In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#data input
df = pd.read_csv("../DATA/Ames_Housing_Data.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

5 rows × 81 columns

In [4]:

```
#correlation between all features and sales price in sorted oder
#positive co-relation and value close to 1 means sales price closely depends on that parame
#in colrelation we can see sales price highly depends on Overall Qual
df.corr()['SalePrice'].sort_values()
```

Out[4]:

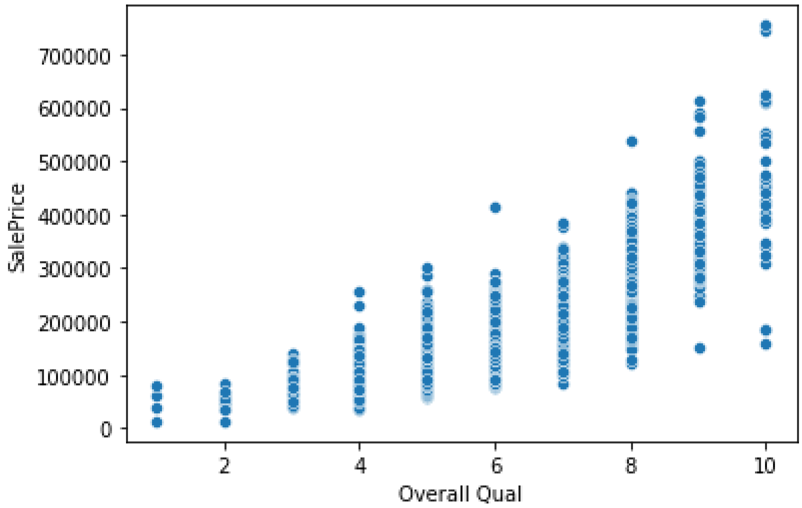
```
PID               -0.246521
Enclosed Porch    -0.128787
Kitchen AbvGr     -0.119814
Overall Cond      -0.101697
MS SubClass       -0.085092
Low Qual Fin SF   -0.037660
Bsmt Half Bath    -0.035835
Yr Sold           -0.030569
Misc Val          -0.015691
BsmtFin SF 2       0.005891
3Ssn Porch         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
```

In [5]:

```
#since sales price highly depends on overall quality
sns.scatterplot(x='Overall Qual',y='SalePrice',data=df)
#We notice higher quality higher is sales price but their are few outlier points which we n
```

Out[5]:

```
<AxesSubplot:xlabel='Overall Qual', ylabel='SalePrice'>
```



In [6]:

```
#from above we notice there are 3 houses which have quality between 8-10 but are selling ve
#Therefore they are doubtfull point
```

In [7]:

```
df[(df['Overall Qual']>8) & (df['SalePrice']<200000)]
```

Out[7]:

	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities
1182	533350090	60	RL	NaN	24572	Pave	NaN	IR1	Lvl	AllPub
1498	908154235	60	RL	313.0	63887	Pave	NaN	IR3	Bnk	AllPub
2180	908154195	20	RL	128.0	39290	Pave	NaN	IR1	Bnk	AllPub
2181	908154205	60	RL	130.0	40094	Pave	NaN	IR1	Bnk	AllPub

4 rows × 81 columns

In [8]:

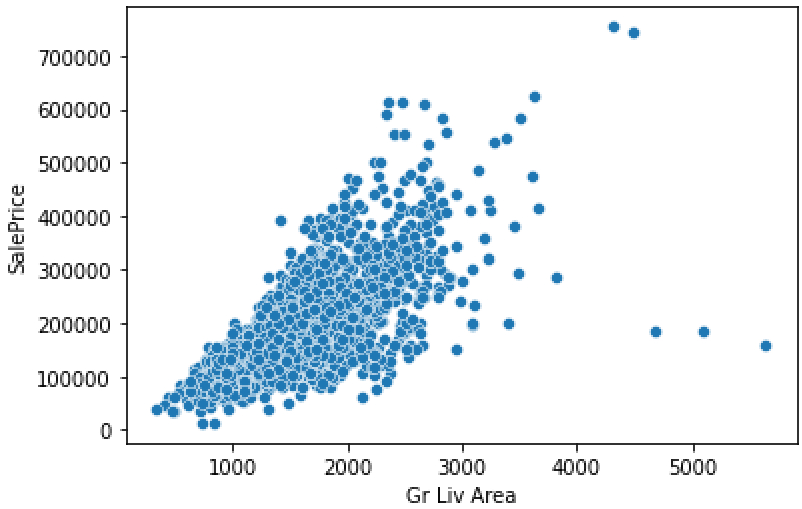
```
#The four suspectful rows are displayed above
```

In [9]:

```
#Gr Liv Area is also highly correlated to sales price
sns.scatterplot(x='Gr Liv Area',y='SalePrice',data=df)
```

Out[9]:

```
<AxesSubplot:xlabel='Gr Liv Area', ylabel='SalePrice'>
```



In [10]:

```
#We again notice that the above mentioned three houses here again show weird behaviour
#The general trend is higher the Gr Liv Area more is the SalesPrice
#But those three houses Have high Gr Liv Area But small selling price
```

In [11]:

```
df[(df['Gr Liv Area']>4000) & (df['SalePrice']<400000)]
```

Out[11]:

	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilitie
1498	908154235	60	RL	313.0	63887	Pave	NaN	IR3	Bnk	AllPu
2180	908154195	20	RL	128.0	39290	Pave	NaN	IR1	Bnk	AllPu
2181	908154205	60	RL	130.0	40094	Pave	NaN	IR1	Bnk	AllPu

3 rows × 81 columns

In [12]:

```
#We get the three rows with high Gr Liv Area but low sales price. These three rows match th
#There are outliers which should be removed
```

In [13]:

```
df[(df['Gr Liv Area']>4000) & (df['SalePrice']<400000)].index
```

Out[13]:

```
Int64Index([1498, 2180, 2181], dtype='int64')
```

In [14]:

```
ind_drop = df[(df['Gr Liv Area']>4000) & (df['SalePrice']<400000)].index
```

In [15]:

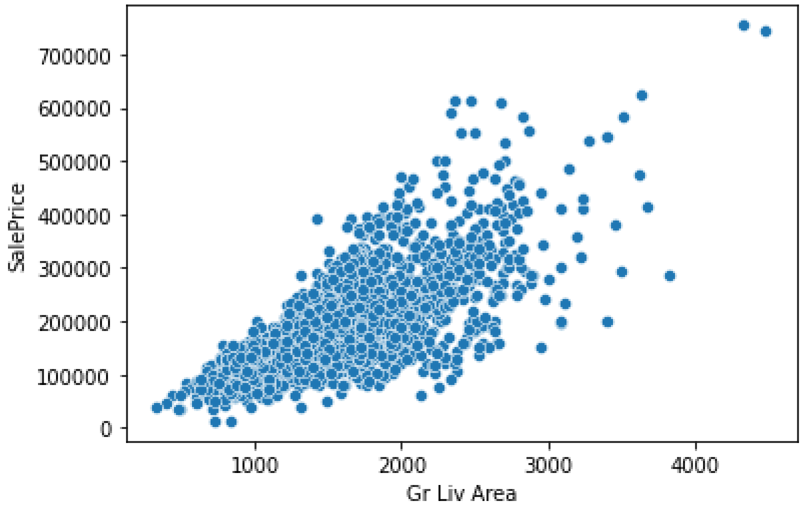
```
#Those three rows dropped
df = df.drop(ind_drop,axis=0)
```

In [16]:

```
sns.scatterplot(x='Gr Liv Area',y='SalePrice',data=df)
```

Out[16]:

```
<AxesSubplot:xlabel='Gr Liv Area', ylabel='SalePrice'>
```



In [17]:

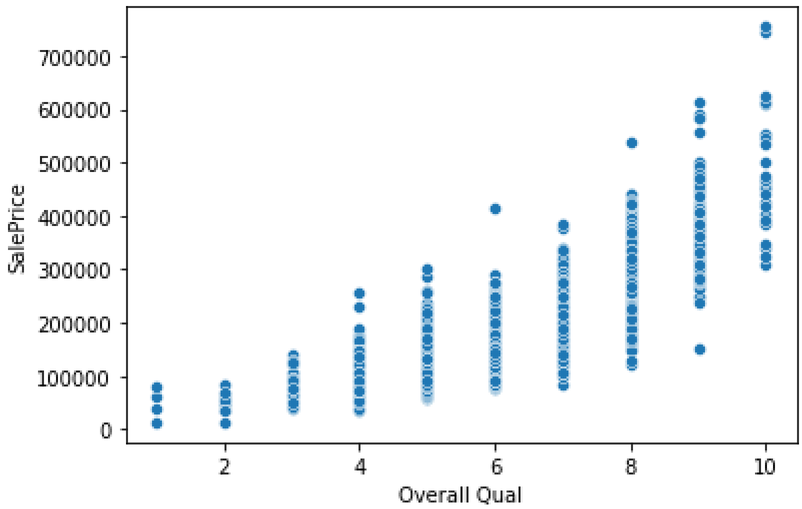
```
#Now the rest of the data set seems to follow the general trend
```

In [18]:

```
sns.scatterplot(x='Overall Qual',y='SalePrice',data=df)
```

Out[18]:

```
<AxesSubplot:xlabel='Overall Qual', ylabel='SalePrice'>
```



In [19]:

```
#Rest of data seems to follow the general trend
```

In [ ]:

```

```

In [20]:

```
#WE DEALT WITH OUTLIERS NOW WE DEAL WITH MISSUNG DATA
```

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2927 entries, 0 to 2929
Data columns (total 81 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   PID              2927 non-null   int64
 1   MS SubClass      2927 non-null   int64
 2   MS Zoning        2927 non-null   object
 3   Lot Frontage     2437 non-null   float64
 4   Lot Area         2927 non-null   int64
 5   Street           2927 non-null   object
 6   Alley            198 non-null    object
 7   Lot Shape        2927 non-null   object
 8   Land Contour     2927 non-null   object
 9   Utilities        2927 non-null   object
 10  Lot Config       2927 non-null   object
 11  Land Slope       2927 non-null   object
 12  Neighborhood     2927 non-null   object
 13  Condition 1      2927 non-null   object
 14  Condition 2      2927 non-null   object
 15  Bldg Type        2927 non-null   object
 16  House Style      2927 non-null   object
 17  Overall Qual     2927 non-null   int64
 18  Overall Cond     2927 non-null   int64
 19  Year Built       2927 non-null   int64
 20  Year Remod/Add   2927 non-null   int64
 21  Roof Style       2927 non-null   object
 22  Roof Matl        2927 non-null   object
 23  Exterior 1st     2927 non-null   object
 24  Exterior 2nd     2927 non-null   object
 25  Mas Vnr Type     2904 non-null   object
 26  Mas Vnr Area     2904 non-null   float64
 27  Exter Qual       2927 non-null   object
 28  Exter Cond       2927 non-null   object
 29  Foundation       2927 non-null   object
 30  Bsmt Qual        2847 non-null   object
 31  Bsmt Cond        2847 non-null   object
 32  Bsmt Exposure    2844 non-null   object
 33  BsmtFin Type 1   2847 non-null   object
 34  BsmtFin SF 1     2926 non-null   float64
 35  BsmtFin Type 2   2846 non-null   object
 36  BsmtFin SF 2     2926 non-null   float64
 37  Bsmt Unf SF      2926 non-null   float64
 38  Total Bsmt SF    2926 non-null   float64
 39  Heating          2927 non-null   object
 40  Heating QC       2927 non-null   object
 41  Central Air      2927 non-null   object
 42  Electrical       2926 non-null   object
 43  1st Flr SF       2927 non-null   int64
 44  2nd Flr SF       2927 non-null   int64
 45  Low Qual Fin SF  2927 non-null   int64
 46  Gr Liv Area      2927 non-null   int64
 47  Bsmt Full Bath   2925 non-null   float64
 48  Bsmt Half Bath   2925 non-null   float64
 49  Full Bath        2927 non-null   int64
 50  Half Bath        2927 non-null   int64
 51  Bedroom AbvGr    2927 non-null   int64
```

```
 52  Kitchen AbvGr    2927 non-null   int64
 53  Kitchen Qual     2927 non-null   object
 54  TotRms AbvGrd    2927 non-null   int64
 55  Functional       2927 non-null   object
 56  Fireplaces       2927 non-null   int64
 57  Fireplace Qu     1505 non-null   object
 58  Garage Type      2770 non-null   object
 59  Garage Yr Blt    2768 non-null   float64
 60  Garage Finish    2768 non-null   object
 61  Garage Cars      2926 non-null   float64
 62  Garage Area      2926 non-null   float64
 63  Garage Qual      2768 non-null   object
 64  Garage Cond      2768 non-null   object
 65  Paved Drive      2927 non-null   object
 66  Wood Deck SF     2927 non-null   int64
 67  Open Porch SF    2927 non-null   int64
 68  Enclosed Porch   2927 non-null   int64
 69  3Ssn Porch       2927 non-null   int64
 70  Screen Porch     2927 non-null   int64
 71  Pool Area        2927 non-null   int64
 72  Pool QC          12 non-null     object
 73  Fence            572 non-null    object
 74  Misc Feature     105 non-null    object
 75  Misc Val         2927 non-null   int64
 76  Mo Sold          2927 non-null   int64
 77  Yr Sold          2927 non-null   int64
 78  Sale Type        2927 non-null   object
 79  Sale Condition   2927 non-null   object
 80  SalePrice        2927 non-null   int64
dtypes: float64(11), int64(27), object(43)
memory usage: 1.8+ MB
```

In [22]:

```
#We Notice that there for few features there are missing values
```

In [23]:

```
df = df.drop('PID',axis=1)
```

In [24]:

```
df.head()
```

Out[24]:

	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities	Lot Config	...	Poo Are
0	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	AllPub	Corner	...	
1	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
2	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	Corner	...	
3	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	AllPub	Corner	...	
4	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	Inside	...	

5 rows × 80 columns

In [25]:

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

Out[25]:

```
MS SubClass          0
MS Zoning            0
Lot Frontage       490
Lot Area             0
Street               0
                  ...
Mo Sold              0
Yr Sold              0
Sale Type            0
Sale Condition       0
SalePrice            0
Length: 80, dtype: int64
```

In [26]:

```
#True treated as 0 and false as 1 therefore we get sum of how many rows for each feature ar
#there are 80 features therefore we cannot see all
```

In [27]:

```
100* df.isnull().sum() / len(df)
```

Out[27]:

```
MS SubClass        0.00000
MS Zoning          0.00000
Lot Frontage      16.74069
Lot Area           0.00000
Street             0.00000
                    ...
Mo Sold            0.00000
Yr Sold            0.00000
Sale Type          0.00000
Sale Condition     0.00000
SalePrice          0.00000
Length: 80, dtype: float64
```

In [28]:

```
#We now get what percentage of data is missing which would help us to evaluate better
```

In [29]:

```
def percent_missing(df):
    percent_nan = 100* df.isnull().sum() / len(df)
    percent_nan = percent_nan[percent_nan>0].sort_values()
    return percent_nan
```

In [30]:

```
percent_nan = percent_missing(df)
```

In [31]:

```
percent_nan
```

Out[31]:

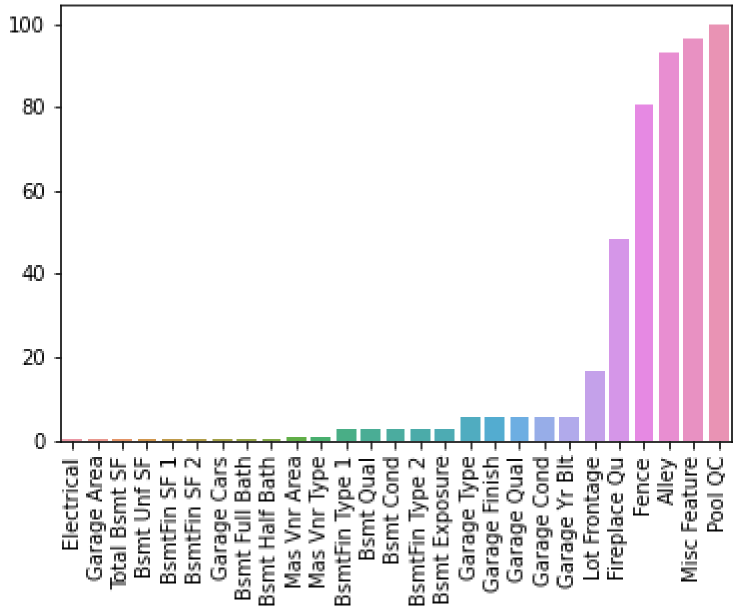
```
Electrical        0.034165
Garage Area       0.034165
Total Bsmt SF     0.034165
Bsmt Unf SF       0.034165
BsmtFin SF 1      0.034165
BsmtFin SF 2      0.034165
Garage Cars       0.034165
Bsmt Full Bath    0.068329
Bsmt Half Bath    0.068329
Mas Vnr Area      0.785787
Mas Vnr Type      0.785787
BsmtFin Type 1    2.733174
Bsmt Qual         2.733174
Bsmt Cond         2.733174
BsmtFin Type 2    2.767339
Bsmt Exposure     2.835668
Garage Type       5.363854
Garage Finish     5.432183
Garage Qual       5.432183
Garage Cond       5.432183
Garage Yr Blt     5.432183
Lot Frontage     16.740690
Fireplace Qu     48.582166
Fence            80.457807
Alley            93.235395
Misc Feature     96.412709
Pool QC          99.590024
dtype: float64
```

In [32]:

```
#We get percentage of missing data in sorted manner using a function call
```

In [33]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [34]:

```
#graphical representation of missing data
```

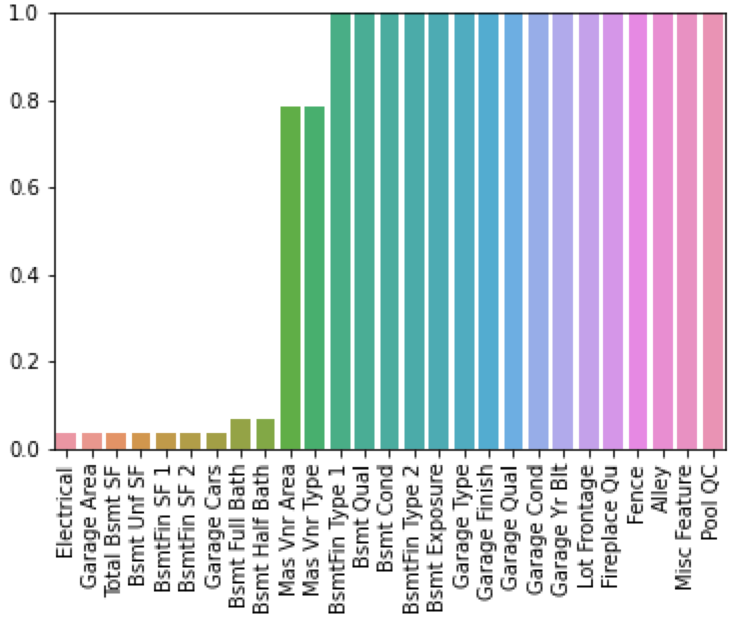
In [35]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);

# Set 1% Threshold
plt.ylim(0,1)
```

Out[35]:

```
(0.0, 1.0)
```



In [36]:

```
#I would not mind dropping rows with 1% missing data
```

In [37]:

```
percent_nan[percent_nan < 1]
```

Out[37]:

```
Electrical        0.034165
Garage Area       0.034165
Total Bsmt SF     0.034165
Bsmt Unf SF       0.034165
BsmtFin SF 1      0.034165
BsmtFin SF 2      0.034165
Garage Cars       0.034165
Bsmt Full Bath    0.068329
Bsmt Half Bath    0.068329
Mas Vnr Area      0.785787
Mas Vnr Type      0.785787
dtype: float64
```

In [38]:

```
#Features with less than 1% missing data
```

In [39]:

```
100/len(df)
```

Out[39]:

```
0.0341646737273659
```

In [40]:

```
#the above calculation tells that features like Electrical, Garage Area, Bsmt Unf SF etc ha
#is missing data
```

In [41]:

```
df = df.dropna(axis = 0, subset = ['Electrical','Garage Area'])
```

In [42]:

```
percent_nan = percent_missing(df)
```

In [43]:

```
percent_nan[percent_nan < 1]
```

Out[43]:

```
Bsmt Unf SF       0.034188
Total Bsmt SF     0.034188
BsmtFin SF 2      0.034188
BsmtFin SF 1      0.034188
Bsmt Full Bath    0.068376
Bsmt Half Bath    0.068376
Mas Vnr Type      0.786325
Mas Vnr Area      0.786325
dtype: float64
```

In [44]:

```
#We notice by dropping electrical and garage area we also dropped many other features which
#same row as above two features missing
```

In [45]:

```
df[df['Bsmt Half Bath'].isnull()]
```

Out[45]:

	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities	Lot Config	...
1341	20	RM	99.0	5940	Pave	NaN	IR1	Lvl	AllPub	FR3	...
1497	20	RL	123.0	47007	Pave	NaN	IR1	Lvl	AllPub	Inside	...

2 rows × 80 columns

In [46]:

```
df[df['Bsmt Full Bath'].isnull()]
```

Out[46]:

	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities	Lot Config	...
1341	20	RM	99.0	5940	Pave	NaN	IR1	Lvl	AllPub	FR3	...
1497	20	RL	123.0	47007	Pave	NaN	IR1	Lvl	AllPub	Inside	...

2 rows × 80 columns

In [47]:

```
#We notice that the same two rows are missing information for full bath and half bath
```

In [48]:

```
with open('../DATA/Ames_Housing_Feature_Description.txt','r') as f:
    print(f.read())
```

```
MSSubClass: Identifies the type of dwelling involved in the sale.

        20      1-STORY 1946 & NEWER ALL STYLES
        30      1-STORY 1945 & OLDER
        40      1-STORY W/FINISHED ATTIC ALL AGES
        45      1-1/2 STORY - UNFINISHED ALL AGES
        50      1-1/2 STORY FINISHED ALL AGES
        60      2-STORY 1946 & NEWER
        70      2-STORY 1945 & OLDER
        75      2-1/2 STORY ALL AGES
        80      SPLIT OR MULTI-LEVEL
        85      SPLIT FOYER
        90      DUPLEX - ALL STYLES AND AGES
       120      1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150      1-1/2 STORY PUD - ALL AGES
       160      2-STORY PUD - 1946 & NEWER
       180      PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190      2 FAMILY CONVERSION - ALL STYLES AND AGES
```

In [49]:

```
#In data description we see NA for various basement parameters means there is no basement i
#Therefore instead of dropping them for numerical basement values we can fill them with zer
#And for string basement values we can fill them with none thus telling that there are no b
```

In [50]:

```
#BSMT numeric coloumn --> 0
bsmt_num_cols = ['BsmtFin SF 1', 'BsmtFin SF 2', 'Bsmt Unf SF','Total Bsmt SF', 'Bsmt Full
df[bsmt_num_cols] = df[bsmt_num_cols].fillna(0)
```

In [51]:

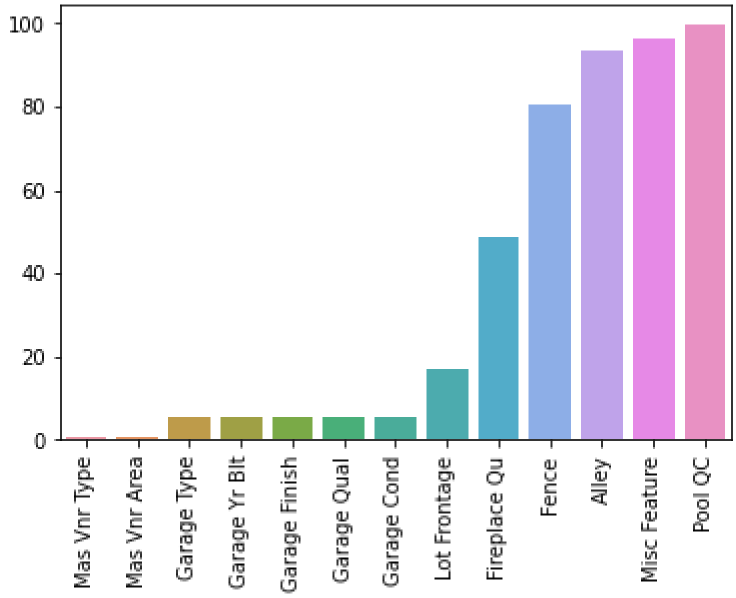
```
#BSMT string colom --> NONE
bsmt_str_cols =  ['Bsmt Qual', 'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin Typ
df[bsmt_str_cols] = df[bsmt_str_cols].fillna('None')
```

In [52]:

```
percent_nan = percent_missing(df)
```

In [53]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [54]:

```
#Now again in case of Mas VNR Type and Area when we refer dataframe detail we find there no
#Therefore repeat same above process for them as well
```

In [55]:

```
df['Mas Vnr Type'] = df['Mas Vnr Type'].fillna('None')
```

In [56]:

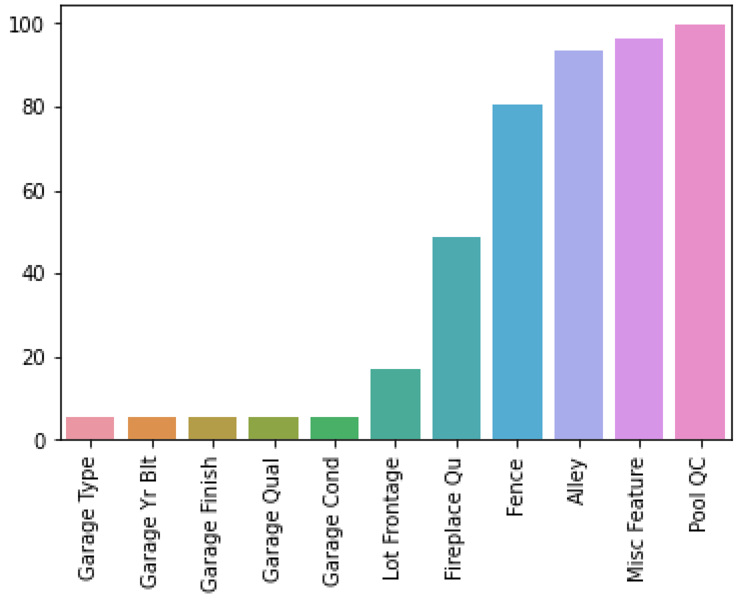
```
df['Mas Vnr Area'] = df['Mas Vnr Area'].fillna(0)
```

In [57]:

```
percent_nan = percent_missing(df)
```

In [58]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [59]:

```
#Now we are above the 1% thresholdd range for dropping rows,therefore we need to consider c
```

In [60]:

```
#Now again for garage coloms NA n=means no garage exit therefore we repeat same above proce
```

In [61]:

```
gar_str_cols = ['Garage Type', 'Garage Finish', 'Garage Qual', 'Garage Cond']
df[gar_str_cols] = df[gar_str_cols].fillna('None')
```

In [62]:

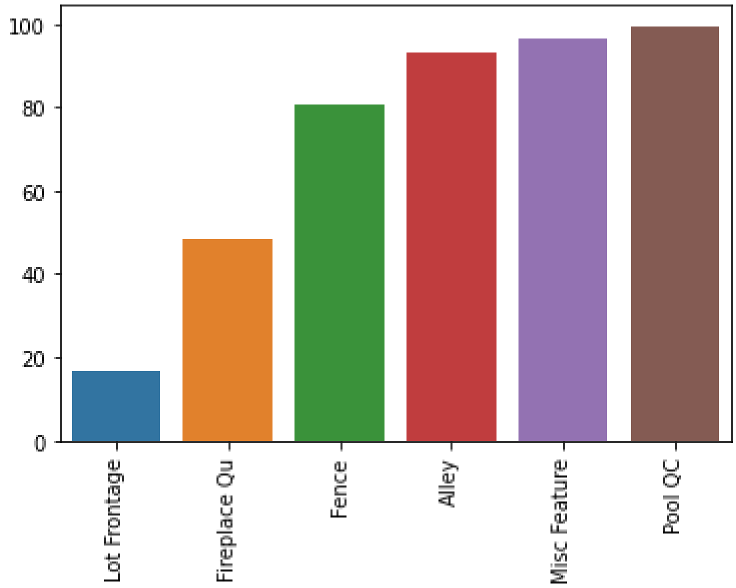
```
df['Garage Yr Blt'] = df['Garage Yr Blt'].fillna(0)
```

In [63]:

```
percent_nan = percent_missing(df)
```

In [64]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [65]:

```
#Fence Alley Misc Feature Pool QC are missing more than 80% of data therefore it is better
```

In [66]:

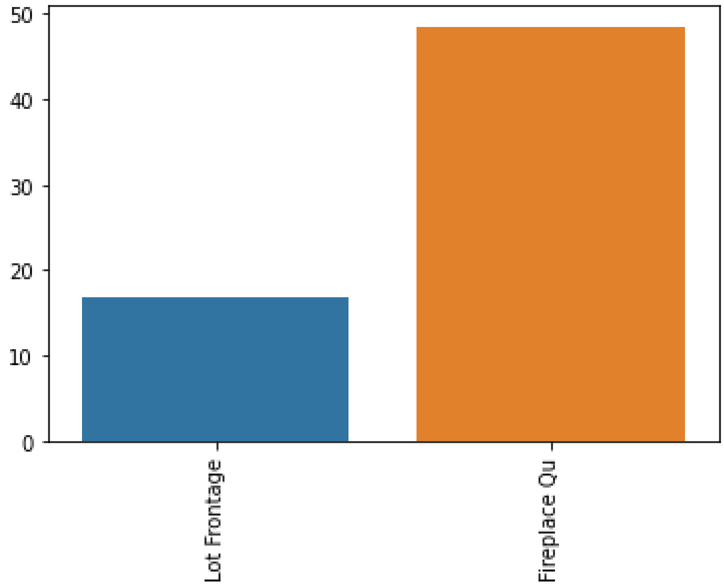
```
df = df.drop(['Pool QC','Misc Feature','Alley','Fence'],axis=1)
```

In [67]:

```
percent_nan = percent_missing(df)
```

In [68]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [69]:

```
#In remaining two features we can neither drop rows because too many rows are missing
#nore we can drop coloms because not that much data is missing
```

In [70]:

```
df['Fireplace Qu'].value_counts()
```

Out[70]:

```
Gd    741
TA    600
Fa     75
Po     46
Ex     43
Name: Fireplace Qu, dtype: int64
```

In [71]:

```
#We notice it is a string colom therefore we just fill None in missing values
```

In [72]:

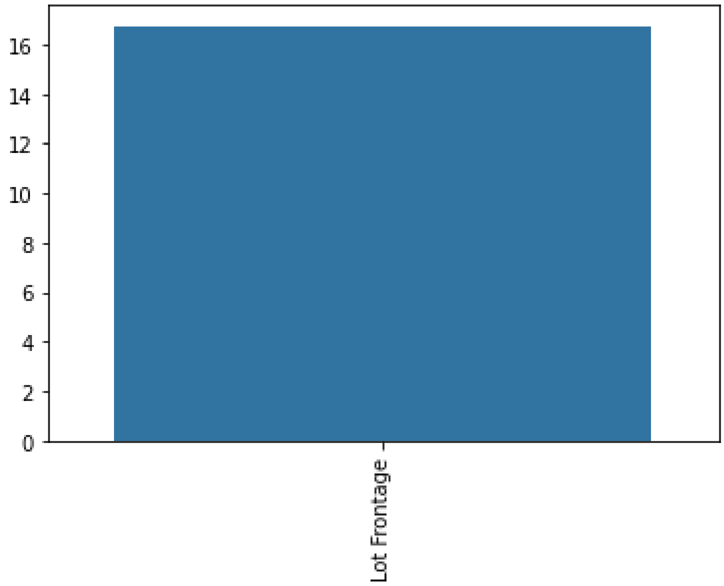
```
df['Fireplace Qu'] = df['Fireplace Qu'].fillna("None")
```

In [73]:

```
percent_nan = percent_missing(df)
```

In [74]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [75]:

```
df['Lot Frontage']
```

Out[75]:

```
0       141.0
1        80.0
2        81.0
3        93.0
4        74.0
        ...
2925     37.0
2926      NaN
2927     62.0
2928     77.0
2929     74.0
Name: Lot Frontage, Length: 2925, dtype: float64
```

In [76]:

```
with open('../DATA/Ames_Housing_Feature_Description.txt','r') as f:
    print(f.read())
```

```
MSSubClass: Identifies the type of dwelling involved in the sale.

        20      1-STORY 1946 & NEWER ALL STYLES
        30      1-STORY 1945 & OLDER
        40      1-STORY W/FINISHED ATTIC ALL AGES
        45      1-1/2 STORY - UNFINISHED ALL AGES
        50      1-1/2 STORY FINISHED ALL AGES
        60      2-STORY 1946 & NEWER
        70      2-STORY 1945 & OLDER
        75      2-1/2 STORY ALL AGES
        80      SPLIT OR MULTI-LEVEL
        85      SPLIT FOYER
        90      DUPLEX - ALL STYLES AND AGES
       120      1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150      1-1/2 STORY PUD - ALL AGES
       160      2-STORY PUD - 1946 & NEWER
       180      PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190      2 FAMILY CONVERSION - ALL STYLES AND AGES
```

In [77]:

```
#Thus we see Lot Frontage is Numeric data and it is Linear feet of street connected to prop
```

In [78]:

```
# Neighborhood: Physical locations within Ames city limits

# LotFrontage: Linear feet of street connected to property

# We will operate under the assumption that the Lot Frontage is related to what neighborhoo
```

In [79]:

```
df['Neighborhood'].unique()
```

Out[79]:

```
array(['NAmes', 'Gilbert', 'StoneBr', 'NWAmes', 'Somerst', 'BrDale',
       'NPkVill', 'NridgHt', 'Blmngtn', 'NoRidge', 'SawyerW', 'Sawyer',
       'Greens', 'BrkSide', 'OldTown', 'IDOTRR', 'ClearCr', 'SWISU',
       'Edwards', 'CollgCr', 'Crawfor', 'Blueste', 'Mitchel', 'Timber',
       'MeadowV', 'Veenker', 'GrnHill', 'Landmrk'], dtype=object)
```

In [80]:

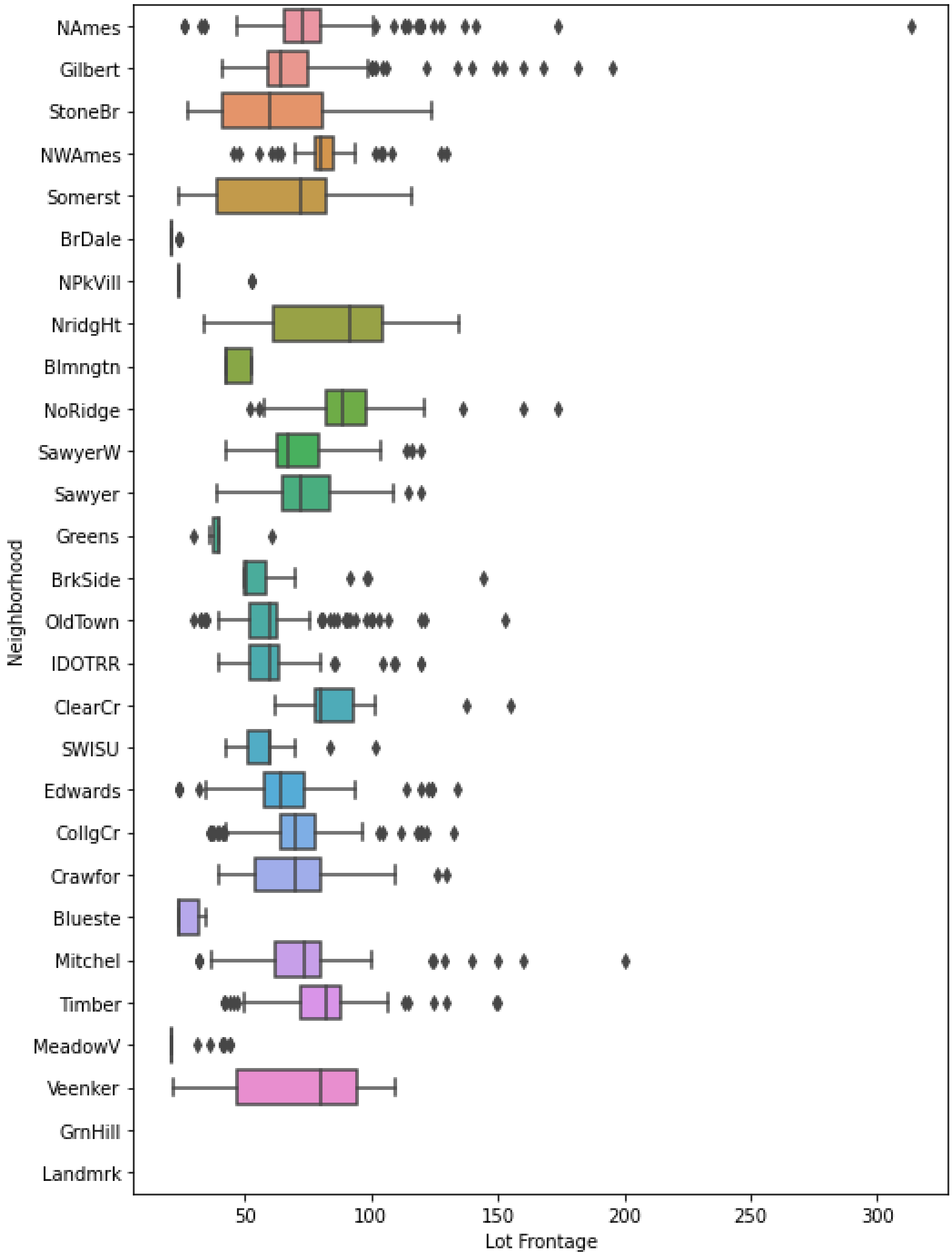
```
#All neighbourhoods
```

In [81]:

```
plt.figure(figsize=(8,12))
sns.boxplot(x='Lot Frontage',y='Neighborhood',data=df,orient='h')
```

Out[81]:

<AxesSubplot:xlabel='Lot Frontage', ylabel='Neighborhood'>



In [82]:

```
df.groupby('Neighborhood')['Lot Frontage']
```

Out[82]:

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x0000016520CBDFA0>
```

In [83]:

```
df.groupby('Neighborhood')['Lot Frontage'].mean()
```

Out[83]:

```
Neighborhood
Blmngtn    46.900000
Blueste    27.300000
BrDale     21.500000
BrkSide    55.789474
ClearCr    88.150000
CollgCr    71.336364
Crawfor    69.951807
Edwards    64.794286
Gilbert    74.207207
Greens     41.000000
GrnHill          NaN
IDOTRR     62.383721
Landmrk          NaN
MeadowV    25.606061
Mitchel    75.144444
NAmes      75.210667
NPkVill    28.142857
NWAmes     81.517647
NoRidge    91.629630
NridgHt    84.184049
OldTown    61.777293
SWISU      59.068182
Sawyer     74.551020
SawyerW    70.669811
Somerst    64.549383
StoneBr    62.173913
Timber     81.303571
Veenker    72.000000
Name: Lot Frontage, dtype: float64
```

In [84]:

```
#Therefore now we have average lot frontage value for each neighbourhood
#for missing data we fill it with average value of that neighbourhod
```

In [85]:

```
df.groupby('Neighborhood')['Lot Frontage'].transform(lambda val: val.fillna(val.mean()))
```

Out[85]:

```
0       141.000000
1        80.000000
2        81.000000
3        93.000000
4        74.000000
           ...
2925     37.000000
2926     75.144444
2927     62.000000
2928     77.000000
2929     74.000000
Name: Lot Frontage, Length: 2925, dtype: float64
```

In [86]:

```
df['Lot Frontage'] = df.groupby('Neighborhood')['Lot Frontage'].transform(lambda val: val.f
```

In [87]:

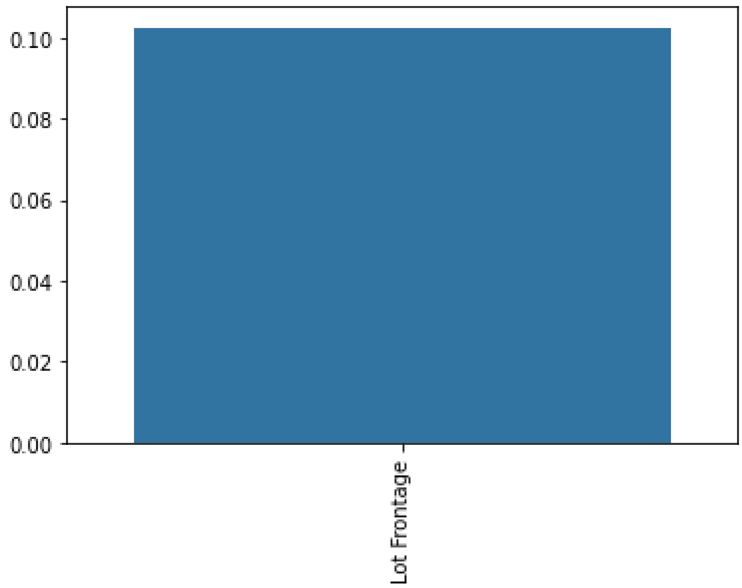
```
#We filled the missing Lot Frontage values
```

In [88]:

```
percent_nan = percent_missing(df)
```

In [89]:

```
sns.barplot(x=percent_nan.index,y=percent_nan)
plt.xticks(rotation=90);
```



In [90]:

```
df['Lot Frontage'] = df['Lot Frontage'].fillna(0)
```

In [91]:

```
percent_nan = percent_missing(df)
```

Dealing with categorical data

In [92]:

```
# Convert to String
df['MS SubClass'] = df['MS SubClass'].apply(str)
```

In [93]:

```
df.select_dtypes(include='object')
```

Out[93]:

	MS SubClass	MS Zoning	Street	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	Con
0	20	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	
1	20	RH	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	
2	20	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	
3	20	RL	Pave	Reg	Lvl	AllPub	Corner	Gtl	NAmes	
4	60	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	
...	...	...	...	...	...	...	...	...	...	
2925	80	RL	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	Mitchel	
2926	20	RL	Pave	IR1	Low	AllPub	Inside	Mod	Mitchel	
2927	85	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	
2928	20	RL	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	
2929	60	RL	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	

2925 rows × 40 columns

In [94]:

```
df_nums = df.select_dtypes(exclude='object')
df_objs = df.select_dtypes(include='object')
```

In [95]:

```
df_nums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2925 entries, 0 to 2929
Data columns (total 36 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Lot Frontage     2925 non-null   float64
 1   Lot Area         2925 non-null   int64  
 2   Overall Qual     2925 non-null   int64  
 3   Overall Cond     2925 non-null   int64  
 4   Year Built       2925 non-null   int64  
 5   Year Remod/Add   2925 non-null   int64  
 6   Mas Vnr Area     2925 non-null   float64
 7   BsmtFin SF 1     2925 non-null   float64
 8   BsmtFin SF 2     2925 non-null   float64
 9   Bsmt Unf SF      2925 non-null   float64
 10  Total Bsmt SF    2925 non-null   float64
 11  1st Flr SF       2925 non-null   int64  
 12  2nd Flr SF       2925 non-null   int64  
 13  Low Qual Fin SF  2925 non-null   int64  
 14  Gr Liv Area      2925 non-null   int64  
 15  Bsmt Full Bath   2925 non-null   float64
 16  Bsmt Half Bath   2925 non-null   float64
 17  Full Bath        2925 non-null   int64  
 18  Half Bath        2925 non-null   int64  
 19  Bedroom AbvGr    2925 non-null   int64  
 20  Kitchen AbvGr    2925 non-null   int64  
 21  TotRms AbvGrd    2925 non-null   int64  
 22  Fireplaces       2925 non-null   int64  
 23  Garage Yr Blt    2925 non-null   float64
 24  Garage Cars      2925 non-null   float64
 25  Garage Area      2925 non-null   float64
 26  Wood Deck SF     2925 non-null   int64  
 27  Open Porch SF    2925 non-null   int64  
 28  Enclosed Porch   2925 non-null   int64  
 29  3Ssn Porch       2925 non-null   int64  
 30  Screen Porch     2925 non-null   int64  
 31  Pool Area        2925 non-null   int64  
 32  Misc Val         2925 non-null   int64  
 33  Mo Sold          2925 non-null   int64  
 34  Yr Sold          2925 non-null   int64  
 35  SalePrice        2925 non-null   int64  
dtypes: float64(11), int64(25)
memory usage: 910.0 KB
```

In [96]:

```
df_objs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2925 entries, 0 to 2929
Data columns (total 40 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   MS SubClass     2925 non-null   object
 1   MS Zoning       2925 non-null   object
 2   Street          2925 non-null   object
 3   Lot Shape       2925 non-null   object
 4   Land Contour    2925 non-null   object
 5   Utilities       2925 non-null   object
 6   Lot Config      2925 non-null   object
 7   Land Slope      2925 non-null   object
 8   Neighborhood    2925 non-null   object
 9   Condition 1     2925 non-null   object
 10  Condition 2     2925 non-null   object
 11  Bldg Type       2925 non-null   object
 12  House Style     2925 non-null   object
 13  Roof Style      2925 non-null   object
 14  Roof Matl       2925 non-null   object
 15  Exterior 1st    2925 non-null   object
 16  Exterior 2nd    2925 non-null   object
 17  Mas Vnr Type    2925 non-null   object
 18  Exter Qual      2925 non-null   object
 19  Exter Cond      2925 non-null   object
 20  Foundation      2925 non-null   object
 21  Bsmt Qual       2925 non-null   object
 22  Bsmt Cond       2925 non-null   object
 23  Bsmt Exposure   2925 non-null   object
 24  BsmtFin Type 1  2925 non-null   object
 25  BsmtFin Type 2  2925 non-null   object
 26  Heating         2925 non-null   object
 27  Heating QC      2925 non-null   object
 28  Central Air     2925 non-null   object
 29  Electrical      2925 non-null   object
 30  Kitchen Qual    2925 non-null   object
 31  Functional      2925 non-null   object
 32  Fireplace Qu    2925 non-null   object
 33  Garage Type     2925 non-null   object
 34  Garage Finish   2925 non-null   object
 35  Garage Qual     2925 non-null   object
 36  Garage Cond     2925 non-null   object
 37  Paved Drive     2925 non-null   object
 38  Sale Type       2925 non-null   object
 39  Sale Condition  2925 non-null   object
dtypes: object(40)
memory usage: 1001.5+ KB
```

Converting

In [97]:

```
df_objs = pd.get_dummies(df_objs,drop_first=True)
```

In [98]:

```
final_df = pd.concat([df_nums,df_objs],axis=1)
```

In [99]:

```
final_df
```

Out[99]:

	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built	Year Remod/Add	Mas Vnr Area	BsmtFin SF 1	BsmtFin SF 2	Bsmt Unf SF
0	141.000000	31770	6	5	1960	1960	112.0	639.0	0.0	441.0
1	80.000000	11622	5	6	1961	1961	0.0	468.0	144.0	270.0
2	81.000000	14267	6	6	1958	1958	108.0	923.0	0.0	406.0
3	93.000000	11160	7	5	1968	1968	0.0	1065.0	0.0	1045.0
4	74.000000	13830	5	5	1997	1998	0.0	791.0	0.0	137.0
...	...	...	...	...	...	...	...	...	...	...
2925	37.000000	7937	6	6	1984	1984	0.0	819.0	0.0	184.0
2926	75.144444	8885	5	5	1983	1983	0.0	301.0	324.0	239.0
2927	62.000000	10441	5	5	1992	1992	0.0	337.0	0.0	575.0
2928	77.000000	10010	5	5	1974	1975	0.0	1071.0	123.0	195.0
2929	74.000000	9627	7	5	1993	1994	94.0	758.0	0.0	238.0

2925 rows × 274 columns

In [100]:

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

Out[100]:

```
Exter Qual_TA       -0.591459
Kitchen Qual_TA     -0.527461
Fireplace Qu_None   -0.481740
Bsmt Qual_TA        -0.453022
Garage Finish_Unf   -0.422363
                       ...   
Garage Cars          0.648488
Total Bsmt SF        0.660983
Gr Liv Area          0.727279
Overall Qual         0.802637
SalePrice            1.000000
Name: SalePrice, Length: 274, dtype: float64
```

Data is cleaned and Now we create the Machine Learning model

We Use Linear Regression Model

In [108]:

```
df = pd.read_csv("../DATA/AMES_Final_DF.csv")
```

In [109]:

```
df.head()
```

Out[109]:

	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built	Year Remod/Add	Mas Vnr Area	BsmtFin SF 1	BsmtFin SF 2	Bsmt Unf SF	...
0	141.0	31770	6	5	1960	1960	112.0	639.0	0.0	441.0	...
1	80.0	11622	5	6	1961	1961	0.0	468.0	144.0	270.0	...
2	81.0	14267	6	6	1958	1958	108.0	923.0	0.0	406.0	...
3	93.0	11160	7	5	1968	1968	0.0	1065.0	0.0	1045.0	...
4	74.0	13830	5	5	1997	1998	0.0	791.0	0.0	137.0	...

5 rows × 274 columns

In [110]:

```
#We are trying to predict sales price label. Therefore we take sales price as y and rest fe
```

In [111]:

```
X = df.drop('SalePrice',axis=1)
```

In [112]:

```
y = df['SalePrice']
```

In [113]:

```
#We would now be using train_test_split from sklearn
```

In [114]:

```
from sklearn.model_selection import train_test_split
```

In [115]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=101)
```

In [116]:

```
#We would now scale the data
```

In [117]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

In [118]:

```
#We use the Elastic model which is a combinaton of both Lasso and Ridge
```

In [119]:

```
from sklearn.linear_model import ElasticNet
```

In [120]:

```
base_elastic_model = ElasticNet()
```

In [121]:

```
#The Elastic Net model has two main parameters, alpha and the L1 ratio.
#Therefore we create grid of these two to choose most suitable value
```

In [122]:

```
param_grid = {'alpha':[0.1,1,5,10,50,100],
              'l1_ratio':[.1, .5, .7, .9, .95, .99, 1]}
```

In [123]:

```
from sklearn.model_selection import GridSearchCV
```

In [124]:

```
grid_model = GridSearchCV(estimator=base_elastic_model,
                          param_grid=param_grid,
                          scoring='neg_mean_squared_error',
                          cv=5,
                          verbose=1)
```

In [125]:

```
grid_model.fit(scaled_X_train,y_train)
```

```
530: ConvergenceWarning: Objective did not converge. You might want to inc
rease the number of iterations. Duality gap: 139422008252.62378, toleranc
e: 1355206692.5276783
  model = cd_fast.enet_coordinate_descent(
c:\python39\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:
530: ConvergenceWarning: Objective did not converge. You might want to inc
rease the number of iterations. Duality gap: 165405536738.32166, toleranc
e: 1307913805.6588454
  model = cd_fast.enet_coordinate_descent(
c:\python39\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:
530: ConvergenceWarning: Objective did not converge. You might want to inc
rease the number of iterations. Duality gap: 132401459345.90894, toleranc
e: 1415056940.0061066
  model = cd_fast.enet_coordinate_descent(
c:\python39\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:
530: ConvergenceWarning: Objective did not converge. You might want to inc
rease the number of iterations. Duality gap: 198623915721.08862, toleranc
e: 1438198040.0882876
  model = cd_fast.enet_coordinate_descent(
c:\python39\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:
```

In [126]:

```
grid_model.best_params_
```

Out[126]:

```
{'alpha': 100, 'l1_ratio': 1}
```

In [127]:

```
y_pred = grid_model.predict(scaled_X_test)
```

In [128]:

```
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

In [129]:

```
mean_absolute_error(y_test,y_pred)
```

Out[129]:

```
14195.354900562173
```

In [130]:

```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[130]:

```
20558.508566893164
```

In [131]:

```
np.mean(df['SalePrice'])
```

Out[131]:

```
180815.53743589742
```

In [132]:

```
import pickle
```

In [133]:

```
saved_model = pickle.dumps(grid_model)
```

In [134]:

```
#Model is saved
```

In [ ]:

```

```