Introduction to Big Data Midterm

Assign date: 2018-2-08

Due date: 2018-2-16 11:59pm

Please submit only one copy of your Jupyter Notebook file with your answers for each question provided in the following cells below to Moodle. There will be no extensions or excemptions.

You will lose 10% each day your submission is late.

Please do not post any answers to the general question forum. You can however ask for technical help on the course page if you are having actualy technical issues. I reserve the right to delete any question/response that I deem being too much "begging for hints" in nature. You may email me directly at mtenney@yorku.ca if you have any questions, but do note I will not provide you any answers and my response may take up to 48hrs. And of course, you're free to use any class material, the Internet, or additional resources to complete the exam. The exam covers material from weeks 1-4 inclusively.

Note you MUST RUN THE CODE and print out the required results when submitting. Any empty cells or un-run code will not be accepted.

1) Import the following libraries to your notebook (you can use alias' if you want) in the cell below:
pandas, seaborn, os, matplotlib.pyplot
And add the following flag: %matplotlib inline

In [171]:
```
import matplotlib.pyplot as plt
```

In [1]:
```
import pandas as pd
import seaborn as sns
import os
%matplotlib inline
```

2) Index a list in Python

Run the code below, it will give a result of 4. If we want to get ONLY the number 8 from the list of lists below, how would you alter the code? Alter the code directly and run to print the desired output.

In [6]:
```
question2 = [[1,2],[3,4],[5,6],[7,[8,9]]]
print question2[3][1][0]
```

```
8
```

3) Define a variable called bigdata, to be a list in python with the following elements and print only the last two elements of the list:

1, "Big Data Class", 2, "Is Really Fun",3, "I Love Python"

In [7]:
```
bigdata=[1,"Big Data Class",2,"Is Really Fun",3,"I Love Python"]
print bigdata[-2],bigdata[-1]
```

```
3 I Love Python
```

4) Cast the following list to be a string in Python:

Define the list [1,2,4,3,'k'] and cast it to a string where you print out '1243k'. Insert your code below.

In [325]:
```
list1 = [1,2,4,3,'k']
''.join(str(x) for x in list1)

#list1 = [1,2,4,3,'k']
#x=''
#for i in range(len(list1)):
#    x=x+str(list1[i])
#print(x)
```

Out[325]:
```
'1243k'
```

5) Test whether the two lists are equal too each other and if not what elements are shared between the two lists.

list1 = ['1',2,'3',4,5]

list2 = [1,2,3,4,'5',6]

Define the lists and print whether the lists are equal to one another, and find which elements are a match. (***Hint you can go element by element or use you can use a "for" loop, or, you can use a different data structure all together***).

In [35]:
```
list1 = ['1',2,'3',4,5]
list2 = [1,2,3,4,'5',6]

if set(list1) == set(list2):
    print "List are equal"
else :
    print "Lists are not equal and common elements are " + str(list(set(list1).intersection(set(list2))))
```

```
Lists are not equal and common elements are [2, 4]
```

6) Convert list2 to be indentical to list1 by casting the datatypes to match those in list1.

In [332]:
```
list1 = ['1',2,'3',4,5]
list2 = [1,2,3,4,'5',6]

list1 = list1 + [6]

list2[0] = str(list2[0])
list2[2] = str(list2[2])
list2[4] = int(list2[4])

print list1
print list2
```

```
['1', 2, '3', 4, 5, 6]
['1', 2, '3', 4, 5, 6]
```

7) What are the "5 V's" commonly used to describe Big Data? Write your answer below.

In [ ]:
```
Velocity, Volume, Value, Variety, and Veracity
```

8) Finish the quote:
"All models are wrong. Some are [...]"

In [ ]:
```
useful.
```

9) If a variable dogs = 103x23/2 and a variable cats = 10x12x4, Use python to determing if dogs are greater than cats.

In [37]:
```
dogs = 103*23/2
cats = 10*12*4

if dogs > cats:
    print "dogs are greater"
else:
    print "cats are greater"
```

```
dogs are greater
```

10) What is the name for the 3 primary measures of central tendency and give their definition

the mode, the median and the mean

The mode is the most commonly occurring value in a distribution. The median is the middle value in distribution when the values are arranged in ascending or descending order. The mean is the sum of the value of each observation in a dataset divided by the number of observations. This is also known as the arithmetic average.

11) Calculate the descriptive statistice for each of the above central measures in Python on this list of numbers and print out the values:

[12,44,23,2,1,66,7,85,3,34,23,6,34,23,7,21,14,34,23,88]
(Hint: it might be easier to import a function from some other library)

In [333]:
```
list3 = [12,44,23,2,1,66,7,85,3,34,23,6,34,23,7,21,14,34,23,88]
n = len(list3)
sum_of_elements = sum(list3)
mean = sum_of_elements/float(n)
print (mean)

sorted_elements = sorted(list3)

middle_elements = [sorted_elements[int(n)/2], sorted_elements[int(n)/2 - 1]]

median_of_elements = sum(middle_elements) / 2.0

print float(median_of_elements)

highest_freq, highest_freq_elem = 0, 0
for elem in sorted_elements:
    current_count = sorted_elements.count(elem)
    if (current_count > highest_freq):
        highest_freq = current_count
        highest_freq_elem = elem

mode = highest_freq_elem
print mode

#import statistics
#list1 = [12,44,23,2,1,66,7,85,3,34,23,6,34,23,7,21,14,34,23,88]
#x = statistics.mean(list1)
#print(x)
#y = statistics.median(list1)
#print(y)
#z = statistics.mode(list1)
#print(z)
```

```
27.5
23.0
23
```

12) Delete the empty cell between question 12 (this one) and Question 13 from the Jupyter notebook

13) Create a list, called mylist, containing a sequential list of numbers that starts with 2 and stops at 1000, counting by 2's in Python

In [312]:
```
mylist=[]
for i in range(2,1000,2):
    mylist.append(i)
print (mylist)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56,
58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108,
110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152,
154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196,
198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240,
242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284,
286, 288, 290, 292, 294, 296, 298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328,
330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372,
374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400, 402, 404, 406, 408, 410, 412, 414, 416,
418, 420, 422, 424, 426, 428, 430, 432, 434, 436, 438, 440, 442, 444, 446, 448, 450, 452, 454, 456, 458, 460,
462, 464, 466, 468, 470, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490, 492, 494, 496, 498, 500, 502, 504,
506, 508, 510, 512, 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536, 538, 540, 542, 544, 546, 548,
550, 552, 554, 556, 558, 560, 562, 564, 566, 568, 570, 572, 574, 576, 578, 580, 582, 584, 586, 588, 590, 592,
594, 596, 598, 600, 602, 604, 606, 608, 610, 612, 614, 616, 618, 620, 622, 624, 626, 628, 630, 632, 634, 636,
638, 640, 642, 644, 646, 648, 650, 652, 654, 656, 658, 660, 662, 664, 666, 668, 670, 672, 674, 676, 678, 680,
682, 684, 686, 688, 690, 692, 694, 696, 698, 700, 702, 704, 706, 708, 710, 712, 714, 716, 718, 720, 722, 724,
726, 728, 730, 732, 734, 736, 738, 740, 742, 744, 746, 748, 750, 752, 754, 756, 758, 760, 762, 764, 766, 768,
770, 772, 774, 776, 778, 780, 782, 784, 786, 788, 790, 792, 794, 796, 798, 800, 802, 804, 806, 808, 810, 812,
814, 816, 818, 820, 822, 824, 826, 828, 830, 832, 834, 836, 838, 840, 842, 844, 846, 848, 850, 852, 854, 856,
858, 860, 862, 864, 866, 868, 870, 872, 874, 876, 878, 880, 882, 884, 886, 888, 890, 892, 894, 896, 898, 900,
902, 904, 906, 908, 910, 912, 914, 916, 918, 920, 922, 924, 926, 928, 930, 932, 934, 936, 938, 940, 942, 944,
946, 948, 950, 952, 954, 956, 958, 960, 962, 964, 966, 968, 970, 972, 974, 976, 978, 980, 982, 984, 986, 988,
990, 992, 994, 996, 998]
```

14) Is the sum of the first 50 elements in "mylist" greater than the last element of "mylist"?

Use Python below to print out the answer. (mylist is defined in question 13).

In [314]:
```
a = 0
for x in range(50):
    a = a + mylist[x]

if a > mylist[-1]:
    print ("First 50 elements greater than " + str(mylist[-1]))
else:
    print ("Last element is greater")
```

```
First 50 elements greater than 998
```

15) What is the length of "mylist"?

In [17]:
```
len(mylist)
```

Out[17]: 499

16) Create another list, called my2ndlist, containing a list of numbers starting at 1002 and stops at 1052,counting by 3's in Python. THEN concatenate my2ndlist with mylist into a new list called theMegaList.

Print theMegaList below.

In [21]:
```
my2ndlist=[]
for x in range(1002,1052,3):
    my2ndlist.append(x)
print (my2ndlist)

theMegalist=mylist + my2ndlist
print (theMegalist)
```

```
[1002, 1005, 1008, 1011, 1014, 1017, 1020, 1023, 1026, 1029, 1032, 1035, 1038, 1041, 1044, 1047, 1050]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56,
58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108,
110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152,
154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196,
198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240,
242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284,
286, 288, 290, 292, 294, 296, 298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328,
330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372,
374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400, 402, 404, 406, 408, 410, 412, 414, 416,
418, 420, 422, 424, 426, 428, 430, 432, 434, 436, 438, 440, 442, 444, 446, 448, 450, 452, 454, 456, 458, 460,
462, 464, 466, 468, 470, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490, 492, 494, 496, 498, 500, 502, 504,
506, 508, 510, 512, 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536, 538, 540, 542, 544, 546, 548,
550, 552, 554, 556, 558, 560, 562, 564, 566, 568, 570, 572, 574, 576, 578, 580, 582, 584, 586, 588, 590, 592,
594, 596, 598, 600, 602, 604, 606, 608, 610, 612, 614, 616, 618, 620, 622, 624, 626, 628, 630, 632, 634, 636,
638, 640, 642, 644, 646, 648, 650, 652, 654, 656, 658, 660, 662, 664, 666, 668, 670, 672, 674, 676, 678, 680,
682, 684, 686, 688, 690, 692, 694, 696, 698, 700, 702, 704, 706, 708, 710, 712, 714, 716, 718, 720, 722, 724,
726, 728, 730, 732, 734, 736, 738, 740, 742, 744, 746, 748, 750, 752, 754, 756, 758, 760, 762, 764, 766, 768,
770, 772, 774, 776, 778, 780, 782, 784, 786, 788, 790, 792, 794, 796, 798, 800, 802, 804, 806, 808, 810, 812,
814, 816, 818, 820, 822, 824, 826, 828, 830, 832, 834, 836, 838, 840, 842, 844, 846, 848, 850, 852, 854, 856,
858, 860, 862, 864, 866, 868, 870, 872, 874, 876, 878, 880, 882, 884, 886, 888, 890, 892, 894, 896, 898, 900,
902, 904, 906, 908, 910, 912, 914, 916, 918, 920, 922, 924, 926, 928, 930, 932, 934, 936, 938, 940, 942, 944,
946, 948, 950, 952, 954, 956, 958, 960, 962, 964, 966, 968, 970, 972, 974, 976, 978, 980, 982, 984, 986, 988,
990, 992, 994, 996, 998, 1002, 1005, 1008, 1011, 1014, 1017, 1020, 1023, 1026, 1029, 1032, 1035, 1038, 1041,
1044, 1047, 1050]
```

17) What is the index for the value 822 in theMegaList?

In [23]:
```
theMegalist.index(822)
```

Out[23]: 410

18) Write a for loop that will test if an element in theMegaList is an even or odd number and keeps a count of how many of each there are in the list.

Print out the total number of even and odd numbers found in theMegalist below.

In [26]:
```
odd=[]
even=[]

for x in theMegalist:
    if x%2==0:
        even.append(x)
    else:
        odd.append(x)
count_odd = len(odd)
count_even = len(even)

print ("There are " + str(count_odd) + " odd and " + str(count_even) + " even numbers")
```

```
There are 8 odd and 508 even numbers
```

19) Write a brief definition of what Data Latency and the difference between "real-time" and "near-time" data.

Data Latency refers to how much time delay exists between an actual event happening to the data and database being made aware of it. It can also refer to how frequently data is being updated.

Real Time: Low latency describes the condition when there is a “small” amount of time between a real event and the data is updated to reflect it. Depending on the application, low latency may be sub second in an automation environment and several minutes in a business environment. Near Time: Refers to the latency that is less than a day but greater than an hour. High latency describes the condition when the delay is larger and typically measured in hours or days.

19) What does the statistic Standard Deviation tell you about a numerical dataset?

In [ ]:
```
The standard deviation of a dataset gives you a measure of how spread out it is.
On an average, it helps you ascertain how close each point is from the mean.
```

20) Suppose you have the following loop:

```
things = [1,2,3,5,6,7,8,9,10]
previous_thing = 0
for thing in things:
    new_thing = thing * previous_thing
    x = 3+previous_thing
    previous_thing = thing
```

At the end of the loop:

What does new_thing equal?
What does x equal?

In [ ]:
```
new_thing = 90
x = 12
```

21) What's the difference between Zero and None or Null?

Zero is a valid value. It is of integer datatype. NULL is undefined, unknown or no value. E.g. If someone have 0 money, it means that person have no money. However, if he have NULL amount of money then we are not sure how many money he has.

22) In the following loop:

```
x = 1
sure_thing = True
while sure_thing:
    x = x +100
    if x < 20001:
        x = x-1
        else:
        sure_thing = False
```

What does x equal to make our sure_thing False?

In [ ]:
```
20099
```

Based on the DataFrame df and df1 defined below, add needed code to answer the following questions.

In [335]:
```
#from pandas import Series, DataFrame
import pandas as pd
raw_data = {'first_name': ['Jason', 'Jason', 'Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', 'Miller', 'Ali', 'Milner', 'Milner'],
        'age': [42, 42, 36, 24, 73],
        'preTest': [4, 4, 31, 2, 3],
        'postTest': [25, 25, 57, 62, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'preTest', 'postTest'])
df1 = df
```

23) What are the column names in df1?print the result.

In [338]:
```
print (df1.columns.values)
#print (list(df1))
```

```
['first_name' 'last_name' 'age' 'preTest' 'postTest']
```

24) What is the average (mean) of the preTest and postTest scores?print the result.

In [342]:
```
print (df1['preTest'].mean())
print (df1['postTest'].mean())

#count = len(raw_data['preTest'])
#Mean = float(sum(raw_data['preTest']))/count
#print (Mean)
#count = len(raw_data['postTest'])
#Mean = float(sum(raw_data['postTest']))/count
#print (Mean)
```

```
8.8
47.8
```

25) Remove duplicated rows in df based on last_name only. Instead of keeping the first value, now keep the last of the duplicated values.print the result.

In [19]:
```
df.drop_duplicates('last_name',keep='last')
```

Out[19]:

	first_name	last_name	age	preTest	postTest
1	Jason	Miller	42	4	25
2	Tina	Ali	36	31	57
4	Amy	Milner	73	3	70

26) Remove duplicated rows in df1 based on both first_name and last_name. print the result.

In [20]:
```
df1.drop_duplicates(subset=['first_name','last_name'])
```

Out[20]:

	first_name	last_name	age	preTest	postTest
0	Jason	Miller	42	4	25
2	Tina	Ali	36	31	57
3	Jake	Milner	24	2	62
4	Amy	Milner	73	3	70

Use the dataframe defined in the following cell to answer the next few questions:

In [23]:
```
from pandas import Series, DataFrame
import pandas as pd
from numpy.random import randn
import numpy as np
np.random.seed(42)
randomdata = DataFrame(np.random.randn(400, 4))
print randomdata
%matplotlib inline
```

	0	1	2	3
0	0.496714	-0.138264	0.647689	1.523030
1	-0.234153	-0.234137	1.579213	0.767435
2	-0.469474	0.542560	-0.463418	-0.465730
3	0.241962	-1.913280	-1.724918	-0.562288
4	-1.012831	0.314247	-0.908024	-1.412304
5	1.465649	-0.225776	0.067528	-1.424748
6	-0.544383	0.110923	-1.150994	0.375698
7	-0.600639	-0.291694	-0.601707	1.852278
8	-0.013497	-1.057711	0.822545	-1.220844
9	0.208864	-1.959670	-1.328186	0.196861
10	0.738467	0.171368	-0.115648	-0.301104
11	-1.478522	-0.719844	-0.460639	1.057122
12	0.343618	-1.763040	0.324084	-0.385082
13	-0.676922	0.611676	1.031000	0.931280
14	-0.839218	-0.309212	0.331263	0.975545
15	-0.479174	-0.185659	-1.106335	-1.196207
16	0.812526	1.356240	-0.072010	1.003533
17	0.361636	-0.645120	0.361396	1.538037
18	-0.035826	1.564644	-2.619745	0.821903
19	0.087047	-0.299007	0.091761	-1.987569
20	-0.219672	0.357113	1.477894	-0.518270
21	-0.808494	-0.501757	0.915402	0.328751
22	-0.529760	0.513267	0.097078	0.968645
23	-0.702053	-0.327662	-0.392108	-1.463515
24	0.296120	0.261055	0.005113	-0.234587
25	-1.415371	-0.420645	-0.342715	-0.802277
26	-0.161286	0.404051	1.886186	0.174578
27	0.257550	-0.074446	-1.918771	-0.026514
28	0.060230	2.463242	-0.192361	0.301547
29	-0.034712	-1.168678	1.142823	0.751933
..	...	...	...	...
370	0.722381	-0.372833	1.726964	-0.399636
371	0.224685	0.932591	-1.418366	-1.760809
372	-1.525656	1.262584	-0.551858	2.558199
373	-0.564248	0.184551	1.542110	2.006093
374	2.061504	1.208366	1.024063	0.592527
375	0.778361	-0.551186	-0.818199	-0.003374
376	-0.170185	-0.453228	0.696387	0.955305
377	0.088407	1.477530	-1.141689	-0.193659
378	-0.716822	-1.866537	-0.082681	-0.121748
379	1.513450	0.630812	-1.024187	1.854093
380	1.221034	0.582098	-0.226484	-0.959439

```
381 -0.372207  1.088749  1.884586  1.543244
382 -0.488849 -1.119617  0.140886 -1.768439
383  0.323168 -0.147603 -0.466037 -1.594703
384  0.513600 -0.532701 -1.169917 -2.872262
385 -0.027515  1.772252  1.661259 -0.457096
386 -0.602212  0.468774 -0.998385  0.301792
387  0.766080  1.226933 -0.100154 -0.203674
388 -0.877983 -0.826880 -0.226479  0.367366
389  0.913585 -0.803179  1.492689 -0.271124
390 -0.021367 -0.747212 -2.424240  0.884045
391  0.736844 -0.281328  0.066991  0.515939
392 -1.562546 -0.529053  0.794265 -1.254289
393  0.293558 -1.356582  0.466430 -0.035641
394 -1.615132  1.164739 -0.734592 -0.810252
395  0.200569  1.148637 -1.015822  0.061680
396  0.428817  0.693106  0.176442 -0.367028
397 -0.827590  0.086144 -1.072139 -2.921350
398  0.436560  0.903935 -2.362932 -1.009731
399  0.619154  2.057495  0.020794 -0.728003

[400 rows x 4 columns]
```

27) Use the "describe" function to summarize the randomdata created above.

In [24]:
```
randomdata.describe()
```

Out[24]:

	0	1	2	3
count	400.000000	400.000000	400.000000	400.000000
mean	0.035296	0.041202	0.046713	0.035939
std	0.924153	1.028034	0.997816	1.024632
min	-2.650970	-2.896255	-3.241267	-2.921350
25%	-0.612942	-0.651856	-0.602684	-0.683766
50%	0.052292	0.037928	0.036043	0.047190
75%	0.634986	0.676014	0.652834	0.744011
max	2.526932	3.852731	3.078881	2.720169

28) Identify the rows which contains at least one item with absolute value greater than 1

In [76]:
```
randomdata[(abs(randomdata[[0,1,2,3]]) > 1).any(axis='columns')]
```

Out[76]:

	0	1	2	3
0	0.496714	-0.138264	0.647689	1.523030
1	-0.234153	-0.234137	1.579213	0.767435
3	0.241962	-1.913280	-1.724918	-0.562288
4	-1.012831	0.314247	-0.908024	-1.412304
5	1.465649	-0.225776	0.067528	-1.424748
6	-0.544383	0.110923	-1.150994	0.375698
7	-0.600639	-0.291694	-0.601707	1.852278
8	-0.013497	-1.057711	0.822545	-1.220844
9	0.208864	-1.959670	-1.328186	0.196861
11	-1.478522	-0.719844	-0.460639	1.057122
12	0.343618	-1.763040	0.324084	-0.385082
13	-0.676922	0.611676	1.031000	0.931280
15	-0.479174	-0.185659	-1.106335	-1.196207
16	0.812526	1.356240	-0.072010	1.003533
17	0.361636	-0.645120	0.361396	1.538037
18	-0.035826	1.564644	-2.619745	0.821903
19	0.087047	-0.299007	0.091761	-1.987569
20	-0.219672	0.357113	1.477894	-0.518270
23	-0.702053	-0.327662	-0.392108	-1.463515
25	-1.415371	-0.420645	-0.342715	-0.802277
26	-0.161286	0.404051	1.886186	0.174578
27	0.257550	-0.074446	-1.918771	-0.026514
28	0.060230	2.463242	-0.192361	0.301547

	0	1	2	3
29	-0.034712	-1.168678	1.142823	0.751933
30	0.791032	-0.909387	1.402794	-1.401851
31	0.586857	2.190456	-0.990536	-0.566298
32	0.099651	-0.503476	-1.550663	0.068563
33	-1.062304	0.473592	-0.919424	1.549934
34	-0.783253	-0.322062	0.813517	-1.230864
35	0.227460	1.307143	-1.607483	0.184634
...	...	...	...	...
362	1.238283	0.021272	0.308833	1.702215
363	0.240753	2.601683	0.565510	-1.760763
364	0.753342	0.381158	1.289753	0.673181
365	-0.138456	-1.224298	-0.209023	-0.850520
366	-0.580523	0.588578	1.669905	0.394672
367	-1.195883	0.444603	1.196631	-0.609783
370	0.722381	-0.372833	1.726964	-0.399636
371	0.224685	0.932591	-1.418366	-1.760809
372	-1.525656	1.262584	-0.551858	2.558199
373	-0.564248	0.184551	1.542110	2.006093
374	2.061504	1.208366	1.024063	0.592527
377	0.088407	1.477530	-1.141689	-0.193659
378	-0.716822	-1.866537	-0.082681	-0.121748
379	1.513450	0.630812	-1.024187	1.854093
380	1.221034	0.582098	-0.226484	-0.959439
381	-0.372207	1.088749	1.884586	1.543244

	0	1	2	3
382	-0.488849	-1.119617	0.140886	-1.768439
383	0.323168	-0.147603	-0.466037	-1.594703
384	0.513600	-0.532701	-1.169917	-2.872262
385	-0.027515	1.772252	1.661259	-0.457096
387	0.766080	1.226933	-0.100154	-0.203674
389	0.913585	-0.803179	1.492689	-0.271124
390	-0.021367	-0.747212	-2.424240	0.884045
392	-1.562546	-0.529053	0.794265	-1.254289
393	0.293558	-1.356582	0.466430	-0.035641
394	-1.615132	1.164739	-0.734592	-0.810252
395	0.200569	1.148637	-1.015822	0.061680
397	-0.827590	0.086144	-1.072139	-2.921350
398	0.436560	0.903935	-2.362932	-1.009731
399	0.619154	2.057495	0.020794	-0.728003

306 rows × 4 columns

29) Set the values greater than 3 to 3, and the values less than -3 to -3. Rerun the 'describe' function to summarize the randomdata dataset.

In [118]:
```
for i in range(4):
    for j in range(400):
        if (randomdata.iloc[[j],[i]]).values.sum()>3.0:
            randomdata.iloc[[j],[i]]=3
        if randomdata.iloc[[j],[i]].values.sum()<-3:
            randomdata.iloc[[j],[i]]=-3

randomdata.describe()
```

Out[118]:

	0	1	2	3
count	400.000000	400.000000	400.000000	400.000000
mean	0.035296	0.039070	0.047119	0.035939
std	0.924153	1.020970	0.995301	1.024632
min	-2.650970	-2.896255	-3.000000	-2.921350
25%	-0.612942	-0.651856	-0.602684	-0.683766
50%	0.052292	0.037928	0.036043	0.047190
75%	0.634986	0.676014	0.652834	0.744011
max	2.526932	3.000000	3.000000	2.720169

Use the dataframe defined in the following cell to answer the next few questions:

In [119]:
```
df2 = pd.DataFrame(np.random.randn(59,4))
cols = ['GrossIn', 'GrossOut', 'NetIn', 'NetOut']
df2.columns = cols
dates = pd.date_range(start='2015-11-29', freq='W', periods=59)
df2.index = dates
df2
```

Out[119]:

	GrossIn	GrossOut	NetIn	NetOut
2015-11-29	-0.182896	1.374876	-0.645964	-0.799192
2015-12-06	-0.482744	-0.953329	0.122670	1.624678
2015-12-13	0.323079	-0.252354	-0.291811	-1.563191
2015-12-20	0.883110	-0.077837	-0.180480	3.193108
2015-12-27	0.298753	-0.751791	-0.426358	1.148446
2016-01-03	0.113270	-1.438278	0.919229	-0.668144
2016-01-10	1.873298	1.080048	-0.447322	1.281016
2016-01-17	0.067856	0.852774	0.484733	-0.846357
2016-01-24	-0.643550	1.029961	-0.334775	-0.403648
2016-01-31	-0.955123	0.423599	2.062525	-1.067533
2016-02-07	0.024219	1.412221	-0.079641	0.452372
2016-02-14	-1.062394	0.428307	-0.187144	0.985730
2016-02-21	1.187386	2.589564	0.579633	0.325796
2016-02-28	0.194384	-0.353166	0.338484	-0.295401
2016-03-06	0.168461	1.317598	-1.006543	1.139879
2016-03-13	1.317115	-0.118069	-2.121855	-0.607822
2016-03-20	1.296995	-0.022868	-0.999302	-0.504775
2016-03-27	0.840620	0.546734	-0.238932	-0.366824
2016-04-03	-0.391758	-0.922410	1.615376	-0.322320
2016-04-10	1.217159	1.521316	0.998311	-0.431620
2016-04-17	0.403730	-0.024196	-0.903702	0.324359
2016-04-24	-1.179040	1.187679	-0.464617	0.201160
2016-05-01	0.283288	-0.258905	0.586694	-0.474904

	GrossIn	GrossOut	NetIn	NetOut
2016-05-08	0.871297	-1.345980	0.126380	1.938929
2016-05-15	-1.000331	-0.677745	0.513908	0.179582
2016-05-22	0.350630	0.489187	0.634721	1.109700
2016-05-29	0.409819	-0.241258	0.672574	1.899882
2016-06-05	-0.132634	-0.974529	1.107081	-0.120381
2016-06-12	-2.172670	0.847422	-0.535328	-0.090533
2016-06-19	0.331980	0.190500	0.709452	-0.435486
2016-06-26	0.513106	-0.259547	0.738810	0.615367
2016-07-03	-0.935439	1.085982	-0.535963	0.808058
2016-07-10	0.367287	1.838184	-0.223466	-0.349317
2016-07-17	-0.019420	-0.303180	0.799942	-1.616311
2016-07-24	-1.053682	-1.067803	0.950308	1.710613
2016-07-31	-0.104449	-0.168822	0.070052	1.161878
2016-08-07	-0.927353	0.238369	0.975198	0.501094
2016-08-14	0.189582	1.001046	-2.703232	0.677875
2016-08-21	-0.654076	-1.830633	0.511203	1.373659
2016-08-28	-0.137449	0.952875	1.612278	1.314914
2016-09-04	1.639965	0.742127	0.075434	-1.601966
2016-09-11	-0.246062	-0.843247	2.170943	-0.175886
2016-09-18	0.123205	0.551485	0.043602	1.695051
2016-09-25	-0.622649	0.194607	-0.742471	-1.320023
2016-10-02	-0.611769	-0.037037	-0.429302	-0.692421
2016-10-09	-1.406317	-0.083106	-1.504720	0.760056
2016-10-16	0.082440	-1.457551	-0.309209	-0.752156

	GrossIn	GrossOut	NetIn	NetOut
2016-10-23	0.319175	1.340450	-1.875172	0.115026
2016-10-30	-0.160133	0.671340	0.213197	-0.751969
2016-11-06	-0.319054	-0.796026	1.076007	0.021312
2016-11-13	1.901191	-0.060661	-0.708407	-1.513714
2016-11-20	-1.803140	-1.584136	0.267127	0.508725
2016-11-27	-1.581191	0.895038	-0.483061	0.146793
2016-12-04	1.612221	0.896839	-0.268531	-0.891192
2016-12-11	-2.151815	-0.719153	-0.211130	-0.987180
2016-12-18	-0.131257	0.076852	-0.224856	-0.650003
2016-12-25	0.168655	0.441941	-1.090399	1.410932
2017-01-01	-0.098588	0.018850	0.708214	0.233216
2017-01-08	0.953137	0.287124	-0.612437	0.361504

30) Print all the data from 2016-06-19 to 2016-10-20 in df2 above.

In [126]:
```
df2.loc['2016-06-19':'2016-10-20']
```

Out[126]:

	GrossIn	GrossOut	NetIn	NetOut
2016-06-19	0.331980	0.190500	0.709452	-0.435486
2016-06-26	0.513106	-0.259547	0.738810	0.615367
2016-07-03	-0.935439	1.085982	-0.535963	0.808058
2016-07-10	0.367287	1.838184	-0.223466	-0.349317
2016-07-17	-0.019420	-0.303180	0.799942	-1.616311
2016-07-24	-1.053682	-1.067803	0.950308	1.710613
2016-07-31	-0.104449	-0.168822	0.070052	1.161878
2016-08-07	-0.927353	0.238369	0.975198	0.501094
2016-08-14	0.189582	1.001046	-2.703232	0.677875
2016-08-21	-0.654076	-1.830633	0.511203	1.373659
2016-08-28	-0.137449	0.952875	1.612278	1.314914
2016-09-04	1.639965	0.742127	0.075434	-1.601966
2016-09-11	-0.246062	-0.843247	2.170943	-0.175886
2016-09-18	0.123205	0.551485	0.043602	1.695051
2016-09-25	-0.622649	0.194607	-0.742471	-1.320023
2016-10-02	-0.611769	-0.037037	-0.429302	-0.692421
2016-10-09	-1.406317	-0.083106	-1.504720	0.760056
2016-10-16	0.082440	-1.457551	-0.309209	-0.752156

31) What was the GrossIn and NetIn for 2016-09-18? Print the results.

In [149]:
```
df2.loc['2016-09-18',['GrossIn','NetIn']]
```

Out[149]:
```
GrossIn    0.123205
NetIn      0.043602
Name: 2016-09-18 00:00:00, dtype: float64
```

32) Sum all the GrossOut and NetOut for the month of March 2016 in df2. Print the results.

In [348]:
```
print (df2.loc[np.logical_and(df2.index.month==3,df2.index.year==2016),['GrossOut']].sum())
print (df2.loc[np.logical_and(df2.index.month==3,df2.index.year==2016),['NetOut']].sum())

#how to add year
#print (df2.loc['2016-03-01':'2016-04-01',['GrossOut']].sum())
#print (df2.loc['2016-03-01':'2016-04-01',['NetOut']].sum())
```

```
GrossOut    1.723395
dtype: float64
NetOut   -0.339543
dtype: float64
```

33) Find all the days GrossOut was greater than GrossIn in df2. Print the results.

In [166]:
```
df2[df2['GrossOut'] > df2['GrossIn']]
```

Out[166]:

	GrossIn	GrossOut	NetIn	NetOut
2015-11-29	-0.182896	1.374876	-0.645964	-0.799192
2016-01-17	0.067856	0.852774	0.484733	-0.846357
2016-01-24	-0.643550	1.029961	-0.334775	-0.403648
2016-01-31	-0.955123	0.423599	2.062525	-1.067533
2016-02-07	0.024219	1.412221	-0.079641	0.452372
2016-02-14	-1.062394	0.428307	-0.187144	0.985730
2016-02-21	1.187386	2.589564	0.579633	0.325796
2016-03-06	0.168461	1.317598	-1.006543	1.139879
2016-04-10	1.217159	1.521316	0.998311	-0.431620
2016-04-24	-1.179040	1.187679	-0.464617	0.201160
2016-05-15	-1.000331	-0.677745	0.513908	0.179582
2016-05-22	0.350630	0.489187	0.634721	1.109700
2016-06-12	-2.172670	0.847422	-0.535328	-0.090533
2016-07-03	-0.935439	1.085982	-0.535963	0.808058
2016-07-10	0.367287	1.838184	-0.223466	-0.349317
2016-08-07	-0.927353	0.238369	0.975198	0.501094
2016-08-14	0.189582	1.001046	-2.703232	0.677875
2016-08-28	-0.137449	0.952875	1.612278	1.314914
2016-09-18	0.123205	0.551485	0.043602	1.695051
2016-09-25	-0.622649	0.194607	-0.742471	-1.320023
2016-10-02	-0.611769	-0.037037	-0.429302	-0.692421
2016-10-09	-1.406317	-0.083106	-1.504720	0.760056
2016-10-23	0.319175	1.340450	-1.875172	0.115026

	GrossIn	GrossOut	NetIn	NetOut
2016-10-30	-0.160133	0.671340	0.213197	-0.751969
2016-11-20	-1.803140	-1.584136	0.267127	0.508725
2016-11-27	-1.581191	0.895038	-0.483061	0.146793
2016-12-11	-2.151815	-0.719153	-0.211130	-0.987180
2016-12-18	-0.131257	0.076852	-0.224856	-0.650003
2016-12-25	0.168655	0.441941	-1.090399	1.410932
2017-01-01	-0.098588	0.018850	0.708214	0.233216

34) What percentage was the total NetOut of the total NetIn in the year 2016 from df2 above?

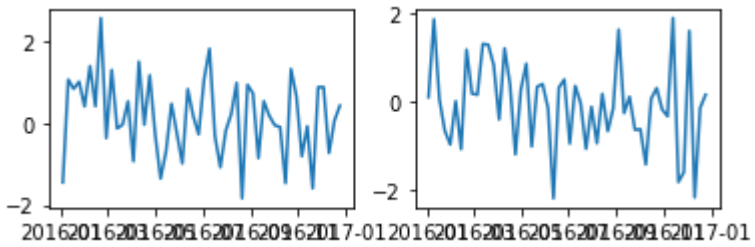
In [315]:
```
NetOut = df2.loc[df2.index.year==2016,['NetOut']].sum()
NetIn = df2.loc[df2.index.year==2016,['NetIn']].sum()

Percentage = (NetOut/float(NetIn))*100

print (Percentage)
```

```
NetOut    212.303243
dtype: float64
```

35) Plot values for 2016 on a line graph for the GrossOut and GrossIn in df2:

In [352]:
```
x = df2.loc[df2.index.year==2016,['GrossOut']]
y = df2.loc[df2.index.year==2016,['GrossIn']]
plt.tight_layout()
plt.subplot(2,2,1)
plt.plot(x)
plt.subplot(2,2,2)
plt.plot(y)
```

Out[352]: [<matplotlib.lines.Line2D at 0x7f862c32ed90>]

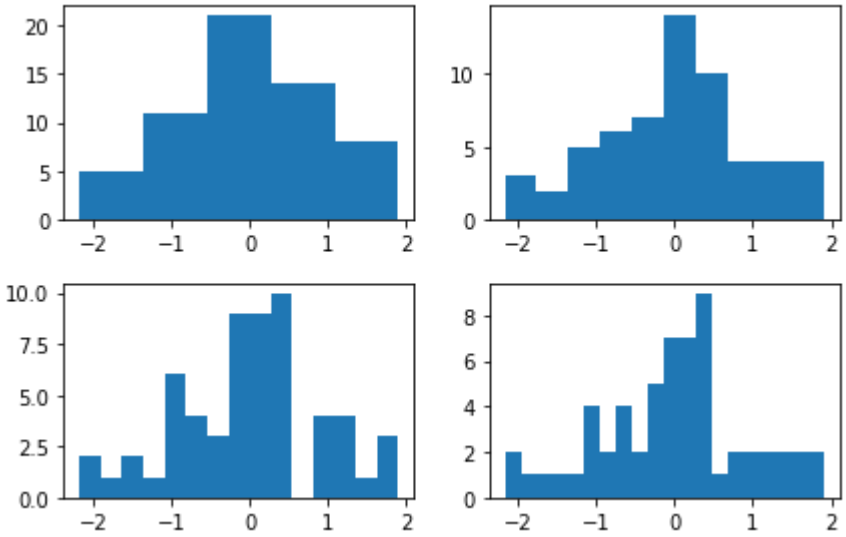


36) Plot 4 different HISTOGRAMS of df2.GrossIn each having 5,10,15,20 bins.

Hint: use the following code to plot the histograms on different subplots.

import matplotlib.pyplot as plt
fig, ax = plt.subplots(nrows=2, ncols=2)
dataframe.plot(ax=ax[0,0])# this is just an example of how to plot on a top right subplot (2x2)

In [193]:
```
#fig, ax = plt.subplots(nrows=2, ncols=2)
x = df2['GrossIn']

plt.subplot(2,2,1)
plt.hist(x, bins = 5)

plt.subplot(2,2,2)
plt.hist(x, bins = 10)

plt.subplot(2,2,3)
plt.hist(x, bins = 15)

plt.subplot(2,2,4)
plt.hist(x, bins = 20)

#df2.plot(ax=ax[0,0])

#df2['GrossIn'].plot(ax=ax[0,1])

#df2['GrossIn'].plot(ax=ax[1,0])

#df2['GrossIn'].plot(ax=ax[1,1])

plt.tight_layout()

#plt.subplot(2,2,1)
```



37) What is your favorite pizza topping? None is not a valid answer. I don't care if you are off gluten.

In [ ]:
```
salami
```

38) Import pandas libary with the alias pd

In [197]:
```
import pandas as pd
```

39)You are given the Medicare spending dataset. Read it into a pandas dataframe. Then remove the 2 unused columns 'Measure Start Date' and 'Measure End Date' from the DataFrame all in the following cell. Provide your code as the answer.

You need to put the data file 'Medicare_Hospital_Spending_by_Claim.csv' at the same place as your notebook file.

In [361]:
```
Medicare = pd.read_csv("Medicare_Hospital_Spending_by_Claim.csv", index_col = 0)
Medicare.drop(['Measure Start Date', 'Measure End Date'],axis = 'columns', inplace = True)
#print (Medicare.dtypes)
Medicare.head()
```

Out[361]:

	Provider Number	State	Period	Claim Type	Avg Spending Per Episode (Hospital)	Avg Spending Per Episode (State)	Avg Spending Per Episode (Nation)	Percent of Spending (Hospital)	Percent of Spending (State)	Percent of Spending (Nation)
Hospital Name										
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Home Health Agency	$12	$14	$13	0.06%	0.07%	0.07%
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Hospice	$1	$1	$1	0.01%	0.01%	0%
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Inpatient	$6	$6	$5	0.03%	0.03%	0.03%
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Outpatient	$160	$85	$117	0.84%	0.46%	0.58%
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Skilled Nursing Facility	$1	$2	$2	0.01%	0.01%	0.01%

40) Continuing with data from above, convert the dataframe content to be the right "data type" (i.e., int, float, etc) in order to perform statistical calculations for the DataFrame. Provide your code and the resulting new print out as the answer.
Hint convert the columns with % and $ using regex

In [376]:
```
#Medicare.dtypes
#Medicare.columns

#cols=Medicare.columns.values
#for key,value in Medicare.iteritems():
    #print key,value
 #   value.replace('$','')
  #  value.replace('%','')

#if key in cols[4:7]:
#        Medicare[key] = Medicare[key].astype(int)


#for x in range(4,7):
#Medicare.iloc[:,[0]]


#for x in range(7,10):
 #   Medicare.iloc[:,x] = Medicare.iloc[:,x].astype(float)

Medicare['Avg Spending Per Episode (Hospital)'] = Medicare['Avg Spending Per Episode (Hospital)'].str.replace
('$','')
Medicare['Avg Spending Per Episode (State)'] = Medicare['Avg Spending Per Episode (State)'].str.replace('$',
'')
Medicare['Avg Spending Per Episode (Nation)'] = Medicare['Avg Spending Per Episode (Nation)'].str.replace('$'
,'')
Medicare['Percent of Spending (Hospital)'] = Medicare['Percent of Spending (Hospital)'].str.replace('%','')
Medicare['Percent of Spending (State)'] = Medicare['Percent of Spending (State)'].str.replace('%','')
Medicare['Percent of Spending (Nation)'] = Medicare['Percent of Spending (Nation)'].str.replace('%','')


Medicare['Avg Spending Per Episode (Hospital)'] = Medicare['Avg Spending Per Episode (Hospital)'].astype(int)
Medicare['Avg Spending Per Episode (State)'] = Medicare['Avg Spending Per Episode (State)'].astype(int)
Medicare['Avg Spending Per Episode (Nation)'] = Medicare['Avg Spending Per Episode (Nation)'].astype(int)
Medicare['Percent of Spending (Hospital)'] = Medicare['Percent of Spending (Hospital)'].astype(float)
Medicare['Percent of Spending (State)'] = Medicare['Percent of Spending (State)'].astype(float)
Medicare['Percent of Spending (Nation)'] = Medicare['Percent of Spending (Nation)'].astype(float)

print (Medicare.dtypes)
Medicare.head()
```

```
Provider Number                          int64
State                                   object
Period                                  object
Claim Type                              object
Avg Spending Per Episode (Hospital)      int64
Avg Spending Per Episode (State)         int64
Avg Spending Per Episode (Nation)        int64
Percent of Spending (Hospital)         float64
Percent of Spending (State)            float64
Percent of Spending (Nation)           float64
dtype: object
```

Out[376]:

	Provider Number	State	Period	Claim Type	Avg Spending Per Episode (Hospital)	Avg Spending Per Episode (State)	Avg Spending Per Episode (Nation)	Percent of Spending (Hospital)	Percent of Spending (State)	Percent of Spending (Nation)
Hospital Name										
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Home Health Agency	12	14	13	0.06	0.07	0.07
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Hospice	1	1	1	0.01	0.01	0.00
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Inpatient	6	6	5	0.03	0.03	0.03
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Outpatient	160	85	117	0.84	0.46	0.58
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Skilled Nursing Facility	1	2	2	0.01	0.01	0.01

41) Replace columns names...replace the original long column names in the dataframe above (Q - 39) with shorter ones provided below in "newColumns". Provide your code and the resulting new column names as the answer.

newColumns = ['Hospital', 'Provider ID', 'State', 'Period', 'Claim Type', 'Avg Spending Hospital', 'Avg Spending State', 'Avg Spending Nation', 'Percent Spending Hospital', 'Percent Spending State', 'Percent Spending Nation']

In [255]:
```
newColumns = ['Hospital', 'Provider ID', 'State', 'Period', 'Claim Type', 'Avg Spending Hospital', 'Avg Spend
ing State', 'Avg Spending Nation', 'Percent Spending Hospital', 'Percent Spending State', 'Percent Spending N
ation']


Medicare.index.name = newColumns[0]
Medicare.columns = newColumns[1:]

#Medicare.index.names
Medicare.head()
```

Out[255]:

	Provider ID	State	Period	Claim Type	Avg Spending Hospital	Avg Spending State	Avg Spending Nation	Percent Spending Hospital	Percent Spending State	Percent Spending Nation
Hospital										
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Home Health Agency	12	14	13	0.06	0.07	0.07
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Hospice	1	1	1	0.01	0.01	0.00
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Inpatient	6	6	5	0.03	0.03	0.03
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Outpatient	160	85	117	0.84	0.46	0.58
SOUTHEAST ALABAMA MEDICAL CENTER	10001	AL	1 to 3 days Prior to Index Hospital Admission	Skilled Nursing Facility	1	2	2	0.01	0.01	0.01

42) Continuing with the dataframe above, extract only the TX state claims into a new dataframe. Provide your code and print the new dataframe (with only TX claims) below as the answer.

In [259]:
```
TX_Claims = Medicare[Medicare['State'] == 'TX']
TX_Claims.head()
```

Out[259]:

	Provider ID	State	Period	Claim Type	Avg Spending Hospital	Avg Spending State	Avg Spending Nation	Percent Spending Hospital	Percent Spending State	Percent Spending Nation
Hospital										
PROVIDENCE MEMORIAL HOSPITAL	450002	TX	1 to 3 days Prior to Index Hospital Admission	Home Health Agency	52	23	13	0.27	0.11	0.07
PROVIDENCE MEMORIAL HOSPITAL	450002	TX	1 to 3 days Prior to Index Hospital Admission	Hospice	0	1	1	0.00	0.01	0.00
PROVIDENCE MEMORIAL HOSPITAL	450002	TX	1 to 3 days Prior to Index Hospital Admission	Inpatient	2	5	5	0.01	0.02	0.03
PROVIDENCE MEMORIAL HOSPITAL	450002	TX	1 to 3 days Prior to Index Hospital Admission	Outpatient	80	122	117	0.41	0.57	0.58
PROVIDENCE MEMORIAL HOSPITAL	450002	TX	1 to 3 days Prior to Index Hospital Admission	Skilled Nursing Facility	1	3	2	0.01	0.01	0.01

43) Using the dataframe from above (Q 42), find the average spending for each different TX hospitals groupedby each kind of different claim types. Provide your code and the resulting new print out as the answer.

In [268]:
```
TX_Claims.groupby(['Hospital','Claim Type'])['Avg Spending Hospital','Avg Spending State','Avg Spending Nation'].mean()

#Medicare.groupby(['Hospital','Claim Type']).mean()
```

Out[268]:

		Avg Spending Hospital	Avg Spending State	Avg Spending Nation
Hospital	**Claim Type**			
ABILENE REGIONAL MEDICAL CENTER	**Carrier**	988.666667	1119.666667	1043.000000
	Durable Medical Equipment	42.333333	51.000000	44.666667
	Home Health Agency	322.666667	293.666667	261.333333
	Hospice	38.333333	48.666667	39.666667
	Inpatient	4476.333333	4402.333333	3926.000000
	Outpatient	281.666667	277.333333	275.666667
	Skilled Nursing Facility	789.666667	968.666667	1084.333333
	Total	20821.000000	21484.000000	20025.000000
ANSON GENERAL HOSPITAL	**Carrier**	341.000000	1119.666667	1043.000000
	Durable Medical Equipment	24.666667	51.000000	44.666667
	Home Health Agency	81.666667	293.666667	261.333333
	Hospice	33.333333	48.666667	39.666667
	Inpatient	2073.000000	4402.333333	3926.000000
	Outpatient	275.666667	277.333333	275.666667
	Skilled Nursing Facility	1013.333333	968.666667	1084.333333
	Total	11528.000000	21484.000000	20025.000000
ARISE AUSTIN MEDICAL CENTER	**Carrier**	1105.666667	1119.666667	1043.000000
	Durable Medical Equipment	73.666667	51.000000	44.666667

		Avg Spending Hospital	Avg Spending State	Avg Spending Nation
Hospital	**Claim Type**			
	Home Health Agency	516.333333	293.666667	261.333333
	Hospice	0.000000	48.666667	39.666667
	Inpatient	4689.333333	4402.333333	3926.000000
	Outpatient	76.333333	277.333333	275.666667
	Skilled Nursing Facility	147.000000	968.666667	1084.333333
	Total	19825.000000	21484.000000	20025.000000
BAPTIST BEAUMONT HOSPITAL	**Carrier**	935.666667	1119.666667	1043.000000
	Durable Medical Equipment	36.333333	51.000000	44.666667
	Home Health Agency	283.333333	293.666667	261.333333
	Hospice	101.666667	48.666667	39.666667
	Inpatient	4094.333333	4402.333333	3926.000000
	Outpatient	263.666667	277.333333	275.666667
...	**...**	...	...	...
WILSON N. JONES REGIONAL MEDICAL CENTER	**Home Health Agency**	289.333333	293.666667	261.333333
	Hospice	81.666667	48.666667	39.666667
	Inpatient	4071.333333	4402.333333	3926.000000
	Outpatient	185.333333	277.333333	275.666667
	Skilled Nursing Facility	1397.333333	968.666667	1084.333333
	Total	21467.000000	21484.000000	20025.000000
WISE REGIONAL HEALTH SYSTEM	**Carrier**	1084.000000	1119.666667	1043.000000

		Avg Spending Hospital	Avg Spending State	Avg Spending Nation
Hospital	**Claim Type**			
	Durable Medical Equipment	70.333333	51.000000	44.666667
	Home Health Agency	330.333333	293.666667	261.333333
	Hospice	58.000000	48.666667	39.666667
	Inpatient	4312.666667	4402.333333	3926.000000
	Outpatient	271.000000	277.333333	275.666667
	Skilled Nursing Facility	1458.666667	968.666667	1084.333333
	Total	22756.000000	21484.000000	20025.000000
WOMANS HOSPITAL OF TEXAS,THE	**Carrier**	876.333333	1119.666667	1043.000000
	Durable Medical Equipment	30.666667	51.000000	44.666667
	Home Health Agency	31.666667	293.666667	261.333333
	Hospice	0.000000	48.666667	39.666667
	Inpatient	2454.666667	4402.333333	3926.000000
	Outpatient	126.666667	277.333333	275.666667
	Skilled Nursing Facility	81.333333	968.666667	1084.333333
	Total	10803.000000	21484.000000	20025.000000
WOODLAND HEIGHTS MEDICAL CENTER	**Carrier**	959.666667	1119.666667	1043.000000
	Durable Medical Equipment	34.666667	51.000000	44.666667
	Home Health Agency	262.666667	293.666667	261.333333
	Hospice	34.333333	48.666667	39.666667

		Avg Spending Hospital	Avg Spending State	Avg Spending Nation
Hospital	**Claim Type**			
	Inpatient	3640.333333	4402.333333	3926.000000
	Outpatient	242.333333	277.333333	275.666667
	Skilled Nursing Facility	893.000000	968.666667	1084.333333
	Total	18199.000000	21484.000000	20025.000000

2352 rows × 3 columns

44) Continuing from above (Q 43), what is the mean and max percentage spending of different TX hospitals for each different claim types. Provide your code and print out the answer.

In [283]:
```
#TX_Claims.head()
print TX_Claims.groupby(['Hospital','Claim Type'])['Percent Spending Hospital'].mean()

print ('************************************'
       '************************************')

print TX_Claims.groupby(['Hospital','Claim Type'])['Percent Spending Hospital'].max()

#Mean_Spending_Per.head()
#Max_Spending_Per.head()
```

Hospital	Claim Type	
ABILENE REGIONAL MEDICAL CENTER	Carrier	4.750000
	Durable Medical Equipment	0.203333
	Home Health Agency	1.550000
	Hospice	0.183333
	Inpatient	21.500000
	Outpatient	1.353333
	Skilled Nursing Facility	3.793333
	Total	100.000000
ANSON GENERAL HOSPITAL	Carrier	2.960000
	Durable Medical Equipment	0.213333
	Home Health Agency	0.706667
	Hospice	0.290000
	Inpatient	17.980000
	Outpatient	2.390000
	Skilled Nursing Facility	8.790000
	Total	100.000000
ARISE AUSTIN MEDICAL CENTER	Carrier	5.576667
	Durable Medical Equipment	0.370000
	Home Health Agency	2.603333
	Hospice	0.000000
	Inpatient	23.656667
	Outpatient	0.386667
	Skilled Nursing Facility	0.743333
	Total	100.000000
BAPTIST BEAUMONT HOSPITAL	Carrier	4.763333
	Durable Medical Equipment	0.186667
	Home Health Agency	1.443333
	Hospice	0.520000
	Inpatient	20.846667
	Outpatient	1.343333
		...
WILSON N. JONES REGIONAL MEDICAL CENTER	Home Health Agency	1.350000
	Hospice	0.380000
	Inpatient	18.963333
	Outpatient	0.866667
	Skilled Nursing Facility	6.510000
	Total	100.000000
WISE REGIONAL HEALTH SYSTEM	Carrier	4.763333
	Durable Medical Equipment	0.306667
	Home Health Agency	1.450000
	Hospice	0.256667
	Inpatient	18.953333

```
                                    Outpatient                    1.190000
                                    Skilled Nursing Facility      6.410000
                                    Total                       100.000000
WOMANS HOSPITAL OF TEXAS,THE        Carrier                       8.110000
                                    Durable Medical Equipment     0.283333
                                    Home Health Agency            0.293333
                                    Hospice                       0.000000
                                    Inpatient                    22.720000
                                    Outpatient                    1.173333
                                    Skilled Nursing Facility      0.750000
                                    Total                       100.000000
WOODLAND HEIGHTS MEDICAL CENTER     Carrier                       5.270000
                                    Durable Medical Equipment     0.190000
                                    Home Health Agency            1.443333
                                    Hospice                       0.190000
                                    Inpatient                    20.003333
                                    Outpatient                    1.330000
                                    Skilled Nursing Facility      4.906667
                                    Total                       100.000000
Name: Percent Spending Hospital, Length: 2352, dtype: float64
**************************************************************************
Hospital                            Claim Type
ABILENE REGIONAL MEDICAL CENTER     Carrier                       7.12
                                    Durable Medical Equipment     0.42
                                    Home Health Agency            4.56
                                    Hospice                       0.55
                                    Inpatient                    47.46
                                    Outpatient                    3.02
                                    Skilled Nursing Facility     11.37
                                    Total                       100.00
ANSON GENERAL HOSPITAL              Carrier                       4.77
                                    Durable Medical Equipment     0.30
                                    Home Health Agency            2.12
                                    Hospice                       0.87
                                    Inpatient                    43.02
                                    Outpatient                    5.98
                                    Skilled Nursing Facility     26.37
                                    Total                       100.00
ARISE AUSTIN MEDICAL CENTER         Carrier                      10.45
                                    Durable Medical Equipment     0.69
                                    Home Health Agency            7.64
                                    Hospice                       0.00
                                    Inpatient                    59.14
```

```
                                        Outpatient                    1.05
                                        Skilled Nursing Facility      2.23
                                        Total                       100.00
BAPTIST BEAUMONT HOSPITAL               Carrier                       5.83
                                        Durable Medical Equipment     0.34
                                        Home Health Agency            4.20
                                        Hospice                       1.55
                                        Inpatient                    43.14
                                        Outpatient                    3.64
                                                                       ...
WILSON N. JONES REGIONAL MEDICAL CENTER Home Health Agency            3.91
                                        Hospice                       1.09
                                        Inpatient                    41.21
                                        Outpatient                    2.24
                                        Skilled Nursing Facility     19.50
                                        Total                       100.00
WISE REGIONAL HEALTH SYSTEM             Carrier                       7.01
                                        Durable Medical Equipment     0.56
                                        Home Health Agency            4.22
                                        Hospice                       0.74
                                        Inpatient                    42.29
                                        Outpatient                    3.40
                                        Skilled Nursing Facility     19.21
                                        Total                       100.00
WOMANS HOSPITAL OF TEXAS,THE            Carrier                      12.88
                                        Durable Medical Equipment     0.62
                                        Home Health Agency            0.88
                                        Hospice                       0.00
                                        Inpatient                    60.20
                                        Outpatient                    2.28
                                        Skilled Nursing Facility      2.25
                                        Total                       100.00
WOODLAND HEIGHTS MEDICAL CENTER         Carrier                       7.86
                                        Durable Medical Equipment     0.46
                                        Home Health Agency            4.27
                                        Hospice                       0.57
                                        Inpatient                    48.01
                                        Outpatient                    3.35
                                        Skilled Nursing Facility     14.71
                                        Total                       100.00
Name: Percent Spending Hospital, Length: 2352, dtype: float64
```

45) The below DataFrame has a column of groups 'grps' and and column of numbers 'vals'.
For each group, find the *sum of the three greatest values.***

In [321]:
```
df = pd.DataFrame({'grps': list('aaabbcaabcccbbc'),
                   'vals': [12,345,3,1,45,14,4,52,54,23,235,21,57,3,87]})
grouped = df.groupby('grps')

for name,group in grouped:
    sorted_group=group.sort_values(by='vals')
    print (sorted_group[-3:].aggregate(np.sum))
```

```
grps    aaa
vals    409
dtype: object
grps    bbb
vals    156
dtype: object
grps    ccc
vals    345
dtype: object
```

46) Some values in the values in the FlightNumber column are missing. These numbers are meant to increase by 10 with each row so 10055 and 10075 need to be put in place. Fill in these missing numbers and make the column an integer column (instead of a float column). ***Use the dataframe below***

In [305]:
```
df = pd.DataFrame({'From_To': ['LoNDon_paris', 'MAdrid_miLAN', 'londON_StockhOlm','Budapest_PaRis','Brussels_
londOn'],'FlightNumber': [10045, np.nan, 10065, np.nan, 10085],'RecentDelays': [[23, 47], [], [24, 43, 87], [
13], [67, 32]],'Airline': ['KLM(!)', '<Air France> (12)', '(British Airways. )', '12. Air France', '"Swiss Ai
r"']})
df
```

Out[305]:

	Airline	FlightNumber	From_To	RecentDelays
0	KLM(!)	10045.0	LoNDon_paris	[23, 47]
1	<Air France> (12)	NaN	MAdrid_miLAN	[]
2	(British Airways.)	10065.0	londON_StockhOlm	[24, 43, 87]
3	12. Air France	NaN	Budapest_PaRis	[13]
4	"Swiss Air"	10085.0	Brussels_londOn	[67, 32]

In [310]:
```
FlightNum = 10045
for i in range(5):
    df.loc[[i],['FlightNumber']] = FlightNum
    FlightNum = FlightNum + 10

df['FlightNumber'] = df['FlightNumber'].astype(int)

df.head()
```

Out[310]:

	Airline	FlightNumber	From_To	RecentDelays
0	KLM(!)	10045	LoNDon_paris	[23, 47]
1	<Air France> (12)	10055	MAdrid_miLAN	[]
2	(British Airways.)	10065	londON_StockhOlm	[24, 43, 87]
3	12. Air France	10075	Budapest_PaRis	[13]
4	"Swiss Air"	10085	Brussels_londOn	[67, 32]

47) Give a very general and short definition of what an Analytical Model is or is supposed to do?

An analytical model estimates or classifies data values by essentially drawing a line through data points. When applied to new data or records, a model can predict outcomes based on historical patterns.

48) Briefly describe the differences between "supervised" and "unsupervised" learning.

Supervised Learning Training data consist of a set of training examples. Each example consists of an input object and a desired output value. It analyzes the training data and produces an inferred function, which can be used for mapping new examples and will allow for the algorithm to correctly determine the class labels for unseen instances. It requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. Unsupervised Learning The task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). The examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm.

49) Briefly describe the differences between "classification" and "clustering".

The difference between "classification" and "clustering" is classification is unsupervised learning and the task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations).

Clustering is in the Data Mining which is the analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining, sequential pattern mining).

50) Name 3 different "kinds" of Analytical Models and provide a brief definition of each.

Statistical modeling:

It is a mathematical model where we can make assumptions about the generation of sample data and similar data from a larger population. It represent the data-generating process and describes a set of probability distributions that are assumed to approximate the distribution for a particular data set. It is specified by equations that relate one or more random variables and possibly other non-random variables.

Machine Learning:

It isfield of computer science which gives computers the ability to learn without being explicitly programmed. It evolved from the study of pattern recognition and computational learning theory in artificial intelligence and explores the study and construction of algorithms that can learn from and make predictions on data. It is sometimes conflated with data mining where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. In the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction.

Data Mining:

It is computing process of discovering patterns in large data sets which involves methods at the intersection of machine learning, statistics, and database systems. It is an interdisciplinary subfield of computer science The overall goal s to extract information from a data set and transform it into an understandable structure for further use and involves database and data management aspects, data pre processing, model and inference considerations, post-processing of discovered structures, visualization, and online updating. This term is a misnomer because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself

EXTRA CREDIT 51) Why is Ordinary Least Squares Regression, called Ordinary Least Squares?

Ordinary Least Squares Regression is called Ordinary Least Squares which is a method for performing linear regression. It is called Ordinary Least Squares because it chooses the parameters of a linear function of a set of explanatory variables by minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being predicted) in the given dataset and those predicted by the linear function.

EXTRA CREDIT 52) Why do vicious feedback loops matter in data analytics?

In [ ]:
```
vicious circle refer to complex chains of events that reinforce themselves through a feedback loop.
vicious feedback loops matter in data analytics because the vicious circle has detrimental results in the sho
rt run.
Vicious feedback loops in which each iteration of the cycle reinforces the previous one (positive feedback).
```

EXTRA CREDIT 53) What are the 10 simple rules Zook et. al. Suggest to follow when doing big data analytics?

In [ ]:
```
1. Acknowledge that data are people and can do harm
2. Recognize that privacy is more than a binary value
3. Guard against the reidentification of your data
4. Practice ethical data sharing
5. Consider the strengths and limitations of your data; big does not automatically mean better
6. Debate the tough, ethical choices
7. Develop a code of conduct for your organization, research community, or industry
8. Design your data and systems for auditability
9. Engage with the broader consequences of data and analysis practices
10. Know when to break these rules
```