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

```
Requirement already satisfied: pandas in c:\users\samil\anaconda3\lib\site-packages (2.0.3)
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: pytz>=2020.1 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: tzdata>=2022.1 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\samil\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

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

```
Requirement already satisfied: plotly in c:\users\samil\anaconda3\lib\site-packages (5.18.0)Note: you may need to restart the kernel to use updated packages.

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)
```

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' currently install
ed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [4]:
```
raw_dataset=pd.read_csv("Doublet_EAF_35.csv",sep=",")
```

In [5]:
```
 Doublet_EAF_35= raw_dataset.copy()
Doublet_EAF_35.head()
```

Out[5]:

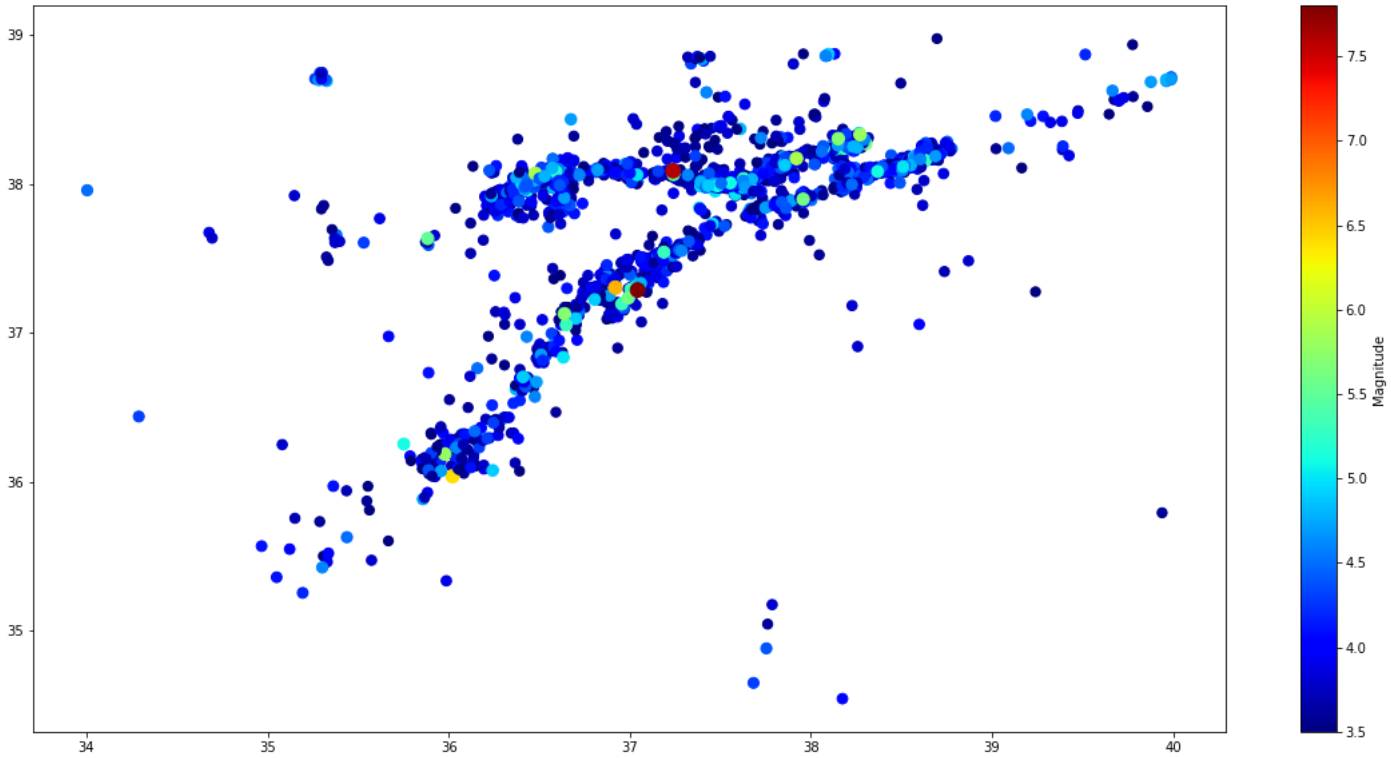
	Date	Longitude	Latitude	Depth	Magnitude
0	08/01/2024 13:19:12	38.7525	38.2842	8.97	4.4
1	07/01/2024 15:58:00	37.2725	38.3222	6.99	3.9
2	06/01/2024 12:10:09	38.5897	38.1694	8.76	4.0
3	05/01/2024 14:03:03	37.4519	38.3753	6.99	3.6
4	30/12/2023 13:26:24	39.0192	38.4564	9.36	4.2

In [6]:
```
Doublet_EAF_35.shape
```

Out[6]: (1821, 5)

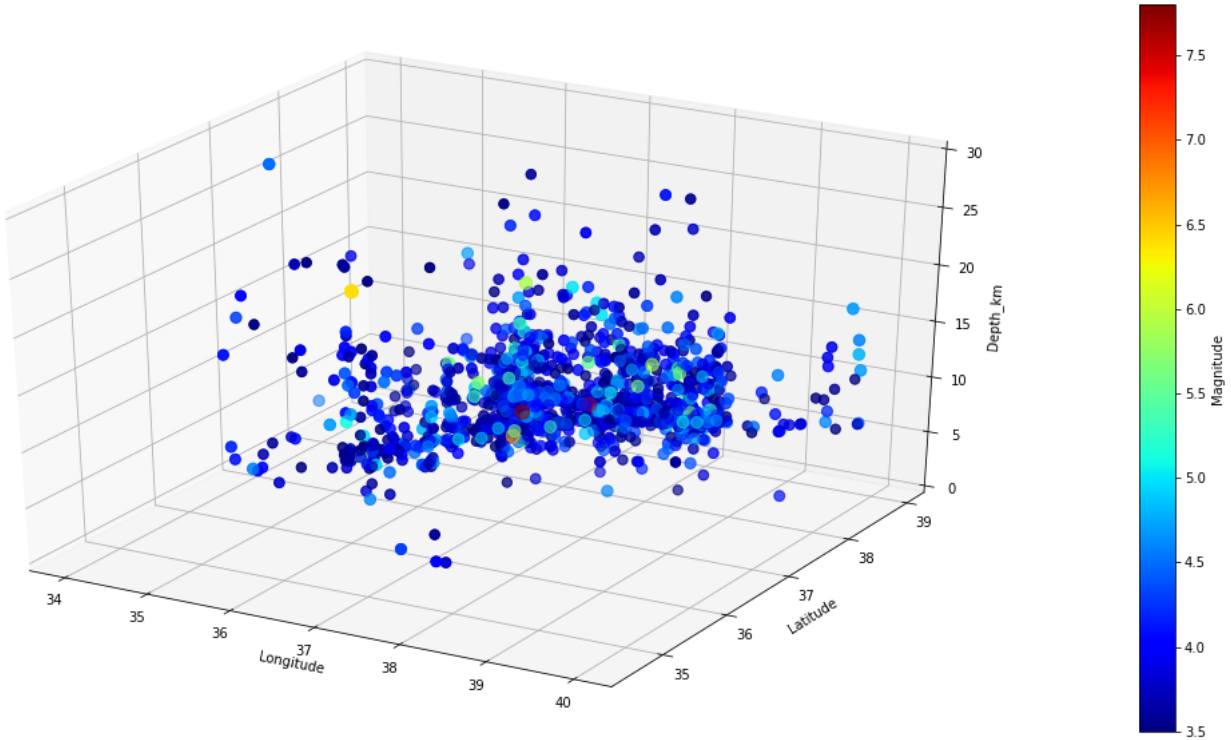
In [70]:
```
x = Doublet_EAF_35.iloc[:,1].values
y = Doublet_EAF_35.iloc[:,2].values
z = Doublet_EAF_35.iloc[:,3].values
colors = Doublet_EAF_35.iloc[:,4].values
sizes = Doublet_EAF_35.iloc[:,4].values*15
```

In [71]:
```
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)

plt.show()
```



In [72]:
```
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 [73]:
```
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, 30)


plt.show()
```



In [74]:
```
x = Doublet_EAF_35.iloc[:,1].values
y = Doublet_EAF_35.iloc[:,2].values
z = Doublet_EAF_35.iloc[:,3].values
colors = Doublet_EAF_35.iloc[:,4].values+1
sizes = Doublet_EAF_35.iloc[:,4].values*1
```

In [75]:
```
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=1, 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), fromcurrent=True

fig.show()
```

In [76]:
```
x = Doublet_EAF_35.iloc[:,1].values
y = Doublet_EAF_35.iloc[:,2].values
z = Doublet_EAF_35.iloc[:,0].values
colors = Doublet_EAF_35.iloc[:,4].values
sizes = Doublet_EAF_35.iloc[:,4].values*1
```

In [77]:
```
z
```

Out[77]:
```
array(['08/01/2024 13:19:12', '07/01/2024 15:58:00',
       '06/01/2024 12:10:09', ..., '06/02/2023 01:26:49',
       '06/02/2023 01:23:16', '06/02/2023 01:17:32'], dtype=object)
```

In [78]:
```
from datetime import datetime

# Verileri oluştur
timestamps = z

# Zaman damgalarını aylara dönüştür
months = [datetime.strptime(timestamp, '%d/%m/%Y %H:%M:%S').month for timestamp in timestamps]

print(months)  # Ay bilgilerini görüntüle
```

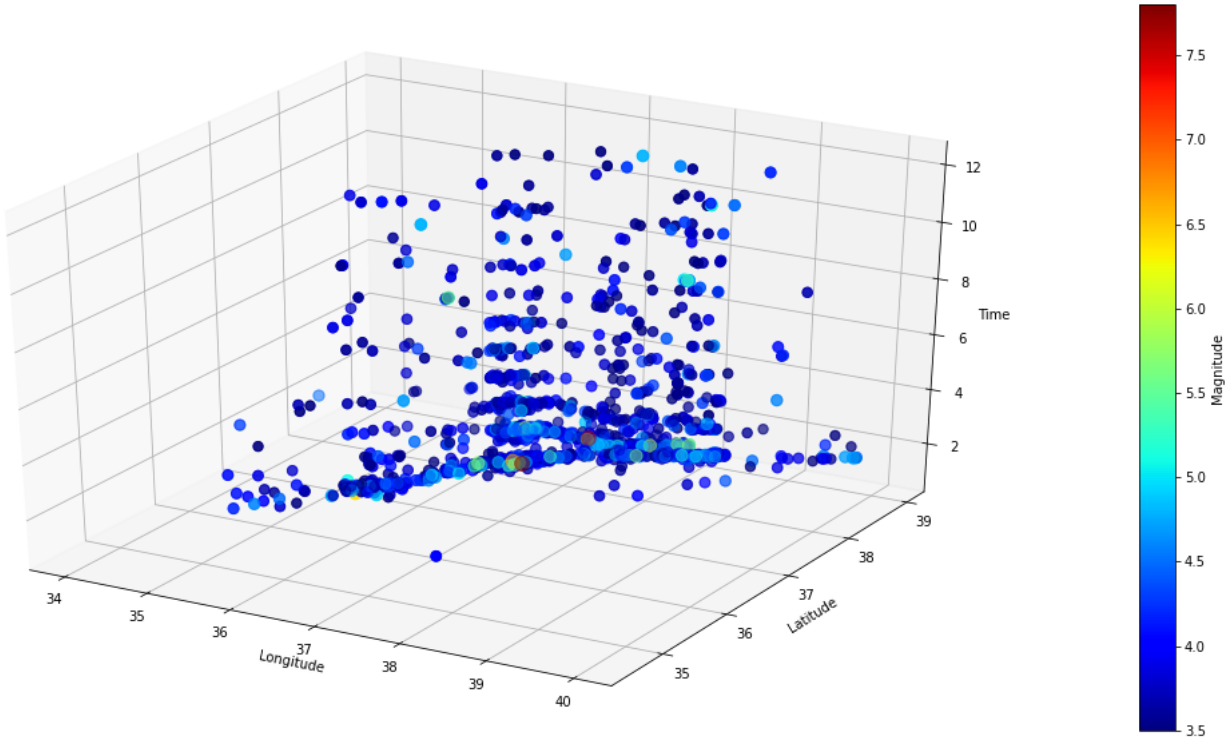
```
[1, 1, 1, 1, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1
0, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

In [79]:
```
months
```

Out[79]:
```
[1,
 1,
 1,
 1,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 11,
```

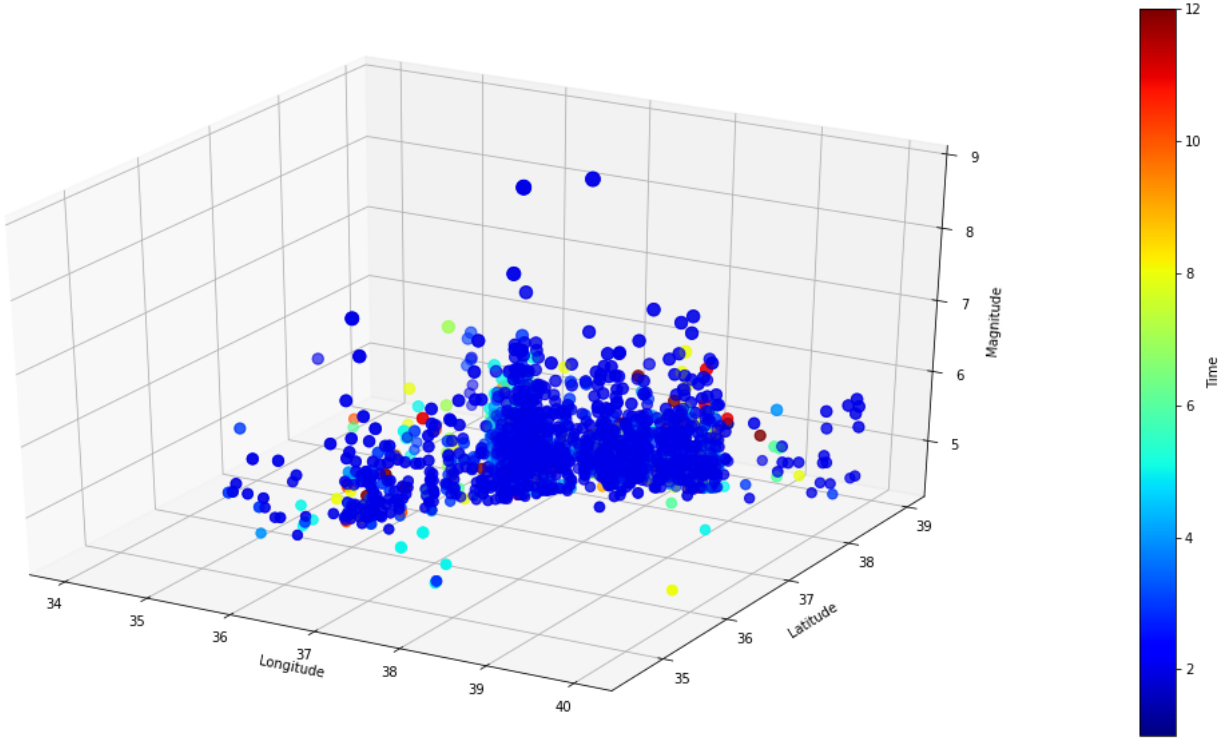
In [80]:
```
x = Doublet_EAF_35.iloc[:,1].values
y = Doublet_EAF_35.iloc[:,2].values
z = months
colors = Doublet_EAF_35.iloc[:,4].values
sizes = Doublet_EAF_35.iloc[:,4].values*15
```

In [81]:
```
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
```



In [82]:
```
x = Doublet_EAF_35.iloc[:,1].values
y = Doublet_EAF_35.iloc[:,2].values
z = Doublet_EAF_35.iloc[:,4].values+1
colors = months
sizes = Doublet_EAF_35.iloc[:,4].values*15
```

In [83]:
```
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




plt.show()
```



In [ ]: