In [1]:

```
import numpy as np
import pandas as pd
import scipy.stats
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [2]:

```
boston_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDev
boston_df=pd.read_csv(boston_url)
```

In [3]:

```
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   CRIM        506 non-null    float64
 2   ZN          506 non-null    float64
 3   INDUS       506 non-null    float64
 4   CHAS        506 non-null    float64
 5   NOX         506 non-null    float64
 6   RM          506 non-null    float64
 7   AGE         506 non-null    float64
 8   DIS         506 non-null    float64
 9   RAD         506 non-null    float64
 10  TAX         506 non-null    float64
 11  PTRATIO     506 non-null    float64
 12  LSTAT       506 non-null    float64
 13  MEDV        506 non-null    float64
dtypes: float64(13), int64(1)
memory usage: 55.5 KB
```

In [4]:

```
boston_df.describe()
```

Out[4]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	252.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	126.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	252.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	378.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
max	505.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

In [5]:

```
boston_df.columns
```

Out[5]:

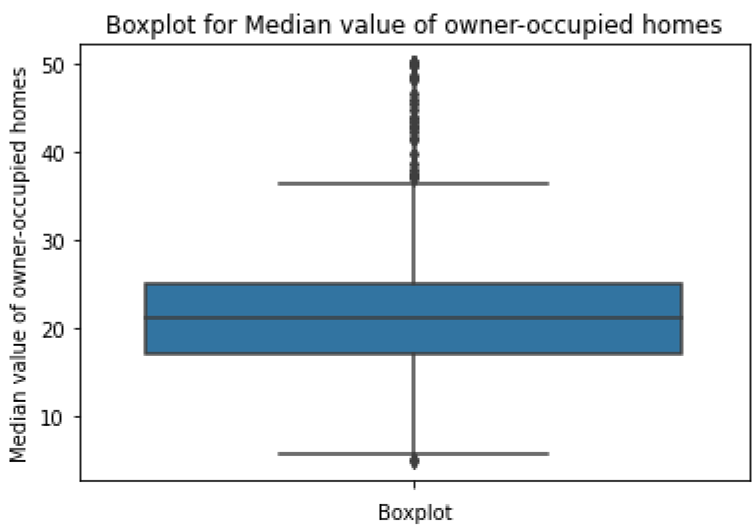
```
Index(['Unnamed: 0', 'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
       'RAD', 'TAX', 'PTRATIO', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [6]:

```
# Display a Boxplot for the "Median value of owner-occupied homes" column.
box = sns.boxplot(y = 'MEDV', data = boston_df)
box.set(ylabel = "Median value of owner-occupied homes"
        , xlabel = "Boxplot"
        , title = "Boxplot for Median value of owner-occupied homes")
```

Out[6]:

```
[Text(0, 0.5, 'Median value of owner-occupied homes'),
 Text(0.5, 0, 'Boxplot'),
 Text(0.5, 1.0, 'Boxplot for Median value of owner-occupied homes')]
```



In [7]:

```
np.unique(boston_df['CHAS'])
```

Out[7]:

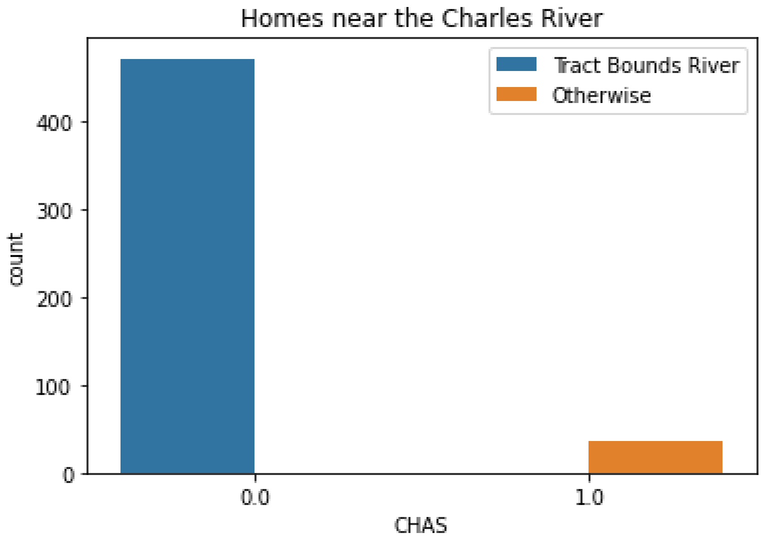
```
array([0., 1.])
```

In [8]:

```
#Provide a  bar plot for the Charles river variable, 1 if tract bounds river; 0 otherwi
barplot = sns.countplot(x = 'CHAS', data = boston_df, hue='CHAS')
barplot.set_title('Homes near the Charles River')
plt.legend(labels=["Tract Bounds River", "Otherwise"])
```

Out[8]:

```
<matplotlib.legend.Legend at 0x25b7eb9fc08>
```

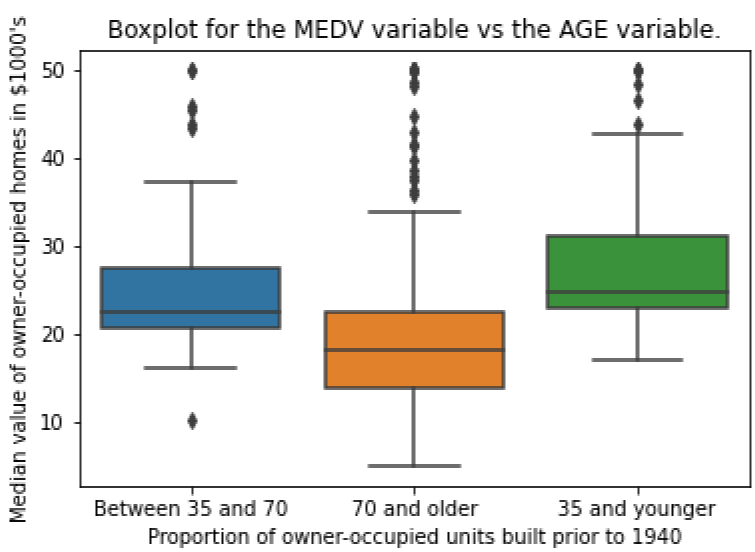


In [9]:
```
#Provide a boxplot for the MEDV variable vs the AGE variable.
#(Discretize the age variable into three groups of 35 years and younger, between 35 and

boston_df.loc[boston_df['AGE'] < 35, 'Age_Group'] = "35 and younger"
boston_df.loc[(boston_df['AGE'] >= 35) & (boston_df['AGE'] < 70), 'Age_Group'] = "Betwe
boston_df.loc[(boston_df['AGE'] >= 70), 'Age_Group'] = "70 and older"
```

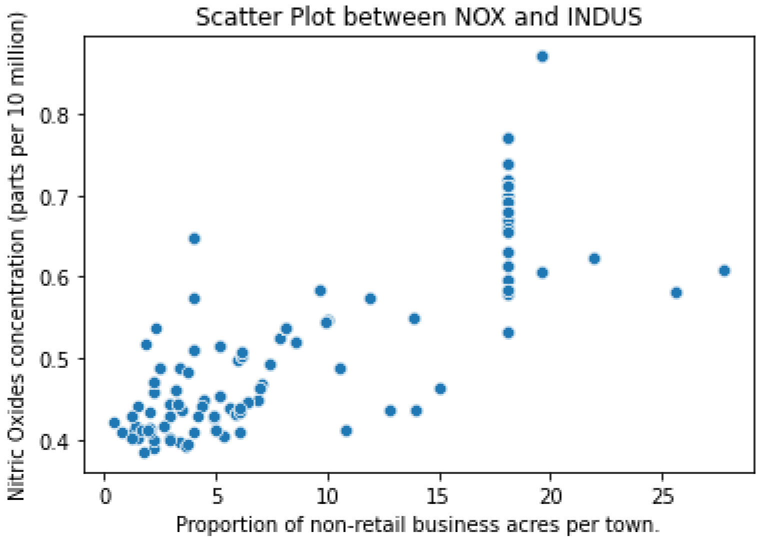
In [23]:
```
boxplot = sns.boxplot(x = 'Age_Group', y = 'MEDV', data = boston_df)
boxplot.set(xlabel = "Proportion of owner-occupied units built prior to 1940"
           , ylabel = "Median value of owner-occupied homes in $1000's"
           , title = "Boxplot for the MEDV variable vs the AGE variable.")
```

Out[23]:
```
[Text(0.5, 0, 'Proportion of owner-occupied units built prior to 1940'),
 Text(0, 0.5, "Median value of owner-occupied homes in $1000's"),
 Text(0.5, 1.0, 'Boxplot for the MEDV variable vs the AGE variable.')]
```



In [11]:
```
#Provide a scatter plot to show the relationship between Nitric oxide concentrations an
#What can you say about the relationship?
scatter = sns.scatterplot(x = 'INDUS', y = 'NOX', data = boston_df)
scatter.set(ylabel = "Nitric Oxides concentration (parts per 10 million)"
            , xlabel = "Proportion of non-retail business acres per town."
           , title = "Scatter Plot between NOX and INDUS")
```

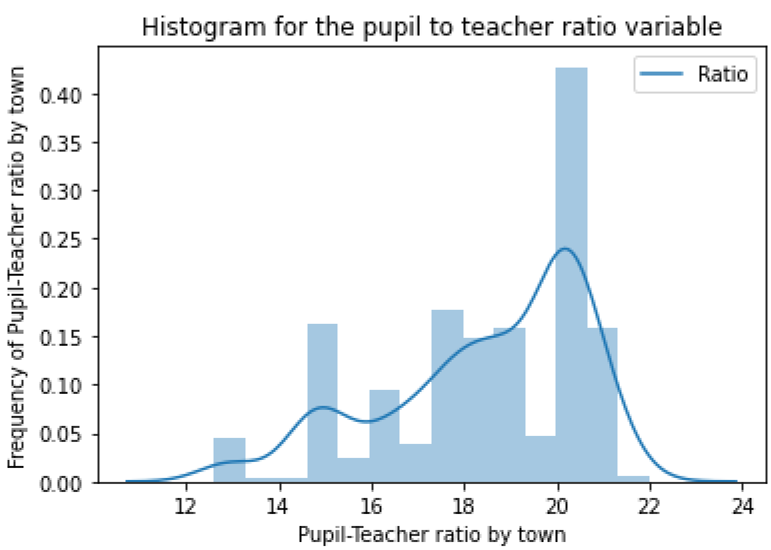
Out[11]:
```
[Text(0, 0.5, 'Nitric Oxides concentration (parts per 10 million)'),
 Text(0.5, 0, 'Proportion of non-retail business acres per town.'),
 Text(0.5, 1.0, 'Scatter Plot between NOX and INDUS')]
```



In [24]:
```
#Create a histogram for the pupil to teacher ratio variable
histplot = sns.distplot(boston_df['PTRATIO'])
histplot.set(xlabel = "Pupil-Teacher ratio by town"
             , ylabel = "Frequency of Pupil-Teacher ratio by town"
            , title = "Histogram for the pupil to teacher ratio variable")
plt.legend(labels=["Ratio"])
```

```
C:\Users\fformat\anaconda4\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[24]:
```
<matplotlib.legend.Legend at 0x25b015e4ac8>
```



The Levene test checks whether several groups have the same variance in the population. Levene's test is therefore used to test the null hypothesis that the samples to be compared come from a population with the same variance.

In [21]:
```
#Question:
#Is there a significant difference in median value of houses bounded by the Charles riv
#(T-test for independent samples)

#Hypothesis:
#Null Hypothesis:μ1=μ2, There is no difference in median value of houses bounded by the
#Alternate Hypotherisis:μ1≠μ2, "There is a difference in median value of houses bounded

scipy.stats.levene(boston_df['MEDV'], boston_df['CHAS'], center = 'mean')
#T test
scipy.stats.ttest_ind(boston_df['MEDV'], boston_df['CHAS'])

#Conclusion:
#As the p value is less than 0.05, reject null hypothesis
```

Out[21]:
```
Ttest_indResult(statistic=54.9210289745203, pvalue=1.4651540072350996e-305)
```

In [15]:
```
#Question:
#Is there a difference in Median values of houses (MEDV) for each proportion of owner o


#Hypothesis:
#Null Hypothesis: H0:μ1=μ2=μ3, the three population means are equal
#Alternate Hypothesis: H1: At least one of the means differ

#ANOVA Test:
a = boston_df[boston_df['Age_Group'] == "35 and younger"]['MEDV']
b = boston_df[boston_df['Age_Group'] == "Between 35 and 70"]['MEDV']
c = boston_df[boston_df['Age_Group'] == "70 and older"]['MEDV']
scipy.stats.f_oneway(a, b, c)

#Conclusion:
#As the p value is less than 0.05, reject the null hypothesis
```

Out[15]:
```
F_onewayResult(statistic=36.40764999196599, pvalue=1.7105011022702984e-15)
```

In [25]:
```
#Questio: can we conclude that there is no relationship between Nitric oxide concentrat

#Hypothesis:
#Null Hypothesis:H0: There is no correlation between Nitric oxide concentrations and pro
#Alternate Hypothesis: H1: There is a relationship between Nitric oxide concentrations a

#Pearson Test:
scipy.stats.pearsonr(boston_df['INDUS'], boston_df['NOX'])

#Conclusion:
#As the p is less than 0.05, reject null hypothesis
```

Out[25]:
```
(0.763651446920915, 7.913361061239593e-98)
```

In [20]:
```
#Question: What is the impact of an additional weighted distance
#to the five Boston employment centres on the median value of owner occupied homes? (Reg

#Hypothesis:
```

```
#Null Hypothesis: H0:β1 = 0 (There is no impact of an additional weighted distance to t
#Alternate Hypothesis: H1:β1 is not equal to 0 (There is an impact of an additional wei

#Regeression Test:
## X is the input variables (or independent variables)
X = boston_df['DIS']
## y is the target/dependent variable
y = boston_df['MEDV']
## add an intercept (beta_0) to our model
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
predictions = model.predict(X)
# Print out the statistics
model.summary()

#Conclusion:
#As the p is less than 0.05, reject null hypothesis
```

```
C:\Users\fformat\anaconda4\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarn
ing: In a future version of pandas all arguments of concat except for the argument 'obj
s' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[20]:

OLS Regression Results

Dep. Variable:	MEDV	**R-squared:**	0.062
Model:	OLS	**Adj. R-squared:**	0.061
Method:	Least Squares	**F-statistic:**	33.58
Date:	Thu, 21 Dec 2023	**Prob (F-statistic):**	1.21e-08
Time:	21:28:55	**Log-Likelihood:**	-1823.9
No. Observations:	506	**AIC:**	3652.
Df Residuals:	504	**BIC:**	3660.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	18.3901	0.817	22.499	0.000	16.784	19.996
DIS	1.0916	0.188	5.795	0.000	0.722	1.462

Omnibus:	139.779	**Durbin-Watson:**	0.570
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	305.104
Skew:	1.466	**Prob(JB):**	5.59e-67
Kurtosis:	5.424	**Cond. No.**	9.32

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.