Import Libraries

In [ ]:
```
import random
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import *
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import RepeatedKFold
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
```

Read the database

In [ ]:
```
df=pd.read_csv('C:/Users/Asus/Desktop/HOME/train.csv')
df.head()
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub

5 rows × 81 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     588 non-null    object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
```

```
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

Check Null values for Numerical features and replacing if less then 70% or removing column if more then 70%

In [ ]:
```
for col in df.columns:
    if df[col].dtype != 'object':
        num_null=df[col].isnull().sum()
        percent_num_null=round((num_null/1460)*100,2)
        if percent_num_null > 0.01:
            print(col,":",percent_num_null," %")
```

```
LotFrontage : 17.74  %
MasVnrArea : 0.55  %
GarageYrBlt : 5.55  %
```

In [ ]:
```
for col in df.columns:
    if df[col].dtype != 'object':
        num_fill=df[col].fillna(df[col].mean(),inplace=True)
```

verifying null values as 0% of numerical feature after applying 'Fillna' to the features

In [ ]:
```
for col in df.columns:
    if df[col].dtype != 'object':
        num_null=df[col].isnull().sum()
        percent_num_null=round((num_null/1460)*100,2)
        if percent_num_null == 0.00:
            print(col,":",percent_num_null," %")
```

```
Id : 0.0  %
MSSubClass : 0.0  %
LotFrontage : 0.0  %
LotArea : 0.0  %
OverallQual : 0.0  %
OverallCond : 0.0  %
YearBuilt : 0.0  %
YearRemodAdd : 0.0  %
MasVnrArea : 0.0  %
BsmtFinSF1 : 0.0  %
BsmtFinSF2 : 0.0  %
BsmtUnfSF : 0.0  %
TotalBsmtSF : 0.0  %
1stFlrSF : 0.0  %
2ndFlrSF : 0.0  %
LowQualFinSF : 0.0  %
GrLivArea : 0.0  %
BsmtFullBath : 0.0  %
BsmtHalfBath : 0.0  %
FullBath : 0.0  %
HalfBath : 0.0  %
BedroomAbvGr : 0.0  %
KitchenAbvGr : 0.0  %
TotRmsAbvGrd : 0.0  %
Fireplaces : 0.0  %
GarageYrBlt : 0.0  %
GarageCars : 0.0  %
GarageArea : 0.0  %
WoodDeckSF : 0.0  %
OpenPorchSF : 0.0  %
EnclosedPorch : 0.0  %
3SsnPorch : 0.0  %
ScreenPorch : 0.0  %
PoolArea : 0.0  %
MiscVal : 0.0  %
MoSold : 0.0  %
YrSold : 0.0  %
SalePrice : 0.0  %
```

Check Null values for Categorical features and replacing if less then 70% or removing column if more then 70%

In [ ]:
```
for col in df.columns:
    if df[col].dtype == 'object' :
        cat_null=df[col].isnull().sum()
        percent_cat_null=round((cat_null/1460)*100,2)
        if percent_cat_null >= 70:
            print(col,":",percent_cat_null," %")
```

```
Alley : 93.77  %
PoolQC : 99.52  %
Fence : 80.75  %
MiscFeature : 96.3  %
```

Dropping Null Values above 70% from categorical features

In [ ]:
```
df.drop('Alley',axis=1,inplace=True)
df.drop('PoolQC',axis=1,inplace=True)
df.drop('Fence',axis=1,inplace=True)
df.drop('MiscFeature',axis=1,inplace=True)
```

Filling null values with 'bfill' for categorical features

In [ ]:
```
for col in df.columns:
    if df[col].dtype == 'object':
        df[col].bfill(inplace=True)
```

Verifying null values for categorical features

In [ ]:
```
for col in df.columns:
    if df[col].dtype == 'object':
        cat_null=df[col].isnull().sum()
        percent_cat_null=round((cat_null/1460)*100,2)
        print(col,":",percent_cat_null," %")
```

```
MSZoning : 0.0  %
Street : 0.0  %
LotShape : 0.0  %
LandContour : 0.0  %
Utilities : 0.0  %
LotConfig : 0.0  %
LandSlope : 0.0  %
Neighborhood : 0.0  %
Condition1 : 0.0  %
Condition2 : 0.0  %
BldgType : 0.0  %
HouseStyle : 0.0  %
RoofStyle : 0.0  %
RoofMatl : 0.0  %
Exterior1st : 0.0  %
Exterior2nd : 0.0  %
MasVnrType : 0.21  %
ExterQual : 0.0  %
ExterCond : 0.0  %
Foundation : 0.0  %
BsmtQual : 0.0  %
BsmtCond : 0.0  %
BsmtExposure : 0.0  %
BsmtFinType1 : 0.0  %
BsmtFinType2 : 0.0  %
Heating : 0.0  %
HeatingQC : 0.0  %
CentralAir : 0.0  %
Electrical : 0.0  %
KitchenQual : 0.0  %
Functional : 0.0  %
FireplaceQu : 0.14  %
GarageType : 0.0  %
GarageFinish : 0.0  %
GarageQual : 0.0  %
GarageCond : 0.0  %
PavedDrive : 0.0  %
SaleType : 0.0  %
SaleCondition : 0.0  %
```

In [ ]:
```
df.drop('MasVnrType',axis=1,inplace=True)
df.drop('FireplaceQu',axis=1,inplace=True)
```

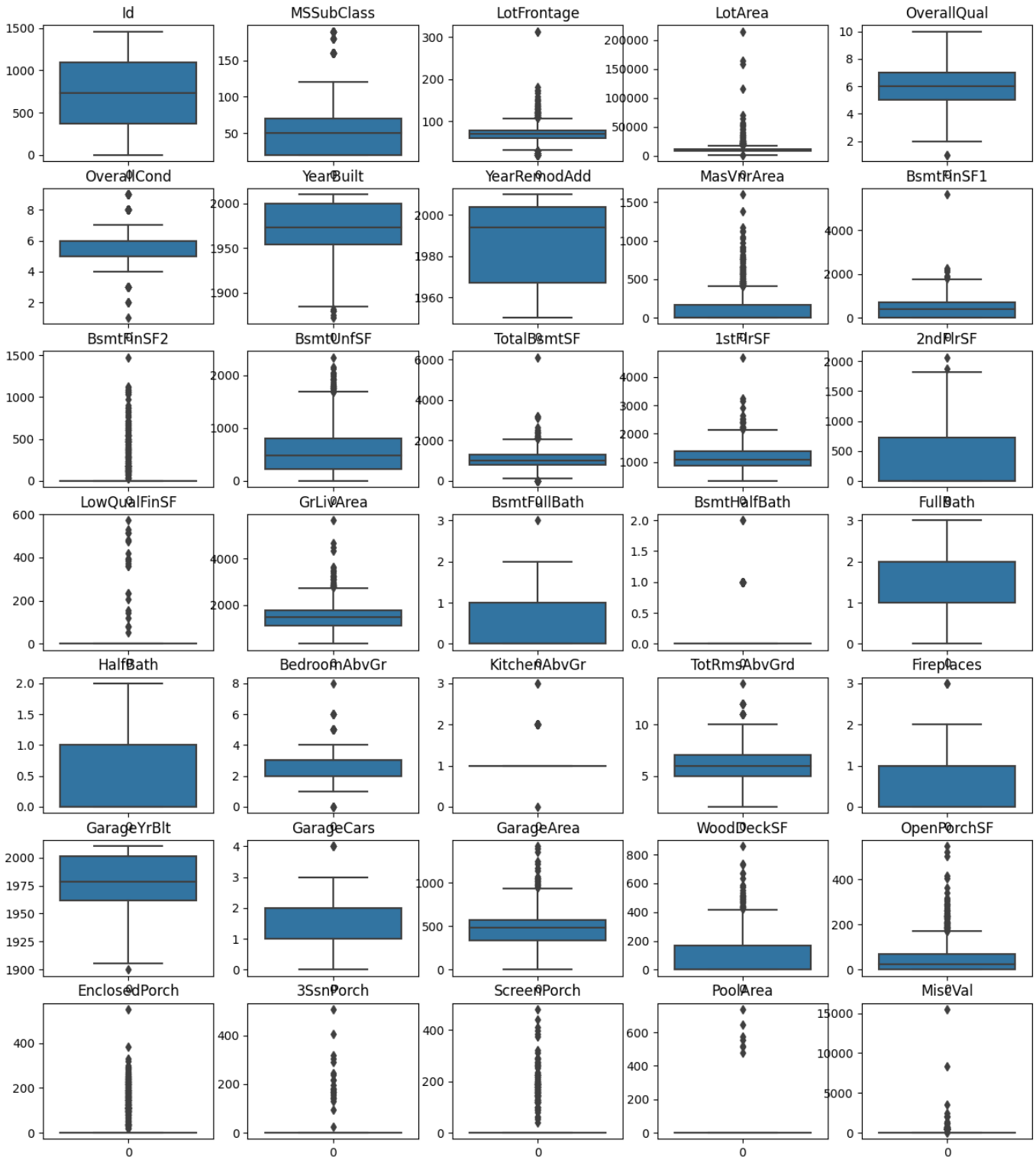
Checking Outliers for the data by plotting BoxPlot of every Numeric feature

In [ ]:
```
numerical_columns=[i for i in df.columns if df[i].dtype != 'object']
```

Checking outliers with Boxplot

In [ ]:
```
fig, ax = plt.subplots(7,5,figsize=(15, 17))
count = 0
```

```
for i in range(7):
    for j in range(5):
        sns.boxplot(df[numerical_columns[count]],ax=ax[i][j])
        ax[i][j].set_title(numerical_columns[count])
        count = count + 1
```



Function for replacing outliers

In [ ]:
```
def check_out(numerical_columns):
    Q1=df[numerical_columns].quantile(0.25)
    Q3=df[numerical_columns].quantile(0.75)
    IQR=Q3-Q1
    up_whisk=Q3+(1.5*IQR)
    low_whisk=Q1-(1.5*IQR)
    return up_whisk, low_whisk
```

In [ ]:
```
for i in numerical_columns:
    a,b=check_out(i)
    df.loc[df[i]>a,i]=a
```

```
    df.loc[df[i]<b,i]=b
```

Code to remove the columns whose values contains mostly 0

In [ ]:
```
for i in numerical_columns:
    a = dict(df[i].value_counts())
    b = list(a.values())
    if(b[0]==1460):
        df.drop(columns=i, inplace=True)
```

In [ ]:
```
df.columns
```

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

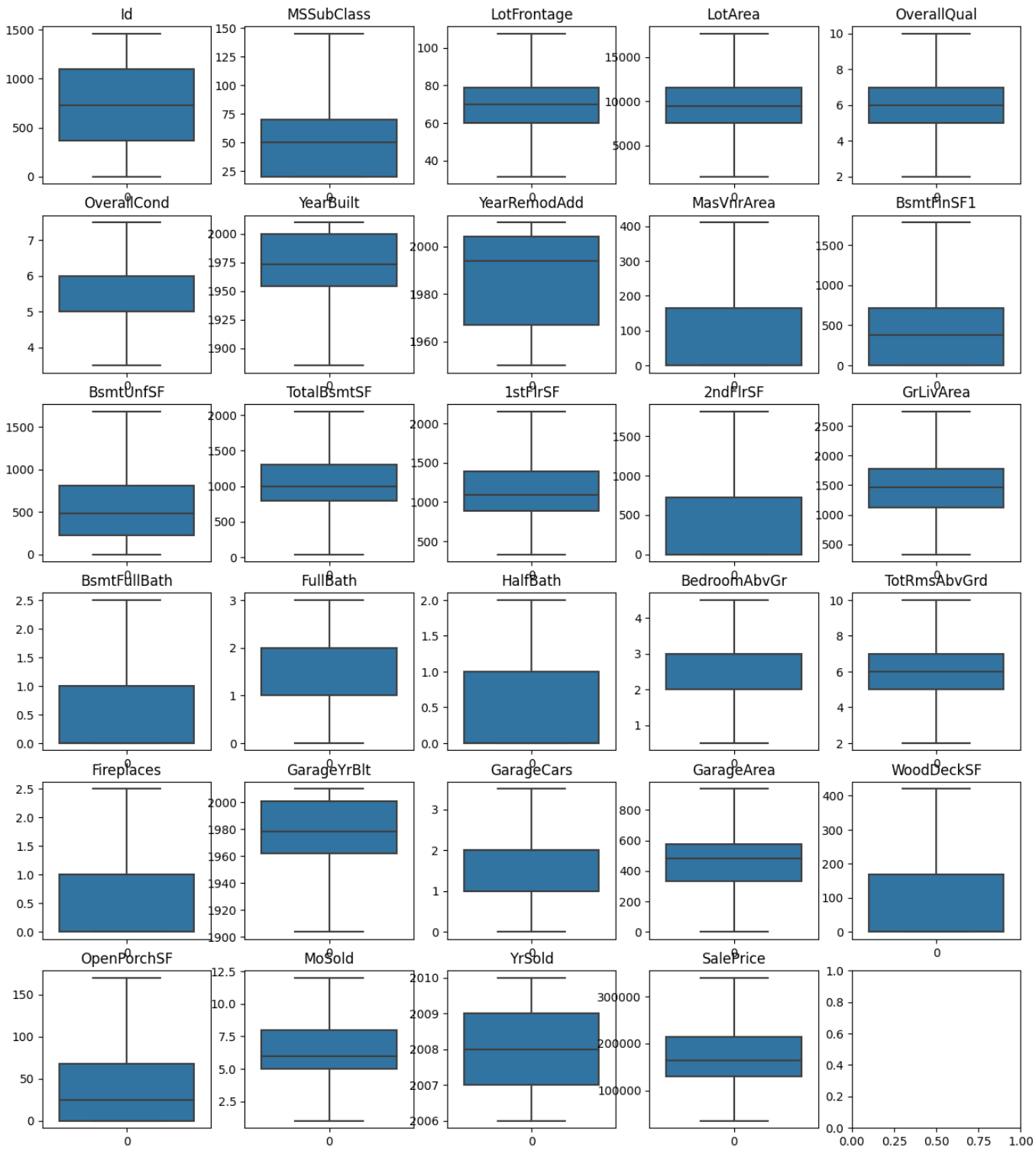
Seperating numeric columns from the new columns list after removing the columns with 0 value

In [ ]:
```
numeric_columns = [i for i in df.columns if df[i].dtype != 'object']
len(numeric_columns)
```

Out[ ]:
```
29
```

Again Checking final boxplot for outliers

In [ ]:
```
fig, ax = plt.subplots(6,5,figsize=(15, 17))
count = 0
for i in range(6):
    for j in range(5):
        sns.boxplot(df[numeric_columns[count]],ax=ax[i][j])
        ax[i][j].set_title(numeric_columns[count])
        count = count + 1
        if count == 29:
            break
```



Data Visualization

Removing/Dropping unnecessory columns to avoid false model fitting and visualization

In [ ]:
```
df.drop('MSSubClass',axis=1,inplace=True)
df.drop('BsmtExposure',axis=1,inplace=True)
df.drop('BsmtFinType1',axis=1,inplace=True)
df.drop('BsmtFinSF1',axis=1,inplace=True)
df.drop('BsmtFinType2',axis=1,inplace=True)
df.drop('BsmtUnfSF',axis=1,inplace=True)
df.drop('HeatingQC',axis=1,inplace=True)
df.drop('BsmtFullBath',axis=1,inplace=True)
df.drop('GarageFinish',axis=1,inplace=True)
df.drop('WoodDeckSF',axis=1,inplace=True)
```

In [ ]:
```
df.drop('LandContour',axis=1,inplace=True)
df.drop('Utilities',axis=1,inplace=True)
df.drop('LotConfig',axis=1,inplace=True)
df.drop('LandSlope',axis=1,inplace=True)
```

In [ ]:
```
df.drop('Exterior1st',axis=1,inplace=True)
df.drop('Exterior2nd',axis=1,inplace=True)
df.drop('ExterQual',axis=1,inplace=True)
```

In [ ]:
```
df.drop('YearRemodAdd',axis=1,inplace=True)
df.drop('Functional',axis=1,inplace=True)
df.drop('GarageCars',axis=1,inplace=True)
df.drop('GarageQual',axis=1,inplace=True)
df.drop('PavedDrive',axis=1,inplace=True)
```

In [ ]:
```
df.drop('LotFrontage',axis=1,inplace=True)
```

In [ ]:
```
df.drop('OverallQual',axis=1,inplace=True)
```

In [ ]:
```
df.drop('RoofStyle',axis=1,inplace=True)
```

In [ ]:
```
df.drop('RoofMatl',axis=1,inplace=True)
df.drop('BsmtQual',axis=1,inplace=True)
```

Checking out reduced columns size

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Id', 'MSZoning', 'LotArea', 'Street', 'LotShape', 'Neighborhood',
       'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallCond',
       'YearBuilt', 'MasVnrArea', 'ExterCond', 'Foundation', 'BsmtCond',
       'TotalBsmtSF', 'Heating', 'CentralAir', 'Electrical', '1stFlrSF',
       '2ndFlrSF', 'GrLivArea', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'KitchenQual', 'TotRmsAbvGrd', 'Fireplaces', 'GarageType',
       'GarageYrBlt', 'GarageArea', 'GarageCond', 'OpenPorchSF', 'MoSold',
       'YrSold', 'SaleType', 'SaleCondition', 'SalePrice'],
      dtype='object')
```

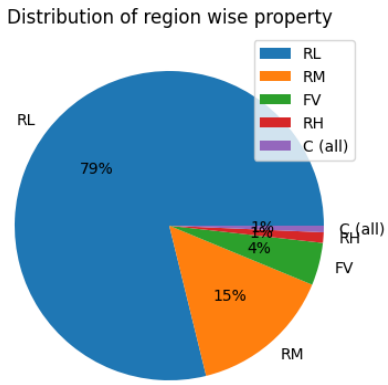
In [ ]:
```
df.shape
```

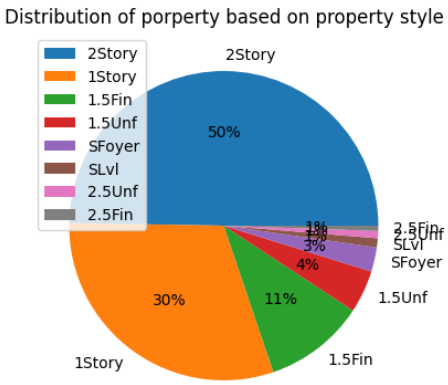
Out[ ]:
```
(1460, 39)
```

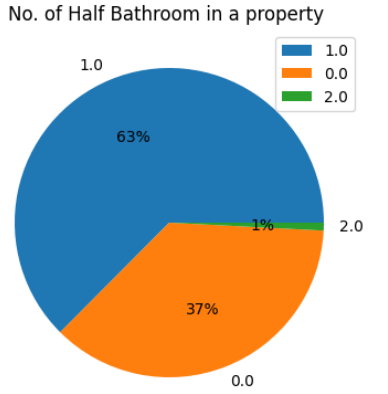
Pie Charts

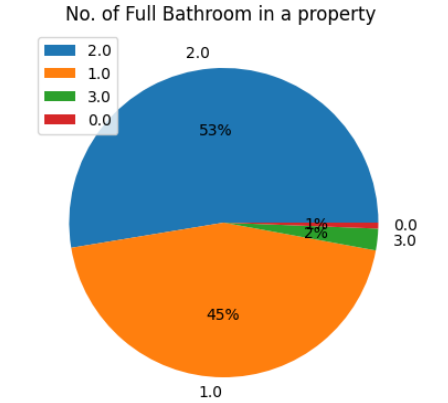
In [ ]:
```
fig,ax=plt.subplots(2,2,figsize=(20,10))
label=['RL','RM','FV','RH','C (all)']
ax[0][0].pie(df['MSZoning'].value_counts(),labels=label,autopct='%.1f%%')
ax[0][0].set_title('Distribution of region wise property')
ax[0][0].legend(loc='upper right')

label=df['HouseStyle'].unique()
ax[0][1].pie(df['HouseStyle'].value_counts(),labels=label,autopct='%.1f%%')
ax[0][1].set_title('Distribution of porperty based on property style')
ax[0][1].legend(loc='upper left')

label=df['HalfBath'].unique()
ax[1][0].pie(df['HalfBath'].value_counts(),labels=label,autopct='%.1f%%')
ax[1][0].set_title('No. of Half Bathroom in a property ')
ax[1][0].legend(loc='upper right')

label=df['FullBath'].unique()
ax[1][1].pie(df['FullBath'].value_counts(),labels=label,autopct='%.1f%%')
```

```
ax[1][1].set_title('No. of Full Bathroom in a property ')
ax[1][1].legend(loc='upper left')

plt.show()
```





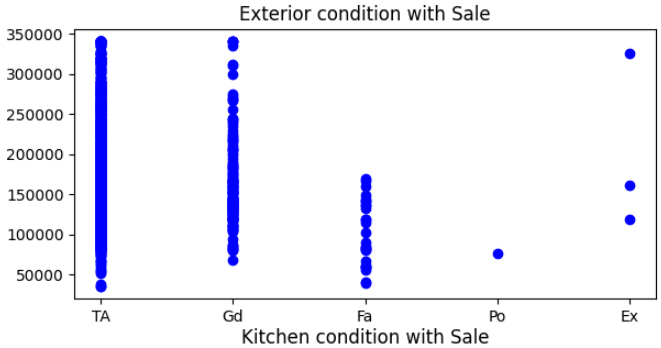


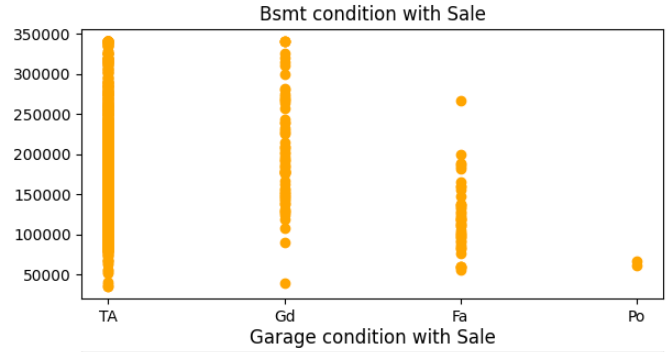


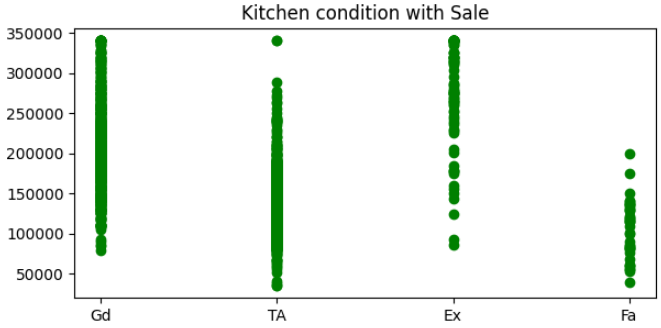
Scatter Plots

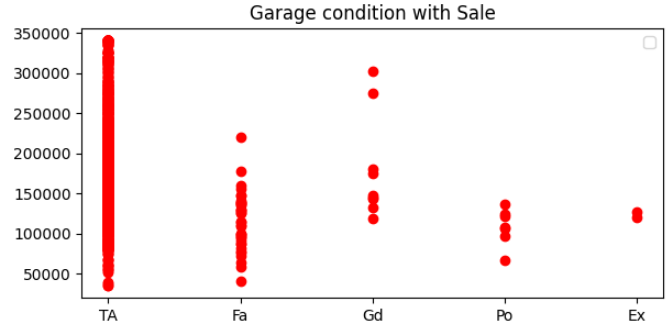
In [ ]:
```
fig,ax=plt.subplots(2,2,figsize=(15,7))
ax[0][0].scatter(y=df['SalePrice'],x=df['ExterCond'],color='Blue')
ax[0][0].set_title('Exterior condition with Sale')
ax[0][1].scatter(y=df['SalePrice'],x=df['BsmtCond'],color='Orange')
ax[0][1].set_title('Bsmt condition with Sale')
ax[1][0].scatter(y=df['SalePrice'],x=df['KitchenQual'],color='green')
ax[1][0].set_title('Kitchen condition with Sale')
ax[1][1].scatter(y=df['SalePrice'],x=df['GarageCond'],color='red')
ax[1][1].set_title('Garage condition with Sale')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



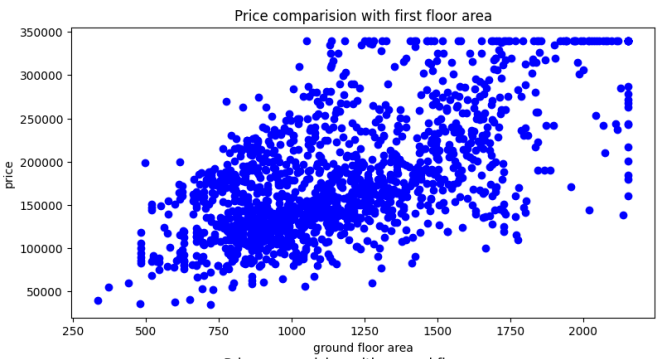


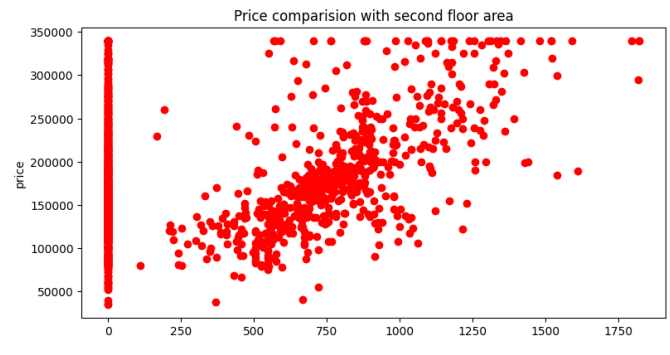


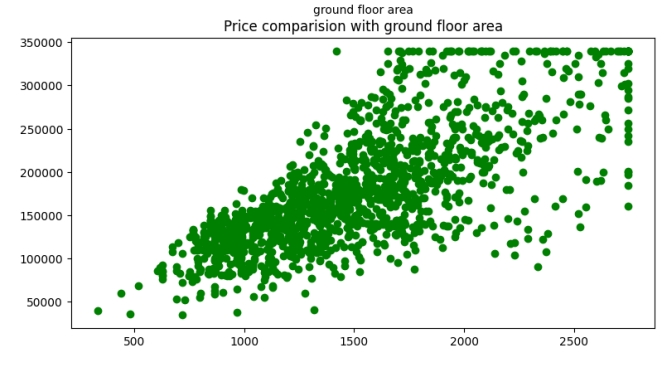


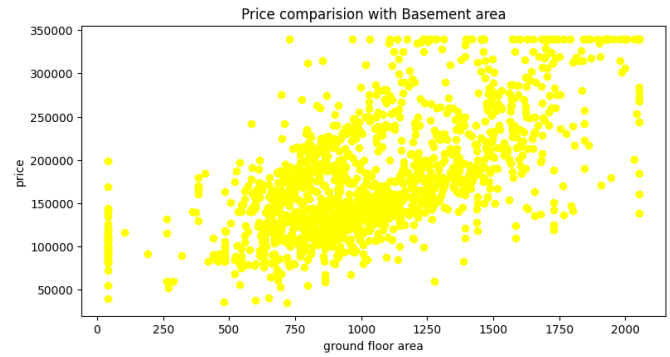
In [ ]:
```
fig,ax=plt.subplots(2,2,figsize=(20,10))

ax[0][0].scatter(x=df['1stFlrSF'],y=df['SalePrice'],color='Blue')
ax[0][0].set_title('Price comparision with first floor area')
ax[0][0].set_xlabel('first floor area')
ax[0][0].set_ylabel('price')

ax[0][1].scatter(x=df['2ndFlrSF'],y=df['SalePrice'],color='red')
ax[0][1].set_title('Price comparision with second floor area')
ax[0][0].set_xlabel('second floor area')
ax[0][1].set_ylabel('price')

ax[1][0].scatter(x=df['GrLivArea'],y=df['SalePrice'],color='green')
ax[1][0].set_title('Price comparision with ground floor area')
ax[0][0].set_xlabel('ground floor area')
ax[0][1].set_ylabel('price')

ax[1][1].scatter(x=df['TotalBsmtSF'],y=df['SalePrice'],color='Yellow')
ax[1][1].set_title('Price comparision with Basement area')
ax[1][1].set_xlabel('ground floor area')
ax[1][1].set_ylabel('price')
```

Out[ ]: Text(0, 0.5, 'price')





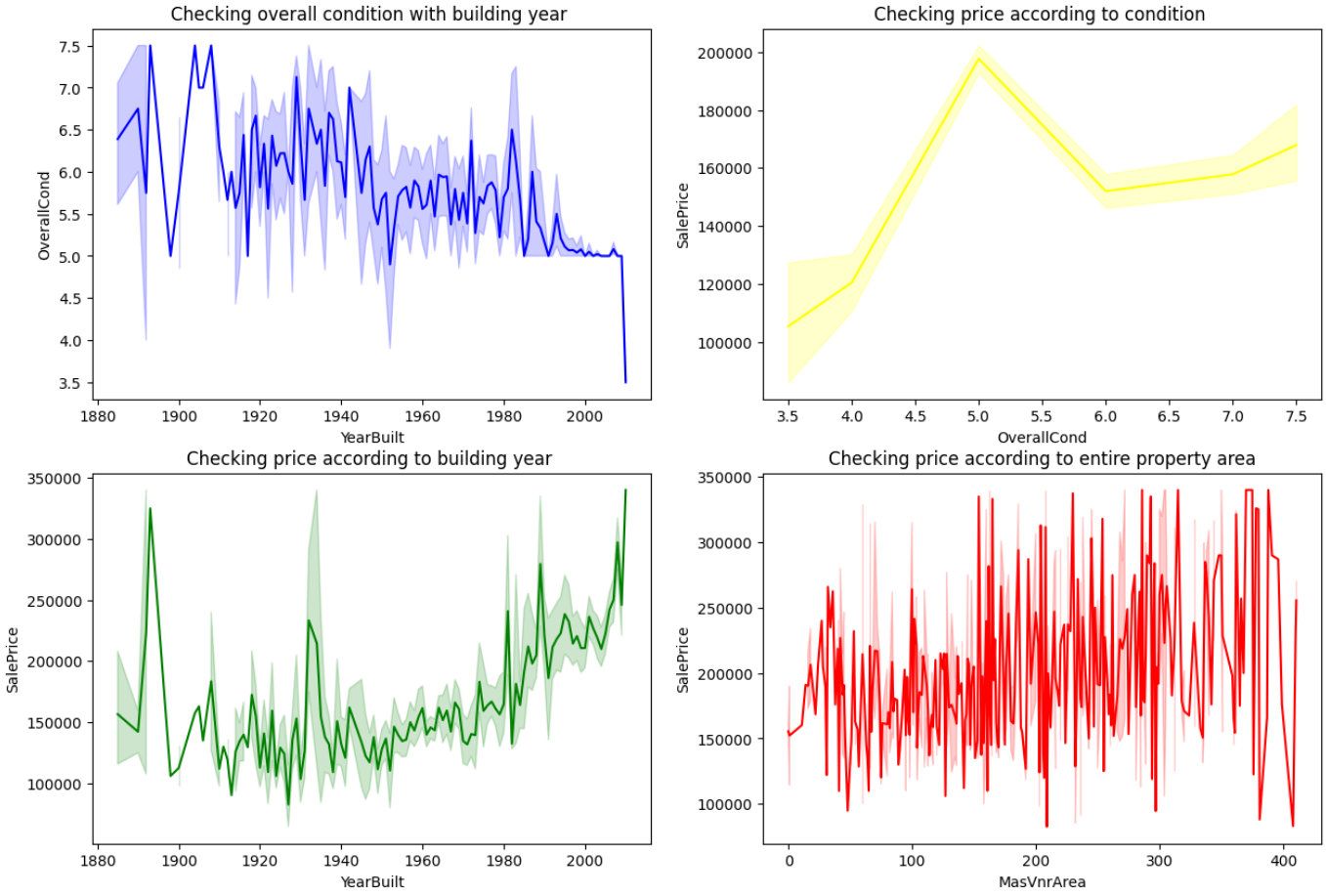




Line Graphs

In [ ]:
```
fig,ax=plt.subplots(2,2,figsize=(15,10))
sns.lineplot(x=df['YearBuilt'],y=df['OverallCond'],color='Blue',ax=ax[0][0])
ax[0][0].set_title('Checking overall condition with building year ')
sns.lineplot(x=df['OverallCond'],y=df['SalePrice'],color='yellow',ax=ax[0][1])
ax[0][1].set_title('Checking price according to condition ')
sns.lineplot(x=df['YearBuilt'],y=df['SalePrice'],color='Green',ax=ax[1][0])
ax[1][0].set_title('Checking price according to building year ')
sns.lineplot(x=df['MasVnrArea'],y=df['SalePrice'],color='red',ax=ax[1][1])
ax[1][1].set_title('Checking price according to entire property area')
```

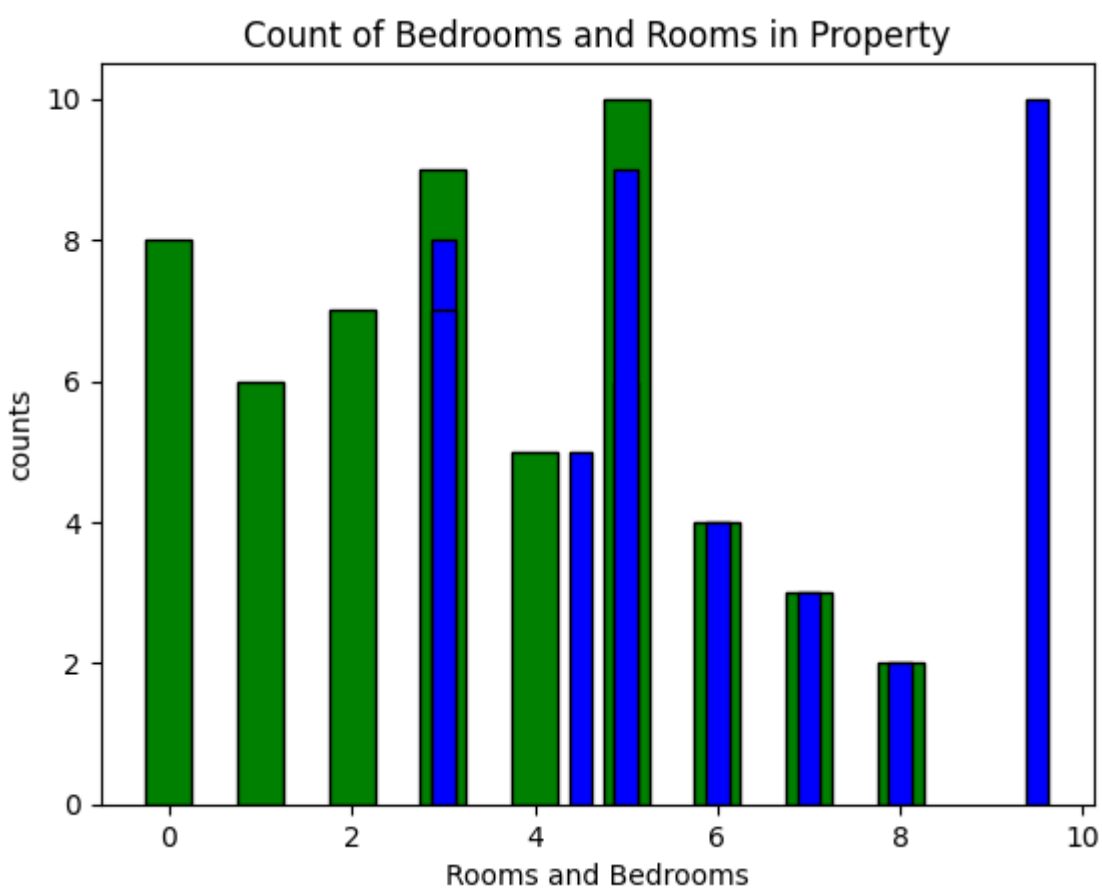
Out[ ]:
```
Text(0.5, 1.0, 'Checking price according to entire property area')
```



Bar Graphs

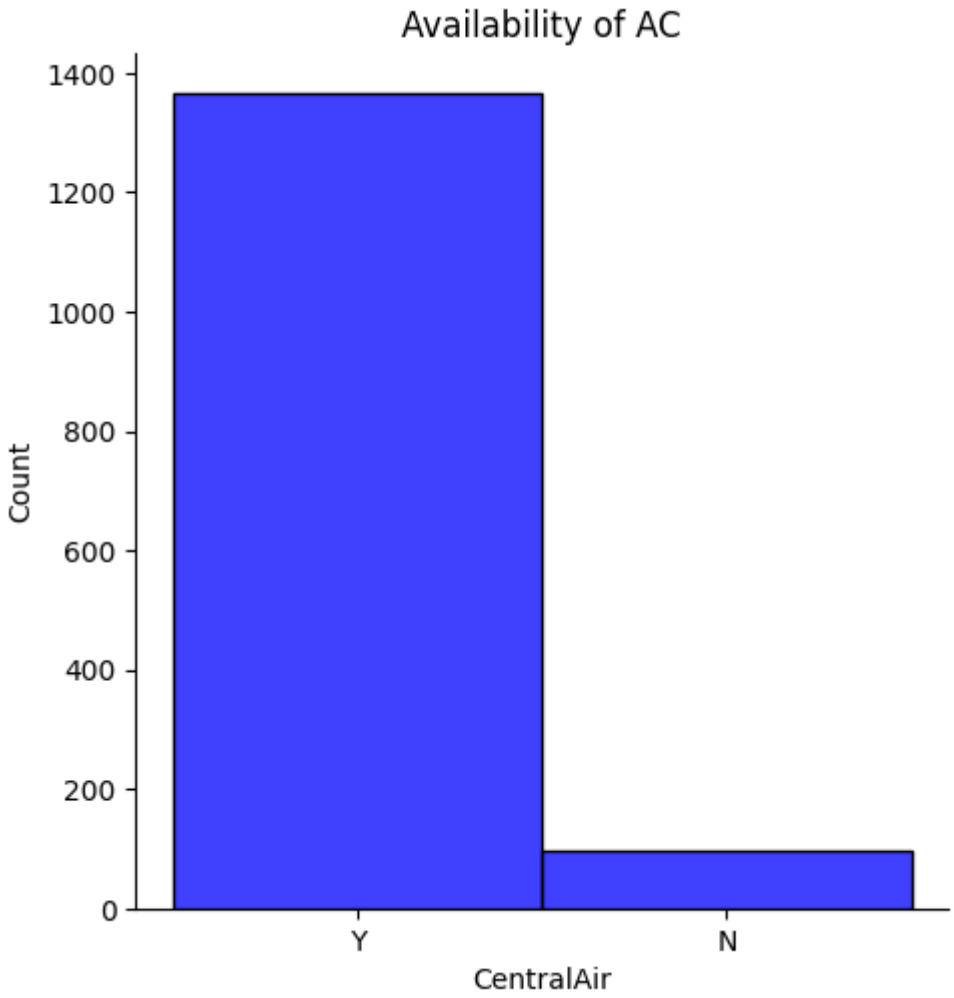
In [ ]:
```
rooms=df['TotRmsAbvGrd'].unique()
bedrooms=df['BedroomAbvGr'].unique()
bedrooms = np.pad(bedrooms, (0, len(rooms) - len(bedrooms)), 'constant')
n=9
r=np.arange(n)
plt.bar(r, rooms, color = 'g',width = 0.50, edgecolor = 'black')
plt.bar(r + bedrooms, rooms, color = 'b',width = 0.25, edgecolor = 'black')
plt.xlabel('Rooms and Bedrooms')
plt.ylabel('counts')
plt.title('Count of Bedrooms and Rooms in Property')
```

Out[ ]:
```
Text(0.5, 1.0, 'Count of Bedrooms and Rooms in Property')
```



In [ ]:
```
sns.displot(df['CentralAir'],color='b')
plt.title('Availability of AC')
```

Out[ ]:
```
Text(0.5, 1.0, 'Availability of AC')
```



Normalization

Normlization of Numerical columns and categorical columns in dataset

In [ ]:
```
for col in df.columns:
    if (df[col].dtype != 'object'):                          # For numerical dataset
        number=df[col]
        reshaped_data=number.values.reshape(-1,1)
        scaler=MinMaxScaler()
        scaled_data=scaler.fit_transform(reshaped_data)
        scaled_df=pd.DataFrame(scaled_data)
        df[col]=scaled_df
    else:
        category=df[col]                                     # for categorical dataset
        reshaped_data=category.values.reshape(-1,1)
        scaler=LabelEncoder()
        scaled_data=scaler.fit_transform(reshaped_data)
        scaled_df=pd.DataFrame(scaled_data)
        df[col]=scaled_df
```

```
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
```

```
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing
\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	MSZoning	LotArea	Street	LotShape	Neighborhood	Condition1	Condition2	BldgType
0	0.000000	3	0.430367	1	3	5	2	2	0
1	0.000685	3	0.501390	1	3	24	1	2	0
2	0.001371	3	0.603292	1	0	5	2	2	0
3	0.002056	3	0.498302	1	0	6	2	2	0
4	0.002742	3	0.789186	1	0	15	2	2	0

5 rows × 39 columns

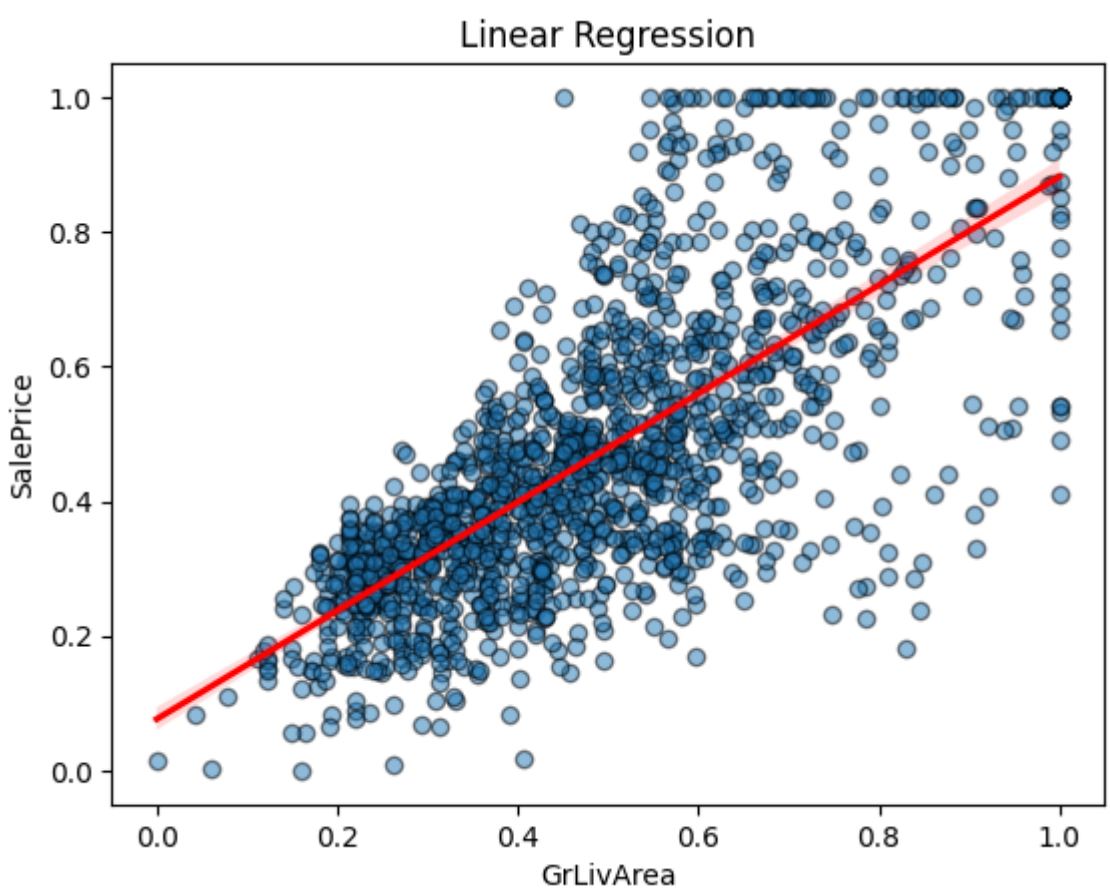
Analyzing Correlation in Dataset through HeatMap

Linear Regression Model

In [ ]:
```
x=df['GrLivArea']
y=df['SalePrice']
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.6,random_state=10)
x_train=x_train.values.reshape(-1,1)
y_train=y_train.values.reshape(-1,1)
x_test=x_test.values.reshape(-1,1)
y_test=y_test.values.reshape(-1,1)
model=LinearRegression()
linear=model.fit(x_train,y_train)
prediction=linear.predict(x_test)
```

In [ ]:
```
sns.regplot(x='GrLivArea',y='SalePrice',data=df,line_kws={'color':'RED'}, scatter_kws={'alpha
plt.title('Linear Regression')
```

Out[ ]:
```
Text(0.5, 1.0, 'Linear Regression')
```



In [ ]:
```
linear_accuracy=linear.score(x_test,y_test)
print(linear_accuracy)
```

```
0.5574204006567671
```

Multiple Linear Regression Model

In [ ]:
```
x=df[['LotArea','YearBuilt','TotalBsmtSF','1stFlrSF','2ndFlrSF','GrLivArea','KitchenQual','Ga
y=df['SalePrice']
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70,random_state=1)
model=LinearRegression()
Mlinear=model.fit(x_train,y_train)
prediction=Mlinear.predict(x_test)
```

In [ ]:
```
multiple_accuracy=Mlinear.score(x_test,y_test)
print(multiple_accuracy)
```

```
0.8398349472057939
```

Polynomial Regression Model

In [ ]:
```
x=df['GrLivArea'].values.reshape(-1,1)
y=df['SalePrice']
poly = PolynomialFeatures(degree=5,include_bias=False)
x_poly = poly.fit_transform(x)
plinear=poly.fit(x_poly, y)
model= LinearRegression()
plinear=model.fit(x_poly, y)
prediction=model.predict(x_poly)
```

In [ ]:
```
polynomial_accuracy=plinear.score(x_poly,y)
print(polynomial_accuracy)
```

```
0.5353363380347977
```

Logistic Regression Model

In [ ]:
```
x=df['Street']
y=df['SaleType']
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70,random_state=1)
x_train=x_train.values.reshape(-1,1)
x_test=x_test.values.reshape(-1,1)
y_train=y_train.values.reshape(-1,1)
y_test=y_test.values.reshape(-1,1)
model=LogisticRegression()
loglinear=model.fit(x_train,y_train)
prediction=model.predict(x_test)
```

```
c:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validati
on.py:1184: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:
```
logistic_accuracy=loglinear.score(x_test,y_test)
```

Ridge Regression Model

In [ ]:
```
x=df['GrLivArea']
y=df['SalePrice']
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=1)
x_train=x_train.values.reshape(-1,1)
x_test=x_test.values.reshape(-1,1)
y_train=y_train.values.reshape(-1,1)
y_test=y_test.values.reshape(-1,1)
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
model=RidgeCV(alphas=(1,2,0.01),cv=cv)
ridgeReg=model.fit(x_test,y_test)
prediction=model.predict(x_test)
```

In [ ]:
```
ridge_accuracy=ridgeReg.score(x_test,y_test)
```

The Final R2 Score Comparision

In [ ]:
```
print('The R2-Score value obtained by Linear Regression Model: ',linear_accuracy)
print('The R2-Score value obtained by Multi-linear Regression Model: ',multiple_accuracy)
print('The R2-Score value obtained by polynomial Regression Model: ',polynomial_accuracy)
print('The R2-Score value obtained by Logistic Regression Model: ',logistic_accuracy)
print('The R2-Score value obtained by Ridge Regression Model: ',ridge_accuracy)
```

```
The R2-Score value obtained by Linear Regression Model:  0.5574204006567671
The R2-Score value obtained by Multi-linear Regression Model:  0.8398349472057939
The R2-Score value obtained by polynomial Regression Model:  0.5353363380347977
The R2-Score value obtained by Logistic Regression Model:  0.8656036446469249
The R2-Score value obtained by Ridge Regression Model:  0.5322770689496097
```