Exploratory Data Analysis(EDA) & Regression using Ames Housing Data

Objectives:

- To analyze and investigate the Ames Housing dataset and summarize its main characteristics.
- To build machine learning models to predict prices of houses

Imports and Reading Data

Importing Libraries:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import plotly.express as px
import seaborn as sns

%matplotlib inline
plt.style.use('ggplot') #default style for all visualisations
pd.set_option('display.max_columns', 100) #expanding no. of columns shown
pd.set_option('display.max_rows', 100) #expanding no. of rows shown
```

About the dataset:

The Ames Housing dataset contains information about individual residential property in Ames, Iowa, from 2006 to 2010. The dataset was collected by Dean De Cock in 2011, and additional information is available via the following links:

- A report describing the dataset
- Detailed documentation regarding the dataset's features
- The dataset in a tab-separated format

Reading the data:

In [2]:
```
# Loading the dataset into a dataframe
url = "https://jse.amstat.org/v19n3/decock/AmesHousing.txt"
df = pd.read_csv(url, delimiter='\t')
```

Understanding the dataset

An Overview:

In [3]:
```
df.shape
```

Out[3]: (2930, 82)

In [4]:
```
df.head()
```

Out[4]:

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilit
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	AllP
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllP
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllP
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	AllP
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllP

In [5]:
```
df.tail()
```

Out[5]:

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

In [6]:
```
#listing all columns at a glance
df.columns
```

Out[6]:
```
Index(['Order', 'PID', 'MS SubClass', 'MS Zoning', 'Lot Frontage', 'Lot Area',
       'Street', 'Alley', 'Lot Shape', 'Land Contour', 'Utilities',
       'Lot Config', 'Land Slope', 'Neighborhood', 'Condition 1',
       'Condition 2', 'Bldg Type', 'House Style', 'Overall Qual',
       'Overall Cond', 'Year Built', 'Year Remod/Add', 'Roof Style',
       'Roof Matl', 'Exterior 1st', 'Exterior 2nd', 'Mas Vnr Type',
       'Mas Vnr Area', 'Exter Qual', 'Exter Cond', 'Foundation', 'Bsmt Qual',
       'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin SF 1',
       'BsmtFin Type 2', 'BsmtFin SF 2', 'Bsmt Unf SF', 'Total Bsmt SF',
       'Heating', 'Heating QC', 'Central Air', 'Electrical', '1st Flr SF',
       '2nd Flr SF', 'Low Qual Fin SF', 'Gr Liv Area', 'Bsmt Full Bath',
       'Bsmt Half Bath', 'Full Bath', 'Half Bath', 'Bedroom AbvGr',
       'Kitchen AbvGr', 'Kitchen Qual', 'TotRms AbvGrd', 'Functional',
       'Fireplaces', 'Fireplace Qu', 'Garage Type', 'Garage Yr Blt',
       'Garage Finish', 'Garage Cars', 'Garage Area', 'Garage Qual',
       'Garage Cond', 'Paved Drive', 'Wood Deck SF', 'Open Porch SF',
       'Enclosed Porch', '3Ssn Porch', 'Screen Porch', 'Pool Area', 'Pool QC',
       'Fence', 'Misc Feature', 'Misc Val', 'Mo Sold', 'Yr Sold', 'Sale Type',
       'Sale Condition', 'SalePrice'],
      dtype='object')
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 82 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Order            2930 non-null   int64  
 1   PID              2930 non-null   int64  
 2   MS SubClass      2930 non-null   int64  
 3   MS Zoning        2930 non-null   object 
 4   Lot Frontage     2440 non-null   float64
 5   Lot Area         2930 non-null   int64  
 6   Street           2930 non-null   object 
 7   Alley            198 non-null    object 
 8   Lot Shape        2930 non-null   object 
 9   Land Contour     2930 non-null   object 
 10  Utilities        2930 non-null   object 
 11  Lot Config       2930 non-null   object 
 12  Land Slope       2930 non-null   object 
 13  Neighborhood     2930 non-null   object 
 14  Condition 1      2930 non-null   object 
 15  Condition 2      2930 non-null   object 
 16  Bldg Type        2930 non-null   object 
 17  House Style      2930 non-null   object 
 18  Overall Qual     2930 non-null   int64  
 19  Overall Cond     2930 non-null   int64  
 20  Year Built       2930 non-null   int64  
 21  Year Remod/Add   2930 non-null   int64  
 22  Roof Style       2930 non-null   object 
 23  Roof Matl        2930 non-null   object 
 24  Exterior 1st     2930 non-null   object 
 25  Exterior 2nd     2930 non-null   object 
 26  Mas Vnr Type     2907 non-null   object 
 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
```

General observations:

Looking at the dataframe general information:

- There are 82 columns/variables
- The dataset contains 2930 records/rows of data.
- Also, the type of data is heterogeneous: both numerical and categorical columns of data are available.
- Most of the columns are assigned the object datatype
- There are some null values in the dataset

Data Preparation

Combining, cleansing, enriching and transforming the raw Ames housing data to make it usable for Exploratory Data Analysis(EDA) and regression.

Dropping columns:

Focusing the analysis on a few house features of major interest because the dataset has 82 features which are too many to analyse at once.

In [8]:
```
#dropping columns by commenting them out. The purpose is to keep track of those
# retained and those that have been dropped.
df=df[[
```

```
        #'Order', 'PID', 'MS SubClass',
        'MS Zoning', 'Lot Frontage', 'Lot Area',
        #'Street', 'Alley', 'Lot Shape',
        'Land Contour',
        #'Utilities','Lot Config', 'Land Slope',
        'Neighborhood',
        #'Condition 1', 'Condition 2',
        'Bldg Type',
        #'House Style',
        'Overall Qual', 'Overall Cond', 'Year Built',
        #'Year Remod/Add', 'Roof Style',
        #'Roof Matl', 'Exterior 1st', 'Exterior 2nd', 'Mas Vnr Type',
        #'Mas Vnr Area', 'Exter Qual', 'Exter Cond', 'Foundation', 'Bsmt Qual',
        #'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin SF 1',
        #'BsmtFin Type 2', 'BsmtFin SF 2', 'Bsmt Unf SF',
        'Total Bsmt SF',
        #'Heating', 'Heating QC', 'Central Air', 'Electrical', '1st Flr SF',
        #'2nd Flr SF', 'Low Qual Fin SF',
        'Gr Liv Area',
        #'Bsmt Full Bath','Bsmt Half Bath', 'Full Bath', 'Half Bath', 'Kitchen Ab
        'Bedroom AbvGr', 'Full Bath',
        'Kitchen Qual', 'TotRms AbvGrd',
        #'Functional','Fireplaces', 'Fireplace Qu', 'Garage Type', 'Garage Yr Blt
        #'Garage Finish', 'Garage Cars', 'Garage Area', 'Garage Qual',
        #'Garage Cond', 'Paved Drive', 'Wood Deck SF', 'Open Porch SF',
        #'Enclosed Porch', '3Ssn Porch', 'Screen Porch', 'Pool Area', 'Pool QC',
        #'Fence', 'Misc Feature', 'Misc Val',
        'Mo Sold', 'Yr Sold',
        'Sale Type', 'Sale Condition', 'SalePrice']].copy()
```

- **Columns have been reduced from 82 columns down to 20 columns**

In [9]:
```
df.shape
```

Out[9]:
```
(2930, 20)
```

Renaming columns:

Removing spaces between column names and giving variables clear names with reference to the data description guide of the Ames Housing Dataset:

In [10]:
```
df = df.rename(columns={'MS Zoning':'MS_Zoning', 'Lot Frontage':'LotFrontage_ft'
           'Land Contour':'Land_Contour',
           #'Neighborhood',
           'Bldg Type':'Building_Type', 'Overall Qual':'Overall_Quality', 'Overa
           'Year Built':'Year_Built','Total Bsmt SF':'TotalBsmt_sqft', 'Gr Liv A
           'Bedroom AbvGr':'Bedroom_AbvGr', 'Full Bath':'Full_Bath',
           'Kitchen Qual':'Kitchen_Quality', 'TotRms AbvGrd':'TotRms_AbvGrd', 'M
           'Sale Type':'Sale_Type', 'Sale Condition':'Sale_Condition', 'SalePric
```

Assigning appropriate datatypes:

Overall_Quality, Overall_Condition and Month_Sold are categorical columns whose values are to be changed from numbers to more descriptive representative text. Which operation is not possible if they remain allocated to the int64 datatype.

In [11]:
```python
#changing from int64 to object datatype
df[['Overall_Quality', 'Overall_Condition', 'Month_Sold']] =\
df[['Overall_Quality', 'Overall_Condition', 'Month_Sold']].astype(object)
```

Renaming categorical feature values:

Replacing the unique values of the columns with more descriptive values in reference to the official Ames Housing Data Description Documentation

In [12]:
```python
# MS_Zoning Column
df['MS_Zoning'] = df['MS_Zoning'].replace({
    'RL': '(Res) Low Density', #(Res) stands for residential
    'RH': '(Res) High Density',
    'FV': '(Res) Floating Village',
    'RM': '(Res) Medium Density',
    'C (all)': 'Commercial',
    'I (all)': 'Industrial',
    'A (agr)': 'Agriculture'})
```

In [13]:
```python
# Overall_Quality Column
df['Overall_Quality'] = df['Overall_Quality'].replace({
       10: 'Very Excellent',
       9: 'Excellent',
       8: 'Very Good',
       7: 'Good',
       6: 'Above Average',
       5: 'Average',
       4: 'Below Average',
       3: 'Fair',
       2: 'Poor',
       1: 'Very Poor'})
```

In [14]:
```python
#overall condition columns
df['Overall_Condition'] = df['Overall_Condition'].replace({
       10: 'Very Excellent',
       9: 'Excellent',
       8: 'Very Good',
       7: 'Good',
       6: 'Above Average',
       5: 'Average',
       4: 'Below Average',
       3: 'Fair',
       2: 'Poor',
       1: 'Very Poor'})
```

In [15]:
```python
#Building_Type Column
df['Building_Type'] = df['Building_Type'].replace({
     '1Fam': 'Single-family Detached',
     '2fmCon': 'Two-family Conversion',
     'Duplx': 'Duplex',
     'TwnhsE': 'TwnHs End Unit',
     'Twnhs': 'TwnHs Inside Unit'})
```

In [16]:
```python
#Sale_Type Column
df['Sale_Type'] = df['Sale_Type'].replace({
    'WD ': 'Conventional WD', # WD stands for warranty deed
```

```
    'CWD': 'Cash WD',
    'VWD': 'VA Loan WD',
    'New': 'New on mkt',
    'COD': 'Court Officer Deed',
    'Con': 'Contract regular',
    'ConLw': '(Con) LowDown payt&I', #Con stands for contract
    'ConLI': '(Con) Low I', #I stands for Interest
    'ConLD': 'Contract LowDown',
    'Oth': 'Other'})
```

In [17]:
```
#Transforming Kitchen Column
df['Kitchen_Quality'] = df['Kitchen_Quality'].replace({
    'Ex': 'Excellent',
    'Gd': 'Good',
    'TA': 'Typical',
    'Fa': 'Fair',
    'Po': 'Poor'})
```

In [18]:
```
#Transforming Land Contour column
df['Land_Contour'] = df['Land_Contour'].replace({
    'Lvl': 'Near Flat/Level',
    'Bnk': 'Banked', #- Quick and significant rise from street grade to building
    'HLS': 'Hillside', #- Significant slope from side to side
    'Low': 'Depression'})
```

In [19]:
```
#Transforming Neighborhood column
df['Neighborhood'] = df['Neighborhood'].replace({
        'Blmngtn': 'Bloomington Heights',
        'Blueste': 'Bluestem',
        'BrDale': 'Briardale',
        'BrkSide': 'Brookside',
        'ClearCr': 'Clear Creek',
        'CollgCr': 'College Creek',
        'Crawfor': 'Crawford',
        'Edwards': 'Edwards',
        'Gilbert': 'Gilbert',
        'Greens': 'Greens',
        'GrnHill': 'Green Hills',
        'IDOTRR': 'Iowa DOT and Rail Road',
        'Landmrk': 'Landmark',
        'MeadowV': 'Meadow Village',
        'Mitchel': 'Mitchell',
        'NAmes': 'North Ames',
        'NoRidge': 'Northridge',
        'NPkVill': 'Northpark Villa',
        'NridgHt': 'Northridge Heights',
        'NWAmes': 'Northwest Ames',
        'OldTown': 'Old Town',
        'SWISU': 'South & West of Iowa State University',
        'Sawyer': 'Sawyer',
        'SawyerW': 'Sawyer West',
        'Somerst': 'Somerset',
        'StoneBr': 'Stone Brook',
        'Timber': 'Timberland',
        'Veenker': 'Veenker'})
```

In [20]:
```
#Transforming 'Month_Sold' column
df['Month_Sold'] = df['Month_Sold'].replace({
    1: 'Jan',
```

```
    2: 'Feb',
    3: 'Mar',
    4: 'Apr',
    5: 'May',
    6: 'Jun',
    7: 'Jul',
    8: 'Aug',
    9: 'Sep',
    10: 'Oct',
    11: 'Nov',
    12: 'Dec'})
```

Feature Engineering:

House age is a critical house feature to analyse as it highly influences the sale price. Though its not readily available in the dataset, it can be created using the readily available Year_Sold & Year_Built features.

In [21]:
```
#finding how old the property was as of the sale date
df['Property_Age'] = df['Year_Sold'] - df['Year_Built']
```

In [22]:
```
df['Property_Age']
```

Out[22]:
```
0       50
1       49
2       52
3       42
4       13
        ..
2925    22
2926    23
2927    14
2928    32
2929    13
Name: Property_Age, Length: 2930, dtype: int64
```

Checking for null values:

In [23]:
```
df.isna().sum()
```

Out[23]:
```
MS_Zoning              0
LotFrontage_ft       490
LotArea_sqft           0
Land_Contour           0
Neighborhood           0
Building_Type          0
Overall_Quality        0
Overall_Condition      0
Year_Built             0
TotalBsmt_sqft         1
GroundLivArea_sqft     0
Bedroom_AbvGr          0
Full_Bath              0
Kitchen_Quality        0
TotRms_AbvGrd          0
Month_Sold             0
Year_Sold              0
Sale_Type              0
Sale_Condition         0
SalePrice_USD          0
Property_Age           0
dtype: int64
```

OBSERVATION:

- The lotFrontage_ft and TotalBsmt_sqft are the only columns with null values.
- lotFrontage_ft column has 490 missing values and GroundLivArea_sqft column has 1 missing value

Checking for duplicates:

In [24]:
```
#count of duplicates
df.duplicated().sum()
```

Out[24]: 2

In [25]:
```
#eliminating the duplicates by making the inverse of the duplicates the new work
df = df[~df.duplicated()].reset_index(drop=True).copy()
```

In [26]:
```
df.shape
```

Out[26]: (2928, 21)

- 2 Duplicate rows have been removed thus reducing the row count from 2930 to 2928

Data Overview after data transformation & data cleaning

In [27]:
```
df.head()
```

Out[27]:

	MS_Zoning	LotFrontage_ft	LotArea_sqft	Land_Contour	Neighborhood	Building_Type	Ove
0	(Res) Low Density	141.0	31770	Near Flat/Level	North Ames	Single-family Detached	Ab
1	(Res) High Density	80.0	11622	Near Flat/Level	North Ames	Single-family Detached	
2	(Res) Low Density	81.0	14267	Near Flat/Level	North Ames	Single-family Detached	Ab
3	(Res) Low Density	93.0	11160	Near Flat/Level	North Ames	Single-family Detached	
4	(Res) Low Density	74.0	13830	Near Flat/Level	Gilbert	Single-family Detached	

Exploratory Data Analysis(EDA)

Analyzing and investigating the Ames Housing dataset and summarizing its main characteristics.

Objectives:

- To understand patterns within the Ames housing data
- To detect outliers or anomalous events
- To find interesting relations among the housing variables

Summary Statistics

In [28]:

```
df.describe().T
```

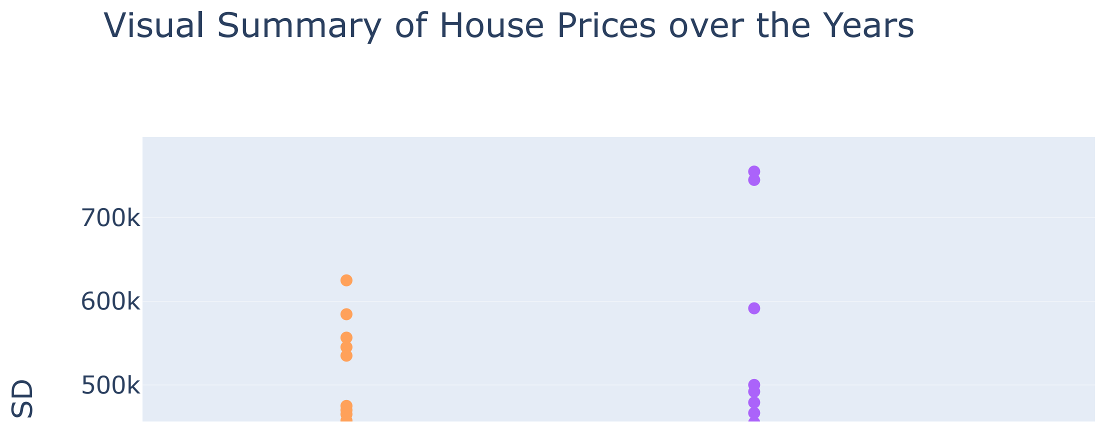
Out[28]:

	count	mean	std	min	25%	50%	75%
LotFrontage_ft	2438.0	69.230517	23.373933	21.0	58.00	68.0	80.
LotArea_sqft	2928.0	10148.768101	7882.487925	1300.0	7440.75	9436.5	11556.
Year_Built	2928.0	1971.348361	30.253979	1872.0	1954.00	1973.0	2001.
TotalBsmt_sqft	2927.0	1051.923129	440.328019	0.0	793.00	990.0	1302.
GroundLivArea_sqft	2928.0	1499.780738	505.650793	334.0	1126.00	1442.0	1743.
Bedroom_AbvGr	2928.0	2.853825	0.827739	0.0	2.00	3.0	3.
Full_Bath	2928.0	1.565915	0.552436	0.0	1.00	2.0	2.
TotRms_AbvGrd	2928.0	6.442964	1.573012	2.0	5.00	6.0	7.
Year_Sold	2928.0	2007.789617	1.316683	2006.0	2007.00	2008.0	2009.
SalePrice_USD	2928.0	180817.827186	79905.769953	12789.0	129500.00	160000.0	213500.
Property_Age	2928.0	36.441257	30.300296	-1.0	7.00	34.0	54.

In [29]:

```
fig = px.box(df, x='Year_Sold', y='SalePrice_USD',
             points ='outliers', color='Year_Sold',
```

```
            title='Visual Summary of House Prices over the Years')
fig.show()
```



OBSERVATIONS:

- All the years have scenarios where the houses being sold at extremely high prices. The biggest outlier is detected in 2007 as USD 755,000.
- There is not a lot of variation in the selling price of the different houses throughout the 5 years.
- Nearly all houses were sold for a price below USD 400,000

Feature Understanding

Exploring and analysing each housing variable in the Ames housing dataset, separately.

In [30]:
```
# Creating a grid of subplots
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(10, 12),
                        gridspec_kw={'hspace': 0.4, 'wspace': 0.5})

# Plotting the Kernel Density Plots in each subplot
sns.kdeplot(df['LotFrontage_ft'], ax=axs[0, 0], fill=True, color='blue')
sns.kdeplot(df['LotArea_sqft'], ax=axs[0, 1], fill=True, color='green')
```

```
sns.kdeplot(df['Year_Built'], ax=axs[0, 2], fill=True, color='red')
sns.kdeplot(df['TotalBsmt_sqft'], ax=axs[1, 0], fill=True, color='purple')
sns.kdeplot(df['GroundLivArea_sqft'], ax=axs[1, 1], fill=True, color='green')
sns.kdeplot(df['Bedroom_AbvGr'], ax=axs[1, 2], fill=True, color='red')
sns.kdeplot(df['TotRms_AbvGrd'], ax=axs[2, 0], fill=True, color='blue')
sns.kdeplot(df['SalePrice_USD'], ax=axs[2, 1], fill=True, color='green')
sns.kdeplot(df['Property_Age'], ax=axs[2, 2], fill=True, color='red')

# Setting titles for each subplot
axs[0, 0].set_title('Lot Frontage (lft)')
axs[0, 1].set_title('Lot Area Size (sqft)')
axs[0, 2].set_title('Year Built')
axs[1, 0].set_title('Basement Area (sqft)')
axs[1, 1].set_title('Living Area (sqft)')
axs[1, 2].set_title('Bedrooms')
axs[2, 0].set_title('Total Rooms')
axs[2, 1].set_title('Sale Price (USD)')
axs[2, 2].set_title('Property Age (yrs)')

#Removing the x-axis label
axs[0, 0].set(xlabel='')
axs[0, 1].set(xlabel='')
axs[0, 2].set(xlabel='')
axs[1, 0].set(xlabel='')
axs[1, 1].set(xlabel='')
axs[1, 2].set(xlabel='')
axs[2, 0].set(xlabel='')
axs[2, 1].set(xlabel='')
axs[2, 2].set(xlabel='')

# Setting a title for the overall plot
fig.suptitle('DISTRIBUTION OF VARIOUS HOUSING VARIABLES', fontsize=16, fontweigh

# Displaying the plot
plt.show()
```

OBSERVATIONS:

- Lot frontage, lot area size, basement area, living area and sale price features have a positively skewed distribution therefore most of the extreme values/outliers are on the right side thus they are higher values.

- Most of the houses have a basement size and living area size that ranges between 1000 - 2000 sqft

Visualising categorical housing data

Neighbourhoods in the dataset are quite many, so it would make sense to visualize just a few of them, for example, the top 10. Therefore, below, a separate dataframe with top neighbourhoods from which the visualization is to be made has been created.

In [31]:
```
# finding the top 10 neighbourhoods
df['Neighborhood'].value_counts().head(10)
```

Out[31]:
```
North Ames            443
College Creek         267
Old Town              239
Edwards               193
Somerset              182
Northridge Heights    166
Gilbert               165
Sawyer                151
Northwest Ames        131
Sawyer West           124
Name: Neighborhood, dtype: int64
```

In [32]:
```
#creating dataframe with only top 10 neighbourhoods
top_10 = df.loc[df['Neighborhood'].isin(['College Creek', 'Old Town',
'Edwards', 'Somerset', 'Northridge Heights', 'Gilbert',
'Sawyer', 'Northwest Ames', 'Sawyer West'])]
```

In [33]:
```
# Creating a grid of subplots
fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(16, 14),
            gridspec_kw={'hspace': 0.5, 'wspace': 0.5})

# Plotting the horizontal Count Plots in each subplot
ax1 = sns.countplot(data=df,  y="MS_Zoning", ax=axs[0, 0],
                   order=df["MS_Zoning"].value_counts().index)
ax2 = sns.countplot(data=df,  y="Land_Contour", ax=axs[0, 1])
ax3 = sns.countplot(data=top_10,  y="Neighborhood", ax=axs[0, 2],
                   order=top_10["Neighborhood"].value_counts().index) #data sour
ax4 = sns.countplot(data=df,  y="Building_Type", ax=axs[1, 0],
                   order=df["Building_Type"].value_counts().index)
ax5 = sns.countplot(data=df,  y="Overall_Quality", ax=axs[1, 1],
                   order=df["Overall_Quality"].value_counts().index)
ax6 = sns.countplot(data=df,  y="Overall_Condition", ax=axs[1, 2],
                   order=df["Overall_Condition"].value_counts().index)
ax7 = sns.countplot(data=df,  y="Kitchen_Quality", ax=axs[2, 0],
                   order=df["Kitchen_Quality"].value_counts().index)
ax8 = sns.countplot(data=df,  y="Month_Sold", ax=axs[2, 1],
                   order=df["Month_Sold"].value_counts().index)
ax9 = sns.countplot(data=df,  y="Sale_Type", ax=axs[2, 2],
                   order=df["Sale_Type"].value_counts().index)
ax10 = sns.countplot(data=df,  y="Sale_Condition", ax=axs[3, 0],
                    order=df["Sale_Condition"].value_counts().index)
ax11 = sns.countplot(data=df,  y="Full_Bath", ax=axs[3, 1],
                    order=df["Full_Bath"].value_counts().index)
ax12 = sns.countplot(data=df,  y="Year_Sold", ax=axs[3, 2],
                    order=df["Year_Sold"].value_counts().index)

# Looping through each subplot and moving x-axis ticks to the top
for ax in [ax1, ax2, ax2, ax3, ax4, ax5, ax6,
           ax7, ax8, ax9, ax10, ax11, ax12]:
    ax.xaxis.set_ticks_position('top')

# Setting titles for each subplot
axs[0, 0].set_title('MS Zoning')
axs[0, 1].set_title('Land Contour')
axs[0, 2].set_title('Top 10 Neighbourhoods')
```

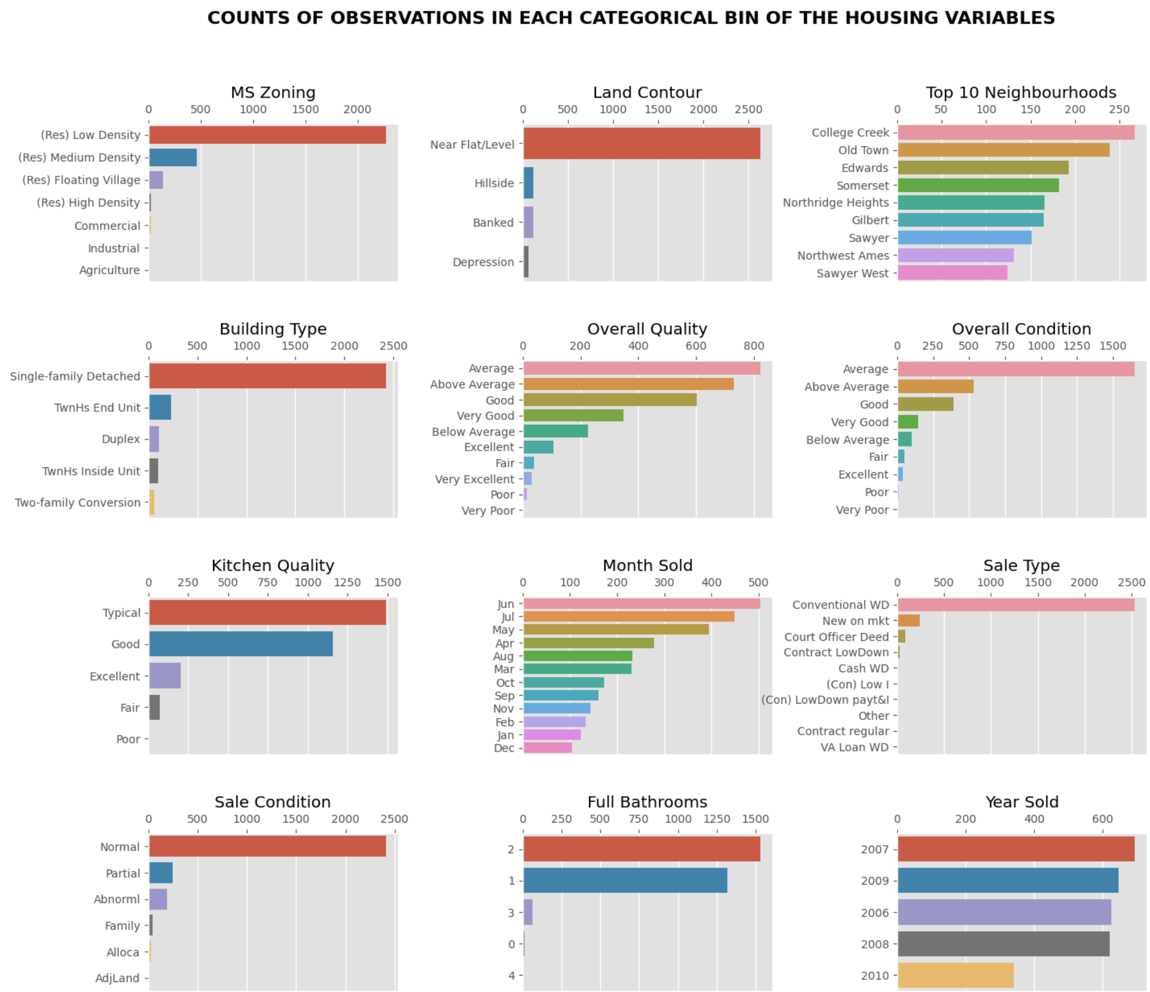
```
axs[1, 0].set_title('Building Type')
axs[1, 1].set_title('Overall Quality')
axs[1, 2].set_title('Overall Condition')
axs[2, 0].set_title('Kitchen Quality')
axs[2, 1].set_title('Month Sold')
axs[2, 2].set_title('Sale Type')
axs[3, 0].set_title('Sale Condition')
axs[3, 1].set_title('Full Bathrooms')
axs[3, 2].set_title('Year Sold')

#Looping through each subplot and removing the y-axis label
for ax in [ax1, ax2, ax2, ax3, ax4, ax5, ax6,
           ax7, ax8, ax9, ax10, ax11, ax12]:
    ax.set(ylabel='')

#Looping through each subplot and removing the x-axis label
for ax in [ax1, ax2, ax2, ax3, ax4, ax5, ax6,
           ax7, ax8, ax9, ax10, ax11, ax12]:
    ax.set(xlabel='')

# Setting a title for the overall plot
fig.suptitle('COUNTS OF OBSERVATIONS IN EACH CATEGORICAL BIN OF THE HOUSING VARI

# Displaying the plot
plt.show()
```



OBSERVATIONS:

- Most of the houses sold were of average condition and quality

- More than 600 houses were sold in 2007, 2009, 2006 and also 2008
- Fewer houses were sold between January and December. This might probably be because of the holidays.
- Most individuals preferred to purchase houses in low density areas
- There was a low demand for Two-family conversion type of houses
- People in Ames mostly preferred purchasing houses neighbouring the Colege Creek area

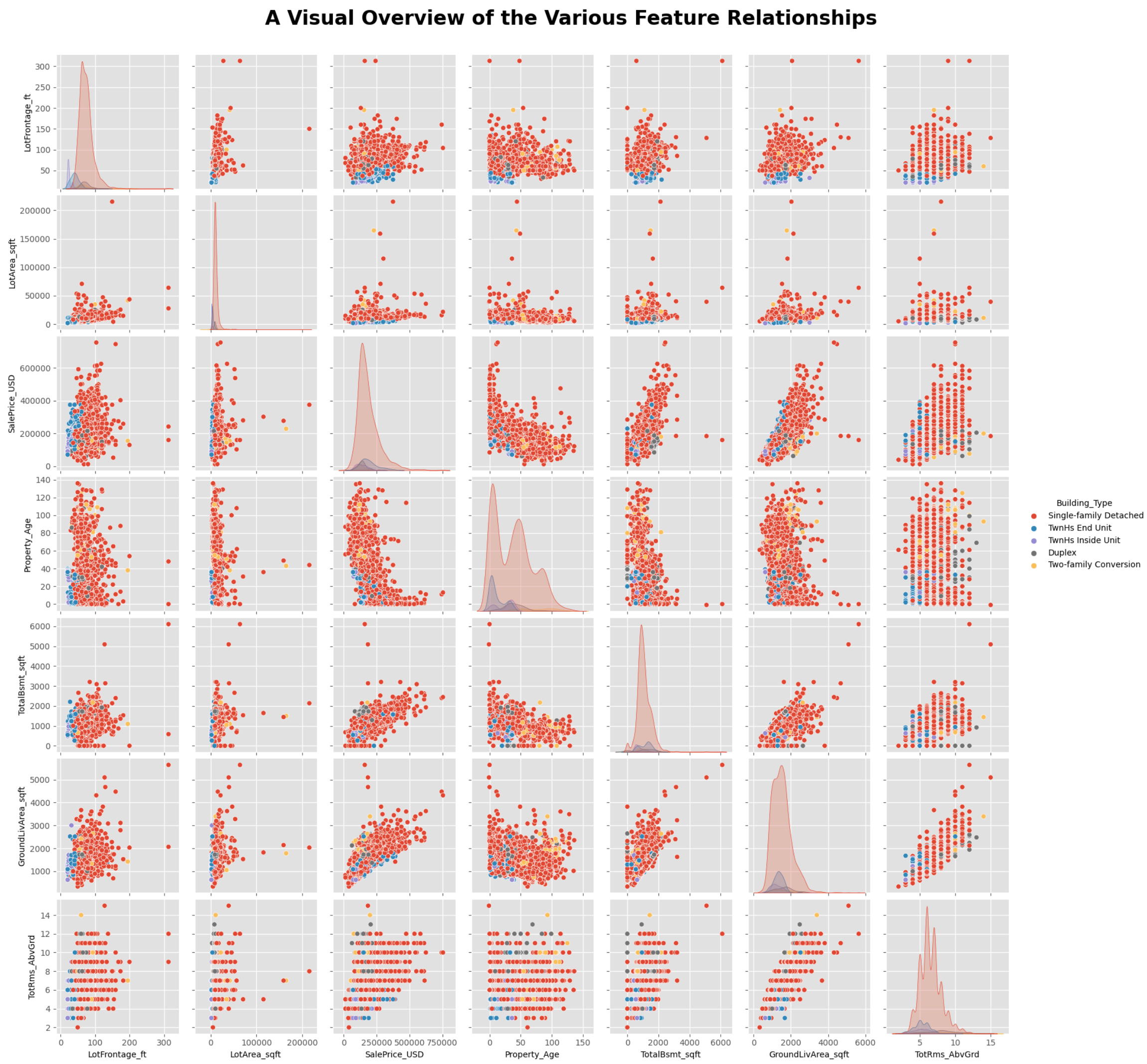
Feature Relationships

Analysing two quantitative housing variables to determine the nature of relationships between them.

In [34]:
```
#creating a pairplot
g = sns.pairplot(df,
             vars=['LotFrontage_ft', 'LotArea_sqft', 'SalePrice_USD', 'Property_
                      'TotalBsmt_sqft', 'GroundLivArea_sqft', 'TotRms_AbvGrd'],
            hue='Building_Type')

# Adding overall title
g.fig.suptitle('A Visual Overview of the Various Feature Relationships', fontsiz

#showing the pairplot
plt.show()
```



OBSERVATIONS:

- There is a positive relationship between:

Ground living Area and total number of rooms, Ground living Area and sale price , Basement area and sale price

- There is a negative relationship between property age and sale price. As the property age of the house increases, the price for which its sold decreases

- There is no relationship between lot Frontage and the total rooms of a house. Therefore they have no measurable effect on each other

In [35]:

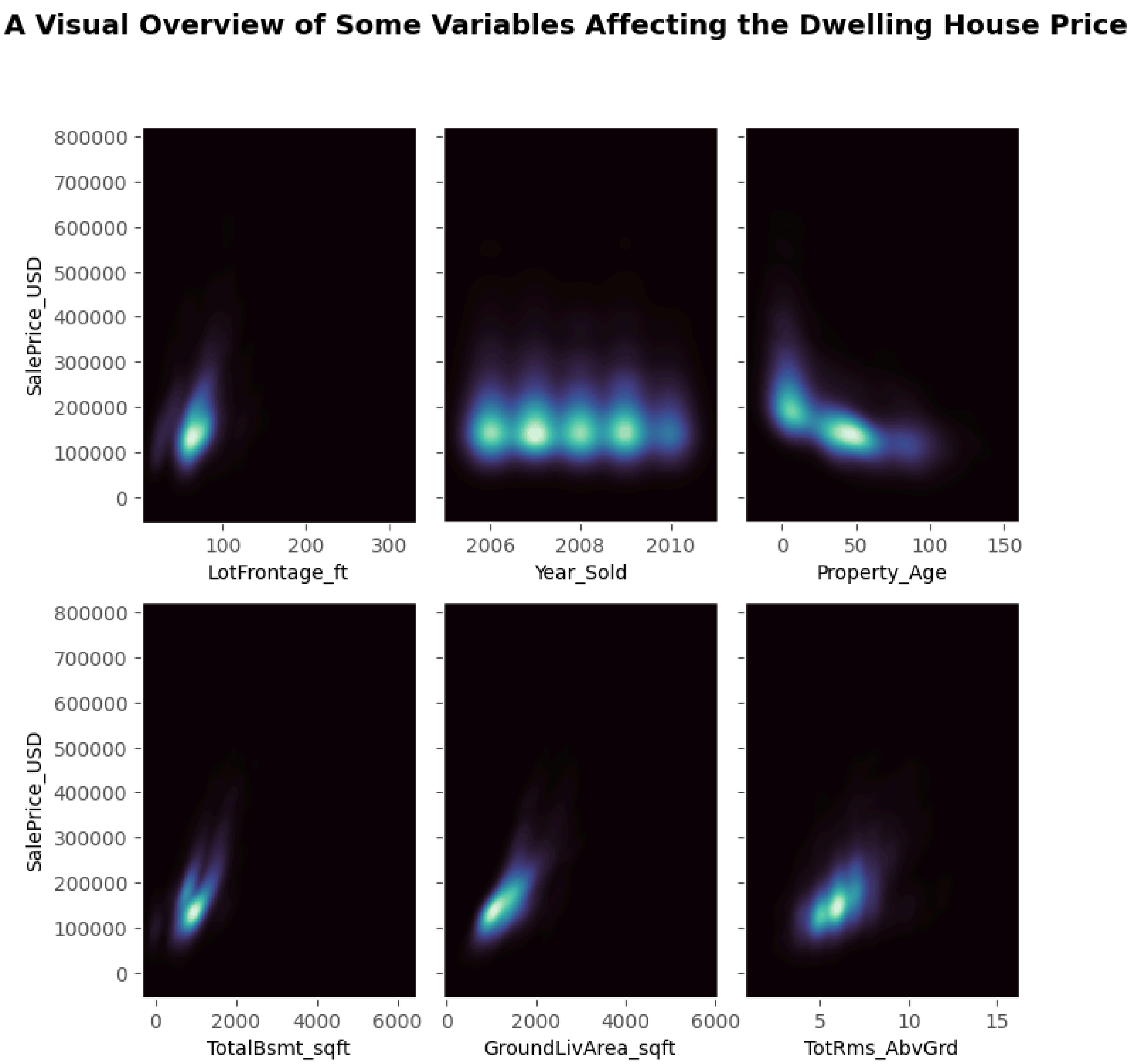
```
#creating a pairplot
# Creating a grid of subplots
fig, axs = plt.subplots(nrows=2, ncols=3,
                        figsize=(8, 8), sharey=True,
            gridspec_kw={'hspace': 0.2, 'wspace': 0.1})

ax1 = sns.kdeplot(
        data=df, x="LotFrontage_ft", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[0, 0])
ax2 = sns.kdeplot(
        data=df, x="Year_Sold", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[0, 1])
```

```
ax3 = sns.kdeplot(
        data=df, x="Property_Age", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[0, 2])
ax4 = sns.kdeplot(
        data=df, x="TotalBsmt_sqft", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[1, 0])
ax5 = sns.kdeplot(
        data=df, x="GroundLivArea_sqft", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[1, 1])
ax6 = sns.kdeplot(
        data=df, x="TotRms_AbvGrd", y="SalePrice_USD",
        fill=True, thresh=0, levels=100, cmap="mako", ax=axs[1, 2])


# Adding overall title
fig.suptitle('A Visual Overview of Some Variables Affecting the Dwelling House P

#showing the pairplot
plt.show()
```



Observations:

- Lot frontage, total basement area, ground living are and total rooms of houses have a positive relationship with the sale price.
- The year a house is sold doesn't have a relationship with its price

- There's a negative relationship between the Sale price and property age

CORRELATION:

Going a step futher than just identifying if the relationship is positive or negative.

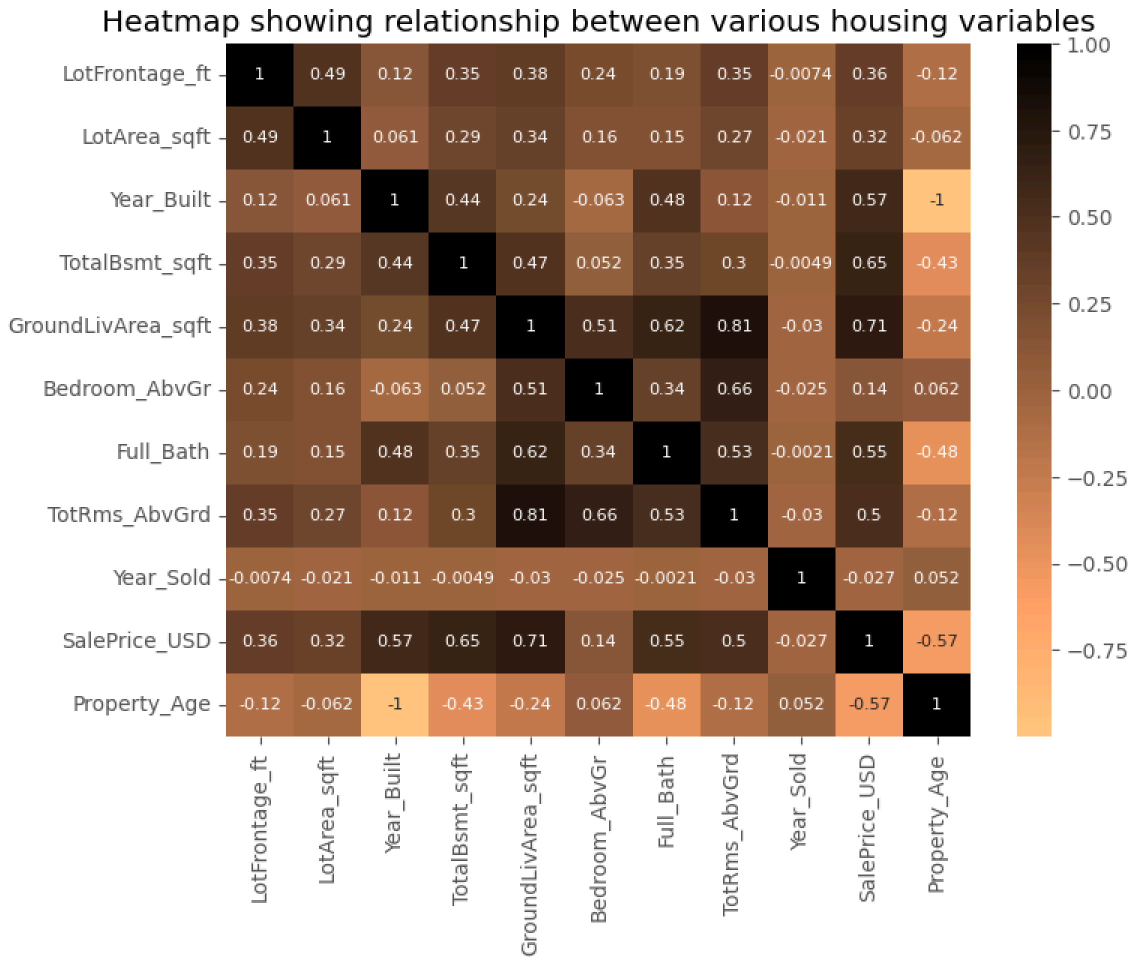
Below, the **extent** to which two housing variables are linearly related is measured:

In [36]:
```
df_corr = df[['LotFrontage_ft', 'LotArea_sqft', 'Year_Built', 'TotalBsmt_sqft',
         'GroundLivArea_sqft', 'Bedroom_AbvGr', 'Full_Bath', 'TotRms_AbvGrd',
         'Year_Sold', 'SalePrice_USD', 'Property_Age']].dropna().corr()
df_corr
```

Out[36]:

	LotFrontage_ft	LotArea_sqft	Year_Built	TotalBsmt_sqft	GroundLivArea_sq
LotFrontage_ft	1.000000	0.491849	0.122147	0.354031	0.38462
LotArea_sqft	0.491849	1.000000	0.061409	0.294912	0.34254
Year_Built	0.122147	0.061409	1.000000	0.435021	0.24278
TotalBsmt_sqft	0.354031	0.294912	0.435021	1.000000	0.46977
GroundLivArea_sqft	0.384629	0.342541	0.242788	0.469770	1.00000
Bedroom_AbvGr	0.241380	0.161172	-0.063066	0.052087	0.50731
Full_Bath	0.185950	0.149263	0.476621	0.346055	0.62405
TotRms_AbvGrd	0.354322	0.274950	0.116985	0.302828	0.81115
Year_Sold	-0.007405	-0.021224	-0.010509	-0.004913	-0.02951
SalePrice_USD	0.358163	0.321026	0.566885	0.647416	0.70609
Property_Age	-0.122296	-0.062217	-0.999122	-0.434653	-0.24370

In [37]:
```
plt.figure(figsize=(8,6))
sns.heatmap(df_corr,  annot=True, annot_kws={"fontsize":8},
            cmap="copper_r").set_title("Heatmap showing relationship between var
plt.show()
```



Observations:

- There is a strong positive correlation(0.71) between the ground living area size of houses and their selling price. Therefore, the `GroundLivArea_sqft` feature can be considered one of the most important features for model training in a bid to predict house prices.

REGRESSION

Building machine learning models to predict prices of houses based on their ground living area size.

Linear Regression

- **Splitting data into X and Y variables**

Separating input values (features) and the expected output (label) into separate numpy arrays

In [38]:
```
y = df['SalePrice_USD']
y
```

Out[38]:
```
0       215000
1       105000
2       172000
3       244000
4       189900
         ...
2923    142500
2924    131000
2925    132000
2926    170000
2927    188000
Name: SalePrice_USD, Length: 2928, dtype: int64
```

Below, a `reshape` on the input data is performed in order for the Linear Regression package to understand it correctly. Linear Regression expects a 2D-array as an input, where each row of the array corresponds to a vector of input features. In this case, since I have only one input - an array with shape N×1 is needed, where N is the dataset size.

In [39]:
```
X = df['GroundLivArea_sqft'].to_numpy().reshape(-1,1)
X
```

Out[39]:
```
array([[1656],
       [ 896],
       [1329],
       ...,
       [ 970],
       [1389],
       [2000]], dtype=int64)
```

- **Performing 80/20 data split**

Splitting the data into train and test datasets, so as to validate the model after training.

In [40]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Now the following four variables have been created:

- X_train: The `GroundLivArea_sqft` feature values to be used to train the model
- y_train: The corresponding `SalePrice_USD` labels to be used to train the model
- X_test: The `GroundLivArea_sqft` feature values to be used to validate the model
- y_test: The corresponding `SalePrice_USD` labels to be used to validate the model

- **Training the model**

Defining the `LinearRegression object`, and fitting it to the data using the `fit` method:

In [41]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train)
```

Out[41]:

```
▾ LinearRegression
LinearRegression()
```

- **Applying the model to make house sale price predictions**

In [42]:

```
y_lr_train_pred = lin_reg.predict(X_train)

# predicting prices on a test dataset
y_lr_test_pred = lin_reg.predict(X_test)
```

In [43]:

```
y_lr_train_pred
```

Out[43]:

```
array([195202.24929397, 109967.13297948, 100811.44273598, ...,
       194112.28616975, 206537.86578593, 236402.85538973])
```

In [44]:

```
y_lr_test_pred
```

```
Out[44]: array([314771.2040216 , 198581.13497908, 211551.69615737, 145063.94557957,
                145935.91607895, 200543.06860268, 104408.32104592, 172531.01631007,
                150513.7612007 , 111711.07397824, 227029.17252138, 109640.14404221,
                233132.96601705, 157925.51044544, 199998.08704057, 146262.90501621,
                150949.74645039, 111711.07397824, 179833.76924238, 182449.68074053,
                154437.62844791, 231607.01764313, 201306.04278964, 123264.68309504,
                193131.31935794, 148333.83495224, 241416.68576117, 195638.23454366,
                172095.03106038, 193894.2935449 , 165991.23756471, 242179.65994813,
                111711.07397824, 132747.36227581, 191169.38573434, 207409.83628531,
                182558.67705295, 150949.74645039, 294824.87884826, 212205.6740319 ,
                160977.40719327, 108332.18829314, 274987.54998734, 189316.44842315,
                260055.05518544, 295696.84934764, 115416.94860061, 178307.82086847,
                150186.77226343, 166100.23387713, 161413.39244296, 106806.23991922,
                167408.1896262 , 246103.52719534, 147897.84970255, 181795.70286599,
                228228.13195803, 241852.67101086,  86205.93687135, 208826.7883468 ,
                164901.27444048, 162285.36294234, 105607.28048257, 210243.7404083 ,
                213731.62240582, 165228.26337775, 131984.38808885, 178961.798743  ,
                270627.69749043, 153565.65794853, 197055.18660516, 168934.13800012,
                187572.50742439, 234767.91070339, 147243.87182802, 199889.09072815,
                177871.83561878, 195856.22716851, 136344.24058575, 196619.20135547,
                151821.71694977, 204575.93216232, 167953.17118832, 183757.6364896 ,
                 96996.57180119, 151385.73170008, 178525.81349331, 132965.35490065,
                155527.59157214, 257112.15475003, 192259.34885856, 147897.84970255,
                148333.83495224, 176236.89093244, 196946.19029274, 202178.01328902,
                204793.92478717, 173402.98680945, 195202.24929397,  93617.68611608,
                345072.17887509, 147788.85339013, 119558.80847267, 246103.52719534,
                192368.34517099,  89475.82624402, 207191.84366046, 176236.89093244,
                194112.28616975, 140050.11520812, 177653.84299393, 111711.07397824,
                191169.38573434, 234658.91439097, 128932.49134102, 353028.90968194,
                144191.97508018, 119558.80847267, 174928.93518337, 136562.2332106 ,
                130131.45077767, 113673.00760185, 217655.48965304, 194766.26404428,
                198254.14604181, 150513.7612007 , 278257.43936002, 254278.25062704,
                238582.78163818, 164356.29287837, 322182.95326634, 206210.87684866,
                288503.09272774, 191496.3746716 , 111711.07397824, 199780.09441572,
                359786.68105214, 303108.59859238, 487966.34446115, 196728.19766789,
                255586.20637611, 171659.04581069, 292208.96735011, 160977.40719327,
                 74216.34250486, 192041.35623372, 150513.7612007 , 126098.58721803,
                112801.03710247, 143755.98983049, 130240.44709009,  90783.7819931 ,
                179288.78768027, 172531.01631007, 163157.33344172, 275314.53892461,
                136453.23689818, 298312.76084578, 305942.50271536, 179506.78030512,
                387580.74071991, 197927.15710454, 145717.9234541 , 260927.02568482,
                148333.83495224, 159887.44406904, 131330.41021431, 227901.14302076,
                114980.96335092, 152148.70588704, 168607.14906285, 136562.2332106 ,
                189316.44842315, 150840.75013796, 198472.13866665, 253079.29119039,
                138851.15577148, 229754.08033195, 159451.45881935, 244250.58988416,
                190188.41892253, 374937.16847889, 130894.42496462, 264087.91874508,
                162503.35556719, 172531.01631007, 158688.48463239,  96015.60498938,
                169261.12693739, 283816.25129357, 178743.80611816, 261363.01093451,
                111711.07397824, 114217.98916396, 122174.71997081, 131439.40652674,
                122610.7052205 , 221797.3495251 , 239236.75951272, 107460.21779376,
                217764.48596546,  90783.7819931 , 169261.12693739, 169479.11956223,
                206537.86578593, 203812.95797536, 195529.23823124, 253842.26537735,
                163375.32606657, 214385.60028036, 310411.35152469, 193458.30829521,
                122174.71997081, 166645.21543925, 177108.86143182, 167408.1896262 ,
                 99721.47961175, 186482.54430016, 178089.82824362, 131984.38808885,
                181141.72499146, 132311.37702612, 230081.06926922, 152257.70219946,
                245122.56038354, 150731.75382554, 236184.86276488, 164901.27444048,
                200107.08335299, 274551.56473765, 199453.10547846, 171441.05318584,
                126970.55771741, 165991.23756471, 176781.87249455, 132420.37333854,
                183539.64386475, 136562.2332106 , 136562.2332106 , 150295.76857585,
```

```
353900.88018132, 198908.12391634, 172967.00155976, 272589.63111404,
166863.20806409, 109095.1624801 , 136562.2332106 , 198036.15341696,
114980.96335092, 368724.3786708 , 139723.12627086, 114435.98178881,
 76396.26875331, 189970.42629769, 121956.72734597, 167299.19331378,
132311.37702612, 197818.16079212, 149532.79438889, 188662.47054862,
242942.63413509, 196401.20873062, 236947.83695184, 214167.60765551,
186700.53692501, 356734.78430431, 154982.61001002, 318150.0897067 ,
188226.48529893, 236511.85170215, 146480.89764106, 174492.94993367,
159669.4514442 , 172967.00155976, 204684.92847474, 219399.4306518 ,
176236.89093244, 213513.62978098, 294170.90097372, 129368.47659071,
153565.65794853, 175909.90199517, 213731.62240582, 305288.52484083,
175800.90568275, 233568.95126674, 146153.90870379, 208826.7883468 ,
226048.20570958, 292317.96366254, 146371.90132864, 163266.32975414,
 90129.80411856, 130894.42496462, 151603.72432492, 192913.3267331 ,
181032.72867903, 136562.2332106 , 155527.59157214, 142012.04883173,
195420.24191882, 195856.22716851, 186809.53323743, 105062.29892046,
330575.66932288, 116942.89697453, 153783.65057338, 188226.48529893,
160759.41456842, 134491.30327457, 174165.96099641, 173402.98680945,
128714.49871617, 237056.83326426, 152693.68744915, 231171.03239344,
188117.4889865 , 148333.83495224, 173729.97574672, 207409.83628531,
146589.89395348, 141467.06726962, 116724.90434968, 147897.84970255,
111711.07397824, 205992.88422381, 111711.07397824, 150295.76857585,
181250.72130388, 194548.27141944, 155854.58050941, 130894.42496462,
182885.66599022, 161849.37769265, 271717.66061466, 141794.05620689,
205120.91372443, 175800.90568275, 208281.80678469, 118250.8527236 ,
245776.53825807, 233459.95495432, 136562.2332106 , 226920.17620896,
169915.10481192, 123700.66834473, 148660.82388951, 225503.22414746,
200870.05753995, 227029.17252138, 153892.6468858 , 175800.90568275,
196728.19766789, 176236.89093244, 276186.50942399, 152257.70219946,
180160.75817965, 137107.21477271, 207191.84366046, 224849.24627293,
206537.86578593, 196510.20504305, 231389.02501829, 325779.83157628,
188117.4889865 , 172095.03106038, 125226.61671865, 204793.92478717,
198036.15341696, 153783.65057338, 199562.10179088, 314880.20033402,
267684.79705502, 149532.79438889, 203921.95428778, 250790.36862951,
103645.34685897, 168171.16381316, 123046.69047019, 248828.43500591,
125117.62040622, 130458.43971493, 218527.46015242, 218636.45646484,
141249.07464477, 127406.5429671 , 189970.42629769, 111929.06660309,
110839.10347886, 204357.93953747, 264196.9150575 , 196292.2124182 ,
200652.06491511, 106479.25098195, 323817.89795268, 109640.14404221,
179070.79505543, 254278.25062704, 217655.48965304, 247738.47188168,
164247.29656595, 147897.84970255, 221143.37165056, 151494.7280125 ,
180378.7508045 , 247847.4681941 , 139287.14102117, 111711.07397824,
155854.58050941, 223323.29789901, 131875.39177643, 113346.01866458,
130022.45446524, 412867.88520196, 134927.28852426, 194221.28248217,
126098.58721803, 150077.77595101, 194657.26773186, 165555.25231502,
150840.75013796, 307032.46583959, 120321.78265963, 201524.03541449,
106479.25098195, 187354.51479954, 205011.91741201, 132638.36596339,
287849.11485321, 164029.3039411 , 264305.91136992, 236511.85170215,
120866.76422174,  93181.70086639, 114762.97072607, 159015.47356966,
182994.66230264, 116942.89697453, 124790.63146896, 198472.13866665,
 65278.6448862 , 179942.76555481, 202178.01328902, 177980.8319312 ,
223650.28683628, 140922.08570751, 134382.30696215, 181577.71024115,
306378.48796505, 126425.5761553 , 156290.5657591 , 155745.58419698,
229427.09139468, 130894.42496462, 116942.89697453, 122937.69415777,
211551.69615737, 320003.02701788, 111711.07397824, 168389.15643801,
230953.0397686 , 184956.59592625, 181250.72130388, 173838.97205914,
 97323.56073845, 216674.52284123, 201633.03172691, 111384.08504097,
243814.60463447, 160541.42194358, 163266.32975414, 136562.2332106 ,
248828.43500591, 173402.98680945, 111711.07397824, 201306.04278964,
227247.16514623, 263869.92612023, 205883.88791139, 166645.21543925,
```

```
       180487.74711692, 209262.77359649, 106479.25098195, 140268.10783297,
       120103.79003478, 161304.39613054, 129913.45815282, 257330.14737487,
       156399.56207152, 310738.34046196, 196510.20504305, 148333.83495224,
       142448.03408142, 174056.96468398, 194984.25666913, 193022.32304552,
       130458.43971493, 292426.95997496, 177544.84668151, 173947.96837156,
       263433.94087054, 127079.55402983, 136562.2332106 , 202831.99116356,
       179288.78768027,  90783.7819931 , 198145.14972939, 221252.36796298,
       237819.80745122, 116942.89697453, 148333.83495224, 231171.03239344,
       202396.00591387, 221034.37533814, 215039.57815489, 190297.41523496,
       166536.21912682,  98958.50542479, 182122.69180326, 243705.60832204,
       161849.37769265, 150186.77226343, 178743.80611816, 130894.42496462,
       203049.9837884 , 259183.08468606, 245340.55300838, 119122.82322298,
       199671.0981033 , 255150.22112642, 168934.13800012, 353900.88018132,
       265395.87449415, 217982.4785903 , 156726.55100879, 125226.61671865,
       181468.71392872, 125226.61671865, 173402.98680945, 219726.41958907,
       197382.17554243, 172967.00155976, 114980.96335092, 135363.27377395,
       201742.02803933, 179288.78768027, 154873.6136976 , 228337.12827045,
       131330.41021431, 199562.10179088, 311610.31096134, 102991.36898443,
       139614.12995843, 125117.62040622, 237601.81482638, 180160.75817965,
       228882.10983257, 229863.07664437, 119558.80847267, 285887.1812296 ,
       126970.55771741, 201742.02803933, 151058.74276281, 259837.06256059,
       182231.68811568, 179833.76924238, 181141.72499146, 314662.20770917,
       155854.58050941, 189098.45579831, 168171.16381316, 134382.30696215,
        96015.60498938, 128714.49871617, 180378.7508045 , 184847.59961382,
       212859.65190644, 111711.07397824, 243051.63044751, 186700.53692501,
       172204.0273728 , 221906.34583752, 149859.78332616, 182449.68074053,
       305288.52484083, 194766.26404428])
```

- **Evaluating model performance**

Measuring how close house price predictions are to the expected prices.

In [45]:
```
from sklearn.metrics import mean_squared_error,r2_score

lr_train_mse = mean_squared_error(y_train, y_lr_train_pred)
lr_train_r2 = r2_score(y_train, y_lr_train_pred)

lr_test_mse = mean_squared_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)
```

In [46]:
```
print('LR MSE (Train): ',lr_train_mse)
print('LR R2 (Train): ',lr_train_r2)
print('LR MSE (Test): ',lr_test_mse)
print('LR R2 (Test): ',lr_test_r2)
```

```
LR MSE (Train):  3215818093.8122363
LR R2 (Train):  0.47869722670799963
LR MSE (Test):  3112173832.7581115
LR R2 (Test):  0.5696028228959853
```

In [47]:
```
lr_results = pd.DataFrame(['Linear Regression', lr_train_mse, lr_train_r2,
                          lr_test_mse, lr_test_r2]).transpose()
#rename columns
lr_results.columns = ['Method', 'Training MSE', 'Training R2', 'Test MSE', 'Test

lr_results
```

Out[47]:

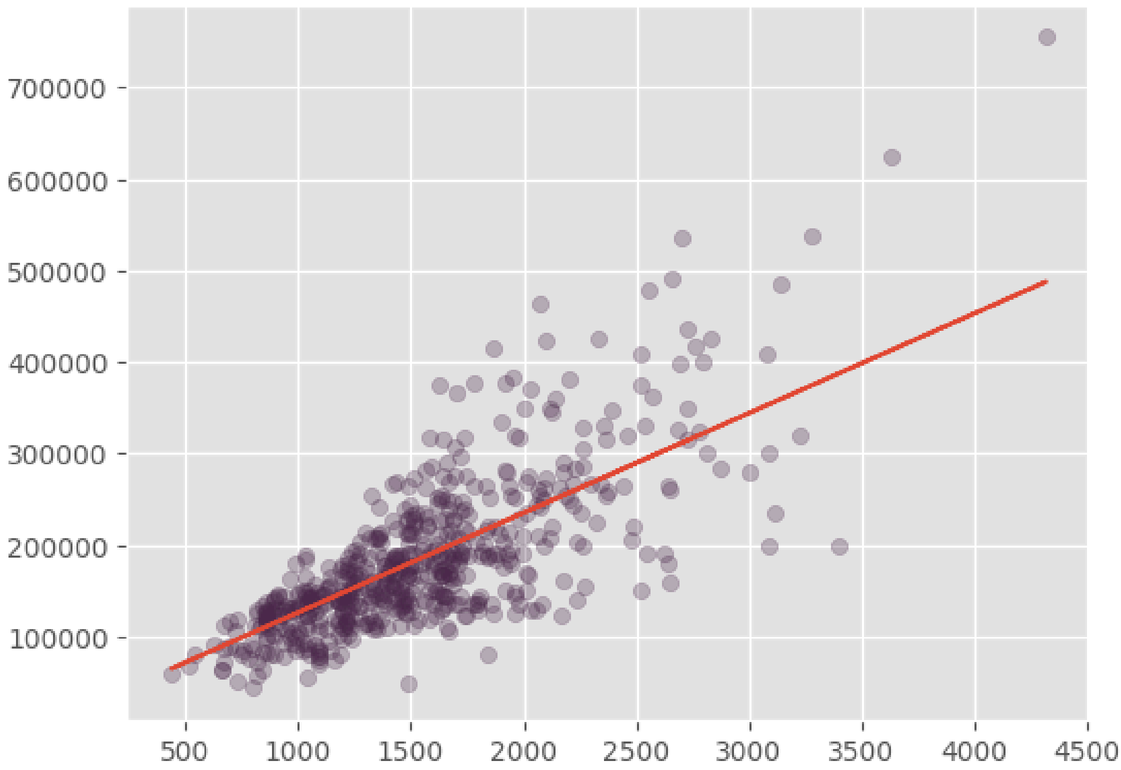
	Method	Training MSE	Training R2	Test MSE	Test R2
0	Linear Regression	3215818093.812236	0.478697	3112173832.758111	0.569603

Observation:

- The co-efficient of determination(Test R2) of the linear regression model is 0.57, a value that lies between 0 and 1. Therefore, this linear regression model partially predicts the price of houses. If the co-efficient of determination was 1, it could have meant that this model perfectly predicts the price of houses

Plotting the test data together with the regression line to better evaluate the linear regression model performance

In [48]:
```
plt.scatter(X_test,y_test,c="#4c2a4c", alpha=0.3)
plt.plot(X_test,y_lr_test_pred)
plt.show()
```



Observation: The regression line is closely following the scatter points. This suggests that the linear regression model is making accurate house price predictions.

Random Forest Regression

- **Training the model**

In [49]:
```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(max_depth=2, random_state=0)
rf.fit(X_train, y_train)
```

Out[49]:

```
RandomForestRegressor
RandomForestRegressor(max_depth=2, random_state=0)
```

- **Applying the model to make house sale price predictions**

In [50]:

```
y_rf_train_pred = rf.predict(X_train)
y_rf_test_pred = rf.predict(X_test)
```

- **Evaluating model performance**

In [51]:

```
from sklearn.metrics import mean_squared_error,r2_score

rf_train_mse = mean_squared_error(y_train, y_rf_train_pred)
rf_train_r2 = r2_score(y_train, y_rf_train_pred)

rf_test_mse = mean_squared_error(y_test, y_rf_test_pred)
rf_test_r2 = r2_score(y_test, y_rf_test_pred)
```

In [52]:

```
rf_results = pd.DataFrame(['Random Forest', rf_train_mse, rf_train_r2,
                          rf_test_mse, rf_test_r2]).transpose()
#rename columns
rf_results.columns = ['Method', 'Training MSE', 'Training R2', 'Test MSE', 'Test

rf_results
```

Out[52]:

	Method	Training MSE	Training R2	Test MSE	Test R2
0	Random Forest	3133501584.973712	0.492041	3474153679.222308	0.519543

Model Comparison: Linear regression vs. Random forest

In [53]:

```
# Creating rf_results data as a list
rf_results = ['Random Forest', 3133501584.973712, 0.492041, 3474153679.222308, 0

# Adding the rf_results to the lr_results DataFrame
lr_results.loc[len(lr_results)] = rf_results
df_models = lr_results
df_models
```

Out[53]:

	Method	Training MSE	Training R2	Test MSE	Test R2
0	Linear Regression	3215818093.812236	0.478697	3112173832.758111	0.569603
1	Random Forest	3133501584.973712	0.492041	3474153679.222308	0.519543

Observation: The Mean Squared Error(Test MSE) of the linear regression model is a bit lower than that of the Random Forest regression model. This means the amount of prediction errors made by the model is lower. Therefore, the linear regression model is slightly better at predicting house prices compared to the random forest model.