

In [200]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
%pprint off
# plt.style.use('seaborn')
```

Pretty printing has been turned OFF

In [201]:

```
sns.set_style("ticks")
```

## Preparing the dataframes

In [202]:

```
df_all = pd.read_csv('data/dataport-export_gas_oct2015-mar2016.csv')
len(df_all)
```

Out[202]:

1584823

In [203]:

```
df_all = df_all.set_index(pd.to_datetime(df_all['localminute']))
```

In [204]:

```
df_all = df_all.drop(columns='localminute')
display(df_all.head())
```

	dataid	meter_value
localminute		
2015-10-01 05:00:10	739	88858
2015-10-01 05:00:13	8890	197164
2015-10-01 05:00:20	6910	179118
2015-10-01 05:00:22	3635	151318
2015-10-01 05:00:22	1507	390354

In [205]:

```
groups = df_all.groupby('dataid')
keys = groups.groups.keys() # keys: an iterable of dataids or meter ids

# check if each group (grouped by meter id) is sorted in ascending order by datetime.
# for key in keys:
#     df_i = groups.get_group(key)
#     print(df_i.index.is_monotonic_increasing)
# each group is already sorted in ascending order by datetime.
```

## Check meterids

In [206]:

```
keys_list = list(keys)
print(keys_list)
```

```
[35, 44, 77, 94, 114, 187, 222, 252, 370, 483, 484, 661, 739, 744, 871, 10
42, 1086, 1103, 1185, 1283, 1403, 1415, 1507, 1556, 1589, 1619, 1697, 171
4, 1718, 1790, 1791, 1792, 1800, 1801, 2018, 2034, 2072, 2094, 2129, 2233,
2335, 2378, 2449, 2461, 2470, 2575, 2638, 2645, 2755, 2814, 2818, 2945, 29
46, 2965, 2980, 3036, 3039, 3134, 3310, 3367, 3527, 3544, 3577, 3635, 372
3, 3778, 3849, 3893, 3918, 4029, 4031, 4193, 4228, 4296, 4352, 4356, 4373,
4421, 4447, 4514, 4671, 4732, 4767, 4874, 4998, 5129, 5131, 5193, 5275, 53
17, 5395, 5403, 5439, 5484, 5545, 5636, 5658, 5785, 5810, 5814, 5892, 597
2, 6101, 6412, 6505, 6578, 6673, 6685, 6830, 6836, 6863, 6910, 7016, 7017,
7030, 7117, 7287, 7429, 7460, 7566, 7674, 7682, 7739, 7741, 7794, 7900, 79
19, 7965, 7989, 8059, 8084, 8086, 8155, 8156, 8244, 8386, 8467, 8703, 882
9, 8890, 8967, 9052, 9121, 9134, 9160, 9278, 9295, 9474, 9600, 9620, 9631,
9639, 9729, 9766, 9849, 9956, 9982]
```

## Print full-length (6 mth) plot by meterid.

In [207]:

```
# for key in keys_list:
#     df_i = groups.get_group(key)
#     df_i.drop(columns='dataid').plot(figsize=(15,4), title=str(f'meter {key}'))
```

In [208]:

```
key = 3134
df_i = groups.get_group(key).drop(columns='dataid')
```

## Select data by datetime period. e.g. 1 month. i.e. Zoom in.

In [209]:

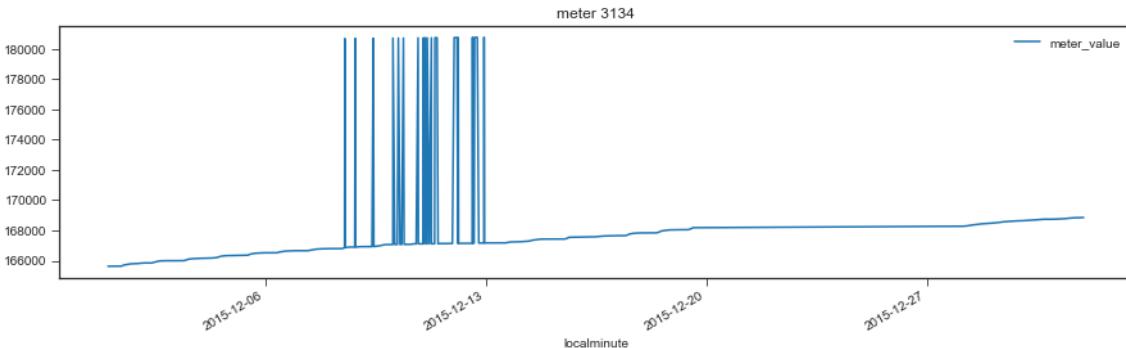
```
def zoom(df, start_date, end_date):
    # pre-condition: df is indexed by datetime.
    start_date = pd.to_datetime(start_date)
    end_date = pd.to_datetime(end_date)
    mask = (df.index >= start_date) & (df.index <= end_date)

    new_df = df.iloc[mask]
    return new_df

df_i_bymonth = zoom(df_i, '2015-12-01', '2016-01-01')
df_i_bymonth.plot(figsize=(15,4), title=str(f'meter {key}'))
```

Out[209]:

&lt;matplotlib.axes.\_subplots.AxesSubplot object at 0x000001FE9A104DA0&gt;

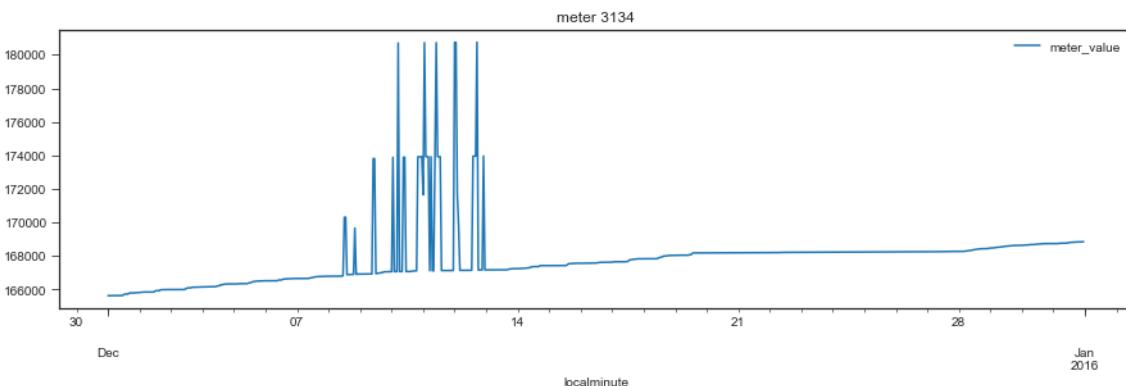


## Resample with hourly frequency.

In [210]:

```
df_i_bymonth_resampled = df_i_bymonth.resample('H').mean().ffill()
df_i_bymonth_resampled.plot(figsize=(15,4), title=str(f'meter {key}'))
display(len(df_i_bymonth_resampled))
```

744



**Iterate and plot december's hourly-resampled data for each suspicious meter (spiking values).**

In [211]:

```
suspicious_meters_list = [1185, 1556, 2335, 2449, 3134, 3544, 4447, 4514, \
                           5129, 5403, 6836, 7030, 7117, 8156, 9134, 9639, 9982]

# for meter in suspicious_meters_list:
#     df_i = groups.get_group(meter)
#     df_i_bymonth = zoom(df_i, '2015-12-01', '2016-01-01').drop(columns='dataid')
#     df_i_bymonth_resampled = df_i_bymonth.resample('H').mean().ffill()
#     df_i_bymonth_resampled.plot(figsize=(15,4), title=str(f'meter {meter}, {len(df_i_bymonth_resampled)} samples'))
```

We clearly see that there is some obvious malfunctioning happening within 7-14 Dec 2015.

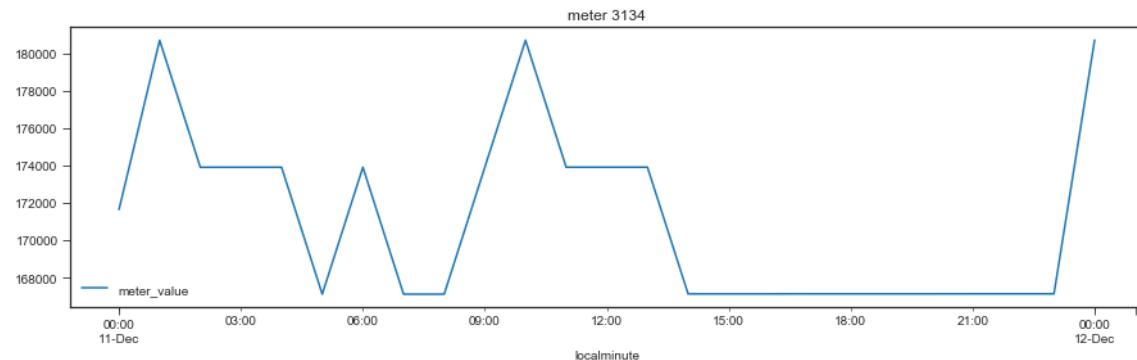
## Zoom in deeper within the time period (e.g. one day/24h)

In [212]:

```
df_i_byday = zoom(df_i_bymonth_resampled, '2015-12-11', '2015-12-12')
df_i_byday.plot(figsize=(15,4), title=str(f'meter {key}'))
```

Out[212]:

```
<matplotlib.axes._subplots.AxesSubplot object at 0x000001FEB86D8588>
```



In [213]:

```
# for i in range(7):
#     start_date = f'2015-12-{8+i}'
#     end_date = f'2015-12-{9+i}'
#     df_i_byday = zoom(df_i_bymonth_resampled, start_date, end_date)
#     df_i_byday.plot(figsize=(15,4), title=str(f'meter {key}'))
```

In [214]:

```
less500_list = [4874, 6101, 9620, 7566, 5545, 2814, 2946, 2755,\n                9160, 2645, 6685, 1403, 8703, 9600, 3036, 5658]\n\ndef get_month_counts(df_i):\n    # get count of samples per month\n    df_i_resample_monthly_count = df_i.resample('M').count().rename(columns={'meter_value': 'count'})\n    return df_i_resample_monthly_count\n\nfor meter in less500_list:\n    df_i = groups.get_group(meter)\n    df_i_counts = get_month_counts(df_i)\n    display(df_i_counts.head(6))
```

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-02-29</b>	1	1
<b>2016-03-31</b>	1	1

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	3	3

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-10-31</b>	1	1
<b>2015-11-30</b>	0	0
<b>2015-12-31</b>	0	0
<b>2016-01-31</b>	3	3
<b>2016-02-29</b>	13	13
<b>2016-03-31</b>	6	6

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	32	32

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-12-31</b>	3	3
<b>2016-01-31</b>	16	16
<b>2016-02-29</b>	13	13
<b>2016-03-31</b>	1	1

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	6	6
<b>2016-02-29</b>	31	31

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	45	45

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	68	68

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	72	72

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-10-31</b>	12	12
<b>2015-11-30</b>	10	10
<b>2015-12-31</b>	1	1
<b>2016-01-31</b>	19	19
<b>2016-02-29</b>	32	32

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-10-31</b>	1	1
<b>2015-11-30</b>	0	0
<b>2015-12-31</b>	14	14
<b>2016-01-31</b>	21	21
<b>2016-02-29</b>	39	39
<b>2016-03-31</b>	3	3

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	202	202

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-10-31</b>	134	134
<b>2015-11-30</b>	0	0
<b>2015-12-31</b>	3	3
<b>2016-01-31</b>	117	117
<b>2016-02-29</b>	3	3
<b>2016-03-31</b>	1	1

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	330	330

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2016-01-31</b>	336	336

	<b>dataid</b>	<b>count</b>
<b>localminute</b>		
<b>2015-11-30</b>	98	98
<b>2015-12-31</b>	176	176
<b>2016-01-31</b>	201	201
<b>2016-02-29</b>	16	16
<b>2016-03-31</b>	2	2

In [215]:

```
count_list = []
for key in keys_list:
    df_i = groups.get_group(key)
    count_list.append(len(df_i.index))

less2000_mask = np.asarray(count_list) < 2000
display(less2000_mask)

array([False, True, False, False, False, True, False, False, False,
       False, False, False, False, False, False, False, True,
       False, False, True, True, False, False, False, False, False,
       False, False, False, True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, True,
       False, False, True, True, True, False, True, False,
       False, True, False, False, False, False, False, False, False,
       False, False, False, True, False, False, False, False, True,
       True, True, False, True, True, False, False, False, True,
       False, False, True, False, False, False, False, False, True,
       True, False, False, True, False, False, False, False, True,
       True, False, False, True, False, False, False, False, True,
       False, False, False, True, False, False, False, False, True,
       False, False, False, True, False, False, False, False, True,
       True, False, True, False, False, False, False, False, True,
       True, False, True, False, False, False, False, False, True,
       True, False, False, False, True, True, False, False, False,
       False, False, True, True, False, False, False, False, True])
```

In [216]:

```
less2000_list = []
for i in range(len(less2000_mask)):
    if less2000_mask[i] == True:
        less2000_list.append(keys_list[i])

display(less2000_list)

[44, 187, 1103, 1403, 1415, 1792, 2470, 2645, 2755, 2814, 2818, 2946, 303
6, 3849, 4193, 4228, 4296, 4356, 4373, 4671, 4874, 5317, 5395, 5545, 5658,
6101, 6505, 6578, 6685, 6863, 7566, 7965, 8059, 8244, 8386, 8703, 9160, 96
00, 9620, 9956, 9982]
```

In [217]:

```
# for meter in Less2000_List:
#     df_i = groups.get_group(meter).drop(columns='dataid')
#     df_i_resample_monthly_count = df_i.resample('M').count() # get count of samples
#     per month
#
#     display(f'avg sample per month: {len(df_i)/len(df_i_resample_monthly_count)}', \
#            f'nmonths={len(df_i_resample_monthly_count)})')
```

## Resample data hourly, taking last value in the hour as new value.

For each meter:

- Visualise 6month usage
- Visualise 1month usage
- Visualise 1day usage

In [218]:

```
# # Resample by taking Last cumulative reading for each hour.

# key = keys_list[4]
# df_i = groups.get_group(key).drop(columns='dataid')
# df_i.plot(figsize=(15,4), title=str(f'meter {key}'))

# # resample hourly using last reading for each hour, forward-filling any missing values
# df_i_resample_hourly = df_i.resample('H').last().ffill()
# df_i_resample_hourly.plot(figsize=(15,4), title=str(f'meter {key}'))

# display(f'total samples: {len(df_i)}')
# display(f'no. of hours: {len(df_i_resample_hourly)}')
# first_date = df_i.index.values[0]
# last_date = df_i.index.values[-1]
# display(f'first date: {first_date}')
# display(f'last date: {last_date}')
# display(df_i_resample_hourly.tail())

# # zoom in to particular month
# df_i_month = zoom(df_i_resample_hourly, '2016-01-01', '2016-02-01')
# df_i_month.plot(figsize=(15,4), title=str(f'meter {key}'))

# # zoom in to particular day
# df_i_day = zoom(df_i_month, '2016-01-02', '2016-01-03')
# df_i_day.plot(figsize=(15,4), title=str(f'meter {key}'))

# # find hour on hour change, i.e. marginal hourly usage
# # this is useful to notice hourly consumption patterns
# df_i_day.diff().plot(figsize=(15,4), title=str(f'meter {key}'))
```

In [219]:

```
def remove_mal_data(df):
    # this method finds the start datetime and end datetime of the malfunctioning period and returns a new df without data
    # from that period. NOTE that a new column 'marginal_change' is added to the df.

    # from visualising the data, we can arbitrarily define a spike as a marginal difference of > 2000 cubic metres
    # we have defined malfunction as a marginal increase > 2000
    # and a marginal increase < -2000.
    threshold = 2000
    df['marginal_change'] = df['meter_value'].diff()

    flagged_dates = df[df.marginal_change > threshold].index
    flagged_dates = flagged_dates.append(df[df.marginal_change < -1*threshold].index)

    if len(flagged_dates) > 0:
        start = flagged_dates[0]
        end = flagged_dates[-1]
        mask = (df.index >= start) & (df.index <= end)
        return df.loc[~mask]
    else:
        return df
```

In [220]:

```
key = 1185
df_i = groups.get_group(key).drop(columns='dataid')
display(len(df_i))

df_i_clean = remove_mal_data(df_i)
display(len(df_i_clean))

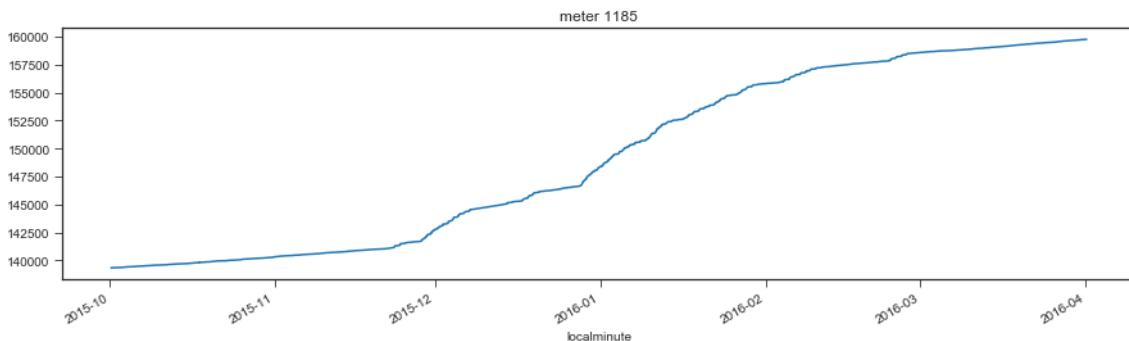
df_i_clean.meter_value.plot(figsize=(15,4), title=str(f'meter {key}'))
```

18456

17804

Out[220]:

&lt;matplotlib.axes.\_subplots.AxesSubplot object at 0x000001FEB54BAAC8&gt;



In [221]:

```
def remove_negative_marginal(df):
    # remove data points where the marginal change (from prev value) is negative.
    return df.loc[df.marginal_change > 0]
```

In [222]:

```
def remove_outliers(df, coeff=1.5):
    # remove outliers using *IQR rule.
    tmp_df = df.loc[df.marginal_change > 1]
    Q1 = tmp_df['marginal_change'].quantile(0.25)
    Q3 = tmp_df['marginal_change'].quantile(0.75)
    IQR = Q3 - Q1

    #print(f'{Q1, Q3, IQR}')

    # Filtering Values between Q1-1.5IQR and Q3+1.5IQR
    return df.query('(@Q1 - @coeff * @IQR) <= marginal_change <= (@Q3 + @coeff * @IQR)')
)
```

In [223]:

```
display(len(df_i_clean))
```

17804

In [224]:

```
df_i_filtered = remove_outliers(df_i_clean)
display(len(df_i_filtered))
```

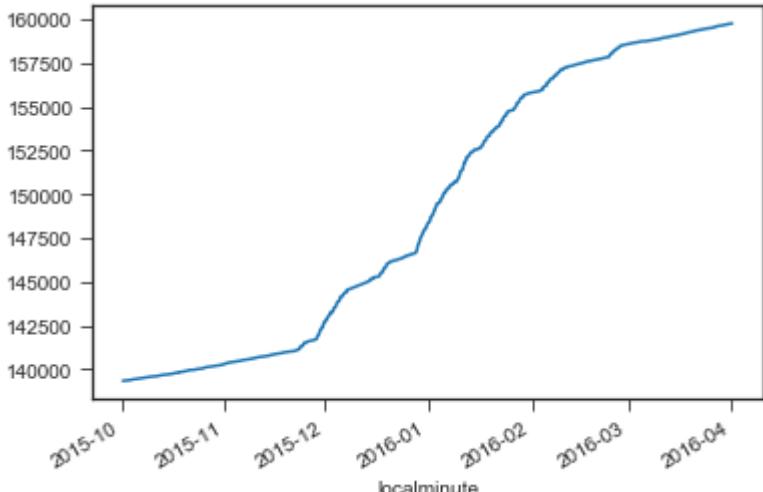
17481

In [225]:

```
df_i_filtered['meter_value'].plot()
```

Out[225]:

```
<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE831CE748>
```



**Apply cleaning, resampling, and 1.5IQR-filtering to entire 6 month period.**

In [226]:

```
for key in suspicious_meters_list:
    df_i = groups.get_group(key).drop(columns='dataid')
    display(len(df_i))

    # first, clean data by removing obvious dirt
    df_i_clean = remove_mal_data(df_i)
    df_i_clean = remove_negative_marginal(df_i_clean)

    # next, resample data by hour.
    df_i_resampled = df_i_clean.drop(columns='marginal_change').resample('H').mean().ffill()
    display(df_i_resampled.head())

    # then, obtain new marginal changes.
    df_i_resampled['marginal_change'] = df_i_resampled['meter_value'].diff()
    display(df_i_resampled.head())
    df_i_resampled['marginal_change'].plot(figsize=(15,4), title=str(f'meter {key}'))
    display(len(df_i_resampled))

    # finally, filter by the 1.5IQR rule on marginal_change.
    df_i_filtered = remove_outliers(df_i_resampled)
    display(len(df_i_filtered))

    df_i_filtered['marginal_change'].plot(figsize=(15,4), title=str(f'meter {key}'))
    break
```

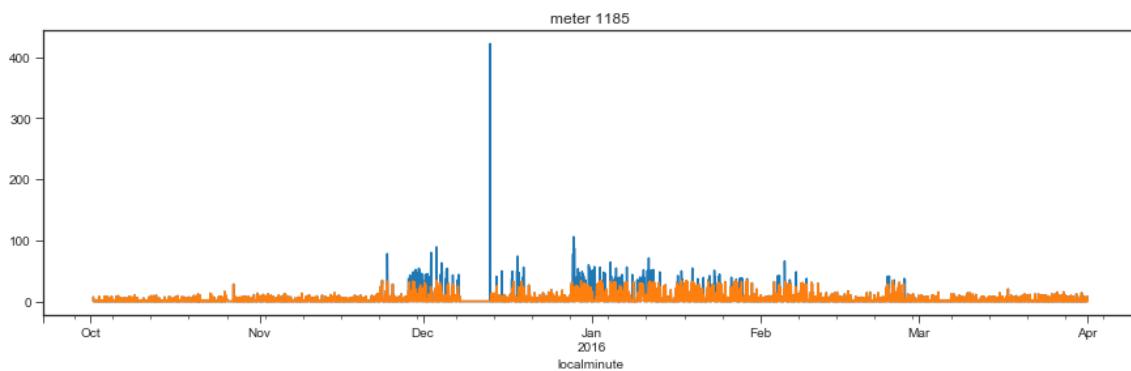
18456

	<b>meter_value</b>
<b>localminute</b>	
<b>2015-10-01 10:00:00</b>	139338.0
<b>2015-10-01 11:00:00</b>	139345.0
<b>2015-10-01 12:00:00</b>	139345.0
<b>2015-10-01 13:00:00</b>	139348.0
<b>2015-10-01 14:00:00</b>	139348.0

	<b>meter_value</b>	<b>marginal_change</b>
<b>localminute</b>		
<b>2015-10-01 10:00:00</b>	139338.0	NaN
<b>2015-10-01 11:00:00</b>	139345.0	7.0
<b>2015-10-01 12:00:00</b>	139345.0	0.0
<b>2015-10-01 13:00:00</b>	139348.0	3.0
<b>2015-10-01 14:00:00</b>	139348.0	0.0

4386

4247



Above code snippet and output shows that IQR-filtering should not be applied to the entire 6 months, because of seasonal changes in household use of gas.

In [239]:

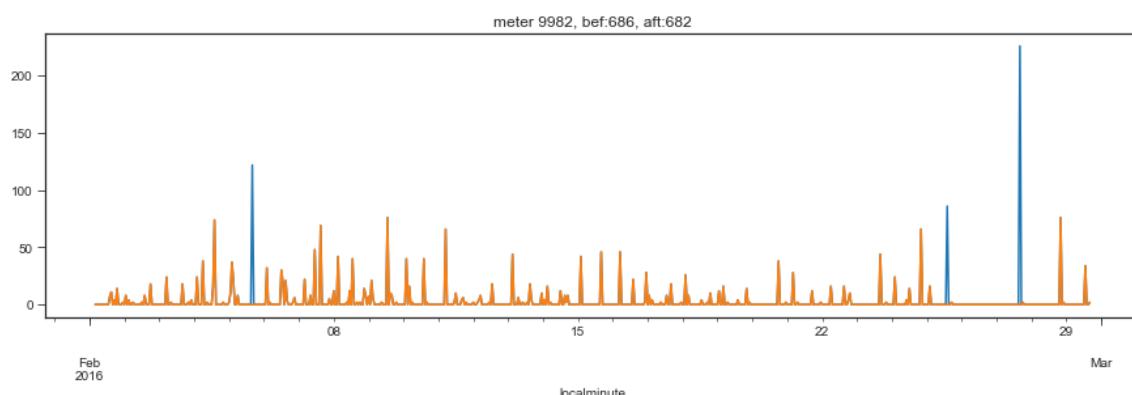
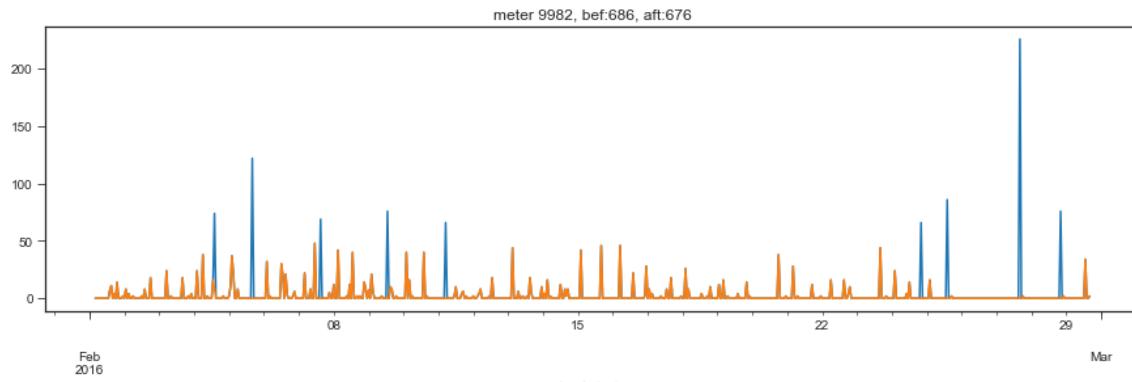
```
clean_resample_filter(df_i_1mth)
```

C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel\_launcher.py:9: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-versus-copy>

```
if __name__ == '__main__':
```



<Figure size 432x288 with 0 Axes>

IQR-filtering seems more reasonable after constraining the time period to one month.

In [240]:

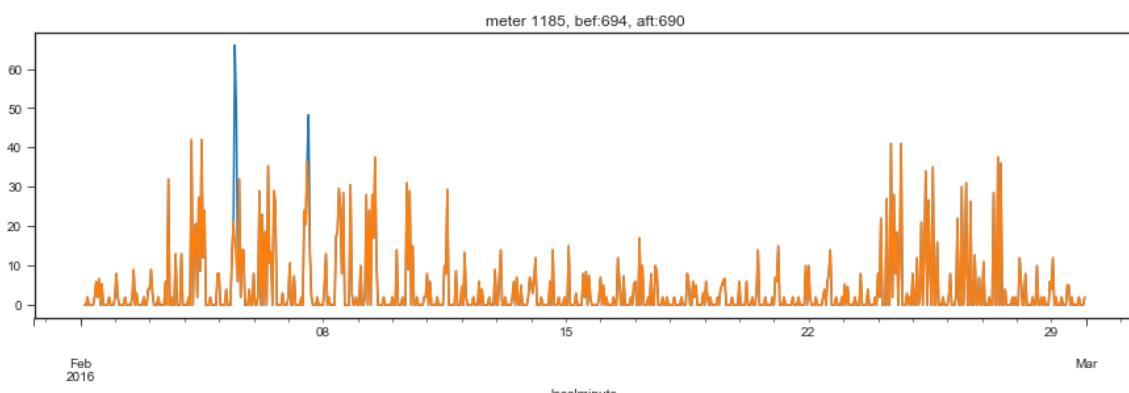
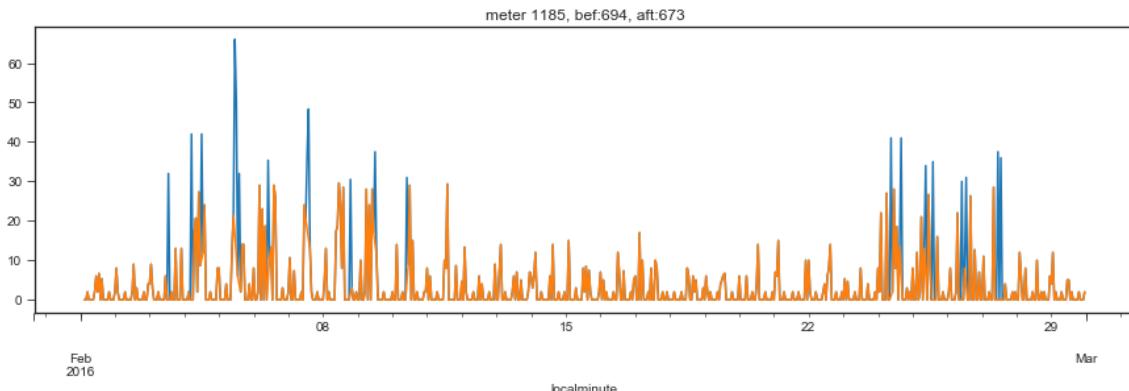
```
for key in suspicious_meters_list:  
    df_i = groups.get_group(key).drop(columns='dataid')  
    display(len(df_i))  
    df_i_1mth = zoom(df_i, '2016-02-01', '2016-03-01')  
    clean_resample_filter(df_i)
```

18456

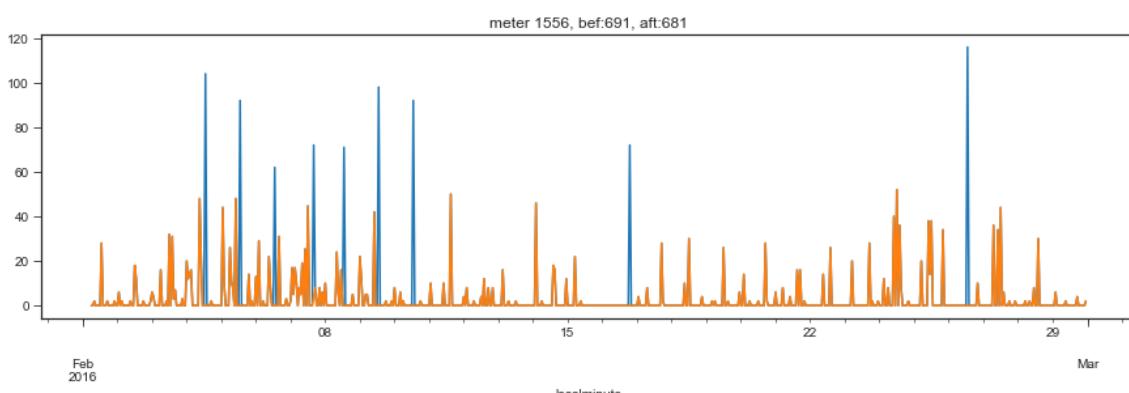
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel\_launcher.py:9: SettingWithCopyWarning:  
 A value is trying to be set on a copy of a slice from a DataFrame.  
 Try using .loc[row\_indexer,col\_indexer] = value instead

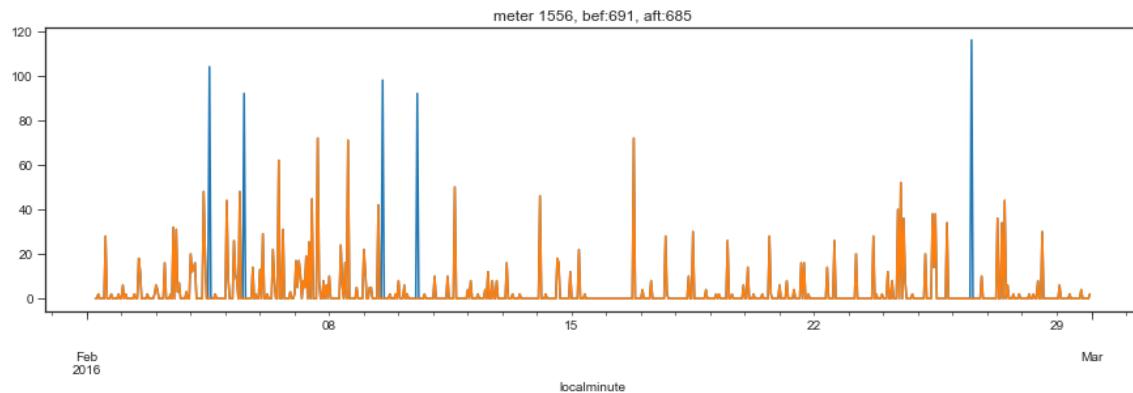
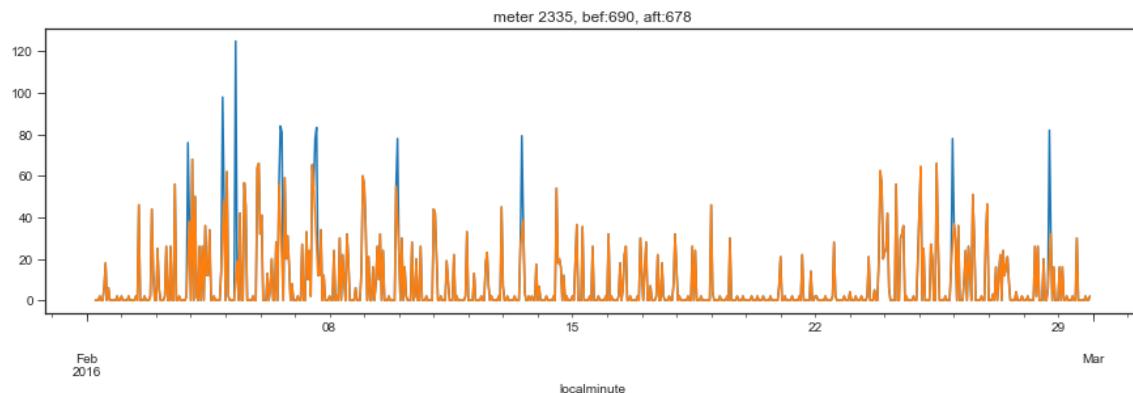
See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

```
if __name__ == '__main__':
```

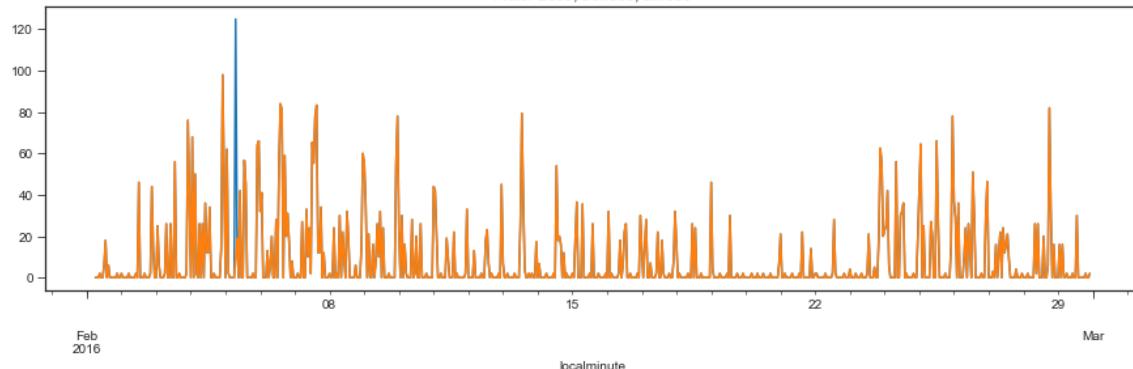
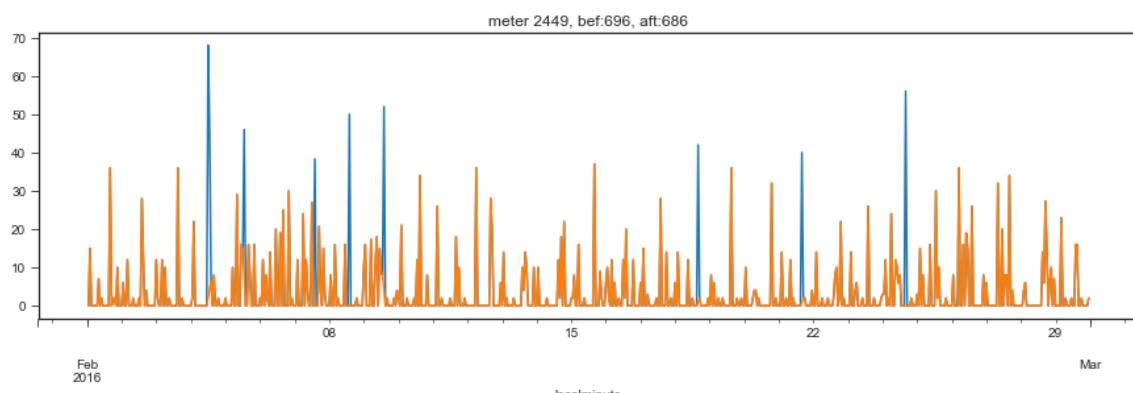


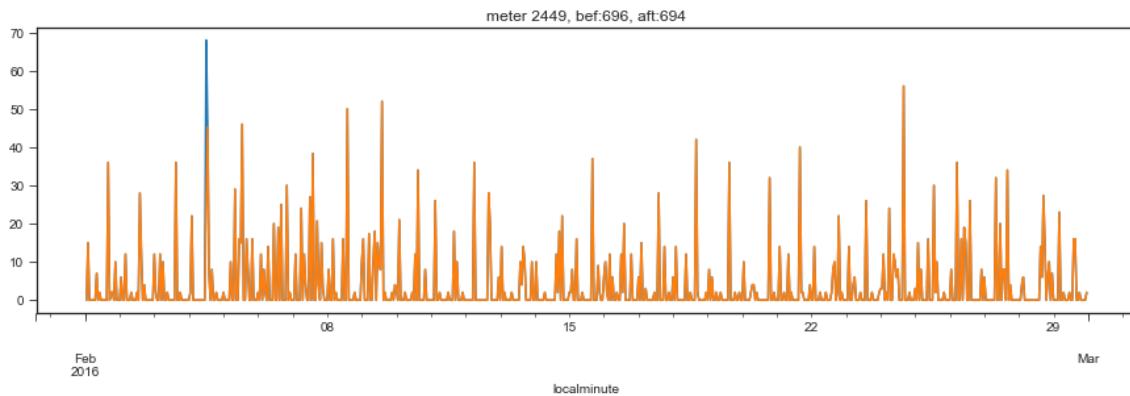
3690



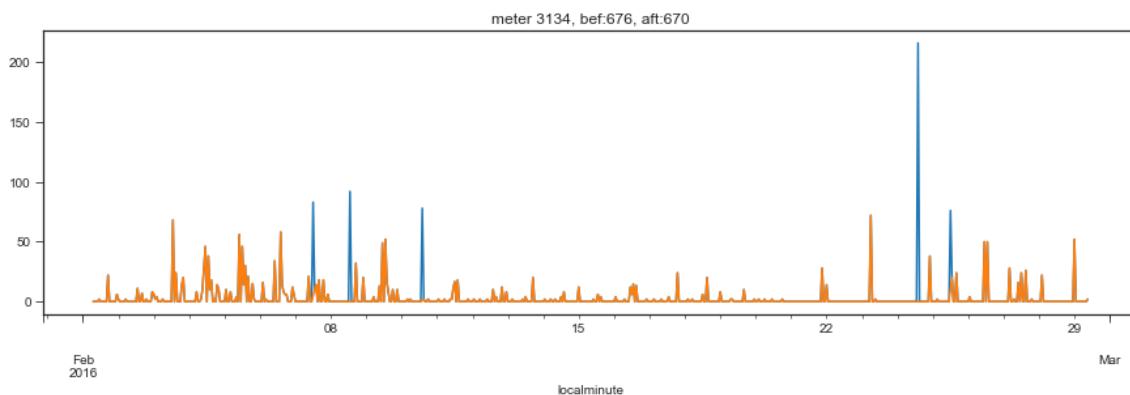
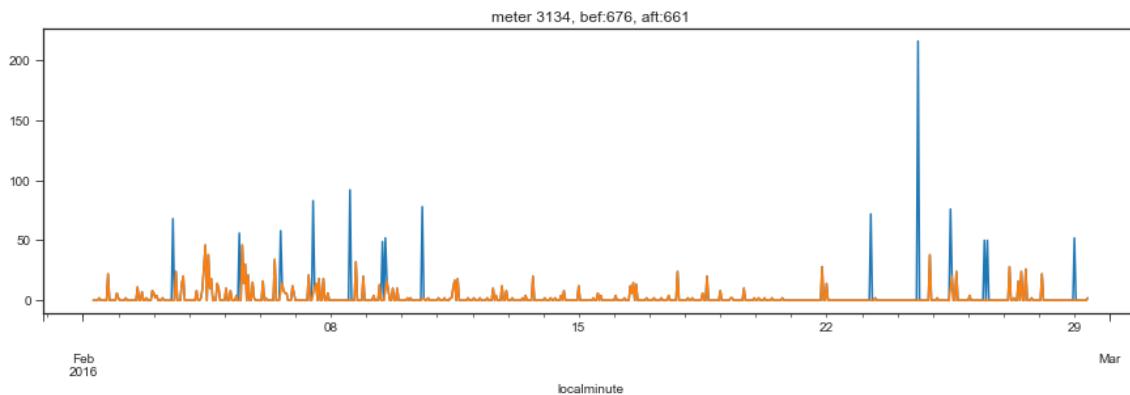
**8910**

meter 2335, bef:690, aft:688

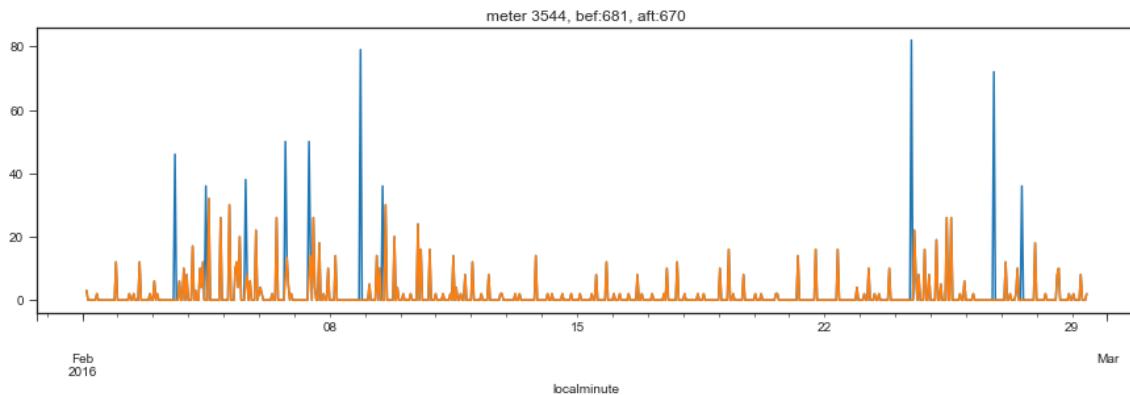
**5449**

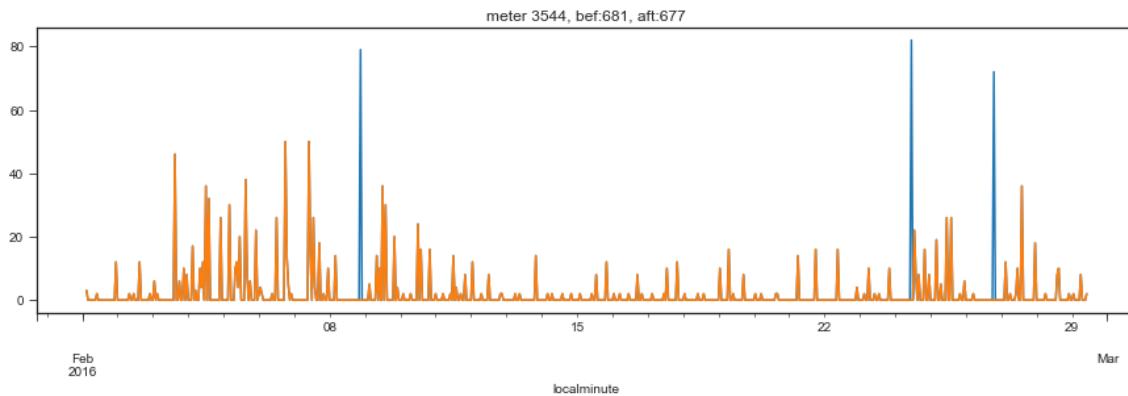


4017

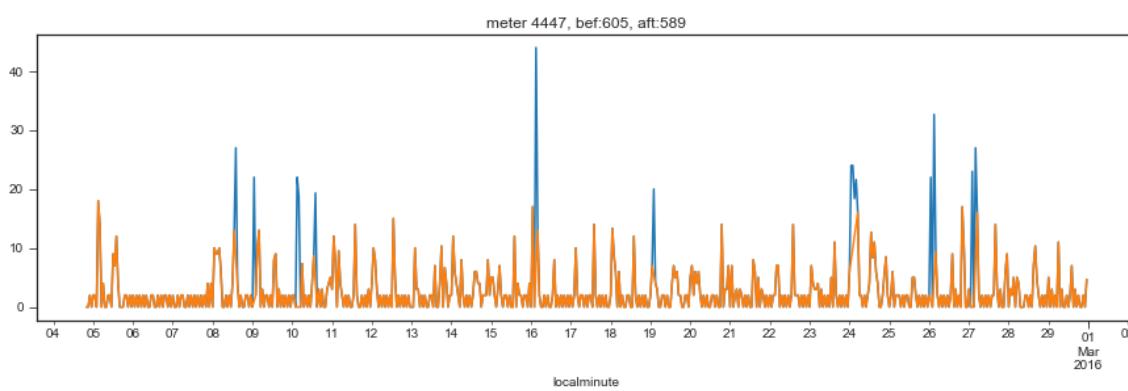
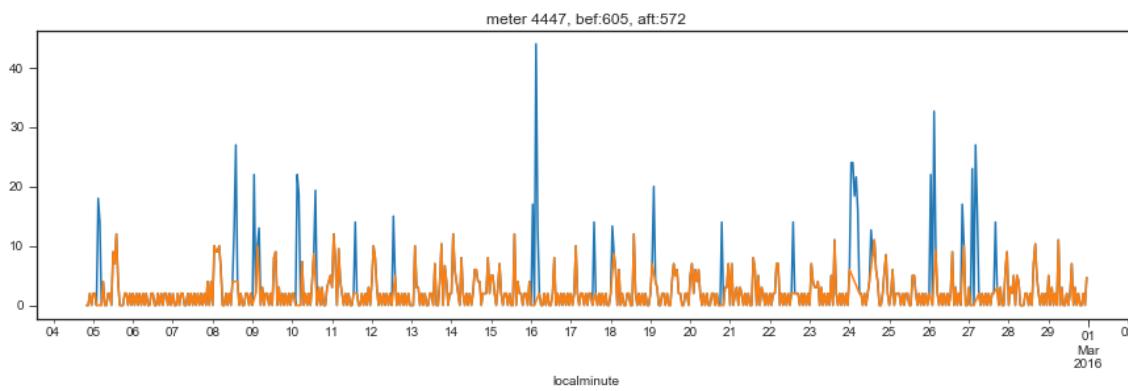


2221

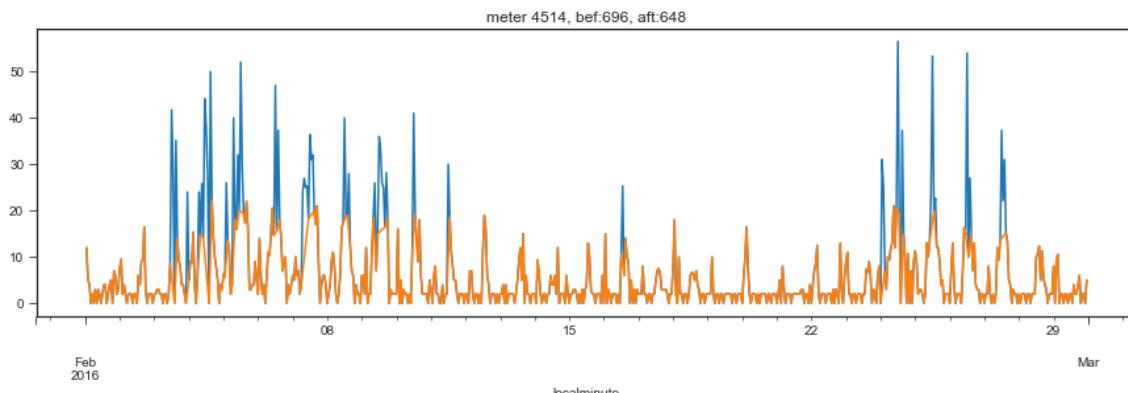


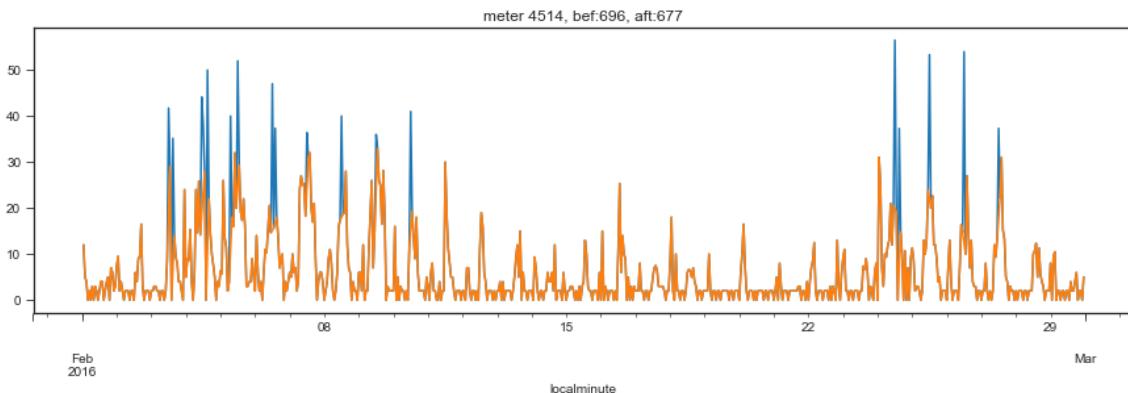


9158

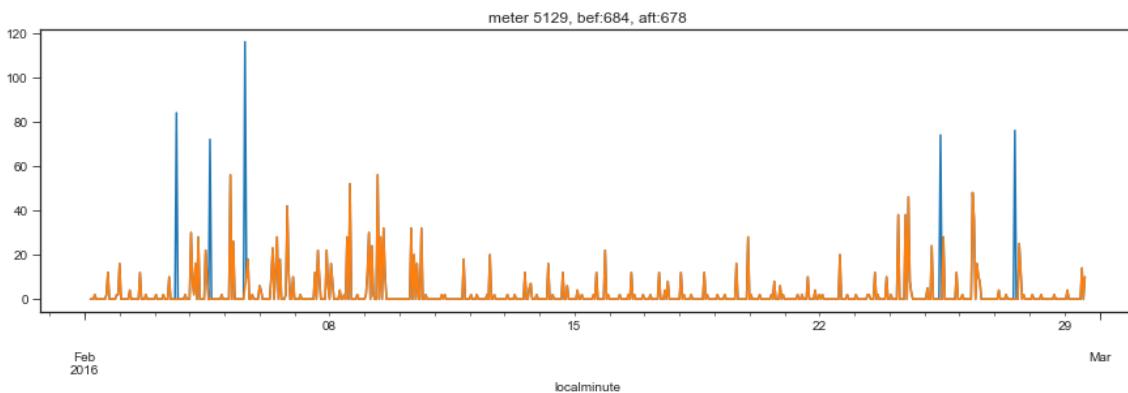
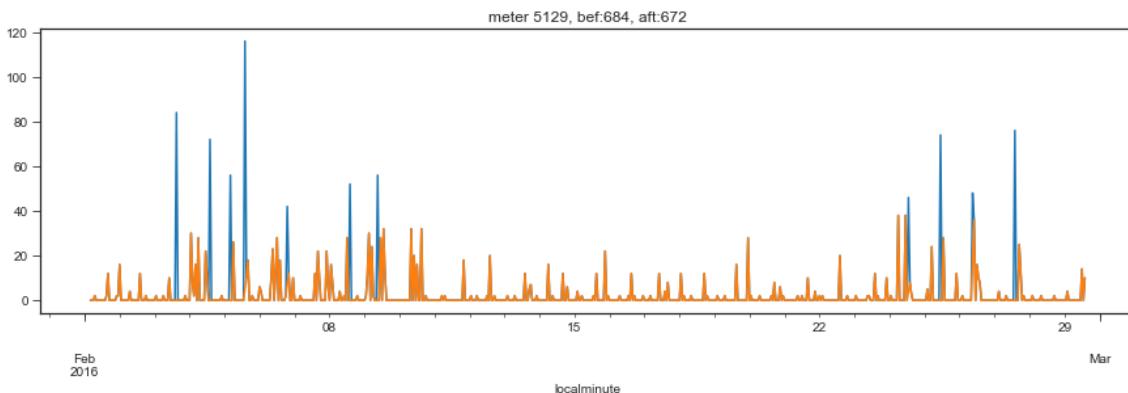


19074

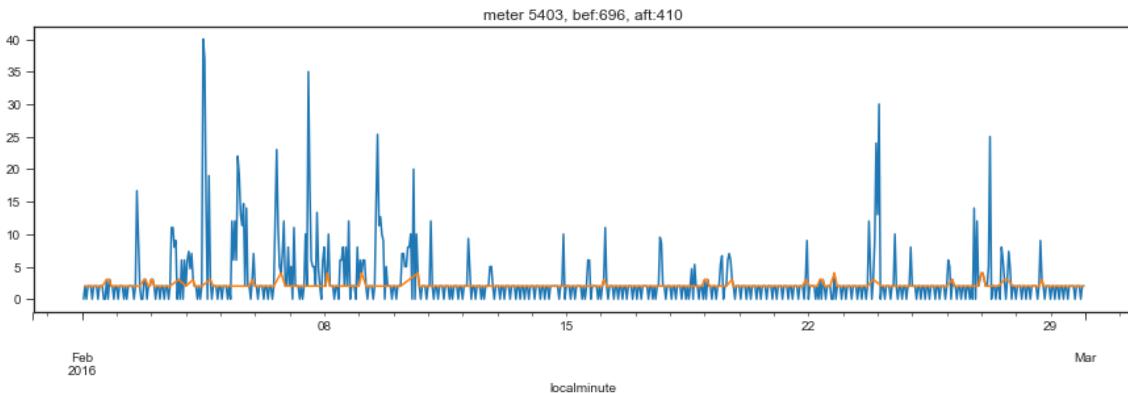


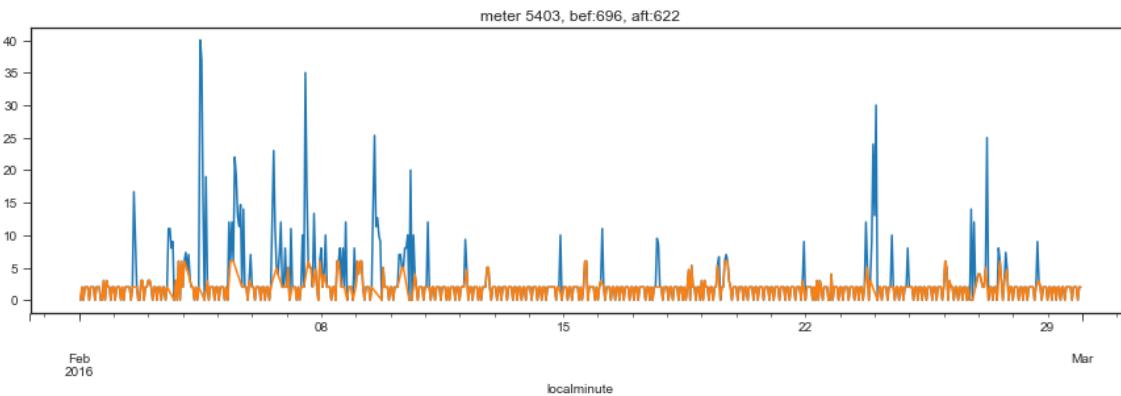


4486

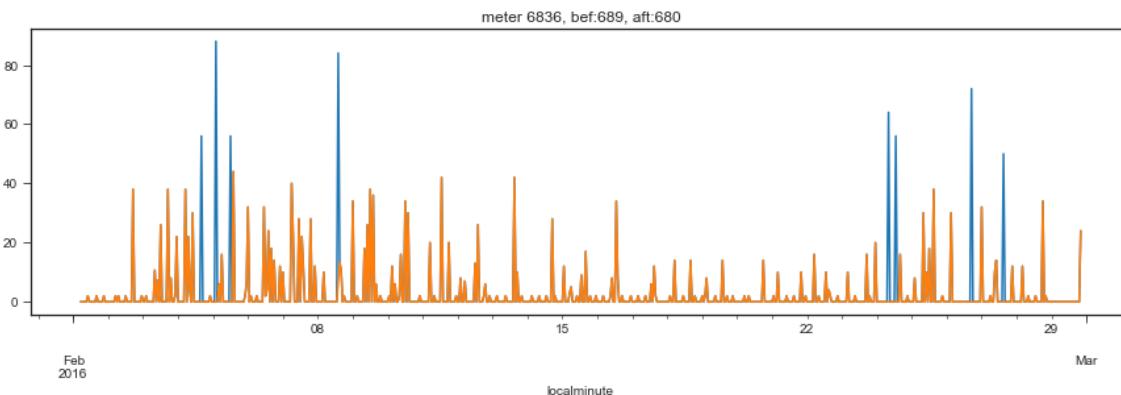


25559

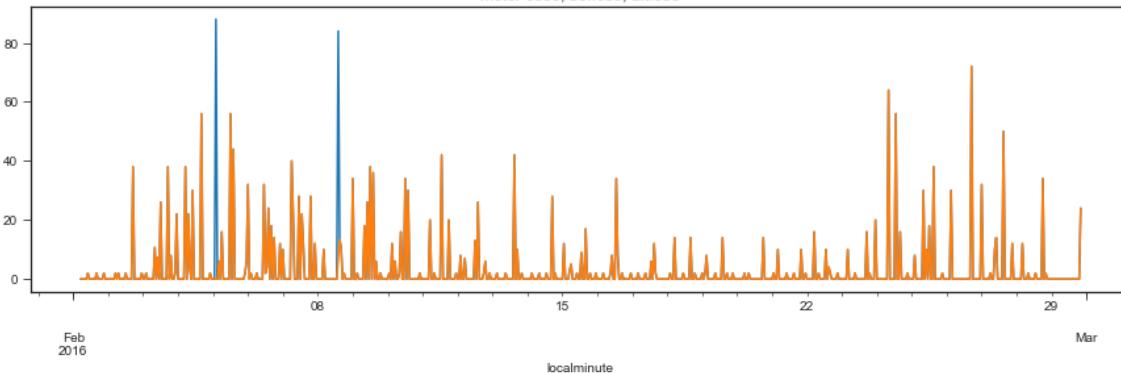




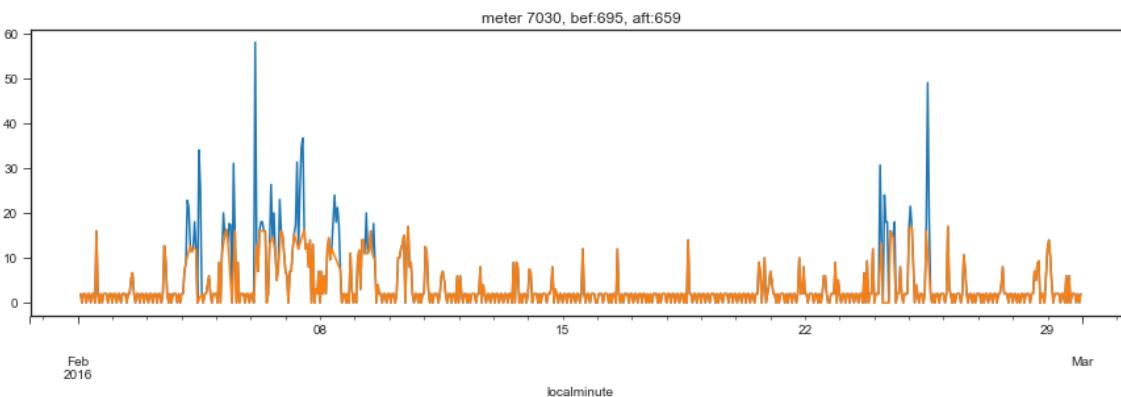
4520

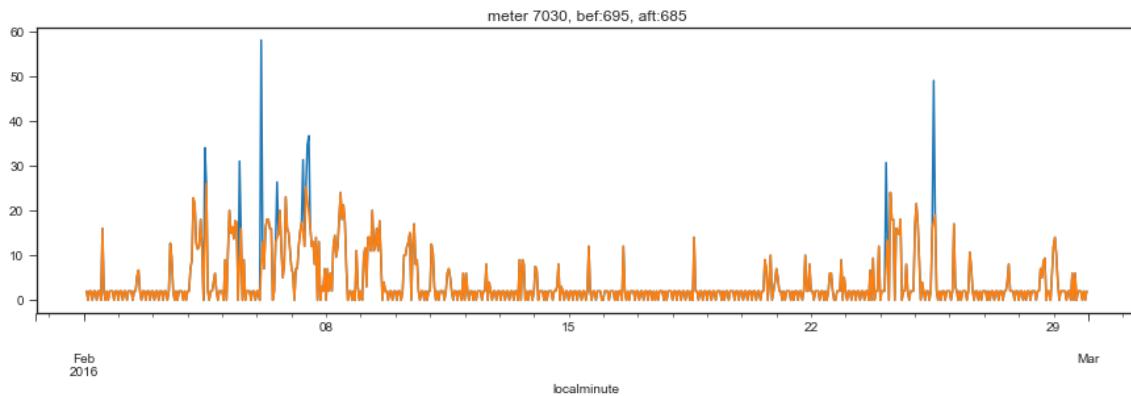


meter 6836, bef:689, aft:686

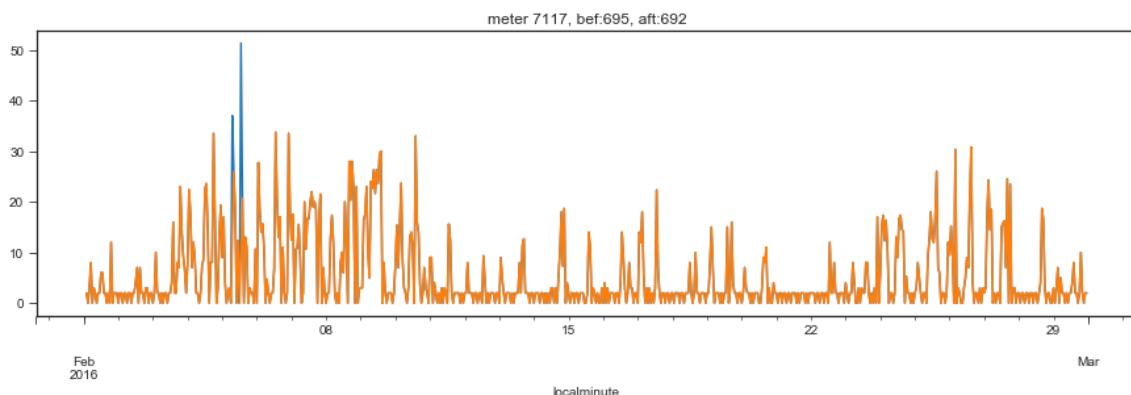
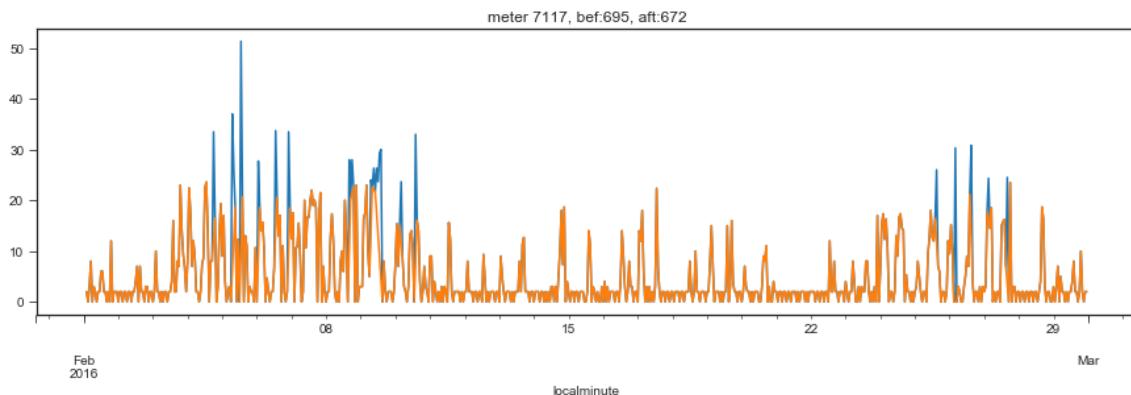


17915

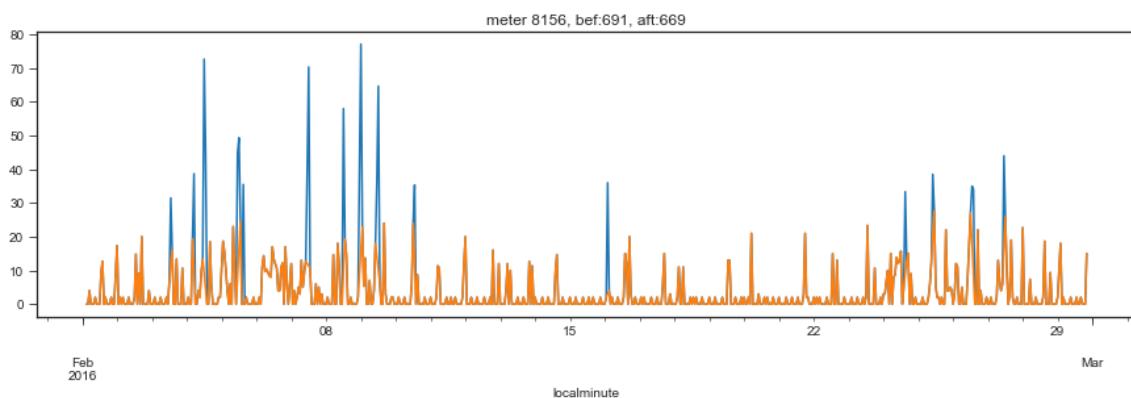


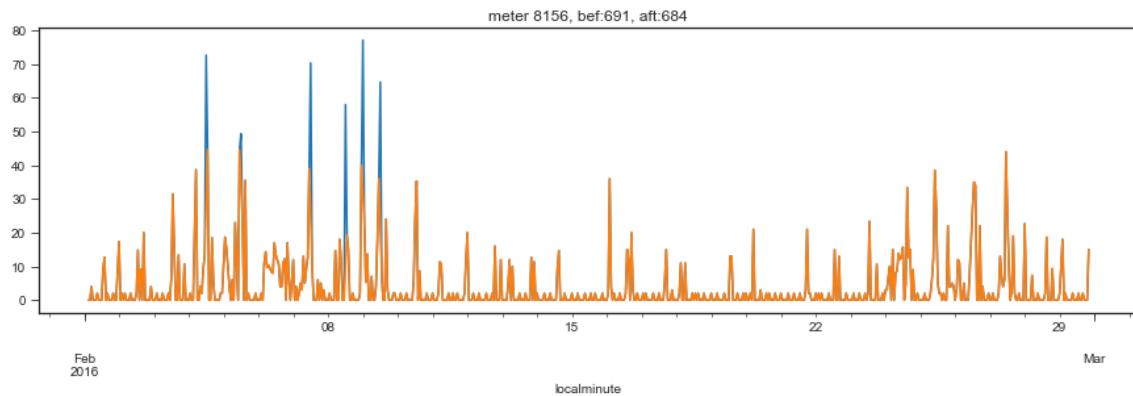
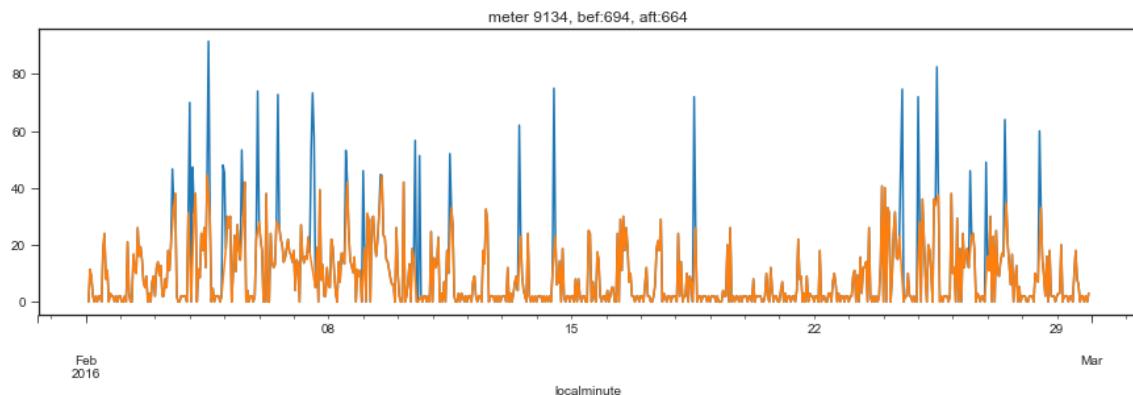


20493

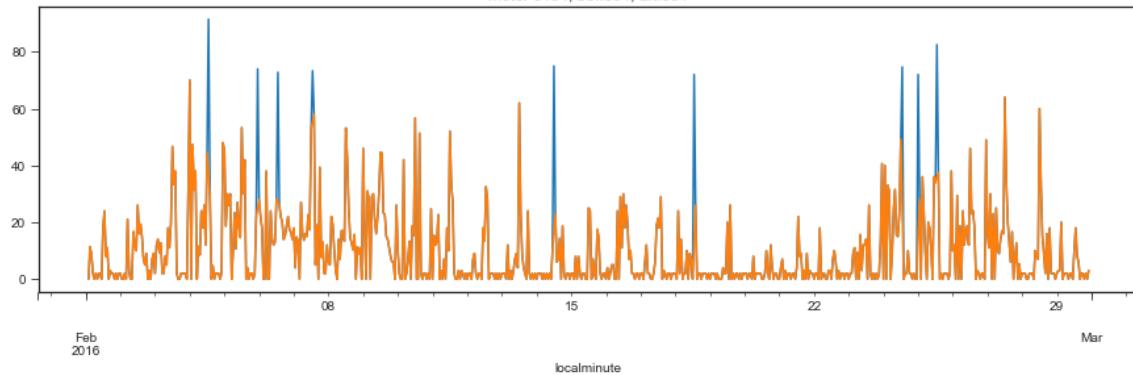
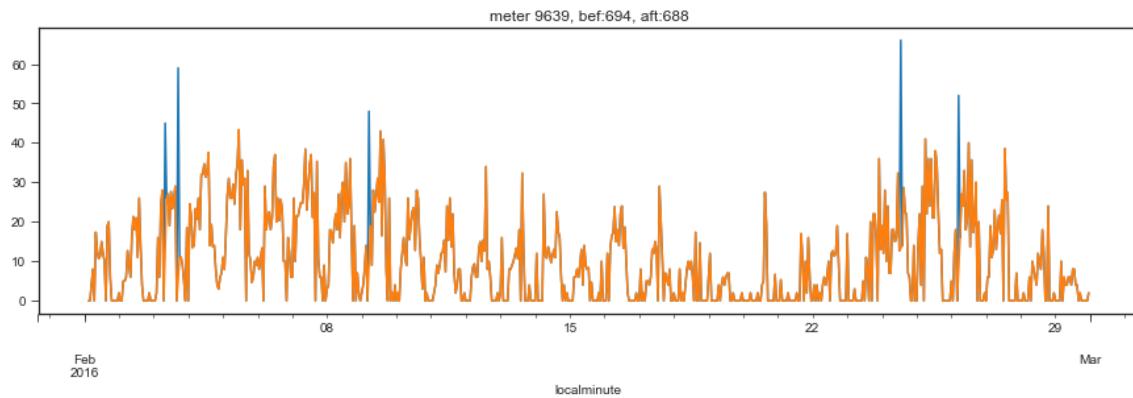


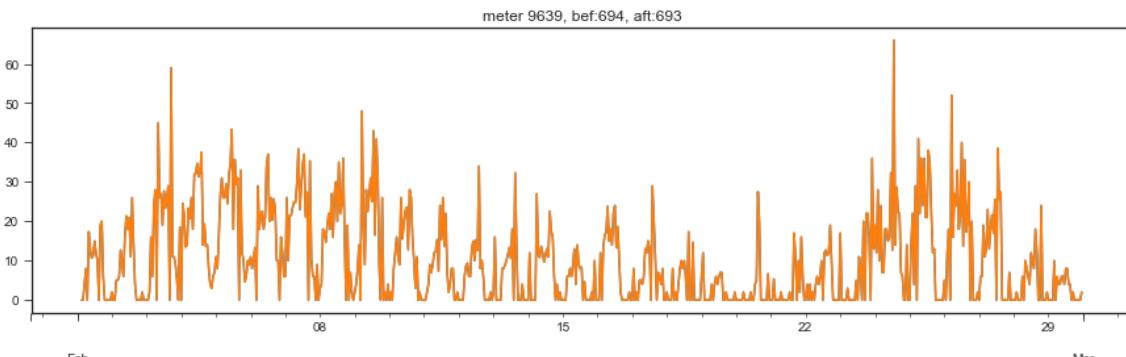
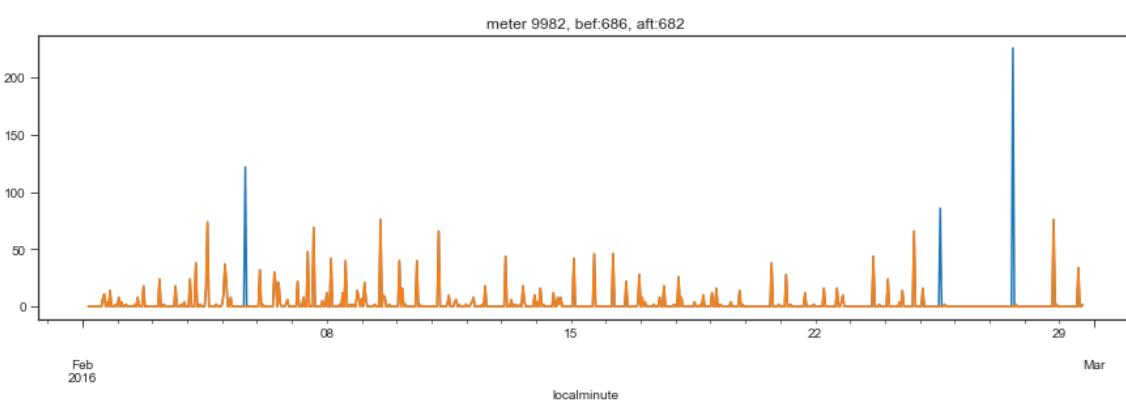
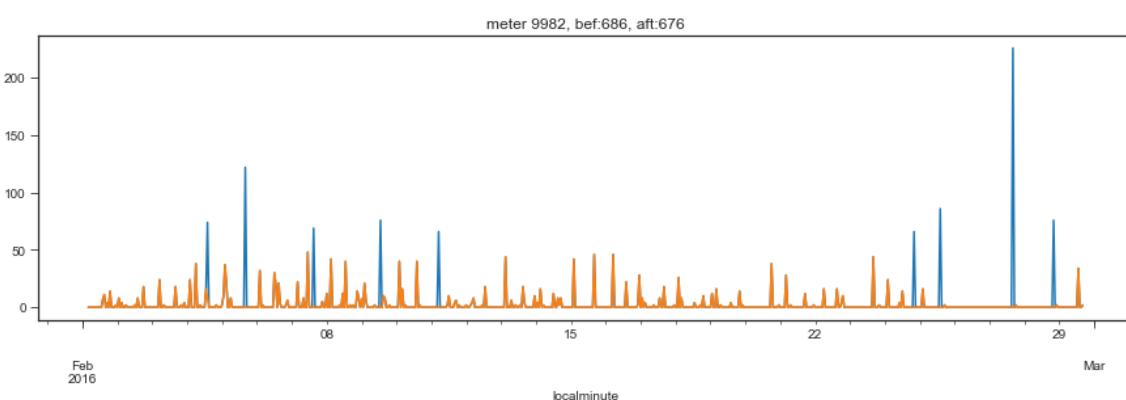
25296



**14064**

meter 9134, bef:694, aft:684

**13796**

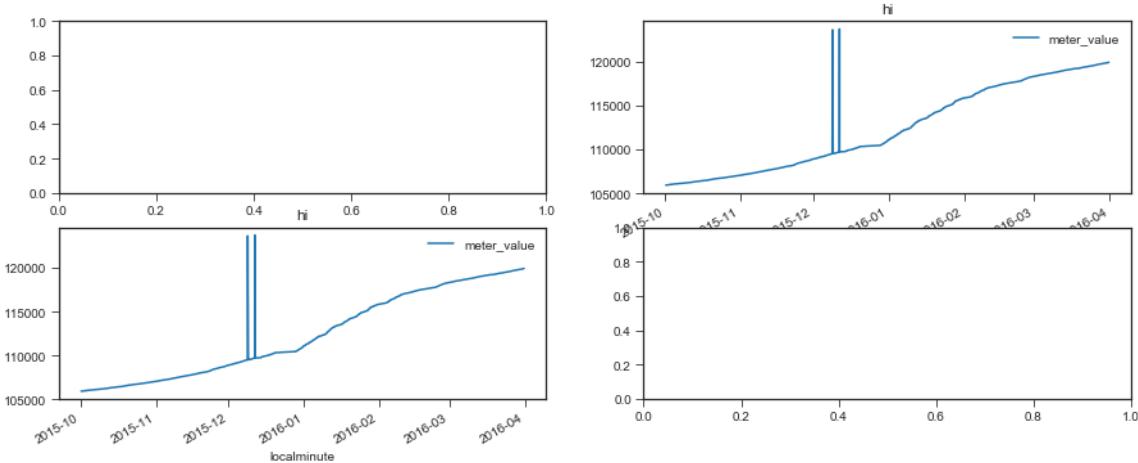
**1540**

&lt;Figure size 432x288 with 0 Axes&gt;

In [241]:

```
# Plotting test
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,6))

df_i.plot(ax=axes[0,0], title='hi')
df_i.plot(ax=axes[0,1], title='hi')
df_i.plot(ax=axes[1,0], title='hi')
axes[0,0].clear()
```



In [242]:

```
df_i.plot(ax=axes[0,0], title='hi')
df_i.plot(ax=axes[0,1], title='hi')
df_i.plot(ax=axes[1,0], title='hi')
```

Out[242]:

```
<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE9D7C6940>
```

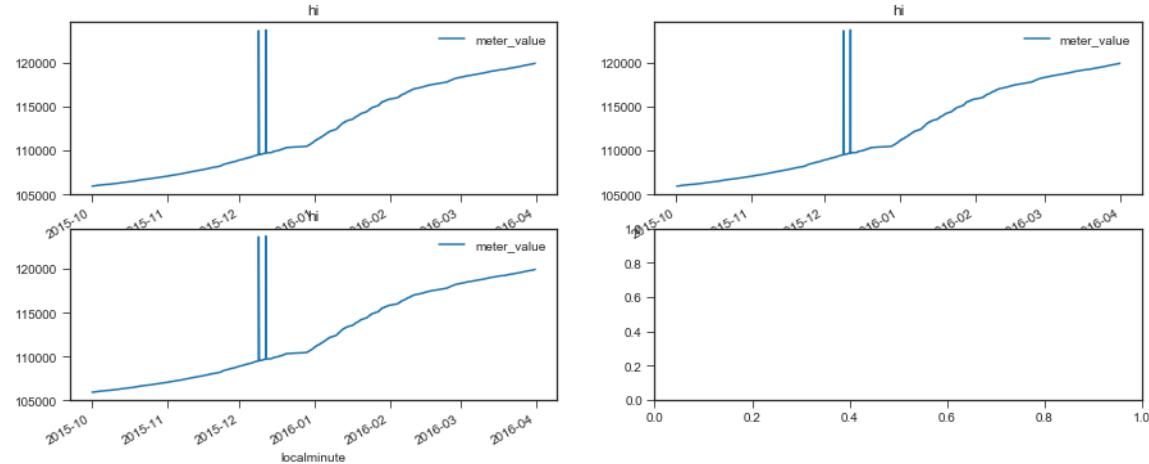
In [243]:

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,6))
# seems like I must call this again everytime i plot in a new cell?
# otherwise, no plot is shown when I call plot().

df_i.plot(ax=axes[0,0], title='hi')
df_i.plot(ax=axes[0,1], title='hi')
df_i.plot(ax=axes[1,0], title='hi')
```

Out[243]:

```
<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE95EEC9E8>
```



In [244]:

```
# zoom into 1 month
key = 1185
df_i = groups.get_group(key).drop(columns='dataid')
df_i_1mth = zoom(df_i, '2016-01-01', '2016-02-01')

def clean_resample_filter2(key, df_i):
    # v2 of this method uses/tests multiple plots
    # pre-condition, df_i has has numerical data to plot.

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15,3))

    # first, clean data by removing obvious dirt
    df_i_clean = remove_mal_data(df_i_1mth)
    df_i_clean = remove_negative_marginal(df_i_clean)

    # next, resample data by hour.
    df_i_resampled = df_i_clean.drop(columns='marginal_change').resample('H').mean().ffill()
    #display(df_i_resampled.head())

    # then, obtain new marginal changes.
    df_i_resampled['marginal_change'] = df_i_resampled['meter_value'].diff()

    # finally, filter by the IQR rule on marginal_change.
    # 1.5IQR filter
    df_i_filtered15 = remove_outliers(df_i_resampled, 1.5)

    # plot
    df_i_resampled['marginal_change'].plot(ax=axes[0])
    if len(df_i_filtered15) > 0:
        df_i_filtered15['marginal_change'].plot(ax=axes[0], \
                                                title=str(f'meter {key}, 1.5IQR, bef:{len(df_i_resampled)}, aft:{len(df_i_filtered15)}'))
    # 3.0IQR filter
    df_i_filtered30 = remove_outliers(df_i_resampled, 3.0)

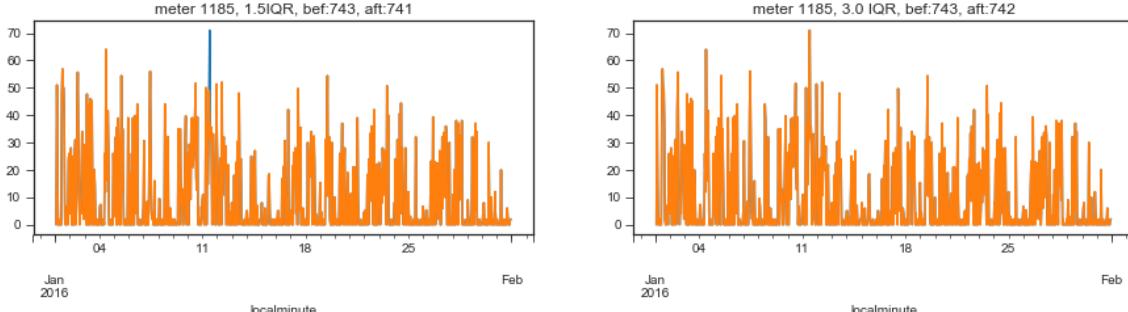
    # plot
    df_i_resampled['marginal_change'].plot(ax=axes[1])
    if len(df_i_filtered30) > 0:
        df_i_filtered30['marginal_change'].plot(ax=axes[1], \
                                                title=str(f'meter {key}, 3.0 IQR, bef:{len(df_i_resampled)}, aft:{len(df_i_filtered30)}'))

clean_resample_filter2(key, df_i_1mth)
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

```
if __name__ == '__main__':
```



In [245]:

```
start = '2016-01-01'
end = '2016-02-01'

lack_data_for_monthx = []
for key in keys_list:
    df_i = groups.get_group(key).drop(columns='dataid')
    df_i_1mth = zoom(df_i, start, end)

    if len(df_i_1mth) > 1:
        df_i_counts = get_month_counts(df_i_1mth)
        if df_i_counts['count'].values[0] < 3:
            lack_data_for_monthx.append(key)
    else:
        lack_data_for_monthx.append(key)
print(lack_data_for_monthx)
```

```
[2638, 3039, 4352, 4421, 4447, 4671, 4874, 7460, 7919, 8467, 9474]
```

In [256]:

```
# malfunctioning and sparse readings
to_remove = [1185, 1556, 2335, 2449, 3134, 3544, 4447, 4514,\n
              5129, 5403, 6836, 7030, 7117, 8156, 9134, 9639, 9982]

to_remove.extend(lack_data_for_monthx)

display(to_remove, len(to_remove))
```

```
[1185, 1556, 2335, 2449, 3134, 3544, 4447, 4514, 5129, 5403, 6836, 7030, 7\n117, 8156, 9134, 9639, 9982, 2638, 3039, 4352, 4421, 4447, 4671, 4874, 746\n0, 7919, 8467, 9474]
```

In [257]:

```
to_keep = [elem for elem in keys_list if elem not in to_remove]

for key in to_keep:
    df_i = groups.get_group(key).drop(columns='dataid')
    df_i_1mth = zoom(df_i, start, end)
    clean_resample_filter2(key, df_i_1mth)
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

```
if __name__ == '__main__':
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\matplotlib\pyplot.py:537: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
max_open_warning, RuntimeWarning)
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\matplotlib\pyplot.py:537: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
max_open_warning, RuntimeWarning)
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame.

```
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

```
if __name__ == '__main__':
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\matplotlib\pyplot.py:537: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
max_open_warning, RuntimeWarning)
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame.

```
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

```
if __name__ == '__main__':
```

```
C:\Users\Melvin\Anaconda3\lib\site-packages\matplotlib\pyplot.py:537: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
max_open_warning, RuntimeWarning)
```

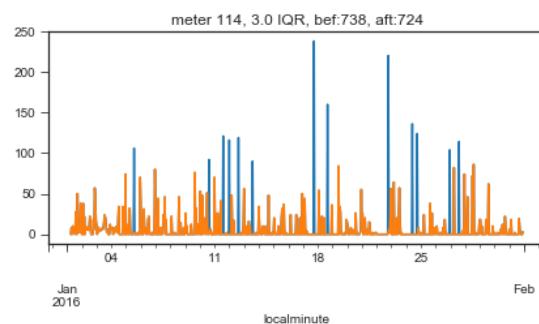
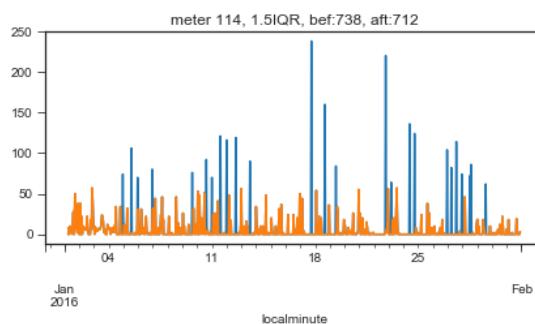
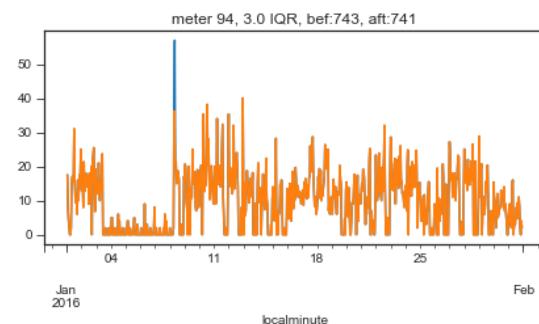
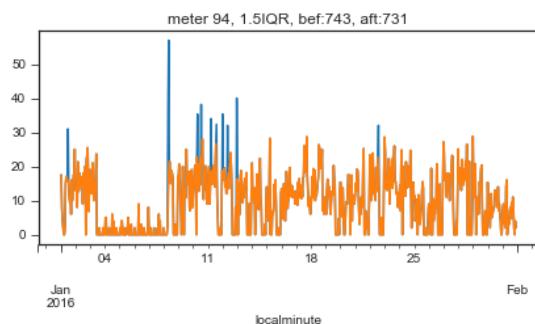
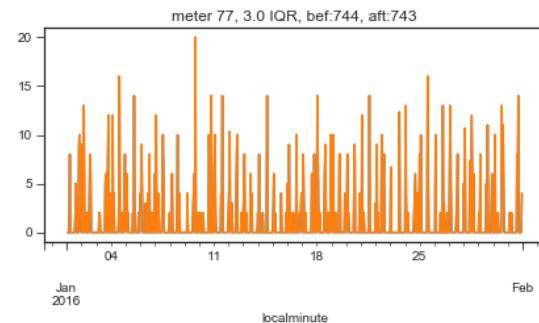
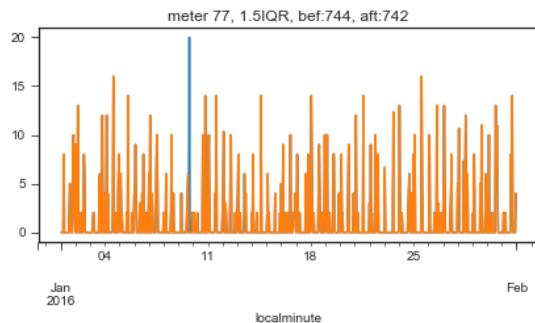
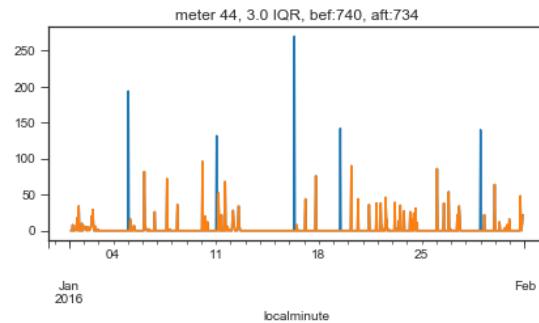
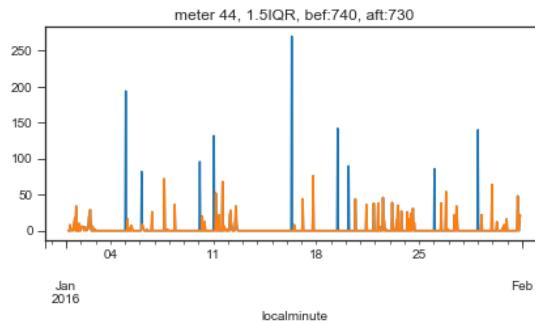
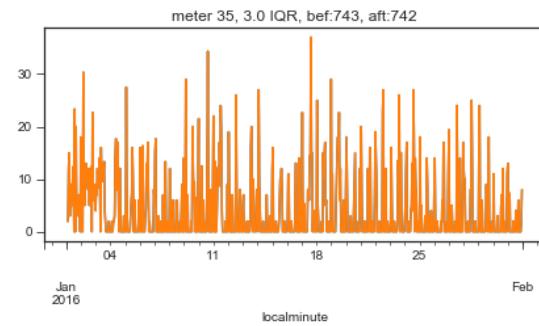
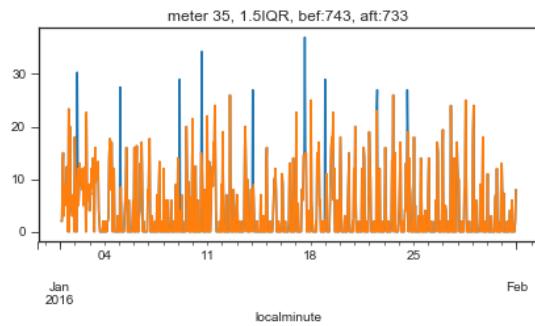
```
C:\Users\Melvin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
```

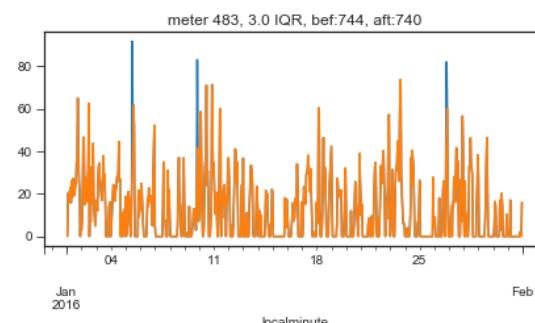
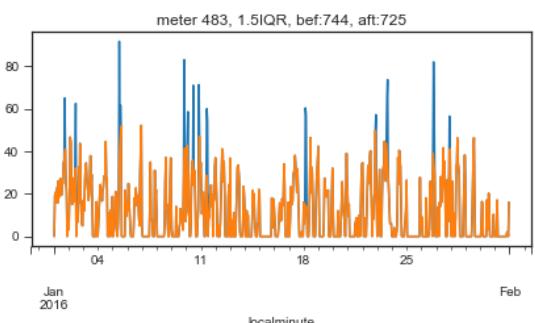
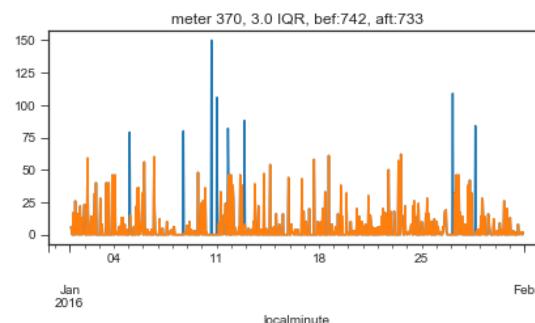
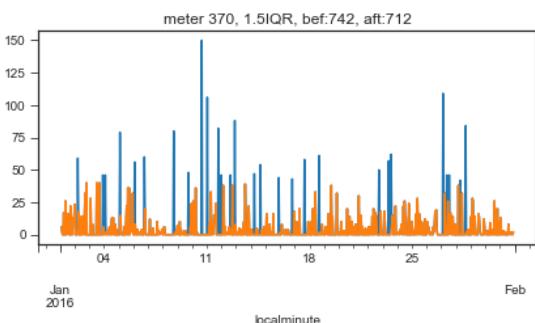
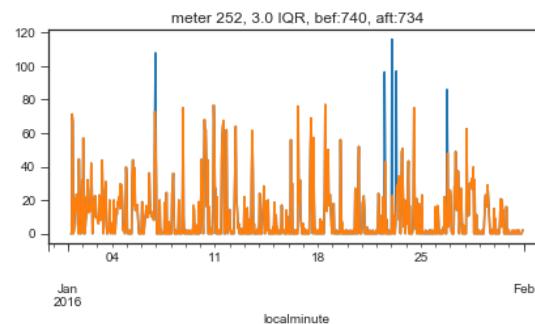
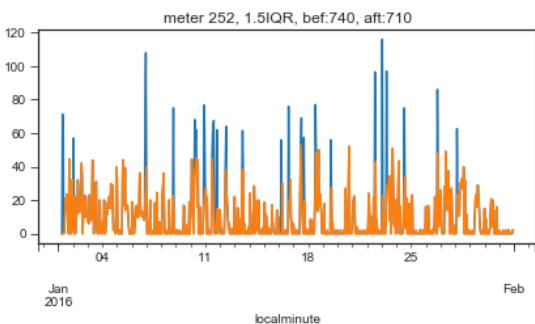
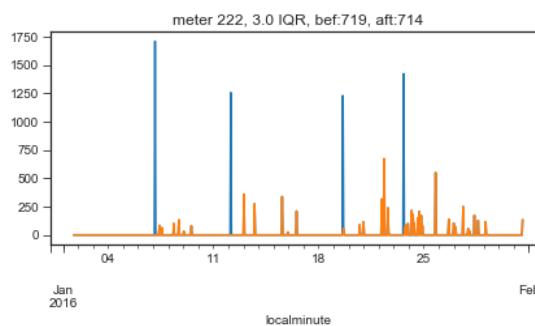
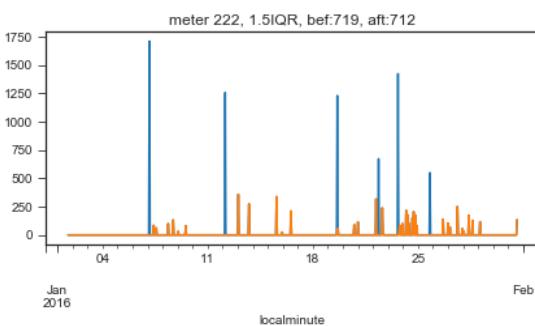
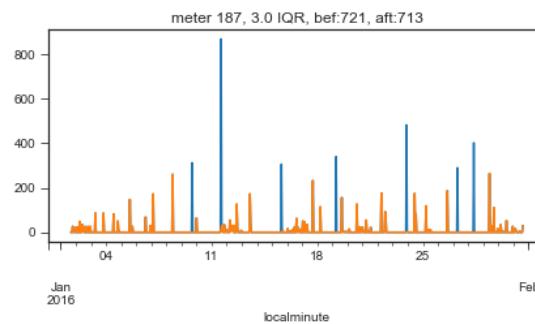
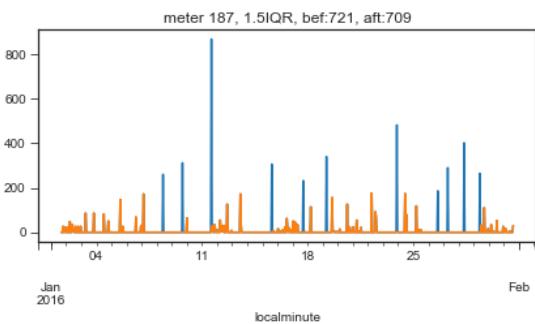
A value is trying to be set on a copy of a slice from a DataFrame.

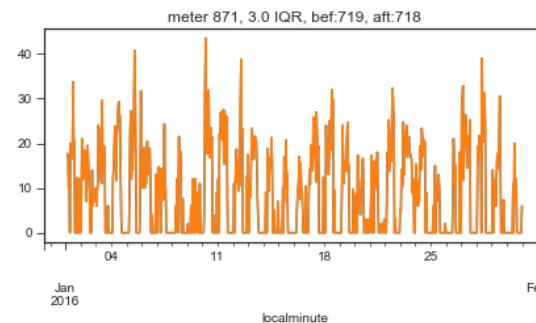
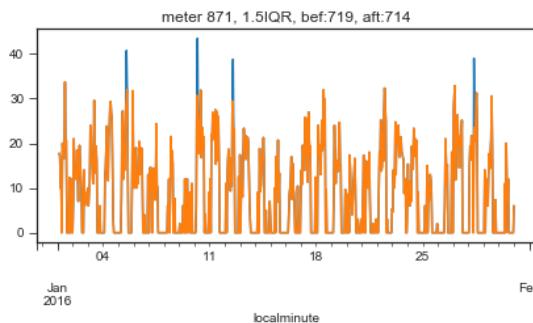
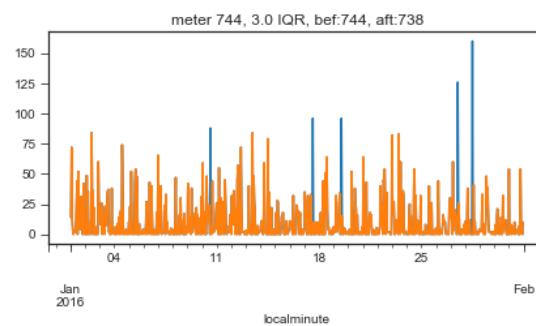
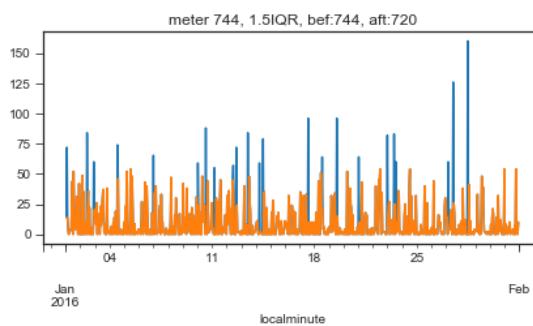
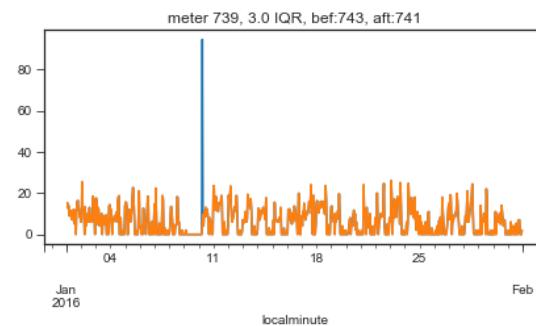
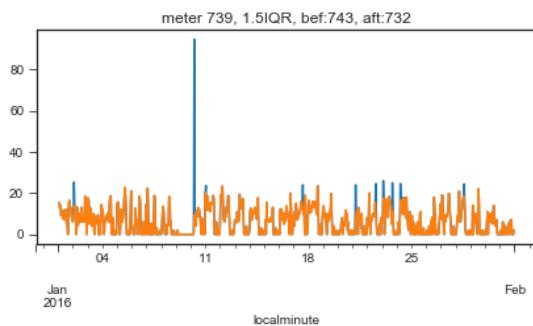
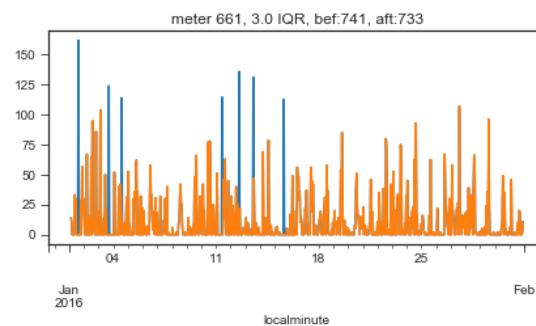
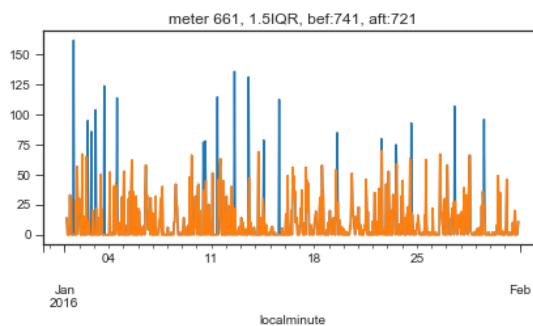
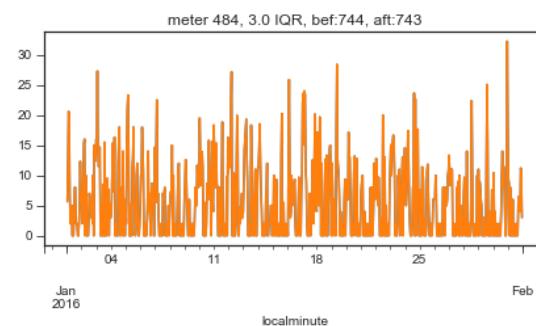
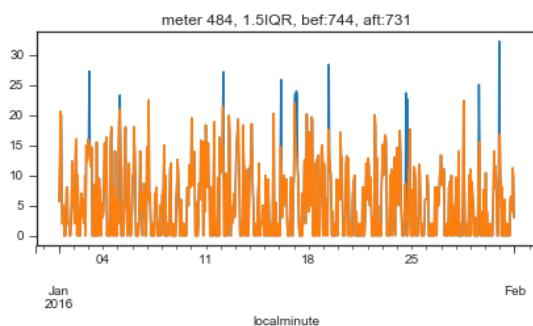
```
Try using .loc[row_indexer,col_indexer] = value instead
```

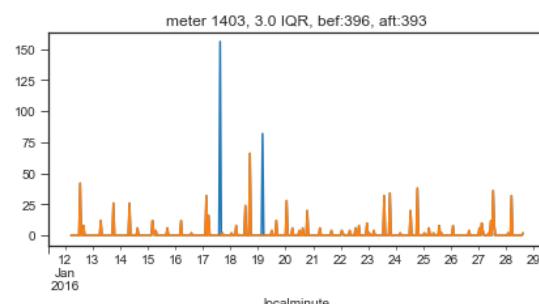
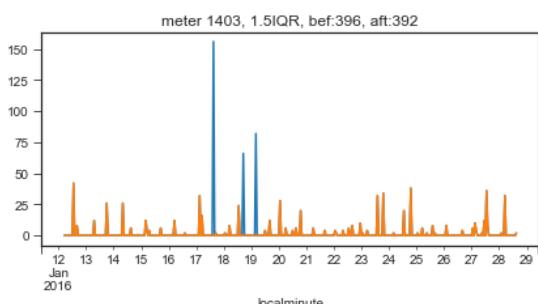
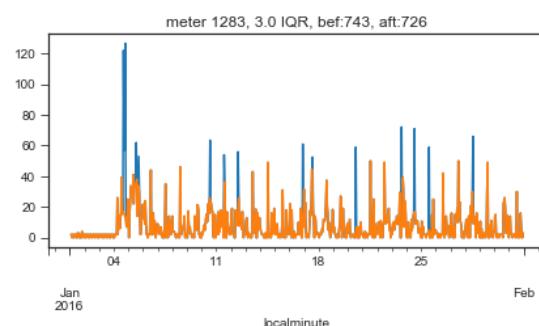
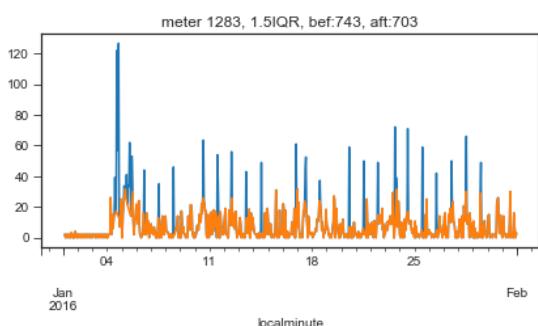
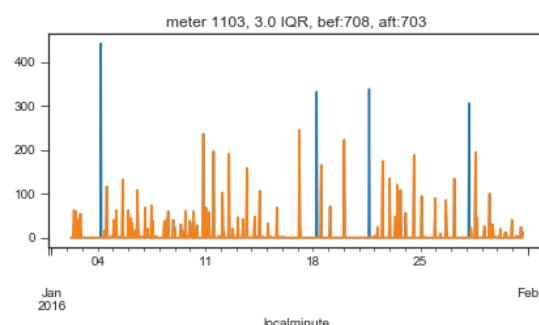
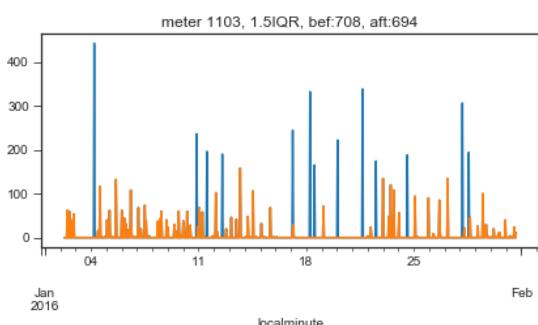
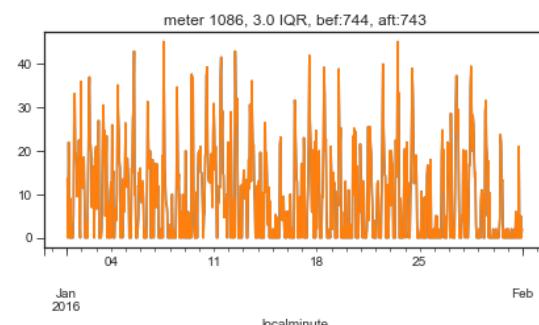
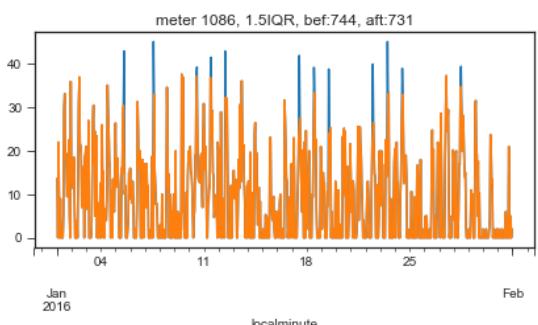
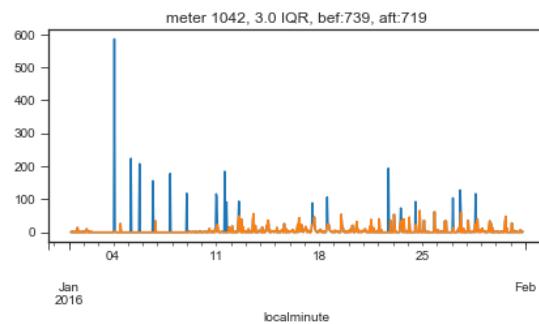
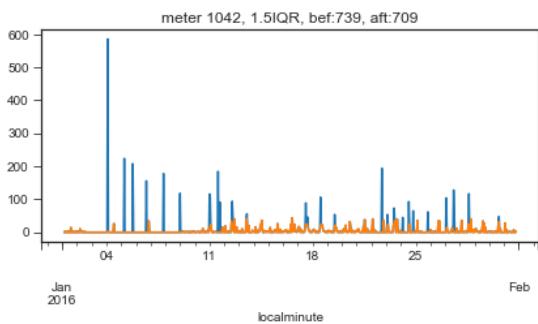
See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>

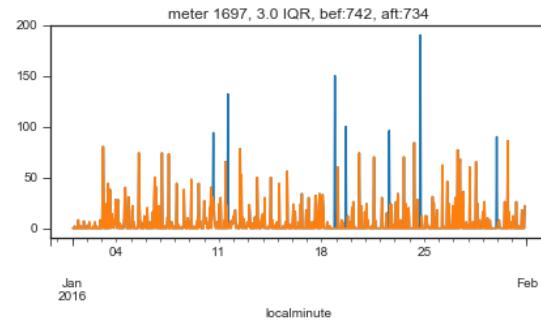
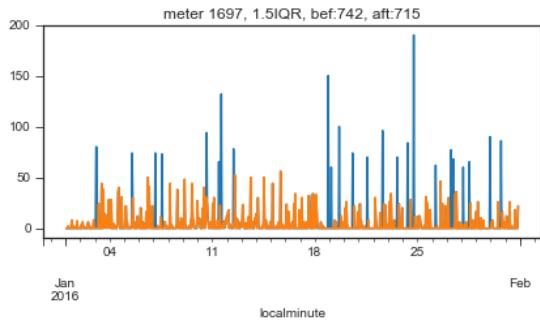
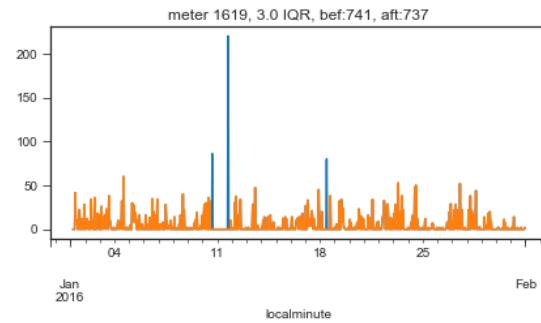
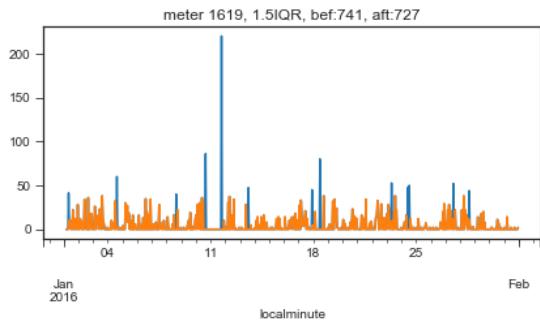
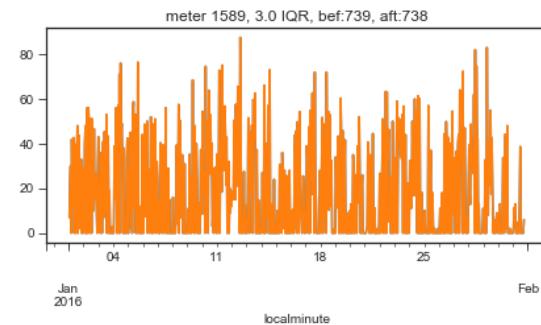
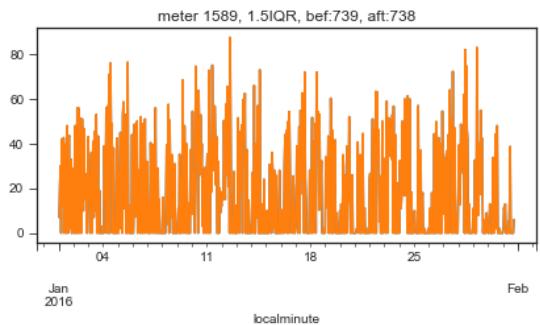
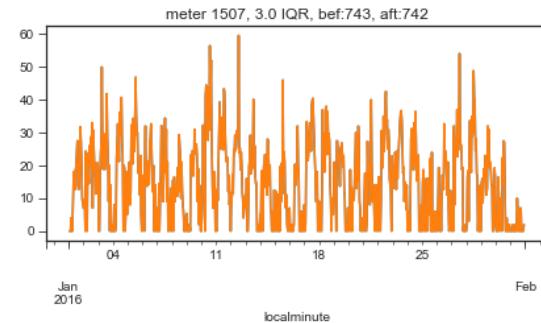
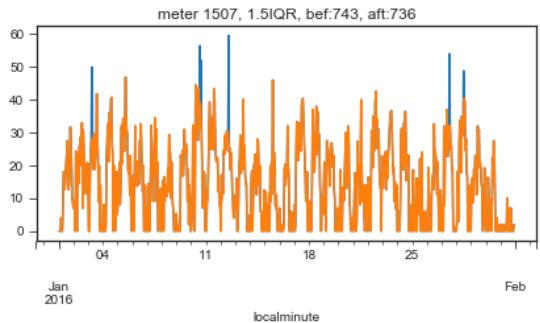
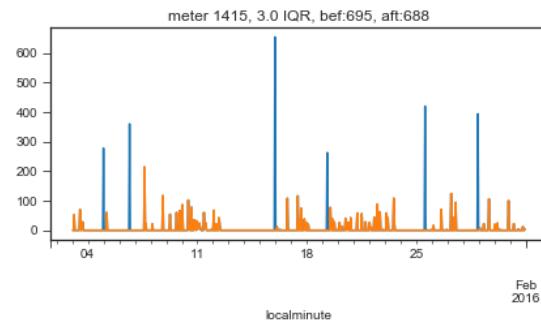
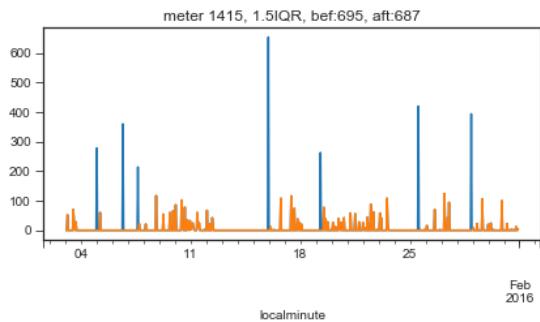
```
if __name__ == '__main__':
```

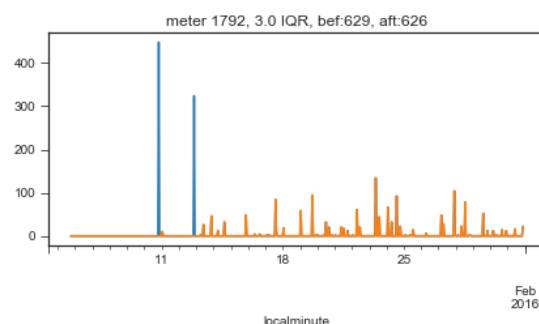
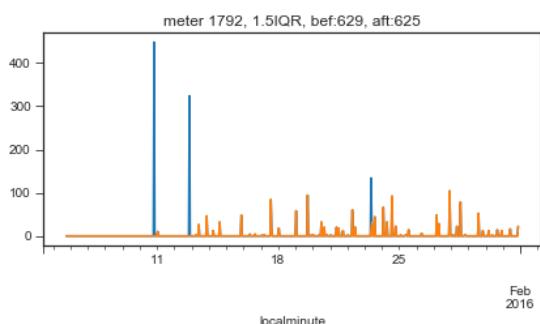
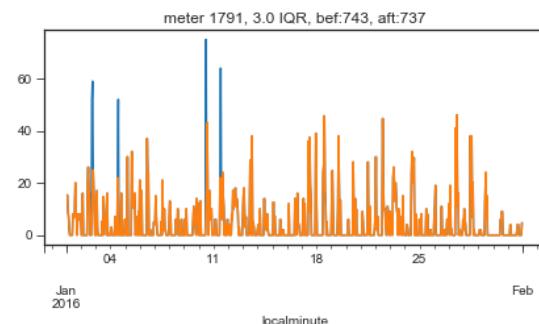
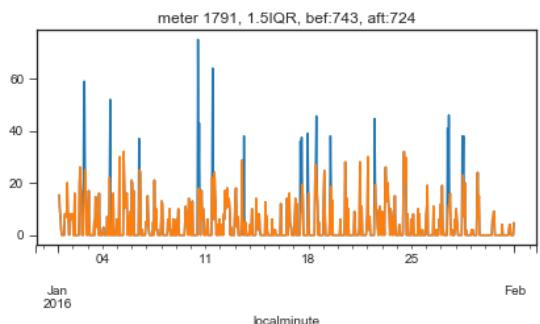
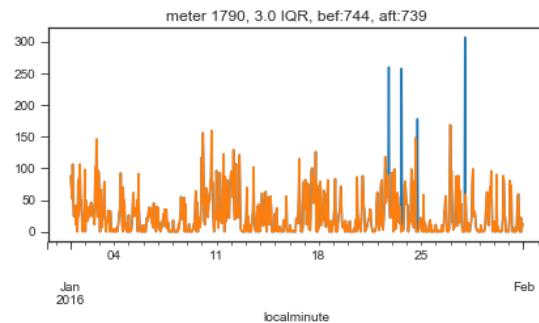
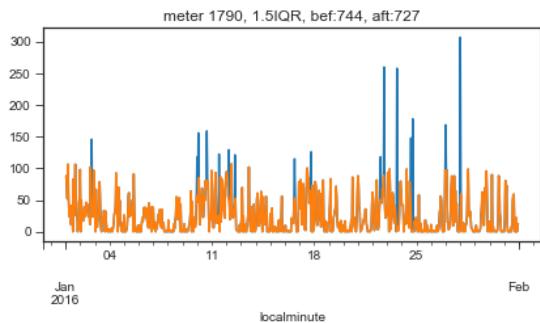
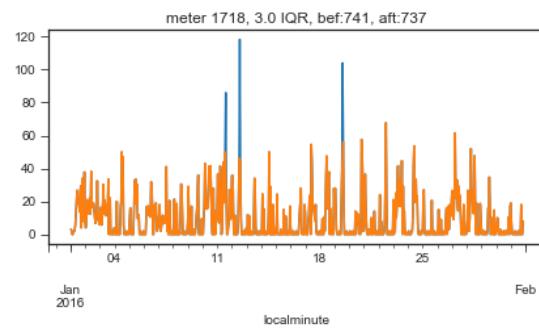
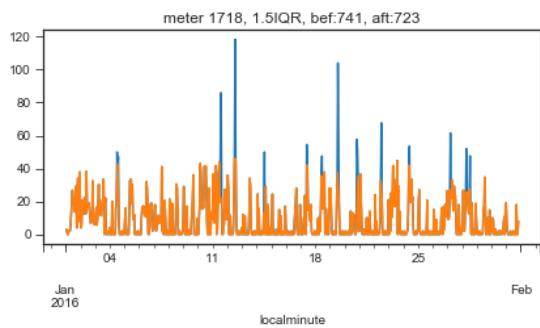
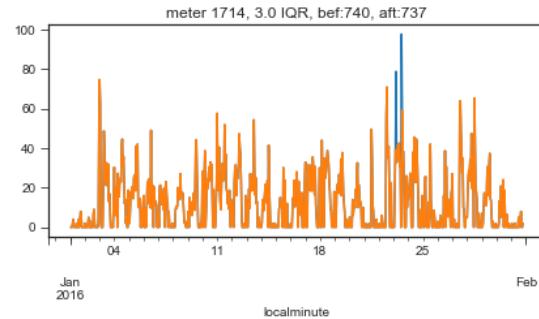
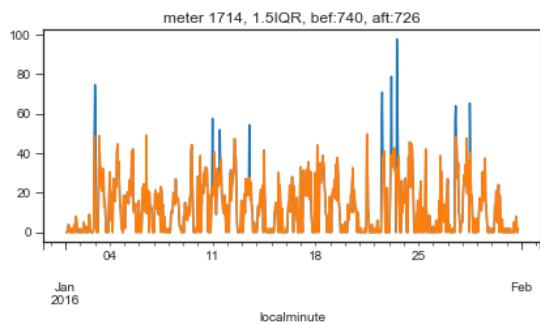


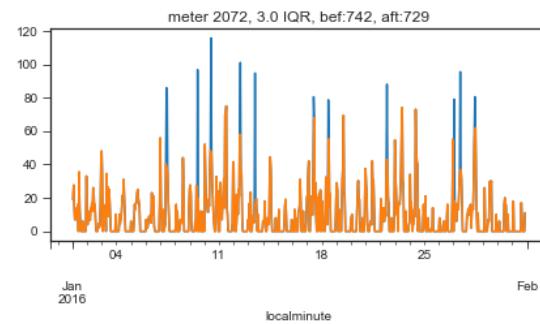
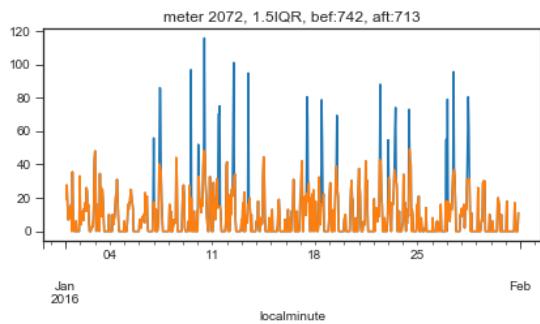
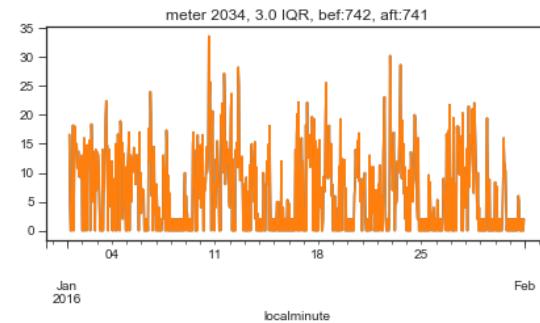
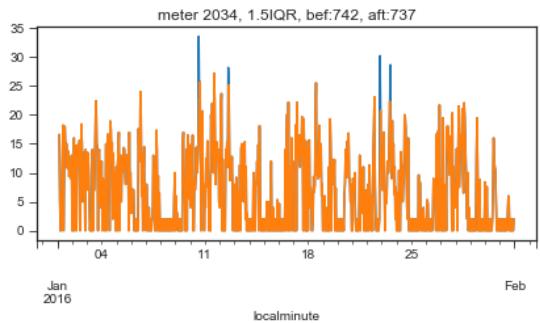
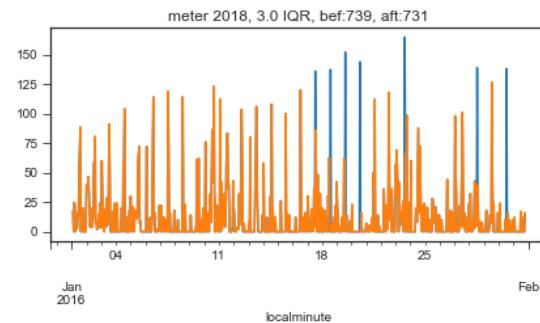
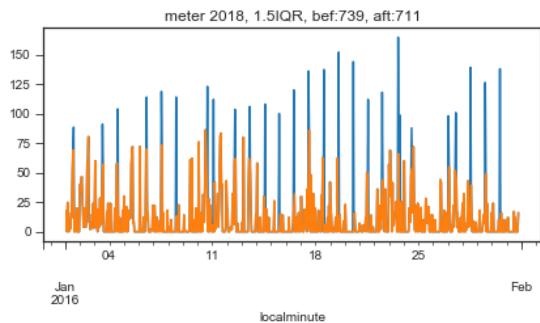
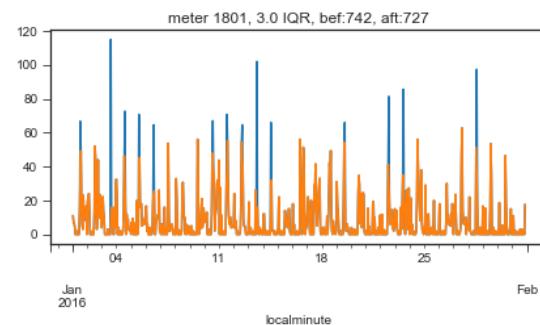
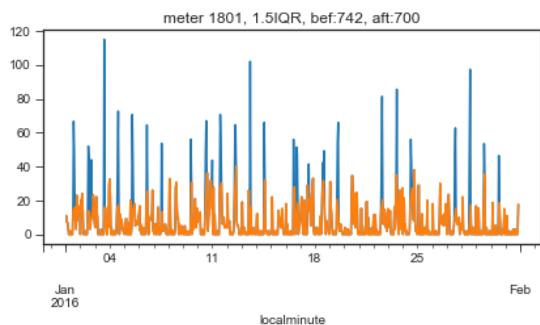
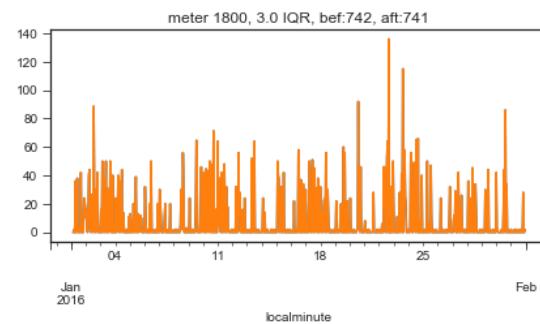
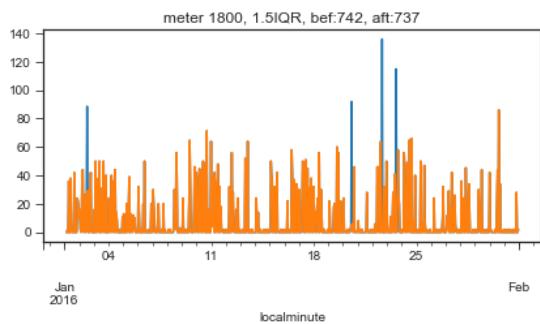




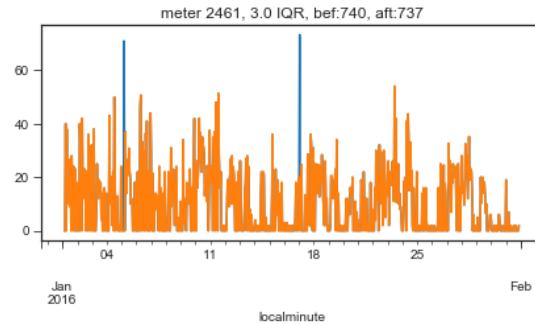
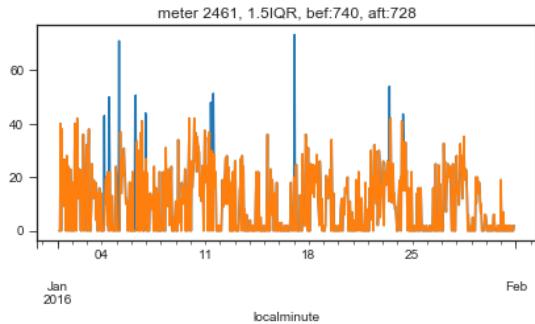
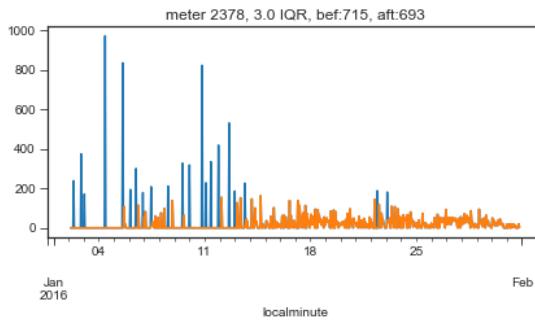
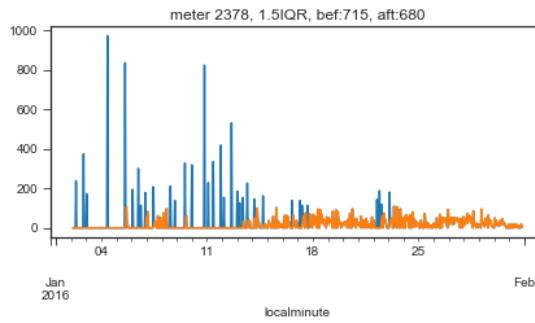
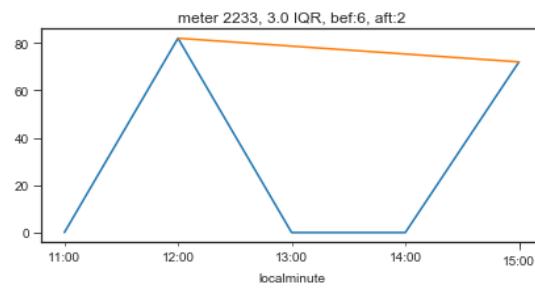
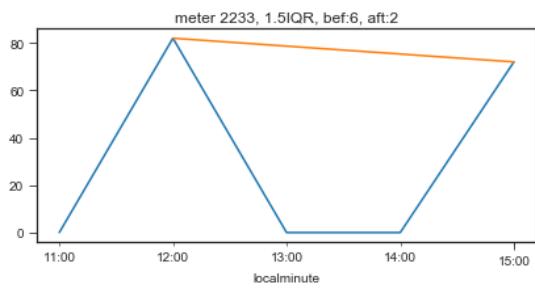
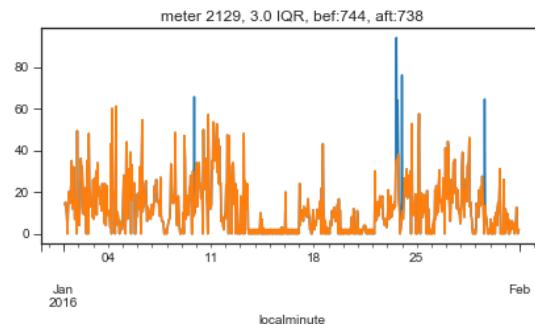
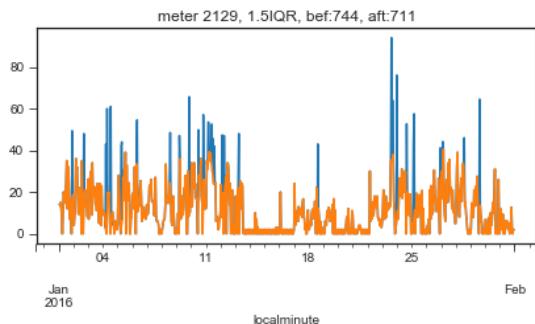
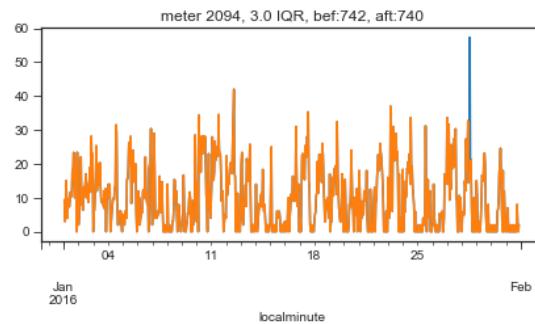
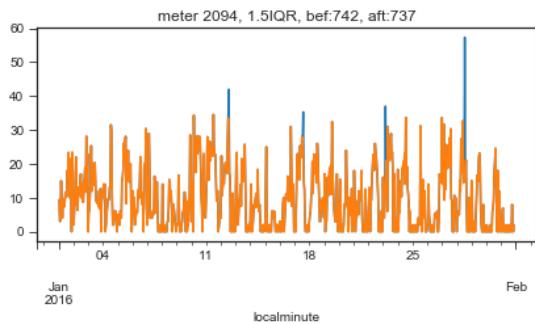


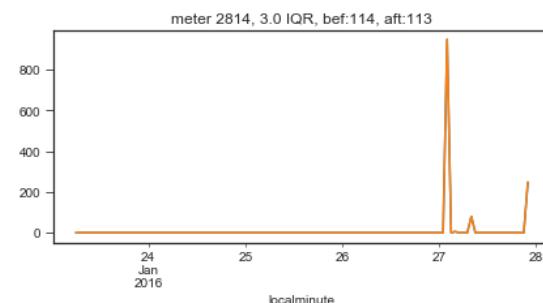
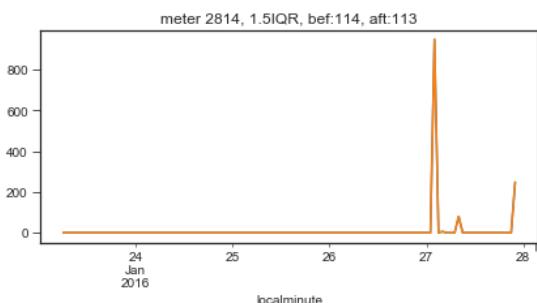
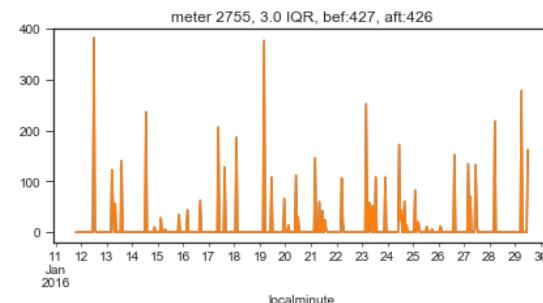
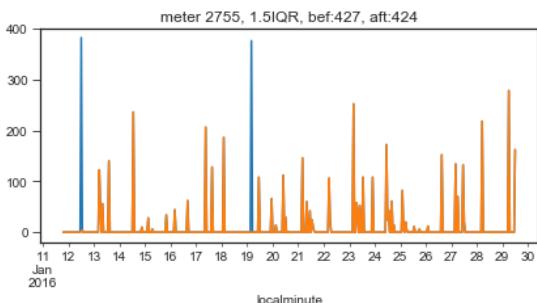
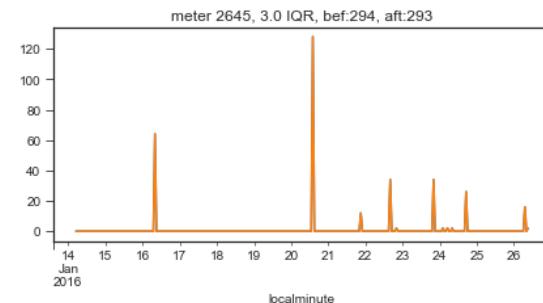
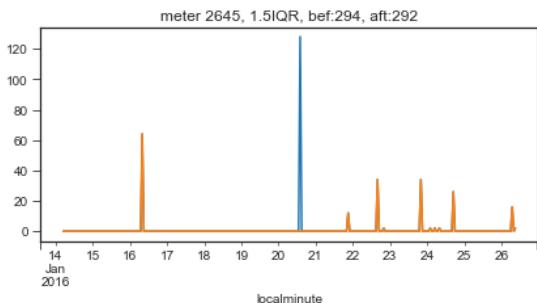
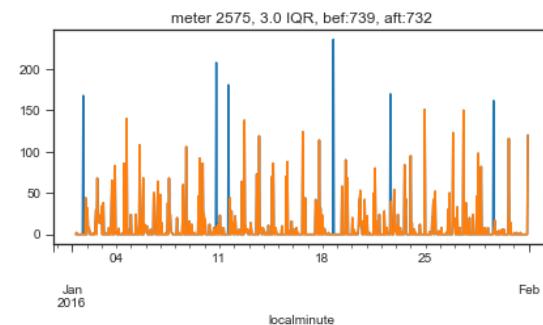
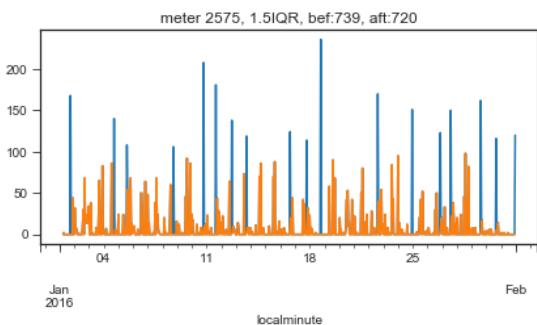
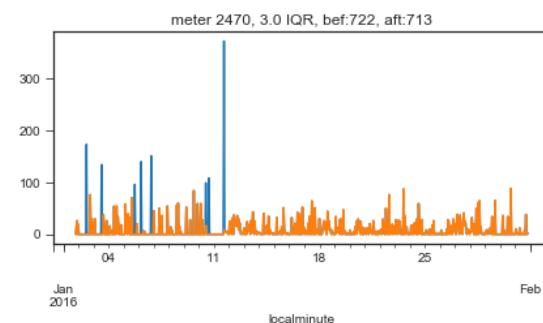
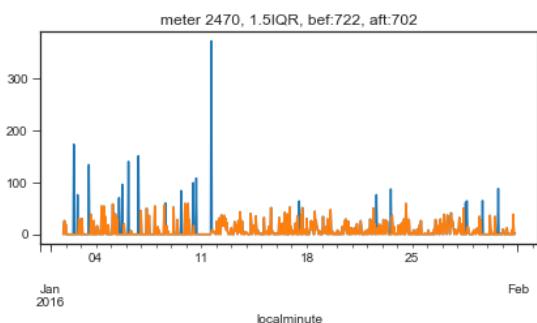


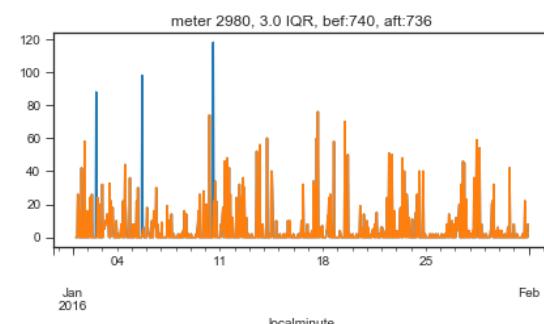
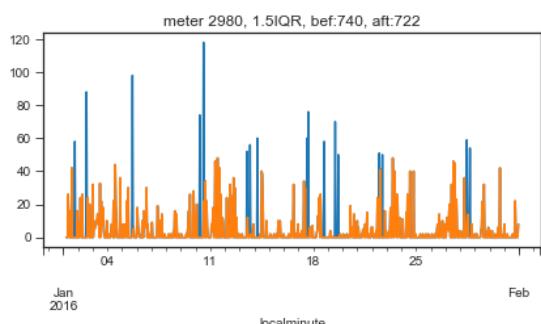
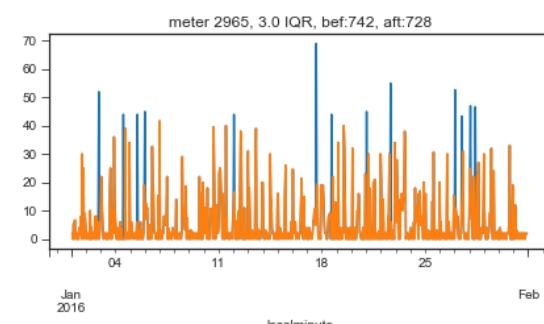
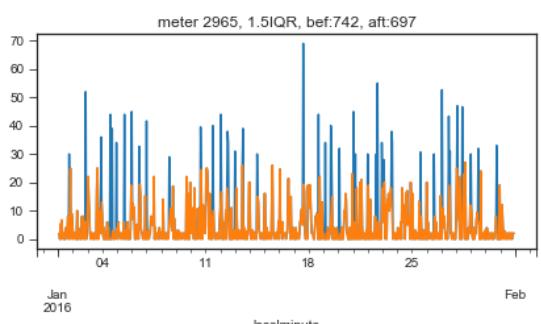
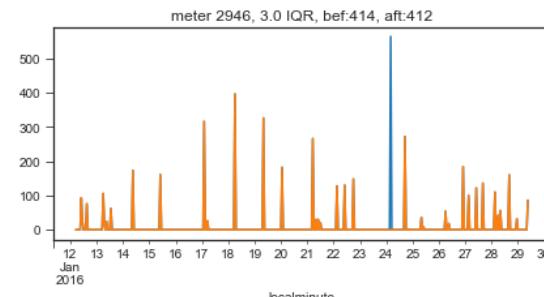
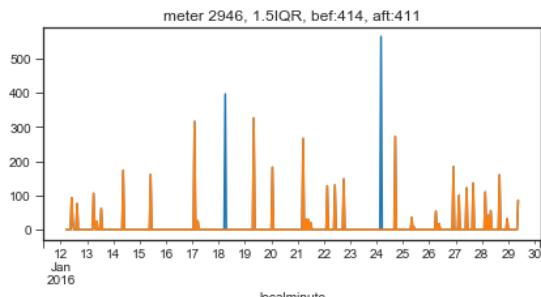
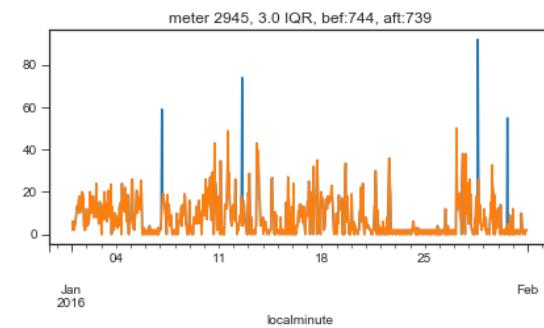
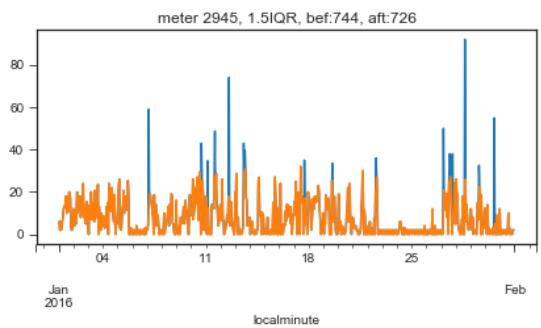
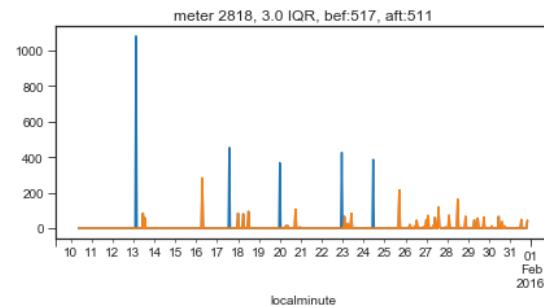
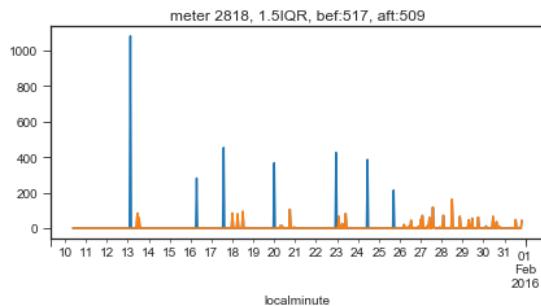


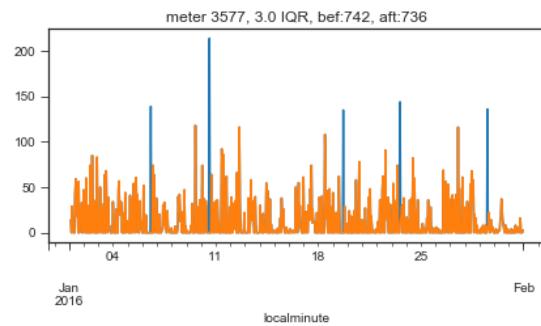
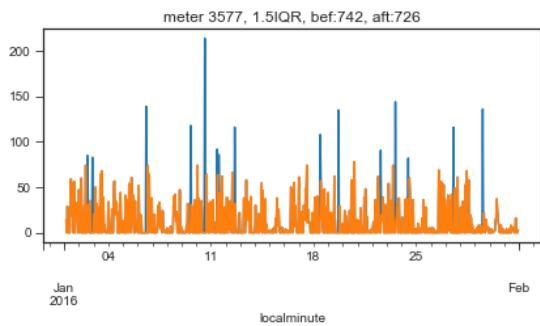
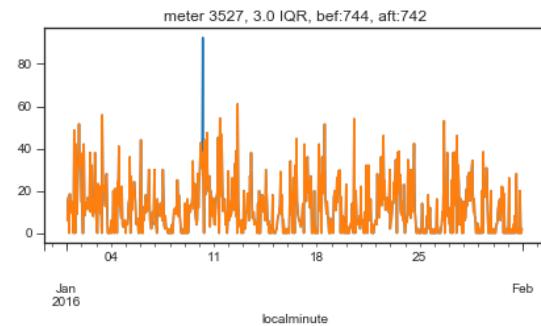
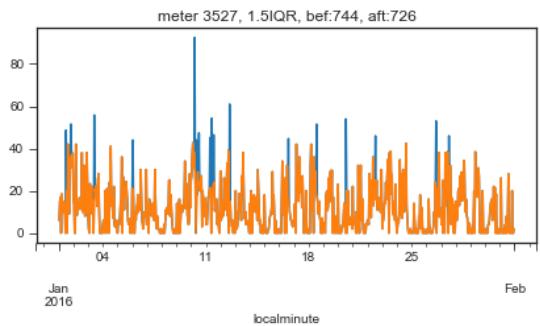
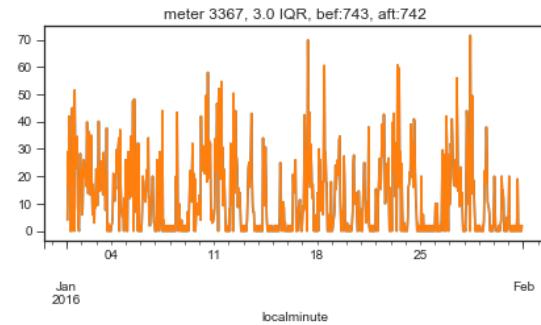
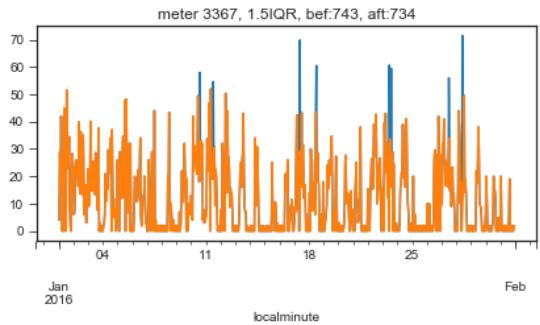
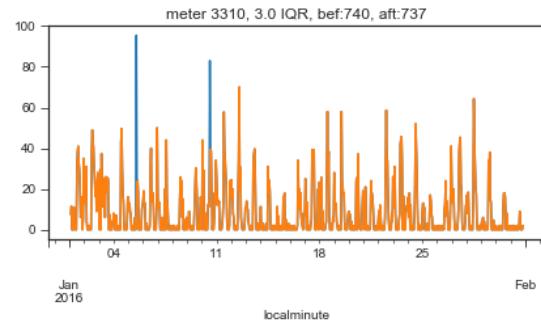
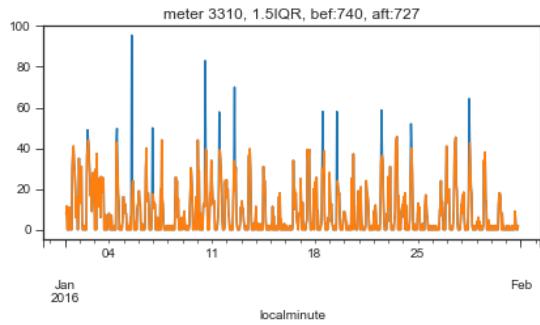
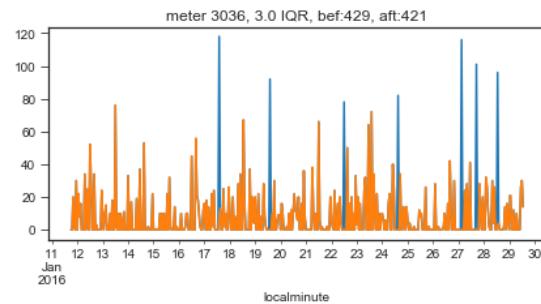
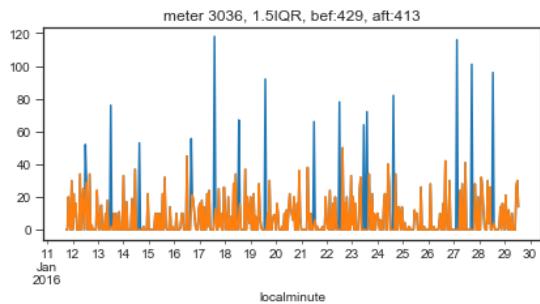


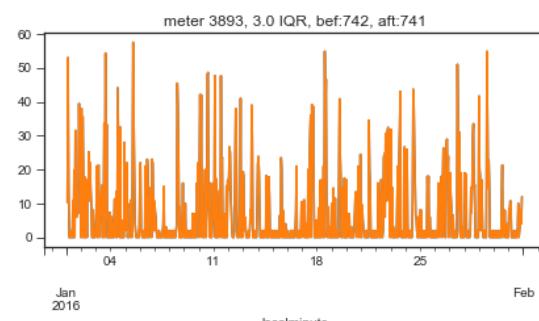
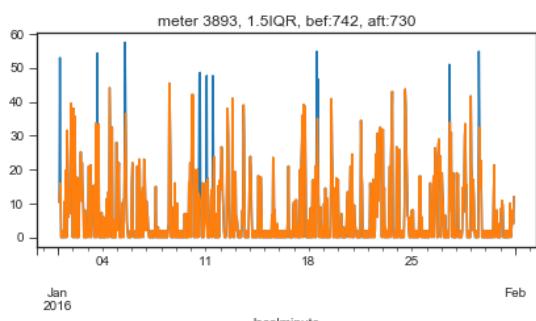
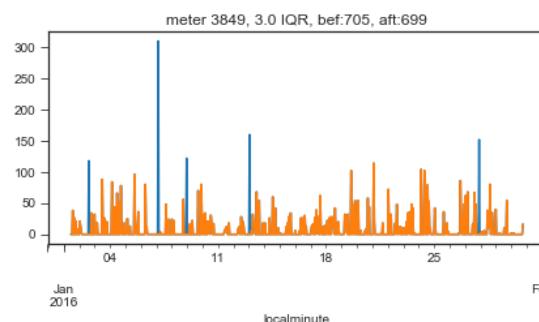
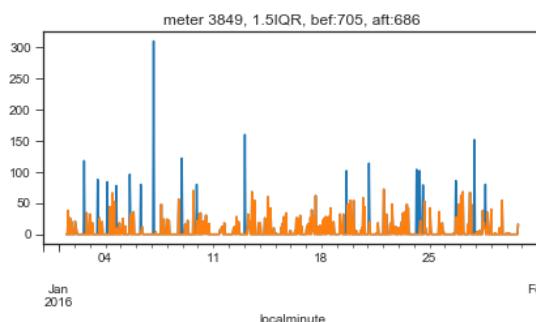
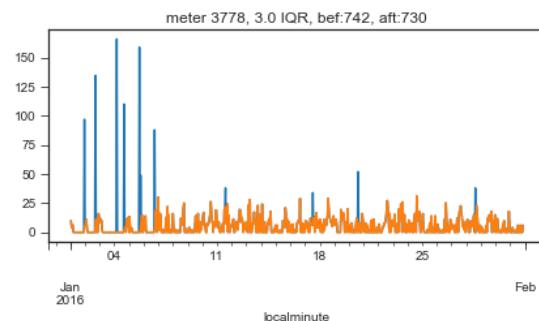
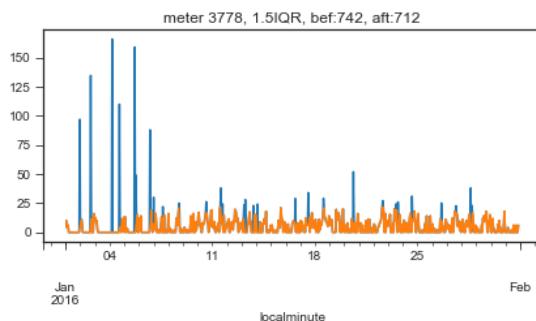
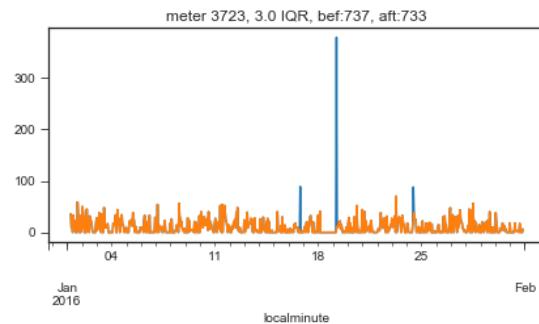
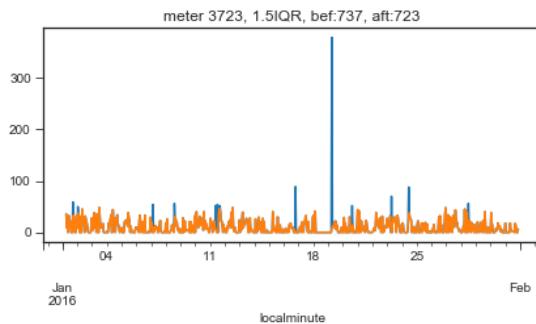
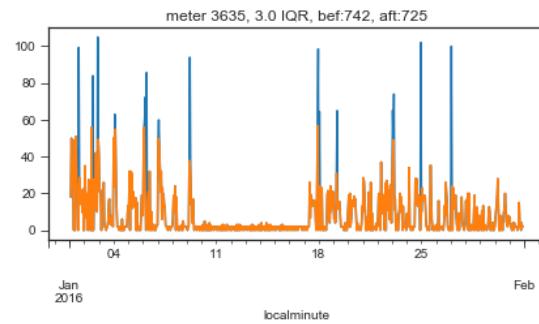
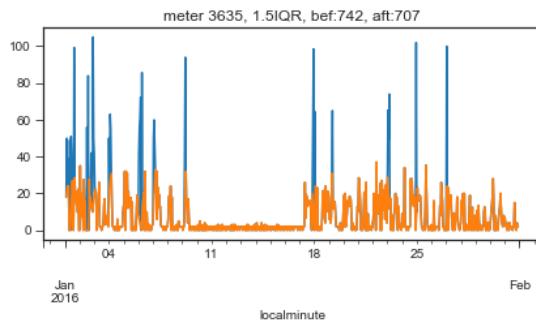
## exploration\_mel

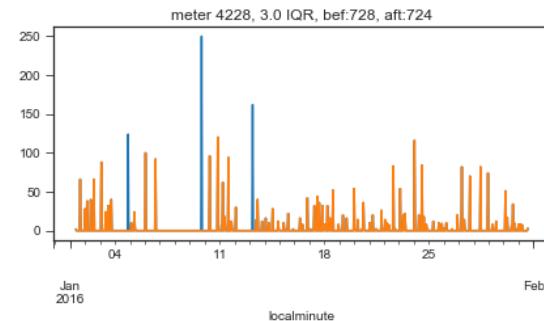
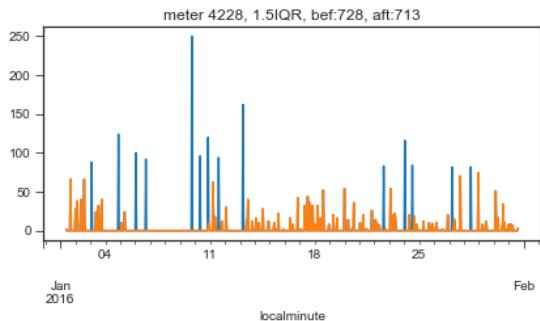
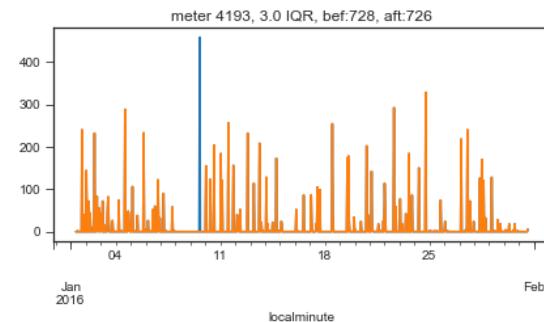
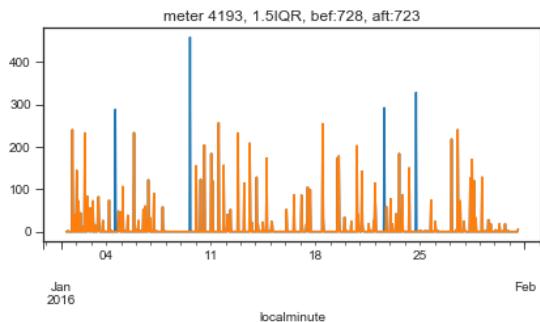
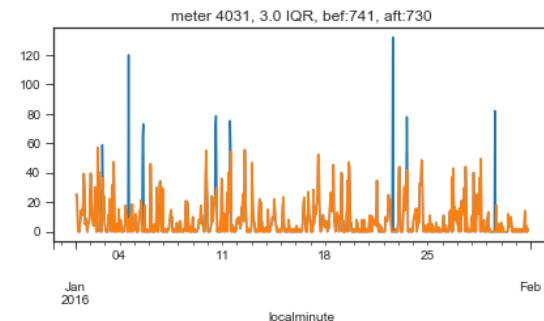
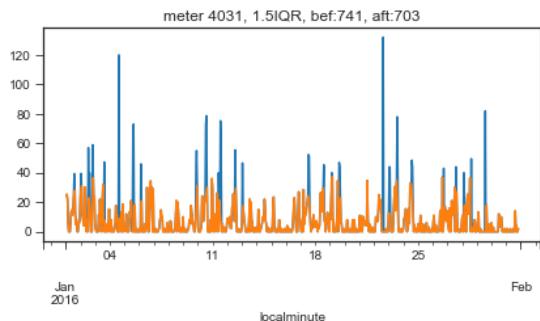
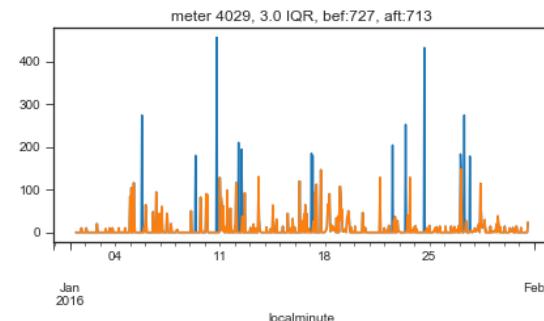
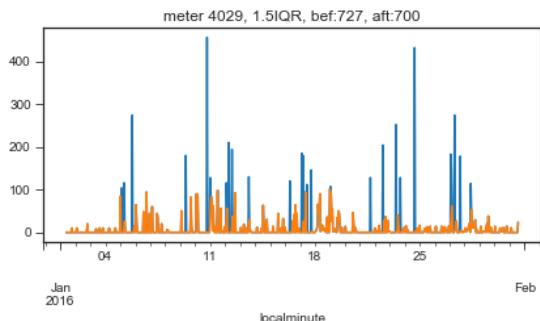
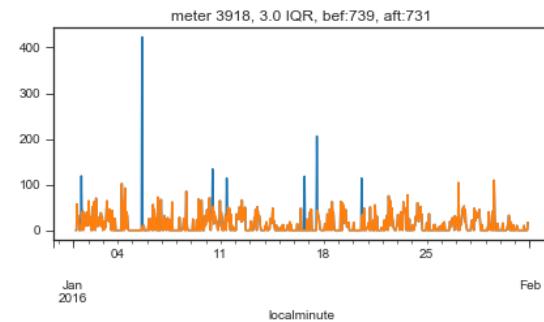
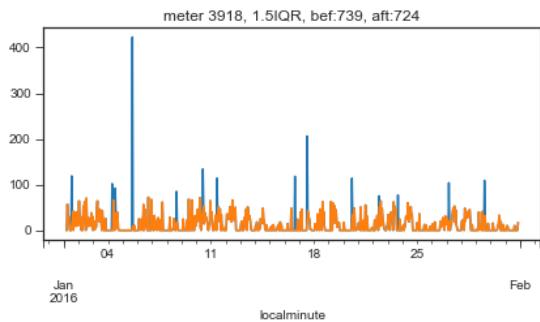


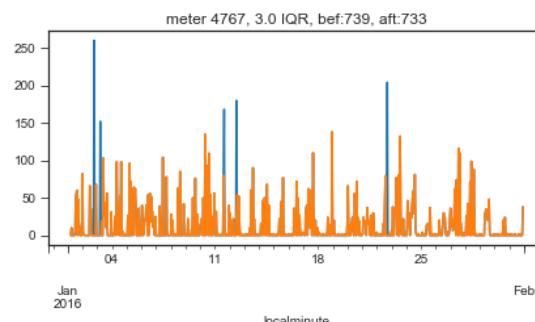
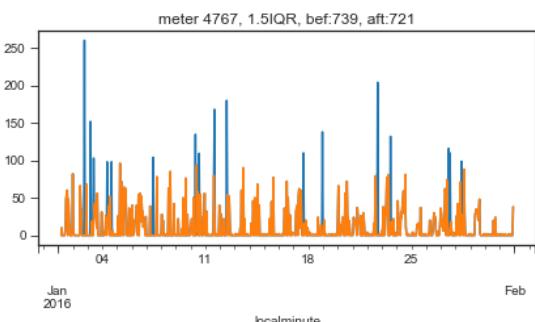
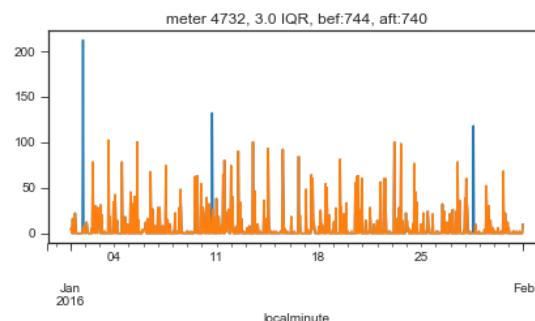
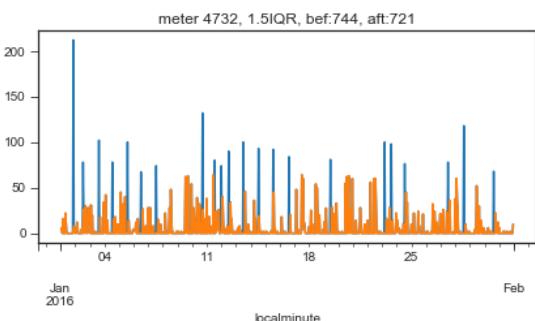
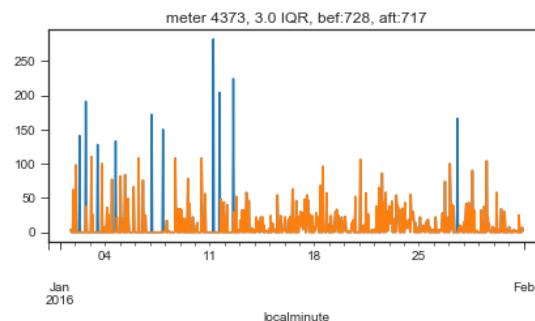
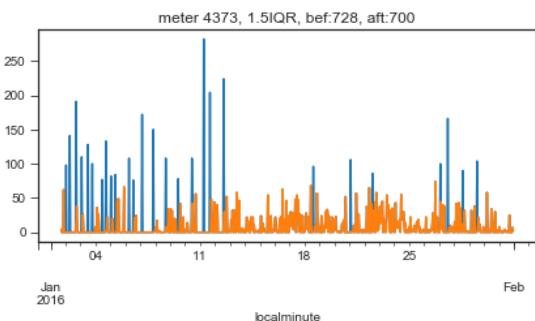
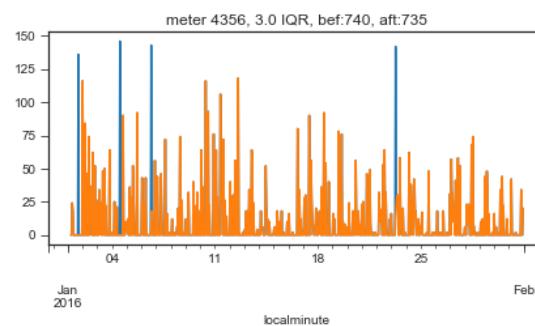
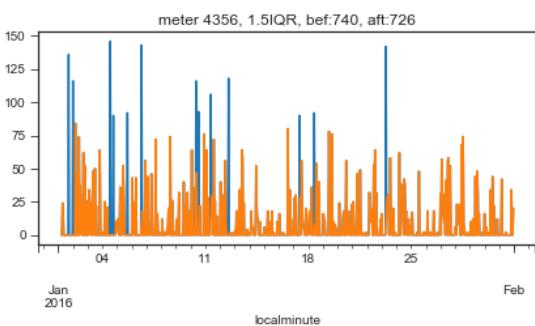
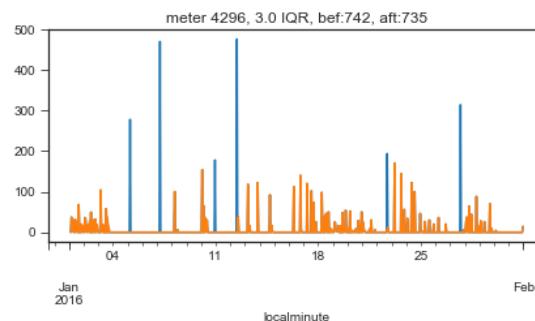
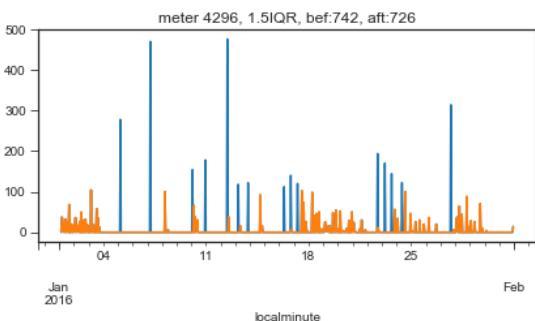


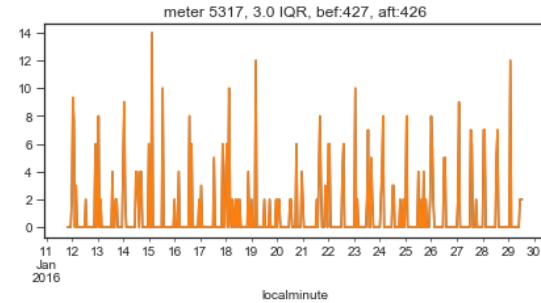
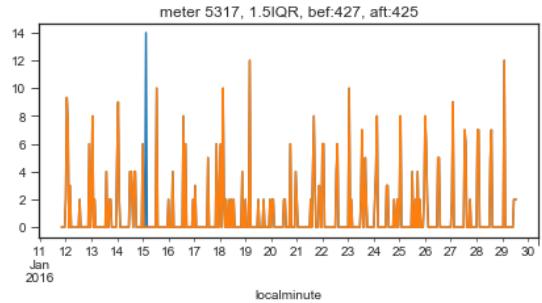
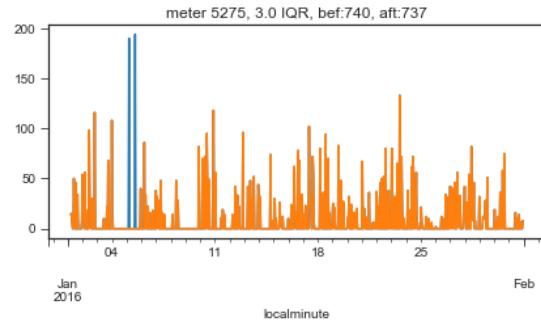
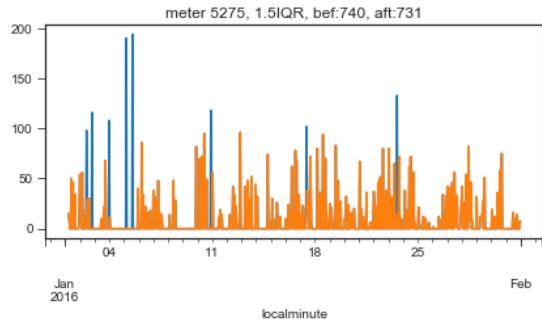
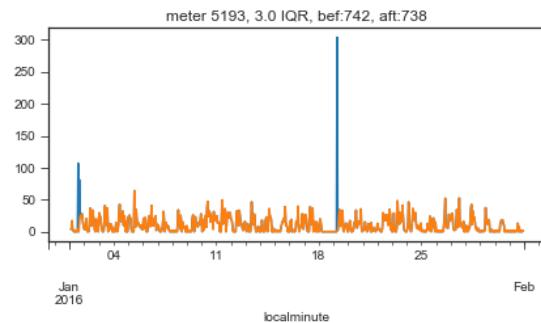
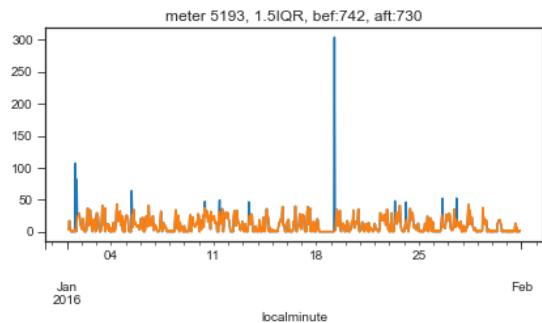
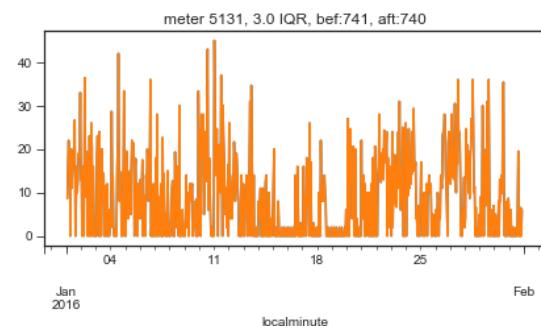
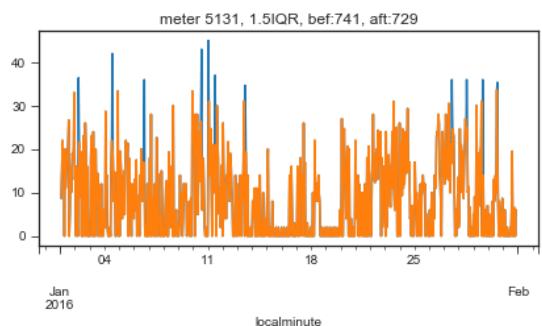
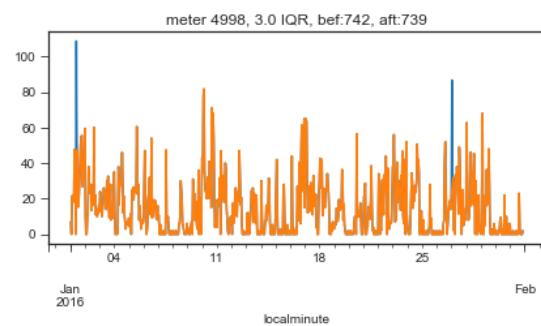
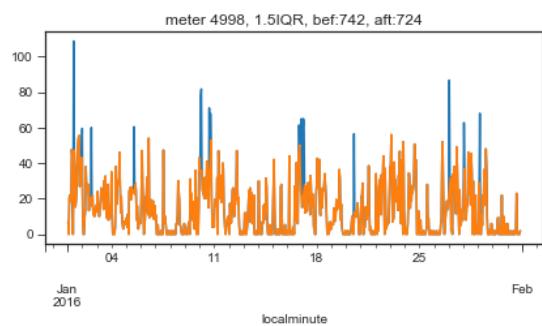


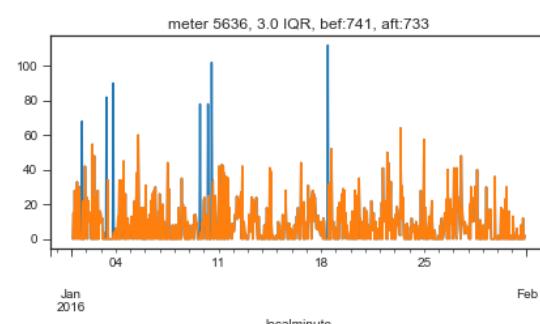
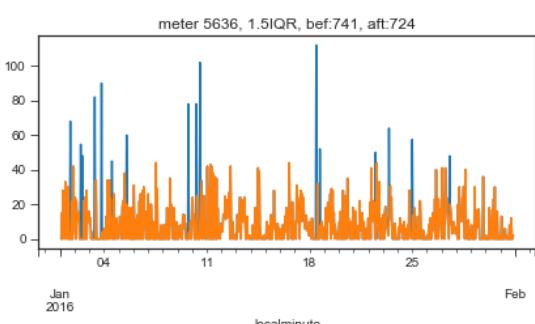
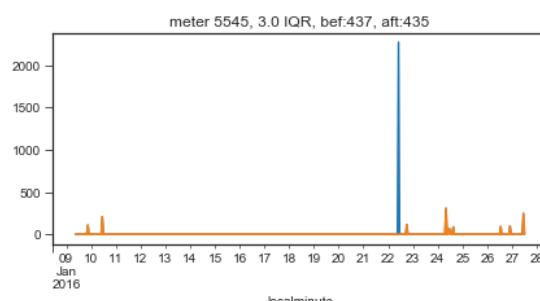
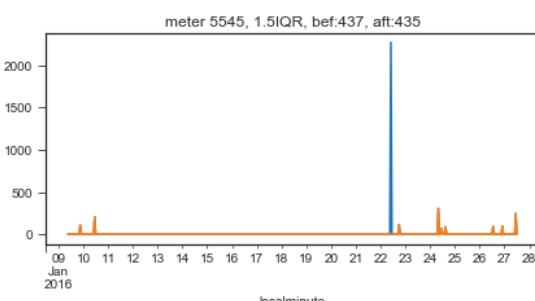
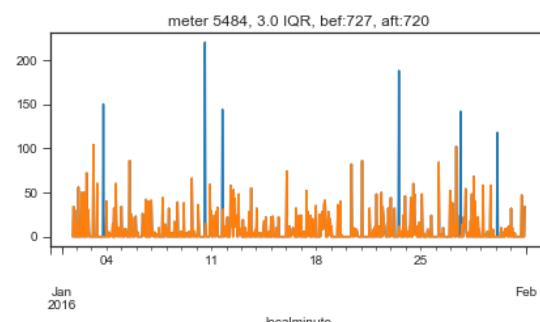
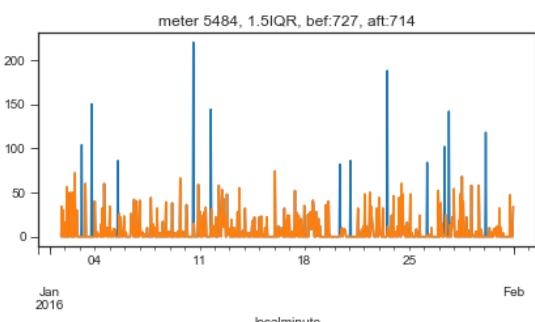
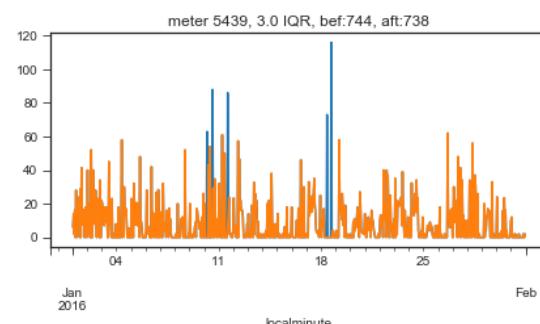
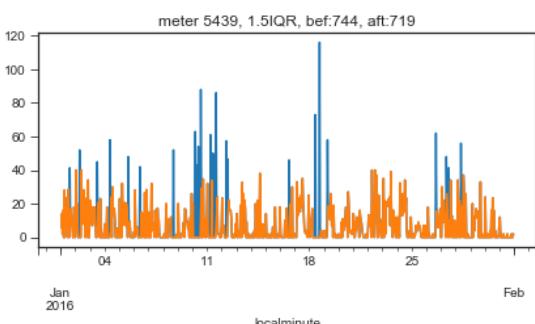
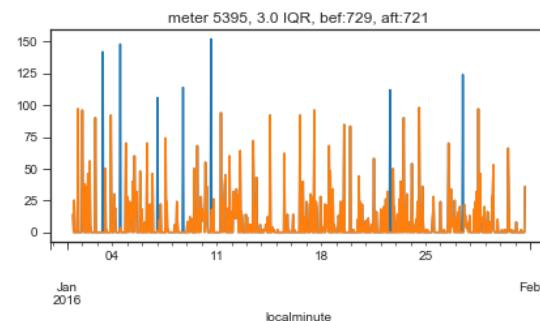
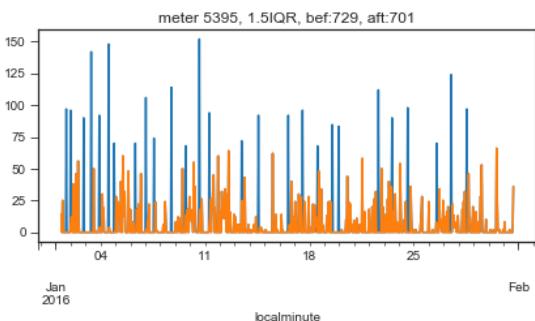




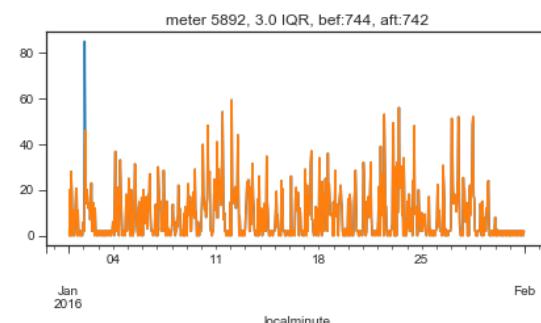
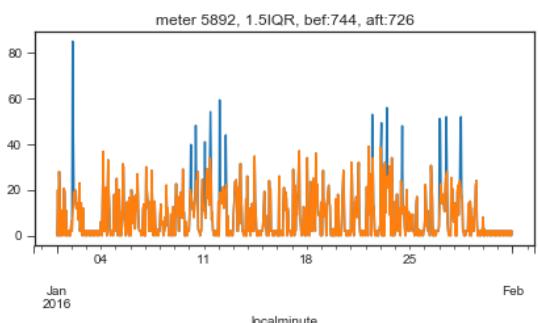
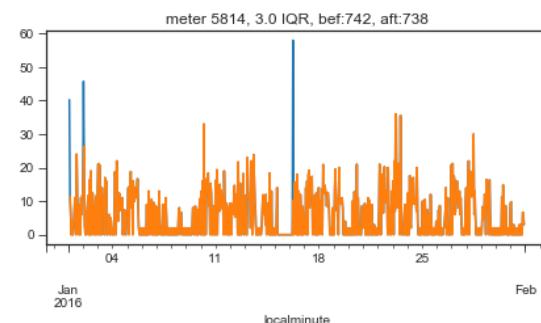
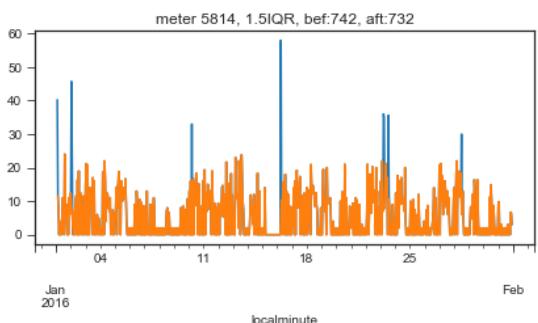
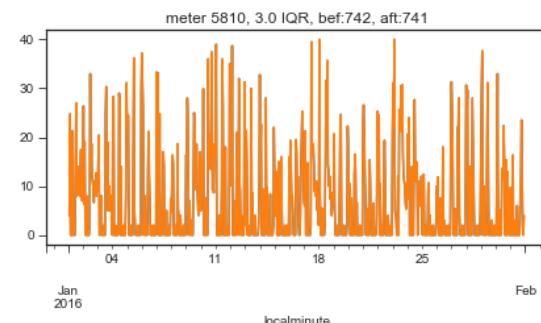
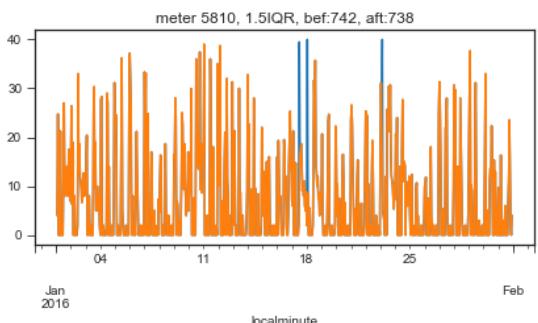
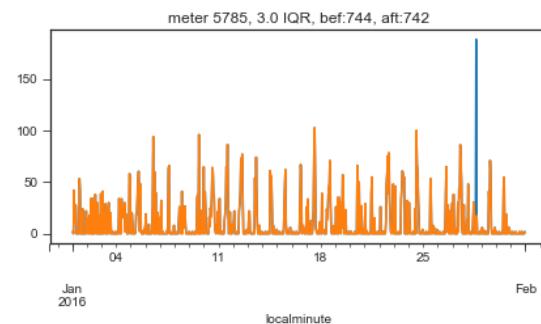
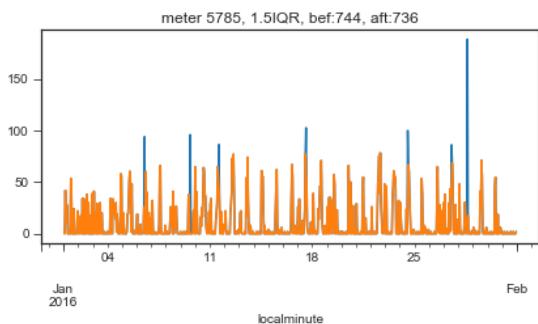
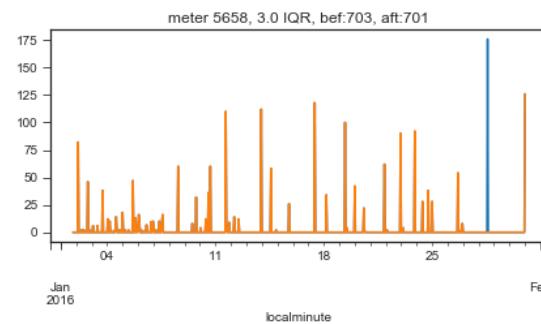
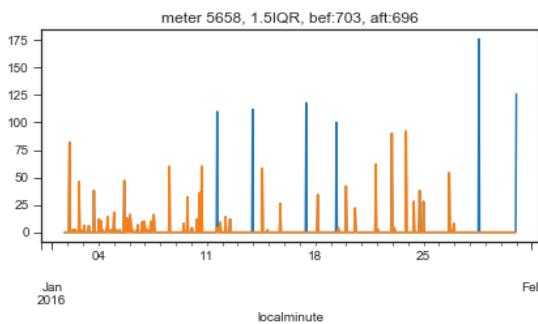


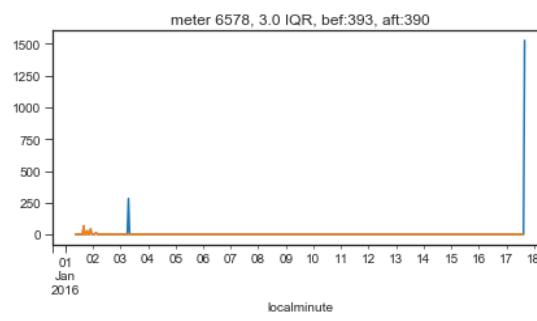
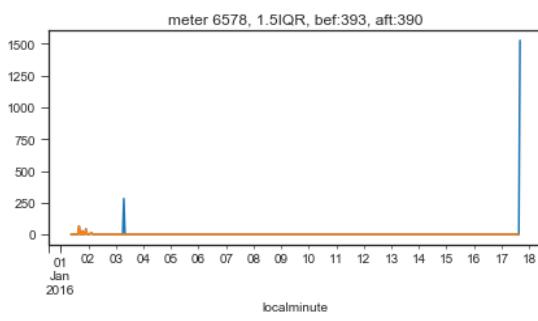
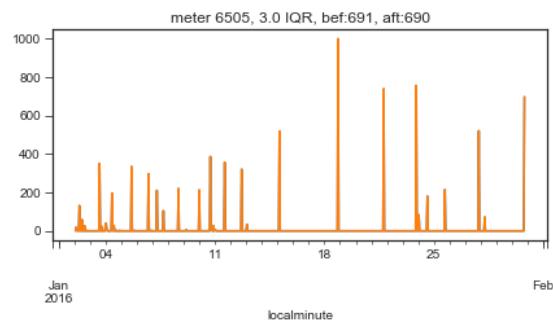
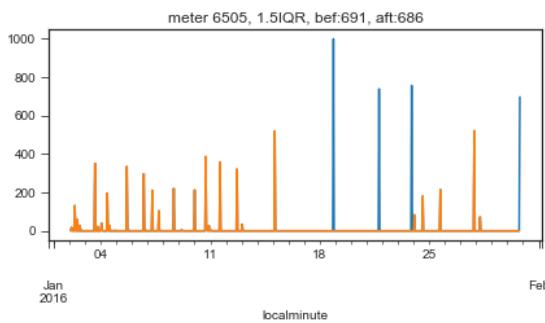
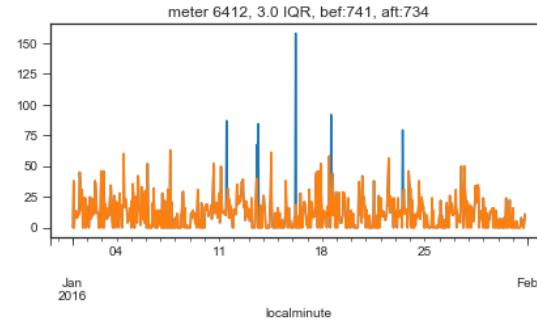
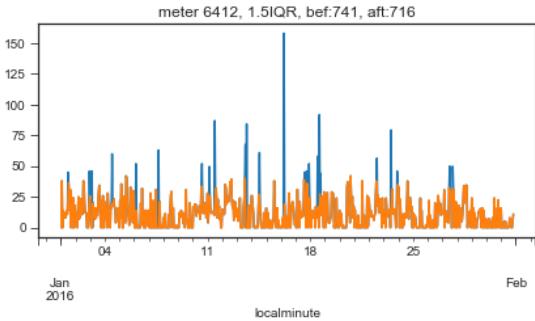
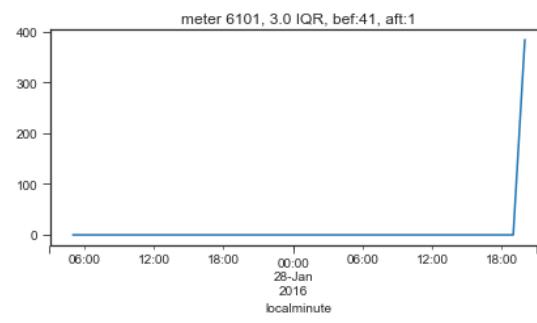
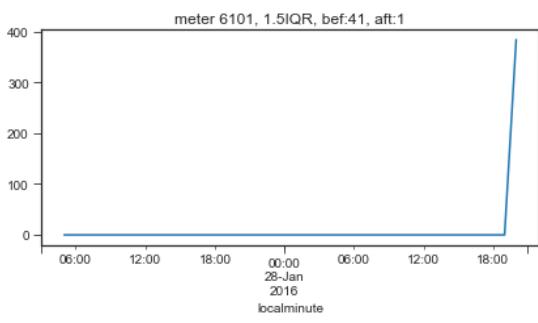
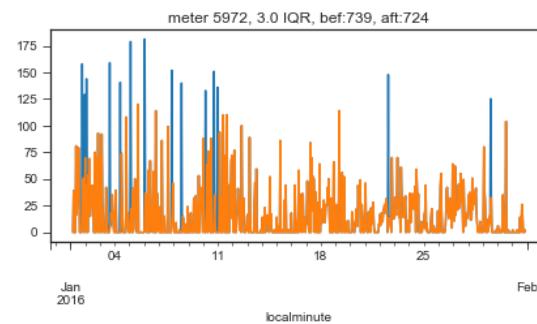
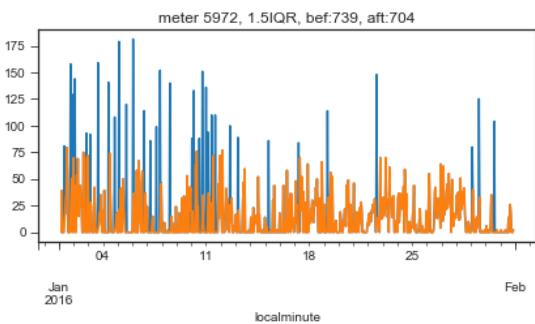


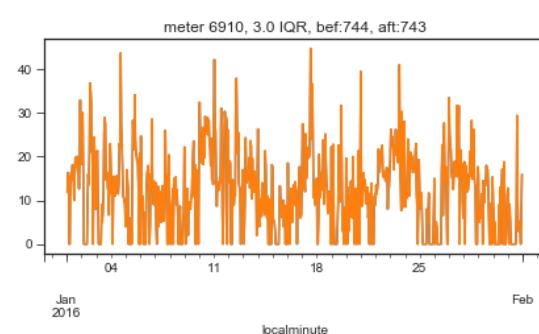
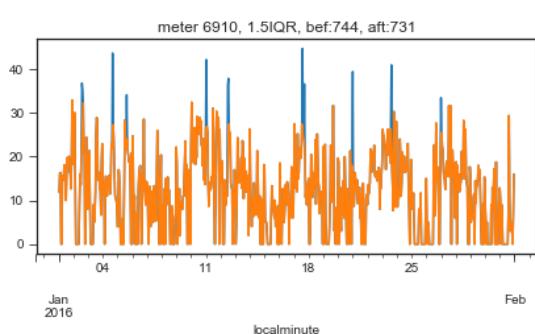
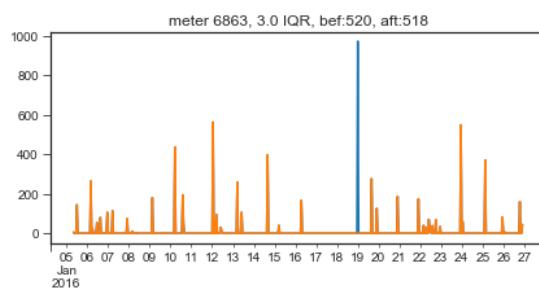
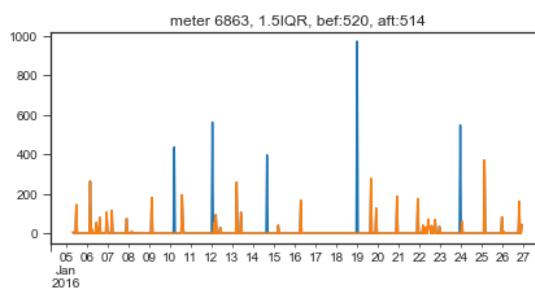
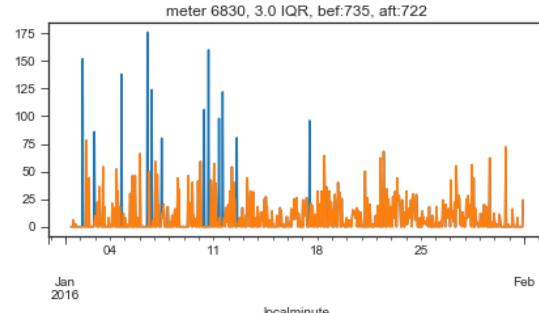
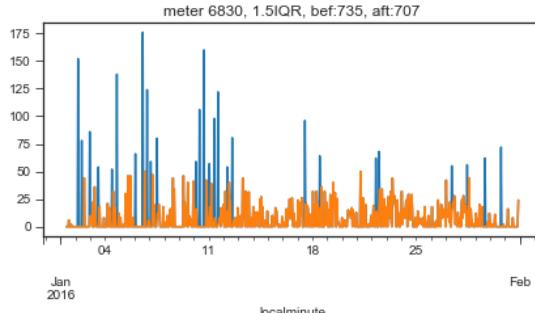
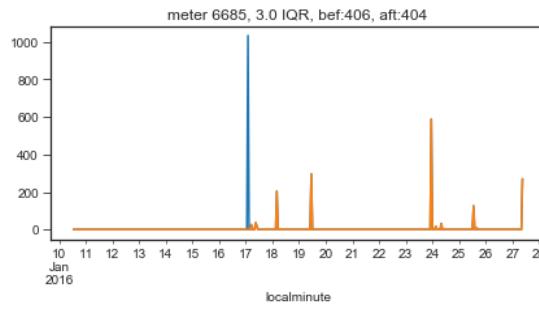
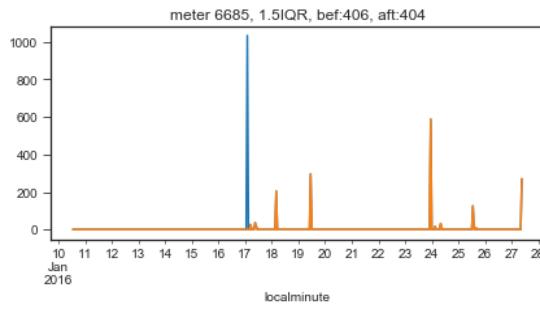
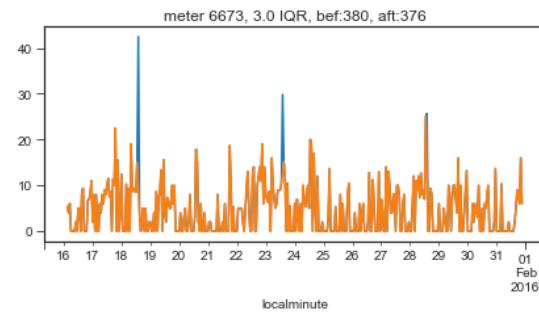
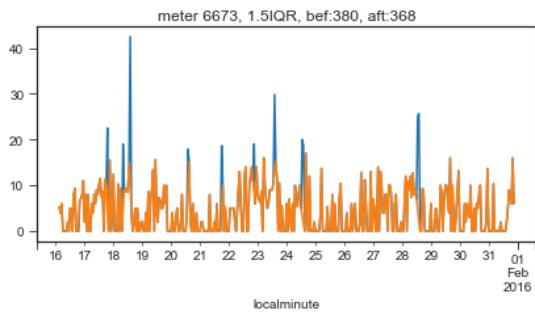


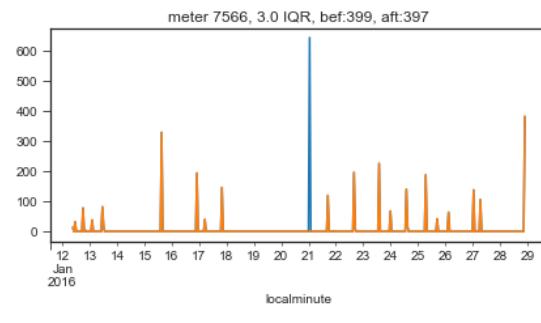
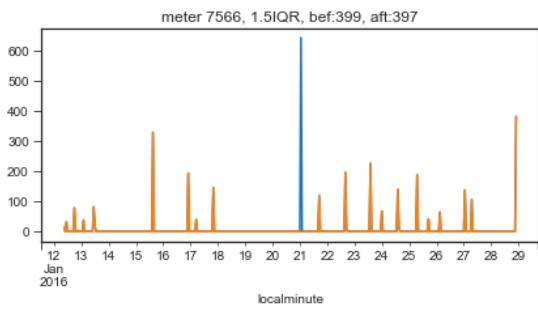
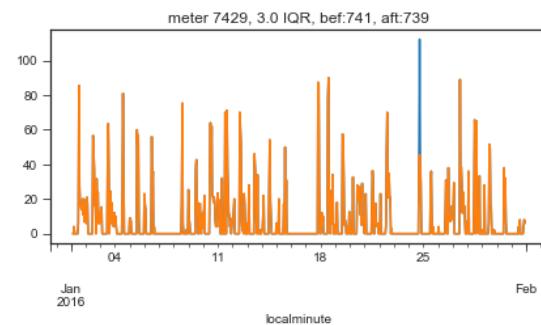
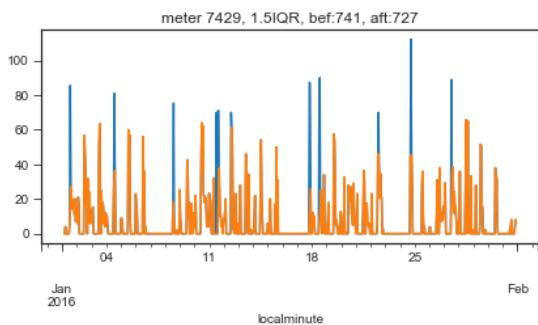
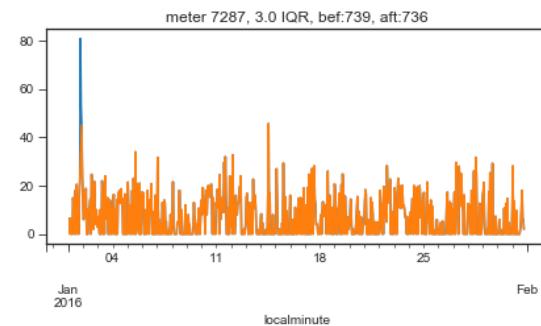
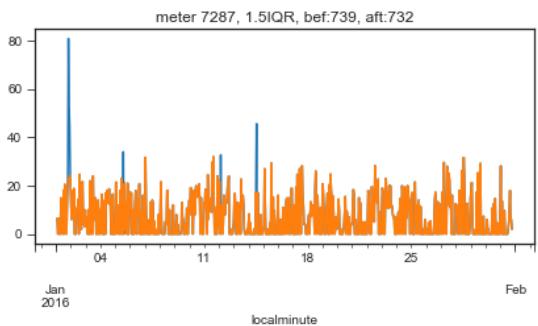
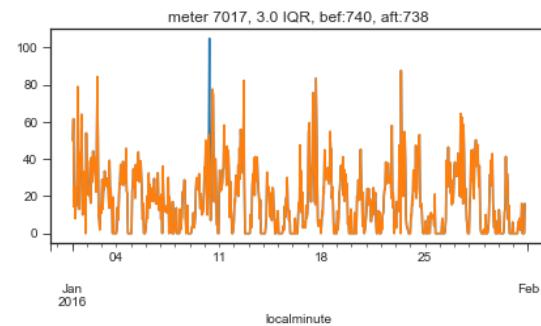
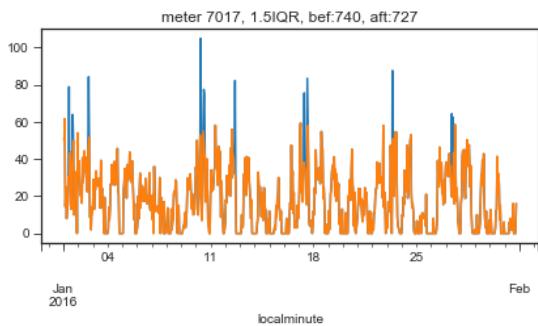
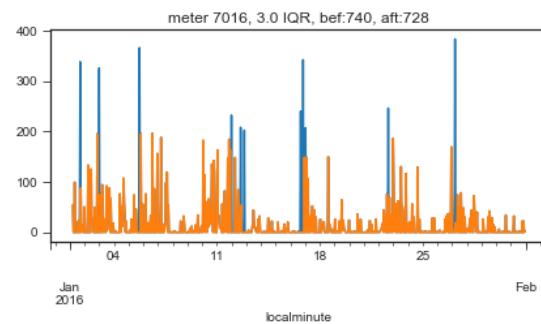
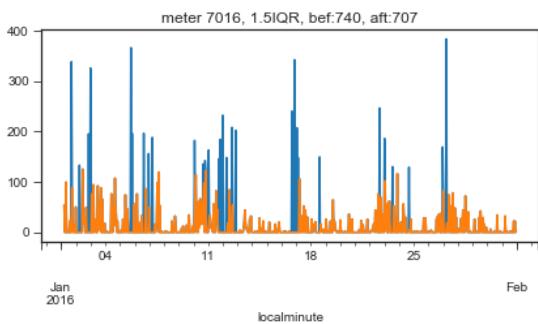


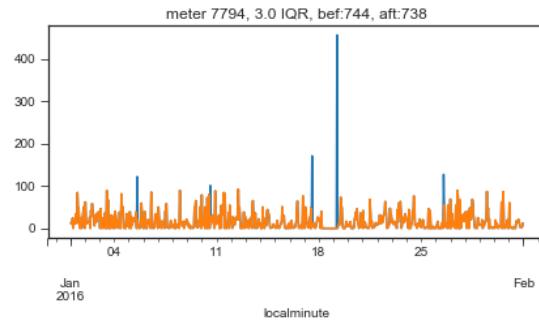
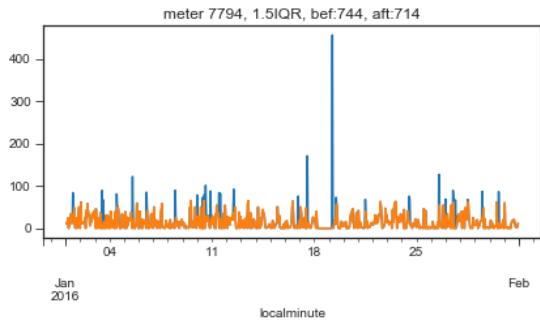
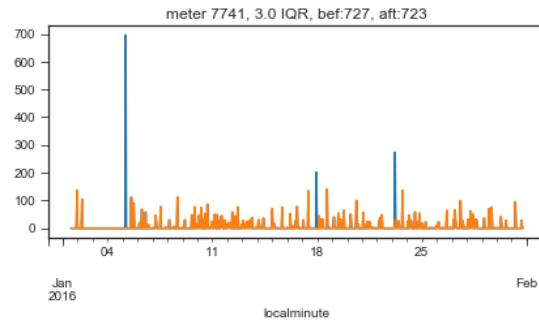
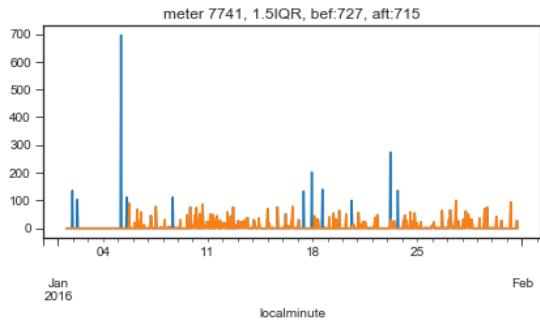
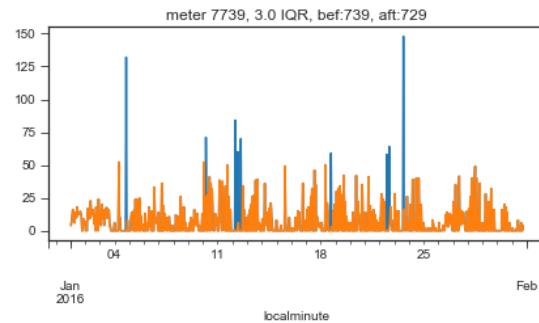
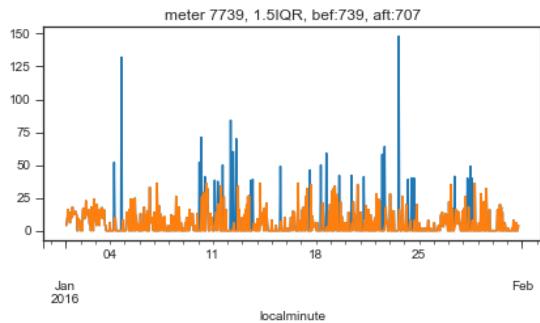
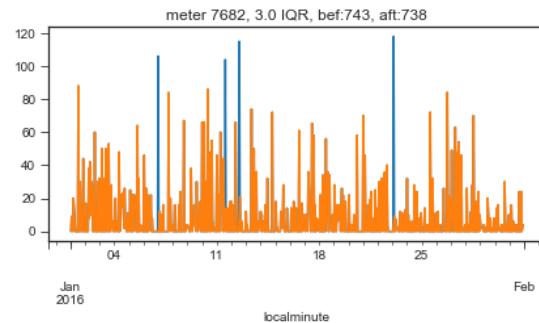
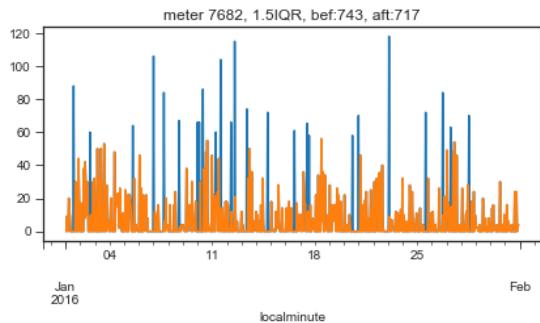
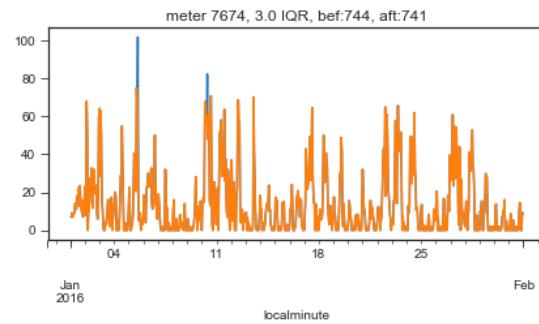
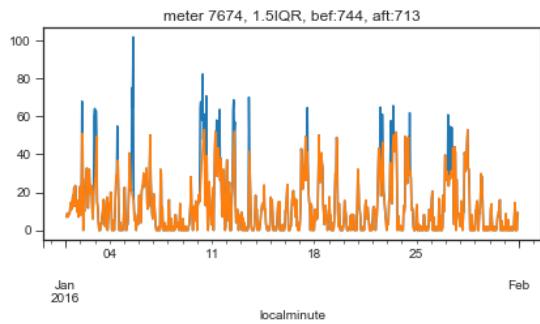
## exploration\_mel

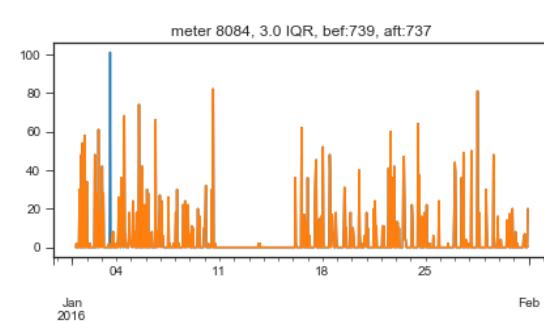
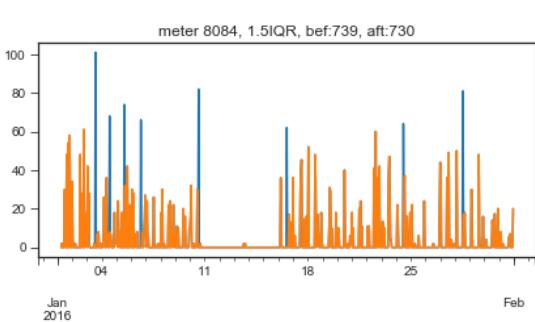
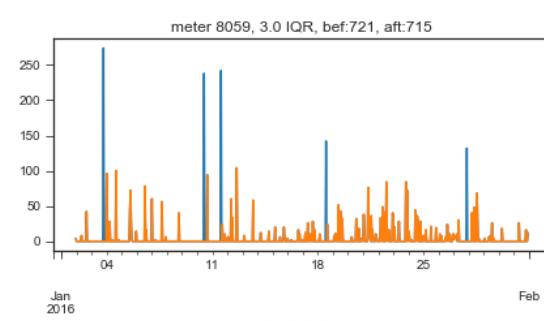
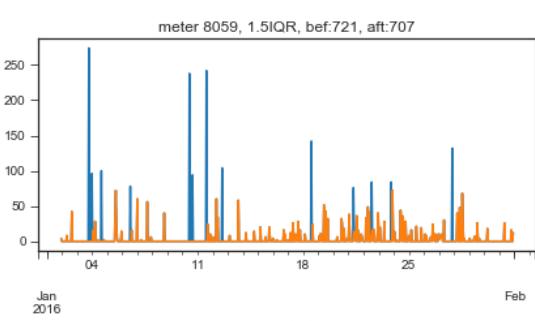
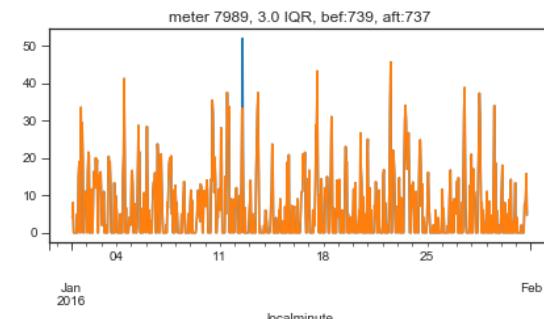
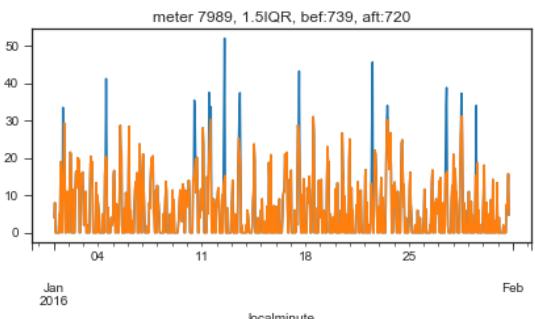
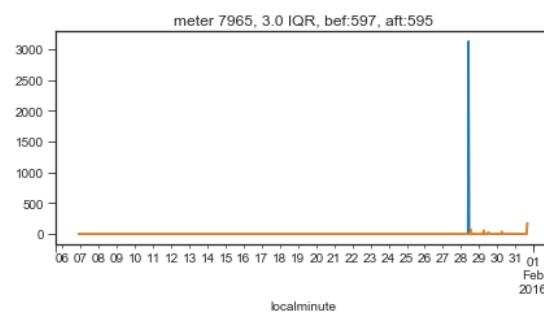
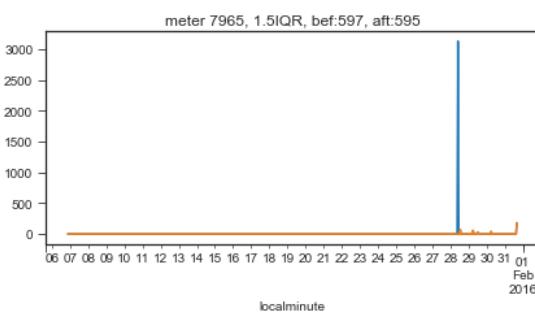
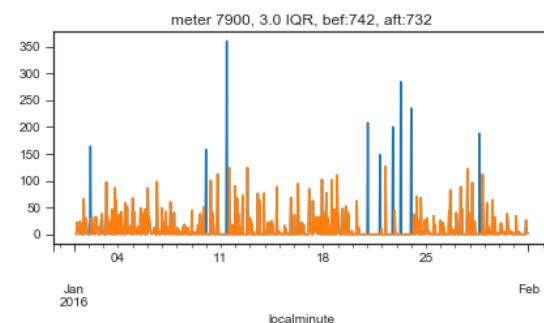
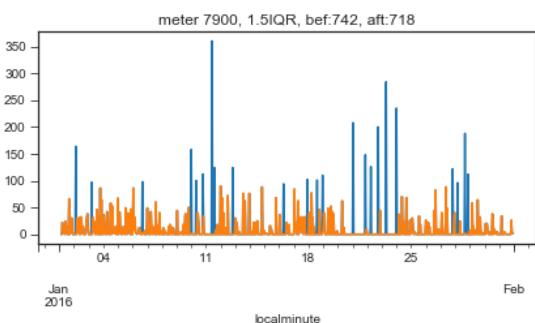


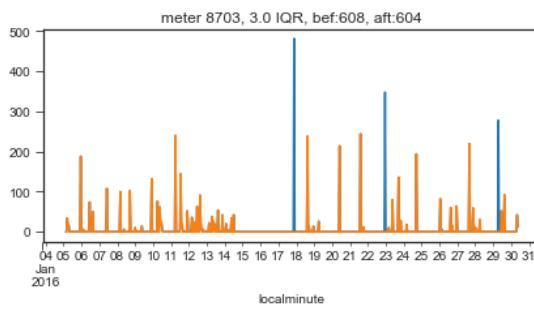
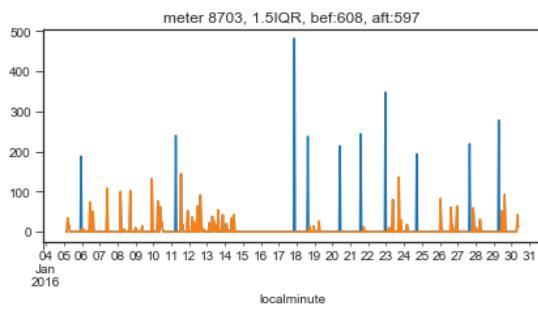
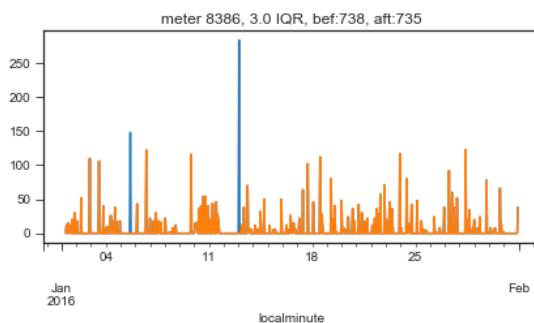
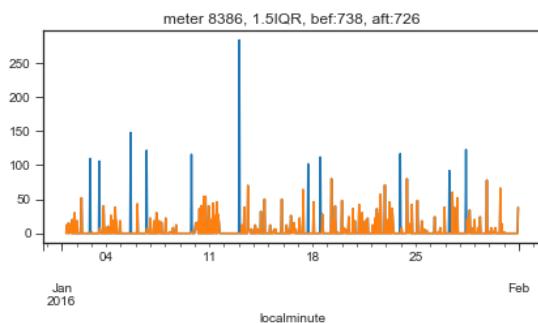
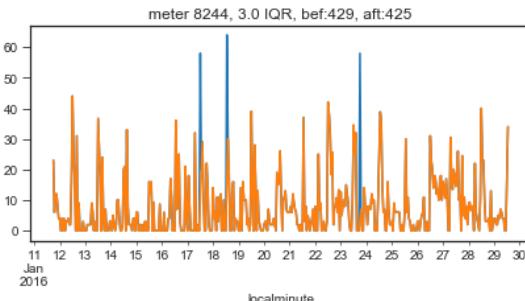
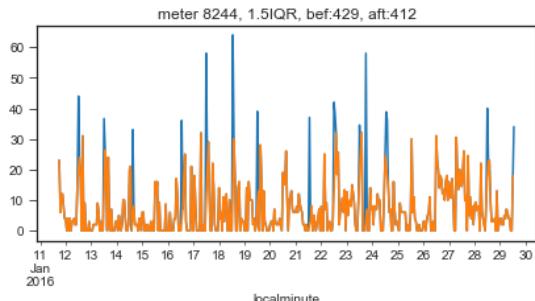
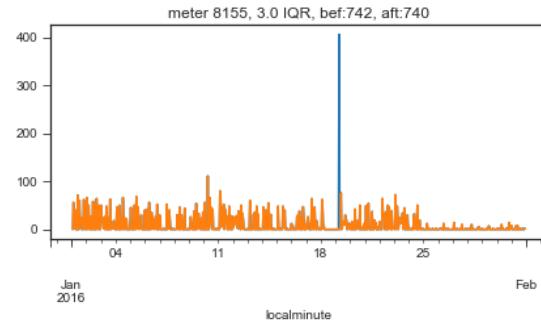
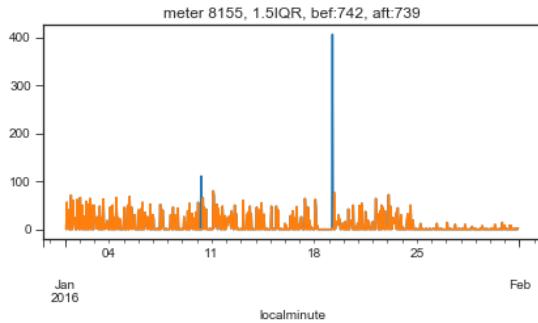
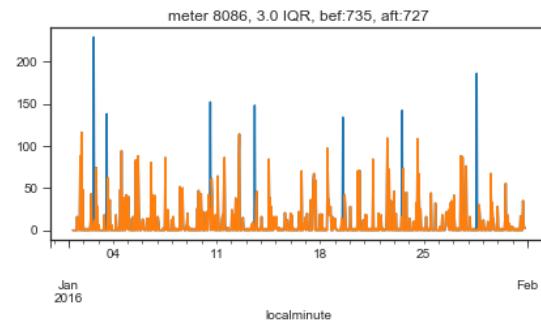
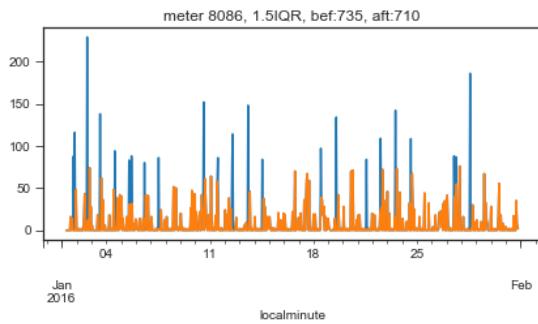


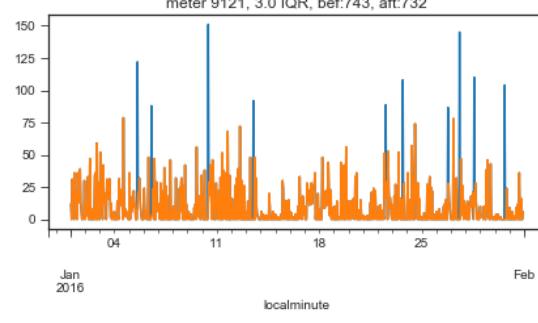
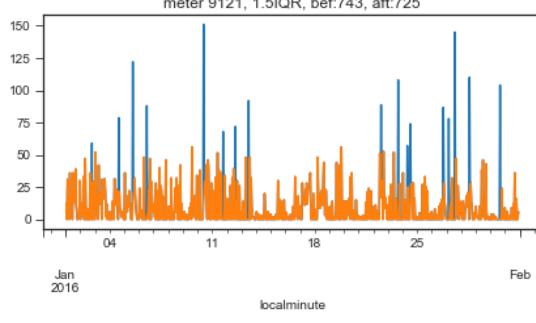
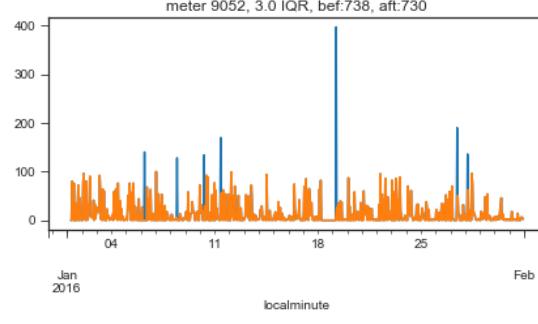
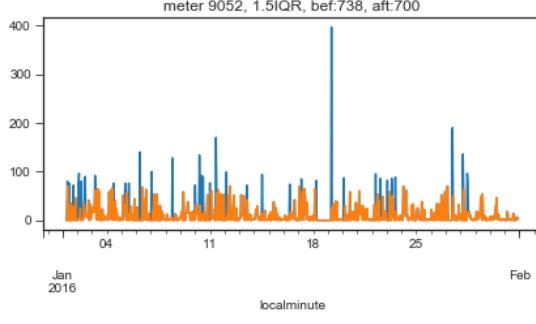
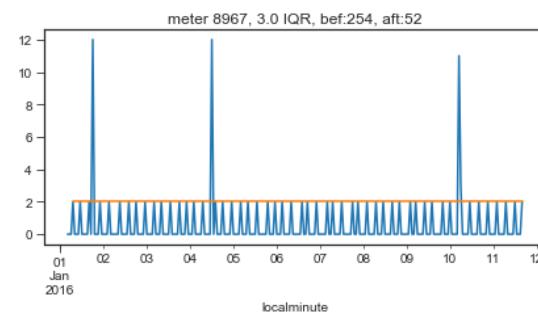
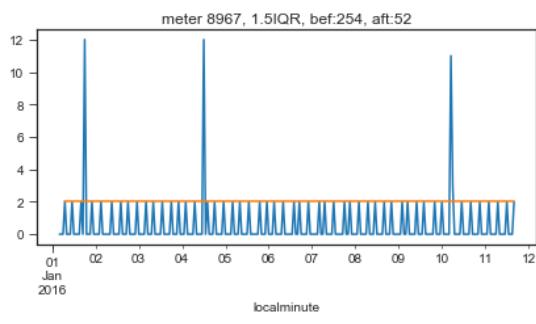
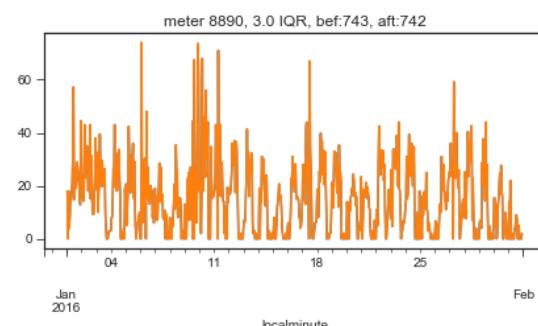
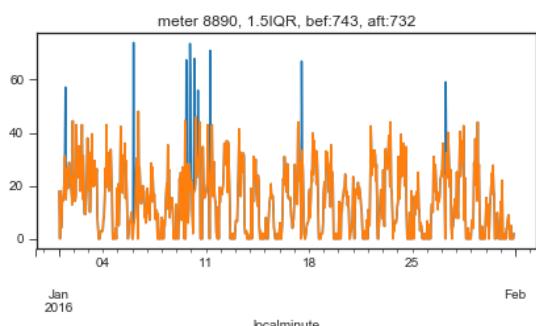
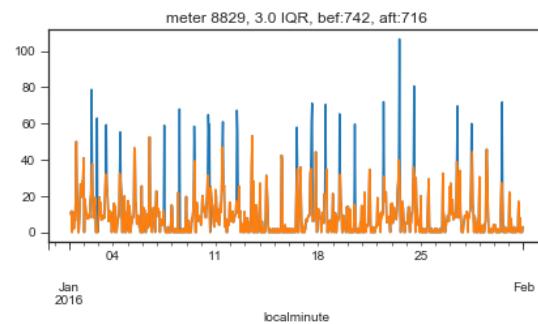
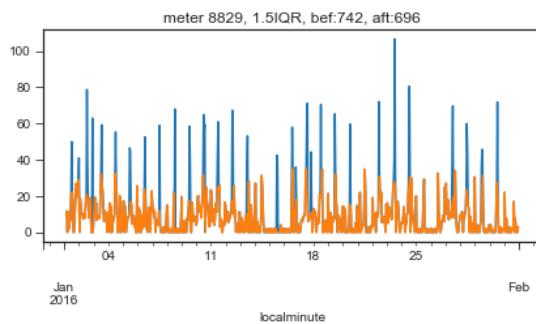


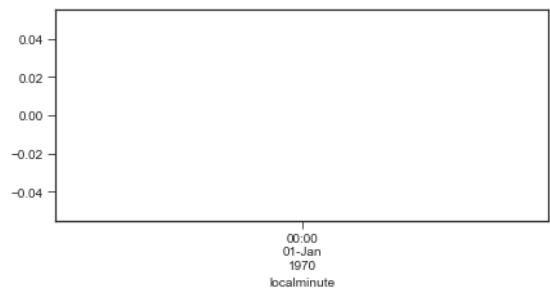
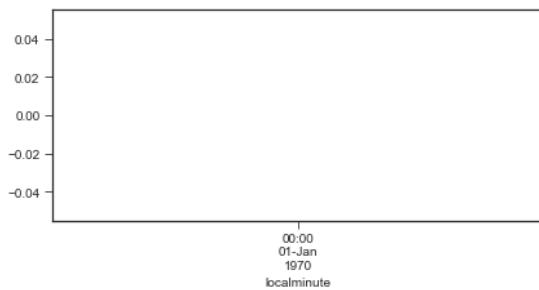
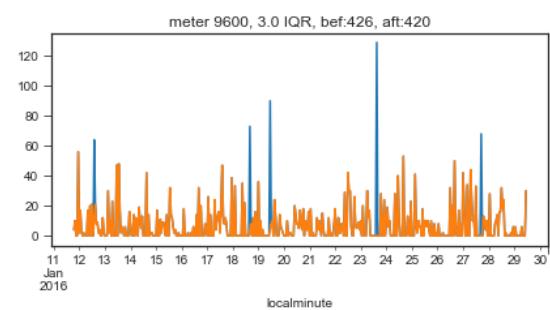
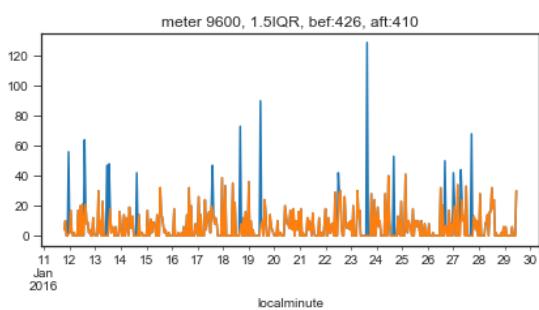
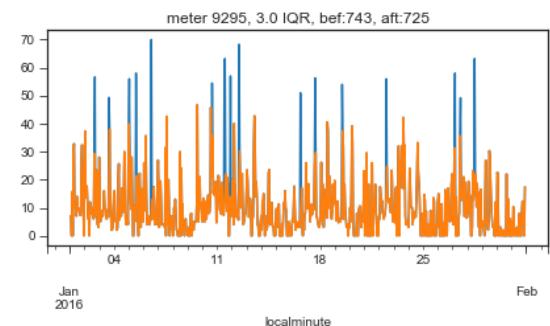
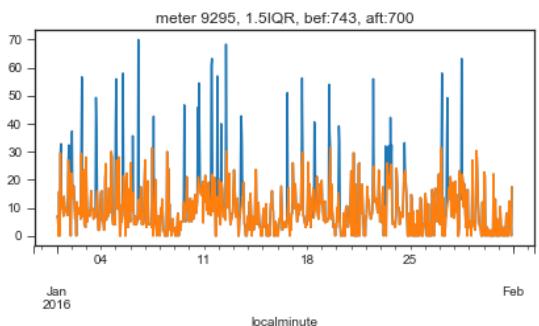
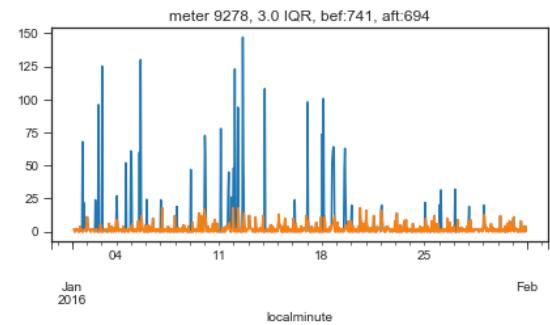
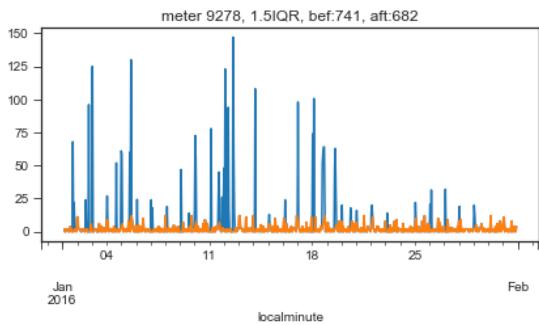
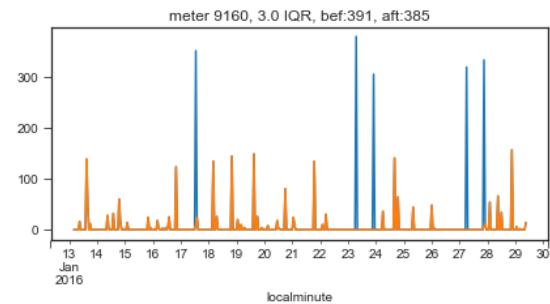
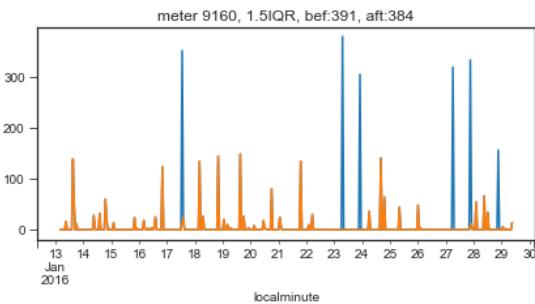


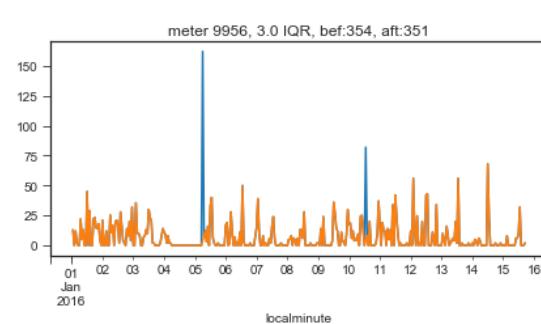
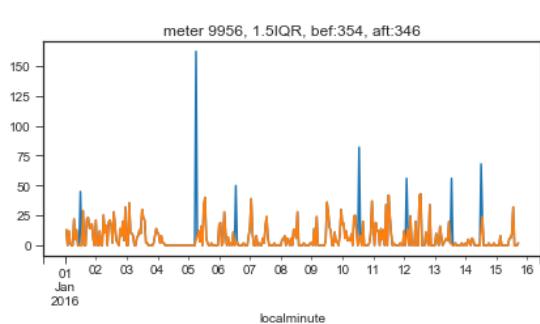
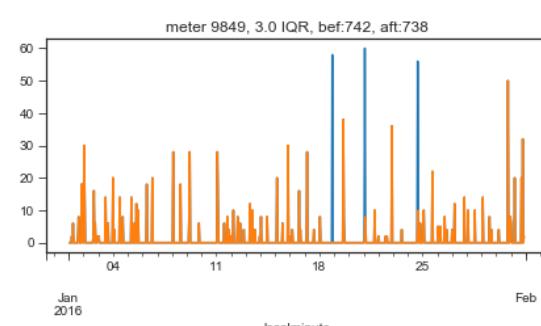
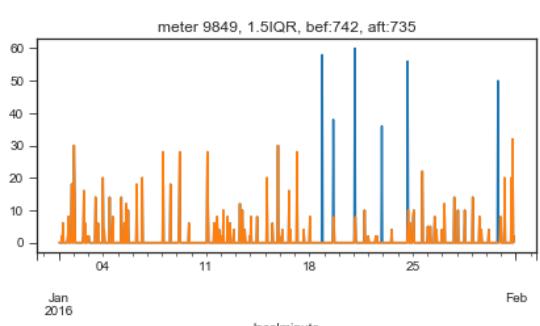
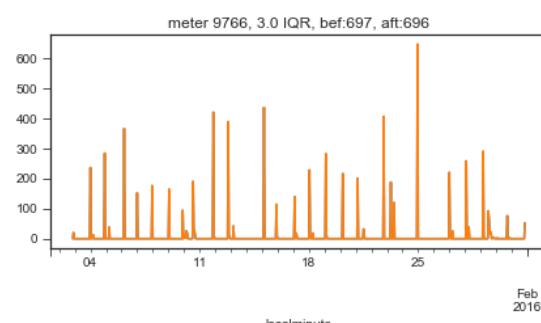
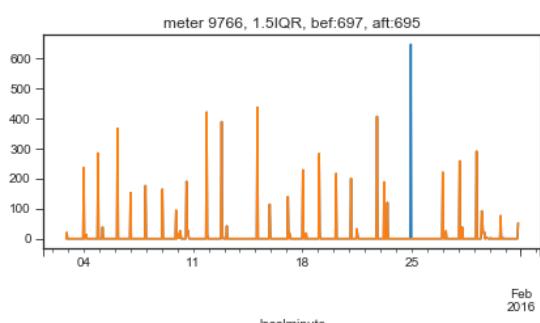
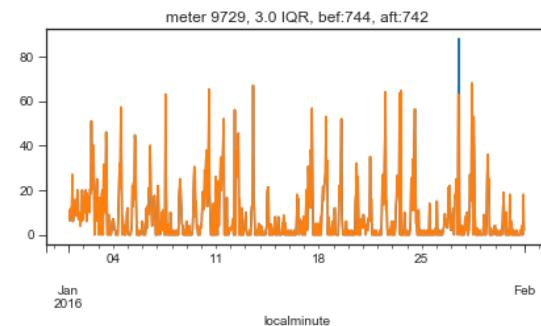
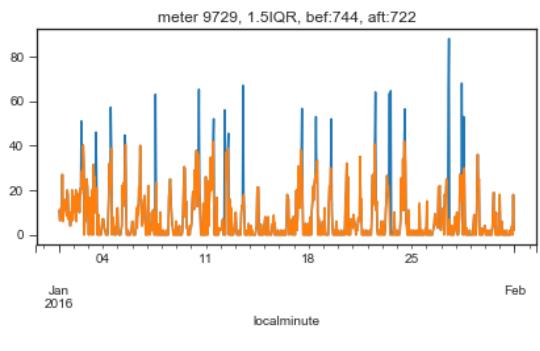
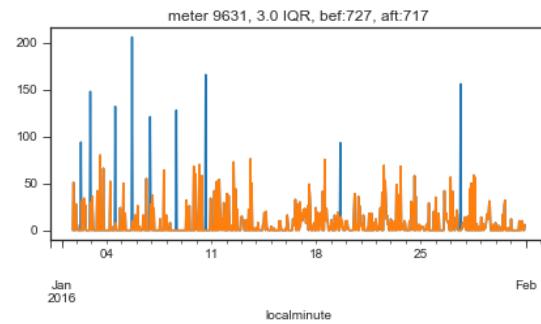
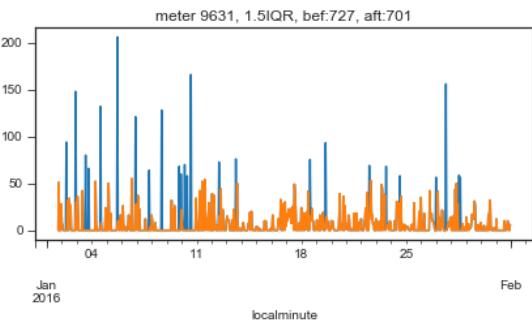












**integrate datasets with the correlation section (q1.3)**