Problem Statement:

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

• Which variables are important to predict the price of variable?

• How do these variables describe the price of the house?

Business Goal:

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Technical Requirements:

• Data contains 1460 entries each having 81 variables. • Data contains Null values. You need to treat them using the domain knowledge and your own understanding. • Extensive EDA has to be performed to gain relationships of important variable and price. • Data contains numerical as well as categorical variable. You need to handle them accordingly. • You have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters. • You need to find important features which affect the price positively or negatively. • Two datasets are being provided to you (test.csv, train.csv). You will train on train.csv dataset and predict on test.csv file.

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest,f_classif
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor,HistGradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import *

import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df=pd.read_csv("train.csv")
df
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

EDA

In [5]:
```
pd.set_option('display.max_columns',None)
df
```

Out[5]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Feedr	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	NPkVill	Norm	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	Inside	Gtl	IDOTRR	Feedr	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	

1168 rows × 81 columns

In [6]:
```
df.shape
```

Out[6]:
```
(1168, 81)
```

In [7]:
```
df.info
```

Out[7]:
```
<bound method DataFrame.info of        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0     127         120       RL          NaN     4928   Pave   NaN      IR1
1     889          20       RL         95.0    15865   Pave   NaN      IR1
2     793          60       RL         92.0     9920   Pave   NaN      IR1
3     110          20       RL        105.0    11751   Pave   NaN      IR1
4     422          20       RL          NaN    16635   Pave   NaN      IR1
...   ...         ...      ...          ...      ...    ...   ...      ...
1163  289          20       RL          NaN     9819   Pave   NaN      IR1
1164  554          20       RL         67.0     8777   Pave   NaN      Reg
1165  196         160       RL         24.0     2280   Pave   NaN      Reg
1166   31          70  C (all)         50.0     8500   Pave  Pave      Reg
1167  617          60       RL          NaN     7861   Pave   NaN      IR1

     LandContour Utilities LotConfig LandSlope Neighborhood Condition1  \
0            Lvl    AllPub    Inside       Gtl      NPkVill       Norm
1            Lvl    AllPub    Inside       Mod        NAmes       Norm
2            Lvl    AllPub   CulDSac       Gtl      NoRidge       Norm
3            Lvl    AllPub    Inside       Gtl       NWAmes       Norm
4            Lvl    AllPub       FR2       Gtl       NWAmes       Norm
```

In [8]:
```
df.columns[df.isnull().any()]
```

Out[8]:
```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish',
       'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature'],
      dtype='object')
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.isin(['NA','N/A','-',' ','?',' ?']).sum().any()
```

Out[10]:
```
False
```

In [11]:
```
df.drop(columns = ['MiscFeature','PoolQC','Alley'],axis = 1, inplace = True)
df.shape
```

Out[11]:
```
(1168, 78)
```

In [12]:
```
null = ['LotFrontage', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish',
       'GarageQual', 'GarageCond', 'Fence']
```

In [13]:
```
for i in null:
    print('\n',i)
    print(df[i].unique(),'\n ----------------------------------------')
```

```
 LotFrontage
[ nan  95.  92. 105.  58.  88.  70.  80.  50.  44. 129.  59.  55.  64.
  24.  68.  71.  74.  61.  60. 120.  84. 141.  30.  65.  76. 100.  85.
  75. 107. 122.  82.  62.  73.  79.  77.  41.  69.  90.  96.  72.  34.
  78.  63.  40.  98. 160. 108. 128.  51.  81.  99.  66.  37. 174.  87.
  53. 152.  47.  86.  56.  89.  35.  52.  21. 104.  57.  83.  46. 101.
 112. 149.  93.  49.  43. 130.  54.  91.  67.  97. 110. 103. 115.  94.
  48.  36. 313. 109. 144. 121. 102. 116. 182.  32.  42. 168. 118.  38.
 140. 134. 114. 124.  39. 111.  45. 106. 153.]
 ----------------------------------------

 MasVnrType
['None' 'BrkFace' 'Stone' 'BrkCmn' nan]
 ----------------------------------------

 MasVnrArea
[0.000e+00 4.800e+02 1.260e+02 1.800e+02 6.700e+01 2.230e+02 6.600e+01
 8.200e+01 1.740e+02 3.040e+02 1.720e+02 1.660e+02 1.840e+02 3.500e+02
 4.120e+02 1.000e+00 1.890e+02 1.120e+02 5.000e+01 2.200e+02 1.600e+01
 6.300e+02 2.810e+02 2.870e+02 3.400e+02 2.160e+02       nan 1.400e+02
 1.830e+02 3.360e+02 3.960e+02 2.320e+02 3.200e+02 8.500e+01 1.620e+02
 1.540e+02 1.760e+02 1.200e+02 1.080e+02 2.520e+02 1.300e+02 3.510e+02
 5.710e+02 5.300e+01 2.040e+02 9.750e+02 6.530e+02 8.000e+01 4.720e+02
 3.400e+01 1.650e+02 2.370e+02 1.130e+02 9.000e+01 1.600e+02 1.470e+02
 1.360e+02 3.760e+02 8.900e+01 4.150e+02 2.000e+02 5.060e+02 2.860e+02
 4.500e+01 2.450e+02 2.470e+02 2.400e+02 7.000e+01 9.800e+01 6.040e+02
 8.600e+02 2.700e+01 1.530e+02 2.120e+02 2.060e+02 7.600e+02 2.960e+02
 5.280e+02 2.990e+02 3.380e+02 1.230e+02 2.330e+02 6.800e+01 3.910e+02
 2.100e+02 7.500e+01 9.600e+01 3.800e+02 3.620e+02 1.560e+02 3.990e+02
 7.400e+01 1.860e+02 4.100e+01 6.500e+01 1.440e+02 8.400e+01 2.080e+02
 2.600e+02 4.810e+02 5.670e+02 1.940e+02 1.250e+02 1.800e+01 2.050e+02
 3.090e+02 6.510e+02 7.200e+01 1.450e+02 7.660e+02 7.880e+02 7.620e+02
 2.720e+02 2.560e+02 4.400e+01 3.000e+02 1.680e+02 2.880e+02 4.660e+02
 2.800e+01 3.660e+02 2.920e+02 9.900e+01 2.240e+02 4.200e+02 1.040e+02
 4.910e+02 2.890e+02 2.680e+02 6.400e+02 6.000e+02 3.020e+02 3.600e+02
 4.240e+02 1.780e+02 1.960e+02 5.000e+02 1.047e+03 2.590e+02 2.540e+02
 1.020e+02 1.370e+02 1.640e+02 4.320e+02 2.980e+02 1.280e+02 4.250e+02
 5.130e+02 2.700e+02 8.100e+01 1.700e+02 5.730e+02 3.100e+02 2.030e+02
 1.820e+02 7.050e+02 1.380e+02 6.300e+01 6.600e+02 6.500e+02 4.420e+02
 2.430e+02 3.000e+01 2.260e+02 1.060e+02 2.200e+01 4.000e+01 3.810e+02
 3.440e+02 2.150e+02 2.380e+02 2.460e+02 1.010e+02 4.280e+02 7.960e+02
 3.240e+02 1.000e+02 8.940e+02 1.160e+02 1.220e+02 3.120e+02 5.600e+01
 4.480e+02 3.330e+02 2.660e+02 2.750e+02 5.760e+02 1.430e+02 2.360e+02
```

```
 6.400e+01 1.400e+01 8.800e+01 3.480e+02 4.800e+01 5.640e+02 6.730e+02
 2.800e+02 2.580e+02 1.170e+03 2.340e+02 1.350e+02 1.115e+03 4.380e+02
 2.900e+02 1.050e+02 2.500e+02 1.129e+03 4.260e+02 5.100e+01 4.200e+01
 7.680e+02 9.200e+01 3.420e+02 3.880e+02 1.670e+02 2.950e+02 1.100e+02
 7.720e+02 5.100e+02 9.220e+02 2.440e+02 3.060e+02 4.600e+01 4.560e+02
 2.020e+02 2.780e+02 4.510e+02 1.690e+02 1.570e+02 4.520e+02 4.430e+02
 1.140e+02 2.630e+02 1.378e+03 1.610e+02 1.580e+02 8.700e+02 1.320e+02
 3.150e+02 1.270e+02 4.590e+02 7.310e+02 7.600e+01 2.280e+02 2.250e+02
 4.100e+02 1.920e+02 8.160e+02 4.230e+02 7.480e+02 4.790e+02 1.510e+02
 5.400e+01 1.100e+01 8.600e+01 1.490e+02 5.940e+02 5.700e+01 3.700e+02
 3.350e+02 1.170e+02 1.150e+02 1.500e+02 4.350e+02 2.610e+02 6.000e+01
 3.050e+02 1.630e+02 1.090e+02 2.620e+02 1.600e+03 1.480e+02 2.850e+02
 1.710e+02 9.400e+01 1.190e+02 2.840e+02 9.500e+01 3.600e+01 6.160e+02
 3.650e+02 5.540e+02 4.640e+02 3.100e+01]
 ----------------------------------------

 BsmtQual
['Gd' 'TA' 'Ex' nan 'Fa']
 ----------------------------------------

 BsmtCond
['TA' 'Gd' 'Fa' nan 'Po']
 ----------------------------------------

 BsmtExposure
['No' 'Gd' 'Av' 'Mn' nan]
 ----------------------------------------

 BsmtFinType1
['ALQ' 'GLQ' 'BLQ' 'Unf' 'Rec' 'LwQ' nan]
 ----------------------------------------

 BsmtFinType2
['Unf' 'Rec' 'BLQ' 'GLQ' nan 'ALQ' 'LwQ']
 ----------------------------------------

 FireplaceQu
['TA' 'Gd' nan 'Fa' 'Ex' 'Po']
 ----------------------------------------

 GarageType
['Attchd' 'BuiltIn' 'Detchd' 'Basment' nan '2Types' 'CarPort']
 ----------------------------------------
```

```
 GarageYrBlt
[1977. 1970. 1997. 2006. 1957. 1965. 1947. 1937. 2003. 1974. 1955. 1923.
 2002. 2007. 1987. 2001. 1988. 1950. 1961. 1953. 2010. 1922. 1939. 2005.
 1991. 1979. 1975. 1976. 1978. 1960. 1956. 2004. 1982. 2000. 1948.   nan
 1964. 1920. 1930. 1968. 1946. 1992. 1936. 1967. 1989. 1959. 1966. 1916.
 1941. 1998. 1962. 1926. 1925. 1983. 1999. 1969. 1985. 1993. 2008. 1971.
 1980. 1945. 1995. 1981. 1994. 1949. 1996. 1921. 1963. 1938. 1958. 1935.
 1940. 1990. 1910. 1954. 1927. 2009. 1986. 1929. 1984. 1973. 1924. 1942.
 1900. 1931. 1951. 1934. 1972. 1932. 1928. 1918. 1908. 1933. 1906. 1914.
 1952. 1915.]
 ----------------------------------------

 GarageFinish
['RFn' 'Unf' 'Fin' nan]
 ----------------------------------------

 GarageQual
['TA' 'Fa' nan 'Gd' 'Ex' 'Po']
 ----------------------------------------

 GarageCond
['TA' 'Fa' 'Gd' nan 'Po' 'Ex']
 ----------------------------------------

 Fence
[nan 'MnPrv' 'GdPrv' 'GdWo' 'MnWw']
 ----------------------------------------
```

In [14]:
```
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].mean())
```

In [15]:
```
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].median())
df['GarageYrBlt'] = df['GarageYrBlt'].fillna(df['GarageYrBlt'].median())
```

In [16]:
```
def fill(x):
    df[x] = df[x].fillna(df[x].mode()[0])
```

In [17]:
```
n = ['MasVnrType','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','FireplaceQu','GarageType',
'GarageFinish','GarageQual','GarageCond','Fence']
```

In [18]:
```
for i in n:
    fill(i)
```

In [19]:
```
df.columns[df.isnull().any()]
```

Out[19]:
```
Index([], dtype='object')
```

Now, No null values present in our dataset.

ENCODING

In [20]:
```
ob = []
for col in df:
    if df[col].dtype == 'object':
        ob.append(col)
```

In [21]:
```
for i in ob:
    print('\n',i)
    print(df[i].unique(),'\n ----------------------------------------')
```

```
 MSZoning
['RL' 'RM' 'FV' 'RH' 'C (all)']
 ----------------------------------------

 Street
['Pave' 'Grvl']
 ----------------------------------------

 LotShape
['IR1' 'Reg' 'IR2' 'IR3']
 ----------------------------------------

 LandContour
['Lvl' 'Bnk' 'HLS' 'Low']
 ----------------------------------------

 Utilities
['AllPub']
```

In [22]:
```
df.drop('Utilities',axis = 1, inplace = True )
```

In [23]:
```
ob.remove('Utilities')
```

In [24]:
```
lb = LabelEncoder()
for i in ob:
    df[i] = lb.fit_transform(df[i])
```

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 77 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1168 non-null   int64  
 1   MSSubClass     1168 non-null   int64  
 2   MSZoning       1168 non-null   int32  
 3   LotFrontage    1168 non-null   float64
 4   LotArea        1168 non-null   int64  
 5   Street         1168 non-null   int32  
 6   LotShape       1168 non-null   int32  
 7   LandContour    1168 non-null   int32  
 8   LotConfig      1168 non-null   int32  
 9   LandSlope      1168 non-null   int32  
 10  Neighborhood   1168 non-null   int32  
 11  Condition1     1168 non-null   int32  
 12  Condition2     1168 non-null   int32  
 13  BldgType       1168 non-null   int32  
 14  HouseStyle     1168 non-null   int32  
 15  OverallQual    1168 non-null   int64  
 16  OverallCond    1168 non-null   int64  
 17  YearBuilt      1168 non-null   int64  
 18  YearRemodAdd   1168 non-null   int64  
 19  RoofStyle      1168 non-null   int32  
 20  RoofMatl       1168 non-null   int32  
 21  Exterior1st    1168 non-null   int32  
 22  Exterior2nd    1168 non-null   int32  
 23  MasVnrType     1168 non-null   int32  
 24  MasVnrArea     1168 non-null   float64
 25  ExterQual      1168 non-null   int32  
 26  ExterCond      1168 non-null   int32  
 27  Foundation     1168 non-null   int32  
 28  BsmtQual       1168 non-null   int32  
 29  BsmtCond       1168 non-null   int32  
 30  BsmtExposure   1168 non-null   int32  
 31  BsmtFinType1   1168 non-null   int32  
 32  BsmtFinSF1     1168 non-null   int64  
 33  BsmtFinType2   1168 non-null   int32  
 34  BsmtFinSF2     1168 non-null   int64  
 35  BsmtUnfSF      1168 non-null   int64  
 36  TotalBsmtSF    1168 non-null   int64  
 37  Heating        1168 non-null   int32  
 38  HeatingQC      1168 non-null   int32  
```

```
 39  CentralAir     1168 non-null   int32
 40  Electrical     1168 non-null   int32
 41  1stFlrSF       1168 non-null   int64
 42  2ndFlrSF       1168 non-null   int64
 43  LowQualFinSF   1168 non-null   int64
 44  GrLivArea      1168 non-null   int64
 45  BsmtFullBath   1168 non-null   int64
 46  BsmtHalfBath   1168 non-null   int64
 47  FullBath       1168 non-null   int64
 48  HalfBath       1168 non-null   int64
 49  BedroomAbvGr   1168 non-null   int64
 50  KitchenAbvGr   1168 non-null   int64
 51  KitchenQual    1168 non-null   int32
 52  TotRmsAbvGrd   1168 non-null   int64
 53  Functional     1168 non-null   int32
 54  Fireplaces     1168 non-null   int64
 55  FireplaceQu    1168 non-null   int32
 56  GarageType     1168 non-null   int32
 57  GarageYrBlt    1168 non-null   float64
 58  GarageFinish   1168 non-null   int32
 59  GarageCars     1168 non-null   int64
 60  GarageArea     1168 non-null   int64
 61  GarageQual     1168 non-null   int32
 62  GarageCond     1168 non-null   int32
 63  PavedDrive     1168 non-null   int32
 64  WoodDeckSF     1168 non-null   int64
 65  OpenPorchSF    1168 non-null   int64
 66  EnclosedPorch  1168 non-null   int64
 67  3SsnPorch      1168 non-null   int64
 68  ScreenPorch    1168 non-null   int64
 69  PoolArea       1168 non-null   int64
 70  Fence          1168 non-null   int32
 71  MiscVal        1168 non-null   int64
 72  MoSold         1168 non-null   int64
 73  YrSold         1168 non-null   int64
 74  SaleType       1168 non-null   int32
 75  SaleCondition  1168 non-null   int32
 76  SalePrice      1168 non-null   int64
dtypes: float64(3), int32(39), int64(35)
memory usage: 524.8 KB
```

Features Selection

In [26]:
```
x = df.drop('SalePrice',axis=1)
y = df.SalePrice
```

In [27]:
```
best_feature = SelectKBest(score_func = f_classif, k =60)
fit = best_feature.fit(x,y)
```

In [28]:
```
score = pd.DataFrame(fit.scores_)
column = pd.DataFrame(x.columns)
```

In [43]:
```
feature_score = pd.concat([column,score], axis = 1)
feature_score.columns = ['Feature Name','Scores']
feature_score.nlargest(40,'Scores')
```

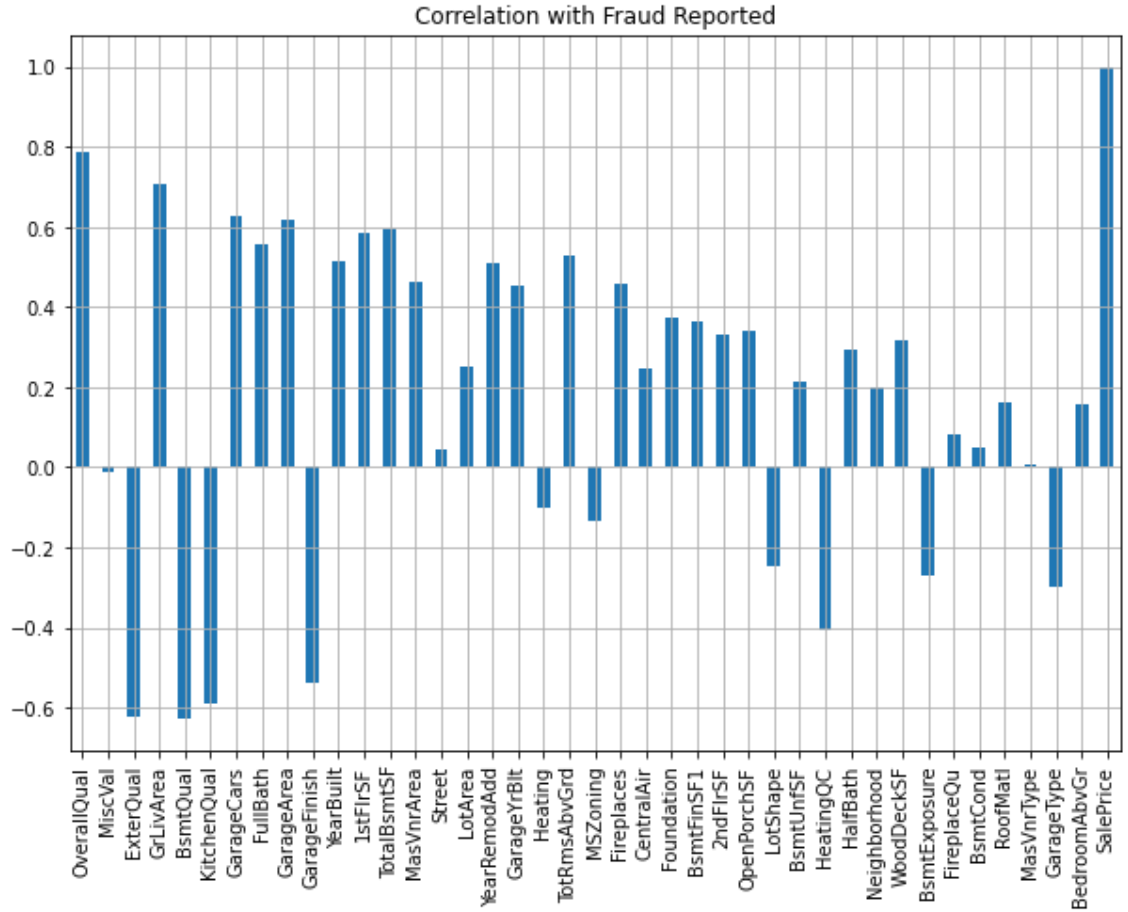
Out[43]:

	Feature Name	Scores
15	OverallQual	5.303071
71	MiscVal	3.564855
25	ExterQual	3.514221
44	GrLivArea	2.977506
28	BsmtQual	2.876879
51	KitchenQual	2.617125
59	GarageCars	2.578547
47	FullBath	2.435854
60	GarageArea	2.316328
58	GarageFinish	2.187163
17	YearBuilt	2.133300
41	1stFlrSF	2.036680
36	TotalBsmtSF	1.867714
24	MasVnrArea	1.852976
5	Street	1.835751
4	LotArea	1.826320
18	YearRemodAdd	1.813783
57	GarageYrBlt	1.712783
37	Heating	1.707885
52	TotRmsAbvGrd	1.656866
2	MSZoning	1.640044
54	Fireplaces	1.591973
39	CentralAir	1.557680
27	Foundation	1.528516
32	BsmtFinSF1	1.482500
42	2ndFlrSF	1.476305
65	OpenPorchSF	1.460290
6	LotShape	1.407526

	Feature Name	Scores
35	BsmtUnfSF	1.401635
38	HeatingQC	1.358939
48	HalfBath	1.337597
10	Neighborhood	1.281079
64	WoodDeckSF	1.254936
30	BsmtExposure	1.214154
55	FireplaceQu	1.203591
29	BsmtCond	1.190161
20	RoofMatl	1.179426
23	MasVnrType	1.171156
56	GarageType	1.169772
49	BedroomAbvGr	1.146460

In [45]:
```
top = df[['OverallQual','MiscVal','ExterQual','GrLivArea','BsmtQual','KitchenQual','GarageCars','FullBath',
          'GarageArea','GarageFinish','YearBuilt','1stFlrSF','TotalBsmtSF','MasVnrArea','Street','LotArea',
          'YearRemodAdd','GarageYrBlt','Heating','TotRmsAbvGrd','MSZoning','Fireplaces','CentralAir',
          'Foundation','BsmtFinSF1','2ndFlrSF','OpenPorchSF','LotShape','BsmtUnfSF','HeatingQC','HalfBath',
          'Neighborhood','WoodDeckSF','BsmtExposure','FireplaceQu','BsmtCond','RoofMatl','MasVnrType',
          'GarageType','BedroomAbvGr','SalePrice']]
```

In [46]:
```
top.corrwith(y).plot(kind = 'bar',grid = True, figsize = (10,7), title = 'Correlation with Fraud Reported')
plt.show()
```

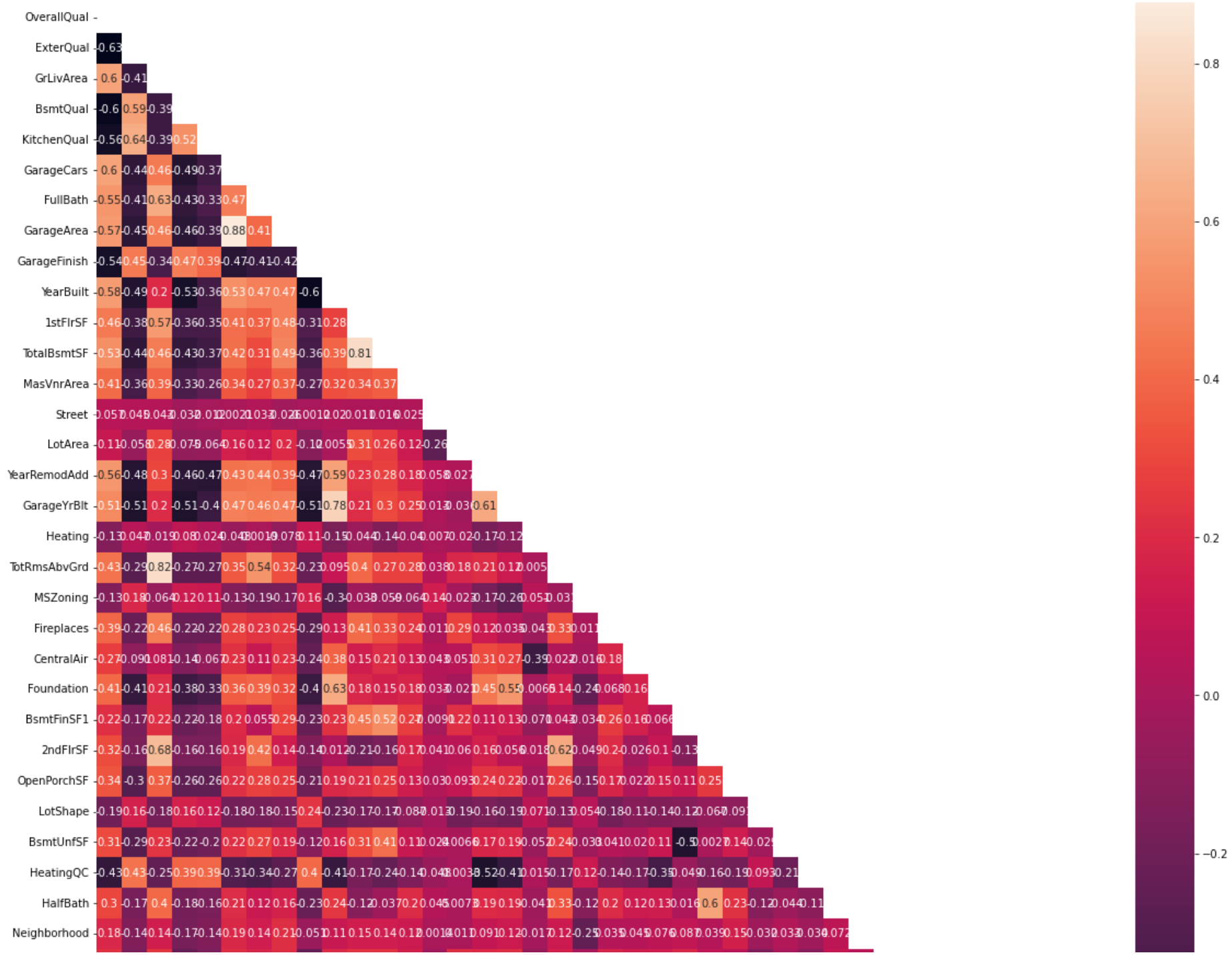


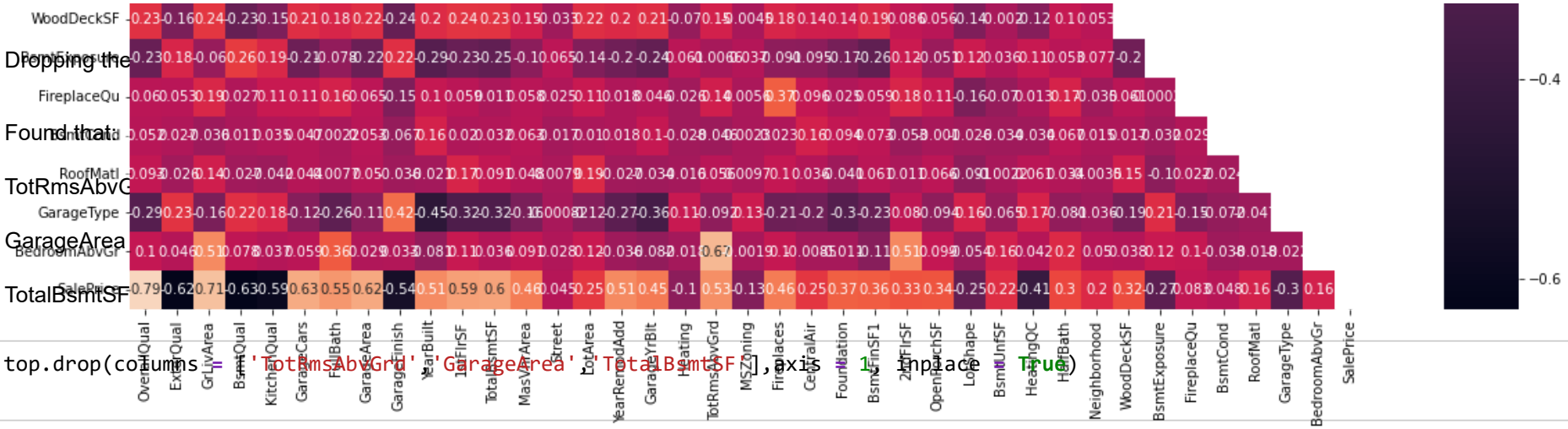
In [47]:
```
top.drop(columns = ['MasVnrType','MiscVal'],axis = 1, inplace = True)
```

In [48]:
```
corr = top.corr()
mask = np.triu(np.ones_like(corr))
```

In [49]:
```
plt.figure(figsize = (20,20))
sns.heatmap(corr,annot = True,fmt = '.2g',mask = mask)
```

Out[49]:
```
<AxesSubplot:>
```


OverallQual
ExterQual
GrLivArea
BsmtQual
KitchenQual
GarageCars
FullBath
GarageArea
GarageFinish
YearBuilt
1stFlrSF
TotalBsmtSF
MasVnrArea
Street
LotArea
YearRemodAdd
GarageYrBlt
Heating
TotRmsAbvGrd
MSZoning
Fireplaces
CentralAir
Foundation
BsmtFinSF1
2ndFlrSF
OpenPorchSF
LotShape
BsmtUnfSF
HeatingQC
HalfBath
Neighborhood
0.8
0.6
0.4
0.2
0.0
−0.2



Dropping the

Found that

TotRmsAbvG

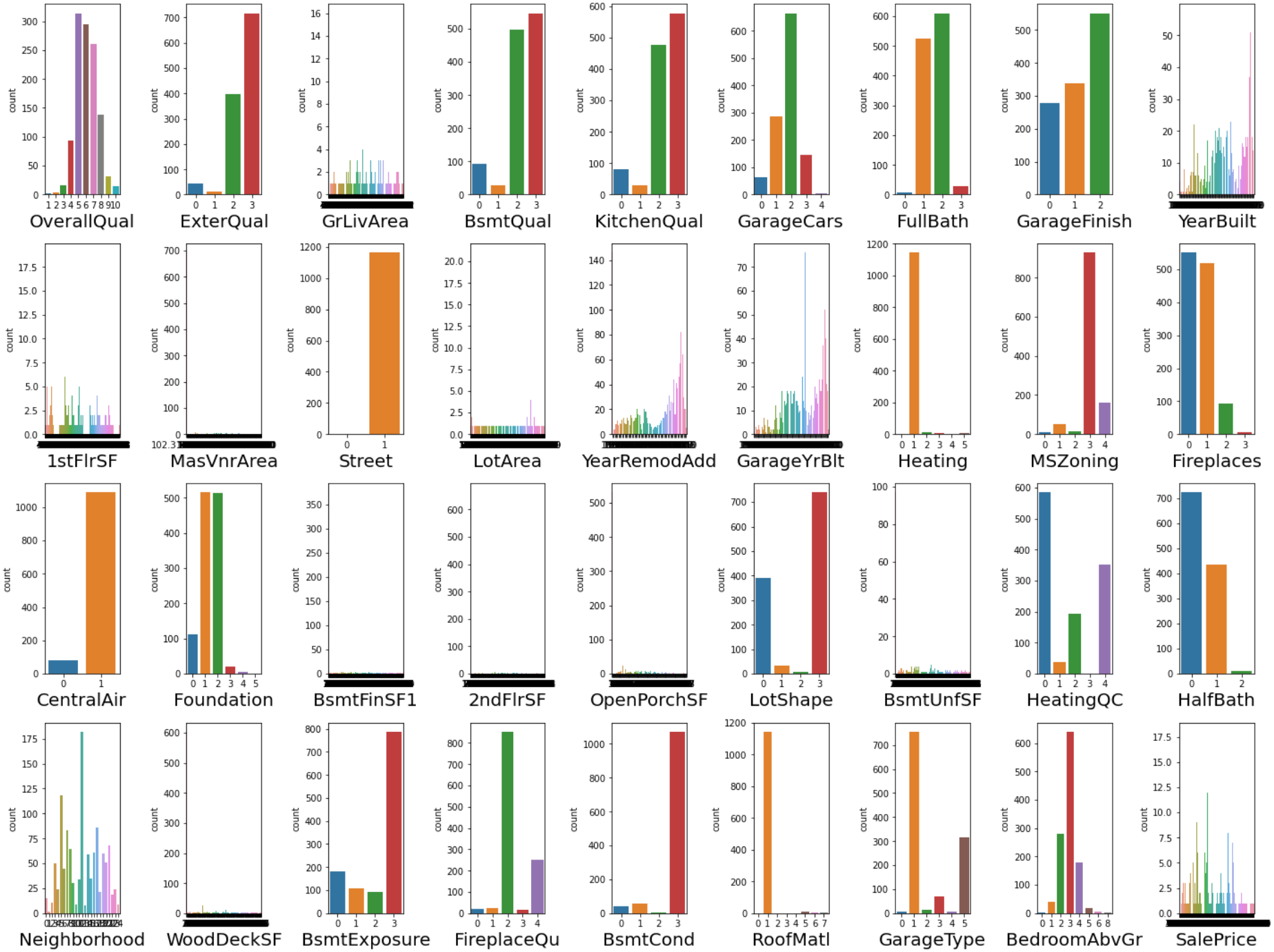
GarageArea

TotalBsmtSF

In [50]:
```
top.drop(columns = 'TotRmsAbvGrd''GarageArea''TotalBsmtSF'],axis = 1, inplace = True)
```

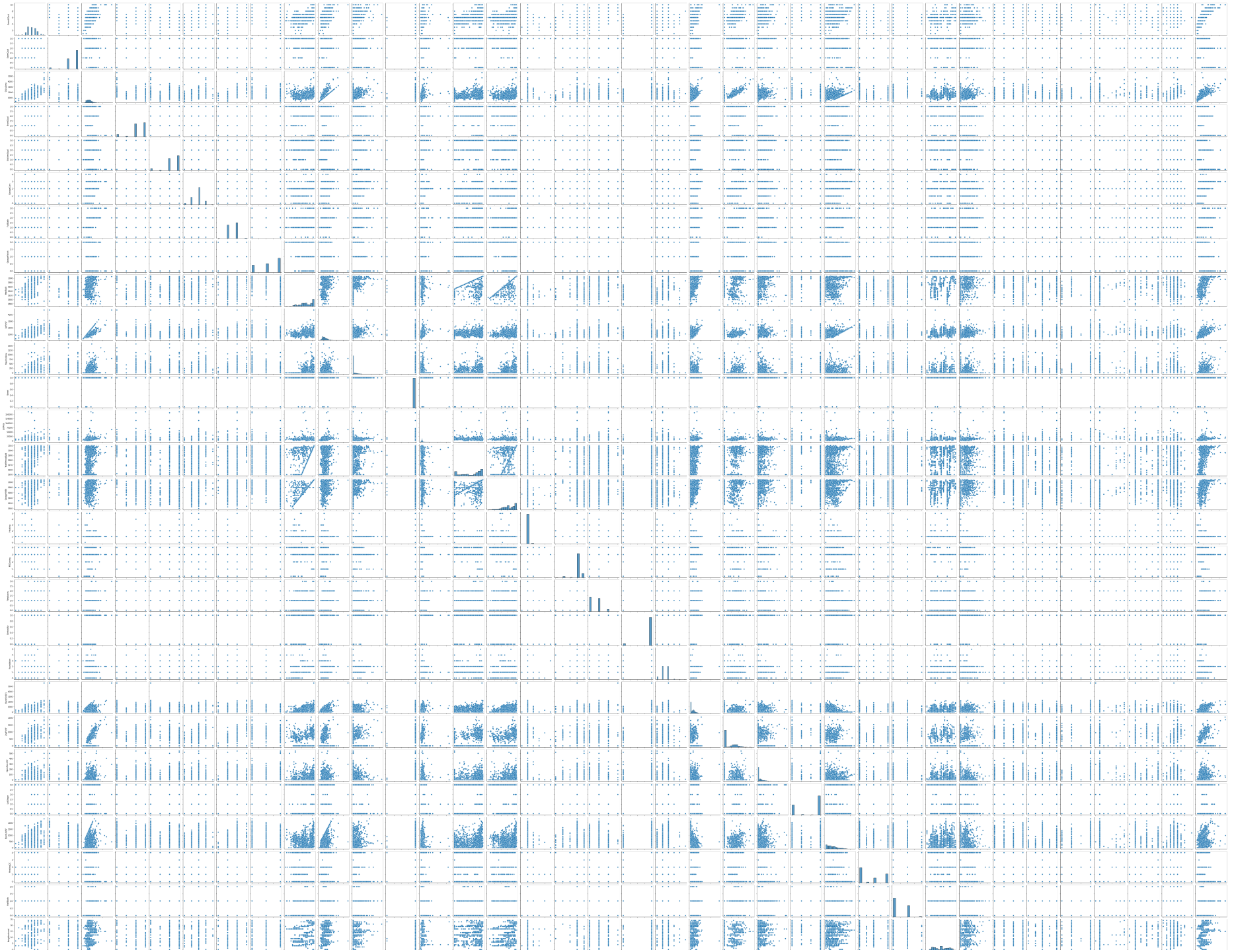
Visualisation

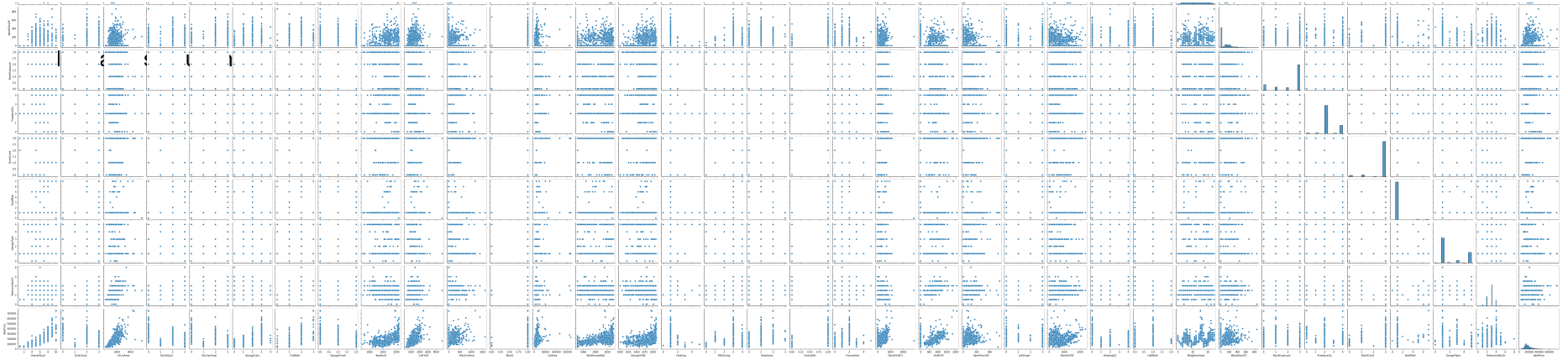
In [51]:
```
plt.figure(figsize = (20,15))
plotnumber = 1

for column in top:
    if plotnumber <= 36:
        ax = plt.subplot(4,9,plotnumber)
        sns.countplot(top[column])
        plt.xlabel(column,fontsize = 20)

    plotnumber+=1
plt.tight_layout()
```


OverallQual
ExterQual
GrLivArea
BsmtQual
KitchenQual
GarageCars
FullBath
GarageFinish
YearBuilt
1stFlrSF
MasVnrArea
Street
LotArea
YearRemodAdd
GarageYrBlt
Heating
MSZoning
Fireplaces
CentralAir
Foundation
BsmtFinSF1
2ndFlrSF
OpenPorchSF
LotShape
BsmtUnfSF
HeatingQC
HalfBath
Neighborhood
WoodDeckSF
BsmtExposure
FireplaceQu
BsmtCond
RoofMatl
GarageType
BedroomAbvGr
SalePrice

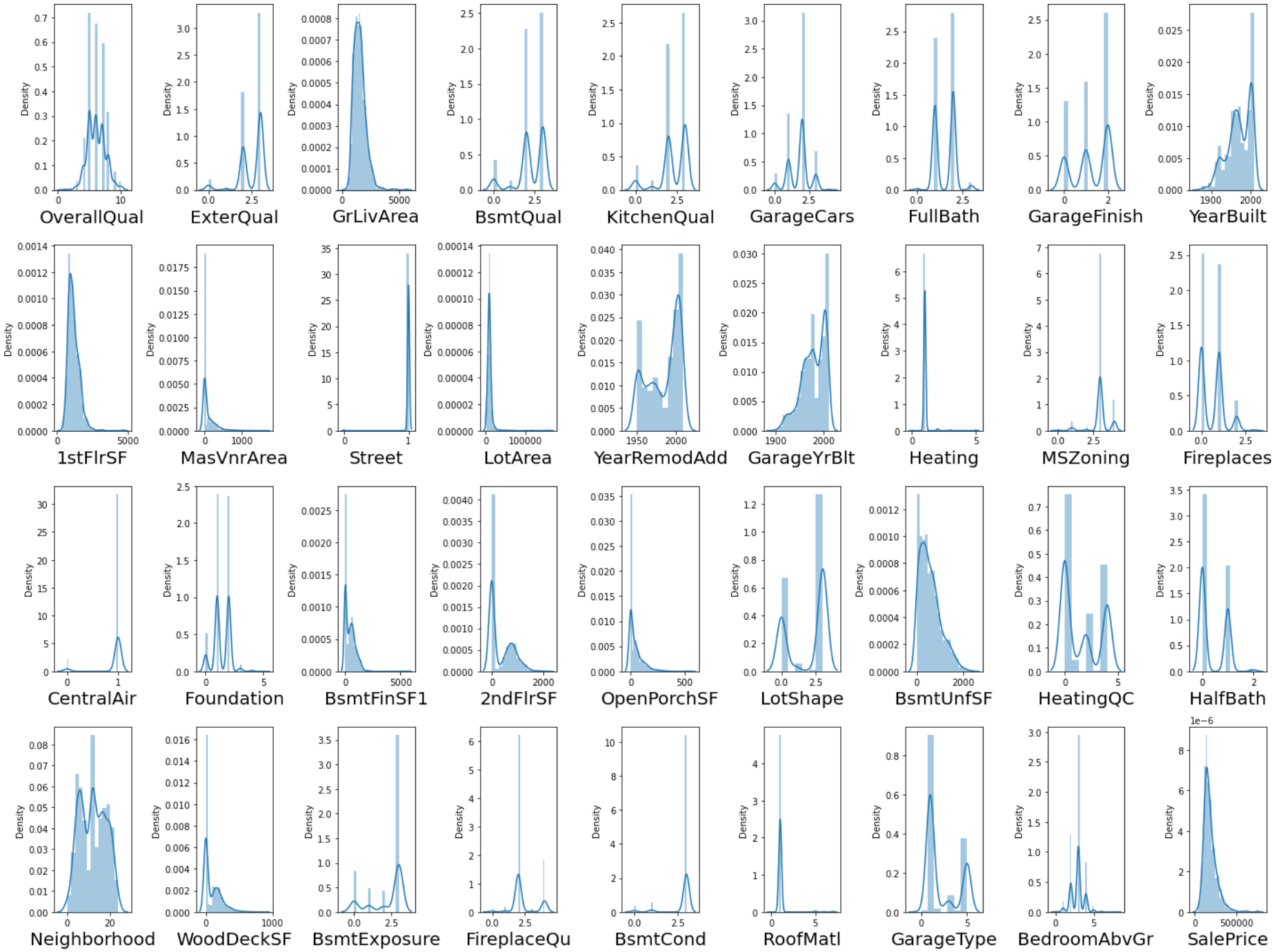
In [52]:
```
sns.pairplot(top)
```

Out[52]:
```
<seaborn.axisgrid.PairGrid at 0x1bf59efb2b0>
```



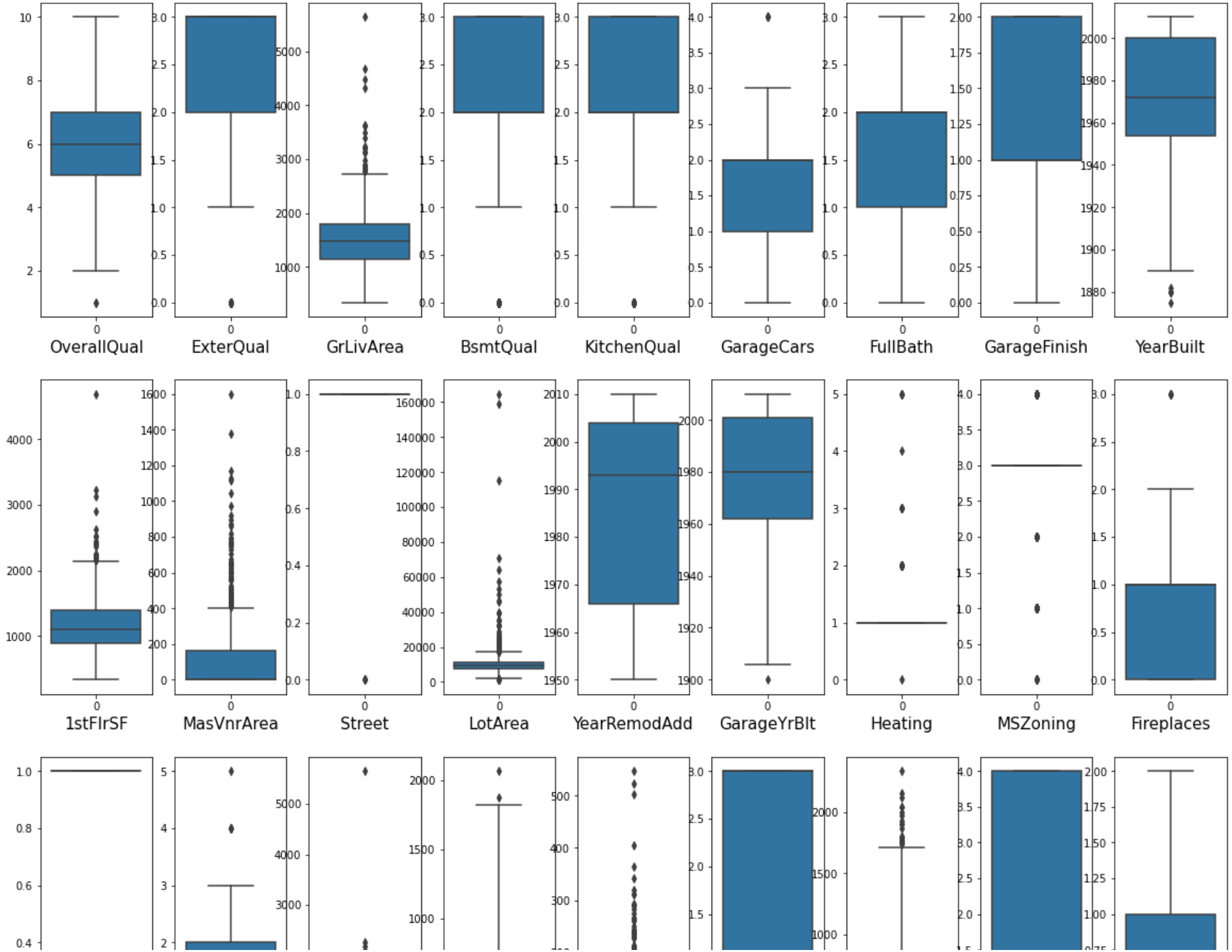


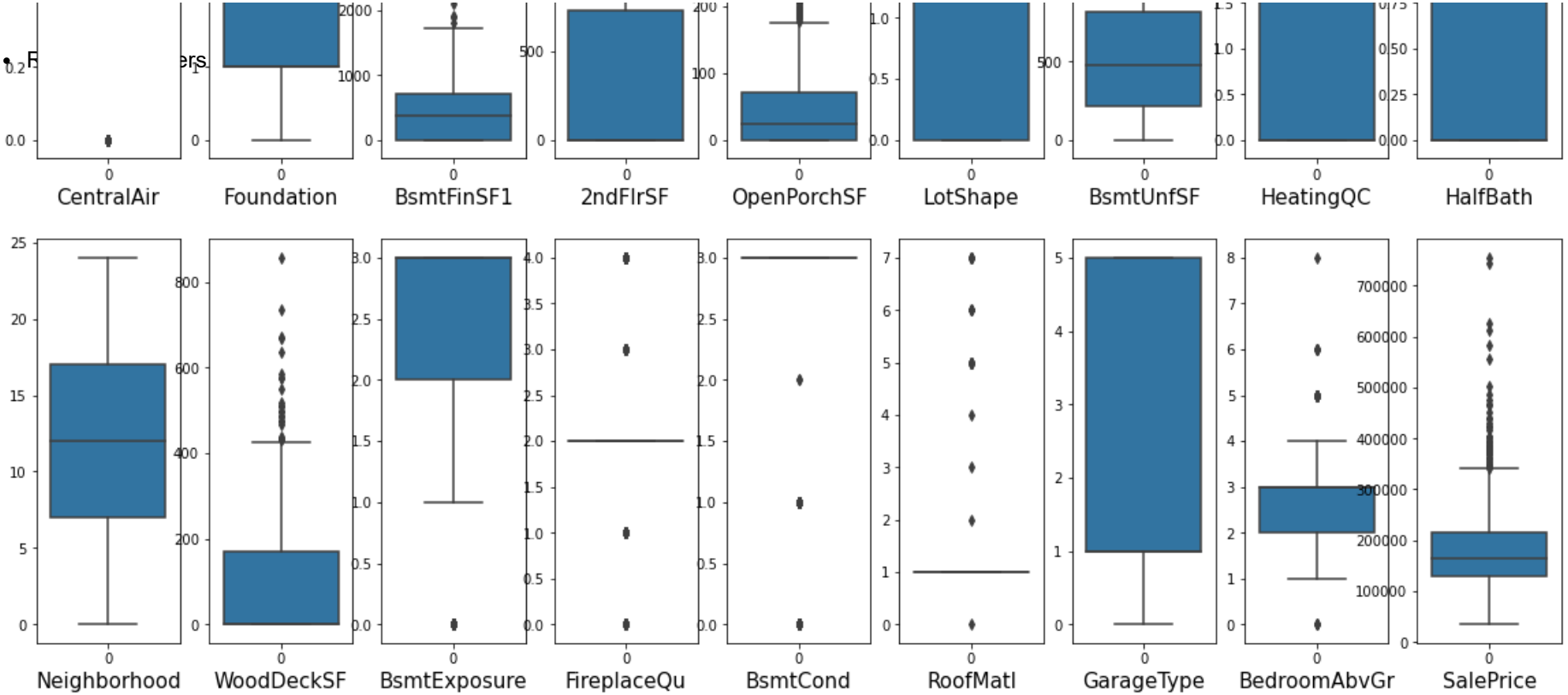
In [53]:
```
plt.figure(figsize = (20,15))
plotnumber = 1

for column in top:
    if plotnumber <= 36:
        ax = plt.subplot(4,9,plotnumber)
        sns.distplot(top[column])
        plt.xlabel(column,fontsize = 20)

    plotnumber+=1
plt.tight_layout()
```


OverallQual
ExterQual
GrLivArea
BsmtQual
KitchenQual
GarageCars
FullBath
GarageFinish
YearBuilt
1stFlrSF
MasVnrArea
Street
LotArea
YearRemodAdd
GarageYrBlt
Heating
MSZoning
Fireplaces
CentralAir
Foundation
BsmtFinSF1
2ndFlrSF
OpenPorchSF
LotShape
BsmtUnfSF
HeatingQC
HalfBath
Neighborhood
WoodDeckSF
BsmtExposure
FireplaceQu
BsmtCond
RoofMatl
GarageType
BedroomAbvGr
SalePrice
Density

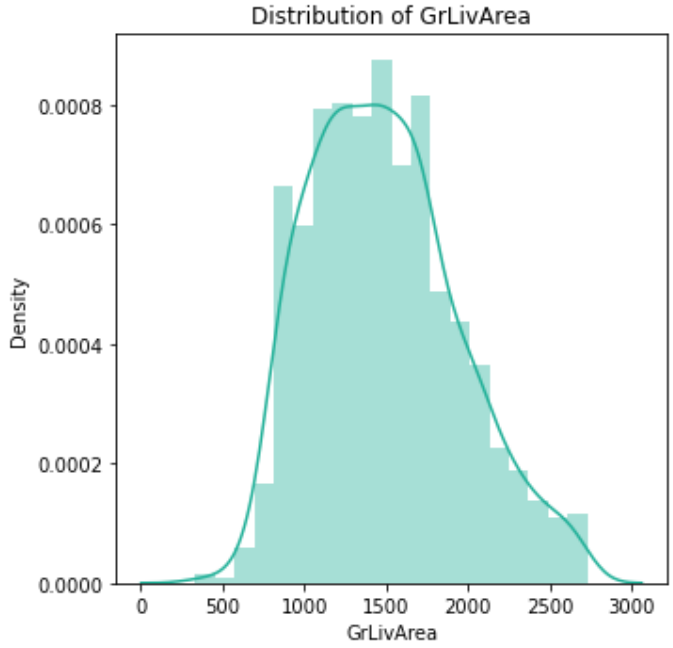
- Outliers.

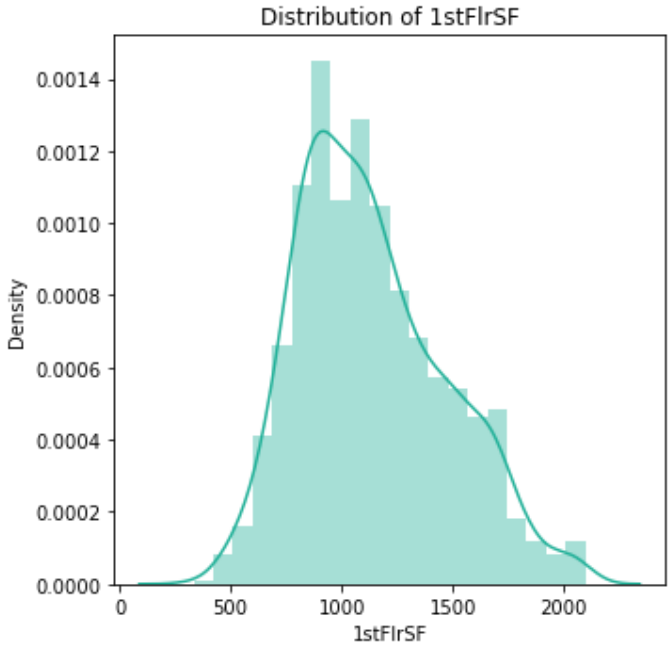
In [54]:
```
plt.figure(figsize = (20,25))
graph = 1
for column in top:
    if graph <= 36:
        plt.subplot(4,9,graph)
        ax = sns.boxplot(data = top[column])
        plt.xlabel(column, fontsize = 15)
    graph += 1
plt.show()
```

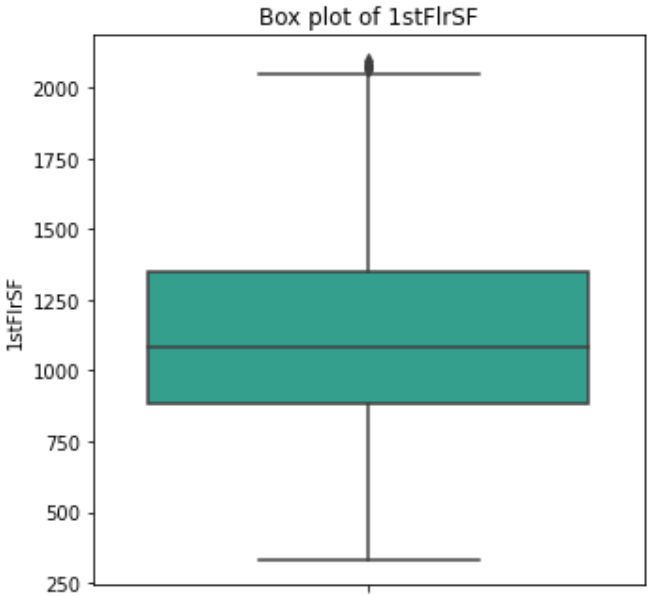

OverallQual
ExterQual
GrLivArea
BsmtQual
KitchenQual
GarageCars
FullBath
GarageFinish
YearBuilt
1stFlrSF
MasVnrArea
Street
LotArea
YearRemodAdd
GarageYrBlt
Heating
MSZoning
Fireplaces

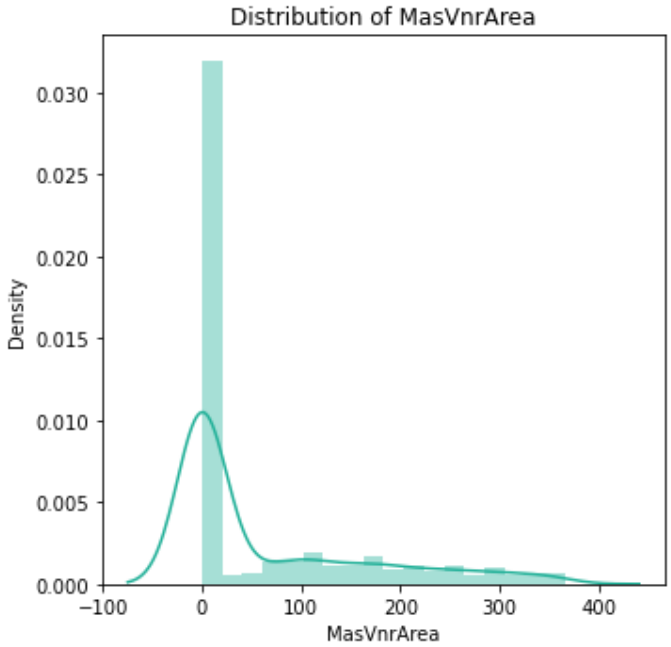

CentralAir
Foundation
BsmtFinSF1
2ndFlrSF
OpenPorchSF
LotShape
BsmtUnfSF
HeatingQC
HalfBath
Neighborhood
WoodDeckSF
BsmtExposure
FireplaceQu
BsmtCond
RoofMatl
GarageType
BedroomAbvGr
SalePrice

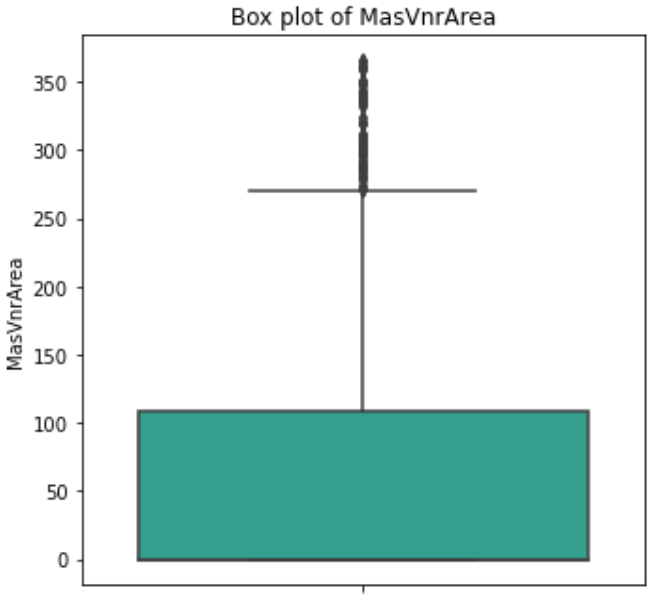
In [55]:
```
def remove_outliers(x):
    global top
    Q1= top[x].quantile(0.25)
    Q3 = top[x].quantile(0.75)
    IQR = Q3 - Q1
    upper_limit = Q3 + 1.5 * IQR

    index = np.where(top[x] > upper_limit)
    top = top.drop(top.index[index])
    top.reset_index()

    fig, axes = plt.subplots(1,2,figsize=(10,5))
    sns.distplot(top[x],color='#22b199',ax=axes[0])
    sns.boxplot(y=top[x],color='#22b199',ax=axes[1])
    axes[0].set_title('Distribution of '+ x )
    axes[1].set_title('Box plot of '+ x)
    plt.tight_layout()
    plt.show()

for i in ['GrLivArea','1stFlrSF','MasVnrArea','LotArea','BsmtFinSF1','OpenPorchSF','BsmtUnfSF','WoodDeckSF']:
    remove_outliers(i)
print(top.shape)
```

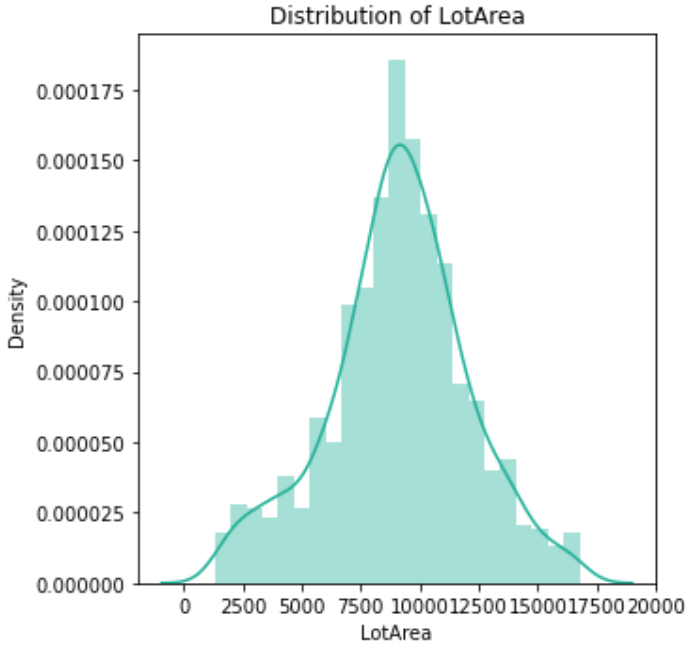


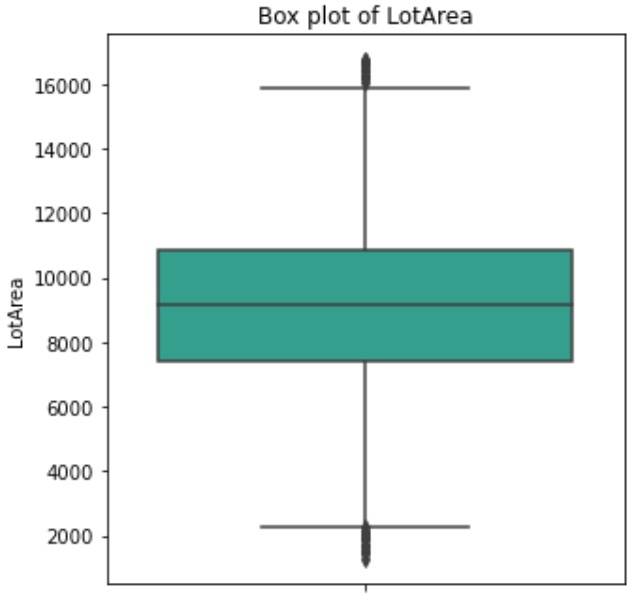

Distribution of 1stFlrSF
Density
1stFlrSF
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0
500
1000
1500
2000

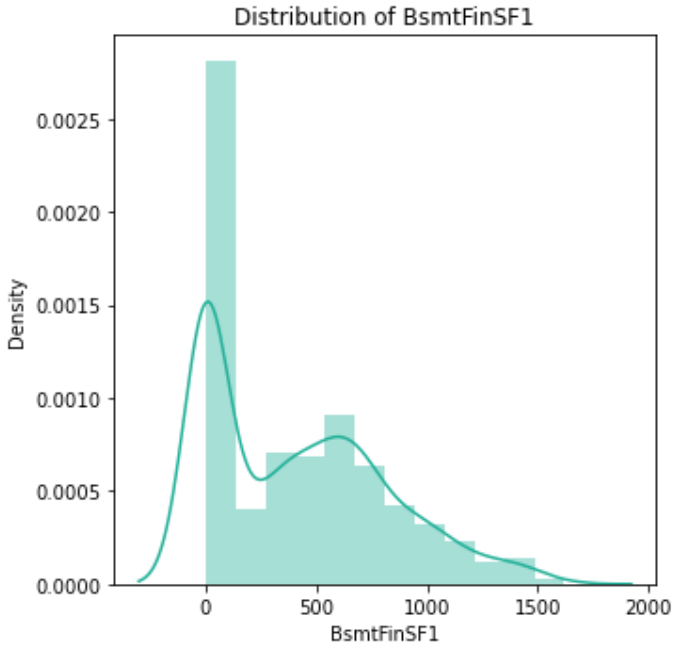

Box plot of 1stFlrSF
1stFlrSF
250
500
750
1000
1250
1500
1750
2000

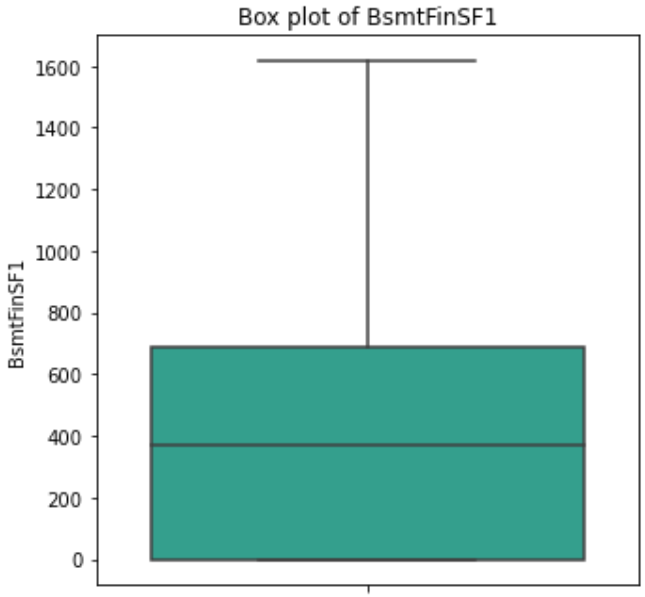

Distribution of MasVnrArea
Density
MasVnrArea
0.000
0.005
0.010
0.015
0.020
0.025
0.030
−100
0
100
200
300
400

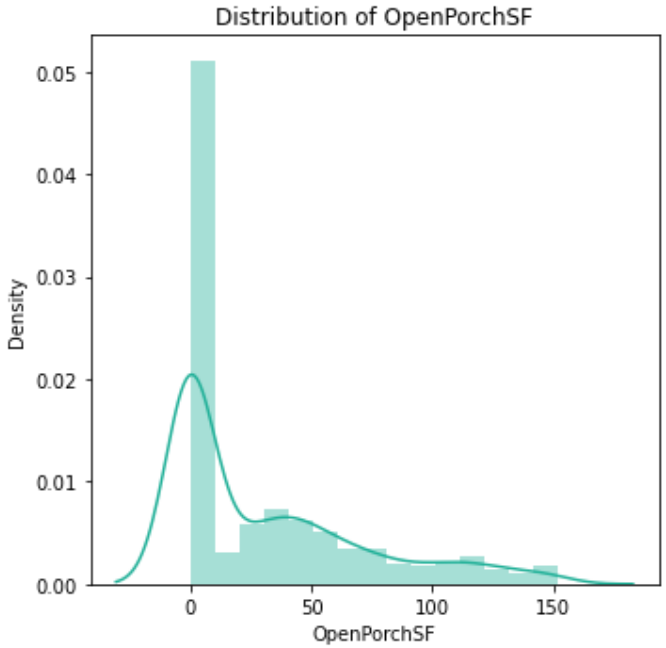

Box plot of MasVnrArea
MasVnrArea
0
50
100
150
200
250
300
350

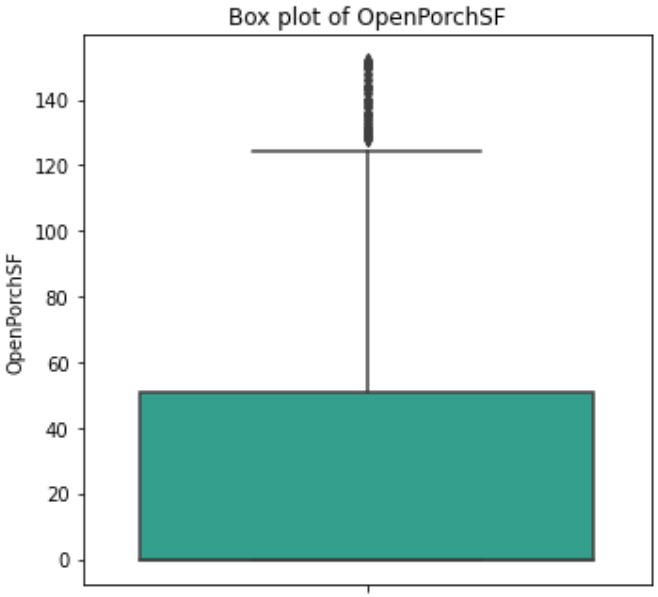

Distribution of LotArea
Density
LotArea
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
0
2500
5000
7500
10000
12500
15000
17500
20000

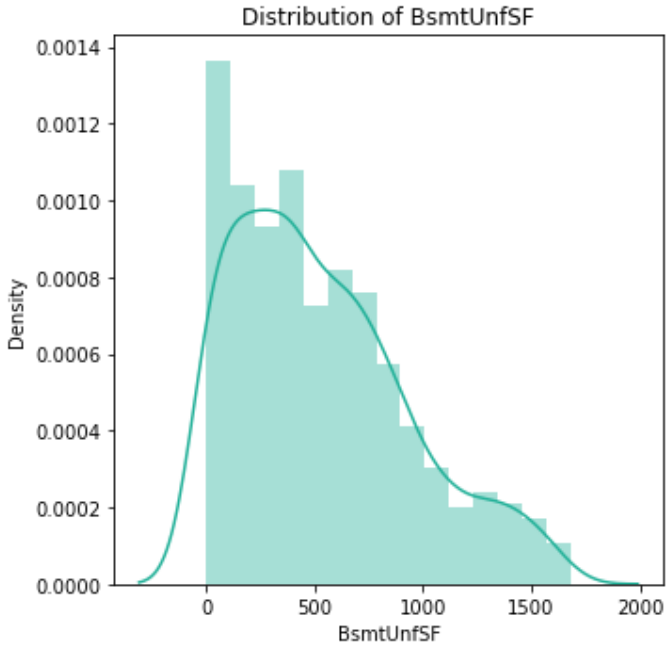

Box plot of LotArea
LotArea
16000
14000
12000
10000
8000
6000
4000
2000

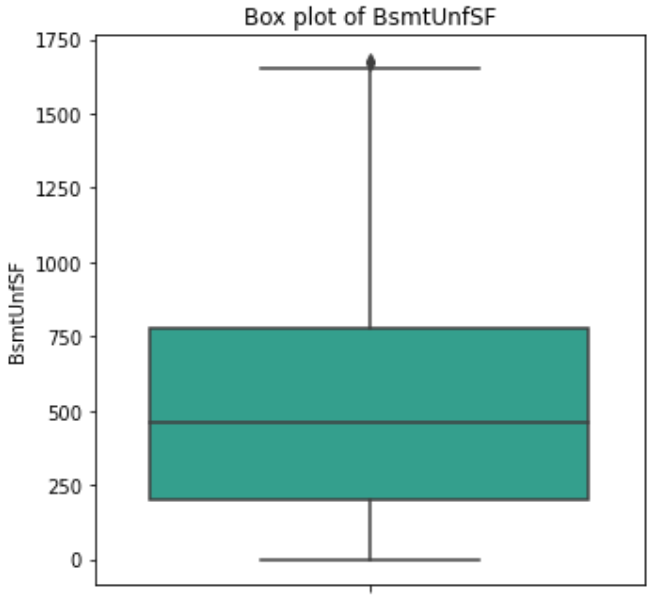

Distribution of BsmtFinSF1
Density
BsmtFinSF1
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000

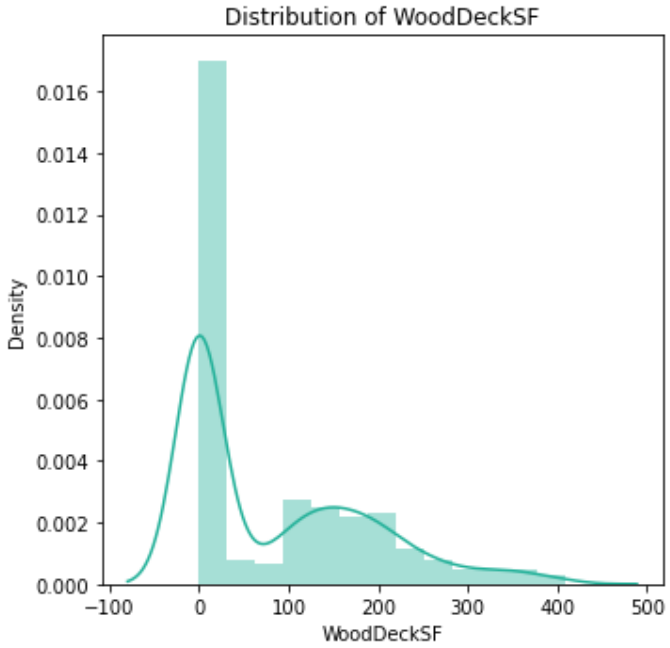

Box plot of BsmtFinSF1
BsmtFinSF1
1600
1400
1200
1000
800
600
400
200
0

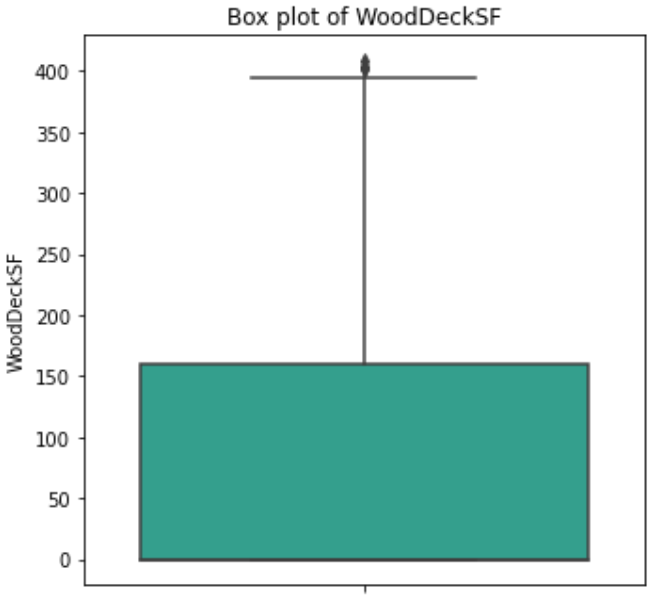

Distribution of OpenPorchSF
Density
OpenPorchSF
0.00
0.01
0.02
0.03
0.04
0.05
0
50
100
150


Box plot of OpenPorchSF
OpenPorchSF
0
20
40
60
80
100
120
140


Distribution of BsmtUnfSF
Density
BsmtUnfSF
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0
500
1000
1500
2000


Box plot of BsmtUnfSF
BsmtUnfSF
0
250
500
750
1000
1250
1500
1750






```
(923, 36)
```

Model Building.

In [56]:
```
x_top = top.drop('SalePrice',axis = 1)
y_top = top.SalePrice
```

In [57]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x_top)
```

In [58]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y_top,test_size=0.20,random_state=90)
```

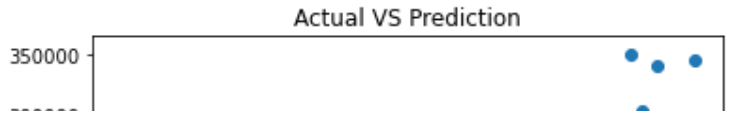
In [59]:
```
def score(reg, x_train, x_test, y_train, y_test, train = True):
    if train:
        y_pred = reg.predict(x_train)
        print('\n ----- Train Result ----- \n')
        print('R2 Score:', metrics.r2_score(y_train,y_pred))

    elif train == False:
        pred = reg.predict(x_test)
        print('\n ----- Test Result ----- \n')
        print('R2 Score:', metrics.r2_score(y_test,pred))
        print('\n ----- Model Evalution ----- \n')
        print('Mean Absolute Error:',mean_absolute_error(y_test,pred))
        print('\n Scatter Plot \n')
        plt.scatter(y_test, pred)
        plt.xlabel("Actual Sales")
        plt.ylabel("Predicted Sales")
        plt.title("Actual VS Prediction")
        plt.show()
```

In [76]:
```
lr = LinearRegression()
rf = RandomForestRegressor()
ada = AdaBoostRegressor()
gb = GradientBoostingRegressor()
hgb = HistGradientBoostingRegressor()
knn = KNeighborsRegressor()
```

Linear Regression

In [77]:
```
lr.fit(x_train,y_train)
score(lr, x_train,x_test,y_train,y_test,train = True)
score(lr, x_train,x_test,y_train,y_test,train = False)
```

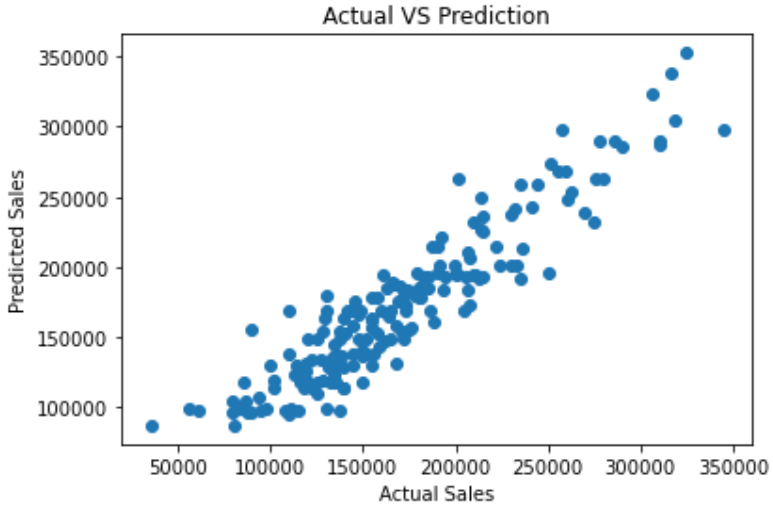
```
 ----- Train Result -----

R2 Score: 0.8760041136039864

 ----- Test Result -----

R2 Score: 0.8971125451644468

 ----- Model Evalution -----

Mean Absolute Error: 14211.539583153673

 Scatter Plot
```



- AdaBoost Regressor

In [78]:
```
ada.fit(x_train,y_train)
score(ada, x_train,x_test,y_train,y_test,train = True)
score(ada, x_train,x_test,y_train,y_test,train = False)
```

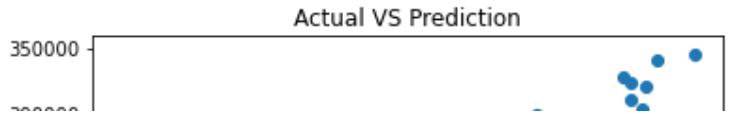
```
 ----- Train Result -----

R2 Score: 0.8872274621687314

 ----- Test Result -----

R2 Score: 0.8673312194058771

 ----- Model Evalution -----

Mean Absolute Error: 16694.31094466189

 Scatter Plot
```



- GradientBoosting Regressor

In [79]:
```
gb.fit(x_train,y_train)
score(gb, x_train,x_test,y_train,y_test,train = True)
score(gb, x_train,x_test,y_train,y_test,train = False)
```

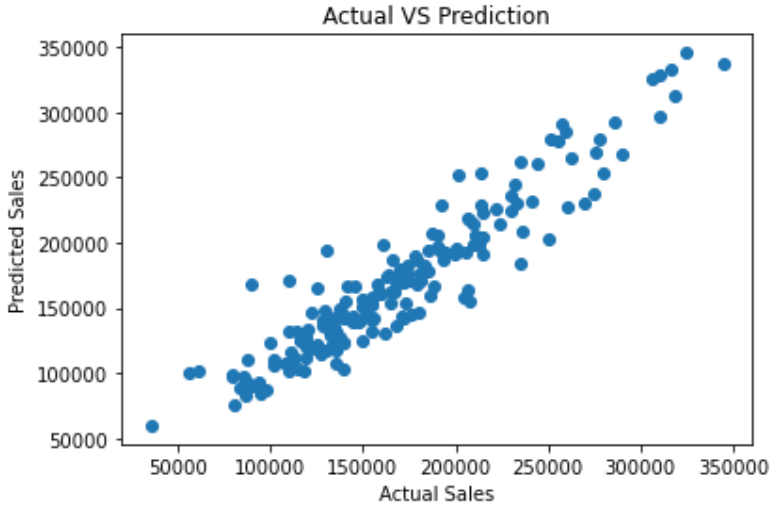
```
 ----- Train Result -----

R2 Score: 0.9595292656376133

 ----- Test Result -----

R2 Score: 0.8920440224945367

 ----- Model Evalution -----

Mean Absolute Error: 14322.815806683744

 Scatter Plot
```



HistGradientBoosting Regressor

In [80]:
```
hgb.fit(x_train,y_train)
score(hgb, x_train,x_test,y_train,y_test,train = True)
score(hgb, x_train,x_test,y_train,y_test,train = False)
```

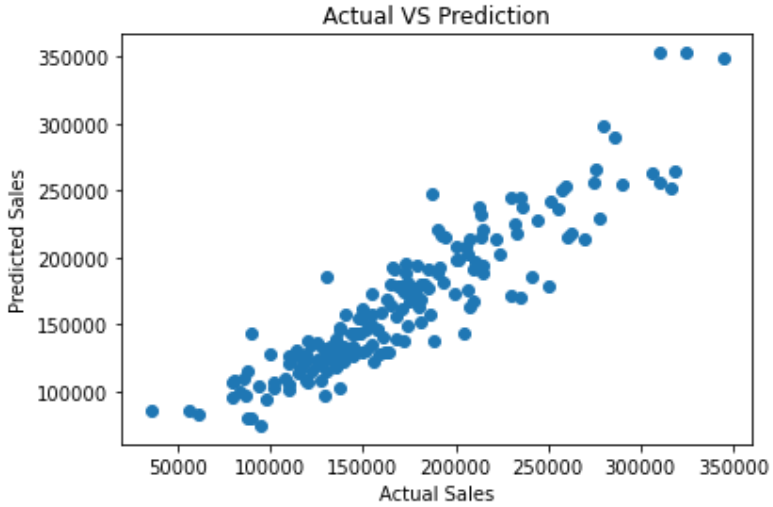
```
 ----- Train Result -----

R2 Score: 0.9870244410795318

 ----- Test Result -----

R2 Score: 0.8799940131761304

 ----- Model Evalution -----

Mean Absolute Error: 14616.643366379834

 Scatter Plot
```



- KNeighbors Regressor.

In [82]:
```
knn.fit(x_train,y_train)
score(knn, x_train,x_test,y_train,y_test,train = True)
score(knn, x_train,x_test,y_train,y_test,train = False)
```

```
 ----- Train Result -----

R2 Score: 0.8754474973001353

 ----- Test Result -----

R2 Score: 0.836770321726306

 ----- Model Evalution -----

Mean Absolute Error: 17267.794594594594

 Scatter Plot
```



Hyperparameter Tuning.

In [83]:
```
gb.get_params()
```

Out[83]:
```
{'alpha': 0.9,
 'ccp_alpha': 0.0,
 'criterion': 'friedman_mse',
 'init': None,
 'learning_rate': 0.1,
 'loss': 'squared_error',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_iter_no_change': None,
 'random_state': None,
 'subsample': 1.0,
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': 0,
 'warm_start': False}
```

In [88]:
```
param = {'n_estimators':range(0,100,10),
         'learning_rate':[0.1,0.3,0.5,0.7,0.8,1.0],
         'criterion':['friedman_mse', 'squared_error'],
         'loss':['squared_error', 'absolute_error'],
         'random_state':range(1,10),
         'max_features':['auto', 'sqrt']}
```

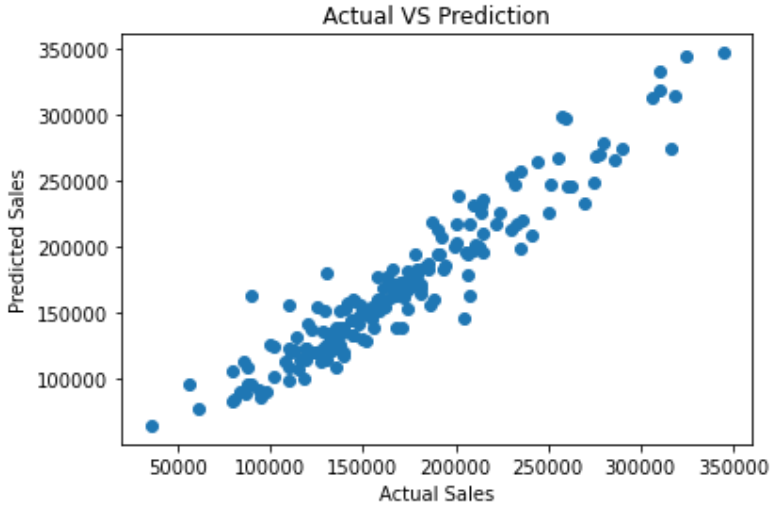
In [89]:
```
grid = GridSearchCV(gb,param_grid = param)
grid.fit(x_train,y_train)
grid.best_params_
```

Out[89]:
```
{'criterion': 'friedman_mse',
 'learning_rate': 0.1,
 'loss': 'squared_error',
 'max_features': 'sqrt',
 'n_estimators': 90,
 'random_state': 2}
```

In [90]:
```
gb_hyp = GradientBoostingRegressor(criterion = 'friedman_mse',learning_rate =  0.1,loss = 'squared_error',
                                   max_features = 'sqrt',n_estimators = 90,random_state = 2)
```

In [91]:
```
gb_hyp.fit(x_train,y_train)
score(gb_hyp, x_train,x_test,y_train,y_test,train = True)
score(gb_hyp, x_train,x_test,y_train,y_test,train = False)
```

```
 ----- Train Result -----

R2 Score: 0.95027979876775

 ----- Test Result -----

R2 Score: 0.9060750260311207

 ----- Model Evalution -----

Mean Absolute Error: 13037.56131117366

 Scatter Plot
```



Testing Dataset

In [92]:
```
df1 = pd.read_csv('test.csv')
df1
```

Out[92]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Feedr
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	BrDale	Norm

In [94]:
```
df1.columns[df1.isnull().any()]
```

Out[94]:
```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',
       'MiscFeature'],
      dtype='object')
```

In [95]:
```
df1.drop(columns = ['MiscFeature','PoolQC','Alley'],axis = 1, inplace = True)
```

In [97]:
```
df1['MasVnrArea'] = df1['MasVnrArea'].fillna(df1['MasVnrArea'].mean())
```

In [98]:
```
df1['LotFrontage'] = df1['LotFrontage'].fillna(df1['LotFrontage'].median())
df1['GarageYrBlt'] = df1['GarageYrBlt'].fillna(df1['GarageYrBlt'].median())
```

In [100]:
```
def fill(x):
    df1[x] = df1[x].fillna(df1[x].mode()[0])
```

In [101]:
```
n = ['MasVnrType','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','FireplaceQu','GarageType',
'GarageFinish','GarageQual','GarageCond','Fence','Electrical']
```

In [102]:
```
for i in n:
    fill(i)
```

In [103]:
```
df1.columns[df1.isnull().any()]
```

Out[103]:
```
Index([], dtype='object')
```

In [104]:
```
df1.drop('Utilities',axis = 1, inplace = True )
```

In [105]:
```
ob_test = []
for col in df1:
    if df1[col].dtype == 'object':
        ob_test.append(col)
```

In [106]:
```
lb = LabelEncoder()
for i in ob_test:
    df1[i] = lb.fit_transform(df1[i])
```

In [108]:
```
top_test = df1[['OverallQual','ExterQual','GrLivArea','BsmtQual','KitchenQual','GarageCars','FullBath',
                'GarageFinish','YearBuilt','1stFlrSF','MasVnrArea','Street','LotArea',
                'YearRemodAdd','GarageYrBlt','Heating','MSZoning','Fireplaces','CentralAir',
                'Foundation','BsmtFinSF1','2ndFlrSF','OpenPorchSF','LotShape','BsmtUnfSF','HeatingQC',
                'HalfBath','Neighborhood','WoodDeckSF','BsmtExposure','FireplaceQu','BsmtCond','RoofMatl',
                'GarageType','BedroomAbvGr']]
```

In [110]:
```
top_test
```

Out[110]:

	OverallQual	ExterQual	GrLivArea	BsmtQual	KitchenQual	GarageCars	FullBath	GarageFinish	YearBuilt	1stFlrSF	MasVnrArea	Street	LotArea	YearRem
0	9	2	1922	0	2	3	2	0	2005	1922	200.0	1	14157	
1	8	2	1360	2	2	2	1	1	1984	1360	0.0	1	5814	
2	8	2	1788	2	0	2	2	1	2001	1788	0.0	1	11838	
3	7	3	1564	3	1	1	1	2	1941	860	0.0	1	12000	
4	6	2	1933	2	2	3	2	0	2007	894	74.0	1	14598	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	8	3	1563	2	2	3	2	1	2007	1563	468.0	1	10206	
288	5	3	990	2	3	2	1	2	1994	990	0.0	1	9245	
289	6	3	1004	3	3	2	1	0	1970	1004	180.0	1	11241	
290	6	3	1664	3	3	2	2	2	1947	1004	0.0	1	5000	
291	6	3	987	3	3	1	1	2	1973	483	408.0	1	1953	

292 rows × 35 columns

In [111]:
```
y_pred = gb_hyp.predict(top_test)
```

In [112]:
```
a = []
for i in y_pred:
    a.append(i)
```

In [113]:
```
test = pd.DataFrame({'TEST' : a})
test.head(50)
```

Out[113]:

	TEST
0	310088.543882
1	290352.395591
2	257274.190800
3	279147.659430
4	299193.188175
5	261860.182107
6	302525.141075
7	274537.009266
8	322419.231011
9	299086.387709
10	250385.838736
11	280148.827281
12	237117.269632
13	276435.464627
14	274626.697456
15	283643.082669
16	259523.166298
17	267150.473323
18	295808.847253
19	299981.162257
20	275709.706966
21	258349.359916
22	277322.262453
23	264801.849559
24	257360.246716
25	284726.869228
26	267075.100582
27	299285.711020

	TEST
28	297510.105139
29	236371.026794
30	262443.879657
31	274537.009266
32	301226.295129
33	278096.095930
34	250385.838736
35	278258.320000
36	307169.937356
37	313705.325539
38	298001.982103
39	306613.341950
40	259523.166298
41	317731.489372
42	293305.269603
43	265674.760512
44	261392.558897
45	286219.576468
46	284726.869228
47	231934.904905
48	288192.030563
49	321813.245488

Saving the Model.

In [114]:
```
filename = 'House_model.pickle'
pickle.dump(gb_hyp, open(filename, 'wb'))
```