In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

Data preparation

In [3]:

```
df=pd.read_csv(r"C:\Users\Sonu\OneDrive\Desktop\NIT\29th- REGRESSION PROJECT\29th-
```

Explore the data

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object 
 12  year          18249 non-null  int64  
 13  region        18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

In [5]:

```
df.head()
```

Out[5]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26

Missing value checking

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

Droping uncessary columns

In [7]:

```
df = df.drop(['Unnamed: 0','4046','4225','4770','Date'],axis=1)
```

In [8]:

```
df.head()
```

Out[8]:

	AveragePrice	Total Volume	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

Answering questions

In [9]:

```python
def get_avarage(df,column):
    """
    Description: This function to return the average value of the column

    Arguments:
        df: the DataFrame.
        column: the selected column.
    Returns:
        column's average
    """
    return sum(df[column])/len(df)
```

In [10]:

```python
def get_avarge_between_two_columns(df,column1,column2):
    """
    Description: This function calculate the average between two columns in the dat

    Arguments:
        df: the DataFrame.
        column1:the first column.
        column2:the scond column.
    Returns:
        Sorted data for relation between column1 and column2
    """

    List=list(df[column1].unique())
    average=[]

    for i in List:
        x=df[df[column1]==i]
        column1_average= get_avarage(x,column2)
        average.append(column1_average)

    df_column1_column2=pd.DataFrame({'column1':List,'column2':average})
    column1_column2_sorted_index=df_column1_column2.column2.sort_values(ascending=F
    column1_column2_sorted_data=df_column1_column2.reindex(column1_column2_sorted_i

    return column1_column2_sorted_data
```

In [11]:

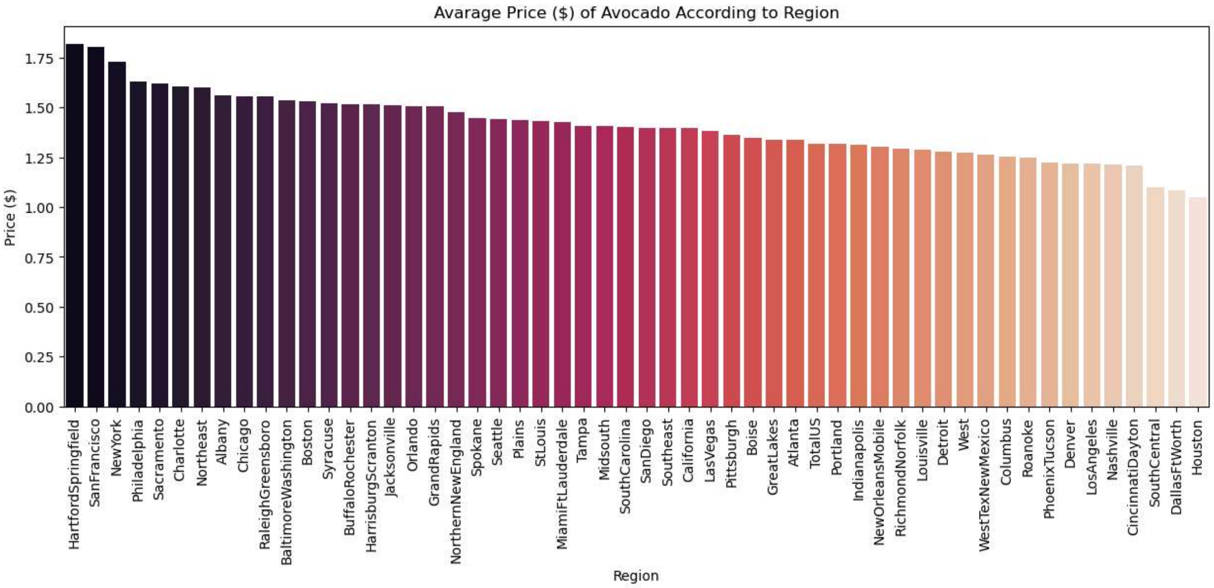
```python
def plot(data,xlabel,ylabel):
    """
    Description: This function to draw a barplot

    Arguments:
        data: the DataFrame.
        xlabel: the label of the first column.
        ylabel: the label of the second column.
    Returns:
        None
    """

    plt.figure(figsize=(15,5))
    ax=sns.barplot(x=data.column1,y=data.column2,palette='rocket')
    plt.xticks(rotation=90)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(('Avarage '+ylabel+' of Avocado According to '+xlabel));
```

Which regression are the lowest and highest prices of Avocado?

In [12]:
```
data1 = get_avarge_between_two_columns(df,'region','AveragePrice')
plot(data1,'Region','Price ($)')
```



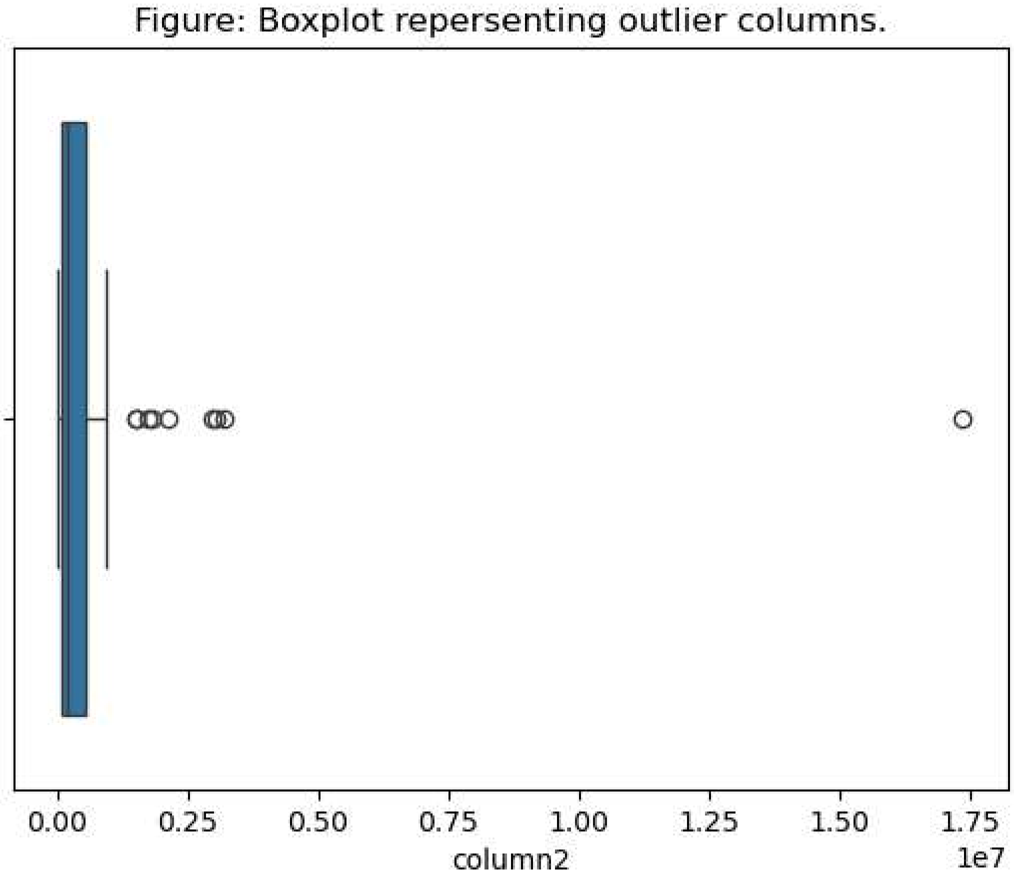
In [13]:
```
print(data1['column1'].iloc[-1], " is the region producing avocado with the lowest
```

```
Houston  is the region producing avocado with the lowest price.
```

What is the highest region of avocado production?

In [14]:
```
data2 = get_avarge_between_two_columns(df,'region','Total Volume')
sns.boxplot(x=data2.column2).set_title("Figure: Boxplot repersenting outlier column
```

Out[14]:
```
Text(0.5, 1.0, 'Figure: Boxplot repersenting outlier columns.')
```



In [15]:

```
outlier_region = data2[data2.column2>10000000]
print(outlier_region['column1'].iloc[-1], "is outlier value")
```

```
TotalUS is outlier value
```

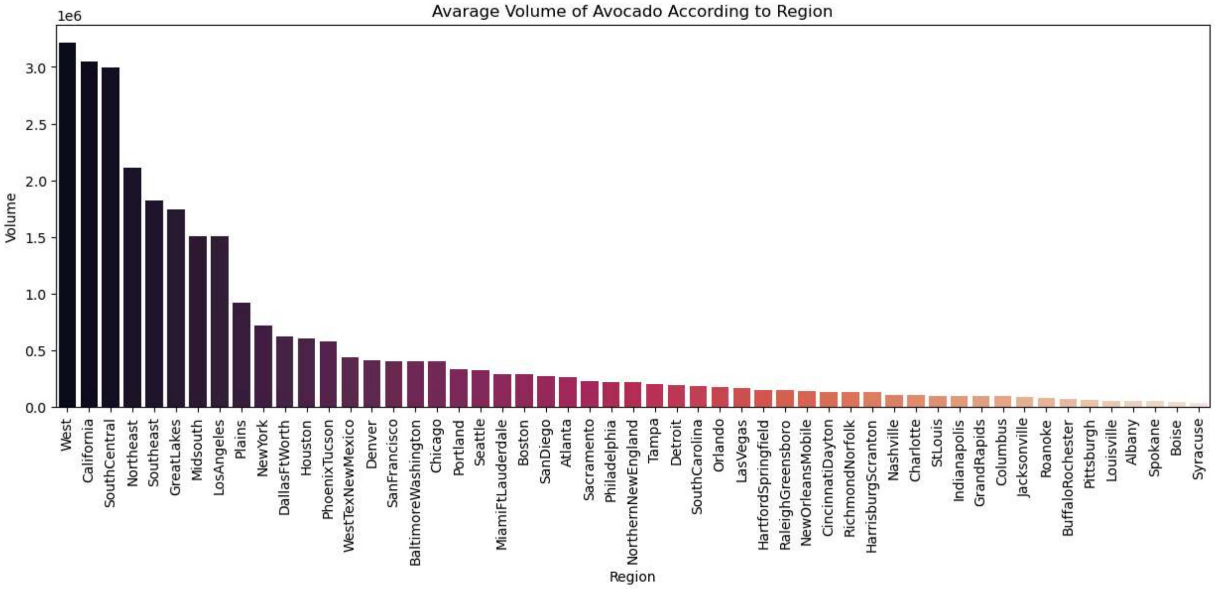
Remove the outlier values

In [16]:

```
outlier_region.index
data2 = data2.drop(outlier_region.index,axis=0)
```

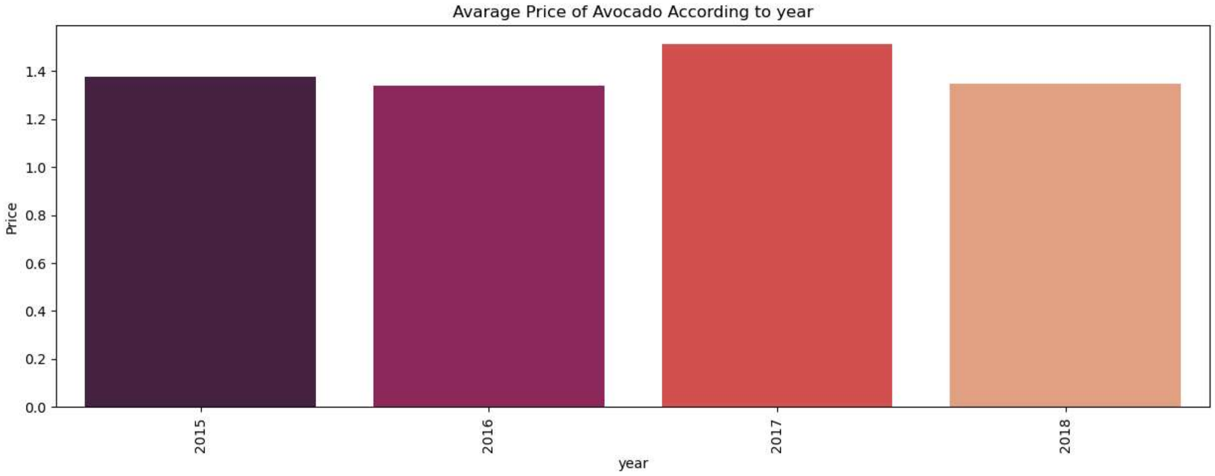
In [17]:

```
plot(data2,'Region','Volume')
```



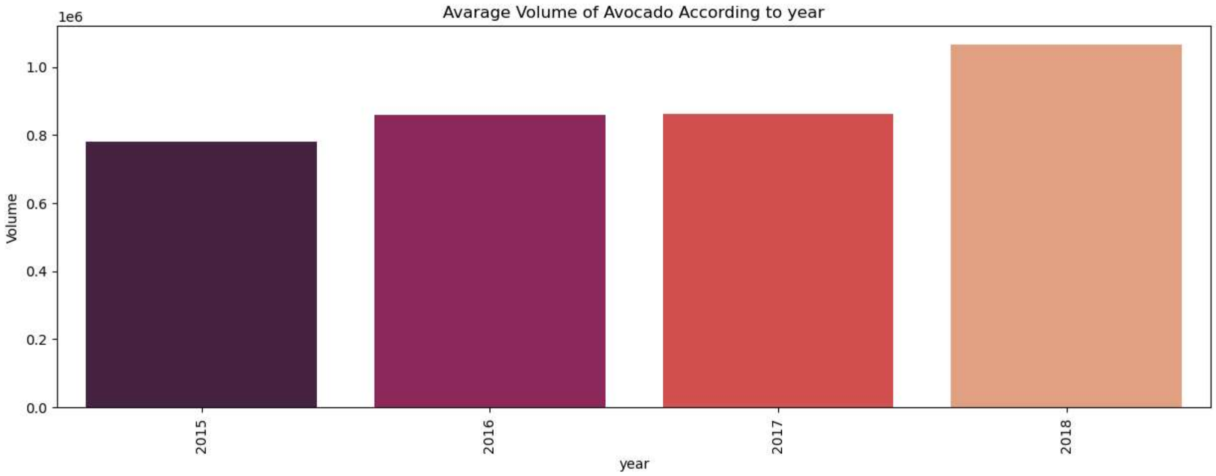
What is the average avocado prices in each year?

In [18]:
```
data3 = get_avarge_between_two_columns(df,'year','AveragePrice')
plot(data3,'year','Price')
```



What is the average avocado volume in each year?

In [19]:
```
data4 = get_avarge_between_two_columns(df,'year','Total Volume')
plot(data4,'year','Volume')
```



Data Modeling

In [20]:

```
df['region'] = df['region'].astype('category')
df['region'] = df['region'].cat.codes

df['type'] = df['type'].astype('category')
df['type'] = df['type'].cat.codes
```

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   AveragePrice  18249 non-null  float64
 1   Total Volume  18249 non-null  float64
 2   Total Bags    18249 non-null  float64
 3   Small Bags    18249 non-null  float64
 4   Large Bags    18249 non-null  float64
 5   XLarge Bags   18249 non-null  float64
 6   type          18249 non-null  int8   
 7   year          18249 non-null  int64  
 8   region        18249 non-null  int8   
dtypes: float64(6), int64(1), int8(2)
memory usage: 1.0 MB
```

In [22]:

```
df.head()
```

Out[22]:

	AveragePrice	Total Volume	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	0	2015	0
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	0	2015	0
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	0	2015	0
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	0	2015	0
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	0	2015	0

In [23]:

```
# split data into X and y
X = df.drop(['AveragePrice'],axis=1)
y = df['AveragePrice']

# split data into traing and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=15)
```

In [24]:

```
print("training set:",X_train.shape,' - ',y_train.shape[0],' samples')
print("testing set:",X_test.shape,' - ',y_test.shape[0],' samples')
```

```
training set: (12774, 8)  -  12774  samples
testing set: (5475, 8)  -  5475  samples
```

In [27]:

```
model=LinearRegression(normalize=True)
model.fit(X_train,y_train)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 model=LinearRegression(normalize=True)
      2 model.fit(X_train,y_train)

TypeError: LinearRegression.__init__() got an unexpected keyword argument 'normaliz
e'
```

Evaluate the results

In [29]:

```
# prediction and calculate the accuracy for the testing dataset
test_pre = model.predict(X_test)
test_score = r2_score(y_test,test_pre)
print("The accuracy of testing dataset ",test_score*100)
```

```
The accuracy of testing dataset  38.58074176452857
```

In [30]:

```
# prediction and calculate the accuracy for the testing dataset
train_pre = model.predict(X_train)
train_score = r2_score(y_train,train_pre)
print("The accuracy of training dataset ",train_score*100)
```

```
The accuracy of training dataset  39.7068604241106
```

```
In [ ]:
```