IS388 Data Analysis

PROJECT AKHIR KELOMPOK

Semester Ganjil 2023/2024

GROUP : Nathan Vilbert K – 00000069903 Julius Calvin Saputra – 00000068626 Kevin Aditya Hartono – 00000069875 Juanito Arvin William – 00000069483 Timothy Liong Jonathan – 00000068616

---

1. Business Understanding

1. Penjelasan Dataset

Tabel di bawah ini berisi data penjualan alpukat Hass mingguan tahun 2018 di seluruh Indonesia. Data penjualan ini berasal langsung dari kasir toko berdasarkan penjualan nyata alpukat Hass. Dataset ini menyimpan data penjualan dari tahun 2013 yang mencakup data penjualan dari berbagai jenis toko seperti kelontong, ritel massal, klub belanja, apotek, toko dollar, dan militer. Harga Rata-rata (alpukat) dalam tabel menunjukkan biaya per alpukat, meskipun beberapa alpukat dijual dalam kantong. Kode Pencarian Produk (PLU) hanya berlaku untuk alpukat Hass, sementara jenis alpukat lainnya seperti greenskins tidak termasuk dalam tabel ini. Artinya, kita bisa melihat seberapa banyak dan seberapa mahal alpukat Hass yang terjual di berbagai toko selama tahun 2018.

2. Penjelasan Kolom

• Tanggal (Date): Tanggal pengamatan data penjualan alpukat. • Harga Rata-rata (AveragePrice): Harga rata-rata untuk satu alpukat. • Tipe (Type): Jenis alpukat, apakah konvensional atau organik. • Tahun (Year): Tahun pengamatan data. • Wilayah (Region): Nama kota atau daerah tempat pengamatan dilakukan. • Total Volume: Jumlah total alpukat yang terjual. • 4046: Jumlah total alpukat dengan kode PLU 4046 yang terjual. • 4225: Jumlah total alpukat dengan kode PLU 4225 yang terjual. • 4770: Jumlah total alpukat dengan kode PLU 4770 yang terjual. • Total Bags: Jumlah total kantong yang berisi beberapa alpukat. • Small Bags: Jumlah total kantong kecil yang berisi beberapa alpukat. • Large Bags: Jumlah total kantong besar yang berisi beberapa alpukat. • XLarge Bags: Jumlah total kantong sangat besar yang berisi beberapa alpukat.

Dengan kata lain, data ini memberikan informasi tentang harga, jenis, tahun, wilayah, dan volume penjualan alpukat. Selain itu, kita juga dapat melihat rincian penjualan berdasarkan jenis kemasan, seperti kantong kecil, kantong besar, dan sebagainya. Jumlah alpukat dengan kode PLU tertentu (4046, 4225, dan 4770) juga dicantumkan, memberikan informasi lebih lanjut tentang variasi jenis alpukat yang terjual.

Tujuan dari analisis data adalah memahami faktor-faktor yang mempengaruhi penjualan alpukat Hass untuk membantu pengambilan keputusan dan strategi pemasaran

CODE

2. Data Understanding

In [1]:
```
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

In [2]:
```
avocado = pd.read_csv("avocado-updated-2020.csv")
avocado.head(10)
```

Out[2]:

	date	average_price	total_volume	4046	4225	4770	total_bags	smal
0	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	9
1	2015-01-04	1.79	1373.95	57.42	153.88	0.00	1162.65	1
2	2015-01-04	1.00	435021.49	364302.39	23821.16	82.15	46815.79	16
3	2015-01-04	1.76	3846.69	1500.15	938.35	0.00	1408.19	1
4	2015-01-04	1.08	788025.06	53987.31	552906.04	39995.03	141136.68	137
5	2015-01-04	1.29	19137.28	8040.64	6557.47	657.48	3881.69	3
6	2015-01-04	1.01	80034.32	44562.12	24964.23	2752.35	7755.62	6
7	2015-01-04	1.64	1505.12	1.27	1129.50	0.00	374.35	
8	2015-01-04	1.02	491738.00	7193.87	396752.18	128.82	87663.13	87
9	2015-01-04	1.83	2192.13	8.66	939.43	0.00	1244.04	1

In [3]:
```
avocado.describe()
```

Out[3]:

	average_price	total_volume	4046	4225	4770	total
count	33045.000000	3.304500e+04	3.304500e+04	3.304500e+04	3.304500e+04	3.30450
mean	1.379941	9.683997e+05	3.023914e+05	2.797693e+05	2.148255e+04	3.64673
std	0.378972	3.934533e+06	1.301026e+06	1.151052e+06	1.001607e+05	1.56400
min	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
25%	1.100000	1.511895e+04	7.673100e+02	2.712470e+03	0.000000e+00	9.12186
50%	1.350000	1.291170e+05	1.099477e+04	2.343600e+04	1.780900e+02	5.32222
75%	1.620000	5.058285e+05	1.190219e+05	1.352389e+05	5.096530e+03	1.74431
max	3.250000	6.371614e+07	2.274362e+07	2.047057e+07	2.546439e+06	3.16891

In [4]:

```
avocado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33045 entries, 0 to 33044
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           33045 non-null  object 
 1   average_price  33045 non-null  float64
 2   total_volume   33045 non-null  float64
 3   4046           33045 non-null  float64
 4   4225           33045 non-null  float64
 5   4770           33045 non-null  float64
 6   total_bags     33045 non-null  float64
 7   small_bags     33045 non-null  float64
 8   large_bags     33045 non-null  float64
 9   xlarge_bags    33045 non-null  float64
 10  type           33045 non-null  object 
 11  year           33045 non-null  int64  
 12  geography      33045 non-null  object 
dtypes: float64(9), int64(1), object(3)
memory usage: 3.3+ MB
```

In [5]:

```
avocado.shape
```

Out[5]: (33045, 13)

In [6]:

```
avocado.isnull().sum()
```

Out[6]:
```
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
geography        0
dtype: int64
```

In [7]:
```
avocado.dropna()
```

Out[7]:

	date	average_price	total_volume	4046	4225	4770	total_bags
0	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46
1	2015-01-04	1.79	1373.95	57.42	153.88	0.00	1162.65
2	2015-01-04	1.00	435021.49	364302.39	23821.16	82.15	46815.79
3	2015-01-04	1.76	3846.69	1500.15	938.35	0.00	1408.19
4	2015-01-04	1.08	788025.06	53987.31	552906.04	39995.03	141136.68
...	...	...	...	...	...	...	...
33040	2020-11-29	1.47	1583056.27	67544.48	97996.46	2617.17	1414878.10
33041	2020-11-29	0.91	5811114.22	1352877.53	589061.83	19741.90	3790665.29
33042	2020-11-29	1.48	289961.27	13273.75	19341.09	636.51	256709.92
33043	2020-11-29	0.67	822818.75	234688.01	80205.15	10543.63	497381.96
33044	2020-11-29	1.35	24106.58	1236.96	617.80	1564.98	20686.84

33045 rows × 13 columns

In [8]:
```
avocado.head(10)
```

Out[8]:

	date	average_price	total_volume	4046	4225	4770	total_bags	smal
0	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	9
1	2015-01-04	1.79	1373.95	57.42	153.88	0.00	1162.65	1
2	2015-01-04	1.00	435021.49	364302.39	23821.16	82.15	46815.79	16
3	2015-01-04	1.76	3846.69	1500.15	938.35	0.00	1408.19	1
4	2015-01-04	1.08	788025.06	53987.31	552906.04	39995.03	141136.68	137
5	2015-01-04	1.29	19137.28	8040.64	6557.47	657.48	3881.69	3
6	2015-01-04	1.01	80034.32	44562.12	24964.23	2752.35	7755.62	6
7	2015-01-04	1.64	1505.12	1.27	1129.50	0.00	374.35	
8	2015-01-04	1.02	491738.00	7193.87	396752.18	128.82	87663.13	87
9	2015-01-04	1.83	2192.13	8.66	939.43	0.00	1244.04	1

In [9]:

```
avocado.columns = ['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770
                   'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags',
                   'type', 'year', 'region']
avocado.head()
avocado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33045 entries, 0 to 33044
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          33045 non-null  object
 1   AveragePrice  33045 non-null  float64
 2   Total Volume  33045 non-null  float64
 3   4046          33045 non-null  float64
 4   4225          33045 non-null  float64
 5   4770          33045 non-null  float64
 6   Total Bags    33045 non-null  float64
 7   Small Bags    33045 non-null  float64
 8   Large Bags    33045 non-null  float64
 9   XLarge Bags   33045 non-null  float64
 10  type          33045 non-null  object
 11  year          33045 non-null  int64
 12  region        33045 non-null  object
dtypes: float64(9), int64(1), object(3)
memory usage: 3.3+ MB
```

Exploratory Data Analysis

3. DATA PREPARATION

In [10]:
```
#Handling Missing Values
```

In [11]:
```
avocado.fillna(avocado.mean(), inplace = True)
avocado.head(5)
```

```
C:\Users\calvi\AppData\Local\Temp\ipykernel_10720\3087528212.py:1: FutureWarning:
The default value of numeric_only in DataFrame.mean is deprecated. In a future ve
rsion, it will default to False. In addition, specifying 'numeric_only=None' is d
eprecated. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  avocado.fillna(avocado.mean(), inplace = True)
```

Out[11]:

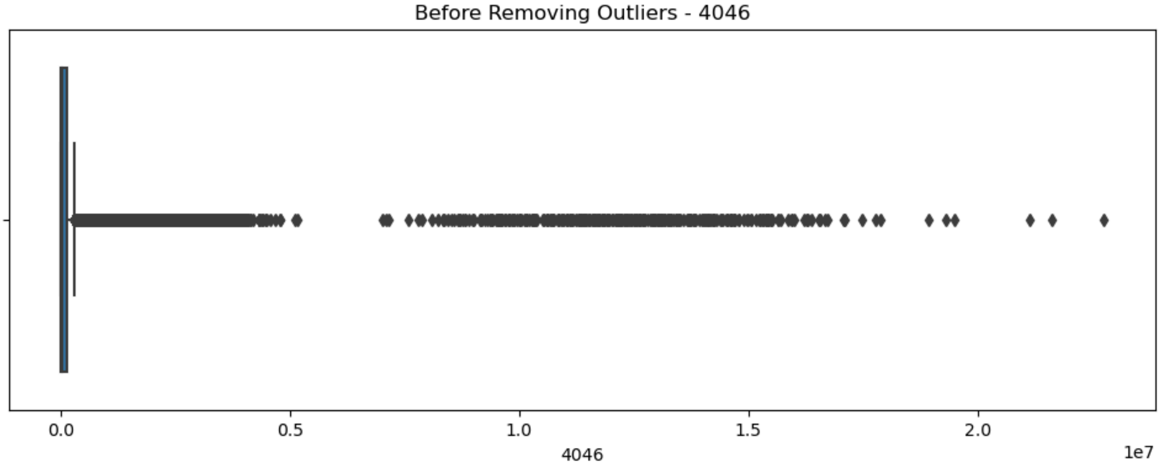
	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags
0	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	9186.93
1	2015-01-04	1.79	1373.95	57.42	153.88	0.00	1162.65	1162.65
2	2015-01-04	1.00	435021.49	364302.39	23821.16	82.15	46815.79	16707.15
3	2015-01-04	1.76	3846.69	1500.15	938.35	0.00	1408.19	1071.35
4	2015-01-04	1.08	788025.06	53987.31	552906.04	39995.03	141136.68	137146.07

In [12]:
```
#Handling Outliers
```

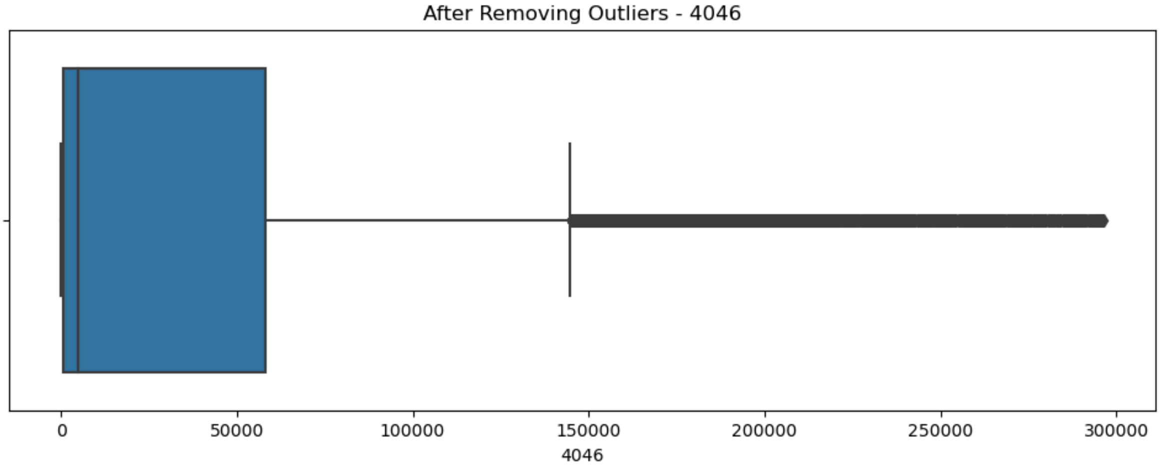
In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print("Jumlah Data Sebelum Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4046'])
plt.title('Before Removing Outliers - 4046')
plt.show()

Q1 = avocado['4046'].quantile(0.25)
Q3 = avocado['4046'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

avocado = avocado[(avocado['4046'] >= lower_bound) & (avocado['4046'] <= upper_b
```

```
print("\nJumlah Data Setelah Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4046'])
plt.title('After Removing Outliers - 4046')
plt.show()
```

```
Jumlah Data Sebelum Menghilangkan Outlier: 33045
```




```
Jumlah Data Setelah Menghilangkan Outlier: 28405
```

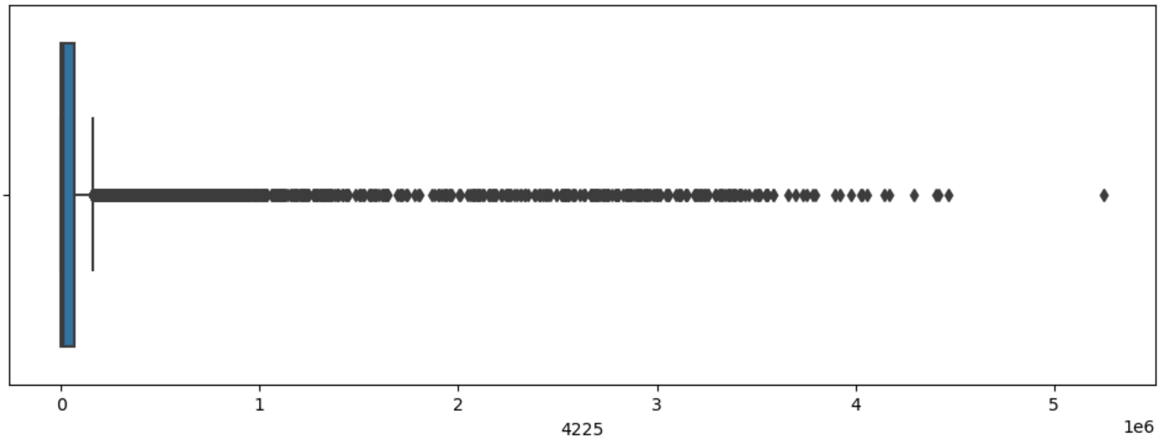


In [14]:

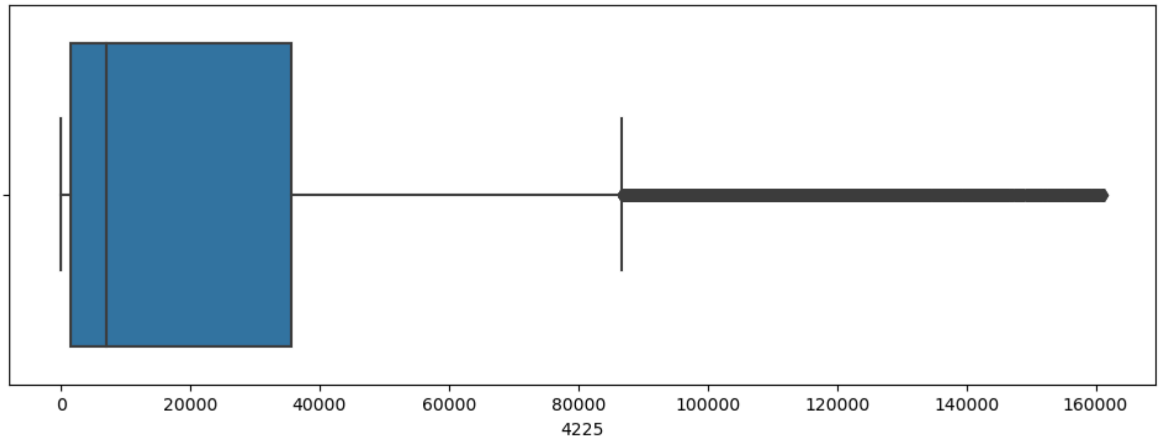
```
print("Jumlah Data Sebelum Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4225'])
plt.title('Before Removing Outliers - 4225')
plt.show()

Q1 = avocado['4225'].quantile(0.25)
Q3 = avocado['4225'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

avocado = avocado[(avocado['4225'] >= lower_bound) & (avocado['4225'] <= upper_b

print("\nJumlah Data Setelah Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4225'])
```

```
plt.title('After Removing Outliers - 4225')
plt.show()
```

Jumlah Data Sebelum Menghilangkan Outlier: 28405

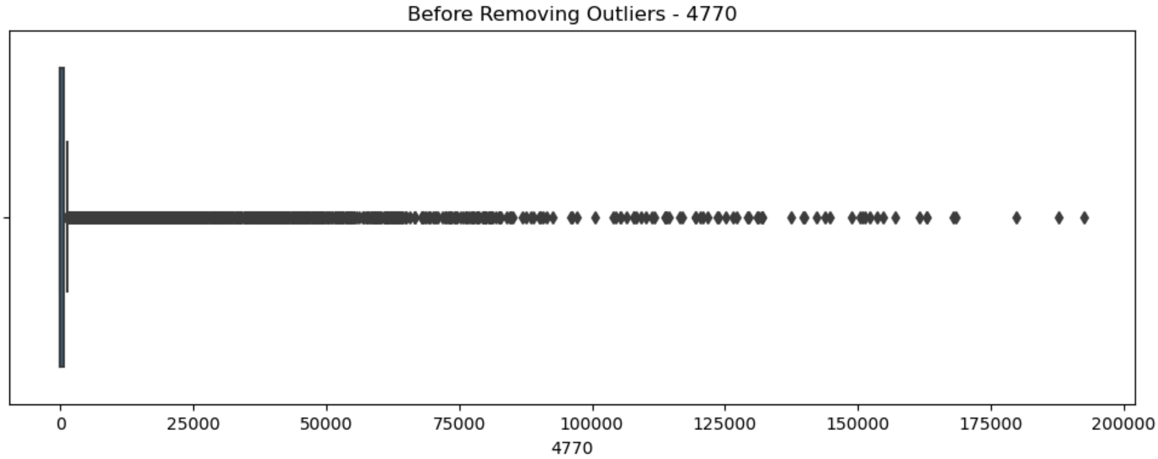


Jumlah Data Setelah Menghilangkan Outlier: 24460

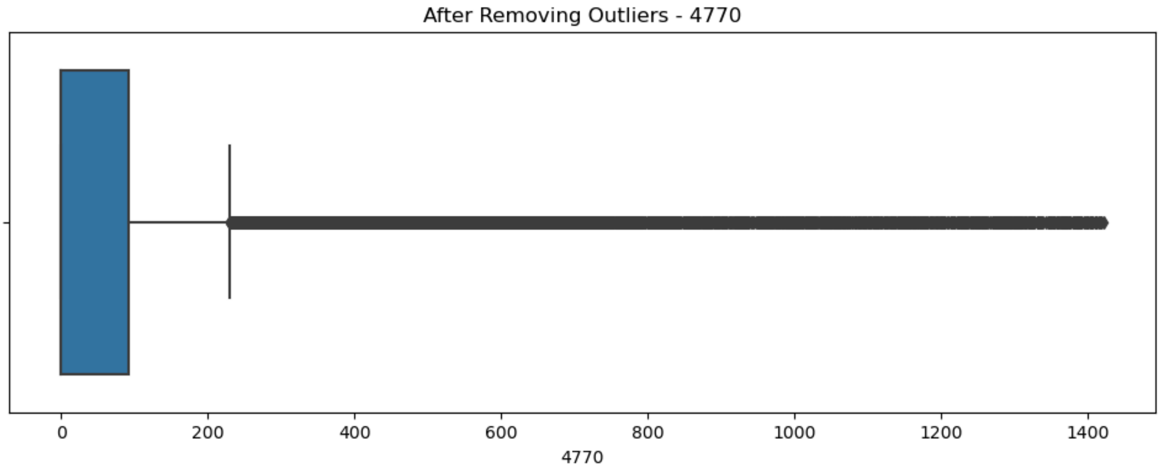


In [15]:
```
print("Jumlah Data Sebelum Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4770'])
plt.title('Before Removing Outliers - 4770')
plt.show()

Q1 = avocado['4770'].quantile(0.25)
Q3 = avocado['4770'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

avocado = avocado[(avocado['4770'] >= lower_bound) & (avocado['4770'] <= upper_b

print("\nJumlah Data Setelah Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['4770'])
plt.title('After Removing Outliers - 4770')
plt.show()
```

Jumlah Data Sebelum Menghilangkan Outlier: 24460

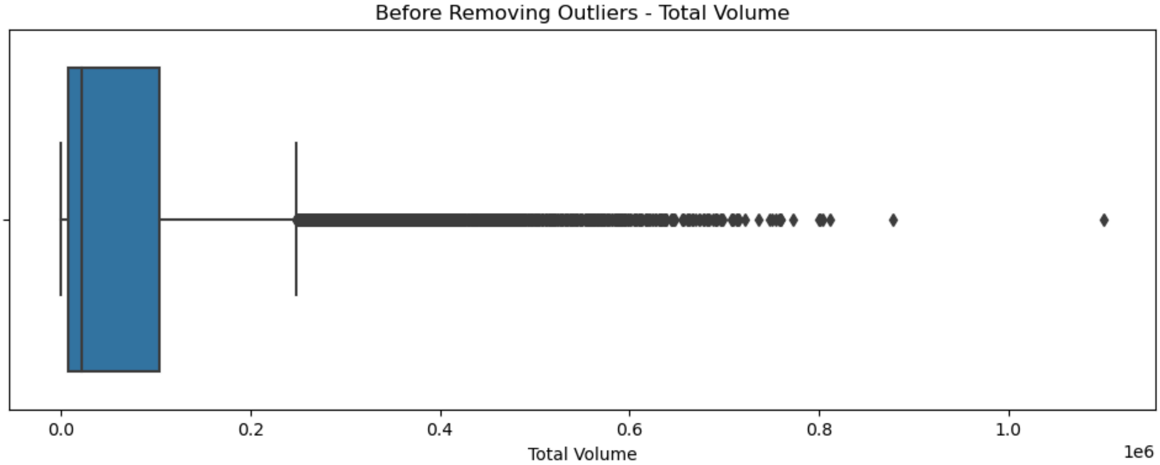



```
Jumlah Data Setelah Menghilangkan Outlier: 19823
```

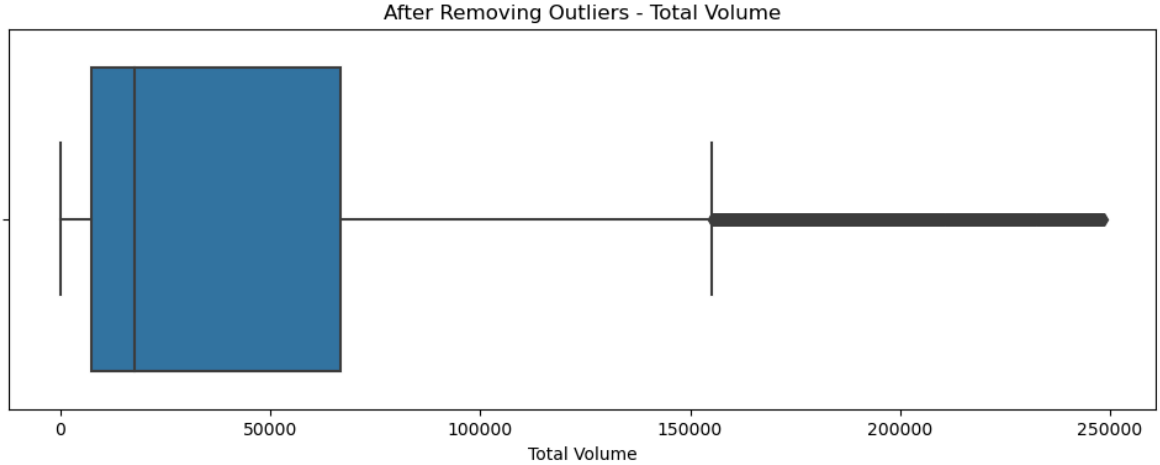


In [16]:
```
print("Jumlah Data Sebelum Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['Total Volume'])
plt.title('Before Removing Outliers - Total Volume')
plt.show()

Q1 = avocado['Total Volume'].quantile(0.25)
Q3 = avocado['Total Volume'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

avocado = avocado[(avocado['Total Volume'] >= lower_bound) & (avocado['Total Vol

print("\nJumlah Data Setelah Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['Total Volume'])
plt.title('After Removing Outliers - Total Volume')
plt.show()
```

```
Jumlah Data Sebelum Menghilangkan Outlier: 19823
```

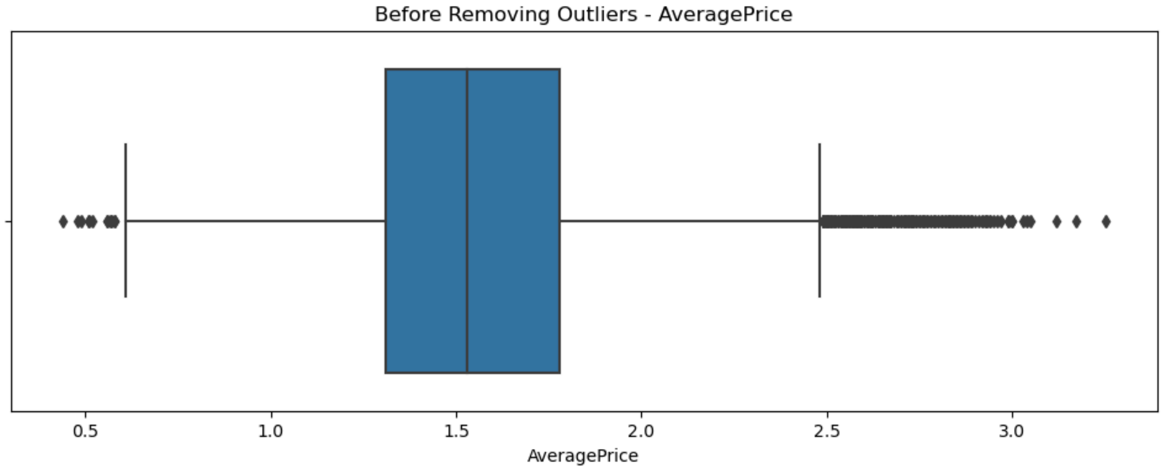



```
Jumlah Data Setelah Menghilangkan Outlier: 18064
```

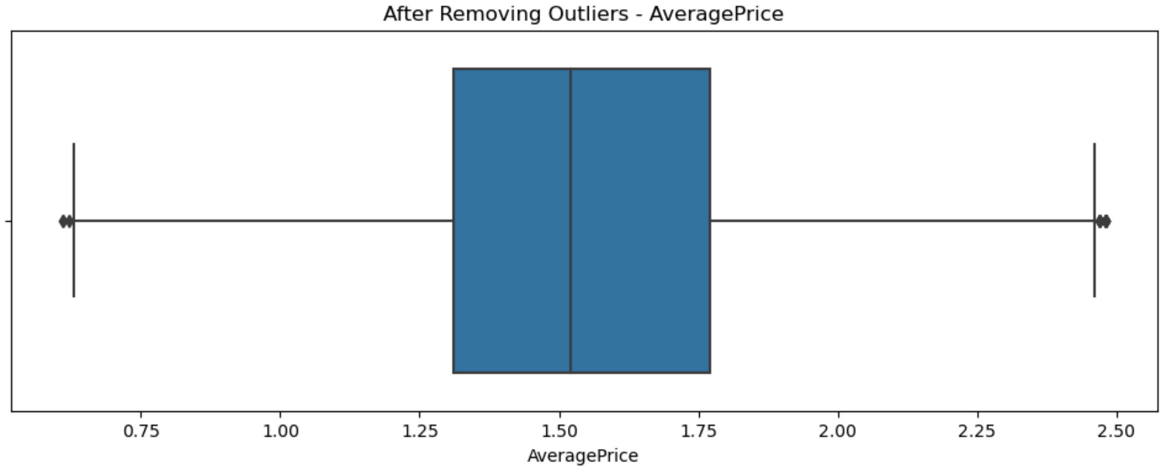


In [17]:
```
print("Jumlah Data Sebelum Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['AveragePrice'])
plt.title('Before Removing Outliers - AveragePrice')
plt.show()

Q1 = avocado['AveragePrice'].quantile(0.25)
Q3 = avocado['AveragePrice'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

avocado = avocado[(avocado['AveragePrice'] >= lower_bound) & (avocado['AveragePr

print("\nJumlah Data Setelah Menghilangkan Outlier:", len(avocado))

plt.figure(figsize=(12, 4))
sns.boxplot(x=avocado['AveragePrice'])
plt.title('After Removing Outliers - AveragePrice')
plt.show()
```

```
Jumlah Data Sebelum Menghilangkan Outlier: 18064
```




```
Jumlah Data Setelah Menghilangkan Outlier: 17801
```



In [18]:
```
#Formatting
```

In [19]:
```
avocado['Date'] = pd.to_datetime(avocado['Date'])
```

In [20]:
```
datavis = avocado[['AveragePrice', 'Date']]
datavis['Date']= pd.to_datetime(datavis['Date'], infer_datetime_format=True)
print(datavis.dtypes)
```

```
AveragePrice           float64
Date            datetime64[ns]
dtype: object
```

```
C:\Users\calvi\AppData\Local\Temp\ipykernel_10720\252701598.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  datavis['Date']= pd.to_datetime(datavis['Date'], infer_datetime_format=True)
```

In [21]:
```
#Normalization
```

In [22]:
```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd

scaler = MinMaxScaler()
avocado['AveragePrice'] = scaler.fit_transform(avocado['AveragePrice'].values.re
```

```
avocado.head()
```

Out[22]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
0	2015-01-04	0.326203	40873.28	2819.50	28287.42	49.90	9716.46	9186.93	529.53
1	2015-01-04	0.631016	1373.95	57.42	153.88	0.00	1162.65	1162.65	0.00
3	2015-01-04	0.614973	3846.69	1500.15	938.35	0.00	1408.19	1071.35	336.84
5	2015-01-04	0.363636	19137.28	8040.64	6557.47	657.48	3881.69	3881.69	0.00
7	2015-01-04	0.550802	1505.12	1.27	1129.50	0.00	374.35	186.67	187.68

In [23]:

```
#Encoding
```

In [24]:

```
avocado['type'] = avocado['type'].replace({'conventional': 1, 'organic': 2})
print(avocado)
```

```
             Date  AveragePrice  Total Volume      4046      4225    4770  \
0      2015-01-04      0.326203      40873.28   2819.50  28287.42   49.90
1      2015-01-04      0.631016       1373.95     57.42    153.88    0.00
3      2015-01-04      0.614973       3846.69   1500.15    938.35    0.00
5      2015-01-04      0.363636      19137.28   8040.64   6557.47  657.48
7      2015-01-04      0.550802       1505.12      1.27   1129.50    0.00
...           ...           ...           ...       ...       ...     ...
33033  2020-11-29      0.117647     189187.58  78597.67   9497.22   65.16
33034  2020-11-29      0.657754       5898.33    677.71    912.70    0.00
33035  2020-11-29      0.181818      72128.91   6789.51  31201.09  627.87
33036  2020-11-29      0.454545       3191.59    166.36     89.78    0.00
33038  2020-11-29      0.181818      11883.88    101.71      0.00    0.00

       Total Bags  Small Bags  Large Bags  XLarge Bags  type  year  \
0         9716.46     9186.93      529.53         0.00     1  2015
1         1162.65     1162.65        0.00         0.00     2  2015
3         1408.19     1071.35      336.84         0.00     2  2015
5         3881.69     3881.69        0.00         0.00     2  2015
7          374.35      186.67      187.68         0.00     2  2015
...           ...         ...         ...          ...   ...   ...
33033   101027.53    93625.26     7402.27         0.00     1  2020
33034     4307.92     4301.25        6.67         0.00     2  2020
33035    33510.44    20587.54    11866.23      1056.67     1  2020
33036     2935.45     2618.57      316.88         0.00     2  2020
33038    11782.17    11782.17        0.00         0.00     2  2020

                    region
0                   Albany
1                   Albany
3                  Atlanta
5      Baltimore/Washington
7                    Boise
...                    ...
33033            St. Louis
33034            St. Louis
33035             Syracuse
33036             Syracuse
33038                Tampa

[17801 rows x 13 columns]
```

In [25]:
```python
#Binning
```

In [26]:
```python
import pandas as pd

bin_edges = [0, 0.2, 0.4, 0.6, 0.8, 1.0]
bin_labels = ['Very Low', 'Low', 'Average', 'High', 'Very High']

avocado['Price Level'] = pd.cut(avocado['AveragePrice'], bins=bin_edges, labels=

avocado.head()
```

Out[26]:

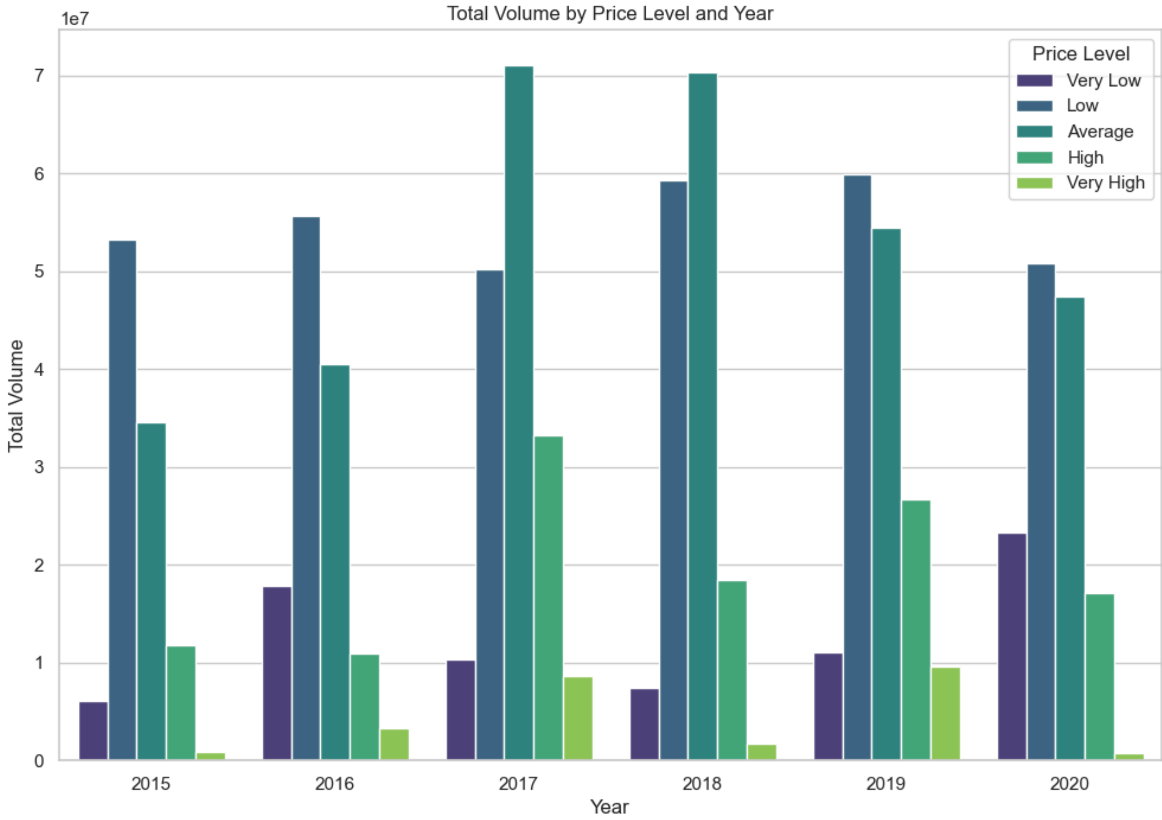
	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
0	2015-01-04	0.326203	40873.28	2819.50	28287.42	49.90	9716.46	9186.93	529.53
1	2015-01-04	0.631016	1373.95	57.42	153.88	0.00	1162.65	1162.65	0.00
3	2015-01-04	0.614973	3846.69	1500.15	938.35	0.00	1408.19	1071.35	336.84
5	2015-01-04	0.363636	19137.28	8040.64	6557.47	657.48	3881.69	3881.69	0.00
7	2015-01-04	0.550802	1505.12	1.27	1129.50	0.00	374.35	186.67	187.68

In [27]:

```
#Grouping
```

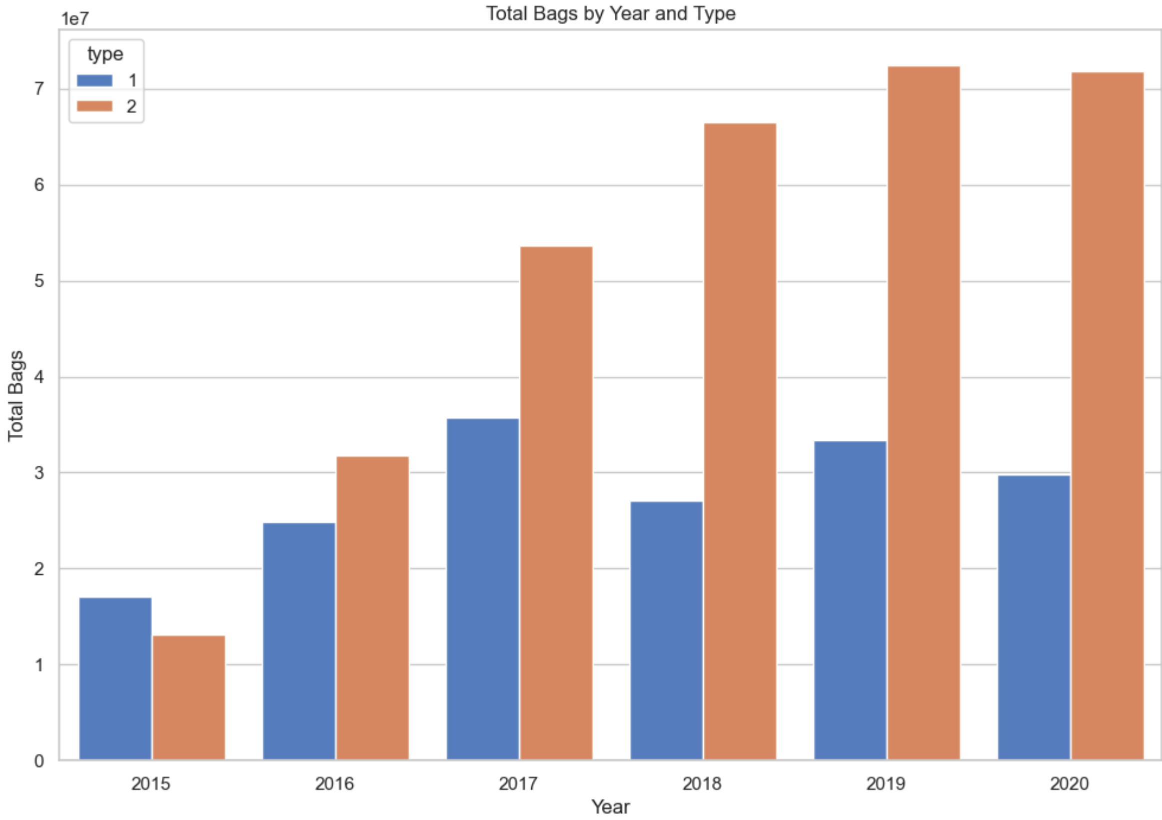
In [28]:

```
total_volume_by_price_year = avocado.groupby(['Price Level', 'year'])['Total Vol
total_volume_by_price_year.head()

sns.set(style="whitegrid")

plt.figure(figsize=(12, 8))
sns.barplot(x='year', y='Total Volume', hue='Price Level', data=total_volume_by_

plt.xlabel('Year')
plt.ylabel('Total Volume')
plt.title('Total Volume by Price Level and Year')

plt.show()
```

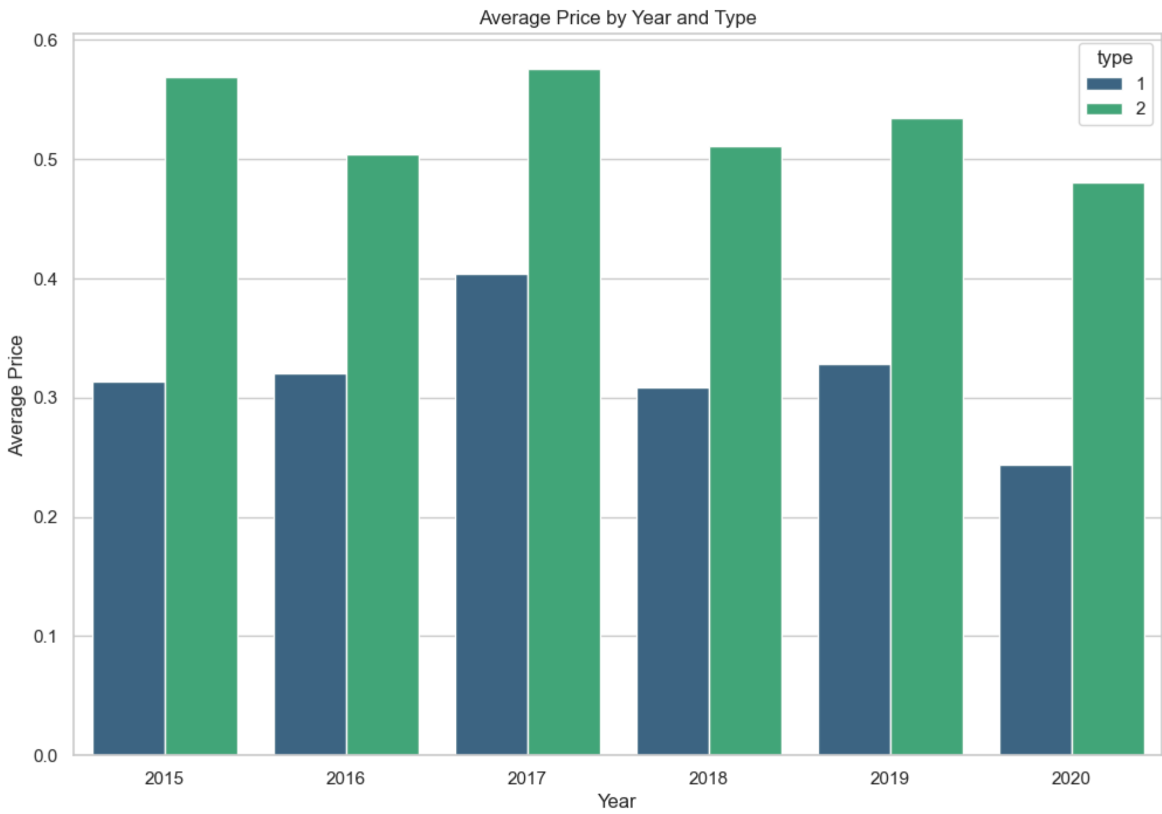


In [29]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming avocado is your DataFrame

# Group by 'year' and 'type' and calculate the sum of 'Total Bags'
TotalBags_by_year_type = avocado.groupby(['year', 'type'])['Total Bags'].sum().r

# Set the plotting style
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='year', y='Total Bags', hue='type', data=TotalBags_by_year_type, p

# Set plot labels and title
plt.xlabel('Year')
plt.ylabel('Total Bags')
plt.title('Total Bags by Year and Type')

# Show the plot
plt.show()
```



In [30]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming avocado is your DataFrame

# Group by 'year' and 'type' and calculate the mean of 'AveragePrice'
AveragePrices_by_year_type = avocado.groupby(['year', 'type'])['AveragePrice'].m

# Set the plotting style
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='year', y='AveragePrice', hue='type', data=AveragePrices_by_year_t

# Set plot labels and title
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.title('Average Price by Year and Type')

# Show the plot
plt.show()
```



In [31]:
```
avocado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17801 entries, 0 to 33038
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          17801 non-null  datetime64[ns]
 1   AveragePrice  17801 non-null  float64
 2   Total Volume  17801 non-null  float64
 3   4046          17801 non-null  float64
 4   4225          17801 non-null  float64
 5   4770          17801 non-null  float64
 6   Total Bags    17801 non-null  float64
 7   Small Bags    17801 non-null  float64
 8   Large Bags    17801 non-null  float64
 9   XLarge Bags   17801 non-null  float64
 10  type          17801 non-null  int64
 11  year          17801 non-null  int64
 12  region        17801 non-null  object
 13  Price Level   17801 non-null  category
dtypes: category(1), datetime64[ns](1), float64(9), int64(2), object(1)
memory usage: 1.9+ MB
```

In [32]:
```
datavis = datavis.set_index(['Date'])
datavis.head()
```

Out[32]:

	AveragePrice
Date	
2015-01-04	1.22
2015-01-04	1.79
2015-01-04	1.76
2015-01-04	1.29
2015-01-04	1.64

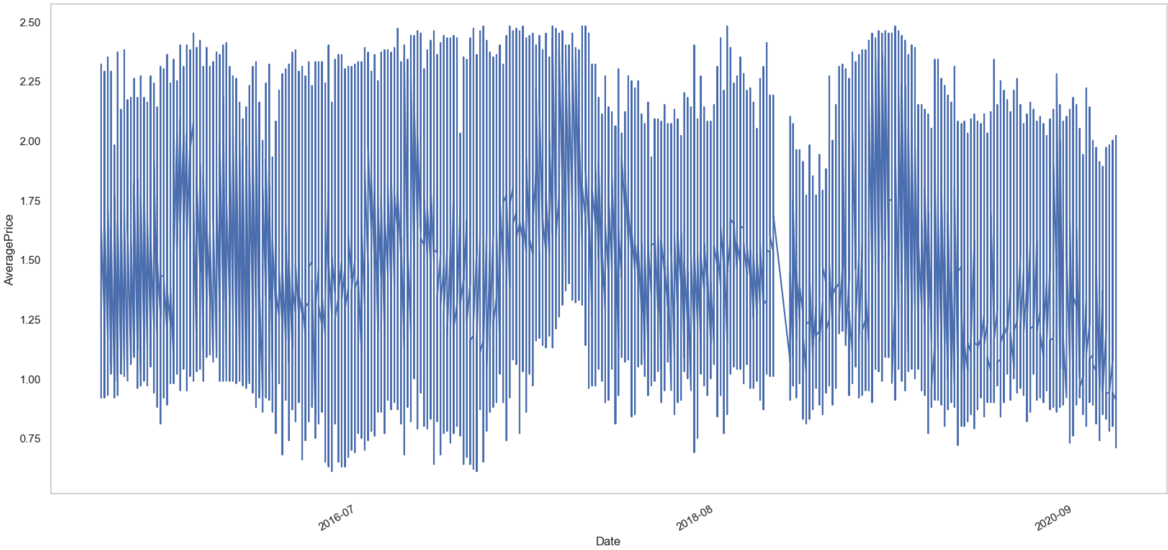
In [33]:

```
print(datavis.describe().T)
print('AveragePrice')

print(datavis.describe().T.round(2))
```

```
                count      mean       std   min   25%   50%   75%   max
AveragePrice  17801.0  1.543478  0.333942  0.61  1.31  1.52  1.77  2.48
AveragePrice
                count  mean   std   min   25%   50%   75%   max
AveragePrice  17801.0  1.54  0.33  0.61  1.31  1.52  1.77  2.48
```

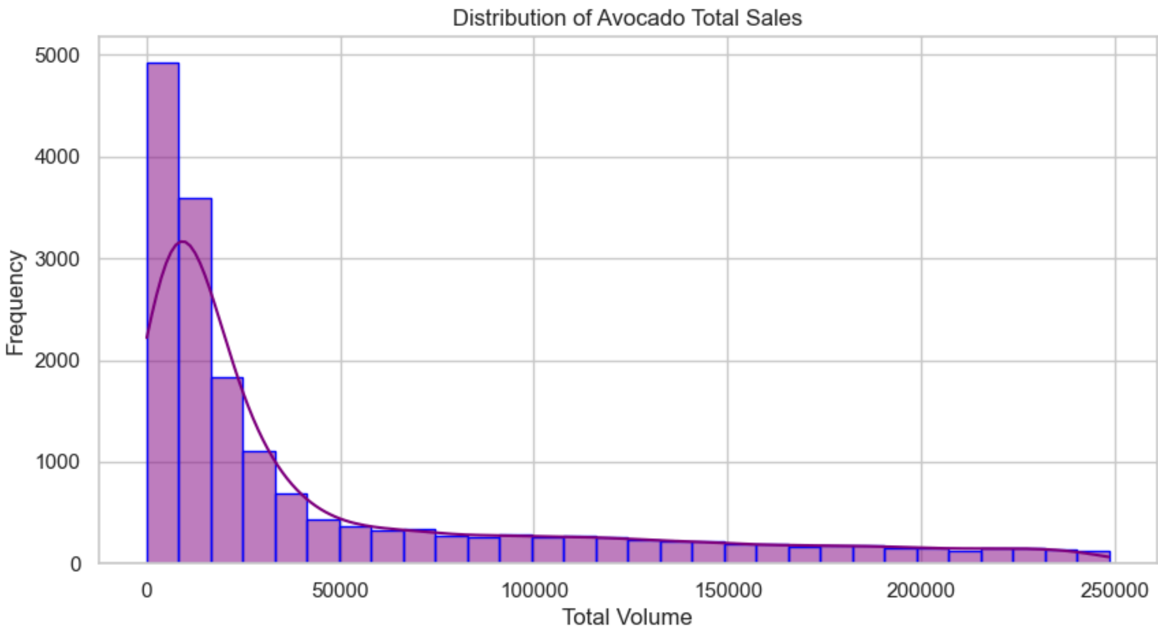
In [34]:

```
import matplotlib.dates as mdates
import matplotlib.pyplot as plty

fig, ax = plty.subplots(figsize=(20,10))
plty.xlabel("Date")
plty.ylabel("AveragePrice")

half_year_locator = mdates.MonthLocator(interval = 25)
year_month_formatter = mdates.DateFormatter('%Y-%m')

ax.xaxis.set_major_locator(half_year_locator)
ax.xaxis.set_major_formatter(year_month_formatter)

ax.plot(datavis)
fig.autofmt_xdate()
plty.grid()
plty.show()
```

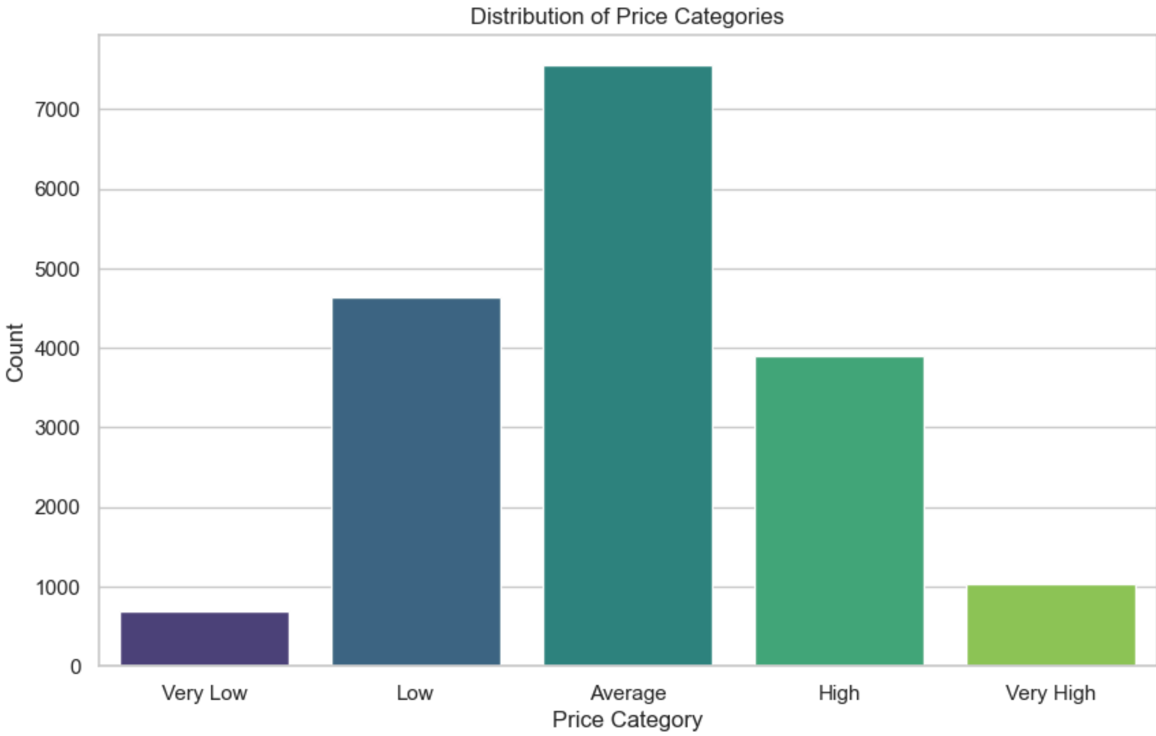


1. Distribution

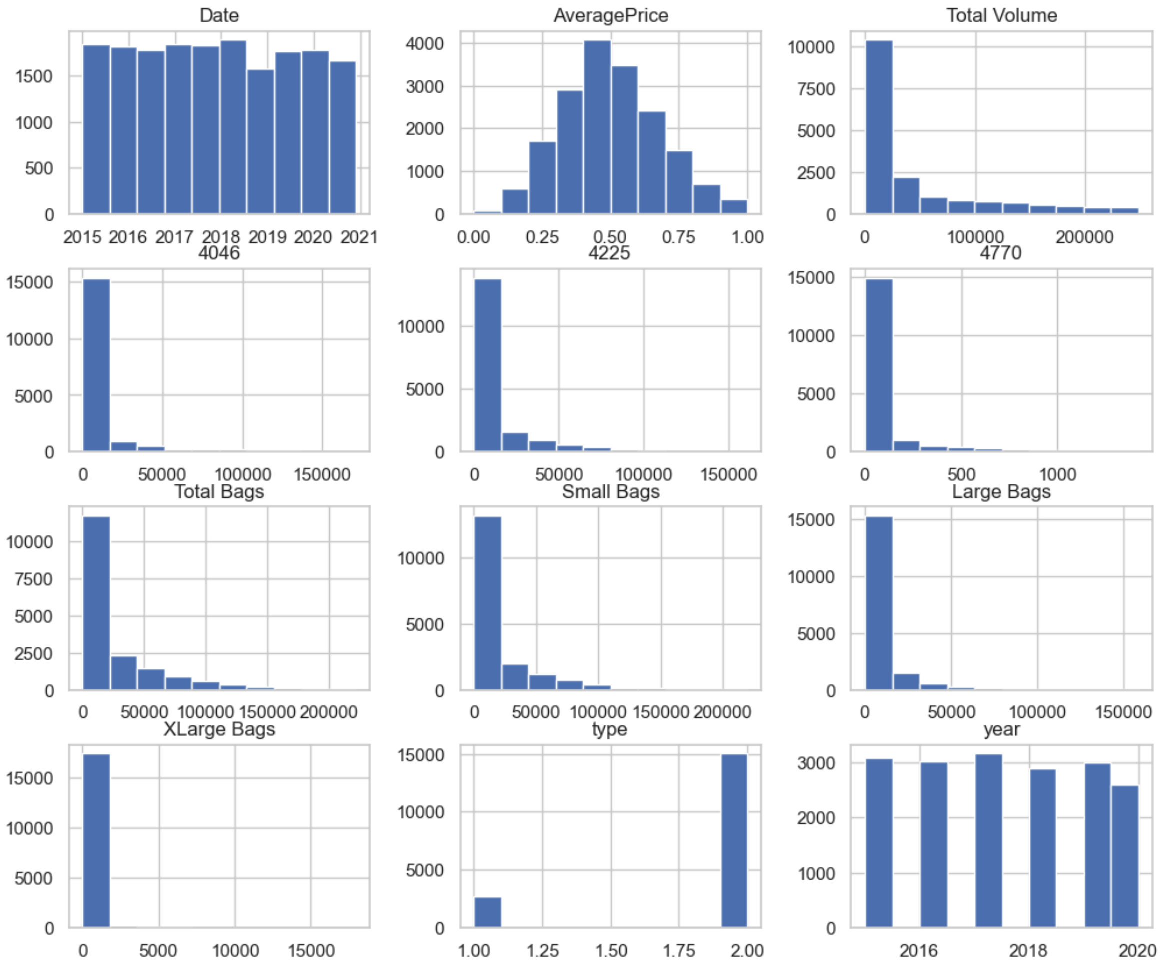
In [35]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.histplot(avocado['Total Volume'], color='purple', bins = 30, kde= True, edge
plt.xlabel('Total Volume')
plt.ylabel('Frequency')
plt.title('Distribution of Avocado Total Sales')
plt.show()
```



In [36]:
```
sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))
sns.countplot(x='Price Level', data=avocado, palette='viridis')

plt.xlabel('Price Category')
plt.ylabel('Count')
plt.title('Distribution of Price Categories')

plt.show()
```

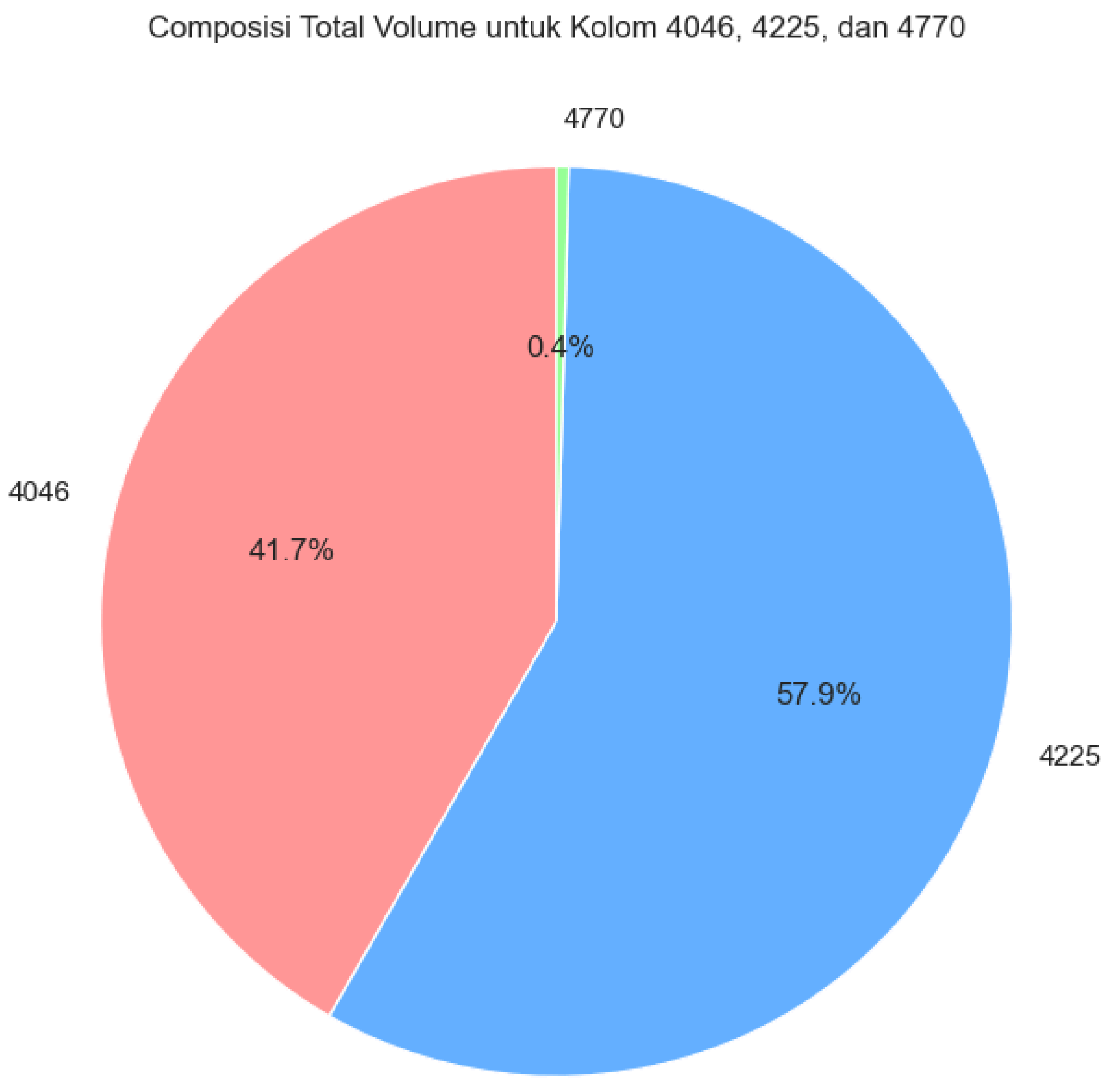


In [37]:
```
avocado.hist(figsize=(12, 10))
plt.show()
```



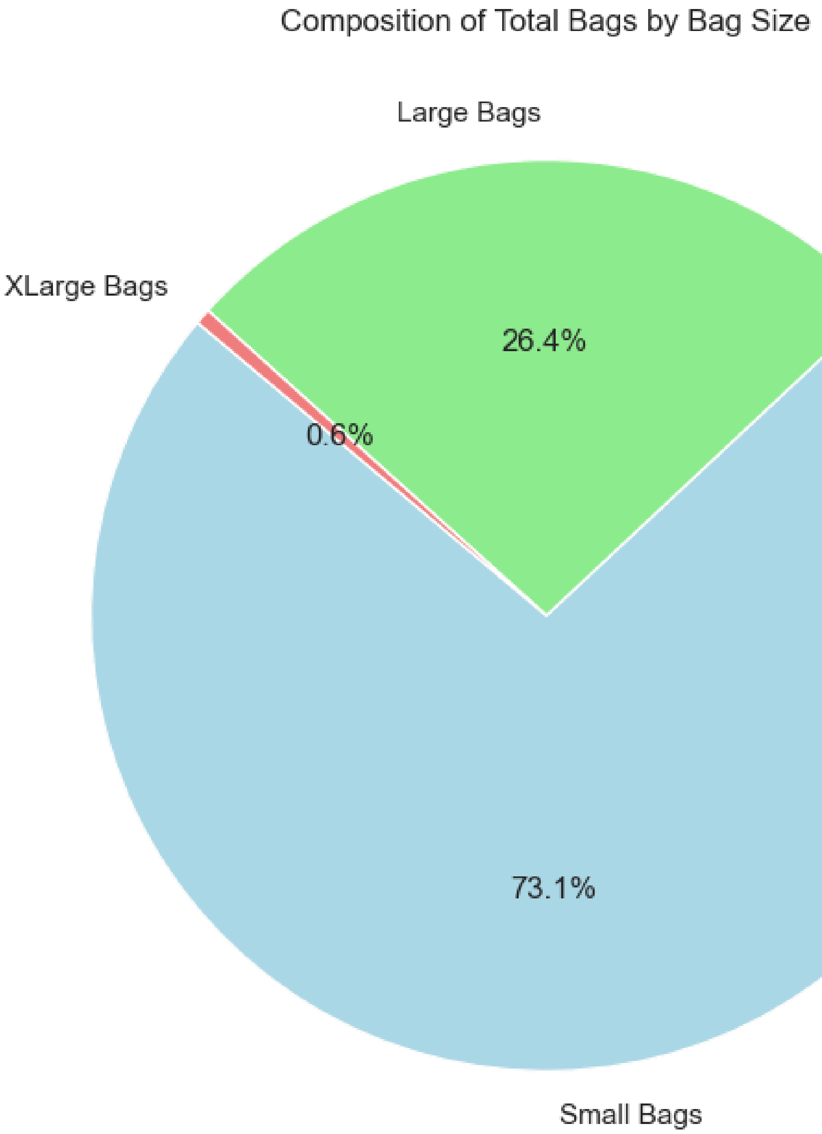
2.Composition

In [38]:
```
#Komposisi Total Penjualan Berdasarkan Kode PLU Alpukat
total_4046 = avocado['4046'].sum()
total_4225 = avocado['4225'].sum()
total_4770 = avocado['4770'].sum()

# Menyiapkan data untuk pie chart
labels = ['4046', '4225', '4770']
sizes = [total_4046, total_4225, total_4770]
colors = ['#ff9999','#66b3ff','#99ff99']

# Membuat pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title('Composisi Total Volume untuk Kolom 4046, 4225, dan 4770')
plt.show()
```



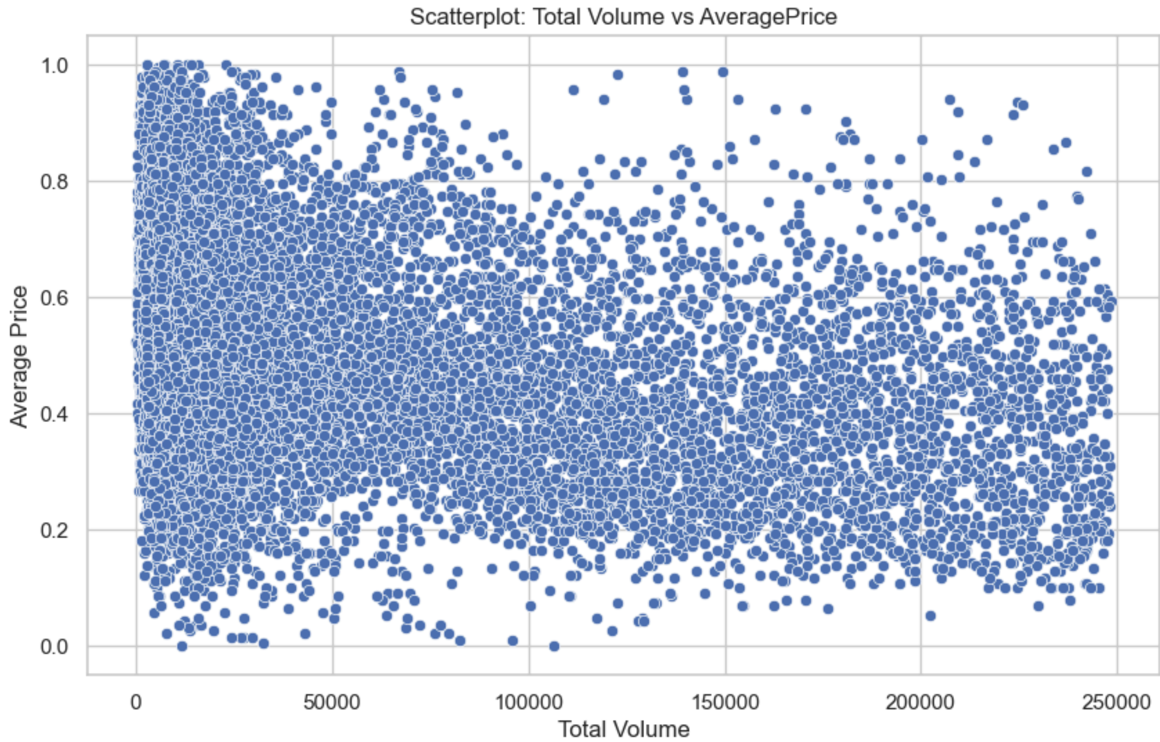
In [39]:
```
import matplotlib.pyplot as plt

small_bags = avocado['Small Bags'].sum()
large_bags = avocado['Large Bags'].sum()
xlarge_bags = avocado['XLarge Bags'].sum()

total_bags = [small_bags, large_bags, xlarge_bags]
```

```
labels = ['Small Bags', 'Large Bags', 'XLarge Bags']

plt.figure(figsize=(8, 8))
plt.pie(total_bags, labels=labels, autopct='%1.1f%%', startangle=140, colors=['l
plt.title('Composition of Total Bags by Bag Size')
plt.show()
```

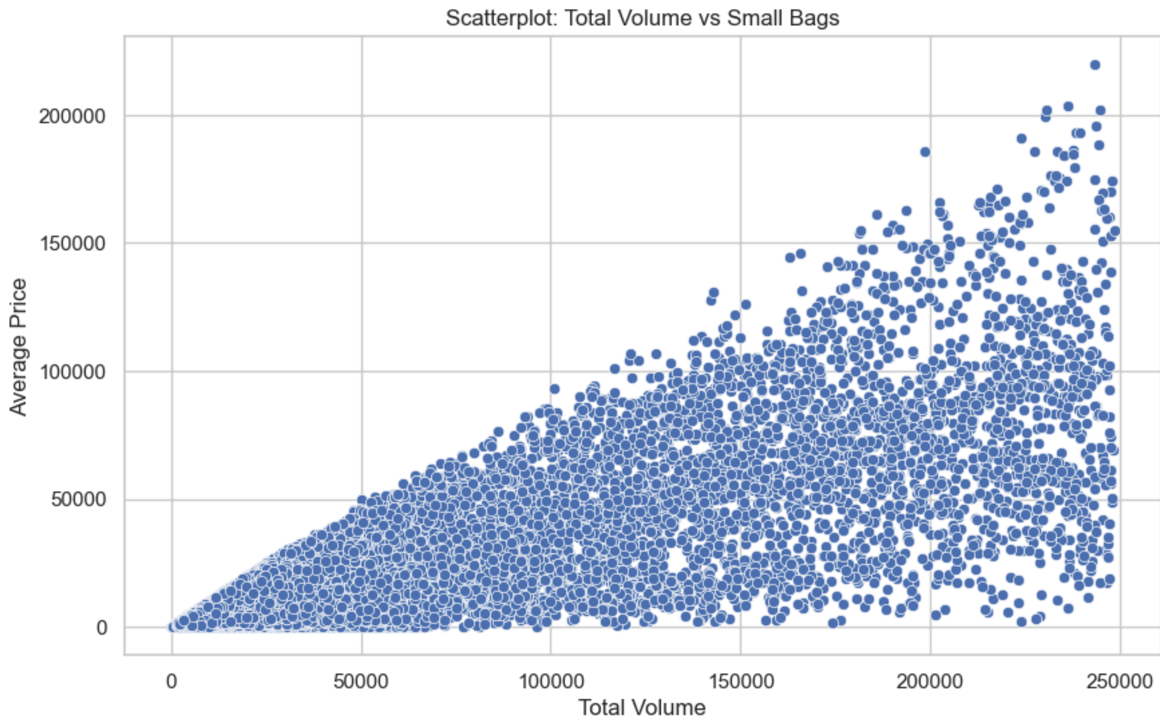


3.Relation

In [40]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Total Volume', y='AveragePrice', data=avocado)
plt.title('Scatterplot: Total Volume vs AveragePrice')
plt.xlabel('Total Volume')
plt.ylabel('Average Price')
plt.show()
```



In [41]:
```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Total Volume', y='Small Bags', data=avocado)
plt.title('Scatterplot: Total Volume vs Small Bags')
plt.xlabel('Total Volume')
plt.ylabel('Average Price')
plt.show()
```

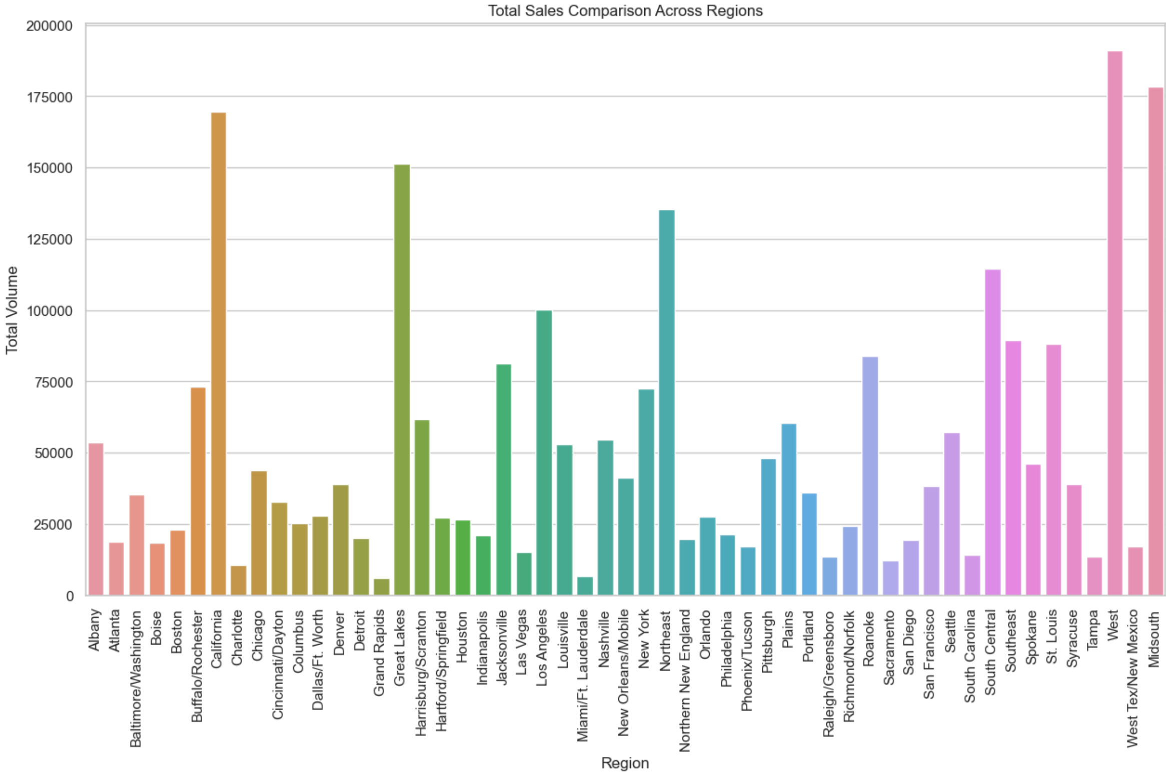


4.Comparison

In [42]:
```python
plt.figure(figsize=(15, 8))
sns.barplot(x='region', y='Total Volume', data=avocado, ci=None)
plt.title('Total Sales Comparison Across Regions')
plt.xlabel('Region')
```

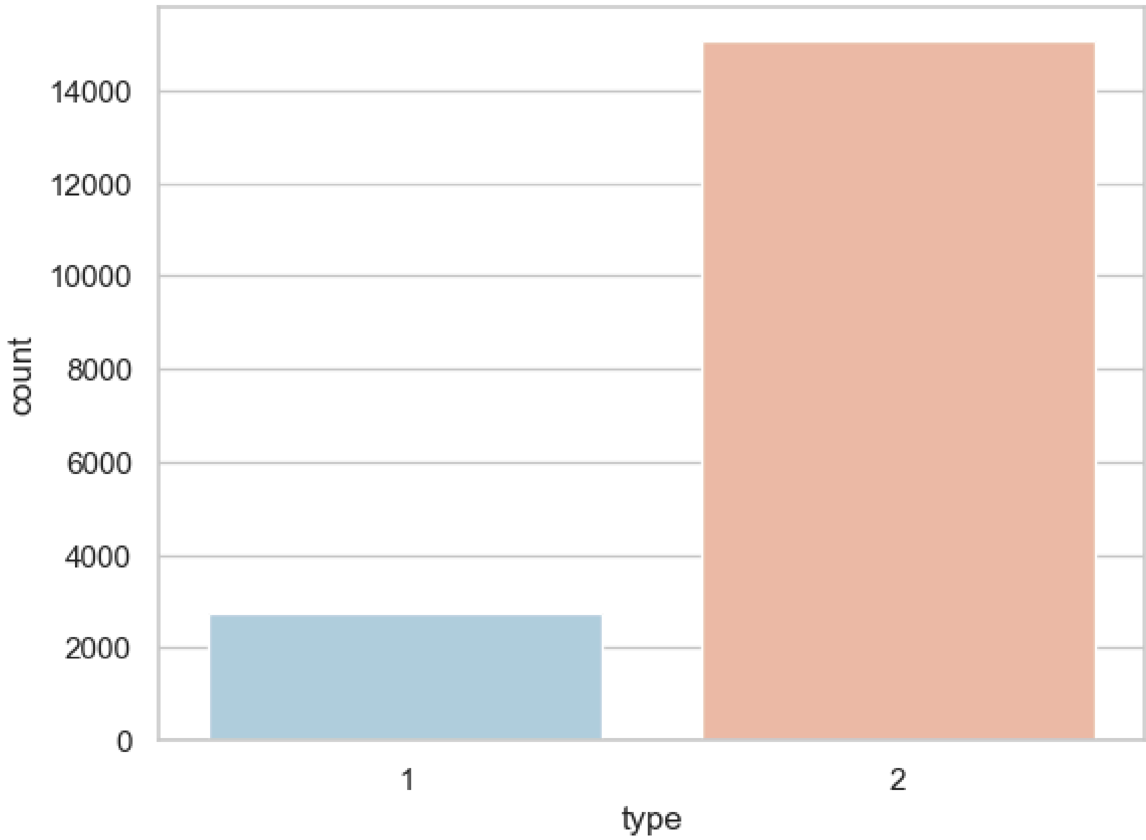
```
plt.ylabel('Total Volume')
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\calvi\AppData\Local\Temp\ipykernel_10720\375233967.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='region', y='Total Volume', data=avocado, ci=None)
```

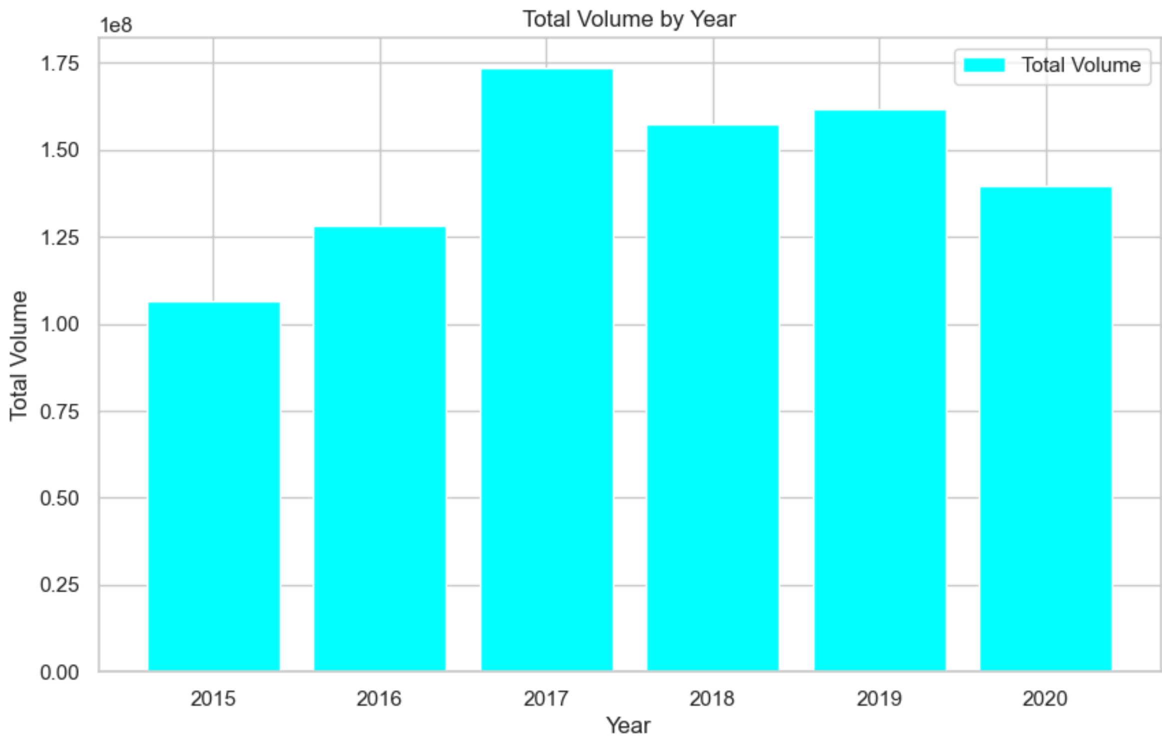


In [43]:
```
sns.set_style('whitegrid')
sns.countplot(x='type',data=avocado,palette='RdBu_r')
```

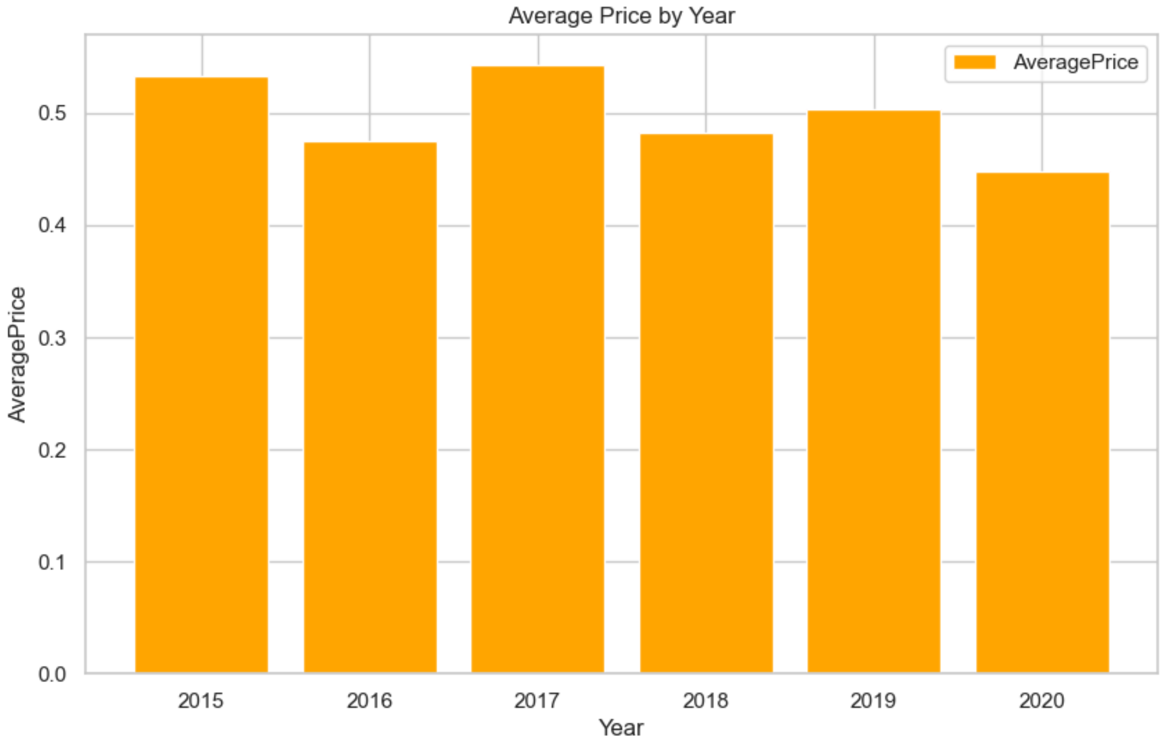
Out[43]: <Axes: xlabel='type', ylabel='count'>



In [44]:
```
grouped_data = avocado.groupby('year').agg({'Total Volume': 'sum', 'AveragePrice

fig, ax1 = plt.subplots(figsize=(10, 6))
ax1.bar(grouped_data['year'], grouped_data['Total Volume'], label='Total Volume'
ax1.set_xlabel('Year')
ax1.set_ylabel('Total Volume')
ax1.set_title('Total Volume by Year')
ax1.legend()
plt.show()
```



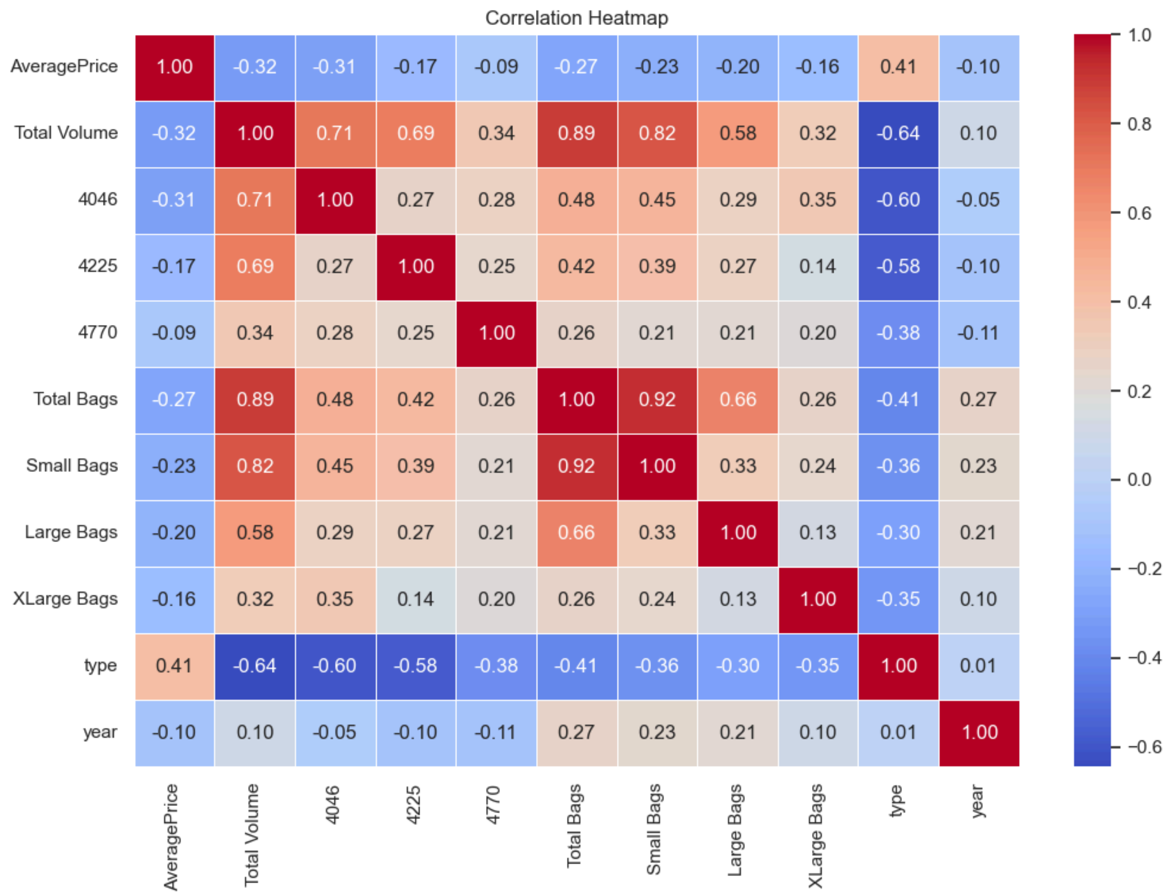
In [45]:
```
fig, ax2 = plt.subplots(figsize=(10, 6))
ax2.bar(grouped_data['year'], grouped_data['AveragePrice'], label='AveragePrice'
ax2.set_xlabel('Year')
ax2.set_ylabel('AveragePrice')
ax2.set_title('Average Price by Year')
ax2.legend()
plt.show()
```



4. MODELLING

In [46]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

numerical_columns = avocado.drop(['Date', 'region'], axis=1)

correlation_matrix = numerical_columns.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidt
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\calvi\AppData\Local\Temp\ipykernel_10720\1358551466.py:7: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
rsion, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
  correlation_matrix = numerical_columns.corr()
```



In [47]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

1. Multiple Regression

In [48]:
```
X = avocado[['AveragePrice', '4046', '4225', '4770', 'type', 'year']]
y = avocado['Total Volume']
y = y.values.reshape(-1, 1)

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_siz
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(14240, 6) (3561, 6)
(14240, 1) (3561, 1)
```

In [49]:
```
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
print('Coefficients: ',model.coef_)
print('Intercept: ',model.intercept_)
```

```
Coefficients:  [[-0.0498538   0.57374552  0.56741675  0.07202317  0.0703294   0.1
9089092]]
Intercept:  [-0.00224199]
```

In [50]:
```
y_pred = model.predict(X_test)
print('Predicted: ', y_pred)
```

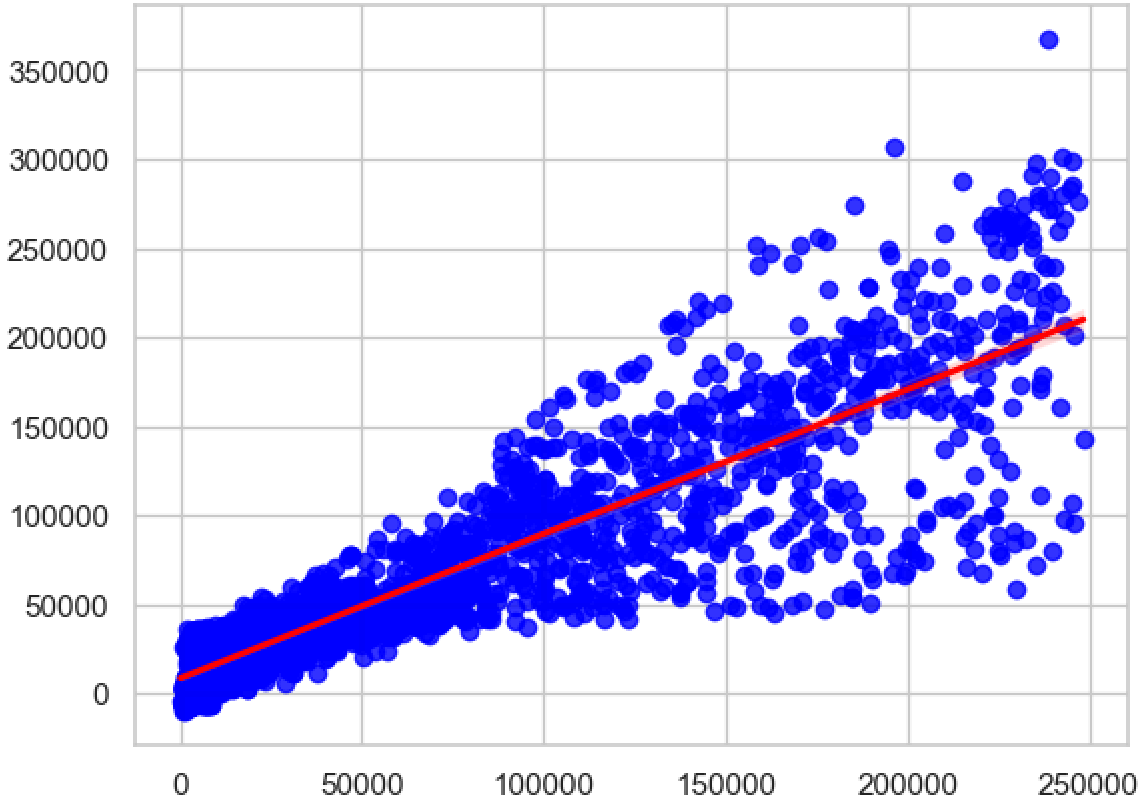
```
Predicted:  [[-0.25308275]
 [ 1.75275325]
 [-0.13618126]
 ...
 [-0.18343706]
 [ 0.90383144]
 [-0.6384793 ]]
```

In [51]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred = model.predict(X_test)

mae_multi = mean_absolute_error(y_test, y_pred)
mse_multi = mean_squared_error(y_test, y_pred)
r2_multi = r2_score(y_test, y_pred)

print("Mean Absolute Error: %.2f" % mae_multi)
print("Mean Squared Error: %.2f" % mse_multi)
print("R-squared:", r2_multi)
```

```
Mean Absolute Error: 0.26
Mean Squared Error: 0.18
R-squared: 0.8143322822814907
```

In [52]:
```
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1,1))
y_pred_inverse = scaler.inverse_transform(y_pred.reshape(-1,1))
```

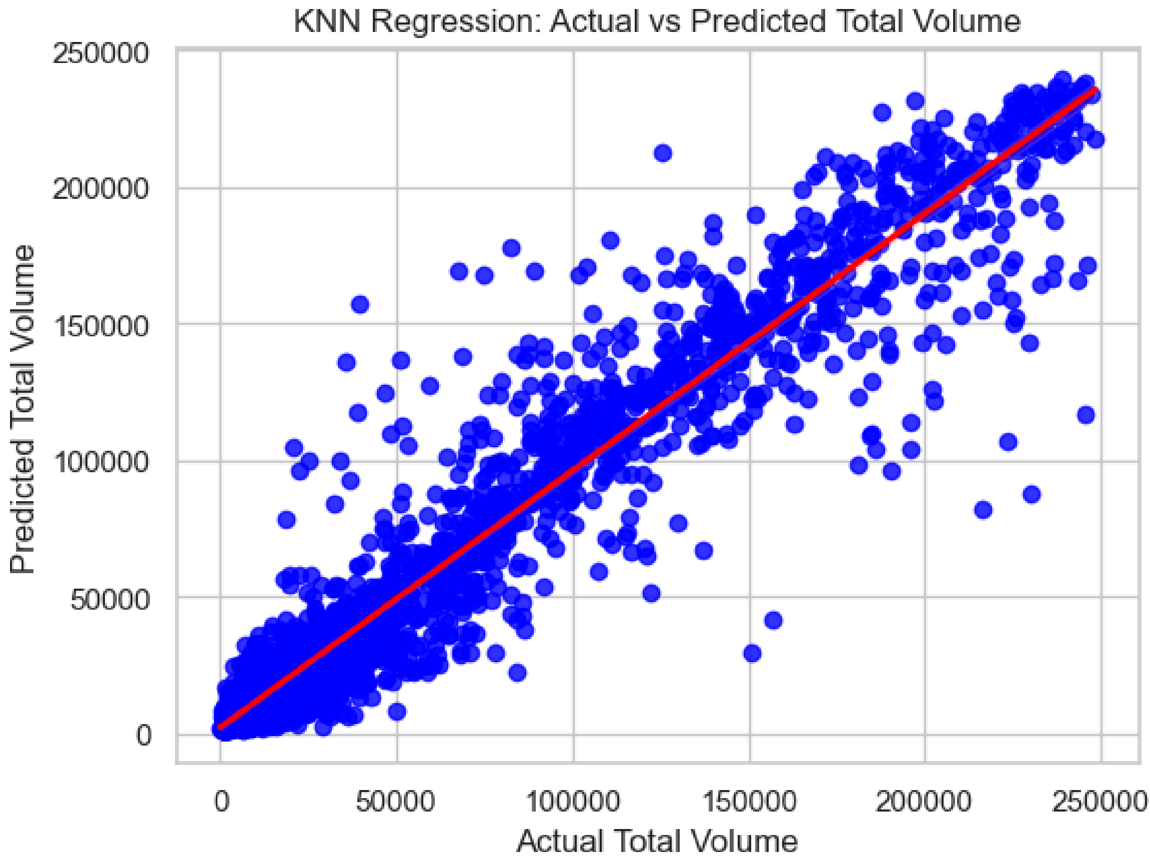
In [53]:
```
line_color = 'red'
ax = sns.regplot(x=y_test_inverse, y=y_pred_inverse, scatter_kws={'color': 'blue
plt.show()
```



In [67]:
```
KNN
# Select independent variables (features) and the dependent variable (target)
X = avocado[['AveragePrice', '4046', '4225', '4770', 'type', 'year']]
y = avocado['Total Volume']

# Reshape y to a 2D array
y = y.values.reshape(-1, 1)

# Standardize the features and target
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_siz

# Create a KNN Regression model
knn_model = KNeighborsRegressor(n_neighbors=5)  # You can adjust the number of n

# Train the model on the training data
knn_model.fit(X_train, y_train)

# Predict the target variable on the test set
y_pred = knn_model.predict(X_test)

# Inverse transform the scaled predictions and true values
y_test_inverse = scaler_y.inverse_transform(y_test)
y_pred_inverse = scaler_y.inverse_transform(y_pred)

# Calculate evaluation metrics
mae_knn = mean_absolute_error(y_test_inverse, y_pred_inverse)
mse_knn = mean_squared_error(y_test_inverse, y_pred_inverse)
```

```
r2_knn = r2_score(y_test_inverse, y_pred_inverse)

# Print the evaluation metrics
print("Mean Absolute Error: %.2f" % mae_knn)
print("Mean Squared Error: %.2f" % mse_knn)
print("R-squared:", r2_knn)

# Plot the predicted vs actual values
line_color = 'red'
ax = sns.regplot(x=y_test_inverse.flatten(), y=y_pred_inverse.flatten(), scatter
plt.xlabel('Actual Total Volume')
plt.ylabel('Predicted Total Volume')
plt.title('KNN Regression: Actual vs Predicted Total Volume')
plt.show()
```

```
Mean Absolute Error: 8345.46
Mean Squared Error: 242068419.00
R-squared: 0.9372185099931346
```



2. Polynomial Regression

In [54]:
```
from sklearn.preprocessing import PolynomialFeatures

polynomial = PolynomialFeatures(degree = 5)
polynomial_train_x = polynomial.fit_transform(X_train)
polynomial_train_x
```

Out[54]:
```
array([[ 1.00000000e+00, -2.20037988e-01, -4.08157893e-01, ...,
          7.37971516e-03,  5.94909850e-03,  4.79581829e-03],
       [ 1.00000000e+00, -3.39822579e-01, -3.66322668e-01, ...,
         -8.54353852e-02, -9.08765886e-03, -9.66643310e-04],
       [ 1.00000000e+00,  7.38238737e-01, -3.69670424e-01, ...,
         -5.38093150e-01,  1.81220912e+00, -6.10322190e+00],
       ...,
       [ 1.00000000e+00, -1.89702226e+00, -3.94808610e-01, ...,
         -1.08768131e-01,  2.14981249e-01, -4.24912491e-01],
       [ 1.00000000e+00, -5.49445612e-01,  3.23026388e+00, ...,
         -1.62859662e+01, -9.96979766e+00, -6.10322190e+00],
       [ 1.00000000e+00,  1.51683858e+00, -4.16894514e-01, ...,
          6.51103994e-01,  2.33667784e+00,  8.38585445e+00]])
```

In [55]:
```
from sklearn.linear_model import LinearRegression

polynomial = PolynomialFeatures(degree=5)
polynomial_train_x = polynomial.fit_transform(X_train)

poly_model = LinearRegression()

poly_model.fit(polynomial_train_x, y_train)

intercept = poly_model.intercept_
coefficients = poly_model.coef_

print("Intercept:", intercept)
print("Coefficients:", coefficients)
```

```
Intercept: [7669.48358512]
Coefficients: [[-5.30406611e+01 -2.08376686e+02  1.40048492e+03  3.03803045e+02
   2.72024784e+04  1.75346193e+04 -2.43226044e+02 -3.29041351e+01
   2.12325798e+02 -3.94485432e+02  5.39722049e+01 -7.97986418e+02
   1.75818629e+02 -1.73417623e+03 -3.70988252e+01  1.72666321e+02
  -1.25930177e+03  4.06394320e+01  1.54830453e+03 -1.04115544e+03
  -1.48277315e+03  2.95438418e+02  2.52576291e+04 -2.21403762e+04
  -2.60622569e+02  2.35047284e+04  7.41352727e+01 -2.70723334e+03
  -3.61888073e+03 -2.94880992e+02 -2.08254449e+02 -9.46155093e+02
  -3.37481017e+03 -1.71344415e+01  4.33927087e+02 -2.17802011e+00
  -5.90163725e+02 -2.64901531e+03 -5.90170081e+00 -3.87052118e+02
  -1.80978602e+02  2.41386834e+02 -2.21573756e+00 -6.74342040e+02
  -4.03933447e+02  2.46411392e+01 -2.26692961e+03 -3.91516604e+02
  -1.50169166e+02  6.91089258e+03 -2.08760990e+01 -2.60179048e+03
   3.13875492e+03 -3.33246644e+01  1.02755605e+02 -2.64740742e+02
   4.37340998e+02 -1.30261785e+01  1.44069161e+02 -6.79579622e+02
  -9.84006905e+01 -3.77964600e+03 -1.53719275e+02  8.81391222e+01
  -1.01847665e+02 -4.03139167e+02  1.71510400e+03 -1.46339705e+00
  -4.63100988e+02  1.51526285e+03 -5.40837470e+01  3.14474051e+03
  -4.48119746e+02 -1.50576255e+02 -1.41395328e+04  1.98147108e+04
   1.48360621e+03 -1.43489234e+04  1.94861143e+02 -1.35605236e+03
  -1.53680581e+04 -6.11755364e+01 -5.40875952e+03 -1.24445899e+02
  -2.41403424e-03 -4.30554180e-02  3.33554448e-02  1.51248982e-02
   6.94528174e+03 -2.47534539e-03  2.68596875e-02  2.98211667e-02
  -3.78279503e-02  5.65790508e+02 -9.45712684e-02 -2.98989327e-02
  -1.09103167e-01  3.99697495e+02 -3.73353706e-02 -4.19439139e-02
   1.81577220e+03  4.12166555e-02  6.63099140e+03  3.28788414e+01
  -7.39059438e-03 -3.82861619e-02  5.11115490e-02 -1.33428254e-01
  -8.32646185e+02  5.05607413e-02 -1.33819064e-01  1.12533573e-01
   4.62017200e+00  5.68002547e-02 -3.45981146e-02  1.13267591e+03
   1.09913316e-01  4.08935286e+03  1.18078242e+01  1.32319532e-01
  -3.89539815e-02 -5.51285332e-02  7.43030336e+02  9.87037070e-02
  -1.77838644e-02  3.47866625e+02  1.08816415e-01  1.38355515e+03
   3.91317239e+00 -5.07059463e-02  8.25763769e-03  1.29425378e+03
   1.95735249e-02  5.22531242e+02 -4.73984612e+01 -5.87583501e-02
   1.03572301e+04 -7.29183461e+01  2.88282975e+02 -1.60684158e-02
   1.23641062e-02  1.13176243e-01 -3.62172369e-03 -1.32631344e+04
   8.69445650e-02  3.90100862e-02 -5.13918206e-02  3.93157777e+01
  -2.36487697e-02  1.05581070e-01  4.99246437e+03 -4.83167475e-02
   2.09656937e+03  6.35357181e+01 -1.16085871e-01  3.48573815e-02
  -9.17758298e-02 -1.97189552e+02  1.76675585e-01 -1.95754547e-01
   5.09270138e+02 -1.33348943e-01 -6.65611073e+02  2.39927957e+01
  -3.26152086e-02  2.41909569e-02 -2.76616961e+02  4.18102396e-02
   4.94434229e+02  1.88195507e+02  2.41403233e-03 -2.11386245e+03
   1.04814067e+02 -1.69278954e+02  3.85016805e-03  2.93894798e-03
   2.91183518e-02  1.95410935e+02 -2.76508295e-02  8.15169176e-02
   7.72570659e+02 -2.03965356e-01 -1.05433125e+04  2.50477702e+00
  -1.13118112e-01  3.52732936e-02  8.88152202e+02 -4.52706756e-02
   1.97260299e+03  1.04158703e+02 -1.50459006e-01 -2.39087487e+03
  -5.20752376e+02  2.89216303e+02 -1.62749122e-02 -3.10160681e-02
   2.71365316e+04  9.89563932e-03 -1.56316955e+05 -2.84728349e+03
   4.68750486e-02 -1.65469767e+05  8.46969234e+02  2.60266581e+03
  -5.75356045e-02 -4.66926822e+05  2.42858753e+03  2.30593483e+04
   2.39060484e+02  1.11922931e-02  2.79618890e-04  1.60272125e-02
   3.23593942e-03 -1.46116640e-02  1.18895936e-02 -8.89889208e-04
   1.01546538e-02 -7.49575170e-03 -3.93847970e-02 -3.39873972e-03
  -6.50393315e-02 -1.30454045e-02  4.85215024e-03 -1.02624207e-02
   6.04270440e-02  6.40331232e-03  1.30062629e-02  9.32159865e-03
   3.61888520e+03  2.05759876e-02 -4.61676246e-03 -4.58793190e-03
  -1.04814150e-03  1.03770448e-02 -6.47880312e-03  1.82627278e-02
```

```
  2.35651814e-04  1.76889381e-02  2.45057265e-02  4.85684188e-02
  2.40230742e-02  7.31073902e-02  3.47380811e-02  2.94786962e+02
  1.46698971e-01 -2.82636750e-02  2.48259517e-03  3.91550052e-02
  7.07686805e-04 -2.61400188e-02  2.25188163e-02  2.59457279e-02
 -1.50472746e-02  2.08275871e+02 -1.30161249e-01 -7.73383942e-02
  3.62208618e-03  1.88869287e-02 -1.47582249e-02  9.46206626e+02
 -1.18572533e-02  2.54676101e-02  3.43796602e+03  1.70986020e+01
 -6.18102991e-02 -1.77446930e-02  2.47428614e-03 -2.59408217e-03
 -1.75718367e-03 -3.93166583e-03  8.89129642e-04  1.20952751e-02
 -1.23121453e-02  2.64304532e-02 -1.88635154e-02  1.55935261e-02
 -5.17955157e-02 -1.99001095e-02 -4.33845655e+02 -3.41681225e-03
 -2.84326063e-02  5.72904636e-03 -2.48659425e-02 -2.62390702e-02
 -2.43606351e-02  2.19913765e-03  2.82020376e-03 -2.95021552e-02
  2.40187124e+00 -6.81904582e-02 -4.59790771e-02  1.37899036e-03
  3.33388159e-03 -3.24926697e-02  5.90179009e+02 -3.98567801e-02
  2.22097358e-02  2.24147559e+03  5.95301135e+00  3.72688806e-02
  2.30254196e-02  5.26974438e-03 -3.96153372e-03  7.82848359e-03
  3.32107675e-03  3.93933958e-03 -4.83472832e-02  3.99604388e-03
  3.87180796e+02  7.04434149e-02  5.20070713e-02 -5.06215186e-03
 -3.47505039e-02 -1.60935042e-02  1.81140566e+02  6.67606684e-02
  7.07710763e-02  5.15449981e+02  2.13204648e+00  1.44878174e-01
 -2.17971148e-02 -1.35246284e-03 -7.56573082e-03  1.39814523e-04
  6.74339602e+02  3.46268284e-03 -3.93848951e-03  3.00327820e+02
 -2.46500380e+01 -5.43969365e-03 -2.53425069e-02  4.77285550e+03
  5.37038935e+01  1.50319495e+02  9.52360712e-02 -1.11006229e-02
 -1.03423379e-03 -9.63612517e-03  1.39216332e-04 -3.39006326e-03
 -4.20549555e-03 -8.46929943e-03  1.16985944e-02  9.03510993e-03
 -6.69125518e-03  3.14565360e-03  5.57650071e-03  7.28680945e-03
 -6.91088377e+03  1.76481888e-02  4.75615463e-03 -2.38048416e-03
  1.50526235e-02 -5.82683507e-03 -4.85140074e-03  1.18487900e-02
  4.10696011e-02  4.47509666e-02  2.05069665e+01  5.99616367e-03
  8.28447698e-03  6.70134635e-04  5.07497726e-02  4.48034701e-03
  2.60145606e+03  3.79935805e-02  1.13276856e-02  1.88915017e+02
  3.31336155e+01 -1.03255063e-02 -4.43614650e-03 -2.84862415e-03
  3.21788769e-03 -3.29965645e-03 -6.95170213e-03  9.75672911e-03
  1.27503457e-02  3.91552260e-02 -1.02791871e+02  8.21772675e-02
  2.43631952e-02  4.89460808e-03 -4.65512389e-02  1.80122200e-02
  2.65341233e+02  1.03694297e-01  3.31078135e-02 -3.66194154e+02
  1.26726156e+01  5.63999683e-02 -2.30457660e-02 -2.53420397e-04
  8.75338290e-03 -3.52292831e-04 -1.44125473e+02  4.96862065e-02
  6.18298307e-03  3.47714885e+02  9.82434271e+01  2.88921098e-02
 -3.92558364e-03 -3.58066477e+02  7.58099656e+01 -8.81420217e+01
 -6.32554148e-03 -4.78679579e-03 -7.41413169e-06  1.15196558e-03
 -4.24364101e-03  1.98674549e-03  4.29723816e-03  2.95953094e-02
  1.58456135e-02  1.01796204e+02 -3.87889428e-03  1.28294844e-02
 -2.45448499e-03  4.03849007e-02  1.56600981e-02  4.02702933e+02
 -8.77183871e-03  1.60765245e-03 -4.68689752e+03  1.34023769e+00
 -3.45610387e-02  7.39200604e-03 -2.87961419e-03 -3.95669452e-04
 -2.54239066e-03  4.62822512e+02  1.20190188e-02  1.78758287e-02
  4.84918708e+02  5.43397731e+01 -5.16399747e-02 -1.39480066e-02
 -1.71591130e+03 -1.17667592e+02  1.50636696e+02  9.77915391e-03
  1.17399510e-02  2.26329385e-03 -5.73506222e-05 -1.75769957e-03
  1.41396869e+04 -8.97121950e-03 -4.79911182e-03 -6.82892393e+04
 -1.48360996e+03  1.29524107e-02  7.20931706e-03 -7.05535483e+04
  3.05493129e+02  1.35615128e+03 -2.53421958e-03 -8.24825096e-02
 -1.94928902e+05  1.06032553e+03  1.06045533e+04  1.24531683e+02
  7.39493599e-04 -4.05034782e-02]]
```

In [56]:

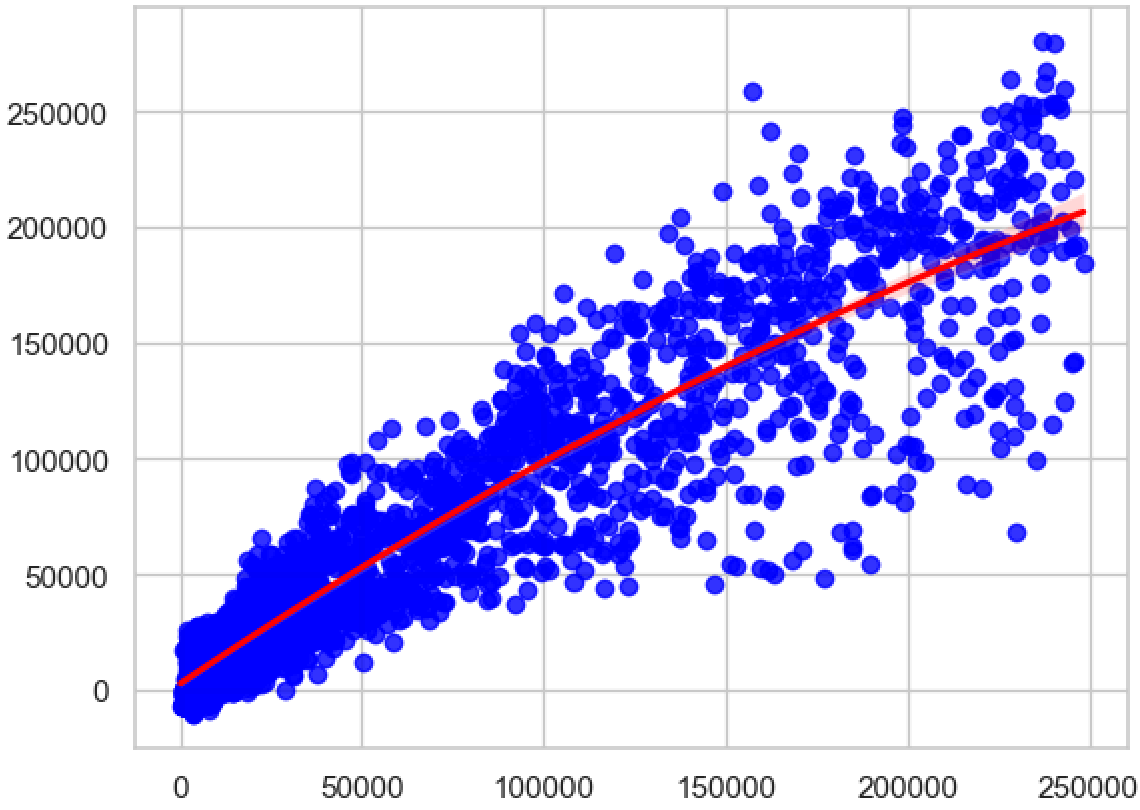
```
poly = PolynomialFeatures()
X_train_poly = poly.fit_transform(X_train)
```

```
X_test_poly = poly.transform(X_test)

poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

y_pred_poly = poly_model.predict(X_test_poly)

mae_poly = mean_absolute_error(y_test, y_pred_poly)
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print("Mean Absolute Error: %.2f" % mae_poly)
print("Mean Squared Error: %.2f" % mse_poly)
print("R-squared:", r2_poly)
```

```
Mean Absolute Error: 0.22
Mean Squared Error: 0.13
R-squared: 0.8730429776166446
```

In [57]:
```
y_test_poly_inverse = scaler.inverse_transform(y_test.reshape(-1,1))
y_pred_poly_inverse = scaler.inverse_transform(y_pred_poly.reshape(-1,1))
```

In [58]:
```
ax = sns.regplot(x=y_test_poly_inverse, y=y_pred_poly_inverse,
                 order = 2, scatter_kws = {"color" : "blue"}, line_kws = {"color"
plt.show()
```



3. Decision Tree Regressor

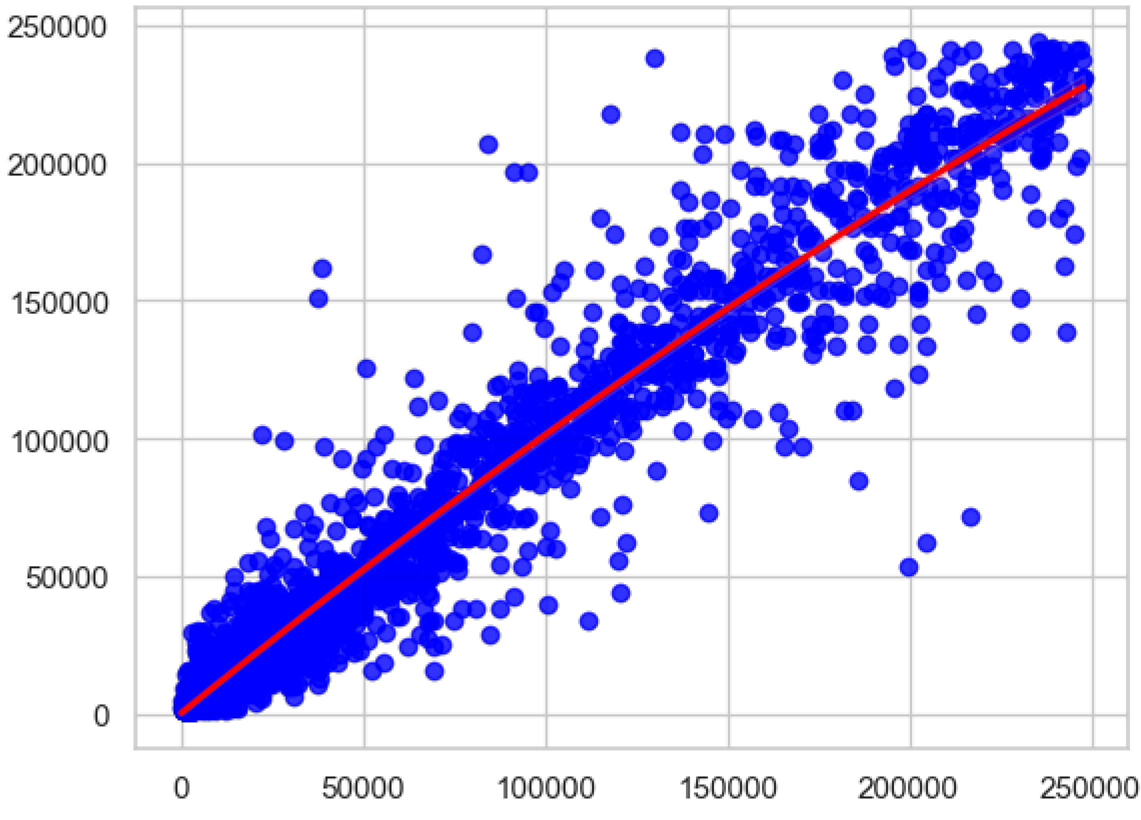
In [59]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.tree import DecisionTreeRegressor, plot_tree

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_siz
```

```
Dt = DecisionTreeRegressor(min_samples_leaf=5, max_depth=20)
Dt.fit(X_train, y_train)
y_preddt = Dt.predict(X_test)

mae_dt = mean_absolute_error(y_test, y_preddt)
mse_dt = mean_squared_error(y_test, y_preddt)
r2_dt = r2_score(y_test, y_preddt)

print("Mean Absolute Error: %.2f" % mae_dt)
print("Mean Squared Error: %.2f" % mse_dt)
print("R-squared:", r2_dt)
```

```
Mean Absolute Error: 0.13
Mean Squared Error: 0.06
R-squared: 0.9412050902744146
```

In [60]:
```
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1,1))
y_preddt_inverse = scaler.inverse_transform(y_preddt.reshape(-1,1))
```

In [61]:
```
ax = sns.regplot(x=y_test_inverse, y=y_preddt_inverse,
                 order=2, scatter_kws={"color":"blue"}, line_kws={"color":"red"})
plt.show()
```



5. Evaluation

In [62]:
```
#1. Multi Linear Regression Evaluation

print("Mean Absolute Error: %.2f" % mae_multi)
print("Mean Squared Error: %.2f" % mse_multi)
print("R-squared:", r2_multi)
```

```
Mean Absolute Error: 0.26
Mean Squared Error: 0.18
R-squared: 0.8143322822814907
```

Mean Absolute Error (MAE): Pada hasil ini, nilai MAE sebesar 0.26 menunjukkan bahwa rata-rata perbedaan absolut antara volume total yang diamati dan volume total yang diprediksi oleh model adalah sekitar 0.26. Semakin rendah nilai MAE, semakin baik model dalam membuat prediksi yang akurat.

Mean Squared Error (MSE): Pada hasil ini, nilai MSE sebesar 0.18 menunjukkan bahwa rata-rata perbedaan kuadrat antara volume total yang diamati dan volume total yang diprediksi oleh model adalah sekitar 0.18. Seperti MAE, semakin rendah nilai MSE, semakin baik model dalam membuat prediksi yang akurat.

R-squared (R^2): Nilai R-squared sebesar 0.8143 menunjukkan bahwa sekitar 81.43% variasi dalam total volume dapat dijelaskan oleh variabel independen yang digunakan dalam model. Nilai R-squared berkisar antara 0 dan 1, di mana nilai 1 menunjukkan model yang sempurna. Semakin tinggi nilai R-squared, semakin baik model dalam menjelaskan variasi dalam data.

In [63]:
```
#2. Polynomial Regression Evaluation

print("Mean Absolute Error: %.2f" % mae_poly)
print("Mean Squared Error: %.2f" % mse_poly)
print("R-squared:", r2_poly)
```

```
Mean Absolute Error: 0.22
Mean Squared Error: 0.13
R-squared: 0.8730429776166446
```

Mean Absolute Error (MAE): Pada hasil ini, nilai MAE sebesar 0.22 menunjukkan bahwa rata-rata perbedaan absolut antara volume total yang diamati dan volume total yang diprediksi oleh model polynomial regression adalah sekitar 0.22. Semakin rendah nilai MAE, semakin baik model dalam membuat prediksi yang akurat.

Mean Squared Error (MSE): Pada hasil ini, nilai MSE sebesar 0.13 menunjukkan bahwa rata-rata perbedaan kuadrat antara volume total yang diamati dan volume total yang diprediksi oleh model polynomial regression adalah sekitar 0.13. Seperti MAE, semakin rendah nilai MSE, semakin baik model dalam membuat prediksi yang akurat.

R-squared (R^2): Nilai R-squared sebesar 0.8730 menunjukkan bahwa sekitar 87.30% variasi dalam total volume dapat dijelaskan oleh variabel independen yang digunakan dalam model polynomial regression. Nilai R-squared yang tinggi menunjukkan bahwa model polynomial regression cukup baik dalam menjelaskan variasi dalam data.

In [64]:
```
#3. Decision Tree Regressor Evaluation

print("Mean Absolute Error: %.2f" % mae_dt)
print("Mean Squared Error: %.2f" % mse_dt)
print("R-squared:", r2_dt)
```

```
Mean Absolute Error: 0.13
Mean Squared Error: 0.06
R-squared: 0.9412050902744146
```

Mean Absolute Error (MAE): Pada hasil ini, nilai MAE sebesar 0.13 menunjukkan bahwa rata-rata perbedaan absolut antara volume total yang diamati dan volume total yang diprediksi oleh model Decision Tree Regressor adalah sekitar 0.13. Semakin rendah nilai MAE, semakin baik model dalam membuat prediksi yang akurat.

Mean Squared Error (MSE): Pada hasil ini, nilai MSE sebesar 0.06 menunjukkan bahwa rata-rata perbedaan kuadrat antara volume total yang diamati dan volume total yang diprediksi oleh model Decision Tree Regressor adalah sekitar 0.06. Seperti MAE, semakin rendah nilai MSE, semakin baik model dalam membuat prediksi yang akurat.

R-squared (R^2): Nilai R-squared sebesar 0.9412 menunjukkan bahwa sekitar 94.12% variasi dalam total volume dapat dijelaskan oleh variabel independen yang digunakan dalam model Decision Tree Regressor. Nilai R-squared yang tinggi menunjukkan bahwa model ini memiliki kemampuan yang baik dalam menjelaskan variasi dalam data.

CONCLUSION

Berdasarkan evaluasi performa tiga algoritma regresi, yaitu Multi Linear Regression, Polynomial Regression, dan Decision Tree Regressor, dapat disimpulkan bahwa Decision Tree Regressor menunjukkan kinerja yang paling baik dalam memprediksi total volume berdasarkan variabel independen yang digunakan. Decision Tree Regressor memberikan nilai Mean Absolute Error (MAE) yang rendah sebesar 0.13, Mean Squared Error (MSE) sebesar 0.06, dan R-squared sebesar 0.9412, menandakan akurasi dan kemampuan model dalam menjelaskan variasi dalam data yang tinggi. Sebagai hasilnya, Decision Tree Regressor merupakan pilihan yang lebih unggul dibandingkan dengan Multi Linear Regression dan Polynomial Regression dalam konteks pemodelan hubungan antara variabel dependen (Total Volume) dengan variabel independen (Average Price, 4046, 4225, 4770, Type, dan Year) pada dataset avocado ini.Berdasarkan evaluasi performa tiga algoritma regresi, yaitu Multi Linear Regression, Polynomial Regression, dan Decision Tree Regressor, dapat disimpulkan bahwa Decision Tree Regressor menunjukkan kinerja yang paling baik dalam memprediksi total volume berdasarkan variabel independen yang digunakan. Decision Tree Regressor memberikan nilai Mean Absolute Error (MAE) yang rendah sebesar 0.13, Mean Squared Error (MSE) sebesar 0.06, dan R-squared sebesar 0.9412, menandakan akurasi dan kemampuan model dalam menjelaskan variasi dalam data yang tinggi. Sebagai hasilnya, Decision Tree Regressor merupakan pilihan yang lebih unggul dibandingkan dengan Multi Linear Regression dan Polynomial Regression dalam konteks pemodelan hubungan antara variabel dependen (Total Volume) dengan variabel independen (Average Price, 4046, 4225, 4770, Type, dan Year) pada dataset avocado ini.

Reference

Input Your Reference Here (Jika ada):

https://www.kaggle.com/datasets/neuromusic/avocado-pricesS