Project Title : House Prices Prediction

Problem Statement

```
The aim of this project is to evaluate proficiency in end-to-end Machine
Learning Projects.
```

Dataset

```
The dataset for this assignment is the House Prices Dataset. It contains
various features of the houses. The sale price is the target variable.
```

Data description

MSSubClass: Identifies the type of dwelling involved in the sale.

```
        20  1-STORY 1946 & NEWER ALL STYLES
        30  1-STORY 1945 & OLDER
        40  1-STORY W/FINISHED ATTIC ALL AGES
        45  1-1/2 STORY - UNFINISHED ALL AGES
        50  1-1/2 STORY FINISHED ALL AGES
        60  2-STORY 1946 & NEWER
        70  2-STORY 1945 & OLDER
        75  2-1/2 STORY ALL AGES
        80  SPLIT OR MULTI-LEVEL
        85  SPLIT FOYER
        90  DUPLEX - ALL STYLES AND AGES
       120  1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150  1-1/2 STORY PUD - ALL AGES
       160  2-STORY PUD - 1946 & NEWER
       180  PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190  2 FAMILY CONVERSION - ALL STYLES AND AGES
```

MSZoning: Identifies the general zoning classification of the sale.

Typesetting math: 0%

```
A    Agriculture
C    Commercial
FV   Floating Village Residential
I    Industrial
RH   Residential High Density
RL   Residential Low Density
RP   Residential Low Density Park
RM   Residential Medium Density
```

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

```
Grvl Gravel
Pave Paved
```

Alley: Type of alley access to property

```
Grvl Gravel
Pave Paved
NA   No alley access
```

LotShape: General shape of property

```
Reg  Regular
IR1  Slightly irregular
IR2  Moderately Irregular
IR3  Irregular
```

LandContour: Flatness of the property

```
   Lvl  Near Flat/Level
   Bnk  Banked - Quick and significant rise from street grade to build
ing
   HLS  Hillside - Significant slope from side to side
   Low  Depression
```

Utilities: Type of utilities available

```
AllPub       All public Utilities (E,G,W,& S)
NoSewr       Electricity, Gas, and Water (Septic Tank)
NoSeWa       Electricity and Gas Only
ELO  Electricity only
```

LotConfig: Lot configuration

Typesetting math: 0%

```
Inside       Inside lot
Corner       Corner lot
CulDSac      Cul-de-sac
FR2  Frontage on 2 sides of property
FR3  Frontage on 3 sides of property
```

LandSlope: Slope of property

```
Gtl  Gentle slope
Mod  Moderate Slope
Sev  Severe Slope
```

Neighborhood: Physical locations within Ames city limits

```
Blmngtn      Bloomington Heights
Blueste      Bluestem
BrDale       Briardale
BrkSide      Brookside
ClearCr      Clear Creek
CollgCr      College Creek
Crawfor      Crawford
Edwards      Edwards
Gilbert      Gilbert
IDOTRR       Iowa DOT and Rail Road
MeadowV      Meadow Village
Mitchel      Mitchell
Names        North Ames
NoRidge      Northridge
NPkVill      Northpark Villa
NridgHt      Northridge Heights
NWAmes       Northwest Ames
OldTown      Old Town
SWISU        South & West of Iowa State University
Sawyer       Sawyer
SawyerW      Sawyer West
Somerst      Somerset
StoneBr      Stone Brook
Timber       Timberland
Veenker      Veenker
```

Condition1: Proximity to various conditions

Typesetting math: 0%

```
   Artery       Adjacent to arterial street
   Feedr        Adjacent to feeder street
   Norm Normal
   RRNn Within 200' of North-South Railroad
   RRAn Adjacent to North-South Railroad
   PosN Near positive off-site feature--park, greenbelt, etc.
   PosA Adjacent to postive off-site feature
   RRNe Within 200' of East-West Railroad
   RRAe Adjacent to East-West Railroad
```

Condition2: Proximity to various conditions (if more than one is present)

```
   Artery       Adjacent to arterial street
   Feedr        Adjacent to feeder street
   Norm Normal
   RRNn Within 200' of North-South Railroad
   RRAn Adjacent to North-South Railroad
   PosN Near positive off-site feature--park, greenbelt, etc.
   PosA Adjacent to postive off-site feature
   RRNe Within 200' of East-West Railroad
   RRAe Adjacent to East-West Railroad
```

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
```

OverallQual: Rates the overall material and finish of the house

Typesetting math: 0%

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

OverallCond: Rates the overall condition of the house

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

```
Flat Flat
Gable        Gable
Gambrel      Gabrel (Barn)
Hip  Hip
Mansard      Mansard
Shed Shed
```

RoofMatl: Roof material

```
ClyTile      Clay or Tile
CompShg      Standard (Composite) Shingle
Membran      Membrane
Metal        Metal
Roll Roll
Tar&Grv      Gravel & Tar
WdShake      Wood Shakes
WdShngl      Wood Shingles
```

Typesetting math: 0%

Exterior1st: Exterior covering on house

```
AsbShng      Asbestos Shingles
AsphShn      Asphalt Shingles
BrkComm      Brick Common
BrkFace      Brick Face
CBlock       Cinder Block
CemntBd      Cement Board
HdBoard      Hard Board
ImStucc      Imitation Stucco
MetalSd      Metal Siding
Other        Other
Plywood      Plywood
PreCast      PreCast
Stone        Stone
Stucco       Stucco
VinylSd      Vinyl Siding
Wd Sdng      Wood Siding
WdShing      Wood Shingles
```

Exterior2nd: Exterior covering on house (if more than one material)

```
AsbShng      Asbestos Shingles
AsphShn      Asphalt Shingles
BrkComm      Brick Common
BrkFace      Brick Face
CBlock       Cinder Block
CemntBd      Cement Board
HdBoard      Hard Board
ImStucc      Imitation Stucco
MetalSd      Metal Siding
Other        Other
Plywood      Plywood
PreCast      PreCast
Stone        Stone
Stucco       Stucco
VinylSd      Vinyl Siding
Wd Sdng      Wood Siding
WdShing      Wood Shingles
```

MasVnrType: Masonry veneer type

```
BrkCmn       Brick Common
BrkFace      Brick Face
CBlock       Cinder Block
None None
Stone        Stone
```

MasVnrArea: Masonry veneer area in square feet

Typesetting math: 0%

ExterQual: Evaluates the quality of the material on the exterior

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

ExterCond: Evaluates the present condition of the material on the exterior

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

Foundation: Type of foundation

```
BrkTil       Brick & Tile
CBlock       Cinder Block
PConc        Poured Contrete
Slab Slab
Stone        Stone
Wood Wood
```

BsmtQual: Evaluates the height of the basement

```
Ex   Excellent (100+ inches)
Gd   Good (90-99 inches)
TA   Typical (80-89 inches)
Fa   Fair (70-79 inches)
Po   Poor (<70 inches
NA   No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
Ex   Excellent
Gd   Good
TA   Typical - slight dampness allowed
Fa   Fair - dampness or some cracking or settling
Po   Poor - Severe cracking, settling, or wetness
NA   No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
       Gd   Good Exposure
       Av   Average Exposure (split levels or foyers typically score avera
    ge or above)
       Mn   Mimimum Exposure
       No   No Exposure
       NA   No Basement
```

BsmtFinType1: Rating of basement finished area

```
       GLQ  Good Living Quarters
       ALQ  Average Living Quarters
       BLQ  Below Average Living Quarters
       Rec  Average Rec Room
       LwQ  Low Quality
       Unf  Unfinshed
       NA   No Basement
```

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

```
       GLQ  Good Living Quarters
       ALQ  Average Living Quarters
       BLQ  Below Average Living Quarters
       Rec  Average Rec Room
       LwQ  Low Quality
       Unf  Unfinshed
       NA   No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

```
       Floor        Floor Furnace
       GasA Gas forced warm air furnace
       GasW Gas hot water or steam heat
       Grav Gravity furnace
       OthW Hot water or steam heat other than gas
       Wall Wall furnace
```

HeatingQC: Heating quality and condition

Typesetting math: 0%

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

CentralAir: Central air conditioning

```
N    No
Y    Yes
```

Electrical: Electrical system

```
SBrkr        Standard Circuit Breakers & Romex
FuseA        Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF        60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP        60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix  Mixed
```

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
```

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Typesetting math: 0%

Functional: Home functionality (Assume typical unless deductions are warranted)

```
       Typ  Typical Functionality
       Min1 Minor Deductions 1
       Min2 Minor Deductions 2
       Mod  Moderate Deductions
       Maj1 Major Deductions 1
       Maj2 Major Deductions 2
       Sev  Severely Damaged
       Sal  Salvage only
```

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
       Ex   Excellent - Exceptional Masonry Fireplace
       Gd   Good - Masonry Fireplace in main level
       TA   Average - Prefabricated Fireplace in main living area or Mason
    ry Fireplace in basement
       Fa   Fair - Prefabricated Fireplace in basement
       Po   Poor - Ben Franklin Stove
       NA   No Fireplace
```

GarageType: Garage location

```
       2Types       More than one type of garage
       Attchd       Attached to home
       Basment      Basement Garage
       BuiltIn      Built-In (Garage part of house - typically has room ab
    ove garage)
       CarPort      Car Port
       Detchd       Detached from home
       NA   No Garage
```

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

```
       Fin  Finished
       RFn  Rough Finished
       Unf  Unfinished
       NA   No Garage
```

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Typesetting math: 0%

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

GarageCond: Garage condition

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

PavedDrive: Paved driveway

```
Y    Paved
P    Partial Pavement
N    Dirt/Gravel
```

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
NA   No Pool
```

Fence: Fence quality

```
GdPrv        Good Privacy
MnPrv        Minimum Privacy
GdWo Good Wood
MnWw Minimum Wood/Wire
NA   No Fence
```

Typesetting math: 0%

MiscFeature: Miscellaneous feature not covered in other categories

```
Elev Elevator
Gar2 2nd Garage (if not described in garage section)
```

Instructions

Typesetting math: 0%

```
1) Submission is to be made in .ipynb format.
*Should have a table of contents.
*Should have the necessary documentation.
*Important observations to be noted.

2)Perform exploratory data analysis (EDA) to gain insights about the d
ata.
*Univariate Analysis
    i. Visualize the distribution of the continuous variables.
    ii. Visualize the categories for the categorical variables.
*Bi-variate Analysis
    i. Visualize the relationship between the continuous variables and
the target variable
    ii. Visualize the relationship between the categorical variables a
nd the target variable
    iii. Calculate the correlation coefficients between the pairs of c
ontinuous-continuous, continuous-categorical, and categorical-categori
cal variables.
*Summarize descriptive statistics of the variables.

3)Preprocess the data as necessary to prepare it for modeling. This sh
ould include handling missing values, dealing with outliers, and trans
forming the data as necessary.

4)Apply at least the following types of models:
*A linear model with regularization
*A Bagging model (e.g. Random Forest, Extra trees, etc.)
*A boosting model (e.g. Adaboost, Gradient boosting, etc.)

5)Use cross-validation techniques.

6)Tune the hyperparameters for all the models.

7)Try feature selection techniques.

8)Use stacking of multiple models to improve performance.

9)Check the residuals for homoscedasticity and normal distribution.

10)Analyze the feature importances from different models to gain insig
hts about which features are more important for prediction.
```

Typesetting math: 0%

In [1]:
```
#importing Libraries
import pandas as pd
import numpy as np

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
#Loading Dataset
data=pd.read_csv('data.csv')
print('Shape of data:',data.shape)
```

```
Shape of data: (1460, 81)
```

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

Out[3]:

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

Typesetting math: 0%

In [4]:
```
data.columns
```

Out[4]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAd
d',
       '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', 'FullBat
h',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageTyp
e',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQua
l',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Typesetting math: 0%

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

Typesetting math: 0%

```
<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     588 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  
```

Typesetting math: 0%

```
 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]:

```
#taking all the numerical variables having datatype integers in one list
data_integer=data.select_dtypes(include=['int64']).columns.tolist()
print('Total number of Integers: ',len(data_integer))
print(data_integer)
```

```
Total number of Integers:  35
['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'Y
earRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlr
SF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fire
places', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPo
rch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold',
'SalePrice']
```

In [7]:

```
#taking all the float variables having datatype float in one list
data_float=data.select_dtypes(include=['float64']).columns.tolist()
print('Total number of Floats: ',len(data_float))
print(data_float)
```

```
Total number of Floats:  3
['LotFrontage', 'MasVnrArea', 'GarageYrBlt']
```

Typesetting math: 0%

In [8]:
```
#taking all the categorical variables in one list
data_categorical=data.select_dtypes(include=['object']).columns.tolist()
print('Total number of categories: ',len(data_categorical))
print(data_categorical)
```

```
Total number of categories:  43
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotC
onfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrT
ype', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtEx
posure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAi
r', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType',
'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence',
'MiscFeature', 'SaleType', 'SaleCondition']
```

In [9]:
```
data.describe()
```

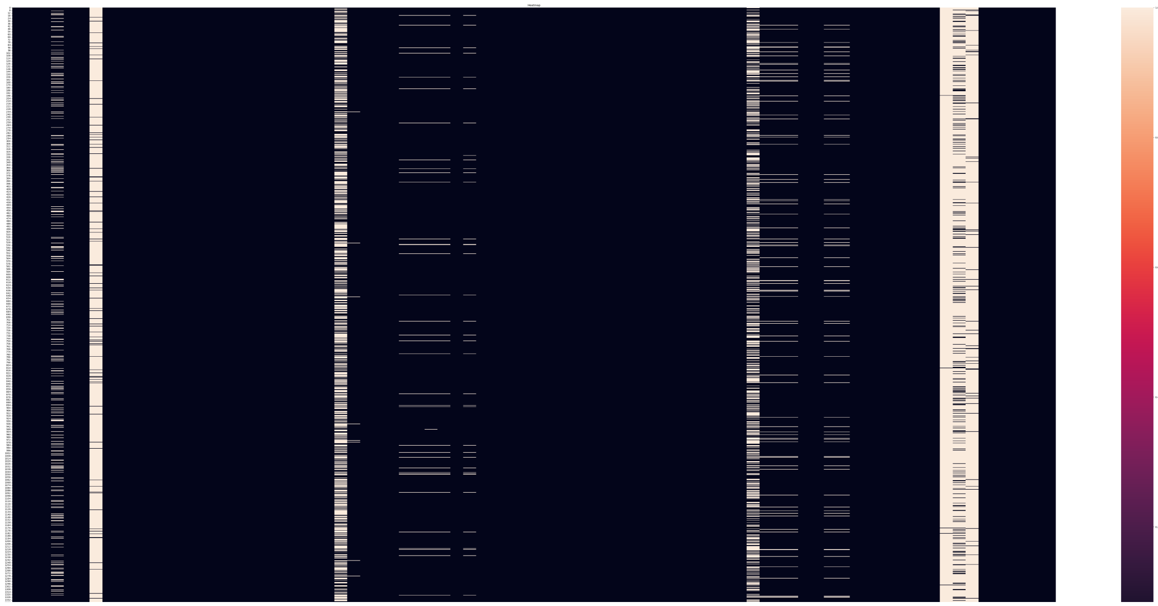
Out[9]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Year
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.00
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.26
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.20
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.00
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.00
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.00
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.00
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.00

Typesetting math: 0%

In [10]:
```
#heatmao of the dataset
plt.figure(figsize=(100,50))
sns.heatmap(data.isnull())
plt.title('Heatmap')
```

Out[10]: Text(0.5, 1.0, 'Heatmap')



Imputing missing values

In [11]:
```
#taking the percentage of missing values
nullvalues_percentage=data.isnull().sum()/data.shape[0]*100
nullvalues_percentage
```

Out[11]:
```
Id              0.000000
MSSubClass      0.000000
MSZoning        0.000000
LotFrontage    17.739726
LotArea         0.000000
Street          0.000000
Alley          93.767123
LotShape        0.000000
LandContour     0.000000
Utilities       0.000000
LotConfig       0.000000
LandSlope       0.000000
Neighborhood    0.000000
Condition1      0.000000
Condition2      0.000000
BldgType        0.000000
HouseStyle      0.000000
OverallQual     0.000000
OverallCond     0.000000
```

Typesetting math: 0%

In [12]:
```
Missing_value_feature=nullvalues_percentage[nullvalues_percentage>0]
Missing_value_feature
```

Out[12]:
```
LotFrontage     17.739726
Alley           93.767123
MasVnrType      59.726027
MasVnrArea       0.547945
BsmtQual         2.534247
BsmtCond         2.534247
BsmtExposure     2.602740
BsmtFinType1     2.534247
BsmtFinType2     2.602740
Electrical       0.068493
FireplaceQu     47.260274
GarageType       5.547945
GarageYrBlt      5.547945
GarageFinish     5.547945
GarageQual       5.547945
GarageCond       5.547945
PoolQC          99.520548
Fence           80.753425
MiscFeature     96.301370
dtype: float64
```

Typesetting math: 0%

In [13]:
```
data.info()
```

Typesetting math: 0%

```
<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     588 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
```

Typesetting math: 0%

In [14]:
```
def plot_boxplot_and_histogram(data, variable):

    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    # Before Imputation
    sns.boxplot(data=data, x=variable, ax=axes[0])
    axes[0].set_title(f'Box Plot (Before Imputation) for {variable}')
    axes[0].set_xlabel(variable)
    axes[0].set_ylabel('Value')
    sns.histplot(data=data, x=variable, kde=True, ax=axes[1])
    axes[1].set_title(f'Histogram (Before Imputation) for {variable}')
    axes[1].set_xlabel(variable)
    axes[1].set_ylabel('Count')
    plt.tight_layout()
    plt.show()

    # Impute missing values with mean
    data[variable].fillna(data[variable].mean(), inplace=True)
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    # After Imputation
    sns.boxplot(data=data, x=variable, ax=axes[0])
    axes[0].set_title(f'Box Plot (After Imputation) for {variable}')
    axes[0].set_xlabel(variable)
    axes[0].set_ylabel('Value')
    sns.histplot(data=data, x=variable, kde=True, ax=axes[1])
    axes[1].set_title(f'Histogram (After Imputation) for {variable}')
    axes[1].set_xlabel(variable)
    axes[1].set_ylabel('Count')
    plt.tight_layout()
    plt.show()
```

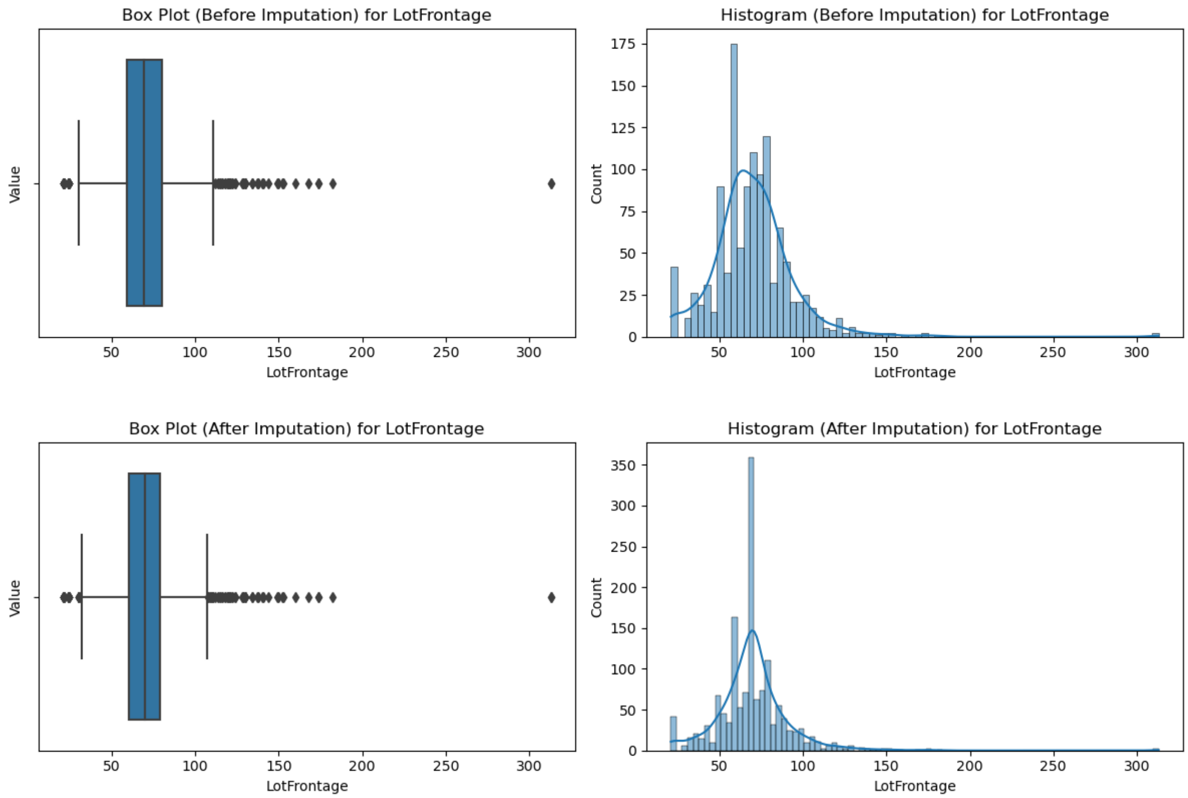
In [15]:
```
def plot_count_plot(data, variable):
    fig, axes = plt.subplots(1, 2, figsize=(15, 5))
    # Plot before imputation
    sns.countplot(data=data, x=variable, ax=axes[0])
    axes[0].set_title('Before Imputation')
    axes[0].set_xlabel(variable)
    axes[0].set_ylabel('Count')
    # Impute missing values as 'NA'
    data[variable].fillna('NAA', inplace=True)
    # Plot after imputation
    sns.countplot(data=data, x=variable, ax=axes[1])
    axes[1].set_title('After Imputation')
    axes[1].set_xlabel(variable)
    axes[1].set_ylabel('Count')

    plt.show()
```

LotFrontage = 17.739726

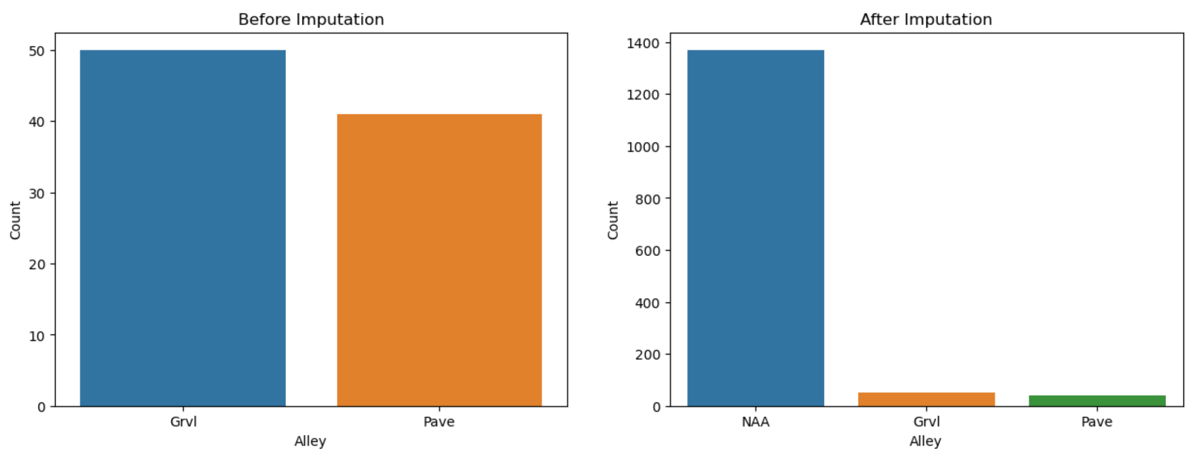
Typesetting math: 0%

In [16]:
```
plot_boxplot_and_histogram(data, 'LotFrontage')
```



Alley = 93.767123

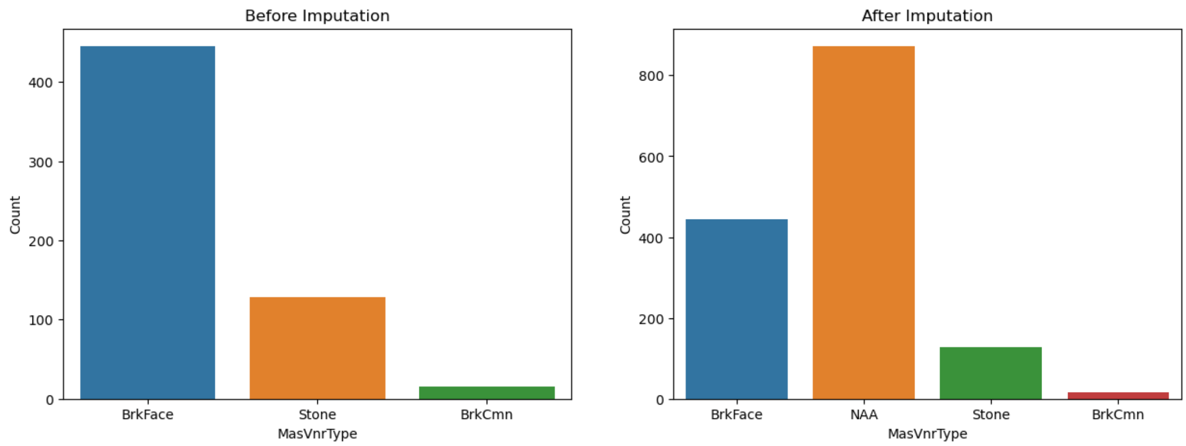
In [17]:
```
plot_count_plot(data, 'Alley')
```



MasVnrType = 0.547945

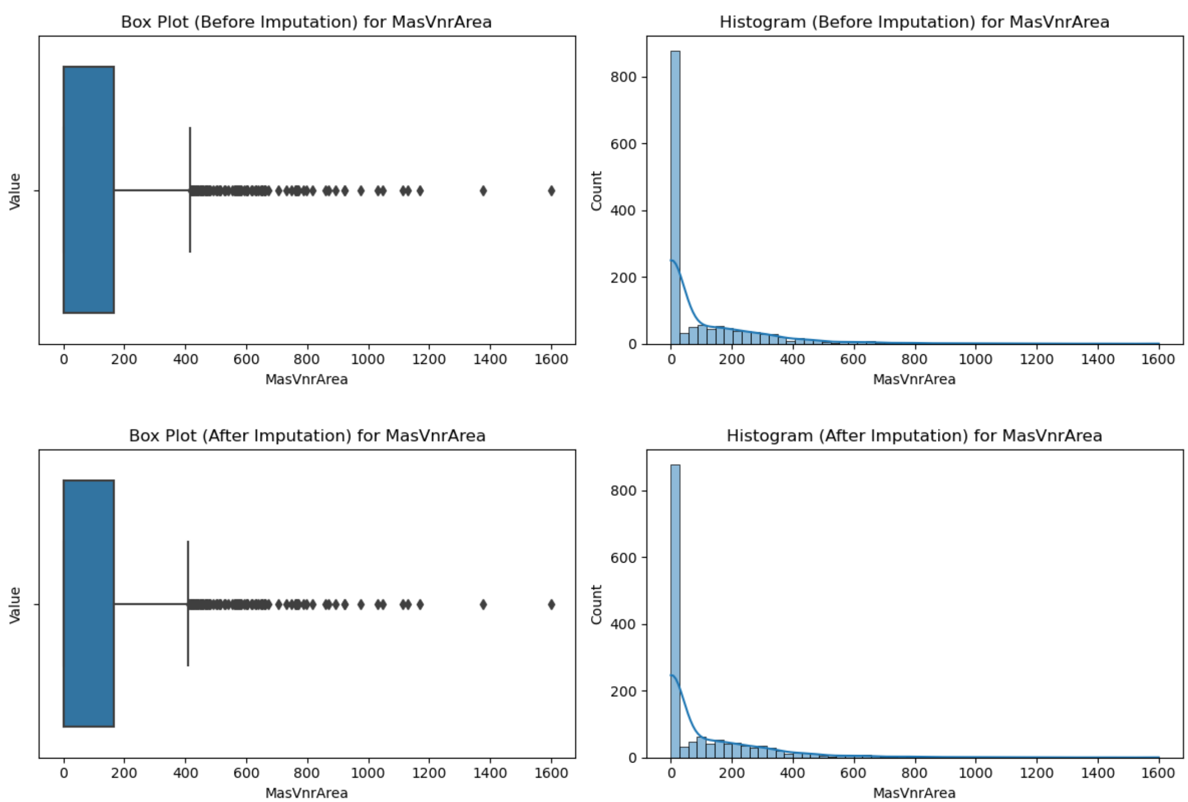
Typesetting math: 0%

In [18]:
```
plot_count_plot(data, 'MasVnrType')
```



MasVnrArea = 0.547945

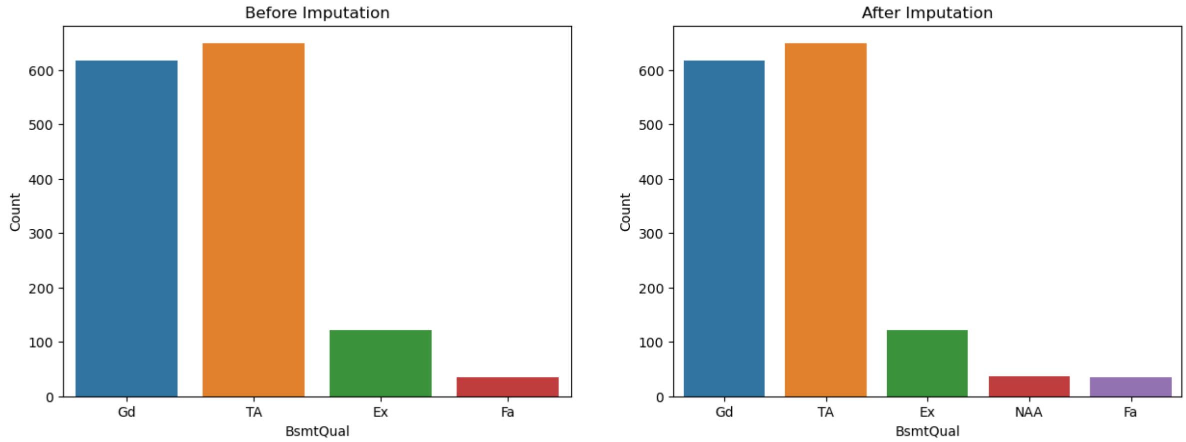
In [19]:
```
plot_boxplot_and_histogram(data, 'MasVnrArea')
```



BsmtQual = 2.534247

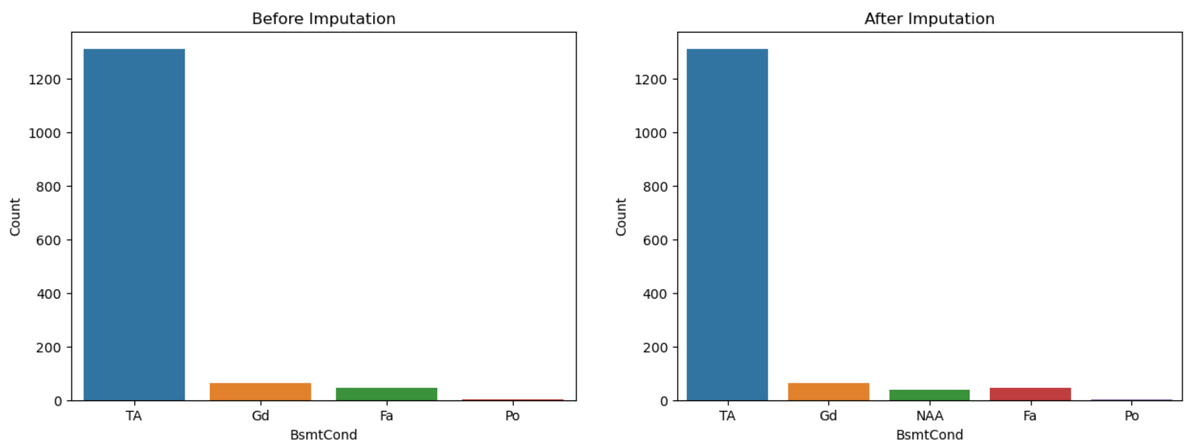
Typesetting math: 0%

In [20]:
```
plot_count_plot(data, 'BsmtQual')
```



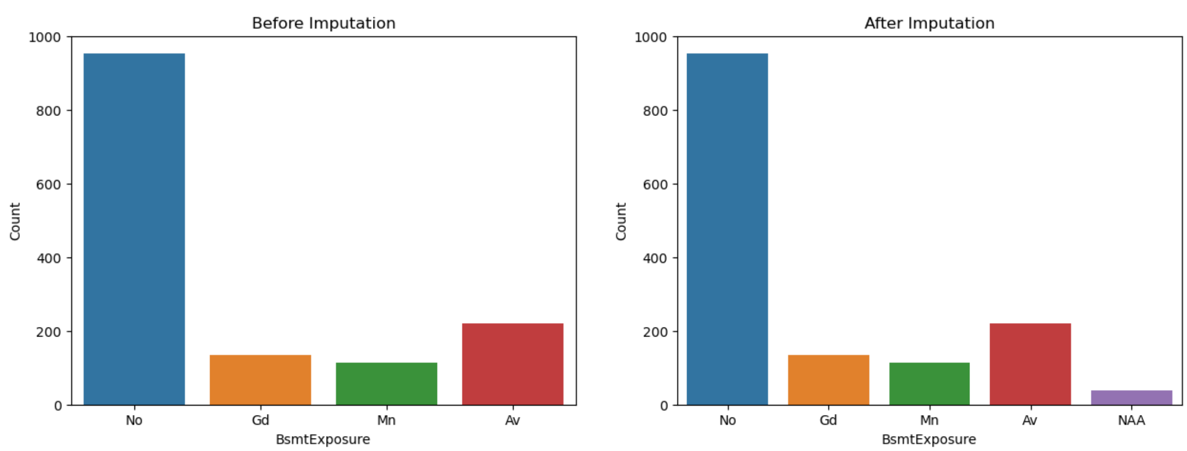
BsmtCond = 2.534247

In [21]:
```
plot_count_plot(data, 'BsmtCond')
```



BsmtExposure = 2.602740

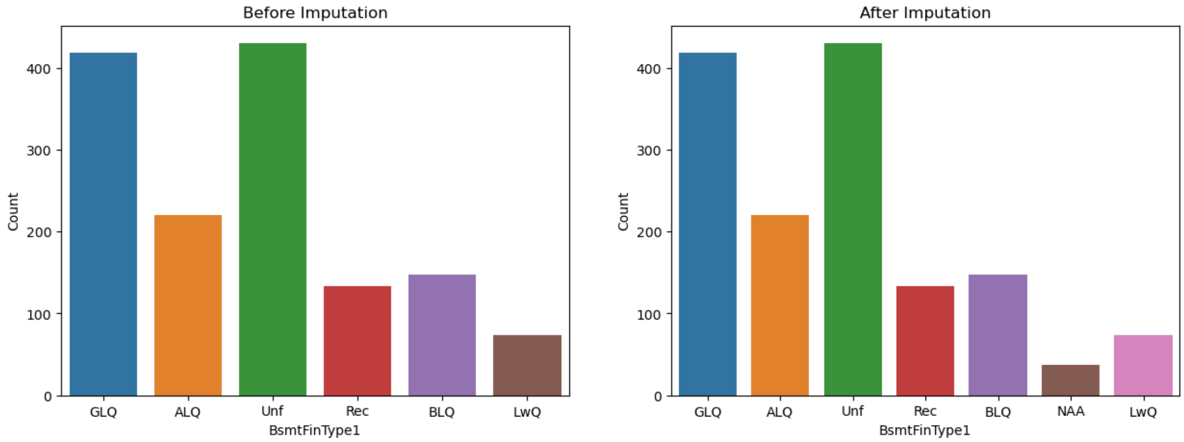
In [22]:
```
plot_count_plot(data, 'BsmtExposure')
```



Typesetting math: 0%

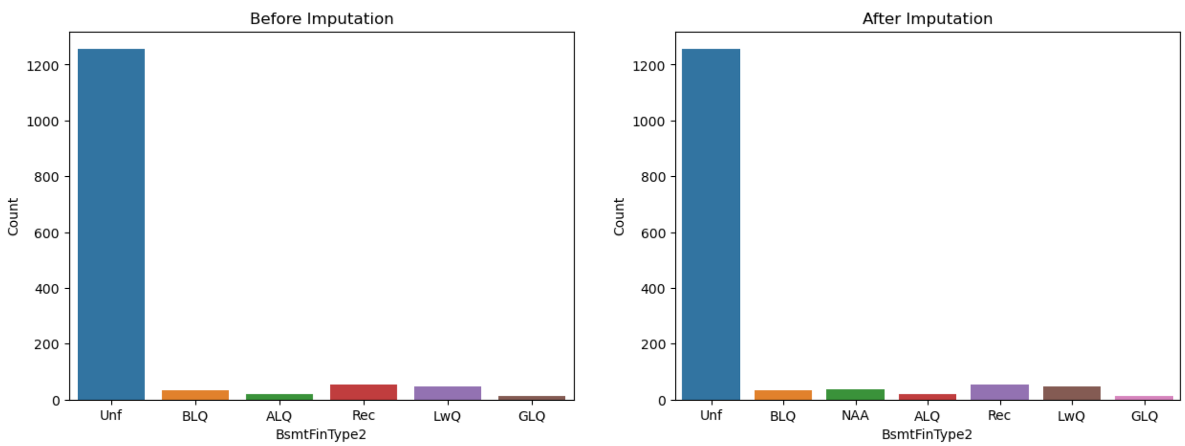
BsmtFinType1 2.534247

In [23]:
```
plot_count_plot(data, 'BsmtFinType1')
```



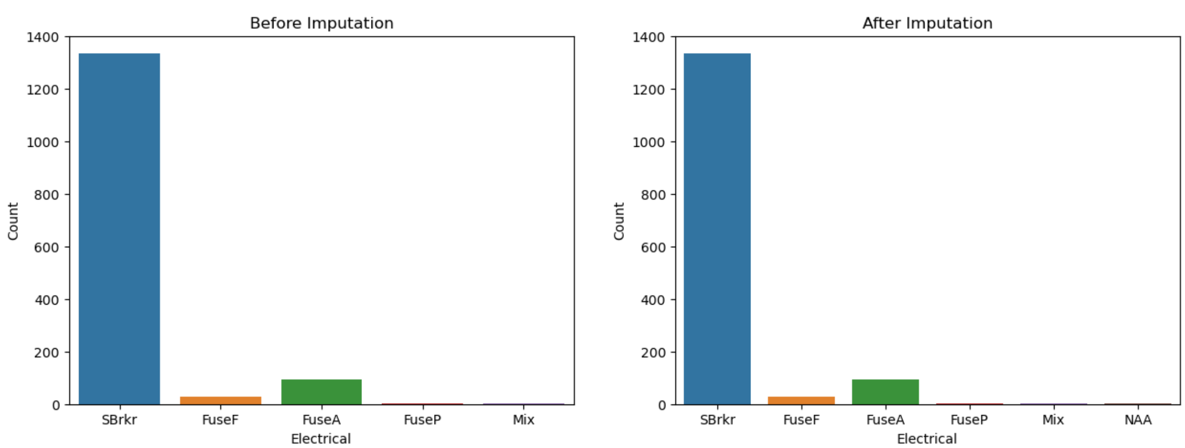
BsmtFinType2 2.602740

In [24]:
```
plot_count_plot(data, 'BsmtFinType2')
```



Electrical 0.068493

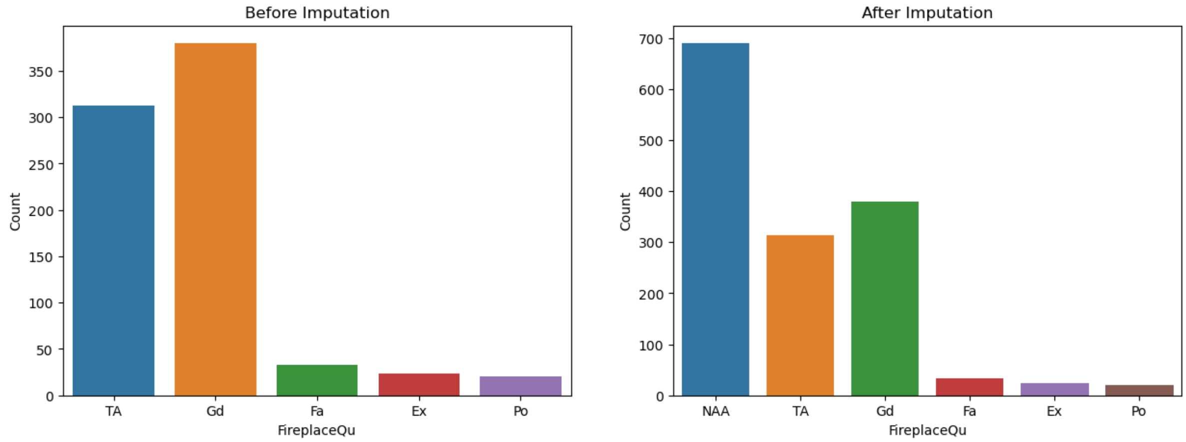
In [25]:
```
plot_count_plot(data, 'Electrical')
```



Typesetting math: 0%

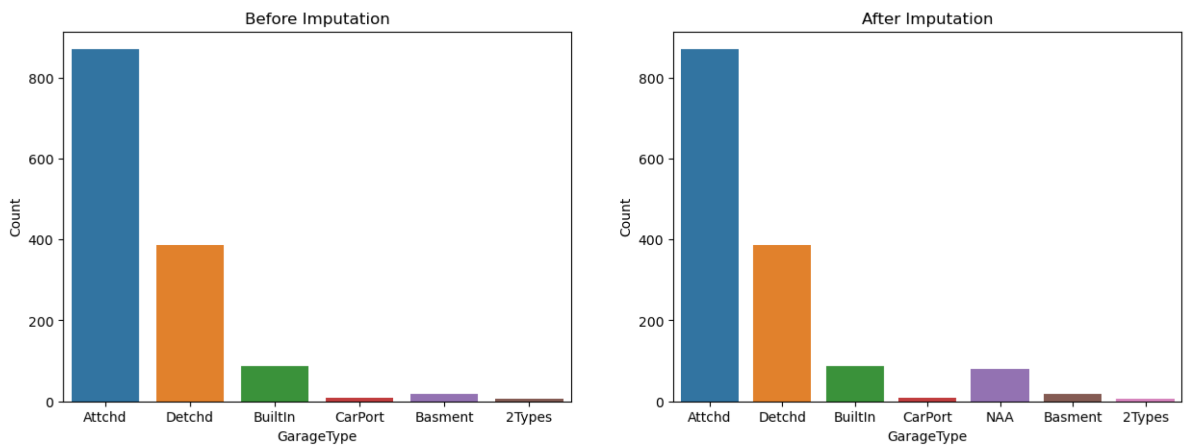
FireplaceQu 47.260274

In [26]:
```
plot_count_plot(data, 'FireplaceQu')
```



GarageType 5.547945

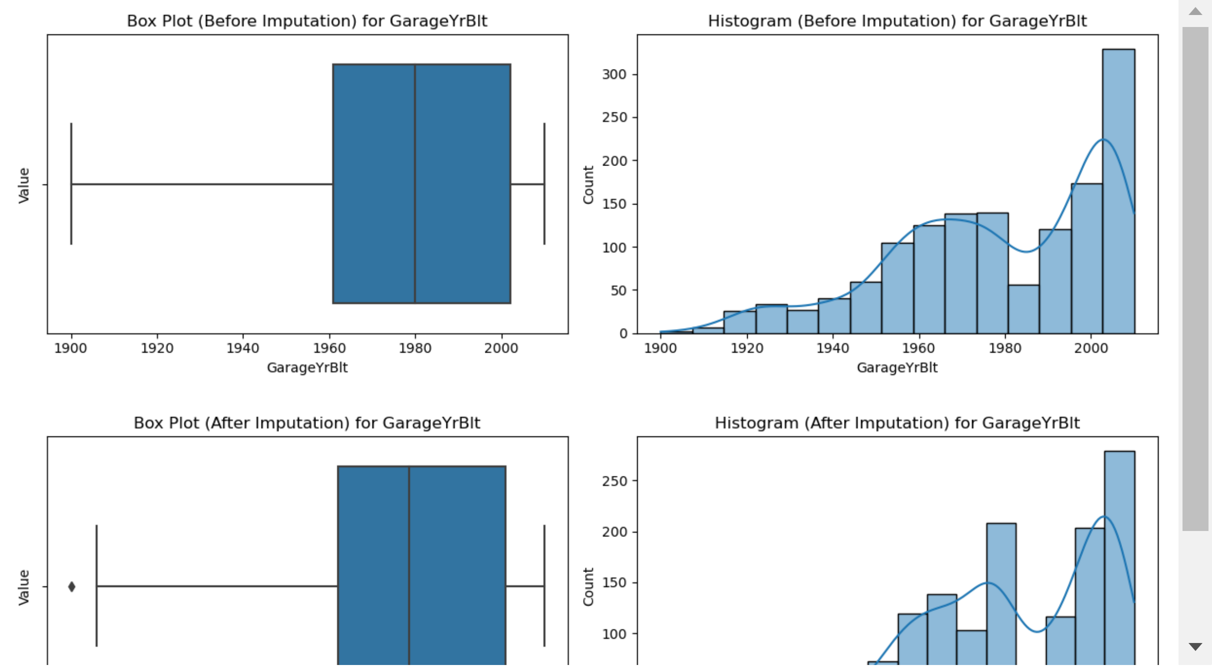
In [27]:
```
plot_count_plot(data, 'GarageType')
```



GarageYrBlt 5.547945

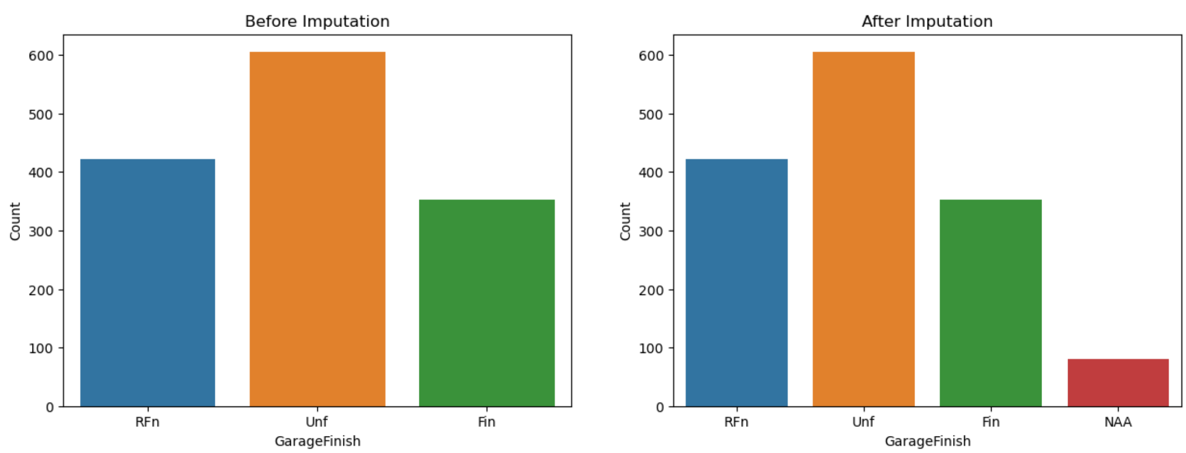
Typesetting math: 0%

In [28]:
```
plot_boxplot_and_histogram(data, 'GarageYrBlt')
```



GarageFinish 5.547945

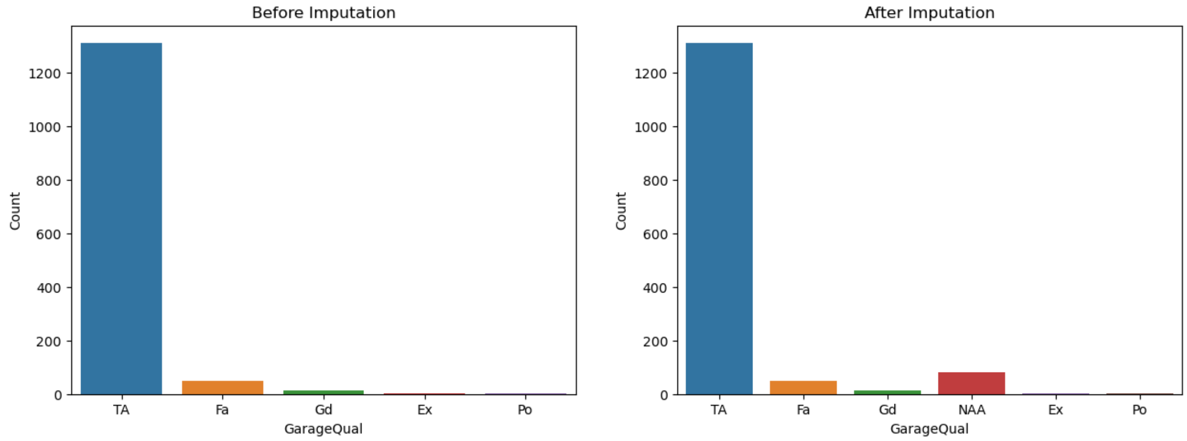
In [29]:
```
plot_count_plot(data, 'GarageFinish')
```



GarageQual 5.547945

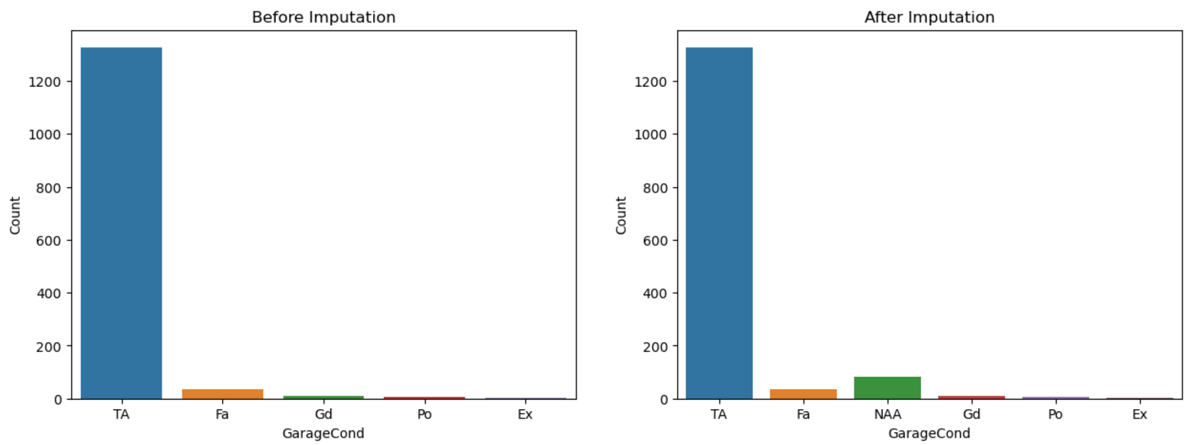
Typesetting math: 0%

In [30]:
```
plot_count_plot(data, 'GarageQual')
```



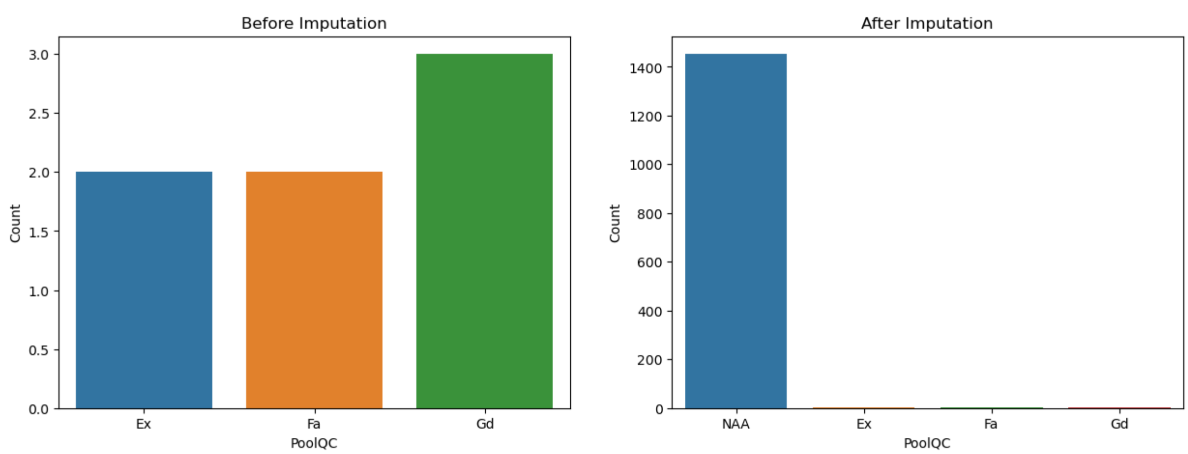
GarageCond 5.547945

In [31]:
```
plot_count_plot(data, 'GarageCond')
```



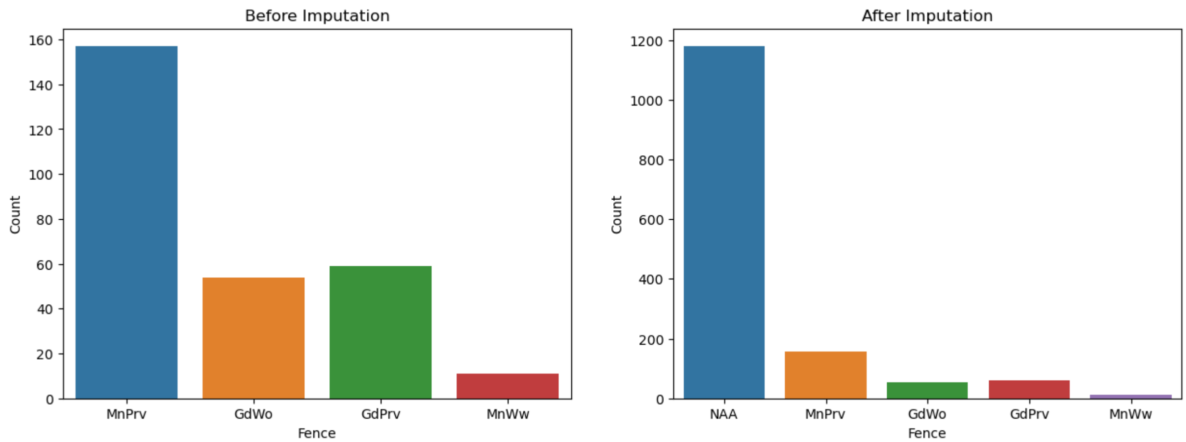
PoolQC 99.520548

In [32]:
```
plot_count_plot(data, 'PoolQC')
```



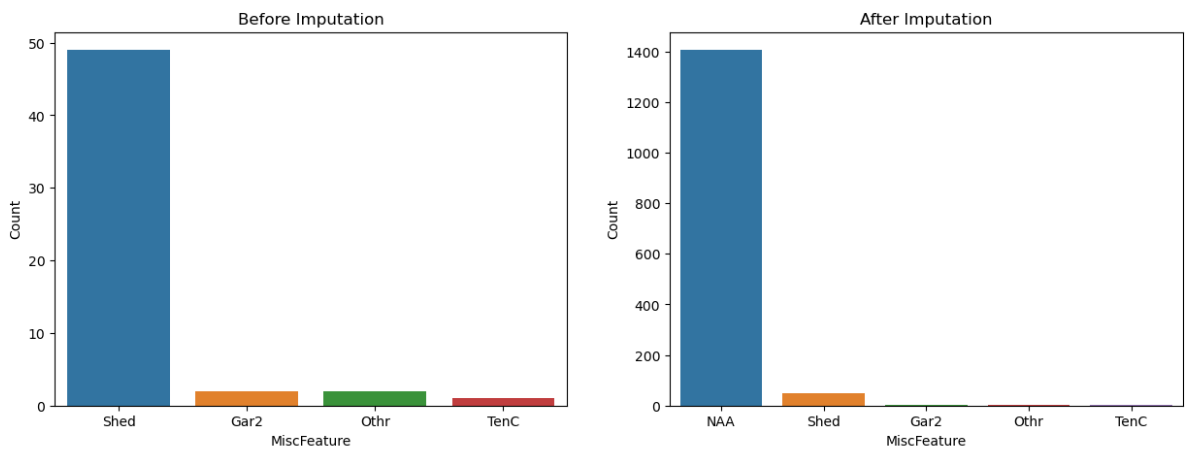
Fence 80.753425

In [33]:
```
plot_count_plot(data, 'Fence')
```



MiscFeature 96.301370

In [34]:
```
plot_count_plot(data, 'MiscFeature')
```



As per data set the value of variable which is not given in a colum is concedered as "NAA" according to domain this this where not present in the house but we can't neglect that its simply not present there.

Typesetting math: 0%

In [35]:
```
data.isnull().sum().head()
```

Out[35]:
```
Id             0
MSSubClass     0
MSZoning       0
LotFrontage    0
LotArea        0
dtype: int64
```

In [36]:
```
data.head()
```

Out[36]:

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

Conveting Numerical feature to categorical feature

In [37]:
```
con_categorical=['MoSold','YrSold','GarageYrBlt','YearRemodAdd','MSSubClass']
```

In [38]:
```
data[con_categorical].dtypes
```

Out[38]:
```
MoSold            int64
YrSold            int64
GarageYrBlt     float64
YearRemodAdd      int64
MSSubClass        int64
dtype: object
```

In [39]:
```
import calendar
```

In [40]:
```
data['MoSold']=data['MoSold'].apply(lambda x:calendar.month_abbr[x])
```

Typesetting math: 0%

In [41]:
```
data['MoSold'].value_counts()
```

Out[41]:
```
MoSold
Jun    253
Jul    234
May    204
Apr    141
Aug    122
Mar    106
Oct     89
Nov     79
Sep     63
Dec     59
Jan     58
Feb     52
Name: count, dtype: int64
```

In [42]:
```
data[con_categorical]=data[con_categorical].astype('object')
```

In [43]:
```
data[con_categorical].dtypes
```

Out[43]:
```
MoSold          object
YrSold          object
GarageYrBlt     object
YearRemodAdd    object
MSSubClass      object
dtype: object
```

In [44]:
```
data['PavedDrive'].value_counts()
```

Out[44]:
```
PavedDrive
Y    1340
N      90
P      30
Name: count, dtype: int64
```

Converting categorical feature to numerical

Typesetting math: 0%

In [45]:
```
data['ExterQual']=data['ExterQual'].map({'Po':0,'Fa':1,'TA':2,'Gd':3,'Ex':4})
data['ExterCond']=data['ExterCond'].map({'Po':0,'Fa':1,'TA':2,'Gd':3,'Ex':4})
data['BsmtQual']=data['BsmtQual'].map({'NAA':0,'Po':1,'Fa':2,'TA':3,'Gd':4,'Ex
data['BsmtCond']=data['BsmtCond'].map({'NAA':0,'Po':1,'Fa':2,'TA':3,'Gd':4,'Ex
data['BsmtExposure']=data['BsmtExposure'].map({'NAA':0,'No':1,'Mn':2,'Av':3,'G
data['HeatingQC']=data['HeatingQC'].map({'Po':0,'Fa':1,'TA':2,'Gd':3,'Ex':4})
data['KitchenQual']=data['KitchenQual'].map({'Po':0,'Fa':1,'TA':2,'Gd':3,'Ex':
data['FireplaceQu']=data['FireplaceQu'].map({'NAA':0,'Po':1,'Fa':2,'TA':3,'Gd'
data['GarageQual']=data['GarageQual'].map({'NAA':0,'Po':1,'Fa':2,'TA':3,'Gd':4
data['GarageCond']=data['GarageCond'].map({'NAA':0,'Po':1,'Fa':2,'TA':3,'Gd':4
data['PoolQC']=data['PoolQC'].map({'NAA':0,'Fa':1,'TA':2,'Gd':3,'Ex':4})
data['Functional']=data['Functional'].map({'Sal':0,'Sev':1,'Maj2':2,'Maj1':3,'
data['PavedDrive']=data['PavedDrive'].map({'N':0,'P':1,'Y':2})
data['GarageFinish']=data['GarageFinish'].map({'NAA':0,'Unf':1,'RFn':2,'Fin':3
data['Fence']=data['Fence'].map({'NAA':0,'MnWw':1,'GdWo':2,'MnPrv':3,'GdPrv':4
data['Utilities']=data['Utilities'].map({'ELO':0,'NoSeWa':1,'NoSewr':2,'AllPub
```

One hot encodinf for Categorical variable

In [46]:
```
data_object=data.select_dtypes(include=['object']).columns.tolist()
print(data_object)
```

```
['MSSubClass', 'MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Lot
Config', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
'HouseStyle', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exteri
or2nd', 'MasVnrType', 'Foundation', 'BsmtFinType1', 'BsmtFinType2', 'Heatin
g', 'CentralAir', 'Electrical', 'GarageType', 'GarageYrBlt', 'MiscFeature',
'MoSold', 'YrSold', 'SaleType', 'SaleCondition']
```

In [47]:
```
def display_value_counts(data):
    for column in data_object:
        print(f"Column: {column}")
        print(data[column].unique())
        print(data[column].nunique())
        print("-" * 80)
```

Typesetting math: 0%

In [48]:
```
data_object = data.select_dtypes(include=['object'])
display_value_counts(data_object)
```

Typesetting math: 0%

```
Column: MSSubClass
[60 20 70 50 190 45 90 120 30 85 80 160 75 180 40]
15
-----------------------------------------------------------------------------
---
Column: MSZoning
['RL' 'RM' 'C (all)' 'FV' 'RH']
5
-----------------------------------------------------------------------------
---
Column: Street
['Pave' 'Grvl']
2
-----------------------------------------------------------------------------
---
Column: Alley
['NAA' 'Grvl' 'Pave']
3
-----------------------------------------------------------------------------
---
Column: LotShape
['Reg' 'IR1' 'IR2' 'IR3']
4
-----------------------------------------------------------------------------
---
Column: LandContour
['Lvl' 'Bnk' 'Low' 'HLS']
4
-----------------------------------------------------------------------------
---
Column: LotConfig
['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
5
-----------------------------------------------------------------------------
---
Column: LandSlope
['Gtl' 'Mod' 'Sev']
3
-----------------------------------------------------------------------------
---
Column: Neighborhood
['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst' 'NWAmes'
 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
25
-----------------------------------------------------------------------------
---
Column: Condition1
['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
9
-----------------------------------------------------------------------------
---
Column: Condition2
['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RRAn' 'RRAe']
8
-----------------------------------------------------------------------------
```

Typesetting math: 0%

```
---
Column: BldgType
['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
5
-----------------------------------------------------------------------------
---
Column: HouseStyle
['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf' '2.5Fin']
8
-----------------------------------------------------------------------------
---
Column: YearRemodAdd
[2003 1976 2002 1970 2000 1995 2005 1973 1950 1965 2006 1962 2007 1960
 2001 1967 2004 2008 1997 1959 1990 1955 1983 1980 1966 1963 1987 1964
 1972 1996 1998 1989 1953 1956 1968 1981 1992 2009 1982 1961 1993 1999
 1985 1979 1977 1969 1958 1991 1971 1952 1975 2010 1984 1986 1994 1988
 1954 1957 1951 1978 1974]
61
-----------------------------------------------------------------------------
---
Column: RoofStyle
['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
6
-----------------------------------------------------------------------------
---
Column: RoofMatl
['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll'
 'ClyTile']
8
-----------------------------------------------------------------------------
---
Column: Exterior1st
['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFace' 'WdShing' 'CemntBd'
 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
 'CBlock']
15
-----------------------------------------------------------------------------
---
Column: Exterior2nd
['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywood' 'Wd Sdng' 'CmentBd'
 'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone'
 'Other' 'CBlock']
16
-----------------------------------------------------------------------------
---
Column: MasVnrType
['BrkFace' 'NAA' 'Stone' 'BrkCmn']
4
-----------------------------------------------------------------------------
---
Column: Foundation
['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
6
-----------------------------------------------------------------------------
---
Column: BsmtFinType1
['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' 'NAA' 'LwQ']
```

Typesetting math: 0%

```
7
--------------------------------------------------------------------------------
---
Column: BsmtFinType2
['Unf' 'BLQ' 'NAA' 'ALQ' 'Rec' 'LwQ' 'GLQ']
7
--------------------------------------------------------------------------------
---
Column: Heating
['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
6
--------------------------------------------------------------------------------
---
Column: CentralAir
['Y' 'N']
2
--------------------------------------------------------------------------------
---
Column: Electrical
['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' 'NAA']
6
--------------------------------------------------------------------------------
---
Column: GarageType
['Attchd' 'Detchd' 'BuiltIn' 'CarPort' 'NAA' 'Basment' '2Types']
7
--------------------------------------------------------------------------------
---
Column: GarageYrBlt
[2003.0 1976.0 2001.0 1998.0 2000.0 1993.0 2004.0 1973.0 1931.0 1939.0
 1965.0 2005.0 1962.0 2006.0 1960.0 1991.0 1970.0 1967.0 1958.0 1930.0
 2002.0 1968.0 2007.0 2008.0 1957.0 1920.0 1966.0 1959.0 1995.0 1954.0
 1953.0 1978.5061638868744 1983.0 1977.0 1997.0 1985.0 1963.0 1981.0
 1964.0 1999.0 1935.0 1990.0 1945.0 1987.0 1989.0 1915.0 1956.0 1948.0
 1974.0 2009.0 1950.0 1961.0 1921.0 1900.0 1979.0 1951.0 1969.0 1936.0
 1975.0 1971.0 1923.0 1984.0 1926.0 1955.0 1986.0 1988.0 1916.0 1932.0
 1972.0 1918.0 1980.0 1924.0 1996.0 1940.0 1949.0 1994.0 1910.0 1978.0
 1982.0 1992.0 1925.0 1941.0 2010.0 1927.0 1947.0 1937.0 1942.0 1938.0
 1952.0 1928.0 1922.0 1934.0 1906.0 1914.0 1946.0 1908.0 1929.0 1933.0]
98
--------------------------------------------------------------------------------
---
Column: MiscFeature
['NAA' 'Shed' 'Gar2' 'Othr' 'TenC']
5
--------------------------------------------------------------------------------
---
Column: MoSold
['Feb' 'May' 'Sep' 'Dec' 'Oct' 'Aug' 'Nov' 'Apr' 'Jan' 'Jul' 'Mar' 'Jun']
12
--------------------------------------------------------------------------------
---
Column: YrSold
[2008 2007 2006 2009 2010]
5
--------------------------------------------------------------------------------
---
```

Typesetting math: 0%

```
Column: SaleType
['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']
9
----------------------------------------------------------------------------
---
Column: SaleCondition
['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']
6
----------------------------------------------------------------------------
---
```

In [49]:
```
#382
data=data.drop(['Id'],axis=1)
```

In [50]:
```
data.head()
```

Out[50]:

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

In [51]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [52]:
```
encoder= OneHotEncoder()
```

In [53]:
```
data_object = data.select_dtypes(include=['object'])
```

In [54]:
```
encoded_data = encoder.fit_transform(data_object)
```

In [55]:
```
encoded_data_dense = encoded_data.toarray()
```

In [56]:
```
category_names = encoder.get_feature_names_out(input_features=data_object.colu
```

In [57]:
```
encoded_df = pd.DataFrame(encoded_data_dense, columns=category_names)
```

In [58]:
```
data = data.drop(columns=data_object.columns)
data = pd.concat([data, encoded_df], axis=1)
```

In [59]:
```
data.shape
```

Out[59]:
```
(1460, 430)
```

Outlier Treatment

Typesetting math: 0%

In [60]:
```
def outliers(data, variable):
    # Before outlier treatment: Display the box plot
    plt.figure(figsize=(8, 3))
    plt.subplot(1, 2, 1)
    sns.boxplot(data=data, y=variable)
    plt.title(f'Before Outlier Treatment - {variable}')

    # Calculate the Interquartile Range (IQR)
    Q1 = data[variable].quantile(0.25)
    Q3 = data[variable].quantile(0.75)
    IQR = Q3 - Q1

    # Define the upper and lower bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Treat outliers by replacing them with the upper or lower bound
    data[variable] = np.where(data[variable] < lower_bound, lower_bound, data[
    data[variable] = np.where(data[variable] > upper_bound, upper_bound, data[

    # After outlier treatment: Display the box plot
    plt.subplot(1, 2, 2)
    sns.boxplot(data=data, y=variable)
    plt.title(f'After Outlier Treatment - {variable}')

    plt.tight_layout()
    plt.show()
```

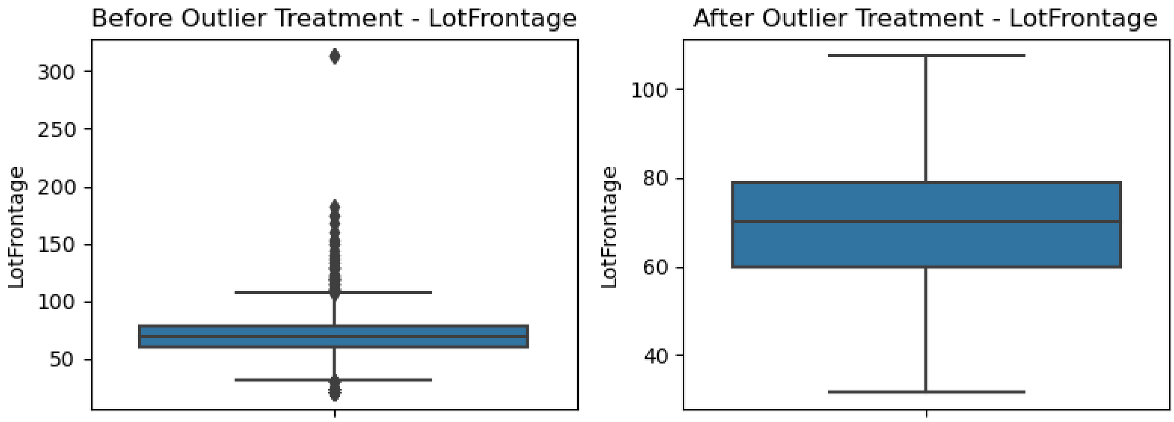
In [61]:
```
data.describe()
```

Out[61]:

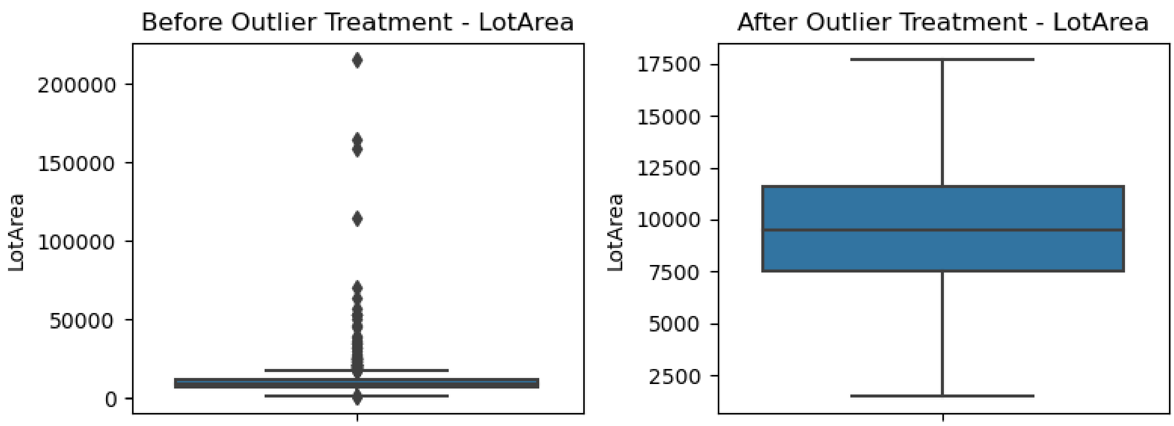
	LotFrontage	LotArea	Utilities	OverallQual	OverallCond	YearBuilt	MasVnr
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.00
mean	70.049958	10516.828082	2.998630	6.099315	5.575342	1971.267808	103.68
std	22.024023	9981.264932	0.052342	1.382997	1.112799	30.202904	180.56
min	21.000000	1300.000000	1.000000	1.000000	1.000000	1872.000000	0.00
25%	60.000000	7553.500000	3.000000	5.000000	5.000000	1954.000000	0.00
50%	70.049958	9478.500000	3.000000	6.000000	5.000000	1973.000000	0.00
75%	79.000000	11601.500000	3.000000	7.000000	6.000000	2000.000000	164.25
max	313.000000	215245.000000	3.000000	10.000000	9.000000	2010.000000	1600.00

Typesetting math: 0%

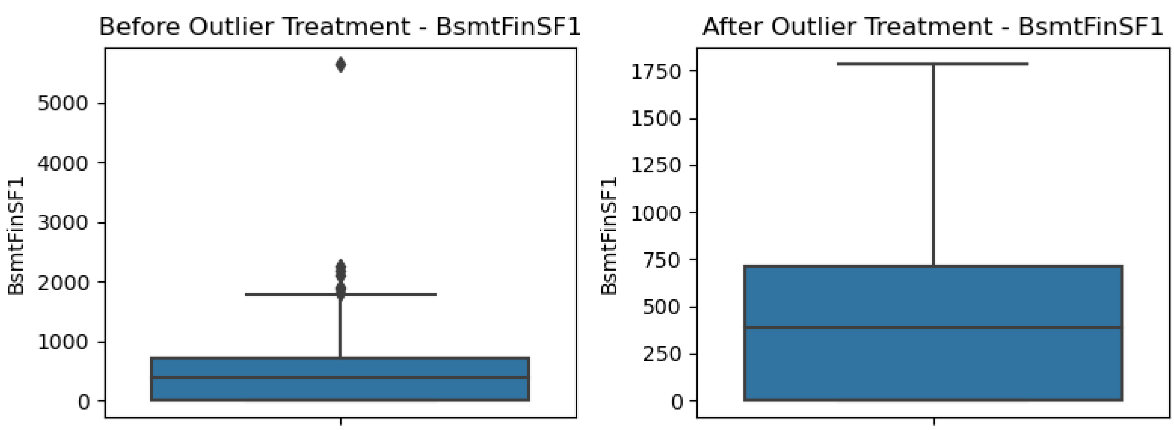
In [62]:
```
outliers(data, 'LotFrontage')
```



In [63]:
```
outliers(data, 'LotArea')
```

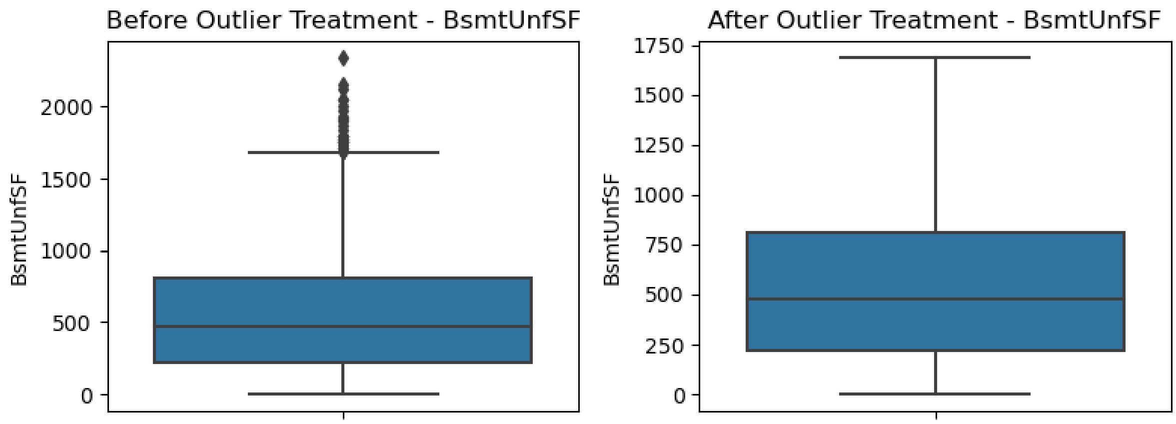


In [64]:
```
outliers(data, 'BsmtFinSF1')
```

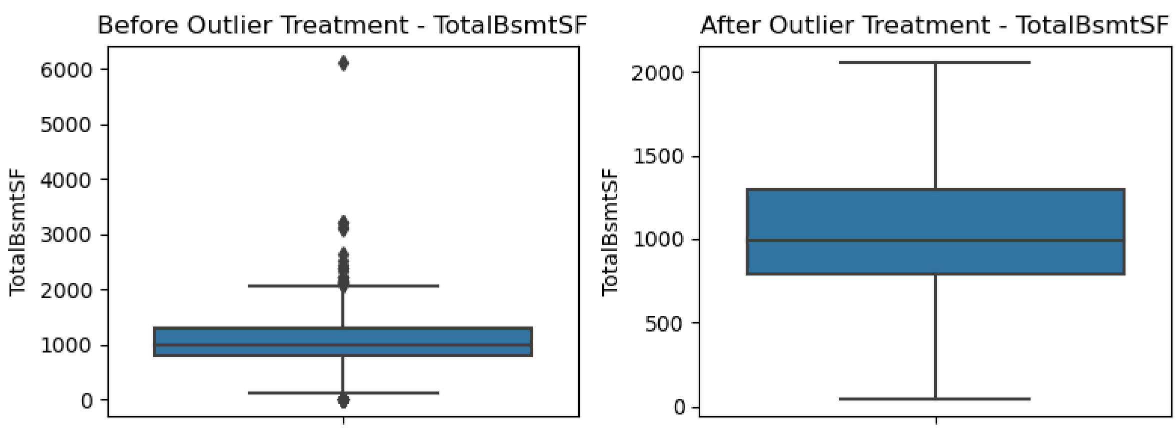


Typesetting math: 0%

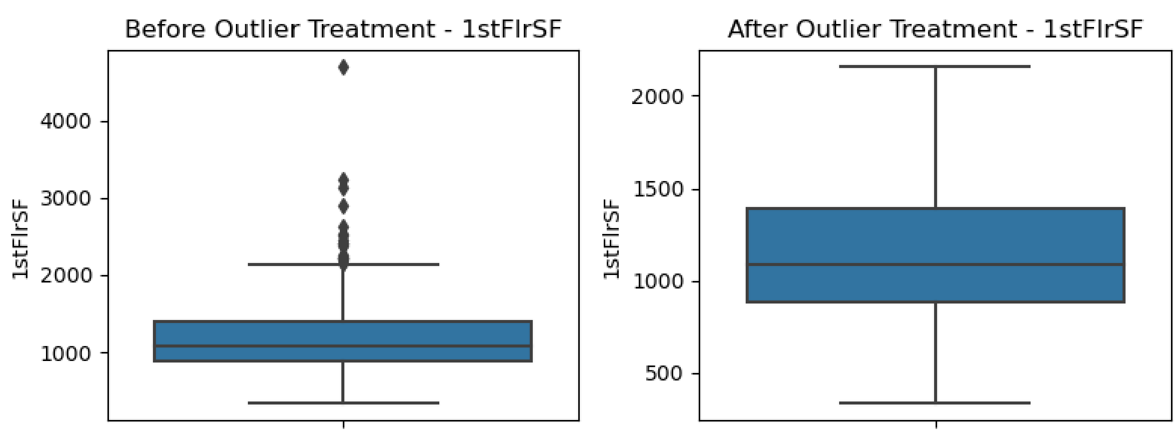
In [65]:
```
outliers(data, 'BsmtUnfSF')
```



In [66]:
```
outliers(data, 'TotalBsmtSF')
```

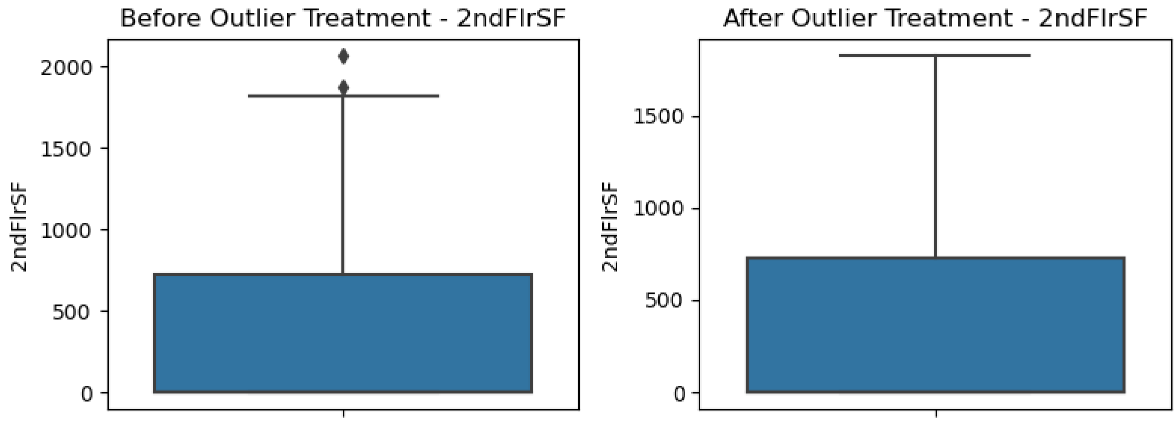


In [67]:
```
outliers(data, '1stFlrSF')
```

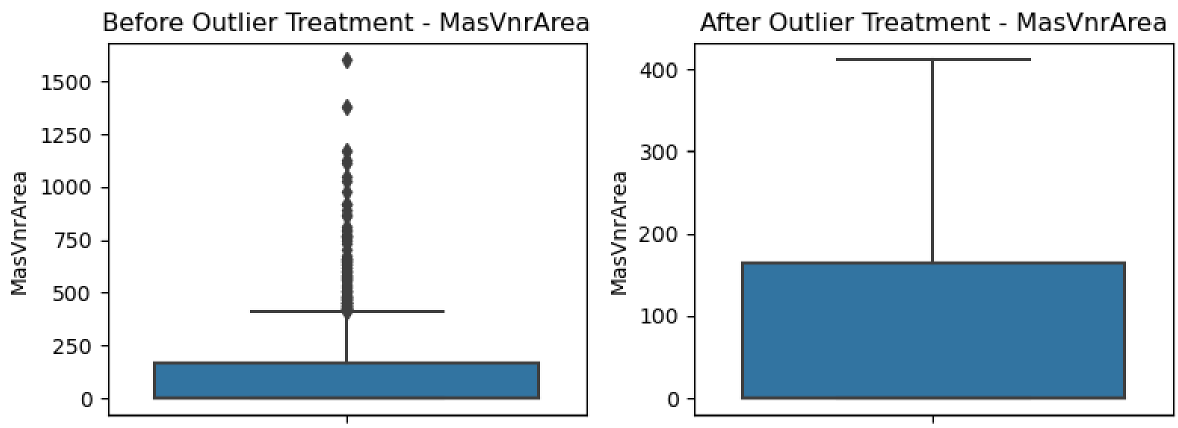


Typesetting math: 0%

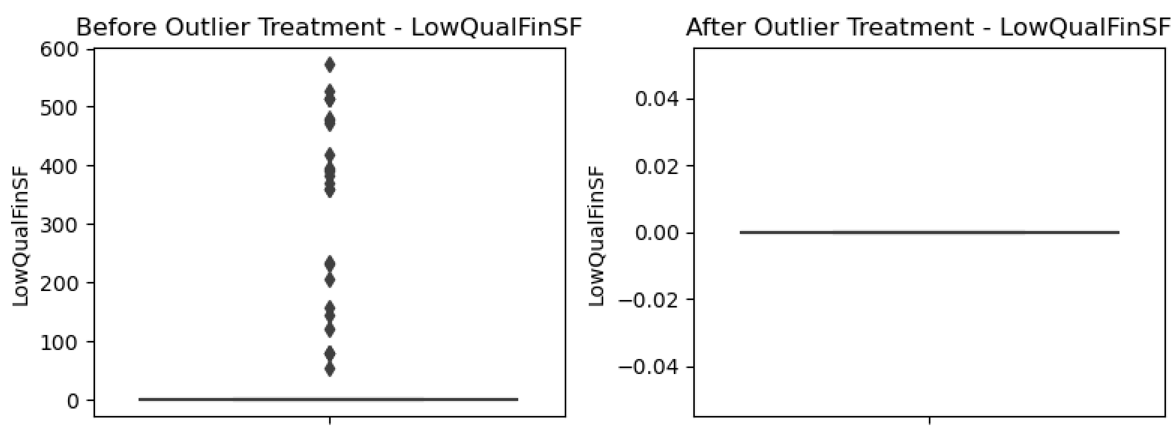
In [68]:
```
outliers(data, '2ndFlrSF')
```



In [69]:
```
outliers(data, 'MasVnrArea')
```

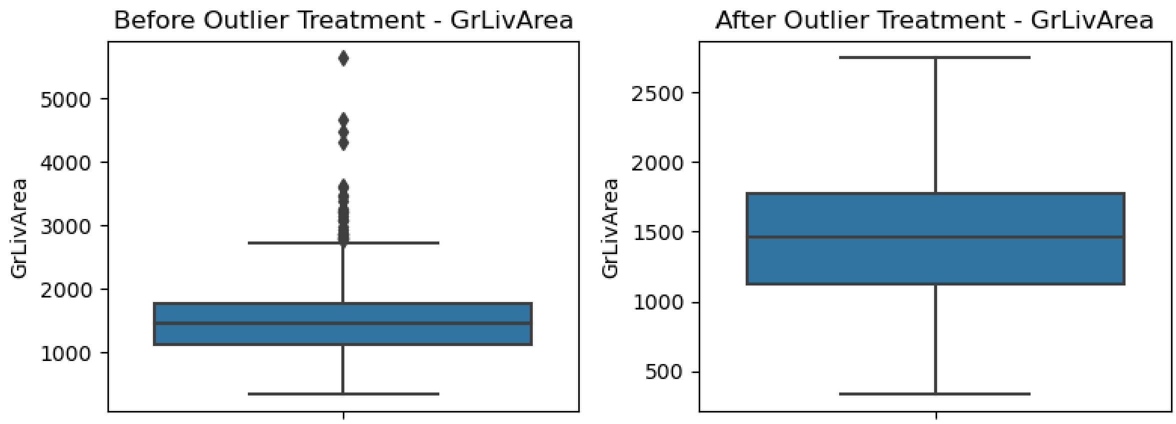


In [70]:
```
outliers(data, 'LowQualFinSF')
```

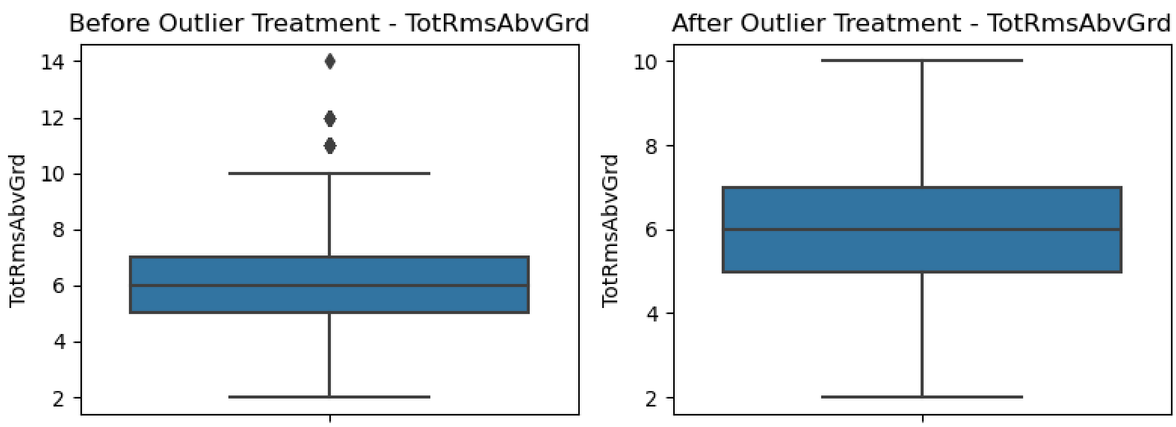


Typesetting math: 0%

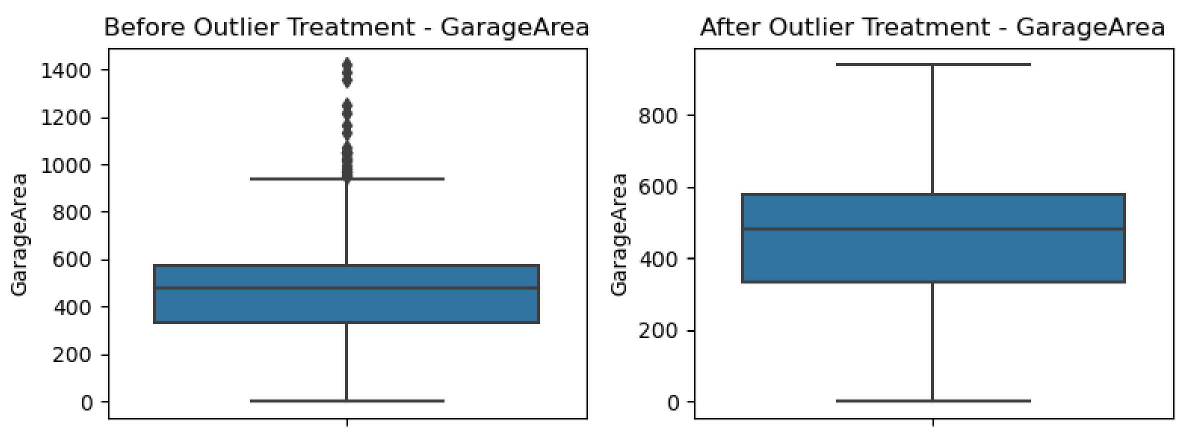
In [71]:
```
outliers(data, 'GrLivArea')
```



In [72]:
```
outliers(data, 'TotRmsAbvGrd')
```



In [73]:
```
outliers(data, 'GarageArea')
```



Typesetting math: 0%

In [74]:
```
data.describe()
```

Out[74]:

	LotFrontage	LotArea	Utilities	OverallQual	OverallCond	YearBuilt	MasVnrA
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000
mean	69.276671	9647.388014	2.998630	6.099315	5.575342	1971.267808	89.974
std	17.235602	3594.356399	0.052342	1.382997	1.112799	30.202904	133.856
min	31.500000	1481.500000	1.000000	1.000000	1.000000	1872.000000	0.000
25%	60.000000	7553.500000	3.000000	5.000000	5.000000	1954.000000	0.000
50%	70.049958	9478.500000	3.000000	6.000000	5.000000	1973.000000	0.000
75%	79.000000	11601.500000	3.000000	7.000000	6.000000	2000.000000	164.250
max	107.500000	17673.500000	3.000000	10.000000	9.000000	2010.000000	410.625

Spliting the Dataset

In [75]:
```
x=data.drop(["SalePrice"],axis=1)
y=data["SalePrice"]
```

In [76]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

In [77]:
```
print('x_train =',x_train.shape)
print('y_train =',y_train.shape)
print('x_test =',x_test.shape)
print('y_test =',y_test.shape)
```

```
x_train = (1168, 429)
y_train = (1168,)
x_test = (292, 429)
y_test = (292,)
```

In [78]:
```
sc=StandardScaler()
sc.fit(x_train)

x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

Train ML Model

Typesetting math: 0%

In [79]:
```
#Importing Required libraries to train the model

from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor

from xgboost import XGBRegressor

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

In [80]:
```
lr=LinearRegression()
knr=KNeighborsRegressor()
gpr=GaussianProcessRegressor()
dtr=DecisionTreeRegressor()
gbr=GradientBoostingRegressor()
rfr=RandomForestRegressor()
xgbr=XGBRegressor()
```

In [81]:
```
# Train the model for LinearRegression
lr.fit(x_train, y_train)

# Make predictions on the test data
y_pred = lr.predict(x_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse}")
print(f"R-squared (R^2) Score: {r2}")
```

```
Mean Squared Error: 8.999838009788706e+31
R-squared (R^2) Score: -1.470367828550555e+22
```

Typesetting math: 0%

In [82]:
```
# Model Training for KNeighborsRegressor
knr.fit(x_train, y_train)

# Model Prediction
y_pred = knr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: 0.6246090369128263
Model Accuracy: 0.6246090369128263
```

In [83]:
```
# Model Training GaussianProcessRegressor
gpr.fit(x_train, y_train)

# Model Prediction
y_pred = gpr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: -5.439196305991505
Model Accuracy: -5.439196305991505
```

Typesetting math: 0%

In [84]:
```
# Model Training DecisionTreeRegressor
dtr.fit(x_train, y_train)

# Model Prediction
y_pred = dtr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: 0.7130434198310284
Model Accuracy: 0.7130434198310284
```

In [85]:
```
# Model Training GradientBoostingRegressor
gbr.fit(x_train, y_train)

# Model Prediction
y_pred = gbr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: 0.8464026365851895
Model Accuracy: 0.8464026365851895
```

Typesetting math: 0%

In [86]:
```
# Model Training RandomForestRegressor
rfr.fit(x_train, y_train)

# Model Prediction
y_pred = rfr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: 0.8478529135260945
Model Accuracy: 0.8478529135260945
```

In [87]:
```
# Model Training XGBRegressor
xgbr.fit(x_train, y_train)

# Model Prediction
y_pred = xgbr.predict(x_test)

# Evaluation
r2 = r2_score(y_test, y_pred)

# Calculate Accuracy (R-squared score)
accuracy = r2

# Print Results
print(f"R-squared (R2) Score: {r2}")
print(f"Model Accuracy: {accuracy}")
```

```
R-squared (R2) Score: 0.7556655915287752
Model Accuracy: 0.7556655915287752
```

Typesetting math: 0%

In [88]:
```
def evaluate_models(x_train, y_train, x_test, y_test):
    models = {
        'Linear Regression': LinearRegression(),
        'K-Nearest Neighbors': KNeighborsRegressor(),
        'Gaussian Process': GaussianProcessRegressor(),
        'Decision Tree': DecisionTreeRegressor(),
        'Gradient Boosting': GradientBoostingRegressor(),
        'Random Forest': RandomForestRegressor(),
        'XGBoost': XGBRegressor()
    }

    results = {}

    for model_name, model in models.items():
        # Model Training
        model.fit(x_train, y_train)

        # Cross-validation
        cv_scores = cross_val_score(model, x_train, y_train, cv=5, scoring='r2
        cv_mean_score = np.mean(cv_scores)

        # Model Prediction
        y_pred = model.predict(x_test)

        # Evaluation
        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        results[model_name] = {
            'Cross-Validation Mean R2 Score': cv_mean_score,
            'Mean Squared Error (MSE)': mse,
            'R-squared (R2) Score': r2
        }

    return results

results = evaluate_models(x_train, y_train, x_test, y_test)

# results for each model
for model_name, metrics in results.items():
    print(f"Model: {model_name}")
    print(f"Cross-Validation Mean R2 Score: {metrics['Cross-Validation Mean R2
    print(f"Mean Squared Error (MSE): {metrics['Mean Squared Error (MSE)']}")
    print(f"R-squared (R2) Score: {metrics['R-squared (R2) Score']}")
    print("\n")
    print("-----------------------------------------------------------------------
```

Typesetting math: 0%

```
Model: Linear Regression
Cross-Validation Mean R2 Score: -1.7332794544318849e+25
Mean Squared Error (MSE): 8.999838009788706e+31
R-squared (R2) Score: -1.470367828550555e+22


-----------------------------------------------------------------------------
------------------------
Model: K-Nearest Neighbors
Cross-Validation Mean R2 Score: 0.6072733914829841
Mean Squared Error (MSE): 2297695714.312192
R-squared (R2) Score: 0.6246090369128263


-----------------------------------------------------------------------------
------------------------
Model: Gaussian Process
Cross-Validation Mean R2 Score: -5.294053823012925
Mean Squared Error (MSE): 39413079191.402374
R-squared (R2) Score: -5.439196305991505


-----------------------------------------------------------------------------
------------------------
Model: Decision Tree
Cross-Validation Mean R2 Score: 0.6659240710241738
Mean Squared Error (MSE): 2019951507.5753424
R-squared (R2) Score: 0.6699860921118173


-----------------------------------------------------------------------------
------------------------
Model: Gradient Boosting
Cross-Validation Mean R2 Score: 0.8718722063485167
Mean Squared Error (MSE): 986719905.6182823
R-squared (R2) Score: 0.8387925201060786


-----------------------------------------------------------------------------
------------------------
Model: Random Forest
Cross-Validation Mean R2 Score: 0.8527200327162129
Mean Squared Error (MSE): 914411112.4992875
R-squared (R2) Score: 0.8506061241962687


-----------------------------------------------------------------------------
------------------------
Model: XGBoost
Cross-Validation Mean R2 Score: 0.857261758192894
Mean Squared Error (MSE): 1495523809.5941255
R-squared (R2) Score: 0.7556655915287752


-----------------------------------------------------------------------------
------------------------
```

Typesetting math: 0%

Typesetting math: 0%