**Q1. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a bar chart showing the sales of different products.**

import matplotlib.pyplot as plt

# Sample data for products and their sales

products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']

sales = [250, 300, 150, 400, 350]

# Create a bar chart

plt.bar(products, sales, color=['blue', 'green', 'red', 'orange', 'purple'])

# Adding labels and title

plt.xlabel('Products', fontsize=12)

plt.ylabel('Sales (in units)', fontsize=12)

plt.title('Sales of Different Products', fontsize=14)

# Adding a legend

plt.legend(['Sales'], loc='upper right')

plt.tight\_layout()

plt.show()

**Q2. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a pie chart showing the sales of different products.**

import matplotlib.pyplot as plt

# Sample data for products and their sales

products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']

sales = [250, 300, 150, 400, 350]

# Create a pie chart

plt.pie(sales, labels=products, autopct='%1.1f%%', colors=['blue', 'green', 'red', 'orange', 'purple'], startangle=140)

# Adding title

plt.title('Sales Distribution of Different Products', fontsize=14)

# Adding a legend

plt.legend(products, title="Products", loc='upper right')

# Display the plot

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

**Q3. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a line chart on weather dataset.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset from the CSV file

df = pd.read\_csv('weather\_data.csv')

# Create a line chart for both temperature and humidity

plt.plot(df['Day'], df['Temperature'], label='Temperature (°C)', color='tab:red', marker='o', linestyle='-', linewidth=2, markersize=8)

plt.plot(df['Day'], df['Humidity'], label='Humidity (%)', color='tab:blue', marker='s', linestyle='--', linewidth=2, markersize=8)

# Adding labels and title

plt.xlabel('Days of the Week', fontsize=12)

plt.ylabel('Values', fontsize=12)

plt.title('Weekly Weather Report: Temperature and Humidity', fontsize=14)

# Adding a legend

plt.legend(title="Weather Parameters", loc='upper left')

# Adding gridlines for better readability

plt.grid(True)

# Display the plot

plt.tight\_layout()

plt.show()

**Q4. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a line chart on Employee dataset.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset from the CSV file

df = pd.read\_csv('employee\_data.csv')

# Create a line chart for each employee's data

plt.plot(df['Month'], df['Employee\_A'], label='Employee A', color='tab:blue', marker='o', linestyle='-', linewidth=2, markersize=8)

plt.plot(df['Month'], df['Employee\_B'], label='Employee B', color='tab:green', marker='s', linestyle='--', linewidth=2, markersize=8)

plt.plot(df['Month'], df['Employee\_C'], label='Employee C', color='tab:red', marker='^', linestyle='-.', linewidth=2, markersize=8)

# Adding labels and title

plt.xlabel('Month', fontsize=12)

plt.ylabel('Hours Worked', fontsize=12)

plt.title('Employee Work Hours Over the Year', fontsize=14)

# Adding a legend

plt.legend(title="Employees", loc='upper left')

# Adding gridlines for better readability

plt.grid(True)

# Display the plot

plt.tight\_layout()

plt.show()

**Q5. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a scatter plot on Employee Dataset.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset from a CSV file

df = pd.read\_csv('employee\_data.csv')

# Create a scatter plot

plt.scatter(df['Work\_Hours'], df['Performance\_Score'], color='tab:orange', marker='o', s=100)

# Adding labels and title

plt.xlabel('Work Hours', fontsize=12)

plt.ylabel('Performance Score', fontsize=12)

plt.title('Employee Work Hours vs Performance Scores', fontsize=14)

# Adding employee labels to each point

for i, employee in enumerate(df['Employee']):

plt.text(df['Work\_Hours'][i], df['Performance\_Score'][i], employee, fontsize=9, ha='right')

# Display the plot

plt.tight\_layout()

plt.show()

**Q6. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create histograms on weather dataset.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset from a CSV file

df = pd.read\_csv('weather\_data.csv')

# Create a figure with subplots for two histograms

fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plotting the temperature histogram

ax[0].hist(df['Temperature'], bins=7, color='tab:blue', edgecolor='black', alpha=0.7)

ax[0].set\_title('Temperature Distribution', fontsize=14)

ax[0].set\_xlabel('Temperature (°C)', fontsize=12)

ax[0].set\_ylabel('Frequency', fontsize=12)

# Plotting the humidity histogram

ax[1].hist(df['Humidity'], bins=7, color='tab:green', edgecolor='black', alpha=0.7)

ax[1].set\_title('Humidity Distribution', fontsize=14)

ax[1].set\_xlabel('Humidity (%)', fontsize=12)

ax[1].set\_ylabel('Frequency', fontsize=12)

# Adding a main title to the figure

plt.suptitle('Weather Data Analysis', fontsize=16)

# Display the plot

plt.tight\_layout()

plt.subplots\_adjust(top=0.85) # Adjusting to make room for the suptitle

plt.show()

**Q7. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a heat map on weather dataset.**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset from the CSV file

df = pd.read\_csv('weather\_data.csv')

# Reshape the data for heatmap: use 'Day' as index and 'Temperature' and 'Humidity' as columns

data = df[['Temperature', 'Humidity']].T # Transpose the DataFrame

# Create a heatmap using Seaborn

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

sns.heatmap(data, annot=True, cmap='YlGnBu', cbar=True, linewidths=0.5)

# Adding labels and title

plt.xlabel('Days of the Week', fontsize=12)

plt.ylabel('Weather Parameters', fontsize=12)

plt.title('Heatmap of Temperature and Humidity Over the Week', fontsize=14)

# Display the plot

plt.tight\_layout()

plt.show()

**Q8. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a Box Plot on weather dataset.**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset from a CSV file

df = pd.read\_csv('weather\_data.csv')

# Create a box plot for Temperature and Humidity

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

# Plot box plots for Temperature and Humidity

sns.boxplot(data=df[['Temperature', 'Humidity']], palette='Set2')

# Adding labels and title

plt.xlabel('Weather Parameters', fontsize=12)

plt.ylabel('Values', fontsize=12)

plt.title('Box Plot of Temperature and Humidity Over the Week', fontsize=14)

# Adding a grid for better readability

plt.grid(True)

# Display the plot

plt.tight\_layout()

plt.show()

**Q9. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a Box Plot on Employee dataset.**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset from the CSV file

df = pd.read\_csv('employee\_data.csv')

# Create a box plot for Work Hours and Performance Score

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

# Plot box plots for Work Hours and Performance Score

sns.boxplot(data=df[['Work\_Hours', 'Performance\_Score']], palette='Set1')

# Adding labels and title

plt.xlabel('Employee Parameters', fontsize=12)

plt.ylabel('Values', fontsize=12)

plt.title('Box Plot of Work Hours and Performance Scores of Employees', fontsize=14)

# Adding a grid for better readability

plt.grid(True)

# Display the plot

plt.tight\_layout()

plt.show()

**Q10. Design visually appealing and informative plot with appropriate labels, titles, and Legends to create a pie chart on weather dataset.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset from the CSV file

df = pd.read\_csv('weather\_data.csv')

# Calculate the frequency of each weather condition

weather\_counts = df['Weather\_Condition'].value\_counts()

# Create a pie chart to visualize the weather condition distribution

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

plt.pie(weather\_counts, labels=weather\_counts.index, autopct='%1.1f%%', colors=['#ff9999','#66b3ff','#99ff99'], startangle=90, wedgeprops={'edgecolor': 'black'})

# Adding title

plt.title('Distribution of Weather Conditions Over the Week', fontsize=14)

# Display the plot

plt.tight\_layout()

plt.show()