ATHARVA CHAUDHARI
ROLL NO: 42

Experiment 1

Aim: To analyze the student performance dataset using **NumPy, Pandas, Matplotlib, and Seaborn** in order to compute statistical measures, perform data normalization, and visualize the relationship between study-related factors and final student scores.

Theory:

In machine learning and data analysis, understanding the dataset is a crucial step before building any predictive model. This experiment focuses on **numerical analysis and visualization** of student performance data.

NumPy

NumPy is a fundamental Python library used for numerical computations. It provides efficient array operations and mathematical functions. In this experiment, NumPy is used to:

- Convert dataset columns into arrays
- Compute mean, median, and standard deviation
- Perform Min–Max normalization

Matplotlib

Matplotlib is a plotting library used for creating static visualizations. It helps in understanding trends and distributions within the data. In this experiment, Matplotlib is used to:

- Plot a line graph between Hours_Studied and Final_Score
- Plot a histogram of Final_Score

Seaborn

Seaborn is a high-level visualization library built on top of Matplotlib. It provides attractive and informative statistical plots. In this experiment, Seaborn is used to:

- Create scatter plots to analyze relationships
- Generate heatmaps for correlation analysis
- Draw boxplots for categorical performance analysis

Normalization

Min–Max normalization scales data into a fixed range (0 to 1). This helps in improving data consistency and prepares the dataset for future machine learning models.

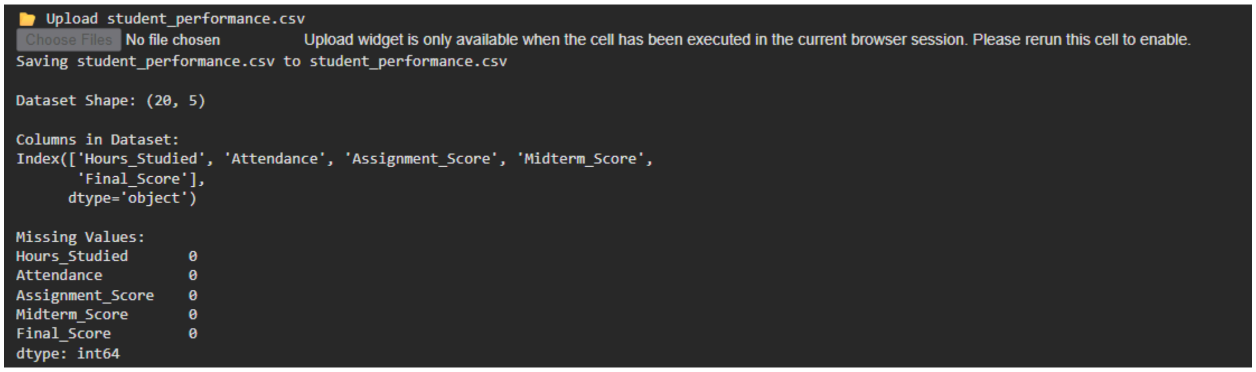
CODE:

```
# IMPORT LIBRARIES

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from google.colab import files

# 1. Load CSV file using Pandas

print("📂 Upload student_performance.csv")

uploaded = files.upload()
```

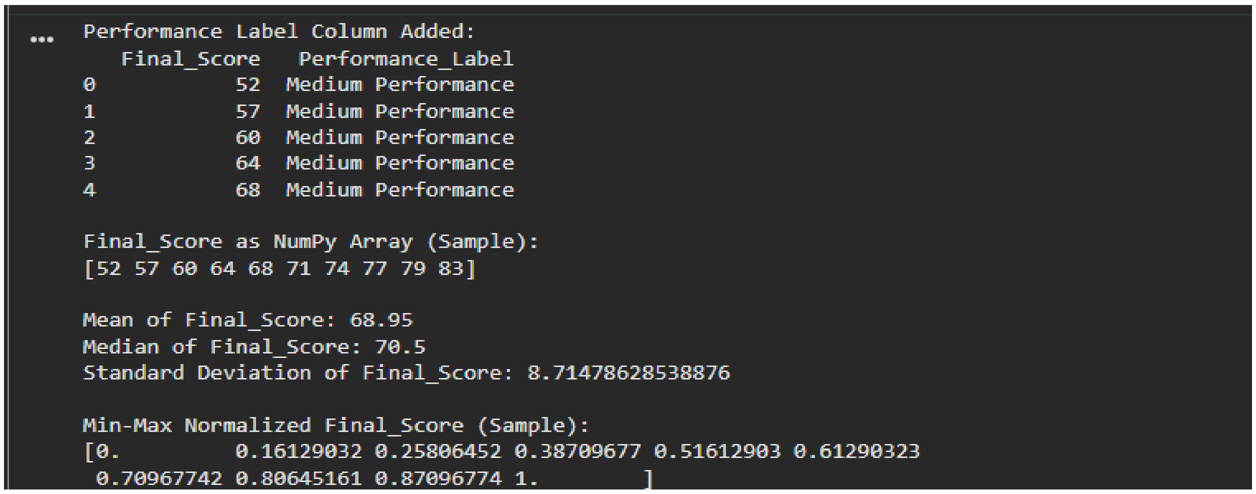
```
df = pd.read_csv(list(uploaded.keys())[0])
# 2. Check shape, columns, and missing values
print("\nDataset Shape:", df.shape)
print("\nColumns in Dataset:")
print(df.columns)
print("\nMissing Values:")
print(df.isnull().sum())
# 3. Create Performance label based on Final_Score
def performance_label(score):
if score < 50:
return "Low Performance"
elif score < 75:
return "Medium Performance"
else:
return "High Performance"
df['Performance_Label'] = df['Final_Score'].apply(performance_label)
print("\nPerformance Label Column Added:")
print(df[['Final_Score', 'Performance_Label']].head())
# EXERCISE 1: NUMPY BASICS
# 1. Load Final_Score as NumPy array
final_score_np = df['Final_Score'].to_numpy()
print("\nFinal_Score as NumPy Array (Sample):")
print(final_score_np[:10])
```

```
# 2. Mean, Median, Standard Deviation

print("\nMean of Final_Score:", np.mean(final_score_np))

print("Median of Final_Score:", np.median(final_score_np))

print("Standard Deviation of Final_Score:", np.std(final_score_np))

# 3. Min-Max Normalization

min_val = final_score_np.min()

max_val = final_score_np.max()

normalized_scores = (final_score_np - min_val) / (max_val - min_val)

print("\nMin-Max Normalized Final_Score (Sample):")

print(normalized_scores[:10])

# EXERCISE 3: MATPLOTLIB VISUALIZATION

# 1. Line plot: Hours_Studied vs Final_Score

plt.figure()

plt.plot(df['Hours_Studied'], df['Final_Score'])

plt.xlabel("Hours Studied")

plt.ylabel("Final Score")

plt.title("Hours Studied vs Final Score (Line Plot)")

plt.show()

# 2. Histogram of Final_Score

plt.figure()

plt.hist(df['Final_Score'], bins=20)

plt.xlabel("Final Score")

plt.ylabel("Frequency")
plt.title("Histogram of Final Score")

plt.show()

# EXERCISE 4: SEABORN VISUALIZATION

# 1. Scatter plot

plt.figure()

sns.scatterplot(x='Hours_Studied', y='Final_Score', data=df)

plt.title("Scatter Plot: Hours Studied vs Final Score")

plt.show()

# 2. Heatmap for correlation analysis

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

sns.heatmap(df[['Hours_Studied', 'Attendance',

'Assignment_Score', 'Midterm_Score',

'Final_Score']].corr(),

annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()

# 3. Boxplot for categorical analysis

plt.figure()

sns.boxplot(x='Performance_Label', y='Final_Score', data=df)

plt.title("Final Score by Performance Category")

plt.show()

print("\n✅ All exercises completed successfully")
```

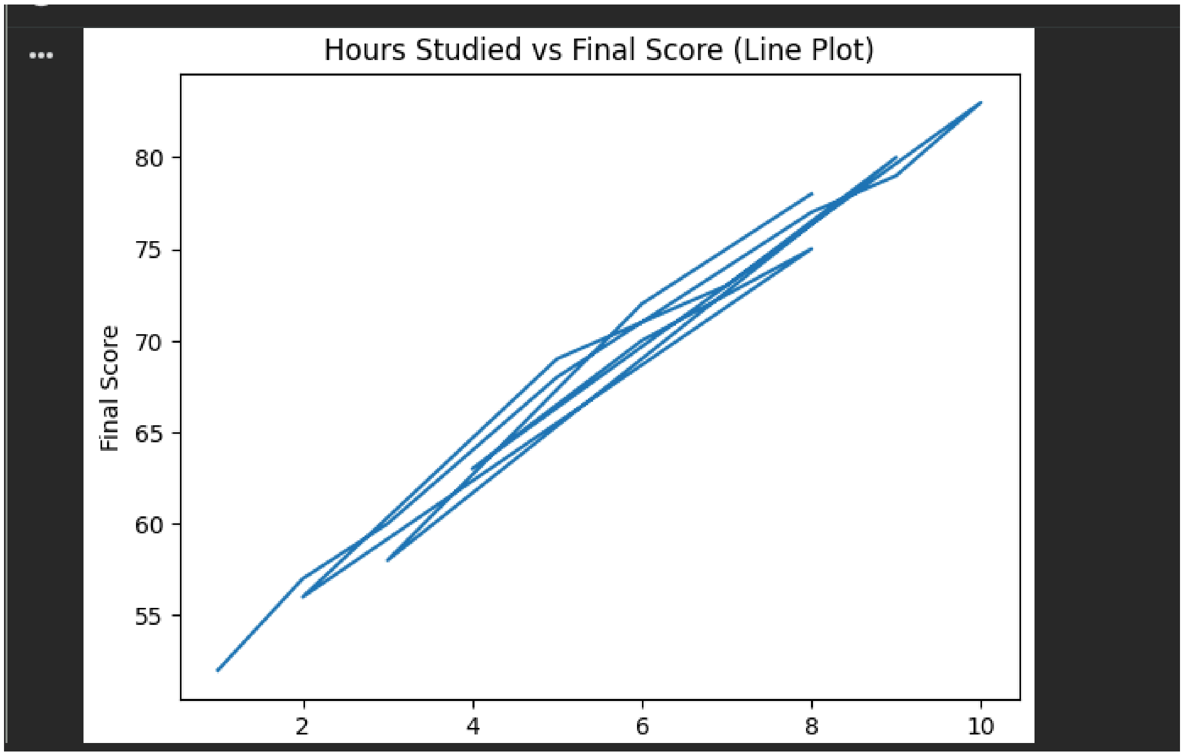
OUTPUT:

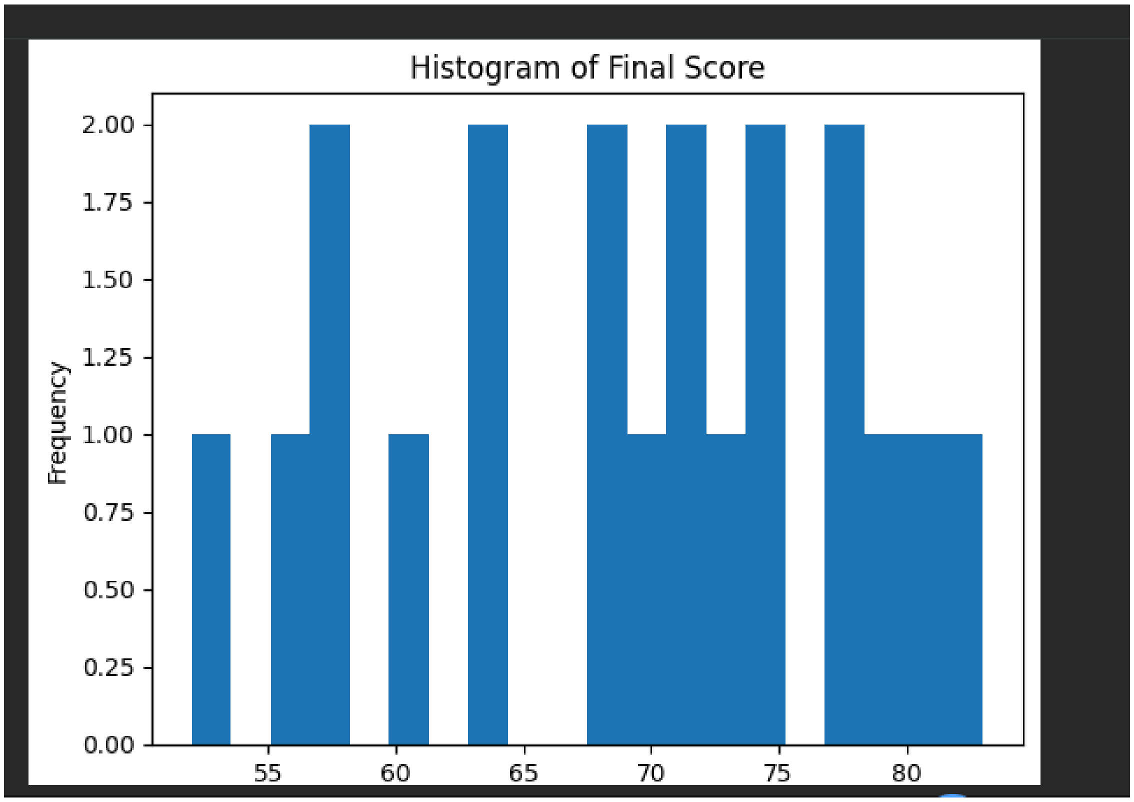


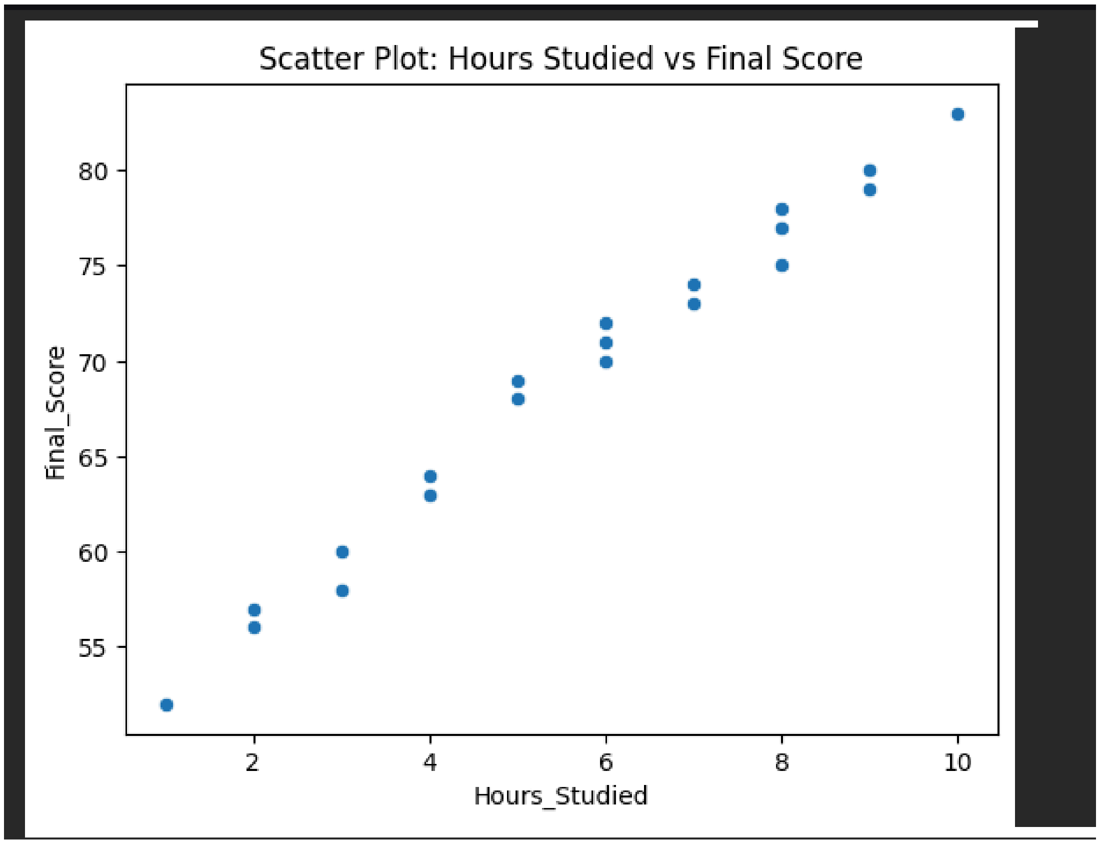
```
Upload student_performance.csv
Choose Files No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving student_performance.csv to student_performance.csv

Dataset Shape: (20, 5)

Columns in Dataset:
Index(['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score',
       'Final_Score'],
      dtype='object')

Missing Values:
Hours_Studied       0
Attendance          0
Assignment_Score    0
Midterm_Score       0
Final_Score         0
dtype: int64
```

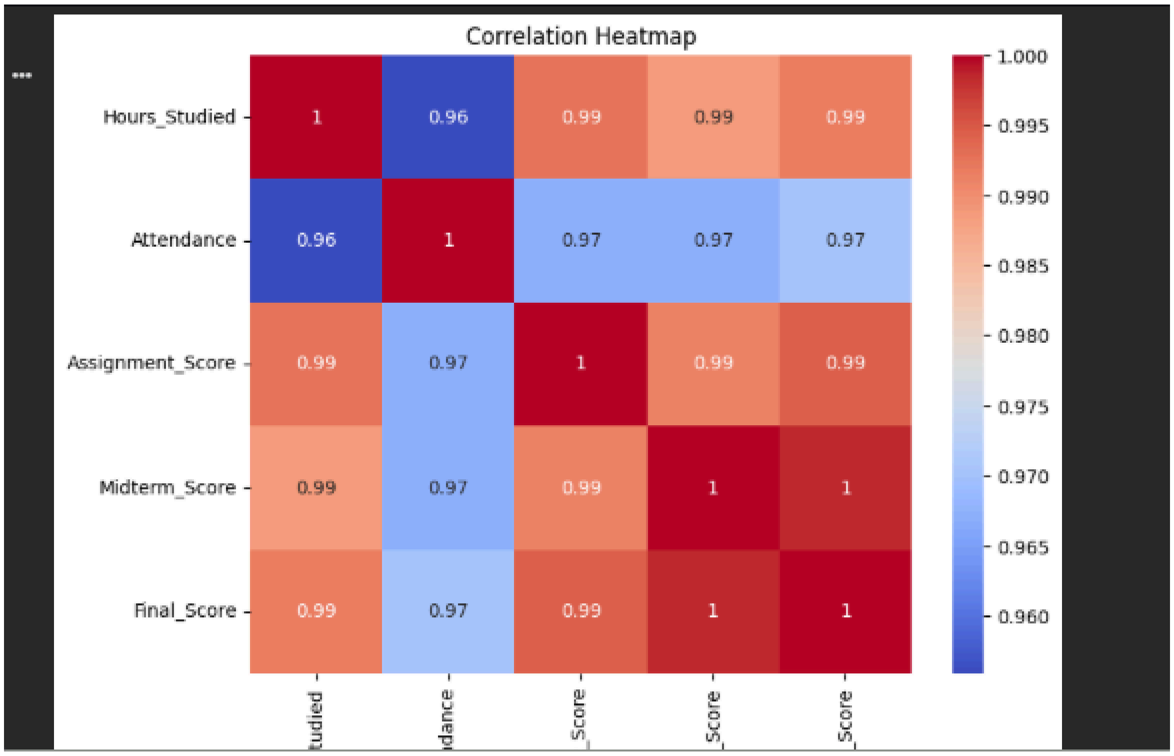


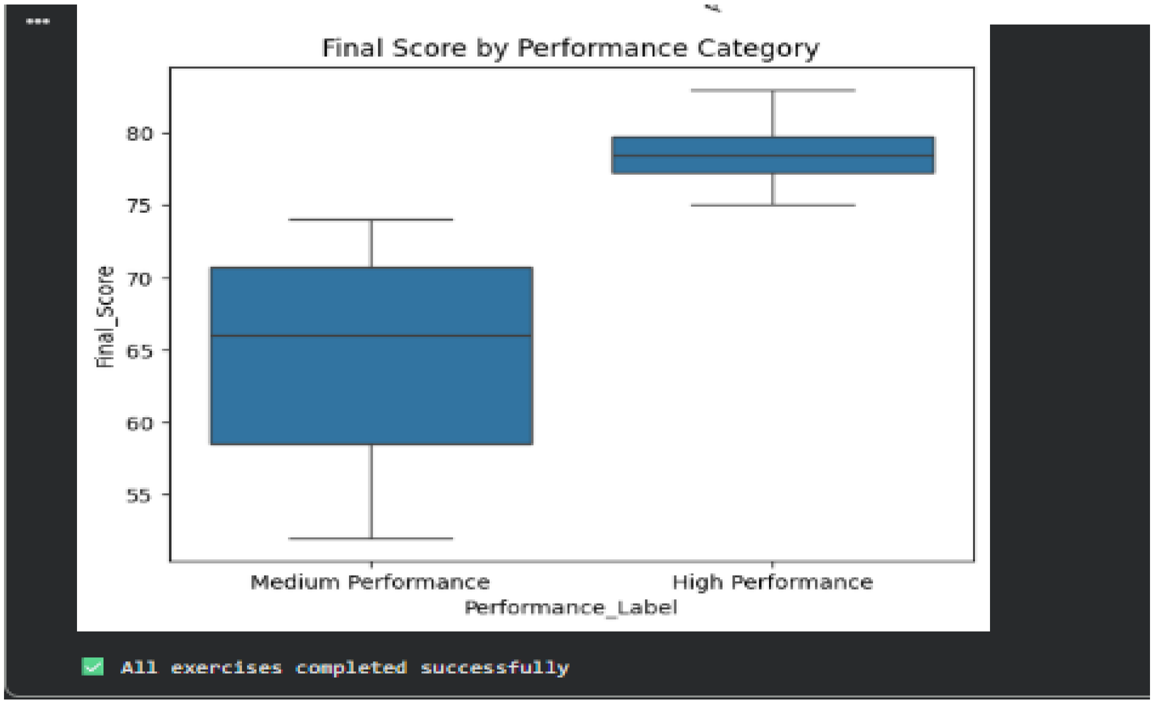
```
Performance Label Column Added:
   Final_Score   Performance_Label
0           52  Medium Performance
1           57  Medium Performance
2           60  Medium Performance
3           64  Medium Performance
4           68  Medium Performance

Final_Score as NumPy Array (Sample):
[52 57 60 64 68 71 74 77 79 83]

Mean of Final_Score: 68.95
Median of Final_Score: 70.5
Standard Deviation of Final_Score: 8.714786285388776

Min-Max Normalized Final_Score (Sample):
[0.         0.16129032 0.25806452 0.38709677 0.51612903 0.61290323
 0.70967742 0.80645161 0.87096774 1.        ]
```




Histogram of Final Score
Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
55
60
65
70
75
80


Scatter Plot: Hours Studied vs Final Score
Final_Score
80
75
70
65
60
55
2
4
6
8
10
Hours_Studied





```
✅ All exercises completed successfully
```

CONCLUSION:

This experiment successfully demonstrated the use of **NumPy, Pandas, Matplotlib, and Seaborn** for analyzing and visualizing student performance data. Statistical measures such as mean, median, and standard deviation provided insights into the overall distribution of final scores, while Min–Max normalization helped in scaling the data for consistent analysis.