In [6]:
```python
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#ignore/ disable warnings
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [7]:
```python
#import data
df=pd.read_csv(r"C:\Users\Sanjay Lohar\Downloads\House_Prices.csv")
```

In [8]:
```python
df.head()
```

Out[8]:

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790.0	2	2	2	No	East
1	2	114200	2030.0	4	2	3	No	East
2	3	114800	1740.0	3	2	1	No	East
3	4	94700	1980.0	3	2	3	No	East
4	5	119800	2130.0	3	3	3	No	East

In [9]:
```python
#check shape, dtypes
df.shape #128 observations, 8 variables
```

Out[9]:
```
(128, 8)
```

In [10]:
```python
df.dtypes #all variables are stored with appropriate data types
```

Out[10]:
```
Home              int64
Price             int64
SqFt            float64
Bedrooms          int64
Bathrooms         int64
Offers            int64
Brick            object
Neighborhood     object
dtype: object
```

In [11]:
```python
# check missing values
df.isnull().sum() #there are 2 missing values in the data
```

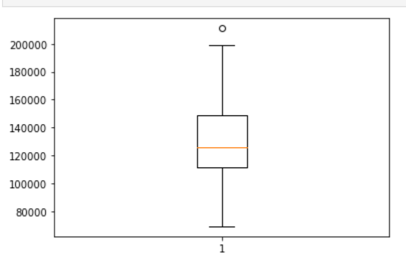
Out[11]:
```
Home            0
Price           0
SqFt            1
Bedrooms        0
Bathrooms       0
Offers          0
Brick           0
Neighborhood    1
dtype: int64
```

In [12]:
```python
#missing value treatment (Remove records having missing values)
df=df.dropna(axis=0)
```

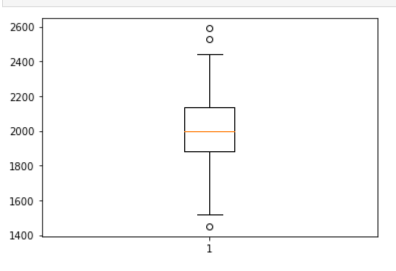
In [13]:
```python
#check whther missing value records are removed or not
df.isnull().sum()
```

Out[13]:
```
Home            0
Price           0
SqFt            0
Bedrooms        0
Bathrooms       0
Offers          0
Brick           0
Neighborhood    0
dtype: int64
```

In [14]:
```python
#Check outliers
plt.boxplot(df['Price']) #has outliers
plt.show()
```



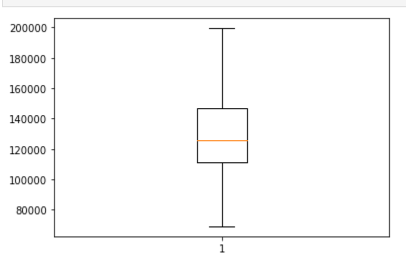
In [15]:
```python
plt.boxplot(df['SqFt']) #has outliers
plt.show()
```



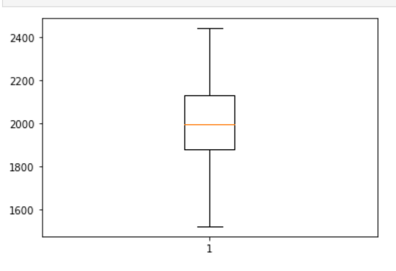
In [16]:
```python
#outlier treatement
#we will write a user defined function to remove outliers
```

In [17]:
```python
def remove_outlier(d,c):
    #find q1 and q3
    q1=d[c].quantile(0.25)
    q3=d[c].quantile(0.75)

    #find iqr (inter quartile range)
    iqr=q3-q1

    #find upper and lowe bound
    ub=q3+1.5*iqr
    lb=q1-1.5*iqr

    final_data= d[(d[c]>lb) & (d[c]<ub)]
    return final_data
```

In [26]:
```python
#remove outliers from price
df=remove_outlier(df,"Price")
plt.boxplot(df['Price'])
plt.show()
```

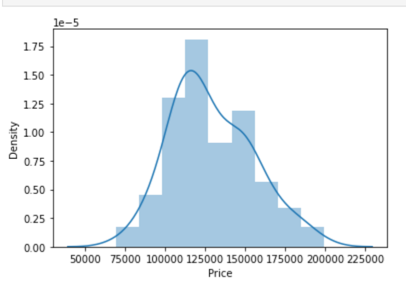


In [33]:
```python
#remove outliers from SqFt
df=remove_outlier(df,"SqFt")
plt.boxplot(df['SqFt'])
plt.show()
```

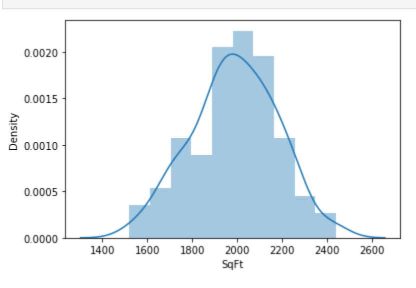


In [34]:
```python
#--------------------Start EDA (Exploratory Data Analysis)----------------
```

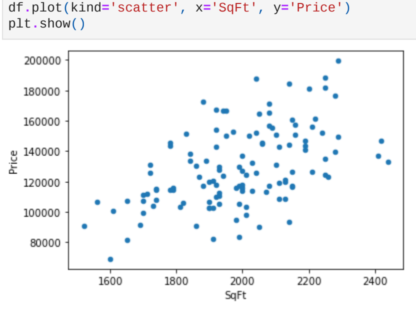
In [35]:
```python
#Price distribution
sns.distplot(df["Price"])
plt.show()
```



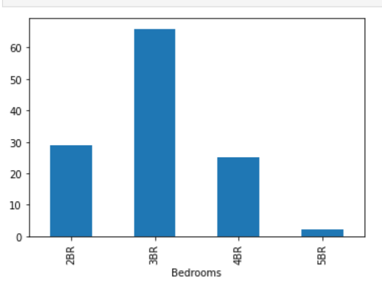
In [36]:
```python
#SqFt Distribution
sns.distplot(df["SqFt"])
plt.show()
```



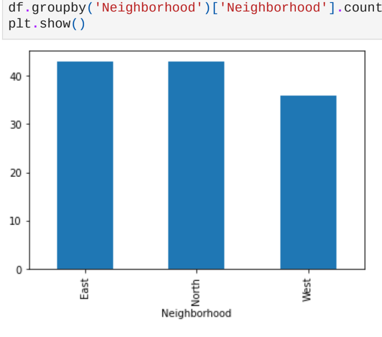
In [37]:
```python
#Correlation between Price and SqFt
df.plot(kind='scatter', x='SqFt', y='Price')
plt.show()
```



In [66]:
```python
#count of house by bedrooms
df.groupby('Bedrooms')['Bedrooms'].count().plot(kind='bar')
plt.show()
```



In [39]:
```python
#count of house by neighborhood
df.groupby('Neighborhood')['Neighborhood'].count().plot(kind='bar')
plt.show()
```



In [40]:
```python
#Check the correlation of numeric variables
df_numeric = df.select_dtypes(include=['float64', 'int64'])
df_numeric.head()
```

Out[40]:

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers
0	1	114300	1790.0	2	2	2
1	2	114200	2030.0	4	2	3
2	3	114800	1740.0	3	2	1
3	4	94700	1980.0	3	2	3
4	5	119800	2130.0	3	3	3

In [41]:
```python
# drop Home variable because it is just a serial number
df_numeric = df_numeric.drop('Home', axis=1)
df_numeric.head()
```

Out[41]:

	Price	SqFt	Bedrooms	Bathrooms	Offers
0	114300	1790.0	2	2	2
1	114200	2030.0	4	2	3
2	114800	1740.0	3	2	1
3	94700	1980.0	3	2	3
4	119800	2130.0	3	3	3

In [42]:
```python
# correlation matrix
cor_mat = df_numeric.corr()
cor_mat
```

Out[42]:

	Price	SqFt	Bedrooms	Bathrooms	Offers
Price	1.000000	0.512462	0.499146	0.501489	-0.386868
SqFt	0.512462	1.000000	0.436659	0.490578	0.293352
Bedrooms	0.499146	0.436659	1.000000	0.382643	0.064872
Bathrooms	0.501489	0.490578	0.382643	1.000000	0.114522
Offers	-0.386868	0.293352	0.064872	0.114522	1.000000

In [43]:
```python
# plot correlations on a heatmap

# figure size
plt.figure(figsize=(10,5))

# heatmap
sns.heatmap(cor_mat, cmap="YlGnBu", annot=True) #YlGnBu
plt.show()
```

In [44]:
```python
#-------- END OF EDA -------------------
```

In [45]:
```python
#print column names
df.columns
```

Out[45]:
```
Index(['Home', 'Price', 'SqFt', 'Bedrooms', 'Bathrooms', 'Offers', 'Brick',
       'Neighborhood'],
      dtype='object')
```

In [46]:
```python
df.dtypes
```

Out[46]:
```
Home              int64
Price             int64
SqFt            float64
Bedrooms          int64
Bathrooms         int64
Offers            int64
Brick            object
Neighborhood     object
dtype: object
```

In [47]:
```python
#transform Bedrooms and bathrooms to object
df['Bedrooms']=df['Bedrooms'].replace([1,2,3,4,5],['1BR','2BR','3BR','4BR','5BR'])
df['Bathrooms']=df['Bathrooms'].replace([2,3,4],['2Bath','3Bath','4Bath'])
```

In [48]:
```python
df.head()
```

Out[48]:

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790.0	2BR	2Bath	2	No	East
1	2	114200	2030.0	4BR	2Bath	3	No	East
2	3	114800	1740.0	3BR	2Bath	1	No	East
3	4	94700	1980.0	3BR	2Bath	3	No	East
4	5	119800	2130.0	3BR	3Bath	3	No	East

In [49]:
```python
df.Bathrooms.unique()
```

Out[49]:
```
array(['2Bath', '3Bath', '4Bath'], dtype=object)
```

In [50]:
```python
# create dummy variables for categorical variables

# subset all categorical variables
df_categorical = df.select_dtypes(include=['object'])
df_categorical.head()
```

Out[50]:

	Bedrooms	Bathrooms	Brick	Neighborhood
0	2BR	2Bath	No	East
1	4BR	2Bath	No	East
2	3BR	2Bath	No	East
3	3BR	2Bath	No	East
4	3BR	3Bath	No	East

In [51]:
```python
# convert into dummies
df_dummies = pd.get_dummies(categoricat_col)
df_dummies
```

Out[51]:

	Bedrooms_2BR	Bedrooms_3BR	Bedrooms_4BR	Bedrooms_5BR	Bathrooms_2Bath	Bathrooms_3Bath	Bathrooms_4Bath	Brick_No	Brick_Yes	Neighborhood_East	Neighborhood_North	Neighborhood_West
0	1	0	0	0	1	0	0	1	0	1	0	0
1	0	0	1	0	1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	0	1	0	1	0	0
3	0	1	0	0	1	0	0	1	0	1	0	0
4	0	1	0	0	0	1	0	1	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
122	1	0	0	0	1	0	0	0	1	1	0	0
123	0	1	0	0	0	1	0	0	1	1	0	0
125	1	0	0	0	1	0	0	1	0	0	1	0
126	0	1	0	0	0	1	0	1	0	0	0	1
127	0	1	0	0	0	1	0	1	0	0	1	0

122 rows × 12 columns

In [52]:
```python
#Create master data by combining only numeric columns and dummies
df_numeric = df_numeric.drop(['Bedrooms', 'Bathrooms', 'Offers'], axis=1)
df_numeric.head()
```

Out[52]:

	Price	SqFt
0	114300	1790.0
1	114200	2030.0
2	114800	1740.0
3	94700	1980.0
4	119800	2130.0

In [53]:
```python
#Combine data
master_df=pd.concat([df_numeric,df_dummies], axis=1)
```

In [54]:
```python
#export master df to excel to validate the data
#master_df.to_excel(r'C:\Users\Sanjay Lohar\Desktop\prepared_data.xlsx')
```

In [55]:
```python
#Create x and y
y=master_df['Price']

x=master_df.drop('Price', axis=1)
```

In [56]:
```python
#Split x and y into training and test samples
from sklearn.model_selection import train_test_split
```

In [57]:
```python
xtrain, xtest, ytrain,ytest= train_test_split(x,y, test_size=0.3,random_state=0)
```

In [58]:
```python
print(xtrain.shape,ytrain.shape, xtest.shape,ytest.shape)
```

```
(85, 13) (85,) (37, 13) (37,)
```

In [ ]:

Linear regression

In [59]:
```python
#Buil Linear Regression model
from sklearn.linear_model import LinearRegression

#create an instance of Linear Regression
model=LinearRegression()

#fit the model using training sample
model.fit(xtrain,ytrain)
```

Out[59]:
```
LinearRegression()
```

In [60]:
```python
#check accuracy of training model
model.score(xtrain,ytrain)
```

Out[60]:
```
0.8363909621740817
```

In [61]:
```python
#predict house price
y_pred=model.predict(xtest)
```

In [62]:
```python
#check accuracy of test model
model.score(xtest,ytest)
```

Out[62]:
```
0.7037367460033985
```

In [63]:
```python
# print coefficients and intercept
print(model.coef_) #B1, B2, B3,....Bn
print(model.intercept_) #B0
```

```
[    38.27609761   -536.97256168   -822.6711758    8944.00051005
  -7584.35677257 -12489.83835604 -10093.65428241  22583.49263845
 -10669.78390424  10669.78390424  -8228.66813913 -10382.75582474
  18611.42396387]
67567.39272921086
```

In [64]:
```python
y_pred
```

Out[64]:
```
array([157891.33865404, 110137.29967318, 134396.47711957, 115835.85436085,
       125885.66880607, 112053.66040863, 124204.40893494, 154639.56578979,
       159332.31572027, 117366.89826542, 110422.99828729, 124454.24311827,
       110040.23731115, 115113.19236304, 101956.84100292, 142916.83681149,
       129998.01047815, 133017.46276467, 121719.74717424, 128700.47828663,
       141045.2843425 , 114730.43138689,  95112.55924156, 111767.96179452,
       165546.55817691, 106692.45088789, 139231.7052213 , 182968.44938562,
       106212.62754971, 100425.79709835, 160097.83767256, 111145.6602851 ,
       148800.12208214, 160187.9045109 , 118177.83602692, 111143.10443036,
       109471.39593766])
```

In [ ]:

In [ ]:

In [ ]: