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

sns.set(color_codes = True)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
df = pd.read_csv('/content/raw_house_data.csv')
df
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_bu
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1
...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1

5000 rows × 16 columns

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

```
#Code to check for duplicates
#df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

```
0
```

```
# Check the structure of the Dataset
df.dtypes
```

```
                         0
             MLS       int64
      sold_price     float64
         zipcode       int64
       longitude     float64
        latitude     float64
       lot_acres     float64
           taxes     float64
      year_built       int64
        bedrooms       int64
       bathrooms     float64
         sqrt_ft     float64
          garage     float64
kitchen_features      object
      fireplaces      object
  floor_covering      object
             HOA      object

dtype: object
```

```
#Check for null/empty values in the dataset and count
df.isnull().sum()
```

	0
MLS	0
sold_price	0
zipcode	0
longitude	0
latitude	0
lot_acres	10
taxes	0
year_built	0
bedrooms	0
bathrooms	6
sqrt_ft	56
garage	7
kitchen_features	33
fireplaces	0
floor_covering	1
HOA	562

dtype: int64

```
#This is bathrooms column with values Null / Nan
vals = [np.NaN]
maskBath = df["sqrt_ft"].isin(vals)#.sum()
df[maskBath]
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	
490	3055989	950000.0	85646	-111.073405	31.619537	4.40	13193.80	
967	3058213	695000.0	85645	-111.183593	31.702330	NaN	2480.58	
1064	3056708	785045.0	85646	-110.942060	31.552399	73.42	20761.40	
1373	3059704	750000.0	85622	-111.001762	31.841975	2.72	7169.90	
1659	3055188	700000.0	85646	-111.046366	31.623839	NaN	6740.66	
1728	3057818	565000.0	85646	-111.050885	31.627210	0.72	4651.00	
1729	3044500	675000.0	85629	-110.961128	31.869810	1.02	4662.64	
1730	3053678	700000.0	85645	-111.239637	31.662369	172.76	7501.42	
1731	3059581	715000.0	85622	-111.040615	31.804808	4.72	3841.03	
1863	3052969	750000.0	85622	-111.002640	31.846861	4.58	4578.00	
2025	3044867	660000.0	85614	-110.969465	31.836723	3.60	5526.00	
2106	3056848	550000.0	85645	-111.047608	31.700763	50.00	25113.45	
2108	3059493	705000.0	85614	-110.960333	31.854886	1.06	6628.17	

2357	3060312	690000.0	85646	-111.052693	31.630004	1.85	4884.00
2401	3062128	685000.0	85614	-110.961349	31.856615	1.10	5898.42
2447	3057749	620000.0	85646	-111.066782	31.601831	10.31	5365.14
2564	3051223	680000.0	85622	-111.000925	31.836922	NaN	4158.40
2635	3060713	650000.0	85646	-111.043573	31.633469	0.90	4235.00
2636	3050955	565000.0	85622	-111.000616	31.843046	2.74	3787.80
2766	3042851	575000.0	85614	-110.960497	31.854446	0.87	4623.05
2876	3059328	560100.0	85646	-111.050957	31.626585	0.78	4716.00
2915	3055533	625000.0	85614	-110.974513	31.836495	2.01	5605.00
2917	3055386	580000.0	85629	-110.941544	31.879379	0.27	765.87
2932	3052988	625000.0	85614	-110.960215	31.857286	0.97	5518.75
2939	3060029	655000.0	85614	-110.971212	31.835117	NaN	6433.00
3108	3047540	610000.0	85614	-111.002544	31.840061	1.70	3800.00
3299	3047349	540000.0	85646	-111.051221	31.631630	0.90	8590.38
3372	3061474	600000.0	85646	-111.027596	31.648478	10.00	2903.76

3404	3061325	625000.0	85614	-110.963597	31.848245	1.38	5947.00
3420	3059875	605000.0	85622	-111.009941	31.839620	NaN	6134.56
3529	3046317	535000.0	85614	-110.986426	31.806614	4.27	3826.25
3530	3050480	580000.0	85622	-111.007069	31.846199	NaN	4498.01
3531	3054250	630000.0	85622	-111.012511	31.843514	2.98	7248.68
3556	3054492	624900.0	85614	-110.943269	31.880472	0.25	302.26
3620	3056206	580000.0	85614	-110.975396	31.830974	6.52	5902.03
3647	3061363	606000.0	85614	-110.970832	31.842606	1.54	2016.00
3672	3061172	610000.0	85622	-111.007363	31.841543	1.58	5461.97
3818	3052078	570000.0	85622	-111.013274	31.841175	1.00	3377.68
3819	3060897	570000.0	85614	-110.978669	31.832586	4.00	5222.20
3822	3045347	550000.0	85614	-111.008754	31.841141	0.99	3702.07
3983	3050843	579000.0	85622	-111.005401	31.838947	1.60	3689.00
3984	3052794	540000.0	85622	-111.001859	31.837066	0.95	4093.07

```
4003  3059616  600000.0  85614  -110.978365  31.823326   3.00  5493.00

4020  3059457  585000.0  85614  -110.942704  31.897662   0.21  4578.68

4173  3059167  550000.0  85614  -110.940656  31.878329    NaN  5630.10

4273  3056944  490000.0  85601  -111.299661  31.584170  38.98  5739.00

4286  3050688  584165.0  85622  -111.039164  31.790671    NaN   653.47

4365  3053499  545000.0  85614  -110.978688  31.832598    NaN  4817.00

4662  3051343  544000.0  85622  -111.004190  31.835595   1.00  3275.00

4723  3053192  540000.0  85614  -110.970464  31.837728   2.08  5077.95

4724  3055336  510000.0  85646  -111.091642  31.582936   7.00  4405.70

4783  3058623  530000.0  85622  -111.009020  31.839576    NaN  5264.34

4812  3046287  500000.0  85646  -111.051431  31.636207   1.03  8102.00

4991  3052471  525000.0  85622  -111.038888  31.791324   0.95  3919.93

4992  3056450  525000.0  85614  -110.980945  31.824287   3.01  5122.84
```

```
#This is to return indexes or rows for lot_acres column with values Null / Nan
vals = [np.NaN]
mask = df["lot_acres"].isin(vals)
df[mask].index
```

```
Index([967, 1659, 2564, 2939, 3420, 3530, 4173, 4286, 4365, 4783], dtype='int64')
```

```
#this removes all records/rows with null value in the Lot_acres column. (This is because the
df = df.drop(df.index[df[mask].index], axis=0)#, inplace=True)
```

```
df
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_bu
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1
...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1

4990 rows × 16 columns

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

```
#Updating some few Columns[Garage,fireplaces,HOA,Kitchen_Features,Floor_Covering] with 0 / "
df["garage"] = df["garage"].fillna(0)
df["HOA"] = df["HOA"].fillna(0)
df["fireplaces"] = df["fireplaces"].replace(" ", 0)

# updating Kitchen_features and Floor_covering to Nothing where value is null
df["kitchen_features"] = df["kitchen_features"].replace(np.NaN, "Nothing")
```

```
df["floor_covering"] = df["floor_covering"].replace(np.NaN, "Nothing")
df
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_bu
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1
...	...	...	...	...	...	...	...	
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
4997	21832452	475000.0	85192	-110.755428	32.964708	12.06	1000.00	1
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2
4999	4111490	450000.0	85621	-110.913054	31.385259	4.16	2814.48	1

4990 rows × 16 columns

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

```
#Fixing null / empty values for the bathroom column

zeroBathrooms = df[df["bathrooms"].isnull()] #The immediate code snippet assigns the columns
zeroBathrooms
```

```
zBathrmsIndex = zeroBathrooms.index #The index of the columns is stored in a variable so tha
zBathrmsIndex
```

```
Index([2025, 2766, 3108, 3529, 3822, 4812], dtype='int64')
```

```
#This code Iterates through the dataframe using the zBathrooms index
for i in zBathrmsIndex:
  bathVals = df[(df["bedrooms"] == df.loc[i, "bedrooms"])][["bathrooms"]].mode() # for every
  bathVals = bathVals["bathrooms"] # The bathroom column of the bathVals dataframe is assigr
  df.loc[i, "bathrooms"] = bathVals[0] # The value of bathVals is assigned to the respective
  #print(df.loc[i, "bathrooms"]) # The update values are printed out.
```

```
nSqrft = df[df["sqrt_ft"].isnull()]
nSqrft
nSqrftIndex = nSqrft.index
nSqrftIndex
```

```
Index([   2,  490, 1064, 1373, 1728, 1729, 1730, 1731, 1863, 2025, 2106, 2108,
        2357, 2401, 2447, 2635, 2636, 2766, 2876, 2915, 2917, 2932, 3108, 3299,
        3372, 3404, 3529, 3531, 3556, 3620, 3647, 3672, 3818, 3819, 3822, 3983,
        3984, 4003, 4020, 4273, 4662, 4723, 4724, 4812, 4991, 4992],
       dtype='int64')
```

```
for i in nSqrftIndex:
  sqrtftVals = df[(df["bedrooms"] == df.loc[i, "bedrooms"])][["sqrt_ft"]].mean() // 1 # for
  sqrtftVals = sqrtftVals["sqrt_ft"] # The sqrt_ft column of the sqrtftVals dataframe is ass
  df.loc[i, "sqrt_ft"] = sqrtftVals # The value of sqrtftVals is assigned to the respective
  #print(sqrtftVals) # The value of sqrtftVals is assigned to the respective sqrt_ft columns
```

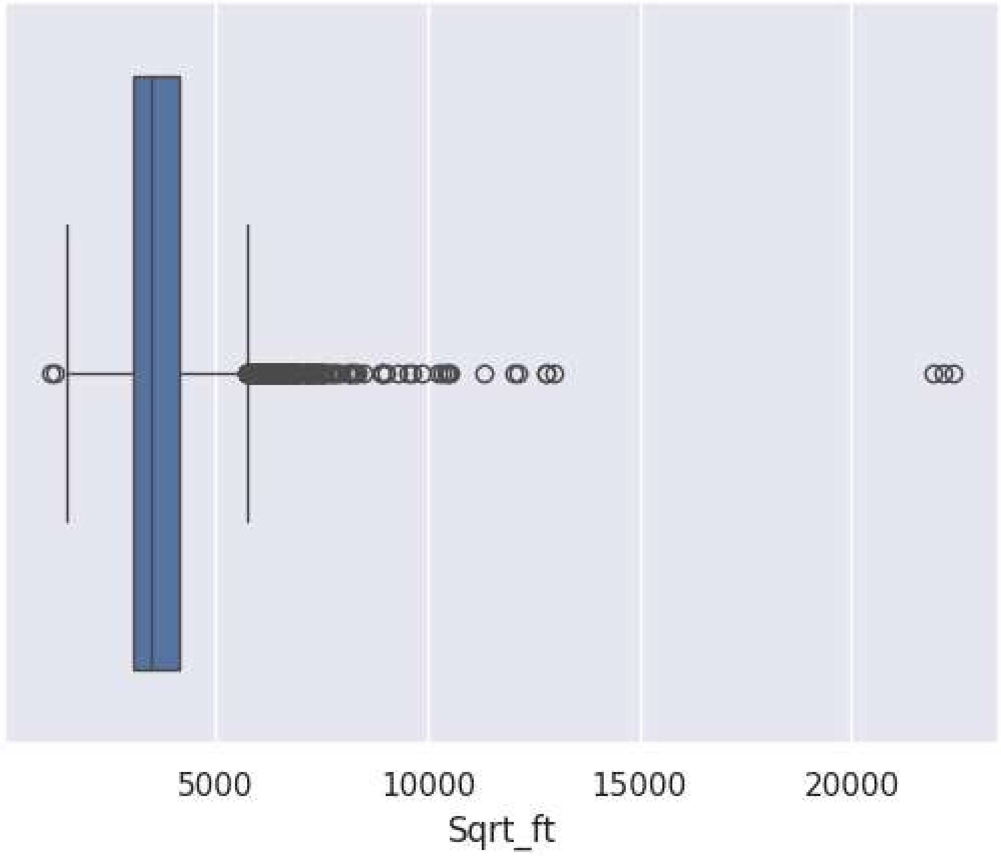
```
df.describe()
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	
count	4.990000e+03	4.990000e+03	4990.000000	4990.000000	4990.000000	4990.000000	4.9
mean	2.130720e+07	7.749513e+05	85723.223447	-110.911893	32.309526	4.661317	9.4
std	2.257876e+06	3.187799e+05	37.838772	0.120623	0.176727	51.685230	1.7
min	3.042851e+06	1.690000e+05	85118.000000	-112.520168	31.356362	0.000000	0.0
25%	2.140750e+07	5.850000e+05	85718.000000	-110.979109	32.277974	0.580000	4.8
50%	2.161501e+07	6.750000e+05	85737.000000	-110.923309	32.318570	0.990000	6.2
75%	2.180494e+07	8.367500e+05	85749.000000	-110.859025	32.394625	1.757500	8.0
max	2.192856e+07	5.300000e+06	86323.000000	-109.454637	34.927884	2154.000000	1.2

```
#Renaming columns
df = df.rename(columns={'sold_price' : 'Sold_Price', 'zipcode' : 'Zipcode', 'longitutde' : '
                   'taxes' : 'Taxes', 'year_built':'Year_Built', 'bedrooms' : 'Bedrooms', 'b
                        'kitchen_features' : 'Kitchen_Features', 'fireplaces' : 'Fireplaces'
```

```
sns.boxplot(x=df["Sqrt_ft"])
```

```
<Axes: xlabel='Sqrt_ft'>
```



```
plt.figure(figsize=(10,7))
plt.hist(df['Bedrooms'], bins=30)
plt.title('Bedroom Distribution of Houses Sold')
plt.xlabel('Bedroom')
plt.ylabel('Count')
```