Importing Libaries

In [115]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
pd.options.display.max_columns=150
pd.options.display.max_columns=200
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from sklearn.model_selection import train_test_split
from sklearn import metrics
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

```

Retreving dataset

In [116]:

```
test=pd.read_csv(r"C:\DsTraining\five dataset for clening\house price\test.csv")
train_data=pd.read_csv(r"C:\DsTraining\five dataset for clening\house price\train.csv")
test_sample=pd.read_csv(r"C:\DsTraining\five dataset for clening\house price\sample_submission.csv")
test=pd.merge(test,test_sample,on="Id")
data=pd.concat([train_data,test],ignore_index=True)
data
```

Out[116]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighb
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2914	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	M
2915	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	M
2916	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	
2917	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	
2918	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	Inside	Mod	

2919 rows × 81 columns

Data Preprocessing and EDA.

In [117]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             2919 non-null   int64
 1   MSSubClass     2919 non-null   int64
 2   MSZoning       2915 non-null   object
 3   LotFrontage    2433 non-null   float64
 4   LotArea        2919 non-null   int64
 5   Street         2919 non-null   object
 6   Alley          198 non-null    object
 7   LotShape       2919 non-null   object
 8   LandContour    2919 non-null   object
 9   Utilities      2917 non-null   object
 10  LotConfig      2919 non-null   object
 11  LandSlope      2919 non-null   object
 12  Neighborhood   2919 non-null   object
 13  Condition1     2919 non-null   object
```

Step-1 find missing values.

In [118]:
```
rows=train_data.shape[0]
rows
r_rows=train_data.isnull().sum()
r_rows
a=r_rows/rows*100
a
```

Out[118]:
```
Id                0.000000
MSSubClass        0.000000
MSZoning          0.000000
LotFrontage      17.739726
LotArea           0.000000
                    ...
MoSold            0.000000
YrSold            0.000000
SaleType          0.000000
SaleCondition     0.000000
SalePrice         0.000000
Length: 81, dtype: float64
```

Step-2 Drop variables having more than 25% missing values,and less than 25% missing values will be fill by using appropriate central tendencies

In [119]:
```
data=data.drop(columns=["Alley","FireplaceQu","PoolQC","Fence","MiscFeature"],axis=1)
```

In [120]:

```
#numerical data we used to fill mean.
data["LotFrontage"]=data["LotFrontage"].fillna(data["LotFrontage"].mean())
data["GarageYrBlt"]=data["GarageYrBlt"].fillna(data["GarageYrBlt"].mean())
data["MasVnrArea"]=data["MasVnrArea"].fillna(data["MasVnrArea"].mean())
data["BsmtFinSF1"]=data["BsmtFinSF1"].fillna(data["BsmtFinSF1"].mean())
data["BsmtUnfSF"]=data["BsmtUnfSF"].fillna(data["BsmtUnfSF"].mean())
data["TotalBsmtSF"]=data["TotalBsmtSF"].fillna(data["TotalBsmtSF"].mean())
data["BsmtFinSF2"]=data["BsmtFinSF2"].fillna(data["BsmtFinSF2"].mean())
data["BsmtFullBath"]=data["BsmtFullBath"].fillna(data["BsmtFullBath"].mean())
data["BsmtHalfBath"]=data["BsmtHalfBath"].fillna(data["BsmtHalfBath"].mean())
data["GarageCars"]=data["GarageCars"].fillna(data["GarageCars"].mean())
data["GarageArea"]=data["GarageArea"].fillna(data["GarageArea"].mean())
```

In [121]:

```
#chracteries data we used to mode or so on.
data["Electrical"]=data["Electrical"].fillna(data["Electrical"].mode()[0])
data["MSZoning"]=data["MSZoning"].fillna(data["MSZoning"].mode()[0])
data["Utilities"]=data["Utilities"].fillna(data["Utilities"].mode()[0])
data["Exterior1st"]=data["Exterior1st"].fillna(data["Exterior1st"].mode()[0])
data["Exterior2nd"]=data["Exterior2nd"].fillna(data["Exterior2nd"].mode()[0])
data["MasVnrType"]=data["MasVnrType"].fillna(data["MasVnrType"].mode()[0])
data["BsmtQual"]=data["BsmtQual"].fillna(data["BsmtQual"].mode()[0])
data["BsmtCond"]=data["BsmtCond"].fillna(data["BsmtCond"].mode()[0])
data["BsmtExposure"]=data["BsmtExposure"].fillna(data["BsmtExposure"].mode()[0])
data["BsmtFinType1"]=data["BsmtFinType1"].fillna(data["BsmtFinType1"].mode()[0])
data["BsmtFinType2"]=data["BsmtFinType2"].fillna(data["BsmtFinType2"].mode()[0])
data["KitchenQual"]=data["KitchenQual"].fillna(data["KitchenQual"].mode()[0])
data["Functional"]=data["Functional"].fillna(data["Functional"].mode()[0])
data["GarageType"]=data["GarageType"].fillna(data["GarageType"].mode()[0])
data["GarageQual"]=data["GarageQual"].fillna(data["GarageQual"].mode()[0])
data["GarageCond"]=data["GarageCond"].fillna(data["GarageCond"].mode()[0])
data["SaleType"]=data["SaleType"].fillna(data["SaleType"].mode()[0])
data["GarageFinish"]=data["GarageFinish"].fillna(data["GarageFinish"].mode()[0])
```

EDA

In [122]:

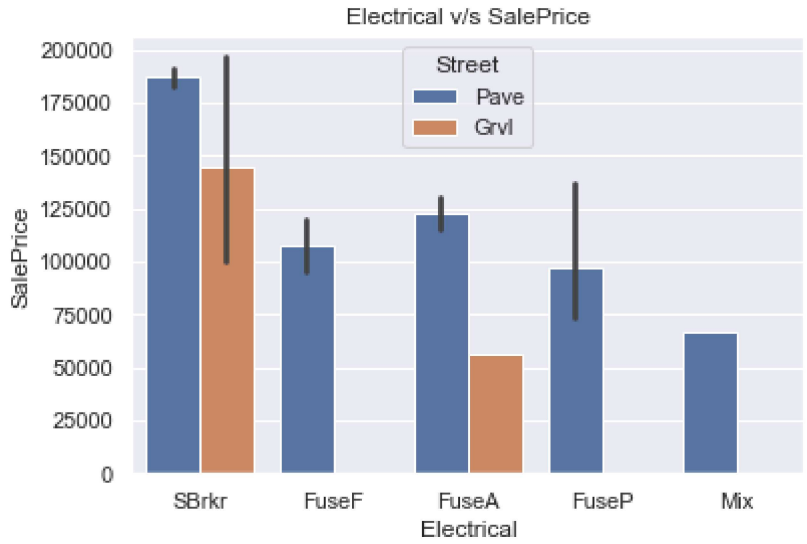
```
#In this graph we see that in street "Pave(Paved)" is more higher as compared to "Grvl(Gravel)".
sns.barplot(y="SalePrice",x="Street",data=train_data)
plt.title("Street v/s SalePrice")
plt.show()
```

In [123]:

```
#In this graph we observed that in "SaleType" new constucted houses and con(15% down payment) at high price
sns.barplot(x="SaleType",y="SalePrice",data=train_data)
plt.title("SaleType v/s SalePrice")
plt.show()
```

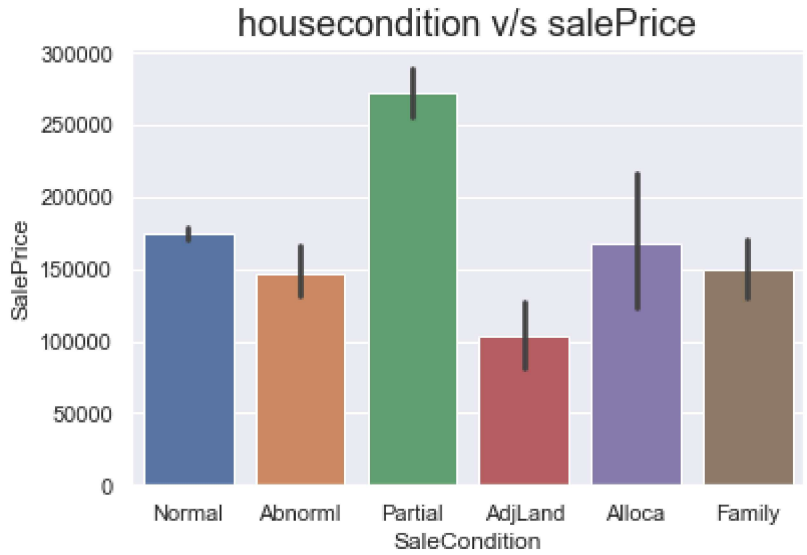
In [124]:

```
#we observed that in electrical(SBrkr) "street" price rate is very high.
sns.barplot(data=train_data,x="Electrical",y="SalePrice",hue="Street")
plt.title("Electrical v/s SalePrice")
plt.show()
```



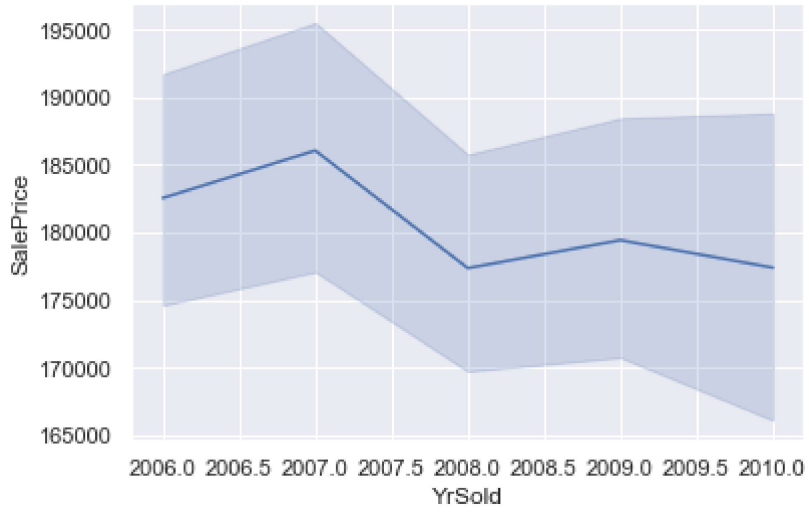
In [125]:

```
# We observed that in salecondition (partial) saleprice is very high.
sns.barplot(data=train_data, y="SalePrice", x="SaleCondition")
plt.title("housecondition v/s salePrice", size=18)
plt.show()
```



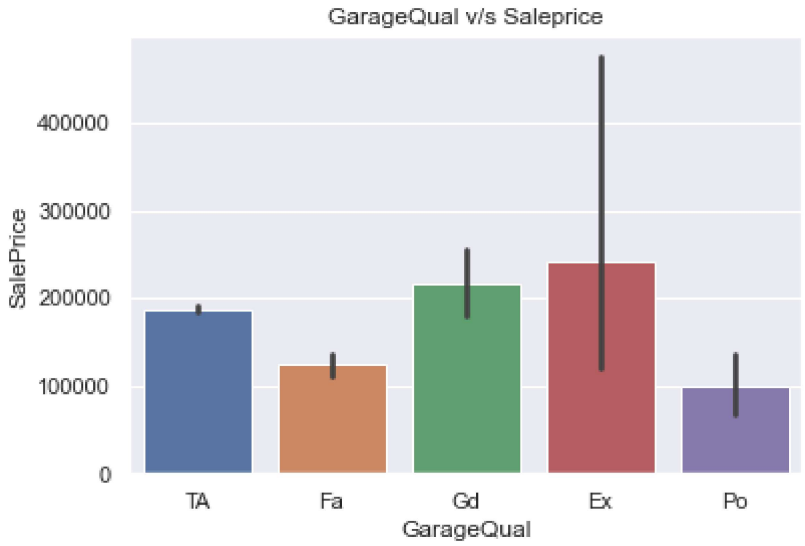
In [126]:

```
#In this graph were we observed saleprice is less than privious(2009) year.
sns.lineplot(data=train_data, x="YrSold", y="SalePrice")
plt.show()
```



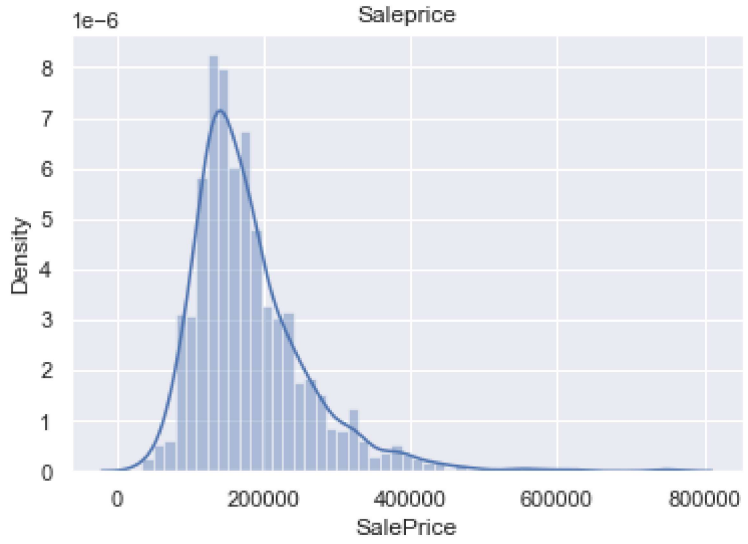
In [127]:

```
#if your Garage quality is excellent and good so the sale price of the house is high
sns.barplot(x="GarageQual",y="SalePrice",data=train_data)
plt.title("GarageQual v/s Saleprice")
plt.show()
```



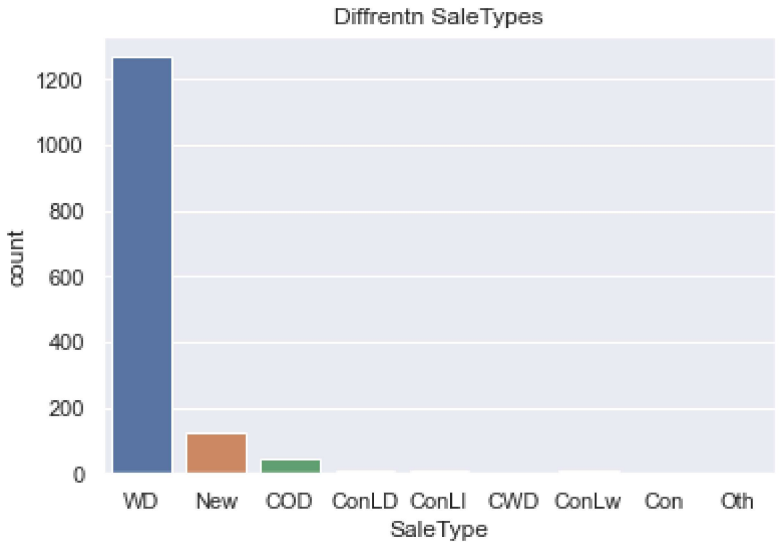
In [128]:

```
#we obseved that maximumm pepole buy houses in the range of 2lakh.
sns.distplot(train_data["SalePrice"])
plt.title("Saleprice")
plt.show()
```



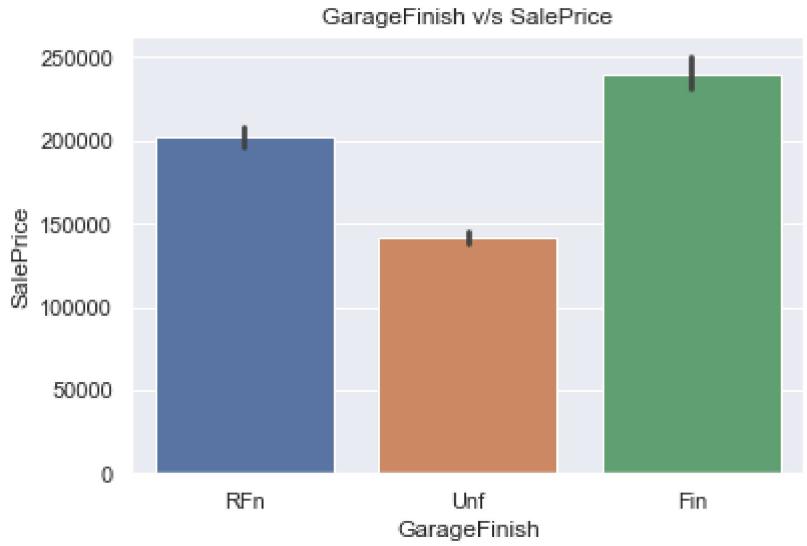
In [129]:

```
sns.countplot(x="SaleType", data=train_data)
plt.title("Diffrentn SaleTypes")
plt.show()
```



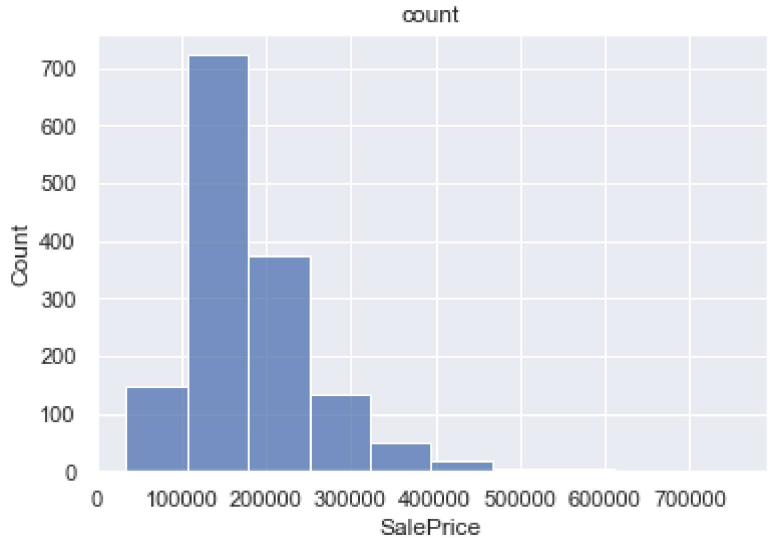
In [130]:

```
#we observed that in GrageFininsh (Finish) sale price is vary high.
sns.barplot(x="GarageFinish",y="SalePrice",data=train_data)
plt.title("GarageFinish v/s SalePrice")
plt.show()
```



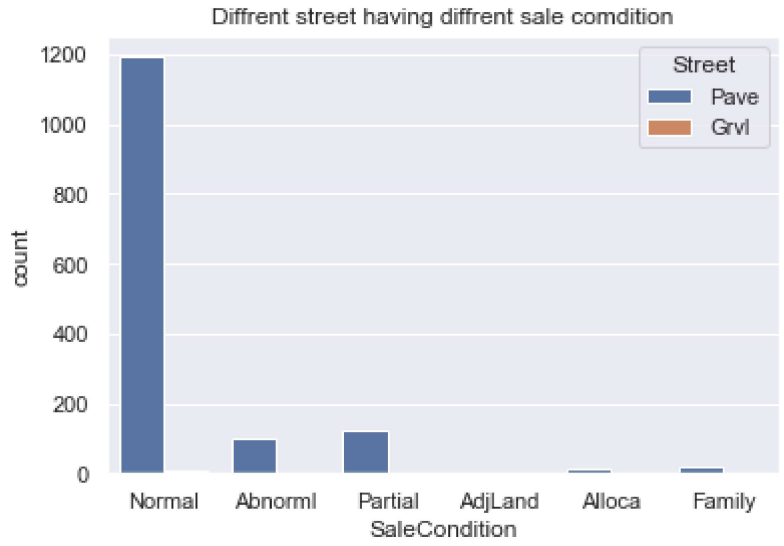
In [131]:

```
#we obseved that maximumm pepole buy houses in the range of 2lakh.
sns.histplot(train_data["SalePrice"],kde=False,bins=10)
plt.title("count")
plt.show()
```



In [132]:

```
#we observed that more pepoles having normal houses in street(Pave-Paved).
sns.countplot(data=train_data, x="SaleCondition", hue="Street")
plt.title("Diffrent street having diffrent sale comdition")
plt.show()
```



In [136]:

```
sns.set(rc={"figure.figsize":(15,8)})
num_col=data.select_dtypes(exclude='object')
sns.heatmap(data.corr(),vmin=1,vmax=-1,annot=True,cmap="Spectral")
plt.show()
```

Multicollinary(using VIF method)

In [134]:

```
vif_data=pd.DataFrame()
vif_data["feature"]=num_col.columns
vif_data["VIF"]=[variance_inflation_factor(num_col.values,i)
                 for i in range(len(num_col.columns))]
vif_data
vif_data.round(1)
data=data.drop(columns=["BsmtFinSF1","LotFrontage","GarageArea","BedroomAbvGr","YrSold","KitchenAbvGr","Tot
#if vif<10 we drop that variable
#variable having range 5 to 10 showa high multicolinarity.
```

Outlier Detection(Extream end outliers will be removed.)

In [135]:

```
data=data.drop(columns=['LotArea','MasVnrArea','WoodDeckSF','OpenPorchSF','EnclosedPorch','ScreenPorch'],ax
```

In [137]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 50 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              2919 non-null   int64
 1   MSSubClass      2919 non-null   int64
 2   MSZoning        2919 non-null   object
 3   Street          2919 non-null   object
 4   LotShape        2919 non-null   object
 5   LandContour     2919 non-null   object
 6   Utilities       2919 non-null   object
 7   LotConfig       2919 non-null   object
 8   LandSlope       2919 non-null   object
 9   Neighborhood    2919 non-null   object
 10  Condition1      2919 non-null   object
 11  Condition2      2919 non-null   object
 12  BldgType        2919 non-null   object
 13  HouseStyle      2919 non-null   object
 14  OverallCond     2919 non-null   int64
 15  RoofStyle       2919 non-null   object
 16  RoofMatl        2919 non-null   object
 17  Exterior1st     2919 non-null   object
 18  Exterior2nd     2919 non-null   object
 19  MasVnrType      2919 non-null   object
 20  ExterQual       2919 non-null   object
 21  ExterCond       2919 non-null   object
 22  Foundation      2919 non-null   object
 23  BsmtQual        2919 non-null   object
 24  BsmtCond        2919 non-null   object
 25  BsmtExposure    2919 non-null   object
 26  BsmtFinType1    2919 non-null   object
 27  BsmtFinType2    2919 non-null   object
 28  Heating         2919 non-null   object
 29  HeatingQC       2919 non-null   object
 30  CentralAir      2919 non-null   object
 31  Electrical      2919 non-null   object
 32  BsmtFullBath    2919 non-null   float64
 33  BsmtHalfBath    2919 non-null   float64
 34  HalfBath        2919 non-null   int64
```

```
 35  KitchenQual    2919 non-null   object
 36  Functional     2919 non-null   object
 37  Fireplaces     2919 non-null   int64
 38  GarageType     2919 non-null   object
 39  GarageFinish   2919 non-null   object
 40  GarageQual     2919 non-null   object
 41  GarageCond     2919 non-null   object
 42  PavedDrive     2919 non-null   object
 43  3SsnPorch      2919 non-null   int64
 44  PoolArea       2919 non-null   int64
 45  MiscVal        2919 non-null   int64
 46  MoSold         2919 non-null   int64
 47  SaleType       2919 non-null   object
 48  SaleCondition  2919 non-null   object
 49  SalePrice      2919 non-null   float64
dtypes: float64(3), int64(9), object(38)
memory usage: 1.1+ MB
```

Fill appropriate outlier using z_score method.

In [138]:

```
upper_limit=data['MSSubClass'].mean()+3*data['MSSubClass'].std()

lower_limit=data['MSSubClass'].mean()-3*data['MSSubClass'].std()
data['MSSubClass'] = np.where(
    data['MSSubClass']>upper_limit,
    upper_limit,
    np.where(
        data['MSSubClass']<0,
        0,
        data['MSSubClass']
    )
)
```

In [139]:
```
upper_limit=data['BsmtFullBath'].mean()+3*data['BsmtFullBath'].std()

lower_limit=data['BsmtFullBath'].mean()-3*data['BsmtFullBath'].std()
data['BsmtFullBath'] = np.where(
    data['BsmtFullBath']>upper_limit,
    upper_limit,
    np.where(
        data['BsmtFullBath']<0,
        0,
        data['BsmtFullBath']
    )
)
```

In [140]:
```
upper_limit=data['BsmtHalfBath'].mean()+3*data['BsmtHalfBath'].std()

lower_limit=data['BsmtHalfBath'].mean()-3*data['BsmtHalfBath'].std()
data['BsmtHalfBath'] = np.where(
    data['BsmtHalfBath']>upper_limit,
    upper_limit,
    np.where(
        data['BsmtHalfBath']<0,
        0,
        data['BsmtHalfBath']
    )
)
```

In [141]:
```
upper_limit=data['Fireplaces'].mean()+3*data['Fireplaces'].std()

lower_limit=data['Fireplaces'].mean()-3*data['Fireplaces'].std()
data['Fireplaces'] = np.where(
    data['Fireplaces']>upper_limit,
    upper_limit,
    np.where(
        data['Fireplaces']<0,
        0,
        data['Fireplaces']
    )
)
```

In [142]:

```
upper_limit=data['3SsnPorch'].mean()+3*data['3SsnPorch'].std()

lower_limit=data['3SsnPorch'].mean()-3*data['3SsnPorch'].std()
data['3SsnPorch'] = np.where(
    data['3SsnPorch']>upper_limit,
    upper_limit,
    np.where(
        data['3SsnPorch']<0,
        0,
        data['3SsnPorch']
    )
)
```

In [143]:

```
upper_limit=data['PoolArea'].mean()+3*data['PoolArea'].std()

lower_limit=data['PoolArea'].mean()-3*data['PoolArea'].std()
data['PoolArea'] = np.where(
    data['PoolArea']>upper_limit,
    upper_limit,
    np.where(
        data['PoolArea']<0,
        0,
        data['PoolArea']
    )
)
```

In [144]:

```
upper_limit=data['MiscVal'].mean()+3*data['MiscVal'].std()

lower_limit=data['MiscVal'].mean()-3*data['MiscVal'].std()
data['MiscVal'] = np.where(
    data['MiscVal']>upper_limit,
    upper_limit,
    np.where(
        data['MiscVal']<0,
        0,
        data['MiscVal']
    )
)
```

All object converted into integers.

In [145]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
object_list=data.select_dtypes(include=['object']).columns
for i in object_list:
    data[i]=le.fit_transform(data[i])
```

Train-Test-Spliting

In [146]:
```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

In [147]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

Linear Regression

In [148]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
print(f'The score of Linear regression is {reg.score(x_test,y_test)*100} %')
```

```
The score of Linear regression is 33.90204040111284 %
```

In [149]:
```
y_pred=reg.predict(x_test)
```

XGBOOST Modeling.

In [150]:
```
import xgboost as xgb
xgb.XGBRegressor().get_params()
xg_reg=xgb.XGBRegressor()
xg_reg.fit(x_train,y_train)
print(f'The score of xgboost is {xg_reg.score(x_test,y_test)*100} %')
```

```
The score of xgboost is 69.23874049911308 %
```

In [151]:
```
xgb_preds = xg_reg.predict(x_test)
```

Decision Tree

In [152]:
```
from sklearn.tree import DecisionTreeRegressor
regr=DecisionTreeRegressor()
regr.fit(x_train,y_train)
print(f'The score of Decision Tree is {reg.score(x_test,y_test)*100}%')
```

```
The score of Decision Tree is 33.90204040111284%
```

In [153]:
```
y_pred1=regr.predict(x_test)
```

In [ ]:
```

```

In [ ]:
```

```