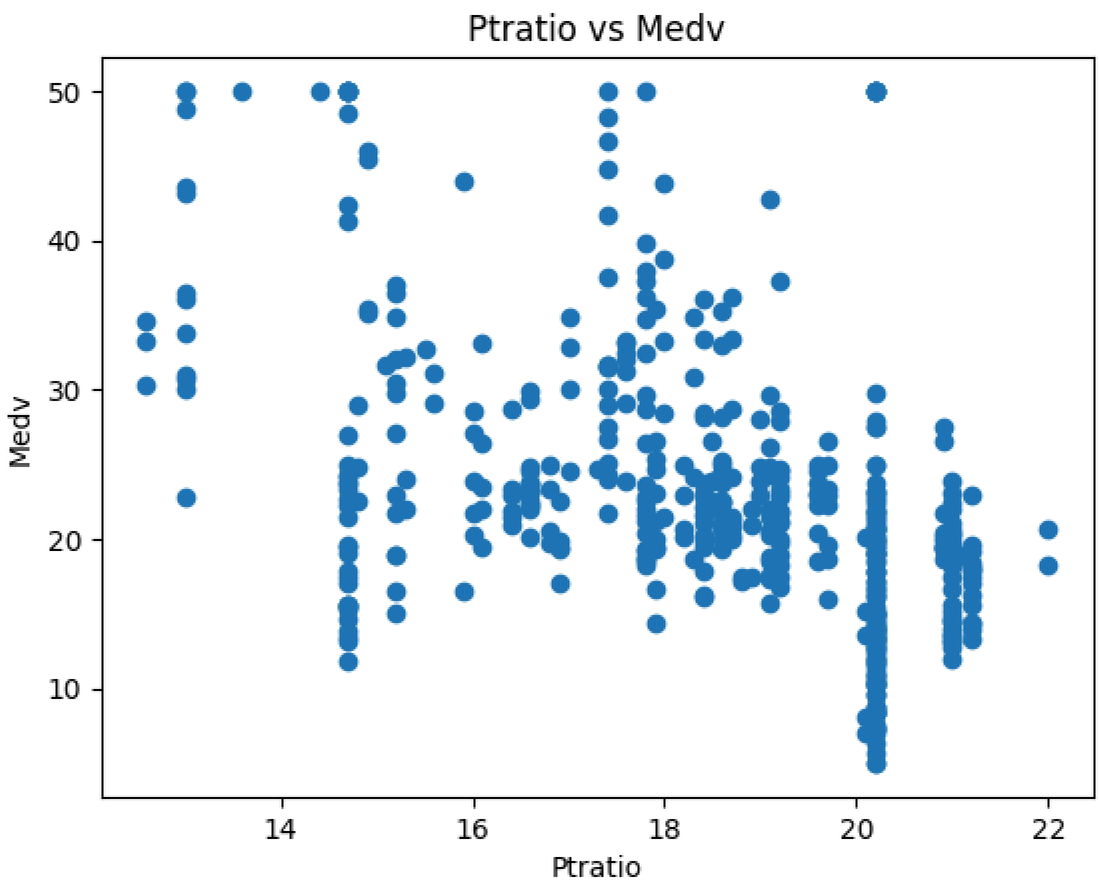

Rm vs Medv
Medv
Rm
50
40
30
20
10
4
5
6
7
8
9

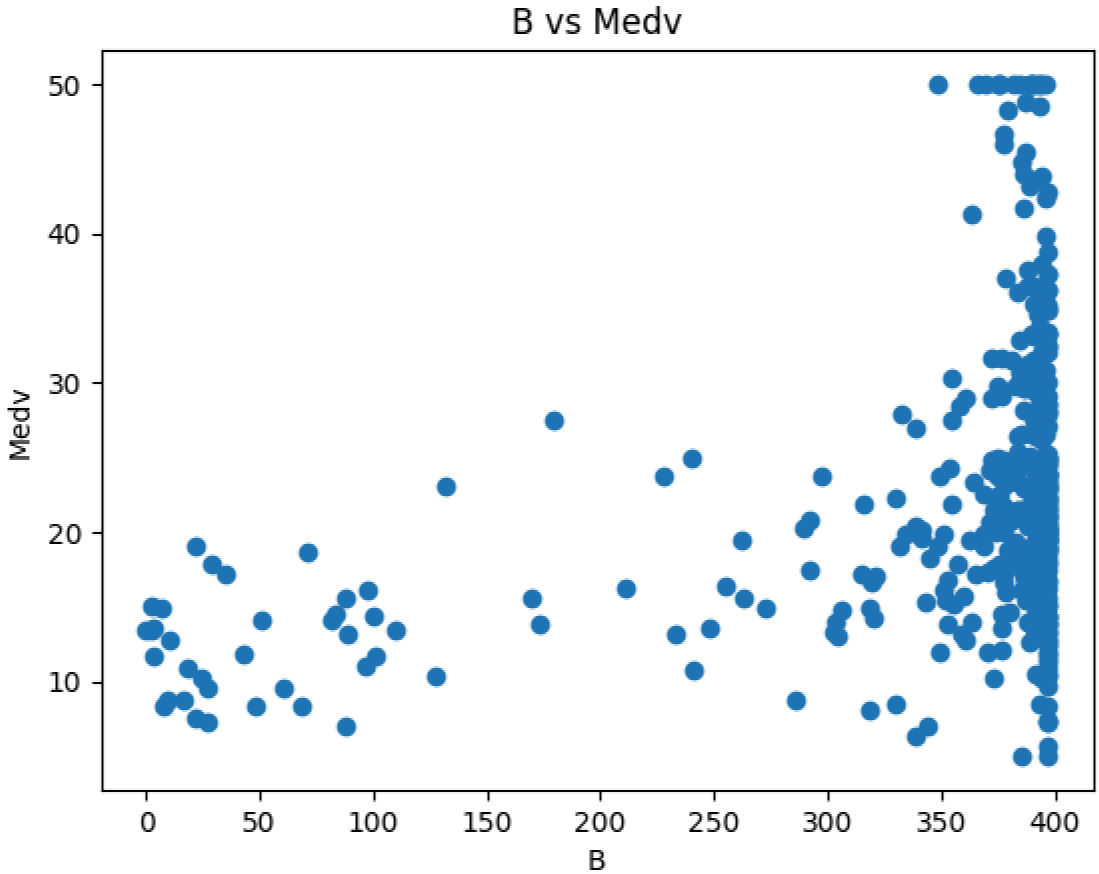

Age vs Medv
Medv
Age
50
40
30
20
10
0
20
40
60
80
100

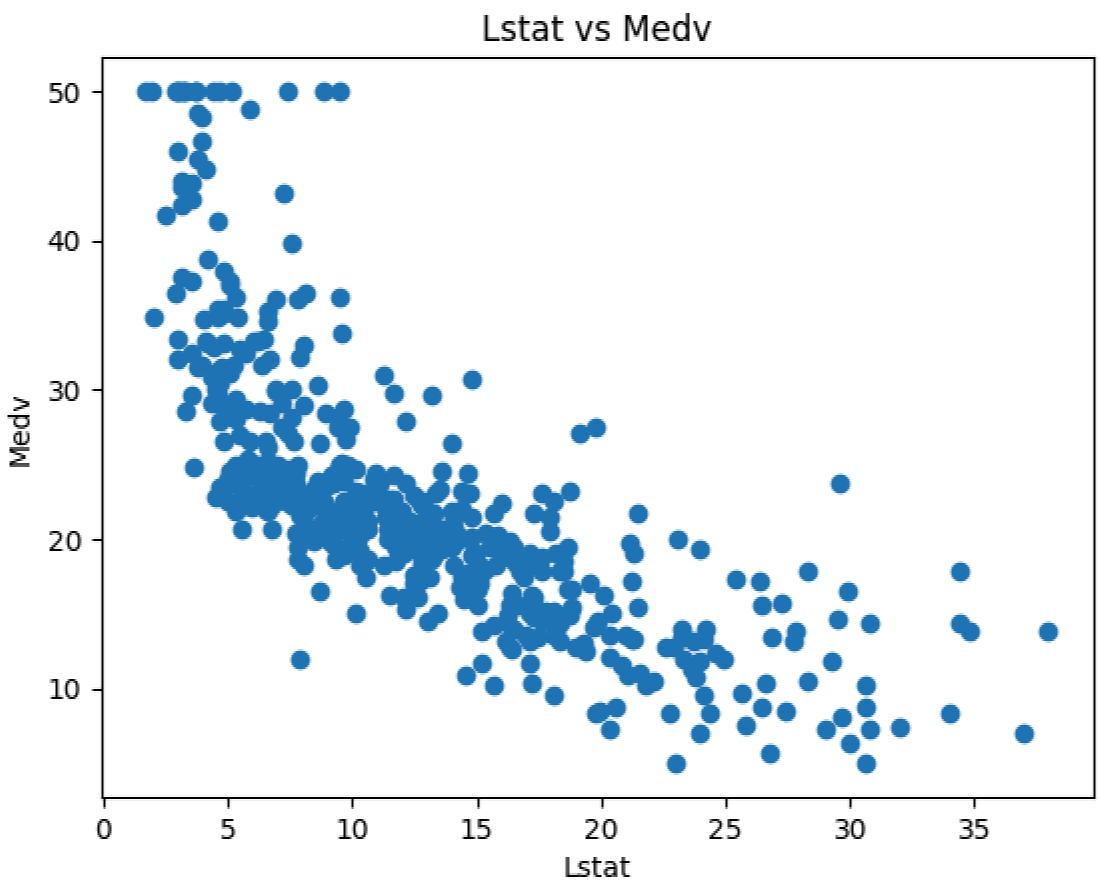

Dis vs Medv
Medv
Dis
50
40
30
20
10
2
4
6
8
10
12


Rad vs Medv
Medv
Rad
50
40
30
20
10
0
5
10
15
20
25


Tax vs Medv
Medv
Tax
10
20
30
40
50
200
300
400
500
600
700


Ptratio vs Medv
Medv
Ptratio
10
20
30
40
50
14
16
18
20
22


B vs Medv
Medv
B
0
50
100
150
200
250
300
350
400
10
20
30
40
50


Lstat vs Medv
Medv
Lstat
0
5
10
15
20
25
30
35
10
20
30
40
50

In [18]:
```
# 9. Create a new instance with specified values and include it in the dataframe
new_instance = {
    'Crim': 1.0, 'Zn': 0.2, 'Indus': 6, 'Chas': 0.1, 'Nox': 6.5, 'Rm': 5,
    'Age': 100, 'Dis': 4.1, 'Rad': 4.5, 'Tax': 21, 'Ptratio': 20, 'B': 300,
    'Lstat': 12, 'Medv': 20.5
}

# Convert the new instance to a DataFrame
new_instance_df = pd.DataFrame([new_instance])

# Concatenate the new instance with the original DataFrame
new_df = pd.concat([df, new_instance_df], ignore_index=True)

# Print the number of instances and features
print(f'New number of instances: {new_df.shape[0]}')
print(f'New number of features: {new_df.shape[1]}')
```

```
New number of instances: 507
New number of features: 14
```

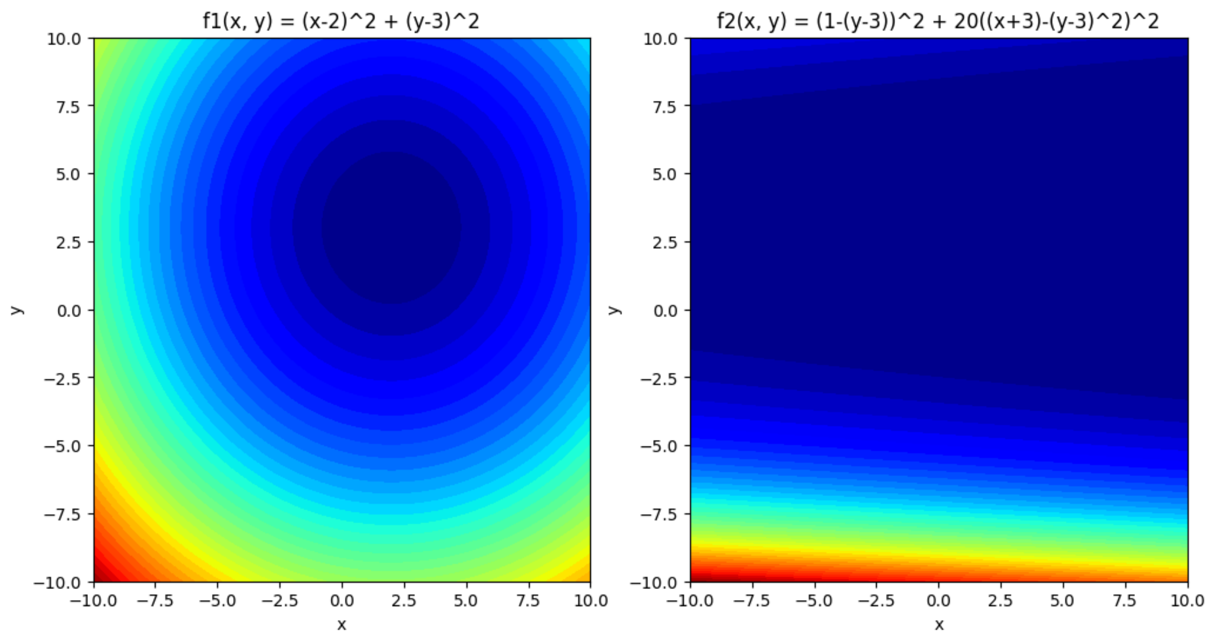
In [19]:
```
# 10. Create a new feature "Dummy" with values randomly generated in [0, 5]
new_df['Dummy'] = np.random.uniform(0, 5, new_df.shape[0])
print(new_df.head())
```

```
      Crim    Zn  Indus  Chas    Nox     Rm   Age     Dis  Rad  Tax  Ptratio  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296     15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222     18.7

        B  Lstat  Medv     Dummy
0  396.90   4.98  24.0  3.326759
1  396.90   9.14  21.6  3.982851
2  392.83   4.03  34.7  0.182601
3  394.63   2.94  33.4  0.259406
4  396.90   5.33  36.2  2.286092
```

Question 9

In [20]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define f1 and f2
def f1(x, y):
    return (x - 2)**2 + (y - 3)**2

def f2(x, y):
    return (1 - (y - 3))**2 + 20 * ((x + 3) - (y - 3)**2)**2

# Generate a grid of values for plotting
x_vals = np.linspace(-10, 10, 400)
y_vals = np.linspace(-10, 10, 400)
X, Y = np.meshgrid(x_vals, y_vals)
```

```
# Compute function values for the grid
Z1 = f1(X, Y)
Z2 = f2(X, Y)

# Plot f1 and f2
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot f1(x, y)
axs[0].contourf(X, Y, Z1, 50, cmap='jet')
axs[0].set_title('f1(x, y) = (x-2)^2 + (y-3)^2')
axs[0].set_xlabel('x')
axs[0].set_ylabel('y')

# Plot f2(x, y)
axs[1].contourf(X, Y, Z2, 50, cmap='jet')
axs[1].set_title('f2(x, y) = (1-(y-3))^2 + 20((x+3)-(y-3)^2)^2')
axs[1].set_xlabel('x')
axs[1].set_ylabel('y')

plt.show()
```



In [21]:

```
# Define gradients of f1 and f2
def grad_f1(x, y):
    df_dx = 2 * (x - 2)
    df_dy = 2 * (y - 3)
    return np.array([df_dx, df_dy])

def grad_f2(x, y):
    df_dx = 40 * ((x + 3) - (y - 3)**2)
    df_dy = 2 * (y - 3) - 80 * (y - 3) * ((x + 3) - (y - 3)**2)
    return np.array([df_dx, df_dy])

# Gradient descent parameters
eta = 0.5
T = 100
```

```
initial_point = np.array([0.0, 0.0])

# Gradient descent function
def gradient_descent(grad_func, initial_point, eta, T):
    x, y = initial_point
    values = []

    for t in range(T):
        grad = grad_func(x, y)
        x -= eta * grad[0]
        y -= eta * grad[1]
        values.append((x, y))
    return values

# Perform gradient descent for f1 and f2
values_f1 = gradient_descent(grad_f1, initial_point, eta, T)
values_f2 = gradient_descent(grad_f2, initial_point, eta, T)

# Print values for f1 and f2 at each iteration
print("Iteration | f1(x, y) values | f2(x, y) values")
for t in range(T):
    x1, y1 = values_f1[t]
    x2, y2 = values_f2[t]
    print(f"{t:3d}       | {f1(x1, y1):.4f}          | {f2(x2, y2):.4f}")
```

```
Iteration | f1(x, y) values | f2(x, y) values
  0       | 0.0000          | 5372221491541.0000
  1       | 0.0000          | 99276782068473523704828477821880561251l232.0000
  2       | 0.0000          | 62621439810068095608469582833341281498271638083715574341
34456589389808713780532535934327329141801123726800685683908201843968180224.0000
  3       | 0.0000          | inf
  4       | 0.0000          | inf
  5       | 0.0000          | nan
  6       | 0.0000          | nan
  7       | 0.0000          | nan
  8       | 0.0000          | nan
  9       | 0.0000          | nan
 10       | 0.0000          | nan
 11       | 0.0000          | nan
 12       | 0.0000          | nan
 13       | 0.0000          | nan
 14       | 0.0000          | nan
 15       | 0.0000          | nan
 16       | 0.0000          | nan
 17       | 0.0000          | nan
 18       | 0.0000          | nan
 19       | 0.0000          | nan
 20       | 0.0000          | nan
 21       | 0.0000          | nan
 22       | 0.0000          | nan
 23       | 0.0000          | nan
 24       | 0.0000          | nan
 25       | 0.0000          | nan
 26       | 0.0000          | nan
 27       | 0.0000          | nan
 28       | 0.0000          | nan
 29       | 0.0000          | nan
 30       | 0.0000          | nan
 31       | 0.0000          | nan
 32       | 0.0000          | nan
 33       | 0.0000          | nan
 34       | 0.0000          | nan
 35       | 0.0000          | nan
 36       | 0.0000          | nan
 37       | 0.0000          | nan
 38       | 0.0000          | nan
 39       | 0.0000          | nan
 40       | 0.0000          | nan
 41       | 0.0000          | nan
 42       | 0.0000          | nan
 43       | 0.0000          | nan
 44       | 0.0000          | nan
 45       | 0.0000          | nan
 46       | 0.0000          | nan
 47       | 0.0000          | nan
 48       | 0.0000          | nan
 49       | 0.0000          | nan
 50       | 0.0000          | nan
 51       | 0.0000          | nan
 52       | 0.0000          | nan
 53       | 0.0000          | nan
```

```
54       | 0.0000         | nan
55       | 0.0000         | nan
56       | 0.0000         | nan
57       | 0.0000         | nan
58       | 0.0000         | nan
59       | 0.0000         | nan
60       | 0.0000         | nan
61       | 0.0000         | nan
62       | 0.0000         | nan
63       | 0.0000         | nan
64       | 0.0000         | nan
65       | 0.0000         | nan
66       | 0.0000         | nan
67       | 0.0000         | nan
68       | 0.0000         | nan
69       | 0.0000         | nan
70       | 0.0000         | nan
71       | 0.0000         | nan
72       | 0.0000         | nan
73       | 0.0000         | nan
74       | 0.0000         | nan
75       | 0.0000         | nan
76       | 0.0000         | nan
77       | 0.0000         | nan
78       | 0.0000         | nan
79       | 0.0000         | nan
80       | 0.0000         | nan
81       | 0.0000         | nan
82       | 0.0000         | nan
83       | 0.0000         | nan
84       | 0.0000         | nan
85       | 0.0000         | nan
86       | 0.0000         | nan
87       | 0.0000         | nan
88       | 0.0000         | nan
89       | 0.0000         | nan
90       | 0.0000         | nan
91       | 0.0000         | nan
92       | 0.0000         | nan
93       | 0.0000         | nan
94       | 0.0000         | nan
95       | 0.0000         | nan
96       | 0.0000         | nan
97       | 0.0000         | nan
98       | 0.0000         | nan
99       | 0.0000         | nan
```

```
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:8: RuntimeWarning: overf
low encountered in scalar power
  df_dx = 40 * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:9: RuntimeWarning: overf
low encountered in scalar power
  df_dy = 2 * (y - 3) - 80 * (y - 3) * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:8: RuntimeWarning: inval
id value encountered in scalar subtract
  df_dx = 40 * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:9: RuntimeWarning: inval
id value encountered in scalar subtract
  df_dy = 2 * (y - 3) - 80 * (y - 3) * ((x + 3) - (y - 3)**2)
```

In [23]:

```python
# Perform gradient descent with eta = 0.01
eta_new = 0.01
values_f1_new = gradient_descent(grad_f1, initial_point, eta_new, T)
values_f2_new = gradient_descent(grad_f2, initial_point, eta_new, T)

# Check the size of the lists
print(f"Length of values_f1_new: {len(values_f1_new)}")
print(f"Length of values_f2_new: {len(values_f2_new)}")

# Ensure the loop doesn't exceed the length of the list
iterations = min(len(values_f1_new), len(values_f2_new))

# Report f1(x, y) and f2(x, y) values over iterations with new learning rate
print(f"\nUsing learning rate eta = 0.01:")
print(f"{'Iteration':>10} | {'f1(x, y)':>10} | {'f2(x, y)':>10}")
for t in range(iterations):
    x1, y1 = values_f1_new[t]
    x2, y2 = values_f2_new[t]
    print(f"{t:>10} | {f1(x1, y1):>10.4f} | {f2(x2, y2):>10.4f}")
```

```
Length of values_f1_new: 100
Length of values_f2_new: 100

Using learning rate eta = 0.01:
 Iteration |   f1(x, y) |   f2(x, y)
         0 |    12.4852 | 317284.7692
         1 |    11.9908 | 34170326220204.5781
         2 |    11.5160 | 40858527829439322292549606533402460160.0000
         3 |    11.0599 | 698470551352992560533122819350193726096551034998755994360761
628068834843190354818187609708374626572986671 43168.0000
         4 |    10.6219 |        inf
         5 |    10.2013 |        inf
         6 |     9.7973 |        nan
         7 |     9.4094 |        nan
         8 |     9.0368 |        nan
         9 |     8.6789 |        nan
        10 |     8.3352 |        nan
        11 |     8.0051 |        nan
        12 |     7.6881 |        nan
        13 |     7.3837 |        nan
        14 |     7.0913 |        nan
        15 |     6.8105 |        nan
        16 |     6.5408 |        nan
        17 |     6.2818 |        nan
        18 |     6.0330 |        nan
        19 |     5.7941 |        nan
        20 |     5.5647 |        nan
        21 |     5.3443 |        nan
        22 |     5.1327 |        nan
        23 |     4.9294 |        nan
        24 |     4.7342 |        nan
        25 |     4.5467 |        nan
        26 |     4.3667 |        nan
        27 |     4.1938 |        nan
        28 |     4.0277 |        nan
        29 |     3.8682 |        nan
        30 |     3.7150 |        nan
        31 |     3.5679 |        nan
        32 |     3.4266 |        nan
        33 |     3.2909 |        nan
        34 |     3.1606 |        nan
        35 |     3.0354 |        nan
        36 |     2.9152 |        nan
        37 |     2.7998 |        nan
        38 |     2.6889 |        nan
        39 |     2.5824 |        nan
        40 |     2.4802 |        nan
        41 |     2.3820 |        nan
        42 |     2.2876 |        nan
        43 |     2.1970 |        nan
        44 |     2.1100 |        nan
        45 |     2.0265 |        nan
        46 |     1.9462 |        nan
        47 |     1.8692 |        nan
        48 |     1.7951 |        nan
        49 |     1.7241 |        nan
```

```
   50 |     1.6558 |         nan
   51 |     1.5902 |         nan
   52 |     1.5272 |         nan
   53 |     1.4668 |         nan
   54 |     1.4087 |         nan
   55 |     1.3529 |         nan
   56 |     1.2993 |         nan
   57 |     1.2479 |         nan
   58 |     1.1985 |         nan
   59 |     1.1510 |         nan
   60 |     1.1054 |         nan
   61 |     1.0616 |         nan
   62 |     1.0196 |         nan
   63 |     0.9792 |         nan
   64 |     0.9404 |         nan
   65 |     0.9032 |         nan
   66 |     0.8674 |         nan
   67 |     0.8331 |         nan
   68 |     0.8001 |         nan
   69 |     0.7684 |         nan
   70 |     0.7380 |         nan
   71 |     0.7088 |         nan
   72 |     0.6807 |         nan
   73 |     0.6537 |         nan
   74 |     0.6278 |         nan
   75 |     0.6030 |         nan
   76 |     0.5791 |         nan
   77 |     0.5562 |         nan
   78 |     0.5342 |         nan
   79 |     0.5130 |         nan
   80 |     0.4927 |         nan
   81 |     0.4732 |         nan
   82 |     0.4544 |         nan
   83 |     0.4364 |         nan
   84 |     0.4192 |         nan
   85 |     0.4026 |         nan
   86 |     0.3866 |         nan
   87 |     0.3713 |         nan
   88 |     0.3566 |         nan
   89 |     0.3425 |         nan
   90 |     0.3289 |         nan
   91 |     0.3159 |         nan
   92 |     0.3034 |         nan
   93 |     0.2914 |         nan
   94 |     0.2798 |         nan
   95 |     0.2688 |         nan
   96 |     0.2581 |         nan
   97 |     0.2479 |         nan
   98 |     0.2381 |         nan
   99 |     0.2286 |         nan
```

```
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:8: RuntimeWarning: overf
low encountered in scalar power
  df_dx = 40 * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:9: RuntimeWarning: overf
low encountered in scalar power
  df_dy = 2 * (y - 3) - 80 * (y - 3) * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:8: RuntimeWarning: inval
id value encountered in scalar subtract
  df_dx = 40 * ((x + 3) - (y - 3)**2)
C:\Users\DeLL\AppData\Local\Temp\ipykernel_19620\148922353.py:9: RuntimeWarning: inval
id value encountered in scalar subtract
  df_dy = 2 * (y - 3) - 80 * (y - 3) * ((x + 3) - (y - 3)**2)
```

Gradient Descent: Why?

In fact, Gradient descent can shift input values in the direction of the steepest descent of the function (the negative gradient) allowing for iterations. More specifically, it is helpful in functions characterized by a wide array of local minima or non-linearity since this provides room for gradual convergence towards a minimum assuming proper tuning of the learning rate takes place.

Influence of Learning Rate: During gradient descent, the learning rate (η) governs how far one can go at each time step. If it’s too great, one could overshoot and miss out on finding the minimum because they are too high up their curve; if it’s too low, however, one may not have enough energy to move much closer to it at all thereby resulting into a very slow convergence process.