In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm, skew
from scipy.special import boxcox1p
import warnings
```

In [2]:
```python
warnings.filterwarnings("ignore")
```

Exploratory Data Analysis

In [3]:
```python
df = pd.read_csv("dataset.csv")
pd.set_option('max_columns',81)
```

In [4]:
```python
df
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lv
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lv
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lv
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lv
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lv
...	...	...	...	...	...	...	...	...	.
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lv
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lv
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lv
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lv
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lv

1460 rows × 81 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     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
```

In [6]:
```
print("Data types and their frequency\n{}".format(df.dtypes.value_counts()))
```

```
Data types and their frequency
object     43
int64      35
float64     3
dtype: int64
```

Checking the Data for Missing Values

In [7]:

```
df.isnull().sum().sort_values(ascending = False).to_dict()
```

Out[7]:
```
{'PoolQC': 1453,
 'MiscFeature': 1406,
 'Alley': 1369,
 'Fence': 1179,
 'FireplaceQu': 690,
 'LotFrontage': 259,
 'GarageYrBlt': 81,
 'GarageCond': 81,
 'GarageType': 81,
 'GarageFinish': 81,
 'GarageQual': 81,
 'BsmtFinType2': 38,
 'BsmtExposure': 38,
 'BsmtQual': 37,
 'BsmtCond': 37,
 'BsmtFinType1': 37,
 'MasVnrArea': 8,
 'MasVnrType': 8,
 'Electrical': 1,
 'Id': 0,
 'Functional': 0,
 'Fireplaces': 0,
 'KitchenQual': 0,
 'KitchenAbvGr': 0,
 'BedroomAbvGr': 0,
 'HalfBath': 0,
 'FullBath': 0,
 'BsmtHalfBath': 0,
 'TotRmsAbvGrd': 0,
 'GarageCars': 0,
 'GrLivArea': 0,
 'GarageArea': 0,
 'PavedDrive': 0,
 'WoodDeckSF': 0,
 'OpenPorchSF': 0,
 'EnclosedPorch': 0,
 '3SsnPorch': 0,
 'ScreenPorch': 0,
 'PoolArea': 0,
 'MiscVal': 0,
 'MoSold': 0,
 'YrSold': 0,
 'SaleType': 0,
 'SaleCondition': 0,
 'BsmtFullBath': 0,
 'HeatingQC': 0,
 'LowQualFinSF': 0,
 'LandSlope': 0,
 'OverallQual': 0,
 'HouseStyle': 0,
 'BldgType': 0,
 'Condition2': 0,
 'Condition1': 0,
 'Neighborhood': 0,
 'LotConfig': 0,
 'YearBuilt': 0,
 'Utilities': 0,
```

```
 'LandContour': 0,
 'LotShape': 0,
 'Street': 0,
 'LotArea': 0,
 'MSZoning': 0,
 'OverallCond': 0,
 'YearRemodAdd': 0,
 '2ndFlrSF': 0,
 'BsmtFinSF2': 0,
 '1stFlrSF': 0,
 'CentralAir': 0,
 'MSSubClass': 0,
 'Heating': 0,
 'TotalBsmtSF': 0,
 'BsmtUnfSF': 0,
 'BsmtFinSF1': 0,
 'RoofStyle': 0,
 'Foundation': 0,
 'ExterCond': 0,
 'ExterQual': 0,
 'Exterior2nd': 0,
 'Exterior1st': 0,
 'RoofMatl': 0,
 'SalePrice': 0}
```

In [8]:
```
#Variables that contain more than 80 percent of null values are dropped
df = df.drop(['Id','PoolQC','MiscFeature','Alley','Fence'], axis=1)
```

In [9]:
```
#to replace the following column with missing values with most occuring value
 i.e. Mode of column
missing_val = ['GarageCond','GarageType','GarageYrBlt','GarageFinish','GarageQ
ual','BsmtExposure',
               'BsmtFinType2','BsmtFinType1','BsmtCond','BsmtQual','MasVnrAre
a','MasVnrType','Electrical']
```

In [10]:
```
def impute_nan(DataFrame,ColName):
    Mode_Category = DataFrame[ColName].mode()[0]
    DataFrame[ColName].fillna(Mode_Category,inplace=True)

for Columns in missing_val:
    impute_nan(df,Columns)
```

In [11]:
```
#Imputing missing values with a None Category
df['FireplaceQu'].fillna('None',inplace=True)
#Imputing missing values with a Mean Value
df['LotFrontage'].fillna(int(df['LotFrontage'].mean()),inplace=True)
```

In [12]:
```
df.isnull().sum().sort_values(ascending = False).to_dict()
```

```
Out[12]: {'MSSubClass': 0,
          'HalfBath': 0,
          'FireplaceQu': 0,
          'Fireplaces': 0,
          'Functional': 0,
          'TotRmsAbvGrd': 0,
          'KitchenQual': 0,
          'KitchenAbvGr': 0,
          'BedroomAbvGr': 0,
          'FullBath': 0,
          'MSZoning': 0,
          'BsmtHalfBath': 0,
          'BsmtFullBath': 0,
          'GrLivArea': 0,
          'LowQualFinSF': 0,
          '2ndFlrSF': 0,
          '1stFlrSF': 0,
          'Electrical': 0,
          'GarageType': 0,
          'GarageYrBlt': 0,
          'GarageFinish': 0,
          'GarageCars': 0,
          'SaleCondition': 0,
          'SaleType': 0,
          'YrSold': 0,
          'MoSold': 0,
          'MiscVal': 0,
          'PoolArea': 0,
          'ScreenPorch': 0,
          '3SsnPorch': 0,
          'EnclosedPorch': 0,
          'OpenPorchSF': 0,
          'WoodDeckSF': 0,
          'PavedDrive': 0,
          'GarageCond': 0,
          'GarageQual': 0,
          'GarageArea': 0,
          'CentralAir': 0,
          'HeatingQC': 0,
          'Heating': 0,
          'YearRemodAdd': 0,
          'OverallCond': 0,
          'OverallQual': 0,
          'HouseStyle': 0,
          'BldgType': 0,
          'Condition2': 0,
          'Condition1': 0,
          'Neighborhood': 0,
          'LandSlope': 0,
          'LotConfig': 0,
          'Utilities': 0,
          'LandContour': 0,
          'LotShape': 0,
          'Street': 0,
          'LotArea': 0,
          'LotFrontage': 0,
          'YearBuilt': 0,
```

```
 'RoofStyle': 0,
 'TotalBsmtSF': 0,
 'RoofMatl': 0,
 'BsmtUnfSF': 0,
 'BsmtFinSF2': 0,
 'BsmtFinType2': 0,
 'BsmtFinSF1': 0,
 'BsmtFinType1': 0,
 'BsmtExposure': 0,
 'BsmtCond': 0,
 'BsmtQual': 0,
 'Foundation': 0,
 'ExterCond': 0,
 'ExterQual': 0,
 'MasVnrArea': 0,
 'MasVnrType': 0,
 'Exterior2nd': 0,
 'Exterior1st': 0,
 'SalePrice': 0}
```

Identifying Categorical and Numerical Data

In [13]:

```
#list of categorical variables
categorical_list = ['MSZoning','Street','LotShape','LandContour','Utilities',
'LotConfig','LandSlope','Neighborhood',
 'Condition1','Condition2','BldgType','HouseStyle','RoofStyle','RoofMatl','Ext
erior1st','Exterior2nd','MasVnrType',
 'ExterQual','ExterCond','Foundation','BsmtQual','BsmtCond','BsmtExposure','Bs
mtFinType1','BsmtFinType2','Heating',
 'HeatingQC','CentralAir','Electrical','KitchenQual','Functional','FireplaceQ
u','GarageType','GarageFinish','GarageQual',
 'GarageCond','PavedDrive','SaleType','SaleCondition','YearBuilt','YearRemodAd
d','YrSold']
```

In [14]:

```
#Variables with ordinal values
ordinal_var = ['LotShape','LandContour','Utility','LandSlope','HouseStyle','Ex
terQual','ExterCond','BsmtQual','BsmtCond',
               'BsmtExposure','BsmtFinType1','BsmtFinType2','HeatingQC','Kitch
enQual','Functional','FireplaceQu',
               'GarageFinish','GarageQual','GarageCond','PavedDrive','YearBuil
t','YearRemodAdd','YrSold']

#Variables that have both ordinal and nominal data
mix_var = ['MSZoning','LotConfig','Electrical','SaleType']

#Variables with nominal values
nominal_var = ['Street','Utilities','Neighborhood','Condition1','Condition2',
'BldgType','RoofStyle','RoofMatl',
               'Exterior1st','Exterior2nd','MasVnrType','Foundation','Heating'
,'CentralAir','GarageType','SaleCondition']
```

In [15]:

```
#finding the unique values in each column (type object)

for col in df.select_dtypes('O').columns:
    print('We have {} unique values in {} column : {}'.format(len(df[col].uniq
ue()),col,df[col].unique()))
    print('Value count =',df[col].value_counts())
    print('__'*30)
```

```
We have 5 unique values in MSZoning column : ['RL' 'RM' 'C (all)' 'FV' 'RH']
Value count = RL         1151
RM          218
FV           65
RH           16
C (all)      10
Name: MSZoning, dtype: int64
```

```
We have 2 unique values in Street column : ['Pave' 'Grvl']
Value count = Pave    1454
Grvl       6
Name: Street, dtype: int64
```

```
We have 4 unique values in LotShape column : ['Reg' 'IR1' 'IR2' 'IR3']
Value count = Reg    925
IR1    484
IR2     41
IR3     10
Name: LotShape, dtype: int64
```

```
We have 4 unique values in LandContour column : ['Lvl' 'Bnk' 'Low' 'HLS']
Value count = Lvl    1311
Bnk      63
HLS      50
Low      36
Name: LandContour, dtype: int64
```

```
We have 2 unique values in Utilities column : ['AllPub' 'NoSeWa']
Value count = AllPub    1459
NoSeWa       1
Name: Utilities, dtype: int64
```

```
We have 5 unique values in LotConfig column : ['Inside' 'FR2' 'Corner' 'CulDS
ac' 'FR3']
Value count = Inside     1052
Corner      263
CulDSac      94
FR2          47
FR3           4
Name: LotConfig, dtype: int64
```

```
We have 3 unique values in LandSlope column : ['Gtl' 'Mod' 'Sev']
Value count = Gtl    1382
Mod      65
Sev      13
Name: LandSlope, dtype: int64
```

```
We have 25 unique values in Neighborhood column : ['CollgCr' 'Veenker' 'Crawf
or' 'NoRidge' 'Mitchel' 'Somerst' 'NWAmes'
 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
Value count = NAmes      225
CollgCr    150
OldTown    113
Edwards    100
Somerst     86
```

```
Gilbert     79
NridgHt     77
Sawyer      74
NWAmes      73
SawyerW     59
BrkSide     58
Crawfor     51
Mitchel     49
NoRidge     41
Timber      38
IDOTRR      37
ClearCr     28
StoneBr     25
SWISU       25
MeadowV     17
Blmngtn     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
____________________________________________________________
We have 9 unique values in Condition1 column : ['Norm' 'Feedr' 'PosN' 'Arter
y' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
Value count = Norm      1260
Feedr       81
Artery      48
RRAn        26
PosN        19
RRAe        11
PosA         8
RRNn         5
RRNe         2
Name: Condition1, dtype: int64
____________________________________________________________
We have 8 unique values in Condition2 column : ['Norm' 'Artery' 'RRNn' 'Feed
r' 'PosN' 'PosA' 'RRAn' 'RRAe']
Value count = Norm      1445
Feedr        6
PosN         2
RRNn         2
Artery       2
RRAn         1
RRAe         1
PosA         1
Name: Condition2, dtype: int64
____________________________________________________________
We have 5 unique values in BldgType column : ['1Fam' '2fmCon' 'Duplex' 'Twnhs
E' 'Twnhs']
Value count = 1Fam      1220
TwnhsE     114
Duplex      52
Twnhs       43
2fmCon      31
Name: BldgType, dtype: int64
____________________________________________________________
We have 8 unique values in HouseStyle column : ['2Story' '1Story' '1.5Fin'
```

```
 '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf' '2.5Fin']
Value count = 1Story    726
2Story    445
1.5Fin    154
SLvl       65
SFoyer     37
1.5Unf     14
2.5Unf     11
2.5Fin      8
Name: HouseStyle, dtype: int64
____________________________________________________________
We have 6 unique values in RoofStyle column : ['Gable' 'Hip' 'Gambrel' 'Mansa
rd' 'Flat' 'Shed']
Value count = Gable      1141
Hip         286
Flat         13
Gambrel      11
Mansard       7
Shed          2
Name: RoofStyle, dtype: int64
____________________________________________________________
We have 8 unique values in RoofMatl column : ['CompShg' 'WdShngl' 'Metal' 'Wd
Shake' 'Membran' 'Tar&Grv' 'Roll'
 'ClyTile']
Value count = CompShg    1434
Tar&Grv      11
WdShngl       6
WdShake       5
Metal         1
ClyTile       1
Membran       1
Roll          1
Name: RoofMatl, dtype: int64
____________________________________________________________
We have 15 unique values in Exterior1st column : ['VinylSd' 'MetalSd' 'Wd Sdn
g' 'HdBoard' 'BrkFace' 'WdShing' 'CemntBd'
 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
 'CBlock']
Value count = VinylSd    515
HdBoard    222
MetalSd    220
Wd Sdng    206
Plywood    108
CemntBd     61
BrkFace     50
WdShing     26
Stucco      25
AsbShng     20
BrkComm      2
Stone        2
AsphShn      1
CBlock       1
ImStucc      1
Name: Exterior1st, dtype: int64
____________________________________________________________
We have 16 unique values in Exterior2nd column : ['VinylSd' 'MetalSd' 'Wd Shn
g' 'HdBoard' 'Plywood' 'Wd Sdng' 'CmentBd'
```

```
 'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone'
 'Other' 'CBlock']
Value count = VinylSd    504
MetalSd    214
HdBoard    207
Wd Sdng    197
Plywood    142
CmentBd     60
Wd Shng     38
Stucco      26
BrkFace     25
AsbShng     20
ImStucc     10
Brk Cmn      7
Stone        5
AsphShn      3
CBlock       1
Other        1
Name: Exterior2nd, dtype: int64
```

```
We have 4 unique values in MasVnrType column : ['BrkFace' 'None' 'Stone' 'Brk
Cmn']
Value count = None       872
BrkFace    445
Stone      128
BrkCmn      15
Name: MasVnrType, dtype: int64
```

```
We have 4 unique values in ExterQual column : ['Gd' 'TA' 'Ex' 'Fa']
Value count = TA    906
Gd    488
Ex     52
Fa     14
Name: ExterQual, dtype: int64
```

```
We have 5 unique values in ExterCond column : ['TA' 'Gd' 'Fa' 'Po' 'Ex']
Value count = TA    1282
Gd     146
Fa      28
Ex       3
Po       1
Name: ExterCond, dtype: int64
```

```
We have 6 unique values in Foundation column : ['PConc' 'CBlock' 'BrkTil' 'Wo
od' 'Slab' 'Stone']
Value count = PConc     647
CBlock    634
BrkTil    146
Slab       24
Stone       6
Wood        3
Name: Foundation, dtype: int64
```

```
We have 4 unique values in BsmtQual column : ['Gd' 'TA' 'Ex' 'Fa']
Value count = TA    686
Gd    618
Ex    121
```

```
Fa     35
Name: BsmtQual, dtype: int64
```

```
We have 4 unique values in BsmtCond column : ['TA' 'Gd' 'Fa' 'Po']
Value count = TA    1348
Gd      65
Fa      45
Po       2
Name: BsmtCond, dtype: int64
```

```
We have 4 unique values in BsmtExposure column : ['No' 'Gd' 'Mn' 'Av']
Value count = No    991
Av    221
Gd    134
Mn    114
Name: BsmtExposure, dtype: int64
```

```
We have 6 unique values in BsmtFinType1 column : ['GLQ' 'ALQ' 'Unf' 'Rec' 'BL
Q' 'LwQ']
Value count = Unf    467
GLQ    418
ALQ    220
BLQ    148
Rec    133
LwQ     74
Name: BsmtFinType1, dtype: int64
```

```
We have 6 unique values in BsmtFinType2 column : ['Unf' 'BLQ' 'ALQ' 'Rec' 'Lw
Q' 'GLQ']
Value count = Unf    1294
Rec      54
LwQ      46
BLQ      33
ALQ      19
GLQ      14
Name: BsmtFinType2, dtype: int64
```

```
We have 6 unique values in Heating column : ['GasA' 'GasW' 'Grav' 'Wall' 'Oth
W' 'Floor']
Value count = GasA     1428
GasW       18
Grav        7
Wall        4
OthW        2
Floor       1
Name: Heating, dtype: int64
```

```
We have 5 unique values in HeatingQC column : ['Ex' 'Gd' 'TA' 'Fa' 'Po']
Value count = Ex    741
TA    428
Gd    241
Fa     49
Po      1
Name: HeatingQC, dtype: int64
```

```
We have 2 unique values in CentralAir column : ['Y' 'N']
Value count = Y    1365
```

```
N      95
Name: CentralAir, dtype: int64
```

```
We have 5 unique values in Electrical column : ['SBrkr' 'FuseF' 'FuseA' 'Fuse
P' 'Mix']
Value count = SBrkr    1335
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: Electrical, dtype: int64
```

```
We have 4 unique values in KitchenQual column : ['Gd' 'TA' 'Ex' 'Fa']
Value count = TA    735
Gd    586
Ex    100
Fa     39
Name: KitchenQual, dtype: int64
```

```
We have 7 unique values in Functional column : ['Typ' 'Min1' 'Maj1' 'Min2' 'M
od' 'Maj2' 'Sev']
Value count = Typ     1360
Min2      34
Min1      31
Mod       15
Maj1      14
Maj2       5
Sev        1
Name: Functional, dtype: int64
```

```
We have 6 unique values in FireplaceQu column : ['None' 'TA' 'Gd' 'Fa' 'Ex'
'Po']
Value count = None    690
Gd      380
TA      313
Fa       33
Ex       24
Po       20
Name: FireplaceQu, dtype: int64
```

```
We have 6 unique values in GarageType column : ['Attchd' 'Detchd' 'BuiltIn'
'CarPort' 'Basment' '2Types']
Value count = Attchd     951
Detchd     387
BuiltIn     88
Basment     19
CarPort      9
2Types       6
Name: GarageType, dtype: int64
```

```
We have 3 unique values in GarageFinish column : ['RFn' 'Unf' 'Fin']
Value count = Unf    686
RFn    422
Fin    352
Name: GarageFinish, dtype: int64
```

```
We have 5 unique values in GarageQual column : ['TA' 'Fa' 'Gd' 'Ex' 'Po']
```

```
Value count = TA    1392
Fa      48
Gd      14
Ex       3
Po       3
Name: GarageQual, dtype: int64
____________________________________________________________
We have 5 unique values in GarageCond column : ['TA' 'Fa' 'Gd' 'Po' 'Ex']
Value count = TA    1407
Fa      35
Gd       9
Po       7
Ex       2
Name: GarageCond, dtype: int64
____________________________________________________________
We have 3 unique values in PavedDrive column : ['Y' 'N' 'P']
Value count = Y    1340
N      90
P      30
Name: PavedDrive, dtype: int64
____________________________________________________________
We have 9 unique values in SaleType column : ['WD' 'New' 'COD' 'ConLD' 'ConL
I' 'CWD' 'ConLw' 'Con' 'Oth']
Value count = WD       1267
New       122
COD        43
ConLD       9
ConLI       5
ConLw       5
CWD         4
Oth         3
Con         2
Name: SaleType, dtype: int64
____________________________________________________________
We have 6 unique values in SaleCondition column : ['Normal' 'Abnorml' 'Partia
l' 'AdjLand' 'Alloca' 'Family']
Value count = Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
____________________________________________________________
```

In [16]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [17]:
```python
for column in df.columns:
    if column in categorical_list:
        labelencoder = LabelEncoder()
        df[column] = labelencoder.fit_transform(df[column])
```

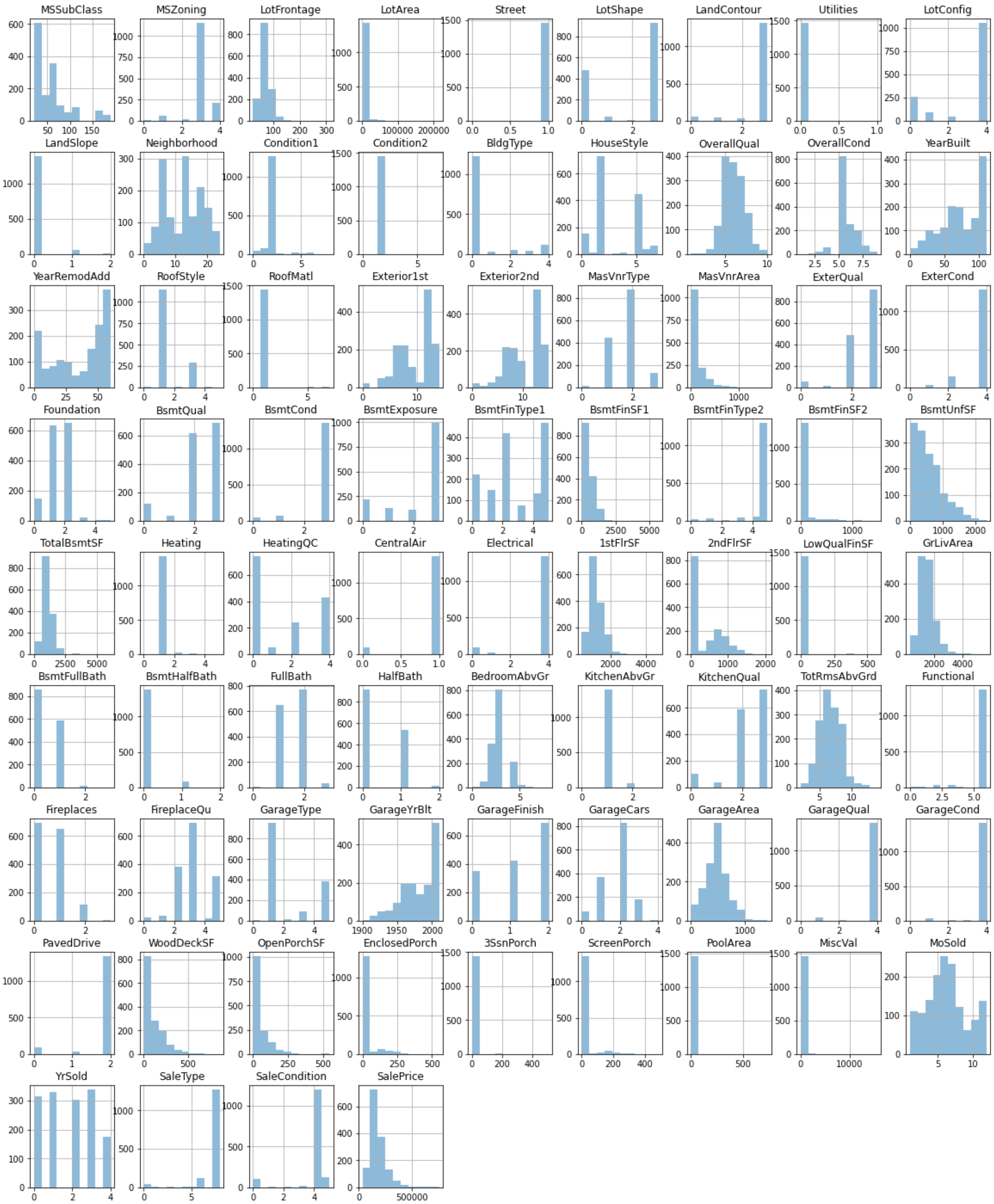
In [18]:
```
numeric_list = []
for i in df.columns:
    if i not in categorical_list:
        numeric_list.append(i)
numeric_list
```

Out[18]:
```
['MSSubClass',
 'LotFrontage',
 'LotArea',
 'OverallQual',
 'OverallCond',
 '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',
 'SalePrice']
```

Checking for Skewness in numerical columns

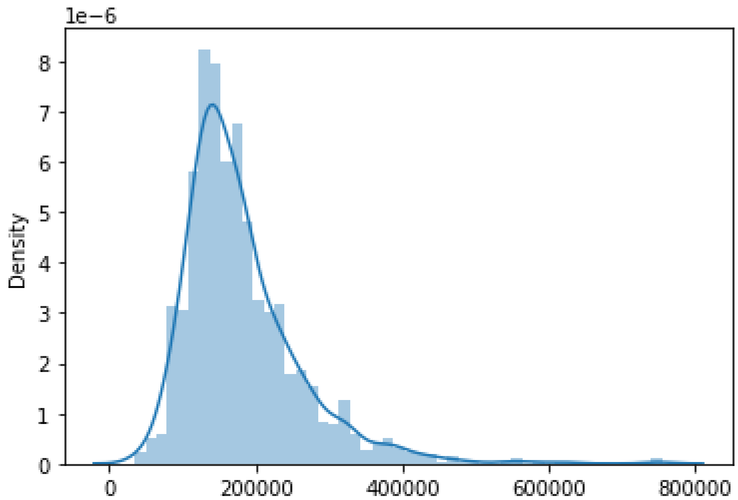
In [19]:

```
df.hist(alpha=0.5, figsize=(20,25))
plt.show()
```



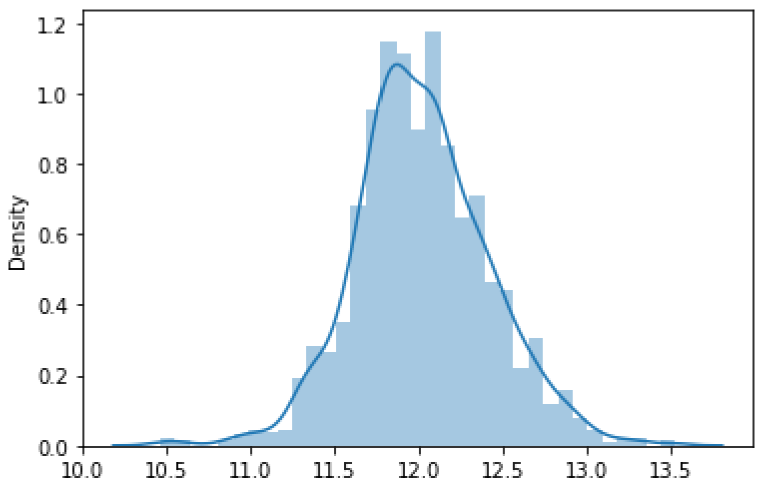
In [20]:

```python
try:
    sns.distplot(df['SalePrice'] , fit='norm');
    (mu, sigma) = norm.fit(df['SalePrice'])
    print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))
    plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu
, sigma)],
            loc='best')
    plt.ylabel('Frequency')
    plt.title('SalePrice distribution')
    fig = plt.figure()
    res = stats.probplot(df['SalePrice'], plot=plt)
    plt.show()
except AttributeError:
    pass
```

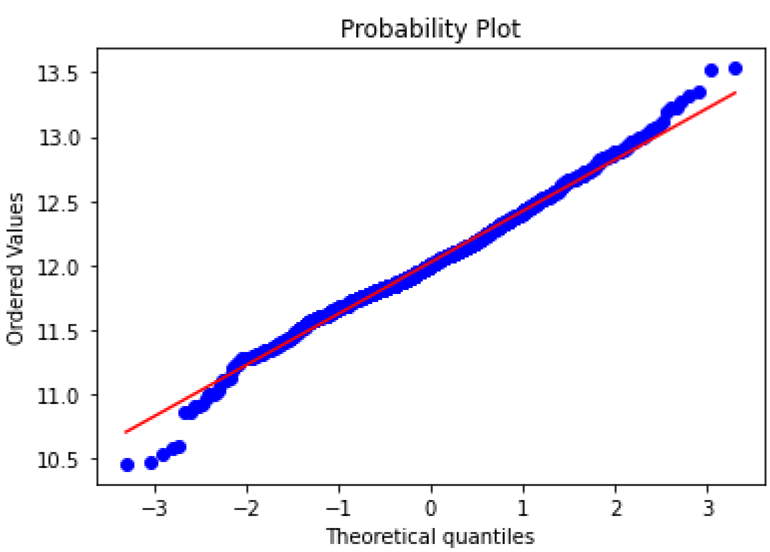


The target variable is right skewed. As (linear) models love normally distributed data , we need to transform this variable and make it more normally distributed.

In [21]:
```
try :
    #We use the numpy fuction log1p which  applies log(1+x) to all elements of
the column
    df["SalePrice"] = np.log1p(df["SalePrice"])
    sns.distplot(df['SalePrice'] , fit='norm');
    (mu, sigma) = norm.fit(df['SalePrice'])
    print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))
    plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu
, sigma)],
            loc='best')
    plt.ylabel('Frequency')
    plt.title('SalePrice distribution')
except AttributeError:
    pass
```



In [22]:
```
#Get also the QQ-plot
fig = plt.figure()
res = stats.probplot(df['SalePrice'], plot=plt)
plt.show()
```



In [23]:

```
num = df.dtypes[df.dtypes != "object"].index
num= num.drop(['SalePrice'])
# Check the skew of all numerical features
skewed_feats = df[numeric_list].apply(lambda x: skew(x)).sort_values(ascending
=False)
print("\nSkew in numeric features: \n")
skewness = pd.DataFrame({'Skew' :skewed_feats})
skewness
```

Skew in numeric features:

Out[23]:

	Skew
MiscVal	24.451640
PoolArea	14.813135
LotArea	12.195142
3SsnPorch	10.293752
LowQualFinSF	9.002080
KitchenAbvGr	4.483784
BsmtFinSF2	4.250888
ScreenPorch	4.117977
BsmtHalfBath	4.099186
EnclosedPorch	3.086696
MasVnrArea	2.674865
LotFrontage	2.383704
OpenPorchSF	2.361912
BsmtFinSF1	1.683771
WoodDeckSF	1.539792
TotalBsmtSF	1.522688
MSSubClass	1.406210
1stFlrSF	1.375342
GrLivArea	1.365156
BsmtUnfSF	0.919323
2ndFlrSF	0.812194
OverallCond	0.692355
TotRmsAbvGrd	0.675646
HalfBath	0.675203
Fireplaces	0.648898
BsmtFullBath	0.595454
OverallQual	0.216721
MoSold	0.211835
BedroomAbvGr	0.211572
GarageArea	0.179796
SalePrice	0.121222
FullBath	0.036524
GarageCars	-0.342197
GarageYrBlt	-0.718552

In [24]:
```
skewness = skewness[abs(skewness) > 0.75]
print("There are {} skewed features for Box Cox transformation".format(skewnes
s.shape[0]))
skewed_features = skewness.index
lam = 0.15
for i in skewed_features:
    df[i] = boxcox1p(df[i], lam)
df[skewed_features] = np.log1p(df[skewed_features])
```

```
There are 34 skewed features for Box Cox transformation
```

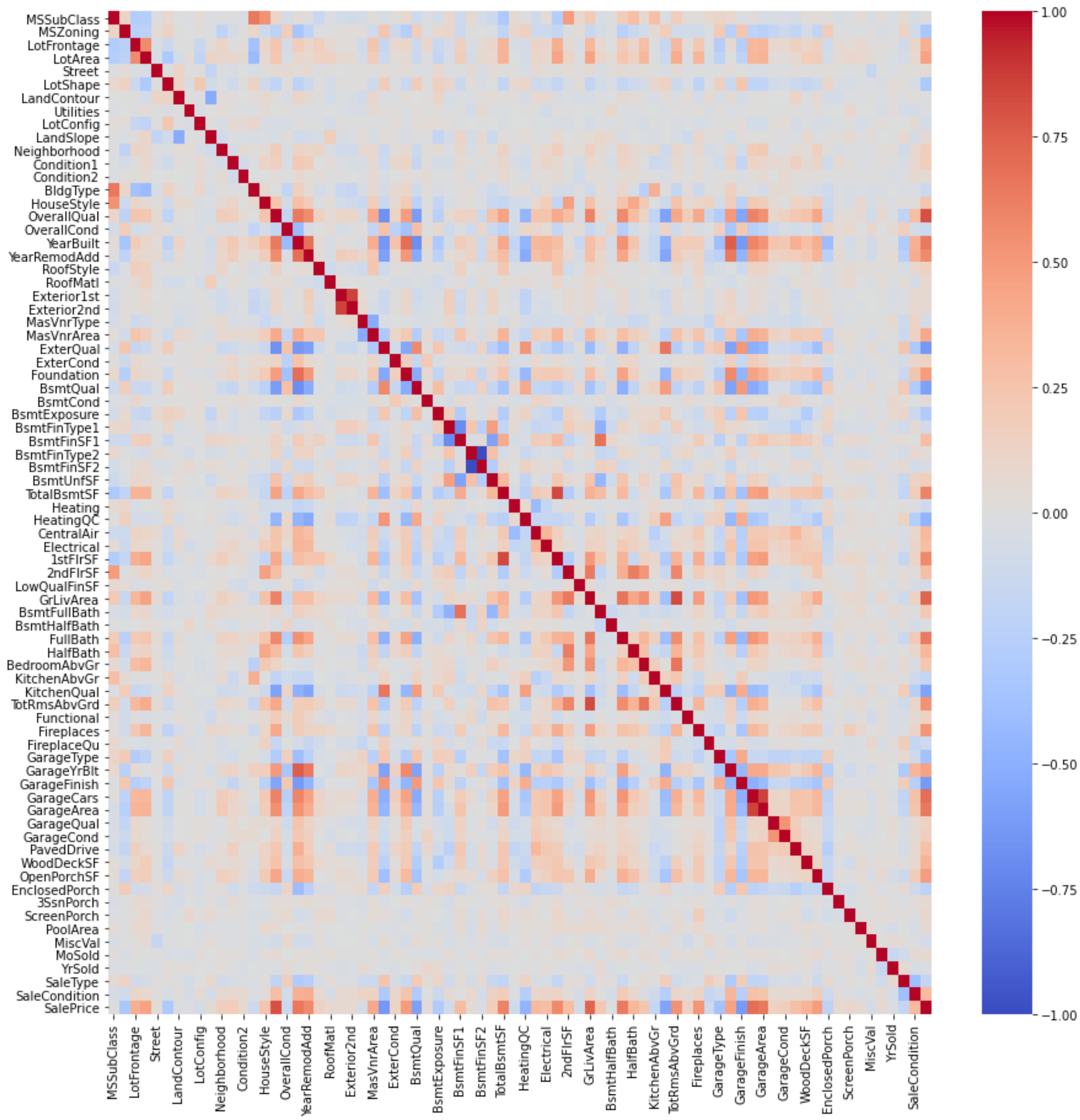
Finding Correlations in Data

In [25]:
```
corr = df.corr(method ='spearman')
```

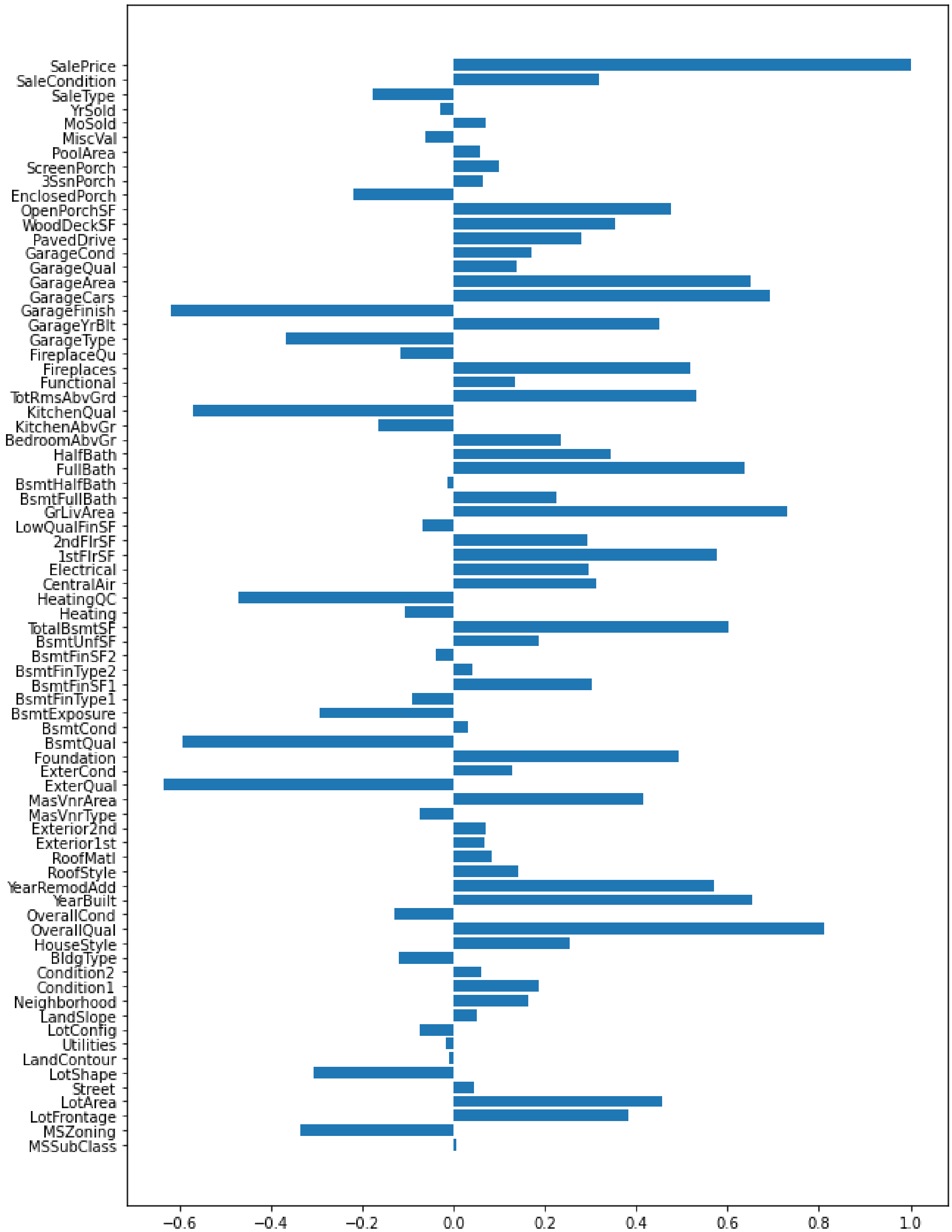
Through Spearman method of correlation we observe the highest positive correlation for OverallQual. Taken this method of correlation as spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. The Spearman rank correlation test does not carry any assumptions about the distribution of the data and is the appropriate correlation analysis when the variables are measured on a scale that is at least ordinal.

In [43]:
```
plt.figure(figsize=(15,15))
sns.heatmap(corr,cmap='coolwarm',vmin=-1,vmax=1,center=0)
```

Out[43]: <AxesSubplot:>



In [27]:
```
plt.figure(figsize=(10,15))
plt.barh(corr.columns,corr['SalePrice'])
plt.show()
```



In [28]:
```
positive_index = corr['SalePrice'].sort_values(ascending=False)[1:10].index
corr['SalePrice'].sort_values(ascending=False)[1:15].to_frame()\
.style.background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True
))
```

Out[28]:

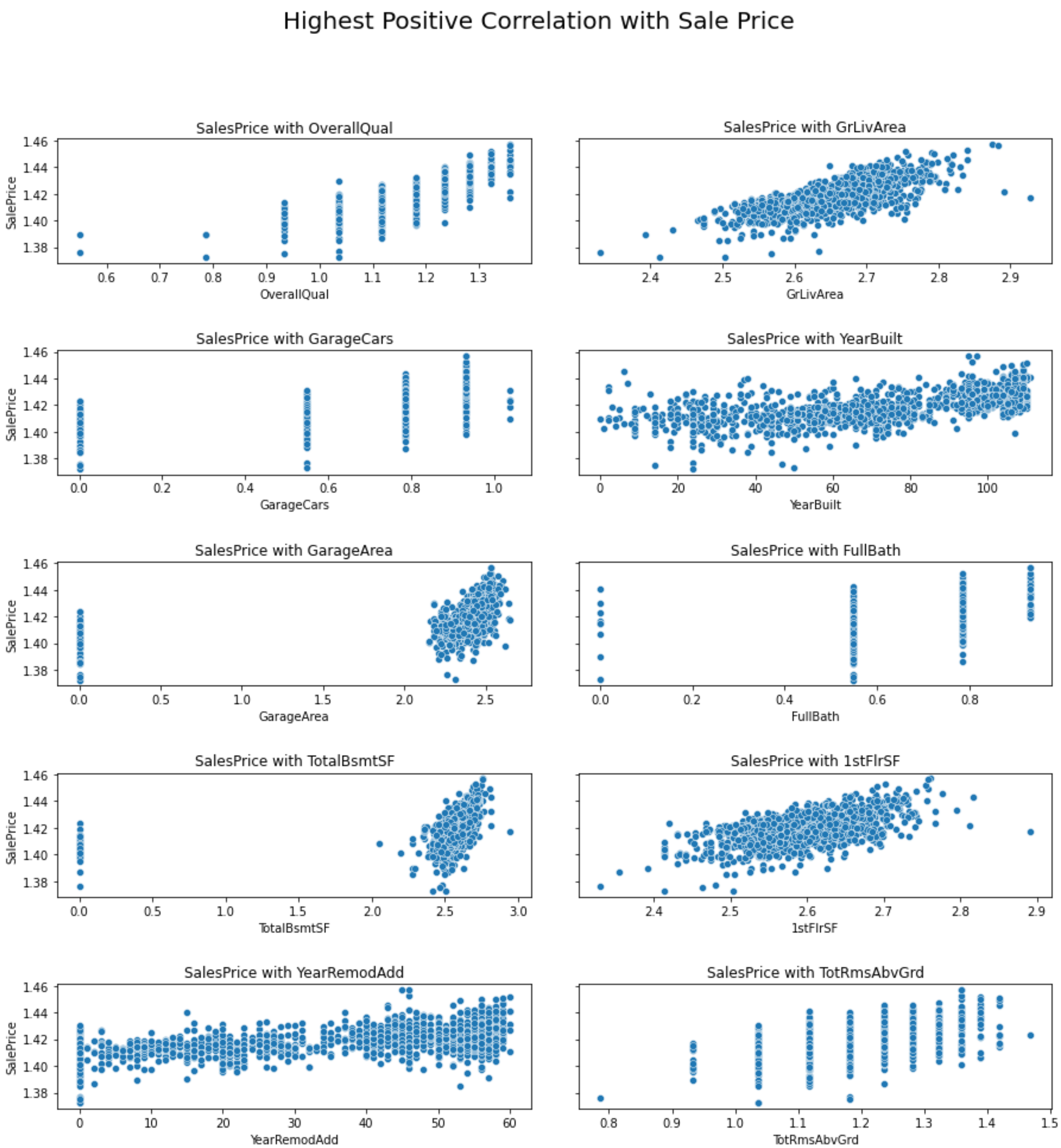
	SalePrice
OverallQual	0.809829
GrLivArea	0.731310
GarageCars	0.690711
YearBuilt	0.652682
GarageArea	0.649379
FullBath	0.635957
TotalBsmtSF	0.602725
1stFlrSF	0.575408
YearRemodAdd	0.571159
TotRmsAbvGrd	0.532586
Fireplaces	0.519247
Foundation	0.491932
OpenPorchSF	0.477561
LotArea	0.456461

In [29]:
```
#Visualize columns have positive corr with SalePrice

high_corr = corr['SalePrice'].sort_values(ascending=False)[1:][:10].index.toli
st()

fig, axes = plt.subplots(5,2, figsize=(15, 15), sharey=True);
plt.subplots_adjust(hspace = 0.7, wspace=0.1)
fig.suptitle('Highest Positive Correlation with Sale Price', fontsize=20);

for i,col in zip(range(10),high_corr):
    sns.scatterplot(y=df['SalePrice'], x=df[col], ax=axes[i//2][i%2])
    axes[i//2][i%2].set_title('SalesPrice with '+col)
```



In [30]:

```
negative_index = corr['SalePrice'].sort_values(ascending=True)[1:10]
corr['SalePrice'].sort_values(ascending=True)[1:10].to_frame()\
.style.background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True
))
```

Out[30]:

	SalePrice
GarageFinish	-0.617556
BsmtQual	-0.591242
KitchenQual	-0.569857
HeatingQC	-0.471338
GarageType	-0.368480
MSZoning	-0.334909
LotShape	-0.305923
BsmtExposure	-0.292192
EnclosedPorch	-0.218394

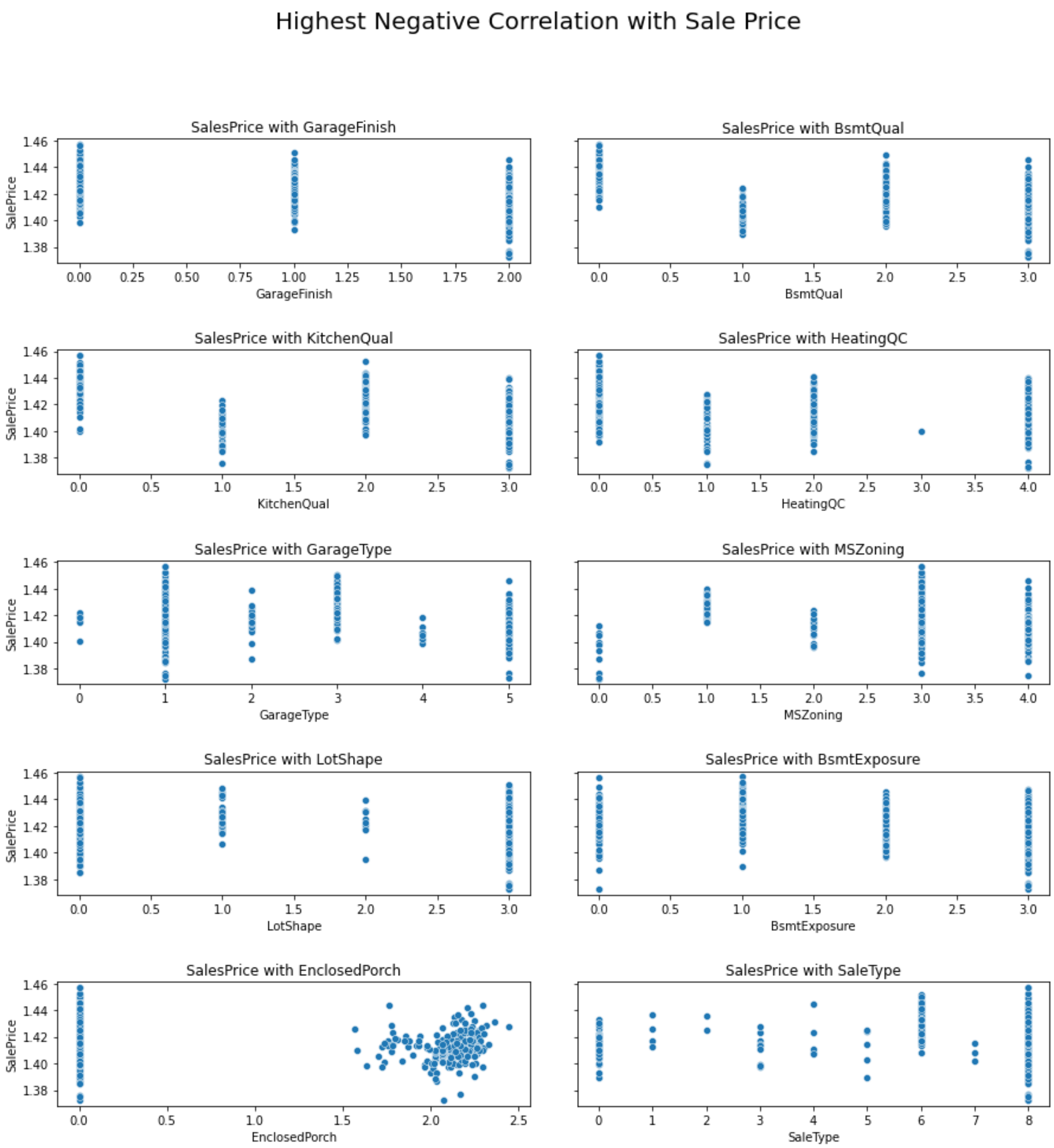
In [31]:

```
#Visualize columns with negative corr with SalePrice

high_corr = corr['SalePrice'].sort_values(ascending=True)[1:][:10].index.tolis
t()

fig, axes = plt.subplots(5,2, figsize=(15, 15), sharey=True);
plt.subplots_adjust(hspace = 0.7, wspace=0.1)
fig.suptitle('Highest Negative Correlation with Sale Price', fontsize=20);

for i,col in zip(range(10),high_corr):
    sns.scatterplot(y=df['SalePrice'], x=df[col], ax=axes[i//2][i%2])
    axes[i//2][i%2].set_title('SalesPrice with '+col)
```



Future Plans

- Remove Outliers
- Use better Encoding for ordinal data
- Normalise data
- Find new correlation and important feature using machine learning algorithms
- Optimise notebook for better performance is second_contribution

Observations

- The highest positive correlation has been found for OverallQual 0.809829
- The highest nnegative correlation is found for GarageFinish -0.617556
- The correlation has been calculated using the filter method. Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally costly, and in the case of massive datasets, wrapper methods are not the most effective feature selection method to consider.