House Prices - Advanced Regression Techniques

1. EDA (Univariate, Multivariate, KDE, Pearson Correlation)

2. Data Preprocessing (Imputation, create at least 2 new features)

3. Cross-validation

4. An inference pipeline consisting of Data Preprocessing and prediction.

IMPORTING REQUIRED LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

LOADING THE DATA SETS

In [2]:
```
house_train=pd.read_csv("D:\\data set\\house-prices-advanced-regression-techniques\\
house_test= pd.read_csv("D:\\data set\\house-prices-advanced-regression-techniques\\
```

Exploratory Data Analysis (EDA)

In [3]:
```
house_train.head() # to display top 5 rows
```

Out[3]:

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

5 rows × 81 columns

In [4]:
```
house_train.tail()  # to display last 5 rows
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl

5 rows × 81 columns

In [5]:
```
house_train.info() # to get the information on the different features, its data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
```

```
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

FROM ABOVE WE CAN OBSERVE THAT OUR DATA SET CONTAINS "43" OBJECT DATA TYPE, "38" NUMERICAL DATA TYPE ("3" FLOAT, "35" INT)

In [6]:
```
house_train.isna().sum().sort_values(ascending=False)
```

Out[6]:
```
PoolQC         1453
MiscFeature    1406
Alley          1369
Fence          1179
FireplaceQu     690
               ...
ExterQual         0
Exterior2nd       0
Exterior1st       0
RoofMatl          0
```

```
SalePrice         0
Length: 81, dtype: int64
```

from above we can observe that column as "PoolQ","MiscFeature","Alley","Fence" and "FireplaceQu" have max no of missing values

In [7]:
```
house_train.describe() # statistical describtion on numerical columns
```

Out[7]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuil
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.00000(
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.26780{
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.20290⁴
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.00000(
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.00000(
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.00000(
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.00000(
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.00000(

8 rows × 38 columns

FROM THE ABOVE CODE WE CAN FIND THE MIN VALUE AND THE MAX VALUE OF THE RESPECTIVE COLUMN IE THE SPREAD OF DATA. WE CAN OBSERVE THAT IN MANY COLUMNS MEAN IS GREATER THAN MEDIAN SO OUR DATA HAVE POSITIVE/RIGHT SKEW.

In [8]:
```
house_train.select_dtypes(include=['int64', 'float64']).columns
```

Out[8]:
```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

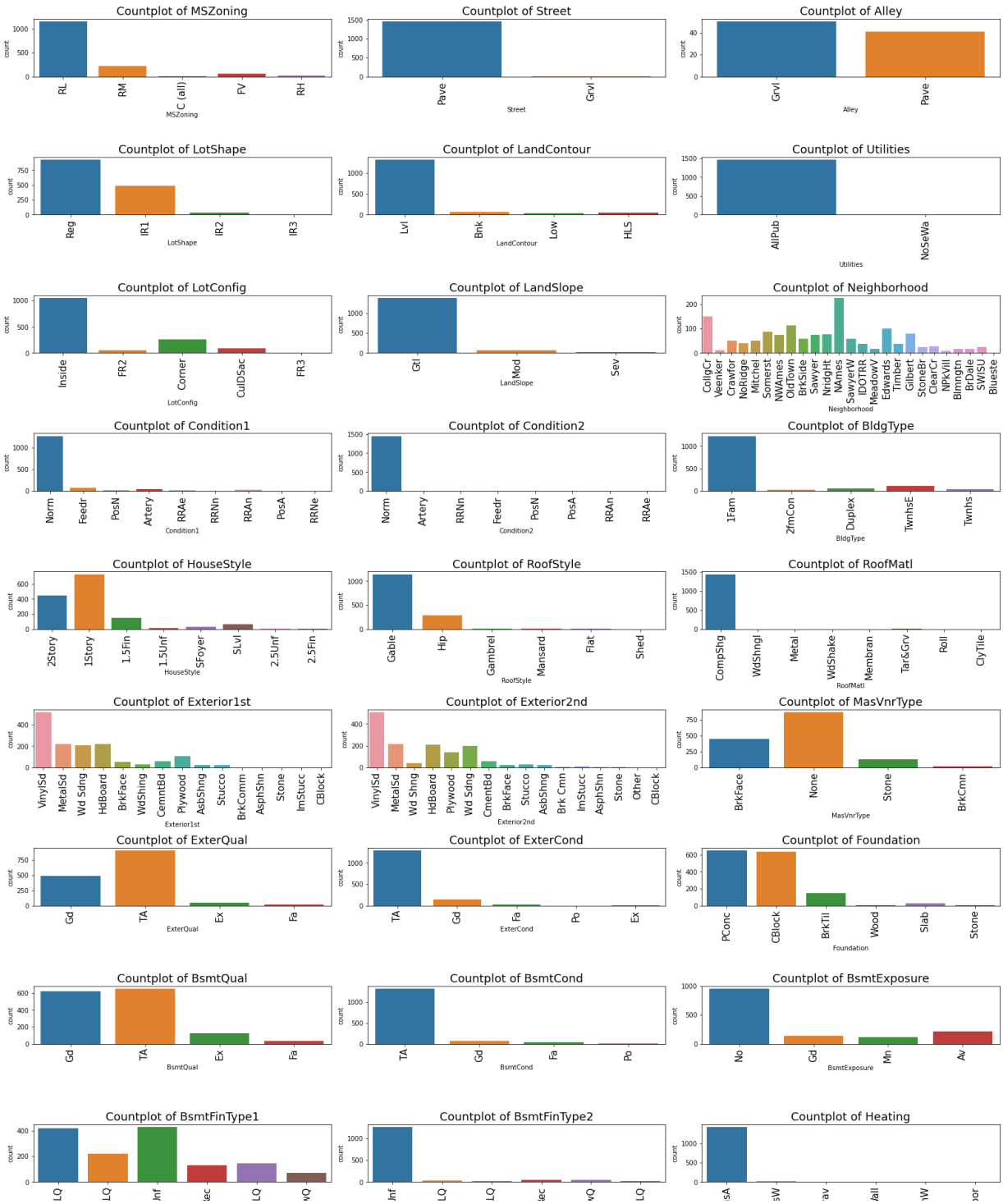
In [9]:
```
house_train.select_dtypes(include=['object']).columns
```

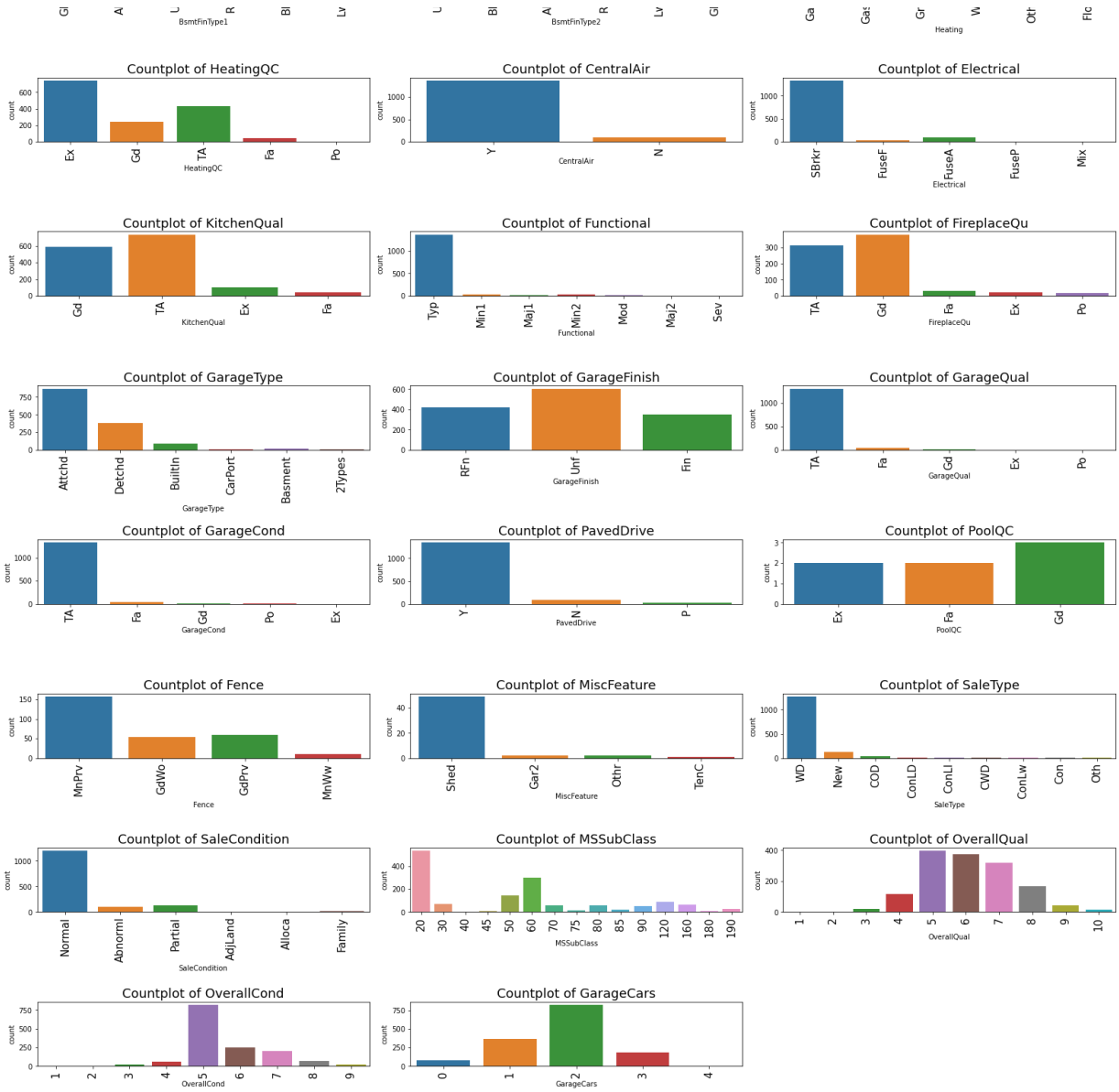
Out[9]:
```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

Univariate Analysis (EDA)

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

In [12]:
```
plt.figure(figsize=(22,300))
for i in range(len(col)):
  plt.subplot(100,3,i+1)
  sns.countplot(house_train[col[i]])
  plt.title(f"Countplot of {col[i]}",fontsize=18)
  plt.xticks(rotation=90,fontsize=15)
  plt.tight_layout()
```





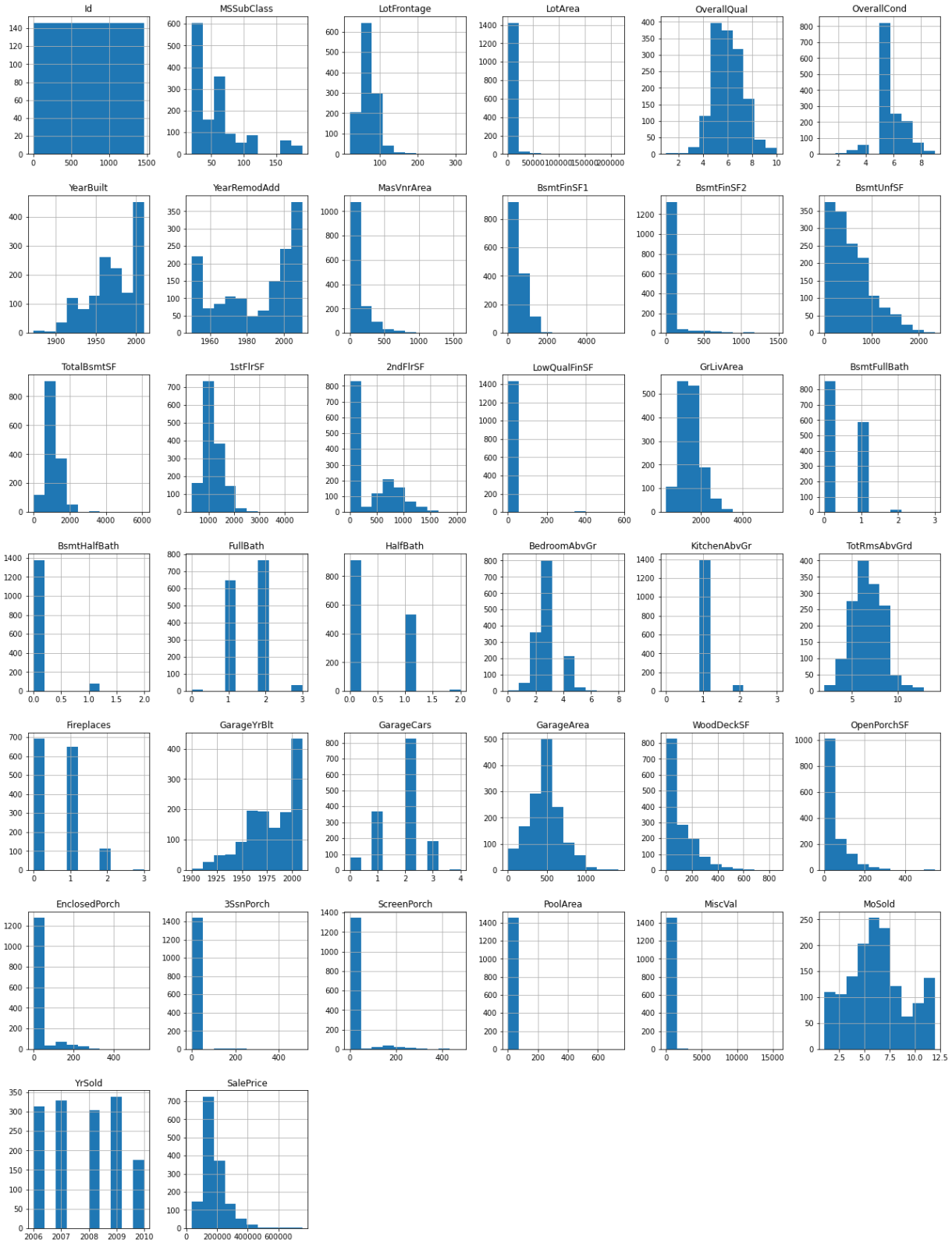
Observation

From the above graphs we can observe that a) In MSZoning people like to stay where there is low population. b) people like to have their house in Pave street type. c) from lotshape people like Reg and dont like IR2,IR3.etc

In [13]:
```
house_train.hist(figsize=(22,30))
```

Out[13]:
```
array([[<AxesSubplot:title={'center':'Id'}>,
        <AxesSubplot:title={'center':'MSSubClass'}>,
        <AxesSubplot:title={'center':'LotFrontage'}>,
        <AxesSubplot:title={'center':'LotArea'}>,
        <AxesSubplot:title={'center':'OverallQual'}>,
        <AxesSubplot:title={'center':'OverallCond'}>],
       [<AxesSubplot:title={'center':'YearBuilt'}>,
        <AxesSubplot:title={'center':'YearRemodAdd'}>,
        <AxesSubplot:title={'center':'MasVnrArea'}>,
        <AxesSubplot:title={'center':'BsmtFinSF1'}>,
        <AxesSubplot:title={'center':'BsmtFinSF2'}>,
        <AxesSubplot:title={'center':'BsmtUnfSF'}>],
       [<AxesSubplot:title={'center':'TotalBsmtSF'}>,
        <AxesSubplot:title={'center':'1stFlrSF'}>,
        <AxesSubplot:title={'center':'2ndFlrSF'}>,
```

```
 <AxesSubplot:title={'center':'LowQualFinSF'}>,
 <AxesSubplot:title={'center':'GrLivArea'}>,
 <AxesSubplot:title={'center':'BsmtFullBath'}>],
[<AxesSubplot:title={'center':'BsmtHalfBath'}>,
 <AxesSubplot:title={'center':'FullBath'}>,
 <AxesSubplot:title={'center':'HalfBath'}>,
 <AxesSubplot:title={'center':'BedroomAbvGr'}>,
 <AxesSubplot:title={'center':'KitchenAbvGr'}>,
 <AxesSubplot:title={'center':'TotRmsAbvGrd'}>],
[<AxesSubplot:title={'center':'Fireplaces'}>,
 <AxesSubplot:title={'center':'GarageYrBlt'}>,
 <AxesSubplot:title={'center':'GarageCars'}>,
 <AxesSubplot:title={'center':'GarageArea'}>,
 <AxesSubplot:title={'center':'WoodDeckSF'}>,
 <AxesSubplot:title={'center':'OpenPorchSF'}>],
[<AxesSubplot:title={'center':'EnclosedPorch'}>,
 <AxesSubplot:title={'center':'3SsnPorch'}>,
 <AxesSubplot:title={'center':'ScreenPorch'}>,
 <AxesSubplot:title={'center':'PoolArea'}>,
 <AxesSubplot:title={'center':'MiscVal'}>,
 <AxesSubplot:title={'center':'MoSold'}>],
[<AxesSubplot:title={'center':'YrSold'}>,
 <AxesSubplot:title={'center':'SalePrice'}>, <AxesSubplot:>,
 <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



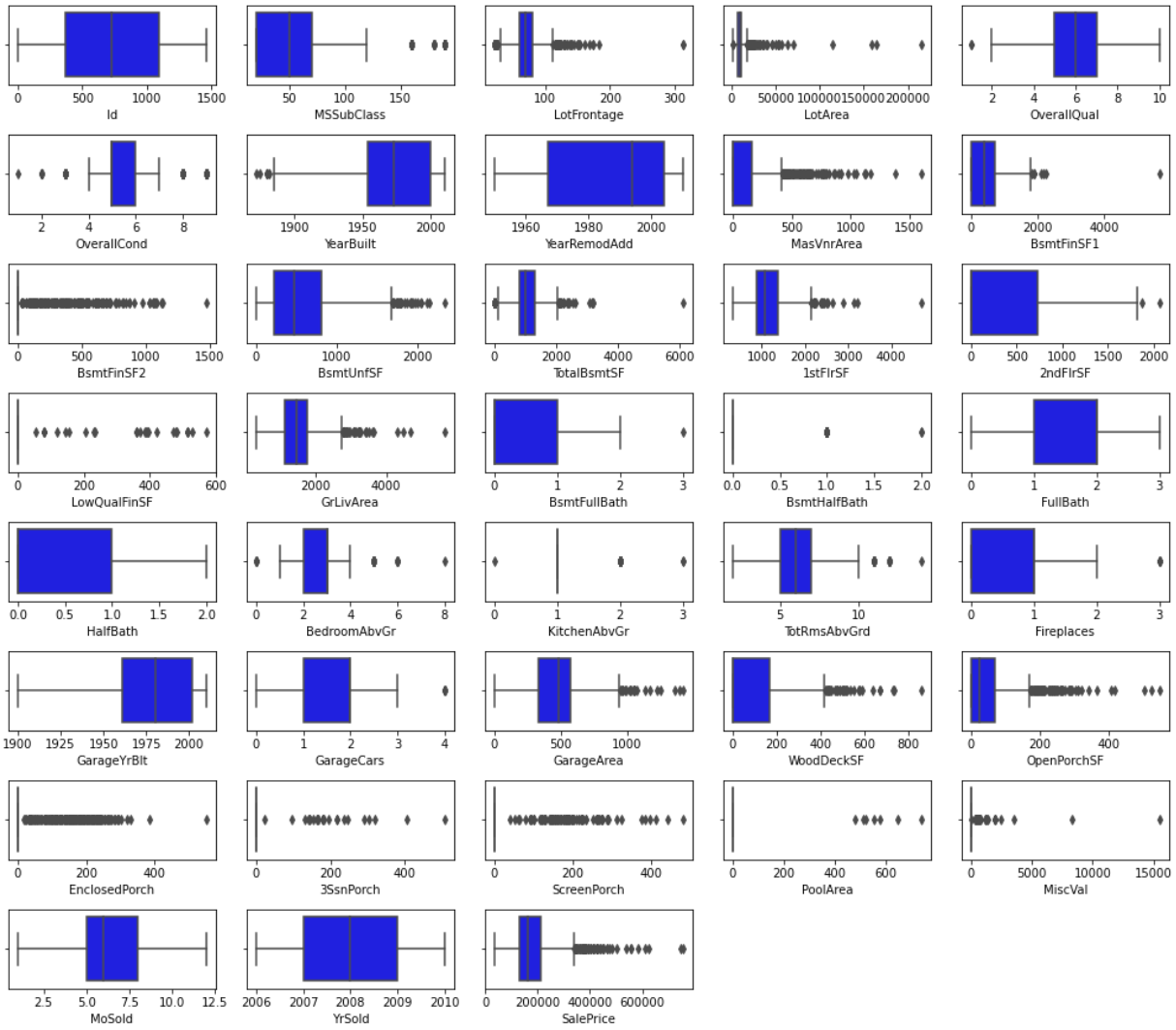
In [14]:
```
data=house_train.select_dtypes(exclude='object')
```

In [15]:
```
X_col=data.columns.values
```

Checking outliner

In [16]:
```
plt.figure(figsize=(14,30))
for i in range(0,len(X_col)):
  plt.subplot(20,5,i+1)
```

```
    ax=sns.boxplot(data[X_col[i]],color='blue')
    plt.tight_layout()
```

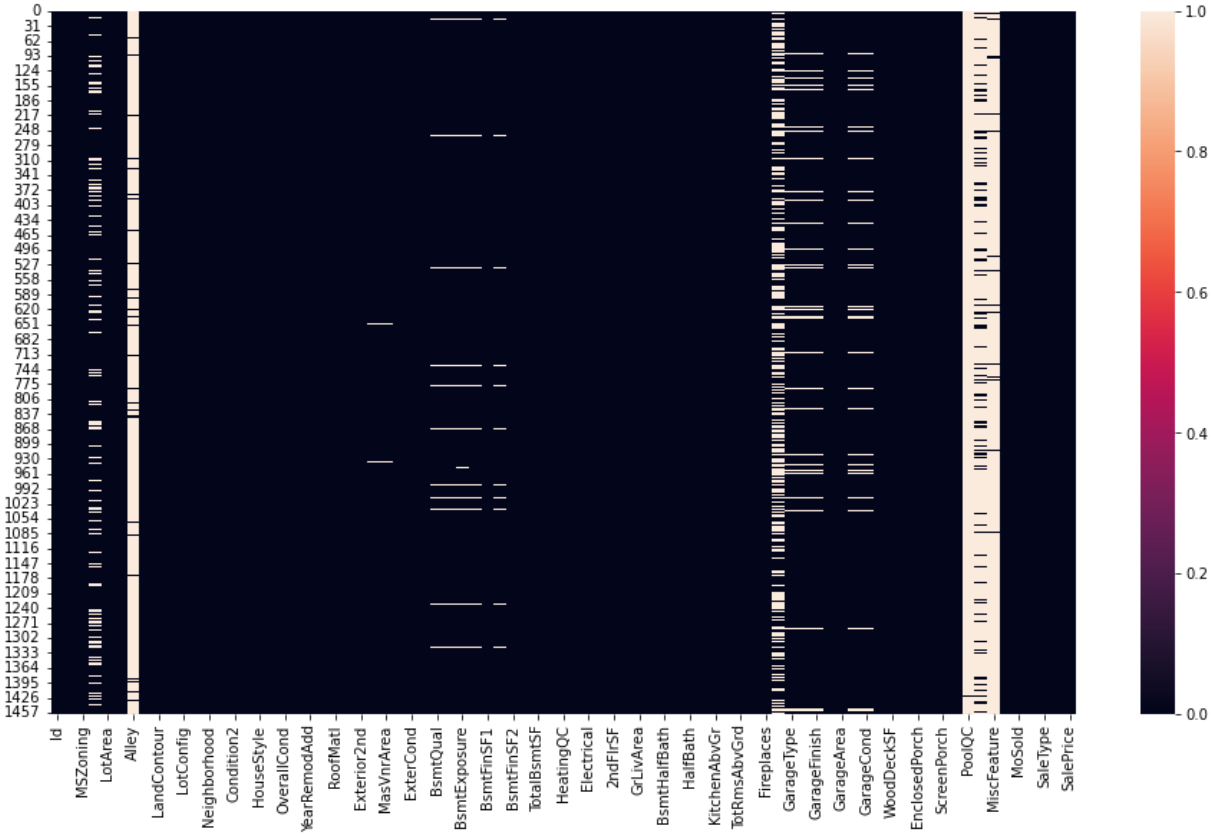


There are many outliners we can observe from the above box plot.

Treating missing values

In [17]:
```
# Show the null values using heatmap
plt.figure(figsize=(16,9))
sns.heatmap(house_train.isnull())
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
# Get the percentages of null value
null_percent = house_train.isnull().sum()/house_train.shape[0]*100
null_percent
```

Out[18]:
```
Id               0.000000
MSSubClass       0.000000
MSZoning         0.000000
LotFrontage     17.739726
LotArea          0.000000
                  ...
MoSold           0.000000
YrSold           0.000000
SaleType         0.000000
SaleCondition    0.000000
SalePrice        0.000000
Length: 81, dtype: float64
```

In [20]:
```
col_for_drop = null_percent[null_percent > 20].keys() # if the null value % 20 or >
col_for_drop
```

Out[20]:
```
Index(['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'], dtype='object')
```

In [22]:
```
# drop columns
df = house_train.drop(col_for_drop, "columns")
df.shape
```

Out[22]:
```
(1460, 76)
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
```

```
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
       'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
       'Functional', 'Fireplaces', 'GarageType', 'GarageYrBlt', 'GarageFinish',
       'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive',
       'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Data Processing

Fill Missing Values

In [24]:
```
df['LotFrontage']=df['LotFrontage'].fillna(df['LotFrontage'].mean())
```

In [25]:
```
df['BsmtCond']=df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtQual']=df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['GarageType']=df['GarageType'].fillna(df['GarageType'].mode()[0])
df['GarageFinish']=df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual']=df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond']=df['GarageCond'].fillna(df['GarageCond'].mode()[0])
```

In [26]:
```
df.shape
```

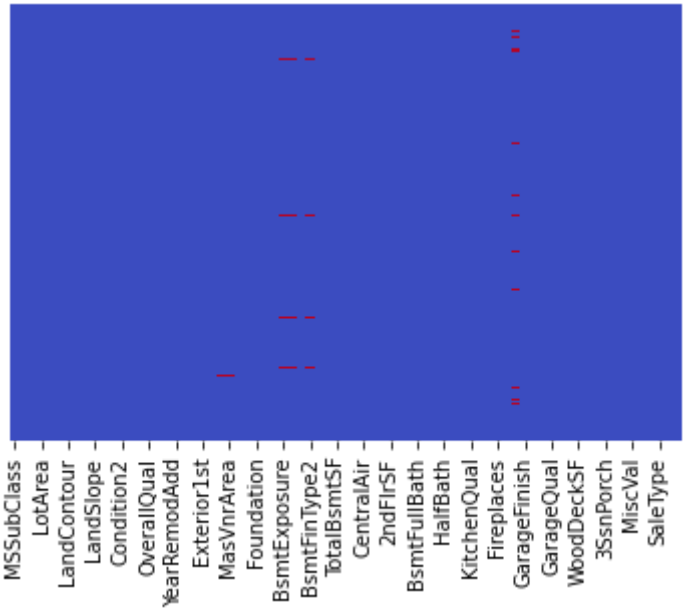
Out[26]:
```
(1460, 76)
```

In [27]:
```
df.drop(['Id'],axis=1,inplace=True)
df.isnull().sum()
```

Out[27]:
```
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
Street           0
                ..
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 75, dtype: int64
```

In [28]:
```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
```

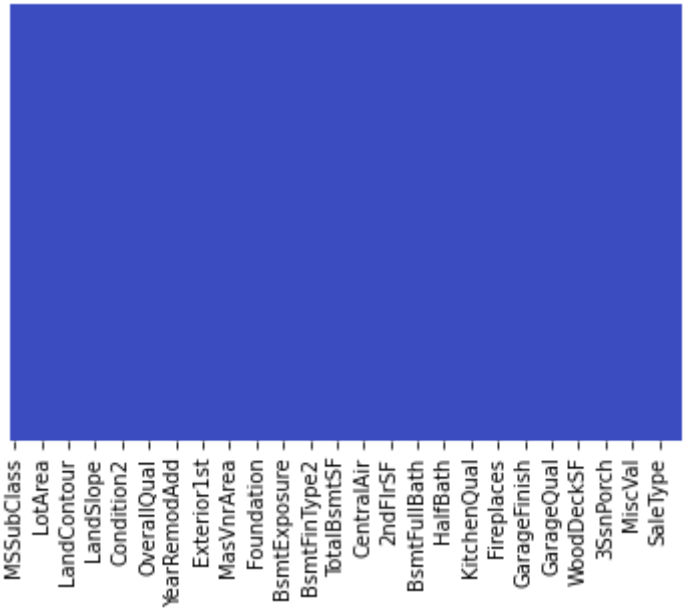
Out[28]:
```
<AxesSubplot:>
```



In [29]:
```
df.dropna(inplace=True)
```

In [30]:
```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
```

Out[30]: <AxesSubplot:>

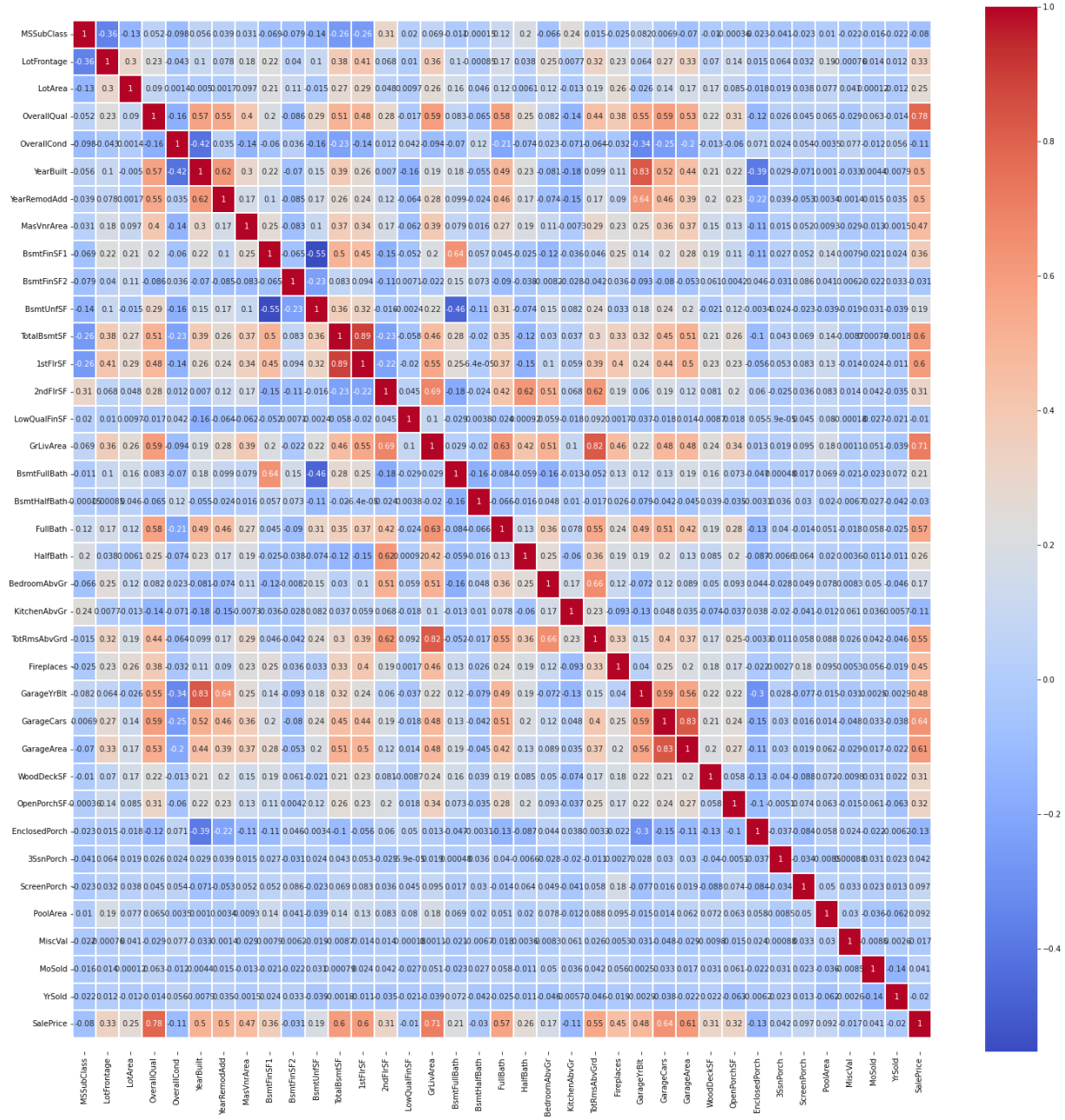


In [31]:
```
df.shape
```

Out[31]: (1338, 75)

In [32]:
```
# correlation heatmap
plt.figure(figsize=(25,25))
ax = sns.heatmap(df.corr(), cmap = "coolwarm", annot=True, linewidth=2)

# to fix the bug "first and last row cut in half of heatmap plot"
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[32]: (37.5, -0.5)



Correlation

In [34]:
```
corr=df.corr()["SalePrice"]
corr[np.argsort(corr, axis=0)[::-1]]
```

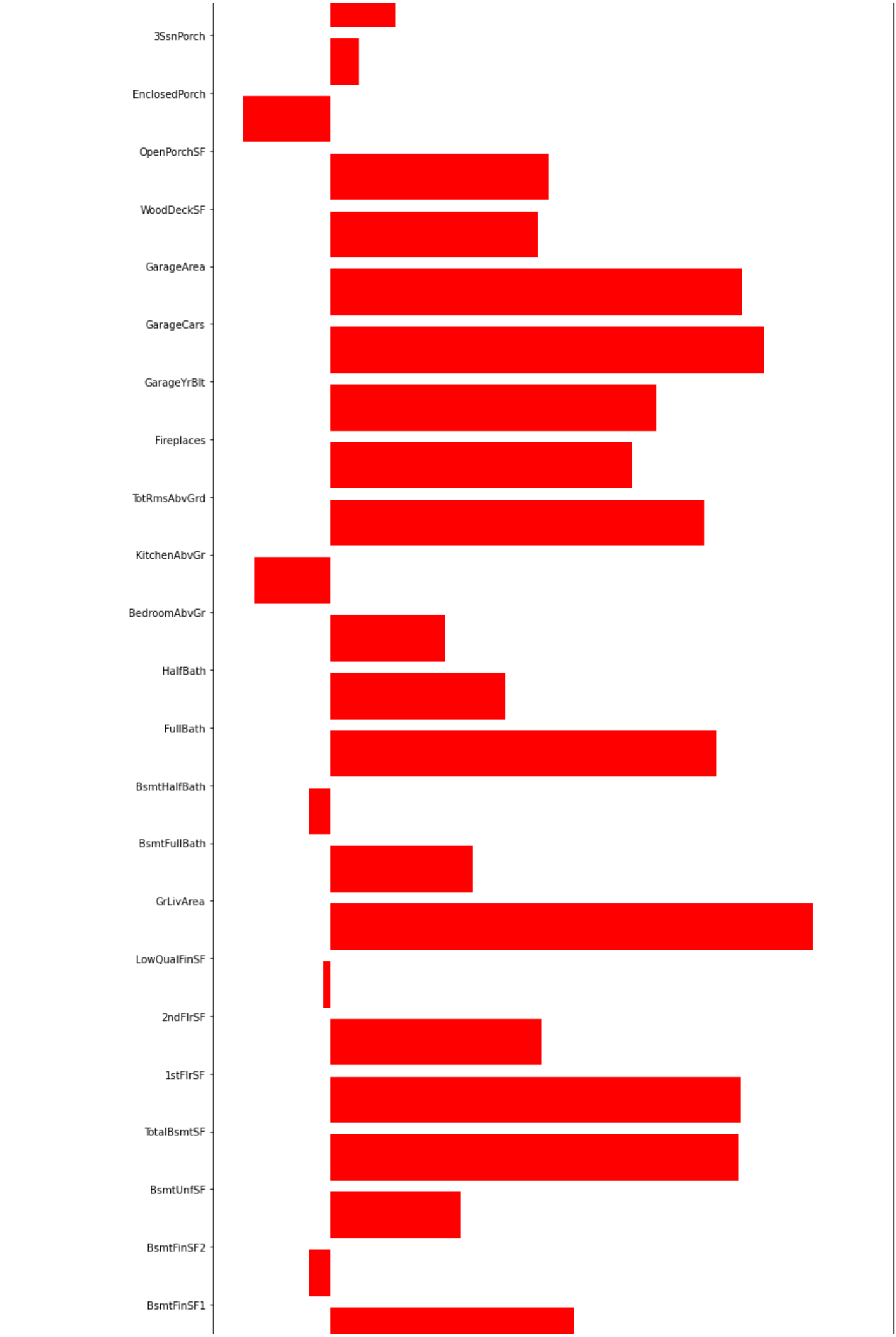
Out[34]:
```
SalePrice        1.000000
OverallQual      0.783546
GrLivArea        0.711706
GarageCars       0.640154
GarageArea       0.607535
1stFlrSF         0.604714
TotalBsmtSF      0.602042
FullBath         0.569313
TotRmsAbvGrd     0.551821
YearBuilt        0.504297
YearRemodAdd     0.501435
GarageYrBlt      0.481730
MasVnrArea       0.465811
Fireplaces       0.445434
BsmtFinSF1       0.359677
LotFrontage      0.327831
```

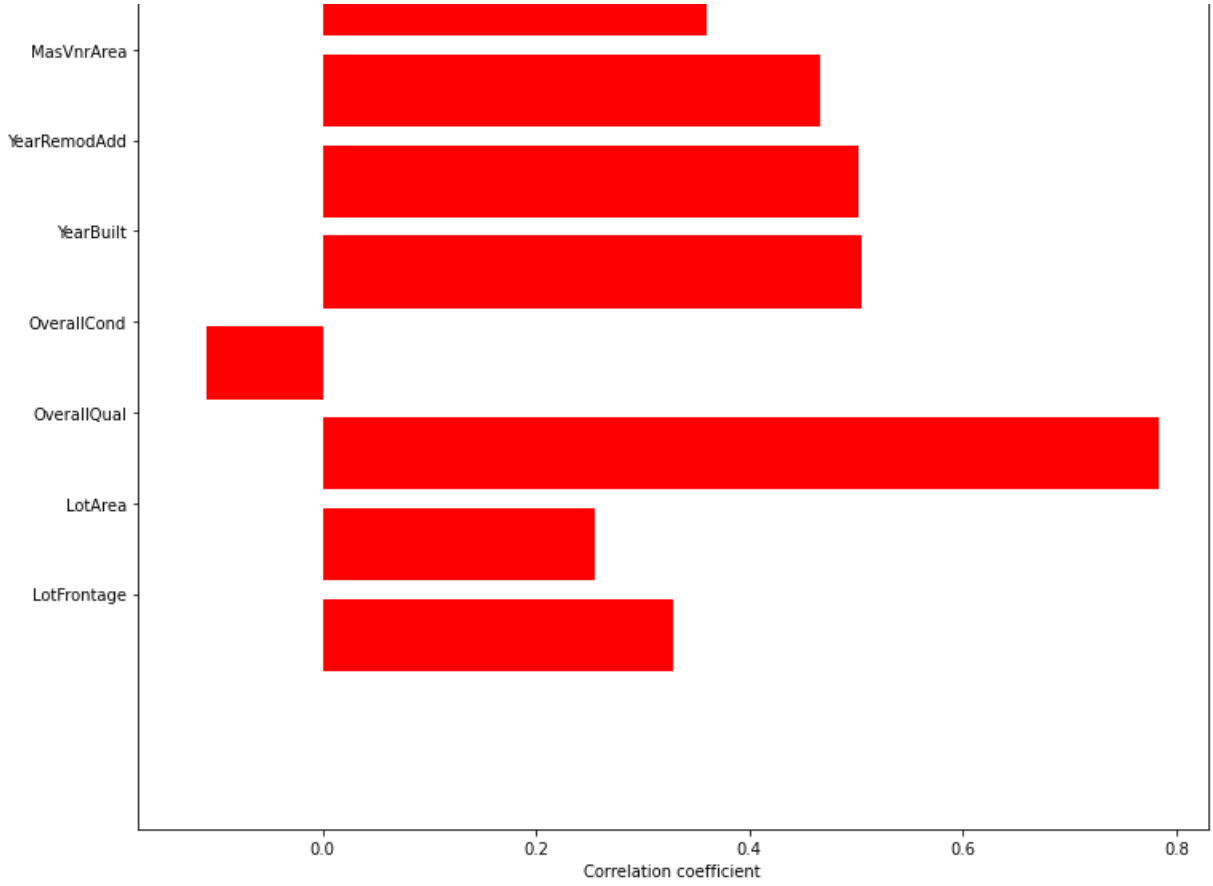
```
OpenPorchSF      0.322786
2ndFlrSF         0.311354
WoodDeckSF       0.305983
HalfBath         0.258175
LotArea          0.254757
BsmtFullBath     0.209695
BsmtUnfSF        0.191689
BedroomAbvGr     0.169266
ScreenPorch      0.096624
PoolArea         0.091881
3SsnPorch        0.042159
MoSold           0.041310
LowQualFinSF    -0.009992
MiscVal         -0.016990
YrSold          -0.020451
BsmtHalfBath    -0.030175
BsmtFinSF2      -0.031226
MSSubClass      -0.079599
OverallCond     -0.108627
KitchenAbvGr    -0.111408
EnclosedPorch   -0.127385
Name: SalePrice, dtype: float64
```

from the above table we can find that some columns like "OverallQual ", " GrLivArea ", " GrLivArea " etc have correlation more than 0.5 with sales price. We can make use of such features to predict our target value ie sales price

In [37]:

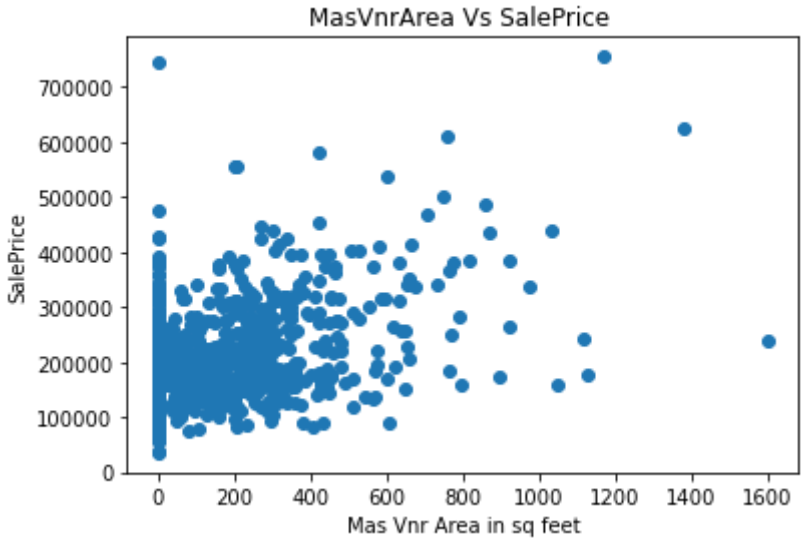
```
#plotting correlations
num_feat=df.columns[df.dtypes!=object]
num_feat=num_feat[1:-1]
labels = []
values = []
for col in num_feat:
    labels.append(col)
    values.append(np.corrcoef(df[col].values, df.SalePrice.values)[0,1])

ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(12,40))
rects = ax.barh(ind, np.array(values), color='red')
ax.set_yticks(ind+((width)/2.))
ax.set_yticklabels(labels, rotation='horizontal')
ax.set_xlabel("Correlation coefficient")
ax.set_title("Correlation Coefficients w.r.t Sale Price");
```


3SsnPorch
EnclosedPorch
OpenPorchSF
WoodDeckSF
GarageArea
GarageCars
GarageYrBlt
Fireplaces
TotRmsAbvGrd
KitchenAbvGr
BedroomAbvGr
HalfBath
FullBath
BsmtHalfBath
BsmtFullBath
GrLivArea
LowQualFinSF
2ndFlrSF
1stFlrSF
TotalBsmtSF
BsmtUnfSF
BsmtFinSF2
BsmtFinSF1

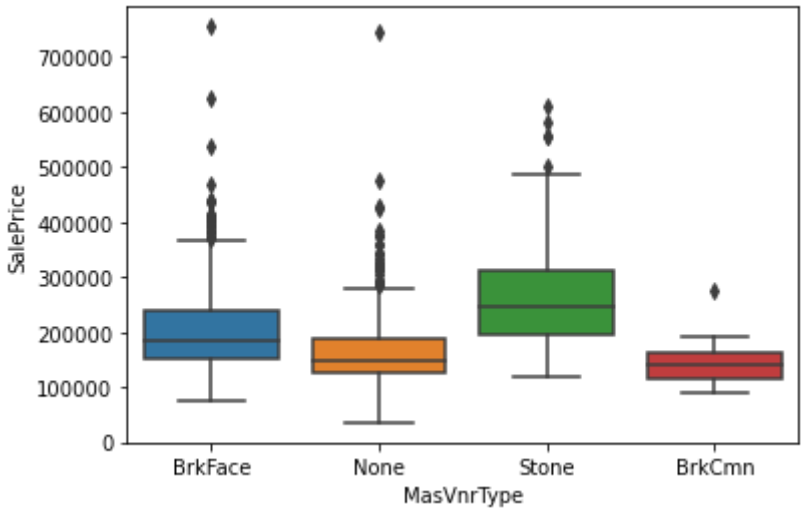


we can find that some columns like "overallcod","BsmtfinSF2",etc have negative correlation with respect to salesprice.

In [41]:
```
# MasVnrArea Vs SalePrice


plt.scatter(df["MasVnrArea"],df["SalePrice"])
plt.title("MasVnrArea Vs SalePrice ")
plt.ylabel("SalePrice")
plt.xlabel("Mas Vnr Area in sq feet");
```

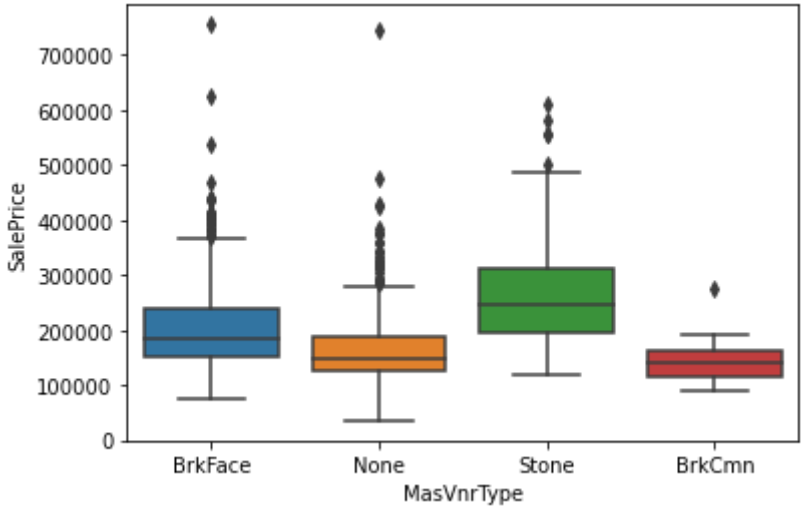


In [43]:
```
sns.boxplot("MasVnrType","SalePrice",data=df);
```

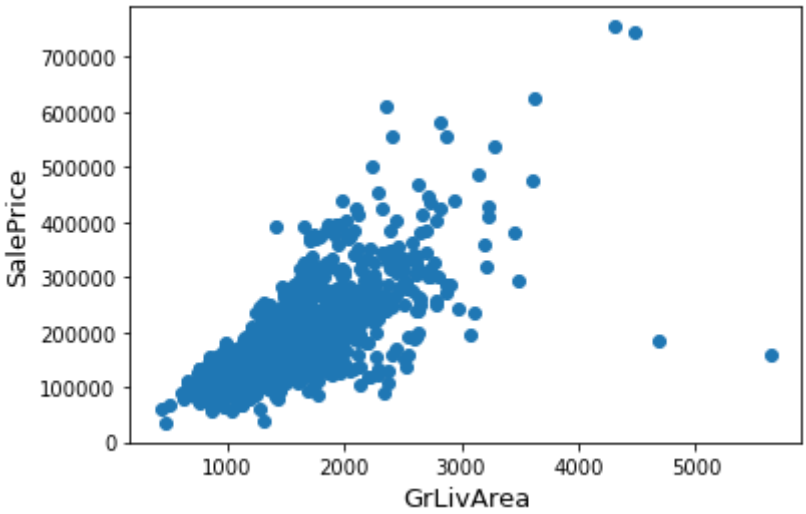


In [44]:
```
df["MasVnrType"] = df["MasVnrType"].fillna('None')
df["MasVnrArea"] = df["MasVnrArea"].fillna(0.0)
```

In [45]:
```
sns.boxplot("MasVnrType","SalePrice",data=df);
```

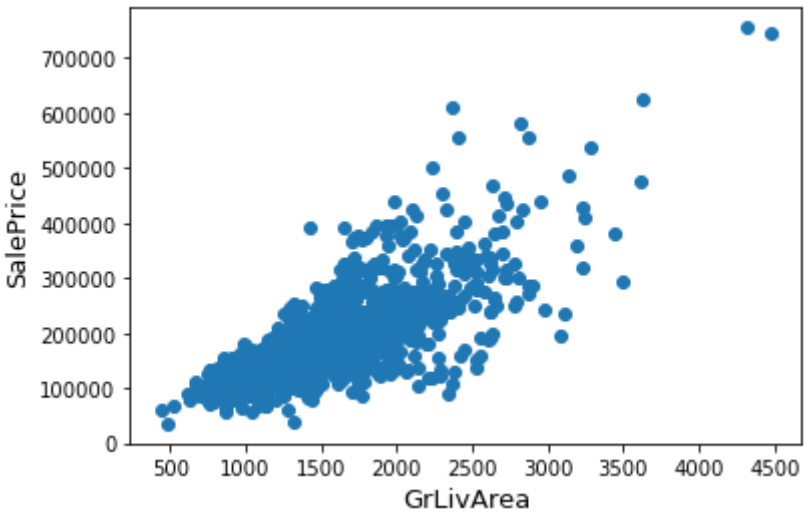


In [46]:
```
fig, ax = plt.subplots()
ax.scatter(x = df['GrLivArea'], y = df['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()
```



In [48]:

```
#Deleting outliers
train = df.drop(df[(df['GrLivArea']>4000) & (df['SalePrice']<300000)].index)

#Check the graphic again
fig, ax = plt.subplots()
ax.scatter(train['GrLivArea'], train['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()
```



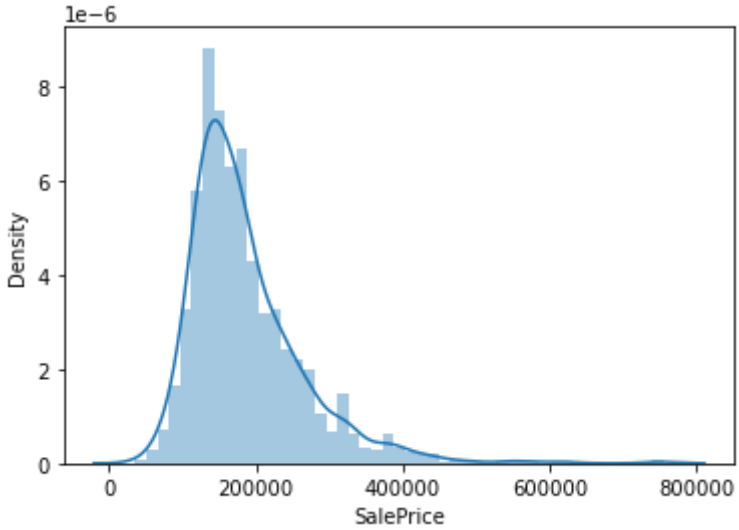
TARGET VALUE

In [54]:

```
sns.distplot(df['SalePrice']);

# Get the fitted parameters used by the function
(mu, sigma) = norm.fit(df['SalePrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))

#Now plot the distribution
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
            loc='best')
plt.ylabel('Frequency')
plt.title('SalePrice distribution')

#Get also the QQ-plot
fig = plt.figure()
```

```
res = stats.probplot(df['SalePrice'], plot=plt)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13284/3772241319.py in <module>
      2
      3 # Get the fitted parameters used by the function
----> 4 (mu, sigma) = norm.fit(df['SalePrice'])
      5 print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))
      6

NameError: name 'norm' is not defined
```



In [ ]: