Import required libraries

In [1]:
```
# list of all the libararies required in the code for easy reference
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Load data

In [2]:
```
# load the required data
housing_df=pd.read_csv("BostonHousing.csv")
```

In [3]:
```
# view the data
print(housing_df)
housing_df.shape
# the dataset has 506 cases (rows) and 14 attributes (columns)
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

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

[506 rows x 14 columns]
```

Out[3]:
```
(506, 14)
```

Rename Column CAT.MEDV

In [4]:
```
# rename column CAT.MEDV to CAT_MEDV for better understanding
housing_df=housing_df.rename(columns={'CAT. MEDV':'CAT_MEDV'})
```

In [5]:
```
# view the updated dataset
housing_df.head()
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	M
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	

Overall statitics for each column

In [6]:
```
# Describe the overall dataset
housing_df.describe()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
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

Count of missing values

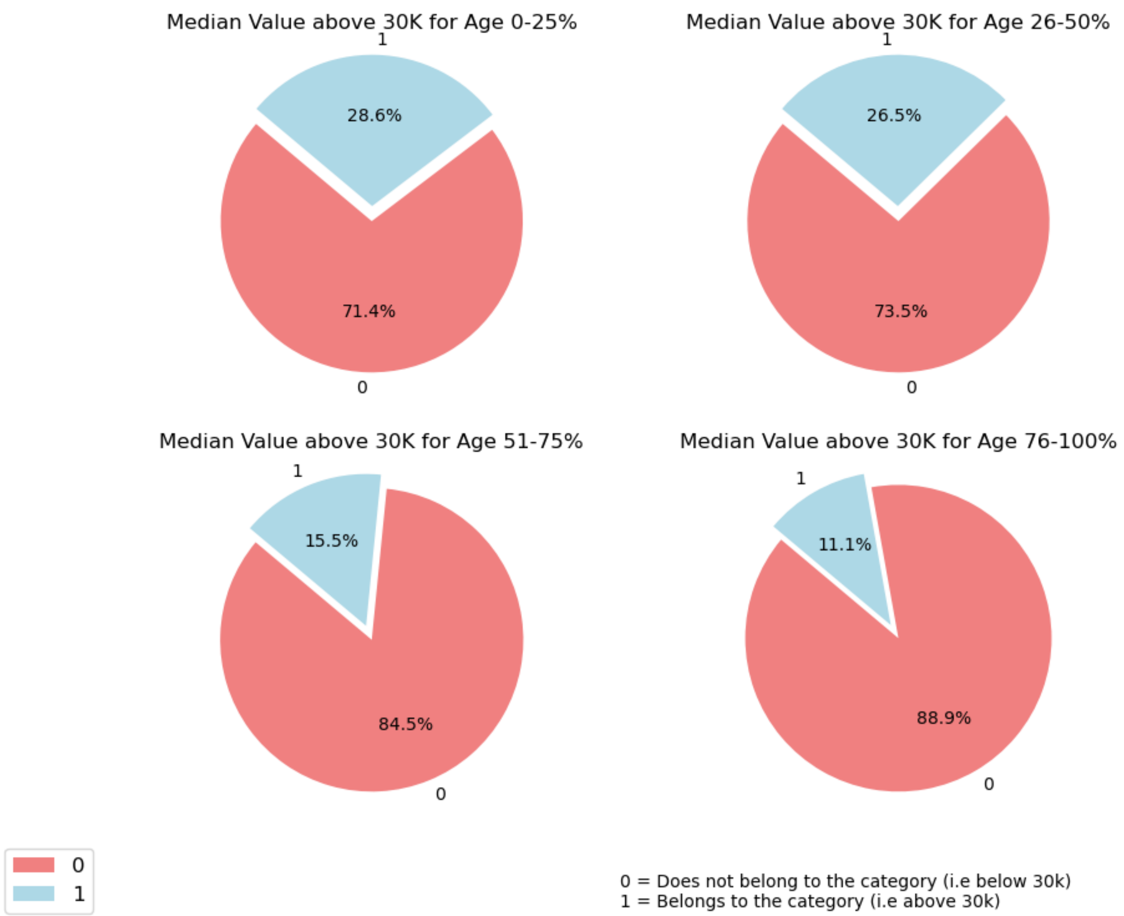
In [7]:
```
# count of missing values in each column
missingvalues=housing_df.isnull().sum()
print(missingvalues)

# the sum of the missing values are zero indicating that the data has no mi
```

```
CRIM        0
ZN          0
INDUS       0
CHAS        0
NOX         0
RM          0
AGE         0
DIS         0
RAD         0
TAX         0
PTRATIO     0
LSTAT       0
MEDV        0
CAT_MEDV    0
dtype: int64
```

Pie chart for AGE at different levels

In [8]:
```
# Categorize 'AGE' into different levels
bins = [0, 25, 50, 75, 100]
labels = ['0-25%', '26-50%', '51-75%', '76-100%']

#copy the datframe
housing_df_pie =housing_df.copy()

# create bins for the dataframe and create comparison dataframe
housing_df_pie['AGE_Category'] = pd.cut(housing_df_pie['AGE'], bins=bins, l
comparison_df = housing_df_pie.groupby(['AGE_Category', 'CAT_MEDV']).size()

# plott the pie chart
colors = ['lightcoral', 'lightblue']
explode = (0.1, 0)  # explode the 1st slice for emphasis
fig, axes = plt.subplots(nrows=2,ncols=2, figsize=(10,8))

for (row_index, row), ax in zip(comparison_df.iterrows(), axes.flatten()):
    ax.pie(row, labels=row.index, autopct='%1.1f%%', colors=colors, startar
    ax.set_title(f"Median Value above 30K for Age {row_index}")
    ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a c

#plt.tight_layout()
fig.legend(row.index,loc="lower left", fontsize = 'large')
fig.text(0.5,0.01, "0 = Does not belong to the category (i.e below 30k)\n1
```

Out[8]: Text(0.5, 0.01, '0 = Does not belong to the category (i.e below 30k)\n1 = Belongs to the category (i.e above 30k)')



Rows of outliers based on PTRATIO

In [9]:

```
# find the outliers for PTRATIO column irrescpetive of the values in CAT_ME

# get the interquartile range
Q1 = housing_df['PTRATIO'].quantile(0.25)
Q3 = housing_df['PTRATIO'].quantile(0.75)
IQR = Q3 - Q1

# define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
print("The lower bound is ", round(lower_bound,4), "\nThe upper bound is ",

# identifying the outliers
outliers = housing_df['PTRATIO'][(housing_df['PTRATIO'] < lower_bound) | (h
if not outliers.empty:
        # all the values for the outlier value
        outlier_value = housing_df.loc[outliers.index]
        print("\nAll Outliers:")
        print(outlier_value)
        #print(outlier_value.count())

# the data records seem to be genuine outliers.
```

```
The lower bound is  13.2
The upper bound is  24.4

All Outliers:
        CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS  RAD  TAX  \
196  0.04011  80.0   1.52     0  0.404  7.287   34.1  7.3090    2  329
197  0.04666  80.0   1.52     0  0.404  7.107   36.6  7.3090    2  329
198  0.03768  80.0   1.52     0  0.404  7.274   38.3  7.3090    2  329
257  0.61154  20.0   3.97     0  0.647  8.704   86.9  1.8010    5  264
258  0.66351  20.0   3.97     0  0.647  7.333  100.0  1.8946    5  264
259  0.65665  20.0   3.97     0  0.647  6.842  100.0  2.0107    5  264
260  0.54011  20.0   3.97     0  0.647  7.203   81.8  2.1121    5  264
261  0.53412  20.0   3.97     0  0.647  7.520   89.4  2.1398    5  264
262  0.52014  20.0   3.97     0  0.647  8.398   91.5  2.2885    5  264
263  0.82526  20.0   3.97     0  0.647  7.327   94.5  2.0788    5  264
264  0.55007  20.0   3.97     0  0.647  7.206   91.6  1.9301    5  264
265  0.76162  20.0   3.97     0  0.647  5.560   62.8  1.9865    5  264
266  0.78570  20.0   3.97     0  0.647  7.014   84.6  2.1329    5  264
267  0.57834  20.0   3.97     0  0.575  8.297   67.0  2.4216    5  264
268  0.54050  20.0   3.97     0  0.575  7.470   52.6  2.8720    5  264

     PTRATIO  LSTAT  MEDV  CAT_MEDV
196     12.6   4.08  33.3         1
197     12.6   8.61  30.3         1
198     12.6   6.62  34.6         1
257     13.0   5.12  50.0         1
258     13.0   7.79  36.0         1
259     13.0   6.90  30.1         1
260     13.0   9.59  33.8         1
261     13.0   7.26  43.1         1
262     13.0   5.91  48.8         1
263     13.0  11.25  31.0         1
264     13.0   8.10  36.5         1
265     13.0  10.45  22.8         0
266     13.0  14.79  30.7         1
267     13.0   7.44  50.0         1
268     13.0   3.16  43.5         1
```

Fix outliers

In [10]:
```
# outliers account for 2.9% of the entire dataset
# we chose to omit these outliers.


housing_df_new = housing_df[~((housing_df['PTRATIO'] < lower_bound) | (hous
housing_df_new.shape
#housing_df_new.isna().sum()

# the new dataset has 491 cases (rows) and 14 attributes (columns)
```

Out[10]:
```
(491, 14)
```

Statistics for each column

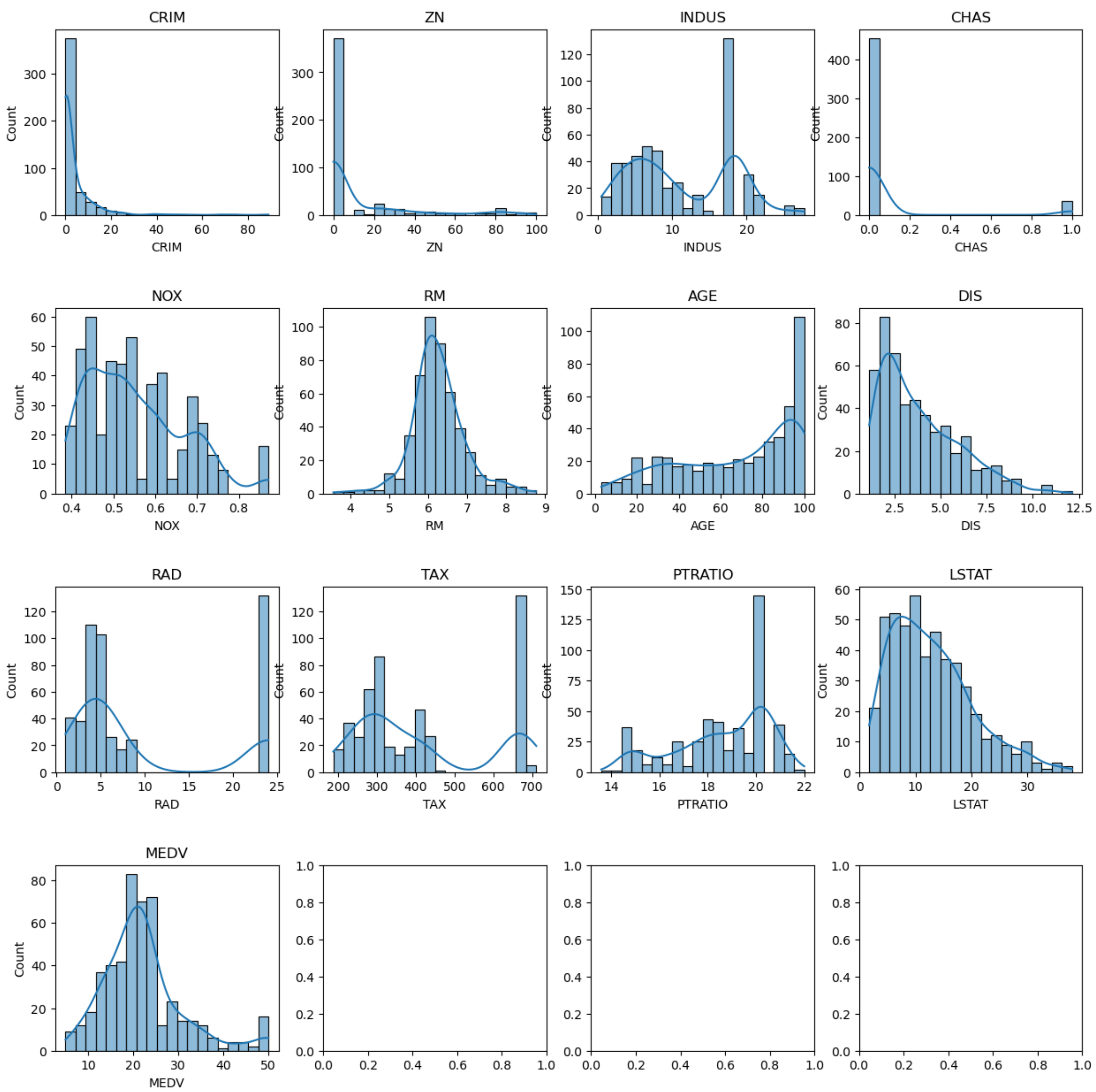
In [11]:
```
# the mean, standard deviation, min, max, median, length, and missing value

pd.DataFrame({'mean': housing_df_new.mean(),
              'median': housing_df_new.median(),
              'min': housing_df_new.min(),
              'max': housing_df_new.max(),
              'range': housing_df_new.max() - housing_df_new.min(),
              'Std. dev': housing_df_new.std(),
              'length': len(housing_df_new),
              'miss_val': housing_df_new.isnull().sum(),
             })
```

Out[11]:

	mean	median	min	max	range	Std. dev	length	miss
CRIM	3.708250	0.24522	0.00632	88.9762	88.96988	8.714712	491	
ZN	10.733198	0.00000	0.00000	100.0000	100.00000	23.011313	491	
INDUS	11.370692	9.90000	0.46000	27.7400	27.28000	6.828344	491	
CHAS	0.071283	0.00000	0.00000	1.0000	1.00000	0.257560	491	
NOX	0.553653	0.53200	0.38500	0.8710	0.48600	0.116288	491	
RM	6.251493	6.18500	3.56100	8.7800	5.21900	0.675457	491	
AGE	68.405703	77.00000	2.90000	100.0000	97.10000	28.277211	491	
DIS	3.814045	3.27210	1.12960	12.1265	10.99690	2.103519	491	
RAD	9.706721	5.00000	1.00000	24.0000	23.00000	8.789577	491	
TAX	412.246436	337.00000	187.00000	711.0000	524.00000	169.440997	491	
PTRATIO	18.624644	19.10000	13.60000	22.0000	8.40000	1.965453	491	
LSTAT	12.801181	11.65000	1.73000	37.9700	36.24000	7.181111	491	
MEDV	22.091853	20.90000	5.00000	50.0000	45.00000	8.868464	491	
CAT_MEDV	0.142566	0.00000	0.00000	1.0000	1.00000	0.349986	491	

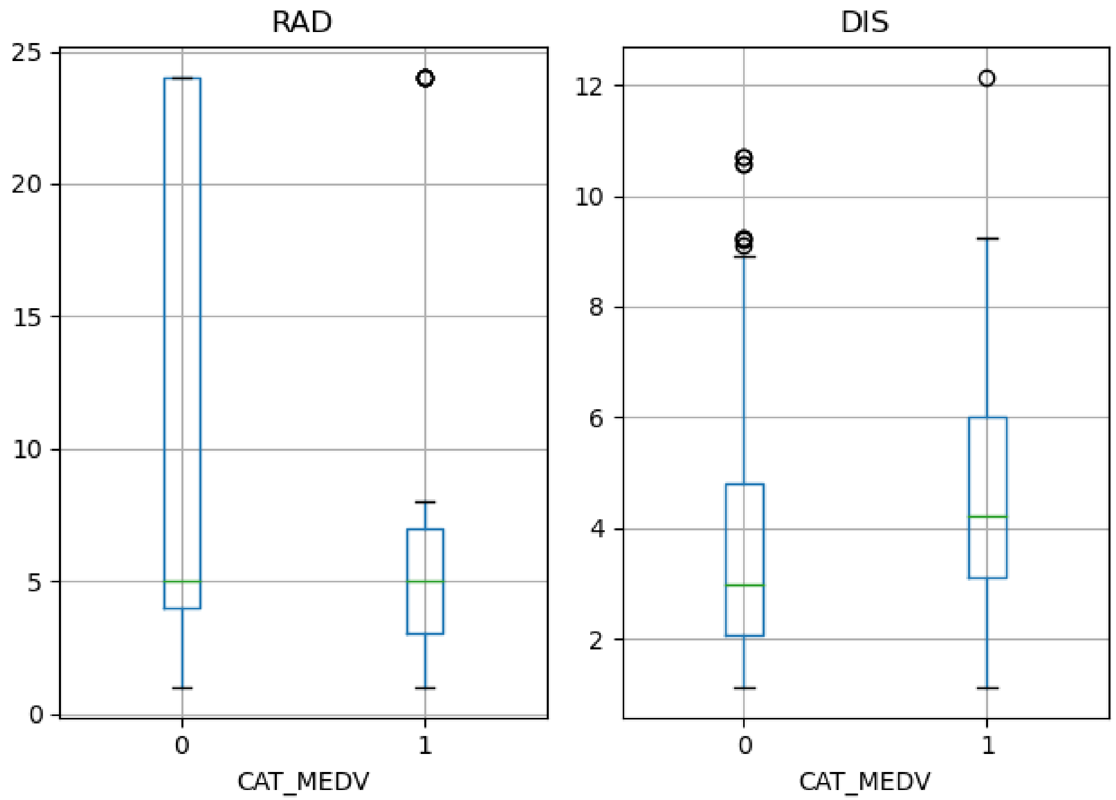
Histogram for each quantitative variables

In [12]:
```
# plot histogram for each variable in the new dataframe

# list of quantitative variable names
quantitative_vars = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DI

# set up subplots
fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(15, 15))
fig.subplots_adjust(hspace=0.5)

# flatten the axes for easier iteration
axes = axes.flatten()

# plot histograms for each variable
for i, var in enumerate(quantitative_vars):
    sns.histplot(housing_df_new[var], ax=axes[i], bins=20, kde=True)
    axes[i].set_title(var)

plt.show()


# TAX varibale has the highest level of variability.

# CRIM, ZN, CHAS, AGE, DIS, LSTAT are the variables which show skewness.

# the variables CRIM, AGE, DIS and LSTAT seem to have extreme values.
```



Boxplot comparing two variables

In [13]:
```
# plot the boxplot for RAD & DIS
fig, axes = plt.subplots(nrows = 1, ncols = 2)
housing_df_new.boxplot(column='RAD', by='CAT_MEDV', ax=axes[0])
housing_df_new.boxplot(column='DIS', by='CAT_MEDV', ax=axes[1])
for ax in axes:
    ax.set_xlabel('CAT_MEDV')
plt.suptitle('') # Suppress the overall title
plt.tight_layout()

# the below comparative boxplot below indicates the following points:
# 1. the median value for the index of accessiblity(RAD) and the weighted d
# 2. there is a huge gap between q3 value for RAD and DIS for both the cate
# 3. there are more outliers in DIS for category 0 of CAT_MEDV.
```



Correlation table

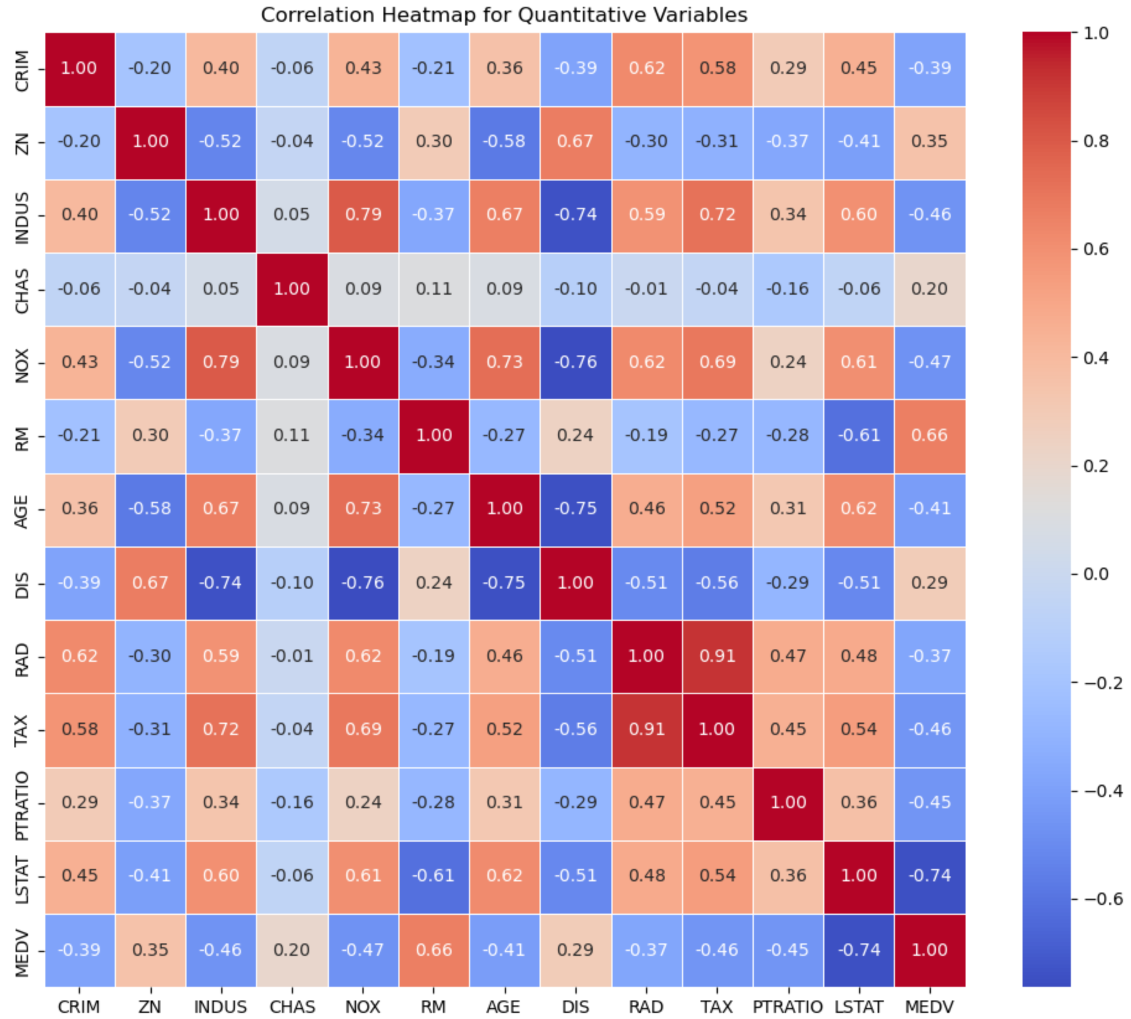
In [14]:

```
quantitative_vars = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DI

# Calculate the correlation matrix
correlation_matrix = housing_df_new[quantitative_vars].corr()
correlation_matrix_df = pd.DataFrame(housing_df_new[quantitative_vars].corr

# Display the correlation table
print("Correlation Table:")
print(correlation_matrix_df)
```

```
Correlation Table:
             CRIM        ZN     INDUS      CHAS       NOX        RM  
AGE  \
CRIM     1.000000 -0.195751  0.402822 -0.059082  0.429326 -0.213881  0.35
8810
ZN      -0.195751  1.000000 -0.523917 -0.036383 -0.520639  0.297024 -0.57
6421
INDUS    0.402822 -0.523917  1.000000  0.054766  0.794809 -0.365529  0.66
8595
CHAS    -0.059082 -0.036383  0.054766  1.000000  0.094851  0.110079  0.08
9192
NOX      0.429326 -0.520639  0.794809  0.094851  1.000000 -0.338535  0.72
7605
RM      -0.213881  0.297024 -0.365529  0.110079 -0.338535  1.000000 -0.27
0840
AGE      0.358810 -0.576421  0.668595  0.089192  0.727605 -0.270840  1.00
0000
DIS     -0.389106  0.671706 -0.740027 -0.103407 -0.763542  0.237893 -0.74
5027
RAD      0.623634 -0.301859  0.589127 -0.012382  0.624151 -0.193630  0.46
4296
TAX      0.581076 -0.309810  0.715600 -0.042537  0.693280 -0.271667  0.52
4742
PTRATIO  0.293197 -0.368373  0.337121 -0.159898  0.237800 -0.277370  0.31
1267
LSTAT    0.453143 -0.406844  0.597411 -0.060225  0.606360 -0.614479  0.61
5702
MEDV    -0.391368  0.352029 -0.462905  0.198515 -0.473486  0.664089 -0.41
4797

              DIS       RAD       TAX   PTRATIO     LSTAT      MEDV  
CRIM    -0.389106  0.623634  0.581076  0.293197  0.453143 -0.391368  
ZN       0.671706 -0.301859 -0.309810 -0.368373 -0.406844  0.352029  
INDUS   -0.740027  0.589127  0.715600  0.337121  0.597411 -0.462905  
CHAS    -0.103407 -0.012382 -0.042537 -0.159898 -0.060225  0.198515  
NOX     -0.763542  0.624151  0.693280  0.237800  0.606360 -0.473486  
RM       0.237893 -0.193630 -0.271667 -0.277370 -0.614479  0.664089  
AGE     -0.745027  0.464296  0.524742  0.311267  0.615702 -0.414797  
DIS      1.000000 -0.506837 -0.560267 -0.288003 -0.512839  0.288708  
RAD     -0.506837  1.000000  0.910676  0.470248  0.483260 -0.374980  
TAX     -0.560267  0.910676  1.000000  0.451943  0.538496 -0.457212  
PTRATIO -0.288003  0.470248  0.451943  1.000000  0.362028 -0.453797  
LSTAT   -0.512839  0.483260  0.538496  0.362028  1.000000 -0.743389  
MEDV     0.288708 -0.374980 -0.457212 -0.453797 -0.743389  1.000000  
```

Heatmap

In [15]:
```
# Generate a heatmap (matrix plot) for the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", lir
plt.title('Correlation Heatmap for Quantitative Variables')
plt.show()

# the variables TAX and RAD are strongly positively correlated: 0.910676.

# we can reduce the number of variables by removing the variables which are
```



Normalize new data

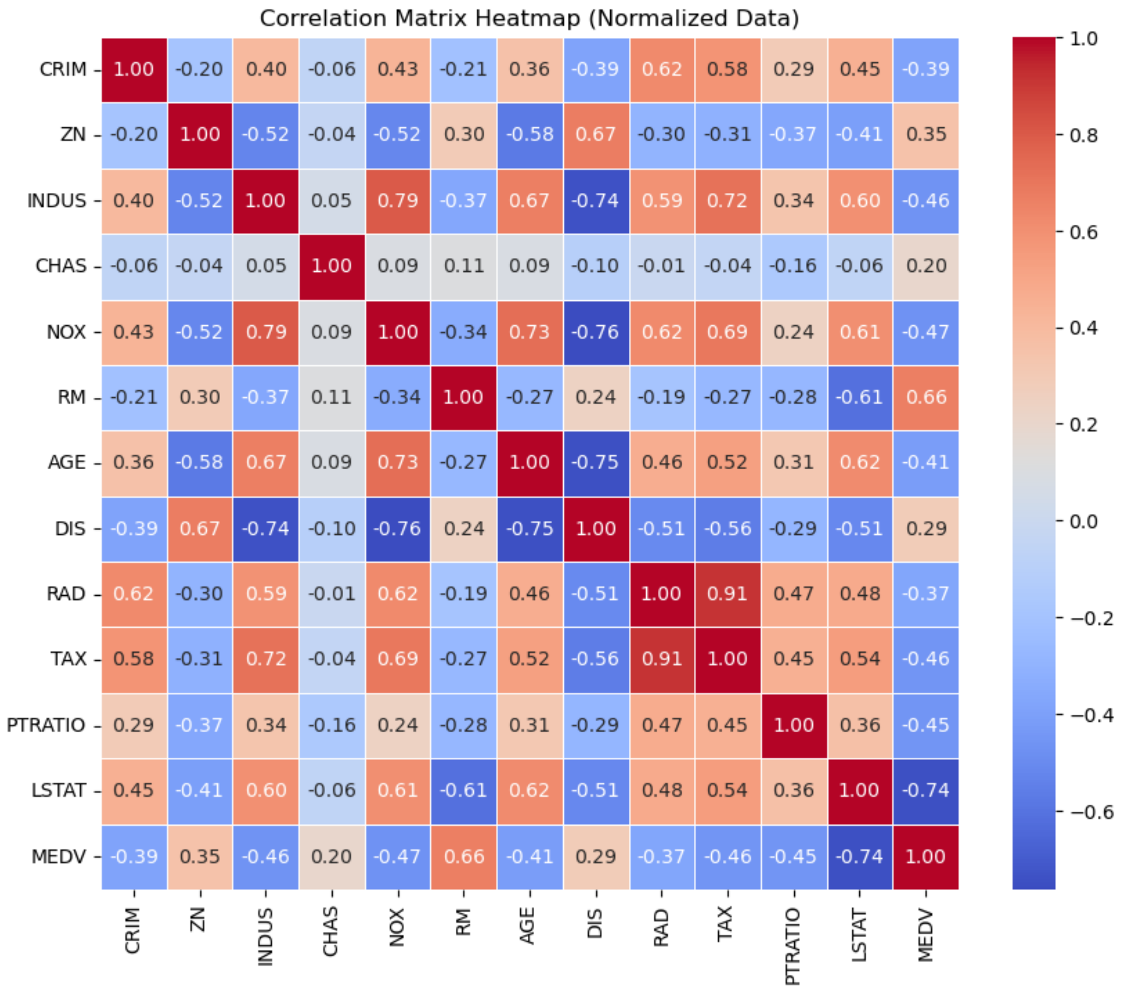
In [16]:
```
# Normalize the data using Z-score normalization
normalized_data = StandardScaler().fit_transform(housing_df_new) #scales th

# Create a DataFrame with normalized data
normalized_df = pd.DataFrame(normalized_data, columns=housing_df_new.column

# Compute the correlation matrix for normalized data
normalized_correlation_matrix = normalized_df[quantitative_vars].corr()
normalized_correlation_matrix
```

Out[16]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	D
CRIM	1.000000	-0.195751	0.402822	-0.059082	0.429326	-0.213881	0.358810	-0.3891
ZN	-0.195751	1.000000	-0.523917	-0.036383	-0.520639	0.297024	-0.576421	0.6717
INDUS	0.402822	-0.523917	1.000000	0.054766	0.794809	-0.365529	0.668595	-0.7400
CHAS	-0.059082	-0.036383	0.054766	1.000000	0.094851	0.110079	0.089192	-0.1034
NOX	0.429326	-0.520639	0.794809	0.094851	1.000000	-0.338535	0.727605	-0.7635
RM	-0.213881	0.297024	-0.365529	0.110079	-0.338535	1.000000	-0.270840	0.2378
AGE	0.358810	-0.576421	0.668595	0.089192	0.727605	-0.270840	1.000000	-0.7450
DIS	-0.389106	0.671706	-0.740027	-0.103407	-0.763542	0.237893	-0.745027	1.0000
RAD	0.623634	-0.301859	0.589127	-0.012382	0.624151	-0.193630	0.464296	-0.5068
TAX	0.581076	-0.309810	0.715600	-0.042537	0.693280	-0.271667	0.524742	-0.5602
PTRATIO	0.293197	-0.368373	0.337121	-0.159898	0.237800	-0.277370	0.311267	-0.2880
LSTAT	0.453143	-0.406844	0.597411	-0.060225	0.606360	-0.614479	0.615702	-0.5128
MEDV	-0.391368	0.352029	-0.462905	0.198515	-0.473486	0.664089	-0.414797	0.2887

In [17]:

```
# Display the heatmap for normalized data
plt.figure(figsize=(10, 8))
sns.heatmap(normalized_correlation_matrix, annot=True, cmap='coolwarm', fmt
plt.title('Correlation Matrix Heatmap (Normalized Data)')
plt.show()

# the correlation values have not changed indicating that the variables in
```



Linear Regression

Split data in training and test dataset

In [18]:

```
x = housing_df_new.drop('MEDV', axis =1)
y = housing_df_new['MEDV']

# split the dataset in training and test dataset, 80-20 split.
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, ra
```

Fit linear regression model

In [19]:
```
model = LinearRegression()

# Fit the model on the training data
model.fit(x_train, y_train)
```

Out[19]:
```
▾ LinearRegression
LinearRegression()
```

Predict and Evaluate the model

In [20]:
```
y_pred = model.predict(x_test)

# to understand how well the model is doing, we compare the MSE of the mode
# calulcate the mean value of the training dataset
mean_baseline = y_train.mean()

# create predictions based on the mean for the test set
y_pred_baseline = [mean_baseline] * len(y_test)

# evaluate the baseline model using MSE
mse_baseline = mean_squared_error([mean_baseline] * len(y_test), y_test)
print(f"Mean Squared Error (Baseline model): {mse_baseline}")

# evaluate the model (for example, using Mean Squared Error)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error:\t\t     {mse}")

# this indicates the MSE value calculated is relatively low.
```

```
Mean Squared Error (Baseline model): 55.92864108270859
Mean Squared Error:                  12.104631354705866
```

Extract the important features

In [21]:
```
# get the coefficients from the model
coefficients = model.coef_

# create a DataFrame to display feature names and their corresponding coeff
feature_importance = pd.DataFrame({
    'Feature': x.columns,
    'Coefficient': coefficients
})

# display the DataFrame
print(feature_importance)

# we ignore the coefficient for CAT_MEDV as the CAT_MEDV variable is derive
# The NOX has the highest negative coefficient, this states that an increas
# The second variable CHAS has a positive coefficient of 2.79 indicating th
```

```
     Feature  Coefficient
0       CRIM    -0.122795
1         ZN    -0.005215
2      INDUS     0.137449
3       CHAS     2.790639
4        NOX   -15.558904
5         RM     0.542332
6        AGE     0.000753
7        DIS    -0.607606
8        RAD     0.152872
9        TAX    -0.006184
10   PTRATIO    -0.551640
11     LSTAT    -0.533303
12  CAT_MEDV    12.696295
```

In [ ]: