Import the required libraries we need for the lab.

```
import piplite
await piplite.install(['numpy'],['pandas'])
await piplite.install(['seaborn'])

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as pyplot
import matplotlib.pyplot as plt
import scipy.stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

Read the dataset in the csv file from the URL

```
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ST0151EN-
SkillsNetwork/labs/boston_housing.csv'
resp = await fetch(URL)
boston_url = io.BytesIO((await resp.arrayBuffer()).to_py())

boston_df=pd.read_csv(boston_url)
```

Add your code below following the instructions given in the course to complete the peer graded assignment

```
boston_df
```

```
     Unnamed: 0     CRIM    ZN  INDUS  CHAS    NOX     RM   AGE
DIS  RAD  \
0             0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2
4.0900  1.0
1             1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9
4.9671  2.0
2             2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1
4.9671  2.0
3             3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8
6.0622  3.0
4             4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2
6.0622  3.0
..          ...      ...   ...    ...   ...    ...    ...   ...     ..
.  ...
501         501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1
2.4786  1.0
502         502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7
```

```
2.2875  1.0
503         503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0
2.1675  1.0
504         504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3
2.3889  1.0
505         505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8
2.5050  1.0

       TAX  PTRATIO  LSTAT  MEDV
0    296.0     15.3   4.98  24.0
1    242.0     17.8   9.14  21.6
2    242.0     17.8   4.03  34.7
3    222.0     18.7   2.94  33.4
4    222.0     18.7   5.33  36.2
..     ...      ...    ...   ...
501  273.0     21.0   9.67  22.4
502  273.0     21.0   9.08  20.6
503  273.0     21.0   5.64  23.9
504  273.0     21.0   6.48  22.0
505  273.0     21.0   7.88  11.9

[506 rows x 14 columns]
```

```
boston_df.info()
```

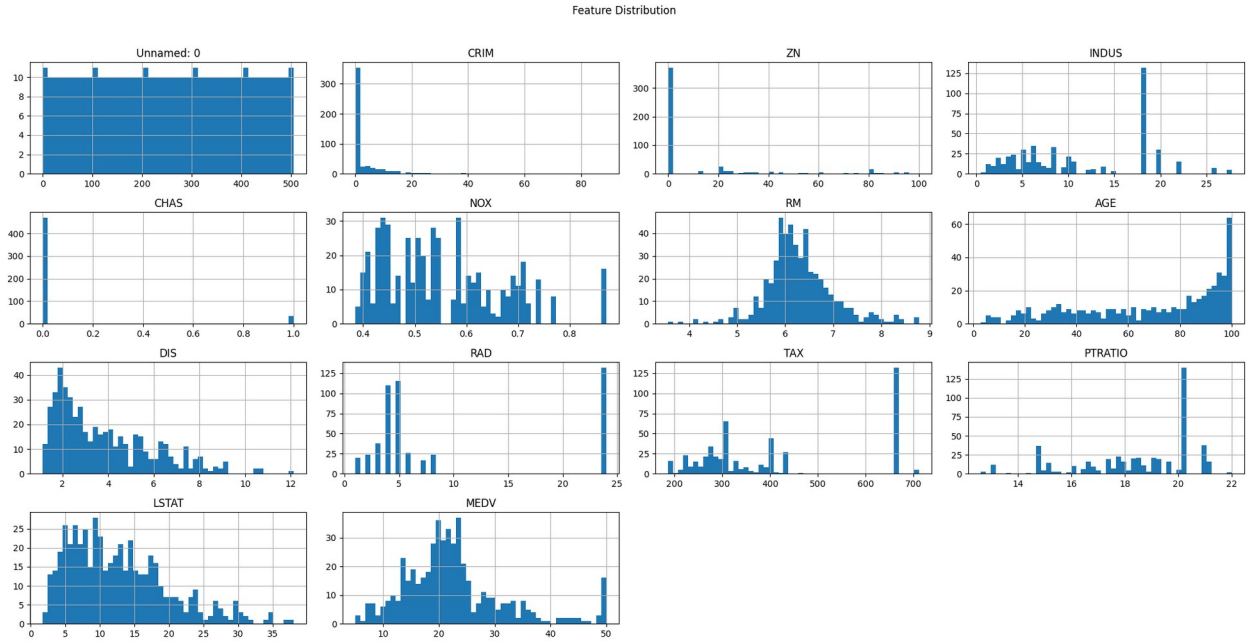
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   CRIM        506 non-null    float64
 2   ZN          506 non-null    float64
 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: 55.4 KB
```

```
boston_df.describe()
```

```
       Unnamed: 0        CRIM          ZN       INDUS        CHAS
NOX  \
count  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
std    146.213884    8.601545   23.322453    6.860353    0.253994
0.115878
min      0.000000    0.006320    0.000000    0.460000    0.000000
0.385000
25%    126.250000    0.082045    0.000000    5.190000    0.000000
0.449000
50%    252.500000    0.256510    0.000000    9.690000    0.000000
0.538000
75%    378.750000    3.677083   12.500000   18.100000    0.000000
0.624000
max    505.000000   88.976200  100.000000   27.740000    1.000000
0.871000

                RM          AGE          DIS          RAD          TAX
PTRATIO  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean     6.284634   68.574901    3.795043    9.549407  408.237154
18.455534
std      0.702617   28.148861    2.105710    8.707259  168.537116
2.164946
min      3.561000    2.900000    1.129600    1.000000  187.000000
12.600000
25%      5.885500   45.025000    2.100175    4.000000  279.000000
17.400000
50%      6.208500   77.500000    3.207450    5.000000  330.000000
19.050000
75%      6.623500   94.075000    5.188425   24.000000  666.000000
20.200000
max      8.780000  100.000000   12.126500   24.000000  711.000000
22.000000

             LSTAT        MEDV
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

HISTOGRAM

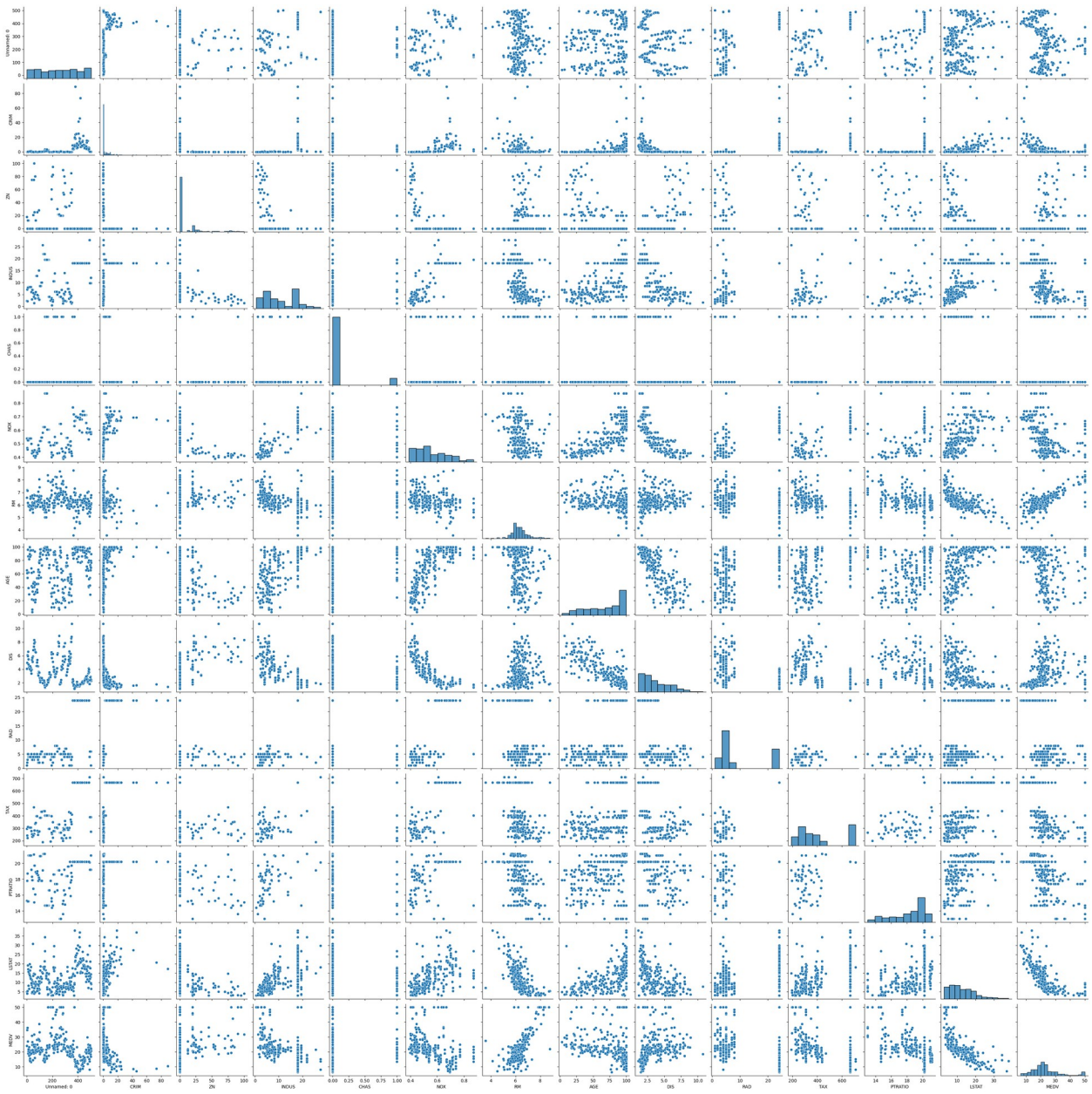
```
boston_df.hist(bins=50, figsize=(20,10))
plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center',
fontsize='large')
plt.tight_layout()
plt.show()
```



PairPlot Features

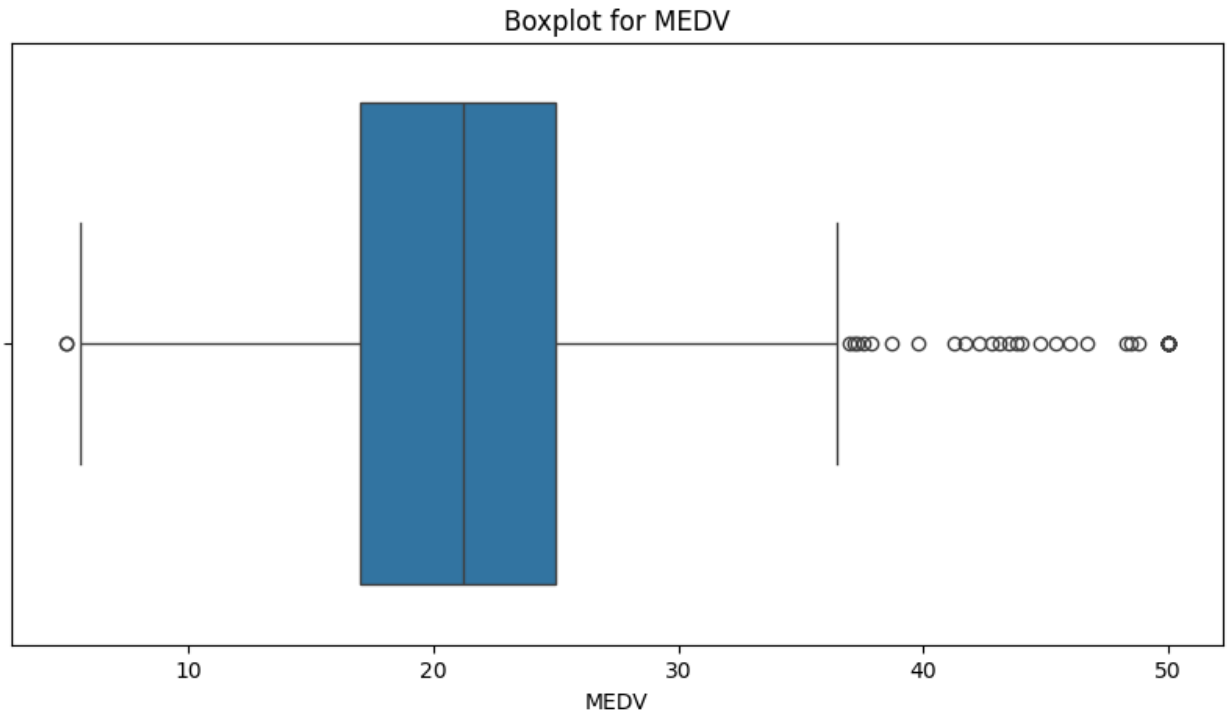
```
plt.figure(figsize=(20,20))
plt.suptitle('Pairplots of features', x=0.5, y=1.02, ha='center',
fontsize='large')
sns.pairplot(boston_df.sample(250))
plt.show()
```

```
<Figure size 2000x2000 with 0 Axes>
```



Median value of owner-occupied homes : Box Plot

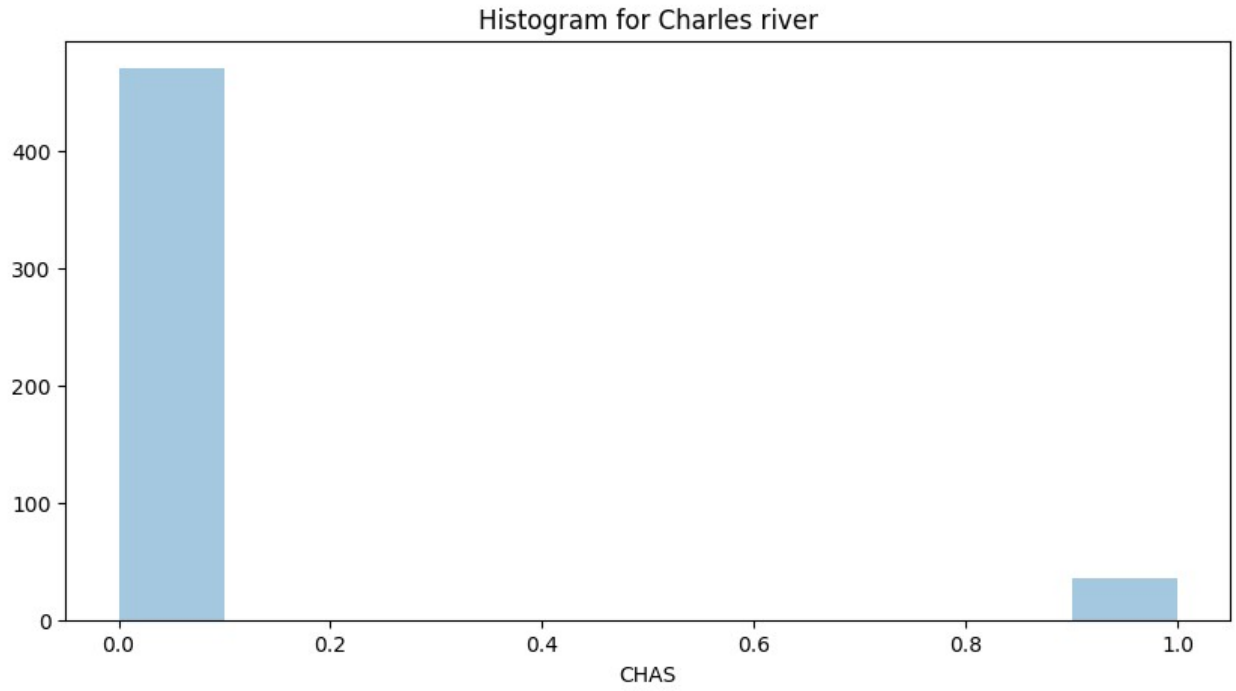
```
plt.figure(figsize=(10,5))
sns.boxplot(x=boston_df.MEDV)
plt.title("Boxplot for MEDV")
plt.show()
```



Charles river variable : Histogram

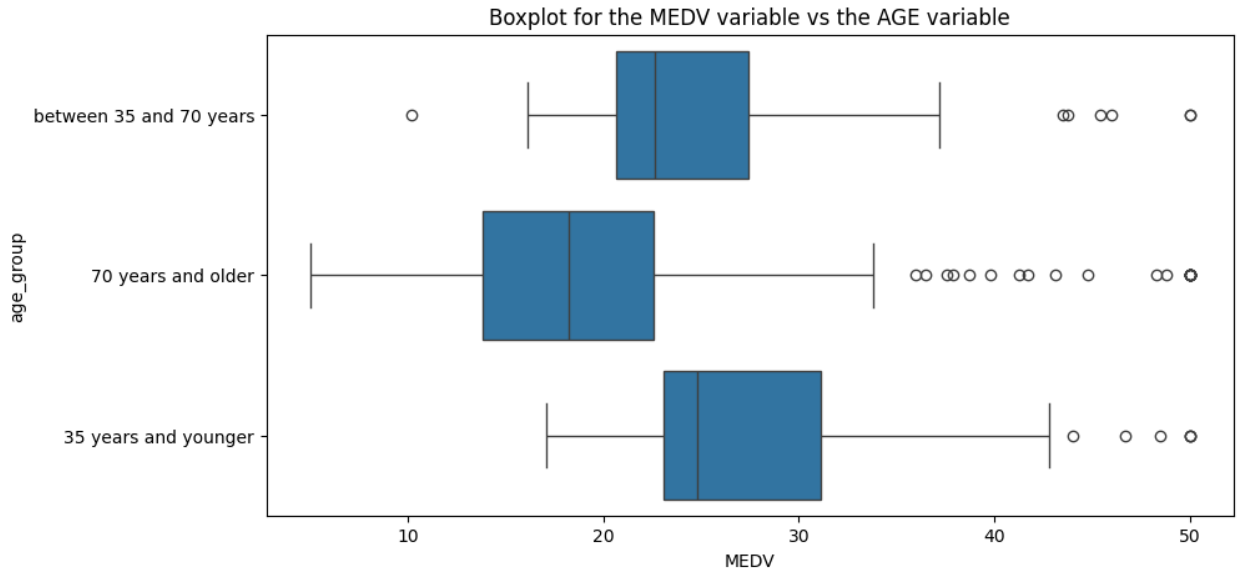
```
plt.figure(figsize=(10,5))
sns.distplot(a=boston_df.CHAS,bins=10, kde=False)
plt.title("Histogram for Charles river")
plt.show()
```

```
<ipython-input-23-8525a3986b5f>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(a=boston_df.CHAS,bins=10, kde=False)
```



MEDV variable vs the AGE variable.

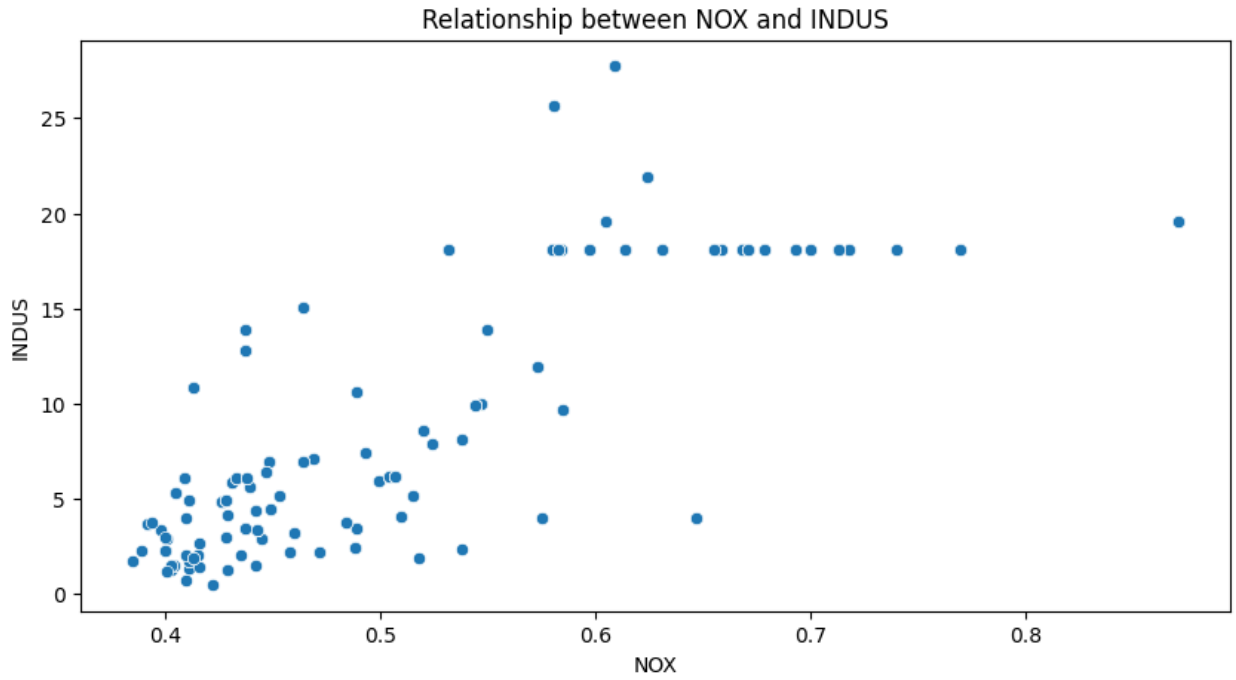
```
boston_df.loc[(boston_df["AGE"] <= 35),'age_group'] = '35 years and
younger'
boston_df.loc[(boston_df["AGE"] > 35) &
(boston_df["AGE"]<70),'age_group'] = 'between 35 and 70 years'
boston_df.loc[(boston_df["AGE"] >= 70),'age_group'] = '70 years and
older'

plt.figure(figsize=(10,5))
sns.boxplot(x=boston_df.MEDV, y=boston_df.age_group, data=boston_df)
plt.title("Boxplot for the MEDV variable vs the AGE variable")
plt.show()
```



Note: 35 years or younger group pays the highest median house price while above 70s are shifting to cheaper houses

Relationship between Nitric oxide concentrations and the proportion of non-retail business acres per town

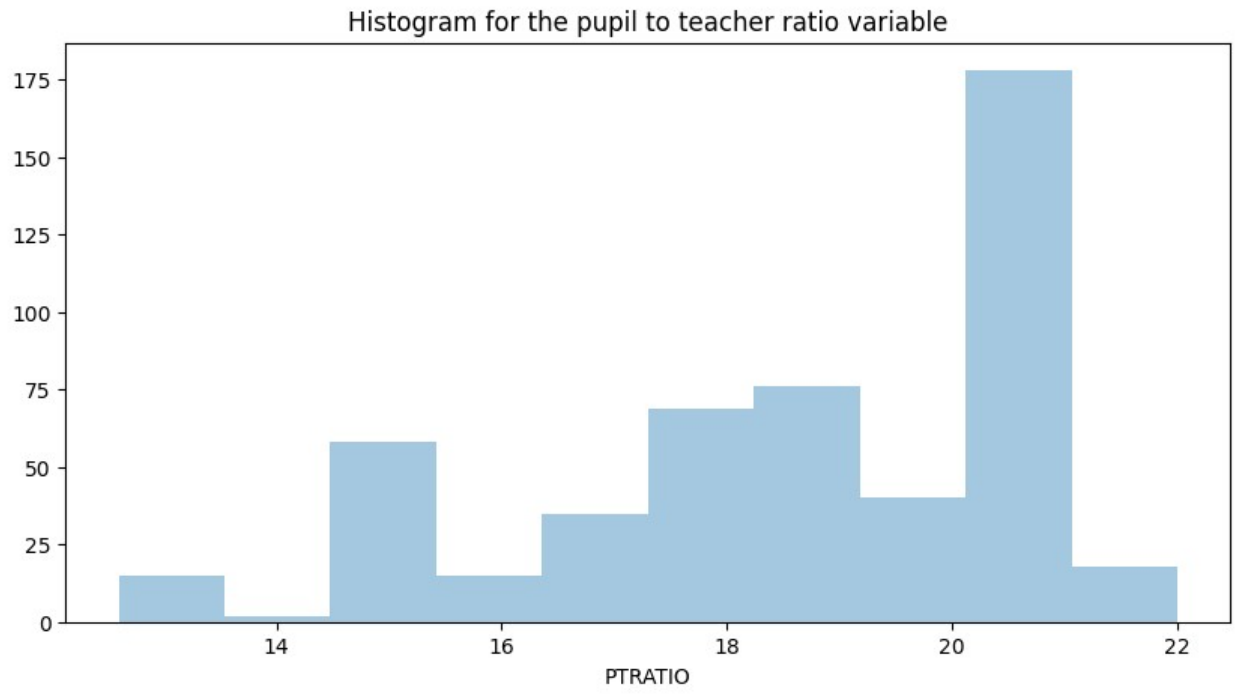
```
plt.figure(figsize=(10,5))
sns.scatterplot(x=boston_df.NOX, y=boston_df.INDUS, data=boston_df)
plt.title("Relationship between NOX and INDUS")
plt.show()
```



Note: There seems to be a linear relationship till NOX=0.6

```
plt.figure(figsize=(10,5))
sns.distplot(a=boston_df.PTRATIO,bins=10, kde=False)
plt.title("Histogram for the pupil to teacher ratio variable")
plt.show()
```

```
<ipython-input-29-a8db33a5b6f2>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(a=boston_df.PTRATIO,bins=10, kde=False)
```



Note: Pupil to teacher ratio is highest at 20-21 range.

Is there a significant difference in median value of houses bounded by the Charles river or not? (T-test for independent samples)

Null Hypothesis(H0): Both average MEDV are the same

Alternative Hypothesis(HA): Both average MEDV are NOT the same

```
boston_df["CHAS"].value_counts()
a = boston_df[boston_df["CHAS"] == 0]["MEDV"]
a
```

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 471, dtype: float64
```

```
b = boston_df[boston_df["CHAS"] == 1]["MEDV"]
b
```

```
142    13.4
152    15.3
154    17.0
155    15.6
160    27.0
162    50.0
163    50.0
208    24.4
209    20.0
210    21.7
211    19.3
212    22.4
216    23.3
218    21.5
219    23.0
220    26.7
221    21.7
222    27.5
234    29.0
236    25.1
269    20.7
```

```
273    35.2
274    32.4
276    33.2
277    33.1
282    46.0
283    50.0
356    17.8
357    21.7
358    22.7
363    16.8
364    21.9
369    50.0
370    50.0
372    50.0
Name: MEDV, dtype: float64
```

```
scipy.stats.ttest_ind(a,b,axis=0,equal_var=True)
```

```
TtestResult(statistic=-3.996437466090509, pvalue=7.390623170519905e-
05, df=504.0)
```

Since p-value more than alpha value of 0.05, we failed to reject null hypothesis since there is NO statistical significance.

Is there a difference in Median values of houses (MEDV) for each proportion of owner occupied units built prior to 1940 (AGE)?

```
low = boston_df[boston_df["age_group"] == '35 years and younger']
["MEDV"]
mid = boston_df[boston_df["age_group"] == 'between 35 and 70 years']
["MEDV"]
high = boston_df[boston_df["age_group"] == '70 years and older']
["MEDV"]

f_stats, p_value = scipy.stats.f_oneway(low,mid,high,axis=0)
print("F-Statistic={0}, P-value={1}".format(f_stats,p_value))
```

```
F-Statistic=36.40764999196599, P-value=1.7105011022702984e-15
```

Since p-value more than alpha value of 0.05, we failed to reject null hypothesis since there is NO statistical significance.

Can we conclude that there is no relationship between Nitric oxide concentrations and proportion of non-retail business acres per town?

```
pearson,p_value =
scipy.stats.pearsonr(boston_df["NOX"],boston_df["INDUS"])
print("Pearson Coefficient value={0}, P-
value={1}".format(pearson,p_value))
```

```
Pearson Coefficient value=0.7636514469209192, P-
value=7.913361061210442e-98
```

Since the p-value (Sig. (2-tailed) < 0.05, we reject the Null hypothesis and conclude that there exists a relationship between Nitric Oxide and non-retail business acres per town.

What is the impact of an additional weighted distance to the five Boston employment centres on the median value of owner occupied homes?

Null Hypothesis: weighted distances to five Boston employment centres are not related to median value

Alternative Hypothesis: weighted distances to five Boston employment centres are related to median value

```
y = boston_df['MEDV']
x = boston_df['DIS']
x = sm.add_constant(x)
results = sm.OLS(y,x).fit()
results.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            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:                Mon, 07 Oct 2024   Prob (F-statistic):
1.21e-08
Time:                        11:44:49   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.
"""

np.sqrt(0.062)

0.24899799195977465
```

The square root of R-squared is 0.25, which implies weak correlation between both features

Correlation

```
plt.figure(figsize=(16,9))
sns.heatmap(boston_df.corr(),cmap="coolwarm",annot=True,fmt='.2f',line
widths=2, cbar=False)
plt.show()

<Figure size 1600x900 with 0 Axes>
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[45], line 2
      1 plt.figure(figsize=(16,9))
----> 2 sns.heatmap(boston_df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2, cbar=False)
      3 plt.show()

File /lib/python3.12/site-packages/pandas/core/frame.py:11022, in DataFrame.corr(self, method, min_periods, numeric_only)
  11020 cols = data.columns
  11021 idx = cols.copy()
> 11022 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11024 if method == "pearson":
  11025     correl = libalgos.nancorr(mat, minp=min_periods)

File /lib/python3.12/site-packages/pandas/core/frame.py:1981, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1979 if dtype is not None:
   1980     dtype = np.dtype(dtype)
-> 1981 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1982 if result.dtype is not dtype:
   1983     result = np.array(result, dtype=dtype, copy=False)

File /lib/python3.12/site-packages/pandas/core/internals/managers.py:1693, in BlockManager.as_array(self, dtype, copy, na_value)
   1691         arr.flags.writeable = False
   1692 else:
-> 1693     arr = self._interleave(dtype=dtype, na_value=na_value)
   1694     # The underlying data was copied within _interleave, so no need
   1695     # to further copy if copy=True or setting na_value
   1697 if na_value is lib.no_default:

File /lib/python3.12/site-packages/pandas/core/internals/managers.py:1752, in BlockManager._interleave(self, dtype, na_value)
   1750     else:
   1751         arr = blk.get_values(dtype)
-> 1752     result[rl.indexer] = arr
   1753     itemmask[rl.indexer] = 1
   1755 if not itemmask.all():

ValueError: could not convert string to float: 'between 35 and 70 years'
```