**ASSIGNMENT-1**

**Q1.** Using NumPy, create a 3D array of shape (4, 4, 4) filled with random numbers between 1 and 100. Find the z score normalisation of each element of the array.

Sure! Here’s how you can create a 3D array of shape (4, 4, 4) filled with random numbers between 1 and 100 using NumPy, and then find the z-score normalization of each element:

**Python**

import numpy as np

# Create a 3D array of shape (4, 4, 4) with random numbers between 1 and 100

array = np.random.randint(1, 101, size=(4, 4, 4))

# Calculate the mean and standard deviation of the array

mean = np.mean(array)

std = np.std(array)

# Calculate the z-score normalization

z\_score\_normalized\_array = (array - mean) / std

print("Original Array:\n", array)

print("\nZ-Score Normalized Array:\n", z\_score\_normalized\_array)

**Q2.** Using Pandas, filter the rows in the Iris dataset where the petal length is greater than 3.0, and display the filtered data

import pandas as pd

import seaborn as sns

# Load the Iris dataset using seaborn

iris = sns.load\_dataset('iris')

# Filter the rows where 'petal\_length' is greater than 3.0

filtered\_data = iris[iris['petal\_length'] > 3.0]

# Display the filtered data

print(filtered\_data)

**Q3.** Using Matplotlib, plot a line graph showing the trend of a numerical dataset. Customize the plot with title, axis labels, and a legend.

import matplotlib.pyplot as plt

import numpy as np

# Generate a numerical dataset (e.g., a simple sine wave trend)

x = np.linspace(0, 10, 100) # 100 data points from 0 to 10

y = np.sin(x) # y values are the sine of x

# Create the line plot

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

plt.plot(x, y, label='Sine Wave', color='b', linestyle='-', linewidth=2)

# Add a title and axis labels

plt.title('Trend of Sine Wave

**Q4.**Create a Pandas DataFrame with random values with at least 3 columns and perform the following operations: (i) Replace NaN values with the mean of the column. (ii) Sort the DataFrame based on any two columns.

import numpy as np

# Define two 2D matrices (e.g., 3x3 matrices)

A = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

B = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

# Calculate the dot product of A and B

resultant\_matrix = np.dot(A, B)

# Print the resulting matrix

print("Dot Product of A and B:")

print(resultant\_matrix)

# Calculate the determinant of the resulting matrix

determinant = np.linalg.det(resultant\_matrix)

# Print the determinant

print("\nDeterminant of the resulting matrix:", determinant)

**Q5.** Using NumPy, calculate the dot product of two matrices and find the determinant of the resulting matrix.

import numpy as np

# Define two 2D matrices (e.g., 3x3 matrices)

A = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

B = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

# Calculate the dot product of A and B

resultant\_matrix = np.dot(A, B)

# Print the resulting matrix

print("Dot Product of A and B:")

print(resultant\_matrix)

# Calculate the determinant of the resulting matrix

determinant = np.linalg.det(resultant\_matrix)

# Print the determinant

print("\nDeterminant of the resulting matrix:", determinant)

**Q6.** With Pandas, group a DataFrame by a categorical column and calculate the sum of another numerical column for each group.

import pandas as pd

import seaborn as sns

# Load the tips dataset from seaborn

tips = sns.load\_dataset('tips')

# Group by the 'sex' column (categorical) and calculate the sum of the 'tip' column (numerical)

grouped = tips.groupby('sex')['tip'].

**Q7**. Using Matplotlib, create a histogram to show the distribution of a numerical dataset. Customize the number of bins and add grid lines for better readability.

import matplotlib.pyplot as plt

import numpy as np

# Generate a random dataset of 1000 values from a normal distribution

data = np.random.randn(1000)

# Create the histogram

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

plt.hist(data, bins=30, color='skyblue', edgecolor='black', alpha=0.7)

# Add title and labels

plt.title('Histogram of Normally Distributed Data', fontsize=16)

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

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

# Add grid lines for better readability

plt.grid(True, linestyle='--', alpha=0.6)

# Show the plot

plt.show()

**Q8.** Using NumPy, create a 2D array and perform a slicing operation to extract a specific sub-array.

import numpy as np

# Create a 2D array (5x5 matrix)

array\_2d = np.array([[1, 2, 3, 4, 5],

[6, 7, 8, 9, 10],

[11, 12, 13, 14, 15],

[16, 17, 18, 19, 20],

[21, 22, 23, 24, 25]])

# Print the original 2D array

print("Original 2D Array:")

print(array\_2d)

# Perform slicing to extract a sub-array

# For example, extract the sub-array consisting of rows 1 to 3 and columns 2 to 4 (0-based indexing)

sub\_array = array\_2d[1:4, 2:5]

# Print the extracted sub-array

print("\nExtracted Sub-array:")

print(sub\_array)

**Q9.** Using Pandas, perform a merge operation between two DataFrames based on a common column for dataset of your choice.

import pandas as pd

import seaborn as sns

# Load the tips dataset from seaborn

tips = sns.load\_dataset('tips')

# Create a second DataFrame with summary data (for example, average tips by gender)

df2 = tips.groupby('sex')['tip'].mean().reset\_index()

df2.columns = ['sex', 'avg\_tip']

# Now, merge the original `tips` DataFrame with this summary DataFrame on the 'sex' column

merged\_df = pd.merge(tips, df2, on='sex', how='left')

# Display the merged DataFrame

print(merged\_df.head())

**Q10.** Using Matplotlib, create a scatter plot comparing two numerical variables in the dataset of your choice. Customize the plot by changing the color and shape of the data points.

import matplotlib.pyplot as plt

import seaborn as sns

# Load an example dataset

data = sns.load\_dataset('tips')

# Create a scatter plot comparing 'total\_bill' and 'tip'

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

plt.scatter(data['total\_bill'], data['tip'],

color='purple', # Set the color of the points

marker='x', # Set the shape of the points

alpha=0.6) # Set transparency of the points

# Add title and labels

plt.title('Scatter Plot of Total Bill vs Tip', fontsize=16)

plt.xlabel('Total Bill', fontsize=12)

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

# Display the plot

plt.grid(True)

plt.show()