Introduction

The aim to conduct the coursework is to analyze the AmesHousing.txt file with the help of google colab and python and its software library. This data hold the information of the Ames,Iowa housing values for individual housing property that has been sold in Ames between 2006 to 2010. After converting this txt file to csv with the help of panda dataframe we can get basic understanding regarding the data that provided in the file.

Here first after running the below code there will be one link that redirects you to get your authentication code and needs to enter it on an appeared textbox. That's how colab authenticate the drive and try to look out towards that specific drive storage.

```
#importing drive and mounting i to the colab.

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

Below I am importing panda as it is going to help us with the analyze, format, manipulation and cleaning the data.

Using the NumPy that helps us to works with the arrays. Furthermore it will help us to projects through the linear algebra or metrices.

After I Have imported the matplotlib which is one of the framing library that helps us to visualize in static and interactive manner.

Using seaborn mainly to visualize the data and that is build on top of the matplotlib. It works easily with pandas dataframe and library.

For output of plotting command I am using the inline matplotlib function. It will print just as below after the code that produced it. And it stores resulting plots to the colab notebook.

```
#importing panda and np
import pandas as pd
import numpy as np

# importing seaborn and matplotlib library for visualization
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

Here I am reading the txt file from google drive with the help of the panda. Provided txt file will be in tab-delimited so I am using sep='\t' to tell the read_csv function to separation will be in the tab character. After processing the text file I stored that in the df_amshscsv variable and printed it. So We can get a simple visualization idea of how data looks like at first.

```
df_amshscsv=pd.read_csv("/content/drive/My Drive/AmesHousing.txt",sep='\t')
df_amshscsv
```

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Con
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	...	
2925	2926	923275080	80	RL	37.0	7937	Pave	NaN	IR1	
2926	2927	923276100	20	RL	NaN	8885	Pave	NaN	IR1	
2927	2928	923400125	85	RL	62.0	10441	Pave	NaN	Reg	
2928	2929	924100070	20	RL	77.0	10010	Pave	NaN	Reg	
2929	2930	924151050	60	RL	74.0	9627	Pave	NaN	Reg	

2930 rows × 82 columns

Here I am printing the numbers of rows and columns with the help of shape property.

After running we can easily get the idea of the shape of the dataframe. For me it is 2930 rows and 82 column in the dataset.

```
df_amshscsv.shape
```

```
(2930, 82)
```

First and foremost I tried to make a data formatting. And it can be done in many ways.

Lets look out at some point here. Space in column name can be difficult sometimes for any kind of operation so I removed the space and replace it with the underscores. And I will remove the duplicates row(if any) from the dataframe with the drop_duplicates() function.

```
df_amshscsv.columns = df_amshscsv.columns.str.replace(' ', '_')
df_amshscsv.drop_duplicates()
# #After running the above query I can say that there is not any single data that duplicat
```

	Order	PID	MS_SubClass	MS_Zoning	Lot_Frontage	Lot_Area	Street	Alle
0	1	526301100	20	RL	141.0	31770	Pave	Nal
1	2	526350040	20	RH	80.0	11622	Pave	Nal
2	3	526351010	20	RL	81.0	14267	Pave	Nal
3	4	526353030	20	RL	93.0	11160	Pave	Nal
4	5	527105010	60	RL	74.0	13830	Pave	Nal
...	...	...	...	...	...	...	...	.
2925	2926	923275080	80	RL	37.0	7937	Pave	Nal
2926	2927	923276100	20	RL	NaN	8885	Pave	Nal
2927	2928	923400125	85	RL	62.0	10441	Pave	Nal
2928	2929	924100070	20	RL	77.0	10010	Pave	Nal
2929	2930	924151050	60	RL	74.0	9627	Pave	Nal

2930 rows × 82 columns

```
#In order to obtain the data types, we need to run the following command.
df_amshscsv.dtypes
# or
# train_data.info()
```

```
Order              int64
PID                int64
MS_SubClass        int64
MS_Zoning         object
Lot_Frontage     float64
                   ...
Mo_Sold            int64
Yr_Sold            int64
Sale_Type         object
Sale_Condition    object
SalePrice          int64
Length: 82, dtype: object
```

As a part of this tasks I don't need this two columns with the orderId and PID(Parcel Identification Number). So I dropped them with the help of drop function and I passed the parameter as an array of two columns and second parameter as inplace = true which means it will remove the existing dataframe and won't create the new dataframe and axis=1 describe that

we are trying to remove the columns parts if we write axis=0 that means we are trying to remove the row. And after that printing the dataframe as usual.

```
df_amshscsv.drop(['Order', 'PID'] ,inplace=True ,axis=1);
```

```
df_amshscsv
```

	MS_SubClass	MS_Zoning	Lot_Frontage	Lot_Area	Street	Alley	Lot_Shape	Land
0	20	RL	141.0	31770	Pave	NaN	IR1	
1	20	RH	80.0	11622	Pave	NaN	Reg	
2	20	RL	81.0	14267	Pave	NaN	IR1	
3	20	RL	93.0	11160	Pave	NaN	Reg	
4	60	RL	74.0	13830	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	
2925	80	RL	37.0	7937	Pave	NaN	IR1	
2926	20	RL	NaN	8885	Pave	NaN	IR1	
2927	85	RL	62.0	10441	Pave	NaN	Reg	
2928	20	RL	77.0	10010	Pave	NaN	Reg	
2929	60	RL	74.0	9627	Pave	NaN	Reg	

2930 rows × 80 columns

In the below section I am trying to identify how many record does have the null value with the help of the isnull() function. And will get the true and false as displayed in the above result.

Cell value with true has null and with the false it is not null.

So it will be easy to figure out the null value if any.

```
df_amshscsv.isnull()
```

	MS_SubClass	MS_Zoning	Lot_Frontage	Lot_Area	Street	Alley	Lot_Shape	Land
0	False	False	False	False	False	True	False	
1	False	False	False	False	False	True	False	
2	False	False	False	False	False	True	False	
3	False	False	False	False	False	True	False	
4	False	False	False	False	False	True	False	
...	...	...	...	...	...	...	...	

From the above function it seems easy to get the null value but what if we wants to identify the count of null value per column. In order to get that column wise result here are some operations that I am trying to make with the help of pandas.

First I set an options in pandas for displaying the max rows so I dont miss out any columns that does have the missing values. With the help of the set_option function and passing 'display.max_rows' as a 'true' will not limit the result from the dataframe.

After I use the isnull() function with the sum and with the to_frame function with the passing parameter as 'Null_Count'.

to_frame function will convert the series the dataframe. So I after that it will be easy for us for any oepration.

```
# Here I am retrieving the null value and sum it and frame
pd.set_option('display.max_rows', None)

# Converting the sum of null value into the dataframe with the column name Null_count
s = df_amshscsv.isnull().sum().to_frame('Null_Count')

# Soring the dataframe with the function sort_valuesand passing two parameter, one is by t
# This will sort the result from the max null value to least null value.

s.sort_values(by=['Null_Count'], ascending=False)
```

	Null_Count
Pool_QC	2917
Misc_Feature	2824
Alley	2732
Fence	2358
Fireplace_Qu	1422
Lot_Frontage	490
Garage_Cond	159
Garage_Finish	159
Garage_Yr_Blt	159
Garage_Qual	159
Garage_Type	157
Bsmt_Exposure	83
BsmtFin_Type_2	81
Bsmt_Cond	80
Bsmt_Qual	80
BsmtFin_Type_1	80
Mas_Vnr_Area	23
Mas_Vnr_Type	23
Bsmt_Full_Bath	2
Bsmt_Half_Bath	2
Bsmt_Unf_SF	1
Garage_Cars	1
Electrical	1
Total_Bsmt_SF	1
BsmtFin_SF_1	1
Garage_Area	1
BsmtFin_SF_2	1
Paved_Drive	0
Sale_Condition	0
Half_Bath	0
Bedroom_AbvGr	0
Kitchen_AbvGr	0
Kitchen_Qual	0

Kitchen_Qual	0
TotRms_AbvGrd	0
Functional	0
Fireplaces	0
Sale_Type	0
Yr_Sold	0
Mo_Sold	0
Misc_Val	0
Pool_Area	0
Screen_Porch	0
3Ssn_Porch	0
Enclosed_Porch	0
Open_Porch_SF	0
Full_Bath	0
Wood_Deck_SF	0
MS_SubClass	0
Central_Air	0
Gr_Liv_Area	0
Overall_Cond	0
Lot_Area	0
Street	0
Lot_Shape	0
Land_Contour	0
Utilities	0
Lot_Config	0
Land_Slope	0
Neighborhood	0
Condition_1	0
Condition_2	0
Bldg_Type	0
House_Style	0
Overall_Qual	0
Year_Built	0
Low_Qual_Fin_SF	0

Year_Remod/Add	0
Roof_Style	0
Roof_Matl	0
Exterior_1st	0
Exterior_2nd	0
Exter_Qual	0
Exter_Cond	0

Here I am resetting the options for max rows display

Heating	0

```
#To reset the display of max rows from dataframe.
pd.reset_option('display.max_rows')
```

```
# From the above result I am going to remove that top two columns which does have max coun
```

```
to_drop = ['Pool_QC', 'Alley']
```

```
# In the drop function I have passed the name of the columns to drop, inplace=True that me
```

```
df_amshscsv.drop(to_drop,axis=1,inplace=True)
```

```
df_amshscsv['Electrical'].value_counts()
```

```
SBrkr    2682
FuseA     188
FuseF      50
FuseP       8
Mix         1
Name: Electrical, dtype: int64
```

Above cell code and below one will helps us to identify that what we can replace exaclty in the place of missing value.

As a Electrical column is object type and while referring the document that has provided it shows that it does have upto five value excluding the missign na value.

So with the helps of value_counts function we can get the unique output count from the specific series excluding na and after we can replace it with na value as below code.

From the above code we can see that 'SBrkr' value is represented 2682 times in that specific column.

```
df_amshscsv['Electrical'] = df_amshscsv['Electrical'].fillna('SBrkr')
```

Here in the below some codes I am trying to replace the missing value with the suitable value I think. First we need to identify the missing if there is any missing value and we can replace it.

For identifying the missing values we use the same data we have printed before.

```
# Here I am filling missing value with the 0 if the column type is integer and for the som
# And if there is an object type column we simply get the na or none from

df_amshscsv['Lot_Frontage'].fillna(0, inplace = True)
df_amshscsv['Garage_Yr_Blt'].fillna(df_amshscsv['Garage_Yr_Blt'].mean(), inplace = True)
df_amshscsv['Bsmt_Full_Bath'].fillna(0, inplace = True)
df_amshscsv['Bsmt_Half_Bath'].fillna(0, inplace = True)

df_amshscsv['Bsmt_Unf_SF'].fillna(df_amshscsv['Bsmt_Unf_SF'].mean(), inplace = True)
df_amshscsv['BsmtFin_SF_2'].fillna(df_amshscsv['BsmtFin_SF_2'].mean(), inplace = True)
df_amshscsv['Total_Bsmt_SF'].fillna(df_amshscsv['BsmtFin_SF_2'].mean(), inplace = True)

df_amshscsv['Misc_Feature'] = df_amshscsv['Misc_Feature'].replace({np.nan: 'None'})
df_amshscsv['Misc_Val'] = df_amshscsv['Misc_Val'].replace({np.nan: 'No misc. values'})
df_amshscsv['Fence'] = df_amshscsv['Fence'].replace({np.nan: 'NA'})

df_amshscsv['Fireplace_Qu'].fillna(0, inplace = True)

df_amshscsv['Garage_Type'] = df_amshscsv['Garage_Type'].replace({np.nan: 'NA'})
df_amshscsv['Garage_Qual'] = df_amshscsv['Garage_Qual'].replace({np.nan: 'NA'})
df_amshscsv['Garage_Cond'] = df_amshscsv['Garage_Cond'].replace({np.nan: 'NA'})
df_amshscsv['Garage_Finish'] = df_amshscsv['Garage_Finish'].replace({np.nan: 'NA'})
df_amshscsv['Garage_Cars'].fillna(df_amshscsv['Garage_Cars'].mean(), inplace = True)
df_amshscsv['Garage_Area'].fillna(df_amshscsv['Garage_Area'].mean(), inplace = True)

df_amshscsv['Bsmt_Exposure'] = df_amshscsv['Bsmt_Exposure'].replace({np.nan: 'NA'})
df_amshscsv['Bsmt_Cond'] = df_amshscsv['Bsmt_Cond'].replace({np.nan: 'NA'})
df_amshscsv['BsmtFin_Type_2'] = df_amshscsv['BsmtFin_Type_2'].replace({np.nan: 'NA'})
df_amshscsv['BsmtFin_Type_1'] = df_amshscsv['BsmtFin_Type_1'].replace({np.nan: 'NA'})
df_amshscsv['Bsmt_Qual'] = df_amshscsv['Bsmt_Qual'].replace({np.nan: 'NA'})
df_amshscsv['BsmtFin_SF_1'].fillna(df_amshscsv['BsmtFin_SF_1'].mean(), inplace = True)
df_amshscsv['BsmtFin_SF_1'].fillna(df_amshscsv['BsmtFin_SF_1'].mean(), inplace = True)

df_amshscsv['Mas_Vnr_Type'] = df_amshscsv['Mas_Vnr_Type'].replace({np.nan: 'None'})
df_amshscsv['Mas_Vnr_Area'].fillna(0, inplace = True)
```

Now with the describe function we are looking at the statistical abstract for our dataset. The describe () function is used to induce descriptive statistics that epitomize the central tendency, dissipation and shape of a dataset's distribution, eliminate NaN values.

```
df_amshscsv['Total_Bsmt_SF'].describe()
```

```
count    2930.000000
mean     1051.272602
std       440.928503
min         0.000000
25%       793.000000
50%       990.000000
```

```
75%      1301.500000
max      6110.000000
Name: Total_Bsmt_SF, dtype: float64
```

Now we have replace all the missing value with the appropriate values. So we dont get any trouble with that.

```
df_amshscsv.isnull().sum()
```

```
MS_SubClass       0
MS_Zoning         0
Lot_Frontage      0
Lot_Area          0
Street            0
                 ..
Mo_Sold           0
Yr_Sold           0
Sale_Type         0
Sale_Condition    0
SalePrice         0
Length: 78, dtype: int64
```

Exploratory Data Analysis and Visualization

Now we will use seaborn and matplotlib to exploratory data analysis and visualization of the data with different plots and views.

First I am trying to find out the co relation of the SalePrice column with the other columns. so it will be easy to get the idea of price of the house depends on which column.

After running the below code it is easily understandable that selling price mainly depends on the overall quality , Gr_Liv_Area and Garage_Cars and so on.

The value of relation will be vary from the plus 1 to minus 1. It means that plus 1 is the highest cell in correlation with the SalePrice and minus 1 is lowest one in related with the SalePrice. And on top of that the darker color shed also describe the most related cell with the SalePrice and lighter one with the least related column with the SalePrice.

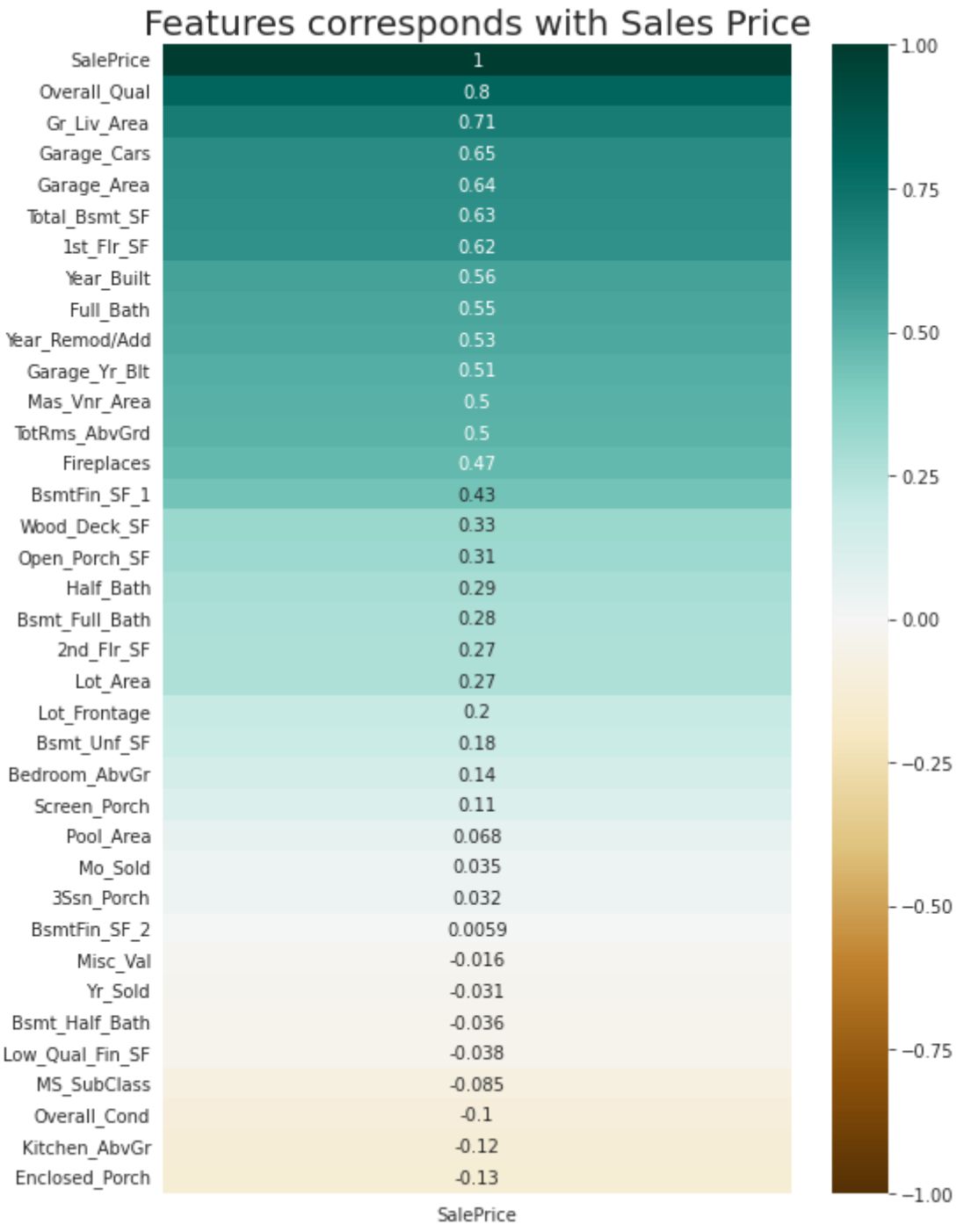
```
# With the plt.figure I am setting the size of the output with the function figure and pas
# is height of the plot.
plt.figure(figsize=(8, 12))

# For this co relation I am using the heatmap and it will plot the rectangular data as a c
# With the heatmap function I have passed some argument that is described as below.
# First parameter is dataframe co relation with the respect to the sale price column.
# vmin, vmax are used to values to anchor the colormap.
# annot = true will print the correlation value over the each cell.
# cmap function will customize the color of the each cell.

heatmap = sns.heatmap(df_amshscsv.corr()[['SalePrice']].sort_values(by='SalePrice', ascend
```

```
# With the set_title function I am passing the name of the title and fontsize as a fontdic

heatmap.set_title('Features corresponds with Sales Price', fontdict={'fontsize':20});
```

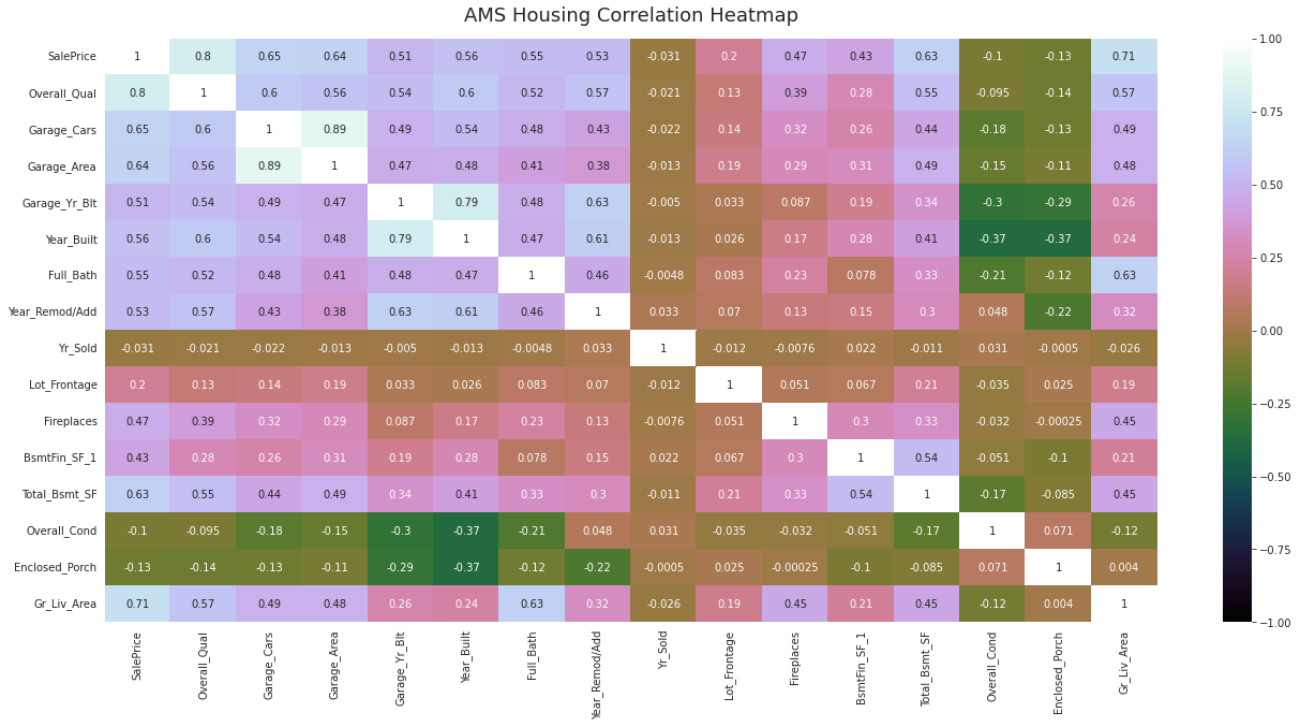


The aboe and below plot describe the co relation. As I said the above one describe the corelation only with in the center of the saleprice and the below one provide the co relation of the whole dataframe.

As this data is property selling so I assume that how the sale price is related with the others.

And for the overall dataframe correlation I have written the below heatmap. Here I haven't include the whole dataframe but just some columns that highly related with the SalePrice. And it shows how each and every column is related to each other.

```
plt.figure(figsize=(22, 10))
df_corr_amshcsv = df_amshscsv[['SalePrice','Overall_Qual','Garage_Cars','Garage_Area','Gar
heatmap = sns.heatmap(df_corr_amshcsv.corr(), vmin=-1, vmax=1, annot=True, cmap='cubehelix
heatmap.set_title('AMS Housing Correlation Heatmap', fontdict={'fontsize':18}, pad=16);
```



Here I have used the scatter plot to plot out some of the findings with the help of SalePrice, Gr_liv_area and Street column. And hue will group the data point based on the street value with the different colors.

Findings from the plot - After running easily we can get the idea that selling is major depends on the Gr_Liv_Area and this property is easily accessible to the pave street as we can see with the help of hue. And we also can say that people are finding the property that is high in ground level and also with the pave street accessible.

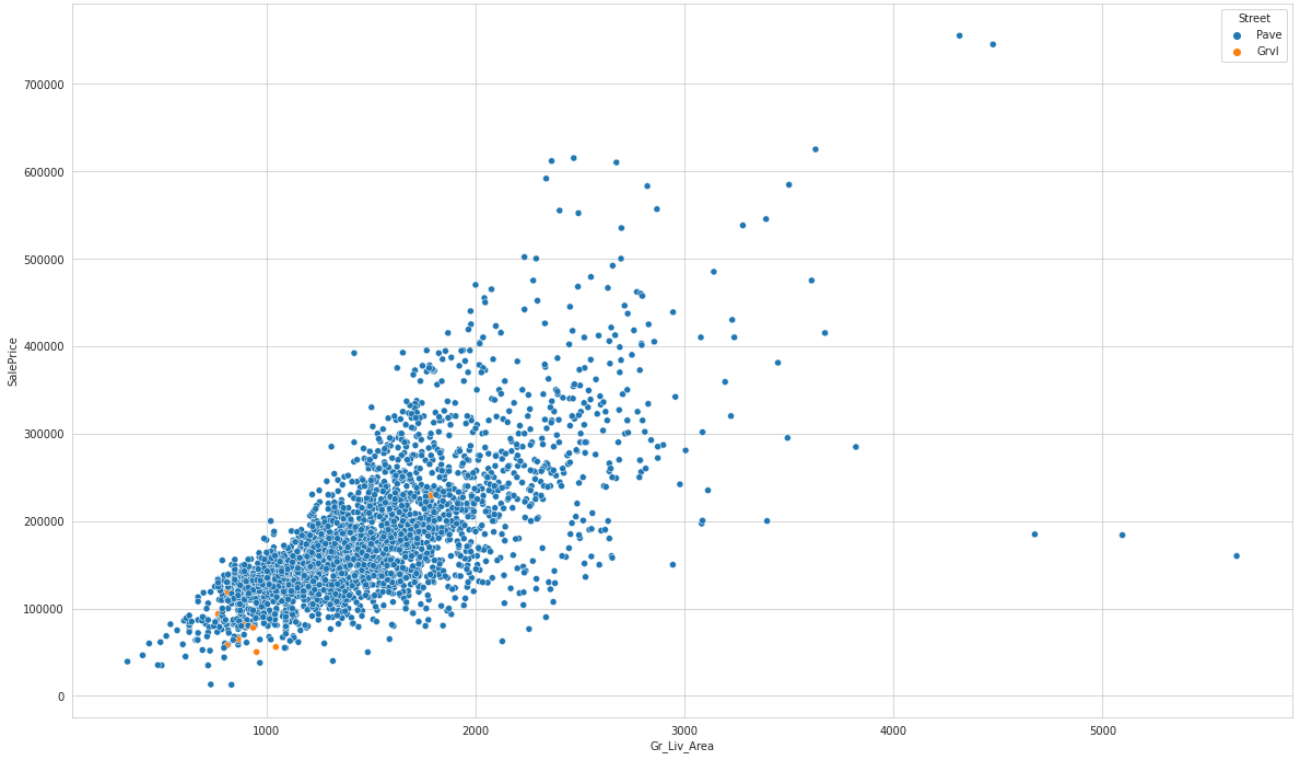
Selling price is higher for the property that is having the high numbers in above grade living area.

And we can get the outliers too from the below plot.

```
plt.figure(figsize=(20, 12))

# using scatterplot to get the findings.
sns.scatterplot(x="Gr_Liv_Area", y="SalePrice", hue = 'Street', data=df_amshscsv)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd273fc6cd0>
```



Here I ma trying to find out if the price per square foot has been changed over the span of four years or not.

I used the boxplot and with dividing the saleprice with the Gr_Liv_Area(which is in square feet already). We can easily get the price per square foot. Stored that value in dataframe column

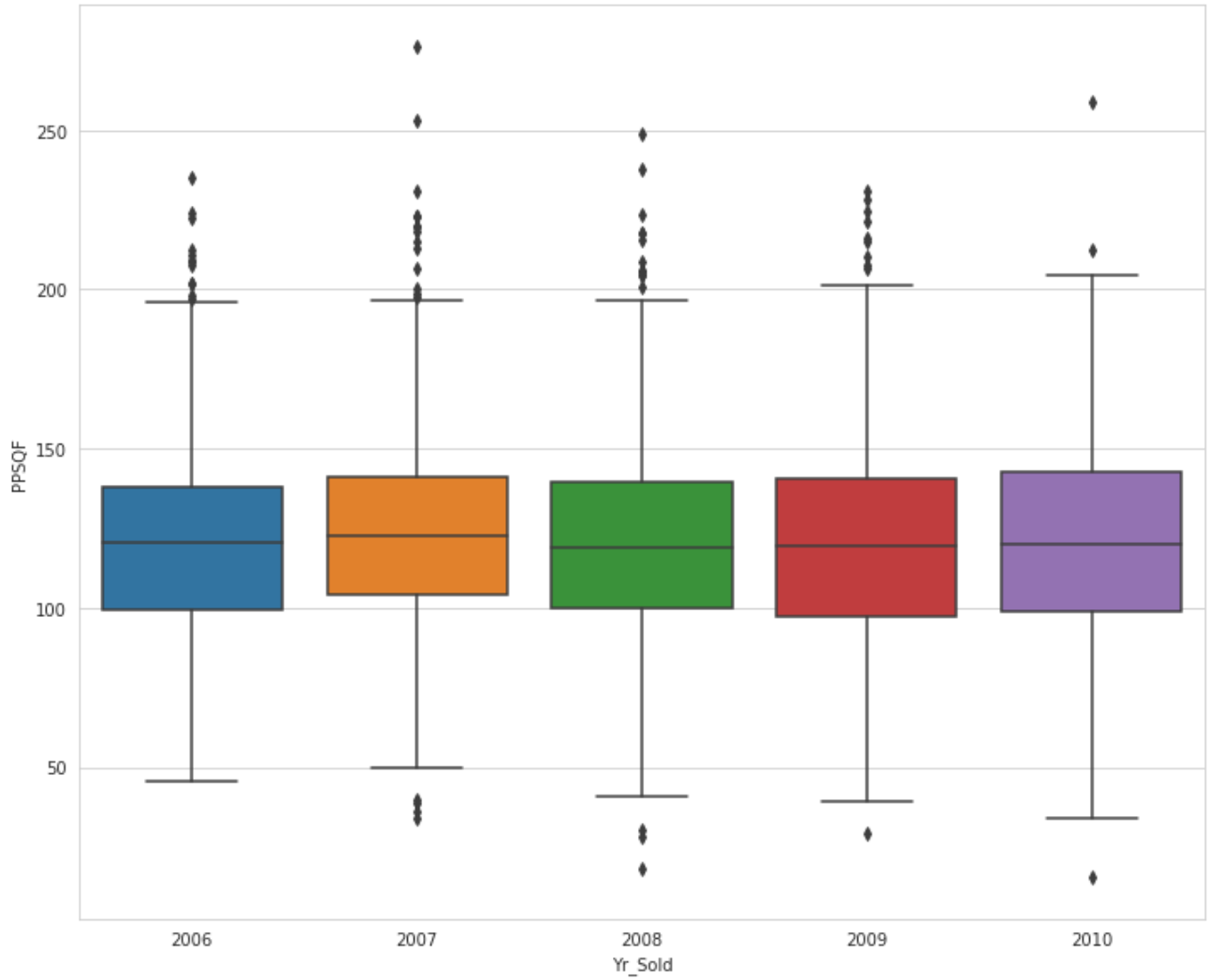
PPSQF and use that in boxplot as below code.

Findings from the plot - This Plot easily states that there are some property that hase been sold over the price per square foot. These property plots too far from the median value and interquartile range.

```
#lets get the histogram plot for the price per square foot year wise.

plt.figure(figsize=(12, 10))
df_amshscsv['PPSQF'] = df_amshscsv['SalePrice']/df_amshscsv['Gr_Liv_Area']
sns.boxplot(x="Yr_Sold", y="PPSQF",data=df_amshscsv)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2746c26d0>
```



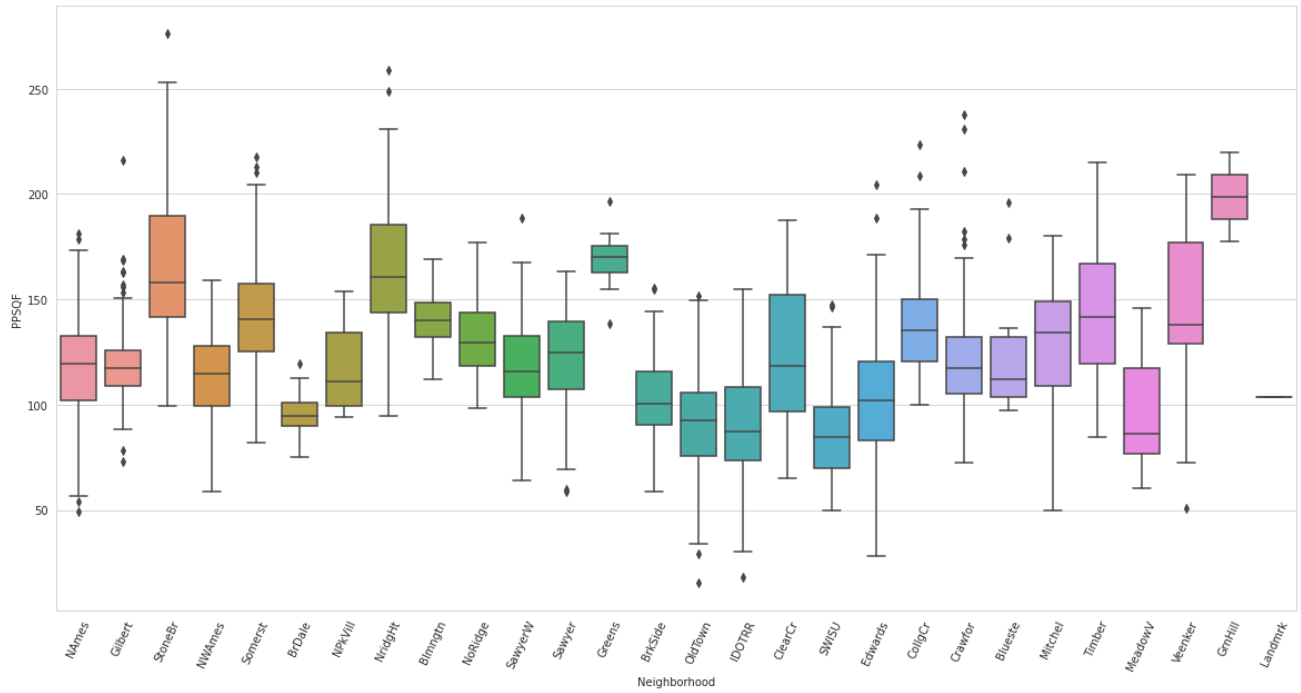
Here I have used the boxplot to visualize the PPSQF column versus the neighbourhood.

Findings from the plot - This Plot stats that price per square foot are higher on the some areas like StoneBr, NridgHt. And sold price is lower in oldtown and so.

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation=65)
```

```
sns.boxplot(x="Neighborhood", y="PPSQF", data=df_amshscsv)
```

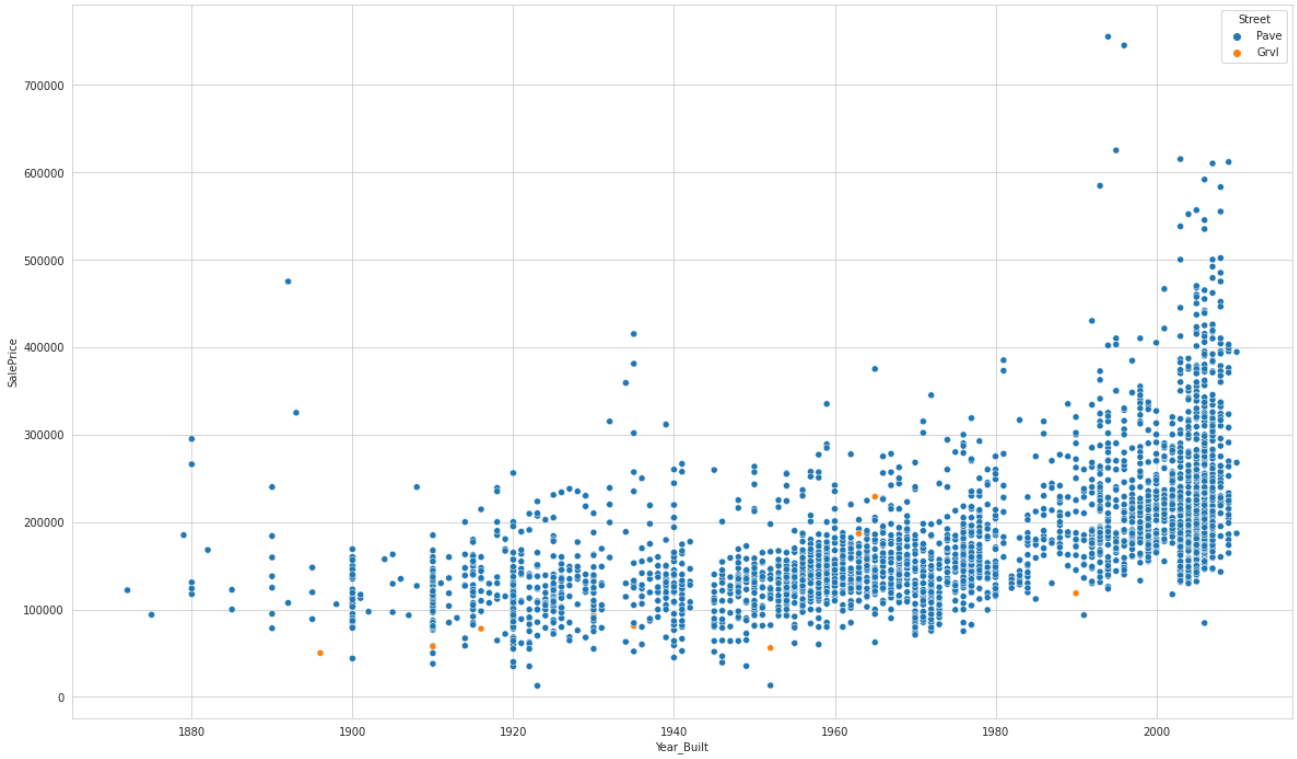
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2746b4790>
```




```
# Setting the grid style to whitegrid for better understanding of the plot.
sns.set_style("whitegrid")
```

```
plt.figure(figsize=(20, 12))
sns.scatterplot(x="Year_Built", y="SalePrice", hue = 'Street', data=df_amshscsv)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2745e4f10>
```

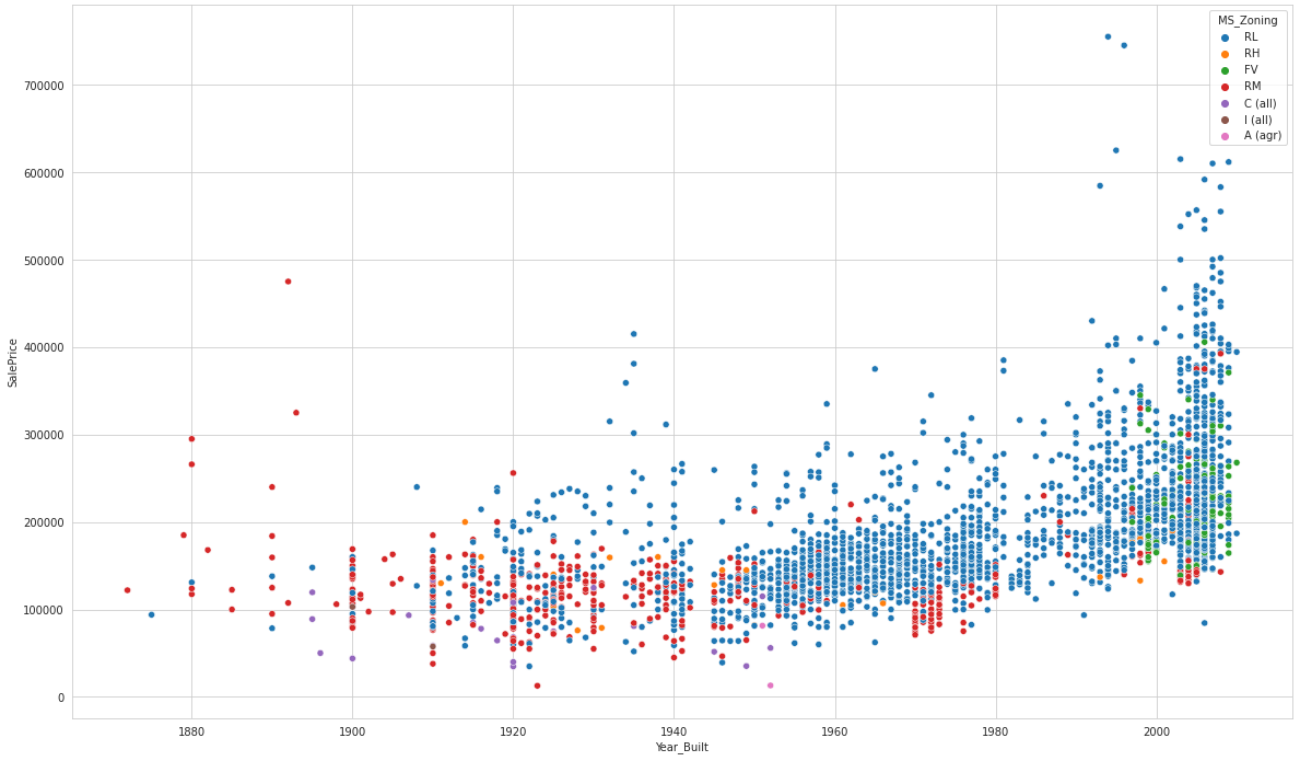


Here I am using the scatterplot to get the data from the year build vs the saleprice on top of the MS Zoning.

Findings from the plot - That's an obvious thing that property that is old has lower price then the young ones. but here plot says that property that has been built after 1950 and main reason behind selling is the zone. Most of them falls under the low density zone so people are going countryside and wants to live in low density zone.

```
plt.figure(figsize=(20, 12))
sns.scatterplot(x="Year_Built", y="SalePrice", hue = 'MS_Zoning', data=df_amshscsv)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd274371c10>
```



Here I have creted one boolean column with the name of IsGarage. That will identify if this property does have garage or not.

```
# df_amshscsv["Garage_Cars"]
df_amshscsv["IsGarage"] = df_amshscsv["Garage_Cars"] > 0
```

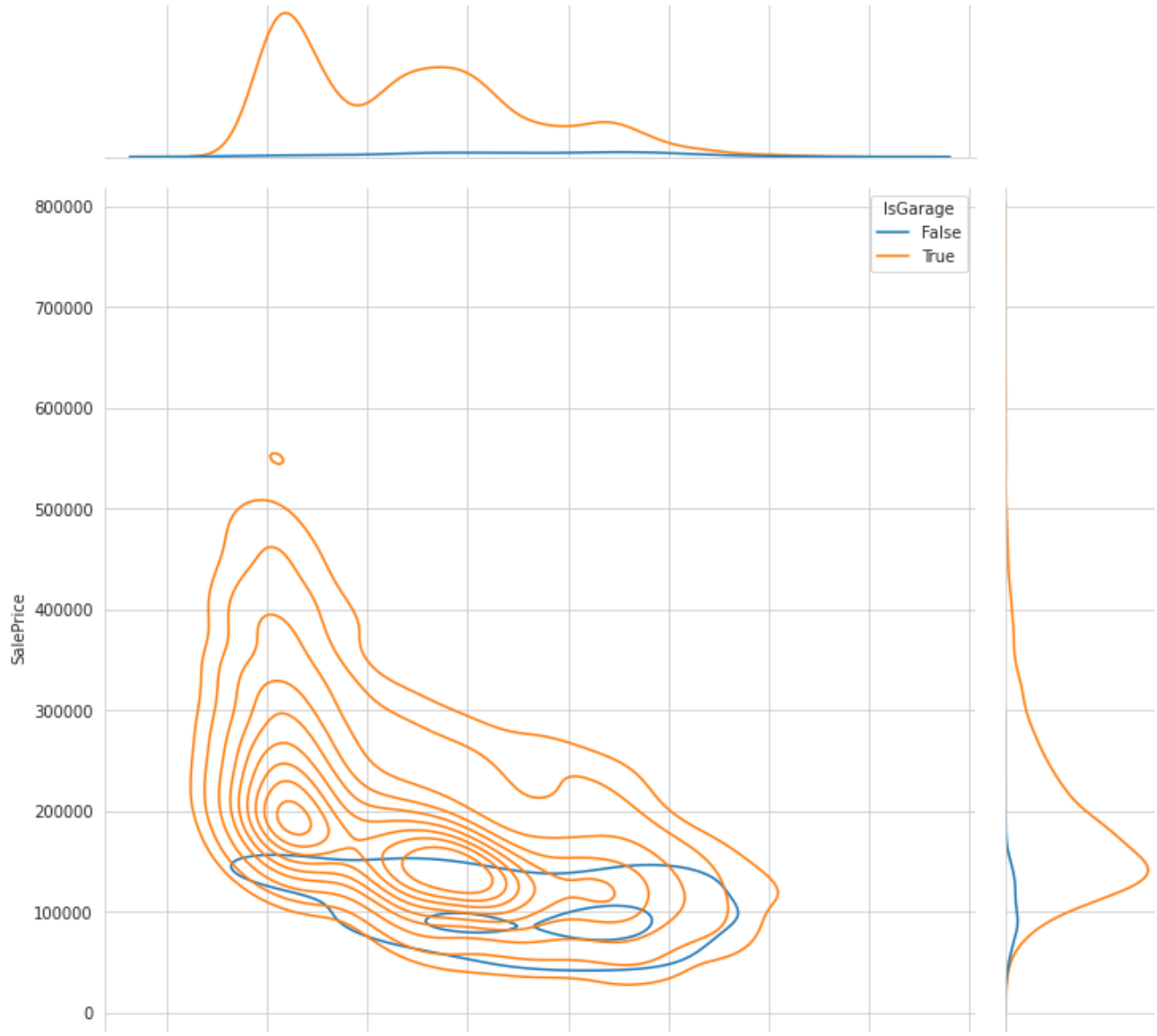
Here I have used the joint plot with the after calculating the house age at that time when property has been sold. With the help of the Yr_sold and Year_build I got that value.

Visualized this plot on the kde(Kernel density estimation) and used IsGarage as a hue parameter.

Findings from the plot - Here I can say that property with the garage and young one used to sell on a higher price then the property that does not have a garage and old.

```
df_amshscsv["House_Age"] = df_amshscsv["Yr_Sold"] - df_amshscsv["Year_Built"]
sns.jointplot(x="House_Age", y="SalePrice", data=df_amshscsv,kind="kde", hue="IsGarage" ,
```

```
<seaborn.axisgrid.JointGrid at 0x7fd2742a8d10>
```



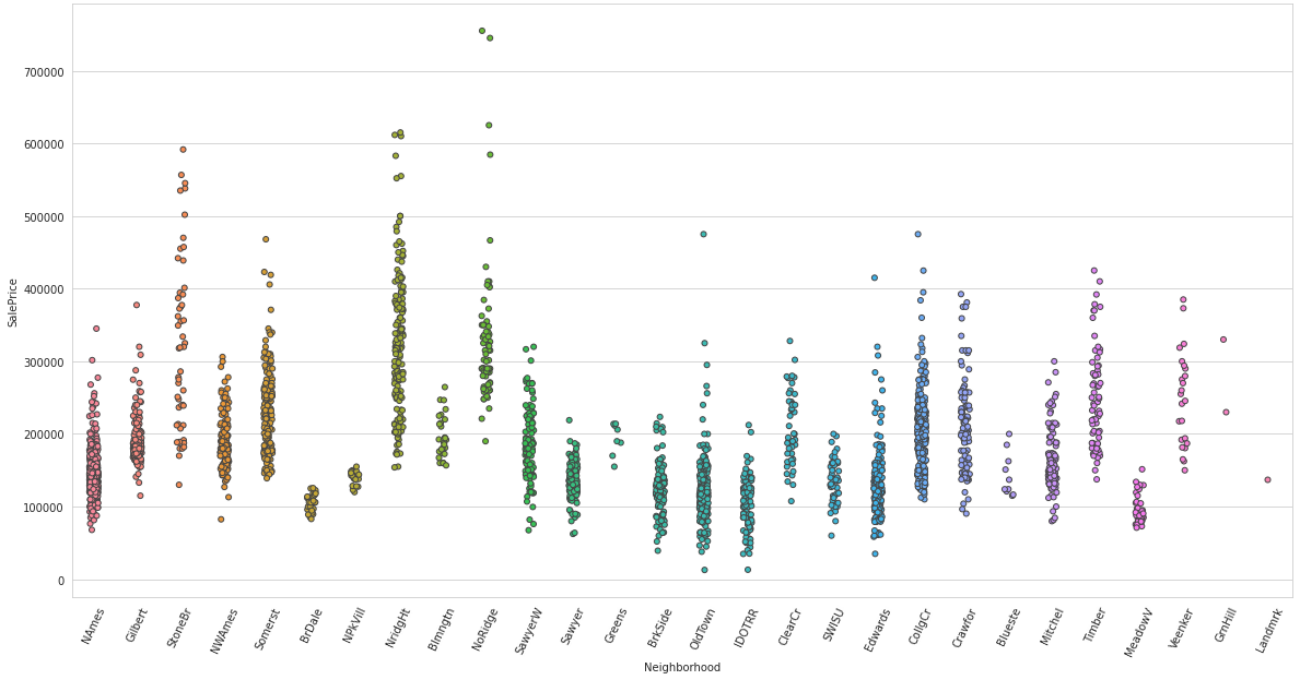
Using the stripplot to get the idea of the sales pricing vs neighbour. Likewise I did for the price per square foot vs neighbourhood.

Findings from the plot - This Plot stats that SalePrice are higher near to some areas like StoneBr, NridgHt. And sold price is lower in oldtown and so.

```
plt.figure(figsize=(20, 10))
plt.xticks(rotation=65)

# I have used the jitter which helps if there are many points are overlapping to each othe
# And linewidth that helps me to control the widhth of the grey line.
sns.stripplot(x="Neighborhood", y="SalePrice", data=df_amshscsv,jitter=True,linewidth=1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2742496d0>
```



Conclusion

From this analysis a number of variables are explored from the massive amount of data provided. And I can say that some variables are standard and have more effect on the sale price of the property. Many sized variables are there but mainly overall_qual (Rating of the overall property), Gr_liv_area and Total_bsmt_Sf(Total basement area in square feet) are positively correlated to the sale price and have a higher impact on the selling price. The location and types of road access to the property has a vital role in the selling price of the property. Specific amount of the area has the highest selling price.

✓ 2s completed at 10:11 PM