In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv ("avocado.csv",parse_dates=["Date"])
df.head(10)
```

Out[2]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conver
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conver
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conver
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conver
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conver
5	5	2015-11-22	1.26	55979.78	1184.27	48067.99	43.61	6683.91	6556.47	127.44	0.0	conver
6	6	2015-11-15	0.99	83453.76	1368.92	73672.72	93.26	8318.86	8196.81	122.05	0.0	conver
7	7	2015-11-08	0.98	109428.33	703.75	101815.36	80.00	6829.22	6266.85	562.37	0.0	conver
8	8	2015-11-01	1.02	99811.42	1022.15	87315.57	85.34	11388.36	11104.53	283.83	0.0	conver
9	9	2015-10-25	1.07	74338.76	842.40	64757.44	113.00	8625.92	8061.47	564.45	0.0	conver

In [3]:
```
parse_dates=['Date']
```

In [4]:
```
df.shape
```

Out[4]:
```
(18249, 14)
```

In [5]:
```
df.dtypes
#  here now the 'Date' column , earlier which is showing as an 'object' datatype, now after par
#  datetime formate.
```

Out[5]:
```
Unnamed: 0               int64
Date            datetime64[ns]
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 [6]:
```
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',
       '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',
       'year', 'region'],
      dtype='object')
```

In [7]:
```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
#  here the column "unnamed" having no relevance with the dataset, it is just having indexing r
#  so we have to drop this irrelevent column form the dataset.
```

In [8]:
```
df.columns
#  here we can see that the above mentioned column is now removed successfully.
```

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

In [9]:
```
#  here as above we can see that in the name of the columns, there are also some capital alphal
#  now simpler use we have make it all in small latter capitls..
```

In [10]:
```
column = ['date', 'average price', 'total volume', '4046', '4225', '4770', 'total bags','small
```

In [11]:
```
df.columns=column
```

In [12]:
```
df.head(5)
# here below we have changed all column names into small characters, and also one column is dr
```

Out[12]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

In [13]:
```
df.shape
# here we identifying the shape of our dataset whic is having  18249 ROWS & 13 COLUMNS
```

Out[13]:
```
(18249, 13)
```

In [14]:
```
df.info()
#  here we can see that
# 1) total number for columns present : 13
# 2) total number of rows presnet : 18,249
# 3) total "data types present in data set" : 4 (i.e "datetime64[ns](1), float64(9), int64(1),
# 4)NULL VALUES are may not be present in our dataset.
#    because there is no difference between the total count and present values.
#  5) So we can also to check 'null values' & 'whitespaces' in our dataset by futher different
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           18249 non-null  datetime64[ns]
 1   average price  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
dtypes: datetime64[ns](1), float64(9), int64(1), object(2)
memory usage: 1.8+ MB
```

In [ ]:
```

```

CHECKING NULL VALUES

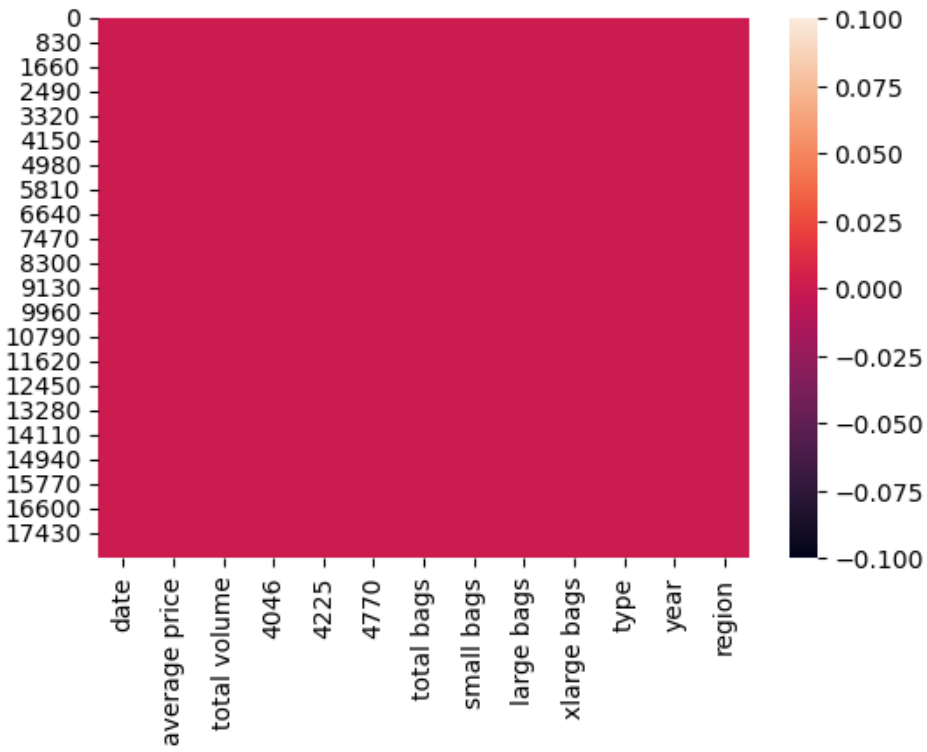
==========

In [15]:
```
df.isnull().sum()
#  here by following, we can clearly says that , there is NO NULL value is present in out datas
```

Out[15]:
```
date             0
average price    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
```

In [16]:
```
plt.figure(figsize=(6,4))
sns.heatmap(df.isnull())

# here with the help of heatmap also we can see that there is not any presence of null values
```

Out[16]: <AxesSubplot:>



In [ ]:
```

```

CHECKING UNIQUE VALUES & PERFORMING UNIVARIATE ANALYSIS FOR EACH COLUMN

==========

In [17]:
```
df.nunique()
#  here we can find the number of unique values in each column.
# as we know that there are total 18,249 rows are present in the given dataset.
# and here we can see that the column which consist very less number of unique value is 'type(
# ... so we can say that the above mentioned columns are 'Categorical Columns'
# here we can see that 'date' & 'average price' is also having very less unique numbers as com
```

Out[17]:
```
date              169
average price     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
dtype: int64
```

In [ ]:
```

```

In [18]:
```
# 1) Analysing DATE column-1 ====>>
```

In [19]:
```
df['date'].nunique()
#  total 169 unique dates are present in the column, out of 18,249 rows
```

Out[19]:
```
169
```

In [20]:
```
df['date'].value_counts()
```

Out[20]:
```
2015-12-27    108
2017-12-24    108
2017-12-10    108
2017-12-03    108
2017-11-26    108
              ...
2016-11-06    108
2018-01-07    108
2017-06-18    107
2017-06-25    107
2015-12-06    107
Name: date, Length: 169, dtype: int64
```

In [21]:
```
df['date'].value_counts().min()

# the minimum values assigned for any date is 107
```

Out[21]:
```
107
```

In [22]:
```
df['date'].value_counts().max()
# and the maximum values assigned for any date is 108
# that means all of 169 unique dates are having values upto 107 or 108.
```

Out[22]:
```
108
```

In [ ]:
```
```

In [23]:
```
# 2) Analysing Average Price column =====>>
```

In [24]:
```
df['average price'].nunique()
# out of 18,249 rows , there are 259 unique average prices are present in the column.
```

Out[24]:
```
259
```

In [25]:
```
df['average price'].value_counts()
```

Out[25]:
```
1.15    202
1.18    199
1.08    194
1.26    193
1.13    192
       ...
3.25      1
3.12      1
2.68      1
3.03      1
3.17      1
Name: average price, Length: 259, dtype: int64
```

In [26]:
```
df['average price'].value_counts().min()
#  the minimum assigned value for any average price is '1'
```

Out[26]:
```
1
```

In [27]:
```
df['average price'].min()
#  the Minimum average price present in the column is '0.44'
```

Out[27]:
```
0.44
```

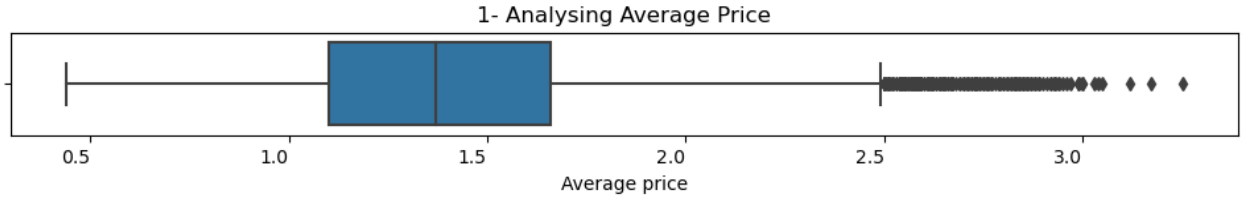
In [28]:
```
df['average price'].value_counts().max()
#  the maximum value assigned for any average price is '202'
```

Out[28]:
```
202
```

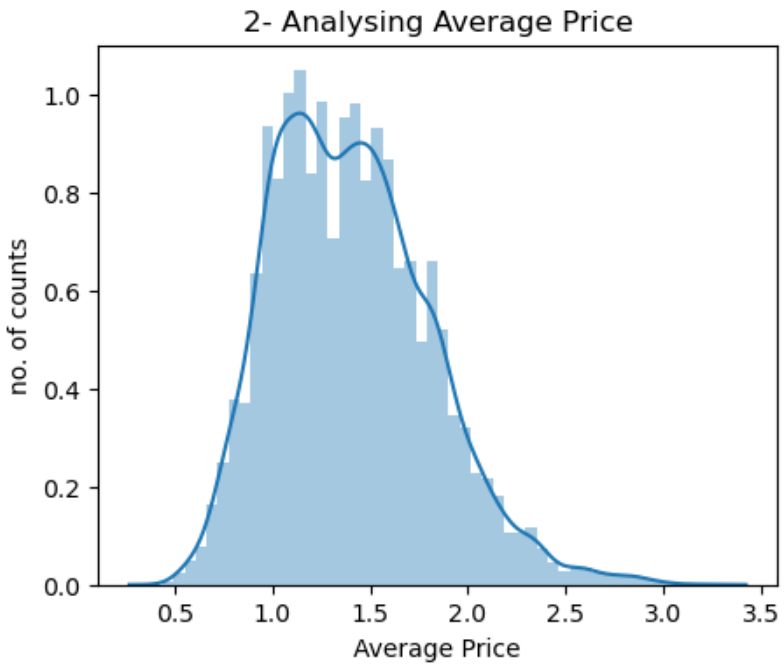
In [29]:
```
df['average price'].max()
#  the Maximum average price present in the column is '3.25'
```

Out[29]:
```
3.25
```

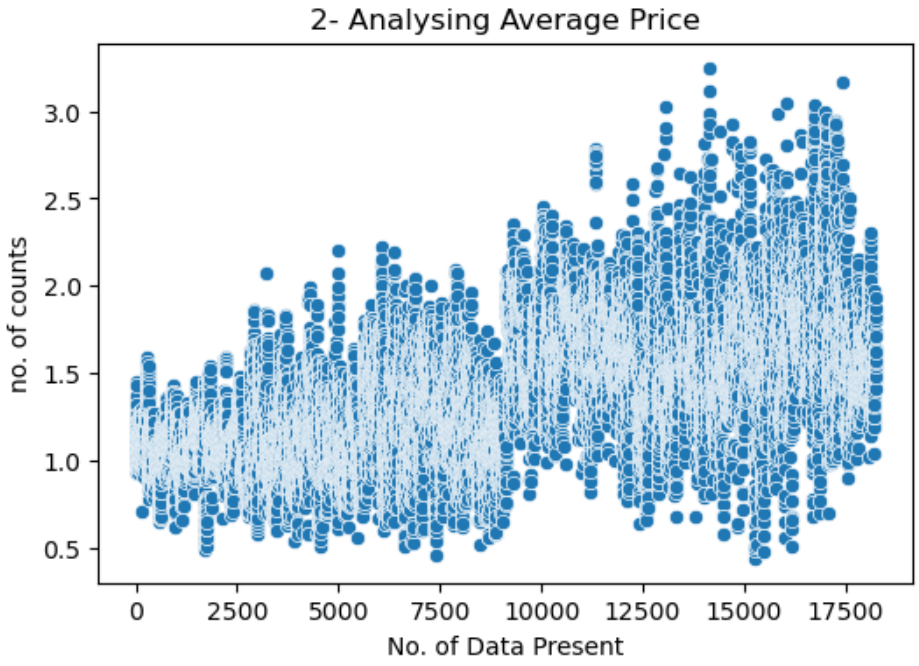
In [30]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('1- Analysing Average Price')
sns.boxplot(x='average price', data=df)
plt.xlabel('Average price', fontsize = 10)
plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below boxplot we can find that ,
#  The Minimum Average Price is below = 0.5 (i.e- 0.44)
#  The Maximum Average Price is Higher then = 3.0 (i.e- 3.25)
#  & most of the average prices are present in between (1.0 - 1.7)
# there may be presence of OUTLIERS in the following column.
```



In [31]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('2- Analysing Average Price')
sns.distplot(df['average price'])
plt.xlabel('Average Price', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the most of the values p
# .....as we have mentioned above also
#  here from the below distribution plot we can say that the data is 'RIGHT SKEWED' that means
```



In [32]:
```
plt.figure(figsize = (6,4), facecolor='white')
plt.title('2- Analysing Average Price')
sns.scatterplot(df.index,df['average price'])
plt.xlabel('No. of Data Present', fontsize = 10)
# plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here in the below scatterplot we can find that the NO. OF AVERAGE PRICE is INCREASING
#  The most of the Data is present in between the Average price of 1.0 -to- 2.5
```



In [ ]:
```

```

In [33]:
```
#  3) Analysing Total Volumn column ====>>
```

In [34]:
```
df['total volume'].nunique()
# outof 18,249 , there 18,237 unique values are present in the column, that means there are no
```

Out[34]: 18237

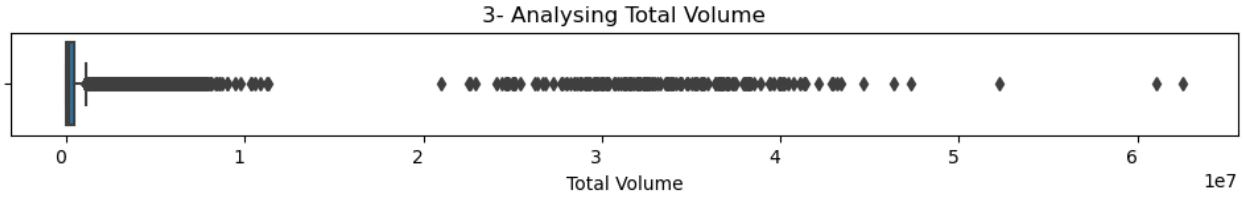
In [35]:
```
df['total volume'].min()
# Minimum 'total volume' vlaue prresent in the column is = 84.56
```

Out[35]: 84.56

In [36]:
```
df['total volume'].max()
#  Maximum 'total volume' value present in the column is = 62505646.52
```

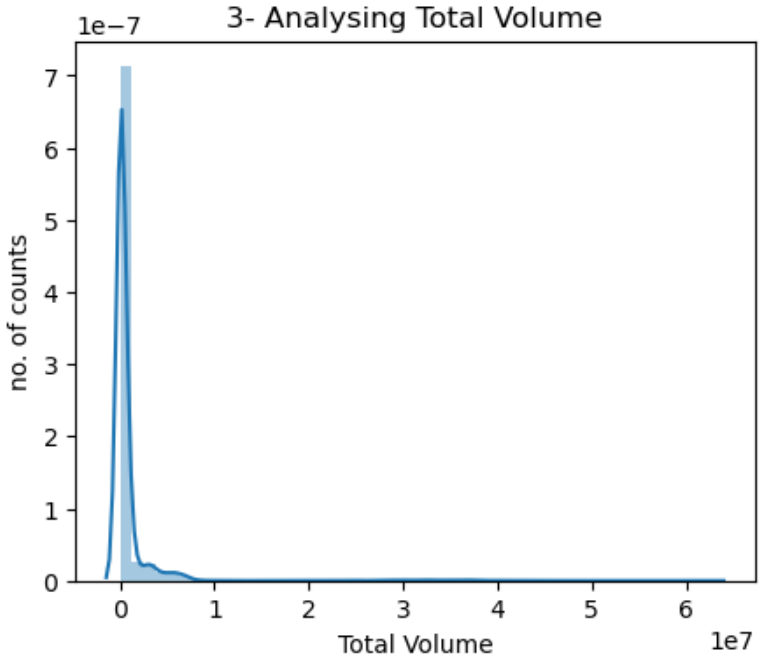
Out[36]: 62505646.52

In [37]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('3- Analysing Total Volume')
sns.boxplot(x='total volume', data=df)
plt.xlabel('Total Volume', fontsize = 10)
plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  It is very difficult to make any statement with the help of following plot.
#  but from below boxplot we can surely say that , there is the PRESENCE OF OUTLIERS in the co
```

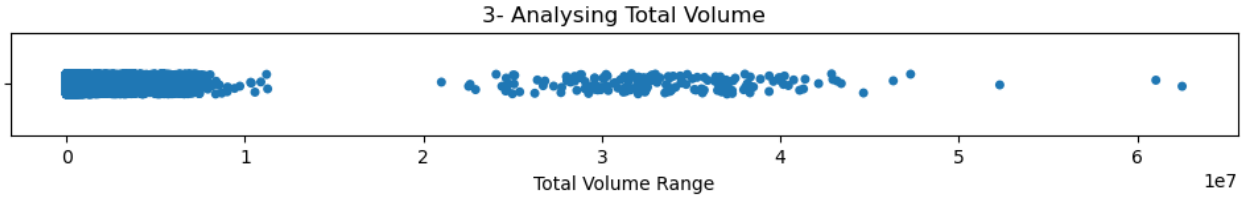


In [ ]:
```
```

In [38]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('3- Analysing Total Volume')
sns.distplot(df['total volume'])
plt.xlabel('Total Volume', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the most of the values p
#  here from the below distribution plot we can say that the data is 'VERY MUCH RIGHT SKEWED' t
# ...there may be presence of outliers
```

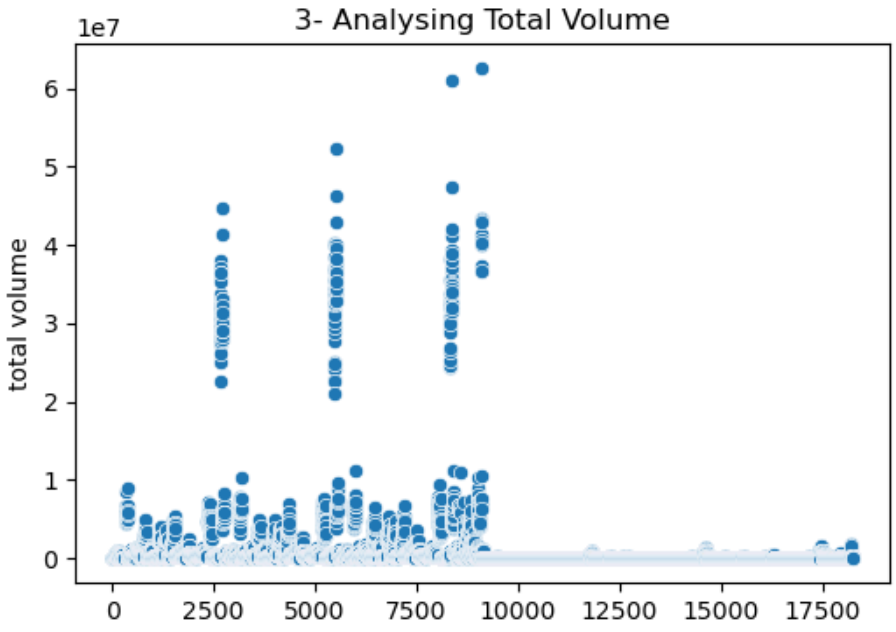


In [39]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('3- Analysing Total Volume')
sns.stripplot(x=df['total volume'])
plt.xlabel('Total Volume Range', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'Total Volume' is very DENSE i
# and the volume is SCATTERED IN THE RANGE OF 2 - 5.
#  above 5, there may be presence of outliers
```



In [ ]:
```
```

In [40]:
```
plt.figure(figsize = (6,4), facecolor='white')
plt.title('3- Analysing Total Volume')
sns.scatterplot(df.index,df['total volume'])
# plt.xlabel('Total Volume', fontsize = 10)
# plt.xticks(rotation=0,ha ='center')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here in the below scatterplot we can find that the maximum no. of total volume ranges betwe
```



In [41]:
```
(df['total volume']>100000).value_counts()
# here below we can find that the value count of 'total volume' above 1 lakh is 9,301
#                                                          & below is = 8,948
```

Out[41]:
```
True     9301
False    8948
Name: total volume, dtype: int64
```

In [ ]:

In [42]:
```
# 4) Analysing LPU 4046 (Product Lookup Code) =====>>
```

In [43]:
```
df['4046'].unique()
```

Out[43]:
```
array([1036.74,  674.28,  794.7 , ..., 1191.92, 1527.63, 2894.77])
```

In [44]:
```
df['4046'].nunique()
# out of 18249, there are 17,702 unique values are present in the column, which shows that it
```

Out[44]:
```
17702
```

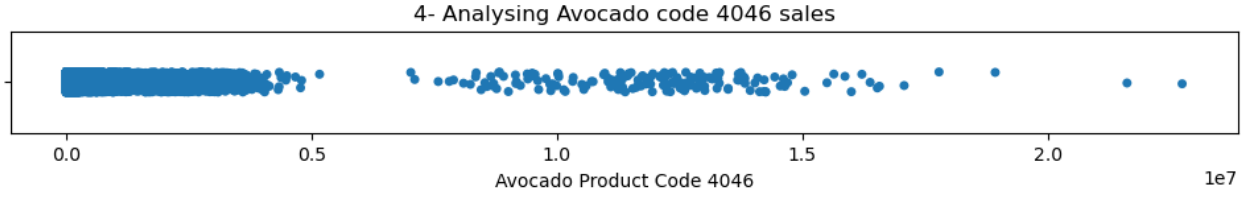
In [45]:
```
df['4046'].min()
# the minimum value is 0.0
```

Out[45]:
```
0.0
```

In [46]:
```
df['4046'].max()
#  the maximum value or quantity is 22743616.17
```

Out[46]:
```
22743616.17
```

In [47]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('4- Analysing Avocado code 4046 sales')
sns.stripplot(x=df['4046'])
plt.xlabel('Avocado Product Code 4046', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'product 4046' is very DENSE i
# and the volume is SCATTERED IN THE RANGE OF 0.7 - 1.5
#  above 2.0, there may be presence of outliers
```



In [ ]:

In [48]:
```
#  5) Analysing Avocado Product Code 4225 ====>>
```

In [49]:
```
df['4225'].nunique()
# out of 18,249 , there are 18,103 unique values are present, which meant it ia not a categori
```

Out[49]:
```
18103
```

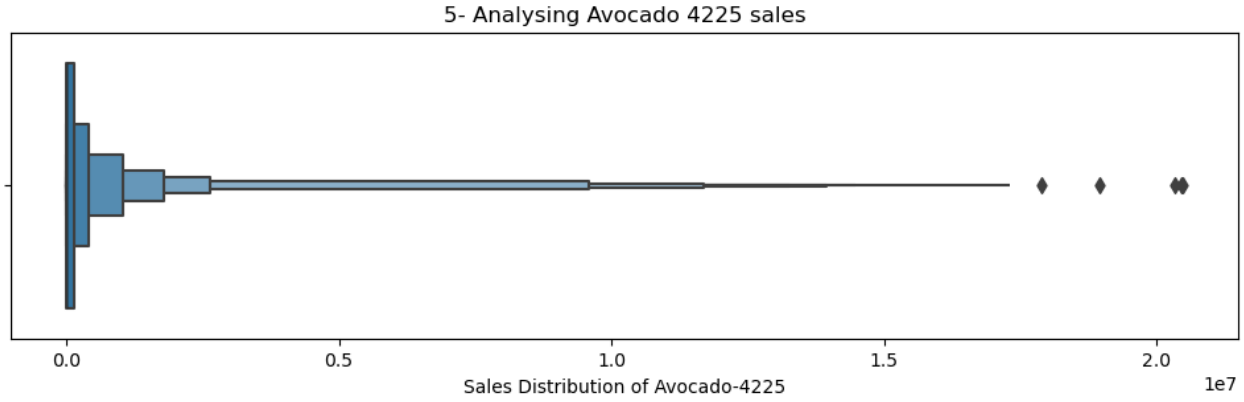
In [50]:
```
df['4225'].min()
```

Out[50]:
```
0.0
```

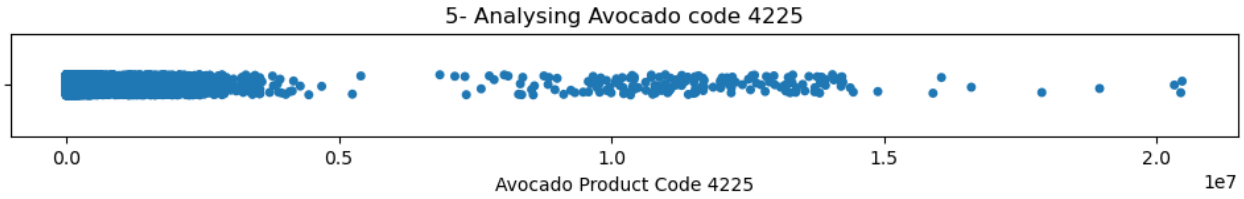
In [51]:
```
df['4225'].max()
# Mamimum no. sold  Avocado of -product code-4225
```

Out[51]:
```
20470572.61
```

In [52]:
```
plt.figure(figsize=(12,3))
plt.title('5- Analysing Avocado 4225 sales')
sns.boxenplot(data=df, x='4225')
plt.xlabel('Sales Distribution of Avocado-4225')
plt.show()

# here in the below 'boxenplot' we can see the distributioin of sales of 'AVOCADO-4225'.
# there may be presence of outliers in this column.
```



In [53]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('5- Analysing Avocado code 4225')
sns.stripplot(x=df['4225'])
plt.xlabel('Avocado Product Code 4225', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'product 4225' is very DENSE i
# and the volume is SCATTERED IN THE RANGE OF 0.7 - 1.5
#  above 1.5, there may be presence of outliers
```



In [ ]:
```

```

In [54]:
```
#  6) Analysing Avocado-4770 sales ======>>
```

In [55]:
```
df['4770'].nunique()
# outof 18,249, there are 12,071 unique values are present in the columns, but we can't say th
```

Out[55]:
```
12071
```

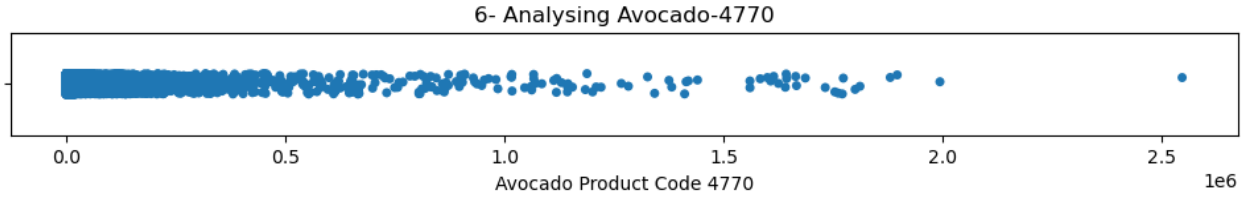
In [56]:
```
df['4770'].min()
```

Out[56]:
```
0.0
```

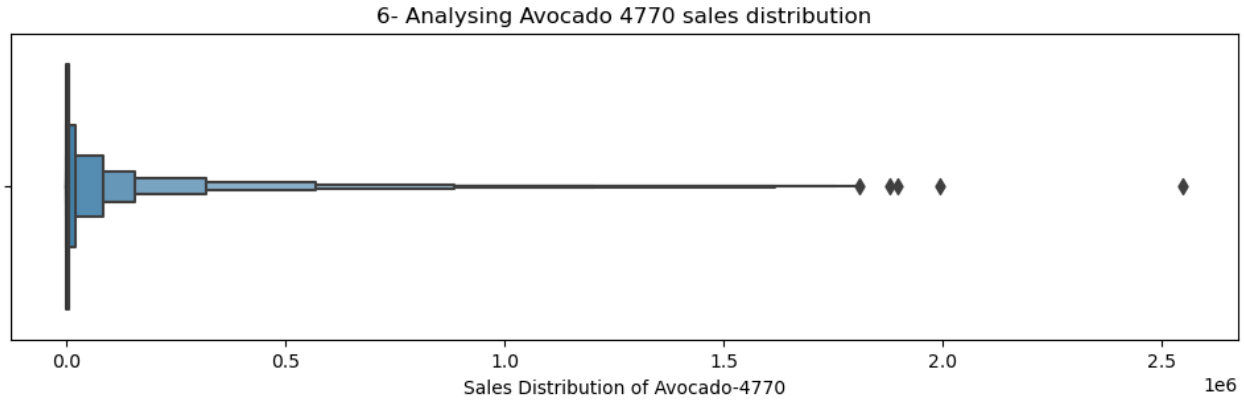
In [57]:
```
df['4770'].max()
```

Out[57]:
```
2546439.11
```

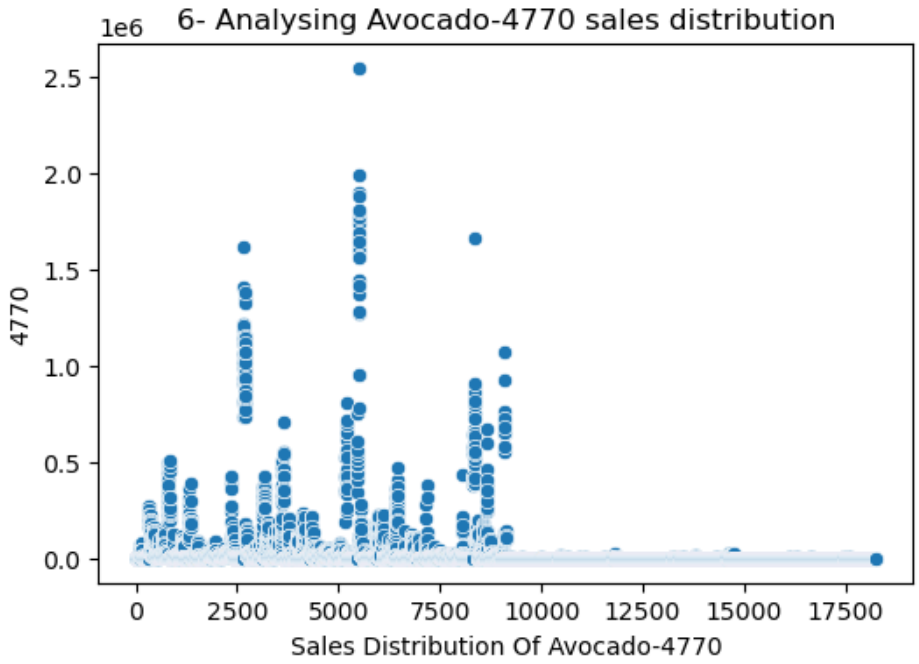
In [58]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('6- Analysing Avocado-4770')
sns.stripplot(x=df['4770'])
plt.xlabel('Avocado Product Code 4770', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'product 4770' is very DENSE i
# and the volume is SCATTERED IN THE RANGE OF 0.5 - 2.0
#  above 2.5, there may be presence of outliers
```



In [59]:
```
plt.figure(figsize=(12,3))
plt.title('6- Analysing Avocado 4770 sales distribution')
sns.boxenplot(data=df, x='4770')
plt.xlabel('Sales Distribution of Avocado-4770')
plt.show()

# here in the below 'boxenplot' we can see the distributioin of sales of 'AVOCADO-4225'.
# there may be presence of outliers in this column.
```



In [60]:
```
plt.figure(figsize = (6,4), facecolor='white')
plt.title('6- Analysing Avocado-4770 sales distribution')
sns.scatterplot(df.index,df['4770'])
plt.xlabel('Sales Distribution Of Avocado-4770', fontsize = 10)
# plt.xticks(rotation=0,ha ='center')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here in the below scatterplot we can find that the NO. OF Avocado-4770 sales
```



In [ ]:
```

```

In [61]:
```
#  7) Analysing Total Bag's Column ====>>
```

In [62]:
```
df['total bags'].unique()
```

Out[62]:
```
array([ 8696.87,  9505.56,  8145.35, ...,  9394.11, 10969.54, 12014.15])
```

In [63]:
```
df['total bags'].nunique()
#  it is not a categorical column, because out of 18,249, there are 18,097 unique no. of values
```

Out[63]:
```
18097
```

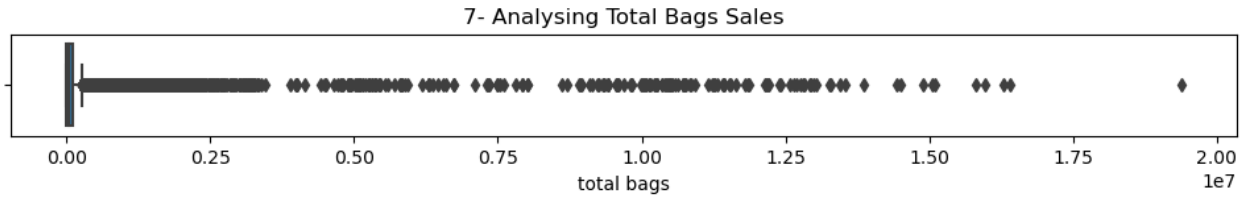
In [64]:
```
df['total bags'].min()
```

Out[64]:
```
0.0
```

In [65]:
```
df['total bags'].max()
# the Maximum no. of total bags of AVOCADO sold is - 1,93,73,134.37
```

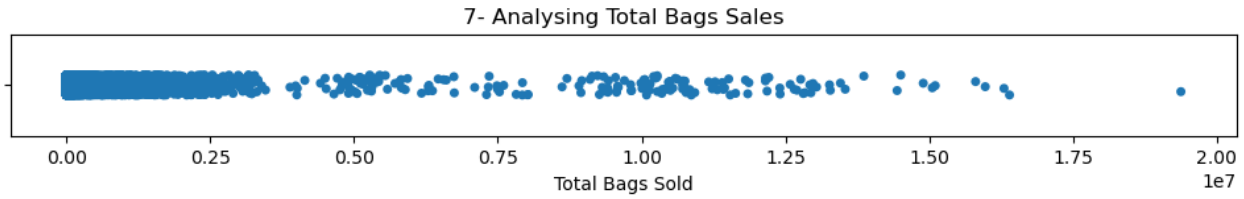
Out[65]:
```
19373134.37
```

In [66]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('7- Analysing Total Bags Sales')
sns.boxplot(x='total bags', data=df)
# plt.xlabel('Total Volume', fontsize = 10)
# plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  It is very difficult to make any statement with the help of following plot.
#  but from below boxplot we can surely say that , there is the PRESENCE OF OUTLIERS in the co
```



In [67]:
```
plt.figure(figsize=(12,3))
plt.title('7- Analysing Total Bags Sales')
sns.boxenplot(data=df, x='total bags')
plt.xlabel('Sales Distribution of Total no. of Bags')
plt.show()

# here in the below 'boxenplot' we can see the distributioin of sales of 'AVOCADO-4225'.
# there may be presence of outliers in this column.
```

In [68]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('7- Analysing Total Bags Sales')
sns.stripplot(x=df['total bags'])
plt.xlabel('Total Bags Sold', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'Total Bags Sold' is very DENSE
# and the volume is SCATTERED IN THE RANGE OF 0.50 - 1.50
#  above 1.75, there may be presence of outliers
```



In [ ]:

In [69]:
```
# 8) Analysing Small Bag sale ====>>
```

In [70]:
```
df['small bags'].nunique()
#  out of 18,249, there are 17,321 no. of unique values are present in the column.
# so it is not consider as a categorical column.
```

Out[70]: 17321

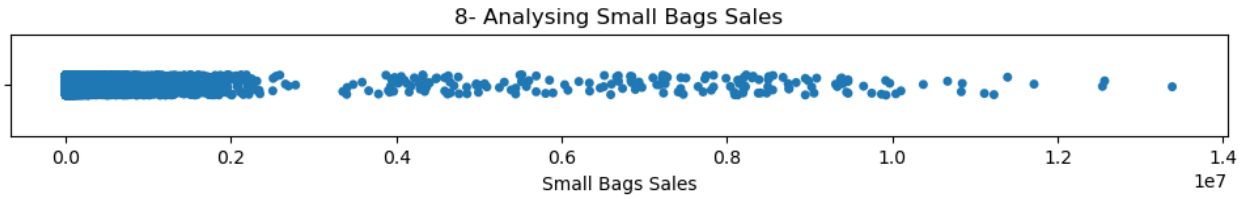
In [71]:
```
df['small bags'].min()
```

Out[71]: 0.0

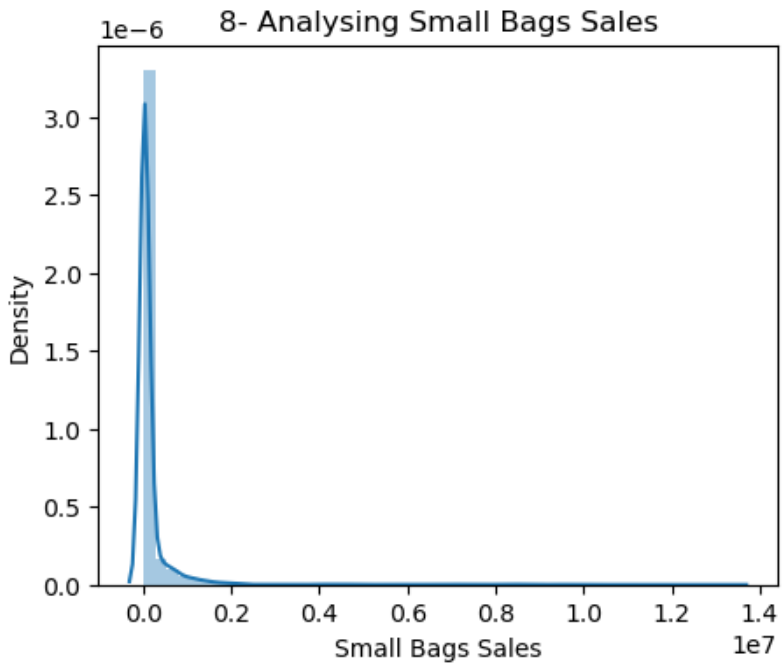
In [72]:
```
df['small bags'].max()
#  the maximum no. of small bag sales is = 1,33,84,586.8
#  As above we have seen that the TOTAL BAGS SALE IS = 1,93,73,134.37, OUTOF which only SMALL B
# .... which is a very huge amount.
```

Out[72]: 13384586.8

In [73]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('8- Analysing Small Bags Sales')
sns.stripplot(x=df['small bags'])
plt.xlabel('Small Bags Sales', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'small bag sales' is very DENSE
# and the volume is SCATTERED IN THE RANGE OF 0.25 - 1.2
#  above 1.2, there may be presence of outliers
```



In [74]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('8- Analysing Small Bags Sales')
sns.distplot(df['small bags'])
plt.xlabel('Small Bags Sales', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the DENSITY is verymuch
```



In [ ]:
```

```

In [75]:
```
#  9) Analysing Large Bags Sales ======>>>>
```

In [76]:
```
df['large bags'].nunique()
# there are 15,082 no. of unique values are present out of total 18,249 values
#  not consider as a categorical column.
```

Out[76]: 15082

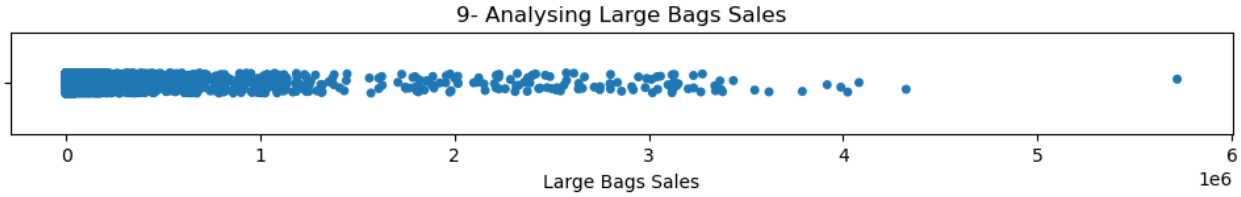
In [77]:
```
df['large bags'].min()
```

Out[77]: 0.0

In [78]:
```
df['large bags'].max()
# out of TOTAL BAGS SALES = 1,93,73,134.37,  57,19,096 Sales is of LARGE BAGS ONLY
```

Out[78]: 5719096.61

In [79]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('9- Analysing Large Bags Sales')
sns.stripplot(x=df['large bags'])
plt.xlabel('Large Bags Sales', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'Large bag sales' is very DENSE
# and the volume/density is SCATTERED IN THE RANGE OF 1.5 - 4
#  above 4, there may be presence of outliers
```



In [ ]:
```
```

In [80]:
```
# 10) Analysing Xtra Large Bags Sales =====>>>
```

In [81]:
```
df['xlarge bags'].unique()
```

Out[81]:
```
array([  0.  ,  33.33, 104.17, ..., 228.27,  12.12,  24.18])
```

In [82]:
```
df['xlarge bags'].nunique()
# 5588 no. of unique values are present in the column, out of 18,249
# not considered as a categorical column.
```

Out[82]:
```
5588
```

In [83]:
```
df['xlarge bags'].value_counts()
# out of 18,249 , 12,048 values are present in 0.00 quantity
```

Out[83]:
```
0.00       12048
3.33          29
6.67          16
1.11          15
5.00          12
             ...
3018.05        1
2739.44        1
9301.67        1
8640.00        1
24.18          1
Name: xlarge bags, Length: 5588, dtype: int64
```

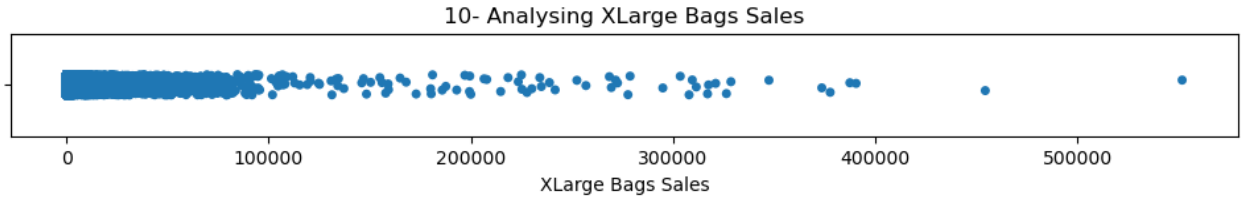
In [84]:
```
df['xlarge bags'].value_counts().min()
#  minimum value is 1 for any quantity of 'xlarge bags'
```

Out[84]:
```
1
```

In [85]:
```
df['xlarge bags'].value_counts().max()
#  maximum quantity is at 0.00 value
```

Out[85]:
```
12048
```

In [86]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('10- Analysing XLarge Bags Sales')
sns.stripplot(x=df['xlarge bags'])
plt.xlabel('XLarge Bags Sales', fontsize=10)
plt.show()
# here in the below stripplot graph we can clearly see that the 'Xtra Large bag sales' is very
# and the volume/density is SCATTERED IN THE RANGE OF 1 Lakh - 4 Lakh
#  above 4, there may be presence of outliers
```



In [87]:
```
(df['xlarge bags']>100000).value_counts()
#  above 1 Lakh only 87 no. of quantities are present
# and below this 18,162 total qunatities are present, if we deduct 12,048 quantities which are
#  ... below 1 Lakh remaining only = 6,114 X-Large Bags are Sold
```

Out[87]:
```
False    18162
True        87
Name: xlarge bags, dtype: int64
```

In [88]:
```
# 11) Analysing Types of Avocado ====>>>
```

In [89]:
```
df['type'].unique()
# here we can find that there are two types are present which is - CONVENTIONAL & ORGANIC
# which is the types of AVOCADO FRUIT, that they are ORGANIC Or CONVENTIONAL
```

Out[89]:
```
array(['conventional', 'organic'], dtype=object)
```

In [90]:
```
df['type'].nunique()
```

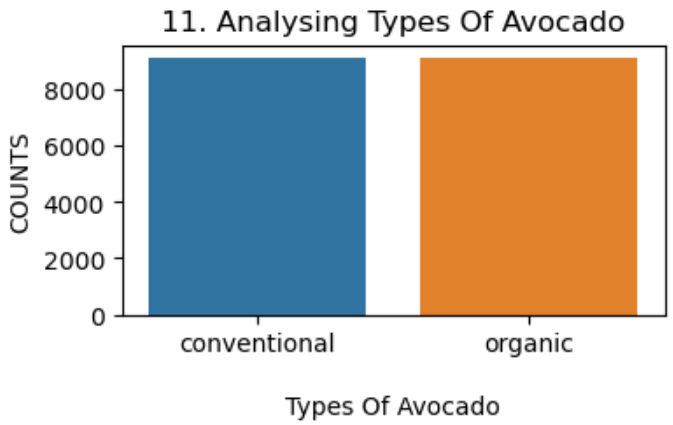
Out[90]:
```
2
```

In [91]:
```
df['type'].value_counts()
# both categories are having equivalent values
```

Out[91]:
```
conventional    9126
organic         9123
Name: type, dtype: int64
```

In [92]:
```
plt.figure (figsize = (4,2), facecolor = "white")
plt.title('11. Analysing Types Of Avocado')
sns.countplot(x='type', data = df)
plt.xlabel('\nTypes Of Avocado', fontsize=10)
# plt.xticks(rotation=30, ha = 'right')
plt.ylabel('COUNTS',fontsize=10)
# plt.yticks(rotation=30, ha = 'right')

# Here we can see that the most of counts are present in category 'A' & 'A-'.
# as above also we find the same in numerical form.
#  similarly we can also find same for all other columns.
```

Out[92]:
```
Text(0, 0.5, 'COUNTS')
```



In [ ]:
```

```

In [93]:
```
#  12) Analysing Year column =====>
```

In [94]:
```
df['year'].unique()
# the data which is available for the year of 2015,2016,2017 & 2018
```

Out[94]:
```
array([2015, 2016, 2017, 2018], dtype=int64)
```

In [95]:
```
df['year'].nunique()
```

Out[95]:
```
4
```

In [ ]:
```

```

In [96]:
```
# 13) Analysing Regions =====>
```

In [97]:
```
df['region'].unique()
```

Out[97]:
```
array(['Albany', 'Atlanta', 'BaltimoreWashington', 'Boise', 'Boston',
       'BuffaloRochester', 'California', 'Charlotte', 'Chicago',
       'CincinnatiDayton', 'Columbus', 'DallasFtWorth', 'Denver',
       'Detroit', 'GrandRapids', 'GreatLakes', 'HarrisburgScranton',
       'HartfordSpringfield', 'Houston', 'Indianapolis', 'Jacksonville',
       'LasVegas', 'LosAngeles', 'Louisville', 'MiamiFtLauderdale',
       'Midsouth', 'Nashville', 'NewOrleansMobile', 'NewYork',
       'Northeast', 'NorthernNewEngland', 'Orlando', 'Philadelphia',
       'PhoenixTucson', 'Pittsburgh', 'Plains', 'Portland',
       'RaleighGreensboro', 'RichmondNorfolk', 'Roanoke', 'Sacramento',
       'SanDiego', 'SanFrancisco', 'Seattle', 'SouthCarolina',
       'SouthCentral', 'Southeast', 'Spokane', 'StLouis', 'Syracuse',
       'Tampa', 'TotalUS', 'West', 'WestTexNewMexico'], dtype=object)
```

In [98]:
```
df['region'].nunique()
# there are 54 Regions are present in the column
```

Out[98]:
```
54
```

In [ ]:
```

```

========== upto here UNIVARIATE ANALYSIS FOR EACH COLUMN IS COMPLETED ==========

In [ ]:
```

```

========== Now Movng Forward Looking For BIVRIATE & MULTI-VARIATE ANALYSIS ==========

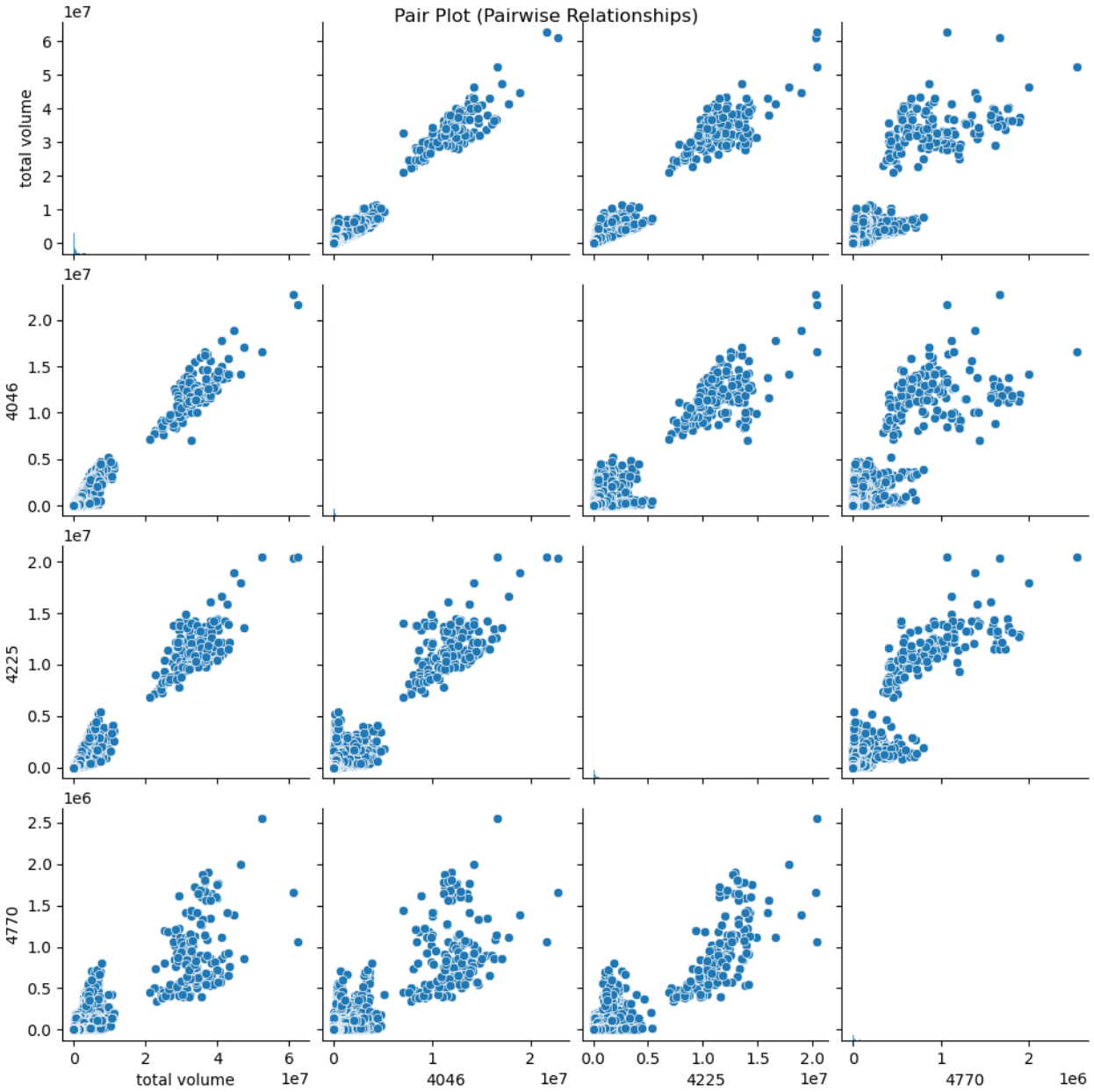
In [ ]:
```

```

In [99]:
```
df.head(5)
```

Out[99]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

In [100]:
```
sns.pairplot(data=df, vars=['total volume','4046', '4225', '4770'])
plt.suptitle('Pair Plot (Pairwise Relationships)')
plt.show()

# Here with the help of PAIRPLOT we are analysing TOTAL VOLUME with - '4046' - '4225' - '4770'
# here in the below SCATTER PLOT  we find the distribution of all three different varities of A
```



In [ ]:
```
```

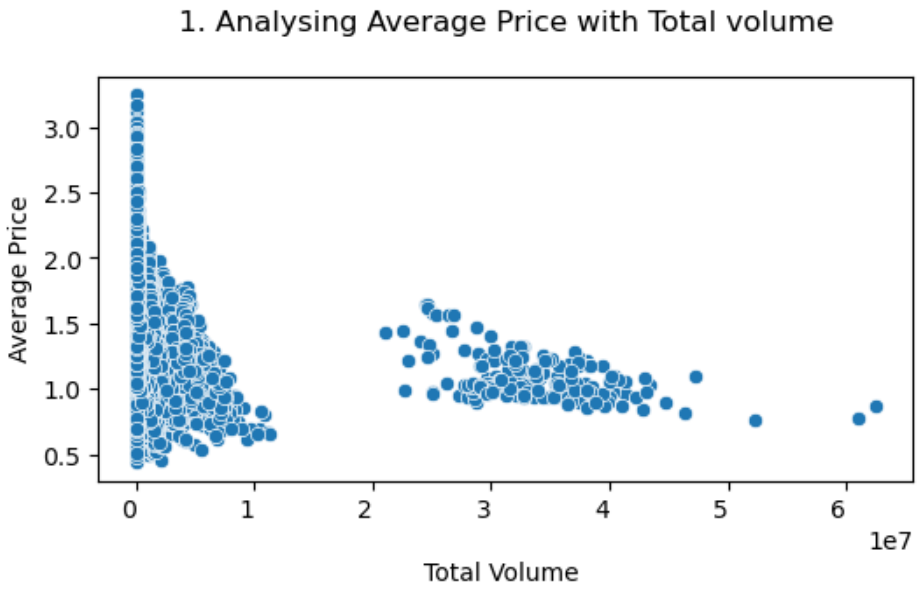
In [101]:
```
df.head(2)
```

Out[101]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany

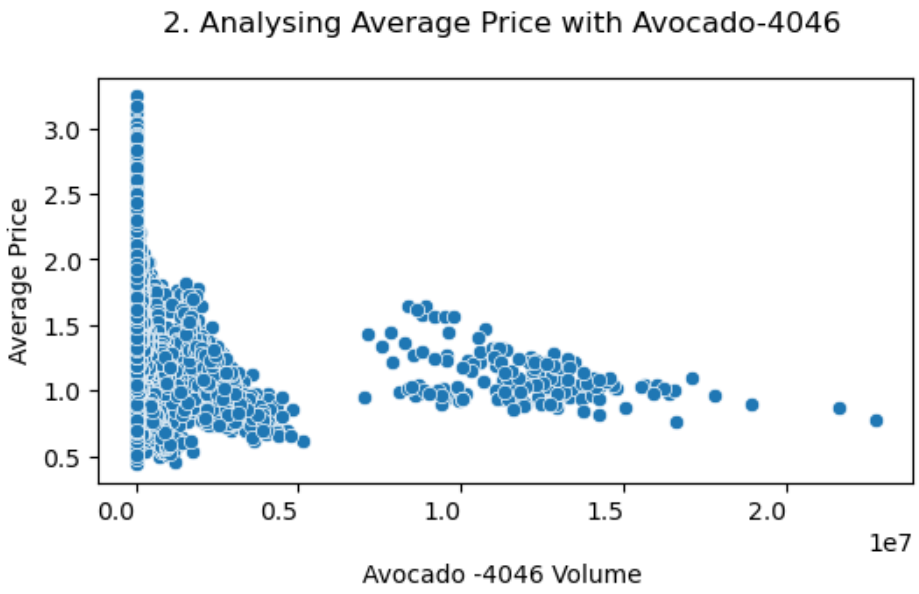
In [102]:
```
# 1) Analysing Average price v/s Total Volumne ======>>>
```

In [103]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n1. Analysing Average Price with Total volume\n')
sns.scatterplot (x= 'total volume', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nTotal Volume ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-1 ra
# and the Average Price is REDUCING as the no. of volume increses.
```



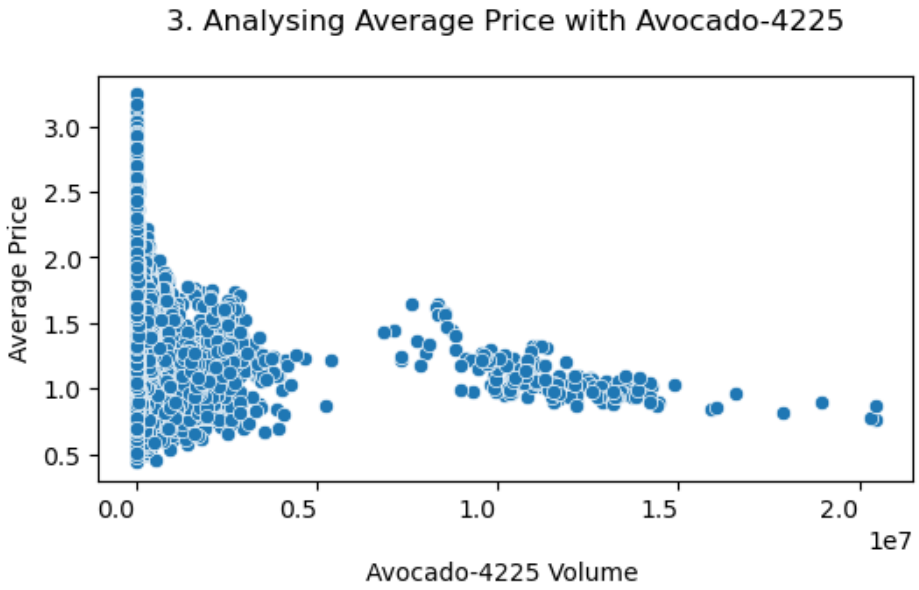
In [104]:
```
#  2) Analysing Average Price with Avocado-4046 ====>>
```

In [105]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n2. Analysing Average Price with Avocado-4046 \n')
sns.scatterplot (x= '4046', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nAvocado -4046 Volume ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-0.5 
# and the Average Price is REDUCING as the no. of volume increses.
# HERE WE GETTING SAME GRAPH AS ABOVE GRAPH OF ' TOTAL VOLUME', THE RESION BEHIND THIS IS ....
# ... FO THE QUANTITY IS OF AVOCADO-4046 SALES. which is already mentioned above in the univar
```



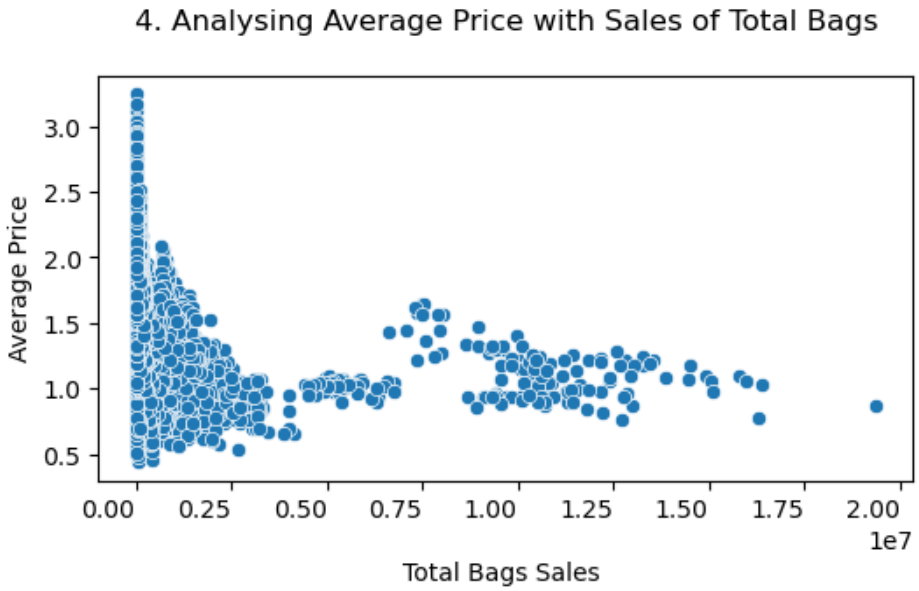
In [106]:
```
#  3) Analysing Average price with Avocad-4225  =====>>>
```

In [107]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n3. Analysing Average Price with Avocado-4225\n')
sns.scatterplot (x= '4225', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nAvocado-4225 Volume ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-.5 r
# and the Average Price is REDUCING with the INCREASING IN NO. OF VOLUME OF AVOCADO-4225
# here we can also find that the DENSITY OF VOLUME is HIGHER in the 0.0 -to- 0.5 volume range.
```



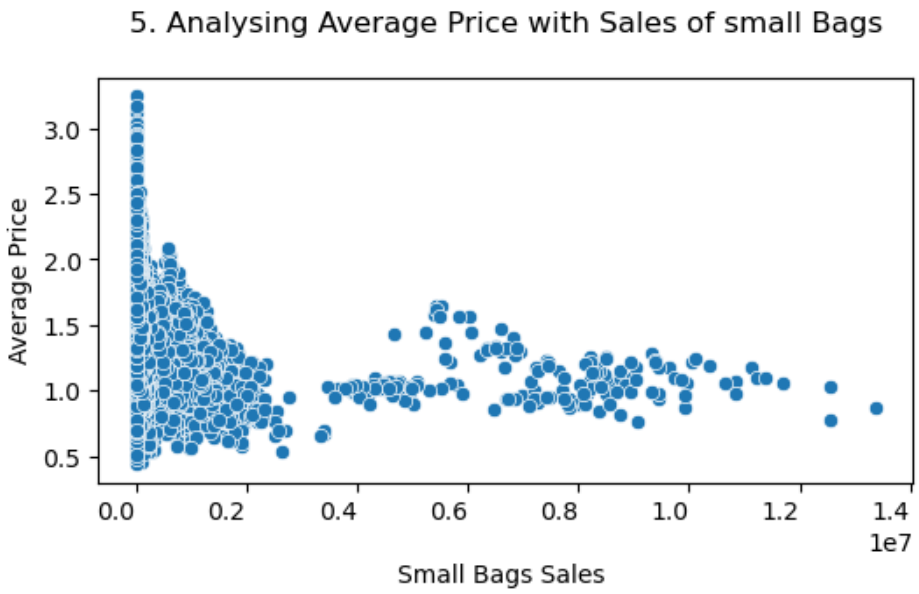
In [108]:
```
# 4) Analysing Average Price with Total bags sales =====>>>>
```

In [109]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n4. Analysing Average Price with Sales of Total Bags\n')
sns.scatterplot (x= 'total bags', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nTotal Bags Sales ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-.50 r
# and the Average Price is REDUCING with the INCREASING IN NO. OF VOLUME OF Total Bags Sales.
# here we can also find that the DENSITY OF Total bags sale  is HIGHER in the 0.0 -to- 0.50 ra
```



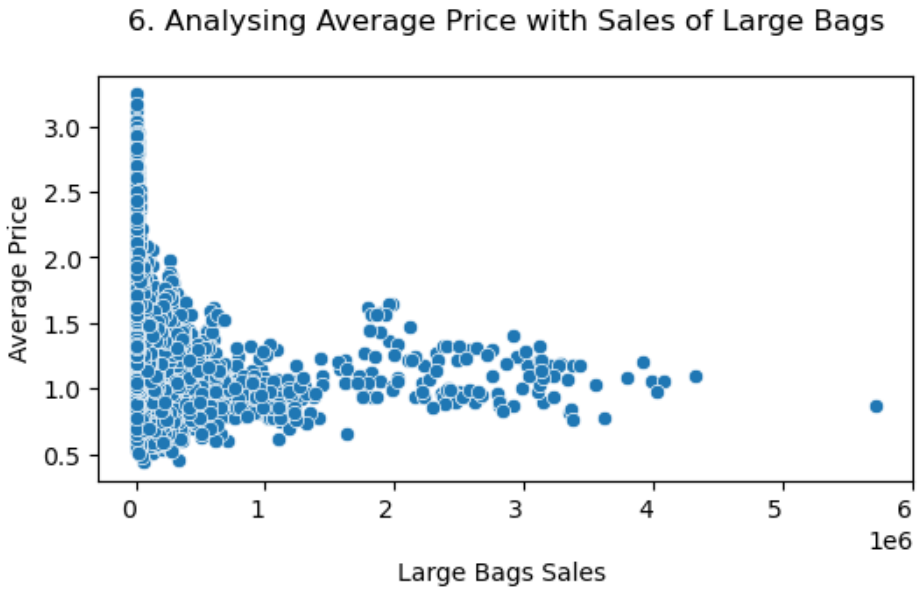
In [110]:
```
# 5) Analysing Average Price with Small Bags sales ====>>>
```

In [111]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n5. Analysing Average Price with Sales of small Bags\n')
sns.scatterplot (x= 'small bags', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nSmall Bags Sales ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-.20 r
# and the Average Price is REDUCING with the INCREASING IN NO. OF VOLUME OF Small Bags Sales.
# here we can also find that the DENSITY OF Small bags sale  is HIGHER in the 0.0 -to- 0.20 ra
# ....but it is also SCATTERED upto 1.4
```



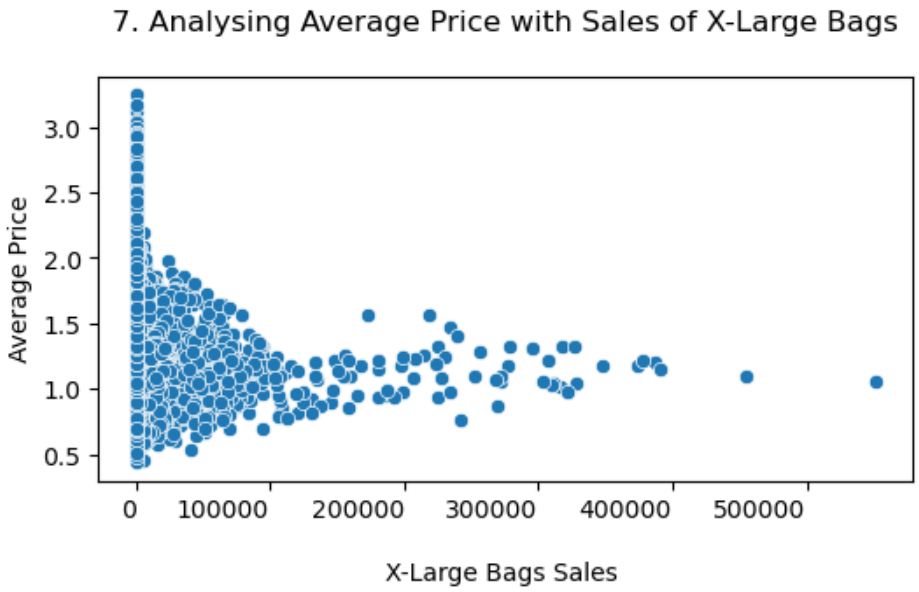
In [112]:
```
# 6) Analysing Average Price with Large Bags Sales ======>>>
```

In [113]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n6. Analysing Average Price with Sales of Large Bags\n')
sns.scatterplot (x= 'large bags', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nLarge Bags Sales ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-1 ra
# Here we can find an intresting fact the AVERAGE PRICE is NOT MUCH REDUCING with the INCRESIN
#  It may be due to , sales of LARGE BAGS is not much as compared to SMALL BAGS
```



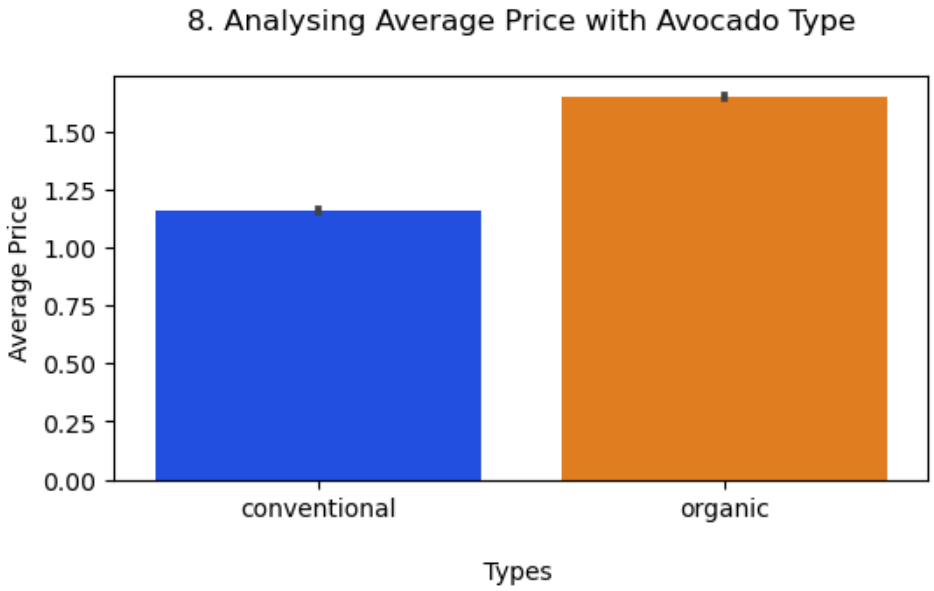
In [114]:
```
# 7) Analysing Average Price with X-Large Bags sales =====>>
```

In [115]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n7. Analysing Average Price with Sales of X-Large Bags\n')
sns.scatterplot (x= 'xlarge bags', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nX-Large Bags Sales ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price is Higher , in between the volume of 0-1 lak
# Here we can find an intresting fact the AVERAGE PRICE is NOT MUCH REDUCING with the INCRESING
#  It may be due to , sales of X-LARGE BAGS is not much as compared to LARGE BAGS
# there may be presence of OUTLIERS near 5 lakh quantity
# we can see that the DENSITY is deacresing from 0 to 5 lakh quanity of sales.
```



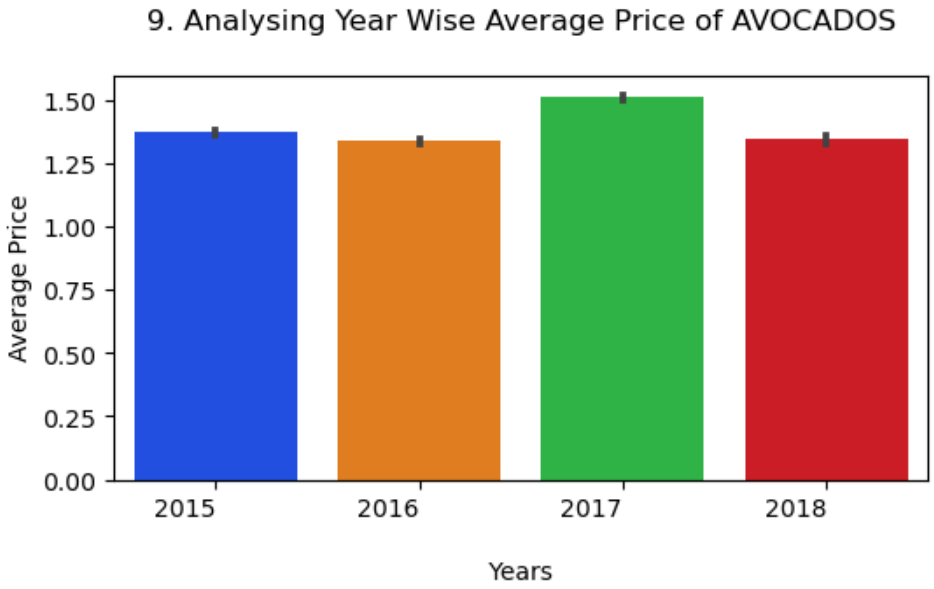
In [116]:
```
# 8) Analysing Average Price with type ======>>>
```

In [117]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n8. Analysing Average Price with Avocado Type\n')
sns.barplot (x= 'type', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Types  ')
# plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the Average price of 'ORAGNIC AVOCADOS' is MUCH MORE HIGHER TH
# So here can clearly see the Average price difference between the ORGANIC & CONVENTIONAL AVOCA
```



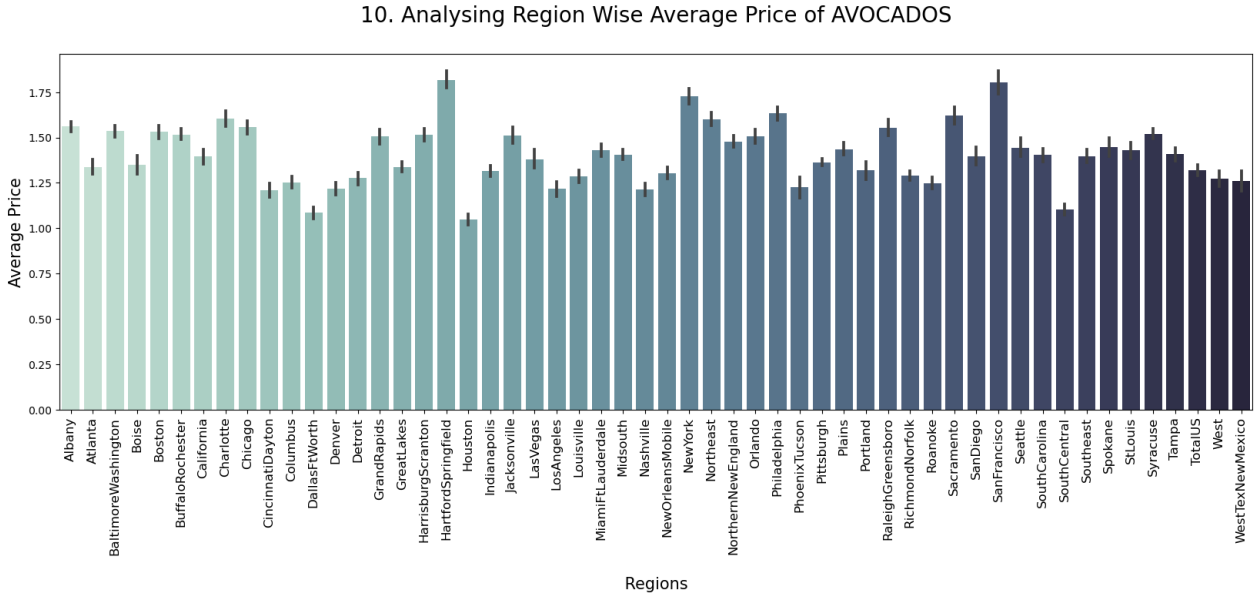
In [118]:
```
# 9) Analysing Average Price with year ====>>>
```

In [119]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n9. Analysing Year Wise Average Price of AVOCADOS\n')
sns.barplot (x= 'year', y = 'average price', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Years ')
plt.xticks (rotation = 0, ha= 'right')
plt.ylabel('Average Price')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can that the Average prices are Higher in the year 2017.
#  whereas in 2015, 2016 & 2018 the Average Prices are Almost Same.
```



In [120]:
```
# 10) Analysing Region wise Average price of avocados ======>>>>
```

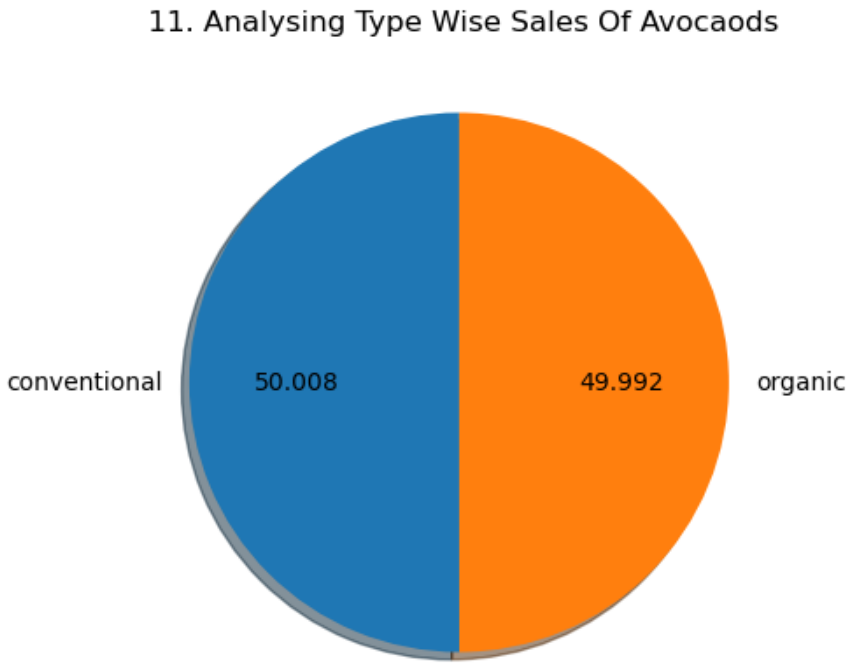
In [121]:
```
plt.figure (figsize = (20,6), facecolor = "white")
plt.title('\n10. Analysing Region Wise Average Price of AVOCADOS\n', fontsize = 20)
sns.barplot (x= 'region', y = 'average price', data= df, palette='ch:start=.2,rot=-.4')
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Regions ',fontsize=15)
plt.xticks (rotation = 90, ha= 'center',fontsize=12)
plt.ylabel('Average Price',fontsize=15)
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can that the Average prices of avocados are lying in betweeen 1.25 -to- 1.50 range
#  but in few of the regions average prices are higher,
# like = 'HartfordSpringfield', 'SanFrancisco', 'NEWYORK', 'NORTHEAST','PHILADELPHIA'
# & the Lowest avverage prices are in = 'DALLSFTWORTH','HOUSTON','PHOENIXTUCSON','SOUTHCENTRAL
```



In [ ]:
```
```

In [122]:
```
# 11) Analysing Type wise Total Sales of Avocados ====>>>
```

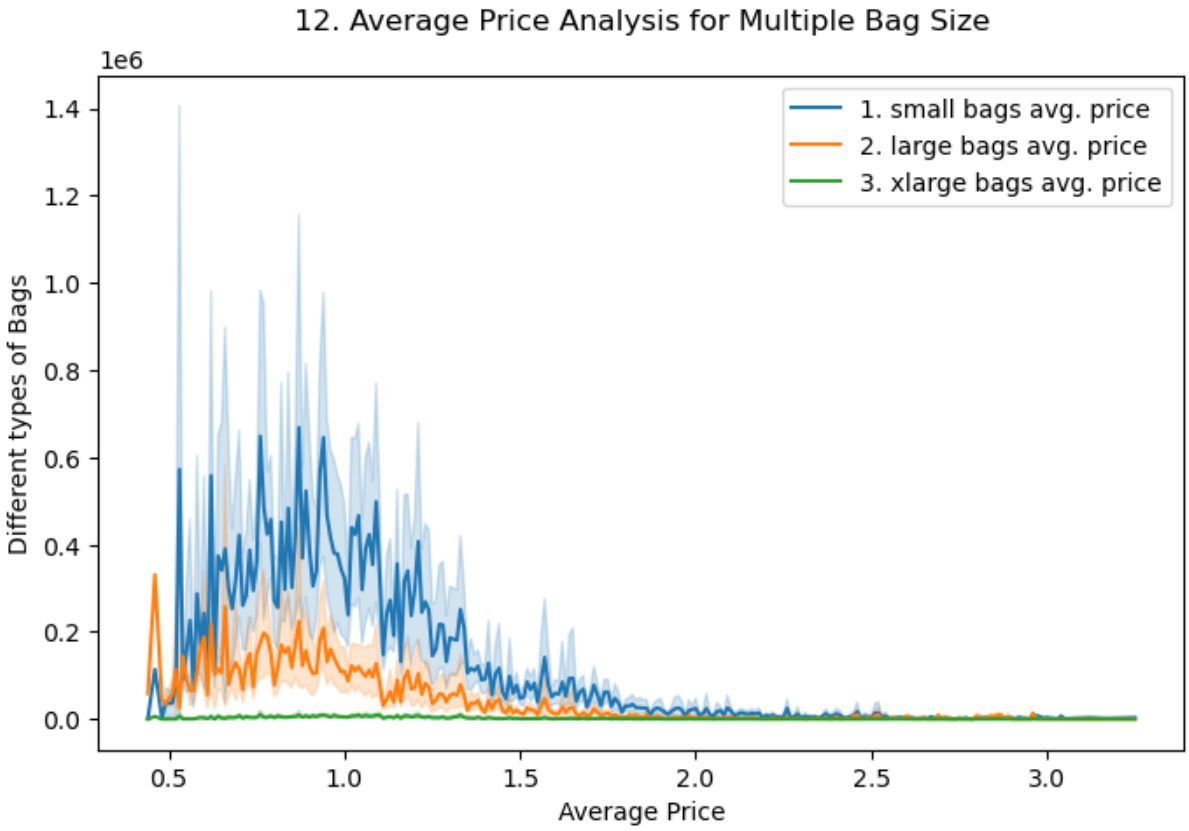
In [123]:
```
plt.figure(figsize=(5,5))
plt.title('\n 11. Analysing Type Wise Sales Of Avocaods')
plt.pie(df['type'].value_counts(),startangle=90,autopct='%.3f',labels=['conventional', 'organi
plt.show()

# here in the below pie plot we can fidn that the sales of both CONVENTIONAL & ORGANIC both are
```



In [ ]:
```

```

In [124]:
```
# 12) Analysing Average Price according to Bags size =====>>>>
```

In [125]:
```
plt.figure(figsize=(8,5))
plt.title('\n12. Average Price Analysis for Multiple Bag Size\n')
sns.lineplot (data=df, x='average price', y='small bags', label = '1. small bags avg. price')
sns.lineplot (data=df, x='average price', y='large bags', label = '2. large bags avg. price')
sns.lineplot (data=df, x='average price', y='xlarge bags', label = '3. xlarge bags avg. price'
plt.xlabel('Average Price')
plt.ylabel('Different types of Bags')
plt.legend()
plt.show()

# here in the following LINE GRAPH we can clearly see the difference between the AVERAGE PRICES
# the Average price for per Avocado in small bags is higher then Large & x-large Bags average p
```

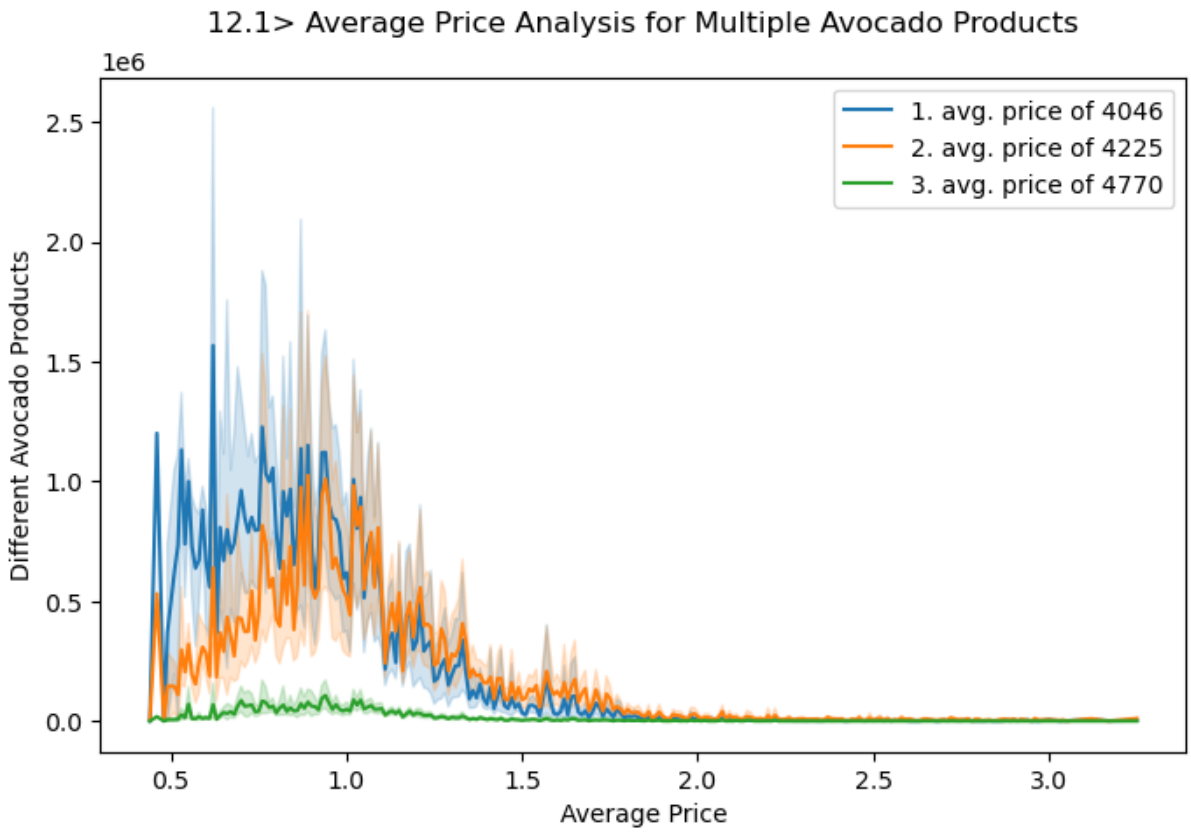


In [ ]:

In [ ]:

In [126]:
```
#  12.1 > Analysing the Average Price For the Multiple Avocado Products =====>>>>>
```

In [127]:
```
plt.figure(figsize=(8,5))
sns.lineplot (data=df, x='average price', y='4046', label = '1. avg. price of 4046')
sns.lineplot (data=df, x='average price', y='4225', label = '2. avg. price of 4225')
sns.lineplot (data=df, x='average price', y='4770', label = '3. avg. price of 4770')
plt.xlabel('Average Price')
plt.ylabel('Different Avocado Products')
plt.title('\n12.1> Average Price Analysis for Multiple Avocado Products\n')
plt.legend()
plt.show()

# here in the below LINE GRAPH we can find that the average price for '4046' > '4225' > '4770'
#  it is in the correct manner because we also finds the sales of above products in the same m
```

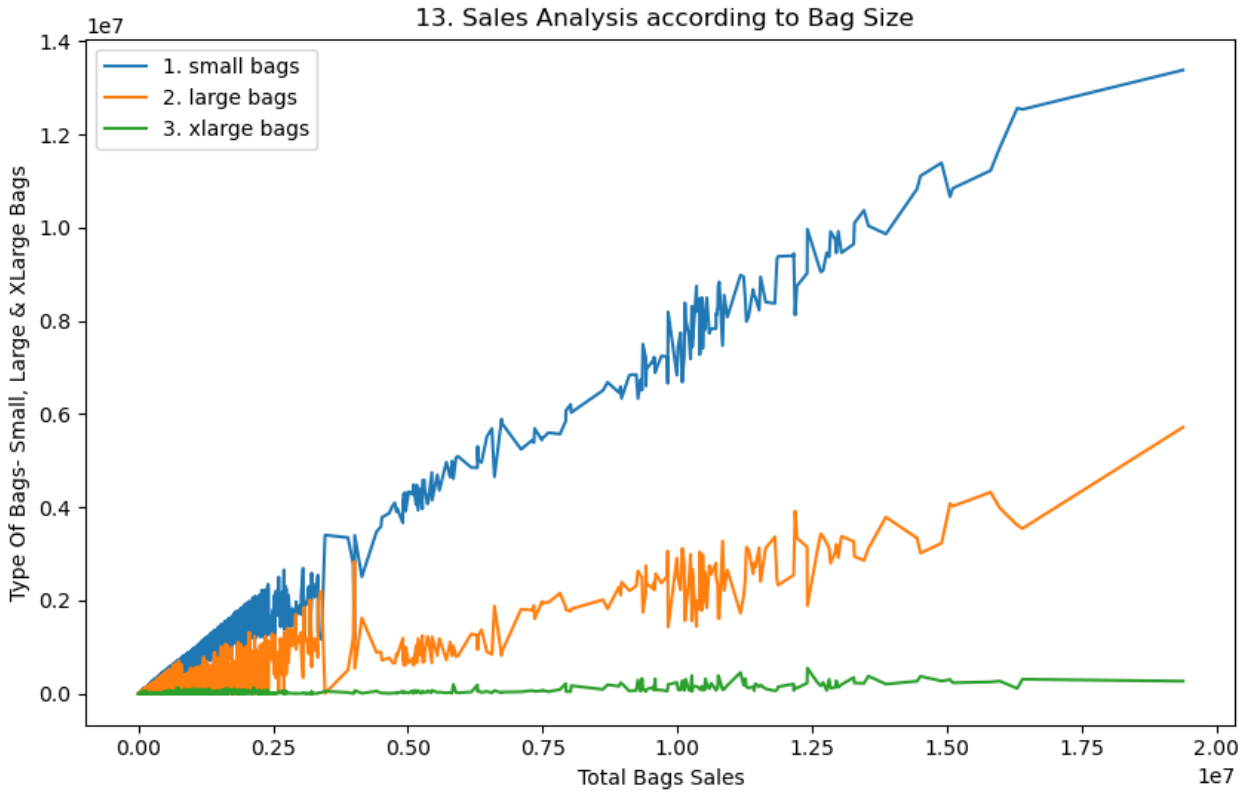


In [ ]:

In [ ]:

In [128]:
```
# 13) Analysing Sales according to Bag Size ====>>>
```

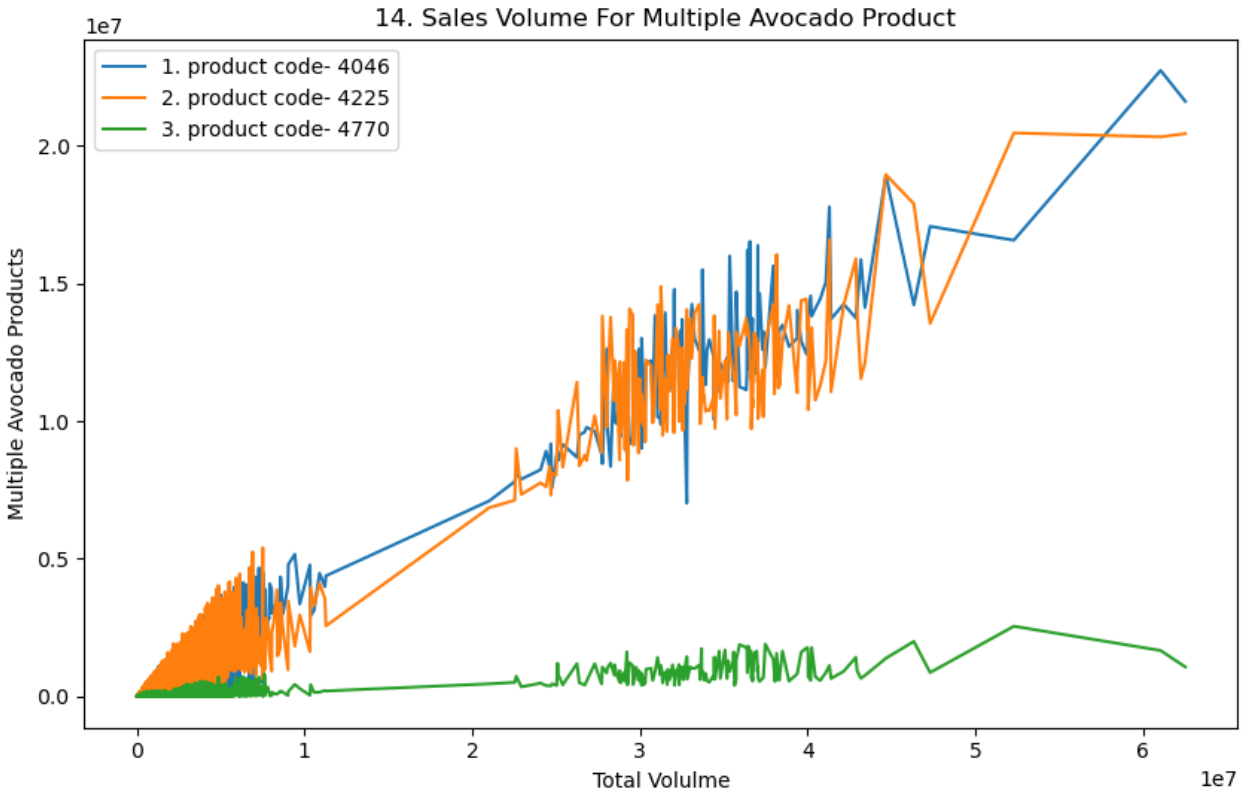
In [129]:
```
plt.figure(figsize=(10,6))
sns.lineplot (data=df, x='total bags', y='small bags', label = '1. small bags')
sns.lineplot (data=df, x='total bags', y='large bags', label = '2. large bags')
sns.lineplot (data=df, x='total bags', y='xlarge bags', label = '3. xlarge bags')
plt.xlabel('Total Bags Sales')
plt.ylabel('Type Of Bags- Small, Large & XLarge Bags')
plt.title('\n 13. Sales Analysis according to Bag Size')
plt.legend()
plt.show()

# here in the following LINE GRAPH we can find the SALES GROWTH of SMALL, LARGE & X-LARGE BAGS
# here below we can find that the sales growth is : SMALL BAGS > LARGE BAGS > X-LARGE BAGS
# there is HUGE DIFFERENCE BETWEEN the sales of SMALL BAGS & X-LARGE BAGS
# due to which we can conclude that the CUSTOMERS ARE LIKE TO BUY SMALL BAGS as compared to LA
```



In [ ]:

In [130]:
```
# 14) Analysing Sales Volume for Multiple Avocado Prduct
```

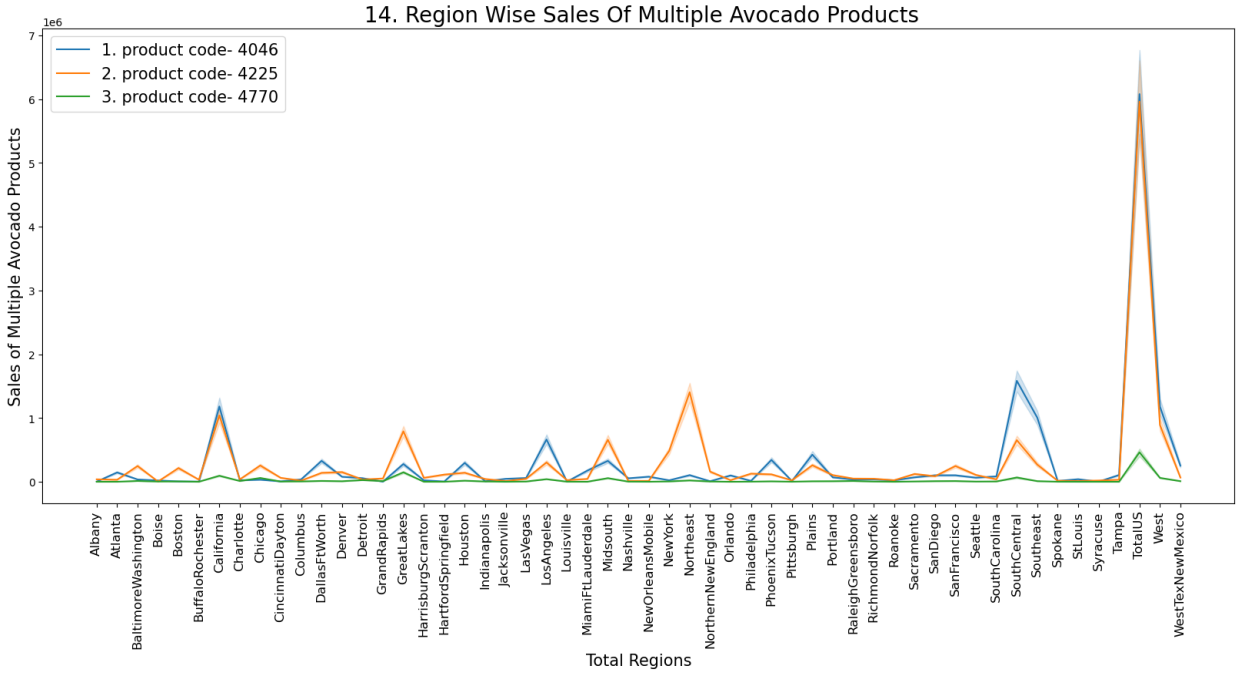
In [131]:
```
plt.figure(figsize=(10,6))
sns.lineplot (data=df, x='total volume', y='4046', label = '1. product code- 4046')
sns.lineplot (data=df, x='total volume', y='4225', label = '2. product code- 4225')
sns.lineplot (data=df, x='total volume', y='4770', label = '3. product code- 4770')
plt.xlabel('Total Volulme')
plt.ylabel('Multiple Avocado Products')
plt.title('\n 14. Sales Volume For Multiple Avocado Product')
plt.legend()
plt.show()

#  here below in the LINE GRAPH we find the SALES VOLUME for the MULTIPLE AVOCADO PRODUCTS (404
# here below we can find that the sales volume of AVOCADO PRODUCT WITH THE CODE OF - 4046 & 422
# ....HIGHER AS COMPARED to AVOCADO-4770.
# From this we can conclude that the CUSTOMERS ARE LIKING SVOCADO-4046 & 4225 MORE AS COMPARED
```



In [ ]:

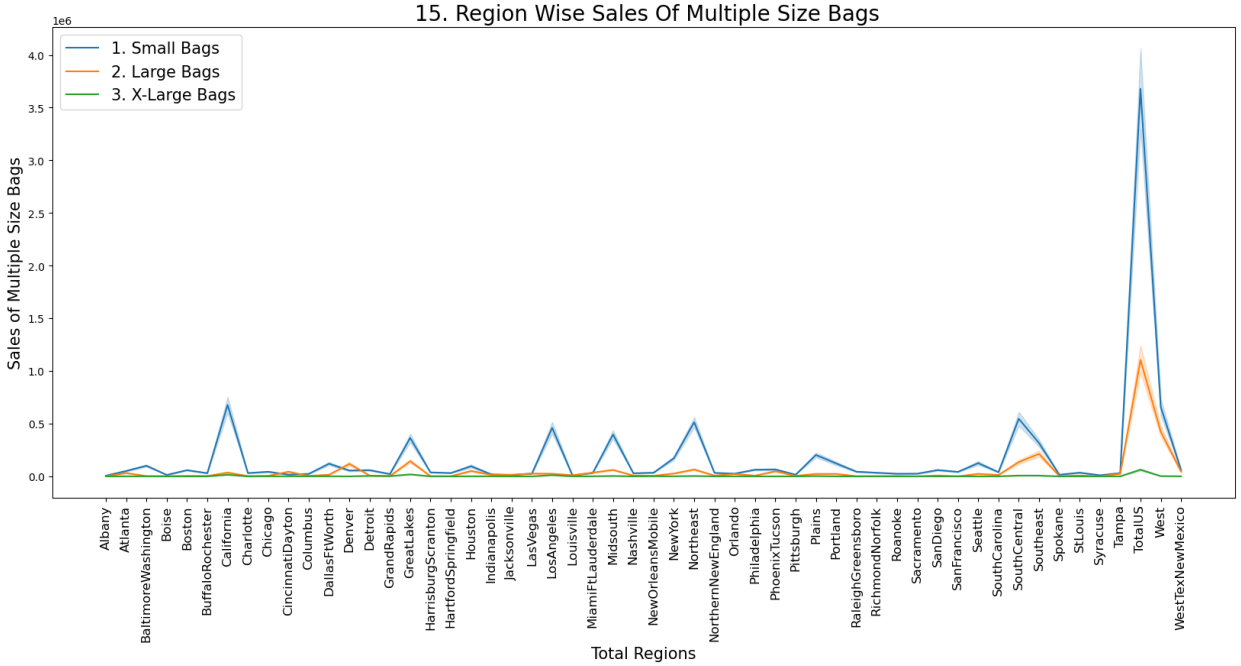
In [132]:
```
# 14) Analysing Region Wise Sales of Mutiplt Avocado Products ====>>>>
```

In [133]:
```
plt.figure(figsize=(20,8))
plt.title('\n 14. Region Wise Sales Of Multiple Avocado Products',fontsize=20)
sns.lineplot (data=df, x='region', y='4046', label = '1. product code- 4046')
sns.lineplot (data=df, x='region', y='4225', label = '2. product code- 4225')
sns.lineplot (data=df, x='region', y='4770', label = '3. product code- 4770')
plt.xlabel('Total Regions',fontsize=15)
plt.xticks(rotation=90,ha='center',fontsize=12)
plt.ylabel('\nSales of Multiple Avocado Products',fontsize=15)
plt.legend(loc='upper left', fontsize=15)
plt.show()

#  here below in the LINE GRAPH we find the SALES OF MILTIPLE AVOCADO PRODUCTS
# Here below we can find that the sales of 4770 is LOW as compared to 4225 & 4046 in overall RE

# the Sales of 4046 & 4225 both are Higher in = 'BuffaloRochester', 'California', 'Charlotte',

# the Sales of 4046 is Higher in = 'LasVegas', 'LosAngeles', 'Louisville','Philadelphia','Phoen
#                                  'Plains', 'Portland','SouthCarolina','SouthCentral', 'Southe

# the Sales of 4225 is Higher in = 'GrandRapids', 'GreatLakes', 'HarrisburgScranton','MiamiFtLa
#                                  'NewOrleansMobile', 'NewYork','Northeast', 'NorthernNewEngla

#  from the above analysis we can conclude that AVOCADO 4046 & 4225 are more liked by CUSTOMERS
```



In [134]:
```
#  15) Analysing Region Wise Sales of Multiple Size Bags =====>>>
```

In [135]:
```
plt.figure(figsize=(20,8))
plt.title('\n 15. Region Wise Sales Of Multiple Size Bags',fontsize=20)
sns.lineplot (data=df, x='region', y='small bags', label = '1. Small Bags')
sns.lineplot (data=df, x='region', y='large bags', label = '2. Large Bags')
sns.lineplot (data=df, x='region', y='xlarge bags', label = '3. X-Large Bags')
plt.xlabel('Total Regions',fontsize=15)
plt.xticks(rotation=90,ha='center',fontsize=12)
plt.ylabel('\nSales of Multiple Size Bags',fontsize=15)
plt.legend(loc='upper left', fontsize=15)
plt.show()

#  here below in the LINE GRAPH we find the SALES OF MILTIPLE AVOCADO PRODUCTS
# Here below we can find that the sales of 4770 is LOW as compared to 4225 & 4046 in overall RE

# the Sales of SMALL ABGS are Higher in = 'BuffaloRochester', 'California', 'Charlotte','SouthC
#                                          'TotalUS', 'West','WestTexNewMexico'
#                                         ,'GrandRapids', 'GreatLakes', 'HarrisburgScranton','L
#                                       'LosAngeles', 'Louisville','MiamiFtLauderdale','Midsout
#                                       'NewOrleansMobile', 'NewYork','Northeast', 'NorthernNew
#                                       'Pittsburgh', 'Plains', 'Portland','RaleighGreensboro',
#                                     'Southeast', 'Spokane'

#  THE HIGHEST SALE OF SMALL & LARGE BAGS ARE IN = 'Tampa', 'TotalUS', 'West', 'WestTexNewMexic
```



In [ ]:
```
```

========== UP TO HERE HERE ANALYSIS PART IS COMPLETED
==========

In [136]:
```
df.head(5)
```

Out[136]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

In [137]:
```
#  here above as we can see in our dataset 'type' & 'region'is of 'object data type'
```

In [138]:
```
df.dtypes
```

Out[138]:
```
date             datetime64[ns]
average price           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 [139]:
```
#  so we need to encode them before aplly any other methods.
#  so first we need to apply ENCODING TECHNIQUE.
```

ENCODING TECHNIQUES

==========

In [140]:
```
# here in our dataset we are having two 'object' columns which need's to be ENCODED for further
# so here we are apllying "LABEL ENCODER" for both columns:-
#  for which we have to import some libraries
```

In [141]:
```
from sklearn.preprocessing import LabelEncoder
```

In [142]:
```
le = LabelEncoder()
```

In [143]:
```
df.head(1)
```

Out[143]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany

In [144]:
```
df["type"] = le.fit_transform(df["type"])
df["region"] = le.fit_transform(df["region"])
```

In [145]:
```
df.head(1)
```

Out[145]:

	date	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	2015	0

In [146]:
```
# here above as we can see our type and region column hasbeen succeffuly converted.
```

In [147]:
```
df.dtypes
# as we can see both of the column types are converted from 'object' to 'integer'
```

Out[147]:
```
date             datetime64[ns]
average price           float64
total volume            float64
4046                    float64
4225                    float64
4770                    float64
total bags              float64
small bags              float64
large bags              float64
xlarge bags             float64
type                      int32
year                      int64
region                    int32
dtype: object
```

In [ ]:
```

```

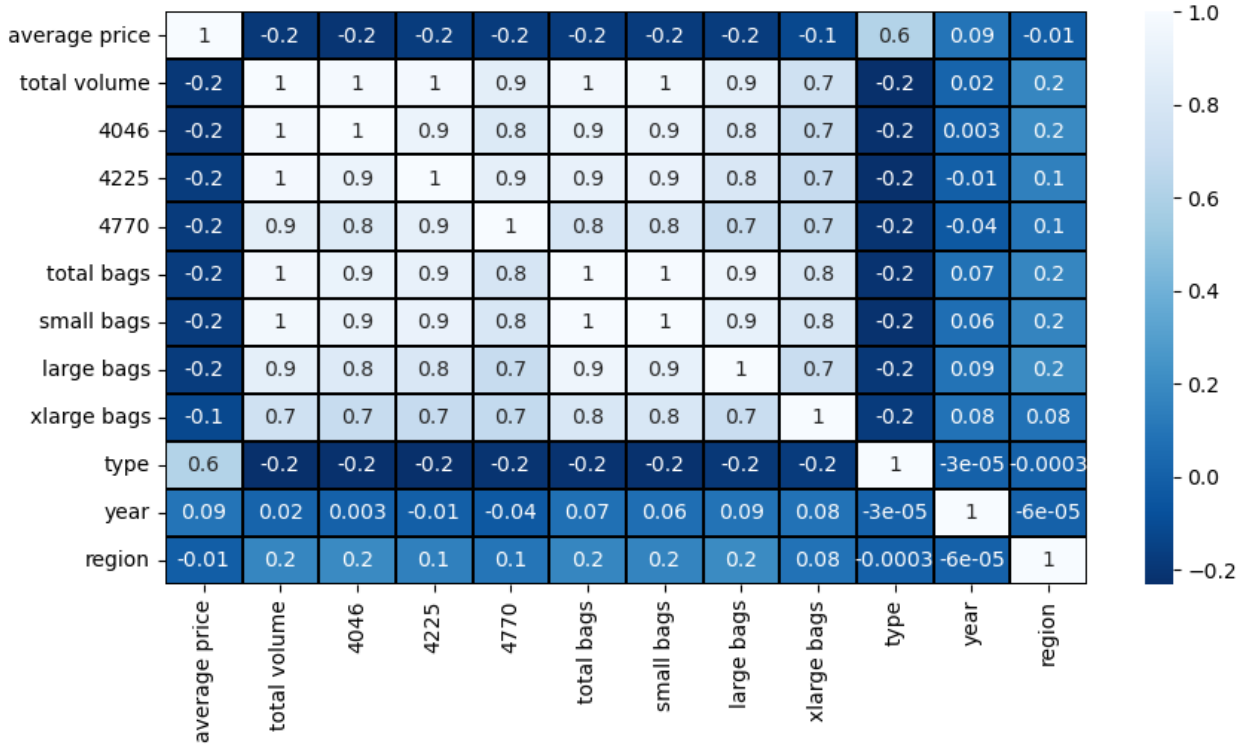
========== FINDING CORRELATION IN DATASET ==========

In [148]:
```
cor = df.corr()
cor

# non graphically finding correlation, here we can see that it is difficult to understand this
# ....so further we find the correlation graphically by HEAT MAP.
```

Out[148]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type
average price	1.000000	-0.192752	-0.208317	-0.172928	-0.179446	-0.177088	-0.174730	-0.172940	-0.117592	0.615845
total volume	-0.192752	1.000000	0.977863	0.974181	0.872202	0.963047	0.967238	0.880640	0.747157	-0.232434
4046	-0.208317	0.977863	1.000000	0.926110	0.833389	0.920057	0.925280	0.838645	0.699377	-0.225819
4225	-0.172928	0.974181	0.926110	1.000000	0.887855	0.905787	0.916031	0.810015	0.688809	-0.232289
4770	-0.179446	0.872202	0.833389	0.887855	1.000000	0.792314	0.802733	0.698471	0.679861	-0.210027
total bags	-0.177088	0.963047	0.920057	0.905787	0.792314	1.000000	0.994335	0.943009	0.804233	-0.217788
small bags	-0.174730	0.967238	0.925280	0.916031	0.802733	0.994335	1.000000	0.902589	0.806845	-0.220535
large bags	-0.172940	0.880640	0.838645	0.810015	0.698471	0.943009	0.902589	1.000000	0.710858	-0.193177
xlarge bags	-0.117592	0.747157	0.699377	0.688809	0.679861	0.804233	0.806845	0.710858	1.000000	-0.175483
type	0.615845	-0.232434	-0.225819	-0.232289	-0.210027	-0.217788	-0.220535	-0.193177	-0.175483	1.000000
year	0.093197	0.017193	0.003353	-0.009559	-0.036531	0.071552	0.063915	0.087891	0.081033	-0.000032
region	-0.011716	0.174176	0.192073	0.145726	0.095252	0.175256	0.164702	0.198768	0.082281	-0.000280

In [149]:
```
plt.figure (figsize = (10,5), facecolor = "white")
sns.heatmap(df.corr(),linewidth=0.1,fmt="0.1g",linecolor="black",annot=True,cmap="Blues_r")
plt.yticks(rotation=0);
plt.show()


#  Here in the following heat map , we can clearly see the correlation.
# here the columns- 'total volume' is highly correlated with '4046'
#                   'total bags' is higly correlated with 'small bags'
#                    'small bag' & large bags are also highly correlated with each other.
```



In [ ]:
```
```

SITUATION-1 (where target column is AVERAGE PRICE) ============

In [150]:
```
cor['average price'].sort_values(ascending=False)

#  here we can see with 'average price' the highest correlated column is - 'type'
```

Out[150]:
```
average price    1.000000
type             0.615845
year             0.093197
region          -0.011716
xlarge bags     -0.117592
4225            -0.172928
large bags      -0.172940
small bags      -0.174730
total bags      -0.177088
4770            -0.179446
total volume    -0.192752
4046            -0.208317
Name: average price, dtype: float64
```

In [ ]:

SITUATION - 2 (where target column is REGION)================

In [151]:
```
cor['region'].sort_values(ascending=False)
# here with the 'region column the HIGHEST CORRELATED COLUMN IS 'LARGE BAGS' & 4046
```

Out[151]:
```
region          1.000000
large bags      0.198768
4046            0.192073
total bags      0.175256
total volume    0.174176
small bags      0.164702
4225            0.145726
4770            0.095252
xlarge bags     0.082281
year           -0.000055
type           -0.000280
average price  -0.011716
Name: region, dtype: float64
```

In [ ]:

In [ ]:

CHECKING FOR OUTLIERS

==========

In [152]:
```
df.describe()
# here in the describe methode we are getting soo many STATISTICAL INFORMATION about the datase
# 1. first of all above we are getting 'count' for each of the column.
#                 as we know the total number of row counts for each column is 18,249. and here
#                 ... column is smame. not a single blank/'nan' is present in any of the columr

# 2. MEAN : In this, we can get MEAN VALUE for the every column.
# 3. STD : which is Standard Deviation , which shows that how the data of the column is deviate
# 4. MIN : It shows the Minimum value present in the column.
# 5. 25% : It gives us the 25th Percentile Value in the column.
# 6. 50% : It gives us the 50th Percentile Value in the column.
# 7. 75% : It gives us the 75th Percentile Value in the column.
# 8. Max : It gives us the MAXIMUM VALUE present the column.

# As If in any column the Difference between the value at 75th Percentile & MAX is Higher then,
# so we have to check the 75th% & MAX for each of the column.
#  here we find that in the following columns there is huge difference betweeen 75% & MAX :
#  1)- 4771, 2)- small bags, 3)-large bags, 4)- xlarge bags
# so in the above mentioned columns there may be presence of outliers, but we have to check al
```

Out[152]:

	average price	total volume	4046	4225	4770	total bags	small bags	large l
count	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e
mean	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+05	1.821947e+05	5.433809e
std	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+05	7.461785e+05	2.439660e
min	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e
25%	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+03	2.849420e+03	1.274700e
50%	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+04	2.636282e+04	2.647710e
75%	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+05	8.333767e+04	2.202925e
max	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+07	1.338459e+07	5.719097e

In [153]:
```
# By using BOXPLOT METHOD we can chek PRESENCE OF OUTLIERS  in every column.
```

In [154]:
```
df.columns
```

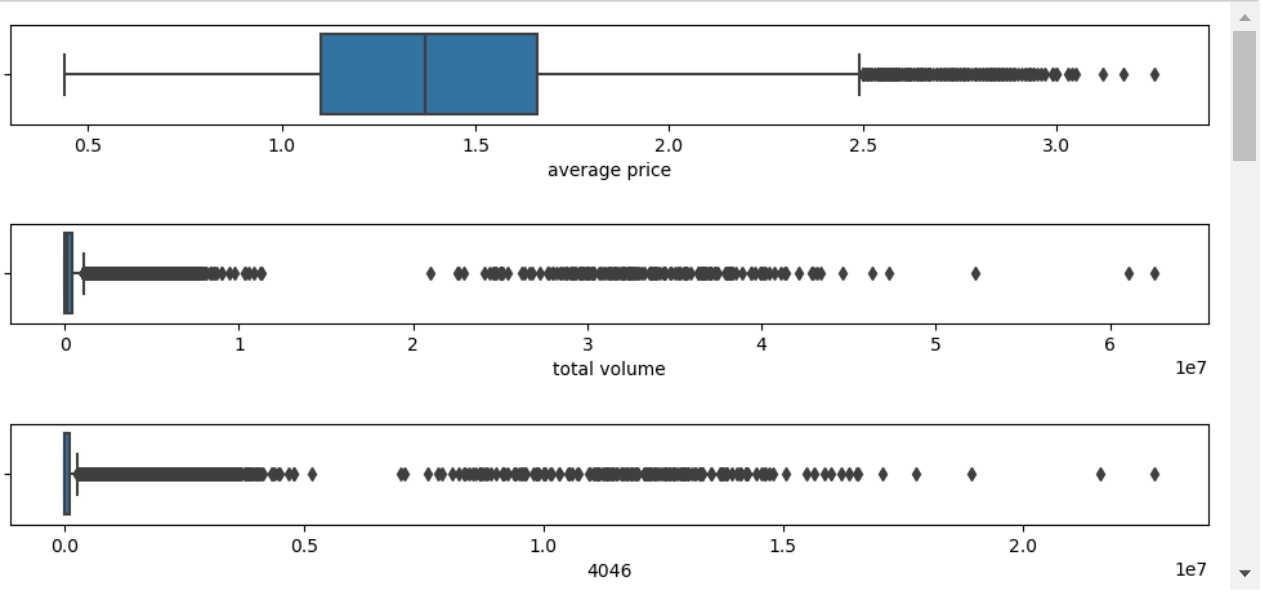
Out[154]:
```
Index(['date', 'average price', 'total volume', '4046', '4225', '4770',
       'total bags', 'small bags', 'large bags', 'xlarge bags', 'type', 'year',
       'region'],
      dtype='object')
```

In [155]:
```
df1 = df[['average price', 'total volume', '4046', '4225', '4770',
       'total bags', 'small bags', 'large bags', 'xlarge bags', 'type', 'year',
       'region']]
# here we are leaving 'date column' because boxplot allows only numeric column. and date columr
# therefore for just checking outliers we are making new dataframe 'df1' and apllying boxplot c
```

In [156]:
```
df.columns.nunique()
# there are 13 no. of unique columns are present in the dataset.
#  by using forloop we can chek outliers for each column in a single code.
```

Out[156]:
```
13
```

In [157]:
```
for i in df1.columns[0:13]:
    plt.figure (figsize = (12,1), facecolor = "white")
    sns.boxplot(x=i,data=df1)
    plt.show()

#  here below we can find the outliers for all the cloumns by using boxplot.
#  and we are found outliers in :
#  average price, total volume, 4046, 4225, 4770, total bag, small bags, large bage, x-large ba
# so out of 13 columns we found OUTLIERS IN 9 COLUMNS , now we have to remove those outliers fr
```



In [158]:
```
# As we given a situation in which we have to performe operations in 2 ways :
#  1st) we have 'Region' as our Target Column.
#  2nd) we have to take 'Average price' as a our Target Column.

#  we are making two different new datasets df_new1 & df_new2 in thoose datasets we will perfor
# ...in df_new1 we will take 'Region' as our TARGET VARIABLE ['CONDITION-1']
# ...in df_new2 we will take 'Average Price' as our TARGET VARIABLE.['CONDITION-2']
```

In [ ]:
```

```

CONDITION-1 ============== (Target column = Region)

In [159]:
```
df_new_01 = df[['average price', 'total volume', '4046', '4225', '4770',
       'total bags', 'small bags', 'large bags', 'xlarge bags', 'type', 'year','region']]

# here in df_new1 dataset, we are taking 'region' also which is our TARGET COLUMN, because here
# ... but we don't need to perform the following operation on our TARGET COLUMN.
```

In [160]:
```
df_new_01.head(2)
```

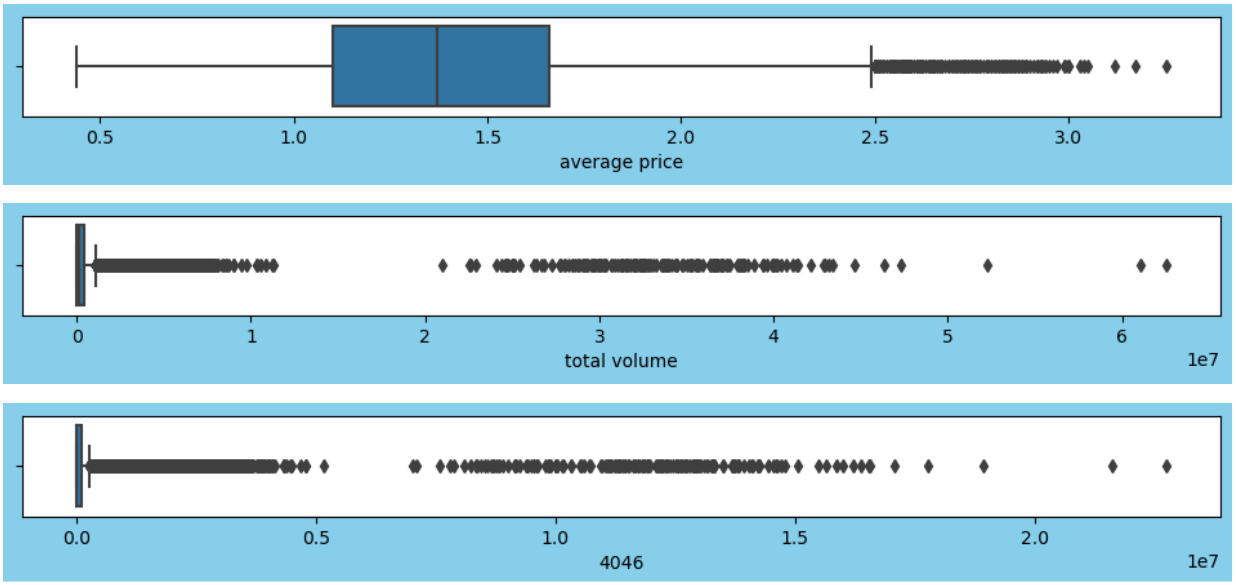
Out[160]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	2015	0
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	2015	0

In [161]:
```
df_new_01.shape
```

Out[161]:
```
(18249, 12)
```

In [162]:
```
for i in df_new_01.columns[0:12]:
    plt.figure (figsize = (12,1), facecolor = "skyblue")
    sns.boxplot(x=i,data=df_new_01)
    plt.show()

#  here below we can find the outliers for all the cloumns by using boxplot.
#  and we are found outliers in :
#  average price, total volume, 4046, 4225, 4770, total bag, small bags, large bage, x-large ba
# so out of 13 columns we found OUTLIERS IN 9 COLUMNS , now we have to remove those outliers fr
```



In [163]:
```
# here in CONDITION 1, we are having 'REGION' as our TARGET COLUMN. so we can't REMOVE OUTLERS
```

========== REMOVING OF OUTLIERS BY USING Z-SCORE METHOD (for condition-1)
====================

In [164]:
```
#  we can not remove outliers from out TARGET COLUMN, so first we have to seprate target columm
# For this first we need to identify the ZSCORE VALUES, for which we have to import some librar
```

In [165]:
```
from scipy.stats import zscore
```

In [166]:
```
z = np.abs(zscore(df_new_01))
z.head(5)

# by applying 'abs' (absolute method), we are getting all the entries whose z-score value is po
#  Ideally we can call the OUTLIERS whos ZSCORE VALUE is LESS THEN 3 AND MORE THEN 3
# so we have to remove all the data whose ZSCORE >3 & <3
#  below here we apllying "abs" i.e absolute method it returns us the all zscore values greater
#  so we just need to remove lesserr then 3 zscore values.
```

Out[166]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	re
0	0.188689	0.227716	0.230816	0.199902	0.212091	0.234170	0.232647	0.222352	0.17558	0.999836	1.221282	1.70
1	0.139020	0.230427	0.231103	0.208054	0.211997	0.233350	0.231568	0.222335	0.17558	0.999836	1.221282	1.70
2	1.182069	0.212085	0.231007	0.154478	0.211325	0.234730	0.233399	0.222311	0.17558	0.999836	1.221282	1.70
3	0.809551	0.223444	0.230741	0.185350	0.211864	0.237096	0.236568	0.222186	0.17558	0.999836	1.221282	1.70
4	0.312861	0.231538	0.230891	0.208719	0.211834	0.236718	0.236154	0.221924	0.17558	0.999836	1.221282	1.70

In [167]:
```
threshold = 3
print(np.where(z>3))
```

```
(array([  346,   359,   780, ..., 17304, 17402, 17428], dtype=int64), array([2, 2, 8, ..., 0, 0, 0], dtype=int64))
```

In [168]:
```
# here above we found only 598 outliers,  whose z-score is more then > 3
# i.e means we are having 598 outlier still present in our dataset, and we have to remove those
```

In [169]:
```
df_condition_01 = df_new_01[(z<3).all(axis=1)]
df_condition_01

# here we can see the difference clearly that, earlier there was 18,249 total rows are there, a
# ...there are only 17,651 rows are present in our dataset.
# so there are  598 OUTLIERS are removed during this process.
```

Out[169]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	2015	0
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	2015	0
2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	2015	0
3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	2015	0
4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	2015	0
...	...	...	...	...	...	...	...	...	...	...	...	...
18244	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	1	2018	53
18245	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	2018	53
18246	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	2018	53
18247	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	2018	53
18248	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	2018	53

17651 rows × 12 columns

In [170]:
```
df_condition_01.shape
```

Out[170]:
```
(17651, 12)
```

In [171]:
```
df_new_01.shape
```

Out[171]:
```
(18249, 12)
```

In [172]:
```
#  here above we can clearly see the difference, 598 outliers are removed from the new dataset
```

==========CHECKING REMOVAL OF OUTLIERS BY BOXPLOT (COMPARING 'df_new_01' & 'df_condition_01') ==========

In [173]:
```
df_condition_01.columns
```

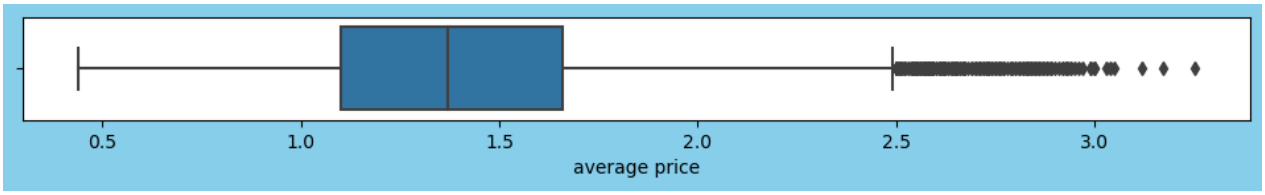
Out[173]:
```
Index(['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region'],
      dtype='object')
```

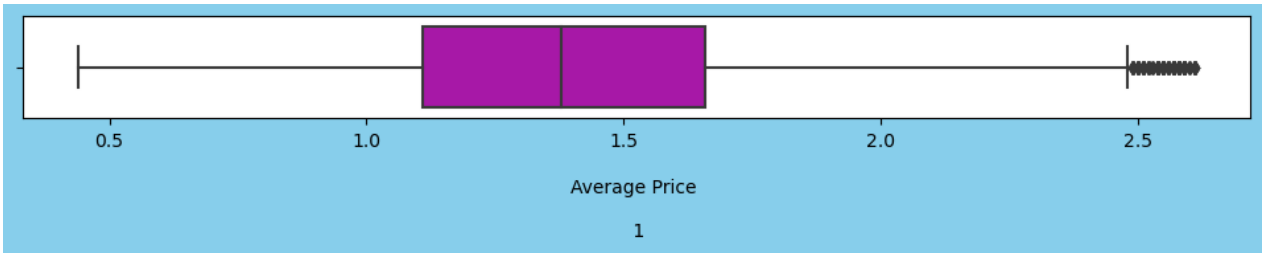
In [174]:
```
df_condition_01.columns.nunique()
```

Out[174]: 12

In [175]:
```
# now in the earlier findings we found that there are may presence of outliers in some of the c
# ....i.e (average price, total volume, 4046, 4225, 4770, total bag, small bags, large bage, x
#  now we have to check out of these columns which outlier hasbeen removed from the dataset.
```

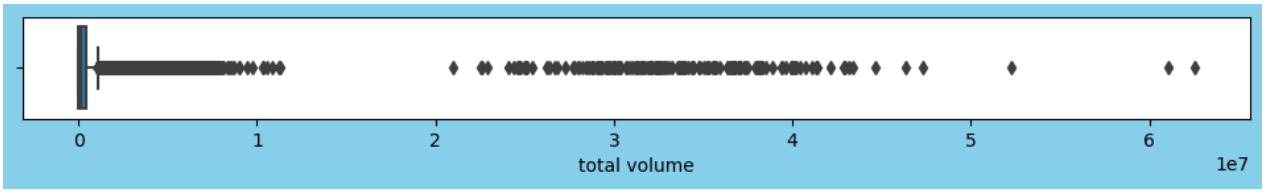
In [176]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='average price',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='average price',data=df_condition_01, color='m')
plt.xlabel('\nAverage Price \n\n 1')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'AVERAGE PRICE' c
```

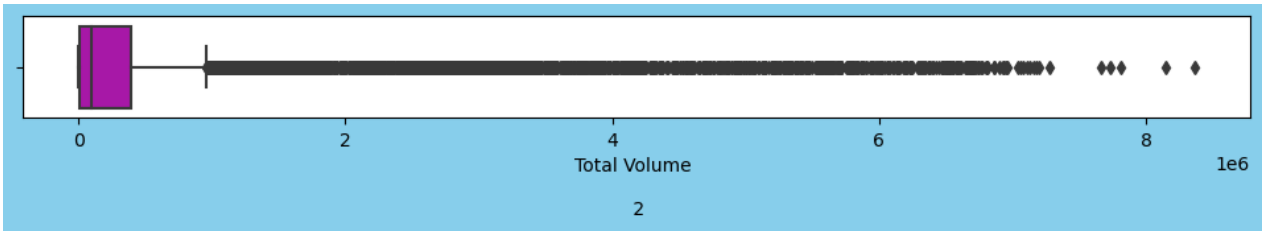




In [ ]:

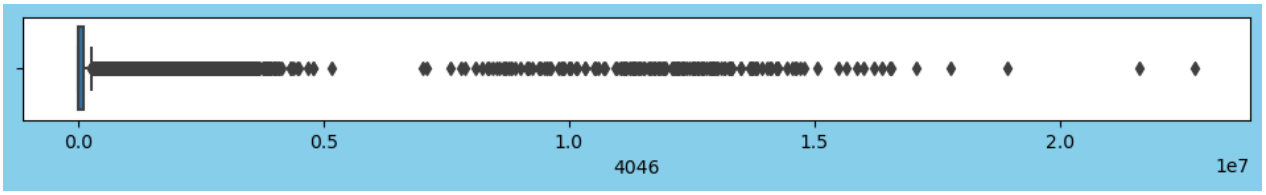
In [177]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='total volume',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='total volume',data=df_condition_01, color='m')
plt.xlabel('Total Volume \n\n 2')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'TOTAL VOLUME' co
```

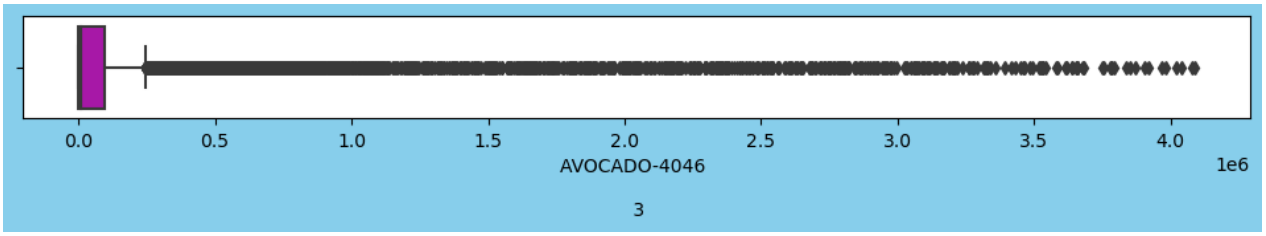




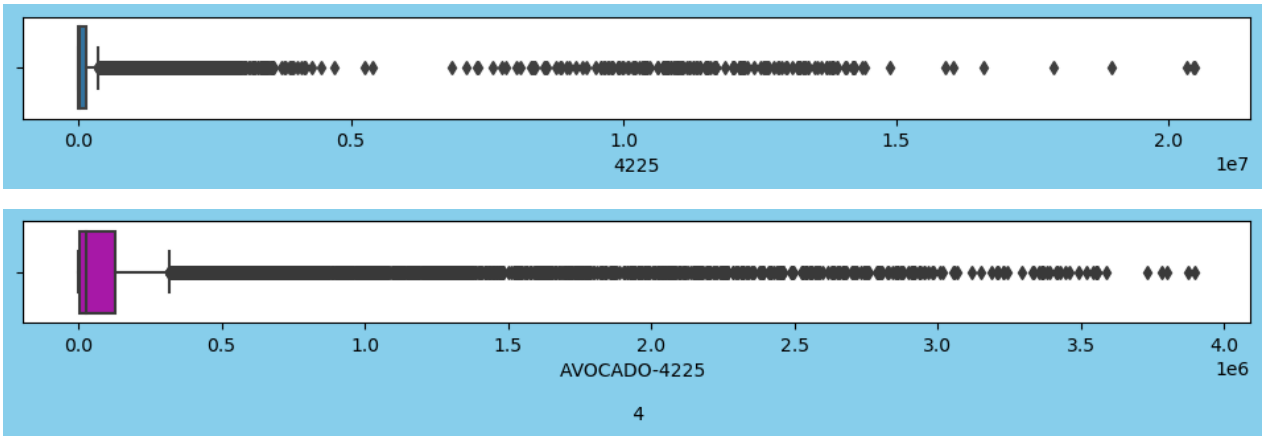
In [ ]:
```
```

In [178]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4046',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4046',data=df_condition_01, color='m')
plt.xlabel('AVOCADO-4046 \n\n 3')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from '4046' column.
```

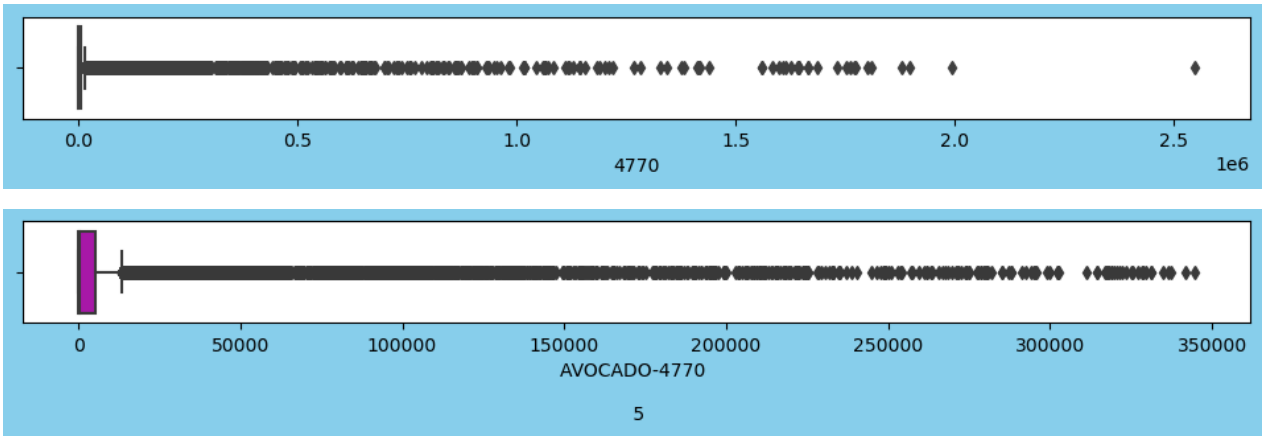




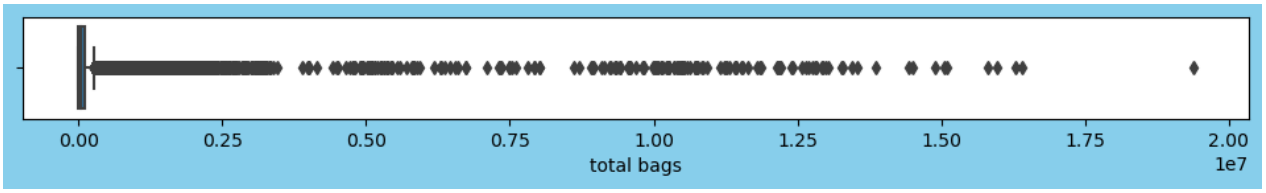
In [179]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4225',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4225',data=df_condition_01, color='m')
plt.xlabel('AVOCADO-4225 \n\n 4')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from '4225' column.
```

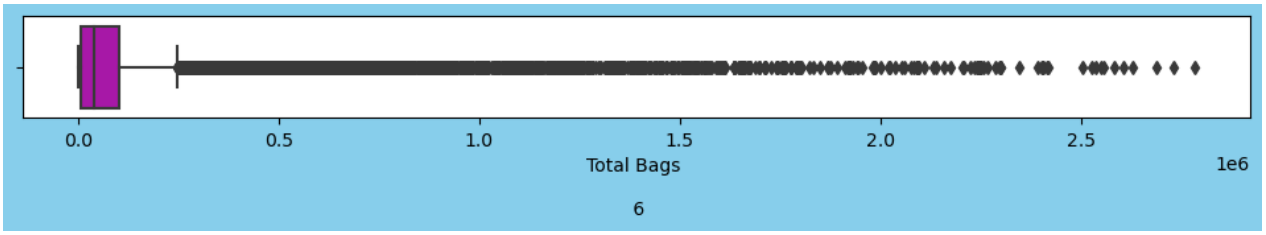


In [180]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4770',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='4770',data=df_condition_01, color='m')
plt.xlabel('AVOCADO-4770 \n\n 5')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from '4770' column.
```

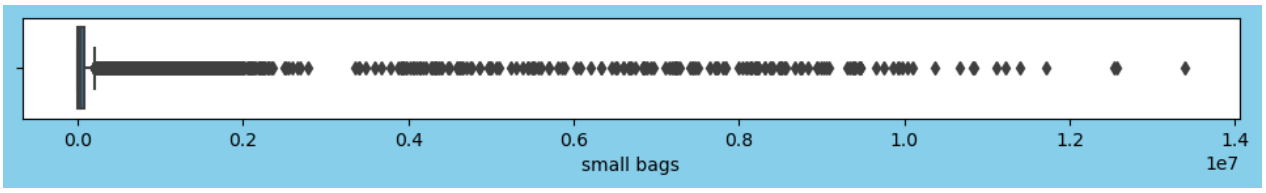


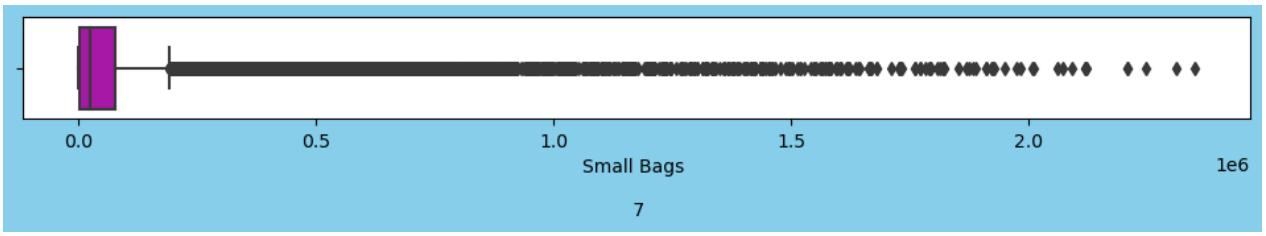
In [181]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='total bags',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='total bags',data=df_condition_01, color='m')
plt.xlabel('Total Bags \n\n 6')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'TOTAL BAGS' colu
```



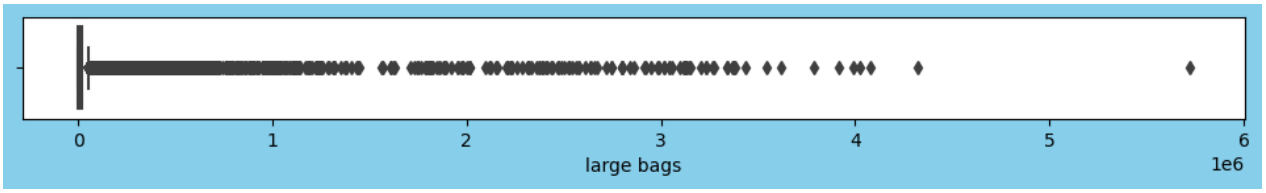


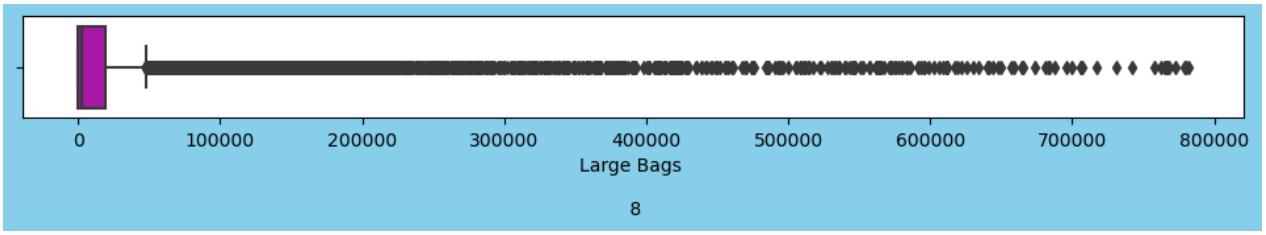
In [182]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='small bags',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='small bags',data=df_condition_01, color='m')
plt.xlabel('Small Bags \n\n 7')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'SMALL BAGS' colu
```



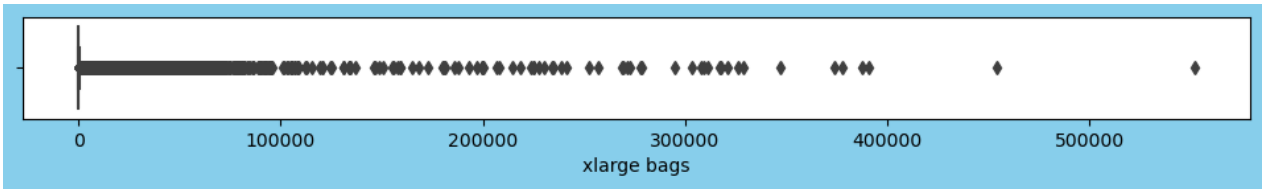


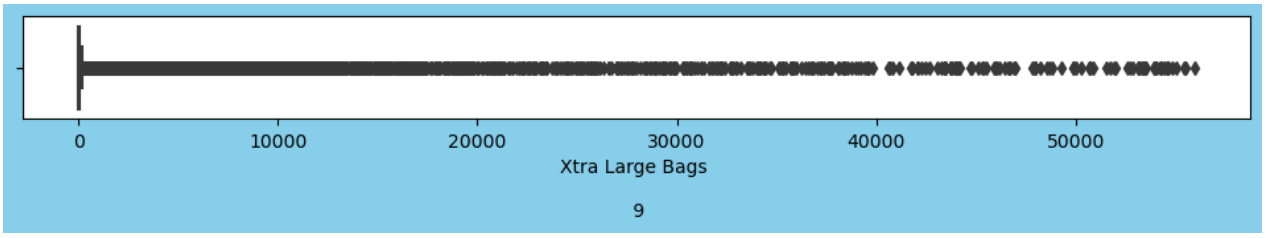
In [183]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='large bags',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='large bags',data=df_condition_01, color='m')
plt.xlabel('Large Bags \n\n 8')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'LARGE BAGS' colu
```





In [184]:
```
plt.figure (figsize = (12,1), facecolor = "skyblue")
# sns.set(style="darkgrid")
sns.boxplot(x='xlarge bags',data=df_new_01)
plt.show()

# it is the EARLIER (df_new1 dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "skyblue")
sns.boxplot(x='xlarge bags',data=df_condition_01, color='m')
plt.xlabel('Xtra Large Bags \n\n 9')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_condition1 dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'X-LARGE BAGS' co
```





In [185]:
```
#  here above we have sucessfully removed OutLiers from the Mentioned 9 Columns
```

In [ ]:
```
```

In [186]:
```
df_condition_01.shape

# TARGET COLUMN IS 'REGION'
```

Out[186]:
```
(17651, 12)
```

========== FINDING & REMOVAL OF OUTLIERS ARE COMPLETED ==========

In [ ]:
```
```

CHECKING SKEWNESS ==========

In [187]:
```
# the skewness shows the distribution of data, if the data is widely skewed that means it is no
#  ideal range of skewness is ( -0.5 to +0.5)
# We can't remove skewness from our Target Column
```

FOR SITUATION -1 ==========>>>>>

In [188]:
```
df_condition_01.skew()

# here below w find that highesht skewd columns are - 'total volume''total bags' 'xlarge/large/
# first fo all we have to remove skewness form the 'xlarge bags' & other colunm, then we can ch
```

Out[188]:
```
average price    0.377432
total volume     3.767138
4046             4.909848
4225             4.455745
4770             5.117170
total bags       4.066771
small bags       4.222706
large bags       5.053434
xlarge bags      6.135607
type            -0.037741
year             0.229976
region           0.012798
dtype: float64
```

In [189]:
```
# the skewness shows the distribution of data, if the data is widely skewed that means it is no
#  ideal range of skewness is ( -0.5 to +0.5)
# We can't remove skewness from our Target Column
# here we can see the skewness is present in 'total volume''total bags' 'xlarge/large/small bag
#  so we need to remove skewness from those mentioned columns.
```

In [190]:
```
# so we have to remove skewness from those columns by using 'cuberoot' method.
```

In [191]:
```
df_condition_01['xlarge bags'] = np.cbrt(df_condition_01['xlarge bags'])
```

In [192]:
```
df_condition_01['large bags'] = np.cbrt(df_condition_01['large bags'])
```

In [193]:
```
df_condition_01['small bags'] = np.cbrt(df_condition_01['small bags'])
```

In [194]:
```
df_condition_01['4770'] = np.cbrt(df_condition_01['4770'])
```

In [195]:
```
df_condition_01['4225'] = np.cbrt(df_condition_01['4225'])
```

In [196]:
```
df_condition_01['4046'] = np.cbrt(df_condition_01['4046'])
```

In [197]:
```
df_condition_01['total volume'] = np.cbrt(df_condition_01['total volume'])
```

In [198]:
```
df_condition_01['total bags'] = np.cbrt(df_condition_01['total bags'])
```

In [199]:
```
df_condition_01.skew()
# here in the following we can see that the skewness not in the ideal condition but it is much
```

Out[199]:
```
average price    0.377432
total volume     1.240435
4046             1.555876
4225             1.277025
4770             1.635614
total bags       1.231831
small bags       1.246033
large bags       1.276213
xlarge bags      2.277028
type            -0.037741
year             0.229976
region           0.012798
dtype: float64
```

In [ ]:
```

```

In [200]:
```
#  here above we can see that the skewness for both of the situations is reduced.
```

========== REMOVED SKEWNESS SUCCESSFULLY
==========

In [ ]:
```

```

DIVIDING DATA INTO INDEPENDENT & TARGET VARIABLE
==========

In [201]:
```
#  CONDITION-1  (Where Target Column is =  'REGION' )
```

In [202]:
```
df_condition_01.shape
```

Out[202]:
```
(17651, 12)
```

In [203]:
```
df_condition_01.columns
```

Out[203]:
```
Index(['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region'],
      dtype='object')
```

In [204]:
```
# here we are giving with two situations :
#  1) SITUATION 1 = where Target column is = 'Average Price'
#  2) SITUATION 2 = where target column is = 'region'

# so we are going to make 2 different datasets for SITUATION 1 & 2 , with the name of x1-y1, &
#  in x1-y1 ( target column is - AVERAGE PRICE)
#  in x2-y2 ( target column is - REGION)
```

SITUATION-1 === (where target column is - AVERAGE PRICE)

In [205]:
```
x1 = df_condition_01[['total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region']]
```

In [206]:
```
y1 = df_condition_01[['average price']]
```

In [207]:
```
x1.head(2)
```

Out[207]:

	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	region
0	40.049235	10.120997	37.903460	3.638275	20.564634	20.490870	4.534711	0.0	0	2015	0
1	38.001150	8.768933	35.473513	3.878204	21.183249	21.110581	4.602425	0.0	0	2015	0

In [208]:
```
y1.head(2)
```

Out[208]:

	average price
0	1.33
1	1.35

In [209]:
```
x1.shape
```

Out[209]: (17651, 11)

In [210]:
```
y1.shape
```

Out[210]: (17651, 1)

In [ ]:
```

```

SITUATION -2 ======== (where target column is = REGION)

In [211]:
```
df_condition_01.shape
```

Out[211]: (17651, 12)

In [212]:
```
df_condition_01.columns
```

Out[212]:
```
Index(['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region'],
      dtype='object')
```

In [213]:
```
x2 = df_condition_01[['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year']]
```

In [214]:
```
y2 = df_condition_01[['region']]
```

In [215]:
```
x2.head(2)
```

Out[215]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year
0	1.33	40.049235	10.120997	37.903460	3.638275	20.564634	20.490870	4.534711	0.0	0	2015
1	1.35	38.001150	8.768933	35.473513	3.878204	21.183249	21.110581	4.602425	0.0	0	2015

In [216]:
```
y2.head(2)
```

Out[216]:

	region
0	0
1	0

In [217]:
```
x2.shape
```

Out[217]:
```
(17651, 11)
```

In [218]:
```
y2.shape
```

Out[218]:
```
(17651, 1)
```

In [219]:
```
# here above we seprates independent and dependent columns successfully.
```

APPLYING SCALING TECHNIQUES
==========

In [220]:
```
# here we need to apply scaling techniques on our dataset,by scaling techniques we normalise th
#  we can't apply SCALING TECHNIQUES on TARGET VARIABLE
#  to aplly scaling techinuque we need to import some libraries first.
```

In [221]:
```
from sklearn.preprocessing import StandardScaler
```

In [222]:
```
st = StandardScaler()
```

SITUATION-1 ====== (where target column is 'AVERAGE PRICE') ==========

In [223]:
```
x1 = st.fit_transform(x1)
x1
```

Out[223]:
```
array([[-0.36001678, -0.7096478 ,  0.05964226, ..., -1.01904715,
        -1.21001338, -1.69549505],
       [-0.41358063, -0.75506378, -0.0254953 , ..., -1.01904715,
        -1.21001338, -1.69549505],
       [-0.12385558, -0.73848038,  0.40604959, ..., -1.01904715,
        -1.21001338, -1.69549505],
       ...,
       [-0.78062205, -0.69346796, -0.79586353, ...,  0.98130886,
         1.98075592,  1.74705401],
       [-0.74560645, -0.66275586, -0.76412245, ...,  0.98130886,
         1.98075592,  1.74705401],
       [-0.72856457, -0.57089182, -0.80215382, ...,  0.98130886,
         1.98075592,  1.74705401]])
```

In [224]:
```
xf1 = pd.DataFrame(data=x1)
print(xf1)

# here we get our dataset (xf1) after applying SCALING TECHING (STANDARD SCALER)
```

```
             0         1         2         3         4         5         6   \
0     -0.360017 -0.709648  0.059642 -0.533970 -0.648548 -0.517773 -0.796802
1     -0.413581 -0.755064 -0.025495 -0.516157 -0.623078 -0.490834 -0.792817
2     -0.123856 -0.738480  0.406050 -0.427502 -0.666838 -0.537584 -0.787683
3     -0.285278 -0.699539  0.189057 -0.494395 -0.755024 -0.633026 -0.762704
4     -0.437308 -0.720397 -0.032969 -0.489909 -0.739523 -0.619211 -0.720851
...         ...       ...       ...       ...       ...       ...       ...
17646 -0.733974 -0.623118 -0.864717 -0.804096 -0.514919 -0.384638 -0.618863
17647 -0.778787 -0.693490 -0.739905 -0.804096 -0.630504 -0.506311 -0.659154
17648 -0.780622 -0.693468 -0.795864 -0.136211 -0.626500 -0.492667 -0.858596
17649 -0.745606 -0.662756 -0.764122 -0.136496 -0.580423 -0.444110 -0.846859
17650 -0.728565 -0.570892 -0.802154 -0.352835 -0.552260 -0.413625 -0.889294

            7         8         9         10
0     -0.51836 -1.019047 -1.210013 -1.695495
1     -0.51836 -1.019047 -1.210013 -1.695495
2     -0.51836 -1.019047 -1.210013 -1.695495
3     -0.51836 -1.019047 -1.210013 -1.695495
4     -0.51836 -1.019047 -1.210013 -1.695495
...        ...       ...       ...       ...
17646 -0.51836  0.981309  1.980756  1.747054
17647 -0.51836  0.981309  1.980756  1.747054
17648 -0.51836  0.981309  1.980756  1.747054
17649 -0.51836  0.981309  1.980756  1.747054
17650 -0.51836  0.981309  1.980756  1.747054

[17651 rows x 11 columns]
```

In [225]:
```
xf1.columns
```

Out[225]:
```
RangeIndex(start=0, stop=11, step=1)
```

In [226]:
```
df_condition_01.columns
```

Out[226]:
```
Index(['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region'],
      dtype='object')
```

In [227]:
```
column1 = ['total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region']
```

In [228]:
```
xf1.columns = column1
```

In [229]:
```
xf1.head(2)
```

Out[229]:

	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	year	regio
0	-0.360017	-0.709648	0.059642	-0.533970	-0.648548	-0.517773	-0.796802	-0.51836	-1.019047	-1.210013	-1.69549
1	-0.413581	-0.755064	-0.025495	-0.516157	-0.623078	-0.490834	-0.792817	-0.51836	-1.019047	-1.210013	-1.69549

In [230]:
```
yf1 = y1
```

In [231]:
```
yf1.head(2)
```

Out[231]:

	average price
0	1.33
1	1.35

In [ ]:
```

```

SITUATION-2 ==== (where target column is REGION)==========================

In [232]:
```
x2 = st.fit_transform(x2)
x2
```

Out[232]:
```
array([[-0.19535234, -0.36001678, -0.7096478 , ..., -0.51835993,
        -1.01904715, -1.21001338],
       [-0.1435096 , -0.41358063, -0.75506378, ..., -0.51835993,
        -1.01904715, -1.21001338],
       [-1.2322071 , -0.12385558, -0.73848038, ..., -0.51835993,
        -1.01904715, -1.21001338],
       ...,
       [ 1.20440158, -0.78062205, -0.69346796, ..., -0.51835993,
         0.98130886,  1.98075592],
       [ 1.35992979, -0.74560645, -0.66275586, ..., -0.51835993,
         0.98130886,  1.98075592],
       [ 0.55636736, -0.72856457, -0.57089182, ..., -0.51835993,
         0.98130886,  1.98075592]])
```

In [233]:
```
xf2 = pd.DataFrame(data=x2)
print(xf2)

# here we get our dataset (xf2) after applying SCALING TECHING (STANDARD SCALER)
```

```
             0         1         2         3         4         5         6   \
0     -0.195352 -0.360017 -0.709648  0.059642 -0.533970 -0.648548 -0.517773
1     -0.143510 -0.413581 -0.755064 -0.025495 -0.516157 -0.623078 -0.490834
2     -1.232207 -0.123856 -0.738480  0.406050 -0.427502 -0.666838 -0.537584
3     -0.843387 -0.285278 -0.699539  0.189057 -0.494395 -0.755024 -0.633026
4     -0.324959 -0.437308 -0.720397 -0.032969 -0.489909 -0.739523 -0.619211
...         ...       ...       ...       ...       ...       ...       ...
17646  0.582289 -0.733974 -0.623118 -0.864717 -0.804096 -0.514919 -0.384638
17647  0.789660 -0.778787 -0.693490 -0.739905 -0.804096 -0.630504 -0.506311
17648  1.204402 -0.780622 -0.693468 -0.795864 -0.136211 -0.626500 -0.492667
17649  1.359930 -0.745606 -0.662756 -0.764122 -0.136496 -0.580423 -0.444110
17650  0.556367 -0.728565 -0.570892 -0.802154 -0.352835 -0.552260 -0.413625

             7        8         9         10
0     -0.796802 -0.51836 -1.019047 -1.210013
1     -0.792817 -0.51836 -1.019047 -1.210013
2     -0.787683 -0.51836 -1.019047 -1.210013
3     -0.762704 -0.51836 -1.019047 -1.210013
4     -0.720851 -0.51836 -1.019047 -1.210013
...         ...      ...       ...       ...
17646 -0.618863 -0.51836  0.981309  1.980756
17647 -0.659154 -0.51836  0.981309  1.980756
17648 -0.858596 -0.51836  0.981309  1.980756
17649 -0.846859 -0.51836  0.981309  1.980756
17650 -0.889294 -0.51836  0.981309  1.980756

[17651 rows x 11 columns]
```

In [234]:
```
xf2.columns
```

Out[234]:
```
RangeIndex(start=0, stop=11, step=1)
```

In [235]:
```
df_condition_01.columns
```

Out[235]:
```
Index(['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year', 'region'],
      dtype='object')
```

In [236]:
```
column2 = ['average price', 'total volume', '4046', '4225', '4770', 'total bags',
       'small bags', 'large bags', 'xlarge bags', 'type', 'year']
```

In [237]:
```
xf2.columns = column2
```

In [238]:
```
xf2.head(2)
```

Out[238]:

	average price	total volume	4046	4225	4770	total bags	small bags	large bags	xlarge bags	type	yea
0	-0.195352	-0.360017	-0.709648	0.059642	-0.533970	-0.648548	-0.517773	-0.796802	-0.51836	-1.019047	-1.21001
1	-0.143510	-0.413581	-0.755064	-0.025495	-0.516157	-0.623078	-0.490834	-0.792817	-0.51836	-1.019047	-1.21001

In [239]:
```
yf2 = y2
```

In [240]:
```
yf2.head(2)
```

Out[240]:

	region
0	0
1	0

In [ ]:
```

```

FINDING MULTICOLINEARITY
==========

In [241]:
```
# We have to find the multicollinearity between the features and to remove it we can use VIF (V
# we can not apply VIF on the TARGET COLUMN
# for apllyin VIF we have to import some libraries as follows
```

In [242]:
```
import statsmodels.api as sm
from scipy import stats
from statsmodels .stats.outliers_influence import variance_inflation_factor
```

FINDING MULTICOLINEARITY IN SITUATION-1 ========== (where 'AVERAGE PRICE' is target column==========

In [243]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf1):
    vif = pd.DataFrame()
    vif["FETURES"] = xf1.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf1.values,i) for i in range (xf1.shape[1])
    return (vif)
```

In [244]:
```
xf1.shape
```

Out[244]: (17651, 11)

In [245]:
```
yf1.shape
```

Out[245]: (17651, 1)

In [246]:
```
calc_vif(xf1)
# here we find that VIF VALUE of 'TOTAL VOLUME ' & TOTAL BAGS is very high.
#  they are Highly MULTICOLLINERED COLUMNS. we have to drop one of the column between them.
#
```

Out[246]:

	FETURES	VIF FACTOR
0	total volume	202.503330
1	4046	24.866826
2	4225	40.464243
3	4770	3.301017
4	total bags	117.508648
5	small bags	63.622958
6	large bags	9.031383
7	xlarge bags	1.968262
8	type	2.182953
9	year	1.317551
10	region	1.114124

In [247]:
```
cor['average price'].sort_values(ascending=False)
# here we can see that the 'total volume column ' & 'total bags' are not much correlated with t
# So we can drop 'total volume' & 'total bags' column
```

Out[247]:
```
average price    1.000000
type             0.615845
year             0.093197
region          -0.011716
xlarge bags     -0.117592
4225            -0.172928
large bags      -0.172940
small bags      -0.174730
total bags      -0.177088
4770            -0.179446
total volume    -0.192752
4046            -0.208317
Name: average price, dtype: float64
```

In [248]:
```
xf1.drop(['total volume'],axis=1,inplace=True)
```

In [249]:
```
xf1.drop(['total bags'],axis=1,inplace=True)
```

In [250]:
```
calc_vif(xf1)
# here now multicolinearity is very much reduced as compared to earlier
```

Out[250]:

	FETURES	VIF FACTOR
0	4046	4.143428
1	4225	5.477961
2	4770	3.269070
3	small bags	6.541084
4	large bags	2.116928
5	xlarge bags	1.934397
6	type	2.166444
7	year	1.311936
8	region	1.089408

In [ ]:

In [ ]:

In [ ]:

FINDING MULTICOLINEARITY IN SITUATION-2 ================== (where 'REGION' is target column)================

In [251]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf2):
    vif = pd.DataFrame()
    vif["FETURES"] = xf2.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf2.values,i) for i in range (xf2.shape[1])
    return (vif)
```

In [252]:
```
calc_vif(xf2)
# here we find that VIF VALUE of 'TOTAL VOLUME ' & TOTAL BAGS is very high.
#  they are Highly MULTICOLLINERED COLUMNS. we have to drop one of the column between them.
```

Out[252]:

	FETURES	VIF FACTOR
0	average price	2.018698
1	total volume	203.647267
2	4046	23.961766
3	4225	42.078549
4	4770	3.284587
5	total bags	120.152468
6	small bags	66.701023
7	large bags	9.113256
8	xlarge bags	1.975705
9	type	2.696459
10	year	1.352954

In [253]:
```
cor['region'].sort_values(ascending=False)
# here we can see that the 'total volume column ' & 'total bags' are not much correlated with t
# So we can drop 'total volume' & 'total bags' column
```

Out[253]:
```
region           1.000000
large bags       0.198768
4046             0.192073
total bags       0.175256
total volume     0.174176
small bags       0.164702
4225             0.145726
4770             0.095252
xlarge bags      0.082281
year            -0.000055
type            -0.000280
average price   -0.011716
Name: region, dtype: float64
```

In [254]:
```
xf2.drop(['total volume'],axis=1,inplace=True)
xf2.drop(['total bags'],axis=1,inplace=True)
```

In [255]:
```
calc_vif(xf2)
# now here we did't find any higher multicolinearity .
```

Out[255]:

	FETURES	VIF FACTOR
0	average price	1.856196
1	4046	4.063847
2	4225	5.648659
3	4770	3.249044
4	small bags	6.545230
5	large bags	2.215059
6	xlarge bags	1.938176
7	type	2.671380
8	year	1.339645

In [256]:
```
xf1.shape
```

Out[256]: (17651, 9)

In [257]:
```
yf1.shape
```

Out[257]: (17651, 1)

In [258]:
```
#  xf1 & yf1 is dataset where the target column is - AVERAGE PRICE
```

In [259]:
```
xf2.shape
```

Out[259]: (17651, 9)

In [260]:
```
yf2.shape
```

Out[260]: (17651, 1)

In [261]:
```
#  xf2 & yf2 is dataset where the target column is - REGION
```

========== Upto here EDA & OTHER TECHNIQUES are completed for both of SITUATION-1 & SITUATION-2 ==========

========== NOW WE NEED TO APPLY ML-MODELS FOR BOTH OF THE SITUATIONS ==========

In [ ]:
```
```

In [262]:
```
# Here in the given dataset we are provided with the 2 SITUATION, where...
#  in SITUATION-1 = We have to Predict 'AVERAGE PRICE' for AVOCADO, which is in DECIMAL FORM.
#                   so for situation-1 we have to aplly REGRESSION MODEL.

#  in SITUATION-2 = We have to predict the 'REGION' for the AVOCADO, which is a CATEGORICAL & C
#                   therefore for situation-2 we have to aplly CLASSIFICATION MODEL.

# the CONCLUSION IS THAT WE ARE MAKEING 2 DIFFERENT ML-MODELS, THEREFORE WE WOULD HAVE 2 DIFFE
```

APPLYING MODEL FOR SITUATION-1 (REGRESSION MODEL) ==========

In [263]:
```
xf1.head()
```

Out[263]:

	4046	4225	4770	small bags	large bags	xlarge bags	type	year	region
0	-0.709648	0.059642	-0.533970	-0.517773	-0.796802	-0.51836	-1.019047	-1.210013	-1.695495
1	-0.755064	-0.025495	-0.516157	-0.490834	-0.792817	-0.51836	-1.019047	-1.210013	-1.695495
2	-0.738480	0.406050	-0.427502	-0.537584	-0.787683	-0.51836	-1.019047	-1.210013	-1.695495
3	-0.699539	0.189057	-0.494395	-0.633026	-0.762704	-0.51836	-1.019047	-1.210013	-1.695495
4	-0.720397	-0.032969	-0.489909	-0.619211	-0.720851	-0.51836	-1.019047	-1.210013	-1.695495

In [264]:
```
yf1.head()
```

Out[264]:

	average price
0	1.33
1	1.35
2	0.93
3	1.08
4	1.28

In [265]:
```
# NOW HERE WE CAN SEE THAT OUR TARGET/LABEL COLUMN IN NOT A CATEGORICAL DATA, IT IS HAVING NUMI
# AND WHEN WE ARE HAVING "Y" (TARGET) IN NUMERICAL/DECIMAL FORM THEN WE CAN APPLY "REGRESSION M
# SO HERE WE CAN APPLY REGRESSION MODEL ON OUR DATASET TO PREDICT, "AVERAGE PRICE" for AVOCADO
```

In [266]:
```
#  LOGISTIC REGRESSION = here we can't aplly this logistic regression, because our target targe
#                        ...numeric categorical. we can aplly logistic regression where the tar

#  RIDGE REGRESSION = we can apply ridge regression wher the data is HIGHLY MULTICOLINEARED wi
#                     ....no such situation like this.

# LINEAR REGRESSION = here we are going to aplly LINEAR REGRESSION MODEL with the above situat
#           for this we need to import some libraries first.
```

In [267]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [268]:
```
lr = LinearRegression()
```

In [269]:
```
yf1.head(1)
```

Out[269]:

	average price
0	1.33

In [270]:
```
 x_train,x_test,y_train,y_test = train_test_split(xf1,yf1,test_size=0.20,random_state=42)
```

In [271]:
```
lr.fit(x_train,y_train)
print(lr.score(x_train,y_train))
pred_y = lr.predict (x_test)
pred_y
print("mean squared error:", np.sqrt(mean_squared_error(y_test,pred_y)))
```

```
0.46200710047406046
mean squared error: 0.28020957259038215
```

In [272]:
```
lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
y_test.head(),y_pred[0:5]

# here below we can find 5 actual and 5 predicted values.
```

Out[272]:
```
(       average price
 3596            1.36
 7141            1.70
 4625            1.22
 5456            0.81
 10969           1.85,
 array([[0.993422  ],
        [1.40137287],
        [1.22445309],
        [1.03190436],
        [1.58992131]]))
```

In [273]:
```
Accuracy=r2_score(y_test,y_pred)*100
print(" Accuracy of the model is %.2f" %Accuracy)
```

```
 Accuracy of the model is 46.85
```

In [274]:
```
mse = mean_squared_error (y_test,y_pred)
mse
# here our MEAN SQURED ERROR is very less, which is a very good indication for the model.
```

Out[274]: 0.07851740457128466

In [275]:
```
r2 = r2_score(y_test,y_pred)
r2
```

Out[275]: 0.46845561121038504

In [276]:
```
print(f'Intercept: {lr.intercept_[0]}')
print(f'Coefficient: {lr.coef_[0][0]}')
print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'R-squared (R2): {r2:.2f}')
```

```
Intercept: 1.4075941669671277
Coefficient: -0.08008733000220762
Mean Squared Error (MSE): 0.08
R-squared (R2): 0.47
```

In [277]:
```
# plt.scatter(y_test,y_pred);
# plt.plot(x_test, y_pred, color='red', linewidth=2, label='Regression Line')
# plt.xlabel('Actual');
# plt.ylabel('Predicted');
```

In [278]:
```
print("X shape:", x_test.shape)
print("y shape:", y_test.shape)
```

```
X shape: (3531, 9)
y shape: (3531, 1)
```

In [279]:
```
xf1.shape
```

Out[279]: (17651, 9)

In [280]:
```
def pred_func(ap):
    ap= ap.reshape(1,9)
    average_price = lr.predict(ap)
    print(average_price)

# making 'def' function to predict Average Price for Avocado.
```

In [281]:
```
# TEST SAMPLE-1
```

In [282]:
```
ap= np.array([-0.709648,0.059642,-0.533970,-0.517773,-0.796802,-0.51836,-1.019047,-1.210013,-1
pred_func(ap)

#  data taken from 0th position from 'xf1.head(5)'
# here the ATUAL PRICE FOR GIVEN DATA IS = 1.33
#       & THE PREDICTED PRICE BY MODEL IS = 1.26
```

```
[[1.25945713]]
```

In [283]:
```
# TEST SAMPLE-2
```

In [284]:
```
ap= np.array([-0.738480,0.406050,-0.427502,-0.537584,-0.787683,-0.51836,-1.019047,-1.210013,-1
pred_func(ap)

#  data taken from 2nd position from 'xf1.head(5)'
# here the ATUAL PRICE FOR GIVEN DATA IS = 0.93
#       & THE PREDICTED PRICE BY MODEL IS = 1.30
```

```
[[1.30014429]]
```

In [285]:
```
# TEST SAMPLE-3
```

In [286]:
```
ap= np.array([-0.720397,-0.032969,-0.489909,-0.619211,-0.720851,-0.51836,-1.019047,-1.210013,-
pred_func(ap)

#  data taken from 4th position from 'xf1.head(5)'
# here the ATUAL PRICE FOR GIVEN DATA IS = 1.28
#       & THE PREDICTED PRICE BY MODEL IS = 1.24
```

```
[[1.24458365]]
```

In [287]:
```
xf1.head(5)
```

Out[287]:

	4046	4225	4770	small bags	large bags	xlarge bags	type	year	region
0	-0.709648	0.059642	-0.533970	-0.517773	-0.796802	-0.51836	-1.019047	-1.210013	-1.695495
1	-0.755064	-0.025495	-0.516157	-0.490834	-0.792817	-0.51836	-1.019047	-1.210013	-1.695495
2	-0.738480	0.406050	-0.427502	-0.537584	-0.787683	-0.51836	-1.019047	-1.210013	-1.695495
3	-0.699539	0.189057	-0.494395	-0.633026	-0.762704	-0.51836	-1.019047	-1.210013	-1.695495
4	-0.720397	-0.032969	-0.489909	-0.619211	-0.720851	-0.51836	-1.019047	-1.210013	-1.695495

In [288]:
```
yf1.head(5)
```

Out[288]:

	average price
0	1.33
1	1.35
2	0.93
3	1.08
4	1.28

In [ ]:
```

```

SAVING THE MODEL

==========

In [289]:
```
import pickle
```

In [290]:
```
file_name = 'Avocado Average Price Prediction.pkl'
pickle.dump(lr,open(file_name,'wb'))
```

In [ ]:

In [ ]:

APPLYING MODEL FOR SITUATION-2 (CLASSIFICATION MODEL) ==========

In [291]:
```
xf2.head()
```

Out[291]:

	average price	4046	4225	4770	small bags	large bags	xlarge bags	type	year
0	-0.195352	-0.709648	0.059642	-0.533970	-0.517773	-0.796802	-0.51836	-1.019047	-1.210013
1	-0.143510	-0.755064	-0.025495	-0.516157	-0.490834	-0.792817	-0.51836	-1.019047	-1.210013
2	-1.232207	-0.738480	0.406050	-0.427502	-0.537584	-0.787683	-0.51836	-1.019047	-1.210013
3	-0.843387	-0.699539	0.189057	-0.494395	-0.633026	-0.762704	-0.51836	-1.019047	-1.210013
4	-0.324959	-0.720397	-0.032969	-0.489909	-0.619211	-0.720851	-0.51836	-1.019047	-1.210013

In [292]:
```
yf2.head()
```

Out[292]:

	region
0	0
1	0
2	0
3	0
4	0

In [293]:
```
# NOW HERE WE CAN SEE THAT OUR TARGET/LABEL COLUMN ('REGION') IS CATEGORICAL DATA, IT IS HAVIN
# AND WHEN WE ARE HAVING "Y" (TARGET) IN NUMERICAL/CATEGORICAL FORM THEN WE CAN APPLY "CLASSIFI
# SO HERE WE CAN APPLY CLASSIFICATION MODEL'S ON OUR DATASET TO PREDICT, "REGION" for AVOCADO.
```

In [294]:
```
x_train,x_test,y_train,y_test = train_test_split(xf2,yf2,test_size=0.20,random_state=42)
```

In [295]:
```
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
```

In [296]:
```
lg = LogisticRegression()
gnb = GaussianNB()
svc = SVC()
dtc = DecisionTreeClassifier()
knn = KNeighborsClassifier()
```

In [297]:
```
ml_model = [lg,gnb,svc,dtc,knn]
```

In [298]:
```
for i in ml_model:
    i.fit(x_train,y_train)
    i.score(x_train,y_train)
    ipred = i.predict(x_train)

    print('Accuracy Score of ', i,'is:')
    print (accuracy_score(y_train,ipred))

    print(confusion_matrix(y_train,ipred))
    print(classification_report(y_train,ipred))
    print('\n')
```

```
Accuracy Score of  LogisticRegression() is:
0.5086402266288952
[[131   0   0 ...   0   0   0]
 [  0  88   0 ...   0   0   2]
 [  0   0 183 ...   0   0  21]
 ...
 [  0   0   0 ... 124   2   0]
 [  0   0   0 ...   0 190   0]
 [  0  14   0 ...   0   0  66]]
              precision    recall  f1-score   support

           0       0.40      0.49      0.44       270
           1       0.32      0.33      0.32       268
           2       0.66      0.67      0.66       274
           3       0.53      0.46      0.49       267
           4       0.49      0.52      0.50       269
           5       0.44      0.53      0.48       266
           6       0.78      0.85      0.81       234
           7       0.60      0.70      0.65       254
```

In [299]:
```
# here above we can find the accuracy score for the following ML-MOdels is :
# accuracy score for LogisticRegression() =51 %
# accuracy score for GaussianNB() = 22 %
# accuracy score for SVC() = 76 %
# accuracy score for DecisionTreeClassifier() = 100 %
# accuracy score for KNeighborsClassifier() = 89 %

#  here above we can see that the SVC(), DTC(), & KNN() MODELS are working good as compared to
# dtc & knn is best performing models, out of this dtc is perrforming good
# therefore we can decide DTC model has to apply as a FINAL MODEL.
```

In [300]:
```
final_model = DecisionTreeClassifier()
```

In [301]:
```
final_model.fit(x_train,y_train)
final_model.score(x_train,y_train)
final_model_pred = final_model.predict(x_test)
print(accuracy_score(y_test,final_model_pred))
print(confusion_matrix(y_test,final_model_pred))
print(classification_report(y_test,final_model_pred))

# here DECISION TREE CLASSIFIER as FINAL MODEL with ACCURACY OF = 81 %
```

```
0.8014726706315491
[[51  0  0 ...  0  0  0]
 [ 0 50  0 ...  0  0  3]
 [ 0  0 56 ...  0  0  1]
 ...
 [ 0  0  0 ... 38  0  0]
 [ 0  0  0 ...  1 36  0]
 [ 0  1  1 ...  0  0 41]]
              precision    recall  f1-score   support

           0       0.78      0.75      0.77        68
           1       0.74      0.72      0.73        69
           2       0.92      0.88      0.90        64
           3       0.82      0.87      0.84        62
           4       0.90      0.91      0.91        69
           5       0.76      0.75      0.76        72
           6       0.93      0.95      0.94        56
           7       0.90      0.81      0.85        78
           8       0.91      0.94      0.93        54
```

In [ ]:
```
```

In [302]:
```
xf2.shape
```

Out[302]:
```
(17651, 9)
```

In [303]:
```
def pred_func(dt):
    dt= dt.reshape(1,9)
    region = final_model.predict(dt)
    print(region)

    if region == 0:
        print("Albay")
    elif (region == 1):
        print ("Atlanta")
    elif (region == 2):
        print ("Baltimore Washingtone")
    elif (region == 3):
        print ("Bois")
    elif (region == 4):
        print ("Boston")
    elif (region == 5):
        print ("BuffaloRochester")
    elif (region == 6):
        print("California")
    elif (region == 7):
        print ("Charllote")
    elif (region == 8):
        print ("Chicago")
    elif (region == 9):
        print ("Cincinnati")
    elif (region == 10):
        print ("Columbus")
    elif (region == 11):
        print ("DallaFTWorth")
    elif (region == 12):
        print ("Denver")
    elif (region ==13):
        print ("Datroit")
    elif (region == 14):
        print ("Grandrapids")
    elif (region == 15):
        print ("GreatLakes")
    elif (region == 16):
        print ("Harrisburg")
    elif (region == 17):
        print ("HartFord")
    elif (region == 18):
        print ("Huston")
    elif (region == 19):
        print ("IndianaPolis")
    elif (region == 20):
        print ("Jaksonville")
    elif (region == 21):
        print ("Lasvegas")
    elif (region == 22):
        print ("Losangeles")
    elif (region == 23):
        print ("Louisville")
    elif (region == 24):
        print ("Miami")
    elif (region == 25):
        print ("Midsouth")
    elif (region == 26):
        print ("Nashville")
    elif (region == 27):
        print ("Neworleans")
    elif (region == 28):
        print ("Newyork")
    elif (region == 29):
        print ("Northeast")
```

```
    elif (region == 30):
        print ("NothernNewEngland")
    elif (region == 31):
        print ("Orland")
    elif (region == 32):
        print ("Philadeplphia")
    elif (region == 33):
        print ("Phoenix")
    elif (region == 34):
        print ("Pittsburg")
    elif (region == 35):
        print ("Plains")
    elif (region == 36):
        print ("Portlands")
    elif (region == 37):
        print ("Ralaighgreenboro")
    elif (region == 38):
        print ("Richmond")
    elif (region == 39):
        print ("Roanoke")
    elif (region == 40):
        print ("Scaramento")
    elif (region == 41):
        print ("Sandiago")
    elif (region == 42):
        print ("Sanfrancisko")
    elif (region == 43):
        print ("Seattle")
    elif (region == 44):
        print ("SouthCalofornia")
    elif (region == 45):
        print ("SouthCentral")
    elif (region == 46):
        print ("SouthEast")
    elif (region == 47):
        print ("Spokane")
    elif (region == 48):
        print ("StLouis")
    elif (region == 49):
        print ("Syracus")
    elif (region == 50):
        print ("tempa")
    elif (region == 51):
        print ("WestexNewMaxico")
    elif (region == 52):
        print ("TotalUS")
    elif (region == 53):
        print ("West")
    else:
        print('Not Found')
# making 'def' function to predict REGION for Avocado.
```

In [ ]:

In [304]:
```
# TEST SAMPLE-1 ====>>
```

In [305]:
```
xf2.head(1)
```

Out[305]:

	average price	4046	4225	4770	small bags	large bags	xlarge bags	type	year
0	-0.195352	-0.709648	0.059642	-0.53397	-0.517773	-0.796802	-0.51836	-1.019047	-1.210013

In [306]:
```
dt= np.array([-0.195352,-0.709648,0.059642,-0.53397,-0.517773,-0.796802,-0.51836,-1.019047,-1.1
pred_func(dt)

#  data taken from position 'xf2.head(1)'
# here the ATUAL REGION GIVEN DATA IS = ALBAY
#       & THE PREDICTED REGION BY MODEL IS = ALBAY
```

```
[0]
Albay
```

In [ ]:

In [307]:
```
# TEST SAMPLE-2 ====>>
```

In [308]:
```
xf2.tail(1)
```

Out[308]:

	average price	4046	4225	4770	small bags	large bags	xlarge bags	type	year
17650	0.556367	-0.570892	-0.802154	-0.352835	-0.413625	-0.889294	-0.51836	0.981309	1.980756

In [309]:
```
dt= np.array([0.556367,-0.570892,-0.802154,-0.352835,-0.413625,-0.889294,-0.51836,0.981309,1.98
pred_func(dt)

#  data taken from position 'xf2.tail(1)'
# here the ATUAL REGION GIVEN DATA IS = WEST
#       & THE PREDICTED REGION BY MODEL IS = WEST
```

```
[53]
West
```

In [ ]:

In [310]:
```
# TEST SAMPLE-3 ====>>
```

In [311]:
```
# xf2.sample(1)
```

In [312]:
```
dt= np.array([-0.610094,0.726759,0.785801,1.121468,0.221783,-0.708166,-0.293617,-1.019047,-1.21
pred_func(dt)

#  data taken from 1884th position 'xf2.sample(1)'
# here the ATUAL REGION GIVEN DATA IS = Portlands
#       & THE PREDICTED REGION BY MODEL IS = Portlands
```

```
[36]
Portlands
```

In [313]:
```
# HERE ABOVE IN ALL TEST THREE SAMPLES , MODEL PRICTING REGION ACCURATELY
```

In [ ]:

SAVING MODEL

==========

In [314]:
```
import pickle
```

In [315]:
```
file_name = 'avocado_region_identification.pkl'
pickle.dump(final_model,open(file_name,'wb'))
```

========== FINISHED
==========

In [ ]:
```

```