Project Documentation: Exploratory Data Analysis

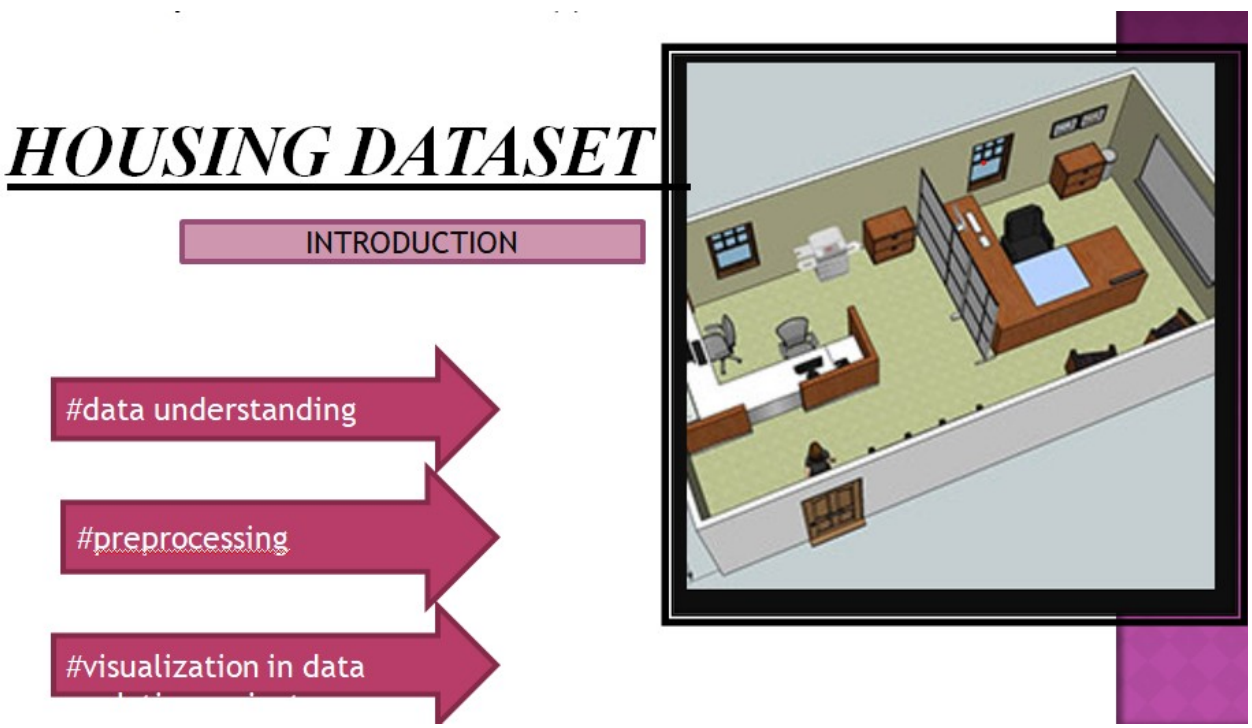
HOUSING DATASET

Vijayabharathi M

DA&DS (online)Batch.no -4

Introduction:

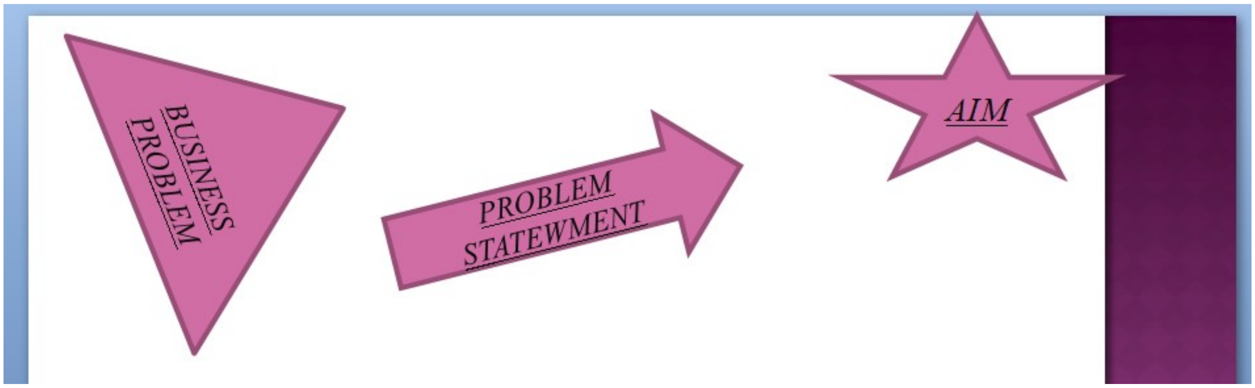
Housing prices have risen dramatically in the past year, driven by slow housing construction and increased demand driven by the pandemic. Buying or selling a home can be a monumental decision and being informed can save, or net, you money. This can be done by properly weighing and sorting the importance of various features of a home. Is it better for a home to be in a better neighborhood or have a larger lot? What are the effects of zoning? Does painting the walls yellow increase the sale price? This project aims to use data to accurately predict the price of novel homes and offer you a glimpse into the importance, or lack thereof, of the features of your home.



Aim:

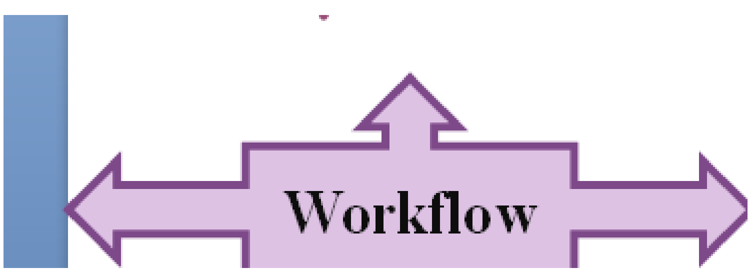
The Housing Dataset serves as a valuable resource for various data analysis and machine learning tasks. Here are some common aims associated with this dataset:The primary goal is to predict housing prices based on features such as square footage, number of bedrooms, location, etc. This task often involves regression models. Researchers and practitioners use this dataset to develop predictive models capable of estimating house prices accuratelyAnalysts explore relationships between different features (e.g., crime rate, accessibility to highways, etc.) and the target variable (housing price).By understanding the dataset, stakeholders (such as real estate agents, buyers, or investors) can make informed decisions related to housing investments.Insights gained from analyzing this data can guide pricing strategies, property investments, and market trends. The specific aim may vary depending on the context of the analysis or the project. I am working with this dataset, consider specific objectives approach accordingly..

Business Problem / Problem Statement:

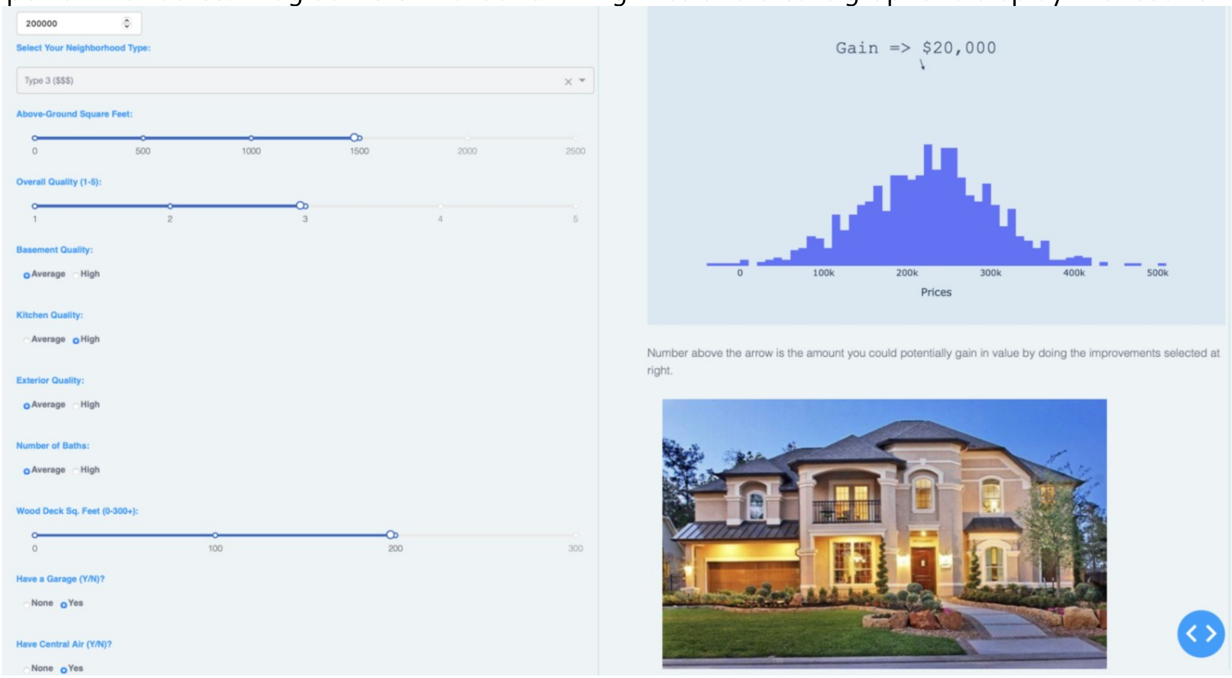


Its problem statement revolves around predicting housing prices based on various features. Here are the key points:_ _ 1)Dataset Description:The dataset contains information about housing prices in Boston.The target variable is the median value of owner-occupied homes._ _ 2)Business Problem:The goal is to build a regression model that accurately predicts housing prices. This prediction can help real estate agents, buyers, and sellers make informed decisions_ _ 3)Exploratory Data Analysis (EDA):EDA involves understanding the data distribution, identifying outliers, and exploring relationships between features and the target variable. Visualizations and statistical summaries are used to gain insights_ _ 4)Regression Analysis:Regression models (such as linear regression) are trained using the features to predict housing prices. The model’s performance is evaluated using metrics like mean squared error (MSE) or R-squared_ _ 5)Hypothesis Testing:Hypothesis testing may involve assessing whether certain features significantly impact housing prices. For example, testing whether the crime rate affects housing prices_ _ 6)Decision-Making: The insights gained from the analysis can guide decisions related to property investments, pricing, and development....o.

Project Workflow:



*Read the Housing dataset. Calculate summary statistics for important variables.*Create a histogram to show the distribution of the Sale Price variable.*Create a graph to visualize the relationship between Sale Price and Basement Square Footage.*Generate a correlation matrix to understand relationships between important variables.*Flag outliers in Ground Living Area and create graphs to display the results.



Requirements:

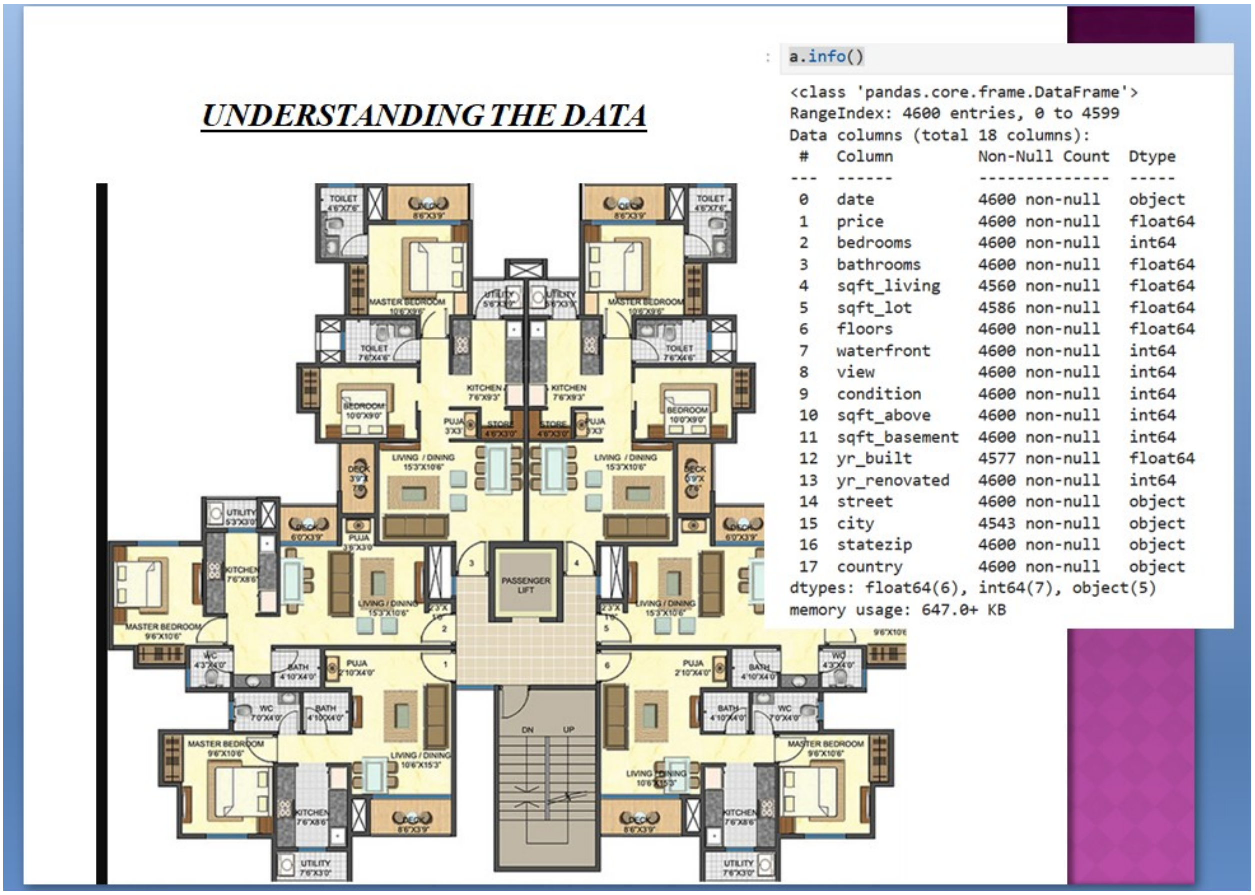
To run the code, you will need the following libraries:*numpy*pandas*matplotlib*seaborn

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

```
import pandas as pd
a=pd.read_csv("C:\kgisl class\miles stone 1\M housing (1) 22.07.24.csv")
a
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	
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	De
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	E
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	14
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	85

Data Understanding:



Data science would be much easier if all data was immediately ready for analysis but that keeps us all employed. Before training a model on the housing data we need to remove outliers, clean the columns, feature engineering, and standardize the information.

Exploratory Data Analysis (EDA):

```
a.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_b
count	4.600000e+03	4600.000000	4600.000000	4560.000000	4.586000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4577.000
mean	5.539483e+05	3.400870	2.160815	2138.935526	1.485981e+04	1.512065	0.007174	0.240652	3.451739	1840.825435	312.081522	1970.808
std	5.808371e+05	0.908848	0.783781	965.011449	3.592050e+04	0.538288	0.084404	0.778405	0.677230	970.705795	464.137228	29.724
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	350.000000	0.000000	1900.000
25%	3.225000e+05	3.000000	1.750000	1460.000000	5.000000e+03	1.000000	0.000000	0.000000	3.000000	1190.000000	0.000000	1951.000
50%	4.610000e+05	3.000000	2.250000	1980.000000	7.683500e+03	1.500000	0.000000	0.000000	3.000000	1590.000000	0.000000	1976.000
75%	6.550000e+05	4.000000	2.500000	2620.000000	1.101850e+04	2.000000	0.000000	0.000000	4.000000	2300.000000	610.000000	1997.000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	20450.000000	4820.000000	2014.000

```python
a.columns
```

```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

```python
b=a.iloc[:,1:]
b
```

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

```python
b.isnull()            # null values are turning true
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip
0	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False
4596	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False
4597	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False
4598	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False
4599	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False

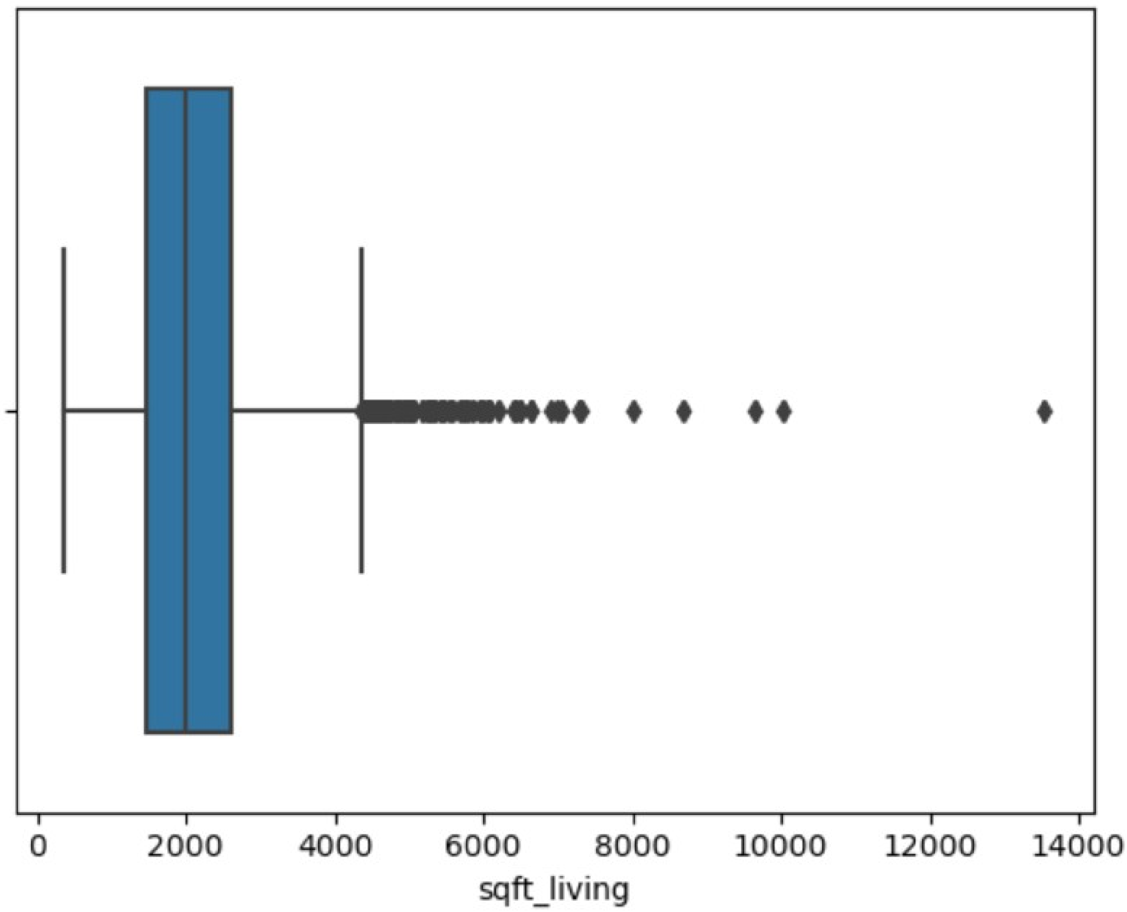
4600 rows × 17 columns

```python
b.isnull().sum()                                   #finding missing values and imputations
```

```
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
```

]:
```
sns.boxplot(x=b['sqft_living'])
```

]: <Axes: xlabel='sqft_living'>



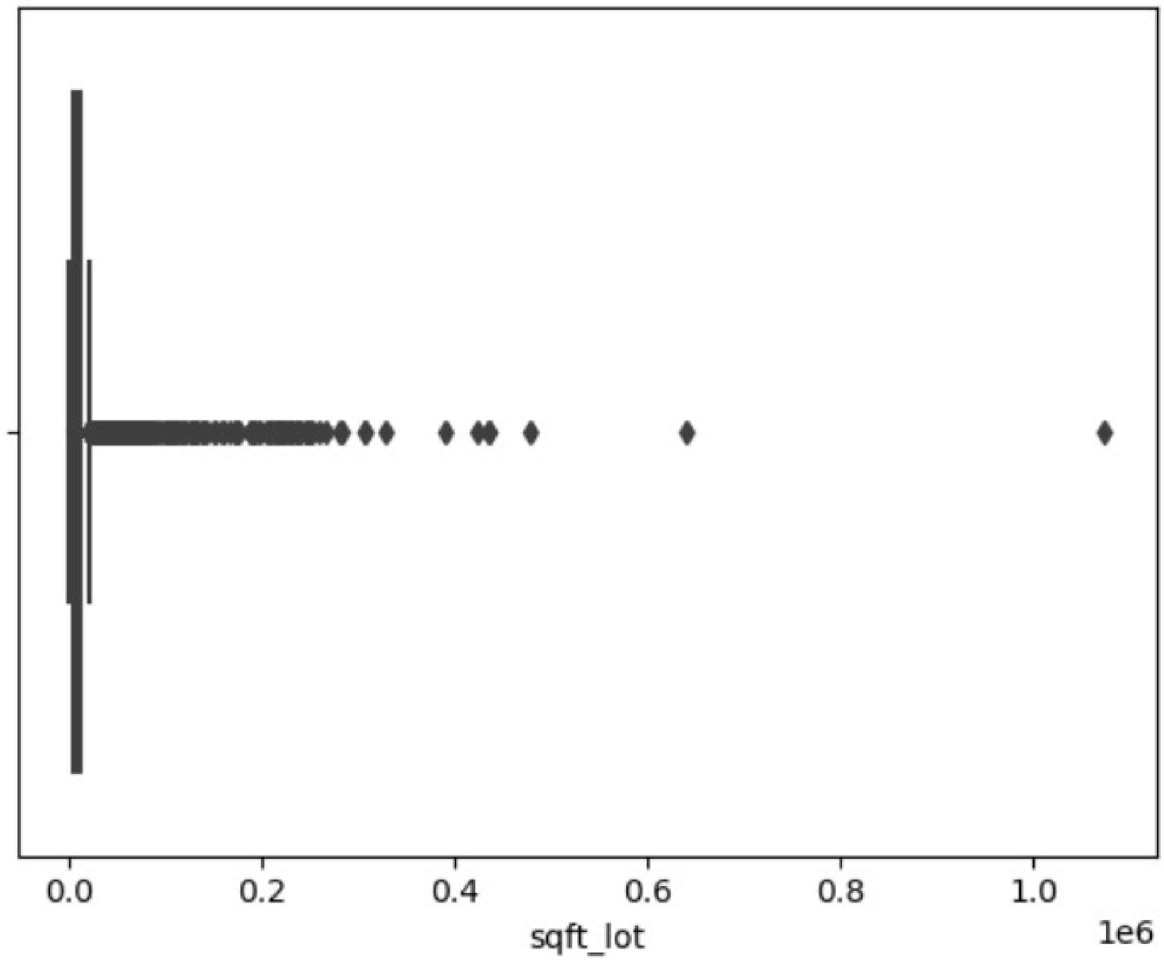
:
```
sqft_living=b['sqft_living'].median()
sqft_living
```

: 1980.0

:
```
b.sqft_living.fillna(sqft_living,inplace=True)
```

```
sns.boxplot(x=b['sqft_lot'])
```

<Axes: xlabel='sqft_lot'>



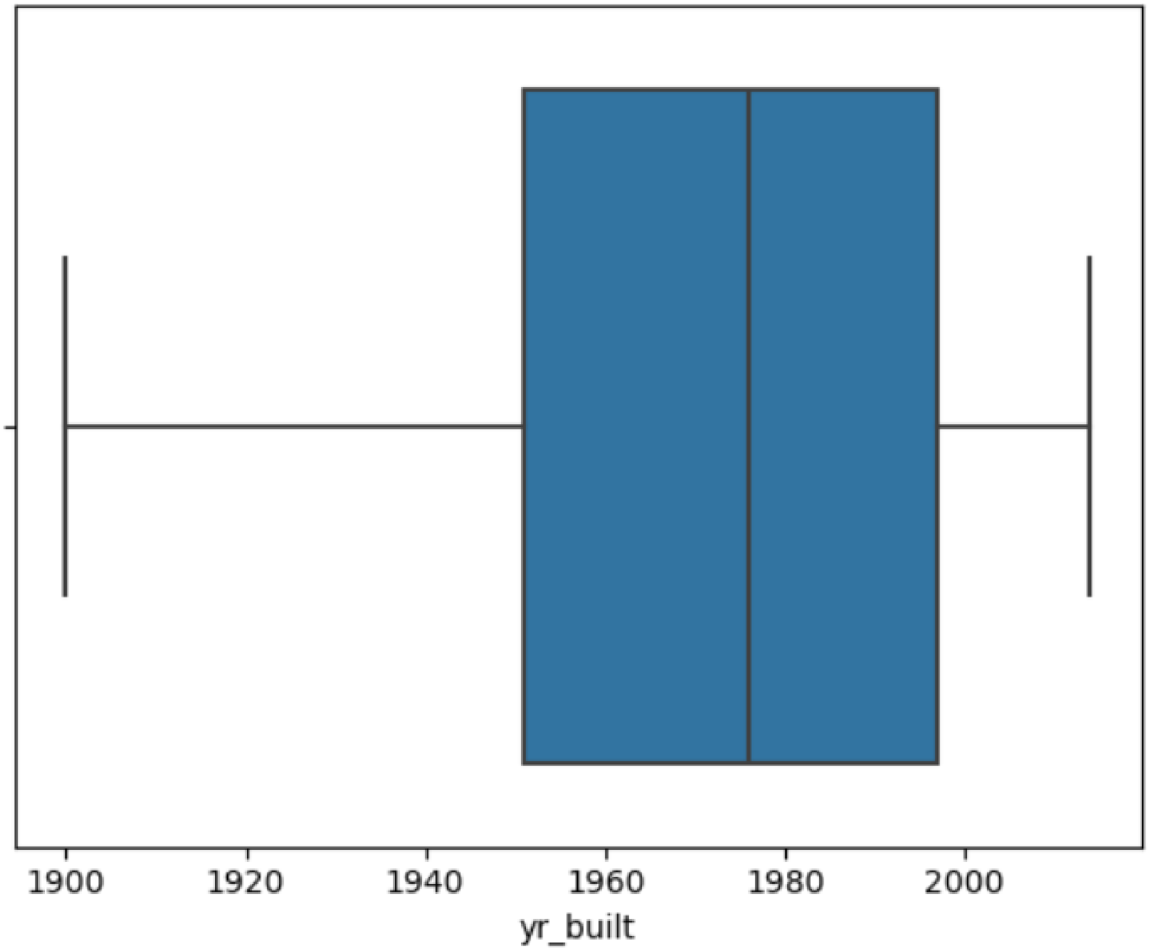

```
sqft_lot=b['sqft_lot'].median()
sqft_lot
```

7683.5

```
b.sqft_lot.fillna(sqft_lot,inplace=True)
```

```
sns.boxplot(x=b['yr_built'])
```

<Axes: xlabel='yr_built'>



Screenshot 2024-07-23 113756.png

```
city=b['city'].mode()[0]
city
```

'Seattle'

```
b.city.fillna(city,inplace=True)
```

```
b.isnull().sum()
```

```
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
```

[28]:
```
#find=iqr             #using quantile for removing and detectioning those outlayers
r=b.select_dtypes(exclude=['object'])
r
```

[28]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	3.130000e+05	3	1.50	1340.0	7683.5	1.5	0	0	3	1340	0	1955.000000	2005
1	2.384000e+06	5	2.50	3650.0	7683.5	2.0	0	4	5	3370	280	1921.000000	0
2	3.420000e+05	3	2.00	1930.0	7683.5	1.0	0	0	4	1930	0	1966.000000	0
3	4.200000e+05	3	2.25	2000.0	7683.5	1.0	0	0	4	1000	1000	1963.000000	0
4	5.500000e+05	4	2.50	1940.0	7683.5	1.0	0	0	4	1140	800	1976.000000	1992
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	3.081667e+05	3	1.75	1510.0	6360.0	1.0	0	0	4	1510	0	1970.808827	1979
4596	5.343333e+05	3	2.50	1460.0	7573.0	2.0	0	0	3	1460	0	1970.808827	2009
4597	4.169042e+05	3	2.50	3010.0	7014.0	2.0	0	0	3	3010	0	1970.808827	0
4598	2.034000e+05	4	2.00	2090.0	6630.0	1.0	0	0	3	1070	1020	1970.808827	0
4599	2.206000e+05	3	2.50	1490.0	8102.0	2.0	0	0	4	1490	0	1970.808827	0

```
:9]: q1=r.quantile(0.25)
     q1
```

```
:9]: price              322500.00
     bedrooms                3.00
     bathrooms               1.75
     sqft_living          1470.00
     sqft_lot             5001.00
     floors                  1.00
     waterfront              0.00
     view                    0.00
     condition               3.00
     sqft_above           1190.00
     sqft_basement           0.00
     yr_built             1951.00
     yr_renovated            0.00
     Name: 0.25, dtype: float64
```

```
:0]: q3=r.quantile(0.75)
     q3
```

```
:0]: price              655000.0
     bedrooms                4.0
     bathrooms               2.5
     sqft_living          2610.0
     sqft_lot            11000.0
     floors                  2.0
     waterfront              0.0
     view                    0.0
     condition               4.0
     sqft_above           2300.0
     sqft_basement         610.0
     yr_built             1997.0
     yr_renovated         1999.0
     Name: 0.75, dtype: float64
```

```
IQR=q3-q1
IQR
```

```
price            332500.00
bedrooms              1.00
bathrooms             0.75
sqft_living        1140.00
sqft_lot           5999.00
floors                1.00
waterfront            0.00
view                  0.00
condition             1.00
sqft_above         1110.00
sqft_basement       610.00
yr_built             46.00
yr_renovated       1999.00
dtype: float64
```

```
a=((r<q1-1.5*IQR)|(r>q1+1.5*IQR))
a
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	True	True	False	True	False	False	False	True	True	True	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	True	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	False	False	False	False	False	False	False	False	False	False	False	False	False
4596	False	False	False	False	False	False	False	False	False	False	False	False	False
4597	False	False	False	False	False	False	False	False	False	True	False	False	False
4598	False	False	False	False	False	False	False	False	False	False	True	False	False
4599	False	False	False	False	False	False	False	False	False	False	False	False	False

4600 rows × 13 columns

[33]:
```
filter=b[~((r<q1-1.5*IQR)|(r>q1+1.5*IQR)).any(axis=1)]
filter
```

[33]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street
0	313000.0000	3	1.50	1340.0	7683.5	1.5	0	0	3	1340	0	1955.000000	2005	18810 Densmore Ave N
2	342000.0000	3	2.00	1930.0	7683.5	1.0	0	0	4	1930	0	1966.000000	0	26206-26214 143rd Ave SE
4	550000.0000	4	2.50	1940.0	7683.5	1.0	0	0	4	1140	800	1976.000000	1992	9105 170th Ave NE
5	490000.0000	2	1.00	880.0	7683.5	1.0	0	0	3	880	0	1938.000000	1994	522 NE 88th St
6	335000.0000	2	2.00	1350.0	7683.5	1.0	0	0	3	1350	0	1976.000000	0	2616 174th Ave NE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4593	289373.3077	3	2.50	2538.0	4600.0	2.0	0	0	3	2538	0	1970.808827	1923	5703 Charlotte Ave SE

Univariate Analysis&Bivariate Analysis and Multivariate Analysis:

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

```
filter.groupby(['price']).count()
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	co
price																
0.0	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
20000.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
83000.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
83300.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
87500.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
808000.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
809000.0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
810000.0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
815000.0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
820000.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

981 rows × 16 columns

```
filter.groupby(['price']).size()
```

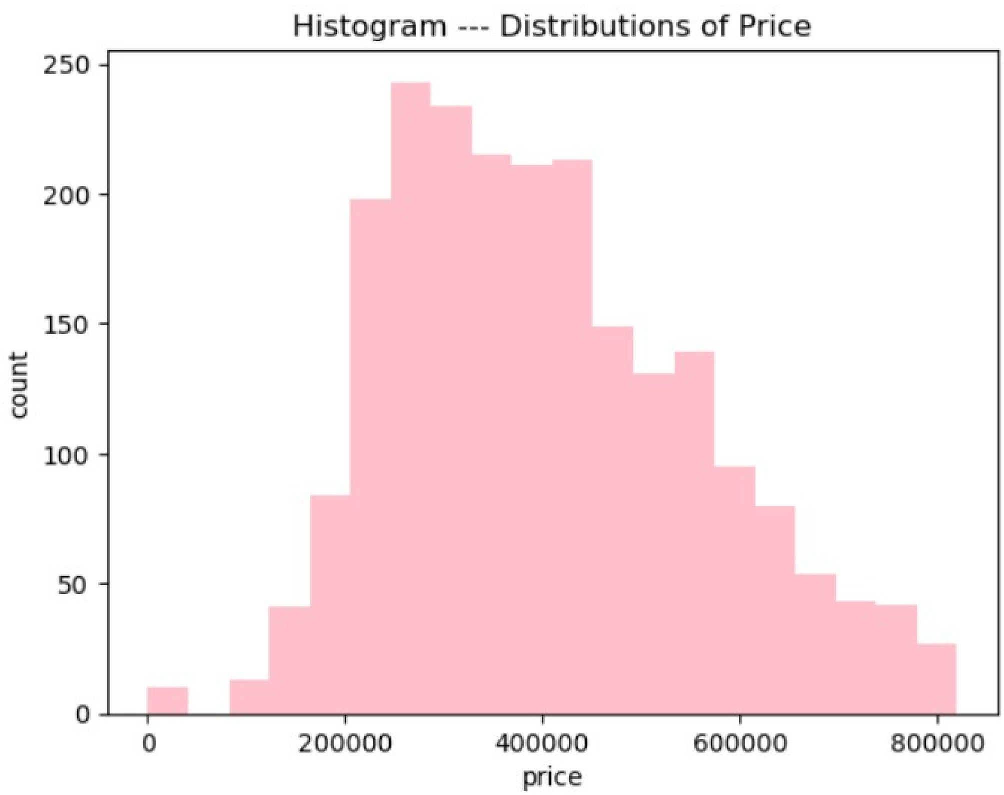
```
price
0.0          9
20000.0      1
83000.0      1
83300.0      1
87500.0      1
            ..
808000.0     1
809000.0     2
810000.0     3
815000.0     2
820000.0     1
Length: 981, dtype: int64
```

```
a=filter.groupby(['price']).size().reset_index(name='count')
a
```

	price	count
0	0.0	9
1	20000.0	1
2	83000.0	1
3	83300.0	1
4	87500.0	1
...	...	...
976	808000.0	1
977	809000.0	2
978	810000.0	3
979	815000.0	2
980	820000.0	1

981 rows × 2 columns

```python
x=filter['price']
plt.hist(x,bins=20,color="pink")
plt.title("Histogram --- Distributions of Price")
plt.xlabel("price")                                          #numerical univarient Histogram
plt.ylabel("count")
plt.show()
```




```python
filter.columns                          #square footage  includes (sqft_living,sqft_lot,sqft_above,sqft_basement)
```

```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
       'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement',
       'yr_built', 'yr_renovated', 'street', 'city', 'statezip', 'country'],
      dtype='object')
```

```python
sqft=filter[["sqft_living","sqft_lot","sqft_above","sqft_basement"]]
sqft
```

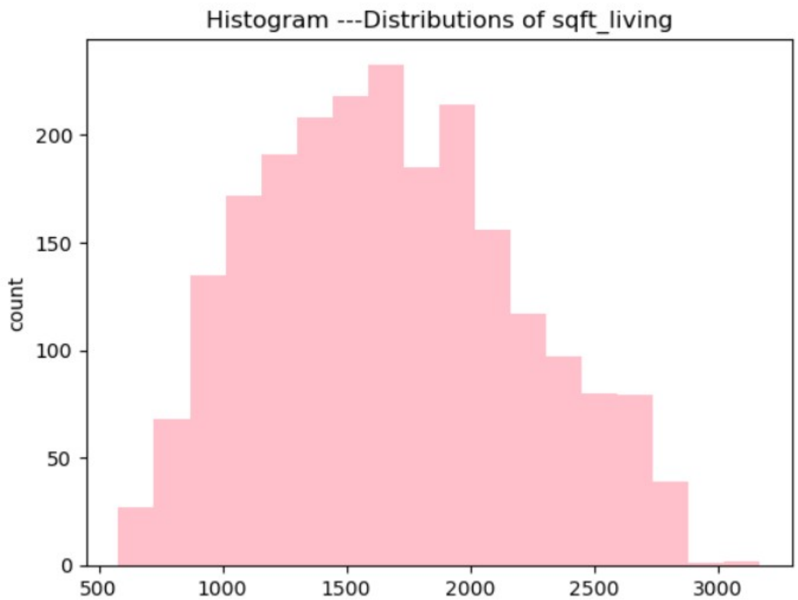
	sqft_living	sqft_lot	sqft_above	sqft_basement
0	1340.0	7683.5	1340	0
2	1930.0	7683.5	1930	0
4	1940.0	7683.5	1140	800
5	880.0	7683.5	880	0
6	1350.0	7683.5	1350	0
...	...	...	...	...
4593	2538.0	4600.0	2538	0
4594	1610.0	7223.0	1610	0
4595	1510.0	6360.0	1510	0
4596	1460.0	7573.0	1460	0
4599	1490.0	8102.0	1490	0

2222 rows × 4 columns

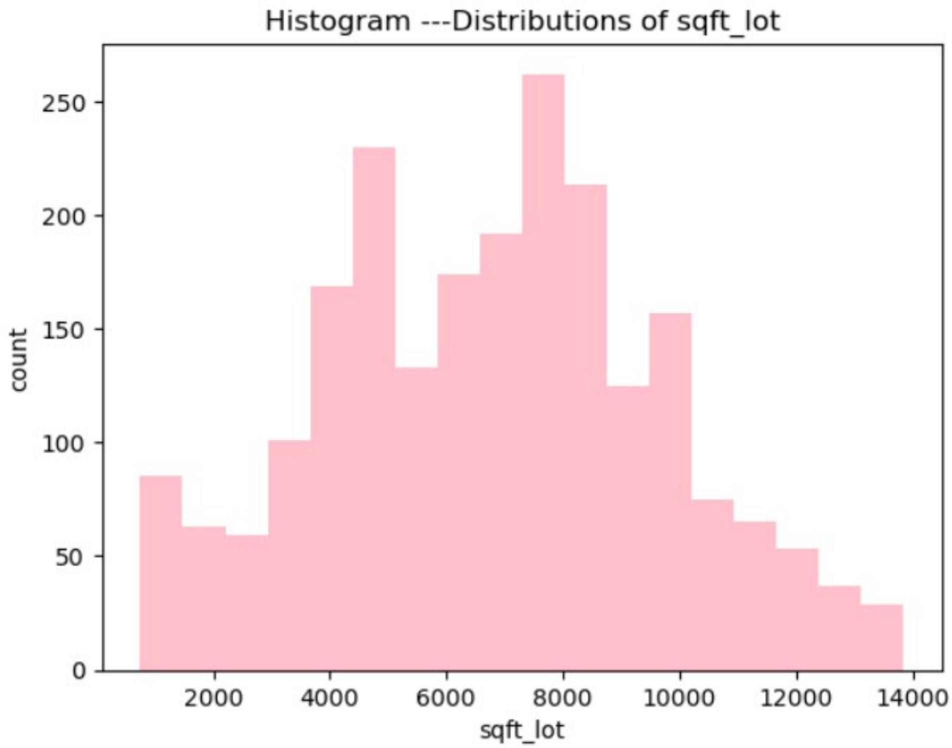
Screenshot 2024-07-23 181553.png

[43]:

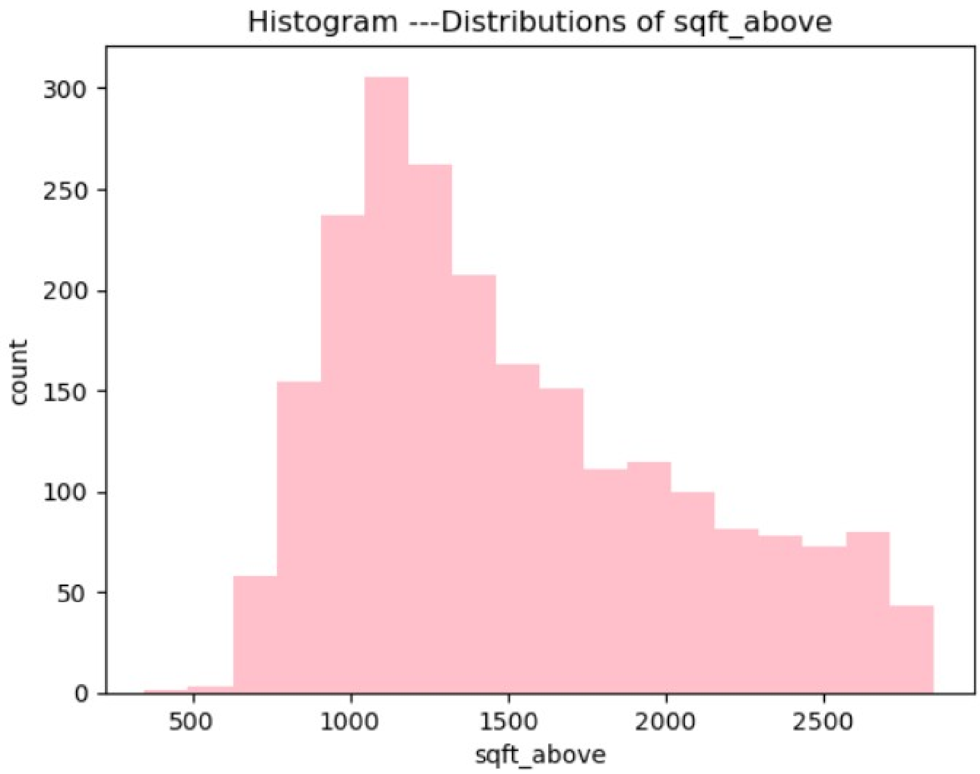
```
x=filter['sqft_living']
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---Distributions of sqft_living")
plt.xlabel("sqft_living")                                    #numerical univarient Histogram
plt.ylabel("count")
plt.show()
```



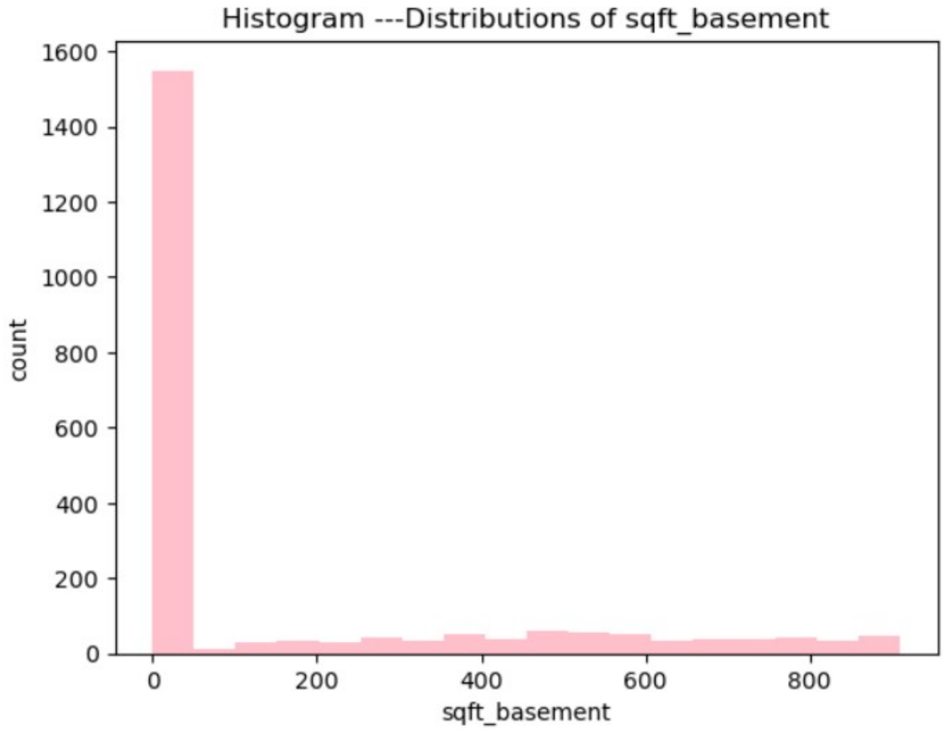
```python
x=filter['sqft_lot']
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---Distributions of sqft_lot")
plt.xlabel("sqft_lot")                                    #numerical univarient Histogram
plt.ylabel("count")
plt.show()
```



```python
x=filter['sqft_above']
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---Distributions of sqft_above")
plt.xlabel("sqft_above")                                    #numerical univarient Histogram
plt.ylabel("count")
plt.show()
```



```
x=filter['sqft_basement']
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---Distributions of sqft_basement")
plt.xlabel("sqft_basement")                                    #numerical univarient Histogram
plt.ylabel("count")
plt.show()
```



```python
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---distributions and summary statistics of Number of Bedrooms")
plt.xlabel("Bedrooms")                                                  #numerical univarient Histogram
plt.ylabel("Frequency")
plt.show()
```

```python
x=filter['bathrooms']
plt.hist(x,bins=18,color="pink")
plt.title("Histogram ---distributions and summary statistics of Number of Bathrooms")
plt.xlabel("bathrooms")                                                    #numerical univarient Histogram
plt.ylabel("Frequency")
plt.show()
```

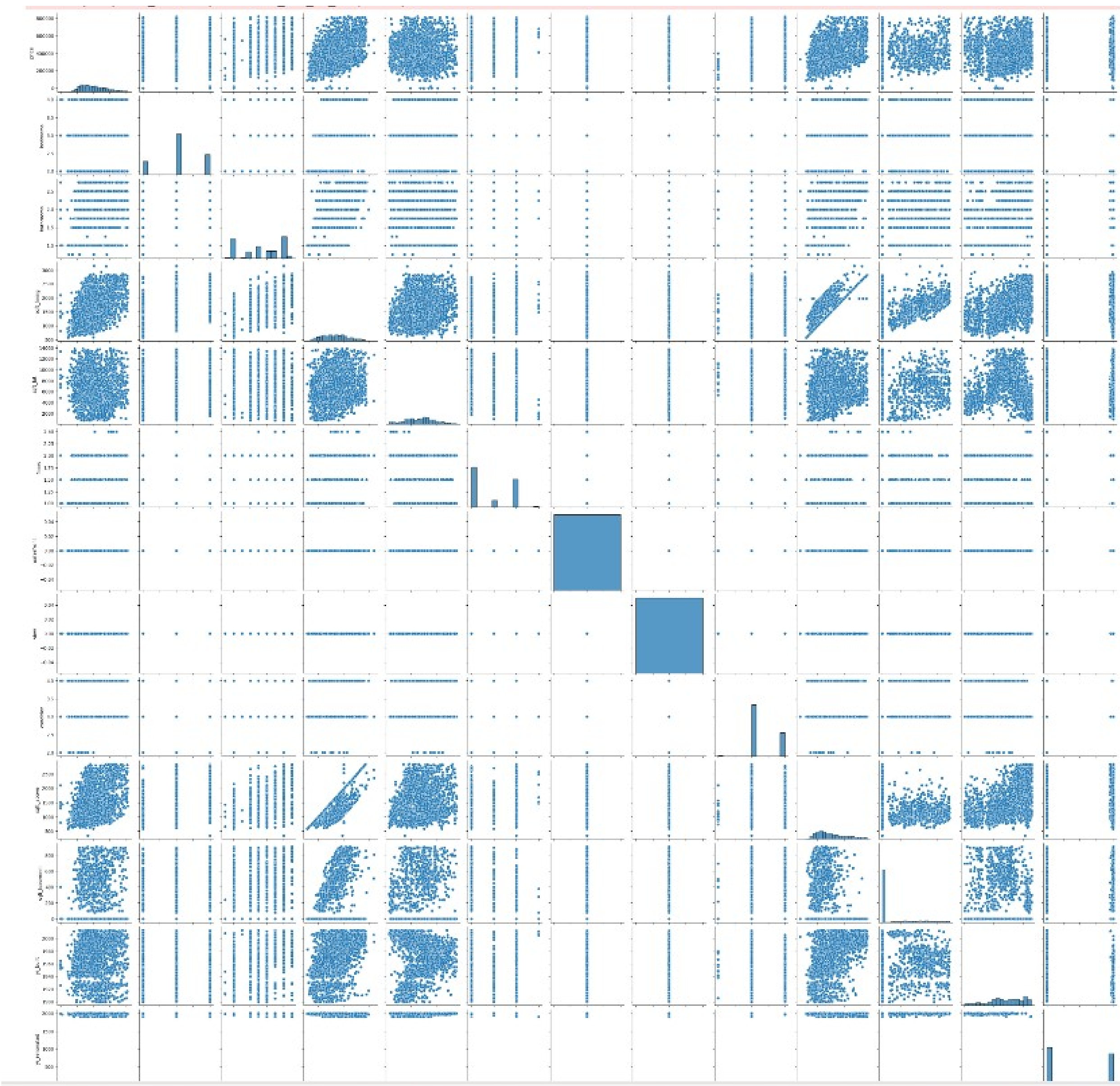

```python
filter[['price','sqft_living','sqft_lot','sqft_above','sqft_basement']]          # numeric+numeric=corelation(heatmap)
```

	price	sqft_living	sqft_lot	sqft_above	sqft_basement
0	313000.0000	1340.0	7683.5	1340	0
2	342000.0000	1930.0	7683.5	1930	0
4	550000.0000	1940.0	7683.5	1140	800
5	490000.0000	880.0	7683.5	880	0
6	335000.0000	1350.0	7683.5	1350	0
...	...	...	...	...	...
4593	289373.3077	2538.0	4600.0	2538	0
4594	210614.2857	1610.0	7223.0	1610	0
4595	308166.6667	1510.0	6360.0	1510	0
4596	534333.3333	1460.0	7573.0	1460	0
4599	220600.0000	1490.0	8102.0	1490	0

2222 rows × 5 columns

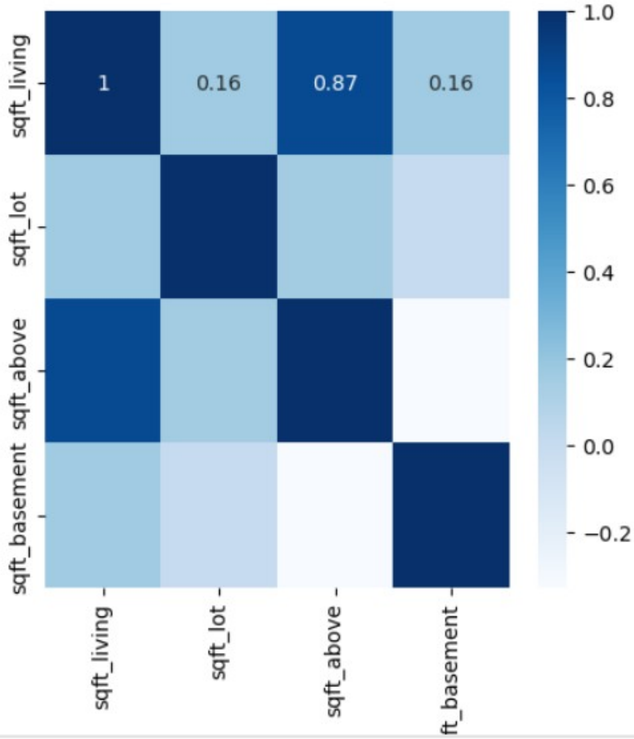
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
p=sns.pairplot(filter)
```



```
a=df1.corr()                                    # num+num=correlation
```

```
plt.figure(figsize=(5,5))
sns.heatmap(a,annot=True,cmap='Blues')
plt.show()
```




```
df1=filter.drop(columns=['price','waterfront','view','city','statezip','floors','condition','yr_built','yr_renovated','street','country','sqft_living','s
df1
```

	bedrooms	bathrooms
0	3	1.50
2	3	2.00
4	4	2.50
5	2	1.00
6	2	2.00
...	...	...
4593	3	2.50
4594	3	2.50
4595	3	1.75
4596	3	2.50
4599	3	2.50

2222 rows × 2 columns

Screenshot 2024-07-23 183557.png

```python
df1=filter.drop(columns=['price','waterfront','view','city','statezip','floors','condition','yr_built','yr_renovated','street','country',])
df1
```

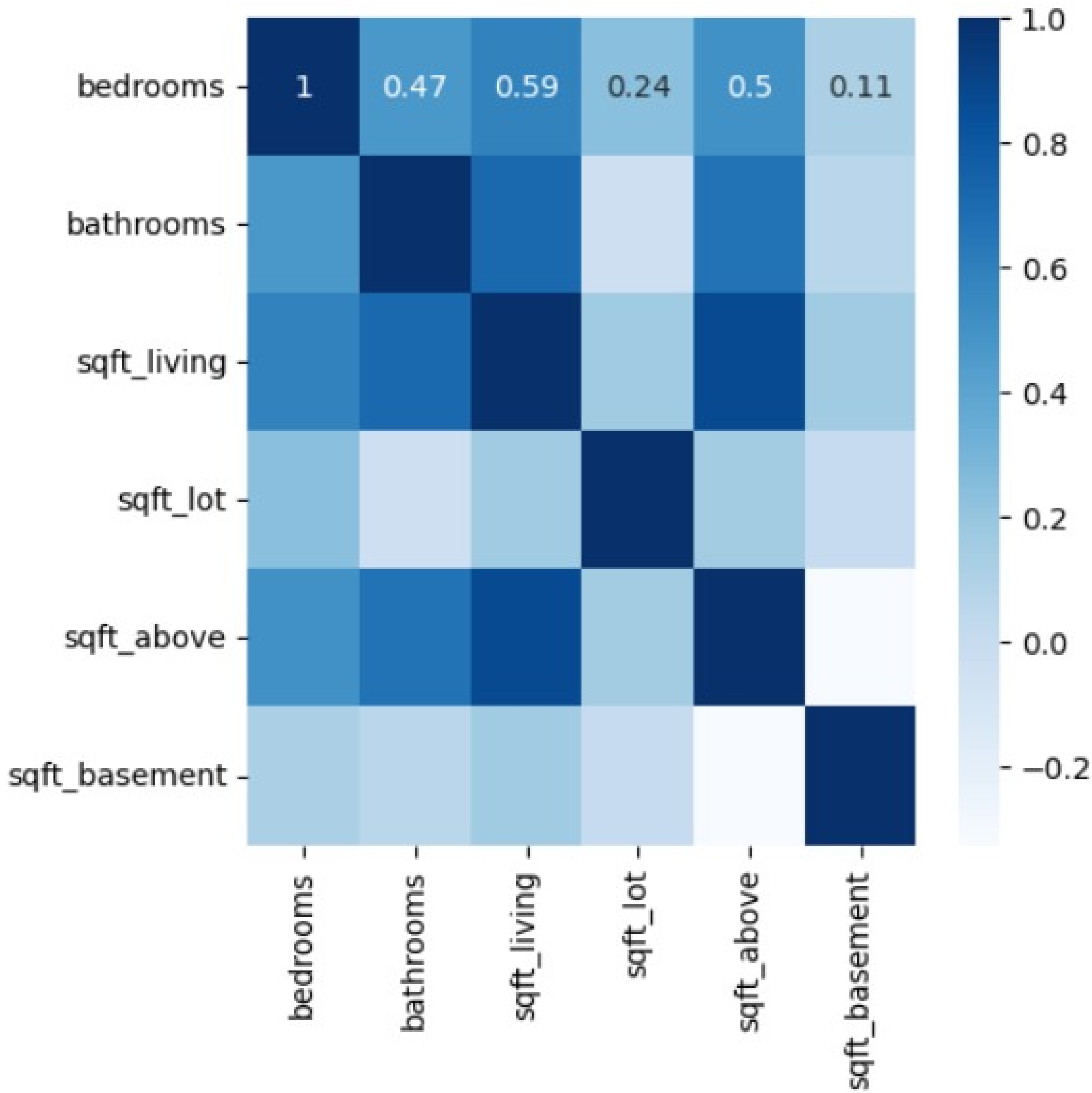
	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_above	sqft_basement
0	3	1.50	1340.0	7683.5	1340	0
2	3	2.00	1930.0	7683.5	1930	0
4	4	2.50	1940.0	7683.5	1140	800
5	2	1.00	880.0	7683.5	880	0
6	2	2.00	1350.0	7683.5	1350	0
...	...	...	...	...	...	...
4593	3	2.50	2538.0	4600.0	2538	0
4594	3	2.50	1610.0	7223.0	1610	0
4595	3	1.75	1510.0	6360.0	1510	0
4596	3	2.50	1460.0	7573.0	1460	0
4599	3	2.50	1490.0	8102.0	1490	0

2222 rows × 6 columns

```python
a=df1.corr()
a
```

	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_above	sqft_basement
bedrooms	1.000000	0.469843	0.585309	0.237811	0.504566	0.114133
bathrooms	0.469843	1.000000	0.715028	-0.040213	0.658553	0.058133
sqft_living	0.585309	0.715028	1.000000	0.164434	0.870316	0.164683
sqft_lot	0.237811	-0.040213	0.164434	1.000000	0.152184	0.008168
sqft_above	0.504566	0.658553	0.870316	0.152184	1.000000	-0.326412
sqft_basement	0.114133	0.058133	0.164683	0.008168	-0.326412	1.000000

```
plt.figure(figsize=(5,5))
sns.heatmap(a,annot=True,cmap='Blues')
plt.show()
```



```
filter                                                    #cath+cath=chi^2
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street
0	313000.0000	3	1.50	1340.0	7683.5	1.5	0	0	3	1340	0	1955.000000	2005	18810 Densmore Ave N
2	342000.0000	3	2.00	1930.0	7683.5	1.0	0	0	4	1930	0	1966.000000	0	26206-26214 143rd Ave SE
4	550000.0000	4	2.50	1940.0	7683.5	1.0	0	0	4	1140	800	1976.000000	1992	9105 170th Ave NE
5	490000.0000	2	1.00	880.0	7683.5	1.0	0	0	3	880	0	1938.000000	1994	522 NE 88th St
6	335000.0000	2	2.00	1350.0	7683.5	1.0	0	0	3	1350	0	1976.000000	0	2616 174th Ave NE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4593	289373.3077	3	2.50	2538.0	4600.0	2.0	0	0	3	2538	0	1970.808827	1923	5703 Charlotte Ave SE

```
filter['bedrooms'].unique()
```

```
array([3, 4, 2], dtype=int64)
```

```
filter['city'].unique()
```

```
array(['Shoreline', 'Kent', 'Redmond', 'Seattle', 'Maple Valley',
       'Snoqualmie', 'Sammamish', 'Bellevue', 'Issaquah', 'Kirkland',
       'Des Moines', 'Auburn', 'Federal Way', 'Duvall', 'Covington',
       'Carnation', 'Kenmore', 'Renton', 'North Bend', 'Burien',
       'Woodinville', 'Algona', 'Skykomish', 'Tukwila',
       'Lake Forest Park', 'Black Diamond', 'Normandy Park', 'SeaTac',
       'Mercer Island', 'Bothell', 'Vashon', 'Newcastle',
       'Snoqualmie Pass', 'Pacific', 'Ravensdale', 'Enumclaw',
       'Beaux Arts Village', 'Fall City', 'Milton'], dtype=object)
```

```
filter['condition'].unique()
```

```
array([3, 4, 2], dtype=int64)
```

```
filter['statezip'].unique()
```

```
array(['WA 98133', 'WA 98042', 'WA 98052', 'WA 98115', 'WA 98038',
       'WA 98155', 'WA 98074', 'WA 98106', 'WA 98007', 'WA 98045',
       'WA 98006', 'WA 98102', 'WA 98011', 'WA 98125', 'WA 98003',
       'WA 98117', 'WA 98034', 'WA 98072', 'WA 98107', 'WA 98055',
       'WA 98116', 'WA 98077', 'WA 98059', 'WA 98065', 'WA 98053',
       'WA 98122', 'WA 98005', 'WA 98029', 'WA 98027', 'WA 98033',
       'WA 98198', 'WA 98112', 'WA 98199', 'WA 98004', 'WA 98092',
       'WA 98008', 'WA 98136', 'WA 98023', 'WA 98103', 'WA 98019',
       'WA 98119', 'WA 98144', 'WA 98146', 'WA 98014', 'WA 98177',
       'WA 98028', 'WA 98057', 'WA 98058', 'WA 98105', 'WA 98118',
       'WA 98001', 'WA 98166', 'WA 98056', 'WA 98030', 'WA 98126',
       'WA 98168', 'WA 98075', 'WA 98288', 'WA 98031', 'WA 98178',
       'WA 98108', 'WA 98148', 'WA 98032', 'WA 98010', 'WA 98188',
       'WA 98040', 'WA 98002', 'WA 98070', 'WA 98109', 'WA 98068',
       'WA 98047', 'WA 98051', 'WA 98022', 'WA 98024', 'WA 98354'],
      dtype=object)
```

```
filter['bathrooms'].unique()
```

```
array([1.5 , 2.  , 2.5 , 1.  , 1.75, 2.25, 2.75, 1.25, 0.75])
```

```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
filter["bedrooms"]=label.fit_transform(filter['bedrooms'])
filter['city']=label.fit_transform(filter['city'])
filter['statezip']=label.fit_transform(filter['statezip'])
filter['bathrooms']=label.fit_transform(filter['bathrooms'])
filter['condition']=label.fit_transform(filter['condition'])
filter
```

```
newdf=filter[['bedrooms','bathrooms','condition','city','statezip']]
newdf
```

	bedrooms	bathrooms	condition	city	statezip
0	1	3	1	32	60
2	1	5	2	16	25
4	2	7	2	27	29
5	0	1	1	31	52
6	0	5	1	27	29
...	...	...	...	...	...
4593	1	7	1	1	43
4594	1	7	1	16	18
4595	1	4	2	31	60
4596	1	7	1	3	6
4599	1	7	2	8	25

2222 rows × 5 columns

```
x=newdf[['bathrooms','condition','city','statezip']]
x
```

	bathrooms	condition	city	statezip
0	3	1	32	60
2	5	2	16	25
4	7	2	27	29
5	1	1	31	52
6	5	1	27	29
...	...	...	...	...
4593	7	1	1	43
4594	7	1	16	18
4595	4	2	31	60
4596	7	1	3	6
4599	7	2	8	25

2222 rows × 4 columns

1]:
```
y=newdf[['bedrooms']]
y
```

1]:

	bedrooms
0	1
2	1
4	2
5	0
6	0
...	...
4593	1
4594	1
4595	1
4596	1
4599	1

2222 rows × 1 columns

```python
from sklearn.feature_selection import chi2
values=chi2(x,y)
values
```

```
(array([6.57482725e+02, 3.93905631e-01, 5.57566480e+02, 1.85219820e+03]),
 array([1.69605620e-143, 8.21229382e-001, 8.43290713e-122, 0.00000000e+000]))
```

```python
newdf.groupby(['bathrooms','bedrooms']).count()
```

		condition	city	statezip
bathrooms	**bedrooms**			
0	**0**	4	4	4
	2	1	1	1
1	**0**	253	253	253
	1	262	262	262
	2	47	47	47
2	**0**	2	2	2
3	**0**	42	42	42
	1	119	119	119
	2	25	25	25
4	**0**	40	40	40
	1	227	227	227

```python
newdf.groupby(['condition','bedrooms']).count()
```

condition	bedrooms	bathrooms	city	statezip
0	0	6	6	6
	1	4	4	4
	2	4	4	4
1	0	273	273	273
	1	818	818	818
	2	429	429	429
2	0	129	129	129
	1	385	385	385
	2	174	174	174

```
newdf.groupby(['city','bedrooms']).count()
```

		bathrooms	condition	statezip
city	**bedrooms**			
0	**1**	1	1	1
	2	1	1	1
1	**0**	6	6	6
	1	58	58	58
	2	37	37	37
...	**...**	...	...	...
37	**0**	1	1	1
	1	1	1	1
38	**0**	1	1	1
	1	24	24	24
	2	4	4	4

94 rows × 3 columns

```
newdf.groupby(['statezip','bedrooms']).count()
```

		bathrooms	condition	city
statezip	**bedrooms**			
0	**0**	2	2	2
	1	24	24	24
	2	9	9	9
1	**0**	4	4	4
	1	17	17	17
...	**...**	...	...	...
72	**0**	6	6	6
	1	14	14	14
	2	4	4	4
73	**0**	1	1	1
74	**1**	2	2	2

197 rows × 3 columns

7]:

```
filter                                                          #cath+numeri=Anova
```

7]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street
0	313000.0000	1	3	1340.0	7683.5	1.5	0	0	1	1340	0	1955.000000	2005	18810 Densmore Ave N
2	342000.0000	1	5	1930.0	7683.5	1.0	0	0	2	1930	0	1966.000000	0	26206-26214 143rd Ave SE
4	550000.0000	2	7	1940.0	7683.5	1.0	0	0	2	1140	800	1976.000000	1992	9105 170th Ave NE
5	490000.0000	0	1	880.0	7683.5	1.0	0	0	1	880	0	1938.000000	1994	522 NE 88th St
6	335000.0000	0	5	1350.0	7683.5	1.0	0	0	1	1350	0	1976.000000	0	2616 174th Ave NE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4593	289373.3077	1	7	2538.0	4600.0	2.0	0	0	1	2538	0	1970.808827	1923	5703 Charlotte Ave SE
4594	210614.2857	1	7	1610.0	7223.0	2.0	0	0	1	1610	0	1970.808827	0	26306 127th Ave

```
newdf1=filter[['sqft_living','sqft_lot','yr_built','yr_renovated','be
newdf1
```

	sqft_living	sqft_lot	yr_built	yr_renovated	bedrooms
0	1340.0	7683.5	1955.000000	2005	1
2	1930.0	7683.5	1966.000000	0	1
4	1940.0	7683.5	1976.000000	1992	2
5	880.0	7683.5	1938.000000	1994	0
6	1350.0	7683.5	1976.000000	0	0
...	...	...	...	...	...
4593	2538.0	4600.0	1970.808827	1923	1
4594	1610.0	7223.0	1970.808827	0	1
4595	1510.0	6360.0	1970.808827	1979	1
4596	1460.0	7573.0	1970.808827	2009	1
4599	1490.0	8102.0	1970.808827	0	1

2222 rows × 5 columns

```
X=newdf1[['sqft_living','sqft_lot','yr_built','yr_renovated']]
x
```

	bathrooms	condition	city	statezip
0	3	1	32	60
2	5	2	16	25
4	7	2	27	29
5	1	1	31	52
6	5	1	27	29
...	...	...	...	...
4593	7	1	1	43
4594	7	1	16	18
4595	4	2	31	60
4596	7	1	3	6
4599	7	2	8	25

2222 rows × 4 columns

```
Y=newdf1[['bedrooms']]
Y
```

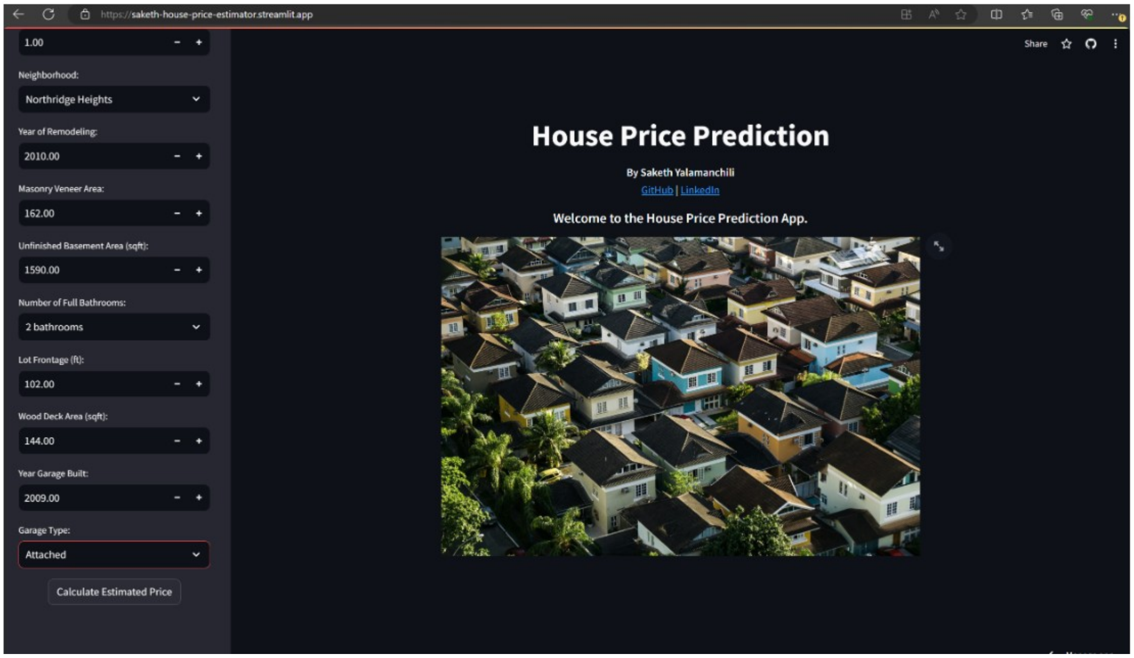
	bedrooms
0	1
2	1
4	2
5	0
6	0
...	...
4593	1
4594	1
4595	1
4596	1
4599	1

2222 rows × 1 columns

```
import pandas as pd
P_value=pd.Series(P_values[1])
P_value.index=X.columns
P_value
```

```
sqft_living     3.485676e-124
sqft_lot         3.180970e-01
yr_built         7.514685e-28
yr_renovated     2.783287e-39
dtype: float64
```

Conclusion:



In conclusion, this project represents a comprehensive journey from data preprocessing and feature engineering to model selection, training, evaluation, and deployment. It involved overcoming various challenges and culminated in the creation of a functional predictive model and user-friendly interface for house price prediction.