In [ ]:
```
import pandas
import seaborn
import matplotlib.pyplot as plt
import numpy
from scipy import stats
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.feature_selection import RFE,VarianceThreshold
from sklearn.linear_model import LinearRegression,Ridge,Lasso
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import r2_score,mean_squared_error
from decimal import Decimal
import warnings
warnings.filterwarnings('ignore')
```

Data Exploration

In [ ]:
```
housing_raw_df=pandas.read_csv('/Users/karanprinja/Downloads/train.csv')
```

In [ ]:
```
housing_raw_df.head()
```

Out[ ]:

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

5 rows × 81 columns

In [ ]:
```
housing_raw_df.describe()
```

Out[ ]:

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

8 rows × 38 columns

In [ ]:
```
housing_raw_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
```

Dropping Id column as it only serial numbers

In [ ]:
```
housing_raw_df.drop(columns='Id',axis=1,inplace=True)
```

Checking the null value count in columns

In [ ]:
```
null_precent_cols=(housing_raw_df.isna().sum()/len(housing_raw_df.index))*10
null_columns=list(null_precent_cols[null_precent_cols>5].index)
null_columns
```

Out[ ]:
```
['LotFrontage',
 'Alley',
 'FireplaceQu',
 'GarageType',
 'GarageYrBlt',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PoolQC',
 'Fence',
 'MiscFeature']
```

Dropping the columns which have more than 5% Null values

In [ ]:
```
for col in null_columns:
    housing_raw_df.drop(columns=col,axis=1,inplace=True)
```

In [ ]:
```
housing_raw_df.dtypes.value_counts()
```

Out[ ]:
```
int64      34
object     34
float64     1
dtype: int64
```

Collecting the numerical and categorical columns and filling their NA values. Filling NA values for Numerical columns with median and for Categorical columns we make use of Mode

In [ ]:
```
numerical_columns=housing_raw_df.select_dtypes(include=['int64','float64'])
```

In [ ]:
```
numerical_columns.head()
```

Out[ ]:

	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea
0	60	8450	7	5	2003	2003	196.0
1	20	9600	6	8	1976	1976	0.0
2	60	11250	7	5	2001	2002	162.0
3	70	9550	7	5	1915	1970	0.0
4	60	14260	8	5	2000	2000	350.0

5 rows × 35 columns

In [ ]:
```
for columns in numerical_columns.columns:
    median_val=housing_raw_df[columns].median()
    print("Filling for column {} with median value {}".format(columns,median
    housing_raw_df[columns]=housing_raw_df[columns].fillna(value=median_val)
```

```
Filling for column MSSubClass with median value 50.0
Filling for column LotArea with median value 9478.5
Filling for column OverallQual with median value 6.0
Filling for column OverallCond with median value 5.0
Filling for column YearBuilt with median value 1973.0
Filling for column YearRemodAdd with median value 1994.0
Filling for column MasVnrArea with median value 0.0
Filling for column BsmtFinSF1 with median value 383.5
Filling for column BsmtFinSF2 with median value 0.0
Filling for column BsmtUnfSF with median value 477.5
Filling for column TotalBsmtSF with median value 991.5
Filling for column 1stFlrSF with median value 1087.0
Filling for column 2ndFlrSF with median value 0.0
Filling for column LowQualFinSF with median value 0.0
Filling for column GrLivArea with median value 1464.0
Filling for column BsmtFullBath with median value 0.0
Filling for column BsmtHalfBath with median value 0.0
Filling for column FullBath with median value 2.0
Filling for column HalfBath with median value 0.0
Filling for column BedroomAbvGr with median value 3.0
Filling for column KitchenAbvGr with median value 1.0
Filling for column TotRmsAbvGrd with median value 6.0
Filling for column Fireplaces with median value 1.0
Filling for column GarageCars with median value 2.0
Filling for column GarageArea with median value 480.0
Filling for column WoodDeckSF with median value 0.0
Filling for column OpenPorchSF with median value 25.0
Filling for column EnclosedPorch with median value 0.0
Filling for column 3SsnPorch with median value 0.0
Filling for column ScreenPorch with median value 0.0
Filling for column PoolArea with median value 0.0
Filling for column MiscVal with median value 0.0
Filling for column MoSold with median value 6.0
Filling for column YrSold with median value 2008.0
Filling for column SalePrice with median value 163000.0
```

In [ ]:
```
categorical_columns=housing_raw_df.select_dtypes(include=['object'])
```

In [ ]:
```
categorical_columns.head()
```

Out[ ]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood
0	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	CollgCr
1	RL	Pave	Reg	Lvl	AllPub	FR2	Gtl	Veenker
2	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	CollgCr
3	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	Crawfor
4	RL	Pave	IR1	Lvl	AllPub	FR2	Gtl	NoRidge

5 rows × 34 columns

Filling Columns with None where NA values makes sense

In [ ]:
```
null_with_meaning = [ "MasVnrType", "BsmtQual", "BsmtCond", "BsmtExposure",
for col in null_with_meaning:
    housing_raw_df[col].fillna("none", inplace=True)
```

In [ ]:
```
remaining_col_without_meaning=set(categorical_columns.columns)-set(null_with
```

In [ ]:
```
for i in remaining_col_without_meaning:
    m_value=housing_raw_df[i].mode()
    print("Taking mode value {} for column {}".format(m_value[0],i))
    housing_raw_df[i]=housing_raw_df[i].fillna(value=m_value[0])
```

```
Taking mode value NAmes for column Neighborhood
Taking mode value Inside for column LotConfig
Taking mode value Norm for column Condition1
Taking mode value Y for column CentralAir
Taking mode value TA for column ExterCond
Taking mode value Gtl for column LandSlope
Taking mode value WD for column SaleType
Taking mode value PConc for column Foundation
Taking mode value VinylSd for column Exterior2nd
Taking mode value Pave for column Street
Taking mode value Normal for column SaleCondition
Taking mode value 1Story for column HouseStyle
Taking mode value TA for column KitchenQual
Taking mode value VinylSd for column Exterior1st
Taking mode value Reg for column LotShape
Taking mode value TA for column ExterQual
Taking mode value Typ for column Functional
Taking mode value Lvl for column LandContour
Taking mode value Gable for column RoofStyle
Taking mode value GasA for column Heating
Taking mode value Ex for column HeatingQC
Taking mode value Y for column PavedDrive
Taking mode value CompShg for column RoofMatl
Taking mode value 1Fam for column BldgType
Taking mode value Norm for column Condition2
Taking mode value AllPub for column Utilities
Taking mode value SBrkr for column Electrical
Taking mode value RL for column MSZoning
```

Checking to see if any column has Null Value

In [ ]:
```
housing_raw_df.columns[housing_raw_df.isnull().any()]

null_2 = housing_raw_df.isnull().sum()/len(housing_raw_df)*100
null_2 = null_2[null_2>0]
null_2.sort_values(inplace=True, ascending=False)
null_2
```

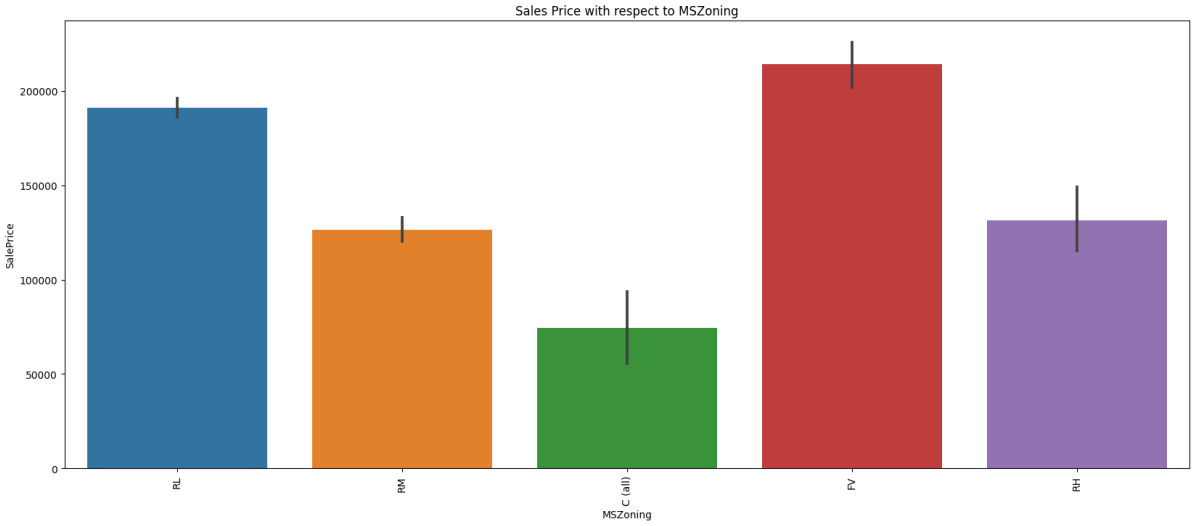
Out[ ]:
```
Series([], dtype: float64)
```

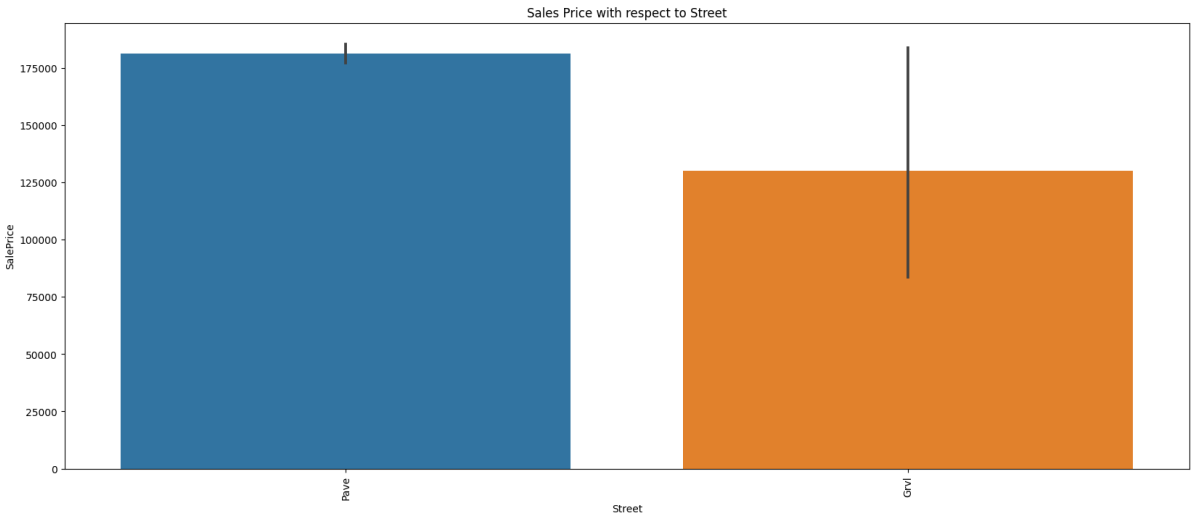
Exploratory Data Analysis

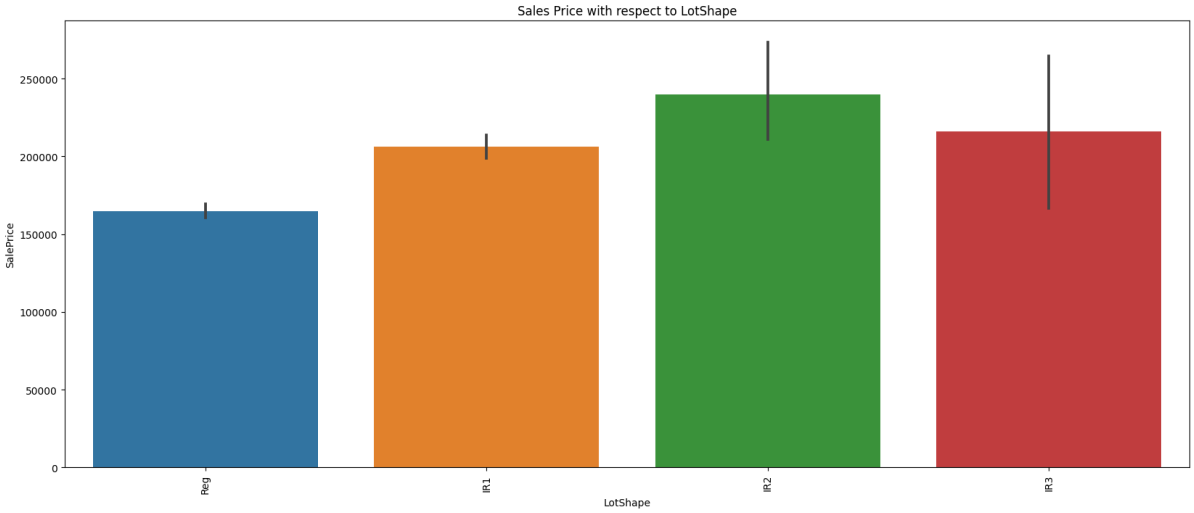
Categorical Graphs

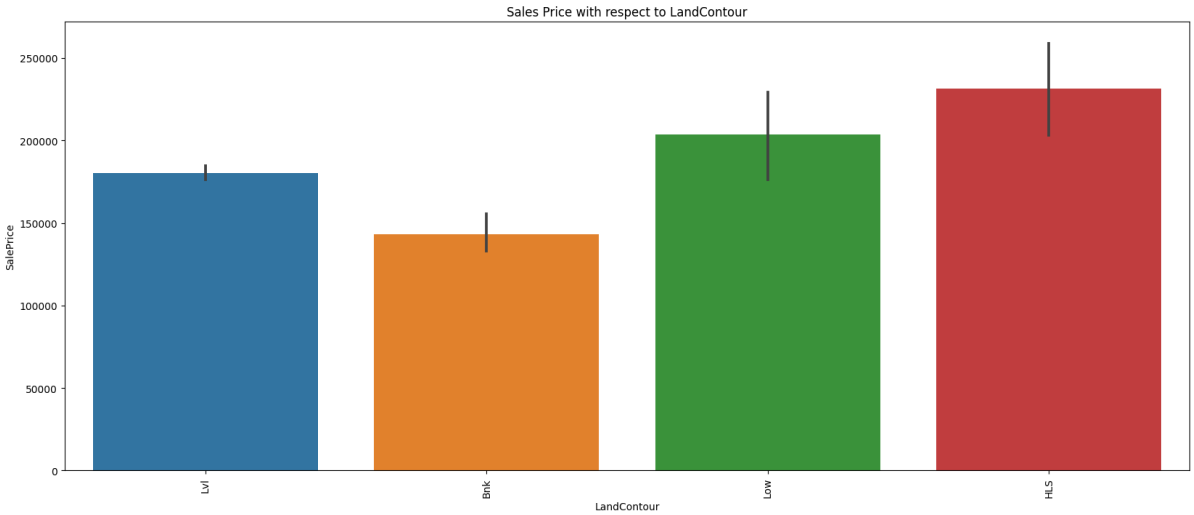
In [ ]:
```
def categorial_columns_graph(columns):
    plt.figure(figsize=(20, 8))
    seaborn.barplot(x=columns, y="SalePrice", data= housing_raw_df)
    plt.title("Sales Price with respect to {}".format(columns))
    plt.xticks(rotation=90)
```

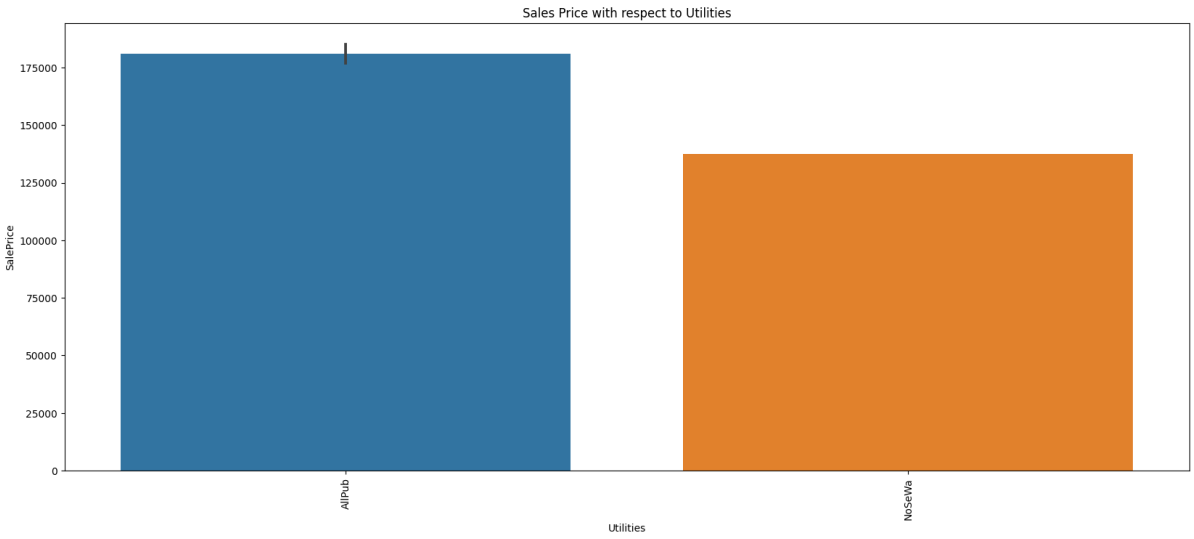
In [ ]:
```
for x in categorical_columns.columns:
    categorial_columns_graph(x)
```

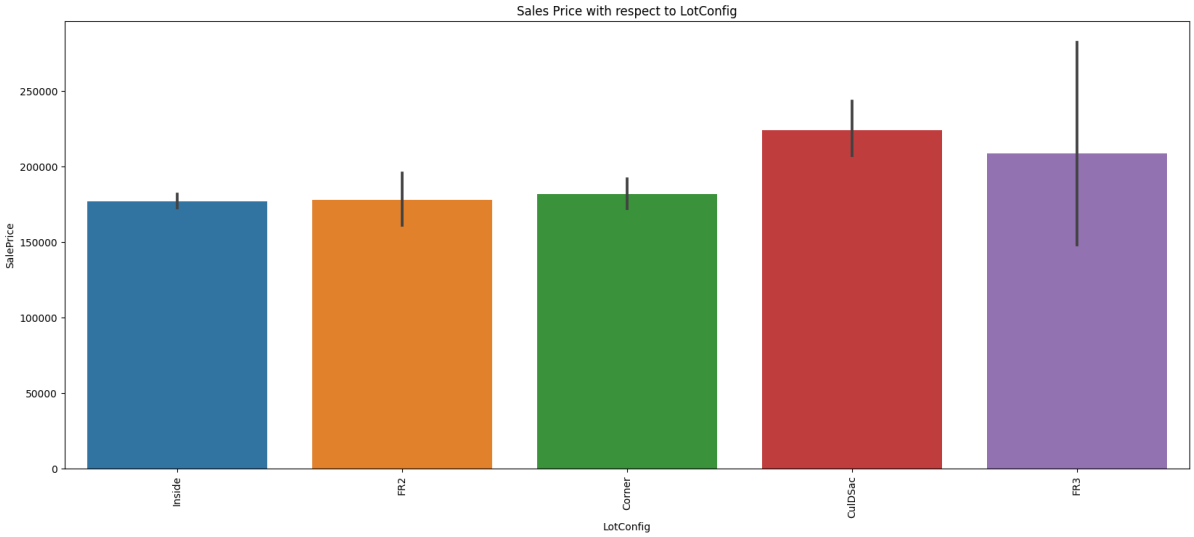


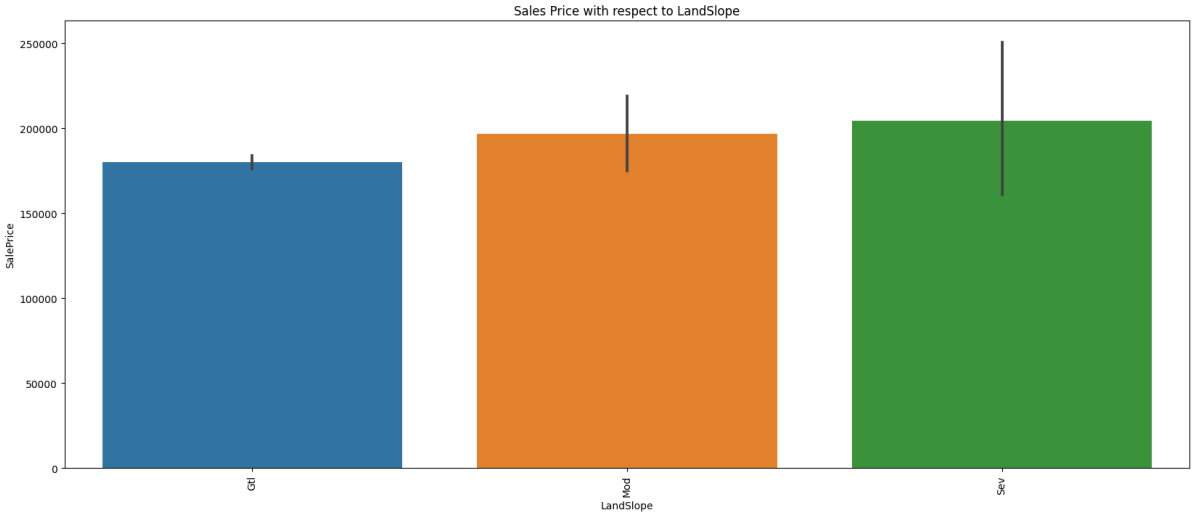


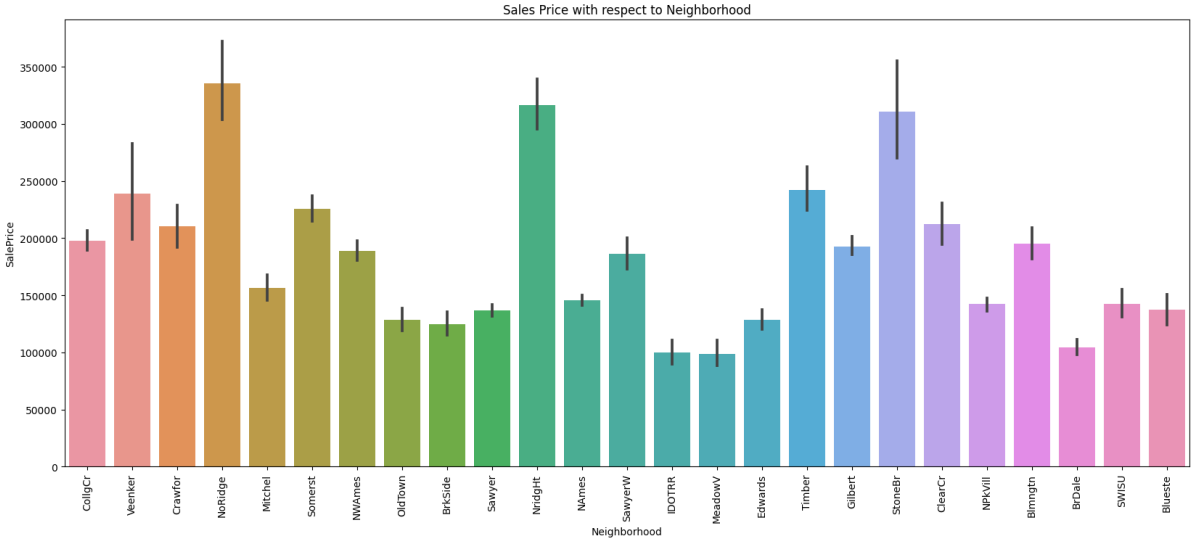

Sales Price with respect to LotShape
SalePrice
0
50000
100000
150000
200000
250000
Reg
IR1
IR2
IR3
LotShape

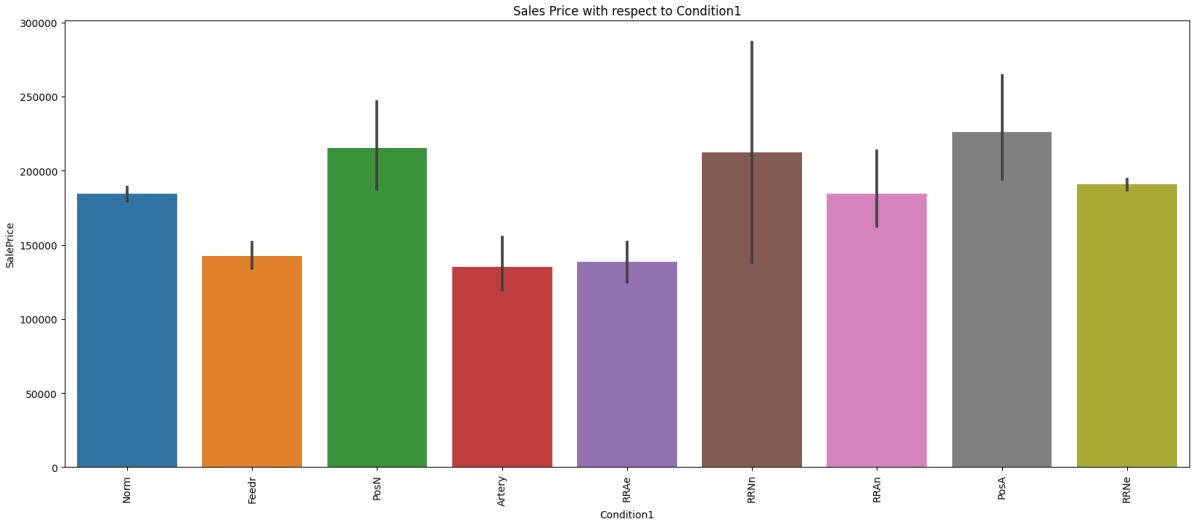

Sales Price with respect to LandContour
SalePrice
0
50000
100000
150000
200000
250000
Lvl
Bnk
Low
HLS
LandContour

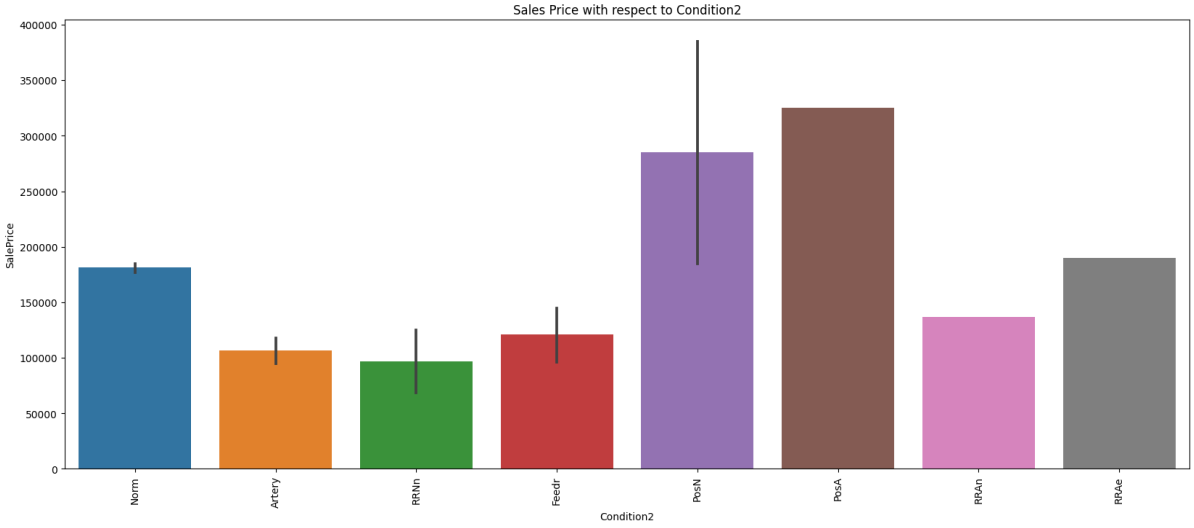

Sales Price with respect to Utilities
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
AllPub
NoSeWa
Utilities

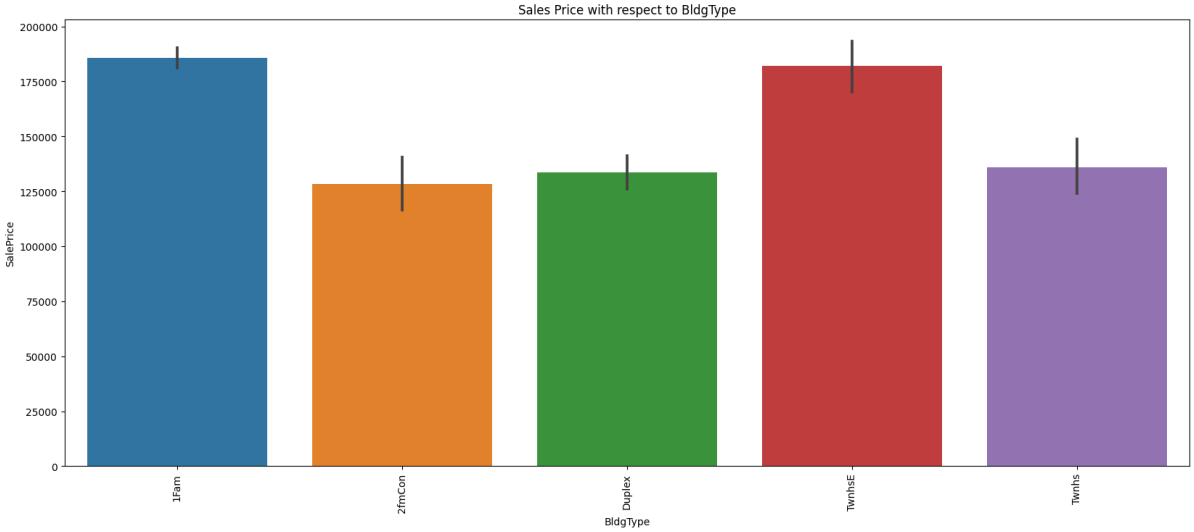

Sales Price with respect to LotConfig
SalePrice
0
50000
100000
150000
200000
250000
Inside
FR2
Corner
CulDSac
FR3
LotConfig

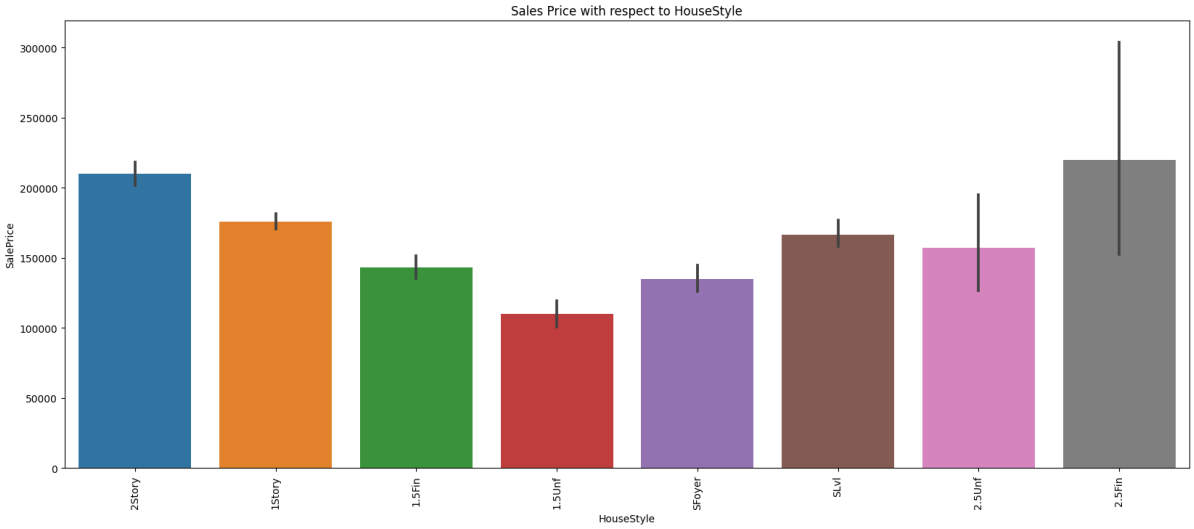

Sales Price with respect to LandSlope
SalePrice
0
50000
100000
150000
200000
250000
Gtl
Mod
Sev
LandSlope

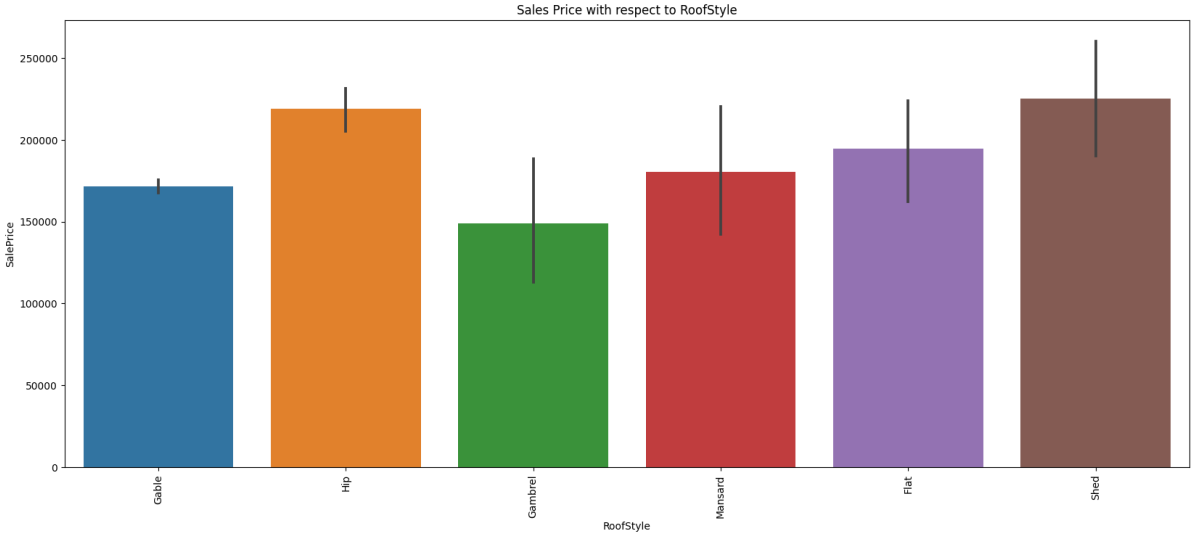

Sales Price with respect to Neighborhood
SalePrice
0
50000
100000
150000
200000
250000
300000
350000
CollgCr
Veenker
Crawfor
NoRidge
Mitchel
Somerst
NWAmes
OldTown
BrkSide
Sawyer
NridgHt
NAmes
SawyerW
IDOTRR
MeadowV
Edwards
Timber
Gilbert
StoneBr
ClearCr
NPkVill
Blmngtn
BrDale
SWISU
Blueste
Neighborhood

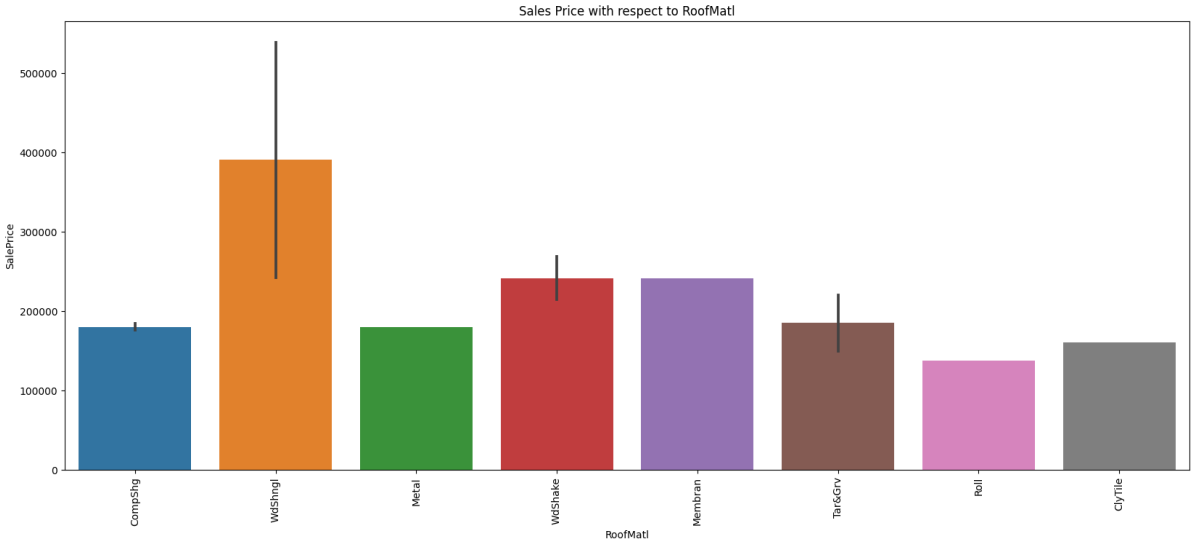

Sales Price with respect to Condition1
SalePrice
Condition1
Norm
Feedr
PosN
Artery
RRAe
RRNn
RRAn
PosA
RRNe

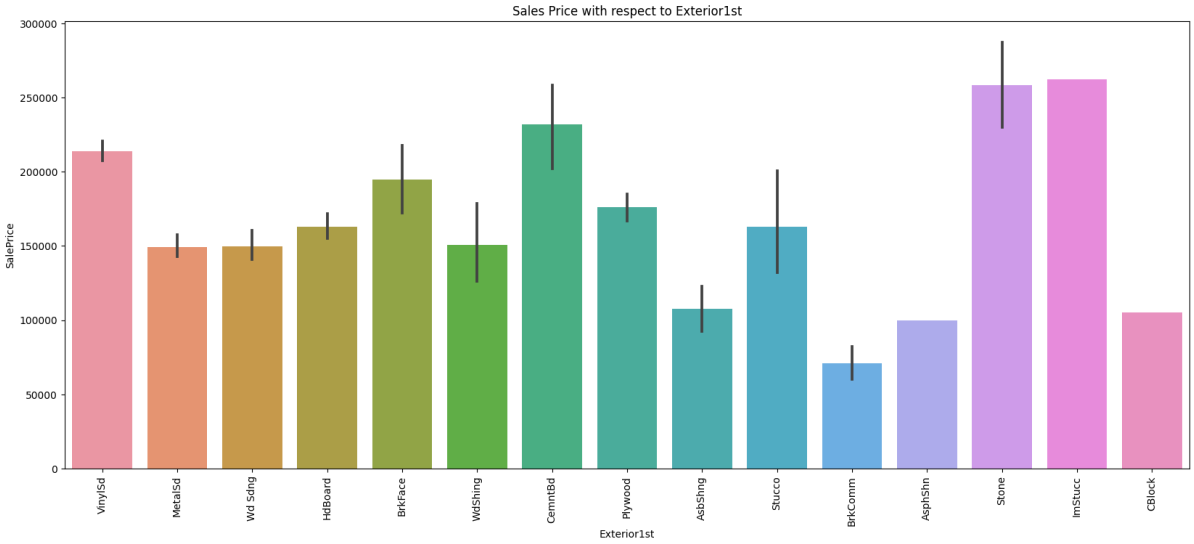

Sales Price with respect to Condition2
SalePrice
Condition2
Norm
Artery
RRNn
Feedr
PosN
PosA
RRAn
RRAe

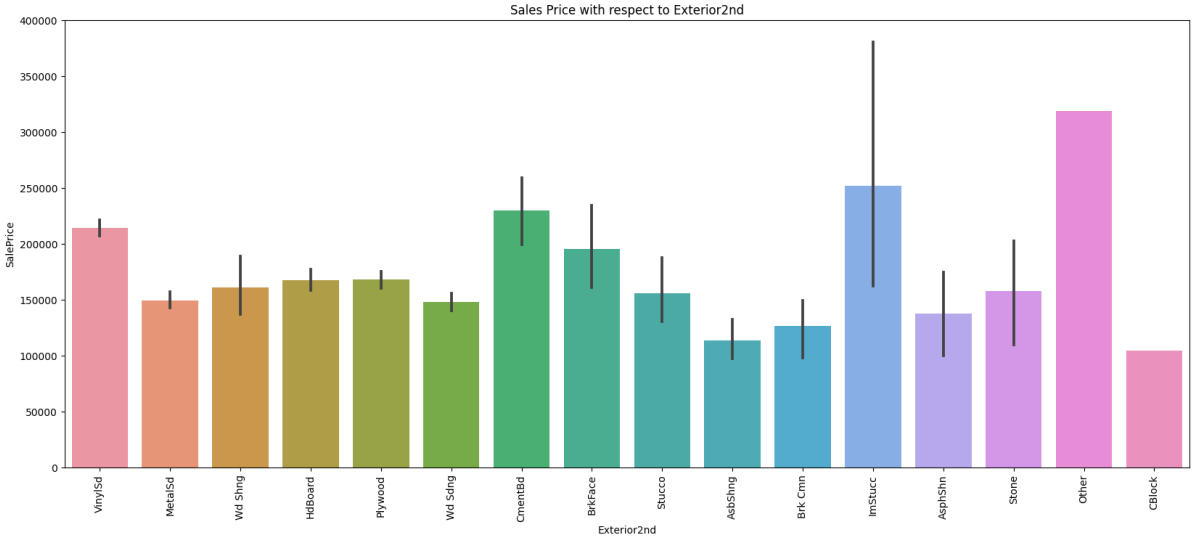

Sales Price with respect to BldgType
SalePrice
BldgType
1Fam
2fmCon
Duplex
TwnhsE
Twnhs

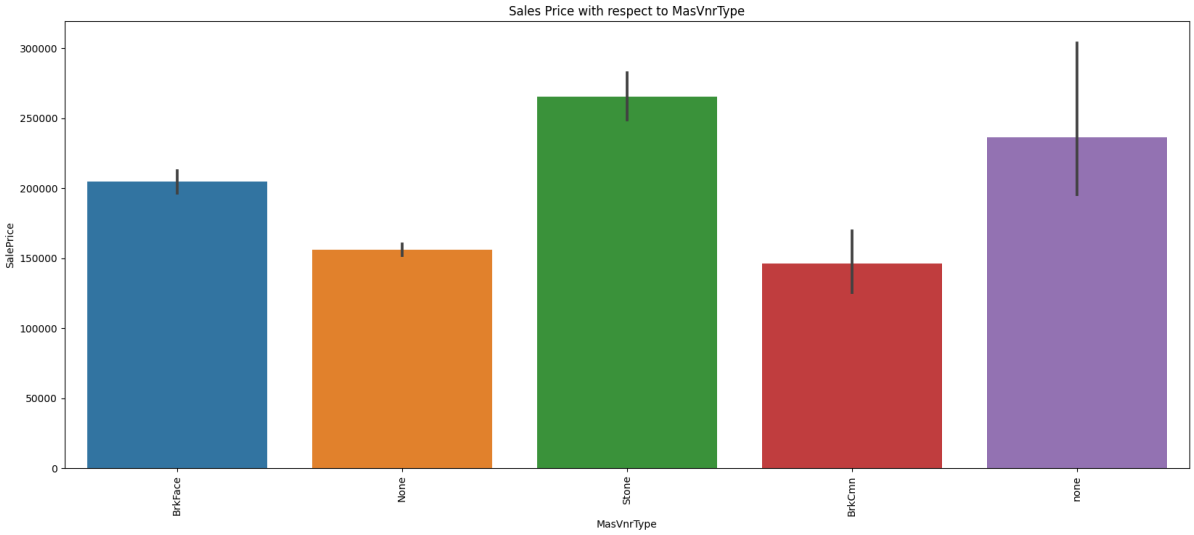

Sales Price with respect to HouseStyle
SalePrice
300000
250000
200000
150000
100000
50000
0
2Story
1Story
1.5Fin
1.5Unf
SFoyer
SLvl
2.5Unf
2.5Fin
HouseStyle

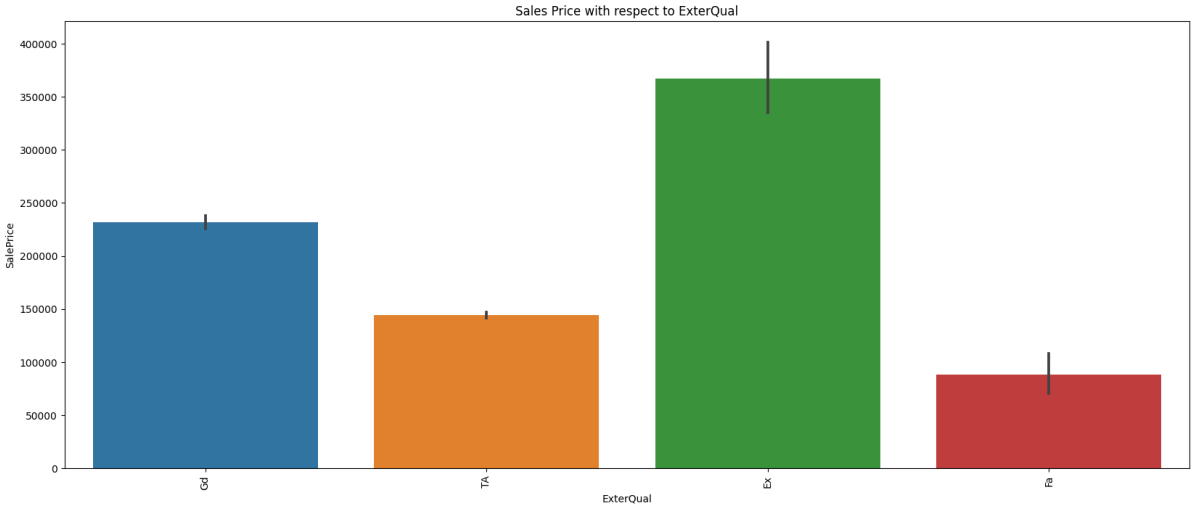

Sales Price with respect to RoofStyle
SalePrice
250000
200000
150000
100000
50000
0
Gable
Hip
Gambrel
Mansard
Flat
Shed
RoofStyle

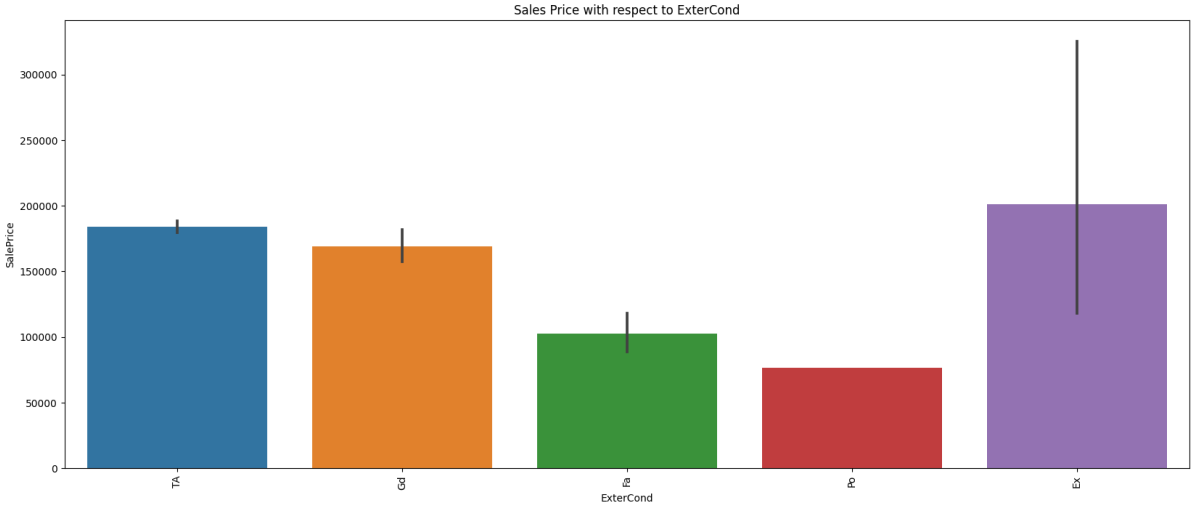

Sales Price with respect to RoofMatl
SalePrice
500000
400000
300000
200000
100000
0
CompShg
WdShngl
Metal
WdShake
Membran
Tar&Grv
Roll
ClyTile
RoofMatl

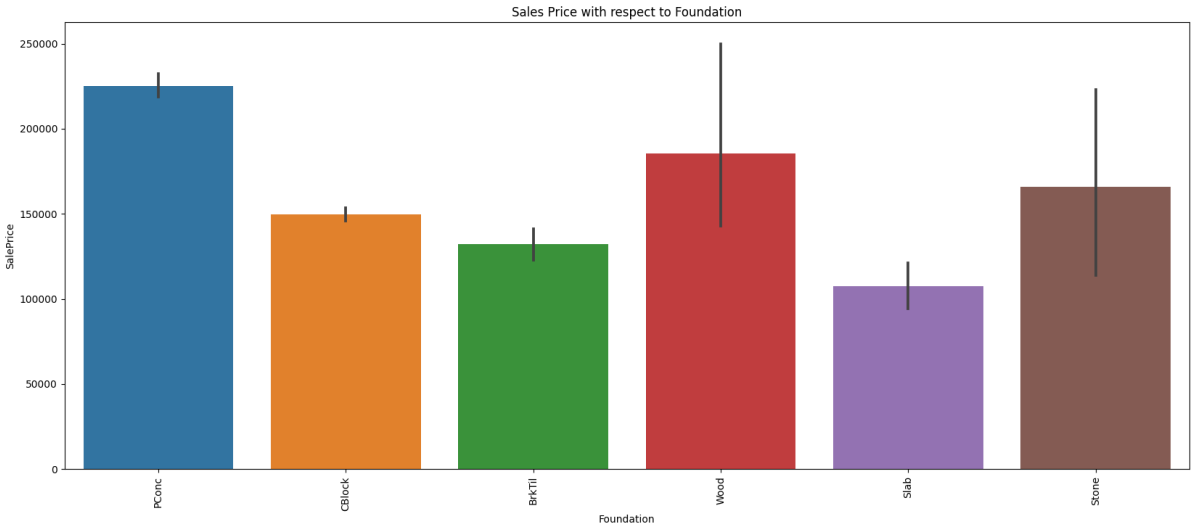

Sales Price with respect to Exterior1st
SalePrice
Exterior1st
VinylSd
MetalSd
Wd Sdng
HdBoard
BrkFace
WdShing
CemntBd
Plywood
AsbShng
Stucco
BrkComm
AsphShn
Stone
ImStucc
CBlock

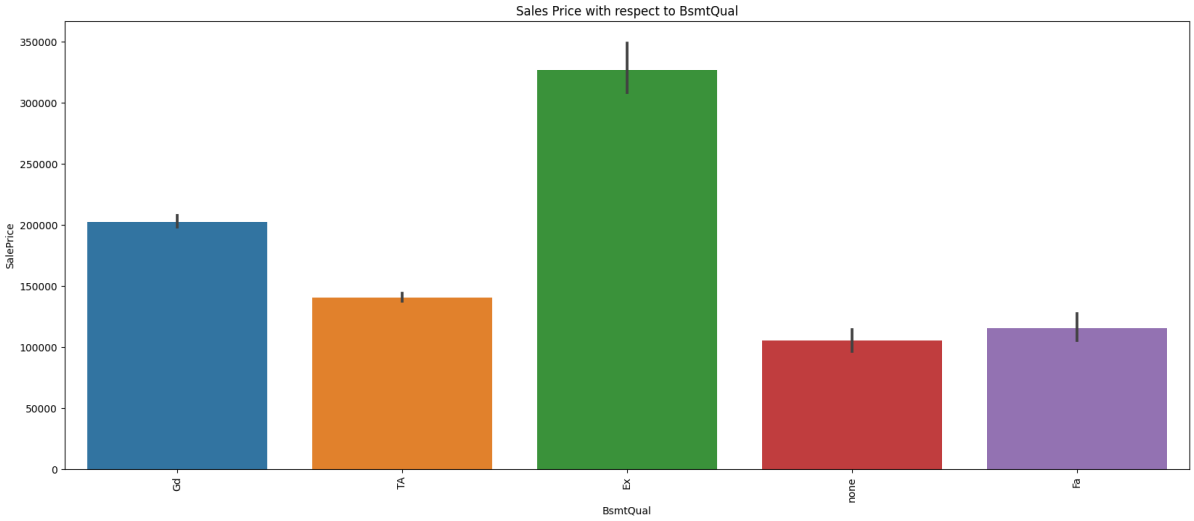

Sales Price with respect to Exterior2nd
SalePrice
Exterior2nd
VinylSd
MetalSd
Wd Shng
HdBoard
Plywood
Wd Sdng
CmentBd
BrkFace
Stucco
AsbShng
Brk Cmn
ImStucc
AsphShn
Stone
Other
CBlock

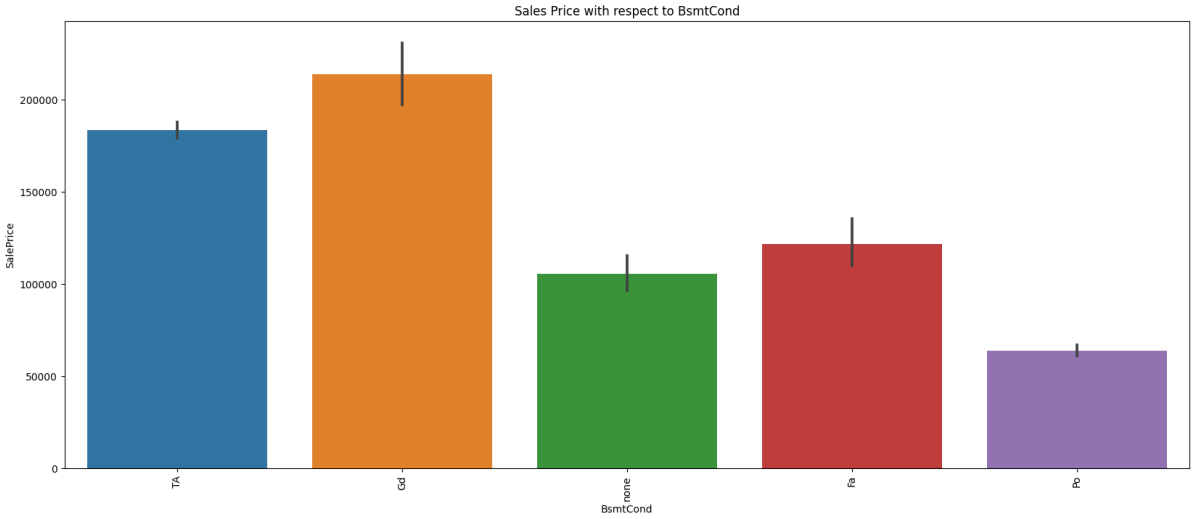

Sales Price with respect to MasVnrType
SalePrice
MasVnrType
BrkFace
None
Stone
BrkCmn
none

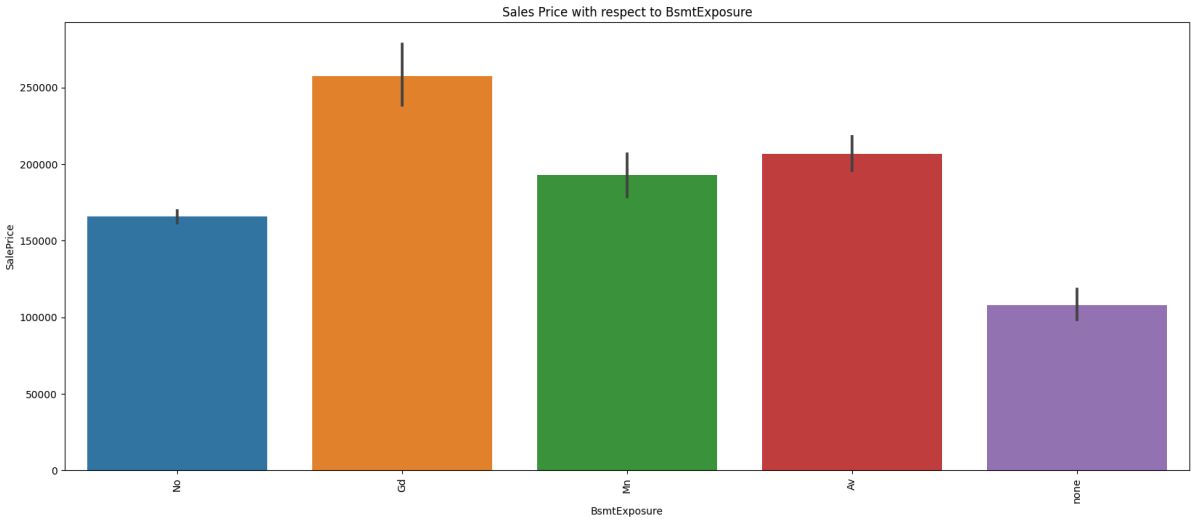

Sales Price with respect to ExterQual
SalePrice
ExterQual
Gd
TA
Ex
Fa

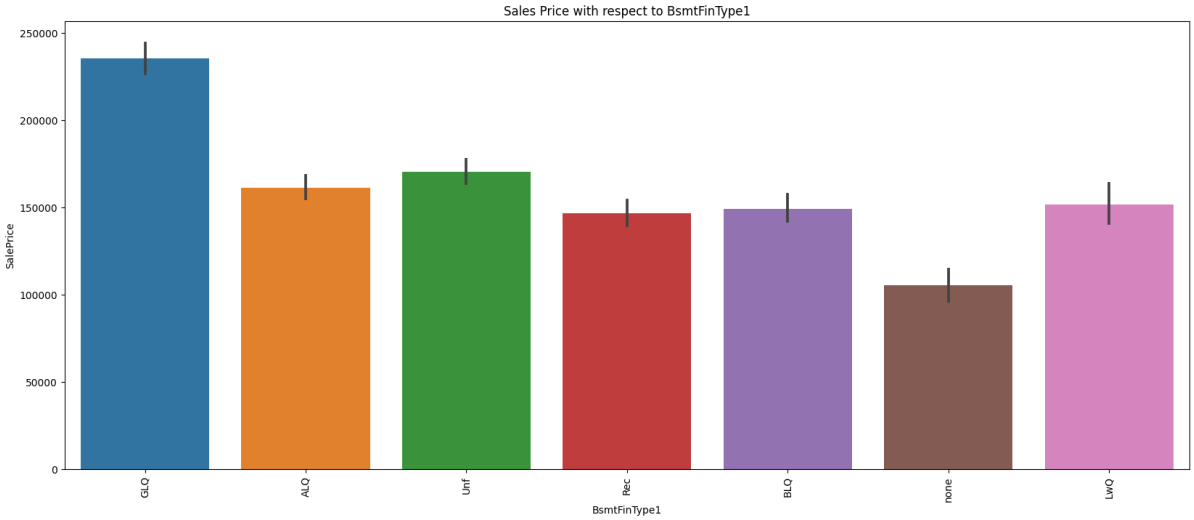

Sales Price with respect to ExterCond
SalePrice
ExterCond
TA
Gd
Fa
Po
Ex

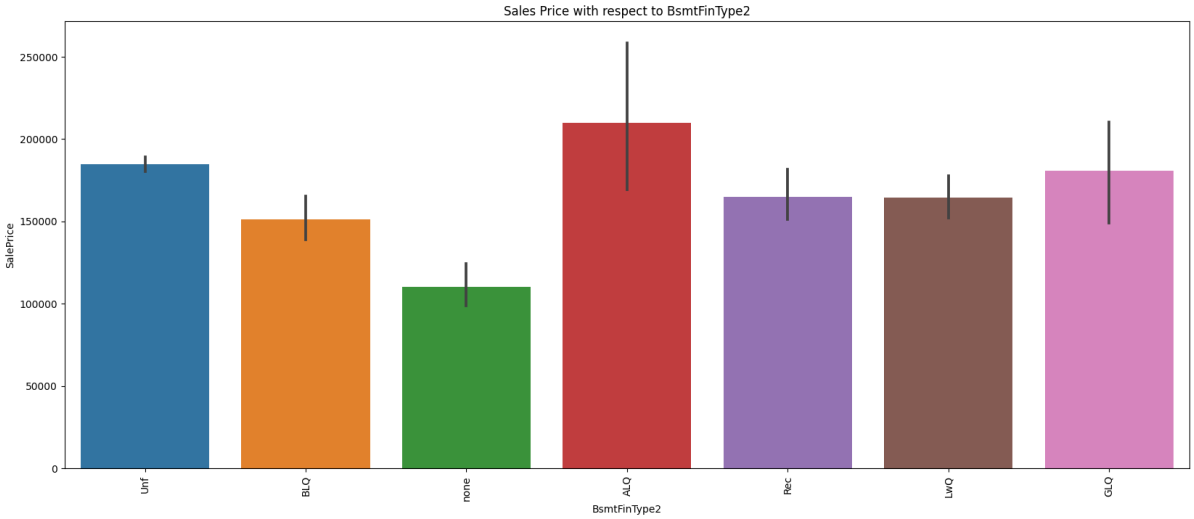

Sales Price with respect to Foundation
SalePrice
Foundation
PConc
CBlock
BrkTil
Wood
Slab
Stone

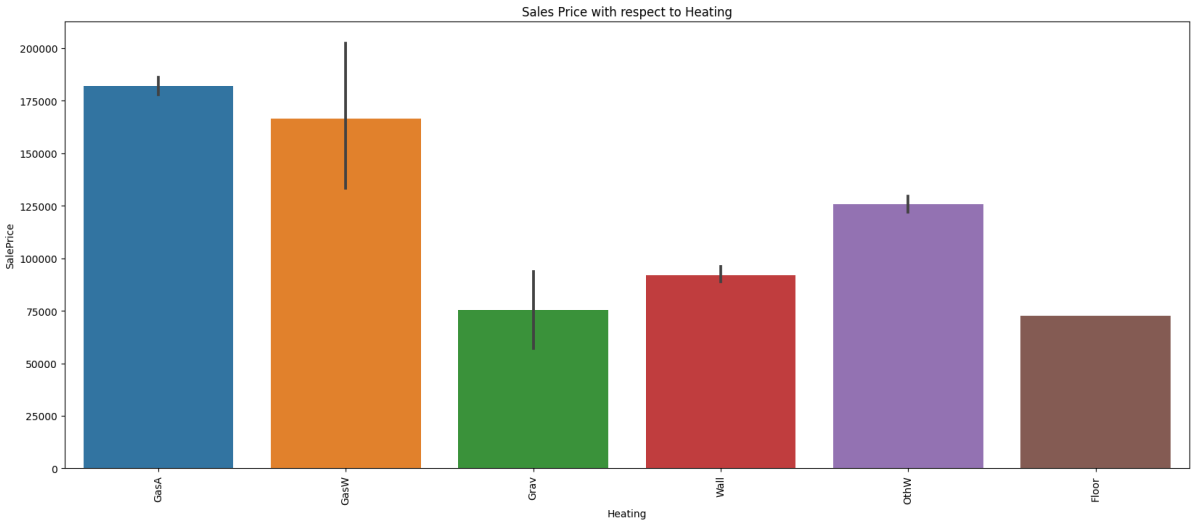

Sales Price with respect to BsmtQual
SalePrice
BsmtQual
Gd
TA
Ex
none
Fa

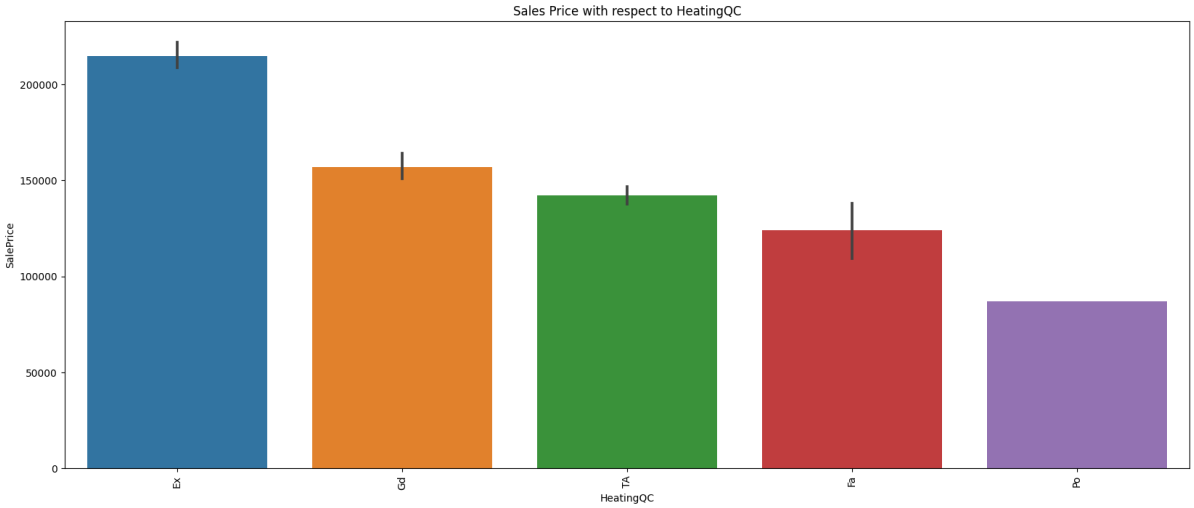

Sales Price with respect to BsmtCond
SalePrice
BsmtCond
TA
Gd
none
Fa
Po

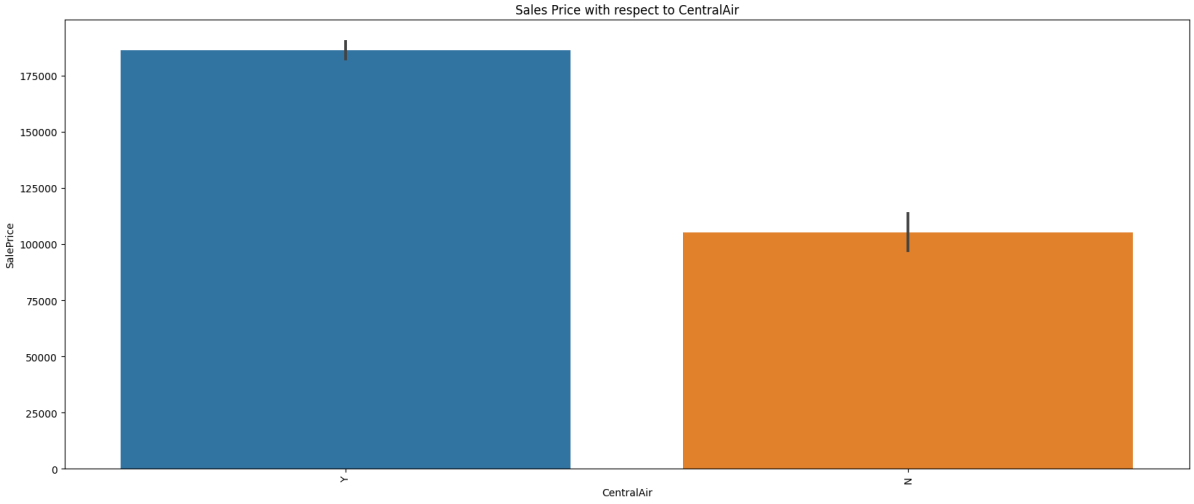

Sales Price with respect to BsmtExposure
SalePrice
BsmtExposure
No
Gd
Mn
Av
none

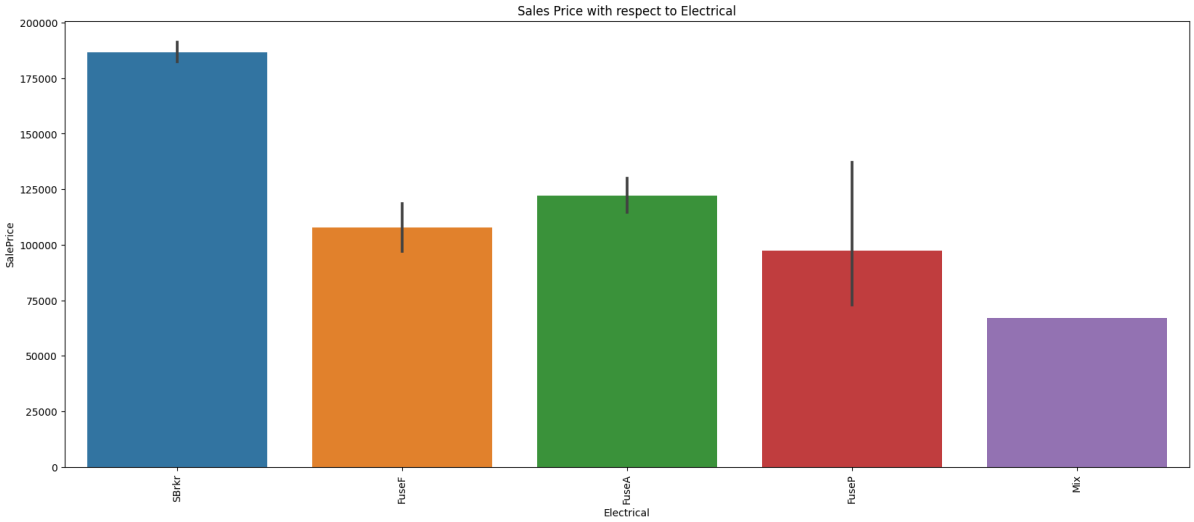

Sales Price with respect to BsmtFinType1
SalePrice
BsmtFinType1
GLQ
ALQ
Unf
Rec
BLQ
none
LwQ

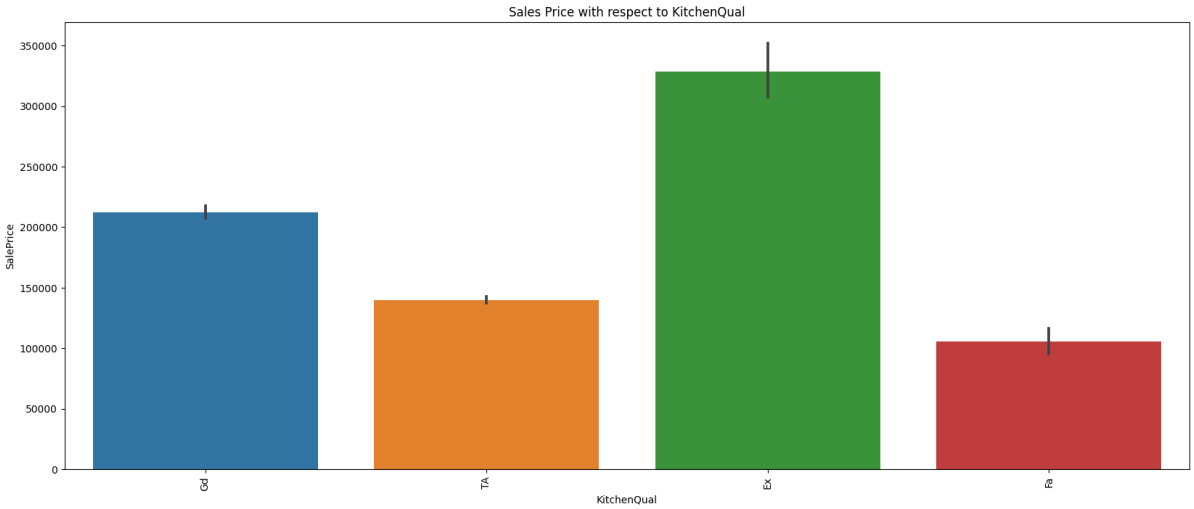

Sales Price with respect to BsmtFinType2
SalePrice
BsmtFinType2
Unf
BLQ
none
ALQ
Rec
LwQ
GLQ

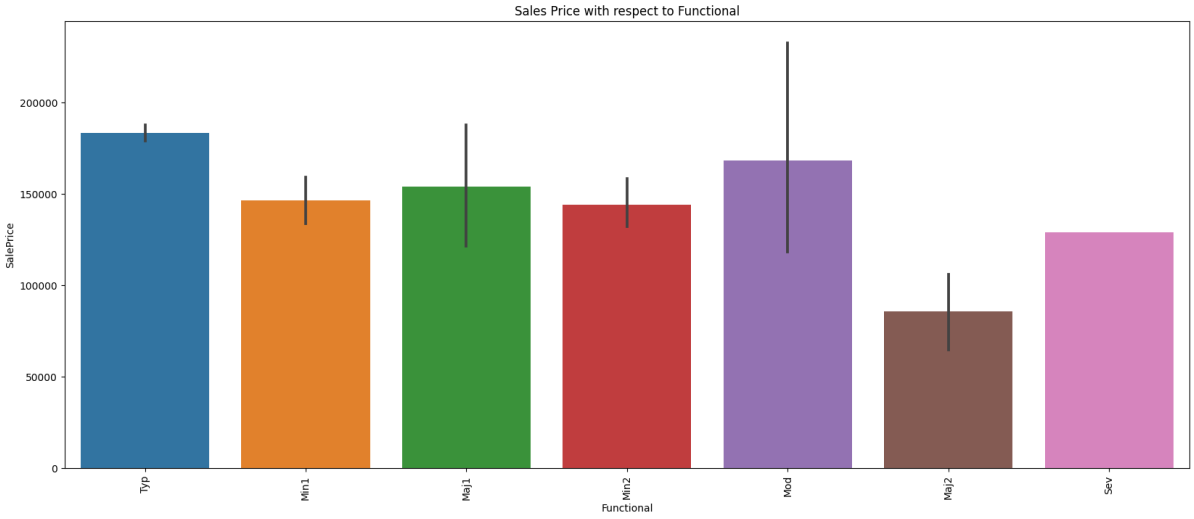

Sales Price with respect to Heating
SalePrice
Heating
GasA
GasW
Grav
Wall
OthW
Floor

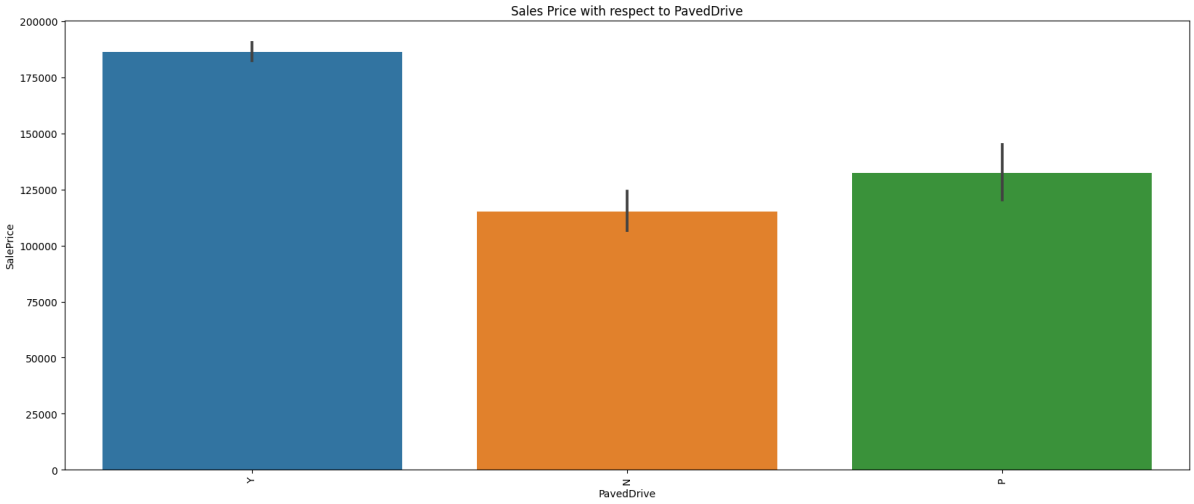

Sales Price with respect to HeatingQC
SalePrice
0
50000
100000
150000
200000
Ex
Gd
TA
Fa
Po
HeatingQC

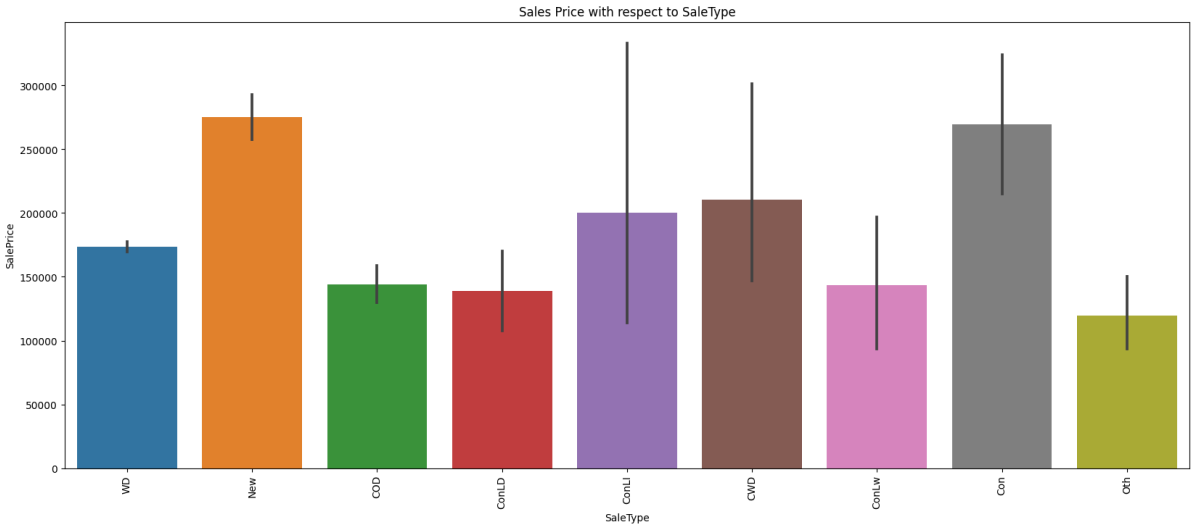

Sales Price with respect to CentralAir
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
Y
N
CentralAir

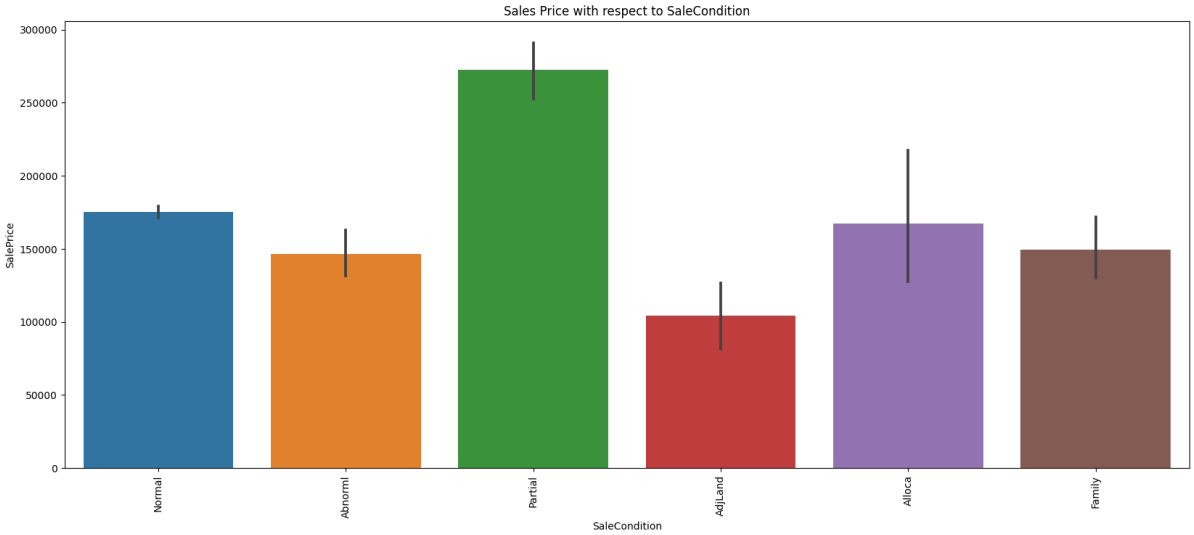

Sales Price with respect to Electrical
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
SBrkr
FuseF
FuseA
FuseP
Mix
Electrical


Sales Price with respect to KitchenQual
SalePrice
0
50000
100000
150000
200000
250000
300000
350000
Gd
TA
Ex
Fa
KitchenQual


Sales Price with respect to Functional
SalePrice
200000
150000
100000
50000
0
Typ
Min1
Min2
Mod
Maj2
Sev
Functional


Sales Price with respect to PavedDrive
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
Y
N
P
PavedDrive


Sales Price with respect to SaleType
SalePrice
300000
250000
200000
150000
100000
50000
0
WD
New
COD
ConLD
ConLI
CWD
ConLw
Con
Oth
SaleType



Conclusion

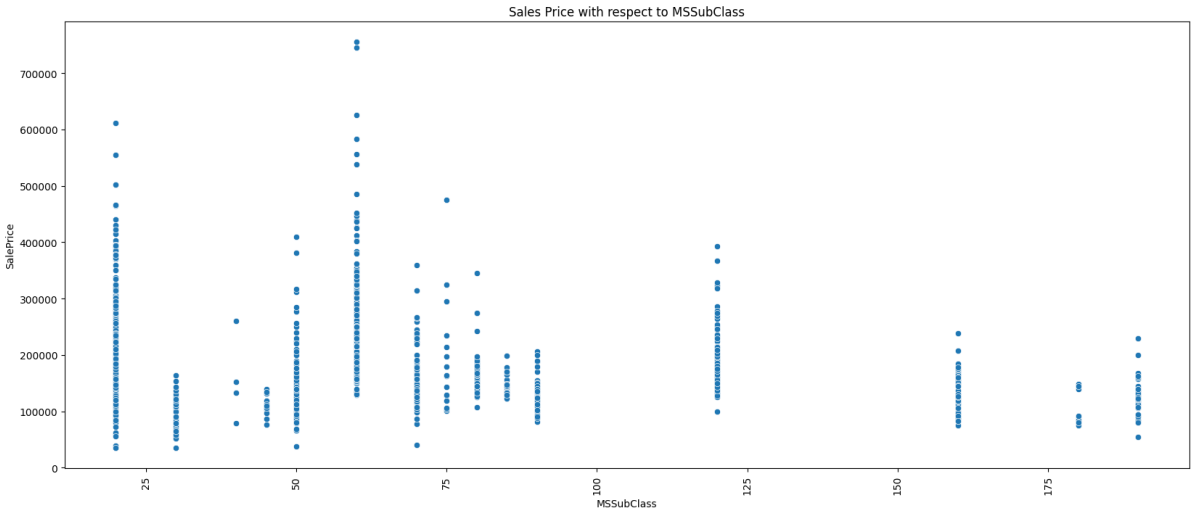
- Sale Price tends to increase for Paved Street and housing with Pavements

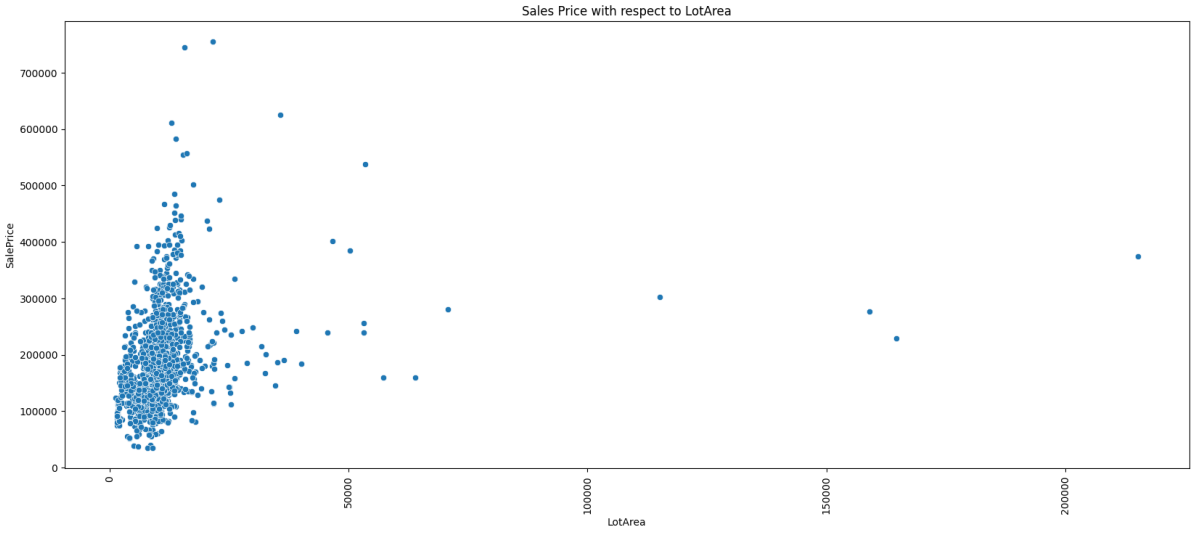
- Sale is more for Centrally AC houses and housese made of Stones

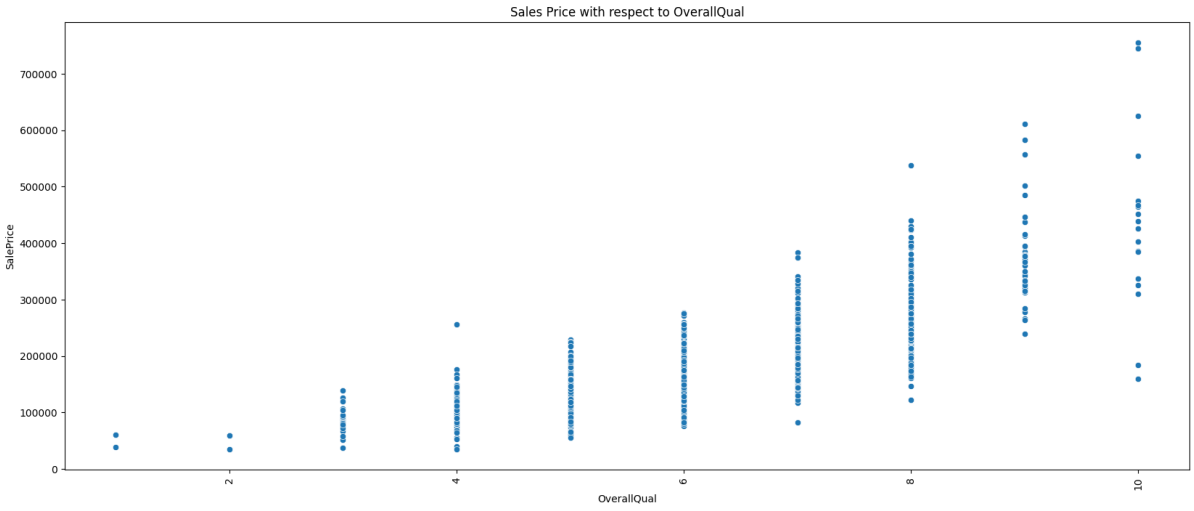
- New Houses tend to demand more Price as compared to Old Houses

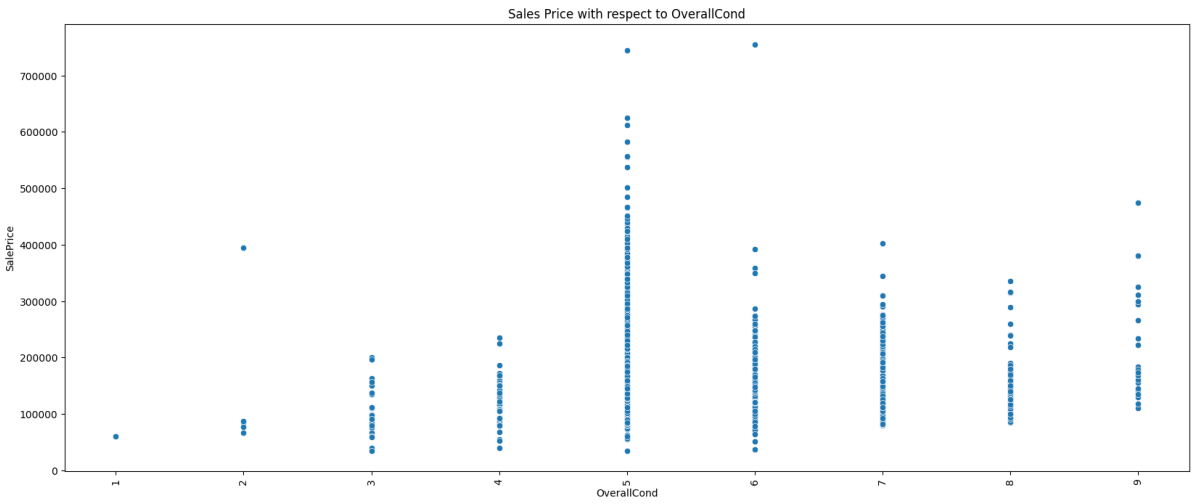
In [ ]:
```
def numerical_columns_graph(columns):
    plt.figure(figsize=(20, 8))
    seaborn.scatterplot(x=columns, y="SalePrice", data= housing_raw_df)
    plt.title("Sales Price with respect to {}".format(columns))
    plt.xticks(rotation=90)
```

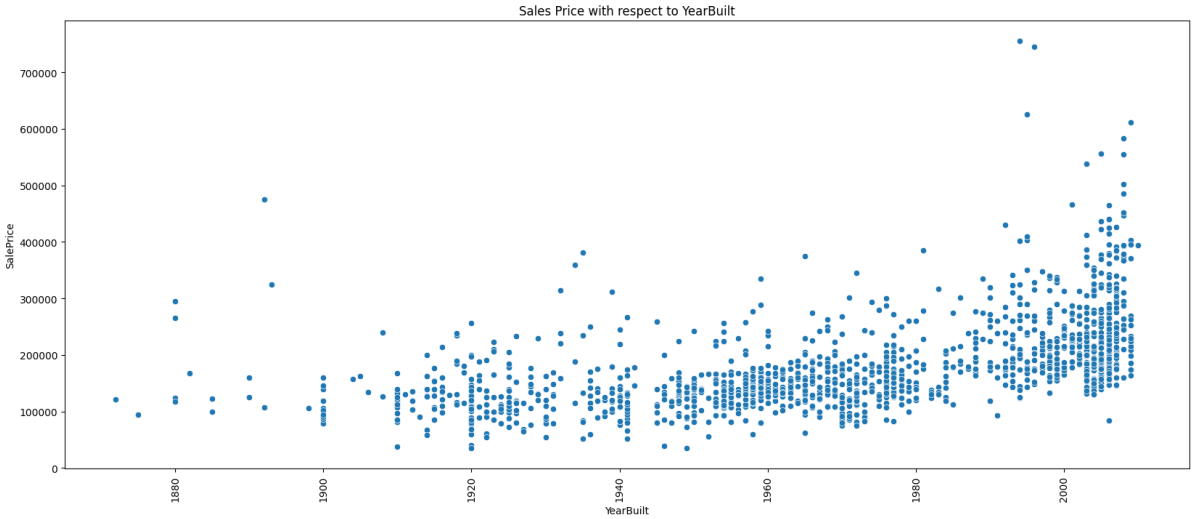
In [ ]:
```
for col in numerical_columns.columns:
   numerical_columns_graph(col)
```

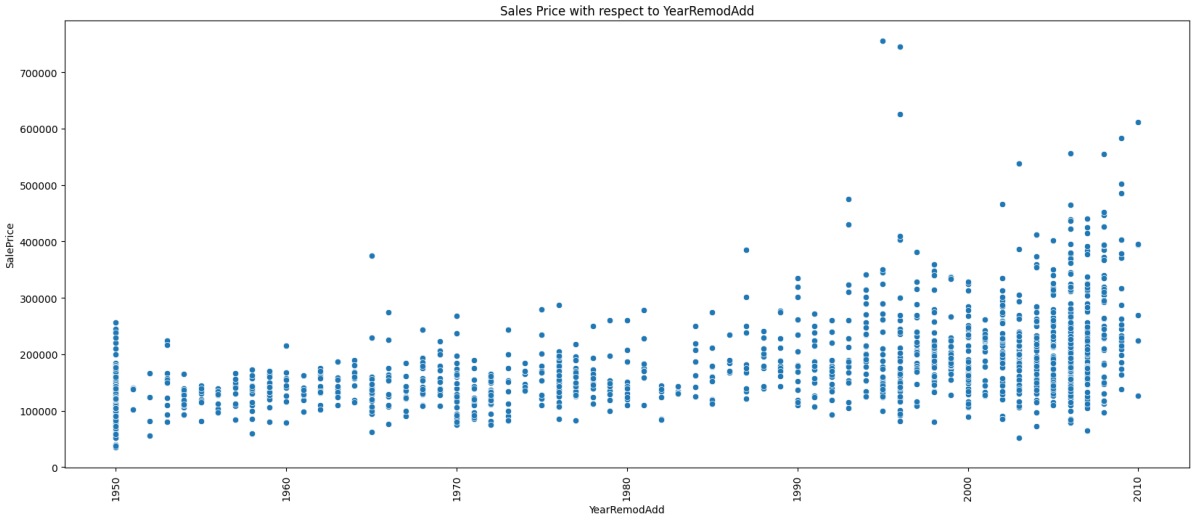


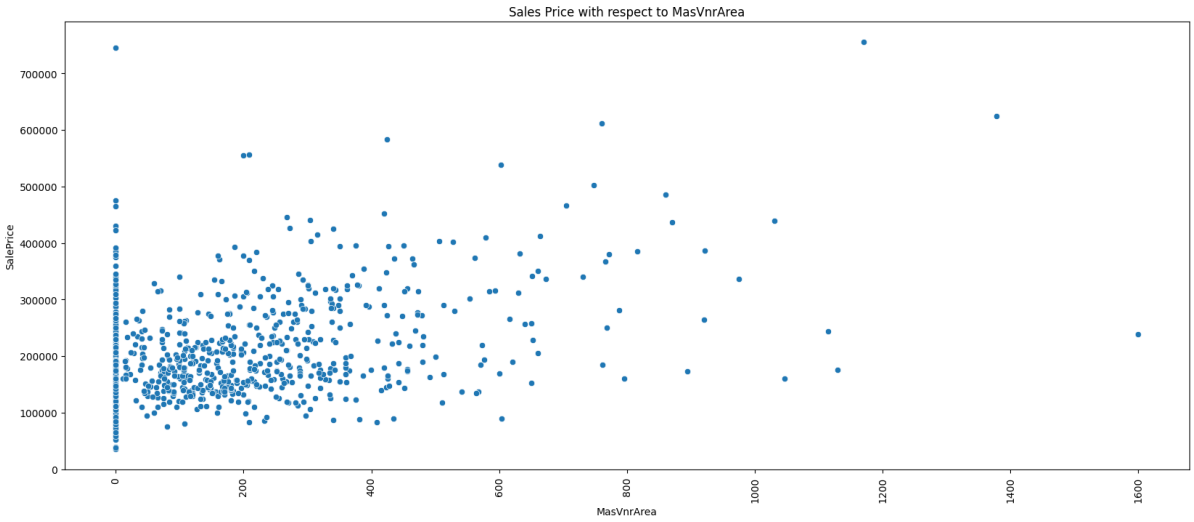

Sales Price with respect to LotArea
SalePrice
LotArea
0
100000
200000
300000
400000
500000
600000
700000
0
50000
100000
150000
200000

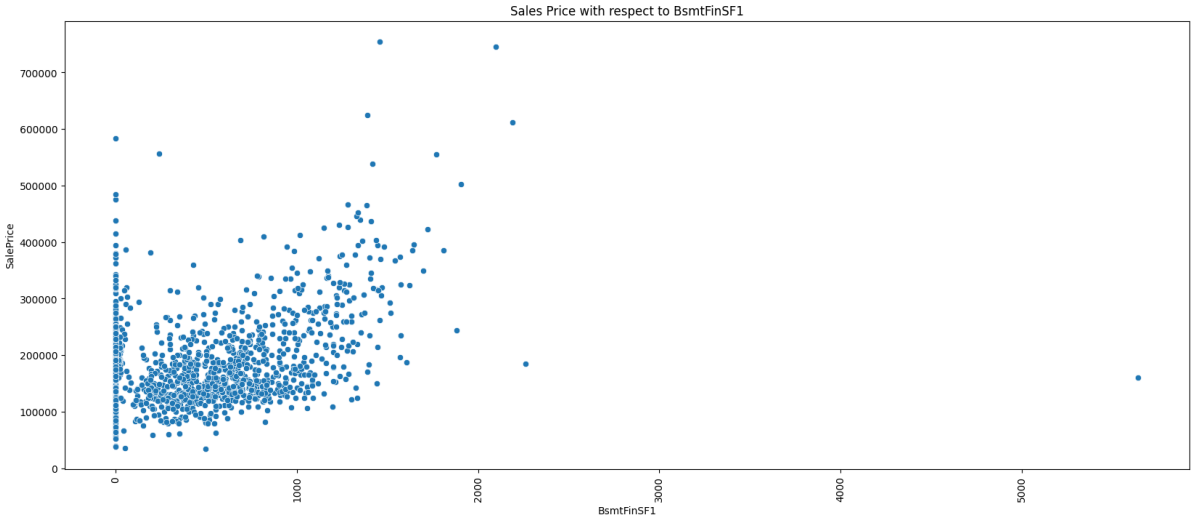

Sales Price with respect to OverallQual
SalePrice
OverallQual
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10

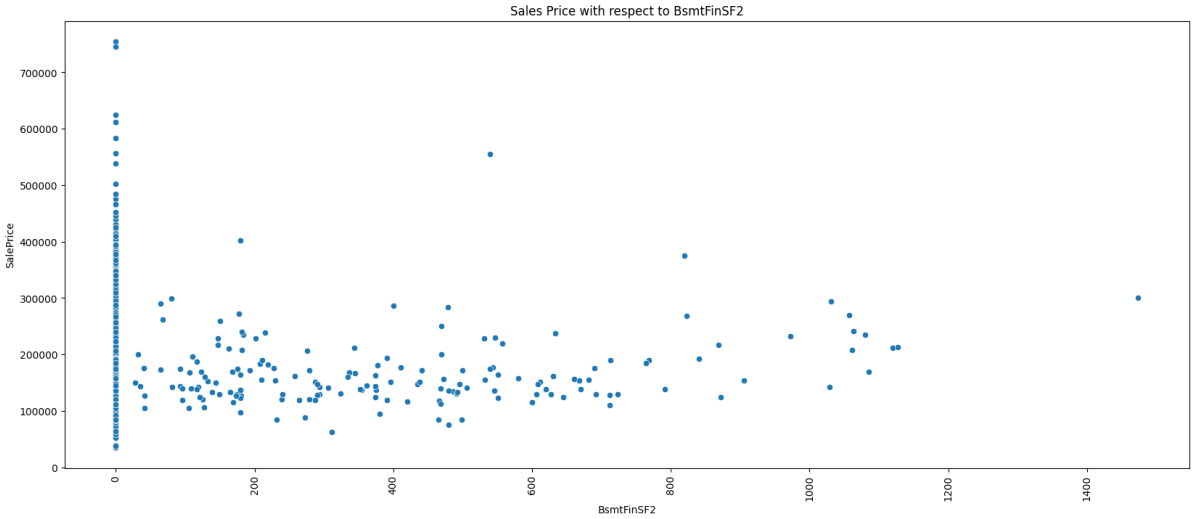

Sales Price with respect to OverallCond
SalePrice
OverallCond
0
100000
200000
300000
400000
500000
600000
700000
1
2
3
4
5
6
7
8
9

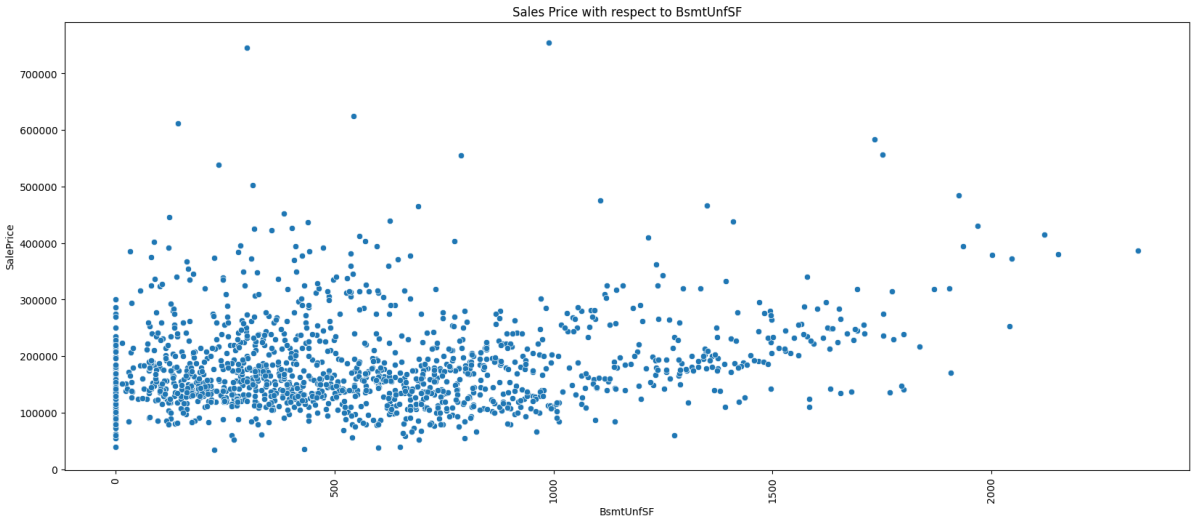

Sales Price with respect to YearBuilt
SalePrice
YearBuilt

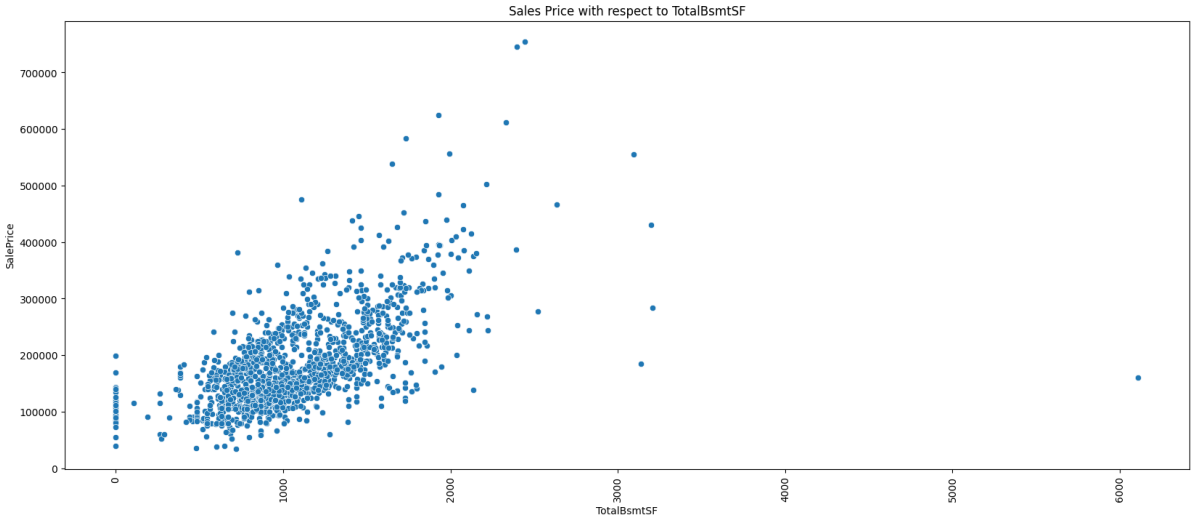

Sales Price with respect to YearRemodAdd
SalePrice
YearRemodAdd

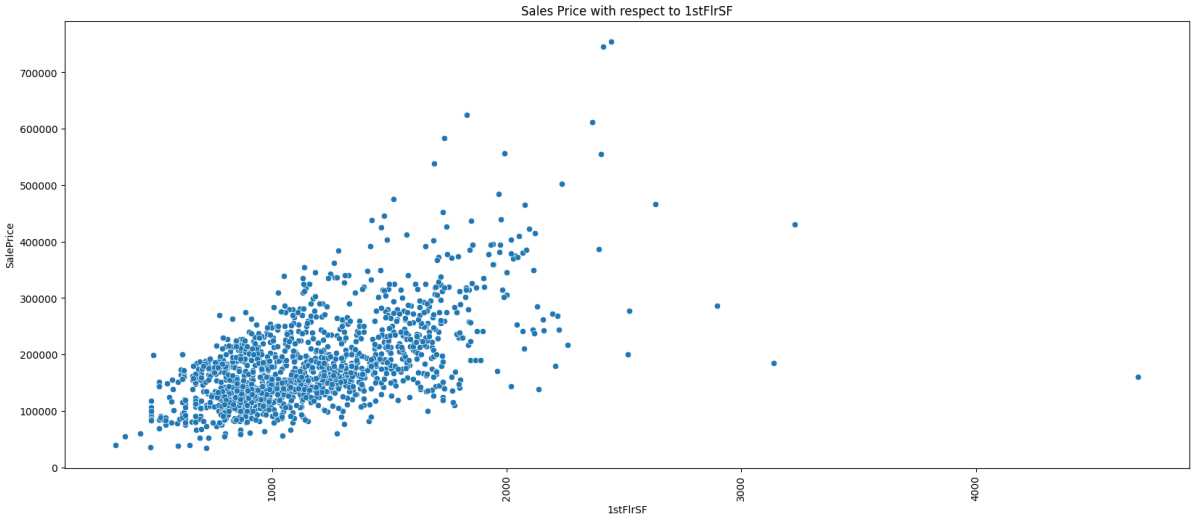

Sales Price with respect to MasVnrArea
SalePrice
MasVnrArea

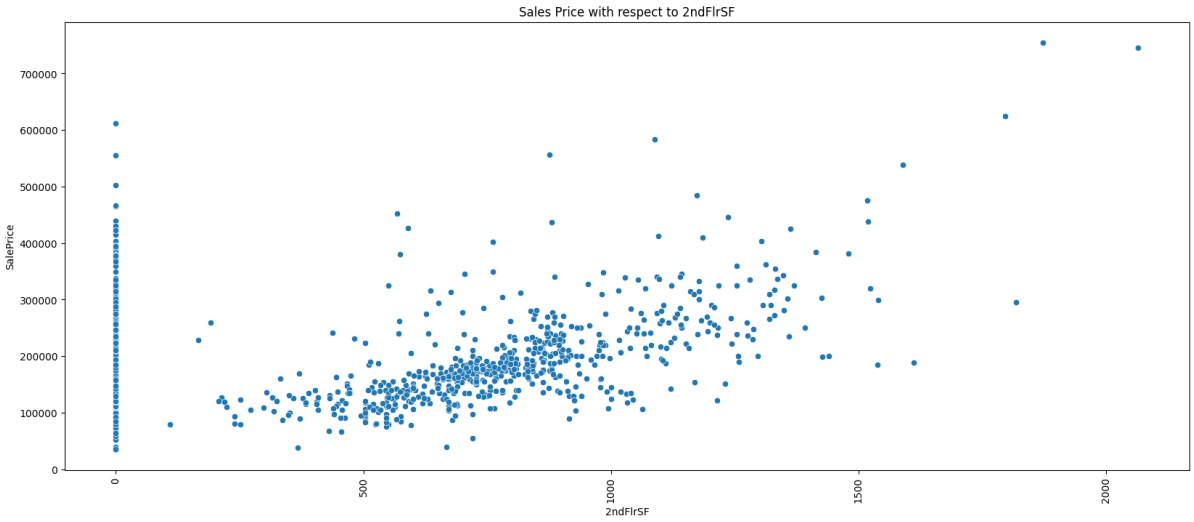

Sales Price with respect to BsmtFinSF1
SalePrice
BsmtFinSF1

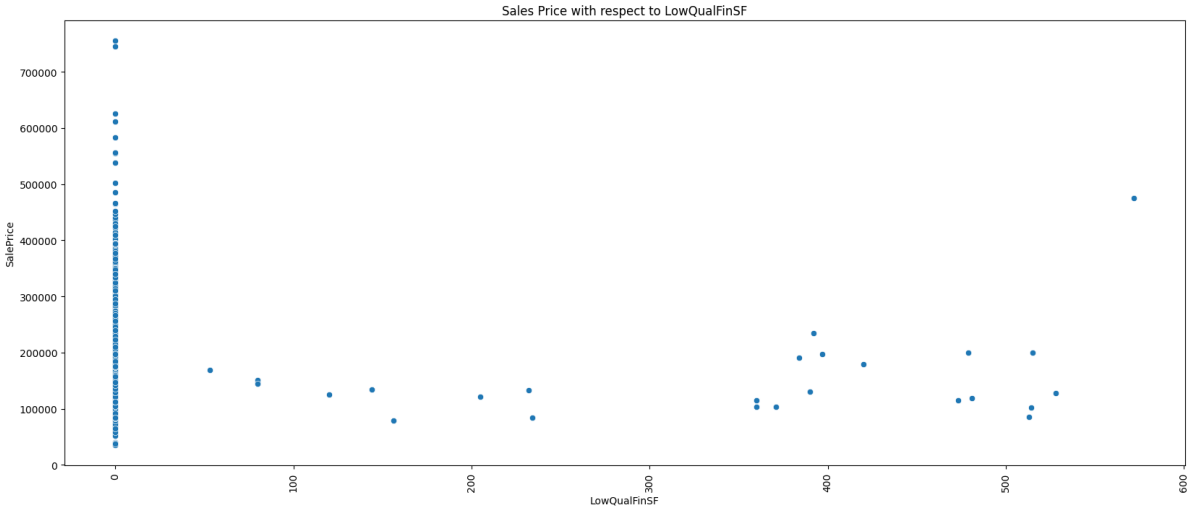

Sales Price with respect to BsmtFinSF2
SalePrice
BsmtFinSF2

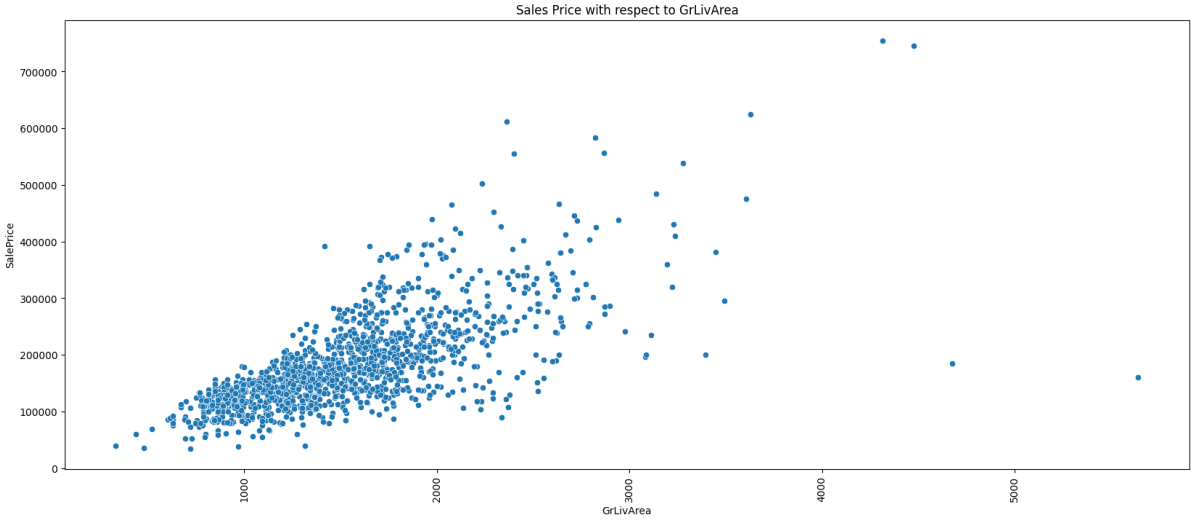

Sales Price with respect to BsmtUnfSF
SalePrice
BsmtUnfSF

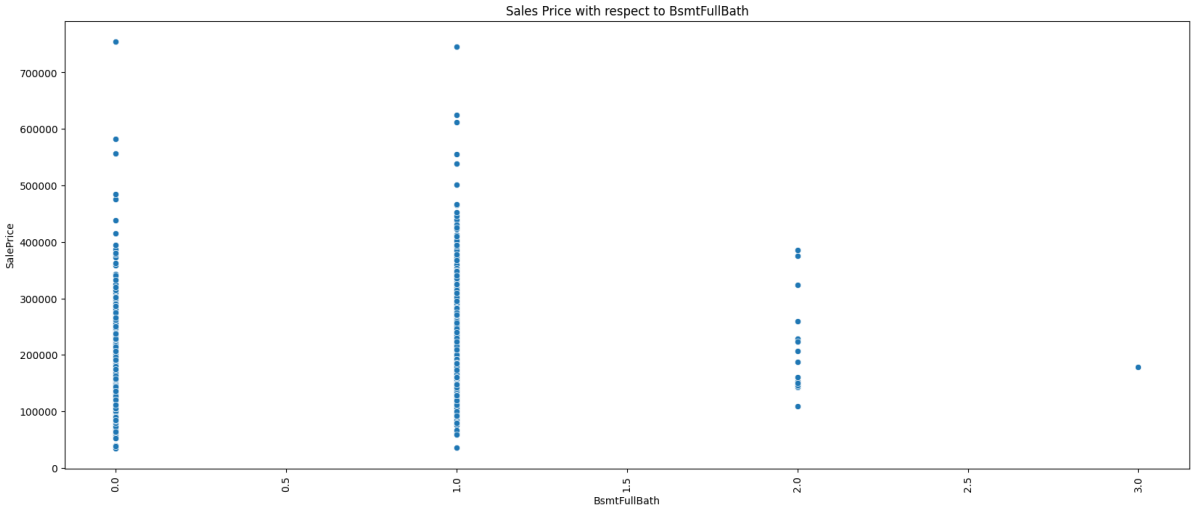

Sales Price with respect to TotalBsmtSF
SalePrice
TotalBsmtSF

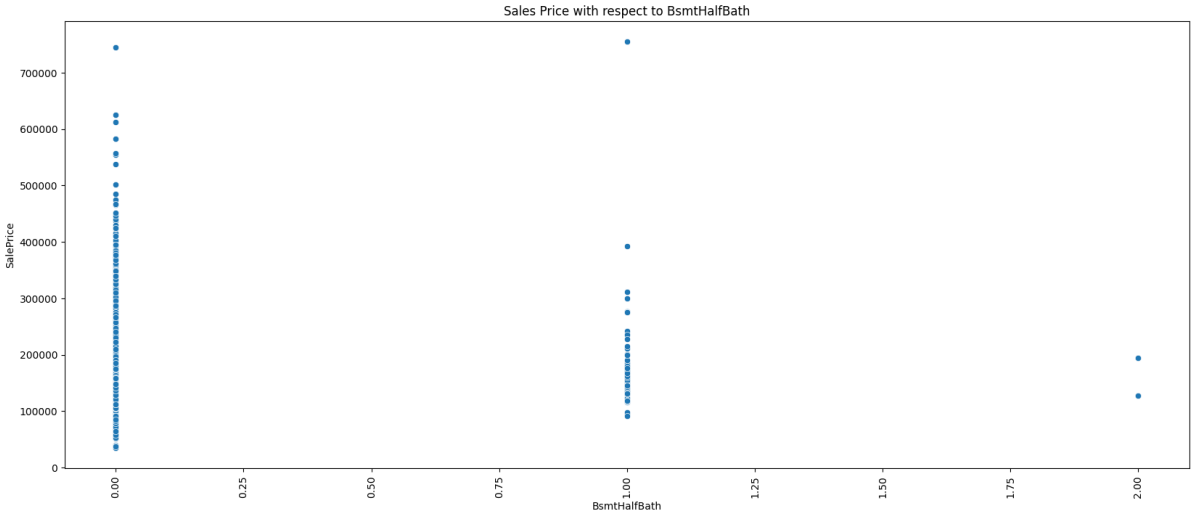

Sales Price with respect to 1stFlrSF
SalePrice
1stFlrSF

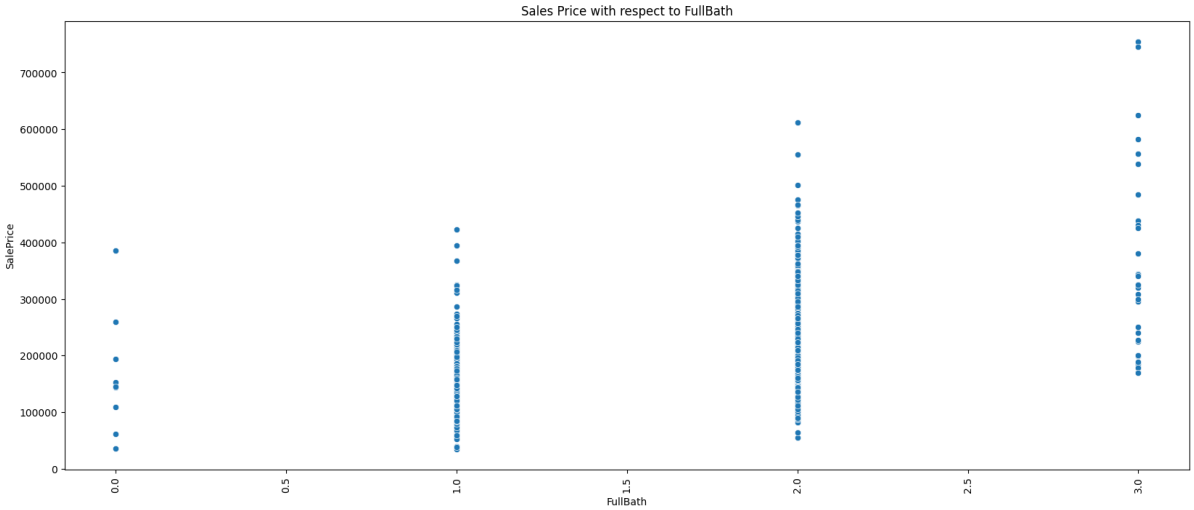

Sales Price with respect to 2ndFlrSF
SalePrice
2ndFlrSF

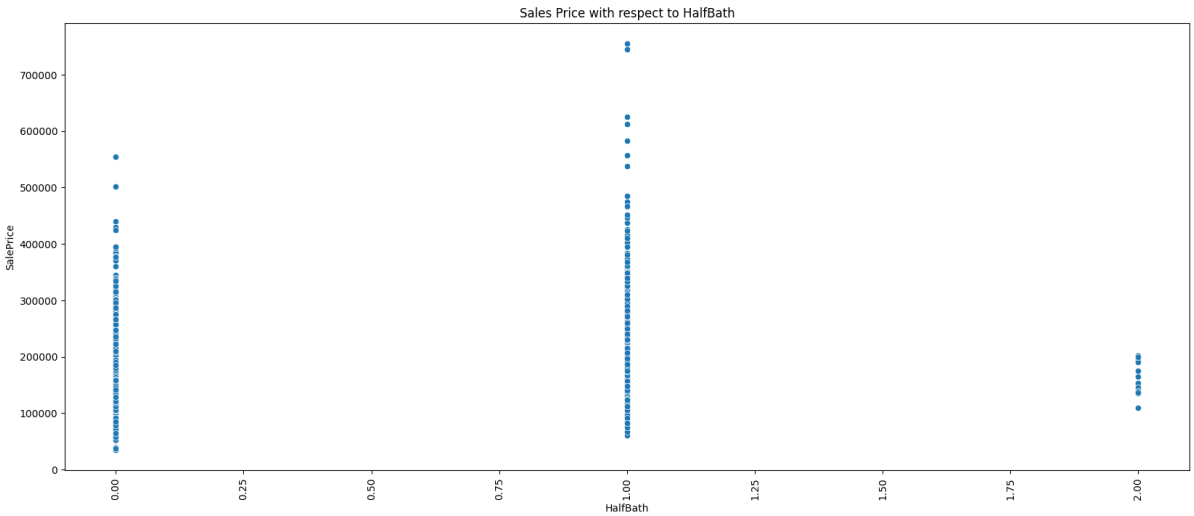

Sales Price with respect to LowQualFinSF
SalePrice
LowQualFinSF

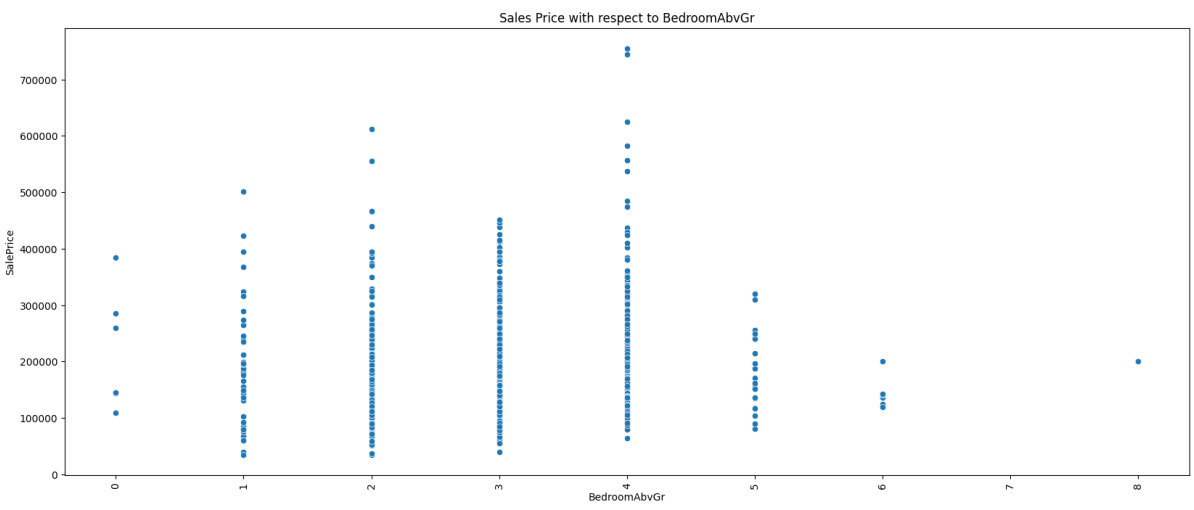

Sales Price with respect to GrLivArea
SalePrice
GrLivArea

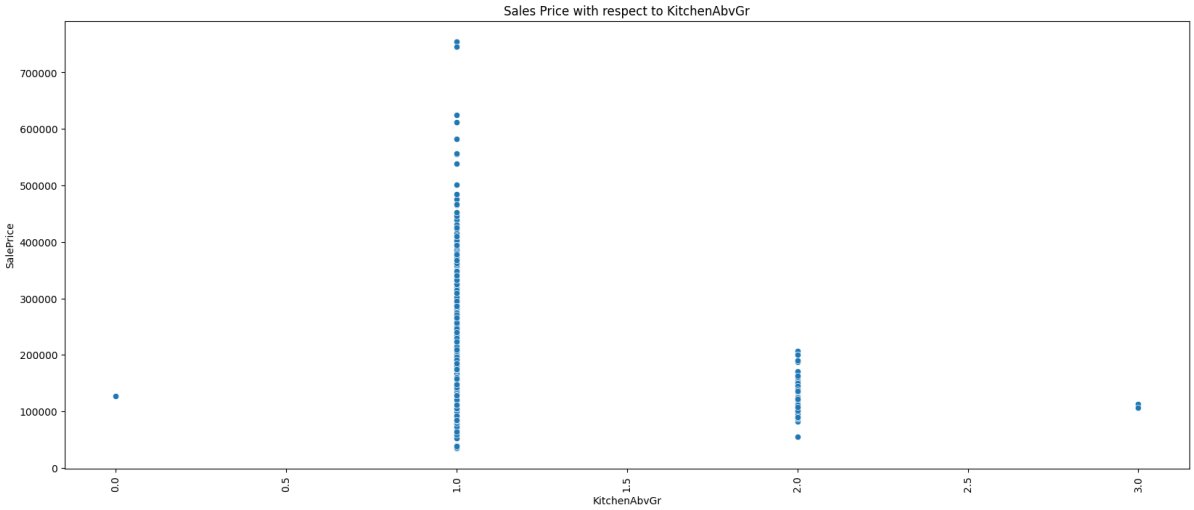

Sales Price with respect to BsmtFullBath
SalePrice
BsmtFullBath

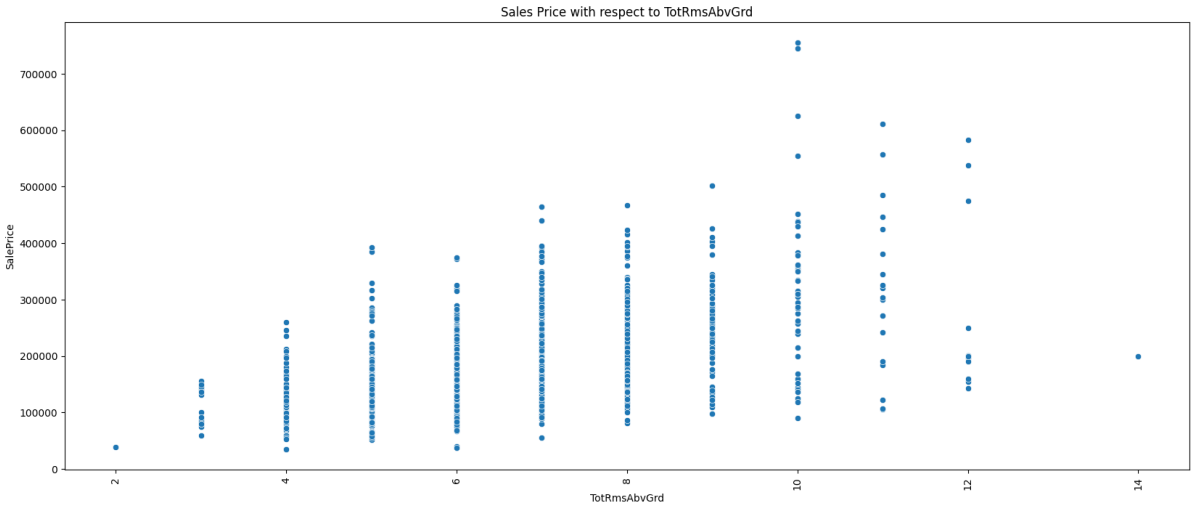

Sales Price with respect to BsmtHalfBath
SalePrice
BsmtHalfBath

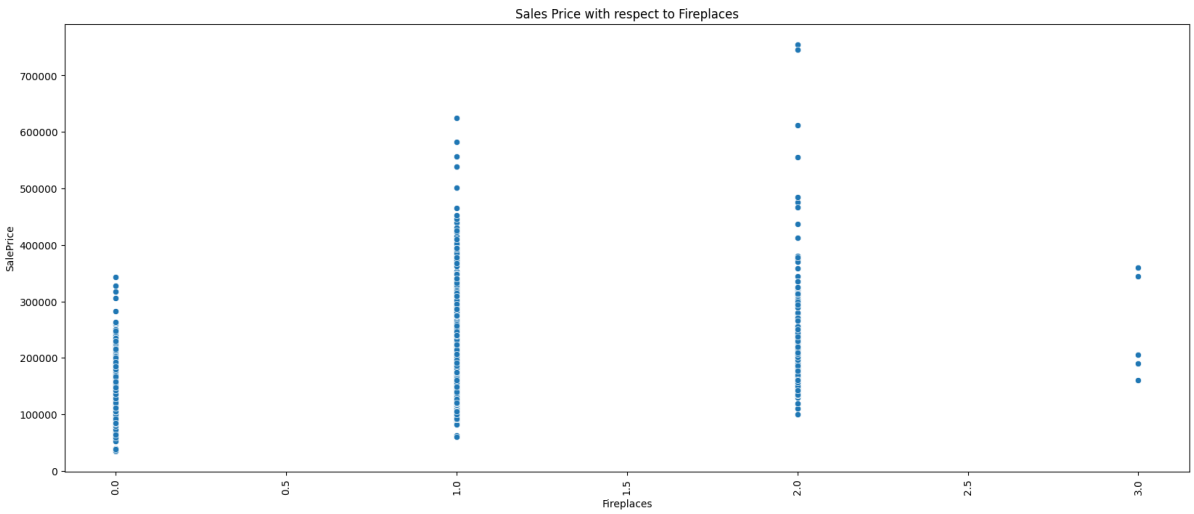

Sales Price with respect to FullBath
SalePrice
FullBath

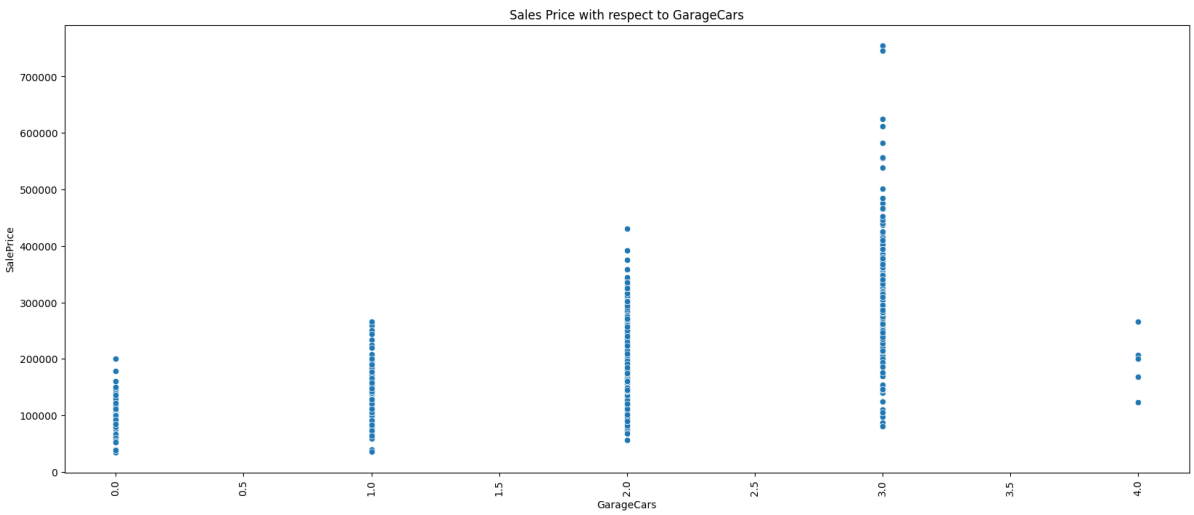

Sales Price with respect to HalfBath
SalePrice
HalfBath

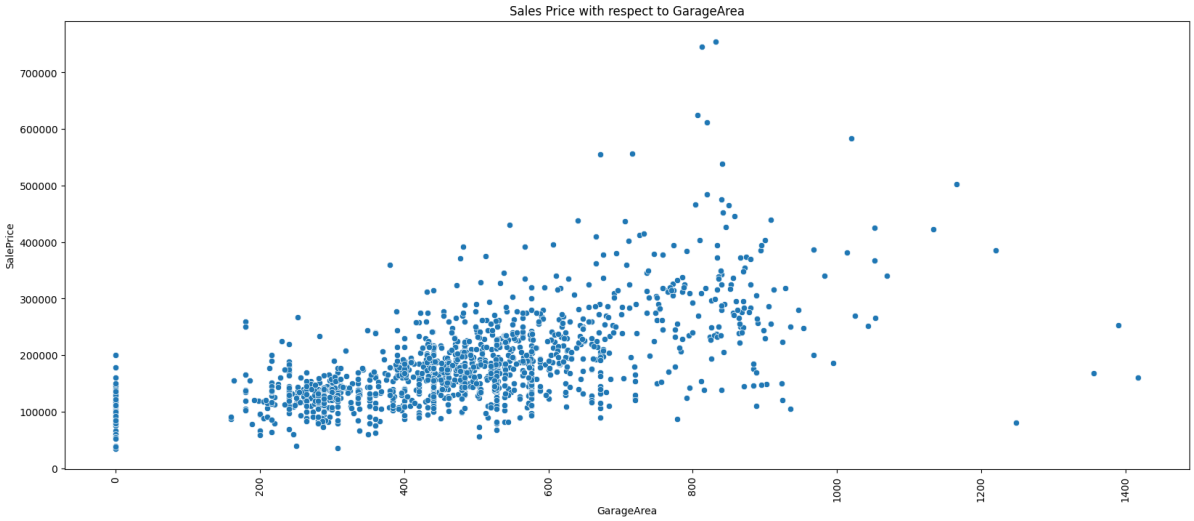

Sales Price with respect to BedroomAbvGr
SalePrice
BedroomAbvGr

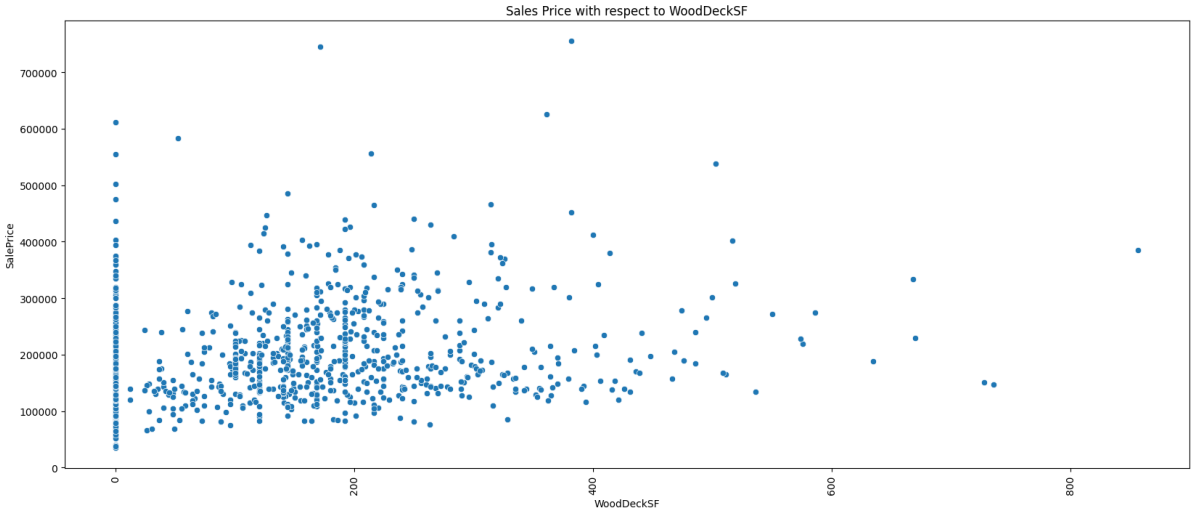

Sales Price with respect to KitchenAbvGr
SalePrice
KitchenAbvGr

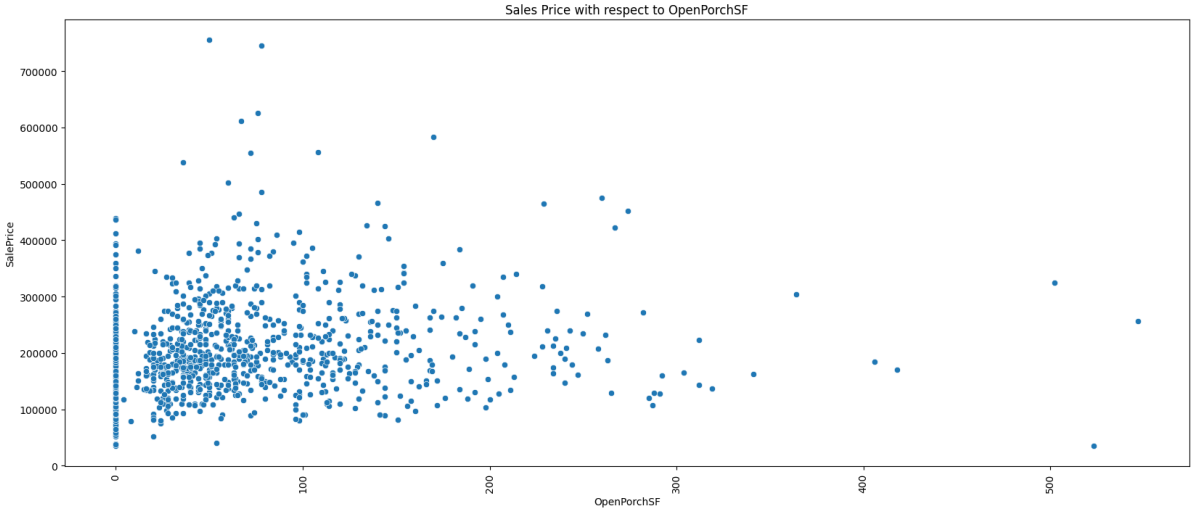

Sales Price with respect to TotRmsAbvGrd
SalePrice
TotRmsAbvGrd

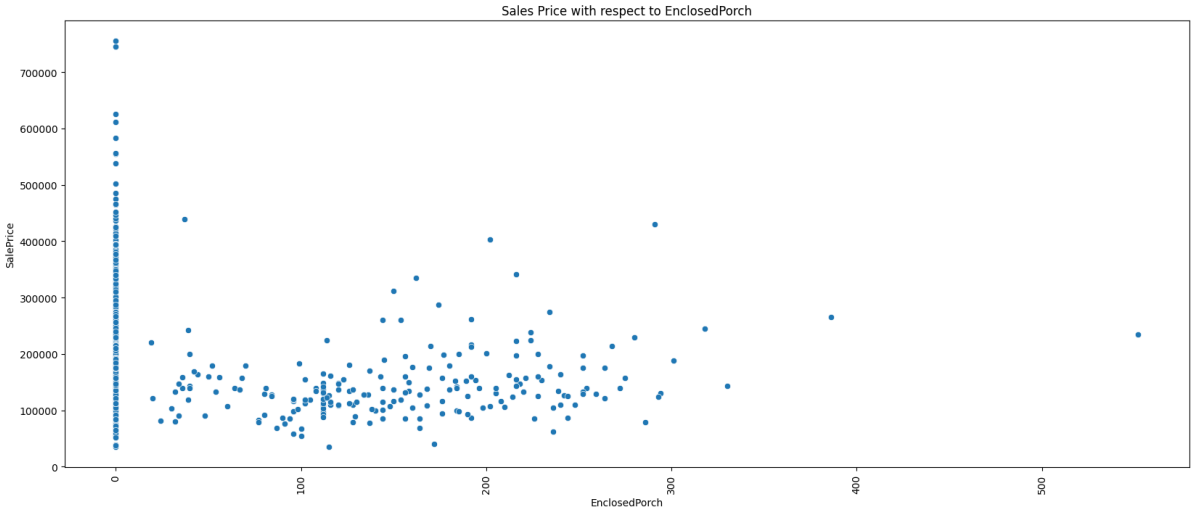

Sales Price with respect to Fireplaces
SalePrice
Fireplaces

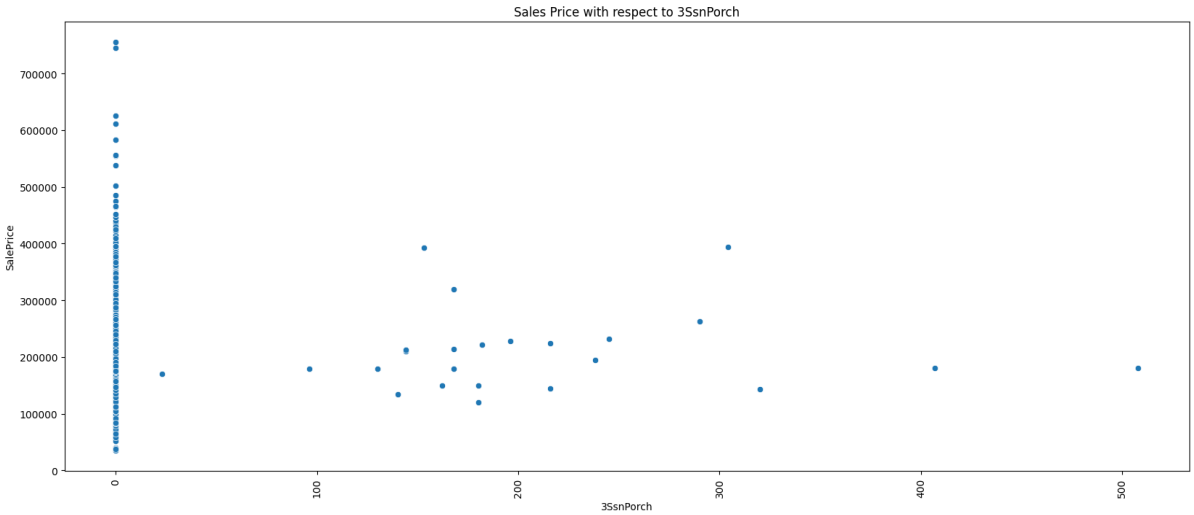

Sales Price with respect to GarageCars
SalePrice
GarageCars

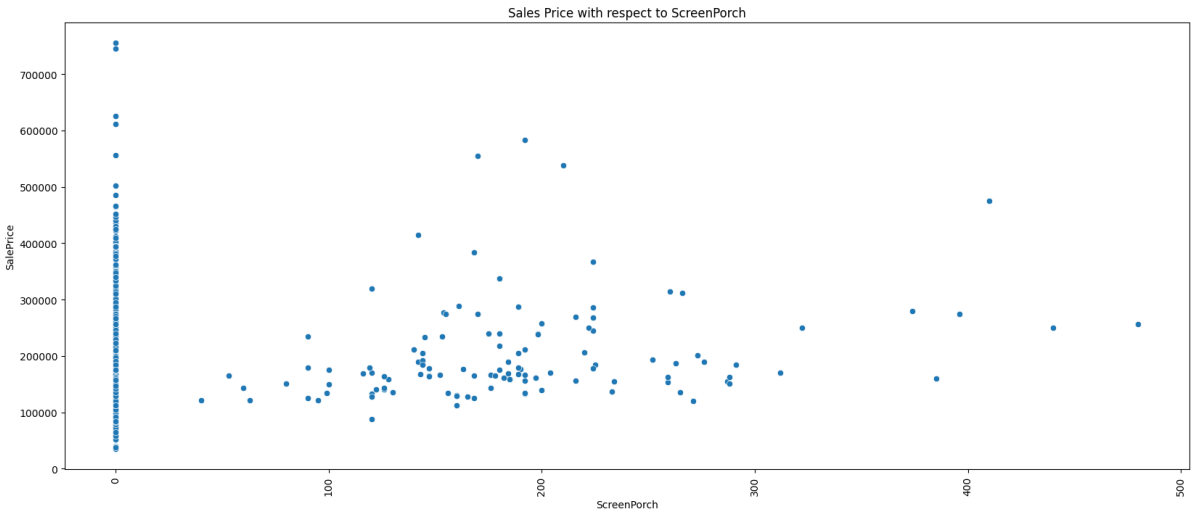

Sales Price with respect to GarageArea
SalePrice
GarageArea
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

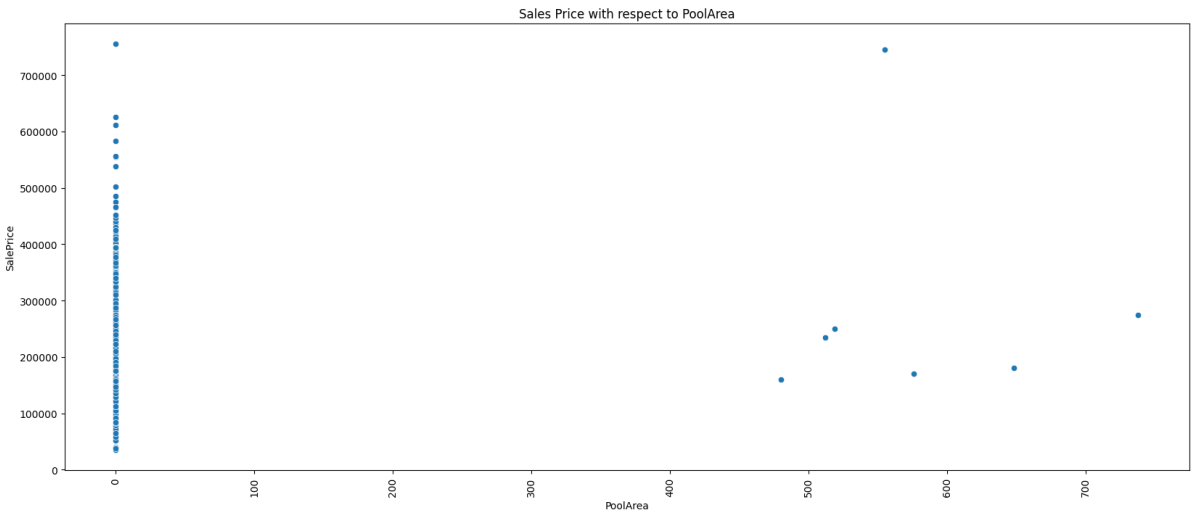

Sales Price with respect to WoodDeckSF
SalePrice
WoodDeckSF
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800

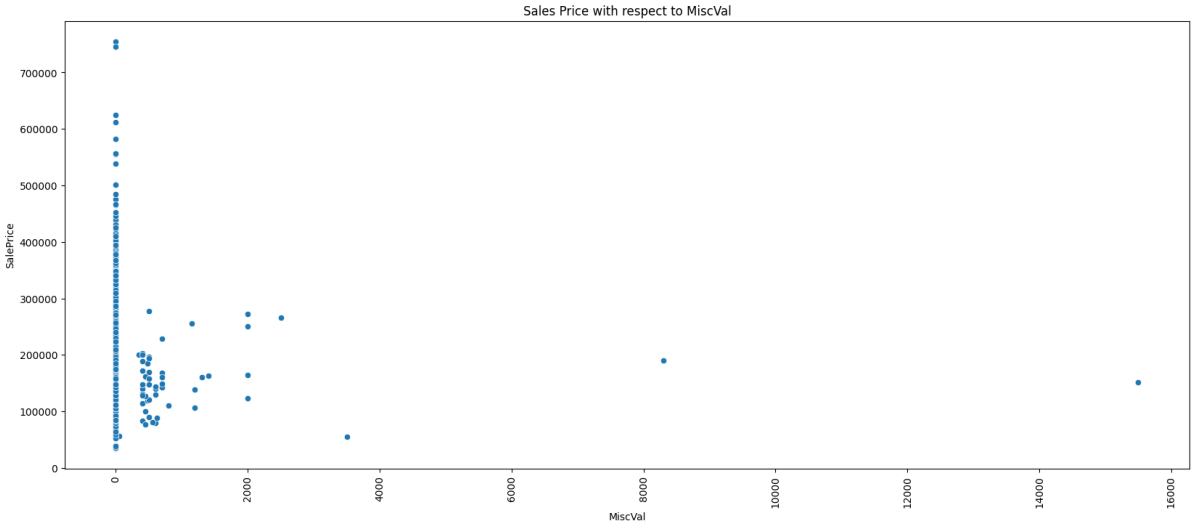

Sales Price with respect to OpenPorchSF
SalePrice
OpenPorchSF
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500

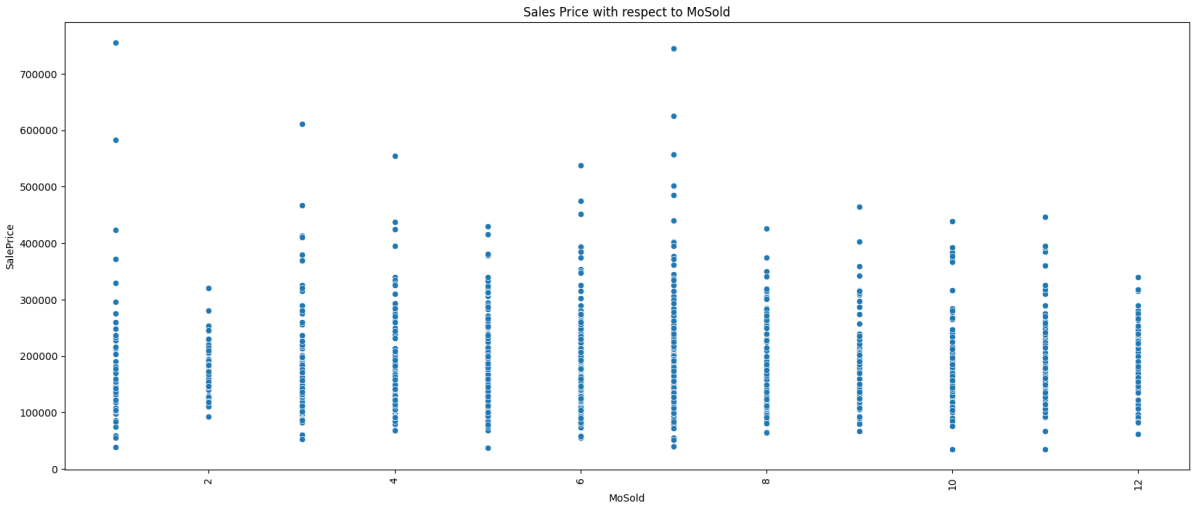

Sales Price with respect to EnclosedPorch
SalePrice
EnclosedPorch

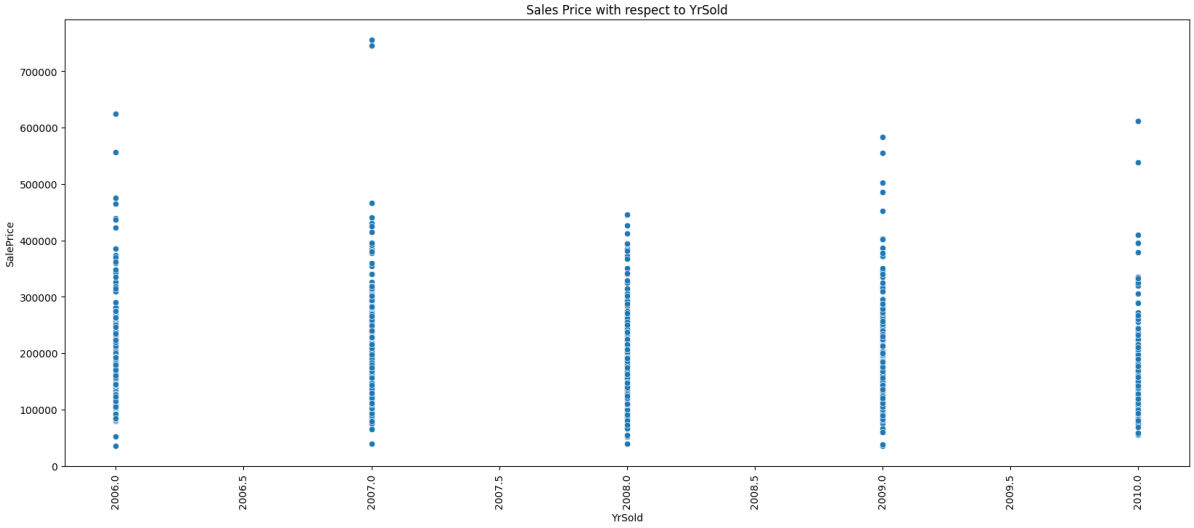

Sales Price with respect to 3SsnPorch
SalePrice
3SsnPorch

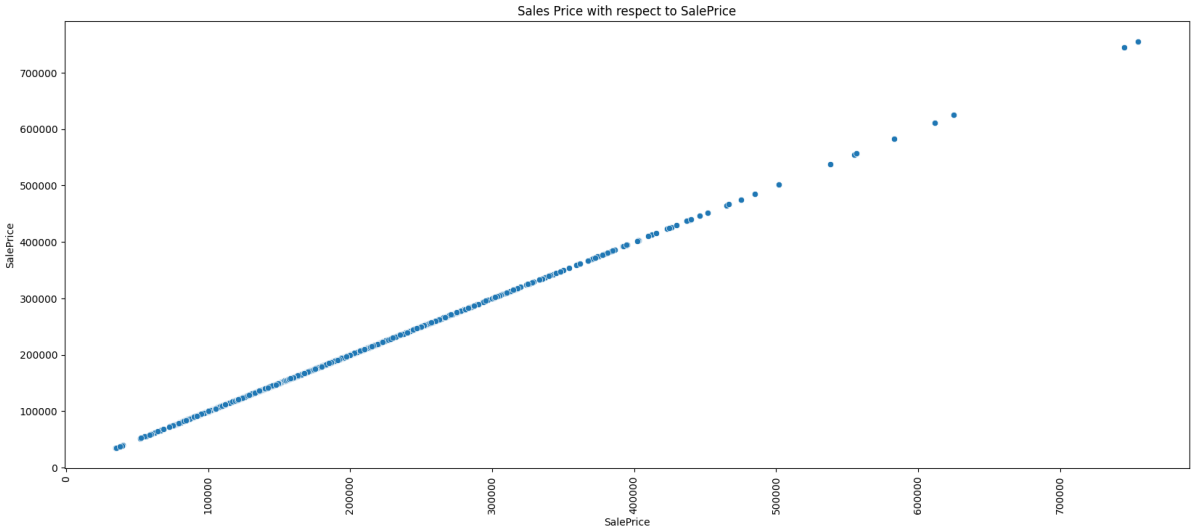

Sales Price with respect to ScreenPorch
SalePrice
ScreenPorch


Sales Price with respect to PoolArea
SalePrice
PoolArea


Sales Price with respect to MiscVal
SalePrice
MiscVal


Sales Price with respect to MoSold
SalePrice
MoSold


Sales Price with respect to YrSold
SalePrice
YrSold



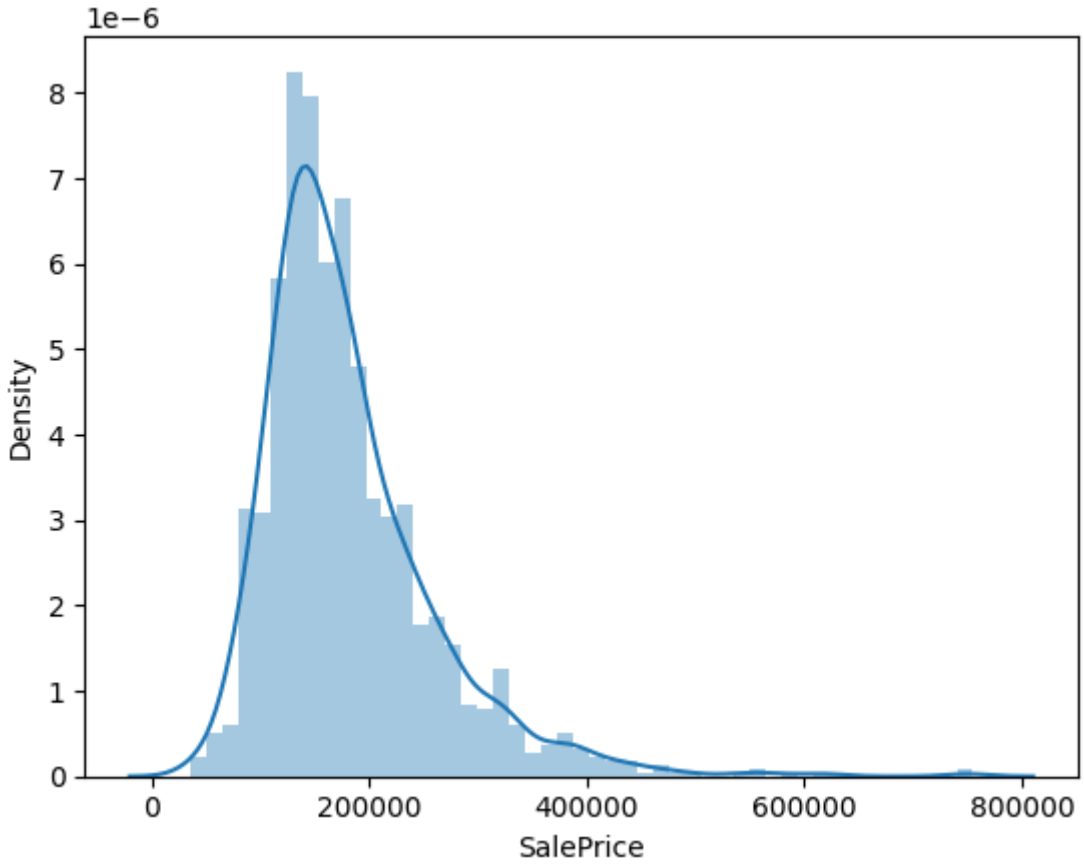
Conclusion

- SalePrice has a linear relation with GrLivingArea,1stFlrSF,2ndFlrSF,TotalBSMTSF,YearBuilt,LotArea

Checking Skewness of the Target Variable

In [ ]:
```
seaborn.distplot(housing_raw_df['SalePrice'])
```

Out[ ]:
```
<Axes: xlabel='SalePrice', ylabel='Density'>
```



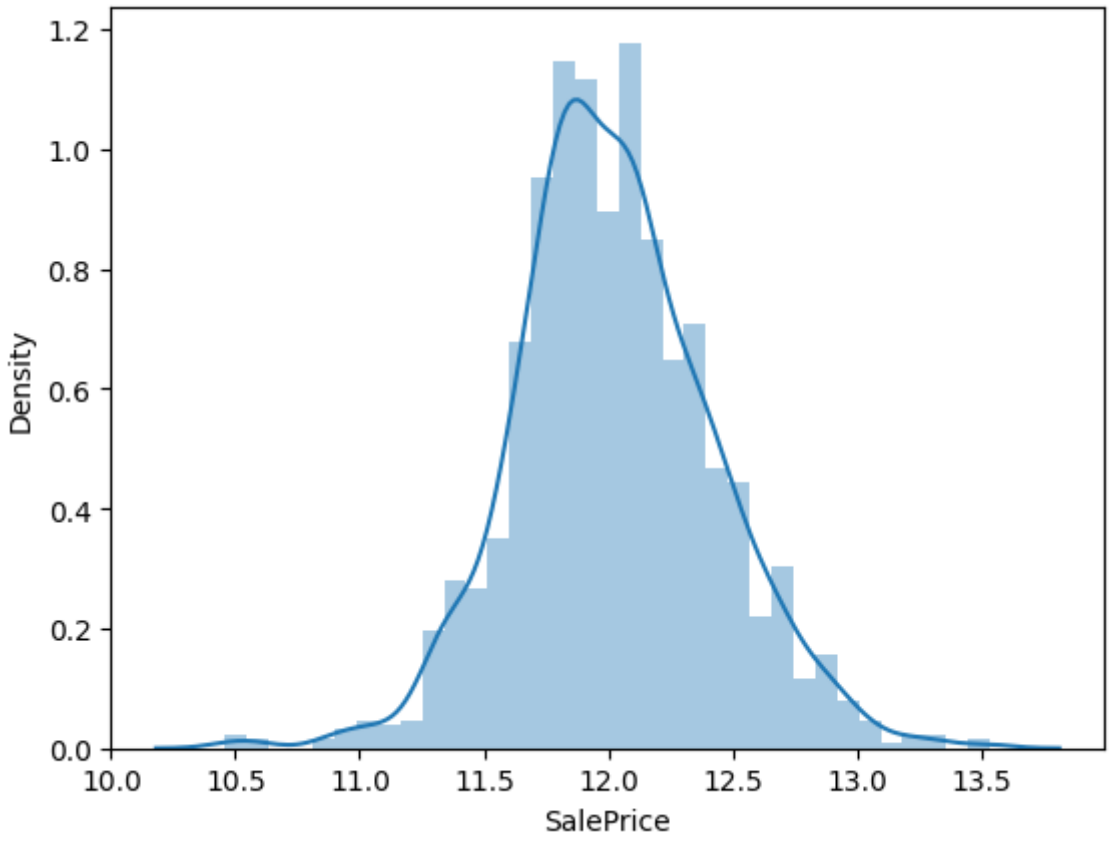
Target Variable is right Skewed so we apply log transformation to it to make it normal

In [ ]:
```
housing_raw_df['SalePrice']=numpy.log1p(housing_raw_df['SalePrice'])
```

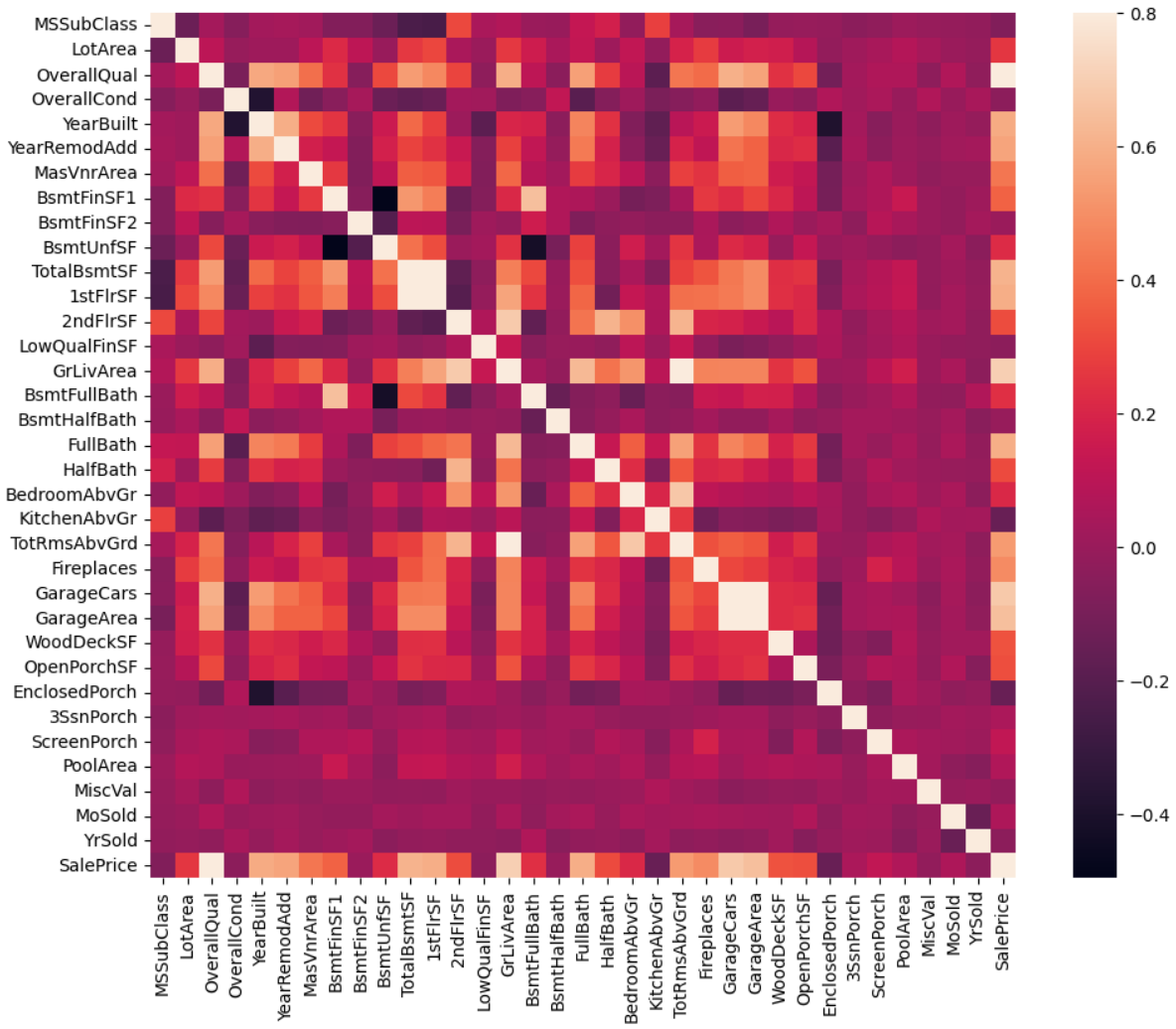
After Log Transformation the Graph is normally distributed

In [ ]:
```
seaborn.distplot(housing_raw_df['SalePrice'])
```

Out[ ]:
```
<Axes: xlabel='SalePrice', ylabel='Density'>
```

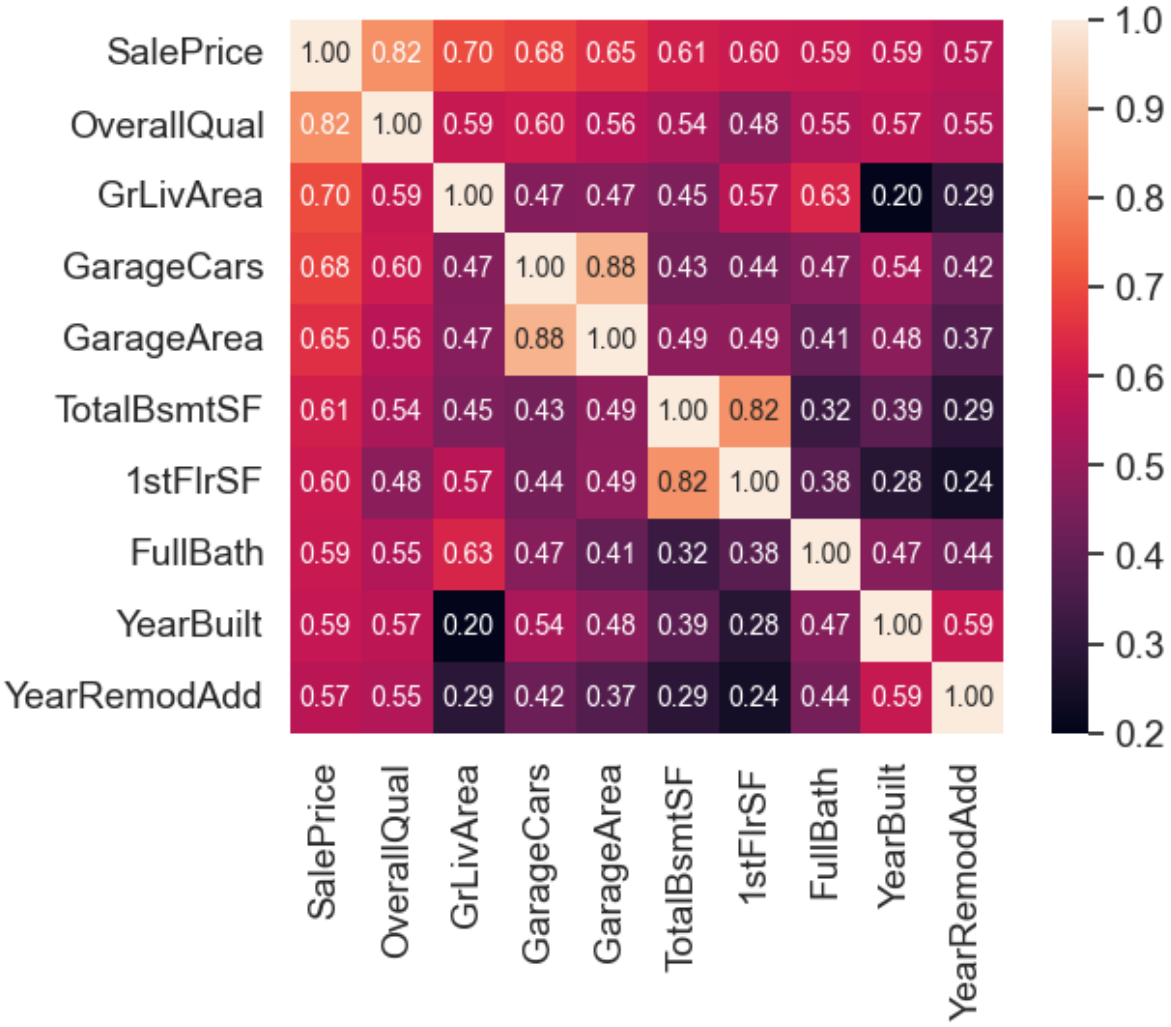


In [ ]:
```
corrmat = housing_raw_df.corr()
f, ax = plt.subplots(figsize=(12, 9))
seaborn.heatmap(corrmat, vmax=.8, square=True);
```



Getting top 10 most Correlated Features with Sale Price

In [ ]:
```
k = 10 #number of variables for heatmap
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = numpy.corrcoef(housing_raw_df[cols].values.T)
seaborn.set(font_scale=1.25)
hm = seaborn.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annc
plt.show()
```



Since there is high correlation between some of the variables.We will drop them

In [ ]:
```
corr_columns=['GarageCars','1stFlrSF','TotRmsAbvGrd']
housing_raw_df.drop(columns=corr_columns,axis=1,inplace=True)
```

Preparing Data for Model

In [ ]:
```
housing_raw_df['Age'] = (housing_raw_df['YrSold'] - housing_raw_df['YearBuil
```

In [ ]:
```
housing_raw_df.drop(['MoSold'], axis = 1,inplace=True)
housing_raw_df.drop(['YrSold'], axis = 1,inplace=True)
housing_raw_df.drop(['YearBuilt'], axis = 1,inplace=True)
housing_raw_df.drop(['YearRemodAdd'], axis = 1,inplace=True)
```

In [ ]:
```
housing_raw_df.dtypes.value_counts()
```

Out[ ]:
```
object     34
int64      27
float64     2
dtype: int64
```

In [ ]:
```
num_col=housing_raw_df.select_dtypes(exclude='object')
cat_col=housing_raw_df.select_dtypes(include='object')
```

In [ ]:
```
num_col_list = list(num_col.columns)
print("Numerical Columns")
print(num_col_list)
cat_col_list = list(cat_col.columns)
print("Categorical Columns")
print(cat_col_list)
```

```
Numerical Columns
['MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'MasVnrArea', 'BsmtF
inSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '2ndFlrSF', 'LowQualFinS
F', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'Be
droomAbvGr', 'KitchenAbvGr', 'Fireplaces', 'GarageArea', 'WoodDeckSF', 'Open
PorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVa
l', 'SalePrice', 'Age']
Categorical Columns
['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseS
tyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposur
e', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'E
lectrical', 'KitchenQual', 'Functional', 'PavedDrive', 'SaleType', 'SaleCond
ition']
```

Creating Dummy Variables

In [ ]:
```
housing_raw_df = pandas.get_dummies(housing_raw_df, drop_first=True )
housing_raw_df.head()
```

Out[ ]:

	MSSubClass	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	BsmtFinSF2	E
0	60	8450	7	5	196.0	706	0	
1	20	9600	6	8	0.0	978	0	
2	60	11250	7	5	162.0	486	0	
3	70	9550	7	5	0.0	216	0	
4	60	14260	8	5	350.0	655	0	

5 rows × 216 columns

Getting the Target Variables

In [ ]:
```
X = housing_raw_df.drop(['SalePrice'], axis=1)

X.head()
```

Out[ ]:

	MSSubClass	LotArea	OverallQual	OverallCond	MasVnrArea	BsmtFinSF1	BsmtFinSF2	E
0	60	8450	7	5	196.0	706	0	
1	20	9600	6	8	0.0	978	0	
2	60	11250	7	5	162.0	486	0	
3	70	9550	7	5	0.0	216	0	
4	60	14260	8	5	350.0	655	0	

5 rows × 215 columns

In [ ]:
```
y = housing_raw_df['SalePrice']

y.head()
```

Out[ ]:
```
0    12.247699
1    12.109016
2    12.317171
3    11.849405
4    12.429220
Name: SalePrice, dtype: float64
```

Splitting the Test and Train Data

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, te
```

Scaling the Data

In [ ]:
```
scaler = StandardScaler()

X_train[X_train.columns] = scaler.fit_transform(X_train[X_train.columns])

X_test[X_train.columns] = scaler.fit_transform(X_test[X_train.columns])
```

In [ ]:
```
print("X_train", X_train.shape)
print("y_train", y_train.shape)
```

```
X_train (1168, 215)
y_train (1168,)
```

Model Creation

Linear Regression

In [ ]:
```
LR=LinearRegression()
LR.fit(X_train,y_train)
LR.coef_
```

Out[ ]:
```
array([-3.00243285e-02,  2.45858441e-02,  5.82777074e-02,  4.64712478e-02,
        3.40386087e-03, -2.56198804e+10, -8.82981849e+09, -2.49107588e+10,
        2.45933100e+10, -1.24733188e-02,  3.90529235e-03,  1.29668229e-01,
        1.43016092e-02,  7.36551587e-04,  1.13625998e-02,  1.53643634e-02,
        1.17387914e-02, -8.97683749e-03,  1.75000281e-02,  4.14383147e-02,
        1.07305057e-02,  2.27059559e-03,  5.07733199e-03,  6.69974776e-03,
        1.12849467e-02,  1.34457380e-03,  2.13114600e-03, -6.25117974e-02,
        7.47177855e-02,  3.85394208e-02,  1.30775566e-01,  1.03982414e-01,
        2.94880594e-03,  7.51632452e-03,  2.56752968e-03,  1.86991692e-03,
        8.43858719e-03, -2.47597694e-04,  1.02753639e-02, -6.53791428e-03,
        7.70521164e-03, -7.39812851e-03, -7.35092163e-03, -5.40447235e-03,
       -6.43253326e-04, -1.34761035e-02, -3.52644920e-03, -1.18728876e-02,
       -4.33516502e-03,  1.03831291e-03, -1.35858059e-02,  1.46534443e-02,
       -2.62804031e-02, -7.93170929e-03, -1.23758316e-02, -2.08063126e-02,
       -1.28255486e-02, -2.28588581e-02,  1.92165375e-03, -1.50642395e-02,
        8.10384750e-04,  9.95492935e-03, -1.80058479e-02, -2.64883041e-03,
       -9.72807407e-03, -1.05607510e-02, -4.42314148e-03,  1.42886639e-02,
       -2.25782394e-03,  3.69930267e-03,  5.37061691e-03,  2.49755383e-02,
        2.43604183e-03,  8.56423378e-03, -3.18717957e-03,  5.81192970e-03,
        2.96729803e-03,  5.89585304e-03,  1.20474100e-02,  4.77933884e-03,
        4.74810600e-03, -3.53622437e-02, -1.11988783e-02,  1.55925751e-04,
        2.59828568e-03,  1.55043602e-02, -8.44383240e-03, -4.25648689e-03,
        3.71646881e-03,  3.12471390e-03, -1.54767036e-02, -4.50229645e-03,
        8.93497467e-03, -4.95362282e-03, -5.18798828e-04,  1.43456459e-03,
       -3.83317471e-02, -4.67765331e-03, -3.81877422e-02, -2.95639038e-04,
        9.87839699e-03,  3.31998110e-01, -5.78744839e+07,  8.05516243e-02,
        7.93714523e-02,  2.28806138e-01,  1.33271217e-01,  1.58191204e-01,
        1.80673599e-03, -6.48355484e-03,  1.71852112e-02, -4.51807088e+09,
       -9.03797150e-03,  1.06620789e-03, -1.91450119e-04,  2.03509331e-02,
        6.61849976e-04,  3.16262245e-03,  2.74753571e-03,  4.84943390e-03,
       -5.24318218e-03, -5.74445724e-03, -5.24520874e-04, -2.58064270e-03,
       -4.22859192e-03,  4.51807088e+09,  1.97381973e-02,  3.34024429e-03,
        2.62713432e-03, -9.07659531e-03, -2.77185440e-03, -1.52587891e-05,
       -3.04174423e-03,  4.59122658e-03,  7.23052025e-03,  1.42225027e-02,
        2.66647339e-03,  6.59275055e-03,  4.81987000e-03,  7.15613365e-03,
       -1.58643723e-03, -4.72760201e-03, -2.98762321e-03,  2.16960907e-04,
       -1.16720200e-02, -2.33235359e-02, -3.15153599e-03, -1.77131891e-02,
        7.01689720e-03,  1.97770596e-02, -8.56971741e-03,  5.29122353e-03,
       -6.02412224e-03, -6.02686405e-03, -1.84395313e-02, -2.65111923e-02,
        4.81239403e+10,  5.28550148e-03,  8.74006748e-03,  1.21402740e-02,
       -1.47360666e+11,  8.86207819e-03, -3.98826599e-03, -6.95419312e-03,
        2.50017755e+10, -2.98249722e-03,  4.76813316e-03, -4.52589989e-03,
       -2.82740593e-03, -6.13260269e-03,  2.23934042e+10, -8.76331329e-03,
        2.37309933e-03, -3.89468670e-03, -1.30379200e-03, -4.75883484e-04,
        5.18415462e+10,  2.46586800e-02,  2.30991840e-02, -3.23867798e-03,
        4.65726852e-03,  1.93967819e-02, -1.22594833e-03, -1.04753971e-02,
       -2.15840340e-03, -1.35083199e-02,  1.98678970e-02, -9.92774963e-04,
       -2.73013115e-03, -1.72521754e-01, -2.17843056e-03, -8.51249695e-03,
       -3.01704407e-02, -3.56459618e-02, -1.38053894e-02,  5.38563728e-03,
        2.30836868e-03, -6.79063797e-03, -1.00944042e-02,  1.61170959e-02,
        1.99937820e-03,  2.80058384e-03,  3.86071205e-03,  3.23259830e-03,
        1.01451874e-02, -1.78754330e-03, -1.45709515e-03,  1.37195587e-02,
        7.42626190e-03, -4.18615341e-03,  5.69820404e-03,  1.71270370e-02,
        2.73919106e-03,  2.43692398e-02,  1.38320923e-02])
```

In [ ]:
```
rfe = RFE(LR,n_features_to_select=100)
rfe = rfe.fit(X_train, y_train)
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[ ]:
```
[('MSSubClass', True, 1),
 ('LotArea', True, 1),
 ('OverallQual', True, 1),
 ('OverallCond', True, 1),
 ('MasVnrArea', False, 37),
 ('BsmtFinSF1', True, 1),
 ('BsmtFinSF2', True, 1),
 ('BsmtUnfSF', True, 1),
 ('TotalBsmtSF', True, 1),
 ('2ndFlrSF', True, 1),
 ('LowQualFinSF', False, 49),
 ('GrLivArea', True, 1),
 ('BsmtFullBath', True, 1),
 ('BsmtHalfBath', False, 106),
 ('FullBath', True, 1),
 ('HalfBath', True, 1),
 ('BedroomAbvGr', True, 1),
 ('KitchenAbvGr', True, 1),
 ('Fireplaces', True, 1),
 ('GarageArea', True, 1),
 ('WoodDeckSF', True, 1),
 ('OpenPorchSF', False, 79),
 ('EnclosedPorch', False, 29),
 ('3SsnPorch', False, 17),
 ('ScreenPorch', True, 1),
 ('PoolArea', False, 98),
 ('MiscVal', False, 81),
 ('Age', True, 1),
 ('MSZoning_FV', True, 1),
 ('MSZoning_RH', True, 1),
 ('MSZoning_RL', True, 1),
 ('MSZoning_RM', True, 1),
 ('Street_Pave', False, 53),
 ('LotShape_IR2', True, 1),
 ('LotShape_IR3', False, 72),
 ('LotShape_Reg', False, 91),
 ('LandContour_HLS', True, 1),
 ('LandContour_Low', False, 113),
 ('LandContour_Lvl', True, 1),
 ('Utilities_NoSeWa', False, 13),
 ('LotConfig_CulDSac', True, 1),
 ('LotConfig_FR2', False, 23),
 ('LotConfig_FR3', False, 20),
 ('LotConfig_Inside', False, 24),
 ('LandSlope_Mod', False, 108),
 ('LandSlope_Sev', False, 4),
 ('Neighborhood_Blueste', False, 55),
 ('Neighborhood_BrDale', False, 7),
 ('Neighborhood_BrkSide', False, 61),
 ('Neighborhood_ClearCr', False, 101),
 ('Neighborhood_CollgCr', False, 8),
 ('Neighborhood_Crawfor', True, 1),
 ('Neighborhood_Edwards', True, 1),
 ('Neighborhood_Gilbert', False, 18),
 ('Neighborhood_IDOTRR', False, 11),
 ('Neighborhood_MeadowV', True, 1),
 ('Neighborhood_Mitchel', False, 6),
 ('Neighborhood_NAmes', True, 1),
 ('Neighborhood_NPkVill', False, 93),
 ('Neighborhood_NWAmes', True, 1),
 ('Neighborhood_NoRidge', False, 102),
 ('Neighborhood_NridgHt', True, 1),
 ('Neighborhood_OldTown', False, 10),
 ('Neighborhood_SWISU', False, 67),
```

```
('Neighborhood_Sawyer', False, 12),
('Neighborhood_SawyerW', False, 9),
('Neighborhood_Somerst', False, 60),
('Neighborhood_StoneBr', True, 1),
('Neighborhood_Timber', False, 62),
('Neighborhood_Veenker', False, 47),
('Condition1_Feedr', False, 42),
('Condition1_Norm', True, 1),
('Condition1_PosA', False, 83),
('Condition1_PosN', True, 1),
('Condition1_RRAe', True, 1),
('Condition1_RRAn', False, 41),
('Condition1_RRNe', False, 63),
('Condition1_RRNn', False, 40),
('Condition2_Feedr', True, 1),
('Condition2_Norm', False, 65),
('Condition2_PosA', False, 45),
('Condition2_PosN', True, 1),
('Condition2_RRAe', False, 2),
('Condition2_RRAn', False, 115),
('Condition2_RRNn', False, 73),
('BldgType_2fmCon', True, 1),
('BldgType_Duplex', True, 1),
('BldgType_Twnhs', True, 1),
('BldgType_TwnhsE', False, 82),
('HouseStyle_1.5Unf', False, 71),
('HouseStyle_1Story', True, 1),
('HouseStyle_2.5Fin', False, 48),
('HouseStyle_2.5Unf', True, 1),
('HouseStyle_2Story', False, 28),
('HouseStyle_SFoyer', False, 110),
('HouseStyle_SLvl', False, 87),
('RoofStyle_Gable', True, 1),
('RoofStyle_Gambrel', False, 30),
('RoofStyle_Hip', True, 1),
('RoofStyle_Mansard', False, 112),
('RoofStyle_Shed', False, 3),
('RoofMatl_CompShg', True, 1),
('RoofMatl_Membran', True, 1),
('RoofMatl_Metal', True, 1),
('RoofMatl_Roll', True, 1),
('RoofMatl_Tar&Grv', True, 1),
('RoofMatl_WdShake', True, 1),
('RoofMatl_WdShngl', True, 1),
('Exterior1st_AsphShn', False, 95),
('Exterior1st_BrkComm', True, 1),
('Exterior1st_BrkFace', True, 1),
('Exterior1st_CBlock', True, 1),
('Exterior1st_CemntBd', False, 25),
('Exterior1st_HdBoard', False, 104),
('Exterior1st_ImStucc', False, 114),
('Exterior1st_MetalSd', True, 1),
('Exterior1st_Plywood', False, 105),
('Exterior1st_Stone', False, 50),
('Exterior1st_Stucco', False, 21),
('Exterior1st_VinylSd', False, 22),
('Exterior1st_Wd Sdng', True, 1),
('Exterior1st_WdShing', False, 31),
('Exterior2nd_AsphShn', False, 107),
('Exterior2nd_Brk Cmn', False, 92),
('Exterior2nd_BrkFace', False, 46),
('Exterior2nd_CBlock', True, 1),
('Exterior2nd_CmentBd', False, 14),
('Exterior2nd_HdBoard', False, 76),
```

```
('Exterior2nd_ImStucc', False, 77),
('Exterior2nd_MetalSd', True, 1),
('Exterior2nd_Other', False, 52),
('Exterior2nd_Plywood', False, 116),
('Exterior2nd_Stone', False, 51),
('Exterior2nd_Stucco', False, 68),
('Exterior2nd_VinylSd', False, 75),
('Exterior2nd_Wd Sdng', True, 1),
('Exterior2nd_Wd Shng', False, 78),
('MasVnrType_BrkFace', True, 1),
('MasVnrType_None', True, 1),
('MasVnrType_Stone', True, 1),
('MasVnrType_none', False, 94),
('ExterQual_Fa', False, 32),
('ExterQual_Gd', False, 64),
('ExterQual_TA', False, 111),
('ExterCond_Fa', True, 1),
('ExterCond_Gd', True, 1),
('ExterCond_Po', False, 33),
('ExterCond_TA', True, 1),
('Foundation_CBlock', False, 26),
('Foundation_PConc', True, 1),
('Foundation_Slab', True, 1),
('Foundation_Stone', False, 27),
('Foundation_Wood', False, 16),
('BsmtQual_Fa', True, 1),
('BsmtQual_Gd', True, 1),
('BsmtQual_TA', True, 1),
('BsmtQual_none', True, 1),
('BsmtCond_Gd', False, 15),
('BsmtCond_Po', False, 5),
('BsmtCond_TA', True, 1),
('BsmtCond_none', True, 1),
('BsmtExposure_Gd', True, 1),
('BsmtExposure_Mn', False, 39),
('BsmtExposure_No', False, 38),
('BsmtExposure_none', True, 1),
('BsmtFinType1_BLQ', False, 58),
('BsmtFinType1_GLQ', True, 1),
('BsmtFinType1_LwQ', False, 57),
('BsmtFinType1_Rec', False, 59),
('BsmtFinType1_Unf', False, 56),
('BsmtFinType1_none', True, 1),
('BsmtFinType2_BLQ', True, 1),
('BsmtFinType2_GLQ', False, 70),
('BsmtFinType2_LwQ', False, 44),
('BsmtFinType2_Rec', False, 97),
('BsmtFinType2_Unf', False, 109),
('BsmtFinType2_none', True, 1),
('Heating_GasA', True, 1),
('Heating_GasW', True, 1),
('Heating_Grav', False, 69),
('Heating_OthW', False, 19),
('Heating_Wall', True, 1),
('HeatingQC_Fa', False, 99),
('HeatingQC_Gd', True, 1),
('HeatingQC_Po', False, 84),
('HeatingQC_TA', True, 1),
('CentralAir_Y', True, 1),
('Electrical_FuseF', False, 100),
('Electrical_FuseP', False, 66),
('Electrical_Mix', False, 103),
('Electrical_SBrkr', False, 89),
('KitchenQual_Fa', True, 1),
```

```
('KitchenQual_Gd', True, 1),
('KitchenQual_TA', True, 1),
('Functional_Maj2', True, 1),
('Functional_Min1', False, 54),
('Functional_Min2', False, 74),
('Functional_Mod', True, 1),
('Functional_Sev', True, 1),
('Functional_Typ', True, 1),
('PavedDrive_P', False, 86),
('PavedDrive_Y', False, 85),
('SaleType_CWD', False, 35),
('SaleType_Con', False, 36),
('SaleType_ConLD', True, 1),
('SaleType_ConLI', False, 90),
('SaleType_ConLw', False, 96),
('SaleType_New', True, 1),
('SaleType_Oth', True, 1),
('SaleType_WD', False, 88),
('SaleCondition_AdjLand', False, 34),
('SaleCondition_Alloca', True, 1),
('SaleCondition_Family', False, 80),
('SaleCondition_Normal', True, 1),
('SaleCondition_Partial', False, 43)]
```

In [ ]:
```
col = X_train.columns[rfe.support_]
X_train_rfe = X_train[col]
X_test_rfe=X_test[col]
X_train_rfe.head()
```

Out[ ]:

	MSSubClass	LotArea	OverallQual	OverallCond	BsmtFinSF1	BsmtFinSF2	BsmtUnf
254	-0.866764	-0.212896	-0.820445	0.372217	1.037269	-0.285504	-0.4002
1066	0.074110	-0.265245	-0.088934	1.268609	-0.971996	-0.285504	0.5119
638	-0.631546	-0.177841	-0.820445	1.268609	-0.971996	-0.285504	0.5051
799	-0.161109	-0.324474	-0.820445	1.268609	0.267995	-0.285504	-0.9157
380	-0.161109	-0.529035	-0.820445	0.372217	-0.496920	-0.285504	0.5320

5 rows × 100 columns

In [ ]:
```
X_train_rfe.shape
```

Out[ ]:
```
(1168, 100)
```

Retraining the model on the RFE Train Dataset

In [ ]:
```
LR.fit(X_train_rfe,y_train)
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

Performing Prediction on the Test Dataset with the columns supporting RFE

In [ ]:
```
y_pred=LR.predict(X_test_rfe)
print("R2 Score for the LR {}".format(r2_score(y_test,y_pred)))
```

```
print("MSE for the LR {}".format(mean_squared_error(y_test,y_pred)))
print("RMSE Score for the LR {}".format(numpy.sqrt(mean_squared_error(y_test
df_lr=pandas.DataFrame(data=[[r2_score(y_test,y_pred),mean_squared_error(y_t
```

```
R2 Score for the LR -2.4078182176572294e+21
MSE for the LR 4.4932533793969465e+20
RMSE Score for the LR 21197295533.62161
```

Lasso Regression

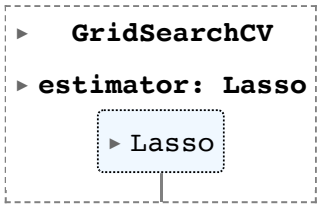
We have made use of HyperParameter tuning to get the best result for alpha based on which we will create the model.For Hyperparameter tuning we have made use GridSearchCV

In [ ]:
```
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}
lasso = Lasso()

list_var=['OverallQual','GrLivArea','GarageArea','OverallCond','BsmtFullBath
X_test.drop(columns=list_var,inplace=True)
X_train.drop(columns=list_var,inplace=True)
# cross validation
folds = 5
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'r2',
                        cv = folds,
                        return_train_score=True)

model_cv.fit(X_train, y_train)
```

Out[ ]:



Getting the results for different values for alpha

In [ ]:
```
cv_results = pandas.DataFrame(model_cv.cv_results_)
cv_results = cv_results[cv_results['param_alpha']<=1]
cv_results.head()
```

Out[ ]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split(
0	0.176825	0.038668	0.007344	0.002424	0.0001	{'alpha': 0.0001}	
1	0.072115	0.058419	0.006807	0.001135	0.001	{'alpha': 0.001}	
2	0.014078	0.001535	0.005929	0.000957	0.01	{'alpha': 0.01}	
3	0.011271	0.000576	0.005822	0.000631	0.05	{'alpha': 0.05}	
4	0.009864	0.001271	0.006560	0.000377	0.1	{'alpha': 0.1}	

5 rows × 21 columns

In [ ]:
```
estimator=model_cv.best_estimator_
print("The best value for alpha for Lasso Regression is {}".format(estimator
```

```
The best value for alpha for Lasso Regression is Lasso(alpha=0.01)
```

As it can be seen that we are getting better results with alpha .01

In [ ]:
```
alpha = 0.01
lasso = Lasso(alpha=alpha)

lasso.fit(X_train, y_train)
lasso.coef_
```

Out[ ]:
```
array([-0.00457713,  0.01340231,  0.00628023,  0.00181155,  0.        ,
       -0.        ,  0.09876522,  0.06695656,  0.        , -0.        ,
        0.04826563,  0.02337321,  0.00769097, -0.00931789,  0.04631898,
        0.01659052,  0.        ,  0.        ,  0.00131629,  0.008856  ,
       -0.        , -0.        , -0.03188293,  0.        , -0.        ,
        0.        , -0.00910799,  0.        ,  0.00206902, -0.00818136,
       -0.        ,  0.0060383 , -0.        ,  0.        , -0.        ,
        0.01150222, -0.        , -0.        , -0.        ,  0.        ,
        0.        , -0.        , -0.00265056, -0.        ,  0.        ,
        0.        ,  0.0225657 , -0.02045853, -0.        , -0.00438856,
       -0.01215167, -0.        , -0.        , -0.        ,  0.        ,
        0.0092138 ,  0.03461057, -0.        , -0.        , -0.        ,
       -0.        ,  0.00977362,  0.0201722 ,  0.        ,  0.00386879,
       -0.0078088 ,  0.        ,  0.        ,  0.        , -0.        ,
        0.        , -0.        ,  0.        ,  0.        ,  0.        ,
        0.00171674, -0.00874233, -0.        , -0.        ,  0.        ,
        0.        , -0.00040364, -0.01905624, -0.01103528, -0.        ,
       -0.        ,  0.0036792 ,  0.        , -0.        , -0.        ,
        0.00063753, -0.01742973, -0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -0.        ,
        0.        ,  0.        ,  0.00597315,  0.        , -0.00338898,
        0.01043989, -0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.        ,  0.        , -0.        ,  0.        ,
        0.        , -0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.00136901, -0.        ,  0.        , -0.        ,
        0.        , -0.        ,  0.        , -0.        ,  0.        ,
        0.        , -0.00019375,  0.        , -0.        ,  0.00329169,
       -0.        , -0.01731703,  0.        , -0.02663565, -0.00217648,
        0.        , -0.00101703,  0.        , -0.        ,  0.01195207,
       -0.        ,  0.        ,  0.        , -0.        , -0.        ,
       -0.01159466, -0.        ,  0.        , -0.        ,  0.        ,
       -0.        ,  0.00970226, -0.        , -0.01487253, -0.        ,
        0.        ,  0.02181825, -0.        , -0.        , -0.00369408,
       -0.        , -0.        ,  0.        ,  0.        , -0.        ,
       -0.        , -0.        , -0.        ,  0.        , -0.01099162,
        0.        , -0.        , -0.        , -0.00148022, -0.        ,
       -0.0061594 ,  0.03096976, -0.        , -0.        ,  0.        ,
        0.00357375, -0.01148858, -0.        , -0.0342847 , -0.00826804,
        0.        , -0.        , -0.00167334, -0.00044249,  0.00503502,
        0.        ,  0.00415497,  0.        ,  0.        ,  0.        ,
       -0.        , -0.        ,  0.        , -0.        , -0.        ,
       -0.        ,  0.00298309, -0.        ,  0.        ,  0.00838619])
```

Printing the coffecient values for the Lasso Regression.As it can be seen some of the coffecients have been set to 0

In [ ]:
```
model_parameters = list(lasso.coef_ )
model_parameters.insert(0, lasso.intercept_)
model_parameters = [round(x, 3) for x in model_parameters]
cols = X.columns
cols = cols.insert(0, "constant")
sorted(list(zip(cols, model_parameters)),key=(lambda x:-(x[1])))
```

Out[ ]:
```
[('constant', 12.031),
 ('BsmtFinSF2', 0.099),
 ('BsmtUnfSF', 0.067),
 ('LowQualFinSF', 0.048),
 ('FullBath', 0.046),
 ('Neighborhood_Mitchel', 0.035),
 ('Heating_Wall', 0.031),
 ('GrLivArea', 0.023),
 ('Neighborhood_Blueste', 0.023),
 ('BsmtExposure_Gd', 0.022),
 ('Neighborhood_OldTown', 0.02),
 ('HalfBath', 0.017),
 ('LotArea', 0.013),
 ('LotShape_Reg', 0.012),
 ('ExterCond_Fa', 0.012),
 ('Neighborhood_NridgHt', 0.01),
 ('RoofMatl_Tar&Grv', 0.01),
 ('BsmtQual_none', 0.01),
 ('GarageArea', 0.009),
 ('Neighborhood_MeadowV', 0.009),
 ('BsmtFullBath', 0.008),
 ('SaleType_WD', 0.008),
 ('OverallQual', 0.006),
 ('MSZoning_RM', 0.006),
 ('RoofMatl_Membran', 0.006),
 ('Functional_Maj2', 0.005),
 ('Neighborhood_Sawyer', 0.004),
 ('BldgType_Duplex', 0.004),
 ('HeatingQC_TA', 0.004),
 ('Functional_Min2', 0.004),
 ('Exterior2nd_VinylSd', 0.003),
 ('SaleType_ConLw', 0.003),
 ('OverallCond', 0.002),
 ('MSZoning_FV', 0.002),
 ('Condition1_RRAn', 0.002),
 ('Fireplaces', 0.001),
 ('HouseStyle_1Story', 0.001),
 ('Exterior1st_WdShing', 0.001),
 ('MasVnrArea', 0.0),
 ('BsmtFinSF1', -0.0),
 ('TotalBsmtSF', 0.0),
 ('2ndFlrSF', -0.0),
 ('BedroomAbvGr', 0.0),
 ('KitchenAbvGr', 0.0),
 ('WoodDeckSF', -0.0),
 ('OpenPorchSF', -0.0),
 ('3SsnPorch', 0.0),
 ('ScreenPorch', -0.0),
 ('PoolArea', 0.0),
 ('Age', 0.0),
 ('MSZoning_RL', -0.0),
 ('Street_Pave', -0.0),
 ('LotShape_IR2', 0.0),
 ('LotShape_IR3', -0.0),
 ('LandContour_HLS', -0.0),
 ('LandContour_Low', -0.0),
 ('LandContour_Lvl', -0.0),
 ('Utilities_NoSeWa', 0.0),
 ('LotConfig_CulDSac', 0.0),
 ('LotConfig_FR2', -0.0),
 ('LotConfig_Inside', -0.0),
 ('LandSlope_Mod', 0.0),
 ('LandSlope_Sev', 0.0),
 ('Neighborhood_BrkSide', -0.0),
```

```
('Neighborhood_Crawfor', -0.0),
('Neighborhood_Edwards', -0.0),
('Neighborhood_Gilbert', -0.0),
('Neighborhood_IDOTRR', 0.0),
('Neighborhood_NAmes', -0.0),
('Neighborhood_NPkVill', -0.0),
('Neighborhood_NWAmes', -0.0),
('Neighborhood_NoRidge', -0.0),
('Neighborhood_SWISU', 0.0),
('Neighborhood_Somerst', 0.0),
('Neighborhood_StoneBr', 0.0),
('Neighborhood_Timber', 0.0),
('Neighborhood_Veenker', -0.0),
('Condition1_Feedr', 0.0),
('Condition1_Norm', -0.0),
('Condition1_PosA', 0.0),
('Condition1_PosN', 0.0),
('Condition1_RRAe', 0.0),
('Condition1_RRNn', -0.0),
('Condition2_Feedr', -0.0),
('Condition2_Norm', 0.0),
('Condition2_PosA', 0.0),
('Condition2_PosN', -0.0),
('Condition2_RRNn', -0.0),
('BldgType_2fmCon', -0.0),
('BldgType_Twnhs', 0.0),
('BldgType_TwnhsE', -0.0),
('HouseStyle_1.5Unf', -0.0),
('HouseStyle_2.5Unf', -0.0),
('HouseStyle_2Story', 0.0),
('HouseStyle_SFoyer', 0.0),
('HouseStyle_SLvl', 0.0),
('RoofStyle_Gable', 0.0),
('RoofStyle_Gambrel', 0.0),
('RoofStyle_Hip', 0.0),
('RoofStyle_Mansard', -0.0),
('RoofStyle_Shed', 0.0),
('RoofMatl_CompShg', 0.0),
('RoofMatl_Metal', 0.0),
('RoofMatl_WdShake', -0.0),
('RoofMatl_WdShngl', 0.0),
('Exterior1st_AsphShn', -0.0),
('Exterior1st_BrkComm', 0.0),
('Exterior1st_BrkFace', -0.0),
('Exterior1st_CBlock', 0.0),
('Exterior1st_CemntBd', 0.0),
('Exterior1st_HdBoard', -0.0),
('Exterior1st_ImStucc', 0.0),
('Exterior1st_MetalSd', 0.0),
('Exterior1st_Plywood', -0.0),
('Exterior1st_Stone', 0.0),
('Exterior1st_Stucco', -0.0),
('Exterior1st_VinylSd', 0.0),
('Exterior1st_Wd Sdng', -0.0),
('Exterior2nd_AsphShn', -0.0),
('Exterior2nd_Brk Cmn', 0.0),
('Exterior2nd_BrkFace', -0.0),
('Exterior2nd_CBlock', 0.0),
('Exterior2nd_CmentBd', -0.0),
('Exterior2nd_HdBoard', 0.0),
('Exterior2nd_ImStucc', -0.0),
('Exterior2nd_MetalSd', 0.0),
('Exterior2nd_Other', 0.0),
('Exterior2nd_Plywood', -0.0),
```

```
('Exterior2nd_Stone', 0.0),
('Exterior2nd_Stucco', -0.0),
('Exterior2nd_Wd Sdng', -0.0),
('MasVnrType_BrkFace', 0.0),
('MasVnrType_none', 0.0),
('ExterQual_Gd', 0.0),
('ExterQual_TA', -0.0),
('ExterCond_Gd', -0.0),
('ExterCond_Po', 0.0),
('ExterCond_TA', 0.0),
('Foundation_CBlock', -0.0),
('Foundation_PConc', -0.0),
('Foundation_Stone', -0.0),
('Foundation_Wood', 0.0),
('BsmtQual_Fa', -0.0),
('BsmtQual_Gd', 0.0),
('BsmtQual_TA', -0.0),
('BsmtCond_Gd', -0.0),
('BsmtCond_TA', -0.0),
('BsmtCond_none', 0.0),
('BsmtExposure_Mn', -0.0),
('BsmtExposure_No', -0.0),
('BsmtFinType1_BLQ', -0.0),
('BsmtFinType1_GLQ', -0.0),
('BsmtFinType1_LwQ', 0.0),
('BsmtFinType1_Rec', 0.0),
('BsmtFinType1_Unf', -0.0),
('BsmtFinType1_none', -0.0),
('BsmtFinType2_BLQ', -0.0),
('BsmtFinType2_GLQ', -0.0),
('BsmtFinType2_LwQ', 0.0),
('BsmtFinType2_Unf', 0.0),
('BsmtFinType2_none', -0.0),
('Heating_GasA', -0.0),
('Heating_Grav', -0.0),
('HeatingQC_Fa', -0.0),
('HeatingQC_Gd', -0.0),
('HeatingQC_Po', 0.0),
('Electrical_FuseF', -0.0),
('Electrical_SBrkr', 0.0),
('KitchenQual_Fa', -0.0),
('KitchenQual_TA', -0.0),
('Functional_Min1', 0.0),
('Functional_Mod', 0.0),
('Functional_Sev', 0.0),
('Functional_Typ', 0.0),
('PavedDrive_P', -0.0),
('PavedDrive_Y', -0.0),
('SaleType_CWD', 0.0),
('SaleType_Con', -0.0),
('SaleType_ConLD', -0.0),
('SaleType_ConLI', -0.0),
('SaleType_New', -0.0),
('SaleType_Oth', 0.0),
('ExterQual_Fa', -0.001),
('Heating_GasW', -0.001),
('MasVnrType_Stone', -0.002),
('KitchenQual_Gd', -0.002),
('LotConfig_FR3', -0.003),
('RoofMatl_Roll', -0.003),
('Neighborhood_ClearCr', -0.004),
('BsmtExposure_none', -0.004),
('MSSubClass', -0.005),
('Heating_OthW', -0.006),
```

```
('MSZoning_RH', -0.008),
('Neighborhood_SawyerW', -0.008),
('Electrical_Mix', -0.008),
('BsmtHalfBath', -0.009),
('MiscVal', -0.009),
('Condition1_RRNe', -0.009),
('Condition2_RRAn', -0.011),
('BsmtFinType2_Rec', -0.011),
('CentralAir_Y', -0.011),
('Neighborhood_CollgCr', -0.012),
('Foundation_Slab', -0.012),
('BsmtCond_Po', -0.015),
('HouseStyle_2.5Fin', -0.017),
('Exterior2nd_Wd Shng', -0.017),
('Condition2_RRAe', -0.019),
('Neighborhood_BrDale', -0.02),
('MasVnrType_None', -0.027),
('EnclosedPorch', -0.032),
('Electrical_FuseP', -0.034)]
```

Getting the R2 score

In [ ]:
```
lm = Lasso(alpha=0.01)
lm.fit(X_train, y_train)
y_test_pred = lm.predict(X_test)
print("R2 SCore on Test data {}".format(r2_score(y_true=y_test, y_pred=y_tes
print('MSE :', (mean_squared_error(y_test, y_test_pred)))
print('RMSE :', numpy.sqrt(mean_squared_error(y_test, y_test_pred)))
df_lasso=pandas.DataFrame(data=[[r2_score(y_true=y_test, y_pred=y_test_pred)
```

```
R2 SCore on Test data 0.8437857617091579
MSE : 0.02915129343913655
RMSE : 0.1707374986320713
```

In [ ]:
```
mod = list(zip(cols, model_parameters))
para = pandas.DataFrame(mod)
para.columns = ['Variable', 'Coeff']
para = para.sort_values((['Coeff']), axis = 0, ascending = False)
para
```

Out[ ]:

	Variable	Coeff
0	constant	12.031
7	BsmtFinSF2	0.099
8	BsmtUnfSF	0.067
11	LowQualFinSF	0.048
15	FullBath	0.046
...	...	...
83	Condition2_RRAe	-0.019
48	Neighborhood_BrDale	-0.020
139	MasVnrType_None	-0.027
23	EnclosedPorch	-0.032
189	Electrical_FuseP	-0.034

211 rows × 2 columns

Ridge Regression

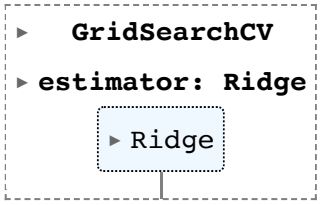
We have made use of HyperParameter tuning to get the best result for alpha based on which we will create the model.For Hyperparameter tuning we have made use GridSearchCV

In [ ]:
```
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0]}


ridge = Ridge()

# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True)
model_cv.fit(X_train, y_train)
```

Out[ ]:



Printing out the results for different value of alpha

In [ ]:
```
cv_results = pandas.DataFrame(model_cv.cv_results_)
cv_results
```

Out[ ]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split
0	0.014926	0.007467	0.006542	0.000938	0.0001	{'alpha': 0.0001}	
1	0.012791	0.001400	0.008250	0.001646	0.001	{'alpha': 0.001}	
2	0.012710	0.000547	0.007322	0.000334	0.01	{'alpha': 0.01}	
3	0.012111	0.001058	0.005348	0.000680	0.05	{'alpha': 0.05}	
4	0.011773	0.000978	0.006303	0.000389	0.1	{'alpha': 0.1}	
5	0.011377	0.000903	0.005605	0.000420	0.2	{'alpha': 0.2}	
6	0.011395	0.000712	0.006209	0.000589	0.3	{'alpha': 0.3}	
7	0.013198	0.002162	0.006413	0.001164	0.4	{'alpha': 0.4}	
8	0.061505	0.075963	0.009092	0.003402	0.5	{'alpha': 0.5}	
9	0.011335	0.001177	0.006463	0.000253	0.6	{'alpha': 0.6}	
10	0.012188	0.001183	0.006032	0.000670	0.7	{'alpha': 0.7}	
11	0.011807	0.000563	0.006345	0.000397	0.8	{'alpha': 0.8}	
12	0.011253	0.000809	0.006324	0.000889	0.9	{'alpha': 0.9}	
13	0.011514	0.000802	0.006266	0.000537	1.0	{'alpha': 1.0}	
14	0.011282	0.000775	0.006029	0.000421	2.0	{'alpha': 2.0}	

15 rows × 21 columns

In [ ]:
```
estimator=model_cv.best_estimator_
print("The best value for alpha for Ridge Regression is {}".format(estimator
```

```
The best value for alpha for Ridge Regression is Ridge(alpha=2.0)
```

We will take the value of alpha as 2

In [ ]:
```
alpha = 2
ridge = Ridge(alpha=alpha)

ridge.fit(X_train, y_train)
ridge.coef_
```

Out[ ]:
```
array([-2.37181846e-02,  2.79788670e-02,  2.64541520e-03,  5.44984268e-02,
        1.10881148e-02,  2.68925154e-02,  8.79939373e-02,  9.58889147e-02,
        1.68822193e-02, -3.38994716e-03,  3.15178663e-02,  2.42207750e-02,
        1.75055314e-02, -1.17328710e-02,  2.77484320e-02,  1.62299493e-02,
        1.65679662e-03,  2.73639683e-03,  9.95108310e-03,  1.16367742e-02,
       -1.74059810e-03,  2.55804898e-03, -3.11594913e-02,  6.31785678e-02,
        3.94889004e-02,  1.25491042e-01,  1.09159079e-01,  3.93582992e-03,
        8.76159176e-03, -1.15111842e-03, -5.16614751e-06,  9.82140186e-03,
       -3.73041013e-03,  5.89293429e-03, -4.51233030e-03,  8.57155929e-03,
       -1.41199703e-02, -8.83114527e-03, -9.30862807e-03, -8.82567549e-04,
       -1.27364884e-02, -2.94492591e-03, -1.84831384e-02, -1.47446859e-02,
       -7.78951204e-03, -1.91485561e-02,  1.46544534e-02, -4.49585368e-02,
       -1.82136186e-02, -2.26360501e-02, -3.37104583e-02, -1.87964716e-02,
       -3.09860881e-02,  3.25744151e-03, -1.13804776e-02,  5.20932475e-03,
        1.60543001e-02, -3.40779102e-02, -1.11517662e-02, -1.99168622e-02,
       -1.36599786e-02,  6.72808924e-03,  1.89715820e-02, -7.07547690e-03,
        8.74109154e-03, -7.91894126e-04,  1.56138150e-02,  8.51716821e-03,
        7.19060988e-03, -3.39010108e-03,  4.34107939e-03, -7.64256065e-04,
        4.49053253e-03,  2.25666223e-02,  1.81253342e-02,  1.38710072e-02,
       -2.35748560e-02, -1.21387625e-02,  3.95907823e-03,  9.91403945e-03,
        1.72380990e-02, -7.91820298e-03, -1.07858590e-02, -8.51552717e-03,
        4.76239859e-03, -3.04017716e-03,  1.75486355e-03,  1.12953910e-02,
       -8.54136067e-03,  2.63506485e-03,  1.62281127e-02, -5.42131502e-02,
       -8.37590328e-03, -4.28992341e-02, -3.07348786e-03,  1.34172794e-02,
        2.88779032e-01,  0.00000000e+00,  6.88205599e-02,  6.80155794e-02,
        1.98880004e-01,  1.18556625e-01,  1.41309993e-01,  5.26937593e-03,
       -6.80539023e-03,  3.00831304e-02,  9.18303455e-04, -1.78231376e-03,
        1.48820659e-02,  2.25155097e-03,  2.65838362e-02,  1.19335163e-02,
        8.82961701e-03,  1.40401171e-02,  3.89198443e-03,  1.46662668e-02,
       -2.17479727e-03,  7.16393543e-04, -4.57393326e-03, -9.32240441e-03,
        9.18303455e-04,  2.31650290e-02, -3.03800753e-03,  4.04346160e-03,
       -7.26174020e-03, -4.02435902e-04, -5.62551107e-03, -3.39561679e-03,
        1.35306868e-03,  2.34018421e-02,  8.43328370e-03, -3.04696795e-03,
        1.54449357e-02,  6.08822003e-03,  1.45340285e-02, -3.43348423e-04,
       -1.97312973e-02, -7.23624860e-03, -1.15070441e-02, -2.23915662e-02,
       -3.09763330e-02, -1.12818062e-02, -3.37117963e-02,  1.03686771e-05,
        7.50045532e-03, -1.11446762e-02,  7.33974176e-03, -1.52868181e-03,
       -1.15146209e-02, -3.41434762e-02, -4.60800251e-02,  1.87855869e-03,
        1.41440893e-02,  4.04903333e-03,  1.77458089e-02,  1.87855869e-03,
        1.03102675e-02, -2.78096078e-03, -1.37911122e-02,  1.87855869e-03,
       -4.95378736e-03,  1.03304259e-02, -1.00159067e-02, -5.26741531e-03,
       -6.50832131e-03,  1.87855869e-03, -8.78240357e-03,  4.12663078e-03,
       -5.92260608e-03, -5.24021198e-03, -1.41160686e-02,  1.87855869e-03,
        3.65902641e-02,  3.43184698e-02, -1.51197747e-03,  9.85861989e-03,
        2.09528664e-02, -7.06629062e-03, -1.48312216e-02, -6.54716030e-04,
       -1.98356346e-02,  3.31664361e-02, -1.01691091e-03,  2.75944349e-03,
        0.00000000e+00,  3.98851713e-03, -2.33814001e-02, -4.36820322e-02,
       -7.22313412e-02, -1.80614074e-02,  7.46505927e-03,  3.24856431e-03,
       -7.59648454e-03, -7.95033120e-03,  9.98239526e-03,  9.14065664e-03,
        7.47886413e-03,  2.55635566e-03,  1.97887788e-03,  8.12150806e-03,
       -5.45674247e-03, -3.95997415e-03, -1.27692491e-02,  1.45473848e-03,
       -1.29594685e-02,  2.33877487e-03,  2.29758113e-02,  3.56886432e-03,
        3.06167466e-02,  3.46746937e-02])
```

Printing the coffecients for the ridge regression

In [ ]:
```
model_parameters = list(ridge.coef_)
model_parameters.insert(0, ridge.intercept_)
model_parameters = [round(x, 3) for x in model_parameters]
cols = X.columns
cols = cols.insert(0, "constant")
sorted(list(zip(cols, model_parameters)),key=(lambda x:-(x[1])))
```

Out[ ]:
```
[('constant', 12.031),
 ('RoofStyle_Gable', 0.289),
 ('RoofStyle_Shed', 0.199),
 ('RoofMatl_Membran', 0.141),
 ('PoolArea', 0.125),
 ('RoofMatl_CompShg', 0.119),
 ('MiscVal', 0.109),
 ('BsmtUnfSF', 0.096),
 ('BsmtFinSF2', 0.088),
 ('RoofStyle_Hip', 0.069),
 ('RoofStyle_Mansard', 0.068),
 ('3SsnPorch', 0.063),
 ('OverallCond', 0.054),
 ('ScreenPorch', 0.039),
 ('BsmtFinType2_GLQ', 0.037),
 ('SaleType_WD', 0.035),
 ('BsmtFinType2_LwQ', 0.034),
 ('Heating_Wall', 0.033),
 ('LowQualFinSF', 0.032),
 ('SaleType_Oth', 0.031),
 ('RoofMatl_Tar&Grv', 0.03),
 ('LotArea', 0.028),
 ('FullBath', 0.028),
 ('BsmtFinSF1', 0.027),
 ('Exterior1st_BrkFace', 0.027),
 ('GrLivArea', 0.024),
 ('Condition1_PosN', 0.023),
 ('Exterior1st_WdShing', 0.023),
 ('Exterior2nd_MetalSd', 0.023),
 ('SaleType_ConLw', 0.023),
 ('BsmtFinType2_none', 0.021),
 ('Neighborhood_OldTown', 0.019),
 ('BsmtFullBath', 0.018),
 ('Condition1_RRAe', 0.018),
 ('BsmtQual_Gd', 0.018),
 ('TotalBsmtSF', 0.017),
 ('Condition2_PosA', 0.017),
 ('HalfBath', 0.016),
 ('Neighborhood_Mitchel', 0.016),
 ('Neighborhood_Somerst', 0.016),
 ('HouseStyle_1Story', 0.016),
 ('Neighborhood_Blueste', 0.015),
 ('Exterior1st_AsphShn', 0.015),
 ('Exterior1st_MetalSd', 0.015),
 ('Exterior2nd_Stone', 0.015),
 ('Exterior2nd_VinylSd', 0.015),
 ('Condition1_RRAn', 0.014),
 ('Exterior1st_HdBoard', 0.014),
 ('Foundation_Wood', 0.014),
 ('HouseStyle_SLvl', 0.013),
 ('GarageArea', 0.012),
 ('Exterior1st_CBlock', 0.012),
 ('MasVnrArea', 0.011),
 ('BldgType_Twnhs', 0.011),
 ('Fireplaces', 0.01),
 ('MSZoning_RM', 0.01),
 ('Condition2_Norm', 0.01),
 ('BsmtQual_none', 0.01),
 ('BsmtExposure_Gd', 0.01),
 ('BsmtFinType2_Unf', 0.01),
 ('Functional_Maj2', 0.01),
 ('MSZoning_FV', 0.009),
 ('LotShape_Reg', 0.009),
 ('Neighborhood_Sawyer', 0.009),
```

```
('Neighborhood_StoneBr', 0.009),
('Exterior1st_CemntBd', 0.009),
('Functional_Min1', 0.009),
('Exterior2nd_Other', 0.008),
('ExterCond_Fa', 0.008),
('Functional_Typ', 0.008),
('Neighborhood_NridgHt', 0.007),
('Neighborhood_Timber', 0.007),
('ExterCond_Po', 0.007),
('Electrical_SBrkr', 0.007),
('Functional_Min2', 0.007),
('LotShape_IR2', 0.006),
('Exterior2nd_Stucco', 0.006),
('Neighborhood_MeadowV', 0.005),
('Condition2_RRNn', 0.005),
('RoofMatl_Metal', 0.005),
('Age', 0.004),
('Condition1_Feedr', 0.004),
('Condition1_PosA', 0.004),
('Condition2_Feedr', 0.004),
('Exterior1st_ImStucc', 0.004),
('Exterior2nd_Brk Cmn', 0.004),
('BsmtQual_Fa', 0.004),
('BsmtFinType1_LwQ', 0.004),
('HeatingQC_TA', 0.004),
('SaleType_New', 0.004),
('OverallQual', 0.003),
('KitchenAbvGr', 0.003),
('OpenPorchSF', 0.003),
('Neighborhood_Gilbert', 0.003),
('HouseStyle_1.5Unf', 0.003),
('HeatingQC_Gd', 0.003),
('KitchenQual_Fa', 0.003),
('Functional_Mod', 0.003),
('BedroomAbvGr', 0.002),
('BldgType_Duplex', 0.002),
('Exterior1st_BrkComm', 0.002),
('Foundation_Stone', 0.002),
('BsmtQual_TA', 0.002),
('BsmtCond_TA', 0.002),
('BsmtFinType1_BLQ', 0.002),
('BsmtFinType2_BLQ', 0.002),
('Functional_Sev', 0.002),
('SaleType_ConLI', 0.002),
('RoofMatl_WdShake', 0.001),
('Exterior1st_Stone', 0.001),
('Exterior1st_Wd Sdng', 0.001),
('Exterior2nd_ImStucc', 0.001),
('SaleType_Con', 0.001),
('MSZoning_RL', -0.0),
('RoofStyle_Gambrel', 0.0),
('Exterior2nd_CBlock', -0.0),
('Exterior2nd_Wd Sdng', -0.0),
('ExterQual_TA', 0.0),
('HeatingQC_Po', 0.0),
('MSZoning_RH', -0.001),
('Utilities_NoSeWa', -0.001),
('Neighborhood_SawyerW', -0.001),
('Condition1_Norm', -0.001),
('Heating_Grav', -0.001),
('HeatingQC_Fa', -0.001),
('WoodDeckSF', -0.002),
('RoofMatl_WdShngl', -0.002),
('Exterior1st_Plywood', -0.002),
```

```
('ExterCond_TA', -0.002),
('BsmtFinType2_Rec', -0.002),
('2ndFlrSF', -0.003),
('LotConfig_FR2', -0.003),
('Neighborhood_Veenker', -0.003),
('BldgType_2fmCon', -0.003),
('HouseStyle_SFoyer', -0.003),
('Exterior2nd_AsphShn', -0.003),
('Exterior2nd_HdBoard', -0.003),
('Exterior2nd_Plywood', -0.003),
('BsmtCond_Gd', -0.003),
('Street_Pave', -0.004),
('PavedDrive_Y', -0.004),
('LotShape_IR3', -0.005),
('Exterior1st_Stucco', -0.005),
('BsmtCond_none', -0.005),
('BsmtExposure_No', -0.005),
('BsmtFinType1_Unf', -0.005),
('PavedDrive_P', -0.005),
('Exterior2nd_CmentBd', -0.006),
('BsmtFinType1_Rec', -0.006),
('Neighborhood_SWISU', -0.007),
('RoofMatl_Roll', -0.007),
('Exterior2nd_BrkFace', -0.007),
('MasVnrType_BrkFace', -0.007),
('BsmtExposure_none', -0.007),
('Heating_GasA', -0.007),
('LandSlope_Mod', -0.008),
('Condition2_PosN', -0.008),
('HouseStyle_2.5Unf', -0.008),
('KitchenQual_Gd', -0.008),
('KitchenQual_TA', -0.008),
('LandContour_Low', -0.009),
('LandContour_Lvl', -0.009),
('Condition2_RRAn', -0.009),
('BldgType_TwnhsE', -0.009),
('Exterior1st_VinylSd', -0.009),
('BsmtFinType1_GLQ', -0.009),
('BsmtExposure_Mn', -0.01),
('Neighborhood_IDOTRR', -0.011),
('Neighborhood_NPkVill', -0.011),
('Condition2_RRAe', -0.011),
('ExterQual_Fa', -0.011),
('ExterCond_Gd', -0.011),
('BsmtHalfBath', -0.012),
('Condition1_RRNn', -0.012),
('MasVnrType_None', -0.012),
('Foundation_CBlock', -0.012),
('LotConfig_CulDSac', -0.013),
('SaleType_CWD', -0.013),
('SaleType_ConLD', -0.013),
('LandContour_HLS', -0.014),
('Neighborhood_NoRidge', -0.014),
('BsmtCond_Po', -0.014),
('BsmtFinType1_none', -0.014),
('LotConfig_Inside', -0.015),
('Heating_GasW', -0.015),
('LotConfig_FR3', -0.018),
('Neighborhood_BrkSide', -0.018),
('Electrical_Mix', -0.018),
('LandSlope_Sev', -0.019),
('Neighborhood_Crawfor', -0.019),
('Neighborhood_NWAmes', -0.02),
('Exterior2nd_Wd Shng', -0.02),
```

```
('Heating_OthW', -0.02),
('MasVnrType_Stone', -0.022),
('Neighborhood_ClearCr', -0.023),
('CentralAir_Y', -0.023),
('MSSubClass', -0.024),
('Condition1_RRNe', -0.024),
('EnclosedPorch', -0.031),
('Neighborhood_Edwards', -0.031),
('MasVnrType_none', -0.031),
('Neighborhood_CollgCr', -0.034),
('Neighborhood_NAmes', -0.034),
('ExterQual_Gd', -0.034),
('Foundation_PConc', -0.034),
('HouseStyle_2Story', -0.043),
('Electrical_FuseF', -0.044),
('Neighborhood_BrDale', -0.045),
('Foundation_Slab', -0.046),
('HouseStyle_2.5Fin', -0.054),
('Electrical_FuseP', -0.072)]
```

In [ ]:
```
lm = Ridge(alpha=2)
lm.fit(X_train, y_train)

y_test_pred = lm.predict(X_test)
print("R2 Score {}".format(r2_score(y_true=y_test, y_pred=y_test_pred)))
print('MSE :{}'.format(mean_squared_error(y_test, y_test_pred)))
print('RMSE :', numpy.sqrt(mean_squared_error(y_test, y_test_pred)))
df_ridge=pandas.DataFrame(data=[[r2_score(y_true=y_test, y_pred=y_test_pred)
```

```
R2 Score 0.756894901165279
MSE :0.04536608282458142
RMSE : 0.21299315206029845
```

In [ ]:
```
mod_ridge = list(zip(cols, model_parameters))
paraRFE = pandas.DataFrame(mod_ridge)
paraRFE.columns = ['Variable', 'Coeff']
res=paraRFE.sort_values(by=['Coeff'], ascending = False)
paraRFE = paraRFE.sort_values((['Coeff']), axis = 0, ascending = False)
paraRFE
```

Out[ ]:

	Variable	Coeff
0	constant	12.031
97	RoofStyle_Gable	0.289
101	RoofStyle_Shed	0.199
103	RoofMatl_Membran	0.141
26	PoolArea	0.125
...	...	...
188	Electrical_FuseF	-0.044
48	Neighborhood_BrDale	-0.045
151	Foundation_Slab	-0.046
92	HouseStyle_2.5Fin	-0.054
189	Electrical_FuseP	-0.072

211 rows × 2 columns

In [ ]:
```
fin_df=pandas.concat([df_lr,df_lasso,df_ridge])
fin_df['R2 Score']=fin_df['R2 Score'].apply(lambda x:Decimal(x))
fin_df['MSE']=fin_df['MSE'].apply(lambda x:Decimal(x))
fin_df['RMSE']=fin_df['RMSE'].apply(lambda x:Decimal(x))
fin_df.insert(0,'Model',['Linear Regression','Lasso','Ridge'])
fin_df
```

Out[ ]:

	Model	R2 Score	
0	Linear Regression	-2407818217657229377536	
0	Lasso	0.84378576170915786658355273175402544438838958...	0.029151293439136548
0	Ridge	0.75689490116527902152654405654175207018852233...	0.045366082824581419

Conclusion

It can be seen from the results above that Lasso Regression has the best Score among all the values for the different models that we have considered