ABOUT DATASET 'BOSTON HOUSING'

Boston Housing Dataset Analysis

Variable Notes

1. crim: Per capita crime rate by town.
2. zn: Proportion of residential land zoned for lots over 25,000 sq. ft.
3. indus: Proportion of non-retail business acres per town.
4. chas: Charles River dummy variable (1 if tract bounds river; 0 otherwise).
5. nox: Nitric oxides concentration (parts per 10 million).
6. rm: Average number of rooms per dwelling.
7. age: Proportion of owner-occupied units built prior to 1940.
8. dis: Weighted distances to five Boston employment centers.
9. rad: Index of accessibility to radial highways.
10. tax: Full-value property tax rate per 10000.
11. ptratio: Pupil-teacher ratio by town.
12. b: proportion of black people by town.
13. lstat: Percentage of lower status of the population.
14. medv: Median value of owner-occupied homes in $1000's.

1. Steps for Analysis
2. Load the Data
3. Preprocess the Data
4. Exploratory Data Analysis (EDA)
5. Feature Engineering
6. Model Building and Evaluation
7. Insights and Conclusions

In [46]:
```
import pandas as pd
import numpy as np
```

In [25]:
```
# Load the dataset
columns = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
data_url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing
boston = pd.read_csv(data_url, names=columns, skiprows=1)
```

In [26]:
```
boston.head()
```

Out[26]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [27]:
```
boston.tail()
```

Out[27]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

In [28]:
```
boston.info()
```

```
<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    int64  
 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(11), int64(3)
memory usage: 55.5 KB
```

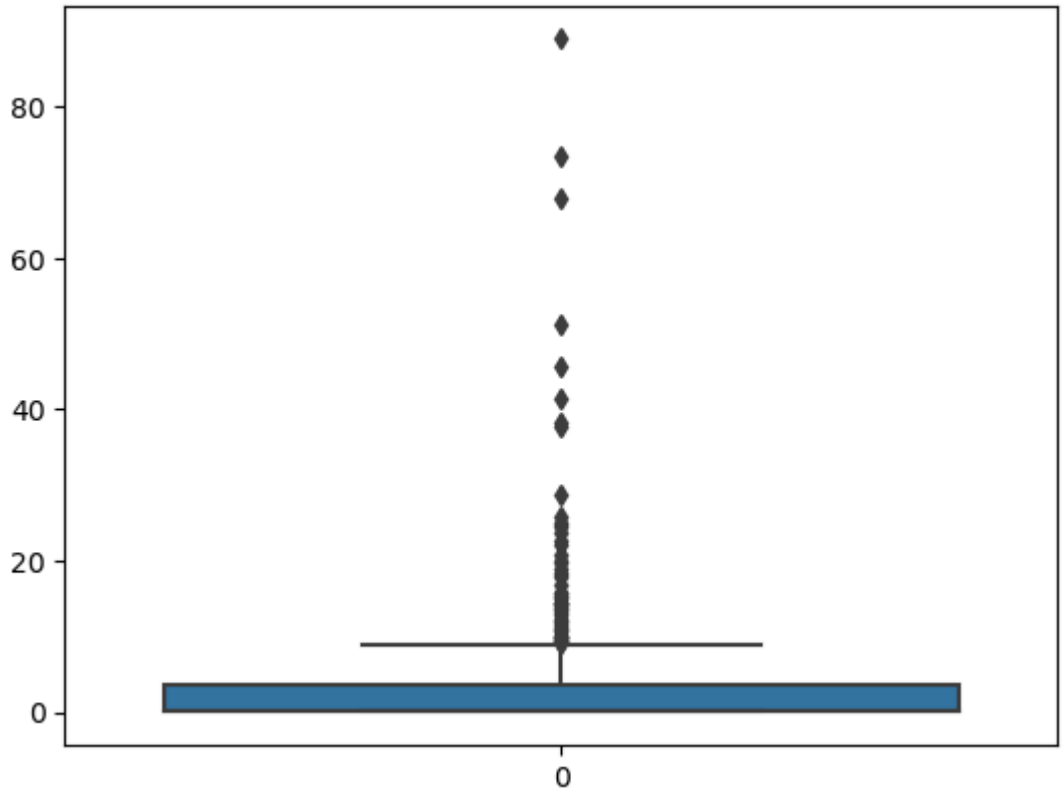
In [29]:
```
boston.describe()
```

Out[29]:

	crim	zn	indus	chas	nox	rm	age	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79!
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10!
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12!
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20!
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18!
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12(

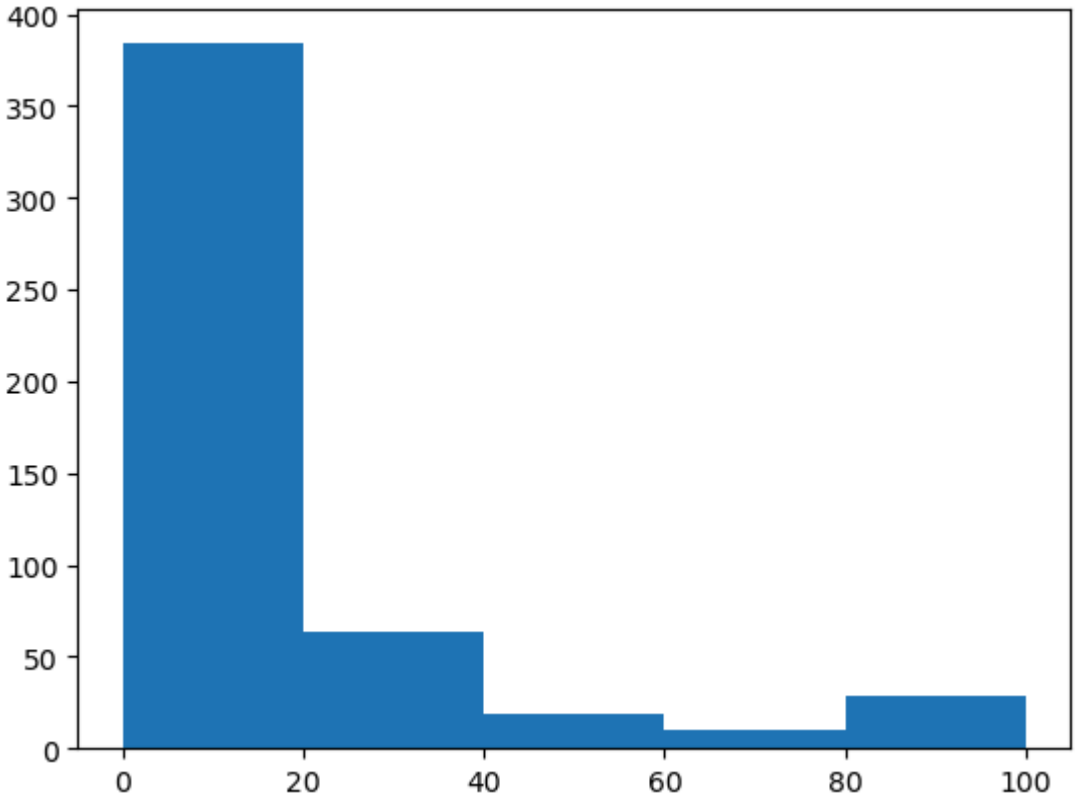
In [57]:
```
sns.boxplot(boston['crim'])
```

Out[57]:
```
<Axes: >
```



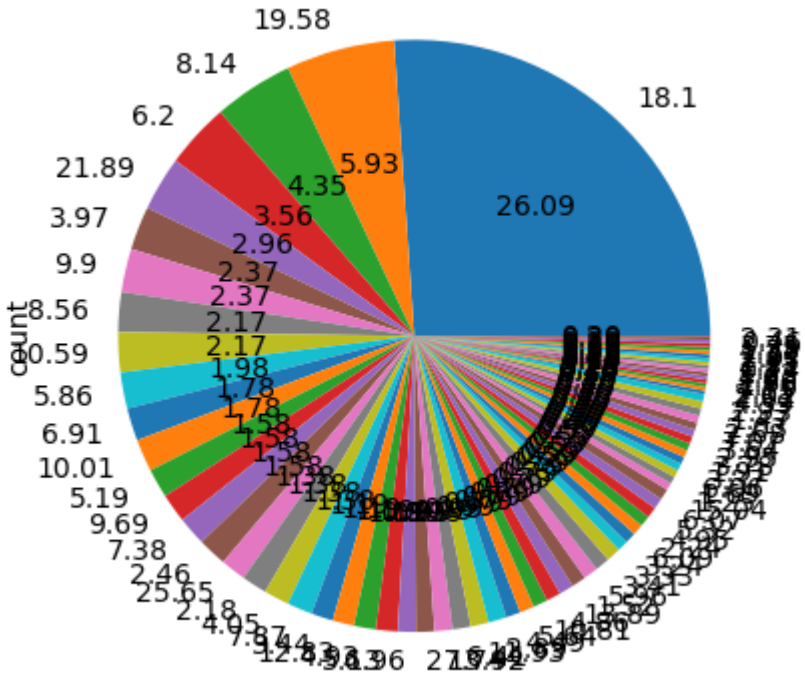
In [58]:
```
plt.hist(boston['zn'],bins=5)
```

Out[58]:
```
(array([384.,  64.,  19.,  10.,  29.]),
 array([  0.,  20.,  40.,  60.,  80., 100.]),
 <BarContainer object of 5 artists>)
```



In [59]:
```
boston['indus'].value_counts().plot(kind='pie', autopct='%.2f')
```

Out[59]:
```
<Axes: ylabel='count'>
```



PRE PROCESSING

In [33]:
```
# Check for missing values
print(boston.isnull().sum())
```

```
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
```

SCALING

In [34]:
```
from sklearn.preprocessing import StandardScaler
```

In [35]:
```
# SCALER THE FEATURES
scaler = StandardScaler()
```

In [36]:
```
feature_columns = columns[:-1]
boston_scaled = scaler.fit_transform(boston[feature_columns])
```

In [37]:
```
# CREATE A DATAFRAME WITH SCALED VALUES
boston_scaled_df = pd.DataFrame(boston_scaled, columns = columns[:-1])
boston_scaled_df['medv'] = boston['medv']
```

In [38]:
```
boston_scaled_df.head()
```

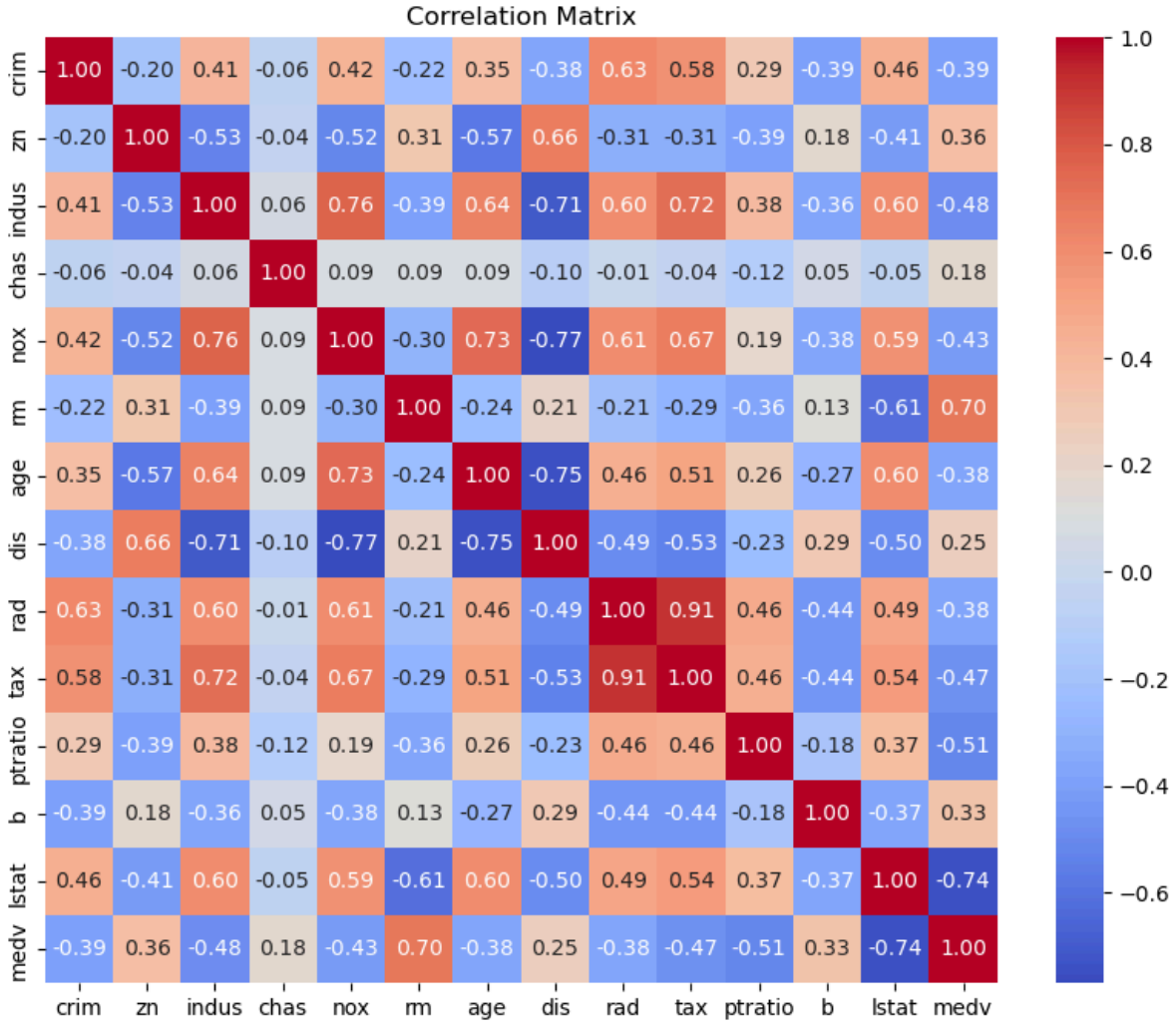
Out[38]:

	crim	zn	indus	chas	nox	rm	age	dis	rad
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922

EDA

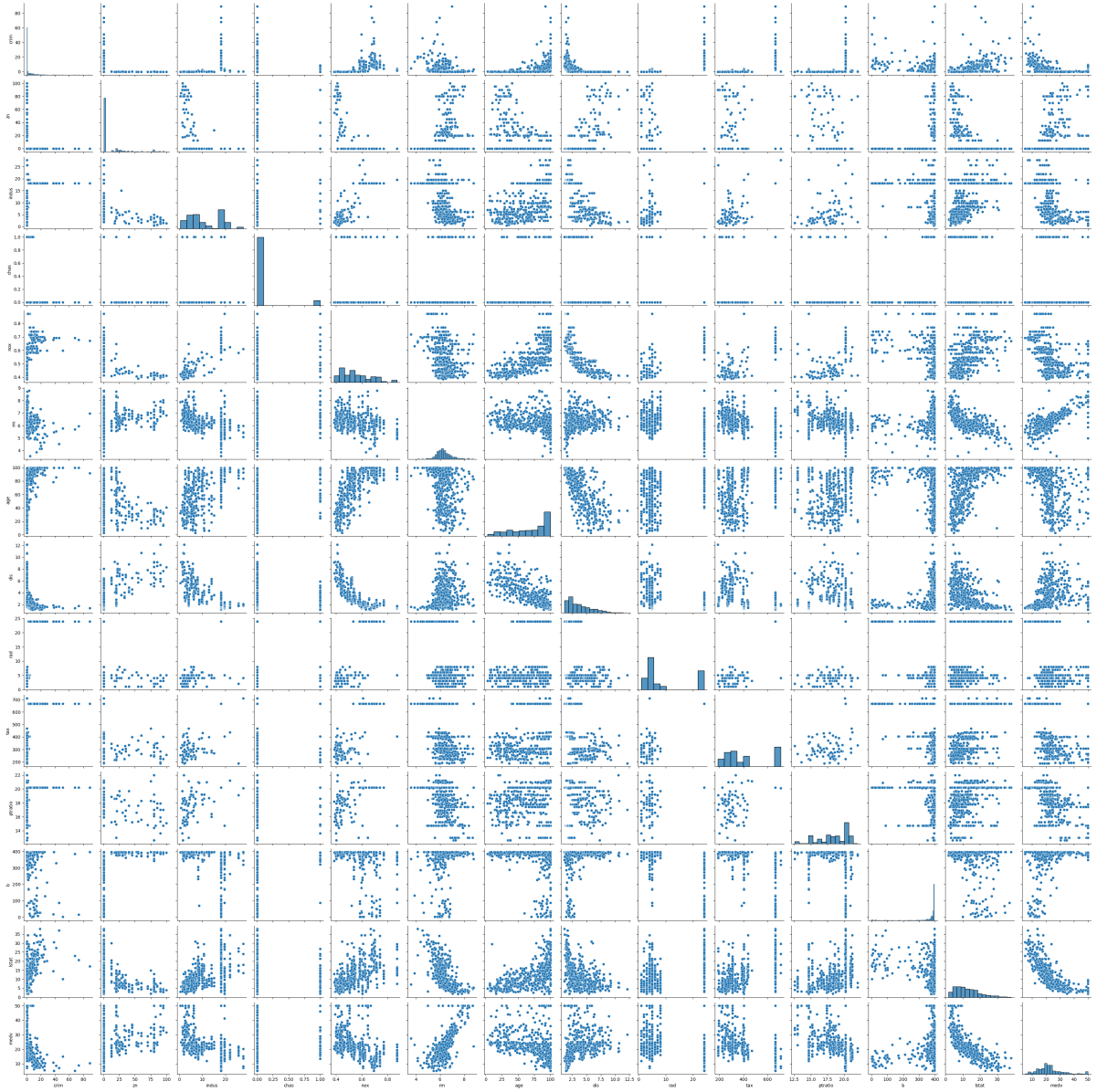
In [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [41]:
```
# CORRELATION HEAPMAP
plt.figure(figsize=(10, 8))
sns.heatmap(boston.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

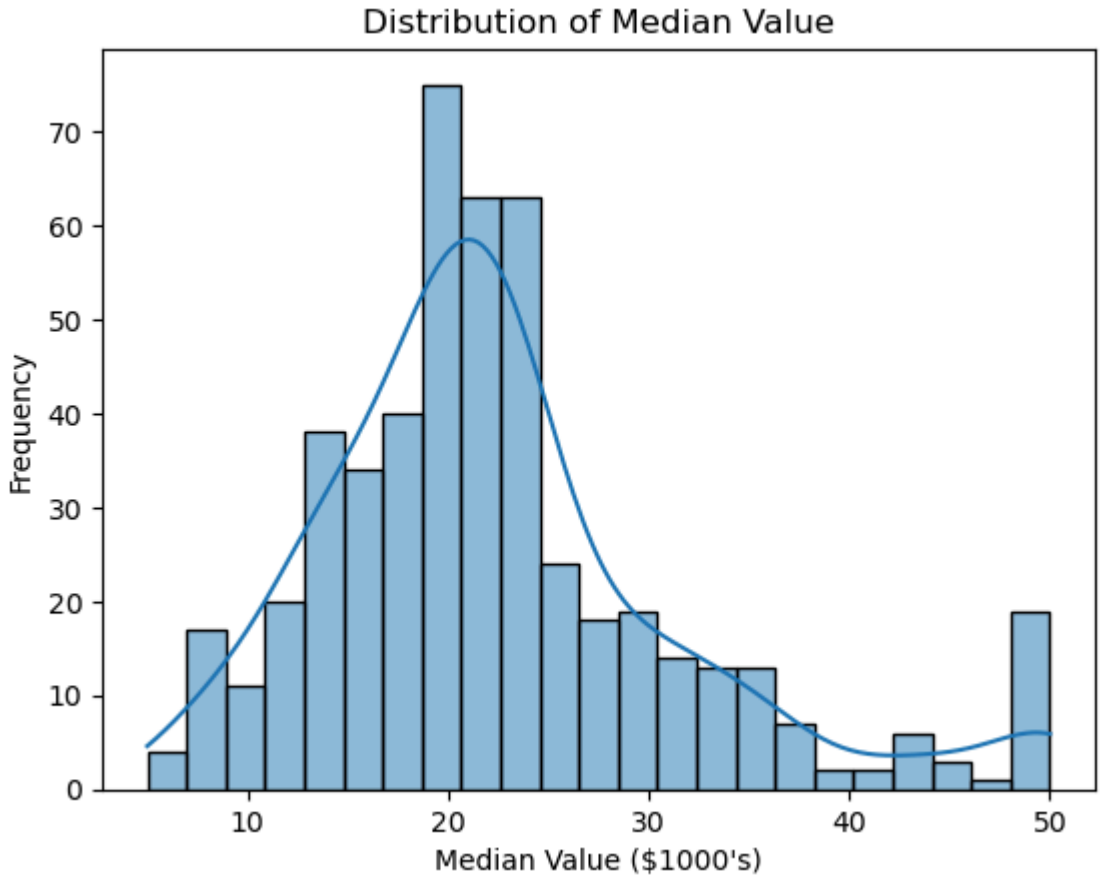


In [42]:
```
# Pair plot
sns.pairplot(boston)
plt.show()
```

```
C:\Users\Nishita Bala\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWar
ning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [43]:
```
# Distribution of the target variable
sns.histplot(boston['medv'], kde=True)
plt.title("Distribution of Median Value")
plt.xlabel("Median Value ($1000's)")
plt.ylabel("Frequency")
plt.show()
```



Key Insights:

1. medv has a strong negative correlation with lstat (lower status of the population) and crim (crime rate).
2. rm (average number of rooms per dwelling) shows a strong positive correlation with medv.
3. medv is roughly normally distributed but appears to have some skewness and possible outliers.

FEATURE ENGINEERING

In [44]:
```
# Adding a new feature: room per capita crime rate
boston['rm_per_crim'] = boston['rm'] / boston['crim']
```

In [47]:
```
# Log transform to handle skewness
boston['log_crim'] = np.log1p(boston['crim'])
```

In [48]:
```
# Update the scaled DataFrame
boston_scaled_df['rm_per_crim'] = scaler.fit_transform(boston[['rm_per_crim']])
boston_scaled_df['log_crim'] = scaler.fit_transform(boston[['log_crim']])
```

MODEL BUILDING AND EVALUATION

In [49]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
```

In [50]:
```
# Split the data
X = boston.drop('medv', axis=1)
y = boston['medv']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [51]:
```
# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[51]:
```
▾ LinearRegression

LinearRegression()
```

In [52]:
```
# Make predictions
y_pred = model.predict(X_test)
```

In [53]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

Model Performance:

1. The RMSE provides a measure of how well the model's predictions match the actual values. Lower values indicate better fit.
2. The R^2 score indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. A value closer to 1 indicates a better fit.

In [54]:
```
print(f'RMSE: {rmse:.2f}')
print(f'R^2 Score: {r2:.2f}')
```

```
RMSE: 4.95
R^2 Score: 0.67
```

Insights and Conclusions

1. Crime Rate (crim): A higher crime rate is associated with lower house prices. This suggests that safety is a significant factor for homeowners.
2. Number of Rooms (rm): More rooms are associated with higher house prices. This implies that larger homes are valued higher.
3. Lower Status Population (lstat): A higher percentage of lower status residents correlates with lower home values. This could indicate socio-economic disparities influencing housing markets.
4. Proximity to Employment Centers (dis): Greater distances to employment centers are correlated with lower house prices, highlighting the importance of accessibility and commute times.
5. Property Taxes (tax): Higher tax rates are negatively correlated with house prices, possibly reflecting the burden of higher taxes on property values.
6. Nitric Oxides Concentration (nox): Higher pollution levels are associated with lower home values, indicating environmental factors play a role in property pricing.