In [1]:

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
boston = load_boston()
```

In [2]:

```
print(boston.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [4]:

```
print(boston.data.shape)
```

```
(506, 13)
```

In [5]:

```
print(boston.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [6]:

```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (at
tribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 s
q.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by t
own
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegi
e Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
```

```
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that
address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential D
ata and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-24
3, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [19]:

```python
boston_pd = pd.DataFrame(boston.data)

print(boston_pd.head())
```

```
         0     1     2    3      4      5     6       7    8      9     10  \
0  0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0  15.3
1  0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0  17.8
2  0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0  17.8
3  0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0  18.7
4  0.06905   0.0  2.18  0.0  0.458  7.147  54.2  6.0622  3.0  222.0  18.7

       11    12
0  396.90  4.98
1  396.90  9.14
2  392.83  4.03
3  394.63  2.94
4  396.90  5.33
```

In [20]:

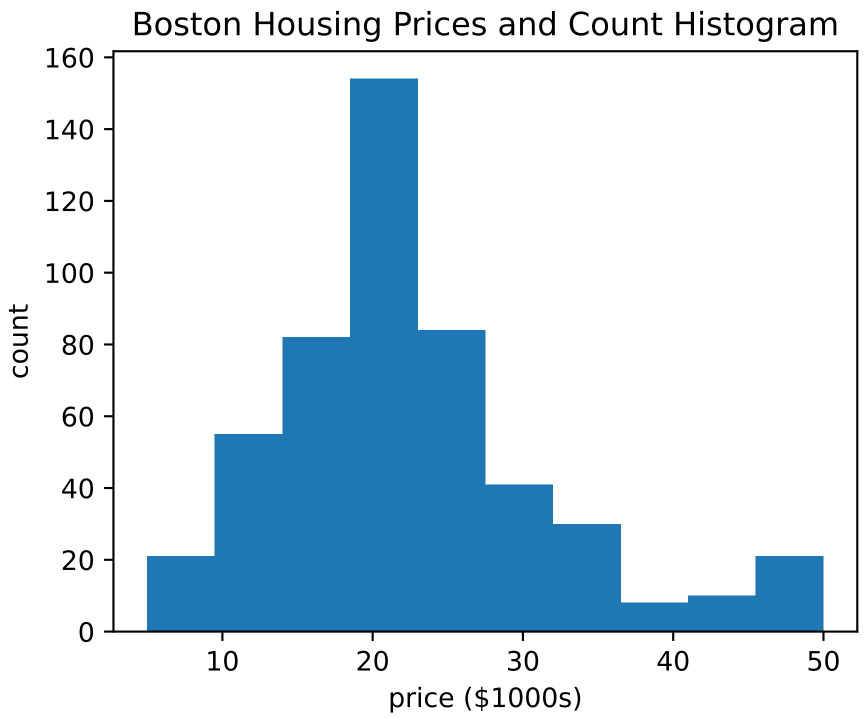
```python
boston_pd.columns = boston.feature_names
print(boston_pd.head())
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

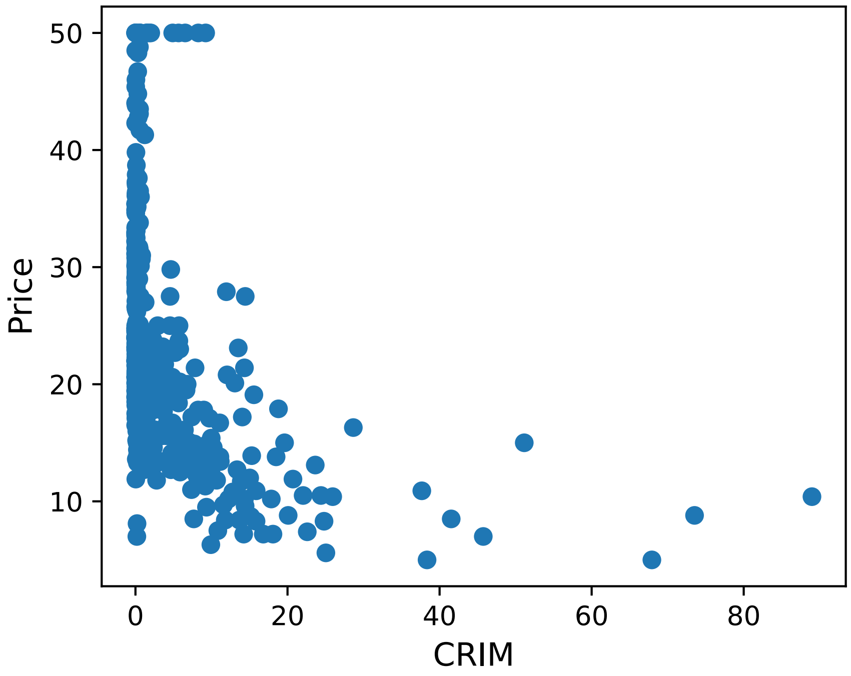
In [8]:

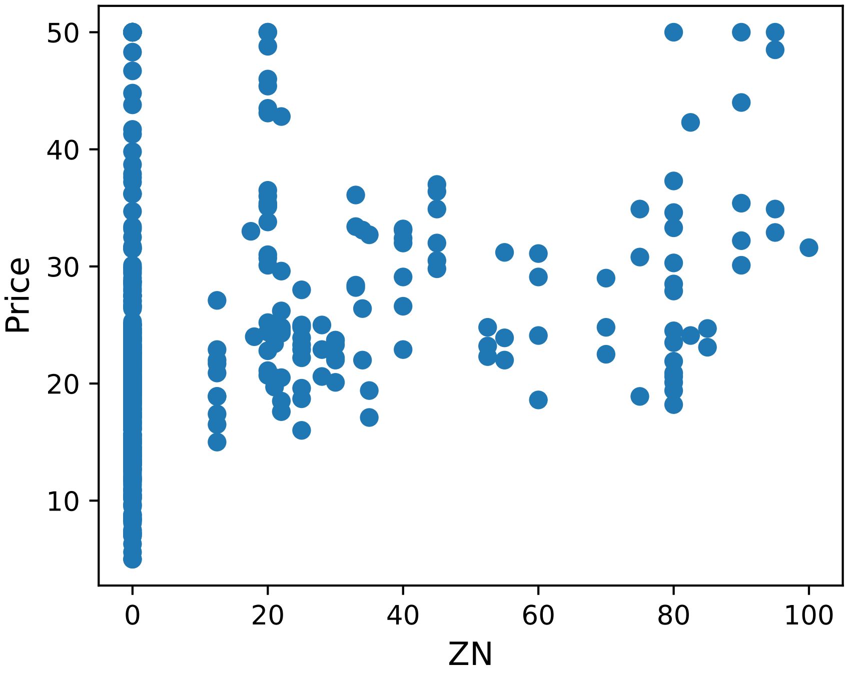
```python
from sklearn.datasets import load_boston
import pandas as pd
import matplotlib.pyplot as plt
boston = load_boston()
plt.figure(figsize=(5, 4))
plt.hist(boston.target)
plt.title('Boston Housing Prices and Count Histogram')
plt.xlabel('price ($1000s)')
plt.ylabel('count')
plt.show()
```

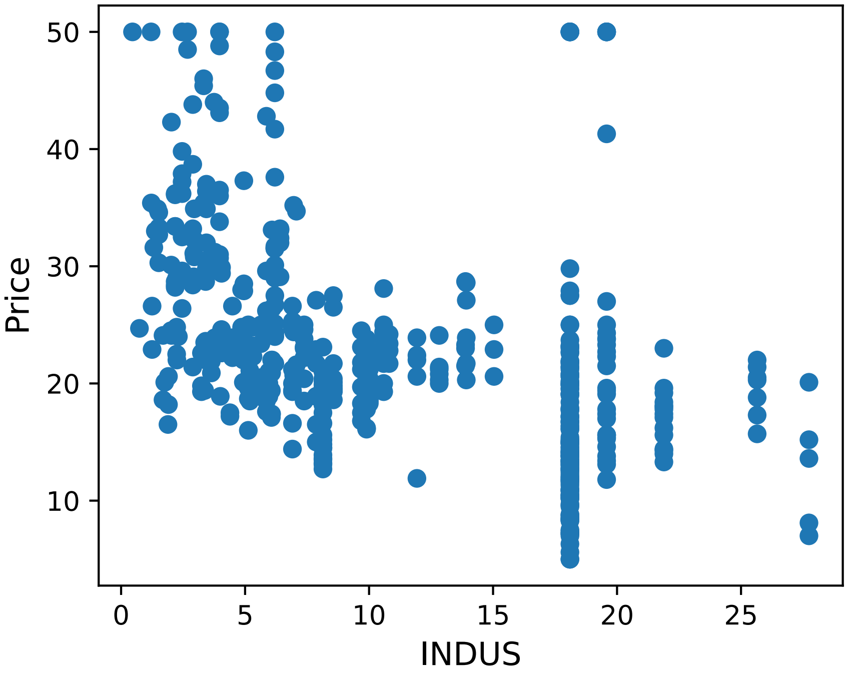


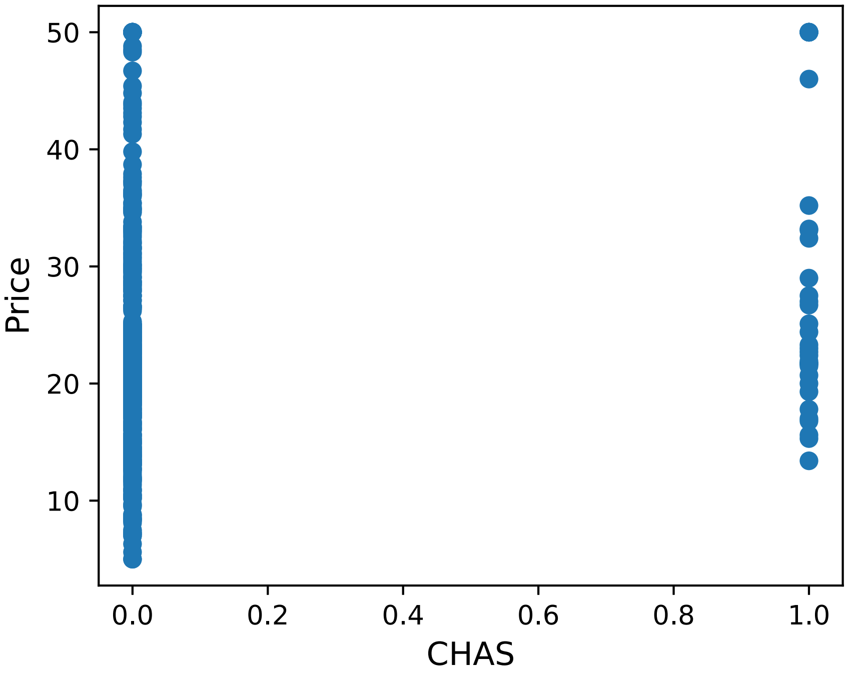
In [9]:

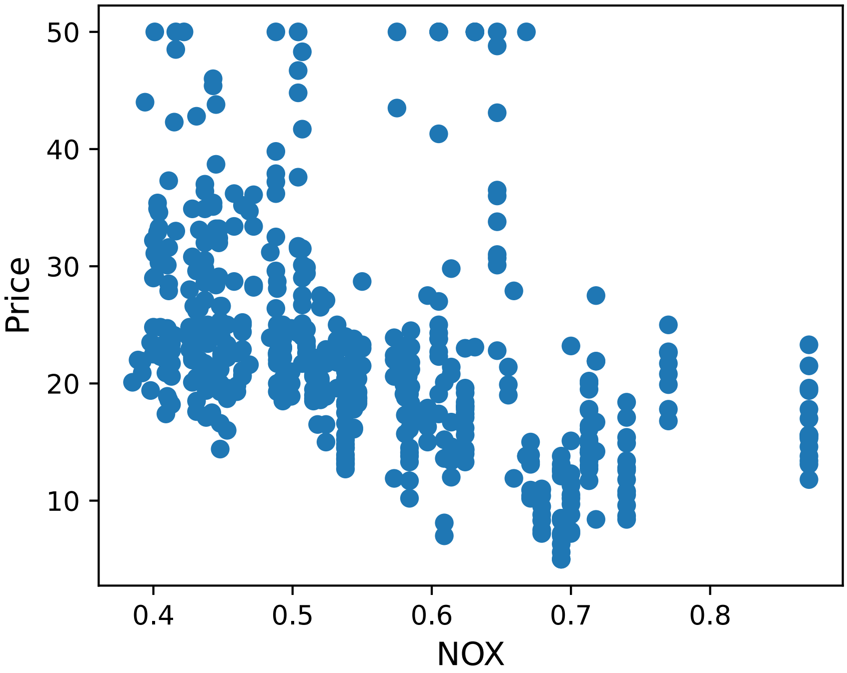
```
for index, feature_name in enumerate(boston.feature_names):
    plt.figure(figsize=(5, 4))
    plt.scatter(boston.data[:, index], boston.target)
    plt.ylabel('Price', size=12)
    plt.xlabel(feature_name, size=12)
    plt.show()
```

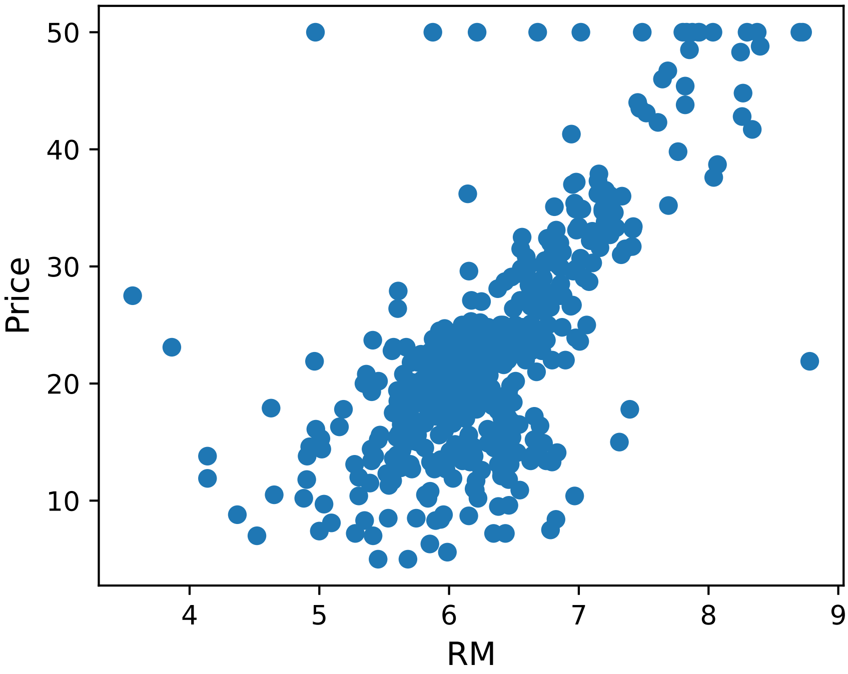


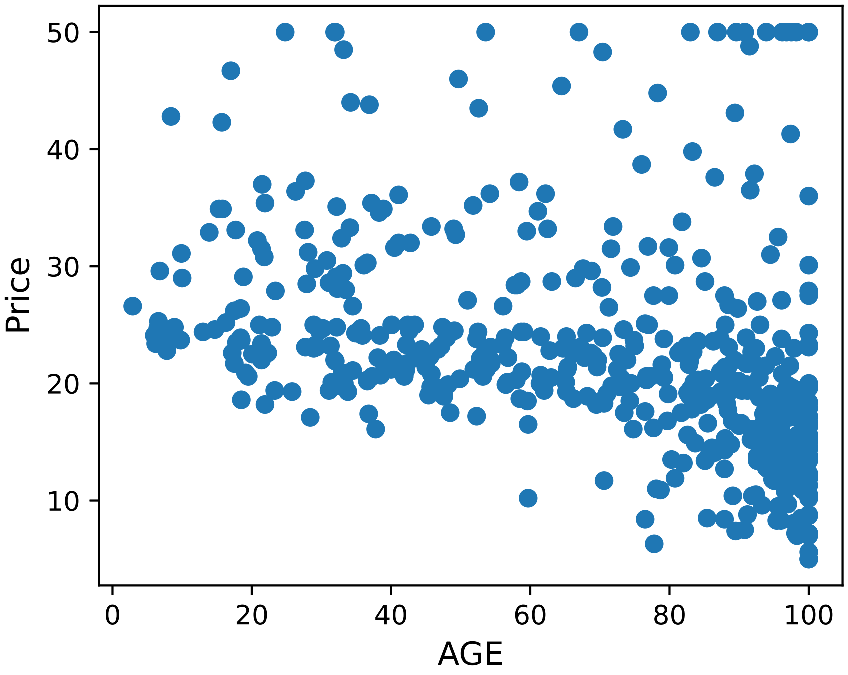

Price
50
40
30
20
10
0
20
40
60
80
100
ZN

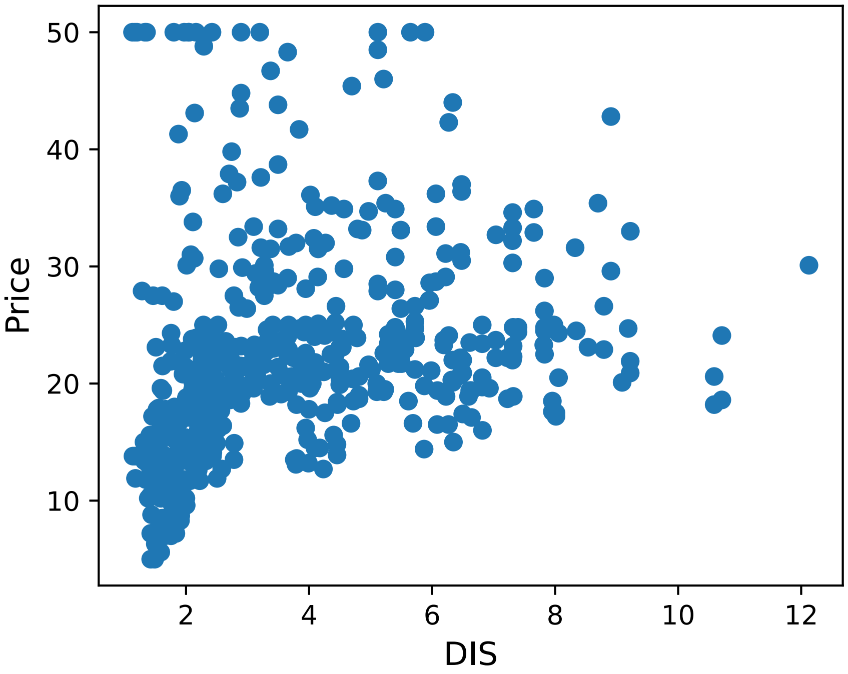

Price
50
40
30
20
10
0
5
10
15
20
25
INDUS

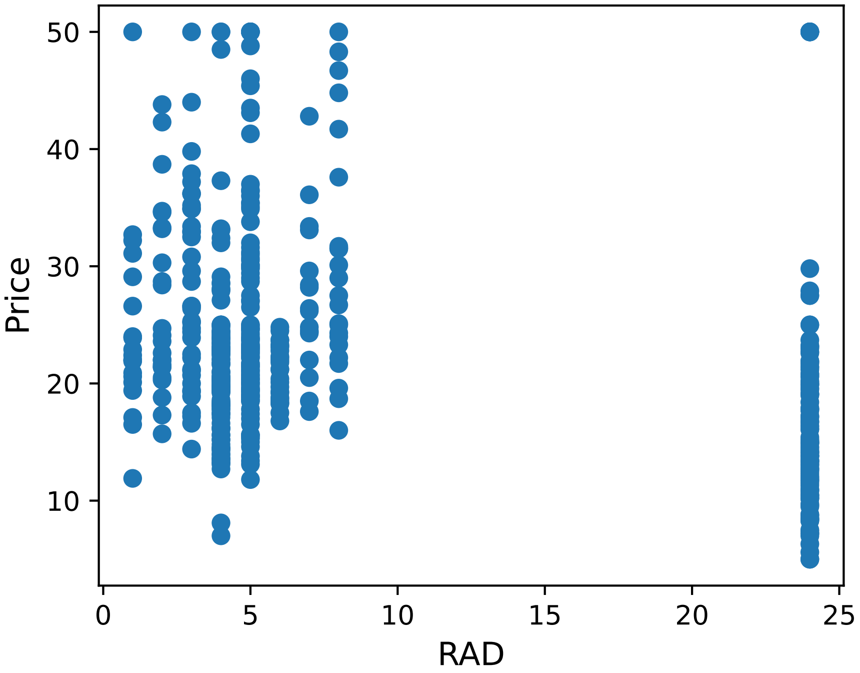

Price
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0
CHAS

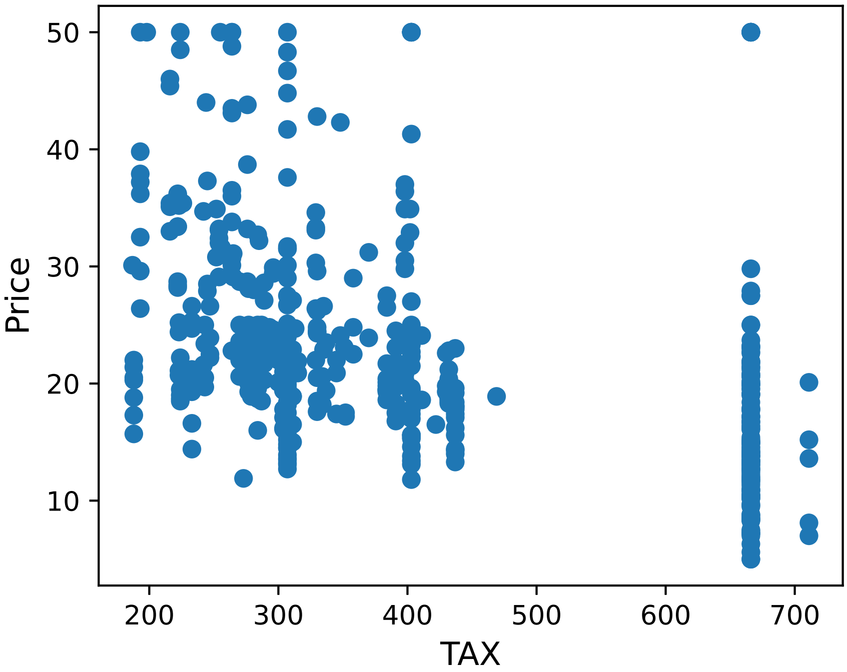

Price
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8
NOX

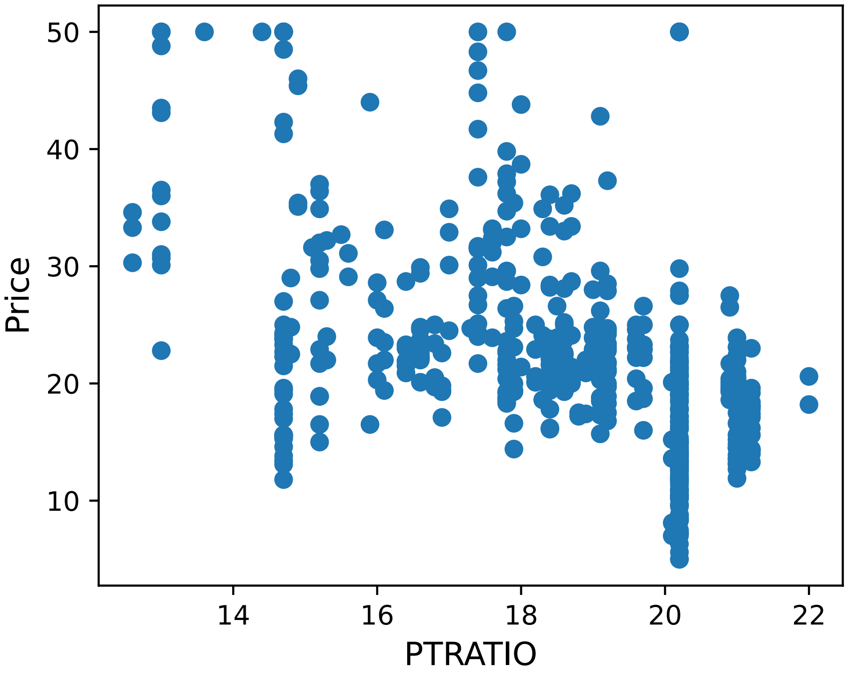

Price
50
40
30
20
10
4
5
6
7
8
9
RM

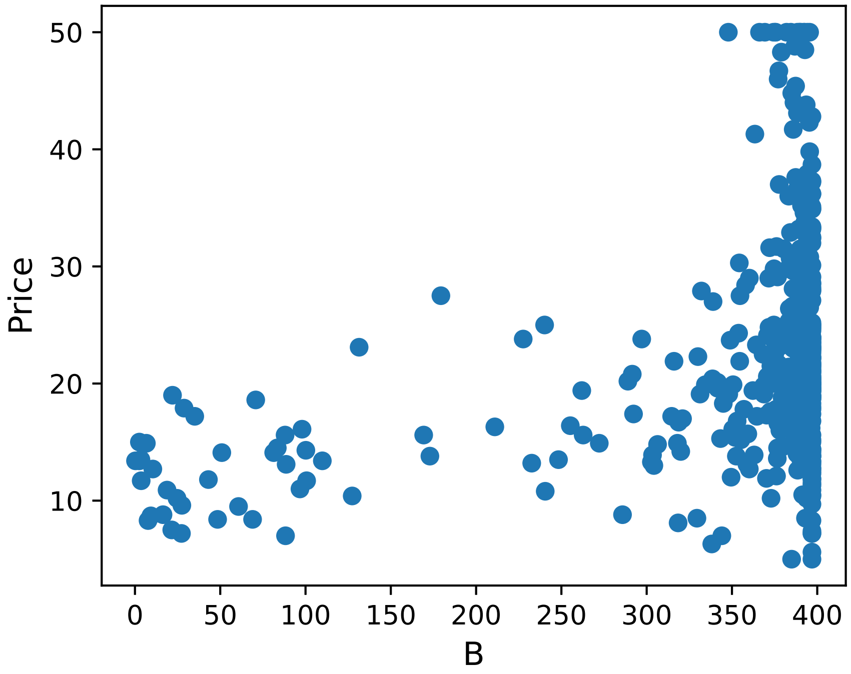

Price
50
40
30
20
10
0
20
40
60
80
100
AGE

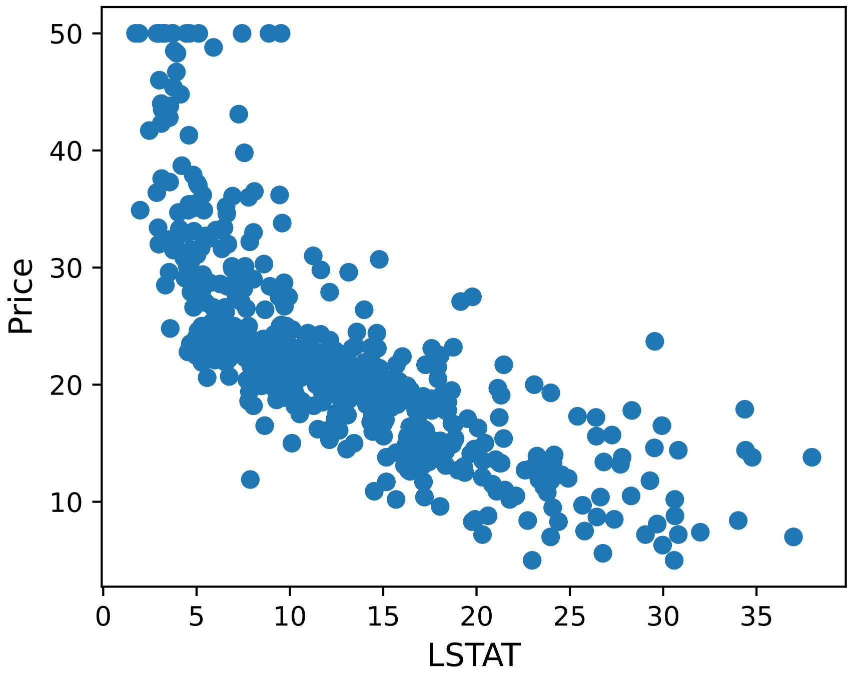

Price
50
40
30
20
10
2
4
6
8
10
12
DIS


Price
50
40
30
20
10
0
5
10
15
20
25
RAD


Price
50
40
30
20
10
200
300
400
500
600
700
TAX


Price
50
40
30
20
10
14
16
18
20
22
PTRATIO

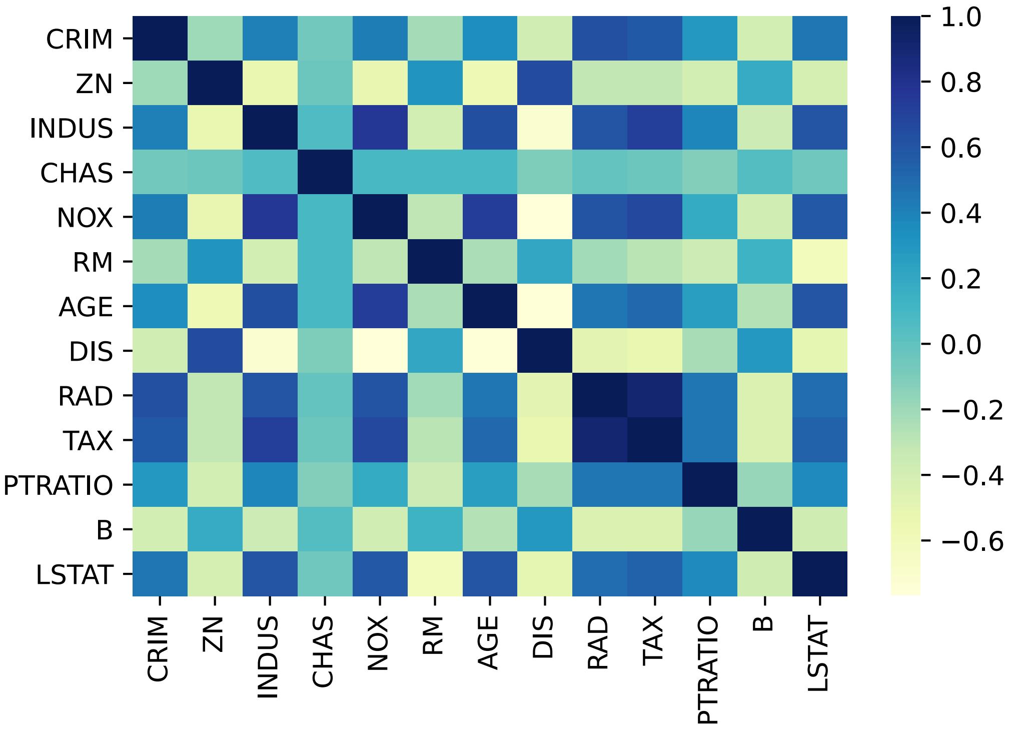




In [10]:

```
from sklearn.datasets import load_boston
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

boston = load_boston()
boston_pd = pd.DataFrame(boston.data, columns = boston.feature_names)
correlation_matrix = boston_pd.corr().round(2)
sns.heatmap(correlation_matrix, cmap="YlGnBu")
plt.show()
```

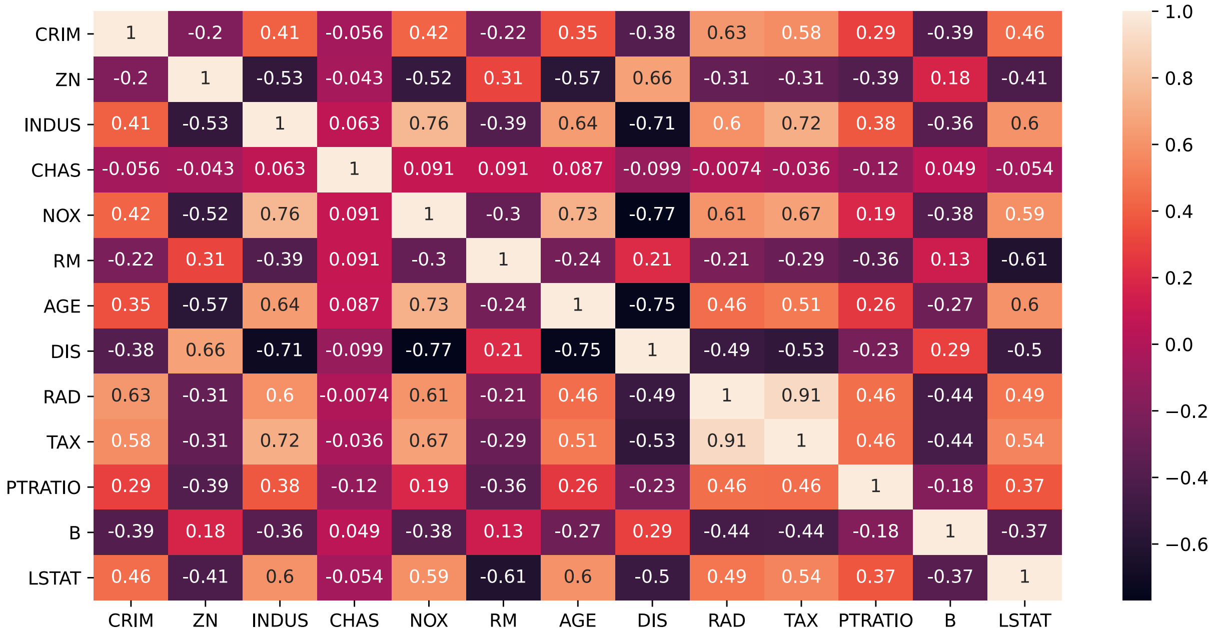


In [14]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
sns.heatmap(boston_pd.corr(), annot=True)
```

Out[14]:

```
<AxesSubplot:>
```



In [29]:

```
sns.jointplot(data=boston_pd, x="DIS", y="Price", kind="hex", color="m");
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-29-bf81801feecd> in <module>
----> 1 sns.jointplot(data=boston_pd, x="DIS", y="Price", kind="hex", color=
"m");

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py in inner_f
(*args, **kwargs)
```

```
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)}
)
---> 46         return f(**kwargs)
     47     return inner_f
     48 

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/axisgrid.py in jointplot
(x, y, data, kind, color, height, ratio, space, dropna, xlim, ylim, marginal_t
icks, joint_kws, marginal_kws, hue, palette, hue_order, hue_norm, **kwargs)
   2119 
   2120     # Initialize the JointGrid object
-> 2121     grid = JointGrid(
   2122         data=data, x=x, y=y, hue=hue,
   2123         palette=palette, hue_order=hue_order, hue_norm=hue_norm,

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py in inner_f
(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)}
)
---> 46         return f(**kwargs)
     47     return inner_f
     48 

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/axisgrid.py in __init__(se
lf, x, y, data, height, ratio, space, dropna, xlim, ylim, size, marginal_tick
s, hue, palette, hue_order, hue_norm)
   1628 
   1629         # Process the input variables
-> 1630         p = VectorPlotter(data=data, variables=dict(x=x, y=y, hue=hue)
)
   1631         plot_data = p.plot_data.loc[:, p.plot_data.notna().any()]
   1632 

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_core.py in __init__(self,
data, variables)
    602     def __init__(self, data=None, variables={}):
    603 
--> 604         self.assign_variables(data, variables)
    605 
    606         for var, cls in self._semantic_mappings.items():

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_core.py in assign_variabl
es(self, data, variables)
    665         else:
    666             self.input_format = "long"
--> 667             plot_data, variables = self._assign_variables_longform(
    668                 data, **variables,
    669             )

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_core.py in _assign_variab
les_longform(self, data, **kwargs)
    900 
    901                 err = f"Could not interpret value `{val}` for paramete
r `{key}`"
--> 902                 raise ValueError(err)
    903 
    904             else:

ValueError: Could not interpret value `Price` for parameter `y`
```

In [28]:

```
sns.distplot(boston_pd["RM", color="g"])
```

```
  File "<ipython-input-28-38955e9141eb>", line 1
    sns.distplot(boston_pd["RM", color="g"])
                                      ^
SyntaxError: invalid syntax
```

In [ ]: