In [1]:
```
import pandas as pd
import numpy as np


from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("boston.csv")

df.head()
```

Out[2]:

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

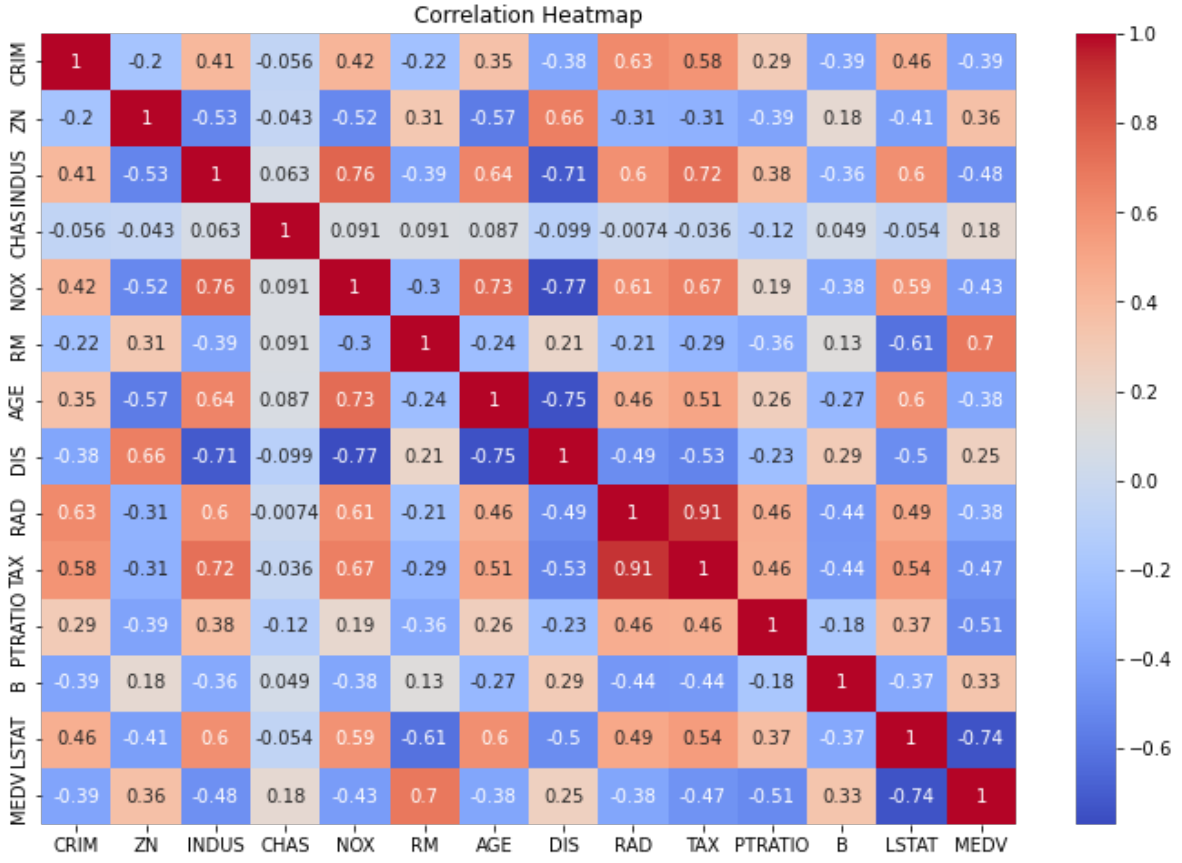
In [3]:
```
df.info()
df.describe()
df.isnull().sum()
```

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

```
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [4]:

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



In [5]:

```
X = df.drop('MEDV', axis=1)

y = df['MEDV']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [7]:

```
# Create the model
lr_model = LinearRegression()

# Train the model
lr_model.fit(X_train, y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:
```
y_pred = lr_model.predict(X_test)


pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).head()
```

Out[8]:

	Actual	Predicted
173	23.6	28.996724
274	32.4	36.025565
491	13.6	14.816944
72	22.8	25.031979
452	16.1	18.769880

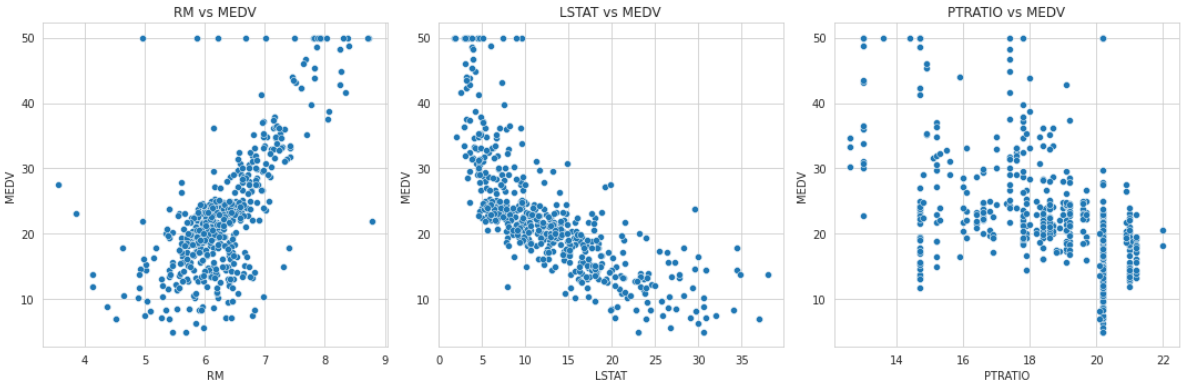
In [9]:
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)


rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)


r2 = r2_score(y_test, y_pred)
print("R-squared score:", r2)
```

```
Mean Squared Error: 24.29111947497371
Root Mean Squared Error: 4.928602182665355
R-squared score: 0.6687594935356294
```

In [10]:
```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color=
plt.xlabel("Actual MEDV")
plt.ylabel("Predicted MEDV")
plt.title("Actual vs Predicted Home Prices")
plt.show()
```

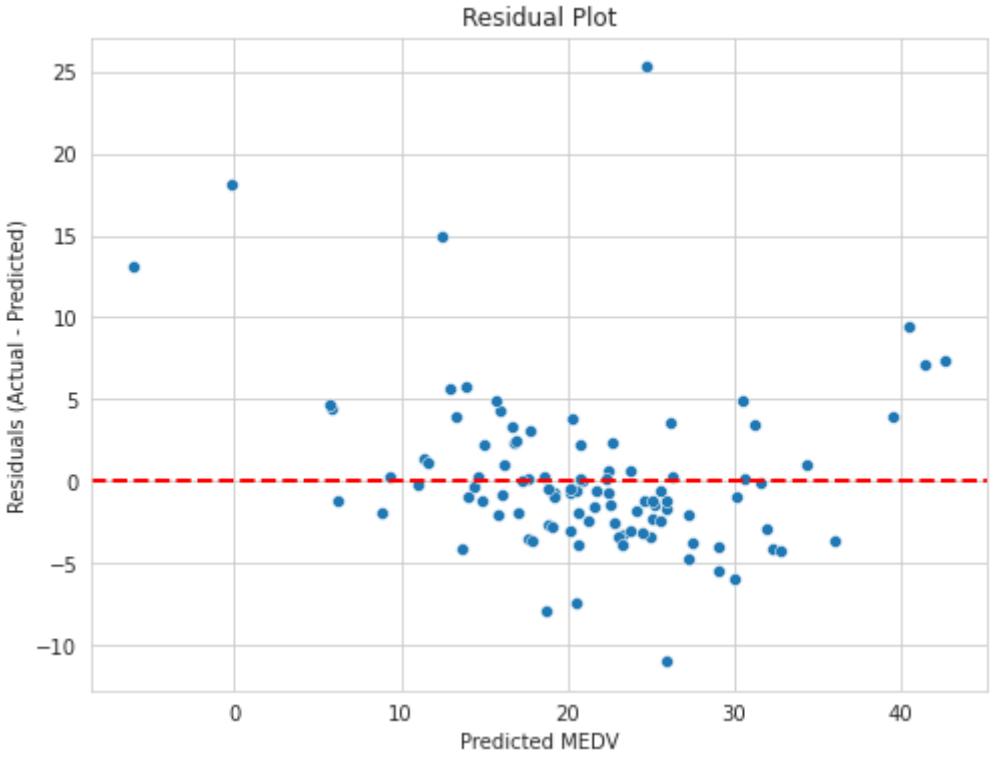
In [12]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set plot style
sns.set_style("whitegrid")

# Select features to plot
features_to_plot = ['RM', 'LSTAT', 'PTRATIO']

# Create subplots
plt.figure(figsize=(15,5))

for i, feature in enumerate(features_to_plot):
    plt.subplot(1, 3, i+1)  # 1 row x 3 columns
    sns.scatterplot(x=df[feature], y=df['MEDV'])
    plt.xlabel(feature)
    plt.ylabel('MEDV')
    plt.title(f'{feature} vs MEDV')

plt.tight_layout()
plt.show()
```



In [13]:

```
# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(8,6))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(0, color='red', linestyle='--', linewidth=2)
plt.xlabel("Predicted MEDV")
plt.ylabel("Residuals (Actual - Predicted)")
plt.title("Residual Plot")
plt.show()
```



In [ ]: