```
import pandas as pd import numpy as
np import matplotlib.pyplot as plt
from matplotlib.pyplot import
figure import scipy.stats as scpsts
%matplotlib inline import seaborn
import warnings
warnings.filterwarnings("ignore"
) import seaborn as sns
```

PERFORMING THE FOLLOWING STEPS: 1.Reaading the data 2.Cleanig the data 3.Analysis 4.Skewing 5.Standard Deviation 6.

1. Reaading the data
2. Cleanig the data
3. Analysis
4. Skewing
5. Standard Deviation
. Visualizing
7. Nominal variable boxplot analysis
. Correlation

About Dataset:

The data collection comprises information from the Ames Assessor's O ce that was utilised in calculating assessed values for individual residential properties sold in Ames, Iowa between 2006 and 2010.

There are 82 columns in the dataset, including 23 nominal, 23 ordinal, 14 discrete, and 20 continuous variables.

```
#converting the csv file to a data frame using pandas data frame for further process
df = pd.read_csv("./AmesHousing.csv",na_values=['nan'])
```

```
#Finding the first 5 rows od the data frame created using head()
df.head()
```

Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	...	Po Ar
1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	...	
2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	...	
3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	...	
4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	...	
5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	...	

s × 82 columns

Let us investigate the dataset's nature by determining the number of records and features.

```
#the shape functions prints out the total number of rows and columns of the data set.
df.shape
    (2930, 82)
```

now we can see that we have 2930 rows and 82 columns for the analysis

Let's obtain some more information about the dataset, such as the number of columns, column labels, column data types, memory use, and so on, using info ()

```
df.info()
```

```
 26  Mas Vnr Type     2907 non-null   object
 27  Mas Vnr Area     2907 non-null   float64
 28  Exter Qual       2930 non-null   object
 29  Exter Cond       2930 non-null   object
 30  Foundation       2930 non-null   object
 31  Bsmt Qual        2850 non-null   object
 32  Bsmt Cond        2850 non-null   object
 33  Bsmt Exposure    2847 non-null   object
 34  BsmtFin Type 1   2850 non-null   object
 35  BsmtFin SF 1     2929 non-null   float64
 36  BsmtFin Type 2   2849 non-null   object
 37  BsmtFin SF 2     2929 non-null   float64
 38  Bsmt Unf SF      2929 non-null   float64
 39  Total Bsmt SF    2929 non-null   float64
 40  Heating          2930 non-null   object
 41  Heating QC       2930 non-null   object
 42  Central Air      2930 non-null   object
 43  Electrical       2929 non-null   object
 44  1st Flr SF       2930 non-null   int64
 45  2nd Flr SF       2930 non-null   int64
 46  Low Qual Fin SF  2930 non-null   int64
 47  Gr Liv Area      2930 non-null   int64
 48  Bsmt Full Bath   2928 non-null   float64
 49  Bsmt Half Bath   2928 non-null   float64
 50  Full Bath        2930 non-null   int64
 51  Half Bath        2930 non-null   int64
 52  Bedroom AbvGr    2930 non-null   int64
 53  Kitchen AbvGr    2930 non-null   int64
 54  Kitchen Qual     2930 non-null   object
 55  TotRms AbvGrd    2930 non-null   int64
 56  Functional       2930 non-null   object
 57  Fireplaces       2930 non-null   int64
 58  Fireplace Qu     1508 non-null   object
 59  Garage Type      2773 non-null   object
 60  Garage Yr Blt    2771 non-null   float64
 61  Garage Finish    2771 non-null   object
 62  Garage Cars      2929 non-null   float64
 63  Garage Area      2929 non-null   float64
 64  Garage Qual      2771 non-null   object
 65  Garage Cond      2771 non-null   object
 66  Paved Drive      2930 non-null   object
 67  Wood Deck SF     2930 non-null   int64
 68  Open Porch SF    2930 non-null   int64
 69  Enclosed Porch   2930 non-null   int64
 70  3Ssn Porch       2930 non-null   int64
 71  Screen Porch     2930 non-null   int64
 72  Pool Area        2930 non-null   int64
 73  Pool QC          13 non-null     object
 74  Fence            572 non-null    object
 75  Misc Feature     106 non-null    object
 76  Misc Val         2930 non-null   int64
 77  Mo Sold          2930 non-null   int64
 78  Yr Sold          2930 non-null   int64
 79  Sale Type        2930 non-null   object
 80  Sale Condition   2930 non-null   object  81  SalePrice        2930 non-null   int64  dtypes: float64(11),
     int64(28), object(43) memory usage: 1.8+ MB
```

CLEANING THE DATA

```
# first let's check the null values
NaN=df.isnull().sum().to_frame('NaN')
NaN[NaN['NaN']>0]
```

	NaN
Lot Frontage	490
Alley	2732
Mas Vnr Type	23
Mas Vnr Area	23
Bsmt Qual	80
Bsmt Cond	80

Bsmt Exposure	83
BsmtFin Type 1	80
BsmtFin SF 1	1
BsmtFin Type 2	81
BsmtFin SF 2	1
Bsmt Unf SF	1
Total Bsmt SF	1
Electrical	1
Bsmt Full Bath	2
Bsmt Half Bath	2
Fireplace Qu	1422
Garage Type	157
Garage Yr Blt	159
Garage Finish	159
Garage Cars	1
Garage Area	1
Garage Qual	159
Garage Cond	159
Pool QC	2917
Fence	2358
Misc Feature	2824

Checking the percentage of Null Values

```
#Check % of null values null_cols=['Lot Frontage', 'Alley', 'Mas Vnr Type', 'Mas
Vnr Area', 'Bsmt Qual',
       'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin SF 1',
       'BsmtFin Type 2', 'BsmtFin SF 2', 'Bsmt Unf SF', 'Total Bsmt SF',
'Electrical', 'Bsmt Full Bath', 'Bsmt Half Bath', 'Fireplace Qu',
       'Garage Type', 'Garage Yr Blt', 'Garage Finish', 'Garage Cars',
       'Garage Area', 'Garage Qual', 'Garage Cond', 'Pool QC', 'Fence',
       'Misc Feature']
for colName in
 null_cols:
    print('The % of null values:',colName,':',round((df[colName].isnull().sum()/df.shape[0])*100,2))
if(round((df[colName].isnull().sum()/df.shape[0])*100,2)>30):
        df.drop([colName],axis=1,inplace=True)
        print(colName,' dropped because percentage of null values was greater than 30%')
```

```
The % of null values: Lot Frontage : 16.72
The % of null values: Alley : 93.24
Alley  dropped because percentage of null values was greater than 30%
The % of null values: Mas Vnr Type : 0.78
The % of null values: Mas Vnr Area : 0.78
The % of null values: Bsmt Qual : 2.73
The % of null values: Bsmt Cond : 2.73
The % of null values: Bsmt Exposure : 2.83
The % of null values: BsmtFin Type 1 : 2.73
The % of null values: BsmtFin SF 1 : 0.03
The % of null values: BsmtFin Type 2 : 2.76
The % of null values: BsmtFin SF 2 : 0.03
The % of null values: Bsmt Unf SF : 0.03
The % of null values: Total Bsmt SF : 0.03
The % of null values: Electrical : 0.03
The % of null values: Bsmt Full Bath : 0.07
The % of null values: Bsmt Half Bath : 0.07
The % of null values: Fireplace Qu : 48.53
Fireplace Qu  dropped because percentage of null values was greater than 30%
The % of null values: Garage Type : 5.36
The % of null values: Garage Yr Blt : 5.43
The % of null values: Garage Finish : 5.43
The % of null values: Garage Cars : 0.03
```

```
The % of null values: Garage Area : 0.03
The % of null values: Garage Qual : 5.43
The % of null values: Garage Cond : 5.43
The % of null values: Pool QC : 99.56
Pool QC  dropped because percentage of null values was greater than 30%
The % of null values: Fence : 80.48
Fence  dropped because percentage of null values was greater than 30%
The % of null values: Misc Feature : 96.38
Misc Feature  dropped because percentage of null values was greater than 30%
```

We automatically removed columns with missing values greater than 30%. We discovered that Bsmt Qual and Bsmt Cond have the same percentage of null values, indicating that these residences do not have basements.

```
 #In such if its a categorical variable then we can introduce a new category for such cases. #If
its a continious then we can set it to zero i.e. no basement area.
```

```
percent= 100*(len(df.loc[:,df.isnull().sum(axis=0)>=1 ].index) / len(df.index))
print(round(percent,2))
```

```
100.0
```

```
df.columns
```

```
Index(['Order', 'PID', 'MS SubClass', 'MS Zoning', 'Lot Frontage', 'Lot Area',
       'Street', 'Lot Shape', 'Land Contour', 'Utilities', 'Lot Config',
'Land Slope', 'Neighborhood', 'Condition 1', 'Condition 2', 'Bldg Type',
       'House Style', 'Overall Qual', 'Overall Cond', 'Year Built',
       'Year Remod/Add', 'Roof Style', 'Roof Matl', 'Exterior 1st',
       'Exterior 2nd', 'Mas Vnr Type', 'Mas Vnr Area', 'Exter Qual',
       'Exter Cond', 'Foundation', 'Bsmt Qual', 'Bsmt Cond', 'Bsmt Exposure',
       'BsmtFin Type 1', 'BsmtFin SF 1', 'BsmtFin Type 2', 'BsmtFin SF 2',
       'Bsmt Unf SF', 'Total Bsmt SF', 'Heating', 'Heating QC', 'Central Air',
       'Electrical', '1st Flr SF', '2nd Flr SF', 'Low Qual Fin SF',
       'Gr Liv Area', 'Bsmt Full Bath', 'Bsmt Half Bath', 'Full Bath',
       'Half Bath', 'Bedroom AbvGr', 'Kitchen AbvGr', 'Kitchen Qual',
       'TotRms AbvGrd', 'Functional', 'Fireplaces', 'Garage Type',
       'Garage Yr Blt', 'Garage Finish', 'Garage Cars', 'Garage Area',
       'Garage Qual', 'Garage Cond', 'Paved Drive', 'Wood Deck SF',
       'Open Porch SF', 'Enclosed Porch', '3Ssn Porch', 'Screen Porch',
       'Pool Area', 'Misc Val', 'Mo Sold', 'Yr Sold', 'Sale Type',
       'Sale Condition', 'SalePrice'],
dtype='object')
```

Now we're going to use a continuous predictor to check the null percentages in the speci c columns.

```
## function to print out percentage of null values for each column in dataset def
NaNperct_cols(df,colName):     print('Null column prectentage of
',colName,':',round((df[colName].isnull().sum()/df.shape[0])*100,2))
```

```
NaNperct_cols(df,'1st Flr SF')
NaNperct_cols(df,'Pool Area')
NaNperct_cols(df,'Garage Area')
NaNperct_cols(df,'BsmtFin SF 2')
```

```
Null column prectentage of  1st Flr SF : 0.0
Null column prectentage of  Pool Area : 0.0
Null column prectentage of  Garage Area : 0.03
Null column prectentage of  BsmtFin SF 2 : 0.03
```

we can see that the columns garage area and Type 2 nished square feet basement has the same percentage of null values.

Double-click (or enter) to edit

Continuous variable scatter plots versus SalePrice

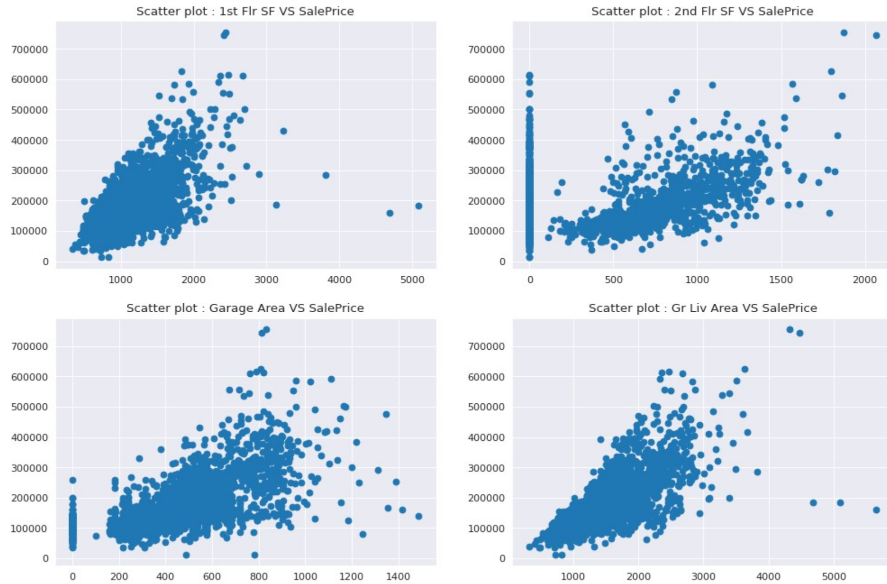
```
fig,[[fg1,fg2],[fg3,fg4]]=plt.subplots(nrows=2,ncols=2,figsize=(15,10))
```

```
fg1.scatter(df['1st Flr SF'],df['SalePrice'])
fg1.set_title('Scatter plot : 1st Flr SF VS SalePrice')

fg2.scatter(df['2nd Flr SF'],df['SalePrice'])
fg2.set_title('Scatter plot : 2nd Flr SF VS SalePrice')

fg3.scatter(df['Garage Area'],df['SalePrice'])
fg3.set_title('Scatter plot : Garage Area VS SalePrice')

fg4.scatter(df['Gr Liv Area'],df['SalePrice'])
fg4.set_title('Scatter plot : Gr Liv Area VS SalePrice')
plt.show()
```



In the above gure we have taken four cloumns to nd the linear relationship between the All four plots clearly demonstrate a linear relationship with the scatter plot as the second FLr being signi cantly more scarped than the others.

ANALYSIS

Finding the best value of bins.

The towers or bars of a histogram are called bins. The height of each bin shows how many values from that data fall into that range

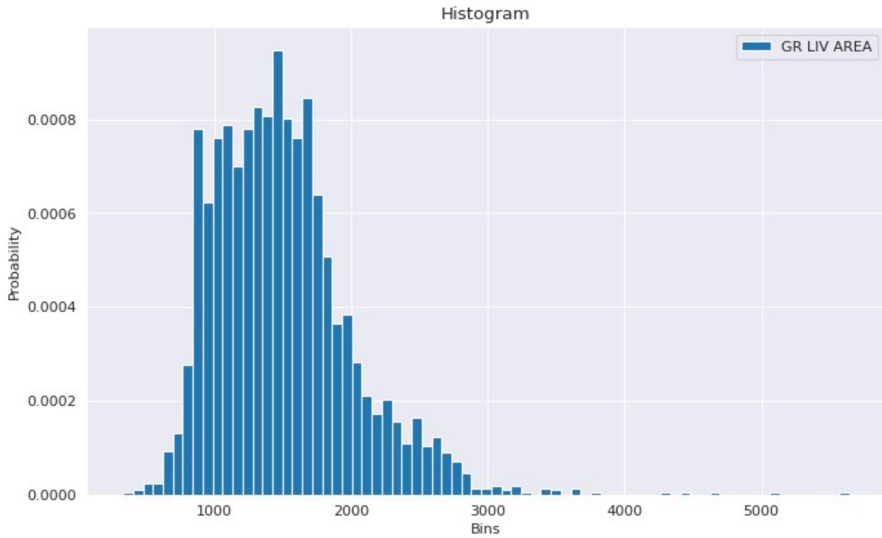
```
# Bin width= 2(q3 - q1)/ (n)1/3 (q1 is 1st quartile and q3 is third quartile and n is sample size)
# Bin = ceil(max(x)-min(x)/bin width)
# to make histogram we need to find the best number of bins.
```

```
gl_ar1,gl_ar3=np.percentile(df['Gr Liv Area'],[50,85]) bin_width = 2 * (gl_ar3 -
gl_ar1) * len(df['Gr Liv Area']) ** (-1/3) bins_count=int(np.ceil((df['Gr Liv
Area'].max() - df['Gr Liv Area'].min())/bin_width)) bins_count
```

```
73
```

We have found that the best number of Ground Living Area bins count is 73.

```
figure(figsize=(10,6)) plt.hist(df['Gr Liv
Area'],density=True,bins=bins_count,label="GR LIV AREA")
plt.legend(loc="upper right") plt.ylabel("Probability") plt.xlabel("Bins")
plt.title("Histogram");
```



The SKEWNESS of the histogram is a measure of the asymmetry of a distribution. A skew of 0 indicates that the data is normally distributed. A positive skew indicates that the tail of the distribution is skewed to the right side of the graph. A -ve skew indicates that the tial of the distribution is skewed to the left side of the graph. The histogram is skewed to the left, indicating that the majority of the residences are between 1000 and 2000 square feet.

```
df['Gr Liv Area'].describe()
```

```
count    2930.000000
mean     1499.690444
std       505.508887
min       334.000000
25%      1126.000000
50%      1442.000000 75%
1742.750000 max      5642.000000
Name: Gr Liv Area, dtype: float64
```

```
round(df['Gr Liv Area'].skew(),3)
```

```
1.274
```

The value is positive, indicating that the tail of the distribution is skewed to the right side of the graph.

Finding the vaalue of std.

The Pandas std() method is described as a way to gure out the standard deviation of a set of provided values, a DataFrame, a column, and a set of rows.

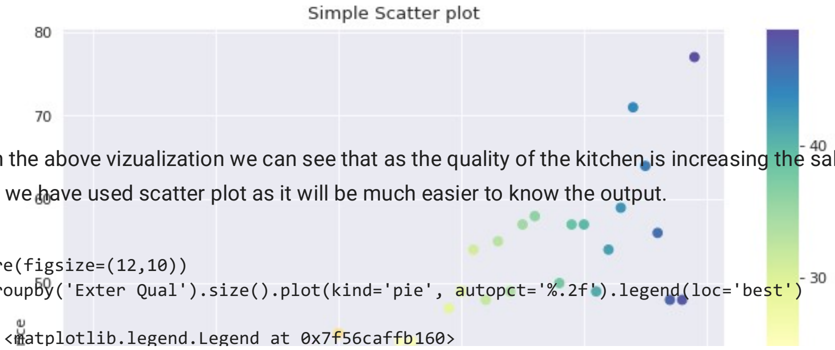
```
df.std()
```

```
Order              8.459625e+02
PID                1.887308e+08
MS SubClass        4.263802e+01
Lot Frontage       2.336533e+01
Lot Area           7.880018e+03
Overall Qual       1.411026e+00
Overall Cond       1.111537e+00
Year Built         3.024536e+01
Year Remod/Add     2.086029e+01
Mas Vnr Area       1.791126e+02
BsmtFin SF 1       4.555908e+02
BsmtFin SF 2       1.691685e+02
Bsmt Unf SF        4.394942e+02
Total Bsmt SF      4.406151e+02
1st Flr SF         3.918909e+02
2nd Flr SF         4.283957e+02
Low Qual Fin SF    4.631051e+01
```

```
Gr Liv Area        5.055089e+02
Bsmt Full Bath     5.248202e-01
Bsmt Half Bath     2.452536e-01
Full Bath          5.529406e-01
Half Bath          5.026293e-01
Bedroom AbvGr      8.277311e-01
Kitchen AbvGr      2.140762e-01
TotRms AbvGrd      1.572964e+00
Fireplaces         6.479209e-01
Garage Yr Blt      2.552841e+01
Garage Cars        7.605664e-01
Garage Area        2.150465e+02
Wood Deck SF       1.263616e+02
Open Porch SF      6.748340e+01
Enclosed Porch     6.413906e+01
3Ssn Porch         2.514133e+01
Screen Porch       5.608737e+01
Pool Area          3.559718e+01
Misc Val           5.663443e+02
Mo Sold            2.714492e+00
Yr Sold            1.316613e+00
SalePrice          7.988669e+04
dtype: float64
```

```
# VISUALISATION
```

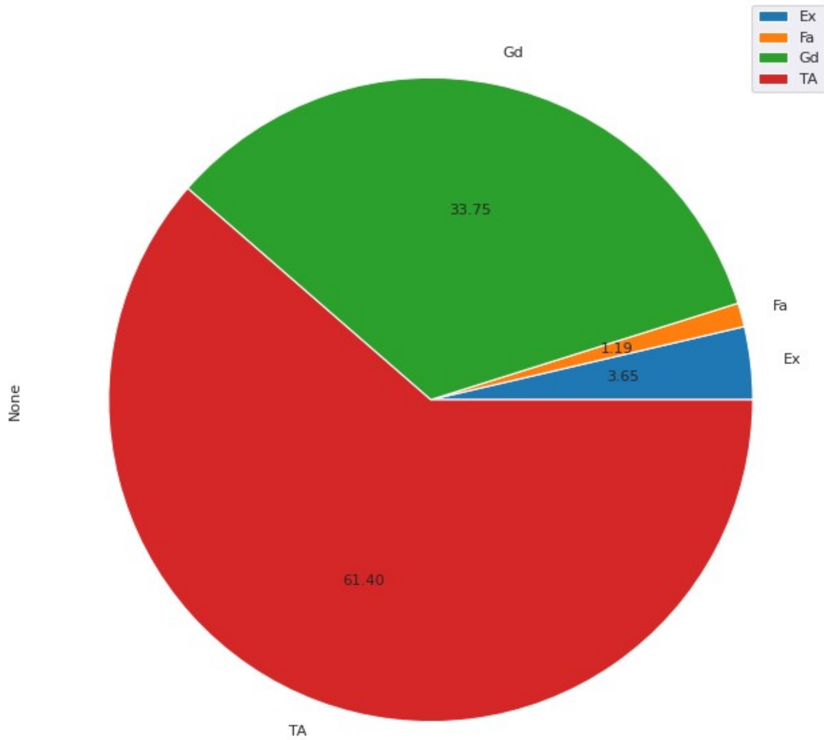
```
KitchenQual = range(50)
SalePrice = range(50) + np.random.randint(0,30,50)
plt.rcParams.update({'figure.figsize':(10,8), 'figure.dpi':80})
plt.scatter(KitchenQual, SalePrice, c=KitchenQual,
cmap='Spectral') plt.colorbar() plt.title('Simple Scatter plot')
plt.xlabel('KitchenQual') plt.ylabel('SalePrice') plt.show()
```



From the above vizualization we can see that as the quality of the kitchen is increasing the sale price automatically increasing. To vizulize the bove we have used scatter plot as it will be much easier to know the output.

```
figure(figsize=(12,10))
df.groupby('Exter Qual').size().plot(kind='pie', autopct='%.2f').legend(loc='best')
```

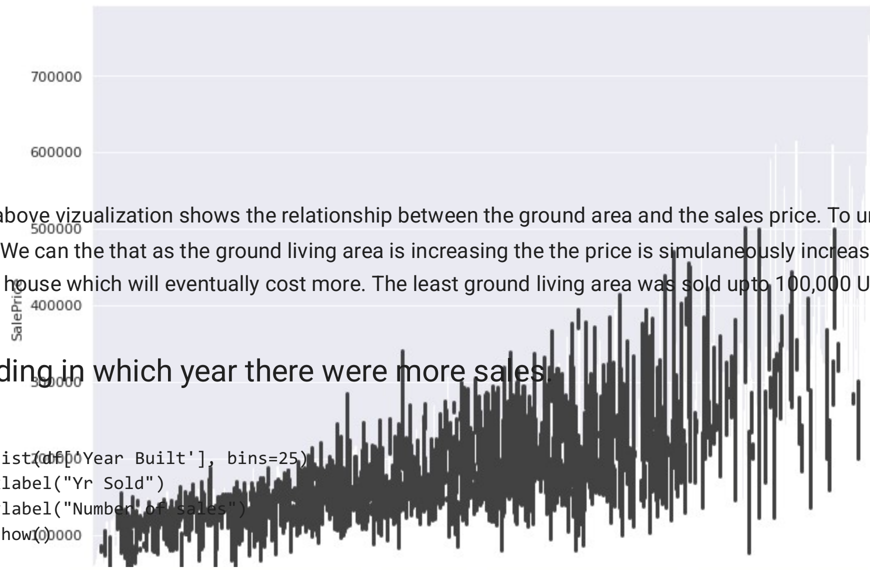
```
<matplotlib.legend.Legend at 0x7f56caffb160>
```



To understand the above vizualization we have used pie to to get a better understanding of the quality of the material used in the exterior. Most of the houses have an averge quality of the material which was used in exterior. although 33.75% of the houses have used a good quality of the material. only 1.19% of the total houses used a fair quality material.

```
sns.barplot(df["Gr Liv Area"], df["SalePrice"])
```

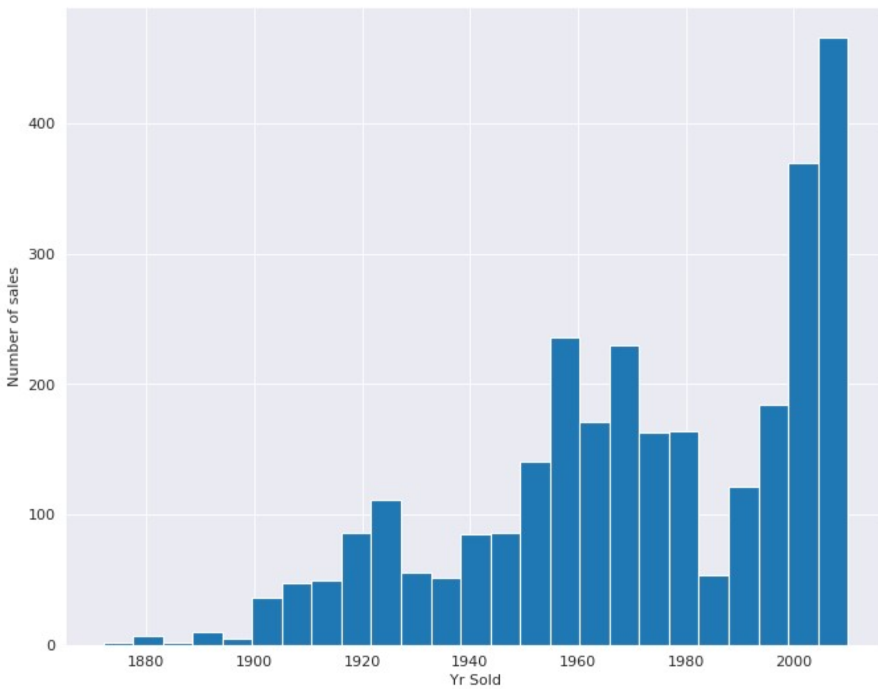
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56cad67c70>
```



The above vizualization shows the relationship between the ground area and the sales price. To understand our vizualization we have used bar plot. We can the that as the ground living area is increasing the the price is simulaneously increasing. If there is a bigger family they need to buy a big house which will eventually cost more. The least ground living area was sold upto 100,000 USD.

Finding in which year there were more sales.

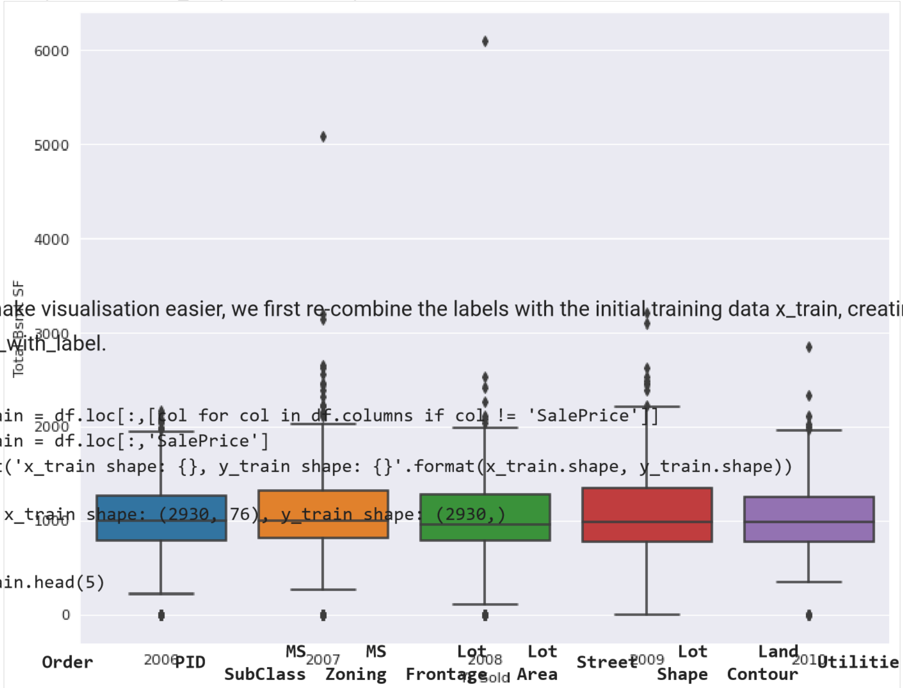
```
plt.hist(df['Year Built'], bins=25)
plt.xlabel("Yr Sold")
plt.ylabel("Number of sales")
plt.show()
```



We have used histogram to depict our relationship in year most of the houses were sold. From the above vizualization we can see that most of the houses were 2000

```
sns.boxplot(df["Yr Sold"], df["Total Bsmt SF"])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f56c8e693a0>



To make visualisation easier, we first re-combine the labels with the initial training data x_train, creating a dataset we'll refer to as train_with_label.

```
x_train = df.loc[:,[col for col in df.columns if col != 'SalePrice']]
y_train = df.loc[:,'SalePrice']
print('x_train shape: {}, y_train shape: {}'.format(x_train.shape, y_train.shape))
```

    x_train shape: (2930, 76), y_train shape: (2930,)

```
x_train.head(5)
```

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Lot Shape	Land Contour	Utilitie
0	1	526301100	20	RL	141.0	31770	Pave	IR1	Lvl	AllPu
1	2	526350040	20	RH	80.0	11622	Pave	Reg	Lvl	AllPu
2	3	526351010	20	RL	81.0	14267	Pave	IR1	Lvl	AllPu
3	4	526353030	20	RL	93.0	11160	Pave	Reg	Lvl	AllPu
4	5	527105010	60	RL	74.0	13830	Pave	IR1	Lvl	AllPu

5 rows × 76 columns

```
# label = pd.concat([x_train, y_train],
axis=1) label.shape
```

    (2930, 77)

Analysis of the relation between the Foundation and the Sale Price.

let's create a dataframe of Foundations with the median prices sorted down.

Nominal variable boxplot analysis

The link between foundation and SalePrice is examined.

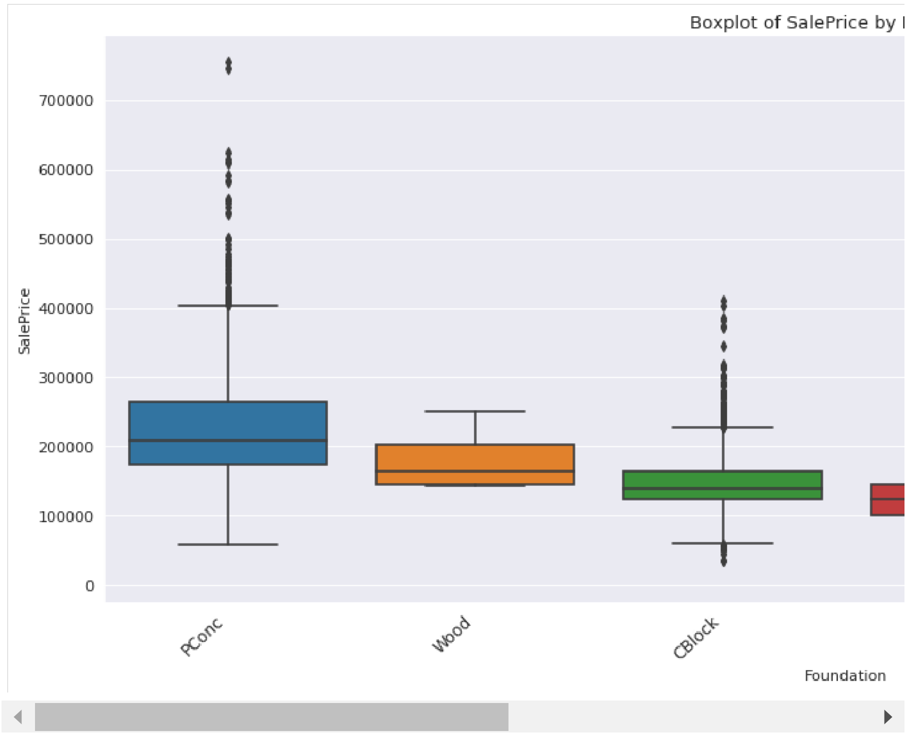
Aside from the size of the living space, foundation is an important aspect in deciding the price of a property. This is recorded in the original train dataset's Foundation column.

```
df['Foundation'].describe()
```

```
count      2930 unique       6
top       PConc freq       1310
Name: Foundation, dtype: object
```

```
median_price_Foundation = label.groupby('Foundation') \
                                    .agg(Sale_Price=('SalePrice',np.median)) \
                                    .sort_values('Sale_Price', ascending=False)
import seaborn as sns sns.set_style("darkgrid") g = sns.catplot(x='Foundation',                 y='SalePrice',
data=label,                 order = list(median_price_Foundation.index), # index of median table contains Foundation
sorted by price                 kind='box', height=6, aspect=2.5)
g.set_xticklabels(rotation=43, ha='right', size=11)
plt.title(label='Boxplot of SalePrice by Foundation')
```

```
plt.show()
```



From the above boxplot we can see that Pcon(poured concrete) has the highest number of sale when compared to different foundation layers. The least foundation layer that was sold was wood.

Describing sale price

```
df['SalePrice'].describe()
```

```
count       2930.000000
mean      180796.060068
std        79886.692357
min        12789.000000
25%       129500.000000
50%      160000.000000 75%
213500.000000 max
755000.000000 Name: SalePrice,
dtype: float64
```

```
df.corr()['SalePrice'].sort_values()
```

```
PID               -0.246521
Enclosed Porch    -0.128787
Kitchen AbvGr     -0.119814
Overall Cond      -0.101697
MS SubClass       -0.085092
Low Qual Fin SF   -0.037660
Bsmt Half Bath    -0.035835
Order             -0.031408
Yr Sold           -0.030569
Misc Val          -0.015691
BsmtFin SF 2       0.005891
3Ssn Porch         0.032225
Mo Sold            0.035259
Pool Area          0.068403
Screen Porch       0.112151
Bedroom AbvGr      0.143913
Bsmt Unf SF        0.182855
Lot Area           0.266549
2nd Flr SF         0.269373
Bsmt Full Bath     0.276050
Half Bath          0.285056
Open Porch SF      0.312951
Wood Deck SF       0.327143
Lot Frontage       0.357318
```

```
BsmtFin SF 1       0.432914
Fireplaces         0.474558
TotRms AbvGrd      0.495474
Mas Vnr Area       0.508285
Garage Yr Blt      0.526965
Year Remod/Add     0.532974
Full Bath          0.545604
Year Built         0.558426
1st Flr SF         0.621676
Total Bsmt SF      0.632280
Garage Area        0.640401
Garage Cars        0.647877
Gr Liv Area        0.706780
Overall Qual       0.799262
SalePrice          1.000000
Name: SalePrice, dtype: float64
```

Corr

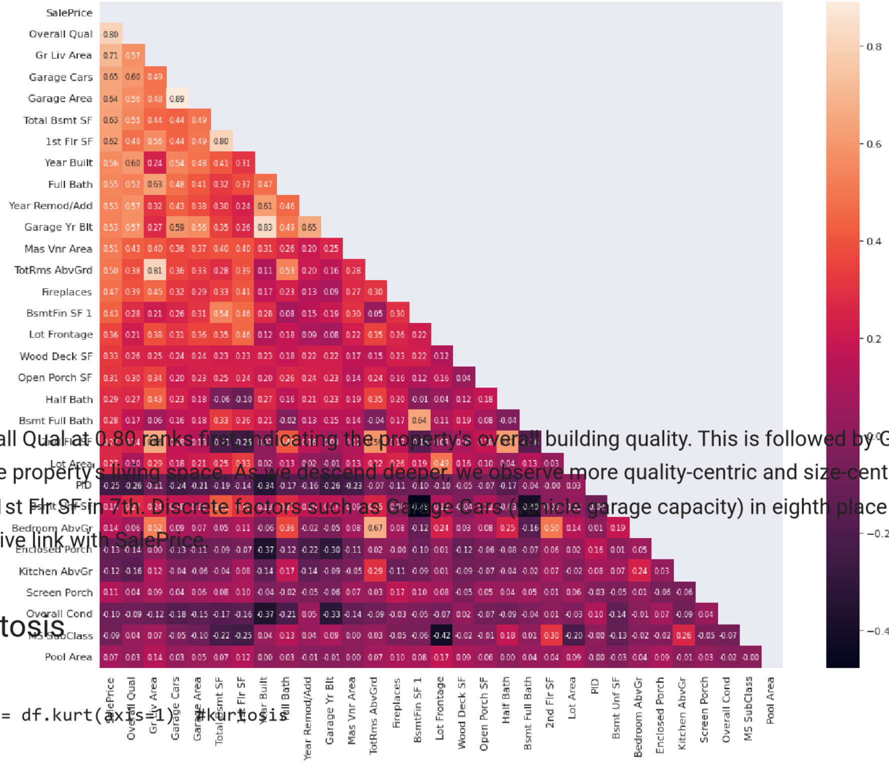
We see a general rising trend between the two columns, indicating that the sale price rises in tandem with the size of the living area. We also see that the majority of property living space sizes range from 750 to 2,000 square feet, and the majority of costs appear to range from 75,000 to 200,000 dollars.

```
import seaborn as sns correlation =
label.corr().loc[:,'SalePrice'] top_corr =
correlation.abs().sort_values(ascending=False).head(31) top_corr
= label.loc[:,list(top_corr.index)].corr()

mask = np.zeros_like(top_corr)
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize=(15, 12))

g = sns.heatmap(top_corr, annot=True, annot_kws={"size":7}, fmt='.2f', mask=mask)
g.set_xticklabels(g.get_xticklabels(), rotation=90)

plt.show()
```



Overall Qual at 0.80 ranks first, indicating the property's overall building quality. This is followed by Gr Liv Area at 0.72, which represents the size of the property's living space. As we descend deeper, we observe more quality-centric and size-centric variables, such as Bsmt Qual in 9th place and 1st Flr SF in 7th. Discrete factors such as Garage Cars (vehicle garage capacity) in eighth place and Year Built in tenth place show a positive link with SalePrice.

Kurtosis

```
kurt = df.kurt(axis=1) #kurtosis
```

```
kurt
```

```
0       38.999986
1       38.999997
2       38.999991
3       38.999982
4       38.999989
          ...
2925    38.999998
2926    37.999998
2927    37.999998
2928    38.999997
2929    38.999997
Length: 2930, dtype: float64
```