

# House Price Predictions with Exploratory Data Analysis and important Observations



Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But in real dataset, it proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

```
In [1]: import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline
```

```
In [2]: # To display the entire columns  
pd.set_option('display.max_columns', None)
```

```
In [3]: # Read the dataset  
  
df = pd.read_csv(r"C:\Users\460379\Python Projects ineuron\Python_EDA_Preprocessing\Py
```

```
In [4]: # Calling the dataset  
df
```

Out[4]:

	<b>Id</b>	<b>MSSubClass</b>	<b>MSZoning</b>	<b>LotFrontage</b>	<b>LotArea</b>	<b>Street</b>	<b>Alley</b>	<b>LotShape</b>	<b>LandContour</b>	<b>U</b>
<b>0</b>	1	60	RL	65.0	8450	Pave	NaN	Reg	Reg	Lvl
<b>1</b>	2	20	RL	80.0	9600	Pave	NaN	Reg	Reg	Lvl
<b>2</b>	3	60	RL	68.0	11250	Pave	NaN	IR1	Reg	Lvl
<b>3</b>	4	70	RL	60.0	9550	Pave	NaN	IR1	Reg	Lvl
<b>4</b>	5	60	RL	84.0	14260	Pave	NaN	IR1	Reg	Lvl
...	...	...	...	...	...	...	...	...	...	...
<b>1455</b>	1456	60	RL	62.0	7917	Pave	NaN	Reg	Reg	Lvl
<b>1456</b>	1457	20	RL	85.0	13175	Pave	NaN	Reg	Reg	Lvl
<b>1457</b>	1458	70	RL	66.0	9042	Pave	NaN	Reg	Reg	Lvl
<b>1458</b>	1459	20	RL	68.0	9717	Pave	NaN	Reg	Reg	Lvl
<b>1459</b>	1460	20	RL	75.0	9937	Pave	NaN	Reg	Reg	Lvl

1460 rows × 81 columns

In [5]: `#Check the number of features in the datasets`  
`df.columns`

Out[5]: `Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC', 'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice'],  
dtype='object')`

In [6]: `#Check the number of columns and number of rows in the dataset`  
`df.shape`

Out[6]: `(1460, 81)`

In [7]: `# Check for the total missing values in the dataset`  
`df.isnull().sum()`

```
Out[7]: Id          0
        MSSubClass   0
        MSZoning     0
        LotFrontage  259
        LotArea       0
        ...
        MoSold       0
        YrSold       0
        SaleType     0
        SaleCondition 0
        SalePrice    0
Length: 81, dtype: int64
```

```
In [8]: # Find the percentage of missing values for all the features in the datasets with decimal places

feature_with_na = [feature for feature in df.columns if df[feature].isnull().sum() > 1]

for feature in feature_with_na:
    print(feature, np.round(df[feature].isnull().mean(), 4))
```

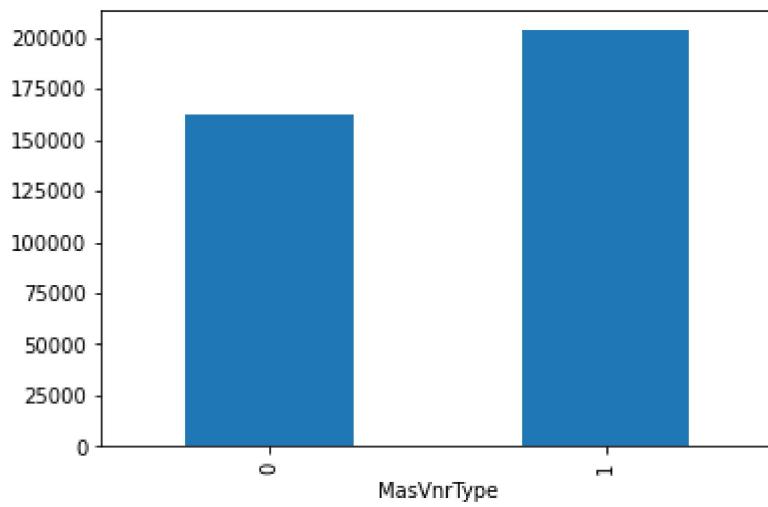
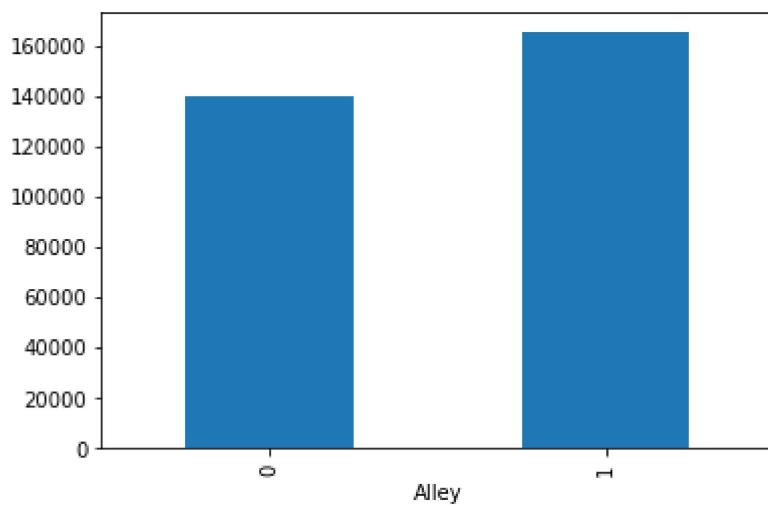
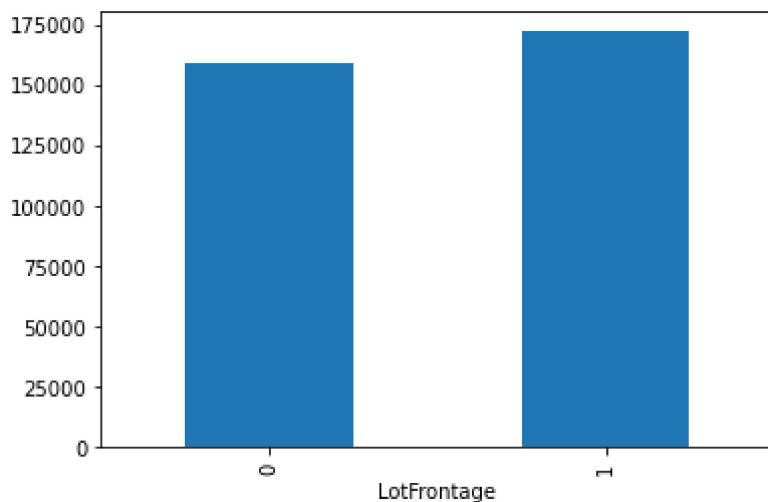
```
LotFrontage 0.1774
Alley 0.9377
MasVnrType 0.0055
MasVnrArea 0.0055
BsmtQual 0.0253
BsmtCond 0.0253
BsmtExposure 0.026
BsmtFinType1 0.0253
BsmtFinType2 0.026
FireplaceQu 0.4726
GarageType 0.0555
GarageYrBlt 0.0555
GarageFinish 0.0555
GarageQual 0.0555
GarageCond 0.0555
PoolQC 0.9952
Fence 0.8075
MiscFeature 0.963
```

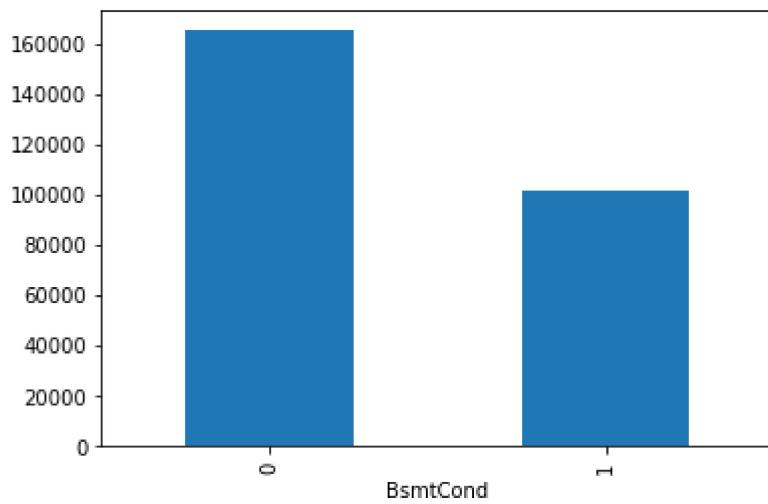
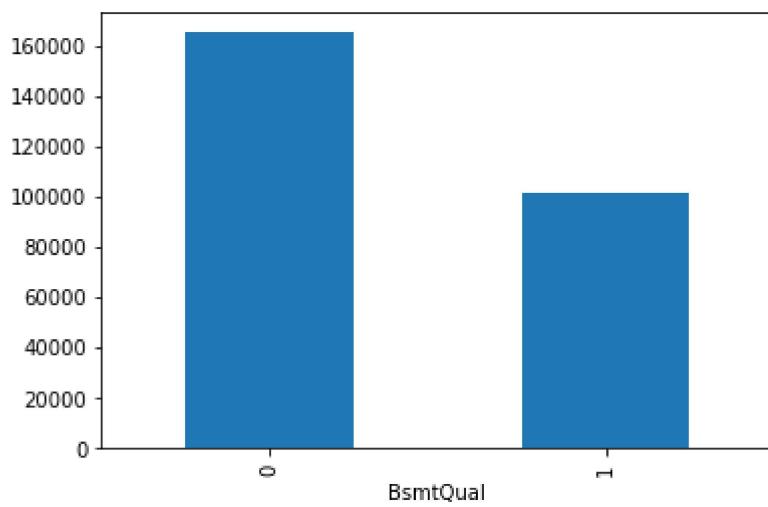
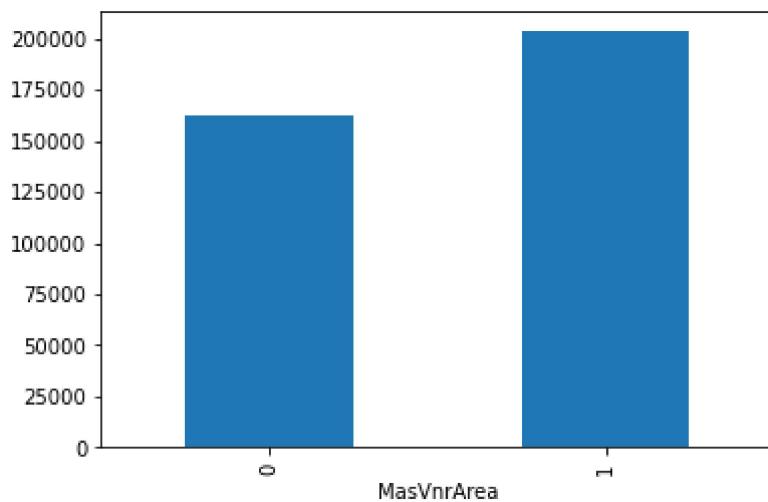
```
In [9]: # Let's plot the missing values of the dataset for all the features

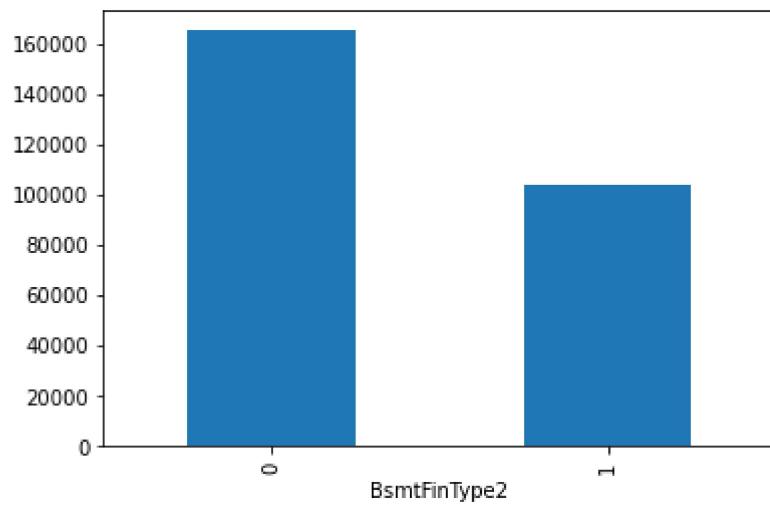
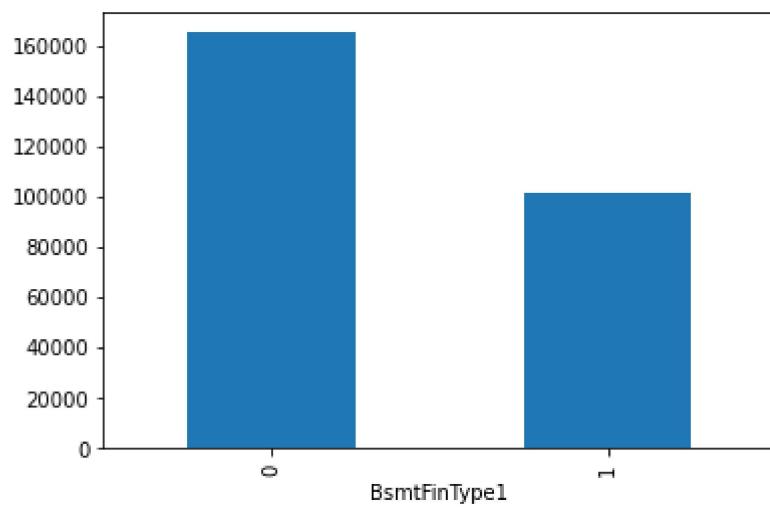
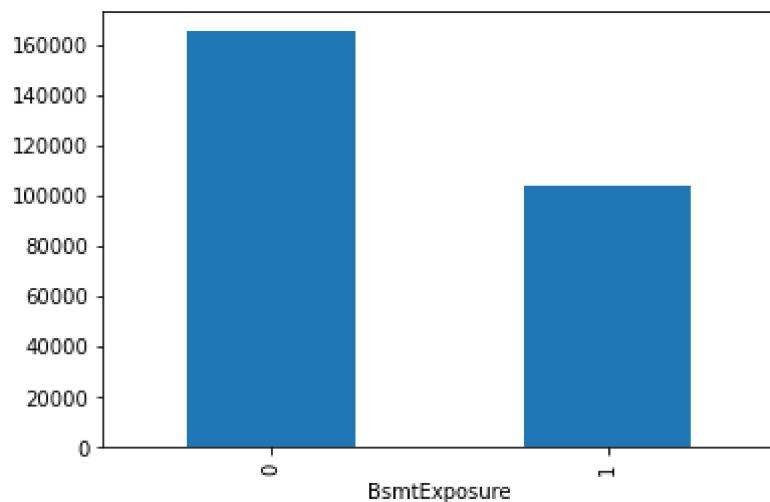
for feature in feature_with_na:
    df1 = df.copy()

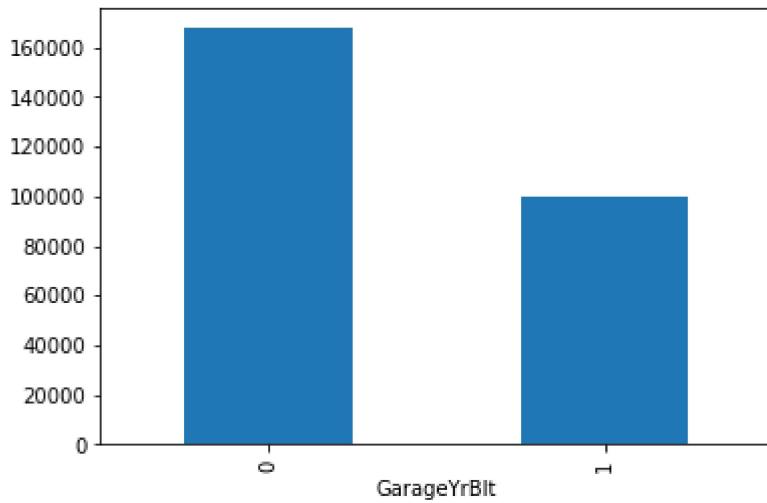
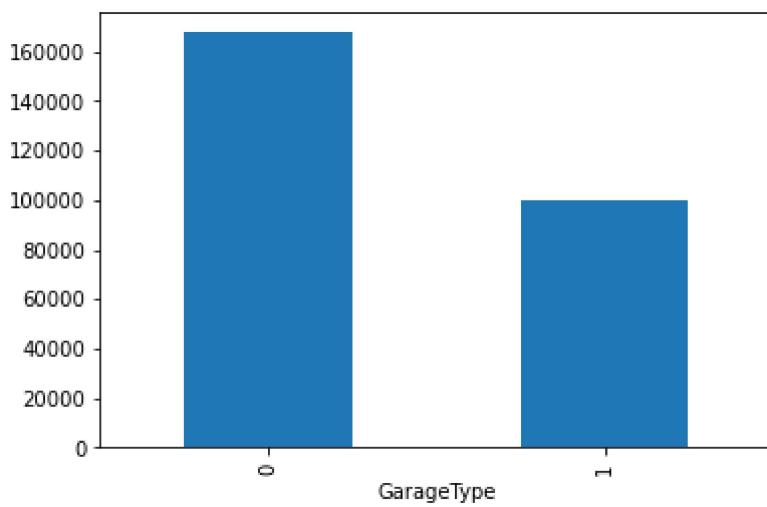
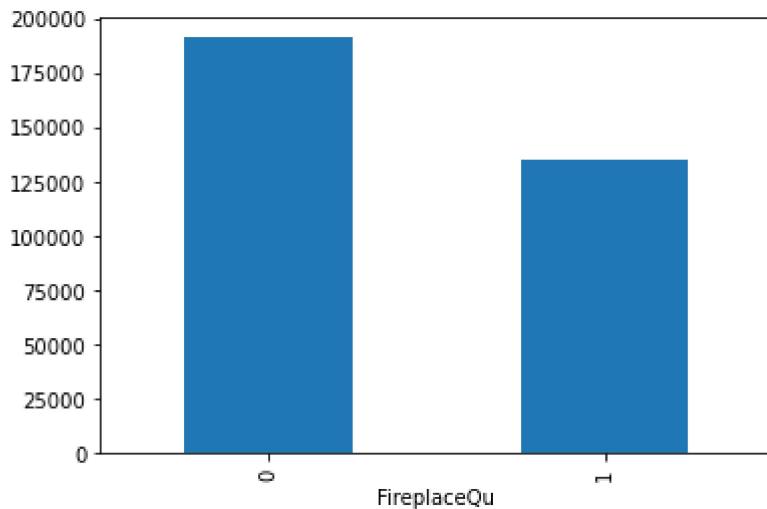
    # Lets create a variable which will indicate 1 if there is null value else 0 if
    df1[feature] = np.where(df1[feature].isnull(), 1, 0)

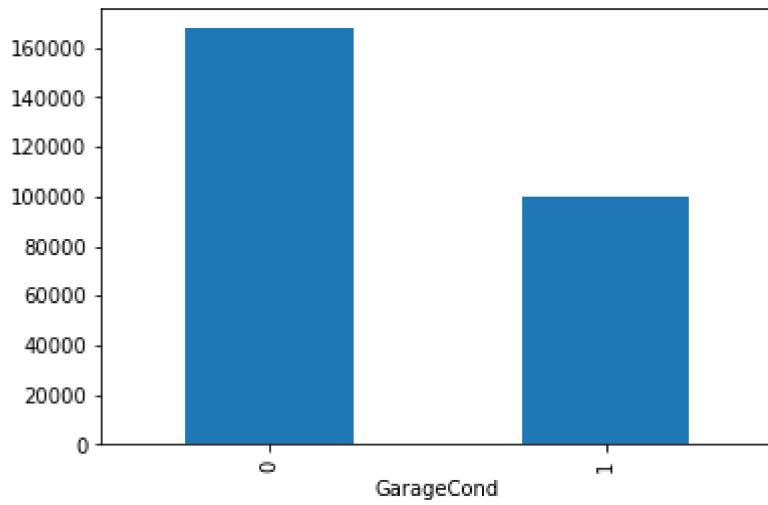
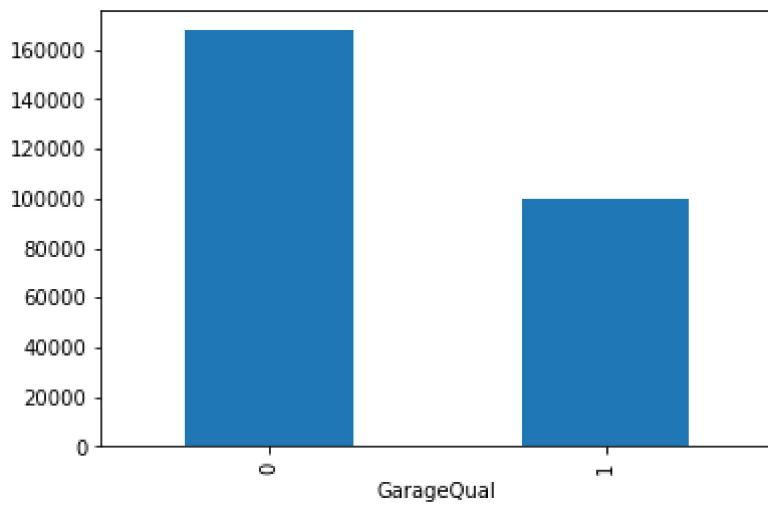
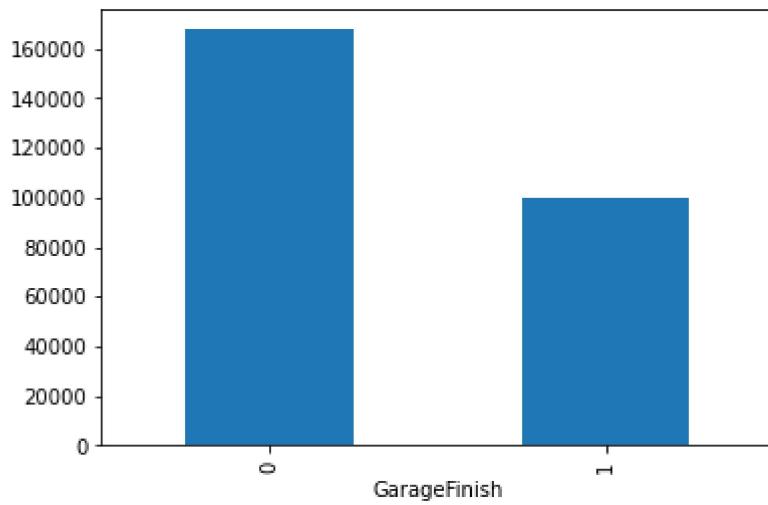
    # Let's plot it using count plot with median sales price in Y axis and understand it
    df1.groupby(feature)['SalePrice'].median().plot.bar()
    plt.show()
```

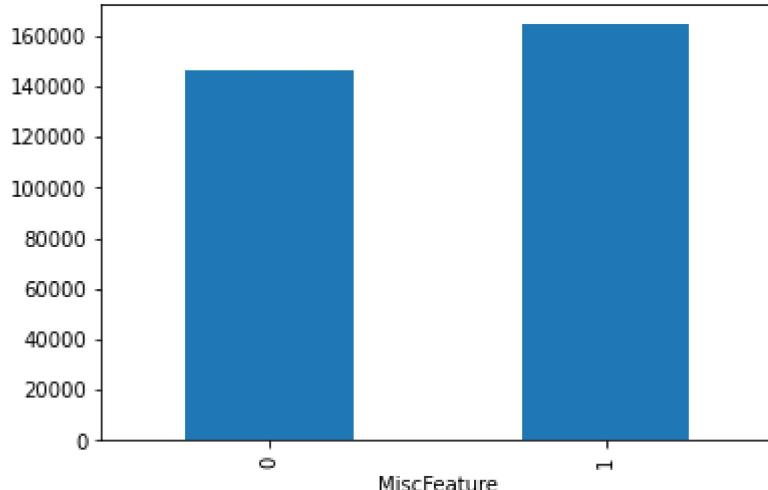
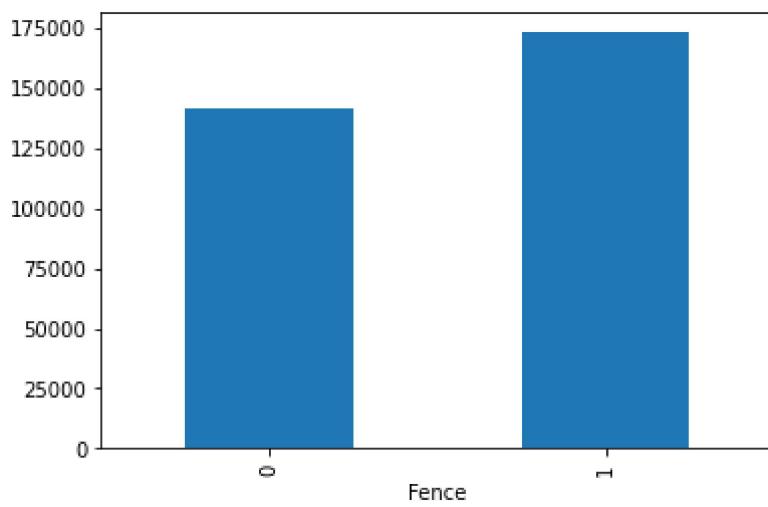
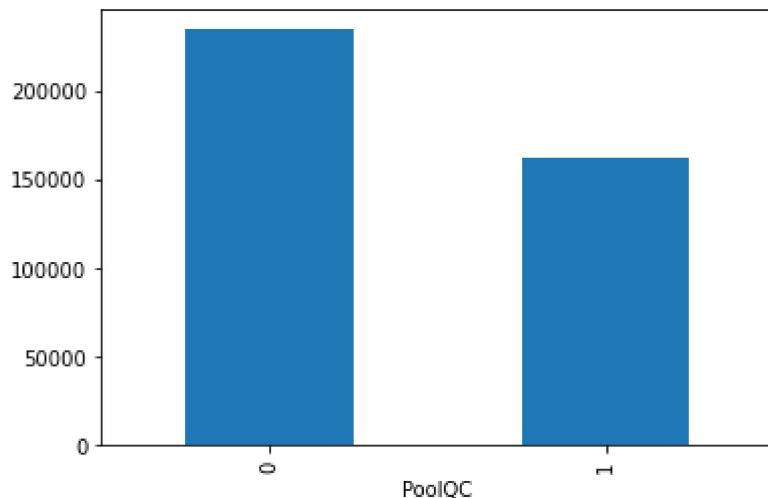












**Observation 1 :** There are significant impact of features with null values with target variable that is "Sales Price" hence we need to treat these missing values in feature engineering very carefully

In [10]: `# Find the categorical features in the dataset`

```
categorical_features = [feature for feature in df.columns if df[feature].dtypes ==  
categorical_features]
```

```
Out[10]: ['MSZoning',
 'Street',
 'Alley',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'PoolQC',
 'Fence',
 'MiscFeature',
 'SaleType',
 'SaleCondition']
```

```
In [11]: # Find the numerical features in the dataset
```

```
Numerical_features = [features for features in df.columns if df[features].dtypes != 'O'
```

```
In [12]: # Find the year features in the datasets since there are multiple year features
```

```
year_features = [features for features in Numerical_features if 'Year' in features or
year_features
```

```
Out[12]: ['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

```
In [13]: # Find the discreet features of the dataset
```

```
discreet_features = [features for features in Numerical_features if len(df[features]).u
```

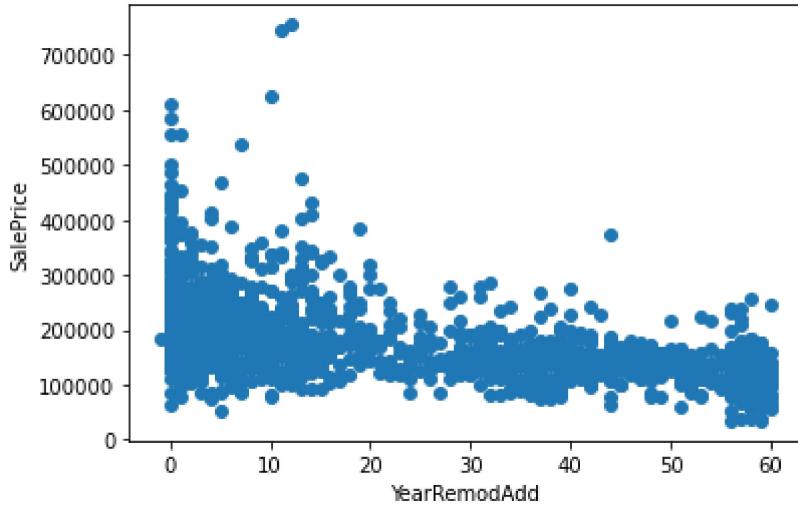
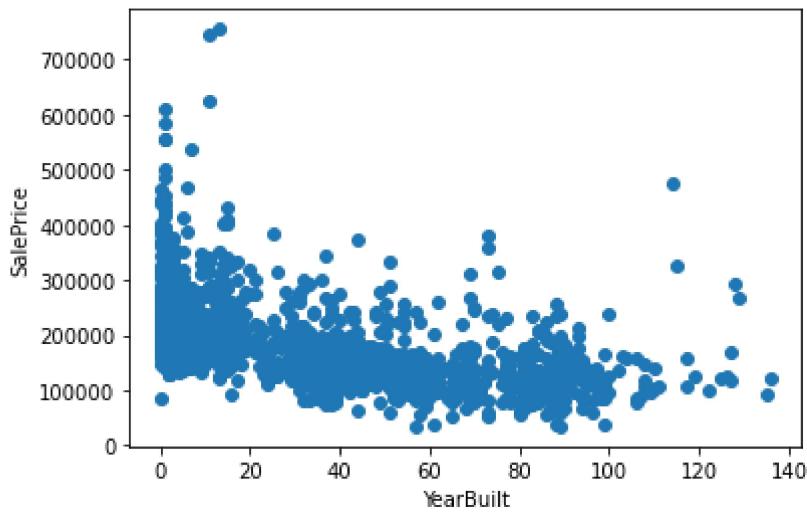
## Bivariate Analysis of Year Features v/s Target Variable ("SalePrice")

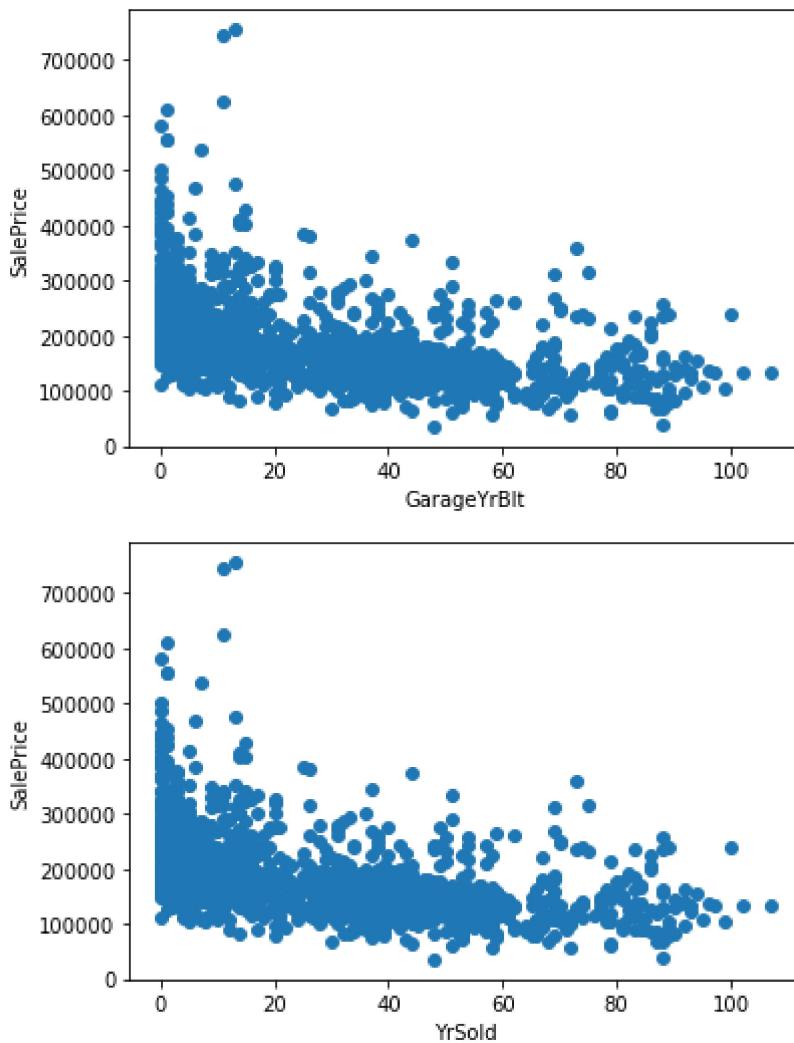
```
In [14]: # Perform Bi-variate analysis of "year feature" with target feature using scatter plot

for features in year_features:
    if features != 'YrSold':
        df1 = df.copy()

# Let's capture the difference between the year variables and year the house was sold
df1[feature] = df1['YrSold']-df1[features]

#Let's plot the difference between year variables and year sold using scatter plot
plt.scatter(df1[feature], df1['SalePrice'])
plt.xlabel(features)
plt.ylabel('SalePrice')
plt.show()
```

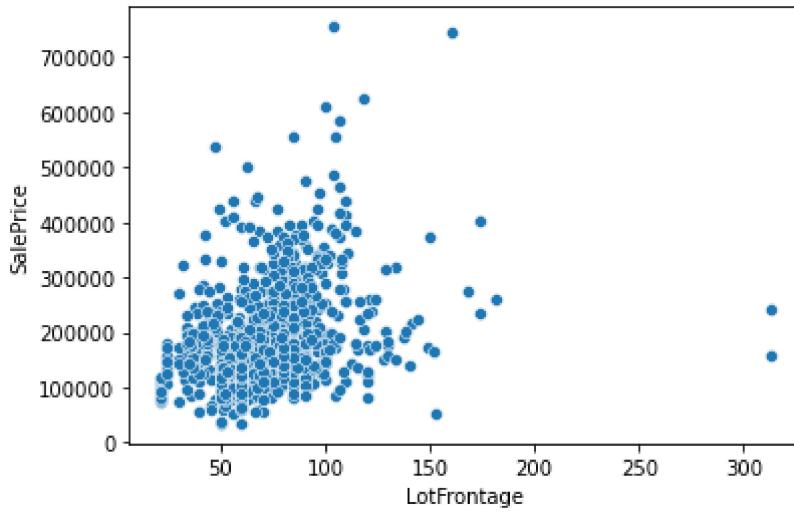
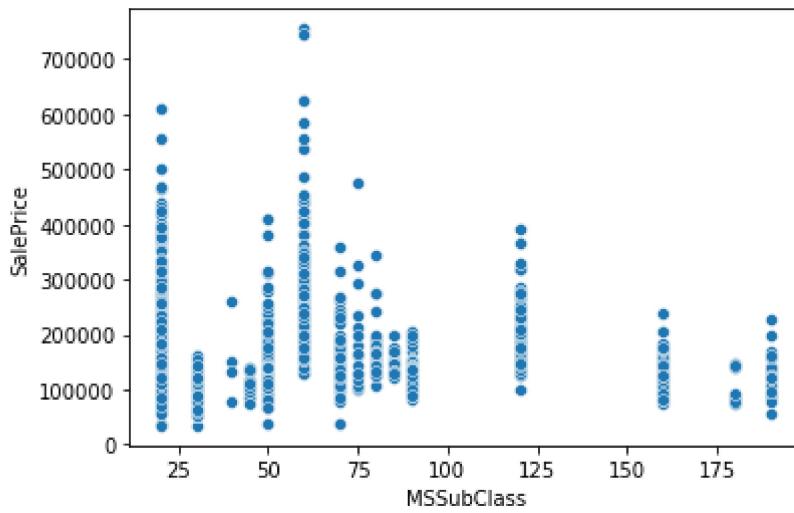
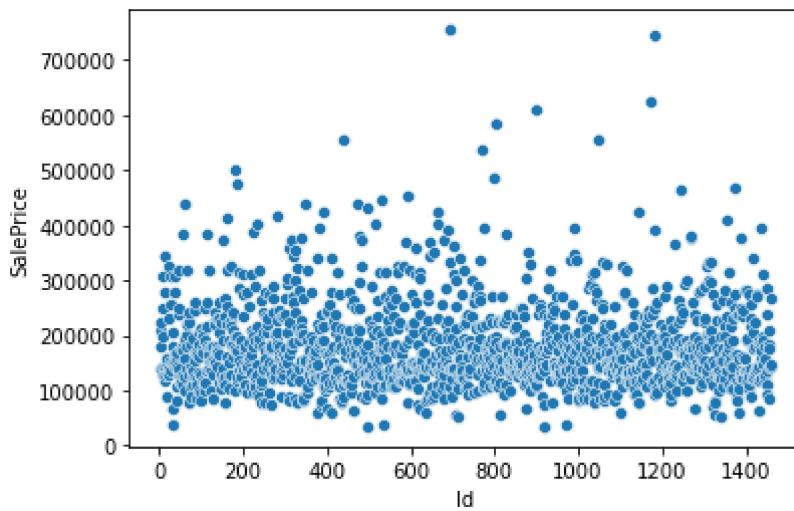


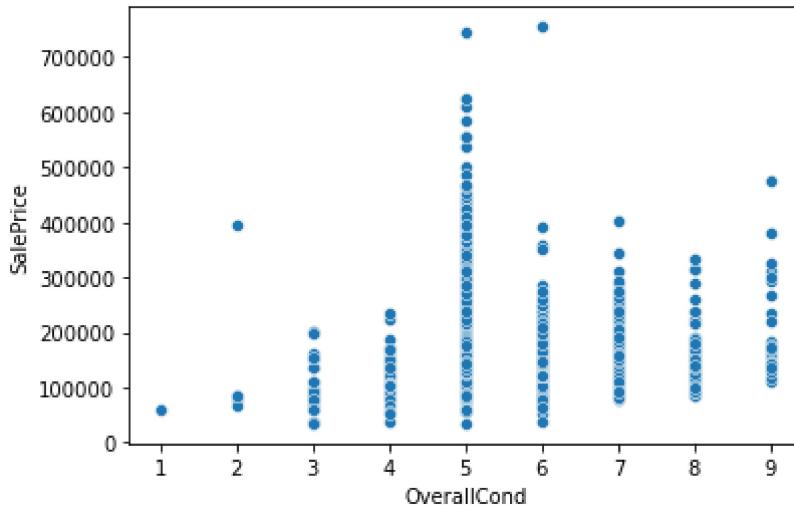
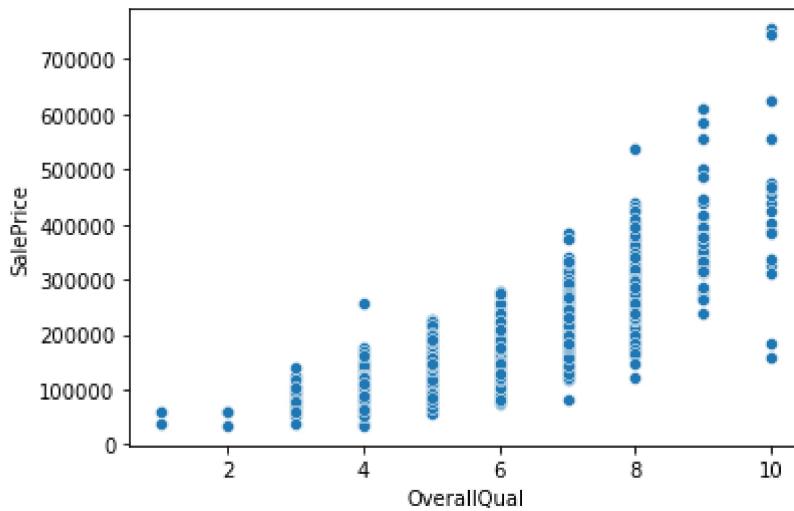
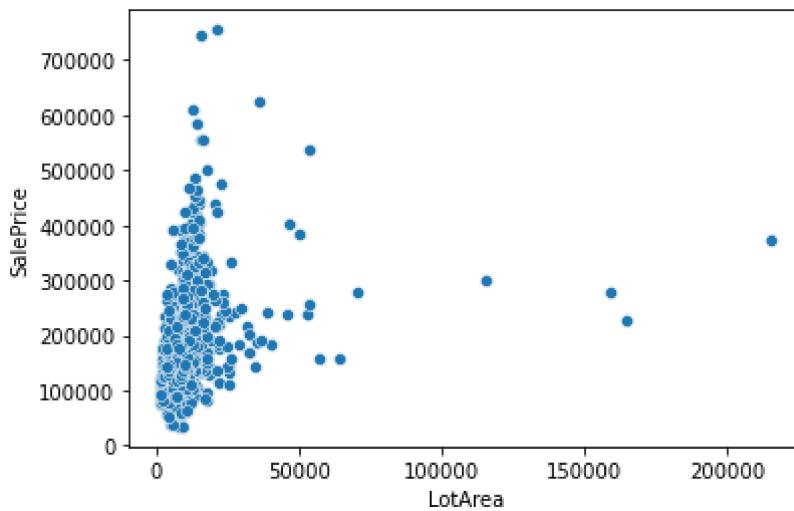


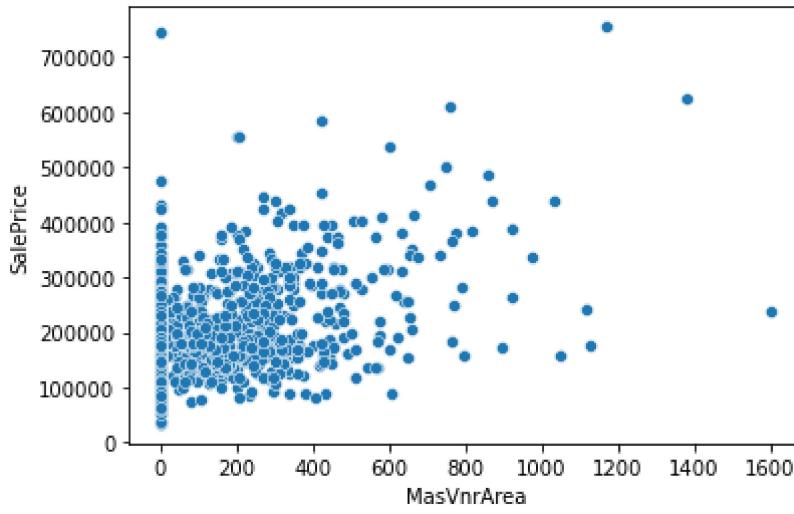
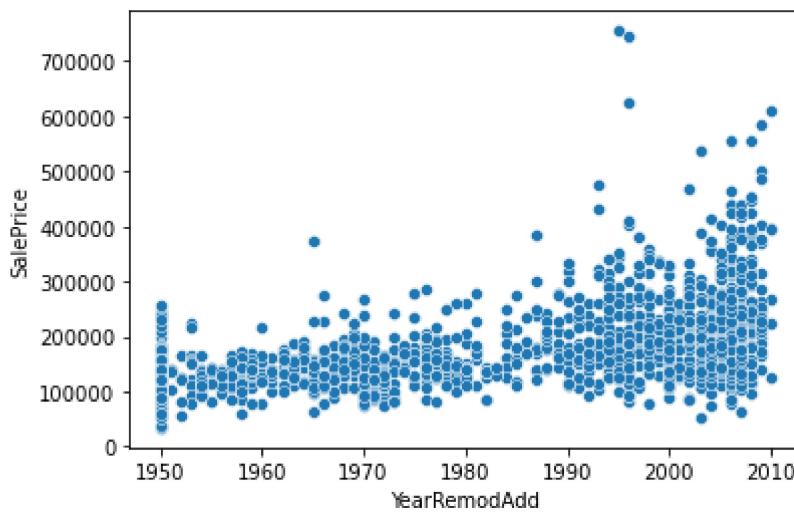
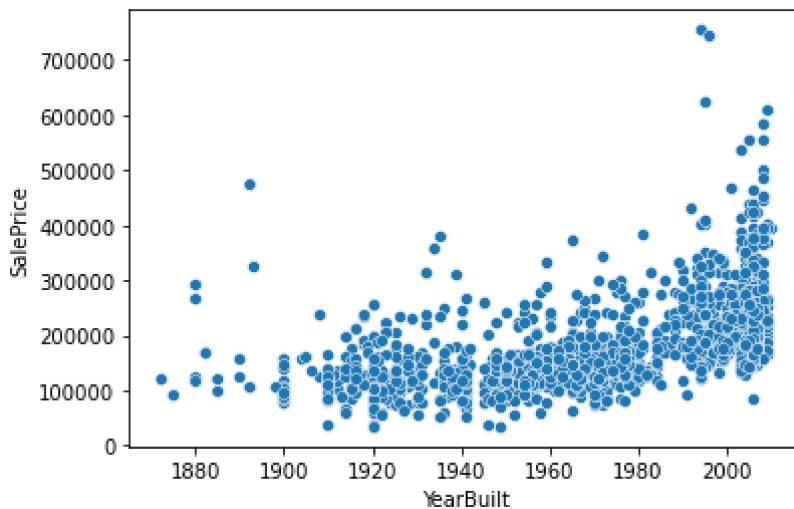
**Observation 2 :** We see that the "sales price" decreases substantially as the "Yearbuilt", "Yearremodel", "GarageYearBuilt" increases and it decreases faster rate after 20 to 25 years

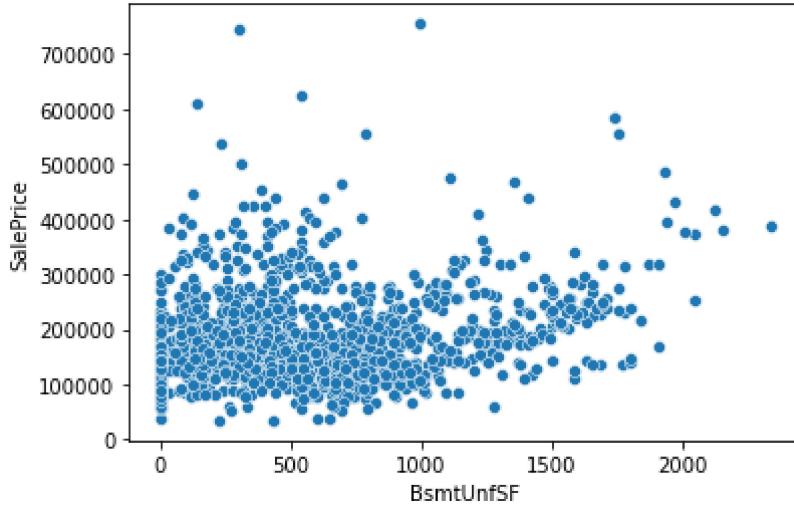
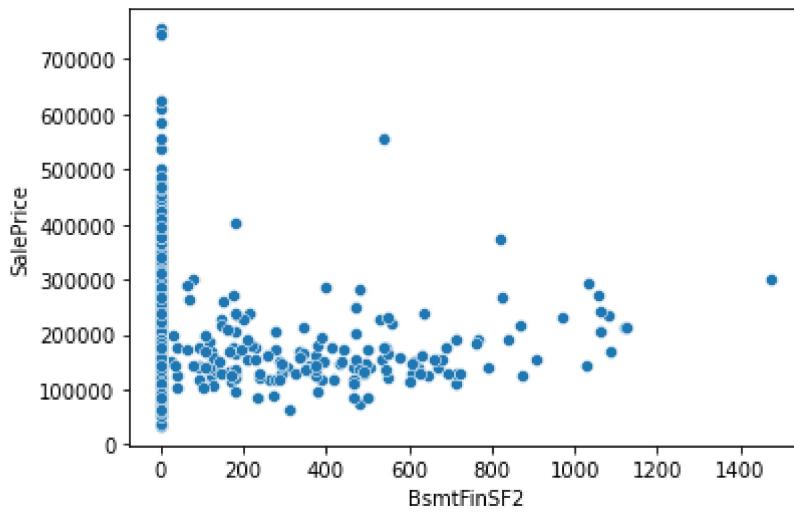
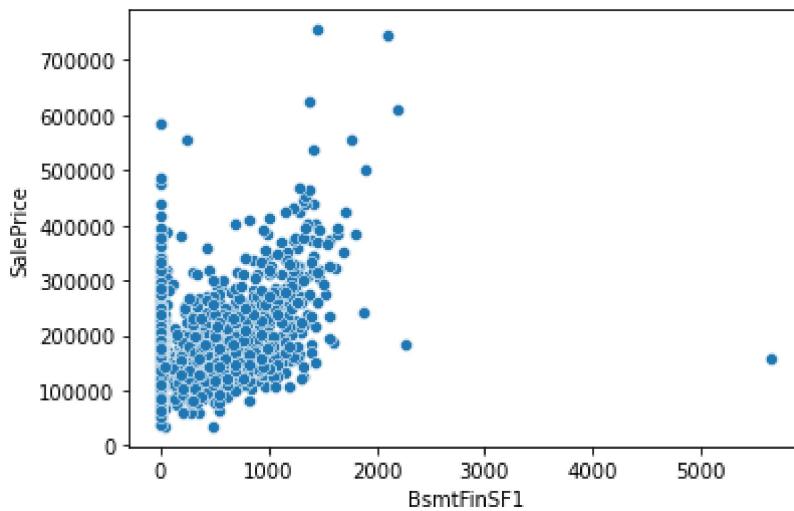
### Bivariate Analysis of Numerical Features v/s Target Feature ("SalePrice")

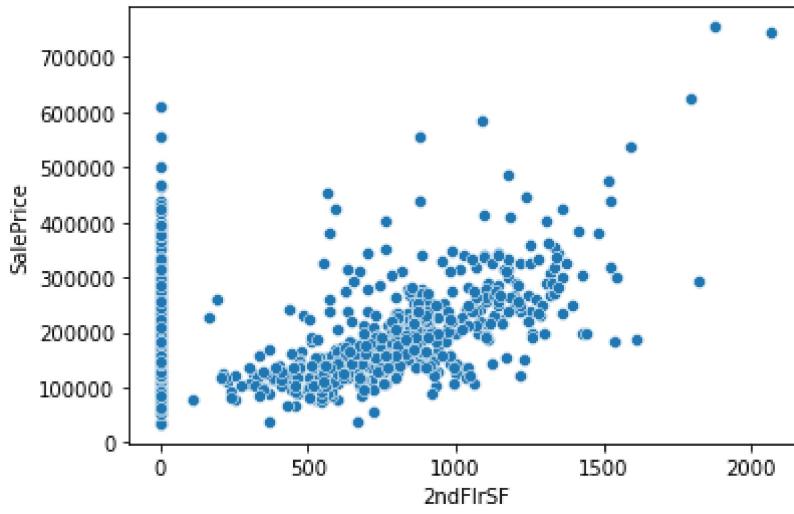
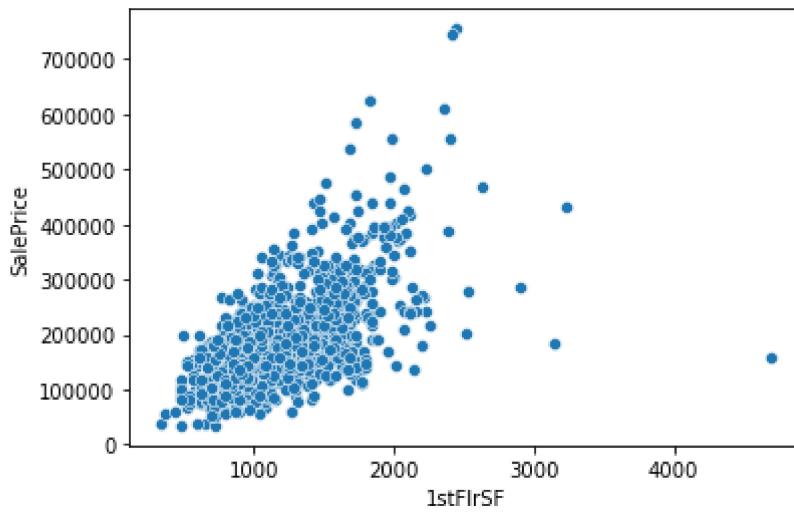
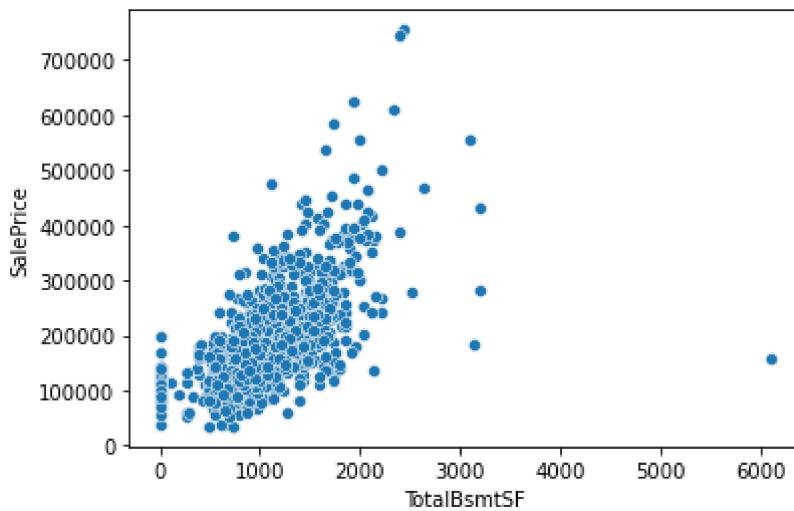
```
In [15]: # Let's perform Bivariate analysis of Numerical features with Target feature using sns  
  
for features in Numerical_features:  
    df1 = df.copy()  
    sns.scatterplot(x = features, y = 'SalePrice', data = df1)  
    plt.show()
```

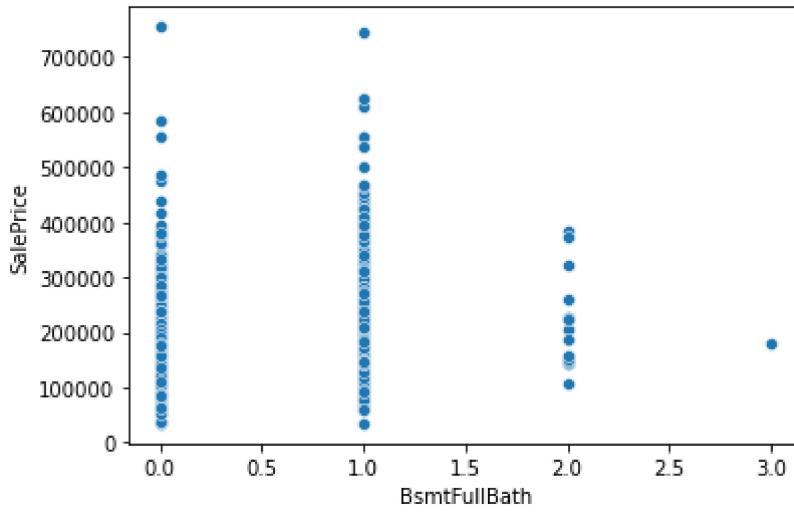
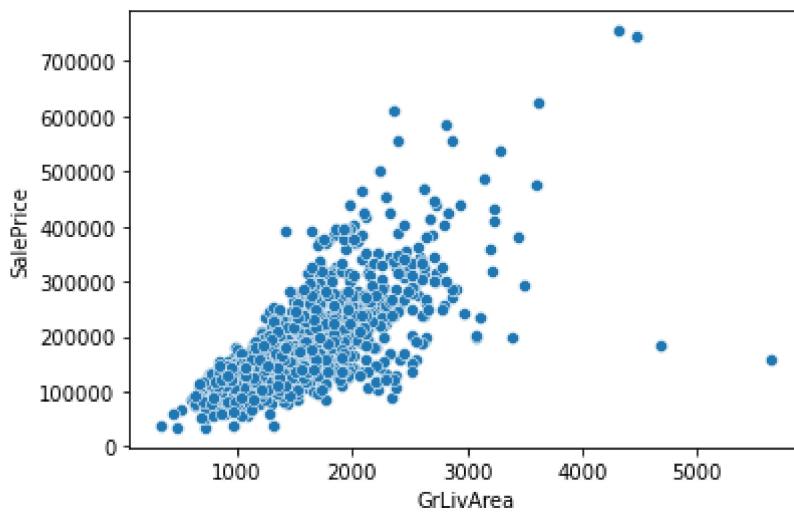
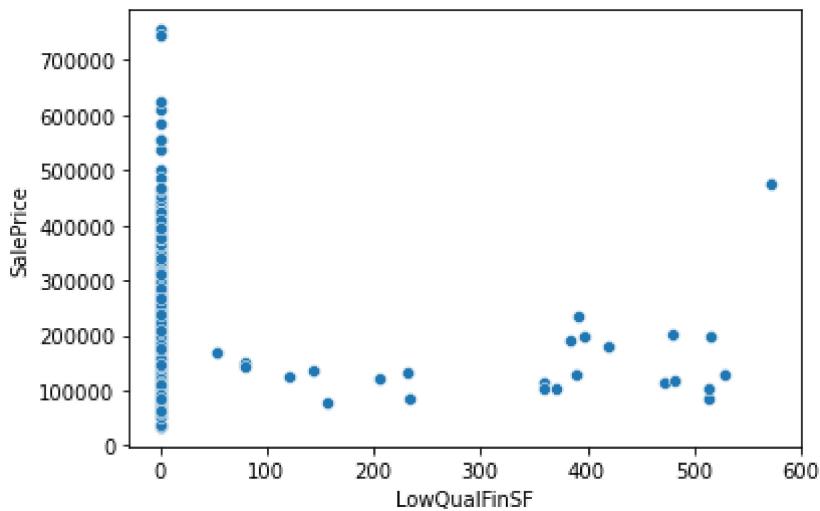


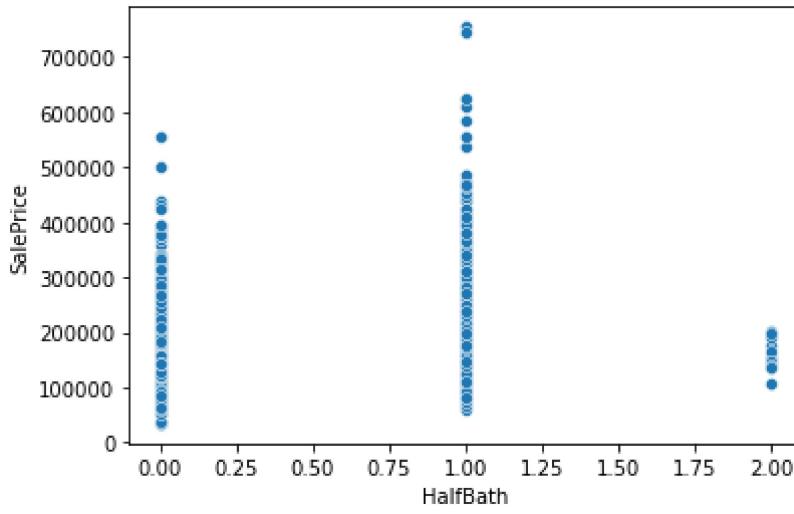
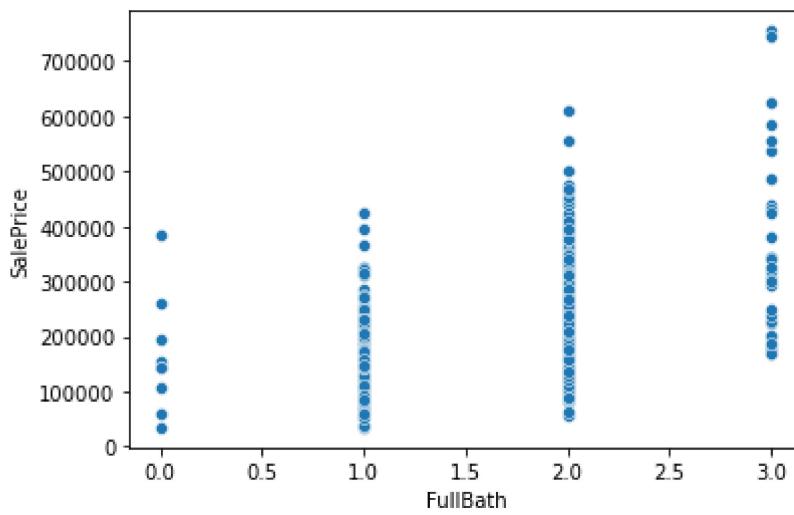
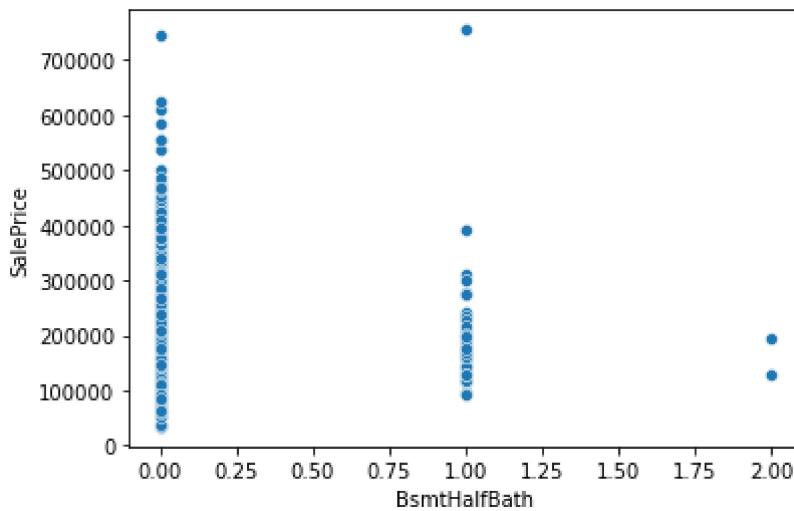


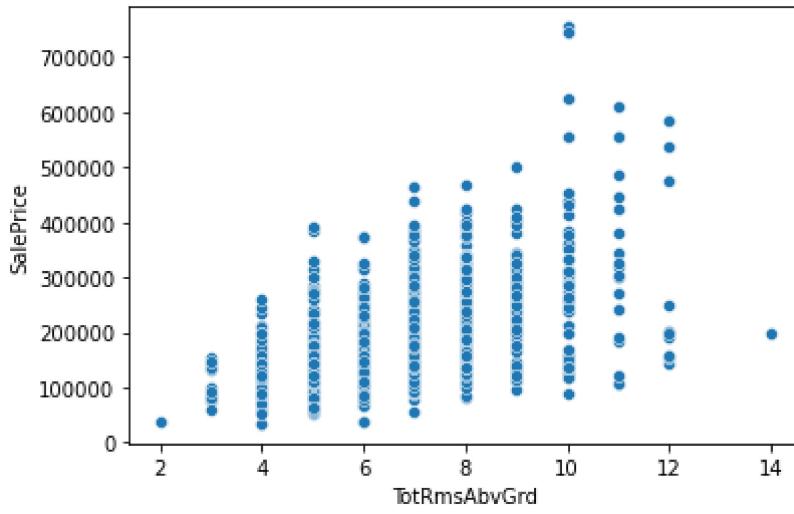
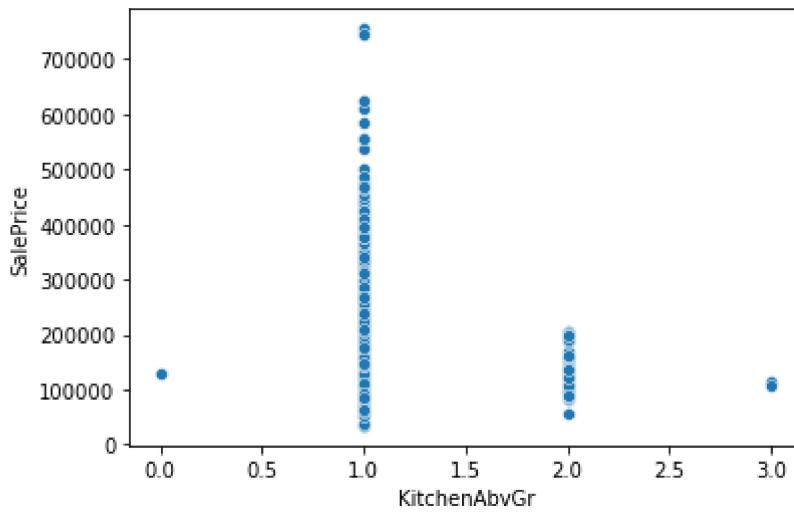
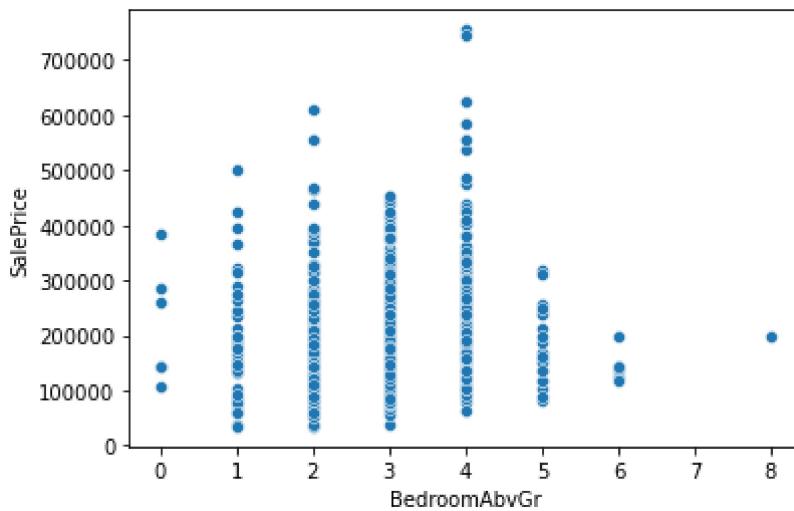


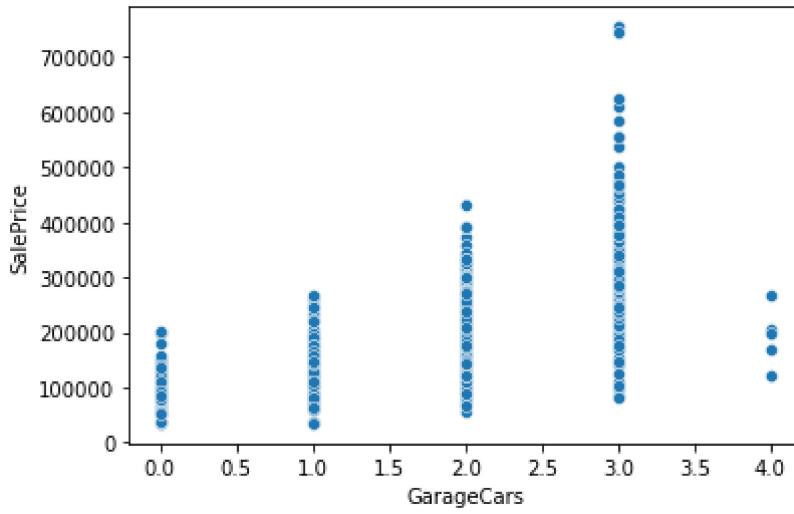
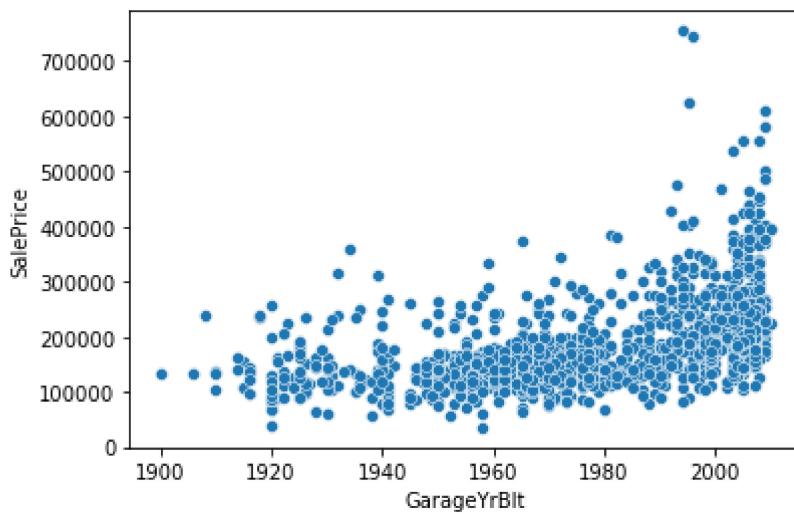
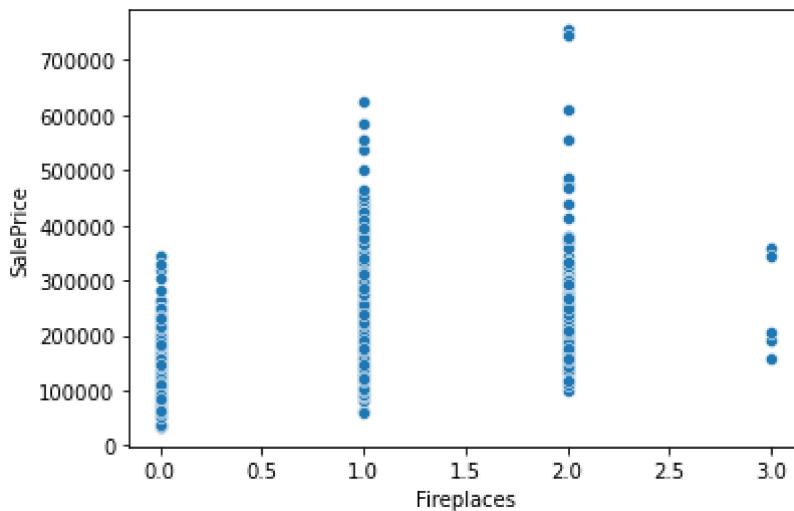


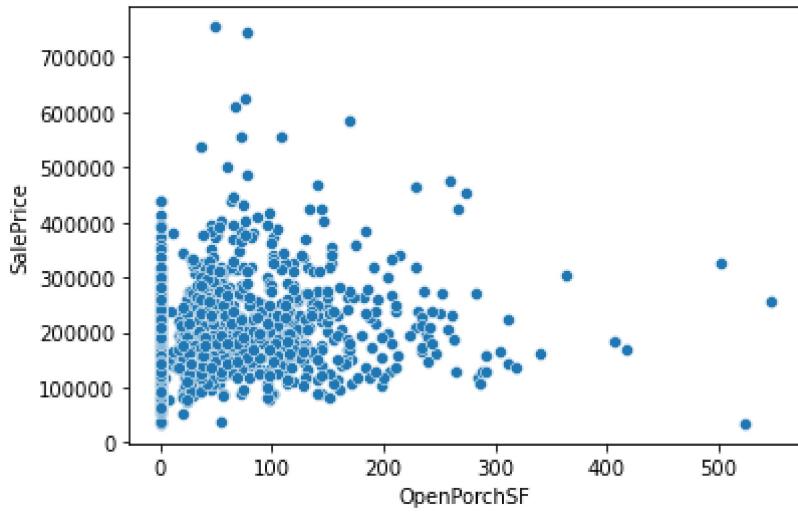
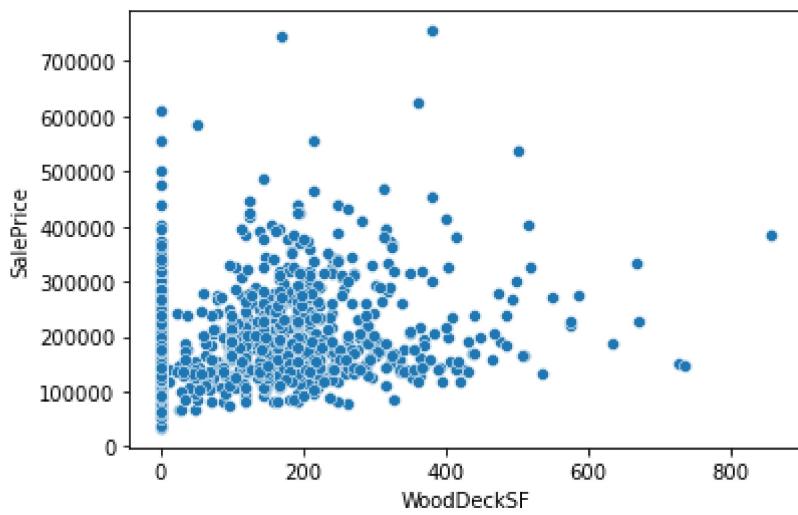
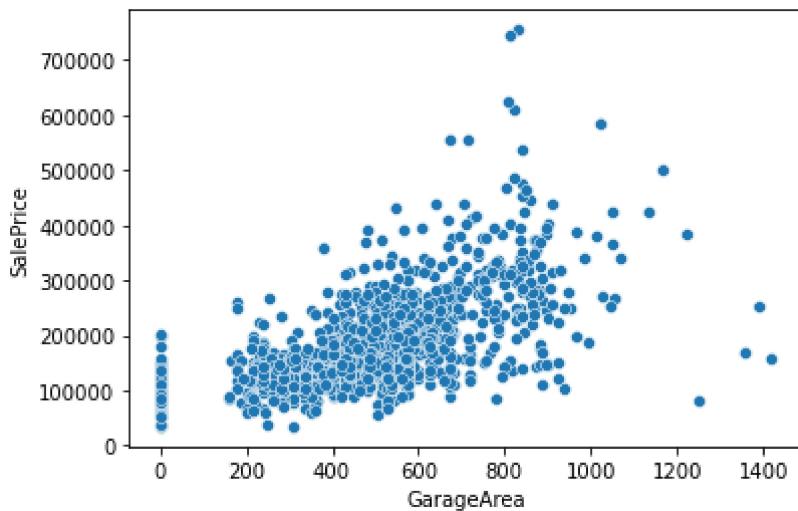


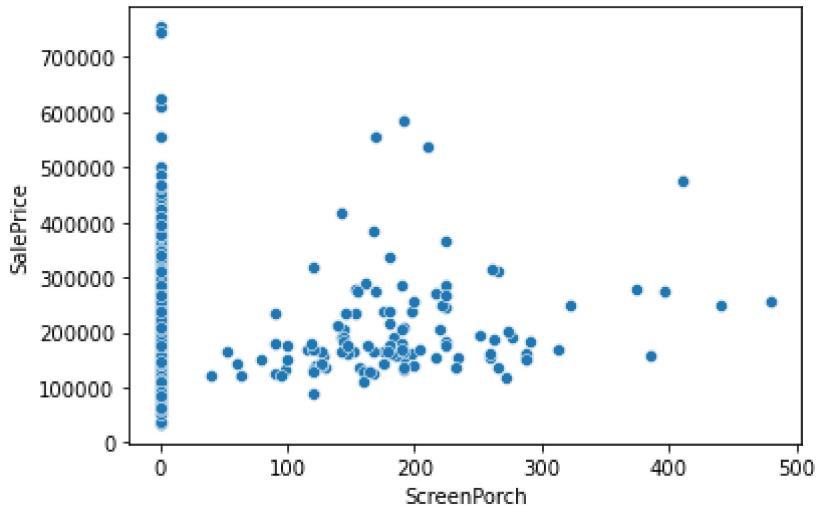
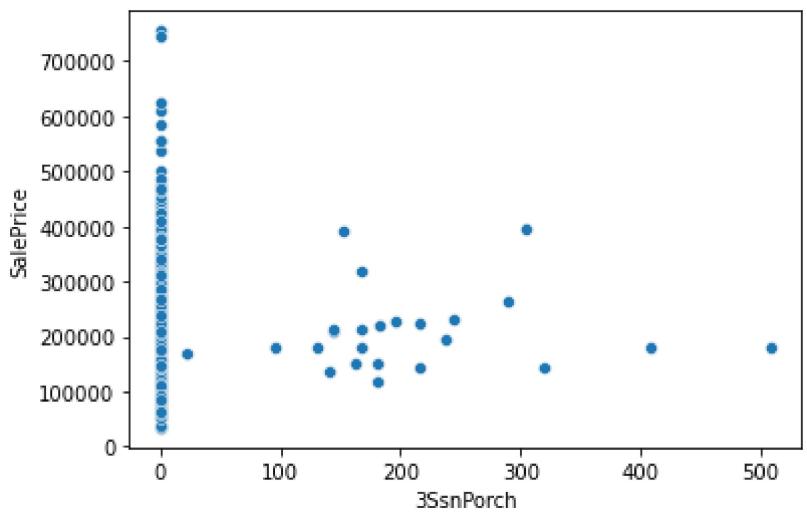
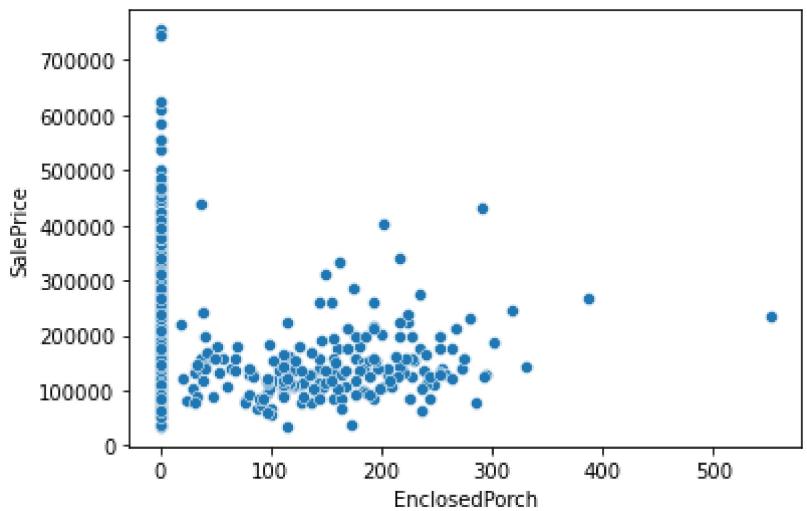


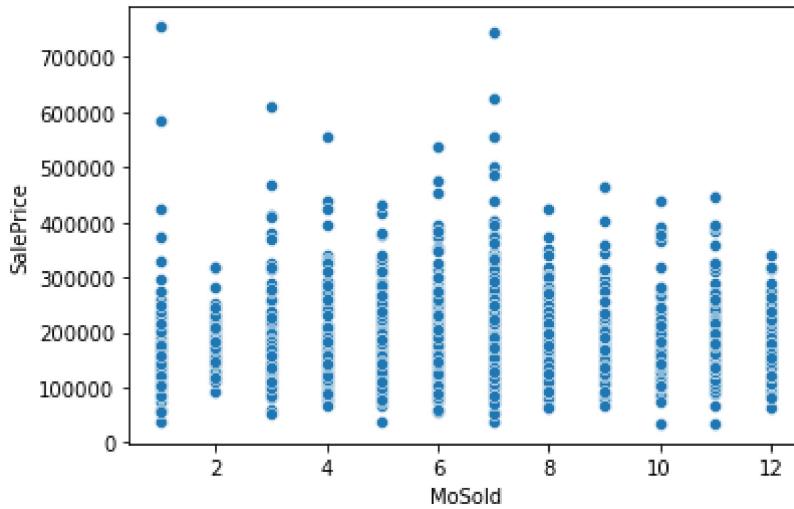
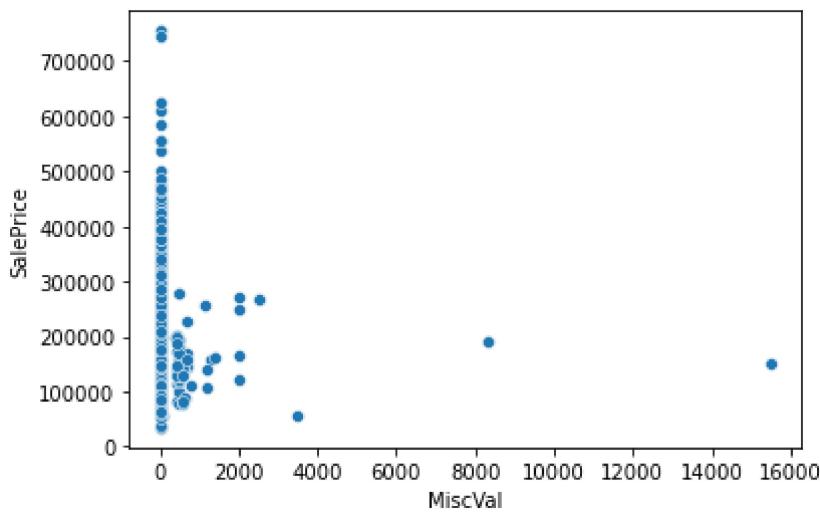
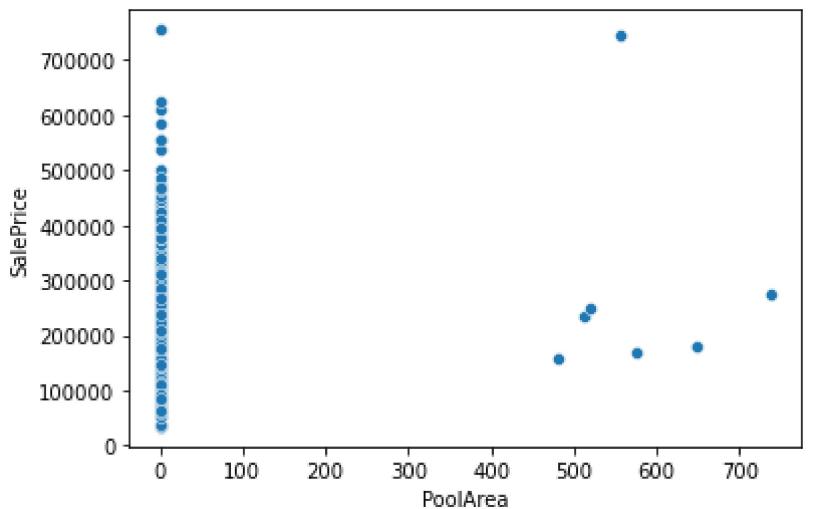


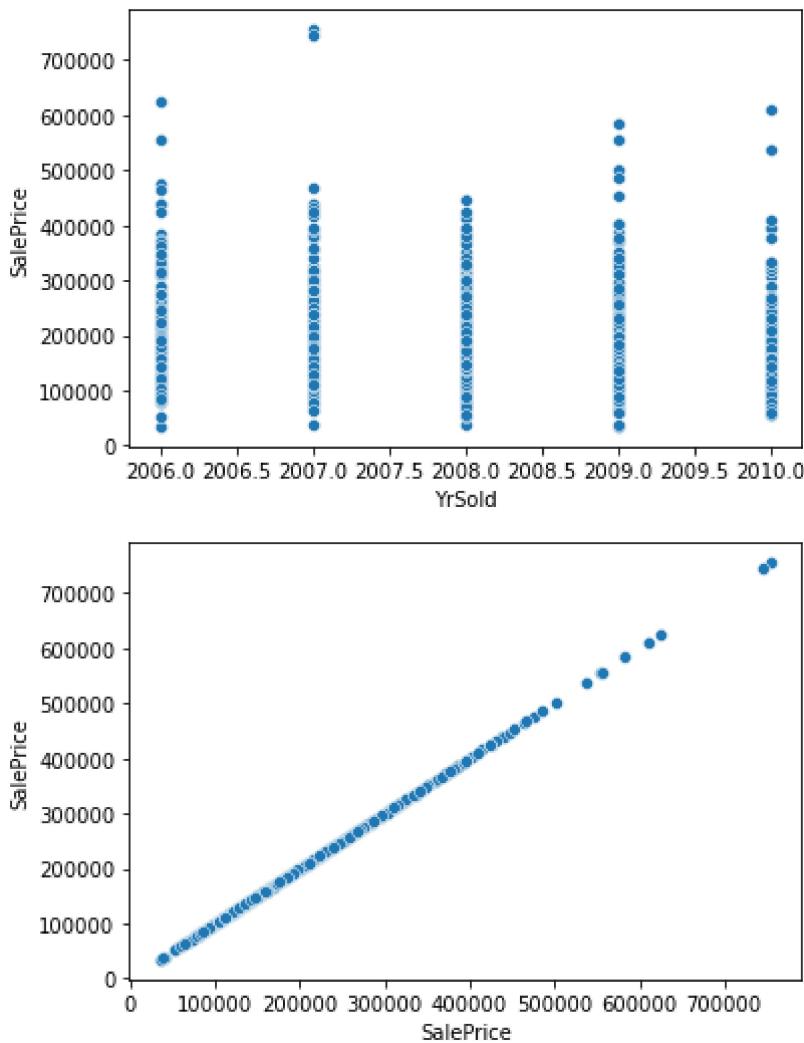










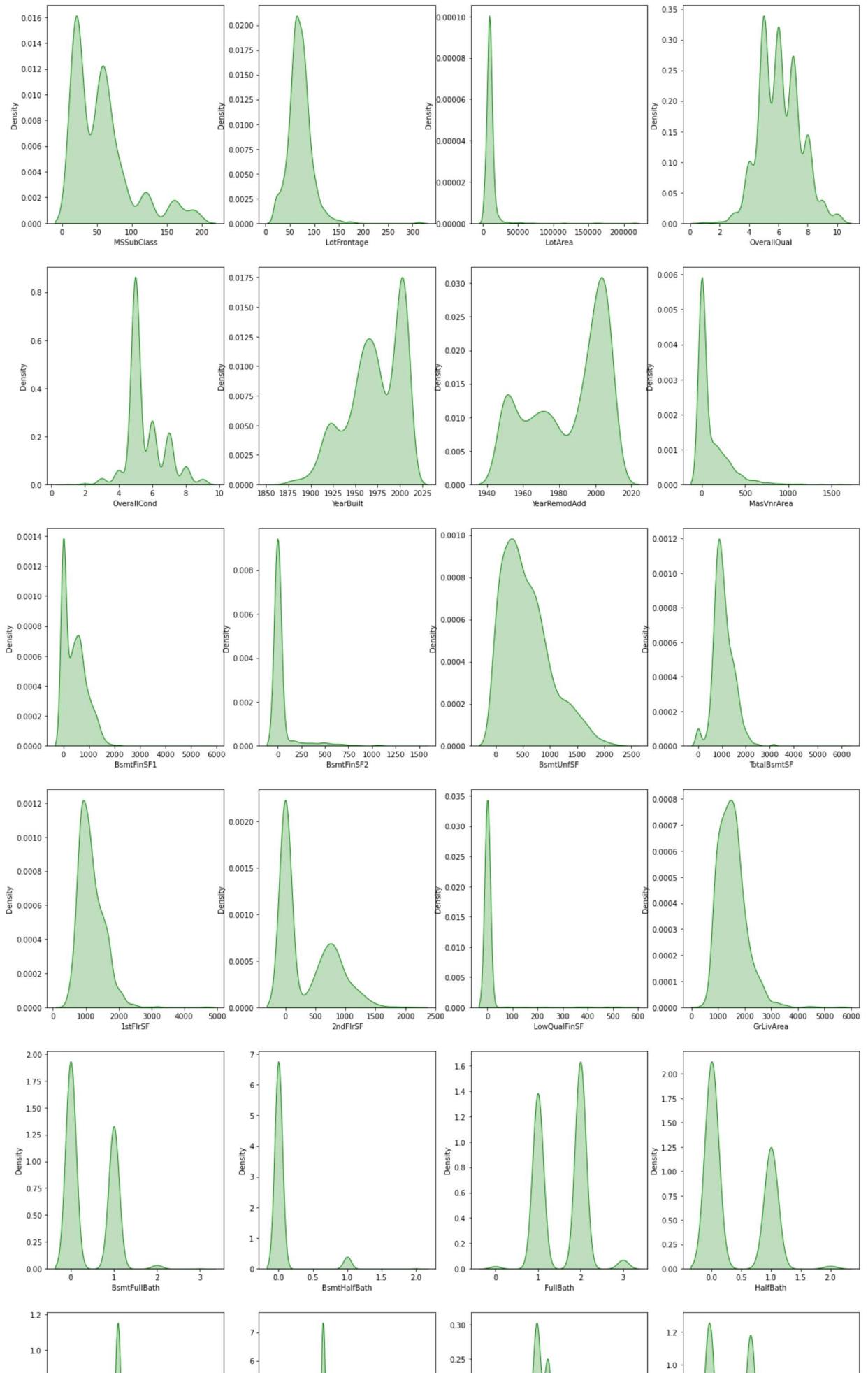


## Univariate Analysis of Numerical Features and Discrete Features

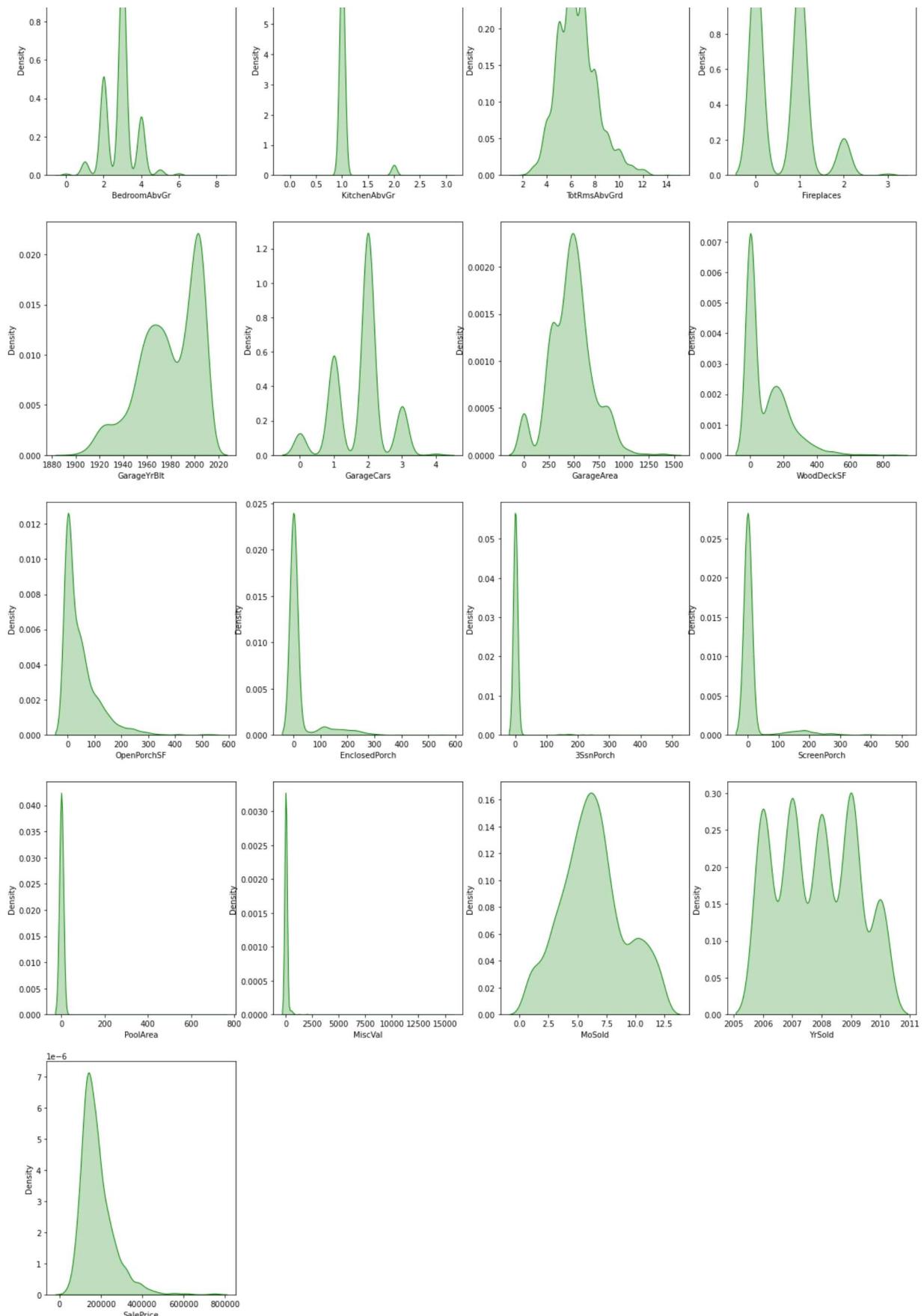
```
In [16]: # Let's first perform Univariate analysis of Numerical features using kdeplot

Numerical_features = [features for features in Numerical_features if features not in discrete_features]
plt.figure(figsize=(20,65))

for i in range(0, len(Numerical_features)):
    df1 = df.copy()
    plt.subplot(10, 4, i+1)
    sns.kdeplot(x=df1[Numerical_features[i]], shade = True, color = 'g',in_layout = True)
    plt.xlabel(Numerical_features[i])
```



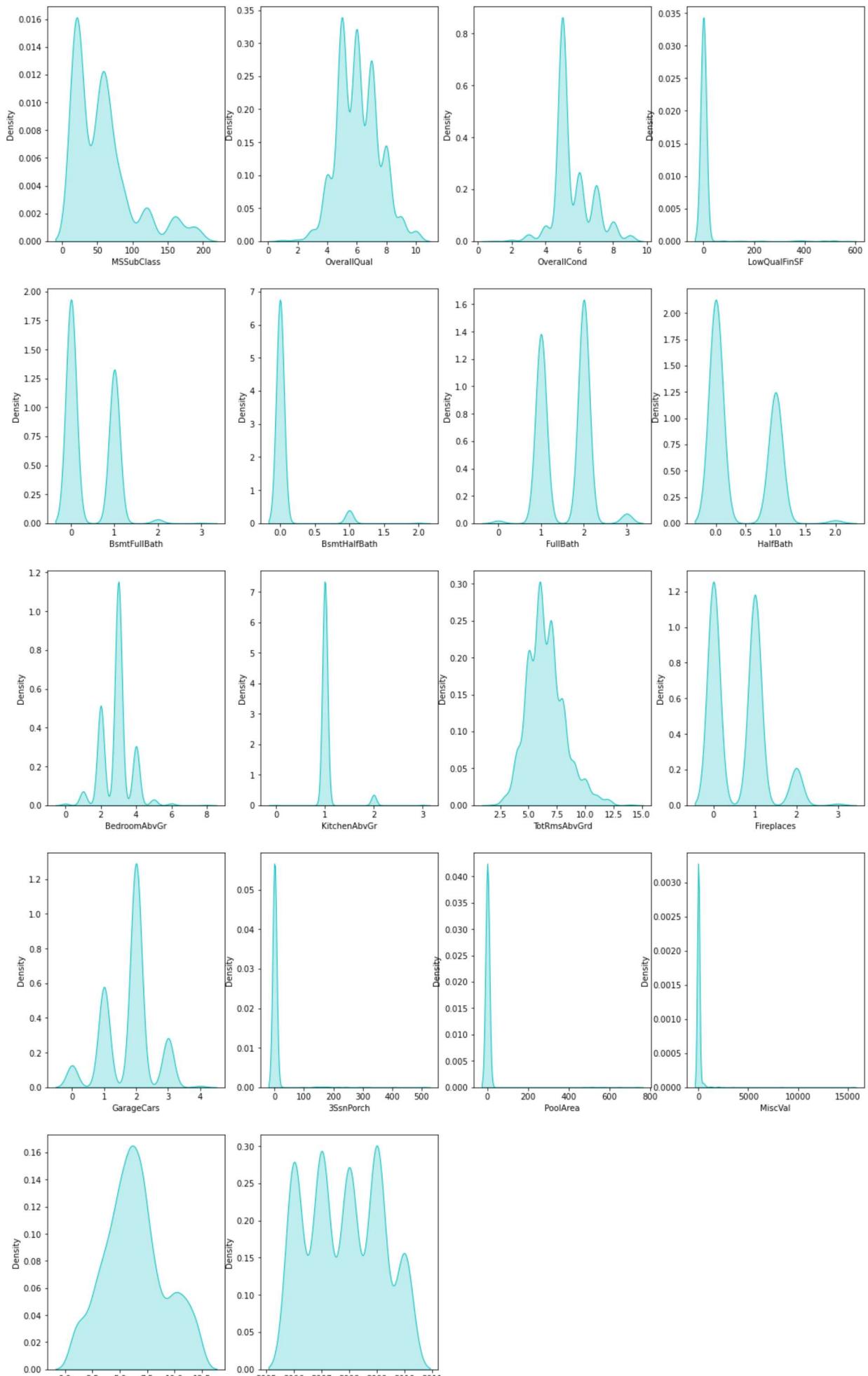
## Python\_HousePriceDataset\_AV\_EDA



```
In [17]: # Let's perform univariate analysis of "Discrete features" using kdeplot
plt.figure(figsize= (18,50))
for i in range(0, len(discrete_features)):
    df1 = df.copy()
    plt.subplot(8, 4, i+1)
```

```
sns.kdeplot(x= df1[discreet_features[i]], shade = True, color = 'c', in_layout = Tr
```

## Python\_HousePriceDataset\_AV\_EDA



0.0 2.5 5.0 7.5 10.0 12.5  
MoSold2003 2006 2007 2009 2010 2011  
YrSold

## Multivariate Analysis of all House Features with Target Feature "SalePrice" using Correlation and Heatmap

In [18]:

```
# Find the correlation of different numerical features
df.corr()
```

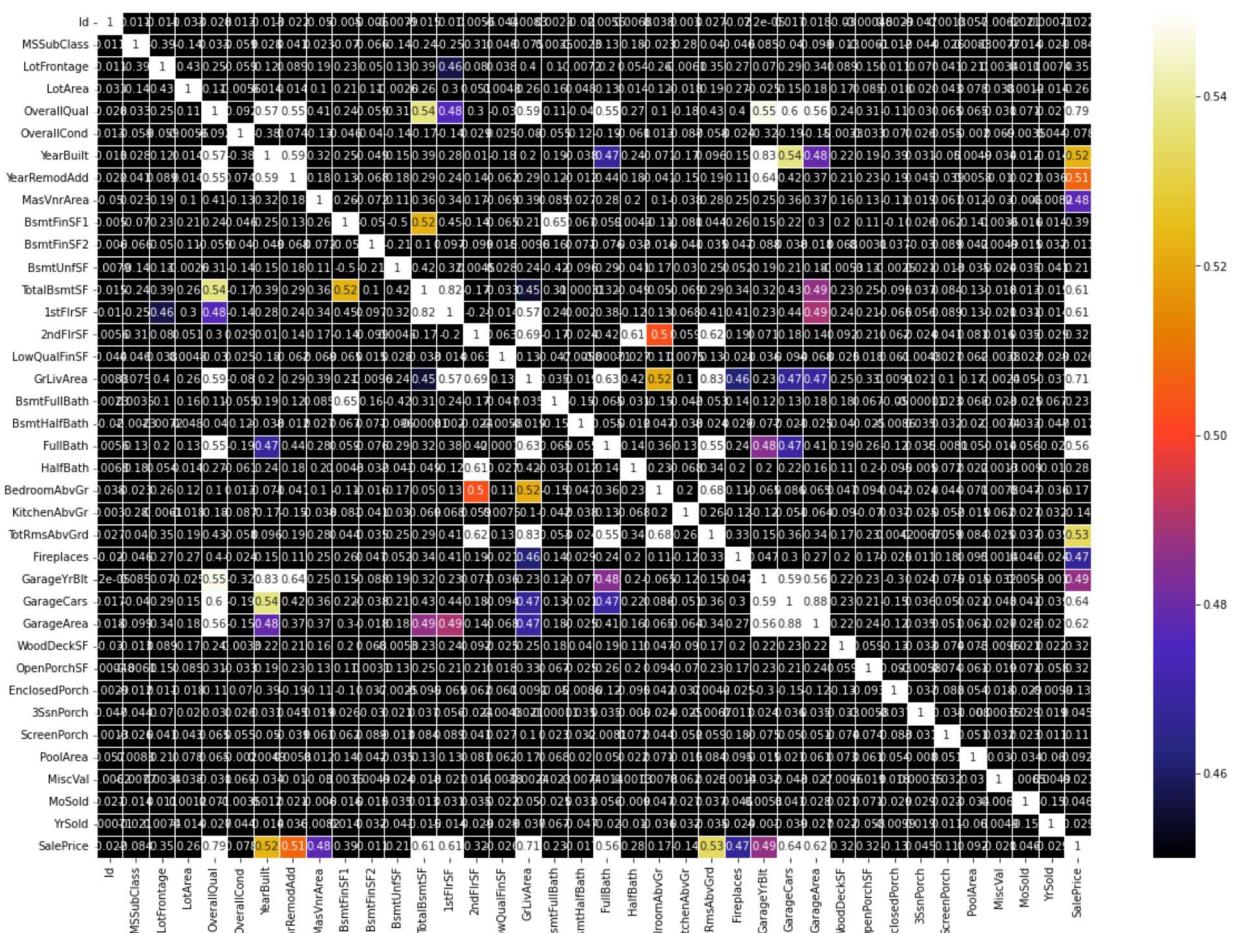
Out[18]:

	<b>Id</b>	<b>MSSubClass</b>	<b>LotFrontage</b>	<b>LotArea</b>	<b>OverallQual</b>	<b>OverallCond</b>	<b>YearBuilt</b>
<b>Id</b>	1.000000	0.011156	-0.010601	-0.033226	-0.028365	0.012609	-0.012713
<b>MSSubClass</b>	0.011156	1.000000	-0.386347	-0.139781	0.032628	-0.059316	0.027850
<b>LotFrontage</b>	-0.010601	-0.386347	1.000000	0.426095	0.251646	-0.059213	0.123349
<b>LotArea</b>	-0.033226	-0.139781	0.426095	1.000000	0.105806	-0.005636	0.014228
<b>OverallQual</b>	-0.028365	0.032628	0.251646	0.105806	1.000000	-0.091932	0.572323
<b>OverallCond</b>	0.012609	-0.059316	-0.059213	-0.005636	-0.091932	1.000000	-0.375983
<b>YearBuilt</b>	-0.012713	0.027850	0.123349	0.014228	0.572323	-0.375983	1.000000
<b>YearRemodAdd</b>	-0.021998	0.040581	0.088866	0.013788	0.550684	0.073741	0.592855
<b>MasVnrArea</b>	-0.050298	0.022936	0.193458	0.104160	0.411876	-0.128101	0.315707
<b>BsmtFinSF1</b>	-0.005024	-0.069836	0.233633	0.214103	0.239666	-0.046231	0.249503
<b>BsmtFinSF2</b>	-0.005968	-0.065649	0.049900	0.111170	-0.059119	0.040229	-0.049107
<b>BsmtUnfSF</b>	-0.007940	-0.140759	0.132644	-0.002618	0.308159	-0.136841	0.149040
<b>TotalBsmtSF</b>	-0.015415	-0.238518	0.392075	0.260833	0.537808	-0.171098	0.391452
<b>1stFlrSF</b>	0.010496	-0.251758	0.457181	0.299475	0.476224	-0.144203	0.281986
<b>2ndFlrSF</b>	0.005590	0.307886	0.080177	0.050986	0.295493	0.028942	0.010308
<b>LowQualFinSF</b>	-0.044230	0.046474	0.038469	0.004779	-0.030429	0.025494	-0.183784
<b>GrLivArea</b>	0.008273	0.074853	0.402797	0.263116	0.593007	-0.079686	0.199010
<b>BsmtFullBath</b>	0.002289	0.003491	0.100949	0.158155	0.111098	-0.054942	0.187599
<b>BsmtHalfBath</b>	-0.020155	-0.002333	-0.007234	0.048046	-0.040150	0.117821	-0.038162
<b>FullBath</b>	0.005587	0.131608	0.198769	0.126031	0.550600	-0.194149	0.468271
<b>HalfBath</b>	0.006784	0.177354	0.053532	0.014259	0.273458	-0.060769	0.242656
<b>BedroomAbvGr</b>	0.037719	-0.023438	0.263170	0.119690	0.101676	0.012980	-0.070651
<b>KitchenAbvGr</b>	0.002951	0.281721	-0.006069	-0.017784	-0.183882	-0.087001	-0.174800
<b>TotRmsAbvGrd</b>	0.027239	0.040380	0.352096	0.190015	0.427452	-0.057583	0.095589
<b>Fireplaces</b>	-0.019772	-0.045569	0.266639	0.271364	0.396765	-0.023820	0.147716
<b>GarageYrBlt</b>	0.000072	0.085072	0.070250	-0.024947	0.547766	-0.324297	0.825667
<b>GarageCars</b>	0.016570	-0.040110	0.285691	0.154871	0.600671	-0.185758	0.537850
<b>GarageArea</b>	0.017634	-0.098672	0.344997	0.180403	0.562022	-0.151521	0.478954
<b>WoodDeckSF</b>	-0.029643	-0.012579	0.088521	0.171698	0.238923	-0.003334	0.224880
<b>OpenPorchSF</b>	-0.000477	-0.006100	0.151972	0.084774	0.308819	-0.032589	0.188686
<b>EnclosedPorch</b>	0.002889	-0.012037	0.010700	-0.018340	-0.113937	0.070356	-0.387268
<b>3SsnPorch</b>	-0.046635	-0.043825	0.070029	0.020423	0.030371	0.025504	0.031355
<b>ScreenPorch</b>	0.001330	-0.026030	0.041383	0.043160	0.064886	0.054811	-0.050364

	<b>Id</b>	<b>MSSubClass</b>	<b>LotFrontage</b>	<b>LotArea</b>	<b>OverallQual</b>	<b>OverallCond</b>	<b>YearBuilt</b>
<b>PoolArea</b>	0.057044	0.008283	0.206167	0.077672	0.065166	-0.001985	0.004950
<b>MiscVal</b>	-0.006242	-0.007683	0.003368	0.038068	-0.031406	0.068777	-0.034383
<b>MoSold</b>	0.021172	-0.013585	0.011200	0.001205	0.070815	-0.003511	0.012398
<b>YrSold</b>	0.000712	-0.021407	0.007450	-0.014261	-0.027347	0.043950	-0.013618
<b>SalePrice</b>	-0.021917	-0.084284	0.351799	0.263843	0.790982	-0.077856	0.522897

In [19]: # Now, Let's plot the heatmap to find the multicollinearity between different features

```
plt.figure(figsize = (20, 14))
sns.heatmap(df.corr(), cmap="CMRmap", annot=True, vmin = .5, vmax = .5, linewidth = .5)
plt.show()
```



## Bivariate Analysis of Categorical Features

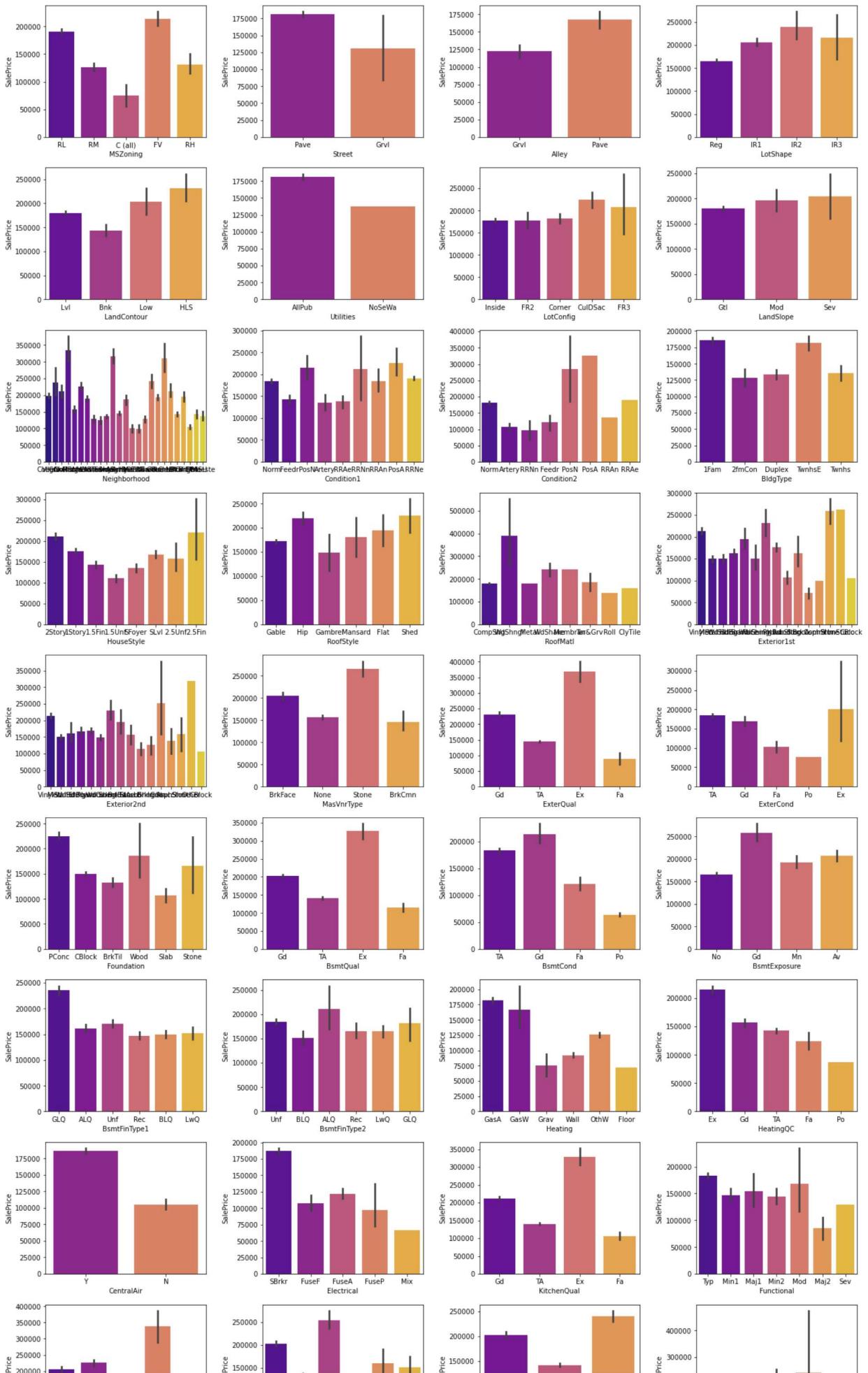
In [20]: categorical\_features

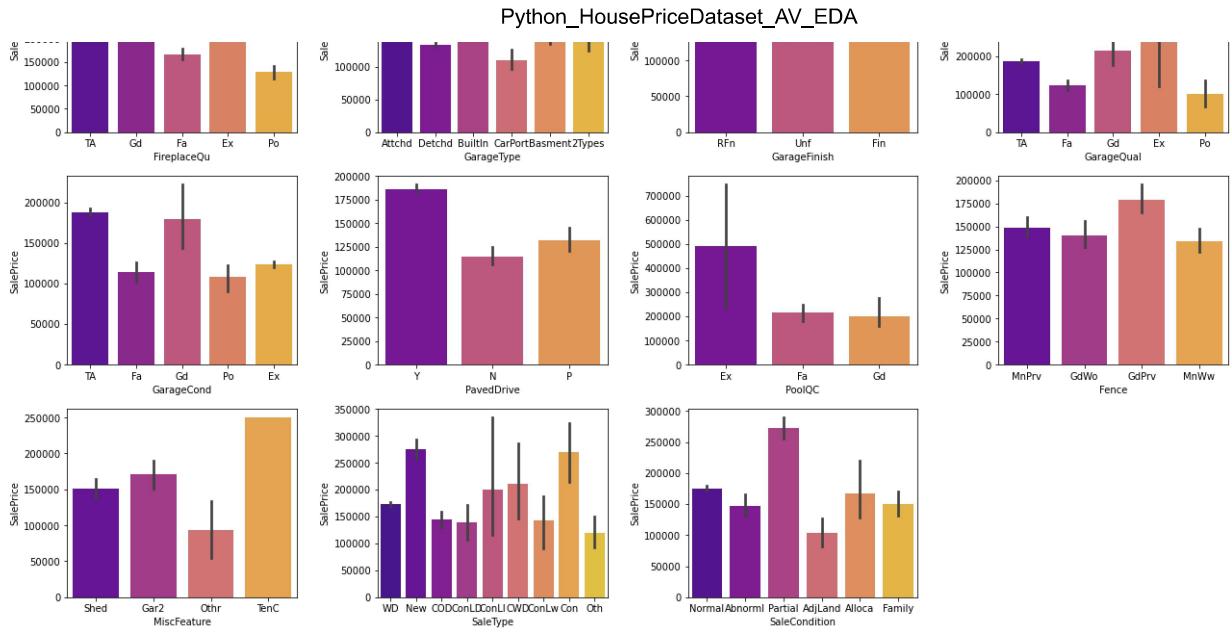
```
Out[20]: ['MSZoning',
 'Street',
 'Alley',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'PoolQC',
 'Fence',
 'MiscFeature',
 'SaleType',
 'SaleCondition']
```

```
In [21]: # Perform Bivariate analysis of Categorical features with target feature using "Barplot"

plt.figure(figsize= (18,50))
for i in range(0, len(categorical_features)):
    df1 = df.copy()
    plt.subplot(15, 4, i+1)
    sns.barplot(x = df1[categorical_features[i]], y ='SalePrice', data = df1, palette
    plt.tight_layout()
```

## Python\_HousePriceDataset\_AV\_EDA

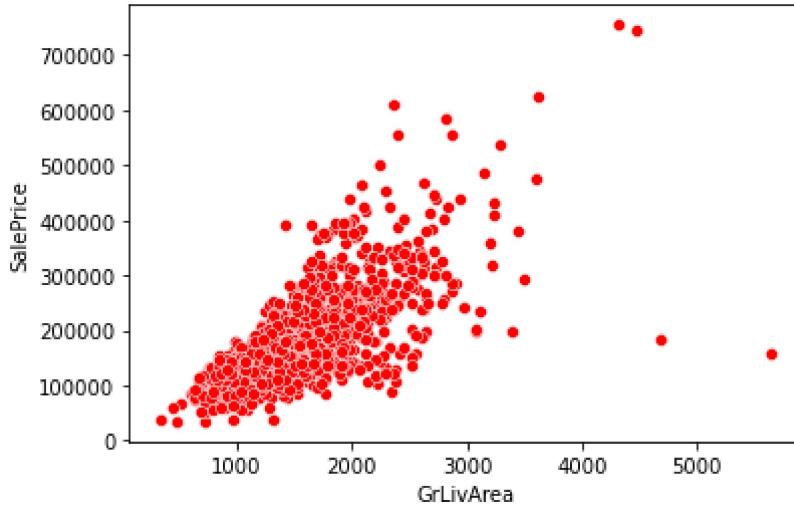




```
In [22]: # Just checking multicollinear features in order to find out whether or not drop any of them

for features in df.columns:
    df1 = df.copy()
    sns.scatterplot(x = 'GrLivArea', y = 'SalePrice', data = df1, color = 'r')
```

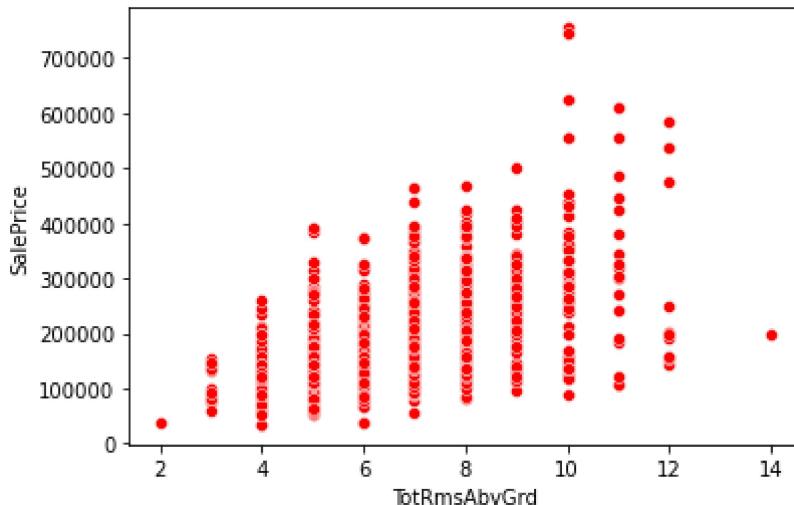
Out[22]: <AxesSubplot:xlabel='GrLivArea', ylabel='SalePrice'>



```
In [23]: # Just checking multicollinear features in order to find out whether or not drop any of them

for features in df.columns:
    df1 = df.copy()
    sns.scatterplot(x = 'TotRmsAbvGrd', y = 'SalePrice', data = df1, color = 'r')
```

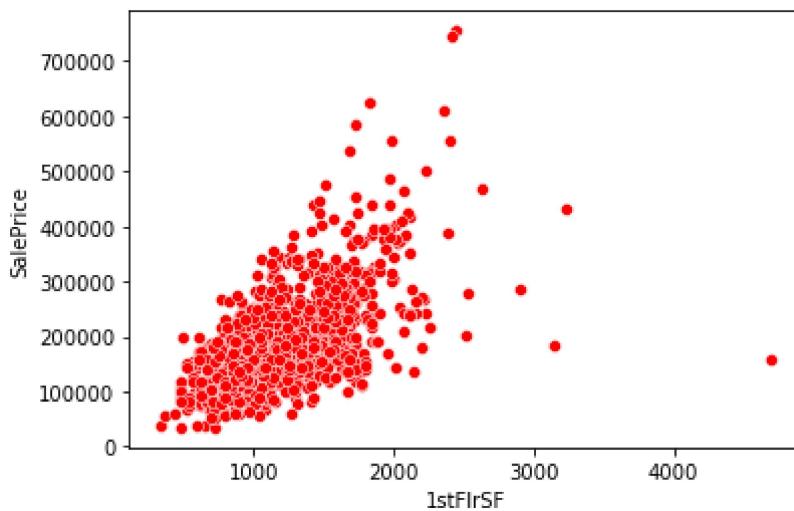
Out[23]: <AxesSubplot:xlabel='TotRmsAbvGrd', ylabel='SalePrice'>



**Observation : Both "TotRmsAbvGrd" and "GrLivArea" are highly correlated hence any one feature can be dropped**

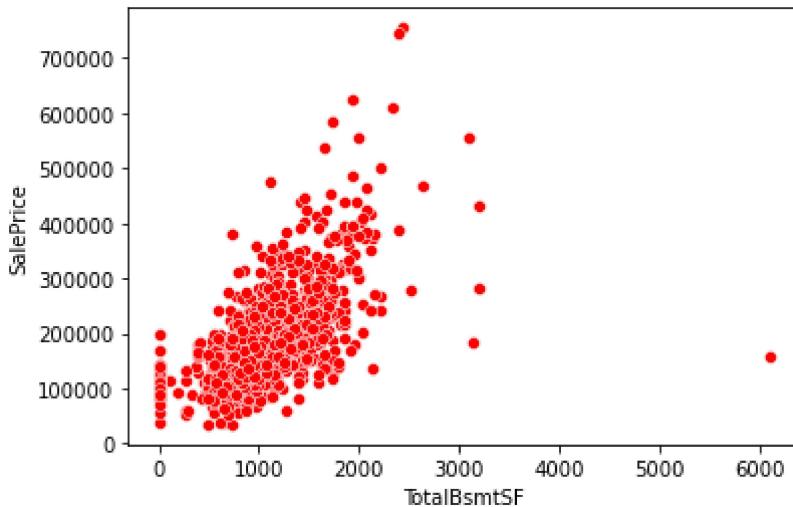
```
In [24]: # Just checking multicollinear features in order to find out whether or not drop any c
for features in df.columns:
    df1 = df.copy()
sns.scatterplot(x = '1stFlrSF', y = 'SalePrice', data = df1, color = 'r')
```

```
Out[24]: <AxesSubplot:xlabel='1stFlrSF', ylabel='SalePrice'>
```



```
In [25]: # Just checking multicollinear features in order to find out whether or not drop any c
for features in df.columns:
    df1 = df.copy()
sns.scatterplot(x = 'TotalBsmtSF', y = 'SalePrice', data = df1, color = 'r')
```

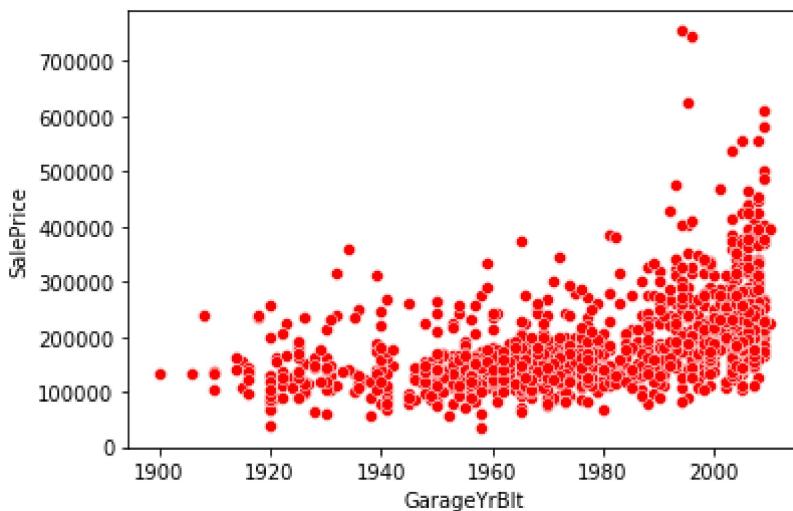
```
Out[25]: <AxesSubplot:xlabel='TotalBsmtSF', ylabel='SalePrice'>
```



**Observation : Both "TotalBsmtSF" and "1stFlrSF" are highly correlated hence any one feature can be dropped**

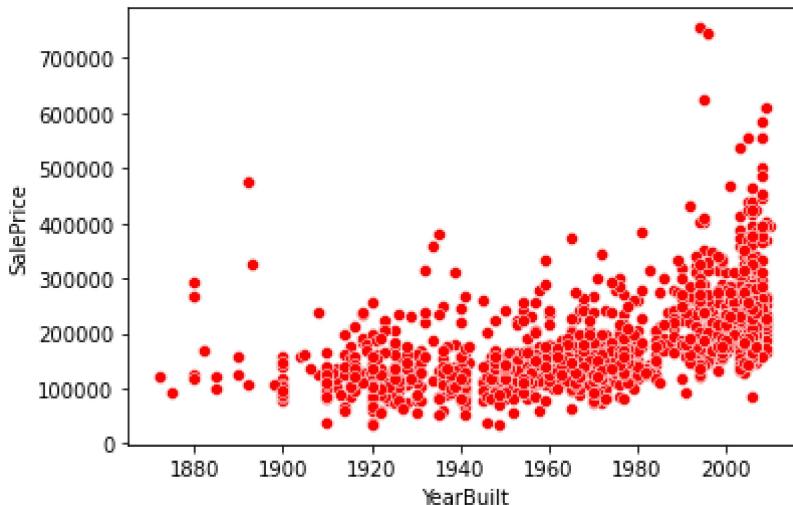
```
In [26]: # Just checking multicollinear features in order to find out whether or not drop any c
for features in df.columns:
    df1 = df.copy()
sns.scatterplot(x = 'GarageYrBlt', y = 'SalePrice', data = df1, color = 'r')
```

```
Out[26]: <AxesSubplot:xlabel='GarageYrBlt', ylabel='SalePrice'>
```



```
In [27]: # Just checking multicollinear features in order to find out whether or not drop any c
for features in df.columns:
    df1 = df.copy()
sns.scatterplot(x = 'YearBuilt', y = 'SalePrice', data = df1, color = 'r')
```

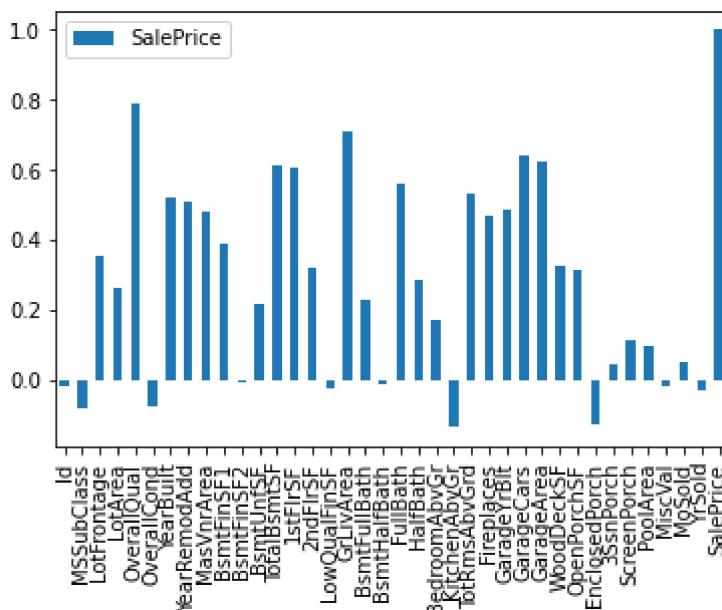
```
Out[27]: <AxesSubplot:xlabel='YearBuilt', ylabel='SalePrice'>
```



**Observation :** Both "YearBuilt" and "GarageYrBlt" are highly correlated hence any one feature can be dropped

```
In [28]: # Performing multicollinearity with target feature using bar plot
df.corr()[['SalePrice']].plot(kind = 'bar')
```

```
Out[28]: <AxesSubplot:>
```



## Treatment for Missing Values

```
In [29]: # Finding the number of missing values in each and every feature of the dataset
features_with_na = [feature for feature in df.columns if df[feature].isnull().sum()
for feature in features_with_na:
    df1 = df.copy()
    print(feature, np.round(df1[feature].isnull().sum(), 4))
```

```

LotFrontage 259
Alley 1369
MasVnrType 8
MasVnrArea 8
BsmtQual 37
BsmtCond 37
BsmtExposure 38
BsmtFinType1 37
BsmtFinType2 38
Electrical 1
FireplaceQu 690
GarageType 81
GarageYrBlt 81
GarageFinish 81
GarageQual 81
GarageCond 81
PoolQC 1453
Fence 1179
MiscFeature 1406

```

In [30]: # finding the categories in every feature to derive whether to use mean, median or mode

```

df['Electrical'].value_counts()
#df1.columns

```

Out[30]:

SBrkr	1334
FuseA	94
FuseF	27
FuseP	3
Mix	1

Name: Electrical, dtype: int64

In [31]: # Missing values treatment : used mode to fill the missing values for categorical variables

```

df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mode()[0])
df['Electrical'] = df['Electrical'].fillna(df['Electrical'].mode()[0])
df['MasVnrType'] = df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].mean())
df['BsmtQual'] = df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['BsmtCond'] = df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtFinType1'] = df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
df['BsmtFinType2'] = df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0])
df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
df['GarageYrBlt'] = df['GarageYrBlt'].fillna(df['GarageYrBlt'].mean())
df['GarageFinish'] = df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond'] = df['GarageCond'].fillna(df['GarageCond'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])

```

In [32]: # Again, after the missing values treatment validating if there are any more missing values

```

df.isnull().sum()

```

```
Out[32]:
```

Id	0
MSSubClass	0
MSZoning	0
LotFrontage	0
LotArea	0
..	
MoSold	0
YrSold	0
SaleType	0
SaleCondition	0
SalePrice	0

Length: 81, dtype: int64

```
In [33]: df.isnull().sum().sum()
```

```
Out[33]: 6097
```

```
In [34]: # Let's drop the features which has high missing values
```

```
df.drop('Alley', axis = 1, inplace = True)  
df.drop('Id', axis = 1, inplace = True)
```

```
In [35]: # Let's drop the features which has high missing values
```

```
df.drop('FireplaceQu', axis = 1, inplace = True)
```

```
In [36]: # Let's drop the features which has high missing values
```

```
df.drop('PoolQC', axis = 1, inplace = True)
```

```
In [37]: # Let's drop the features which has high missing values
```

```
df.drop('Fence', axis = 1, inplace = True)
```

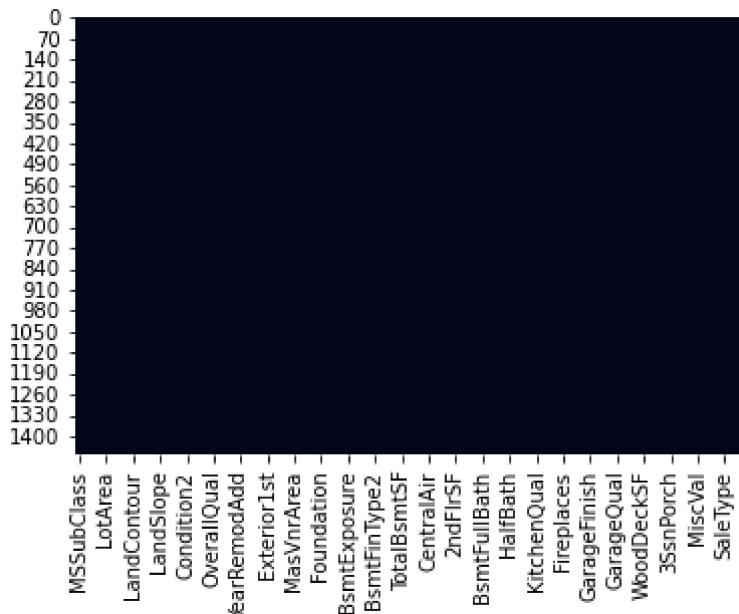
```
In [38]: # Let's drop the features which has high missing values
```

```
df.drop('MiscFeature', axis = 1, inplace = True)
```

```
In [39]: # Perform plotting to confirm that there are no missing values present in the dataset
```

```
sns.heatmap(df.isnull(), cbar = False)
```

```
Out[39]: <AxesSubplot:>
```



**Observation : All the missing values has been treated and there are no null values in the dataset**

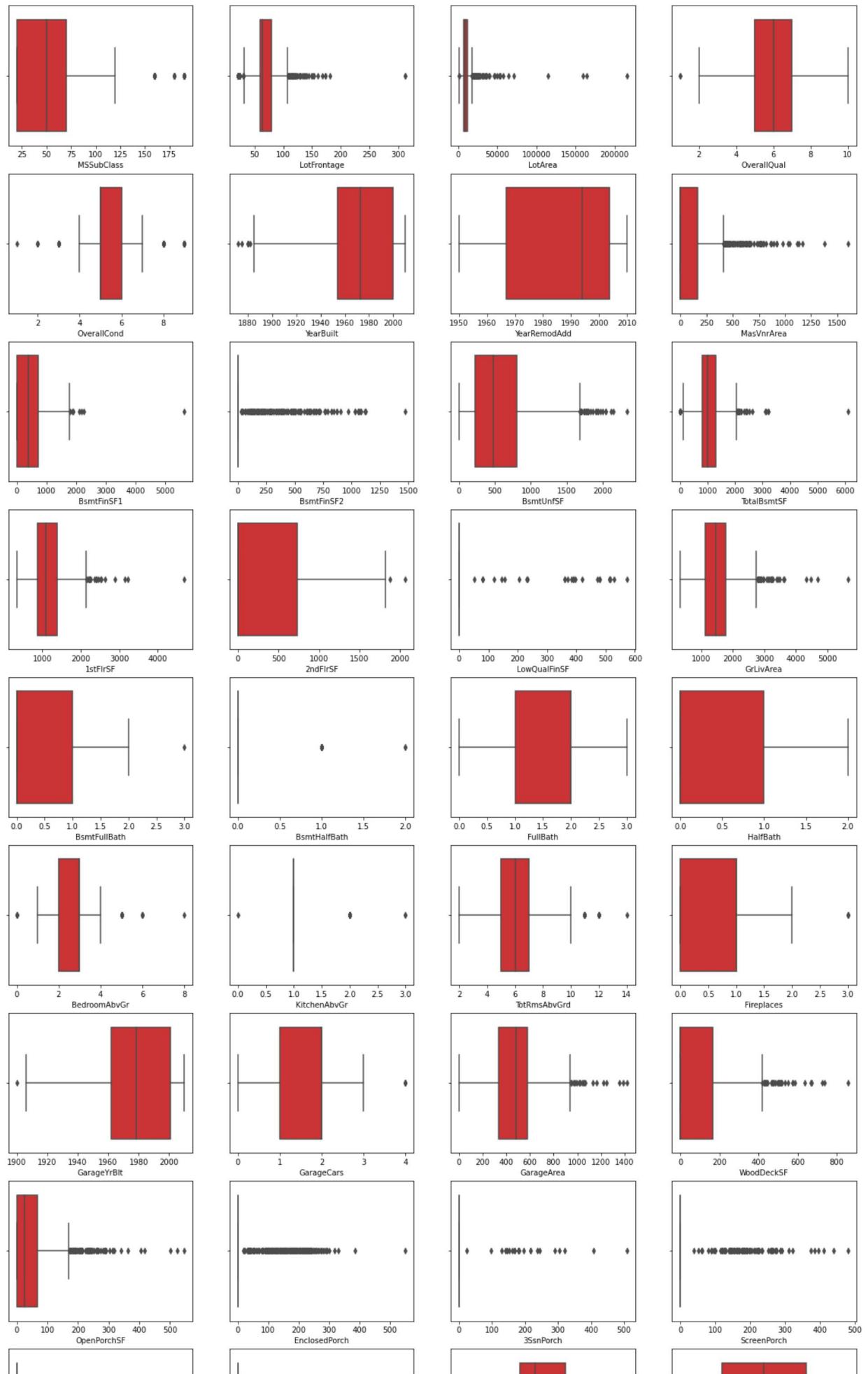
## Finding the Outliers for the Numerical features using Boxplot

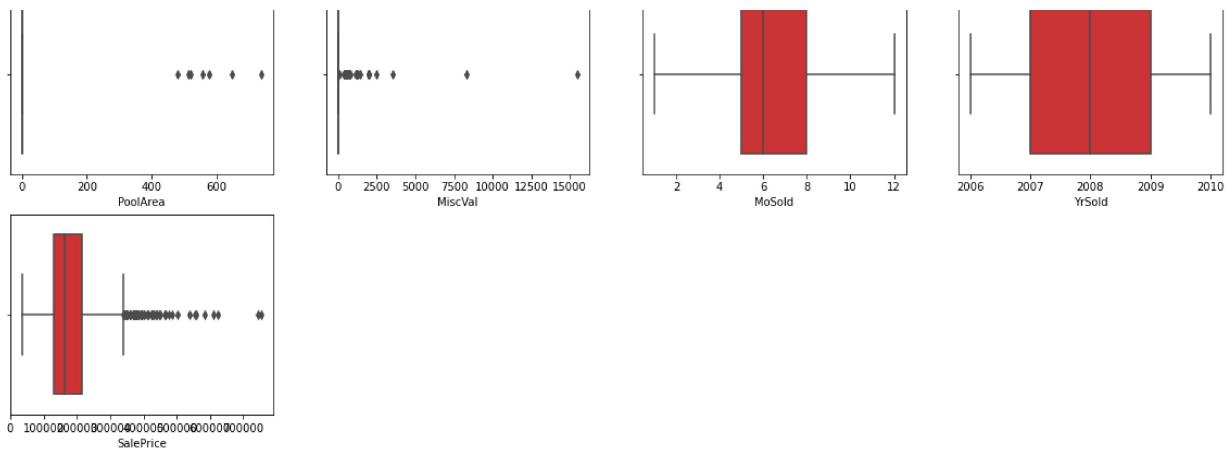
```
In [40]: len(Numerical_features)
```

```
Out[40]: 37
```

```
In [41]: # Finding the Outliers using Boxplot
```

```
plt.figure(figsize = (20,40))
for i in range(0, len(Numerical_features)):
    df1 = df.copy()
    plt.subplot(10, 4, i+1)
    sns.boxplot(x = df1[Numerical_features[i]], palette="Set1")
```





## Let us now use 5 Number summary for the Outliers

1. Minimum
2. 25th Percentile
3. Median
4. 75th Percentile
5. Maximum

```
In [42]: # Finding the Minimum value in numerical features
```

```
for features in Numerical_features:
    df1 = df.copy()
    print(features, np.min(df1[features]))
```

```
MSSubClass 20
LotFrontage 21.0
LotArea 1300
OverallQual 1
OverallCond 1
YearBuilt 1872
YearRemodAdd 1950
MasVnrArea 0.0
BsmtFinSF1 0
BsmtFinSF2 0
BsmtUnfSF 0
TotalBsmtSF 0
1stFlrSF 334
2ndFlrSF 0
LowQualFinSF 0
GrLivArea 334
BsmtFullBath 0
BsmtHalfBath 0
FullBath 0
HalfBath 0
BedroomAbvGr 0
KitchenAbvGr 0
TotRmsAbvGrd 2
Fireplaces 0
GarageYrBlt 1900.0
GarageCars 0
GarageArea 0
WoodDeckSF 0
OpenPorchSF 0
EnclosedPorch 0
3SsnPorch 0
ScreenPorch 0
PoolArea 0
MiscVal 0
MoSold 1
YrSold 2006
SalePrice 34900
```

```
In [43]: # Let's find 25th, 75th Percentile of Numerical features
```

```
for features in Numerical_features:
    df1 = df.copy()
    print(features, "Q1 Percentile is : ", np.percentile(df[features], [25]))
    print(features, "Q3 Percentile is : ", np.percentile(df[features], [75]), "\n")
```

```
MSSubClass Q1 Percentile is : [20.]
MSSubClass Q3 Percentile is : [70.]

LotFrontage Q1 Percentile is : [60.]
LotFrontage Q3 Percentile is : [79.]

LotArea Q1 Percentile is : [7553.5]
LotArea Q3 Percentile is : [11601.5]

OverallQual Q1 Percentile is : [5.]
OverallQual Q3 Percentile is : [7.]

OverallCond Q1 Percentile is : [5.]
OverallCond Q3 Percentile is : [6.]

YearBuilt Q1 Percentile is : [1954.]
YearBuilt Q3 Percentile is : [2000.>

YearRemodAdd Q1 Percentile is : [1967.]
YearRemodAdd Q3 Percentile is : [2004.>

MasVnrArea Q1 Percentile is : [0.]
MasVnrArea Q3 Percentile is : [164.25]

BsmtFinSF1 Q1 Percentile is : [0.]
BsmtFinSF1 Q3 Percentile is : [712.25]

BsmtFinSF2 Q1 Percentile is : [0.]
BsmtFinSF2 Q3 Percentile is : [0.>

BsmtUnfSF Q1 Percentile is : [223.]
BsmtUnfSF Q3 Percentile is : [808.>

TotalBsmtSF Q1 Percentile is : [795.75]
TotalBsmtSF Q3 Percentile is : [1298.25]

1stFlrSF Q1 Percentile is : [882.]
1stFlrSF Q3 Percentile is : [1391.25]

2ndFlrSF Q1 Percentile is : [0.]
2ndFlrSF Q3 Percentile is : [728.>

LowQualFinSF Q1 Percentile is : [0.]
LowQualFinSF Q3 Percentile is : [0.>

GrLivArea Q1 Percentile is : [1129.5]
GrLivArea Q3 Percentile is : [1776.75]

BsmtFullBath Q1 Percentile is : [0.]
BsmtFullBath Q3 Percentile is : [1.>

BsmtHalfBath Q1 Percentile is : [0.]
BsmtHalfBath Q3 Percentile is : [0.>

FullBath Q1 Percentile is : [1.]
FullBath Q3 Percentile is : [2.>

HalfBath Q1 Percentile is : [0.]
HalfBath Q3 Percentile is : [1.]
```

```
BedroomAbvGr Q1 Percentile is : [2.]
BedroomAbvGr Q3 Percentile is : [3.]

KitchenAbvGr Q1 Percentile is : [1.]
KitchenAbvGr Q3 Percentile is : [1.1]

TotRmsAbvGrd Q1 Percentile is : [5.]
TotRmsAbvGrd Q3 Percentile is : [7.1]

Fireplaces Q1 Percentile is : [0.]
Fireplaces Q3 Percentile is : [1.1]

GarageYrBlt Q1 Percentile is : [1962.]
GarageYrBlt Q3 Percentile is : [2001.1]

GarageCars Q1 Percentile is : [1.]
GarageCars Q3 Percentile is : [2.1]

GarageArea Q1 Percentile is : [334.5]
GarageArea Q3 Percentile is : [576.1]

WoodDeckSF Q1 Percentile is : [0.]
WoodDeckSF Q3 Percentile is : [168.1]

OpenPorchSF Q1 Percentile is : [0.]
OpenPorchSF Q3 Percentile is : [68.1]

EnclosedPorch Q1 Percentile is : [0.]
EnclosedPorch Q3 Percentile is : [0.1]

3SsnPorch Q1 Percentile is : [0.]
3SsnPorch Q3 Percentile is : [0.1]

ScreenPorch Q1 Percentile is : [0.]
ScreenPorch Q3 Percentile is : [0.1]

PoolArea Q1 Percentile is : [0.]
PoolArea Q3 Percentile is : [0.1]

MiscVal Q1 Percentile is : [0.]
MiscVal Q3 Percentile is : [0.1]

MoSold Q1 Percentile is : [5.]
MoSold Q3 Percentile is : [8.1]

YrSold Q1 Percentile is : [2007.]
YrSold Q3 Percentile is : [2009.1]

SalePrice Q1 Percentile is : [129975.]
SalePrice Q3 Percentile is : [214000.]
```

```
In [44]: # Finding the median of all numerical features
```

```
for features in Numerical_features:
    df1 = df.copy()
    print(features, "Median Value is : ", np.median(df1[features]), "\n")
```

```
MSSubClass Median Value is : 50.0
LotFrontage Median Value is : 63.0
LotArea Median Value is : 9478.5
OverallQual Median Value is : 6.0
OverallCond Median Value is : 5.0
YearBuilt Median Value is : 1973.0
YearRemodAdd Median Value is : 1994.0
MasVnrArea Median Value is : 0.0
BsmtFinSF1 Median Value is : 383.5
BsmtFinSF2 Median Value is : 0.0
BsmtUnfSF Median Value is : 477.5
TotalBsmtSF Median Value is : 991.5
1stFlrSF Median Value is : 1087.0
2ndFlrSF Median Value is : 0.0
LowQualFinSF Median Value is : 0.0
GrLivArea Median Value is : 1464.0
BsmtFullBath Median Value is : 0.0
BsmtHalfBath Median Value is : 0.0
FullBath Median Value is : 2.0
HalfBath Median Value is : 0.0
BedroomAbvGr Median Value is : 3.0
KitchenAbvGr Median Value is : 1.0
TotRmsAbvGrd Median Value is : 6.0
Fireplaces Median Value is : 1.0
GarageYrBlt Median Value is : 1978.5061638868744
GarageCars Median Value is : 2.0
GarageArea Median Value is : 480.0
WoodDeckSF Median Value is : 0.0
OpenPorchSF Median Value is : 25.0
EnclosedPorch Median Value is : 0.0
```

```
3SsnPorch Median Value is :  0.0
ScreenPorch Median Value is :  0.0
PoolArea Median Value is :  0.0
MiscVal Median Value is :  0.0
MoSold Median Value is :  6.0
YrSold Median Value is :  2008.0
SalePrice Median Value is :  163000.0
```

```
In [45]: # Created a function to find the 5 number summary such as q1, q3, IQR, Lower_fence, Higher_fence

def outliers_detection(self):
    df1 = df.copy()
    q1 = self.quantile(0.25)
    q3 = self.quantile(0.75)
    IQR = q3-q1
    lower_fence = q1 - (1.5*IQR)
    higher_fence = q3 + (1.5*IQR)
    return IQR, lower_fence, higher_fence
```

```
In [46]: # Testing the function on one feature whether it is working fine or not

outliers_detection(df1['BsmtUnfSF'])
```

```
Out[46]: (585.0, -654.5, 1685.5)
```

```
In [47]: # Applying the function to call 5 number summary on all the features

for features in Numerical_features:
    df1 = df.copy()
    print(features, outliers_detection(df1[features]))
```

```
MSSubClass (50.0, -55.0, 145.0)
LotFrontage (19.0, 31.5, 107.5)
LotArea (4048.0, 1481.5, 17673.5)
OverallQual (2.0, 2.0, 10.0)
OverallCond (1.0, 3.5, 7.5)
YearBuilt (46.0, 1885.0, 2069.0)
YearRemodAdd (37.0, 1911.5, 2059.5)
MasVnrArea (164.25, -246.375, 410.625)
BsmtFinSF1 (712.25, -1068.375, 1780.625)
BsmtFinSF2 (0.0, 0.0, 0.0)
BsmtUnfSF (585.0, -654.5, 1685.5)
TotalBsmtSF (502.5, 42.0, 2052.0)
1stFlrSF (509.25, 118.125, 2155.125)
2ndFlrSF (728.0, -1092.0, 1820.0)
LowQualFinSF (0.0, 0.0, 0.0)
GrLivArea (647.25, 158.625, 2747.625)
BsmtFullBath (1.0, -1.5, 2.5)
BsmtHalfBath (0.0, 0.0, 0.0)
FullBath (1.0, -0.5, 3.5)
HalfBath (1.0, -1.5, 2.5)
BedroomAbvGr (1.0, 0.5, 4.5)
KitchenAbvGr (0.0, 1.0, 1.0)
TotRmsAbvGrd (2.0, 2.0, 10.0)
Fireplaces (1.0, -1.5, 2.5)
GarageYrBlt (39.0, 1903.5, 2059.5)
GarageCars (1.0, -0.5, 3.5)
GarageArea (241.5, -27.75, 938.25)
WoodDeckSF (168.0, -252.0, 420.0)
OpenPorchSF (68.0, -102.0, 170.0)
EnclosedPorch (0.0, 0.0, 0.0)
3SsnPorch (0.0, 0.0, 0.0)
ScreenPorch (0.0, 0.0, 0.0)
PoolArea (0.0, 0.0, 0.0)
MiscVal (0.0, 0.0, 0.0)
MoSold (3.0, 0.5, 12.5)
YrSold (2.0, 2004.0, 2012.0)
SalePrice (84025.0, 3937.5, 340037.5)
```

In [48]: # 1st type of outlier function : Created a function to find the outliers above the upper bound

```
def outliers_handler_fit(df1, feature_name):
    q1=df1.loc[:,[feature_name]].quantile(0.25)
    q3=df1.loc[:,[feature_name]].quantile(0.75)
    IQR = q3-q1
    lower = q1 - (1.5*IQR)
    upper = q3 + (1.5*IQR)
    df1.loc[df[feature_name]<=lower[0],feature_name]=lower[0]
    df1.loc[df[feature_name]>=upper[0],feature_name]=upper[0]
    return df1
```

In [49]: # 2nd type of Outlier function : Created a function to find the outliers above the upper bound

```
def outlier_treatment(df1, feature_name):
    q1 = df1.loc[:, [feature_name]].quantile(0.25)
    q3 = df1.loc[:, [feature_name]].quantile(0.75)
    IQR = q3-q1
    lower_fence = q1 - (1.5*IQR)
    higher_fence = q3 + (1.5*IQR)
```

```
    return IQR, lower_fence, higher_fence
```

```
In [50]: outlier_treatment(df1, 'LotFrontage')
```

```
Out[50]: (LotFrontage    19.0
          dtype: float64,
          LotFrontage    31.5
          dtype: float64,
          LotFrontage   107.5
          dtype: float64)
```

```
In [51]: # 3rd type of Outlier function : Created a function to find the outliers above the upper
```

```
def detect_outliers(col):

    # Finding the IQR

    percentile25 = df1[col].quantile(0.25)

    percentile75 = df1[col].quantile(0.75)

    print('\n #####', col , '#####')

    print("percentile25",percentile25)

    print("percentile75",percentile75)

    iqr = percentile75 - percentile25

    higher_fence = percentile75 + 1.5 * iqr

    lower_fence = percentile25 - 1.5 * iqr

    print("higher_fence",higher_fence)

    print("lower_fence",lower_fence)

    df1.loc[(df1[col]>higher_fence), col]= higher_fence

    df1.loc[(df1[col]<lower_fence), col]= lower_fence

    return df1
```

```
In [52]: # Applying the function on one of the feature called "LotArea"
```

```
detect_outliers('LotArea')
```

```
##### LotArea #####
percentile25 7553.5
percentile75 11601.5
higher_fence 17673.5
lower_fence 1481.5
```

Out[52]:

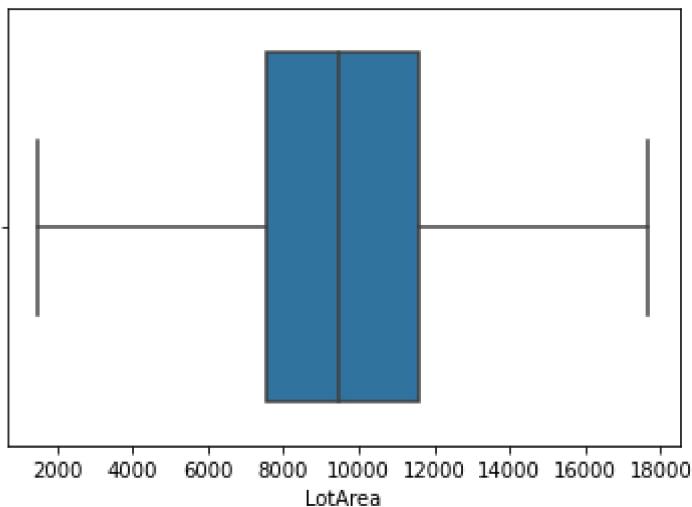
	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig
0	60	RL	65.0	8450.0	Pave	Reg		AllPub	Imp
1	20	RL	80.0	9600.0	Pave	Reg		AllPub	Imp
2	60	RL	68.0	11250.0	Pave	IR1		AllPub	Imp
3	70	RL	60.0	9550.0	Pave	IR1		AllPub	Con
4	60	RL	84.0	14260.0	Pave	IR1		AllPub	Imp
...	...	...	...	...	...	...	...	...	...
1455	60	RL	62.0	7917.0	Pave	Reg		AllPub	Imp
1456	20	RL	85.0	13175.0	Pave	Reg		AllPub	Imp
1457	70	RL	66.0	9042.0	Pave	Reg		AllPub	Imp
1458	20	RL	68.0	9717.0	Pave	Reg		AllPub	Imp
1459	20	RL	75.0	9937.0	Pave	Reg		AllPub	Imp

1460 rows × 75 columns

In [53]: # After applying the function on "LotArea", validating the outliers for "LotArea" using boxplot

```
sns.boxplot(x = df1['LotArea'])
```

Out[53]: &lt;AxesSubplot:xlabel='LotArea'&gt;

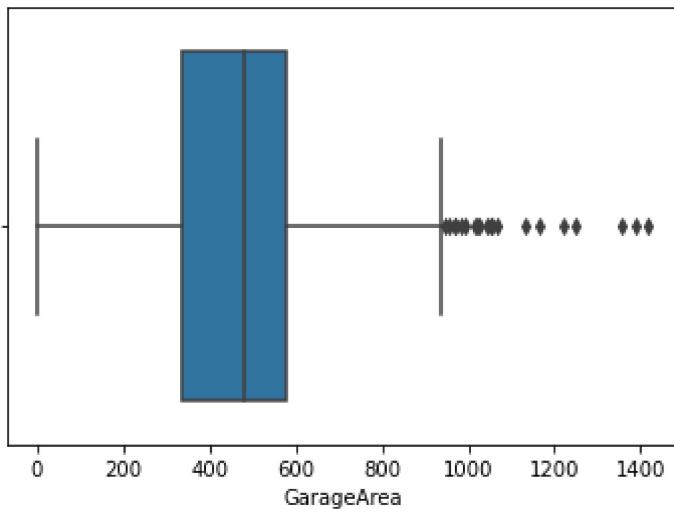


In [54]: # Checking for outliers on feature called "GarageArea" before applying the outlier function

```
sns.boxplot(df1['GarageArea'])
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. 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.  
warnings.warn(

Out[54]: &lt;AxesSubplot:xlabel='GarageArea'&gt;



```
In [55]: # Now, applied the outlier function
```

```
detect_outliers('GarageArea')
```

```
#### GarageArea ####
percentile25 334.5
percentile75 576.0
higher_fence 938.25
lower_fence -27.75
```

```
Out[55]:
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotC
<b>0</b>	60	RL	65.0	8450.0	Pave	Reg		Lvl	AllPub
<b>1</b>	20	RL	80.0	9600.0	Pave	Reg		Lvl	AllPub
<b>2</b>	60	RL	68.0	11250.0	Pave	IR1		Lvl	AllPub
<b>3</b>	70	RL	60.0	9550.0	Pave	IR1		Lvl	AllPub
<b>4</b>	60	RL	84.0	14260.0	Pave	IR1		Lvl	AllPub
...	...	...	...	...	...	...	...	...	...
<b>1455</b>	60	RL	62.0	7917.0	Pave	Reg		Lvl	AllPub
<b>1456</b>	20	RL	85.0	13175.0	Pave	Reg		Lvl	AllPub
<b>1457</b>	70	RL	66.0	9042.0	Pave	Reg		Lvl	AllPub
<b>1458</b>	20	RL	68.0	9717.0	Pave	Reg		Lvl	AllPub
<b>1459</b>	20	RL	75.0	9937.0	Pave	Reg		Lvl	AllPub

1460 rows × 75 columns

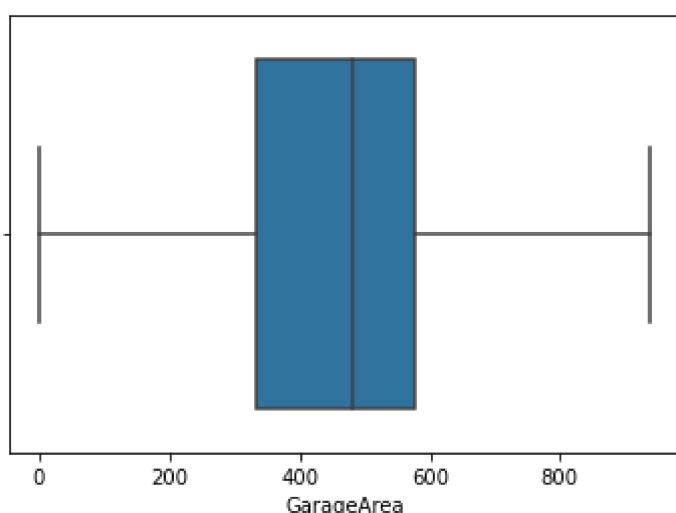
```
In [56]: # Post applying the outlier function above, validating with the boxplot
```

```
sns.boxplot(df1['GarageArea'])
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

```
warnings.warn(
```

```
Out[56]: <AxesSubplot:xlabel='GarageArea'>
```



```
In [57]: # Now, In order to manipulate the outliers, applying the functions to each and every fe
```

```
detect_outliers('OverallQual')  
detect_outliers('OverallCond')  
detect_outliers('YearBuilt')  
detect_outliers('MasVnrArea')  
detect_outliers('BsmtFinSF1')  
detect_outliers('BsmtFinSF2')  
detect_outliers('BsmtUnfSF')  
detect_outliers('1stFlrSF')  
detect_outliers('2ndFlrSF')  
detect_outliers('LowQualFinsF')  
detect_outliers('GrLivArea')  
detect_outliers('BsmtFullBath')  
detect_outliers('BsmtHalfBath')  
detect_outliers('BedroomAbvGr')  
detect_outliers('KitchenAbvGr')  
detect_outliers('TotRmsAbvGrd')  
detect_outliers('Fireplaces')  
detect_outliers('GarageYrBlt')  
detect_outliers('GarageArea')  
detect_outliers('WoodDeckSF')  
detect_outliers('OpenPorchSF')  
detect_outliers('EnclosedPorch')  
detect_outliers('3SsnPorch')  
detect_outliers('ScreenPorch')  
detect_outliers('PoolArea')  
detect_outliers('MiscVal')
```

```
##### OverallQual #####
percentile25 5.0
percentile75 7.0
higher_fence 10.0
lower_fence 2.0

##### OverallCond #####
percentile25 5.0
percentile75 6.0
higher_fence 7.5
lower_fence 3.5

##### YearBuilt #####
percentile25 1954.0
percentile75 2000.0
higher_fence 2069.0
lower_fence 1885.0

##### MasVnrArea #####
percentile25 0.0
percentile75 164.25
higher_fence 410.625
lower_fence -246.375

##### BsmtFinSF1 #####
percentile25 0.0
percentile75 712.25
higher_fence 1780.625
lower_fence -1068.375

##### BsmtFinSF2 #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0

##### BsmtUnfSF #####
percentile25 223.0
percentile75 808.0
higher_fence 1685.5
lower_fence -654.5

##### 1stFlrSF #####
percentile25 882.0
percentile75 1391.25
higher_fence 2155.125
lower_fence 118.125

##### 2ndFlrSF #####
percentile25 0.0
percentile75 728.0
higher_fence 1820.0
lower_fence -1092.0

##### LowQualFinSF #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

```
##### GrLivArea #####
percentile25 1129.5
percentile75 1776.75
higher_fence 2747.625
lower_fence 158.625

##### BsmtFullBath #####
percentile25 0.0
percentile75 1.0
higher_fence 2.5
lower_fence -1.5

##### BsmtHalfBath #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0

##### BedroomAbvGr #####
percentile25 2.0
percentile75 3.0
higher_fence 4.5
lower_fence 0.5

##### KitchenAbvGr #####
percentile25 1.0
percentile75 1.0
higher_fence 1.0
lower_fence 1.0

##### TotRmsAbvGrd #####
percentile25 5.0
percentile75 7.0
higher_fence 10.0
lower_fence 2.0

##### Fireplaces #####
percentile25 0.0
percentile75 1.0
higher_fence 2.5
lower_fence -1.5

##### GarageYrBlt #####
percentile25 1962.0
percentile75 2001.0
higher_fence 2059.5
lower_fence 1903.5

##### GarageArea #####
percentile25 334.5
percentile75 576.0
higher_fence 938.25
lower_fence -27.75

##### WoodDeckSF #####
percentile25 0.0
percentile75 168.0
higher_fence 420.0
lower_fence -252.0
```

```
#### OpenPorchSF #####
percentile25 0.0
percentile75 68.0
higher_fence 170.0
lower_fence -102.0
```

```
#### EnclosedPorch #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

```
#### 3SsnPorch #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

```
#### ScreenPorch #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

```
#### PoolArea #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

```
#### MiscVal #####
percentile25 0.0
percentile75 0.0
higher_fence 0.0
lower_fence 0.0
```

Out[57]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotC
<b>0</b>	60	RL	65.0	8450.0	Pave	Reg		Lvl	AllPub
<b>1</b>	20	RL	80.0	9600.0	Pave	Reg		Lvl	AllPub
<b>2</b>	60	RL	68.0	11250.0	Pave	IR1		Lvl	AllPub
<b>3</b>	70	RL	60.0	9550.0	Pave	IR1		Lvl	AllPub
<b>4</b>	60	RL	84.0	14260.0	Pave	IR1		Lvl	AllPub
...	...	...	...	...	...	...	...	...	...
<b>1455</b>	60	RL	62.0	7917.0	Pave	Reg		Lvl	AllPub
<b>1456</b>	20	RL	85.0	13175.0	Pave	Reg		Lvl	AllPub
<b>1457</b>	70	RL	66.0	9042.0	Pave	Reg		Lvl	AllPub
<b>1458</b>	20	RL	68.0	9717.0	Pave	Reg		Lvl	AllPub
<b>1459</b>	20	RL	75.0	9937.0	Pave	Reg		Lvl	AllPub

1460 rows × 75 columns

```
In [58]: # Let us find only the numerical feature from my transformed dataset called "df1"
```

```
df1.describe().columns
```

```
Out[58]: Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
       'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
       'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
       'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

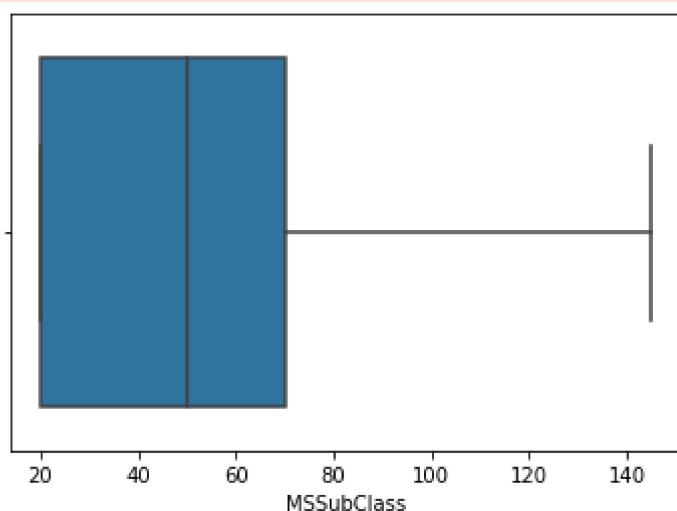
```
In [59]: # Let us now apply the outlier function for all features and BXplotting the same
```

```
for features in df1.describe().columns:
    detect_outliers(features)
    sns.boxplot(df1[features])
    plt.show()
```

```
#### MSSubClass ####
percentile25 20.0
percentile75 70.0
higher_fence 145.0
lower_fence -55.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```

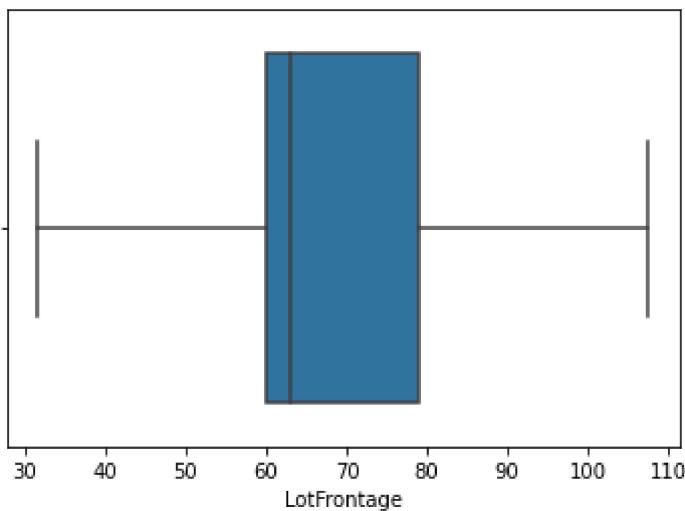


```
#### LotFrontage ####
```

```
percentile25 60.0
percentile75 79.0
higher_fence 107.5
lower_fence 31.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

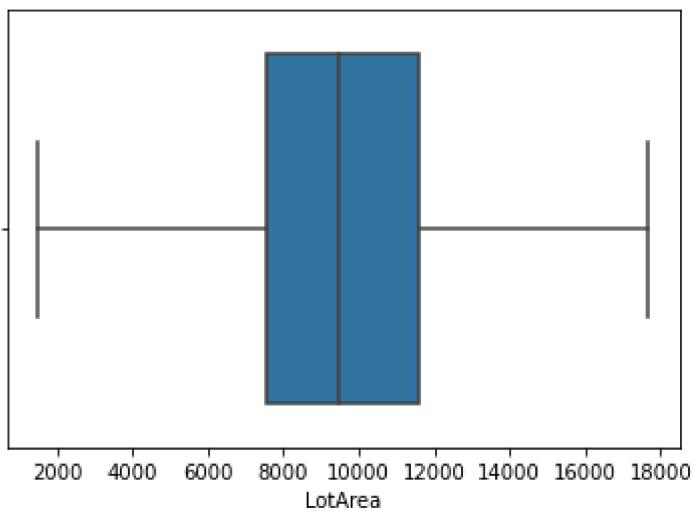
```
warnings.warn(
```



```
#### LotArea ####  
percentile25 7553.5  
percentile75 11601.5  
higher_fence 17673.5  
lower_fence 14815.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

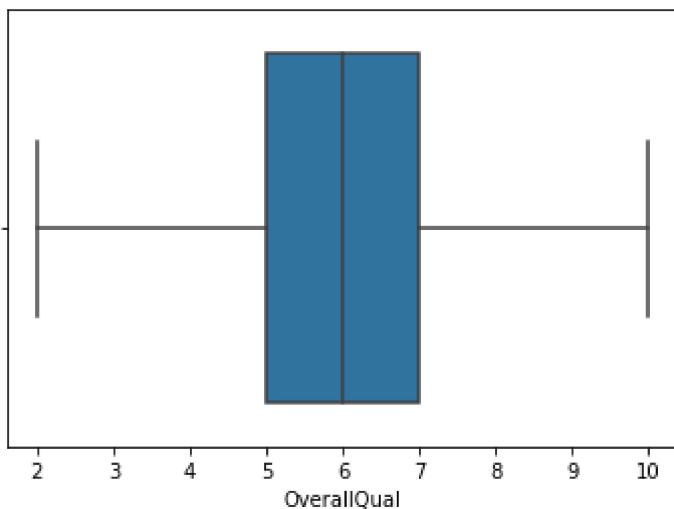
```
warnings.warn(
```



C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```

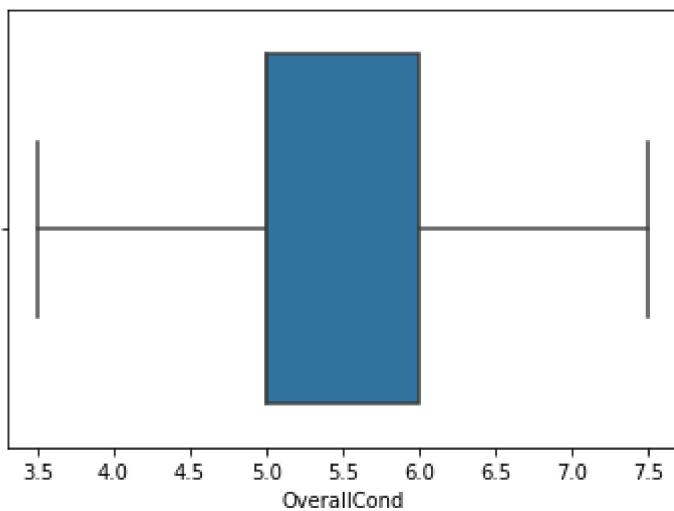
```
#### OverallQual ####  
percentile25 5.0  
percentile75 7.0  
higher_fence 10.0  
lower_fence 2.0
```



```
#### OverallCond ####  
percentile25 5.0  
percentile75 6.0  
higher_fence 7.5  
lower_fence 3.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

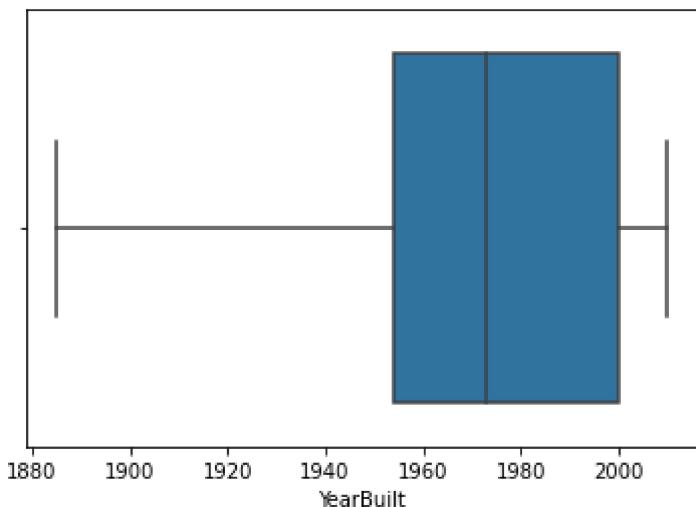
```
warnings.warn(
```



```
#### YearBuilt ####  
percentile25 1954.0  
percentile75 2000.0  
higher_fence 2069.0  
lower_fence 1885.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

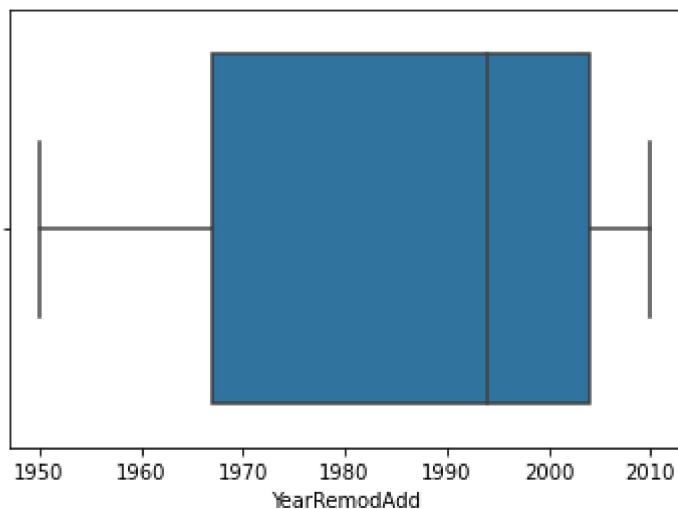
```
warnings.warn(
```



```
#### YearRemodAdd ####
percentile25 1967.0
percentile75 2004.0
higher_fence 2059.5
lower_fence 1911.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

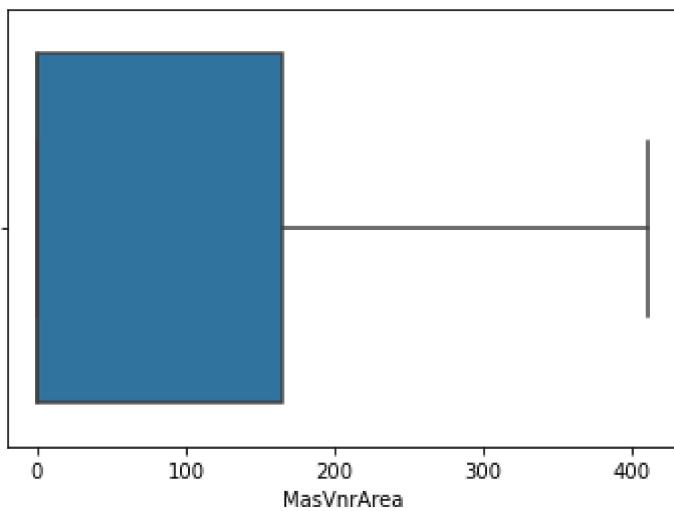
```
warnings.warn(
```



```
#### MasVnrArea ####
percentile25 0.0
percentile75 164.25
higher_fence 410.625
lower_fence -246.375
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

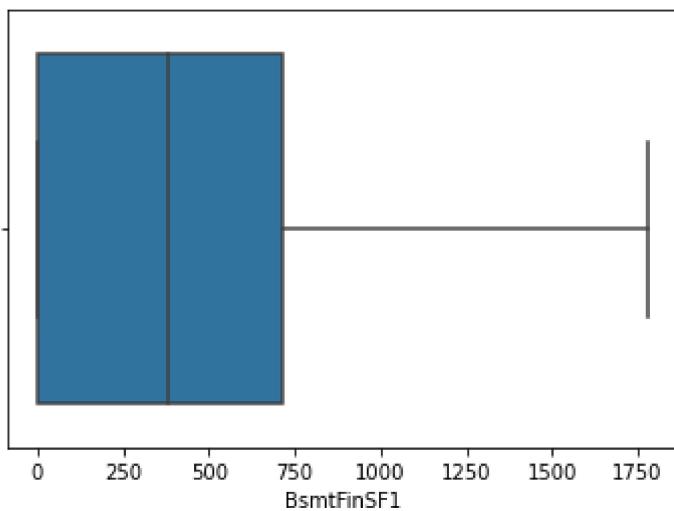
```
warnings.warn(
```



```
#### BsmtFinSF1 ####  
percentile25 0.0  
percentile75 712.25  
higher_fence 1780.625  
lower_fence -1068.375
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

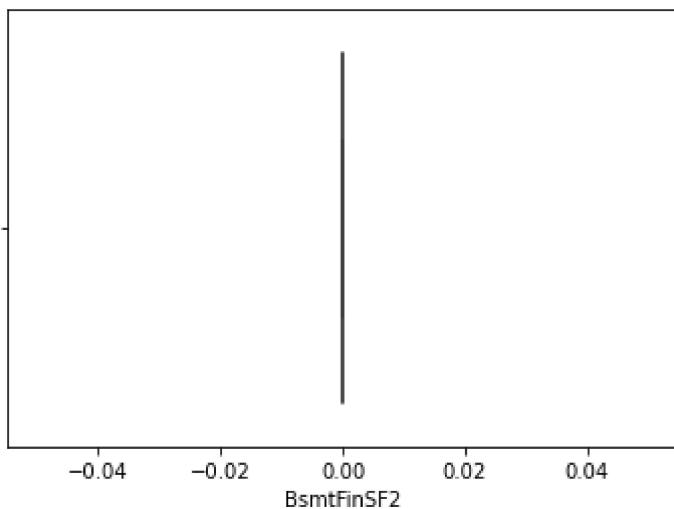
```
warnings.warn(
```



```
#### BsmtFinSF2 ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

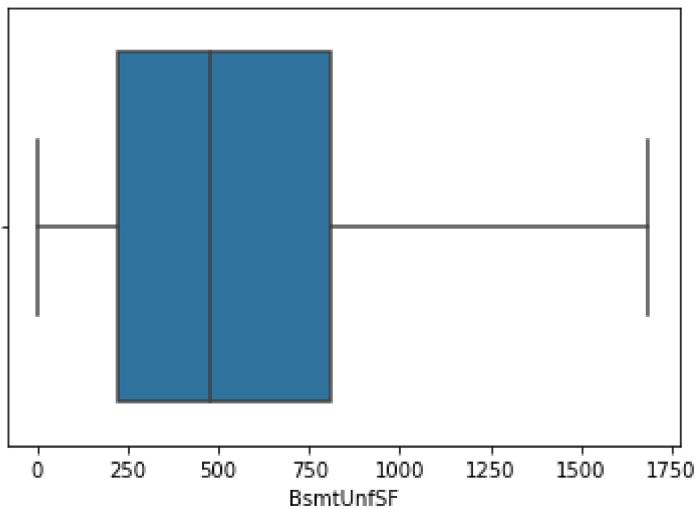
```
warnings.warn(
```



```
#### BsmtUnfSF ####  
percentile25 223.0  
percentile75 808.0  
higher_fence 1685.5  
lower_fence -654.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

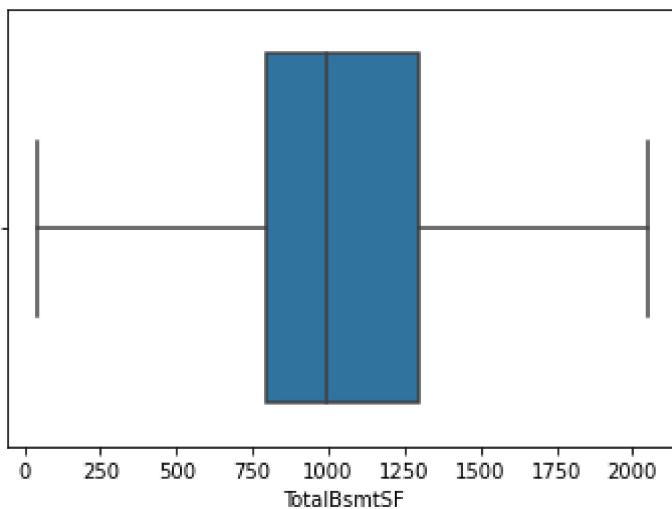
```
warnings.warn(
```



```
#### TotalBsmtSF ####  
percentile25 795.75  
percentile75 1298.25  
higher_fence 2052.0  
lower_fence 42.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

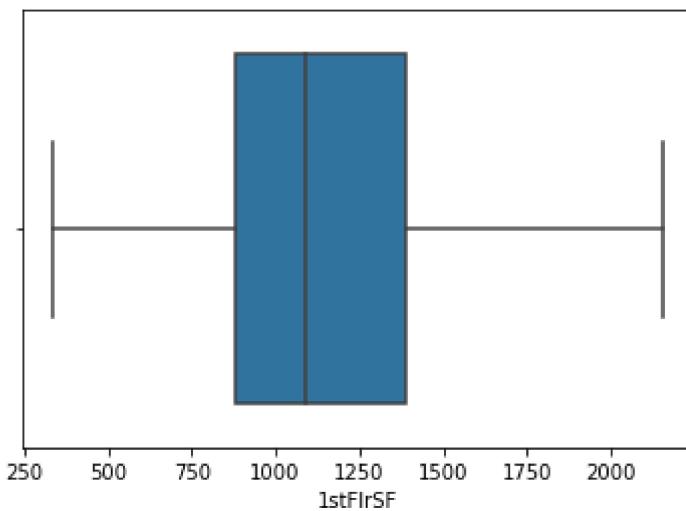
```
warnings.warn(
```



```
#### 1stFlrSF ####  
percentile25 882.0  
percentile75 1391.25  
higher_fence 2155.125  
lower_fence 118.125
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

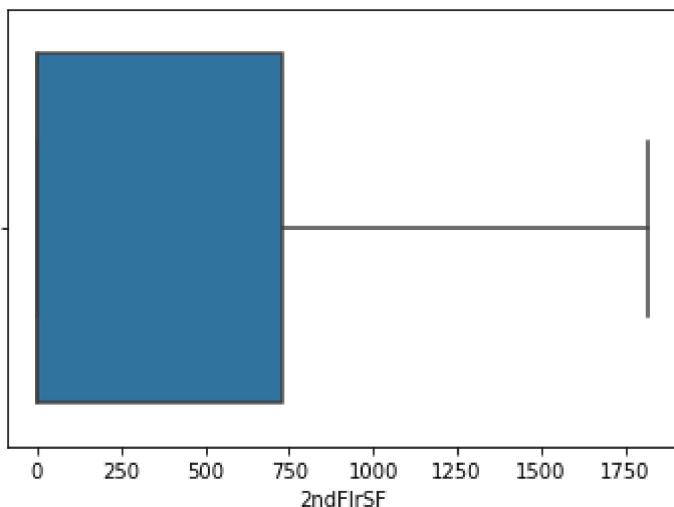
```
warnings.warn(
```



```
#### 2ndFlrSF ####  
percentile25 0.0  
percentile75 728.0  
higher_fence 1820.0  
lower_fence -1092.0
```

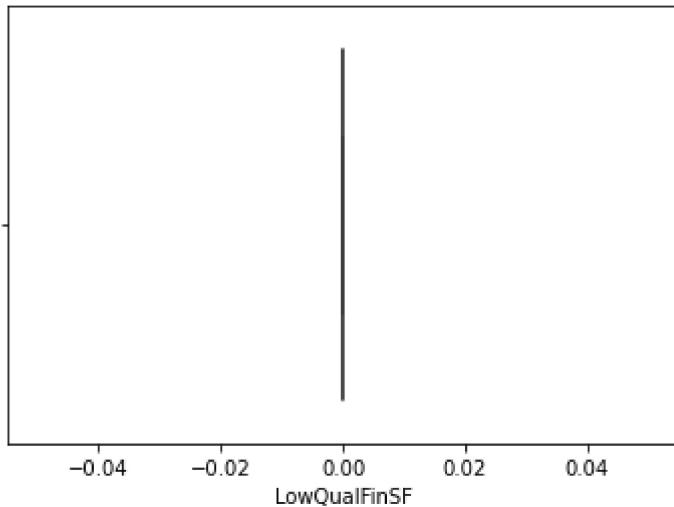
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```



```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

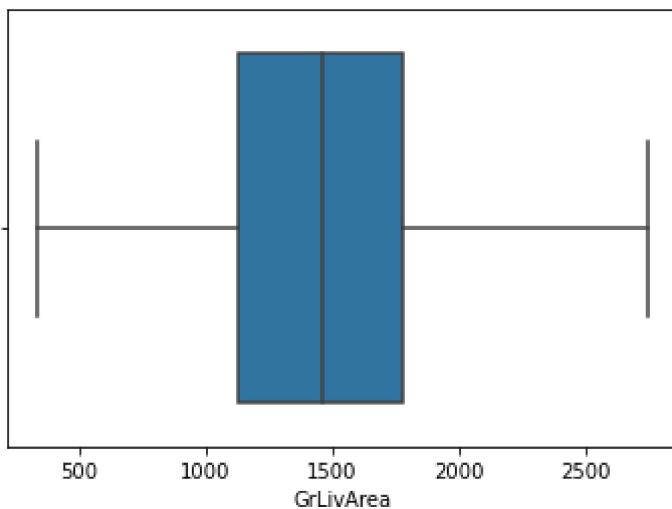
```
    warnings.warn(  
    ##### LowQualFinSF #####  
    percentile25 0.0  
    percentile75 0.0  
    higher_fence 0.0  
    lower_fence 0.0
```



```
    ##### GrLivArea #####  
    percentile25 1129.5  
    percentile75 1776.75  
    higher_fence 2747.625  
    lower_fence 158.625
```

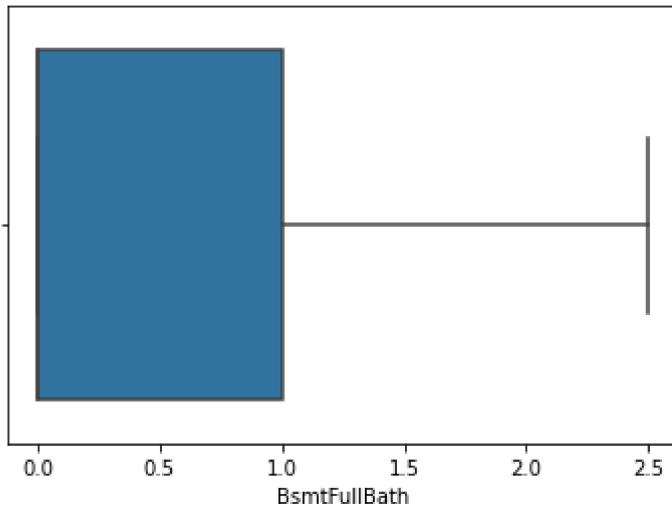
```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

```
    warnings.warn(
```



```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

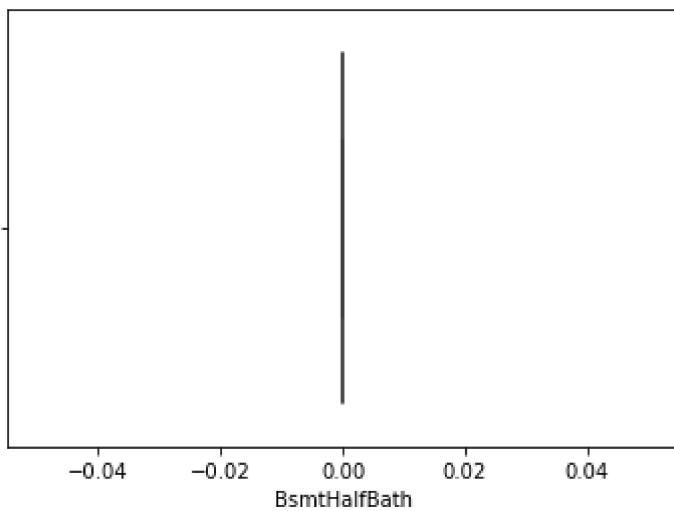
```
    warnings.warn(  
        ##### BsmtFullBath #####  
        percentile25 0.0  
        percentile75 1.0  
        higher_fence 2.5  
        lower_fence -1.5
```



```
    ##### BsmtHalfBath #####  
    percentile25 0.0  
    percentile75 0.0  
    higher_fence 0.0  
    lower_fence 0.0
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

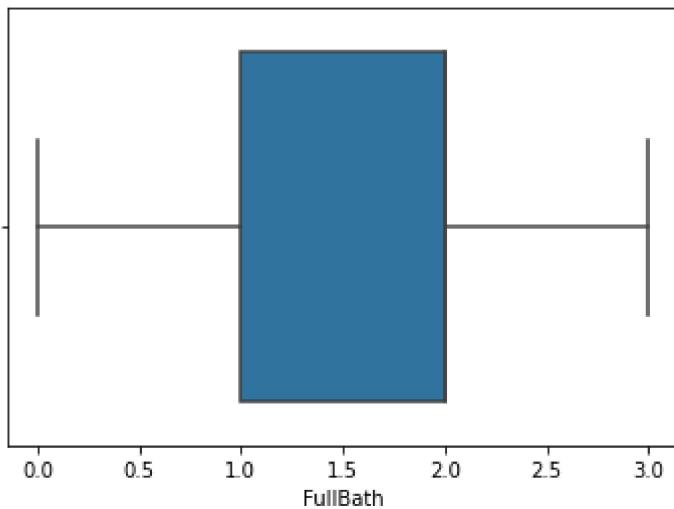
```
    warnings.warn(
```



```
#### FullBath ####  
percentile25 1.0  
percentile75 2.0  
higher_fence 3.5  
lower_fence -0.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

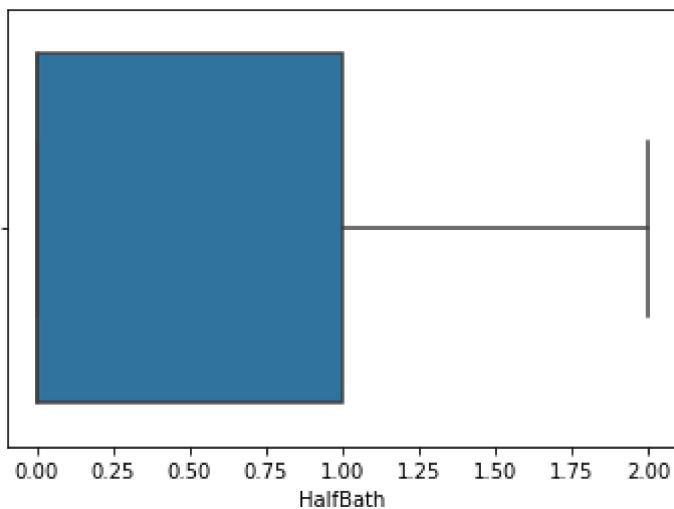
```
warnings.warn(
```



```
#### HalfBath ####  
percentile25 0.0  
percentile75 1.0  
higher_fence 2.5  
lower_fence -1.5
```

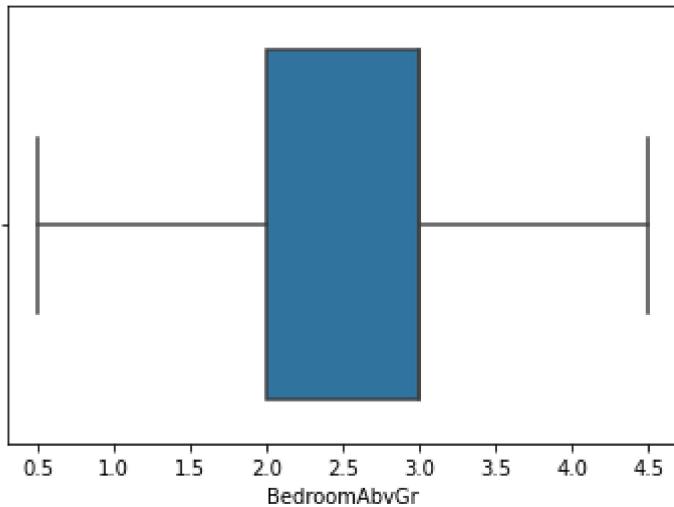
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```



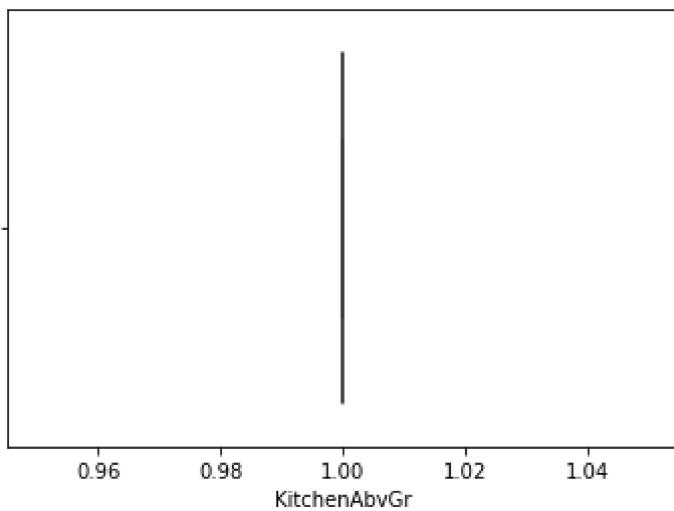
```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

```
    warnings.warn(  
        ##### BedroomAbvGr #####  
        percentile25 2.0  
        percentile75 3.0  
        higher_fence 4.5  
        lower_fence 0.5
```



```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

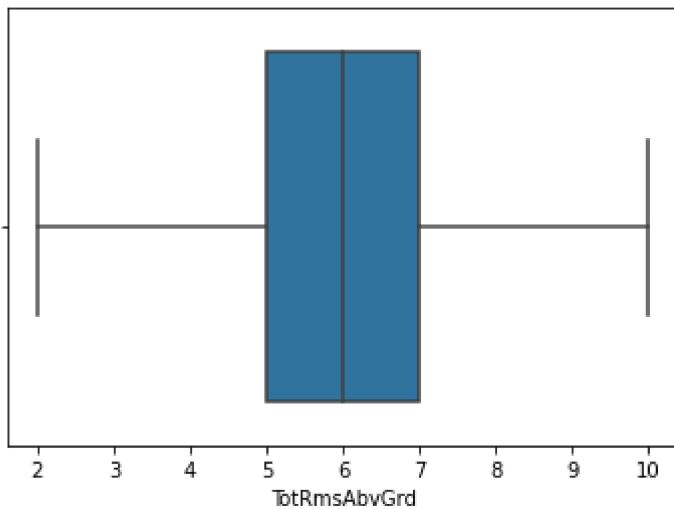
```
    warnings.warn(  
        ##### KitchenAbvGr #####  
        percentile25 1.0  
        percentile75 1.0  
        higher_fence 1.0  
        lower_fence 1.0
```



```
#### TotRmsAbvGr ####  
percentile25 5.0  
percentile75 7.0  
higher_fence 10.0  
lower_fence 2.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

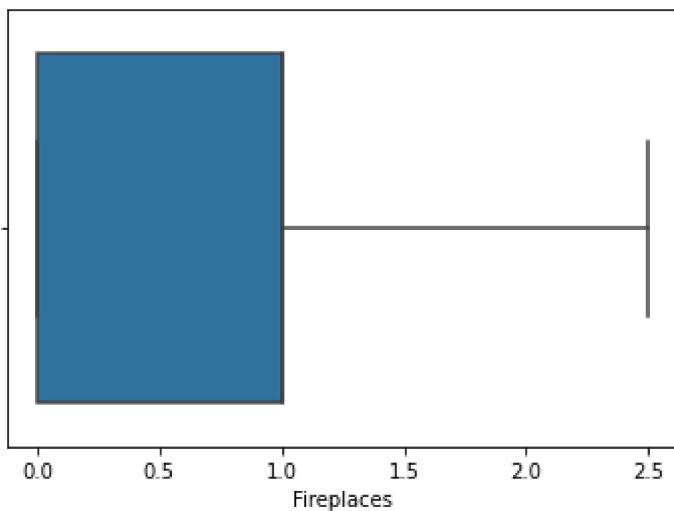
```
warnings.warn(
```



```
#### Fireplaces ####  
percentile25 0.0  
percentile75 1.0  
higher_fence 2.5  
lower_fence -1.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

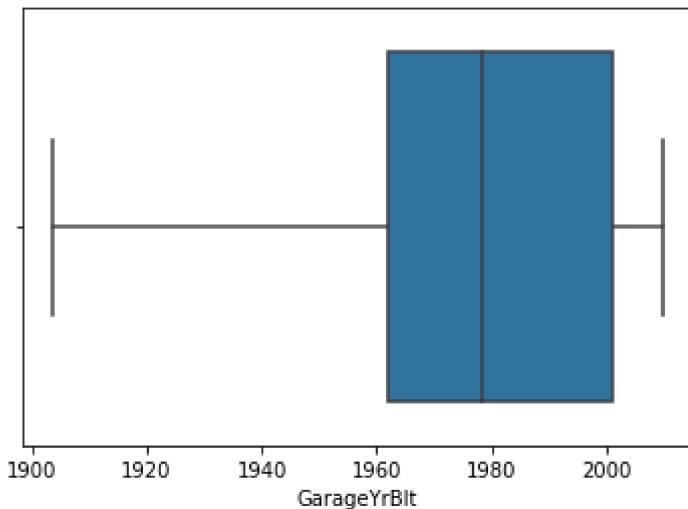
```
warnings.warn(
```



```
#### GarageYrBlt ####  
percentile25 1962.0  
percentile75 2001.0  
higher_fence 2059.5  
lower_fence 1903.5
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

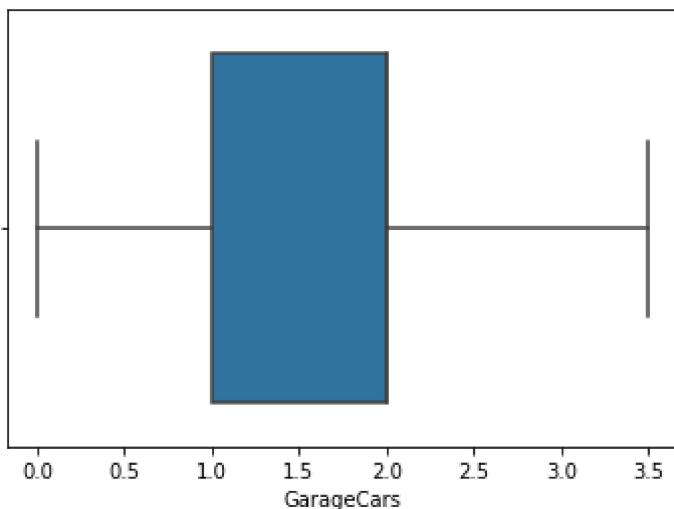
```
warnings.warn(
```



```
#### GarageCars ####  
percentile25 1.0  
percentile75 2.0  
higher_fence 3.5  
lower_fence -0.5
```

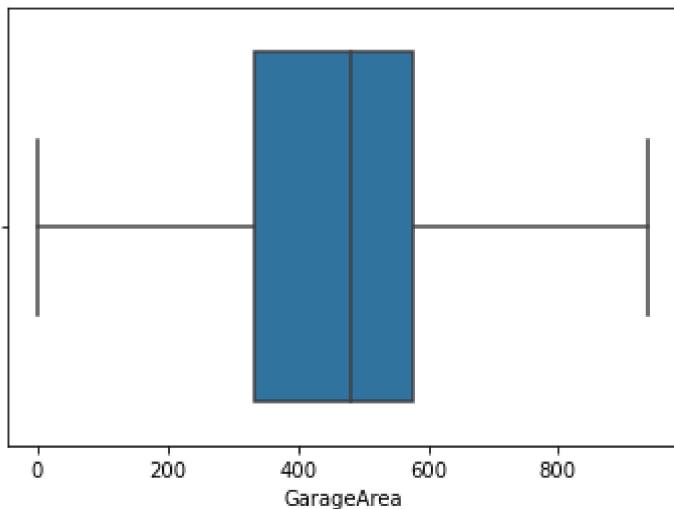
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```



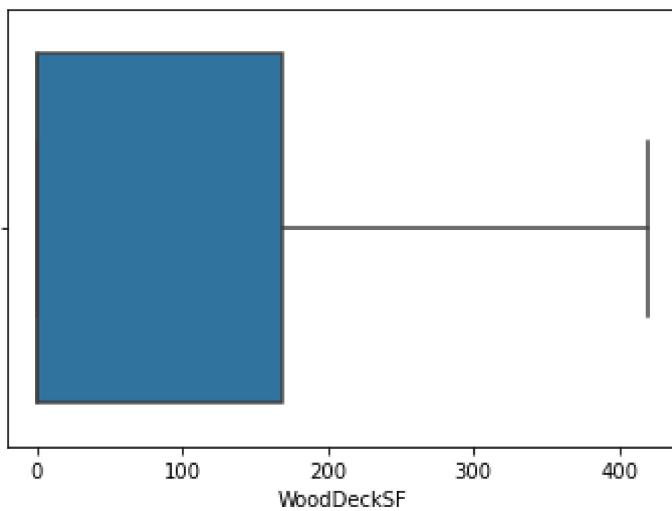
```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

```
    warnings.warn(  
        ##### GarageArea #####  
        percentile25 334.5  
        percentile75 576.0  
        higher_fence 938.25  
        lower_fence -27.75
```



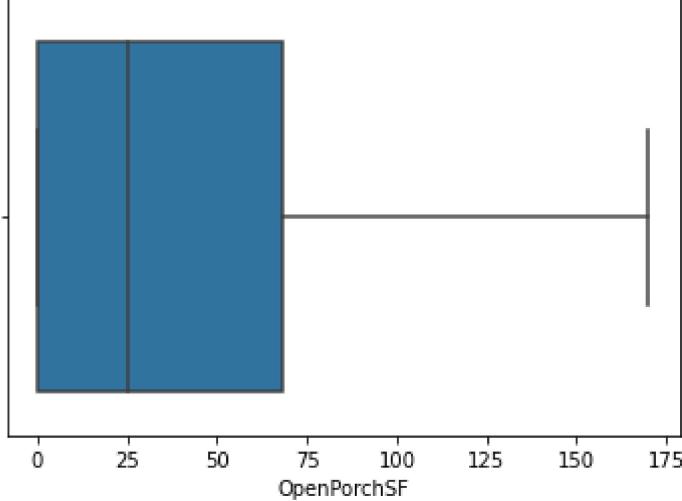
```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

```
    warnings.warn(  
        ##### WoodDeckSF #####  
        percentile25 0.0  
        percentile75 168.0  
        higher_fence 420.0  
        lower_fence -252.0
```



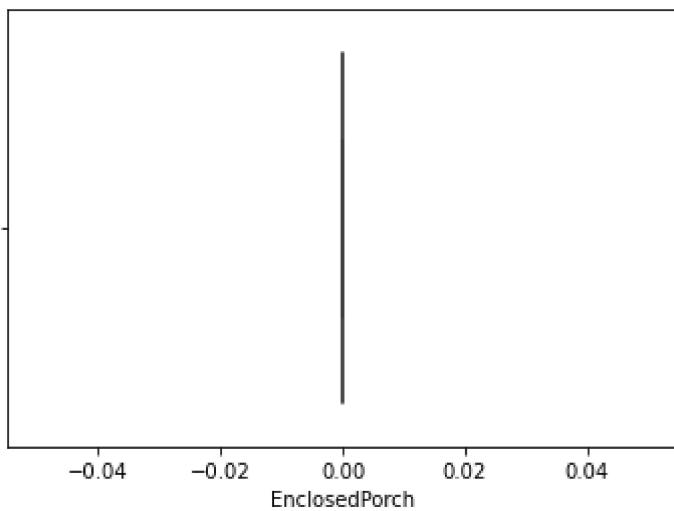
```
#### OpenPorchSF ####  
percentile25 0.0  
percentile75 68.0  
higher_fence 170.0  
lower_fence -102.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.  
warnings.warn(



```
#### EnclosedPorch ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```

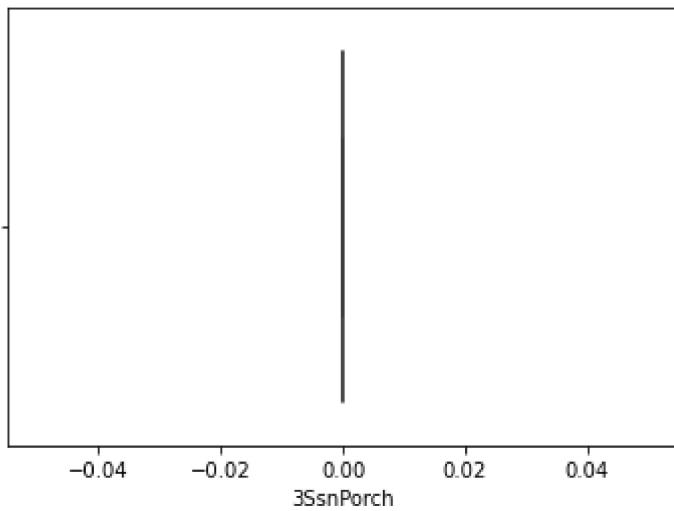
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.  
warnings.warn(



```
#### 3SsnPorch ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```

C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

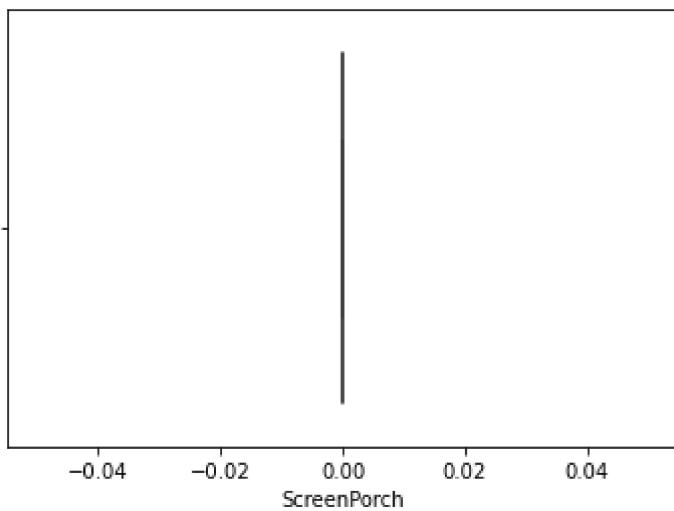
```
warnings.warn(
```



C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

```
warnings.warn(
```

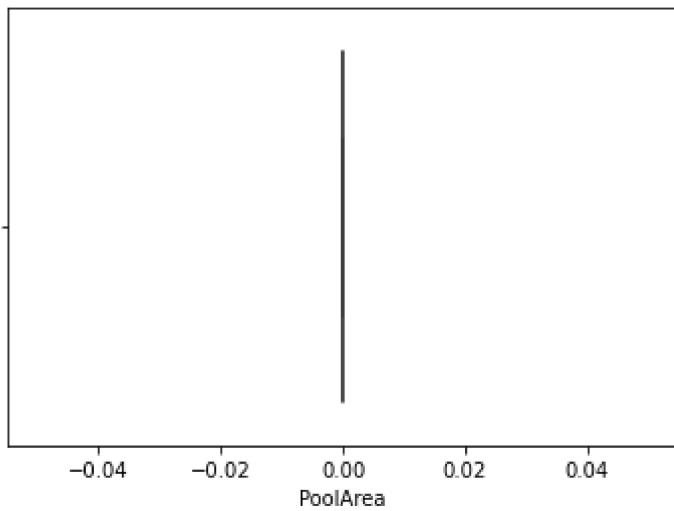
```
#### ScreenPorch ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```



```
#### PoolArea ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

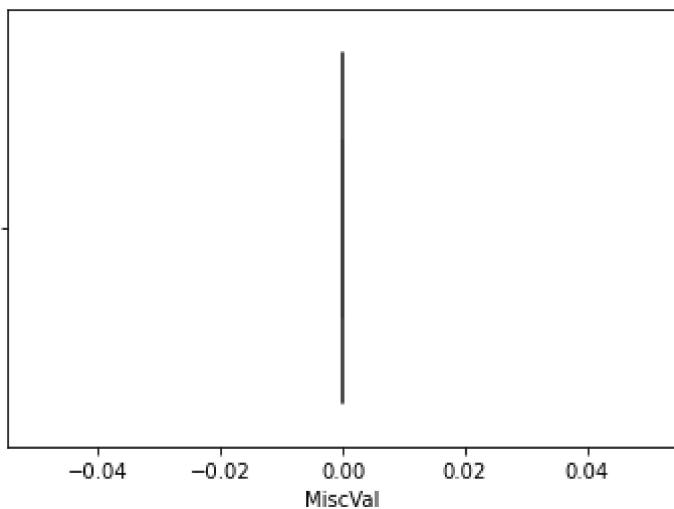
```
warnings.warn(
```



```
#### MiscVal ####  
percentile25 0.0  
percentile75 0.0  
higher_fence 0.0  
lower_fence 0.0
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

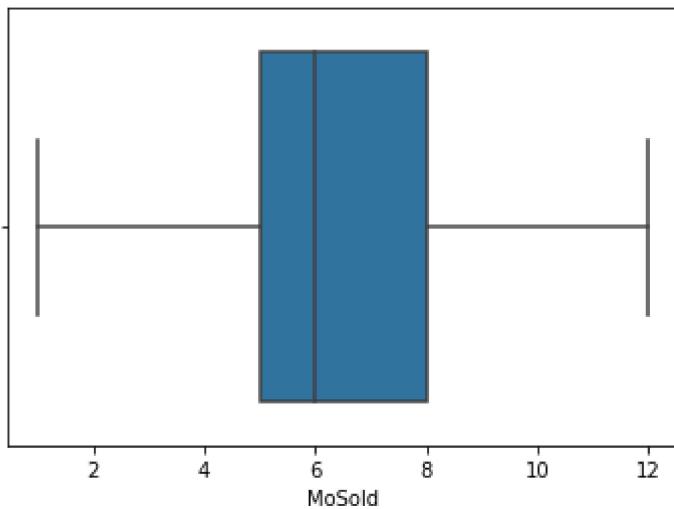
```
warnings.warn(
```



```
#### MoSold ####  
percentile25 5.0  
percentile75 8.0  
higher_fence 12.5  
lower_fence 0.5
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

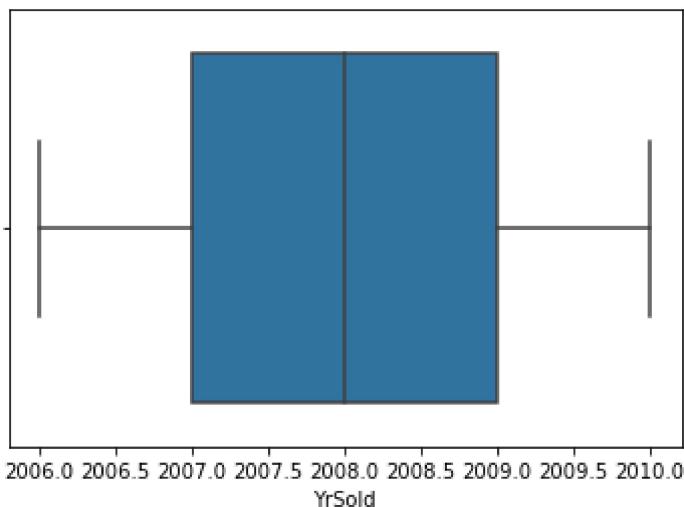
```
warnings.warn(
```



```
#### YrSold ####  
percentile25 2007.0  
percentile75 2009.0  
higher_fence 2012.0  
lower_fence 2004.0
```

```
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.
```

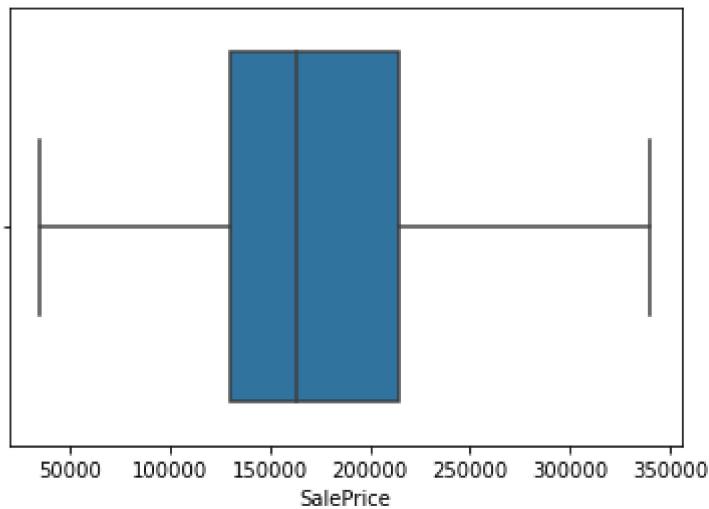
```
warnings.warn(
```



```
#### SalePrice ####
percentile25 129975.0
percentile75 214000.0
higher_fence 340037.5
lower_fence 3937.5
```

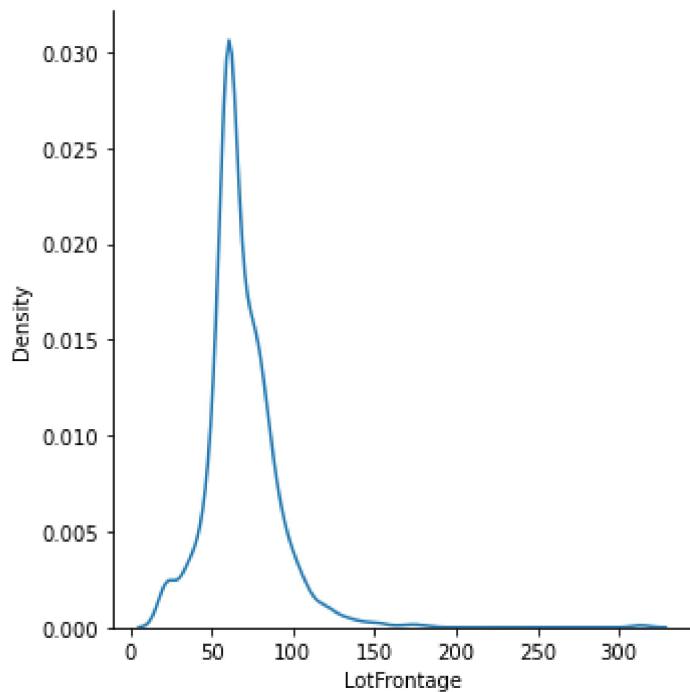
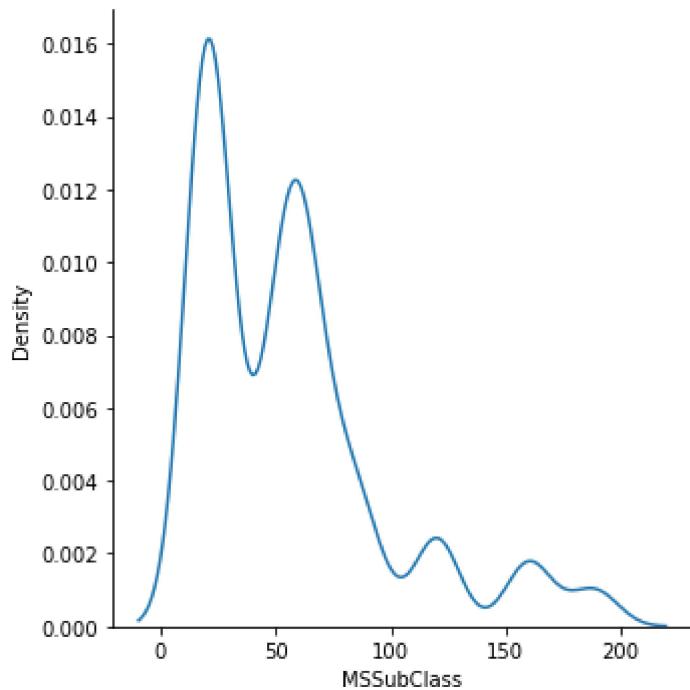
C:\Users\460379\Anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning:  
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po  
sitional argument will be `data`, and passing other arguments without an explicit key  
word will result in an error or misinterpretation.

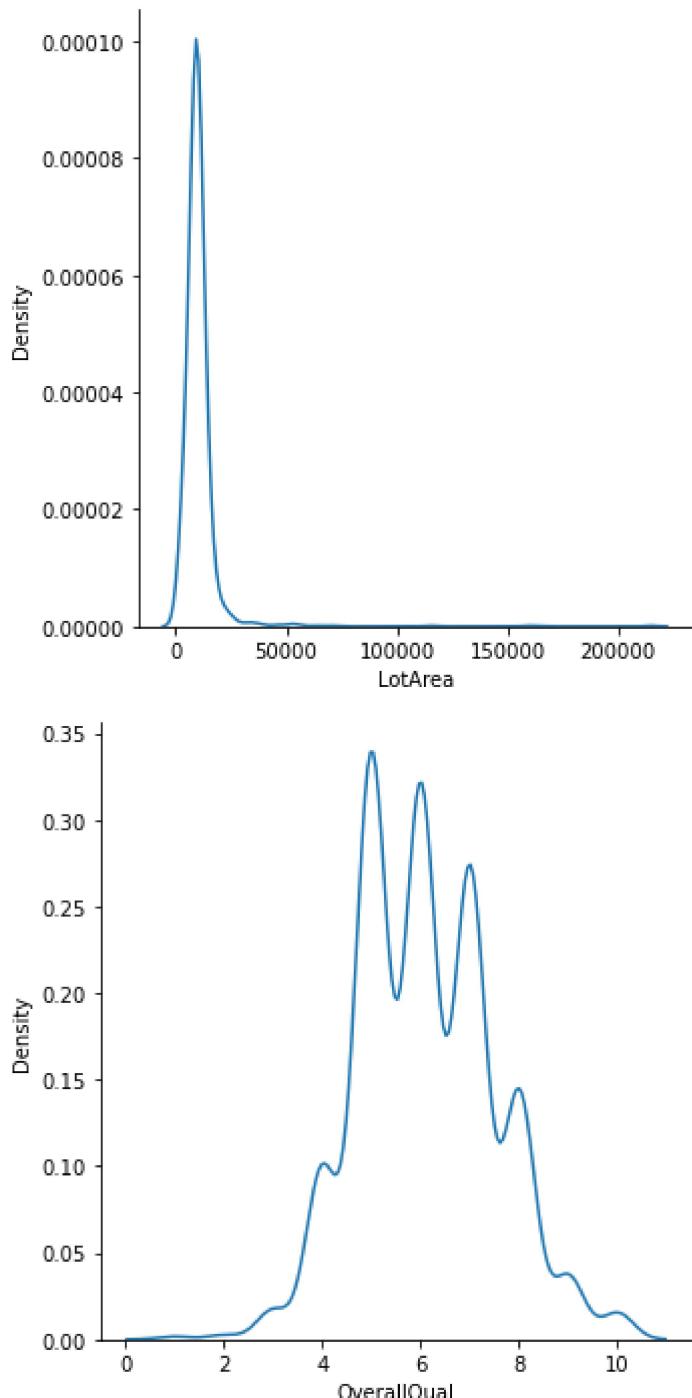
```
warnings.warn(
```

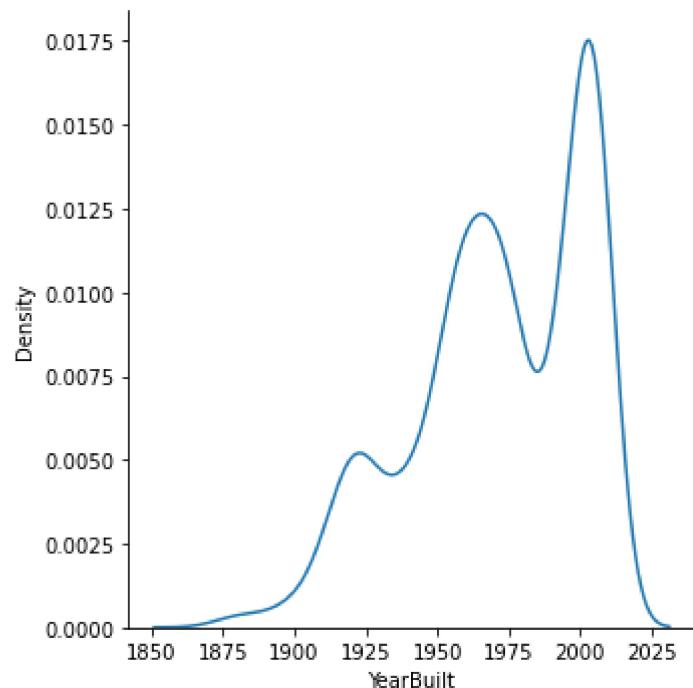
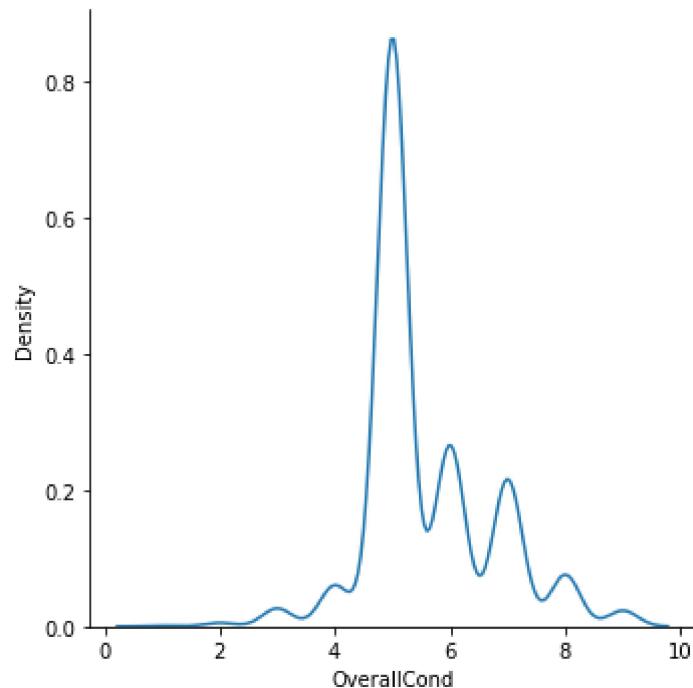


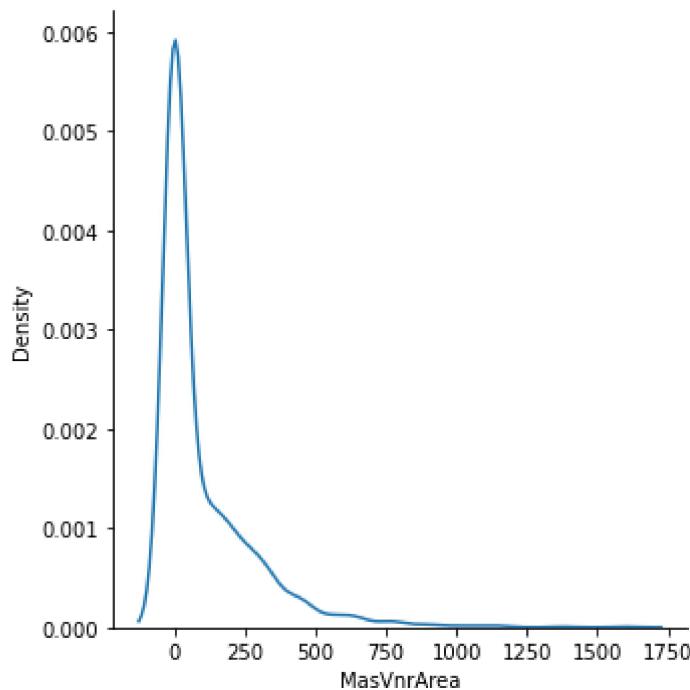
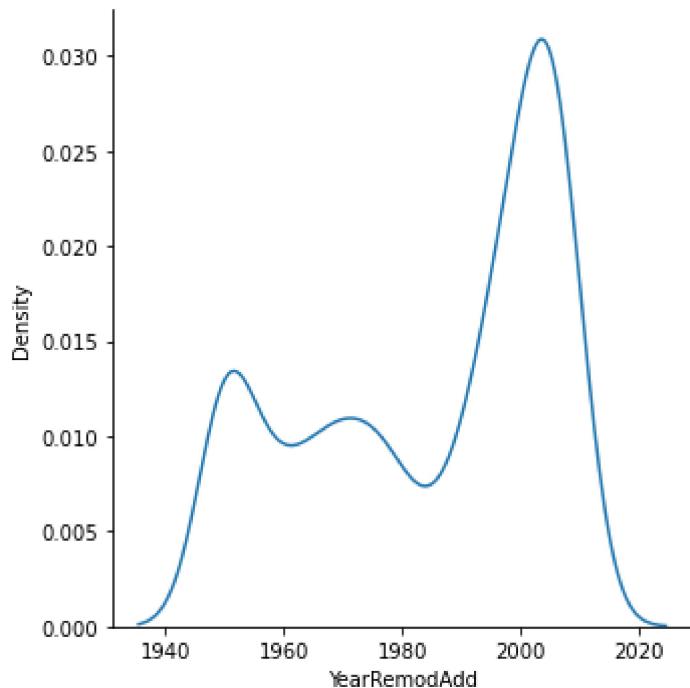
In [60]: *#Let us check the distribution plot for all the numerical features*

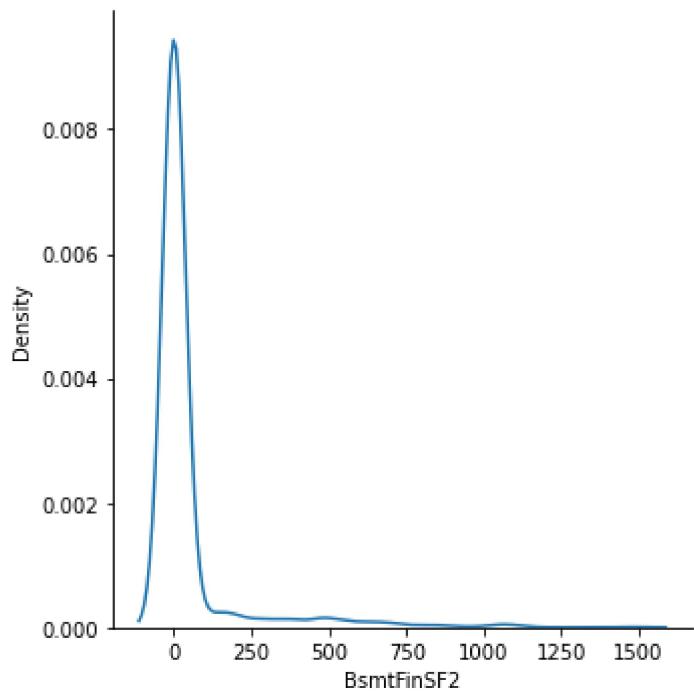
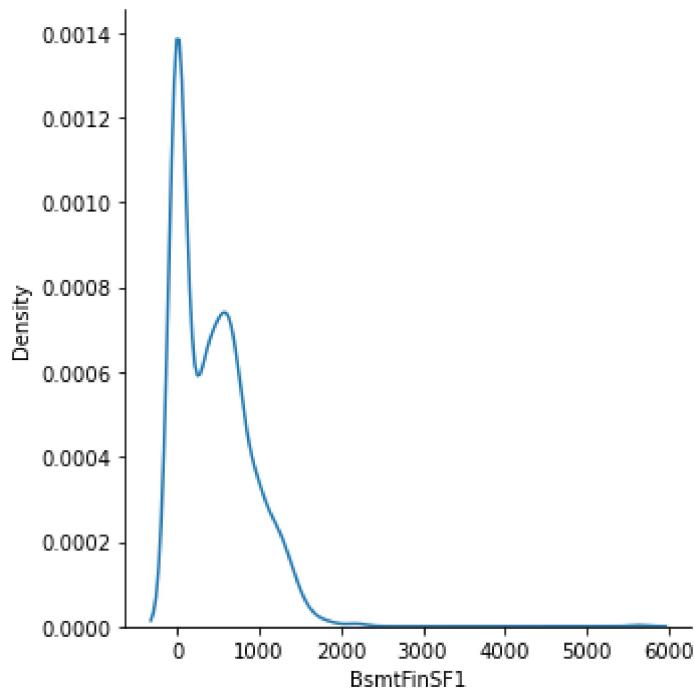
```
for features in df1.describe().columns:
    df1 = df.copy()
    sns.displot(data = df1, x = df1[features], kind = 'kde')
    plt.show()
```

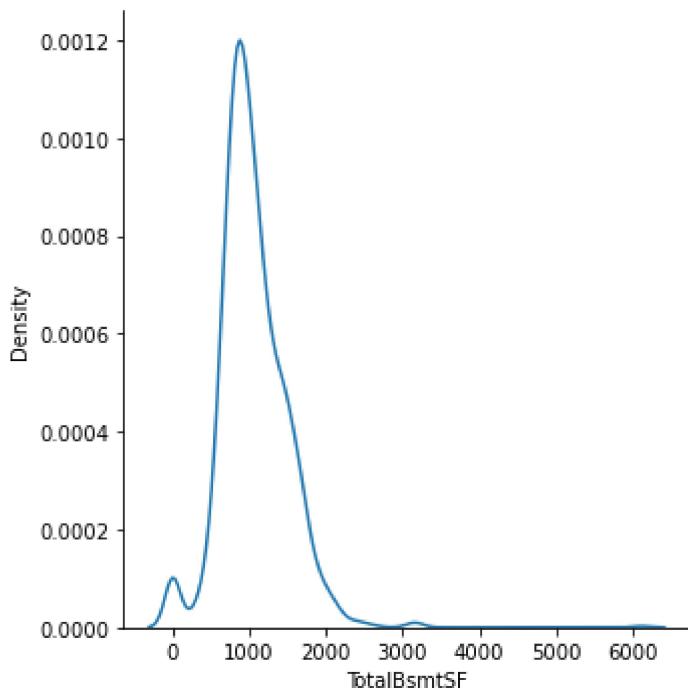
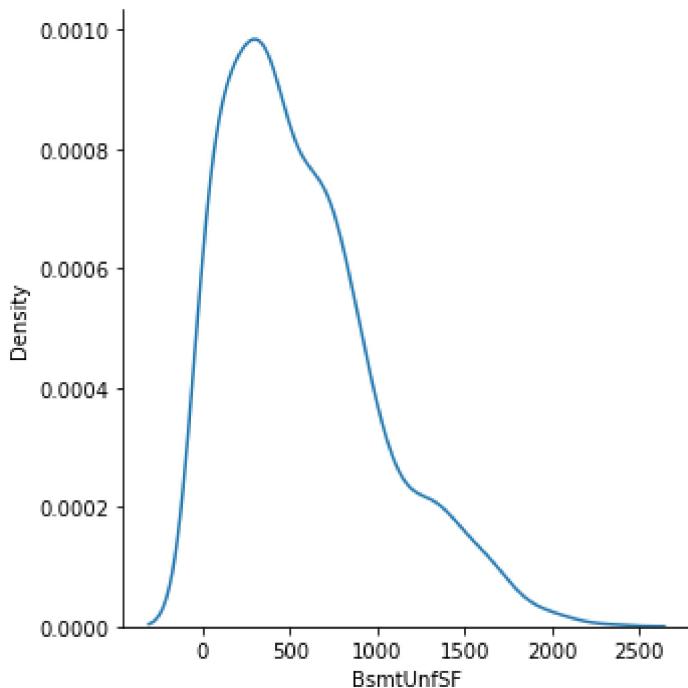


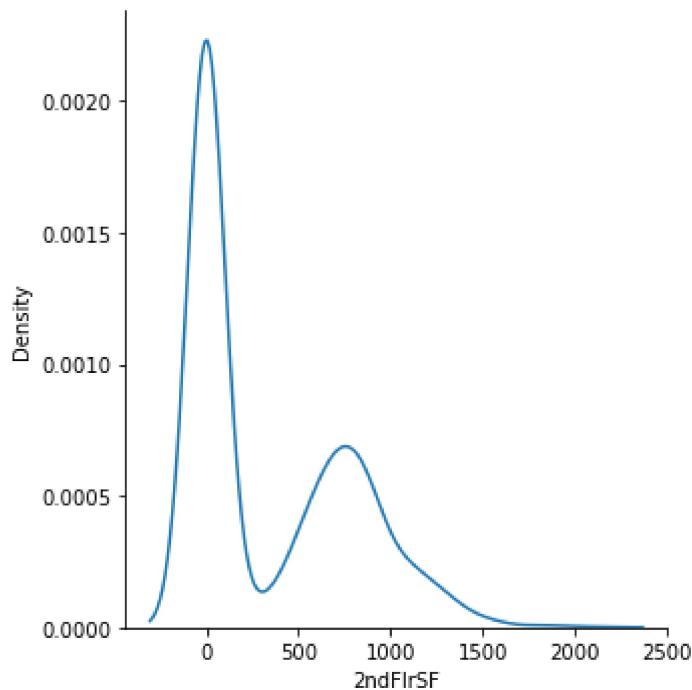
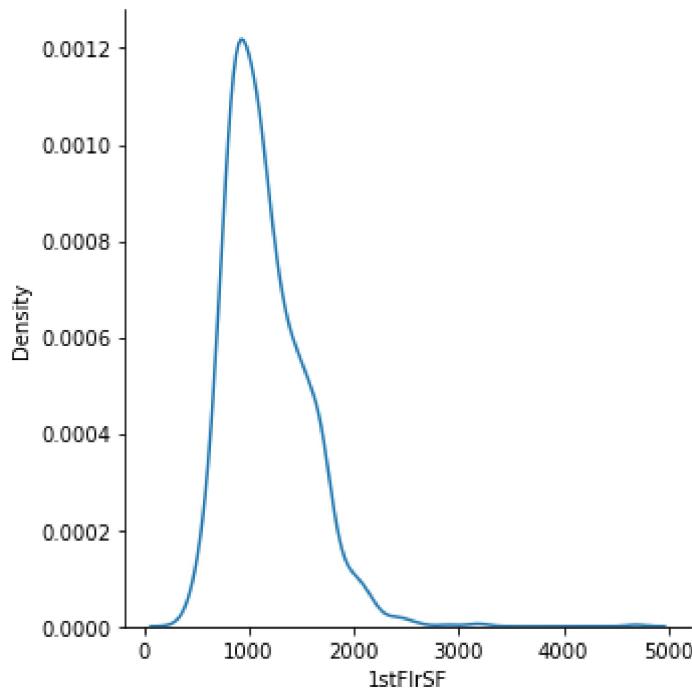


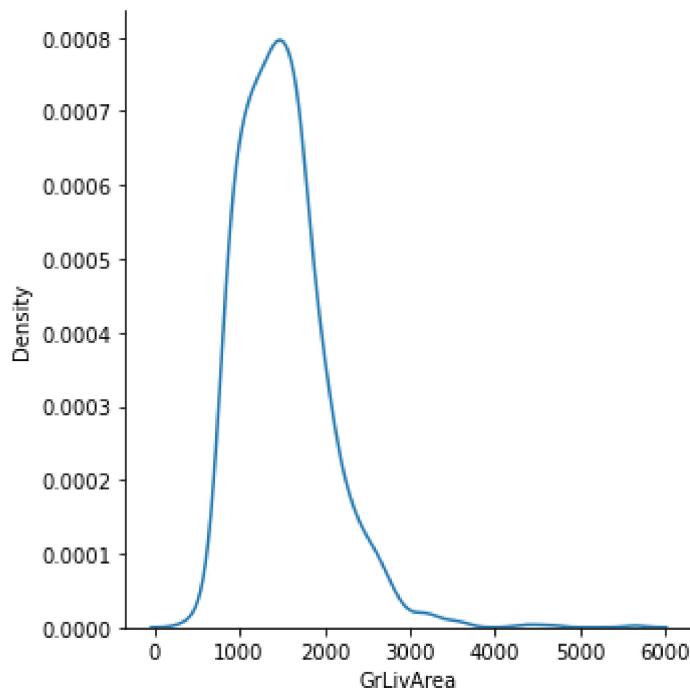
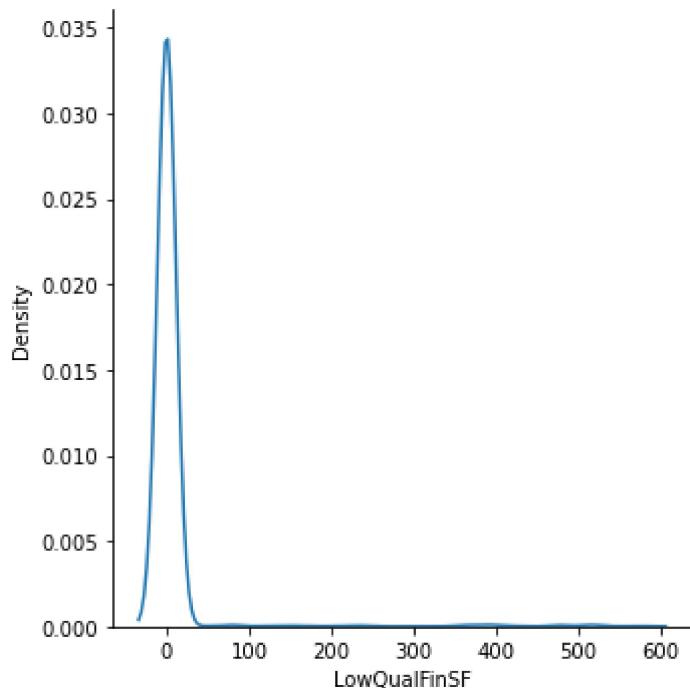


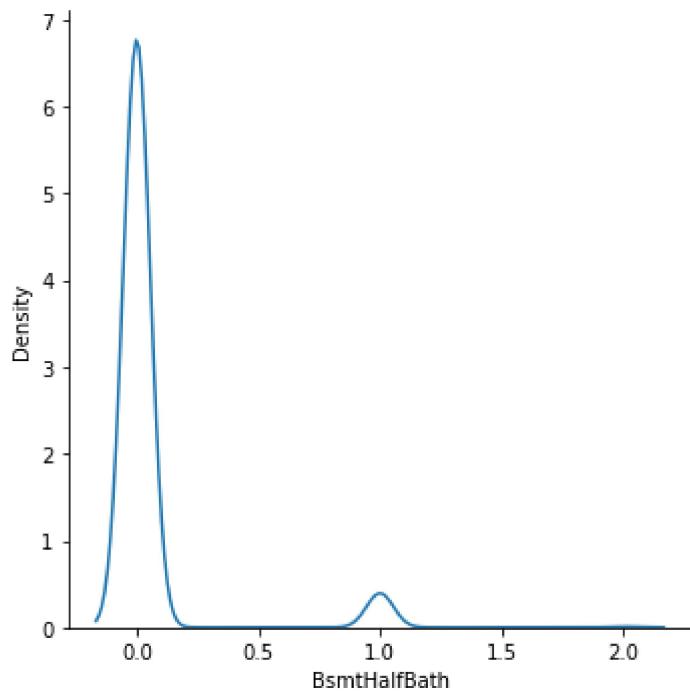
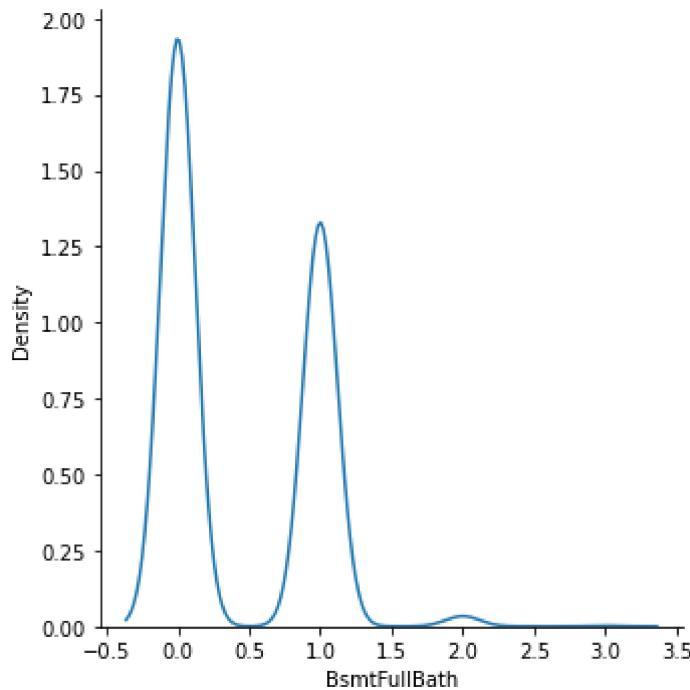


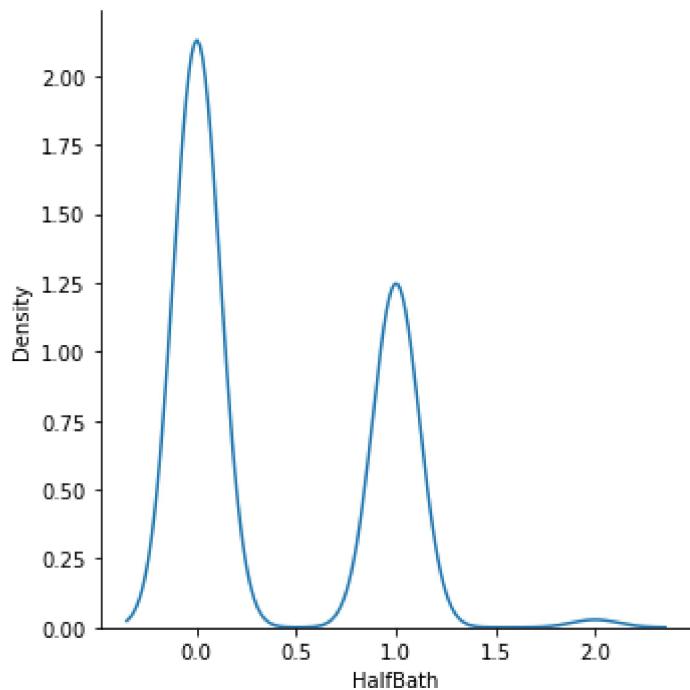
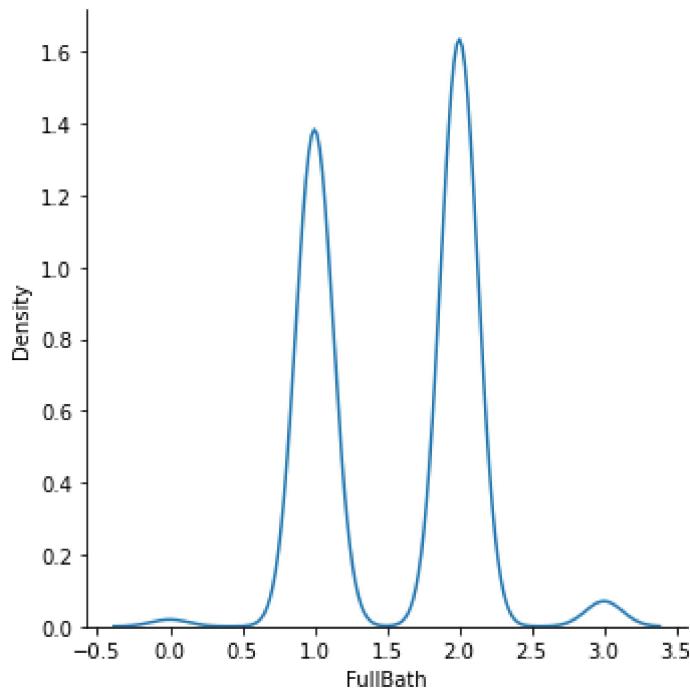


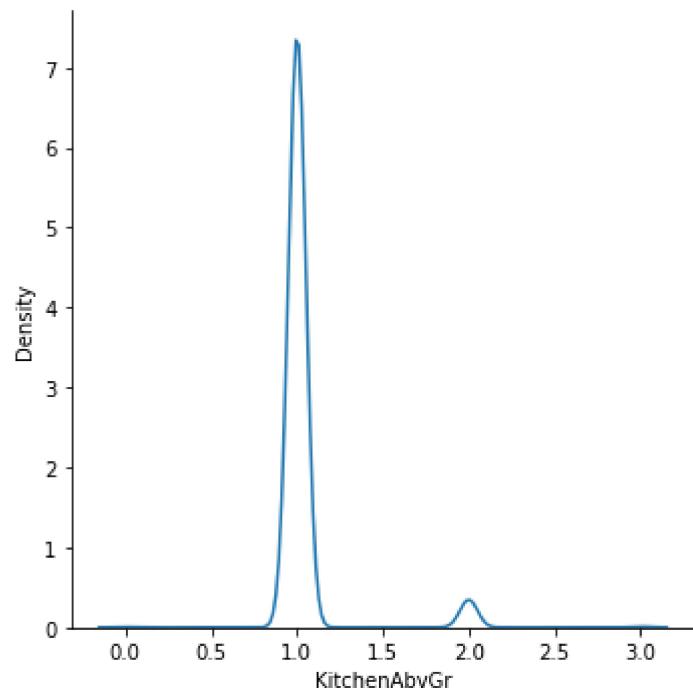
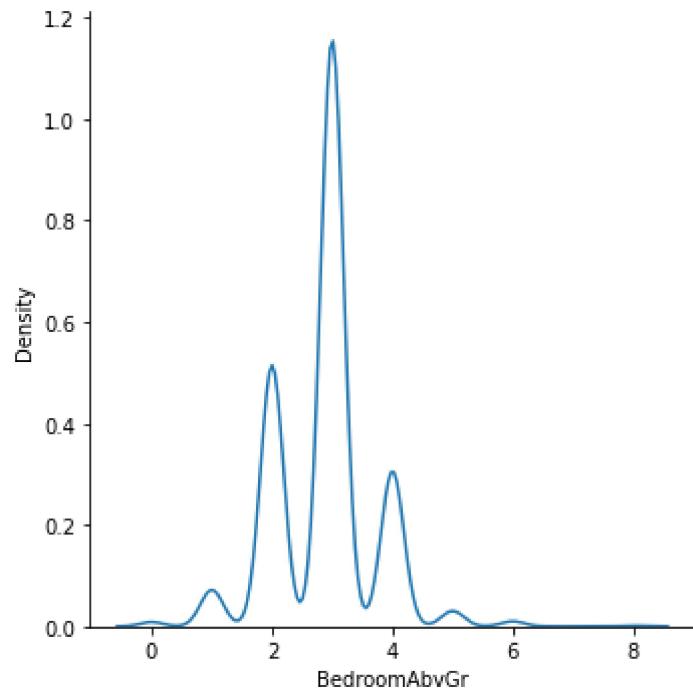


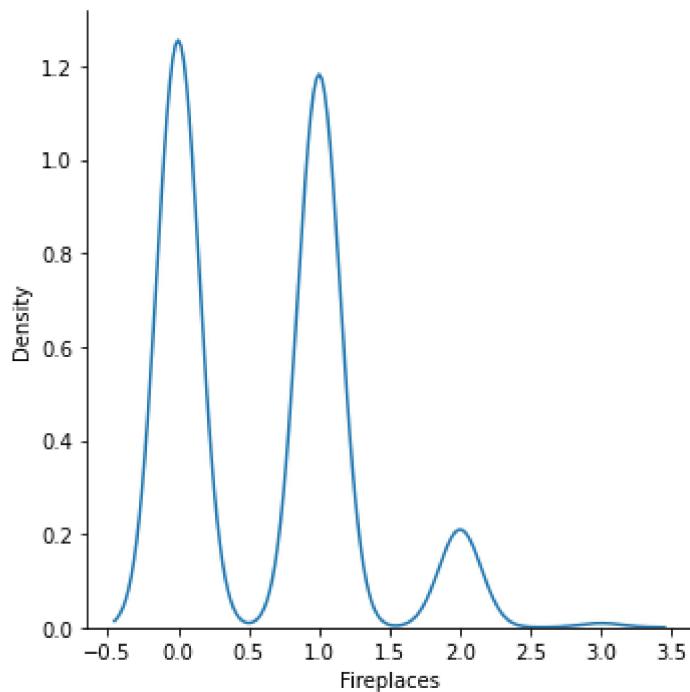
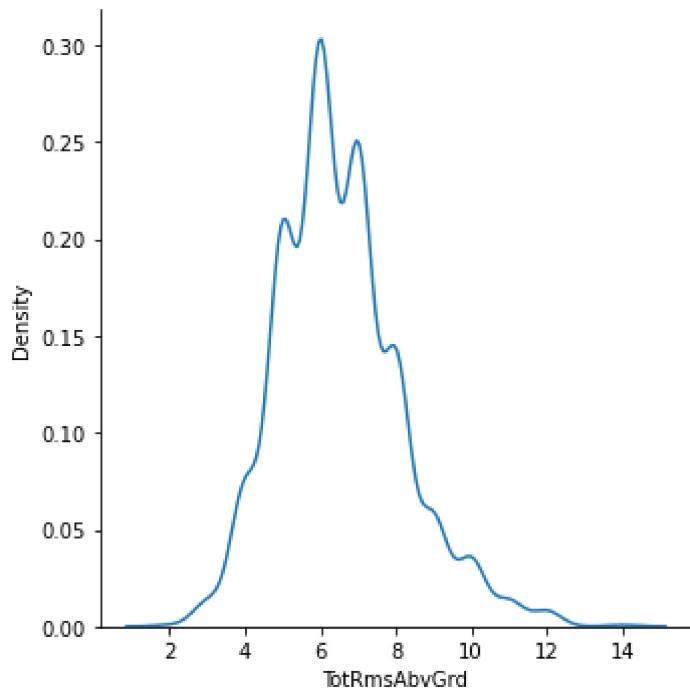


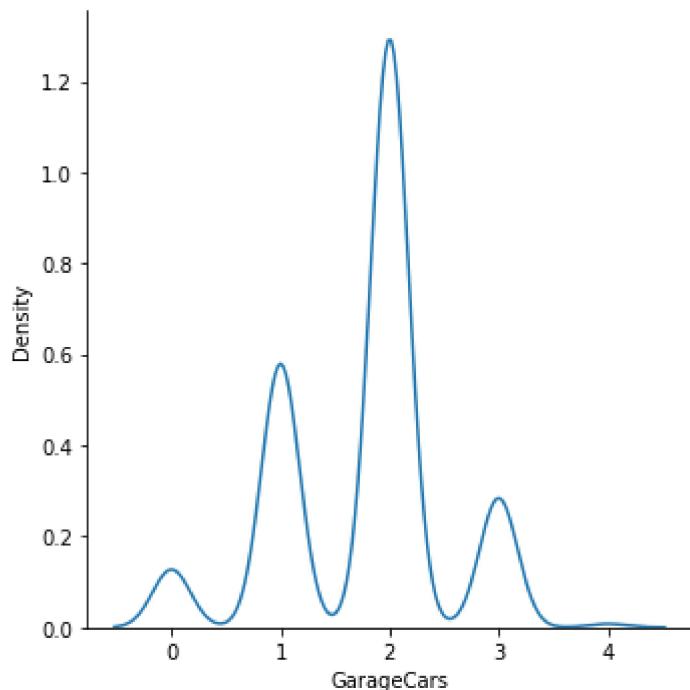
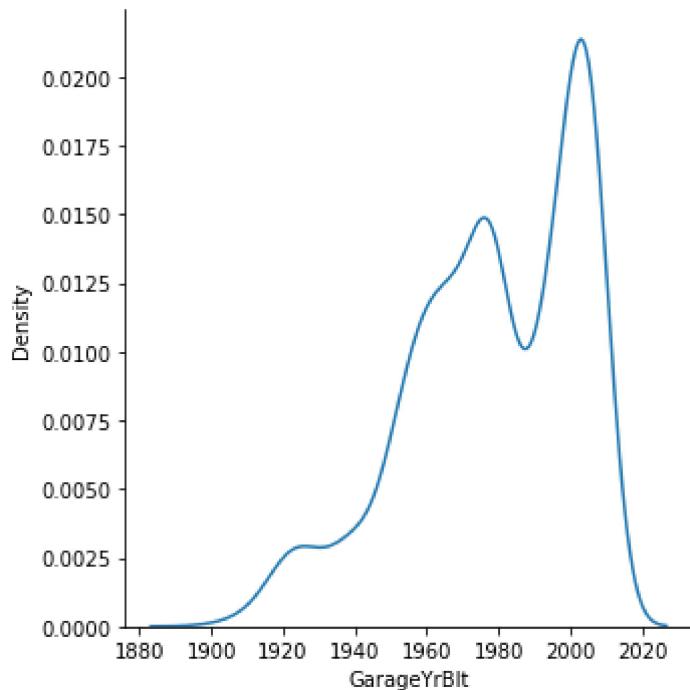


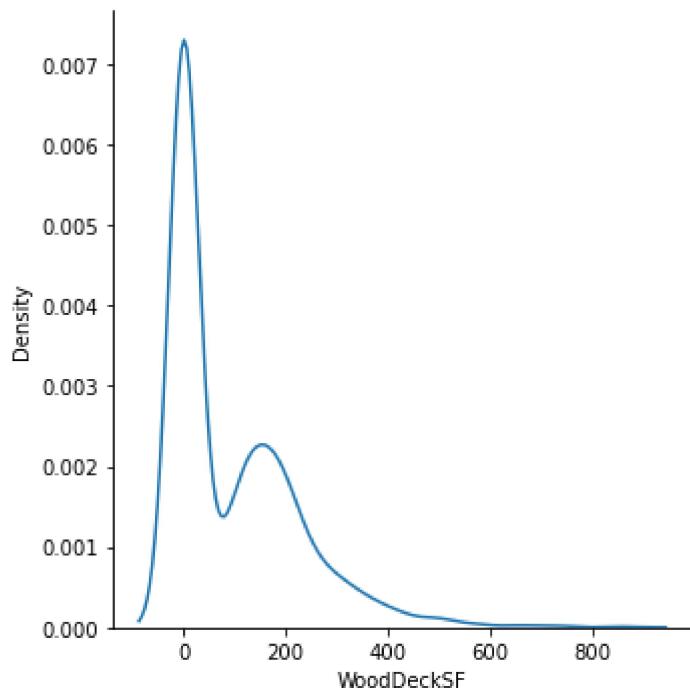
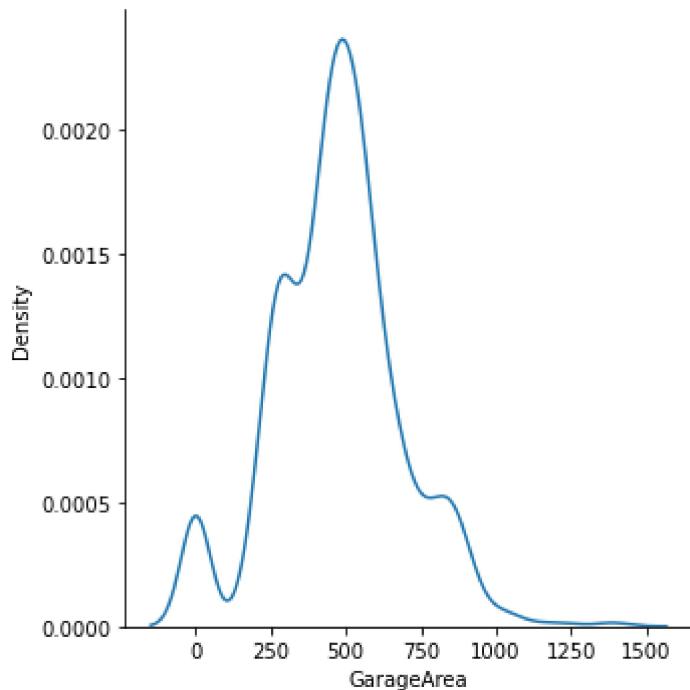


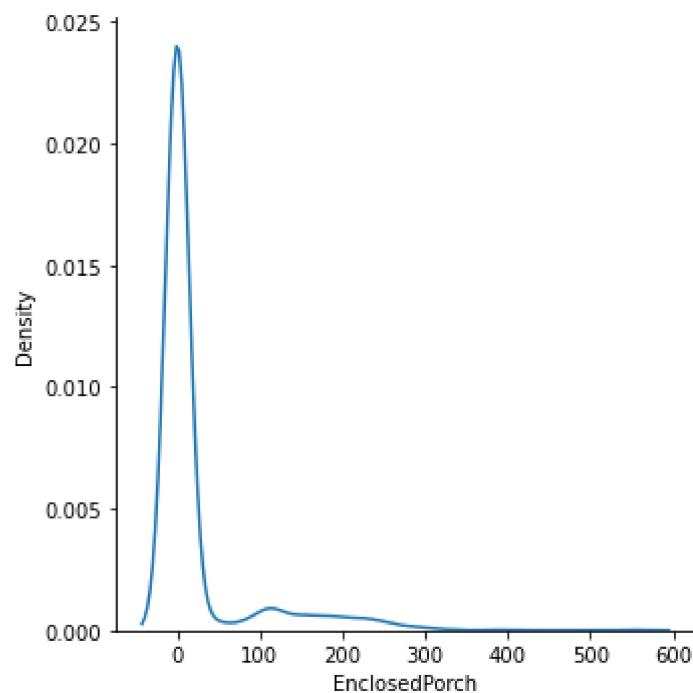
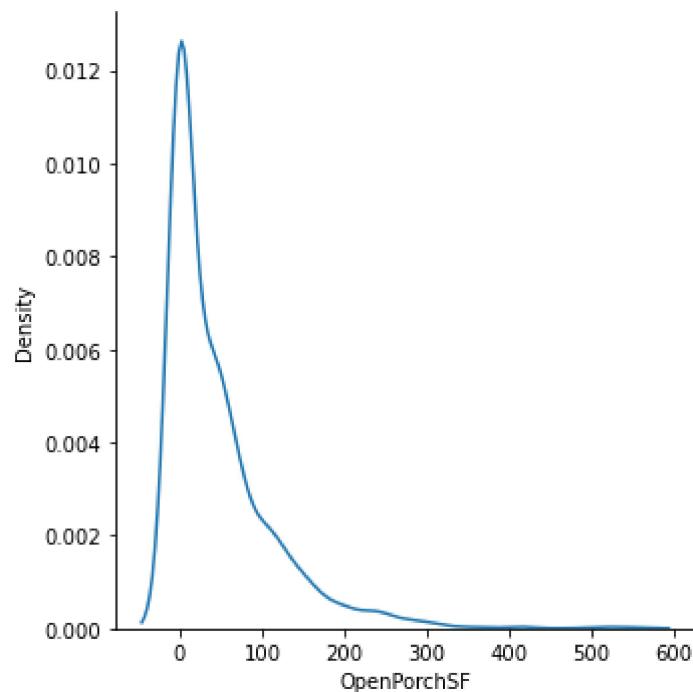


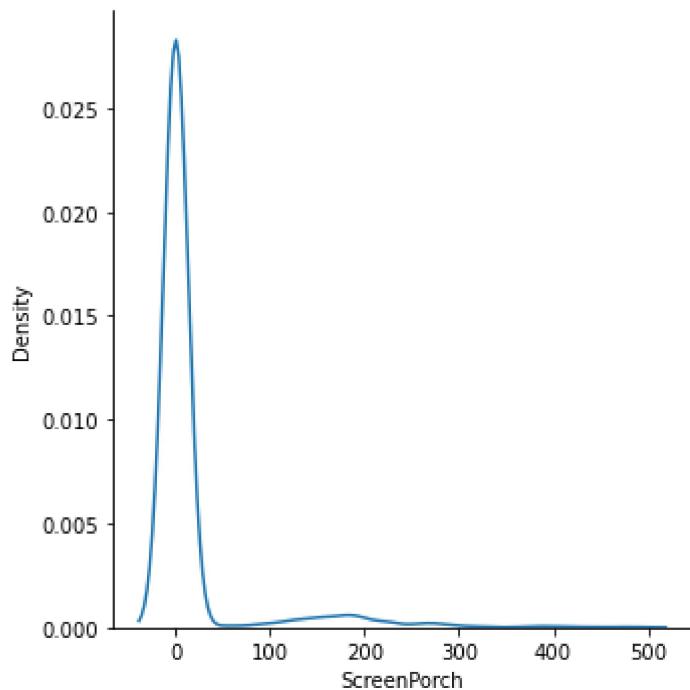
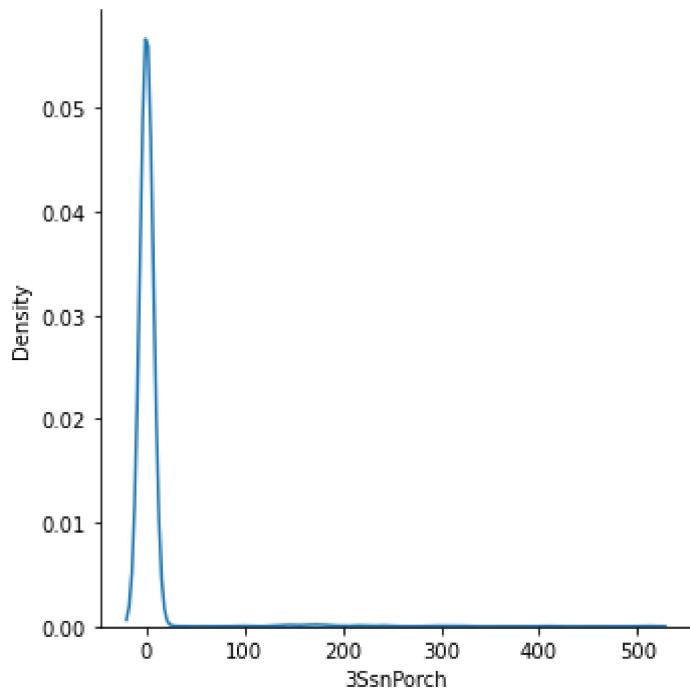


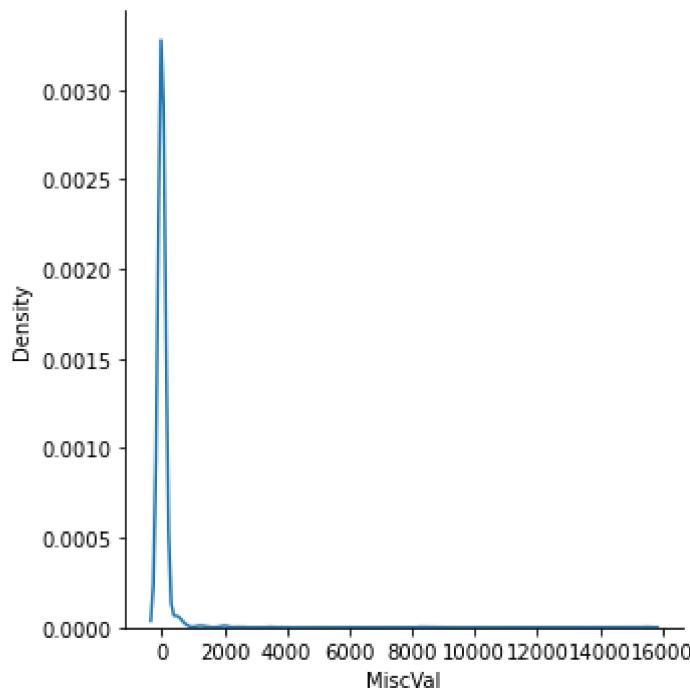
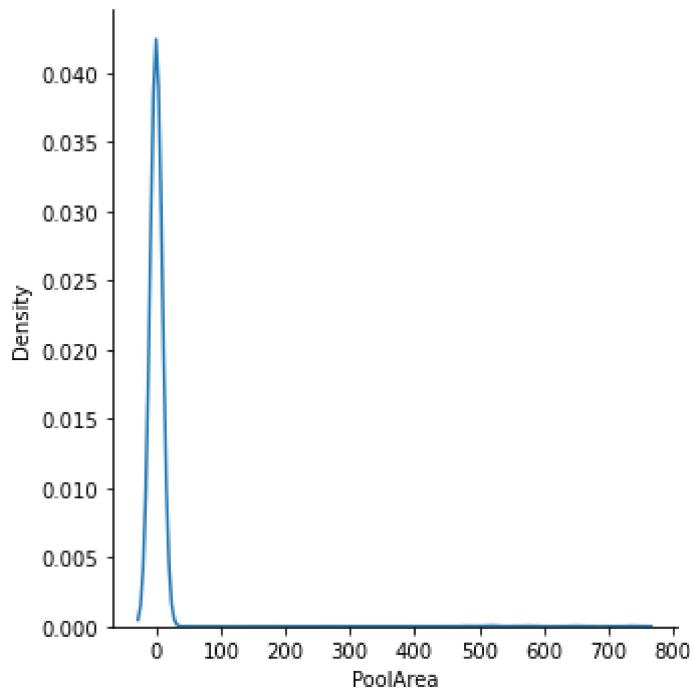


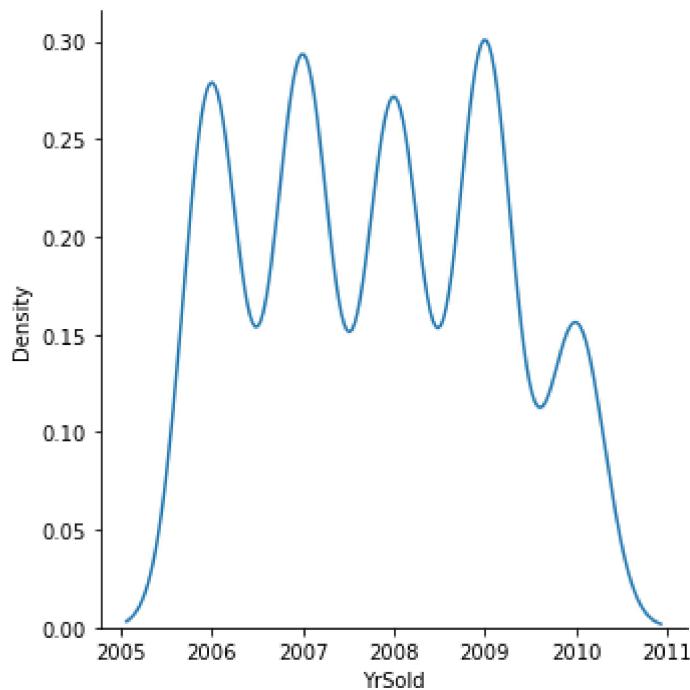
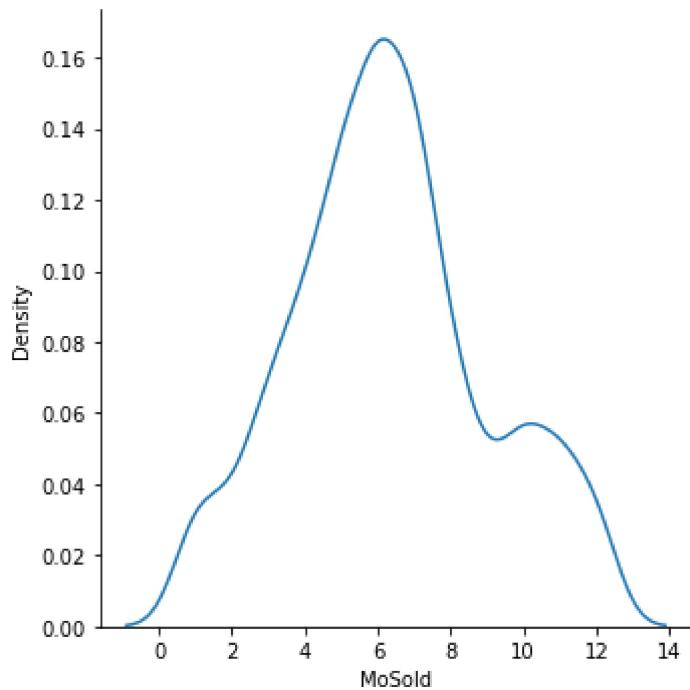


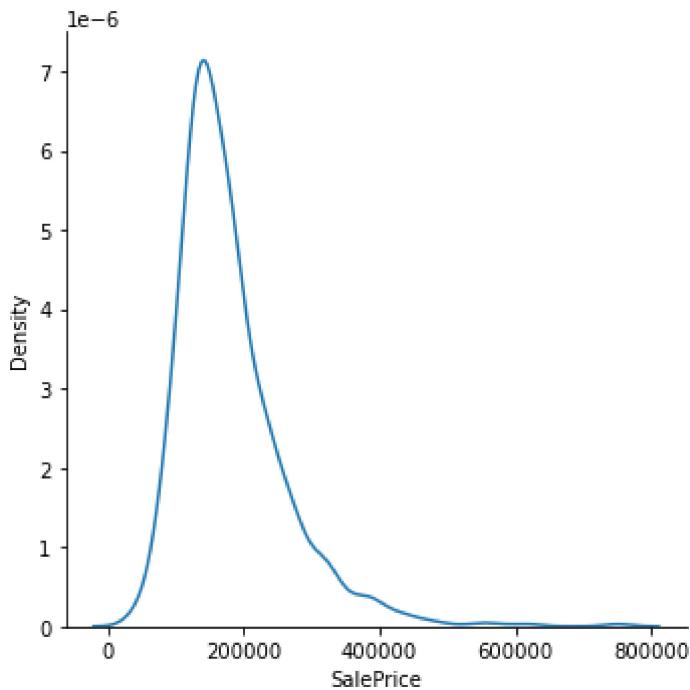






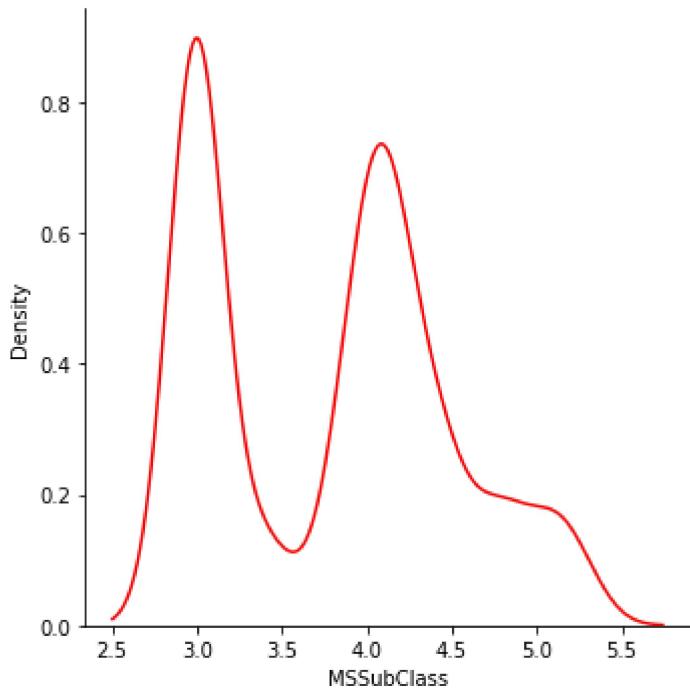


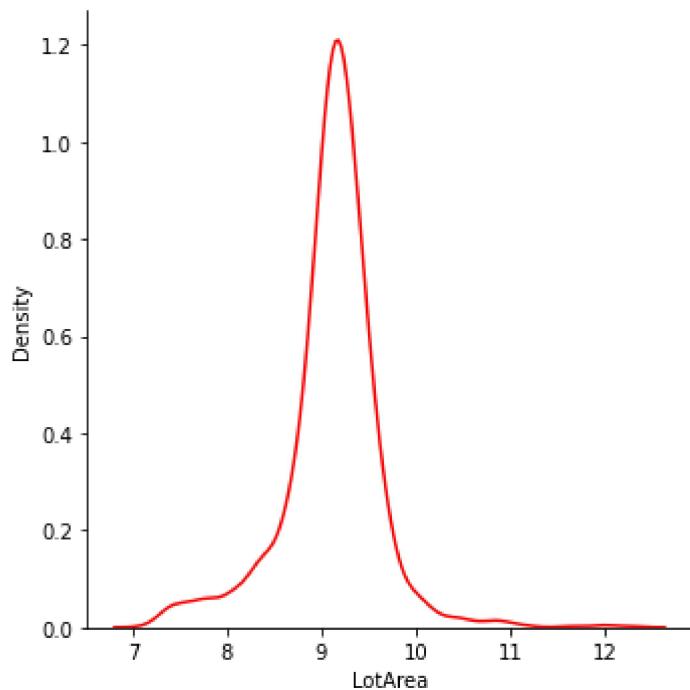
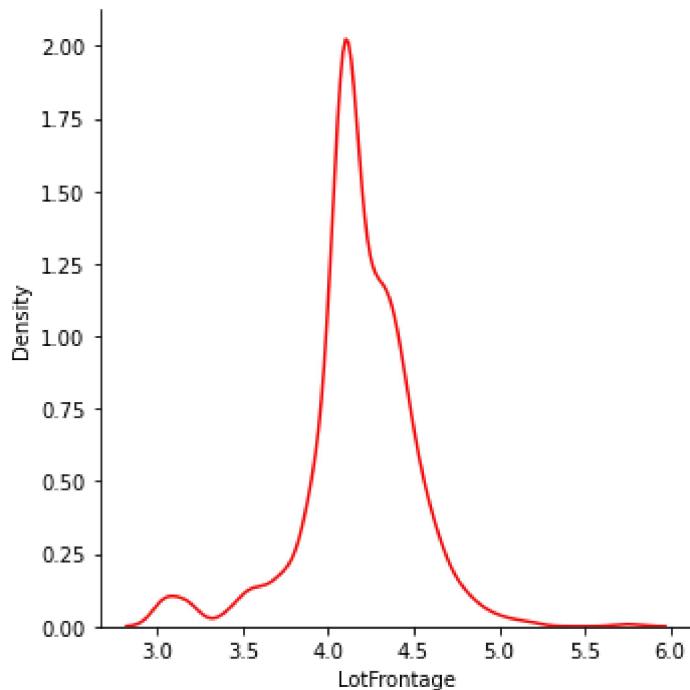


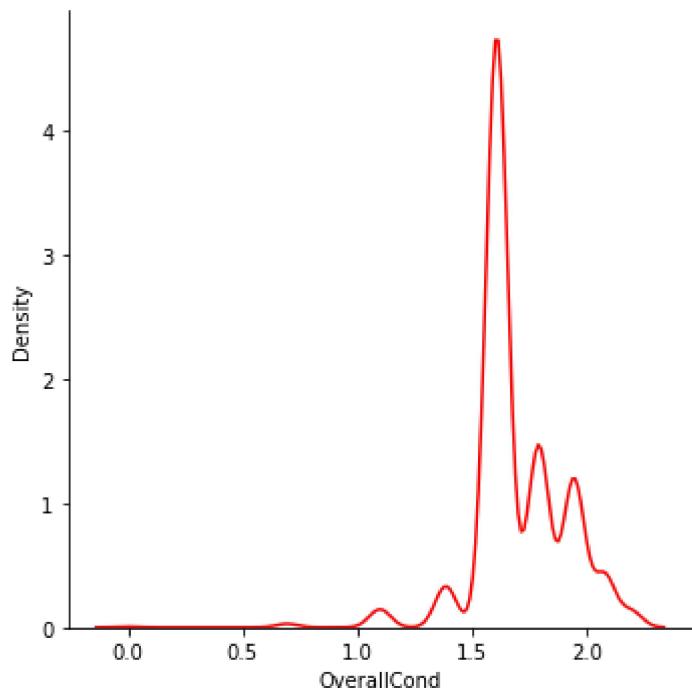
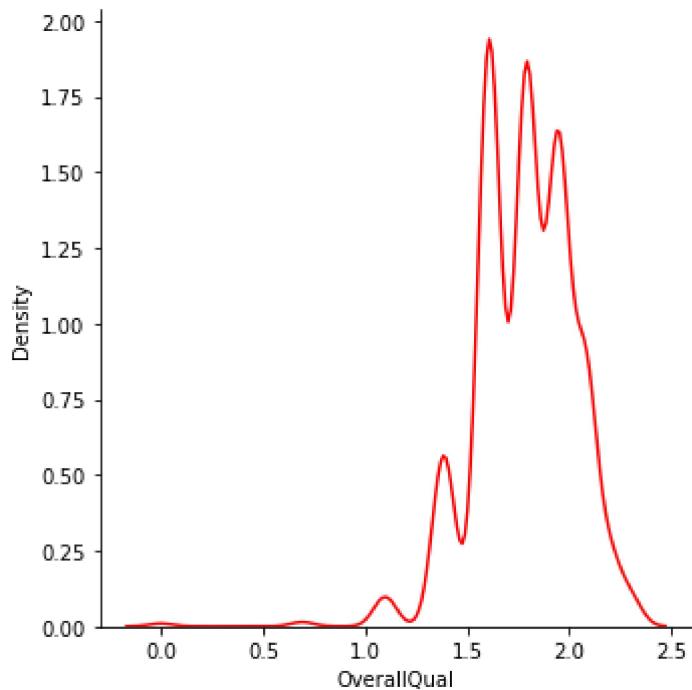


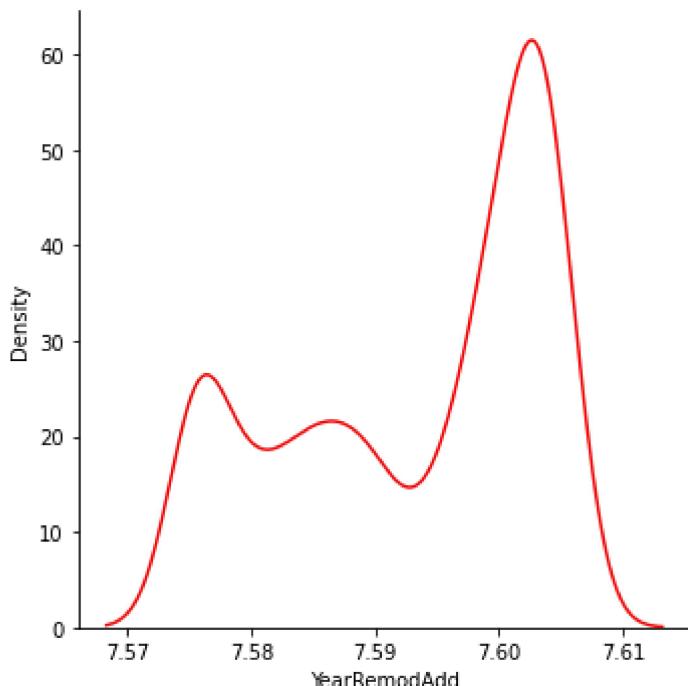
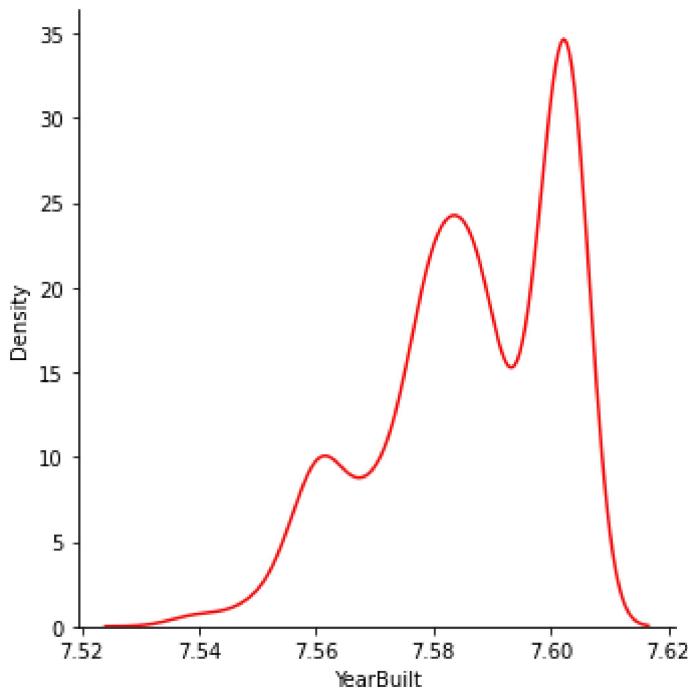
```
In [61]: # Let us now perform Log transformation on all the numerical features except the target

numerical_features = [features for features in Numerical_features if features not in 'Id']
for features in numerical_features:
    df1=df.copy()
    sns.scatterplot(x = np.log(df1[features]), y = 'SalePrice', data=df1)
    sns.displot(data = df1, x = np.log(df1[features]), kind = 'kde', color = 'r')
    plt.show()
```

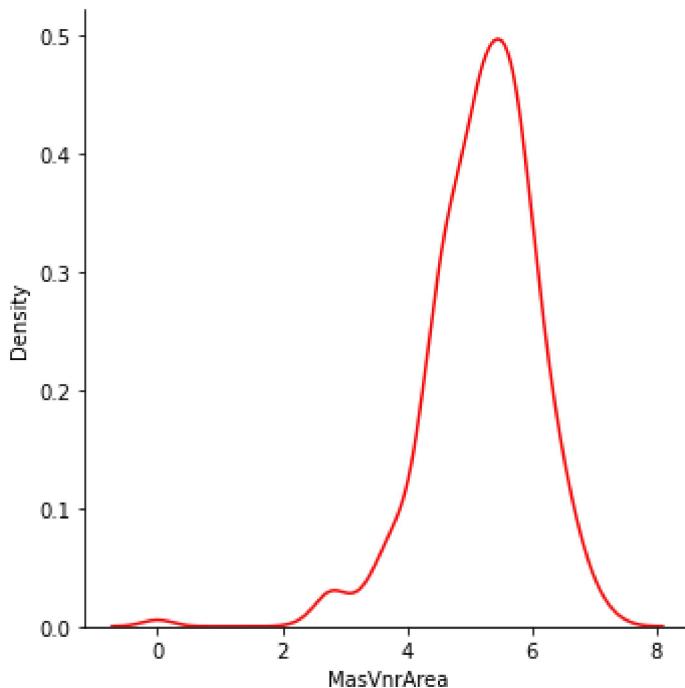




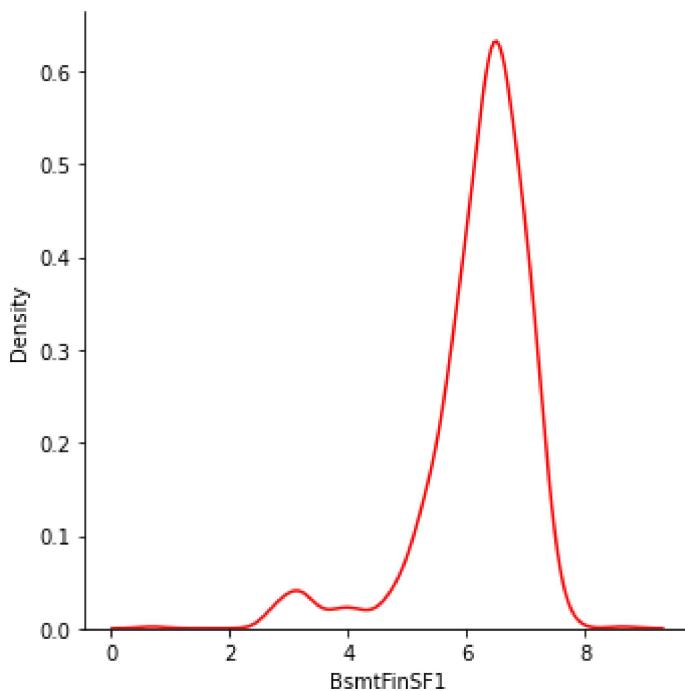




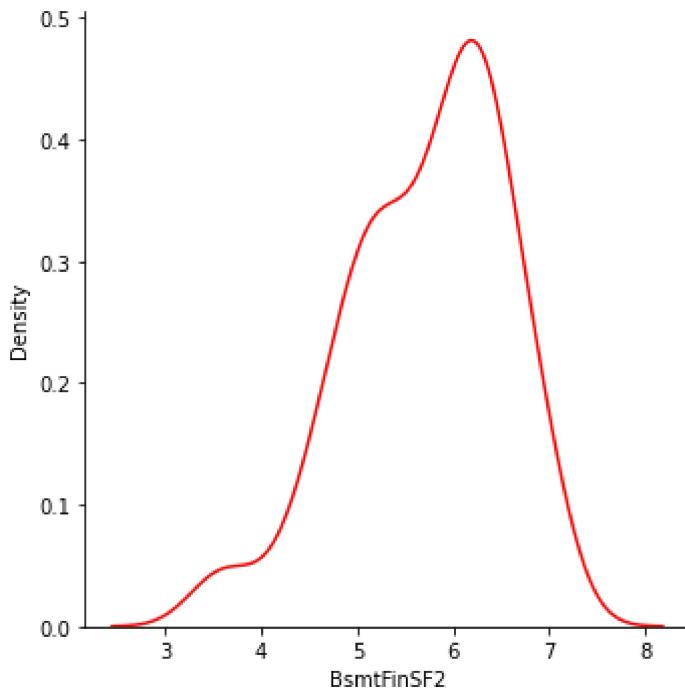
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



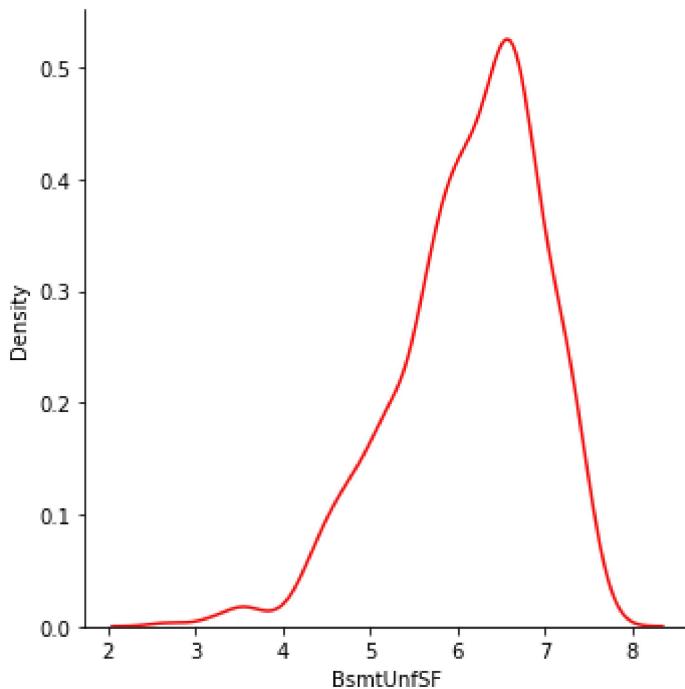
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



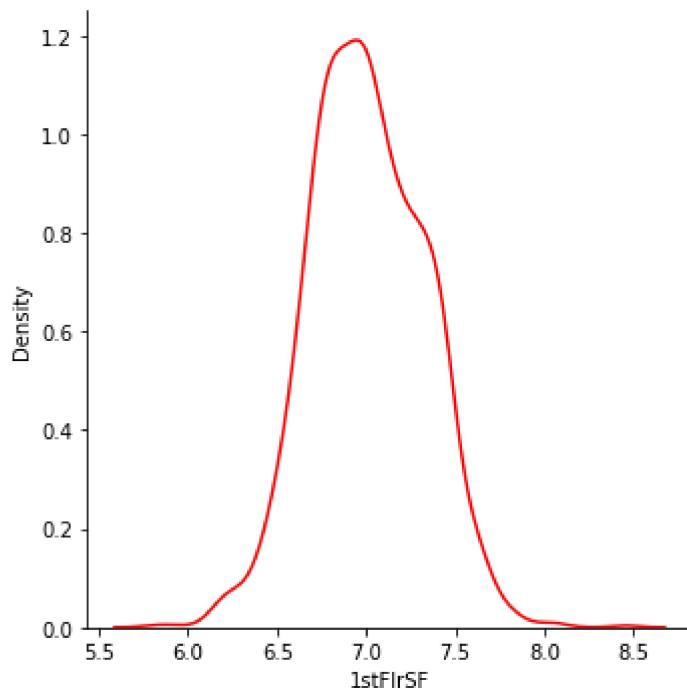
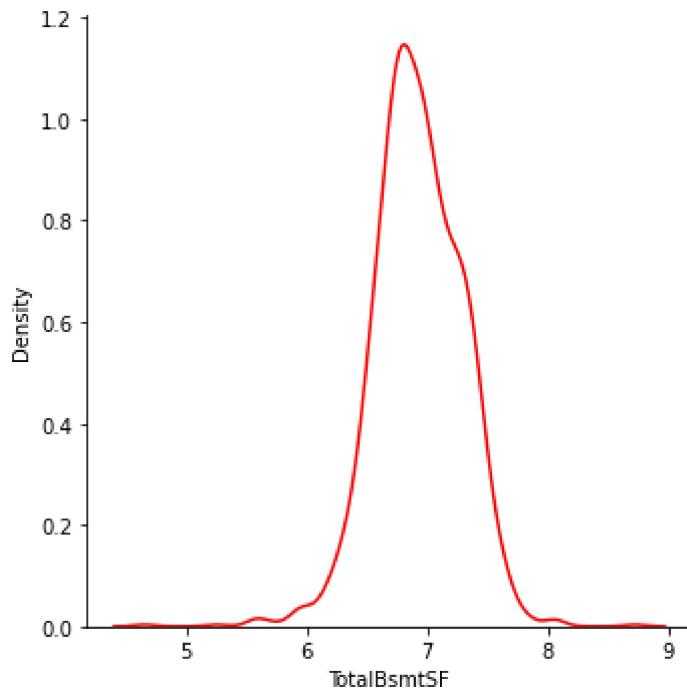
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



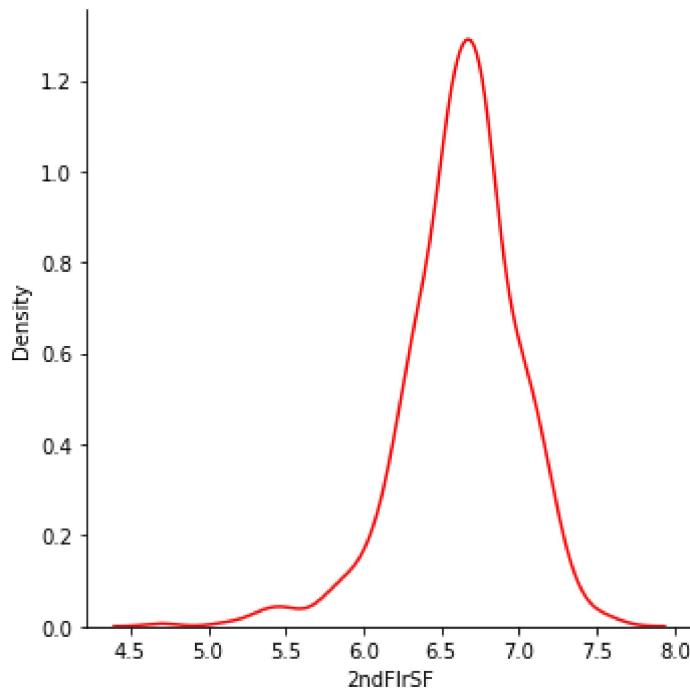
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



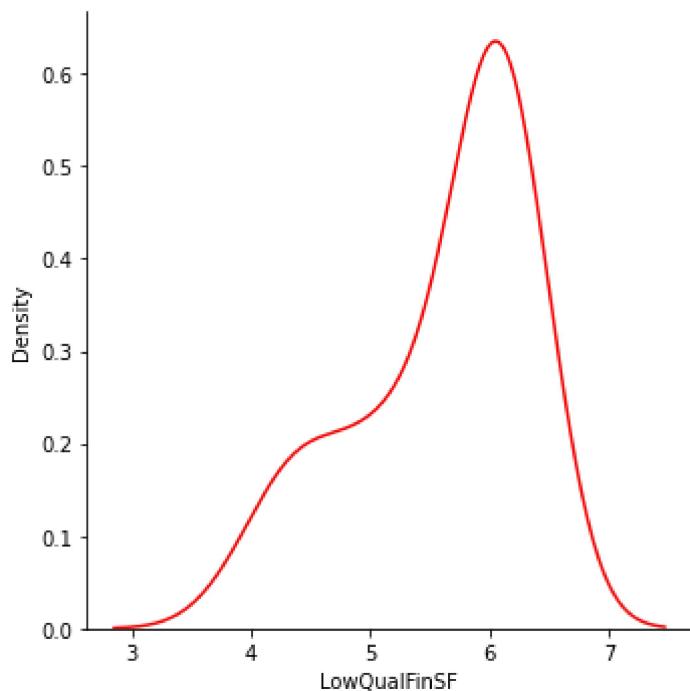
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```

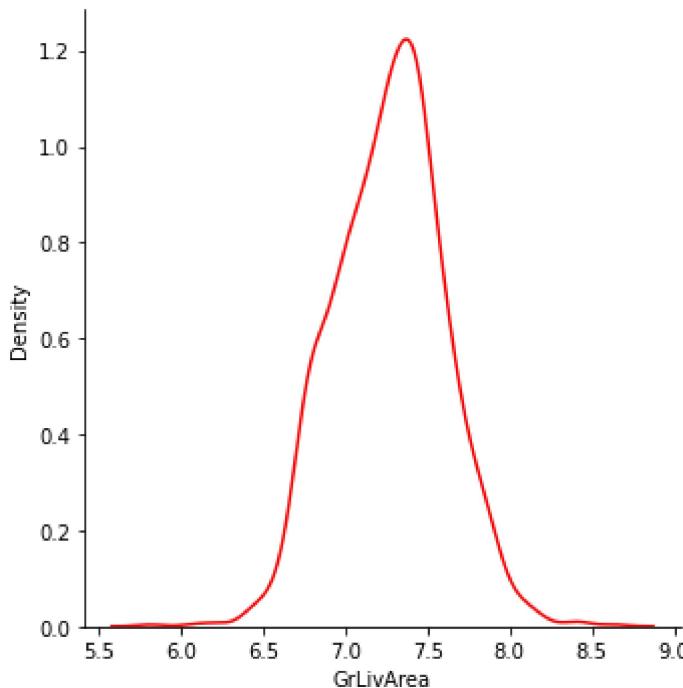


```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

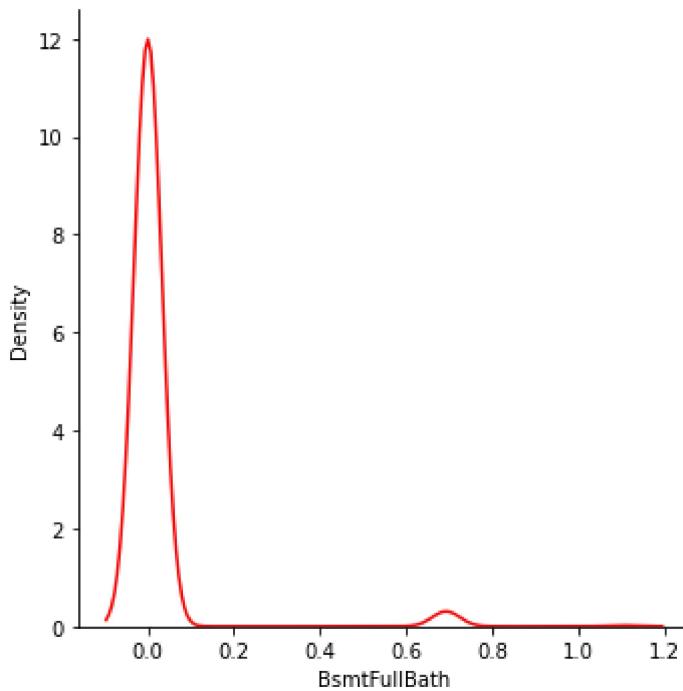


```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
    result = getattr(ufunc, method)(*inputs, **kwargs)
```

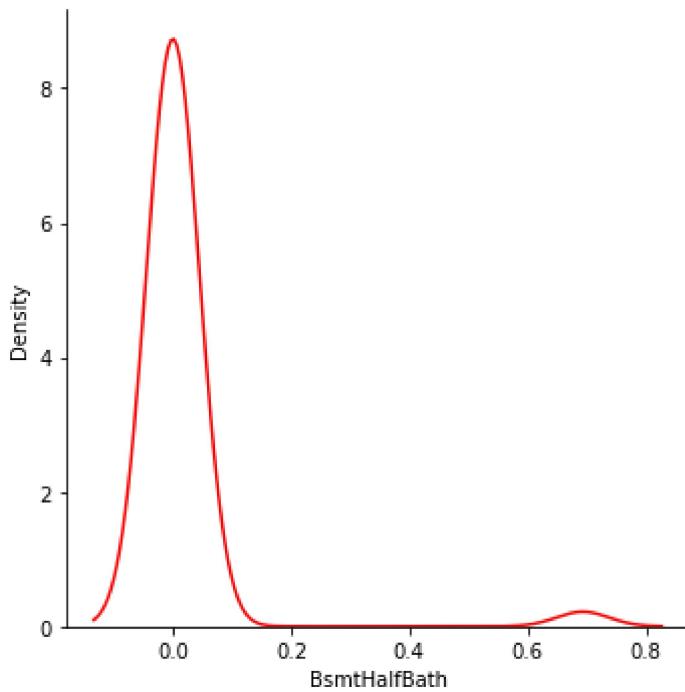




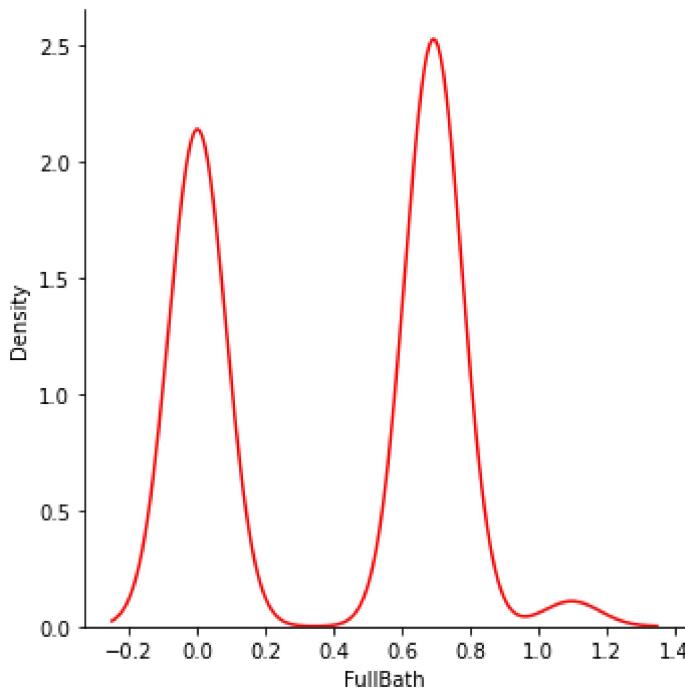
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



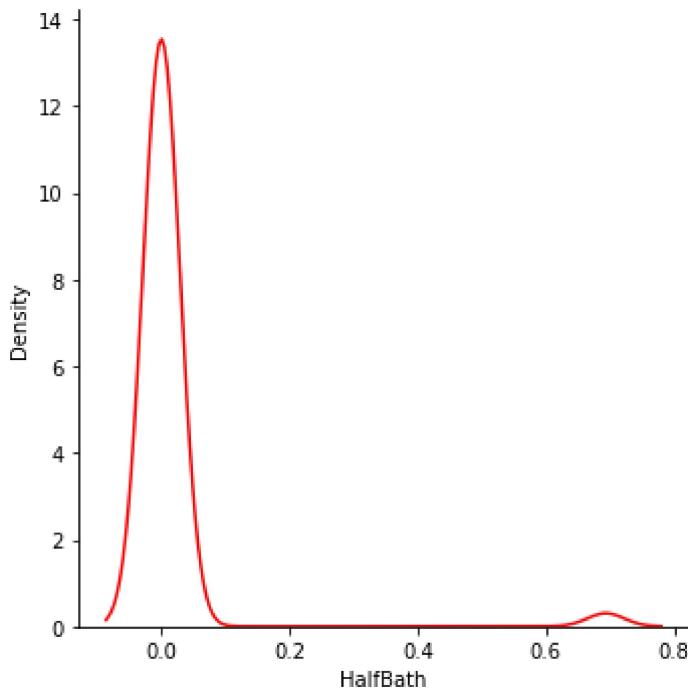
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



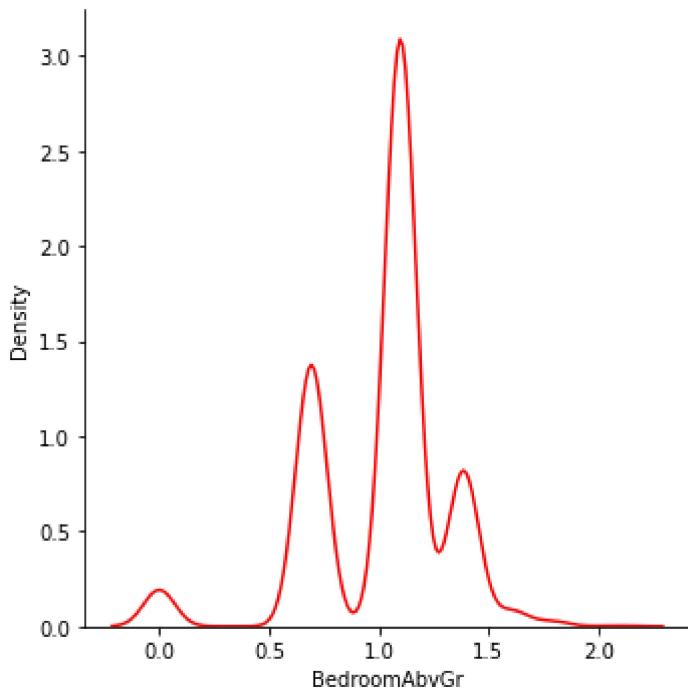
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



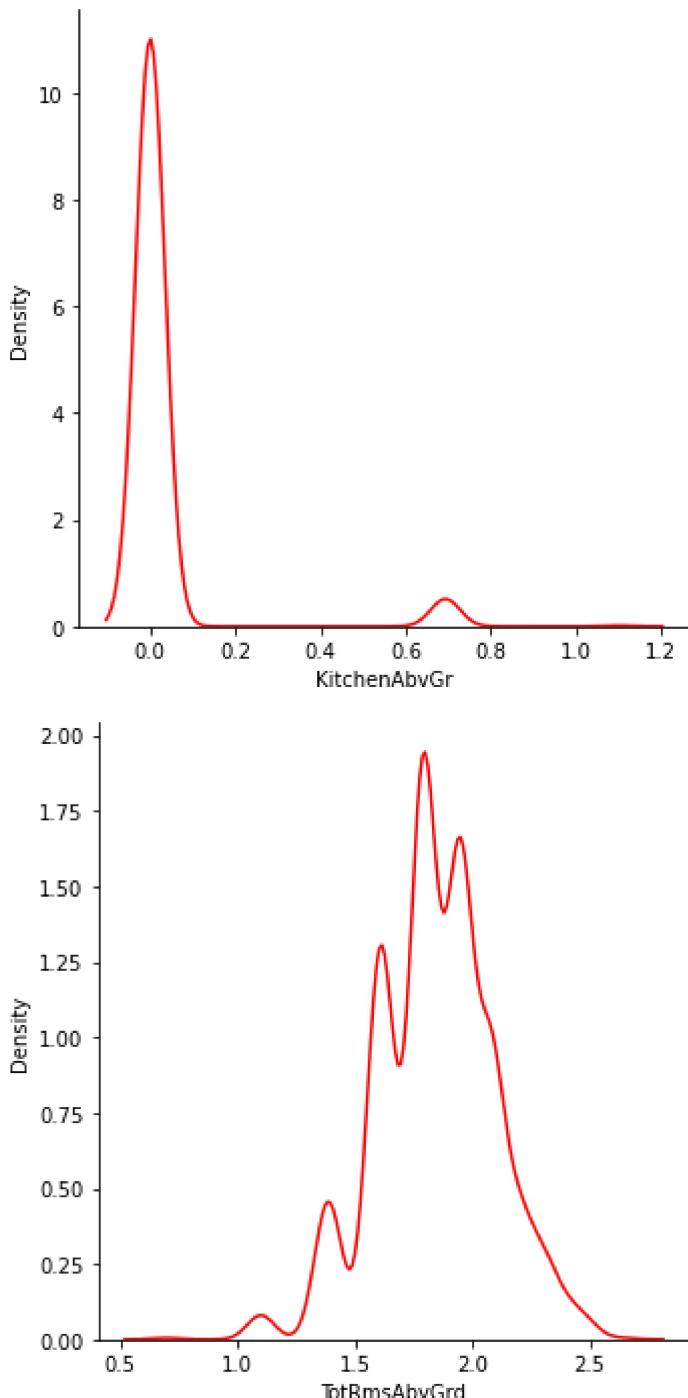
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



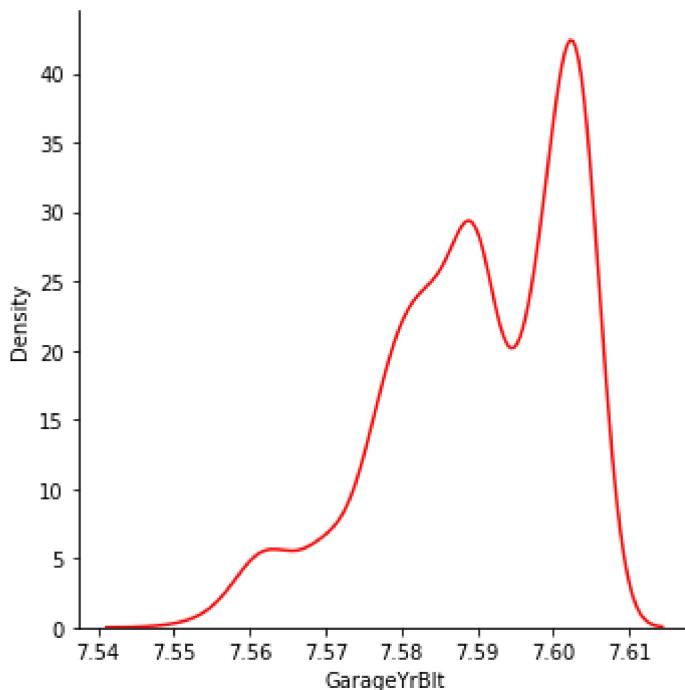
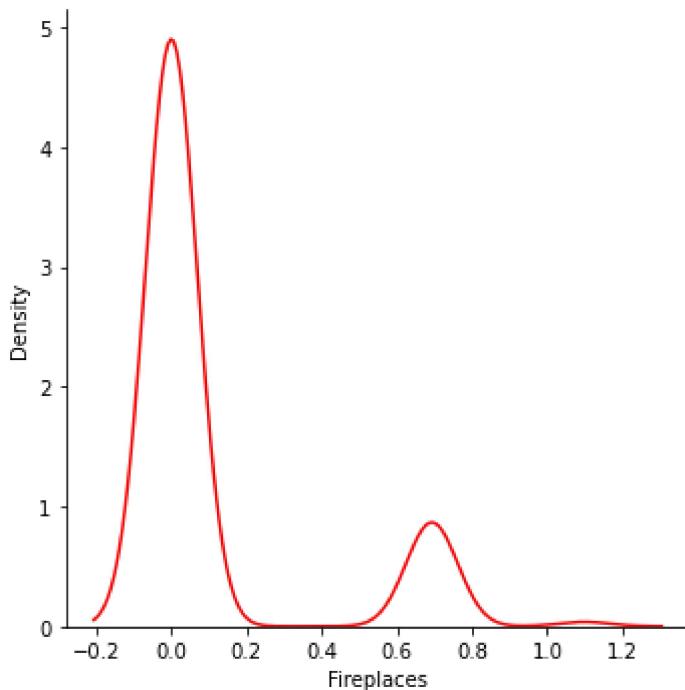
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



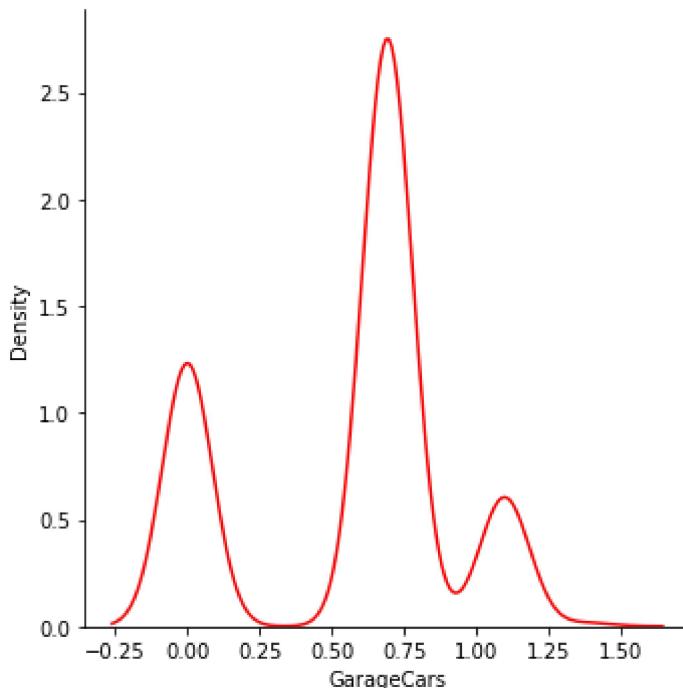
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



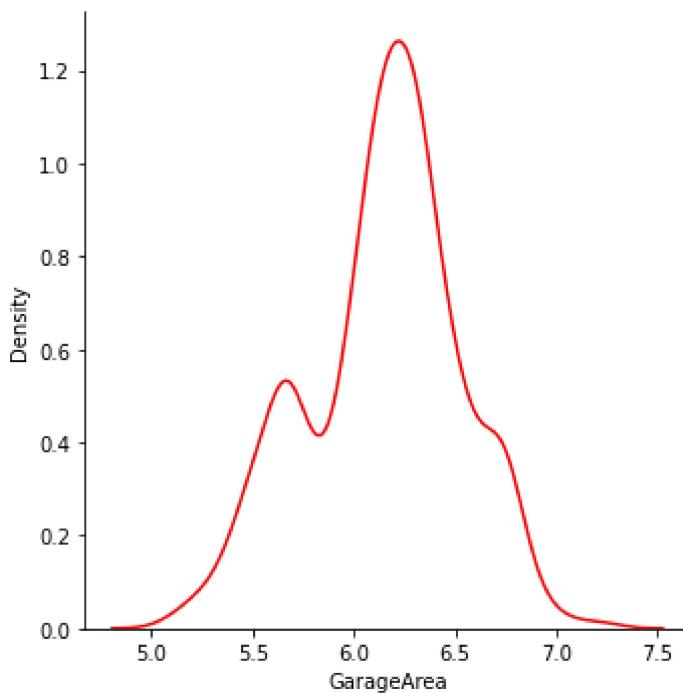
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



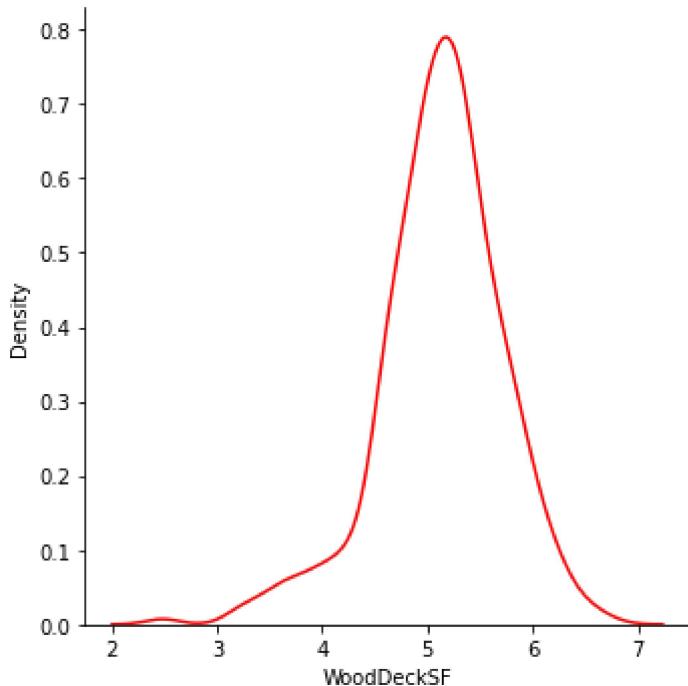
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



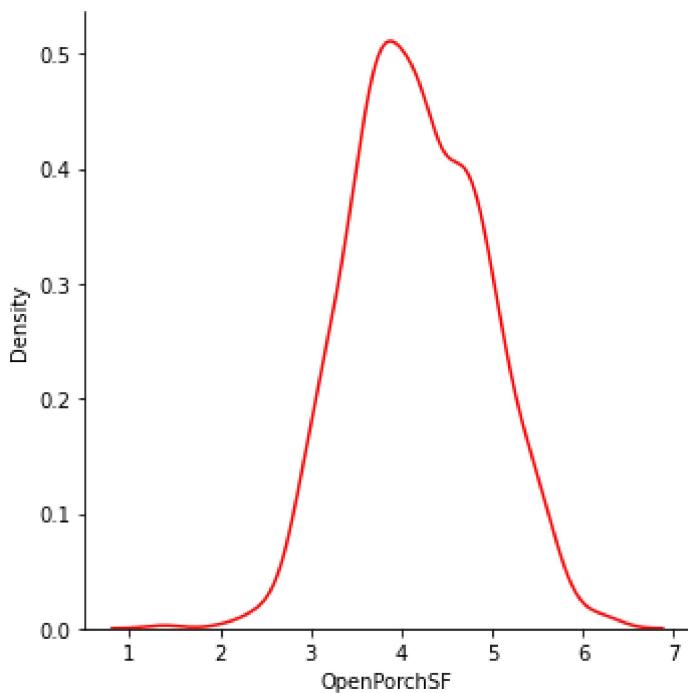
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



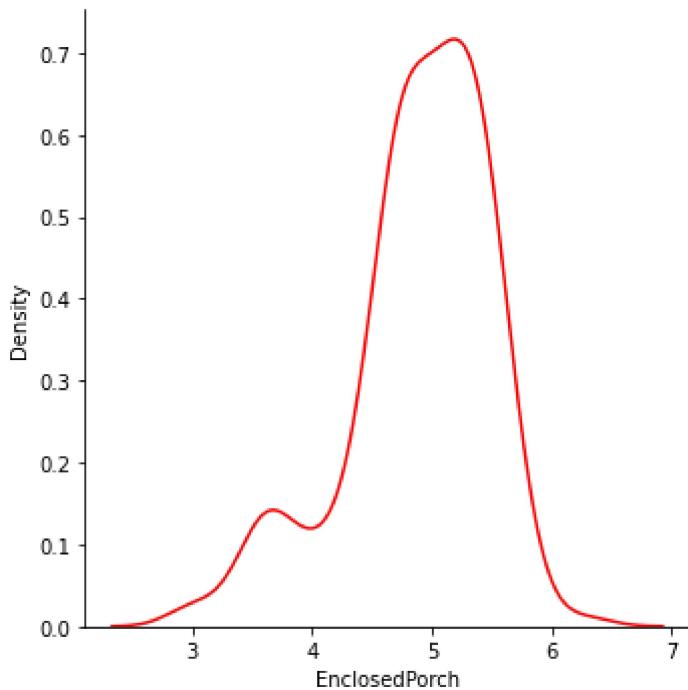
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



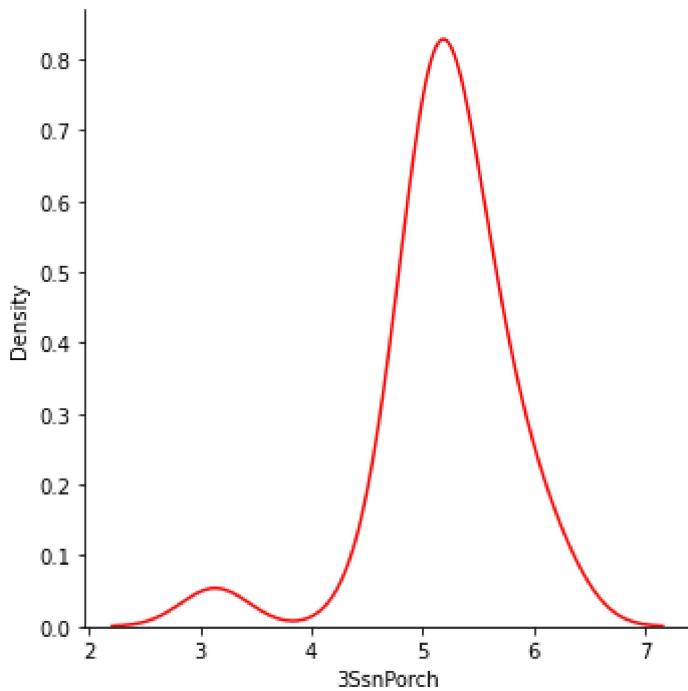
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



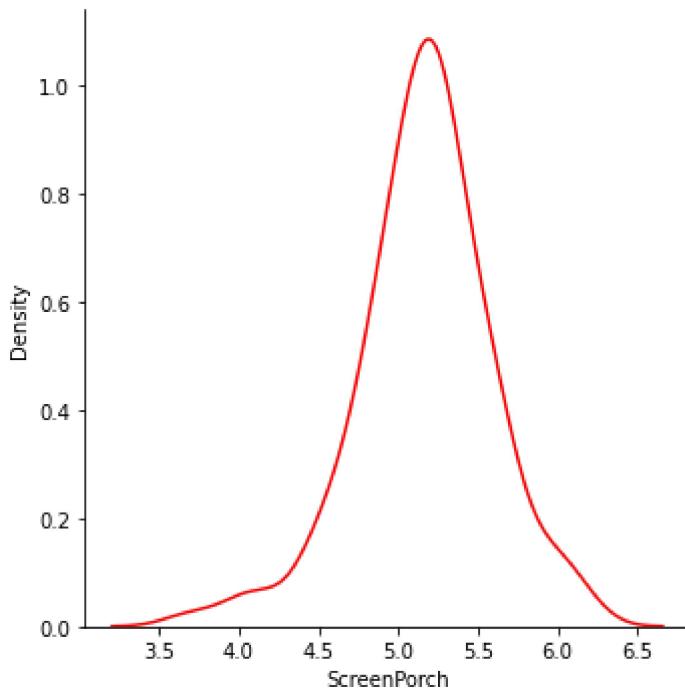
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



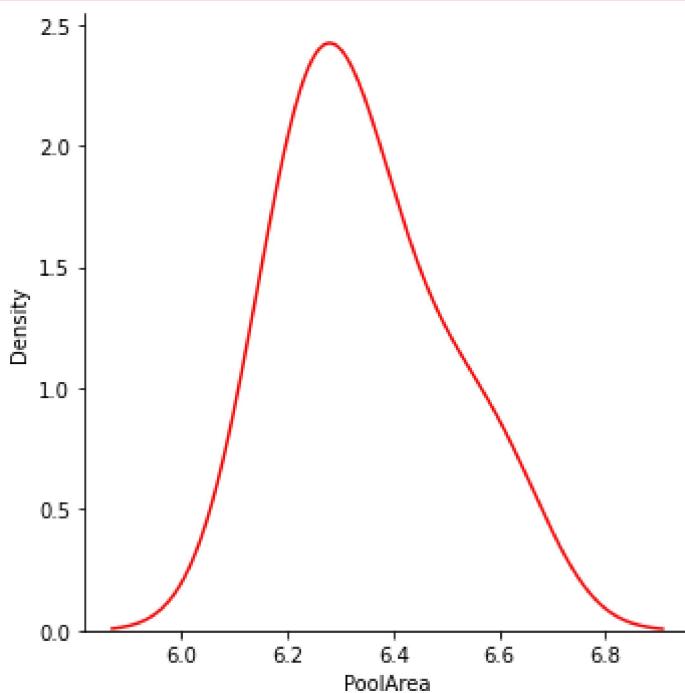
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



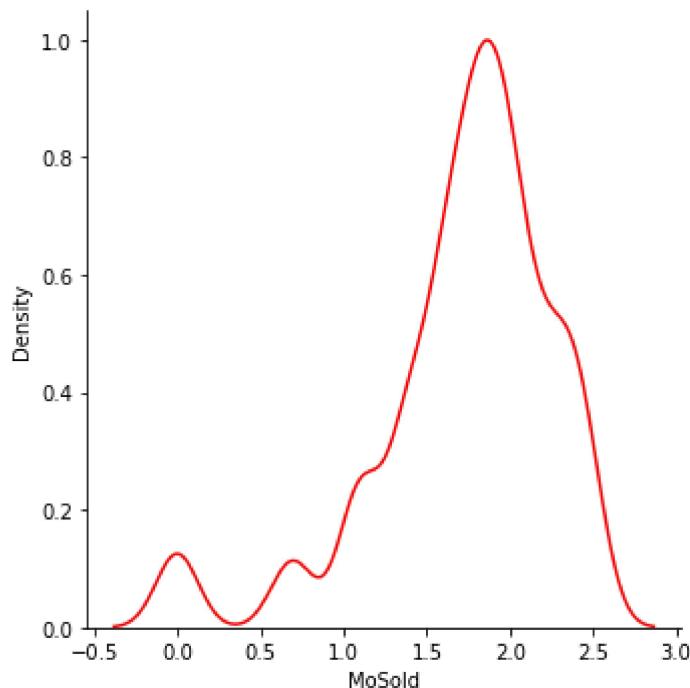
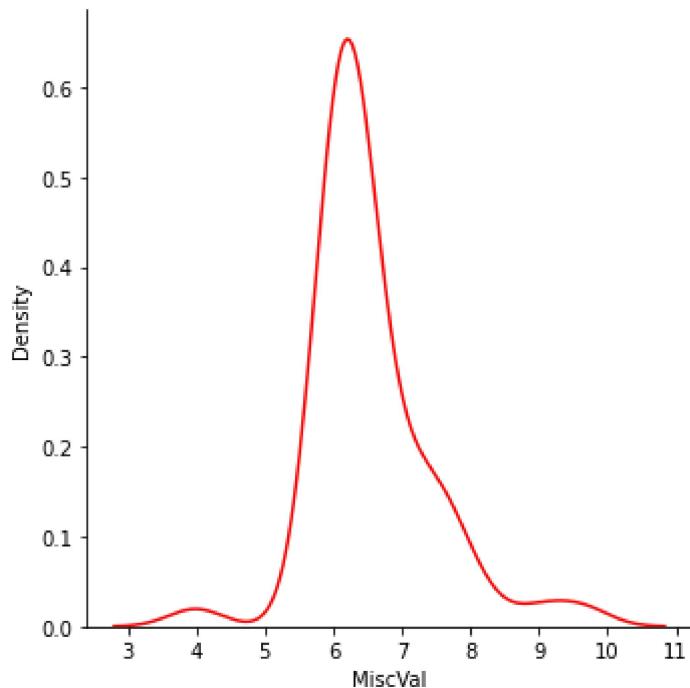
```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```

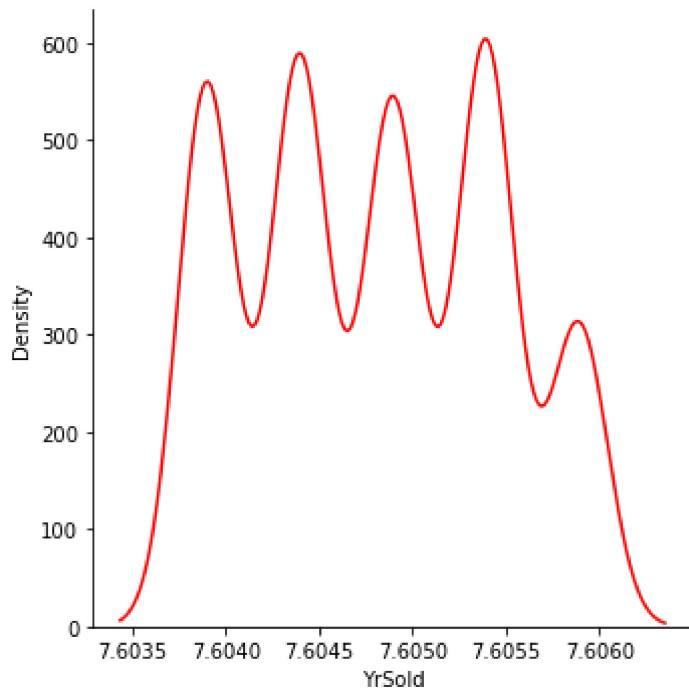


```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```



```
C:\Users\460379\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: divide by zero encountered in log  
    result = getattr(ufunc, method)(*inputs, **kwargs)
```





```
In [63]: # We have below categorical features in this variable
```

```
categorical_features = [features for features in categorical_features if features not in numerical_features]
```

```
Out[63]: ['MSZoning',
 'Street',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'SaleType',
 'SaleCondition']
```

```
In [64]: # Let us find the unique values in the categorical features
```

```
for features in categorical_features:
    df1 = df.copy()
    print(features, df1[features].unique())
```

```

MSZoning ['RL' 'RM' 'C (all)' 'FV' 'RH']
Street ['Pave' 'Grvl']
LotShape ['Reg' 'IR1' 'IR2' 'IR3']
LandContour ['Lvl' 'Bnk' 'Low' 'HLS']
Utilities ['AllPub' 'NoSeWa']
LotConfig ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
LandSlope ['Gtl' 'Mod' 'Sev']
Neighborhood ['CollCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst' 'NWAmes'
  'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
  'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
  'Blmgtn' 'BrDale' 'SWISU' 'Blueste']
Condition1 ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
Condition2 ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RRAn' 'RRAe']
BldgType ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
HouseStyle ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf' '2.5Fin']
RoofStyle ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
RoofMatl ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll'
  'ClyTile']
Exterior1st ['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFace' 'WdShing' 'CemntBd'
  'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
  'CBlock']
Exterior2nd ['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywood' 'Wd Sdng' 'CmentBd'
  'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone'
  'Other' 'CBlock']
MasVnrType ['BrkFace' 'None' 'Stone' 'BrkCmn']
ExterQual ['Gd' 'TA' 'Ex' 'Fa']
ExterCond ['TA' 'Gd' 'Fa' 'Po' 'Ex']
Foundation ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
BsmtQual ['Gd' 'TA' 'Ex' 'Fa']
BsmtCond ['TA' 'Gd' 'Fa' 'Po']
BsmtExposure ['No' 'Gd' 'Mn' 'Av']
BsmtFinType1 ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' 'LwQ']
BsmtFinType2 ['Unf' 'BLQ' 'ALQ' 'Rec' 'LwQ' 'GLQ']
Heating ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
HeatingQC ['Ex' 'Gd' 'TA' 'Fa' 'Po']
CentralAir ['Y' 'N']
Electrical ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix']
KitchenQual ['Gd' 'TA' 'Ex' 'Fa']
Functional ['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']
GarageType ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' 'Basment' '2Types']
GarageFinish ['RFn' 'Unf' 'Fin']
GarageQual ['TA' 'Fa' 'Gd' 'Ex' 'Po']
GarageCond ['TA' 'Fa' 'Gd' 'Po' 'Ex']
PavedDrive ['Y' 'N' 'P']
SaleType ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']
SaleCondition ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']

```

In [65]: # Let us perform encoding technique on categorical features

```

#for features in categorical_features:
    #df1=df.copy()
    #print(pd.get_dummies(df1[features]))

```

In [66]: #LE = LabelEncoder()
#LE.fit\_transform(df1['HouseStyle'])

In [67]: #L = []
#for features in categorical\_features:
 #L.append(LE.fit\_transform(df[features]))

```
#df[ ['col1', 'col2']].apply(LabelEncoder().fit_transform)
```

In [68]: df

Out[68]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotC
0	60	RL	65.0	8450	Pave	Reg	lsl	AllPub	lsl
1	20	RL	80.0	9600	Pave	Reg	lsl	AllPub	lsl
2	60	RL	68.0	11250	Pave	IR1	lsl	AllPub	lsl
3	70	RL	60.0	9550	Pave	IR1	lsl	AllPub	Co
4	60	RL	84.0	14260	Pave	IR1	lsl	AllPub	lsl
...	...	...	...	...	...	...	...	...	...
1455	60	RL	62.0	7917	Pave	Reg	lsl	AllPub	lsl
1456	20	RL	85.0	13175	Pave	Reg	lsl	AllPub	lsl
1457	70	RL	66.0	9042	Pave	Reg	lsl	AllPub	lsl
1458	20	RL	68.0	9717	Pave	Reg	lsl	AllPub	lsl
1459	20	RL	75.0	9937	Pave	Reg	lsl	AllPub	lsl

1460 rows × 75 columns

In [69]:

```
#from sklearn.preprocessing import LabelEncoder
#LE = LabelEncoder()
#LE.fit_transform(df['HouseStyle'])
```

In [70]: df

Out[70]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotC
<b>0</b>	60	RL	65.0	8450	Pave	Reg		AllPub	Impv
<b>1</b>	20	RL	80.0	9600	Pave	Reg		AllPub	Impv
<b>2</b>	60	RL	68.0	11250	Pave	IR1		AllPub	Impv
<b>3</b>	70	RL	60.0	9550	Pave	IR1		AllPub	Con
<b>4</b>	60	RL	84.0	14260	Pave	IR1		AllPub	Impv
...	...	...	...	...	...	...	...	...	...
<b>1455</b>	60	RL	62.0	7917	Pave	Reg		AllPub	Impv
<b>1456</b>	20	RL	85.0	13175	Pave	Reg		AllPub	Impv
<b>1457</b>	70	RL	66.0	9042	Pave	Reg		AllPub	Impv
<b>1458</b>	20	RL	68.0	9717	Pave	Reg		AllPub	Impv
<b>1459</b>	20	RL	75.0	9937	Pave	Reg		AllPub	Impv

1460 rows × 75 columns

In [71]: # Let us perform Ordinal encoding technique to different categories of categorical fea

```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
enc.fit_transform(df[categorical_features])
```

Out[71]: array([[3., 1., 3., ..., 2., 8., 4.],  
 [3., 1., 3., ..., 2., 8., 4.],  
 [3., 1., 0., ..., 2., 8., 4.],  
 ...,  
 [3., 1., 3., ..., 2., 8., 4.],  
 [3., 1., 3., ..., 2., 8., 4.],  
 [3., 1., 3., ..., 2., 8., 4.]])

In [72]: df[categorical\_features] = enc.fit\_transform(df[categorical\_features])

In [73]: df

Out[73]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotC
0	60	3.0	65.0	8450	1.0	3.0	3.0	3.0	0.0
1	20	3.0	80.0	9600	1.0	3.0	3.0	3.0	0.0
2	60	3.0	68.0	11250	1.0	0.0	3.0	3.0	0.0
3	70	3.0	60.0	9550	1.0	0.0	3.0	3.0	0.0
4	60	3.0	84.0	14260	1.0	0.0	3.0	3.0	0.0
...	...	...	...	...	...	...	...	...	...
1455	60	3.0	62.0	7917	1.0	3.0	3.0	3.0	0.0
1456	20	3.0	85.0	13175	1.0	3.0	3.0	3.0	0.0
1457	70	3.0	66.0	9042	1.0	3.0	3.0	3.0	0.0
1458	20	3.0	68.0	9717	1.0	3.0	3.0	3.0	0.0
1459	20	3.0	75.0	9937	1.0	3.0	3.0	3.0	0.0

1460 rows × 75 columns

In [74]: # Let us now split our independant and dependant feature

```
X = df.iloc[:, :-1]
y = df["SalePrice"]
```

In [75]: y

Out[75]:

0	208500
1	181500
2	223500
3	140000
4	250000
	...
1455	175000
1456	210000
1457	266500
1458	142125
1459	147500

Name: SalePrice, Length: 1460, dtype: int64

In [76]: # Lets us now split the data into Train , Test and split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=10, test_size=0.33)
```

In [77]: # Let us scale the data using standard scaler

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [78]: # Let us perform Linear regression model

```
from sklearn import datasets, linear_model, metrics
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
```

Out[78]: LinearRegression()

In [79]: # regression coefficients

```
print('Coefficients: ', reg.coef_)
```

```
Coefficients: [-8.02842258e+03 -1.30415289e+03 -4.45556621e+03  5.06428659e+03
 2.96654302e+03 -1.66404774e+03  3.75024347e+03 -1.87927812e+03
 -7.71131057e+02  2.14522721e+03  3.02670519e+03 -2.80022905e+02
 -2.31866127e+03 -5.92895368e+02 -3.29150333e+03  1.71368114e+04
 4.91738262e+03  4.83601680e+03  3.47105378e+02  1.07906518e+03
 4.27481472e+03 -2.13482189e+03  8.07156514e+02  1.75494124e+03
 5.14577273e+03 -5.65028043e+03  4.07254831e+02  9.14680584e+02
 -8.36776310e+03  3.16458056e+03 -4.29428082e+03 -3.88294283e+03
 8.45520106e+16  1.14452908e+03  2.70086559e+16  7.88208156e+16
 -8.00419400e+16 -6.04294570e+02 -1.27004108e+03  4.35744917e+02
 -9.47544641e+02  1.24043369e+16  1.38119351e+16  1.65550085e+15
 -1.68554285e+16  4.06218558e+03 -2.82834176e+02  9.04550485e+02
 -2.64484819e+03 -1.59031128e+03 -1.61104258e+03 -8.86231157e+03
 4.68499508e+03  2.34642364e+03  2.22431844e+03 -1.68012888e+03
 -5.78573739e+02  1.15063849e+02  9.98676083e+03 -1.27944500e+03
 7.95640322e+01 -5.50338354e+00  6.29555227e+02  2.80547782e+03
 -8.32280714e+02  5.59063905e+02  8.90210730e+02  2.87746676e+03
 5.71580183e+02 -1.95128336e+02 -2.39284632e+02 -1.05202798e+03
 -7.72680934e+02  2.49861534e+03]
```

In [80]: # variance score: 1 means perfect prediction

```
print('Variance score: {}'.format(reg.score(X_test, y_test)))
```

Variance score: 0.8379855358615005

In [83]: ## Prediction for the test data

```
reg_pred= reg.predict(X_test)
reg_pred
```

```
Out[83]: array([169879.67765076, 202836.71882716, 125885.01609103, 139891.32135079,
 304716.02869594, 265353.56040633, 245777.22255767, 93576.42535458,
192864.13763517, 166663.69315385, 158652.83976643, 170670.35023447,
264378.67013601, 119384.09826124, 236866.93679072, 186434.98765883,
270649.1143477 , 134353.67490588, 226032.84589248, 201577.87258518,
407751.62043951, 118380.82483891, 131569.90571031, 105269.19656937,
214008.31262086, 173284.05767226, 146299.44406133, 85168.16201312,
165172.5207528 , 206649.38639147, 114414.98574084, 155973.89501611,
135962.38222017, 137818.67081111, 137834.96453475, 45456.00345128,
135504.97408375, 215716.82676451, 322903.44386826, 138016.3994178 ,
227163.21464488, 122202.9588629 , 88501.94807457, 205104.98242678,
221352.76735728, 247930.13875453, 262282.45246748, 137681.03695603,
126975.77257795, 163320.97619105, 197103.08473885, 16435.859759 ,
195820.32755503, 116480.62554441, 226108.7922838 , 199903.6423214 ,
204120.01594796, 187795.61021642, 131312.74642694, 137474.48777328,
117272.39566832, 232473.5386909 , 129134.10918144, 235013.24927876,
218957.86962585, 211876.50132565, 232316.01627833, 183000.67981018,
128731.1880826 , 180614.09320099, 119261.07702254, 114940.76044865,
123712.79431986, 107642.33868016, 177259.28803245, 251713.30063042,
225785.64126565, 75790.18926545, 208169.17135339, 202893.73269634,
274218.35793686, 229917.62781493, 209997.4607751 , 224029.10782987,
140360.99657375, 136417.84002893, 159714.41627222, 174277.10762714,
254836.19462764, 150937.89770615, 165247.02905438, 113369.995982 ,
58528.98889896, 98336.83021594, 275656.1598368 , 169639.22528958,
169607.88567768, 223367.67535269, 162635.23138661, 259967.97004003,
232005.54567524, 190448.72444354, 41657.76224074, 116722.98287776,
111532.65512834, 318378.75992628, 194345.98247319, 206082.11813005,
78646.64429678, 120208.93763603, 170195.96669325, 218355.37040641,
168285.82003162, 135048.21670259, 112037.40539995, 133496.74501344,
339767.34778116, 323005.8281896 , 143691.99783672, 98594.19969554,
160897.92181307, 341263.23526809, 112364.76601672, 153653.1799585 ,
288983.0925667 , 174698.88802578, 107954.84719975, 174224.22585663,
249366.28702768, 184346.07220255, 121073.99058816, 180808.1574266 ,
171984.39759031, 237549.42408081, 97576.82742378, 219448.33605663,
36151.29513385, 217555.09406414, 90527.2147213 , 184364.9636153 ,
215286.38721572, 109332.85206443, 151264.30928943, 347265.15398399,
196458.97556711, 21933.42023289, 199280.83884741, 173727.61344742,
282591.71213228, 191956.45058482, 253788.95399869, 207393.82965694,
221831.61187349, 179802.76159979, 124535.42613024, 126887.68350945,
152141.96802712, 126912.27847937, 198434.00749318, 376590.94041925,
47781.57447545, 326479.06459216, 296799.52000468, 183196.24425754,
169303.61159456, 236553.63526222, 197836.20115921, 213111.35646345,
301335.73938682, 201099.78019791, 106657.48291983, 234821.41666958,
124298.33529454, 229972.22792706, 92209.35893438, 188836.52250739,
136141.56811038, 244108.0788628 , 315797.41679309, 55166.78868923,
126739.86684723, 98846.10672416, 186620.33465525, 194906.69517479,
95135.29213178, 397830.64673868, 241274.40472462, 156988.59424564,
135851.01594277, 129390.27390871, 126832.25914519, 123833.69435619,
119115.78746293, 156758.40891796, 249475.82806841, 164314.714941 ,
145039.00156703, 121261.00409869, 247631.91803714, 238977.37266166,
187202.06050911, 354533.67469218, 175358.66250543, 353297.67846059,
247875.94634005, 102707.86541992, 173658.28948084, 385735.41728374,
112168.20313391, 87324.70683671, 100338.44850238, 70730.96234285,
85808.42459401, 103935.56657238, 172810.17129356, 165706.36321187,
256163.06396577, 88808.93634258, 31606.54667461, 154721.71039684,
119981.14105143, 171164.83976643, 82174.82839406, 254024.18646941,
100414.7041445 , 116425.17429052, 213438.17029173, 161306.34144256,
293875.6729384 , 112792.19181017, 253852.29165028, 146746.0063159 ,
136134.77178162, 248811.57606372, 100980.34379884, 114561.19261343,
244460.61772292, 156012.30457073, 284461.74357532, 115674.73595888,
```

131972.68843439, 245699.82296817, 191054.32252336, 318617.613837 ,  
366831.98887258, 210065.10178478, 160699.45625221, 271804.563457 ,  
157021.45805714, 226986.67833546, 211000.67440996, 212043.90104176,  
100399.97804119, 116914.91367969, 149597.58099418, 188637.17657576,  
329834.64028511, 115670.03562361, 186291.08505907, 123335.68412916,  
88780.18633932, 198613.14247814, 125696.37827621, 82502.00356511,  
128626.89375971, 88703.71985148, 129329.65814117, 119148.59815623,  
115610.05489939, 141634.9302176 , 131247.01186334, 90980.47799731,  
109262.91859726, 127441.33513791, 171242.83976643, 245623.13332034,  
160532.08134809, 198035.0964392 , 144537.29448388, 150566.41710021,  
90271.40675925, 85625.71582691, 281597.23669479, 186737.66150354,  
197743.7620225 , 312440.75785277, 151496.34397073, 71914.45471514,  
191192.61051204, 362925.70716759, 195068.73740236, 136308.54931632,  
164558.05364586, 141322.15266809, 239306.72835314, 219281.86520366,  
207207.99355407, 156844.77957773, 118141.98510833, 220617.51428717,  
101813.49099012, 152146.5021053 , 109754.23485958, 205204.40140398,  
227864.62614414, 299660.93853268, 135119.11478153, 74770.98964356,  
213184.93328013, 263504.52859672, 178182.90799443, 146243.61961597,  
218675.93528166, 295607.49839148, 189824.9663107 , 284634.32017221,  
302467.90234037, 157869.95303218, 135778.41340282, 72548.32397237,  
191679.61739447, 198223.54579497, 151026.50989147, 273658.40228427,  
193026.01221153, 176838.53596226, 215492.63754601, 65747.09715214,  
136700.57412835, 235882.31020593, 186026.8554784 , 177763.92685408,  
119970.87272899, 160463.03372489, 80103.85334854, 259627.54913857,  
257887.02622194, 185930.31460307, 175524.23878992, 235309.6381574 ,  
121331.23976692, 30624.88190756, 234723.64367082, 184168.91114368,  
163258.15878047, 116653.25881981, 115673.41234605, 305129.68027591,  
157193.98880261, 394605.71078699, 142052.30419309, 160040.75715786,  
227342.54056204, 206360.76323866, 192600.10429217, 133955.69453814,  
127891.5148906 , 190187.67160382, 157294.15053268, 161169.26485789,  
217424.78817829, 106103.74217266, 294259.0850123 , 142224.19527097,  
72590.55623516, 209725.84847162, 121604.3498697 , 246669.8214083 ,  
219003.8945302 , 192706.50917119, 200013.01146863, 142613.04885301,  
197837.75389424, 258473.94303903, 107064.37111048, 41126.30661456,  
110277.06037373, 242049.78005103, 214281.68914093, 241338.9417435 ,  
210788.25715596, 195153.61792032, 127427.2684885 , 183247.78034986,  
288066.7912071 , 188173.93514896, 106466.70297372, 124068.99209548,  
118943.98168689, 297778.97274524, 151491.12495857, 121053.01258968,  
111536.29420661, 166103.90141224, 210379.49521756, 226360.49033794,  
122575.73029784, 224273.94141078, 283185.00637834, 150625.14352299,  
159502.88884152, 199743.30101034, 250931.35256911, 192787.58567159,  
141131.54937978, 138863.64861959, 97860.86727899, 89239.20418026,  
72270.20443707, 173083.37876281, 148986.63971504, 151284.13773233,  
170373.67458343, 110075.22377071, 217419.2007229 , 121436.84472144,  
330164.54235428, 186425.42684282, 269730.19719293, 107485.44120604,  
172761.58275987, 170320.71229536, 206798.63185591, 108256.11533138,  
146172.70033544, 123663.03651393, 47038.13572632, 202323.65937838,  
116062.95879221, 173210.83306046, 166193.92513906, 183584.84889143,  
254120.35601557, 139042.30576112, 86737.12219712, 100820.82660654,  
294144.18527072, 94309.79863152, 83531.50604725, 124775.4812202 ,  
312049.45124168, 209387.11229767, 190507.70428484, 257775.36928227,  
282202.29124943, 73574.04660776, 93206.09692779, 112244.6501356 ,  
400997.40114397, 130600.84280596, 198772.58698954, 213828.16026779,  
169301.61926658, 152243.5606628 , 384360.47114775, 254013.89585626,  
190541.37652111, 249779.36562746, 167029.33651229, 124309.80789881,  
118509.87258465, 84581.06220352, 202462.88279683, 140273.84704608,  
276013.11233187, 118131.7214279 , 357659.37177285, 135457.05321943,  
202691.15699164, 263128.5713006 , 234446.45366865, 106853.1691172 ,  
139343.26133755, 330059.26331257, 111167.92785442, 160763.83774243,

```
274580.65640772, 101798.31237547, 120104.74180929, 231751.80475242,  
143072.80683288, 155703.79523238])
```

```
In [84]: from sklearn.metrics import r2_score  
score=r2_score(y_test,reg_pred)  
print(score)  
  
0.8379855358615005
```

```
In [85]: mae = metrics.mean_absolute_error(y_test, reg_pred)  
mse = metrics.mean_squared_error(y_test, reg_pred)  
rmse = np.sqrt(mse) # or mse**(.5)  
r2 = metrics.r2_score(y_test, reg_pred)  
AR2 = 1 - (1-score)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)  
  
print("Results of sklearn.metrics:")  
print("MAE:", mae)  
print("MSE:", mse)  
print("RMSE:", rmse)  
print("R-Squared:", r2)  
print("Adjusted R2:", AR2)  
  
Results of sklearn.metrics:  
MAE: 22003.07853147816  
MSE: 1045532579.5643027  
RMSE: 32334.696218834386  
R-Squared: 0.8379855358615005  
Adjusted R2: 0.8085283605635915
```

```
In [86]: ## Ridge  
from sklearn.linear_model import Ridge  
ridge=Ridge()
```

```
In [87]: ridge.fit(X_train,y_train)
```

```
Out[87]: Ridge()
```

```
In [88]: ridge.predict(X_test)
```

```
Out[88]: array([169861.26079322, 202859.83422442, 125779.04396378, 139940.70732908,
 304622.09942792, 265284.33959488, 245769.74555628, 93554.80090498,
192969.16553045, 166778.09604315, 158652.36265882, 170545.22578508,
264112.20638677, 119088.42588348, 237008.69031114, 186241.38572623,
270815.30224717, 134494.98081893, 225854.03804266, 201509.21659851,
407743.99253474, 118529.08533536, 131426.59017919, 105314.35304239,
213703.0236241 , 173319.25030645, 146400.90669913, 85186.52819523,
165198.25994201, 206618.83927304, 114403.60018082, 155983.7948401 ,
135930.42910336, 137759.21903769, 137743.1031343 , 45915.93114292,
135665.47671313, 215708.85770306, 322740.9972112 , 138011.56457331,
227061.75658511, 122026.53195157, 88491.6998667 , 205154.84461755,
221332.84567747, 247629.51745907, 262176.77821743, 137412.65262405,
126850.2213213 , 163442.5212533 , 197092.3600316 , 16380.12185198,
195798.45591373, 116511.62238326, 226140.88361325, 199779.27295563,
204138.52791971, 187756.64652741, 131326.91585459, 137410.69626578,
117256.56697666, 232742.44107621, 129214.67336748, 234977.4902731 ,
219041.63135517, 211577.4563861 , 232424.44529134, 182924.34251549,
128759.08421443, 180615.77587824, 119175.0524257 , 115020.27650312,
123743.26181369, 107621.25912156, 177169.62306699, 251594.24415384,
225888.5552099 , 75826.79424477, 208171.74819806, 202874.27501145,
274188.0623218 , 229933.19705564, 209925.44558439, 224088.3186347 ,
140296.911174 , 136285.91546343, 159769.30314021, 174354.28205098,
254944.36811418, 150920.93798896, 165089.69696693, 113576.21140761,
58582.85629085, 98426.25616911, 275660.15762805, 169688.10246471,
169747.19792686, 223361.18285541, 162809.66610795, 259803.66701567,
231991.11369522, 190223.29382441, 42050.32521444, 116605.7468899 ,
111512.00108547, 318296.23751433, 194407.81062734, 206037.31040342,
78775.64561693, 120246.15378284, 170278.68293474, 218325.49052823,
167931.13458039, 135181.98328812, 112058.00465481, 133738.78623291,
339697.56263409, 322812.28970889, 143535.95247404, 98679.61547251,
160944.78862695, 341377.35584338, 112314.72804045, 154065.90501679,
288916.07120808, 174711.28127137, 108169.15971405, 174251.98031552,
249286.9050507 , 184395.59864876, 121126.47489322, 180823.95197775,
171914.83385465, 237518.21047752, 97833.80111364, 219468.15486222,
35678.17398957, 217668.93448371, 90408.23422169, 184673.2730993 ,
215453.6169641 , 109392.51559566, 151199.55824934, 347483.47306151,
196604.31103574, 22565.31604965, 199315.74591285, 173645.84819861,
282575.3916898 , 191749.71439772, 253880.17549304, 207405.0331836 ,
221755.27161234, 179853.58712657, 124464.06622523, 126883.41590131,
152253.70761281, 126989.39662622, 198249.70695824, 376409.75960332,
47459.01349401, 326356.90670949, 296635.96100745, 183219.70994369,
169251.01570287, 236434.55599207, 197636.91642021, 213055.7686726 ,
301345.99106376, 200997.53991374, 106643.40421229, 234570.86288906,
124324.61334006, 230015.76352081, 92005.3830493 , 188785.22551594,
136140.72129655, 244184.40914711, 315691.60107494, 54749.99579047,
126820.41491613, 98978.47157256, 186559.44185832, 195014.44939212,
95419.88417122, 397928.60776786, 241092.15866002, 157138.90215545,
135773.26496423, 129146.27779472, 127024.85269722, 123959.79389112,
118659.36158799, 156921.60116623, 249474.88822874, 164397.56838833,
145121.66468254, 121435.61933658, 247584.68542226, 238846.17068691,
187191.97673683, 354377.83509482, 175526.73245768, 353193.75270319,
247987.2967139 , 102688.89171622, 173388.97236455, 385604.6405754 ,
112166.84110322, 86945.15929797, 100335.20100148, 70652.93420893,
85566.01411734, 103914.17293073, 173043.12242156, 165552.42525835,
256203.63356207, 88902.5317068 , 31678.50471154, 154562.03663229,
120050.15585322, 171123.65853255, 81679.44020252, 253810.85587397,
100498.10991038, 116403.24625595, 213456.2139034 , 161339.58782215,
293930.58795566, 112734.48451176, 253602.74905085, 146933.52477147,
136011.13511067, 248563.75703515, 101094.40627341, 114799.65465199,
244378.27851263, 155855.91819599, 284352.33303553, 115755.23317708,
```

131832.58021564, 245773.90805526, 190816.81235402, 318474.031331 ,  
366707.8719521 , 210122.87323261, 160916.88956933, 271685.59060721,  
157050.89759362, 226882.83836257, 210893.37979873, 212072.94521898,  
100248.86259108, 116981.14599455, 149739.61400346, 188580.28101664,  
329810.60906423, 115735.7611861 , 186512.35604122, 123465.40501861,  
88710.15646963, 198626.96455767, 125559.08720941, 82293.03594304,  
128731.88783421, 88700.79398114, 129346.70572812, 119418.39492957,  
115683.76219533, 141665.92330297, 131309.89543795, 90784.88895243,  
109508.33472603, 127578.14502187, 171186.37632405, 245512.73761188,  
160283.57218901, 198053.48842032, 144923.0824544 , 150473.56619637,  
90306.616273 , 85659.78002697, 281628.48115367, 186698.56538925,  
197708.97209706, 312291.71603575, 151626.26870128, 71662.29318507,  
191235.61161676, 362782.40777758, 195181.63372056, 136377.59746329,  
164371.78026548, 141329.13450319, 239169.45986725, 219377.23483182,  
207165.12906581, 156828.30009787, 118264.61831089, 220516.63405284,  
101628.05091267, 152281.77805816, 109691.92969489, 205140.81758375,  
227782.75359595, 299676.55944999, 135063.34711876, 74784.06709397,  
213008.11622521, 263595.25449009, 177918.21996765, 146121.70990098,  
218623.02289307, 295477.06156688, 189759.69771844, 284713.61915623,  
302315.63554328, 157890.8142556 , 135850.14730671, 72481.80128544,  
191467.93238498, 198238.27403025, 151115.40929491, 273674.48005543,  
192993.93843732, 176994.53747928, 215490.50254837, 65811.972738 ,  
136626.82470728, 235850.02332487, 185870.69684984, 177722.23520006,  
120048.29922113, 160488.51818146, 80280.03672125, 259554.59471308,  
257669.71469851, 186045.69854187, 175099.20896641, 235184.88184199,  
121697.94492464, 30685.282826 , 234545.85449672, 184246.16322057,  
163019.43100064, 116841.86383904, 115336.94103272, 304984.26231639,  
156956.22578765, 394476.33370287, 141938.72293722, 160059.96732379,  
227422.90945581, 206271.49104877, 192693.17016834, 133912.84875889,  
127920.11342626, 190137.36722407, 157414.93575123, 161172.72279157,  
217354.85918633, 106289.57650049, 294171.93154752, 142382.30447435,  
72301.88118809, 209652.62459576, 121979.97353517, 246820.53215851,  
219001.47205313, 192756.01540654, 199837.77473665, 142686.38881905,  
197957.36014088, 258662.6151676 , 107011.81751492, 41111.98474555,  
110335.75176869, 242033.32801904, 214667.66016196, 241174.18134271,  
210942.10118313, 195255.19307902, 127544.37053337, 183170.27213702,  
288115.26657927, 188352.44770439, 106520.96204507, 124100.46732093,  
119063.09217015, 297613.8775423 , 151528.94479695, 120968.45658554,  
111396.00795984, 166339.97470863, 210159.06029212, 226328.91389456,  
122993.22520258, 224302.92185033, 283016.8610063 , 150674.37788272,  
159391.71934124, 199676.21353477, 250739.84245228, 192498.10172553,  
141121.50136283, 138773.49116544, 97914.66363699, 89239.1920172 ,  
72392.63115517, 173159.4713772 , 149097.765402 , 151306.30373519,  
170472.63417637, 110163.96860494, 217777.34311061, 121558.06389473,  
330326.91260486, 186444.08847653, 269664.6885224 , 108064.19022859,  
172677.62563322, 170326.80886913, 206756.79988197, 108217.17278621,  
146196.68496243, 123545.97028329, 47205.15683569, 202401.21598457,  
116152.56695511, 173280.25542816, 166171.11111824, 183636.09953334,  
254196.70286024, 139069.24310586, 86798.72303629, 101195.191752 ,  
294119.12887756, 94310.47754803, 83105.38596896, 124748.59010393,  
312165.34947158, 209413.8587867 , 190516.1621417 , 257857.20406406,  
282214.30309493, 73646.27108698, 92889.89290673, 112176.5748283 ,  
400961.04212261, 130698.08091884, 198785.87025689, 213855.54183171,  
169288.63119843, 152070.17049708, 384191.41261226, 254016.82797825,  
190562.29282428, 249602.25952024, 167030.08804751, 124482.8801115 ,  
118454.50580838, 84569.22498851, 202394.95092472, 140307.61486118,  
276225.3194815 , 118234.54796343, 357598.7763246 , 135499.64395121,  
202702.41275566, 263115.50056007, 234448.90789477, 106799.65634261,  
139415.61943571, 329864.78441785, 111438.45747856, 160803.1510078 ,

```
274411.22511389, 101754.55378936, 120161.27982638, 231831.40863402,  
143067.64132067, 155642.16115172])
```

In [ ]: