EXPLORATORY DATA ANALYSIS USING PYTHON

PROJECT TITLE: HOUSING DATASET ANALYSIS

NAME: POOJA K

BATCH: DA & DS MAY -2025 BATCH

ONLINE / OFFLINE: OFFLINE

CONTENTS

1. INTRODUCTION:

This housing dataset provides detailed information on residential properties, including price, size, layout, and location. The goal of this project is to thoroughly analyze the data and extract meaningful insights to support informed decision-making in the real estate industry.

The dataset consists of 4,600 rows and 18 columns, covering various property features such as price, size, location, and condition.

- **Date**: Record the date of the property.
- **Price**: Property sale price.
- **Bedrooms/Bathrooms**: Number of bedrooms and bathrooms.
- **Sqft_living**: Total living space area.
- **Sqft_lot**: Total lot size.
- **Floors**: Number of floors.
- **Waterfront**: 1 if waterfront, else 0.
- **View**: View quality (0–4).
- **Condition**: Overall condition (1–5).
- **Sqft_above**: Area above ground level.
- **Sqft_basement**: Basement area.
- **Yr_built**: Year built.
- **Yr_renovated**: Last renovation year.
- **Street**: Property street address.
- **City**: Property city.
- **Statezip**: State and ZIP code.
- **Country**: Property country.

2. AIM:

This analysis aims to explore residential housing data to uncover key factors influencing property prices. By examining variables like location, size, condition, and other factors, the project identifies patterns and trends that affect real estate value. It also involves data cleaning, feature engineering, and visualization to enhance insights. The final goal is to support informed decision-making for investors, developers, and buyers through data-driven analysis and geospatial mapping.

3. PROBLEM STATEMENT :

In this project, our goal is to determine the appropriate price for residential properties using the housing dataset. By analyzing key factors such as location, renovations, property size, and layout, we will examine how these attributes influence pricing. The dataset provides detailed information, including price, number of bedrooms and bathrooms, square footage, and location specifics. The goal is to conduct a thorough analysis to gain meaningful insights that can help stakeholders in the real estate industry make informed decisions.

4. PROJECT WORFLOW :

This project follows a systematic workflow to analyze residential housing data, aiming to extract meaningful statistics and actionable insights to support real estate decision-making.

Data Collection

- Load the housing dataset (CSV or database).
- Understand the structure and types of features.

Data Preprocessing

- Handle missing values and duplicates.
- Convert data types (e.g., date columns).
- Encode categorical variables if needed (e.g., city, waterfront).

Exploratory Data Analysis (EDA)

- Summary statistics of key features.

- Visualizations (histograms, scatter plots, box plots).
- Correlation matrix to identify feature relationships.

Feature Engineering

- Create new features (e.g., age of house, total area).
- Normalize or scale features if required.

Data Visualization

- Use charts to display patterns and trends.
- Geospatial maps for location-based price analysis.

Statistics

- Calculate descriptive statistics (mean price, average bedrooms, etc.).
- Analyze distributions of key variables.

Insights

- Identify correlations between features and price.
- Analyze geospatial trends and property conditions.
- Detect outliers and their impact on price distribution.
- Summarize market trends and key takeaways.

4.1 DATA COLLECTION:

The first step involves gathering the housing dataset, which contains detailed information on various property attributes necessary for analysis. This step focuses on acquiring the raw data that forms the foundation of the project. It ensures that all relevant property details—such as price, size, location, and condition—are collected and ready for further processing and analysis.

In this project, I have stored my dataset in the format of a CSV file in the D/ drive. In my project, I have imported libraries such as Pandas, Numpy, Matplotlib, and Seaborn.

1. Libraries

```python
#import required packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Python

After importing the necessary libraries, declaring a variable name as df, and making the CSV file to read as pd.read_csv, and taking a copy of the original data, which is already declared using df, and storing it as df_copydata

Load csv file

```python
df=pd.read_csv(r"C:/Users/Home/Downloads/housing.csv")
df
```

Python

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city
0	02/05/2014 0:00	3.130000e+05	3	1.50	1340.0	NaN	1.5	0	0	3	1340	0	1955.0	2005	18810 Densmore Ave N	Shoreline
1	02/05/2014 0:00	2.384000e+06	5	2.50	3650.0	NaN	2.0	0	4	5	3370	280	1921.0	0	709 W Blaine St	Seattle
2	02/05/2014 0:00	3.420000e+05	3	2.00	1930.0	NaN	1.0	0	0	4	1930	0	1966.0	0	26206-26214 143rd Ave SE	Kent
3	02/05/2014 0:00	4.200000e+05	3	2.25	2000.0	NaN	1.0	0	0	4	1000	1000	1963.0	0	857 170th Pl NE	Bellevue
4	02/05/2014 0:00	5.500000e+05	4	2.50	1940.0	NaN	1.0	0	0	4	1140	800	1976.0	1992	9105 170th Ave NE	Redmond
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	09/07/2014 0:00	3.081667e+05	3	1.75	1510.0	6360.0	1.0	0	0	4	1510	0	NaN	1979	501 N 143rd St	Seattle
4596	09/07/2014 0:00	5.343333e+05	3	2.50	1460.0	7573.0	2.0	0	0	3	1460	0	NaN	2009	14855 SE 10th Pl	Bellevue
4597	09/07/2014 0:00	4.169042e+05	3	2.50	3010.0	7014.0	2.0	0	0	3	3010	0	NaN	0	759 Ilwaco Pl NE	Renton

```python
#declare a variable to copy dataset from the original dataset

df_copydata = df.copy()
df_copydata
```

[64] Python

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	02/05/2014 0:00	3.130000e+05	3	1.50	1340.0	NaN	1.5	0	0	3	1340	0	1955.0	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	02/05/2014 0:00	2.384000e+06	5	2.50	3650.0	NaN	2.0	0	4	5	3370	280	1921.0	0	709 W Blaine St	Seattle	WA 98119	USA
2	02/05/2014 0:00	3.420000e+05	3	2.00	1930.0	NaN	1.0	0	0	4	1930	0	1966.0	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	02/05/2014 0:00	4.200000e+05	3	2.25	2000.0	NaN	1.0	0	0	4	1000	1000	1963.0	0	857 170th Pl NE	Bellevue	WA 98008	USA
4	02/05/2014 0:00	5.500000e+05	4	2.50	1940.0	NaN	1.0	0	0	4	1140	800	1976.0	1992	9105 170th Ave NE	Redmond	WA 98052	USA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	09/07/2014 0:00	3.081667e+05	3	1.75	1510.0	6360.0	1.0	0	0	4	1510	0	NaN	1979	501 N 143rd St	Seattle	WA 98133	USA
4596	09/07/2014 0:00	5.343333e+05	3	2.50	1460.0	7573.0	2.0	0	0	3	1460	0	NaN	2009	14855 SE 10th Pl	Bellevue	WA 98007	USA
4597	09/07/2014 0:00	4.169042e+05	3	2.50	3010.0	7014.0	2.0	0	0	3	3010	0	NaN	0	759 Ilwaco Pl NE	Renton	WA 98059	USA
4598	10/07/2014 0:00	2.034000e+05	4	2.00	2090.0	6630.0	1.0	0	0	3	1070	1020	NaN	0	5148 S Creston St	Seattle	WA 98178	USA
4599	10/07/2014 0:00	2.206000e+05	3	2.50	1490.0	8102.0	2.0	0	0	4	1490	0	NaN	0	18717 SE 258th St	Covington	WA 98042	USA

4600 rows × 18 columns

After this, usually we get the first five rows and the last five rows, shape, length, sample, a range of rows from the dataset, information of the data, describe the mathematical values in the dataset, and its datatype.

4.2 DATA CLEANING

Data cleaning is essential to prepare the dataset for analysis by addressing missing values, removing duplicates, and correcting inconsistencies. This step ensures the quality and accuracy of the data by fixing errors and handling incomplete information. Clean data helps produce reliable analysis results and prevents misleading insights.

First, we need to check the data type of the dataset by using df_copydata.info(), which checks the datatypes, non-null count, and columns.

```
# information about dataset

df_copydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   int64  
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4560 non-null   float64
 5   sqft_lot       4586 non-null   float64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4577 non-null   float64
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4543 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(6), int64(7), object(5)
memory usage: 647.0+ KB
```

Next, we used to identify the null values present in the columns using df_copydata.isnull().sum()

```
# missing values in the dataset

df_copydata.isnull().sum()
```

```
date              0
price             0
bedrooms          0
bathrooms         0
sqft_living      40
sqft_lot         14
floors            0
waterfront        0
view              0
condition         0
sqft_above        0
sqft_basement     0
yr_built         23
yr_renovated      0
street            0
city             57
statezip          0
country           0
dtype: int64
```

Using fillna, it fills the missing values in the dataset. Since the dataset has missing values in both **numerical** and **categorical** columns, the typical strategy is:

- **Numerical columns** → fill with **median**
- **Categorical columns** → fill with **mode**
- sqft_living, sqft_lot, and yr_built are numerical — best filled with the **median** to avoid influence from extreme values.
- City is a categorical variable — best filled with the **mode** to preserve consistency.

```
#check whether the missing values are fill are not

df_copydata.isnull().sum()
```

[77]

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

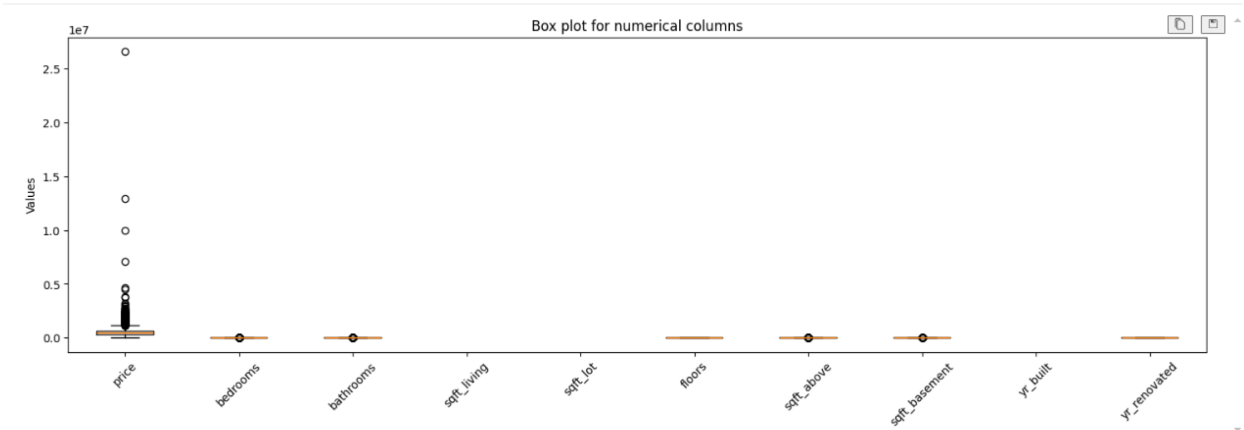
Outliers in the dataset :

Outliers are data points that differ significantly from other observations in the dataset. In the housing data, outliers can appear in features like price or square footage, often due to rare luxury properties or data entry errors. Identifying and handling outliers is important, as they can skew statistical results and impact model accuracy. Common methods to detect outliers include boxplots and the Interquartile Range (IQR) technique. Depending on the context, outliers can be removed, capped, or transformed to improve data quality.

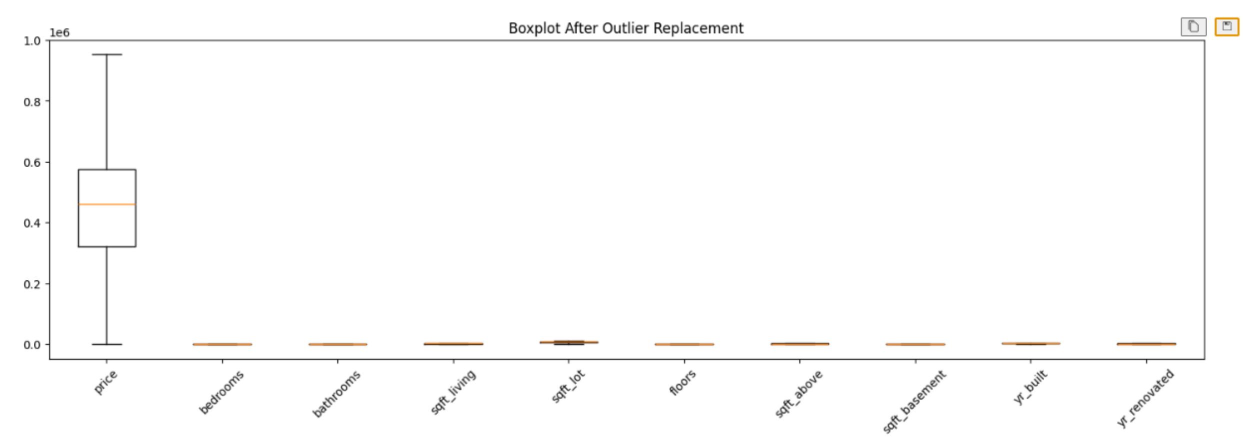
It calculates the median value for each specified column and its upper and lower bounds.

```python
# Select only numeric columns
# Select specific columns from df and create a copy
numeric_col = df[['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
                  'floors', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated']].copy()


# Loop over numeric columns and replace outliers
for i in range(10):  # You can reduce to 1 iteration; 10 is unnecessary unless the distribution changes a lot
    for col in numeric_col.columns:
        Q1 = df_copydata[col].quantile(0.25)
        Q3 = df_copydata[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        median_value = df_copydata[col].median()

        print(f"\nColumn: {col}")
        print(f"Lower bound: {lower_bound}")
        print(f"Upper bound: {upper_bound}")
        print(f"Median value: {median_value}")

        # Replace outliers with the rounded median
        df_copydata.loc[(df_copydata[col] < lower_bound) | (df_copydata[col] > upper_bound), col] = round(median_value, 0)
```

Before outliers



After removing the outliers



4.3 EDA (Exploratory Data Analysis)

EDA is the process of analyzing datasets to summarize their main characteristics using visual and statistical methods. It helps in understanding the data, identifying patterns, and preparing for modeling.

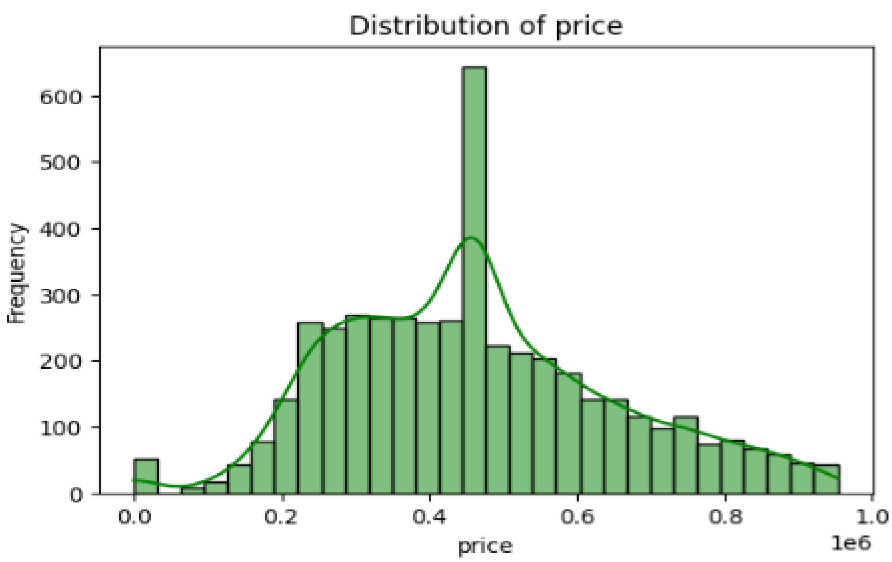
Types of EDA

1. **Univariate Analysis**
   - Examines a single variable.
   - Example: Histogram of `price`, boxplot of `sqft_living`.
2. **Bivariate Analysis**
   - Examines relationships between two variables.
   - Example: Scatter plot of `price` vs `sqft_living`, boxplot of `price` by `condition`.
3. **Multivariate Analysis**
   - Analyzes more than two variables together.
   - Example: Pair plots, grouped bar charts.

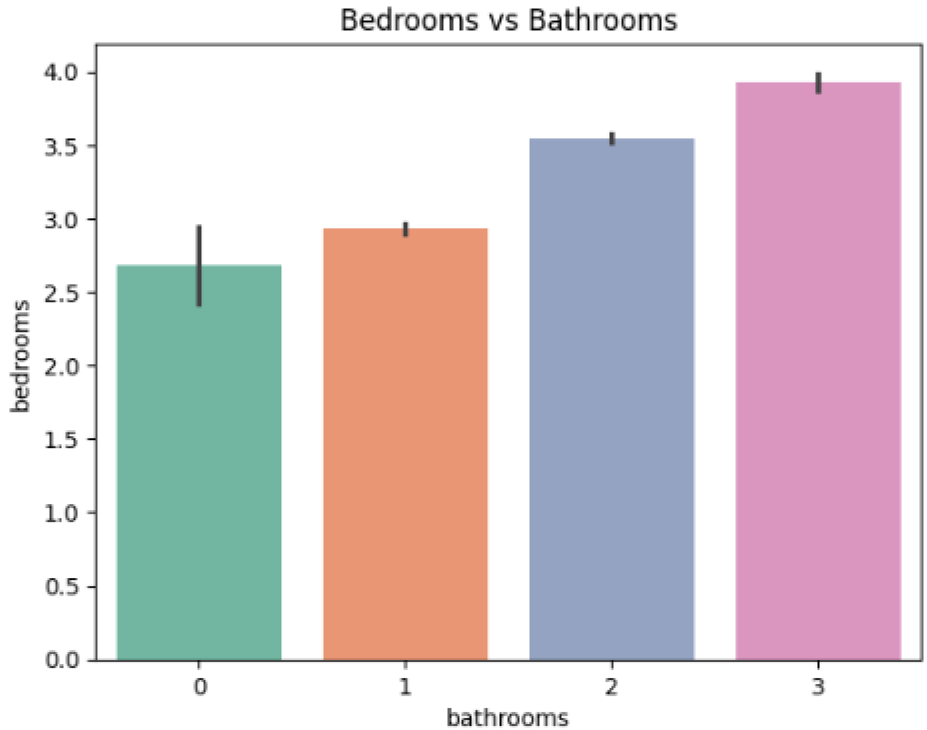
Continous variable:
price,sqft_living,sqft_lot,floors,sqft_above,sqft_basement,yr_built,yr_renovated

Categorical variable:
Bedrooms, Bathrooms, Floors, Waterfront, Views, Condition, City, State, Zip, Country, Year, Month, Day

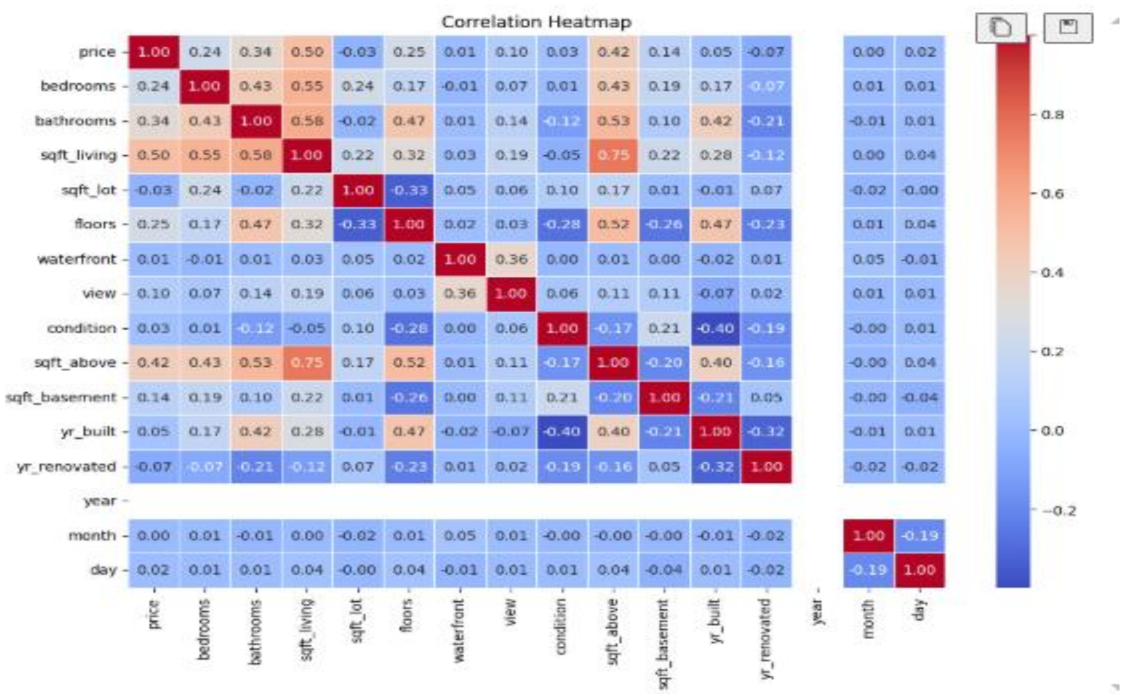
Univariate Analysis:



Bivariate analysis:



Multivariate Analysis



4.4 Feature Engineering

Feature engineering involves creating new features to improve model performance and gain deeper insights. In this project, new variables were derived to enhance the prediction and understanding of housing prices, such as:

- **house_age**: Number of years since the house was built.
- **renovation_age**: Years since the last renovation; 0 if never renovated.
- **price_per_sqft**: Price efficiency — how much each square foot costs.
- **multi_floor**: Binary indicator (1 or 0) if the house has more than one floor.
- **basement_ratio**: Proportion of living space in the basement

```python
#house age

df_copydata['house_age'] = 2025 - df_copydata['yr_built']
```

```python
#renovation age

df_copydata['renovation_age'] = df_copydata['yr_renovated'].apply(lambda x: 2025 - x if x > 0 else 0)
```

```python
#price_per_sqft

df_copydata['price_per_sqft'] = df_copydata['price'] / df_copydata['sqft_living']
```

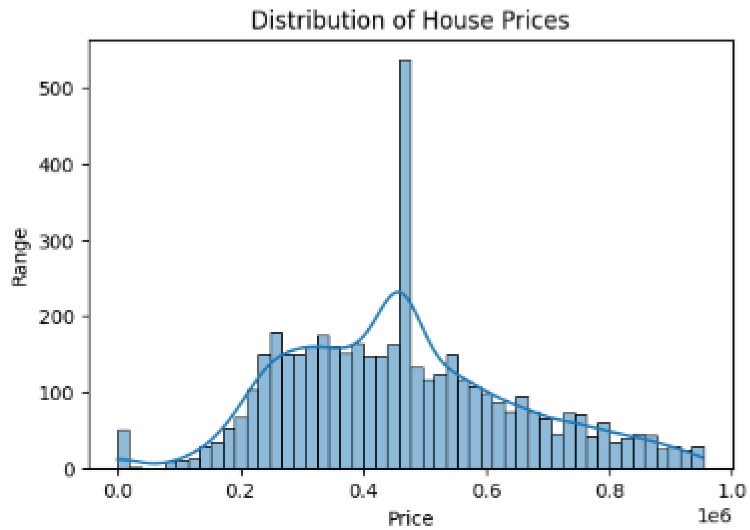
```python
#Is multi floor

df_copydata['multi_floor'] = (df_copydata['floors'] > 1).astype(int)
```

```python
#Basement Ratio

df_copydata['basement_ratio'] = df_copydata['sqft_basement'] / df_copydata['sqft_living']
```

4.5 VISUALIZATION

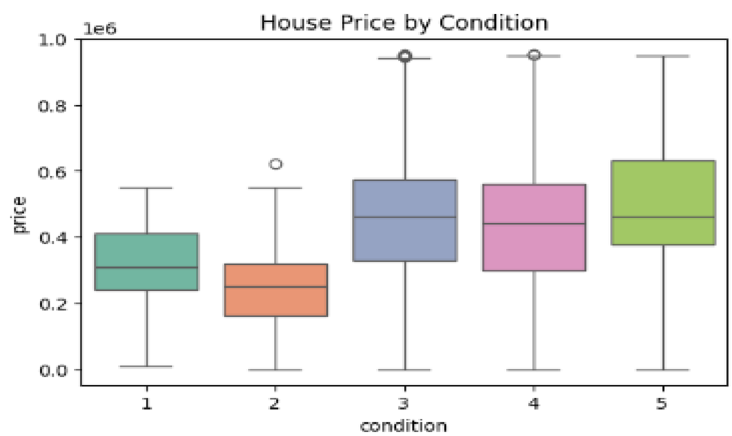
To better understand the relationships and trends within the housing dataset, several visualizations were created. These plots helped reveal hidden patterns, outliers, and correlations among various features. It shows the distribution of house prices using a histogram.

```
#histogram (price)


plt.figure(figsize=(6, 4))
sns.histplot(df_copydata['price'], bins=50, kde=True)
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Range')
plt.show()
```



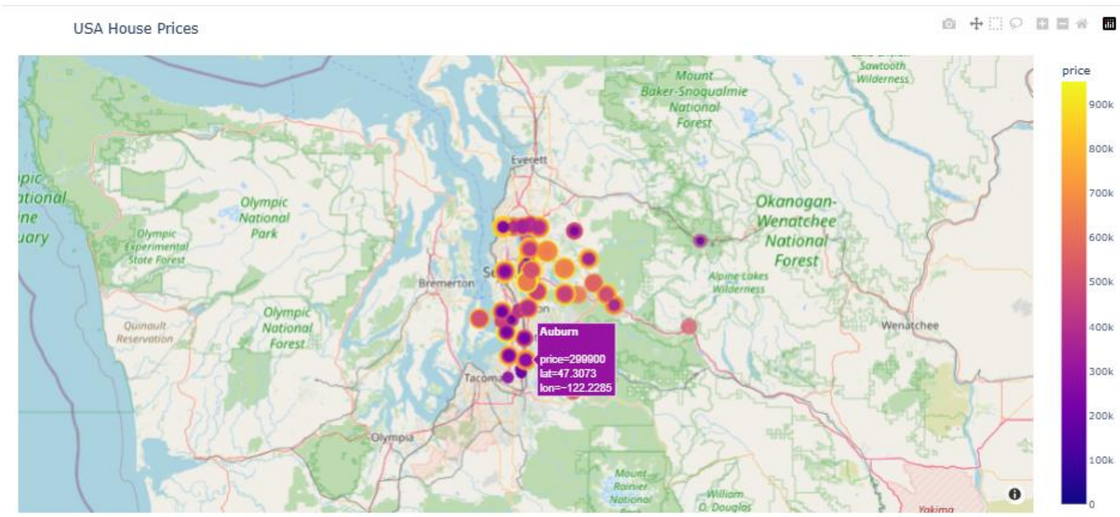
Boxplot for price by condition

```
# Boxplot for Price by Condition

import warnings
warnings.filterwarnings('ignore')
plt.figure(figsize=(6, 4))
sns.boxplot(x='condition', y='price', data=df_copydata, palette='Set2')
plt.title("House Price by Condition")
plt.show()
```



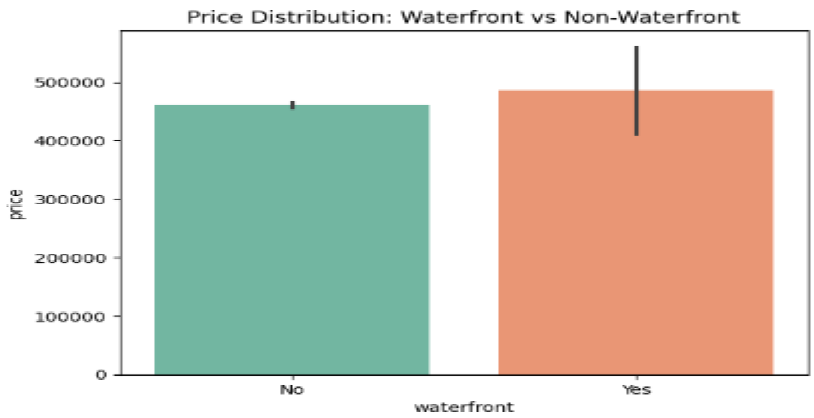
A scatter plot is used to visualize the land area vs price.

Map plot is used to plot the exact country(USA) based on its city and state, zip code, the price of the house in the dataset, and the latitude and longitude of the cities in the dataset.

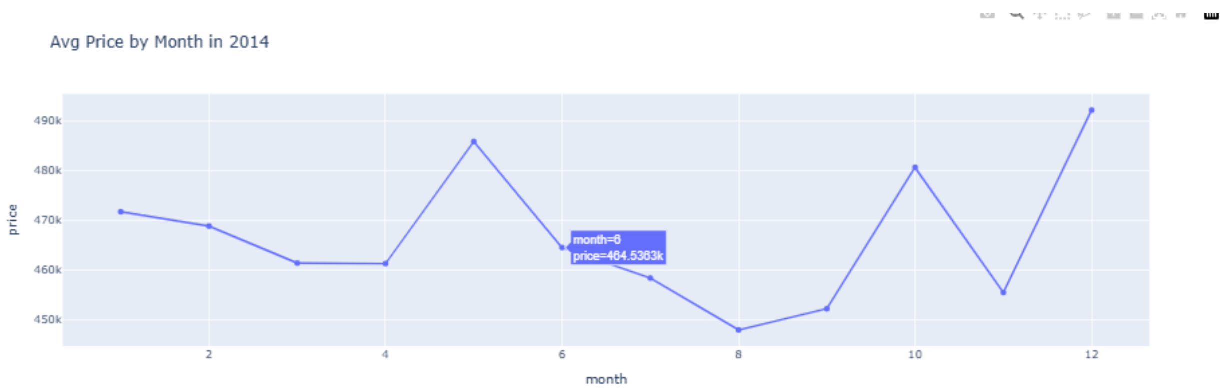


A bar plot is used for the waterfront based on the price in the dataset.

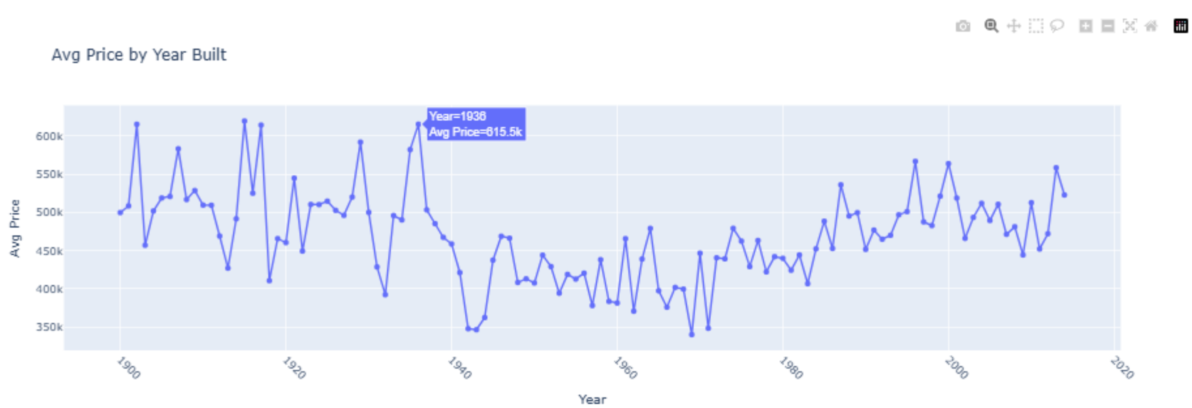
```
sns.barplot(x='waterfront', y='price', data=df_copydata, palette='Set2')
plt.title("Price Distribution: Waterfront vs Non-Waterfront")
plt.xticks([0, 1], ['No', 'Yes'])
plt.show()
```



Average price in the months



Average Price by the year built in the dataset



4.6 OBTAINED DERIVED METRICS

The **age of the property** was calculated by subtracting the `year_built` from the current year to show how old each property is, allowing better comparison than using raw build years. The **renovation age** was derived by subtracting `yr_renovated` from the current year when applicable, indicating how recently the property was updated. Additionally, **price per square foot** was computed by dividing the price by the living area (`price / sqft_living`), providing a standardized measure for comparing properties of different sizes.

4.7 FILTERING DATA FOR ANALYSIS

Data filtering involves selecting relevant subsets of the dataset based on specific criteria to improve the quality and focus of the analysis. This process removes irrelevant, duplicate, or erroneous records and narrows down the data to the most meaningful entries. Filtering can be done by applying conditions on columns such as date ranges, location, price limits, or property features, ensuring that the analysis reflects accurate and targeted insights.

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	...	month	day	weekday_name	lat	lon	house_age	renovation_age	price_per_sqft	multi_floor	basement_ratio
0	313000.0	3	1	1340.0	7683.5	1.5	0	None	3	1340	...	2.0	5.0	Wednesday	47.7557	-122.3415	70	20	233.582090	1	0.000000
1	461000.0	5	2	3650.0	7683.5	2.0	0	Excellent	5	3370	...	2.0	5.0	Wednesday	47.6062	-122.3321	104	0	126.301370	1	0.076712
2	342000.0	3	2	1930.0	7683.5	1.0	0	None	4	1930	...	2.0	5.0	Wednesday	47.3809	-122.2348	59	0	177.202073	0	0.000000
3	420000.0	3	2	2000.0	7683.5	1.0	0	None	4	1000	...	2.0	5.0	Wednesday	47.6101	-122.2015	62	0	210.000000	0	0.500000
4	550000.0	4	2	1940.0	7683.5	1.0	0	None	4	1140	...	2.0	5.0	Wednesday	47.6739	-122.1215	49	33	283.505155	0	0.412371

5 rows × 27 columns

4.8 STATISTICAL ANALYSIS

Statistical analysis involves applying mathematical techniques to summarize, describe, and interpret data patterns. It helps identify relationships, trends, and differences within the dataset. Common methods include calculating measures of central tendency (mean, median), dispersion (variance, standard deviation), and performing hypothesis tests to validate assumptions. This analysis supports data-driven decisions by providing quantitative evidence and insights.

Descriptive statistics are used to describe specified data in the dataset.

```
# Descriptive Statistics

df_copydata[['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot']].describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000
mean	461630.855241	3.363261	1.738696	2012.634348	6837.873261
std	187921.400563	0.794358	0.647794	728.949969	2556.285837
min	0.000000	2.000000	0.000000	370.000000	638.000000
25%	322500.000000	3.000000	1.000000	1470.000000	5001.000000
50%	461000.000000	3.000000	2.000000	1980.000000	7683.500000
75%	575000.000000	4.000000	2.000000	2480.000000	8100.000000
max	953007.000000	5.000000	3.000000	3990.000000	12731.000000

Two-way t-test:

A two-sample t-test was conducted to compare the mean house prices between different view categories. This test helps determine whether the differences in average prices are statistically significant based on the property's view rating. By evaluating the p-value, we assess if the variation in price is likely due to the view factor or occurred by chance, thus revealing whether view influences pricing trends.

Two way t-test view vs price

Hypothesis:

H_0 (Null): Waterfront has no effect on price

H_1 (Alt): Based on waterfront they have a different mean price

```python
from scipy.stats import ttest_ind

# Split data into two groups based on waterfront view
has_view = df_copydata[df_copydata['waterfront'] == 1]['price']
no_view = df_copydata[df_copydata['waterfront'] == 0]['price']

# Perform t-test
t_stat, p_val = ttest_ind(has_view, no_view, equal_var=False)

# Significance level
alpha = 0.05

# Print results
print(f"T-Statistic: {t_stat:.2f}")
print(f"P-Value: {p_val:.5f}")

# Interpret results
if p_val < alpha:
    print(f"Since p-value ({p_val:.5f}) < alpha ({alpha}), we reject the null hypothesis.")
    print("Conclusion: There is a significant difference in mean price between homes with and without a water view.")
else:
    print(f"Since p-value ({p_val:.5f}) >= alpha ({alpha}), we fail to reject the null hypothesis.")
    print("Conclusion: No significant difference in mean price between homes with and without a water view.")
```

```
T-Statistic: 0.70
P-Value: 0.48945
Since p-value (0.48945) >= alpha (0.05), we fail to reject the null hypothesis.
Conclusion: No significant difference in mean price between homes with and without a water view.
```

One-way ANOVA:

A one-way ANOVA test was performed to examine whether there are significant differences in average house prices across multiple view categories. This statistical method helps identify if at least one view group has a mean price significantly different from the others. A low p-value indicates that the view rating has a meaningful impact on property pricing.

One way anova

Hypothesis:

H_0: Mean price is the same across all views

H_1: At least one view has different mean price

```
from scipy.stats import f_oneway

# Group prices by view levels (0 to 4)
groups = [df_copydata[df_copydata['view'] == level]['price'] for level in sorted(df['view'].unique())]

# Perform one-way ANOVA
f_stat, p_val = f_oneway(*groups)

# Results
print(f"F-statistic: {f_stat:.2f}")
print(f"P-value: {p_val:.5f}")

# Interpretation
alpha = 0.05
if p_val < alpha:
    print("Conclusion: Significant difference in price across view levels.")
else:
    print("Conclusion: No significant difference in price across view levels.")
```

```
F-statistic: 18.21
P-value: 0.00000
Conclusion: Significant difference in price across view levels.
```

Chi-square:

A chi-square test of independence was conducted to examine the relationship between property view and condition. This test assesses whether these two categorical variables are statistically dependent or independent. A significant result (low p-value) indicates that the distribution of property condition varies meaningfully across different view categories, suggesting a potential association between the two.

Chi - square

The Chi-Square test checks if two categorical variables are independent (i.e., not related) or if there's a significant association between them.

```
import pandas as pd
from scipy.stats import chi2_contingency

# Create a contingency table
contingency = pd.crosstab(df_copydata['view'], df_copydata['condition'])

# Run the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency)

# Print results
print(f"Chi-square statistic: {chi2:.2f}")
print(f"Degrees of freedom: {dof}")
print(f"P-value: {p:.5f}")

# Interpret result
alpha = 0.05
if p < alpha:
    print("Conclusion: There is a significant relationship between view and condition.")
else:
    print("Conclusion: No significant relationship between view and condition.")
```

```
Chi-square statistic: 29.60
Degrees of freedom: 16
P-value: 0.02019
Conclusion: There is a significant relationship between view and condition.
```

5. OVERALL INSIGHT:

The housing market data reveals that **property price is significantly influenced by factors such as view quality, condition, square footage, and location**. Homes with better views and in good condition tend to be priced higher. Age and renovation history also impact value—newer or recently renovated homes command more. Statistical tests confirm **significant relationships** between view, price, and condition, indicating that aesthetic and physical quality drive market demand. The addition of derived metrics like **price per sqft** and **property age** enables deeper, more actionable insights for buyers, sellers, and investors.

6. CONCLUSION

This housing data analysis demonstrates that **key features such as view quality, condition, square footage, and renovation status have a substantial impact on property prices**. Statistical analyses, including t-tests, ANOVA, and chi-square tests, validate the **significant associations** between these variables. Derived metrics like **house age**, **renovation age**, and **price per square foot** provided enhanced clarity and depth to the insights.

Overall, this study helps understand how both **physical attributes and visual appeal** contribute to housing market trends, aiding **better decision-making** for stakeholders such as buyers, sellers, and real estate investors.