👋 Hi, I'm **Ashish Mishra**

🎯 Internship in AI/ML

🏢 at ShadowFox

Boston House price prediction

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Ashish Mishra\OneDrive\Desktop\ShadowFox\Shadowfox_AI-M

df.head()
```

Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	I
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 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    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

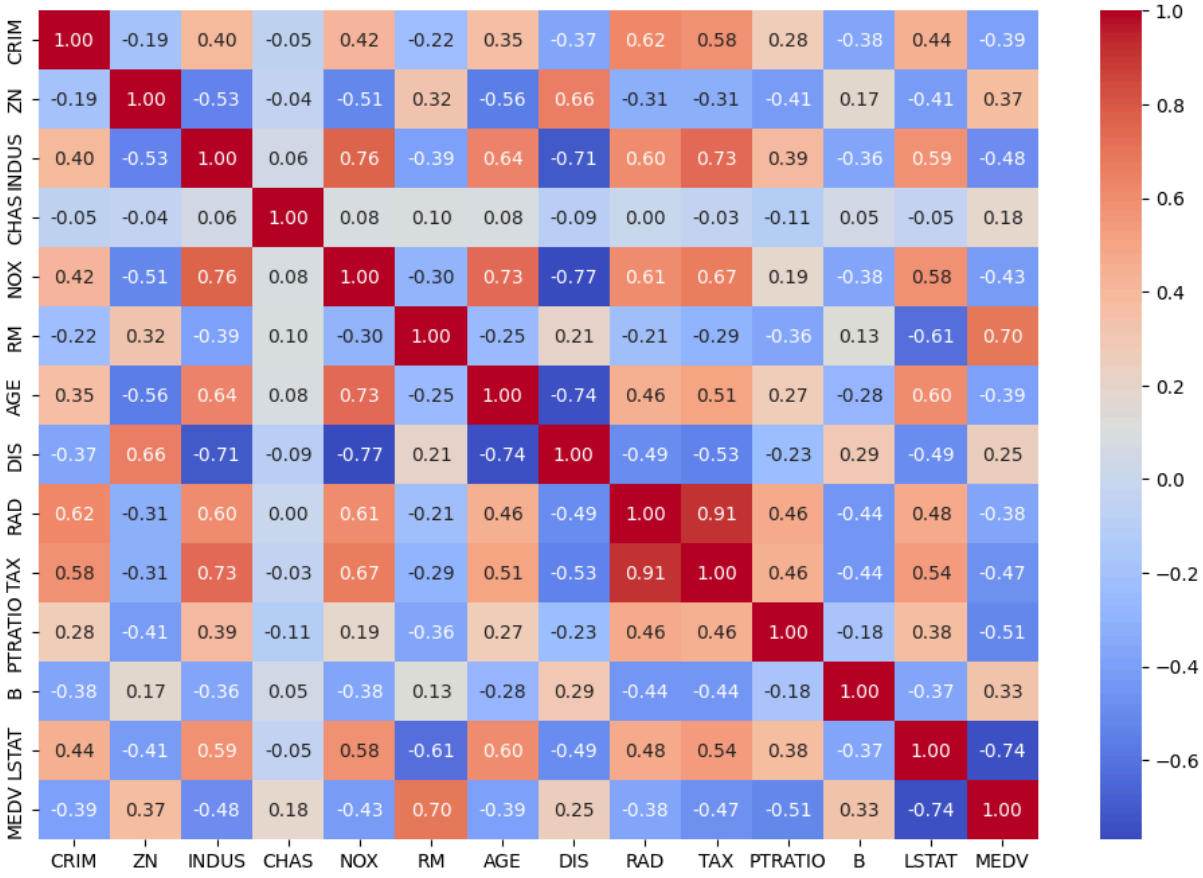
	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	486.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.518519	3.795043	9.549407	408.237154	18.455534	356.674032	
std	27.999513	2.105710	8.707259	168.537116	2.164946	91.294864	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.175000	2.100175	4.000000	279.000000	17.400000	375.377500	
50%	76.800000	3.207450	5.000000	330.000000	19.050000	391.440000	
75%	93.975000	5.188425	24.000000	666.000000	20.200000	396.225000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	

	LSTAT	MEDV
count	486.000000	506.000000
mean	12.715432	22.532806
std	7.155871	9.197104
min	1.730000	5.000000
25%	7.125000	17.025000
50%	11.430000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

In [4]:
```
print("Missing values:", df.isnull().sum())
```

```
Missing values: CRIM       20
ZN          20
INDUS       20
CHAS        20
NOX          0
RM           0
AGE         20
DIS          0
RAD          0
TAX          0
PTRATIO      0
B            0
LSTAT       20
MEDV         0
dtype: int64
```

In [5]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.show()
```



In [6]:
```
X = df.drop('MEDV', axis=1)  # Features
y = df['MEDV']  # Target variable
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [8]:
```
imputer = SimpleImputer(strategy='median')
```

In [9]:
```
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

In [10]:
```
model = LinearRegression()
model.fit(X_train_imputed, y_train)
```

Out[10]:
```
LinearRegression
LinearRegression()
```

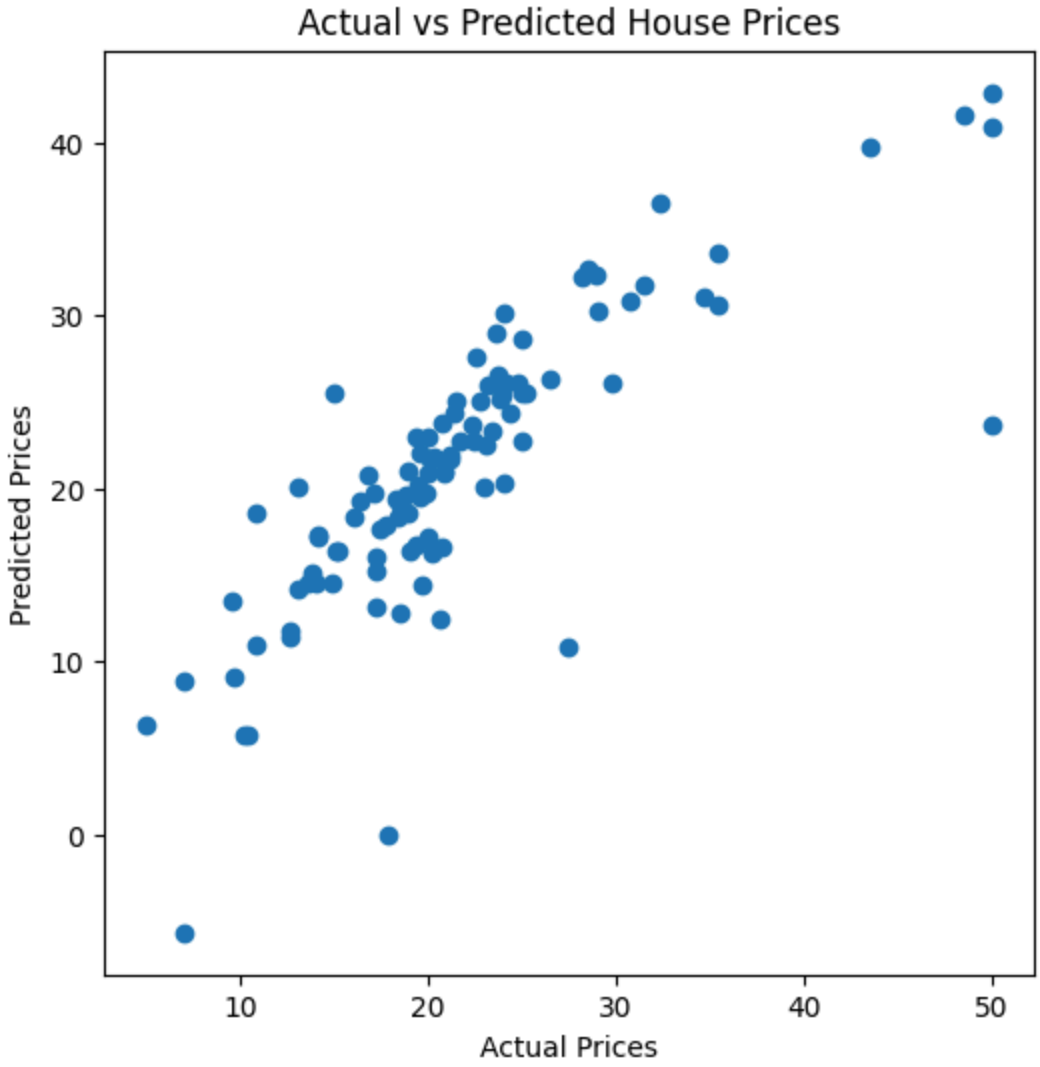
In [11]:
```
y_pred = model.predict(X_test_imputed)
```

In [12]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

In [13]:
```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error (MSE): 24.983445419912112
Root Mean Squared Error (RMSE): 4.998344267846314
R-squared: 0.659318743105165
```

In [14]:
```
plt.figure(figsize=(6, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()
```


Actual vs Predicted House Prices
Predicted Prices
Actual Prices
0
10
20
30
40
10
20
30
40
50