In [1]:
```
!pip install pandas
import pandas
import os
```

```
Defaulting to user installation because normal site-packages is not writeab
le
Requirement already satisfied: pandas in /home/jupyter-doggo/.local/lib/pyt
hon3.9/site-packages (1.5.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/tljh/user/li
b/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.20.3 in /home/jupyter-doggo/.local/
lib/python3.9/site-packages (from pandas) (1.24.1)
Requirement already satisfied: pytz>=2020.1 in /opt/tljh/user/lib/python3.
9/site-packages (from pandas) (2022.7)
Requirement already satisfied: six>=1.5 in /opt/tljh/user/lib/python3.9/sit
e-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [2]:
```
# Important, make sure to use the correct file
filename = 'IL-94846-2013Fall-2014Spring.csv'
temp_table = pandas.read_csv('NSW_Weather/Orig/'+filename, skiprows = 8, ski
```

```
/tmp/ipykernel_132079/3436166416.py:3: ParserWarning: Falling back to the
'python' engine because the 'c' engine does not support skipfooter; you can
avoid this warning by specifying engine='python'.
  temp_table = pandas.read_csv('NSW_Weather/Orig/'+filename, skiprows = 8,
skipfooter = 15)
```

In [3]:
```
start_year = int(filename[9:13])
end_year = int(filename[18:22])
STATE_ID = filename[0:2]
weather_station = filename[3:8]
```

In [4]:
```
temp_table
```

Out[4]:

	Date	Time	Temp (F)	RH (%)	Dewpt (F)	Wind Spd (mph)	Wind Direction (deg)	Peak Wind Gust(mph)	Low Cloud Ht (ft)	Med Cloud Ht (ft)	High Cloud Ht (ft)
0	2012-10-01	00:51	51	82	46	3	10	m	30000	m	m
1	2012-10-01	01:51	50	89	47	0	0	m	30000	m	m
2	2012-10-01	02:51	50	89	47	0	0	m	1000	30000	m
3	2012-10-01	03:51	48	96	47	0	0	m	2400	30000	m
4	2012-10-01	04:51	51	89	48	5	350	m	30000	m	m
...	...	...	...	...	...	...	...	...	...	...	...
4363	2013-03-31	19:51	41	54	26	16	290	23	15000	25000	m
4364	2013-03-31	20:51	39	61	27	15	290	m	4000	15000	20000
4365	2013-03-31	21:51	39	61	27	15	310	24	3700	6000	15000
4366	2013-03-31	22:51	38	57	24	16	310	26	3700	6000	15000
4367	2013-03-31	23:51	36	56	22	11	320	m	4000	25000	m

4368 rows × 19 columns

Optional: You might have noticed that there is an empty column at the end of the table, since the footer of the weather station data contains extra words at the end. Therefore, it would be the best to drop it, but it is optional.

In [5]:
```
try:
    temp_table = temp_table.drop('Unnamed: 18', axis = 1)
    print('Empty Column at the end is dropped.')
except:
    print('Nothing to be dropped here.')
```

```
Empty Column at the end is dropped.
```

Generate all possible time stamps

In [6]:
```
from datetime import timedelta, date

def daterange(date1, date2):
    for n in range(int ((date2 - date1).days) + 1):
        yield date1 + timedelta(n)
```

In [7]:
```
start_dt = date(start_year, 10, 1)
end_dt = date(end_year, 3, 31)
```

In [8]:
```
# Testing
for dt in daterange(start_dt, end_dt):
    print(dt.strftime("%Y%m%d"))
```

20121001
20121002
20121003
20121004
20121005
20121006
20121007
20121008
20121009
20121010
20121011
20121012
20121013
20121014
20121015
20121016
20121017
20121018
20121019
20121020
20121021
20121022
20121023
20121024
20121025
20121026
20121027
20121028
20121029
20121030
20121031
20121101
20121102
20121103
20121104
20121105
20121106
20121107
20121108
20121109
20121110
20121111
20121112
20121113
20121114
20121115
20121116
20121117
20121118
20121119
20121120
20121121
20121122
20121123
20121124
20121125

20121126
20121127
20121128
20121129
20121130
20121201
20121202
20121203
20121204
20121205
20121206
20121207
20121208
20121209
20121210
20121211
20121212
20121213
20121214
20121215
20121216
20121217
20121218
20121219
20121220
20121221
20121222
20121223
20121224
20121225
20121226
20121227
20121228
20121229
20121230
20121231
20130101
20130102
20130103
20130104
20130105
20130106
20130107
20130108
20130109
20130110
20130111
20130112
20130113
20130114
20130115
20130116
20130117
20130118
20130119
20130120

20130121
20130122
20130123
20130124
20130125
20130126
20130127
20130128
20130129
20130130
20130131
20130201
20130202
20130203
20130204
20130205
20130206
20130207
20130208
20130209
20130210
20130211
20130212
20130213
20130214
20130215
20130216
20130217
20130218
20130219
20130220
20130221
20130222
20130223
20130224
20130225
20130226
20130227
20130228
20130301
20130302
20130303
20130304
20130305
20130306
20130307
20130308
20130309
20130310
20130311
20130312
20130313
20130314
20130315
20130316
20130317

```
20130318
20130319
20130320
20130321
20130322
20130323
20130324
20130325
20130326
20130327
20130328
20130329
20130330
20130331
```

In [9]:
```
weather_date_theo = [dt.strftime("%Y%m%d") for dt in daterange(start_dt, end
```

In [10]:
```
# One more inspection
weather_date_theo
```

Out[10]:
```
['20121001',
 '20121002',
 '20121003',
 '20121004',
 '20121005',
 '20121006',
 '20121007',
 '20121008',
 '20121009',
 '20121010',
 '20121011',
 '20121012',
 '20121013',
 '20121014',
 '20121015',
 '20121016',
 '20121017',
 '20121018',
 '20121019',
 '20121020',
 '20121021',
 '20121022',
 '20121023',
 '20121024',
 '20121025',
 '20121026',
 '20121027',
 '20121028',
 '20121029',
 '20121030',
 '20121031',
 '20121101',
 '20121102',
 '20121103',
 '20121104',
 '20121105',
 '20121106',
 '20121107',
 '20121108',
 '20121109',
 '20121110',
 '20121111',
 '20121112',
 '20121113',
 '20121114',
 '20121115',
 '20121116',
 '20121117',
 '20121118',
 '20121119',
 '20121120',
 '20121121',
 '20121122',
 '20121123',
 '20121124',
 '20121125',
```

'20121126',
'20121127',
'20121128',
'20121129',
'20121130',
'20121201',
'20121202',
'20121203',
'20121204',
'20121205',
'20121206',
'20121207',
'20121208',
'20121209',
'20121210',
'20121211',
'20121212',
'20121213',
'20121214',
'20121215',
'20121216',
'20121217',
'20121218',
'20121219',
'20121220',
'20121221',
'20121222',
'20121223',
'20121224',
'20121225',
'20121226',
'20121227',
'20121228',
'20121229',
'20121230',
'20121231',
'20130101',
'20130102',
'20130103',
'20130104',
'20130105',
'20130106',
'20130107',
'20130108',
'20130109',
'20130110',
'20130111',
'20130112',
'20130113',
'20130114',
'20130115',
'20130116',
'20130117',
'20130118',
'20130119',
'20130120',

'20130121',
'20130122',
'20130123',
'20130124',
'20130125',
'20130126',
'20130127',
'20130128',
'20130129',
'20130130',
'20130131',
'20130201',
'20130202',
'20130203',
'20130204',
'20130205',
'20130206',
'20130207',
'20130208',
'20130209',
'20130210',
'20130211',
'20130212',
'20130213',
'20130214',
'20130215',
'20130216',
'20130217',
'20130218',
'20130219',
'20130220',
'20130221',
'20130222',
'20130223',
'20130224',
'20130225',
'20130226',
'20130227',
'20130228',
'20130301',
'20130302',
'20130303',
'20130304',
'20130305',
'20130306',
'20130307',
'20130308',
'20130309',
'20130310',
'20130311',
'20130312',
'20130313',
'20130314',
'20130315',
'20130316',
'20130317',

```
 '20130318',
 '20130319',
 '20130320',
 '20130321',
 '20130322',
 '20130323',
 '20130324',
 '20130325',
 '20130326',
 '20130327',
 '20130328',
 '20130329',
 '20130330',
 '20130331']
```

Check if any missing values

In [11]:
```
range_day_count = len(weather_date_theo)
print('There are supposed to be {0} days in the given period.'.format(range_
```

```
There are supposed to be 182 days in the given period.
```

In [12]:
```
range_hour_count = range_day_count * 24
print('There are supposed to be {0} hours in total in the given period.'.for
```

```
There are supposed to be 4368 hours in total in the given period.
```

Do notice that the weather station data, we would need to round the timestamp to the nearest hour.

In [13]:
```
# Pad it out to 24 hours
import itertools

csv_date_list = list(itertools.chain.from_iterable(itertools.repeat(x, 24) f
```

In [14]:
```
len(csv_date_list)
```

Out[14]:
```
4368
```

Create the list of hours

In [15]:
```
hour_lib = ['00:00', '01:00', '02:00', '03:00', '04:00', '05:00', '06:00', '
            '10:00', '11:00', '12:00', '13:00', '14:00', '15:00', '16:00', '
            '18:00', '19:00', '20:00', '21:00', '22:00', '23:00']
```

In [16]:
```
csv_time_list = hour_lib * len(weather_date_theo)
```

In [17]:
```
# Inspection only
csv_time_list
```

Out[17]:
```
['00:00',
 '01:00',
 '02:00',
 '03:00',
 '04:00',
 '05:00',
 '06:00',
 '07:00',
 '08:00',
 '09:00',
 '10:00',
 '11:00',
 '12:00',
 '13:00',
 '14:00',
 '15:00',
 '16:00',
 '17:00',
 '18:00',
 '19:00',
 '20:00',
 '21:00',
 '22:00',
 '23:00',
 '00:00',
 '01:00',
 '02:00',
 '03:00',
 '04:00',
 '05:00',
 '06:00',
 '07:00',
 '08:00',
 '09:00',
 '10:00',
 '11:00',
 '12:00',
 '13:00',
 '14:00',
 '15:00',
 '16:00',
 '17:00',
 '18:00',
 '19:00',
 '20:00',
 '21:00',
 '22:00',
 '23:00',
 '00:00',
 '01:00',
 '02:00',
 '03:00',
 '04:00',
 '05:00',
 '06:00',
 '07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
```

```
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
```

```
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
```

```
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
```

```
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
```

```
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
```

```
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
'16:00',
'17:00',
'18:00',
'19:00',
'20:00',
'21:00',
'22:00',
'23:00',
'00:00',
'01:00',
'02:00',
'03:00',
'04:00',
'05:00',
'06:00',
'07:00',
'08:00',
'09:00',
'10:00',
'11:00',
'12:00',
'13:00',
'14:00',
'15:00',
```

```
 '16:00',
 '17:00',
 '18:00',
 '19:00',
 '20:00',
 '21:00',
 '22:00',
 '23:00',
 '00:00',
 '01:00',
 '02:00',
 '03:00',
 '04:00',
 '05:00',
 '06:00',
 '07:00',
 '08:00',
 '09:00',
 '10:00',
 '11:00',
 '12:00',
 '13:00',
 '14:00',
 '15:00',
 '16:00',
 '17:00',
 '18:00',
 '19:00',
 '20:00',
 '21:00',
 '22:00',
 '23:00',
 '00:00',
 '01:00',
 '02:00',
 '03:00',
 '04:00',
 '05:00',
 '06:00',
 '07:00',
 '08:00',
 '09:00',
 '10:00',
 '11:00',
 '12:00',
 '13:00',
 '14:00',
 '15:00',
 ...]
```

Round up the time stamp to the nearest hour

In [18]:
```
# from math import floor

# def round_down_time(time):
```

```
#     hours, minutes = time.split(':')
#     hours = int(hours)
#     minutes = int(minutes)

#     if minutes > 30:
#         hours += 1
#         if hours == 24:hours=0

#     return '{}:00'.format(floor(hours))

# rounded_time = round_down_time('6:21')
# print(rounded_time)  # Output: '6:00'

# rounded_time = round_down_time('6:51')
# print(rounded_time)  # Output: '7:00'
```

In [19]:
```
# temp_table['Time'][0]
```

In [20]:
```
# rounded_time = list(round_down_time(x) for x in temp_table['Time'])
```

In [21]:
```
# Quick inspection
# rounded_time
```

In [22]:
```
# temp_table['Time_CST'] = rounded_time
```

We need to notice that there are 2 possible timezones in this case, one is `Central Standard Time`, another one is `Eastern Standard Time`. For states in Illinois and Wisconsin, there is no need to change the date and time after converting the timestamp to the nearest hour. However, due to the timezone difference in other states such as Michigan, you will need to dial back the time by one hour to convert it to Central Standard Time without Daylight Savings.

There is 1 hour difference. Therefore, we rely on one parameter `STATE`, which extracts the first 2 letters of the file name, to properly align the time stamps together.

In [23]:
```
## Extract the state indicator
STATE_2_LTR = filename[:2]
```

In [24]:
```
# Inspection
print('The weather station is from', STATE_2_LTR)
```

```
The weather station is from IL
```

In [25]:
```
# tester = weather_date_theo[:5]
# tester.pop(0)
# tester
```

In [26]:
```
# tester.append(str(int(tester[-1])+1))
# tester
```

In [27]:
```
import math
from datetime import datetime, timedelta
```

```
def round_datetime_to_nearest_hour(obj_arr, STATE_ID):
    arr_len = len(obj_arr)
    Date_CST = []
    Time_CST = []
    for i in range(arr_len):
        # iterate through
        date_item = obj_arr['Date'][i]
        time_item = obj_arr['Time'][i]
        t = datetime.strptime(date_item + " " + time_item, "%Y-%m-%d %H:%M")
        # Calculate the number of minutes past the last full hour
        minutes_past_hour = t.minute + t.second / 60
        # Round up to the next whole number of hours if the time is more tha
        # or round down to the current hour if it's less than 30 minutes pas
        if minutes_past_hour >= 30:
            num_hours = math.ceil(minutes_past_hour / 60)
        else:
            num_hours = 0
        # Create a new datetime object representing the rounded time
        if STATE_ID in ['IL', 'WI']:
            # No need to dial back one hour
            rounded_time_temp = t + timedelta(hours=num_hours)
        else:
            rounded_time_temp = t + timedelta(hours=num_hours-1)
        rounded_time=datetime(year=rounded_time_temp.year,
                              month=rounded_time_temp.month,
                              day=rounded_time_temp.day,
                              hour=rounded_time_temp.hour, minute=0, second=0)
        result_stamp = rounded_time.strftime("%Y%m%d %H:%M")

        new_date, new_time = result_stamp.split(' ')
        Date_CST.append(new_date)
        Time_CST.append(new_time)

    obj_arr['Date_CST'] = Date_CST
    obj_arr['Time_CST'] = Time_CST
    return obj_arr
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

We need to notice that there are 2 possible timezones in this case, one is `Central Standard Time`, another one is `Eastern Standard Time`

There is 1 hour difference. Therefore, we rely on one parameter `STATE`, which extracts the first 2 letters of the file name, to properly align the time stamps together.

Add the rounded time into the weather station data

In [28]:
```
# Example usage
weather_with_CST = round_datetime_to_nearest_hour(temp_table, STATE_ID)
```

In [29]:
```
# Perform inspection
weather_with_CST.head(10)
# Make sure you scroll to the right to check the newly added data
```

Out[29]:

	Date	Time	Temp (F)	RH (%)	Dewpt (F)	Wind Spd (mph)	Wind Direction (deg)	Peak Wind Gust(mph)	Low Cloud Ht (ft)	Med Cloud Ht (ft)	High Cloud Ht (ft)	Vi
0	2012-10-01	00:51	51	82	46	3	10	m	30000	m	m	
1	2012-10-01	01:51	50	89	47	0	0	m	30000	m	m	
2	2012-10-01	02:51	50	89	47	0	0	m	1000	30000	m	
3	2012-10-01	03:51	48	96	47	0	0	m	2400	30000	m	
4	2012-10-01	04:51	51	89	48	5	350	m	30000	m	m	
5	2012-10-01	05:51	50	92	48	0	0	m	30000	m	m	
6	2012-10-01	06:51	51	92	49	0	0	m	25000	m	m	
7	2012-10-01	07:51	57	68	47	3	170	m	25000	m	m	
8	2012-10-01	08:51	60	59	46	6	170	m	25000	m	m	
9	2012-10-01	09:51	62	51	44	0	0	m	25000	m	m	

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Convert to Pandas series for easy computation

In [30]:
```
Date_CST = pandas.Series(csv_date_list, name = 'Date_CST')
Date_CST
```

Out[30]:
```
0       20121001
1       20121001
2       20121001
3       20121001
4       20121001
          ...
4363    20130331
4364    20130331
4365    20130331
4366    20130331
4367    20130331
Name: Date_CST, Length: 4368, dtype: object
```

In [31]:
```
Time_CST = pandas.Series(csv_time_list, name = 'Time_CST')
Time_CST
```

Out[31]:
```
0       00:00
1       01:00
2       02:00
3       03:00
4       04:00
        ...
4363    19:00
4364    20:00
4365    21:00
4366    22:00
4367    23:00
Name: Time_CST, Length: 4368, dtype: object
```

In [32]:
```
df_Reference_Date_Time = pandas.merge(Date_CST, Time_CST, right_index = True
                left_index = True)
df_Reference_Date_Time
```

Out[32]:

	Date_CST	Time_CST
0	20121001	00:00
1	20121001	01:00
2	20121001	02:00
3	20121001	03:00
4	20121001	04:00
...	...	...
4363	20130331	19:00
4364	20130331	20:00
4365	20130331	21:00
4366	20130331	22:00
4367	20130331	23:00

4368 rows × 2 columns

`df_Reference_Date_Time` contains all the possible date and time within the given period of the weather station data. In next step, we are going to perform `join` operation to merge the weather station data and the GOES satellite imagery data.

In [33]:
```
df_time_and_weather = pandas.merge(df_Reference_Date_Time, weather_with_CST,
```

In [34]:
```
# Perform inspection
df_time_and_weather.head(10)
```

Out[34]:

	Date_CST	Time_CST	Date	Time	Temp (F)	RH (%)	Dewpt (F)	Wind Spd (mph)	Wind Direction (deg)	Peak Wind Gust(mph)	C Ht
0	20121001	00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	20121001	01:00	2012-10-01	00:51	51.0	82	46	3.0	10	m	30
2	20121001	02:00	2012-10-01	01:51	50.0	89	47	0.0	0	m	30
3	20121001	03:00	2012-10-01	02:51	50.0	89	47	0.0	0	m	1
4	20121001	04:00	2012-10-01	03:51	48.0	96	47	0.0	0	m	2
5	20121001	05:00	2012-10-01	04:51	51.0	89	48	5.0	350	m	30
6	20121001	06:00	2012-10-01	05:51	50.0	92	48	0.0	0	m	30
7	20121001	07:00	2012-10-01	06:51	51.0	92	49	0.0	0	m	25
8	20121001	08:00	2012-10-01	07:51	57.0	68	47	3.0	170	m	25
9	20121001	09:00	2012-10-01	08:51	60.0	59	46	6.0	170	m	25

Note:

When it comes to the abbr in the table:

`m` or `M` : Data is missing `NC` : Wind Chill/Heat Index do not meet the required thresholds to be calculated.

Merge GOES Satellite Imagery Data

There are 2 parts of the data that need to be merged together: one is the hourly statistics, another one is the observation data based on the hourly imagery. The first piece of data gives you the information regarding the coverage of the cloud which is visible above the Lake Michigan. The second piece of data tells you whether the visible

information is reliable, and if the Lake-Effect Snow cloud can be seen above the lake in the given hour.

Merge Statistical Data

In [35]:
```
df_Hourly_Statistics = pandas.read_csv('GOES_Hourly_Statistics/stats_result/
```

In [36]:
```
types_dict = {'Mean': float, 'Centroid_lon': float, 'Std_lon': float,
              'Std_lat': float, 'Skewness_lon': float, 'Skewness_lat': float
              'Kurtosis_lon': float, 'Kurtosis_lat': float, 'Sample Number':

for col, col_type in types_dict.items():
    df_Hourly_Statistics[col] = df_Hourly_Statistics[col].astype(col_type)
```

In [37]:
```
# Just to confirm the data type of a couple critical columns
type(df_Hourly_Statistics['CST_Time'][6])   # Should be string
```

Out[37]:
```
str
```

In [38]:
```
type(df_Hourly_Statistics['Std_lon'][6])    # Should be float
```

Out[38]:
```
numpy.float64
```

In [39]:
```
df_Hourly_Statistics = df_Hourly_Statistics.rename({'CST_Date': 'Date_CST',
df_Hourly_Statistics = df_Hourly_Statistics.rename({'Date': 'Date_UTC', 'Tim
```

In [40]:
```
df_Hourly_Statistics['Time_CST'] = df_Hourly_Statistics['Time_CST'].apply(la
```

In [41]:
```
# Inspection
df_Hourly_Statistics.head(10)
```

Out[41]:

	Date_CST	Date_UTC	Time_CST	Time_UTC	Mean	Centroid_lon	Centroid_la
0	20130930	20131001	18:00	'0000	0.002116	-86.752361	43.8867915231490
1	20130930	20131001	19:00	'0100	0.001849	-86.758334	43.8829940122162
2	20130930	20131001	20:00	'0200	0.002148	-86.726783	43.9311395879013
3	20130930	20131001	21:00	'0300	0.001968	-86.744962	43.84049622477249
4	20130930	20131001	22:00	'0400	0.001933	-86.749047	43.888691437802
5	20130930	20131001	23:00	'0500	0.001874	-86.726929	43.91144356899289
6	20131001	20131001	00:00	'0600	0.001979	-86.711706	43.9764250678313
7	20131001	20131001	01:00	'0700	0.002175	-86.728150	43.9444317640044
8	20131001	20131001	02:00	'0800	0.007346	-86.727714	43.9266989534256
9	20131001	20131001	03:00	'0900	0.001971	-86.732769	43.9312630358059

In [42]:
```
df_time_weather_stats = pandas.merge(df_time_and_weather, df_Hourly_Statisti
```

In [43]:
```
# Inspection
df_time_weather_stats.head(20)
```

Out[43]:

	Date_CST	Time_CST	Date	Time	Temp (F)	RH (%)	Dewpt (F)	Wind Spd (mph)	Wind Direction (deg)	Peak Wind Gust(mph)	..
0	20121001	00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.
1	20121001	01:00	2012-10-01	00:51	51.0	82	46	3.0	10	m	.
2	20121001	02:00	2012-10-01	01:51	50.0	89	47	0.0	0	m	.
3	20121001	03:00	2012-10-01	02:51	50.0	89	47	0.0	0	m	.
4	20121001	04:00	2012-10-01	03:51	48.0	96	47	0.0	0	m	.
5	20121001	05:00	2012-10-01	04:51	51.0	89	48	5.0	350	m	.
6	20121001	06:00	2012-10-01	05:51	50.0	92	48	0.0	0	m	.
7	20121001	07:00	2012-10-01	06:51	51.0	92	49	0.0	0	m	.
8	20121001	08:00	2012-10-01	07:51	57.0	68	47	3.0	170	m	.
9	20121001	09:00	2012-10-01	08:51	60.0	59	46	6.0	170	m	.
10	20121001	10:00	2012-10-01	09:51	62.0	51	44	0.0	0	m	.
11	20121001	11:00	2012-10-01	10:51	64.0	44	42	6.0	80	m	.
12	20121001	12:00	2012-10-01	11:51	65.0	43	42	7.0	80	m	.
13	20121001	13:00	2012-10-01	12:51	65.0	43	42	8.0	80	m	.
14	20121001	14:00	2012-10-01	13:51	64.0	44	42	10.0	70	m	.
15	20121001	15:00	2012-10-01	14:51	63.0	48	43	10.0	90	m	.
16	20121001	16:00	2012-10-01	15:51	62.0	55	46	10.0	90	m	.
17	20121001	17:00	2012-10-01	16:51	61.0	59	47	9.0	60	m	.
18	20121001	18:00	2012-10-01	17:51	60.0	61	47	7.0	50	m	.
19	20121001	19:00	2012-10-01	18:51	59.0	66	48	8.0	50	m	.

20 rows × 33 columns

In [44]:
```
column_headers = list(df_time_weather_stats.columns.values)
column_headers
```

Out[44]:
```
['Date_CST',
 'Time_CST',
 'Date',
 'Time',
 'Temp (F)',
 'RH (%)',
 'Dewpt (F)',
 'Wind Spd (mph)',
 'Wind Direction (deg)',
 'Peak Wind Gust(mph)',
 'Low Cloud Ht (ft)',
 'Med Cloud Ht (ft)',
 'High Cloud Ht (ft)',
 'Visibility (mi)',
 'Atm Press (hPa)',
 'Sea Lev Press (hPa)',
 'Altimeter (hPa)',
 'Precip (in)',
 'Wind Chill (F)',
 'Heat Index (F)',
 'Date_UTC',
 'Time_UTC',
 'Mean',
 'Centroid_lon',
 'Centroid_lat',
 'Std_lon',
 'Std_lat',
 'Skewness_lon',
 'Skewness_lat',
 'Kurtosis_lon',
 'Kurtosis_lat',
 'Sample Number',
 'Selected']
```

Now, in the `df_time_weather_stats`, you only need to add the observation data into the table to form the final test datasets.

In [45]:
```
df_observ = pandas.read_csv('Observation_Data/2013_2014_observation.csv', dt
```

In [46]:
```
# Inspection
df_observ.head(10)
```

Out[46]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	Lake_Visible	LES_Exist
0	20131001	00:00	20130930	18:00	N	N
1	20131001	01:00	20130930	19:00	N	N
2	20131001	02:00	20130930	20:00	N	N
3	20131001	03:00	20130930	21:00	N	N
4	20131001	04:00	20130930	22:00	N	N
5	20131001	05:00	20130930	23:00	N	N
6	20131001	06:00	20131001	00:00	N	N
7	20131001	07:00	20131001	01:00	N	N
8	20131001	08:00	20131001	02:00	N	N
9	20131001	09:00	20131001	03:00	N	N

In [47]:
```
df_time_weather_stats_label = pandas.merge(df_time_weather_stats, df_observ,
```

In [48]:
```
df_time_weather_stats_label.head(10)
```

Out[48]:

	Date_CST	Time_CST	Date	Time	Temp (F)	RH (%)	Dewpt (F)	Wind Spd (mph)	Wind Direction (deg)	Peak Wind Gust(mph)	...
0	20121001	00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
1	20121001	01:00	2012-10-01	00:51	51.0	82	46	3.0	10	m	...
2	20121001	02:00	2012-10-01	01:51	50.0	89	47	0.0	0	m	...
3	20121001	03:00	2012-10-01	02:51	50.0	89	47	0.0	0	m	...
4	20121001	04:00	2012-10-01	03:51	48.0	96	47	0.0	0	m	...
5	20121001	05:00	2012-10-01	04:51	51.0	89	48	5.0	350	m	...
6	20121001	06:00	2012-10-01	05:51	50.0	92	48	0.0	0	m	...
7	20121001	07:00	2012-10-01	06:51	51.0	92	49	0.0	0	m	...
8	20121001	08:00	2012-10-01	07:51	57.0	68	47	3.0	170	m	...
9	20121001	09:00	2012-10-01	08:51	60.0	59	46	6.0	170	m	...

10 rows × 37 columns

In [49]:
```
last_column_headers = list(df_time_weather_stats_label.columns.values)
```

```
last_column_headers
```

Out[49]:
```
['Date_CST',
 'Time_CST',
 'Date',
 'Time',
 'Temp (F)',
 'RH (%)',
 'Dewpt (F)',
 'Wind Spd (mph)',
 'Wind Direction (deg)',
 'Peak Wind Gust(mph)',
 'Low Cloud Ht (ft)',
 'Med Cloud Ht (ft)',
 'High Cloud Ht (ft)',
 'Visibility (mi)',
 'Atm Press (hPa)',
 'Sea Lev Press (hPa)',
 'Altimeter (hPa)',
 'Precip (in)',
 'Wind Chill (F)',
 'Heat Index (F)',
 'Date_UTC_x',
 'Time_UTC_x',
 'Mean',
 'Centroid_lon',
 'Centroid_lat',
 'Std_lon',
 'Std_lat',
 'Skewness_lon',
 'Skewness_lat',
 'Kurtosis_lon',
 'Kurtosis_lat',
 'Sample Number',
 'Selected',
 'Date_UTC_y',
 'Time_UTC_y',
 'Lake_Visible',
 'LES_Exist']
```

Now, you are free to manipulate or save the datasets in the way you desire.

In [50]:
```
result_filename = 'Combined_Result/' + STATE_ID + '_' + weather_station + '_
result_filename
```

Out[50]:
```
'Combined_Result/IL_94846_2012Fall_2013Spring.csv'
```

In [51]:
```
# For testing purposes

df_time_weather_stats_label.to_csv(result_filename, index = False)

# import csv
# with open(result_filename, 'w') as csvfile:
#     writer = csv.writer(csvfile)
#     writer.writerows(df_time_weather_stats_label)
```

In [ ]: