In [117]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [118]:
```
HS = pd.read_csv('Housing dataset.csv')
```

In [119]:
```
HS.head()
```

Out[119]:

	Id	OverallQual	YearBuilt	TotalBsmtSF	Electrical	GrLivArea	FullBath	GarageType	GarageC
0	1	7	2003	856	SBrkr	1710	2	Attchd	
1	2	6	1976	1262	SBrkr	1262	2	Attchd	
2	3	7	2001	920	SBrkr	1786	2	Attchd	
3	4	7	1915	756	SBrkr	1717	1	Detchd	
4	5	8	2000	1145	SBrkr	2198	2	Attchd	

In [120]:
```
HS.shape
```

Out[120]:
```
(1418, 13)
```

Identifying Missing values for accuracy of analysis

In [121]:
```
HS.isnull().sum().sort_values(ascending=False)
```

Out[121]:
```
MiscFeature    1366
Fence          1148
GarageType      143
Electrical        1
SalePrice         0
GarageArea        0
GarageCars        0
FullBath          0
GrLivArea         0
TotalBsmtSF       0
YearBuilt         0
OverallQual       0
Id                0
dtype: int64
```

Percentage missing values

In [122]:
```
(HS.isnull().sum()/ HS.isnull().count() *100).sort_values(ascending=False)
```

Out[122]:
```
MiscFeature    96.332863
Fence          80.959097
GarageType     10.084626
Electrical      0.070522
SalePrice       0.000000
GarageArea      0.000000
GarageCars      0.000000
FullBath        0.000000
GrLivArea       0.000000
TotalBsmtSF     0.000000
YearBuilt       0.000000
OverallQual     0.000000
Id              0.000000
dtype: float64
```

If we have large amount of missing values in data, let say more than 50%, we need to decide to analyse data or we need to manufacture the data, then do the analysis. But in this housing dataset column 'MiscFeature' and 'Fence' has more than 80% missing data. So, I decided to drop the columns. I will fill values in 'GarageType' and 'Electrical' as percentage missing values is less than 50% and it will not harm our analysis.

In [123]:
```
HS.drop(columns=["MiscFeature","Fence"],inplace=True)
```

In [124]:
```
HS.head()
```

Out[124]:

	Id	OverallQual	YearBuilt	TotalBsmtSF	Electrical	GrLivArea	FullBath	GarageType	GarageC
0	1	7	2003	856	SBrkr	1710	2	Attchd	
1	2	6	1976	1262	SBrkr	1262	2	Attchd	
2	3	7	2001	920	SBrkr	1786	2	Attchd	
3	4	7	1915	756	SBrkr	1717	1	Detchd	
4	5	8	2000	1145	SBrkr	2198	2	Attchd	

Both columns has been dropped

Remaining Missing values

```
In [125]: HS.isnull().sum().sort_values(ascending=False)
```

```
Out[125]: GarageType     143
          Electrical       1
          SalePrice        0
          GarageArea       0
          GarageCars       0
          FullBath         0
          GrLivArea        0
          TotalBsmtSF      0
          YearBuilt        0
          OverallQual      0
          Id               0
          dtype: int64
```

Filling up values in Electrical and Garage type columns

```
In [126]: HS['Electrical'].mode()
```

```
Out[126]: 0    SBrkr
          dtype: object
```

```
In [127]: HS['Electrical'].fillna('SBrkr',inplace=True)
```

```
In [128]: HS.isnull().sum()
```

```
Out[128]: Id               0
          OverallQual      0
          YearBuilt        0
          TotalBsmtSF      0
          Electrical       0
          GrLivArea        0
          FullBath         0
          GarageType     143
          GarageCars       0
          GarageArea       0
          SalePrice        0
          dtype: int64
```

Now, it is turn for GarageType. I will use groupby function to fill missing values. I will relate GarageType by GarageCars so that we can fill values with NoGarage if there is zero cars in GarageCars column and Detchd/Attached/Builtin/CarPort/Basement/2Types where there are cars available in GarageCars

In [129]:
```
HS.groupby('GarageType').median()
```

Out[129]:

	Id	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	FullBath	GarageCars	Garag
GarageType								
2Types	767.5	5.0	1959.5	1172.0	1698.0	1.5	3.0	
Attchd	737.5	7.0	1993.0	1176.0	1565.0	2.0	2.0	
Basment	999.0	6.0	1957.0	920.0	1431.0	1.0	2.0	
BuiltIn	613.0	7.0	2003.0	956.0	2035.0	2.0	2.0	
CarPort	535.0	4.0	1962.0	816.0	1296.0	1.0	2.0	
Detchd	699.0	5.0	1946.0	842.0	1214.0	1.0	1.0	

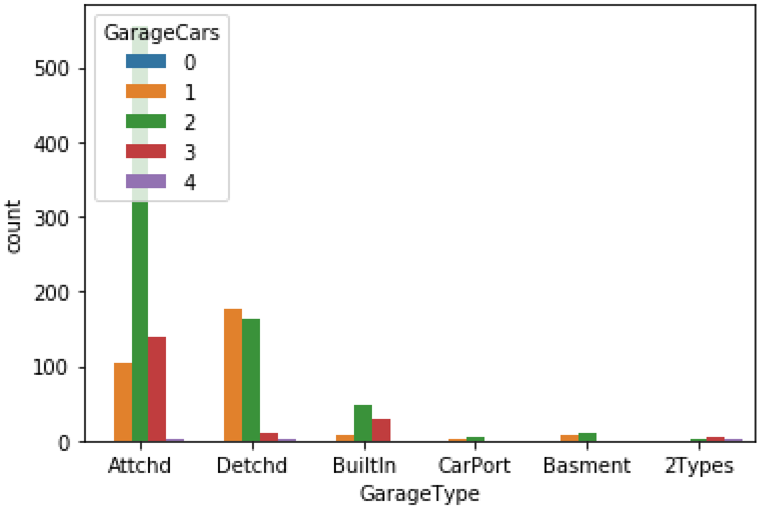
In [130]:
```
HS.groupby('GarageType')['GarageCars'].describe()
```

Out[130]:

	count	mean	std	min	25%	50%	75%	max
GarageType								
2Types	6.0	3.000000	0.632456	2.0	3.0	3.0	3.0	4.0
Attchd	800.0	2.043750	0.554274	1.0	2.0	2.0	2.0	4.0
Basment	19.0	1.578947	0.507257	1.0	1.0	2.0	2.0	2.0
BuiltIn	87.0	2.252874	0.614143	1.0	2.0	2.0	3.0	3.0
CarPort	9.0	1.666667	0.500000	1.0	1.0	2.0	2.0	2.0
Detchd	354.0	1.539548	0.592556	1.0	1.0	1.0	2.0	4.0

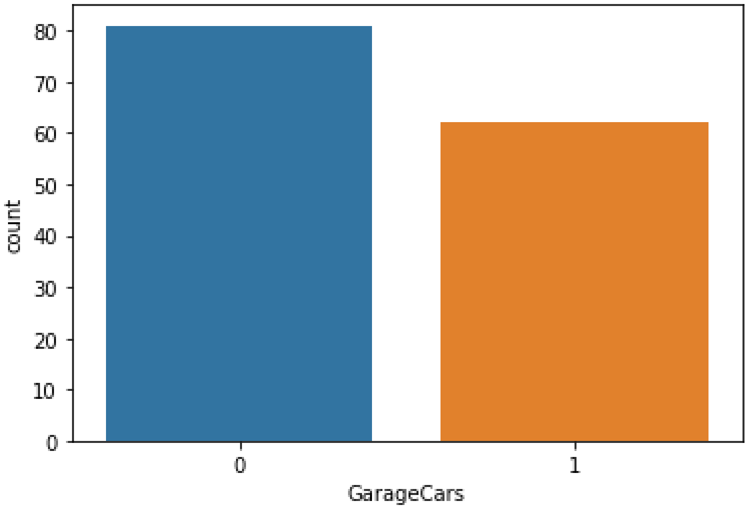
Countplot is made to identify which GarageCars belongs to GarageType

```
In [131]: sns.countplot(x='GarageType', hue='GarageCars', data=HS)

Out[131]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf13cff88>
```



Countplot where GarageType is missing which will tell us th value of GarageCar where GarageType is missing.

```
In [132]: sns.countplot(x='GarageCars', data=HS[HS['GarageType'].isnull()])

Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf1482f88>
```



In [133]:

```
HS[(HS['GarageType'].isnull()) & (HS['GarageCars']==0)]
```

Out[133]:

	Id	OverallQual	YearBuilt	TotalBsmtSF	Electrical	GrLivArea	FullBath	GarageType	Ga
37	38	4	1955	0	FuseP	1152	2	NaN	
46	47	4	1920	736	SBrkr	1452	2	NaN	
73	74	4	1968	1768	SBrkr	1768	2	NaN	
79	80	3	1915	1013	SBrkr	1526	1	NaN	
80	81	4	1994	990	SBrkr	990	1	NaN	
...	...	...	...	...	...	...	...	...	
1310	1311	8	1872	684	SBrkr	2358	2	NaN	
1366	1367	5	1985	833	SBrkr	833	1	NaN	
1407	1408	5	1970	630	SBrkr	630	1	NaN	
1408	1409	5	1974	896	SBrkr	1792	2	NaN	
1411	1412	5	2006	1140	SBrkr	1140	1	NaN	

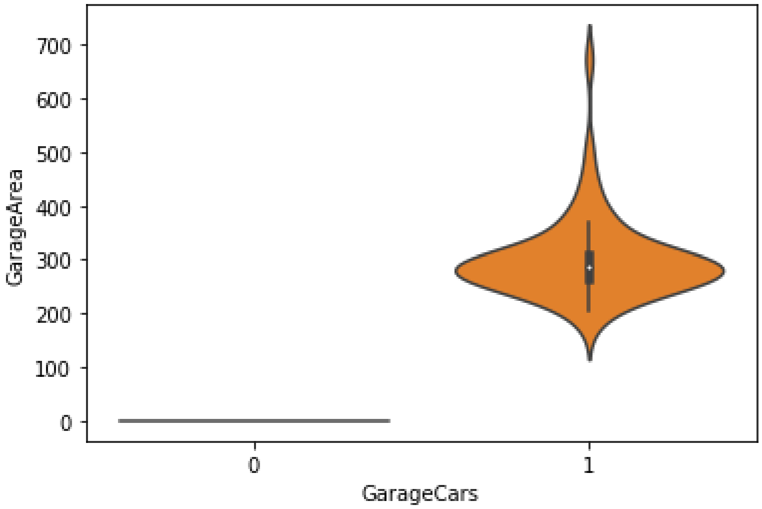
81 rows × 11 columns

In [134]:

```
sns.violinplot('GarageCars', 'GarageArea', data=HS[HS['GarageType'].isnull()])
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18cf170a288>
```



In [135]:

```
HS['GarageType'] = np.where(HS['GarageCars']==1 & HS['GarageType'].isnull(),'D
etchd',HS['GarageType'])
HS['GarageType'] = np.where(HS['GarageCars']==0 & HS['GarageType'].isnull(),'N
oGarage',HS['GarageType'])
```

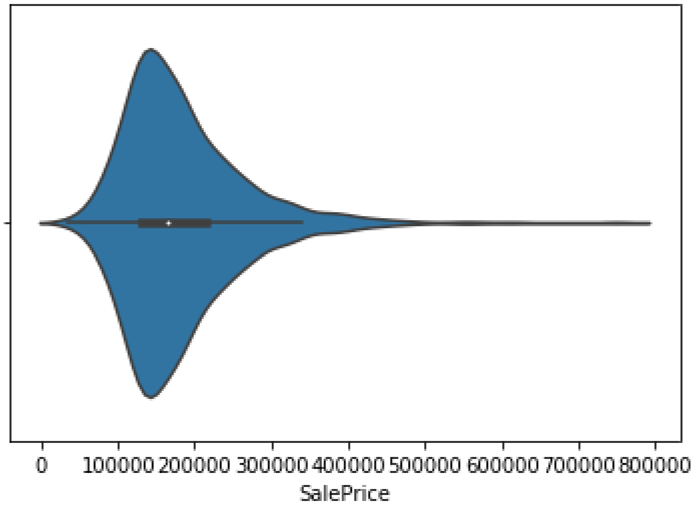
In [136]:
```
HS.isnull().sum()
```

Out[136]:
```
Id             0
OverallQual    0
YearBuilt      0
TotalBsmtSF    0
Electrical     0
GrLivArea      0
FullBath       0
GarageType     0
GarageCars     0
GarageArea     0
SalePrice      0
dtype: int64
```

Below is the plot which shows maximum number of houses under Sale Price

In [156]:
```
sns.violinplot('SalePrice', data = HS)
```

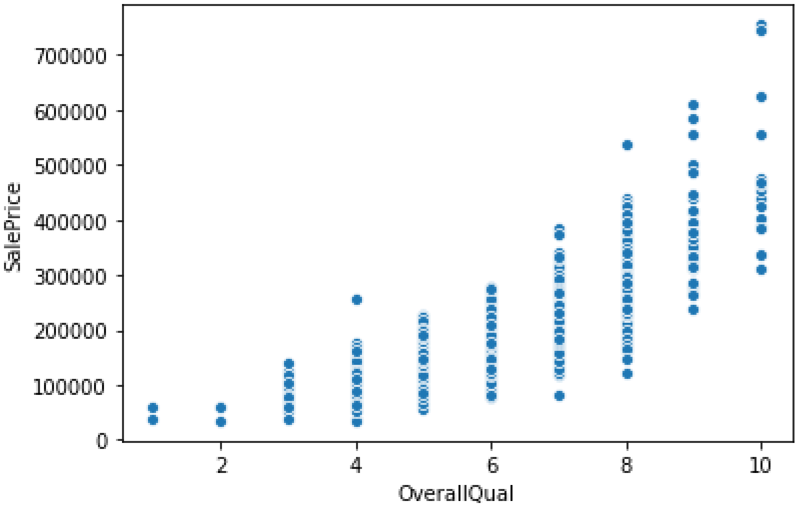
Out[156]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18cf2b307c8>
```



Maximum number of houses lies between Prince range of 150000 to 200000

Relationship Between Overall Quality and SalePrice

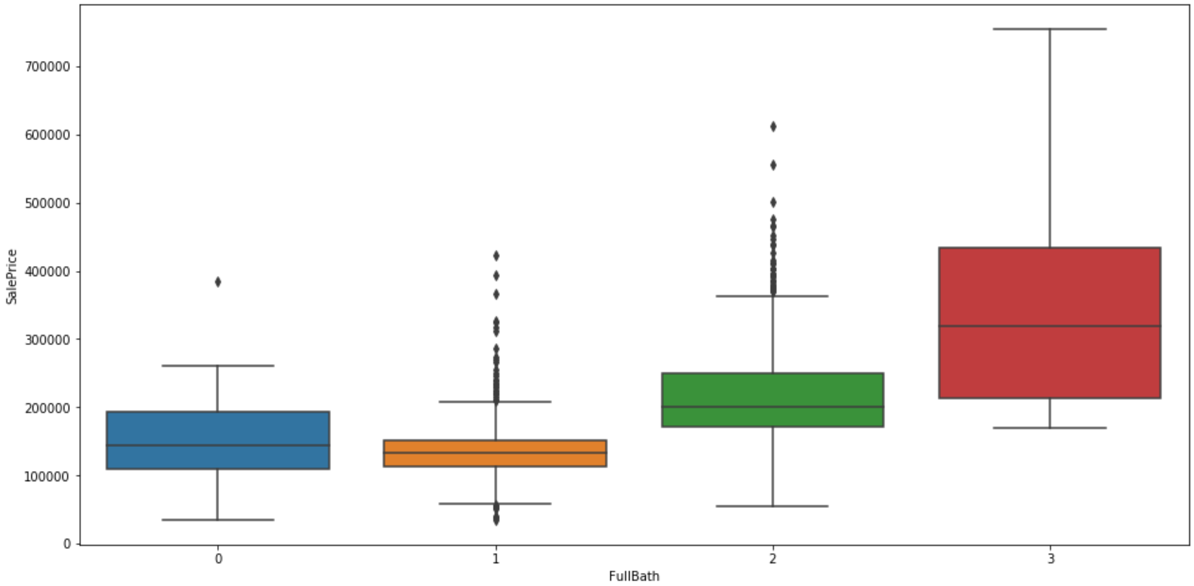
```
In [157]: sns.scatterplot(x='OverallQual', y = 'SalePrice', data = HS)

Out[157]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf2b6eac8>
```



It states that Sale Price inceases as quality increases.

Relationship Between Full Bath and SalePrice

```
In [159]: plt.subplots(figsize = (16,8))
          sns.boxplot(x='FullBath', y='SalePrice', data=HS)

Out[159]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf2c416c8>
```



It states that Sale Price inceases as Full Bath increases.

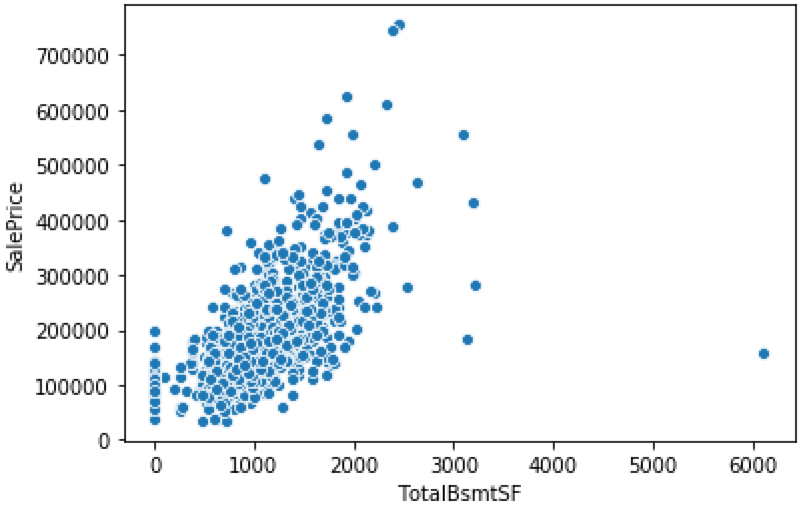
Outliers Detection and Handling

Outliers reflect a mixture of observations from a population other than the target population, analyzing data with such outliers produces biased estimations of the target population parameters. So, we will detect them nd handle accordingly.

Let's start with TotalBsmtSF column and SalePrice coulmn via scatterplot to detect and handle the outlier, so that, we can analyze the relationship between both of them.

```
In [137]: sns.scatterplot(x='TotalBsmtSF', y='SalePrice', data= HS)
```

```
Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf1478b08>
```



```
In [138]: HS.sort_values(by='TotalBsmtSF', ascending = False)
```

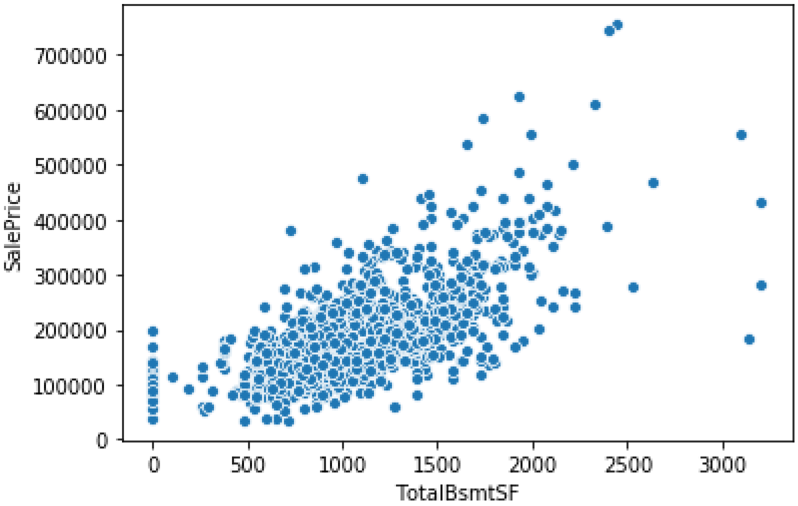
```
Out[138]:
             Id  OverallQual  YearBuilt  TotalBsmtSF  Electrical  GrLivArea  FullBath  GarageType  Ga
      1260  1261          10       2008         6110       SBrkr       5642         2      Attchd
       313   314           8       2003         3206       SBrkr       1629         2      Attchd
       475   476           8       1992         3200       SBrkr       3228         3      Attchd
       500   501          10       2007         3138       SBrkr       4676         3     BuiltIn
       419   420          10       2008         3094       SBrkr       2402         2      Attchd
       ...   ...         ...        ...          ...         ...        ...       ...         ...
      1002  1003           4       1957            0       SBrkr        845         1      Detchd
       679   680           4       1930            0       SBrkr       1092         2    NoGarage
       621   622           5       1950            0       SBrkr       1048         1      Detchd
      1144  1145           5       1954            0       SBrkr       1124         1    NoGarage
       709   710           3       1950            0       FuseF       1040         2      Detchd

      1418 rows × 11 columns
```

1) Through graph we know there is one row in data which is effecting our presentation and analysis.

2) Through HS.sort_values(by='TotalBsmtSF', ascending = False), we found that row# 1260 is outlier.

I will drop that row!

```
In [139]: HS.drop([1260], inplace=True)
In [142]: sns.scatterplot(x='TotalBsmtSF', y='SalePrice', data= HS)

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf120bc08>
```

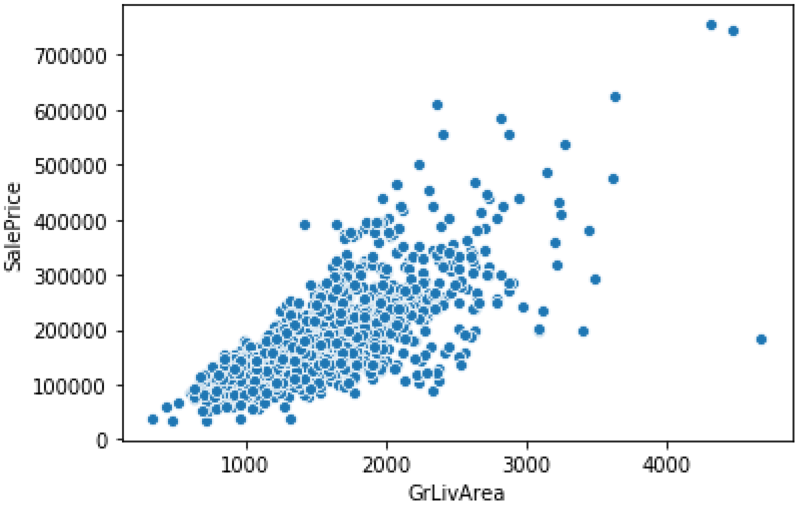


Outlier got dropped and it is looking better than the previous scatter plot and is easy to depict the relationship between SalesPrice and TotalBsmtSF.

Average Sales Price is from 100000 to 200000 till BsmtSF 1000. After that, there are changes in saleprice according to increasing BsmtSF.

Relationship between GrLivArea and SalesPrice

```
In [144]: sns.scatterplot(x='GrLivArea', y='SalePrice', data= HS)

Out[144]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf153c9c8>
```



```
In [146]: HS.sort_values(by='GrLivArea', ascending = False)
```

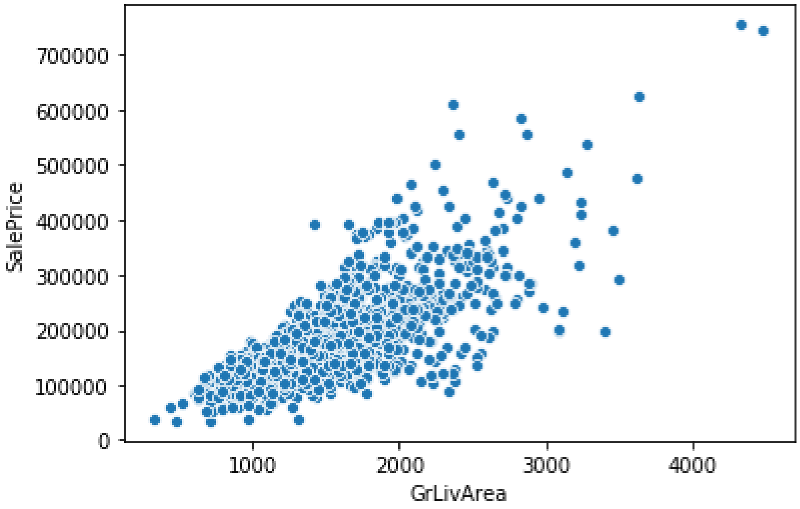
Out[146]:

	Id	OverallQual	YearBuilt	TotalBsmtSF	Electrical	GrLivArea	FullBath	GarageType	G
500	501	10	2007	3138	SBrkr	4676	3	BuiltIn	
1147	1148	10	1996	2396	SBrkr	4476	3	Attchd	
665	666	10	1994	2444	SBrkr	4316	3	Attchd	
1134	1135	10	1995	1930	SBrkr	3627	3	Attchd	
169	170	10	1892	1107	SBrkr	3608	2	Detchd	
...	...	...	...	...	...	...	...	...	
505	506	4	1920	528	SBrkr	605	1	NoGarage	
27	28	4	1927	520	SBrkr	520	1	Detchd	
886	887	2	1949	480	FuseA	480	0	Detchd	
1066	1067	2	1920	290	FuseF	438	1	Detchd	
510	511	1	1946	0	FuseF	334	1	NoGarage	

1417 rows × 11 columns

```
In [147]: HS.drop([500], inplace=True)
          sns.scatterplot(x='GrLivArea', y='SalePrice', data= HS)
```

```
Out[147]: <matplotlib.axes._subplots.AxesSubplot at 0x18cf15a1708>
```



Average Sales Price is from 100000 to 200000 till GrLivArea 2000. After that, there are changes in saleprice according to increasing GrLivArea.

```
In [ ]:
```