# WEEK 4 LAB PROGRAMS

**'''18. Write a program to visualize the correlation matrix of a dataset with a heatmap in Seaborn? '''**

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

# Load an example dataset (e.g., the Iris dataset)
# You can replace this with your own dataset by using pd.read_csv() or any other appropriate method.
df = sns.load_dataset('iris')

# Compute the correlation matrix
corr_matrix = df.corr()

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 8))  # Set figure size
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f', cbar=True)

# Add title
plt.title('Correlation Matrix Heatmap')

# Show the plot
plt.show()
```

```
''' 19. Write a program to apply data aggregation and summarization using pivot tables
inPandas? '''

import pandas as pd

# Sample data: Creating a DataFrame with sales data
data = {
    'Region': ['North', 'South', 'East', 'West', 'North', 'South', 'East', 'West', 'North', 'South'],
    'Salesperson': ['John', 'Jane', 'John', 'Jane', 'John', 'Jane', 'John', 'Jane', 'John', 'Jane'],
    'Product': ['A', 'A', 'B', 'B', 'C', 'C', 'A', 'B', 'C', 'A'],
    'Sales': [250, 200, 300, 220, 150, 180, 350, 190, 120, 210],
    'Units Sold': [25, 20, 30, 22, 15, 18, 35, 19, 12, 21]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("Original DataFrame:")
print(df)

# Pivot table to summarize the total sales by 'Region' and 'Product'
pivot_sales = pd.pivot_table(df,
                             values='Sales',
                             index='Region',
                             columns='Product',
                             aggfunc='sum',
                             fill_value=0)

print("\nPivot Table - Total Sales by Region and Product:")
print(pivot_sales)

# Pivot table to calculate average sales per unit by 'Salesperson' and 'Product'
pivot_avg_sales_per_unit = pd.pivot_table(df,
                                          values='Sales',
                                          index='Salesperson',
                                          columns='Product',
                                          aggfunc=lambda x: x.sum() / df.loc[x.index, 'Units Sold'].sum(),
                                          fill_value=0)

print("\nPivot Table - Average Sales per Unit by Salesperson and Product:")
print(pivot_avg_sales_per_unit)
```

**''' 20. Write a program to sort and filter data based on certain conditions using Pandas?'''**

```
import pandas as pd

# Sample data: Creating a DataFrame with sales data
data = {
    'Region': ['North', 'South', 'East', 'West', 'North', 'South', 'East', 'West', 'North', 'South'],
    'Salesperson': ['John', 'Jane', 'John', 'Jane', 'John', 'Jane', 'John', 'Jane', 'John', 'Jane'],
    'Product': ['A', 'A', 'B', 'B', 'C', 'C', 'A', 'B', 'C', 'A'],
    'Sales': [250, 200, 300, 220, 150, 180, 350, 190, 120, 210],
    'Units Sold': [25, 20, 30, 22, 15, 18, 35, 19, 12, 21]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Filter the data: Get rows where Sales > 200
filtered_data = df[df['Sales'] > 200]
print("\nFiltered Data (Sales > 200):")
print(filtered_data)

# Sort the data: Sort by 'Sales' in descending order
sorted_data = df.sort_values(by='Sales', ascending=False)
print("\nSorted Data (Sales in descending order):")
print(sorted_data)

# Filter and sort together: Get rows where Sales > 200 and sort by 'Units Sold' in ascending order
filtered_sorted_data = df[df['Sales'] > 200].sort_values(by='Units Sold', ascending=True)
print("\nFiltered and Sorted Data (Sales > 200, sorted by Units Sold in ascending order):")
print(filtered_sorted_data)

# Additional filter: Get rows where Region is 'North' or 'South' and Sales > 150
filtered_region_sales = df[(df['Region'].isin(['North', 'South'])) & (df['Sales'] > 150)]
print("\nFiltered Data (Region is North or South, and Sales > 150):")
print(filtered_region_sales)
```

**'''21. Write a program to plot multiple lines or scatter plots on the same axis to compare different datasets in Matplotlib? '''**

```
import matplotlib.pyplot as plt
import numpy as np

# Sample data for line plots
x = np.linspace(0, 10, 100)
y1 = np.sin(x)  # First dataset (Sine function)
y2 = np.cos(x)  # Second dataset (Cosine function)

# Sample data for scatter plots
y3 = np.random.rand(10) * 10  # Random data for scatter
y4 = np.random.rand(10) * 10  # Another set of random data
x_scatter = np.linspace(0, 10, 10)  # x-values for scatter

# Create a figure and axis object
plt.figure(figsize=(8, 6))

# Plot multiple lines on the same axis
plt.plot(x, y1, label='Sine', color='blue', linestyle='-', linewidth=2)
plt.plot(x, y2, label='Cosine', color='red', linestyle='--', linewidth=2)

# Plot multiple scatter plots on the same axis
plt.scatter(x_scatter, y3, color='green', label='Random Scatter 1', s=100, edgecolors='black', marker='o')
plt.scatter(x_scatter, y4, color='purple', label='Random Scatter 2', s=100, edgecolors='black', marker='^')

# Add titles and labels
plt.title('Multiple Lines and Scatter Plots')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Display a legend to label each plot
plt.legend()

# Show grid lines
plt.grid(True)

# Show the plot
plt.show()
```

**Assessment 2: Write a program to create a 3D scatter plot using Matplotlib. The dataset should contain random points in 3D**
**(**Students only must write program for this)