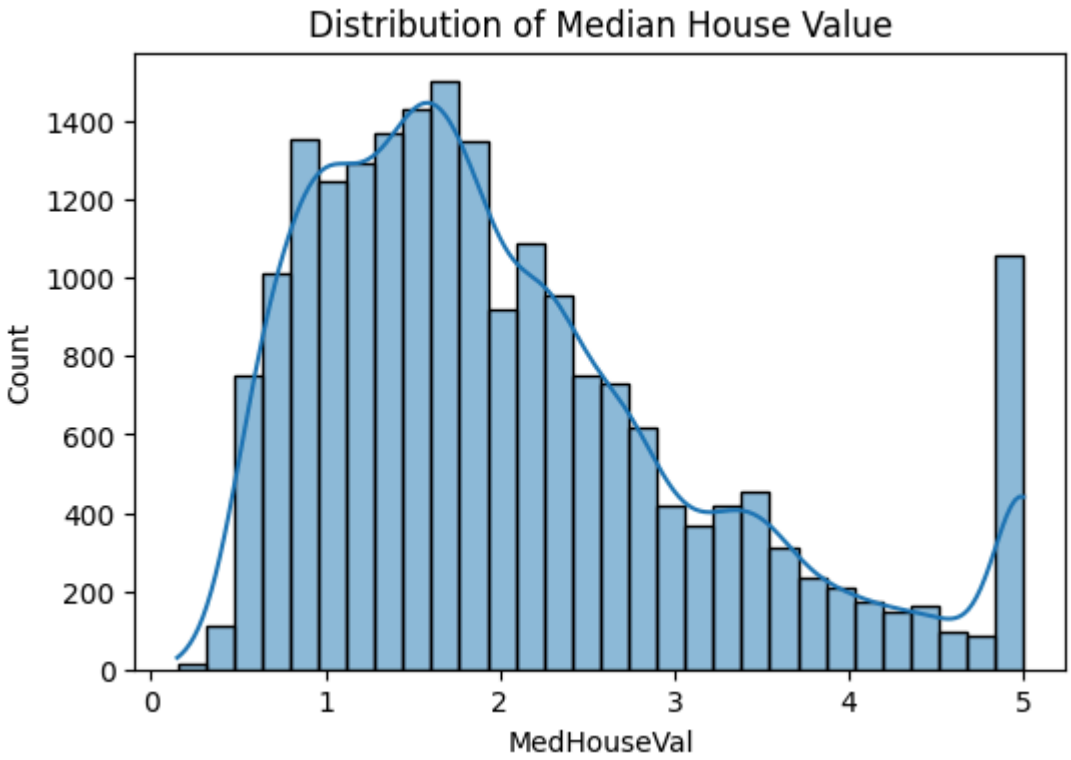
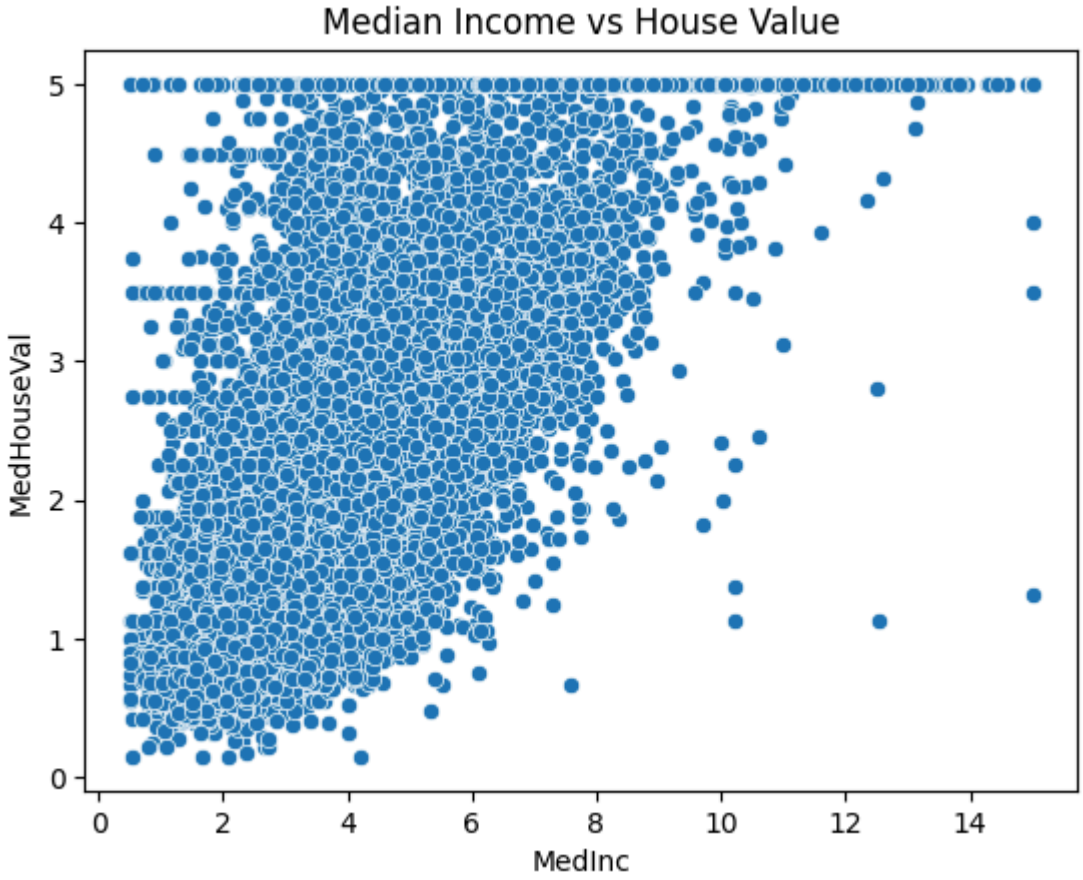
In [1]:
```
# 1. Load and Understand the Dataset
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True)
df = data.frame

print(df.head())
print("Shape:", df.shape)
print("Data types:\n", df.dtypes)
print("Missing values:\n", df.isnull().sum())
```

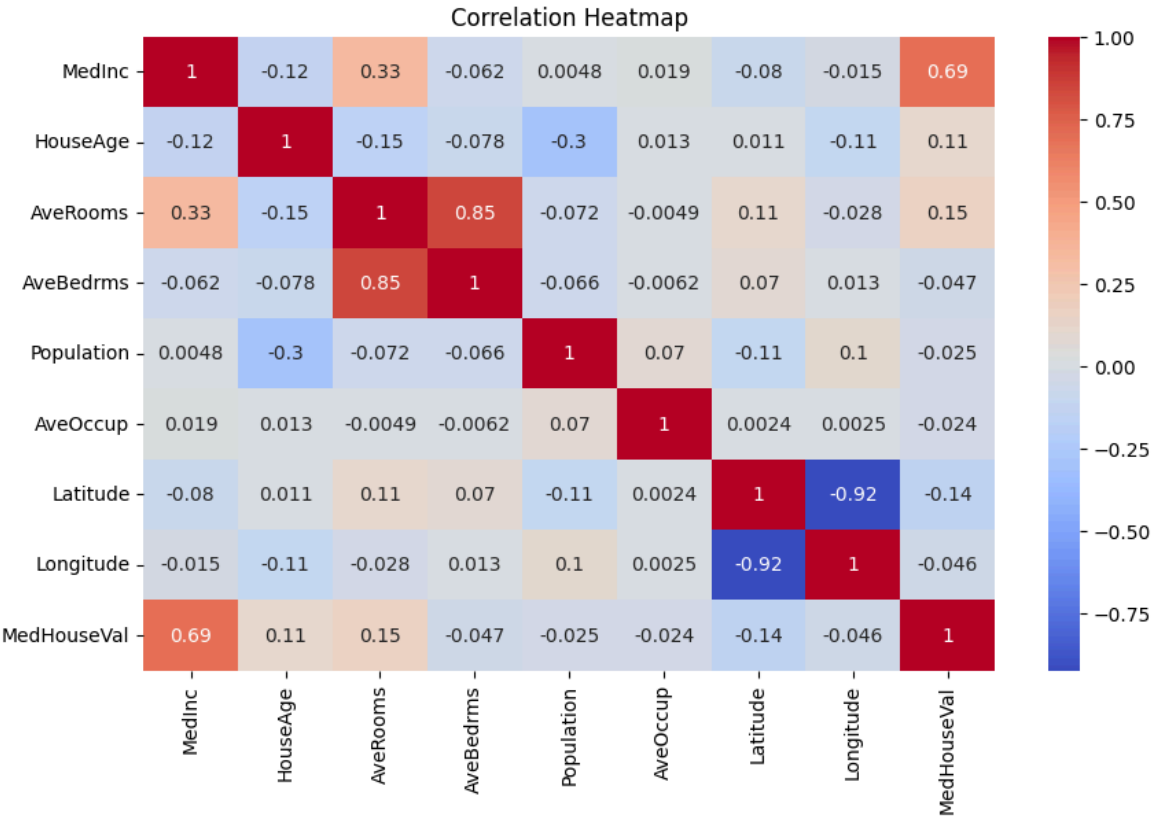
```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  MedHouseVal
0    -122.23        4.526
1    -122.22        3.585
2    -122.24        3.521
3    -122.25        3.413
4    -122.25        3.422
Shape: (20640, 9)
Data types:
 MedInc          float64
HouseAge        float64
AveRooms        float64
AveBedrms       float64
Population      float64
AveOccup        float64
Latitude        float64
Longitude       float64
MedHouseVal     float64
dtype: object
Missing values:
 MedInc          0
HouseAge        0
AveRooms        0
AveBedrms       0
Population      0
AveOccup        0
Latitude        0
Longitude       0
MedHouseVal     0
dtype: int64
```

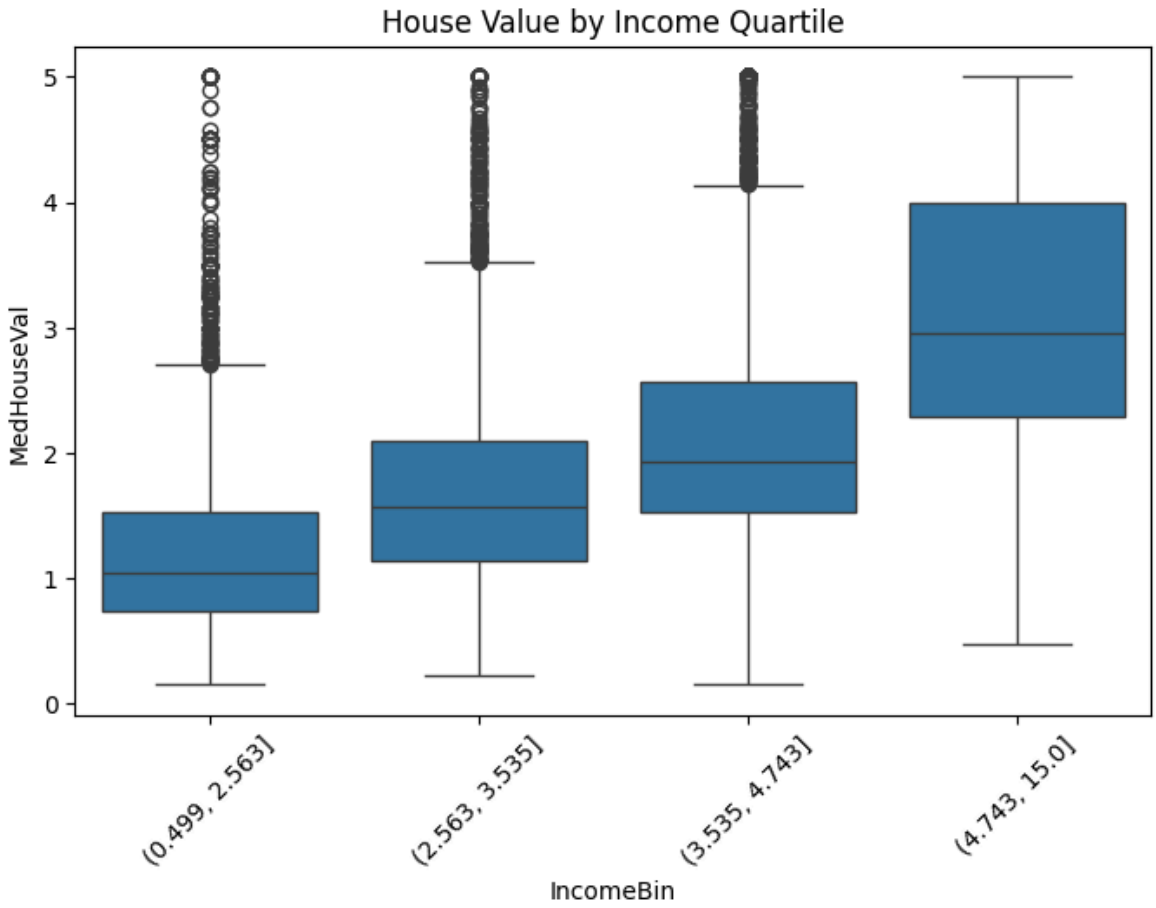
In [2]:
```
# 2. Exploratory Data Analysis (EDA)
plt.figure(figsize=(6,4))
sns.histplot(df['MedHouseVal'], bins=30, kde=True)
plt.title("Distribution of Median House Value")
plt.show()

# Scatter plot:
```

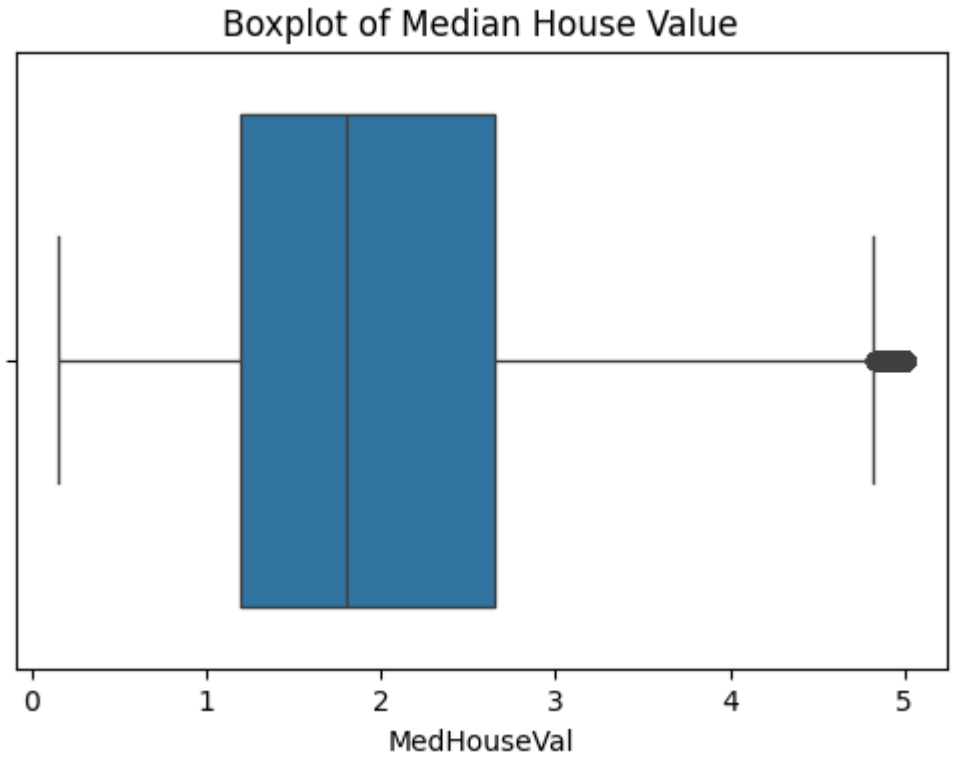
```
sns.scatterplot(x='MedInc', y='MedHouseVal', data=df)
plt.title("Median Income vs House Value")
plt.show()

# Correlation heatmap:
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

# Boxplot:
df['IncomeBin'] = pd.qcut(df['MedInc'], q=4)
plt.figure(figsize=(8,5))
sns.boxplot(x='IncomeBin', y='MedHouseVal', data=df)
plt.title("House Value by Income Quartile")
plt.xticks(rotation=45)
plt.show()

# Visual outlier detection
plt.figure(figsize=(6,4))
sns.boxplot(x=df['MedHouseVal'])
plt.title("Boxplot of Median House Value")
plt.show()
```




Median Income vs House Value
MedHouseVal
MedInc
0
1
2
3
4
5
0
2
4
6
8
10
12
14


Correlation Heatmap
MedInc 1 -0.12 0.33 -0.062 0.0048 0.019 -0.08 -0.015 0.69
HouseAge -0.12 1 -0.15 -0.078 -0.3 0.013 0.011 -0.11 0.11
AveRooms 0.33 -0.15 1 0.85 -0.072 -0.0049 0.11 -0.028 0.15
AveBedrms -0.062 -0.078 0.85 1 -0.066 -0.0062 0.07 0.013 -0.047
Population 0.0048 -0.3 -0.072 -0.066 1 0.07 -0.11 0.1 -0.025
AveOccup 0.019 0.013 -0.0049 -0.0062 0.07 1 0.0024 0.0025 -0.024
Latitude -0.08 0.011 0.11 0.07 -0.11 0.0024 1 -0.92 -0.14
Longitude -0.015 -0.11 -0.028 0.013 0.1 0.0025 -0.92 1 -0.046
MedHouseVal 0.69 0.11 0.15 -0.047 -0.025 -0.024 -0.14 -0.046 1
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75





In [9]:
```
# 3. Data Preprocessing

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Select features
features = ['MedInc', 'AveRooms', 'AveOccup', 'HouseAge', 'AveBedrms']
X = df[features]
```

```
y = df['MedHouseVal']

# Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
print(X_train, X_test, y_train, y_test)
```

```
[[-0.32165429 -0.16625886  0.05980847  0.34647803 -0.19045099]
 [-0.03061993 -0.3861807  -0.12830597  1.61780729 -0.1174719 ]
 [ 0.1503488   0.08764093 -0.03345346 -1.95780625 -0.2354004 ]
 ...
 [-0.49283239 -0.58294927  0.02517025  0.58485227 -0.03582776]
 [ 0.97302487  0.39058403  0.01042151 -1.08376738 -0.06055424]
 [-0.68174943 -0.81905034 -0.09262259  1.85618152 -0.07997264]] [[-1.15248922 -0.
49989596  0.07768129 -0.2891866  -0.15697642]
 [-0.70501534 -0.1574771  -0.03763415  0.10810379  0.20430076]
 [-0.20558796 -0.5868143  -0.164679    1.85618152  0.1882305 ]
 ...
 [ 2.82092723  0.73104024 -0.02697551 -0.2891866  -0.31545104]
 [-0.57147326 -0.05657389 -0.04641411  0.58485227 -0.23961371]
 [-0.16768875 -0.58209158  0.06347664 -0.92485123 -0.13334689]] 14196    1.030
8267     3.821
17445    1.726
14265    0.934
2271     0.965
         ...
11284    2.292
11964    0.978
5390     2.221
860      2.835
15795    3.250
Name: MedHouseVal, Length: 16512, dtype: float64 20046    0.47700
3024     0.45800
15663    5.00001
20484    2.18600
9814     2.78000
          ...
15362    2.63300
16623    2.66800
18086    5.00001
2144     0.72300
3665     1.51500
Name: MedHouseVal, Length: 4128, dtype: float64
```

In [5]:
```
# 4 & 5. Build and Train Two Regression Models

from sklearn.linear_model import LinearRegression

# Model A: Simple Linear Regression
X_simple = df[['MedInc']]
X_simple_scaled = scaler.fit_transform(X_simple)
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_simple_scaled, y,

model_a = LinearRegression()
model_a.fit(X_train_s, y_train_s)

# Model B: Multiple Linear Regression
```

```
model_b = LinearRegression()
model_b.fit(X_train, y_train)
```

Out[5]:

▾ LinearRegression

▶ Parameters

In [6]:

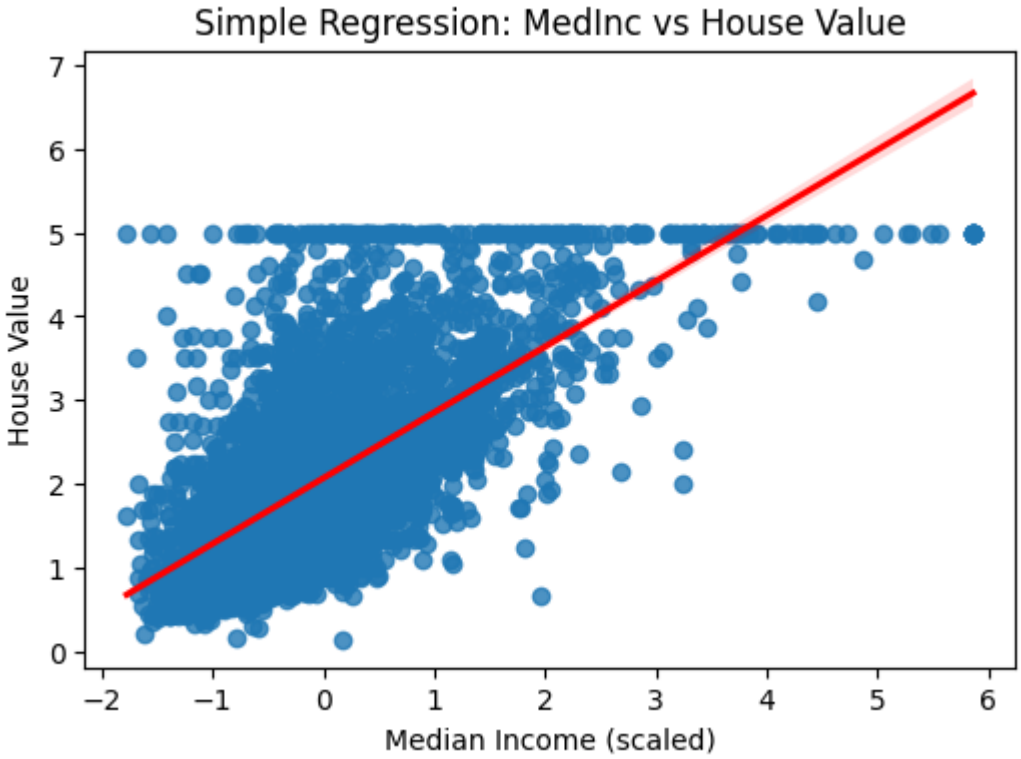
```
# 6. Evaluate the Models

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

def evaluate(model, X_test, y_test, name="Model"):
    y_pred = model.predict(X_test)
    print(f"\n{name} Evaluation:")
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("MSE:", mean_squared_error(y_test, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
    print("R2 Score:", r2_score(y_test, y_pred))
    return y_pred

y_pred_a = evaluate(model_a, X_test_s, y_test_s, "Simple Linear Regression")
y_pred_b = evaluate(model_b, X_test, y_test, "Multiple Linear Regression")
```

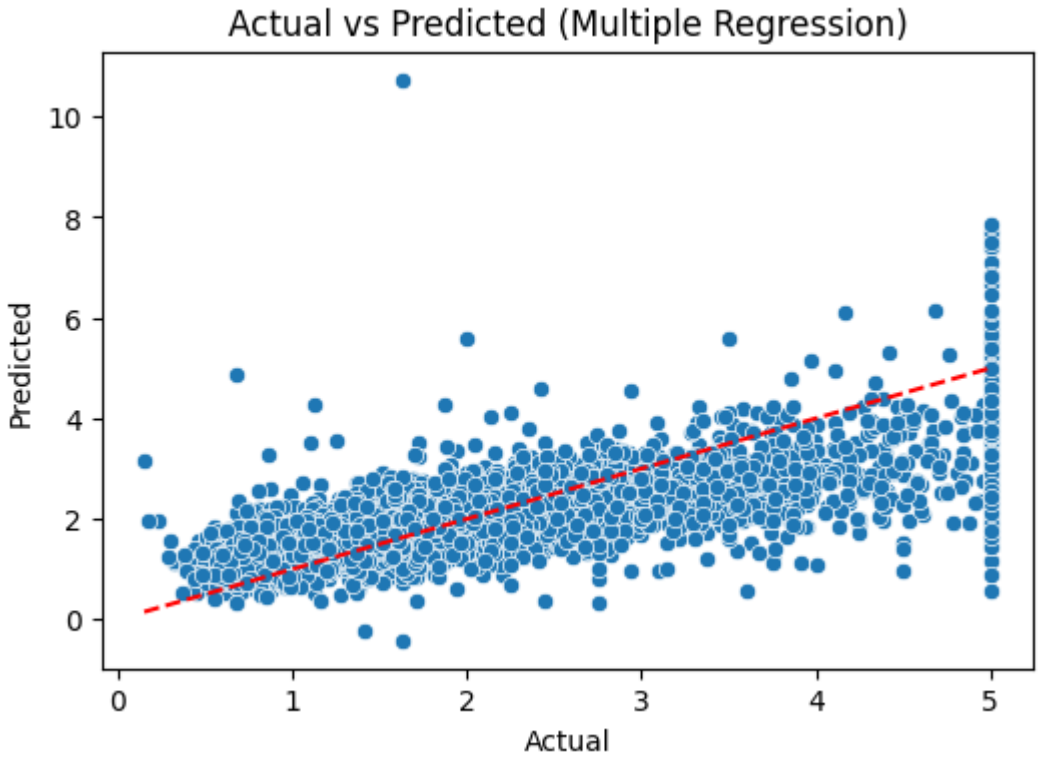
```
Simple Linear Regression Evaluation:
MAE: 0.629908653009376
MSE: 0.7091157771765548
RMSE: 0.8420901241414454
R2 Score: 0.45885918903846656

Multiple Linear Regression Evaluation:
MAE: 0.5795921844399792
MSE: 0.6423670622489219
RMSE: 0.8014780485134462
R2 Score: 0.5097965040568945
```

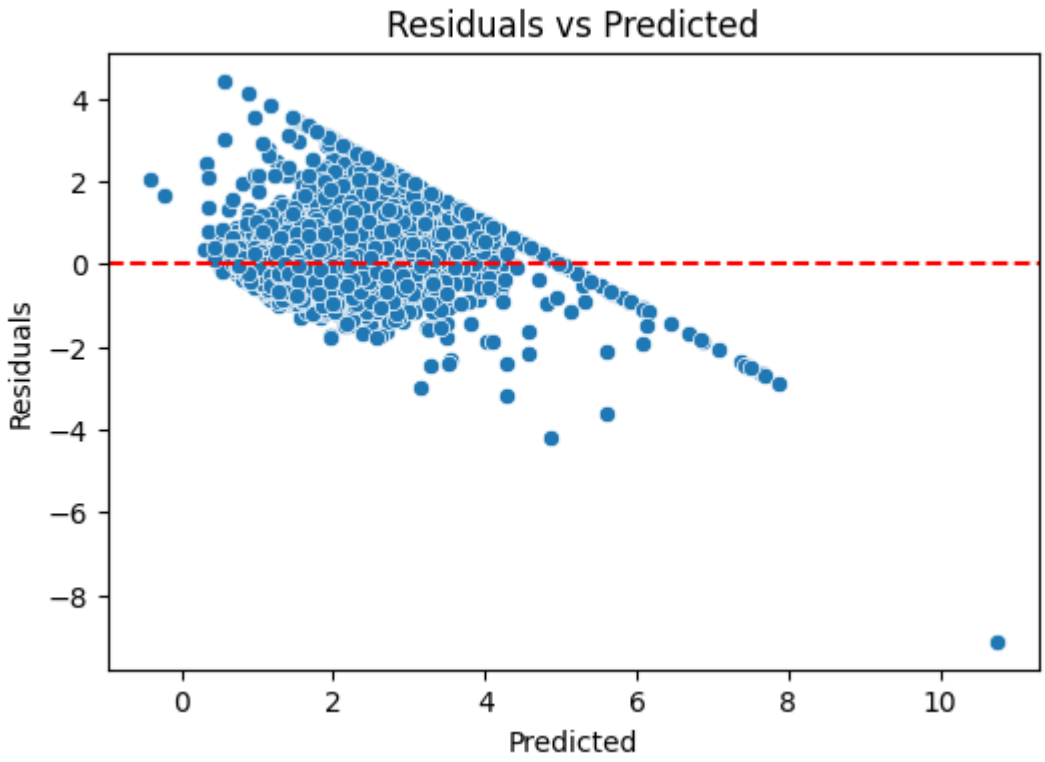
In [7]:

```
# 8. Required Visualizations

# Scatter plot + regression line for simple regression
plt.figure(figsize=(6,4))
sns.regplot(x=X_test_s.flatten(), y=y_test_s, line_kws={"color": "red"})
plt.title("Simple Regression: MedInc vs House Value")
plt.xlabel("Median Income (scaled)")
plt.ylabel("House Value")
plt.show()

# Actual vs Predicted plot for multiple regression
plt.figure(figsize=(6,4))
sns.scatterplot(x=y_test, y=y_pred_b)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted (Multiple Regression)")
plt.show()

# Residual plot
residuals = y_test - y_pred_b
plt.figure(figsize=(6,4))
sns.scatterplot(x=y_pred_b, y=residuals)
plt.axhline(0, color='red', linestyle='--')
```

```
plt.title("Residuals vs Predicted")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```






Residuals vs Predicted
Residuals
Predicted
4
2
0
−2
−4
−6
−8
0
2
4
6
8
10

In [ ]: