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

cake = pd.read_csv('xnas-itch-20240822.mbo.CAKE.csv',parse_dates=['ts_re
```

```
import matplotlib.pyplot as plt
```

```
cake.columns
```

```
Index(['ts_recv', 'ts_event', 'rtype', 'publisher_id', 'instrument_id'
       'action', 'side', 'price', 'size', 'channel_id', 'order_id', 'f
       'ts_in_delta', 'sequence', 'symbol'],
      dtype='object')
```

```
cake['action'].unique()
```

```
array(['R', 'A', 'C', 'T', 'F'], dtype=object)
```

```
cake_trades=pd.read_csv('xnas-itch-20240822.tbbo.CAKE.csv',parse_dates=[
```

```
cake_trades.head()
```

	ts_recv	ts_event	rtype	publisher_id	instrument_id	action	side	c
0	2024-08-22 13:28:00.059387613+00:00	2024-08-22 13:28:00.059217193+00:00	1	2	2639	T	N	(
1	2024-08-22 13:28:00.059387613+00:00	2024-08-22 13:28:00.059217547+00:00	1	2	2639	T	N	(
2	2024-08-22 13:30:00.918500023+00:00	2024-08-22 13:30:00.918318809+00:00	1	2	2639	T	N	(
3	2024-08-22 13:30:05.131885094+00:00	2024-08-22 13:30:05.131717397+00:00	1	2	2639	T	N	(
4	2024-08-22 13:30:45.269318156+00:00	2024-08-22 13:30:45.269151493+00:00	1	2	2639	T	A	(

```
                              ts_recv                           ts_e
0 2024-08-22 13:28:00.059387613+00:00 2024-08-22 13:28:00.059217193+0
1 2024-08-22 13:28:00.059387613+00:00 2024-08-22 13:28:00.059217547+0
2 2024-08-22 13:30:00.918500023+00:00 2024-08-22 13:30:00.918318809+0
3 2024-08-22 13:30:05.131885094+00:00 2024-08-22 13:30:05.131717397+0
4 2024-08-22 13:30:45.269318156+00:00 2024-08-22 13:30:45.269151493+0

   rtype  publisher_id  instrument_id action side  depth  price  size
0      1             2           2639      T    N      0  38.76    60
1      1             2           2639      T    N      0  38.76    40
2      1             2           2639      T    N      0  38.66  3061
3      1             2           2639      T    N      0  38.78    34
4      1             2           2639      T    A      0  38.75     1

   ts_in_delta  sequence  bid_px_00  ask_px_00  bid_sz_00  ask_sz_00
0       170420  14286651      38.00      40.53          1       1000
1       170066  14286652      38.00      40.53          1       1000
2       181214  17284072      38.64      39.36         95        200
3       167697  17937927      38.60      38.97        234        100
4       166643  21303475      38.75      38.97          1        100
```

```
cake_trades.columns
```

```
Index(['ts_recv', 'ts_event', 'rtype', 'publisher_id', 'instrument_id',
       'action', 'side', 'depth', 'price', 'size', 'flags', 'ts_in_del
       'sequence', 'bid_px_00', 'ask_px_00', 'bid_sz_00', 'ask_sz_00',
       'bid_ct_00', 'ask_ct_00', 'symbol'],
      dtype='object')
```

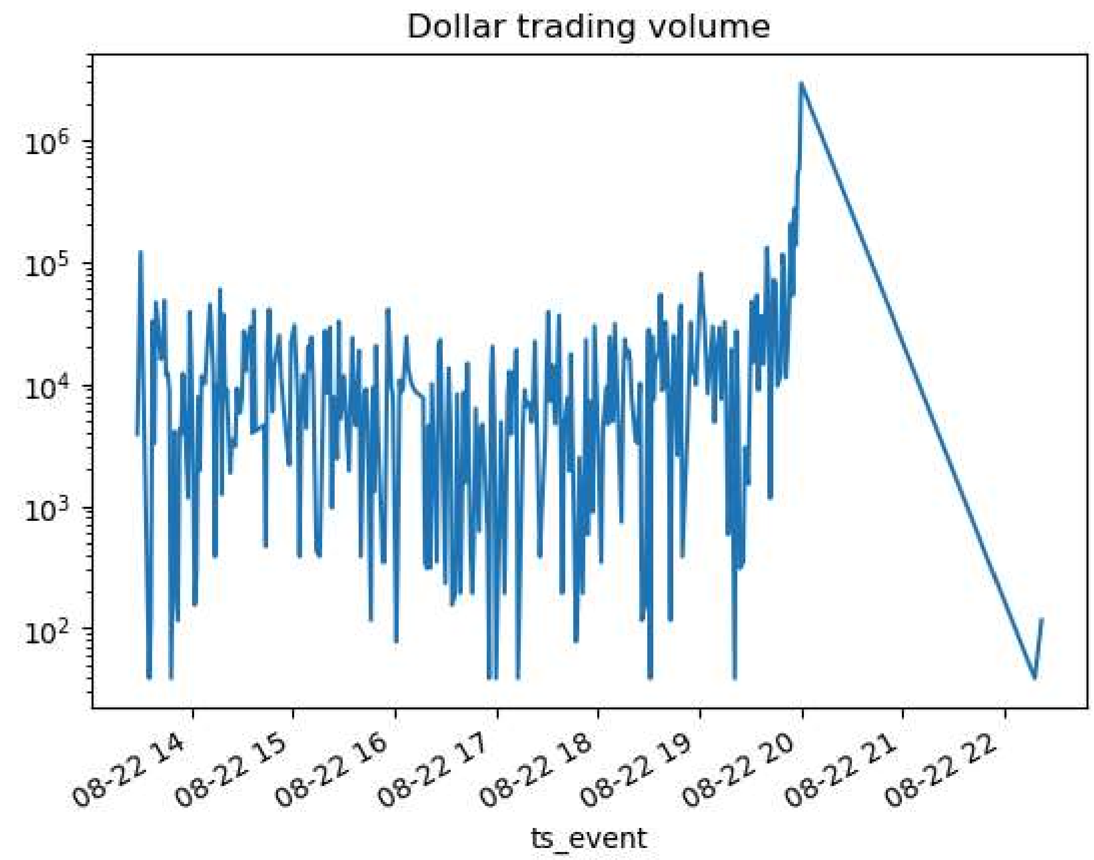
```
#a.
cake_trades['dollar_vol']=cake_trades['price']*cake_trades['size']
```

```
dtv=cake_trades.groupby(cake_trades['ts_event'].dt.floor('min'))['dollar
```

```
dtv.plot(title='Dollar trading volume',logy=True)
```

```
<Axes: title={'center': 'Dollar trading volume'}, xlabel='ts_event'>
```

Download



```
<Figure size 640x480 with 1 Axes>
```

```
dtv.describe()
```

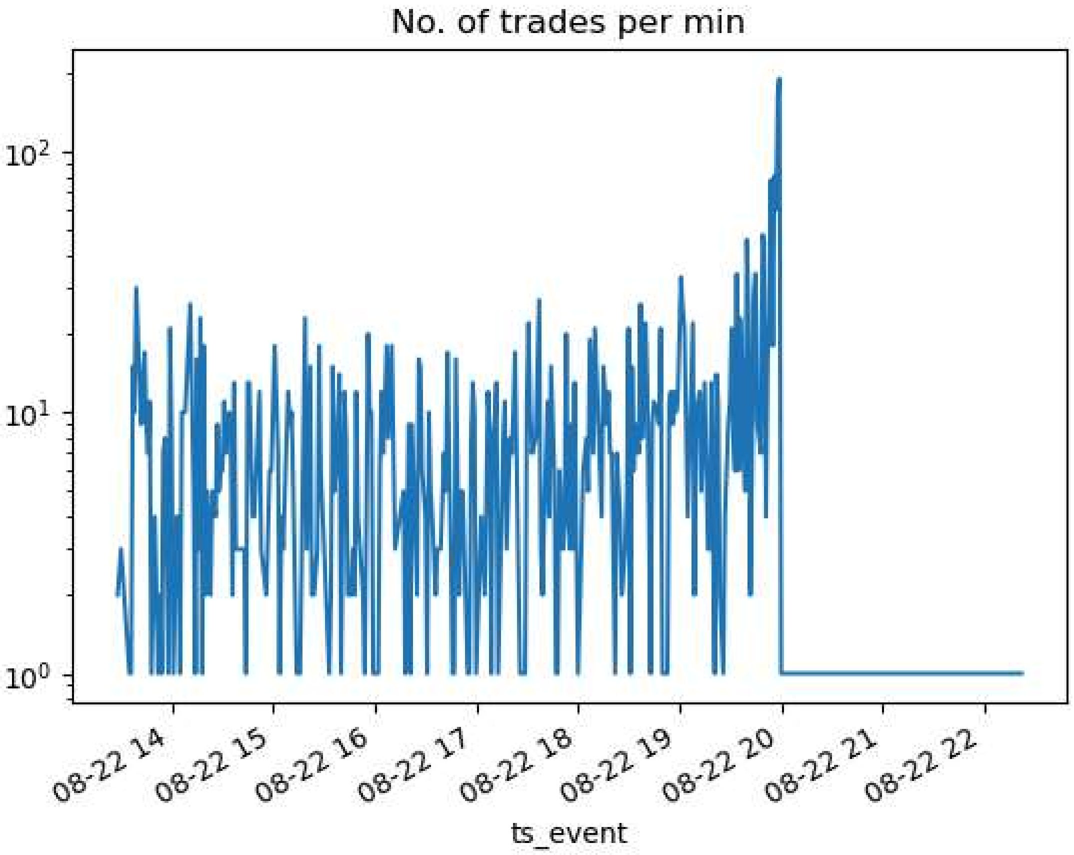
```
count    2.970000e+02
mean     2.912566e+04
std      1.748573e+05
min      3.867000e+01
25%      2.444520e+03
50%      8.180130e+03
75%      1.969411e+04
max      2.895873e+06
Name: dollar_vol, dtype: float64
```

```
#b.
trade_counts_min=cake_trades.groupby(cake_trades['ts_event'].dt.floor('m
```

```
trade_counts_min.plot(title='No. of trades per min',logy=True)
```

```
<Axes: title={'center': 'No. of trades per min'}, xlabel='ts_event'>
```

Download



```
<Figure size 640x480 with 1 Axes>
```

```
trade_counts_min.describe()
```

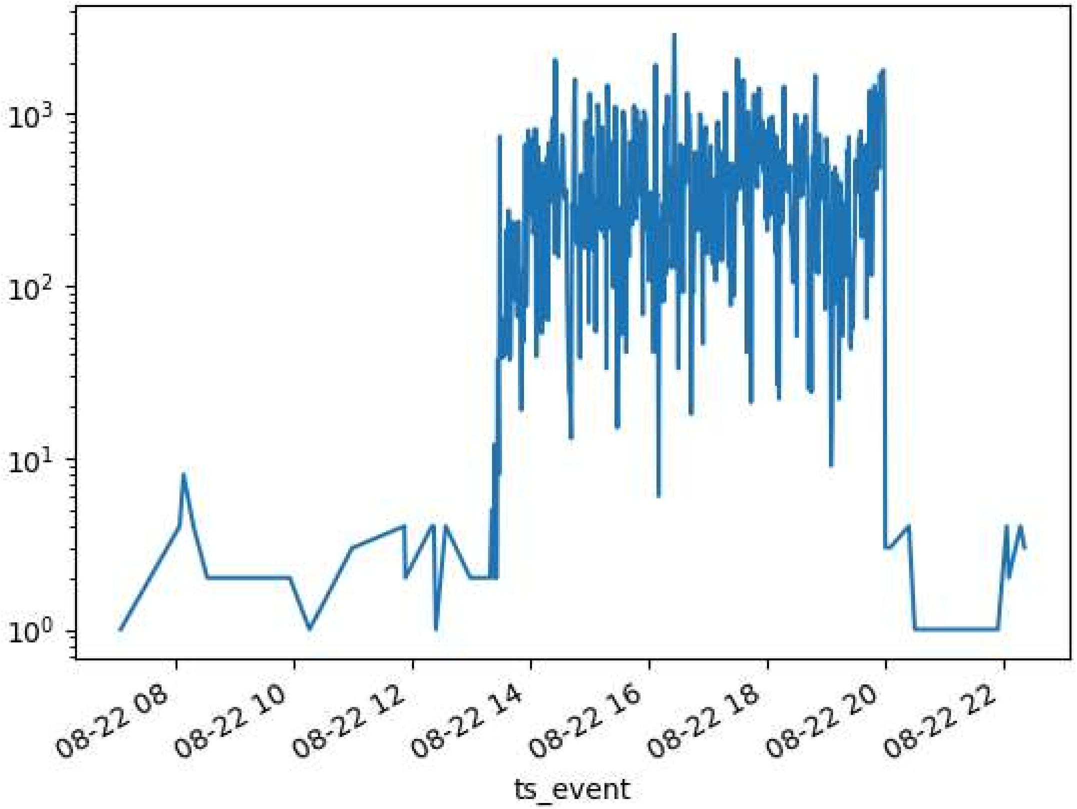
```
count    297.000000
mean      10.218855
std       17.198638
min        1.000000
25%        3.000000
50%        7.000000
75%       12.000000
max      190.000000
Name: sequence, dtype: float64
```

```
order_counts_min=cake.groupby(cake['ts_event'].dt.floor('min'))['sequenc
```

```
order_counts_min.plot(logy=True)
```

```
<Axes: xlabel='ts_event'>
```

Download



```
<Figure size 640x480 with 1 Axes>
```

```
#c.
open=cake_trades.iloc[0]['price']
open
```

```
38.76
```

```
close=cake_trades.iloc[-1]['price']
close
```

```
38.74
```

```
high=max(cake_trades['price'])
high
```

```
39.09
```

```
low=min(cake_trades['price'])
low
```

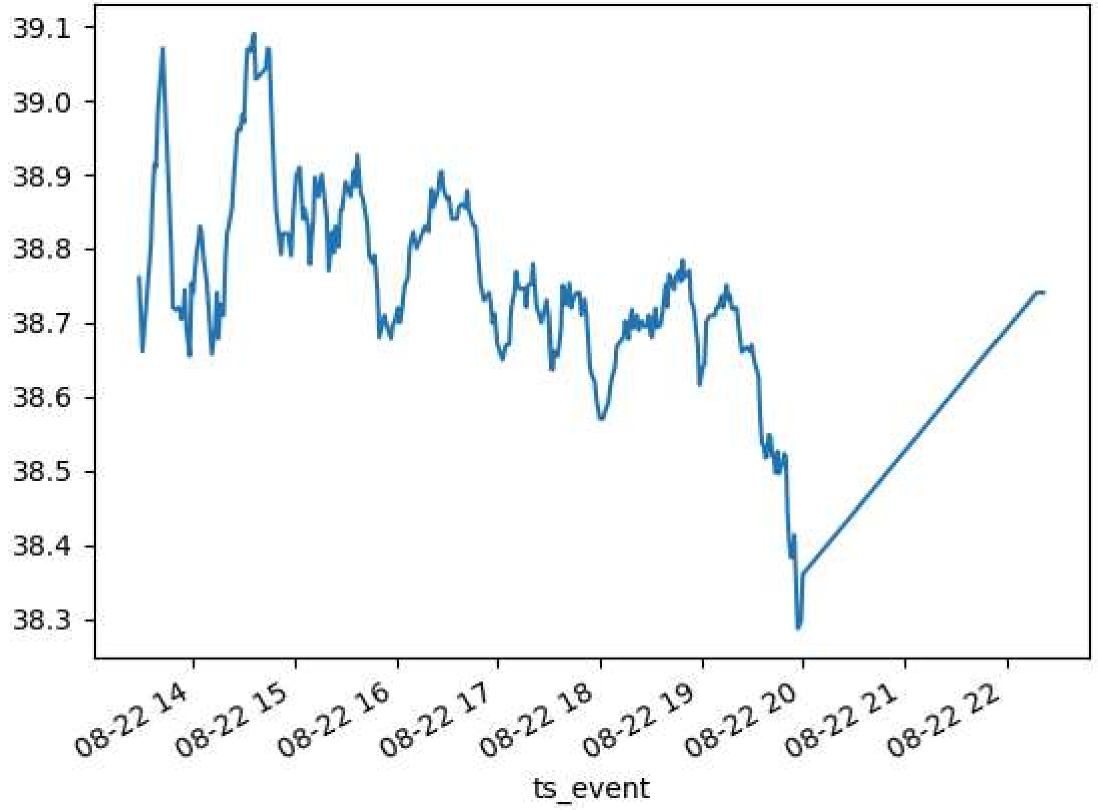
```
38.265
```

```
#d.
cake_vwap=cake_trades.groupby(cake_trades['ts_event'].dt.floor('min')).a
```

```
cake_vwap.plot()
```

```
<Axes: xlabel='ts_event'>
```

Download



```
cake_vwap.describe()
```

```
count    297.000000
mean      38.746883
std        0.136048
min       38.286692
25%       38.690647
50%       38.737566
75%       38.830000
max       39.090000
dtype: float64
```

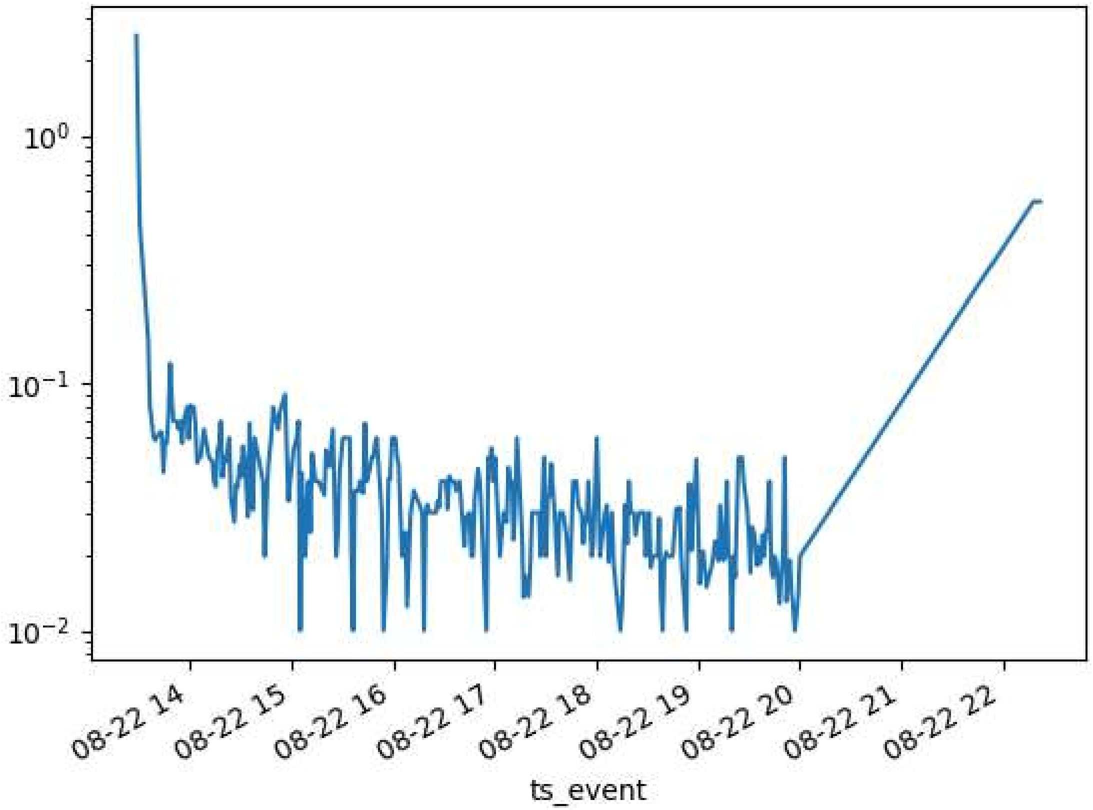
```
#e.
cake_trades['spread']=cake_trades['ask_px_00']-cake_trades['bid_px_00']
cake_trades['avl']=0
for index,row in cake_trades.iterrows():
    if(row['side']=='A'):
        row['avl']=row['bid_sz_00']
    elif(row['side']=='B'):
        row['avl']=row['ask_sz_00']
    else:
        row['avl']=row['bid_sz_00']+row['ask_sz_00']
```

```
cake_spread_min=cake_trades.groupby(cake_trades['ts_event'].dt.floor('mi
```

```
cake_spread_min.plot(logy=True)
```

```
<Axes: xlabel='ts_event'>
```

Download



```
<Figure size 640x480 with 1 Axes>
```

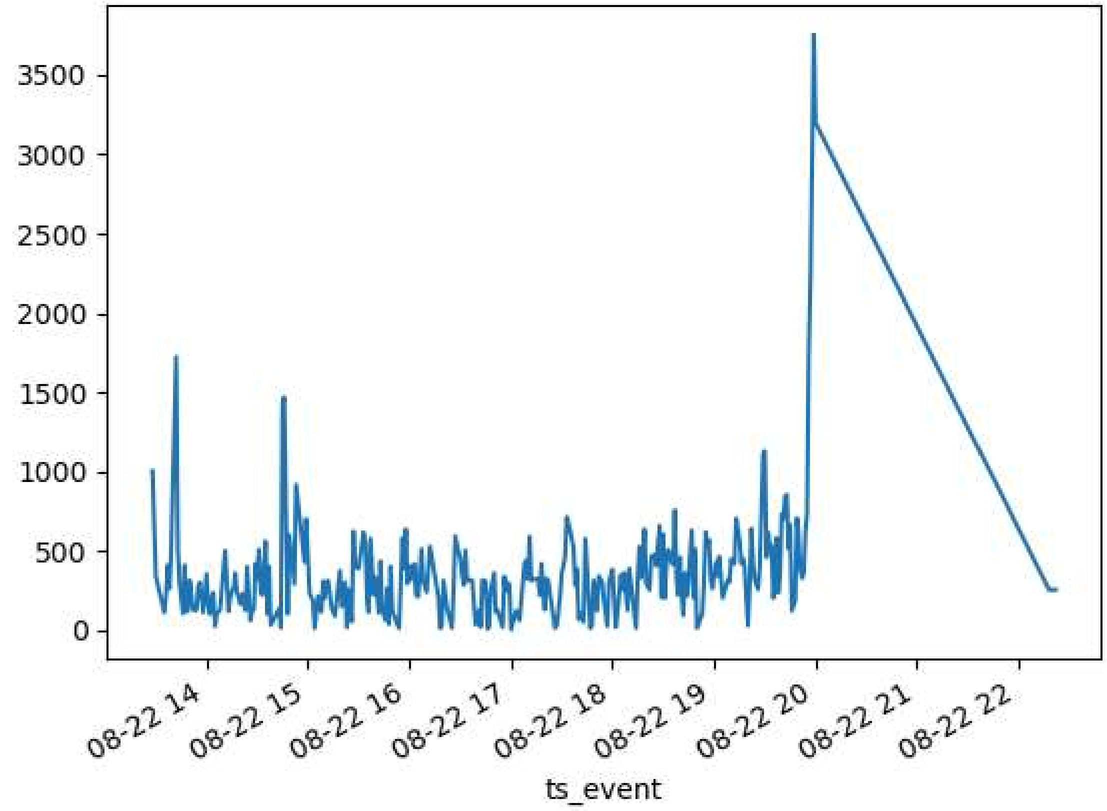
```
cake_trades.loc[np.where(cake_trades['side']=='A')[0],'avl']=cake_trades
cake_trades.loc[np.where(cake_trades['side']=='B')[0],'avl']=cake_trades
cake_trades.loc[np.where(cake_trades['side']=='N')[0],'avl']=cake_trades
```

```
cake_depth_min=cake_trades.groupby(cake_trades['ts_event'].dt.floor('min
```

```
cake_depth_min.plot()
```

```
<Axes: xlabel='ts_event'>
```

Download



```
<Figure size 640x480 with 1 Axes>
```

```
test_depth=2*cake_spread_min.mean()
```

```
#h.
cake_trades['mid']=(cake_trades['ask_px_00']+cake_trades['bid_px_00'])/2
```

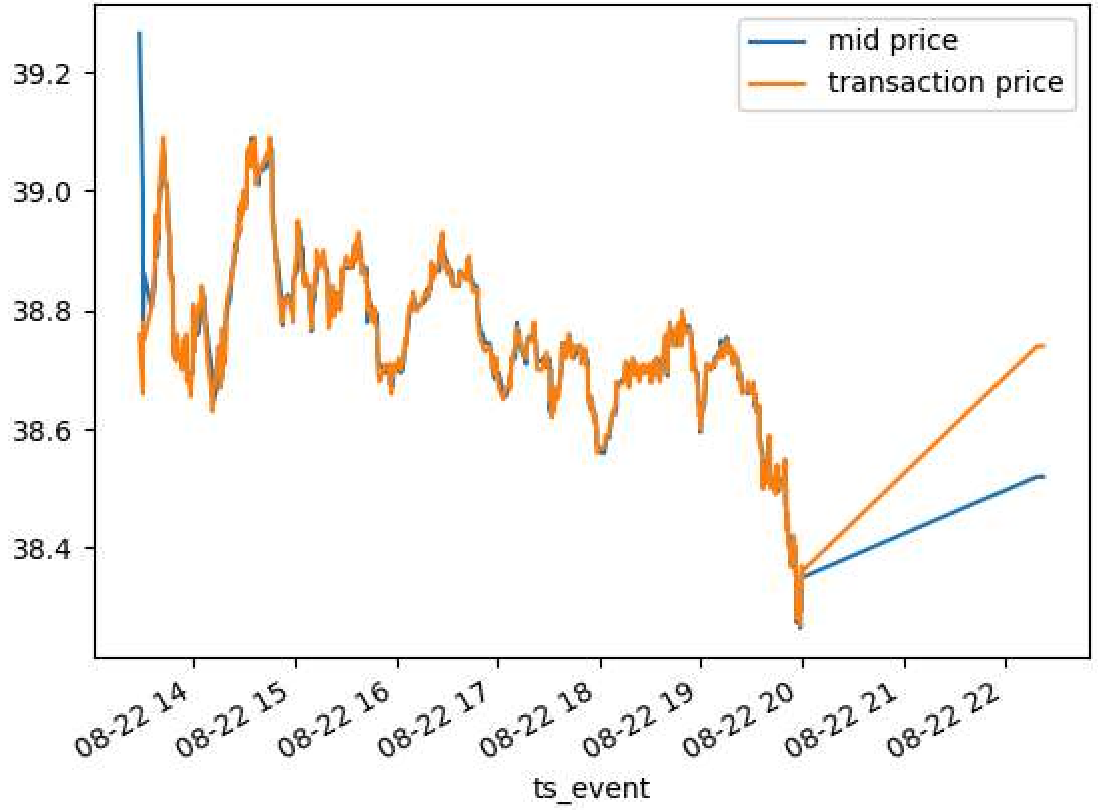
```
cake_mid_1s=cake_trades.groupby(cake_trades['ts_event'].dt.floor('s'))['
cake_tsp_1s=cake_trades.groupby(cake_trades['ts_event'].dt.floor('s'))['
```

```
cake_1s=pd.DataFrame({'mid price':cake_mid_1s,'transaction price':cake_t
cake_1s.plot()
cake_1s.describe()
```

	mid price	transaction price
count	749.000000	749.000000
mean	38.696509	38.696302
std	0.180203	0.178756
min	38.265000	38.270000
25%	38.630000	38.630000
50%	38.720000	38.715000
75%	38.810000	38.810000
max	39.265000	39.090000

```
        mid price  transaction price
count  749.000000         749.000000
mean    38.696509          38.696302
std      0.180203           0.178756
min     38.265000          38.270000
25%     38.630000          38.630000
50%     38.720000          38.715000
75%     38.810000          38.810000
max     39.265000          39.090000
```

Download

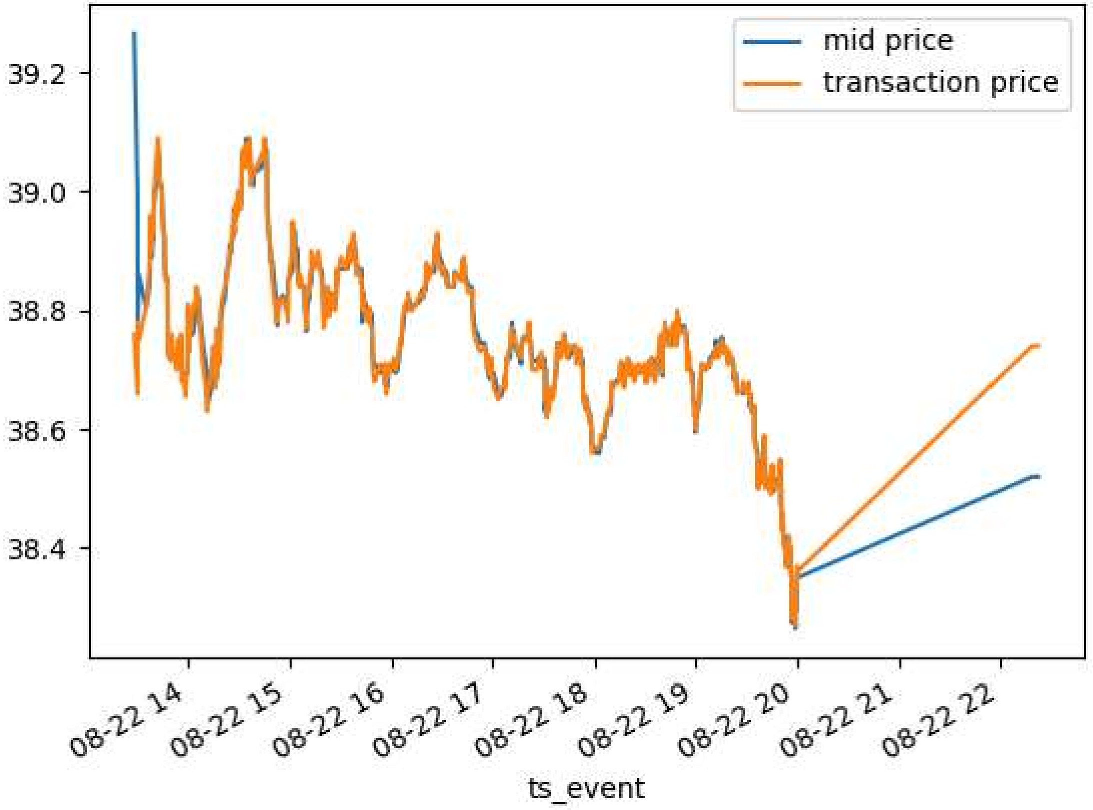


```
cake_mid_1m=cake_trades.groupby(cake_trades['ts_event'].dt.floor('min'))
cake_tsp_1m=cake_trades.groupby(cake_trades['ts_event'].dt.floor('min'))
cake_1m=pd.DataFrame({'mid price':cake_mid_1s,'transaction price':cake_t
cake_1m.plot()
cake_1m.describe()
```

	mid price	transaction price
count	749.000000	749.000000
mean	38.696509	38.696302
std	0.180203	0.178756
min	38.265000	38.270000
25%	38.630000	38.630000
50%	38.720000	38.715000
75%	38.810000	38.810000
max	39.265000	39.090000

```
        mid price  transaction price
count  749.000000         749.000000
mean    38.696509          38.696302
std      0.180203           0.178756
min     38.265000          38.270000
25%     38.630000          38.630000
50%     38.720000          38.715000
75%     38.810000          38.810000
max     39.265000          39.090000
```

Download



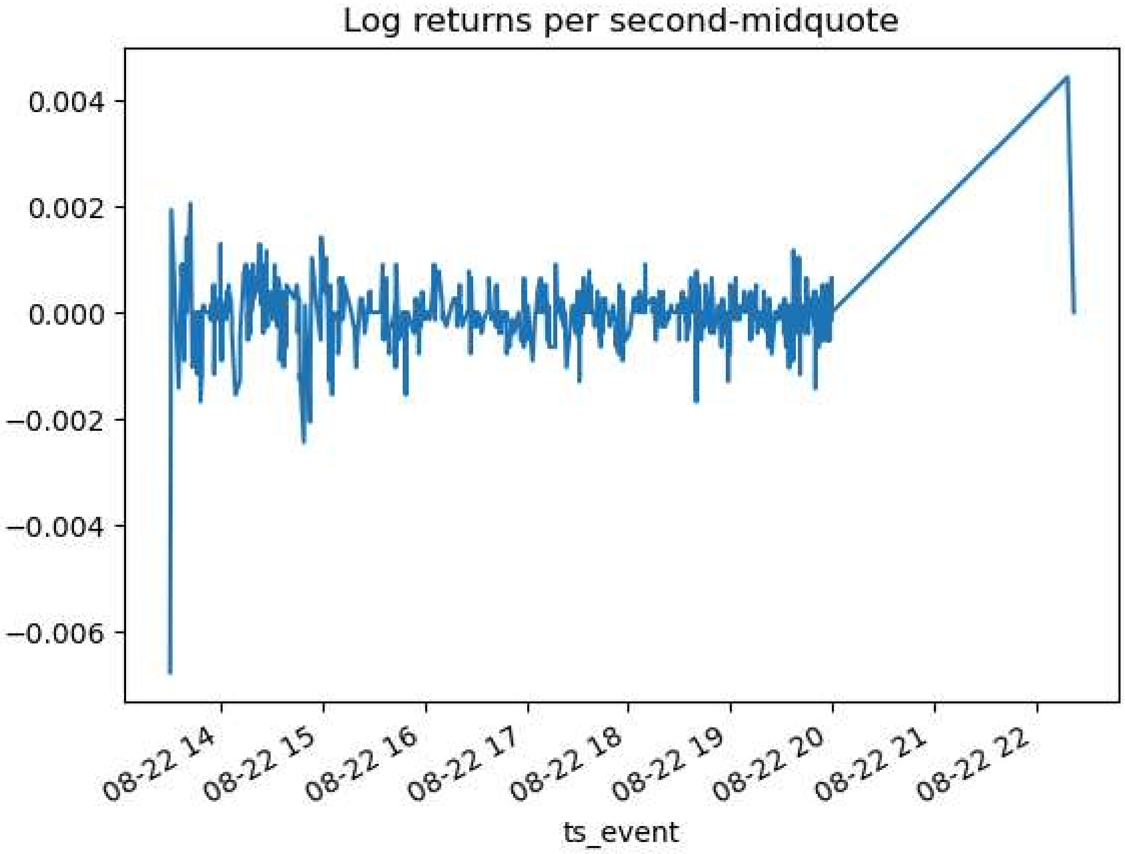
```
<Figure size 640x480 with 1 Axes>
```

```
#i.
cake_logret_mpq_1s=np.log(cake_mid_1s).diff()
cake_logret_mpq_1s.plot(title="Log returns per second-midquote")
cake_logret_mpq_1s.describe()
```

```
count    748.000000
mean      -0.000026
std        0.000574
```

```
min      -0.006772
25%      -0.000257
50%       0.000000
75%       0.000256
max       0.004423
Name: mid, dtype: float64
```

Download



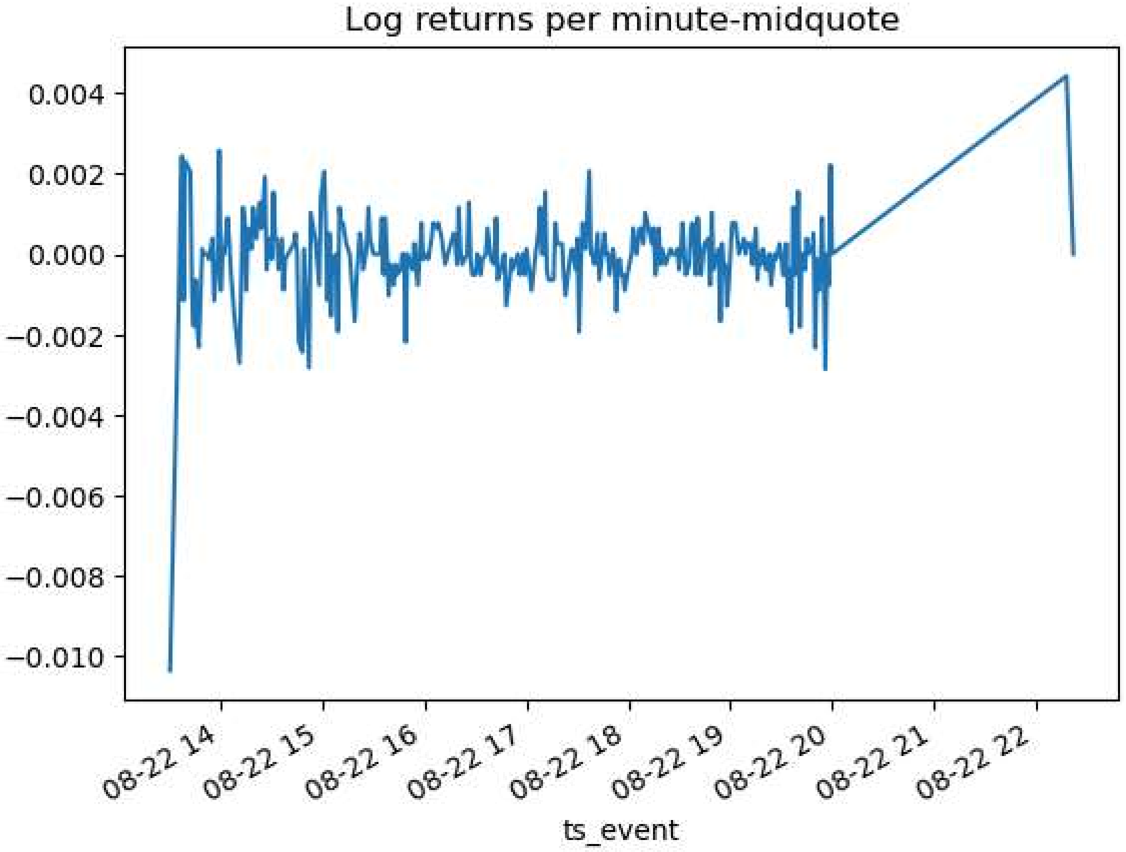
```
<Figure size 640x480 with 1 Axes>
```

```
cake_logret_mpq_1m=np.log(cake_mid_1m).diff()
cake_logret_mpq_1m.plot(title="Log returns per minute-midquote")
cake_logret_mpq_1m.describe()
```

```
count    296.000000
mean      -0.000065
std        0.001068
min       -0.010368
```

```
25%       -0.000389
50%        0.000000
75%        0.000386
max        0.004423
Name: mid, dtype: float64
```

Download



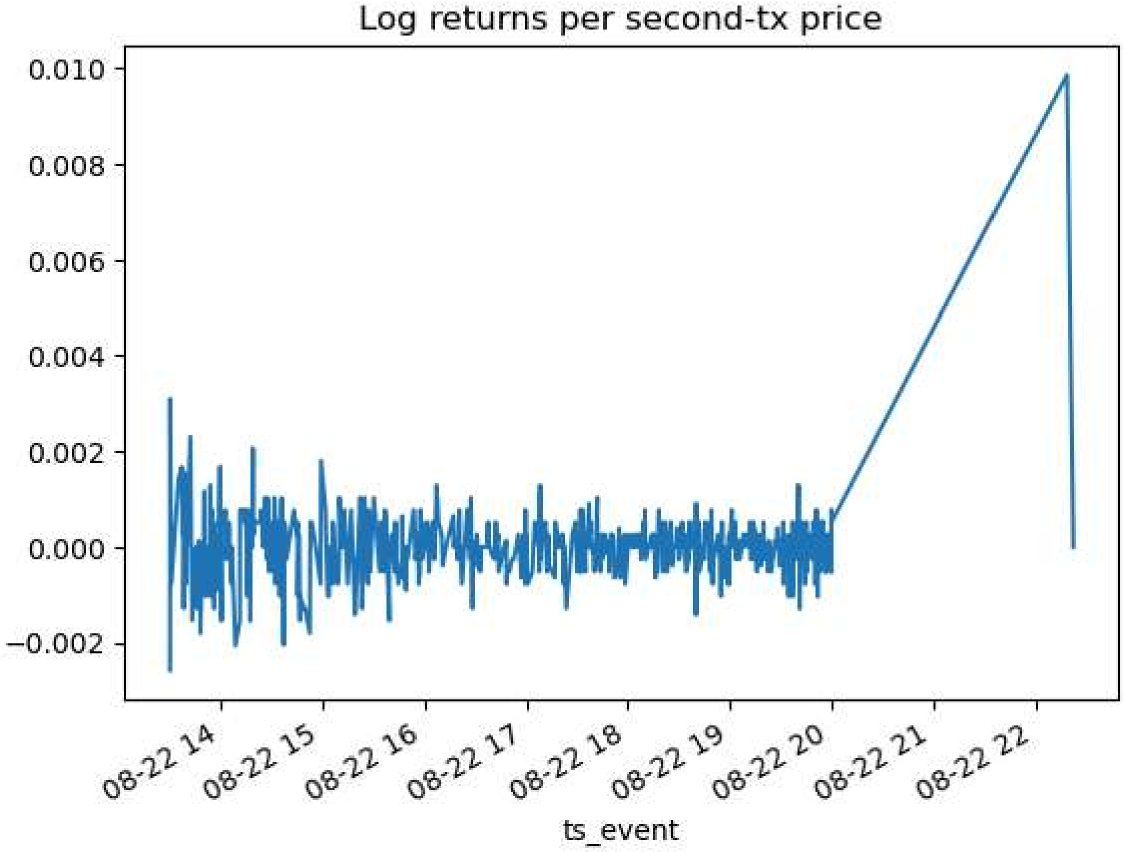
```
<Figure size 640x480 with 1 Axes>
```

```
cake_logret_tsp_1s=np.log(cake_tsp_1s).diff()
cake_logret_tsp_1s.plot(title="Log returns per second-tx price")
cake_logret_tsp_1s.describe()
```

```
count    7.480000e+02
mean    -6.900121e-07
std      6.591143e-04
min     -2.583313e-03
25%     -2.587155e-04
50%      0.000000e+00
```

```
75%      2.585148e-04
max      9.857408e-03
Name: price, dtype: float64
```

Download



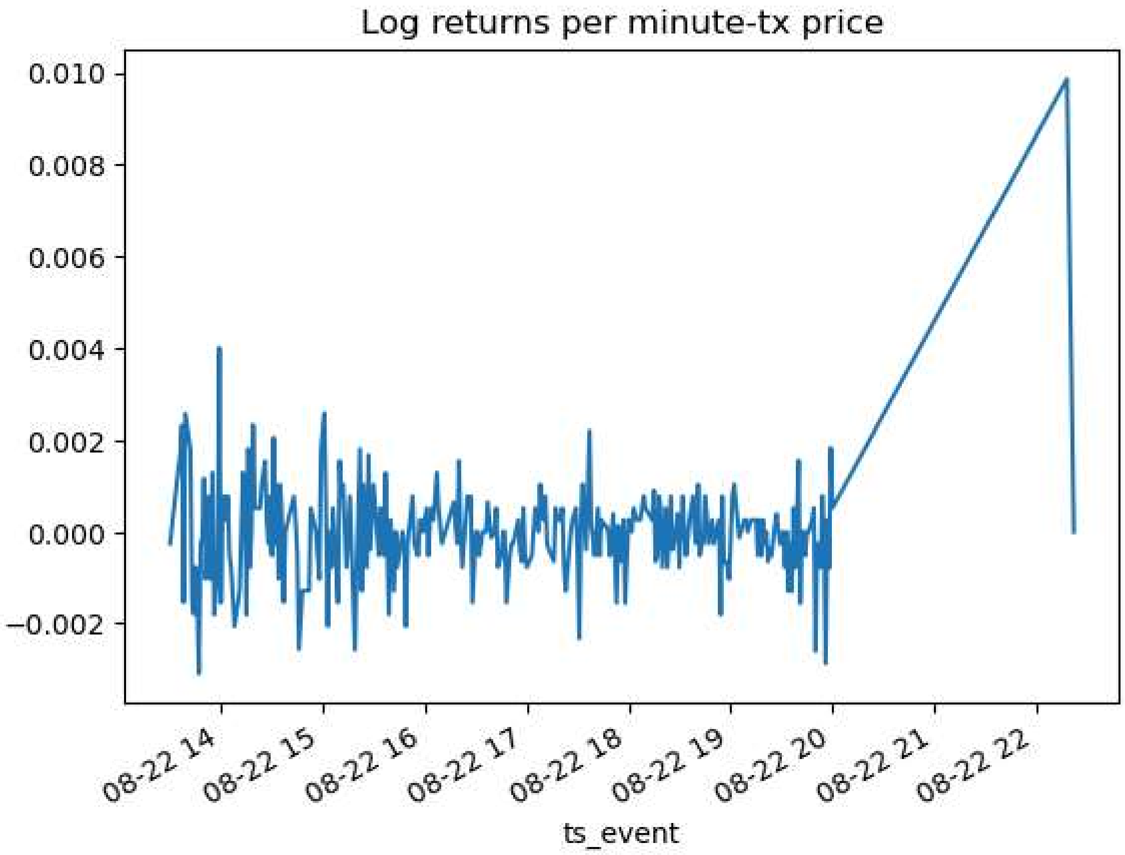
```
<Figure size 640x480 with 1 Axes>
```

```
cake_logret_tsp_1m=np.log(cake_tsp_1m).diff()
cake_logret_tsp_1m.plot(title="Log returns per minute-tx price")
cake_logret_tsp_1m.describe()
```

```
count    296.000000
mean      -0.000002
std        0.001116
min       -0.003094
25%       -0.000517
50%        0.000000
75%        0.000516
```

```
max        0.009857
Name: price, dtype: float64
```

⤓ Download



```
<Figure size 640x480 with 1 Axes>
```

```
#j.
#realized variance calculation
rv_mpq_1s=cake_logret_mpq_1s.var()
rv_mpq_1m=cake_logret_mpq_1m.var()
rv_ts_1s=cake_logret_tsp_1s.var()
rv_ts_1m=cake_logret_tsp_1m.var()
```

```
rv_mpq_1s
```

```
3.292514787501337e-07
```

```
rv_mpq_1m
```

```
1.1396518675140203e-06
```

```
rv_ts_1s
```

```
4.3443172611155505e-07
```

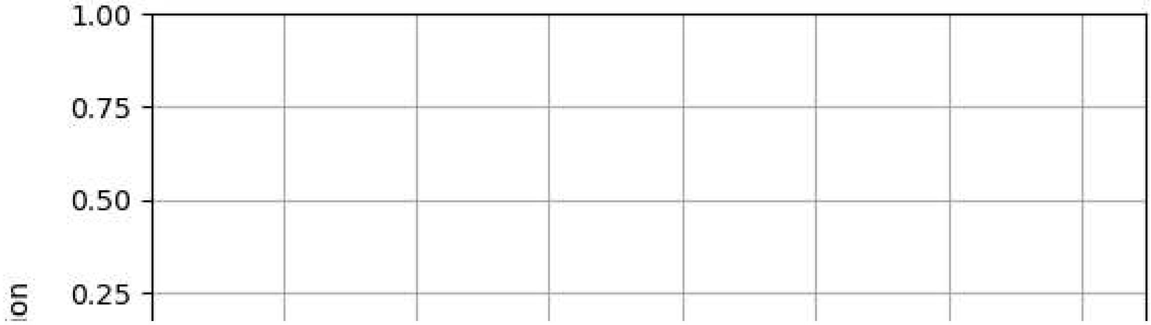
```
rv_ts_1m
```

```
1.2464683371154937e-06
```

```
#k.
x=pd.plotting.autocorrelation_plot(cake_logret_mpq_1s[1:])
x.plot(title='Midquote 1s acf plot')
```

```
[]
```

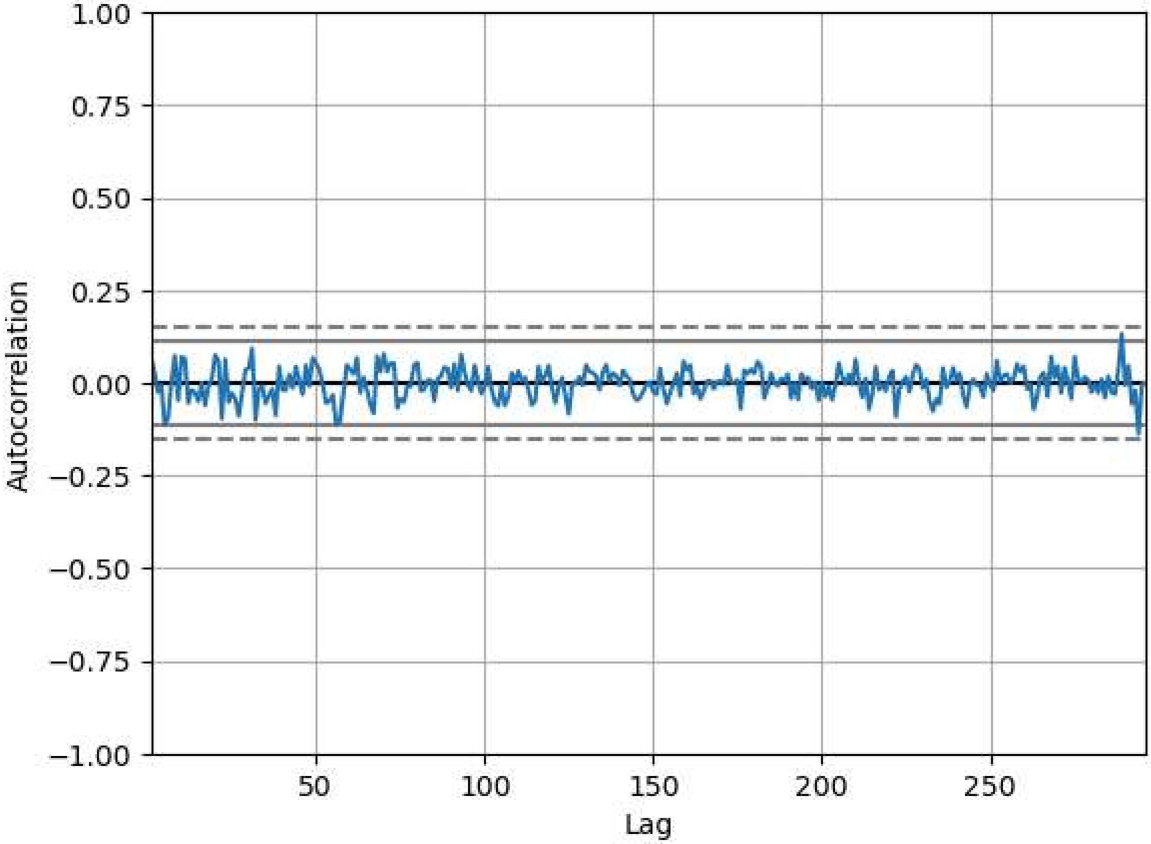
Download



```
x=pd.plotting.autocorrelation_plot(cake_logret_mpq_1m[1:])
x.plot(title='Midquote 1m acf plot')
```

```
[]
```

Download

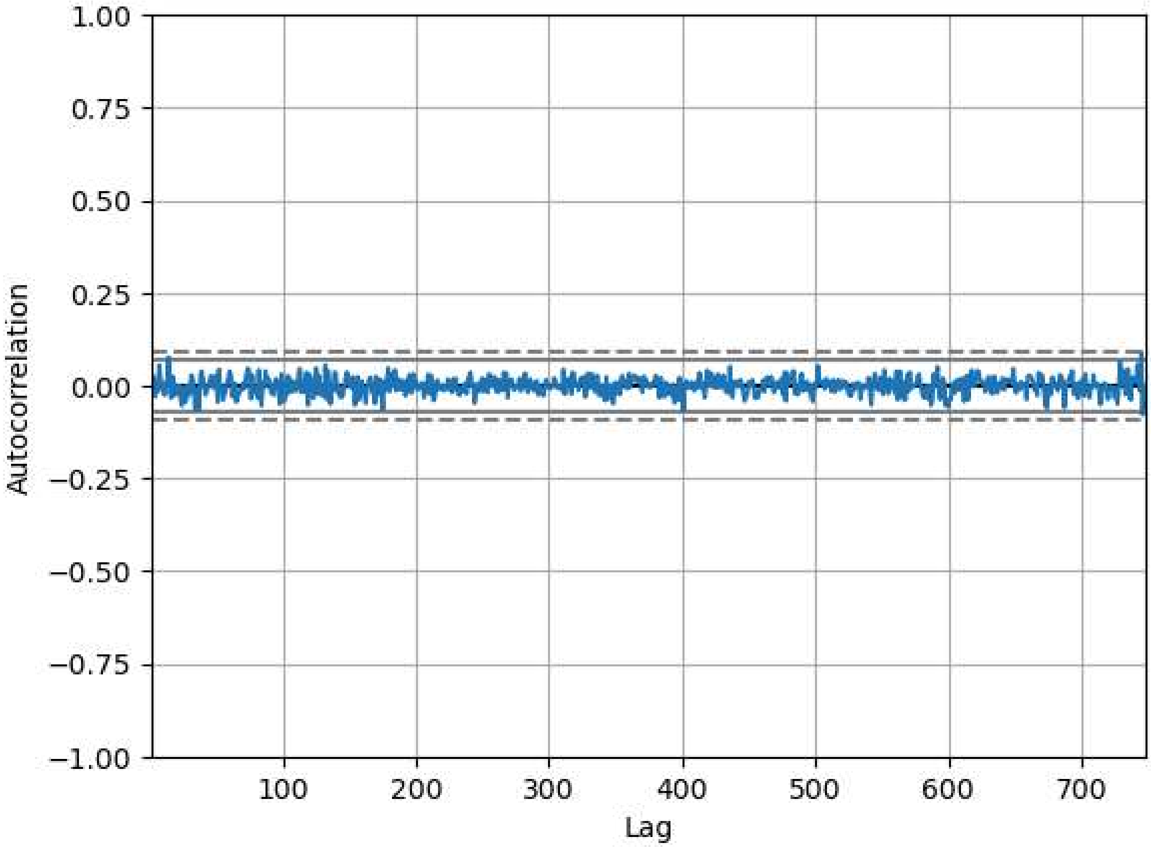


```
<Figure size 640x480 with 1 Axes>
```

```
x=pd.plotting.autocorrelation_plot(cake_logret_tsp_1s[1:])
x.plot(title='Transaction price 1s acf plot')
```

```
[]
```

Download

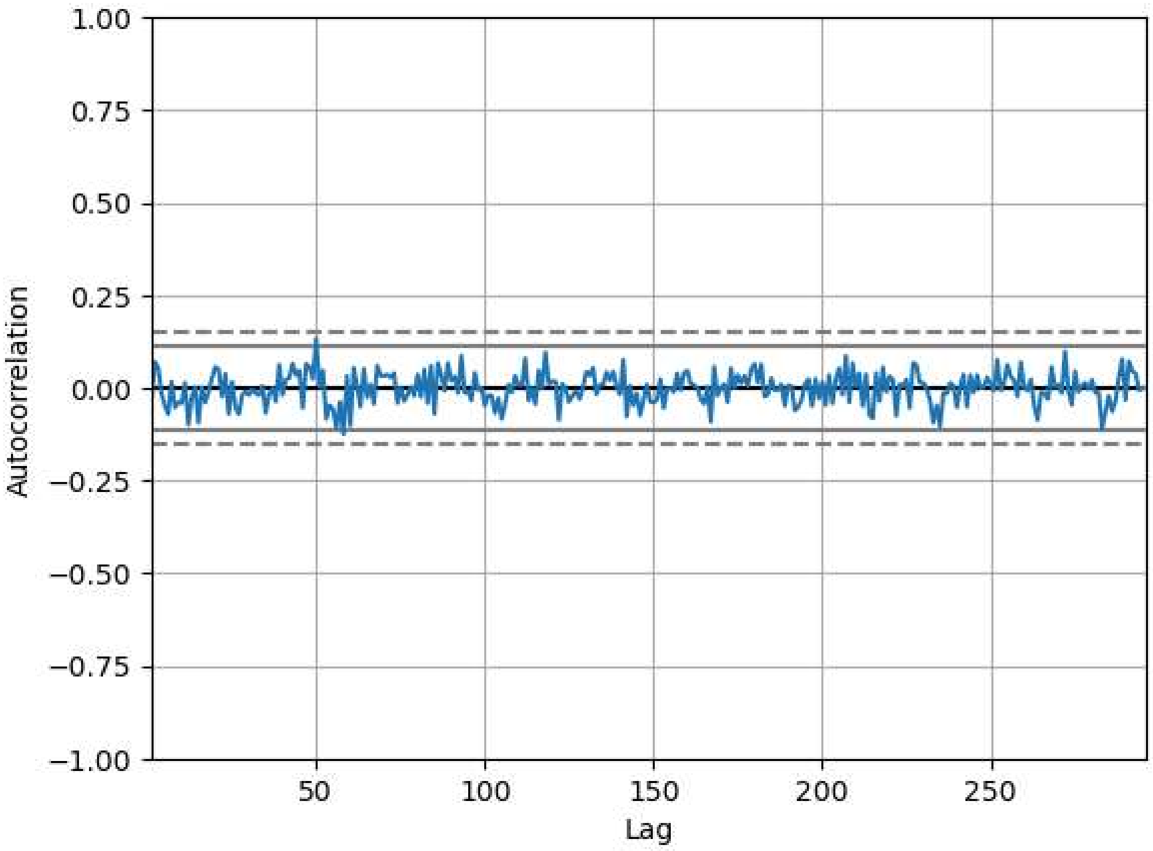


```
<Figure size 640x480 with 1 Axes>
```

```
x=pd.plotting.autocorrelation_plot(cake_logret_tsp_1m[1:])
x.plot(title='Transaction price 1m acf plot')
```

```
[]
```

Download


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Autocorrelation
50
100
150
200
250
Lag

```
<Figure size 640x480 with 1 Axes>
```