```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler , MinMaxScaler
```

```
df = pd.read_csv('Housing.csv')
```

```
df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furn
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.describe()
```

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
df.isnull().sum()
```

	0
price	0
area	0
bedrooms	0
bathrooms	0
stories	0
mainroad	0
guestroom	0
basement	0
hotwaterheating	0
airconditioning	0
parking	0
prefarea	0
furnishingstatus	0

```
df.duplicated().sum()
```

```
0
```

```
df
```

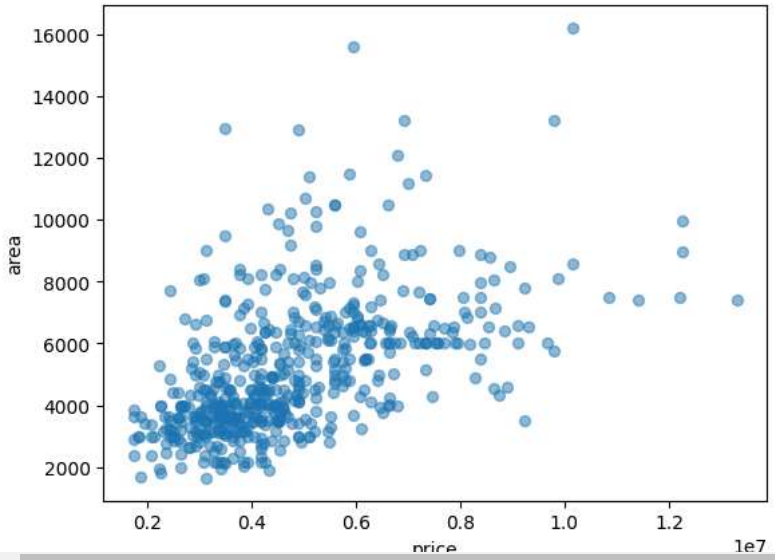
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	fu
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	
...	...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	

545 rows × 13 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

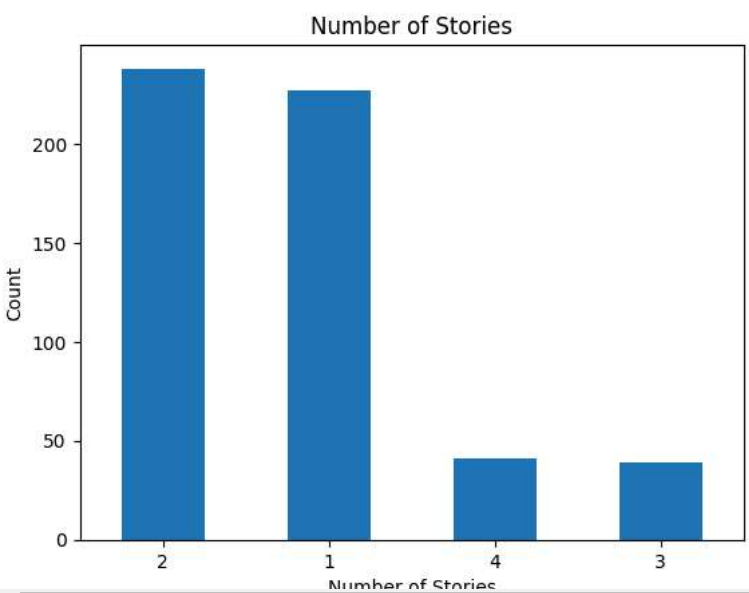
```
df.plot(kind = 'scatter',x = 'price',y = 'area', s = 32 , alpha = 0.5)
```

```
<Axes: xlabel='price', ylabel='area'>
```



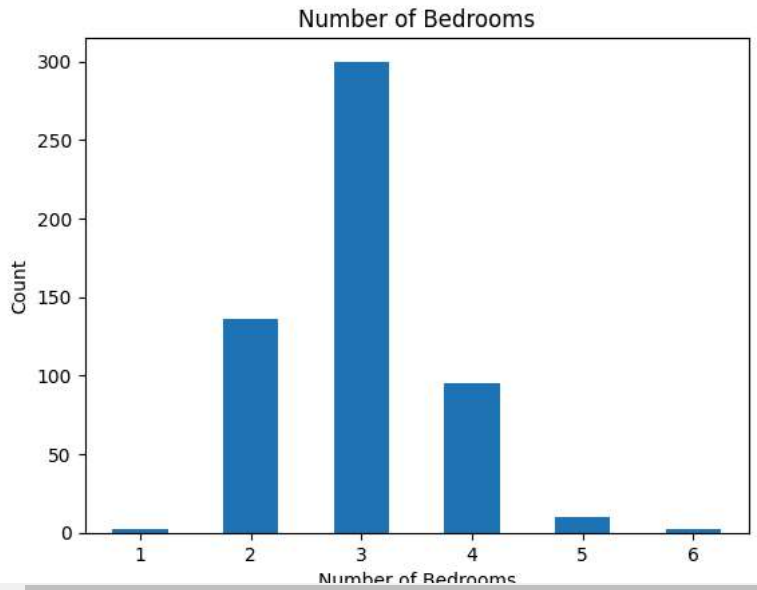

```
stories = df['stories'].value_counts()
```

```
stories.plot(kind = 'bar')
plt.title('Number of Stories')
plt.xlabel('Number of Stories')
plt.ylabel('Count')
plt.xticks(rotation = 0)
plt.show()
```




```
bedroom = df['bedrooms'].value_counts().sort_index()
```

```
bedroom.plot(kind = 'bar')
plt.title('Number of Bedrooms')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Count')
plt.xticks(rotation = 0)
plt.show()
```

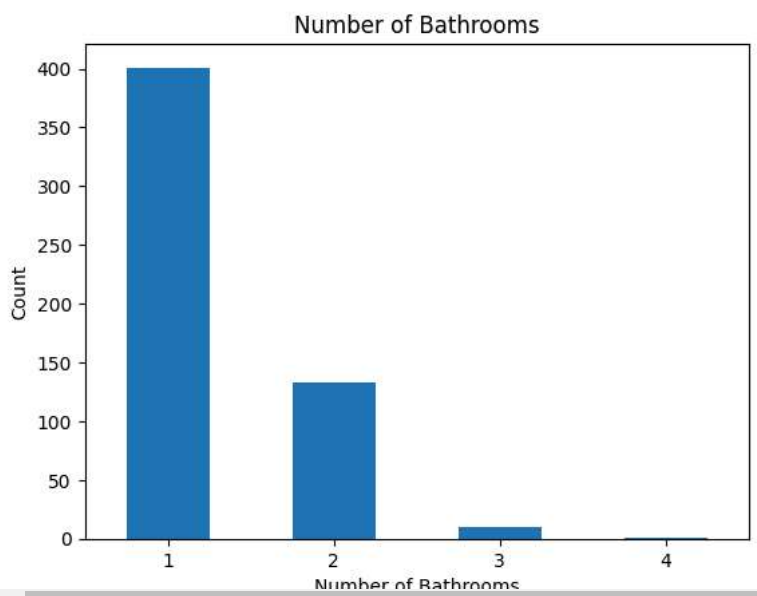



```
for i in range(10):
    print(df.iloc[i])
```

```
airconditioning             yes
parking                       1
prefarea                    yes
furnishingstatus    unfurnished
Name: 9, dtype: object
```

```
bathroom = df['bathrooms'].value_counts().sort_index()
```

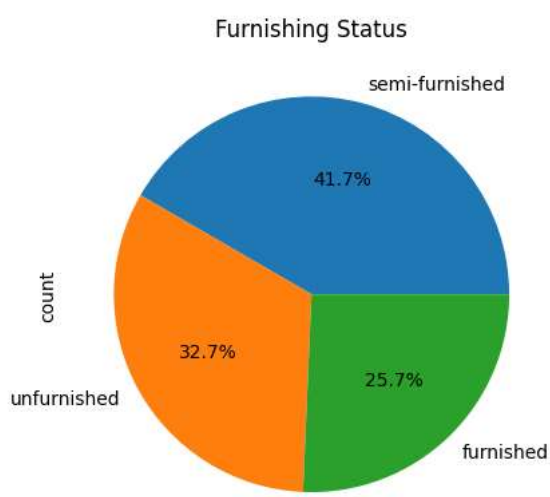
```
bathroom.plot(kind = 'bar')
plt.title('Number of Bathrooms')
plt.xlabel('Number of Bathrooms')
plt.ylabel('Count')
plt.xticks(rotation = 0)
plt.show()
```




```
furnishingstatus = df['furnishingstatus'].value_counts()
```

```
furnishingstatus.plot(kind = 'pie', autopct = "%1.1f%%")
plt.title('Furnishing Status')
```

```
plt.xticks(rotation = 0)
plt.show()
```




```
encoder = LabelEncoder()
```

```
encode = ['furnishingstatus','mainroad','guestroom','basement','hotwaterheating','airconditioning','prefarea']
for i in encode:
    df[i] = encoder.fit_transform(df[i])
```

```
df.columns
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

```
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	fu
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	1	0	1	0	0	2	0	
541	1767150	2400	3	1	1	0	0	0	0	0	0	0	
542	1750000	3620	2	1	1	1	0	0	0	0	0	0	
543	1750000	2910	3	1	1	0	0	0	0	0	0	0	
544	1750000	3850	3	1	2	1	0	0	0	0	0	0	

545 rows × 13 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
x = df.drop('price',axis = 1)
y = df['price']
```

```
scaler = StandardScaler()
x=scaler.fit_transform(x)
```

```
scaler = MinMaxScaler()
x=scaler.fit_transform(x)
y = scaler.fit_transform(y.values.reshape(-1,1))
```

```
x
```

```
array([[0.39656357, 0.6       , 0.33333333, ..., 0.66666667, 1.        ,
        0.        ],
       [0.5024055 , 0.6       , 1.        , ..., 1.        , 0.        ,
        0.        ],
       [0.57113402, 0.4       , 0.33333333, ..., 0.66666667, 1.        ,
        0.5       ],
       ...,
       [0.13539519, 0.2       , 0.        , ..., 0.        , 0.        ,
        1.        ],
       [0.08659794, 0.4       , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.15120275, 0.4       , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

```
y
```

```
       [0.08484848],
       [0.08181818],
       [0.07878788],
       [0.07878788],
       [0.07878788],
       [0.07878788],
       [0.07878788],
       [0.07878788],
       [0.07878788],
       [0.07818182],
       [0.07818182],
       [0.07393939],
       [0.07272727],
       [0.07272727],
       [0.07272727],
       [0.06666667],
       [0.06666667],
       [0.06666667],
       [0.06363636],
       [0.06363636],
       [0.06060606],
       [0.06060606],
       [0.06060606],
       [0.06060606],
       [0.06060606],
       [0.06060606],
       [0.0569697 ],
       [0.05454545],
       [0.05454545],
       [0.05454545],
       [0.05151515],
       [0.04848485],
       [0.04545455],
       [0.04545455],
       [0.04545455],
       [0.04242424],
       [0.04181818],
       [0.03333333],
       [0.03030303],
       [0.03030303],
       [0.03030303],
       [0.01818182],
       [0.01212121],
       [0.01212121],
       [0.00909091],
       [0.00606061],
       [0.00148485],
       [0.        ],
       [0.        ],
       [0.        ]])
```

+ Code + Text

Suggested code may be subject to a licence | BnbKe/MyChatGPT

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 42)
```

```
linear_model = LinearRegression()
```

```
linear_model.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
yPrediction = linear_model.predict(x_test)
```

```
Accuracy = r2_score(y_test,yPrediction)
```