Import Libraries

In [2]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
```

Creating DataFrames

In [3]:
```
list_of_dicts=[
    {"name":"Ginger","breed":"Dachshund","height_cm":22,"weight_kg":10,"date_of_
    {"name":"Scout","breed":"Dalmatian","height_cm":59,"weight_kg":25,"date_of_b
]
new_dogs = pd.DataFrame(list_of_dicts)
new_dogs
```

Out[3]:

	name	breed	height_cm	weight_kg	date_of_birth
0	Ginger	Dachshund	22	10	2019-03-14
1	Scout	Dalmatian	59	25	2019-05-09

In [4]:
```
dict_of_lists ={
    "name":["Ginger","Scout"],
    "breed":["Dachshund","Dalmatian"],
    "height_cm":[22,59],
    "weight_kg":[10,25],
    "date_of_birth":["2019-03-14","2019-05-09"] }
new_dogs = pd.DataFrame(dict_of_lists)
new_dogs
```

Out[4]:

	name	breed	height_cm	weight_kg	date_of_birth
0	Ginger	Dachshund	22	10	2019-03-14
1	Scout	Dalmatian	59	25	2019-05-09

Reading and Writing CSV files

In [5]:
```
avocado = pd.read_csv(r"C:\Users\ankus\Desktop\NareshIT\2. Notes\11.Machine lear
avocado.head()               # first 5 rows returns
```

Out[5]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Sm B
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	567
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	598

Read CSV and assign index

You can assign columns as index using "index_col" attribute.

Since I want to index Date there is another helpful function called "parse_date" which will parse the date in the rows such that we can perform more complex subsetting(eg monthly, weekly etc)

In [6]:
```
avocado = pd.read_csv(r"C:\Users\ankus\Desktop\NareshIT\2. Notes\11.Machine lear
avocado.head()
```

Out[6]:

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

Remove index from dataframe.reset_index(drop)

In [7]:
```
avocado = avocado.reset_index(drop=True)
avocado.head()
```

Out[7]:

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	L
0	0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	9
1	1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	9
2	2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	10
3	3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	13
4	4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	19

To write a CSV file function dataframe.to_csv(FILE_NAME)

In [8]:
```
avocado.to_csv("test_write.csv")
```

Some useful pandas function

In [9]:
```
avocado = pd.read_csv(r"C:\Users\ankus\Desktop\NareshIT\2. Notes\11.Machine lear
avocado.head()
```

Out[9]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Sm B
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	567
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986

In [10]:
```
avocado.tail()
```

Out[10]:

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

In [11]:
```
avocado.info()                          #get concise summary of dataframe
```

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

In [12]:
```
avocado.shape
```

Out[12]: (18249, 14)

In [13]:
```
avocado.describe()                      #statistical description
```

Out[13]:

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

In [14]:
```
avocado.values              #numpy representation of given dataframe
```

Out[14]:
```
array([[0, '2015-12-27', 1.33, ..., 'conventional', 2015, 'Albany'],
       [1, '2015-12-20', 1.35, ..., 'conventional', 2015, 'Albany'],
       [2, '2015-12-13', 0.93, ..., 'conventional', 2015, 'Albany'],
       ...,
       [9, '2018-01-21', 1.87, ..., 'organic', 2018, 'WestTexNewMexico'],
       [10, '2018-01-14', 1.93, ..., 'organic', 2018, 'WestTexNewMexico'],
       [11, '2018-01-07', 1.62, ..., 'organic', 2018, 'WestTexNewMexico']],
      dtype=object)
```

In [15]:
```
avocado.columns
```

Out[15]:
```
Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',
       '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',
       'year', 'region'],
      dtype='object')
```

In [16]:
```
len(avocado.columns)
```

Out[16]:
```
14
```

Sorting

In [17]:
```
# sort values based on "AveragePrice" (ascending) and "year" (descending)
avocado.sort_values(["AveragePrice","year"],ascending = [True,False])
```

Out[17]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	
15261	43	2017-03-05	0.44	64057.04	223.84	4748.88	0.00	5
7412	47	2017-02-05	0.46	2200550.27	1200632.86	531226.65	18324.93	45
15473	43	2017-03-05	0.48	50890.73	717.57	4138.84	0.00	4
15262	44	2017-02-26	0.49	44024.03	252.79	4472.68	0.00	3
1716	0	2015-12-27	0.49	1137707.43	738314.80	286858.37	11642.46	10
...	...	...	...	...	...	...	...	
16720	18	2017-08-27	3.04	12656.32	419.06	4851.90	145.09	
16055	42	2017-03-12	3.05	2068.26	1043.83	77.36	0.00	
14124	7	2016-11-06	3.12	19043.80	5898.49	10039.34	0.00	
17428	37	2017-04-16	3.17	3018.56	1255.55	82.31	0.00	
14125	8	2016-10-30	3.25	16700.94	2325.93	11142.85	0.00	

18249 rows × 14 columns

Subsetting

Subsetting is used to get a slice of the original dataframe

In [20]:
```
# Subsetting columns
avocado["AveragePrice"]
```

Out[20]:
```
0        1.33
1        1.35
2        0.93
3        1.08
4        1.28
         ...
18244    1.63
18245    1.71
18246    1.87
18247    1.93
18248    1.62
Name: AveragePrice, Length: 18249, dtype: float64
```

Subsetting multiple columns

In [22]:
```
avocado[["AveragePrice","Date"]]
```

Out[22]:

	AveragePrice	Date
0	1.33	2015-12-27
1	1.35	2015-12-20
2	0.93	2015-12-13
3	1.08	2015-12-06
4	1.28	2015-11-29
...	...	...
18244	1.63	2018-02-04
18245	1.71	2018-01-28
18246	1.87	2018-01-21
18247	1.93	2018-01-14
18248	1.62	2018-01-07

18249 rows × 2 columns

Subsetting rows

In [29]:
```
avocado["AveragePrice"]<1
```

Out[29]:
```
0        False
1        False
2         True
3        False
4        False
         ...
18244    False
18245    False
18246    False
18247    False
18248    False
Name: AveragePrice, Length: 18249, dtype: bool
```

In [24]:
```
#This will print only the rows with price < 1
avocado[avocado["AveragePrice"]<1]
```

Out[24]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Tot Ba
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.
6	6	2015-11-15	0.99	83453.76	1368.92	73672.72	93.26	8318.
7	7	2015-11-08	0.98	109428.33	703.75	101815.36	80.00	6829.
13	13	2015-09-27	0.99	106803.39	1204.88	99409.21	154.84	6034.
43	43	2015-03-01	0.99	55595.74	629.46	45633.34	181.49	9151.
...	...	...	...	...	...	...	...	
17169	43	2017-03-05	0.99	155011.12	35367.23	5175.81	5.91	114462.
17170	44	2017-02-26	0.99	171145.00	34520.03	6936.39	0.00	129688.
17536	39	2017-04-02	0.98	402676.23	34093.33	58330.53	207.85	310044.
17537	40	2017-03-26	0.90	456645.91	36169.35	51398.72	139.55	368938.
17540	43	2017-03-05	0.99	367519.17	61166.48	55123.99	126.80	251101.

2796 rows × 14 columns

Subsetting based on text data

In [26]:
```
# it will print all rows with "type"="organic"
avocado[avocado["type"]=="organic"]
```

Out[26]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
9126	0	2015-12-27	1.83	989.55	8.16	88.59	0.00	892.80	
9127	1	2015-12-20	1.89	1163.03	30.24	172.14	0.00	960.65	
9128	2	2015-12-13	1.85	995.96	10.44	178.70	0.00	806.82	
9129	3	2015-12-06	1.84	1158.42	90.29	104.18	0.00	963.95	
9130	4	2015-11-29	1.94	831.69	0.00	94.73	0.00	736.96	
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	1
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	1
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	1

9123 rows × 14 columns

Subsetting based on dates

In [27]:
```
#  it will print all rows with "Date" <= 2015-02-04
avocado[avocado["Date"]<="2015-02-04"]
```

Out[27]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
47	47	2015-02-01	0.99	70873.60	1353.90	60017.20	179.32	9323.18
48	48	2015-01-25	1.06	45147.50	941.38	33196.16	164.14	10845.82
49	49	2015-01-18	1.17	44511.28	914.14	31540.32	135.77	11921.05
50	50	2015-01-11	1.24	41195.08	1002.85	31640.34	127.12	8424.77
51	51	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46
...	...	...	...	...	...	...	...	...
11928	46	2015-02-01	1.77	7210.19	1634.42	3012.44	0.00	2563.33
11929	47	2015-01-25	1.63	7324.06	1934.46	3032.72	0.00	2356.88
11930	48	2015-01-18	1.71	5508.20	1793.64	2078.72	0.00	1635.84
11931	49	2015-01-11	1.69	6861.73	1822.28	2377.54	0.00	2661.91
11932	50	2015-01-04	1.64	6182.81	1561.30	2958.17	0.00	1663.34

540 rows × 14 columns

Subsetting using .isin()

In [30]:
```
# subset the avocado in the region Boston or SanDiego
regionFilter = avocado["region"].isin(["Boston","SanDiego"])
avocado[regionFilter]
```

Out[30]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Tot Ba
208	0	2015-12-27	1.13	450816.39	3886.27	346964.70	13952.56	86012.
209	1	2015-12-20	1.07	489802.88	4912.37	390100.99	5887.72	88901.
210	2	2015-12-13	1.01	549945.76	4641.02	455362.38	219.40	89722.
211	3	2015-12-06	1.02	488679.31	5126.32	407520.22	142.99	75889.
212	4	2015-11-29	1.19	350559.81	3609.25	272719.08	105.86	74125.
...	...	...	...	...	...	...	...	
18100	7	2018-02-04	1.81	17454.74	1158.41	7388.27	0.00	8908.
18101	8	2018-01-28	1.91	17579.47	1145.64	8284.41	0.00	8149.
18102	9	2018-01-21	1.95	18676.37	1088.49	9282.37	0.00	8305.
18103	10	2018-01-14	1.81	21770.02	3285.98	14338.52	0.00	4145.
18104	11	2018-01-07	2.06	16746.82	5150.82	9366.31	0.00	2229.

676 rows × 14 columns

Multiple parameter Filtering

Use logical operators to combine different filters

In [32]:
```
print(avocado["year"].dtype)
```

```
int64
```

In [34]:
```
# subset the avocado in the region Boston or SanDiego in the year 2016 or 2017
regionFilter = avocado["region"].isin(["Boston","SanDiego"])
yearFilter = avocado["year"].isin([2016,2017])
avocado[regionFilter & yearFilter]
```

Out[34]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Tota Bag
3016	0	2016-12-25	1.28	447600.75	4349.63	346516.32	4183.69	92551.1
3017	1	2016-12-18	1.09	579577.33	6123.84	488107.01	7765.43	77581.0
3018	2	2016-12-11	1.22	510800.58	3711.20	409645.98	5052.84	92390.5
3019	3	2016-12-04	1.26	473428.36	4371.95	393748.18	3449.16	71859.0
3020	4	2016-11-27	1.45	391257.01	4243.20	317090.39	3069.37	66854.0
...	...	...	...	...	...	...	...	.
16962	48	2017-01-29	1.21	18191.46	1477.75	8949.53	4.86	7759.3
16963	49	2017-01-22	1.73	10842.77	2019.23	6869.87	0.00	1953.6
16964	50	2017-01-15	1.82	11578.42	2529.20	7637.66	0.00	1411.5
16965	51	2017-01-08	1.52	16775.97	2363.28	9429.06	0.00	4983.6
16966	52	2017-01-01	1.45	15752.25	1385.18	8618.28	0.00	5748.7

420 rows × 14 columns

Detecting missing values .isna()

In [35]:
```
avocado.isna()
```

Out[35]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
18244	False	False	False	False	False	False	False	False	False	False
18245	False	False	False	False	False	False	False	False	False	False
18246	False	False	False	False	False	False	False	False	False	False
18247	False	False	False	False	False	False	False	False	False	False
18248	False	False	False	False	False	False	False	False	False	False

18249 rows × 14 columns

We can use .any() function to get a consise info

In [36]:
```
avocado.isna().any()
```

Out[36]:
```
Unnamed: 0      False
Date            False
AveragePrice    False
Total Volume    False
4046            False
4225            False
4770            False
Total Bags      False
Small Bags      False
Large Bags      False
XLarge Bags     False
type            False
year            False
region          False
dtype: bool
```

Counting missing values

In [37]:
```
avocado.isna().sum()
```

Out[37]:
```
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

Removing missing values

In [38]:
```
# Luckily we don't have any NaN but if we have we have we can use any of the two

avocado.dropna()

# **** OR ****

meanVal = avocado["AveragePrice"].mean()
avocado.fillna(meanVal)
```

Out[38]:

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

18249 rows × 14 columns

Adding a new column

In [39]:
```
avocado["AveragePricePer100"]= avocado["AveragePrice"]*100
avocado
```

Out[39]:

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

18249 rows × 15 columns

Deleting columns in DataFrame .drop(lst,axis=1)

In [40]:
```
avocado.drop(["AveragePricePer100"],axis=1)
```

Out[40]:

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

18249 rows × 14 columns

Summary statistics

In [41]:
```
# mean of the AveragePrice of avocado
avocado["AveragePrice"].mean()
```

Out[41]:
```
np.float64(1.405978409775878)
```

Summarizing dates

In [42]:
```
avocado["Date"].max()
```

Out[42]:
```
'2018-03-25'
```

.agg() method

Pandas Series.agg() is used to pass a function or list of function to be applied on a series or even each element of series separately

In [44]:
```
def pct30(column):
    #return the 0.3 quartile
    return column.quantile(0.3)
def pct50(column):
    #return the 0.5 quartile
    return column.quantile(0.5)

avocado[["AveragePrice","Total Bags"]].agg([pct30,pct50])
```

Out[44]:

	AveragePrice	Total Bags
pct30	1.15	7316.634
pct50	1.37	39743.830

Dropping duplicate names .drop_duplicates(lst)

In [45]:
```
temp = avocado.drop_duplicates(subset=["year"])
temp
```

Out[45]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87
2808	0	2016-12-25	1.52	73341.73	3202.39	58280.33	426.92	11432.09
5616	0	2017-12-31	1.47	113514.42	2622.70	101135.53	20.25	9735.94
8478	0	2018-03-25	1.57	149396.50	16361.69	109045.03	65.45	23924.33

Count categorical data .value_counts()

In [46]:
```
# count number of avacado in each year in descending order
avocado["year"].value_counts(sort=True,ascending = False)
```

Out[46]:
```
year
2017    5722
2016    5616
2015    5615
2018    1296
Name: count, dtype: int64
```

Grouped summaries .groupby(col)

This function will group similar categories into one and then we can perform some summary statistics

In [51]:
```
# group by multiple columns and perform multiple summary statistic operations
avocado.groupby(["year","type"])["AveragePrice"].agg(["min","max","mean","median
```

Out[51]:

		min	max	mean	median
year	**type**				
2015	**conventional**	0.49	1.59	1.077963	1.08
	organic	0.81	2.79	1.673324	1.67
2016	**conventional**	0.51	2.20	1.105595	1.08
	organic	0.58	3.25	1.571684	1.53
2017	**conventional**	0.46	2.22	1.294888	1.30
	organic	0.44	3.17	1.735521	1.72
2018	**conventional**	0.56	1.74	1.127886	1.14
	organic	1.01	2.30	1.567176	1.55

Pivot table

A pivot table is a table of statistics that summarizes the data of a more extensive table

In [50]:
```
# this is the same table we build in the previous cell but using pivot table
avocado.pivot_table(index=["year","type"],aggfunc=["min","max","mean","median"],
```

Out[50]:

		min	max	mean	median
		AveragePrice	**AveragePrice**	**AveragePrice**	**AveragePrice**
year	**type**				
2015	**conventional**	0.49	1.59	1.077963	1.08
	organic	0.81	2.79	1.673324	1.67
2016	**conventional**	0.51	2.20	1.105595	1.08
	organic	0.58	3.25	1.571684	1.53
2017	**conventional**	0.46	2.22	1.294888	1.30
	organic	0.44	3.17	1.735521	1.72
2018	**conventional**	0.56	1.74	1.127886	1.14
	organic	1.01	2.30	1.567176	1.55

Explicit indexes

Indexes makes subsetting simpler using .loc and .iloc

Setting column as the index

In [52]:
```
regionIndex = avocado.set_index(["region"])
regionIndex
```

Out[52]:

region	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4
Albany	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48
Albany	1	2015-12-20	1.35	54876.98	674.28	44638.81	58
Albany	2	2015-12-13	0.93	118220.22	794.70	109149.67	130
Albany	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72
Albany	4	2015-11-29	1.28	51039.60	941.48	43838.39	75
...	...	...	...	...	...	...	
WestTexNewMexico	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0
WestTexNewMexico	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0
WestTexNewMexico	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727
WestTexNewMexico	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727
WestTexNewMexico	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224

18249 rows × 14 columns

In [53]:
```
 #Instead of doing this
avocado[avocado["region"].isin(["Albany","WestTexNewMexico"])]
```

Out[53]:

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

673 rows × 15 columns

In [55]:
```
# we can simply do
regionIndex.loc[["Albany","WestTexNewMexico"]]
```

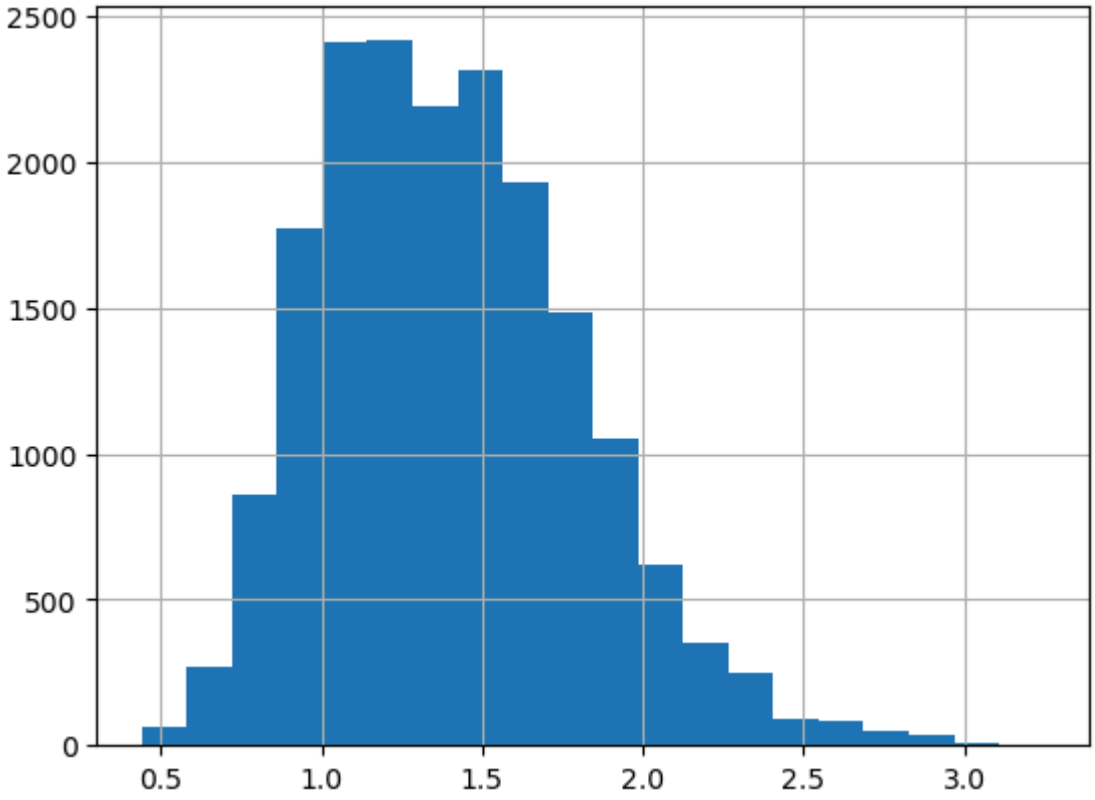
Out[55]:

region	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4
Albany	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48
Albany	1	2015-12-20	1.35	54876.98	674.28	44638.81	58
Albany	2	2015-12-13	0.93	118220.22	794.70	109149.67	130
Albany	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72
Albany	4	2015-11-29	1.28	51039.60	941.48	43838.39	75
...	...	...	...	...	...	...	
WestTexNewMexico	7	2018-02-04	1.63	17074.83	2046.96	1529.20	
WestTexNewMexico	8	2018-01-28	1.71	13888.04	1191.70	3431.50	
WestTexNewMexico	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727
WestTexNewMexico	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727
WestTexNewMexico	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224

673 rows × 14 columns

Visualizing your data

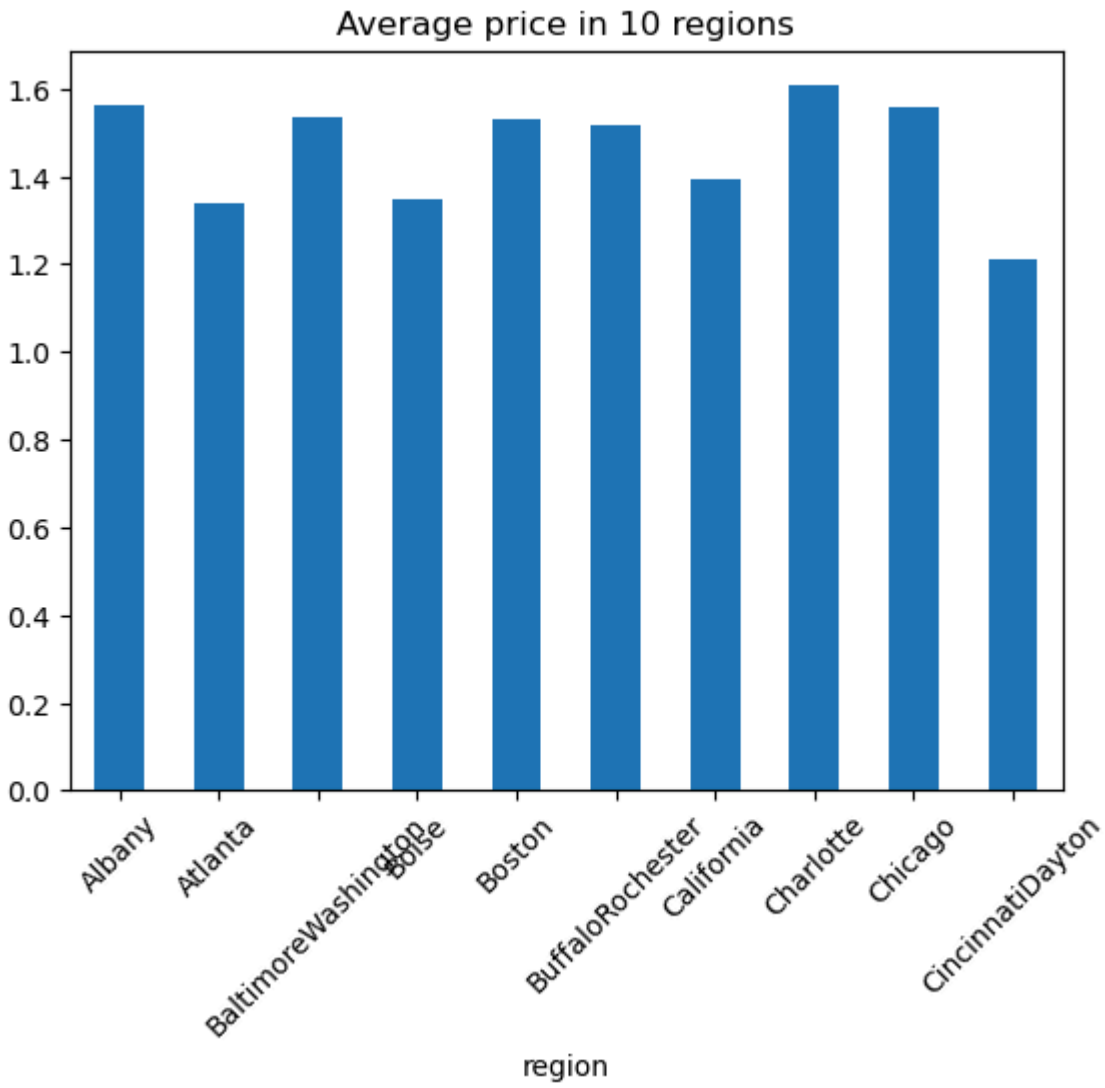
In [57]:
```
#Histograms
avocado["AveragePrice"].hist(bins=20)
plt.show()
```



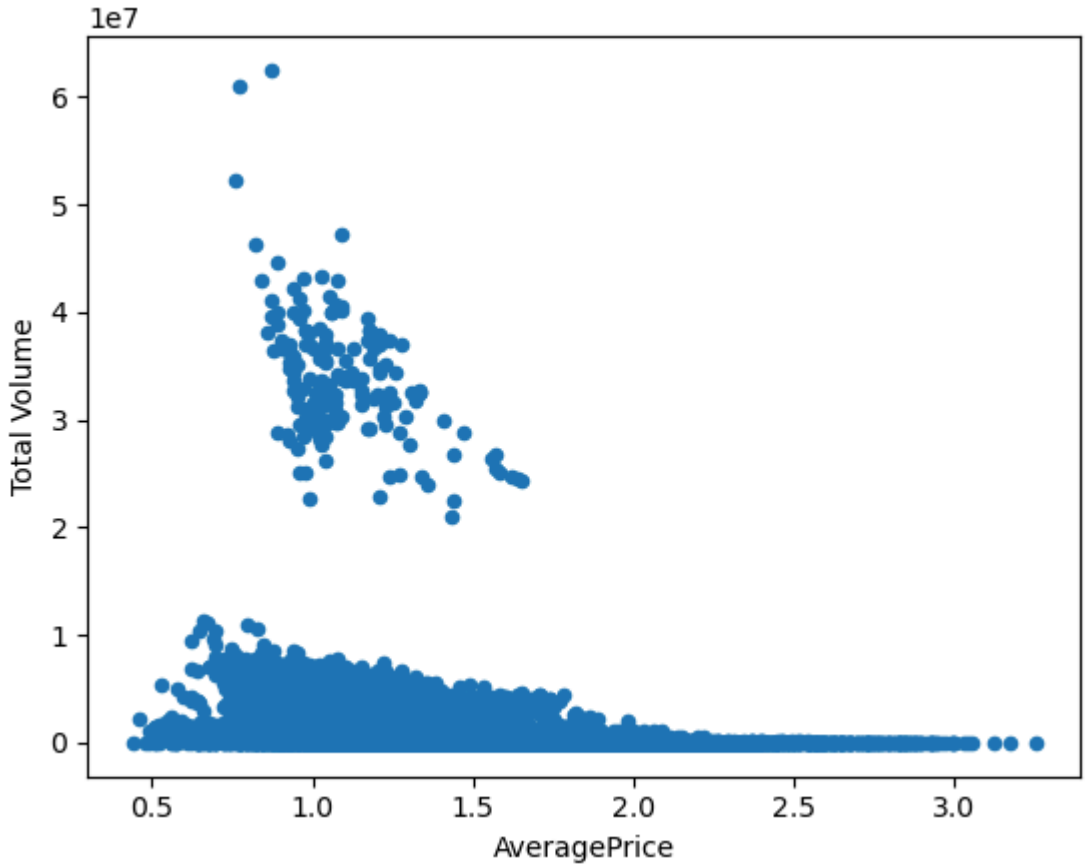
In [58]:
```
#Bar plots
regionFilter = avocado.groupby("region")["AveragePrice"].mean().head(10)
regionFilter
```

Out[58]:
```
region
Albany                  1.561036
Atlanta                 1.337959
BaltimoreWashington     1.534231
Boise                   1.348136
Boston                  1.530888
BuffaloRochester        1.516834
California              1.395325
Charlotte               1.606036
Chicago                 1.556775
CincinnatiDayton        1.209201
Name: AveragePrice, dtype: float64
```

In [60]:
```
regionFilter.plot(kind = "bar",rot=45,title="Average price in 10 regions")
plt.show()
```



In [63]:
```
# Scatter plot
avocado.plot(x="AveragePrice",y="Total Volume",kind="scatter")
plt.show()
```



Arithmetic with Series & DataFrames

In [64]:
```
# subtract Averages with AveragePricee : P
# Dah its 0
avocado["AveragePrice"].sub(avocado["AveragePrice"])
```

Out[64]:
```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
18244    0.0
18245    0.0
18246    0.0
18247    0.0
18248    0.0
Name: AveragePrice, Length: 18249, dtype: float64
```

In [ ]: