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: python-dateutil>=2.8.2 in c:\users\samil\anaconda3\lib\site-packages (from pandas) (2.8.2)
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: 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)
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: six in c:\users\samil\anaconda3\lib\site-packages (from packaging->plotly) (1.15.0)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\samil\anaconda3\lib\site-packages (from packaging->plotly) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\samil\anaconda3\lib\site-packages (3.7.4)Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: numpy<2,>=1.20 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (1.24.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (1.1.1)
Requirement already satisfied: importlib-resources>=3.2.0; python_version < "3.10" in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (6.1.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (4.47.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (7.2.0)
Requirement already satisfied: packaging>=20.0 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (20.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\samil\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: zipp>=3.1.0; python_version < "3.10" in c:\users\samil\anaconda3\lib\site-packages (from importlib-resources>=3.2.0; python_version
< "3.10"->matplotlib) (3.1.0)
Requirement already satisfied: six>=1.5 in c:\users\samil\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
```

In [4]:
```
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 installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [5]:
```
raw_dataset=pd.read_csv("Doublet_EAF_1Y.csv",sep=",")
```

In [6]:
```
 Doublet_EAF_1Y= raw_dataset.copy()
Doublet_EAF_1Y.head()
```

Out[6]:

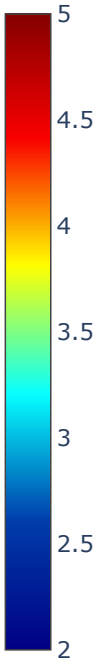
	Date	Longitude	Latitude	Depth	Magnitude
0	05/02/2023 04:16:52	36.044	37.411	7.08	2.0
1	04/02/2023 08:22:17	36.356	37.390	7.00	2.7
2	03/02/2023 22:43:10	38.814	38.274	6.57	2.5
3	03/02/2023 22:06:30	36.360	37.230	7.02	2.1
4	03/02/2023 11:37:12	36.395	37.201	7.01	2.2

In [7]:
```
Doublet_EAF_1Y.shape
```

Out[7]:
```
(481, 5)
```

In [16]:
```
x = Doublet_EAF_1Y.iloc[:,1].values
y = Doublet_EAF_1Y.iloc[:,2].values
z = Doublet_EAF_1Y.iloc[:,3].values
colors = Doublet_EAF_1Y.iloc[:,4].values
sizes = Doublet_EAF_1Y.iloc[:,4].values*70
```

In [17]:
```
import plotly.graph_objects as go

# Obtain high-resolution world map data online
fig = go.Figure(go.Choroplethmapbox(
    geojson="https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json",
    locations=["USA", "CAN", "MEX", "RUS", "CHN"],  # Example country codes (USA, Canada, Mexico, Russia, China)
    z=[1, 1, 1, 1, 1],  # Values to be assigned to countries (all set to 1)
    colorscale='Jet',  # Color scale name (Viridis, YlGnBu, Jet, etc.)
    zmin=2,
    zmax=5,
    marker_opacity=0.9,  # Opacity of country borders
    marker_line_width=1,  # Thickness of country borders
))

# Create sample earthquake data
earthquake_data = {
    'Longitude': x,
    'Latitude': y,
    'Magnitude': colors,
}

# Add earthquake data with Scatter plot
fig.add_trace(go.Scattermapbox(
    lat=earthquake_data['Latitude'],
    lon=earthquake_data['Longitude'],
    mode='markers',
    marker=dict(
        size=earthquake_data['Magnitude'] * 5,  # Set point sizes based on Magnitude value
        color=earthquake_data['Magnitude'],  # Set color scale based on Magnitude value
        colorscale='Jet',  # Color scale name (Viridis, YlGnBu, Jet, etc.)
    ),
))

# Specify map layout and style
fig.update_layout(
    mapbox_style="open-street-map",  # Set map style (for other styles: "open-street-map", "stamen-terrain", etc.)
    mapbox_zoom=6,  # Set map zoom level
    mapbox_center={"lat": 37.000, "lon": 37.0000},  # Set map center (center of the USA)
)

# Increase resolution and font size
fig.update_layout(
    width=700,  # Set width to increase resolution
    height=620,  # Set height to increase resolution
    font=dict(
        size=15  # Set font size for English comments
    )
)

# Display the plot
fig.show()
```


5
4.5
4
3.5
3
2.5
2

In [18]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import ipywidgets as widgets
from ipywidgets import interactive
from IPython.display import display



# İnteraktif işlev

def plot_3d_scatter(elev, azim, zoom, theta):
    fig = plt.figure(figsize=(12, 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.view_init(elev=elev, azim=azim)
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    ax.set_zlabel('Depth_km')
    ax.dist = zoom  # Zoom ayarı
    ax.azim = theta  # Maus ile çevirme

    plt.rc('font', size=16)
    plt.show()

# İnteraktif widget'ı oluşturma
elev_slider = widgets.IntSlider(min=0, max=180, value=30, description='Elevation:')
azim_slider = widgets.IntSlider(min=0, max=360, value=30, description='Azimuth:')
zoom_slider = widgets.FloatSlider(min=1, max=10, value=5, description='Zoom:')
theta_slider = widgets.IntSlider(min=0, max=360, value=30, description='Theta:')
interactive_plot = interactive(plot_3d_scatter, elev=elev_slider, azim=azim_slider, zoom=zoom_slider, theta=theta_slider)

# Widget'ı görüntüleme
display(interactive_plot)
```

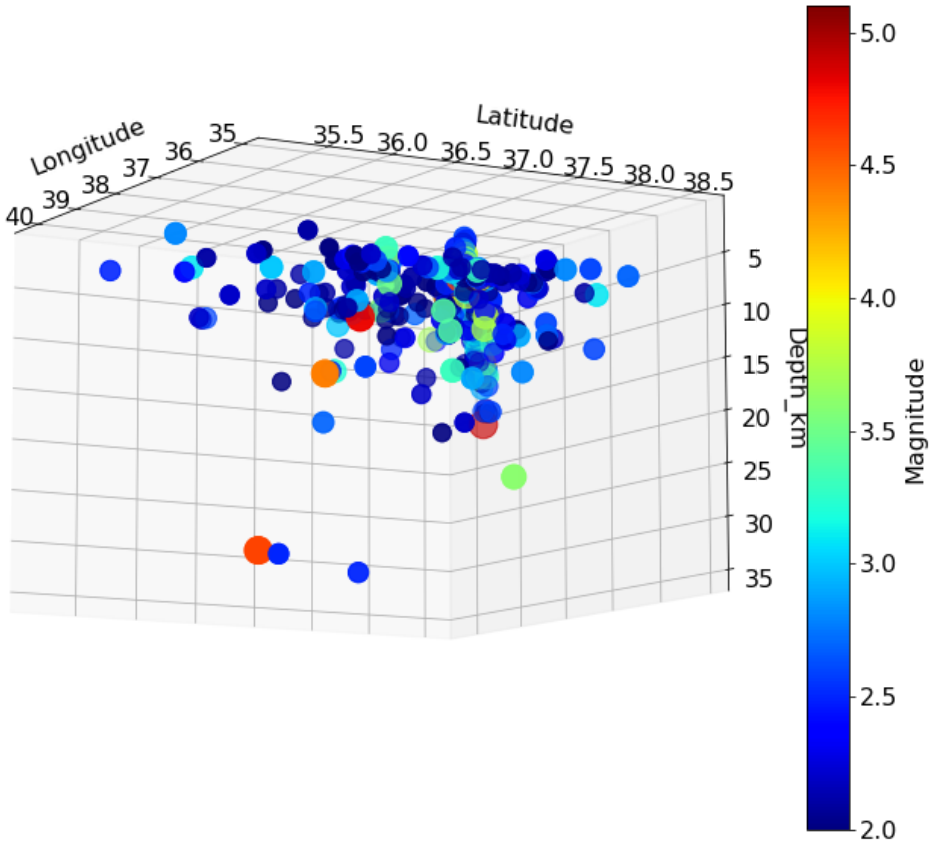
Elevation: 171

Azimuth: 30

Zoom: 10.00

Theta: 30

```
<ipython-input-18-bb80dd266c46>:24: MatplotlibDeprecationWarning:

The dist attribute was deprecated in Matplotlib 3.6 and will be removed two minor releases later.
```



In [11]:
```
x = Doublet_EAF_1Y.iloc[:,1].values
y = Doublet_EAF_1Y.iloc[:,2].values
z = Doublet_EAF_1Y.iloc[:,0].values
colors = Doublet_EAF_1Y.iloc[:,4].values
sizes = Doublet_EAF_1Y.iloc[:,4].values*30
```

In [12]:
```
z
```

```
'07/01/2023 22:58:27', '07/01/2023 20:47:42',
'06/01/2023 01:21:03', '05/01/2023 13:48:35',
'05/01/2023 07:43:08', '05/01/2023 07:41:03',
'04/01/2023 13:20:52', '04/01/2023 09:51:45',
'02/01/2023 18:59:59', '02/01/2023 10:38:03',
'02/01/2023 04:38:02', '02/01/2023 01:51:44',
'01/01/2023 21:25:16', '01/01/2023 18:48:32',
'31/12/2022 08:15:02', '29/12/2022 04:00:15',
'28/12/2022 18:56:23', '28/12/2022 08:21:20',
'28/12/2022 07:31:02', '27/12/2022 19:55:00',
'27/12/2022 19:06:24', '27/12/2022 18:15:06',
'27/12/2022 18:09:33', '27/12/2022 07:31:32',
'26/12/2022 17:14:56', '26/12/2022 10:08:51',
'26/12/2022 08:03:25', '24/12/2022 15:53:04',
'24/12/2022 02:23:51', '23/12/2022 22:14:59',
'23/12/2022 18:17:03', '23/12/2022 17:48:50',
'23/12/2022 17:23:31', '23/12/2022 17:23:01',
'23/12/2022 00:42:11', '23/12/2022 00:40:11',
'22/12/2022 22:19:00', '22/12/2022 22:18:43',
'22/12/2022 21:37:58', '22/12/2022 21:26:07',
```

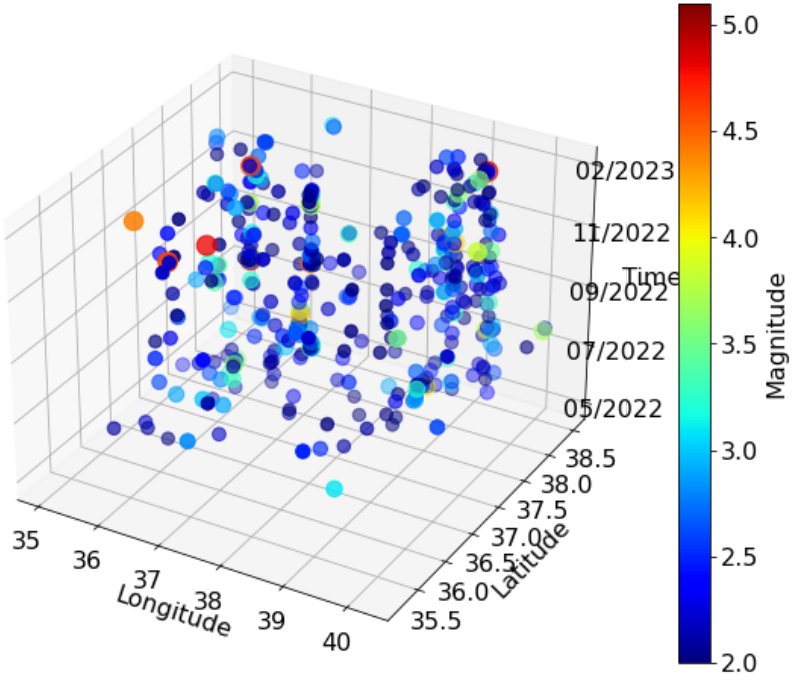
In [13]:
```
from datetime import datetime

# Zaman damgalarını içeren bir liste oluştur
timestamps = z

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

print(seconds) # Saniye cinsinden zaman damgalarını görüntüle
```

```
[1675559812.0, 1675488137.0, 1675453390.0, 1675451190.0, 1675413432.0, 1675411508.0, 1675224124.0, 1675124698.0, 1675045005.0, 1674997959.0, 1674950526.0, 16747118
87.0, 1674679136.0, 1674623185.0, 1674615971.0, 1674613283.0, 1674597456.0, 1674585644.0, 1674343719.0, 1674323232.0, 1674287507.0, 1674257214.0, 1674245039.0, 167
4158668.0, 1674037847.0, 1673968960.0, 1673917950.0, 1673819193.0, 1673751565.0, 1673746357.0, 1673746144.0, 1673744847.0, 1673743177.0, 1673742986.0, 1673640164.
0, 1673631848.0, 1673628484.0, 1673623784.0, 1673583362.0, 1673545249.0, 1673430093.0, 1673267139.0, 1673260074.0, 1673254420.0, 1673233062.0, 1673215917.0, 167312
1507.0, 1673113662.0, 1672957263.0, 1672915715.0, 1672893788.0, 1672893663.0, 1672827652.0, 1672815105.0, 1672675199.0, 1672645083.0, 1672623482.0, 1672613504.0, 1
672597516.0, 1672588112.0, 1672463702.0, 1672275615.0, 1672242983.0, 1672204880.0, 1672201862.0, 1672160100.0, 1672157184.0, 1672154106.0, 1672153773.0, 167211549
2.0, 1672064096.0, 1672038531.0, 1672031005.0, 1671886384.0, 1671837831.0, 1671822899.0, 1671808623.0, 1671806930.0, 1671805411.0, 1671805381.0, 1671745331.0, 1671
745211.0, 1671736740.0, 1671736723.0, 1671734278.0, 1671733567.0, 1671726547.0, 1671726296.0, 1671723191.0, 1671696391.0, 1671655718.0, 1671651725.0, 1671637874.0,
1671537848.0, 1671376389.0, 1671341052.0, 1671341015.0, 1671173979.0, 1671169615.0, 1671040398.0, 1671006260.0, 1670900943.0, 1670887868.0, 1670885530.0, 167082196
0.0, 1670737325.0, 1670725362.0, 1670651087.0, 1670614303.0, 1670581179.0, 1670306001.0, 1670266088.0, 1670080082.0, 1670042257.0, 1670010792.0, 1669869831.0, 1669
752385.0, 1669665987.0, 1669665871.0, 1669631812.0, 1669297491.0, 1669212826.0, 1669202370.0, 1669163671.0, 1669129764.0, 1669056893.0, 1669009451.0, 1668839073.0,
1668769873.0, 1668639192.0, 1668620419.0, 1668620408.0, 1668440521.0, 1668258809.0, 1668158745.0, 1668105532.0, 1667940265.0, 1667899173.0, 1667869993.0, 166779288
9.0, 1667713076.0, 1667679600.0, 1667672742.0, 1667638348.0, 1667627775.0, 1667604988.0, 1667563573.0, 1667384590.0, 1667381019.0, 1667376987.0, 1667357575.0, 1667
230465.0, 1667212381.0, 1667139725.0, 1667075551.0, 1667019891.0, 1667010576.0, 1667006020.0, 1666984405.0, 1666973770.0, 1666955631.0, 1666920365.0, 1666850372.0,
1666821771.0, 1666632808.0, 1666625980.0, 1666622118.0, 1666522738.0, 1666512775.0, 1666502909.0, 1666365201.0, 1666327777.0, 1666301069.0, 1666292346.0, 166628668
2.0, 1666286405.0, 1666285078.0, 1666283831.0, 1666277519.0, 1666277435.0, 1666276387.0, 1666275242.0, 1666274154.0, 1666271067.0, 1666270312.0, 1666262533.0, 1666
261068.0, 1666259470.0, 1666259161.0, 1666257541.0, 1666257192.0, 1666256733.0, 1666256042.0, 1666254899.0, 1666243868.0, 1666170289.0, 1666102623.0, 1666060368.0,
1666053081.0, 1666037783.0, 1666037367.0, 1665905716.0, 1665901721.0, 1665901464.0, 1665857614.0, 1665722263.0, 1665718492.0, 1665684741.0, 1665679310.0, 166560527
3.0, 1665605065.0, 1665594069.0, 1665560723.0, 1665536086.0, 1665535989.0, 1665509135.0, 1665503168.0, 1665498228.0, 1665495284.0, 1665493942.0, 1665492775.0, 1665
492526.0, 1665396450.0, 1665137774.0, 1664966325.0, 1664947721.0, 1664937037.0, 1664932521.0, 1664931776.0, 1664857390.0, 1664850983.0, 1664646941.0, 1664593298.0,
1664591077.0, 1664590957.0, 1664551609.0, 1664543910.0, 1664521182.0, 1664498792.0, 1664395811.0, 1664393287.0, 1664363302.0, 1664344296.0, 1664337434.0, 166433245
2.0, 1664297623.0, 1664197340.0, 1664132610.0, 1663996960.0, 1663808104.0, 1663790000.0, 1663787622.0, 1663766574.0, 1663706202.0, 1663668956.0, 1663660863.0, 1663
660015.0, 1663639132.0, 1663602830.0, 1663452013.0, 1663413290.0, 1663411166.0, 1663387764.0, 1663163479.0, 1663066551.0, 1663053856.0, 1663027421.0, 1662841281.0,
1662827464.0, 1662827416.0, 1662763101.0, 1662754338.0, 1662711639.0, 1662710661.0, 1662702248.0, 1662698432.0, 1662671740.0, 1662530878.0, 1662527005.0, 166249721
0.0, 1662372979.0, 1662371715.0, 1662339336.0, 1662187470.0, 1662121479.0, 1662075011.0, 1662061464.0, 1662060802.0, 1662023884.0, 1661796636.0, 1661702594.0, 1661
681418.0, 1661660898.0, 1661636309.0, 1661337840.0, 1661319203.0, 1661183422.0, 1661137379.0, 1661110757.0, 1661099193.0, 1661012081.0, 1660971382.0, 1660954644.0,
1660686719.0, 1660677793.0, 1660672411.0, 1660594387.0, 1660510656.0, 1660351844.0, 1660342394.0, 1660276127.0, 1660127159.0, 1660039894.0, 1660030832.0, 165997846
4.0, 1659974323.0, 1659912148.0, 1659908956.0, 1659723123.0, 1659677364.0, 1659587214.0, 1659554420.0, 1659549733.0, 1659524521.0, 1659512936.0, 1659487868.0, 1659
458725.0, 1659452997.0, 1659420601.0, 1659388890.0, 1659326599.0, 1659319801.0, 1659309909.0, 1659308191.0, 1659258666.0, 1659219585.0, 1659197687.0, 1659193577.0,
1659190902.0, 1658962073.0, 1658942334.0, 1658940095.0, 1658931658.0, 1658931592.0, 1658925522.0, 1658904979.0, 1658903563.0, 1658889247.0, 1658821769.0, 165880505
9.0, 1658678325.0, 1658678293.0, 1658642630.0, 1658639457.0, 1658638921.0, 1658545407.0, 1658513087.0, 1658511481.0, 1658442066.0, 1658438092.0, 1658387691.0, 1658
375359.0, 1658277667.0, 1658268448.0, 1658265843.0, 1658245885.0, 1658242902.0, 1658218693.0, 1658202113.0, 1658186334.0, 1658061724.0, 1658044060.0, 1657978393.0,
1657976802.0, 1657876426.0, 1657876280.0, 1657873856.0, 1657871005.0, 1657792512.0, 1657766534.0, 1657736253.0, 1657730879.0, 1657729440.0, 1657728499.0, 165772786
0.0, 1657727767.0, 1657725775.0, 1657725351.0, 1657716591.0, 1657713920.0, 1657713505.0, 1657713401.0, 1657713351.0, 1657688904.0, 1657685802.0, 1657603889.0, 1657
598511.0, 1657520627.0, 1657421501.0, 1657320733.0, 1657274776.0, 1657209302.0, 1657060626.0, 1657060239.0, 1656964567.0, 1656952191.0, 1656937636.0, 1656937333.0,
1656815355.0, 1656723257.0, 1656714577.0, 1656699875.0, 1656369436.0, 1656328939.0, 1656014165.0, 1655956505.0, 1655905267.0, 1655904055.0, 1655887587.0, 165570481
0.0, 1655511913.0, 1655291472.0, 1655227551.0, 1654962815.0, 1654927295.0, 1654927084.0, 1654858082.0, 1654792925.0, 1654711662.0, 1654587573.0, 1654523495.0, 1654
522235.0, 1654499943.0, 1654438920.0, 1654353260.0, 1654133043.0, 1654038506.0, 1653952997.0, 1653947456.0, 1653939246.0, 1653923238.0, 1653910379.0, 1653769294.0,
1653681846.0, 1653636444.0, 1653633522.0, 1653452631.0, 1653379751.0, 1653283363.0, 1653261257.0, 1653217288.0, 1653070045.0, 1653032709.0, 1653023839.0, 165293548
2.0, 1652918616.0, 1652857138.0, 1652778058.0, 1652678226.0, 1652574768.0, 1652530132.0, 1652513553.0, 1652473859.0, 1652464724.0, 1652448182.0, 1652305361.0, 1652
216347.0, 1652197489.0, 1652193814.0, 1652119476.0, 1652115649.0, 1652102715.0, 1652067010.0, 1652060879.0, 1651946516.0, 1651834547.0, 1651777177.0, 1651755203.0,
1651720772.0, 1651713784.0, 1651613318.0]
```

In [14]:
```
seconds
```

Out[14]:
```
[1675559812.0,
 1675488137.0,
 1675453390.0,
 1675451190.0,
 1675413432.0,
 1675411508.0,
 1675224124.0,
 1675124698.0,
 1675045005.0,
 1674997959.0,
 1674950526.0,
 1674711887.0,
 1674679136.0,
 1674623185.0,
 1674615971.0,
 1674613283.0,
 1674597456.0,
 1674585644.0,
 1674343719.0,
```

In [15]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from datetime import datetime

# Assuming Doublet_EAF_35_1Y is a DataFrame with appropriate columns
x = Doublet_EAF_1Y.iloc[:, 1].values
y = Doublet_EAF_1Y.iloc[:, 2].values
z = seconds
colors = Doublet_EAF_1Y.iloc[:, 4].values
sizes = Doublet_EAF_1Y.iloc[:, 4].values * 35

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

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

# Colorbar
cbar = plt.colorbar(scatter)
cbar.set_label('Magnitude')

# Labeling axes
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Time')

# Formatting time ticks
# Assuming seconds is a list or array of time values
# Adjust the ticks and labels according to your data
time_ticks = np.linspace(min(seconds), max(seconds), 5)
time_labels = [datetime.fromtimestamp(t).strftime('%m/%Y') for t in time_ticks]
ax.set_zticks(time_ticks)
ax.set_zticklabels(time_labels)

# Adjust font size
plt.rc('font', size=14)

plt.show()
```


02/2023
11/2022
09/2022
07/2022
05/2022
Time
35
36
37
38
39
40
Longitude
38.5
38.0
37.5
37.0
36.5
36.0
35.5
Latitude
5.0
4.5
4.0
3.5
3.0
2.5
2.0
Magnitude

In [ ]:

In [ ]:

In [ ]: