In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
import scipy.stats as stats
```

In [4]:

```
import matplotlib.pyplot as plt
```

In [5]:

```
import sklearn
```

In [6]:

```
import seaborn as sns
```

In [7]:

```
from matplotlib import rcParams
```

In [8]:

```
sns.set_style("whitegrid")
```

In [9]:

```
sns.set_context("poster")
```

In [10]:

```
from sklearn.datasets import load_boston
```

In [11]:

```
boston = load_boston()
```

In [12]:

```
boston.keys()
```

Out[12]:

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

In [13]:

```
boston.data.shape
```

Out[13]:

```
(506, 13)
```

In [14]:

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

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

In [15]:

```
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
(attribute 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
sq.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
town
        - 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/ (https://
archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Carne
gie 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 diagnostic
s
...', 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 th
at address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential
```

```
Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. I
n Proceedings on the Tenth International Conference of Machine Learning, 236
-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [16]:

```
bos = pd.DataFrame(boston.data)
```

In [17]:

```
bos.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [18]:

```
bos.columns = boston.feature_names
```

In [19]:

```
bos.head()
```

Out[19]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5

In [20]:

```
print (boston.target.shape)
```

```
(506,)
```

In [21]:

```
bos['PRICE'] = boston.target
```

In [22]:

```
bos.head()
```

Out[22]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5

In [24]:

```
bos.describe()
```

Out[24]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	50
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	1.

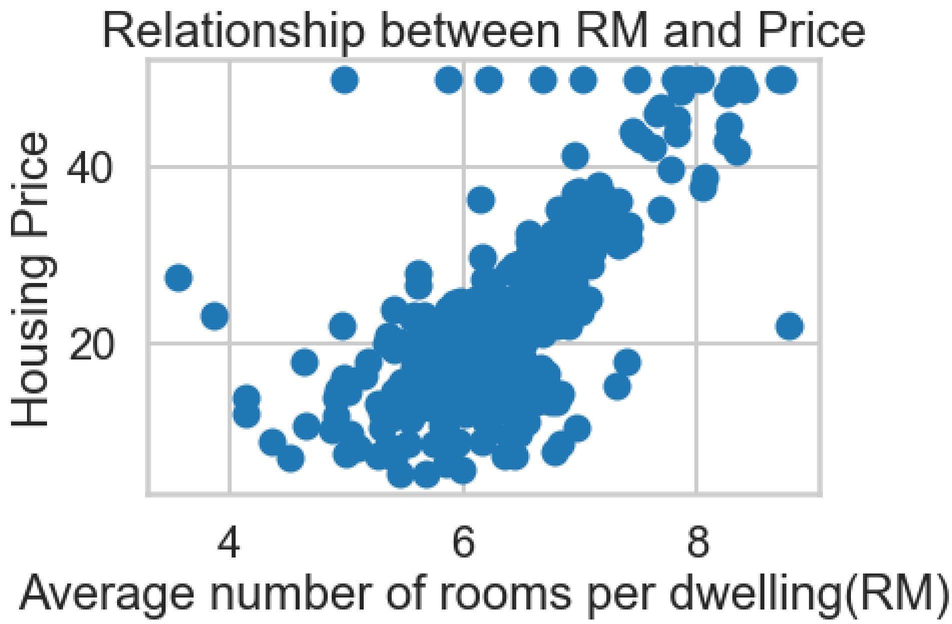
In [ ]:

In [29]:

```
plt.scatter(bos.RM, bos.PRICE)
plt.xlabel("Average number of rooms per dwelling(RM)")
plt.ylabel("Housing Price")
plt.title("Relationship between RM and Price")
```

Out[29]:

```
Text(0.5, 1.0, 'Relationship between RM and Price')
```

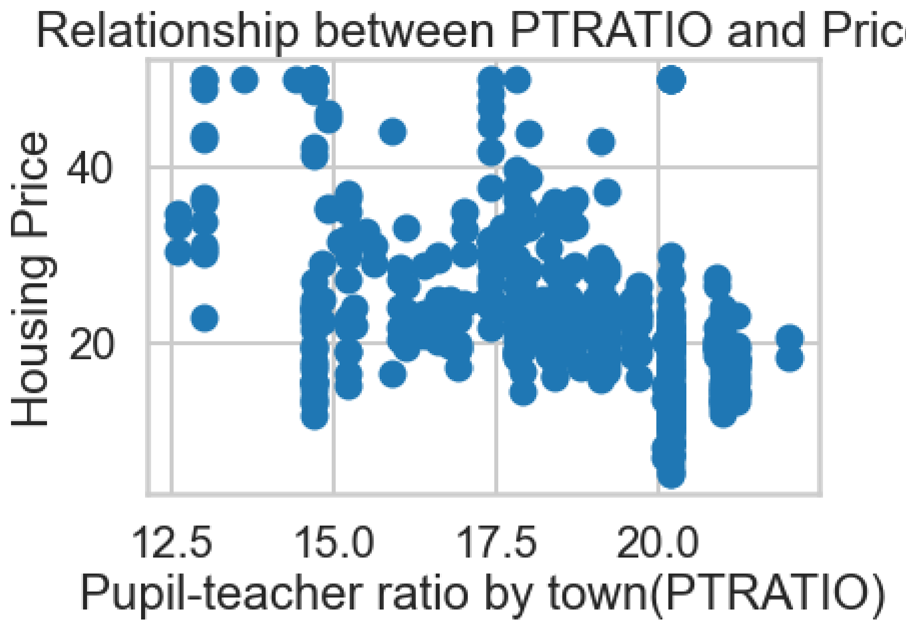


In [30]:

```
plt.scatter(bos.PTRATIO, bos.PRICE)
plt.xlabel("Pupil-teacher ratio by town(PTRATIO)")
plt.ylabel("Housing Price")
plt.title("Relationship between PTRATIO and Price")
```

Out[30]:

```
Text(0.5, 1.0, 'Relationship between PTRATIO and Price')
```

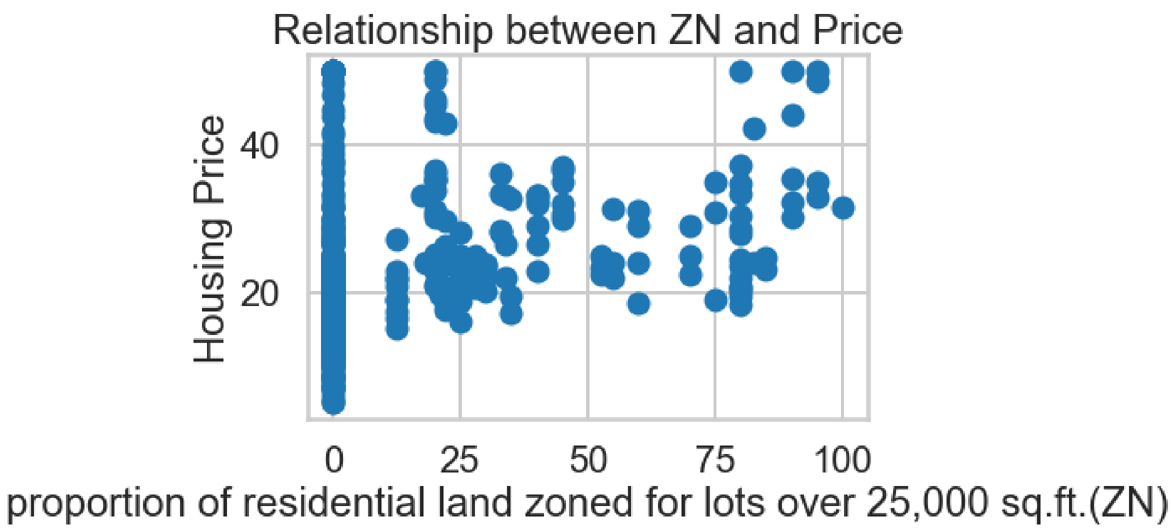


In [31]:

```
plt.scatter(bos.ZN, bos.PRICE)
plt.xlabel("proportion of residential land zoned for lots over 25,000 sq.ft.(ZN)")
plt.ylabel("Housing Price")
plt.title("Relationship between ZN and Price")
```

Out[31]:

```
Text(0.5, 1.0, 'Relationship between ZN and Price')
```

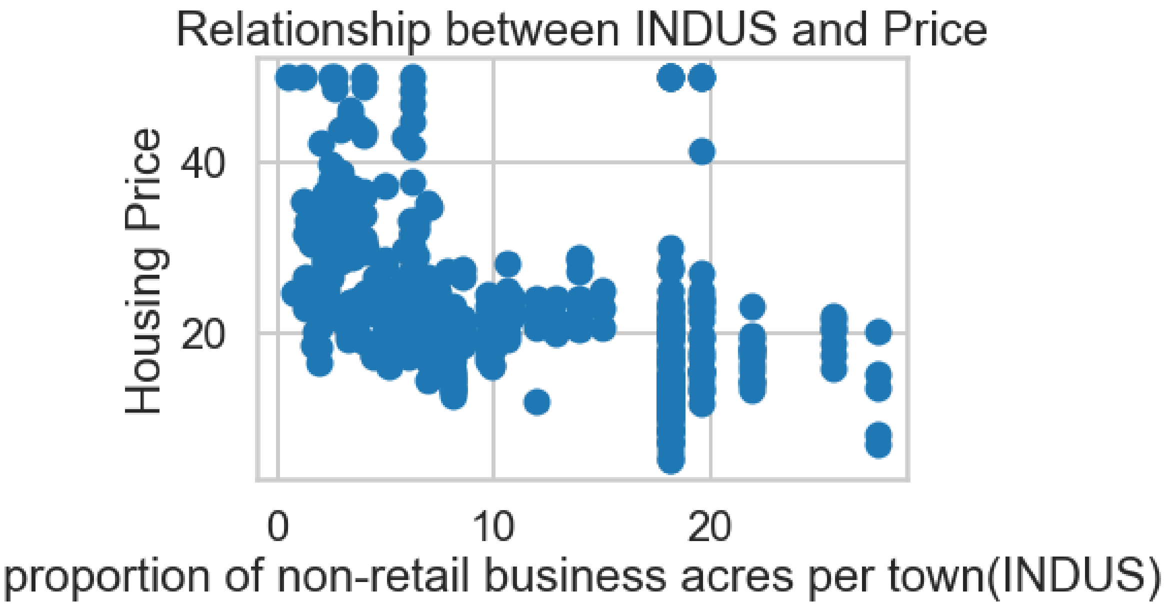


In [32]:

```
plt.scatter(bos.INDUS, bos.PRICE)
plt.xlabel("proportion of non-retail business acres per town(INDUS)")
plt.ylabel("Housing Price")
plt.title("Relationship between INDUS and Price")
```

Out[32]:

```
Text(0.5, 1.0, 'Relationship between INDUS and Price')
```

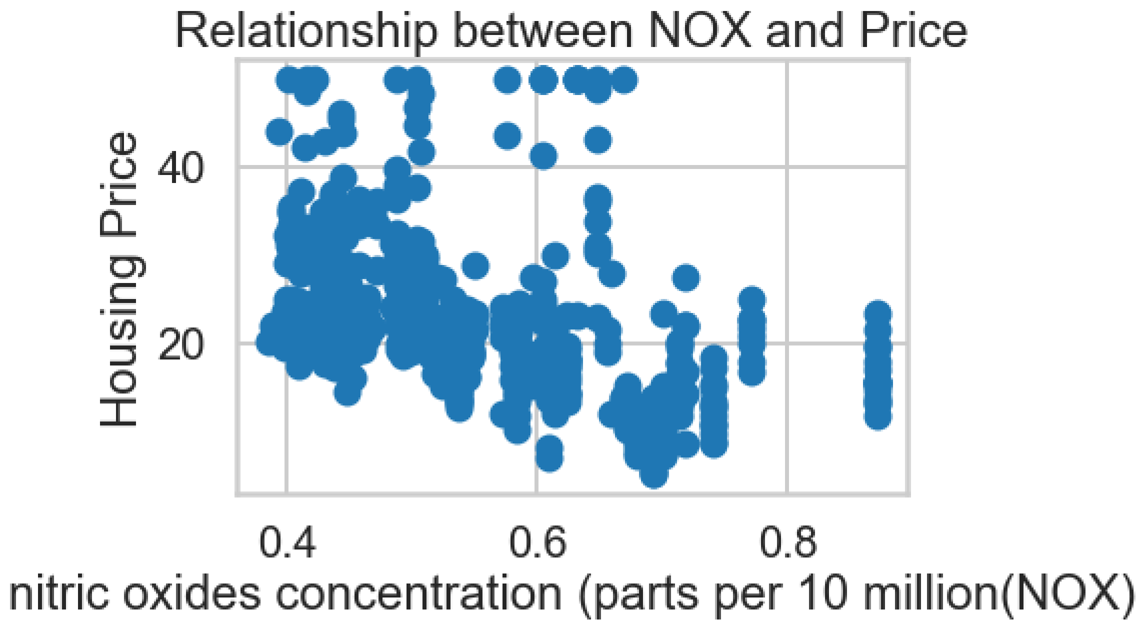


In [33]:

```
plt.scatter(bos.NOX, bos.PRICE)
plt.xlabel("nitric oxides concentration (parts per 10 million(NOX)")
plt.ylabel("Housing Price")
plt.title("Relationship between NOX and Price")
```

Out[33]:

```
Text(0.5, 1.0, 'Relationship between NOX and Price')
```

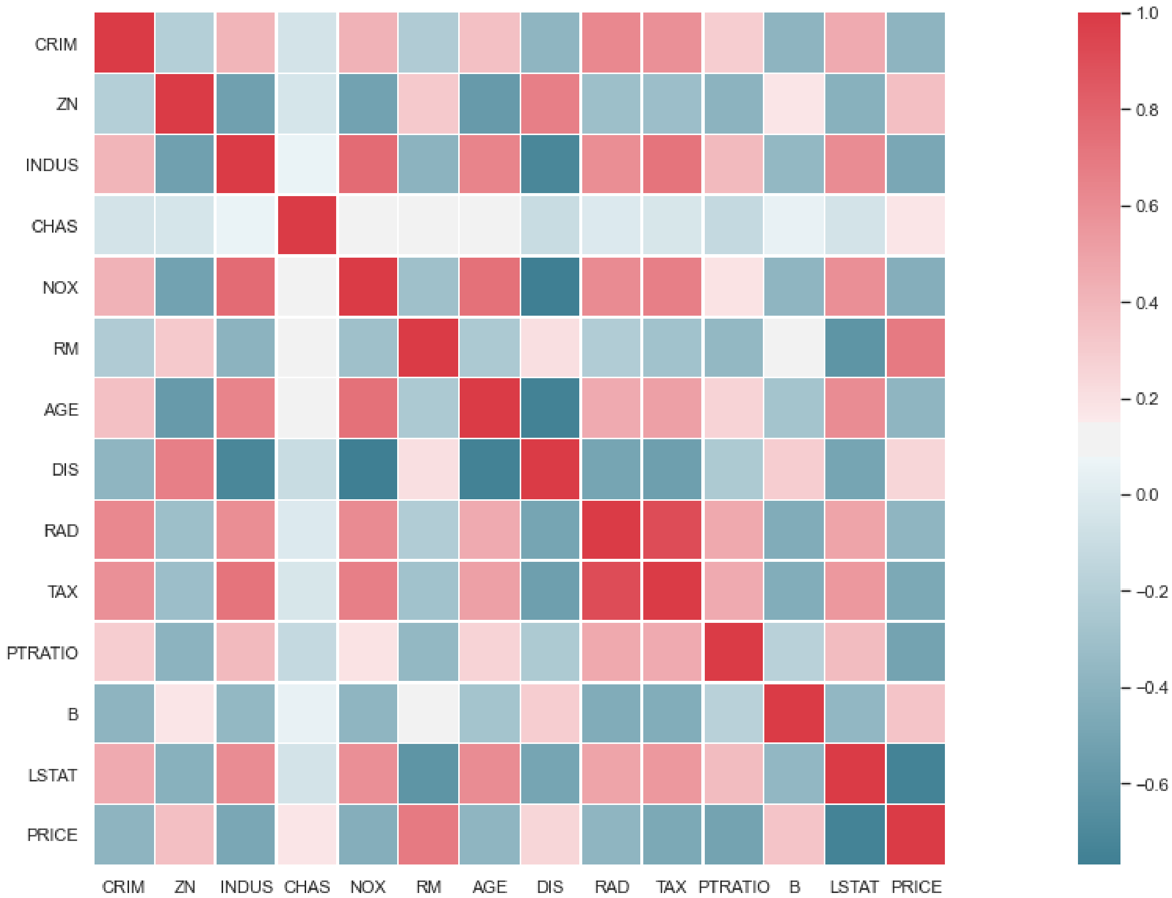


In [34]:

```
sns.set(style="white")

df_corr= bos[:]
# Compute the correlation matrix
corr = df_corr.dropna().corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
#mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(30, 10))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, square=True, linewidths=.5, ax=ax)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1926da994c0>
```

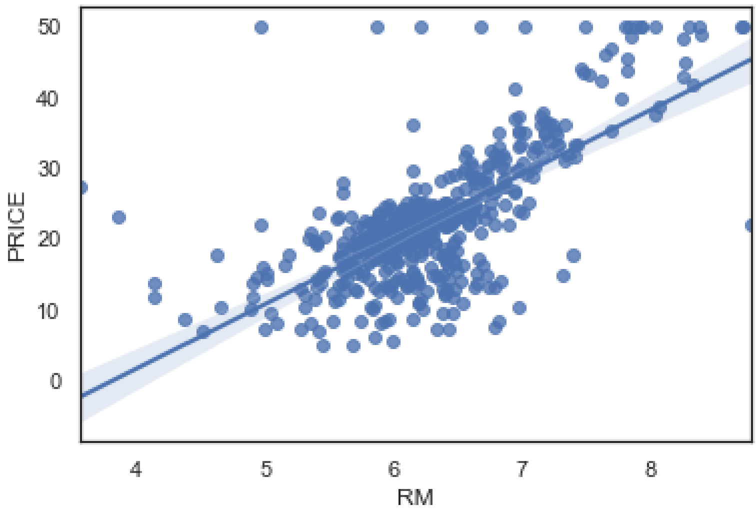


In [35]:

```
sns.regplot(y="PRICE", x="RM", data=bos, fit_reg = True)
```

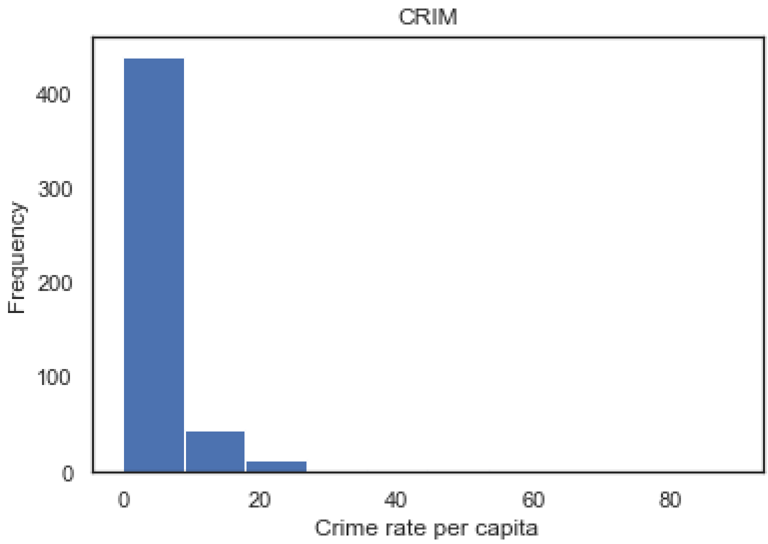
Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1926e1bb3d0>
```



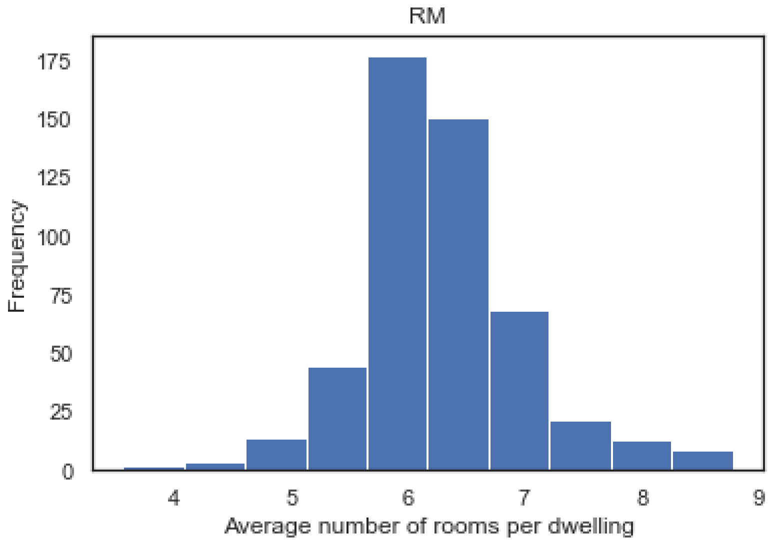
In [36]:

```
plt.hist(bos.CRIM)
plt.title("CRIM")
plt.xlabel("Crime rate per capita")
plt.ylabel("Frequency")
plt.show()
```



In [37]:

```
plt.hist(bos.RM)
plt.title("RM")
plt.xlabel("Average number of rooms per dwelling")
plt.ylabel("Frequency")
plt.show()

bos.RM.describe()
```

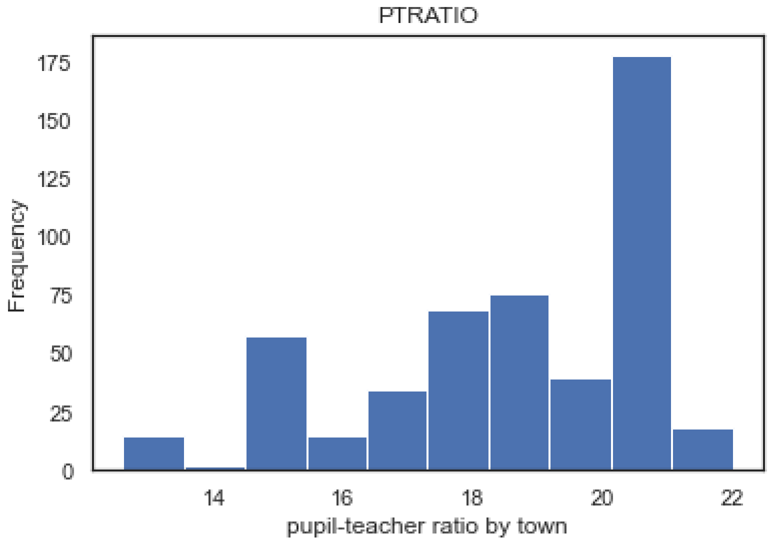


Out[37]:

```
count    506.000000
mean       6.284634
std        0.702617
min        3.561000
25%        5.885500
50%        6.208500
75%        6.623500
max        8.780000
Name: RM, dtype: float64
```

In [38]:

```
plt.hist(bos.PTRATIO)
plt.title("PTRATIO")
plt.xlabel("pupil-teacher ratio by town")
plt.ylabel("Frequency")
plt.show()
```



In [39]:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [40]:

```
m = ols('PRICE ~ RM',bos).fit()
print (m.summary())
```

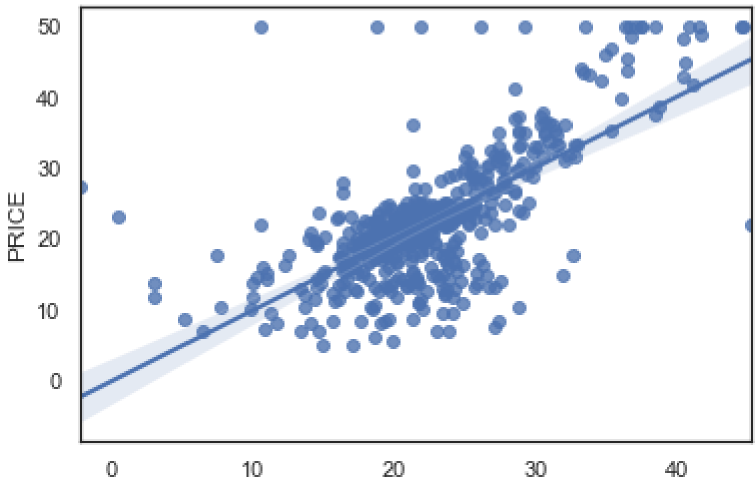
```
                            OLS Regression Results
=============================================================================
==
Dep. Variable:                  PRICE   R-squared:                       0.4
84
Model:                            OLS   Adj. R-squared:                  0.4
83
Method:                 Least Squares   F-statistic:                      47
1.8
Date:                Tue, 09 Nov 2021   Prob (F-statistic):             2.49e-
74
Time:                        16:36:52   Log-Likelihood:                -167
3.1
No. Observations:                 506   AIC:                             335
0.
Df Residuals:                     504   BIC:                             335
9.
Df Model:                           1
Covariance Type:            nonrobust
=============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
-----------------------------------------------------------------------------
--
Intercept    -34.6706      2.650    -13.084      0.000     -39.877     -29.4
65
RM             9.1021      0.419     21.722      0.000       8.279       9.9
25
=============================================================================
==
Omnibus:                      102.585   Durbin-Watson:                   0.6
84
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              612.4
49
Skew:                           0.726   Prob(JB):                     1.02e-1
33
Kurtosis:                       8.190   Cond. No.                          5
8.4
=============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [41]:

```
fdval= m.fittedvalues
sns.regplot(x=fdval, y="PRICE", data=bos, fit_reg = True)
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1926ebeba60>
```



In [45]:

```
from sklearn.linear_model import LinearRegression
X = bos.drop('PRICE', axis = 1)

# This creates a LinearRegression object
lm = LinearRegression()
lm.fit(X, bos.PRICE)
```

Out[45]:

```
LinearRegression()
```

In [46]:

```
print ('Estimated intercept coefficient:', lm.intercept_)
```

```
Estimated intercept coefficient: 36.45948838509015
```

In [47]:

```
print ('Number of coefficients:', len(lm.coef_))
```

```
Number of coefficients: 13
```

In [48]:

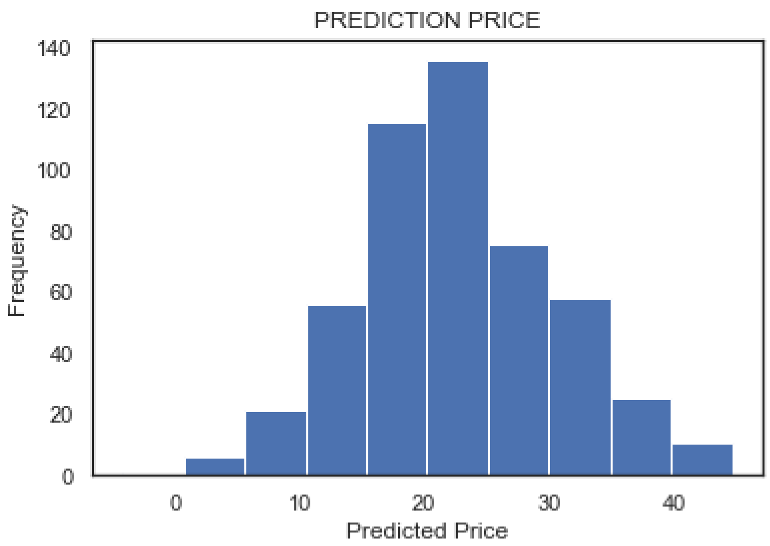
```
lm.predict(X)[0:5]
```

Out[48]:

```
array([30.00384338, 25.02556238, 30.56759672, 28.60703649, 27.94352423])
```

In [49]:

```
plt.hist(lm.predict(X))
plt.title("PREDICTION PRICE")
plt.xlabel("Predicted Price")
plt.ylabel("Frequency")
plt.show()
```



In [51]:

```
print (np.sum((bos.PRICE - lm.predict(X)) ** 2))
```

```
11078.784577954977
```

In [52]:

```
print("Mean squared error: %.2f"
      % np.mean((lm.predict(X) - bos.PRICE) ** 2))
```

```
Mean squared error: 21.89
```

In [53]:

```
lm = LinearRegression()
lm.fit(X[['PTRATIO']], bos.PRICE)
```

Out[53]:

```
LinearRegression()
```

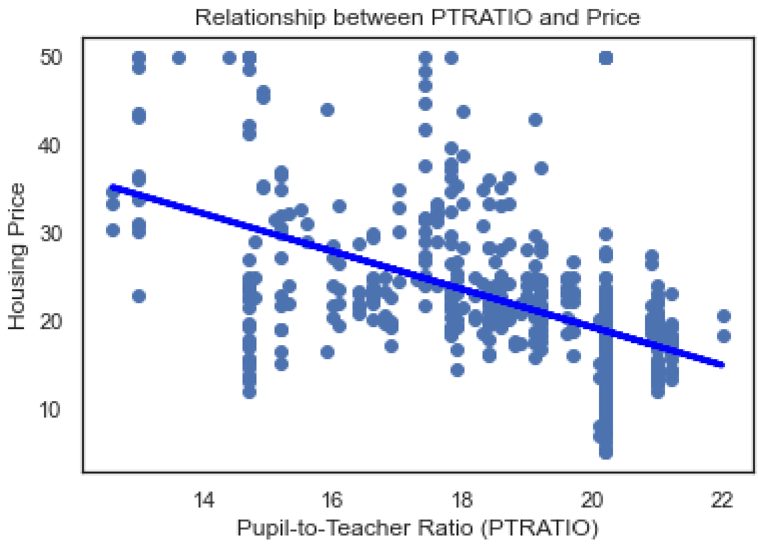
In [54]:

```
msePTRATIO = np.mean((bos.PRICE - lm.predict(X[['PTRATIO']])) ** 2)
print (msePTRATIO)
```

62.65220001376927

In [55]:

```
plt.scatter(bos.PTRATIO, bos.PRICE)
plt.xlabel("Pupil-to-Teacher Ratio (PTRATIO)")
plt.ylabel("Housing Price")
plt.title("Relationship between PTRATIO and Price")

plt.plot(bos.PTRATIO, lm.predict(X[['PTRATIO']]), color='blue', linewidth=3)
plt.show()
```



In [56]:

```
lm.fit(X[['PTRATIO','CRIM','RM']], bos.PRICE)
```

Out[56]:

LinearRegression()

In [57]:

```
msePTRATIO = np.mean((bos.PRICE - lm.predict(X[['PTRATIO','CRIM','RM']])) ** 2)
print (msePTRATIO)
```

34.24552790529693