In [3]:

```
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt # library plotting
import seaborn as sns # statistical plotting
import pylab as p
```

In [25]:

```
pip install statsmodels
```

```
Collecting statsmodels
  Downloading statsmodels-0.13.5-cp37-cp37m-win_amd64.whl (9.1 MB)
     ---------------------------------------- 9.1/9.1 MB 2.2 MB/s eta 0:00:00
Requirement already satisfied: pandas>=0.25 in c:\users\lenovo\anaconda3\envs
\rstudio\lib\site-packages (from statsmodels) (1.3.5)
Collecting patsy>=0.5.2 (from statsmodels)
  Downloading patsy-0.5.3-py2.py3-none-any.whl (233 kB)
     -------------------------------------- 233.8/233.8 kB 2.9 MB/s eta 0:00:
00
Requirement already satisfied: packaging>=21.3 in c:\users\lenovo\anaconda3\e
nvs\rstudio\lib\site-packages (from statsmodels) (21.3)
Requirement already satisfied: scipy>=1.3 in c:\users\lenovo\anaconda3\envs\r
studio\lib\site-packages (from statsmodels) (1.7.3)
Requirement already satisfied: numpy>=1.17 in c:\users\lenovo\anaconda3\envs
\rstudio\lib\site-packages (from statsmodels) (1.21.6)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\lenovo\an
aconda3\envs\rstudio\lib\site-packages (from packaging>=21.3->statsmodels)
(3.0.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\lenovo\anac
onda3\envs\rstudio\lib\site-packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\lenovo\anaconda3\envs
\rstudio\lib\site-packages (from pandas>=0.25->statsmodels) (2023.3)
Requirement already satisfied: six in c:\users\lenovo\anaconda3\envs\rstudio
\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Installing collected packages: patsy, statsmodels
Successfully installed patsy-0.5.3 statsmodels-0.13.5
Note: you may need to restart the kernel to use updated packages.
```

In [26]:

```
from sklearn.metrics import mean_squared_error
from pandas.plotting import lag_plot
from statsmodels.graphics.tsaplots import plot_acf
import statsmodels.graphics.tsaplots as tsa_plots
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.formula.api as smf
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [4]:

```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\lenovo\anaconda3\envs\rst
udio\lib\site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in c:\users\lenovo\anaconda3\envs\r
studio\lib\site-packages (from openpyxl) (1.1.0)
```

In [5]:

```
Silchar = pd.read_csv('SilcharFloods2022.csv')
```

In [6]:

```
Silchar.head(6)
```

Out[6]:

	Time	Water Level	Trend	Difference	Rainfall	Prog.
0	1:00	21.58	Steady	0.0	3.0	1960.6
1	2:00	21.58	Steady	0.0	3.0	1960.6
2	3:00	21.58	Steady	0.0	3.0	1960.6
3	4:00	21.58	Steady	0.0	3.0	1960.6
4	5:00	21.58	Steady	0.0	3.0	1960.6
5	6:00	21.58	Steady	0.0	3.0	1960.6

In [8]:

```
Silchar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Time         23 non-null     object 
 1   Water Level  14 non-null     float64
 2   Trend        14 non-null     object 
 3   Difference   14 non-null     float64
 4   Rainfall     14 non-null     float64
 5   Prog.        14 non-null     float64
dtypes: float64(4), object(2)
memory usage: 1.2+ KB
```

In [9]:

```
Silchar.shape
```

Out[9]:

```
(23, 6)
```

In [10]:

```
Silchar.dtypes
```

Out[10]:

```
Time            object
Water Level    float64
Trend           object
Difference     float64
Rainfall       float64
Prog.          float64
dtype: object
```

In [11]:
```
Silchar.describe()
```

Out[11]:

	Water Level	Difference	Rainfall	Prog.
count	14.000000	14.000000	14.0	14.0
mean	21.576429	0.214286	3.0	1960.6
std	0.006333	0.425815	0.0	0.0
min	21.570000	0.000000	3.0	1960.6
25%	21.570000	0.000000	3.0	1960.6
50%	21.580000	0.000000	3.0	1960.6
75%	21.580000	0.000000	3.0	1960.6
max	21.590000	1.000000	3.0	1960.6

In [13]:
```
Silchar.isnull().sum()
```

Out[13]:
```
Time           0
Water Level    9
Trend          9
Difference     9
Rainfall       9
Prog.          9
dtype: int64
```

In [14]:
```
Silchar
```

Out[14]:

	Time	Water Level	Trend	Difference	Rainfall	Prog.
0	1:00	21.58	Steady	0.0	3.0	1960.6
1	2:00	21.58	Steady	0.0	3.0	1960.6
2	3:00	21.58	Steady	0.0	3.0	1960.6
3	4:00	21.58	Steady	0.0	3.0	1960.6
4	5:00	21.58	Steady	0.0	3.0	1960.6
5	6:00	21.58	Steady	0.0	3.0	1960.6
6	7:00	21.57	Falling	1.0	3.0	1960.6
7	8:00	21.57	Steady	0.0	3.0	1960.6
8	9:00	21.57	Steady	0.0	3.0	1960.6
9	10:00	21.57	Steady	0.0	3.0	1960.6
10	11:00	21.57	Steady	0.0	3.0	1960.6
11	12:00	21.57	Steady	0.0	3.0	1960.6
12	13:00	21.58	Rising	1.0	3.0	1960.6
13	14:00	21.59	Rising	1.0	3.0	1960.6
14	15:00	NaN	NaN	NaN	NaN	NaN
15	16:00	NaN	NaN	NaN	NaN	NaN
16	17:00	NaN	NaN	NaN	NaN	NaN
17	18:00	NaN	NaN	NaN	NaN	NaN
18	19:00	NaN	NaN	NaN	NaN	NaN
19	20:00	NaN	NaN	NaN	NaN	NaN
20	21:00	NaN	NaN	NaN	NaN	NaN
21	22:00	NaN	NaN	NaN	NaN	NaN
22	23:00	NaN	NaN	NaN	NaN	NaN

In [15]:
```
Silchar.dropna(inplace=True)
```

In [16]:
```
Silchar
```

Out[16]:

	Time	Water Level	Trend	Difference	Rainfall	Prog.
0	1:00	21.58	Steady	0.0	3.0	1960.6
1	2:00	21.58	Steady	0.0	3.0	1960.6
2	3:00	21.58	Steady	0.0	3.0	1960.6
3	4:00	21.58	Steady	0.0	3.0	1960.6
4	5:00	21.58	Steady	0.0	3.0	1960.6
5	6:00	21.58	Steady	0.0	3.0	1960.6
6	7:00	21.57	Falling	1.0	3.0	1960.6
7	8:00	21.57	Steady	0.0	3.0	1960.6
8	9:00	21.57	Steady	0.0	3.0	1960.6
9	10:00	21.57	Steady	0.0	3.0	1960.6
10	11:00	21.57	Steady	0.0	3.0	1960.6
11	12:00	21.57	Steady	0.0	3.0	1960.6
12	13:00	21.58	Rising	1.0	3.0	1960.6
13	14:00	21.59	Rising	1.0	3.0	1960.6

In [17]:
```
Silchar=Silchar.rename(columns={'Water Level':'water_level','Prog.':'progressi
```

In [18]:
```
Silchar
```

Out[18]:

	Time	water_level	Trend	Difference	Rainfall	progressive
0	1:00	21.58	Steady	0.0	3.0	1960.6
1	2:00	21.58	Steady	0.0	3.0	1960.6
2	3:00	21.58	Steady	0.0	3.0	1960.6
3	4:00	21.58	Steady	0.0	3.0	1960.6
4	5:00	21.58	Steady	0.0	3.0	1960.6
5	6:00	21.58	Steady	0.0	3.0	1960.6
6	7:00	21.57	Falling	1.0	3.0	1960.6
7	8:00	21.57	Steady	0.0	3.0	1960.6
8	9:00	21.57	Steady	0.0	3.0	1960.6
9	10:00	21.57	Steady	0.0	3.0	1960.6
10	11:00	21.57	Steady	0.0	3.0	1960.6
11	12:00	21.57	Steady	0.0	3.0	1960.6
12	13:00	21.58	Rising	1.0	3.0	1960.6
13	14:00	21.59	Rising	1.0	3.0	1960.6

In [19]:

```
Silchar=Silchar[['Time','water_level']]
```

In [20]:

```
Silchar
```

Out[20]:

	Time	water_level
0	1:00	21.58
1	2:00	21.58
2	3:00	21.58
3	4:00	21.58
4	5:00	21.58
5	6:00	21.58
6	7:00	21.57
7	8:00	21.57
8	9:00	21.57
9	10:00	21.57
10	11:00	21.57
11	12:00	21.57
12	13:00	21.58
13	14:00	21.59

In [21]:

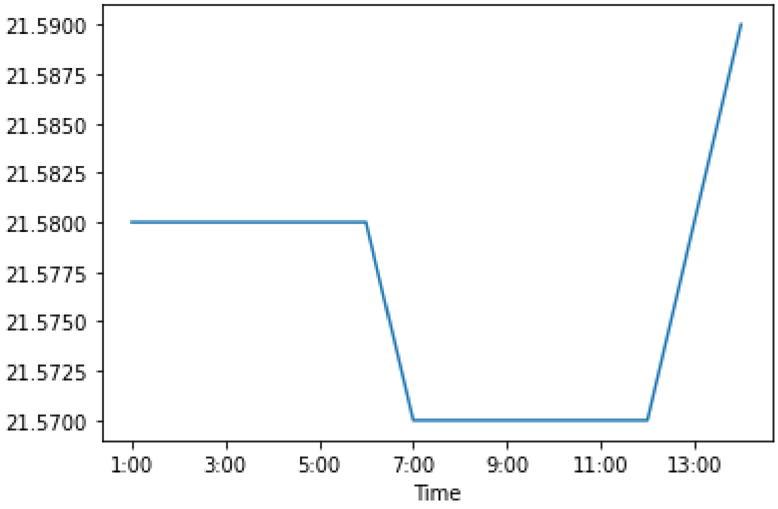
```
Silchar.set_index('Time',inplace=True)
```

In [22]:

```
Silchar.water_level.plot()
```

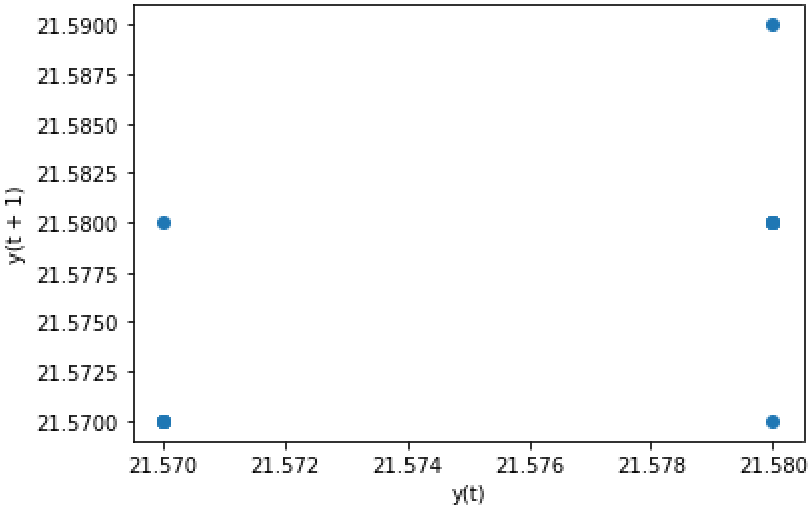
Out[22]:

```
<AxesSubplot:xlabel='Time'>
```

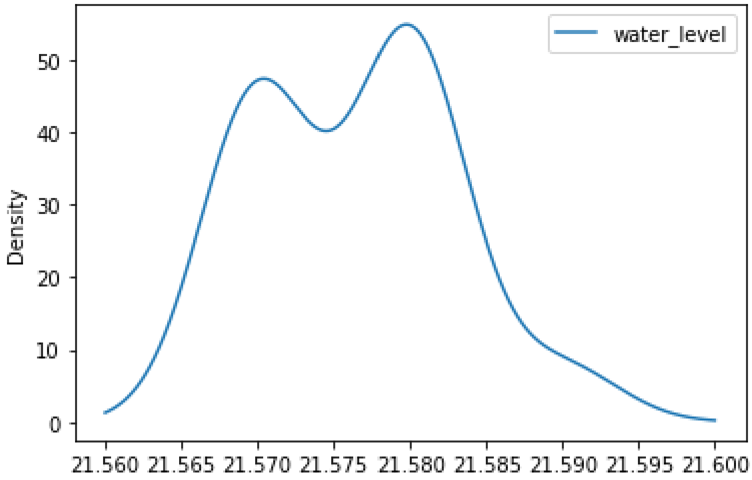


```
In [27]: lag_plot(Silchar)
```

```
Out[27]: <AxesSubplot:xlabel='y(t)', ylabel='y(t + 1)'>
```

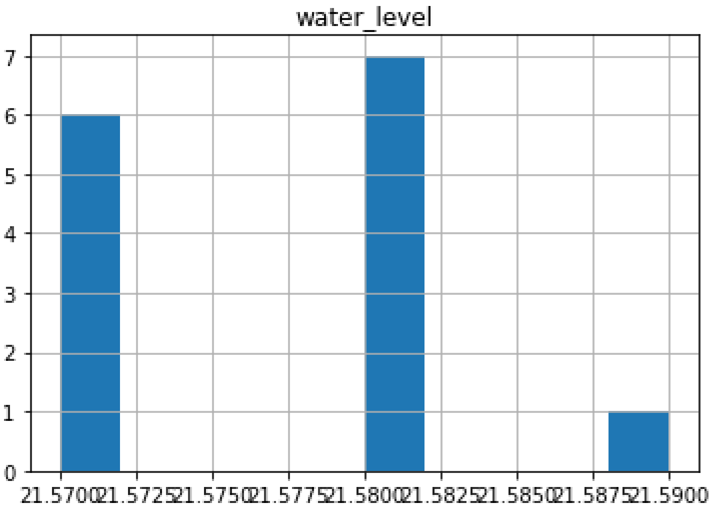


```
In [28]: Silchar.plot(kind='kde')
         plt.show()
```



In [29]:

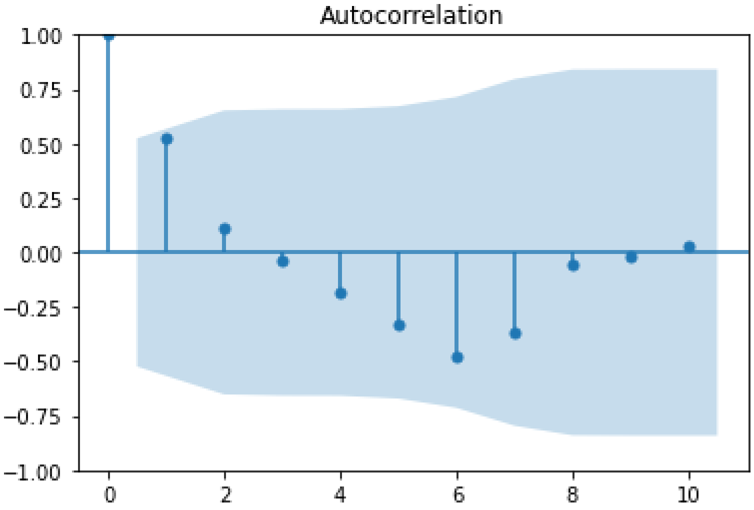
```
Silchar.hist()
plt.show()
```

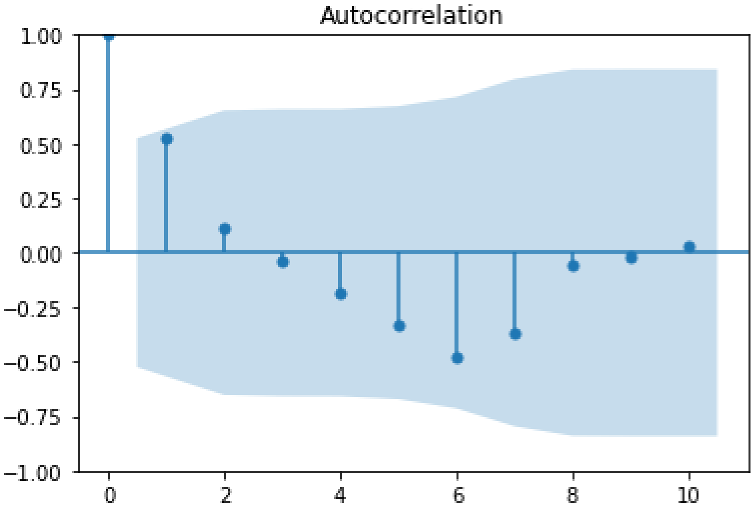


In [30]:

```
plot_acf(Silchar,lags=10)
```

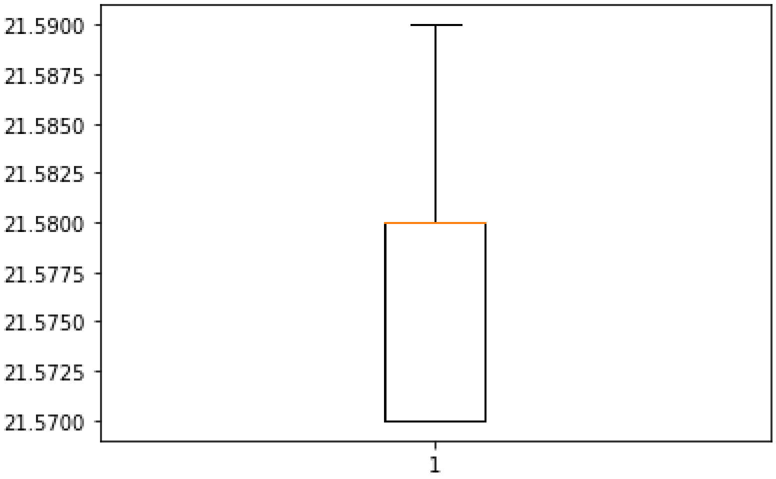
Out[30]:





In [31]:
```
plt.boxplot(Silchar)
```

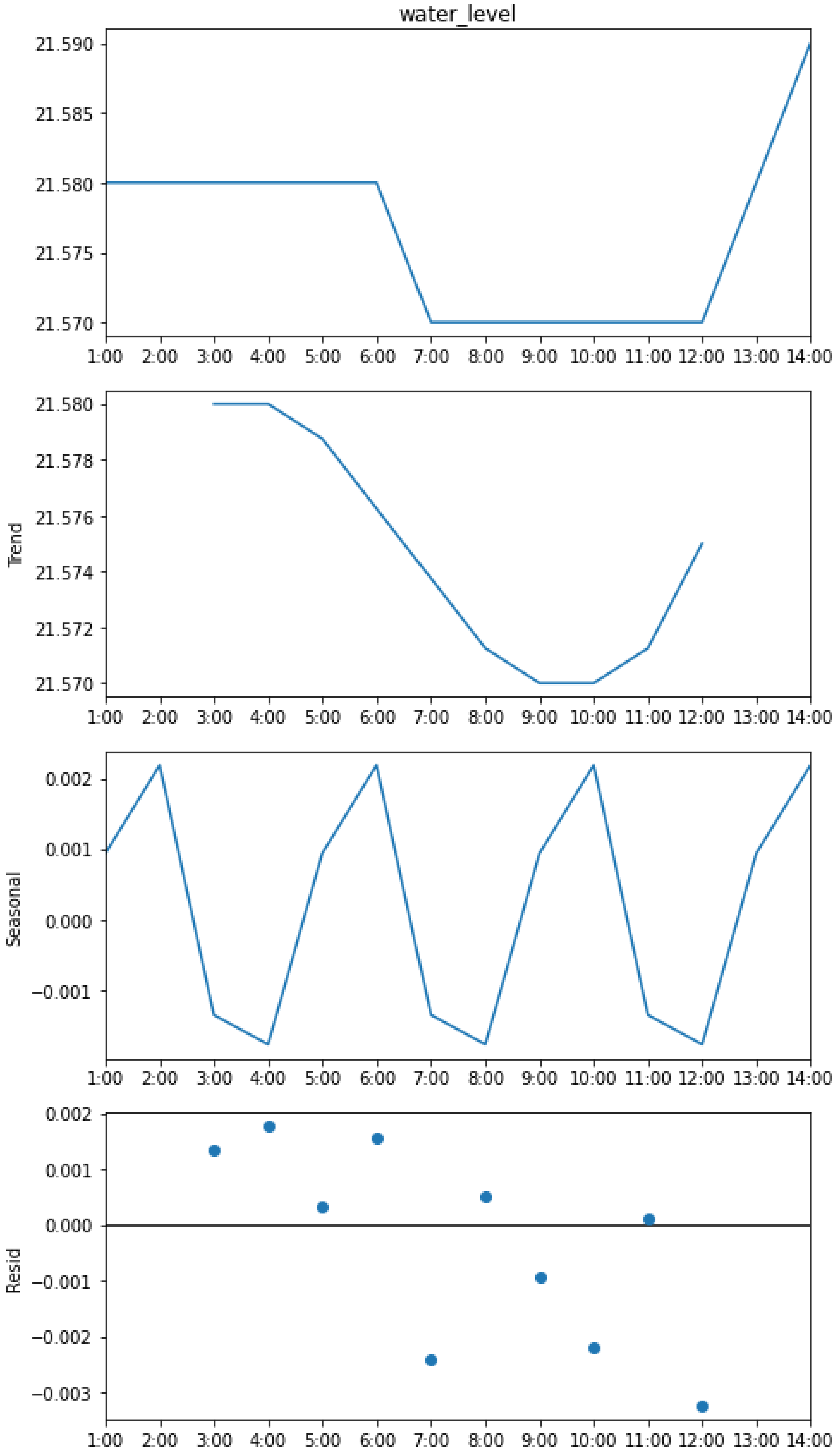
Out[31]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ca0ba8b860>,
  <matplotlib.lines.Line2D at 0x1ca0ba8bba8>],
 'caps': [<matplotlib.lines.Line2D at 0x1ca0ba8bef0>,
  <matplotlib.lines.Line2D at 0x1ca0bb10748>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ca0ba8b128>],
 'medians': [<matplotlib.lines.Line2D at 0x1ca0ba407b8>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ca0ba40e80>],
 'means': []}
```

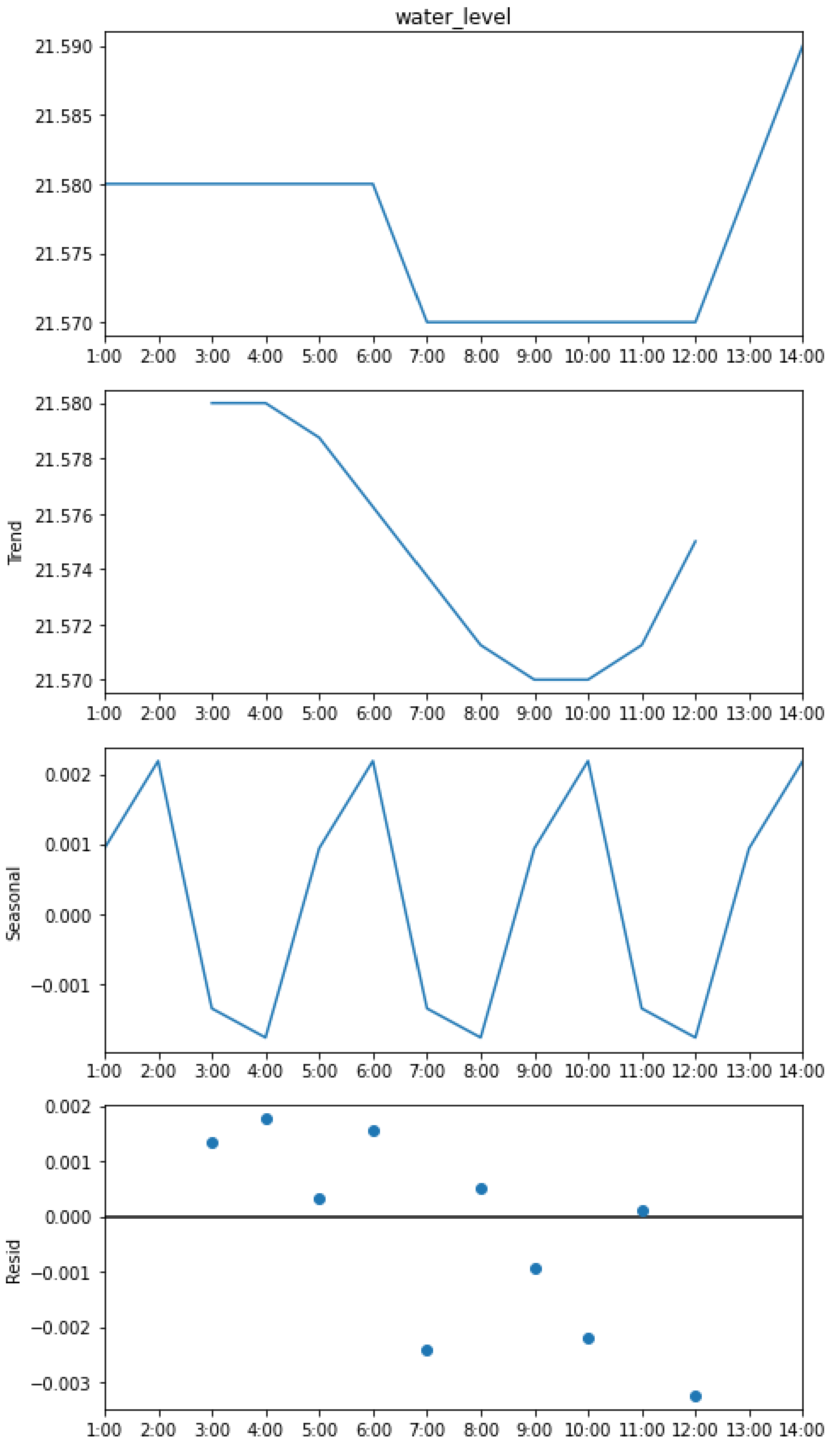


In [32]:

```
plt.rcParams['figure.figsize']=(7,12)
decompose_ts_add = seasonal_decompose(Silchar.water_level, model = "additive",
decompose_ts_add.plot()
```

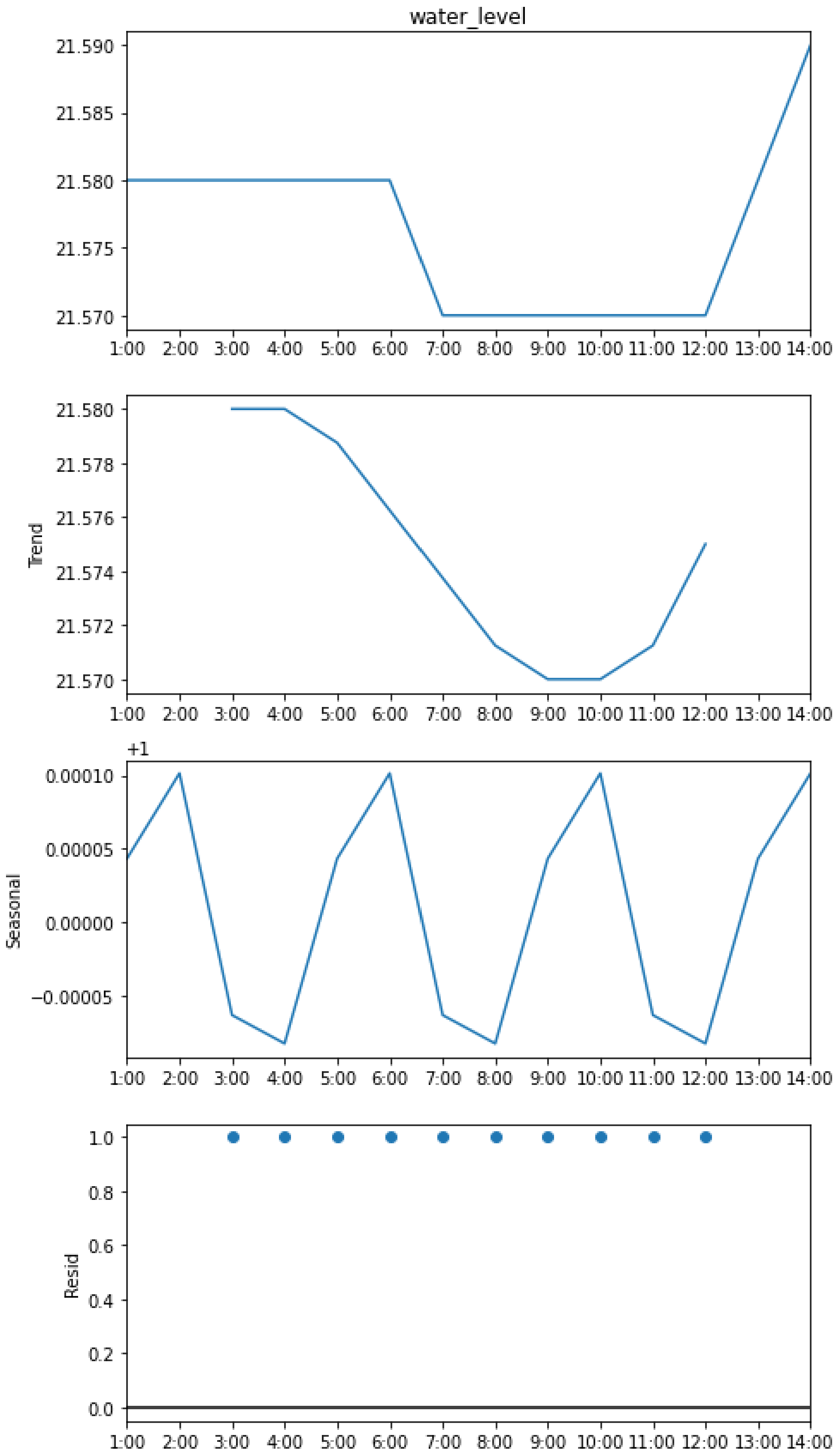
Out[32]:

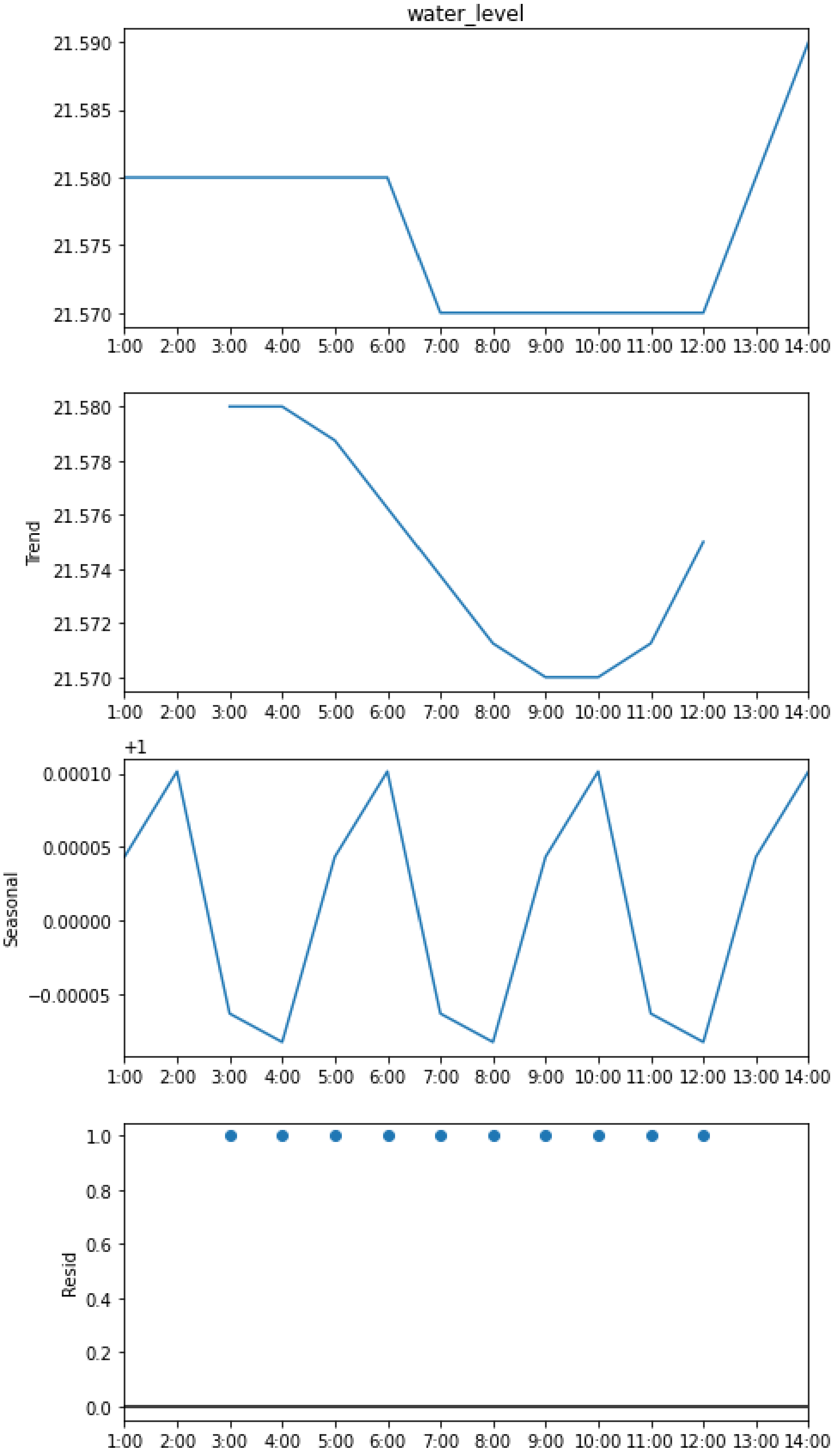



water_level
21.590
21.585
21.580
21.575
21.570
1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00 12:00 13:00 14:00
Trend
21.580
21.578
21.576
21.574
21.572
21.570
1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00 12:00 13:00 14:00
Seasonal
0.002
0.001
0.000
−0.001
1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00 12:00 13:00 14:00
Resid
0.002
0.001
0.000
−0.001
−0.002
−0.003
1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00 12:00 13:00 14:00

In [34]:
```
decompose_ts_mul = seasonal_decompose(Silchar.water_level, model = "multiplica
decompose_ts_mul.plot()
```

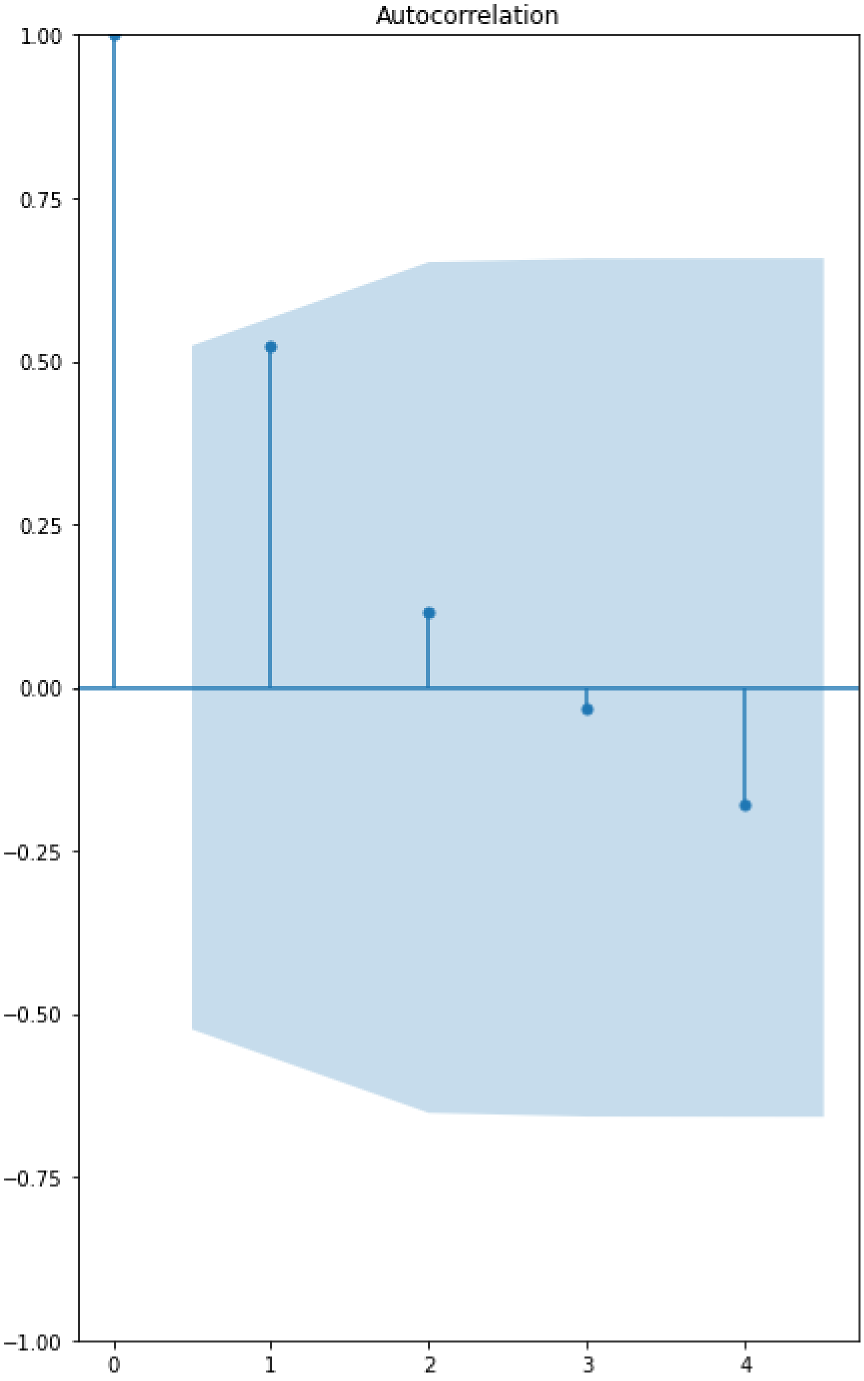
Out[34]:

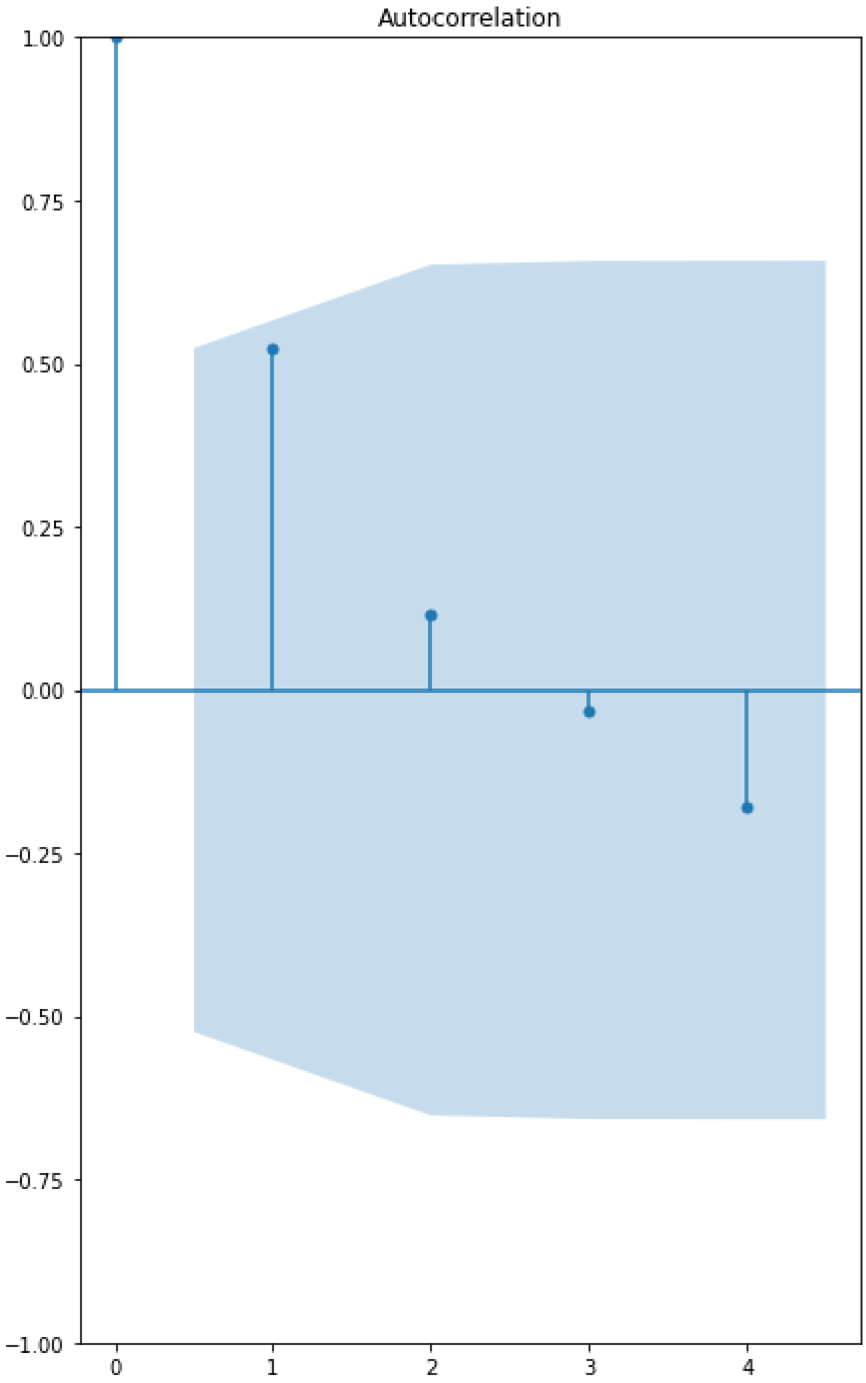



water_level
21.590
21.585
21.580
21.575
21.570
1:00
2:00
3:00
4:00
5:00
6:00
7:00
8:00
9:00
10:00
11:00
12:00
13:00
14:00
Trend
21.580
21.578
21.576
21.574
21.572
21.570
Seasonal
+1
0.00010
0.00005
0.00000
−0.00005
Resid
1.0
0.8
0.6
0.4
0.2
0.0

In [35]:
```
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(Silchar.water_level, lags = 4)
```

Out[35]:



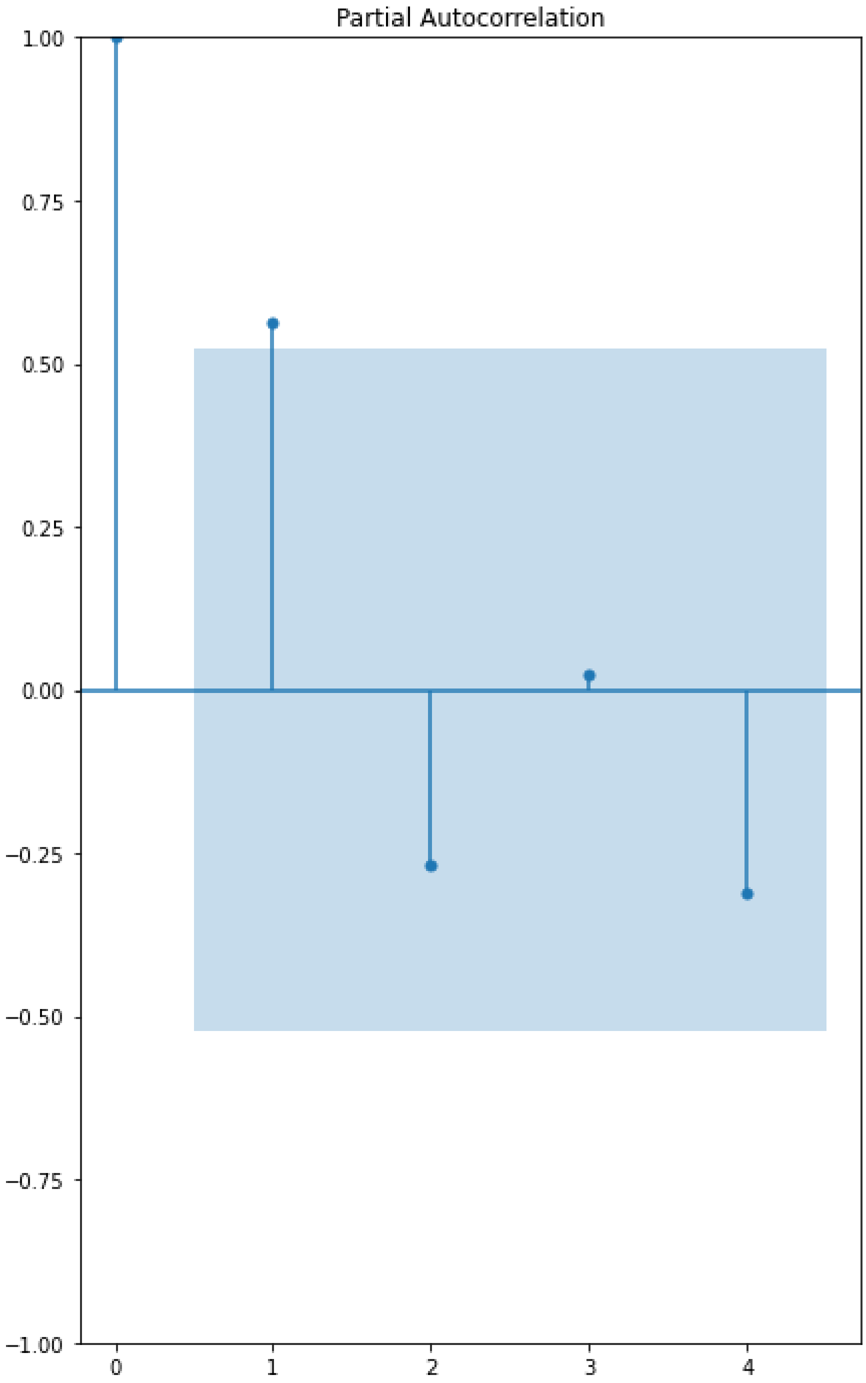

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4

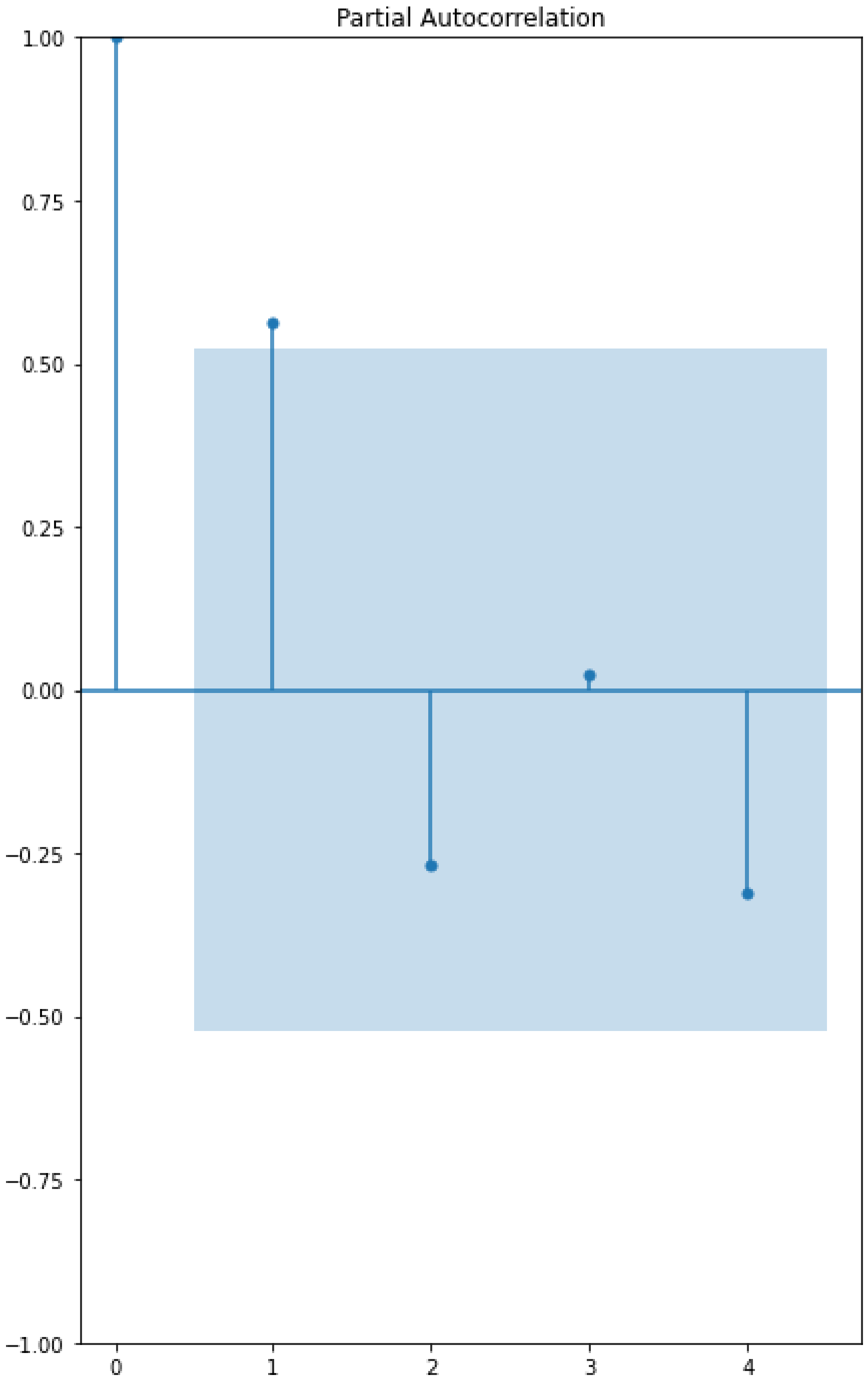
In [36]:

```
tsa_plots.plot_pacf(Silchar.water_level, lags=4)
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\graphics
\tsaplots.py:353: FutureWarning: The default method 'yw' can produce PACF val
ues outside of the [-1,1] interval. After 0.13, the default will change touna
djusted Yule-Walker ('ywm'). You can use this method now by setting method='y
wm'.
  FutureWarning,
```

Out[36]:





In [37]:
```
Train = Silchar.head(6)
Test = Silchar.tail(7)
```

In [38]:
```
def MAPE(pred,org):
    temp = np.abs((pred-org)/org)*100
    return np.mean(temp)
```

In [39]:

```
ses_model = SimpleExpSmoothing(Train["water_level"]).fit(smoothing_level=0.2)
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_ses,Test.water_level)
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\base
\tsa_model.py:471: ValueWarning: No frequency information was provided, so in
ferred frequency H will be used.
  self._init_dates(dates, freq)
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holt
winters\model.py:1409: RuntimeWarning: divide by zero encountered in log
  aic = self.nobs * np.log(sse / self.nobs) + k * 2
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holt
winters\model.py:1415: RuntimeWarning: divide by zero encountered in log
  bic = self.nobs * np.log(sse / self.nobs) + k * np.log(self.nobs)
```

Out[39]:

```
nan
```

In [40]:

```
hw_model = Holt(Train["water_level"]).fit()
pred_hw = hw_model.predict(start = Test.index[0], end = Test.index[-1])
MAPE(pred_hw, Test.water_level)
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\base
\tsa_model.py:471: ValueWarning: No frequency information was provided, so in
ferred frequency H will be used.
  self._init_dates(dates, freq)
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holt
winters\model.py:1414: RuntimeWarning: invalid value encountered in double_sc
alars
  aicc = aic + aicc_penalty
```

Out[40]:

```
nan
```

In [41]:

```
hwe_model_add_add = ExponentialSmoothing(Train["water_level"], seasonal = "add
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0], end = Test
MAPE(pred_hwe_add_add, Test.water_level)
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\base
\tsa_model.py:471: ValueWarning: No frequency information was provided, so in
ferred frequency H will be used.
  self._init_dates(dates, freq)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13536/14934474.py in <module>
----> 1 hwe_model_add_add = ExponentialSmoothing(Train["water_level"], seasonal = "add", trend = "add", seasonal_periods = 4).fit()
      2 pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0], end = Test.index[-1])
      3 MAPE(pred_hwe_add_add, Test.water_level)

~\anaconda3\envs\rstudio\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    205                 else:
    206                     kwargs[new_arg_name] = new_arg_value
--> 207             return func(*args, **kwargs)
    208 
    209         return cast(F, wrapper)

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.py in __init__(self, endog, trend, damped_trend, seasonal, seasonal_periods, initialization_method, initial_level, initial_trend, initial_seasonal, use_boxcox, bounds, dates, freq, missing)
    290         self._lambda = np.nan
    291         self._y = self._boxcox()
--> 292         self._initialize()
    293         self._fixed_parameters = {}
    294 

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.py in _initialize(self)
    428         elif self._initialization_method == "estimated":
    429             if self.nobs < 10 + 2 * (self.seasonal_periods // 2):
--> 430                 return self._initialize_simple()
    431             else:
    432                 return self._initialize_heuristic()

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.py in _initialize_simple(self)
    436         seasonal = self.seasonal if self.has_seasonal else False
    437         lvl, trend, seas = _initialization_simple(
--> 438             self._y, trend, seasonal, self.seasonal_periods
    439         )
    440         self._initial_level = lvl

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\exponential_smoothing\initialization.py in _initialization_simple(endog, trend, seasonal, seasonal_periods)
     24     else:
     25         if nobs < 2 * seasonal_periods:
---> 26             raise ValueError('Cannot compute initial seasonals using'
     27                              ' heuristic method with less than two full'
     28                              ' seasonal cycles in the data.')

ValueError: Cannot compute initial seasonals using heuristic method with less than two full seasonal cycles in the data.
```

In [42]:

```
hwe_model_mul_add = ExponentialSmoothing(Train["water_level"], seasonal = "mul
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0], end = Test
MAPE(pred_hwe_mul_add, Test.water_level)
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\base
\tsa_model.py:471: ValueWarning: No frequency information was provided, so in
ferred frequency H will be used.
  self._init_dates(dates, freq)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13536/2515230076.py in <module>
----> 1 hwe_model_mul_add = ExponentialSmoothing(Train["water_level"], season
al = "mul", trend = "add", seasonal_periods = 4).fit()
      2 pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0], e
nd = Test.index[-1])
      3 MAPE(pred_hwe_mul_add, Test.water_level)

~\anaconda3\envs\rstudio\lib\site-packages\pandas\util\_decorators.py in wrap
per(*args, **kwargs)
    205                 else:
    206                     kwargs[new_arg_name] = new_arg_value
--> 207             return func(*args, **kwargs)
    208 
    209         return cast(F, wrapper)

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.
py in __init__(self, endog, trend, damped_trend, seasonal, seasonal_periods,
initialization_method, initial_level, initial_trend, initial_seasonal, use_bo
xcox, bounds, dates, freq, missing)
    290         self._lambda = np.nan
    291         self._y = self._boxcox()
--> 292         self._initialize()
    293         self._fixed_parameters = {}
    294 

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.
py in _initialize(self)
    428         elif self._initialization_method == "estimated":
    429             if self.nobs < 10 + 2 * (self.seasonal_periods // 2):
--> 430                 return self._initialize_simple()
    431             else:
    432                 return self._initialize_heuristic()

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\holtwinters\model.
py in _initialize_simple(self)
    436         seasonal = self.seasonal if self.has_seasonal else False
    437         lvl, trend, seas = _initialization_simple(
--> 438             self._y, trend, seasonal, self.seasonal_periods
    439         )
    440         self._initial_level = lvl

~\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\exponential_smooth
ing\initialization.py in _initialization_simple(endog, trend, seasonal, seaso
nal_periods)
     24     else:
     25         if nobs < 2 * seasonal_periods:
---> 26             raise ValueError('Cannot compute initial seasonals using'
     27                              ' heuristic method with less than two fu
ll'
     28                              ' seasonal cycles in the data.')

ValueError: Cannot compute initial seasonals using heuristic method with less
than two full seasonal cycles in the data.
```

In [44]:

```
hwe_model_add_add = ExponentialSmoothing(Silchar["water_level"], seasonal = "a
```

```
C:\Users\Lenovo\anaconda3\envs\rstudio\lib\site-packages\statsmodels\tsa\base
\tsa_model.py:471: ValueWarning: No frequency information was provided, so in
ferred frequency H will be used.
  self._init_dates(dates, freq)
```

In [ ]: