Ames Housing Data Analysis: Data Cleaning & Exploratory Data Analysis

PROBLEM STATEMENT

This coursework aims to utilize Google Colab to generate a Jupyter notebook for analysing the Ames Housing dataset. The dataset describes the sale of individual residential property from 2006 to 2010 in Ames (a small town in Iowa, USA).

In this work, I'll be reading the data, performing a number pre-processing required on the data, and utilise the Python Data Science Software Ecosystem to undertake Exploratory Data Analysis of the House prices detailing and appropriately documenting any information I'm able to extract from the data.

SUMMARY

During the course of this work, I'll be importing a number of python libraries for different purposes. Pandas acts as a wrapper over these libraries, allowing us to access many of matplotlib's and NumPy's methods with less code. Libraries that will be used are numpy, pandas, matplotlib and seaborn

DATA FORMATING

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

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

Double-click (or enter) to edit

```
# Importing necessary libraries and tools required for the cleaning, analysis and visualizati
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Reading the text file with pandas and printing out its first five rows
# The dataset is given a variable name 'ames'
```

```
ames = pd.read_csv("/content/gdrive/My Drive/AmesHousing.txt", sep='\t', comment='#', na_valu
ames
```

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

2930 rows × 82 columns

The imported Ames Housing dataset has 2930 rows and 82 columns

DATA CLEANING AND EXPLORATION

```
ames.head()
```

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour

Pandas .head() function prints the first five columns in the dataset

```
ames.tail()
```

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Lan Contou
2925	2926	923275080	80	RL	37.0	7937	Pave	NaN	IR1	Lv
2926	2927	923276100	20	RL	NaN	8885	Pave	NaN	IR1	Lov
2927	2928	923400125	85	RL	62.0	10441	Pave	NaN	Reg	Lv
2928	2929	924100070	20	RL	77.0	10010	Pave	NaN	Reg	Lv
2929	2930	924151050	60	RL	74.0	9627	Pave	NaN	Reg	Lv

5 rows × 82 columns

Pandas .tail() prints the last five rows of the dataset

```
ames.describe()
```

			MS	Lot		Overall

Pandas function 'describe' does the statistical evaluation of the numeric columns in the dataset

count	2930.00000	2.930000e+03	2930.000000	2440.000000	2930.000000	2930.000000	29

```
ames.info()
```

```
 27  Mas Vnr Area     2907 non-null   float64
 28  Exter Qual       2930 non-null   object
 29  Exter Cond       2930 non-null   object
 30  Foundation       2930 non-null   object
 31  Bsmt Qual        2850 non-null   object
 32  Bsmt Cond        2850 non-null   object
 33  Bsmt Exposure    2847 non-null   object
 34  BsmtFin Type 1   2850 non-null   object
 35  BsmtFin SF 1     2929 non-null   float64
 36  BsmtFin Type 2   2849 non-null   object
 37  BsmtFin SF 2     2929 non-null   float64
 38  Bsmt Unf SF      2929 non-null   float64
 39  Total Bsmt SF    2929 non-null   float64
 40  Heating          2930 non-null   object
 41  Heating QC       2930 non-null   object
 42  Central Air      2930 non-null   object

 43  Electrical       2929 non-null   object
 44  1st Flr SF       2930 non-null   int64
 45  2nd Flr SF       2930 non-null   int64
 46  Low Qual Fin SF  2930 non-null   int64
 47  Gr Liv Area      2930 non-null   int64
 48  Bsmt Full Bath   2928 non-null   float64
 49  Bsmt Half Bath   2928 non-null   float64
 50  Full Bath        2930 non-null   int64
 51  Half Bath        2930 non-null   int64
 52  Bedroom AbvGr    2930 non-null   int64
 53  Kitchen AbvGr    2930 non-null   int64
 54  Kitchen Qual     2930 non-null   object
 55  TotRms AbvGrd    2930 non-null   int64
 56  Functional       2930 non-null   object
 57  Fireplaces       2930 non-null   int64
 58  Fireplace Qu     1508 non-null   object
 59  Garage Type      2773 non-null   object
 60  Garage Yr Blt    2771 non-null   float64
 61  Garage Finish    2771 non-null   object
 62  Garage Cars      2929 non-null   float64
 63  Garage Area      2929 non-null   float64
 64  Garage Qual      2771 non-null   object
 65  Garage Cond      2771 non-null   object
 66  Paved Drive      2930 non-null   object
 67  Wood Deck SF     2930 non-null   int64
 68  Open Porch SF    2930 non-null   int64
 69  Enclosed Porch   2930 non-null   int64
 70  3Ssn Porch       2930 non-null   int64
 71  Screen Porch     2930 non-null   int64
 72  Pool Area        2930 non-null   int64
 73  Pool QC          13 non-null     object
 74  Fence            572 non-null    object
```

```
 75  Misc Feature     106 non-null    object
 76  Misc Val         2930 non-null   int64
 77  Mo Sold          2930 non-null   int64
 78  Yr Sold          2930 non-null   int64
 79  Sale Type        2930 non-null   object
 80  Sale Condition   2930 non-null   object
 81  SalePrice        2930 non-null   int64
dtypes: float64(11), int64(28), object(43)
memory usage: 1.8+ MB
```

Pandas function 'info' gives information about each columns in the dataset. The number of non-null values and the data type of each column. Also states the number of rows and columns in the dataset. The dataset has 2930 rows and 81 columns. The dataset has three data types; int64, float64 and object.

```
ames.describe(include=['object', 'object'])
```

	MS Zoning	Street	Alley	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	C
count	2930	2930	198	2930	2930	2930	2930	2930	2930	
unique	7	2	2	4	4	3	5	3	28	
top	RL	Pave	Grvl	Reg	Lvl	AllPub	Inside	Gtl	NAmes	
freq	2273	2918	120	1859	2633	2927	2140	2789	443	

4 rows × 43 columns

Describing the categorical data in the dataset

```
# checking the dataset for null values
ames.isna().sum()
```

```
Order              0
PID                0
MS SubClass        0
MS Zoning          0
Lot Frontage     490
                ...
Mo Sold            0
Yr Sold            0
Sale Type          0
Sale Condition     0
SalePrice          0
Length: 82, dtype: int64
```

The dataset has a number of null values which must be treated by dropping the columns if the null values are prominent or filling them if the numbers are lesser.

```
# Dropping the redudant, irrelevant and columns with too many null values
ames = ames.drop(['Pool QC', 'Fence', 'Misc Feature', 'Alley', 'Lot Frontage', 'PID'], axis=1
ames.head()
```

	Order	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	..
0	1	20	RL	31770	Pave	IR1	Lvl	AllPub	Corner	Gtl	..
1	2	20	RH	11622	Pave	Reg	Lvl	AllPub	Inside	Gtl	..
2	3	20	RL	14267	Pave	IR1	Lvl	AllPub	Corner	Gtl	..
3	4	20	RL	11160	Pave	Reg	Lvl	AllPub	Corner	Gtl	..
4	5	60	RL	13830	Pave	IR1	Lvl	AllPub	Inside	Gtl	..

5 rows × 76 columns

'Pool QC', 'Fence', 'Misc Feature', 'Alley', 'Lot Frontage', 'PID' have too many null values and are therefore removed from the dataset

Double-click (or enter) to edit

```
# checking for shape, dimension and size of the dataset
ames.shape, ames.ndim, ames.size
```

```
((2930, 76), 2, 222680)
```

shape of dataset is (2930, 76)

dimension of dataset is

size of dataset is 222680

Double-click (or enter) to edit

```
# checking for total null values in the dataset
```

```
ames.isna().sum().sum()
```

```
2676
```

After dropping redudant and irrelevant columns, we have left with 2676 null values remianing in the dataset

```
# checking for the percentage of missing value in the dataset
rows, columns = ames.shape
cell_count = rows * columns
number_of_nulls = ames.isnull().sum().sum()
percentage_of_missing = (number_of_nulls / cell_count) * 100
print(f'Percentage of missing values: {percentage_of_missing}%')
```

```
Percentage of missing values: 1.2017244476378661%
```

```
# checking for total number of duplicate values
ames.duplicated().sum()
```

```
0
```

There are no duplicate values in the dataset

Double-click (or enter) to edit

```
# putting all categorical data in the dataset in a separate variable
ames_obj = ames.select_dtypes(include=['object'])
ames_obj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 39 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   MS Zoning       2930 non-null   object
 1   Street          2930 non-null   object
 2   Lot Shape       2930 non-null   object
 3   Land Contour    2930 non-null   object
 4   Utilities       2930 non-null   object
 5   Lot Config      2930 non-null   object
 6   Land Slope      2930 non-null   object
 7   Neighborhood    2930 non-null   object
 8   Condition 1     2930 non-null   object
 9   Condition 2     2930 non-null   object
 10  Bldg Type       2930 non-null   object
 11  House Style     2930 non-null   object
 12  Roof Style      2930 non-null   object
 13  Roof Matl       2930 non-null   object
```

```
 14  Exterior 1st    2930 non-null   object
 15  Exterior 2nd    2930 non-null   object
 16  Mas Vnr Type    2907 non-null   object
 17  Exter Qual      2930 non-null   object
 18  Exter Cond      2930 non-null   object
 19  Foundation      2930 non-null   object
 20  Bsmt Qual       2850 non-null   object
 21  Bsmt Cond       2850 non-null   object
 22  Bsmt Exposure   2847 non-null   object
 23  BsmtFin Type 1  2850 non-null   object
 24  BsmtFin Type 2  2849 non-null   object
 25  Heating         2930 non-null   object
 26  Heating QC      2930 non-null   object
 27  Central Air     2930 non-null   object
 28  Electrical      2929 non-null   object
 29  Kitchen Qual    2930 non-null   object
 30  Functional      2930 non-null   object
 31  Fireplace Qu    1508 non-null   object
 32  Garage Type     2773 non-null   object
 33  Garage Finish   2771 non-null   object
 34  Garage Qual     2771 non-null   object
 35  Garage Cond     2771 non-null   object
 36  Paved Drive     2930 non-null   object
 37  Sale Type       2930 non-null   object
 38  Sale Condition  2930 non-null   object
dtypes: object(39)
memory usage: 892.9+ KB
```

This cell aims to put all object type data in a separate dataframe in order to make pre-processing on them easy

```
# putting all columns of the categorical data inside a seperate list
cols = ['MS Zoning','Street','Lot Shape','Land Contour','Utilities','Lot Config','Land Slope'


# forward-filling all null values of the categorical data
from pandas.core.ops import methods
ames[cols] = ames[cols].fillna(method='ffill')
```

All null data in the categorical data are forward filled to retain an accurate data ready for analysis

Double-click (or enter) to edit

```
# putting all quantitative data in the dataset in a separate variable
ames_num = ames.select_dtypes(exclude=['object'])
ames_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 37 columns):
```

```
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Order            2930 non-null   int64  
 1   MS SubClass      2930 non-null   int64  
 2   Lot Area         2930 non-null   int64  
 3   Overall Qual     2930 non-null   int64  
 4   Overall Cond     2930 non-null   int64  
 5   Year Built       2930 non-null   int64  
 6   Year Remod/Add   2930 non-null   int64  
 7   Mas Vnr Area     2907 non-null   float64
 8   BsmtFin SF 1     2929 non-null   float64
 9   BsmtFin SF 2     2929 non-null   float64
 10  Bsmt Unf SF      2929 non-null   float64
 11  Total Bsmt SF    2929 non-null   float64
 12  1st Flr SF       2930 non-null   int64  
 13  2nd Flr SF       2930 non-null   int64  
 14  Low Qual Fin SF  2930 non-null   int64  
 15  Gr Liv Area      2930 non-null   int64  
 16  Bsmt Full Bath   2928 non-null   float64
 17  Bsmt Half Bath   2928 non-null   float64
 18  Full Bath        2930 non-null   int64  
 19  Half Bath        2930 non-null   int64  
 20  Bedroom AbvGr    2930 non-null   int64  
 21  Kitchen AbvGr    2930 non-null   int64  
 22  TotRms AbvGrd    2930 non-null   int64  
 23  Fireplaces       2930 non-null   int64  
 24  Garage Yr Blt    2771 non-null   float64
 25  Garage Cars      2929 non-null   float64
 26  Garage Area      2929 non-null   float64
 27  Wood Deck SF     2930 non-null   int64  
 28  Open Porch SF    2930 non-null   int64  
 29  Enclosed Porch   2930 non-null   int64  
 30  3Ssn Porch       2930 non-null   int64  
 31  Screen Porch     2930 non-null   int64  
 32  Pool Area        2930 non-null   int64  
 33  Misc Val         2930 non-null   int64  
 34  Mo Sold          2930 non-null   int64  
 35  Yr Sold          2930 non-null   int64  
 36  SalePrice        2930 non-null   int64  
dtypes: float64(10), int64(27)
memory usage: 847.1 KB
```

This cell aims to put all quantitative data in a seeparate dataframe in order to make pre-processing on them easy

```
# putting all columns of the quantitative data inside a seperate list
cols1 = ['Order','MS SubClass','Lot Area','Overall Qual','Overall Cond','Year Built','Year Re


# filling the quantitative data in the dataset with zero
ames[cols1] = ames[cols1].fillna(0)
```

All null data in the quantitative data are filled with 0 to retain an accurate data ready for analysis

```
# re-checking the dataset to be sure its free of null values
ames.info()
```

```
 21  Exterior 1st     2930 non-null   object
 22  Exterior 2nd     2930 non-null   object
 23  Mas Vnr Type     2930 non-null   object
 24  Mas Vnr Area     2930 non-null   float64
 25  Exter Qual       2930 non-null   object
 26  Exter Cond       2930 non-null   object
 27  Foundation       2930 non-null   object
 28  Bsmt Qual        2930 non-null   object
 29  Bsmt Cond        2930 non-null   object
 30  Bsmt Exposure    2930 non-null   object
 31  BsmtFin Type 1   2930 non-null   object
 32  BsmtFin SF 1     2930 non-null   float64
 33  BsmtFin Type 2   2930 non-null   object
 34  BsmtFin SF 2     2930 non-null   float64
 35  Bsmt Unf SF      2930 non-null   float64
 36  Total Bsmt SF    2930 non-null   float64
 37  Heating          2930 non-null   object
 38  Heating QC       2930 non-null   object
 39  Central Air      2930 non-null   object
 40  Electrical       2930 non-null   object
 41  1st Flr SF       2930 non-null   int64
 42  2nd Flr SF       2930 non-null   int64
 43  Low Qual Fin SF  2930 non-null   int64
 44  Gr Liv Area      2930 non-null   int64

 45  Bsmt Full Bath   2930 non-null   float64
 46  Bsmt Half Bath   2930 non-null   float64
 47  Full Bath        2930 non-null   int64
 48  Half Bath        2930 non-null   int64
 49  Bedroom AbvGr    2930 non-null   int64
 50  Kitchen AbvGr    2930 non-null   int64
 51  Kitchen Qual     2930 non-null   object
 52  TotRms AbvGrd    2930 non-null   int64
 53  Functional       2930 non-null   object
 54  Fireplaces       2930 non-null   int64
 55  Fireplace Qu     2930 non-null   object
 56  Garage Type      2930 non-null   object
 57  Garage Yr Blt    2930 non-null   float64
 58  Garage Finish    2930 non-null   object
 59  Garage Cars      2930 non-null   float64
 60  Garage Area      2930 non-null   float64
 61  Garage Qual      2930 non-null   object
 62  Garage Cond      2930 non-null   object
 63  Paved Drive      2930 non-null   object
 64  Wood Deck SF     2930 non-null   int64
 65  Open Porch SF    2930 non-null   int64
 66  Enclosed Porch   2930 non-null   int64
 67  3Ssn Porch       2930 non-null   int64
 68  Screen Porch     2930 non-null   int64
 69  Pool Area        2930 non-null   int64
 70  Misc Val         2930 non-null   int64
```

```
 71  Mo Sold          2930 non-null   int64
 72  Yr Sold          2930 non-null   int64
 73  Sale Type        2930 non-null   object
 74  Sale Condition   2930 non-null   object
 75  SalePrice        2930 non-null   int64
dtypes: float64(10), int64(27), object(39)
memory usage: 1.7+ MB
```

After filling the categorical and quantitative data in the dataset, the data set is free of null values and ready for analysis and visualization

```
ames.isna().sum().sum()
```

```
0
```

```
# checking the five values in the "Gr Liv Area" columns that are greater than 4000
trial =[]
for x in ames['Gr Liv Area']:
    if x >= 4000:
      trial.append(x)
print(trial)
```

```
[5642, 4476, 4316, 5095, 4676]
```

In the documentation of the dataset, it was stated that there are five outliers, which can be visualized in the plot of SALE PRICE versus GR LIV AREA. Three of them are true outliers (Partial Sales that likely don't represent actual market values) and two of them are simply unusual sales (very large houses priced relatively appropriately). It was recommended to remove all houses with more than 4000 square feet from the data set (which eliminates these five unusual observations). These houses have above ground living area of; 5642, 4476, 4316, 5095, 4676 (square feet)

```
ames[["SalePrice", "Gr Liv Area"]].plot(x="Gr Liv Area", y="SalePrice", kind="scatter")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc57cfb6d0>
```

A plot of SALE PRICE versus GR LIV AREA revealing the outliers in the dataset

```
ames = ames.where(ames['Gr Liv Area']<4000)
```

highlighting the values that are not greater than 4000 in the "Gr Liv Area" column

```
ames = ames.dropna(subset=ames.columns.values)
```

The values that are greater than 4000 in the said column, returned as nan and I hereby drop the null values in the dataset again. Only this time, it is nan of the values that are greater than 400

```
ames.isnull().sum().sum()
```

```
0
```

Confirming the dataset is free of null values

EXPLORATORY DATA ANALYSIS & VISUALIZATION

```
# performing correlation on the quantitative data in the dataset
corr = ames.corr()
corr
```

	Order	MS SubClass	Lot Area	Overall Qual	Overall Cond	Year Built	Year Remod/Add	Mas
Order	1.000000	0.012022	0.029228	-0.051160	-0.010747	-0.053282	-0.076466	-0.03
MS SubClass	0.012022	1.000000	-0.206542	0.040287	-0.067500	0.036899	0.043722	0.00
Lot Area	0.029228	-0.206542	1.000000	0.083256	-0.032795	0.017049	0.015887	0.10
Overall Qual	-0.051160	0.040287	0.083256	1.000000	-0.093902	0.596621	0.569498	0.41
Overall Cond	-0.010747	-0.067500	-0.032795	-0.093902	1.000000	-0.368533	0.048356	-0.13
Year Built	-0.053282	0.036899	0.017049	0.596621	-0.368533	1.000000	0.611444	0.30
Year Remod/Add	-0.076466	0.043722	0.015887	0.569498	0.048356	0.611444	1.000000	0.18
Mas Vnr Area	-0.035297	0.005342	0.104548	0.414981	-0.133399	0.305505	0.188842	1.00
BsmtFin SF 1	-0.038057	-0.060244	0.156171	0.269218	-0.048921	0.280189	0.147915	0.27
BsmtFin SF 2	-0.002514	-0.070926	0.086000	-0.039997	0.040839	-0.026836	-0.061531	-0.01
Bsmt Unf SF	0.005244	-0.129738	0.021880	0.270469	-0.137346	0.128720	0.164875	0.08
Total Bsmt SF	-0.035091	-0.226430	0.219666	0.545865	-0.177712	0.414686	0.301000	0.36
1st Flr SF	-0.019245	-0.255465	0.305667	0.470625	-0.159138	0.311505	0.241169	0.36
2nd Flr SF	-0.001958	0.305879	0.024749	0.234033	0.007085	0.013487	0.157391	0.11
Low Qual Fin SF	0.013677	0.025747	0.001399	-0.048521	0.009120	-0.144242	-0.060256	-0.05
Gr Liv Area	-0.015315	0.072477	0.258529	0.564797	-0.116169	0.239305	0.318860	0.37
Bsmt Full Bath	-0.043307	0.014456	0.116413	0.164692	-0.042370	0.210575	0.132905	0.13
Bsmt Half Bath	0.023643	-0.002107	0.021057	-0.048538	0.084651	-0.033008	-0.048443	0.00
Full Bath	-0.046784	0.135201	0.121394	0.518713	-0.214322	0.468485	0.456708	0.24
Half Bath	-0.040843	0.176399	0.028971	0.265096	-0.087569	0.267691	0.210293	0.18
Bedroom AbvGr	0.015351	-0.019601	0.138169	0.061772	-0.006207	-0.055650	-0.021801	0.08
Kitchen AbvGr	-0.017518	0.257698	-0.019392	-0.159819	-0.086523	-0.137621	-0.142199	-0.05

TotRms AbvGrd	-0.000399	0.033655	0.200794	0.371467	-0.088614	0.107042	0.194164	0.26
Fireplaces	-0.020639	-0.049735	0.247698	0.387174	-0.030648	0.167803	0.130699	0.26
Garage Yr Blt	-0.022795	-0.103139	0.072319	0.279026	0.015169	0.260000	0.146457	0.12
Garage Cars	-0.038284	-0.045786	0.176047	0.598127	-0.181384	0.537146	0.423511	0.35
Garage Area	-0.038270	-0.103317	0.198746	0.558836	-0.153265	0.479555	0.373977	0.36
Wood Deck SF	-0.013002	-0.016396	0.150529	0.250281	0.021162	0.226964	0.216077	0.15
Open Porch SF	0.013030	-0.013275	0.082372	0.289399	-0.067171	0.194889	0.239175	0.11
Enclosed Porch	0.028219	-0.022949	0.024212	-0.139566	0.071281	-0.374127	-0.220023	-0.10

Correlation is a statistical measure that expresses the extent to which columns of a dataframe are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect. Highlight of columns that have 0.3 correlation and above with Sale price are: Overall qual, Year Built, Year Remod/Add, MAs Vnr Area, Total Bsmt SF, 1st Flr SF, Gr Liv Area, Full Bath, Garage Yr Blt, and Garage Cars

Mo Sold	0.133082	0.000815	0.005732	0.032102	-0.006852	0.014688	0.018020	-0.00

```
# heatmap visualizes the correlation between each columns
fig, ax = plt.subplots(figsize=(10, 5))
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc69ccb610>
```

The primary purpose of Heat Maps is to better visualize the volume of locations/events within a dataset and assist in directing us towards areas on data visualizations that matter most. We will be specifically looking out for lighter colour visuals having correlation of 0.3 upwards.


```
# printing the sale price column
ames['SalePrice']
```

```
0       215000.0
1       105000.0
2       172000.0
3       244000.0
4       189900.0
          ...
2925    142500.0
2926    131000.0
2927    132000.0
2928    170000.0
2929    188000.0
Name: SalePrice, Length: 2925, dtype: float64
```

```
# checking the highest price a house was sold/bought for
ames['SalePrice'].max()
```

```
625000.0
```

Highest price a house is sold for is 625000

```
# checking the lowest price a house was sold/bought for
ames['SalePrice'].min()
```

```
12789.0
```

Lowest price a house is sold for is 12789

```
# checking the complete details of the house with the highest price
ames[ames['SalePrice'] == ames['SalePrice'].max()]
```

	Order	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Lan Slop
2445	2446.0	60.0	RL	35760.0	Pave	IR1	Lvl	AllPub	CulDSac	C

The row with the highest sale price is on index of 2445 and has an order number of 2446

```
# checking the houses sold for 500,000 or more
ames[ames['SalePrice']> 500000]
```

	Order	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Lan Slop
15	16.0	60.0	RL	53504.0	Pave	IR2	HLS	AllPub	CulDSac	Mo
44	45.0	20.0	RL	12919.0	Pave	IR1	Lvl	AllPub	Inside	C
366	367.0	20.0	RL	17423.0	Pave	IR1	Lvl	AllPub	CulDSac	C
423	424.0	20.0	RL	15431.0	Pave	Reg	Lvl	AllPub	Inside	C
432	433.0	20.0	RL	13693.0	Pave	Reg	Lvl	AllPub	Inside	C
433	434.0	60.0	RL	13891.0	Pave	Reg	Lvl	AllPub	Inside	C
456	457.0	20.0	RL	14836.0	Pave	IR1	HLS	AllPub	Inside	Mo
1063	1064.0	20.0	RL	12720.0	Pave	Reg	HLS	AllPub	Inside	Mo
1637	1638.0	20.0	RL	51974.0	Pave	IR1	Lvl	AllPub	CulDSac	C
1701	1702.0	20.0	RL	17169.0	Pave	IR2	Lvl	AllPub	CulDSac	C
2330	2331.0	60.0	RL	18062.0	Pave	IR1	HLS	AllPub	CulDSac	C
2332	2333.0	60.0	RL	16056.0	Pave	IR1	Lvl	AllPub	Inside	C
2334	2335.0	60.0	RL	16052.0	Pave	IR1	Lvl	AllPub	CulDSac	C
2445	2446.0	60.0	RL	35760.0	Pave	IR1	Lvl	AllPub	CulDSac	C
2450	2451.0	60.0	RL	17242.0	Pave	IR1	Lvl	AllPub	Inside	C

15 rows × 76 columns

15 houses have a price of 500000 and above

```
# checking the houses with price less than 100,000
ames[ames['SalePrice']< 100000]
```

	Order	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Lan Slop
29	30.0	160.0	RM	1680.0	Pave	Reg	Lvl	AllPub	Inside	G
31	32.0	160.0	RM	1680.0	Pave	Reg	Lvl	AllPub	Inside	G
77	78.0	120.0	RL	7892.0	Pave	Reg	Lvl	AllPub	Inside	G
82	83.0	30.0	RH	9800.0	Pave	Reg	Lvl	AllPub	Corner	G
125	126.0	90.0	RL	13260.0	Pave	IR1	Lvl	AllPub	Inside	G
...	...	...	...	...	...	...	...	...	...	.
2916	2917.0	180.0	RM	1533.0	Pave	Reg	Lvl	AllPub	Inside	G
2917	2918.0	160.0	RM	1533.0	Pave	Reg	Lvl	AllPub	Inside	G
2918	2919.0	160.0	RM	1526.0	Pave	Reg	Lvl	AllPub	Inside	G
2919	2920.0	160.0	RM	1936.0	Pave	Reg	Lvl	AllPub	Inside	G
2920	2921.0	160.0	RM	1894.0	Pave	Reg	Lvl	AllPub	Inside	G

237 rows × 76 columns

237 houses are on the low pricing, selling for 100000 and below

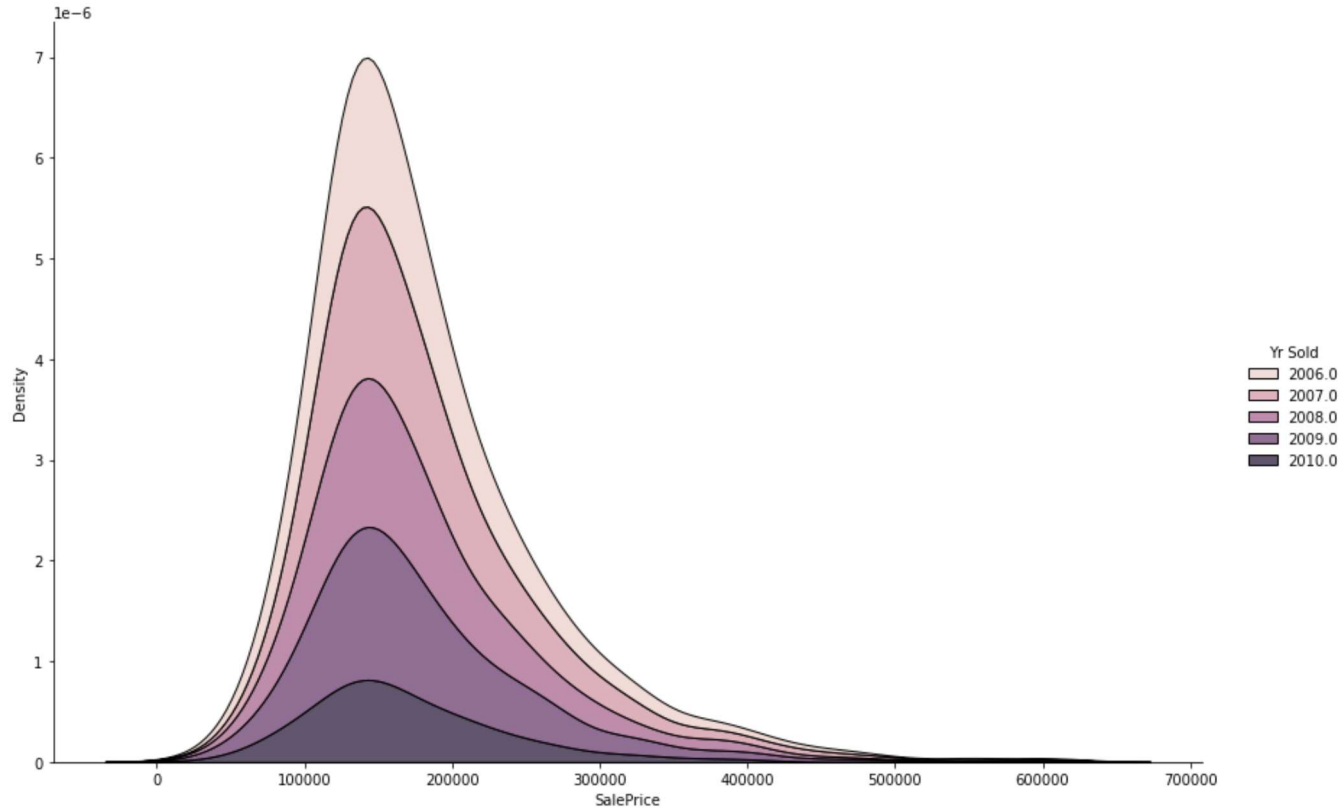
```
ames['Yr Sold'].value_counts()
```

```
2007.0    690
2009.0    648
2006.0    625
2008.0    621
2010.0    341
Name: Yr Sold, dtype: int64
```

checking number of houses sold in each year. more houses were sold in 2007 and half its amount was sold in 2010.

```
sns.displot(ames, x="SalePrice", hue="Yr Sold", kind="kde",multiple="stack", height=8, aspect
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdc5a9eb220>
```



The distribution plot of Year sold against Sale price indicates that more houses were sold between 100,000 and 200,000 from 2006 to 2010 while fewer houses were for 300,000 upwards.

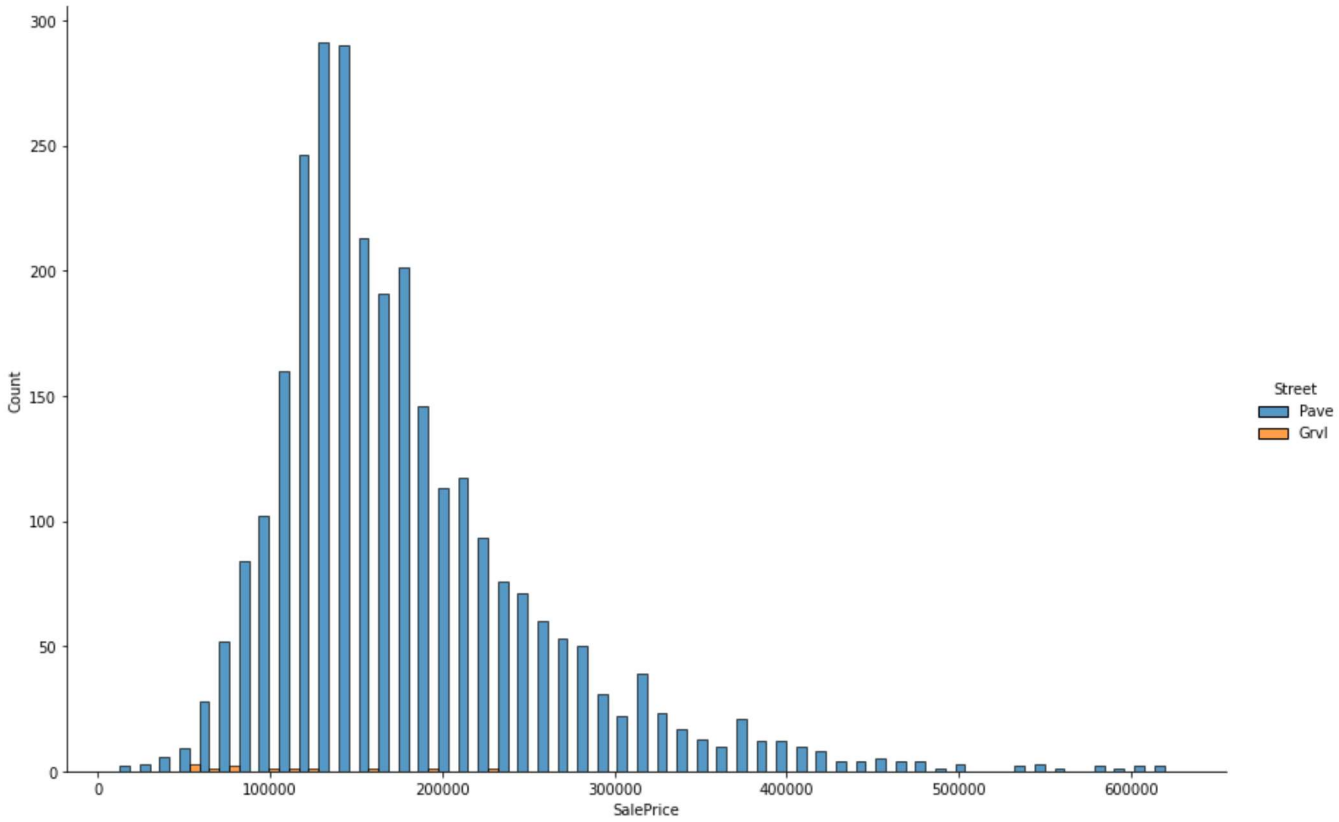
```
# checking the type of road access to property
ames['Street'].value_counts()
```

```
Pave    2913
Grvl      12
Name: Street, dtype: int64
```

Value count of street reveals that there only two types of road access to the houses which are either gravel or paved.

```
# distribution plot of sale price against street acess
sns.displot(ames, x="SalePrice", hue="Street", multiple="dodge", height=8, aspect=1.5)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdc559dea60>
```



It is noted from the distribution plot of sale price agaisnt street acess that only a handful number of houses gravel road access leading to them and the chiefly prominent street acess is paved

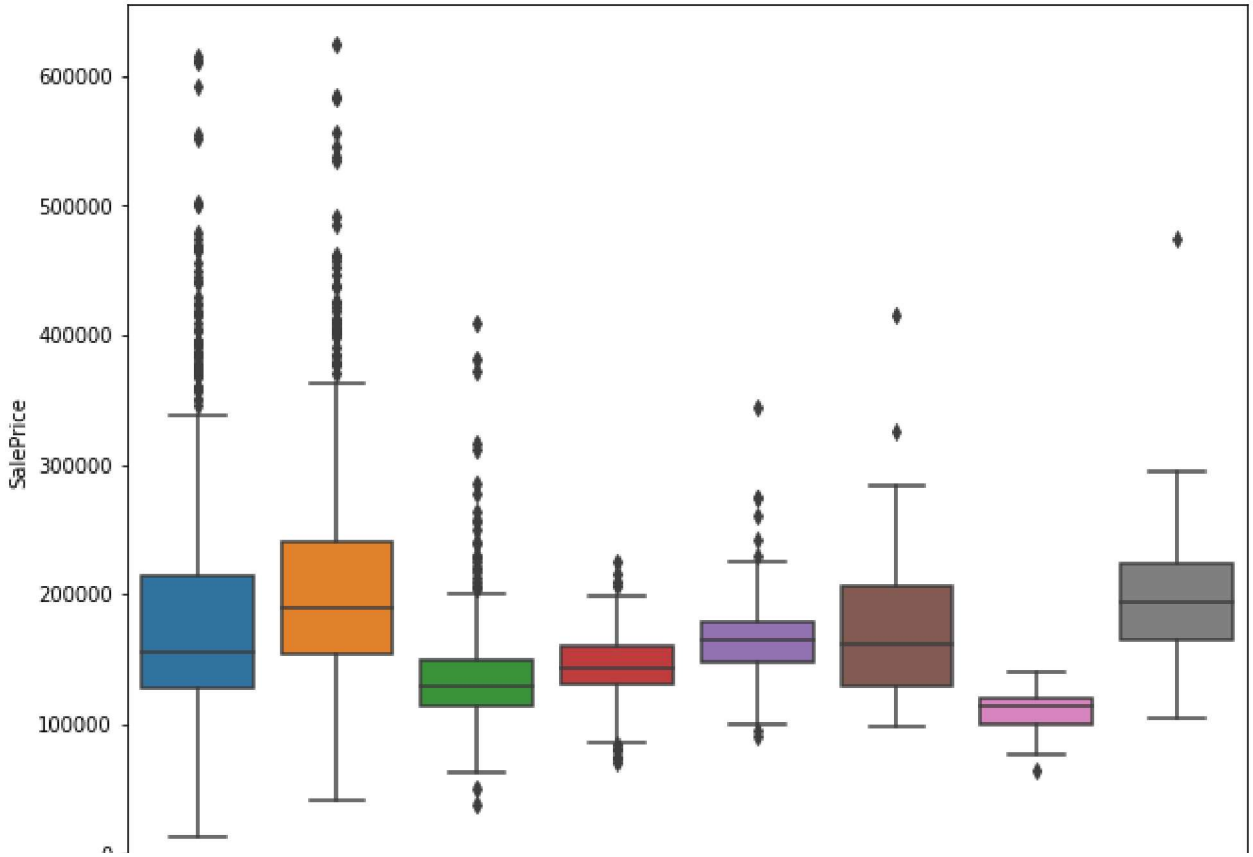
```
# checking the style of dwelling
ames['House Style'].value_counts()
```

```
1Story    1480
2Story     869
1.5Fin     314
SLvl       128
SFoyer      83
2.5Unf      24
1.5Unf      19
2.5Fin       8
Name: House Style, dtype: int64
```

The value count of House style reveals that there 8 different style of dwelling in the dataset.

```
plt.figure(figsize=(10,8))
sns.boxplot(data=ames, x="House Style", y="SalePrice",)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53fbf370>
```



Box plot of House style against Sale price indicates that two storey and one storey building sold for the highest price and the other houses sold for almost the same average prices except One and one-half story: 2nd level unfinished houses which sold for the lowest amount.

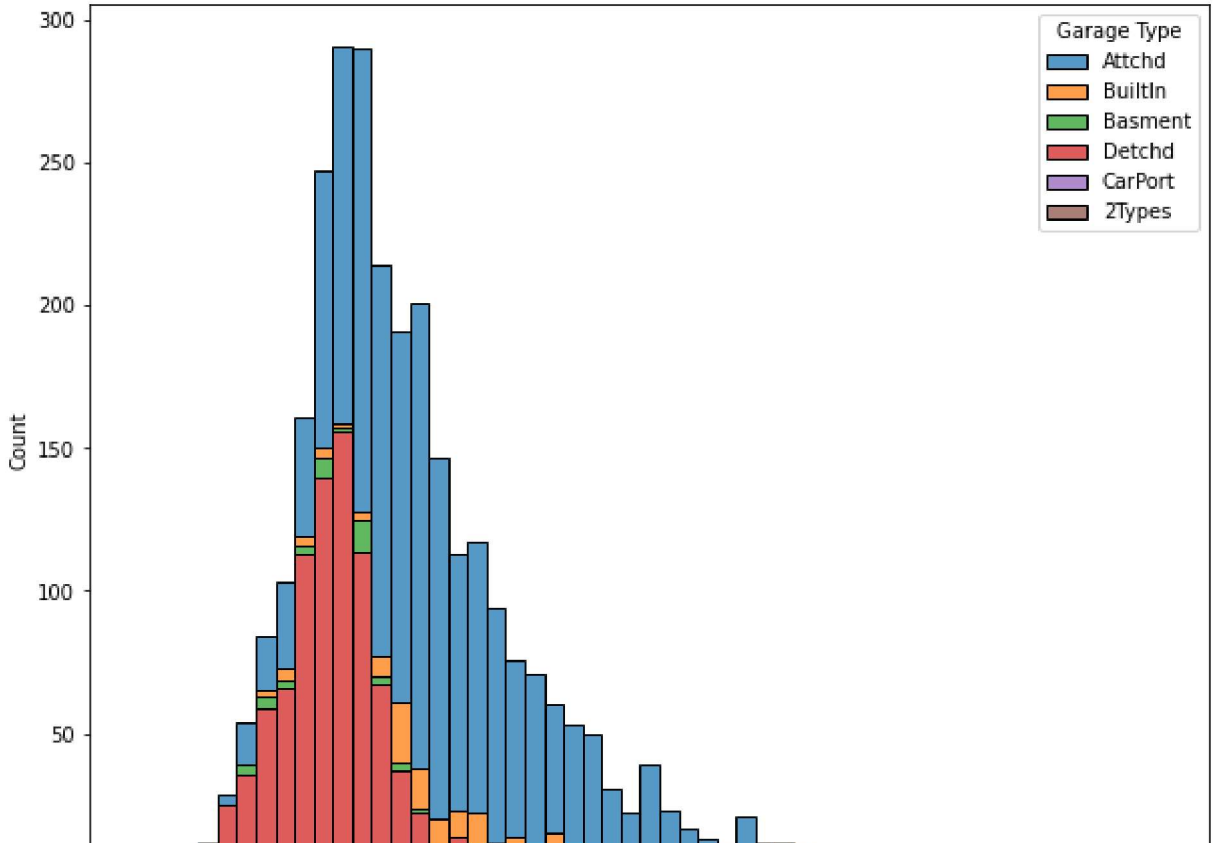
```
# checking the house garage type
ames['Garage Type'].value_counts()
```

```
Attchd     1781
Detchd      864
BuiltIn     188
Basment      46
2Types       24
CarPort      22
Name: Garage Type, dtype: int64
```

The value count of Garage type reveals that there 6 different garage types in the dataset.

```
# histogram plot of sale price and garage type
plt.figure(figsize=(10,8))
sns.histplot(ames, x="SalePrice", hue="Garage Type", multiple="stack")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53f23790>
```



The predominant features on the histogram plot of sale price and garage type are houses that have garage attached and sold for prices between 100,000 and 200,000 and houses that have their garages detached that chiefly sold between 100,000 and 150,000. Also, many built-in garage houses sold for 200,000 and above, although they are not so much in number.

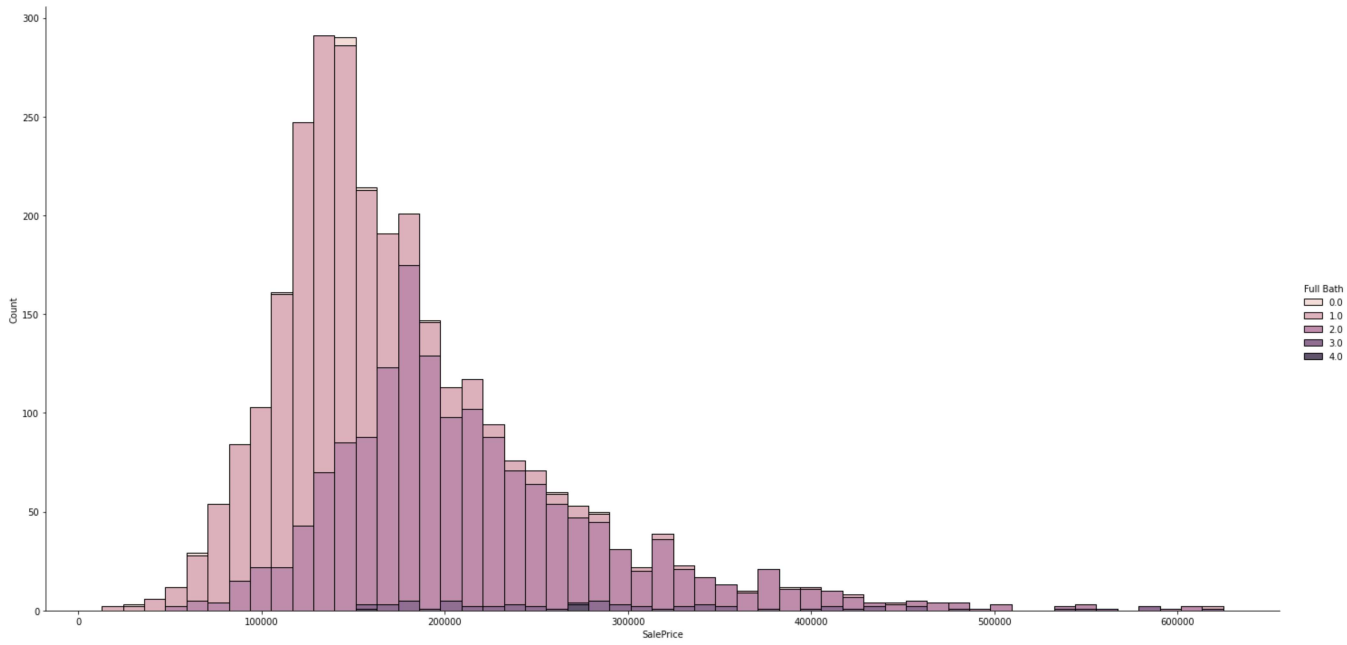
```
ames['Full Bath'].value_counts()
```

```
2.0    1530
1.0    1318
3.0      61
0.0      12
4.0       4
Name: Full Bath, dtype: int64
```

Value count of Full bath reveals that houses in this dataset have 0 to 4 number of toilets.

```
# distribution plot of sale price and number of full bath
plt.figure(figsize=(30,10))
sns.displot(ames, x="SalePrice", hue="Full Bath", multiple="stack", height=10, aspect=2)
```

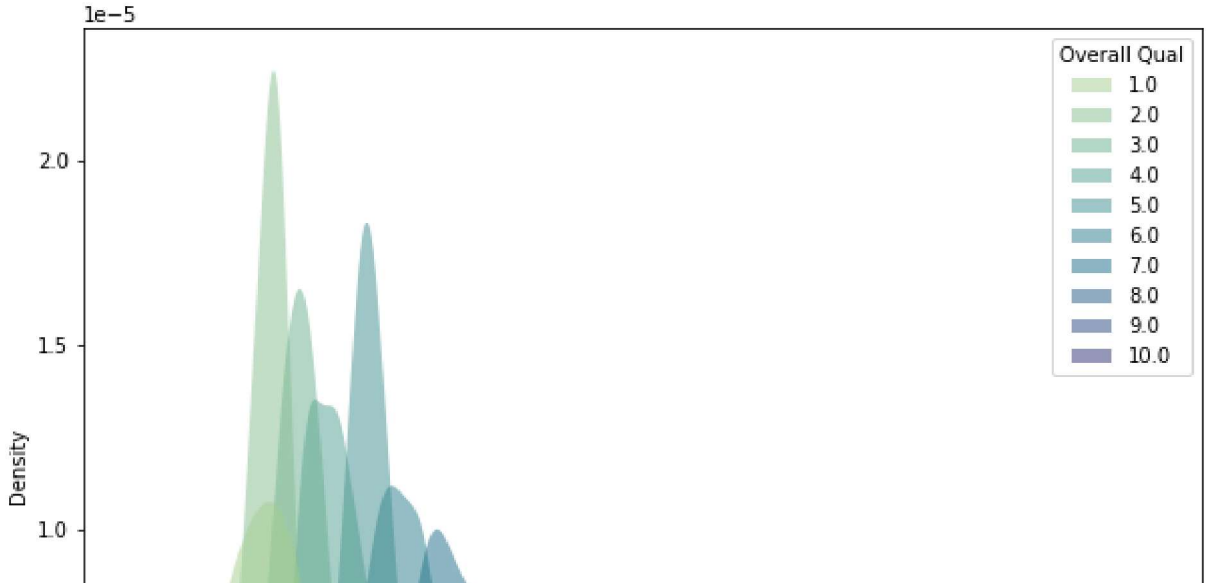
```
<seaborn.axisgrid.FacetGrid at 0x7fdc5aa77d00>
<Figure size 2160x720 with 0 Axes>
```



It is noticeable from the plot of sale price number of full bath that much more houses have 1 or 2 Full bath are sold between 100,000 and 300,000.

```
# Kernel Distribution Estimation Plot of sale price and overall quality
plt.figure(figsize=(10,8))
sns.kdeplot(
   data=ames, x="SalePrice", hue="Overall Qual",
   fill=True, common_norm=False, palette="crest",
   alpha=.5, linewidth=0,
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53815040>
```



It can be deduced from the plot that houses that sold for 100,000 and below had low quality of between 1 and 3 on a scale of 10. Houses that sold for 200,000 and 400,000 have average quality of 4-7 on a scale of 10. Finally, houses that sold for 400,000 and above are very high quality houses with house quality of between 8-10 on a scale of 10.

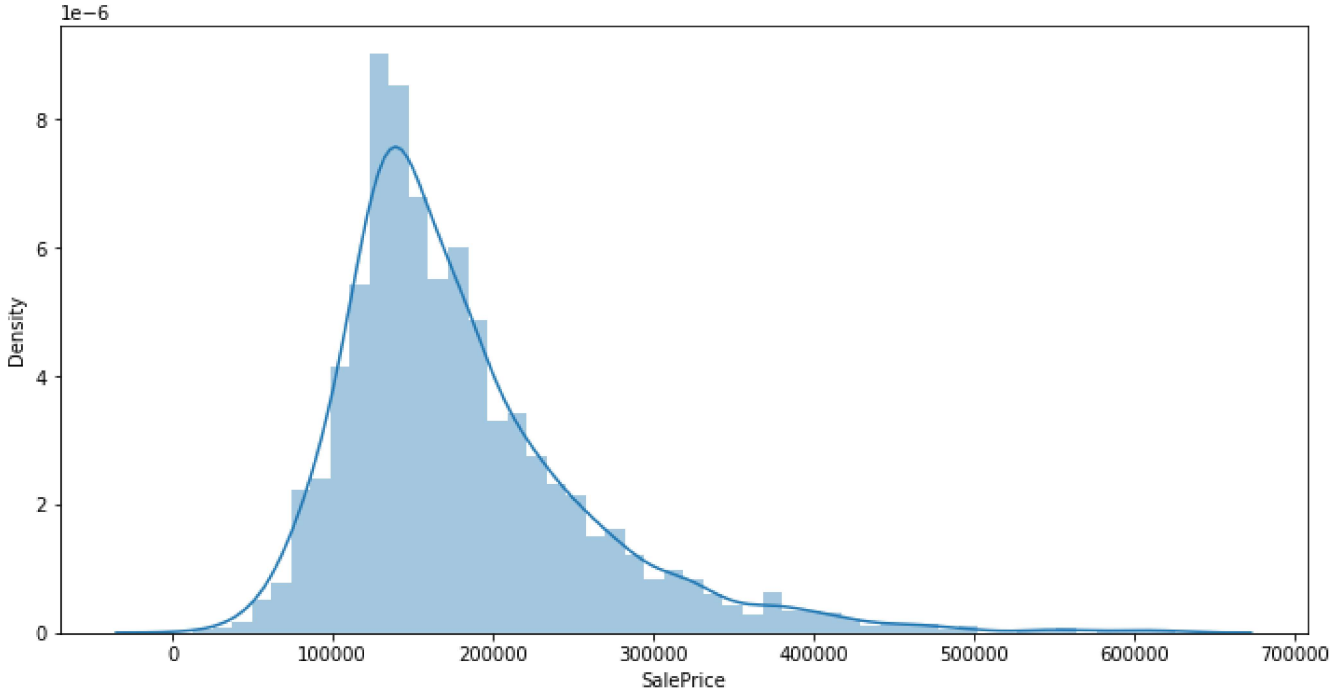
```
# plot of size of garage in car capacity and sale price
plt.figure(figsize=(10,8))
sns.histplot(ames, x="SalePrice", hue="Garage Cars", element="poly",)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53811cd0>
```

It can be noted from the plot of size of garage in car capacity and sale price that houses with no garage and houses with garage size suitable for 1 or 2 cars are major and have prices between 100,000 and 300,000.

```
# distribution plot of sale price
plt.figure(figsize=(12,6))
sns.distplot(ames['SalePrice'])
```

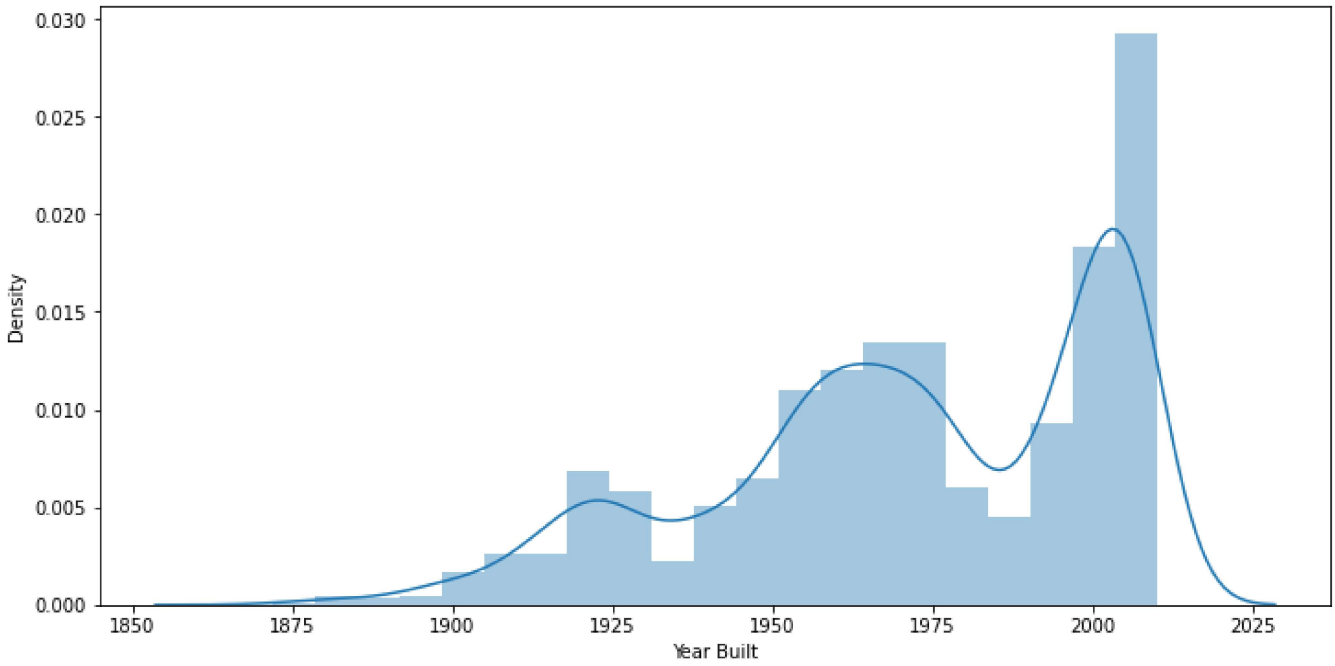
```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53795c70>
```



The distribution plot of sale price reveals that most houses chiefly sold between 100,000 and 200,000.

```
# distribution plot of year built
plt.figure(figsize=(12,6))
sns.distplot(ames['Year Built'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc52161d60>
```



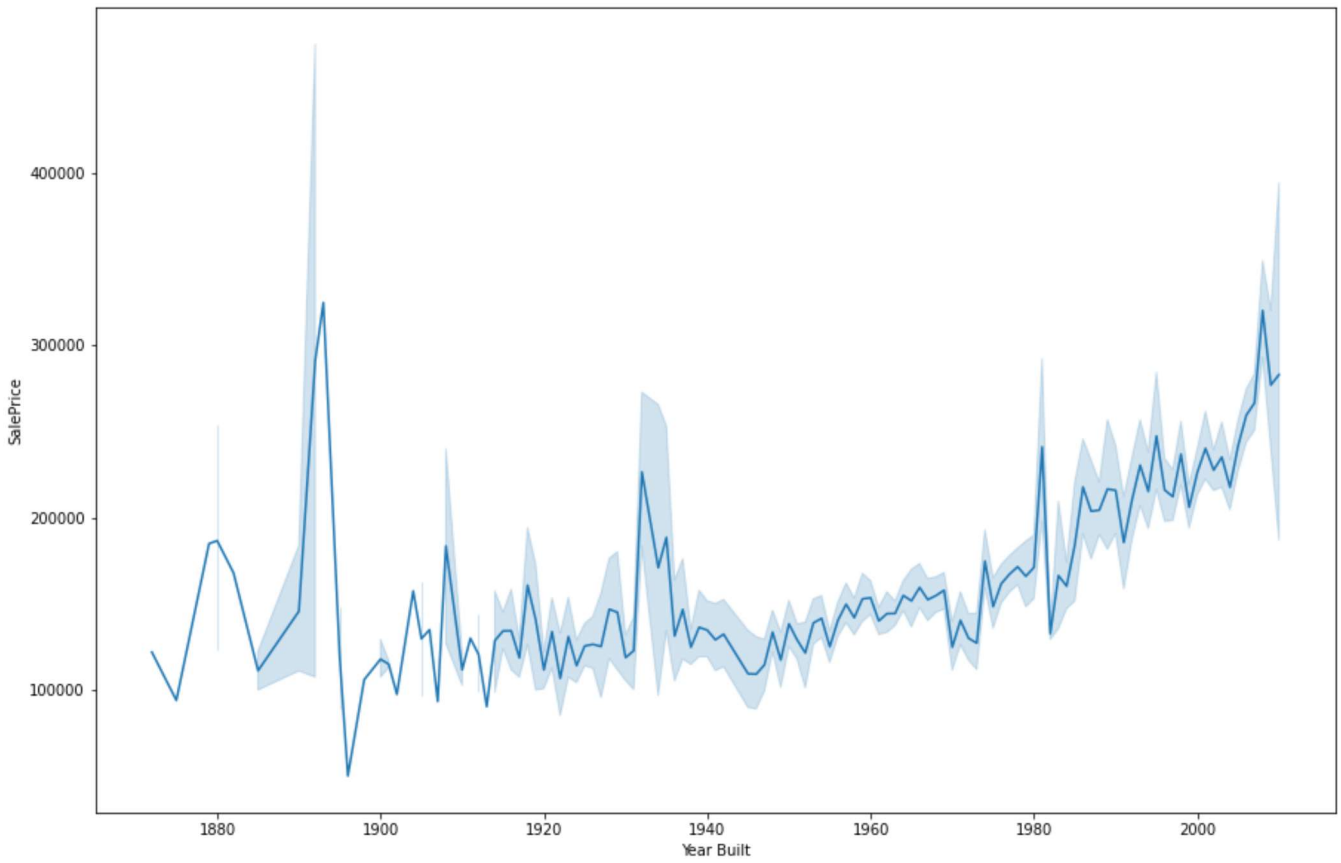
The distribution plot of year built indicates that, although having a few dive in few years within the dataset, houses built increased on a geometric progression between 1850 and 2010, with houses built peaking at a record high in 2010.

```
# grouping the dataset by house construction year
# checking the highest price a house was sold in each building year
ames.groupby('Year Built').SalePrice.max()
```

```
Year Built
1872.0    122000.0
1875.0     94000.0
1879.0    185000.0
1880.0    295000.0
1882.0    168000.0
             ...
2006.0    591587.0
2007.0    610000.0
2008.0    582933.0
2009.0    611657.0
2010.0    394432.0
Name: SalePrice, Length: 118, dtype: float64
```

```
# line plot of sale price against year built
plt.figure(figsize=(15,10))
sns.lineplot(data=ames, x="Year Built", y="SalePrice",)
```

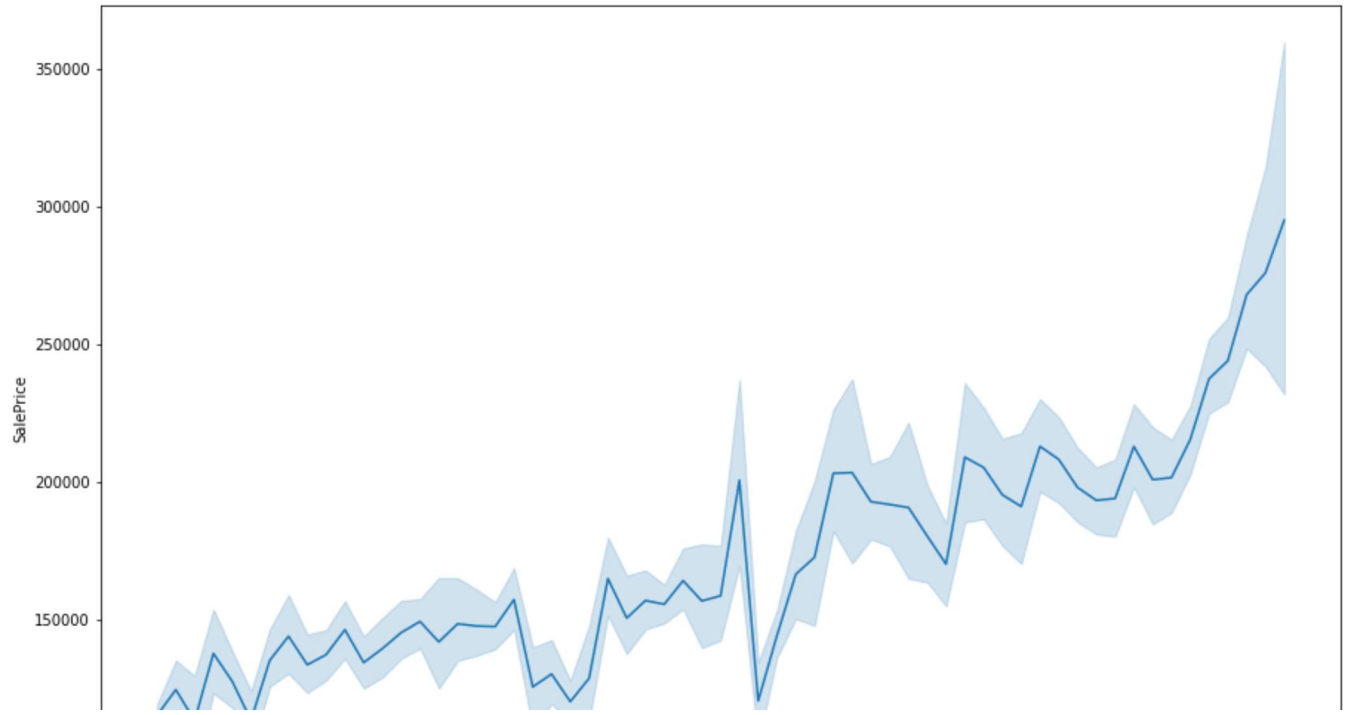
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53014490>
```



Across the years, it is noticed in the plot that houses sold for average pricing between 100,000 and 200,000. Although on four different occassions, houses prices spiked so high within the years.

```
# line plot of year remodel against year
plt.figure(figsize=(15,10))
sns.lineplot(data=ames, x="Year Remod/Add", y="SalePrice",)
```

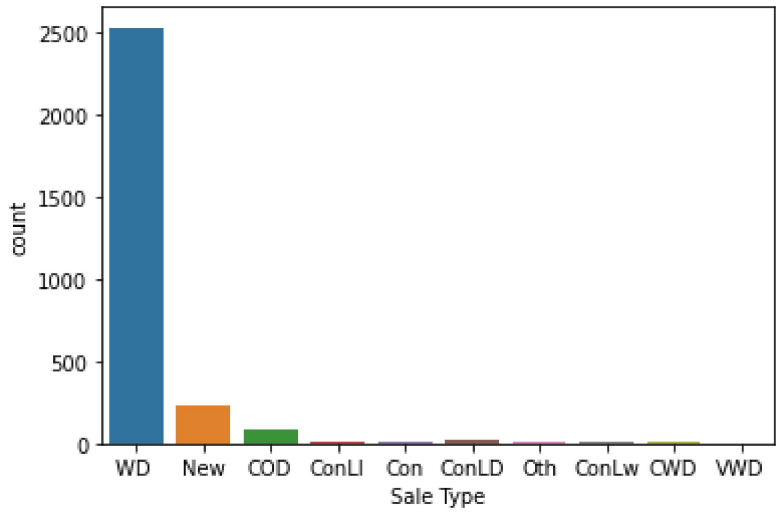
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc53668e20>
```



The most noticeable feature of this plot is that houses that were remodeled from 1985 to 2010 sold higher than houses that were built or remodeled in an earlier year. It is noteworthy to mention that in this column, the construction year is the remodel year for some houses as no remodeling has been done in them since they were built.

```
# countplot of sale type
sns.countplot(x=ames["Sale Type"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc538626d0>
```

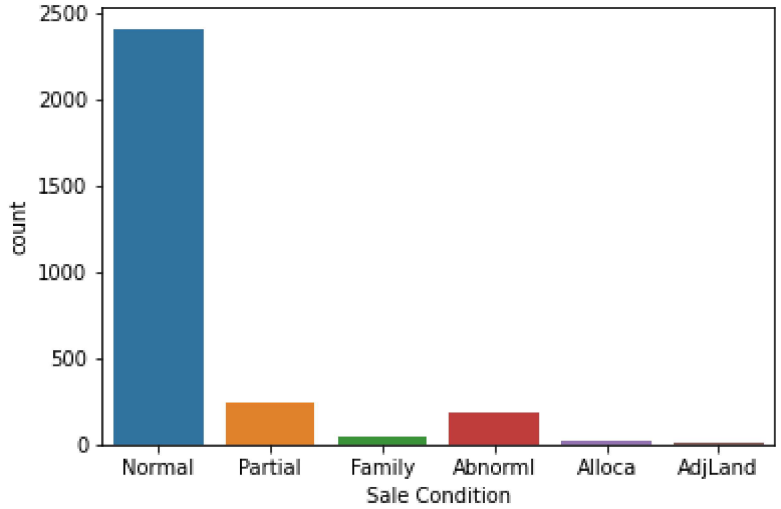


The type of sale that is predominat is the conventional warranty deed sale.

```
# countplot of sale condition
```

```
sns.countplot(x=ames["Sale Condition"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc51a1b610>
```



The main condition of sale is normal sale, as observed from the plot

```
ames["MS Zoning"].value_counts()
```

```
RL         2268
RM          462
FV          139
RH           27
C (all)      25
I (all)       2
A (agr)       2
Name: MS Zoning, dtype: int64
```

The value count of zoning classification of sale suggests that most houses are low density residential while medium density residential is also noticeable.

```
# countplot of zoning classification of sale
sns.countplot(x=ames["MS Zoning"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc518cee50>
```

The plot further visualizes that most houses are low density residences.

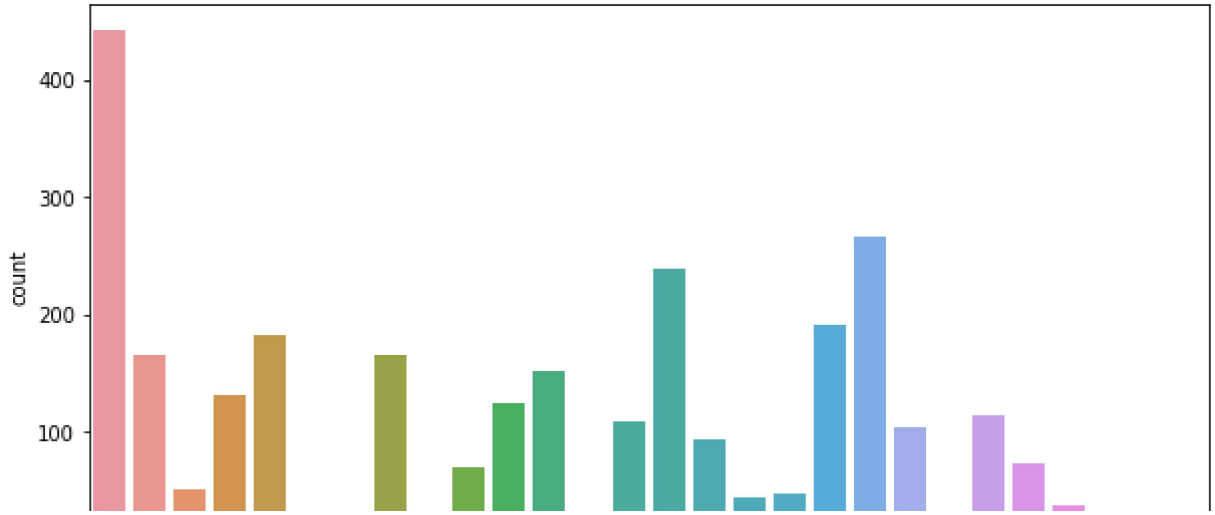
```
ames["Neighborhood"].value_counts()
```

```
NAmes      443
CollgCr    267
OldTown    239
Edwards    191
Somerst    182
NridgHt    166
Gilbert    165
Sawyer     151
NWAmes     131
SawyerW    125
Mitchel    114
BrkSide    108
Crawfor    103
IDOTRR      93
Timber      72
NoRidge     69
StoneBr     51
SWISU       48
ClearCr     44
MeadowV     37
BrDale      30
Blmngtn     28
Veenker     24
NPkVill     23
Blueste     10
Greens       8
GrnHill      2
Landmrk      1
Name: Neighborhood, dtype: int64
```

Value count of Neighborhood reveals the number of houses sold in each neigborhood with the most sold in North Ames.

```
# countplot of number of houses sold in Ames city
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
sns.countplot(x=ames["Neighborhood"])
```

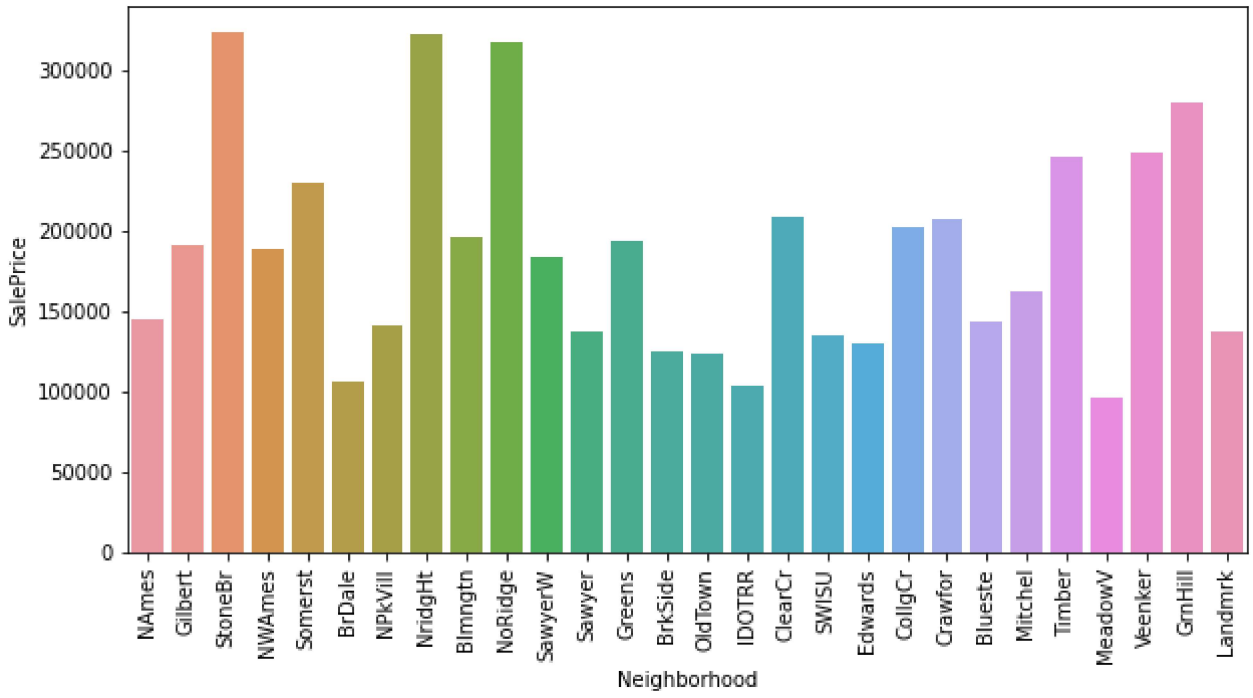
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc57a1a7c0>
```



This visualization further proves that the most houses were sold in North Ames while roughly the same amount of houses were sold in Gilbert, Somerset, Northridge Heights, Sawyer, Old Town, Edwards and College Creek.

```
# bar plot of sale price against neighborhood
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
sns.barplot(data=ames, x="Neighborhood", y="SalePrice", ci=None)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc5175ed60>
```



The plot indicates that the most costly houses were sold in Stone Brook, Northridge Heights, Northridge, Timberland, Venkeer and GreenHill.

CONCLUSION

To conclude I'll group the dataset by overall quality and analyse it against price.

```
# grouping the dataset by house overall quality
# checking the statistical evaluation of the grouped data using the sale price as a reference
ames.groupby('Overall Qual').SalePrice.describe()
```

Overall Qual	count	mean	std	min	25%	50%	75%	
1.0	4.0	48725.000000	29341.935292	13100.0	32750.00	50150.0	66125.00	8
2.0	13.0	52325.307692	17562.955704	12789.0	45000.00	57625.0	60000.00	8
3.0	40.0	83185.975000	23569.801683	37900.0	64875.00	81200.0	95475.00	13
4.0	226.0	106485.097345	29224.943495	34900.0	85500.00	105000.0	124275.00	25
5.0	825.0	134752.516364	27690.596654	55993.0	119600.00	133000.0	147983.00	30
6.0	732.0	162130.318306	37201.296411	76000.0	138000.00	159500.0	182000.00	41
7.0	602.0	205025.760797	43166.268562	82500.0	176421.25	200000.0	229600.00	38
8.0	350.0	270913.594286	61326.207415	122000.0	230000.00	264530.5	305975.00	53
9.0	107.0	368336.766355	79201.265422	150000.0	318030.50	360000.0	405374.50	61

Overall quality being the column that is most correlated with sale price is further discussed here.

It is noted after grouping that houses with quality of 10 sold the highest with a house amongst the group selling at a peak of 625,000: the highest price in the data set. It is also noted that only 26 luxury houses have an overall quality of 10 with the lowest in the group selling for 310,000.

It is also noted that 12 houses with low overall quality of 1 and 2 sold for less than 82,000 with the lowest of it costing around 13,000.

The most important feature of this classification by overall quality is that many houses have an average quality ranging from 4 to 7 on a scale of 10. These houses however, have a median of between 100,000 and 200,000 in the quality average.

It is also noticed from previous analysis and visualization that most houses sold between 100,000 and 200,000 considering various features like full bath, garage type, garage cars, house style, sale price and condition and so on.

I hereby conclude that the predominant sale type is normal sale with warranty deed for houses sold between 100,000 and 200,000.