In [1]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\samil\anaconda3\lib\site-packages (2.0.3)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: pytz>=2020.1 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.20.3; python_version < "3.10" in c:\users\samil\anaconda3\lib\site-packages (from pandas) (1.24.4)
Requirement already satisfied: six>=1.5 in c:\users\samil\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.15.0)
```

In [2]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\samil\anaconda3\lib\site-packages (5.18.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\samil\anaconda3\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\samil\anaconda3\lib\site-packages (from plotly) (20.4)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\samil\anaconda3\lib\site-packages (from packaging->plotly) (2.4.7)
Requirement already satisfied: six in c:\users\samil\anaconda3\lib\site-packages (from packaging->plotly) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
C:\Users\Samil\anaconda3\lib\site-packages\pandas\core\computation\expressions.py:20: UserWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' current
ly installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [14]:
```
raw_dataset=pd.read_csv("MarmaraAll.csv",sep=",")
```

In [15]:
```
 MarmaraAll= raw_dataset.copy()
MarmaraAll.head()
```

Out[15]:

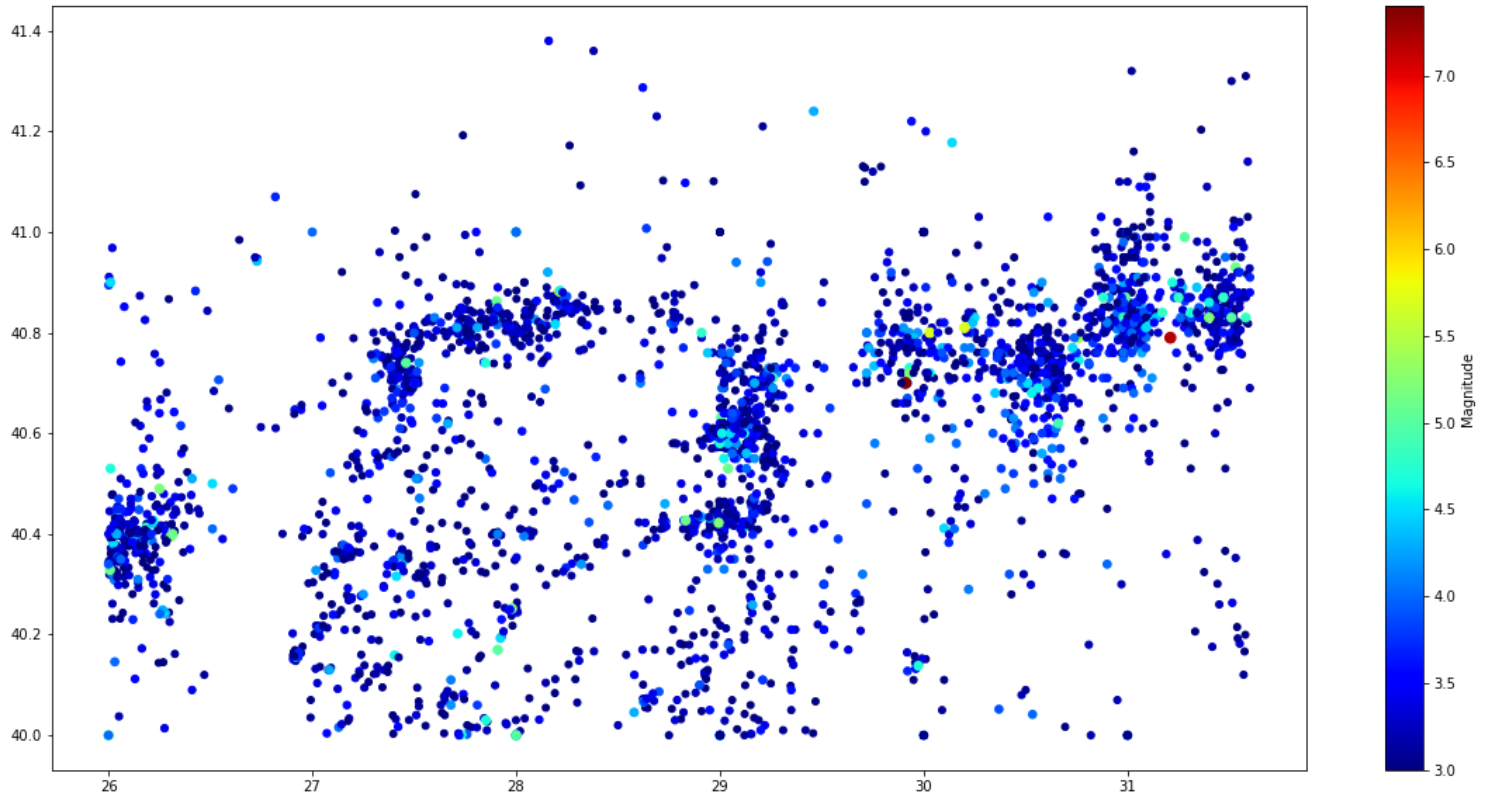
	Date	Longitude	Latitude	Depth	Magnitude
0	10/01/2024 17:10:06	28.8706	40.4117	11.20	3.6
1	09/01/2024 01:56:28	28.8386	40.4133	10.63	3.1
2	30/12/2023 20:43:24	29.1583	40.2431	9.07	3.4
3	24/12/2023 15:18:56	28.8339	40.4225	10.44	3.0
4	21/12/2023 14:03:08	27.4839	40.7247	10.40	3.1

In [33]:
```
MarmaraAll.shape
```

Out[33]:
```
(2836, 5)
```

In [34]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z = MarmaraAll.iloc[:,3].values
colors = MarmaraAll.iloc[:,4].values
sizes = MarmaraAll.iloc[:,4].values*8
```

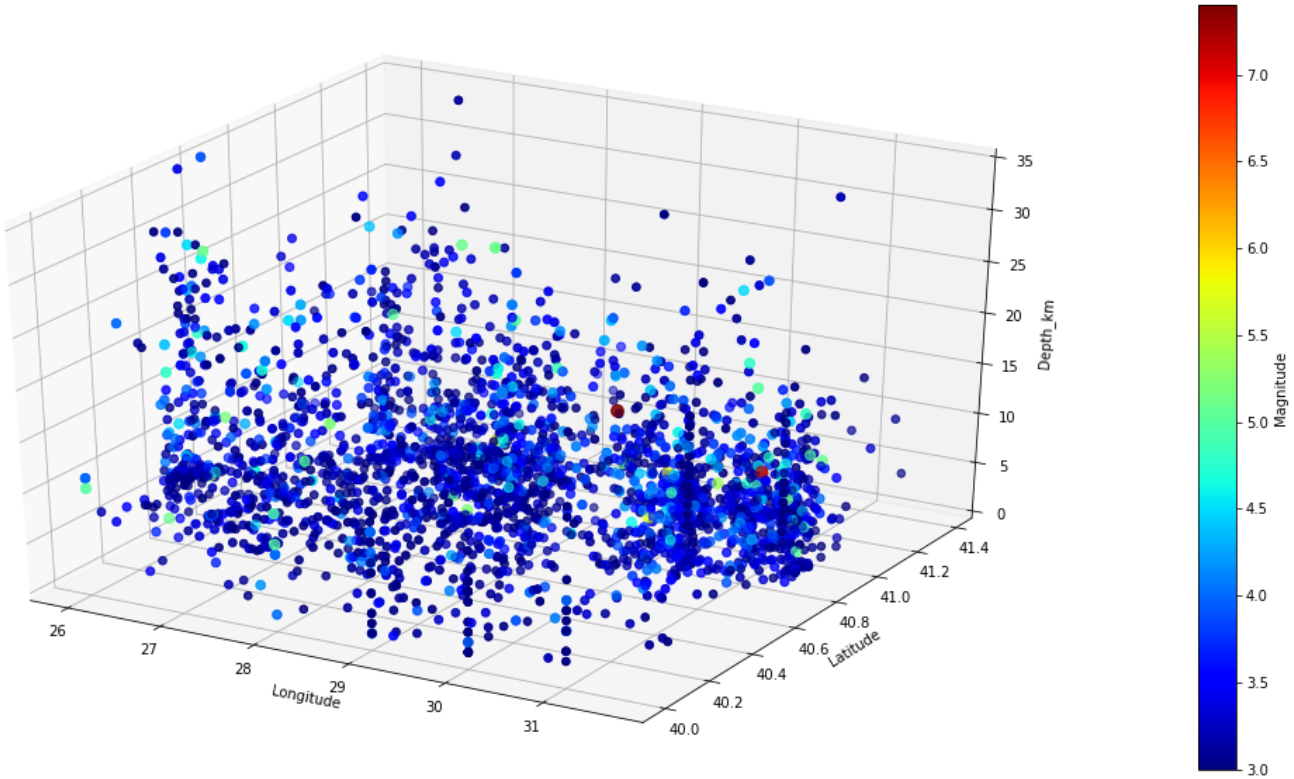
In [35]:
```
fig = plt.figure(figsize=(20, 10))
my_cmap = plt.get_cmap('jet')

plt.scatter(x, y, c=colors, s=sizes, cmap= 'jet')
ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet')
cbar = plt.colorbar(ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet'))
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
cbar.set_label('Magnitude')
ax.set_zlabel('Depth_km')
font_size = 700
dpi = (5000)
font_size = 1000
plt.show()
```



In [19]:
```
import plotly.graph_objects as go
# Yüksek çözünürlüklü dünya haritası verilerini çevrimiçi olarak alın
fig = go.Figure(go.Choroplethmapbox(
 geojson="https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json",
 locations=["USA", "CAN", "MEX", "RUS", "CHN"], # Örnek ülke kodları (ABD, Kanada, Meksika, Rusya, Çin)
 z=[1, 1, 1, 1, 1], # Ülkelere atanacak değerler (hepsi 1 olarak ayarlanmıştır)
 colorscale='Jet', # Renk skalası adı (Viridis, YlGnBu, Jet vb.)
 zmin=3,
 zmax=8,
 marker_opacity=0.9, # Ülke sınırlarının opaklığı
 marker_line_width=1, # Ülke sınırlarının kenarlık kalınlığı
))
# Örnek deprem verilerini oluşturun
deprem_verileri = {
 'Longitude': x,
 'Latitude': y,
 'Magnitude': colors,
 }
# Scatter plot ile deprem verilerini ekleyin
fig.add_trace(go.Scattermapbox(
 lat=deprem_verileri['Latitude'],
 lon=deprem_verileri['Longitude'],
 mode='markers',
 marker=dict(
 size=deprem_verileri['Magnitude'] * 2, # Magnitude değerine göre nokta boyutlarını belirleme
 color=deprem_verileri['Magnitude'], # Magnitude değerine göre renk skalasını belirleme
 colorscale='Jet', # Renk skalası adı (Viridis, YlGnBu, Jet vb.)
 ),
 ))
# Harita düzenini ve stilini belirleyin
fig.update_layout(
 mapbox_style="carto-positron", # Harita stilini belirleme (diğer stiller için: "open-street-map", "stamen-terrain" vb.)
 mapbox_zoom=6, # Harita yakınlaştırma düzeyini belirleme
 mapbox_center={"lat": 30.000, "lon": 30.0000}, # Harita merkezini belirleme (ABD'nin merkezi)
)
dpi = (9000)
font_size = 1000
# Grafiği görüntüleyin
fig.show()
```

In [20]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z = MarmaraAll.iloc[:,3].values
colors = MarmaraAll.iloc[:,4].values
sizes = MarmaraAll.iloc[:,4].values*10
```

In [21]:
```
fig = plt.figure(figsize=(20, 10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet')


cbar = plt.colorbar(ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet'))

cbar.set_label('Magnitude')


ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Depth_km')




dpi = (5000)
font_size = 1000

ax.set_zlim(0, 35)


plt.show()
```



In [22]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z = MarmaraAll.iloc[:,3].values
colors = MarmaraAll.iloc[:,4].values
sizes = MarmaraAll.iloc[:,4].values*2
```

In [23]:
```
import plotly.graph_objs as go
import numpy as np

# Veri oluşturma (x, y, z, colors, sizes tanımlanmış olarak varsayıldı)

trace = go.Scatter3d(
    x=x,
    y=y,
    z=z,
    mode='markers',
    marker=dict(
        size=sizes,
        color=colors,
        colorscale='Jet',
        opacity=0.5,
        colorbar=dict(title='Magnitude')
    )
)

layout = go.Layout(
    scene=dict(
        xaxis=dict(title='Longitude'),
        yaxis=dict(title='Latitude'),
        zaxis=dict(title='Depth_km'),
        aspectmode='manual',
        aspectratio=dict(x=1, y=2, z=1),
        camera=dict(eye=dict(x=2, y=1, z=1))
    ),
    coloraxis=dict(colorbar=dict(len=0.75))
)

fig = go.Figure(data=[trace], layout=layout)

# Grafik döndürme
frames = []
for angle in np.linspace(0, 360, 36):
    frame = go.Frame(layout=dict(scene=dict(camera=dict(eye=dict(x=2*np.cos(np.radians(angle)), y=2*np.sin(np.radians(angle)), z=2)))))
    frames.append(frame)

fig.frames = frames
fig.update_layout(updatemenus=[dict(type='buttons', showactive=False, buttons=[dict(label='Play', method='animate', args=[None, dict(frame=dict(duration=200, redraw=True), fromcu

fig.show()
```

In [24]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z = MarmaraAll.iloc[:,0].values
colors = MarmaraAll.iloc[:,4].values+1
sizes = MarmaraAll.iloc[:,4].values*10
```

In [25]:
```
z
```

Out[25]:
```
array(['10/01/2024 17:10:06', '09/01/2024 01:56:28',
       '30/12/2023 20:43:24', ..., '27/01/1990 14:30:47',
       '04/01/1990 11:32:14', '03/01/1990 13:30:14'], dtype=object)
```

In [26]:
```
from datetime import datetime
old_dates = z
old_format = '%d/%m/%Y %H:%M:%S'
new_format = '%Y-%m-%d %H:%M:%S'
new_dates = []
for date_str in old_dates:
 new_date = datetime.strptime(date_str, old_format).strftime(new_format)
 new_dates.append(new_date)
print(new_dates)
```

```
['2024-01-10 17:10:06', '2024-01-09 01:56:28', '2023-12-30 20:43:24', '2023-12-24 15:18:56', '2023-12-21 14:03:08', '2023-12-17 20:53:52', '2023-12-14 09:50:16', '2023-12-13
20:04:38', '2023-12-08 23:21:05', '2023-12-06 10:01:44', '2023-12-04 14:46:13', '2023-12-04 09:25:03', '2023-12-04 08:18:53', '2023-12-04 07:57:18', '2023-12-04 07:45:33', '2
023-12-04 07:42:20', '2023-11-07 20:05:48', '2023-10-26 17:18:24', '2023-10-15 16:57:33', '2023-10-13 19:54:07', '2023-10-11 16:36:01', '2023-09-13 07:45:09', '2023-08-19 14:
43:46', '2023-08-06 15:29:01', '2023-07-27 08:32:50', '2023-06-16 22:19:59', '2023-06-05 10:11:30', '2023-06-04 03:23:47', '2023-06-03 11:07:51', '2023-06-02 00:52:42', '2023
-05-26 05:59:57', '2023-05-26 05:59:31', '2023-05-09 07:20:49', '2023-05-04 01:50:00', '2023-04-30 11:01:53', '2023-04-30 04:02:54', '2023-04-20 10:41:35', '2023-04-09 00:40:
32', '2023-03-17 11:01:53', '2023-03-17 11:00:37', '2023-03-16 17:18:15', '2023-03-16 16:36:01', '2023-03-16 13:04:46', '2023-03-16 11:04:59', '2023-03-16 10:55:06', '2023-03
-09 05:28:53', '2023-03-07 08:36:30', '2023-03-05 05:28:07', '2023-02-24 02:08:23', '2023-02-06 04:24:32', '2023-02-05 12:49:58', '2023-02-05 00:15:10', '2023-02-02 20:25:4
3', '2023-02-02 20:08:41', '2023-01-30 10:14:04', '2023-01-17 19:38:47', '2023-01-10 06:15:12', '2023-01-07 07:55:49', '2023-01-02 09:26:32', '2022-12-30 10:29:03', '2022-12-
13 03:21:16', '2022-12-12 12:53:43', '2022-12-04 13:38:01', '2022-12-03 23:33:06', '2022-12-03 04:59:20', '2022-12-02 21:09:00', '2022-12-02 18:49:36', '2022-12-02 10:03:18',
'2022-11-27 03:57:46', '2022-11-25 22:01:07', '2022-11-25 09:00:33', '2022-11-24 01:02:09', '2022-11-23 03:50:58', '2022-11-23 01:30:34', '2022-11-23 01:08:15', '2022-10-27 2
3:02:29', '2022-10-13 14:25:51', '2022-10-03 22:57:50', '2022-09-29 00:29:25', '2022-09-14 00:38:37', '2022-08-11 19:23:01', '2022-07-28 16:10:48', '2022-07-22 15:55:30', '20
22-07-21 15:44:23', '2022-07-20 19:49:40', '2022-07-19 18:51:33', '2022-04-30 03:35:54', '2022-04-07 12:36:56', '2022-04-07 12:27:45', '2022-04-05 14:49:45', '2022-03-28 13:4
7:06', '2022-03-22 14:10:55', '2022-03-22 04:22:09', '2022-02-20 20:20:11', '2022-02-06 12:07:53', '2022-01-22 11:13:45', '2022-01-19 02:41:48', '2022-01-17 12:06:49', '2021-
12-29 14:46:17', '2021-11-17 18:35:42', '2021-11-17 12:57:18', '2021-11-17 12:40:15', '2021-09-23 05:57:59', '2021-09-17 09:22:20', '2021-09-07 14:15:58', '2021-07-16 12:29:2
3', '2021-07-12 21:27:35', '2021-06-19 12:07:52', '2021-06-01 07:32:37', '2021-05-05 06:42:11', '2021-04-18 04:33:22', '2021-03-21 21:33:56', '2021-02-27 12:12:06', '2021-02-
11 07:36:25', '2020-12-06 11:30:00', '2020-12-03 02:38:13', '2020-11-20 09:52:54', '2020-11-20 04:20:41', '2020-11-05 05:46:45', '2020-11-04 11:49:35', '2020-10-15 13:55:13',
'2020-09-24 18:06:40', '2020-09-24 13:38:31', '2020-09-08 21:57:23', '2020-07-22 21:14:25', '2020-07-14 20:59:37', '2020-07-04 03:13:22', '2020-06-04 12:14:49', '2020-04-01 1
9:03:24', '2020-03-08 04:21:04', '2020-02-22 01:25:56', '2020-02-02 00:57:43', '2020-01-28 09:47:32', '2020-01-11 13:37:36', '2020-01-11 05:15:26', '2020-01-02 19:35:47', '20
19-12-24 19:35:03', '2019-12-06 22:36:16', '2019-11-22 14:36:42', '2019-11-17 23:23:31', '2019-10-29 15:38:41', '2019-10-27 10:18:46', '2019-10-19 07:34:04', '2019-10-10 19:3
```

In [27]:
```
from datetime import datetime
date_times = new_dates
single_numbers = []
for date_time_str in date_times:
 date_time_obj = datetime.strptime(date_time_str, '%Y-%m-%d %H:%M:%S')
 single_number = int(date_time_obj.strftime('%Y%m%d%H%M%S'))
 single_numbers.append(single_number)
print(single_numbers)
```

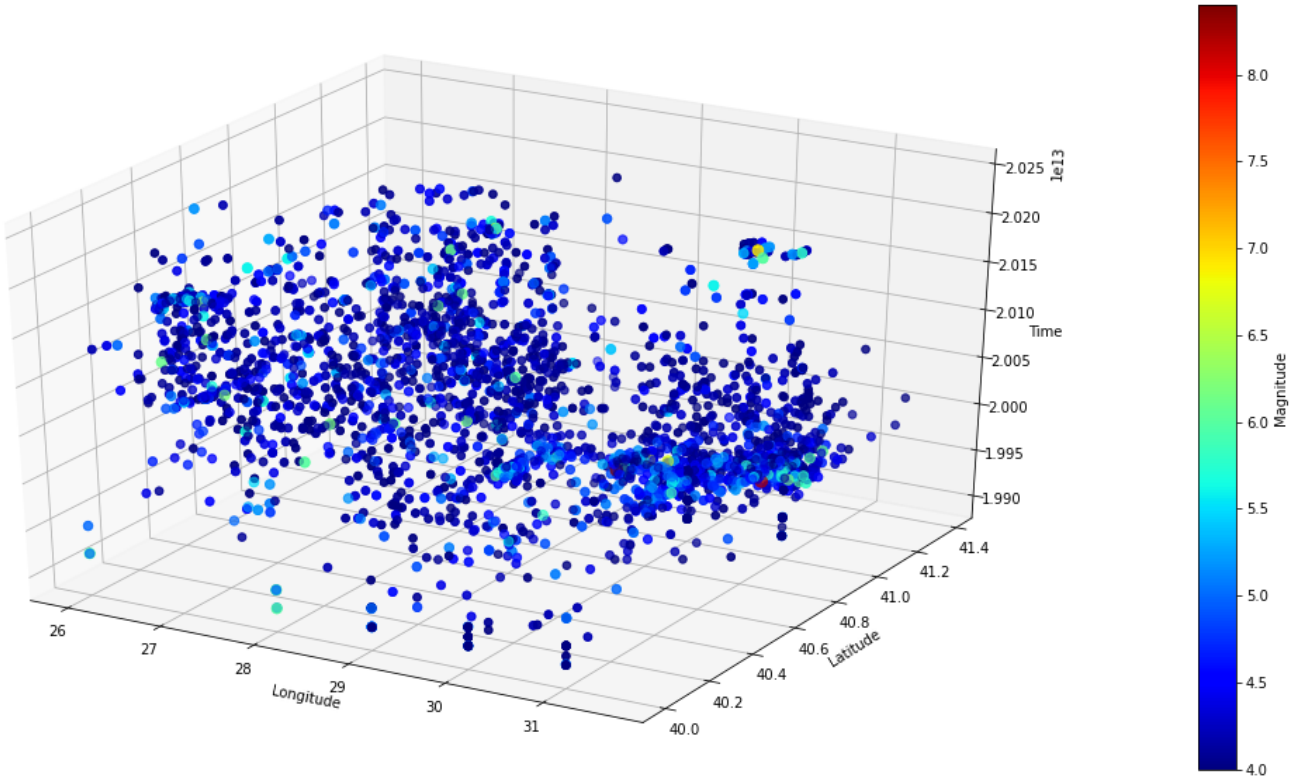
```
[20240110171006, 20240109015628, 20231230204324, 20231224151856, 20231221140308, 20231217205352, 20231214095016, 20231213200438, 20231208232105, 20231206100144, 2023120414461
3, 20231204092503, 20231204081853, 20231204075718, 20231204074533, 20231204074220, 20231107200548, 20231026171824, 20231015165733, 20231013195407, 20231011163601, 20230913074
509, 20230819144346, 20230806152901, 20230727083250, 20230616221959, 20230605101130, 20230604032347, 20230603110751, 20230602005242, 20230526055957, 20230526055931, 202305090
72049, 20230504015000, 20230430110153, 20230430040254, 20230420104135, 20230409004032, 20230317110153, 20230317110037, 20230316171815, 20230316163601, 20230316130446, 2023031
6110459, 20230316105506, 20230309052853, 20230307083630, 20230305052807, 20230224020823, 20230206042432, 20230205124958, 20230205001510, 20230202202543, 20230202200841, 20230
130101404, 20230117193847, 20230110061512, 20230107075549, 20230102092632, 20221230102903, 20221213032116, 20221212125343, 20221204133801, 20221203233306, 20221203045920, 202
21202210900, 20221202184936, 20221202100318, 20221127035746, 20221125220107, 20221125090033, 20221124010209, 20221123035058, 20221123013034, 20221123010815, 20221027230229, 2
0221013142551, 20221003225750, 20220929002925, 20220914003837, 20220811192301, 20220728161048, 20220722155530, 20220721154423, 20220720194940, 20220719185133, 20220430033554,
20220407123656, 20220407122745, 20220405144945, 20220328134706, 20220322141055, 20220322042209, 20220220202011, 20220206120753, 20220122111345, 20220119024148, 2022011712064
9, 20211229144617, 20211117183542, 20211117125718, 20211117124015, 20210923055759, 20210917092220, 20210907141558, 20210716122923, 20210712212735, 20210619120752, 20210601073
237, 20210505064211, 20210418043322, 20210321213356, 20210227121206, 20210211073625, 20201206113000, 20201203023813, 20201120095254, 20201120042041, 20201105054645, 202011041
14935, 20201015135513, 20200924180640, 20200924133831, 20200908215723, 20200722211425, 20200714205937, 20200704031322, 20200604121449, 20200401190324, 20200308042104, 2020022
2012556, 20200202005743, 20200128094732, 20200111133736, 20200111051526, 20200102193547, 20191224193503, 20191206223616, 20191122143642, 20191117232331, 20191029153841, 20191
027101846, 20191019073404, 20191010193206, 20191010170939, 20191010170438, 20191010165203, 20191009062739, 20191004155431, 20190929061057, 20190928142839, 20190928110303, 201
90927133547, 20190927111346, 20190927103230, 20190926202018, 20190926200240, 20190926153921, 20190926135701, 20190926133343, 20190926125859, 20190926122610, 20190926121709, 2
0190926121505, 20190926113402, 20190926113152, 20190926112930, 20190926112636, 20190926111134, 20190926110859, 20190926105925, 20190926073207, 20190925130758, 20190924085722,
20190924080022, 20190924073049, 20190823065821, 20190820233607, 20190719134928, 20190711201855, 20190709102753, 20190621034122, 20190614151143, 20190612224450, 2019060213084
7, 20190525121335, 20190521131737, 20190509160528, 20190509155623, 20190412130347, 20190323151759, 20190323025111, 20190304210727, 20190224012224, 20190224010540, 20190222211
934, 20190220042419, 20190220012740, 20190220003046, 20190220002757, 20190219213355, 20190219212942, 20190219194842, 20190215161427, 20190206193658, 20190111160956, 201901050
```

In [28]:
```
timestamps=single_numbers
```

In [29]:
```
timestamps
```

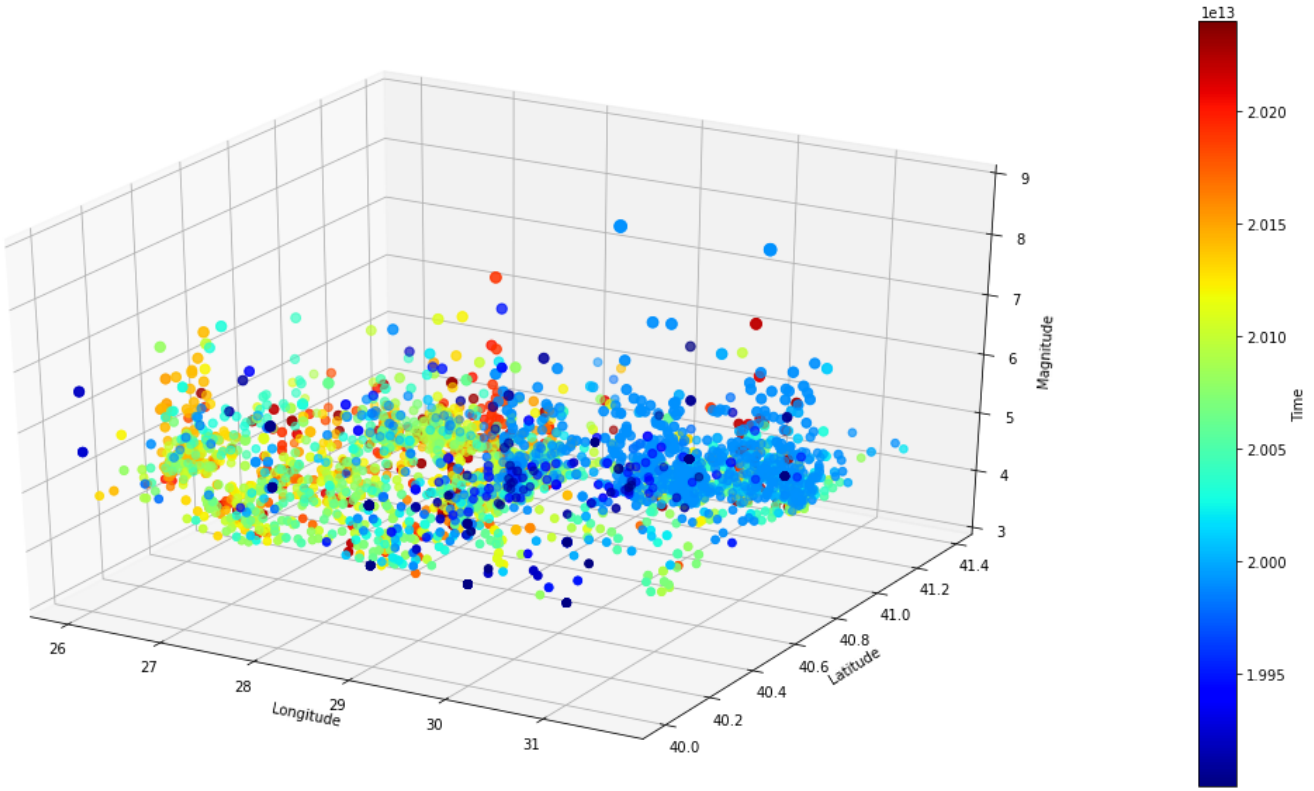
Out[29]:
```
[20240110171006,
 20240109015628,
 20231230204324,
 20231224151856,
 20231221140308,
 20231217205352,
 20231214095016,
 20231213200438,
 20231208232105,
 20231206100144,
 20231204144613,
 20231204092503,
 20231204081853,
 20231204075718,
 20231204074533,
 20231204074220,
 20231107200548,
 20231026171824,
 20231015165733,
```

In [30]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z =  timestamps
colors =  MarmaraAll.iloc[:,4].values+1
sizes = MarmaraAll.iloc[:,4].values*10

fig = plt.figure(figsize=(20, 10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet')


cbar = plt.colorbar(ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet'))

cbar.set_label('Magnitude')


ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Time')

font_size = 700

dpi = (5000)
font_size = 1000




plt.show()
```



In [31]:
```
x = MarmaraAll.iloc[:,1].values
y = MarmaraAll.iloc[:,2].values
z =  MarmaraAll.iloc[:,4].values+1.6
colors =  timestamps
sizes = MarmaraAll.iloc[:,4].values*10
```

In [32]:
```
fig = plt.figure(figsize=(20, 10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet')


cbar = plt.colorbar(ax.scatter(x, y, z, c=colors, s=sizes, cmap='jet'))

cbar.set_label('Time')


ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Magnitude')

font_size = 700

dpi = (5000)
font_size = 1000

ax.set_zlim(3,9)


plt.show()
```


Longitude
Latitude
Magnitude
26
27
28
29
30
31
40.0
40.2
40.4
40.6
40.8
41.0
41.2
41.4
3
4
5
6
7
8
9

1e13
2.020
2.015
2.010
2.005
2.000
1.995
Time

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: