```
# Step 1: Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
```

```
df = pd.read_csv("kc_house_data.csv")
df.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_abov
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180.
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170.
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770.
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050.
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680.

5 rows × 21 columns

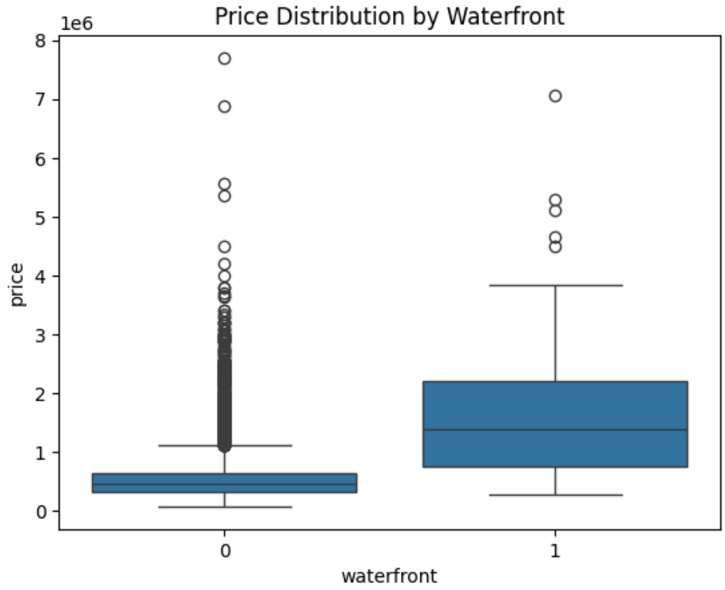
```
# Step 3: Display data types of each column
df.describe()
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	216
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	

```
floor_counts = df["floors"].value_counts().to_frame()
floor_counts.columns = ['count']
print(floor_counts)
```

```
        count
floors
1.0     10680
2.0      8241
1.5      1910
3.0       613
2.5       161
3.5         8
```

```
# Step 6: Boxplot to check price outliers by waterfront view
sns.boxplot(x="waterfront", y="price", data=df)
plt.title("Price Distribution by Waterfront")
plt.show()
```




```
# Step 7: regplot to check correlation between 'sqft_above' and 'price'
sns.regplot(x="sqft_above", y="price", data=df)
plt.title("Price vs Sqft Above")
plt.show()
```


```
# Step 8: Linear Regression with single feature 'sqft_living'
lr = LinearRegression()
lr.fit(df[['sqft_living']], df['price'])
print("R^2 (sqft_living):", lr.score(df[['sqft_living']], df['price']))
```

```
R^2 (sqft_living): 0.4928532179037931
```

```
# Step 9: Linear Regression with multiple features
from sklearn.linear_model import LinearRegression
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above",
            "grade", "sqft_living"]
X = df[features].dropna()
y = df.loc[X.index, 'price']
lr = LinearRegression()
lr.fit(X, y)
print("R^2 (multiple features):", lr.score(X, y))
```

```
R^2 (multiple features): 0.6577312410909923
```

```
# Step 10: Train/Test split for model validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
# Step 11: Pipeline - Scale, Polynomial Transform, Linear Regression
pipe = Pipeline([
    ('scale', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', LinearRegression())
])

pipe.fit(X_train, y_train)
print("R^2 (Pipeline Polynomial Regression):", pipe.score(X_test, y_test))
```

```
R^2 (Pipeline Polynomial Regression): 0.7530582530920498
```

```
# Step 12: Ridge Regression (Linear)
ridge = Ridge(alpha=0.1)
ridge.fit(X_train, y_train)
print("R^2 (Ridge Regression):", ridge.score(X_test, y_test))
```

```
R^2 (Ridge Regression): 0.673131854065206
```

```
# Step 13: Ridge Regression (Polynomial Features)
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

ridge_poly = Ridge(alpha=0.1)
ridge_poly.fit(X_train_poly, y_train)
print("R^2 (Polynomial Ridge):", ridge_poly.score(X_test_poly, y_test))
```

```
R^2 (Polynomial Ridge): 0.7442033165284614
```