Data Visualization Techniques Lab

Pulla Manoj Kumar
Assistant Professor, CSD, ACEEC Hyderabad

Visualization of various massive dataset

Finance - Healthcare - Census - Geospatial

5.1 - Finance

Multi-Company Stock Data Analysis using yFinance

STEP 1: Install Required Package

```
!pip install yfinance
```

STEP 2: Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf
```

STEP 3: Download Massive Finance Dataset

```
stocks = ['AAPL','MSFT','TSLA','AMZN']

data = yf.download(stocks,
                   start="2012-01-01",
                   end="2024-01-01")

data.head()
```

STEP 4: Display Dataset Information

```
data.info()
data.describe()
```

STEP 5: Closing Price Trend Visualization

```
plt.figure(figsize=(12,6))

for stock in stocks:
    plt.plot(data['Close'][stock], label=stock)

plt.title("Stock Closing Price Comparison")
plt.xlabel("Date")
plt.ylabel("Closing Price")
plt.legend()
plt.show()
```

STEP 6: Trading Volume Visualization

```
plt.figure(figsize=(12,6))

for stock in stocks:
    plt.plot(data['Volume'][stock], label=stock)

plt.title("Trading Volume Comparison")
plt.xlabel("Date")
plt.ylabel("Volume")
plt.legend()
plt.show()
```

STEP 7: Correlation Heatmap

```
close_data = data['Close']

plt.figure(figsize=(8,6))
sns.heatmap(close_data.corr(),
            annot=True,
            cmap="coolwarm")

plt.title("Correlation Between Stock Prices")
plt.show()
```

STEP 8: Monthly Average Closing Price

```
monthly = close_data.resample('M').mean()

monthly.plot(figsize=(12,6))
plt.title("Monthly Average Closing Price")
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()
```

STEP 9: Distribution of Closing Prices

```
close_data.plot(kind='hist',
                bins=40,
                alpha=0.5,
                figsize=(10,6))

plt.title("Distribution of Closing Prices")
plt.xlabel("Price")
plt.show()
```

Data Visualization Lab Exercise

5.2 - Healthcare

Healthcare Data Analysis using COVID-19 Global Dataset

STEP 1: Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

STEP 2: Load Massive Healthcare Dataset

```
url = "https://raw.githubusercontent.com/datasets/covid-19/main/data/countries-
    aggregated.csv"

data = pd.read_csv(url)

data.head()
```

STEP 3: Dataset Information

```
data.info()
data.describe()
```

STEP 4: Convert Date Column to Datetime Format

```
data['Date'] = pd.to_datetime(data['Date'])
data.sort_values('Date', inplace=True)
```

STEP 5: Filter Selected Countries

```
countries = ['India', 'US', 'Brazil', 'Russia']

filtered_data = data[data['Country'].isin(countries)]
```

STEP 6: Line Plot – Confirmed Cases

```
plt.figure(figsize=(12,6))

for country in countries:
    country_data = filtered_data[filtered_data['Country'] == country]
    plt.plot(country_data['Date'],
             country_data['Confirmed'],
             label=country)

plt.title("COVID-19 Confirmed Cases Over Time")
plt.xlabel("Date")
plt.ylabel("Confirmed Cases")
plt.legend()
plt.show()
```

STEP 7: Line Plot – Death Cases

```
plt.figure(figsize=(12,6))

for country in countries:
    country_data = filtered_data[filtered_data['Country'] == country]
    plt.plot(country_data['Date'],
             country_data['Deaths'],
             label=country)

plt.title("COVID-19 Death Cases Over Time")
plt.xlabel("Date")
plt.ylabel("Death Cases")
plt.legend()
plt.show()
```

STEP 8: Correlation Heatmap

```
plt.figure(figsize=(8,6))

sns.heatmap(data[['Confirmed',
                  'Recovered',
                  'Deaths']].corr(),
            annot=True,
            cmap='coolwarm')

plt.title("Correlation Between Healthcare Variables")
plt.show()
```

STEP 9: Histogram – Confirmed Cases Distribution

```
plt.figure(figsize=(8,5))
plt.hist(data['Confirmed'], bins=50)

plt.title("Distribution of Confirmed Cases")
plt.xlabel("Confirmed Cases")
plt.ylabel("Frequency")
plt.show()
```

STEP 10: Boxplot – Country-wise Comparison

```
plt.figure(figsize=(8,6))

sns.boxplot(x='Country',
            y='Confirmed',
            data=filtered_data)

plt.title("Country-wise Confirmed Case Distribution")
plt.show()
```

Data Visualization Lab Exercise

5.3 - Census

World Population Census Data Analysis

STEP 1: Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

STEP 2: Load Massive Census Dataset

```
url = "https://raw.githubusercontent.com/datasets/population/master/data/population.
    csv"

data = pd.read_csv(url)

data.head()
```

STEP 3: Dataset Information

```
data.info()
data.describe()
```

STEP 4: Filter Recent Years Data (After 2000)

```
data_recent = data[data['Year'] >= 2000]
```

STEP 5: Select Specific Countries

```
countries = ['India', 'China', 'United States', 'Brazil']

filtered_data = data_recent[
    data_recent['Country Name'].isin(countries)
]
```

STEP 6: Line Plot – Population Growth

```
plt.figure(figsize=(12,6))

for country in countries:
    country_data = filtered_data[
        filtered_data['Country Name'] == country
    ]
    plt.plot(country_data['Year'],
             country_data['Value'],
             label=country)

plt.title("Population Growth Over Time")
plt.xlabel("Year")
plt.ylabel("Population")
plt.legend()
plt.show()
```

STEP 7: Bar Plot – Latest Year Comparison

```
latest_year = filtered_data['Year'].max()
latest_data = filtered_data[
    filtered_data['Year'] == latest_year
]

plt.figure(figsize=(8,5))
plt.bar(latest_data['Country Name'],
        latest_data['Value'])

plt.title("Population Comparison in Latest Year")
plt.xlabel("Country")
plt.ylabel("Population")
plt.show()
```

STEP 8: Histogram – Population Distribution

```
plt.figure(figsize=(8,5))
plt.hist(data_recent['Value'], bins=50)

plt.title("Population Distribution")
plt.xlabel("Population")
plt.ylabel("Frequency")
plt.show()
```

STEP 9: Correlation Heatmap

```
pivot_data = filtered_data.pivot(
    index='Year',
    columns='Country Name',
    values='Value'
)

plt.figure(figsize=(8,6))
sns.heatmap(pivot_data.corr(),
            annot=True,
            cmap='coolwarm')

plt.title("Correlation Between Countries Population Growth")
plt.show()
```

Data Visualization Lab Exercise

5.4 - Geospatial

Geospatial Data Analysis using Earthquake Dataset

STEP 1: Install Required Libraries

```
!pip install geopandas folium
```

STEP 2: Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
import folium
```

STEP 3: Load Massive Geospatial Dataset

```
url = "https://raw.githubusercontent.com/plotly/datasets/master/earthquakes-23k.csv"

data = pd.read_csv(url)

data.head()
```

STEP 4: Dataset Information

```
data.info()
data.describe()
```

STEP 5: Scatter Plot – Longitude vs Latitude

```
plt.figure(figsize=(8,6))
plt.scatter(data['Longitude'],
            data['Latitude'],
            alpha=0.3)

plt.title("Geospatial Distribution of Earthquakes")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

STEP 6: Magnitude Distribution

```
plt.figure(figsize=(8,5))
plt.hist(data['Magnitude'], bins=40)

plt.title("Distribution of Earthquake Magnitude")
plt.xlabel("Magnitude")
plt.ylabel("Frequency")
plt.show()
```

STEP 7: Load World Map

```
world_url = "https://naturalearth.s3.amazonaws.com/110m_cultural/
    ne_110m_admin_0_countries.zip"

world = gpd.read_file(world_url)
```

STEP 8: Convert to GeoDataFrame

```
gdf = gpd.GeoDataFrame(
    data,
    geometry=gpd.points_from_xy(
        data.Longitude,
        data.Latitude
    ),
    crs="EPSG:4326"
)
```

STEP 9: Plot Earthquakes on World Map

```
fig, ax = plt.subplots(figsize=(12,8))

world.plot(ax=ax, color='lightgray')
gdf.plot(ax=ax, markersize=1, color='red')

plt.title("Earthquake Locations on World Map")
plt.show()
```

STEP 10: Interactive Map

```
map_world = folium.Map(
    location=[20,0],
    zoom_start=2
)

for i in range(1000): # limit for performance
    folium.CircleMarker(
        location=[
            data.iloc[i]['Latitude'],
            data.iloc[i]['Longitude']
        ],
        radius=2,
        fill=True
    ).add_to(map_world)

map_world
```