In [1]:

```
import pandas as pd
import numpy as np
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_excel('dataexport_20201121T075029.xlsx')
df.head()
```

Out[2]:

	id	date	time	value
0	1	2019-01-01	00:00:00	5.050529
1	2	2019-01-01	01:00:00	4.900528
2	3	2019-01-01	02:00:00	4.810529
3	4	2019-01-01	03:00:00	4.730528
4	5	2019-01-01	04:00:00	4.600529

In [3]:

```
df['date'] = pd.to_datetime(df['date']).dt.date
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14616 entries, 0 to 14615
Data columns (total 4 columns):
id       14616 non-null int64
date     14616 non-null object
time     14616 non-null object
value    14616 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 456.8+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	id	value
count	14616.000000	14616.000000
mean	7308.500000	13.251613
std	4219.420102	7.770749
min	1.000000	-3.749471
25%	3654.750000	7.060529
50%	7308.500000	12.960529
75%	10962.250000	19.020529
max	14616.000000	36.120530

In [6]:

```
df.set_index(df['date'],inplace=True)
```

In [7]:

```
df.drop('date',axis=1,inplace=True)
```

In [8]:

```
df.head()
```

Out[8]:

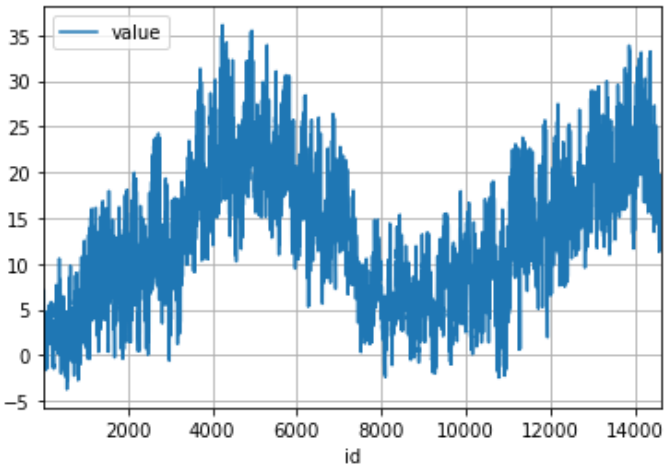
date	id	time	value
2019-01-01	1	00:00:00	5.050529
2019-01-01	2	01:00:00	4.900528
2019-01-01	3	02:00:00	4.810529
2019-01-01	4	03:00:00	4.730528
2019-01-01	5	04:00:00	4.600529

In [9]:

```
df.plot(grid=True, x='id', y='value', kind='line')
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1a948a61d0>
```



In [10]:

```
df.drop('time', axis=1, inplace = True)
```

In [11]:

```
df=df.groupby('date').agg({'value':'mean'})
```

In [12]:

```
df
```

Out[12]:

date	value
2019-01-01	4.839695
2019-01-02	2.189279
2019-01-03	0.052195
2019-01-04	1.189695

date	value
2019-01-05	1.893445
2019-01-06	3.028862
2019-01-07	2.843445
2019-01-08	4.344279
2019-01-09	2.552612
2019-01-10	0.374695
2019-01-11	0.575112
2019-01-12	3.666362
2019-01-13	6.430529
2019-01-14	5.473862
2019-01-15	3.308445
2019-01-16	4.222195
2019-01-17	6.020945
2019-01-18	1.087612
2019-01-19	0.589279
2019-01-20	2.991779
2019-01-21	0.123029
2019-01-22	-0.386138
2019-01-23	1.235945
2019-01-24	-1.514471
2019-01-25	-0.898221
2019-01-26	4.507612
2019-01-27	6.102195
2019-01-28	2.280945
2019-01-29	1.873029
2019-01-30	1.378862
...	...
2020-08-02	22.233446
2020-08-03	17.924279
2020-08-04	16.341362
2020-08-05	18.532195
2020-08-06	20.753446
2020-08-07	23.731363
2020-08-08	25.490946
2020-08-09	26.229280
2020-08-10	26.653029
2020-08-11	27.071779
2020-08-12	27.361779
2020-08-13	25.960946
2020-08-14	21.561779
2020-08-15	22.557196
2020-08-16	23.510946
2020-08-17	20.805946
2020-08-18	19.835946
2020-08-19	22.083029

2020-08-20	**25.705529**
2020-08-21	**26.834696**
2020-08-22	**22.198029**
2020-08-23	**19.730112**
2020-08-24	**19.376363**
2020-08-25	**20.503862**
2020-08-26	**21.866363**
2020-08-27	**20.256779**
2020-08-28	**18.643863**
2020-08-29	**15.507196**
2020-08-30	**13.490946**
2020-08-31	**15.649696**

609 rows × 1 columns

In [13]:

```
df.reset_index(inplace = True, drop = False)
```

In [14]:

```
datetemp = df.copy(deep=True)
```

In [15]:

```
df.set_index(df['date'],inplace = True)
```

In [16]:

```
df.drop('date', axis=1, inplace=True)
```

In [17]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 609 entries, 2019-01-01 to 2020-08-31
Data columns (total 1 columns):
value    609 non-null float64
dtypes: float64(1)
memory usage: 9.5+ KB
```

In [18]:

```
df.describe()
```

Out[18]:

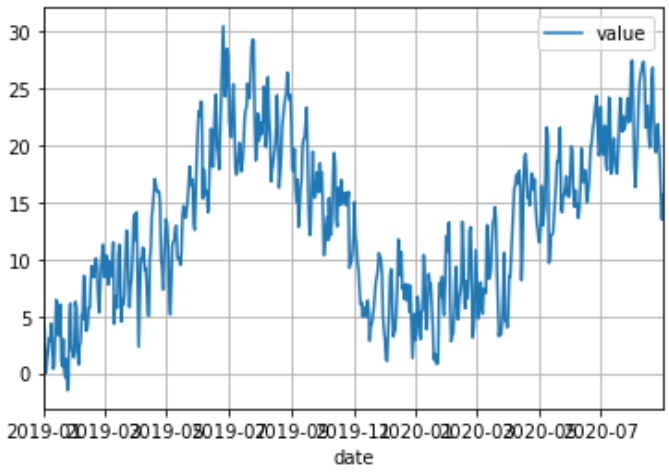
	value
count	**609.000000**
mean	**13.251613**
std	**7.041311**
min	**-1.514471**
25%	**7.331779**
50%	**13.300112**
75%	**18.643863**
max	**30.493863**

In [19]:

```
df.plot(grid=True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1ae8609f50>
```

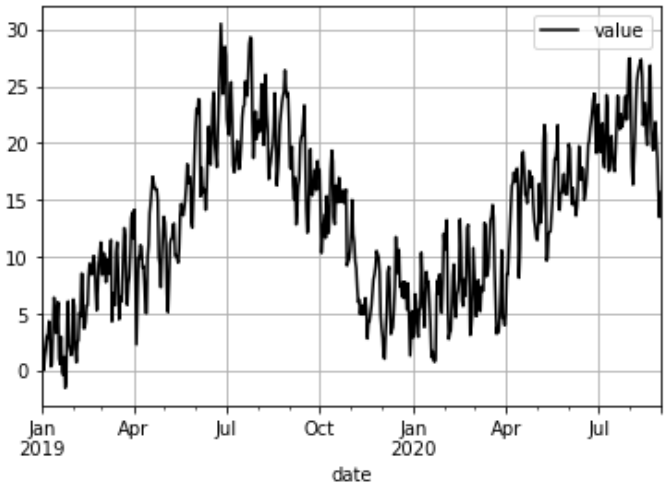


In [20]:

```
df.to_csv('temp1.csv')
```

In [21]:

```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('temp1.csv',header=0)
series.plot(style='k', legend = True, grid = True)
pyplot.show()
```



In [22]:

```
df.reset_index(inplace=True)
```

In [23]:

```
df['year'] = pd.to_datetime(df['date']).dt.year
df['month'] = pd.to_datetime(df['date']).dt.month
df['day'] = pd.to_datetime(df['date']).dt.day
df.drop('date', axis = 1, inplace = True)
```

In [24]:

```
df = df[['year', 'month', 'day', 'value']]
```

In [25]:

```
target = np.array(df['value'])
```

```
df.drop('value', axis = 1, inplace = True)
feature_list = list(target)
features = np.array(df)
```

In [26]:

```
from sklearn.model_selection import train_test_split
```

In [27]:

```
train_features, test_features, train_labels, test_labels = train_test_split(features, tar
get, test_size = 0.25, random_state=42)
```

In [28]:

```
print('Training features shape', train_features.shape)
```

```
Training features shape (456, 3)
```

In [29]:

```
print('Training labels shape', train_labels.shape)
```

```
Training labels shape (456,)
```

In [30]:

```
print('Testing features shape', test_features.shape)
```

```
Testing features shape (153, 3)
```

In [31]:

```
print('Training labels shape', test_labels.shape)
```

```
Training labels shape (153,)
```

In [32]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [33]:

```
rf = RandomForestRegressor(n_estimators = 1000)
```

In [34]:

```
rf.fit(train_features, train_labels)
```

Out[34]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=1000,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

In [35]:

```
prediction = rf.predict(test_features)
```

In [36]:

```
errors = abs(prediction - test_labels)
```

In [37]:

```
print('Mean Absolute Error', round(np.mean(errors),2), 'degress.')
```

```
Mean Absolute Error 1.88 degress.
```

In [38]:

```
mape = 100*(errors/test_labels)
```

In [39]:

```
accuracy = 100 - np.mean(mape)
print('Accuracy: ', round(accuracy,2), '%.')
```

```
Accuracy:  47.96 %.
```

In [61]:

```
a = np.array([2020,11,26])
rf.predict(a.reshape(1,3))
```

Out[61]:

```
array([9.77901346])
```

In [41]:

```
datetemp
```

Out[41]:

	date	value
0	2019-01-01	4.839695
1	2019-01-02	2.189279
2	2019-01-03	0.052195
3	2019-01-04	1.189695
4	2019-01-05	1.893445
5	2019-01-06	3.028862
6	2019-01-07	2.843445
7	2019-01-08	4.344279
8	2019-01-09	2.552612
9	2019-01-10	0.374695
10	2019-01-11	0.575112
11	2019-01-12	3.666362
12	2019-01-13	6.430529
13	2019-01-14	5.473862
14	2019-01-15	3.308445
15	2019-01-16	4.222195
16	2019-01-17	6.020945
17	2019-01-18	1.087612
18	2019-01-19	0.589279
19	2019-01-20	2.991779
20	2019-01-21	0.123029
21	2019-01-22	-0.386138
22	2019-01-23	1.235945
23	2019-01-24	-1.514471
24	2019-01-25	-0.898221
25	2019-01-26	4.507612
26	2019-01-27	6.192195

	date	value
26	2019-01-27	6.102195
27	2019-01-28	2.280945
28	2019-01-29	1.873029
29	2019-01-30	1.378862
...	...	...
579	2020-08-02	22.233446
580	2020-08-03	17.924279
581	2020-08-04	16.341362
582	2020-08-05	18.532195
583	2020-08-06	20.753446
584	2020-08-07	23.731363
585	2020-08-08	25.490946
586	2020-08-09	26.229280
587	2020-08-10	26.653029
588	2020-08-11	27.071779
589	2020-08-12	27.361779
590	2020-08-13	25.960946
591	2020-08-14	21.561779
592	2020-08-15	22.557196
593	2020-08-16	23.510946
594	2020-08-17	20.805946
595	2020-08-18	19.835946
596	2020-08-19	22.083029
597	2020-08-20	25.705529
598	2020-08-21	26.834696
599	2020-08-22	22.198029
600	2020-08-23	19.730112
601	2020-08-24	19.376363
602	2020-08-25	20.503862
603	2020-08-26	21.866363
604	2020-08-27	20.256779
605	2020-08-28	18.643863
606	2020-08-29	15.507196
607	2020-08-30	13.490946
608	2020-08-31	15.649696

609 rows × 2 columns

In [62]:

```
pred_date = '2019-11-26'
```

In [63]:

```
user_date = datetime.strptime(pred_date,'%Y-%m-%d').date()
```

In [64]:

```
datetemp[datetemp['date'] == user_date]
```

Out[64]:

	date	value
329	2019-11-26	10.536362

In [ ]: