Analysis of unstructured data

Lecture 4 - `pandas` examples

Janusz Szwabiński

Overview:

- Reading from CSV files
- Groupby and aggregate
- Searching for information
- Data cleaning
- String operations
- Parsing Unix timestamps
- Loading data from SQL
- Pandas vs SQL

Referencje:

- Homepage of `pandas` project: http://pandas.pydata.org/ (http://pandas.pydata.org/)
- Pandas Cookbook, https://github.com/jvns/pandas-cookbook (https://github.com/jvns/pandas-cookbook)

Data sources:

- http://donnees.ville.montreal.qc.ca/dataset/velos-comptage (http://donnees.ville.montreal.qc.ca/dataset/velos-comptage) - cyclist data from Montreal
- http://climate.weather.gc.ca/index_e.html (http://climate.weather.gc.ca/index_e.html) - Canadian weather data
- https://nycopendata.socrata.com/ (https://nycopendata.socrata.com/) - 311 service requests from NYC

In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
```

Reading from CSV files

In [2]:

```
! head data/bikes.csv
```

In [3]:

```
broken_df = pd.read_csv('data/bikes.csv')
broken_df[:3]
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent cal
l last)
<ipython-input-3-823d15dd87ce> in <module>()
----> 1 broken_df = pd.read_csv('data/bikes.csv')
      2 broken_df[:3]

/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py in parse
r_f(filepath_or_buffer, sep, delimiter, header, names, index_col, us
ecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters,
 true_values, false_values, skipinitialspace, skiprows, nrows, na_va
lues, keep_default_na, na_filter, verbose, skip_blank_lines, parse_d
ates, infer_datetime_format, keep_date_col, date_parser, dayfirst, i
terator, chunksize, compression, thousands, decimal, lineterminator,
 quotechar, quoting, escapechar, comment, encoding, dialect, tupleiz
e_cols, error_bad_lines, warn_bad_lines, skipfooter, skip_footer, do
ublequote, delim_whitespace, as_recarray, compact_ints, use_unsigne
d, low_memory, buffer_lines, memory_map, float_precision)
    643                     skip_blank_lines=skip_blank_lines)
    644
--> 645         return _read(filepath_or_buffer, kwds)
    646
    647     parser_f.__name__ = name

/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py in
_read(filepath_or_buffer, kwds)
    386
    387     # Create the parser.
--> 388     parser = TextFileReader(filepath_or_buffer, **kwds)
    389
    390     if (nrows is not None) and (chunksize is not None):

/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py in __ini
t__(self, f, engine, **kwds)
    727             self.options['has_index_names'] = kwds['has_inde
x_names']
    728
--> 729         self._make_engine(self.engine)
    730
    731     def close(self):

/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py in _make
_engine(self, engine)
    920     def _make_engine(self, engine='c'):
    921         if engine == 'c':
--> 922             self._engine = CParserWrapper(self.f, **self.opt
ions)
    923         else:
    924             if engine == 'python':

/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py in __ini
t__(self, src, **kwds)
   1387         kwds['allow_leading_cols'] = self.index_col is not F
alse
   1388
-> 1389         self._reader = _parser.TextReader(src, **kwds)
   1390
   1391         # XXX

pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/pars
er.c:6077)()

pandas/parser.pyx in pandas.parser.TextReader._get_header (pandas/pa
rser.c:9215)()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe9 in position
 15: invalid continuation byte
```

In [4]:

```
# we set the separator to ';' and change the encoding to latin1
# we want to parse the dates
# we want the dates to have the day first
# we set the index to be the date column
fixed_df = pd.read_csv('data/bikes.csv', sep=';', encoding='latin1',
parse_dates=['Date'], dayfirst=True, index_col='Date')
fixed_df[:3]
```

Out[4]:

	Berri 1	**Brébeuf (données non disponibles)**	**Côte-Sainte-Catherine**	**Maisonneuve 1**	**Maisonneuve 2**	**du Parc**	**Pierre-Dupuy**	**Rach**
Date								
2012-01-01	35	NaN	0	38	51	26	10	16
2012-01-02	83	NaN	1	68	153	53	6	43
2012-01-03	135	NaN	2	104	248	89	3	58

Selecting a column

In [5]:

```
fixed_df['Berri 1']
```

```
Out[5]:

Date
2012-01-01      35
2012-01-02      83
2012-01-03     135
2012-01-04     144
2012-01-05     197
2012-01-06     146
2012-01-07      98
2012-01-08      95
2012-01-09     244
2012-01-10     397
2012-01-11     273
2012-01-12     157
2012-01-13      75
2012-01-14      32
2012-01-15      54
2012-01-16     168
2012-01-17     155
2012-01-18     139
2012-01-19     191
2012-01-20     161
2012-01-21      53
2012-01-22      71
2012-01-23     210
2012-01-24     299
2012-01-25     334
2012-01-26     306
2012-01-27      91
2012-01-28      80
2012-01-29      87
2012-01-30     219
              ... 
2012-10-07    1580
2012-10-08    1854
2012-10-09    4787
2012-10-10    3115
2012-10-11    3746
2012-10-12    3169
2012-10-13    1783
2012-10-14     587
2012-10-15    3292
2012-10-16    3739
2012-10-17    4098
2012-10-18    4671
2012-10-19    1313
2012-10-20    2011
2012-10-21    1277
2012-10-22    3650
2012-10-23    4177
2012-10-24    3744
2012-10-25    3735
2012-10-26    4290
2012-10-27    1857
2012-10-28    1310
2012-10-29    2919
2012-10-30    2887
2012-10-31    2634
2012-11-01    2405
2012-11-02    1582
2012-11-03     844
```

```
2012-11-04     966
2012-11-05    2247
Name: Berri 1, dtype: int64
```

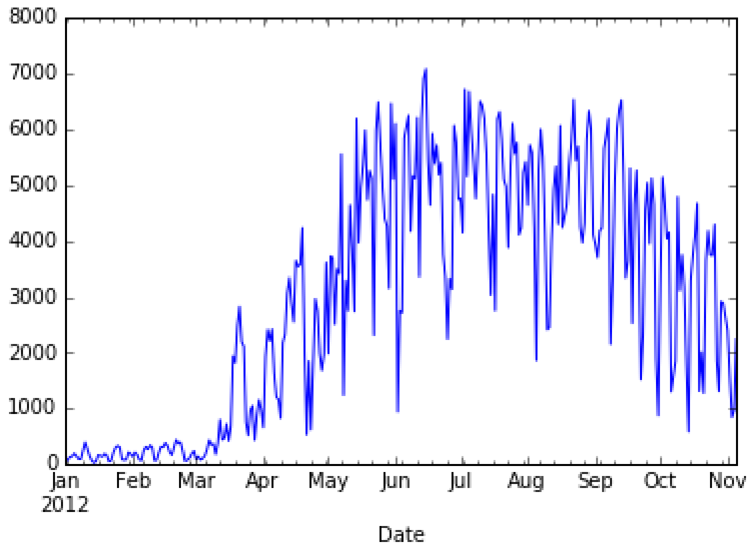
Plotting a column

In [6]:

```
fixed_df['Berri 1'].plot()
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19c341bda0>
```



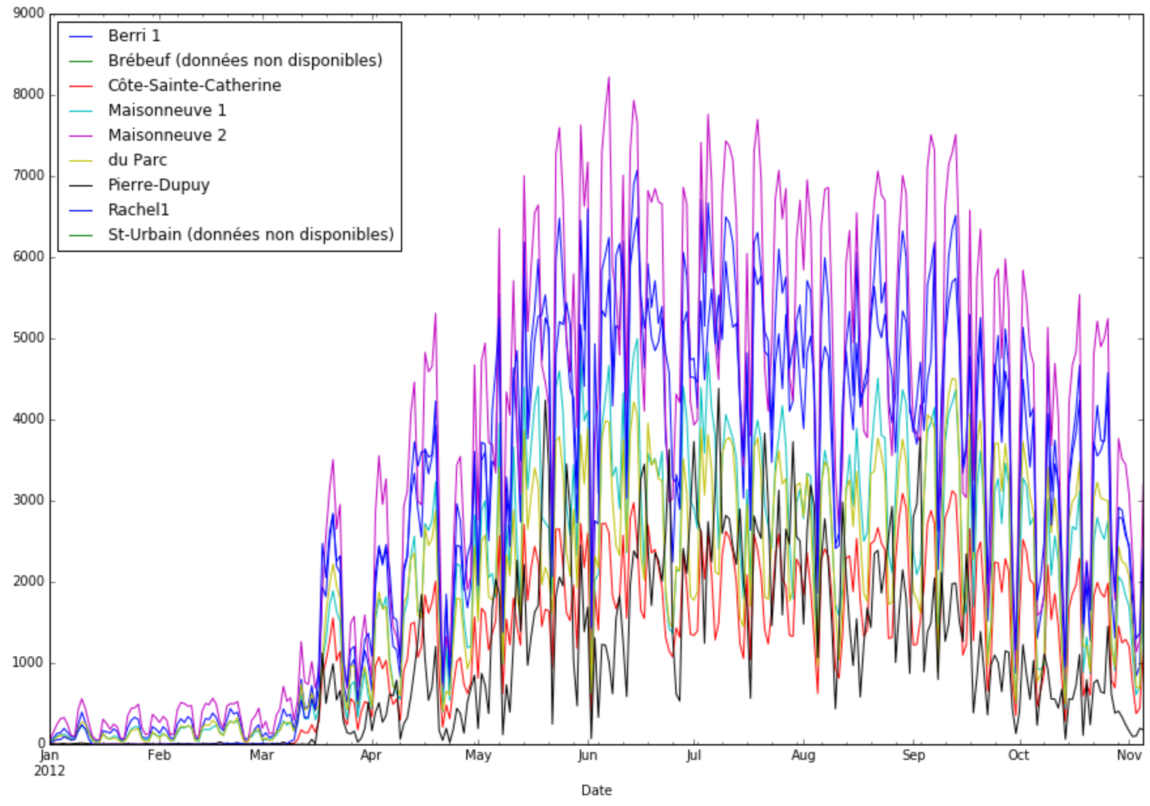
Plotting all columns

In [7]:

```
fixed_df.plot(figsize=(15, 10))
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19c337bda0>
```



Groupby and aggregate

In [8]:

```
berri_bikes = fixed_df[['Berri 1']].copy()
berri_bikes[:5]
```

Out[8]:

	Berri 1
Date	
2012-01-01	35
2012-01-02	83
2012-01-03	135
2012-01-04	144
2012-01-05	197

In [9]:

```
berri_bikes.index
```

Out[9]:

```
DatetimeIndex(['2012-01-01', '2012-01-02', '2012-01-03', '2012-01-0
4',
               '2012-01-05', '2012-01-06', '2012-01-07', '2012-01-0
8',
               '2012-01-09', '2012-01-10',
               ...
               '2012-10-27', '2012-10-28', '2012-10-29', '2012-10-3
0',
               '2012-10-31', '2012-11-01', '2012-11-02', '2012-11-0
3',
               '2012-11-04', '2012-11-05'],
              dtype='datetime64[ns]', name='Date', length=310, freq=
None)
```

In [10]:

```
berri_bikes.index.day
```

Out[10]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 1
6, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,  1,
 2,  3,
        4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
9, 20,
       21, 22, 23, 24, 25, 26, 27, 28, 29,  1,  2,  3,  4,  5,  6,
 7,  8,
        9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25,
       26, 27, 28, 29, 30, 31,  1,  2,  3,  4,  5,  6,  7,  8,  9, 1
0, 11,
       12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 2
7, 28,
       29, 30,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 1
4, 15,
       16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
1,  1,
        2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 1
7, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,  1,  2,  3,
 4,  5,
        6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 22,
       23, 24, 25, 26, 27, 28, 29, 30, 31,  1,  2,  3,  4,  5,  6,
 7,  8,
        9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25,
       26, 27, 28, 29, 30, 31,  1,  2,  3,  4,  5,  6,  7,  8,  9, 1
0, 11,
       12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 2
7, 28,
       29, 30,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 1
4, 15,
       16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
1,  1,
        2,  3,  4,  5], dtype=int32)
```

Adding a weekday column to the dataframe

In [11]:

```
berri_bikes.index.weekday # 0 - Monday
```

Out[11]:

```
array([6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4,
 5, 6, 0,
       1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6,
 0, 1, 2,
       3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1,
 2, 3, 4,
       5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3,
 4, 5, 6,
       0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5,
 6, 0, 1,
       2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0,
 1, 2, 3,
       4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2,
 3, 4, 5,
       6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4,
 5, 6, 0,
       1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6,
 0, 1, 2,
       3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1,
 2, 3, 4,
       5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3,
 4, 5, 6,
       0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5,
 6, 0, 1,
       2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0,
 1, 2, 3,
       4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0], dtype=int32)
```

In [12]:

```
berri_bikes.loc[:,'weekday'] = berri_bikes.index.weekday
berri_bikes[:5]
```

Out[12]:

	Berri 1	weekday
Date		
2012-01-01	35	6
2012-01-02	83	0
2012-01-03	135	1
2012-01-04	144	2
2012-01-05	197	3

Adding up the cyclists by weekday

In [13]:

```
weekday_counts = berri_bikes.groupby('weekday').aggregate(sum)
weekday_counts
```

Out[13]:

	Berri 1
weekday	
0	134298
1	135305
2	152972
3	160131
4	141771
5	101578
6	99310

In [14]:

```
weekday_counts.index = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 
'Saturday', 'Sunday']
weekday_counts
```

Out[14]:

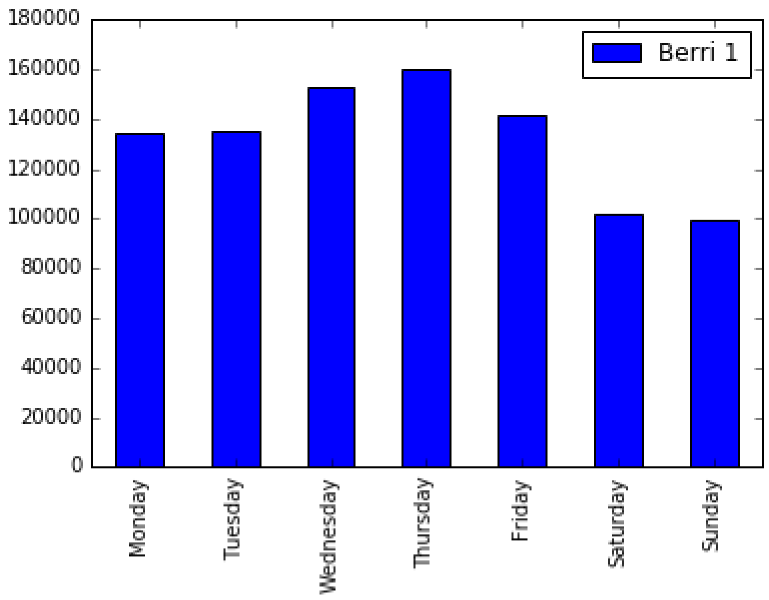
	Berri 1
Monday	134298
Tuesday	135305
Wednesday	152972
Thursday	160131
Friday	141771
Saturday	101578
Sunday	99310

In [15]:

```
weekday_counts.plot(kind='bar')
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19c3385cc0>
```



It looks like Montrealers are commuter cyclists - they bike much more during the week!

Searching for information

We're going to use a larger dataset - a subset of 311 service requests from New York City:

In [16]:

```
complaints = pd.read_csv('data/311-service-requests.csv')
```

```
/usr/local/lib/python3.5/dist-packages/IPython/core/interactiveshel
l.py:2698: DtypeWarning: Columns (8) have mixed types. Specify dtype
option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [17]:

```
complaints = pd.read_csv('data/311-service-requests.csv',low_memory=False)
```

In [18]:

```
complaints
```

Out[18]:

	Unique Key	**Created Date**	**Closed Date**	**Agency**	**Agency Name**	**Complaint Type**	**De**
0	26589651	10/31/2013 02:08:41 AM	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
1	26593698	10/31/2013 02:01:04 AM	NaN	NYPD	New York City Police Department	Illegal Parking	Co Ov
2	26594139	10/31/2013 02:00:24 AM	10/31/2013 02:40:32 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
3	26595721	10/31/2013 01:56:23 AM	10/31/2013 02:21:48 AM	NYPD	New York City Police Department	Noise - Vehicle	Ca
4	26590930	10/31/2013 01:53:44 AM	NaN	DOHMH	Department of Health and Mental Hygiene	Rodent	Co Att Ro
5	26592370	10/31/2013 01:46:52 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Ba
6	26595682	10/31/2013 01:46:40 AM	NaN	NYPD	New York City Police Department	Blocked Driveway	No
7	26595195	10/31/2013 01:44:19 AM	10/31/2013 01:58:49 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
8	26590540	10/31/2013 01:44:14 AM	10/31/2013 02:28:04 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
9	26594392	10/31/2013 01:34:41 AM	10/31/2013 02:23:51 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
10	26595176	10/31/2013 01:25:12 AM	NaN	NYPD	New York City Police Department	Noise - House of Worship	Lo
11	26591982	10/31/2013 01:24:14 AM	10/31/2013 01:54:39 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
12	26594169	10/31/2013 01:20:57 AM	10/31/2013 02:12:31 AM	NYPD	New York City Police Department	Illegal Parking	Do Blo

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	De
13	26594391	10/31/2013 01:20:13 AM	NaN	NYPD	New York City Police Department	Noise - Vehicle	En
14	26590917	10/31/2013 01:19:54 AM	NaN	DOHMH	Department of Health and Mental Hygiene	Rodent	Ra
15	26591458	10/31/2013 01:14:02 AM	10/31/2013 01:30:34 AM	NYPD	New York City Police Department	Noise - House of Worship	Lo
16	26594086	10/31/2013 12:54:03 AM	10/31/2013 02:16:39 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
17	26595117	10/31/2013 12:52:46 AM	NaN	NYPD	New York City Police Department	Illegal Parking	Po Sig
18	26590389	10/31/2013 12:51:00 AM	NaN	DOT	Department of Transportation	Street Light Condition	Str
19	26594210	10/31/2013 12:46:27 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Lo
20	26592932	10/31/2013 12:43:47 AM	10/31/2013 12:56:20 AM	NYPD	New York City Police Department	Noise - House of Worship	Lo
21	26594152	10/31/2013 12:41:17 AM	10/31/2013 01:04:37 AM	NYPD	New York City Police Department	Noise - Commercial	Ba
22	26589678	10/31/2013 12:39:55 AM	NaN	NYPD	New York City Police Department	Noise - Vehicle	Ca
23	26592304	10/31/2013 12:38:00 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Lo
24	26591892	10/31/2013 12:37:16 AM	NaN	NYPD	New York City Police Department	Blocked Driveway	Pa
25	26591573	10/31/2013 12:35:18 AM	10/31/2013 02:41:35 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	De
26	26590509	10/31/2013 12:33:00 AM	NaN	DOT	Department of Transportation	Street Light Condition	Str
27	26591379	10/31/2013 12:32:44 AM	NaN	DOHMH	Department of Health and Mental Hygiene	Harboring Bees/Wasps	Be a b
28	26594085	10/31/2013 12:32:08 AM	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
29	26589201	10/31/2013 12:32:00 AM	NaN	DOT	Department of Transportation	Street Light Condition	Str
...	...	...	...	...	...	...	...
111039	26428764	10/04/2013 12:17:03 AM	10/04/2013 12:38:37 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
111040	26426166	10/04/2013 12:16:22 AM	10/04/2013 05:50:49 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
111041	26438565	10/04/2013 12:16:00 AM	NaN	DEP	Department of Environmental Protection	Noise	No Co Be Ho
111042	26428990	10/04/2013 12:15:52 AM	10/04/2013 12:44:52 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
111043	26432659	10/04/2013 12:15:46 AM	10/04/2013 04:18:45 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
111044	26426096	10/04/2013 12:14:09 AM	10/04/2013 01:03:46 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
111045	26437764	10/04/2013 12:14:00 AM	10/04/2013 12:14:00 AM	DEP	Department of Environmental Protection	Water System	Dir
111046	26436286	10/04/2013 12:14:00 AM	NaN	DEP	Department of Environmental Protection	Noise	No Co Be Ho

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	De
111047	26428989	10/04/2013 12:13:08 AM	10/04/2013 02:12:47 AM	NYPD	New York City Police Department	Illegal Parking	Po: Sig
111048	26430030	10/04/2013 12:12:07 AM	10/04/2013 02:45:24 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lou
111049	26429663	10/04/2013 12:12:07 AM	10/04/2013 01:03:44 AM	NYPD	New York City Police Department	Noise - Commercial	Lou
111050	26437763	10/04/2013 12:11:00 AM	NaN	DEP	Department of Environmental Protection	Noise	No Co Bet Ho
111051	26432955	10/04/2013 12:08:15 AM	10/04/2013 12:48:02 AM	NYPD	New York City Police Department	Noise - Commercial	Lou
111052	26437035	10/04/2013 12:08:00 AM	10/04/2013 12:13:00 AM	DEP	Department of Environmental Protection	Water System	Dir
111053	26433197	10/04/2013 12:08:00 AM	10/04/2013 12:00:00 PM	DSNY	BCC - Queens East	Derelict Vehicles	14 Vel
111054	26426060	10/04/2013 12:06:39 AM	10/04/2013 12:31:16 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lou
111055	26430628	10/04/2013 12:06:28 AM	10/04/2013 12:21:39 AM	NYPD	New York City Police Department	Noise - Commercial	Lou
111056	26431648	10/04/2013 12:06:26 AM	10/23/2013 08:14:52 AM	DOT	Department of Transportation	Street Sign - Missing	Bu:
111057	26437034	10/04/2013 12:06:00 AM	NaN	DEP	Department of Environmental Protection	Noise	No Ha
111058	26426094	10/04/2013 12:05:12 AM	10/04/2013 01:08:29 AM	NYPD	New York City Police Department	Noise - Commercial	Lou
111059	26429040	10/04/2013 12:04:52 AM	10/04/2013 03:01:04 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lou

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	De
111060	26434084	10/04/2013 12:04:00 AM	NaN	DEP	Department of Environmental Protection	Noise	No Co Be Ho
111061	26426164	10/04/2013 12:03:00 AM	10/04/2013 02:14:57 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
111062	26439710	10/04/2013 12:03:00 AM	10/04/2013 12:03:00 AM	DEP	Department of Environmental Protection	Water System	Dir
111063	26435569	10/04/2013 12:02:00 AM	10/04/2013 01:10:00 AM	DEP	Department of Environmental Protection	Water System	Dir
111064	26426013	10/04/2013 12:01:13 AM	10/07/2013 04:07:16 PM	DPR	Department of Parks and Recreation	Maintenance or Facility	Str Ou
111065	26428083	10/04/2013 12:01:05 AM	10/04/2013 02:13:50 AM	NYPD	New York City Police Department	Illegal Parking	Po Sig
111066	26428987	10/04/2013 12:00:45 AM	10/04/2013 01:25:01 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Lo
111067	26426115	10/04/2013 12:00:28 AM	10/04/2013 04:17:32 AM	NYPD	New York City Police Department	Noise - Commercial	Lo
111068	26428033	10/04/2013 12:00:10 AM	10/04/2013 01:20:52 AM	NYPD	New York City Police Department	Blocked Driveway	Pa

111069 rows × 52 columns

Selecting columns and rows

In [19]:

```
complaints['Complaint Type']
```

Out[19]:

```
0          Noise - Street/Sidewalk
1                  Illegal Parking
2               Noise - Commercial
3                  Noise - Vehicle
4                           Rodent
5               Noise - Commercial
6                 Blocked Driveway
7               Noise - Commercial
8               Noise - Commercial
9               Noise - Commercial
10        Noise - House of Worship
11              Noise - Commercial
12                 Illegal Parking
13                 Noise - Vehicle
14                          Rodent
15        Noise - House of Worship
16         Noise - Street/Sidewalk
17                 Illegal Parking
18          Street Light Condition
19              Noise - Commercial
20        Noise - House of Worship
21              Noise - Commercial
22                 Noise - Vehicle
23              Noise - Commercial
24                Blocked Driveway
25         Noise - Street/Sidewalk
26          Street Light Condition
27            Harboring Bees/Wasps
28         Noise - Street/Sidewalk
29          Street Light Condition
                    ...
111039          Noise - Commercial
111040          Noise - Commercial
111041                       Noise
111042     Noise - Street/Sidewalk
111043          Noise - Commercial
111044     Noise - Street/Sidewalk
111045                Water System
111046                       Noise
111047             Illegal Parking
111048     Noise - Street/Sidewalk
111049          Noise - Commercial
111050                       Noise
111051          Noise - Commercial
111052                Water System
111053           Derelict Vehicles
111054     Noise - Street/Sidewalk
111055          Noise - Commercial
111056       Street Sign - Missing
111057                       Noise
111058          Noise - Commercial
111059     Noise - Street/Sidewalk
111060                       Noise
111061          Noise - Commercial
111062                Water System
111063                Water System
111064     Maintenance or Facility
111065             Illegal Parking
111066     Noise - Street/Sidewalk
111067          Noise - Commercial
```

```
111068            Blocked Driveway
Name: Complaint Type, dtype: object
```

In [20]:

```
complaints[:5]
```

Out[20]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor
0	26589651	10/31/2013 02:08:41 AM	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking
1	26593698	10/31/2013 02:01:04 AM	NaN	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking
2	26594139	10/31/2013 02:00:24 AM	10/31/2013 02:40:32 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party
3	26595721	10/31/2013 01:56:23 AM	10/31/2013 02:21:48 AM	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Horn
4	26590930	10/31/2013 01:53:44 AM	NaN	DOHMH	Department of Health and Mental Hygiene	Rodent	Condition Attracting Rodents

5 rows × 52 columns

In [21]:

```
complaints['Complaint Type'][:5]
```

Out[21]:

```
0    Noise - Street/Sidewalk
1            Illegal Parking
2         Noise - Commercial
3            Noise - Vehicle
4                     Rodent
Name: Complaint Type, dtype: object
```

Selecting multiple columns

In [22]:

```
complaints[['Complaint Type', 'Borough']][:5]
```

Out[22]:

	Complaint Type	Borough
0	Noise - Street/Sidewalk	QUEENS
1	Illegal Parking	QUEENS
2	Noise - Commercial	MANHATTAN
3	Noise - Vehicle	MANHATTAN
4	Rodent	MANHATTAN

What is the most common complaint type?

In [23]:

```
complaints['Complaint Type'].value_counts()
```

Out[23]:

```
HEATING                                   14200
GENERAL CONSTRUCTION                       7471
Street Light Condition                     7117
DOF Literature Request                     5797
PLUMBING                                   5373
PAINT - PLASTER                            5149
Blocked Driveway                           4590
NONCONST                                   3998
Street Condition                           3473
Illegal Parking                            3343
Noise                                      3321
Traffic Signal Condition                   3145
Dirty Conditions                           2653
Water System                               2636
Noise - Commercial                         2578
ELECTRIC                                   2350
Broken Muni Meter                          2070
Noise - Street/Sidewalk                    1928
Sanitation Condition                       1824
Rodent                                     1632
Sewer                                      1627
Taxi Complaint                             1227
Consumer Complaint                         1227
Damaged Tree                               1180
Overgrown Tree/Branches                    1083
Graffiti                                    973
Missed Collection (All Materials)           973
Building/Use                                942
Root/Sewer/Sidewalk Condition               836
Derelict Vehicle                            803
                                          ...
Posting Advertisement                         5
Miscellaneous Categories                      5
Fire Alarm - Modification                     5
Poison Ivy                                    5
Internal Code                                 5
Transportation Provider Complaint             4
Special Natural Area District (SNAD)          4
Ferry Complaint                               4
Illegal Animal Sold                           4
Fire Alarm - Replacement                      3
Invitation                                    3
Illegal Fireworks                             3
Adopt-A-Basket                                3
Window Guard                                  2
Legal Services Provider Complaint             2
Opinion for the Mayor                         2
Public Assembly                               2
Misc. Comments                                2
DFTA Literature Request                       2
Highway Sign - Damaged                        1
X-Ray Machine/Equipment                       1
Municipal Parking Facility                    1
DHS Income Savings Requirement                1
Tunnel Condition                              1
Open Flame Permit                             1
Snow                                          1
Trans Fat                                     1
Ferry Permit                                  1
DWD                                           1
```

```
Stalled Sites                                 1
Name: Complaint Type, dtype: int64
```

Now, the top 10 most common complaints:

In [24]:

```
complaint_counts = complaints['Complaint Type'].value_counts()
complaint_counts[:10]
```

Out[24]:

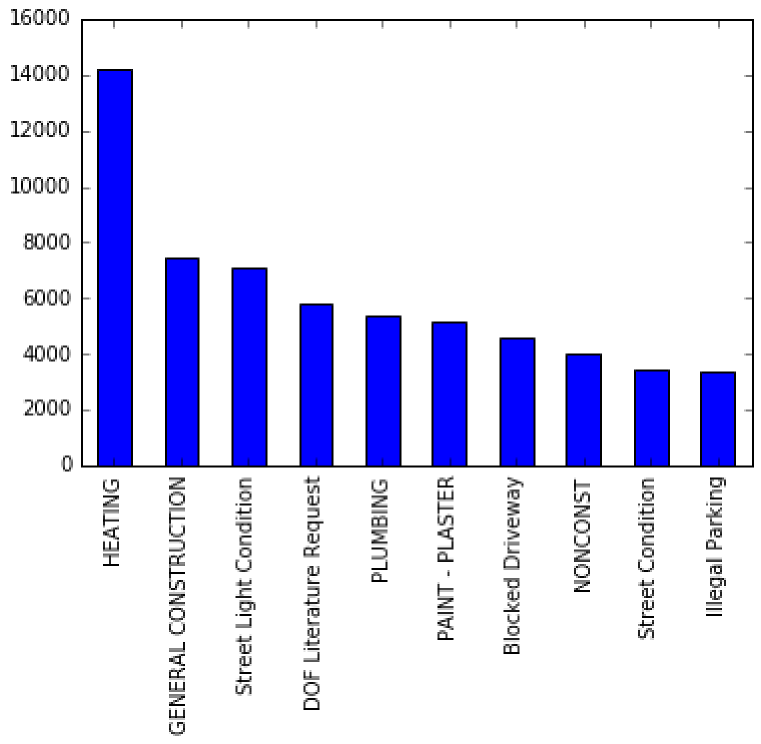
```
HEATING                   14200
GENERAL CONSTRUCTION       7471
Street Light Condition     7117
DOF Literature Request     5797
PLUMBING                   5373
PAINT - PLASTER            5149
Blocked Driveway           4590
NONCONST                   3998
Street Condition           3473
Illegal Parking            3343
Name: Complaint Type, dtype: int64
```

In [25]:

```
complaint_counts[:10].plot(kind='bar')
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bf94af98>
```



Selecting only noise complaints

In [26]:

```
noise_complaints = complaints[complaints['Complaint Type'] == "Noise - Street/Si
dewalk"]
noise_complaints[:3]
```

Out[26]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor
0	26589651	10/31/2013 02:08:41 AM	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking
16	26594086	10/31/2013 12:54:03 AM	10/31/2013 02:16:39 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Part
25	26591573	10/31/2013 12:35:18 AM	10/31/2013 02:41:35 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking

3 rows × 52 columns

Why does it work?

In [27]:

```
complaints['Complaint Type'] == "Noise - Street/Sidewalk"
```

```
Out[27]:

0          True
1         False
2         False
3         False
4         False
5         False
6         False
7         False
8         False
9         False
10        False
11        False
12        False
13        False
14        False
15        False
16         True
17        False
18        False
19        False
20        False
21        False
22        False
23        False
24        False
25         True
26        False
27        False
28         True
29        False
          ...
111039    False
111040    False
111041    False
111042     True
111043    False
111044     True
111045    False
111046    False
111047    False
111048     True
111049    False
111050    False
111051    False
111052    False
111053    False
111054     True
111055    False
111056    False
111057    False
111058    False
111059     True
111060    False
111061    False
111062    False
111063    False
111064    False
111065    False
111066     True
111067    False
```

```
111067    False
111068    False
Name: Complaint Type, dtype: bool
```

- when we index our dataframe with this array, we get just the rows where our boolean array evaluated to True
- important - for row filtering by a boolean array the length of our dataframe's index must be the same length as the boolean array used for filtering

More than one condition may be combined with the & operator:

In [28]:

```
is_noise = complaints['Complaint Type'] == "Noise - Street/Sidewalk"
in_brooklyn = complaints['Borough'] == "BROOKLYN"
complaints[is_noise & in_brooklyn][:5]
```

Out[28]:

	Unique Key	**Created Date**	**Closed Date**	**Agency**	**Agency Name**	**Complaint Type**	**Descripto**
31	26595564	10/31/2013 12:30:36 AM	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Pa
49	26595553	10/31/2013 12:05:10 AM	10/31/2013 02:43:43 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking
109	26594653	10/30/2013 11:26:32 PM	10/31/2013 12:18:54 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Pa
236	26591992	10/30/2013 10:02:58 PM	10/30/2013 10:23:20 PM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking
370	26594167	10/30/2013 08:38:25 PM	10/30/2013 10:26:28 PM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Pa

5 rows × 52 columns

In [29]:

```
complaints[is_noise & in_brooklyn][['Complaint Type', 'Borough', 'Created Date',
 'Descriptor']][:10]
```

Out[29]:

	Complaint Type	Borough	Created Date	Descriptor
31	Noise - Street/Sidewalk	BROOKLYN	10/31/2013 12:30:36 AM	Loud Music/Party
49	Noise - Street/Sidewalk	BROOKLYN	10/31/2013 12:05:10 AM	Loud Talking
109	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 11:26:32 PM	Loud Music/Party
236	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 10:02:58 PM	Loud Talking
370	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 08:38:25 PM	Loud Music/Party
378	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 08:32:13 PM	Loud Talking
656	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 06:07:39 PM	Loud Music/Party
1251	Noise - Street/Sidewalk	BROOKLYN	10/30/2013 03:04:51 PM	Loud Talking
5416	Noise - Street/Sidewalk	BROOKLYN	10/29/2013 10:07:02 PM	Loud Talking
5584	Noise - Street/Sidewalk	BROOKLYN	10/29/2013 08:15:59 PM	Loud Music/Party

Borough with the most noise complaints

In [30]:

```
is_noise = complaints['Complaint Type'] == "Noise - Street/Sidewalk"
noise_complaints = complaints[is_noise]
noise_complaints['Borough'].value_counts()
```

Out[30]:

```
MANHATTAN        917
BROOKLYN         456
BRONX            292
QUEENS           226
STATEN ISLAND     36
Unspecified        1
Name: Borough, dtype: int64
```

We want to normalize the above results by the total number of complaints in a given borough:

In [31]:

```
noise_complaint_counts = noise_complaints['Borough'].value_counts()
complaint_counts = complaints['Borough'].value_counts()
noise_complaint_counts / complaint_counts
```

Out[31]:

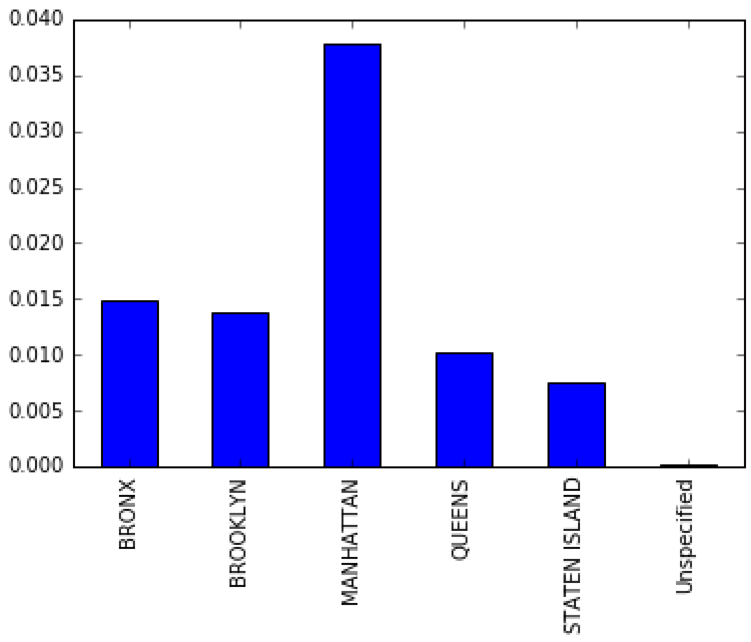
```
BRONX            0.014833
BROOKLYN         0.013864
MANHATTAN        0.037755
QUEENS           0.010143
STATEN ISLAND    0.007474
Unspecified      0.000141
Name: Borough, dtype: float64
```

In [32]:

```
(noise_complaint_counts / complaint_counts).plot(kind='bar')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bf94c8d0>
```



So Manhattan really does complain more about noise than the other boroughs.

Cleaning messy data

In [33]:

```
complaints['Incident Zip'].unique()
```

Out[33]:

```
array(['11432', '11378', '10032', '10023', '10027', '11372', '1141
9',
       '11417', '10011', '11225', '11218', '10003', '10029', '1046
6',
       '11219', '10025', '10310', '11236', nan, '10033', '11216', '1
0016',
       '10305', '10312', '10026', '10309', '10036', '11433', '1123
5',
       '11213', '11379', '11101', '10014', '11231', '11234', '1045
7',
       '10459', '10465', '11207', '10002', '10034', '11233', '1045
3',
       '10456', '10469', '11374', '11221', '11421', '11215', '1000
7',
       '10019', '11205', '11418', '11369', '11249', '10005', '1000
9',
       '11211', '11412', '10458', '11229', '10065', '10030', '1122
2',
       '10024', '10013', '11420', '11365', '10012', '11214', '1121
2',
       '10022', '11232', '11040', '11226', '10281', '11102', '1120
8',
       '10001', '10472', '11414', '11223', '10040', '11220', '1137
3',
       '11203', '11691', '11356', '10017', '10452', '10280', '1121
7',
       '10031', '11201', '11358', '10128', '11423', '10039', '1001
0',
       '11209', '10021', '10037', '11413', '11375', '11238', '1047
3',
       '11103', '11354', '11361', '11106', '11385', '10463', '1046
7',
       '11204', '11237', '11377', '11364', '11434', '11435', '1121
0',
       '11228', '11368', '11694', '10464', '11415', '10314', '1030
1',
       '10018', '10038', '11105', '11230', '10468', '11104', '1047
1',
       '11416', '10075', '11422', '11355', '10028', '10462', '1030
6',
       '10461', '11224', '11429', '10035', '11366', '11362', '1120
6',
       '10460', '10304', '11360', '11411', '10455', '10475', '1006
9',
       '10303', '10308', '10302', '11357', '10470', '11367', '1137
0',
       '10454', '10451', '11436', '11426', '10153', '11004', '1142
8',
       '11427', '11001', '11363', '10004', '10474', '11430', '1000
0',
       '10307', '11239', '10119', '10006', '10048', '11697', '1169
2',
       '11693', '10573', '00083', '11559', '10020', '77056', '1177
6',
       '70711', '10282', '11109', '10044', '02061', '77092-2016', '1
4225',
       '55164-0737', '19711', '07306', '000000', 'NO CLUE', '90010',
       '11747', '23541', '11788', '07604', '10112', '11563', '1158
0',
       '07087', '11042', '07093', '11501', '92123', '00000', '1157
       '07087', '11042', '07093', '11501', '92123', '00000', '1157
5',
       '07109', '11797', '10803', '11716', '11722', '11549-3650', '1
0162',
       '23502', '11518', '07020', '08807', '11577', '07114', '1100
3',
       '07201', '61702', '10103', '29616-0759', '35209-3114', '1152
0',
       '11735', '10129', '11005', '41042', '11590', '06901', '0720
8',
       '11530', '13221', '10954', '11111', '10107'], dtype=object)
```

Some of the problems:

- 'NO CLUE', 'N/A/', 29616-0759, 83 and NaN values
- some codes have been parsed as strings, some as floats

What we can do:

- normalize 'NO CLUE' and 'N/A' into regular NaN values
- make everything strings
- look at the 83 and other strange codes, and decide what to do

Fixing NaNs and string/float confusion

In [34]:

```
na_values = ['NO CLUE', 'N/A', '0']
complaints = pd.read_csv('data/311-service-requests.csv', na_values=na_values, d
type={'Incident Zip': str})
```

In [35]:

```
complaints['Incident Zip'].unique()
```

Out[35]:

```
array(['11432', '11378', '10032', '10023', '10027', '11372', '1141
9',
       '11417', '10011', '11225', '11218', '10003', '10029', '1046
6',
       '11219', '10025', '10310', '11236', nan, '10033', '11216', '1
0016',
       '10305', '10312', '10026', '10309', '10036', '11433', '1123
5',
       '11213', '11379', '11101', '10014', '11231', '11234', '1045
7',
       '10459', '10465', '11207', '10002', '10034', '11233', '1045
3',
       '10456', '10469', '11374', '11221', '11421', '11215', '1000
7',
       '10019', '11205', '11418', '11369', '11249', '10005', '1000
9',
       '11211', '11412', '10458', '11229', '10065', '10030', '1122
2',
       '10024', '10013', '11420', '11365', '10012', '11214', '1121
2',
       '10022', '11232', '11040', '11226', '10281', '11102', '1120
8',
       '10001', '10472', '11414', '11223', '10040', '11220', '1137
3',
       '11203', '11691', '11356', '10017', '10452', '10280', '1121
7',
       '10031', '11201', '11358', '10128', '11423', '10039', '1001
0',
       '11209', '10021', '10037', '11413', '11375', '11238', '1047
3',
       '11103', '11354', '11361', '11106', '11385', '10463', '1046
7',
       '11204', '11237', '11377', '11364', '11434', '11435', '1121
0',
       '11228', '11368', '11694', '10464', '11415', '10314', '1030
1',
       '10018', '10038', '11105', '11230', '10468', '11104', '1047
1',
       '11416', '10075', '11422', '11355', '10028', '10462', '1030
6',
       '10461', '11224', '11429', '10035', '11366', '11362', '1120
6',
       '10460', '10304', '11360', '11411', '10455', '10475', '1006
9',
       '10303', '10308', '10302', '11357', '10470', '11367', '1137
0',
       '10454', '10451', '11436', '11426', '10153', '11004', '1142
8',
       '11427', '11001', '11363', '10004', '10474', '11430', '1000
0',
       '10307', '11239', '10119', '10006', '10048', '11697', '1169
2',
       '11693', '10573', '00083', '11559', '10020', '77056', '1177
6',
       '70711', '10282', '11109', '10044', '02061', '77092-2016', '1
4225',
       '55164-0737', '19711', '07306', '000000', '90010', '11747',
 '23541',
       '11788', '07604', '10112', '11563', '11580', '07087', '1104
2',
       '07093', '11501', '92123', '00000', '11575', '07109', '1179
7',
       '10803', '11716', '11722', '11549-3650', '10162', '23502', '1
1518',
       '07020', '08807', '11577', '07114', '11003', '07201', '6170
2',
       '10103', '29616-0759', '35209-3114', '11520', '11735', '1012
9',
       '11005', '41042', '11590', '06901', '07208', '11530', '1322
1',
       '10954', '11111', '10107'], dtype=object)
```

Zip codes with the dashes

In [36]:

```
rows_with_dashes = complaints['Incident Zip'].str.contains('-').fillna(False)
len(complaints[rows_with_dashes])
```

Out[36]:

```
5
```

In [37]:

```
complaints[rows_with_dashes]
```

Out[37]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor
29136	26550551	10/24/2013 06:16:34 PM	NaN	DCA	Department of Consumer Affairs	Consumer Complaint	False Advertising
30939	26548831	10/24/2013 09:35:10 AM	NaN	DCA	Department of Consumer Affairs	Consumer Complaint	Harassmen
70539	26488417	10/15/2013 03:40:33 PM	NaN	TLC	Taxi and Limousine Commission	Taxi Complaint	Driver Complaint
85821	26468296	10/10/2013 12:36:43 PM	10/26/2013 01:07:07 AM	DCA	Department of Consumer Affairs	Consumer Complaint	Debt Not Owed
89304	26461137	10/09/2013 05:23:46 PM	10/25/2013 01:06:41 AM	DCA	Department of Consumer Affairs	Consumer Complaint	Harassmen

5 rows × 52 columns

In [38]:

```
long_zip_codes = complaints['Incident Zip'].str.len() > 5
complaints['Incident Zip'][long_zip_codes].unique()
```

Out[38]:

```
array(['77092-2016', '55164-0737', '000000', '11549-3650', '29616-07
59',
       '35209-3114'], dtype=object)
```

As far as the long zip codes are concerned, they are normal for the USA, but may be truncated to 5 digits:

In [39]:

```
complaints['Incident Zip'] = complaints['Incident Zip'].str.slice(0, 5)
```

83 zip code

Central Park's zip code is 00083 - this code is correct!

00000 zip code

In [40]:

```
complaints[complaints['Incident Zip'] == '00000']
```

Out[40]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Lo Ty
42600	26529313	10/22/2013 02:51:06 PM	NaN	TLC	Taxi and Limousine Commission	Taxi Complaint	Driver Complaint	Na
60843	26507389	10/17/2013 05:48:44 PM	NaN	TLC	Taxi and Limousine Commission	Taxi Complaint	Driver Complaint	St

2 rows × 52 columns

This entries look bad. So we replace 00000 by NaNs:

In [41]:

```
zero_zips = complaints['Incident Zip'] == '00000'
complaints.loc[zero_zips, 'Incident Zip'] = np.nan
```

In [42]:

```
unique_zips = complaints['Incident Zip'].sort_values().unique()
```

In [43]:

```
unique_zips
```

Out[43]:

```
array(['00083', '02061', '06901', '07020', '07087', '07093', '0710
9',
       '07114', '07201', '07208', '07306', '07604', '08807', '1000
0',
       '10001', '10002', '10003', '10004', '10005', '10006', '1000
7',
       '10009', '10010', '10011', '10012', '10013', '10014', '1001
6',
       '10017', '10018', '10019', '10020', '10021', '10022', '1002
3',
       '10024', '10025', '10026', '10027', '10028', '10029', '1003
0',
       '10031', '10032', '10033', '10034', '10035', '10036', '1003
7',
       '10038', '10039', '10040', '10044', '10048', '10065', '1006
9',
       '10075', '10103', '10107', '10112', '10119', '10128', '1012
9',
       '10153', '10162', '10280', '10281', '10282', '10301', '1030
2',
       '10303', '10304', '10305', '10306', '10307', '10308', '1030
9',
       '10310', '10312', '10314', '10451', '10452', '10453', '1045
4',
       '10455', '10456', '10457', '10458', '10459', '10460', '1046
1',
       '10462', '10463', '10464', '10465', '10466', '10467', '1046
8',
       '10469', '10470', '10471', '10472', '10473', '10474', '1047
5',
       '10573', '10803', '10954', '11001', '11003', '11004', '1100
5',
       '11040', '11042', '11101', '11102', '11103', '11104', '1110
5',
       '11106', '11109', '11111', '11201', '11203', '11204', '1120
5',
       '11206', '11207', '11208', '11209', '11210', '11211', '1121
2',
       '11213', '11214', '11215', '11216', '11217', '11218', '1121
9',
       '11220', '11221', '11222', '11223', '11224', '11225', '1122
6',
       '11228', '11229', '11230', '11231', '11232', '11233', '1123
4',
       '11235', '11236', '11237', '11238', '11239', '11249', '1135
4',
       '11355', '11356', '11357', '11358', '11360', '11361', '1136
2',
       '11363', '11364', '11365', '11366', '11367', '11368', '1136
9',
       '11370', '11372', '11373', '11374', '11375', '11377', '1137
8',
       '11379', '11385', '11411', '11412', '11413', '11414', '1141
5',
       '11416', '11417', '11418', '11419', '11420', '11421', '1142
2',
       '11423', '11426', '11427', '11428', '11429', '11430', '1143
2',
       '11433', '11434', '11435', '11436', '11501', '11518', '1152
0',
0',
       '11530', '11549', '11559', '11563', '11575', '11577', '1158
0',
       '11590', '11691', '11692', '11693', '11694', '11697', '1171
6',
       '11722', '11735', '11747', '11776', '11788', '11797', '1322
1',
       '14225', '19711', '23502', '23541', '29616', '35209', '4104
2',
       '55164', '61702', '70711', '77056', '77092', '90010', '9212
3', nan], dtype=object)
```

Are all complaints from NY?

For the sake of simplicity we can assume that the zips starting with '0' and '1' are okay (there are some exceptions though):

In [44]:

```
zips = complaints['Incident Zip']
is_close = zips.str.startswith('0') | zips.str.startswith('1')
is_far = ~(is_close) & zips.notnull()
```

In [45]:

```
zips[is_far]
```

Out[45]:

```
12102    77056
13450    70711
29136    77092
30939    55164
44008    90010
47048    23541
57636    92123
71001    92123
71834    23502
80573    61702
85821    29616
89304    35209
94201    41042
Name: Incident Zip, dtype: object
```

In [46]:

```
complaints[is_far][['Incident Zip', 'Descriptor', 'City']].sort_values('Incident
 Zip')
```

Out[46]:

	Incident Zip	Descriptor	City
71834	23502	Harassment	NORFOLK
47048	23541	Harassment	NORFOLK
85821	29616	Debt Not Owed	GREENVILLE
89304	35209	Harassment	BIRMINGHAM
94201	41042	Harassment	FLORENCE
30939	55164	Harassment	ST. PAUL
80573	61702	Billing Dispute	BLOOMIGTON
13450	70711	Contract Dispute	CLIFTON
12102	77056	Debt Not Owed	HOUSTON
29136	77092	False Advertising	HOUSTON
44008	90010	Billing Dispute	LOS ANGELES
57636	92123	Harassment	SAN DIEGO
71001	92123	Billing Dispute	SAN DIEGO

There are really requests coming from LA and Houston!

String operations

In [47]:

```
weather_2012 = pd.read_csv('data/weather_2012.csv', parse_dates=True,
index_col='Date/Time')
weather_2012[:5]
```

Out[47]:

	Temp (C)	Dew Point Temp (C)	Rel Hum (%)	Wind Spd (km/h)	Visibility (km)	Stn Press (kPa)	Weather
Date/Time							
2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2012-01-01 02:00:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
2012-01-01 03:00:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
2012-01-01 04:00:00	-1.5	-3.3	88	7	4.8	101.23	Fog

Looking for text

In [48]:

```
weather_description = weather_2012['Weather']
is_snowing = weather_description.str.contains('Snow')
is_snowing[:5] #this is not very useful
```

Out[48]:

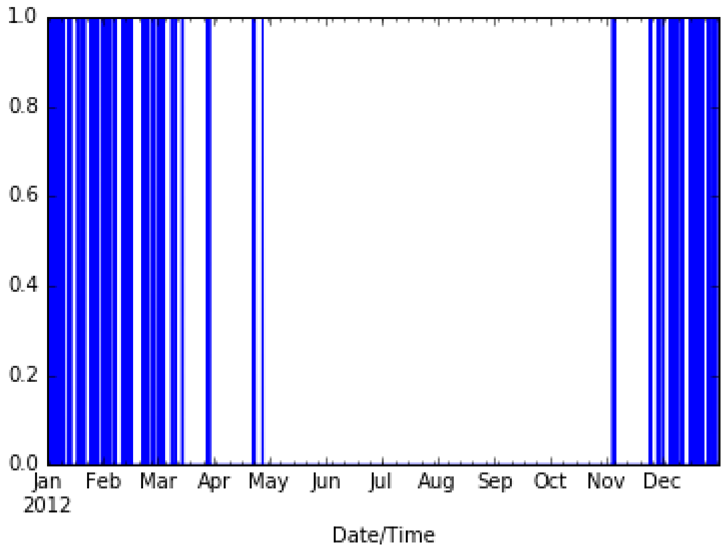
```
Date/Time
2012-01-01 00:00:00    False
2012-01-01 01:00:00    False
2012-01-01 02:00:00    False
2012-01-01 03:00:00    False
2012-01-01 04:00:00    False
Name: Weather, dtype: bool
```

In [49]:

```
is_snowing.plot() #more useful
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bc501eb8>
```



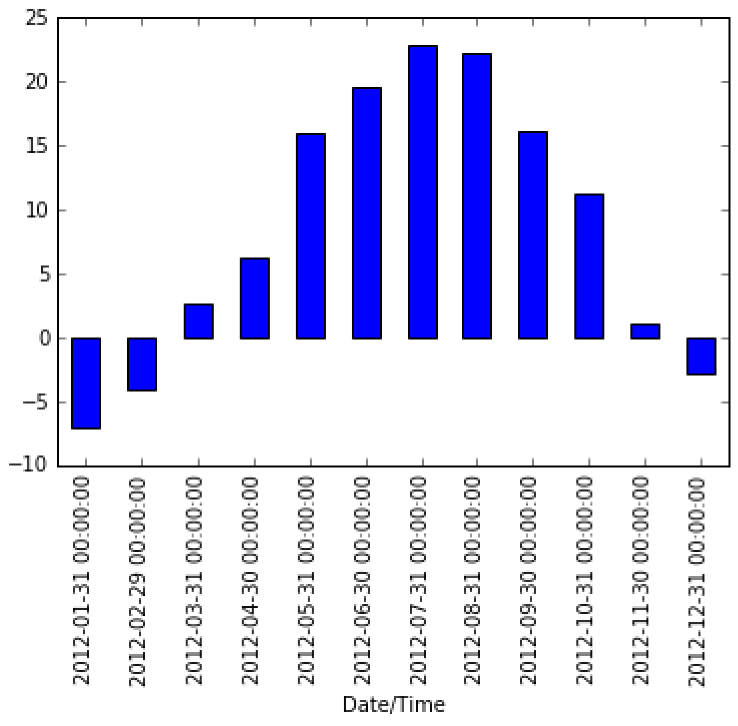
Median temperature in each month

In [50]:

```
weather_2012['Temp (C)'].resample('M').apply(np.median).plot(kind='bar')
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bc2edb70>
```



The snowiest month

We can think of snowiness as being a bunch of 1s and 0s instead of Trues and Falses:

In [51]:

```
is_snowing.astype(float)[:10]
```

Out[51]:

```
Date/Time
2012-01-01 00:00:00    0.0
2012-01-01 01:00:00    0.0
2012-01-01 02:00:00    0.0
2012-01-01 03:00:00    0.0
2012-01-01 04:00:00    0.0
2012-01-01 05:00:00    0.0
2012-01-01 06:00:00    0.0
2012-01-01 07:00:00    0.0
2012-01-01 08:00:00    0.0
2012-01-01 09:00:00    0.0
Name: Weather, dtype: float64
```

Then we use resample to find the percentage of time it was snowing each month:

In [52]:

```
is_snowing.astype(float).resample('M').apply(np.mean)
```

Out[52]:

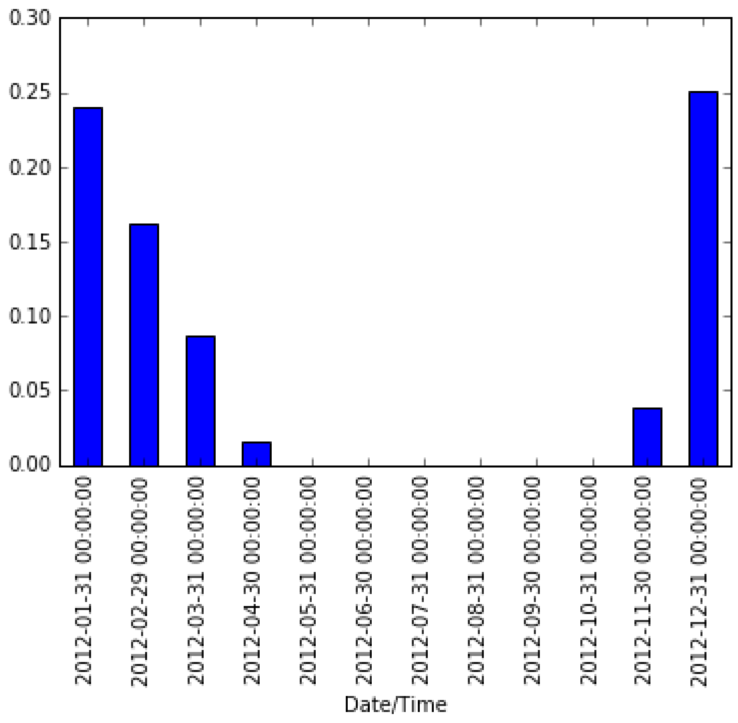
```
Date/Time
2012-01-31    0.240591
2012-02-29    0.162356
2012-03-31    0.087366
2012-04-30    0.015278
2012-05-31    0.000000
2012-06-30    0.000000
2012-07-31    0.000000
2012-08-31    0.000000
2012-09-30    0.000000
2012-10-31    0.000000
2012-11-30    0.038889
2012-12-31    0.251344
Freq: M, Name: Weather, dtype: float64
```

In [53]:

```
is_snowing.astype(float).resample('M').apply(np.mean).plot(kind='bar')
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bc191da0>
```



- in 2012, December was the snowiest month
- it starts snowing pretty abruptly in November, and then tapers off slowly and takes a long time to stop, with the last snow usually being in April or May

Temperature and snowiness - summary

In [54]:

```
temperature = weather_2012['Temp (C)'].resample('M').apply(np.median)
is_snowing = weather_2012['Weather'].str.contains('Snow')
snowiness = is_snowing.astype(float).resample('M').apply(np.mean)

temperature.name = "Temperature"
snowiness.name = "Snowiness"
```

We can use concat to combine the two statistics into a single dataframe:

In [55]:

```
stats = pd.concat([temperature, snowiness], axis=1)
stats
```

Out[55]:

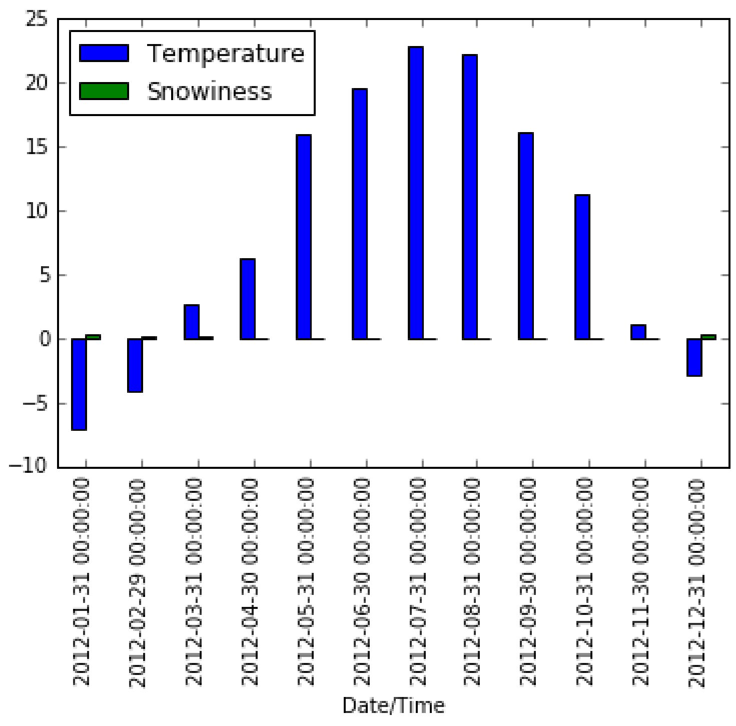
	Temperature	Snowiness
Date/Time		
2012-01-31	-7.05	0.240591
2012-02-29	-4.10	0.162356
2012-03-31	2.60	0.087366
2012-04-30	6.30	0.015278
2012-05-31	16.05	0.000000
2012-06-30	19.60	0.000000
2012-07-31	22.90	0.000000
2012-08-31	22.20	0.000000
2012-09-30	16.10	0.000000
2012-10-31	11.30	0.000000
2012-11-30	1.05	0.038889
2012-12-31	-2.85	0.251344

In [56]:

```
stats.plot(kind='bar') #the scales are wrong
```

Out[56]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19bc12def0>
```

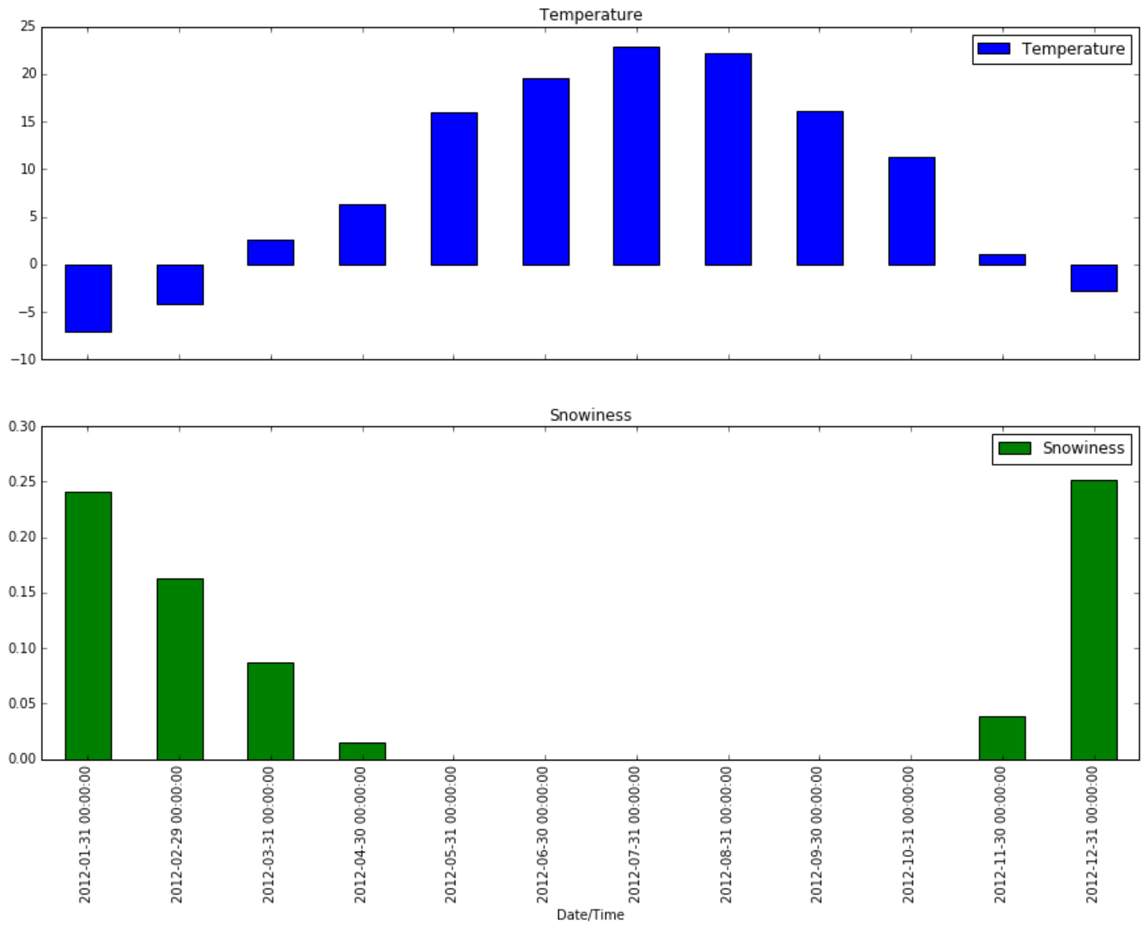


In [57]:

```
stats.plot(kind='bar', subplots=True, figsize=(15, 10))
```

Out[57]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f19bc241b
70>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f19b66d64
38>], dtype=object)
```



Parsing Unix timestamps

In [58]:

```
popcon = pd.read_csv('data/popularity-contest.txt', sep=' ', )[:-1]
popcon.columns = ['atime', 'ctime', 'package-name', 'mru-program', 'tag']
popcon[:5]
```

Out[58]:

	atime	ctime	package-name	mru-program	tag
0	1387295797	1367633260	perl-base	/usr/bin/perl	NaN
1	1387295796	1354370480	login	/bin/su	NaN
2	1387295743	1354341275	libtalloc2	/usr/lib/x86_64-linux-gnu/libtalloc.so.2.0.7	NaN
3	1387295743	1387224204	libwbclient0	/usr/lib/x86_64-linux-gnu/libwbclient.so.0	<RECENT-CTIME>
4	1387295742	1354341253	libselinux1	/lib/x86_64-linux-gnu/libselinux.so.1	NaN

In [59]:

```
popcon['atime'] = popcon['atime'].astype(int)
popcon['ctime'] = popcon['ctime'].astype(int)
```

- internally, numpy datetimes are already stored as Unix timestamps!!!
- all we need to do is to tell pandas that these integers are actually datetimes $\longrightarrow$ it doesn't need to do any conversion at all

In [60]:

```
popcon['atime'] = pd.to_datetime(popcon['atime'], unit='s')
popcon['ctime'] = pd.to_datetime(popcon['ctime'], unit='s')
```

In [61]:

```
popcon[:5]
```

Out[61]:

	atime	ctime	package-name	mru-program	tag
0	2013-12-17 15:56:37	2013-05-04 02:07:40	perl-base	/usr/bin/perl	NaN
1	2013-12-17 15:56:36	2012-12-01 14:01:20	login	/bin/su	NaN
2	2013-12-17 15:55:43	2012-12-01 05:54:35	libtalloc2	/usr/lib/x86_64-linux-gnu/libtalloc.so.2.0.7	NaN
3	2013-12-17 15:55:43	2013-12-16 20:03:24	libwbclient0	/usr/lib/x86_64-linux-gnu/libwbclient.so.0	<RECENT-CTIME>
4	2013-12-17 15:55:42	2012-12-01 05:54:13	libselinux1	/lib/x86_64-linux-gnu/libselinux.so.1	NaN

Suppose we want to look at all packages that aren't libraries.

First, we want to get rid of everything with timestamp 0:

In [62]:

```
popcon = popcon[popcon['atime'] > '1970-01-01']
```

Now, we look at rows not containing the 'lib' string:

In [63]:

```
nonlibraries = popcon[~popcon['package-name'].str.contains('lib')]
nonlibraries.sort_values(by='ctime', ascending=False)[:10]
```

Out[63]:

	atime	ctime	package-name	mru-program	tag
57	2013-12-17 04:55:39	2013-12-17 04:55:42	ddd	/usr/bin/ddd	<RECE CTIME
450	2013-12-16 20:03:20	2013-12-16 20:05:13	nodejs	/usr/bin/npm	<RECE CTIME
454	2013-12-16 20:03:20	2013-12-16 20:05:04	switchboard-plug-keyboard	/usr/lib/plugs/pantheon/keyboard/options.txt	<RECE CTIME
445	2013-12-16 20:03:20	2013-12-16 20:05:04	thunderbird-locale-en	/usr/lib/thunderbird-addons/extensions/langpac...	<RECE CTIME
396	2013-12-16 20:08:27	2013-12-16 20:05:03	software-center	/usr/sbin/update-software-center	<RECE CTIME
449	2013-12-16 20:03:20	2013-12-16 20:05:00	samba-common-bin	/usr/bin/net.samba3	<RECE CTIME
397	2013-12-16 20:08:25	2013-12-16 20:04:59	postgresql-client-9.1	/usr/lib/postgresql/9.1/bin/psql	<RECE CTIME
398	2013-12-16 20:08:23	2013-12-16 20:04:58	postgresql-9.1	/usr/lib/postgresql/9.1/bin/postmaster	<RECE CTIME
452	2013-12-16 20:03:20	2013-12-16 20:04:55	php5-dev	/usr/include/php5/main/snprintf.h	<RECE CTIME
440	2013-12-16 20:03:20	2013-12-16 20:04:54	php-pear	/usr/share/php/XML/Util.php	<RECE CTIME

Loading data from SQL

In [64]:

```
import sqlite3
```

In [65]:

```
con = sqlite3.connect("data/weather_2012.sqlite")
df = pd.read_sql("SELECT * from weather_2012 LIMIT 3", con)
df
```

Out[65]:

	id	date_time	temp
0	1	2012-01-01 00:00:00	-1.8
1	2	2012-01-01 01:00:00	-1.8
2	3	2012-01-01 02:00:00	-1.8

In [66]:

```
df = pd.read_sql("SELECT * from weather_2012 LIMIT 3", con, index_col='id') #ind
exed by 'id' column
df
```

Out[66]:

	date_time	temp
id		
1	2012-01-01 00:00:00	-1.8
2	2012-01-01 01:00:00	-1.8
3	2012-01-01 02:00:00	-1.8

In [67]:

```
df = pd.read_sql("SELECT * from weather_2012 LIMIT 3", con, index_col=['id', 'da
te_time']) #indexed by multiple columns
df
```

Out[67]:

		temp
id	**date_time**	
1	**2012-01-01 00:00:00**	-1.8
2	**2012-01-01 01:00:00**	-1.8
3	**2012-01-01 02:00:00**	-1.8

Pandas vs SQL

In [68]:

```
url = 'https://raw.github.com/pydata/pandas/master/pandas/tests/data/tips.csv'
tips = pd.read_csv(url)
tips.head()
```

Out[68]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

SELECT

```
SELECT total_bill, tip, smoker, time
FROM tips
LIMIT 5;
```

In [69]:

```
tips[['total_bill', 'tip', 'smoker', 'time']].head(5)
```

Out[69]:

	total_bill	tip	smoker	time
0	16.99	1.01	No	Dinner
1	10.34	1.66	No	Dinner
2	21.01	3.50	No	Dinner
3	23.68	3.31	No	Dinner
4	24.59	3.61	No	Dinner

WHERE

```
SELECT *
FROM tips
WHERE time = 'Dinner'
LIMIT 5;
```

In [70]:

```
tips[tips['time'] == 'Dinner'].head(5)
```

Out[70]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
-- tips greater than 5$ at dinner
SELECT *
FROM tips
WHERE time = 'Dinner' AND tip > 5.00;
```

In [71]:

```
tips[(tips['time'] == 'Dinner') & (tips['tip'] > 5.00)]
```

Out[71]:

	total_bill	tip	sex	smoker	day	time	size
23	39.42	7.58	Male	No	Sat	Dinner	4
44	30.40	5.60	Male	No	Sun	Dinner	4
47	32.40	6.00	Male	No	Sun	Dinner	4
52	34.81	5.20	Female	No	Sun	Dinner	4
59	48.27	6.73	Male	No	Sat	Dinner	4
116	29.93	5.07	Male	No	Sun	Dinner	4
155	29.85	5.14	Female	No	Sun	Dinner	5
170	50.81	10.00	Male	Yes	Sat	Dinner	3
172	7.25	5.15	Male	Yes	Sun	Dinner	2
181	23.33	5.65	Male	Yes	Sun	Dinner	2
183	23.17	6.50	Male	Yes	Sun	Dinner	4
211	25.89	5.16	Male	Yes	Sat	Dinner	4
212	48.33	9.00	Male	No	Sat	Dinner	4
214	28.17	6.50	Female	Yes	Sat	Dinner	3
239	29.03	5.92	Male	No	Sat	Dinner	3

```
-- tip at dinner, group size >= 5, total bill > 45
SELECT *
FROM tips
WHERE size >= 5 OR total_bill > 45;
```

In [72]:

```
tips[(tips['size'] >= 5) | (tips['total_bill'] > 45)]
```

Out[72]:

	total_bill	tip	sex	smoker	day	time	size
59	48.27	6.73	Male	No	Sat	Dinner	4
125	29.80	4.20	Female	No	Thur	Lunch	6
141	34.30	6.70	Male	No	Thur	Lunch	6
142	41.19	5.00	Male	No	Thur	Lunch	5
143	27.05	5.00	Female	No	Thur	Lunch	6
155	29.85	5.14	Female	No	Sun	Dinner	5
156	48.17	5.00	Male	No	Sun	Dinner	6
170	50.81	10.00	Male	Yes	Sat	Dinner	3
182	45.35	3.50	Male	Yes	Sun	Dinner	3
185	20.69	5.00	Male	No	Sun	Dinner	5
187	30.46	2.00	Male	Yes	Sun	Dinner	5
212	48.33	9.00	Male	No	Sat	Dinner	4
216	28.15	3.00	Male	Yes	Sat	Dinner	5

GROUPBY

```
SELECT sex, count(*)
FROM tips
GROUP BY sex;
```

In [73]:

```
tips.groupby('sex').size()
```

Out[73]:

```
sex
Female     87
Male      157
dtype: int64
```

In [74]:

```
tips.groupby('sex').count() #non-zero entries in each column
```

Out[74]:

	total_bill	tip	smoker	day	time	size
sex						
Female	87	87	87	87	87	87
Male	157	157	157	157	157	157

In [75]:

```
tips.groupby('sex')['total_bill'].count()
```

Out[75]:

```
sex
Female     87
Male      157
Name: total_bill, dtype: int64
```

```
SELECT day, AVG(tip), COUNT(*)
FROM tips
GROUP BY day;
```

In [76]:

```
tips.groupby('day').agg({'tip': np.mean, 'day': np.size})
```

Out[76]:

	day	tip
day		
Fri	19	2.734737
Sat	87	2.993103
Sun	76	3.255132
Thur	62	2.771452

```
SELECT smoker, day, COUNT(*), AVG(tip)
FROM tips
GROUP BY smoker, day;
```