House-Price-Prediction

In [125]:
```
import numpy as np#Import all libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [126]:
```
train=pd.read_csv("train.csv")#read files
test=pd.read_csv("test.csv")
```

In [127]:
```
train.shape#chking rows and columns
```

Out[127]:
```
(1460, 81)
```

In [4]:
```
test.shape
```

Out[4]:
```
(1459, 80)
```

In [128]:
```
train.head(10)#first 5 records
```

Out[128]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utiliti
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllP
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllP
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllP
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllP
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllP
5	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllP
6	7	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllP
7	8	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllP
8	9	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllP
9	10	190	RL	50.0	7420	Pave	NaN	Reg	Lvl	AllP

10 rows × 81 columns

In [130]:
```
train.info()#columns data types
```

```
<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 [131]:
```
test.info()#columns data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1459 non-null   int64  
 1   MSSubClass     1459 non-null   int64  
 2   MSZoning       1455 non-null   object 
 3   LotFrontage    1232 non-null   float64
 4   LotArea        1459 non-null   int64  
 5   Street         1459 non-null   object 
 6   Alley          107 non-null    object 
 7   LotShape       1459 non-null   object 
 8   LandContour    1459 non-null   object 
 9   Utilities      1457 non-null   object 
 10  LotConfig      1459 non-null   object 
 11  LandSlope      1459 non-null   object 
 12  Neighborhood   1459 non-null   object 
 13  Condition1     1459 non-null   object 
 14  Condition2     1459 non-null   object 
 15  BldgType       1459 non-null   object 
 16  HouseStyle     1459 non-null   object 
 17  OverallQual    1459 non-null   int64  
 18  OverallCond    1459 non-null   int64  
 19  YearBuilt      1459 non-null   int64  
 20  YearRemodAdd   1459 non-null   int64  
 21  RoofStyle      1459 non-null   object 
 22  RoofMatl       1459 non-null   object 
 23  Exterior1st    1458 non-null   object 
 24  Exterior2nd    1458 non-null   object 
 25  MasVnrType     1443 non-null   object 
 26  MasVnrArea     1444 non-null   float64
 27  ExterQual      1459 non-null   object 
 28  ExterCond      1459 non-null   object 
 29  Foundation     1459 non-null   object 
 30  BsmtQual       1415 non-null   object 
 31  BsmtCond       1414 non-null   object 
 32  BsmtExposure   1415 non-null   object 
 33  BsmtFinType1   1417 non-null   object 
 34  BsmtFinSF1     1458 non-null   float64
 35  BsmtFinType2   1417 non-null   object 
 36  BsmtFinSF2     1458 non-null   float64
 37  BsmtUnfSF      1458 non-null   float64
 38  TotalBsmtSF    1458 non-null   float64
 39  Heating        1459 non-null   object 
 40  HeatingQC      1459 non-null   object 
 41  CentralAir     1459 non-null   object 
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1459 non-null   int64  
 44  2ndFlrSF       1459 non-null   int64  
 45  LowQualFinSF   1459 non-null   int64  
 46  GrLivArea      1459 non-null   int64  
 47  BsmtFullBath   1457 non-null   float64
 48  BsmtHalfBath   1457 non-null   float64
```

```
 49  FullBath       1459 non-null   int64
 50  HalfBath       1459 non-null   int64
 51  BedroomAbvGr   1459 non-null   int64
 52  KitchenAbvGr   1459 non-null   int64
 53  KitchenQual    1458 non-null   object
 54  TotRmsAbvGrd   1459 non-null   int64
 55  Functional     1457 non-null   object
 56  Fireplaces     1459 non-null   int64
 57  FireplaceQu    729 non-null    object
 58  GarageType     1383 non-null   object
 59  GarageYrBlt    1381 non-null   float64
 60  GarageFinish   1381 non-null   object
 61  GarageCars     1458 non-null   float64
 62  GarageArea     1458 non-null   float64
 63  GarageQual     1381 non-null   object
 64  GarageCond     1381 non-null   object
 65  PavedDrive     1459 non-null   object
 66  WoodDeckSF     1459 non-null   int64
 67  OpenPorchSF    1459 non-null   int64
 68  EnclosedPorch  1459 non-null   int64
 69  3SsnPorch      1459 non-null   int64
 70  ScreenPorch    1459 non-null   int64
 71  PoolArea       1459 non-null   int64
 72  PoolQC         3 non-null      object
 73  Fence          290 non-null    object
 74  MiscFeature    51 non-null     object
 75  MiscVal        1459 non-null   int64
 76  MoSold         1459 non-null   int64
 77  YrSold         1459 non-null   int64
 78  SaleType       1458 non-null   object
 79  SaleCondition  1459 non-null   object
dtypes: float64(11), int64(26), object(43)
memory usage: 912.0+ KB
```

checking null values

In [8]:

```
train.isnull().sum()#chking null values
```

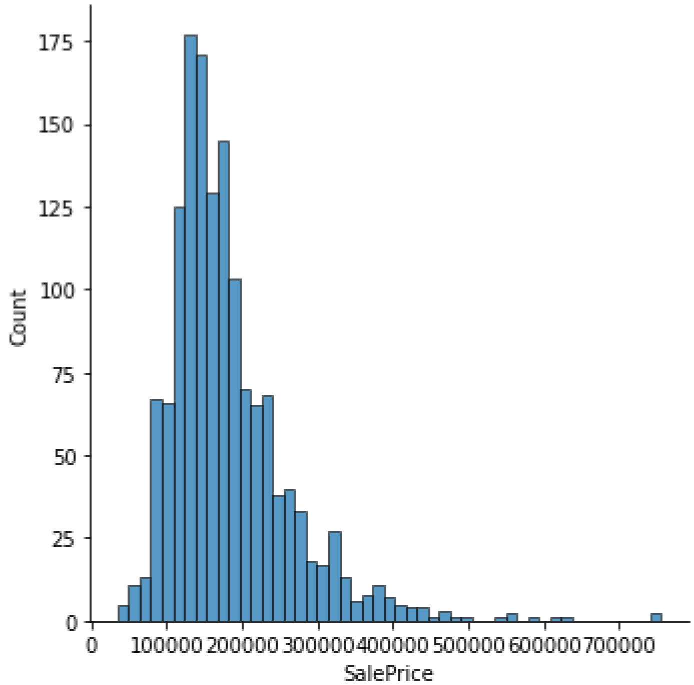
Out[8]:

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

In [132]:
```
plt.figure(figsize=(7,10))
sns.displot(train['SalePrice'])#Distribution of plot
```

Out[132]: <seaborn.axisgrid.FacetGrid at 0x1dd8faa8bb0>

<Figure size 504x720 with 0 Axes>



In [133]:
```
train.describe()#statistical  information
```

Out[133]:

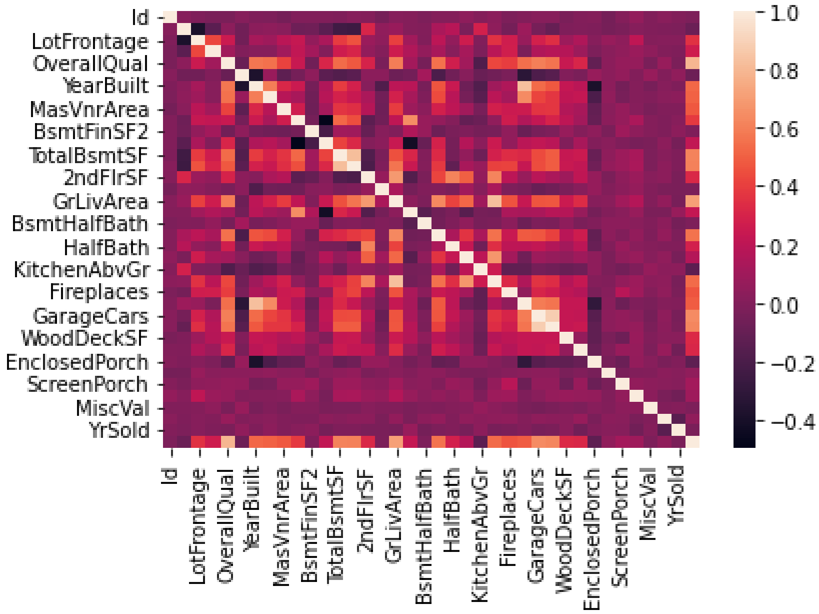
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBı
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.0000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.2678
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.2029
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.0000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.0000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.0000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.0000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.0000

8 rows × 38 columns

In [11]:
```
corr=train.corr()
```

In [12]:
```
sns.heatmap(corr)
```

Out[12]: <AxesSubplot:>



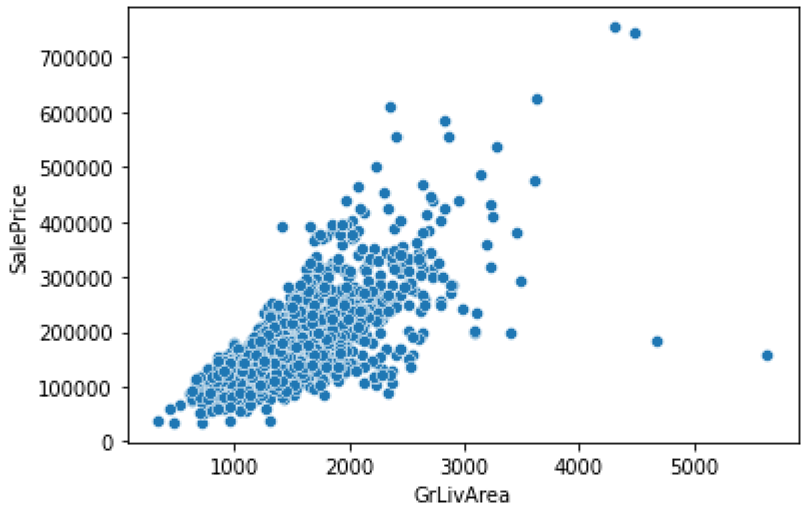
In [134]:
```
corr.sort_values(['SalePrice'],ascending=False,inplace=True)
print(corr.SalePrice)#corelation of saleprice
```

```
SalePrice        1.000000
OverallQual      0.790982
GrLivArea        0.708624
GarageCars       0.640409
GarageArea       0.623431
TotalBsmtSF      0.613581
1stFlrSF         0.605852
FullBath         0.560664
TotRmsAbvGrd     0.533723
YearBuilt        0.522897
YearRemodAdd     0.507101
GarageYrBlt      0.486362
MasVnrArea       0.477493
Fireplaces       0.466929
BsmtFinSF1       0.386420
LotFrontage      0.351799
WoodDeckSF       0.324413
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
MoSold           0.046432
3SsnPorch        0.044584
BsmtFinSF2      -0.011378
BsmtHalfBath    -0.016844
MiscVal         -0.021190
Id              -0.021917
LowQualFinSF    -0.025606
YrSold          -0.028923
OverallCond     -0.077856
MSSubClass      -0.084284
EnclosedPorch   -0.128578
KitchenAbvGr    -0.135907
Name: SalePrice, dtype: float64
```

Visualization on scatter plot

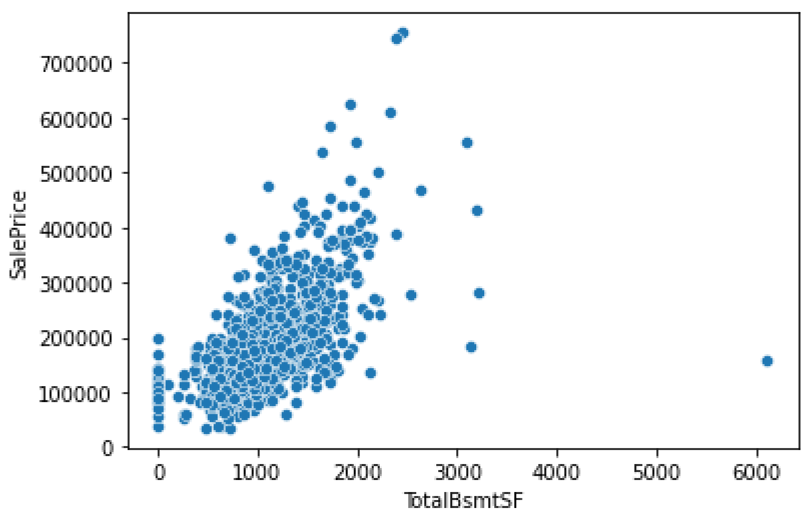
In [14]:
```
sns.scatterplot(x='GrLivArea',y='SalePrice',data=train)
```

Out[14]: <AxesSubplot:xlabel='GrLivArea', ylabel='SalePrice'>



In [15]:
```
sns.scatterplot(x='TotalBsmtSF',y='SalePrice',data=train)
```

Out[15]: <AxesSubplot:xlabel='TotalBsmtSF', ylabel='SalePrice'>



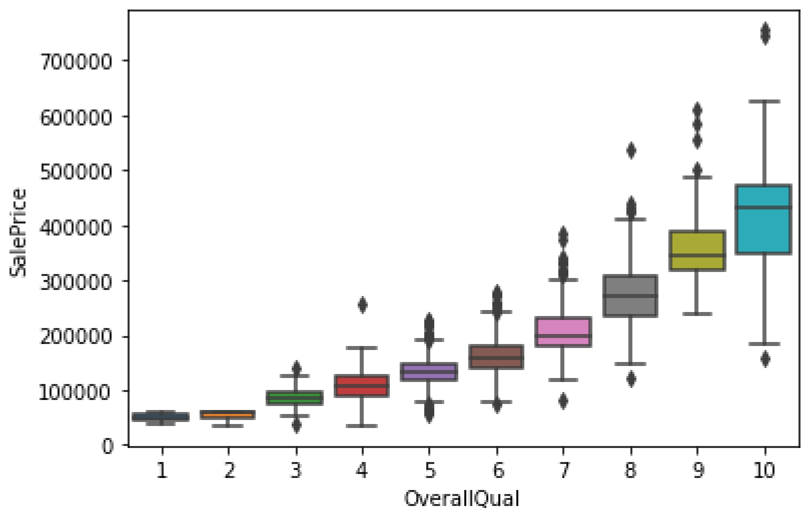
Relationship with categorical features

In [16]:
```
num_col=train.select_dtypes(include=['int','float']).columns
num_col
```

Out[16]:
```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

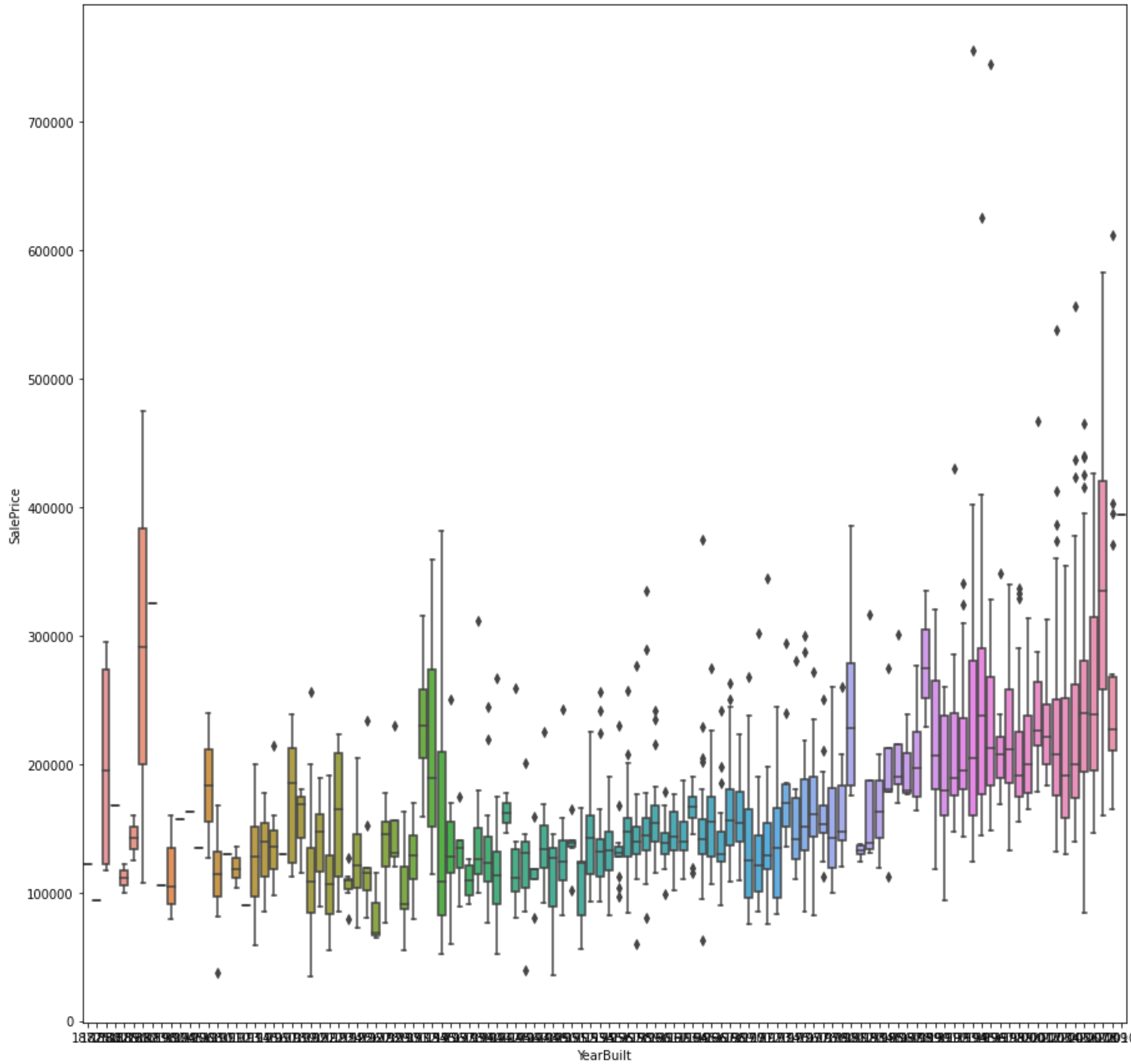
In [135]:
```
sns.boxplot(x='OverallQual',y='SalePrice',data=train)#boxplot easily findout the
```

Out[135]:
```
<AxesSubplot:xlabel='OverallQual', ylabel='SalePrice'>
```



In [18]:
```
plt.figure(figsize=(15,15))
sns.boxplot(x='YearBuilt',y='SalePrice',data=train)
```

Out[18]: <AxesSubplot:xlabel='YearBuilt', ylabel='SalePrice'>



In [136]:
```
num_train=train.select_dtypes(include=['int','float'])#splitting the train and te
num_test=test.select_dtypes(include=['int','float'])
cat_train=train.select_dtypes(include=['object'])
cat_test=test.select_dtypes(include=['object'])
```

In [20]:
```
num_train
```

Out[20]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAd
0	1	60	65.0	8450	7	5	2003	200
1	2	20	80.0	9600	6	8	1976	197
2	3	60	68.0	11250	7	5	2001	200
3	4	70	60.0	9550	7	5	1915	197
4	5	60	84.0	14260	8	5	2000	200
...	...	...	...	...	...	...	...	.
1455	1456	60	62.0	7917	6	5	1999	200
1456	1457	20	85.0	13175	6	6	1978	198
1457	1458	70	66.0	9042	7	9	1941	200
1458	1459	20	68.0	9717	5	6	1950	199
1459	1460	20	75.0	9937	5	6	1965	196

1460 rows × 38 columns

In [137]:
```
num_train.isnull().sum()/len(num_train)*100#chking all null values
```

Out[137]:
```
Id               0.000000
MSSubClass       0.000000
LotFrontage     17.739726
LotArea          0.000000
OverallQual      0.000000
OverallCond      0.000000
YearBuilt        0.000000
YearRemodAdd     0.000000
MasVnrArea       0.547945
BsmtFinSF1       0.000000
BsmtFinSF2       0.000000
BsmtUnfSF        0.000000
TotalBsmtSF      0.000000
1stFlrSF         0.000000
2ndFlrSF         0.000000
LowQualFinSF     0.000000
GrLivArea        0.000000
BsmtFullBath     0.000000
BsmtHalfBath     0.000000
FullBath         0.000000
HalfBath         0.000000
BedroomAbvGr     0.000000
KitchenAbvGr     0.000000
TotRmsAbvGrd     0.000000
Fireplaces       0.000000
GarageYrBlt      5.547945
GarageCars       0.000000
GarageArea       0.000000
WoodDeckSF       0.000000
OpenPorchSF      0.000000
EnclosedPorch    0.000000
3SsnPorch        0.000000
ScreenPorch      0.000000
PoolArea         0.000000
MiscVal          0.000000
MoSold           0.000000
YrSold           0.000000
SalePrice        0.000000
dtype: float64
```

In [138]:
```
cat_train.isnull().sum()/len(cat_train)*100#chking all null values
```

Out[138]:
```
MSZoning          0.000000
Street            0.000000
Alley            93.767123
LotShape          0.000000
LandContour       0.000000
Utilities         0.000000
LotConfig         0.000000
LandSlope         0.000000
Neighborhood      0.000000
Condition1        0.000000
Condition2        0.000000
BldgType          0.000000
HouseStyle        0.000000
RoofStyle         0.000000
RoofMatl          0.000000
Exterior1st       0.000000
Exterior2nd       0.000000
MasVnrType        0.547945
ExterQual         0.000000
ExterCond         0.000000
Foundation        0.000000
BsmtQual          2.534247
BsmtCond          2.534247
BsmtExposure      2.602740
BsmtFinType1      2.534247
BsmtFinType2      2.602740
Heating           0.000000
HeatingQC         0.000000
CentralAir        0.000000
Electrical        0.068493
KitchenQual       0.000000
Functional        0.000000
FireplaceQu      47.260274
GarageType        5.547945
GarageFinish      5.547945
GarageQual        5.547945
GarageCond        5.547945
PavedDrive        0.000000
PoolQC           99.520548
Fence            80.753425
MiscFeature      96.301370
SaleType          0.000000
SaleCondition     0.000000
dtype: float64
```

In [139]:
```
cat_test.isnull().sum()/len(cat_train)*100#chking all null values
```

Out[139]:
```
MSZoning          0.273973
Street            0.000000
Alley            92.602740
LotShape          0.000000
LandContour       0.000000
Utilities         0.136986
LotConfig         0.000000
LandSlope         0.000000
Neighborhood      0.000000
Condition1        0.000000
Condition2        0.000000
BldgType          0.000000
HouseStyle        0.000000
RoofStyle         0.000000
RoofMatl          0.000000
Exterior1st       0.068493
Exterior2nd       0.068493
MasVnrType        1.095890
ExterQual         0.000000
ExterCond         0.000000
Foundation        0.000000
BsmtQual          3.013699
BsmtCond          3.082192
BsmtExposure      3.013699
BsmtFinType1      2.876712
BsmtFinType2      2.876712
Heating           0.000000
HeatingQC         0.000000
CentralAir        0.000000
Electrical        0.000000
KitchenQual       0.068493
Functional        0.136986
FireplaceQu      50.000000
GarageType        5.205479
GarageFinish      5.342466
GarageQual        5.342466
GarageCond        5.342466
PavedDrive        0.000000
PoolQC           99.726027
Fence            80.068493
MiscFeature      96.438356
SaleType          0.068493
SaleCondition     0.000000
dtype: float64
```

In [140]:
```
num_test.isnull().sum()/len(cat_train)*100#chking all null values
```

Out[140]:
```
Id                0.000000
MSSubClass        0.000000
LotFrontage      15.547945
LotArea           0.000000
OverallQual       0.000000
OverallCond       0.000000
YearBuilt         0.000000
YearRemodAdd      0.000000
MasVnrArea        1.027397
BsmtFinSF1        0.068493
BsmtFinSF2        0.068493
BsmtUnfSF         0.068493
TotalBsmtSF       0.068493
1stFlrSF          0.000000
2ndFlrSF          0.000000
LowQualFinSF      0.000000
GrLivArea         0.000000
BsmtFullBath      0.136986
BsmtHalfBath      0.136986
FullBath          0.000000
HalfBath          0.000000
BedroomAbvGr      0.000000
KitchenAbvGr      0.000000
TotRmsAbvGrd      0.000000
Fireplaces        0.000000
GarageYrBlt       5.342466
GarageCars        0.068493
GarageArea        0.068493
WoodDeckSF        0.000000
OpenPorchSF       0.000000
EnclosedPorch     0.000000
3SsnPorch         0.000000
ScreenPorch       0.000000
PoolArea          0.000000
MiscVal           0.000000
MoSold            0.000000
YrSold            0.000000
dtype: float64
```

In [26]:
```
missing_value_0=['BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','BsmtFullBat

for i in missing_value_0:
    num_train[i]=num_train[i].fillna(0)

missing_value_none=['Alley','PoolQC','MiscFeature','Fence','FireplaceQu','GarageT


for i in missing_value_none:
    cat_train[i]=cat_train[i].fillna('None')
```

In [27]:
```
missing_value_0 = ['BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','BsmtFullB

for i in missing_value_0:
    num_test[i] = num_test[i].fillna(0)

missing_value_none = ['Alley','PoolQC','MiscFeature','Fence','FireplaceQu','Garag

for i in missing_value_none:
    cat_test[i] = cat_test[i].fillna('None')
```

Filling the all columns

In [28]:
```
cat_train['Electrical']=cat_train['Electrical'].fillna(cat_train['Electrical'].mo
```

In [29]:
```
cat_test['MSZoning'] = cat_test['MSZoning'].fillna(cat_test['MSZoning'].mode()[0]
cat_test['Utilities'] = cat_test['Utilities'].fillna(cat_test['Utilities'].mode()
cat_test['Functional'] = cat_test['Functional'].fillna(cat_test['Functional'].mod
cat_test['KitchenQual'] = cat_test['KitchenQual'].fillna(cat_test['KitchenQual'].
cat_test['Exterior2nd'] = cat_test['Exterior2nd'].fillna(cat_test['Exterior2nd'].
cat_test['Exterior1st'] = cat_test['Exterior1st'].fillna(cat_test['Exterior1st'].
cat_test['SaleType'] = cat_test['SaleType'].fillna(cat_test['SaleType'].mode()[0]
```

filling numerical na values using median

In [30]:
```
num_train=num_train.fillna(num_train.median())
num_test=num_test.fillna(num_test.median())
```

chking for na values

In [31]:
```
num_train.isnull().sum()
```

Out[31]:
```
Id               0
MSSubClass       0
LotFrontage      0
LotArea          0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
MasVnrArea       0
BsmtFinSF1       0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
TotRmsAbvGrd     0
Fireplaces       0
GarageYrBlt      0
GarageCars       0
GarageArea       0
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
MiscVal          0
MoSold           0
YrSold           0
SalePrice        0
dtype: int64
```

In [32]:
```
cat_train.isnull().sum()
```

Out[32]:
```
MSZoning         0
Street           0
Alley            0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
ExterQual        0
ExterCond        0
Foundation       0
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinType2     0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
KitchenQual      0
Functional       0
FireplaceQu      0
GarageType       0
GarageFinish     0
GarageQual       0
GarageCond       0
PavedDrive       0
PoolQC           0
Fence            0
MiscFeature      0
SaleType         0
SaleCondition    0
dtype: int64
```

In [33]:
```
num_test.isnull().sum()
```

Out[33]:
```
Id               0
MSSubClass       0
LotFrontage      0
LotArea          0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
MasVnrArea       0
BsmtFinSF1       0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
TotRmsAbvGrd     0
Fireplaces       0
GarageYrBlt      0
GarageCars       0
GarageArea       0
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
MiscVal          0
MoSold           0
YrSold           0
dtype: int64
```

In [34]:
```
cat_test.isnull().sum()
```

Out[34]:
```
MSZoning         0
Street           0
Alley            0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
ExterQual        0
ExterCond        0
Foundation       0
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinType2     0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
KitchenQual      0
Functional       0
FireplaceQu      0
GarageType       0
GarageFinish     0
GarageQual       0
GarageCond       0
PavedDrive       0
PoolQC           0
Fence            0
MiscFeature      0
SaleType         0
SaleCondition    0
dtype: int64
```

LabelEncoder:we cant handle categorical data to convert cat into number format use Label_encoder

In [35]:
```
from sklearn.preprocessing import LabelEncoder
```

In [141]:
```
l=LabelEncoder()#initialize the object
```

In [37]:
```
cat_col_train=cat_train.select_dtypes(include=['object']).columns
cat_col_test=cat_test.select_dtypes(include=['object']).columns
```

In [39]:
```
for cat in cat_col_train:
    le=LabelEncoder()
    cat_train[cat]=le.fit_transform(cat_train[cat])
```

In [40]:
```
cat_train.head()
```

Out[40]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood
0	3	1	1	3	3	0	4	0	5
1	3	1	1	3	3	0	2	0	24
2	3	1	1	0	3	0	4	0	5
3	3	1	1	0	3	0	0	0	6
4	3	1	1	0	3	0	2	0	15

5 rows × 43 columns

In [41]:
```
for cat in cat_col_test:
    le=LabelEncoder()
    cat_test[cat]=le.fit_transform(cat_test[cat])
```

In [42]:
```
cat_test.head()
```

Out[42]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood
0	2	1	1	3	3	0	4	0	12
1	3	1	1	0	3	0	0	0	12
2	3	1	1	0	3	0	4	0	8
3	3	1	1	0	3	0	4	0	8
4	3	1	1	0	1	0	4	0	22

5 rows × 43 columns

Merging the numerical and categorical features

In [43]:
```
train=pd.concat([num_train,num_test],axis=1)
test=pd.concat([cat_train,cat_test],axis=1)
```

In [46]:
```
train.head()
```

Out[46]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Ma
0	1	60	65.0	8450	7	5	2003	2003	
1	2	20	80.0	9600	6	8	1976	1976	
2	3	60	68.0	11250	7	5	2001	2002	
3	4	70	60.0	9550	7	5	1915	1970	
4	5	60	84.0	14260	8	5	2000	2000	

5 rows × 75 columns

In [47]:
```
test.head()
```

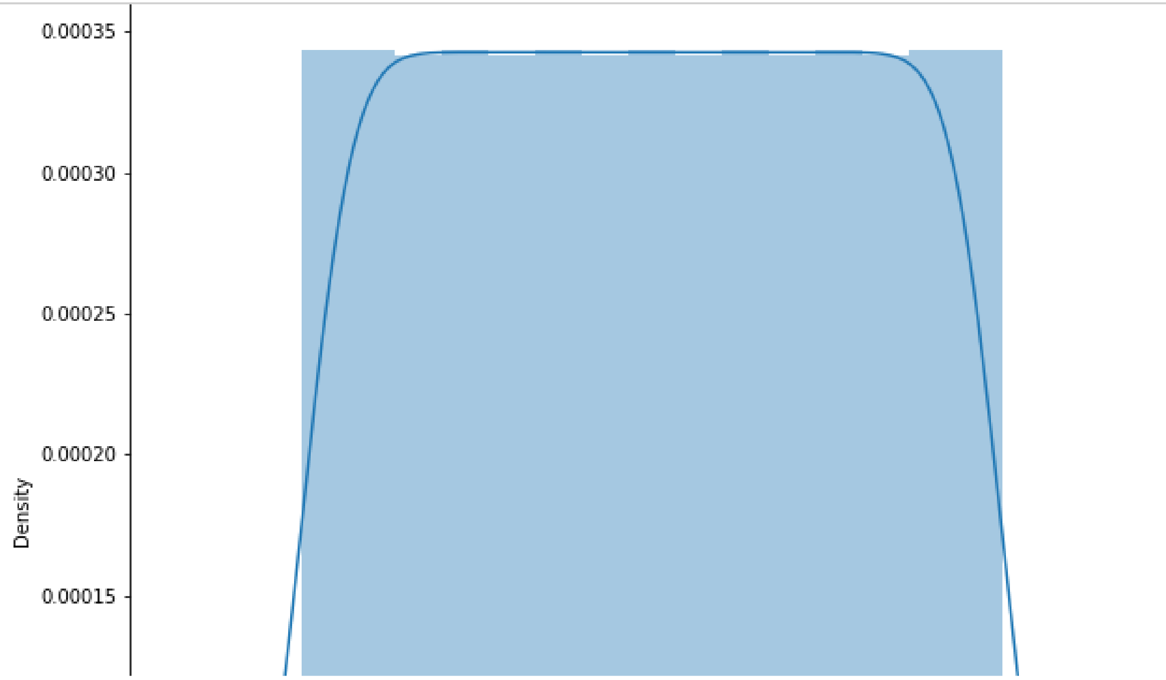
Out[47]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood
0	3	1	1	3	3	0	4	0	5
1	3	1	1	3	3	0	2	0	24
2	3	1	1	0	3	0	4	0	5
3	3	1	1	0	3	0	0	0	6
4	3	1	1	0	3	0	2	0	15

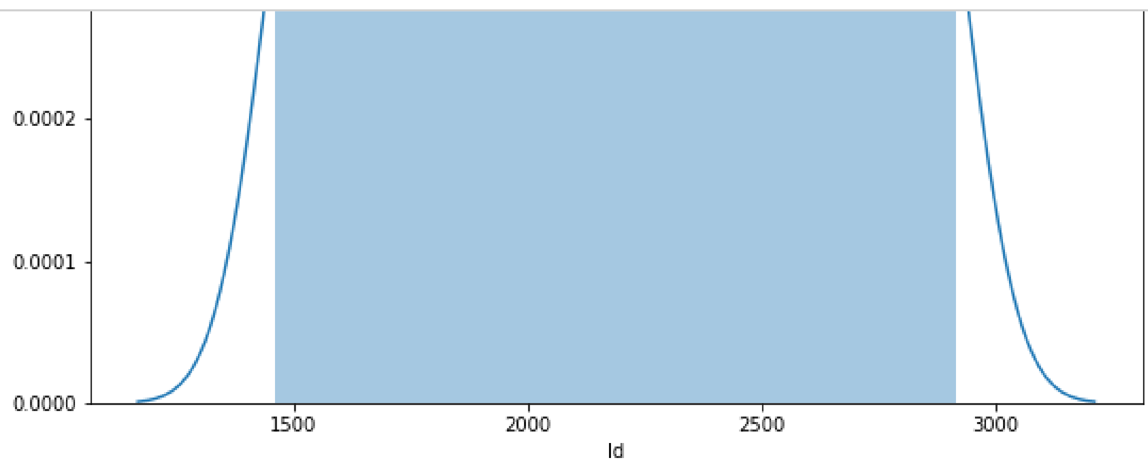
5 rows × 86 columns

Visualization of numerical data

In [50]:
```
for col in num_train:
    print(col)
    print('skewness',train[col].skew())
    print('kurtosis',train[col].kurt())
    plt.figure(figsize=(10,10))
    sns.distplot(train[col])
    plt.show()
```



In [55]:
```
for test in num_test:
    print(test)
    print('skewness',num_test[col].skew())
    print('kurtosis',num_test[col].kurt())
    plt.figure(figsize=(10,10))
    sns.distplot(num_test[col])
    plt.show()
```




```
2ndFlrSF
skewness 0.0
kurtosis -1.2000000000000002
```

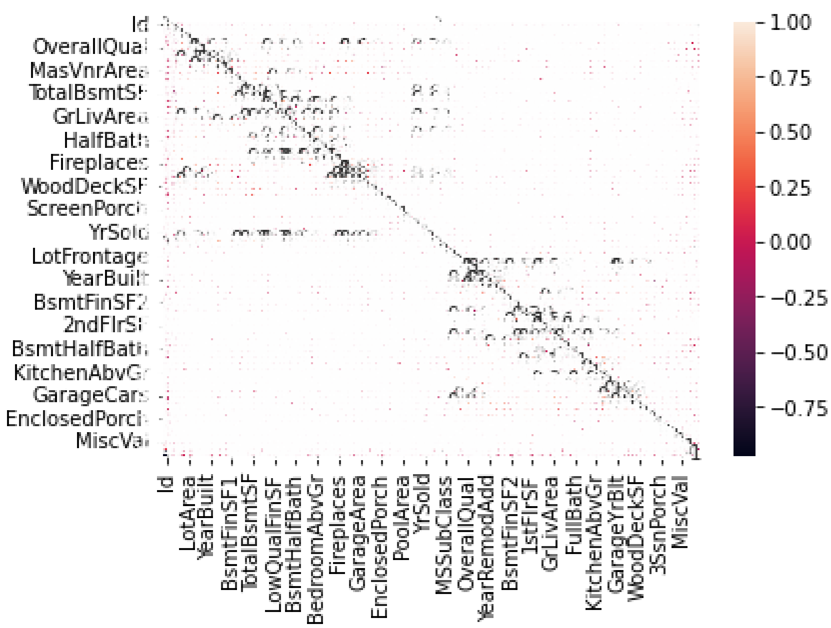
In [56]:
```
corr_train=train.corr()
```

In [60]:
```
num_train.columns
```

Out[60]:
```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [66]:
```
sns.heatmap(corr_train,annot=True)
```

Out[66]: <AxesSubplot:>



In [142]:
```
from sklearn.preprocessing import MinMaxScaler#scaling on traing features
```

In [68]:
```
min_max=MinMaxScaler()
```

In [69]:
```
min_max.fit_transform(num_train)
```

Out[69]:
```
array([[0.00000000e+00, 2.35294118e-01, 1.50684932e-01, ...,
        9.09090909e-02, 5.00000000e-01, 2.41077628e-01],
       [6.85400960e-04, 0.00000000e+00, 2.02054795e-01, ...,
        3.63636364e-01, 2.50000000e-01, 2.03582836e-01],
       [1.37080192e-03, 2.35294118e-01, 1.60958904e-01, ...,
        7.27272727e-01, 5.00000000e-01, 2.61908068e-01],
       ...,
       [9.98629198e-01, 2.94117647e-01, 1.54109589e-01, ...,
        3.63636364e-01, 1.00000000e+00, 3.21621997e-01],
       [9.99314599e-01, 0.00000000e+00, 1.60958904e-01, ...,
        2.72727273e-01, 1.00000000e+00, 1.48902930e-01],
       [1.00000000e+00, 0.00000000e+00, 1.84931507e-01, ...,
        4.54545455e-01, 5.00000000e-01, 1.56367171e-01]])
```

In [70]:
```
min_max.fit_transform(cat_train)
```

Out[70]:
```
array([[0.75, 1.  , 0.5 , ..., 0.25, 1.  , 0.8 ],
       [0.75, 1.  , 0.5 , ..., 0.25, 1.  , 0.8 ],
       [0.75, 1.  , 0.5 , ..., 0.25, 1.  , 0.8 ],
       ...,
       [0.75, 1.  , 0.5 , ..., 0.75, 1.  , 0.8 ],
       [0.75, 1.  , 0.5 , ..., 0.25, 1.  , 0.8 ],
       [0.75, 1.  , 0.5 , ..., 0.25, 1.  , 0.8 ]])
```

In [72]:
```
from sklearn.preprocessing import StandardScaler
```

In [73]:
```
ss=StandardScaler()
```

In [143]:
```
ss.fit_transform(num_train)#scaling on traing features
```

Out[143]:
```
array([[-1.73086488,  0.07337496, -0.20803433, ..., -1.5991111 ,
         0.13877749,  0.34727322],
       [-1.7284922 , -0.87256276,  0.40989452, ..., -0.48911005,
        -0.61443862,  0.00728832],
       [-1.72611953,  0.07337496, -0.08444856, ...,  0.99089135,
         0.13877749,  0.53615372],
       ...,
       [ 1.72611953,  0.30985939, -0.16683907, ..., -0.48911005,
         1.64520971,  1.07761115],
       [ 1.7284922 , -0.87256276, -0.08444856, ..., -0.8591104 ,
         1.64520971, -0.48852299],
       [ 1.73086488, -0.87256276,  0.20391824, ..., -0.1191097 ,
         0.13877749, -0.42084081]])
```

In [75]:
```
ss.fit_transform(cat_train)
```

Out[75]:
```
array([[-0.04553194,  0.06423821,  0.02469891, ..., -0.1859753 ,
         0.31386709,  0.2085023 ],
       [-0.04553194,  0.06423821,  0.02469891, ..., -0.1859753 ,
         0.31386709,  0.2085023 ],
       [-0.04553194,  0.06423821,  0.02469891, ..., -0.1859753 ,
         0.31386709,  0.2085023 ],
       ...,
       [-0.04553194,  0.06423821,  0.02469891, ...,  5.19073639,
         0.31386709,  0.2085023 ],
       [-0.04553194,  0.06423821,  0.02469891, ..., -0.1859753 ,
         0.31386709,  0.2085023 ],
       [-0.04553194,  0.06423821,  0.02469891, ..., -0.1859753 ,
         0.31386709,  0.2085023 ]])
```

In [79]:
```
num_train.columns
```

Out[79]:
```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [144]:
```
x=num_train.iloc[:,:-1]#Training data
y=num_train.iloc[:,-1]#Testing data
```

In [81]:
```
x
```

Out[81]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAd
0	1	60	65.0	8450	7	5	2003	200
1	2	20	80.0	9600	6	8	1976	197
2	3	60	68.0	11250	7	5	2001	200
3	4	70	60.0	9550	7	5	1915	197
4	5	60	84.0	14260	8	5	2000	200
...	...	...	...	...	...	...	...	.
1455	1456	60	62.0	7917	6	5	1999	200
1456	1457	20	85.0	13175	6	6	1978	198
1457	1458	70	66.0	9042	7	9	1941	200
1458	1459	20	68.0	9717	5	6	1950	199
1459	1460	20	75.0	9937	5	6	1965	196

1460 rows × 37 columns

In [82]:
```
y
```

Out[82]:
```
0       208500
1       181500
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

Splitting the price_train dataset into training and testing

In [84]:
```
from sklearn.model_selection import train_test_split
```

In [85]:
```
#Spliting dataset into train and test data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=12
```

In [86]:
```
x_train.shape
```

Out[86]:
```
(1168, 37)
```

In [87]:
```
y_train.shape
```

Out[87]:
```
(1168,)
```

In [88]:
```
y_test.shape
```

Out[88]:
```
(292,)
```

In [89]:
```
x_test.shape
```

Out[89]:
```
(292, 37)
```

Model Building

In [90]:
```
from sklearn.linear_model import LinearRegression
```

In [91]:
```
reg=LinearRegression()
```

In [93]:
```
#Fitting the model or taining the model
reg.fit(x_train,y_train)
```

Out[93]:
```
▾ LinearRegression
LinearRegression()
```

In [94]:
```
#make prediction
y_train_pred=reg.predict(x_train)
y_test_pred=reg.predict(x_test)
```

In [95]:
```
y_train_pred
```

Out[95]:
```
array([298899.27778744, 199932.62005621, 295443.63872736, ...,
       261676.72099484, 128806.06422557, 146055.76823665])
```

In [96]:
```
y_test_pred
```

Out[96]:
```
array([231558.68720498, 112138.92145563, 189761.71754738, 255842.1314891 ,
       132805.21596227, 244628.68645375, 285237.53480931, 148962.29897592,
       154401.98205028, 143349.06274937, 156209.54286885, 251194.90982847,
       148587.22577773, 101760.37916682, 264518.4297312 , 194847.36842302,
       170018.7896581 , 305847.24083893, 224762.02458317, 189676.9590357 ,
       174011.89134749, 196911.58515786, 122794.93371585, 188236.51790348,
       214319.11211778, 137490.87116867, 217776.05619937, 188065.77568861,
       119074.1406605 , 151428.4596399 , 118083.6277496 , 194477.25738885,
       124329.89447951, 203068.23001137, 323340.73253712, 209670.89946653,
       182847.1761697 , 354625.38431651, 197219.90144456,  91336.47076524,
       145128.98898195, 227906.22406561, 202410.83737177, 174444.18982961,
       232483.02476715,  86794.73151769, 287367.97529617, 230474.61826009,
       252176.20664518, 108807.88263876, 194653.19774989, 200689.35040379,
       156864.64009548, 184113.66239773, 132170.73554479, 166174.96576276,
       126893.97087064, 176876.84842449, 267655.89527055, 331319.66536423,
       106095.85341368, 312478.55626937, 114364.45939614, 203989.5818394 ,
       221964.19827907, 126791.15123912, 115068.15819585, 239909.51581503,
       226861.253143  , 153615.01217486, 149816.17149295, 216194.04072529,
       159350.62336733, 257898.51571511, 178450.52948571, 107762.68323977,
       192482.83014914, 147246.5748673 , 224223.29833175, 151907.36155374,
       321958.89501954, 292006.66994075,  81380.25672908,  84194.8760382 ,
       188218.22977294, 165219.63553352, 111954.73374847, 110772.27624601,
        50931.39386731, 304431.59439412, 135502.79348298, 176357.71892693,
       185055.18208843, 196504.69238089, 157604.58925443, 137272.49000865,
       219770.98538954,  96815.35389077, 239502.57518728, 119645.59025854,
       105235.14025943, 348238.03433991, 314248.58057368, 160808.78375448,
       194843.70143411, 110793.43633035, 120227.51735661, 207162.74043914,
       149422.45554196, 286350.9625856 , 267794.27584989, 211239.21172091,
        72367.20897055, 265442.77158351, 182692.33224503, 121164.87190637,
       195407.25358462, 370717.98532083, 115770.42954201, 149521.396354  ,
       178455.56039702, 219327.11959515, 217492.09011028, 134871.94546043,
       258006.51675706, 263648.34792937, 123597.10190718, 122502.34261938,
       142236.00640625, 295882.12456854,  90039.54403057, 314361.32751843,
       290338.47204839, 199570.9896322 , 241118.97217976, 173373.17234651,
       317460.35755545, 109659.45503018, 199492.86868898, 128061.18803231,
       128110.29771518, 268141.20406218, 255349.97340573, 138826.40582973,
       214674.82940927, 291178.46013607, 236606.16082632, 168767.87732144,
       188032.78529234, 122282.48318127, 208511.86807201, 126911.96696767,
       347256.29458414, 180318.53587714, 274496.09563968, 167101.19920505,
       246215.04476742, 121006.67137525, 131757.50187684, 140181.84142844,
       448146.4443726 , 228022.39046585, 244055.11644756, 125399.9579365 ,
       252306.62278021,  89473.55782934, 163142.10468427, 125870.273816  ,
       177486.05415713,  87758.17725701, 120744.78626139, 103121.58280974,
       224989.84038351, 253713.47735387, 120425.88865921, 282945.94849016,
       190830.0806317 ,  96307.23378376, 214542.87123137, 239093.682946  ,
       194158.98118835, 145322.81671231, 147380.94441479, 143547.52877735,
       180816.86716465, 118217.11386826,  98690.89843717, 198139.23862725,
       155369.721957  , 182615.99662592, 110796.98986564, 129430.86046722,
       189311.30753068, 245719.56265983, 122562.07095538, 279880.95008305,
       156203.7781994 , 111697.43072139, 177423.73037125, 121392.32846198,
       207621.48629138, 300191.97060627, 178736.67830877, 309416.40810114,
       176929.19789416, 105831.07058216,  82295.89883277, 281999.46775628,
       156681.94272734, 206663.69771048, 182135.9595253 , 321681.52227999,
       107841.74297863, 167009.78962693, 146648.15706221, 213626.88504949,
```

```
       198231.35839075, 111553.74720088, 117569.55763187,  88683.05469289,
       178391.97121578, 120492.39990548, 141290.75554382, 112095.10861712,
       128380.5251069 , 117719.13609748, 181877.33164647,  89338.97065544,
       131163.08196152, 117535.04968487, 106820.0073277 , 158669.88882659,
       181098.99858504, 106476.63780271, 102285.91242903, 186039.49971339,
       207375.01273059, 135867.0488469 , 171173.66166384, 102468.05261587,
       263940.06609065, 253465.31123442, 199529.77824573, 212626.37907441,
        98441.98859836, 265123.36200078, 145144.11015134, 291915.50266947,
       122377.35400121, 275001.19404766, 240157.3164837 ,  93552.64112955,
       206232.8908422 , 103494.3630505 , 181705.42904197, 199604.28240819,
       271281.40102956, 222318.95525925, 127191.27078337,  83807.14729853,
       221227.38193801, 203862.42529517, 185148.33531447, 196176.18942041,
       183131.75837459, 154125.49046133, 171596.94898422, 202644.40887163,
        63847.27189248,  99617.68787196, 159399.35391885,  46670.42176844,
       214314.74130891,  96947.93996691, 264406.57009201, 160433.31281304,
       210626.84797906, 232402.42929888, 129395.9858778 ,  75125.64944041,
       108932.38260268, 148643.05432739, 113202.9071719 , 122257.16809633,
       166774.06236654, 116893.54655025, 187573.1929301 , 148106.7990398 ,
       196993.77706583, 121179.23827863, 189850.02157414,  97251.04875034])
```

find intercept

In [97]:
```python
reg.intercept_
```

Out[97]:
```
121337.99241742393
```

find coef

In [98]:
```python
reg.coef_
```

Out[98]:
```
array([-1.15851145e+00, -1.97372456e+02, -1.42405512e+02,  3.99000313e-01,
        1.75975129e+04,  4.07843743e+03,  3.24127987e+02,  1.32385366e+02,
        2.33631593e+01,  1.00433633e+01,  1.32265636e+00, -2.53375452e+00,
        8.83226517e+00,  2.40268102e+01,  2.47896093e+01, -2.52028340e+01,
        2.36135855e+01,  7.17858081e+03,  3.35045009e+03,  1.96638893e+03,
       -3.50570292e+03, -9.44214897e+03, -1.34208971e+04,  4.51377848e+03,
        5.48890029e+03, -1.76389772e+01,  1.96830705e+04, -1.14139367e+00,
        2.53547591e+01, -4.44043017e+00,  3.82149233e+00,  1.89932440e+01,
        3.82511583e+01, -1.38850551e+01, -2.14929104e-01,  9.08345878e+01,
       -5.19470454e+02])
```

Find R2 Score:

In [112]:
```python
from sklearn.metrics import r2_score,mean_squared_error
```

In [113]:
```python
train=r2_score(y_train,y_train_pred)
```

In [114]:
```python
test=r2_score(y_test,y_test_pred)
```

In [115]:
```
train
```

Out[115]:
```
0.8095254826280767
```

In [116]:
```
test
```

Out[116]:
```
0.8377642816516859
```

How to chk cross Validation:

In [117]:
```
from sklearn.model_selection import KFold,cross_val_score
```

In [118]:
```
cv=KFold(n_splits=5,shuffle=True, random_state=1)
```

In [119]:
```
score=cross_val_score(reg,x,y,scoring='r2',cv=cv,n_jobs=-1)
```

In [120]:
```
score
```

Out[120]:
```
array([0.81899035, 0.61287395, 0.82132133, 0.74982015, 0.85972589])
```

In [111]:
```
np.mean(score)
```

Out[111]:
```
0.7725463337723129
```

In [121]:
```
def metric(y_actual,y_pred):
    r2=r2_score(y_actual,y_pred)
    RMSE=np.sqrt(mean_squared_error(y_actual,y_pred))
    print("r2 score: {} | RMSE: {} ".format(round(r2,2),round(RMSE,2)))
```

In [124]:
```
print("Training performance")
metric(y_train,y_train_pred)
print("Testing performance")
metric(y_test,y_test_pred)
```

```
Training performance
r2 score: 0.81 | RMSE: 34746.2
Testing performance
r2 score: 0.84 | RMSE: 31663.14
```

In [ ]:
```

```