Analytics and Systems of Big Data Practice

PROBLEM SET 3

Bhaavanaa Thumu - CED17I021

Question 1

Problem statement :

1. Suppose that the data for analysis includes the attribute age. The age values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

(a) Use min-max normalization to transform the values of age to the range [0:1].

(b) Use z-score normalization to transform the values of age.

(c) Use normalization by decimal scaling to transform the values of age such that the transformed value is less than 1.

Logic used :

Compute the mean and std dev using the statistics library and compute the min-max normalization and z-score normalization using the obtained values. Compute the power of 10 just higher than the largest element and compute normalization by decimal using that.

Packages used :

statistics - to compute the mean and std dev

Code snippets :

```python
import statistics

age = [13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70]

min_max_normalization = []
z_score_normalization = []
decimal_scaling = []

mean = statistics.mean(age)
std_dev = statistics.stdev(age)

div = 1
while(div < age[len(age)-1]):
    div = div*10

deno = age[(len(age)-1)] - age[0]

for i in range(len(age)):
    min_max_normalization.append(round((age[i]-age[0])/deno, 6))
    z_score_normalization.append(round((age[i]-mean)/std_dev, 6))
    decimal_scaling.append(round(age[i]/div, 6))

print("min max normalization : ", min_max_normalization, "\n")
print("z-score normalization : ", z_score_normalization, "\n")
print("normalization by decimal scaling : ", decimal_scaling, "\n")

```

Detailed output :

```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_3$ python3 Q1.py
min max normalization :  [0.0, 0.035088, 0.052632, 0.052632, 0.105263, 0.122807, 0.122807, 0.140351, 0.157895,
 0.157895, 0.210526, 0.210526, 0.210526, 0.210526, 0.298246, 0.350877, 0.350877, 0.385965, 0.385965, 0.385965,
 0.385965, 0.403509, 0.473684, 0.561404, 0.578947, 0.684211, 1.0]

z-score normalization :  [-1.310678, -1.156144, -1.078877, -1.078877, -0.847076, -0.769809, -0.769809, -0.6925
42, -0.615275, -0.615275, -0.383474, -0.383474, -0.383474, -0.383474, 0.002862, 0.234663, 0.234663, 0.389197,
0.389197, 0.389197, 0.389197, 0.466464, 0.775532, 1.161868, 1.239135, 1.702737, 3.093545]

normalization by decimal scaling :  [0.13, 0.15, 0.16, 0.16, 0.19, 0.2, 0.2, 0.21, 0.22, 0.22, 0.25, 0.25, 0.2
5, 0.25, 0.3, 0.33, 0.33, 0.35, 0.35, 0.35, 0.35, 0.36, 0.4, 0.45, 0.46, 0.52, 0.7]

dell@bhaavanaa:~/semester_8/big_data/lab/lab_3$
```

Question 2

Problem statement :

Use the given dataset and perform the operations listed below.

a. Sort the attribute “Total Volume” in the given dataset and distribute the data into equal sized/frequency bins. Let the number of bins be 250. Smooth the sorted data by
(i)bin-means

(ii) bin-medians

(iii) bin-boundaries

b. The dataset represents weekly retail scan data for National retail volume (units) and price. However, the company is interested in knowing the monthly (total per month) and annual sales (total per year), rather than the total per week. So, reduce the data accordingly.

c. Summarize the number of missing values for each attribute

d. Populate data for the missing values of the attribute= “Average Price” by averaging all the values of the “Avg Price” attribute that fall under the same “REGION” attribute value.

e. Discretize the attribute= “Date” using concept hierarchy into {Old, New, Recent} (Consider 2015,2016 : Old, 2017: New, 2018: Recent).

Logic used :

a) Take the total volume column in a separate list and calculate the bin size given the number of bins. Sort the list. Initialize the bins for mean, median and boundary. For mean bin, compute the mean of the elements of the bin and replace all the elements of the bin with the mean. Similarly for median also. For the boundaries bin, for each element in the bin, check whether it is closer to the first element of the bin or the last element and replace it with the closer one.

b) Based on the month-year and the region, find the avg of the avg price and the sum of the other quantities and reduce the data accordingly.
c) We must count the missing values for every column of the dataset. This can be done by checking if the element is empty or if there is “not available”. Empty elements can be checked by string comparison with ‘’, while not available can be checked for a few columns by seeing if the element is a floating point number or not. If it is a floating point number, then the value is available, otherwise the count of the respective column is incremented. Display the counts of the not available columns.
d) Check which value is missing and replace it with the average price by calculating for the same region.
e) Check the year from the year column and update the date column accordingly. 2015 and 2016 - old; 2017 - new; 2018 - recent.

Packages used :

csv - to read the dataset

math - to compute ceil and floor

numpy - for working with arrays

Code snippets and output :

Reading the dataset into a variable.

```
### QUESTION 2 - Use the given dataset and perform the operations listed below.


import csv                                              # import the required libraries
import math
import numpy as np


with open("Avocado_Dataset.csv", 'r') as f:             # open the dataset in read mode
    data = list(csv.reader(f, delimiter = ","))         # read the dataset into the variable data

```

a)

```
## a) Sort the attribute “Total Volume” in the given dataset and distribute the data into equal sized/frequency bins.
## Let the number of bins be 250. Smooth the sorted data by (i)bin-means, (ii) bin-medians, (iii) bin-boundaries

tvol = []                                           # initialize list to store the total volume column
for i in range(1, len(data)):                       # convert the elements under the total volume column to float
    tvol.append(float(data[i][2]))

total_volume = np.array(tvol)                       # convert the list to float
total_volume = np.sort(total_volume)                # sort the total_volume

no_of_bins = 250                                    # given that the no. of bins is 250
bin_size = math.ceil(len(total_volume) / no_of_bins)  # so, bin size can be found by total no. of elements / no. of bins

bin1 = np.zeros((no_of_bins, bin_size))             # create bin1 for mean
bin2 = np.zeros((no_of_bins, bin_size))             # create bin2 for median
bin3 = np.zeros((no_of_bins, bin_size))             # create bin3 for boundaries

for i in range (0, no_of_bins*bin_size, bin_size):  # bin mean
    k = int(i/bin_size)                             # required for storing the index of the bin
    mean = sum(total_volume[i:i+bin_size])/bin_size # obtaining the mean of the elements present in the bin
    for j in range(bin_size):                       # replace the elements of the bin with the mean
        bin1[k,j] = round(mean, 6)
print("Bin Mean: \n",bin1)                          # print the mean bin

for i in range (0, no_of_bins*bin_size, bin_size):  # bin median
    k = int(i/bin_size)                             # required for storing the index of the bin
    for j in range (bin_size):                      # replace the elements of the bin with the median
        bin2[k,j] = total_volume[i+math.floor(bin_size/2)]  # compute the median
print("Bin Median: \n",bin2)                        # print the median bin

for i in range (0, no_of_bins*bin_size, bin_size):  # bin boundaries
    k = int(i/bin_size)                             # required for storing the index of the bin
    for j in range (bin_size):                      # replace the elements of the bin with the closest boundary element
        if (total_volume[i+j]-total_volume[i]) < (total_volume[i+bin_size-1]-total_volume[i+j]):
            bin3[k,j] = total_volume[i]             # if the element is closer to the start element
        else:
            bin3[k,j] = total_volume[i+bin_size-1]  # if the element is closer to the last element
print("Bin Boundaries: \n",bin3)                    # print the boundaries bin
```

b)

```
## b) The dataset represents weekly retail scan data for National retail volume (units) and price.
## However, the company is interested in knowing the monthly (total per month) and annual sales (total per year),
## rather than the total per week. So, reduce the data accordingly.

date = data[1][0][-7:]                                  # for obtaining the month-year from date column - first date of dataset
region = data[1][12]                                    # for obtaining region - first region of dataset
c = 0                                                   # for count - calculation of avg price per month

list_date = []                                          # initialize the required lists
list_avg_price = []
list_total_volume = []
list_4046 = []
list_4225 = []
list_4770 = []
list_total_bags = []
list_small_bags = []
list_large_bags = []
list_xlarge_bags = []
list_region = []

sum_avg_price = 0                                       # initialize the sum variables to zero
sum_total_volume = 0
sum_4046 = 0
sum_4225 = 0
sum_4770 = 0
sum_total_bags = 0
sum_small_bags = 0
sum_large_bags = 0
sum_xlarge_bags = 0

for i in range(1, len(data)):
    if(date == data[i][0][-7:] and region == data[i][12]):      # if the month-year and region match with initialized values
        d = data[i][1]                                          # for checking if the avg_price is float or Nan
        if(d.replace('.', '', 1).isdigit() == True):
            sum_avg_price += float(data[i][1])                  # if float - add to the sum_avg_price, else ignore that cell
            c += 1                                              # and increment the count
        sum_total_volume += float(data[i][2])                   # update the respective sum values
        sum_4046 += float(data[i][3])
        sum_4225 += float(data[i][4])
        sum_4770 += float(data[i][5])
        sum_total_bags += float(data[i][6])
        sum_small_bags += float(data[i][7])
        sum_large_bags += float(data[i][8])
        sum_xlarge_bags += float(data[i][9])
    else:                                                       # if month-year doesnt match with the initialized values
        list_date.append(date)                                  # append the computed data to the list initialized
        if(c != 0):                                             # if all values of avg_price is Nan
            list_avg_price.append(sum_avg_price/c)              # avg price is coputed
        else:
            list_avg_price.append(0)                            # else avg_price for the month = 0
        list_total_volume.append(sum_total_volume)              # append the other computed sums to the respective lists
        list_4046.append(sum_4046)
        list_4225.append(sum_4225)
        list_4770.append(sum_4770)
        list_total_bags.append(sum_total_bags)
        list_small_bags.append(sum_small_bags)
        list_large_bags.append(sum_large_bags)
        list_xlarge_bags.append(sum_xlarge_bags)
        list_region.append(region)
        date = data[i][0][-7:]                                  # initilaize the modified values, as it is different from previous one
        region = data[i][12]
        d = data[i][1]
        if(d.replace('.', '', 1).isdigit() == True):
            sum_avg_price = float(data[i][1])
            c = 1
        else:
            sum_avg_price = 0
            c = 0
        sum_total_volume = float(data[i][2])
        sum_4046 = float(data[i][3])
        sum_4225 = float(data[i][4])
        sum_4770 = float(data[i][5])
        sum_total_bags = float(data[i][6])
        sum_small_bags = float(data[i][7])
        sum_large_bags = float(data[i][8])
        sum_xlarge_bags = float(data[i][9])
```

Line 113, Column 29

```
print("Date     avg_price   total_volume    4046    4225    4770    total_bags  small_bags  large_bags  xlarge_bags     list_region")
for i in range(len(list_date)):
    print(list_date[i], "   ", round(list_avg_price[i], 2), "   ", round(list_total_volume[i], 2), "    ", round(list_4046[i], 2), "    ",
        round(list_4225[i], 2), "   ", round(list_4770[i], 2), "    ", round(list_total_bags[i], 2), "  ", round(list_small_bags[i], 2), "  ",
        round(list_large_bags[i], 2), " ", round(list_xlarge_bags[i], 2), " ", list_region[i])
```

c)

```
## c) Summarize the number of missing values for each attribute

col0 = 0                                                # initialize the count for all columns to 0
col1 = 0
col2 = 0
col3 = 0
col4 = 0
col5 = 0
col6 = 0
col7 = 0
col8 = 0
col9 = 0
col10 = 0
col11 = 0
col12 = 0

for i in range(1, len(data)):
    d0 = data[i][0]                                     # store the required values for checking if the string is float
    d1 = data[i][1]
    d2 = data[i][2]
    d3 = data[i][3]
    d4 = data[i][4]
    d5 = data[i][5]
    d6 = data[i][6]
    d7 = data[i][7]
    d8 = data[i][8]
    d9 = data[i][9]
    if(d0 == ''):                                       # if the values are not available, incerement count for that column
        print(d0)
        col0+=1
    elif(d1 == '' or d1.replace('.', '', 1).isdigit() == False):
        col1+=1
    elif(d2 == '' or d2.replace('.', '', 1).isdigit() == False):
        col2+=1
    elif(d3 == '' or d3.replace('.', '', 1).isdigit() == False):
        col3+=1
    elif(d4 == '' or d4.replace('.', '', 1).isdigit() == False):
        col4+=1
    elif(d5 == '' or d5.replace('.', '', 1).isdigit() == False):
        col5+=1
    elif(d6 == '' or d6.replace('.', '', 1).isdigit() == False):
        col6+=1
    elif(d7 == '' or d7.replace('.', '', 1).isdigit() == False):
        col7+=1
    elif(d8 == '' or d8.replace('.', '', 1).isdigit() == False):
        col8+=1
    elif(d9 == '' or d9.replace('.', '', 1).isdigit() == False):
        col9+=1
    elif(data[i][10] == ''):
        col10+=1
    elif(data[i][11].isdigit() == False):
        col11+=1
    elif(data[i][12] == ''):
        col12+=1

print("The count of the missing values are-")               # print the final counts for the columns
print("Date - ", col0)
print("Averge price - ", col1)
print("Total volume - ", col2)
print("4046 - ", col3)
print("4225 - ", col4)
print("4770 - ", col5)
print("Total bags - ", col6)
print("Small bags - ", col7)
print("Large bags - ", col8)
print("XLarge bags - ", col9)
print("Type - ", col10)
print("Year - ", col11)
print("Region - ", col12)
```

Line 191, Column 9

d)

```
## d) Populate data for the missing values of the attribute= “Average Price” by averaging
## all the values of the “Avg Price” attribute that fall under the same “REGION” attribute value.
can = 0
for i in range(1, len(data)):                                   # for every row in data
    d = data[i][1]
    if(data[i][1] == '' or d.replace('.', '', 1).isdigit() == False):   # if it is missing value
        s = 0                                                   # to keep track of the sum
        count = 0                                               # to keep track of the count for finding the avg value
        for j in range(1, len(data)):
            d = data[j][1]                                      # used for checking if it is a float value or no
            if(data[j][12] == data[i][12] and d.replace('.', '', 1).isdigit() == True):     # if it is a float value
                s = s + float(data[j][1])                       # add it to the sum variable
                count+=1                                        # increment the count
        data[i][1] = str(round(s/count, 6))                     # find avg and replace the missing value
# print(data)                                                       # print the modified data
```

e)

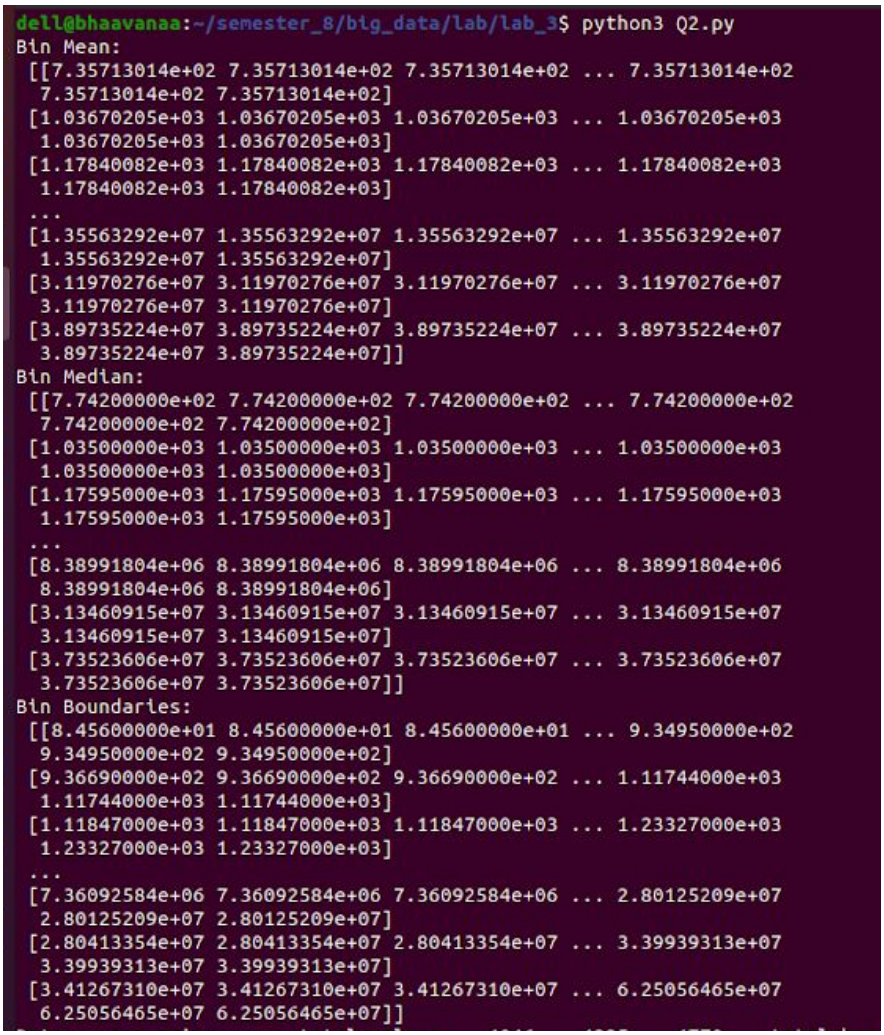
```
## e) Discretize the attribute= “Date” using concept hierarchy into {Old, New, Recent}
## (Consider 2015,2016 : Old, 2017: New, 2018: Recent).

for i in range(1, len(data)):                                 # for every row of data
    if((data[i][11] == '2015') or (data[i][11] == '2016')): # 2015 or 2016 - Old
        data[i][0] = "Old"
    elif(data[i][11] == '2017'):                              # 2017 - New
        data[i][0] = "New"
    elif(data[i][11] == '2018'):                              # 2018 - Recent
        data[i][0] = "Recent"
# print(data)                                                     # print the updated data

```

Detailed output :

a)



```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_3$ python3 Q2.py
Bin Mean:
 [[7.35713014e+02 7.35713014e+02 7.35713014e+02 ... 7.35713014e+02
  7.35713014e+02 7.35713014e+02]
 [1.03670205e+03 1.03670205e+03 1.03670205e+03 ... 1.03670205e+03
  1.03670205e+03 1.03670205e+03]
 [1.17840082e+03 1.17840082e+03 1.17840082e+03 ... 1.17840082e+03
  1.17840082e+03 1.17840082e+03]
 ...
 [1.35563292e+07 1.35563292e+07 1.35563292e+07 ... 1.35563292e+07
  1.35563292e+07 1.35563292e+07]
 [3.11970276e+07 3.11970276e+07 3.11970276e+07 ... 3.11970276e+07
  3.11970276e+07 3.11970276e+07]
 [3.89735224e+07 3.89735224e+07 3.89735224e+07 ... 3.89735224e+07
  3.89735224e+07 3.89735224e+07]]
Bin Median:
 [[7.74200000e+02 7.74200000e+02 7.74200000e+02 ... 7.74200000e+02
  7.74200000e+02 7.74200000e+02]
 [1.03500000e+03 1.03500000e+03 1.03500000e+03 ... 1.03500000e+03
  1.03500000e+03 1.03500000e+03]
 [1.17595000e+03 1.17595000e+03 1.17595000e+03 ... 1.17595000e+03
  1.17595000e+03 1.17595000e+03]
 ...
 [8.38991804e+06 8.38991804e+06 8.38991804e+06 ... 8.38991804e+06
  8.38991804e+06 8.38991804e+06]
 [3.13460915e+07 3.13460915e+07 3.13460915e+07 ... 3.13460915e+07
  3.13460915e+07 3.13460915e+07]
 [3.73523606e+07 3.73523606e+07 3.73523606e+07 ... 3.73523606e+07
  3.73523606e+07 3.73523606e+07]]
Bin Boundaries:
 [[8.45600000e+01 8.45600000e+01 8.45600000e+01 ... 9.34950000e+02
  9.34950000e+02 9.34950000e+02]
 [9.36690000e+02 9.36690000e+02 9.36690000e+02 ... 1.11744000e+03
  1.11744000e+03 1.11744000e+03]
 [1.11847000e+03 1.11847000e+03 1.11847000e+03 ... 1.23327000e+03
  1.23327000e+03 1.23327000e+03]
 ...
 [7.36092584e+06 7.36092584e+06 7.36092584e+06 ... 2.80125209e+07
  2.80125209e+07 2.80125209e+07]
 [2.80413354e+07 2.80413354e+07 2.80413354e+07 ... 3.39939313e+07
  3.39939313e+07 3.39939313e+07]
 [3.41267310e+07 3.41267310e+07 3.41267310e+07 ... 6.25056465e+07
  6.25056465e+07 6.25056465e+07]]
```

b)

Date	avg_price	total_volume	4046	4225	4770	total_bags	small_bags	large_bags	xlarge_bags	list_region	
12-2015	1.17	316325.97	3637.72	280219.74	309.57	32158.94	31731.3	427.64	0.0	Albany	
11-2015	1.29	399712.89	5220.57	354710.03	377.99	39404.3	38110.92	1293.38	0.0	Albany	
10-2015	1.31	284678.47	5617.61	242911.88	436.75	35712.23	34185.58	1526.65	0.0	Albany	
09-2015	1.18	351846.63	4098.09	314481.81	688.13	32578.6	30806.03	1772.57	0.0	Albany	
08-2015	1.26	452003.74	3056.39	393958.99	2285.14	52703.22	52147.26	555.96	0.0	Albany	
07-2015	1.19	436681.53	3232.55	341292.52	13345.43	78811.03	78118.29	659.41	33.33	Albany	
06-2015	1.26	406758.56	2975.53	288856.1	14611.4	100315.53	98631.89	1683.64	0.0	Albany	
05-2015	1.26	486077.58	5093.49	371292.23	606.47	109085.39	107007.23	2078.16	0.0	Albany	
04-2015	1.18	194376.61	3475.84	135735.07	252.03	54913.67	51987.45	2822.05	104.17		Albany
03-2015	1.06	253294.82	7097.51	199387.93	671.51	46137.87	44264.29	1873.58	0.0	Albany	
02-2015	1.03	209370.24	4786.31	164230.65	715.99	39637.29	38263.95	1373.34	0.0	Albany	
01-2015	1.17	171727.14	5677.87	124664.24	476.93	40908.1	38977.41	1930.69	0.0	Albany	
12-2015	1.03	1492887.31	1151827.8	102813.3	858.68	237387.53	171472.45	65898.84	16.24	Atlanta	
11-2015	1.05	1782686.38	1324902.34	184527.78	2005.99	271250.27	178804.14	92388.9	57.23	Atlanta	
10-2015	1.02	1560107.28	987254.98	308167.41	1839.72	262845.17	124746.83	138098.34	0.0	Atlanta	
09-2015	1.01	1776122.48	1189387.45	280795.17	3656.39	302283.47	150356.17	151796.33	130.97		Atlanta
08-2015	1.1	2136930.98	1604475.77	222002.1	11176.36	299276.75	225687.35	73433.91	155.49		Atlanta
07-2015	1.05	1964257.95	1527930.43	184554.11	4406.04	247367.37	186922.42	60420.2	24.75	Atlanta	
06-2015	1.05	2006788.3	1622601.32	108045.6	2705.36	273436.02	207151.49	66027.11	257.42		Atlanta
05-2015	1.05	2643449.02	2259285.26	112875.34	4771.35	266517.07	174711.62	91753.82	51.63	Atlanta	
04-2015	1.07	1713804.48	1452438.41	78081.52	2508.67	180775.88	115422.56	65218.6	134.72		Atlanta
03-2015	1.06	2178237.74	1837895.22	93552.21	1461.35	245328.96	170748.97	74526.28	53.71	Atlanta	
02-2015	1.04	1929355.12	1664218.0	84491.05	1695.44	178950.63	88749.54	90201.09	0.0	Atlanta	
01-2015	1.08	1713388.05	1460358.72	80199.55	2323.27	170506.51	67975.86	102530.65	0.0	Atlanta	
12-2015	1.12	2508152.38	180721.41	1682132.93	86517.62	558780.42	550478.96	8301.46	0.0	BaltimoreWashington	
11-2015	1.1	3409338.23	239445.04	2319095.8	191132.11	659665.28	639415.26	20250.02	0.0	BaltimoreWashington	
10-2015	0	2973499.99	182937.16	1942915.16	199004.9	648642.77	627572.19	21070.58	0.0	BaltimoreWashington	
09-2015	1.18	3154167.4	193531.9	2099436.81	214805.55	646393.14	627065.19	19322.41	5.54	BaltimoreWashington	
08-2015	1.17	4023538.05	304509.68	2647023.62	220756.58	851248.17	835405.06	15375.48	467.63		BaltimoreWashington
07-2015	1.16	3320350.81	231034.01	2107893.23	202054.17	779369.4	760946.71	18038.68	384.01		BaltimoreWashington
06-2015	1.16	3474459.48	250041.31	2118852.39	195128.4	910437.38	894430.55	14805.59	1201.24		BaltimoreWashington
05-2015	1.21	4400870.62	347100.55	2677534.69	262410.26	1113825.12	1080239.65	33145.83	439.64		BaltimoreWashington
04-2015	1.25	3036398.78	249876.52	1783605.74	166205.69	836710.83	812136.01	23598.43	976.39		BaltimoreWashington
03-2015	1.2	3594133.8	290981.9	2185402.72	215246.85	902502.33	880393.62	22108.71	0.0	BaltimoreWashington	
02-2015	1.13	3239515.1	265122.87	2028515.72	229601.46	716275.05	697530.1	18744.95	0.0	BaltimoreWashington	
01-2015	1.17	2808932.8	205091.25	1753502.19	162587.02	687752.34	667629.63	20122.71	0.0	BaltimoreWashington	
12-2015	0.93	274170.13	118850.06	21030.26	23832.94	110456.87	109997.94	21.56	437.37	Boise	
11-2015	1.08	280026.23	136368.82	27142.76	31910.66	84603.99	83701.2	132.76	770.03		Boise
10-2015	1.08	254466.11	120404.43	63396.93	28293.95	42370.8	41584.6	690.11	96.09	Boise	
09-2015	0.99	266814.15	171707.73	37142.24	24888.58	33075.6	32614.74	433.41	27.45	Boise	
08-2015	1.1	323926.09	227066.08	36104.35	28179.46	32576.2	32507.58	62.22	6.4	Boise	
07-2015	1.09	291506.28	210678.46	23981.35	26954.12	29892.35	27604.59	2287.76	0.0	Boise	

c)

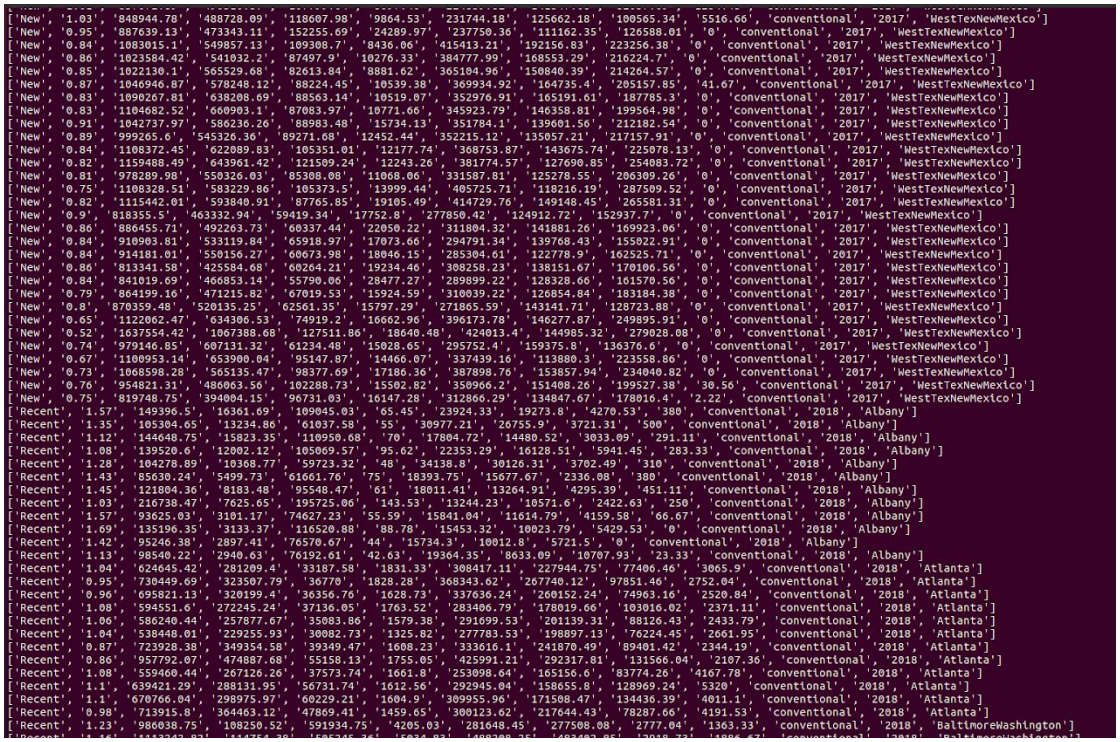
```
The count of the missing values are-
Date -  0
Averge price -  48
Total volume -  0
4046 -  0
4225 -  0
4770 -  0
Total bags -  0
Small bags -  0
Large bags -  0
XLarge bags -  0
Type -  0
Year -  0
Region -  0
```

d)

```
['27-12-2015', '1.33', '64236.62', '1036.74', '54454.85', '48.16', '8696.87', '8603.62', '93.25', '0', 'conventional', '2015', 'Albany']
['20-12-2015', '1.35', '54876.98', '674.28', '44638.81', '58.33', '9505.56', '9408.07', '97.49', '0', 'conventional', '2015', 'Albany']
['13-12-2015', '0.93', '118220.22', '794.7', '109149.67', '130.5', '8145.35', '8042.21', '103.14', '0', 'conventional', '2015', 'Albany']
['06-12-2015', '1.08', '78992.15', '1132', '71976.41', '72.58', '5811.16', '5677.4', '133.76', '0', 'conventional', '2015', 'Albany']
['29-11-2015', '1.29', '51039.6', '941.48', '43838.39', '75.78', '6183.95', '5986.26', '197.69', '0', 'conventional', '2015', 'Albany']
['22-11-2015', '1.570755', '55979.78', '1184.27', '48067.99', '43.61', '6683.91', '6556.47', '127.44', '0', 'conventional', '2015', 'Albany']
['15-11-2015', '1.570755', '83453.76', '1368.92', '73672.72', '93.26', '8318.86', '8196.81', '122.05', '0', 'conventional', '2015', 'Albany']
['08-11-2015', '1.570755', '109428.33', '703.75', '101815.36', '80', '6829.22', '6266.85', '562.37', '0', 'conventional', '2015', 'Albany']
['01-11-2015', '1.570755', '99811.42', '1022.15', '87315.57', '85.34', '11388.36', '11104.53', '283.83', '0', 'conventional', '2015', 'Albany']
['25-10-2015', '1.570755', '74338.76', '842.4', '64757.44', '113', '8625.92', '8061.47', '564.45', '0', 'conventional', '2015', 'Albany']
['18-10-2015', '1.570755', '84843.44', '924.86', '75595.85', '117.07', '8205.66', '7877.86', '327.8', '0', 'conventional', '2015', 'Albany']
['11-10-2015', '1.570755', '64489.17', '1582.03', '52677.92', '105.32', '10123.9', '9866.27', '257.63', '0', 'conventional', '2015', 'Albany']
['04-10-2015', '1.31', '61007.1', '2268.32', '49880.67', '101.36', '8756.75', '8379.98', '376.77', '0', 'conventional', '2015', 'Albany']
['27-09-2015', '0.99', '106803.39', '1204.88', '99409.21', '154.84', '6034.46', '5888.87', '145.59', '0', 'conventional', '2015', 'Albany']
['20-09-2015', '1.33', '69759.01', '1028.03', '59313.12', '150.5', '9267.36', '8489.1', '778.26', '0', 'conventional', '2015', 'Albany']
['13-09-2015', '1.28', '76111.27', '985.73', '65696.86', '142', '9286.68', '8665.19', '621.49', '0', 'conventional', '2015', 'Albany']
['06-09-2015', '1.11', '99172.96', '879.45', '90062.62', '240.79', '7990.1', '7762.87', '227.23', '0', 'conventional', '2015', 'Albany']
['30-08-2015', '1.07', '105693.84', '689.01', '94362.67', '335.43', '10306.73', '10218.93', '87.8', '0', 'conventional', '2015', 'Albany']
['23-08-2015', '1.34', '79992.09', '733.16', '67933.79', '444.78', '10880.36', '10745.79', '134.57', '0', 'conventional', '2015', 'Albany']
['16-08-2015', '1.33', '80043.78', '539.65', '68666.01', '394.9', '10443.22', '10297.68', '145.54', '0', 'conventional', '2015', 'Albany']
['09-08-2015', '1.12', '111140.93', '584.63', '100961.46', '368.95', '9225.89', '9116.34', '109.55', '0', 'conventional', '2015', 'Albany']
['02-08-2015', '1.45', '75133.1', '509.94', '62035.06', '741.08', '11847.02', '11768.52', '78.5', '0', 'conventional', '2015', 'Albany']
['26-07-2015', '1.11', '106757.1', '648.75', '91949.05', '966.61', '13192.69', '13061.53', '131.16', '0', 'conventional', '2015', 'Albany']
['19-07-2015', '1.26', '96617', '1042.1', '82049.4', '2238.02', '11287.48', '11103.49', '183.99', '0', 'conventional', '2015', 'Albany']
['12-07-2015', '1.05', '124055.31', '672.25', '94693.52', '4257.64', '24431.9', '24290.08', '108.49', '33.33', 'conventional', '2015', 'Albany']
['05-07-2015', '1.35', '109252.12', '869.45', '72600.55', '5883.16', '29898.96', '29663.19', '235.77', '0', 'conventional', '2015', 'Albany']
['28-06-2015', '1.37', '89534.81', '664.23', '57545.79', '4662.71', '26662.08', '26311.76', '350.32', '0', 'conventional', '2015', 'Albany']
['21-06-2015', '1.27', '104849.39', '804.01', '76688.55', '5481.18', '21875.65', '21662', '213.65', '0', 'conventional', '2015', 'Albany']
['14-06-2015', '1.32', '89631.3', '850.58', '55400.94', '4377.19', '29002.59', '28343.14', '659.45', '0', 'conventional', '2015', 'Albany']
```

e)

```
['Old', '1.25', '50809.3', '6329.81', '28751.94', '572.18', '15155.37', '5948.7', '9206.67', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.1', '41244.99', '4759.58', '9207.54', '14337.18', '12940.69', '7111.59', '5829.1', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.09', '52358.11', '5422.01', '17237.12', '21948.79', '7750.19', '5167.45', '2582.74', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.36', '38767.82', '2907.01', '25699', '262.86', '9898.95', '5774.63', '4124.32', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.36', '37885.9', '2655.44', '25121.02', '177.35', '9932.09', '5036.29', '4895.8', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.31', '21561.29', '3204.95', '7746.53', '111.55', '10498.26', '8558.96', '1939.3', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.29', '20698.7', '2278.61', '6800.76', '58.05', '11561.28', '10538.62', '1022.66', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.41', '17528.02', '2983.3', '6722.24', '53.08', '7769.4', '7268.06', '501.34', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.47', '15705.35', '2731.96', '6730.01', '52.95', '6190.43', '6185.08', '5.35', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.3', '18611.51', '1716.44', '8405.31', '60.03', '8429.73', '8365.7', '64.03', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.32', '18916.59', '2653.06', '5464.63', '14.37', '10784.53', '10704.67', '79.86', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.36', '16034.56', '2064.7', '5044.66', '7.17', '8918.03', '8879.53', '38.5', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.48', '14413.74', '1308.04', '4865.24', '4.77', '8235.69', '7090.72', '1144.97', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.39', '15805.58', '1483.39', '4975.83', '38.02', '9308.34', '6330.13', '2978.21', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.49', '13712.44', '1451', '4199.43', '37.92', '8024.09', '7913.5', '110.59', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.64', '11626.56', '1618.9', '4032.03', '2.34', '5973.29', '5957.67', '15.62', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.47', '12171.04', '1554.09', '4656.41', '9.36', '5951.18', '5852.36', '98.82', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.45', '13237.48', '1912.6', '4779.17', '25.72', '6519.99', '6447.25', '72.74', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.37', '12647.9', '1591.6', '4070.29', '23.41', '6962.6', '6780.56', '182.04', '0', 'organic', '2016', 'WestTexNewMexico']
['Old', '1.58', '9667.05', '1820.39', '2783.11', '16.41', '5047.14', '4651.16', '395.98', '0', 'organic', '2016', 'WestTexNewMexico']
['New', '1.46', '3463.85', '18.12', '198.16', '0', '3247.57', '3247.57', '0', '0', 'organic', '2017', 'Albany']
['New', '1.58', '3694.13', '31.68', '327.39', '0', '3335.06', '3335.06', '0', '0', 'organic', '2017', 'Albany']
['New', '1.43', '3513.77', '37.46', '209.3', '0', '3267.01', '3267.01', '0', '0', 'organic', '2017', 'Albany']
['New', '1.45', '3779.98', '18.04', '262.14', '0', '3499.8', '3499.8', '0', '0', 'organic', '2017', 'Albany']
['New', '1.44', '3577.04', '118.55', '306.55', '0', '3151.94', '3151.94', '0', '0', 'organic', '2017', 'Albany']
['New', '1.57', '2841.29', '27.75', '182.15', '0', '2631.39', '2631.39', '0', '0', 'organic', '2017', 'Albany']
['New', '1.75', '2506.38', '0', '252.98', '0', '2253.4', '2250.07', '3.33', '0', 'organic', '2017', 'Albany']
['New', '1.71', '2664.62', '0', '245.71', '0', '2418.91', '2418.91', '0', '0', 'organic', '2017', 'Albany']
['New', '1.5', '3425.86', '3.56', '64.14', '0', '3358.16', '3358.16', '0', '0', 'organic', '2017', 'Albany']
['New', '1.38', '5583.79', '12.99', '180.62', '0', '5390.18', '5390.18', '0', '0', 'organic', '2017', 'Albany']
['New', '1.49', '4337.67', '3.52', '268.83', '0', '4065.32', '4061.99', '3.33', '0', 'organic', '2017', 'Albany']
['New', '1.47', '4522.84', '18.71', '237.43', '0', '4266.7', '4266.7', '0', '0', 'organic', '2017', 'Albany']
['New', '1.4', '5229.43', '23.15', '341.52', '0', '4864.76', '4864.76', '0', '0', 'organic', '2017', 'Albany']
['New', '1.59', '3423.95', '31.2', '150.24', '0', '3242.51', '3242.51', '0', '0', 'organic', '2017', 'Albany']
['New', '1.42', '3627.18', '56.82', '95.31', '0', '3475.05', '3475.05', '0', '0', 'organic', '2017', 'Albany']
['New', '1.55', '3431.2', '30.32', '215.1', '0', '3185.78', '3185.78', '0', '0', 'organic', '2017', 'Albany']
['New', '1.78', '3139.84', '45.5', '196.27', '0', '2898.07', '2894.74', '3.33', '0', 'organic', '2017', 'Albany']
['New', '2', '2022', '112.42', '254.49', '0', '1655.09', '1655.09', '0', '0', 'organic', '2017', 'Albany']
['New', '1.91', '2525.28', '44.15', '210.54', '0', '2270.59', '2270.59', '0', '0', 'organic', '2017', 'Albany']
['New', '1.86', '2584.08', '61.21', '143.82', '0', '2379.05', '2379.05', '0', '0', 'organic', '2017', 'Albany']
['New', '1.9', '2259.92', '42.86', '207.29', '0', '2009.77', '2009.77', '0', '0', 'organic', '2017', 'Albany']
['New', '1.98', '2576.49', '39.31', '331.89', '0', '2205.29', '2205.29', '0', '0', 'organic', '2017', 'Albany']
['New', '1.67', '2503.82', '55.26', '92.99', '0', '2355.57', '2355.57', '0', '0', 'organic', '2017', 'Albany']
['New', '1.42', '4233.61', '93.39', '93.39', '0', '4046.83', '4046.83', '0', '0', 'organic', '2017', 'Albany']
['New', '1.87', '2889.03', '155.1', '274.95', '0', '2458.98', '2458.98', '0', '0', 'organic', '2017', 'Albany']
['New', '2', '1883.1', '38.06', '187.9', '0', '1657.14', '1657.14', '0', '0', 'organic', '2017', 'Albany']
['New', '2.03', '2268.86', '59.41', '278.04', '0', '1931.41', '1931.41', '0', '0', 'organic', '2017', 'Albany']
['New', '2.13', '2898.81', '142.46', '295.6', '0', '2460.75', '2460.75', '0', '0', 'organic', '2017', 'Albany']
['New', '2.03', '3185.1', '366.52', '266.88', '0', '2551.7', '2551.7', '0', '0', 'organic', '2017', 'Albany']
['New', '2.04', '2719.24', '21.33', '248.87', '0', '2449.04', '2449.04', '0', '0', 'organic', '2017', 'Albany']
['New', '1.63', '3771.39', '54.47', '352.85', '0', '3364.07', '3364.07', '0', '0', 'organic', '2017', 'Albany']
['New', '1.87', '3365.45', '11.93', '344.88', '0', '3008.64', '3008.64', '0', '0', 'organic', '2017', 'Albany']
['New', '1.84', '3184.37', '20.33', '385.05', '0', '2778.99', '2778.99', '0', '0', 'organic', '2017', 'Albany']
['New', '1.57', '4270.28', '44.12', '450.74', '0', '3775.42', '3775.42', '0', '0', 'organic', '2017', 'Albany']
['New', '1.79', '2897.23', '89.53', '315.14', '0', '2492.56', '2492.56', '0', '0', 'organic', '2017', 'Albany']
```


['New', '1.03', '848944.78', '488728.09', '118607.98', '9864.53', '231744.18', '125662.18', '100565.34', '5516.66', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.95', '887639.13', '473343.11', '152255.69', '24289.97', '237750.36', '111162.35', '126588.01', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.84', '1083015.1', '549857.13', '109308.7', '8436.06', '415413.21', '192156.83', '223256.38', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.86', '1023584.42', '541032.2', '87497.9', '10276.33', '384777.99', '168553.29', '216224.7', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.85', '1022130.1', '565529.68', '82613.84', '8881.62', '365104.96', '150840.39', '214264.57', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.87', '1046946.87', '578248.12', '88224.45', '10539.38', '369934.92', '164735.4', '205157.85', '41.67', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.83', '1090267.81', '638208.69', '88563.14', '10519.07', '352976.91', '165191.61', '187785.3', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.83', '1104682.52', '660903.1', '87083.97', '10771.66', '345923.79', '146358.81', '199564.98', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.91', '1042737.97', '586236.26', '88983.48', '15734.13', '351784.1', '139601.56', '212182.54', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.89', '999265.6', '545326.36', '89271.68', '12452.44', '352215.12', '135057.21', '217157.91', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.84', '1108372.45', '622089.83', '105351.01', '12177.74', '368753.87', '143675.74', '225078.13', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.82', '1159488.49', '643961.42', '121509.24', '12243.26', '381774.57', '127690.85', '254083.72', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.81', '978289.98', '550326.03', '85308.08', '11068.06', '331587.81', '125278.55', '206309.26', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.75', '1108328.51', '583229.86', '105373.5', '13999.44', '405725.71', '118216.19', '287509.52', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.82', '1115442.01', '593840.91', '87765.85', '19105.49', '414729.76', '149148.45', '265581.31', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.9', '818355.5', '463332.94', '59419.34', '17752.8', '277850.42', '124912.72', '152937.7', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.86', '886455.71', '492263.73', '60337.44', '22050.22', '311804.32', '141881.26', '169923.06', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.84', '910903.81', '533119.84', '65918.97', '17073.66', '294791.34', '139768.43', '155022.91', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.84', '914181.01', '550156.27', '60673.98', '18046.15', '285304.61', '122778.9', '162525.71', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.86', '813341.58', '425584.68', '60264.21', '19234.46', '308258.23', '138151.67', '170106.56', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.84', '841019.69', '466853.14', '55790.06', '28477.27', '289899.22', '128328.66', '161570.56', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.79', '864199.16', '471215.82', '67019.53', '15924.59', '310039.22', '126854.84', '183184.38', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.8', '870359.48', '520135.25', '62561.35', '15797.29', '271865.59', '143141.71', '128723.88', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.65', '1122062.47', '634306.53', '74919.2', '16662.96', '396173.78', '146277.87', '249895.91', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.52', '1637554.42', '1067388.68', '127511.86', '18640.48', '424013.4', '144985.32', '279028.08', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.74', '979146.85', '607131.32', '61234.48', '15028.65', '295752.4', '159375.8', '136376.6', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.67', '1100953.14', '653900.04', '95147.87', '14466.07', '337439.16', '113880.3', '223558.86', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.73', '1068598.28', '565135.47', '98377.69', '17186.36', '387898.76', '153857.94', '234040.82', '0', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.76', '954821.31', '486063.56', '102288.73', '15502.82', '350966.2', '151408.26', '199527.38', '30.56', 'conventional', '2017', 'WestTexNewMexico']
['New', '0.75', '819748.75', '394004.15', '96731.03', '16147.28', '312866.29', '134847.67', '178016.4', '2.22', 'conventional', '2017', 'WestTexNewMexico']
['Recent', '1.57', '149396.5', '16361.69', '109045.03', '65.45', '23924.33', '19273.8', '4270.53', '380', 'conventional', '2018', 'Albany']
['Recent', '1.35', '105304.65', '13234.86', '61037.58', '55', '30977.21', '26755.9', '3721.31', '500', 'conventional', '2018', 'Albany']
['Recent', '1.12', '144648.75', '15823.35', '110950.68', '70', '17804.72', '14480.52', '3033.09', '291.11', 'conventional', '2018', 'Albany']
['Recent', '1.08', '139520.6', '12002.12', '105069.57', '95.62', '22353.29', '16128.51', '5941.45', '283.33', 'conventional', '2018', 'Albany']
['Recent', '1.28', '104278.89', '10368.77', '59723.32', '48', '34138.8', '30126.31', '3702.49', '310', 'conventional', '2018', 'Albany']
['Recent', '1.43', '85630.24', '5499.73', '61661.76', '75', '18393.75', '15677.67', '2336.08', '380', 'conventional', '2018', 'Albany']
['Recent', '1.45', '121804.36', '8183.48', '95548.47', '61', '18011.41', '13264.91', '4295.39', '451.11', 'conventional', '2018', 'Albany']
['Recent', '1.03', '216738.47', '7625.65', '195725.06', '143.53', '13244.23', '10571.6', '2422.63', '250', 'conventional', '2018', 'Albany']
['Recent', '1.57', '93625.03', '3101.17', '74627.23', '55.59', '15841.04', '11614.79', '4159.58', '66.67', 'conventional', '2018', 'Albany']
['Recent', '1.69', '135196.35', '3133.37', '116520.88', '88.78', '15453.32', '10023.79', '5429.53', '0', 'conventional', '2018', 'Albany']
['Recent', '1.42', '95246.38', '2897.41', '76570.67', '44', '15734.3', '10012.8', '5721.5', '0', 'conventional', '2018', 'Albany']
['Recent', '1.13', '98540.22', '7940.63', '76192.61', '42.63', '13364.35', '8633.09', '10707.93', '23.33', 'conventional', '2018', 'Albany']
['Recent', '1.04', '624645.42', '281209.4', '33187.58', '1831.33', '308417.11', '227944.75', '77406.46', '3065.9', 'conventional', '2018', 'Atlanta']
['Recent', '0.95', '730449.69', '323507.79', '36770', '1828.28', '368343.62', '267740.12', '97851.46', '2752.04', 'conventional', '2018', 'Atlanta']
['Recent', '0.96', '695821.13', '320199.4', '36356.76', '1628.73', '337636.24', '260152.24', '74963.16', '2520.84', 'conventional', '2018', 'Atlanta']
['Recent', '1.08', '594551.6', '272245.24', '37136.05', '1763.52', '283406.79', '178019.66', '103016.02', '2371.11', 'conventional', '2018', 'Atlanta']
['Recent', '1.06', '586240.44', '257877.67', '35083.86', '1579.38', '291699.53', '201139.31', '88126.43', '2433.79', 'conventional', '2018', 'Atlanta']
['Recent', '1.04', '538448.01', '229255.93', '30082.73', '1325.82', '277783.53', '198897.13', '76224.45', '2661.95', 'conventional', '2018', 'Atlanta']
['Recent', '0.87', '723928.38', '349354.58', '39349.47', '1608.23', '333616.1', '241870.49', '89401.42', '2344.19', 'conventional', '2018', 'Atlanta']
['Recent', '0.86', '957792.07', '474887.68', '55158.13', '1755.05', '425991.21', '292317.81', '131566.04', '2107.36', 'conventional', '2018', 'Atlanta']
['Recent', '1.08', '559460.44', '267126.26', '37573.74', '1661.8', '253098.64', '165156.6', '83774.26', '4167.78', 'conventional', '2018', 'Atlanta']
['Recent', '1.1', '639421.29', '288131.95', '56731.74', '1612.56', '292945.04', '158655.8', '128969.24', '5320', 'conventional', '2018', 'Atlanta']
['Recent', '1.1', '670766.04', '298975.97', '60229.21', '1604.9', '309955.96', '171508.47', '134436.39', '4011.1', 'conventional', '2018', 'Atlanta']
['Recent', '0.98', '713915.8', '364463.12', '47869.41', '1459.65', '300123.62', '217644.43', '78287.66', '4191.53', 'conventional', '2018', 'Atlanta']
['Recent', '1.23', '986038.75', '108250.52', '591934.75', '4205.03', '281648.45', '277508.08', '2777.04', '1363.33', 'conventional', '2018', 'BaltimoreWashington']