House Price Prediction-EDA

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

In [5]:
```
df=pd.read_csv("C:\\Users\\liji\\Downloads\\train.csv")
df.head()
```

Out[5]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeatur
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	Nal
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	Nal
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	Nal
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	Nal
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	Nal

5 rows × 81 columns

In [8]:
```
#print date of the data set with rows and columns
df.shape
```

Out[8]: (1460, 81)

In Data analysis we will analyze to find the below stuff

- Missing Values
- All the numerical variables

- Distribution of the numerical variables
- Categorical Variables
- Outliers
- Relationship between independent and dependent features
- Correlations

In [10]:
```
# Missing values
df.isnull().sum()
```

Out[10]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

In [11]:
```
##these are the features with nan value
features_with_na=[features for features in df.columns if df[features].isnull().sum()>=1]
```

In [12]:
```
1 features_with_na
```

Out[12]:
```
['LotFrontage',
 'Alley',
 'MasVnrType',
 'MasVnrArea',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Electrical',
 'FireplaceQu',
 'GarageType',
 'GarageYrBlt',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PoolQC',
 'Fence',
 'MiscFeature']
```

In [13]:

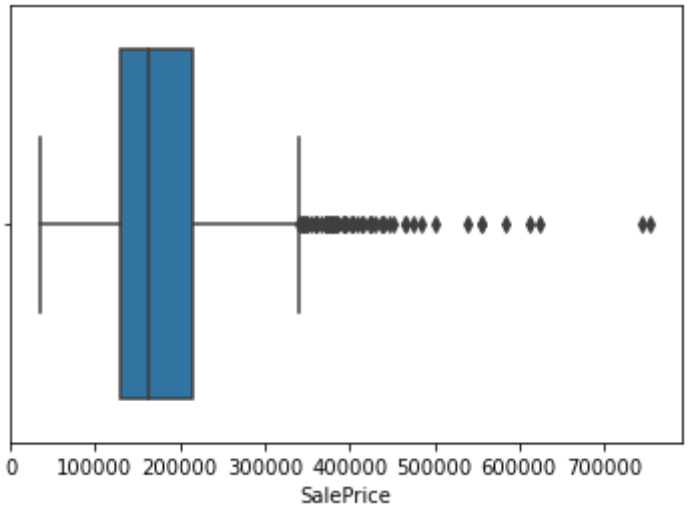
```
#dataset['Electrical'].isnull().mean()
#This will round up the percentage upto 5 decimal places.
for feature in features_with_na:
    print(feature,np.round(df[feature].isnull().mean()*100,5), '% missing values')
```

```
LotFrontage 17.73973 % missing values
Alley 93.76712 % missing values
MasVnrType 0.54795 % missing values
MasVnrArea 0.54795 % missing values
BsmtQual 2.53425 % missing values
BsmtCond 2.53425 % missing values
BsmtExposure 2.60274 % missing values
BsmtFinType1 2.53425 % missing values
BsmtFinType2 2.60274 % missing values
Electrical 0.06849 % missing values
FireplaceQu 47.26027 % missing values
GarageType 5.54795 % missing values
GarageYrBlt 5.54795 % missing values
GarageFinish 5.54795 % missing values
GarageQual 5.54795 % missing values
GarageCond 5.54795 % missing values
PoolQC 99.52055 % missing values
Fence 80.75342 % missing values
MiscFeature 96.30137 % missing values
```

In [15]:
```
1 sns.boxplot(df['SalePrice'])
```

```
C:\Users\liji\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as
a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[15]: <AxesSubplot:xlabel='SalePrice'>

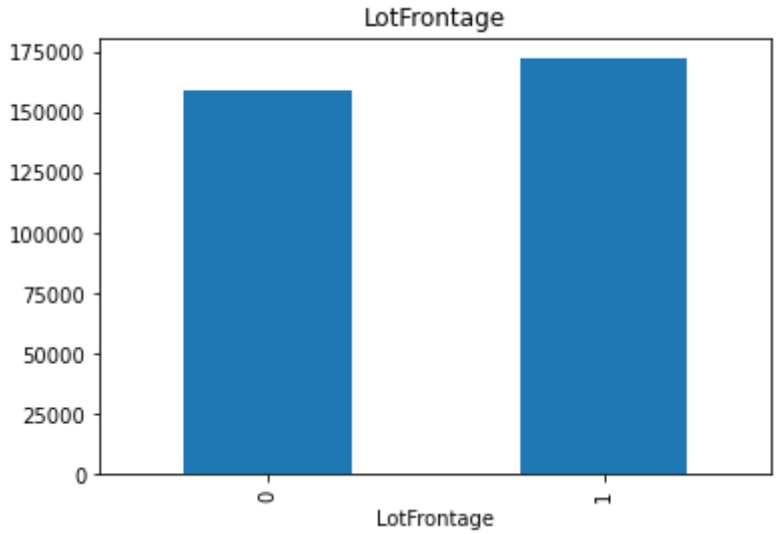


In [16]:

```
## let's plot some amazing diagrams
data=df.copy()
for feature in features_with_na:

    # let's make a variables that indicate 1 if the observation was missing or 0 otherwise
    data[feature]=np.where(data[feature].isnull(),1,0)

    # Lets calculate the mean SalePrice where the information is missing or present
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.title(feature)
    plt.show()
```



In [17]:
```
1 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 [18]:

```
## list of numerical variables

df['SaleType'].dtypes!='O'
```

Out[18]: False

In [19]:

```
numerical_features=[feature for feature in df.columns if df[feature].dtypes!='O']
```

In [21]:

```
print(len(numerical_features))
df[numerical_features].head()
```

38

Out[21]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeckSF	Op
0	1	60	65.0	8450	7	5	2003	2003	196.0	706	...	0	
1	2	20	80.0	9600	6	8	1976	1976	0.0	978	...	298	
2	3	60	68.0	11250	7	5	2001	2002	162.0	486	...	0	
3	4	70	60.0	9550	7	5	1915	1970	0.0	216	...	0	
4	5	60	84.0	14260	8	5	2000	2000	350.0	655	...	192	

5 rows × 38 columns

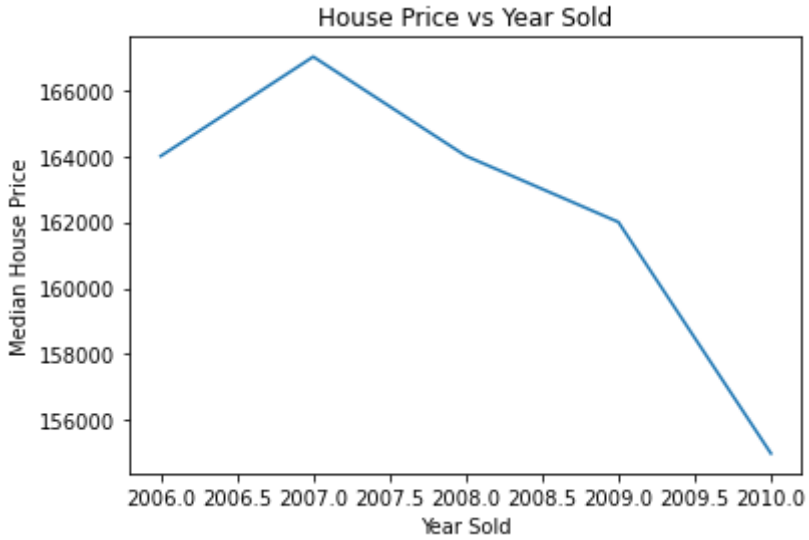
In [23]:
```
## Temporal Variables(Eg: Datetime Variables)
year_feature=[feature for feature in numerical_features if 'Yr' in feature or 'Year' in feature]
```

In [25]:
```
for feature in year_feature:
    print(feature,df[feature].unique())
```

```
YearBuilt [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005 1962 2006
 1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
 1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
 1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
 1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
 1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
 1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
 1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
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]
GarageYrBlt [2003. 1976. 2001. 1998. 2000. 1993. 2004. 1973. 1931. 1939. 1965. 2005.
 1962. 2006. 1960. 1991. 1970. 1967. 1958. 1930. 2002. 1968. 2007. 2008.
 1957. 1920. 1966. 1959. 1995. 1954. 1953.   nan 1983. 1977. 1997. 1985.
 1963. 1981. 1964. 1999. 1935. 1990. 1945. 1987. 1989. 1915. 1956. 1948.
 1974. 2009. 1950. 1961. 1921. 1900. 1979. 1951. 1969. 1936. 1975. 1971.
 1923. 1984. 1926. 1955. 1986. 1988. 1916. 1932. 1972. 1918. 1980. 1924.
 1996. 1940. 1949. 1994. 1910. 1978. 1982. 1992. 1925. 1941. 2010. 1927.
 1947. 1937. 1942. 1938. 1952. 1928. 1922. 1934. 1906. 1914. 1946. 1908.
 1929. 1933.]
YrSold [2008 2007 2006 2009 2010]
```

In [26]:
```
df.groupby('YrSold')['SalePrice'].median().plot()
plt.xlabel('Year Sold')
plt.ylabel('Median House Price')
plt.title('House Price vs Year Sold')
```

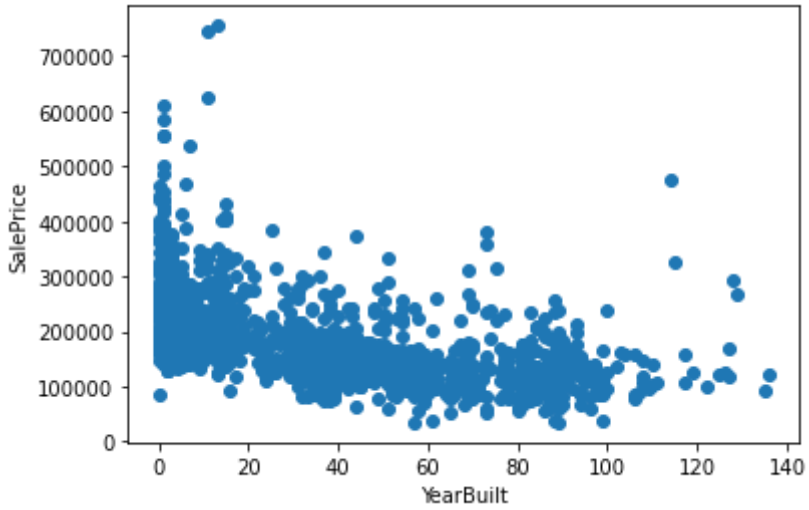
Out[26]: Text(0.5, 1.0, 'House Price vs Year Sold')

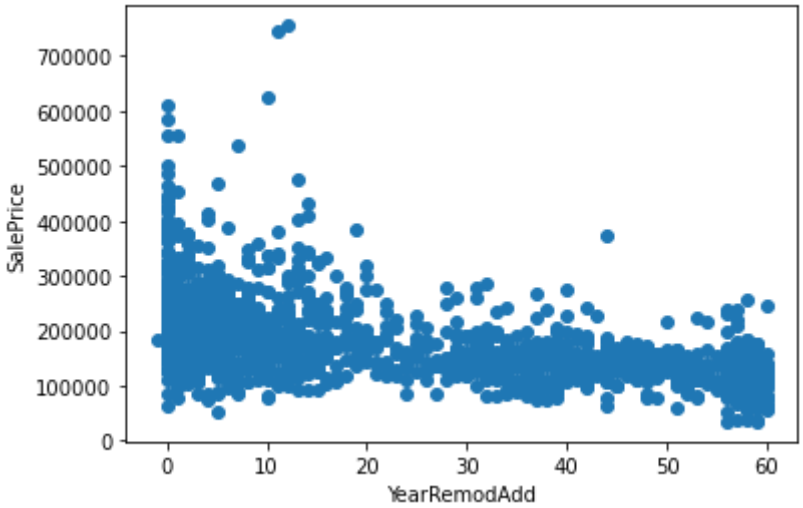


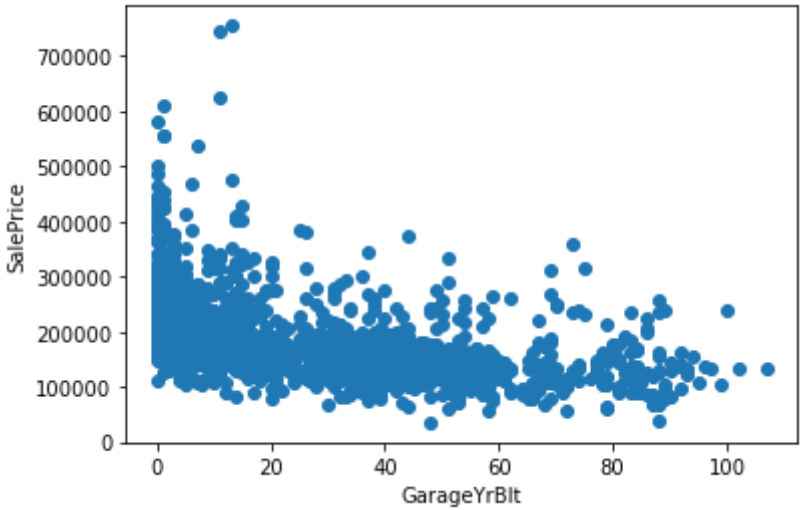
In [27]:

```
### Here we will compare the difference between all years features with Salesprice
data=df.copy()
for feature in year_feature:
    if feature!='YrSold':
        data[feature]=data['YrSold']-data[feature]

        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalePrice')
        plt.show()
```




SalePrice
YearRemodAdd
0
100000
200000
300000
400000
500000
600000
700000
0
10
20
30
40
50
60


SalePrice
GarageYrBlt
0
100000
200000
300000
400000
500000
600000
700000
0
20
40
60
80
100

Observations:

Within all the the yearwise information, it is clearly visible that the the New houses are more costlier that the older one. looks like a pareto or power law distribution\

In [29]:
```
### numerical variables- 2 Types
##1. Continuous variable and Discrete variable

discrete_feature=[feature for feature in numerical_features if len(df[feature].unique())<=25]
print(len(discrete_feature))
```

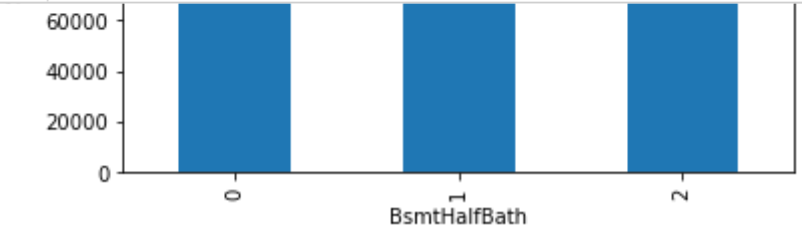
```
18
```

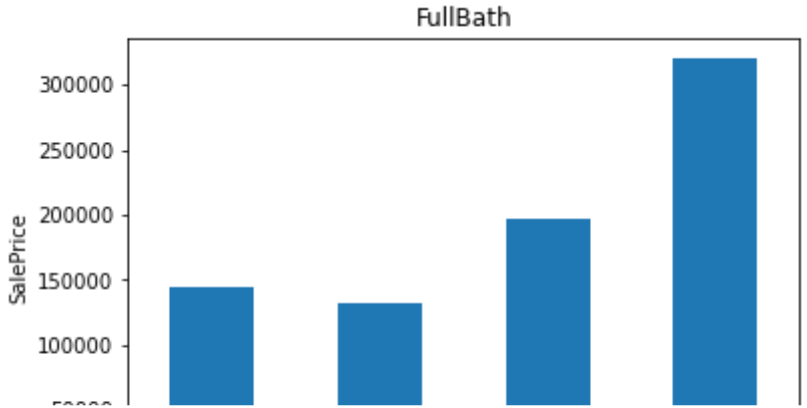
In [30]:
```
df[discrete_feature].head()
```

Out[30]:

	MSSubClass	OverallQual	OverallCond	LowQualFinSF	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr	KitchenAbvGr	TotR
0	60	7	5	0	1	0	2	1	3	1	
1	20	6	8	0	0	1	2	0	3	1	
2	60	7	5	0	1	0	2	1	3	1	
3	70	7	5	0	1	0	1	0	3	1	
4	60	8	5	0	1	0	2	1	4	1	

In [31]:
```
## Lets find the relationship between Discrete and Sales Price
data=df.copy()
for feature in discrete_feature:
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.title(feature)
    plt.show()
```



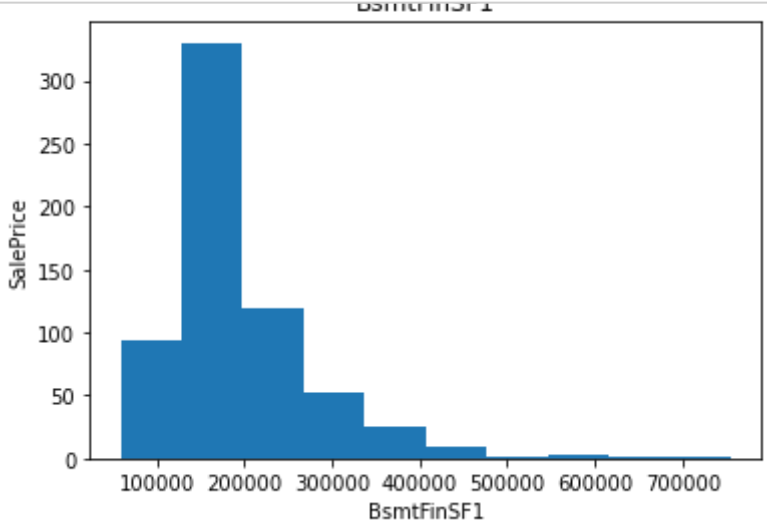


In [32]:
```
## There is a relationship between Discrete features and SalePrice
## Continuous Variable
continuous_feature=[feature for feature in numerical_features if feature not in discrete_feature+year_feature]
print(len(continuous_feature))
```

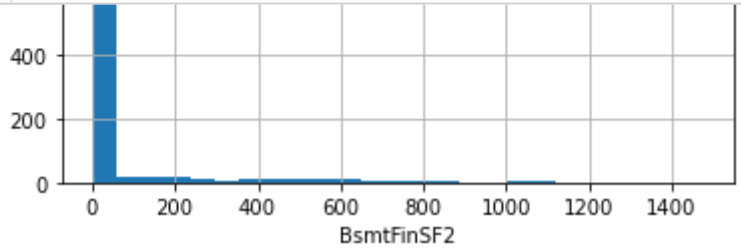
17

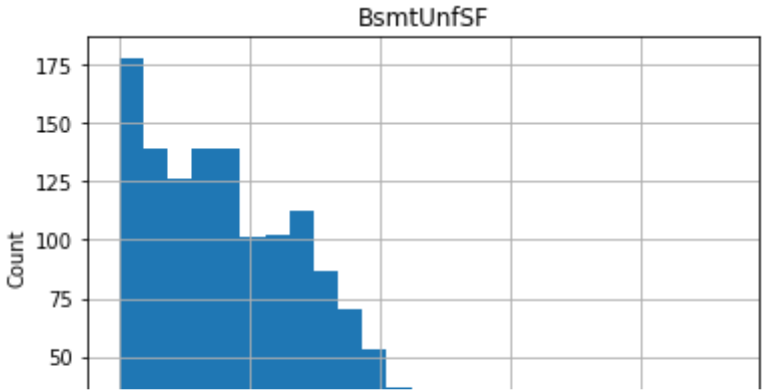
In [33]:

```
data = df.copy()
for feature in continuous_feature:
    data.groupby(feature)['SalePrice'].median().plot.hist()
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.title(feature)
    plt.show()
```



In [34]:
```
### Lets analyze the continous values by creating histograms to understand the distribution
data=df.copy()
for feature in continuous_feature:
    data[feature].hist(bins=25)
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.show()
```

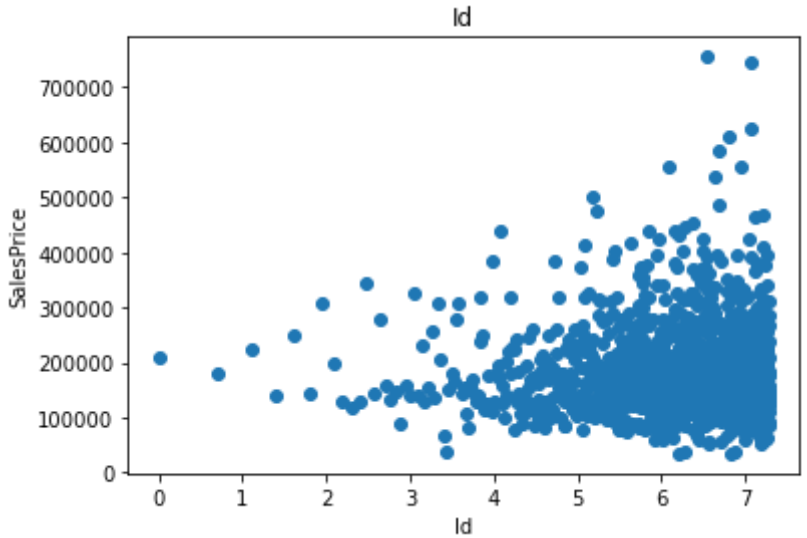


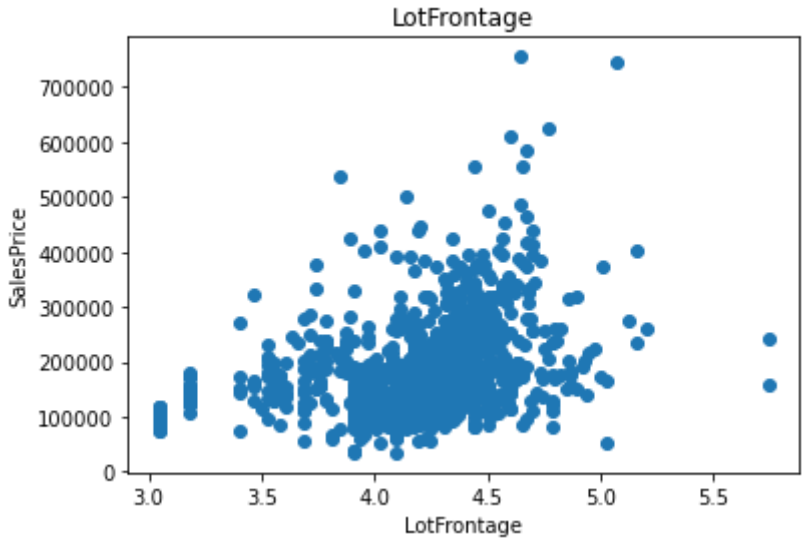


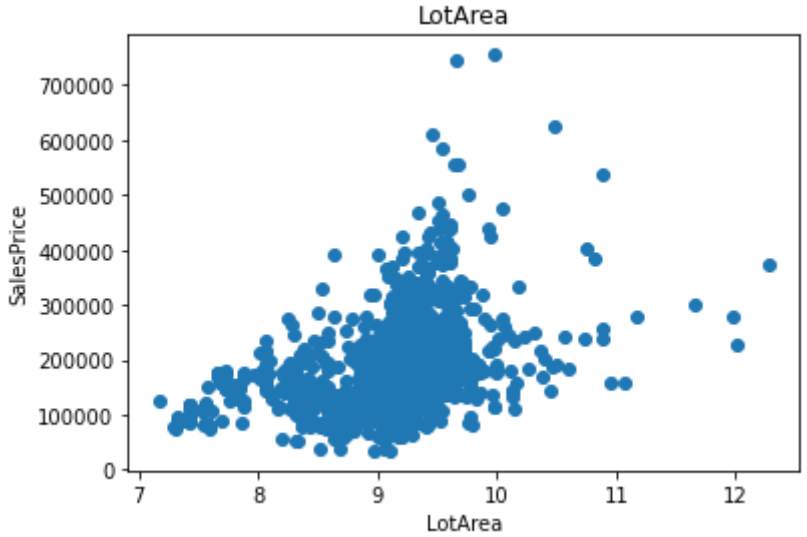
EDA Part 2¶

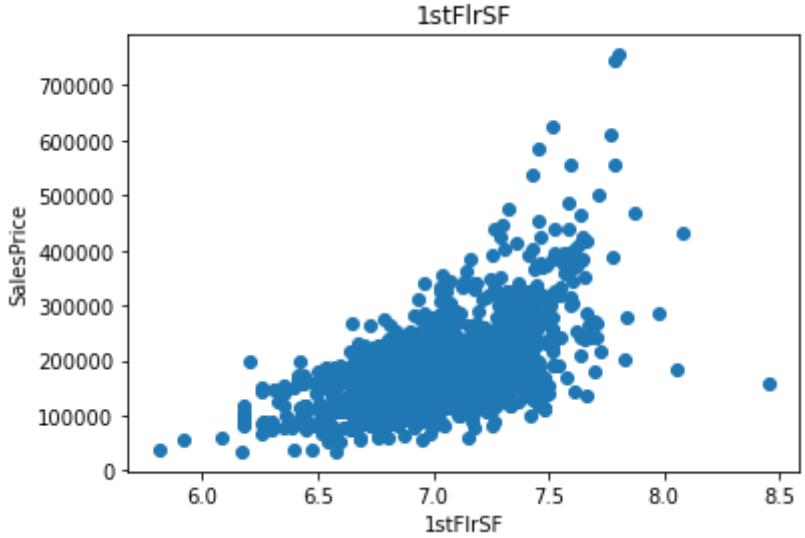
In [35]:

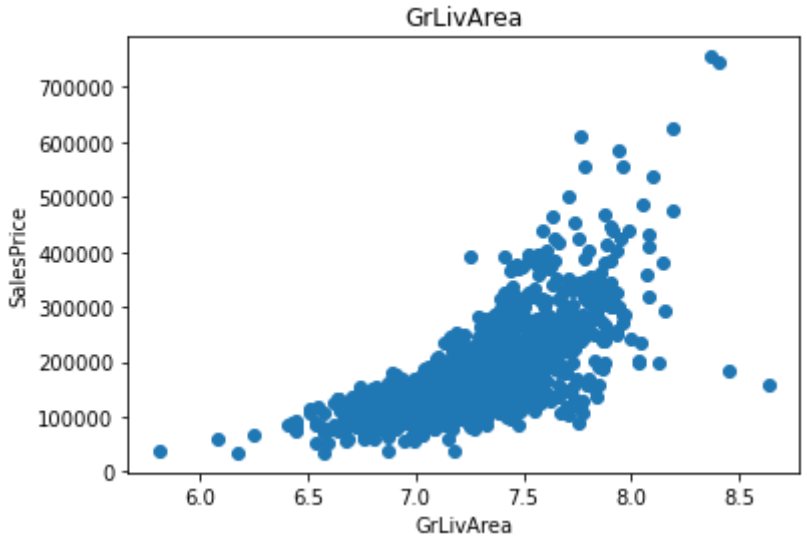
```
## We will be using logarithmic transformation
data=df.copy()
for feature in continuous_feature:
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])
        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalesPrice')
        plt.title(feature)
        plt.show()
```

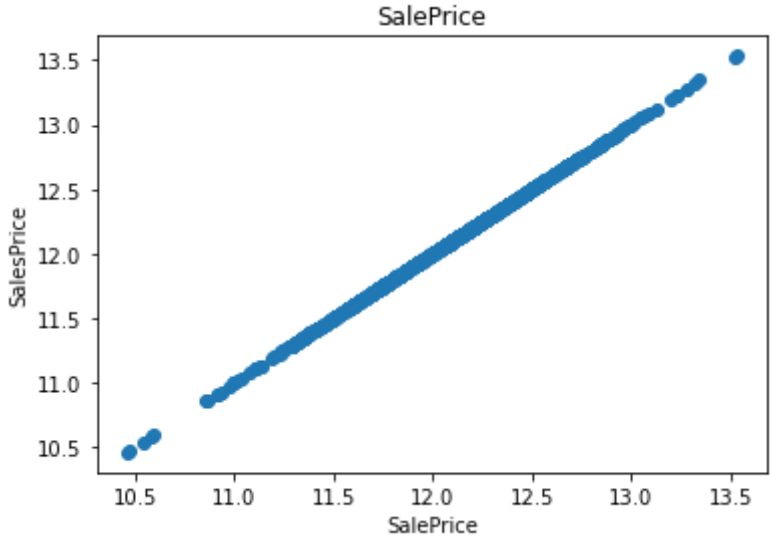



LotFrontage
SalesPrice
LotFrontage
0
100000
200000
300000
400000
500000
600000
700000
3.0
3.5
4.0
4.5
5.0
5.5


LotArea
SalesPrice
LotArea
0
100000
200000
300000
400000
500000
600000
700000
7
8
9
10
11
12

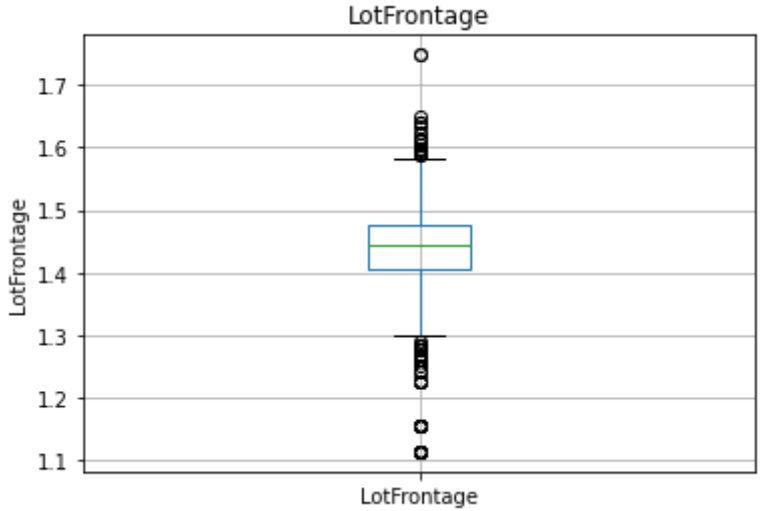

1stFlrSF
SalesPrice
700000
600000
500000
400000
300000
200000
100000
0
6.0
6.5
7.0
7.5
8.0
8.5
1stFlrSF

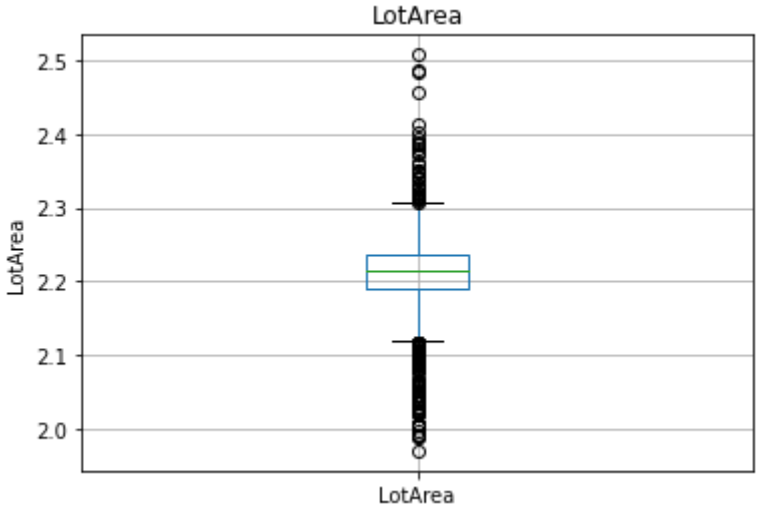

GrLivArea
SalesPrice
GrLivArea
0
100000
200000
300000
400000
500000
600000
700000
6.0
6.5
7.0
7.5
8.0
8.5

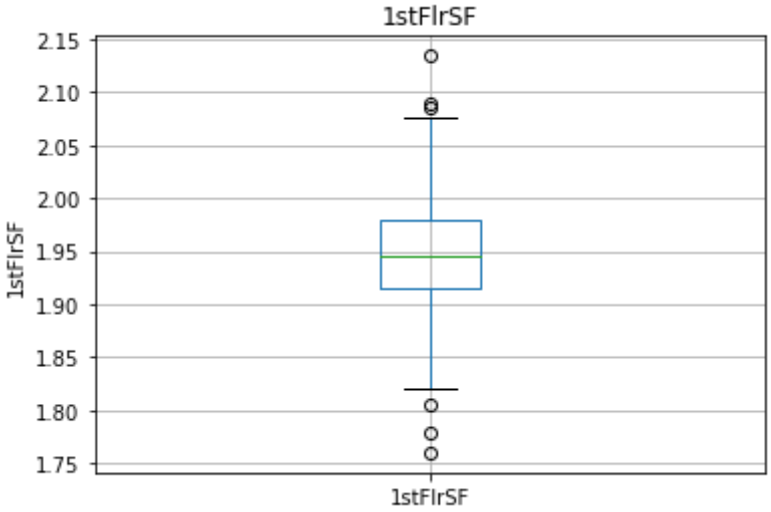

SalePrice
SalesPrice
SalePrice
10.5
11.0
11.5
12.0
12.5
13.0
13.5

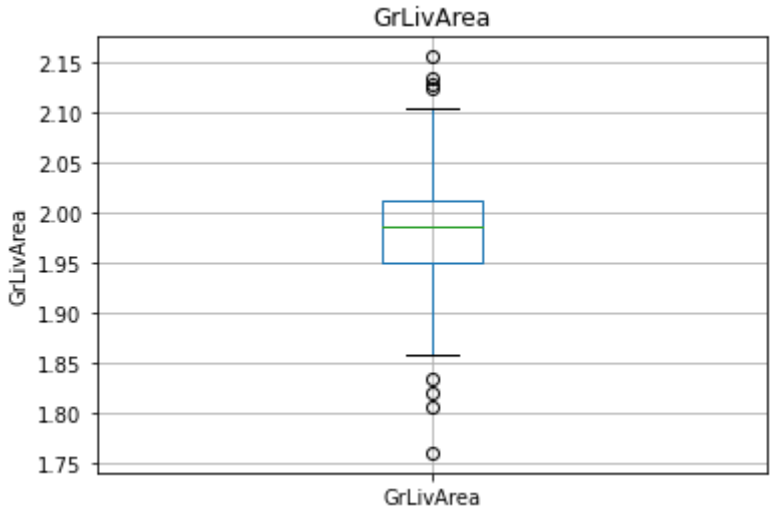
In [36]:

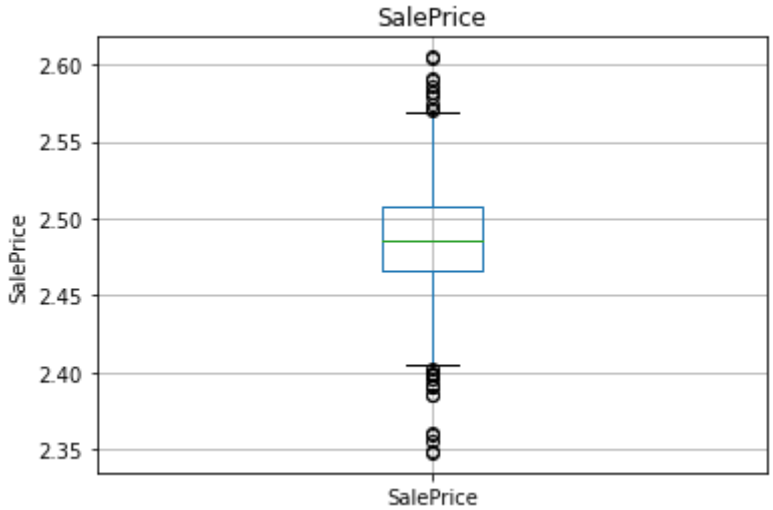
```
## Outliers
for feature in continuous_feature:
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])
        data.boxplot(column=feature)
        plt.ylabel(feature)
        plt.title(feature)
        plt.show()
```




LotArea
2.5
2.4
2.3
2.2
2.1
2.0
LotArea
LotArea

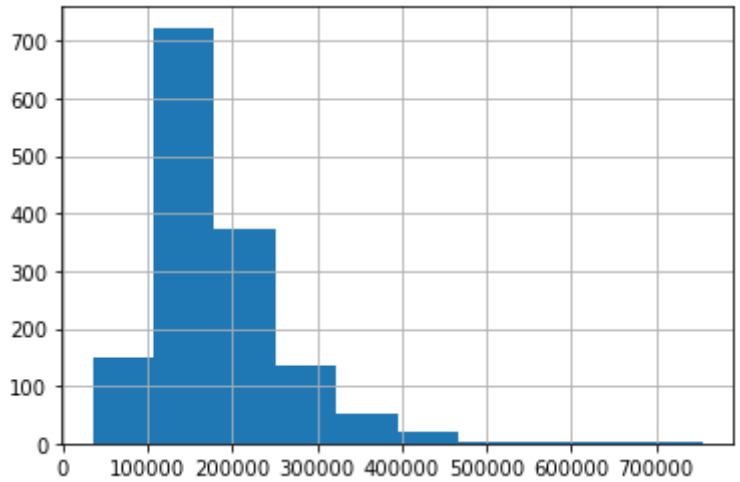

1stFlrSF
2.15
2.10
2.05
2.00
1.95
1.90
1.85
1.80
1.75
1stFlrSF
1stFlrSF


GrLivArea
2.15
2.10
2.05
2.00
1.95
1.90
1.85
1.80
1.75
GrLivArea
GrLivArea


SalePrice
2.60
2.55
2.50
2.45
2.40
2.35
SalePrice
SalePrice

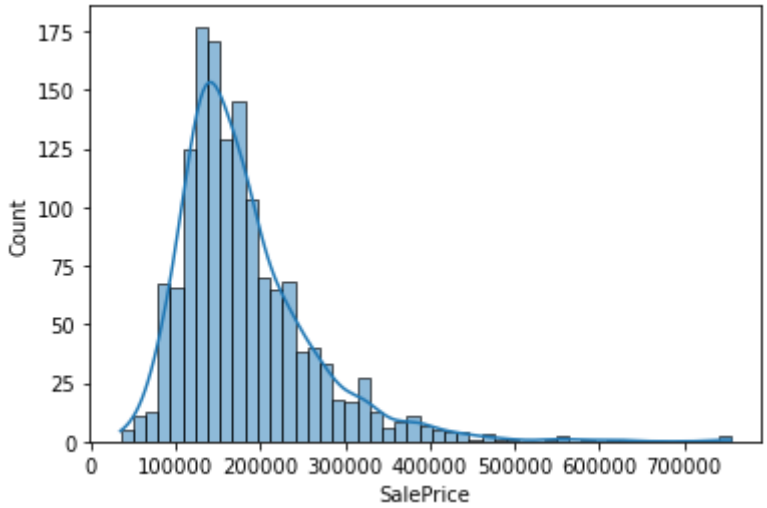
In [38]:
```
1 df['SalePrice'].hist()
```

Out[38]: <AxesSubplot:>



In [39]:
```
1 sns.histplot(df['SalePrice'],kde=True)
```

Out[39]: <AxesSubplot:xlabel='SalePrice', ylabel='Count'>



Categorical Variables

In [40]:
```
1 categorical_features=[feature for feature in df.columns if df[feature].dtypes=='O']
```

In [41]:
```
1 df[categorical_features].head()
```

Out[41]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageType	GarageFinish
0	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	...	Attchd	RFn
1	RL	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	...	Attchd	RFn
2	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	...	Attchd	RFn
3	RL	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	...	Detchd	Unf
4	RL	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	...	Attchd	RFn

5 rows × 43 columns

In [42]:

```
for feature in categorical_features:
    print(" The feature name is {} and the number of categories are {}".format(feature,len(df[feature].unique()))
```

```
The feature name is MSZoning and the number of categories are 5
The feature name is Street and the number of categories are 2
The feature name is Alley and the number of categories are 3
The feature name is LotShape and the number of categories are 4
The feature name is LandContour and the number of categories are 4
The feature name is Utilities and the number of categories are 2
The feature name is LotConfig and the number of categories are 5
The feature name is LandSlope and the number of categories are 3
The feature name is Neighborhood and the number of categories are 25
The feature name is Condition1 and the number of categories are 9
The feature name is Condition2 and the number of categories are 8
The feature name is BldgType and the number of categories are 5
The feature name is HouseStyle and the number of categories are 8
The feature name is RoofStyle and the number of categories are 6
The feature name is RoofMatl and the number of categories are 8
The feature name is Exterior1st and the number of categories are 15
The feature name is Exterior2nd and the number of categories are 16
The feature name is MasVnrType and the number of categories are 5
The feature name is ExterQual and the number of categories are 4
```

In [44]:

```
## find the relationship between categorical feature and Salesprice
data=df.copy()
for feature in categorical_features:
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.title(feature)
    plt.show()
```