QAS

UNIT – 1

1.

Line Plot

Question: Write a Python script to plot the population growth of a country over the years using a line plot. The years are from 2010 to 2020, and the population data is given as:

Years: [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

Population (in millions): [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1, 3.2, 3.25, 3.3]

Include appropriate labels for the X and Y axes, title, and grid lines for better readability.

```
import matplotlib.pyplot as plt

years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

population = [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1, 3.2, 3.25, 3.3]

plt.plot(years, population, marker='o', linestyle='-', color='b')

plt.xlabel('Years')

plt.ylabel('Population (in millions)')

plt.title('Population Growth Over the Years (2010-2020)')

plt.grid(True)

plt.show()
```

2. Bar Chart

Question: Write a Python script to create a bar chart showing the sales of different products in a store. The products and their corresponding sales are as follows:

Products: ['A', 'B', 'C', 'D', 'E'] Sales: [100, 150, 90, 200, 130]

Ensure that each bar has a unique color, label the X-axis as 'Products', the Y-axis as 'Sales', and give the chart a title.

```
import matplotlib.pyplot as plt
```

```
products = ['A', 'B', 'C', 'D', 'E']
sales = [100, 150, 90, 200, 130]
plt.bar(products, sales, color=['red', 'blue', 'green', 'purple', 'orange'])
plt.xlabel('Products')
plt.ylabel('Sales')
plt.title('Product Sales in Store')
plt.show()
```

3. Histogram

Question: Write a Python script to plot a histogram of student scores in a test. The scores are as follows:

Scores: [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65, 70, 91, 87, 82]

Set the number of bins to 5 and make sure to label the X-axis as 'Scores' and the Y-axis as 'Number of Students'. Give the plot an appropriate title.

```
import matplotlib.pyplot as plt
scores = [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65, 70, 91, 87, 82]
plt.hist(scores, bins=5, edgecolor='black', color='skyblue')
plt.xlabel('Scores')
plt.ylabel('Number of Students')
plt.title('Distribution of Student Scores')
plt.show()
```

4. Question: Write a Python script to create a histogram that shows the distribution of ages in a group of 40 people. The age data is as follows:

Ages: [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22,

25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 30, 40, 45, 55, 60]

Create a histogram with 5 bins. Label the X-axis as 'Ages' and the Y-axis as 'Number of People'. Add a grid to the plot and give it an appropriate title.

```
import matplotlib.pyplot as plt

ages = [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 30, 40, 45, 55, 60]

plt.hist(ages, bins=5, edgecolor='black', color='lightblue')

plt.xlabel('Ages')

plt.ylabel('Number of People')

plt.title('Age Distribution in a Group of People')

plt.grid(True)

plt.show()
```

5. Scatter Plot

Question: Write a Python script to create a scatter plot for the relationship between the age of employees and their salaries in a company. The data is given as:

Ages: [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]

Salaries (in $1000s): [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]

Include appropriate labels for the X and Y axes, and a title. Use a different color for each point.

```
import matplotlib.pyplot as plt

ages = [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]

salaries = [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]

plt.scatter(ages, salaries, color='green')

plt.xlabel('Age')

plt.ylabel('Salary (in $1000s)')

plt.title('Age vs Salary of Employees')

plt.show()
```

6. Pie Chart

Question: Write a Python script to create a pie chart representing the market share of different companies in a specific industry. The companies and their market shares are given as:

Companies: ['Company A', 'Company B', 'Company C', 'Company D'] Market Shares (in %): [25, 30, 20, 25]

Make sure to label each slice with the percentage, and give the chart a title.

```
import matplotlib.pyplot as plt

companies = ['Company A', 'Company B', 'Company C', 'Company D']

market_shares = [25, 30, 20, 25]

plt.pie(market_shares, labels=companies, autopct='%1.1f%%', startangle=90, colors=['gold', 'lightblue', 'lightgreen', 'coral'])

plt.title('Market Share of Companies')

plt.show()
```

7. Box Plot

Question: Write a Python script to create a box plot for the test scores of students in three different subjects: Math, Science, and English. The scores are as follows:

Math: [88, 92, 75, 85, 90, 95, 89, 91, 76, 85]

Science: [80, 85, 88, 92, 91, 87, 90, 95, 89, 88]

English: [78, 82, 80, 85, 88, 90, 91, 86, 89, 92]

Plot a box plot for each subject, and label the X-axis as 'Subjects' and the Y-axis as 'Scores'. Give the plot a title and display the outliers, if any.

```
import matplotlib.pyplot as plt

math_scores = [88, 92, 75, 85, 90, 95, 89, 91, 76, 85]

science_scores = [80, 85, 88, 92, 91, 87, 90, 95, 89, 88]

english_scores = [78, 82, 80, 85, 88, 90, 91, 86, 89, 92]

data = [math_scores, science_scores, english_scores]
```

```
plt.boxplot(data, patch_artist=True, labels=['Math', 'Science', 'English'])
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Box Plot of Student Scores in Different Subjects')
plt.grid(True)
plt.show()
```

8. Heatmap

Question: Write a Python script to create a heatmap representing the performance of different salespersons over five months. The sales data is given as:

Salespersons: ['John', 'Sara', 'Mike', 'Linda', 'Paul']

Sales over months (in thousands of dollars) for each salesperson:

John: [35, 40, 50, 45, 48]

Sara: [40, 42, 48, 44, 47]

Mike: [38, 41, 45, 47, 46]

Linda: [30, 35, 42, 38, 40]

Paul: [50, 52, 53, 55, 58]

Create a heatmap of this sales data using matplotlib. Label the axes, and include a color bar to indicate the sales intensity.

```
import matplotlib.pyplot as plt
import numpy as np
salespersons = ['John', 'Sara', 'Mike', 'Linda', 'Paul']
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
sales_data = [
[35, 40, 50, 45, 48], # John
[40, 42, 48, 44, 47], # Sara
[38, 41, 45, 47, 46], # Mike
[30, 35, 42, 38, 40], # Linda
```

```
[50, 52, 53, 55, 58], # Paul
]
plt.imshow(sales_data, cmap='coolwarm', interpolation='nearest')
plt.colorbar(label='Sales (in $1000s)')
plt.xticks(ticks=np.arange(len(months)), labels=months)
plt.yticks(ticks=np.arange(len(salespersons)), labels=salespersons)
plt.xlabel('Months')
plt.ylabel('Salespersons')
plt.title('Sales Performance Heatmap')
plt.show()
```

9. Contour Plot

Question: Write a Python script to create a contour plot of a function

$Z=f(X,Y)$, where $Z=X2+Y2$. The X and Y values should range from -5 to 5. Generate a contour plot with 10 contour levels, and label both the axes and the contour lines.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + Y**2 # Z = X^2 + Y^2
plt.contour(X, Y, Z, levels=10, cmap='viridis')
plt.colorbar(label='Z = X^2 + Y^2')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Contour Plot of Z = X^2 + Y^2')
```

```
plt.clabel(plt.contour(X, Y, Z, levels=10, cmap='viridis'), inline=True, fontsize=8)
plt.show()
```

UNIT 2

1. Calculates the mean, median, mode, and standard deviation for the following array using NumPy and SciPy:

arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])

Explore the importance of each statistic and provide the output from your code.

```
import numpy as np
from scipy import stats
arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])
mean = np.mean(arr)
median = np.median(arr)
mode = stats.mode(arr).mode[0]
std_dev = np.std(arr)
print(mean)
print(median)
print(mode)
print(std_dev)
```

2. Create a 2D NumPy array as follows:

arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

Calculate the mean, median, mode, and standard deviation of the array along both axes (rows and columns) using NumPy and SciPy. Provide explanations of the results obtained.

```
import numpy as np
from scipy.stats import mode
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
mean_rows = np.mean(arr_2d, axis=1)
mean_columns = np.mean(arr_2d, axis=0)
median_rows = np.median(arr_2d, axis=1)
median_columns = np.median(arr_2d, axis=0)
mode_rows = mode(arr_2d, axis=1)
mode_columns = mode(arr_2d, axis=0)
std_rows = np.std(arr_2d, axis=1)
std_columns = np.std(arr_2d, axis=0)
print("Mean:")
print("Rows:", mean_rows)
print("Columns:", mean_columns)
print("\nMedian:")
print("Rows:", median_rows)
print("Columns:", median_columns)
print("\nMode:")
print("Rows:", mode_rows.mode)
print("Columns:", mode_columns.mode)
print("\nStandard Deviation:")
print("Rows:", std_rows)
print("Columns:", std_columns)
```

3. A dataset represents the daily temperatures of a city in degrees Celsius over the last 10 days: temperatures = np.array([29.2, 31.5, 30.1, 32.0, 29.8, 33.1, 30.5, 32.5, 31.2, 29.9]) Calculate the mean, median, mode, and standard deviation of the temperatures. Elaborate how each of these statistics could help in understanding the climate pattern of the city.

```
import numpy as np
from scipy.stats import mode
temperatures = np.array([29.2, 31.5, 30.1, 32.0, 29.8, 33.1, 30.5, 32.5, 31.2, 29.9])
mean_temp = np.mean(temperatures)
median_temp = np.median(temperatures)
mode_temp = mode(temperatures)
std_temp = np.std(temperatures)
print("Temperature Statistics:")
print("Mean:", mean_temp)
print("Median:", median_temp)
print("Mode:", mode_temp.mode[0] if mode_temp.count[0] > 0 else "No mode")
print("Standard Deviation:", std_temp)
```

4. Given the following 1D NumPy arrays representing the monthly sales of two different products in a store:

product_A = np.array([120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230]) product_B = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135]) Compute the mean, median, mode, and standard deviation for both product_A and product_B. Compare and contrast the results for both products, explaining any insights that can be gained from the statistical analysis.

```
import numpy as np
from scipy.stats import mode
product_A = np.array([120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230])
product_B = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135])
mean_A = np.mean(product_A)
median_A = np.median(product_A)
mode_A = mode(product_A)
std_A = np.std(product_A)
```

```
mean_B = np.mean(product_B)
median_B = np.median(product_B)
mode_B = mode(product_B)
std_B = np.std(product_B)
print("Product A Statistics:")
print("Mean:", mean_A)
print("Median:", median_A)
print("Mode:", mode_A.mode[0] if mode_A.count[0] > 0 else "No mode")
print("Standard Deviation:", std_A)
print("\nProduct B Statistics:")
print("Mean:", mean_B)
print("Median:", median_B)
print("Mode:", mode_B.mode[0] if mode_B.count[0] > 0 else "No mode")
print("Standard Deviation:", std_B)
```

5. Suppose you have the following 2D array where some data points are missing (np.nan):

data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]]) Calculates the mean, median, mode, and standard deviation for this array, while handling missing values appropriately. Explain how missing data can affect statistical analysis and how it should be addressed.

```
import numpy as np
from scipy.stats import mode
data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])
mean_data = np.nanmean(data, axis=0)
median_data = np.nanmedian(data, axis=0)
data_no_nan = np.nan_to_num(data, nan=np.nanmin(data) - 1)  # Replace NaN with out-of-range value
mode_data = mode(data_no_nan, axis=0)
```

```
mode_result = [x if x != np.nanmin(data) - 1 else np.nan for x in mode_data.mode[0]]

std_data = np.nanstd(data, axis=0)

print("Statistics for data with missing values:")

print("Mean (column-wise):", mean_data)

print("Median (column-wise):", median_data)

print("Mode (column-wise):", mode_result)

print("Standard Deviation (column-wise):", std_data)
```

6. Create a 1D NumPy array with 50 random integers between 10 and 100 using Python code

```
arr = np.random.randint(10, 100, size=50)
```

Using Python code calculate the mean, median, mode, and standard deviation of the array. Additionally, plot a histogram of the array to visualize its distribution. How do the statistical values relate to the shape of the distribution?

```
import numpy as np

from scipy.stats import mode

import matplotlib.pyplot as plt

arr = np.random.randint(10, 100, size=50)

mean_val = np.mean(arr)

median_val = np.median(arr)

mode_val = mode(arr)

std_val = np.std(arr)

print("Array Statistics:")

print("Mean:", mean_val)

print("Median:", median_val)

print("Mode:", mode_val.mode[0] if mode_val.count[0] > 0 else "No mode")

print("Standard Deviation:", std_val)

plt.hist(arr, bins=10, color='blue', alpha=0.7, edgecolor='black')
```

```
plt.title('Histogram of Random Integers')
plt.xlabel('Value Range')
plt.ylabel('Frequency')
plt.axvline(mean_val, color='red', linestyle='dashed', linewidth=1, label=f"Mean: {mean_val:.2f}")
plt.axvline(median_val, color='green', linestyle='dashed', linewidth=1, label=f"Median: {median_val:.2f}")
plt.legend()
plt.show()
```

7. A NumPy array contains the ages of people in a group:

ages = np.array([18, 22, 25, 26, 35, 40, 45, 46, 50, 55, 60, 85])

Using Python program to calculate the mean, median, mode, and standard deviation of the ages. Identify and discuss any outliers in the dataset and explain how these outliers affect the statistics. What methods can be used to deal with outliers?

```
import numpy as np
from scipy.stats import mode
ages = np.array([18, 22, 25, 26, 35, 40, 45, 46, 50, 55, 60, 85])
mean_ages = np.mean(ages)
median_ages = np.median(ages)
mode_ages = mode(ages)
std_ages = np.std(ages)
print("Mean:", mean_ages)
print("Median:", median_ages)
print("Mode:", mode_ages.mode[0] if mode_ages.count[0] > 0 else "No mode")
print("Standard Deviation:", std_ages)
```

8. Two groups of students took the same test, and their scores are given by the following 1D arrays:

group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])

group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])

```
import numpy as np
from scipy.stats import mode
group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])
group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])
mean_A = np.mean(group_A)
median_A = np.median(group_A)
mode_A = mode(group_A)
std_A = np.std(group_A)
mean_B = np.mean(group_B)
median_B = np.median(group_B)
mode_B = mode(group_B)
std_B = np.std(group_B)
print("Group A Statistics:")
print("Mean:", mean_A)
print("Median:", median_A)
print("Mode:", mode_A.mode[0] if mode_A.count[0] > 0 else "No mode")
print("Standard Deviation:", std_A)
print("\nGroup B Statistics:")
print("Mean:", mean_B)
print("Median:", median_B)
print("Mode:", mode_B.mode[0] if mode_B.count[0] > 0 else "No mode")
print("Standard Deviation:", std_B)
```

9. Using Python code take any 1D or 2D NumPy array as input and returns the mean, median, mode, and standard deviation for the array. Demonstrate the function with an example array of your choice, and explain the code.

```
import numpy as np
from scipy.stats import mode

def analyze_group_statistics(group):
    mean_value = np.mean(group)
    median_value = np.median(group)
    mode_value = mode(group)
    std_value = np.std(group)
    return {
        "Mean": mean_value,
        "Median": median_value,
        "Mode": mode_value.mode[0] if mode_value.count[0] > 0 else "No mode",
        "Standard Deviation": std_value
    }
example_group = np.array([75, 80, 85, 90, 95, 80, 85, 90, 100, 85])
statistics = analyze_group_statistics(example_group)
print("Statistics for the Example Group:")
for stat, value in statistics.items():
    print(f"{stat}: {value}")
```

10. Consider the following 1D array:

arr = np.array([1, 3, 5, 7, 9, 11, 13, 15])

Using Python program to calculate the mean, median, mode, and standard deviation of the array. Now, apply a linear transformation to the array (arr_transformed = 2 * arr + 5). Recalculate the statistical values for the transformed array. Explain how transformations affect the statistical measures and why.

```
import numpy as np
from scipy import stats
arr = np.array([1, 3, 5, 7, 9, 11, 13, 15])
mean_original = np.mean(arr)
median_original = np.median(arr)
mode_original = stats.mode(arr)[0][0]
std_dev_original = np.std(arr)
print(f"Original Array: {arr}")
print(f"Mean: {mean_original}")
print(f"Median: {median_original}")
print(f"Mode: {mode_original}")
print(f"Standard Deviation: {std_dev_original}")
arr_transformed = 2 * arr + 5
mean_transformed = np.mean(arr_transformed)
median_transformed = np.median(arr_transformed)
mode_transformed = stats.mode(arr_transformed)[0][0]
std_dev_transformed = np.std(arr_transformed)
print("\nTransformed Array:", arr_transformed)
print(f"Mean: {mean_transformed}")
print(f"Median: {median_transformed}")
print(f"Mode: {mode_transformed}")
print(f"Standard Deviation: {std_dev_transformed}")
```

Pandas Question

1. import pandas as pd

data = pd.Series([10, 12, 14, 13, 15, 16, 14, 18, 19, 20])

Write a Python program that calculates the mean, median, mode, standard deviation, and variance for this Series. Explain the importance of each statistical measure and provide the output.

```
import pandas as pd
data = pd.Series([10, 12, 14, 13, 15, 16, 14, 18, 19, 20])
mean = data.mean()
median = data.median()
mode = data.mode()[0]
std_dev = data.std()
variance = data.var()
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
print(f"Standard Deviation: {std_dev}")
print(f"Variance: {variance}")
```

2. Basic Statistical Measures on a pandas DataFrame:

Given the following pandas DataFrame representing the ages and heights of 5 individuals:

import pandas as pd

data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170]}

df = pd.DataFrame(data)

Write a Python program to calculate the mean, median, mode, and standard deviation for each column in the DataFrame. Provide a brief explanation of the results obtained.

```
import pandas as pd
data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170]}
df = pd.DataFrame(data)
mean = df.mean()
median = df.median()
mode = df.mode().iloc[0]
std_dev = df.std()
print("Mean:\n", mean)
print("Median:\n", median)
print("Mode:\n", mode)
print("Standard Deviation:\n", std_dev)
```

3. Given the following DataFrame representing the sales data for different products:

```
import pandas as pd
data = {'Product': ['A', 'B', 'A', 'B', 'A', 'B'], 'Sales': [100, 200, 150, 250, 130, 300]}
df = pd.DataFrame(data)
```

Write a Python program to group the data by the Product column and calculate the mean, median, and standard deviation of the Sales for each product. Explain how grouping the data can provide better insights into the sales performance of each product.

```
import pandas as pd
data = {'Product': ['A', 'B', 'A', 'B', 'A', 'B'], 'Sales': [100, 200, 150, 250, 130, 300]}
df = pd.DataFrame(data)
grouped_stats = df.groupby('Product')['Sales'].agg(['mean', 'median', 'std'])
print(grouped_stats)
```

4. Statistical Summary Using describe() in pandas:

Create a pandas DataFrame as follows:

import pandas as pd

data = {'Age': [23, 25, 31, 22, 35],

'Height': [167, 175, 180, 165, 170],

'Weight': [65, 70, 75, 68, 72]}

df = pd.DataFrame(data)

Write a Python program to use the describe() function in pandas to generate a statistical summary of the DataFrame. Explain the meaning of the statistics provided by describe() (e.g., mean, standard deviation, min, max, etc.).

```
import pandas as pd
data = {'Age': [23, 25, 31, 22, 35],
        'Height': [167, 175, 180, 165, 170],
        'Weight': [65, 70, 75, 68, 72]}
df = pd.DataFrame(data)
summary = df.describe()
print(summary)
```

5. Handling Missing Data in pandas DataFrame:

Consider the following pandas DataFrame with some missing data (NaN values):

import pandas as pd

data = {'Age': [23, 25, None, 22, 35], 'Height': [167, 175, 180, None, 170],

'Weight': [65, 70, None, 68, 72]}

df = pd.DataFrame(data)

Write a Python program to calculate the mean, median, and standard deviation of each column, while appropriately handling the missing data (NaN). Explain how missing data can affect statistical analysis and what strategies can be used to handle it.

```
import pandas as pd
data = {'Age': [23, 25, None, 22, 35], 'Height': [167, 175, 180, None, 170],
        'Weight': [65, 70, None, 68, 72]}
df = pd.DataFrame(data)
mean = df.mean()
median = df.median()
std_dev = df.std()
print("Mean:\n", mean)
print("Median:\n", median)
print("Standard Deviation:\n", std_dev)
```

6. Comparing Statistics of Different DataFrames:

Create two DataFrames representing the monthly sales of two different regions:

df1 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [200, 250, 300, 400,

500]})

df2 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'],'Sales': [180, 220, 320, 410, 480]})

Write a Python program to calculate the mean, median, and standard deviation of the Sales in both regions. Compare the sales performance across the two regions using these statistics.

```
import pandas as pd
df1 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [200, 250, 300, 400, 500]})
df2 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [180, 220, 320, 410, 480]})
mean1 = df1['Sales'].mean()
median1 = df1['Sales'].median()
std_dev1 = df1['Sales'].std()
```

```
print("Region 1 - Mean:", mean1)
print("Region 1 - Median:", median1)
print("Region 1 - Standard Deviation:", std_dev1)
mean2 = df2['Sales'].mean()
median2 = df2['Sales'].median()
std_dev2 = df2['Sales'].std()
print("Region 2 - Mean:", mean2)
print("Region 2 - Median:", median2)
print("Region 2 - Standard Deviation:", std_dev2)
```

7. Column-wise and Row-wise Statistical Calculations:

Consider the following DataFrame representing the marks scored by 3 students in 4 subjects:

```
import pandas as pd
data = {'Math': [85, 92, 78], 'English': [88, 79, 85],'Science': [90, 87, 92],'History': [80, 70, 88]}
df = pd.DataFrame(data, index=['Student1', 'Student2', 'Student3'])
```

```
import pandas as pd
data = {'Math': [85, 92, 78], 'English': [88, 79, 85], 'Science': [90, 87, 92], 'History': [80, 70, 88]}
df = pd.DataFrame(data, index=['Student1', 'Student2', 'Student3'])
row_mean = df.mean(axis=1)
row_median = df.median(axis=1)
row_std_dev = df.std(axis=1)
print("Row-wise Mean:")
print(row_mean)
print("Row-wise Median:")
```

```
print(row_median)
print("Row-wise Standard Deviation:")
print(row_std_dev)
col_mean = df.mean(axis=0)
col_median = df.median(axis=0)
col_std_dev = df.std(axis=0)
print("Column-wise Mean:")
print(col_mean)
print("Column-wise Median:")
print(col_median)
print("Column-wise Standard Deviation:")
print(col_std_dev)
```

8.Write a Python program to calculate the mean, median, and standard deviation for each student (row-wise) and for each subject (column-wise). Explain the significance of calculating statistics both row-wise and column-wise in this context.

```
import pandas as pd
def calculate_statistics(df):
    stats = {}
    for col in df.columns:
        stats[col] = {
            'Mean': df[col].mean(),
            'Median': df[col].median(),
            'Mode': df[col].mode()[0],
            'Standard Deviation': df[col].std()
        }
    return stats
data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170], 'Weight': [65, 70, 75, 68, 72]}
```

```
df = pd.DataFrame(data)

stats = calculate_statistics(df)

for col, stat in stats.items():

    print(f"\n{col} - Mean: {stat['Mean']}, Median: {stat['Median']}, Mode: {stat['Mode']}, Standard Deviation: {stat['Standard Deviation']}")
```

9. Custom Function for Statistical Measures in pandas:

Write a Python function that takes a pandas DataFrame as input and returns the mean, median, mode, and standard deviation for each column. Demonstrate the function using the following DataFrame and explain the results:

data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170],'Weight': [65, 70, 75, 68, 72]}

df = pd.DataFrame(data)

```
import pandas as pd

def calculate_statistics(df):

    stats = {}

    for col in df.columns:

        stats[col] = {

            'Mean': df[col].mean(),

            'Median': df[col].median(),

            'Mode': df[col].mode()[0],

            'Standard Deviation': df[col].std()

        }

    return stats

data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170], 'Weight': [65, 70, 75, 68, 72]}

df = pd.DataFrame(data)

stats = calculate_statistics(df)

for col, stat in stats.items():
```

```
    print(f"\n{col} - Mean: {stat['Mean']}, Median: {stat['Median']}, Mode: {stat['Mode']}, Standard Deviation: {stat['Standard Deviation']}")
```

```
print(trials)
print(mean_outcome)
print(variance_outcome)
```

12. Simulating Bernoulli Trials with NumPy:

Write a Python program that simulates 1000 Bernoulli trials with a success probability of 0.4 using NumPy. Calculate the proportion of successes and compare it with the theoretical probability. Plot a bar chart to visualize the number of successes and failures. Explain the relationship between the simulated results and the theoretical probability of success.

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

p_success = 0.4   # Probability of success
n_trials = 1000   # Number of trials
trials = np.random.binomial(1, p_success, n_trials)
proportion_success = np.mean(trials)
# Theoretical probability of success
theoretical_prob = p_success
# Count the number of successes and failures
n_success = np.sum(trials)
n_failure = n_trials - n_success
# Display the results
print(f"Number of successes: {n_success}")
print(f"Number of failures: {n_failure}")
print(f"Proportion of successes (simulated): {proportion_success}")
print(f"Theoretical probability of success: {theoretical_prob}")
# Plot the bar chart
labels = ['Success', 'Failure']
counts = [n_success, n_failure]
plt.bar(labels, counts, color=['green', 'red'])
plt.ylabel('Count')
plt.title('Bernoulli Trials: Successes vs Failures')
plt.show()
```

13. Bernoulli Distribution: Mean and Variance:

Write a Python program to generate 500 Bernoulli trials with p=0.5 and calculate the sample mean and sample variance of the trials. Compare the results with the theoretical mean and variance, explaining any discrepancies.

5. Binomial Distribution as a Series of Bernoulli Trials:

Explain how the binomial distribution is related to a series of Bernoulli trials. Write a Python program to generate 1000 random samples from a binomial distribution with n=10, and p=0.3. Calculate the mean, variance, and standard deviation of the samples, and compare these with the theoretical values of the binomial distribution.

```
import numpy as np,  import matplotlib.pyplot as plt
# Parameters of the binomial distribution
n = 10      # Number of trials
p = 0.3     # Probability of success
samples = 1000  # Number of samples

# Generate 1000 random samples from the binomial distribution
binom_samples = np.random.binomial(n, p, samples)
```

```
# Calculate sample mean, variance, and standard deviation
sample_mean = np.mean(binom_samples)
sample_variance = np.var(binom_samples)
sample_std_dev = np.std(binom_samples)
# Theoretical mean, variance, and standard deviation
theoretical_mean = n * p
theoretical_variance = n * p * (1 - p)
theoretical_std_dev = np.sqrt(theoretical_variance)

# Display the results
print(f"Sample Mean: {sample_mean}")
print(f"Sample Variance: {sample_variance}")
print(f"Sample Standard Deviation: {sample_std_dev}")

print(f"Theoretical Mean: {theoretical_mean}")
print(f"Theoretical Variance: {theoretical_variance}")
print(f"Theoretical Standard Deviation: {theoretical_std_dev}")

# Plot a histogram of the binomial samples
plt.hist(binom_samples, bins=np.arange(0, n+2)-0.5, density=True, color='blue',
alpha=0.7, edgecolor='black')
plt.title('Histogram of Binomial Distribution Samples (n=10, p=0.3)')
plt.xlabel('Number of successes')
plt.ylabel('Probability')
plt.xticks(range(0, n+1))
plt.grid(True)
plt.show()
```

14. Understanding the Normal Distribution:
Define the normal distribution and explain the significance of its parameters μ (mean) and σ(standard deviation). Write a Python program to generate 1000 random samples from a normal distribution with μ=50 and σ=10. Plot a histogram of the samples and superimpose the probability density function (PDF). Explain the shape of the histogram and its relation to the normal PDF.

```
import numpy as np, import matplotlib.pyplot as plt, import scipy.stats as stats
# Parameters of the normal distribution
mu = 50    # Mean
sigma = 10  # Standard deviation
samples = 1000  # Number of random samples
# Generate 1000 random samples from the normal distribution
normal_samples = np.random.normal(mu, sigma, samples)
# Calculate the range of x values for the PDF plot
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
# Compute the PDF for the normal distribution with mu=50 and sigma=10
pdf = stats.norm.pdf(x, mu, sigma)
# Plot the histogram of the random samples
plt.hist(normal_samples, bins=30, density=True, alpha=0.6, color='b',
edgecolor='black')
# Superimpose the PDF
plt.plot(x, pdf, 'r', linewidth=2)
# Add labels and title
plt.title('Histogram of Samples and Normal Distribution PDF ($\mu=50$,
$\sigma=10$)')
plt.xlabel('Value')
```

```
plt.ylabel('Density')
# Show the plot
plt.show()
```

7. Properties of the Normal Distribution:
The 68-95-99.7 rule (Empirical rule) states that for a normal distribution:
68% of the data falls within 1 standard deviation from the mean,
95% within 2 standard deviations, and
99.7% within 3 standard deviations.

Write a Python program to generate 10,000 samples from a normal distribution with mean μ=100, μ=100 and standard deviation σ=15. Verify the empirical rule by calculating the proportion of data points within 1, 2, and 3 standard deviations of the mean.

Questions and Answers: Unit 3

1.Provide an example to illustrate the basic strurctuire of an HTML document. In your response, ensure you include details about the essential tags used to structure an HTML document.

2.Describe the purpose and usage of the HTML tags: <head><body><title><meta><link>
Include examples to demonstrate how these tags are used in an HTML document.

3.How are hyperlinks created in HTML using the <a> tag? Explain the attributes associated with the <a> tag and provide examples to show how links can be opened in new tabs, link to an email, or link to external websites.

4.Enumerate the difference between ordered and unordered lists in HTML. Describe the purpose of the <ol> and <ul>, <li> tags and provide examples showing how to create both types in a web document.

5.Consider the following simple html structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
      <title>Simple Page</title>
  </head>
  <body>
    <h1>Welcome to My Webpage</h1>
    <p>This is a simple paragraph of text.</p>
  </body>
</html>
```

Modify the HTML to achieve the following:
Center the heading <h1> both horizontally on the page.
Change the font size of the heading to 36px and the paragraph text to 18px.
Set the heading's text color to blue and the paragraph's text color to green.

6.Consider the following HTML structure for an ordered list:

```
<!DOCTYPE html>
<html lang="en">
```