Data Wrangling

Hierachial indexing

In [4]:
```
import numpy as np
import pandas as pd

data = pd.Series(np.random.randn(9), index=[['a', 'a', 'a', 'a', 'b',
data
```

Out[4]:
```
a  1    0.464966
   2   -0.116844
   3   -0.004248
   4   -0.749541
b  5   -0.860667
   6   -0.348301
   7   -1.342750
c  8    2.211314
   9   -2.451105
dtype: float64
```

In [5]:
```
data[['a', 'c']]
```

Out[5]:
```
a  1    0.464966
   2   -0.116844
   3   -0.004248
   4   -0.749541
c  8    2.211314
   9   -2.451105
dtype: float64
```

In [8]:
```
data.loc['b', 5]
```

Out[8]:
```
-0.8606670439526144
```

In [9]:
```
data.unstack() #forms a pivot table.
```

Out[9]:

	1	2	3	4	5	6	7	8	
a	0.464966	-0.116844	-0.004248	-0.749541	NaN	NaN	NaN	NaN	N
b	NaN	NaN	NaN	NaN	-0.860667	-0.348301	-1.34275	NaN	N
c	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.211314	-2.451

In [10]:
```
data.unstack().stack()
```

Out[10]:
```
a  1    0.464966
   2   -0.116844
   3   -0.004248
   4   -0.749541
b  5   -0.860667
   6   -0.348301
   7   -1.342750
c  8    2.211314
   9   -2.451105
dtype: float64
```

In [19]:
```
data = pd.DataFrame(np.arange(12).reshape(3, 4), index = [['a', 'a', '
data
```

Out[19]:

		0	1	2	3
a	1	0	1	2	3
	2	4	5	6	7
b	3	8	9	10	11

In [20]:
```
data.index.names = ['key1', 'key2']
data
```

Out[20]:

		0	1	2	3
key1	key2				
a	1	0	1	2	3
	2	4	5	6	7
b	3	8	9	10	11

Reordering and Sorting Levels

In [21]:
```
data.swaplevel('key1', 'key2')
```

Out[21]:

		0	1	2	3
key2	key1				
1	a	0	1	2	3
2	a	4	5	6	7
3	b	8	9	10	11

In [25]:
```
data.sort_index(level=1)
```

Out[25]:

key1	key2	0	1	2	3
a	1	0	1	2	3
	2	4	5	6	7
b	3	8	9	10	11

Summary Statistics by Level

In [30]:
```
data.sum(level='key1', axis=0)
```

Out[30]:

key1	0	1	2	3
a	4	6	8	10
b	8	9	10	11

Combining and Mergine Datasets

In [83]:
```
df1 = pd.DataFrame({'name': ['tom', 'giles', 'mark', 'nicola'], 'temp'
df2 = pd.DataFrame({'name': ['giles', 'tom', 'nicola', 'mark'], 'heigh
```

In [84]:
```
df1
```

Out[84]:

	name	temp
0	tom	36
1	giles	36
2	mark	37
3	nicola	36

In [85]:
```
df2
```

Out[85]:

	name	height
0	giles	165
1	tom	185
2	nicola	134
3	mark	25000

In [88]:
```
df3 = pd.merge(df1, df2) # automatically merges on common column
df3 = pd.merge(df1, df2, on='name') #can choose the column to merge
df3
```

Out[88]:

	name	temp	height
0	tom	36	185
1	giles	36	165
2	mark	37	25000
3	nicola	36	134

merge function arguments

left -- DataFrame to be merged on the left side.
right -- DataFrame to be merged on the right side.
how -- One of 'inner', 'outer', 'left' or 'right; defaults to 'inner.
on -- column names to joing. Must be found in both DataFrame objects, if not sepcified and no other join keys given, will use the intersection of the column names in left and right as the join keys.
left_on -- Columns in left DataFrame to use as join keys.
right_on --Analogous to left_on
left_index -- Use row index in left as its join key
right_index -- Analogous to left_index
sort -- Sort merged data lexicographically by join keys.
suffixes -- Tuple of string values to append to column names in case of overlap
copy -- if False, avoid copying data into resulting data structure in some exceptional cases;
ndicator -- Adds a special column *merge that indicates the source of each row; alues will be 'left*only', 'right_only', or 'both' based on the origin of the joined data in each row

Reshaping and Pivoting

- stack - This "rotates" or pivots from the columns in the data to the rows
- unstack - This pivots from the rows into the columns

In [97]:
```
frame = pd.DataFrame(np.arange(6).reshape(3, 2), index=pd.Index(['east
frame
```

Out[97]:

area	quant	val2
east	0	1
west	2	3
north	4	5

In [107]:
```
frame.stack(0) # can unstack a different level by specifing an integer
```

Out[107]:
```
area   quant
east   val      0
       val2     1
west   val      2
       val2     3
north  val      4
       val2     5
dtype: int32
```