Avocado Data Analysis

Business Understanding

The aim of this project is to answer the following four questions: 1. Which region are the lowest and highest prices of Avocado? 2. What is the highest region of avocado production? 3. What is the average avocado prices in each year? 4. What is the average avocado volume in each year?

Data Understanding

The Avocado dataset was been used in this project.

This dataset contains 13 columns: 1.Date: The date of the observation 2.AveragePrice: The average price of a single avocado 3.Total Volume: Total number of avocados sold 4.Total Bags: Total number of bags 5.Small Bags: Total number of small bags 6.Large Bags: Total number of Large bags 7.XLarge Bags: Total number of XLarge bags 8.type: conventional or organic 9.year: year 10.region: The city or region of the observation 11.4046: Total number of avocados with PLU 4046 sold 12.4225: Total number of avocados with PLU 4225 sold 13.4770: Total number of avocado with PLU 4770 sold

Import necessary libraies

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
```

Data preparation

Load data

In [2]:

```
data = pd.read_csv(r"C:\Users\ankus\Desktop\NareshIT\2. Notes\11.Machine learnin
data
```

Out[2]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95
...	...	...	...	...	...	...	...	...
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15

18249 rows × 14 columns

Explore the data

In [3]:

```
data.head()
```

Out[3]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Sr B
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	860
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	940
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	804
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	567
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	598

In [4]:

```
data.tail()
```

Out[4]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	1
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	1
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	1

In [5]:

```
data.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 [6]:

```
data.describe()
```

Out[6]:

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439

Missing Value checking

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
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 unnecessary columns

In [8]:

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

In [9]:

```python
data.head()
```

Out[9]:

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

Answering questions

In [10]:

```python
# This function to return the average value of the column
def get_average(data,column):
   return sum(data[column])/len(data)
```

In [14]:

```python
# This function calculate the average between two columns in the dataset
def get_average_between_two_columns(data,column1,column2):
  # return sorted data for relation between column1 and column2
    List = list(data[column1].unique())
    average=[]

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

    data_column1_column2 = pd.DataFrame({'column1':List,'column2':average})
    column1_column2_sorted_index = data_column1_column2.column2.sort_values(asce
    column1_column2_sorted_data =data_column1_column2.reindex(column1_column2_so
    return column1_column2_sorted_data
```

In [16]:
```
#this function to draw a barplot
def plot(data,xlabel,ylabel):
    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(('Average '+ylabel+' of Avocado According to '+xlabel))
```

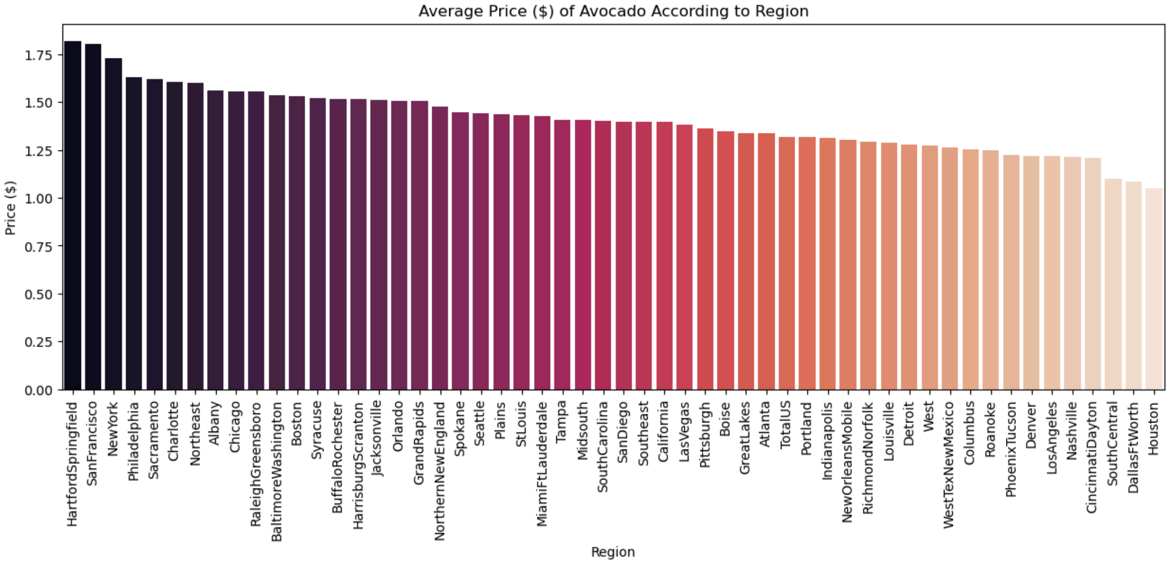
Which region are the lowest and highest prices of Avocado?

In [23]:
```
data1 = get_average_between_two_columns(data,'region','AveragePrice')
plot(data1,'Region','Price ($)')
```

```
C:\Users\ankus\AppData\Local\Temp\ipykernel_28772\988956694.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=data.column1,y=data.column2,palette='rocket')
```



In [25]:
```
print(data1['column1'].iloc[-1], "is the region producing avocado with the lowes
```

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

What is the highest region of avocado production?

Checking if there are outlier values or not

In [26]:
```
data2 = get_average_between_two_columns(data,'region','Total Volume')
sns.boxplot(x=data2.column2).set_title("Figure: Boxplot representating outlier c
```

Out[26]:
```
Text(0.5, 1.0, 'Figure: Boxplot representating outlier columns.')
```

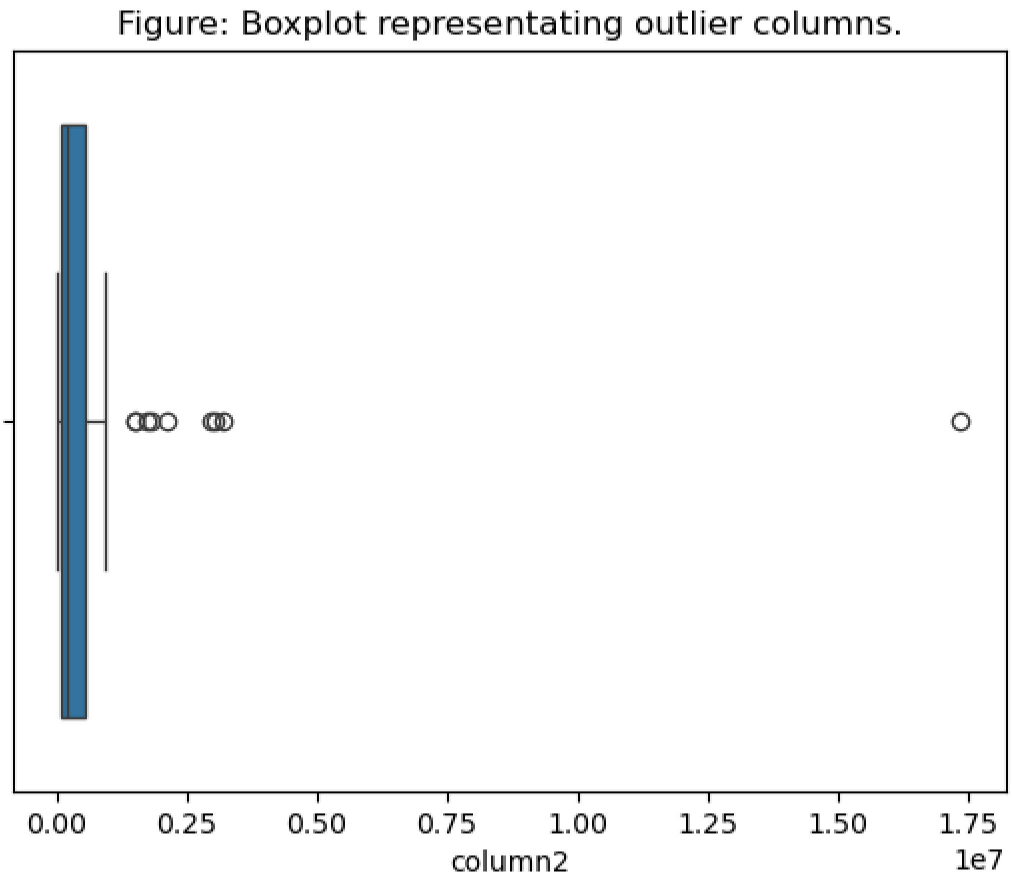


Figure: Boxplot representating outlier columns.

```
In [28]: outlier_region = data2[data2.column2>10000000]
         print(outlier_region['column1'].iloc[-1],"is outlier value")
```

```
TotalUS is outlier value
```

Remove the outlier values

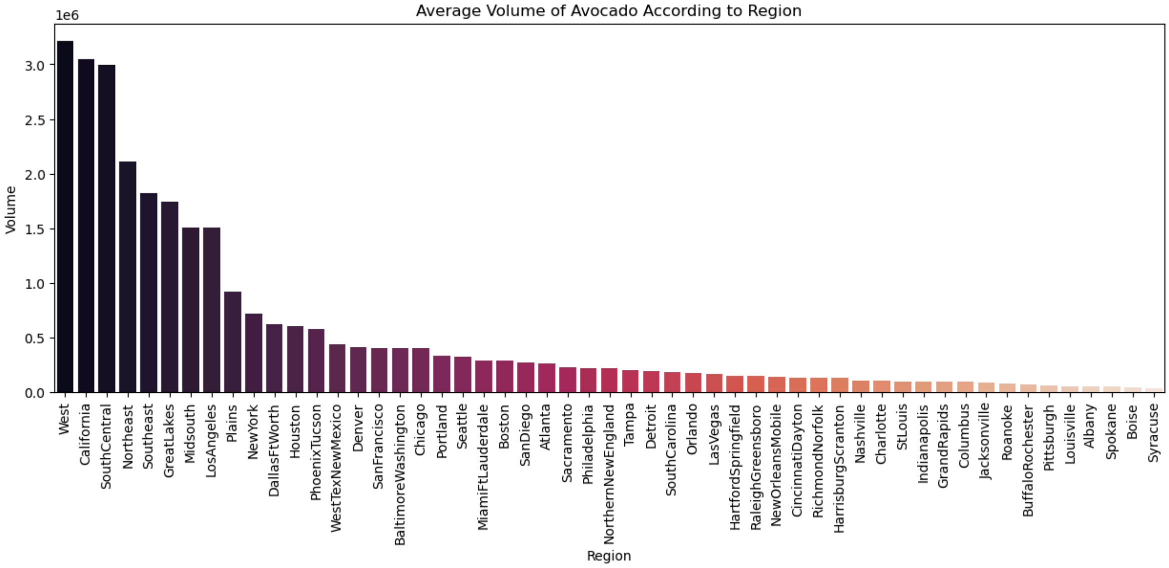
```
In [29]: outlier_region.index
         data2 = data2.drop(outlier_region.index,axis=0)
```

```
In [31]: plot(data2,'Region','Volume')
```

```
C:\Users\ankus\AppData\Local\Temp\ipykernel_28772\988956694.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=data.column1,y=data.column2,palette='rocket')
```



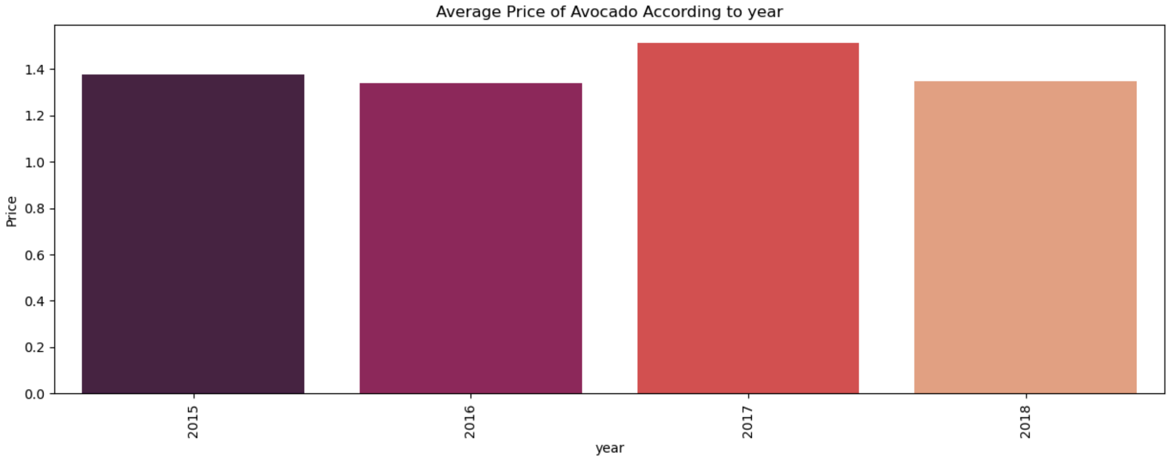
What is the average avocado prices in each year?

In [32]:
```
data3 = get_average_between_two_columns(data,'year','AveragePrice')
plot(data3,'year','Price')
```

```
C:\Users\ankus\AppData\Local\Temp\ipykernel_28772\988956694.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=data.column1,y=data.column2,palette='rocket')
```



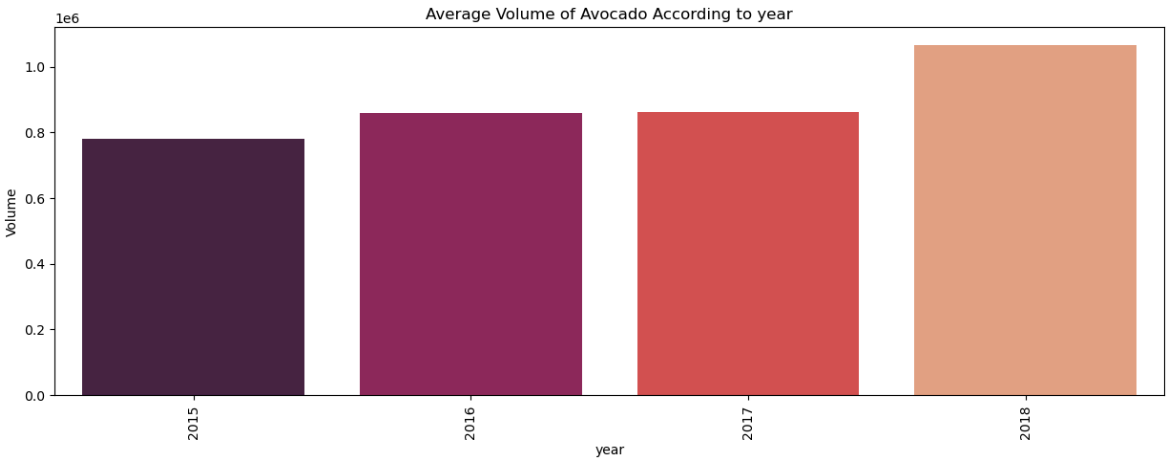
What is the average avocado volume in each year?

In [35]:
```
data4 = get_average_between_two_columns(data,'year','Total Volume')
plot(data4,'year','Volume')
```

```
C:\Users\ankus\AppData\Local\Temp\ipykernel_28772\988956694.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=data.column1,y=data.column2,palette='rocket')
```



Data Modeling

We built the regression model by used Linear regression from sklearn to predict the avocado price

Changing some column types to categories

In [36]:
```
data['region'] = data['region'].astype('category')
data['region'] = data['region'].cat.codes

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

In [37]:
```
data.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 [38]:
```
data.head()
```

Out[38]:

	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 [40]:

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

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

In [41]:

```
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 [45]:

```
# built and fit the model
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[45]:

```
▾ LinearRegression
LinearRegression()
```

Evaluate the results

In [46]:

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

```
The accuracy of testing dataset  38.580741764418356
```

In [47]:

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

```
The accuracy of training dataset  39.70686042410679
```

The model doesn't work well with this dataset, In order to the avocado prices were near together