Exploratory Data Analysis (EDA) - Real State Analysis

```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

df = pd.read_csv('combined_csv.csv')
df.head(10)
```

```
   SALE TYPE          SOLD DATE              PROPERTY TYPE  \
0  PAST SALE  September-15-2022                Condo/Co-op   
1  PAST SALE     August-31-2022                Condo/Co-op   
2  PAST SALE  September-30-2022  Single Family Residential   
3  PAST SALE  September-28-2022                Vacant Land   
4  PAST SALE     August-25-2022                Condo/Co-op   
5  PAST SALE    October-12-2022                Vacant Land   
6  PAST SALE     August-31-2022                Condo/Co-op   
7  PAST SALE  September-30-2022  Single Family Residential   
8  PAST SALE     August-25-2022                Condo/Co-op   
9  PAST SALE    October-12-2022                Vacant Land   

                           ADDRESS           CITY STATE OR PROVINCE  \
0             84-770 Kili Dr #1440        Waianae                HI   
1  85-175 Farrington Hwy Unit C112        Waianae                HI   
2      16-743 Wao Kele Rd (road G)  Mountain View                HI   
3                       Lot Unit 7       Papaikou                HI   
4             4999 Kahala Ave #271       Honolulu                HI   
5                                _  Mountain View                HI   
6  85-175 Farrington Hwy Unit C112        Waianae                HI   
7      16-743 Wao Kele Rd (road G)  Mountain View                HI   
8             4999 Kahala Ave #271       Honolulu                HI   
9                                _  Mountain View                HI   

  ZIP OR POSTAL CODE   PRICE  BEDS  BATHS  ... STATUS  \
0              96792  159000   0.0    1.0  ...    Sold   
1              96792  160000   0.0    1.0  ...    Sold   
2              96771  160000   2.0    1.0  ...    Sold   
3              96781  159000   NaN    NaN  ...    Sold   
4              96816  160000   2.0    2.0  ...    Sold   
5              96771  160000   NaN    NaN  ...    Sold   
6              96792  160000   0.0    1.0  ...    Sold   
7              96771  160000   2.0    1.0  ...    Sold   
8              96816  160000   2.0    2.0  ...    Sold   
9              96771  160000   NaN    NaN  ...    Sold   

   NEXT OPEN HOUSE START TIME  NEXT OPEN HOUSE END TIME  \
0                         NaN                       NaN
1                         NaN                       NaN
2                         NaN                       NaN
3                         NaN                       NaN
4                         NaN                       NaN
5                         NaN                       NaN
6                         NaN                       NaN
7                         NaN                       NaN
8                         NaN                       NaN
9                         NaN                       NaN

   URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)  \
0  https://www.redfin.com/HI/Waianae/84-770-Kili-...

1  https://www.redfin.com/HI/Waianae/85-175-Farri...

2  https://www.redfin.com/HI/Mountain-View/16-743...

3  https://www.redfin.com/HI/Papaikou/Lot-96781/u...

4  https://www.redfin.com/HI/Honolulu/4999-Kahala...

5  https://www.redfin.com/HI/Mountain-View/Unknow...

6  https://www.redfin.com/HI/Waianae/85-175-Farri...

7  https://www.redfin.com/HI/Mountain-View/16-743...

8  https://www.redfin.com/HI/Honolulu/4999-Kahala...

9  https://www.redfin.com/HI/Mountain-View/Unknow...


                   SOURCE         MLS#  FAVORITE INTERESTED   LATITUDE
\
0           HiCentral MLS  202203123.0         N          Y  21.482810

1           HiCentral MLS  202208655.0         N          Y  21.457467

2  HI Information Service     664218.0         N          Y  19.500440

3  HI Information Service     661374.0         N          Y  19.801533

4           HiCentral MLS  202209911.0         N          Y  21.271372

5  HI Information Service     664972.0         N          Y  19.556521
```

```
6           HiCentral MLS  202208655.0         N          Y  21.457467

7  HI Information Service     664218.0         N          Y  19.500440

8           HiCentral MLS  202209911.0         N          Y  21.271372

9  HI Information Service     664972.0         N          Y  19.556521


    LONGITUDE
0 -158.203623
1 -158.202598
2 -155.022297
3 -155.108610
4 -157.775244
5 -155.107028
6 -158.202598
7 -155.022297
8 -157.775244
9 -155.107028

[10 rows x 27 columns]
```

```
df.shape
```

```
(809, 27)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 809 entries, 0 to 808
Data columns (total 27 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   SALE TYPE
809 non-null    object
 1   SOLD DATE
801 non-null    object
 2   PROPERTY TYPE
809 non-null    object
 3   ADDRESS
808 non-null    object
 4   CITY
809 non-null    object
 5   STATE OR PROVINCE
809 non-null    object
 6   ZIP OR POSTAL CODE
809 non-null    object
```

```
 7   PRICE
809 non-null    int64
 8   BEDS
552 non-null    float64
 9   BATHS
533 non-null    float64
 10  LOCATION
801 non-null    object
 11  SQUARE FEET
536 non-null    float64
 12  LOT SIZE
720 non-null    float64
 13  YEAR BUILT
537 non-null    float64
 14  DAYS ON MARKET
0 non-null      float64
 15  $/SQUARE FEET
536 non-null    float64
 16  HOA/MONTH
355 non-null    float64
 17  STATUS
801 non-null    object
 18  NEXT OPEN HOUSE START TIME
0 non-null      float64
 19  NEXT OPEN HOUSE END TIME
0 non-null      float64
 20  URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)  809 non-null    object
 21  SOURCE
801 non-null    object
 22  MLS#
801 non-null    float64
 23  FAVORITE
809 non-null    object
 24  INTERESTED
809 non-null    object
 25  LATITUDE
809 non-null    float64
 26  LONGITUDE
809 non-null    float64
dtypes: float64(13), int64(1), object(13)
memory usage: 170.8+ KB
```

```
df.describe()
```

```
              PRICE        BEDS       BATHS  SQUARE FEET      LOT SIZE
\
count  8.090000e+02  552.000000  533.000000   536.000000  7.200000e+02

mean   4.541100e+05    2.235507    1.733583  1128.283582  1.310433e+05
```

```
std    6.107641e+05    1.794013    0.929031   783.162290  3.787302e+05

min    5.264000e+03    0.000000    1.000000    29.000000  9.920000e+02

25%    9.000000e+04    1.000000    1.000000   545.500000  8.344750e+03

50%    1.820000e+05    2.000000    1.500000   922.000000  2.150000e+04

75%    7.250000e+05    3.000000    2.000000  1614.000000  8.713100e+04

max    6.750000e+06   18.000000   10.000000  6838.000000  4.295321e+06


        YEAR BUILT  DAYS ON MARKET  $/SQUARE FEET    HOA/MONTH  \
count   537.000000             0.0     536.000000   355.000000
mean   1982.923650             NaN     516.440299   531.873239
std      20.228816             NaN     345.825476   371.519955
min    1920.000000             NaN      17.000000     4.000000
25%    1970.000000             NaN     282.750000   243.000000
50%    1978.000000             NaN     471.500000   498.000000
75%    1996.000000             NaN     656.000000   714.000000
max    2023.000000             NaN    3500.000000  2095.000000

       NEXT OPEN HOUSE START TIME  NEXT OPEN HOUSE END TIME
MLS#  \
count                         0.0                       0.0
8.010000e+02
mean                          NaN                       NaN
7.510187e+07
std                           NaN                       NaN
9.737080e+07
min                           NaN                       NaN
3.913990e+05
25%                           NaN                       NaN
6.610580e+05
50%                           NaN                       NaN
6.643950e+05
75%                           NaN                       NaN
2.022097e+08
max                           NaN                       NaN
2.022237e+08

         LATITUDE   LONGITUDE
count  809.000000  809.000000
mean    20.459630 -156.524951
std      0.928859    1.368855
min     19.032161 -159.714066
25%     19.535107 -157.841293
```

```
50%     20.846039 -156.452392
75%     21.309331 -155.103069
max     22.226329 -154.892380
```

```
df.describe(include = ['O'])
```

```
        SALE TYPE       SOLD DATE              PROPERTY TYPE  \
count         809             801                        809   
unique          1              76                          7   
top     PAST SALE  August-25-2022  Single Family Residential   
freq          809              25                        284   

                  ADDRESS      CITY STATE OR PROVINCE ZIP OR POSTAL 
CODE  \
count                 808       809               809               
809   
unique                689        51                 1                
76   
top     4th Ave (awapuhi)  Honolulu                HI             
96778   
freq                    5       122               809                
87   

       LOCATION STATUS  \
count       801    801   
unique      168      1   
top     Waikiki   Sold   
freq         62    801   

       URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)  \
count                                                 809                                                      

unique                                                698                                                      

top     https://www.redfin.com/HI/Waianae/84-770-Kili-...                                                      

freq                                                    2                                                      


                        SOURCE FAVORITE INTERESTED  
count                      801      809        809  
unique                       3        1          1  
top     HI Information Service        N          Y  
freq                       388      809        809  
```

```
df['SALE TYPE'].value_counts()
```

```
PAST SALE    809
Name: SALE TYPE, dtype: int64
```

```
df['STATE OR PROVINCE'].value_counts()
```

```
HI    809
Name: STATE OR PROVINCE, dtype: int64
```

```
df['FAVORITE'].value_counts()
```

```
N    809
Name: FAVORITE, dtype: int64
```

```
df['INTERESTED'].value_counts()
```

```
Y    809
Name: INTERESTED, dtype: int64
```

```
df['STATUS'].value_counts()
```

```
Sold    801
Name: STATUS, dtype: int64
```

```
#drop the columns, which has no data
#drop columns, which has same data

df = df.drop(['SALE TYPE', 'STATE OR PROVINCE', 'FAVORITE',
'INTERESTED', 'STATUS', 'DAYS ON MARKET', 'NEXT OPEN HOUSE START
TIME', 'NEXT OPEN HOUSE END TIME'], axis = 1)
```

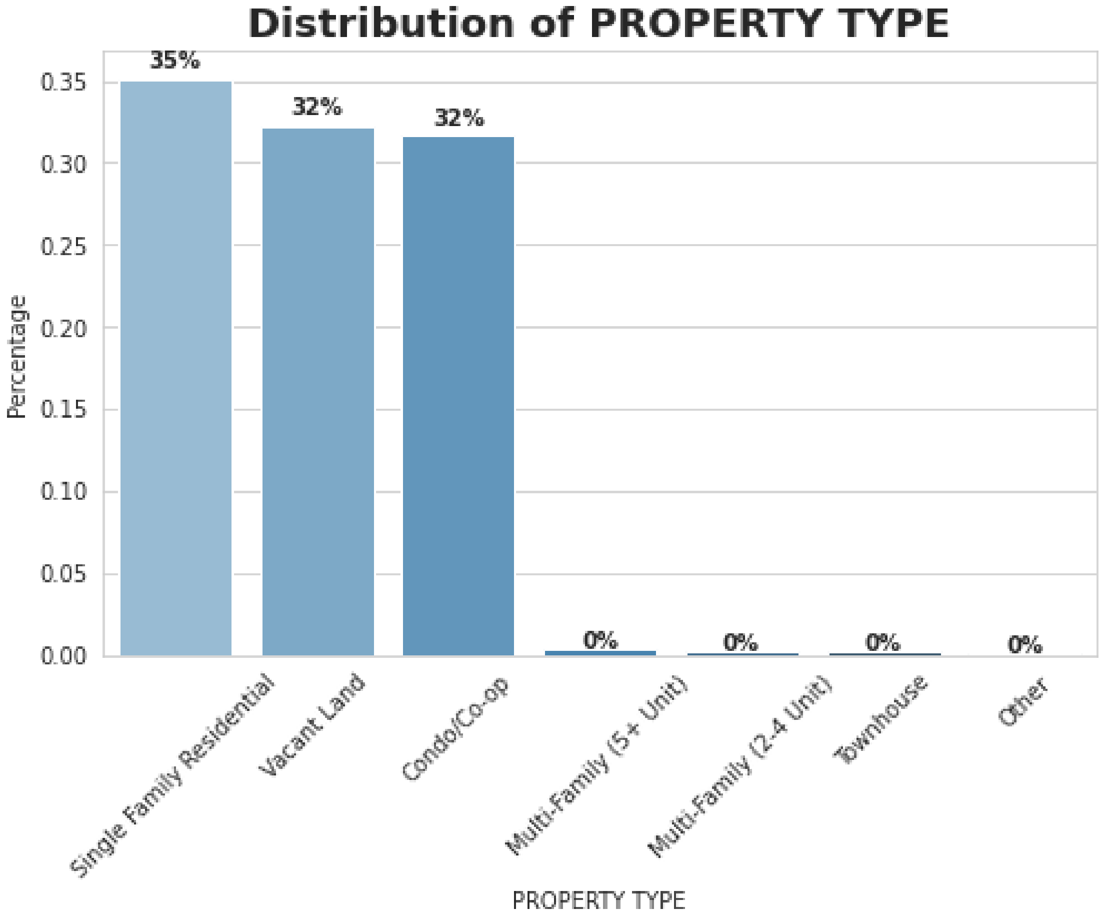
```
df.shape
```

```
(809, 19)
```

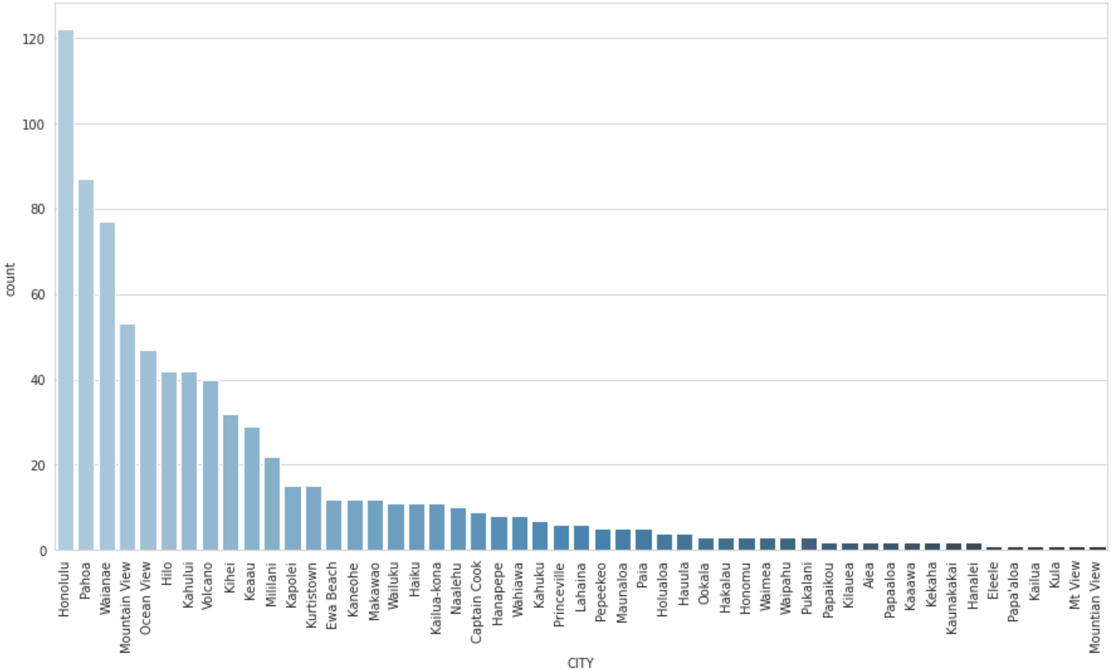
```
df['PROPERTY TYPE'].value_counts().reset_index()
```

```
                       index  PROPERTY TYPE
0  Single Family Residential            284
1                Vacant Land            261
2                Condo/Co-op            256
3     Multi-Family (5+ Unit)              3
4    Multi-Family (2-4 Unit)              2
5                  Townhouse              2
6                      Other              1
```

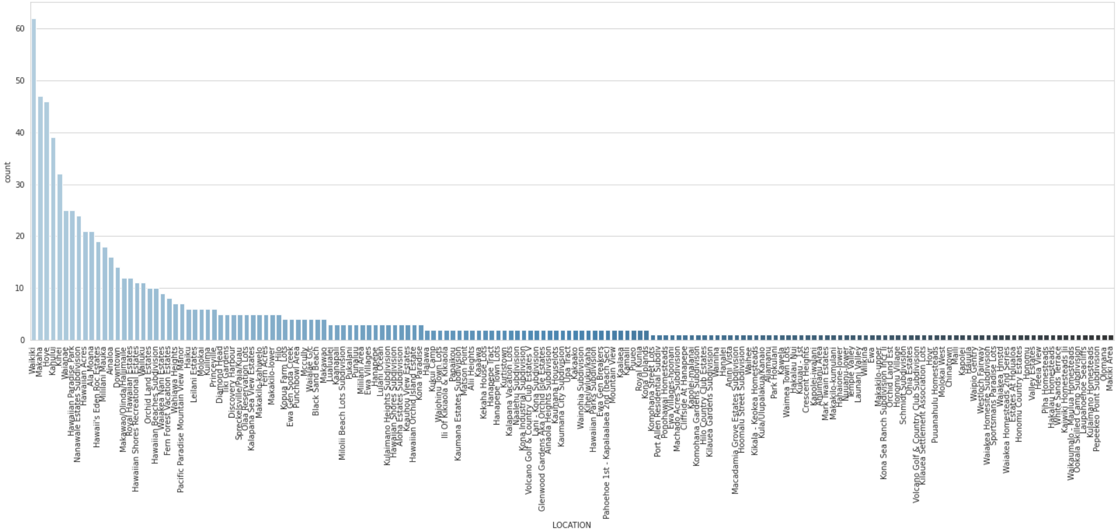
```
sns.set_style("whitegrid")
plt.figure(figsize = (8,5))
plt.title('Distribution of PROPERTY TYPE', fontsize=18,
fontweight='bold')
eda_percentage = df['PROPERTY TYPE'].value_counts(normalize =
True).rename_axis('PROPERTY TYPE').reset_index(name = 'Percentage')

ax = sns.barplot(x = 'PROPERTY TYPE', y = 'Percentage', data =
eda_percentage.head(10), palette='Blues_d')
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02),
ha='center', fontweight='bold')
    plt.setp(ax.get_xticklabels(), rotation=45);
```



```
fig, ax = plt.subplots(figsize=(15, 8))
chart = sns.countplot(x="CITY", data=df, ax=ax, palette='Blues_d',
                      order = df['CITY'].value_counts().index)
chart.set_xticklabels(chart.get_xticklabels(), rotation=90)
plt.show()
```



```
fig, ax = plt.subplots(figsize=(25, 8))
chart = sns.countplot(x="LOCATION", data=df, ax=ax, palette='Blues_d',

                      order = df['LOCATION'].value_counts().index)
chart.set_xticklabels(chart.get_xticklabels(), rotation=90)
plt.show()
```

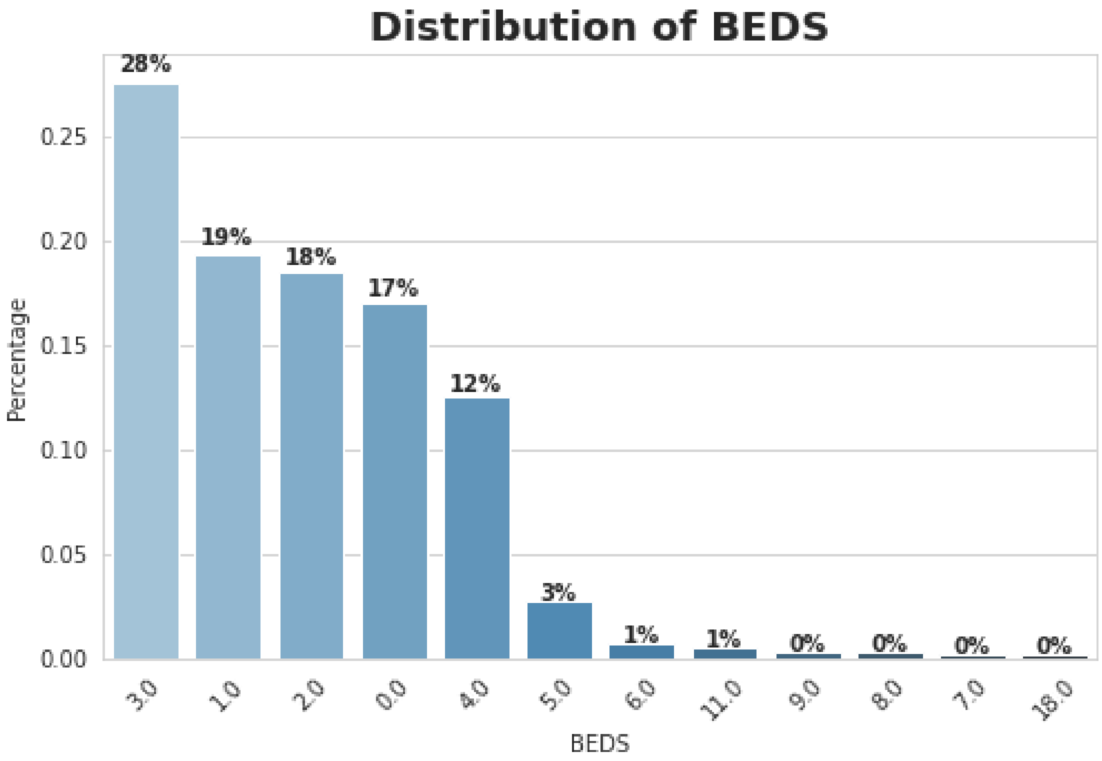


```
df['MLS#'].value_counts()
```

```
202203123.0    2
662254.0       2
663535.0       2
202208007.0    2
```

```
663759.0       2
              ..
202210561.0    1
202213665.0    1
202211939.0    1
664274.0       1
202211701.0    1
Name: MLS#, Length: 690, dtype: int64
```

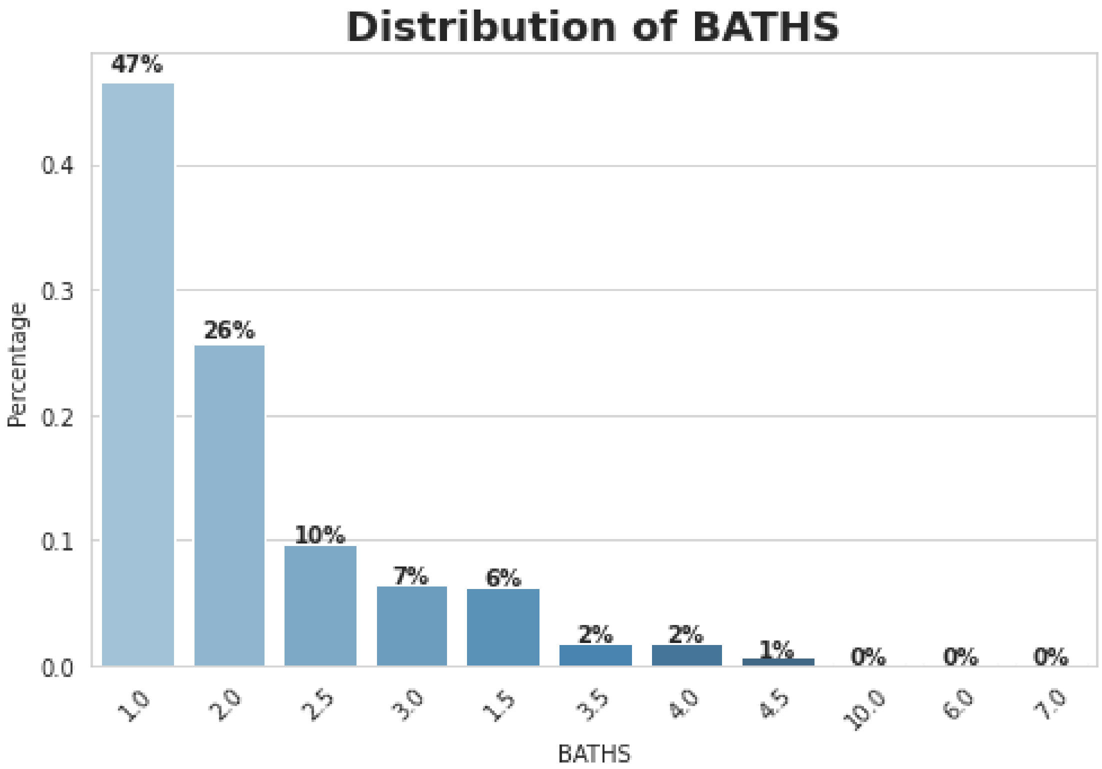
```
sns.set_style("whitegrid")
plt.figure(figsize = (8,5))
plt.title('Distribution of BEDS', fontsize=18, fontweight='bold')
eda_percentage = df['BEDS'].value_counts(normalize =
True).rename_axis('BEDS').reset_index(name = 'Percentage')

ax = sns.barplot(x = 'BEDS', y = 'Percentage', data = eda_percentage,
palette='Blues_d', order =
eda_percentage['BEDS'].value_counts().index)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02),
ha='center', fontweight='bold')
    plt.setp(ax.get_xticklabels(), rotation=45);
```



```
eda_percentage
```

```
    BEDS  Percentage
0    3.0    0.275362
1    1.0    0.193841
2    2.0    0.184783
3    0.0    0.170290
4    4.0    0.125000
5    5.0    0.027174
6    6.0    0.007246
7   11.0    0.005435
8    9.0    0.003623
9    8.0    0.003623
10   7.0    0.001812
11  18.0    0.001812
```

```
sns.set_style("whitegrid")
plt.figure(figsize = (8,5))
plt.title('Distribution of BATHS', fontsize=18, fontweight='bold')
eda_percentage = df['BATHS'].value_counts(normalize =
True).rename_axis('BATHS').reset_index(name = 'Percentage')

ax = sns.barplot(x = 'BATHS', y = 'Percentage', data = eda_percentage,
palette='Blues_d', order =
eda_percentage['BATHS'].value_counts().index)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02),
ha='center', fontweight='bold')
    plt.setp(ax.get_xticklabels(), rotation=45);
```



```
df['BATHS'].value_counts().reset_index()
```

```
    index  BATHS
0     1.0    248
1     2.0    137
2     2.5     52
3     3.0     35
4     1.5     34
5     3.5     10
6     4.0     10
7     4.5      4
8    10.0      1
9     6.0      1
10    7.0      1
```

```
# forward-fill
df = df.fillna(method='ffill')
```

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

```
SOLD DATE
0
PROPERTY TYPE
0
ADDRESS
0
CITY
```

```
0
ZIP OR POSTAL CODE
0
PRICE
0
BEDS
0
BATHS
0
LOCATION
0
SQUARE FEET
0
LOT SIZE
0
YEAR BUILT
0
$/SQUARE FEET
0
HOA/MONTH
0
URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis
FOR INFO ON PRICING)    0
SOURCE
0
MLS#
0
LATITUDE
0
LONGITUDE
0
dtype: int64
```

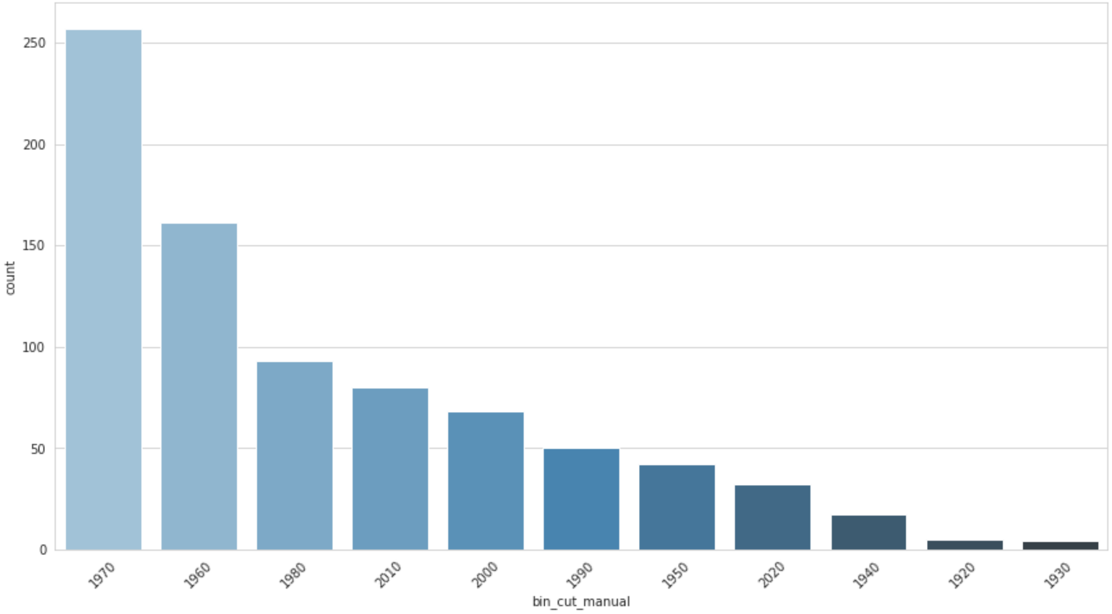
```
df['YEAR BUILT'] = df['YEAR BUILT'].astype("int")
```

```
labels = ['1920', '1930', '1940', '1950', '1960', '1970', '1980',
'1990', '2000', '2010', '2020']
bins = [ 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010,
2020, 2030 ]
df['year_bin'] = pd.cut(df['YEAR BUILT'] , bins=bins, labels=labels,
include_lowest=True)
```

```
df['year_bin'].nunique()
```

```
11
```

```
fig, ax = plt.subplots(figsize=(15, 8))
chart = sns.countplot(x="year_bin", data=df, ax=ax, palette='Blues_d',
                      order = df['year_bin'].value_counts().index)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
plt.show()
```



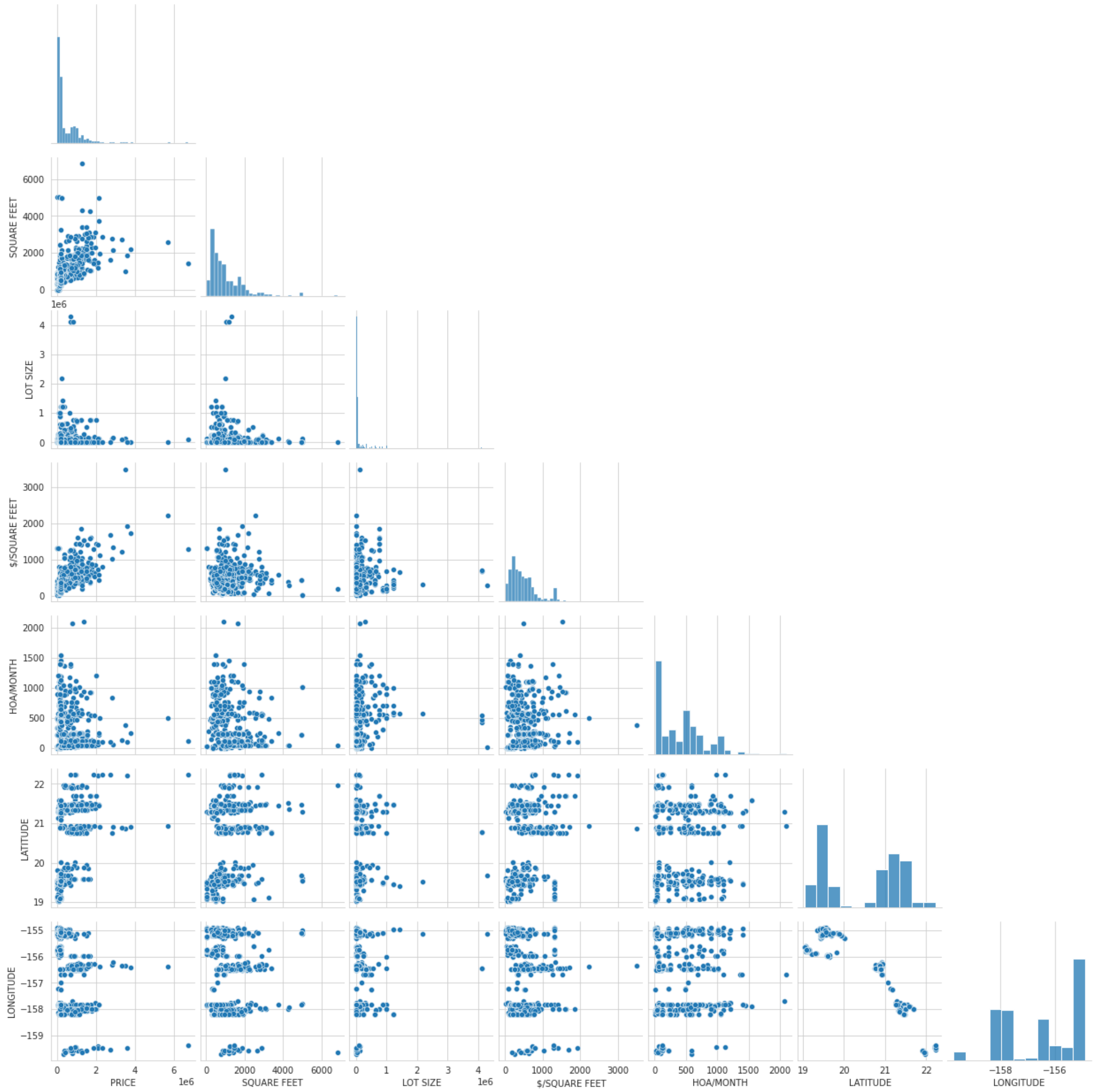
```
df.columns
```

```
Index(['SALE TYPE', 'SOLD DATE', 'PROPERTY TYPE', 'ADDRESS', 'CITY',
       'STATE OR PROVINCE', 'ZIP OR POSTAL CODE', 'PRICE', 'BEDS',
'BATHS',
       'LOCATION', 'SQUARE FEET', 'LOT SIZE', 'YEAR BUILT', '$/SQUARE
FEET',
       'HOA/MONTH', 'STATUS',
       'URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)',
       'SOURCE', 'MLS#', 'FAVORITE', 'INTERESTED', 'LATITUDE',
'LONGITUDE'],
      dtype='object')
```

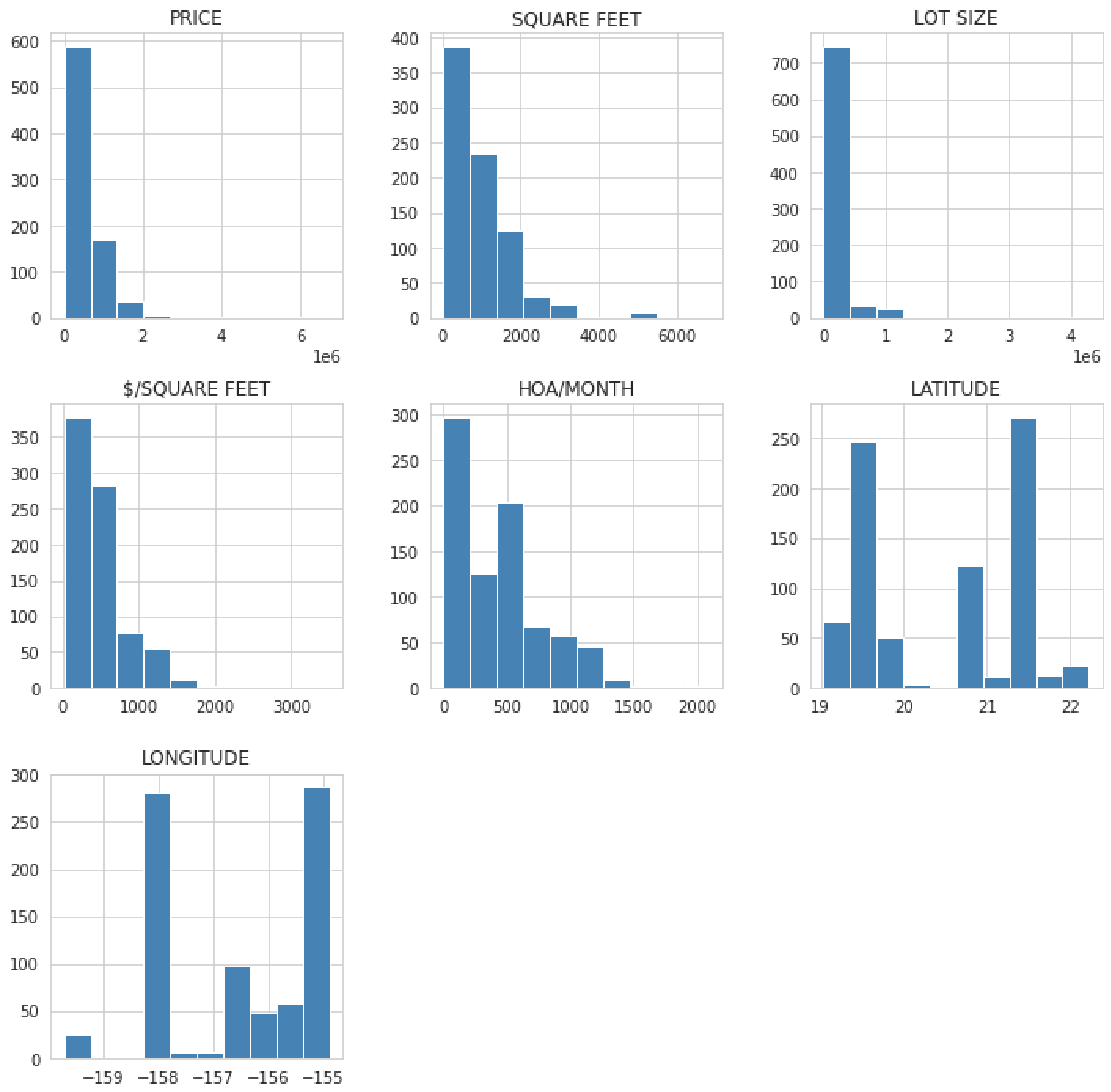
```
cts_variables = ['PRICE', 'SQUARE FEET', 'LOT SIZE', '$/SQUARE FEET',
'HOA/MONTH', 'LATITUDE', 'LONGITUDE']
```

```
sns.pairplot(df[cts_variables], corner=True)
```

```
<seaborn.axisgrid.PairGrid at 0x7f86ae801050>
```



```
# Distribution of continuous Features of the Dataset
distribution = df[cts_variables].hist( linewidth = 1.0, color = 
'steelblue')
fig = plt.gcf()
fig.set_size_inches(12,12)
plt.show()
```



```
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

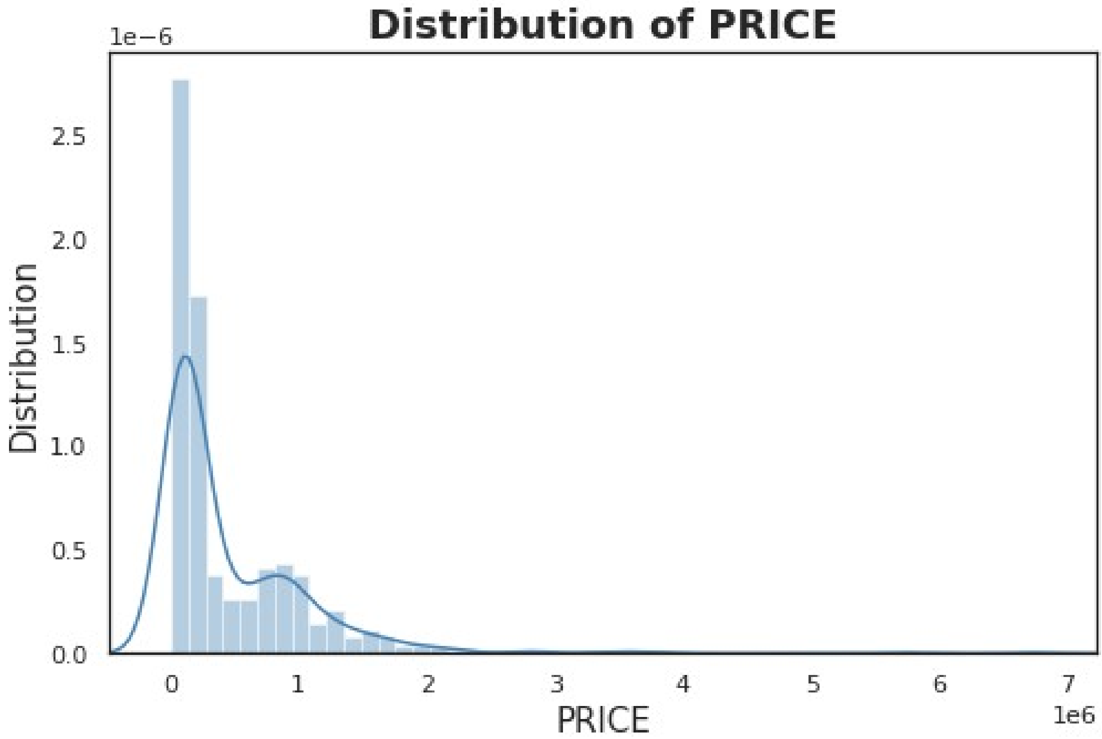
```
df['PRICE'].value_counts()
```

```
165000    14
175000    12
150000    12
140000    12
200000    11
          ..
301000     1
455000     1
669000     1
635000     1
829000     1
Name: PRICE, Length: 412, dtype: int64
```

```
plt.rcParams['figure.figsize'] = [8, 5]
sns.set(style = 'white')
plt.title('Distribution of PRICE', fontsize=18, fontweight='bold')
sns.distplot(df['PRICE'], color = "steelblue")
plt.ylabel("Distribution", fontsize = 15)
plt.xlabel("PRICE", fontsize = 15)
plt.margins(x = 0)

print ("The maximum PRICE is", df['PRICE'].max())
print ("The minimum PRICE is", df['PRICE'].min())
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

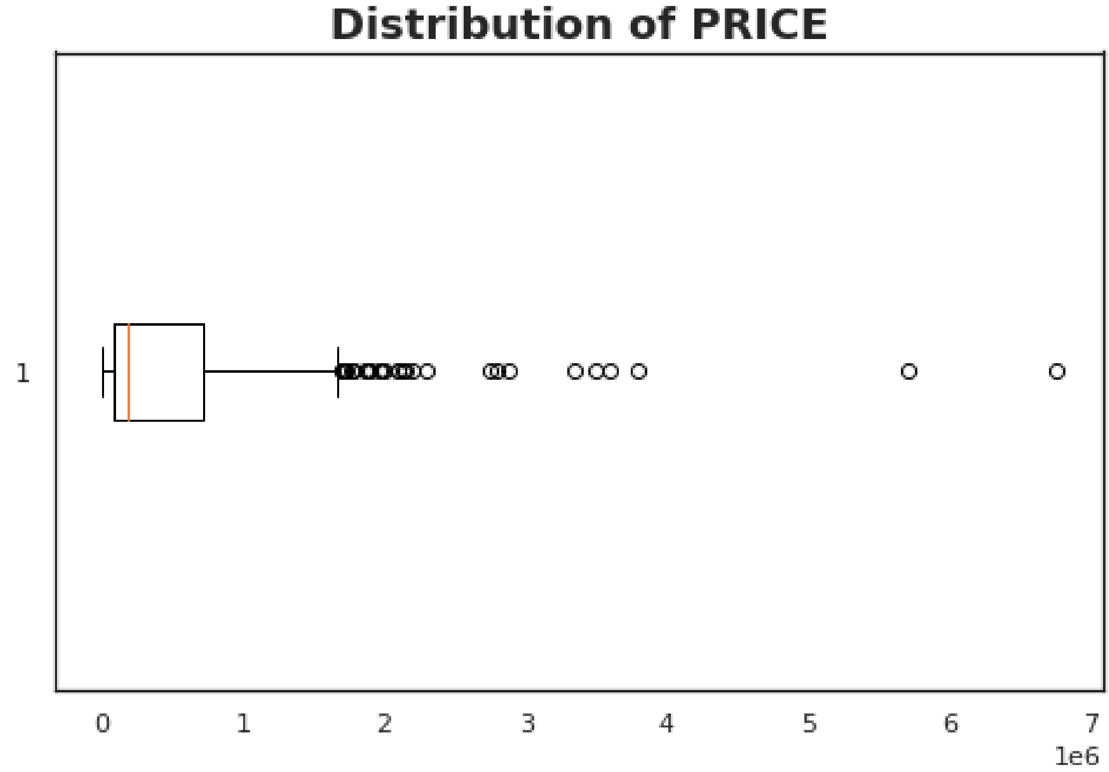
```
The maximum PRICE is 6750000
The minimum PRICE is 5264
```



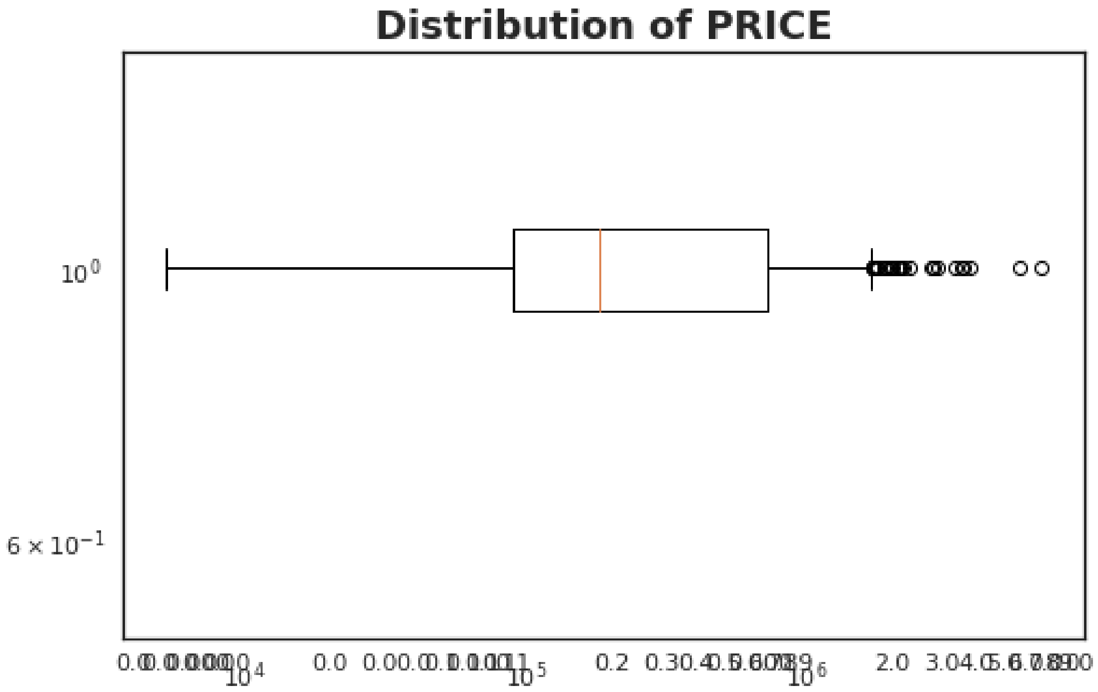
```
df['PRICE'] = df['PRICE'].astype("int")
```

```
from matplotlib import pyplot as plt, ticker as mticker
```

```
fig, ax = plt.subplots(1, 1)
# Creating plot
plt.boxplot(df['PRICE'],
            vert = 0,
           )
plt.title('Distribution of PRICE', fontsize=18, fontweight='bold')
plt.show()
```



```
fig, ax = plt.subplots(1, 1)
# Creating plot
plt.boxplot(df['PRICE'],
            vert = 0,
           )
plt.title('Distribution of PRICE', fontsize=18, fontweight='bold')
ax = plt.gca()
ax.set_xscale('log')
ax.set_yscale('log')
ax.xaxis.set_minor_formatter(mticker.ScalarFormatter())
plt.show()
```



```
df['PRICE'].describe().reset_index()
```

```
   index         PRICE
0  count  8.090000e+02
1   mean  4.541100e+05
2    std  6.107641e+05
3    min  5.264000e+03
4    25%  9.000000e+04
5    50%  1.820000e+05
6    75%  7.250000e+05
7    max  6.750000e+06
```

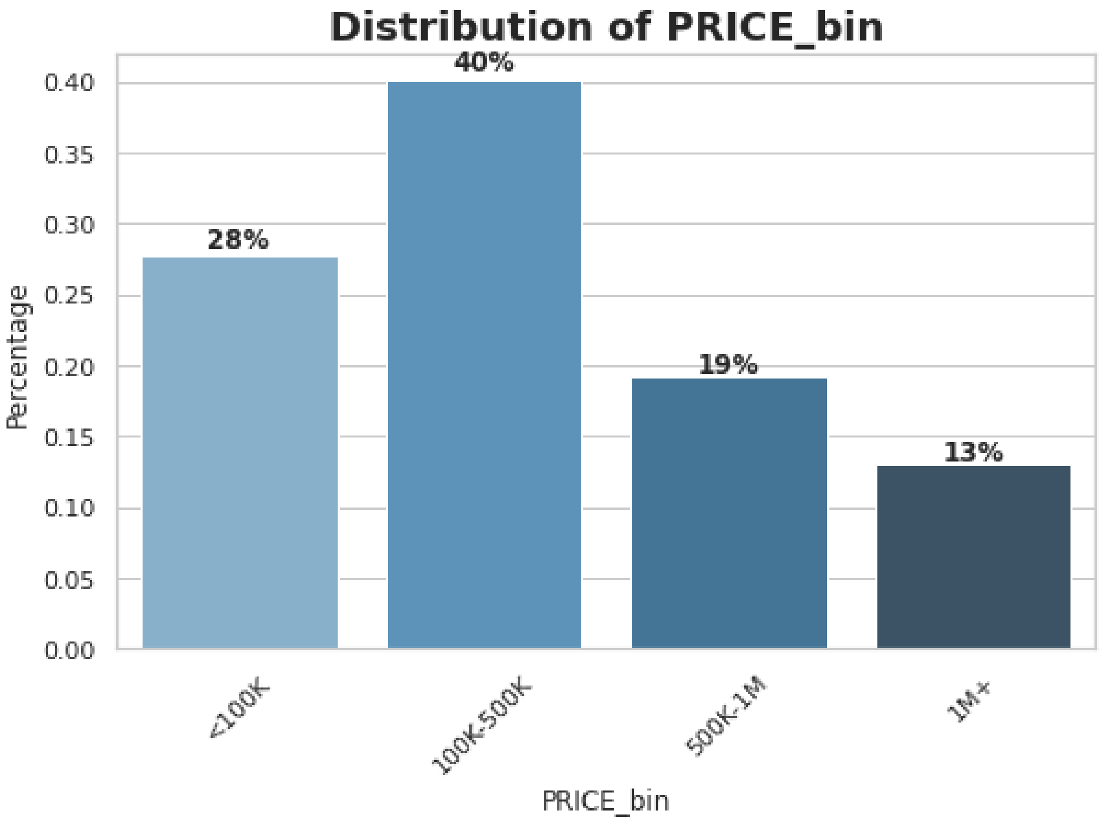
```
labels = ['<100K', '100K-500K', '500K-1M', '1M+']
bins = [ 5264, 100000, 500000, 1000000, 6750000]
df['PRICE_bin'] = pd.cut(df['PRICE'] , bins=bins, labels=labels,
include_lowest=True)

df['PRICE_bin'].nunique()
```

```
4
```

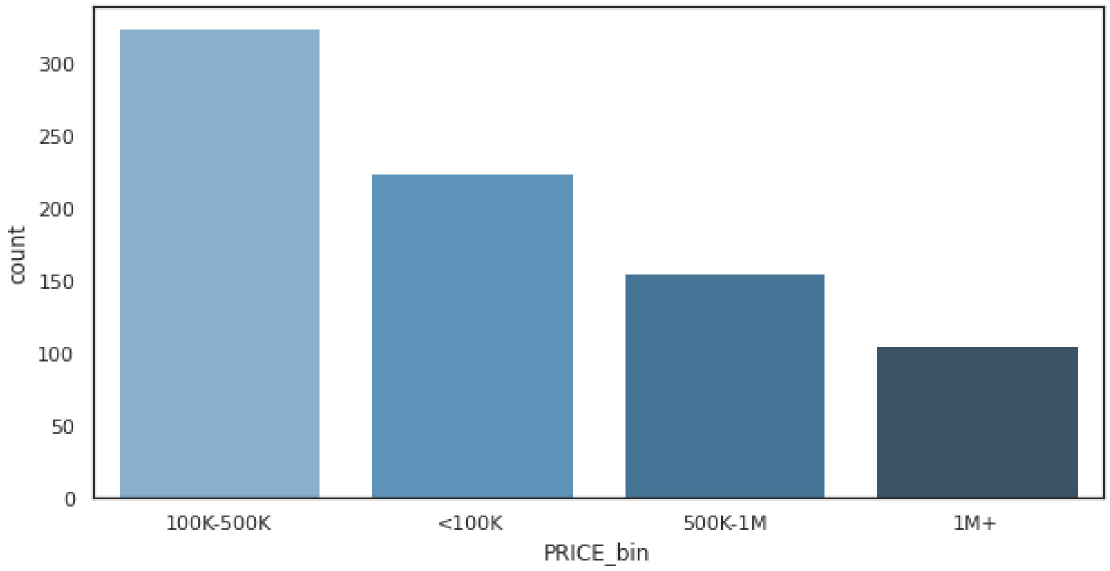
```
df['PRICE_bin'].value_counts().reset_index()
```

```
       index  PRICE_bin
0  100K-500K        324
1      <100K        225
2    500K-1M        155
3        1M+        105
```

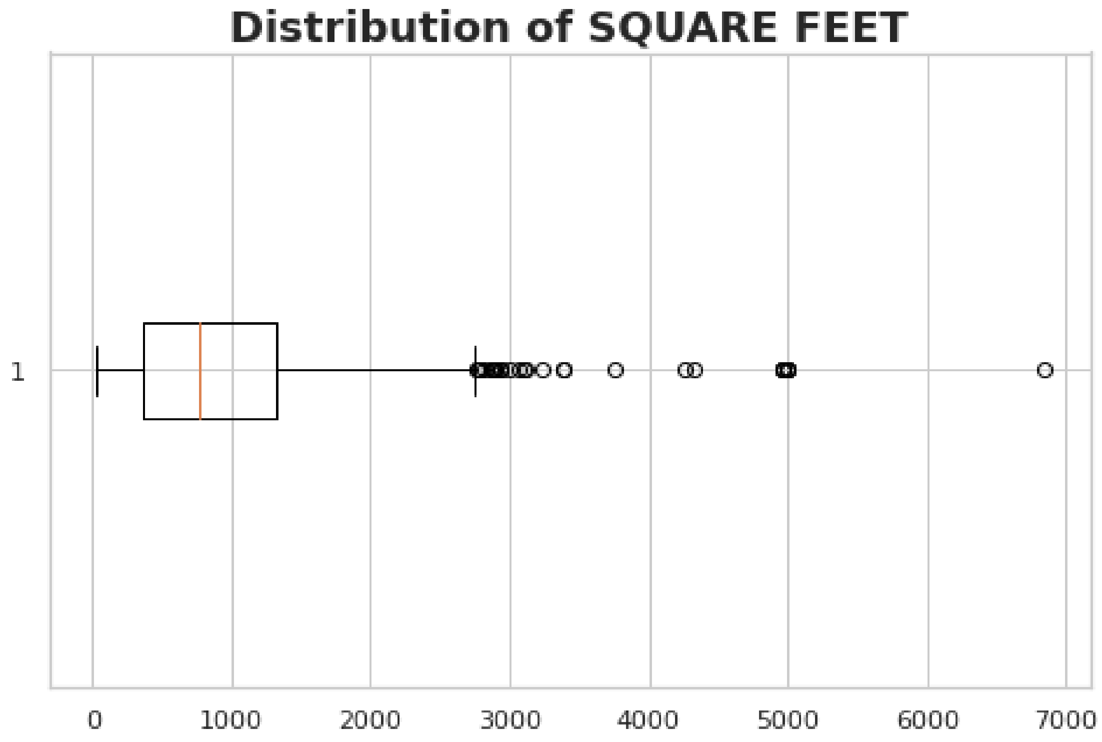
```
sns.set_style("whitegrid")
plt.figure(figsize = (8,5))
plt.title('Distribution of PRICE_bin', fontsize=18, fontweight='bold')
eda_percentage = df['PRICE_bin'].value_counts(normalize =
True).rename_axis('PRICE_bin').reset_index(name = 'Percentage')

ax = sns.barplot(x = 'PRICE_bin', y = 'Percentage', data =
eda_percentage, palette='Blues_d', order =
eda_percentage['PRICE_bin'].value_counts().index)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02),
ha='center', fontweight='bold')
    plt.setp(ax.get_xticklabels(), rotation=45);
```



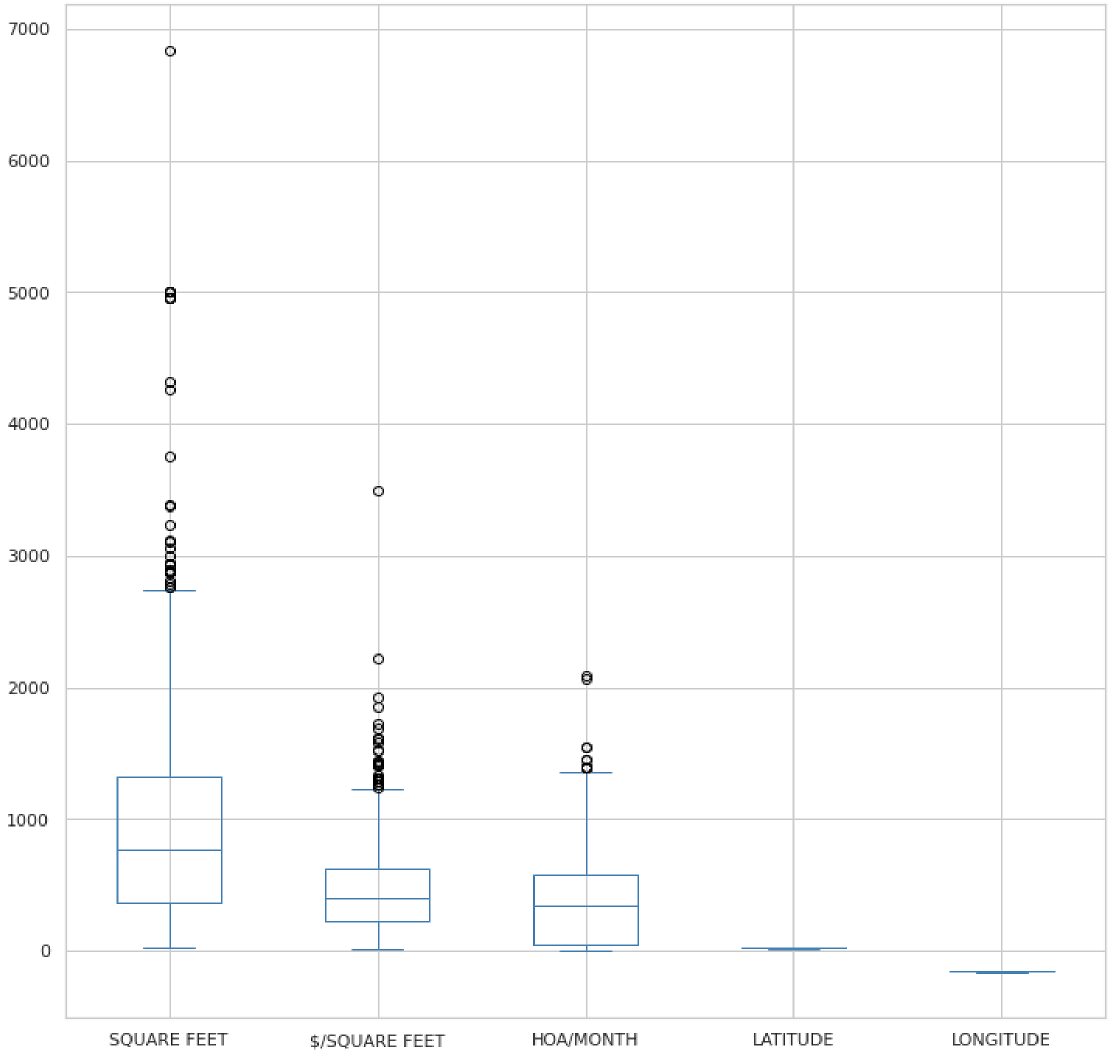
```
fig, ax = plt.subplots(figsize=(10, 5))
chart = sns.countplot(x="PRICE_bin", data=df, ax=ax,
palette='Blues_d',
                      order = df['PRICE_bin'].value_counts().index)
chart.set_xticklabels(chart.get_xticklabels(), rotation=0)
plt.show()
```



```
fig, ax = plt.subplots(1, 1)
# Creating plot
plt.boxplot(df['SQUARE FEET'],
            vert = 0,
           )
plt.title('Distribution of SQUARE FEET', fontsize=18,
fontweight='bold')
plt.show()
```



```
cts_variables2 = ['SQUARE FEET', '$/SQUARE FEET', 'HOA/MONTH',
'LATITUDE', 'LONGITUDE']

distribution = df[cts_variables2].boxplot(color = 'steelblue')
fig = plt.gcf()
fig.set_size_inches(12,12)
plt.show()
```



```
df.columns
```

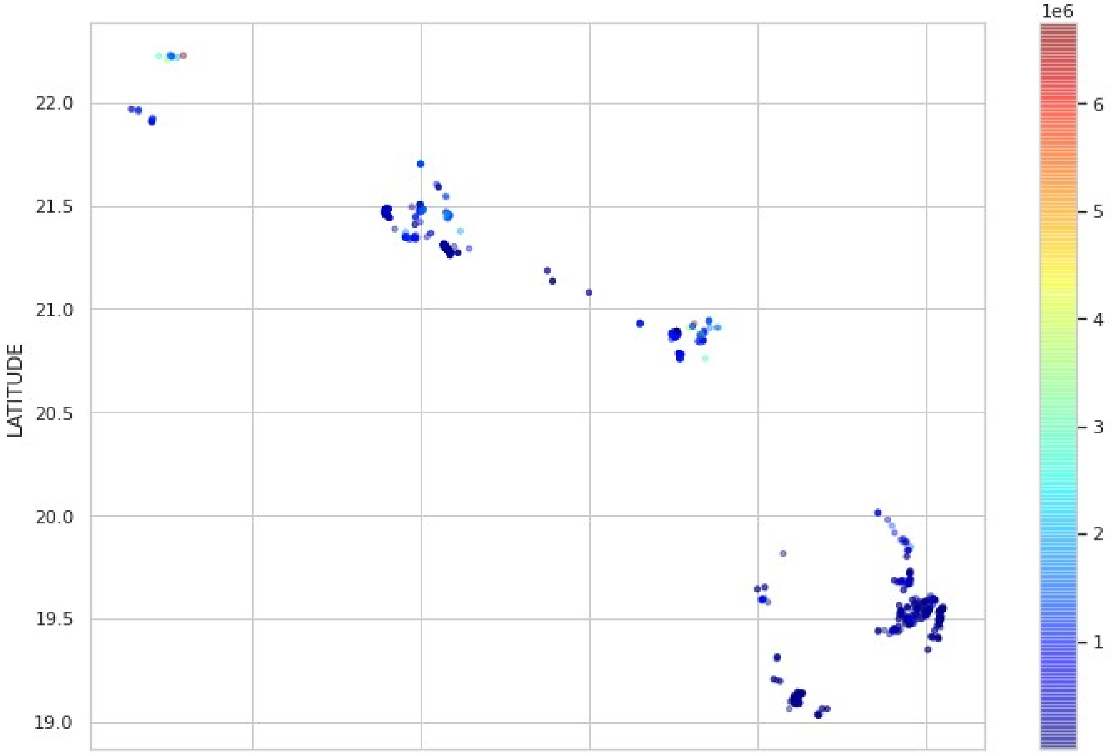
```
Index(['SALE TYPE', 'SOLD DATE', 'PROPERTY TYPE', 'ADDRESS', 'CITY',
       'STATE OR PROVINCE', 'ZIP OR POSTAL CODE', 'PRICE', 'BEDS',
'BATHS',
       'LOCATION', 'SQUARE FEET', 'LOT SIZE', 'YEAR BUILT', 'DAYS ON
MARKET',
       '$/SQUARE FEET', 'HOA/MONTH', 'STATUS', 'NEXT OPEN HOUSE START
TIME',
       'NEXT OPEN HOUSE END TIME',
       'URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)',
       'SOURCE', 'MLS#', 'FAVORITE', 'INTERESTED', 'LATITUDE',
'LONGITUDE',
       'bin_cut_manual', 'PRICE_bin'],
      dtype='object')
```

What is the most expensive location in Hawaii?

```
df_temp =
df.groupby(['CITY']).PRICE.mean().sort_values(ascending=False)[:10]
df_temp.head()
```

```
CITY
Kilauea   4425000.00000
Hanalei   3175000.00000
Paia      2804000.00000
Kula      2800000.00000
Kailua    1980000.00000
Name: PRICE, dtype: float64
```

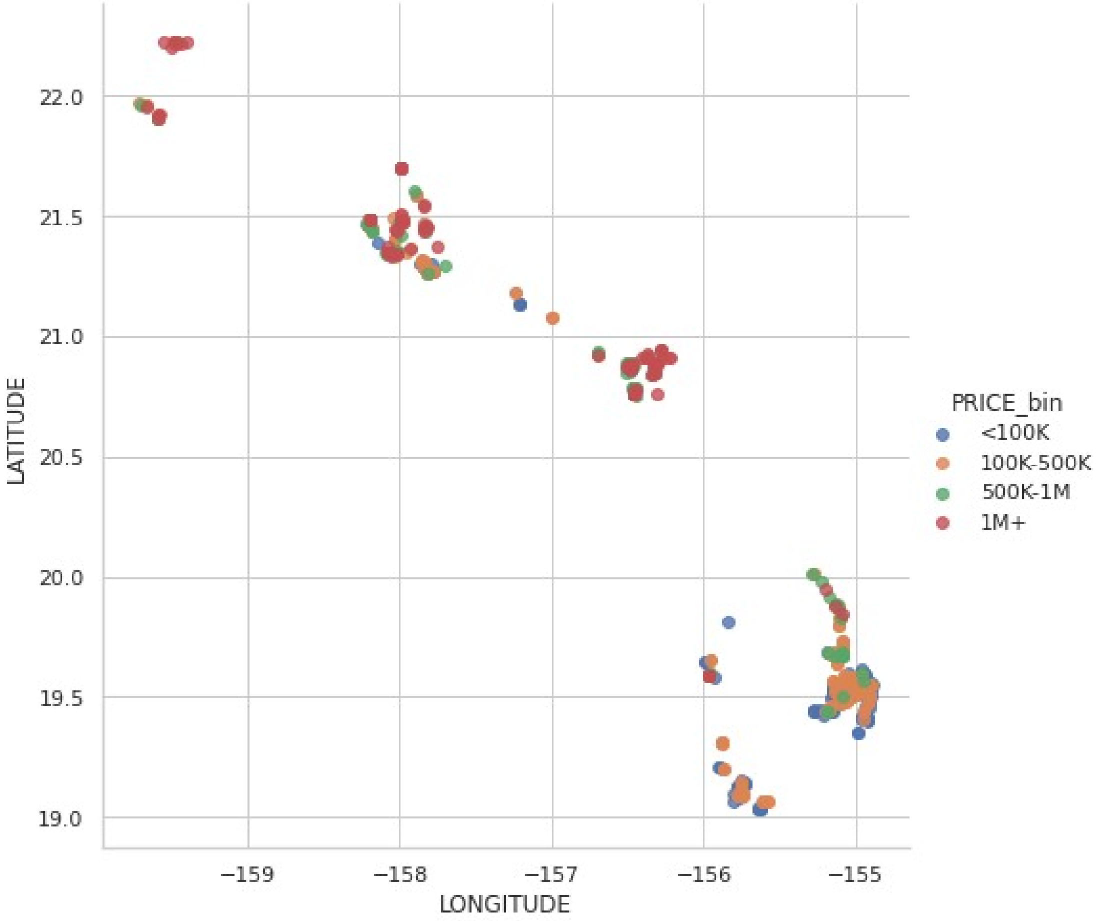
```
df.plot(kind="scatter", x='LONGITUDE', y='LATITUDE', alpha=0.4,
c=df['PRICE'], s=10,
              cmap=plt.get_cmap('jet'), figsize=(12,8));
```



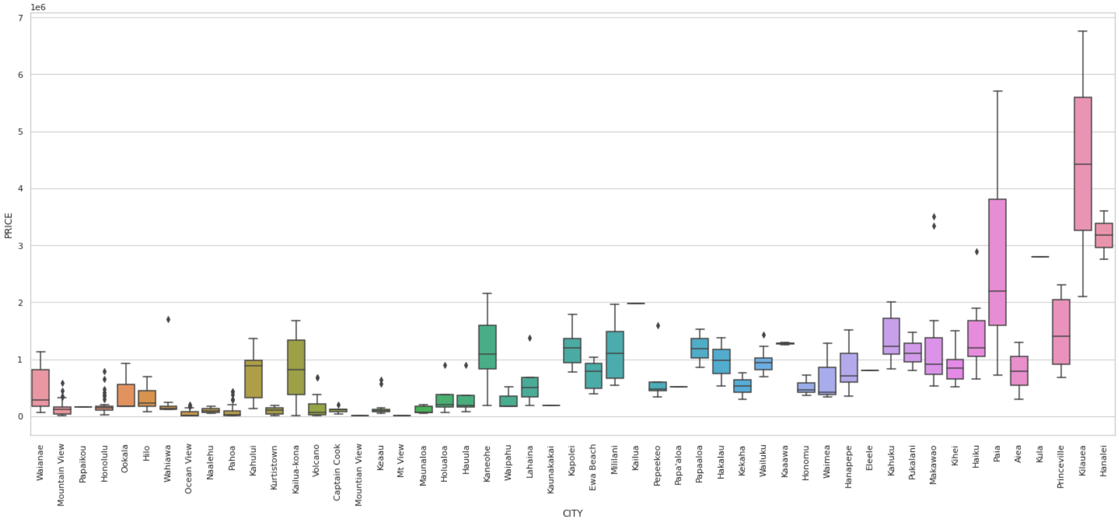
```
sns.lmplot(
    "LONGITUDE", "LATITUDE", data=df, hue="PRICE_bin", fit_reg=False,
size=7
);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:581:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```

```
<Figure size 1728x288 with 0 Axes>
```



```
fig, ax = plt.subplots(figsize=(25, 10))
sns.boxplot(x='CITY', y='PRICE', data=df, ax=ax)
plt.setp(ax.get_xticklabels(), rotation=90);
```



1e6 = 1*10^6

```
df_new =
df.groupby(['CITY']).PRICE.mean().sort_values(ascending=False).reset_i
ndex()
df_new.head(10)
```

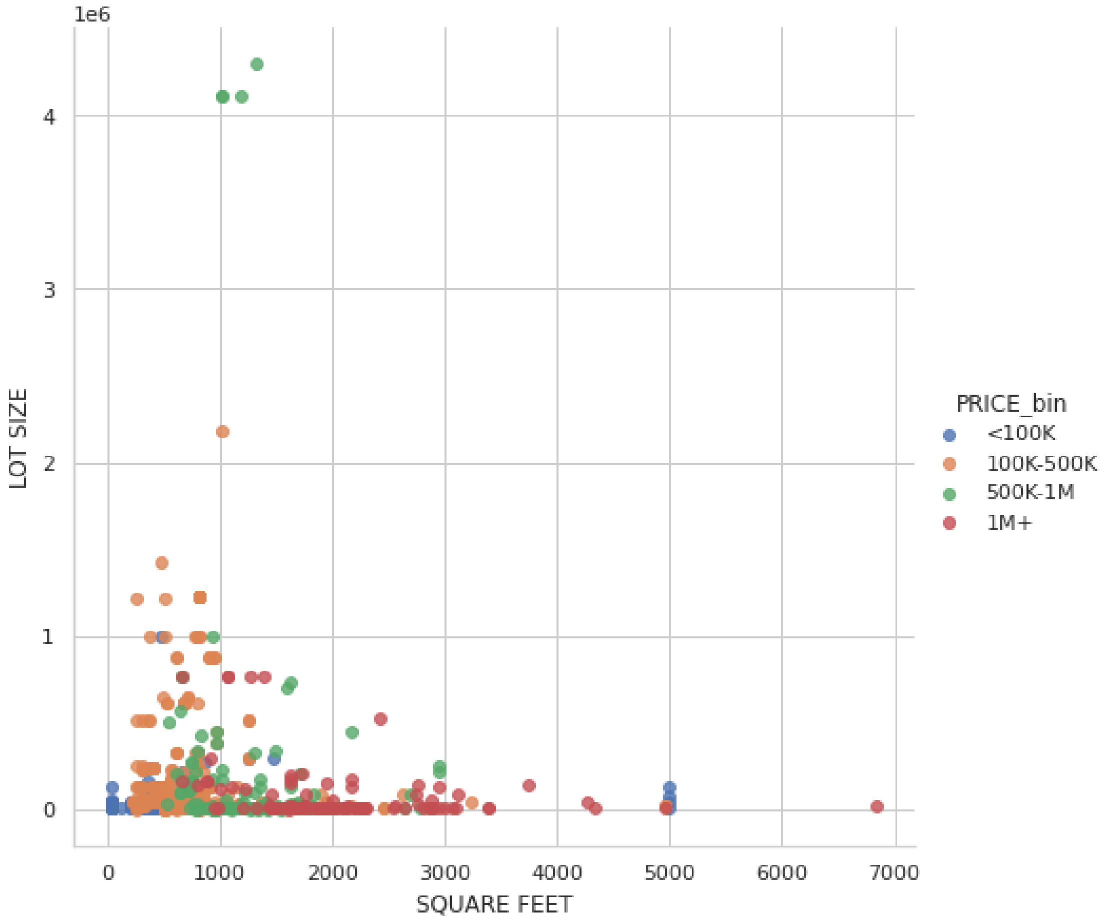
```
          CITY         PRICE
0      Kilauea 4425000.00000
1      Hanalei 3175000.00000
2         Paia 2804000.00000
3         Kula 2800000.00000
4       Kailua 1980000.00000
5  Princeville 1463583.33333
6        Haiku 1427090.81818
7       Kahuku 1379057.00000
8      Makawao 1338416.66667
9       Kaaawa 1277500.00000
```

```
df.columns
```

```
Index(['SALE TYPE', 'SOLD DATE', 'PROPERTY TYPE', 'ADDRESS', 'CITY',
       'STATE OR PROVINCE', 'ZIP OR POSTAL CODE', 'PRICE', 'BEDS',
'BATHS',
       'LOCATION', 'SQUARE FEET', 'LOT SIZE', 'YEAR BUILT', 'DAYS ON
MARKET',
       '$/SQUARE FEET', 'HOA/MONTH', 'STATUS', 'NEXT OPEN HOUSE START
TIME',
       'NEXT OPEN HOUSE END TIME',
       'URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)',
       'SOURCE', 'MLS#', 'FAVORITE', 'INTERESTED', 'LATITUDE',
'LONGITUDE',
       'bin_cut_manual', 'PRICE_bin'],
      dtype='object')
```

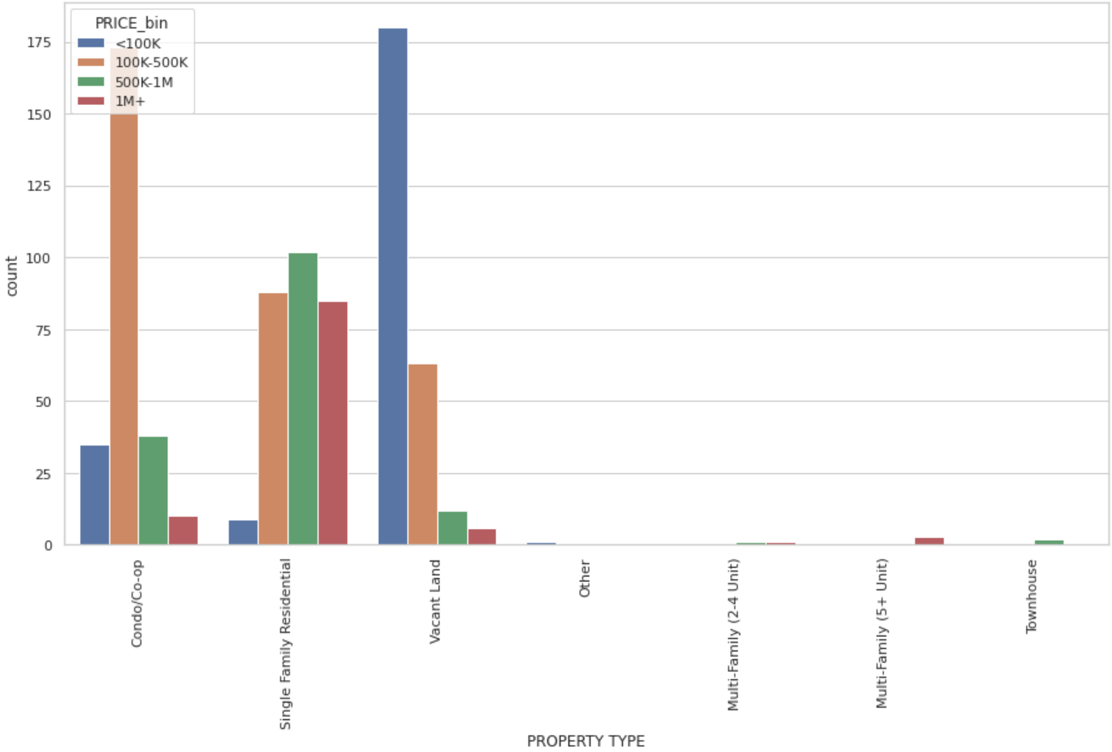
```
sns.lmplot(
    'SQUARE FEET', 'LOT SIZE', data=df, hue="PRICE_bin",
fit_reg=False, size=7
);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:581:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```

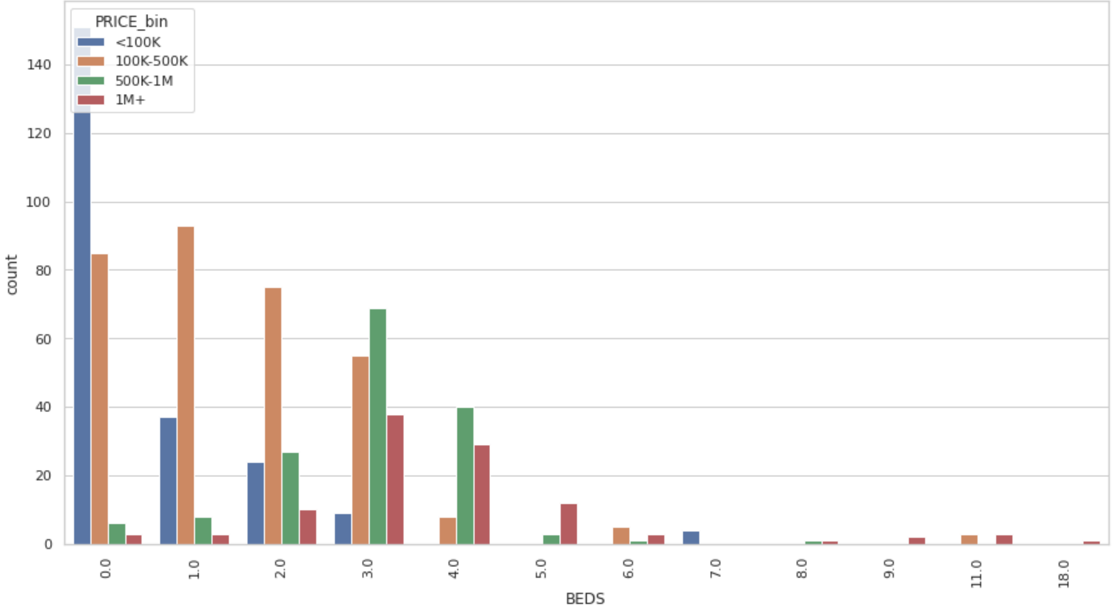


LOT size indicates the size of the piece of land where the property is situated.

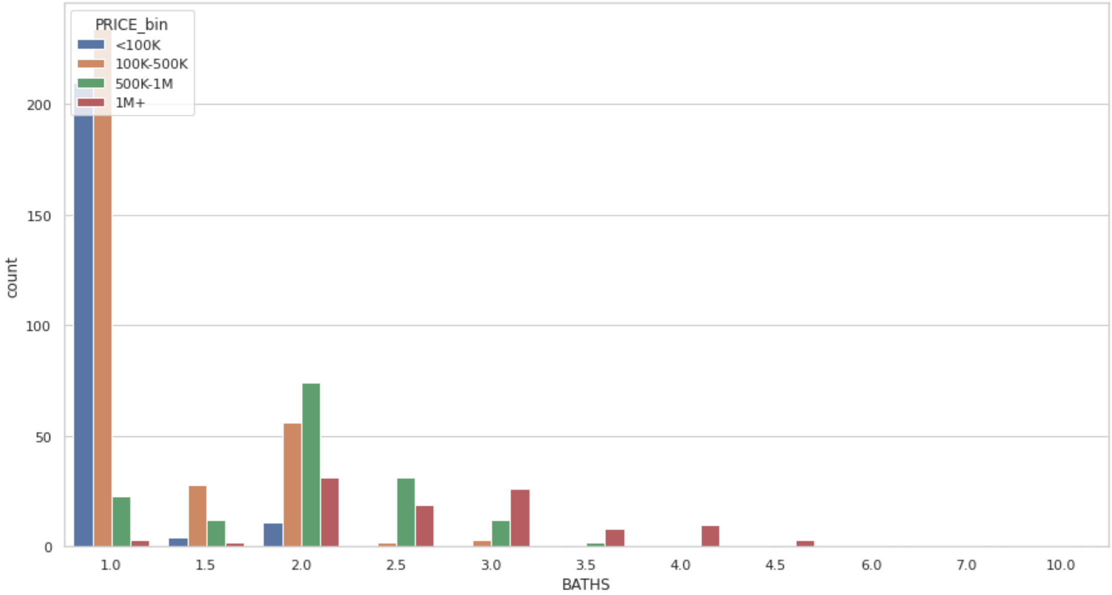
```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="PROPERTY TYPE", hue="PRICE_bin", data=df)
plt.setp(ax.get_xticklabels(), rotation=90);
```



```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="BEDS", hue="PRICE_bin", data=df)
plt.setp(ax.get_xticklabels(), rotation=90);
```

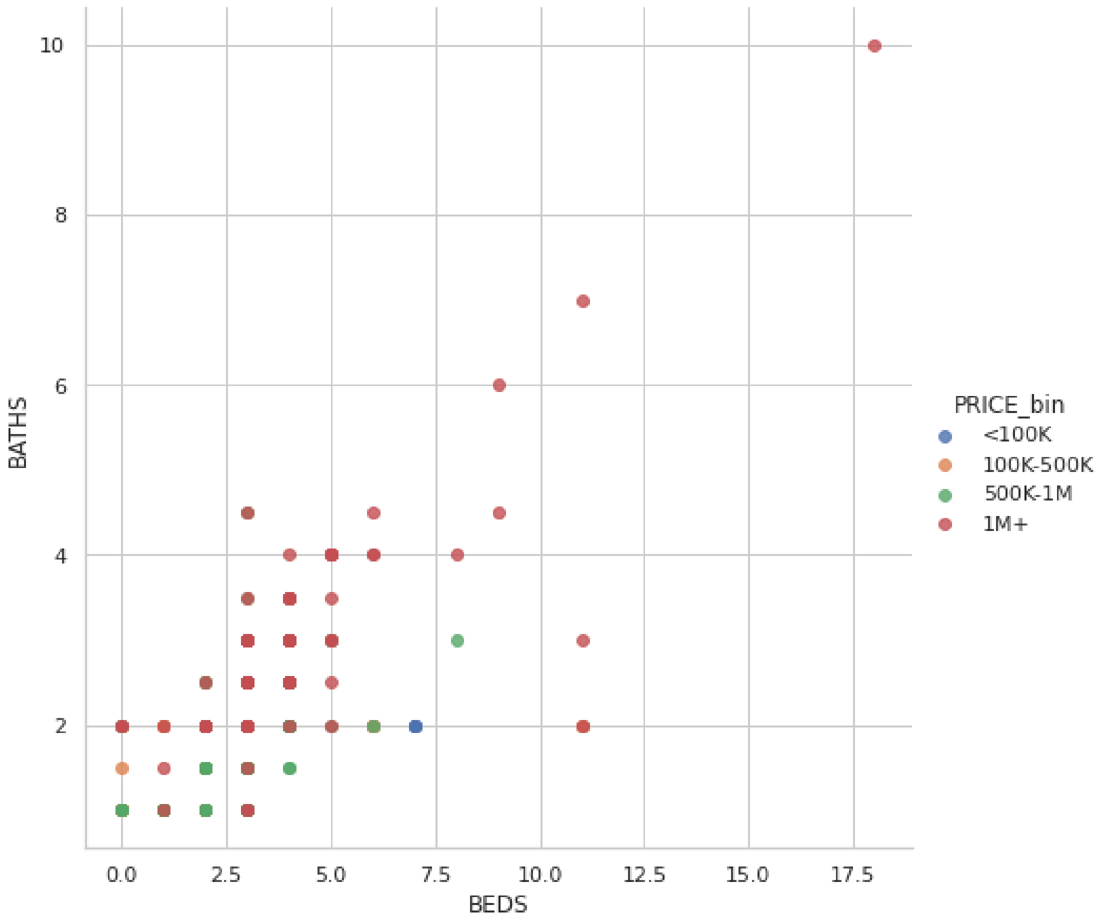


```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="BATHS", hue="PRICE_bin", data=df)
plt.setp(ax.get_xticklabels(), rotation=0);
```

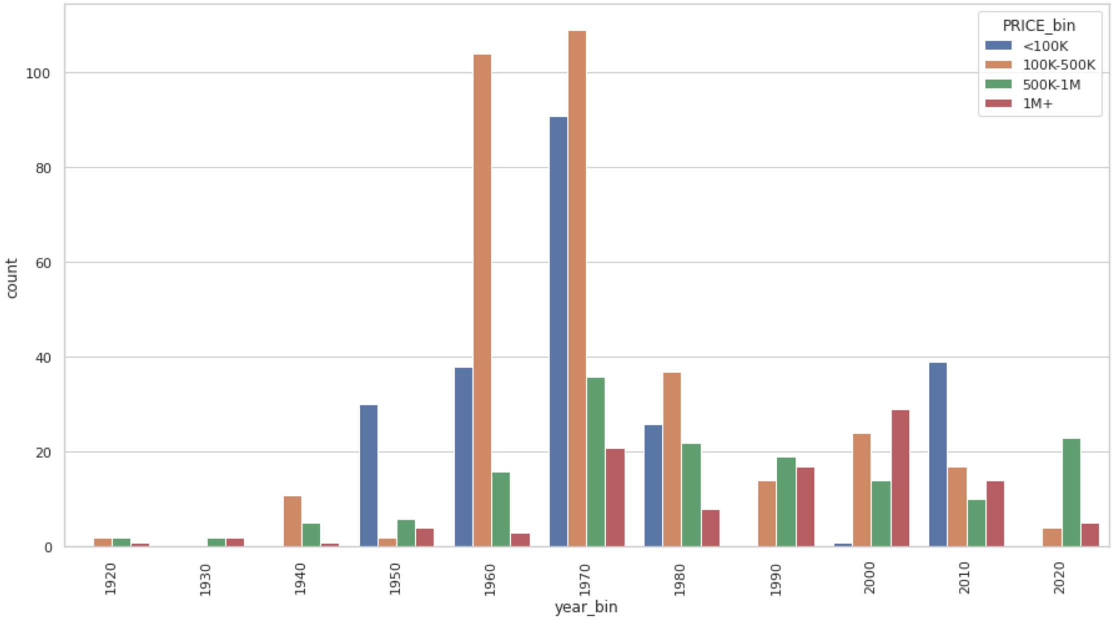


```
sns.lmplot(
    'BEDS', 'BATHS', data=df, hue="PRICE_bin", fit_reg=False, height=7
);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```



```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="year_bin", hue="PRICE_bin", data=df)
plt.setp(ax.get_xticklabels(), rotation=90);
```



```
df.groupby(['year_bin']).PRICE.mean().sort_values(ascending=False).res
et_index()
```

```
   year_bin         PRICE
0      1930  1.105000e+06
1      1990  9.357390e+05
2      2000  8.754926e+05
3      2020  8.399239e+05
4      1920  6.587110e+05
5      1980  4.640043e+05
6      1940  4.267059e+05
7      2010  4.048218e+05
8      1970  3.479158e+05
9      1950  3.428571e+05
10     1960  2.475584e+05
```

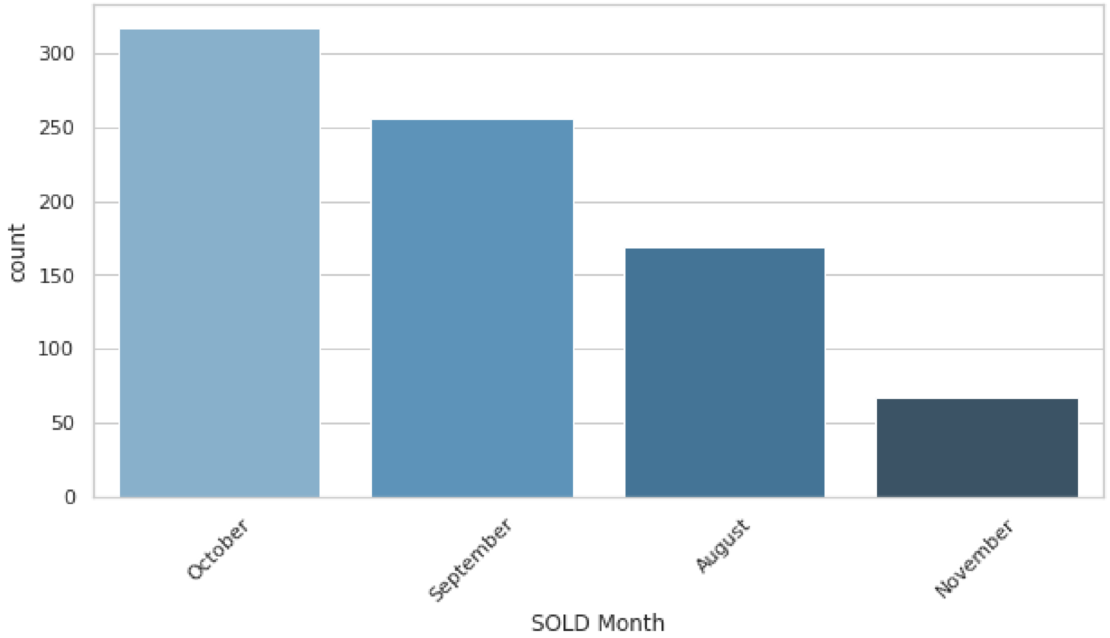
```
df.groupby(['YEAR
BUILT']).PRICE.mean().sort_values(ascending=False).reset_index().head(
10)
```

```
   YEAR BUILT          PRICE
0        1996 2759333.33333
1        1955 2200000.00000
2        2004 1890000.00000
3        2019 1504750.00000
4        1965 1371666.66667
5        1929 1260000.00000
6        1999 1247541.66667
7        1949 1247500.00000
8        2003 1244850.00000
9        2018 1239000.00000
```

```
df['SOLD Month'] = df['SOLD DATE'].str.split('-').str[0]
df['SOLD Month']
```

```
0      September
1         August
2      September
3      September
4         August
         ...
804    September
805      October
806      October
807    September
808    September
Name: SOLD Month, Length: 809, dtype: object
```

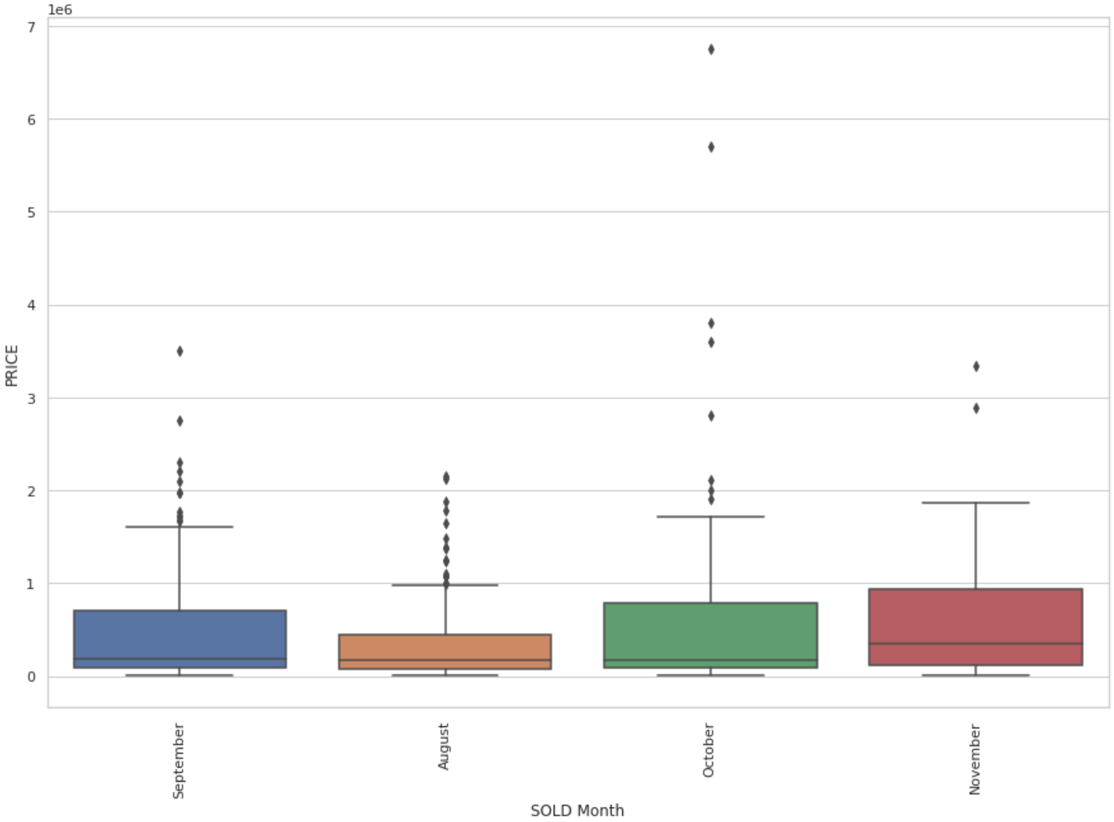
```
fig, ax = plt.subplots(figsize=(10, 5))
chart = sns.countplot(x='SOLD Month', data=df, ax=ax,
palette='Blues_d',
                      order = df['SOLD Month'].value_counts().index)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
plt.show()
```



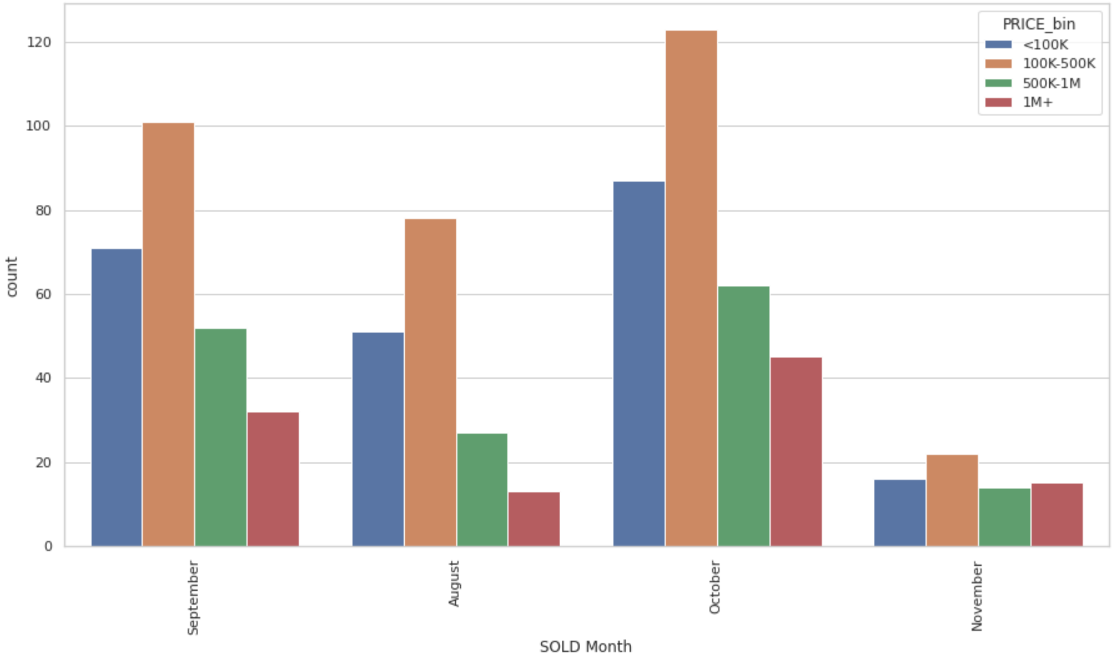
```
df.groupby(['SOLD
Month']).PRICE.mean().sort_values(ascending=False).reset_index()
```

```
  SOLD Month        PRICE
0   November 608129.85075
1    October 476900.89590
2  September 454972.25000
3     August 348992.92308
```

```
fig, ax = plt.subplots(figsize=(15, 10))
sns.boxplot(x='SOLD Month', y='PRICE', data=df, ax=ax)
plt.setp(ax.get_xticklabels(), rotation=90);
```



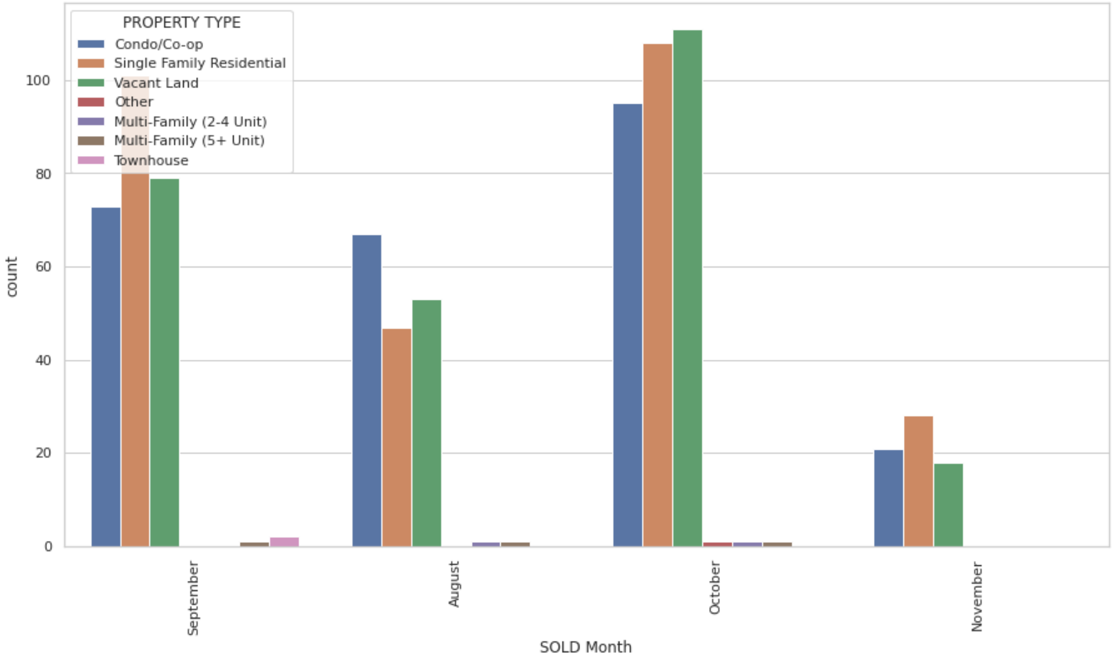
```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="SOLD Month", hue="PRICE_bin", data=df)
plt.setp(ax.get_xticklabels(), rotation=90);
```



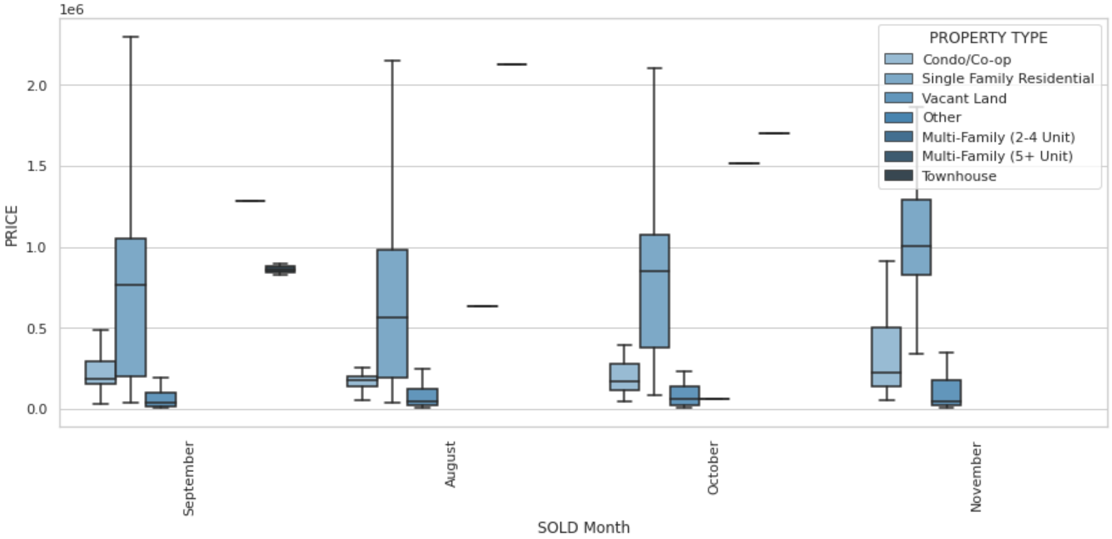
```
df.columns
```

```
Index(['SOLD DATE', 'PROPERTY TYPE', 'ADDRESS', 'CITY', 'ZIP OR POSTAL
CODE',
       'PRICE', 'BEDS', 'BATHS', 'LOCATION', 'SQUARE FEET', 'LOT
SIZE',
       'YEAR BUILT', '$/SQUARE FEET', 'HOA/MONTH',
       'URL (SEE https://www.redfin.com/buy-a-home/comparative-market-
analysis FOR INFO ON PRICING)',
       'SOURCE', 'MLS#', 'LATITUDE', 'LONGITUDE', 'bin_cut_manual',
       'PRICE_bin', 'year_bin', 'SOLD Month'],
      dtype='object')
```

```
fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(x="SOLD Month", hue="PROPERTY TYPE", data=df)
plt.setp(ax.get_xticklabels(), rotation=90);
```

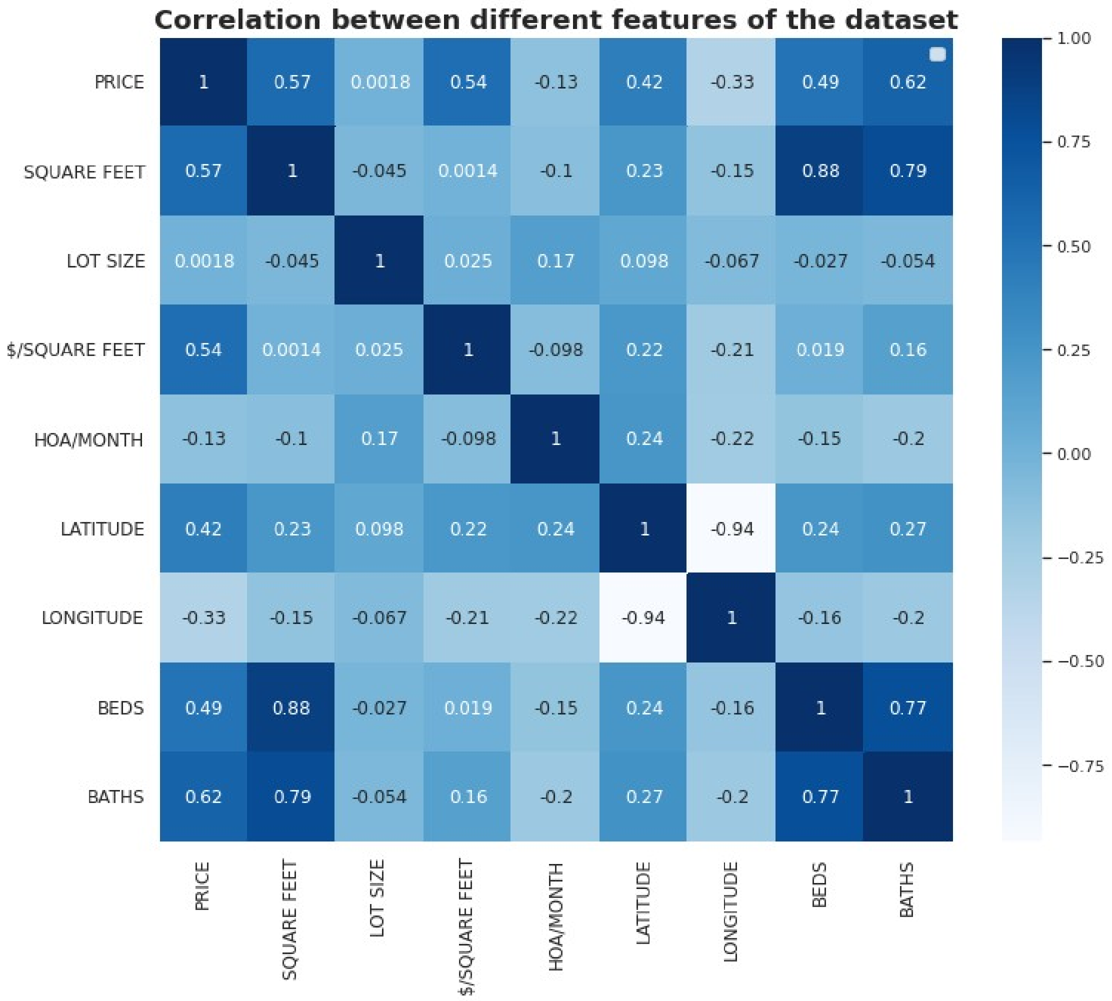


```
def boxplot_variation(feature1, feature2, feature3, width=16):
    fig, ax1 = plt.subplots(ncols=1, figsize=(width,6))
    s = sns.boxplot(ax = ax1, x=feature1, y=feature2, hue=feature3,
                data=df, palette="Blues_d",showfliers=False)
    s.set_xticklabels(s.get_xticklabels(),rotation=90)
    plt.show();

boxplot_variation('SOLD Month','PRICE', 'PROPERTY TYPE',15)
```



```
plt.figure(figsize = (12,10))
plt.title("Correlation between different features of the dataset",
fontsize = 18, fontweight = 'bold')
sns.heatmap(df[['PRICE', 'SQUARE FEET', 'LOT SIZE', '$/SQUARE FEET',
'HOA/MONTH',
       'LATITUDE', 'LONGITUDE', 'BEDS', 'BATHS']].corr(), cmap =
'Blues', annot = True)
plt.xticks(fontsize=12, rotation = 90)
plt.yticks(fontsize=12, rotation = 0)
plt.legend(fontsize=12)
plt.show()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in
legend.
```



Correlation between different features of the dataset