In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Import the data into a Data Frame

In [2]:
```
df = pd.read_csv('Data/train.csv')
df2 = df.copy()

train=pd.read_csv('Data/train.csv')
test=pd.read_csv('Data/test.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Util
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	Al
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	Al
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	Al
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	Al
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	Al

5 rows × 81 columns

In [4]:
```
train.shape
```

Out[4]: (1460, 81)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 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  LotConfig      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  BldgType       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  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     1452 non-null   object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 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     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
 45  LowQualFinSF   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  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     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  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 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(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [6]:
```
corr = train.corr()
corr.style.background_gradient()
```

Out[6]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearI
Id	1.000000	0.011156	-0.010601	-0.033226	-0.028365	0.012609	-0.012
MSSubClass	0.011156	1.000000	-0.386347	-0.139781	0.032628	-0.059316	0.027
LotFrontage	-0.010601	-0.386347	1.000000	0.426095	0.251646	-0.059213	0.123
LotArea	-0.033226	-0.139781	0.426095	1.000000	0.105806	-0.005636	0.014
OverallQual	-0.028365	0.032628	0.251646	0.105806	1.000000	-0.091932	0.572
OverallCond	0.012609	-0.059316	-0.059213	-0.005636	-0.091932	1.000000	-0.375
YearBuilt	-0.012713	0.027850	0.123349	0.014228	0.572323	-0.375983	1.000
YearRemodAdd	-0.021998	0.040581	0.088866	0.013788	0.550684	0.073741	0.592
MasVnrArea	-0.050298	0.022936	0.193458	0.104160	0.411876	-0.128101	0.315
BsmtFinSF1	-0.005024	-0.069836	0.233633	0.214103	0.239666	-0.046231	0.249
BsmtFinSF2	-0.005968	-0.065649	0.049900	0.111170	-0.059119	0.040229	-0.049
BsmtUnfSF	-0.007940	-0.140759	0.132644	-0.002618	0.308159	-0.136841	0.149
TotalBsmtSF	-0.015415	-0.238518	0.392075	0.260833	0.537808	-0.171098	0.391
1stFlrSF	0.010496	-0.251758	0.457181	0.299475	0.476224	-0.144203	0.281
2ndFlrSF	0.005590	0.307886	0.080177	0.050986	0.295493	0.028942	0.010
LowQualFinSF	-0.044230	0.046474	0.038469	0.004779	-0.030429	0.025494	-0.183
GrLivArea	0.008273	0.074853	0.402797	0.263116	0.593007	-0.079686	0.199
BsmtFullBath	0.002289	0.003491	0.100949	0.158155	0.111098	-0.054942	0.187
BsmtHalfBath	-0.020155	-0.002333	-0.007234	0.048046	-0.040150	0.117821	-0.038
FullBath	0.005587	0.131608	0.198769	0.126031	0.550600	-0.194149	0.468
HalfBath	0.006784	0.177354	0.053532	0.014259	0.273458	-0.060769	0.242
BedroomAbvGr	0.037719	-0.023438	0.263170	0.119690	0.101676	0.012980	-0.070
KitchenAbvGr	0.002951	0.281721	-0.006069	-0.017784	-0.183882	-0.087001	-0.174
TotRmsAbvGrd	0.027239	0.040380	0.352096	0.190015	0.427452	-0.057583	0.095
Fireplaces	-0.019772	-0.045569	0.266639	0.271364	0.396765	-0.023820	0.147
GarageYrBlt	0.000072	0.085072	0.070250	-0.024947	0.547766	-0.324297	0.825
GarageCars	0.016570	-0.040110	0.285691	0.154871	0.600671	-0.185758	0.537
GarageArea	0.017634	-0.098672	0.344997	0.180403	0.562022	-0.151521	0.478
WoodDeckSF	-0.029643	-0.012579	0.088521	0.171698	0.238923	-0.003334	0.224
OpenPorchSF	-0.000477	-0.006100	0.151972	0.084774	0.308819	-0.032589	0.188
EnclosedPorch	0.002889	-0.012037	0.010700	-0.018340	-0.113937	0.070356	-0.387
3SsnPorch	-0.046635	-0.043825	0.070029	0.020423	0.030371	0.025504	0.031
ScreenPorch	0.001330	-0.026030	0.041383	0.043160	0.064886	0.054811	-0.050
PoolArea	0.057044	0.008283	0.206167	0.077672	0.065166	-0.001985	0.004
MiscVal	-0.006242	-0.007683	0.003368	0.038068	-0.031406	0.068777	-0.034

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearI
MoSold	0.021172	-0.013585	0.011200	0.001205	0.070815	-0.003511	0.012
YrSold	0.000712	-0.021407	0.007450	-0.014261	-0.027347	0.043950	-0.013
SalePrice	-0.021917	-0.084284	0.351799	0.263843	0.790982	-0.077856	0.522

In [7]:

```
def find_missing_percent(data):
    """
    Returns dataframe containing the total missing values and percentage of to
tal
    missing values of a column.
    """
    miss_df = pd.DataFrame({'ColumnName':[],'TotalMissingVals':[],'PercentMiss
ing':[]})
    for col in data.columns:
        sum_miss_val = data[col].isnull().sum()
        percent_miss_val = round((sum_miss_val/data.shape[0])*100,2)
        miss_df = miss_df.append(dict(zip(miss_df.columns,[col,sum_miss_val,pe
rcent_miss_val])),ignore_index=True)
    return miss_df
```

In [8]:
```
miss_df = find_missing_percent(df2)
'''Displays columns with missing values'''
display(miss_df[miss_df['PercentMissing']>0.0])
print("\n")

print("Number of columns with missing values:"+(str(miss_df[miss_df['PercentMissing']>0.0].shape[0])))
```

	ColumnName	TotalMissingVals	PercentMissing
3	LotFrontage	259.0	17.74
6	Alley	1369.0	93.77
25	MasVnrType	8.0	0.55
26	MasVnrArea	8.0	0.55
30	BsmtQual	37.0	2.53
31	BsmtCond	37.0	2.53
32	BsmtExposure	38.0	2.60
33	BsmtFinType1	37.0	2.53
35	BsmtFinType2	38.0	2.60
42	Electrical	1.0	0.07
57	FireplaceQu	690.0	47.26
58	GarageType	81.0	5.55
59	GarageYrBlt	81.0	5.55
60	GarageFinish	81.0	5.55
63	GarageQual	81.0	5.55
64	GarageCond	81.0	5.55
72	PoolQC	1453.0	99.52
73	Fence	1179.0	80.75
74	MiscFeature	1406.0	96.30

```
Number of columns with missing values:19
```

In [9]:
```
drop_cols = miss_df[miss_df['PercentMissing'] >70.0].ColumnName.tolist()

print("Number of columns with more than 70%:"+ str(len(drop_cols)))
train = train.drop(drop_cols,axis=1)
#test = test.drop(drop_cols,axis =1)

miss_df = miss_df[miss_df['ColumnName'].isin(train.columns)]
'''Columns to Impute'''
impute_cols = miss_df[miss_df['TotalMissingVals']>0.0].ColumnName.tolist()
miss_df[miss_df['TotalMissingVals']>0.0]
```

```
Number of columns with more than 70%:4
```

Out[9]:

	ColumnName	TotalMissingVals	PercentMissing
3	LotFrontage	259.0	17.74
25	MasVnrType	8.0	0.55
26	MasVnrArea	8.0	0.55
30	BsmtQual	37.0	2.53
31	BsmtCond	37.0	2.53
32	BsmtExposure	38.0	2.60
33	BsmtFinType1	37.0	2.53
35	BsmtFinType2	38.0	2.60
42	Electrical	1.0	0.07
57	FireplaceQu	690.0	47.26
58	GarageType	81.0	5.55
59	GarageYrBlt	81.0	5.55
60	GarageFinish	81.0	5.55
63	GarageQual	81.0	5.55
64	GarageCond	81.0	5.55

In [10]:
```
train.shape
```

Out[10]:
```
(1460, 77)
```

In [11]:
```
train.Neighborhood.value_counts()
```

Out[11]:
```
NAmes      225
CollgCr    150
OldTown    113
Edwards    100
Somerst     86
Gilbert     79
NridgHt     77
Sawyer      74
NWAmes      73
SawyerW     59
BrkSide     58
Crawfor     51
Mitchel     49
NoRidge     41
Timber      38
IDOTRR      37
ClearCr     28
StoneBr     25
SWISU       25
MeadowV     17
Blmngtn     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
```

In [12]:

```
train.describe().transpose()
```

Out[12]:

	count	mean	std	min	25%	50%	75%	
Id	1460.0	730.500000	421.610009	1.0	365.75	730.5	1095.25	
MSSubClass	1460.0	56.897260	42.300571	20.0	20.00	50.0	70.00	
LotFrontage	1201.0	70.049958	24.284752	21.0	59.00	69.0	80.00	
LotArea	1460.0	10516.828082	9981.264932	1300.0	7553.50	9478.5	11601.50	2
OverallQual	1460.0	6.099315	1.382997	1.0	5.00	6.0	7.00	
OverallCond	1460.0	5.575342	1.112799	1.0	5.00	5.0	6.00	
YearBuilt	1460.0	1971.267808	30.202904	1872.0	1954.00	1973.0	2000.00	
YearRemodAdd	1460.0	1984.865753	20.645407	1950.0	1967.00	1994.0	2004.00	
MasVnrArea	1452.0	103.685262	181.066207	0.0	0.00	0.0	166.00	
BsmtFinSF1	1460.0	443.639726	456.098091	0.0	0.00	383.5	712.25	
BsmtFinSF2	1460.0	46.549315	161.319273	0.0	0.00	0.0	0.00	
BsmtUnfSF	1460.0	567.240411	441.866955	0.0	223.00	477.5	808.00	
TotalBsmtSF	1460.0	1057.429452	438.705324	0.0	795.75	991.5	1298.25	
1stFlrSF	1460.0	1162.626712	386.587738	334.0	882.00	1087.0	1391.25	
2ndFlrSF	1460.0	346.992466	436.528436	0.0	0.00	0.0	728.00	
LowQualFinSF	1460.0	5.844521	48.623081	0.0	0.00	0.0	0.00	
GrLivArea	1460.0	1515.463699	525.480383	334.0	1129.50	1464.0	1776.75	
BsmtFullBath	1460.0	0.425342	0.518911	0.0	0.00	0.0	1.00	
BsmtHalfBath	1460.0	0.057534	0.238753	0.0	0.00	0.0	0.00	
FullBath	1460.0	1.565068	0.550916	0.0	1.00	2.0	2.00	
HalfBath	1460.0	0.382877	0.502885	0.0	0.00	0.0	1.00	
BedroomAbvGr	1460.0	2.866438	0.815778	0.0	2.00	3.0	3.00	
KitchenAbvGr	1460.0	1.046575	0.220338	0.0	1.00	1.0	1.00	
TotRmsAbvGrd	1460.0	6.517808	1.625393	2.0	5.00	6.0	7.00	
Fireplaces	1460.0	0.613014	0.644666	0.0	0.00	1.0	1.00	
GarageYrBlt	1379.0	1978.506164	24.689725	1900.0	1961.00	1980.0	2002.00	
GarageCars	1460.0	1.767123	0.747315	0.0	1.00	2.0	2.00	
GarageArea	1460.0	472.980137	213.804841	0.0	334.50	480.0	576.00	
WoodDeckSF	1460.0	94.244521	125.338794	0.0	0.00	0.0	168.00	
OpenPorchSF	1460.0	46.660274	66.256028	0.0	0.00	25.0	68.00	
EnclosedPorch	1460.0	21.954110	61.119149	0.0	0.00	0.0	0.00	
3SsnPorch	1460.0	3.409589	29.317331	0.0	0.00	0.0	0.00	
ScreenPorch	1460.0	15.060959	55.757415	0.0	0.00	0.0	0.00	
PoolArea	1460.0	2.758904	40.177307	0.0	0.00	0.0	0.00	
MiscVal	1460.0	43.489041	496.123024	0.0	0.00	0.0	0.00	

	count	mean	std	min	25%	50%	75%
MoSold	1460.0	6.321918	2.703626	1.0	5.00	6.0	8.00
YrSold	1460.0	2007.815753	1.328095	2006.0	2007.00	2008.0	2009.00
SalePrice	1460.0	180921.195890	79442.502883	34900.0	129975.00	163000.0	214000.00

Data pre-processing

We will build a pipeline to do some of the following tasks:

- Missing data
- Feature scaling (important for certain model such as Gradient Descent based models)
- Categorical feature encoding
- Outlier removal
- Transformation
- Custom processing

BldgType: Type of dwelling

```
1Fam Single-family Detached
2FmCon   Two-family Conversion; originally built as one-family dwelling
Duplx    Duplex
TwnhsE   Townhouse End Unit
TwnhsI   Townhouse Inside Unit
```

HouseStyle: Style of dwelling

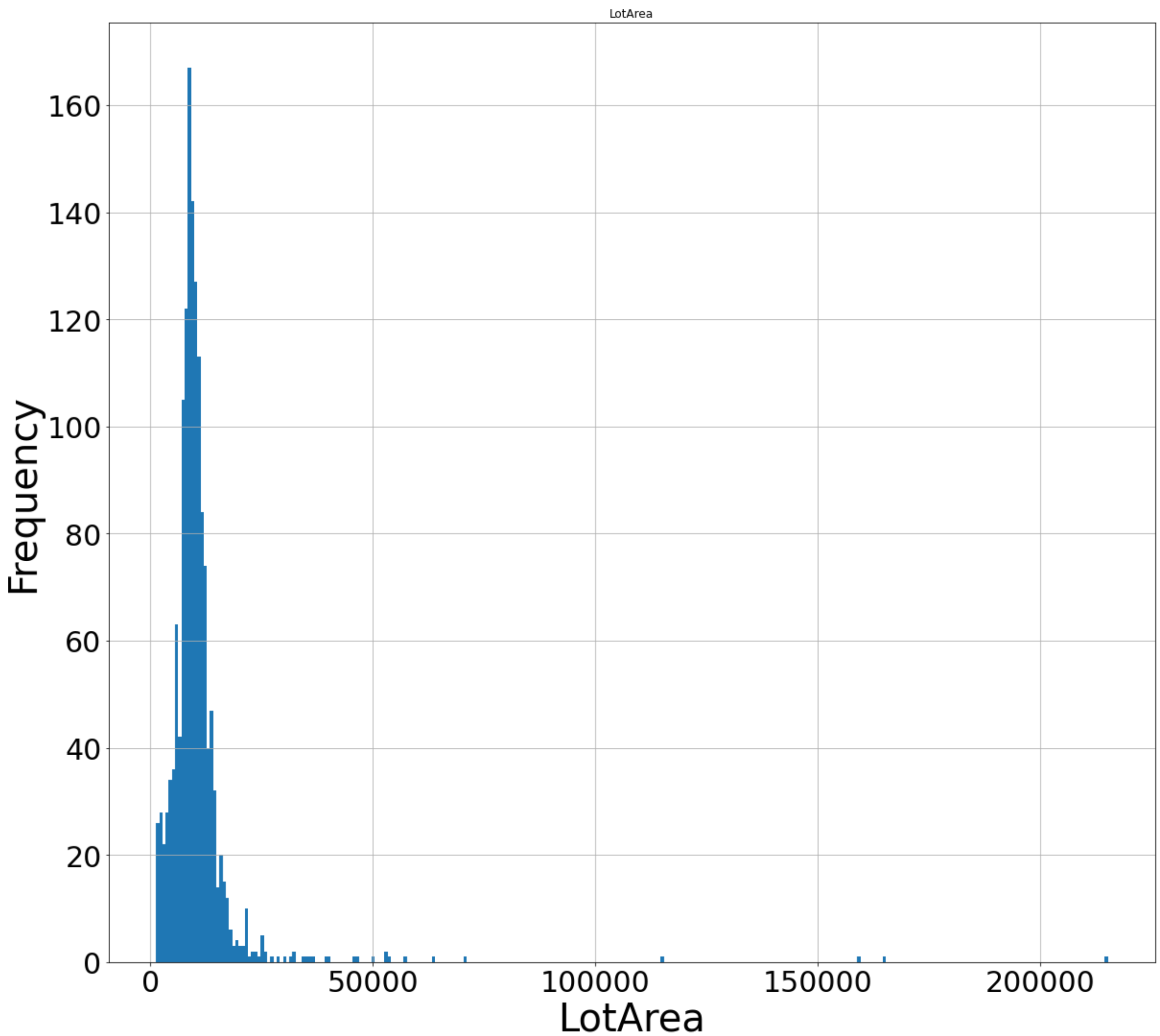
```
1Story   One story
1.5Fin   One and one-half story: 2nd level finished
1.5Unf   One and one-half story: 2nd level unfinished
2Story   Two story
2.5Fin   Two and one-half story: 2nd level finished
2.5Unf   Two and one-half story: 2nd level unfinished
SFoyer   Split Foyer
SLvl Split Level
```

In [13]:
```
def plot_histogram(train, col1, col2, cols_list, last_one =False):
    """
    Plot the histogram for the numerical columns. The bin width
    is calculated by Freedman Diaconis Rule and Sturges rule.

    Freedman-Diaconis Rule:
    Freedman-Diaconis Rule is a rule to find the optimal number of bins.
    Bin width: (2 * IQR)/(N^1/3)
    N - Size of the data
    Number of bins : (Range/ bin-width)

    Disadvantage: The IQR might be zero for certain columns. In
    that case the bin width might be equal to infinity. In that case
    the actual range of the data is returned as bin width.

    Sturges Rule:
    Sturges Rule is a rule to find the optimal number of bins.
    Bin width: (Range/ bin-width)
    N - Size of the data
    Number of bins : ceil(log2(N))+1

    """
    if(col1 in cols_list):
        freq1, bin_edges1 = np.histogram(train[col1],bins='sturges')
    else:
        freq1, bin_edges1 = np.histogram(train[col1],bins='fd')
    if(col2 in cols_list):
        freq2, bin_edges2 = np.histogram(train[col2],bins='sturges')
    else:
        freq2, bin_edges2 = np.histogram(train[col2],bins='fd')

    if(last_one!=True):
        plt.figure(figsize=(45,18))
        ax1 = plt.subplot(1,2,1)
        ax1.set_title(col1,fontsize=45)
        ax1.set_xlabel(col1,fontsize=40)
        ax1.set_ylabel('Frequency',fontsize=40)
        train[col1].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=3
0)

    else:
        plt.figure(figsize=(20,10))
        ax1 = plt.subplot(1,2,1)
        ax1.set_title(col1,fontsize=25)
        ax1.set_xlabel(col1,fontsize=20)
        ax1.set_ylabel('Frequency',fontsize=20)
        train[col1].hist(bins=bin_edges1,ax = ax1, xlabelsize=15, ylabelsize=1
5)

    if(last_one != True):
        ax2 = plt.subplot(1,2,2)
        ax2.set_title(col2,fontsize=45)
        ax2.set_xlabel(col2,fontsize=40)
        ax2.set_ylabel('Frequency',fontsize=40)
        train[col2].hist(bins=bin_edges2, ax = ax2, xlabelsize=30, ylabelsize=
30)
```

In [14]:
```
numeric_cols = train.select_dtypes(['float','int']).columns
```

In [15]:
```
freq1, bin_edges1=np.histogram(train.LotArea, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('LotArea',fontsize=45)
ax1.set_xlabel('LotArea',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['LotArea']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=30
)
```

Out[15]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF18BC4C0>],
      dtype=object)
```

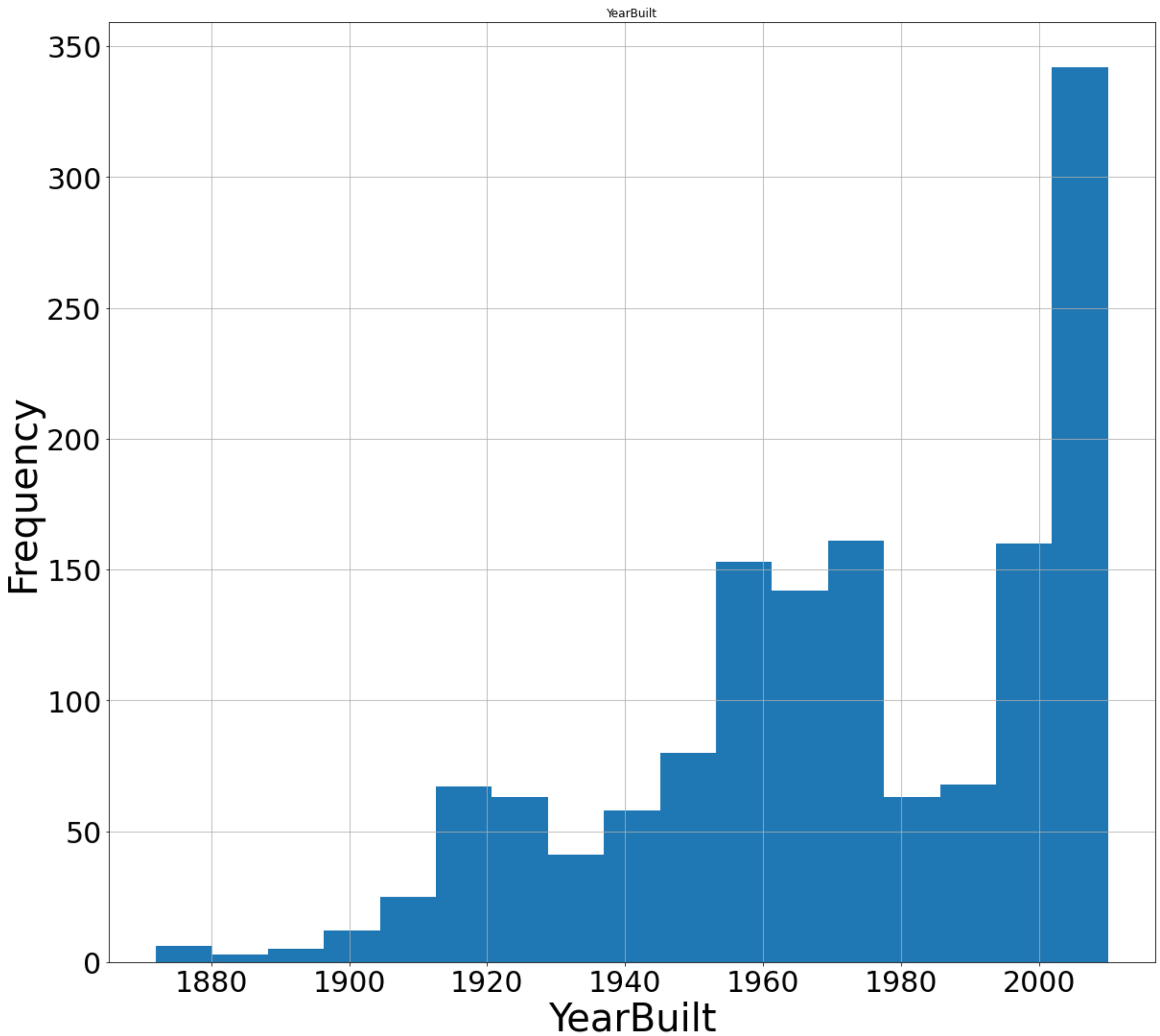


In [16]:
```
# Dropping LotArea greater than 50000 to remove outlier

train = train[train.LotArea <= 50000].copy()
train.shape
```

Out[16]:
```
(1449, 77)
```

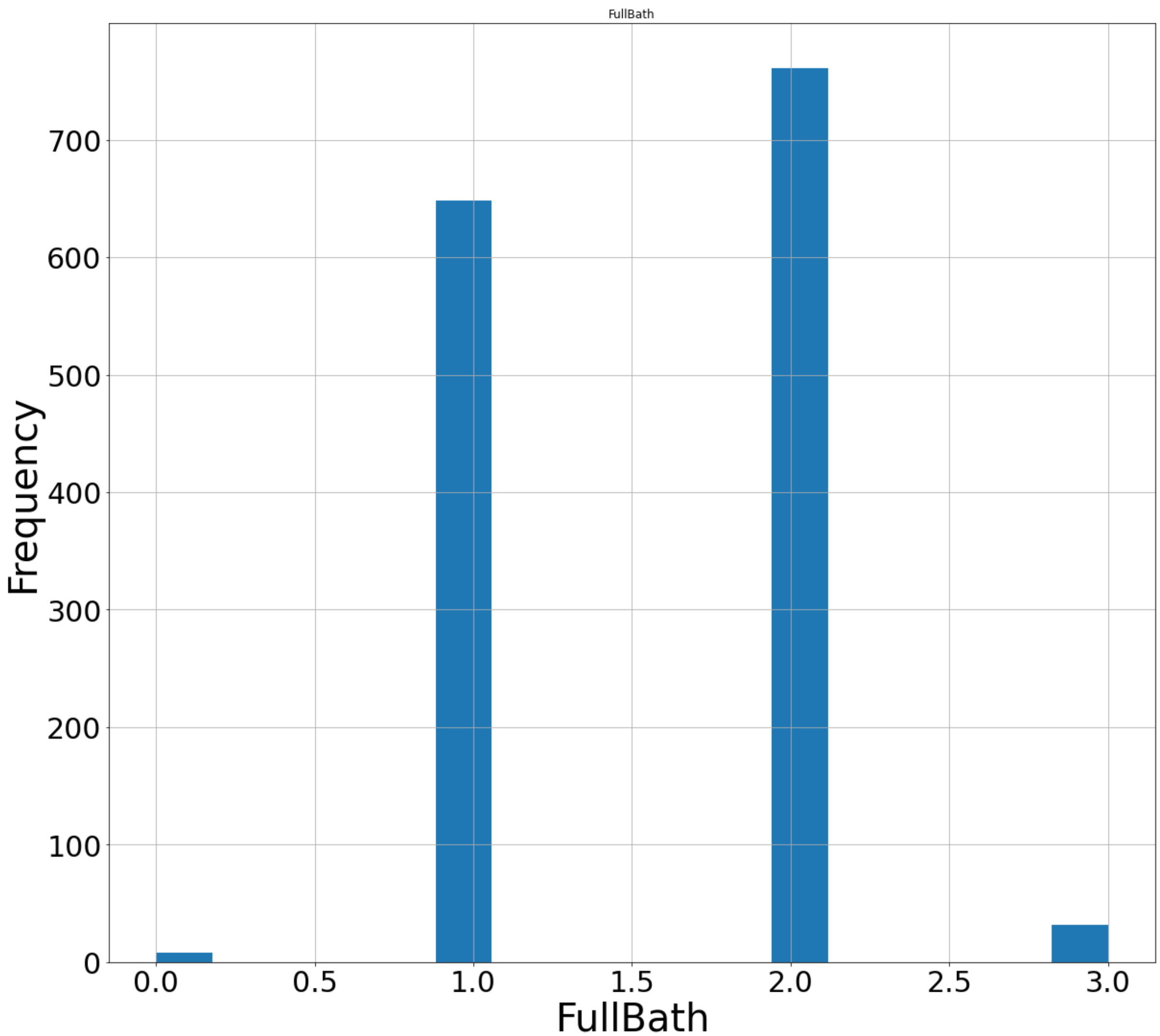
In [17]:
```
freq1, bin_edges1=np.histogram(train.YearBuilt, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('YearBuilt',fontsize=45)
ax1.set_xlabel('YearBuilt',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['YearBuilt']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=
30)
```

Out[17]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF1AC17F0>],
      dtype=object)
```



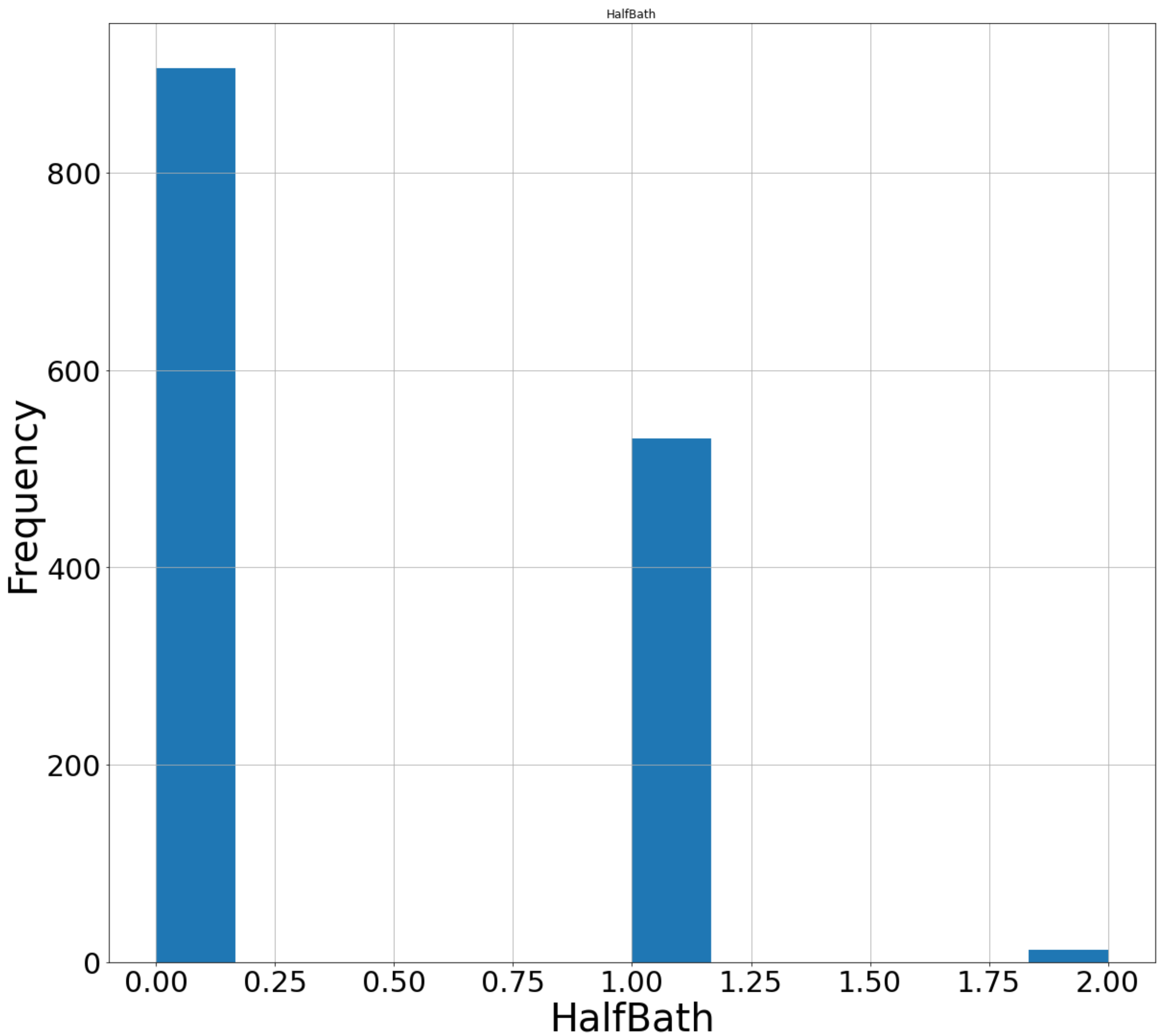
In [18]:
```
freq1, bin_edges1=np.histogram(train.FullBath, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('FullBath',fontsize=45)
ax1.set_xlabel('FullBath',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['FullBath']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=3
0)
```

Out[18]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF19D10A0>],
      dtype=object)
```



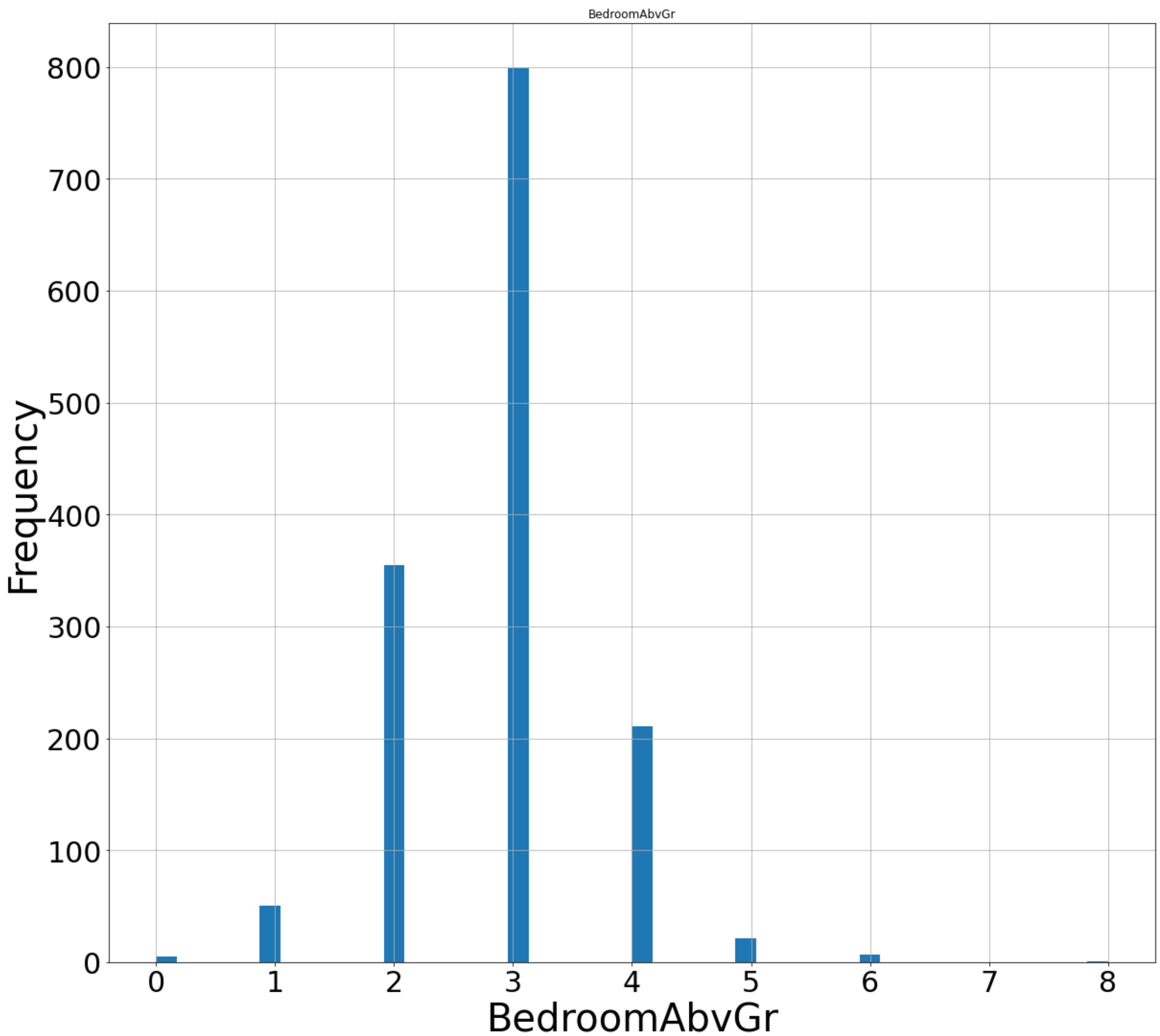
In [19]:
```
freq1, bin_edges1=np.histogram(train.HalfBath, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('HalfBath',fontsize=45)
ax1.set_xlabel('HalfBath',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['HalfBath']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=3
0)
```

Out[19]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF24639A0>],
      dtype=object)
```



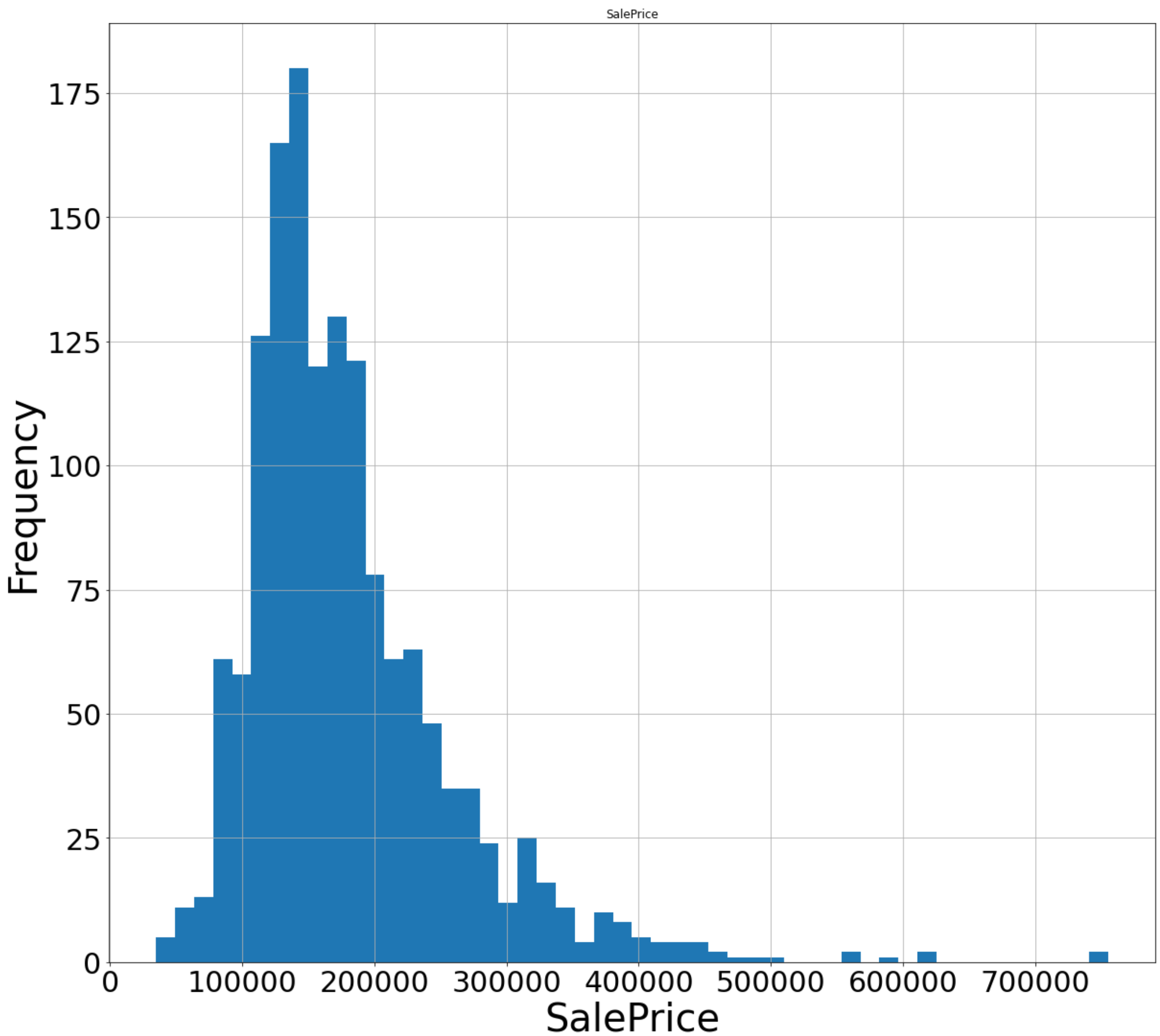
In [20]:
```
freq1, bin_edges1=np.histogram(train.BedroomAbvGr, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('BedroomAbvGr',fontsize=45)
ax1.set_xlabel('BedroomAbvGr',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['BedroomAbvGr']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=30)
```

Out[20]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF1B5C880>],
      dtype=object)
```



In [21]:
```
freq1, bin_edges1=np.histogram(train.SalePrice, bins='fd')

plt.figure(figsize=(45,18))
ax1 = plt.subplot(1,2,1)
ax1.set_title('SalePrice',fontsize=45)
ax1.set_xlabel('SalePrice',fontsize=40)
ax1.set_ylabel('Frequency',fontsize=40)
train[['SalePrice']].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=
30)
```

Out[21]:
```
array([&lt;matplotlib.axes._subplots.AxesSubplot object at 0x0000021CF2377C10&gt;],
      dtype=object)
```



In [22]:
```
df3=train[['LotArea','BldgType','HouseStyle','YearBuilt','FullBath','HalfBath'
,'BedroomAbvGr']].copy()
```

In [23]:
```
df3.describe()
```

Out[23]:

	LotArea	YearBuilt	FullBath	HalfBath	BedroomAbvGr
count	1449.000000	1449.000000	1449.000000	1449.000000	1449.000000
mean	9867.879917	1971.242926	1.563837	0.383023	2.868185
std	4578.300353	30.271932	0.548947	0.503045	0.813314
min	1300.000000	1872.000000	0.000000	0.000000	0.000000
25%	7500.000000	1954.000000	1.000000	0.000000	2.000000
50%	9450.000000	1973.000000	2.000000	0.000000	3.000000
75%	11500.000000	2000.000000	2.000000	1.000000	3.000000
max	46589.000000	2010.000000	3.000000	2.000000	8.000000

In [24]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1449 entries, 0 to 1459
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   LotArea       1449 non-null   int64 
 1   BldgType      1449 non-null   object
 2   HouseStyle    1449 non-null   object
 3   YearBuilt     1449 non-null   int64 
 4   FullBath      1449 non-null   int64 
 5   HalfBath      1449 non-null   int64 
 6   BedroomAbvGr  1449 non-null   int64 
dtypes: int64(5), object(2)
memory usage: 90.6+ KB
```

In [25]:
```
df3.isnull().sum()
```

Out[25]:
```
LotArea         0
BldgType        0
HouseStyle      0
YearBuilt       0
FullBath        0
HalfBath        0
BedroomAbvGr    0
dtype: int64
```

In [26]:
```
df3.corr()
```
Out[26]:

	LotArea	YearBuilt	FullBath	HalfBath	BedroomAbvGr
LotArea	1.000000	0.037304	0.199742	0.085728	0.265837
YearBuilt	0.037304	1.000000	0.469592	0.240004	-0.071460
FullBath	0.199742	0.469592	1.000000	0.137725	0.358389
HalfBath	0.085728	0.240004	0.137725	1.000000	0.229830
BedroomAbvGr	0.265837	-0.071460	0.358389	0.229830	1.000000

In [27]:
```
df3.head()
```
Out[27]:

	LotArea	BldgType	HouseStyle	YearBuilt	FullBath	HalfBath	BedroomAbvGr
0	8450	1Fam	2Story	2003	2	1	3
1	9600	1Fam	1Story	1976	2	0	3
2	11250	1Fam	2Story	2001	2	1	3
3	9550	1Fam	2Story	1915	1	0	3
4	14260	1Fam	2Story	2000	2	1	4

In [28]:
```
one_hot_df = pd.get_dummies(df3[['BldgType','HouseStyle']]) #
df3 = df3.drop(['BldgType','HouseStyle'], axis=1) # Drop column as it is now e
ncoded
df3 = df3.join(one_hot_df) # Join the encoded df
print(df3.columns)
df3.tail()
# and encoding happens
```

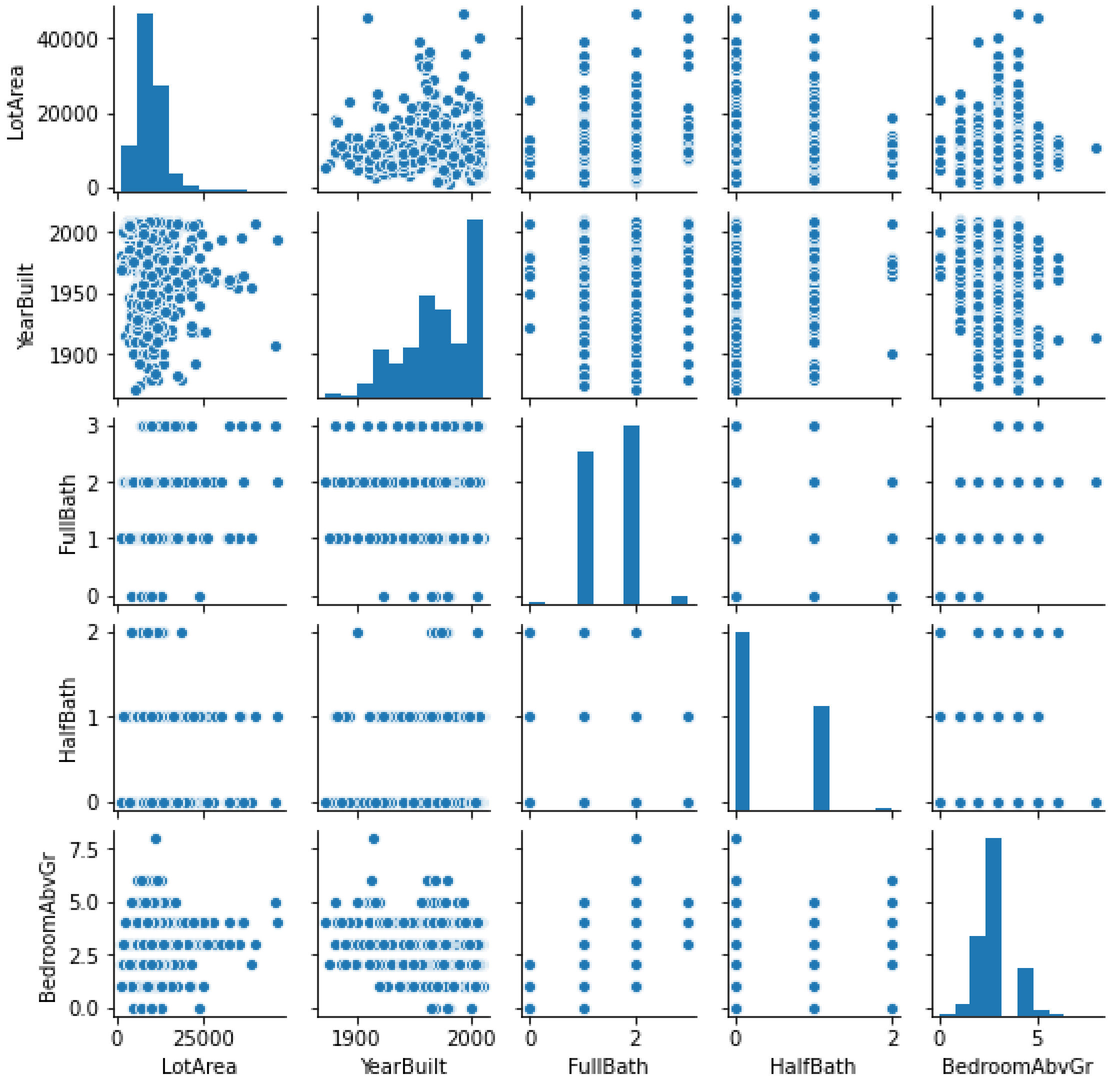
```
Index(['LotArea', 'YearBuilt', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'BldgType_1Fam', 'BldgType_2fmCon', 'BldgType_Duplex', 'BldgType_Twnh
s',
       'BldgType_TwnhsE', 'HouseStyle_1.5Fin', 'HouseStyle_1.5Unf',
       'HouseStyle_1Story', 'HouseStyle_2.5Fin', 'HouseStyle_2.5Unf',
       'HouseStyle_2Story', 'HouseStyle_SFoyer', 'HouseStyle_SLvl'],
      dtype='object')
```

Out[28]:

	LotArea	YearBuilt	FullBath	HalfBath	BedroomAbvGr	BldgType_1Fam	BldgType_2fmCon
1455	7917	1999	2	1	3	1	0
1456	13175	1978	2	0	3	1	0
1457	9042	1941	2	0	4	1	0
1458	9717	1950	1	0	2	1	0
1459	9937	1965	1	1	3	1	0

In [29]:
```
sns.pairplot(df3[['LotArea', 'YearBuilt', 'FullBath', 'HalfBath',
       'BedroomAbvGr']], height=1.5)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x21cec625340>



In [30]:
```
# split data into X and Y dataframes
X = df3.copy() # independent variables
Y = train['SalePrice'].copy() # dependent variable
```

In [31]:
```
# Run regression using statsmodels
import statsmodels.api as sm
import math

X = sm.add_constant(X) # required if a value for alpha is expected
est = sm.OLS(Y,X).fit() # fit model
predictions = est.predict() # get predicted values
print(est.summary())
print("\nAverage error: {:.2f}.".format(math.sqrt(est.mse_resid)))
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:              SalePrice   R-squared:                       0.54
1
Model:                            OLS   Adj. R-squared:                  0.53
5
Method:                 Least Squares   F-statistic:                     105.
3
Date:                Mon, 28 Jun 2021   Prob (F-statistic):          4.06e-22
8
Time:                        22:48:45   Log-Likelihood:                -1782
6.
No. Observations:                1449   AIC:                         3.569e+0
4
Df Residuals:                    1432   BIC:                         3.578e+0
4
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const             -1.256e+06      1e+05    -12.520      0.000   -1.45e+06   -
1.06e+06
LotArea               4.4416      0.362     12.267      0.000       3.731
5.152
YearBuilt           867.2988     69.385     12.500      0.000     731.191
1003.407
FullBath             5.27e+04   3636.254     14.492      0.000    4.56e+04
5.98e+04
HalfBath            2.459e+04   3999.961      6.147      0.000    1.67e+04
3.24e+04
BedroomAbvGr      -2993.4791   2277.696     -1.314      0.189   -7461.458
1474.500
BldgType_1Fam     -2.282e+05   2.05e+04    -11.148      0.000   -2.68e+05   -
1.88e+05
BldgType_2fmCon   -2.467e+05   2.07e+04    -11.931      0.000   -2.87e+05   -
2.06e+05
BldgType_Duplex   -2.825e+05   2.07e+04    -13.646      0.000   -3.23e+05   -
2.42e+05
BldgType_Twnhs    -2.638e+05   2.21e+04    -11.926      0.000   -3.07e+05
-2.2e+05
BldgType_TwnhsE   -2.346e+05   2.18e+04    -10.764      0.000   -2.77e+05   -
1.92e+05
HouseStyle_1.5Fin -1.644e+05   1.29e+04    -12.721      0.000    -1.9e+05   -
1.39e+05
HouseStyle_1.5Unf -1.596e+05   1.82e+04     -8.752      0.000   -1.95e+05   -
1.24e+05
HouseStyle_1Story -1.676e+05   1.49e+04    -11.281      0.000   -1.97e+05   -
1.38e+05
HouseStyle_2.5Fin  -1.01e+05   1.99e+04     -5.065      0.000    -1.4e+05   -
6.19e+04
HouseStyle_2.5Unf -1.409e+05   1.77e+04     -7.942      0.000   -1.76e+05   -
1.06e+05
```

```
HouseStyle_2Story -1.726e+05   1.39e+04    -12.393      0.000      -2e+05   -
1.45e+05
HouseStyle_SFoyer -1.685e+05   1.75e+04     -9.632      0.000   -2.03e+05   -
1.34e+05
HouseStyle_SLvl   -1.812e+05   1.58e+04    -11.467      0.000   -2.12e+05   
-1.5e+05
==============================================================================
=
Omnibus:                      640.645   Durbin-Watson:                   1.98
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5721.20
8
Skew:                           1.831   Prob(JB):                         0.0
0
Kurtosis:                      12.019   Cond. No.                     2.08e+2
0
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The smallest eigenvalue is 4.08e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Average error: 53600.31.
```

Conclusion

1. 19 columns were identified as having NULL values, out of which 4 columns have more than 70 percent were NULL, which were dropped
2. The following attributes were chosen based on our initial correlation analysis; which are some of the common attributest between test datasets LotArea , BldgType, HouseStyle , YearBuilt, FullBath, HalfBath , BedroomAbvGr
3. Another correlation analysis was performed among the selected attributes to avoid multicollinearity
4. In order to avoid outliers, LotArea greater than 50,000 sq.ft. were eliminated
5. Test datasets are gathered from Delaware- Bear, Delaware- Newark, Delaware-Wilmington and the latest data from Iowa- Ames.
6. Full Bath, Half Bath, Year Built and lot area are the most significant predictors in the model
7. With this prediction model, predicted house price is off by an average of $53,600
8. The R^2 statistic shows how well the model explains SalePrice.
9. In this model, since R^2 and Adjusted R^2 are close, model is not overfit.