dt_iso is in UTC format

In [15]:
```
import pandas as pd
```

In [16]:
```
ls *.csv
```

```
 Volume in drive C is OS
 Volume Serial Number is 300E-5120

 Directory of C:\Users\Nikhil\Data_Science_Projects\Weather

08/20/2023  05:40 PM        22,436,386 Jan_2008__Aug_2023_weather_data.csv
               1 File(s)     22,436,386 bytes
               0 Dir(s)  310,845,399,040 bytes free
```

In [17]:
```
weather = pd.read_csv('Jan_2008__Aug_2023_weather_data.csv')
```

In [ ]:
```
dt
```

In [18]:
```
weather.shape
```

Out[18]:
```
(137040, 28)
```

In [19]:
```
weather.columns
```

Out[19]:
```
Index(['dt', 'dt_iso', 'timezone', 'city_name', 'lat', 'lon', 'temp',
       'visibility', 'dew_point', 'feels_like', 'temp_min', 'temp_max',
       'pressure', 'sea_level', 'grnd_level', 'humidity', 'wind_speed',
       'wind_deg', 'wind_gust', 'rain_1h', 'rain_3h', 'snow_1h', 'snow_3h',
       'clouds_all', 'weather_id', 'weather_main', 'weather_description',
       'weather_icon'],
      dtype='object')
```

In [22]:
```
weather['temp'].describe()
```

Out[22]:
```
count    137040.000000
mean         66.304809
std          51.824186
min      -17966.380000
25%          53.380000
50%          68.430000
75%          79.990000
max         109.170000
Name: temp, dtype: float64
```

In [30]:
```
weather.columns
```

Out[30]:
```
Index(['dt', 'dt_iso', 'timezone', 'city_name', 'lat', 'lon', 'temp',
       'visibility', 'dew_point', 'feels_like', 'temp_min', 'temp_max',
       'pressure', 'sea_level', 'grnd_level', 'humidity', 'wind_speed',
       'wind_deg', 'wind_gust', 'rain_1h', 'rain_3h', 'snow_1h', 'snow_3h',
       'clouds_all', 'weather_id', 'weather_main', 'weather_description',
       'weather_icon'],
      dtype='object')
```

In [40]:
```
dt_iso_min_weather = weather[weather['temp'] < 0]['dt_iso'].values[0]
```

In [41]:
```
weather[weather['dt_iso'] ==  dt_iso_min_weather]
```

Out[41]:

	dt	dt_iso	timezone	city_name	lat	lon	temp	visibility	dew_p
136379	1690110000	2023-07-23 11:00:00 +0000 UTC	-18000	Frisco	33.150674	-96.823612	-17966.38	NaN	

1 rows × 28 columns

In [42]:
```
dt_iso_min_weather
```

Out[42]:
```
'2023-07-23 11:00:00 +0000 UTC'
```

In [ ]:
```
def get_day()
```

In [36]:
```
(weather['temp'] - weather['feels_like']).describe()
```

Out[36]:
```
count    137040.000000
mean         -0.067625
std           3.909149
min         -12.600000
25%          -1.280000
50%           0.020000
75%           1.940000
max          12.600000
dtype: float64
```

In [37]:
```
for i, c in enumerate(list(weather.columns)):
    x = weather[weather['temp'] < 0]
    print(f'column # {i}: {c}, value: {x[c].values[0]}')
```

```
column # 0: dt, value: 1690110000
column # 1: dt_iso, value: 2023-07-23 11:00:00 +0000 UTC
column # 2: timezone, value: -18000
column # 3: city_name, value: Frisco
column # 4: lat, value: 33.150674
column # 5: lon, value: -96.823612
column # 6: temp, value: -17966.38
column # 7: visibility, value: nan
column # 8: dew_point, value: nan
column # 9: feels_like, value: -17978.98
column # 10: temp_min, value: -17966.7
column # 11: temp_max, value: -462.1
column # 12: pressure, value: 1016
column # 13: sea_level, value: nan
column # 14: grnd_level, value: nan
column # 15: humidity, value: 0
column # 16: wind_speed, value: 4.25
column # 17: wind_deg, value: 148
column # 18: wind_gust, value: nan
column # 19: rain_1h, value: nan
column # 20: rain_3h, value: nan
column # 21: snow_1h, value: nan
column # 22: snow_3h, value: nan
column # 23: clouds_all, value: 0
column # 24: weather_id, value: 800
column # 25: weather_main, value: Clear
column # 26: weather_description, value: sky is clear
column # 27: weather_icon, value: 01n
```

In [24]:
```
min_weather = weather['temp'].min()
```

In [27]:
```
weather[weather['temp'] == min_weather]
```

Out[27]:

	dt	dt_iso	timezone	city_name	lat	lon	temp	visibility	dew_p
136379	1690110000	2023-07-23 11:00:00 +0000 UTC	-18000	Frisco	33.150674	-96.823612	-17966.38	NaN	

1 rows × 28 columns

In [20]:
```
weather['temp_max'].describe()
```

Out[20]:
```
count    137040.000000
mean         67.485638
std          17.779406
min        -462.100000
25%          54.540000
50%          69.390000
75%          81.010000
max         111.580000
Name: temp_max, dtype: float64
```

In [21]:
```
weather['temp_min'].describe()
```

Out[21]:
```
count    137040.000000
mean         65.210649
std          51.824568
min      -17966.700000
25%          52.270000
50%          67.280000
75%          78.960000
max         107.510000
Name: temp_min, dtype: float64
```

In [43]:
```
from datetime import datetime
date_created = "2016-10-22T16:27:54+0000"
datetime.strptime(date_created, '%Y-%m-%dT%H:%M:%S+0000')
#datetime.datetime(2016, 10, 22, 16, 27, 54)
```

Out[43]:
```
datetime.datetime(2016, 10, 22, 16, 27, 54)
```

In [45]:
```
# '2023-07-23 11:00:00 +0000 UTC'
datetime.strptime(dt_iso_min_weather, '%Y-%m-%d %H:%M:%S +0000 UTC')
```

Out[45]:
```
datetime.datetime(2023, 7, 23, 11, 0)
```

In [46]:
```
from dateutil import tz
```

In [65]:
```
# GO FROM UTC time stamp string to CST time stamp string
def utc_to_cst_timestamp_string(ts):
    from_zone = tz.gettz('UTC')
    to_zone = tz.gettz('America/Chicago')
    json_data = {'time': str(ts)}
    utc = datetime.strptime(json_data['time'], '%Y-%m-%d %H:%M:%S +0000 UTC')
    utc = utc.replace(tzinfo=from_zone)
    cst = utc.astimezone(to_zone)
    date_time = cst.strftime("%Y-%m-%d %H:%M:%S")
    return date_time
```

In [53]:
```
weather.columns
```

Out[53]:
```
Index(['dt', 'dt_iso', 'timezone', 'city_name', 'lat', 'lon', 'temp',
       'visibility', 'dew_point', 'feels_like', 'temp_min', 'temp_max',
       'pressure', 'sea_level', 'grnd_level', 'humidity', 'wind_speed',
       'wind_deg', 'wind_gust', 'rain_1h', 'rain_3h', 'snow_1h', 'snow_3h',
       'clouds_all', 'weather_id', 'weather_main', 'weather_description',
       'weather_icon'],
      dtype='object')
```

In [ ]:
```
"""
 2007-12-31 18:00:00
1         2007-12-31 19:00:00
2         2007-12-31 20:00:00
3         2007-12-31 21:00:00
4         2007-12-31 22:00:00
"""
```

In [66]:
```
%%time
weather['cst_timestamp'] = weather['dt_iso'].apply(lambda x: utc_to_cst_timestamp_stri
weather['cst_timestamp']
```

```
CPU times: total: 8.98 s
Wall time: 13.8 s
```

Out[66]:
```
0         2007-12-31 18:00:00
1         2007-12-31 19:00:00
2         2007-12-31 20:00:00
3         2007-12-31 21:00:00
4         2007-12-31 22:00:00
                 ...
137035    2023-08-19 14:00:00
137036    2023-08-19 15:00:00
137037    2023-08-19 16:00:00
137038    2023-08-19 17:00:00
137039    2023-08-19 18:00:00
Name: cst_timestamp, Length: 137040, dtype: object
```

In [70]:
```
%%time
weather['date'] = weather['cst_timestamp'].apply(lambda x: x.split(' ')[0])
weather['time'] = weather['cst_timestamp'].apply(lambda x: x.split(' ')[1])
```

```
CPU times: total: 62.5 ms
Wall time: 207 ms
```

In [76]:
```
july_1_2023 = weather[weather['date'] == '2023-07-01']
```

In [79]:
```
july_1_2023['temp'].max(), july_1_2023['temp_max'].max()
```

Out[79]:
```
(87.33, 89.1)
```

In [82]:
```
in_scope = weather[weather['date'] >= '2023-05-01']
max_weather = in_scope.groupby('date')['temp'].max()
```

In [84]:
```
max_weather.index
```

Out[84]:
```
Index(['2023-05-01', '2023-05-02', '2023-05-03', '2023-05-04', '2023-05-05',
       '2023-05-06', '2023-05-07', '2023-05-08', '2023-05-09', '2023-05-10',
       ...
       '2023-08-10', '2023-08-11', '2023-08-12', '2023-08-13', '2023-08-14',
       '2023-08-15', '2023-08-16', '2023-08-17', '2023-08-18', '2023-08-19'],
      dtype='object', name='date', length=111)
```

In [86]:
```
import matplotlib.pyplot as plt
```

In [101…
```
type(ax)
```

Out[101]:
```
matplotlib.axes._subplots.AxesSubplot
```

In [103…
```
original_labels = [str(label) for label in ax.get_xticks()]
original_labels
```

Out[103]:
```
['0',
 '1',
 '2',
 '3',
 '4',
 '5',
 '6',
 '7',
 '8',
 '9',
 '10',
 '11',
 '12',
 '13',
 '14',
 '15',
 '16',
 '17',
 '18',
 '19',
 '20',
 '21',
 '22',
 '23',
 '24',
 '25',
 '26',
 '27',
 '28',
 '29',
 '30',
 '31',
 '32',
 '33',
 '34',
 '35',
 '36',
 '37',
 '38',
 '39',
 '40',
 '41',
 '42',
 '43',
 '44',
 '45',
 '46',
 '47',
 '48',
 '49',
 '50',
 '51',
 '52',
 '53',
 '54',
 '55',
 '56',
 '57',
 '58',
 '59',
```

```
 '60',
 '61',
 '62',
 '63',
 '64',
 '65',
 '66',
 '67',
 '68',
 '69',
 '70',
 '71',
 '72',
 '73',
 '74',
 '75',
 '76',
 '77',
 '78',
 '79',
 '80',
 '81',
 '82',
 '83',
 '84',
 '85',
 '86',
 '87',
 '88',
 '89',
 '90',
 '91',
 '92',
 '93',
 '94',
 '95',
 '96',
 '97',
 '98',
 '99',
 '100',
 '101',
 '102',
 '103',
 '104',
 '105',
 '106',
 '107',
 '108',
 '109',
 '110']
```

In [114…

```
[max_weather.index[i] for i in range(0, len(ax.get_xticks()), 5)]
```

Out[114]:
```
['2023-05-01',
 '2023-05-06',
 '2023-05-11',
 '2023-05-16',
 '2023-05-21',
 '2023-05-26',
 '2023-05-31',
 '2023-06-05',
 '2023-06-10',
 '2023-06-15',
 '2023-06-20',
 '2023-06-25',
 '2023-06-30',
 '2023-07-05',
 '2023-07-10',
 '2023-07-15',
 '2023-07-20',
 '2023-07-25',
 '2023-07-30',
 '2023-08-04',
 '2023-08-09',
 '2023-08-14',
 '2023-08-19']
```

In [121…
```
import matplotlib
```

In [122…
```
matplotlib.colors.BASE_COLORS
```

Out[122]:
```
{'b': (0, 0, 1),
 'g': (0, 0.5, 0),
 'r': (1, 0, 0),
 'c': (0, 0.75, 0.75),
 'm': (0.75, 0, 0.75),
 'y': (0.75, 0.75, 0),
 'k': (0, 0, 0),
 'w': (1, 1, 1)}
```

In [ ]:
```
matplotlib.colors.TABLEAU_COLORS
```

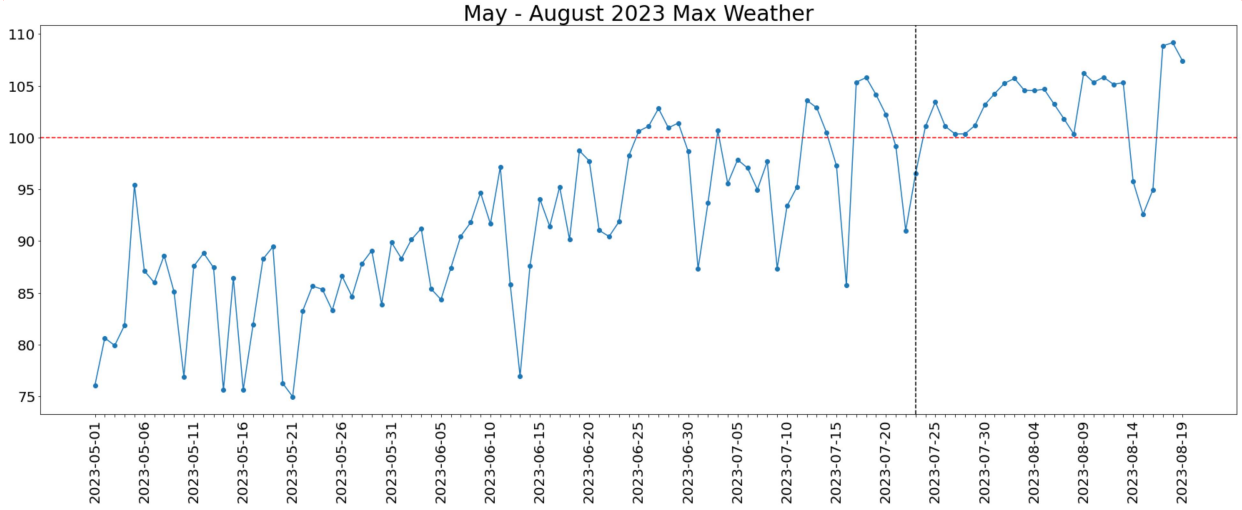
In [ ]:
```

```

In [137…
```
fig, ax = plt.subplots()
ax.plot_date(max_weather.index, max_weather, linestyle='solid')
original_labels = [str(label) for label in ax.get_xticks()]
labels_of_interest = [str(i) for i in np.arange(0,len(original_labels),5)]
new_labels = [str(max_weather.index[int(label)]) if label in labels_of_interest else '
ax.set_xticklabels(new_labels)
plt.xticks(rotation=90)
plt.yticks(fontsize=20)
plt.xticks(fontsize=20)
plt.title('May - August 2023 Max Weather', fontsize=30)
plt.axhline(y = 100, color = 'r', linestyle = 'dashed')
plt.axvline(x = 83, color = 'k', linestyle = 'dashed')

plt.show()
#plt.savefig()
```

```
C:\Users\Nikhil\AppData\Local\Temp\ipykernel_26312\3342501139.py:6: UserWarning: Fixe
dFormatter should only be used together with FixedLocator
  ax.set_xticklabels(new_labels)
```



In [143…
```
max_weather.head(10)
```

Out[143]:
```
date
2023-05-01    76.08
2023-05-02    80.65
2023-05-03    79.92
2023-05-04    81.88
2023-05-05    95.43
2023-05-06    87.13
2023-05-07    86.00
2023-05-08    88.61
2023-05-09    85.12
2023-05-10    76.91
Name: temp, dtype: float64
```

In [144…
```
(76.08 + 80.65 + 79.92)/3
```

Out[144]:
```
78.88333333333334
```

In [146…
```
(81.88 + 95.43 + 87.13)/3
```

Out[146]:
```
88.14666666666666
```

In [155…
```
three_day_average = max_weather.groupby(np.arange(len(max_weather))//3).mean()
three_day_average.index = [max_weather.index[i] for i in range(0, max_weather.shape[0]
three_day_average
```

Out[155]:
```
2023-05-01     78.883333
2023-05-04     88.146667
2023-05-07     86.576667
2023-05-10     84.443333
2023-05-13     83.180000
2023-05-16     81.950000
2023-05-19     80.233333
2023-05-22     84.746667
2023-05-25     84.860000
2023-05-28     86.936667
2023-05-31     89.443333
2023-06-03     86.990000
2023-06-06     89.900000
2023-06-09     94.510000
2023-06-12     83.460000
2023-06-15     93.576667
2023-06-18     95.543333
2023-06-21     91.120000
2023-06-24     99.986667
2023-06-27    101.720000
2023-06-30     93.223333
2023-07-03     98.036667
2023-07-06     96.583333
2023-07-09     91.996667
2023-07-12    102.310000
2023-07-15     96.116667
2023-07-18    104.053333
2023-07-21     95.576667
2023-07-24    101.866667
2023-07-27    100.626667
2023-07-30    104.226667
2023-08-02    104.936667
2023-08-05    103.243333
2023-08-08    103.963333
2023-08-11    105.430000
2023-08-14     94.436667
2023-08-17    108.476667
Name: temp, dtype: float64
```

In [151…
```
three_day_average.shape
```

Out[151]:
```
(37,)
```

In [147…
```
max_weather.groupby(np.arange(len(max_weather))//5).mean()
```

Out[147]:
```
0      82.792
1      84.754
2      85.192
3      82.314
4      82.508
5      86.414
6      88.988
7      89.748
8      87.846
9      93.924
10     93.868
11    101.376
12     95.184
13     94.996
14     99.110
15     99.658
16     98.006
17    101.280
18    104.590
19    102.920
20    105.570
21    100.268
22    107.400
Name: temp, dtype: float64
```

In [128…
```
max_weather
```

Out[128]:
```
date
2023-05-01     76.08
2023-05-02     80.65
2023-05-03     79.92
2023-05-04     81.88
2023-05-05     95.43
               ...
2023-08-15     92.59
2023-08-16     94.96
2023-08-17    108.86
2023-08-18    109.17
2023-08-19    107.40
Name: temp, Length: 111, dtype: float64
```

In [136…
```
(76.08 + 80.65 + 79.92)/3
```

Out[136]:
```
78.88333333333334
```

In [134…
```
max_weather_rolling_mean = max_weather.rolling(3).mean()
max_weather_rolling_mean
```

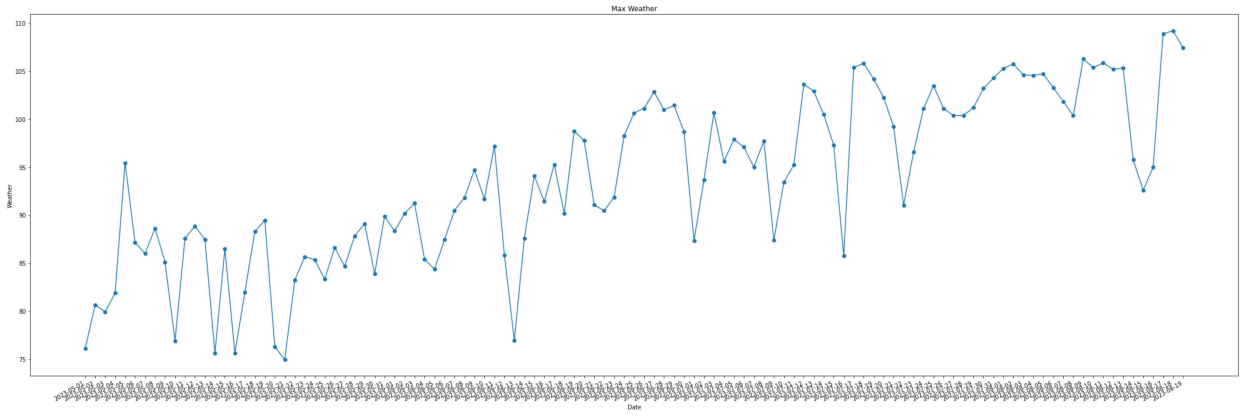
Out[134]:
```
date
2023-05-01           NaN
2023-05-02           NaN
2023-05-03     78.883333
2023-05-04     80.816667
2023-05-05     85.743333
                  ...
2023-08-15     97.883333
2023-08-16     94.436667
2023-08-17     98.803333
2023-08-18    104.330000
2023-08-19    108.476667
Name: temp, Length: 111, dtype: float64
```

In [93]:
```
ax = plt.plot_date(max_weather.index, max_weather, linestyle='solid')

plt.gcf().autofmt_xdate()

plt.title('Max Weather')
plt.xlabel('Date')
plt.ylabel('Weather')

plt.tight_layout()
plt.rcParams["figure.figsize"] = (30, 10)
plt.show()
```



In [ ]:
```
cst_datetime = datetime.strptime(cst, '%Y-%m-%d %H:%M:%S-05:00 UTC')
print(utc)
print(cst)
```

In [94]:
```
ax.get_xticks()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [94], in <cell line: 1>()
----> 1 ax.get_xticks()

AttributeError: 'list' object has no attribute 'get_xticks'
```

In [ ]:
```
original_labels = [str(label) for label in ax.get_xticks()]
labels_of_interest = [str(i) for i in np.arange(235,295,5)]
new_labels = [label if label in labels_of_interest else "" for label in original_label
ax.set_xticklabels(new_labels)
```

In [92]:
```
ax.set_xticks(np.arange(0, 800, 1))
ax.grid()
```

```
original_labels = [str(label) for label in ax.get_xticks()]
labels_of_interest = [str(i) for i in np.arange(235,295,5)]
new_labels = [label if label in labels_of_interest else "" for label in original_label
ax.set_xticklabels(new_labels)
```

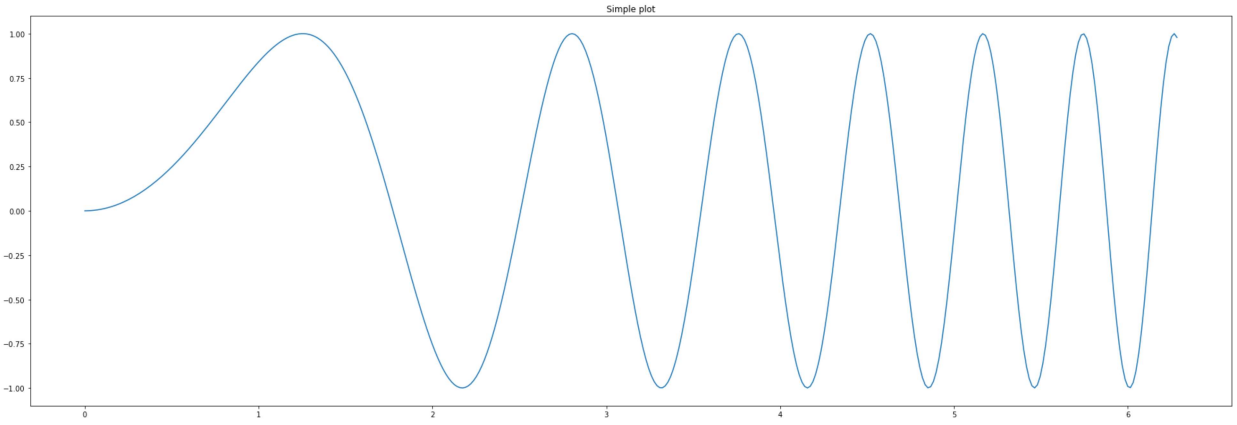
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [92], in <cell line: 1>()
----> 1 ax.set_xticks(np.arange(0, 800, 1))
      2 ax.grid()
      4 original_labels = [str(label) for label in ax.get_xticks()]

NameError: name 'ax' is not defined
```

In [96]:
```
import numpy as np
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)

# Create just a figure and only one subplot
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('Simple plot')
```

Out[96]:
```
Text(0.5, 1.0, 'Simple plot')
```



In [ ]: