MANASA S T 2547133

ADVANCED PYTHON PROGRAMMING

LAB 1

Apply 3D visualization concepts

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

1. Import the “mortality” dataset.

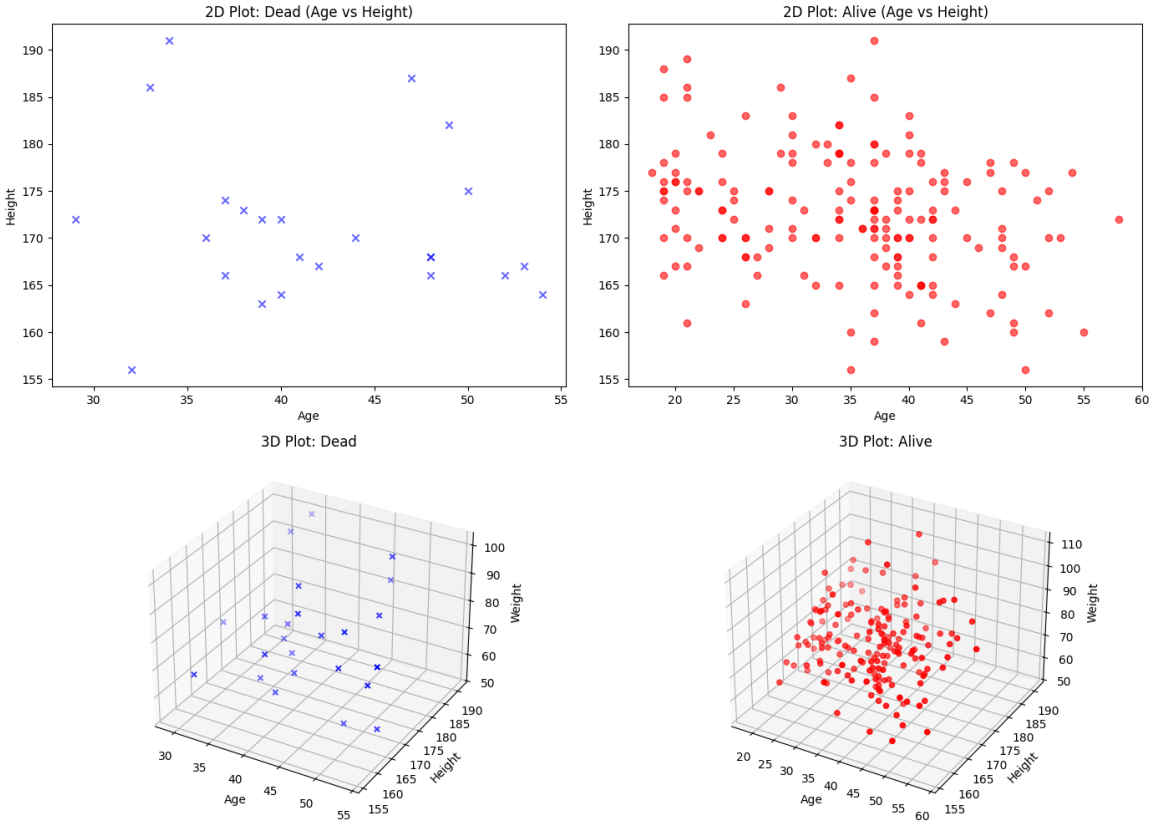
In [3]:
```
df = pd.read_csv('MortalityDataset.csv')
```

In [4]:
```
alive_data = df[df['MORT'] == 'alive']
dead_data = df[df['MORT'] == 'dead']
```

2. Generate an image of four scatter plots(2D plot for alive, 2D plot for notalive, 3D plot for alive, 3D plot for notalive)

In [8]:
```
fig = plt.figure(figsize=(14, 10))

ax2 = fig.add_subplot(2, 2, 1)
ax2.scatter(dead_data['AGE'], dead_data['HEIGHT'], c='blue', marker
ax2.set_title('2D Plot: Dead (Age vs Height)')
ax2.set_xlabel('Age')
ax2.set_ylabel('Height')

ax1 = fig.add_subplot(2, 2, 2)
ax1.scatter(alive_data['AGE'], alive_data['HEIGHT'], c='red', marke
ax1.set_title('2D Plot: Alive (Age vs Height)')
ax1.set_xlabel('Age')
ax1.set_ylabel('Height')


ax4 = fig.add_subplot(2, 2, 3, projection='3d')
ax4.scatter(dead_data['AGE'], dead_data['HEIGHT'], dead_data['WEIGH
ax4.set_title('3D Plot: Dead')
ax4.set_xlabel('Age')
ax4.set_ylabel('Height')
ax4.set_zlabel('Weight')

ax3 = fig.add_subplot(2, 2, 4, projection='3d')
ax3.scatter(alive_data['AGE'], alive_data['HEIGHT'], alive_data['WE
```

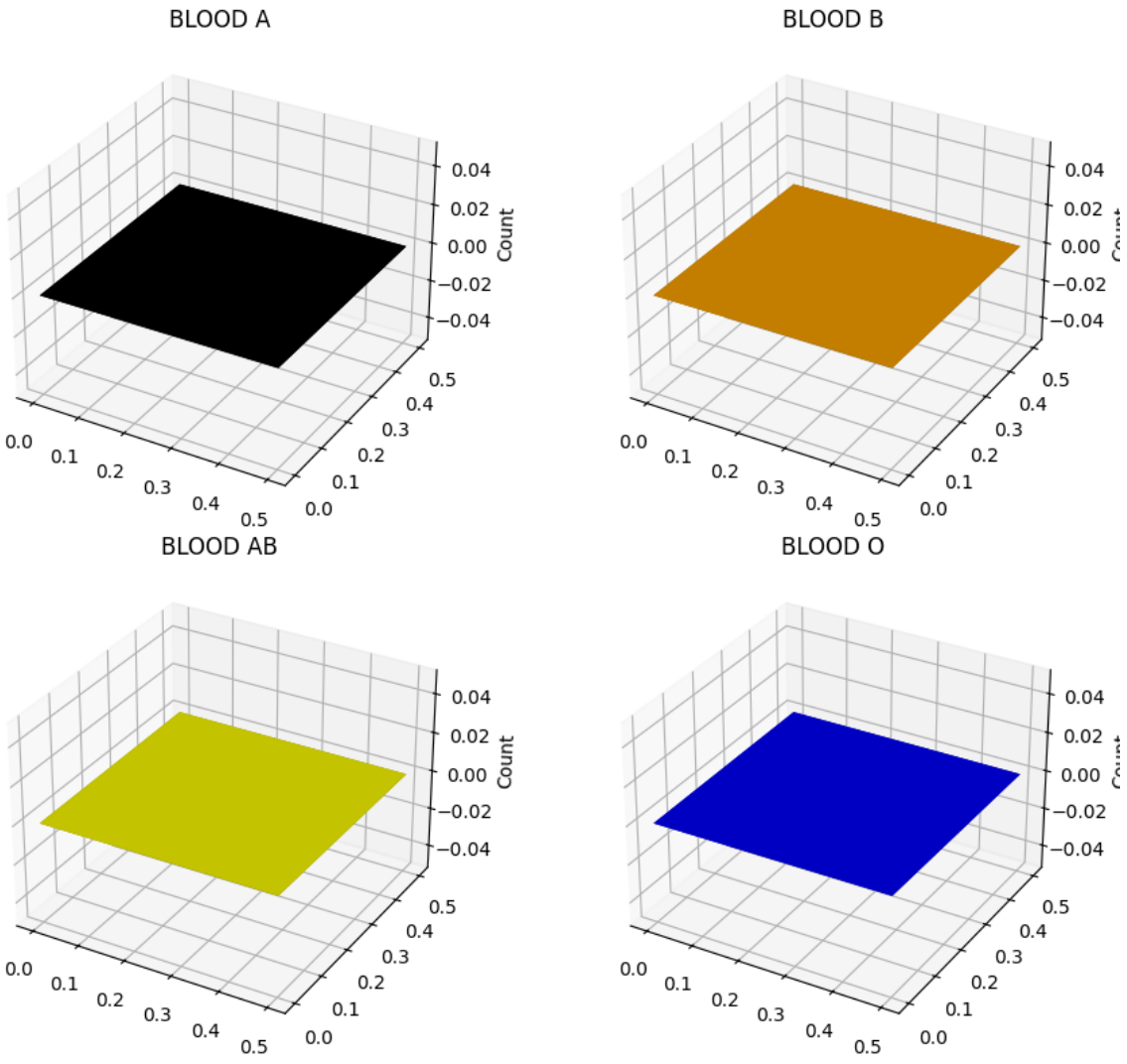
```
ax3.set_title('3D Plot: Alive')
ax3.set_xlabel('Age')
ax3.set_ylabel('Height')
ax3.set_zlabel('Weight')

plt.tight_layout()
plt.show()
```



3. Generate an image of four bar plots(3D plot for BLOOD-A , 3D plot for BLOOD-B, 3D plot for BLOOD-AB, 3D plot for BLOOD-O)

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

count_A  = len(df[df['BLOOD'] == 'A'])
count_B  = len(df[df['BLOOD'] == 'B'])
count_AB = len(df[df['BLOOD'] == 'AB'])
count_O  = len(df[df['BLOOD'] == 'O'])

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

ax1 = fig.add_subplot(2,2,1, projection='3d')
ax1.bar3d(0, 0, 0, 0.5, 0.5, count_A, color='black')
ax1.set_title("BLOOD A")
ax1.set_zlabel("Count")

ax2 = fig.add_subplot(2,2,2, projection='3d')
ax2.bar3d(0, 0, 0, 0.5, 0.5, count_B, color='orange')
ax2.set_title("BLOOD B")
ax2.set_zlabel("Count")
```

```
ax3 = fig.add_subplot(2,2,3, projection='3d')
ax3.bar3d(0, 0, 0, 0.5, 0.5, count_AB, color='yellow')
ax3.set_title("BLOOD AB")
ax3.set_zlabel("Count")

ax4 = fig.add_subplot(2,2,4, projection='3d')
ax4.bar3d(0, 0, 0, 0.5, 0.5, count_O, color='blue')
ax4.set_title("BLOOD O")
ax4.set_zlabel("Count")

plt.tight_layout()
plt.show()
```

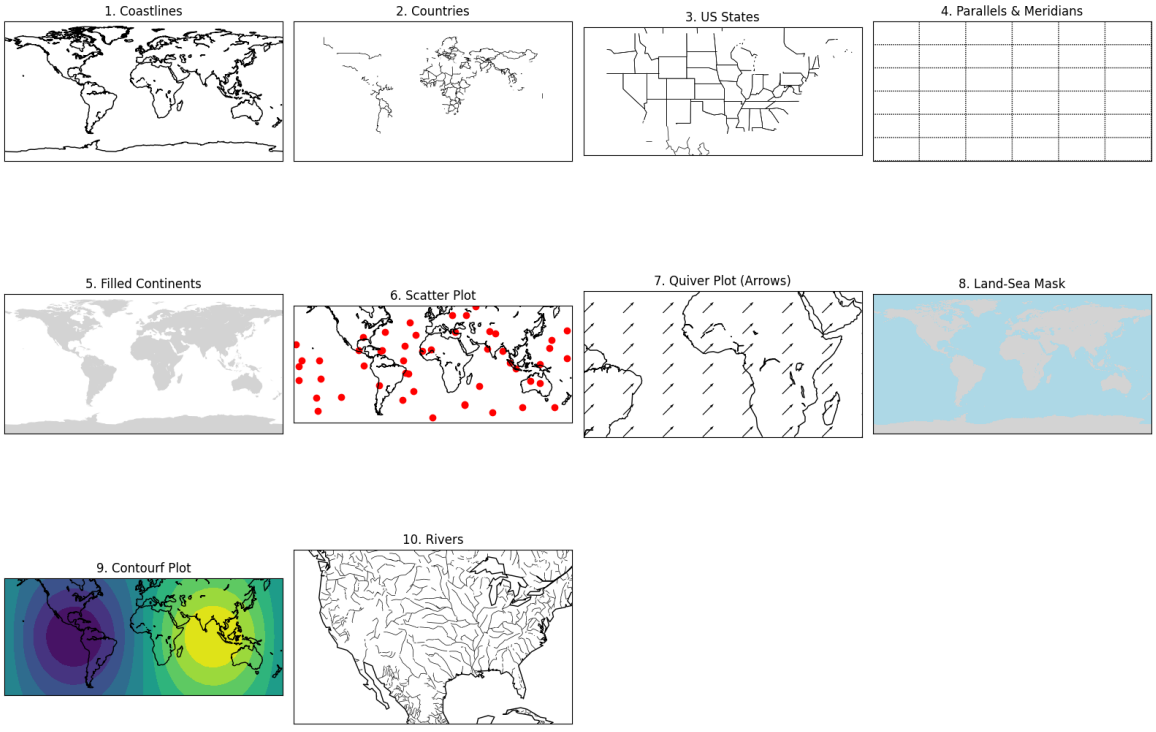


4. Explore the Basemap library documentation. Draw any 10 types of plot using various functions. Make sure each graph is drawn using a unique function.

In [16]:
```
from mpl_toolkits.basemap import Basemap
import numpy as np

fig = plt.figure(figsize=(16, 12))

# 1. Basic map with coastlines
```

```
ax1 = fig.add_subplot(3, 4, 1)
m1 = Basemap(ax=ax1)
m1.drawcoastlines()
ax1.set_title('1. Coastlines')

# 2. Map with countries
ax2 = fig.add_subplot(3, 4, 2)
m2 = Basemap(ax=ax2)
m2.drawcountries()
ax2.set_title('2. Countries')

# 3. Map with states (US)
ax3 = fig.add_subplot(3, 4, 3)
m3 = Basemap(projection='merc', llcrnrlat=25, urcrnrlat=50, llcrnrl
m3.drawstates()
ax3.set_title('3. US States')

# 4. Map with parallels and meridians
ax4 = fig.add_subplot(3, 4, 4)
m4 = Basemap(ax=ax4)
m4.drawparallels(np.arange(-90, 120, 30))
m4.drawmeridians(np.arange(0, 360, 60))
ax4.set_title('4. Parallels & Meridians')

# 5. Map with filled continents
ax5 = fig.add_subplot(3, 4, 5)
m5 = Basemap(ax=ax5)
m5.fillcontinents(color='lightgray')
ax5.set_title('5. Filled Continents')

# 6. Scatter plot on map
ax6 = fig.add_subplot(3, 4, 6)
m6 = Basemap(projection='merc', llcrnrlat=-60, urcrnrlat=60, llcrnr
m6.drawcoastlines()
lons = np.random.uniform(-180, 180, 50)
lats = np.random.uniform(-60, 60, 50)
x, y = m6(lons, lats)
m6.scatter(x, y, marker='o', color='red')
ax6.set_title('6. Scatter Plot')

# 7. Quiver plot (arrows)
ax7 = fig.add_subplot(3, 4, 7)
m7 = Basemap(projection='merc', llcrnrlat=-30, urcrnrlat=30, llcrnr
m7.drawcoastlines()
lons_grid = np.linspace(-60, 60, 8)
lats_grid = np.linspace(-30, 30, 8)
for lon in lons_grid:
    for lat in lats_grid:
        x, y = m7(lon, lat)
        m7.quiver(x, y, 1, 1, width=0.003)
ax7.set_title('7. Quiver Plot (Arrows)')

# 8. Drawlsmask (land-sea mask)
ax8 = fig.add_subplot(3, 4, 8)
m8 = Basemap(ax=ax8)
m8.drawlsmask(land_color='lightgray', ocean_color='lightblue')
```

```
ax8.set_title('8. Land-Sea Mask')

# 9. Contourf plot on map
ax9 = fig.add_subplot(3, 4, 9)
m9 = Basemap(projection='merc', llcrnrlat=-60, urcrnrlat=60, llcrnr
m9.drawcoastlines()
lons_cont = np.linspace(-180, 180, 30)
lats_cont = np.linspace(-60, 60, 30)
lons_grid, lats_grid = np.meshgrid(lons_cont, lats_cont)
data = np.sin(np.radians(lons_grid)) * np.cos(np.radians(lats_grid)
x_cont, y_cont = m9(lons_grid, lats_grid)
m9.contourf(x_cont, y_cont, data, levels=10, cmap='viridis')
ax9.set_title('9. Contourf Plot')

# 10. Drawrivers
ax10 = fig.add_subplot(3, 4, 10)
m10 = Basemap(projection='merc', llcrnrlat=20, urcrnrlat=50, llcrnr
m10.drawcoastlines()
m10.drawrivers()
ax10.set_title('10. Rivers')

plt.tight_layout()
plt.show()
```



Demonstrate Plots with Maps

1. Identify the district-wise COVID death details of any state from the authorized source for a particular month/day of 2024.

In [17]:
```
import geopandas as gpd
import matplotlib.colors as colors
```

In [23]:
```
gdf1 = gpd.read_file("mh1.geojson")
gdf2 = gpd.read_file("mh2.geojson")

gdf_districts = pd.concat([gdf1, gdf2], ignore_index=True)
gdf_districts.head()
```

Out[23]:

	SUB_DIST	DISTRICT	STATE	TYPE	NAME	CEN_20(
0	Gondia	Gondia	Maharashtra	village	Chargaon	2750704046012766(
1	Gondia	Gondia	Maharashtra	village	Birsola	2750704046012712(
2	Gondia	Gondia	Maharashtra	village	Kasa	2750704046012711(
3	Gondia	Gondia	Maharashtra	village	Bhadyatola	2750704046012713(
4	Gondia	Gondia	Maharashtra	village	Satona	2750704046012751(

In [24]:
```
print(gdf_districts.columns)
```

```
Index(['SUB_DIST', 'DISTRICT', 'STATE', 'TYPE', 'NAME', 'CEN_2001',
       'geometry'],
      dtype='object')
```

2. Fetch the corresponding state map details from the “State Geographical Information System” portal.

In [26]:
```
df = pd.read_csv('Maharastra.csv')
print(df.head(5))
```

```
     Districts  Positive Cases  Active Cases  Recovered  Deceased  \
0  Ahmednagar          377661            16     370403      7242
1       Akola           66181             0      64711      1470
2    Amravati          105943             0     104320      1623
3  Aurangabad          176521             3     172234      4284
4        Beed          109179             3     106293      2883

   Recovery Rate(%)  Fatality Rate(%)
0              98.1               1.9
1              97.8               2.2
2              98.5               1.5
3              97.6               2.4
4              97.4               2.6
```

In [35]:
```
gdf_districts['district_norm'] = gdf_districts['DISTRICT'].str.lowe
df['district_norm'] = df['Districts'].str.lower().str.strip()
```

In [36]:
```
gdf_district_level = gdf_districts.dissolve(
    by='district_norm',
    as_index=False
)
```

In [37]:
```
gdf_merged = gdf_district_level.merge(
    df,
    on='district_norm',
    how='left'
)

print(gdf_merged.head())
```

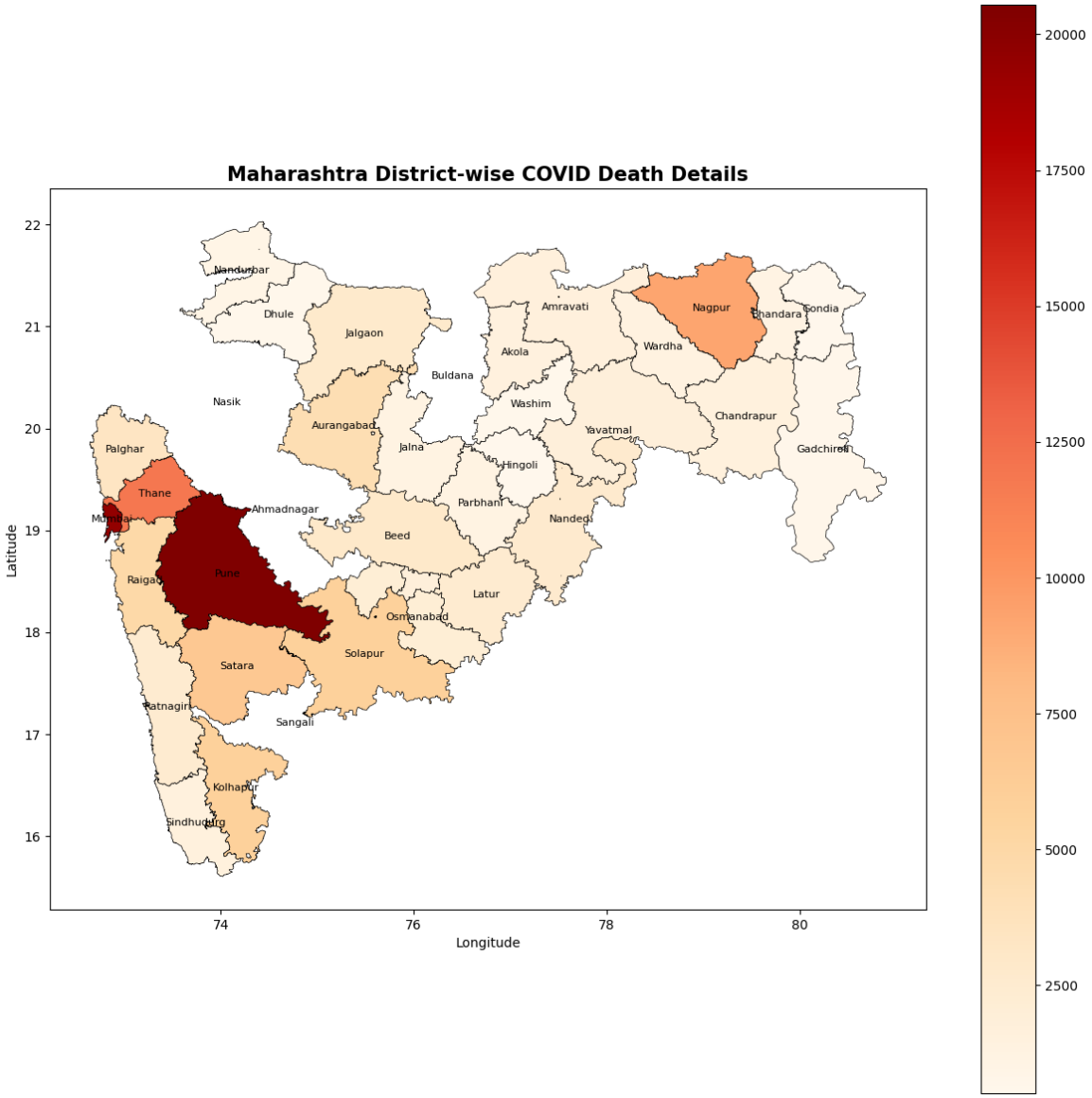
```
  district_norm                                           geometry
\
0    ahmadnagar  MULTIPOLYGON (((73.29698 19.07493, 73.29705 19...
1         akola  MULTIPOLYGON (((76.787 20.29781, 76.78727 20.2...
2      amravati  POLYGON ((76.80936 21.18648, 76.80763 21.18627...
3    aurangabad  MULTIPOLYGON (((74.88156 19.65448, 74.88086 19...
4          beed  POLYGON ((75.1639 18.66838, 75.16337 18.6658, ...

          SUB_DIST    DISTRICT        STATE     TYPE       NAME  \
0         Shevgaon  Ahmadnagar  Maharashtra  village      Mungi
1       Murtijapur       Akola  Maharashtra  village    Jitapur
2  Anjangaon Surji    Amravati  Maharashtra  village  Murha Bk.
3           Kannad  Aurangabad  Maharashtra  village     Digaon
4            Parli        Beed  Maharashtra  village    Borkhed

             CEN_2001   Districts  Positive Cases  Active Cases  Rec
overed  \
0  275220420703272600         NaN             NaN           NaN
NaN
1  275010399200523900       Akola         66181.0           0.0    6
4711.0
2  275030400300678800    Amravati        105943.0           0.0   10
4320.0
3  275150413302321400  Aurangabad        176521.0           3.0   17
2234.0
4  275230422403472000        Beed        109179.0           3.0   10
6293.0

   Deceased  Recovery Rate(%)  Fatality Rate(%)
0       NaN               NaN               NaN
1    1470.0              97.8               2.2
2    1623.0              98.5               1.5
3    4284.0              97.6               2.4
4    2883.0              97.4               2.6
```

3. Plot the district-wise COVID death details in a state map with proper formatting using GeoPandas.

In [38]:
```
fig, ax = plt.subplots(1, 1, figsize=(15, 15))

# Color map
cmap = plt.cm.get_cmap('OrRd')
cmap.set_bad(color='lightgrey')

# Normalize deaths
normalize = colors.Normalize(
    vmin=gdf_merged['Deceased'].min(),
    vmax=gdf_merged['Deceased'].max()
)

# Plot district-wise COVID deaths
gdf_merged.plot(
    column='Deceased',
    cmap=cmap,
    norm=normalize,
```

```
    ax=ax,
    legend=True,
    edgecolor='black',
    linewidth=0.5
)

# Add district labels (use DISTRICT column)
for x, y, label in zip(
    gdf_merged.geometry.centroid.x,
    gdf_merged.geometry.centroid.y,
    gdf_merged['DISTRICT']
):
    ax.text(x, y, label, fontsize=8, ha='center', va='center')

# Titles and labels
ax.set_title(
    'Maharashtra District-wise COVID Death Details',
    fontdict={'fontsize': 15, 'fontweight': 'bold'}
)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

plt.show()
```

```
/var/folders/0z/v3sdyt153lq3dbr23_yy2pxc0000gn/T/ipykernel_84642/311
0763821.py:4: MatplotlibDeprecationWarning: The get_cmap function wa
s deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``ma
tplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()`` or
``pyplot.get_cmap()`` instead.
  cmap = plt.cm.get_cmap('OrRd')
/var/folders/0z/v3sdyt153lq3dbr23_yy2pxc0000gn/T/ipykernel_84642/311
0763821.py:26: UserWarning: Geometry is in a geographic CRS. Results
from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re
-project geometries to a projected CRS before this operation.

  gdf_merged.geometry.centroid.x,
/var/folders/0z/v3sdyt153lq3dbr23_yy2pxc0000gn/T/ipykernel_84642/311
0763821.py:27: UserWarning: Geometry is in a geographic CRS. Results
from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re
-project geometries to a projected CRS before this operation.

  gdf_merged.geometry.centroid.y,
```



4. Provide the source URL of state COVID details and shapefile in the comment statement.

In [ ]:
```
# covid dataset : https://www.kaggle.com/datasets/anandhuh/latest-c
# state information : https://projects.datameet.org/indian_village_
```