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

In [2]:
```
# upload dataset
df = pd.read_csv("F:\\archive (11)\\avocado.csv")
```

In [3]:
```
# see the top 5 rows
df.head(5)
```

Out[3]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLa B
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	

In [4]:
```
# see the last 5 rows
df.tail(5)
```

Out[4]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01

In [5]:
```
# see the shape
```

```
df.shape
```

Out[5]: (18249, 14)

In [6]:
```
# see the size
df.size
```

Out[6]: 255486

In [7]:
```
# get the information of data
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 [8]:
```
# see the columns
df.columns
```

Out[8]:
```
Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',
       '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',
       'year', 'region'],
      dtype='object')
```

In [9]:
```
# get the statistical summary
df.describe()
```

Out[9]:

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Ba
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+(
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+(
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+(
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+(
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+(
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+(
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+(

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Ba
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+(

In [10]:
```
# see the data types
df.dtypes
```

Out[10]:
```
Unnamed: 0        int64
Date             object
AveragePrice    float64
Total Volume    float64
4046            float64
4225            float64
4770            float64
Total Bags      float64
Small Bags      float64
Large Bags      float64
XLarge Bags     float64
type             object
year              int64
region           object
dtype: object
```

In [11]:
```
#dfs = df.copy()
#df = df.drop(["Unnamed: 0","Date"],axis=1)
#df
```

In [12]:
```
# drop the column which is named by "Unnamed"
df.drop("Unnamed: 0", axis= 1, inplace= True)
```

In [13]:
```
# see the data
df
```

Out[13]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0
...	...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0

18249 rows × 13 columns

In [14]:
```
# see the duplicate values if any, there is no duplicate values
df.duplicated().sum().any()
```

Out[14]: False

In [15]:
```
# again check the shape , shape is same as above
df.shape
```

Out[15]: (18249, 13)

In [16]:
```
#converting date column to date format and extracting the month
df['Date']= pd.to_datetime(df['Date'])
df['month']= df['Date'].dt.month
```

In [17]:
```
df
```

Out[17]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0
...	...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0

18249 rows × 14 columns

In [18]:
```
df.info() # perfectly converted to date time format with last column of month
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   Date          18249 non-null  datetime64[ns]
 1   AveragePrice  18249 non-null  float64       
 2   Total Volume  18249 non-null  float64       
 3   4046          18249 non-null  float64       
 4   4225          18249 non-null  float64       
 5   4770          18249 non-null  float64       
 6   Total Bags    18249 non-null  float64       
 7   Small Bags    18249 non-null  float64       
 8   Large Bags    18249 non-null  float64       
 9   XLarge Bags   18249 non-null  float64       
 10  type          18249 non-null  object        
 11  year          18249 non-null  int64         
 12  region        18249 non-null  object        
 13  month         18249 non-null  int64         
dtypes: datetime64[ns](1), float64(9), int64(2), object(2)
memory usage: 1.9+ MB
```

In [19]:
```
# see the unique values
df.nunique()
```

Out[19]:
```
Date             169
AveragePrice     259
Total Volume   18237
4046           17702
4225           18103
4770           12071
Total Bags     18097
Small Bags     17321
Large Bags     15082
XLarge Bags     5588
type               2
year               4
region            54
month             12
dtype: int64
```

In [20]:
```
# check the null values if any , there is no null values , so treatment required
df.isnull().sum() # there is no null values, no treatment requirement
```

Out[20]:
```
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
month           0
dtype: int64
```

In [21]:

```
# change the name of some columns
df.rename(columns={"4046": "new col1"}, inplace=True)
df.rename(columns={"4225": "new col2"}, inplace=True)
df.rename(columns={"4770": "new col3"}, inplace=True)
df
```

Out[21]:

	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0
...	...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0

18249 rows × 14 columns

In [22]:

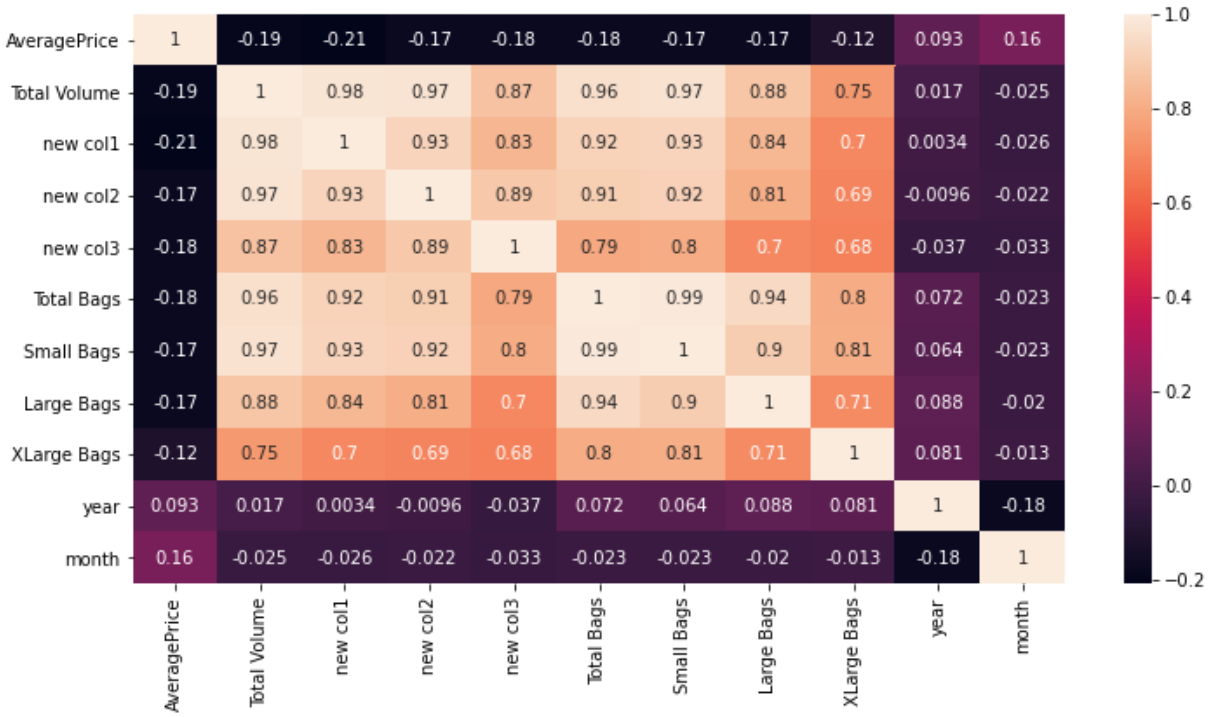
```
# see the unique values in region
df.region.nunique()
```

Out[22]: 54

In [23]:
```
# find the correlation between variables , there is strong between small bags and total bags
corr=df.corr()
```

In [128]:
```
# get a heat map for relation between variables
plt.figure(figsize=(12,6))
sns.heatmap(corr, annot= True)
```

Out[128]: <AxesSubplot:>



In [25]:
```
#df_i3= df.iloc[3]

#df_i18240 = df.iloc[18240]

#pd.concat([df_i3, df_i18240], axis= 1, ignore_index = True)
```

In [129]:
```
# get the value counts of type
df['type'].value_counts()
```

Out[129]:
```
conventional    9126
organic         9123
Name: type, dtype: int64
```

In [27]:
```
# see the value counts of year
df['year'].value_counts()
```

Out[27]:
```
2017    5722
2016    5616
2015    5615
```

```
2018    1296
Name: year, dtype: int64
```

In [28]:
```
# see the value counts of region
df['region'].value_counts()
```

Out[28]:
```
SouthCarolina          338
Denver                 338
NewYork                338
Albany                 338
DallasFtWorth          338
Portland               338
CincinnatiDayton       338
Plains                 338
TotalUS                338
RichmondNorfolk        338
Atlanta                338
BuffaloRochester       338
SouthCentral           338
SanFrancisco           338
Philadelphia           338
West                   338
Syracuse               338
Columbus               338
Pittsburgh             338
SanDiego               338
Jacksonville           338
Houston                338
PhoenixTucson          338
Boise                  338
LasVegas               338
Tampa                  338
Chicago                338
Charlotte              338
StLouis                338
LosAngeles             338
Nashville              338
GreatLakes             338
Midsouth               338
HartfordSpringfield    338
Boston                 338
NewOrleansMobile       338
HarrisburgScranton     338
Louisville             338
California             338
MiamiFtLauderdale      338
Seattle                338
Northeast              338
Southeast              338
RaleighGreensboro      338
Sacramento             338
Detroit                338
Indianapolis           338
BaltimoreWashington    338
Roanoke                338
Orlando                338
NorthernNewEngland     338
GrandRapids            338
Spokane                338
WestTexNewMexico       335
Name: region, dtype: int64
```

In [29]:
```
# see the value counts of month variable
df['month'].value_counts()
```

Out[29]:
```
1     1944
3     1836
2     1728
5     1512
7     1512
10    1512
4     1404
8     1404
11    1404
12    1403
9     1296
6     1294
Name: month, dtype: int64
```

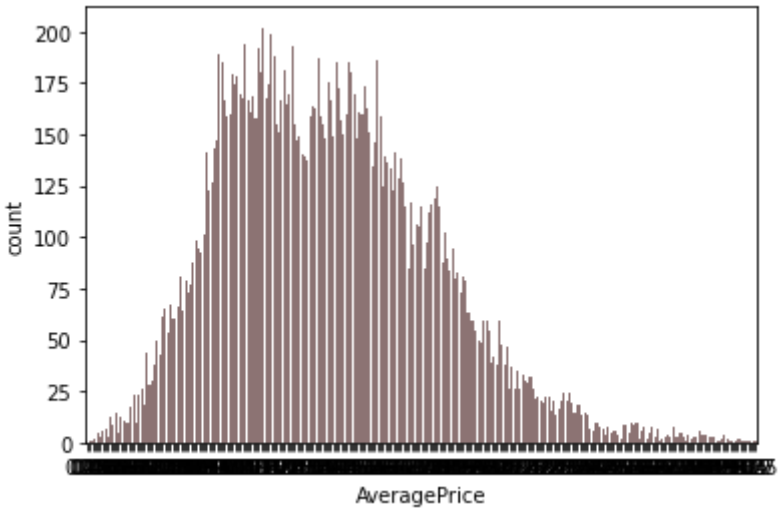
In [30]:
```
# again check the head of data
df.head(4)
```

Out[30]:

	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags	
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conven
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conven
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conven
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conven

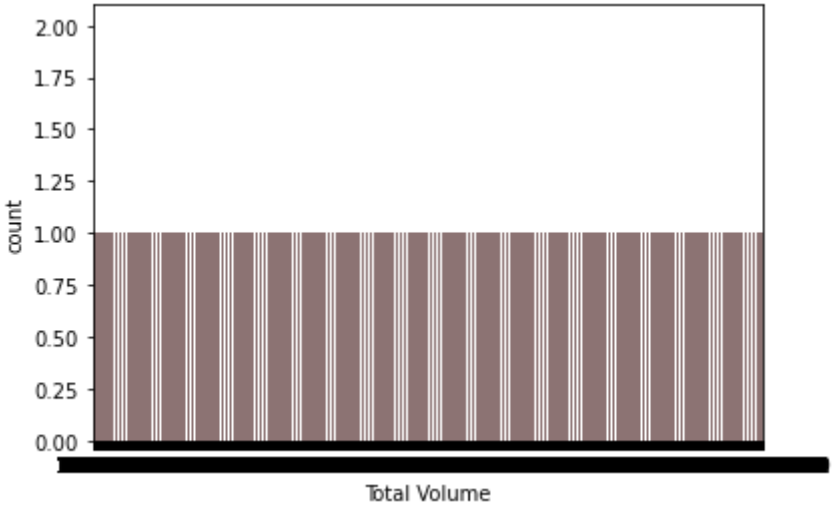
In [31]:
```
# count plot for average price of avocados on differ time and differ region
sns.countplot(x='AveragePrice',data=df, color='red', saturation=0.1)
```

Out[31]:
```
<AxesSubplot:xlabel='AveragePrice', ylabel='count'>
```



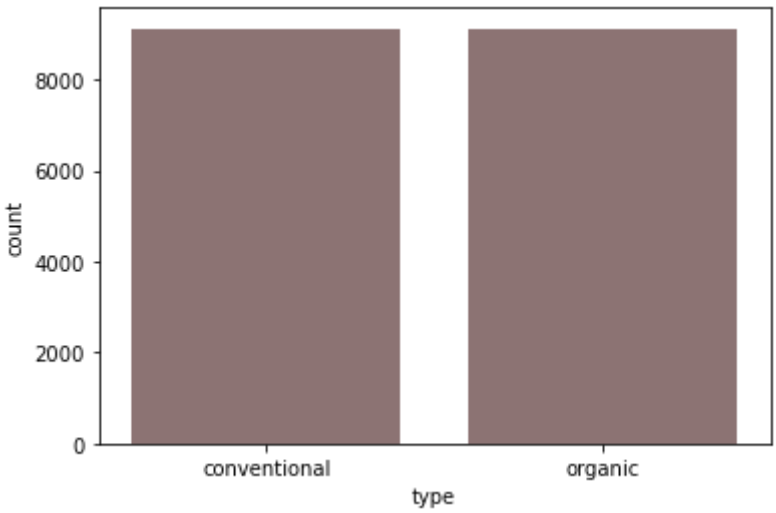
In [41]:
```
# make a count plot for total volume of avocados
sns.countplot(x='Total Volume',data=df, color='red', saturation=0.1)
```

Out[41]:
```
<AxesSubplot:xlabel='Total Volume', ylabel='count'>
```



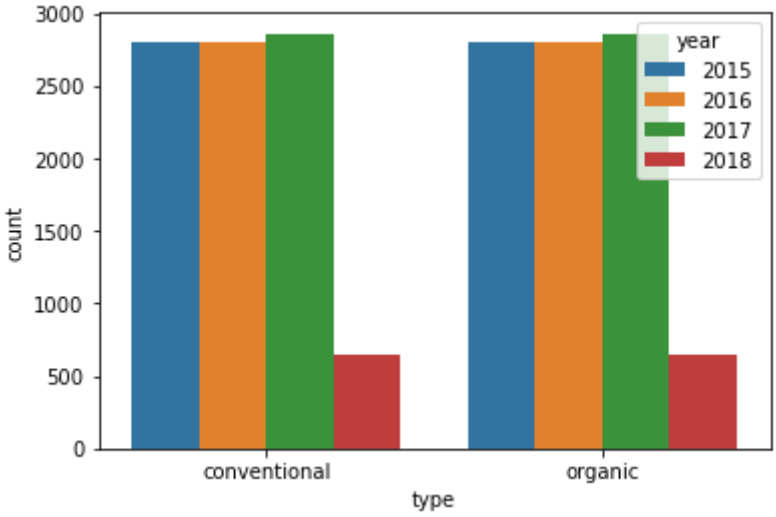
In [42]:
```
# make a count plot for type of avocados, there is conventional and
organic types of avocados
sns.countplot(x='type',data=df, color='red', saturation=0.1)
```

Out[42]: <AxesSubplot:xlabel='type', ylabel='count'>



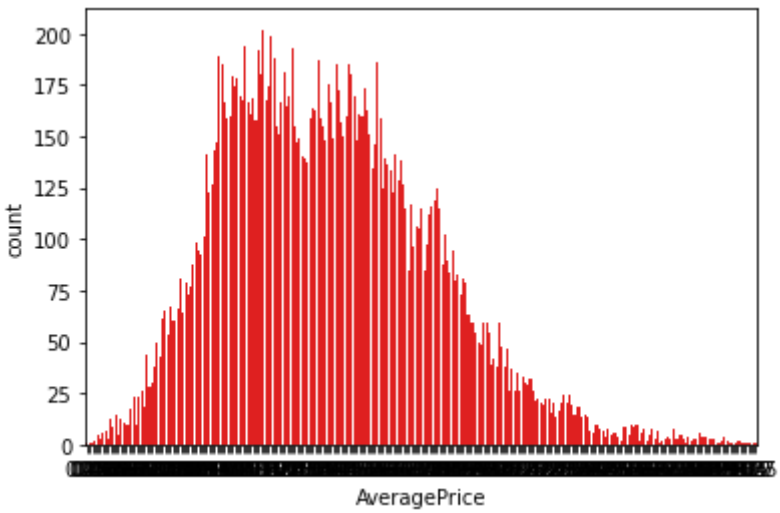
In [43]:
```
# make a countplot of type with year as a hue value , it shows both types
are equal in their corresponding years
sns.countplot(data= df, x= 'type', hue= "year")
```

Out[43]: <AxesSubplot:xlabel='type', ylabel='count'>



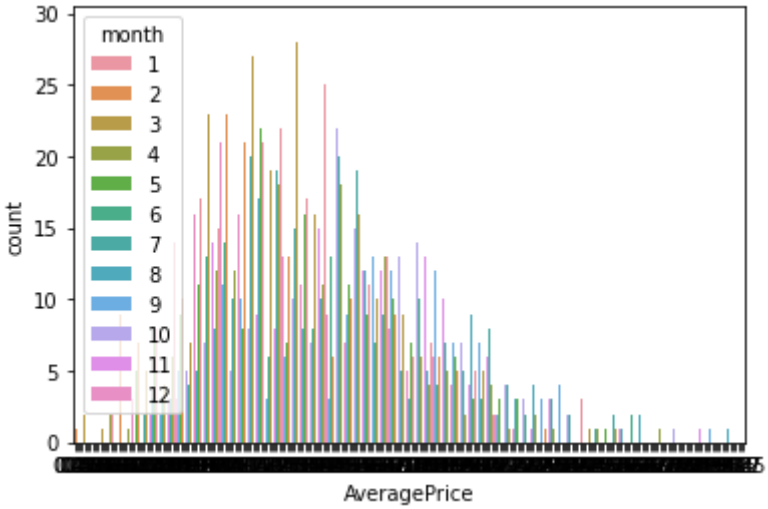
In [44]:
```
# make a countplot of average price, it shows average price of avocados
sns.countplot(x="AveragePrice", data= df, color= 'red')
```

Out[44]: <AxesSubplot:xlabel='AveragePrice', ylabel='count'>



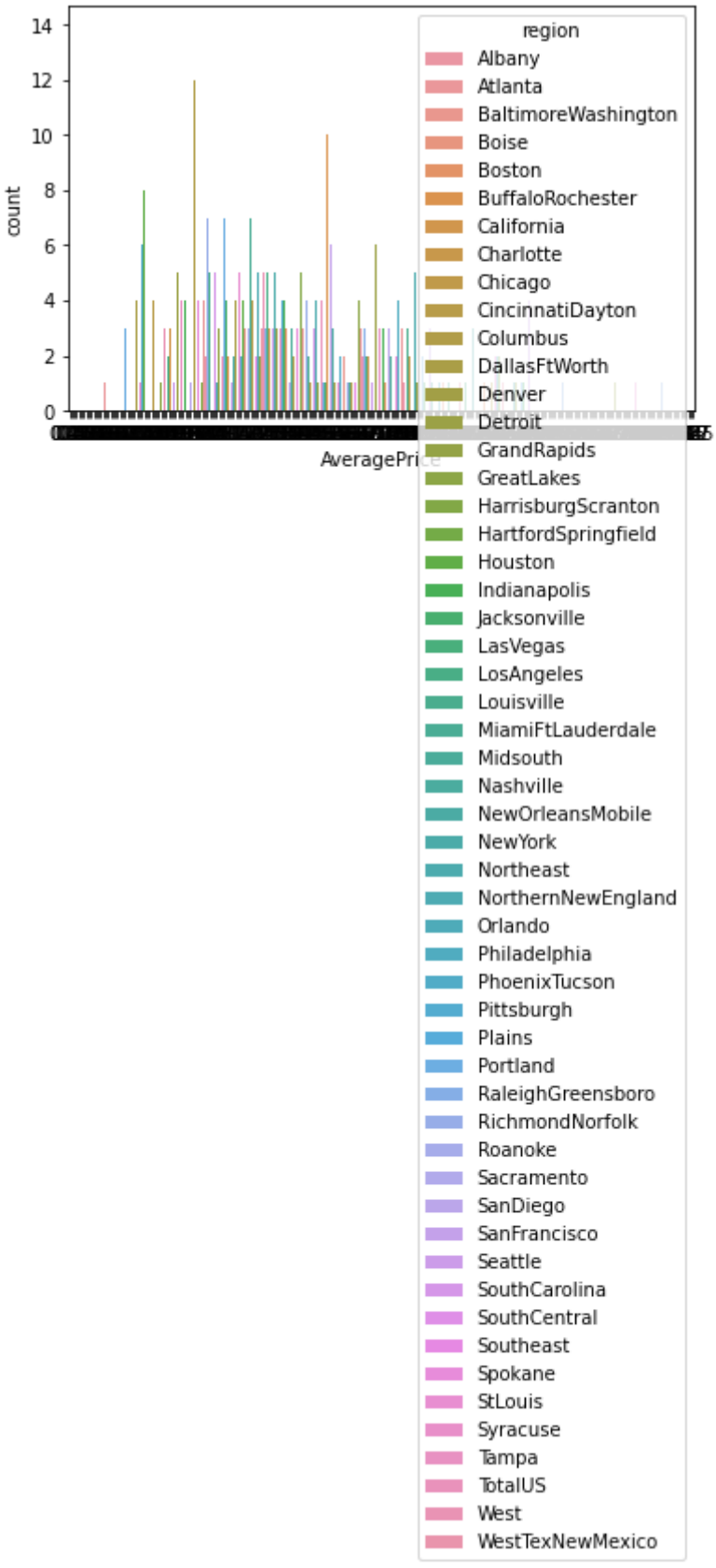
In [45]:
```
# make a count plot of average price with month as a hue, wherein ,how much avocados sale in 1 to 12 month
sns.countplot(data= df, x= 'AveragePrice', hue= "month")
```

Out[45]: <AxesSubplot:xlabel='AveragePrice', ylabel='count'>



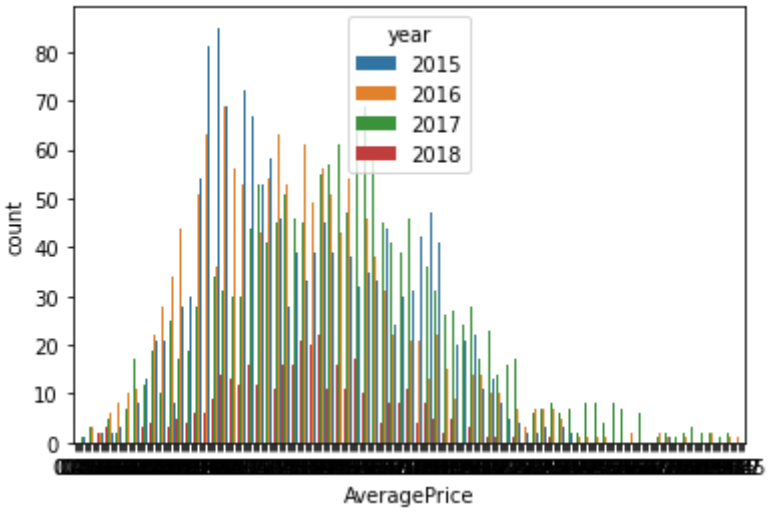
In [37]:
```
# make a countplot of average price and region as a hue value,which shows the average price of avocado
# as per the regions
sns.countplot(data= df, x= 'AveragePrice', hue= "region")
```

Out[37]: <AxesSubplot:xlabel='AveragePrice', ylabel='count'>



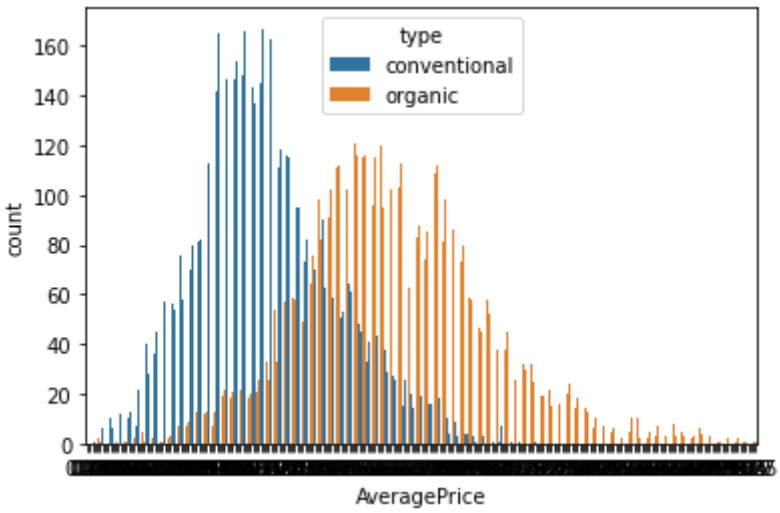
In [38]:
```
# make a countplot of average price with hue value is year, it shows
average price as per the year
sns.countplot(data= df, x= 'AveragePrice', hue= "year")
```

Out[38]: <AxesSubplot:xlabel='AveragePrice', ylabel='count'>



In [39]:
```
# this count shows avearge prices with differ types of avocados, where conventional the more than oragnic.
sns.countplot(data= df, x= 'AveragePrice', hue= "type")
```

Out[39]: <AxesSubplot:xlabel='AveragePrice', ylabel='count'>



In [ ]:
```
# BLANK CELL..
```

In [ ]:
```
# BLANK CELL...
```

In [ ]:
```
# BLANK CELL.....
```

In [40]:
```
df.head()
```

Out[40]:

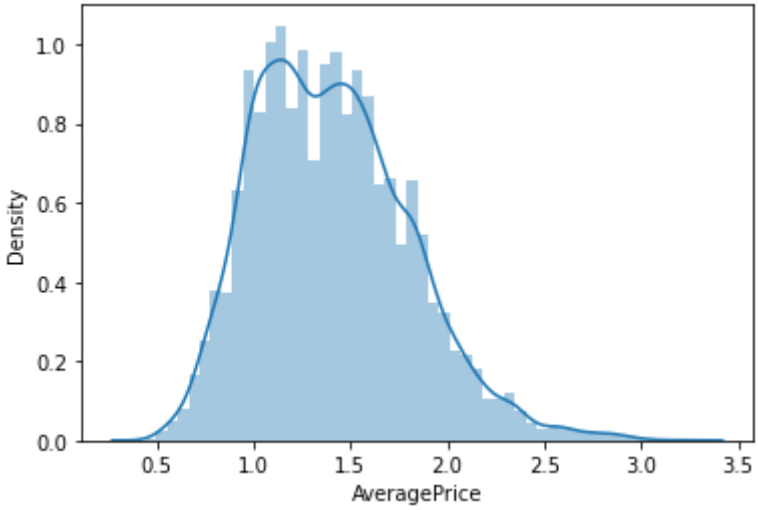
	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags	
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conven
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conven

	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags	
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conven
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conven
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conven

In [47]:
```
# lets compare the price distribution with types of avocados:
sns.distplot(df['AveragePrice'])
```

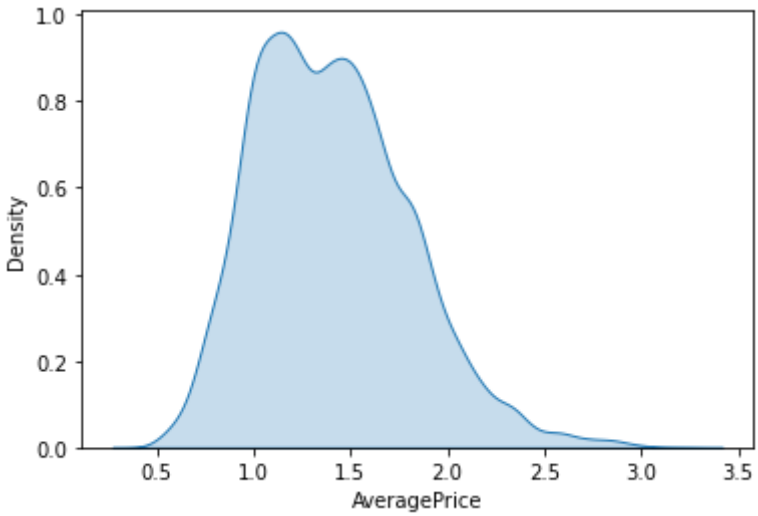
```
C:\Users\Simmy\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[47]: <AxesSubplot:xlabel='AveragePrice', ylabel='Density'>



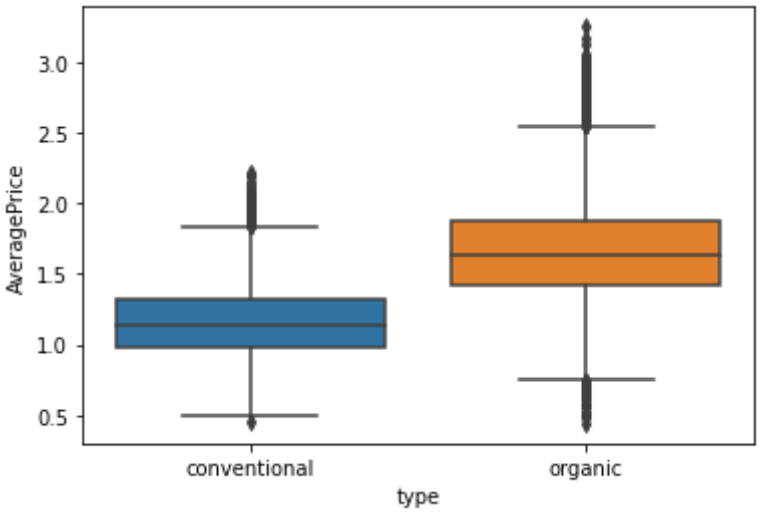
In [48]:
```
# kde plot doesnt mention graphs , only represented by a line.
sns.kdeplot(df['AveragePrice'], shade = True)
```

Out[48]: <AxesSubplot:xlabel='AveragePrice', ylabel='Density'>



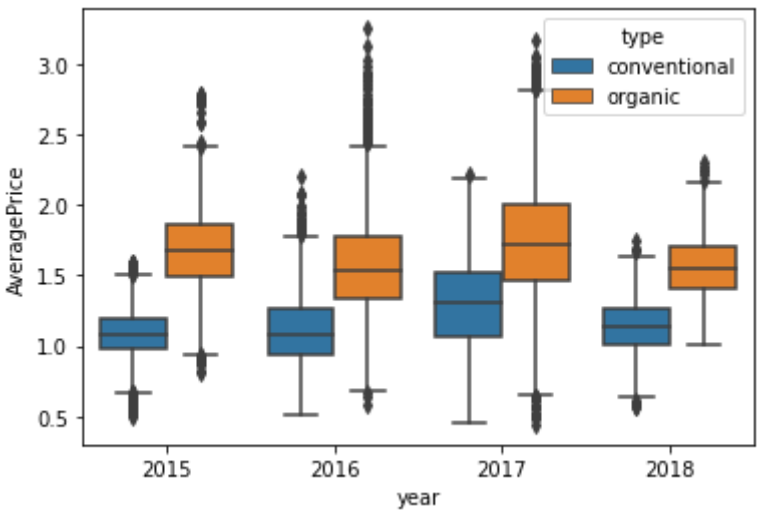
In [49]:
```
# make a boxplot of average price and type variables ,where this shows
organic prices are higher than
# conventional , organic are more costly than conventional avocado.
sns.boxplot(data= df, x= 'type', y =df['AveragePrice'])
```

Out[49]: <AxesSubplot:xlabel='type', ylabel='AveragePrice'>

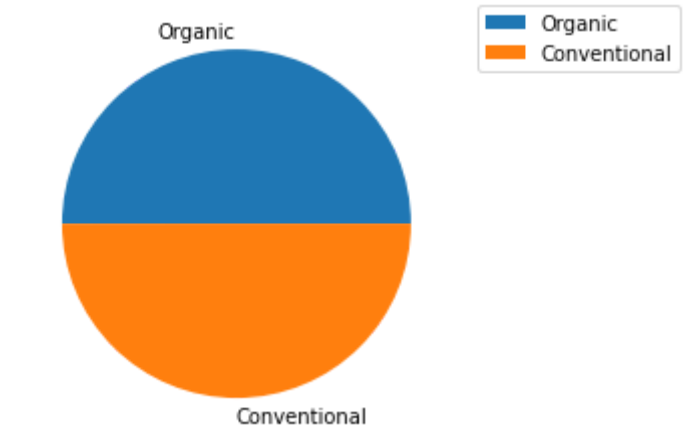


In [51]:
```
# make a boxplot of average price and year with type as hue, it shows
year 2017 is the higher average
# price than the other years for both avocado types
sns.boxplot(data= df, x= "year", y =df["AveragePrice"], hue= 'type')
# 2017 is the highest average prices for both avocados types.
```

Out[51]: <AxesSubplot:xlabel='year', ylabel='AveragePrice'>



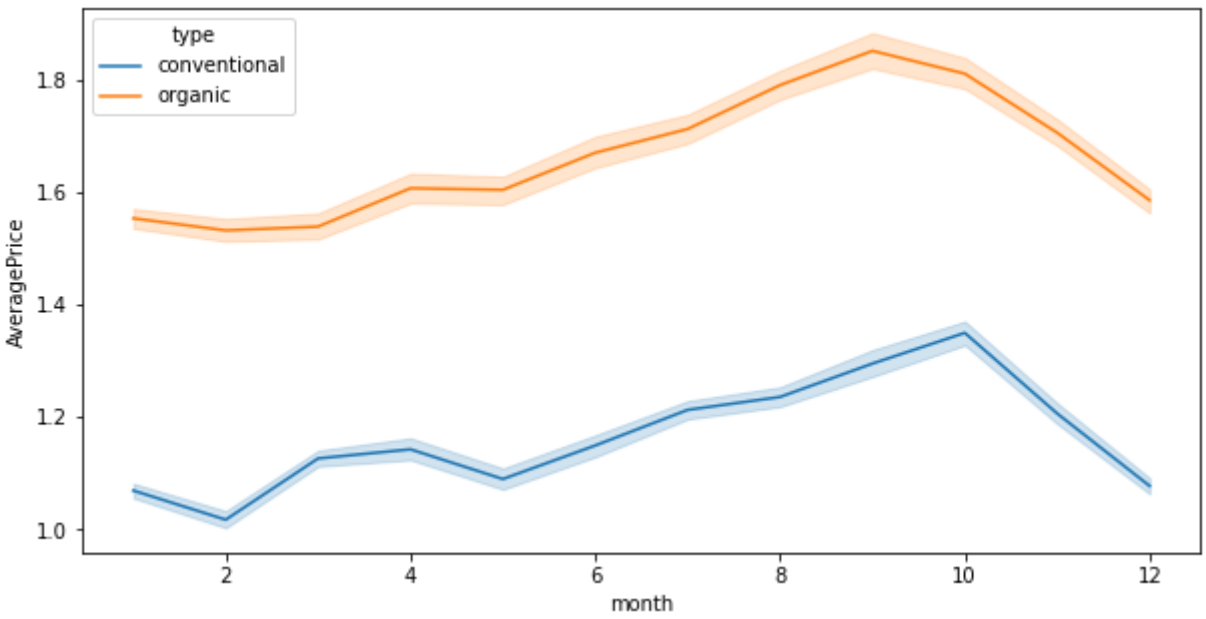
In [55]:
```
# make a pie chart for types of avocado divided
pie_chart=df['type'].value_counts()
slices= ['Organic', 'Conventional']
plt.pie(pie_chart, labels =slices)
plt.legend(title="avocado by type")
plt.legend(bbox_to_anchor= (1.05, 1), loc=2, borderaxespad=0);
```



In [60]:
```
# make a line plot for the price distribution over various month
# from this plot , both types of avocado in the month of jan to feb
prices are down , and there is
# sudden prices jumps in the month of sep to oct , after that there is a
sharp fall in the price in further month


plt.figure(figsize=(10,5))
sns.lineplot(x= 'month', y= 'AveragePrice', data= df, hue= 'type')
```

Out[60]: <AxesSubplot:xlabel='month', ylabel='AveragePrice'>



In [71]:
```
# make a groupby of date with finding their mean value
on_date = df.groupby('Date').mean()
on_date.head(5).T
```

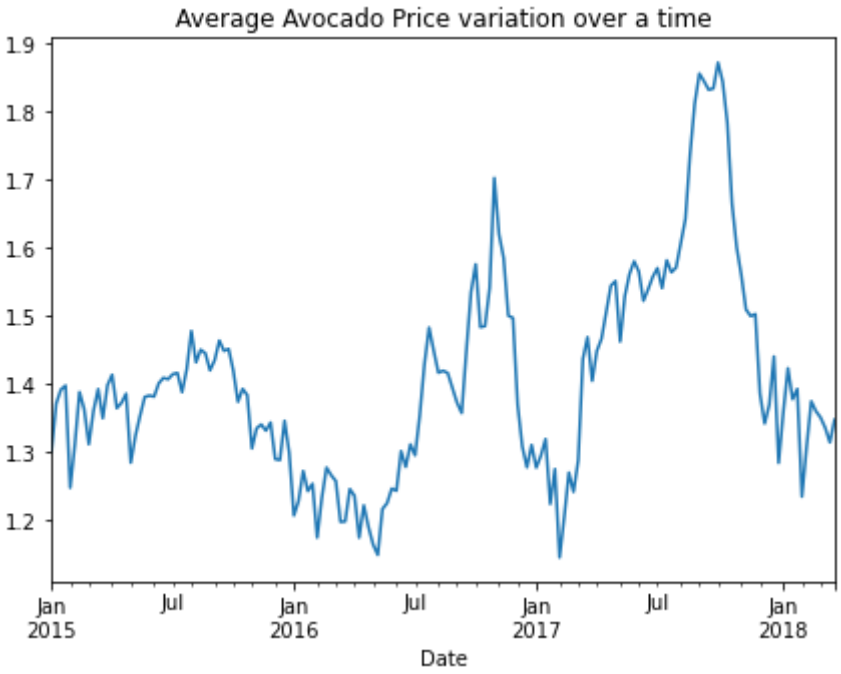
Out[71]:

Date	2015-01-04	2015-01-11	2015-01-18	2015-01-25	2015-02-01
AveragePrice	1.301296	1.370648	1.391111	1.397130	1.247037e+00

Date	2015-01-04	2015-01-11	2015-01-18	2015-01-25	2015-02-01
Total Volume	784021.640741	727368.585556	725822.074815	708021.121019	1.106048e+06
new col1	306465.358704	287260.786944	294469.507963	299121.600648	4.656758e+05
new col2	341213.318796	303781.757778	293167.054907	267862.244167	4.694095e+05
new col3	21100.172593	21751.550463	20446.764352	19244.248704	3.414342e+04
Total Bags	115242.790648	114574.490370	117738.747593	121793.027500	1.368197e+05
Small Bags	91759.341667	95860.522407	97674.575093	100715.897685	1.071929e+05
Large Bags	23015.332407	18313.259259	19866.538241	20859.406667	2.846286e+04
XLarge Bags	468.116574	400.708704	197.634259	217.723148	1.163921e+03
year	2015.000000	2015.000000	2015.000000	2015.000000	2.015000e+03
month	1.000000	1.000000	1.000000	1.000000	2.000000e+00

In [64]:
```
# average price distribution variation over a time
plt.figure(figsize=(7,5))
on_date['AveragePrice'].plot()
plt.title('Average Avocado Price variation over a time')
```

Out[64]: Text(0.5, 1.0, 'Average Avocado Price variation over a time')



In [75]:
```
# make a groupby for the month variable
on_date = df.groupby('month').mean()
on_date.head(5)
```

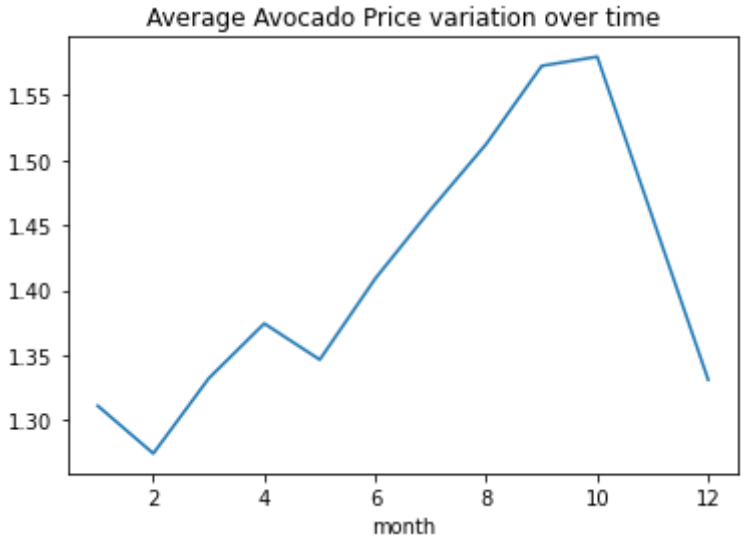
Out[75]:

month	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small
1	1.311019	9.035654e+05	297091.451924	328309.637593	23469.554733	254693.975550	190706.2

month	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small
2	1.274387	1.018825e+06	356672.920885	348558.374931	28394.881146	285196.078073	213055.0
3	1.332255	8.845054e+05	305386.962173	294504.626585	24952.196558	259658.362021	201308.6
4	1.374380	8.801935e+05	314698.492251	291711.817500	25904.380947	247878.811068	189665.5
5	1.346601	9.727150e+05	349946.910390	336679.188056	28377.930767	257710.995238	193425.1

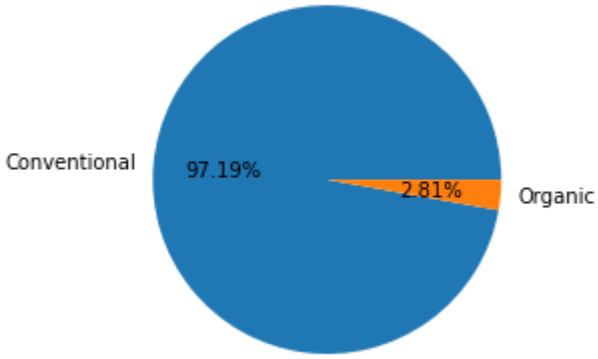
In [77]:
```
# make a line plot for price variation over time ..
on_date['AveragePrice'].plot()
plt.title('Average Avocado Price variation over time')
```

Out[77]: Text(0.5, 1.0, 'Average Avocado Price variation over time')



In [93]:
```
# most popular type of avocado,there is conventional type is higher in
demand avocado type with about
# 97.18% while organic is left with just 2.82%.
plt.figure(figsize= plt.figaspect(1))
total_volume = df.groupby('type')['Total Volume'].sum()
plt.pie(total_volume, labels = ['Conventional', 'Organic'], autopct=
'%.2f%%')
```

Out[93]:
```
([<matplotlib.patches.Wedge at 0x2288a7e6940>,
  <matplotlib.patches.Wedge at 0x2288a7e6fd0>],
 [Text(-1.0957170610706406, 0.09697485281617313, 'Conventional'),
  Text(1.0957170565309189, -0.09697490411039067, 'Organic')],
 [Text(-0.5976638514930765, 0.052895374263367156, '97.19%'),
  Text(0.5976638490168648, -0.05289540224203127, '2.81%')])
```



In [80]:
```
# other ways for doing groupby:
total_volume = df.groupby('type')['Total Volume'].sum()
total_volume
```

Out[80]:
```
type
conventional    1.508722e+10
organic         4.361817e+08
Name: Total Volume, dtype: float64
```

In [81]:
```
# others ways of doing group by as above:
df.groupby('type').sum()
```

Out[81]:

type	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	S
conventional	10568.27	1.508722e+10	5.280410e+09	5.245673e+09	4.143733e+08	4.146764e+09	3.16
organic	15089.43	4.361817e+08	6.670082e+07	1.406024e+08	2.429041e+06	2.264115e+08	1.60

In [83]:
```
df.groupby('Total Volume').sum().head(3)
```

Out[83]:

Total Volume	AveragePrice	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags	year	month
84.56	1.59	3.95	3.95	0.0	76.66	73.33	3.33	0.0	2015	11
379.82	1.73	0.00	59.82	0.0	320.00	320.00	0.00	0.0	2015	1
385.55	1.58	8.13	47.42	0.0	330.00	330.00	0.00	0.0	2016	10

In [91]:
```
df[['type','Total Volume']]
```

Out[91]:

	type	Total Volume
0	conventional	64236.62
1	conventional	54876.98
2	conventional	118220.22

	type	Total Volume
3	conventional	78992.15
4	conventional	51039.60
...	...	...
18244	organic	17074.83
18245	organic	13888.04
18246	organic	13766.76
18247	organic	16205.22
18248	organic	17489.58

18249 rows × 2 columns

In [89]:
```
# pivot table  of type and total volume:
table = pd.pivot_table(data=df,index=['type','Total Volume'])
table
```

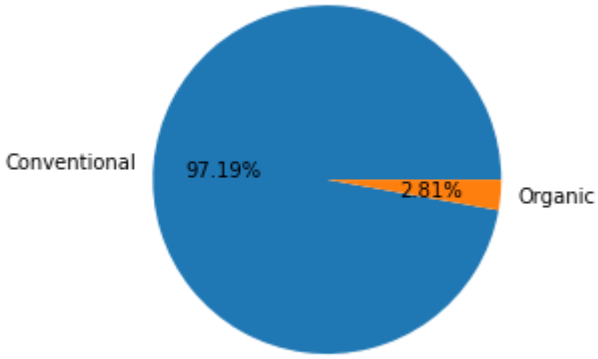
Out[89]:

		AveragePrice	Large Bags	Small Bags	Total Bags	XLarge Bags	month	new col1
type	**Total Volume**							
conventional	**33699.68**	1.56	2.22	6791.83	6794.05	0.00	10.0	967.38
	33757.95	1.55	0.00	8270.22	8270.22	0.00	11.0	690.91
	35852.68	1.41	0.00	13839.08	13839.08	0.00	11.0	719.36
	37045.75	1.44	448.89	8492.34	8941.23	0.00	9.0	775.55
	38598.98	1.61	0.00	9906.39	9906.39	0.00	11.0	814.68
...	**...**	...	...	...	...	...	...	...
organic	**1634430.77**	1.52	267818.31	831885.50	1099871.68	167.87	3.0	142345.03
	1634877.11	1.31	314962.41	433926.87	748889.28	0.00	2.0	250591.21
	1664234.88	1.52	180049.00	944572.50	1124621.50	0.00	3.0	129169.72
	1675804.22	1.54	221129.46	837351.85	1058651.50	170.19	3.0	170801.85
	1814929.97	1.52	103184.01	783017.98	886241.96	39.97	2.0	246515.35

18237 rows × 10 columns

In [94]:
```
plt.pie(total_volume, labels = ['Conventional', 'Organic'], autopct=
'%.2f%%')
```

Out[94]:
```
([<matplotlib.patches.Wedge at 0x2288a82d1c0>,
  <matplotlib.patches.Wedge at 0x2288a82d880>],
 [Text(-1.0957170610706406, 0.09697485281617313, 'Conventional'),
  Text(1.0957170565309189, -0.09697490411039067, 'Organic')],
 [Text(-0.5976638514930765, 0.052895374263367156, '97.19%'),
  Text(0.5976638490168648, -0.05289540224203127, '2.81%')])
```



In [105]:
```
# dis sort values only for regions in df...
most_expensive =df.groupby('region').sum().sort_values(by=
['AveragePrice'], ascending = False).head(10)
most_expensive
```

Out[105]:

region	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags
HartfordSpringfield	614.70	5.067054e+07	1319653.84	3.784762e+07	170253.64	1.133301e+07
SanFrancisco	609.82	1.358302e+08	34136889.78	8.315306e+07	3649315.48	1.489092e+07
NewYork	583.92	2.407341e+08	7639326.78	1.633538e+08	1746802.11	6.799416e+07
Philadelphia	551.66	7.183880e+07	4615200.96	4.274843e+07	599931.09	2.387523e+07
Sacramento	548.09	7.516375e+07	23451404.02	4.117691e+07	1553161.34	8.982268e+06
Charlotte	542.84	3.555554e+07	7563810.71	1.238027e+07	3913522.04	1.169794e+07
Northeast	541.45	7.132809e+08	34991207.79	4.744847e+08	6816644.64	1.969884e+08
Albany	527.63	1.606780e+07	616539.64	1.271597e+07	55037.33	2.680255e+06
Chicago	526.19	1.337023e+08	10844081.69	8.602744e+07	19965391.05	1.686540e+07
RaleighGreensboro	525.63	4.820273e+07	11820812.89	1.667812e+07	4027113.58	1.567668e+07

In [116]:
```
# pivot table of region and average price :
table = pd.pivot_table(data=df,index=['region','AveragePrice'])
table
```

Out[116]:

region	AveragePrice	Large Bags	Small Bags	Total Bags	Total Volume	XLarge Bags	month	new col1
Albany	**0.85**	468.850	10058.250	10527.100	81694.230	0.0	4.0	676.270
	0.93	103.140	8042.210	8145.350	118220.220	0.0	12.0	794.700
	0.98	562.370	6266.850	6829.220	109428.330	0.0	11.0	703.750
	0.99	218.766	8471.132	8689.898	73596.092	0.0	5.4	1182.906
	1.02	379.710	9156.510	9536.220	83805.250	0.0	7.0	1212.215

region	AveragePrice	Large Bags	Small Bags	Total Bags	Total Volume	XLarge Bags	month	new col1
...	...	...	...	...	...	...	...	...
WestTexNewMexico	2.50	31.430	9816.580	9848.010	16137.930	0.0	8.0	2616.960
	2.57	3.330	176.670	180.000	5647.480	0.0	10.0	593.810
	2.79	13.000	1116.940	1129.940	5562.000	0.0	10.0	790.450
	2.83	45.260	1231.670	1276.930	4984.760	0.0	9.0	652.270
	2.93	33.330	1146.070	1179.400	5373.880	0.0	10.0	901.060

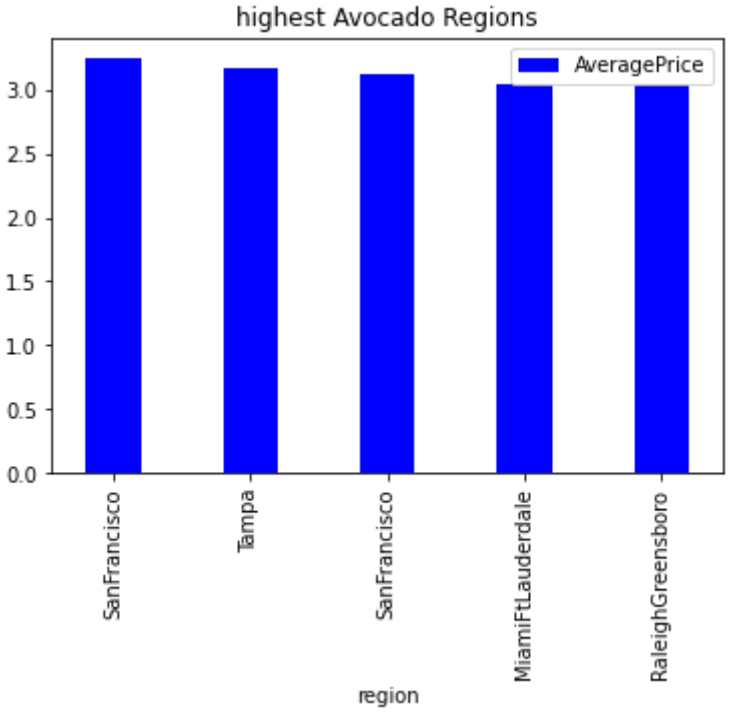
6359 rows × 10 columns

In [126]:
```
# these sort values for whole average price in df
most_expensive= df.sort_values(by=['AveragePrice'],
ascending=False).head()
most_expensive
```

Out[126]:

	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bags	XLarge Bags	ty
14125	2016-10-30	3.25	16700.94	2325.93	11142.85	0.00	3232.16	3232.16	0.00	0.0	orga
17428	2017-04-16	3.17	3018.56	1255.55	82.31	0.00	1680.70	1542.22	138.48	0.0	orga
14124	2016-11-06	3.12	19043.80	5898.49	10039.34	0.00	3105.97	3079.30	26.67	0.0	orga
16055	2017-03-12	3.05	2068.26	1043.83	77.36	0.00	947.07	926.67	20.40	0.0	orga
16720	2017-08-27	3.04	12656.32	419.06	4851.90	145.09	7240.27	6960.97	279.30	0.0	orga

In [127]:
```
# make a bar plot of region vs prices, this plot is telling us there is
place sanfrancisco is the region
# where avocado is most expensive with higher prices as compare to other
regions
most_expensive.plot(x= 'region', y='AveragePrice', kind='bar',
color='blue', width=0.4, title='highest Avocado Regions');
```

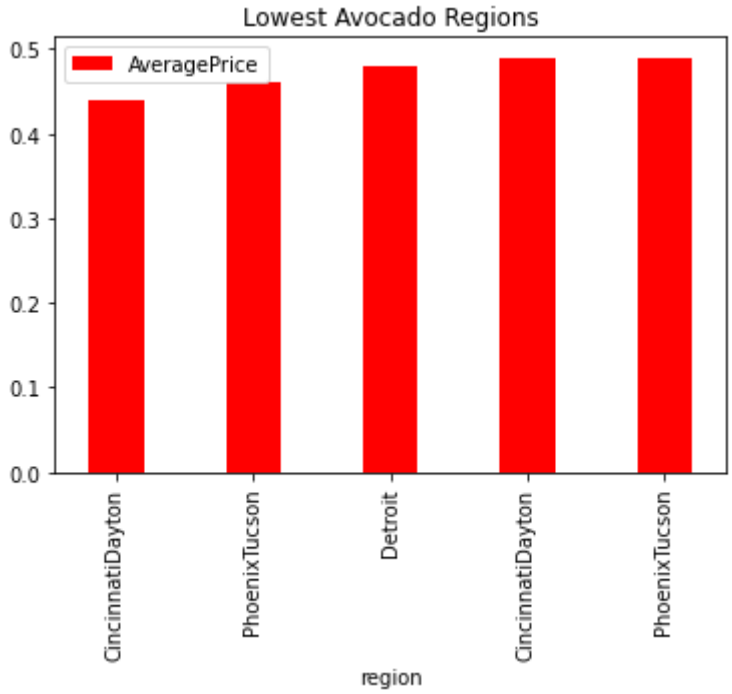


In [135]:
```
# these sort values for whole average price in df, for arranging the prices in ascending order
least_expensive= df.sort_values(by=['AveragePrice'], ascending=True).head()
least_expensive
```

Out[135]:

	Date	AveragePrice	Total Volume	new col1	new col2	new col3	Total Bags	Small Bags	Large Bag
15261	2017-03-05	0.44	64057.04	223.84	4748.88	0.00	59084.32	638.68	58445.64
7412	2017-02-05	0.46	2200550.27	1200632.86	531226.65	18324.93	450365.83	113752.17	330583.1(
15473	2017-03-05	0.48	50890.73	717.57	4138.84	0.00	46034.32	1385.06	44649.2(
15262	2017-02-26	0.49	44024.03	252.79	4472.68	0.00	39298.56	600.00	38698.5(
1716	2015-12-27	0.49	1137707.43	738314.80	286858.37	11642.46	100891.80	70749.02	30142.7(

In [133]:
```
# this bar plot showing that there is a region Cincinnati Dayton has lowest price of avocado, it is least
# expensive place for avocado sale, avocados sales at lowest prices in this region.
least_expensive.plot(x= 'region', y='AveragePrice', kind='bar', color='red', width=0.4, title='Lowest Avocado Regions');
```



Conclusion: Base on the above analysis and with close to 50:50 market share conventional 50.01% and organic 49.99%. Some recommendations could be made to the marketing strategy, although conventional avocado prices might be decreasing, it looks like there is an increased volume of avocados sold. The prices and avocado type sold varied across the regions. The lowest price of 0.44 dollars was recorded in Cincinnati Dayton and in march of 2017 which happened to be organic avocado, on the other hand, highest price of 3.25 dollars obtain in the region of San Francisco in October of 2016 still from Organic avocado. Looking at the total bags sold, we notice that a greater portion of the avocado where not sold in bags, in Seattle and some other regions, the where no avocado sold in bags. Mostly sold was Plu4225. Most of organic avocado where not sold in bag, but conventional where mostly sold in bags. However the postive correlation between total volume and small bags as well as total bags and small bags shows that small bags contributed the most to the total avocado sold. Also the organic appears to have high volitility because it has the lowerst and highest price, likewise the year 2017 appear to be very volitile as well. Having the most highest average price and lowest price at the same time.In all these Management will have to consider environmental, and other socio-economic conditions across these region that could have impact the avocado in the past 4 years.

In [ ]:

In [ ]:

In [ ]: