Task 01

In [ ]:

```
# Importing data into dataframe using pandas

import pandas as pd

df = pd.read_csv("data/WaterAtlas-OneLake.csv")
```

In [ ]:

```
df.head()
```

Out[ ]:

	WBodyID	WaterBodyName	DataSource	StationID	StationName	Actual_StationID	Actual_Latitude	Actual_Longitude	DEP_WBID	SampleDate	...	DepthUnits	Parameter	Characteristic	Sample_Fraction	Resul
0	2003889	Okaloacoochee Branch	WIN_21FLSFWM	32275	CRFW09	32275	26.7629	-81.4001	3235O	5/18/2020 11:11:00 AM	...	m	TN_ugl	Nitrogen	Total	1:
1	2003889	Okaloacoochee Branch	WIN_21FLSFWM	32275	CRFW09	32275	26.7629	-81.4001	3235O	5/18/2020 11:11:00 AM	...	m	NH3_N_ugl	Nitrogen, ammonia as N	Dissolved	:
2	2003889	Okaloacoochee Branch	WIN_21FLSFWM	32275	CRFW09	32275	26.7629	-81.4001	3235O	5/18/2020 11:11:00 AM	...	m	NOx_ugl	Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Dissolved	
3	2003889	Okaloacoochee Branch	WIN_21FLSFWM	32275	CRFW09	32275	26.7629	-81.4001	3235O	5/18/2020 11:11:00 AM	...	m	TP_ugl	Phosphorus as P	Total	
4	2003889	Okaloacoochee Branch	WIN_21FLSFWM	32275	CRFW09	32275	26.7629	-81.4001	3235O	5/18/2020 11:11:00 AM	...	m	OP_mgl	Phosphorus, phosphate (PO4) as P	Dissolved	

5 rows × 21 columns

In [ ]:

```
# Data has 21 columns and the column names are printed down

len(df.columns), df.columns
```

Out[ ]:

```
(21,
 Index(['WBodyID', 'WaterBodyName', 'DataSource', 'StationID', 'StationName',
        'Actual_StationID', 'Actual_Latitude', 'Actual_Longitude', 'DEP_WBID',
        'SampleDate', 'ActivityDepth', 'DepthUnits', 'Parameter',
        'Characteristic', 'Sample_Fraction', 'Result_Value', 'Result_Unit',
        'QACode', 'Result_Comment', 'Original_Result_Value',
        'Original_Result_Unit'],
       dtype='object'))
```

In [ ]:

```
df.describe()
```

Out[ ]:

	WBodyID	Actual_Latitude	Actual_Longitude	ActivityDepth	Result_Value	Original_Result_Value
count	2289.0	2289.000000	2289.000000	2289.000000	2289.000000	2179.000000
mean	2003889.0	26.759472	-81.399168	0.331355	203.284873	72.951471
std	0.0	0.003488	0.000987	0.161500	733.434420	177.903803
min	2003889.0	26.751830	-81.400100	0.100000	0.002000	0.002000
25%	2003889.0	26.756020	-81.400000	0.152439	3.570000	0.255000
50%	2003889.0	26.762710	-81.400000	0.300000	21.000000	6.000000
75%	2003889.0	26.762778	-81.398360	0.500000	89.600000	30.250000
max	2003889.0	26.762900	-81.394680	0.500000	10600.000000	2240.000000

In [ ]:

```
# Listing the number of 'NaN' values for all the columns present in the dataframe.

for i in df.columns:
    count_nan = df[i].isnull().sum()
    print(i,count_nan)
```

```
WBodyID 0
WaterBodyName 0
DataSource 0
StationID 0
StationName 0
Actual_StationID 0
Actual_Latitude 0
Actual_Longitude 0
DEP_WBID 0
SampleDate 0
ActivityDepth 0
DepthUnits 0
Parameter 0
Characteristic 0
Sample_Fraction 554
Result_Value 0
Result_Unit 0
QACode 1961
Result_Comment 1984
Original_Result_Value 110
Original_Result_Unit 110
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2289 entries, 0 to 2288
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   WBodyID                2289 non-null   int64  
 1   WaterBodyName          2289 non-null   object 
 2   DataSource             2289 non-null   object 
 3   StationID              2289 non-null   object 
 4   StationName            2289 non-null   object 
 5   Actual_StationID       2289 non-null   object 
 6   Actual_Latitude        2289 non-null   float64
 7   Actual_Longitude       2289 non-null   float64
 8   DEP_WBID               2289 non-null   object 
 9   SampleDate             2289 non-null   object 
 10  ActivityDepth          2289 non-null   float64
 11  DepthUnits             2289 non-null   object 
 12  Parameter              2289 non-null   object 
 13  Characteristic         2289 non-null   object 
 14  Sample_Fraction        1735 non-null   object 
 15  Result_Value           2289 non-null   float64
 16  Result_Unit            2289 non-null   object 
 17  QACode                 328 non-null    object 
 18  Result_Comment         305 non-null    object 
 19  Original_Result_Value  2179 non-null   float64
 20  Original_Result_Unit   2179 non-null   object 
dtypes: float64(5), int64(1), object(15)
memory usage: 375.7+ KB
```

Task 02

In [ ]:

```
# Extracting all the dates in the variable date, and obtaining a set out of it of the unique date values.
```

```
date = df.SampleDate.values.tolist()
set_date = list(set(date))
```

In [ ]:

```
# Extracting parameter and their characterstic values from dataframe

parameter = df.Parameter.values.tolist()
characteristic = df.Characteristic.values.tolist()
```

In [ ]:

```
# there are a total of 72 different parameters and 61 charactersctic
# Hence, there are 11 paramters with no descriptions or multiple parameters with similar description

len(set(parameter)), len(set(characteristic))
```

Out[ ]:

```
(72, 61)
```

In [ ]:

```
# As the parameter's name itself is not very informative, adding characterstic description adds to it.

pairvalue = [i+" "+j for i,j in zip(parameter, characteristic)]
len(set(pairvalue))
```

Out[ ]:

```
72
```

In [ ]:

```
# All the unique set pairvalues (72)

list_pair = list(set(pairvalue))
```

In [ ]:

```
df['Parameter'] = pairvalue
```

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['WBodyID', 'WaterBodyName', 'DataSource', 'StationID', 'StationName',
       'Actual_StationID', 'Actual_Latitude', 'Actual_Longitude', 'DEP_WBID',
       'SampleDate', 'ActivityDepth', 'DepthUnits', 'Parameter',
       'Characteristic', 'Sample_Fraction', 'Result_Value', 'Result_Unit',
       'QACode', 'Result_Comment', 'Original_Result_Value',
       'Original_Result_Unit'],
      dtype='object')
```

In [ ]:

```
#dropping the columns with similar values or the ones with repeatative / sparse data in them.

df = df.drop(columns=['WBodyID', 'WaterBodyName', 'DataSource', 'StationID', 'StationName',
        'Actual_StationID', 'Actual_Latitude', 'Actual_Longitude', 'DEP_WBID', 'DepthUnits',
        'Characteristic', 'Sample_Fraction','QACode', 'Result_Comment', 'Original_Result_Value',
        'Original_Result_Unit'])
```

In [ ]:

```
df.head()
```

Out[ ]:

	SampleDate	ActivityDepth	Parameter	Result_Value	Result_Unit
0	5/18/2020 11:11:00 AM	0.23	TN_ugl Nitrogen	1280.000	ug/l
1	5/18/2020 11:11:00 AM	0.23	NH3_N_ugl Nitrogen, ammonia as N	203.000	ug/l
2	5/18/2020 11:11:00 AM	0.23	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3...	9.000	ug/l
3	5/18/2020 11:11:00 AM	0.23	TP_ugl Phosphorus as P	52.000	ug/l
4	5/18/2020 11:11:00 AM	0.23	OP_mgl Phosphorus, phosphate (PO4) as P	0.002	mg/l

In [ ]:

```
# Collecting data of all the parameters date wise in the variable final_list

final_list=list()
for d in set_date:
    top = {key:() for key in list_pair+['ActivityDepth']}
    for i,j in df.iterrows():
        if(j[0]==d):
            #print(j[1],j[2],j[3])
            top[j[2]]=(j[3],j[4])
            #print(top)
            #raise KeyboardInterrupt
            if(len(top['ActivityDepth'])==0):
                top['ActivityDepth'] = [j[1]]
    final_list.append(top)
```

In [ ]:

```
# creating varibale fdf with 103 rows, each for one unique date value, containing values for 73 different parameters

fdf = pd.DataFrame(data=None, columns=["Date"])
```

In [ ]:

```
fdf['Date'] = set_date
```

In [ ]:

```
# Adding data to the fdf data frame

for i in list_pair + ['ActivityDepth']:
    top = list()
    for j in final_list:
        top.append(j[i])
    fdf[i] = top
```

In [ ]:

```
fdf.head()
```

Out[ ]:

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	Na_mgl Sodium	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	2 Dichloroph
0	8/24/2020 11:36:00 AM	()	()	()	()	(59.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(48.0, ug/l)	
1	5/18/2020 11:11:00 AM	()	()	()	()	(203.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(9.0, ug/l)	
2	04-05-2021 11:15	()	()	()	()	(303.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(15.0, ug/l)	
3	03-12-1979 0:00	()	()	()	()	()	()	()	()	()	...	()	()	()	()	()	()	()	
4	11/14/2017 1:50:00 PM	(0.47, ug/l)	()	(62.0, mg/l)	()	(150.0, ug/l)	()	(0.01, ug/l)	()	()	...	(44.1, mg/l)	()	(0.02, ug/l)	()	()	(0.29, ug/l)	(200.0, ug/l)	

5 rows × 74 columns

In [ ]:

```
# copying fdf to fdf_values as python creates pointers

fdf_values = fdf.copy()
```

In [ ]:

```
fdf_values
```

Out[ ]:

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	Na_mgl Sodium	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Dichlor
0	8/24/2020 11:36:00 AM	()	()	()	()	(59.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(48.0, ug/l)	
1	5/18/2020 11:11:00 AM	()	()	()	()	(203.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(9.0, ug/l)	
2	04-05-2021 11:15	()	()	()	()	(303.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(15.0, ug/l)	
3	03-12-1979 0:00	()	()	()	()	()	()	()	()	()	...	()	()	()	()	()	()	()	
4	11/14/2017 1:50:00 PM	(0.47, ug/l)	()	(62.0, mg/l)	()	(150.0, ug/l)	()	(0.01, ug/l)	()	()	...	(44.1, mg/l)	()	(0.02, ug/l)	()	()	(0.29, ug/l)	(200.0, ug/l)	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
98	9/16/1980 12:00:00 AM	()	()	()	()	()	()	()	()	()	...	()	(4.0, ug/l)	()	()	()	()	(10.0, ug/l)	
99	4/24/2017 10:50:00 AM	(1.78, ug/l)	()	(49.0, mg/l)	()	(120.0, ug/l)	()	(0.01, ug/l)	()	()	...	(42.5, mg/l)	()	(0.02, ug/l)	()	()	(0.32, ug/l)	(40.0, ug/l)	
100	5/19/1980 12:00:00 AM	()	()	()	()	()	()	()	()	()	...	()	()	()	()	()	()	()	
101	8/28/2017 11:00:00 AM	(0.56, ug/l)	()	(70.0, mg/l)	()	(200.0, ug/l)	()	(0.01, ug/l)	()	()	...	(51.4, mg/l)	()	(0.02, ug/l)	()	()	(0.2, ug/l)	(47.0, ug/l)	
102	10/19/2020 12:16:00 PM	()	()	()	()	(40.0, ug/l)	()	()	()	()	...	()	()	()	()	()	()	(107.0, ug/l)	

103 rows × 74 columns

In [ ]:

```
# value_extract function extracts the num value from the tuples present in the fdf variable
# as the unit ug/l and mg/l are used in different and similar columns at the same time
# this function converts all the values into ug/l.

def value_extract(top):
    unit = list()
    #print(top)
    #raise KeyboardInterrupt
    for i in top:
        try:
            unit.append(i[1])
        except:
            unit.append("")
    check = list(set(unit))
    #print(check)
    #raise KeyboardInterrupt
    if('ug/l' in check and 'mg/l' in check):
        ntop = list()
        for i in top:
            try:
                if(i[1].count("mg/l")>0):
                    ntop.append(i[0]*1000)
                else:
                    ntop.append(i[0])
            except:
                ntop.append("na")
    elif("mg/l" in check):
        ntop = list()
        for i in top:
            try:
                ntop.append(i[0]*1000)
            except:
                ntop.append("na")
    else:
        return False, list()

    return True, ntop

# updating fdf_values dataframe with just num values in the column.

for i in list_pair  + ['ActivityDepth']:
    #print(fdf_values[i])
    flag, temp = value_extract(fdf_values[i])
    #print(flag, temp)
    #raise KeyboardInterrupt
    if(flag):
        fdf_values[i] = temp
    else:
        temp = list()
        for j in fdf_values[i]:
            try:
                temp.append(j[0])
            except:
                temp.append("na")
        #print(temp)
        #raise KeyboardInterrupt
        fdf_values[i] = temp
        #print(fdf_values)
        #raise KeyboardInterrupt
```

In [ ]:

```
list_pair[0]
```

Out[ ]:

```
'Ni_ugl Nickel'
```

In [ ]:

```
fdf_values
```

Out[ ]:

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	Na_mgl Sodium	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Dichlor

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	Na_mgl Sodium	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Dichlor
0	8/24/2020 11:36:00 AM	na	na	na	na	59.0	na	na	na	na	...	na	na	na	na	na	na	48.0	
1	5/18/2020 11:11:00 AM	na	na	na	na	203.0	na	na	na	na	...	na	na	na	na	na	na	9.0	
2	04-05-2021 11:15	na	na	na	na	303.0	na	na	na	na	...	na	na	na	na	na	na	15.0	
3	03-12-1979 0:00	na	na	na	na	na	na	na	na	na	...	na	na	na	na	na	na	na	
4	11/14/2017 1:50:00 PM	0.47	na	62000.0	na	150.0	na	0.01	na	na	...	44100.0	na	0.02	na	na	0.29	200.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
98	9/16/1980 12:00:00 AM	na	na	na	na	na	na	na	na	na	...	na	4.0	na	na	na	na	10.0	
99	4/24/2017 10:50:00 AM	1.78	na	49000.0	na	120.0	na	0.01	na	na	...	42500.0	na	0.02	na	na	0.32	40.0	
100	5/19/1980 12:00:00 AM	na	na	na	na	na	na	na	na	na	...	na	na	na	na	na	na	na	
101	8/28/2017 11:00:00 AM	0.56	na	70000.0	na	200.0	na	0.01	na	na	...	51400.0	na	0.02	na	na	0.2	47.0	
102	10/19/2020 12:16:00 PM	na	na	na	na	40.0	na	na	na	na	...	na	na	na	na	na	na	107.0	

103 rows × 74 columns

In [ ]:

```
!pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.4.3-cp37-cp37m-macosx_10_9_x86_64.whl (7.2 MB)
     |████████████████████████████████| 7.2 MB 6.1 MB/s
Requirement already satisfied: numpy>=1.16 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib) (1.21.0)
Collecting pillow>=6.2.0
  Downloading Pillow-8.3.2-cp37-cp37m-macosx_10_10_x86_64.whl (3.0 MB)
     |████████████████████████████████| 3.0 MB 13.5 MB/s
Requirement already satisfied: python-dateutil>=2.7 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib) (2.4.7)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.2-cp37-cp37m-macosx_10_9_x86_64.whl (61 kB)
     |████████████████████████████████| 61 kB 74 kB/s
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: six in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Installing collected packages: pillow, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.2 matplotlib-3.4.3 pillow-8.3.2
```

In [ ]:

```
import datetime
date_str = '09/12/2017' # The date - 29 Dec 2017
format_str = '%m/%d/%Y' # The format
datetime_obj = datetime.datetime.strptime(date_str, format_str)
print(datetime_obj.date())
```

```
2017-09-12
```

In [ ]:

```
# converting date values to date format to sort the columns based on the date.

import datetime
temp = fdf_values.Date.values.tolist()
temp = [i.split(" ")[0].replace("-","/") for i in temp]
```

In [ ]:

```
format_str = '%m/%d/%Y'
temp = [datetime.datetime.strptime(str(i), format_str) for i in temp]
```

In [ ]:

```
fdf_values['nDate'] = temp
fdf_values = fdf_values.sort_values(by=['nDate'])
```

In [ ]:

```
fdf_values.head()
```

Out[ ]:

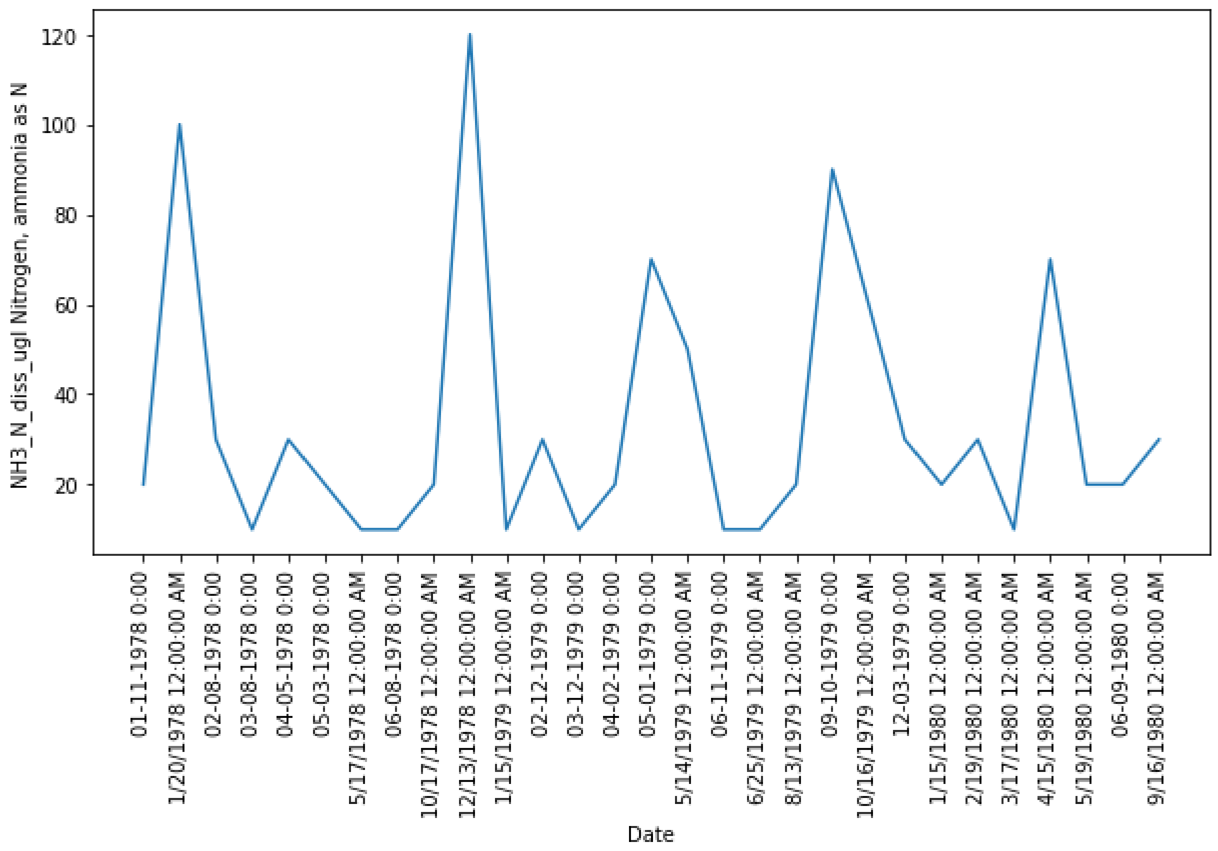
	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	24D_ugl 2 Dichlorophenoxyace acid (2,4
71	01-11-1978 0:00	na	na	na	na	na	16.0	na	na	na	...	na	na	na	na	na	24.0	
68	1/20/1978 12:00:00 AM	na	na	na	na	na	na	na	na	na	...	10.0	na	na	na	na	49.0	
19	02-08-1978 0:00	na	na	na	na	na	18.0	na	na	na	...	na	na	na	na	na	21.0	
91	03-08-1978 0:00	na	na	na	na	na	19.0	na	na	na	...	na	na	na	na	na	6.0	
13	04-04-1978 0:00	na	na	na	na	na	na	na	na	na	...	na	na	na	na	na	na	

5 rows × 75 columns

In [ ]:

```
# plotting all the 73 parameters.
# First drop all the rows with na values in them for that specific parameter

import matplotlib.pyplot as plt
df_new = fdf_values[fdf_values['NH3_N_diss_ugl Nitrogen, ammonia as N']!="na"]
plt.figure(figsize=(10,5))
plt.plot(df_new['Date'], df_new['NH3_N_diss_ugl Nitrogen, ammonia as N'])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel("NH3_N_diss_ugl Nitrogen, ammonia as N")
plt.show()
```

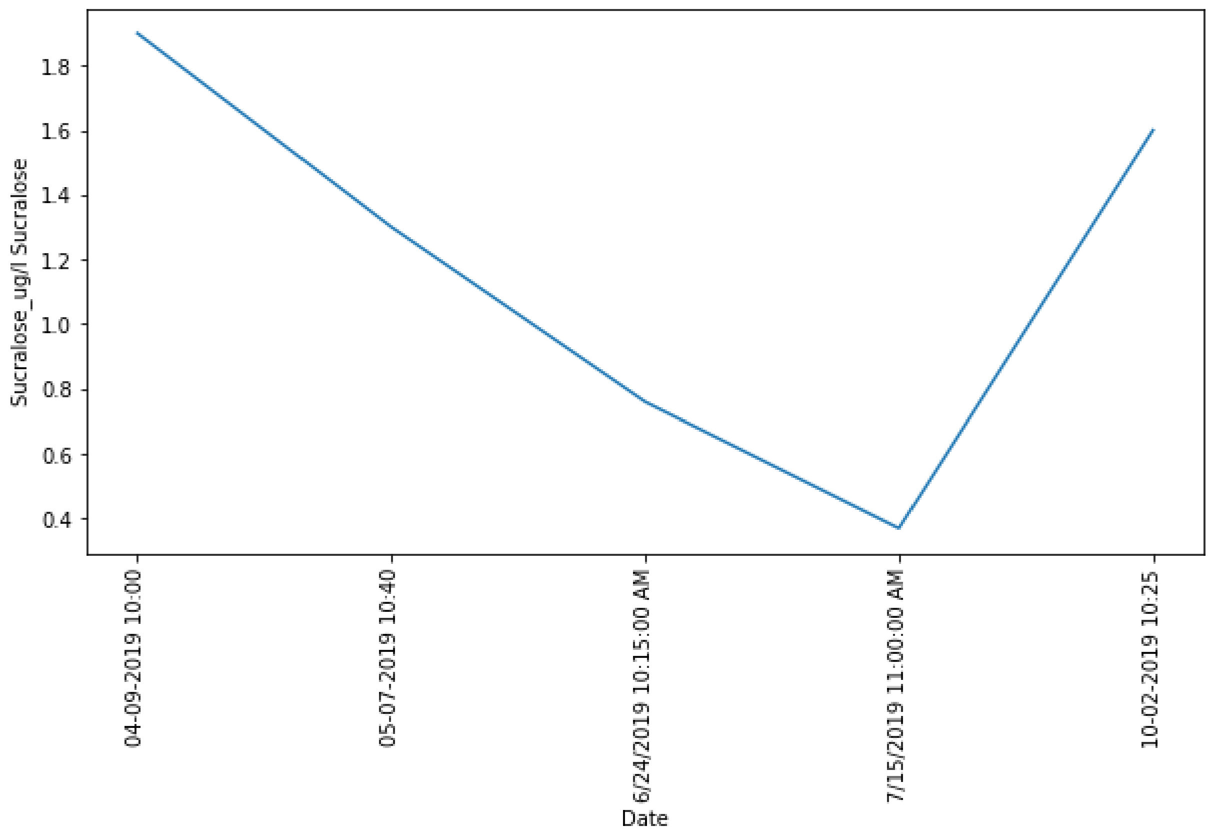


In [ ]:

```
col = fdf_values.columns
```

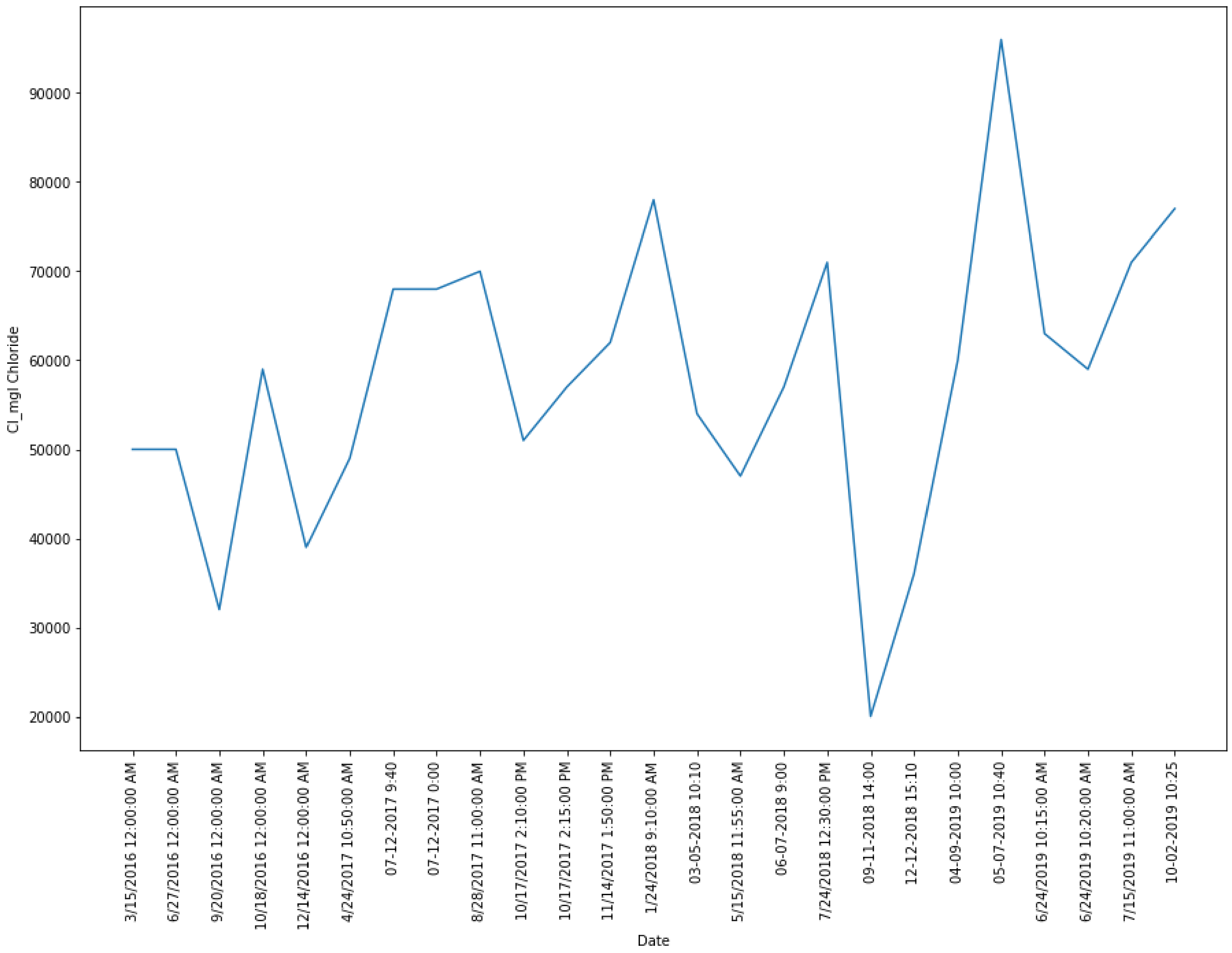
In [ ]:

```
df_new = fdf_values[fdf_values[col[2]]!="na"]
plt.figure(figsize=(10,5))
plt.plot(df_new['Date'], df_new[col[2]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[2])
plt.show()
```



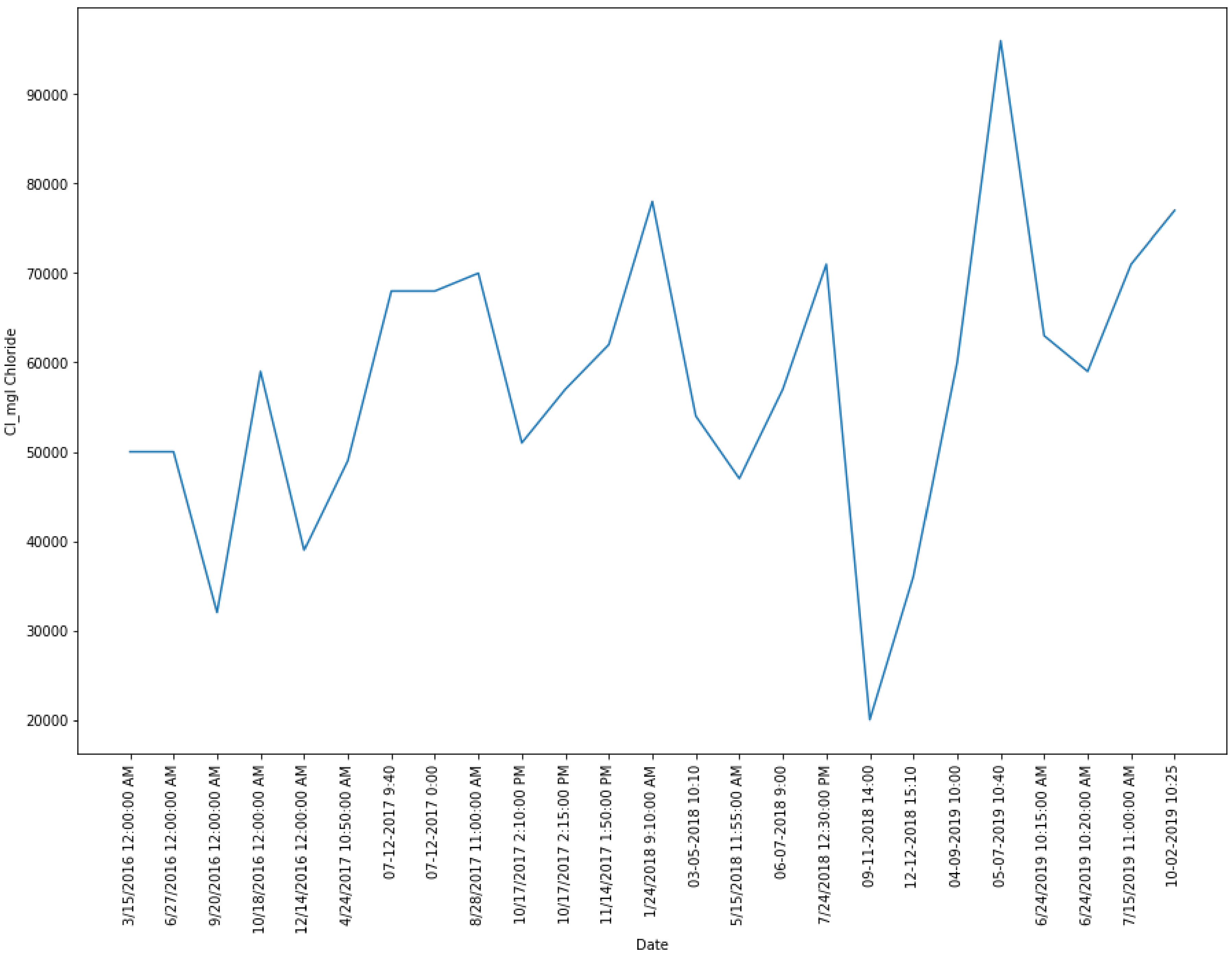
In [ ]:

```
df_new = fdf_values[fdf_values[col[3]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[3]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[3])
plt.show()
```



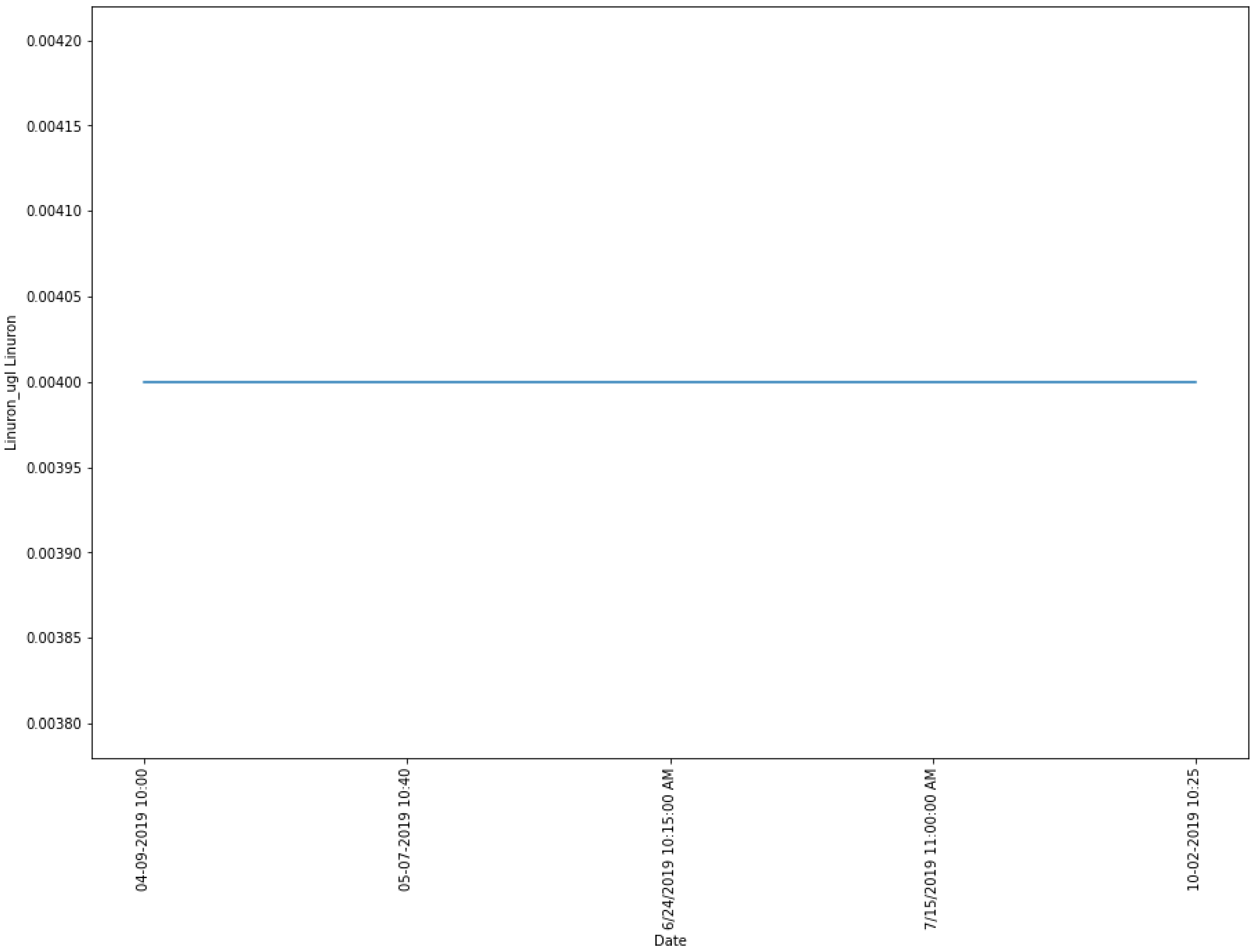
In [ ]:

```
df_new = fdf_values[fdf_values[col[3]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[3]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[3])
plt.show()
```



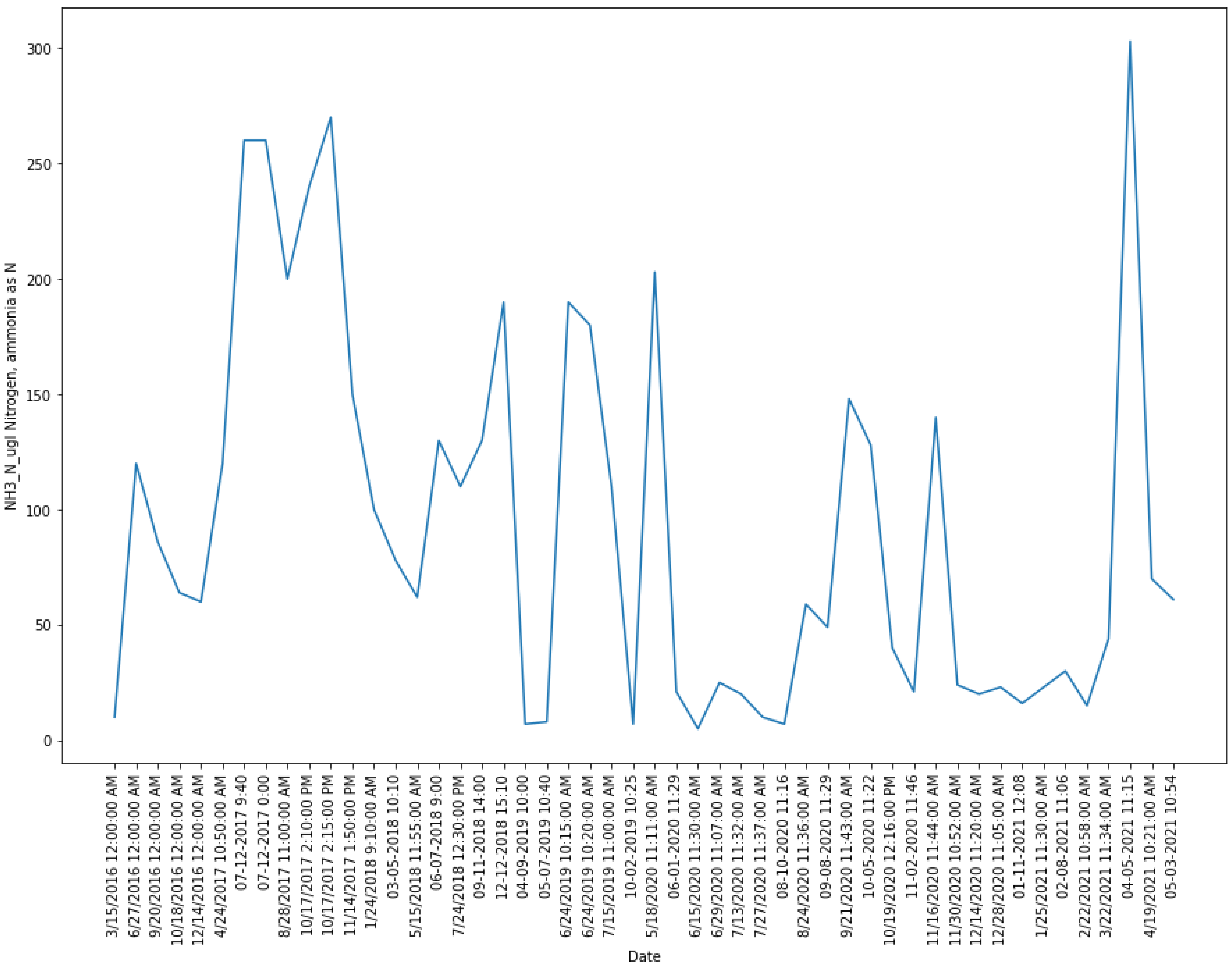
In [ ]:

```
df_new = fdf_values[fdf_values[col[4]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[4]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[4])
plt.show()
```



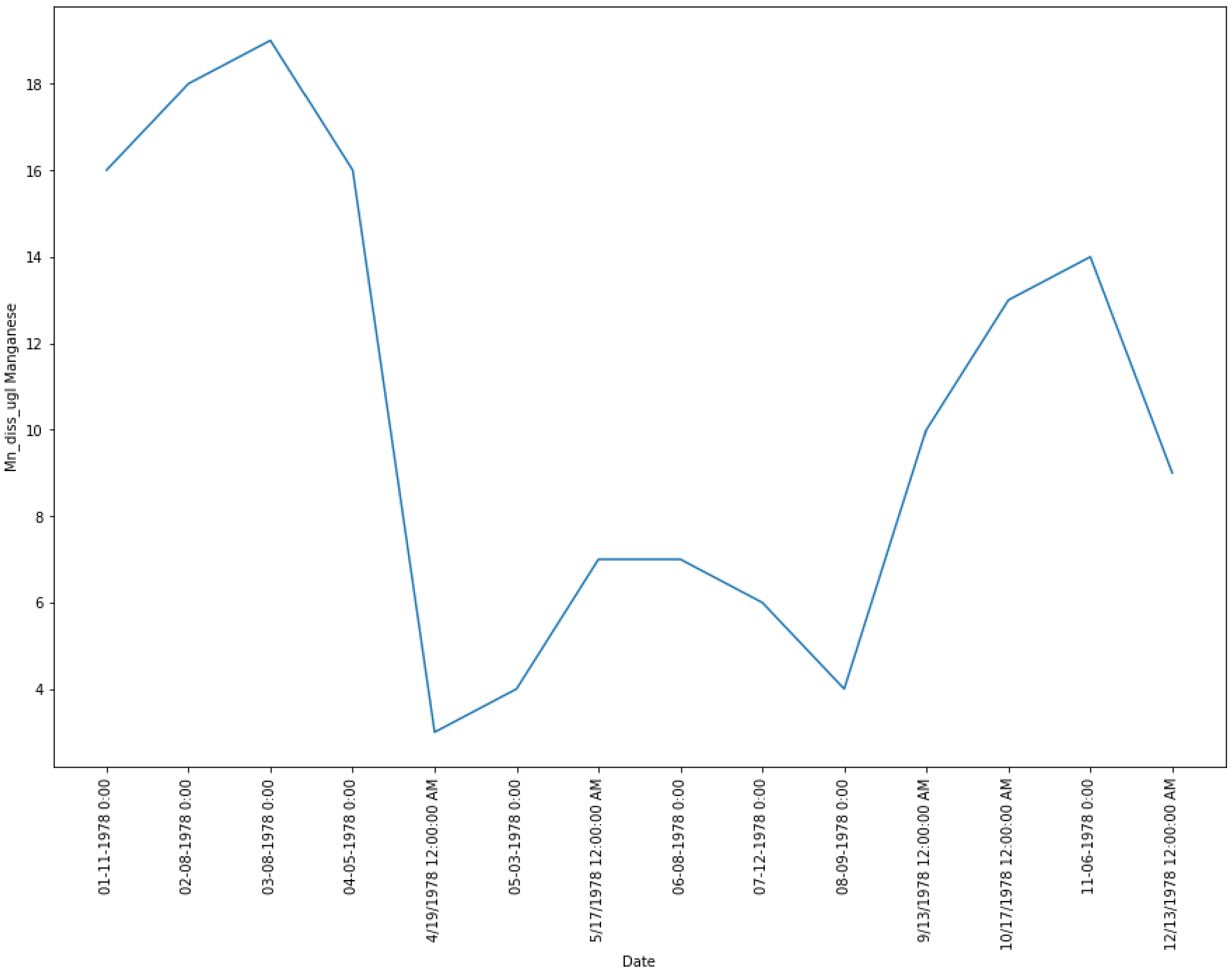
In [ ]:

```
df_new = fdf_values[fdf_values[col[5]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[5]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[5])
plt.show()
```



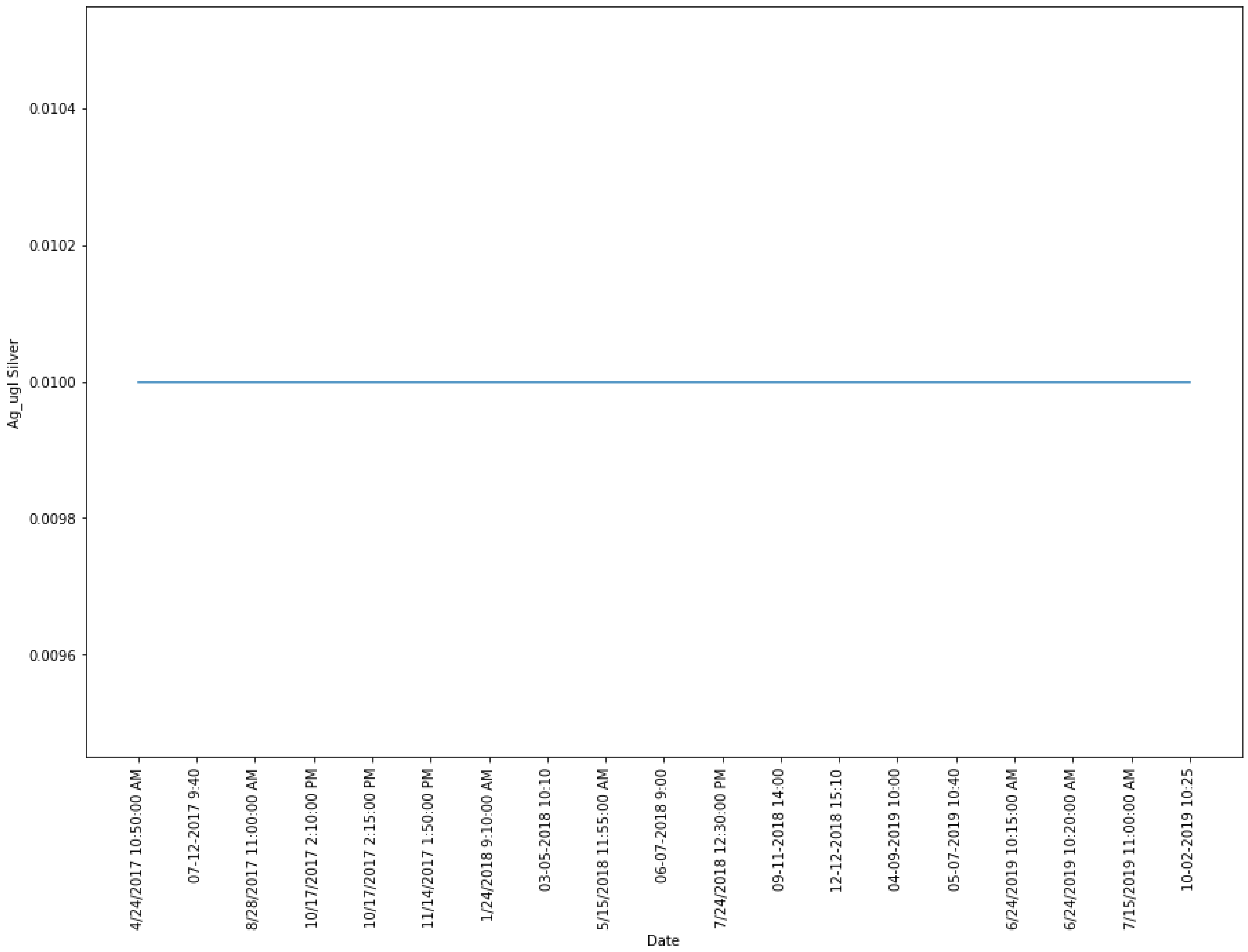
In [ ]:

```
df_new = fdf_values[fdf_values[col[6]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[6]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[6])
plt.show()
```



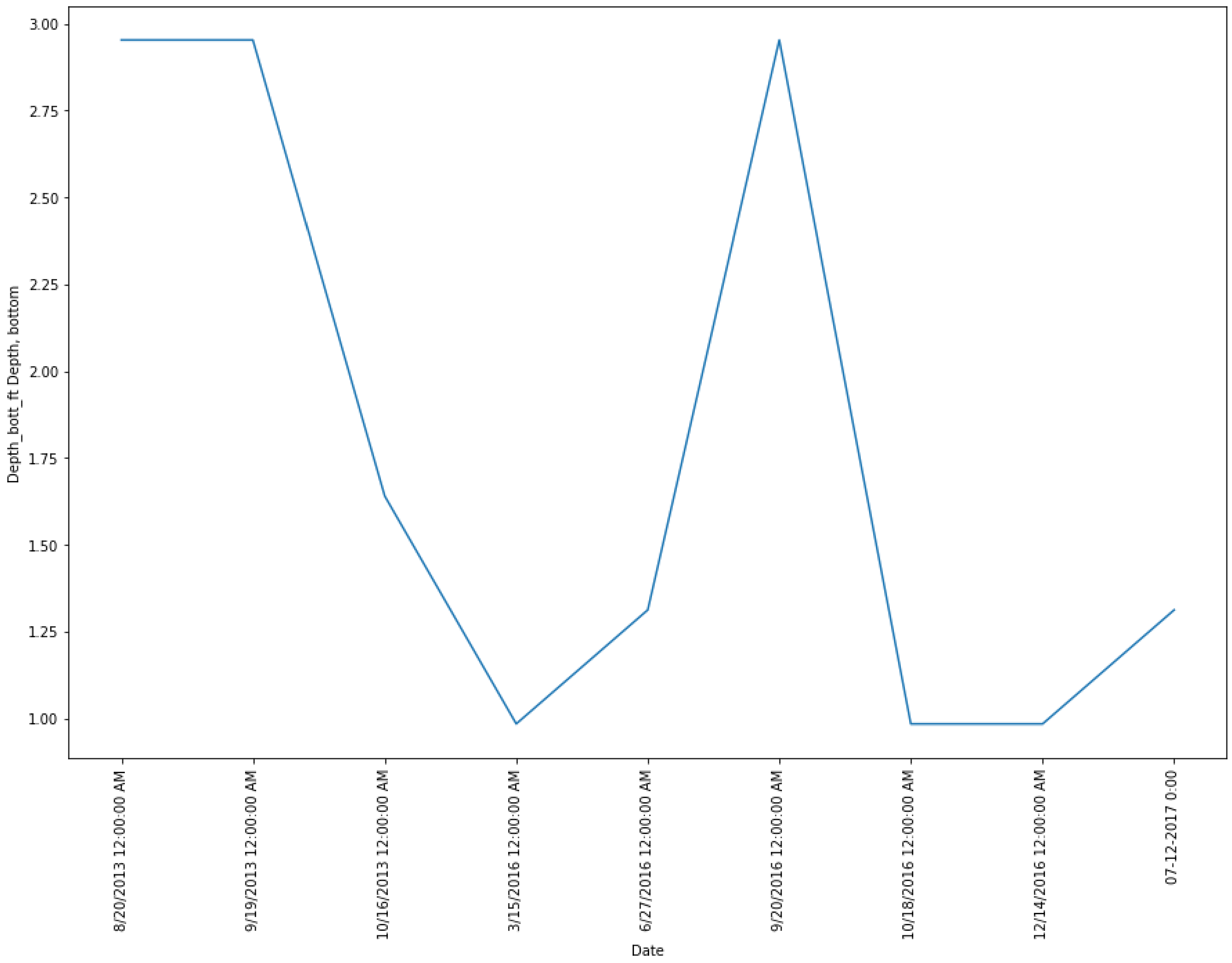
In [ ]:

```
df_new = fdf_values[fdf_values[col[7]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[7]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[7])
plt.show()
```



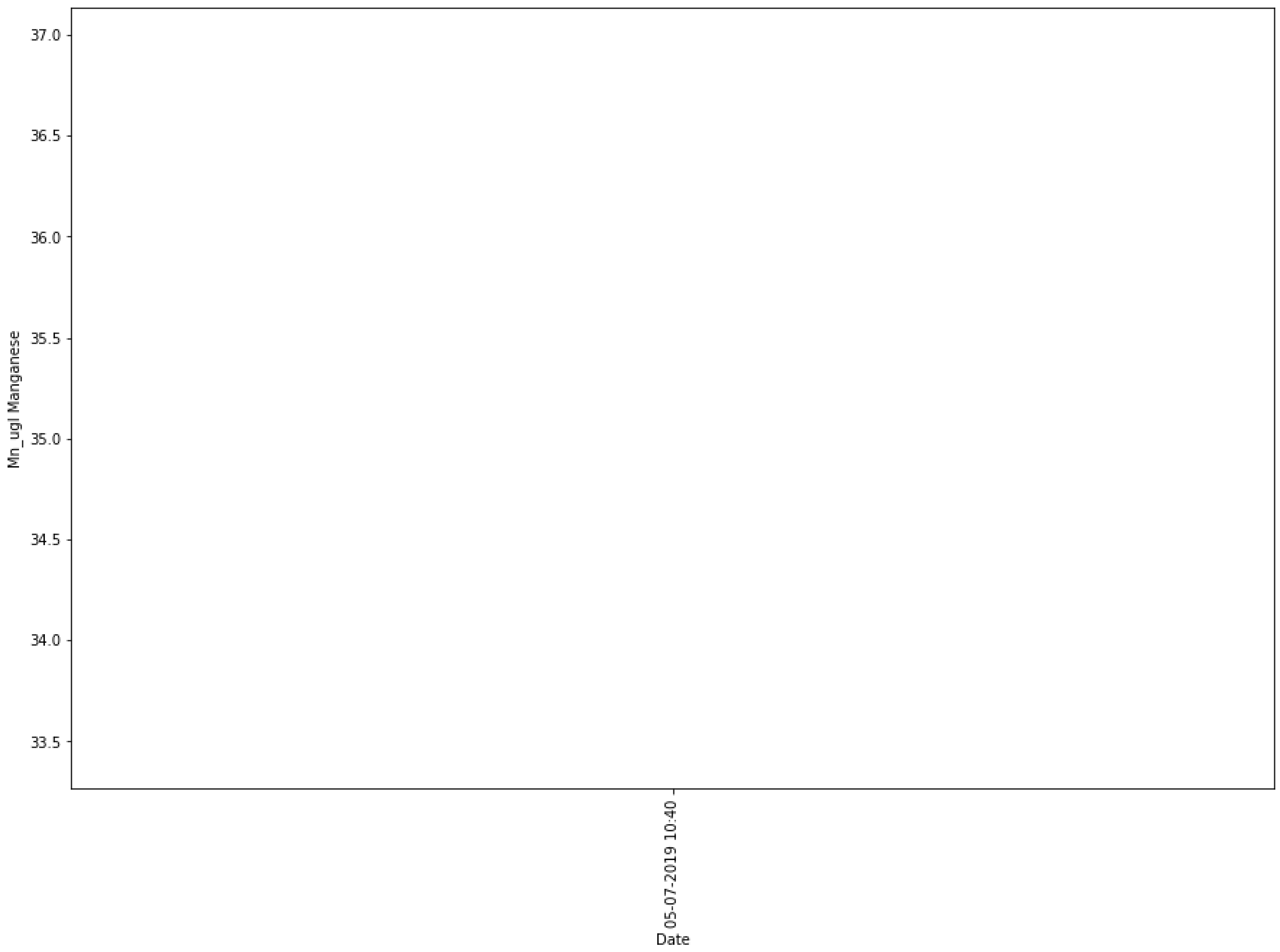
In [ ]:

```
df_new = fdf_values[fdf_values[col[8]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[8]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[8])
plt.show()
```



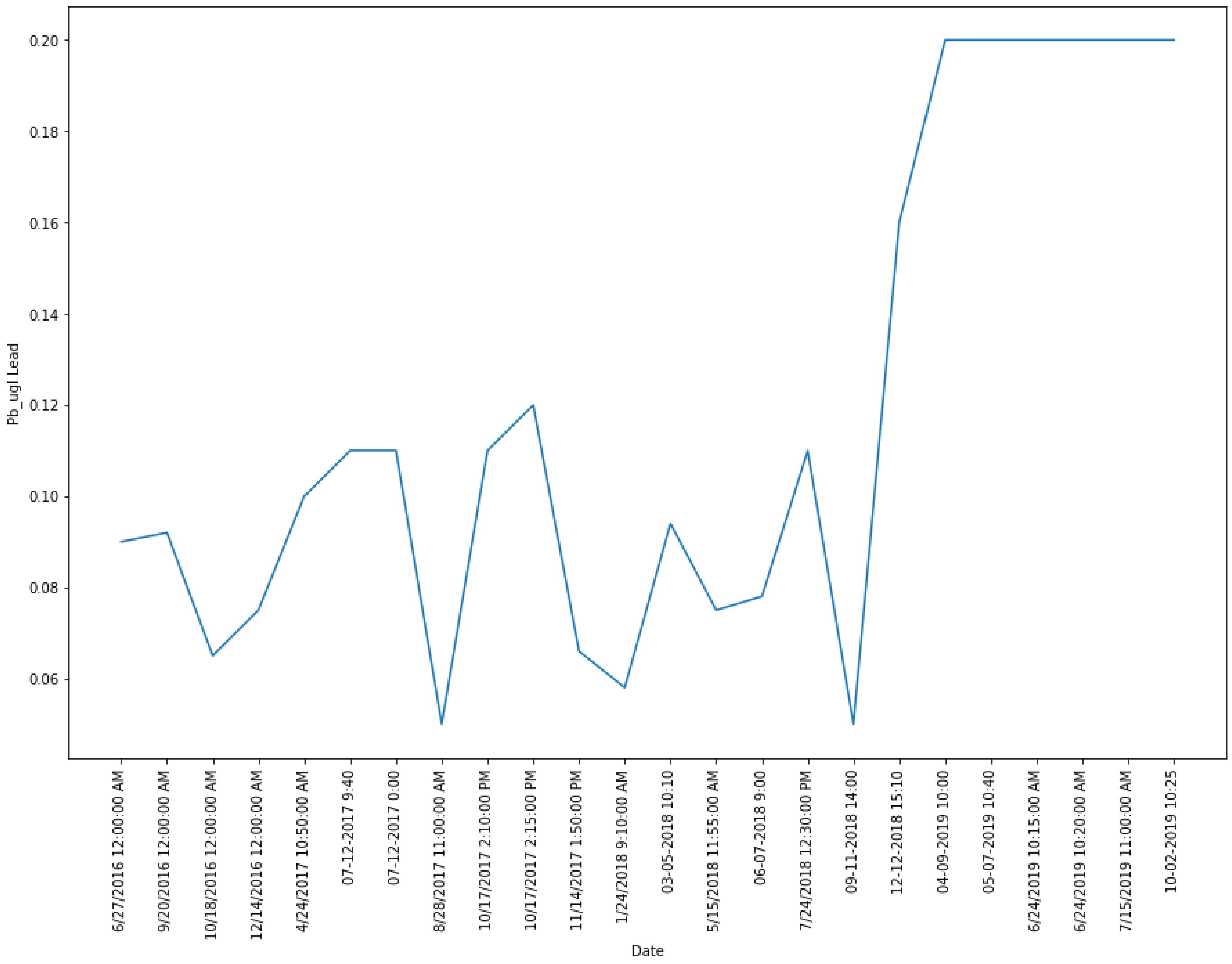
In [ ]:

```
df_new = fdf_values[fdf_values[col[9]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[9]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[9])
plt.show()
```



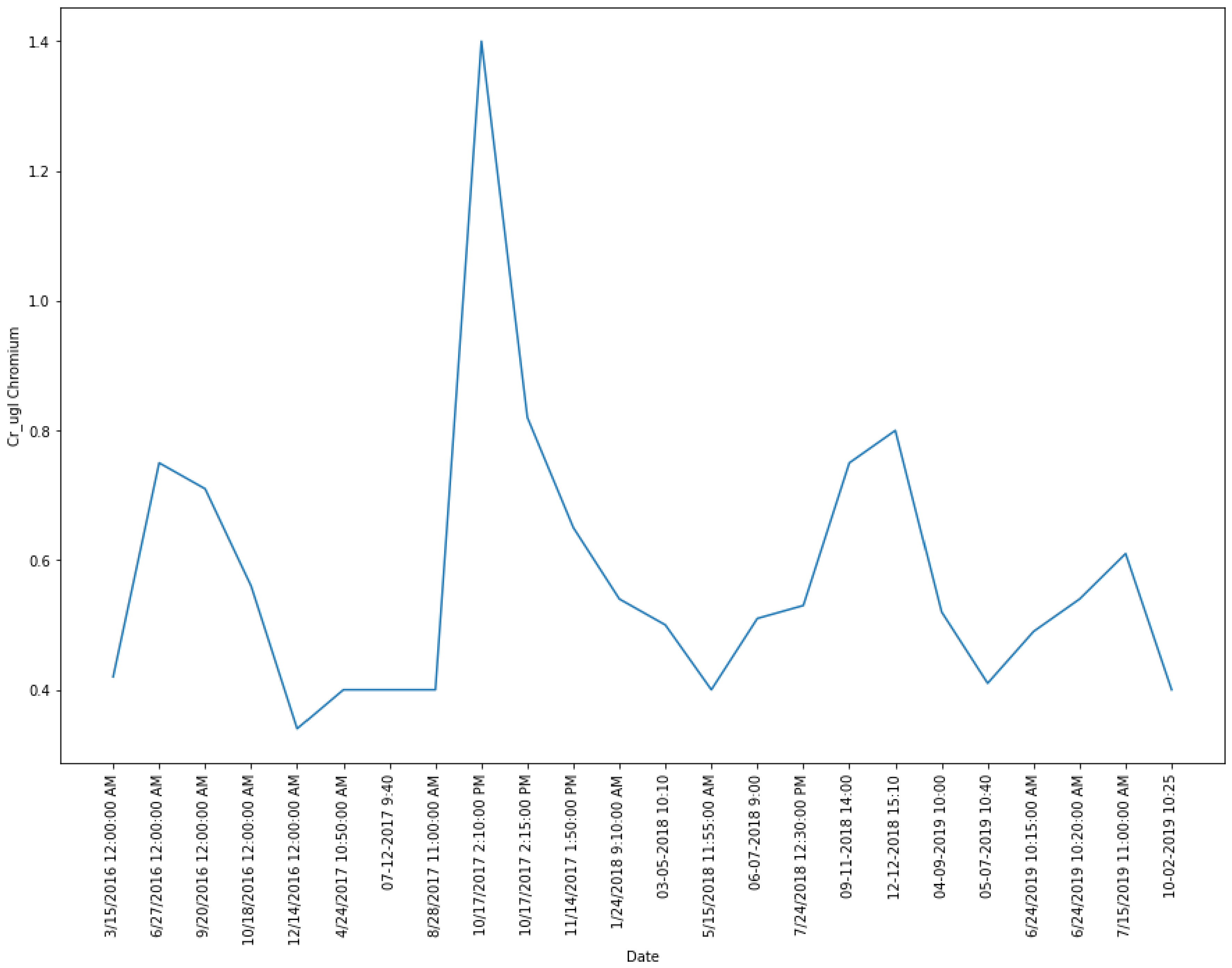
In [ ]:

```
df_new = fdf_values[fdf_values[col[10]]!="na"]
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[10]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[10])
plt.show()
```



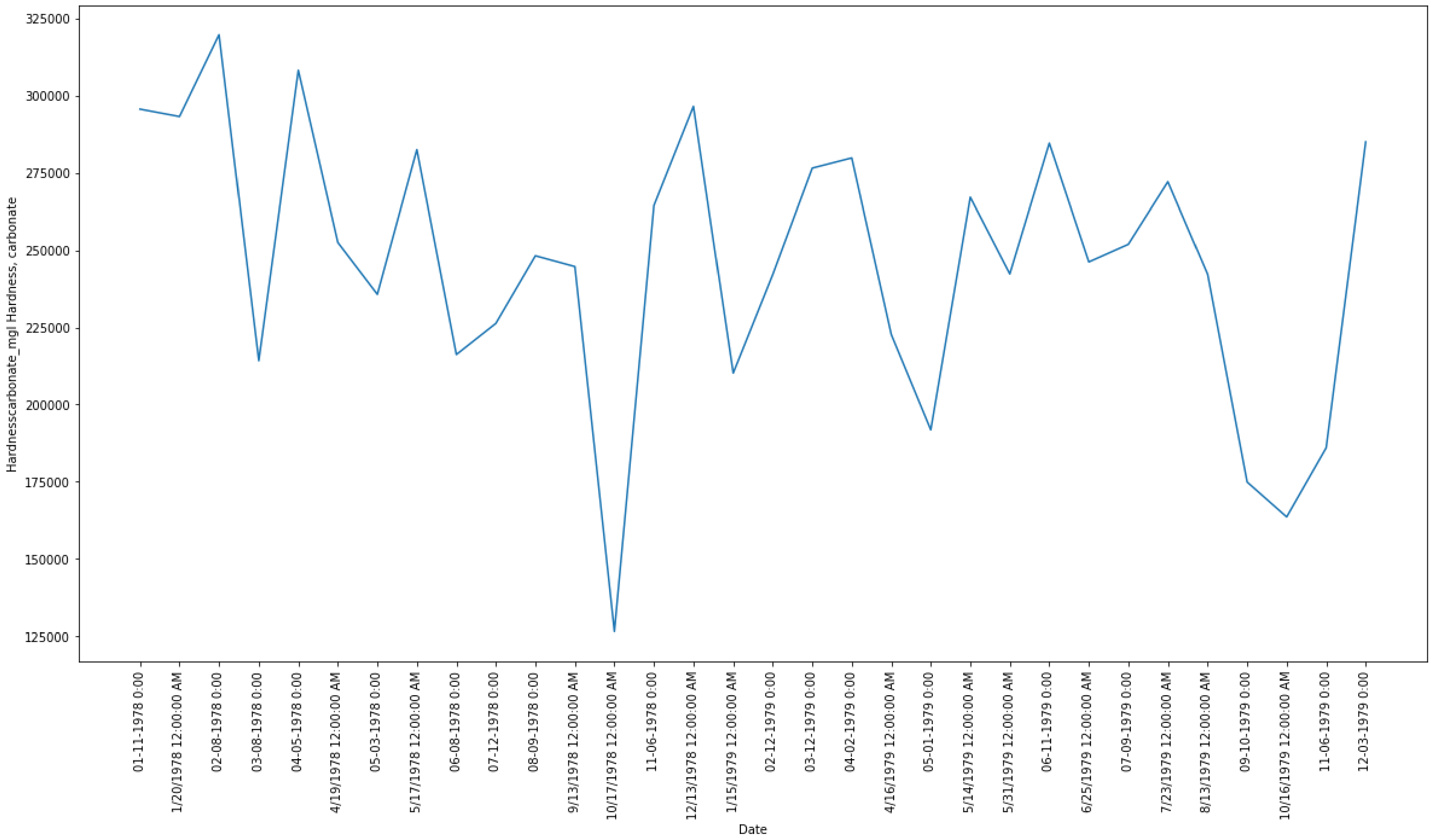
In [ ]:

```
df_new = fdf_values[fdf_values[col[11]]!='na']
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[11]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[11])
plt.show()
```



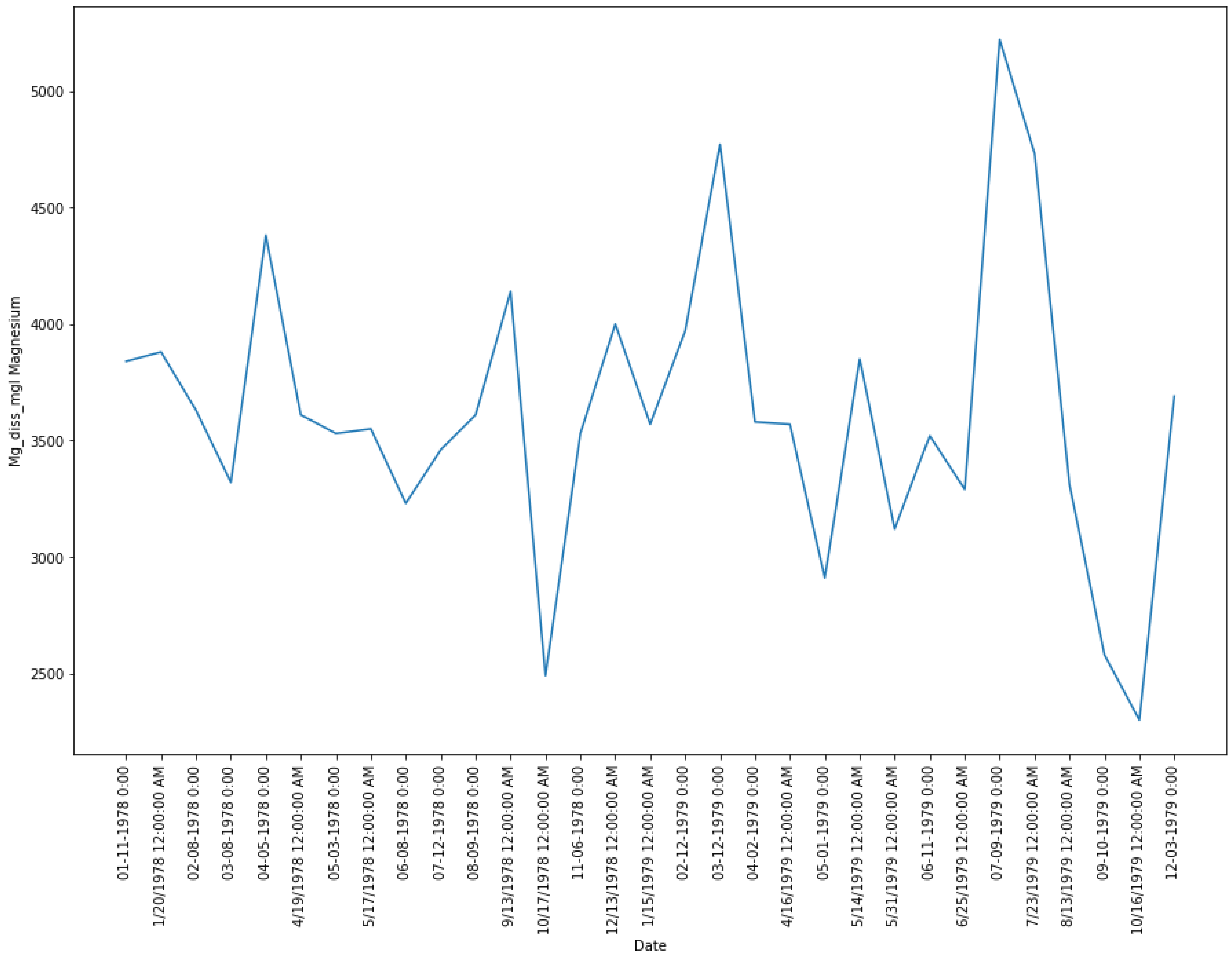
In [ ]:

```
df_new = fdf_values[fdf_values[col[12]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[12]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[12])
plt.show()
```

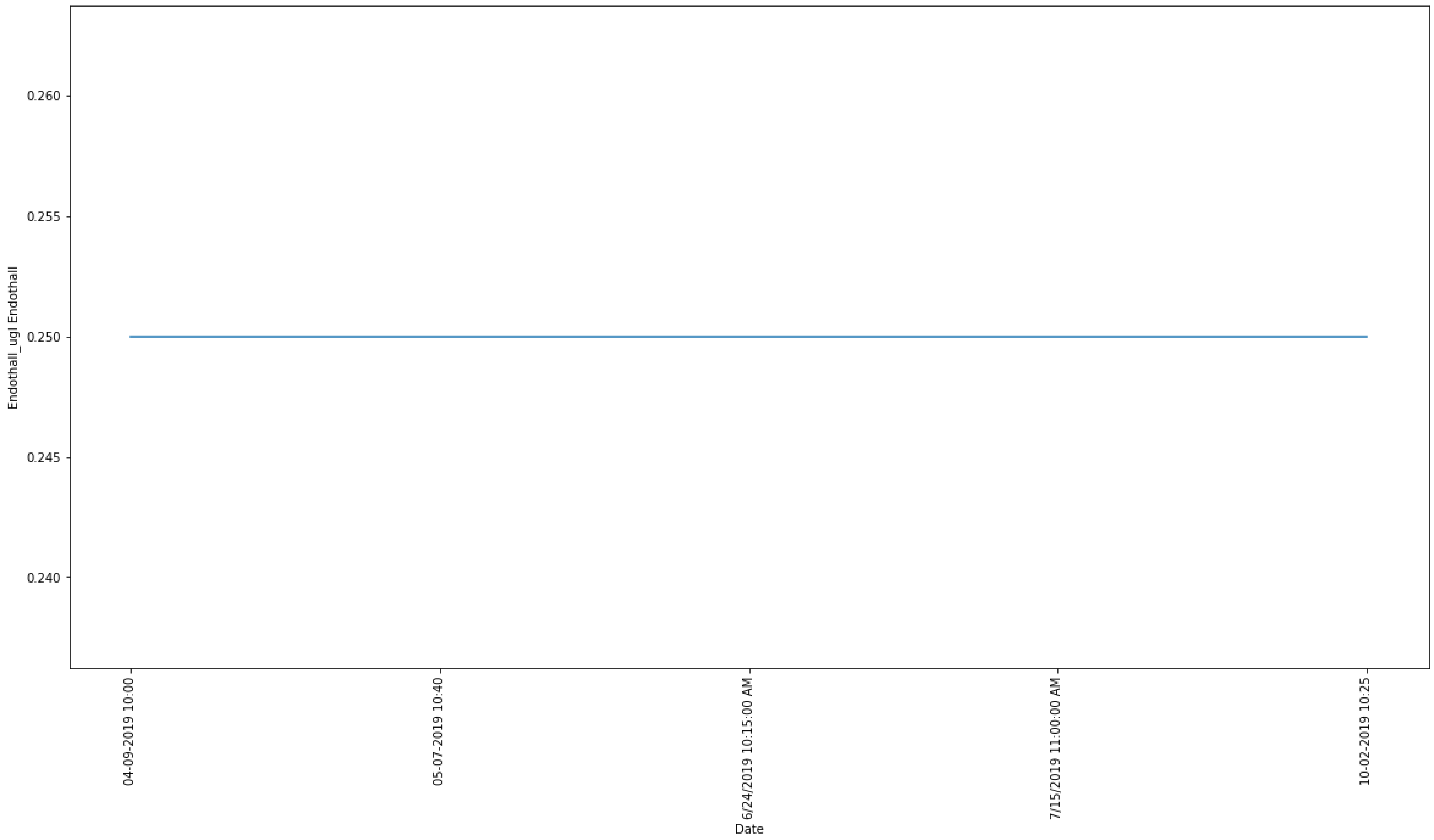


In [ ]:

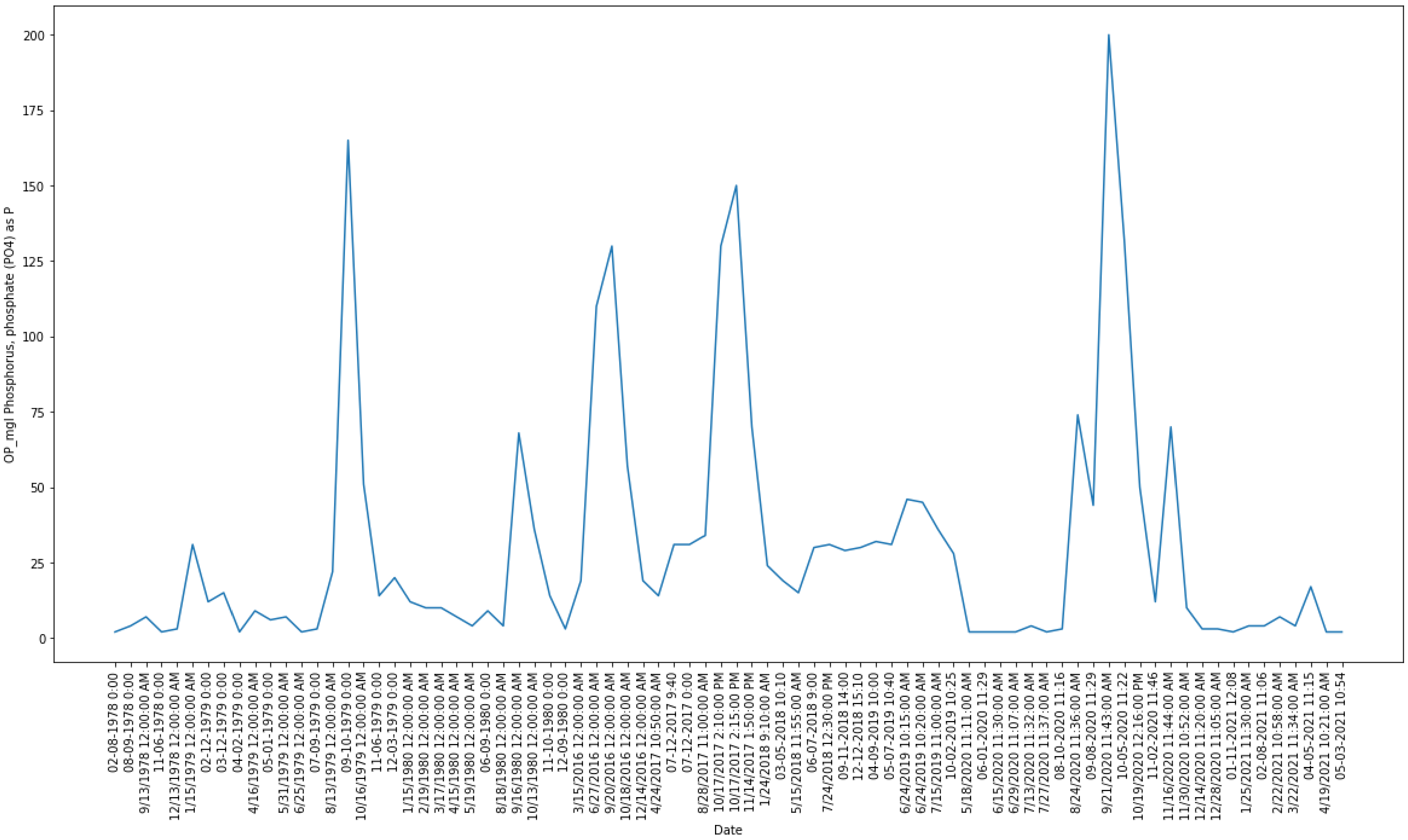
```
df_new = fdf_values[fdf_values[col[13]]!='na']
plt.figure(figsize=(15,10))
plt.plot(df_new['Date'], df_new[col[13]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[13])
plt.show()
```



```
In [ ]:
df_new = fdf_values[fdf_values[col[14]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[14]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[14])
plt.show()
```

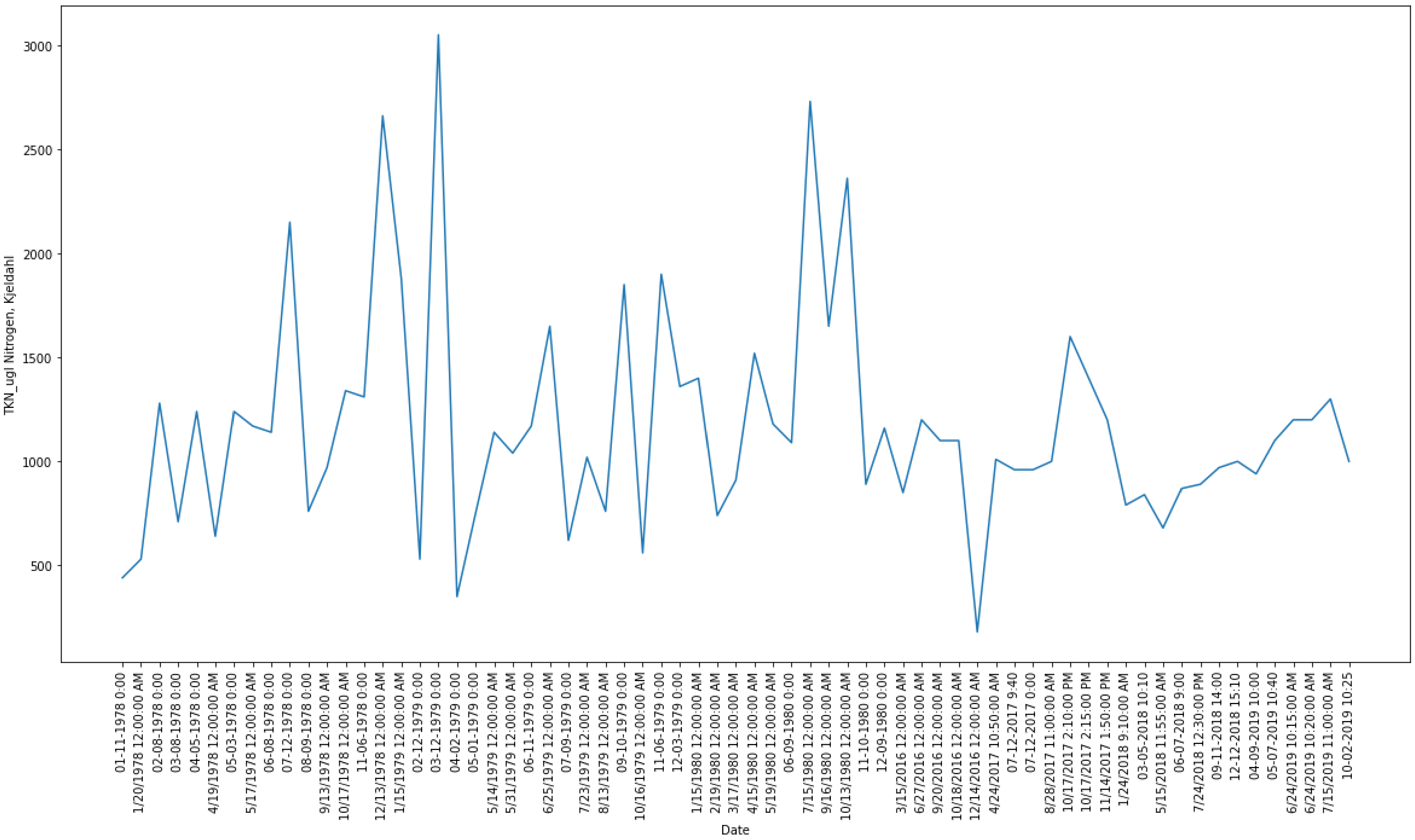


```
In [ ]:
df_new = fdf_values[fdf_values[col[15]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[15]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[15])
plt.show()
```



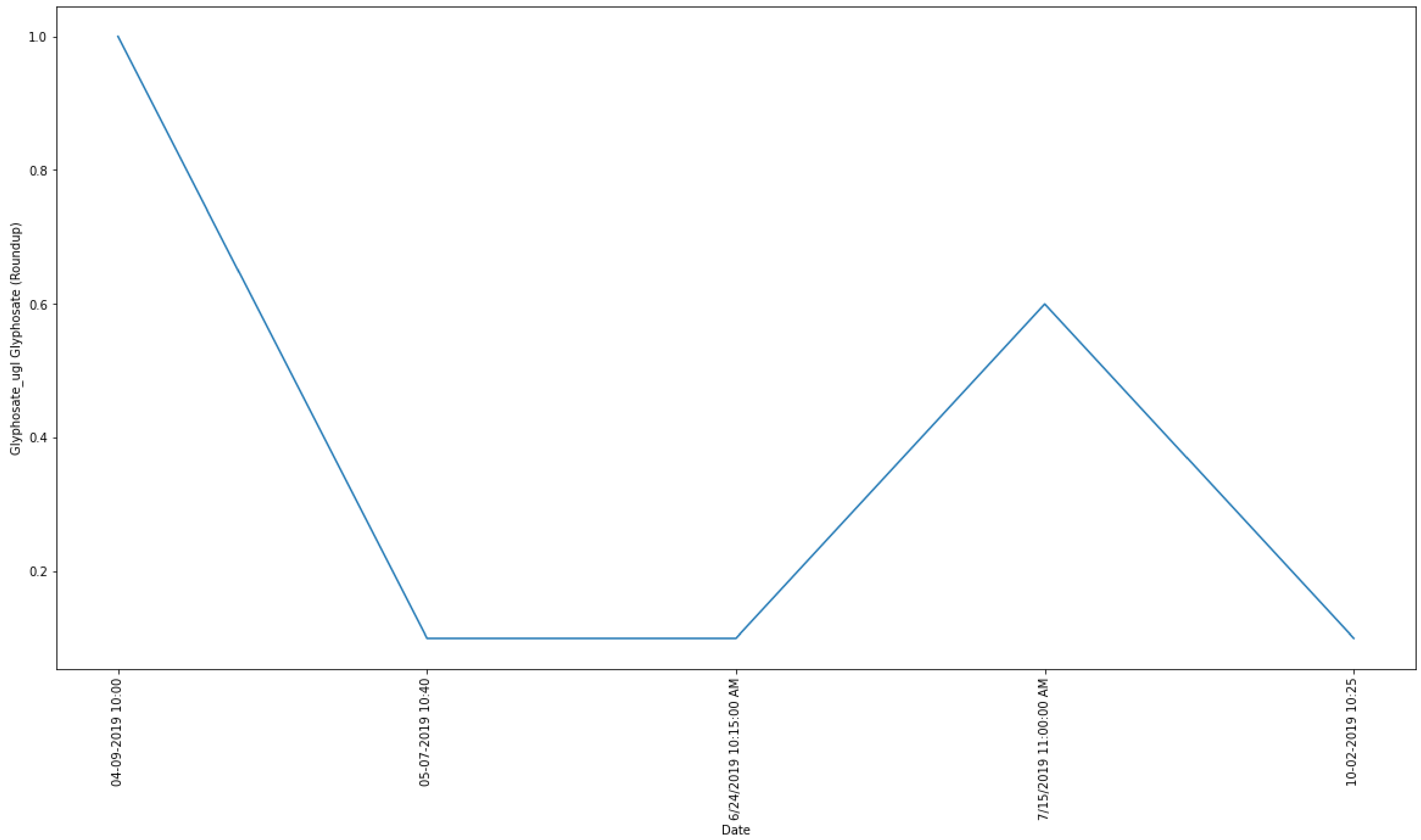
In [ ]:

```
df_new = fdf_values[fdf_values[col[16]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[16]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[16])
plt.show()
```



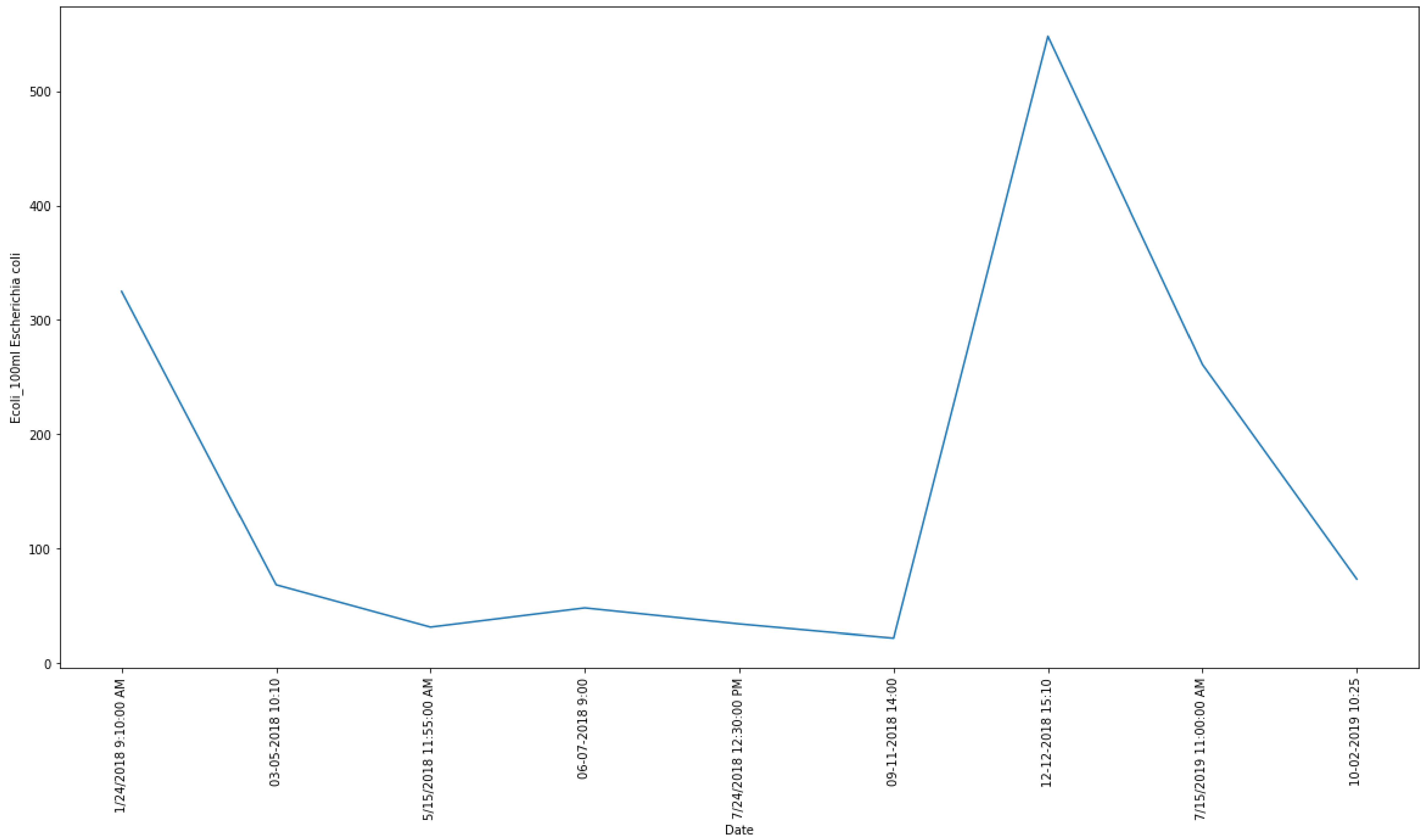
In [ ]:

```
df_new = fdf_values[fdf_values[col[17]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[17]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[17])
plt.show()
```



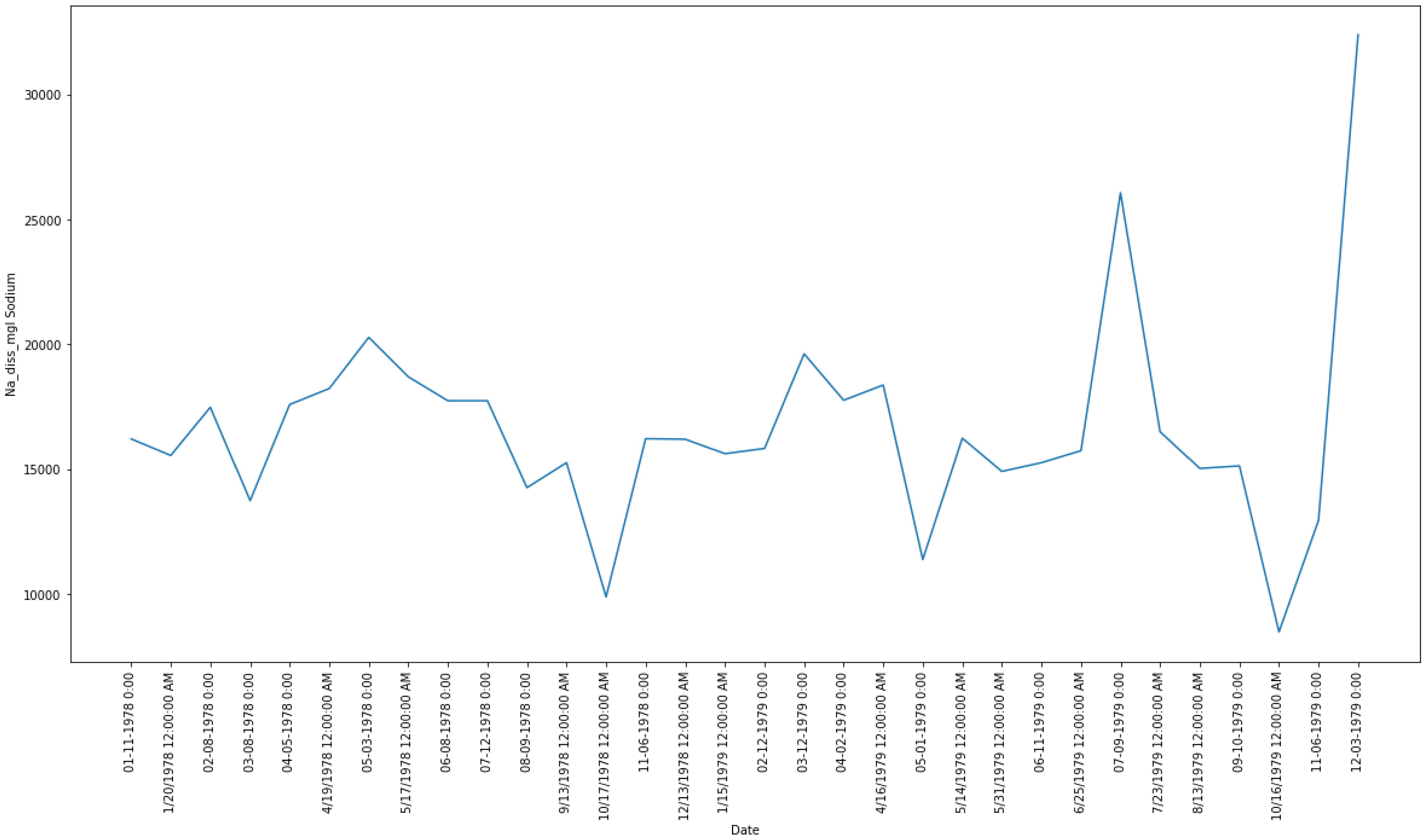
In [ ]:

```
df_new = fdf_values[fdf_values[col[18]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[18]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[18])
plt.show()
```



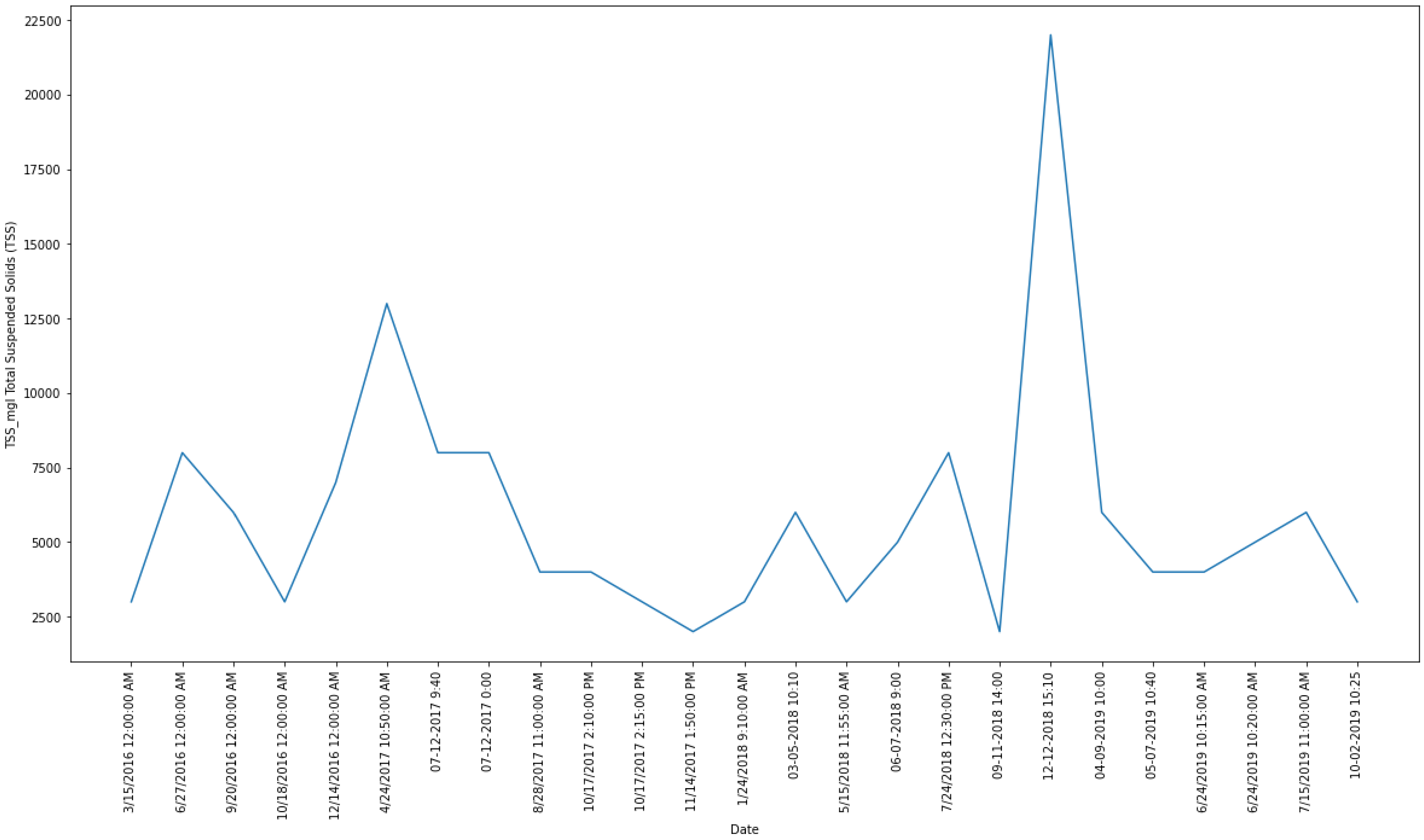
In [ ]:

```
df_new = fdf_values[fdf_values[col[19]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[19]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[19])
plt.show()
```



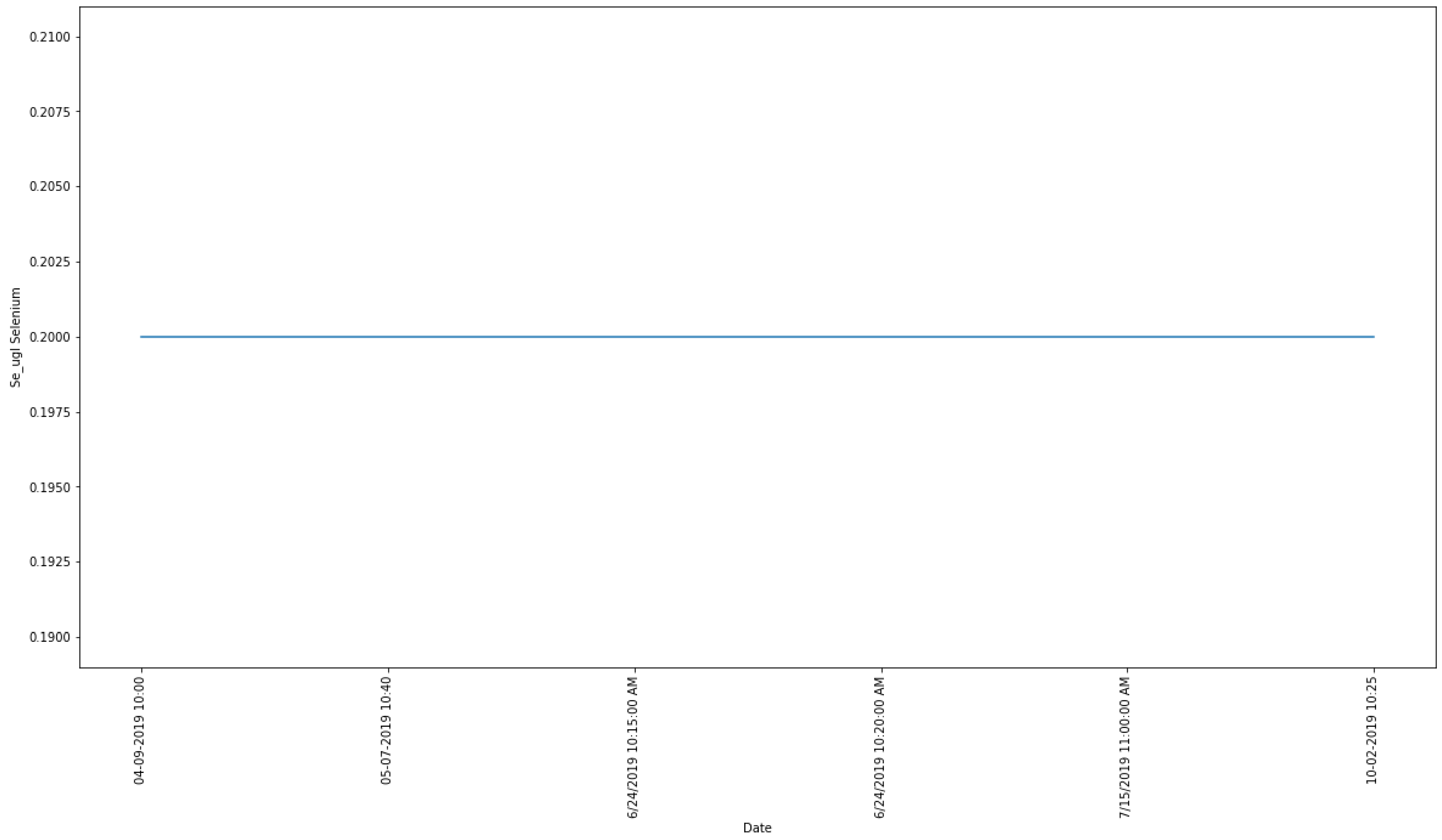
In [ ]:

```
df_new = fdf_values[fdf_values[col[20]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[20]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[20])
plt.show()
```



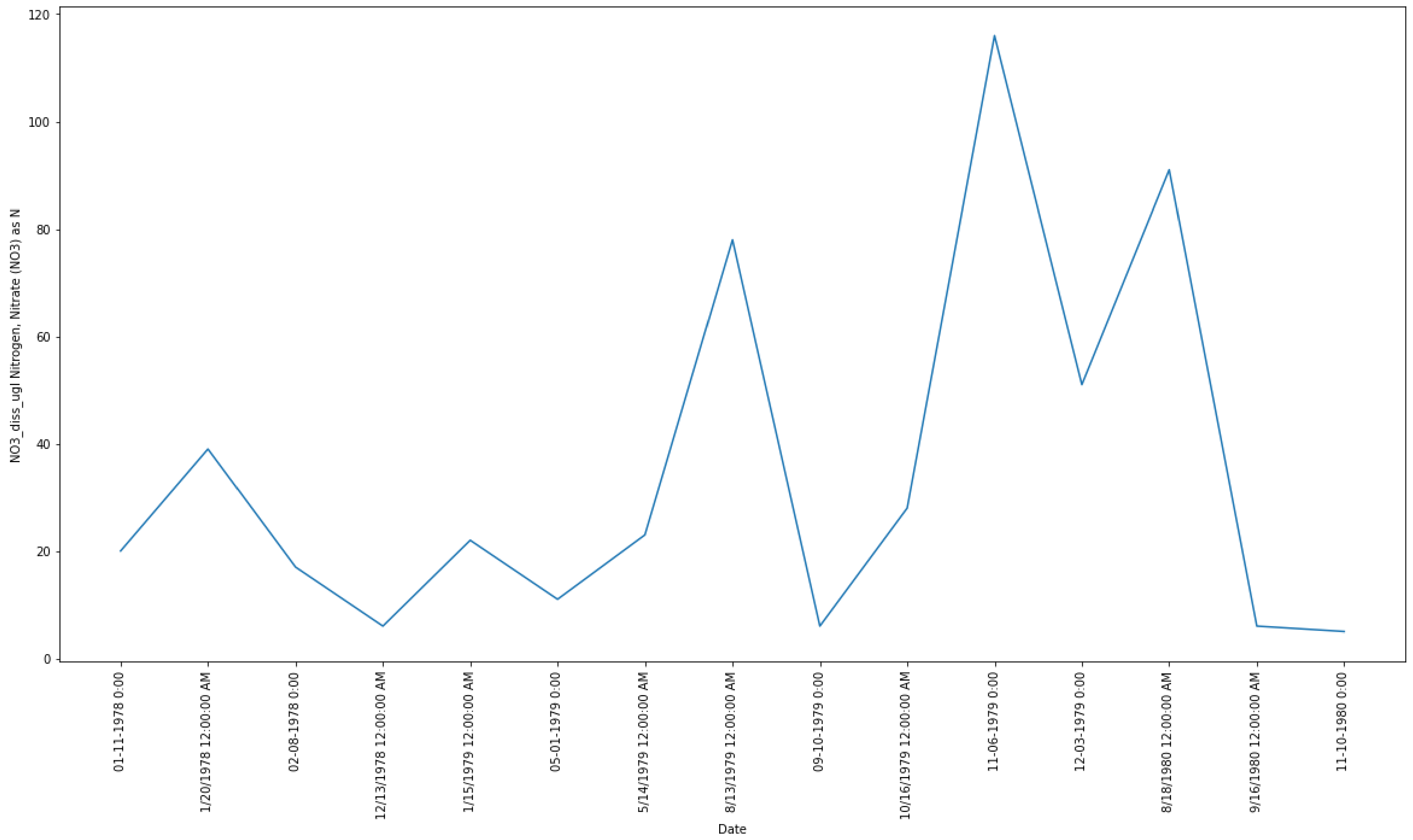
In [ ]:

```
df_new = fdf_values[fdf_values[col[21]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[21]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[21])
plt.show()
```



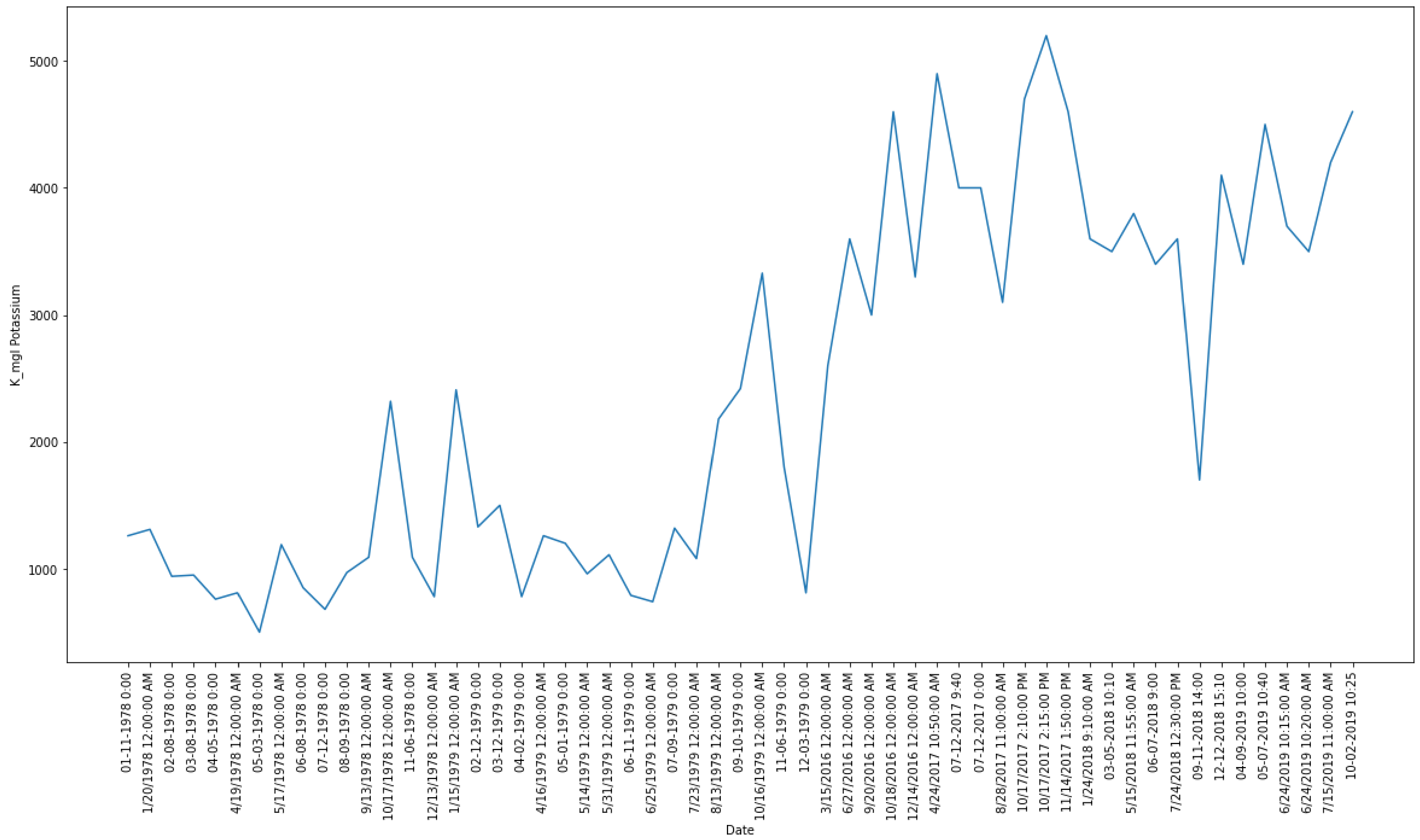
In [ ]:

```
df_new = fdf_values[fdf_values[col[22]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[22]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[22])
plt.show()
```



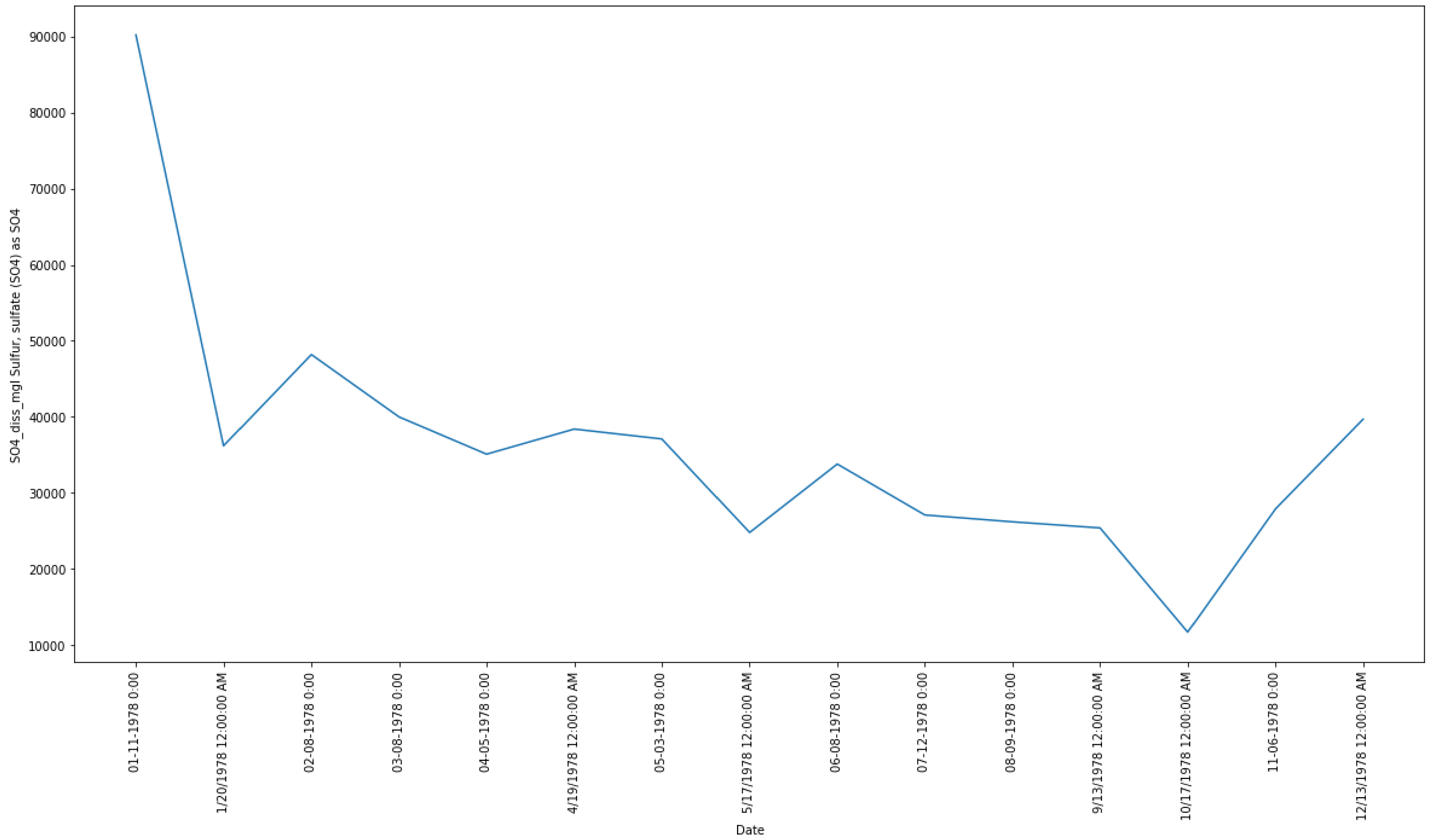
In [ ]:

```
df_new = fdf_values[fdf_values[col[23]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[23]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[23])
plt.show()
```



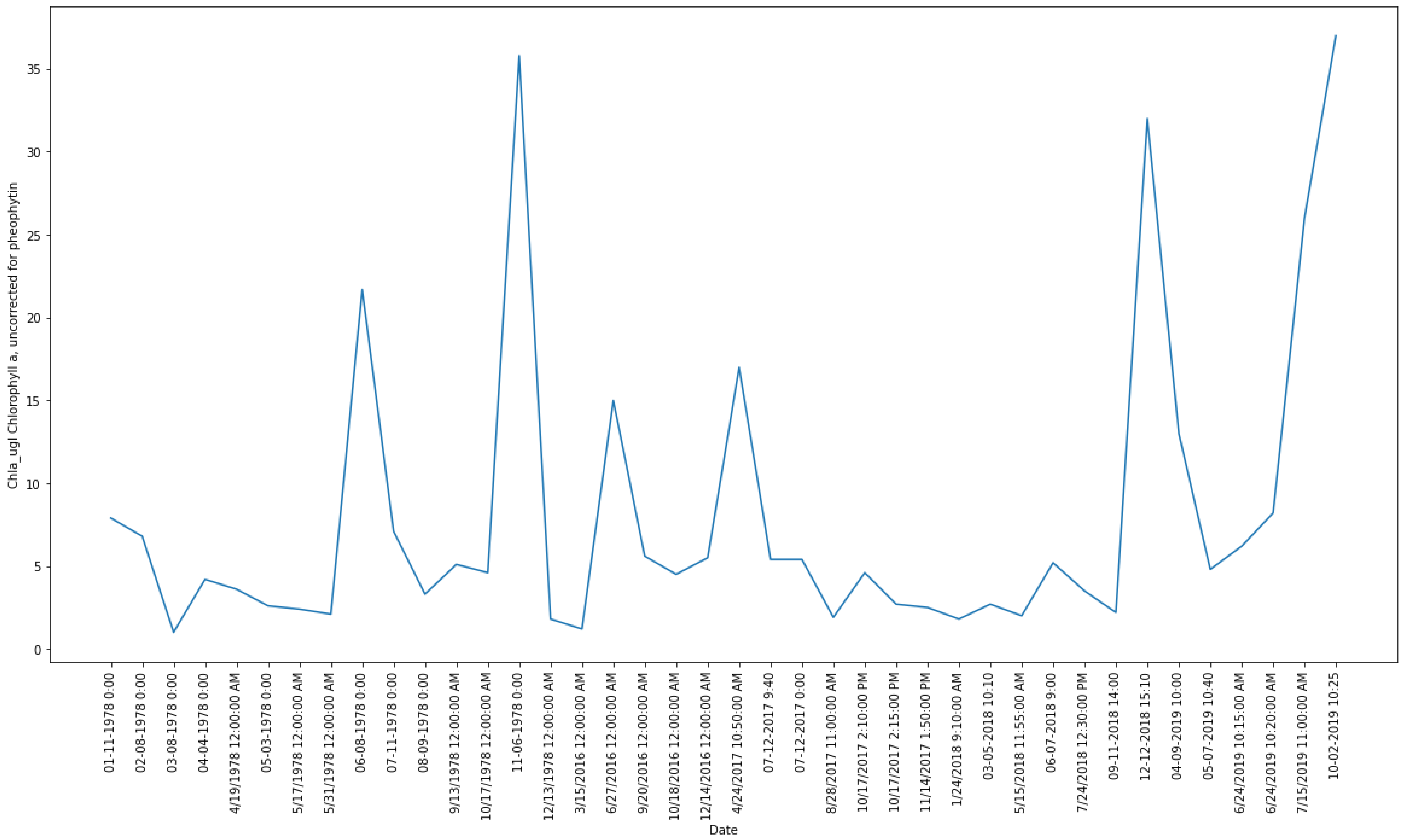
In [ ]:

```
df_new = fdf_values[fdf_values[col[24]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[24]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[24])
plt.show()
```



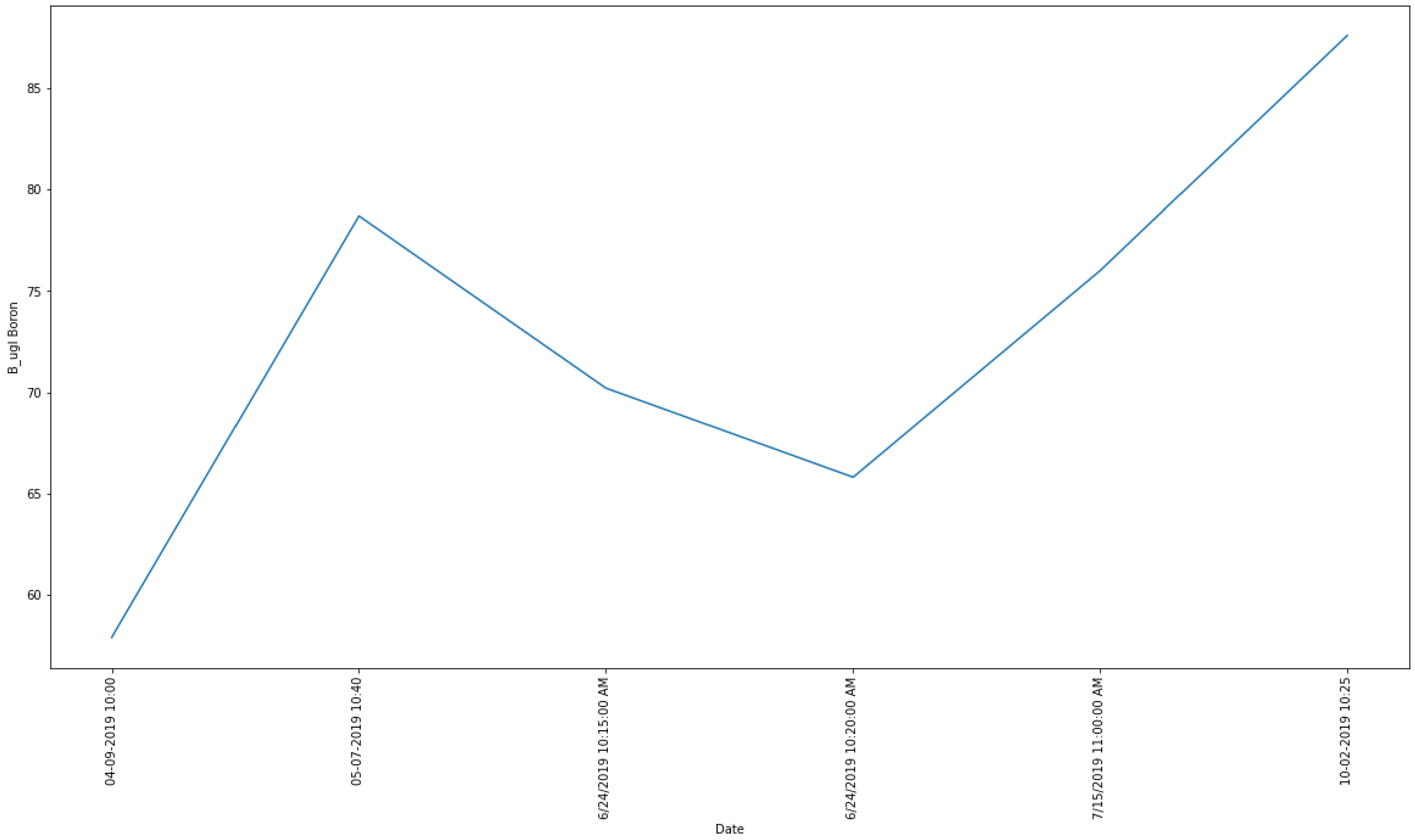
In [ ]:

```
df_new = fdf_values[fdf_values[col[25]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[25]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[25])
plt.show()
```



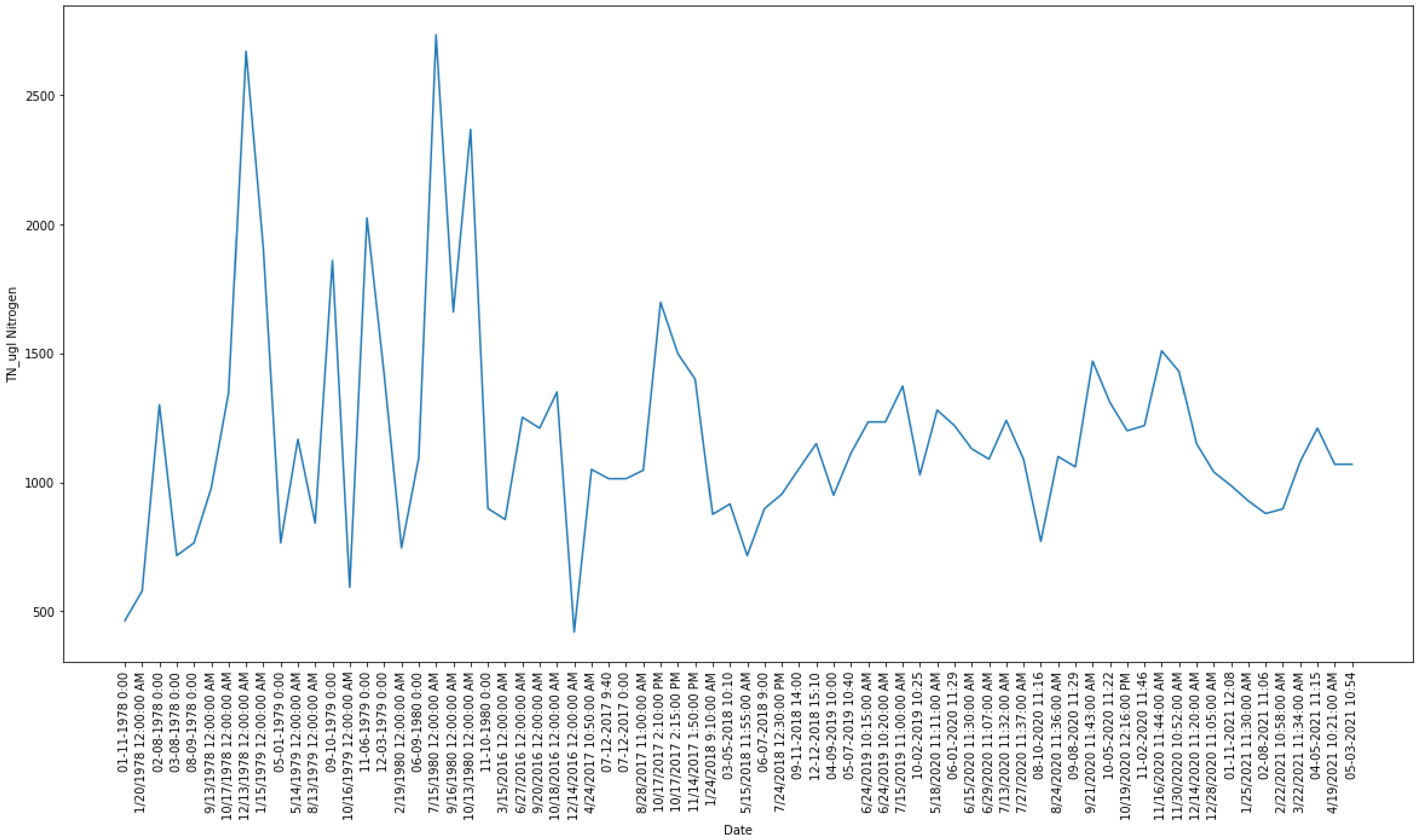
In [ ]:

```
df_new = fdf_values[fdf_values[col[26]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[26]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[26])
plt.show()
```



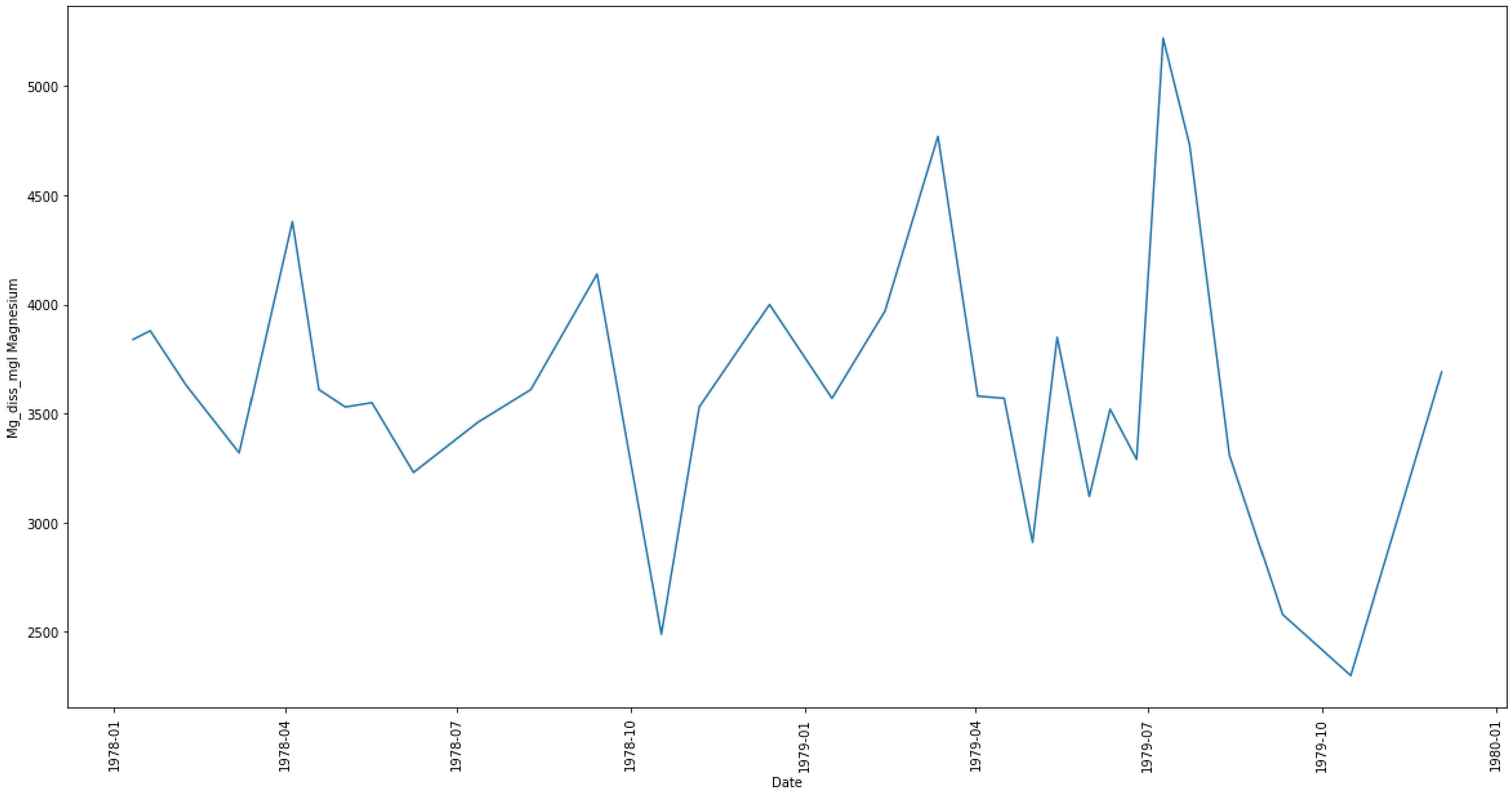
In [ ]:

```
df_new = fdf_values[fdf_values[col[27]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[27]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[27])
plt.show()
```



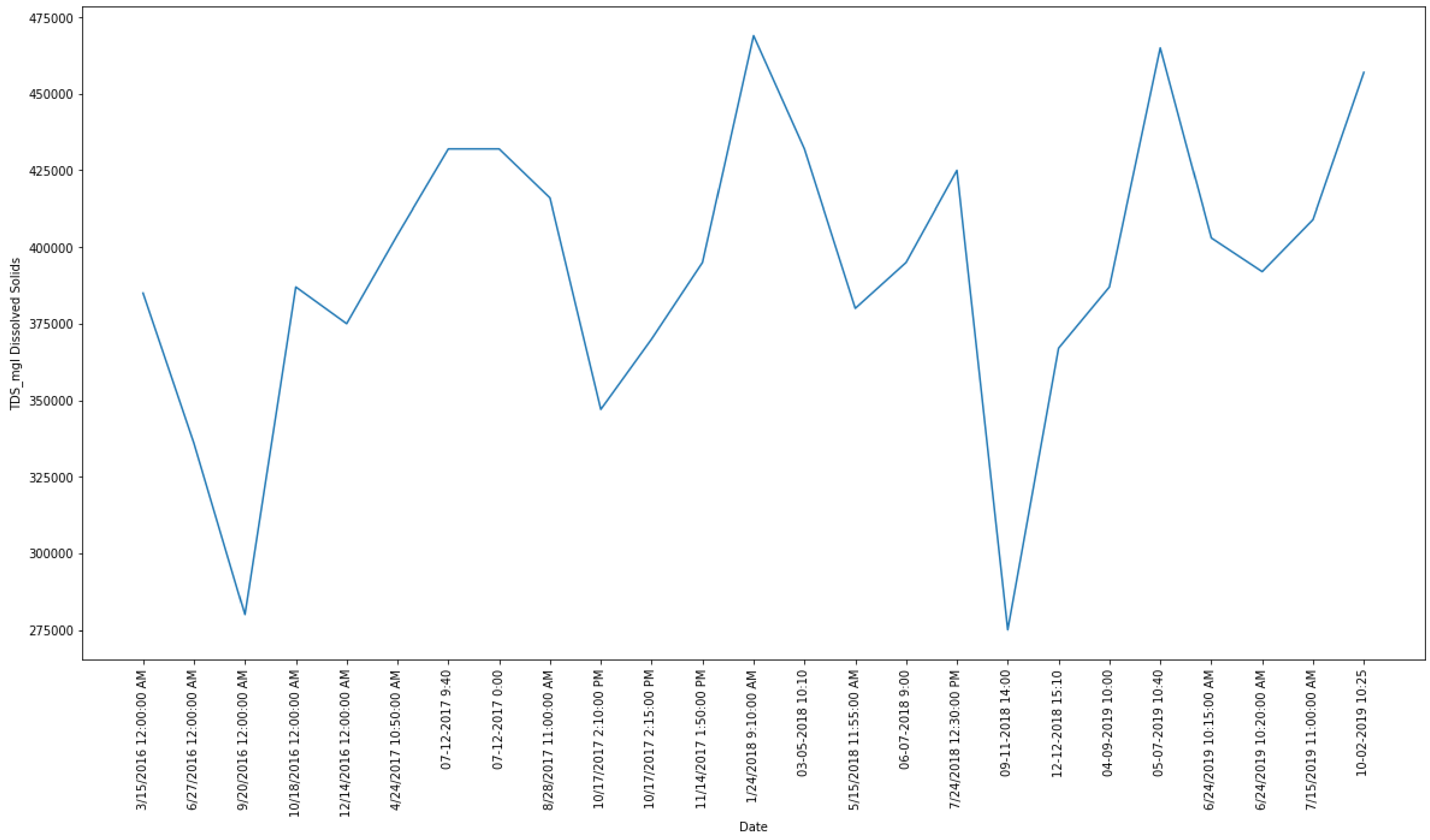
In [ ]:

```
df_new = fdf_values[fdf_values[col[28]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[28]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[28])
plt.show()
```



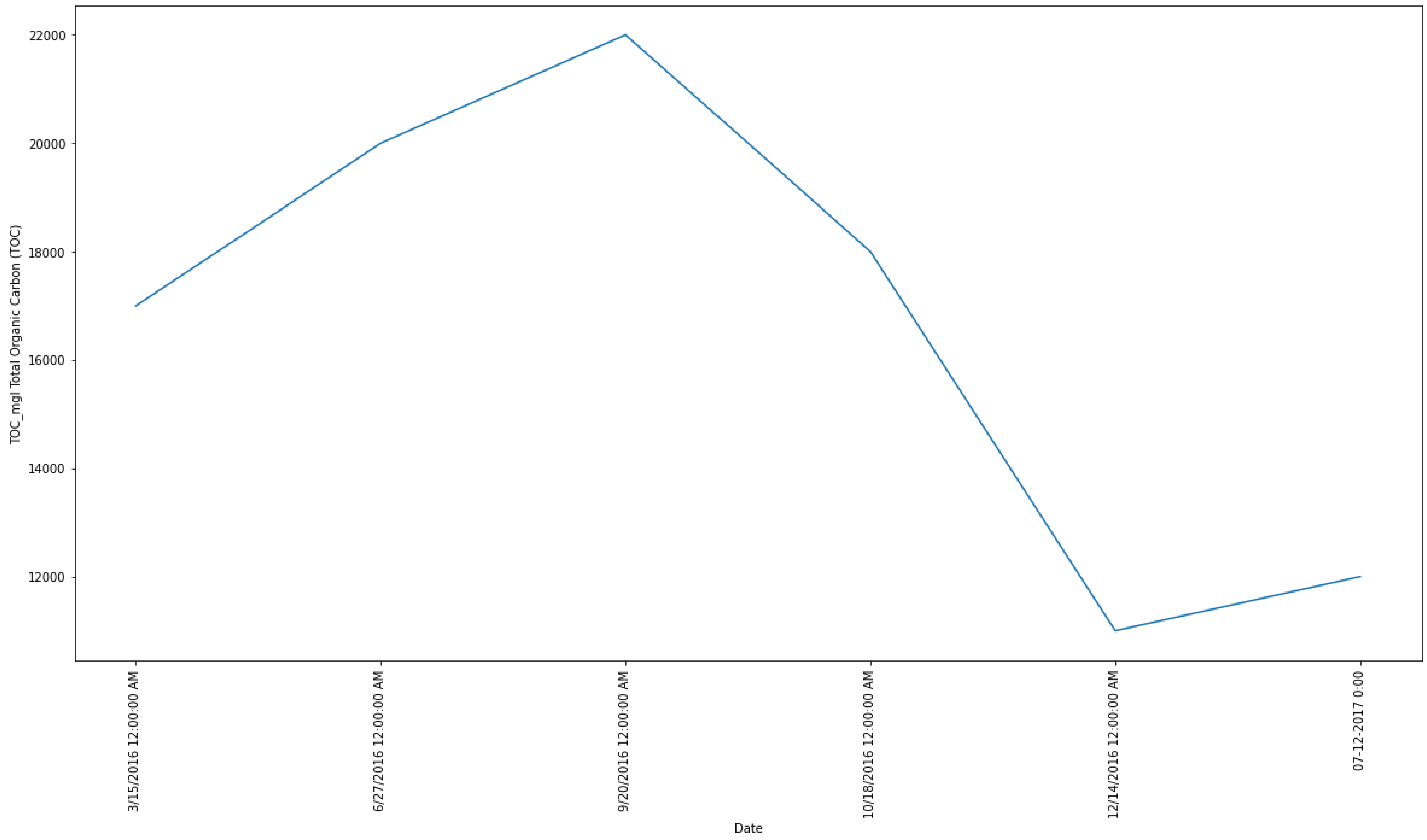
In [ ]:

```
df_new = fdf_values[fdf_values[col[29]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[29]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[29])
plt.show()
```



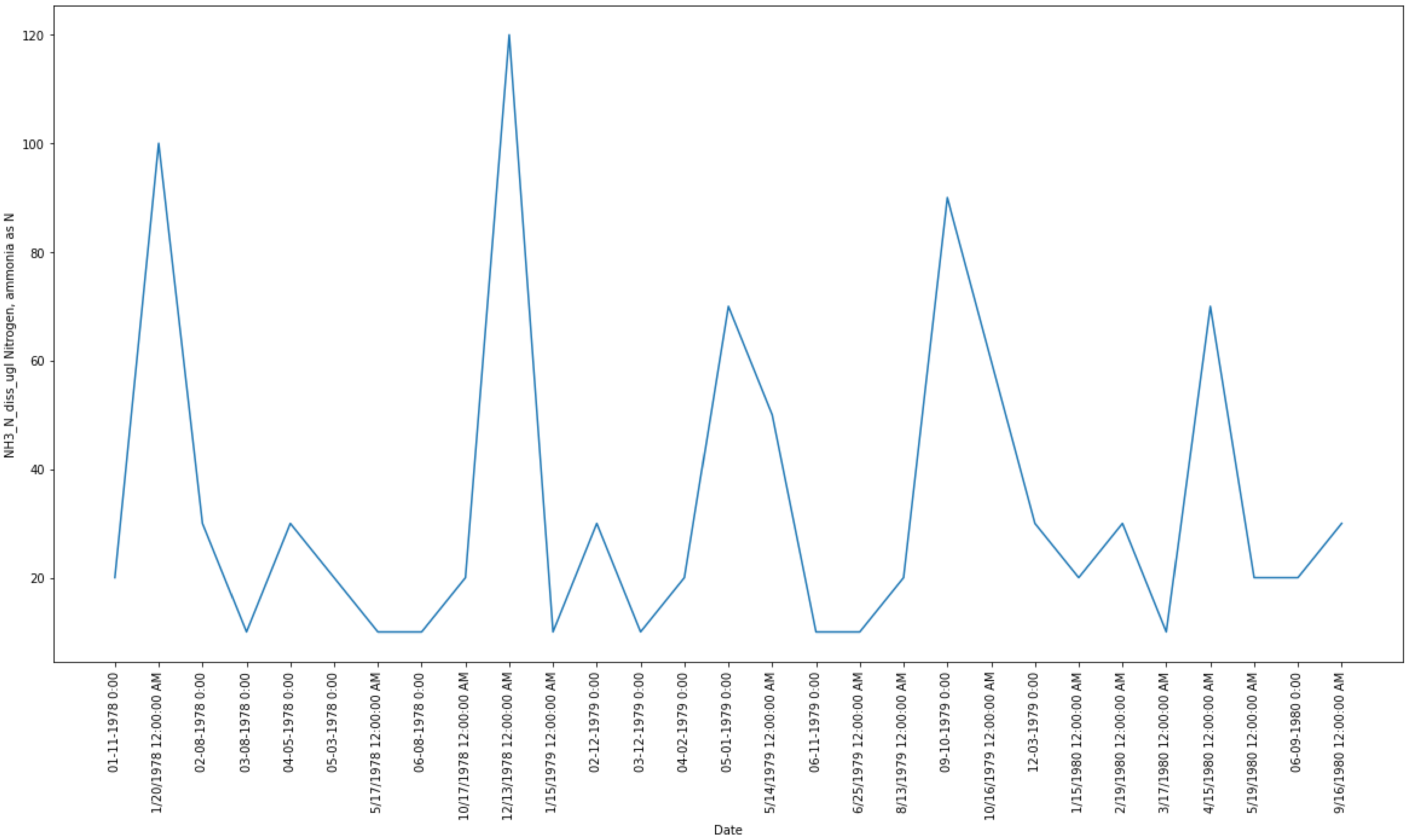
In [ ]:

```
df_new = fdf_values[fdf_values[col[30]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[30]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[30])
plt.show()
```



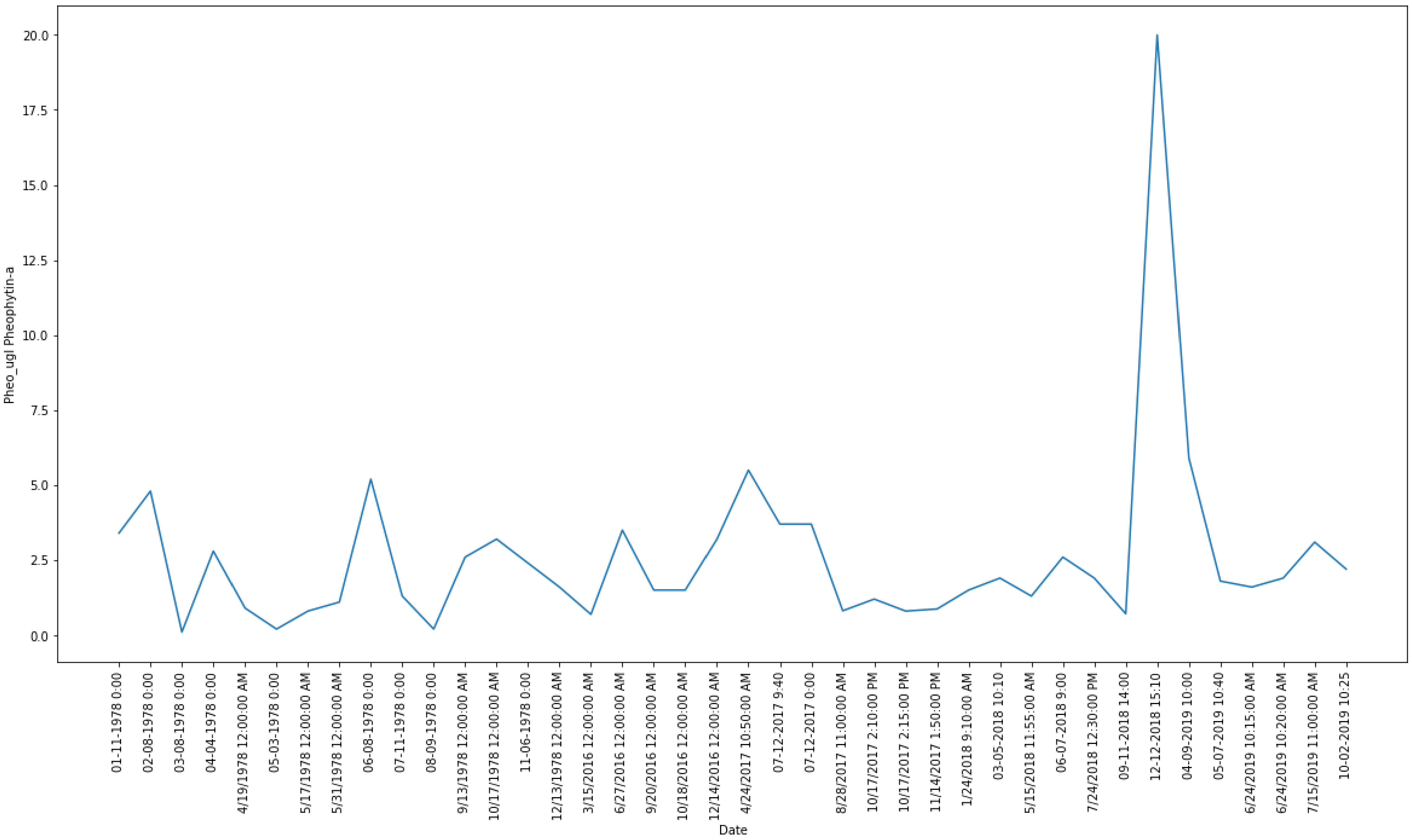
In [ ]:

```
df_new = fdf_values[fdf_values[col[31]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[31]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[31])
plt.show()
```



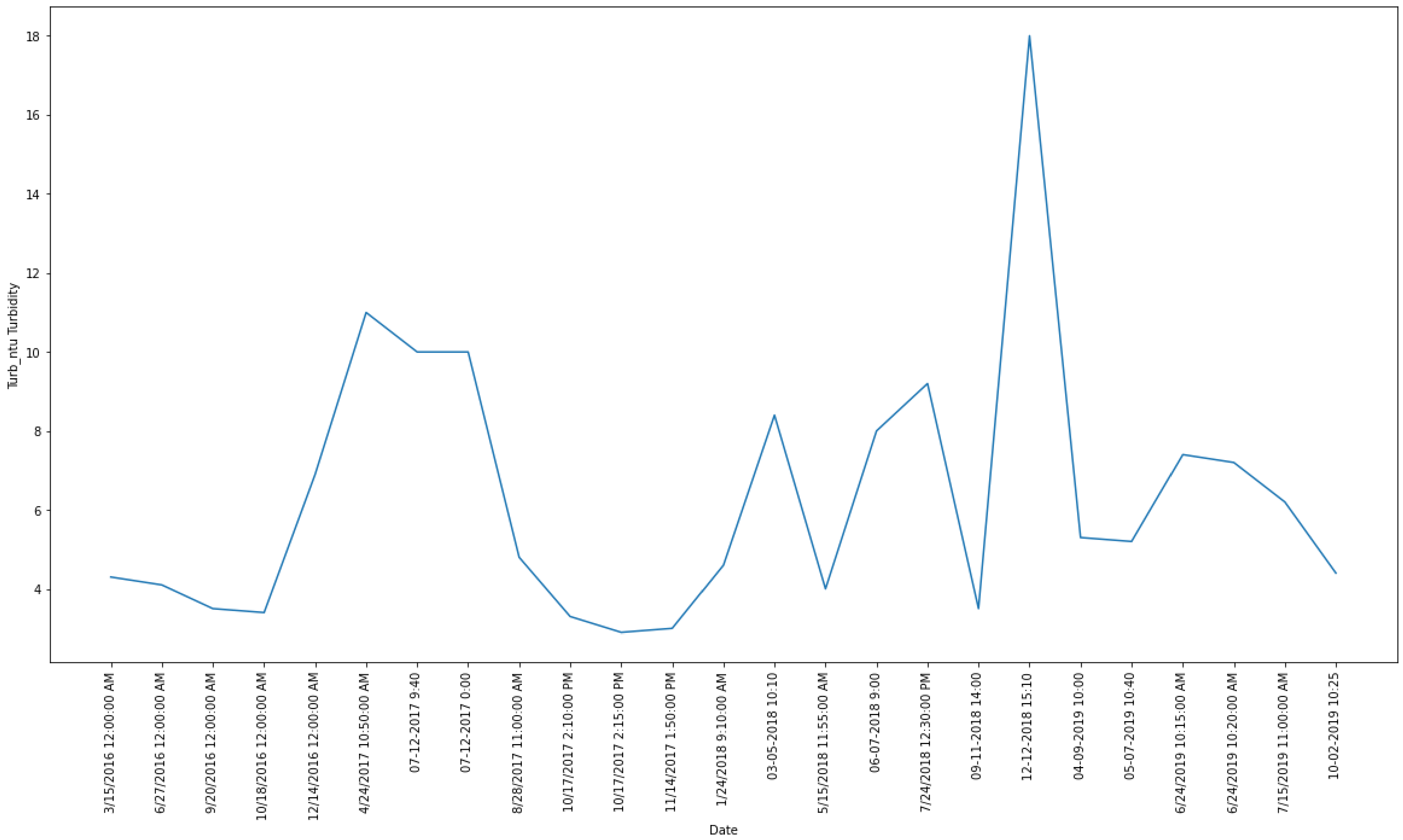
In [ ]:

```
df_new = fdf_values[fdf_values[col[32]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[32]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[32])
plt.show()
```



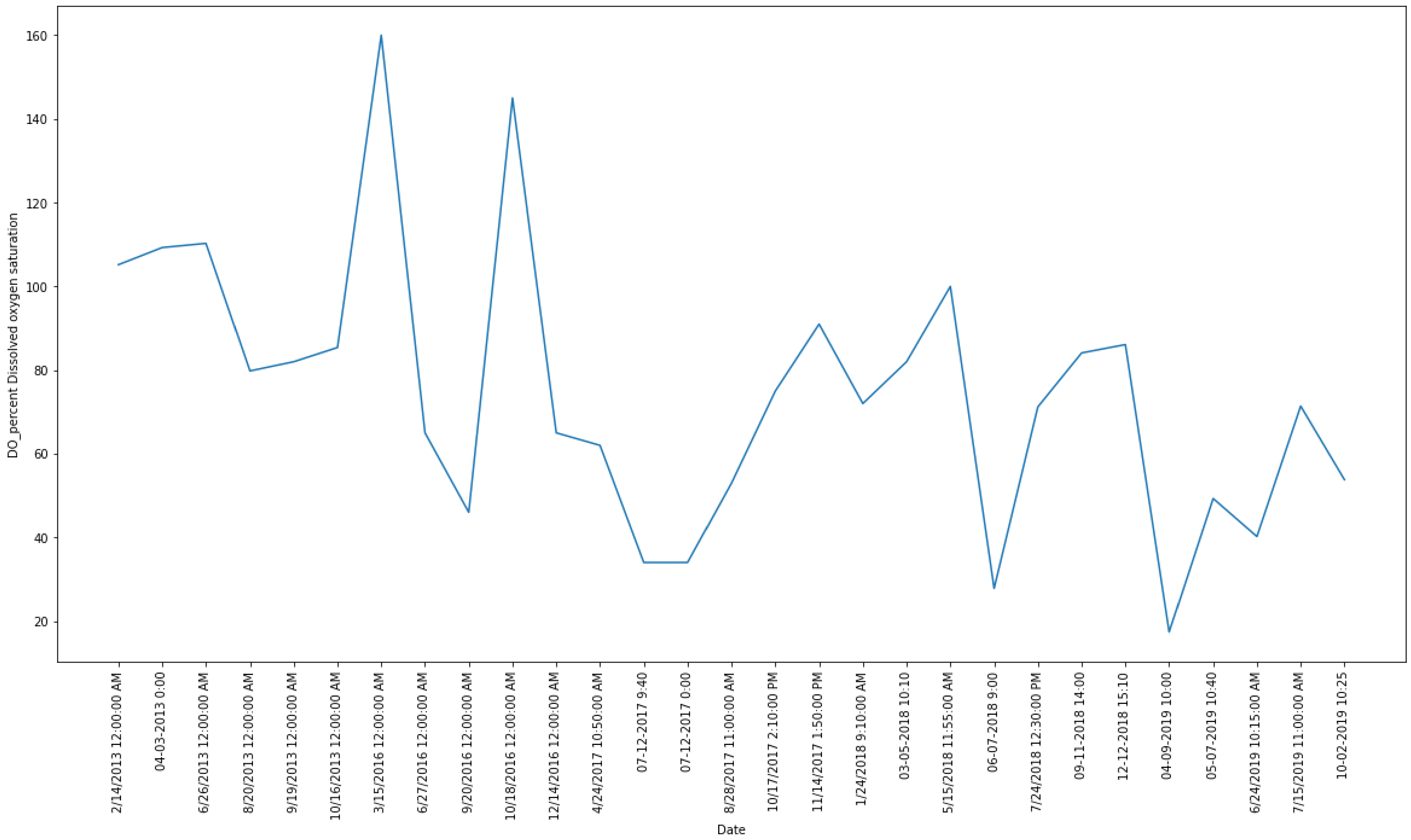
In [ ]:

```
df_new = fdf_values[fdf_values[col[33]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[33]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[33])
plt.show()
```



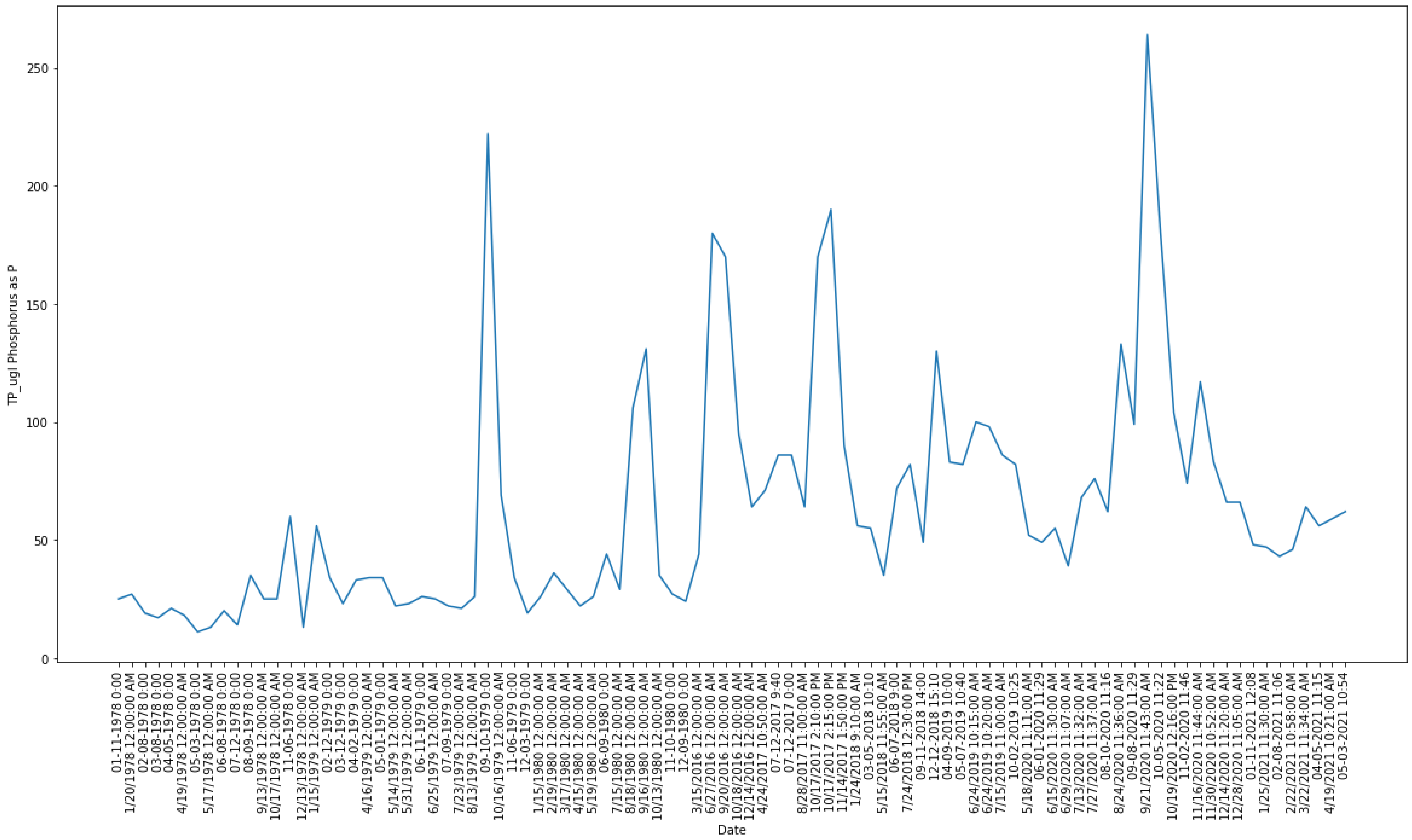
In [ ]:

```
val = 34
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



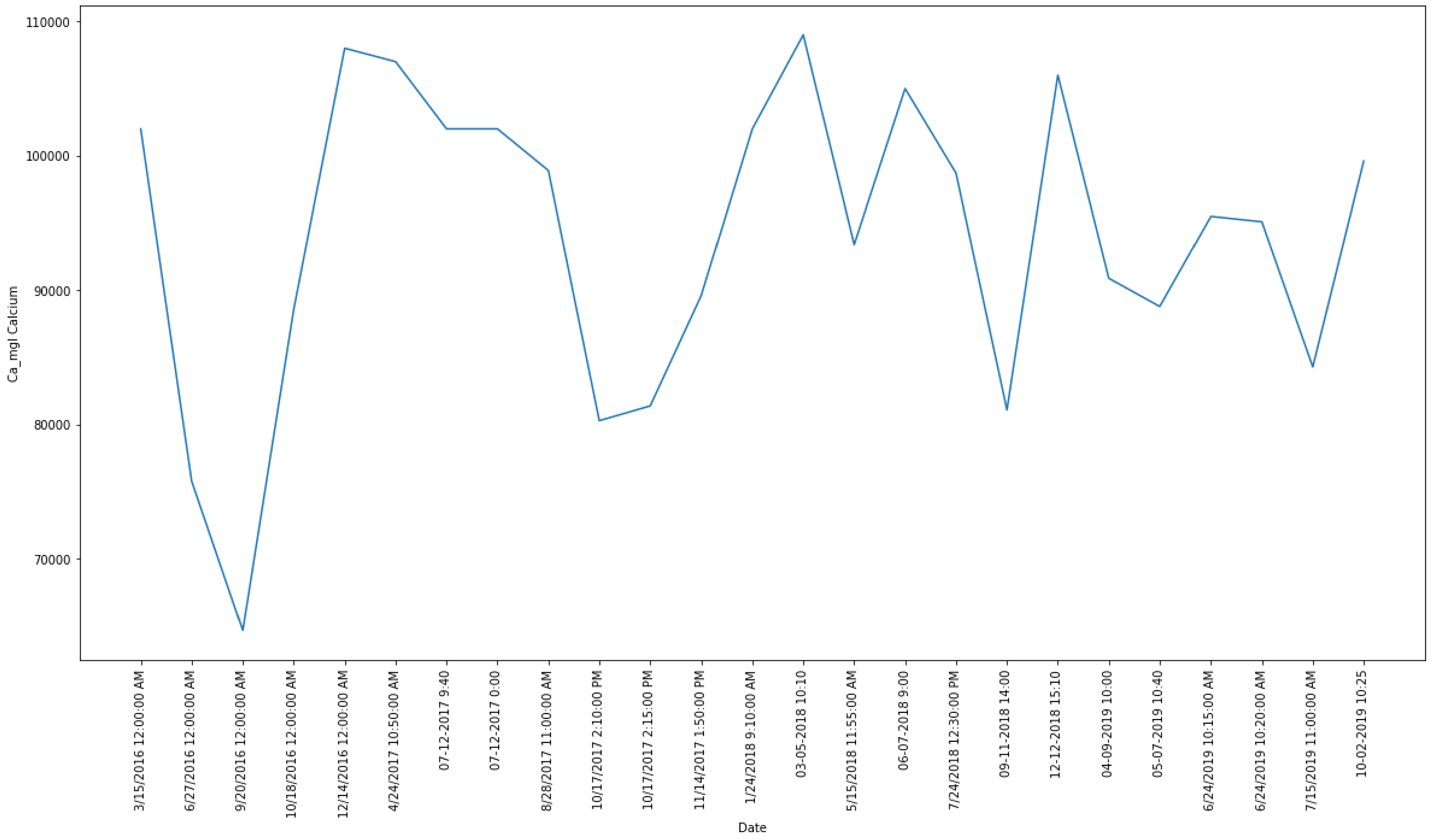
In [ ]:

```
val = 35
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



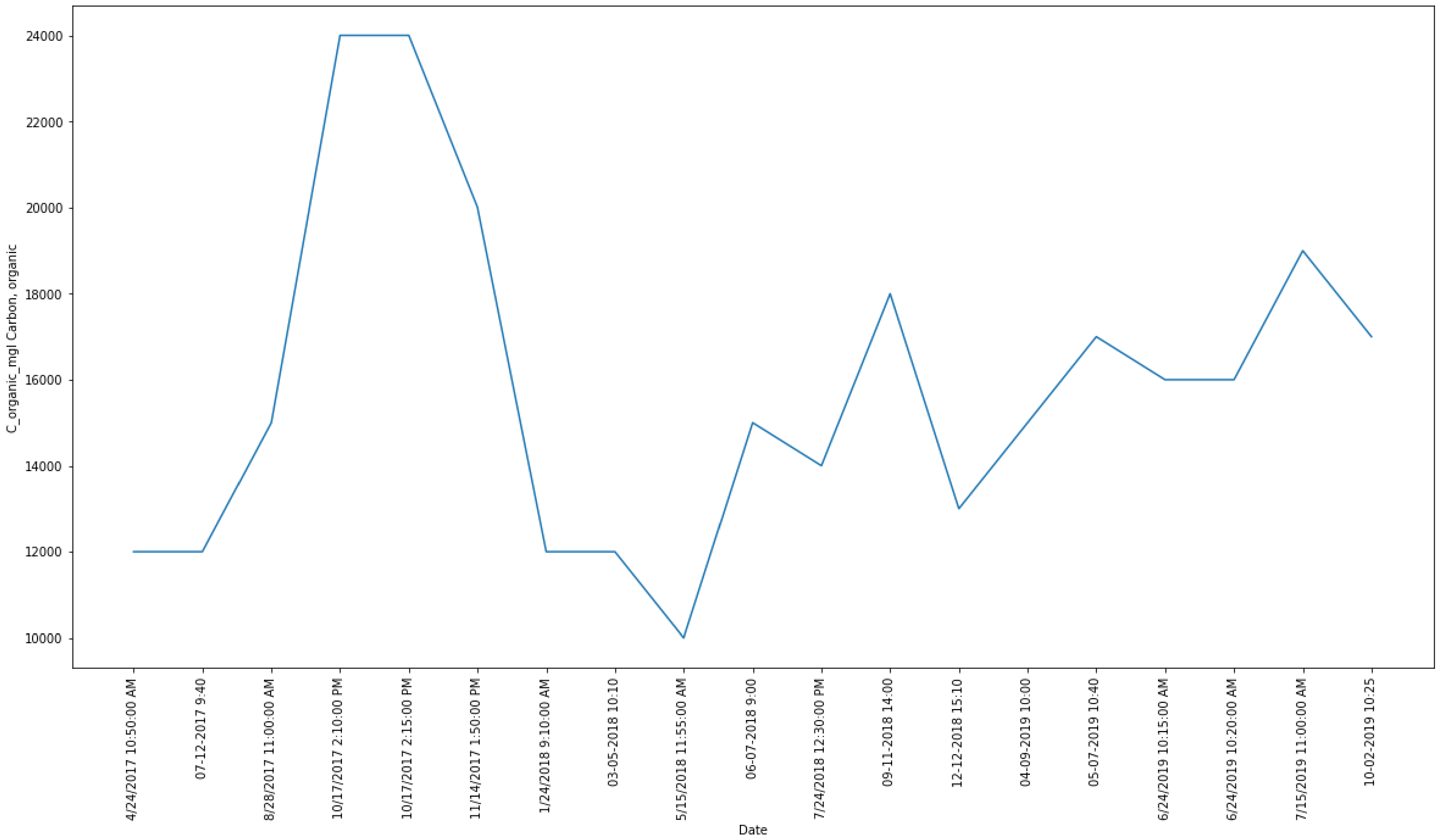
In [ ]:

```
val = 36
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



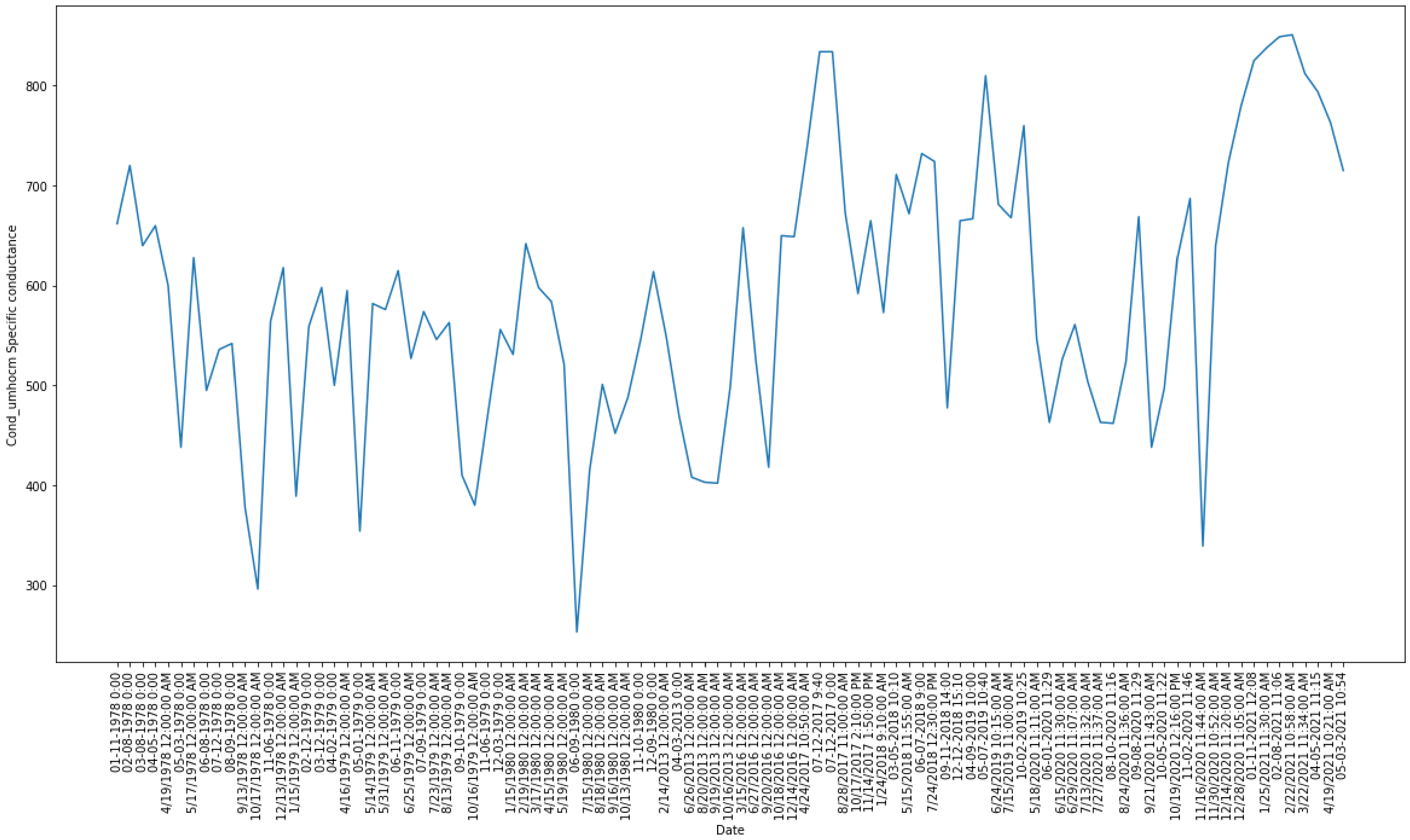
In [ ]:

```
val = 37
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



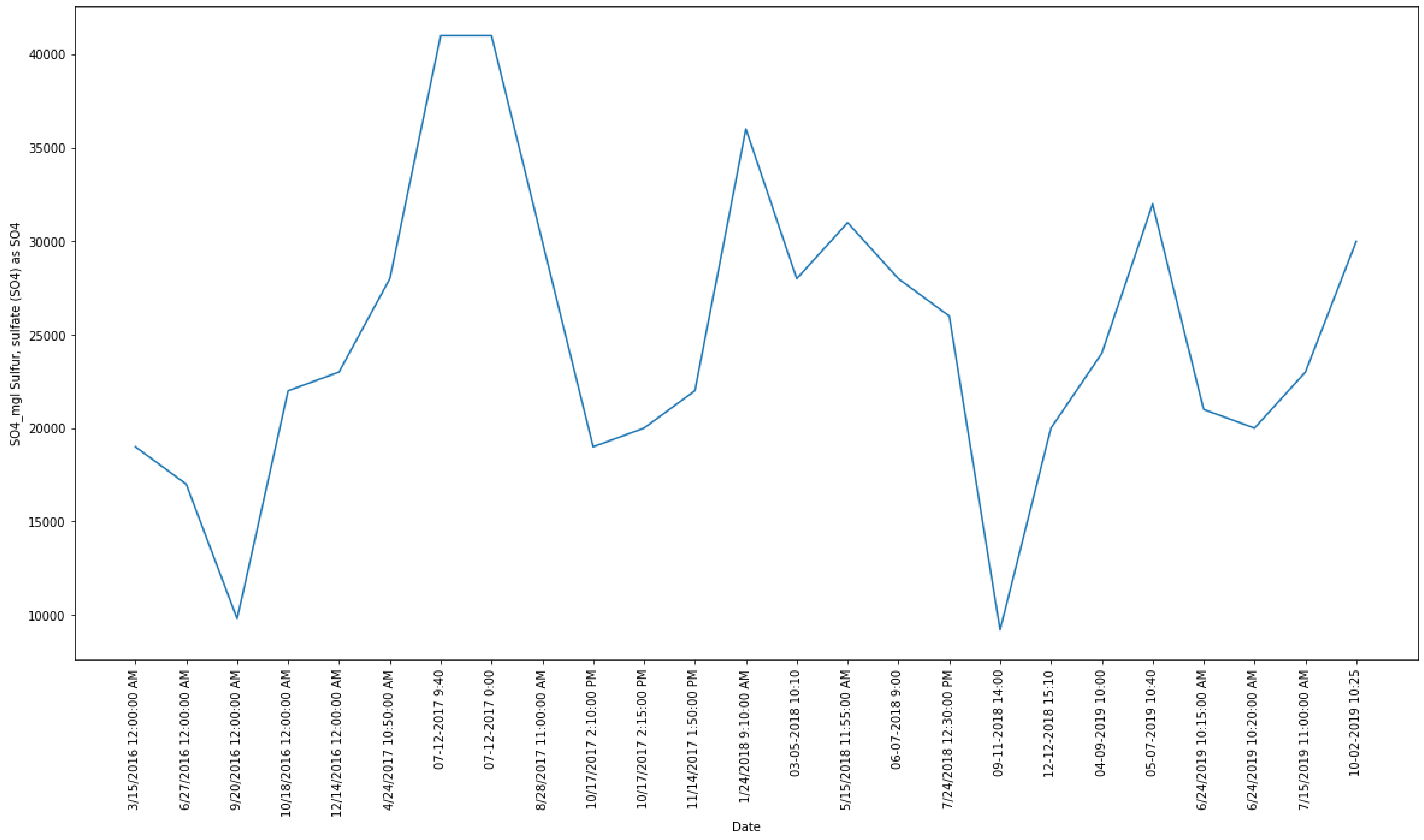
In [ ]:

```
val = 38
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



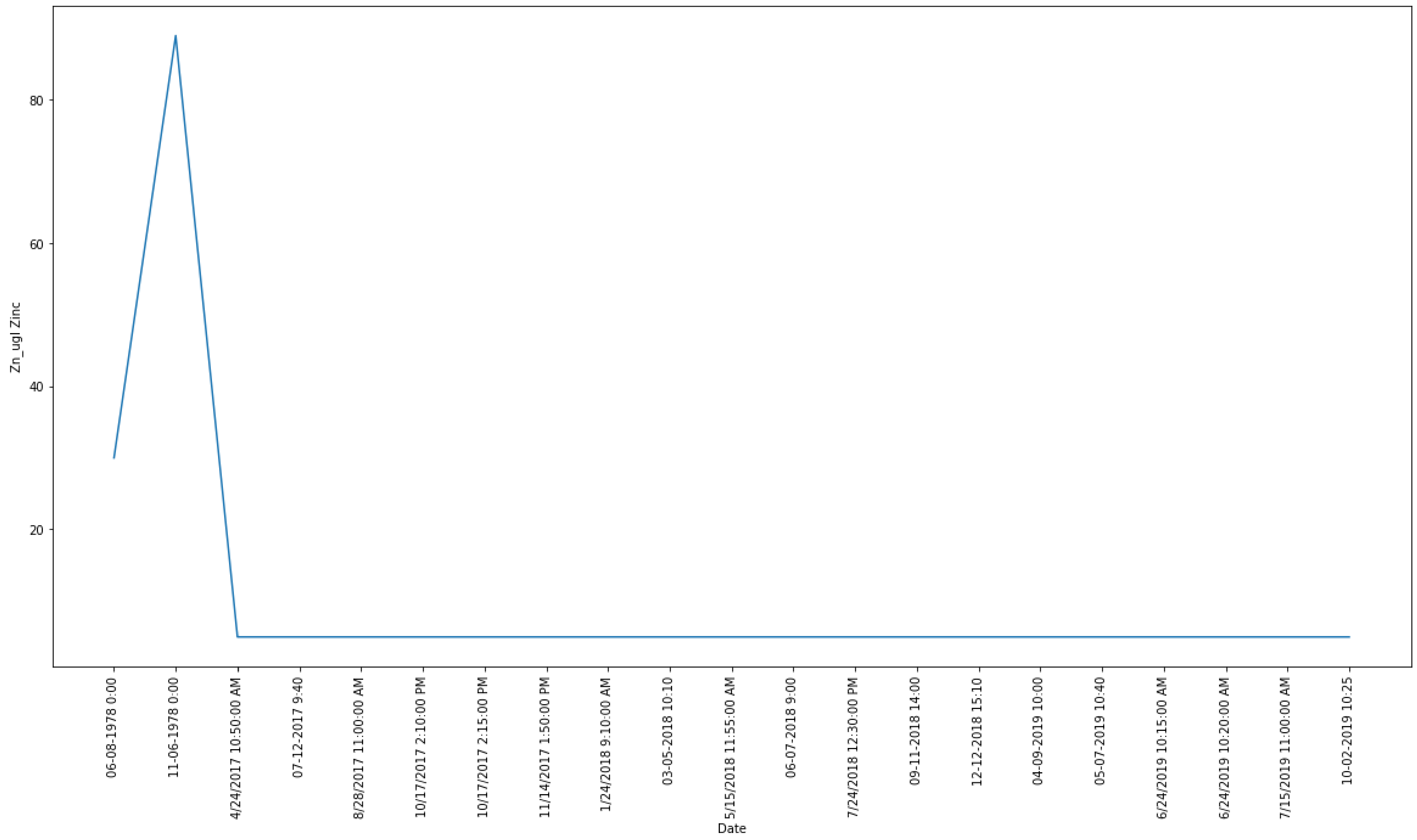
In [ ]:

```
val = 39
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



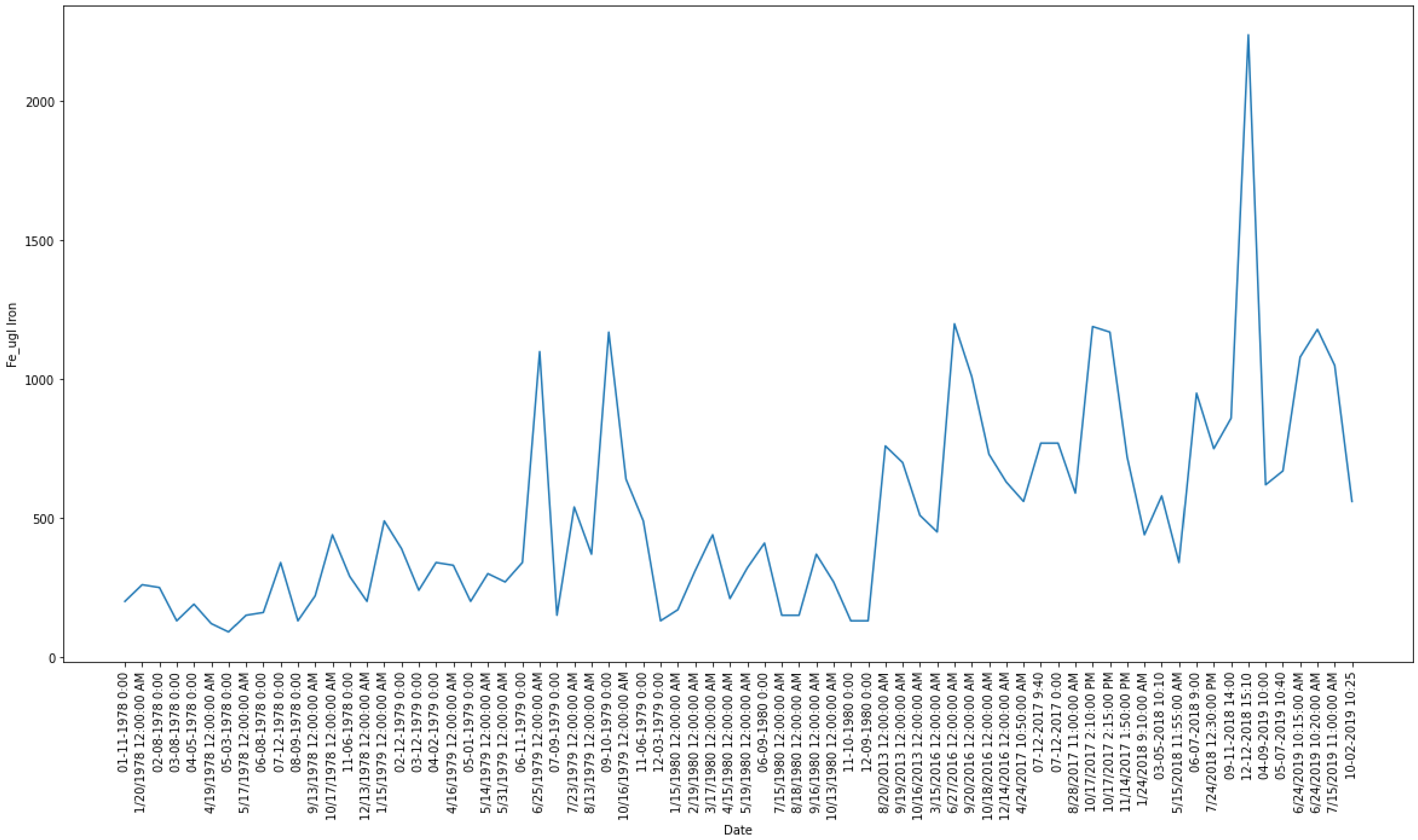
In [ ]:

```
val = 40
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



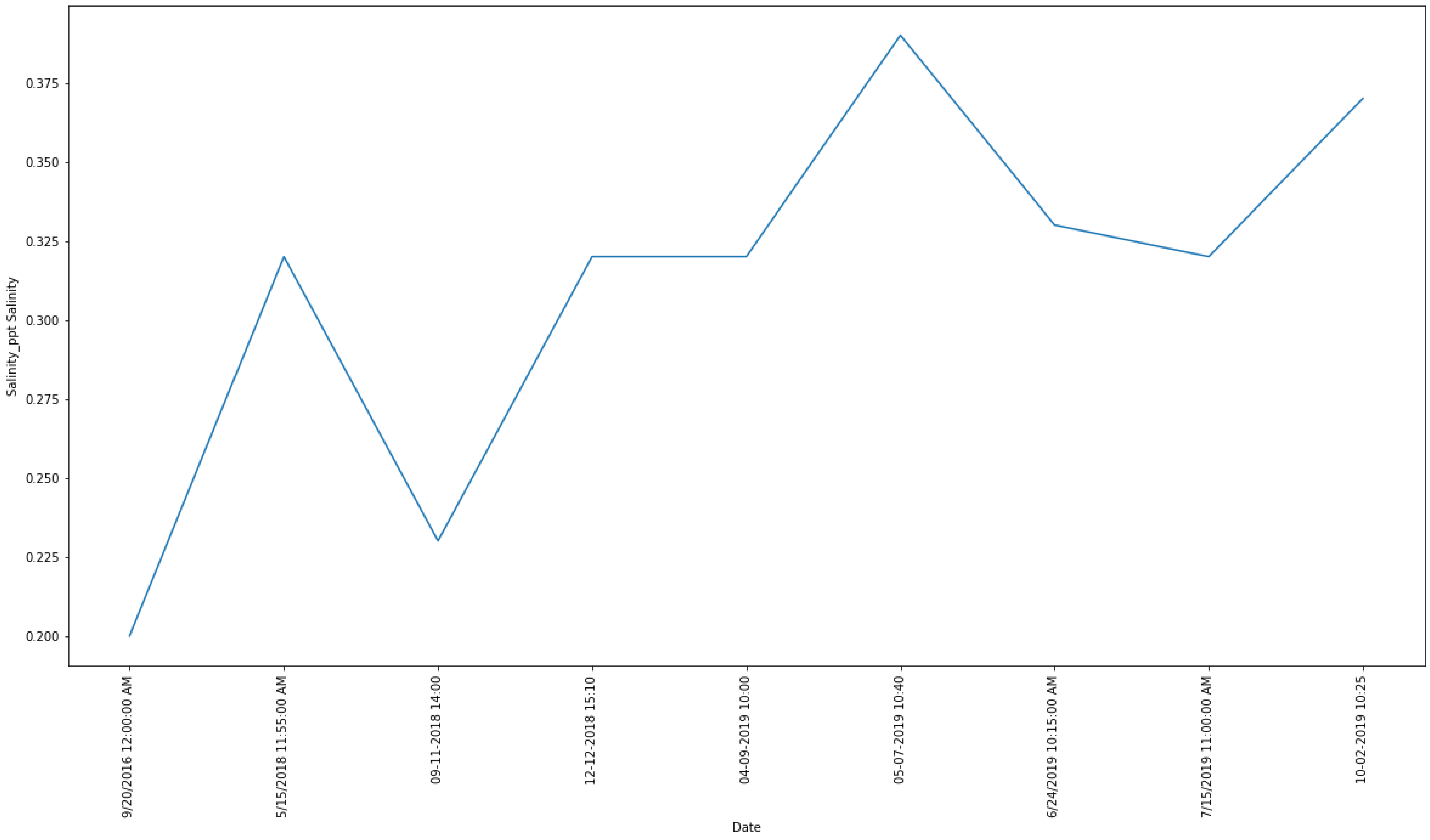
In [ ]:

```
val = 41
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



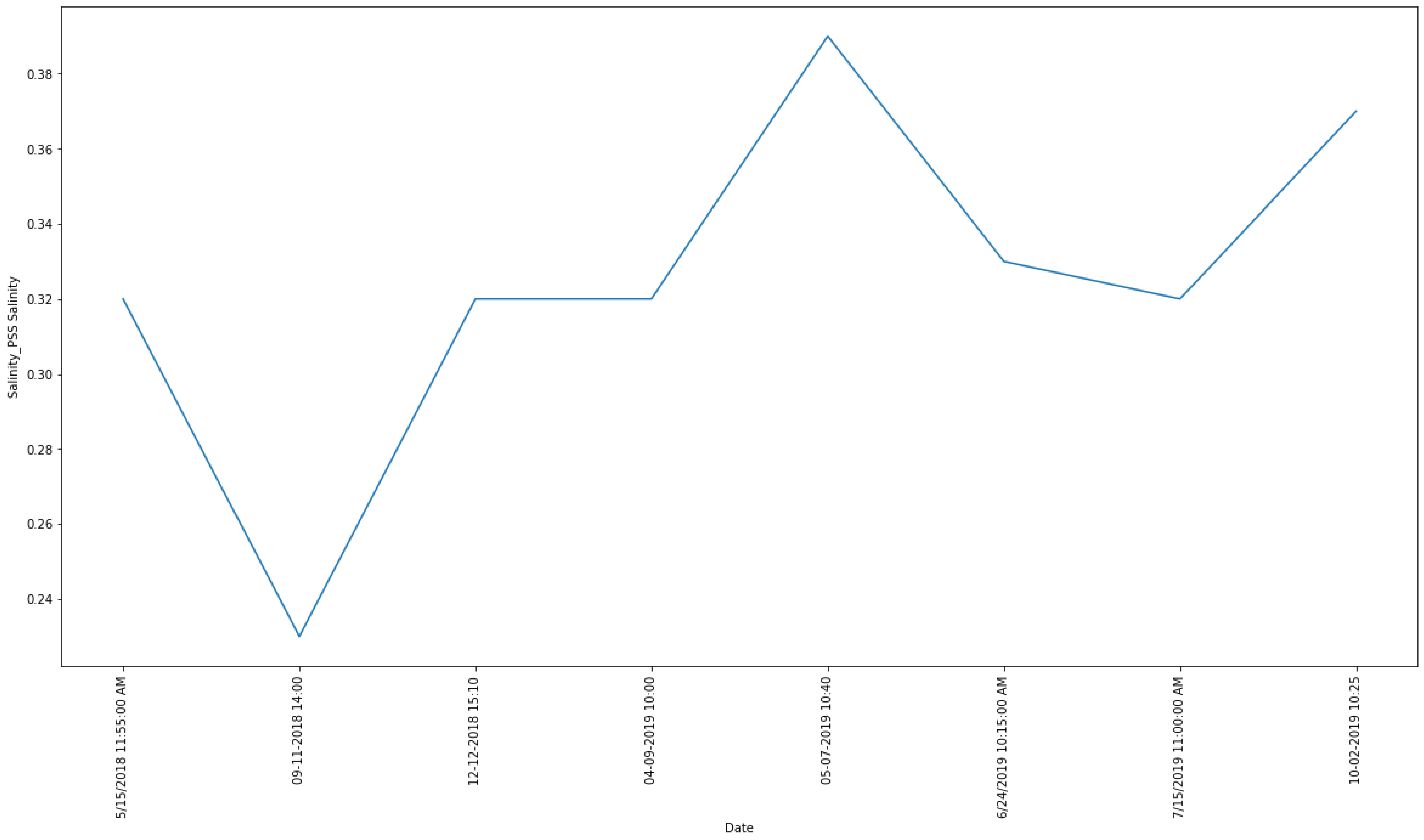
In [ ]:

```
val = 42
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```

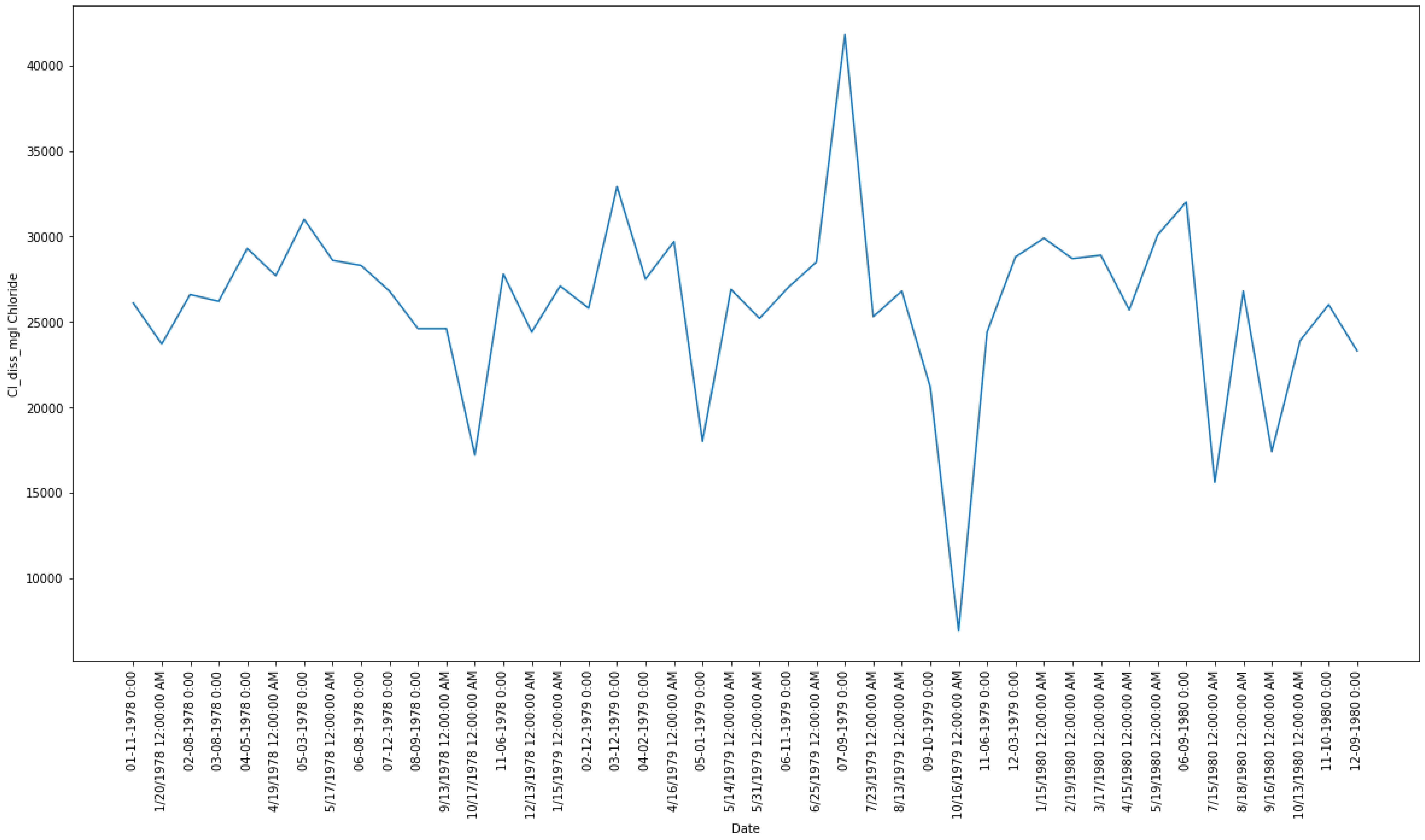


In [ ]:

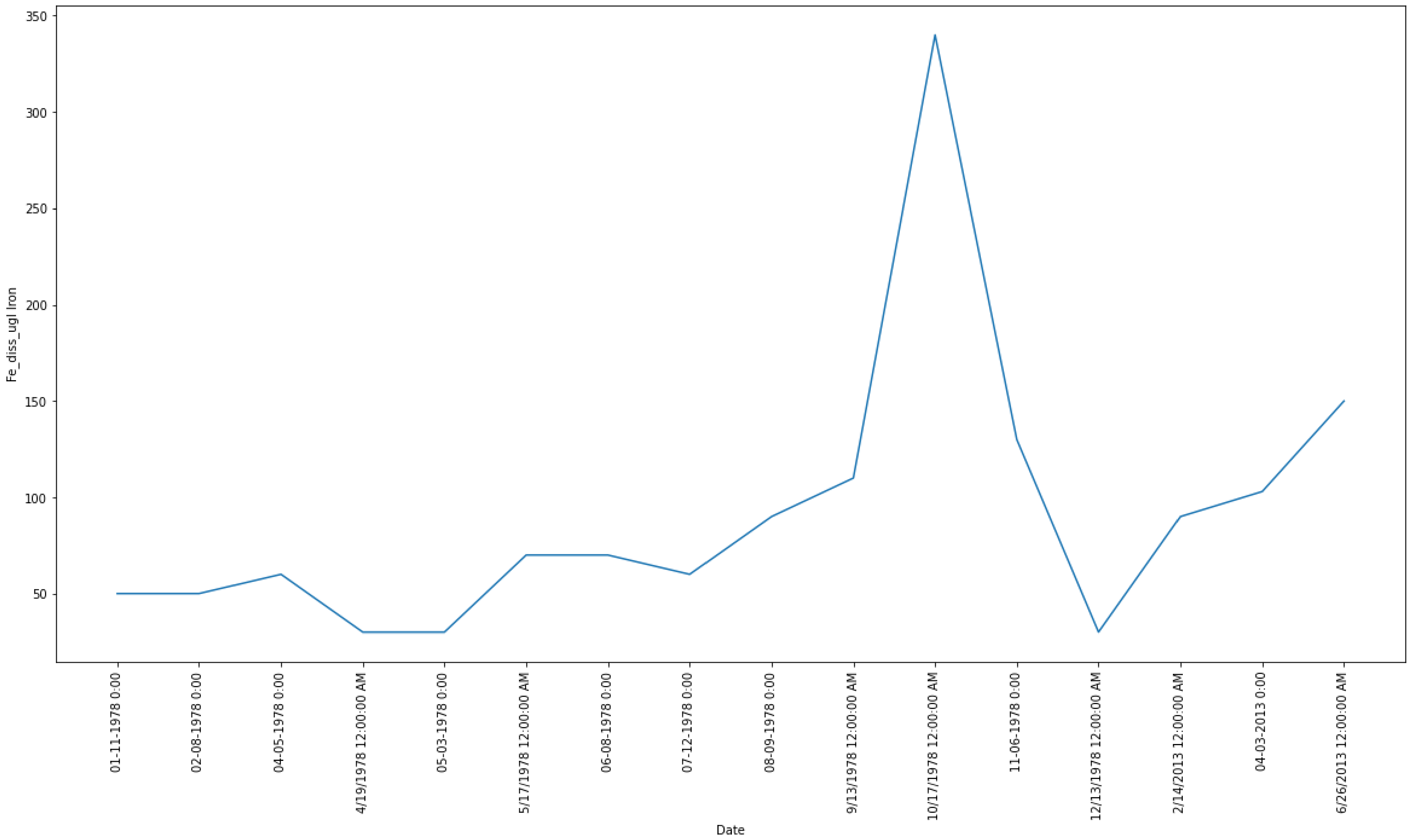
```
val = 43
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



```
In [ ]:  val = 44
          df_new = fdf_values[fdf_values[col[val]]!='na']
          plt.figure(figsize=(20,10))
          plt.plot(df_new['Date'], df_new[col[val]])
          plt.xticks(rotation = 90)
          plt.xlabel("Date")
          plt.ylabel(col[val])
          plt.show()
```

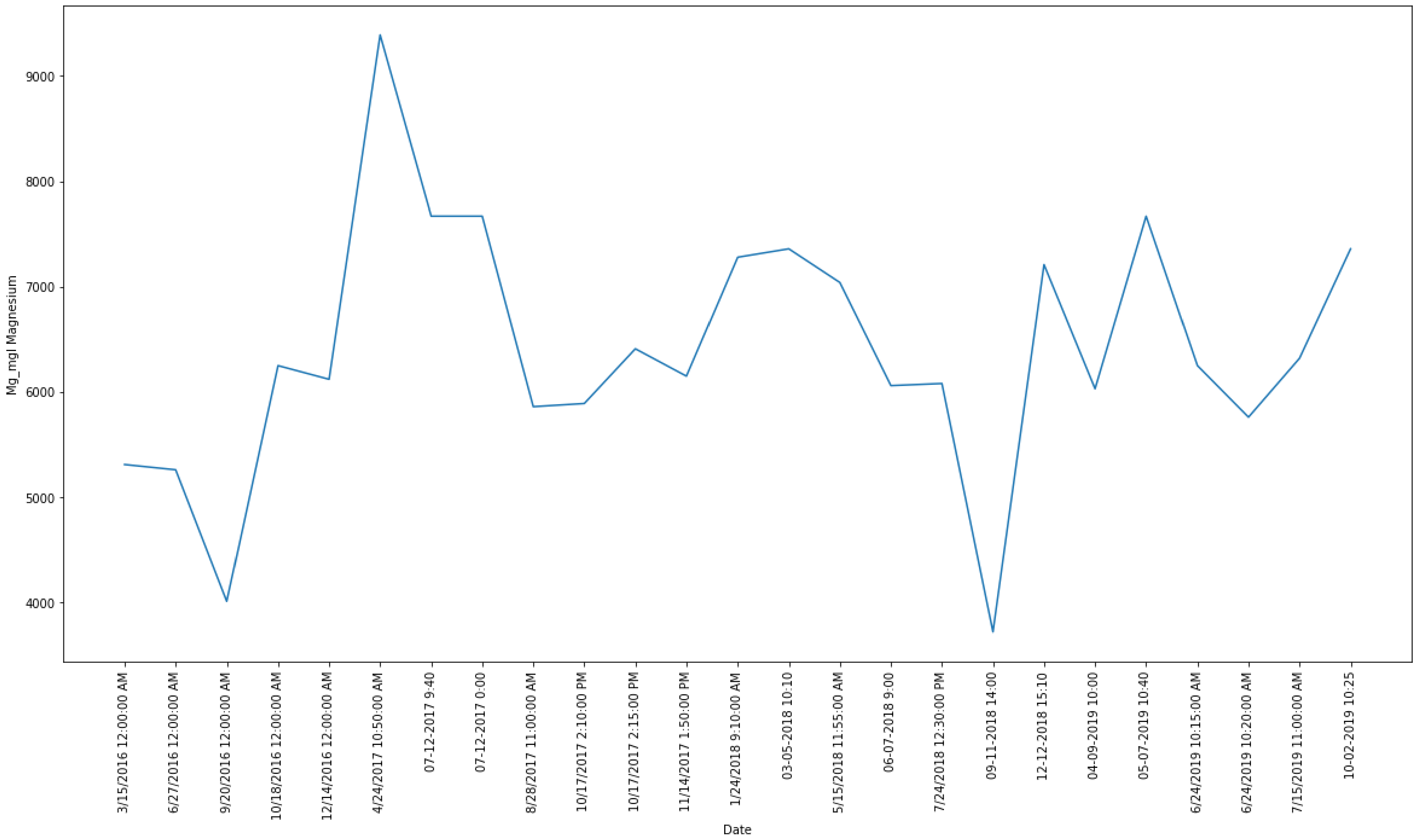


```
In [ ]:  val = 45
          df_new = fdf_values[fdf_values[col[val]]!='na']
          plt.figure(figsize=(20,10))
          plt.plot(df_new['Date'], df_new[col[val]])
          plt.xticks(rotation = 90)
          plt.xlabel("Date")
          plt.ylabel(col[val])
          plt.show()
```



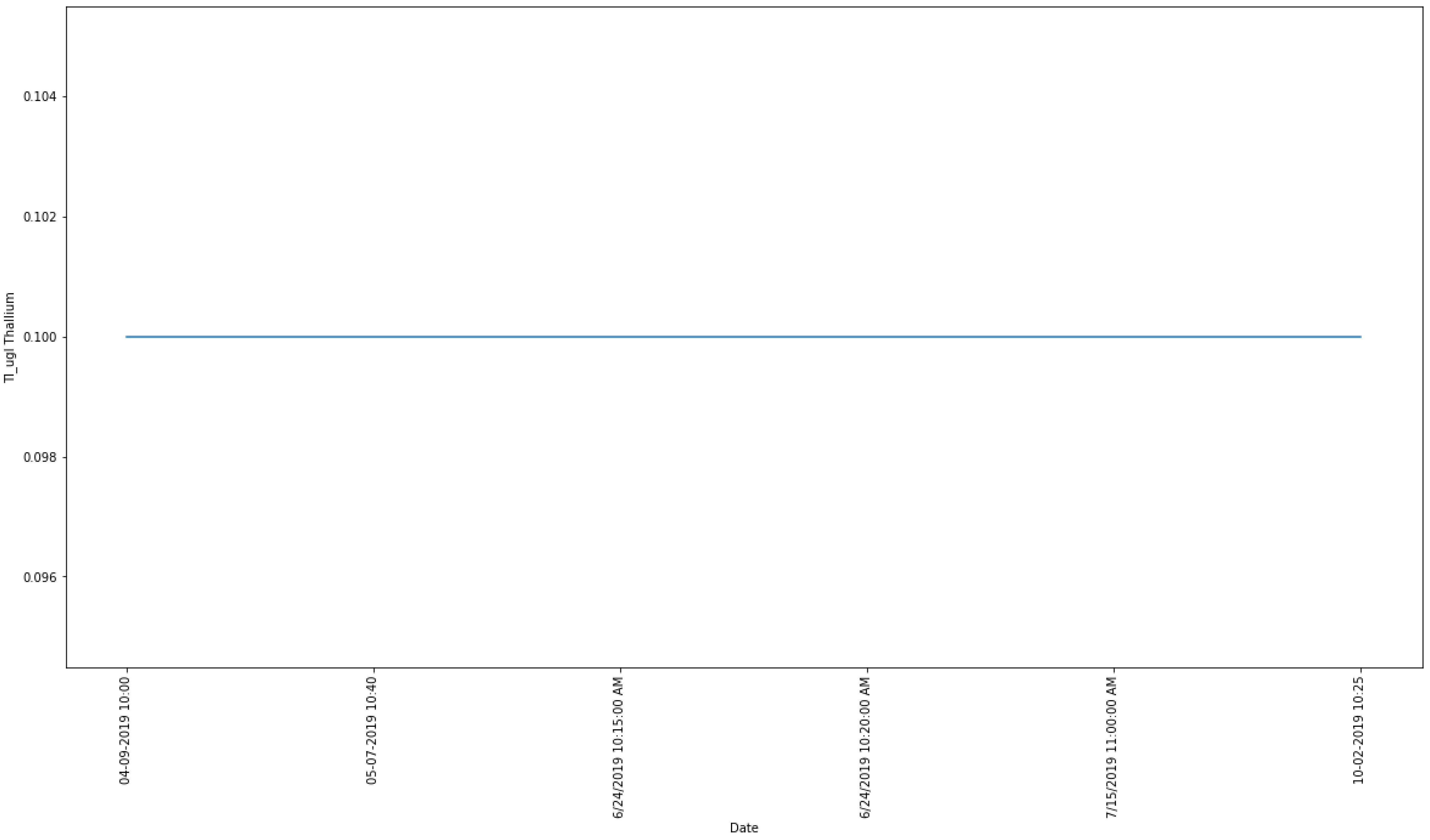
In [ ]:

```
val = 46
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



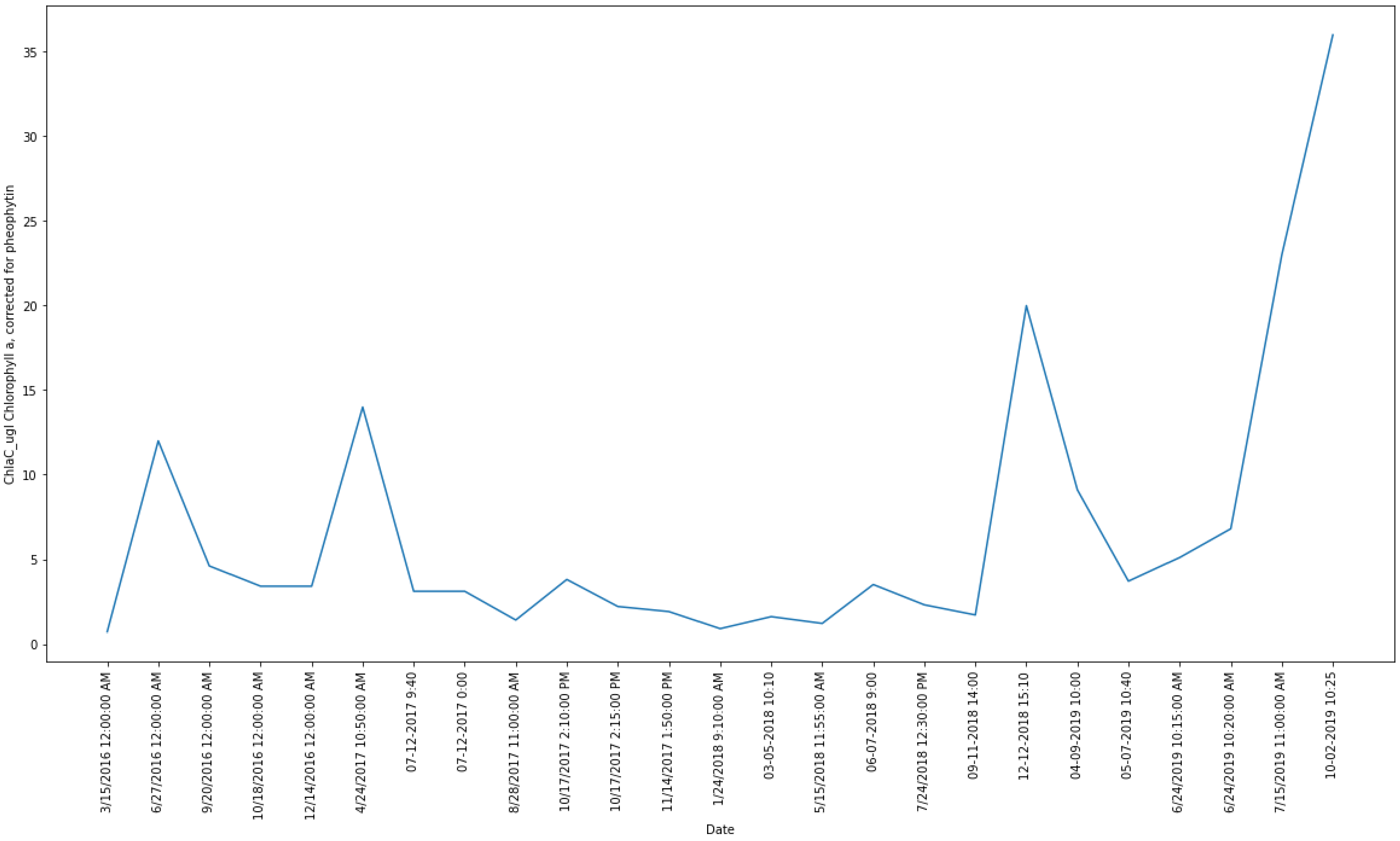
In [ ]:

```
val = 47
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



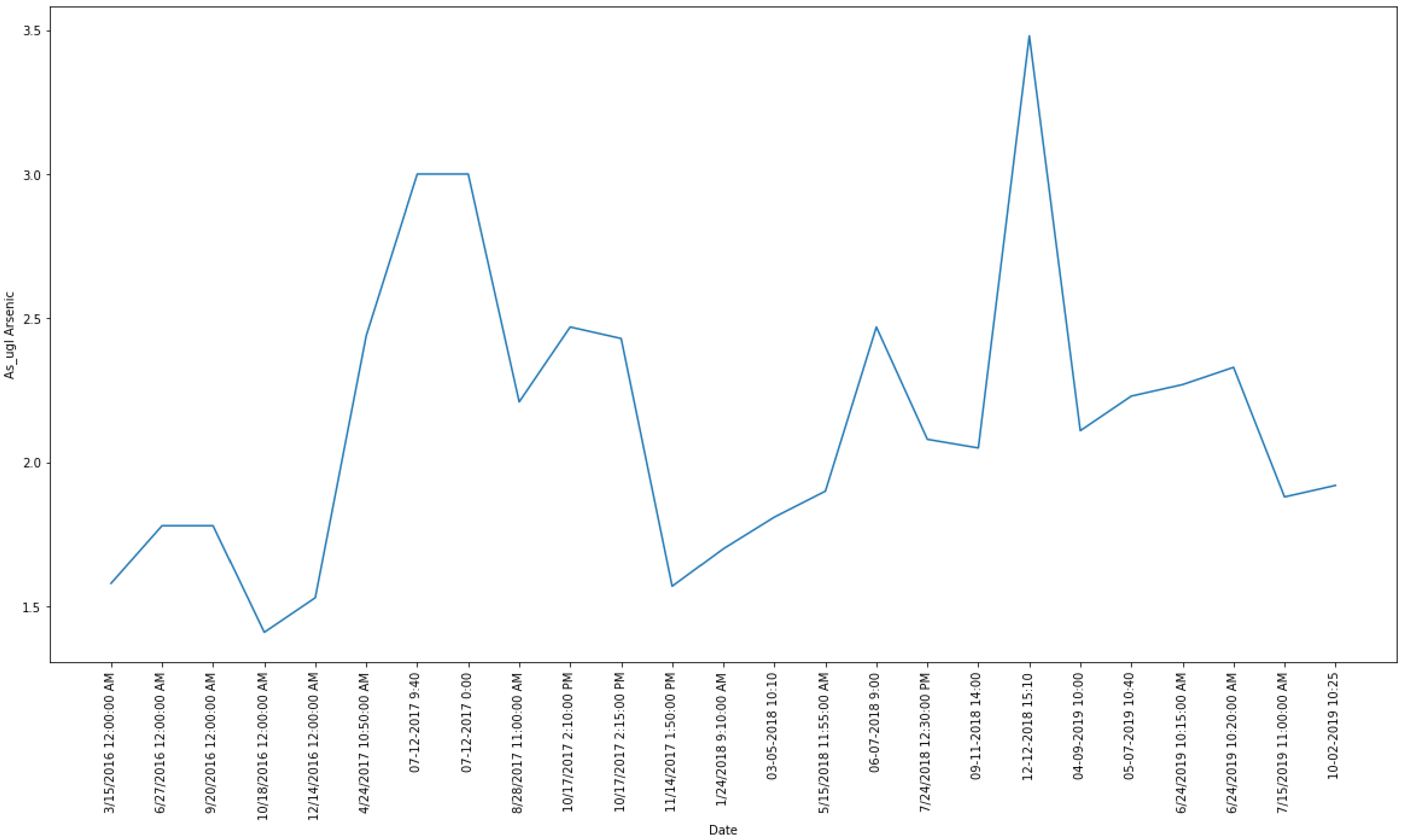
In [ ]:

```
val = 48
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



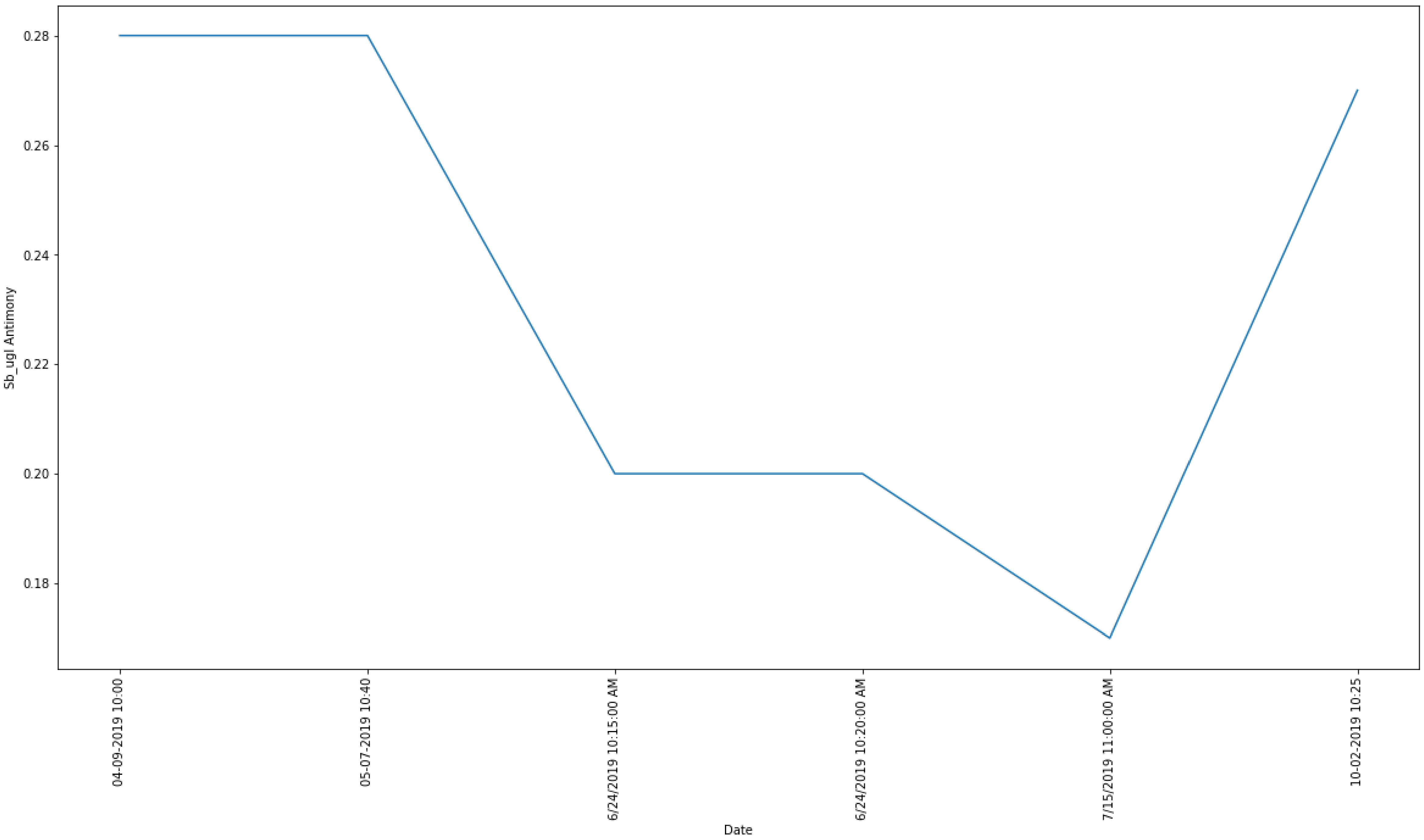
In [ ]:

```
val = 49
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



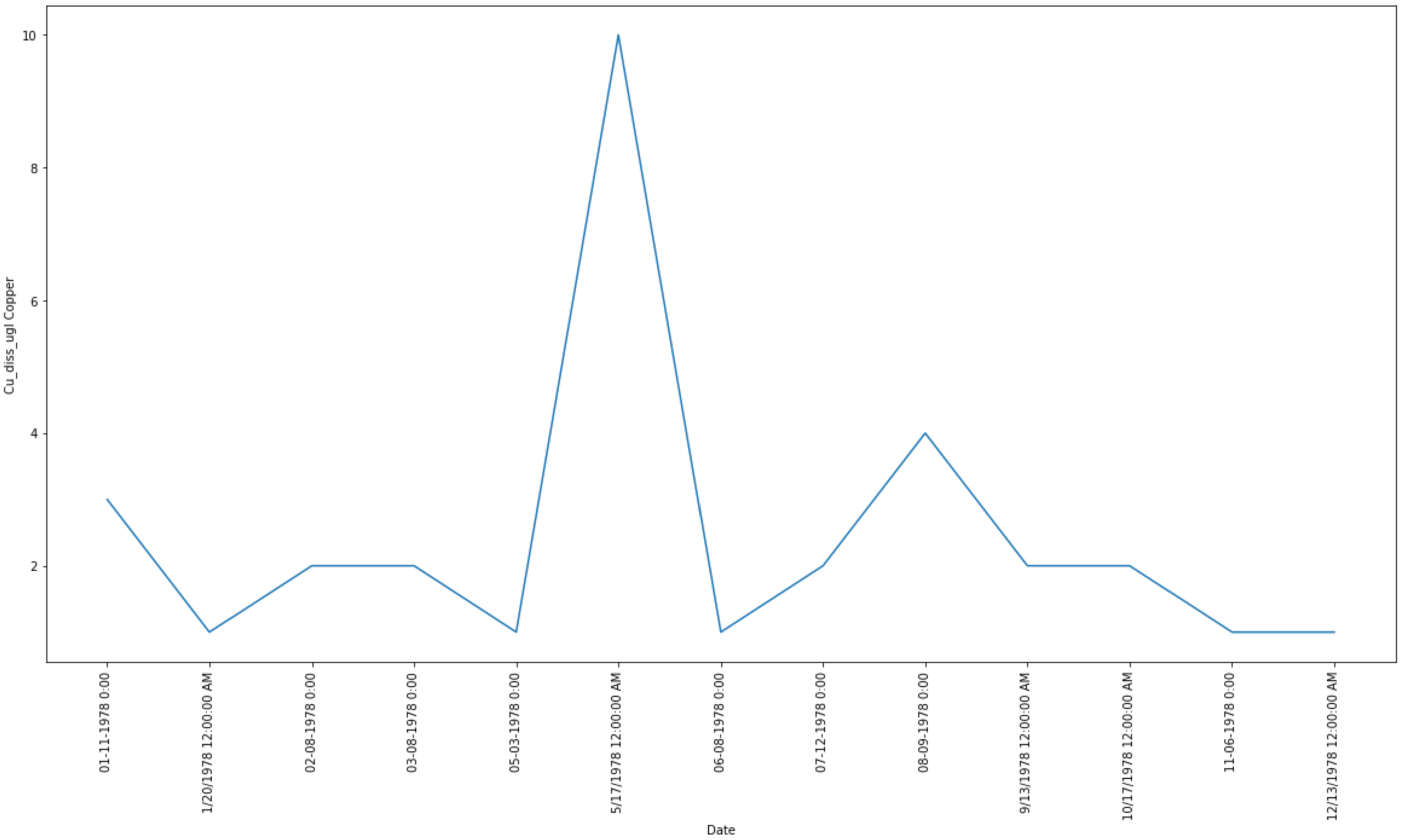
In [ ]:

```
val = 50
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



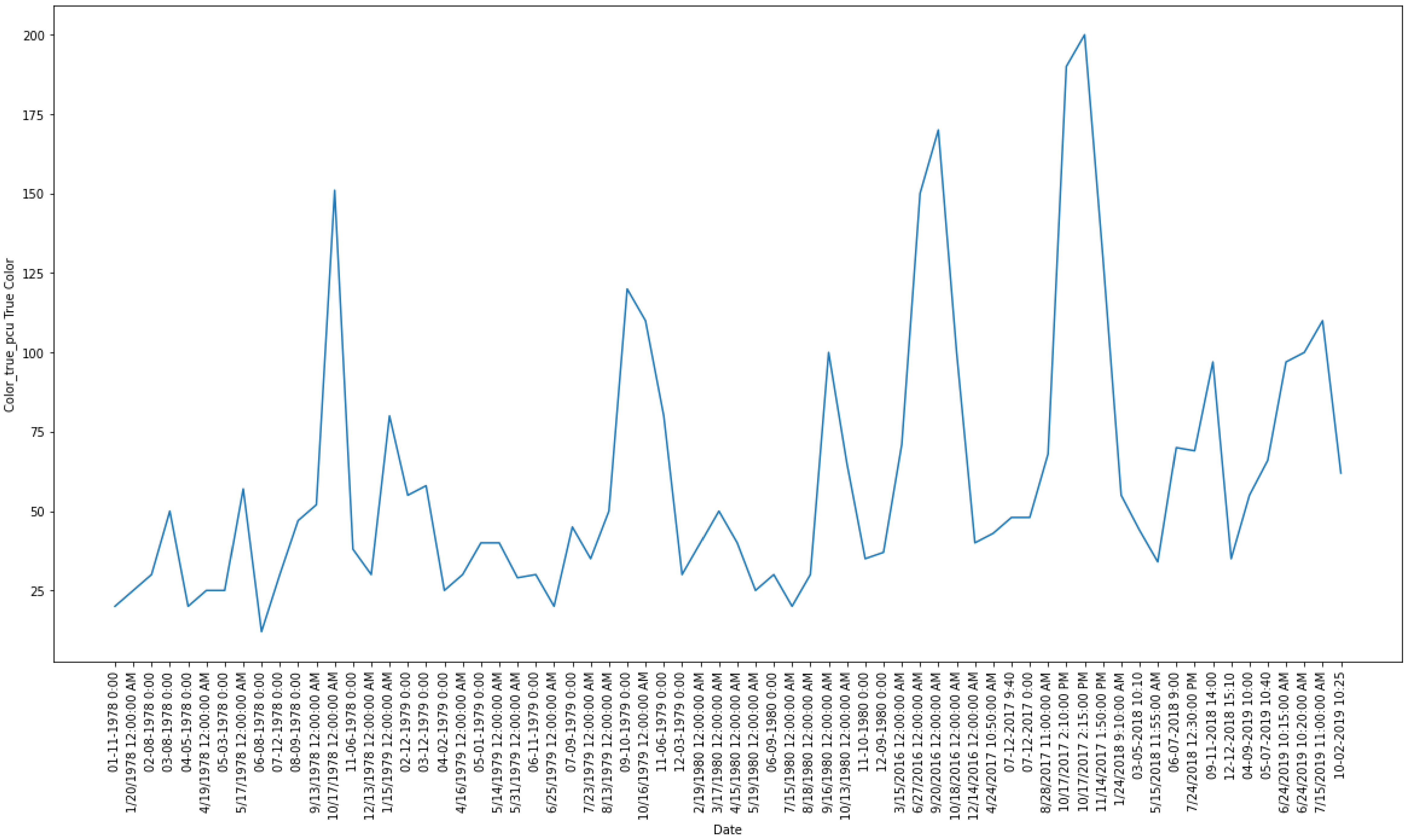
In [ ]:

```
val = 51
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



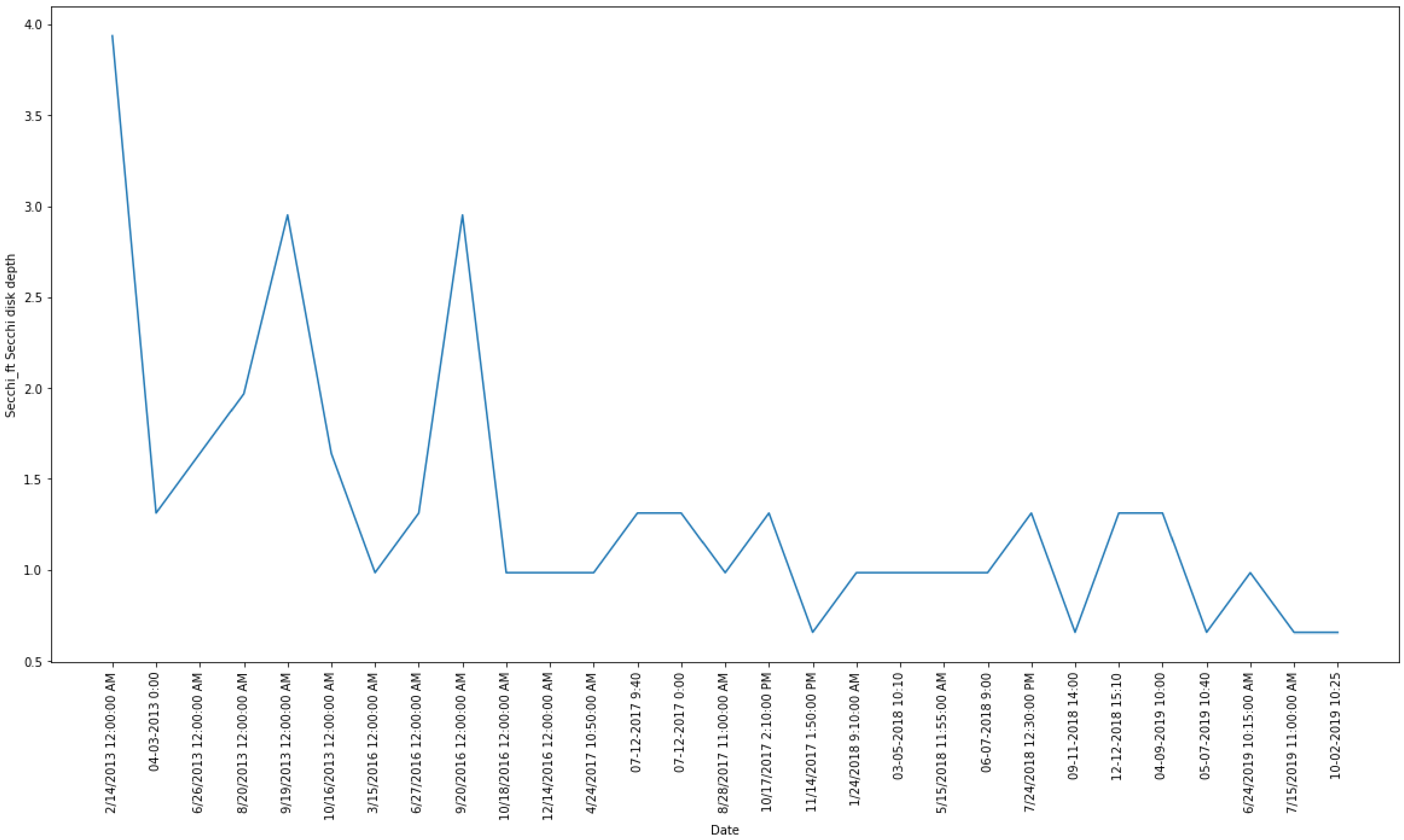
In [ ]:

```
val = 52
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



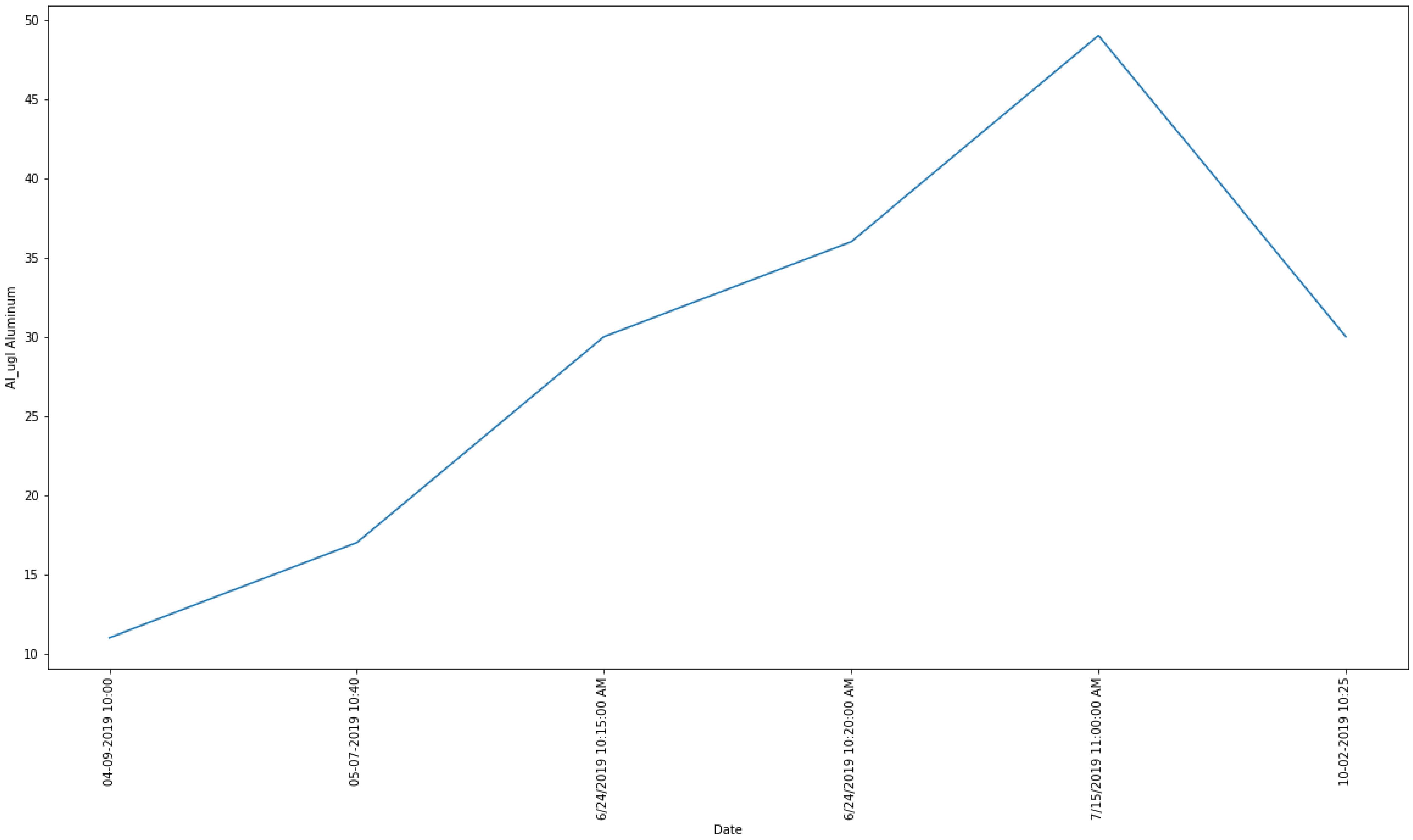
In [ ]:

```
val = 53
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



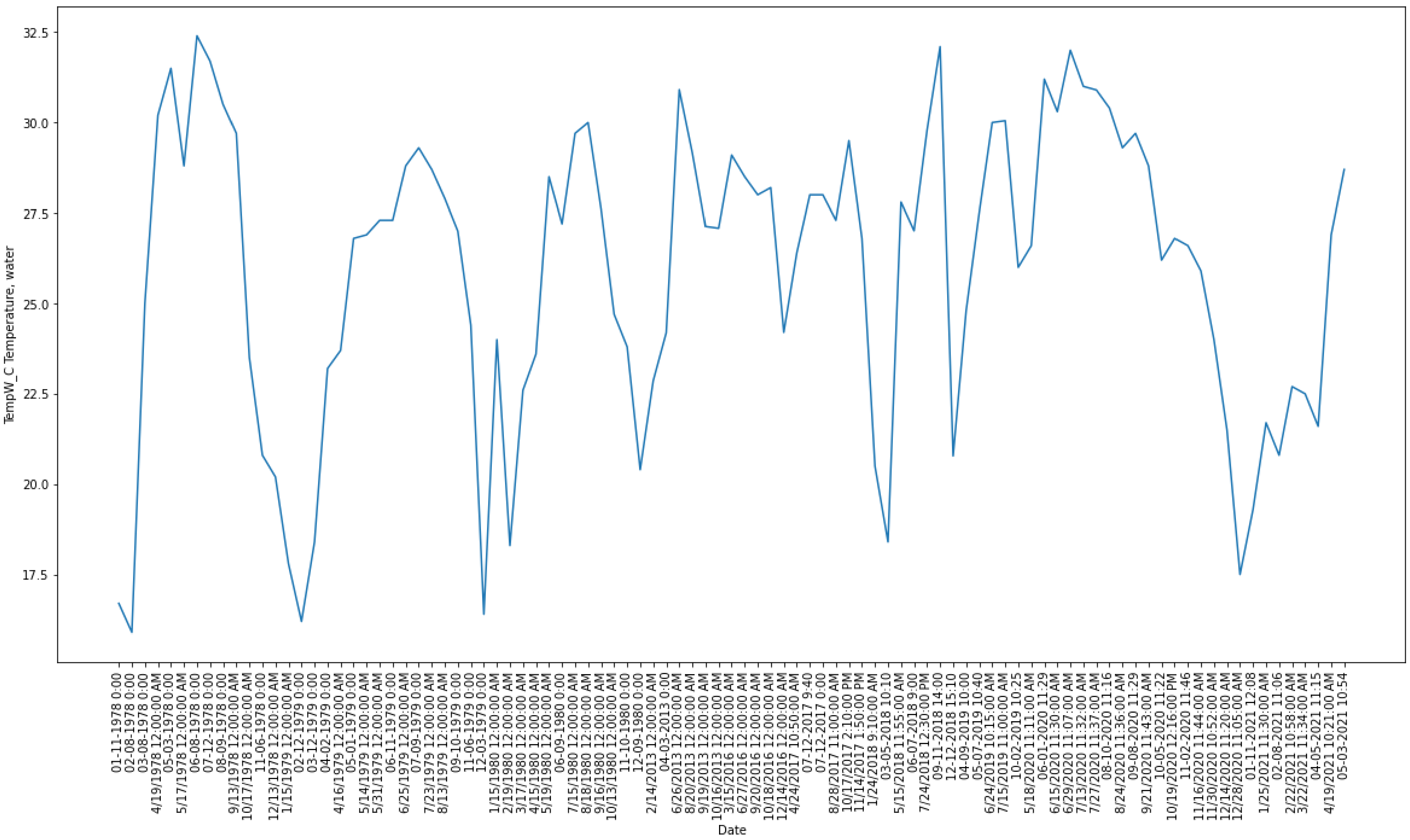
In [ ]:

```
val = 54
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



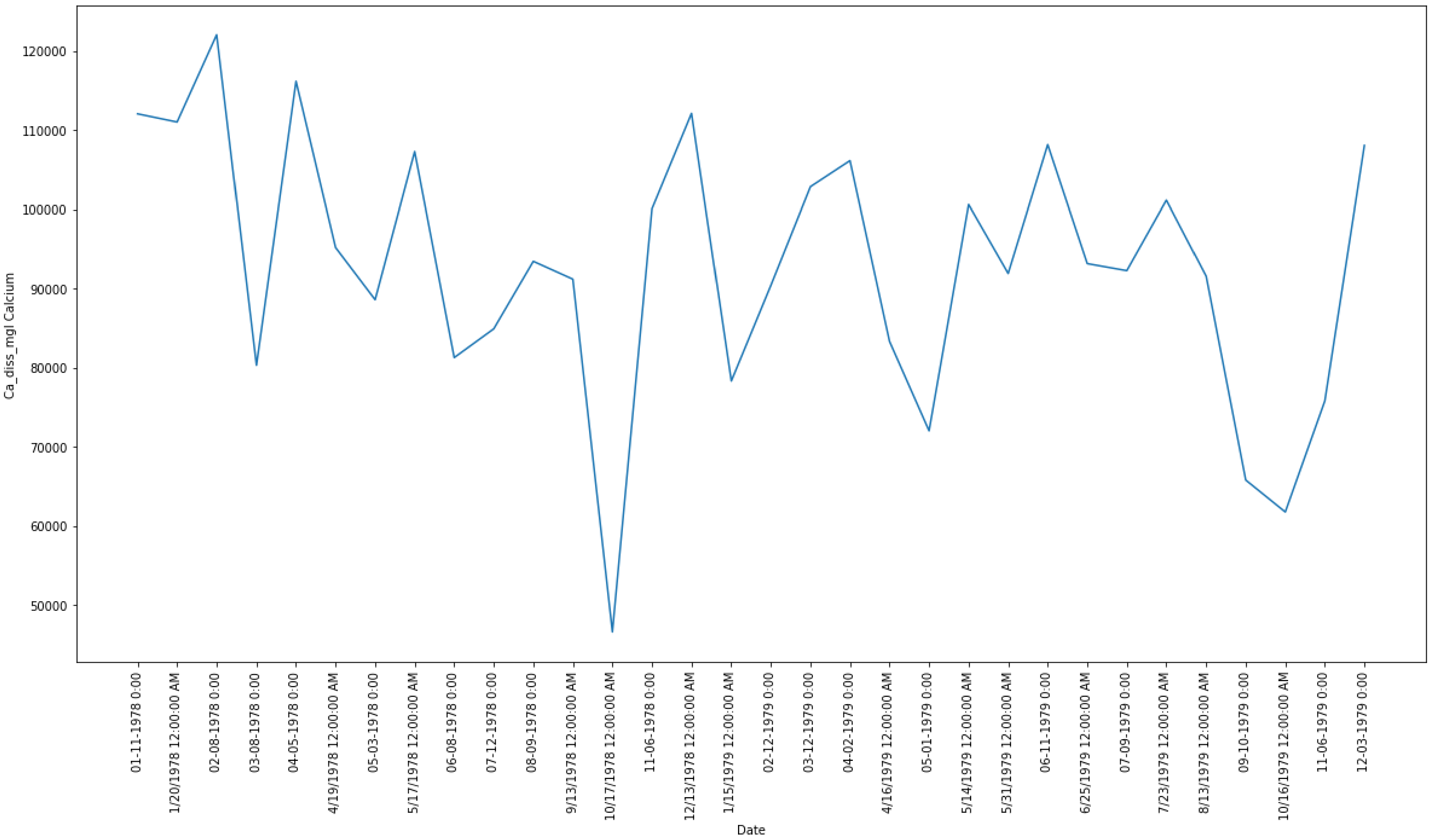
In [ ]:

```
val = 55
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



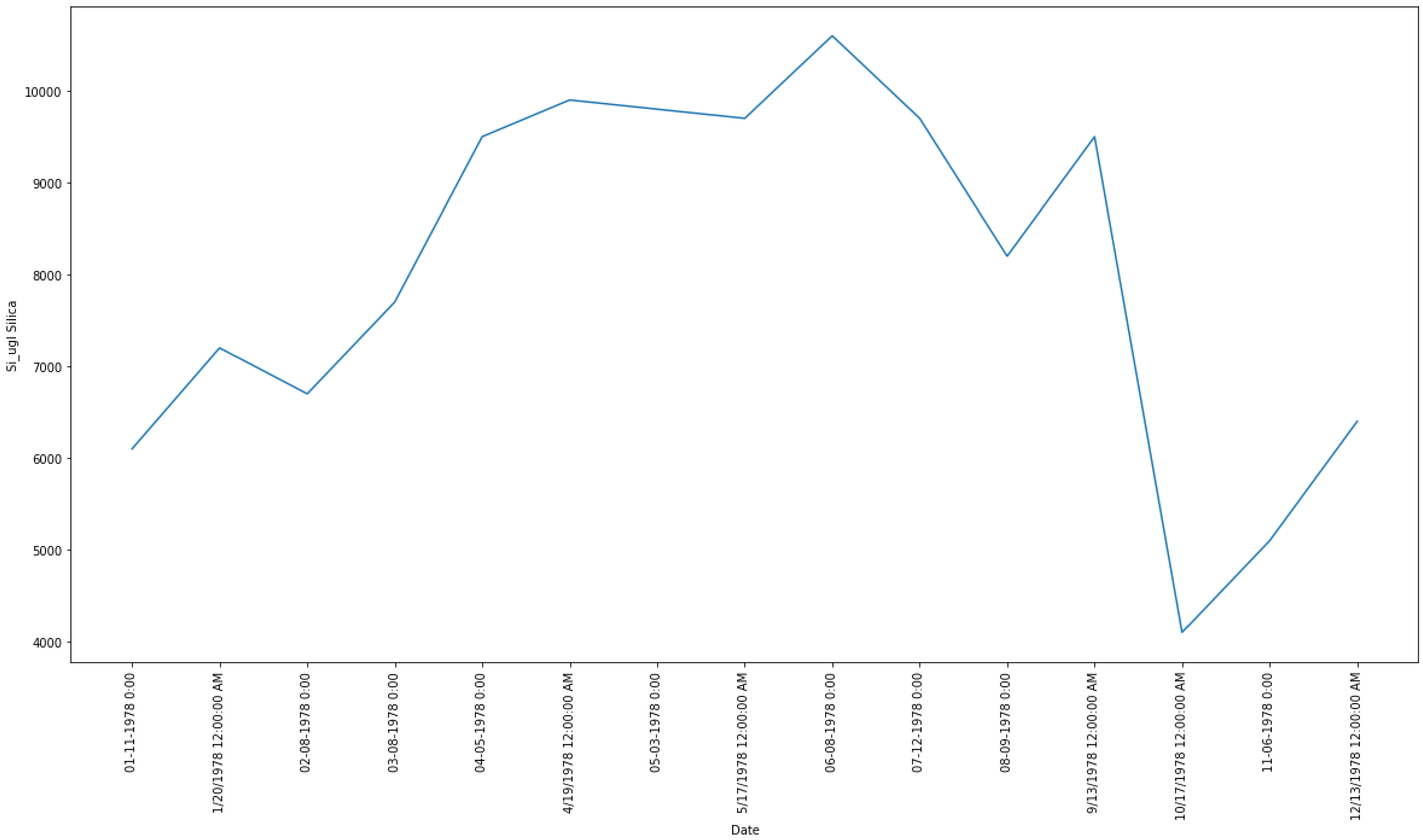
In [ ]:

```
val = 56
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



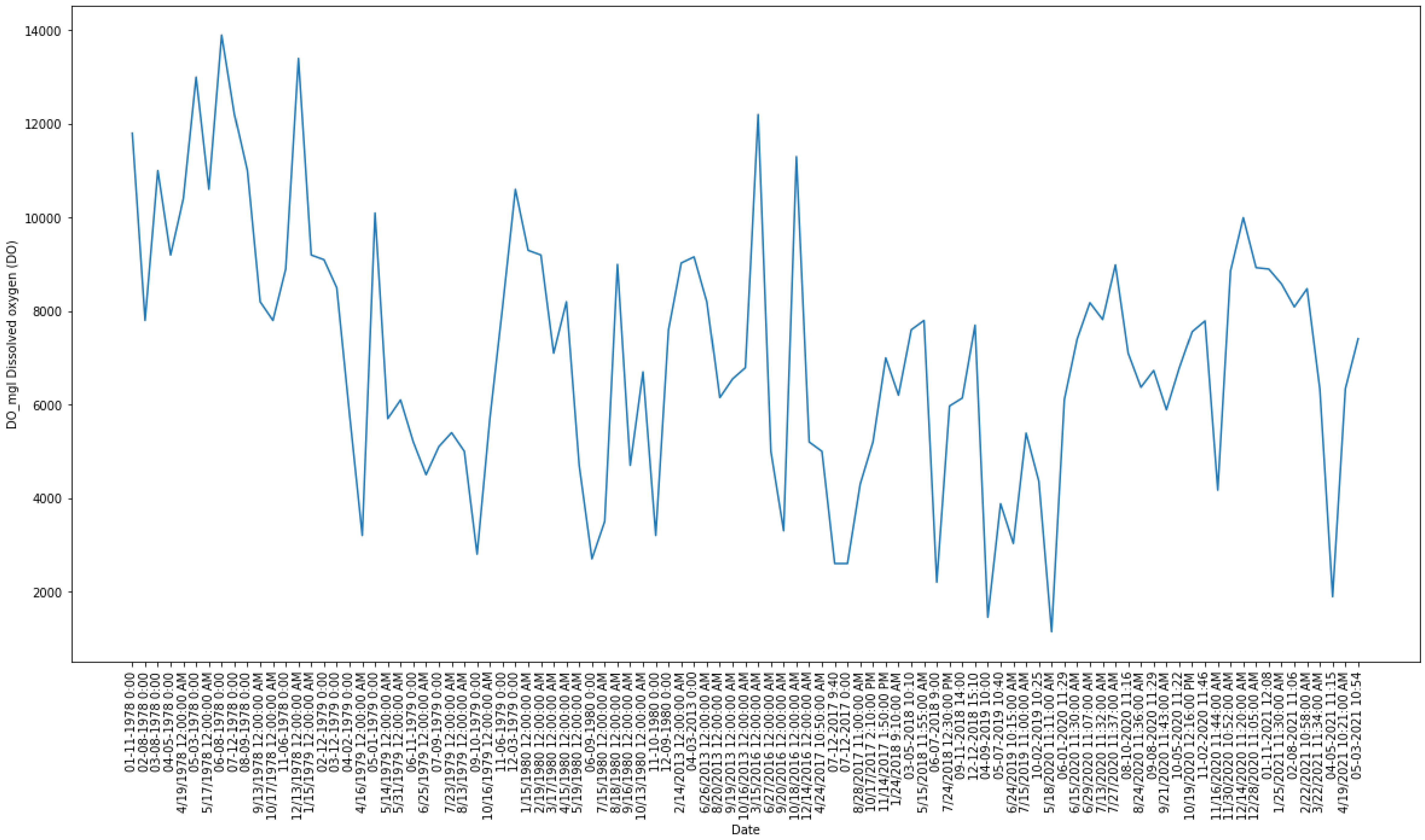
In [ ]:

```
val = 57
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



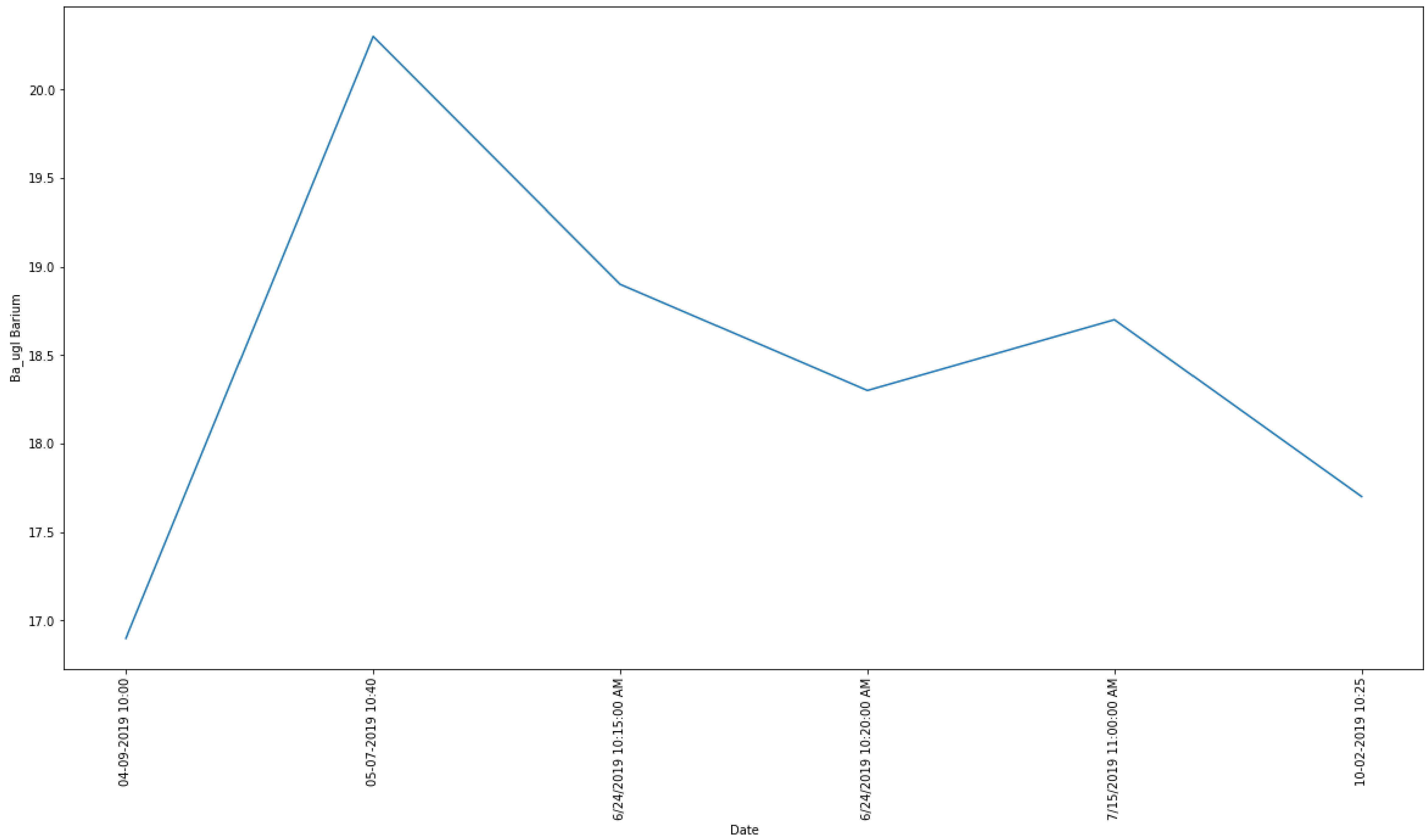
In [ ]:

```
val = 58
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



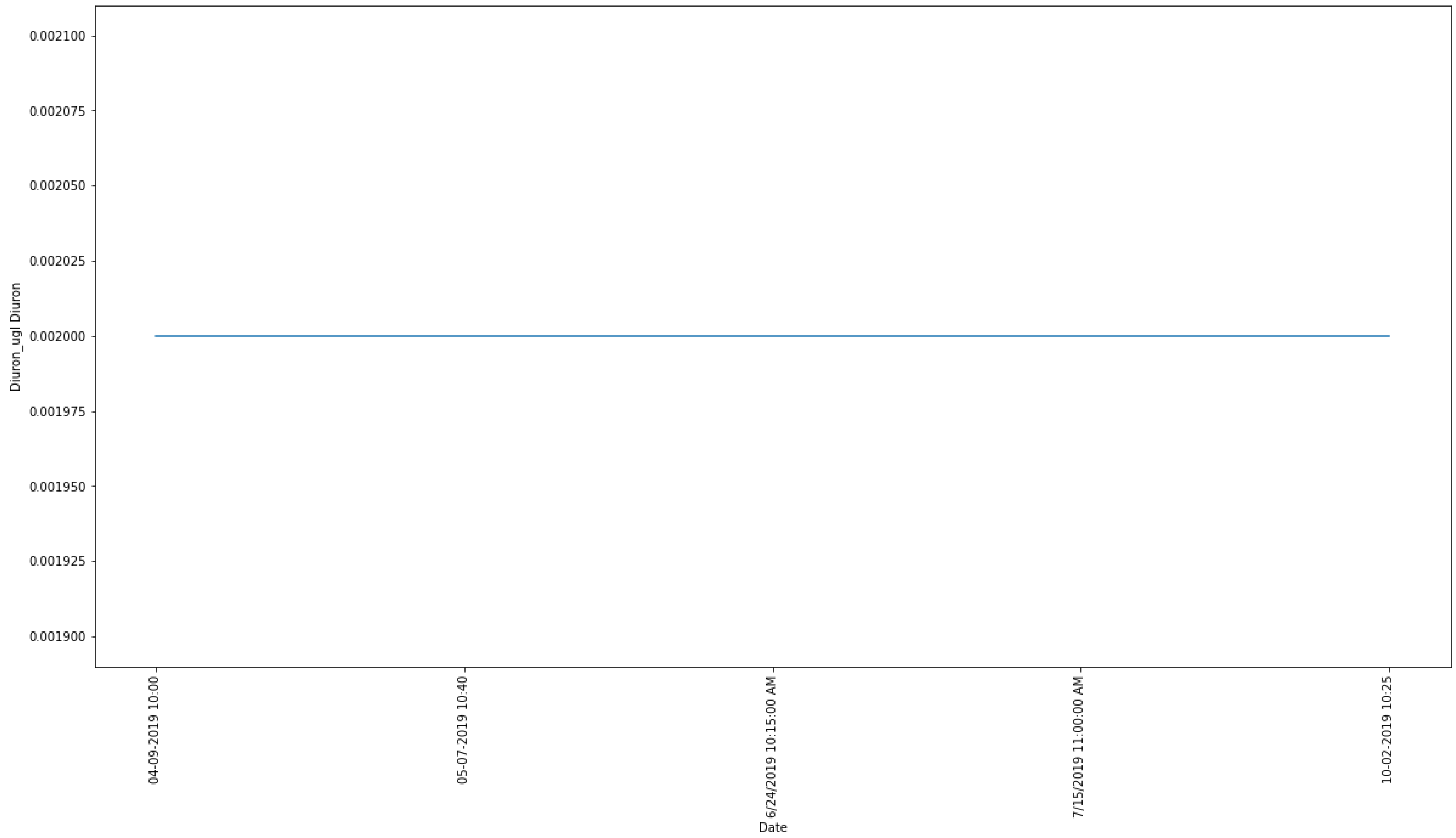
In [ ]:

```
val = 59
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



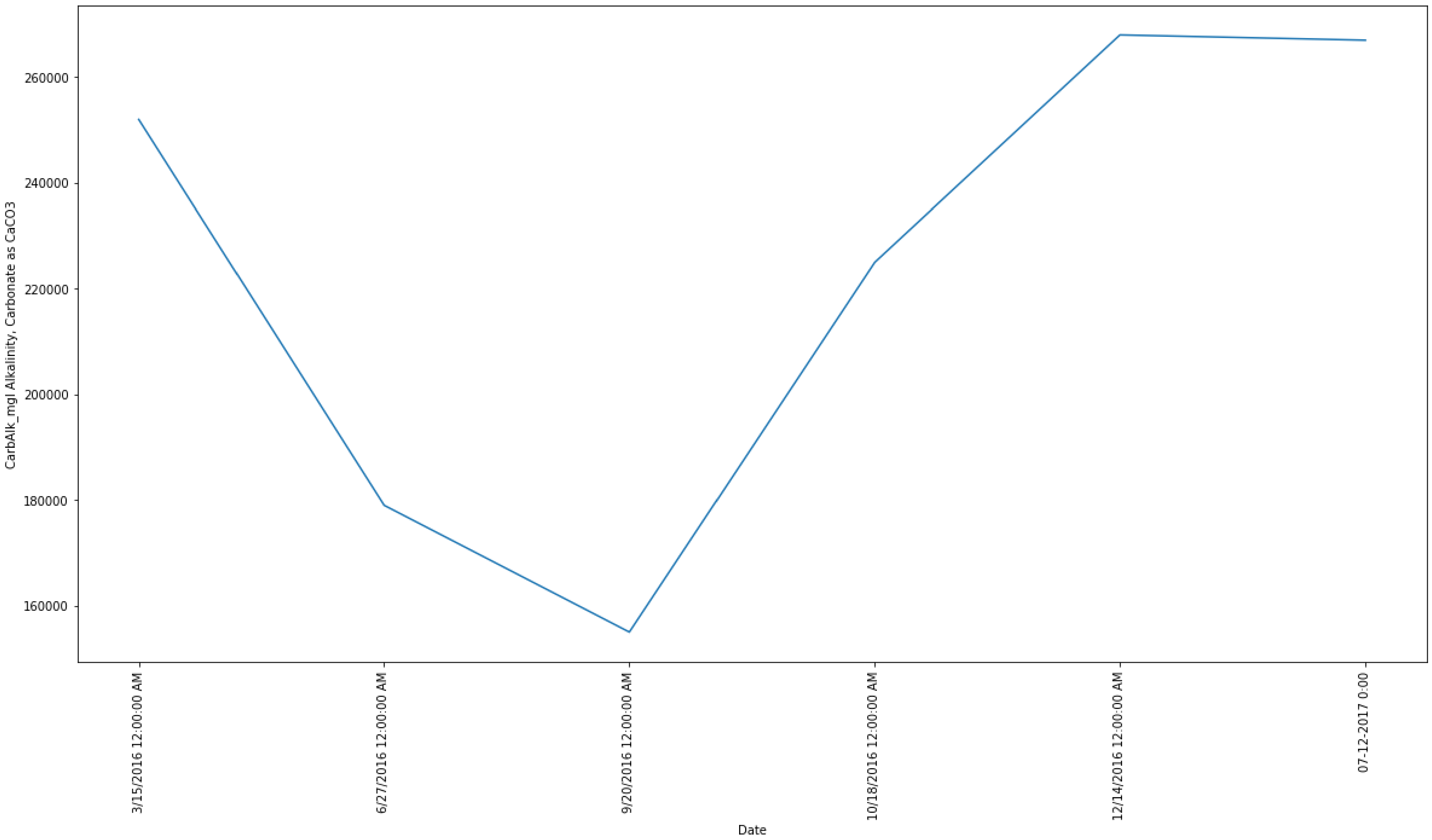
In [ ]:

```
val = 60
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```

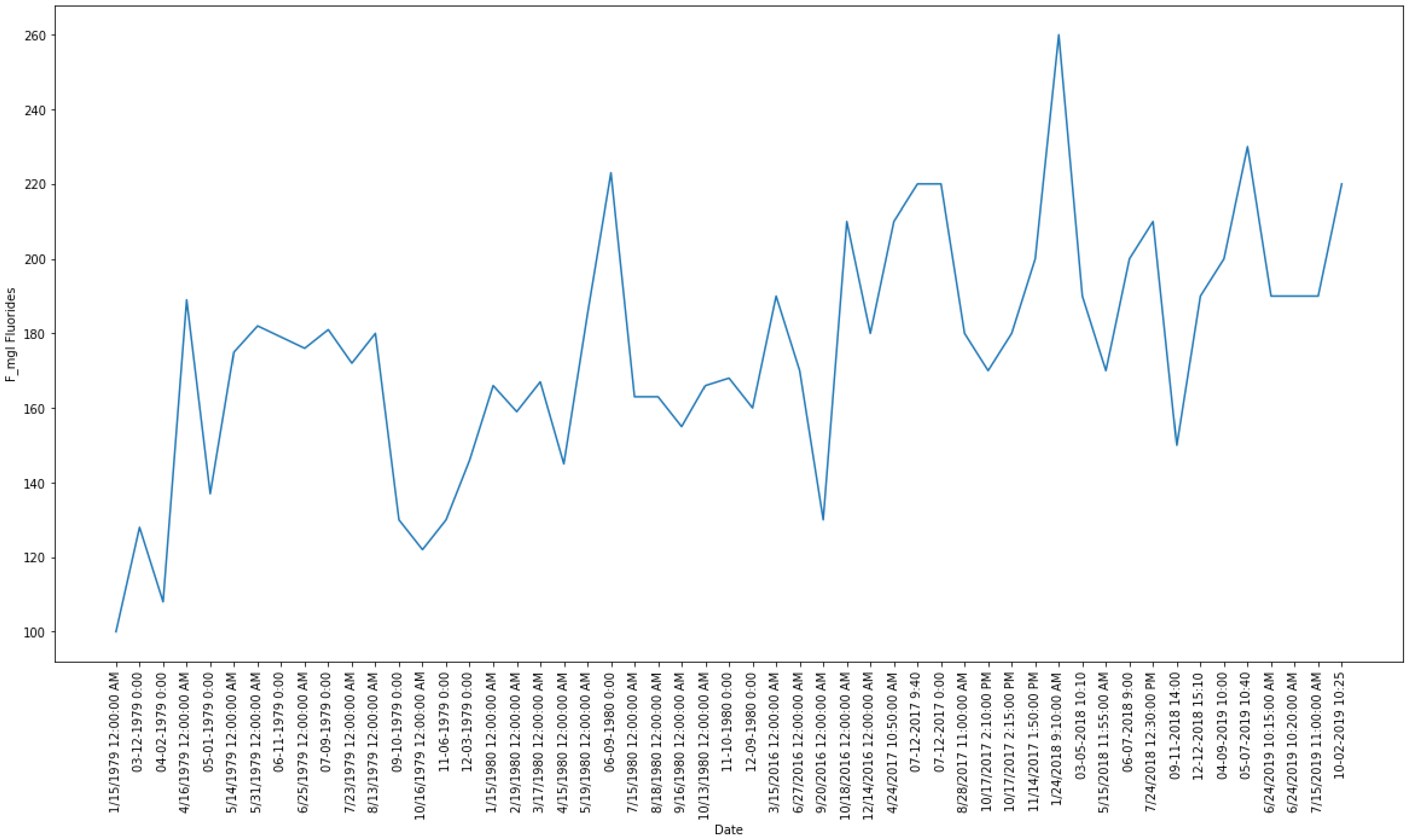


In [ ]:

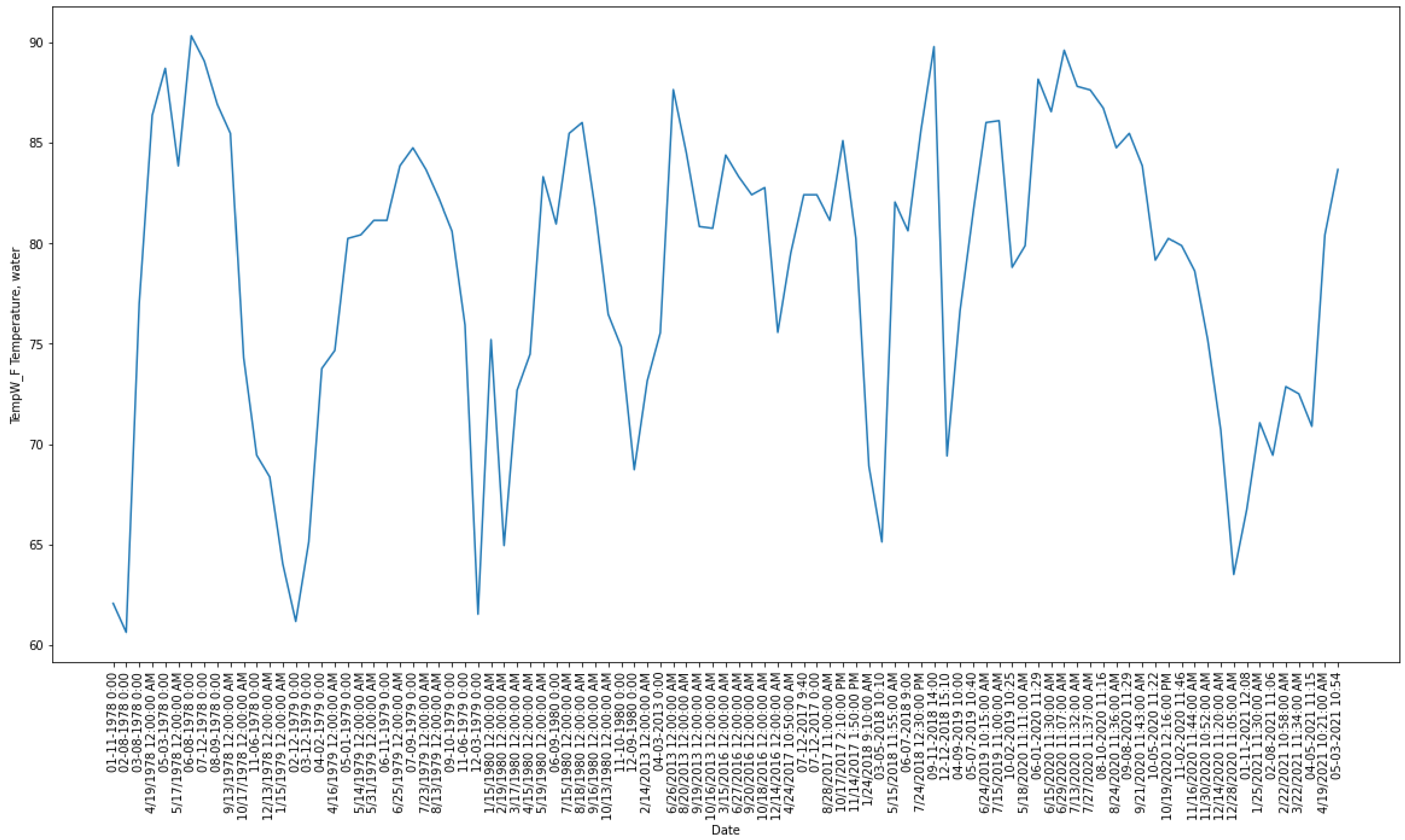
```
val = 61
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



```
In [ ]: val = 62
        df_new = fdf_values[fdf_values[col[val]]!='na']
        plt.figure(figsize=(20,10))
        plt.plot(df_new['Date'], df_new[col[val]])
        plt.xticks(rotation = 90)
        plt.xlabel("Date")
        plt.ylabel(col[val])
        plt.show()
```

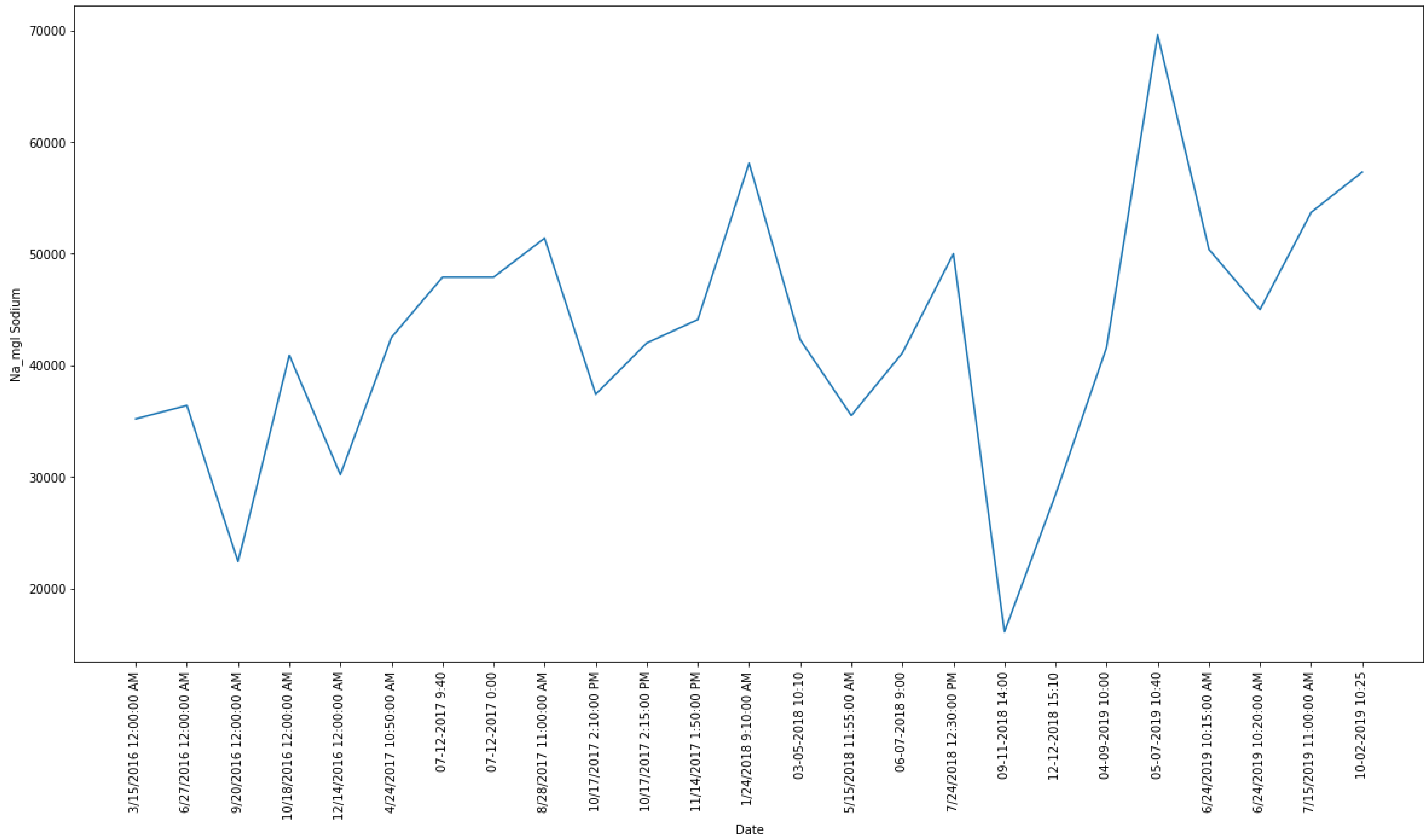


```
In [ ]: val = 63
        df_new = fdf_values[fdf_values[col[val]]!='na']
        plt.figure(figsize=(20,10))
        plt.plot(df_new['Date'], df_new[col[val]])
        plt.xticks(rotation = 90)
        plt.xlabel("Date")
        plt.ylabel(col[val])
        plt.show()
```



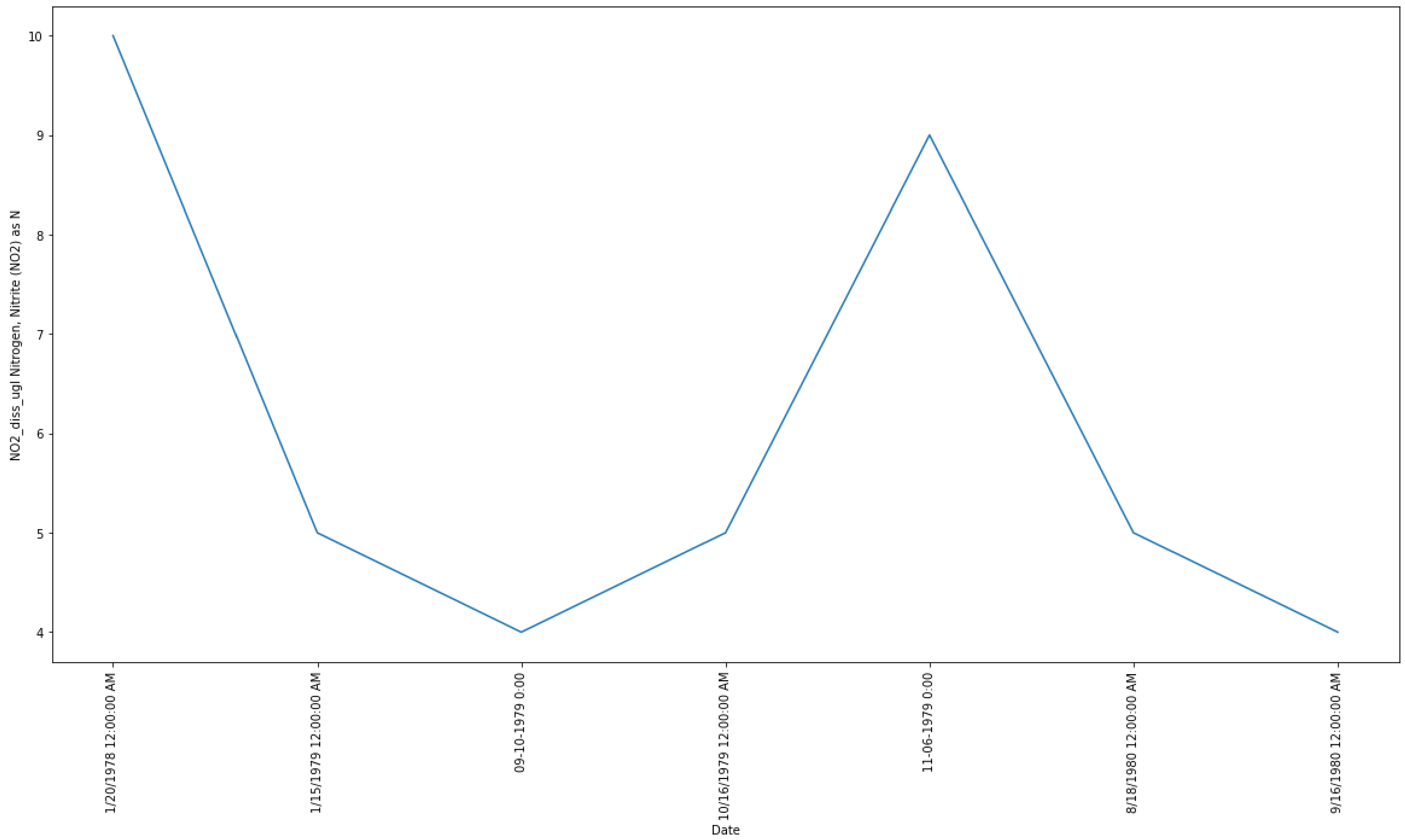
In [ ]:

```
val = 64
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



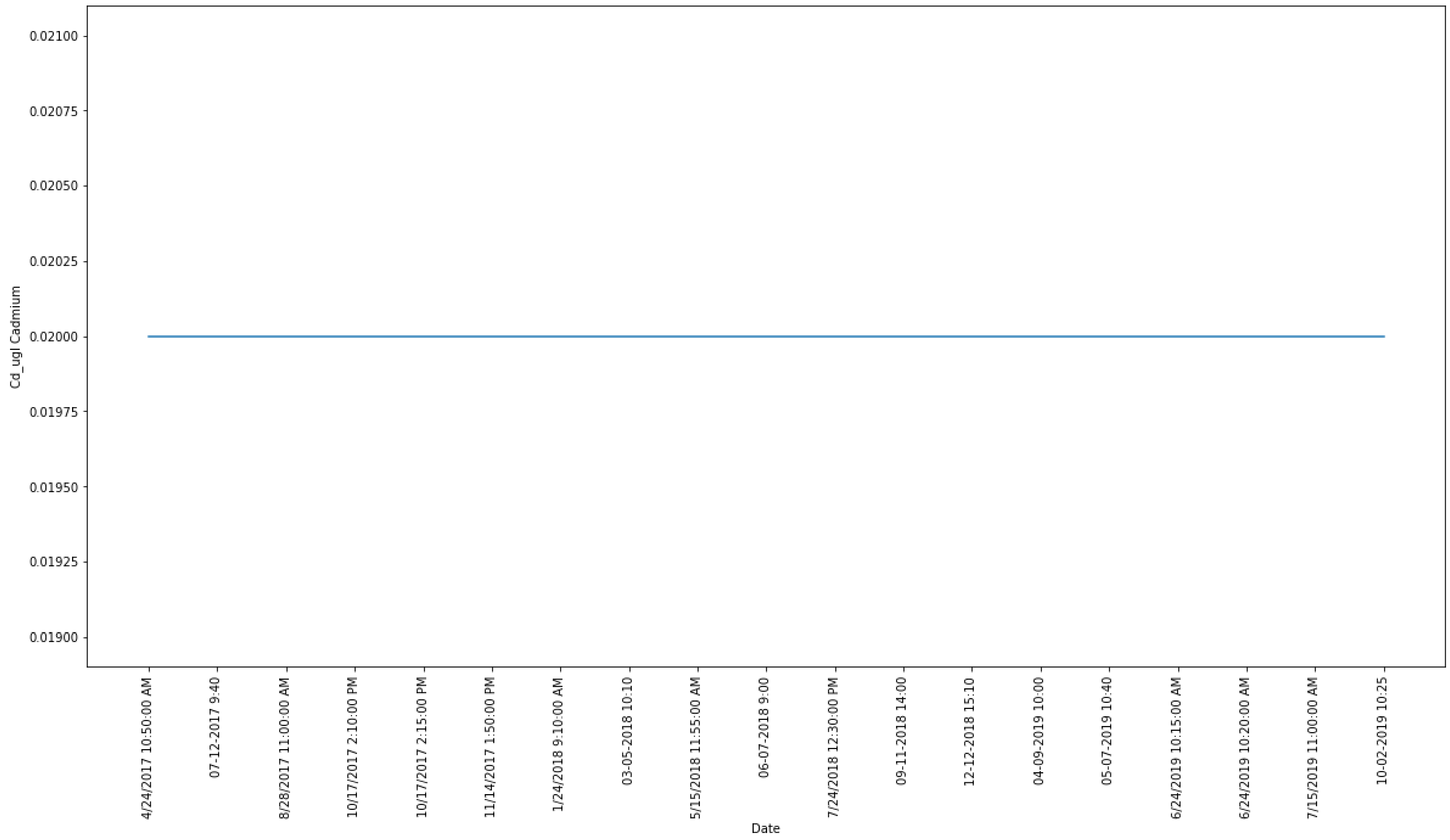
In [ ]:

```
val = 65
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



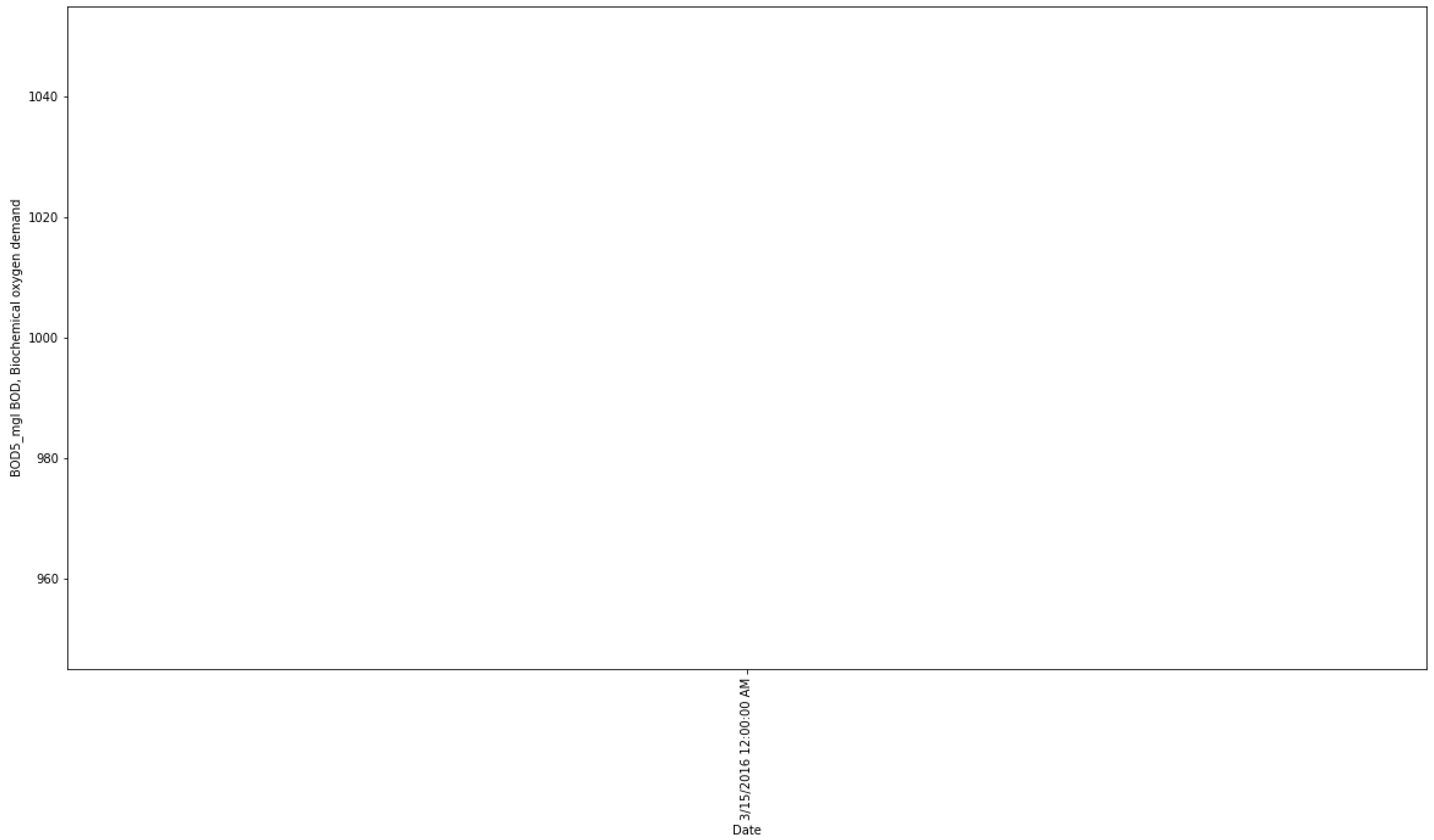
In [ ]:

```
val = 66
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



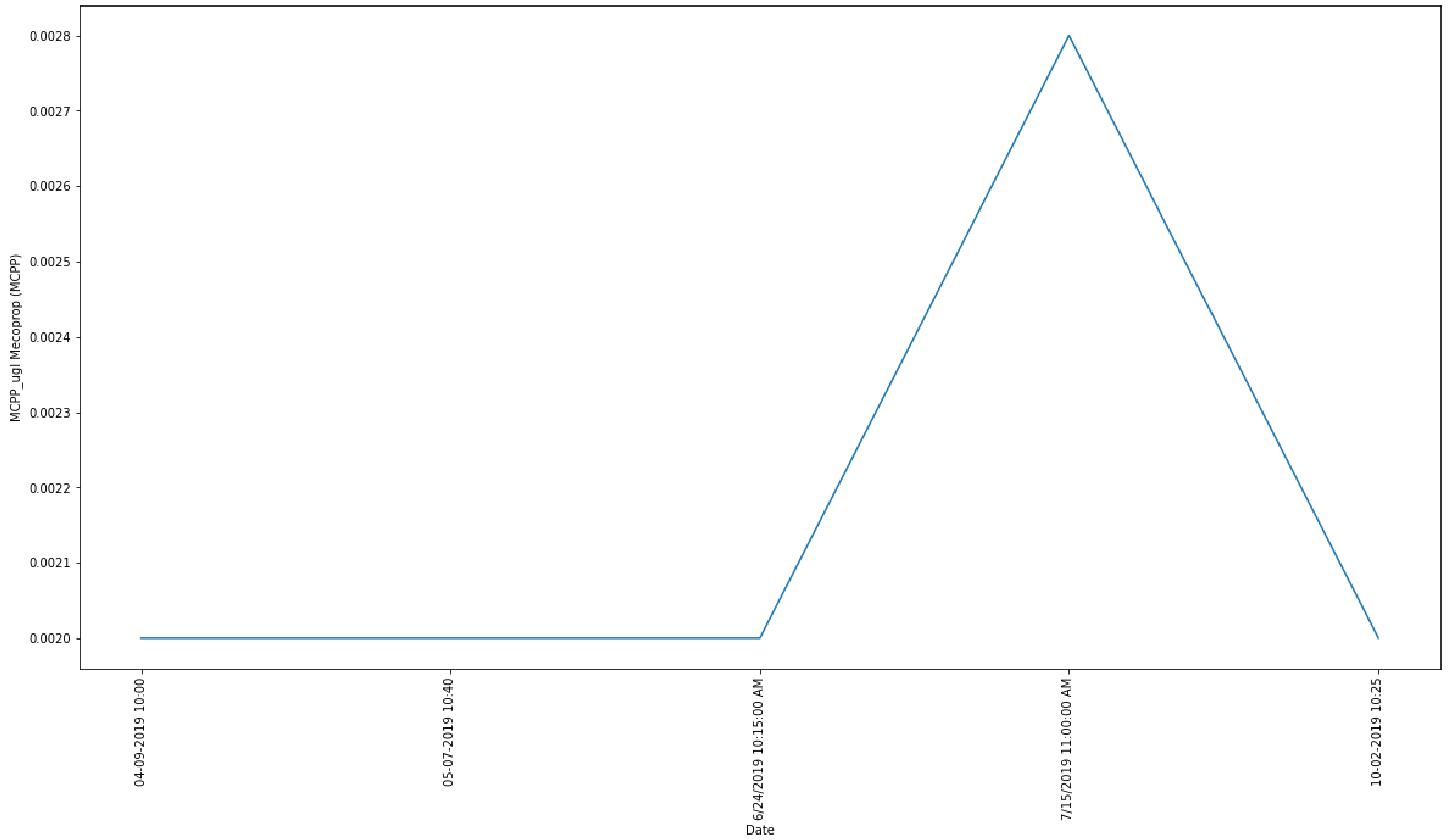
In [ ]:

```
val = 67
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



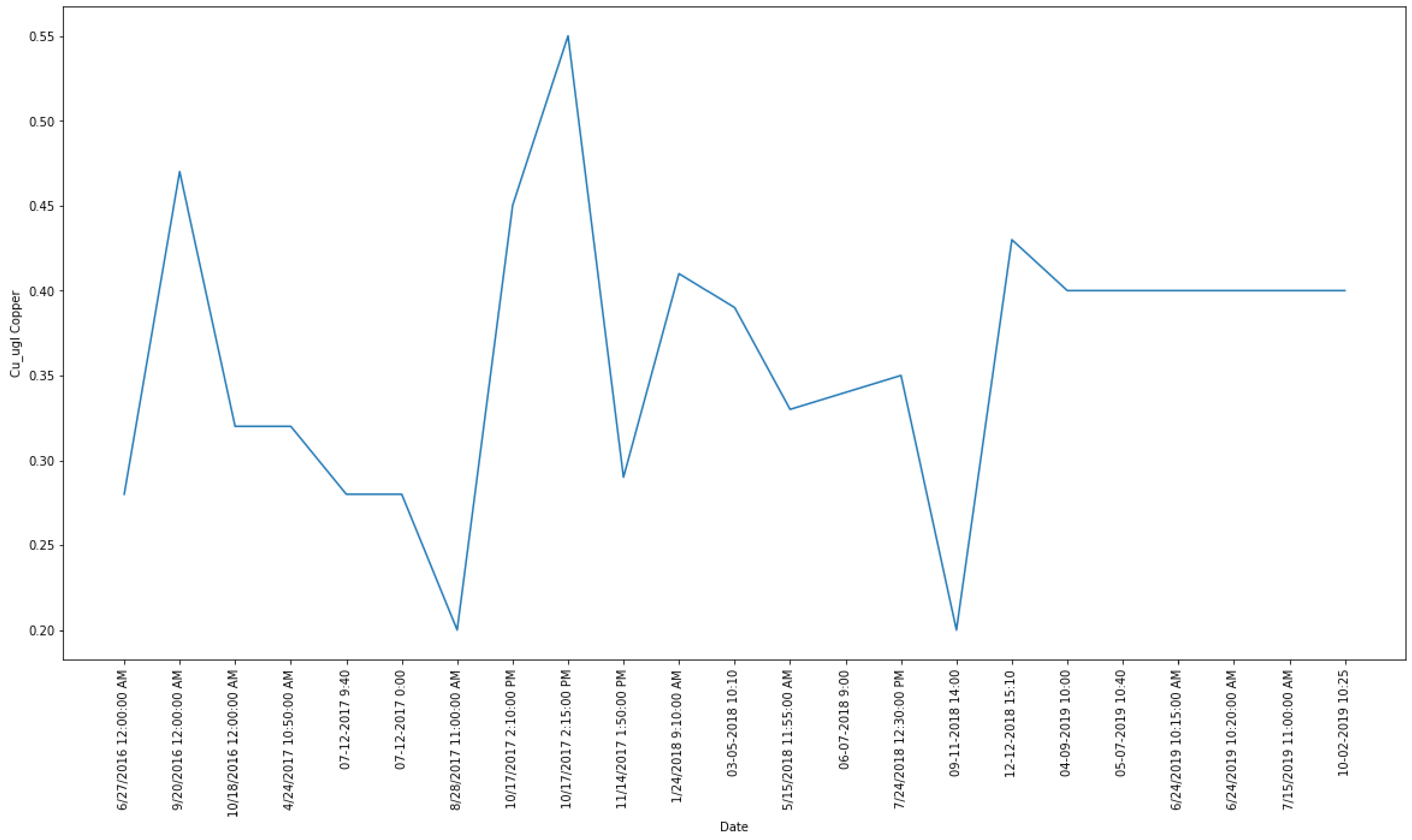
In [ ]:

```
val = 68
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



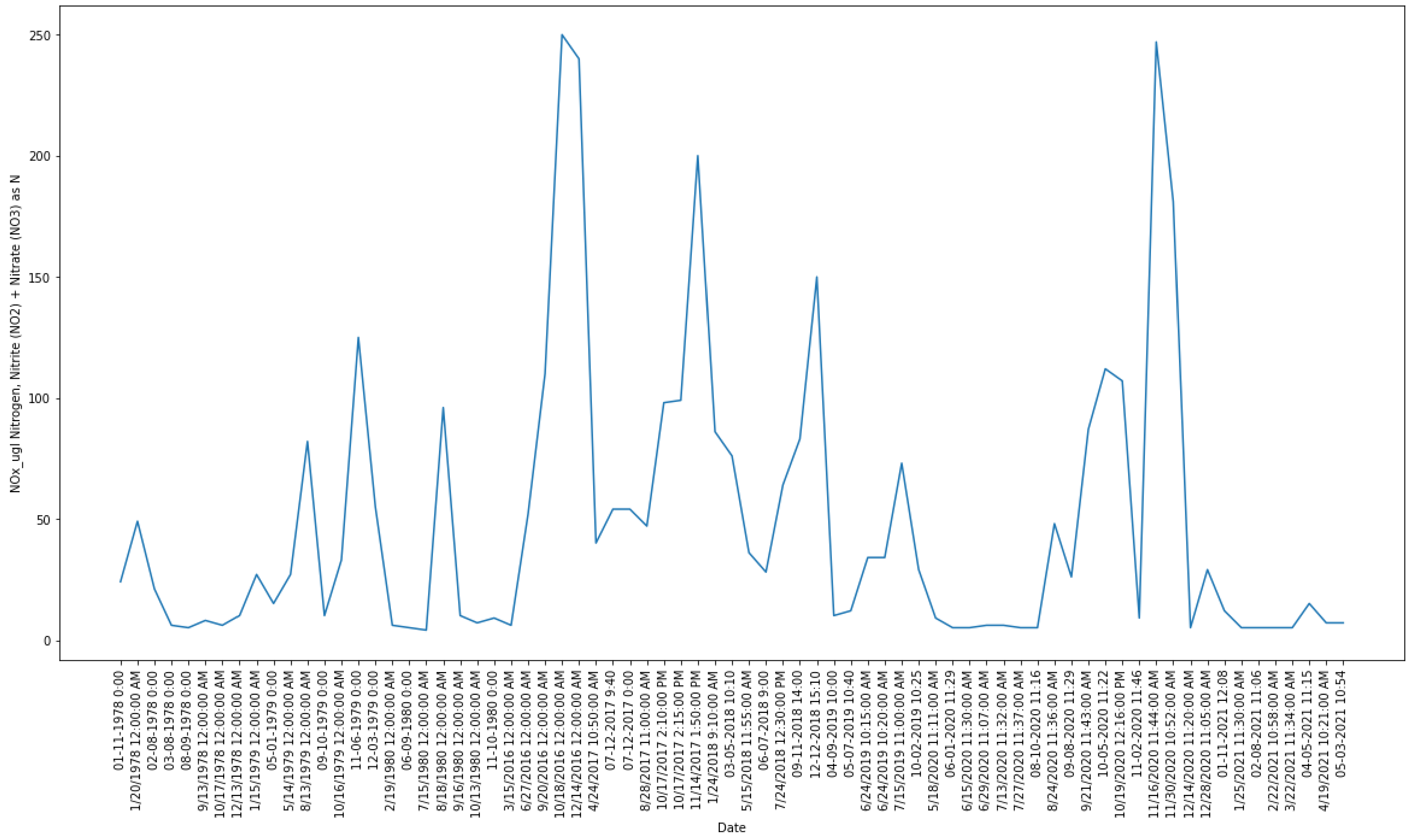
In [ ]:

```
val = 69
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



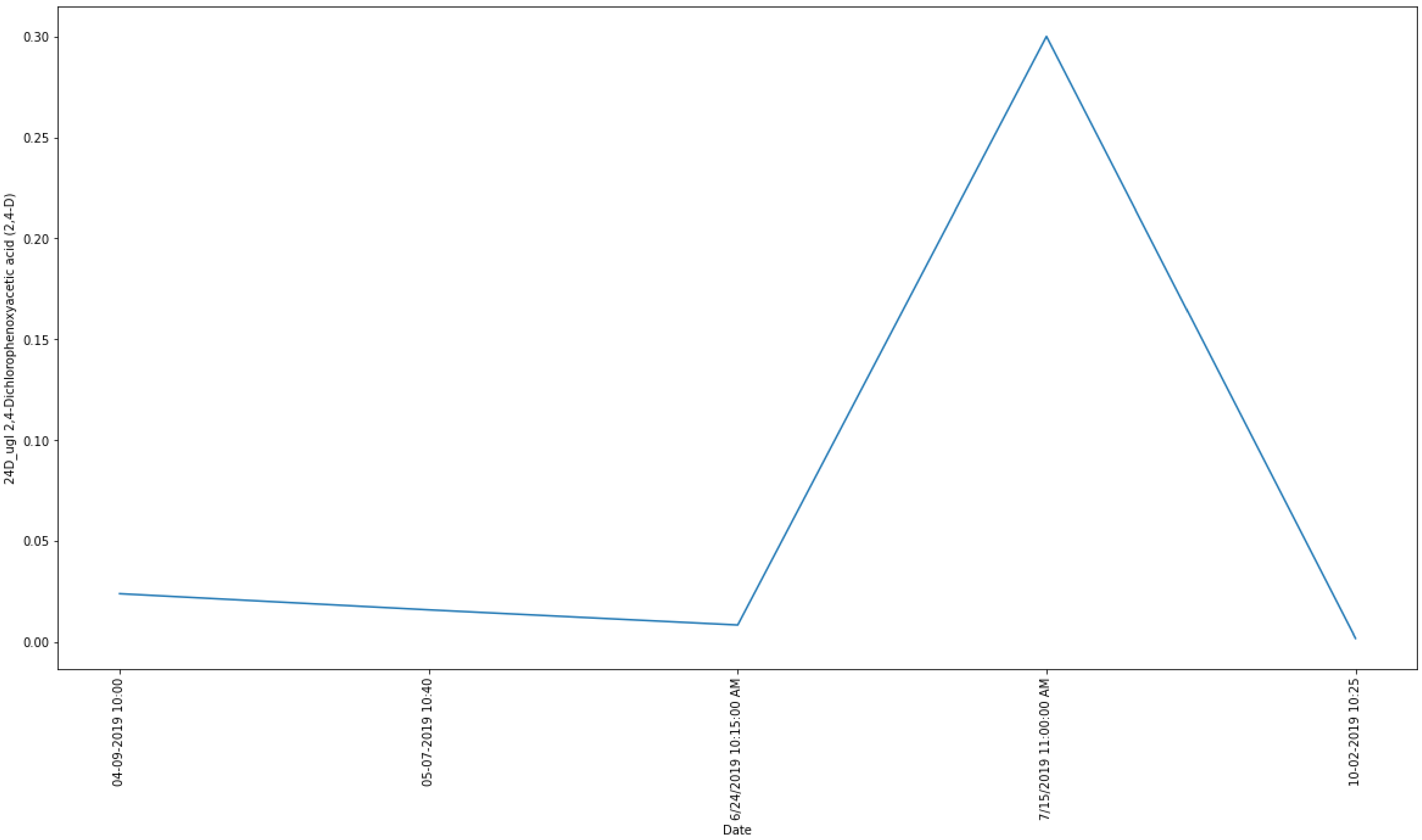
In [ ]:

```
val = 70
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



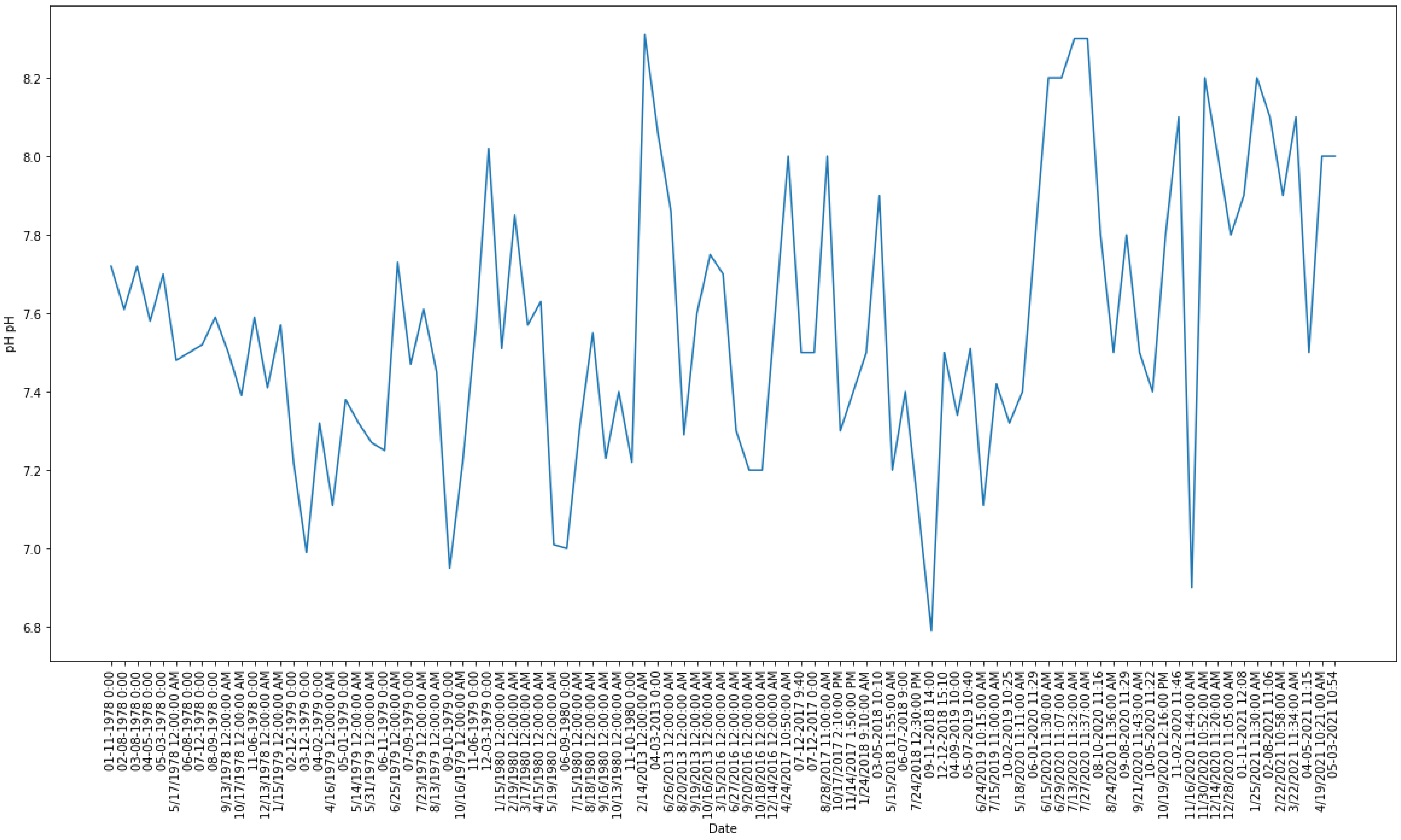
In [ ]:

```
val = 71
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



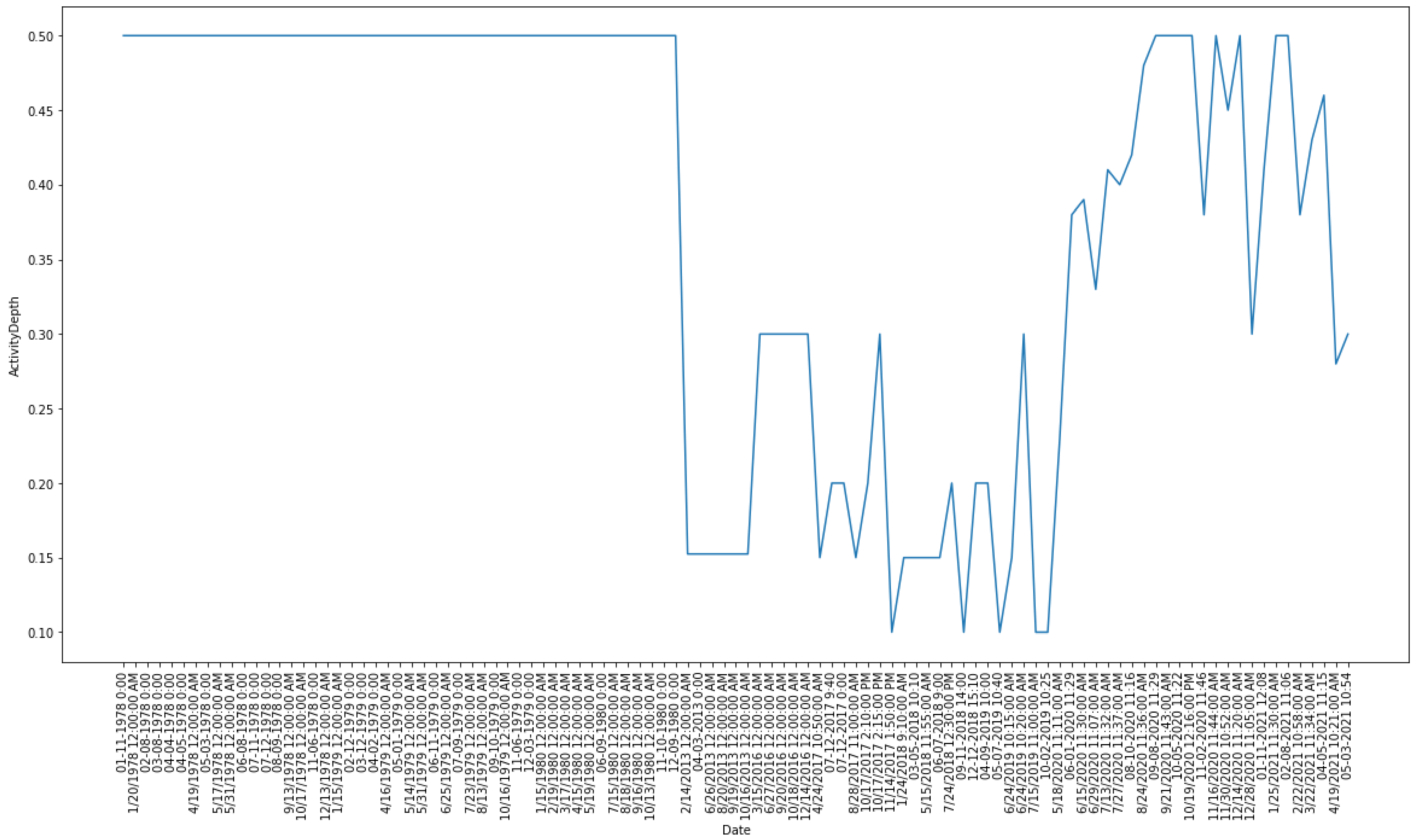
In [ ]:

```
val = 72
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



In [ ]:

```
val = 73
df_new = fdf_values[fdf_values[col[val]]!='na']
plt.figure(figsize=(20,10))
plt.plot(df_new['Date'], df_new[col[val]])
plt.xticks(rotation = 90)
plt.xlabel("Date")
plt.ylabel(col[val])
plt.show()
```



Task 03

In [ ]:

```python
# fdf_values contain values of all the paramters for 103 days.
# Extracting list of the parameters with maximum number of not 'na' values in their column.

col = fdf_values.columns
col = list(col)
sane_value = list()
for name in col:
    df_new = fdf_values[fdf_values[name]!="na"]
    #print(name, len(df_new))
    sane_value.append(len(df_new))
temp_df = pd.DataFrame(data= None, columns=['col','val'])
temp_df['col'] = col
temp_df['val'] = sane_value
temp_df = temp_df.sort_values(by=['val'], ascending=False)
temp_df.head(20)
```

Out[ ]:

	col	val
0	Date	103
73	ActivityDepth	103
74	nDate	103
38	Cond_umhocm Specific conductance	97
58	DO_mgl Dissolved oxygen (DO)	97
63	TempW_F Temperature, water	95
55	TempW_C Temperature, water	95
72	pH pH	95
35	TP_ugl Phosphorus as P	94
15	OP_mgl Phosphorus, phosphate (PO4) as P	80
70	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3...	73
41	Fe_ugl Iron	72
27	TN_ugl Nitrogen	72
52	Color_true_pcu True Color	68
16	TKN_ugl Nitrogen, Kjeldahl	67
23	K_mgl Potassium	57
62	F_mgl Fluorides	53
5	NH3_N_ugl Nitrogen, ammonia as N	50
44	Cl_diss_mgl Chloride	44
32	Pheo_ugl Pheophytin-a	40

In [ ]:

```python
fdf_values
```

Out[ ]:

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	24D_ugl 2 Dichlorophenoxyac acid (2,4
71	01-11-1978 0:00	na	na	na	na	na	16.0	na	na	na	...	na	na	na	na	na	24.0	
68	1/20/1978 12:00:00 AM	na	na	na	na	na	na	na	na	na	...	10.0	na	na	na	na	49.0	
19	02-08-1978 0:00	na	na	na	na	na	18.0	na	na	na	...	na	na	na	na	na	21.0	
91	03-08-1978 0:00	na	na	na	na	na	19.0	na	na	na	...	na	na	na	na	na	6.0	
13	04-04-1978 0:00	na	na	na	na	na	na	na	na	na	...	na	na	na	na	na	na	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
30	2/22/2021 10:58:00 AM	na	na	na	na	15.0	na	na	na	na	...	na	na	na	na	na	5.0	

	Date	Ni_ugl Nickel	Sucralose_ug/l Sucralose	Cl_mgl Chloride	Linuron_ugl Linuron	NH3_N_ugl Nitrogen, ammonia as N	Mn_diss_ugl Manganese	Ag_ugl Silver	Depth_bott_ft Depth, bottom	Mn_ugl Manganese	...	NO2_diss_ugl Nitrogen, Nitrite (NO2) as N	Cd_ugl Cadmium	BOD5_mgl BOD, Biochemical oxygen demand	MCPP_ugl Mecoprop (MCPP)	Cu_ugl Copper	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	24D_ugl 2 Dichlorophenoxyac acid (2,4
41	3/22/2021 11:34:00 AM	na	na	na	na	44.0	na	na	na	na	...	na	na	na	na	na	5.0	
2	04-05- 2021 11:15	na	na	na	na	303.0	na	na	na	na	...	na	na	na	na	na	15.0	
6	4/19/2021 10:21:00 AM	na	na	na	na	70.0	na	na	na	na	...	na	na	na	na	na	7.0	
92	05-03- 2021 10:54	na	na	na	na	61.0	na	na	na	na	...	na	na	na	na	na	7.0	

103 rows × 75 columns

In [ ]:

```
# Taking the parameters with 70 or more non-na values in their columns.
col_corr = temp_df['col'].values.tolist()
col_corr = col_corr[:13]
del temp_df
```

In [ ]:

```
# dropping irrelevant column name
col_corr.pop(1)
```

Out[ ]:

```
'nDate'
```

In [ ]:

```
# list of all the paramters
col_corr
```

Out[ ]:

```
['ActivityDepth',
 'Cond_umhocm Specific conductance',
 'DO_mgl Dissolved oxygen (DO)',
 'TempW_F Temperature, water',
 'TempW_C Temperature, water',
 'pH pH',
 'TP_ugl Phosphorus as P',
 'OP_mgl Phosphorus, phosphate (PO4) as P',
 'NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N',
 'Fe_ugl Iron',
 'TN_ugl Nitrogen']
```

In [ ]:

```
# creating a dataframe to calculate correlation from.
correlation = fdf_values[col_corr]
```

In [ ]:

```
# making a copy of the dataframe, as without copy python creates a pointed which
# essentially draws back changes all the way back to the source variable
correlation = correlation.copy()
```

In [ ]:

```
# removing all the rows with 'na' value in them.
for col in correlation.columns:
    correlation = correlation[correlation[col]!='na']
```

In [ ]:

```
len(correlation)
```

Out[ ]:

```
38
```

In [ ]:

```
!pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
     |████████████████████████████████| 292 kB 2.8 MB/s
Requirement already satisfied: scipy>=1.0 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from seaborn) (1.7.0)
Requirement already satisfied: matplotlib>=2.2 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from seaborn) (3.4.3)
Requirement already satisfied: numpy>=1.15 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from seaborn) (1.21.0)
Requirement already satisfied: pandas>=0.23 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (8.3.2)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: six in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /Users/oldxchange/anaconda3/envs/kgconst/lib/python3.7/site-packages (from pandas>=0.23->seaborn) (2021.1)
Installing collected packages: seaborn
Successfully installed seaborn-0.11.2
```

In [ ]:

```
correlation
```

Out[ ]:

	ActivityDepth	Cond_umhocm Specific conductance	DO_mgl Dissolved oxygen (DO)	TempW_F Temperature, water	TempW_C Temperature, water	pH pH	TP_ugl Phosphorus as P	OP_mgl Phosphorus, phosphate (PO4) as P	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Fe_ugl Iron	TN_ugl Nitrogen
19	0.50	720.0	7800.0	60.62	15.9	7.61	19.0	2.0	21.0	250.0	1301.0
84	0.50	542.0	11000.0	86.9	30.5	7.59	35.0	4.0	5.0	130.0	765.0
27	0.50	378.0	8200.0	85.46	29.7	7.5	25.0	7.0	8.0	220.0	978.0
50	0.50	618.0	13400.0	68.36	20.2	7.41	13.0	3.0	10.0	200.0	2670.0
62	0.50	389.0	9200.0	64.04	17.8	7.57	56.0	31.0	27.0	490.0	1907.0
67	0.50	354.0	10100.0	80.24	26.8	7.38	34.0	6.0	15.0	200.0	765.0
93	0.50	563.0	5000.0	82.22	27.9	7.45	26.0	22.0	82.0	370.0	842.0
43	0.50	410.0	2800.0	80.6	27.0	6.95	222.0	165.0	10.0	1170.0	1860.0
51	0.50	469.0	8100.0	75.92	24.4	7.56	34.0	14.0	125.0	490.0	2025.0
24	0.50	556.0	10600.0	61.52	16.4	8.02	19.0	20.0	55.0	130.0	1415.0
46	0.50	642.0	9200.0	64.94	18.3	7.85	36.0	10.0	6.0	310.0	746.0
44	0.50	253.0	2700.0	80.96	27.2	7.0	44.0	9.0	5.0	410.0	1095.0
98	0.50	452.0	4700.0	81.68	27.6	7.23	131.0	68.0	10.0	370.0	1660.0
25	0.50	488.0	6700.0	76.46	24.7	7.4	35.0	36.0	7.0	270.0	2367.0
54	0.50	546.0	3200.0	74.84	23.8	7.22	27.0	14.0	9.0	130.0	899.0
79	0.30	658.0	12200.0	84.38	29.1	7.7	44.0	19.0	6.0	450.0	856.0
9	0.30	525.0	5000.0	83.3	28.5	7.3	180.0	110.0	52.0	1200.0	1252.0
18	0.30	418.0	3300.0	82.4	28.0	7.2	170.0	130.0	110.0	1010.0	1210.0
34	0.30	650.0	11300.0	82.76	28.2	7.2	95.0	57.0	250.0	730.0	1350.0
64	0.30	649.0	5200.0	75.56	24.2	7.6	64.0	19.0	240.0	630.0	420.0

	ActivityDepth	Cond_umhocm Specific conductance	DO_mgl Dissolved oxygen (DO)	TempW_F Temperature, water	TempW_C Temperature, water	pH pH	TP_ugl Phosphorus as P	OP_mgl Phosphorus, phosphate (PO4) as P	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Fe_ugl Iron	TN_ugl Nitrogen
99	0.15	737.0	5000.0	79.52	26.4	8.0	71.0	14.0	40.0	560.0	1050.0
33	0.20	834.0	2600.0	82.4	28.0	7.5	86.0	31.0	54.0	770.0	1014.0
38	0.20	834.0	2600.0	82.4	28.0	7.5	86.0	31.0	54.0	770.0	1014.0
101	0.15	673.0	4300.0	81.14	27.3	8.0	64.0	34.0	47.0	590.0	1047.0
49	0.20	592.0	5200.0	85.1	29.5	7.3	170.0	130.0	98.0	1190.0	1698.0
4	0.10	665.0	7000.0	80.24	26.8	7.4	90.0	70.0	200.0	720.0	1400.0
12	0.15	573.0	6200.0	68.9	20.5	7.5	56.0	24.0	86.0	440.0	876.0
66	0.15	711.0	7600.0	65.12	18.4	7.9	55.0	19.0	76.0	580.0	916.0
5	0.15	672.0	7800.0	82.04	27.8	7.2	35.0	15.0	36.0	340.0	716.0
53	0.15	732.0	2200.0	80.618	27.01	7.4	72.0	30.0	28.0	950.0	898.0
55	0.20	724.0	5970.0	85.622	29.79	7.1	82.0	31.0	64.0	750.0	954.0
21	0.10	477.5	6140.0	89.78	32.1	6.79	49.0	29.0	83.0	860.0	1053.0
88	0.20	665.0	7700.0	69.404	20.78	7.5	130.0	30.0	150.0	2240.0	1150.0
78	0.20	667.0	1450.0	76.64	24.8	7.34	83.0	32.0	10.0	620.0	950.0
77	0.10	810.0	3880.0	81.5	27.5	7.51	82.0	31.0	12.0	670.0	1112.0
14	0.15	681.0	3030.0	86.0	30.0	7.11	100.0	46.0	34.0	1080.0	1234.0
58	0.10	668.0	5390.0	86.09	30.05	7.42	86.0	36.0	73.0	1050.0	1373.0
85	0.10	760.0	4360.0	78.8	26.0	7.32	82.0	28.0	29.0	560.0	1029.0

In [ ]:

```
# converting all the values to num

for col in correlation.columns:
    temp = correlation[col].values.tolist()
    temp = [float(i) for i in temp]
    correlation[col] = temp
```

In [ ]:

```
# importing library for plotting the heatmap and calculating correlation

import seaborn as sns
%matplotlib inline
import numpy as np

corr = correlation.corr()
```

In [ ]:

```
corr
```

Out[ ]:

	ActivityDepth	Cond_umhocm Specific conductance	DO_mgl Dissolved oxygen (DO)	TempW_F Temperature, water	TempW_C Temperature, water	pH pH	TP_ugl Phosphorus as P	OP_mgl Phosphorus, phosphate (PO4) as P	NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	Fe_ugl Iron	TN_ugl Nitrogen
ActivityDepth	1.000000	-0.649079	0.375597	-0.330009	-0.330009	0.038809	-0.278471	-0.109821	-0.299982	-0.504908	0.332015
Cond_umhocm Specific conductance	-0.649079	1.000000	-0.152869	-0.030690	-0.030690	0.334919	-0.018467	-0.188248	0.147833	0.199683	-0.267100
DO_mgl Dissolved oxygen (DO)	0.375597	-0.152869	1.000000	-0.327614	-0.327614	0.326464	-0.456975	-0.353580	0.088947	-0.349182	0.216412
TempW_F Temperature, water	-0.330009	-0.030690	-0.327614	1.000000	1.000000	-0.514004	0.329162	0.296924	0.033915	0.222477	-0.199251
TempW_C Temperature, water	-0.330009	-0.030690	-0.327614	1.000000	1.000000	-0.514004	0.329162	0.296924	0.033915	0.222477	-0.199251
pH pH	0.038809	0.334919	0.326464	-0.514004	-0.514004	1.000000	-0.377476	-0.403626	-0.024143	-0.267522	-0.133760
TP_ugl Phosphorus as P	-0.278471	-0.018467	-0.456975	0.329162	0.329162	-0.377476	1.000000	0.916089	0.205477	0.735634	0.148973
OP_mgl Phosphorus, phosphate (PO4) as P	-0.109821	-0.188248	-0.353580	0.296924	0.296924	-0.403626	0.916089	1.000000	0.190320	0.536218	0.297481
NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N	-0.299982	0.147833	0.088947	0.033915	0.033915	-0.024143	0.205477	0.190320	1.000000	0.383348	-0.081355
Fe_ugl Iron	-0.504908	0.199683	-0.349182	0.222477	0.222477	-0.267522	0.735634	0.536218	0.383348	1.000000	0.024838
TN_ugl Nitrogen	0.332015	-0.267100	0.216412	-0.199251	-0.199251	-0.133760	0.148973	0.297481	-0.081355	0.024838	1.000000

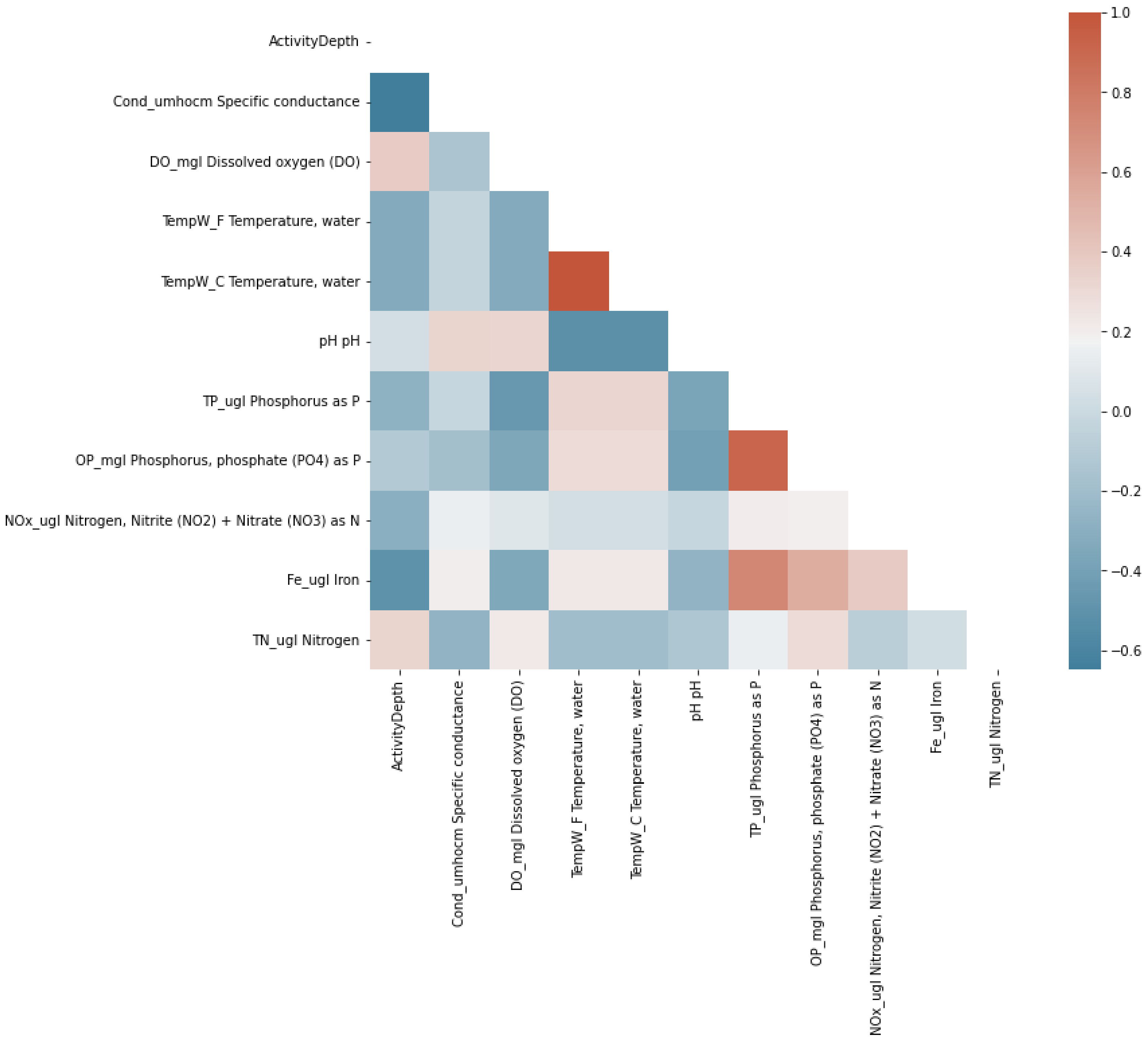
In [ ]:

```
#creating a triangle heat map, for better interpretation of the values.

mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, square=True)
```

Out[ ]:

```
<AxesSubplot:>
```



The correlated parameter pairs are:

Parameter 01	Parameter 02
1. ActivityDepth;	Fe_ugl Iron
2. Cond_umhocm Specific conductance;	ActivityDepth
3. TempW_F Temperature, water;	pH pH
4. TempW_C Temperature, wate;	pH pH
5. TP_ugl Phosphorus as P; OP_mgl Phosphorus,	phosphate (PO4) as P
6. Fe_ugl Iron;	TP_ugl Phosphorus as P
7. OP_mgl Phosphorus, phosphate (PO4) as P;	Fe_ugl Iron

Task 04

Yes, this data can answer these questions.

1. Most of the values which is required to obtain the quality of drinking water are directly present as a parameter or can be derieved from the calculated paramters present in the dataset. [https://www.ag.ndsu.edu/publications/environment-natural-resources/drinking-water-quality-testing-and-interpreting-your-results#section-6]

2. Most of the values which is required to obtain the quality of swimming water are directly present as a parameter or can be derieved from the calculated paramters present in the dataset. [https://www.betterhealth.vic.gov.au/health/healthyliving/swimming-pools-water-quality]

3. Most of the values which is required to obtain the quality of irrigation water are directly present as a parameter or can be derieved from the calculated paramters present in the dataset. [https://extension.psu.edu/interpreting-irrigation-water-tests]

Example of parameters: ActivityDepth; Cond_umhocm Specific conductance; DO_mgl Dissolved oxygen (DO); TempW_F Temperature, water; TempW_C Temperature, water; pH pH; TP_ugl Phosphorus as P; OP_mgl Phosphorus, phosphate (PO4) as P; NOx_ugl Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N; Fe_ugl Iron; TN_ugl Nitrogen

Drawback: The values of all the parameters mentioned above are not collected for all the mentioned 103 different dates.