In [2]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation
from sklearn import datasets
import plotly.express as px
#  python3 -m pip install plotly
import pandas as pd
import numpy as np
from matplotlib import colors
```

In [ ]:
```
iris = datasets.load_iris()
X=iris.data
y=iris.target
labels = iris.target_names
```

Out[ ]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [4]:
```
rows, columns = iris.data.shape
print("Rows:", rows)
print("Columns:", columns)
```

```
Rows: 150
Columns: 4
```

In [5]:
```
iris.target_names
```

Out[5]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [6]:
```
iris
```

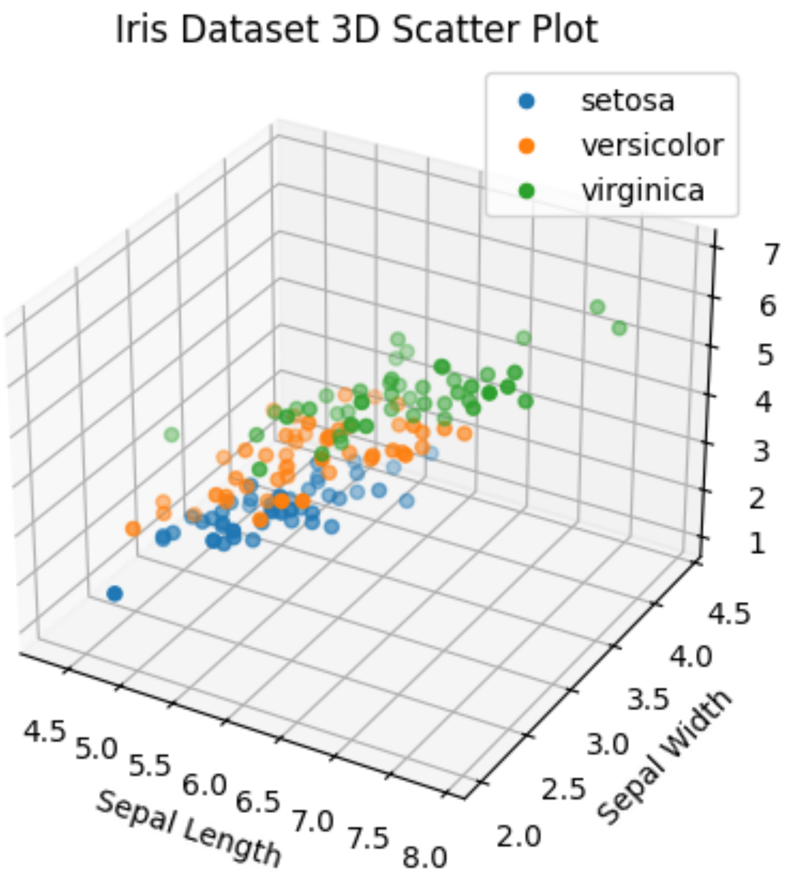
Out[6]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
        [5.1, 3.8, 1.5, 0.3],
        [5.4, 3.4, 1.7, 0.2],
        [5.1, 3.7, 1.5, 0.4],
        [4.6, 3.6, 1. , 0.2],
        [5.1, 3.3, 1.7, 0.5],
        [4.8, 3.4, 1.9, 0.2],
        [5. , 3. , 1.6, 0.2],
        [5. , 3.4, 1.6, 0.4],
        [5.2, 3.5, 1.5, 0.2],
        [5.2, 3.4, 1.4, 0.2],
        [4.7, 3.2, 1.6, 0.2],
        [4.8, 3.1, 1.6, 0.2],
        [5.4, 3.4, 1.5, 0.4],
        [5.2, 4.1, 1.5, 0.1],
        [5.5, 4.2, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.2],
        [5. , 3.2, 1.2, 0.2],
        [5.5, 3.5, 1.3, 0.2],
        [4.9, 3.6, 1.4, 0.1],
        [4.4, 3. , 1.3, 0.2],
        [5.1, 3.4, 1.5, 0.2],
        [5. , 3.5, 1.3, 0.3],
        [4.5, 2.3, 1.3, 0.3],
        [4.4, 3.2, 1.3, 0.2],
        [5. , 3.5, 1.6, 0.6],
        [5.1, 3.8, 1.9, 0.4],
        [4.8, 3. , 1.4, 0.3],
        [5.1, 3.8, 1.6, 0.2],
        [4.6, 3.2, 1.4, 0.2],
        [5.3, 3.7, 1.5, 0.2],
        [5. , 3.3, 1.4, 0.2],
        [7. , 3.2, 4.7, 1.4],
        [6.4, 3.2, 4.5, 1.5],
        [6.9, 3.1, 4.9, 1.5],
        [5.5, 2.3, 4. , 1.3],
        [6.5, 2.8, 4.6, 1.5],
        [5.7, 2.8, 4.5, 1.3],
        [6.3, 3.3, 4.7, 1.6],
        [4.9, 2.4, 3.3, 1. ],
        [6.6, 2.9, 4.6, 1.3],
        [5.2, 2.7, 3.9, 1.4],
        [5. , 2. , 3.5, 1. ],
        [5.9, 3. , 4.2, 1.5],
        [6. , 2.2, 4. , 1. ],
        [6.1, 2.9, 4.7, 1.4],
        [5.6, 2.9, 3.6, 1.3],
        [6.7, 3.1, 4.4, 1.4],
        [5.6, 3. , 4.5, 1.5],
        [5.8, 2.7, 4.1, 1. ],
        [6.2, 2.2, 4.5, 1.5],
        [5.6, 2.5, 3.9, 1.1],
        [5.9, 3.2, 4.8, 1.8],
        [6.1, 2.8, 4. , 1.3],
        [6.3, 2.5, 4.9, 1.5],
        [6.1, 2.8, 4.7, 1.2],
        [6.4, 2.9, 4.3, 1.3],
        [6.6, 3. , 4.4, 1.4],
        [6.8, 2.8, 4.8, 1.4],
        [6.7, 3. , 5. , 1.7],
        [6. , 2.9, 4.5, 1.5],
        [5.7, 2.6, 3.5, 1. ],
        [5.5, 2.4, 3.8, 1.1],
        [5.5, 2.4, 3.7, 1. ],
        [5.8, 2.7, 3.9, 1.2],
        [6. , 2.7, 5.1, 1.6],
        [5.4, 3. , 4.5, 1.5],
        [6. , 3.4, 4.5, 1.6],
        [6.7, 3.1, 4.7, 1.5],
        [6.3, 2.3, 4.4, 1.3],
        [5.6, 3. , 4.1, 1.3],
        [5.5, 2.5, 4. , 1.3],
        [5.5, 2.6, 4.4, 1.2],
        [6.1, 3. , 4.6, 1.4],
        [5.8, 2.6, 4. , 1.2],
        [5. , 2.3, 3.3, 1. ],
        [5.6, 2.7, 4.2, 1.3],
        [5.7, 3. , 4.2, 1.2],
        [5.7, 2.9, 4.2, 1.3],
        [6.2, 2.9, 4.3, 1.3],
        [5.1, 2.5, 3. , 1.1],
        [5.7, 2.8, 4.1, 1.3],
        [6.3, 3.3, 6. , 2.5],
        [5.8, 2.7, 5.1, 1.9],
        [7.1, 3. , 5.9, 2.1],
        [6.3, 2.9, 5.6, 1.8],
        [6.5, 3. , 5.8, 2.2],
        [7.6, 3. , 6.6, 2.1],
        [4.9, 2.5, 4.5, 1.7],
        [7.3, 2.9, 6.3, 1.8],
        [6.7, 2.5, 5.8, 1.8],
        [7.2, 3.6, 6.1, 2.5],
        [6.5, 3.2, 5.1, 2. ],
        [6.4, 2.7, 5.3, 1.9],
        [6.8, 3. , 5.5, 2.1],
        [5.7, 2.5, 5. , 2. ],
        [5.8, 2.8, 5.1, 2.4],
        [6.4, 3.2, 5.3, 2.3],
        [6.5, 3. , 5.5, 1.8],
        [7.7, 3.8, 6.7, 2.2],
        [7.7, 2.6, 6.9, 2.3],
        [6. , 2.2, 5. , 1.5],
        [6.9, 3.2, 5.7, 2.3],
        [5.6, 2.8, 4.9, 2. ],
        [7.7, 2.8, 6.7, 2. ],
        [6.3, 2.7, 4.9, 1.8],
        [6.7, 3.3, 5.7, 2.1],
        [7.2, 3.2, 6. , 1.8],
        [6.2, 2.8, 4.8, 1.8],
        [6.1, 3. , 4.9, 1.8],
        [6.4, 2.8, 5.6, 2.1],
        [7.2, 3. , 5.8, 1.6],
        [7.4, 2.8, 6.1, 1.9],
        [7.9, 3.8, 6.4, 2. ],
        [6.4, 2.8, 5.6, 2.2],
        [6.3, 2.8, 5.1, 1.5],
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3. , 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6. , 3. , 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtyp
e='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n-------------
-------\n\n**Data Set Characteristics:**\n\n:Number of Instances:
150 (50 in each of three classes)\n:Number of Attributes: 4 numeri
c, predictive attributes and the class\n:Attribute Information:\n
- sepal length in cm\n    - sepal width in cm\n    - petal length
in cm\n    - petal width in cm\n    - class:\n            - Iris-S
etosa\n            - Iris-Versicolour\n            - Iris-Virginic
a\n\n:Summary Statistics:\n\n============== ==== ==== ======= ====
= ====================\n                Min  Max   Mean    SD   Cl
ass Correlation\n============== ==== ==== ======= ===== ==========
==========\nsepal length:   4.3  7.9   5.84   0.83    0.7826\nsepa
l width:    2.0  4.4   3.05   0.43   -0.4194\npetal length:   1.0
6.9   3.76   1.76    0.9490  (high!)\npetal width:    0.1  2.5
1.20   0.76    0.9565  (high!)\n============== ==== ==== ======= =
==== ====================\n\n:Missing Attribute Values: None\n:Cla
ss Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. Fish
er\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Dat
e: July, 1988\n\nThe famous Iris database, first used by Sir R.A.
Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'
s the same as in R, but not as in the UCI\nMachine Learning Reposi
tory, which has two wrong data points.\n\nThis is perhaps the best
known database to be found in the\npattern recognition literature.
Fisher\'s paper is a classic in the field and\nis referenced frequ
ently to this day.  (See Duda & Hart, for example.)  The\ndata set
contains 3 classes of 50 instances each, where each class refers t
o a\ntype of iris plant.  One class is linearly separable from the
```

```
other 2; the\nlatter are NOT linearly separable from each other.\n
\n.. dropdown:: References\n\n  - Fisher, R.A. "The use of multipl
e measurements in taxonomic problems"\n    Annual Eugenics, 7, Par
t II, 179-188 (1936); also in "Contributions to\n    Mathematical
Statistics" (John Wiley, NY, 1950).\n  - Duda, R.O., & Hart, P.E.
(1973) Pattern Classification and Scene Analysis.\n    (Q327.D83)
John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n  - Dasara
thy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n
Structure and Classification Rule for Recognition in Partially Exp
osed\n    Environments".  IEEE Transactions on Pattern Analysis an
d Machine\n    Intelligence, Vol. PAMI-2, No. 1, 67-71.\n  - Gate
s, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transact
ions\n    on Information Theory, May 1972, 431-433.\n  - See also:
1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n
conceptual clustering system finds 3 classes in the data.\n  - Man
y, many more ...\n',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

3D plot

In [7]:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i,label in enumerate(labels):
    ax.scatter(
        X[y==i, 0],
        X[y==i, 1],
        X[y==i, 2],
        label=label
    )
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')
ax.set_zlabel('Petal Length')
ax.set_title('Iris Dataset 3D Scatter Plot')
ax.legend()
plt.show()
```

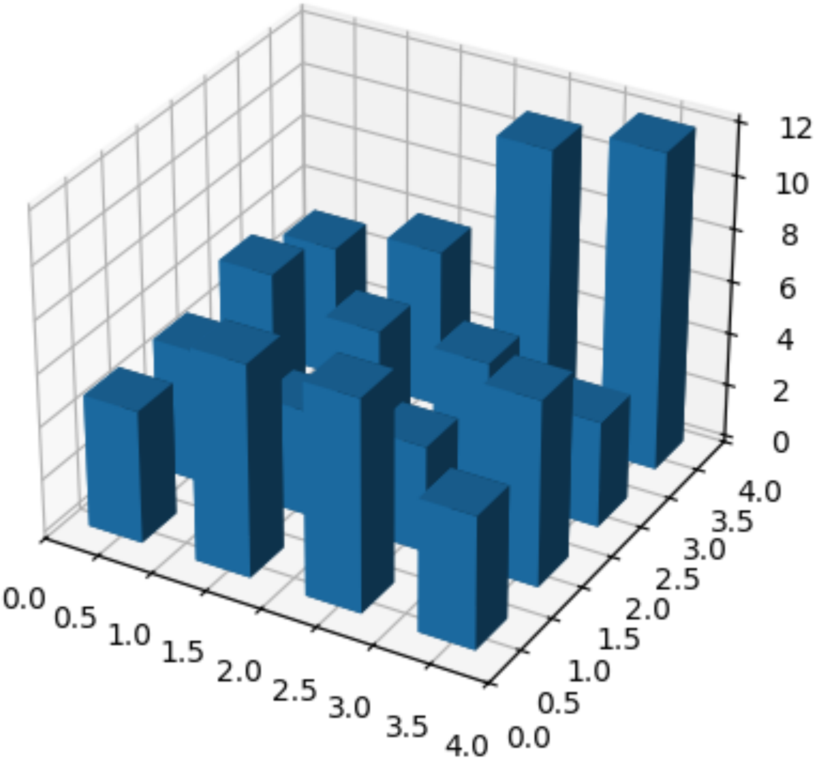


3D rotation

In [8]:
```
df = pd.DataFrame(
    iris.data,
    columns=iris.feature_names
    )

df['species'] = iris.target_names[iris.target]


# Plot
fig = px.scatter_3d(
    df,
    x='sepal length (cm)',
    y='sepal width (cm)',
    z='petal length (cm)',
    color='species',
    title='Iris Dataset 3D Scatter Plot'
)

fig.show(renderer="browser")
```

In [9]:
```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)
```

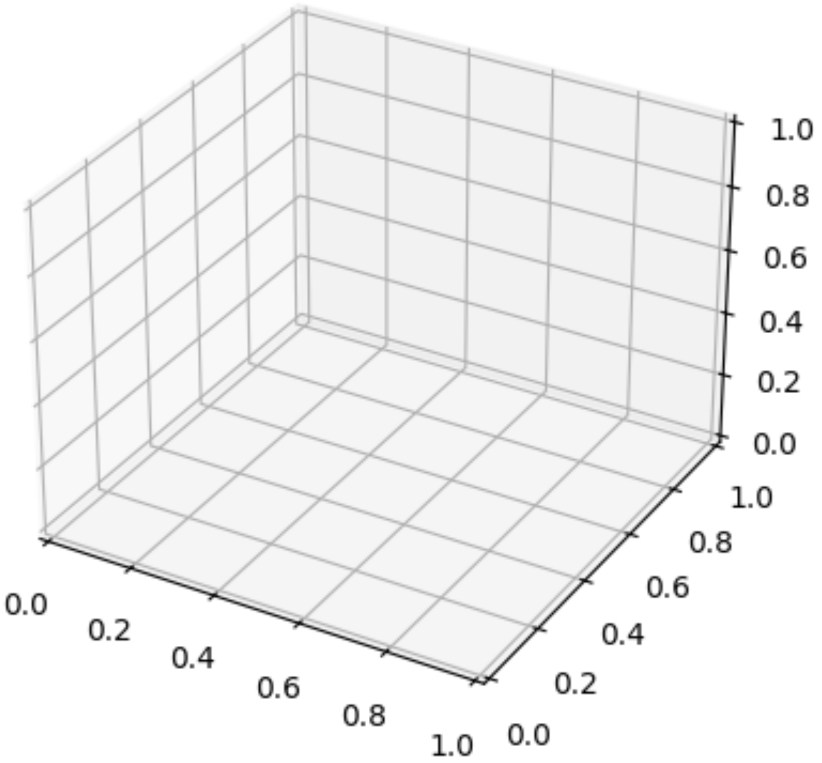
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x, y = np.random.rand(2, 100) * 4
hist, xedges, yedges = np.histogram2d(x, y, bins=4, range=[[0, 4],

# Construct arrays for the anchor positions of the 16 bars.
xpos, ypos = np.meshgrid(xedges[:-1] + 0.25, yedges[:-1] + 0.25, in
xpos = xpos.ravel()
ypos = ypos.ravel()
zpos = 0

# Construct arrays with the dimensions for the 16 bars.
dx = dy = 0.5 * np.ones_like(zpos)
dz = hist.ravel()

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, zsort='average')

plt.show()
```



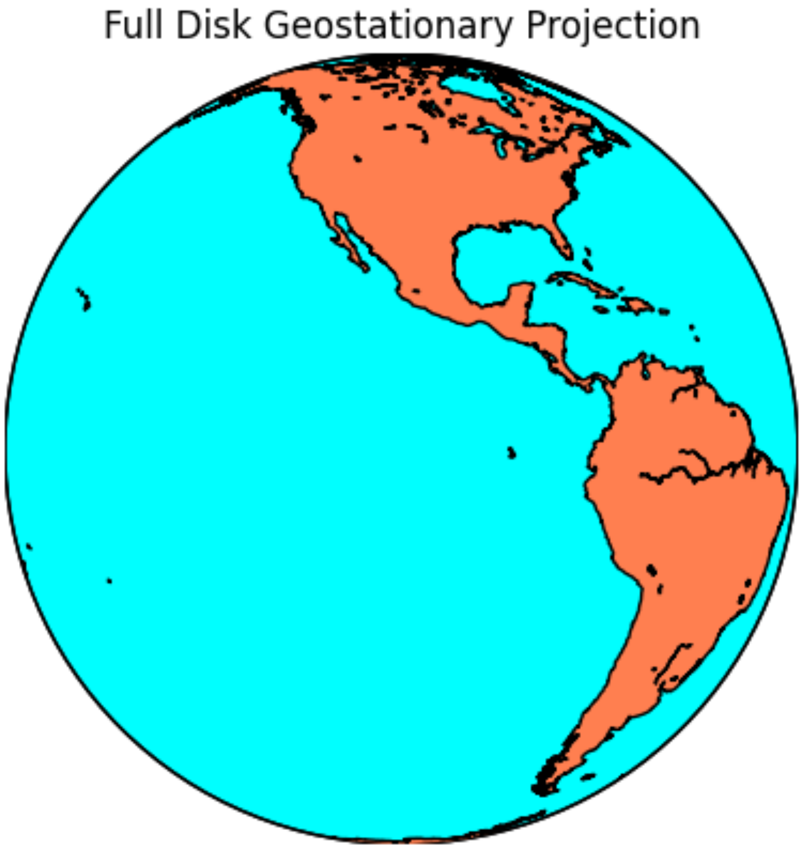
In [11]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

dx = dy = 0.15

for i, label in enumerate(labels):
    xpos = X[y == i, 0]   # Sepal length
    ypos = X[y == i, 1]   # Sepal width
    zpos = np.zeros_like(xpos)
    dz   = X[y == i, 2]   # Petal (height)

    ax.bar3d(xpos, ypos, zpos, dx, dy, dz, alpha=0.6, label=label)
```

```

# Labels and title
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')
ax.set_zlabel('Petal Length')
ax.set_title('Iris Dataset 3D Bar Plot')

ax.legend()
plt.show()
```

```
---------------------------------------------------------------------------
--------
IndexError                                Traceback (most recent cal
l last)
Cell In[11], line 7
      4 dx = dy = 0.15
      6 for i, label in enumerate(labels):
----> 7     xpos = X[y == i, 0]    # Sepal length
      8     ypos = X[y == i, 1]    # Sepal width
      9     zpos = np.zeros_like(xpos)

IndexError: boolean index did not match indexed array along axis 0;
size of axis is 150 but size of corresponding boolean axis is 100
```

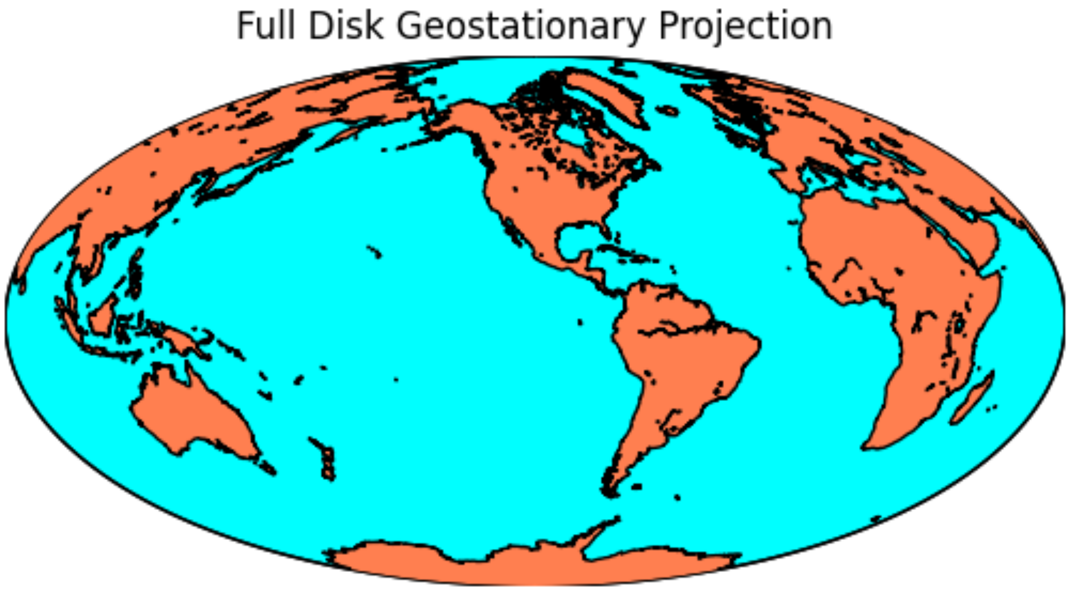


```
In [ ]: #%pip install basemap
        from mpl_toolkits.basemap import Basemap
        m=Basemap(projection='geos',lon_0=-105,resolution='l', rsphere=(637
        m.drawcoastlines()
        m.fillcontinents(color='coral',lake_color='aqua')
        m.drawmapboundary(fill_color='aqua')
        plt.title("Full Disk Geostationary Projection")
        plt.show()
```



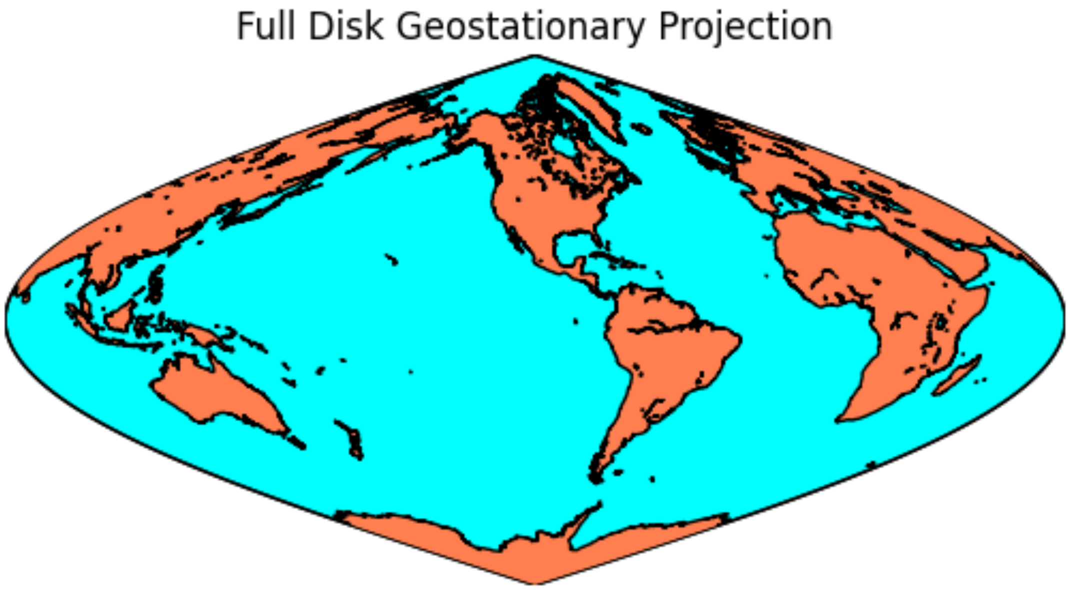
ortho,gnom,hammer,sinu

```
In [ ]: #%pip install basemap
        from mpl_toolkits.basemap import Basemap
        m=Basemap(projection='hammer',lon_0=-105,resolution='l', rsphere=(6
        m.drawcoastlines()
        m.fillcontinents(color='coral',lake_color='aqua')
        m.drawmapboundary(fill_color='aqua')
        plt.title("Full Disk Geostationary Projection")
        plt.show()
```



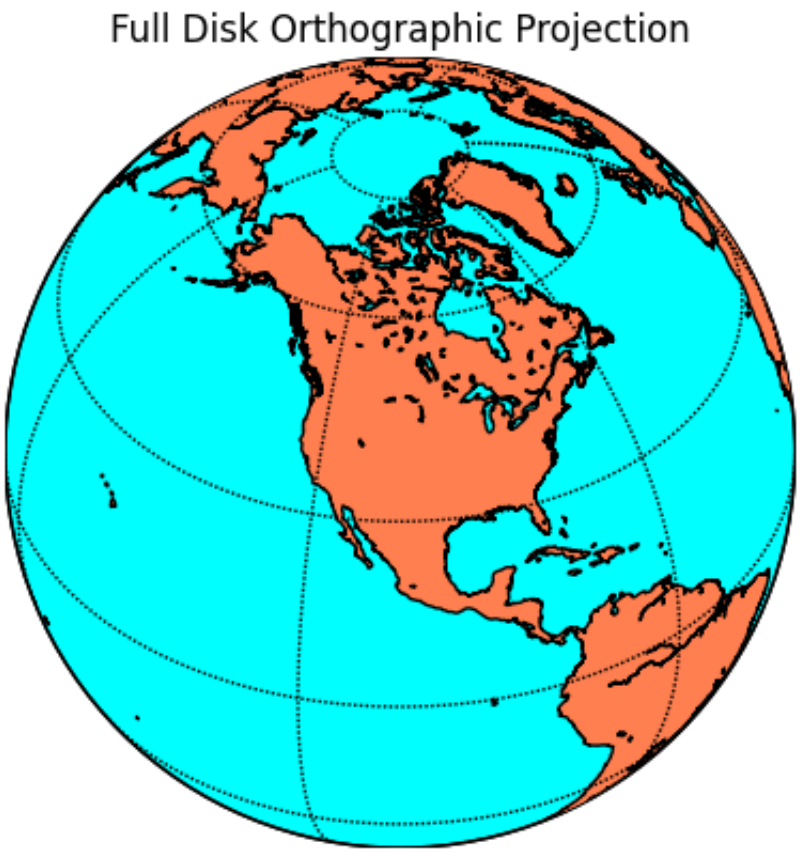
In [ ]:

```
#%pip install basemap
from mpl_toolkits.basemap import Basemap
m=Basemap(projection='sinu',lon_0=-105,resolution='l', rsphere=(637
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary(fill_color='aqua')
plt.title("Full Disk Geostationary Projection")
plt.show()
```



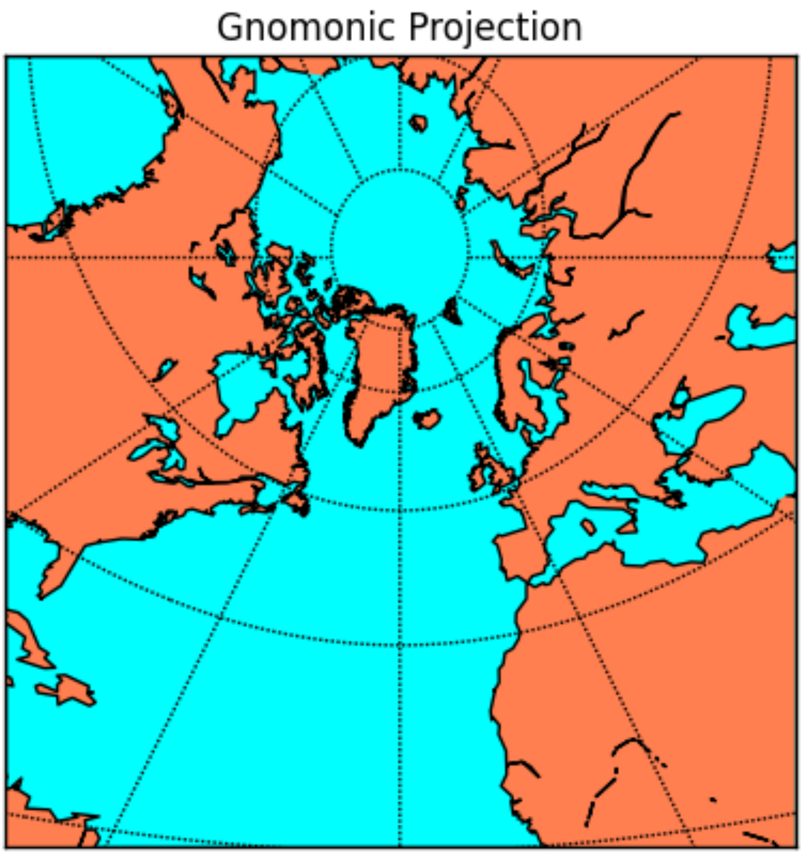
In [ ]:

```
from mpl_toolkits.basemap import Basemap
import numpy as np
import matplotlib.pyplot as plt
# lon_0, lat_0 are the center point of the projection.
# resolution = 'l' means use low resolution coastlines.
m = Basemap(projection='ortho',lon_0=-105,lat_0=40,resolution='l')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
# draw parallels and meridians.
m.drawparallels(np.arange(-90.,120.,30.))
m.drawmeridians(np.arange(0.,420.,60.))
m.drawmapboundary(fill_color='aqua')
plt.title("Full Disk Orthographic Projection")
plt.show()
```



In [ ]:

```
from mpl_toolkits.basemap import Basemap
import numpy as np
import matplotlib.pyplot as plt
m = Basemap(width=15.e6,height=15.e6,\
            projection='gnom',lat_0=60.,lon_0=-30.)
m.drawmapboundary(fill_color='aqua')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawparallels(np.arange(10,90,20))
m.drawmeridians(np.arange(-180,180,30))
plt.title('Gnomonic Projection')
plt.show()
```

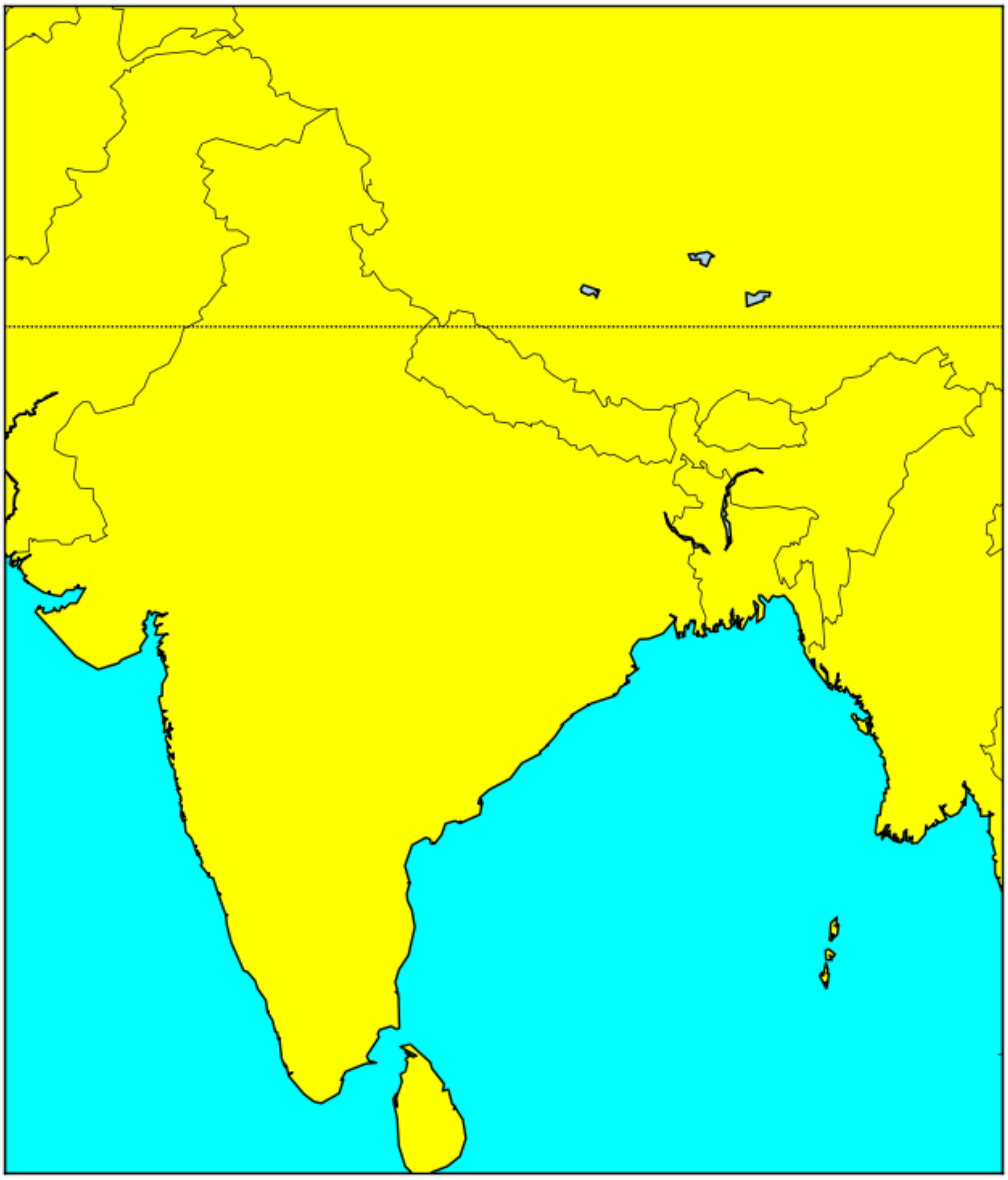


INDIAMAP

In [ ]:

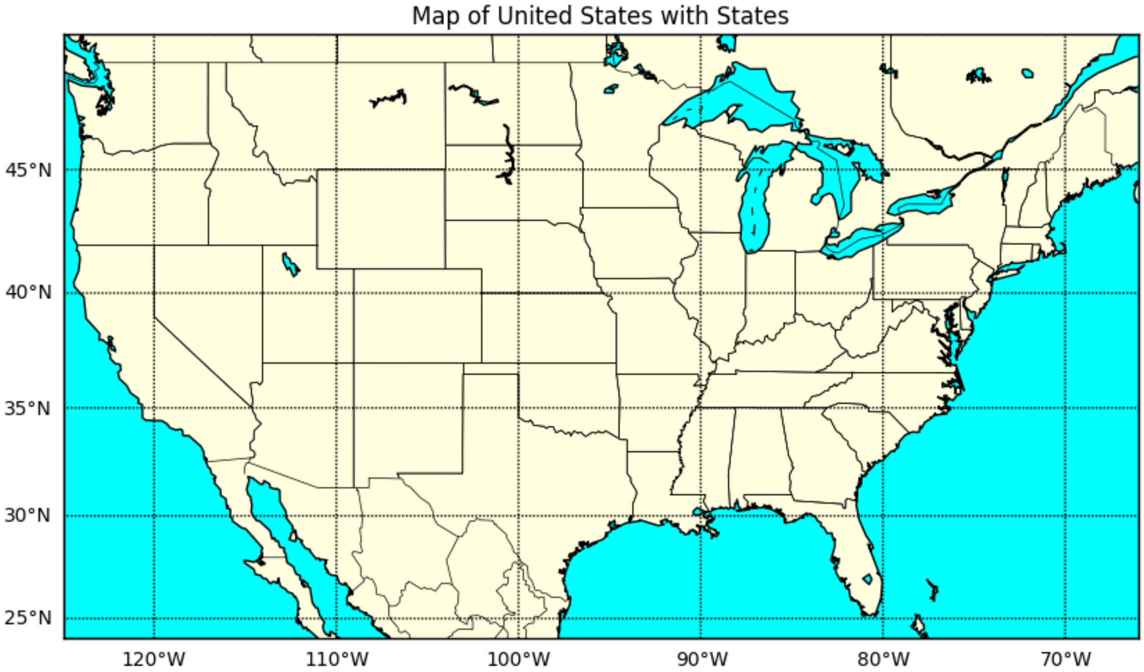
```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

plt.figure(figsize=(8,10))
#create india basemap
m=Basemap(
    projection='merc',
    llcrnrlat=6,
    urcrnrlat=38,
    llcrnrlon=68,
    urcrnrlon=98,
    resolution='l'
)
m.drawcoastlines()
m.drawcountries()
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='yellow',lake_color='lightblue')
m.drawstates()
m.drawparallels(np.arange(-90,120,30.))
plt.title("Map of India")
plt.show()
```

Map of India



```
In [ ]: from mpl_toolkits.basemap import Basemap
        import matplotlib.pyplot as plt
        import numpy as np

        plt.figure(figsize=(10,8))

        # Create USA basemap
        m = Basemap(
            projection='merc',
            llcrnrlat=24,       # southern latitude of USA
            urcrnrlat=50,       # northern latitude of USA
            llcrnrlon=-125,     # western longitude of USA
            urcrnrlon=-66,      # eastern longitude of USA
            resolution='l'
        )

        m.drawcoastlines()
```

```
m.drawcountries()
m.drawstates()
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='lightyellow', lake_color='aqua')

m.drawparallels(np.arange(20, 55, 5), labels=[1,0,0,0])
m.drawmeridians(np.arange(-130, -60, 10), labels=[0,0,0,1])

plt.title("Map of United States with States")
plt.show()
```

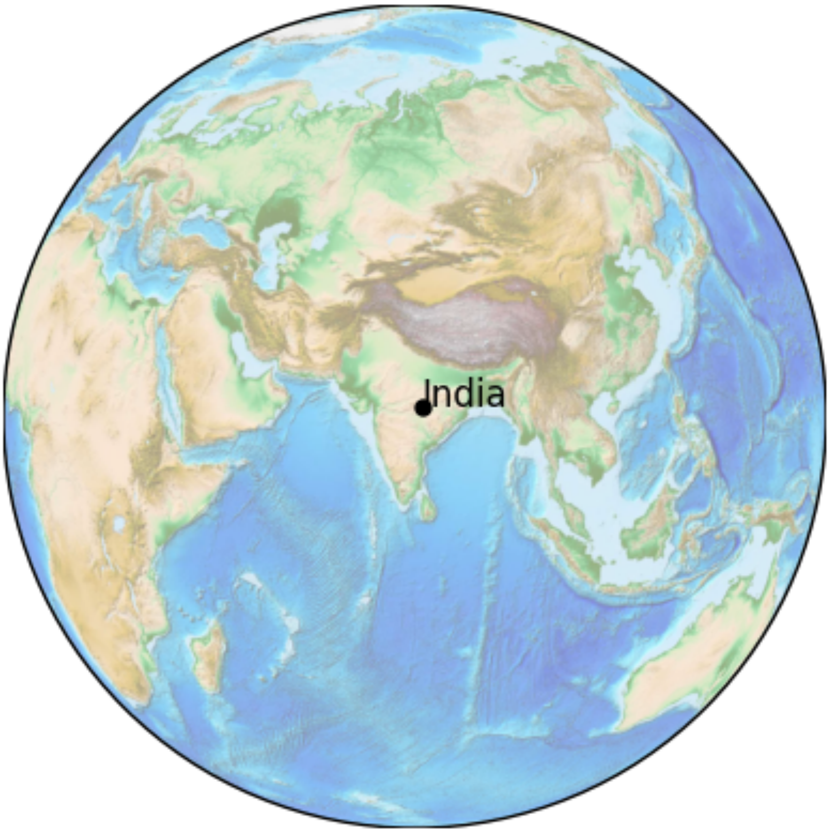


```
In [ ]: fig = plt.figure(figsize=(5,5))
        m=Basemap(projection='ortho',resolution='c',width=15.e6,height=15.e
        m.etopo(scale=0.5,alpha=0.5)
        x,y=m(80,22)
        plt.plot(x,y,'ok',markersize=5)
        plt.text(x,y,'India',fontsize=12)
```

```
warning: width and height keywords ignored for Orthographic projecti
on
```

```
Clipping input data to the valid range for imshow with RGB data ([
0..1] for floats or [0..255] for integers). Got range [0.0..1.000000
0000000002].
```

```
Out[ ]: Text(6477914.192763603, 6527695.018478486, 'India')
```

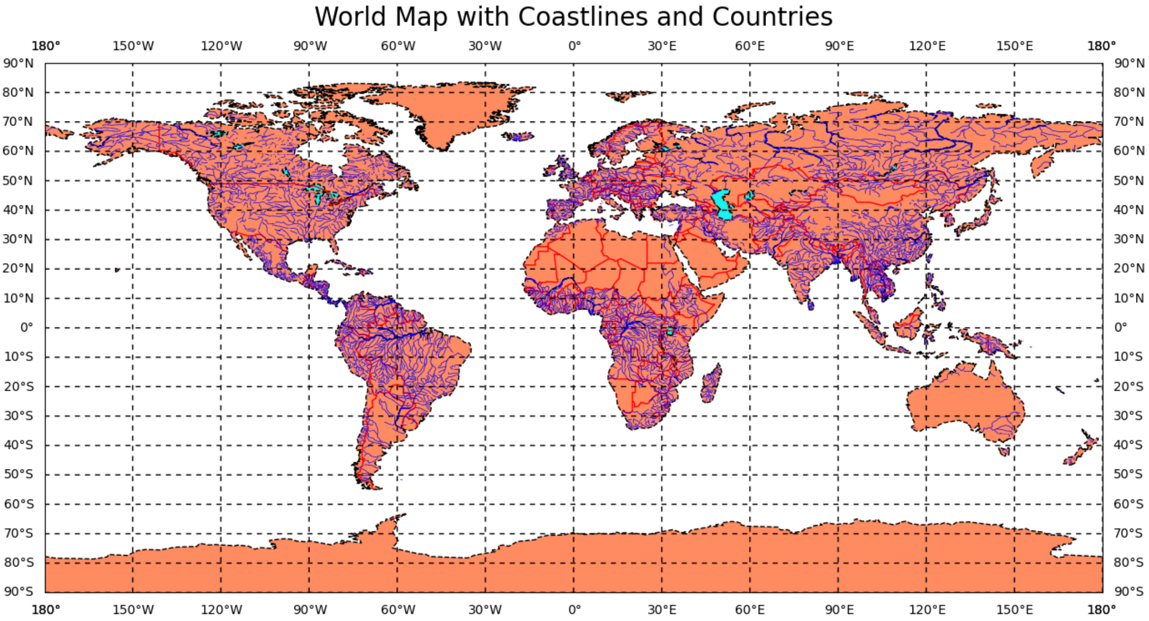


08/01/26

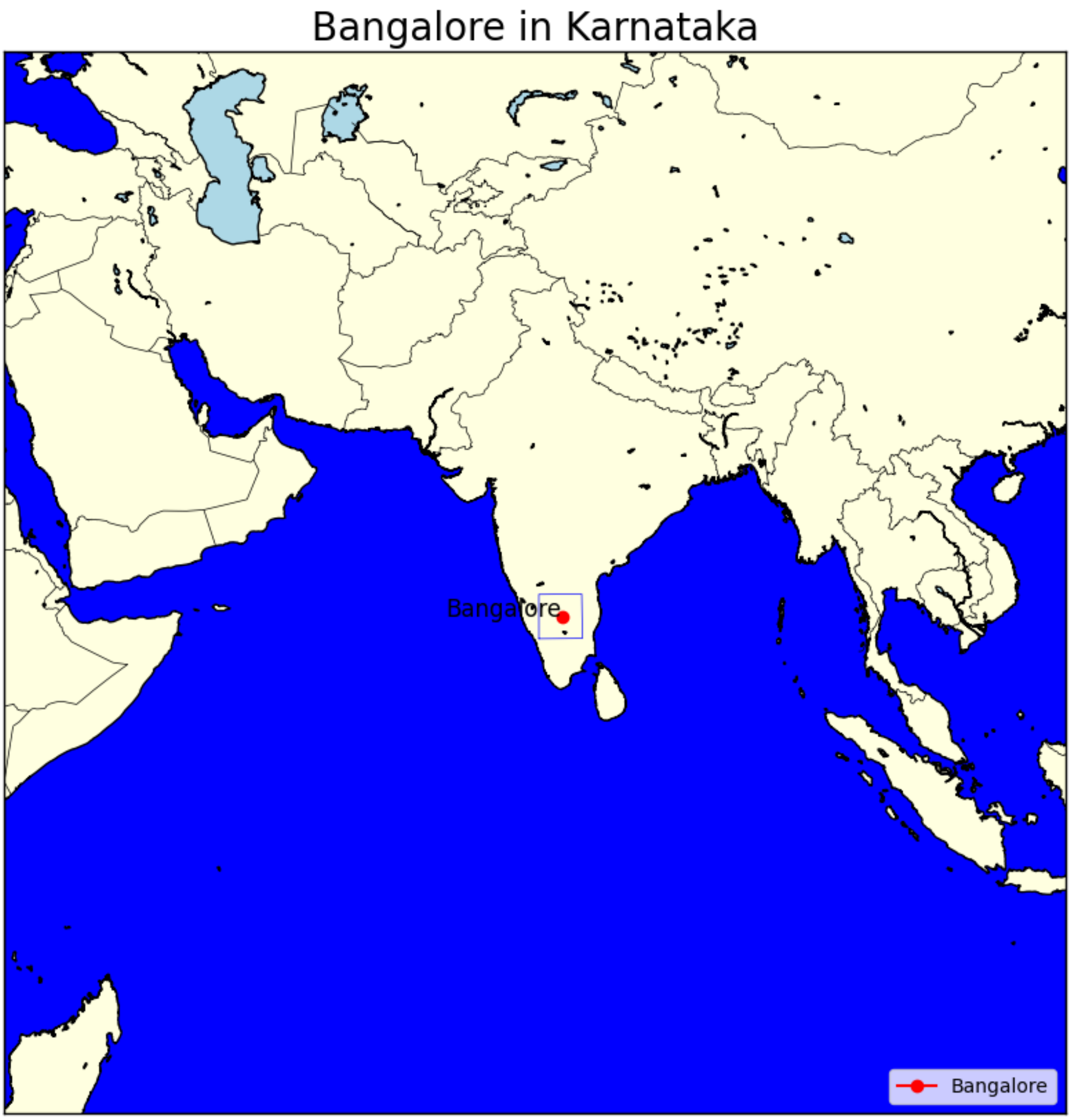
In [ ]:
```
#pip install basemap basemap-data-hires
#pip install geopandas
#pip install rasterio
#pip install contextily

#optional pip install numpy matplotlib pyproj shapely fiona
```

In [ ]:
```
#pip install rasterio
#Rasterio: access to geospatial raster data
#pip install contextily
#contextily is a small Python 3 (3.8 and above) package to retrieve
#It can add those tiles as basemap to matplotlib figures or write t
import pandas as pd
import matplotlib
import geopandas as gpd
import contextily as ctx
```

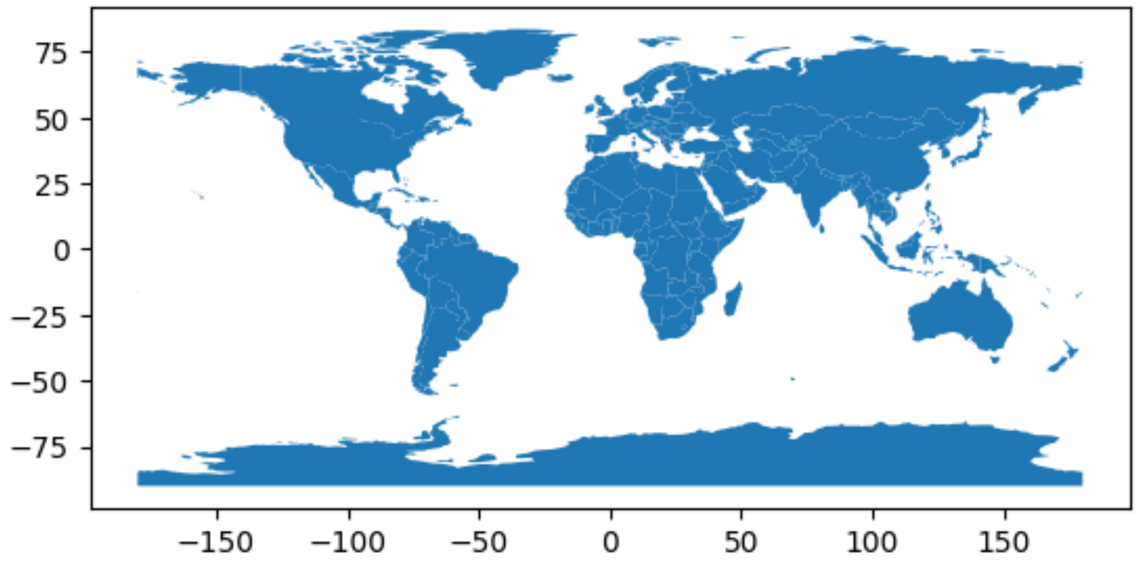
In [ ]:
```
fig= plt.figure(figsize=(15,15))
m = Basemap()
m.drawcoastlines(linewidth=1.0,linestyle='dashed',color='k')
m.drawcountries(linewidth=1.0,linestyle='solid',color='r')
m.fillcontinents(color='coral',lake_color='aqua',alpha=0.9)
m.drawrivers(linewidth=0.5,linestyle='solid', color='#0000FF')
m.drawmeridians(range(0,360,30),color='k',linewidth=1.0,dashes=[4,4
m.drawparallels(range(-90,100,10),color='k',linewidth=1.0,dashes=[4
plt.title("World Map with Coastlines and Countries",fontsize=20,pad
plt.show()
```



```
In [ ]: fig= plt.figure(figsize=(10,10))
        m = Basemap(projection='lcc',resolution='i',width=8E6,height=8E6,la
        #m=Basemap(projection='merc',llcrnlat=11.5,urcnlat=18.5,llcrnlon=74
        m.drawcoastlines()
        m.drawcountries()
        m.drawstates(linewidth=0.5,linestyle='solid',color='k')
        m.drawmapboundary(fill_color='blue')
        m.fillcontinents(color='lightyellow',lake_color='lightblue')
        karnatakaboundary=[
                [76,11.5],[76,14.5],
                [79,14.5],[79,11.5],
                [76,11.5]
        ]
        x,y=m(*zip(*karnatakaboundary))
        m.plot(x,y,marker=None,color='blue',linewidth=.5)
        bangalore_cords = [77.5946, 12.9716]
        x, y = m(*bangalore_cords)
        m.plot(x,y,marker='o',color='red',label='Bangalore')
        plt.text(x,y,'Bangalore',fontsize=12,ha='right')
        plt.title("Bangalore in Karnataka",fontsize=20)
        plt.legend()
        plt.show()
```



```
In [ ]: import geopandas as gpd
        url = 'https://naturalearth.s3.amazonaws.com/110m_cultural/ne_110m_
        world_gdf = gpd.read_file('ne_110m_admin_0_countries')
        world_gdf.plot()

Out[ ]: <Axes: >
```



In [ ]:
```
world_gdf.crs
```

Out[ ]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [ ]:
```
print(world_gdf.columns)
```

```
Index(['featurecla', 'scalerank', 'LABELRANK', 'SOVEREIGNT', 'SOV_A
3',
       'ADM0_DIF', 'LEVEL', 'TYPE', 'TLC', 'ADMIN',
       ...
       'FCLASS_TR', 'FCLASS_ID', 'FCLASS_PL', 'FCLASS_GR', 'FCLASS_I
T',
       'FCLASS_NL', 'FCLASS_SE', 'FCLASS_BD', 'FCLASS_UA', 'geometr
y'],
      dtype='object', length=169)
```

In [ ]:
```
#calculate density
world_gdf['pop_density']= world_gdf.POP_EST/world_gdf.area * 10**6
world_gdf.sort_values(by='pop_density', ascending=False)
```

```
/var/folders/0z/v3sdyt153lq3dbr23_yy2pxc0000gn/T/ipykernel_9463/3725
296465.py:2: UserWarning:

Geometry is in a geographic CRS. Results from 'area' are likely inco
rrect. Use 'GeoSeries.to_crs()' to re-project geometries to a projec
ted CRS before this operation.
```

Out[ ]:

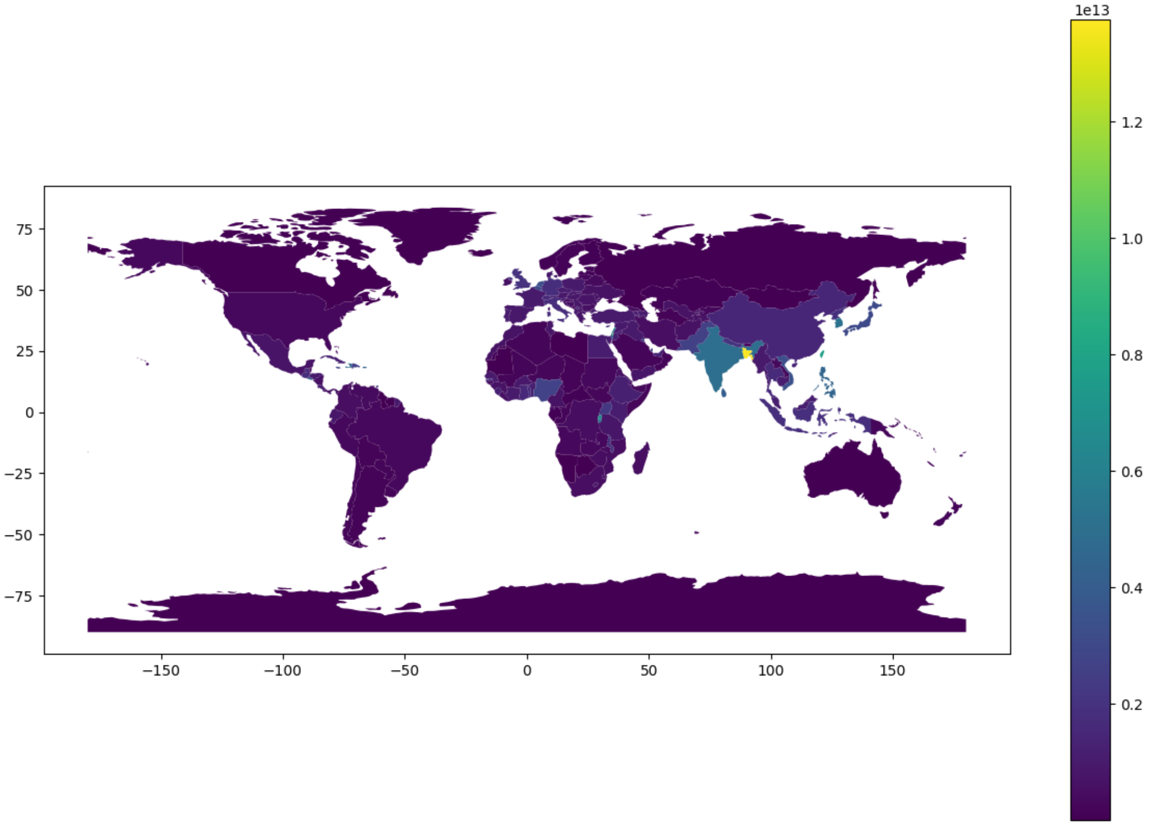
	featurecla	scalerank	LABELRANK	SOVEREIGNT	SOV_A3	ADM0_DIF

99	Admin-0 country	1	3	Bangladesh	BGD	0
79	Admin-0 country	1	5	Israel	IS1	1
140	Admin-0 country	1	3	Taiwan	TWN	0
77	Admin-0 country	1	5	Lebanon	LBN	0
169	Admin-0 country	1	3	Rwanda	RWA	0
...	...	...	...	...	...	...
144	Admin-0 country	1	3	Iceland	ISL	0
20	Admin-0 country	1	5	United Kingdom	GB1	1
23	Admin-0 country	3	6	France	FR1	1
22	Admin-0 country	1	3	Denmark	DN1	1
159	Admin-0 country	1	4	Antarctica	ATA	0

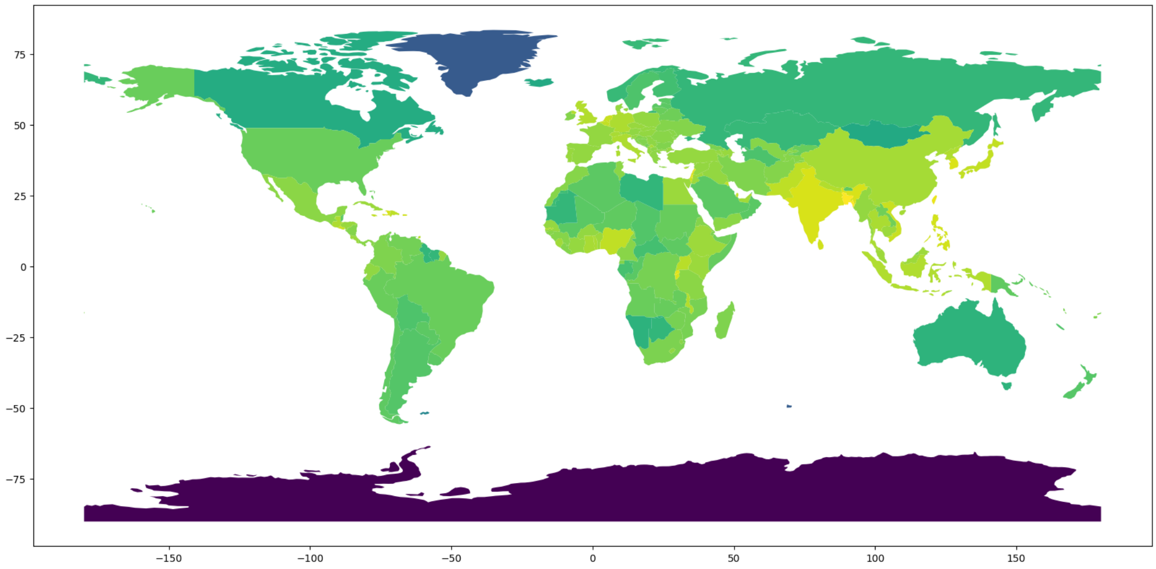
177 rows × 170 columns

In [ ]:
```
world_gdf.plot('pop_density', figsize=(15,10), legend=True,)
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
norm = matplotlib.colors.LogNorm(vmin=world_gdf.pop_density.min(),
world_gdf.to_crs('epsg:4326').plot("pop_density",
                                    figsize=(20,18),
                                    legend=False,
                                    norm=norm);
```



In [ ]:
```
#load the shape file
gdf_districts=gpd.read_file('District/District.shp')
gdf_districts.head(5)
```

Out[ ]:

	KGISDistri	LGD_Distri	KGISDist_1	BhuCodeDis	created_us	created_da
0	01	527	Belagavi	01	None	NaT
1	02	524	Bagalkot	02	None	NaT
2	03	530	Vijayapura	03	None	NaT
3	04	538	Kalburgi	04	None	NaT
4	05	529	Bidar	05	None	NaT

In [ ]:

```
gdf_districts.KGISDist_1
```

Out[ ]:
```
0              Belagavi
1              Bagalkot
2            Vijayapura
3              Kalburgi
4                 Bidar
5               Raichur
6                Koppal
7                 Gadag
8               Dharwad
9        Uttara Kannada
10               Haveri
11              Ballari
12          Chitradurga
13           Davanagere
14           Shivamogga
15                Udupi
16       Chikkamagaluru
17             Tumakuru
18               Kolara
19    Bengaluru (Urban)
20    Bengaluru (Rural)
21               Mandya
22               Hassan
23     Dakshina Kannada
24               Kodagu
25               Mysuru
26      Chamarajanagara
27      Chikkaballapura
28           Ramanagara
29               Yadgir
30         Vijayanagara
Name: KGISDist_1, dtype: object
```

In [ ]:
```
print(type(gdf_districts))
```

```
<class 'pandas.core.frame.DataFrame'>
```

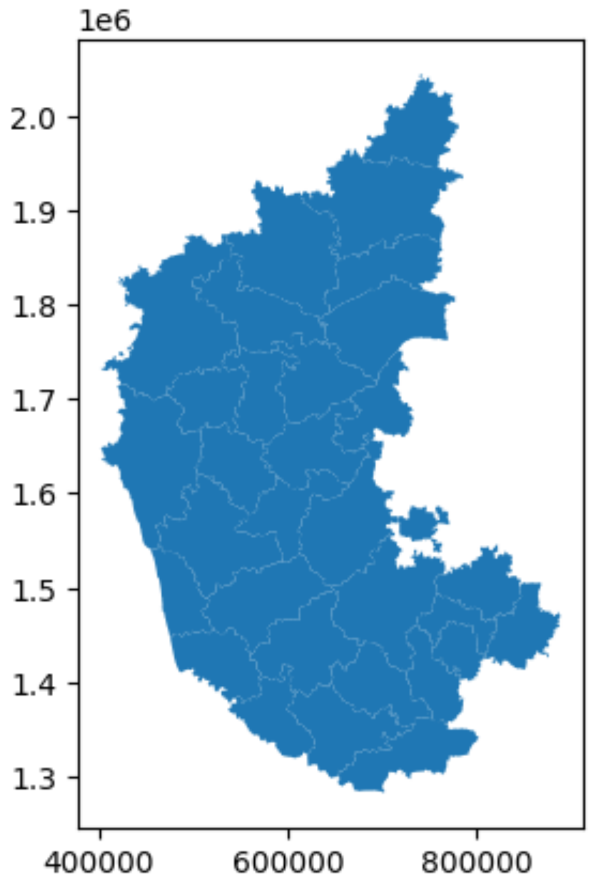
In [ ]:
```
print(gdf_districts.columns)
```

```
Index(['District', 'population'], dtype='object')
```

In [ ]:
```
import geopandas as gpd

gdf_districts = gpd.GeoDataFrame(
    gdf_districts,
    geometry="geometry"
)

gdf_districts = gdf_districts.set_geometry("geometry")
```

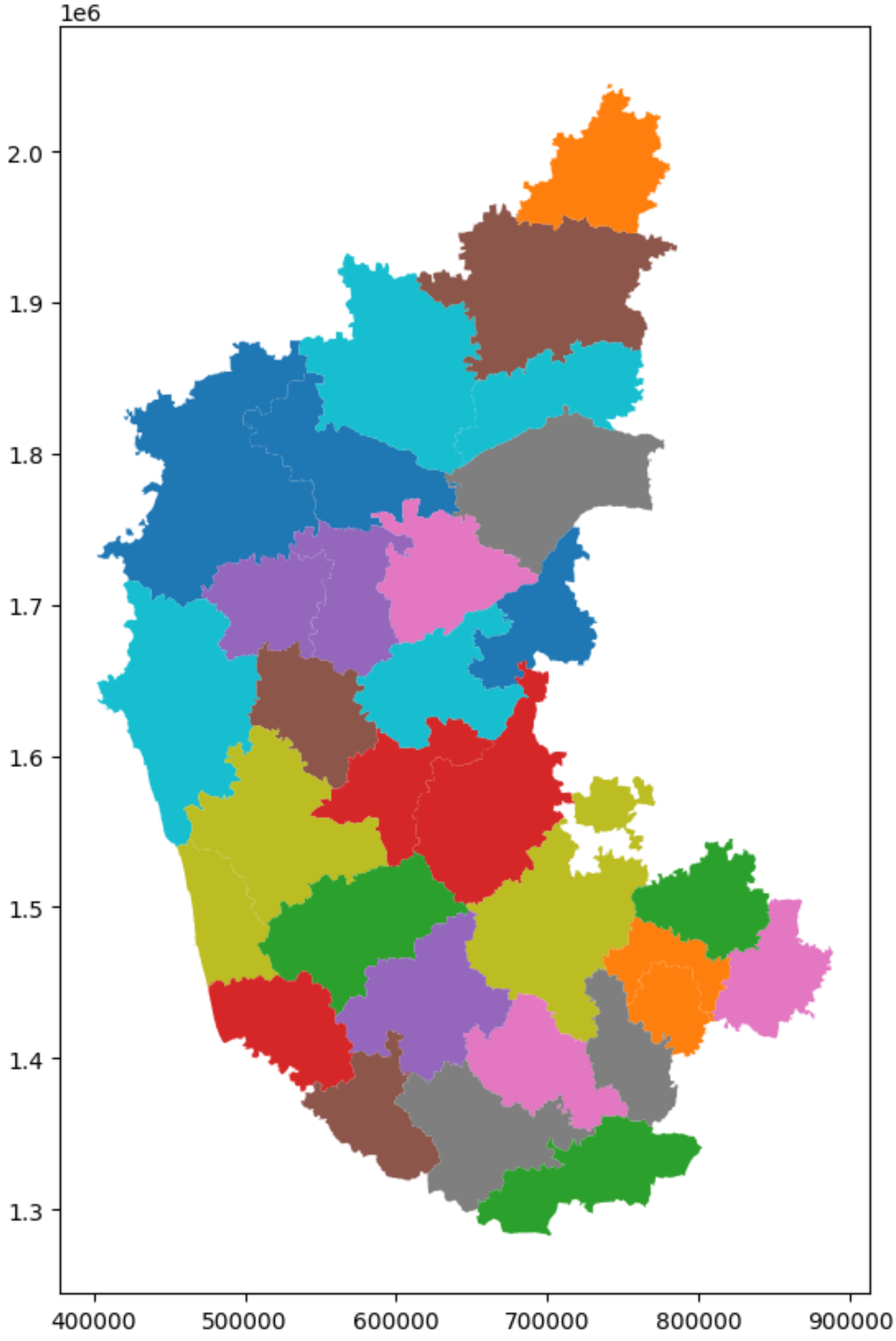
In [ ]:
```
gdf_districts.plot()
```

Out[ ]:
```
<Axes: >
```



```
In [ ]: gdf_districts.plot(figsize=(10,10),column='KGISDist_1')
```

```
Out[ ]: <Axes: >
```



```
In [ ]: df = pd.read_csv('/Users/manasagowda/Desktop/3MCA/APP/Karnataka-Dis
        df.head(5)
```

Out[ ]:

	District	population
0	Bagalkot	83973
1	Bidar	59898
2	Bengaluru (Rural)	17931
3	Bengaluru (Urban)	414125
4	Belagavi	100481

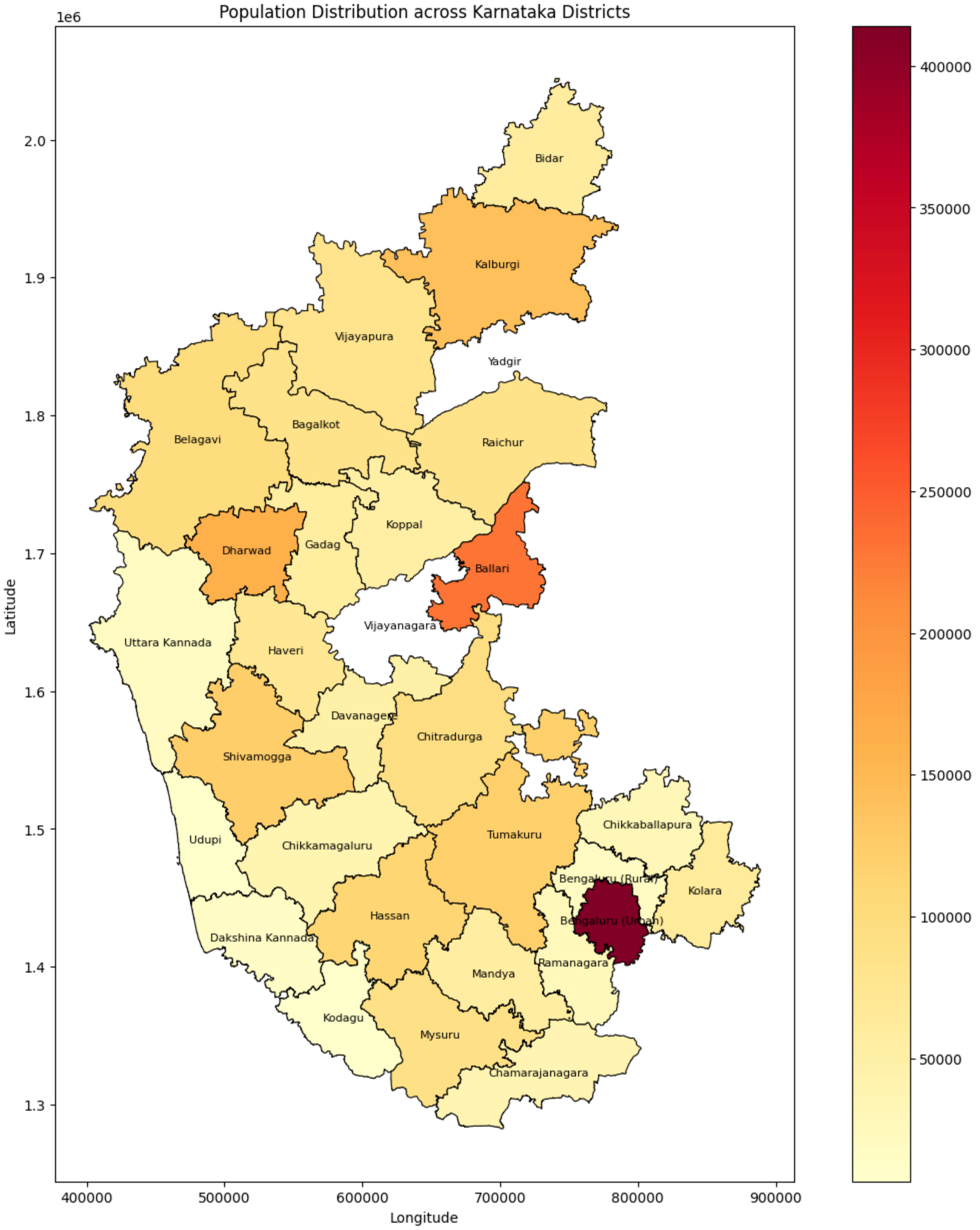
In [ ]:
```
gdf_merged = gdf_districts.merge(df,left_on='KGISDist_1', right_on=
gdf_merged.head()
```

Out[ ]:

	KGISDistri	LGD_Distri	KGISDist_1	BhuCodeDis	created_us	created_da
0	01	527	Belagavi	01	None	NaT
1	02	524	Bagalkot	02	None	NaT
2	03	530	Vijayapura	03	None	NaT
3	04	538	Kalburgi	04	None	NaT
4	05	529	Bidar	05	None	NaT

In [ ]:
```
fig,ax=plt.subplots(figsize=(15,15))
cmap = plt.cm.get_cmap('YlOrRd')
cmap.set_bad('white')
normalize = colors.Normalize(vmin = gdf_merged['population'].min(),

for x,y,label in zip(gdf_merged.centroid.x, gdf_merged.centroid.y,
    ax.text(x, y, label, fontsize=8, ha='center', va='center')
gdf_districts.plot(ax=ax, column=gdf_merged['population'], cmap=cma

ax.set_title('Population Distribution across Karnataka Districts')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.show()
```

```
/var/folders/0z/v3sdyt153lq3dbr23_yy2pxc0000gn/T/ipykernel_9463/2748
327839.py:2: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be r
emoved in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.c
olormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```


Population Distribution across Karnataka Districts
1e6
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
Latitude
400000
500000
600000
700000
800000
900000
Longitude
Bidar
Kalburgi
Vijayapura
Yadgir
Bagalkot
Belagavi
Raichur
Koppal
Gadag
Dharwad
Ballari
Vijayanagara
Uttara Kannada
Haveri
Davanagere
Chitradurga
Shivamogga
Chikkaballapura
Tumakuru
Udupi
Chikkamagaluru
Bengaluru (Rural)
Kolara
Bengaluru (Urban)
Hassan
Dakshina Kannada
Ramanagara
Mandya
Kodagu
Mysuru
Chamarajanagara
400000
350000
300000
250000
200000
150000
100000
50000