In [ ]:

In [423…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import plotly.express as px
import folium
```

In [25]:
```
dataset = pd.read_csv(r"D:\Next hike 3-project-Jan-25\housing_data.csv")
```

In [5]:
```
dataset
```

Out[5]:

	Unnamed: 0	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fen
0	0	SC60	RL	65	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1	1	SC20	RL	80	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
2	2	SC60	RL	68	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
3	3	SC70	RL	60	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
4	4	SC60	RL	84	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1455	1455	SC60	RL	62	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1456	1456	SC20	RL	85	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	No	MnP
1457	1457	SC70	RL	66	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	No	GdP
1458	1458	SC20	RL	68	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1459	1459	SC20	RL	75	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N

1460 rows × 81 columns

In [33]:
```
# Printing first 5 records of the dataset
dataset.head()
```

Out[33]:

	Unnamed: 0	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence
0	0	SC60	RL	65	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	No	No
1	1	SC20	RL	80	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	No	No
2	2	SC60	RL	68	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	No	No
3	3	SC70	RL	60	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	No	No
4	4	SC60	RL	84	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	No	No

5 rows × 81 columns

In [37]:
```
dataset.tail()
```

Out[37]:

	Unnamed: 0	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fen
1455	1455	SC60	RL	62	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1456	1456	SC20	RL	85	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	No	MnP
1457	1457	SC70	RL	66	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	No	GdP
1458	1458	SC20	RL	68	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1459	1459	SC20	RL	75	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N

5 rows × 81 columns

In [35]:
```
dataset = dataset.drop_duplicates()
```

In [39]:
```
dataset.shape
```

Out[39]:
```
(1460, 81)
```

In [30]:
```
dataset
```

Out[30]:

	Unnamed: 0	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fen
0	0	SC60	RL	65	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1	1	SC20	RL	80	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
2	2	SC60	RL	68	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
3	3	SC70	RL	60	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
4	4	SC60	RL	84	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	No	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1455	1455	SC60	RL	62	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1456	1456	SC20	RL	85	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	No	MnP
1457	1457	SC70	RL	66	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	No	GdP
1458	1458	SC20	RL	68	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N
1459	1459	SC20	RL	75	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	No	N

1460 rows × 81 columns

In [41]:
```
dataset.columns
```

Out[41]:
```
Index(['Unnamed: 0', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea',
       'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [43]:
```
# Rename the columns for better understanding:
```

In [47]:
```
rename_col = {'MSSubClass':'Identifies_the_type_of_dwelling','LotConfig':'Lot_configuration','BldgType':'Type_of_dwelling',
              'RoofMatl':'Roof_material','MasVnrType':'Masonry_veneer_type',
              'Electrical':'Electrical_system','LowQualFinSF':'Low_quality_finished_square_feet',
              'KitchenQual':'Kitchen_Quality','GarageType':'Garage_location',
              'GarageCond':'Garage_condition','PavedDrive':'Paved driveway',
              'PoolQC':'Pool_quality','Fence':'Fence_Quality',
              'MiscFeature':'Miscellaneous_feature','MiscVal':'miscellaneous_feature'}
```

In [49]:
```
dataset = dataset.rename(columns=rename_col)
dataset.head()
```

Out[49]:

	Unnamed: 0	Identifies_the_type_of_dwelling	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea
0	0	SC60	RL	65	8450	Pave	NaN	Reg	Lvl	AllPub	...	0
1	1	SC20	RL	80	9600	Pave	NaN	Reg	Lvl	AllPub	...	0
2	2	SC60	RL	68	11250	Pave	NaN	IR1	Lvl	AllPub	...	0
3	3	SC70	RL	60	9550	Pave	NaN	IR1	Lvl	AllPub	...	0
4	4	SC60	RL	84	14260	Pave	NaN	IR1	Lvl	AllPub	...	0

5 rows × 81 columns

In [51]:
```
dataset.columns
```

Out[51]:
```
Index(['Unnamed: 0', 'Identifies_the_type_of_dwelling', 'MSZoning',
       'LotFrontage', 'LotArea', 'Street', 'Alley', 'LotShape', 'LandContour',
       'Utilities', 'Lot_configuration', 'LandSlope', 'Neighborhood',
       'Condition1', 'Condition2', 'Type_of_dwelling', 'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
       'Roof_material', 'Exterior1st', 'Exterior2nd', 'Masonry_veneer_type',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical_system', '1stFlrSF', '2ndFlrSF',
       'Low_quality_finished_square_feet', 'GrLivArea', 'BsmtFullBath',
       'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
       'Kitchen_Quality', 'TotRmsAbvGrd', 'Functional', 'Fireplaces',
       'FireplaceQu', 'Garage_location', 'GarageYrBlt', 'GarageFinish',
       'GarageCars', 'GarageArea', 'GarageQual', 'Garage_condition',
       'Paved driveway', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch',
       '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Pool_quality', 'Fence_Quality',
       'Miscellaneous_feature', 'miscellaneous_feature', 'MoSold', 'YrSold',
       'SaleType', 'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [53]:
```
dataset.shape
```

Out[53]:
```
(1460, 81)
```

In [55]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Unnamed: 0                       1460 non-null   int64  
 1   Identifies_the_type_of_dwelling  1460 non-null   object 
 2   MSZoning                         1460 non-null   object 
 3   LotFrontage                      1460 non-null   int64  
 4   LotArea                          1460 non-null   int64  
 5   Street                           1460 non-null   object 
 6   Alley                            91 non-null     object 
 7   LotShape                         1460 non-null   object 
 8   LandContour                      1460 non-null   object 
 9   Utilities                        1460 non-null   object 
 10  Lot_configuration                1460 non-null   object 
 11  LandSlope                        1460 non-null   object 
 12  Neighborhood                     1460 non-null   object 
 13  Condition1                       1460 non-null   object 
 14  Condition2                       1460 non-null   object 
 15  Type_of_dwelling                 1460 non-null   object 
 16  HouseStyle                       1460 non-null   object 
 17  OverallQual                      1460 non-null   int64  
 18  OverallCond                      1460 non-null   int64  
 19  YearBuilt                        1460 non-null   int64  
 20  YearRemodAdd                     1460 non-null   int64  
 21  RoofStyle                        1460 non-null   object 
 22  Roof_material                    1460 non-null   object 
 23  Exterior1st                      1460 non-null   object 
 24  Exterior2nd                      1460 non-null   object 
 25  Masonry_veneer_type              588 non-null    object 
 26  MasVnrArea                       1460 non-null   int64  
 27  ExterQual                        1460 non-null   object 
 28  ExterCond                        1460 non-null   object 
 29  Foundation                       1460 non-null   object 
 30  BsmtQual                         1460 non-null   object 
 31  BsmtCond                         1460 non-null   object 
 32  BsmtExposure                     1460 non-null   object 
 33  BsmtFinType1                     1460 non-null   object 
 34  BsmtFinSF1                       1460 non-null   int64  
 35  BsmtFinType2                     1460 non-null   object 
 36  BsmtFinSF2                       1460 non-null   int64  
 37  BsmtUnfSF                        1460 non-null   int64  
 38  TotalBsmtSF                      1460 non-null   int64  
 39  Heating                          1460 non-null   object 
 40  HeatingQC                        1460 non-null   object 
 41  CentralAir                       1460 non-null   object 
 42  Electrical_system                1459 non-null   object 
 43  1stFlrSF                         1460 non-null   int64  
 44  2ndFlrSF                         1460 non-null   int64  
 45  Low_quality_finished_square_feet 1460 non-null   int64  
 46  GrLivArea                        1460 non-null   int64  
 47  BsmtFullBath                     1460 non-null   int64  
 48  BsmtHalfBath                     1460 non-null   int64  
 49  FullBath                         1460 non-null   int64  
 50  HalfBath                         1460 non-null   int64  
 51  BedroomAbvGr                     1460 non-null   int64  
 52  KitchenAbvGr                     1460 non-null   int64  
 53  Kitchen_Quality                  1460 non-null   object 
 54  TotRmsAbvGrd                     1460 non-null   int64  
 55  Functional                       1460 non-null   object 
 56  Fireplaces                       1460 non-null   int64  
 57  FireplaceQu                      1460 non-null   object 
 58  Garage_location                  1460 non-null   object 
 59  GarageYrBlt                      1379 non-null   float64
 60  GarageFinish                     1460 non-null   object 
 61  GarageCars                       1460 non-null   int64  
 62  GarageArea                       1460 non-null   int64  
 63  GarageQual                       1460 non-null   object 
 64  Garage_condition                 1460 non-null   object 
 65  Paved driveway                   1460 non-null   object 
 66  WoodDeckSF                       1460 non-null   int64  
 67  OpenPorchSF                      1460 non-null   int64  
 68  EnclosedPorch                    1460 non-null   int64  
 69  3SsnPorch                        1460 non-null   int64  
 70  ScreenPorch                      1460 non-null   int64  
 71  PoolArea                         1460 non-null   int64  
 72  Pool_quality                     1460 non-null   object 
 73  Fence_Quality                    1460 non-null   object 
 74  Miscellaneous_feature            1460 non-null   object 
 75  miscellaneous_feature            1460 non-null   int64  
 76  MoSold                           1460 non-null   object 
 77  YrSold                           1460 non-null   int64  
 78  SaleType                         1460 non-null   object 
 79  SaleCondition                    1460 non-null   object 
 80  SalePrice                        1460 non-null   int64  
dtypes: float64(1), int64(35), object(45)
memory usage: 924.0+ KB
```

In [57]:
```
# Get the desciptive statistics summary of my data:
```

In [59]:
```
dataset.describe()
```

Out[59]:

	Unnamed: 0	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinS
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.0000
mean	729.500000	57.623288	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.117123	443.639726	46.5493
std	421.610009	34.664304	9981.264932	1.382997	1.112799	30.202904	20.645407	180.731373	456.098091	161.3192
min	0.000000	0.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	0.0000
25%	364.750000	42.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	0.0000
50%	729.500000	63.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	0.0000
75%	1094.250000	79.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	164.250000	712.250000	0.0000
max	1459.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.0000

8 rows × 36 columns

In [63]:
```
# Cleaning the data
```

In [65]:
```
# Check the Duplicates In the dateset:
```

In [67]:
```
dataset = dataset.drop_duplicates()
dataset.count()
```

Out[67]:
```
Unnamed: 0                      1460
Identifies_the_type_of_dwelling 1460
MSZoning                        1460
LotFrontage                     1460
LotArea                         1460
                                ...
MoSold                          1460
YrSold                          1460
SaleType                        1460
SaleCondition                   1460
SalePrice                       1460
Length: 81, dtype: int64
```

In [69]:
```
# Check the Null-values:
```

In [71]:
```
dataset.isnull().sum().iloc[51:81]
```

Out[71]:
```
BedroomAbvGr              0
KitchenAbvGr              0
Kitchen_Quality           0
TotRmsAbvGrd              0
Functional                0
Fireplaces                0
FireplaceQu               0
Garage_location           0
GarageYrBlt              81
GarageFinish              0
GarageCars                0
GarageArea                0
GarageQual                0
Garage_condition          0
Paved driveway            0
WoodDeckSF                0
OpenPorchSF               0
EnclosedPorch             0
3SsnPorch                 0
ScreenPorch               0
PoolArea                  0
Pool_quality              0
Fence_Quality             0
Miscellaneous_feature     0
miscellaneous_feature     0
MoSold                    0
YrSold                    0
SaleType                  0
SaleCondition             0
SalePrice                 0
dtype: int64
```

In [73]:
```
# Drop the missing values & Columns:
```

In [77]:
```
dataset['Alley'].isnull().mean()*100
```

Out[77]:
```
93.76712328767123
```

In [81]:
```
dataset['GarageYrBlt'].isnull().mean()*100
```

Out[81]: 5.5479452054794525

In [83]:
```
dataset['Electrical_system'].isnull().mean()*100
```

Out[83]: 0.0684931506849315

In [85]:
```
# Check the Null-Values in columns:
```

In [87]:
```
dataset[dataset['Electrical_system'].isnull()].iloc[:,30:50]
```

Out[87]:

	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	Heati
1379	Gd	TA	No	Unf	0	Unf	0	384	384	GasA	

In [91]:
```
dataset['Electrical_system'].value_counts()
```

Out[91]:
```
Electrical_system
SBrkr    1334
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: count, dtype: int64
```

In [93]:
```
dataset[dataset['GarageYrBlt'].isnull()].iloc[:,30:50]
```

Out[93]:

	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	Heati
39	No	No	No	No	0	No	0	0	0	GasA	
48	TA	TA	No	Unf	0	Unf	0	736	736	GasA	
78	TA	TA	No	Unf	0	Unf	0	1768	1768	GasA	
88	TA	Fa	No	Unf	0	Unf	0	1013	1013	GasA	
89	Gd	TA	No	GLQ	588	Unf	0	402	990	GasA	
…	…	…	…	…	…	…	…	…	…	…	
1349	TA	Fa	No	LwQ	259	Unf	0	425	684	OthW	
1407	TA	TA	No	ALQ	625	Unf	0	208	833	GasA	
1449	Gd	TA	Av	GLQ	553	Unf	0	77	630	GasA	
1450	Gd	TA	No	Unf	0	Unf	0	896	896	GasA	
1453	Gd	TA	No	Unf	0	Unf	0	1140	1140	GasA	

81 rows × 20 columns

In [95]:
```
dataset['GarageYrBlt'].value_counts()
```

Out[95]:
```
GarageYrBlt
2005.0    65
2006.0    59
2004.0    53
2003.0    50
2007.0    49
          ..
1927.0     1
1900.0     1
1906.0     1
1908.0     1
1933.0     1
Name: count, Length: 97, dtype: int64
```

In [97]:
```
# Replace the Null-Values with the most occuring value:
```

In [99]:
```
dataset['Electrical_system']=dataset['Electrical_system'].fillna('SBrkr')
```

In [101…
```
dataset['GarageYrBlt']= dataset['GarageYrBlt'].fillna(2005.0)
```

In [103…
```
dataset['Electrical_system'].value_counts()
```

Out[103…
```
Electrical_system
SBrkr    1335
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: count, dtype: int64
```

In [105…
```
dataset['GarageYrBlt'].value_counts()
```

Out[105…
```
GarageYrBlt
2005.0    146
2006.0     59
2004.0     53
2003.0     50
2007.0     49
          ...
1927.0      1
1900.0      1
1906.0      1
1908.0      1
1933.0      1
Name: count, Length: 97, dtype: int64
```

In [107…
```
# Removing of Null-Values:
```

In [109…
```
dataset[['Electrical_system','GarageYrBlt']].isnull().sum()
```

Out[109…
```
Electrical_system    0
GarageYrBlt          0
dtype: int64
```

In [111…
```
dataset.isnull().sum()
```

Out[111…
```
Unnamed: 0                      0
Identifies_the_type_of_dwelling    0
MSZoning                        0
LotFrontage                     0
LotArea                         0
                               ..
MoSold                          0
YrSold                          0
SaleType                        0
SaleCondition                   0
SalePrice                       0
Length: 81, dtype: int64
```

In [113…
```
dataset.shape
```

Out[113…
```
(1460, 81)
```

In [115…
```
# Drop unnecessary Columns:
```

In [117…
```
dataset= dataset.drop(['Alley','Masonry_veneer_type','Unnamed: 0'], axis=1)
```

In [119…
```
dataset
```

Out[119…

	Identifies_the_type_of_dwelling	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	Lot_configuration	LandSlop
0	SC60	RL	65	8450	Pave	Reg	Lvl	AllPub	Inside	G
1	SC20	RL	80	9600	Pave	Reg	Lvl	AllPub	FR2	G
2	SC60	RL	68	11250	Pave	IR1	Lvl	AllPub	Inside	G
3	SC70	RL	60	9550	Pave	IR1	Lvl	AllPub	Corner	G
4	SC60	RL	84	14260	Pave	IR1	Lvl	AllPub	FR2	G
...	...	...	...	...	...	...	...	...	...	
1455	SC60	RL	62	7917	Pave	Reg	Lvl	AllPub	Inside	G
1456	SC20	RL	85	13175	Pave	Reg	Lvl	AllPub	Inside	G
1457	SC70	RL	66	9042	Pave	Reg	Lvl	AllPub	Inside	G
1458	SC20	RL	68	9717	Pave	Reg	Lvl	AllPub	Inside	G
1459	SC20	RL	75	9937	Pave	Reg	Lvl	AllPub	Inside	G

1460 rows × 78 columns

In [121…
```
# Convert float column into int64:
```

In [123…
```
dataset['GarageYrBlt']=dataset['GarageYrBlt'].replace(to_replace=np.nan,value=0).astype('int64')
```

In [125…
```
dataset['GarageYrBlt'].value_counts
```

Out[125…
```
<bound method IndexOpsMixin.value_counts of 0       2003
1       1976
2       2001
3       1998
4       2000
        ...
1455    1999
1456    1978
1457    1941
1458    1950
1459    1965
Name: GarageYrBlt, Length: 1460, dtype: int64>
```

In [127…
```
dataset['GarageYrBlt'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 1460 entries, 0 to 1459
Series name: GarageYrBlt
Non-Null Count  Dtype
--------------  -----
1460 non-null   int64
dtypes: int64(1)
memory usage: 11.5 KB
```

In [129…
```
dataset['GarageYrBlt'].isnull().sum()
```

Out[129… 0

In [131…
```
dataset.shape
```

Out[131… (1460, 78)

In [133…
```
# Check the Duplicate Entries:
```

In [135…
```
dataset.duplicated().sum()
```

Out[135… 0

In [137…
```
# Replaced the values in the 'Identifies the type of dwelling' column:
```

In [139…
```
dataset['Identifies_the_type_of_dwelling'].value_counts()
```

Out[139…
```
Identifies_the_type_of_dwelling
SC20     536
SC60     299
SC50     144
SC120     87
SC30      69
SC160     63
SC70      60
SC80      58
SC90      52
SC190     30
SC85      20
SC75      16
SC45      12
SC180     10
SC40       4
Name: count, dtype: int64
```

In [145…
```
dataset['Identifies_the_type_of_dwelling']= dataset['Identifies_the_type_of_dwelling'].replace({'SC60':'2-STORY 1946 & NEWER','SC
                                                                                                'SC190':'2 FAMILY CONVERSION - ALL STYLES AND AGES'
```

In [147…
```
dataset.sample(3)
```

Out[147…

	Identifies_the_type_of_dwelling	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	Lot_configuration	LandSlop
1288	1-STORY PUD (Planned Unit Development) - 1946 …	RL	40	5664	Pave	IR1	Lvl	AllPub	Inside	G
1026	1-STORY 1946 & NEWER ALL STYLES	RL	73	9300	Pave	Reg	Lvl	AllPub	Inside	G
252	2-STORY 1946 & NEWER	RL	65	8366	Pave	IR1	Lvl	AllPub	Inside	G

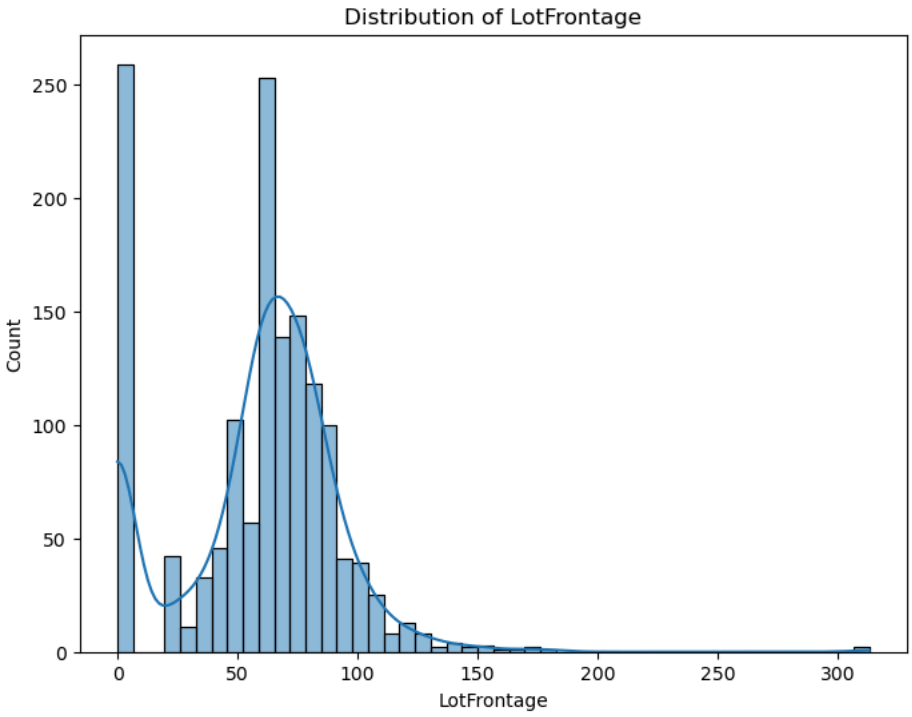
3 rows × 78 columns

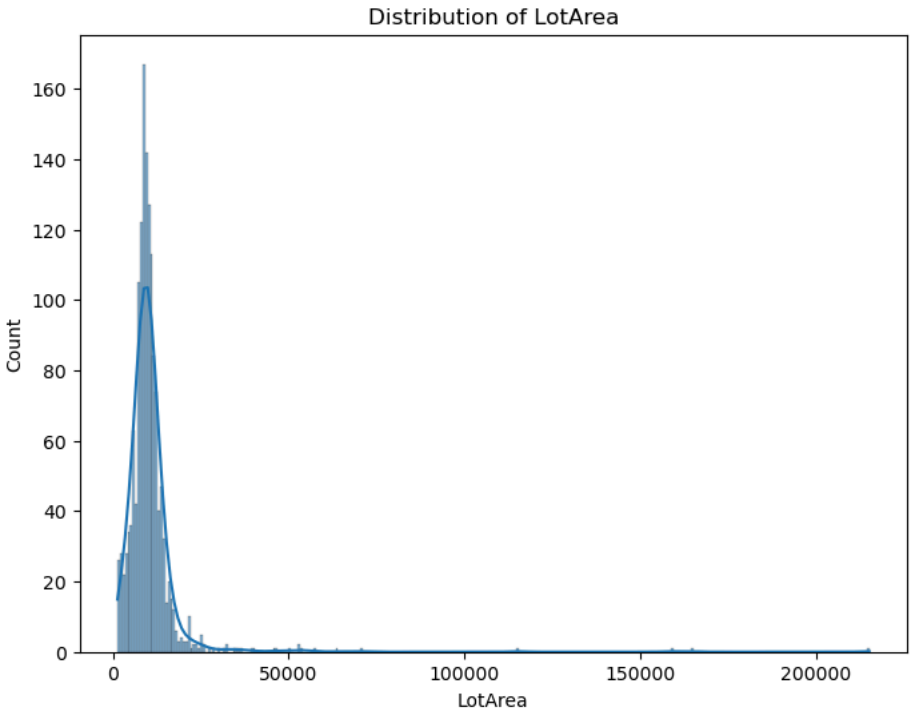
In [151…
```
# Univariate Analysis:
```

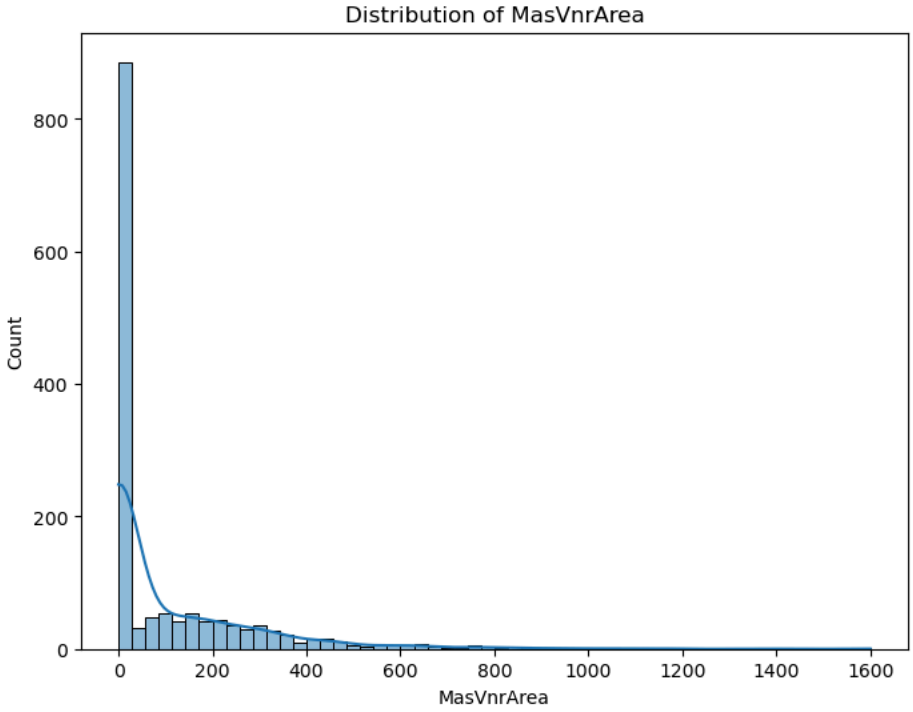
In [153…
```
# Numeric variables:
```

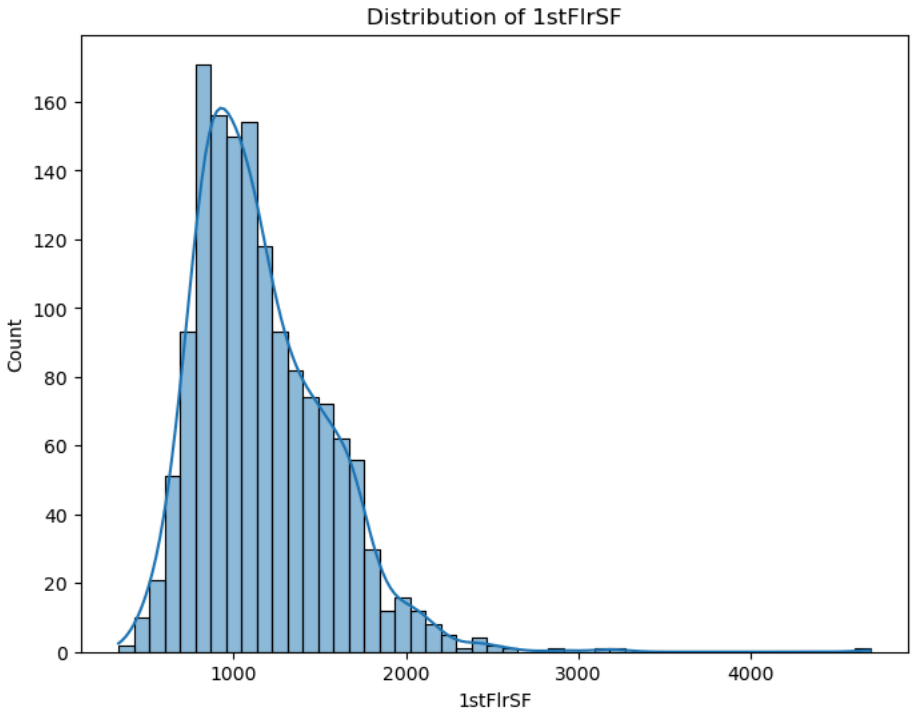
In [163…
```
numeric_vars = ['LotFrontage', 'LotArea', 'MasVnrArea', '1stFlrSF', 'GrLivArea']
for var in numeric_vars:
    plt.figure(figsize=(8, 6))
    sns.histplot(data= dataset, x=var, kde=True)
```

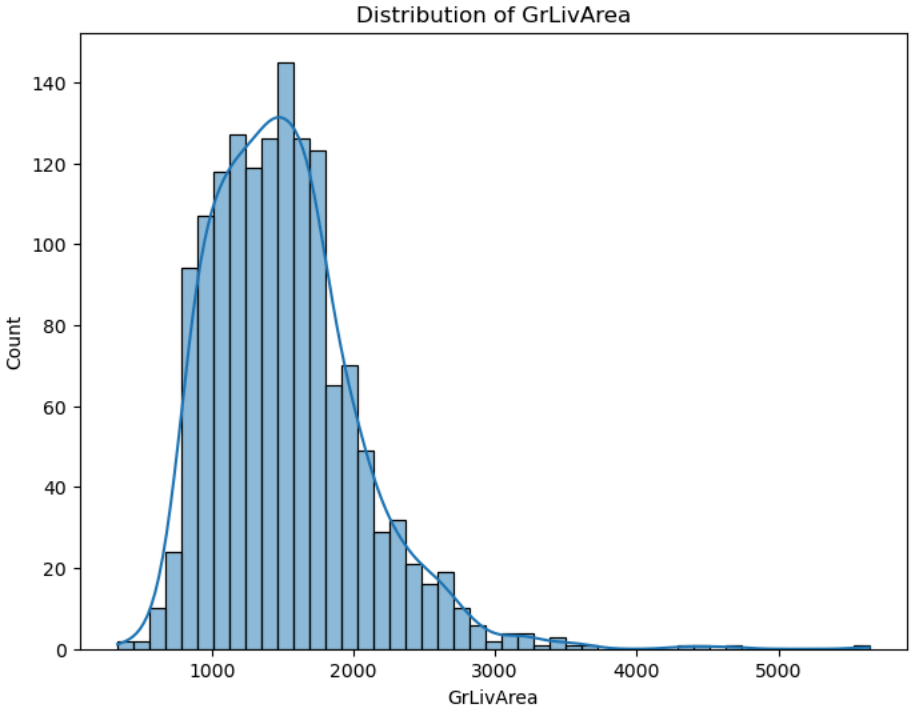
```
    plt.title(f'Distribution of {var}')
    plt.show()
```





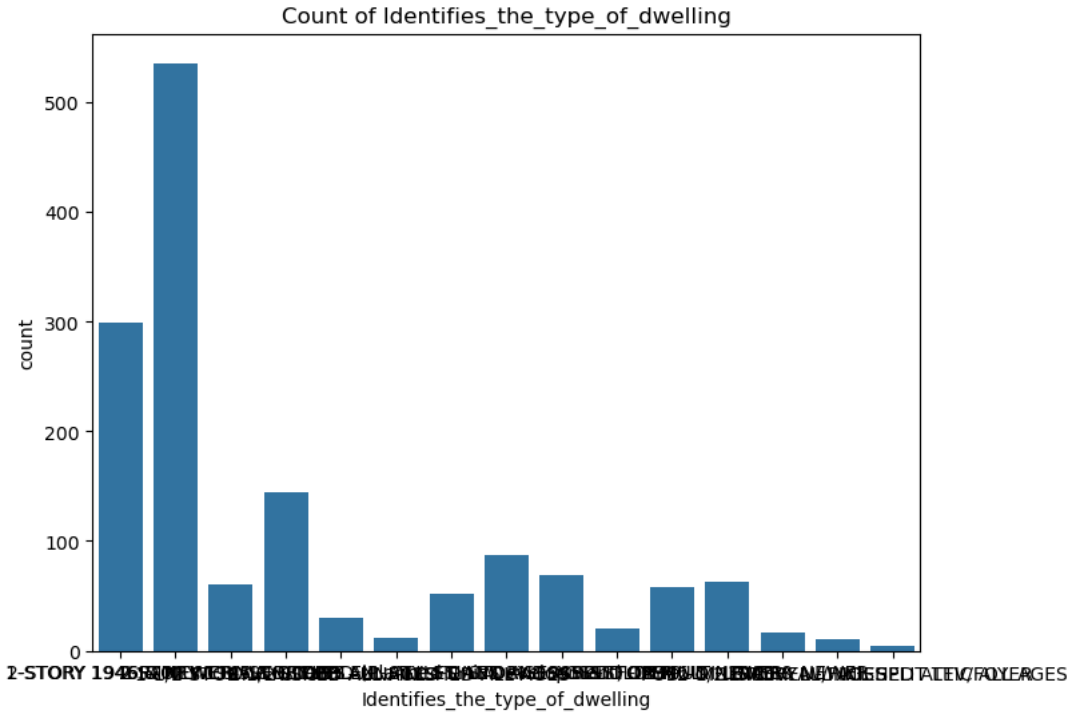

Distribution of MasVnrArea
Count
MasVnrArea
0
200
400
600
800
1000
1200
1400
1600

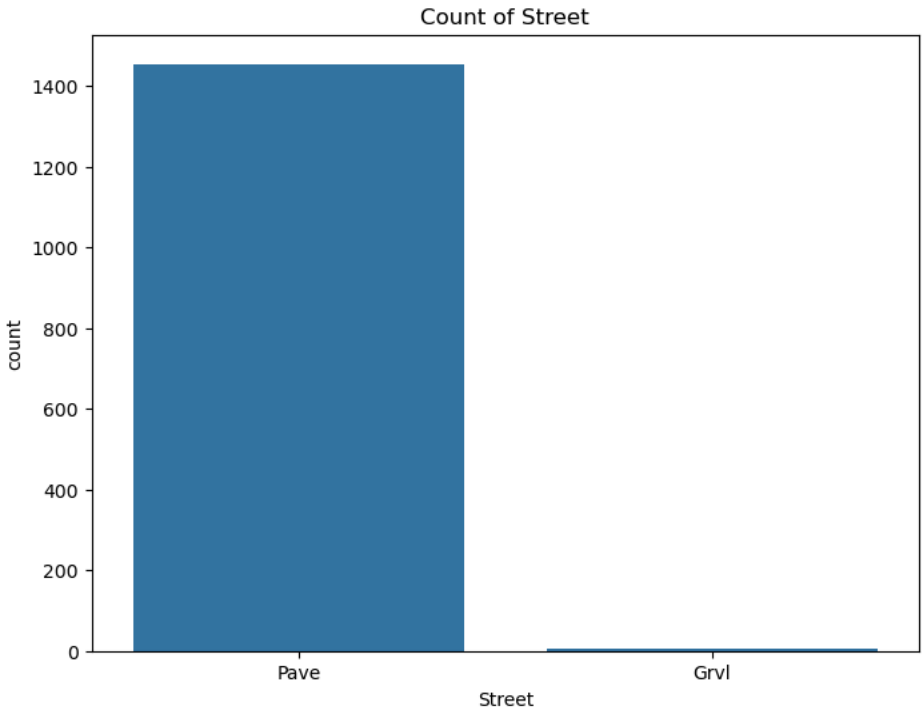

Distribution of 1stFlrSF
Count
1stFlrSF
0
20
40
60
80
100
120
140
160
1000
2000
3000
4000

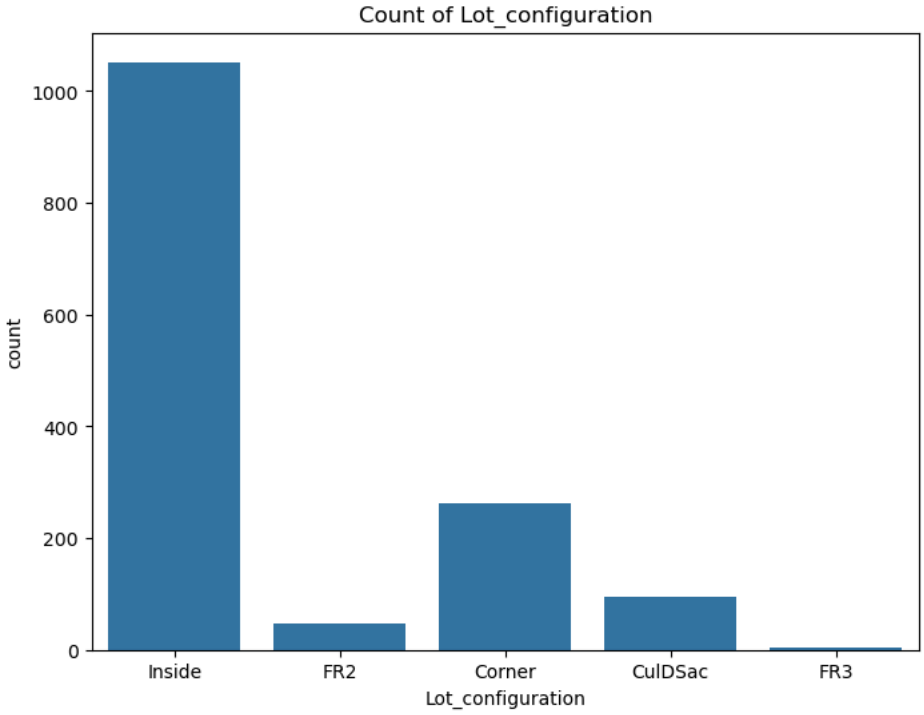


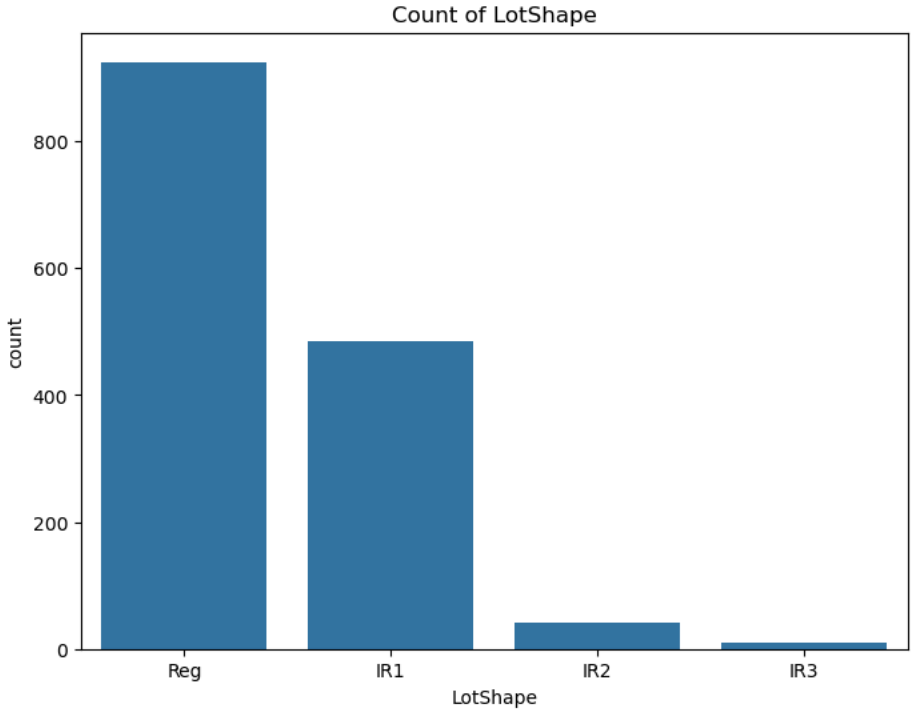
In [165…

```
categorical_vars = ['Identifies_the_type_of_dwelling', 'Street', 'Lot_configuration', 'LotShape']
for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(data=dataset, x=var)
    plt.title(f'Count of {var}')
    plt.show()
```



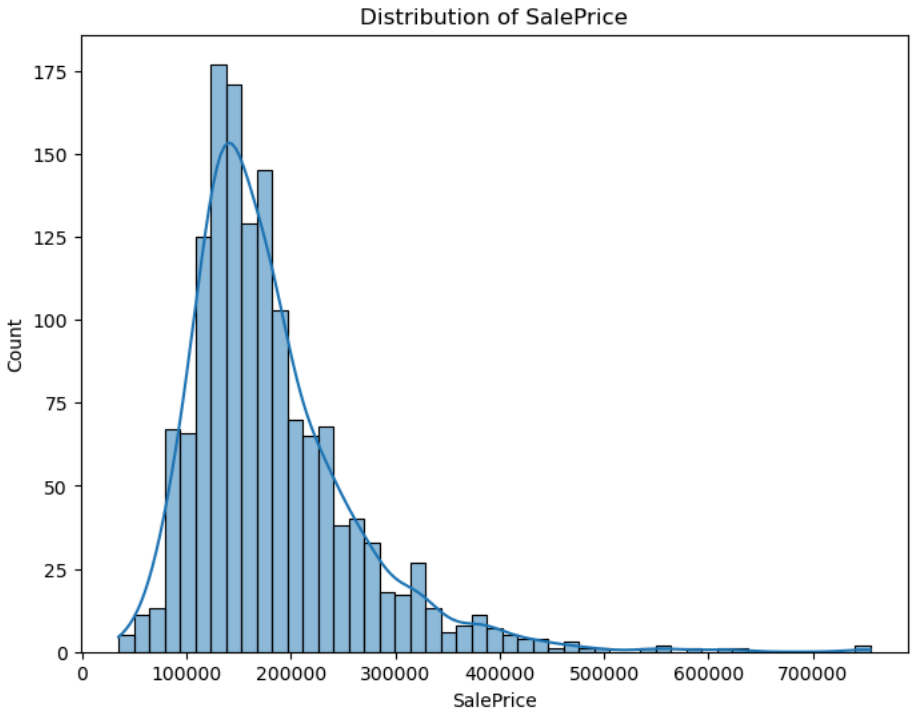

Count of Street
count
1400
1200
1000
800
600
400
200
0
Pave
Grvl
Street


Count of Lot_configuration
count
1000
800
600
400
200
0
Inside
FR2
Corner
CulDSac
FR3
Lot_configuration



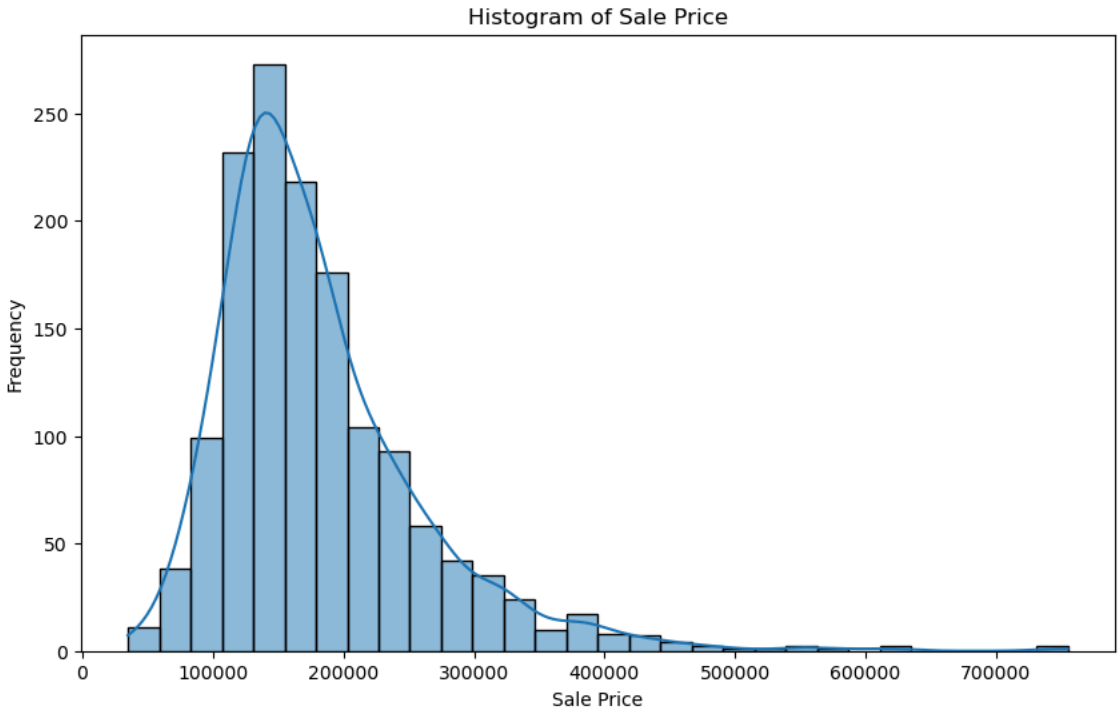
In [167…
```
# Sale-price distribution:
```

In [169…
```
plt.figure(figsize=(8, 6))
sns.histplot(data=dataset, x='SalePrice', kde=True)
plt.title('Distribution of SalePrice')
plt.show()
```



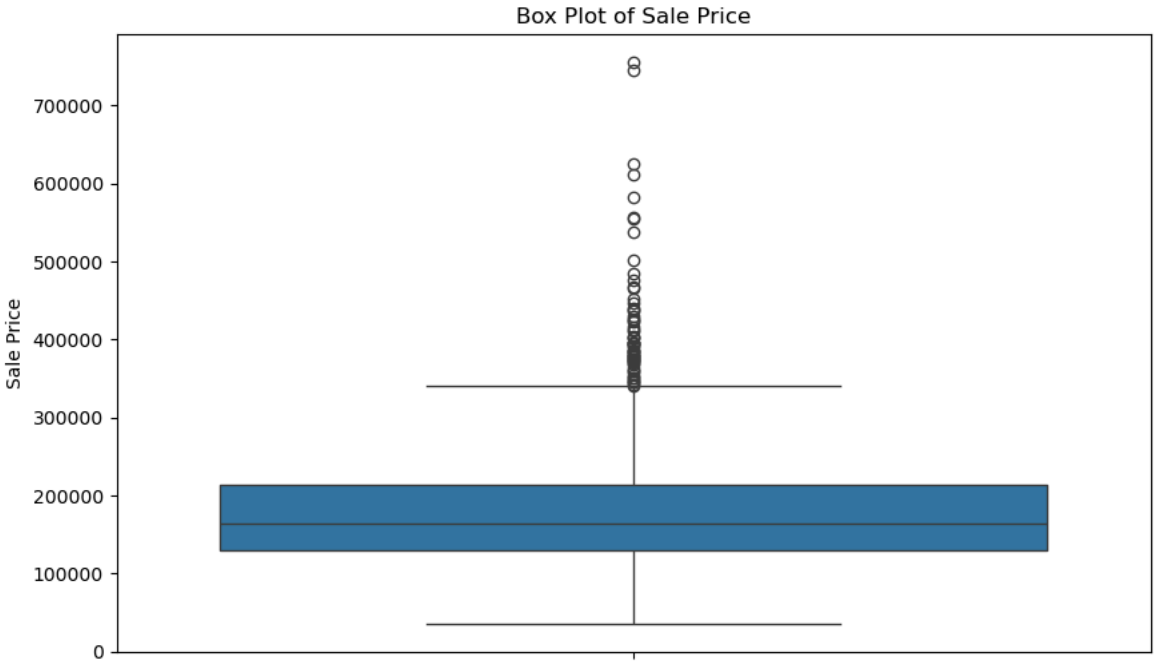
In [171…
```
# Histogram:
```

In [173…
```
plt.figure(figsize=(10, 6))
sns.histplot(data=dataset, x='SalePrice', bins=30, kde=True)
plt.xlabel('Sale Price')
plt.ylabel('Frequency')
plt.title('Histogram of Sale Price')
plt.show()
```



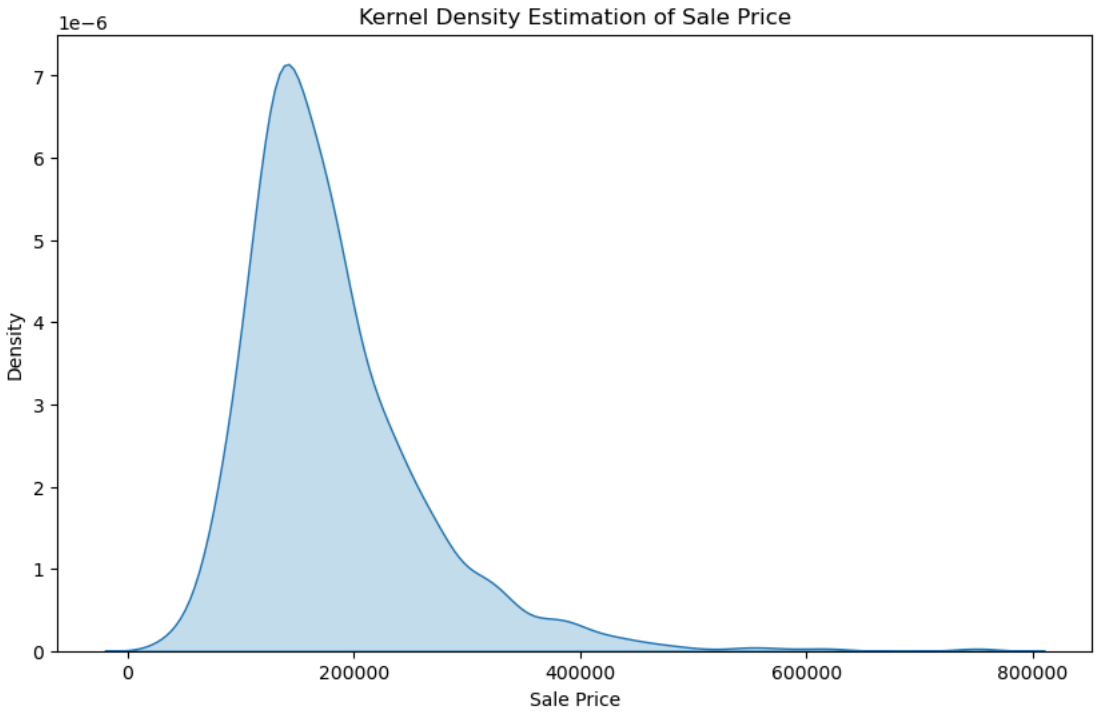
In [175…
```
# Box-plot:
```

In [177…
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=dataset, y='SalePrice')
plt.ylabel('Sale Price')
plt.title('Box Plot of Sale Price')
plt.show()
```



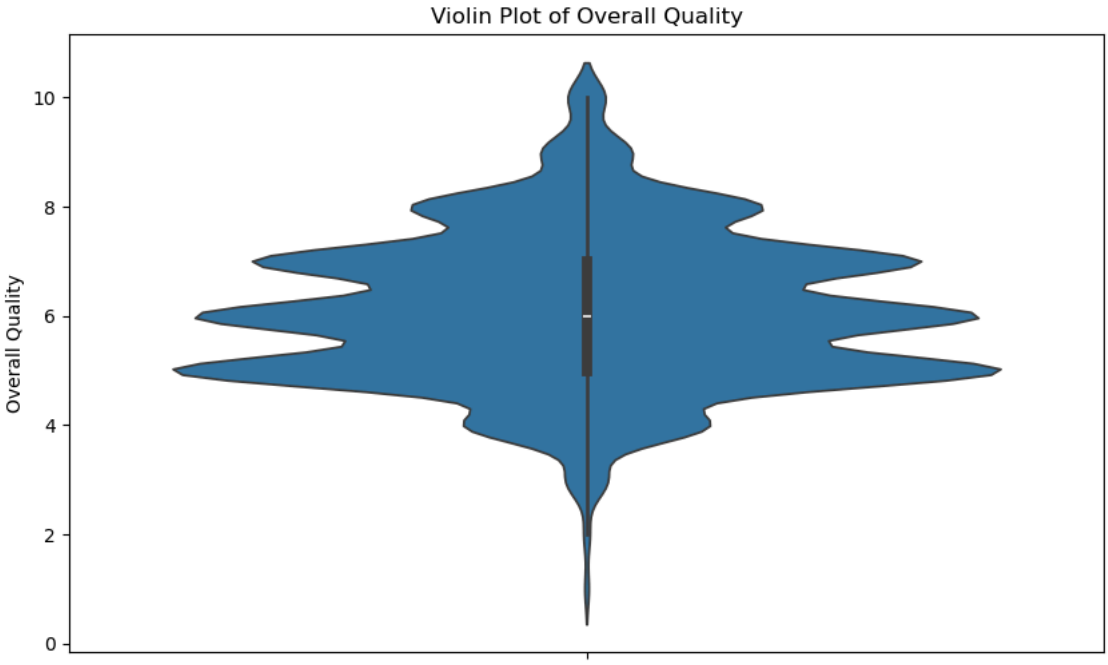
In [179…
```
# Kernel Density Estimation:
```

In [181…
```
plt.figure(figsize=(10, 6))
sns.kdeplot(data=dataset['SalePrice'], shade=True)
plt.xlabel('Sale Price')
plt.ylabel('Density')
plt.title('Kernel Density Estimation of Sale Price')
plt.show()
```



In [183…
```
# Violin-plot:
```

In [185…
```
plt.figure(figsize=(10, 6))
sns.violinplot(data=dataset, y='OverallQual')
plt.ylabel('Overall Quality')
plt.title('Violin Plot of Overall Quality')
plt.show()
```



In [189…
```
# Multivariate Analysis:
```

In [193…
```
exclude_dataset=dataset.select_dtypes(exclude=['object'])
```

In [195…
```
exclude_dataset
```

Out[195…

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	...	Gar
0	65	8450	7	5	2003	2003	196	706	0	150	...	
1	80	9600	6	8	1976	1976	0	978	0	284	...	
2	68	11250	7	5	2001	2002	162	486	0	434	...	
3	60	9550	7	5	1915	1970	0	216	0	540	...	
4	84	14260	8	5	2000	2000	350	655	0	490	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1455	62	7917	6	5	1999	2000	0	0	0	953	...	
1456	85	13175	6	6	1978	1988	119	790	163	589	...	
1457	66	9042	7	9	1941	2006	0	275	0	877	...	
1458	68	9717	5	6	1950	1996	0	49	1029	0	...	
1459	75	9937	5	6	1965	1965	0	830	290	136	...	

1460 rows × 35 columns

In [197…
```
exclude_dataset.columns
```

Out[197…
```
Index(['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
       'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
       'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'Low_quality_finished_square_feet', 'GrLivArea', 'BsmtFullBath',
       'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
       'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea',
       'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'miscellaneous_feature', 'YrSold',
       'SalePrice'],
      dtype='object')
```

In [199…
```
# Compute the correlation matrices:
```

In [201…
```
corr_matrix = exclude_dataset.corr()
```

In [203…
```
corr_matrix
```

Out[203…

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	Bsm
LotFrontage	1.000000	0.100739	0.176561	-0.053457	0.036853	0.078686	0.105010	0.076670	-0
LotArea	0.100739	1.000000	0.105806	-0.005636	0.014228	0.013788	0.103321	0.214103	0
OverallQual	0.176561	0.105806	1.000000	-0.091932	0.572323	0.550684	0.407252	0.239666	-0
OverallCond	-0.053457	-0.005636	-0.091932	1.000000	-0.375983	0.073741	-0.125694	-0.046231	0
YearBuilt	0.036853	0.014228	0.572323	-0.375983	1.000000	0.592855	0.311600	0.249503	-0
YearRemodAdd	0.078686	0.013788	0.550684	0.073741	0.592855	1.000000	0.176529	0.128451	-0
MasVnrArea	0.105010	0.103321	0.407252	-0.125694	0.311600	0.176529	1.000000	0.261256	-0
BsmtFinSF1	0.076670	0.214103	0.239666	-0.046231	0.249503	0.128451	0.261256	1.000000	-0
BsmtFinSF2	-0.009312	0.111170	-0.059119	0.040229	-0.049107	-0.067759	-0.071330	-0.050117	1
BsmtUnfSF	0.160829	-0.002618	0.308159	-0.136841	0.149040	0.181133	0.113862	-0.495251	-0
TotalBsmtSF	0.238274	0.260833	0.537808	-0.171098	0.391452	0.291066	0.360067	0.522396	0
1stFlrSF	0.245181	0.299475	0.476224	-0.144203	0.281986	0.240379	0.339850	0.445863	0
2ndFlrSF	0.042549	0.050986	0.295493	0.028942	0.010308	0.140024	0.173800	-0.137079	-0
Low_quality_finished_square_feet	0.049981	0.004779	-0.030429	0.025494	-0.183784	-0.062419	-0.068628	-0.064503	0
GrLivArea	0.220347	0.263116	0.593007	-0.079686	0.199010	0.287389	0.388052	0.208171	-0
BsmtFullBath	0.010514	0.158155	0.111098	-0.054942	0.187599	0.119470	0.083010	0.649212	0
BsmtHalfBath	-0.027856	0.048046	-0.040150	0.117821	-0.038162	-0.012337	0.027403	0.067418	0
FullBath	0.120548	0.126031	0.550600	-0.194149	0.468271	0.439046	0.272999	0.058543	-0
HalfBath	-0.012952	0.014259	0.273458	-0.060769	0.242656	0.183331	0.199108	0.004262	-0
BedroomAbvGr	0.144494	0.119690	0.101676	0.012980	-0.070651	-0.040581	0.102775	-0.107355	-0
KitchenAbvGr	0.034425	-0.017784	-0.183882	-0.087001	-0.174800	-0.149598	-0.038450	-0.081007	-0
TotRmsAbvGrd	0.221396	0.190015	0.427452	-0.057583	0.095589	0.191740	0.279568	0.044316	-0
Fireplaces	0.044018	0.271364	0.396765	-0.023820	0.147716	0.112581	0.247015	0.260011	0
GarageYrBlt	0.037767	-0.042198	0.437998	-0.299203	0.700098	0.571369	0.209289	0.119267	-0
GarageCars	0.165229	0.154871	0.600671	-0.185758	0.537850	0.420622	0.361945	0.224054	-0
GarageArea	0.201473	0.180403	0.562022	-0.151521	0.478954	0.371600	0.370884	0.296970	-0
WoodDeckSF	-0.016780	0.171698	0.238923	-0.003334	0.224880	0.205726	0.159991	0.204306	0
OpenPorchSF	0.069605	0.084774	0.308819	-0.032589	0.188686	0.226298	0.122528	0.111761	0
EnclosedPorch	0.027366	-0.018340	-0.113937	0.070356	-0.387268	-0.193919	-0.109907	-0.102303	0
3SsnPorch	0.023499	0.020423	0.030371	0.025504	0.031355	0.045286	0.019144	0.026451	-0
ScreenPorch	0.022969	0.043160	0.064886	0.054811	-0.050364	-0.038740	0.062248	0.062021	0
PoolArea	0.114106	0.077672	0.065166	-0.001985	0.004950	0.005829	0.011928	0.140491	0
miscellaneous_feature	-0.059606	0.038068	-0.031406	0.068777	-0.034383	-0.010286	-0.029512	0.003571	0
YrSold	-0.012094	-0.014261	-0.027347	0.043950	-0.013618	0.035743	-0.008317	0.014359	0
SalePrice	0.209624	0.263843	0.790982	-0.077856	0.522897	0.507101	0.472614	0.386420	-0

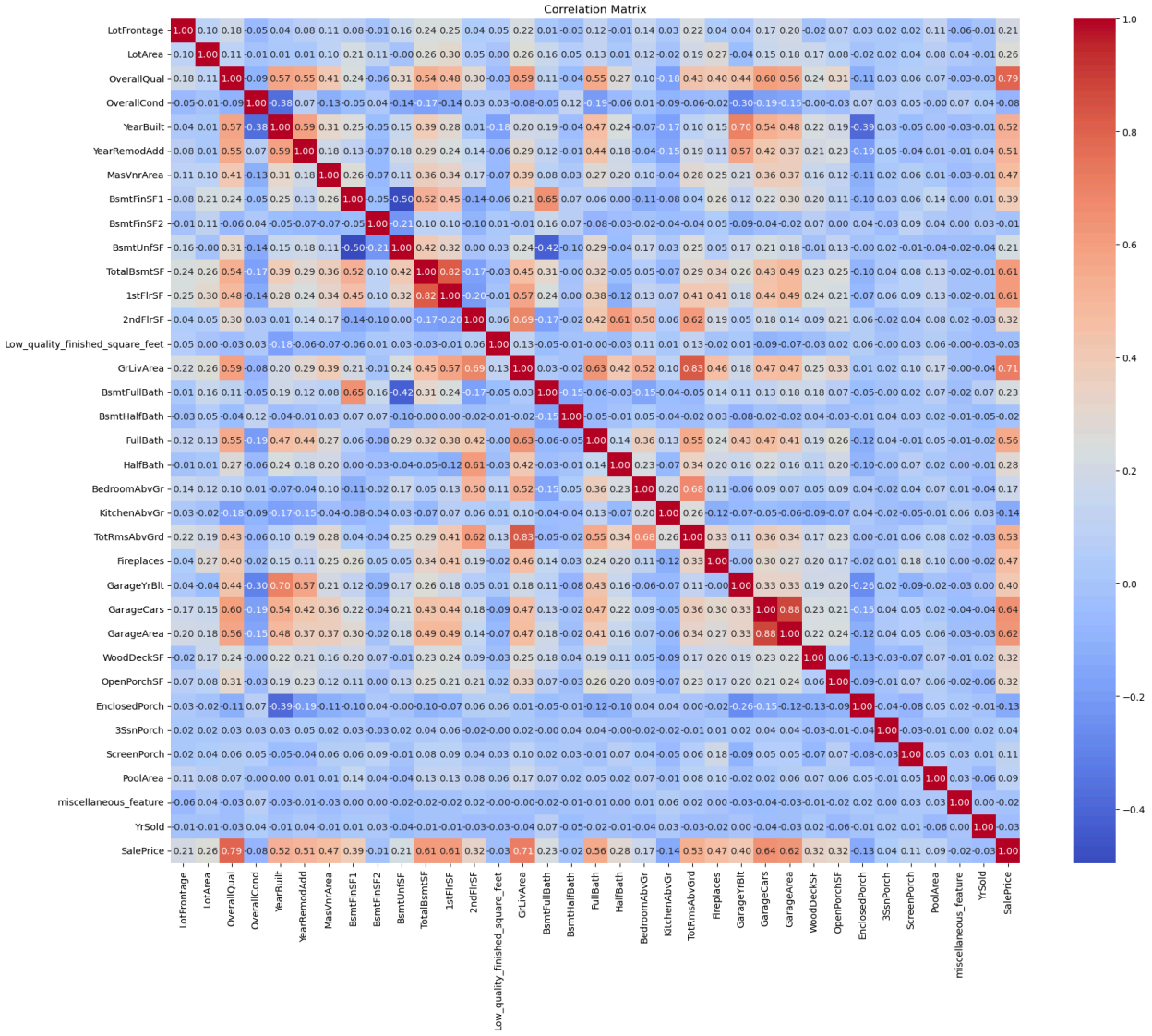
35 rows × 35 columns

In [205…

```
# Heat-map:
```

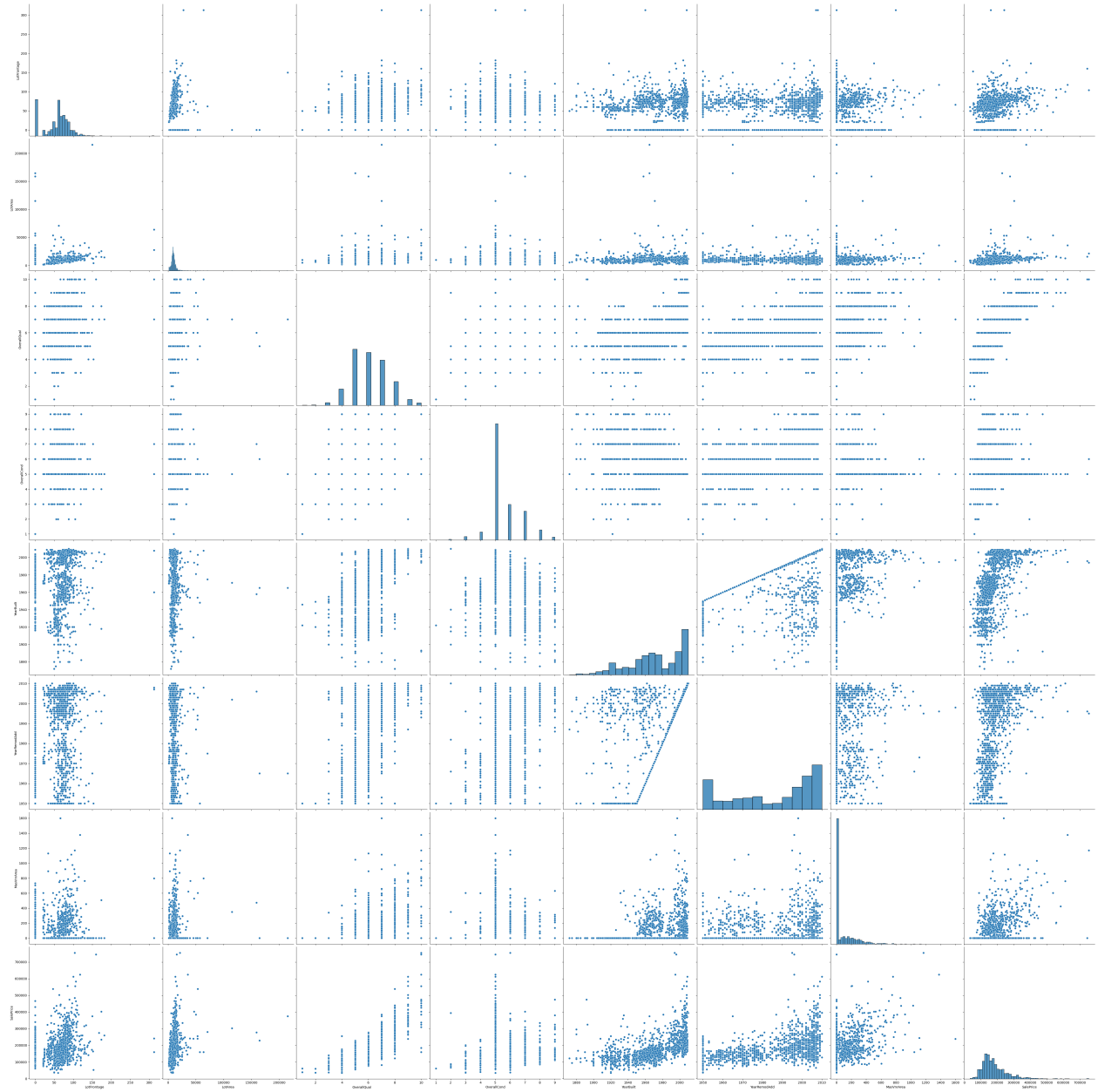
In [207…

```
plt.figure(figsize=(20, 16))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



In [209…
```
# Pair-plot:
```

In [216…
```
sns.pairplot(exclude_dataset[['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
       'YearRemodAdd', 'MasVnrArea','SalePrice']], height=6)
plt.show()
```



In [218…
```
# Feature Engineering:
```

In [220…
```
dataset.head()
```

Out[220…

	Identifies_the_type_of_dwelling	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	Lot_configuration	LandSlope
0	2-STORY 1946 & NEWER	RL	65	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl
1	1-STORY 1946 & NEWER ALL STYLES	RL	80	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl
2	2-STORY 1946 & NEWER	RL	68	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl
3	2-STORY 1945 & OLDER	RL	60	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl
4	2-STORY 1946 & NEWER	RL	84	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl

5 rows × 78 columns

In [222…
```
dataset.columns
```

Out[222…
```
Index(['Identifies_the_type_of_dwelling', 'MSZoning', 'LotFrontage', 'LotArea',
       'Street', 'LotShape', 'LandContour', 'Utilities', 'Lot_configuration',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'Type_of_dwelling', 'HouseStyle', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'Roof_material',
       'Exterior1st', 'Exterior2nd', 'MasVnrArea', 'ExterQual', 'ExterCond',
       'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
       'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical_system', '1stFlrSF',
       '2ndFlrSF', 'Low_quality_finished_square_feet', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'Kitchen_Quality', 'TotRmsAbvGrd', 'Functional',
       'Fireplaces', 'FireplaceQu', 'Garage_location', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'Garage_condition', 'Paved driveway', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Pool_quality',
       'Fence_Quality', 'Miscellaneous_feature', 'miscellaneous_feature',
       'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [224…
```
dataset.shape
```

Out[224…
```
(1460, 78)
```

In [226…
```
# Calculate the price per square foot:
```

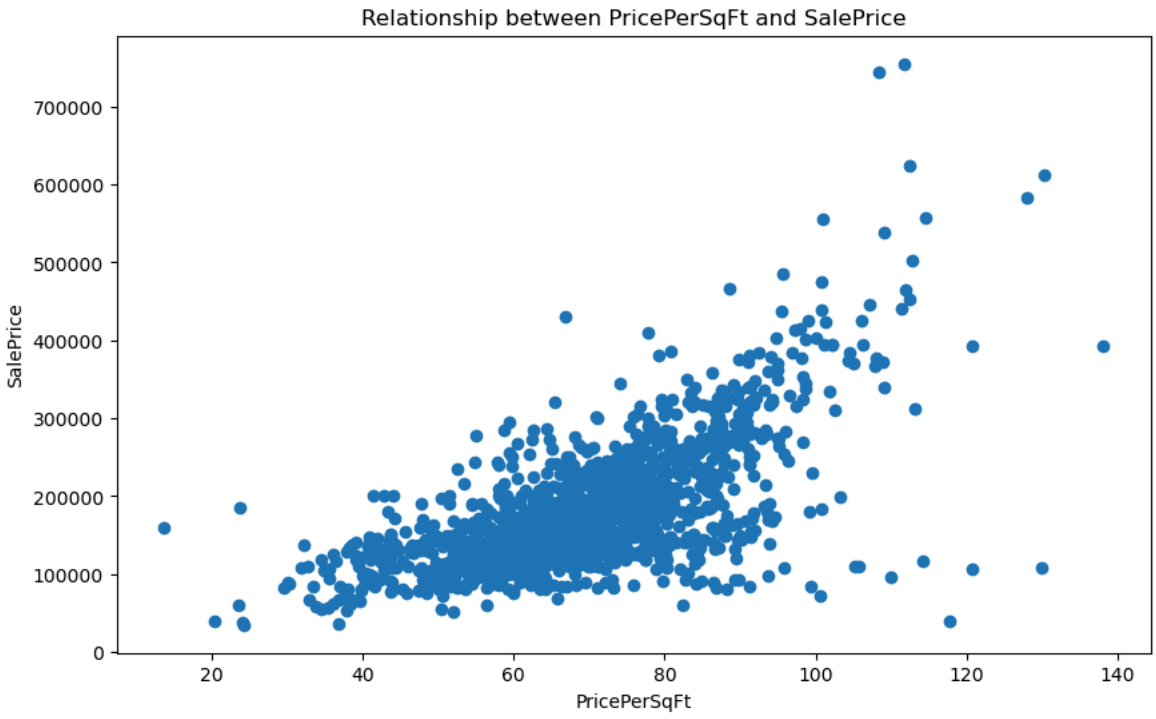
In [238…
```
dataset['PricePerSqFt'] = dataset['SalePrice'] / (dataset['GrLivArea'] +
                                                  dataset['TotalBsmtSF'])
```

In [240…
```
dataset['PricePerSqFt']
```

Out[240…
```
0       81.254871
1       71.909667
2       82.594235
3       56.611403
4       74.783129
          ...
1455    67.307692
1456    58.091286
1457    76.317297
1458    65.920686
1459    58.718153
Name: PricePerSqFt, Length: 1460, dtype: float64
```

In [242…
```
# Visualizing the New column created:
```

In [244…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['PricePerSqFt'],dataset['SalePrice'])
plt.xlabel('PricePerSqFt')
plt.ylabel('SalePrice')
plt.title('Relationship between PricePerSqFt and SalePrice')
plt.show()
```



In [246…
```
# Calculate the age of property:
```

In [248…
```
dataset['AgeAtSale'] = dataset['YrSold'] - dataset['YearBuilt']
```

In [250…
```
dataset['AgeAtSale']
```

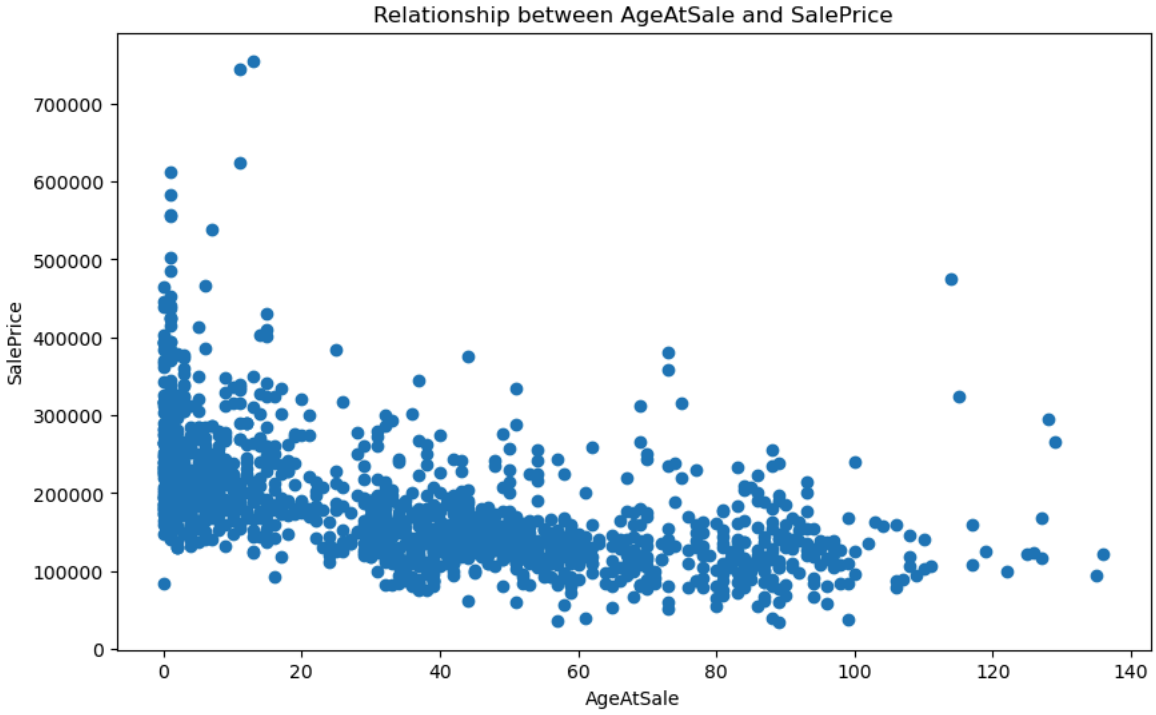
Out[250…
```
0        5
1       31
2        7
3       91
4        8
       ..
1455     8
1456    32
1457    69
1458    60
1459    43
Name: AgeAtSale, Length: 1460, dtype: int64
```

In [252…
```
# Dropping the Unwanted columns:
```

In [254…
```
dataset.drop(columns=['YearBuilt'],inplace=True)
```

In [256…
```
# Visualizing the New column created:
```

In [258…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['AgeAtSale'],dataset['SalePrice'])
plt.xlabel('AgeAtSale')
plt.ylabel('SalePrice')
plt.title('Relationship between AgeAtSale and SalePrice')
plt.show()
```



In [260…
```
# Calculate the total square footage:
```

In [262…
```
dataset['TotalSqFt'] = dataset['1stFlrSF'] + dataset['2ndFlrSF'] + dataset['TotalBsmtSF']
```

In [264…
```
dataset['TotalSqFt']
```

Out[264…
```
0       2566
1       2524
2       2706
3       2473
4       3343
       ...
1455    2600
1456    3615
1457    3492
1458    2156
1459    2512
Name: TotalSqFt, Length: 1460, dtype: int64
```

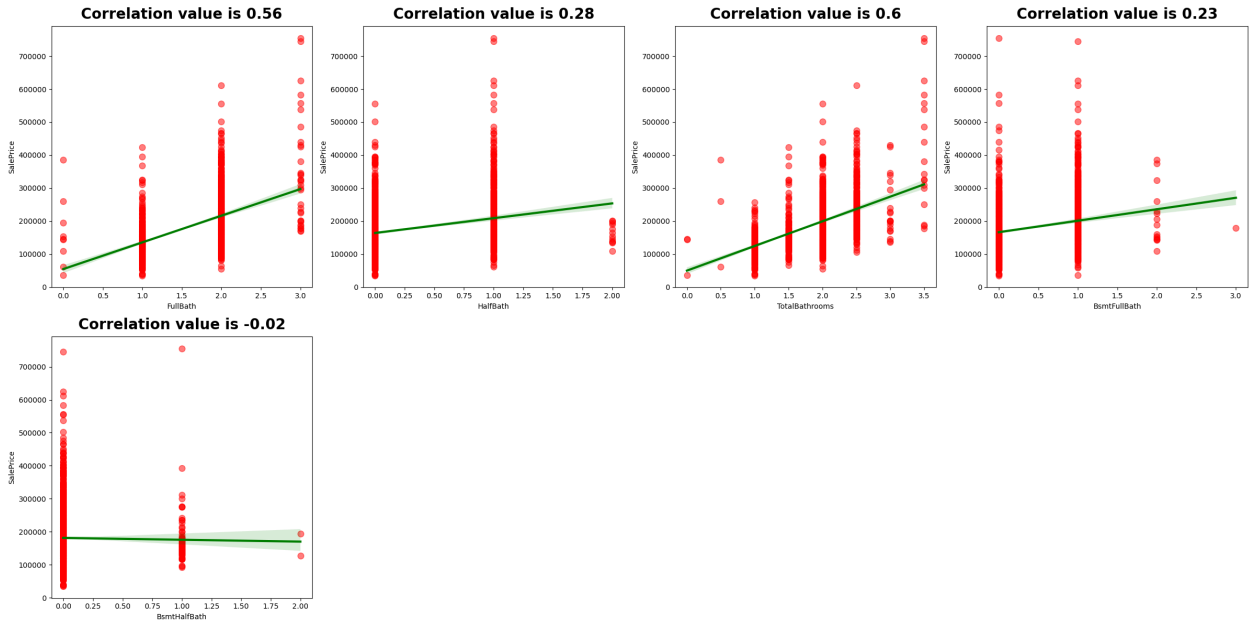
In [266…
```
# Calculate the total number of bathrooms:
```

In [268…
```
dataset['TotalBathrooms'] = dataset['FullBath'] + 0.5 * dataset['HalfBath']
+ dataset['BsmtFullBath'] + 0.5 * dataset['BsmtHalfBath']
```

Out[268…
```
0       1.0
1       0.5
2       1.0
3       1.0
4       1.0
       ...
1455    0.0
1456    1.0
1457    0.0
1458    1.0
1459    1.0
Length: 1460, dtype: float64
```

In [270…
```
# Correlation between Feature-Column and Target-Column:
```

In [280…
```
cols = ['FullBath', 'HalfBath','TotalBathrooms', 'BsmtFullBath', 'BsmtHalfBath']

plt.figure(figsize=(24, 12))
for index, column in enumerate(cols):
    plt.subplot(2, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="black")
plt.tight_layout()
plt.show()
```

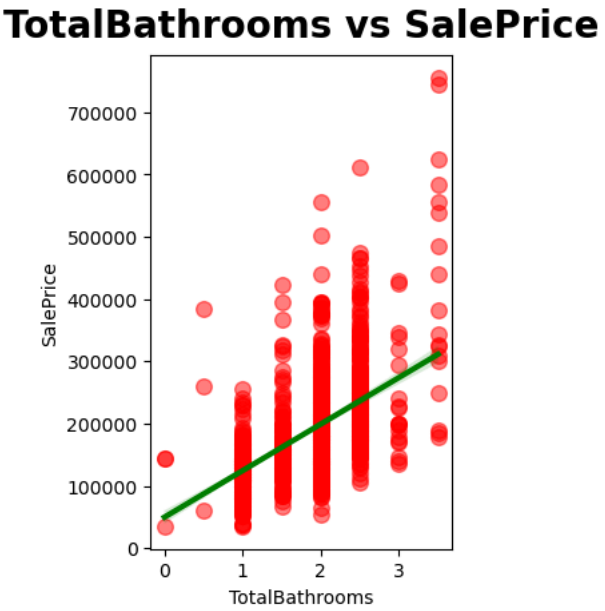


In [282…
```
# Dropping Features columns with weak correlation:
```

In [284…
```
dataset.drop(columns=['FullBath','HalfBath','BsmtFullBath','BsmtHalfBath'],inplace=True)
```

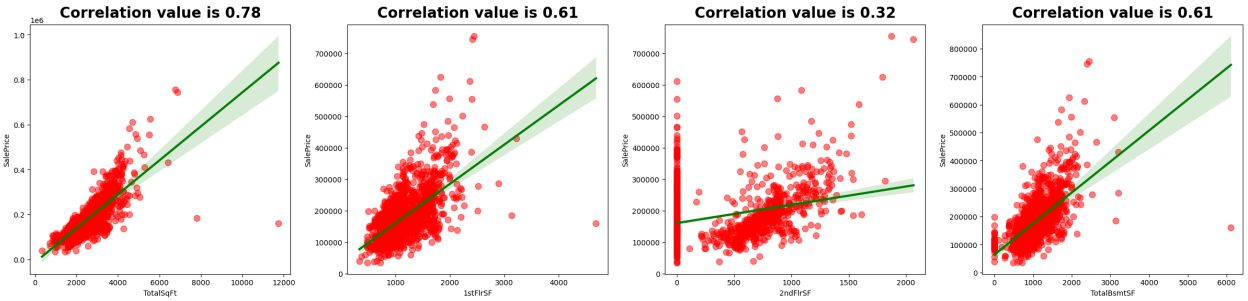
In [286…
```
# Visualizing the New column created:
```

In [288…
```
plt.subplot(1, 2, 2)
sns.regplot(x=dataset["TotalBathrooms"], y=dataset["SalePrice"], color="red",
            scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
plt.title("TotalBathrooms vs SalePrice", pad=10, size=20, fontweight="black")
plt.show()
```



In [290…
```
# Correlation between Feature-Column and Target-Column:
```

In [304…
```
cols = ['TotalSqFt', '1stFlrSF', '2ndFlrSF', 'TotalBsmtSF']

plt.figure(figsize=(24, 6))
for index, column in enumerate(cols):
    plt.subplot(1, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="bold")
plt.tight_layout()
plt.show()
```

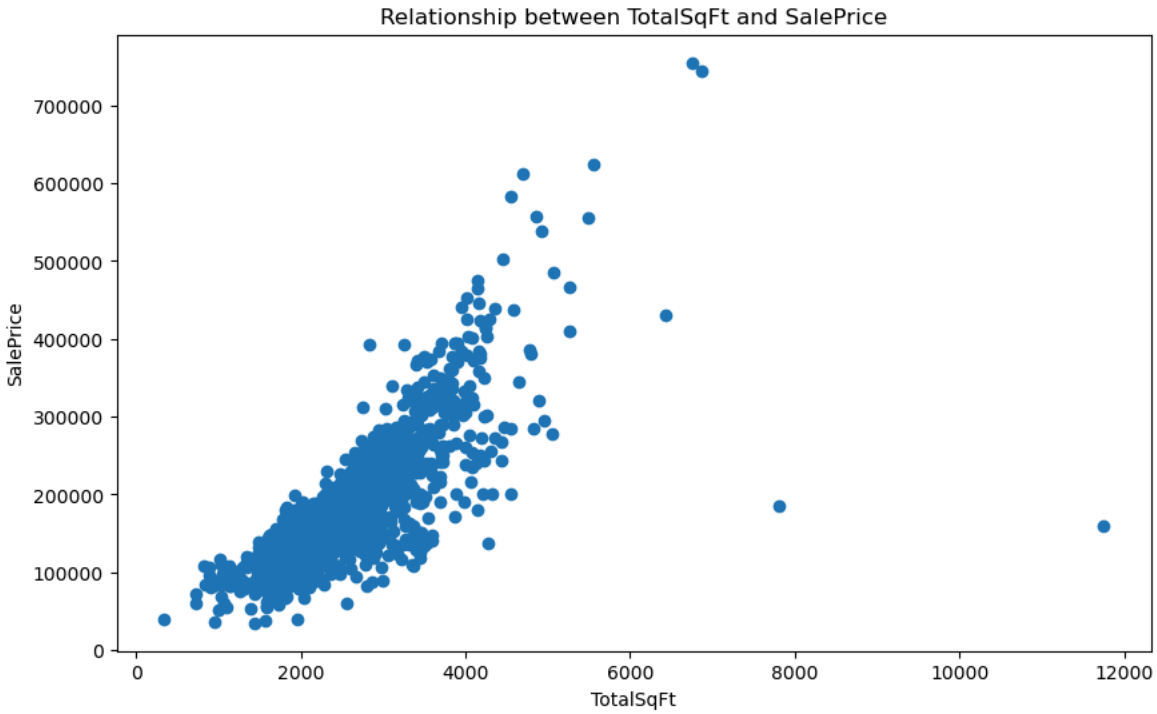


In [306…
```
# Dropping Features columns with weak correlation:
```

In [308…
```
dataset.drop(columns=['2ndFlrSF'],inplace=True)
```

In [310…
```
# Visualizing the New column created:
```

In [318…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['TotalSqFt'],dataset['SalePrice'])
plt.xlabel('TotalSqFt')
plt.ylabel('SalePrice')
plt.title('Relationship between TotalSqFt and SalePrice')
plt.show()
```



In [320…
```
# Calculate the total porch area:
```

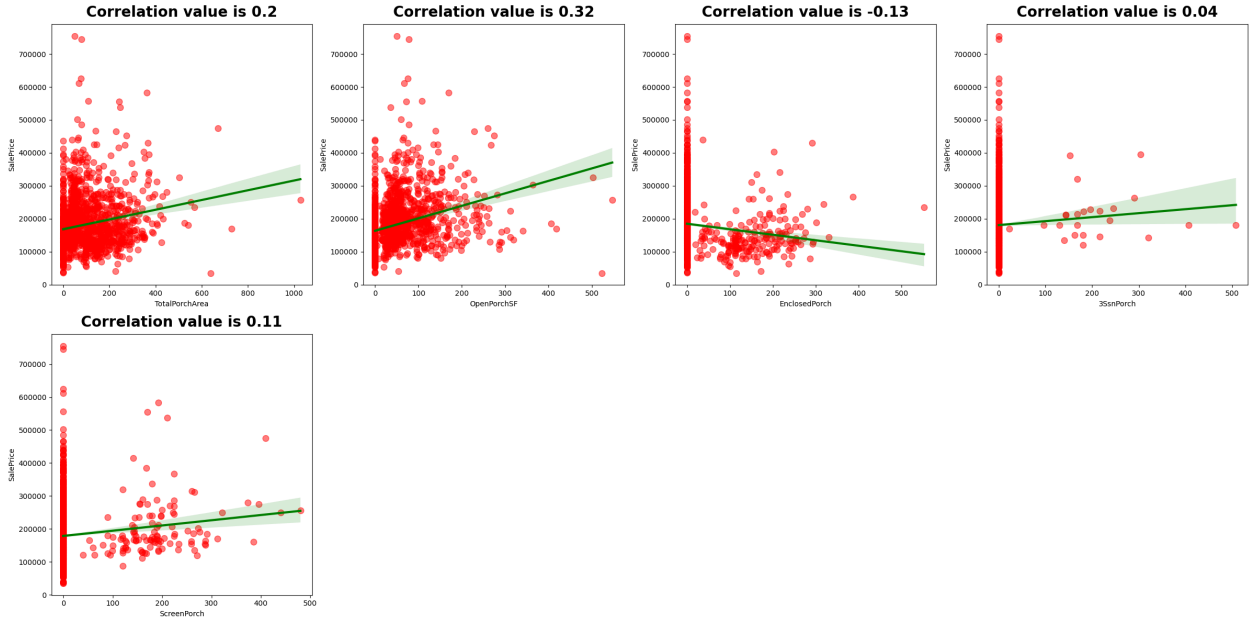
In [322…
```
dataset['TotalPorchArea'] = dataset['OpenPorchSF'] + dataset['EnclosedPorch'] + dataset['3SsnPorch'] + dataset['ScreenPorch']
```

In [324…
```
dataset['TotalPorchArea']
```

Out[324…
```
0        61
1         0
2        42
3       307
4        84
       ...
1455     40
1456      0
1457     60
1458    112
1459     68
Name: TotalPorchArea, Length: 1460, dtype: int64
```

In [326…
```
# Correlation between Feature-Column and Target-Column:
```

In [328…
```
cols = ['TotalPorchArea', 'OpenPorchSF','EnclosedPorch', '3SsnPorch','ScreenPorch']

plt.figure(figsize=(24, 12))
for index, column in enumerate(cols):
    plt.subplot(2, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="black")
plt.tight_layout()
plt.show()
```

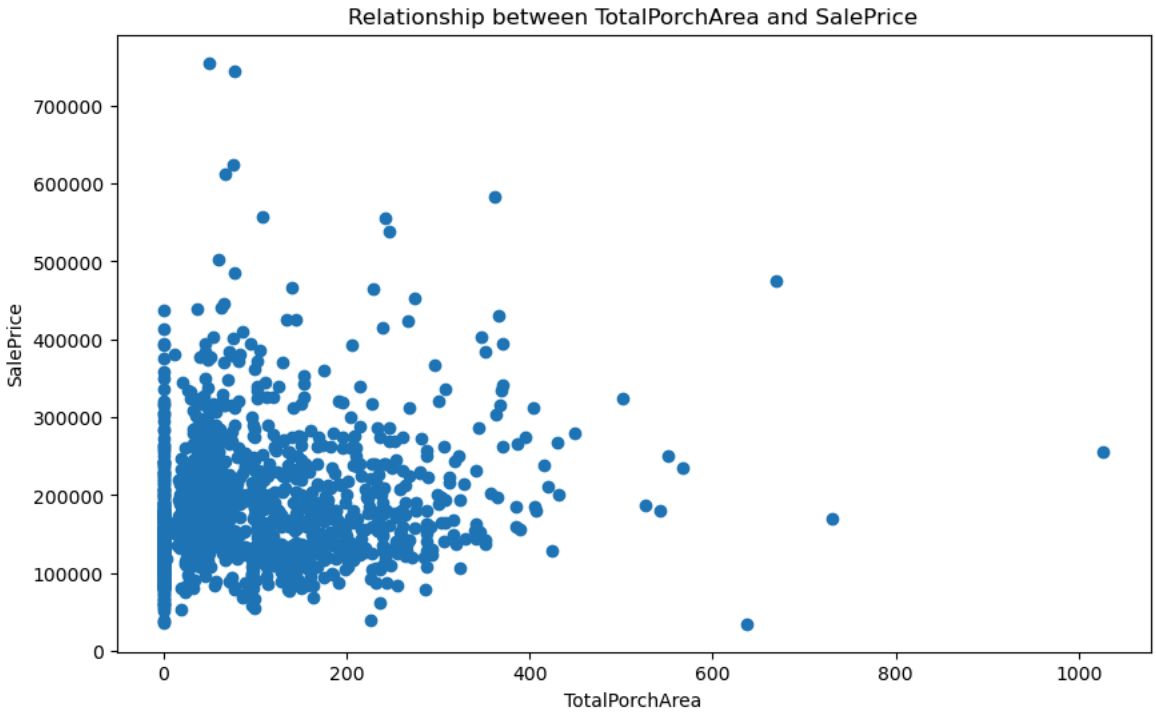


In [330…
```
# Dropping Features columns with weak correlation:
```

In [334…
```
dataset.drop(columns=['ScreenPorch','3SsnPorch','EnclosedPorch'],inplace=True)
```

In [336…
```
# Visualizing the New column created:
```

In [338…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['TotalPorchArea'],dataset['SalePrice'])
plt.xlabel('TotalPorchArea')
plt.ylabel('SalePrice')
plt.title('Relationship between TotalPorchArea and SalePrice')
plt.show()
```



In [340…
```
# Calculate the total finished square footage:
```

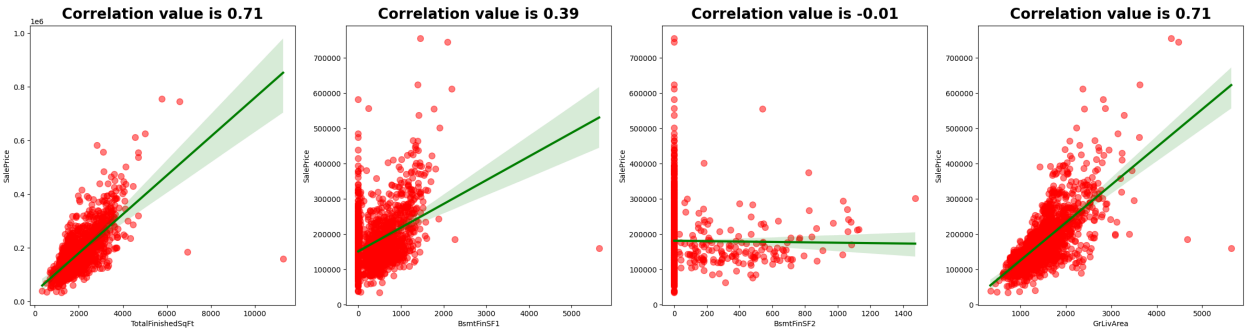
In [342…
```
dataset['TotalFinishedSqFt'] = dataset['BsmtFinSF1'] + dataset['BsmtFinSF2'] + dataset['GrLivArea']
```

In [344…
```
dataset['TotalFinishedSqFt']
```

Out[344…
```
0       2416
1       2240
2       2272
3       1933
4       2853
        ...
1455    1647
1456    3026
1457    2615
1458    2156
1459    2376
Name: TotalFinishedSqFt, Length: 1460, dtype: int64
```

In [346…
```
# Correlation between Feature-Column and Target-Column:
```

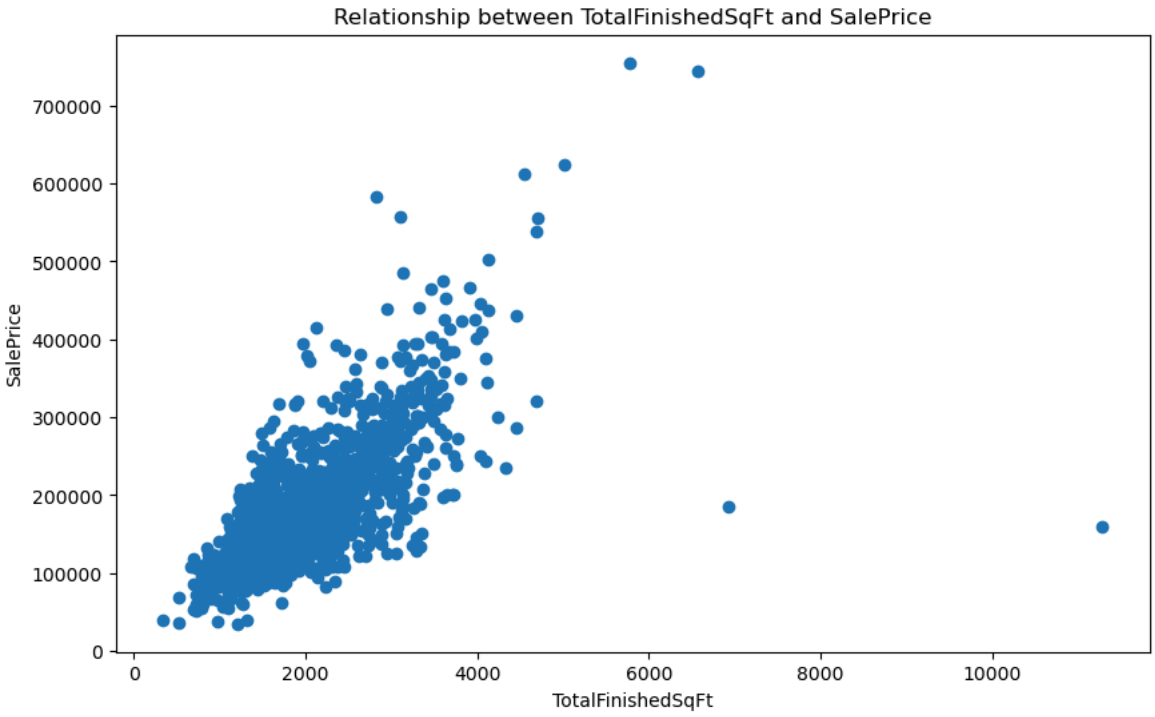
In [348…
```
cols = ['TotalFinishedSqFt', 'BsmtFinSF1','BsmtFinSF2', 'GrLivArea']

plt.figure(figsize=(24, 12))
for index, column in enumerate(cols):
    plt.subplot(2, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="black")
plt.tight_layout()
plt.show()
```



In [350…
```
# Dropping features columns with weak correlation:
```

In [352…
```
dataset.drop(columns=['BsmtFinSF2','BsmtFinSF1'],inplace=True)
```

In [354…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['TotalFinishedSqFt'],dataset['SalePrice'])
plt.xlabel('TotalFinishedSqFt')
plt.ylabel('SalePrice')
plt.title('Relationship between TotalFinishedSqFt and SalePrice')
plt.show()
```



In [356…
```
# Create total Quality-Score:
```

In [358…
```
dataset['TotalQualityScore'] = dataset['OverallQual'] + dataset['OverallCond']
```

In [362…
```
dataset['TotalQualityScore']
```

Out[362…
```
0       12
1       14
2       12
3       12
4       13
        ..
1455    11
1456    12
1457    16
1458    11
1459    11
Name: TotalQualityScore, Length: 1460, dtype: int64
```

In [364…
```
# Create total Garage-Score:
```

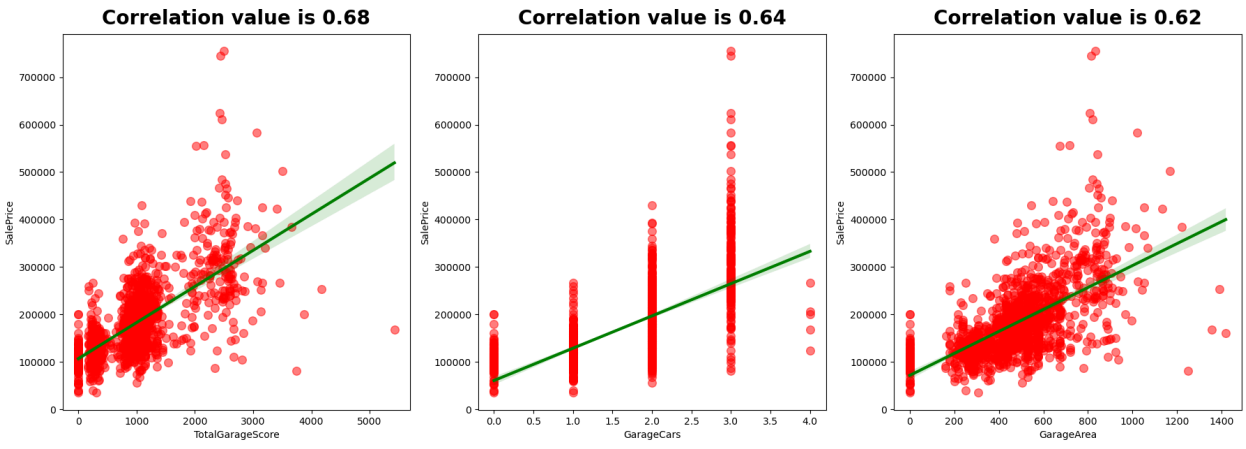
In [366…
```
dataset['TotalGarageScore'] = dataset['GarageCars'] * dataset['GarageArea']
```

In [368…
```
dataset['TotalGarageScore']
```

Out[368…
```
0       1096
1        920
2       1216
3       1926
4       2508
        ...
1455     920
1456    1000
1457     252
1458     240
1459     276
Name: TotalGarageScore, Length: 1460, dtype: int64
```

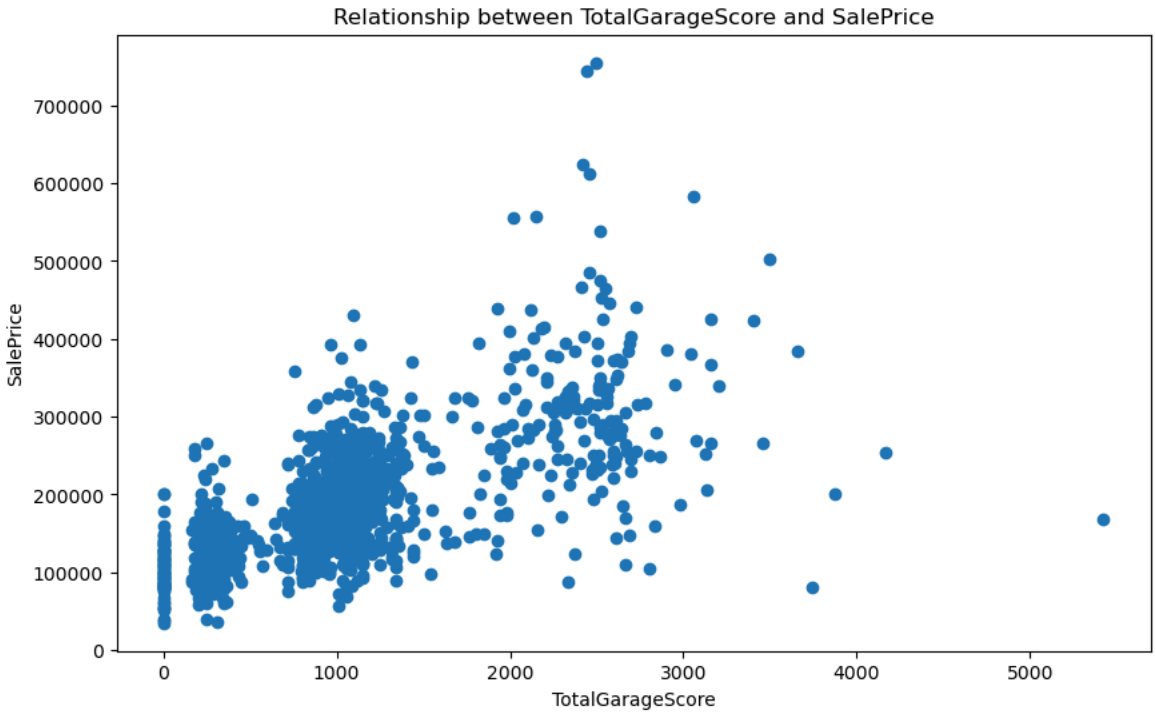
In [370…
```
# Correlation between Feature-Column and Target-Column:
```

In [372…
```
cols = ['TotalGarageScore', 'GarageCars','GarageArea']
plt.figure(figsize=(24, 12))
for index, column in enumerate(cols):
    plt.subplot(2, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="black")
plt.tight_layout()
plt.show()
```



In [374…
```
# Visualizing the New column created:
```

In [376…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['TotalGarageScore'],dataset['SalePrice'])
plt.xlabel('TotalGarageScore')
plt.ylabel('SalePrice')
plt.title('Relationship between TotalGarageScore and SalePrice')
plt.show()
```



In [378…
```
# Calculate the total outdoor-Area:
```

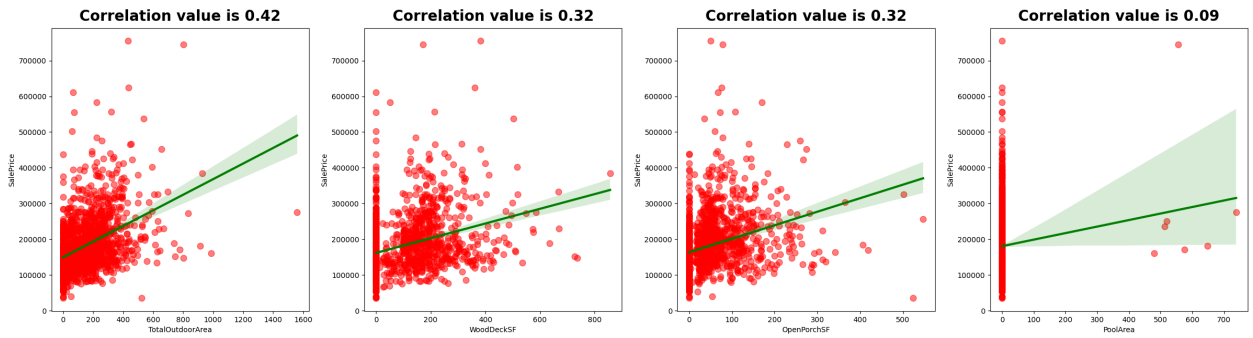
In [382…
```
dataset['TotalOutdoorArea'] = dataset['WoodDeckSF'] + dataset['OpenPorchSF'] +dataset['PoolArea']
```

In [384…
```
dataset['TotalOutdoorArea']
```

Out[384…
```
0        61
1       298
2        42
3        35
4       276
       ...
1455     40
1456    349
1457     60
1458    366
1459    804
Name: TotalOutdoorArea, Length: 1460, dtype: int64
```

In [386…
```
# Correlation between Feature-Column and Target-Column:
```

In [388…
```
cols = ['TotalOutdoorArea', 'WoodDeckSF','OpenPorchSF','PoolArea']
plt.figure(figsize=(24, 12))
for index, column in enumerate(cols):
    plt.subplot(2, 4, index + 1)
    sns.regplot(x=dataset[column], y=dataset["SalePrice"], color="red",
                scatter_kws={'s': 70, 'alpha': 0.5}, line_kws={'color': 'green', 'lw': 3})
    corr = round(dataset[[column, "SalePrice"]].corr()["SalePrice"][0], 2)
    plt.title(f"Correlation value is {corr}", pad=10, size=20, fontweight="black")
plt.tight_layout()
plt.show()
```



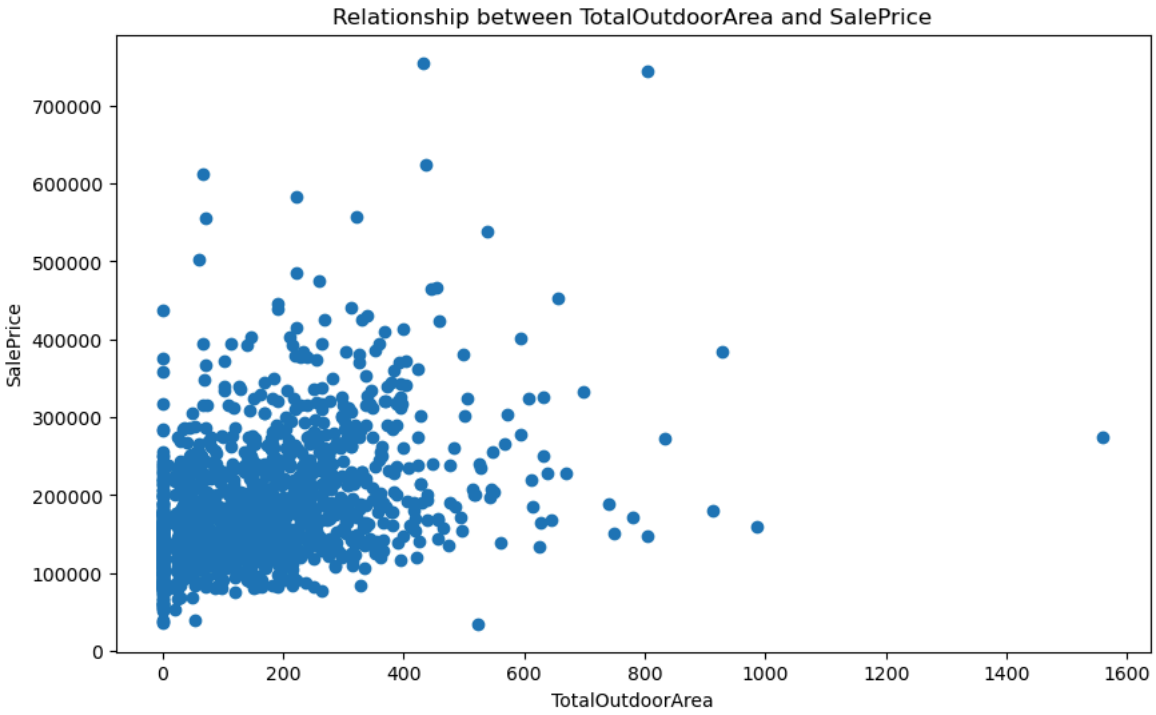
In [390…
```
# Dropping Features columns with weak correlation:
```

In [394…
```
dataset.drop(columns=['WoodDeckSF','OpenPorchSF','PoolArea'],inplace=True)
```

In [396…
```
# Visualizing the New column created:
```

In [400…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['TotalOutdoorArea'],dataset['SalePrice'])
```

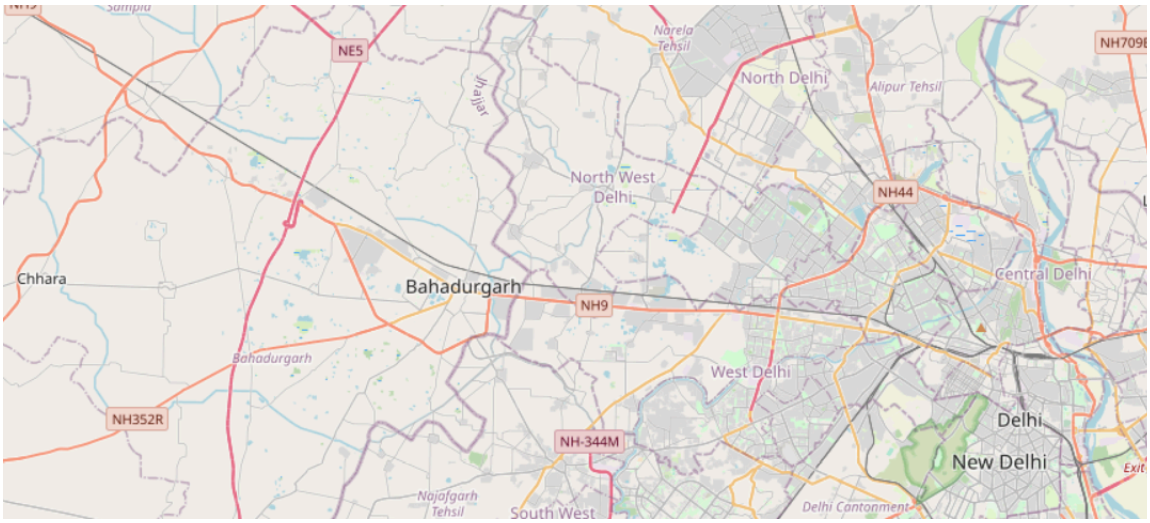
```
plt.xlabel('TotalOutdoorArea')
plt.ylabel('SalePrice')
plt.title('Relationship between TotalOutdoorArea and SalePrice')
plt.show()
```



In [404…
```
# Geospatial Analysis:
```

In [409…
```
# Create a Scatter-Map plot:
```

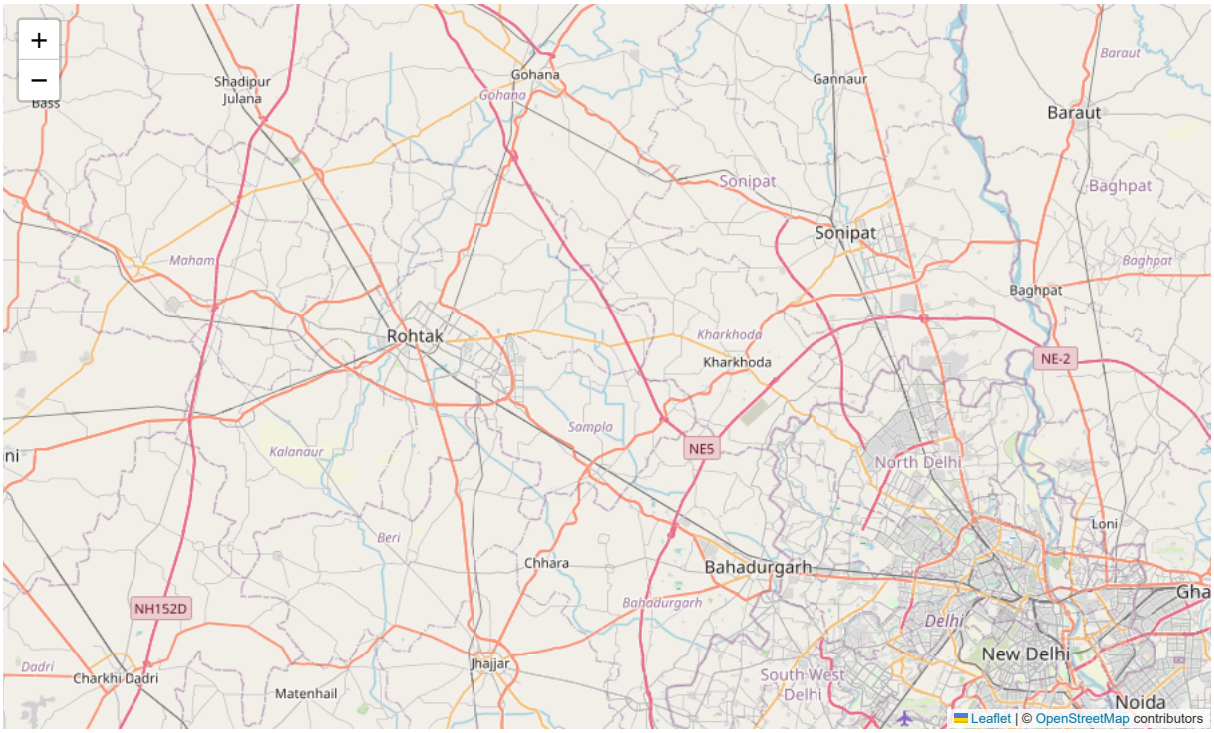
In [411…
```
fig = px.scatter_mapbox(dataset, lat='LotArea', lon='LotFrontage',
                        hover_name='Identifies_the_type_of_dwelling', color='SalePrice',
                        mapbox_style='open-street-map', zoom=10, center={'lat': 28.7041, 'lon': 77.1025},
                        labels={'SalePrice': 'Sale Price'})
fig.show()
```



In [413…
```
# Create a map centered at a Specific-Location:
```

In [425…
```
map = folium.Map(location=[28.7041, 77.1025], zoom_start=10)
# Add markers for each data point
for index, row in dataset.iterrows():
    folium.Marker([row['LotArea'], row['LotFrontage']], popup=f"Price: {row['SalePrice']}").add_to(map)
map
```

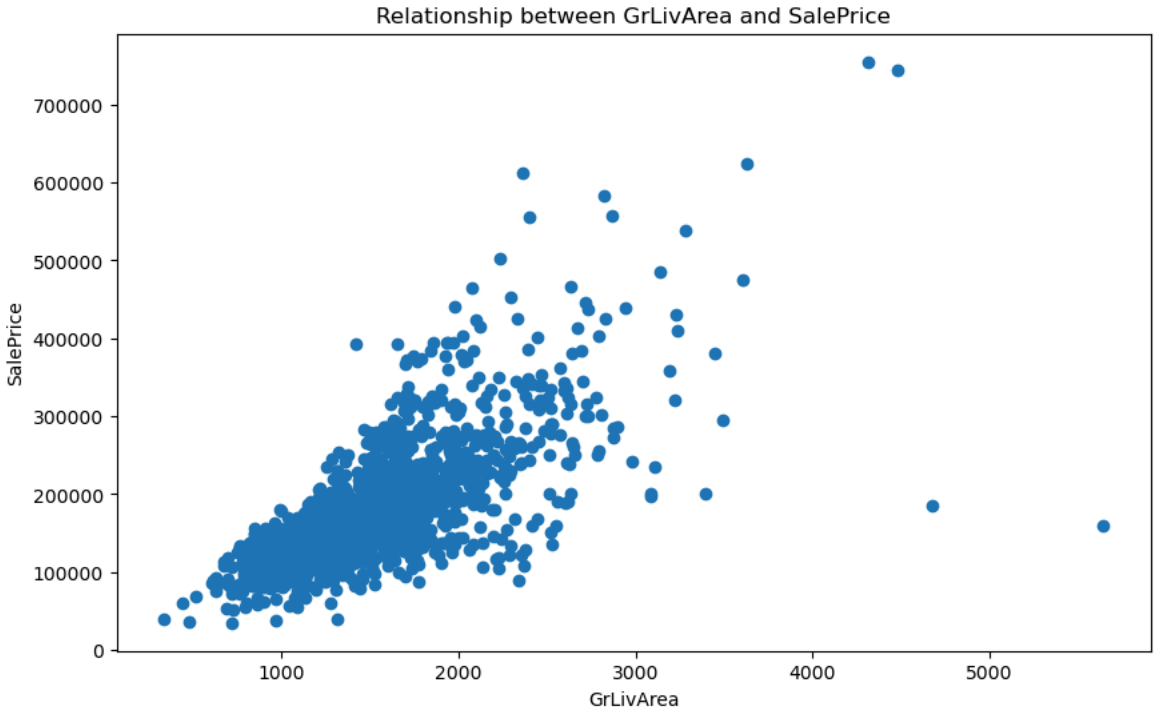
Out[425…



In [429…
```
#Feature Engineering and Size Impact:
```

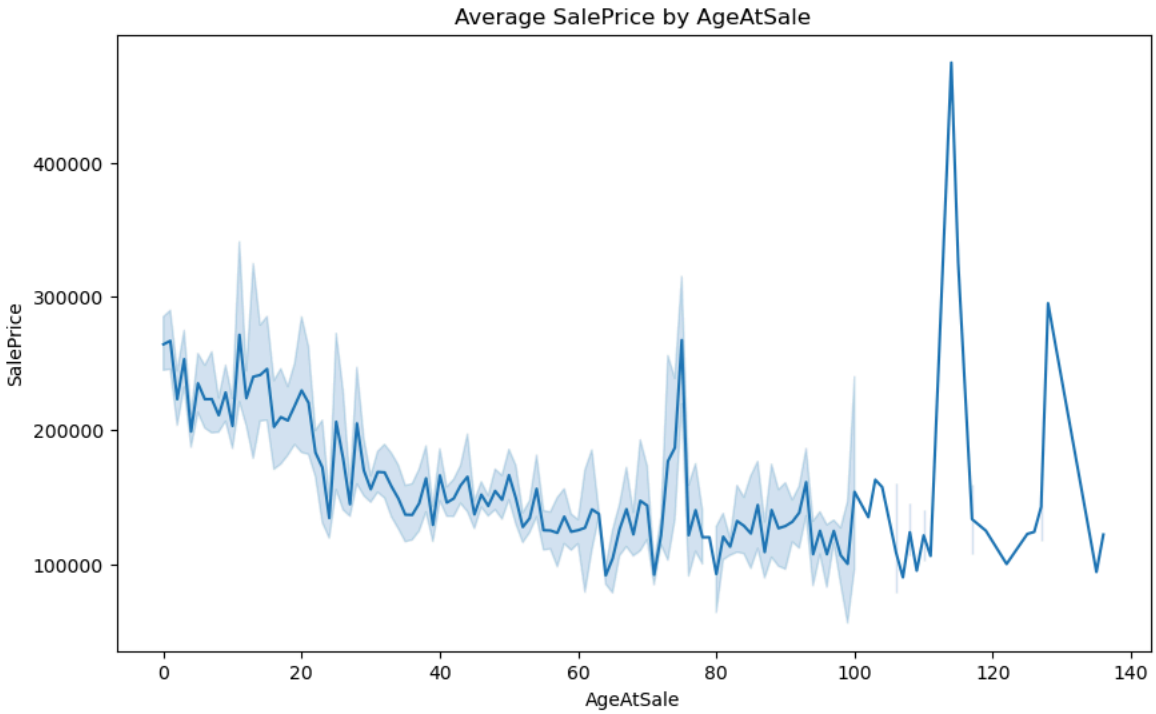
In [431…
```
# Scatter-plot:
```

In [435…
```
plt.figure(figsize=(10, 6))
plt.scatter(dataset['GrLivArea'],dataset['SalePrice'])
plt.xlabel('GrLivArea')
plt.ylabel('SalePrice')
plt.title('Relationship between GrLivArea and SalePrice')
plt.show()
```



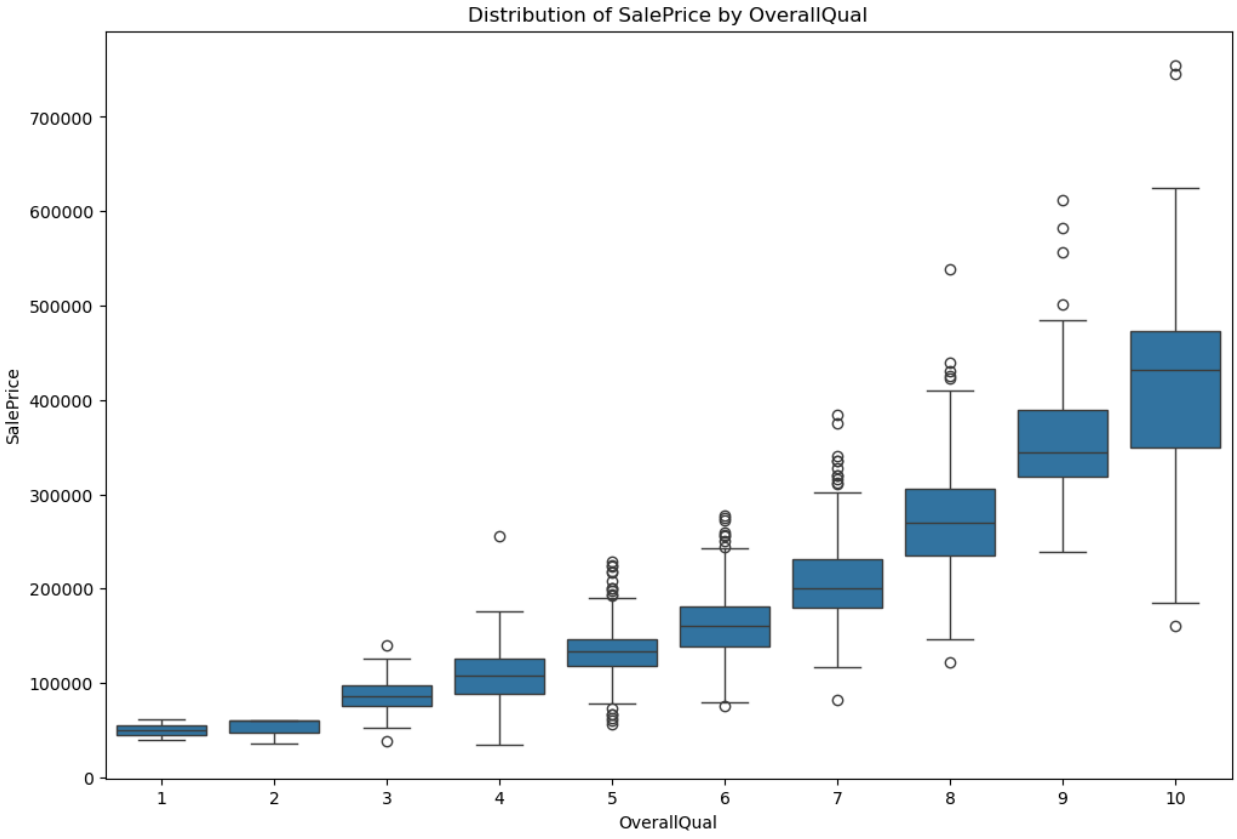
In [437…
```
# Line-plot:
```

In [439…
```
plt.figure(figsize=(10, 6))
sns.lineplot(x='AgeAtSale', y='SalePrice', data= dataset)
plt.xlabel('AgeAtSale')
plt.ylabel('SalePrice')
plt.title('Average SalePrice by AgeAtSale')
plt.show()
```



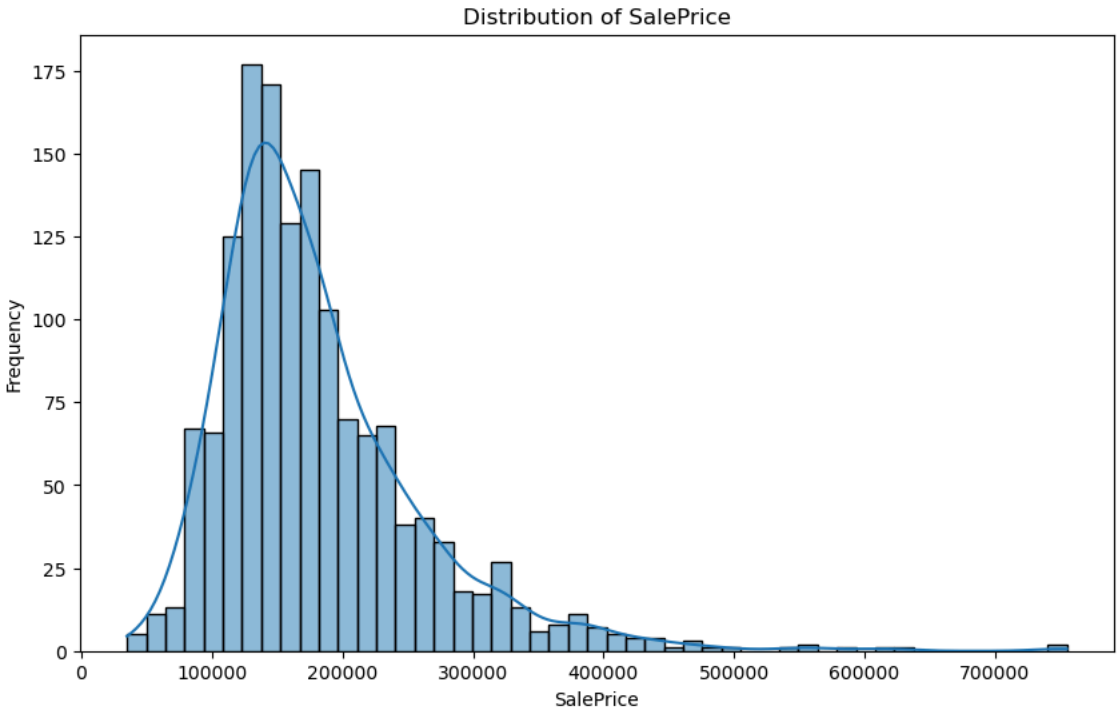
In [441…
```
# Box-plot:
```

In [443…
```
plt.figure(figsize=(12, 8))
sns.boxplot(x='OverallQual', y='SalePrice', data= dataset)
plt.xlabel('OverallQual')
plt.ylabel('SalePrice')
plt.title('Distribution of SalePrice by OverallQual')
plt.show()
```



In [445…
```
# Histogram:
```

In [447…
```
plt.figure(figsize=(10, 6))
sns.histplot(dataset['SalePrice'], kde=True)
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
plt.title('Distribution of SalePrice')
plt.show()
```



In [449…
```
# Feature Engineering:
```

In [453…
```
encoding_dataset=pd.get_dummies(dataset, columns= ['Identifies_the_type_of_dwelling','MSZoning','Street','LotShape','LandContour'
       'Utilities','Lot_configuration','LandSlope',
       'Neighborhood','Condition1','Condition2', 'Type_of_dwelling', 'HouseStyle',
       'RoofStyle', 'Roof_material', 'Exterior1st', 'Exterior2nd', 'ExterQual',
       'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
       'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir',
       'Electrical_system', 'Kitchen_Quality', 'Functional', 'FireplaceQu',
       'Garage_location', 'GarageFinish', 'GarageQual', 'Garage_condition',
       'Paved driveway', 'Pool_quality', 'Fence_Quality',
       'Miscellaneous_feature', 'MoSold', 'SaleType', 'SaleCondition'],drop_first=True,dtype=int)
```

In [455…
```
encoding_dataset
```

Out[455…

	LotFrontage	LotArea	OverallQual	OverallCond	YearRemodAdd	MasVnrArea	BsmtUnfSF	TotalBsmtSF	1stFlrSF	Low_quality_finished_
0	65	8450	7	5	2003	196	150	856	856	
1	80	9600	6	8	1976	0	284	1262	1262	
2	68	11250	7	5	2002	162	434	920	920	
3	60	9550	7	5	1970	0	540	756	961	
4	84	14260	8	5	2000	350	490	1145	1145	
...	...	...	...	...	...	...	...	...	...	
1455	62	7917	6	5	2000	0	953	953	953	
1456	85	13175	6	6	1988	119	589	1542	2073	
1457	66	9042	7	9	2006	0	877	1152	1188	
1458	68	9717	5	6	1996	0	0	1078	1078	
1459	75	9937	5	6	1965	0	136	1256	1256	

1460 rows × 272 columns

In [457…
```
# Total Square column:
```

In [459…
```
encoding_dataset['TotalSqFt']
```

Out[459…
```
0       2566
1       2524
2       2706
3       2473
4       3343
        ...
1455    2600
1456    3615
1457    3492
1458    2156
1459    2512
Name: TotalSqFt, Length: 1460, dtype: int64
```

In [461…
```
# Total Total-Bathrooms column:
```

In [463…
```
encoding_dataset['TotalBathrooms']
```

Out[463…
```
0       2.5
1       2.0
2       2.5
3       1.0
4       2.5
       ...
1455    2.5
1456    2.0
1457    2.0
1458    1.0
1459    1.5
Name: TotalBathrooms, Length: 1460, dtype: float64
```

In [465…
```
# Normalize Numerical Features:
```

In [467…
```
from sklearn.preprocessing import MinMaxScaler
```

In [469…
```
scaler = MinMaxScaler()
encoding_dataset[['TotalSqFt', 'TotalBathrooms']] = scaler.fit_transform(encoding_dataset[['TotalSqFt', 'TotalBathrooms']])
```

In [471…
```
dataset[['TotalSqFt', 'TotalBathrooms']]
```

Out[471…

	TotalSqFt	TotalBathrooms
0	2566	2.5
1	2524	2.0
2	2706	2.5
3	2473	1.0
4	3343	2.5
…	…	…
1455	2600	2.5
1456	3615	2.0
1457	3492	2.0
1458	2156	1.0
1459	2512	1.5

1460 rows × 2 columns

In [473…
```
# Feature Selection:
```

In [477…
```
selected_features = ['TotalSqFt', 'TotalBathrooms', 'Neighborhood_NAmes', 'HouseStyle_2Story']
x = encoding_dataset[selected_features]
y = encoding_dataset['SalePrice']
```

In [479…
```
x
```

Out[479…

	TotalSqFt	TotalBathrooms	Neighborhood_NAmes	HouseStyle_2Story
0	0.195481	0.714286	0	1
1	0.191802	0.571429	0	0
2	0.207742	0.714286	0	1
3	0.187336	0.285714	0	1
4	0.263531	0.714286	0	1
...	...	...	...	...
1455	0.198459	0.714286	0	1
1456	0.287353	0.571429	0	0
1457	0.276581	0.571429	0	1
1458	0.159573	0.285714	1	0
1459	0.190751	0.428571	0	0

1460 rows × 4 columns

In [481…
```
y
```

Out[481…
```
0       208500
1       181500
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

In [483…
```
# Correlation Analysis:
```

In [485…
```
correlation_matrix = encoding_dataset.corr()
```

In [487…
```
correlation_with_saleprice = correlation_matrix['SalePrice'].sort_values(ascending=False)
```

In [489…
```
print(correlation_with_saleprice.head(10))
```

```
SalePrice            1.000000
OverallQual          0.790982
TotalSqFt            0.782260
GrLivArea            0.708624
TotalFinishedSqFt    0.705327
TotalGarageScore     0.680058
PricePerSqFt         0.640819
GarageCars           0.640409
GarageArea           0.623431
TotalBsmtSF          0.613581
Name: SalePrice, dtype: float64
```

In [491…
```
print(correlation_with_saleprice.tail(10))
```

```
HeatingQC_TA             -0.312677
BsmtExposure_No          -0.319990
Foundation_CBlock        -0.343263
Garage_location_Detchd   -0.354141
GarageFinish_Unf         -0.410608
BsmtQual_TA              -0.452394
FireplaceQu_No           -0.471908
Kitchen_Quality_TA       -0.519298
AgeAtSale                -0.523350
ExterQual_TA             -0.589044
Name: SalePrice, dtype: float64
```

In [493…
```
# Model Building:
```

In [495…
```
from sklearn.model_selection import train_test_split
```

In [497…
```
X=encoding_dataset.drop('SalePrice',axis=1)
Y=encoding_dataset.SalePrice
```

In [499…
```
x
```

Out[499…

	TotalSqFt	TotalBathrooms	Neighborhood_NAmes	HouseStyle_2Story
0	0.195481	0.714286	0	1
1	0.191802	0.571429	0	0
2	0.207742	0.714286	0	1
3	0.187336	0.285714	0	1
4	0.263531	0.714286	0	1
...	...	...	...	...
1455	0.198459	0.714286	0	1
1456	0.287353	0.571429	0	0
1457	0.276581	0.571429	0	1
1458	0.159573	0.285714	1	0
1459	0.190751	0.428571	0	0

1460 rows × 4 columns

In [501…
```
y
```

Out[501…
```
0       208500
1       181500
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

In [503…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [505…
```
X_train
```

Out[505…

	LotFrontage	LotArea	OverallQual	OverallCond	YearRemodAdd	MasVnrArea	BsmtUnfSF	TotalBsmtSF	1stFlrSF	Low_quality_finished_
254	70	8400	5	6	1957	0	392	1314	1314	
1066	59	7837	6	7	1994	0	799	799	799	
638	67	8777	5	7	1950	0	796	796	796	
799	60	7200	5	7	1950	252	162	731	981	
380	50	5000	5	6	1950	0	808	1026	1026	
...	...	...	...	...	...	...	...	...	...	
1095	78	9317	6	5	2006	0	1290	1314	1314	
1130	65	7804	4	3	1950	0	500	1122	1328	
1294	60	8172	5	7	1990	0	697	864	864	
860	55	7642	7	8	1998	0	912	912	912	
1126	53	3684	7	5	2007	130	1373	1373	1555	

1168 rows × 271 columns

In [507…
```
Y_train
```

Out[507…
```
254     145000
1066    178000
638      85000
799     175000
380     127000
         ...
1095    176432
1130    135000
1294    115000
860     189950
1126    174000
Name: SalePrice, Length: 1168, dtype: int64
```

In [509…
```
X_test
```

Out[509…

	LotFrontage	LotArea	OverallQual	OverallCond	YearRemodAdd	MasVnrArea	BsmtUnfSF	TotalBsmtSF	1stFlrSF	Low_quality_finished_
892	70	8414	6	8	2003	0	396	1059	1068	
1105	98	12256	8	5	1995	362	431	1463	1500	
413	56	8960	5	6	1950	0	1008	1008	1028	
522	50	5000	6	7	1950	0	605	1004	1004	
1036	89	12898	9	5	2008	70	598	1620	1620	
…	…	…	…	…	…	…	…	…	…	…
479	50	5925	4	7	2000	435	739	907	1131	
1361	124	16158	7	5	2005	16	256	1530	1530	
802	63	8199	7	5	2005	0	80	728	728	
651	60	9084	4	5	1950	0	755	755	755	
722	70	8120	4	7	1970	0	673	864	864	

292 rows × 271 columns

In [511…
```
Y_test
```

Out[511…
```
892     154500
1105    325000
413     115000
522     159000
1036    315500
         ...
479      89471
1361    260000
802     189000
651     108000
722     124500
Name: SalePrice, Length: 292, dtype: int64
```

In [513…
```
from sklearn.ensemble import RandomForestRegressor
```

In [515…
```
model = RandomForestRegressor()
```

In [519…
```
model.fit(X_train,Y_train)
```

Out[519…

▾ RandomForestRegressor

```
RandomForestRegressor()
```

In [520…
```
y_pred = model.predict(X_test)
```

In [523…
```
y_pred
```

Out[523…

```
array([153457.  , 342214.17, 113539.36, 160875.9 , 317956.13,  81874.5 ,
       250165.17, 146940.03,  85244.14, 133000.54, 145975.  , 129685.14,
        83682.16, 217330.9 , 178020.15, 135225.5 , 184075.7 , 134972.75,
       118170.12, 224555.7 , 152811.  , 215454.94, 175149.78, 128759.05,
       189113.55, 153920.  , 180316.88, 135899.02, 180216.99, 209687.91,
       131389.52, 277800.02, 178447.37, 134692.6 , 253465.5 , 142029.  ,
       143084.5 , 215058.19, 312185.06, 103102.  , 100639.  , 206903.36,
       121162.5 , 271747.16, 129114.5 , 119394.55, 117215.  , 125091.5 ,
       447140.11, 144262.87, 122323.5 , 196881.86, 116256.7 , 302646.92,
       141787.  , 256699.52, 213371.  , 171903.2 , 106280.16, 103313.28,
        81863.  , 157960.85, 322026.06, 275063.95, 285795.59, 243952.75,
       108114.  , 309214.46,  94325.  , 179280.87, 124760.56, 134084.  ,
       110554.  ,  90632.  , 533283.2 , 173851.6 , 334628.17, 334583.33,
       142751.5 , 126190.28, 109903.92,  79720.66, 112674.62,  98119.42,
       151084.04, 132165.62, 270810.64, 188168.5 , 146313.  , 158124.59,
       155216.55, 153354.5 , 125072.7 , 280437.8 , 138694.7 , 170055.94,
       210247.3 , 180627.31, 208943.86, 259311.35, 163773.37, 229429.75,
       226321.27, 131795.04, 191261.32, 154781.5 , 156361.  , 272783.8 ,
       143438.  , 141424.05,  72299.  , 124317.5 , 127644.79, 137631.35,
       212378.2 , 118944.28, 108066.5 , 123397.5 , 117897.36, 280098.28,
       130838.95, 145343.  , 169543.94, 207141.97, 175772.4 , 140138.6 ,
       230591.1 ,  79023.83, 142892.  , 166416.5 , 185898.5 , 372317.31,
       197872.91, 105233.  ,  60497.98, 348321.54, 371037.94, 130646.3 ,
       229823.1 , 596792.51, 398884.35, 133464.  , 177119.  , 139116.  ,
       145756.28, 121657.  , 249525.12, 186738.9 , 129433.  ,  68162.81,
       136038.42, 159328.46, 182014.5 , 147909.15,  90313.54, 136551.24,
       115573.25, 142428.  ,  96509.5 , 133463.  , 190355.5 , 144646.53,
       294926.54, 142755.  , 118281.  , 139002.97, 236707.01, 280652.22,
       501033.29, 251289.73, 343773.63, 104221.5 ,  87159.66, 152678.09,
       286630.22, 133205.52, 110743.83, 187002.4 , 125164.58, 176873.  ,
       180350.  ,  69712.98, 134938.9 , 135770.82, 258911.3 , 141732.53,
       270903.08, 246277.29, 202689.07,  81531.66, 123210.  , 104552.  ,
       159239.6 , 165992.57, 195660.48, 227029.6 , 197782.47,  79459.33,
       186518.81, 149491.8 , 270569.42, 179017.97, 149167.14, 312388.44,
       183001.93, 136969.2 , 241786.04, 137107.03, 140545.75, 112668.5 ,
       223432.9 , 142618.52, 129983.5 , 172743.7 , 243463.88, 257720.2 ,
       194816.91, 143318.5 , 134222.37, 133257.64, 142304.  , 266941.1 ,
       219866.5 ,  74681.99, 210328.97, 153733.17,  79049.16,  72860.64,
       172021.34, 100078.04,  98372.64, 169875.67, 119352.5 , 135861.37,
       213869.87, 151183.22, 193990.4 , 149127.96, 270364.35, 144881.07,
       107222.28, 252910.07, 195735.91, 439490.99, 185660.  , 115425.25,
       165368.35, 171501.55, 130053.  , 109853.2 , 191554.93, 166213.32,
       145905.03,  89634.5 , 157101.61, 128010.5 , 114390.23, 120907.04,
       177552.  , 237339.4 , 391909.25, 164367.71, 130526.14, 270521.07,
       313317.3 , 211403.66, 159386.37, 141312.  , 112903.66, 167195.63,
       444226.09, 190527.31, 249967.17,  89283.66,  95290.14, 147200.5 ,
       135427.5 , 319062.09, 146389.5 , 139579.62, 194874.15,  90698.  ,
       200796.35, 110169.14, 305342.1 , 182148.  , 232386.56,  91345.7 ,
       263203.12, 185021.15, 113281.75, 127324.  ])
```

In [525…

```
score = model.score(X_test, Y_test)
print(f'Model Score: {score}')
```

```
Model Score: 0.9782131250980548
```

In [527…

```
from sklearn.metrics import mean_squared_error, r2_score
```

In [529…

```
mse = mean_squared_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 167112437.53309795
R-squared: 0.9782131250980548
```

In [531…

```
# Creating Linear-Regression Model:
```

In [533…

```
from sklearn.linear_model import LinearRegression
```

In [535…

```
model = LinearRegression()
```

In [537…

```
model.fit(X_train,Y_train)
```

Out[537…

LinearRegression

```
LinearRegression()
```

In [541…

```
Y_pred = model.predict(X_test)
```

In [543…

```
Y_pred
```

Out[543…

```
array([ 155834.53736845,  338060.18951049,   99866.86992923,
        159021.50507417,  315216.58709572,   64381.07876678,
        310090.51722852,  142032.70521605,   83936.75210109,
        144167.41548721,  141593.31838553,  140746.6081654 ,
         51478.39277277,  218105.49673923,  176738.9916562 ,
        132612.247506  ,  177964.38343142,  128659.48464585,
        124319.63447357,  223790.71209828,  162610.03057535,
        215691.17215502,  174217.51110072,  124126.30832556,
        187806.48576313,  140352.38056425,  173610.28378741,
        150575.26044831,  178285.6141857 ,  200167.0345512 ,
        148022.17031487,  290550.99044195,  259485.85651562,
        129928.04999672,  251318.14072765,  136801.25795037,
        150015.57401676,  214380.05694296,  307630.26357875,
        140432.25444592,   92501.81196964,  208752.85069686,
        113726.05884707,  312171.2256511 ,  123823.12052506,
        141188.92430253,  112515.91253711,  126938.22967747,
        439869.6220337 ,  151982.55310588,  117292.03480193,
        229431.250887  ,  123686.32308196,  303157.04518913,
        157154.82963434,  253419.25528375,  217494.96074356,
        175395.3124068 ,   95173.52393749,  184232.66936542,
         63676.06562761,  147712.61911114,  317782.46511355,
        261484.1642181 ,  283949.49226837,  225392.04917001,
        111166.92461494,  298012.7176029 ,   84999.69323336,
        182223.6132857 ,  101412.40916307,  129113.07748497,
        100760.35393818,   65736.2117287 ,  546401.24140431,
        162914.96265996,  339411.9296478 ,  364197.07196124,
        137279.23250572,  135594.5668087 ,  136791.4570919 ,
        109392.90498911,  104304.7348013 ,  122601.13664007,
        149033.49008215,  136731.65354356,  268292.91501509,
        181773.40170962,  166908.2338771 ,  160533.84060508,
        168475.36455357,  169514.75927698,  155682.4620806 ,
        304880.93326362,  158094.60004338,  159793.82646774,
        205167.21827172,  177266.40797659,  202203.9084794 ,
        259565.6312262 ,  171878.35645243,  227086.24204585,
        278222.46082222,  138575.15878837,  197118.44745435,
        153108.22819052,  149187.92899951,  276012.15493457,
        147209.25833716,  142210.53098951,   62561.12221118,
        136404.05418653,  120959.3987206 ,  137093.0471025 ,
        212587.34812473,  144397.03514565,  102005.16058511,
        116286.87582278,   60900.42985483,  310143.7163508 ,
        144840.75393212,  156855.37145739,  164780.11374649,
        196087.70865455,  176406.88720118,  139442.41162931,
        228583.98375596,   54147.77900903,  143480.97768942,
        159262.71180111,  184220.73976129,  355989.17470652,
        193497.3952542 ,  106303.36048011,   -5853.9587221 ,
        360052.08256593,  382825.13465354,  136458.98328146,
        232812.13024198,  627638.09584063,  386622.1180506 ,
        129169.76580913,  186112.01108648,  128674.08175804,
        141332.40664111,  115440.2796815 ,  243090.8292391 ,
        186631.63473781,  147212.43348984,   37031.34372995,
        151089.63270968,  158237.02619393,  263753.48348189,
        167012.51893187,   93504.70443529,  126595.05839921,
        127799.1018853 ,  129708.7116606 ,  106603.76131664,
        131354.92878713,  185979.1106043 ,  124503.35243565,
        293902.69776896,  149654.96831326,  118595.56441085,
        152305.31267963,  250941.49344733,  303540.03459333,
        491414.6351707 ,  249660.3871487 ,  357585.76566994,
        105361.9832991 ,   92035.9021033 ,  150574.30376472,
        311284.06214845,  119918.95385262,  113322.96689079,
        191816.69432422,  130620.40401573,  170506.89280281,
        159119.48982986,    2587.67242804,  129182.48792107,
        143908.52092742,  246802.01734029,  147201.11662294,
        275510.84277439,  243194.28166538,  199626.70376103,
         63889.45370698,  110878.84307517,  103358.68345437,
        186369.80443335,  176306.22491604,  193950.37027286,
        229776.36491593,  190477.09320479,   22131.86939695,
        170194.85464111,  167615.85542536,  279637.60743782,
        189631.96751905,  158756.1769261 ,  315859.74439765,
        177350.94418509,  119896.12455885,  236413.75513965,
        133157.59343411,  138338.46833067,  103662.672418  ,
        228240.11923836,  147606.5858386 ,  124227.25985081,
        173152.61791629, -324642.32137785,  257552.59567327,
        203517.22921564,  128641.1231284 ,  139269.77752871,
         86458.74682861,  148903.6281351 ,  256395.53689754,
        208953.68014611,   45620.16214278,  196377.55398957,
        158207.64662854,   53971.41573168,   42236.23462571,
        173986.84333904,   82166.04027919,   91194.95307446,
        165018.61174804,  116150.27150486,  132237.05236583,
        223080.3435069 ,  157027.95328439,  199362.48967463,
        149226.22528111,  265918.18383667,  153311.45551391,
        126966.21156158,  243567.55847856,  191828.38791843,
        436542.53892948,  186234.15161822,  109370.09932672,
        153046.92899137,  174594.43890029,  132836.79678807,
        121702.1839234 ,  186964.71594897,  166845.5932785 ,
        145539.60983271,   96947.04885599,  166586.06066765,
        124942.33130679,  104183.96976109,  117716.40131255,
        173167.51268932,  242317.5269978 ,  374325.02046876,
        166576.42738094,  136323.34653128,  257138.84700451,
        320838.30374086,  215689.93159753,  146324.64236327,
        151108.31295897,  109656.00799588,  197179.32518776,
```

```
437975.23446   ,  183627.65031956,  240249.27163049,
 78307.26015869,   94238.51716498,  148449.55795247,
132255.2535284 ,  324968.30570118,  144474.61641224,
134579.43807557,  189400.66896795,  111487.93508776,
196965.72879553,  118417.10088505,  300039.09134533,
179044.61956813,  225626.53701859,   92122.82484937,
266965.42473463,  187756.98708178,   86339.81236718,
119781.21159851])
```

In [545…
```
score = model.score(X_test, Y_test)
print(f'Model Score: {score}')
```

```
Model Score: 0.8179650559777593
```

In [547…
```
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 1396267401.7576342
R-squared: 0.8179650559777593
```

In [549…
```
# Market Trends and Historical Pricing:
```

In [551…
```
# Convert Date Information:
```

In [553…
```
# Convert 'MoSold' and 'YrSold' to a datetime column:
```

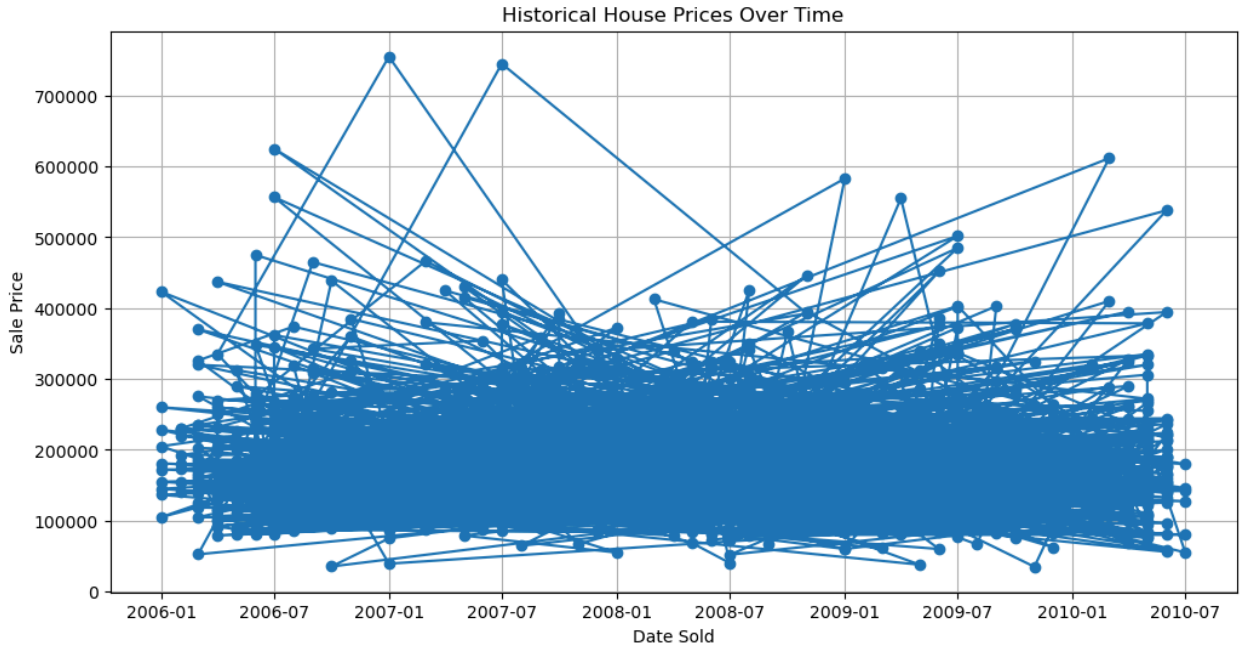
In [559…
```
dataset['DateSold'] = pd.to_datetime(dataset['YrSold'].astype(str) + '-' + dataset['MoSold'].astype(str))
```

In [561…
```
dataset['DateSold']
```

Out[561…
```
0      2008-02-01
1      2007-05-01
2      2008-09-01
3      2006-02-01
4      2008-12-01
          ...
1455   2007-08-01
1456   2010-02-01
1457   2010-05-01
1458   2010-04-01
1459   2008-06-01
Name: DateSold, Length: 1460, dtype: datetime64[ns]
```

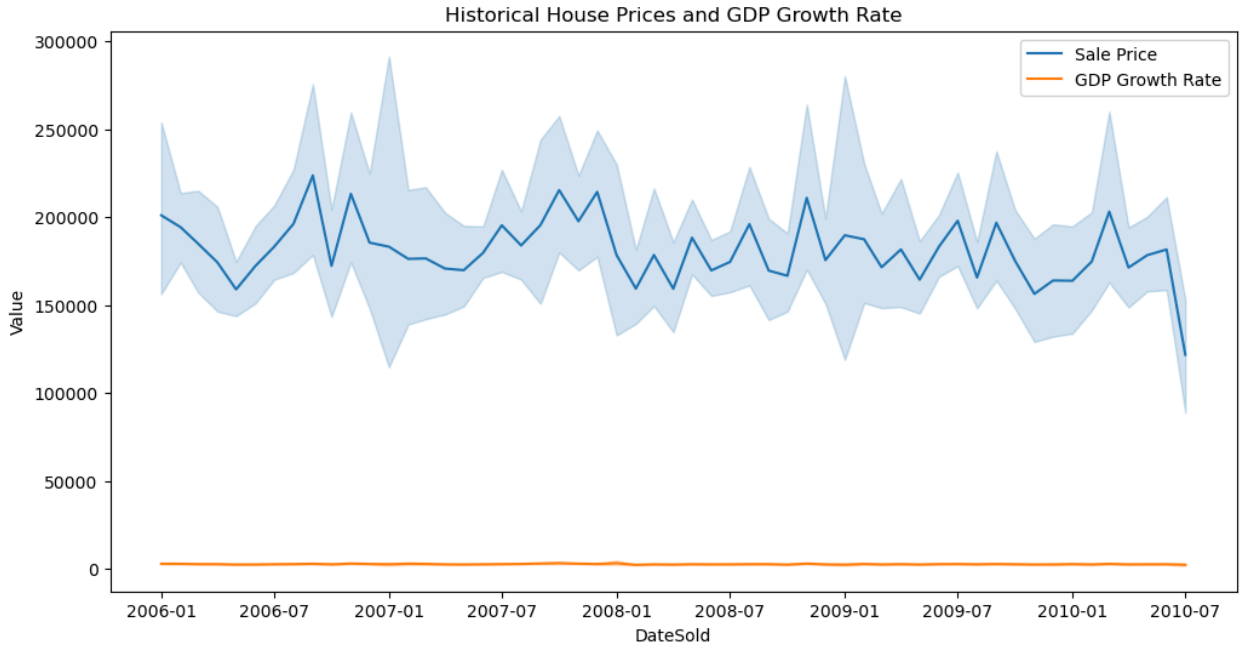
In [563…
```
# Plotting historical prices:
```

In [565…
```
plt.figure(figsize=(12, 6))
plt.plot(dataset['DateSold'],dataset['SalePrice'], marker='o')
plt.title('Historical House Prices Over Time')
plt.xlabel('Date Sold')
plt.ylabel('Sale Price')
plt.grid(True)
plt.show()
```



In [567…
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=dataset, x='DateSold', y='SalePrice', label='Sale Price')
sns.lineplot(data=dataset, x='DateSold', y='TotalSqFt', label='GDP Growth Rate')
plt.title('Historical House Prices and GDP Growth Rate')
```

```
plt.xlabel('DateSold')
plt.ylabel('Value')
plt.legend()
plt.show()
```



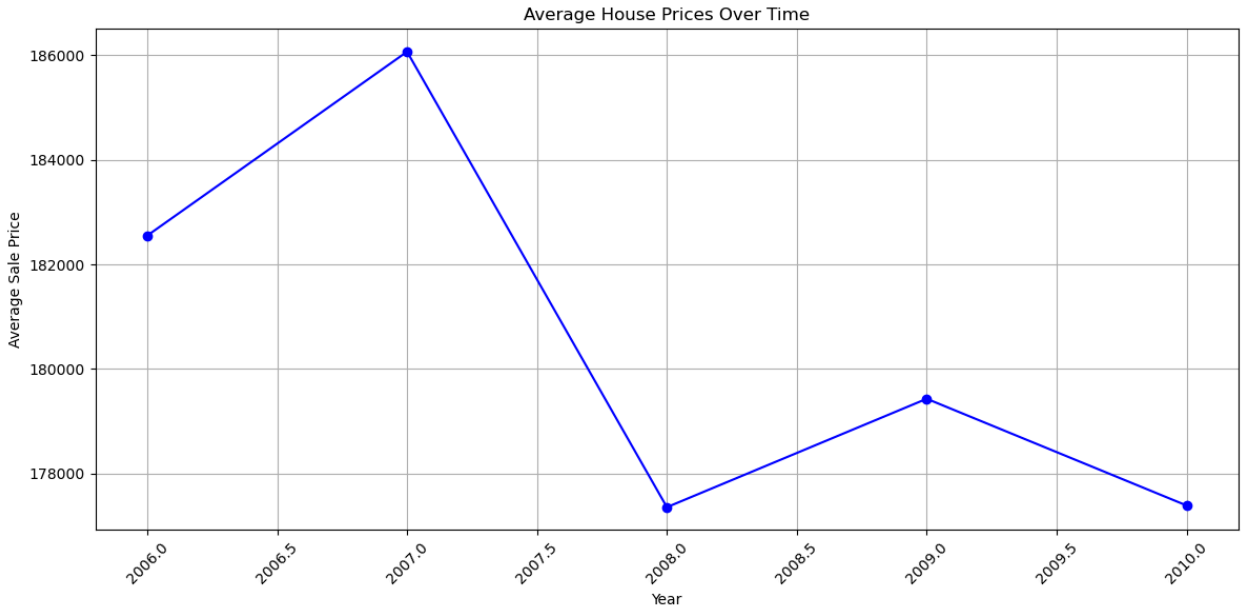
In [569…
```
# Group the data by year and calculate the average house price for each year:
```

In [574…
```
avg_price_by_year = dataset.groupby('YrSold')['SalePrice'].mean()
```

In [576…
```
avg_price_by_year
```

Out[576…
```
YrSold
2006    182549.458599
2007    186063.151976
2008    177360.838816
2009    179432.103550
2010    177393.674286
Name: SalePrice, dtype: float64
```

In [578…
```
plt.figure(figsize=(12, 6))
plt.plot(avg_price_by_year.index, avg_price_by_year.values, marker='o', color='b', linestyle='-')
plt.title('Average House Prices Over Time')
plt.xlabel('Year')
plt.ylabel('Average Sale Price')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [638…
```
# Perform seasonal decomposition:
```

In [640…
```
import statsmodels.api as sm
```

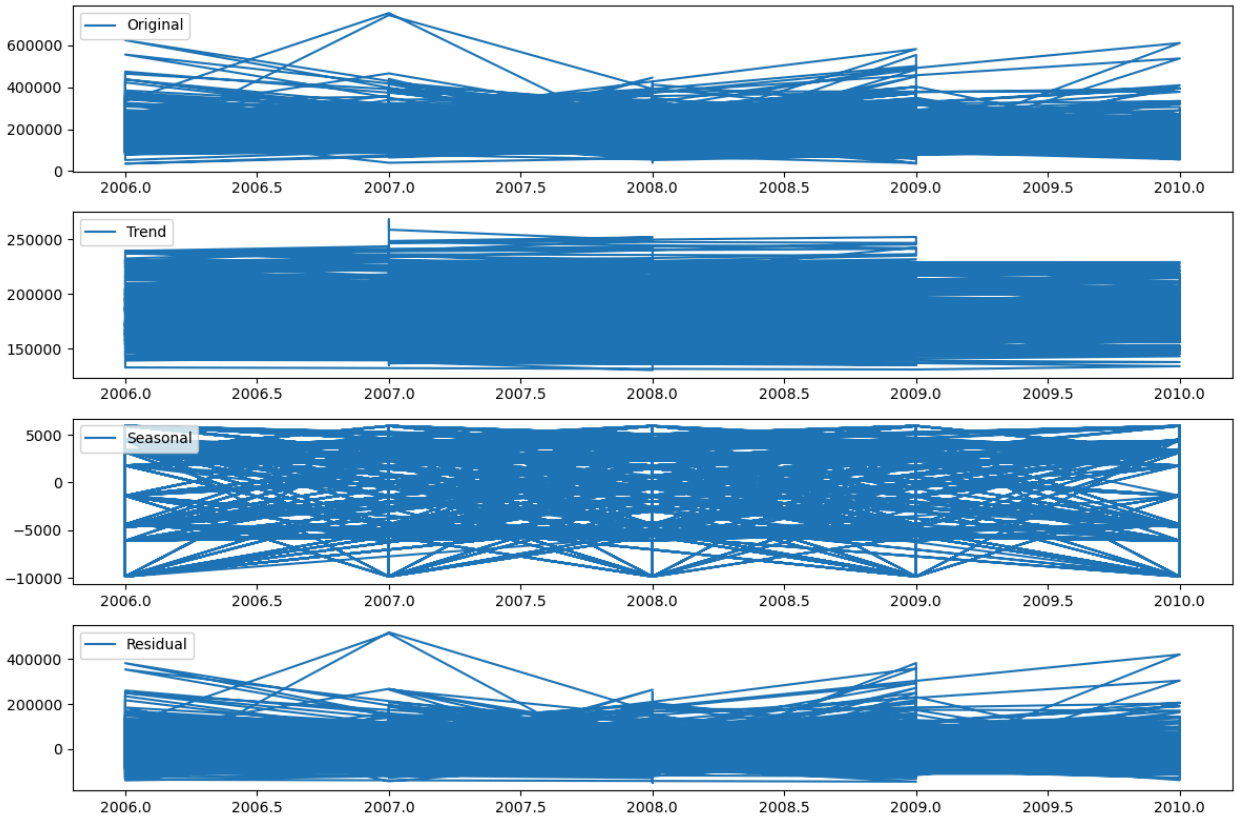
In [644…
```
decomposition = sm.tsa.seasonal_decompose(dataset['SalePrice'], model='additive', period=12)
```

In [646…
```
decomposition
```

Out[646… `<statsmodels.tsa.seasonal.DecomposeResult at 0x116aed73830>`

In [650…
```
# Decomposition-plot:
```

In [652…
```
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(dataset['SalePrice'], label='Original')
plt.legend(loc='upper left')
plt.subplot(412)
plt.plot(decomposition.trend, label='Trend')
plt.legend(loc='upper left')
plt.subplot(413)
plt.plot(decomposition.seasonal, label='Seasonal')
plt.legend(loc='upper left')
plt.subplot(414)
plt.plot(decomposition.resid, label='Residual')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



In [654…
```
# Customer Preferences and Amenities:
```

In [656…
```
# Identify Amenities:
```

In [658…
```
# Print the list of columns in your dataset:
```

In [660…
```
print(dataset.columns)
```

```
Index(['Identifies_the_type_of_dwelling', 'MSZoning', 'LotFrontage', 'LotArea',
       'Street', 'LotShape', 'LandContour', 'Utilities', 'Lot_configuration',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'Type_of_dwelling', 'HouseStyle', 'OverallQual', 'OverallCond',
       'YearRemodAdd', 'RoofStyle', 'Roof_material', 'Exterior1st',
       'Exterior2nd', 'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir',
       'Electrical_system', '1stFlrSF', 'Low_quality_finished_square_feet',
       'GrLivArea', 'BedroomAbvGr', 'KitchenAbvGr', 'Kitchen_Quality',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu',
       'Garage_location', 'GarageYrBlt', 'GarageFinish', 'GarageCars',
       'GarageArea', 'GarageQual', 'Garage_condition', 'Paved driveway',
       'Pool_quality', 'Fence_Quality', 'Miscellaneous_feature',
       'miscellaneous_feature', 'MoSold', 'SaleType', 'SaleCondition',
       'SalePrice', 'PricePerSqFt', 'AgeAtSale', 'TotalSqFt', 'TotalBathrooms',
       'TotalPorchArea', 'TotalFinishedSqFt', 'TotalQualityScore',
       'TotalGarageScore', 'TotalOutdoorArea', 'DateSold'],
      dtype='object')
```

In [662…
```
# Identify the columns related to amenities or features that may impact house prices:
```

In [664…
```
amenities_columns = [ 'GarageArea', 'Fireplaces']
```

In [668…
```
amenities_data = dataset[amenities_columns]
```

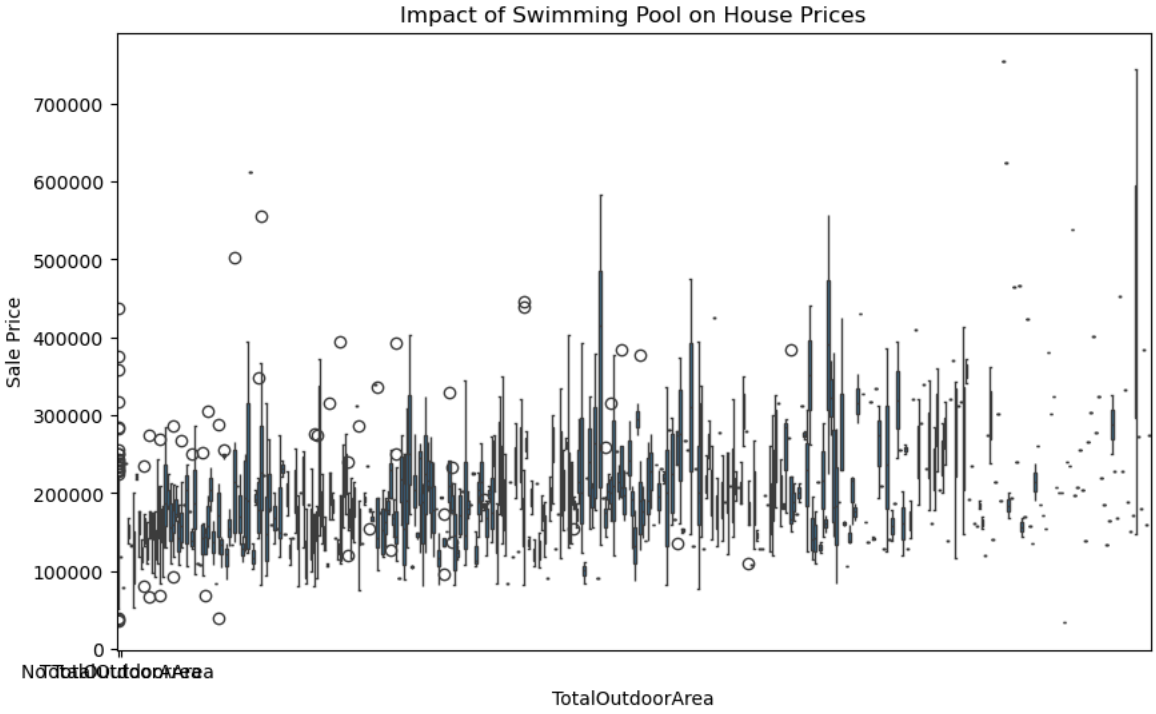
In [670…
```
print(amenities_data.head())
```

```
       GarageArea  Fireplaces
YrSold
2008          548           0
2007          460           1
2008          608           1
2006          642           1
2008          836           1
```

In [672…
```
# Analyze Impact:
```

In [674…
```
# Create a Box-plot to compare house prices with and without a Total Outdoor Area:
```

In [676…
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='TotalOutdoorArea', y='SalePrice', data=dataset)
plt.title('Impact of Swimming Pool on House Prices')
plt.xlabel('TotalOutdoorArea')
plt.ylabel('Sale Price')
plt.xticks([0, 1], ['No TotalOutdoorArea', 'TotalOutdoorArea'])
plt.show()
```



In [678…
```
# Customer Feedback:
```

In [684…
```
from textblob import TextBlob
```

In [686…
```
# Perform sentiment analysis on customer feedback:
```

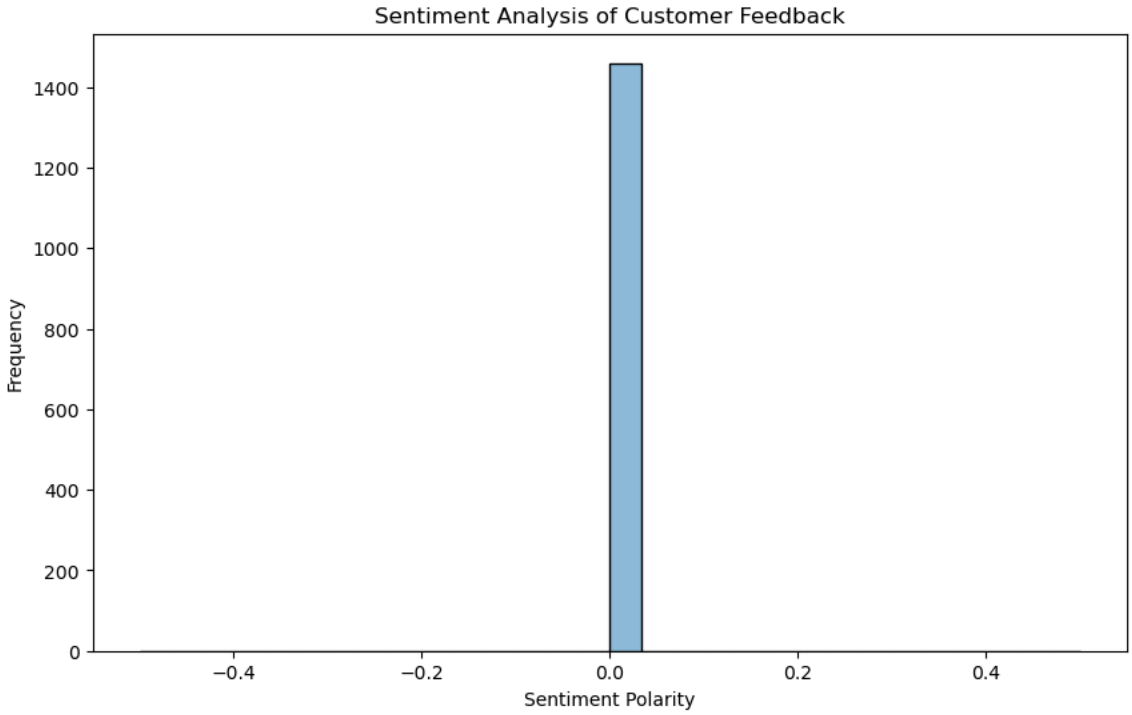
In [688…
```
dataset['Sentiment'] = dataset['SalePrice'].apply(lambda x: TextBlob(str(x)).sentiment.polarity)
```

In [690…
```
dataset['Sentiment']
```

Out[690…
```
YrSold
2008    0.0
2007    0.0
2008    0.0
2006    0.0
2008    0.0
       ...
2007    0.0
2010    0.0
2010    0.0
2010    0.0
2008    0.0
Name: Sentiment, Length: 1460, dtype: float64
```

In [692…
```
# Visualize the sentiment analysis results:
```

In [694…
```
plt.figure(figsize=(10, 6))
sns.histplot(data=dataset, x='Sentiment', bins=30, kde=True)
plt.title('Sentiment Analysis of Customer Feedback')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Frequency')
plt.show()
```



In [696…
```
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
```

Out[696…
```
True
```

In [698…
```
# Customer Feedback Analysis:
```

In [702…
```
customer_feedback = [
    "I love the spacious garage!",
    "The garage is too small for my car.",
    "The garage adds a lot of value to the house."
]

# Initialize the sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Perform sentiment analysis on each feedback
sentiments = [sia.polarity_scores(feedback)['compound'] for feedback in customer_feedback]
```

In [704…
```
customer_feedback
```

Out[704…
```
['I love the spacious garage!',
 'The garage is too small for my car.',
 'The garage adds a lot of value to the house.']
```

In [706…
```
# Storing Final dataset into Excel File:
```

In [714…
```
dataset.to_csv(r"D:\Next hike 3-project-Jan-25\housing_datanew.csv")
```

In [724…
```
dataset.to_csv(r"D:\Next hike 3-project-Jan-25\encoding_housing_data.csv")
```

In [722…
```
# Conclusion:
```

In [ ]:
```
# Key Findings:

# 1. Features like 1stFlrSF,GrLivArea,and GarageArea are having strong relation with the target variable.

# 2. The best performing model is Random Forest Regressor Model with highest R2 & Adjusted_R2 Scores.

# 3. The second best performing model is Linear Regression Model.

# 4. The stacked model performance was impressive becuase of hight accuracy and low error rates.

# 5. The project developed a house price prediction model with strong performance metrics.

# 6. The project effectively addresses the task of house price prediction and contributes as a valuable tool in the dynamic real
```