```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

df = pd.read_csv("/content/MagicBricks.csv")

df.head()
```

```
    Area  BHK  Bathroom      Furnishing  \
0  800.0    3       2.0  Semi-Furnished
1  750.0    2       2.0  Semi-Furnished
2  950.0    2       2.0       Furnished
3  600.0    2       2.0  Semi-Furnished
4  650.0    2       2.0  Semi-Furnished

                                            Locality  Parking
Price  \
0                                  Rohini Sector 25      1.0
6500000
1             J R Designers Floors, Rohini Sector 24      1.0
5000000
2                Citizen Apartment, Rohini Sector 13      1.0
15500000
3                                  Rohini Sector 24      1.0
4200000
4  Rohini Sector 24 carpet area 650 sqft status R...      1.0
6200000

          Status   Transaction           Type  Per_Sqft
0  Ready_to_move  New_Property  Builder_Floor       NaN
1  Ready_to_move  New_Property      Apartment    6667.0
2  Ready_to_move        Resale      Apartment    6667.0
3  Ready_to_move        Resale  Builder_Floor    6667.0
4  Ready_to_move  New_Property  Builder_Floor    6667.0
```

```
df.shape
```

```
(1259, 11)
```

```
df.dtypes
```

```
Area           float64
BHK              int64
Bathroom       float64
Furnishing      object
Locality        object
Parking        float64
Price            int64
Status          object
```

```
Transaction     object
Type            object
Per_Sqft       float64
dtype: object
```

```
df.columns
```

```
Index(['Area', 'BHK', 'Bathroom', 'Furnishing', 'Locality', 'Parking',
'Price',
       'Status', 'Transaction', 'Type', 'Per_Sqft'],
      dtype='object')
```

```
df.duplicated().sum()  # Duplicated Values
```

```
83
```

```
df[df.duplicated()] # duplicated rows
```

```
        Area  BHK  Bathroom      Furnishing  \
84    1540.0    3       3.0  Semi-Furnished
92    1450.0    3       3.0  Semi-Furnished
110   1000.0    2       2.0       Furnished
120   1500.0    1       NaN     Unfurnished
122   1710.0    3       3.0  Semi-Furnished
...      ...  ...       ...             ...
1164  1300.0    2       2.0  Semi-Furnished
1165  1200.0    2       2.0     Unfurnished
1166  1300.0    3       2.0     Unfurnished
1167  1400.0    3       3.0       Furnished
1168  1400.0    3       3.0  Semi-Furnished

                                                Locality  Parking
Price  \
84                Nav Kairali Apartment, Dwarka Sector 3      1.0
14500000
92                                        Lajpat Nagar 3      2.0
30000000
110                                       Lajpat Nagar 3      1.0
20000000
120                                       Lajpat Nagar 2      NaN
13500000
122                                       Lajpat Nagar 2      4.0
26000000
...                                                 ...      ...
...
1164                        Yamuna Apartment, Alaknanda      1.0
15000000
1165                      Nilgiri Apartment, Alaknanda      1.0
14300000
1166                      Nilgiri Apartment, Alaknanda      1.0
18500000
```

```
1167  Tara Apartment, Alaknanda carpet area 1400 sqf...      1.0
19000000
1168                                            Alaknanda      1.0
19000000

             Status   Transaction           Type  Per_Sqft
84    Ready_to_move        Resale      Apartment    3524.0
92    Ready_to_move  New_Property  Builder_Floor       NaN
110   Ready_to_move        Resale      Apartment   20000.0
120   Ready_to_move        Resale      Apartment       NaN
122   Ready_to_move  New_Property  Builder_Floor       NaN
...             ...           ...             ...       ...
1164  Ready_to_move        Resale      Apartment   11538.0
1165  Ready_to_move        Resale      Apartment   11538.0
1166  Ready_to_move        Resale      Apartment   11538.0
1167  Ready_to_move        Resale      Apartment   11538.0
1168  Ready_to_move        Resale      Apartment   11538.0

[83 rows x 11 columns]
```

```
df= df.drop_duplicates() # Duplicates are removed
```

```
df.shape
```

```
(1176, 11)
```

```
df.nunique()
```

```
Area           315
BHK              8
Bathroom         7
Furnishing       3
Locality       365
Parking          9
Price          284
Status           2
Transaction      2
Type             2
Per_Sqft       251
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1176 entries, 0 to 1258
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1176 non-null   float64
 1   BHK          1176 non-null   int64
 2   Bathroom     1175 non-null   float64
```

```
 3   Furnishing   1171 non-null   object
 4   Locality     1176 non-null   object
 5   Parking      1145 non-null   float64
 6   Price        1176 non-null   int64
 7   Status       1176 non-null   object
 8   Transaction  1176 non-null   object
 9   Type         1171 non-null   object
 10  Per_Sqft     949 non-null    float64
dtypes: float64(4), int64(2), object(5)
memory usage: 110.2+ KB
```

```
df.isnull().sum() # Missing Values
```

```
Area             0
BHK              0
Bathroom         1
Furnishing       5
Locality         0
Parking         31
Price            0
Status           0
Transaction      0
Type             5
Per_Sqft       227
dtype: int64
```

```
df.dropna(axis=0,inplace=True) # Missing values are droped
```

```
df.isnull().sum() # Missing Values
```

```
Area           0
BHK            0
Bathroom       0
Furnishing     0
Locality       0
Parking        0
Price          0
Status         0
Transaction    0
Type           0
Per_Sqft       0
dtype: int64
```

```
print(df.shape)
```

```
(936, 11)
```

```
df["Furnishing"].unique()
```

```
array(['Semi-Furnished', 'Furnished', 'Unfurnished'], dtype=object)
```

```
df["Status"].unique()
```

```
array(['Ready_to_move', 'Almost_ready'], dtype=object)
```

```
df["Transaction"].unique()
```

```
array(['New_Property', 'Resale'], dtype=object)
```

```
df["Type"].unique()
```

```
array(['Apartment', 'Builder_Floor'], dtype=object)
```

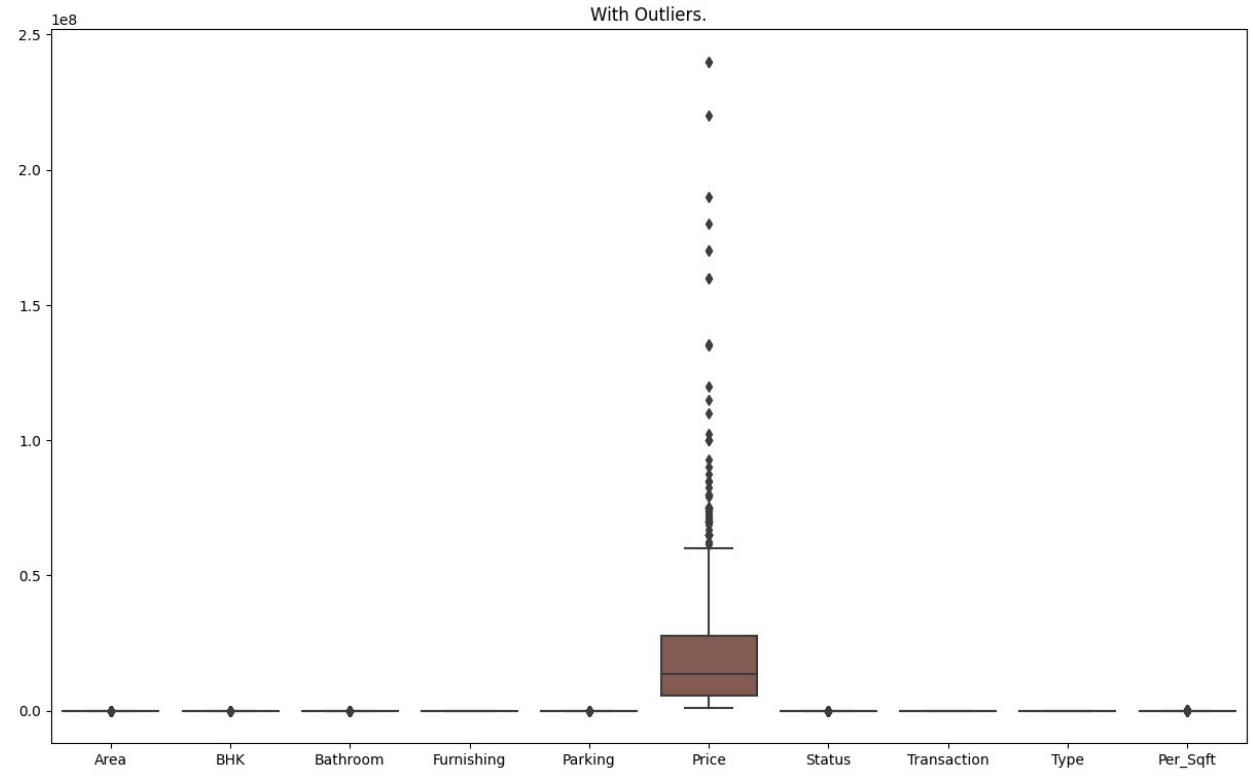
```
from sklearn.preprocessing import LabelEncoder
def label(data,columns):
  le = LabelEncoder()
  for column in columns:
    data[column] = le.fit_transform(data[column])
  return data

label(df,["Furnishing","Status","Transaction","Type"])
```

```
          Area  BHK  Bathroom  Furnishing  \
1       750.0    2       2.0           1
2       950.0    2       2.0           0
3       600.0    2       2.0           1
4       650.0    2       2.0           1
5      1300.0    4       3.0           1
...       ...  ...       ...         ...
1254   4118.0    4       5.0           2
1255   1050.0    3       2.0           1
1256    875.0    3       3.0           1
1257    990.0    2       2.0           2
1258  11050.0    3       3.0           2

                                              Locality  Parking
Price  \
1                J R Designers Floors, Rohini Sector 24      1.0
5000000
2                   Citizen Apartment, Rohini Sector 13      1.0
15500000
3                                    Rohini Sector 24      1.0
4200000
4     Rohini Sector 24 carpet area 650 sqft status R...      1.0
6200000
5                                    Rohini Sector 24      1.0
15500000
...                                                ...      ...
...
1254                                 Chittaranjan Park      3.0
55000000
1255                                 Chittaranjan Park      3.0
```

```
12500000
1256                                  Chittaranjan Park      3.0
17500000
1257                           Chittaranjan Park Block A      1.0
11500000
1258                                  Chittaranjan Park      1.0
18500000

      Status  Transaction  Type  Per_Sqft
1          1            0     0    6667.0
2          1            1     0    6667.0
3          1            1     1    6667.0
4          1            0     1    6667.0
5          1            0     1    6667.0
...      ...          ...   ...       ...
1254       1            0     1   12916.0
1255       1            1     1   12916.0
1256       1            0     1   12916.0
1257       1            1     1   12916.0
1258       1            0     1   12916.0

[936 rows x 11 columns]

df.tail()

         Area  BHK  Bathroom  Furnishing                    Locality
Parking  \
1254   4118.0    4       5.0           2          Chittaranjan Park
3.0
1255   1050.0    3       2.0           1          Chittaranjan Park
3.0
1256    875.0    3       3.0           1          Chittaranjan Park
3.0
1257    990.0    2       2.0           2  Chittaranjan Park Block A
1.0
1258  11050.0    3       3.0           2          Chittaranjan Park
1.0

         Price  Status  Transaction  Type  Per_Sqft
1254  55000000       1            0     1   12916.0
1255  12500000       1            1     1   12916.0
1256  17500000       1            0     1   12916.0
1257  11500000       1            1     1   12916.0
1258  18500000       1            0     1   12916.0

df["Furnishing"].unique()

array([1, 0, 2])

df["Type"].unique()
```

```
array([0, 1])
```

```
# To see Outliers..
plt.figure(figsize=(15,9))
plt.title("With Outliers.")
sns.boxplot(df)
plt.show()
```




```
import warnings
warnings.filterwarnings("ignore")
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
IQR = q3-q1
lower = q1 - 1.5*(IQR)
higher = q3 + 1.5*(IQR)
cleaned_data = df[~((df < lower) | (df > higher)).any(axis=1)]
cleaned_data.head()
```

```
     Area  BHK  Bathroom  Furnishing  \
1   750.0    2       2.0           1
2   950.0    2       2.0           0
3   600.0    2       2.0           1
4   650.0    2       2.0           1
5  1300.0    4       3.0           1

                                          Locality  Parking
```

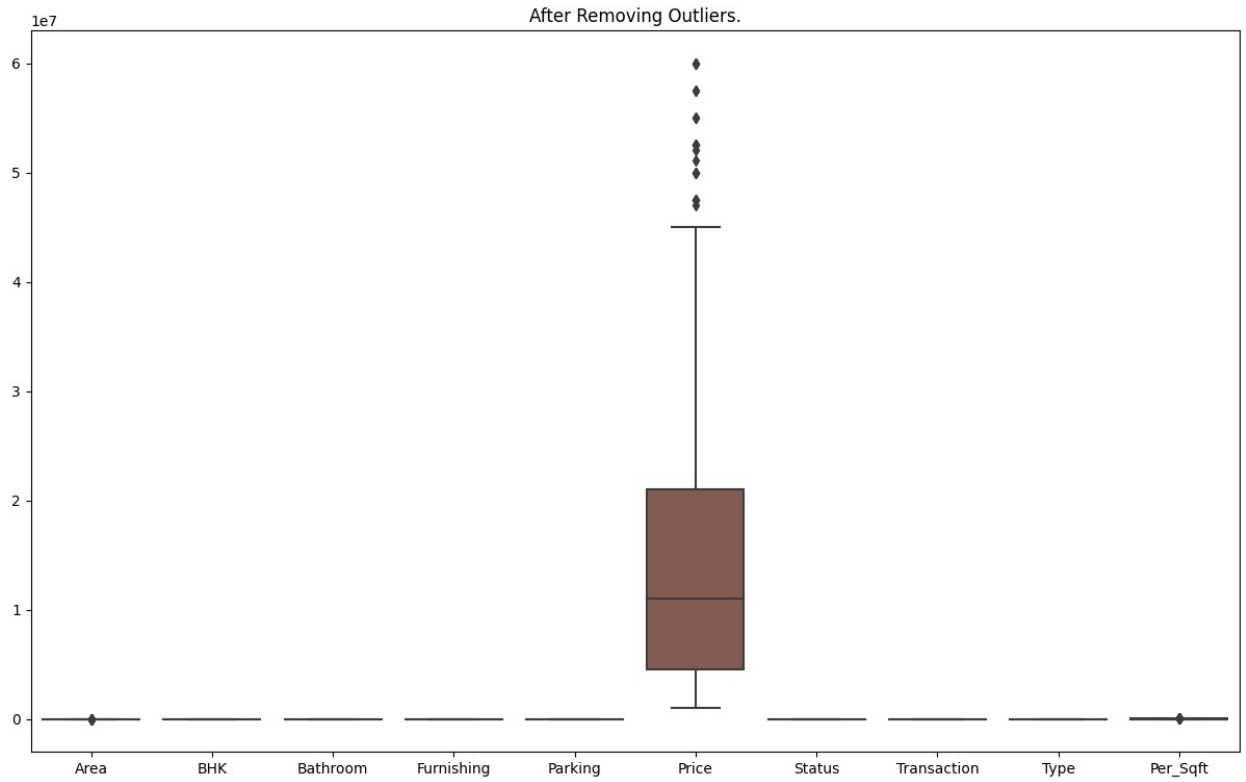
```
Price  \
1             J R Designers Floors, Rohini Sector 24      1.0
5000000
2                Citizen Apartment, Rohini Sector 13      1.0
15500000
3                                     Rohini Sector 24      1.0
4200000
4  Rohini Sector 24 carpet area 650 sqft status R...      1.0
6200000
5                                     Rohini Sector 24      1.0
15500000

   Status  Transaction  Type  Per_Sqft
1       1            0     0    6667.0
2       1            1     0    6667.0
3       1            1     1    6667.0
4       1            0     1    6667.0
5       1            0     1    6667.0
```

```
df1 = cleaned_data
df1.shape # After removing Outliers the shape is this...
```

```
(733, 11)
```

```
plt.figure(figsize=(15,9))
plt.title("After Removing Outliers.")
sns.boxplot(df1)
```

```
<Axes: title={'center': 'After Removing Outliers.'}>
```




```
df.head()
```

```
     Area  BHK  Bathroom  Furnishing  \
1   750.0    2       2.0           1
2   950.0    2       2.0           0
3   600.0    2       2.0           1
4   650.0    2       2.0           1
5  1300.0    4       3.0           1

                                            Locality  Parking
Price  \
1            J R Designers Floors, Rohini Sector 24      1.0
5000000
2               Citizen Apartment, Rohini Sector 13      1.0
15500000
3                                   Rohini Sector 24      1.0
4200000
4  Rohini Sector 24 carpet area 650 sqft status R...      1.0
6200000
5                                   Rohini Sector 24      1.0
15500000

   Status  Transaction  Type  Per_Sqft
1       1            0     0    6667.0
2       1            1     0    6667.0
3       1            1     1    6667.0
```

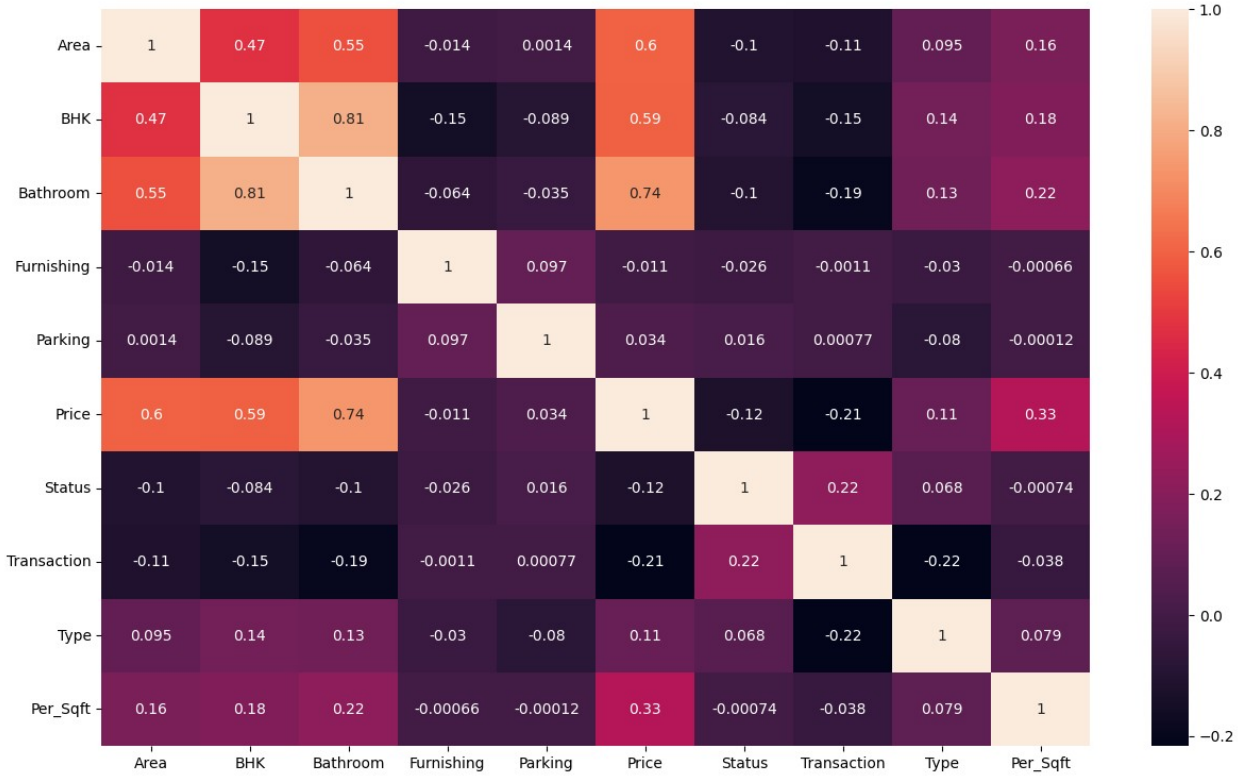
```
4       1            0     1    6667.0
5       1            0     1    6667.0
```

```
df.describe()
```

```
                Area          BHK     Bathroom  Furnishing     Parking  \
count    936.000000  936.000000  936.000000  936.000000  936.000000
mean    1481.921451    2.780983    2.572650    1.147436    1.733974
std     1639.227844    0.966886    1.099859    0.640566    3.334275
min       28.000000    1.000000    1.000000    0.000000    1.000000
25%      750.000000    2.000000    2.000000    1.000000    1.000000
50%     1136.500000    3.000000    2.000000    1.000000    1.000000
75%     1700.000000    3.000000    3.000000    2.000000    2.000000
max    24300.000000    7.000000    7.000000    2.000000   39.000000

               Price      Status  Transaction        Type
Per_Sqft
count  9.360000e+02  936.000000   936.000000  936.000000
936.000000
mean   2.203208e+07    0.945513     0.611111    0.571581
15790.613248
std    2.727633e+07    0.227098     0.487759    0.495114
21807.249640
min    1.000000e+06    0.000000     0.000000    0.000000
1259.000000
25%    5.400000e+06    1.000000     0.000000    0.000000
6444.250000
50%    1.375000e+07    1.000000     1.000000    1.000000
11111.000000
75%    2.750000e+07    1.000000     1.000000    1.000000
18000.000000
max    2.400000e+08    1.000000     1.000000    1.000000
183333.000000
```

```
plt.figure(figsize=(15,9))
corr = df.corr()
sns.heatmap(corr,annot=True)
```

```
<Axes: >
```

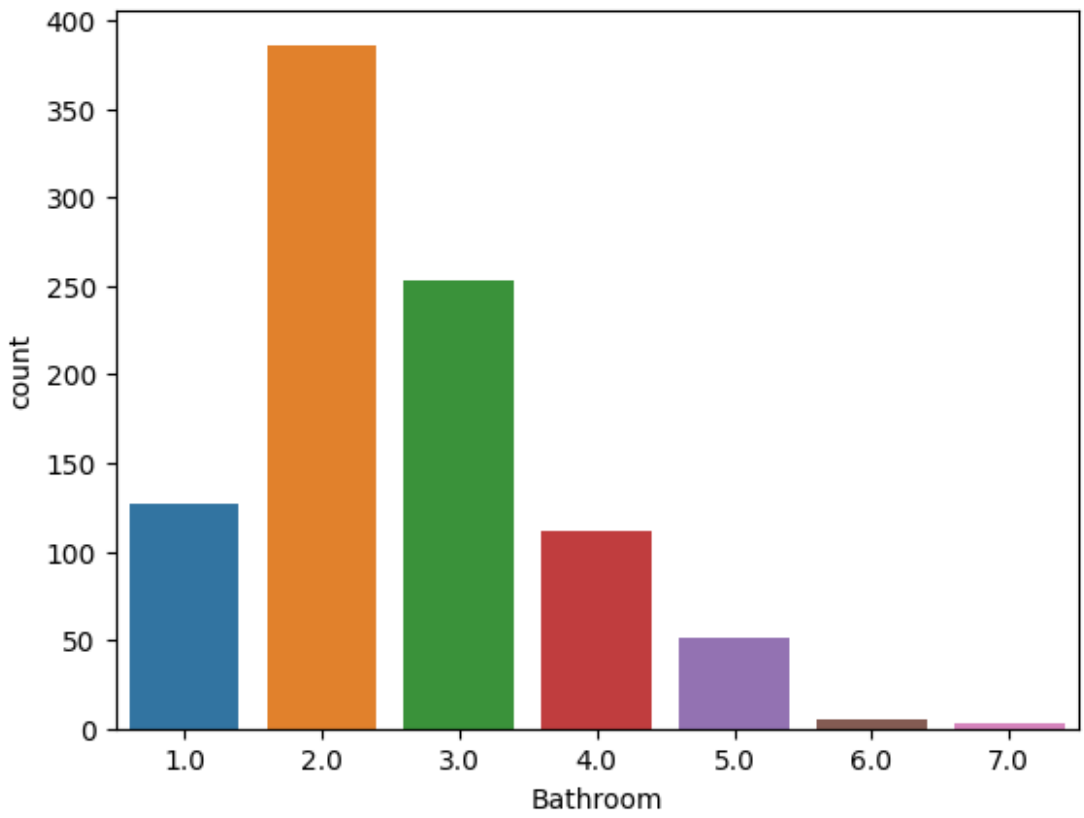



```
sns.pairplot(df)
```

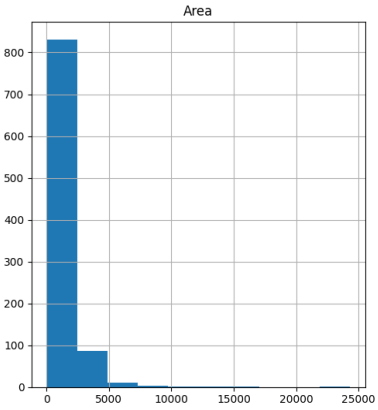
```
<seaborn.axisgrid.PairGrid at 0x7d4af78c2500>
```

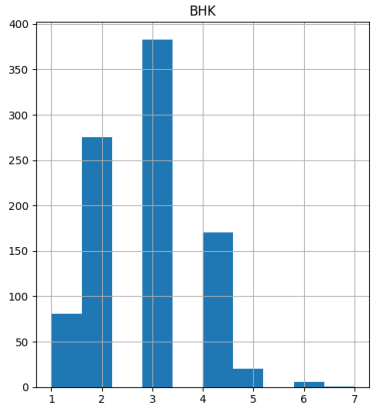

```
sns.countplot(data=df,x=df['Bathroom'])
```

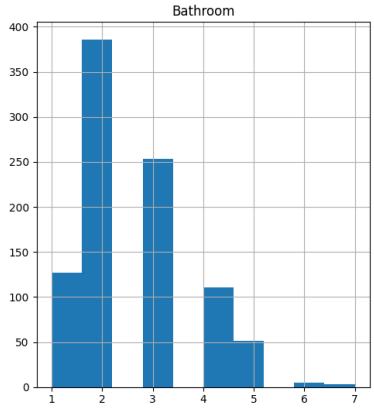
```
<Axes: xlabel='Bathroom', ylabel='count'>
```

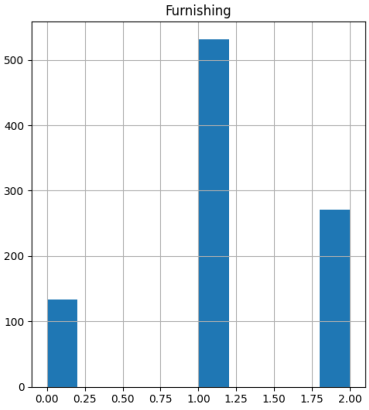


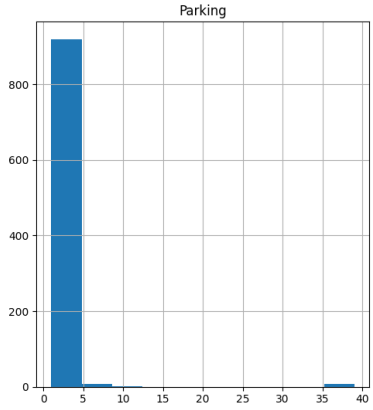

```
df.hist(figsize=(20,30))
plt.show()
```

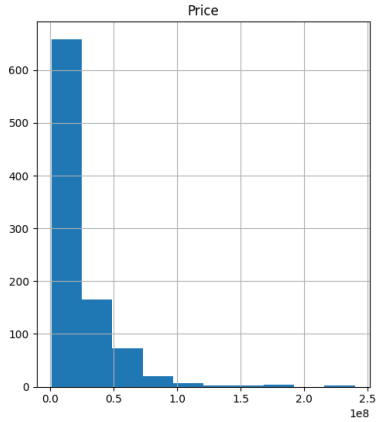

Area
0
5000
10000
15000
20000
25000

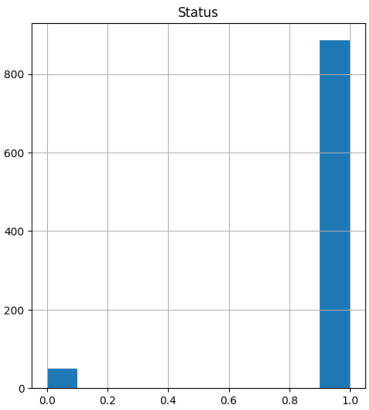

BHK
1
2
3
4
5
6
7

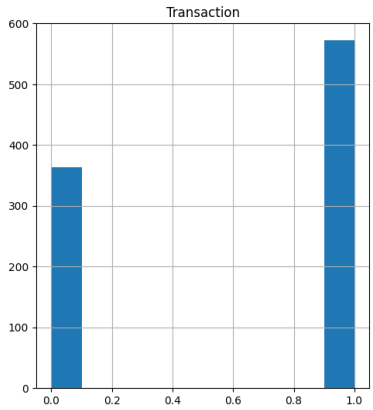

Bathroom
1
2
3
4
5
6
7

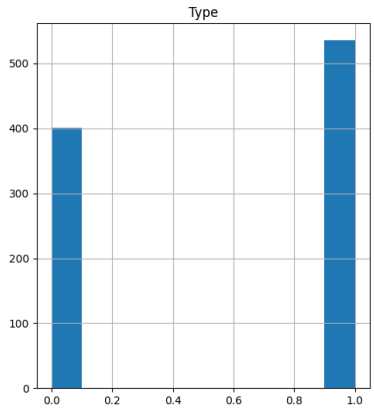

Furnishing
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

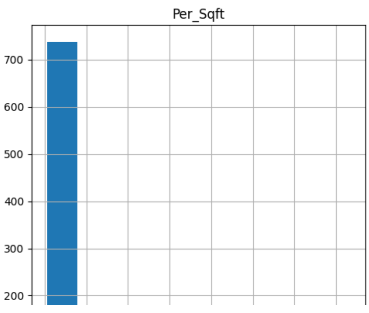

Parking
0
5
10
15
20
25
30
35
40


Price
0.0
0.5
1.0
1.5
2.0
2.5
1e8


Status
0.0
0.2
0.4
0.6
0.8
1.0


Transaction
0.0
0.2
0.4
0.6
0.8
1.0


Type
0.0
0.2
0.4
0.6
0.8
1.0


Per_Sqft

```
y = df["Price"]
y # dependent
```

```
1        5000000
2       15500000
3        4200000
4        6200000
5       15500000
          ...
1254    55000000
1255    12500000
1256    17500000
1257    11500000
1258    18500000
Name: Price, Length: 936, dtype: int64
```

```
df.drop("Price",axis=1,inplace=True)
df.drop("Locality",axis=1,inplace=True)
x = df
x.head() # independent
```

```
     Area  BHK  Bathroom  Furnishing  Parking  Status  Transaction
Type  \
1   750.0    2       2.0           1      1.0       1            0
0
2   950.0    2       2.0           0      1.0       1            1
0
3   600.0    2       2.0           1      1.0       1            1
1
4   650.0    2       2.0           1      1.0       1            0
1
5  1300.0    4       3.0           1      1.0       1            0
1

   Per_Sqft
1    6667.0
2    6667.0
3    6667.0
4    6667.0
5    6667.0
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.20,random_state=90)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(748, 9)
(188, 9)
(748,)
(188,)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
```

```
LinearRegression()
```

```
from sklearn.metrics import r2_score
y_pred = lr.predict(x_test)
print(r2_score(y_test,y_pred))
```

```
0.6236942622116364
```

**Linear Regression : 62.36%**

```
from sklearn.linear_model import Ridge
ridge = Ridge()
ridge.fit(x_train,y_train)
```

```
Ridge()
```

```
y_pred1 = ridge.predict(x_test)
print(r2_score(y_test,y_pred1))
```

```
0.6236598619206521
```

**Ridge : 62.36%**

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

```
y_pred = rfr.predict(x_test)
print(r2_score(y_test,y_pred))
```

```
0.8978425650438068
```

**RandomForestRegressor : 89.78%**

```
from sklearn.tree import DecisionTreeRegressor
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
```

```
DecisionTreeRegressor()
```

```
y_pred = dtc.predict(x_test)
print(r2_score(y_test,y_pred))
```

```
0.7003651392350971
```

```
parameter = {
    'criterion' : ["squared_error", "friedman_mse", "absolute_error",
"poisson"],
    'splitter' : ["best", "random"],
    'max_features':["auto", "sqrt", "log2"]
}
```

```
from sklearn.model_selection import GridSearchCV
gsc = GridSearchCV(dtc,param_grid=parameter,scoring = 'accuracy',cv=5)
gsc.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['squared_error',
'friedman_mse',
                                       'absolute_error', 'poisson'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'splitter': ['best', 'random']},
             scoring='accuracy')
```

```
y_pred = gsc.predict(x_test)
print(r2_score(y_test,y_pred))
```

```
0.8175040357958469
```

**Decision tree regression : 81.75%**

**#Accuracy of all Algorithms :**

**#Linear Regression : 62.36%**

**#Ridge : 62.36%**

**#RandomForestRegressor: 89.78%**

**#Decision tree : 81.75%**

**#So, I can Conclude that RandomForestRegressor is best For prediction based on this dataset...**