

LAB Assignment No. 1

Lab Assignment – Dataset Creation & Analysis Objective

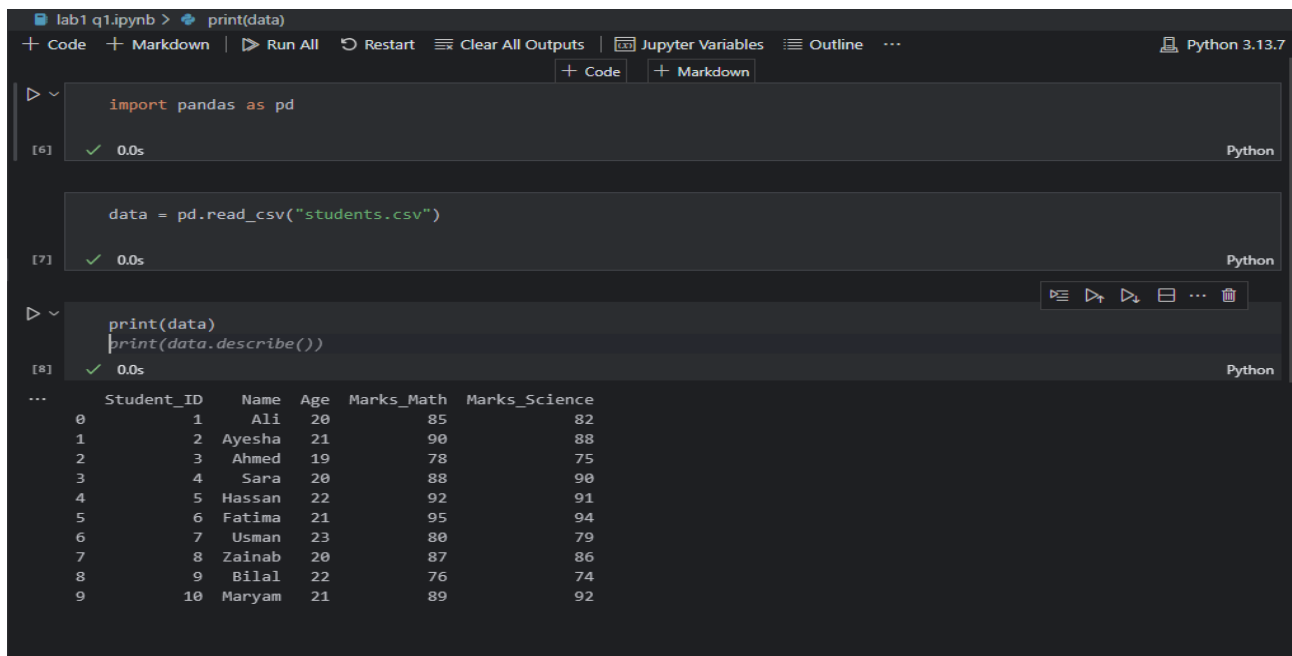
To learn how to create and upload a dataset in Python, perform basic statistical analysis, and visualize data using graphs.

Tasks Q1: Create a Dataset Manually

- Create a dataset of at least **10 students** with the following columns:
 - o Student_ID, o
 - Name, o
 - Age, o
 - Marks_Math, o
 - Marks_Science.
- Store the dataset in a **CSV file** named students.csv.

Q2: Upload Dataset in Python

- Use **Pandas** to load the dataset.



The screenshot shows a Jupyter Notebook interface with three code cells. The first cell imports Pandas as 'pd'. The second cell reads a CSV file named 'students.csv' into a variable 'data'. The third cell prints the data and its description. The output of the third cell is a table with 10 rows of student data.

```
import pandas as pd
```

```
data = pd.read_csv("students.csv")
```

```
print(data)
print(data.describe())
```

	Student_ID	Name	Age	Marks_Math	Marks_Science
0	1	Ali	20	85	82
1	2	Ayesha	21	90	88
2	3	Ahmed	19	78	75
3	4	Sara	20	88	90
4	5	Hassan	22	92	91
5	6	Fatima	21	95	94
6	7	Usman	23	80	79
7	8	Zainab	20	87	86
8	9	Bilal	22	76	74
9	10	Maryam	21	89	92

Q3: Observe Dataset Information

Run the following commands and explain the output:

1. `data.info()` → Dataset structure

```
data.info()
```

[27] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Student_ID  10 non-null    int64
1   Name        10 non-null    object
2   Age         10 non-null    int64
3   Marks_Math  10 non-null    int64
4   Marks_Science 10 non-null    int64
dtypes: int64(4), object(1)
memory usage: 532.0+ bytes
```

2. `data.describe()` → Summary statistics (mean, std, min, max, etc.)

```
data.describe()
```

✓ 0.0s Python

	Student_ID	Age	Marks_Math	Marks_Science
count	10.00000	10.000000	10.000000	10.000000
mean	5.50000	20.900000	86.000000	85.100000
std	3.02765	1.197219	6.218253	7.202623
min	1.00000	19.000000	76.000000	74.000000
25%	3.25000	20.000000	81.250000	79.750000
50%	5.50000	21.000000	87.500000	87.000000
75%	7.75000	21.750000	89.750000	90.750000
max	10.00000	23.000000	95.000000	94.000000

+ Code + Markdown

3. `data['Marks_Math'].mean()` → Mean of Math marks

```
data['Marks_Math'].mean()
```

[29] ✓ 0.0s Python

```
np.float64(86.0)
```

4. `data['Marks_Science'].max()` → Maximum Science marks

```
data['Marks_Science'].max()
```

[30] ✓ 0.0s Python

... np.int64(94)

Q4: Perform Some Data Analysis

- Find how many students have Marks_Math > 50.
- Find the student with the **highest Science marks**.
- Calculate the **correlation** between Marks_Math and Marks_Science.

```
lab1 q1.ipynb > correlation = data['Marks_Math'].corr(data['Marks_Science'])
+ Code + Markdown | ▶ Run All ⏮ Restart ⚙ Clear All Outputs | 📄 Jupyter Variables 📄 Outline ... Python 3.13.7
```

```
count = data[data['Marks_Math'] > 50].shape[0]
print(count)
```

[55] ✓ 0.0s Python

... 10

```
top_student = data.loc[data['Marks_Science'].idxmax()]
print(top_student)
```

[56] ✓ 0.0s Python

... Student_ID 6
Name Fatima
Age 21
Marks_Math 95
Marks_Science 94
Name: 5, dtype: object

```
correlation = data['Marks_Math'].corr(data['Marks_Science'])
print(correlation)
data_sorted = data.sort_values(by='Marks_Math', ascending=False)
```

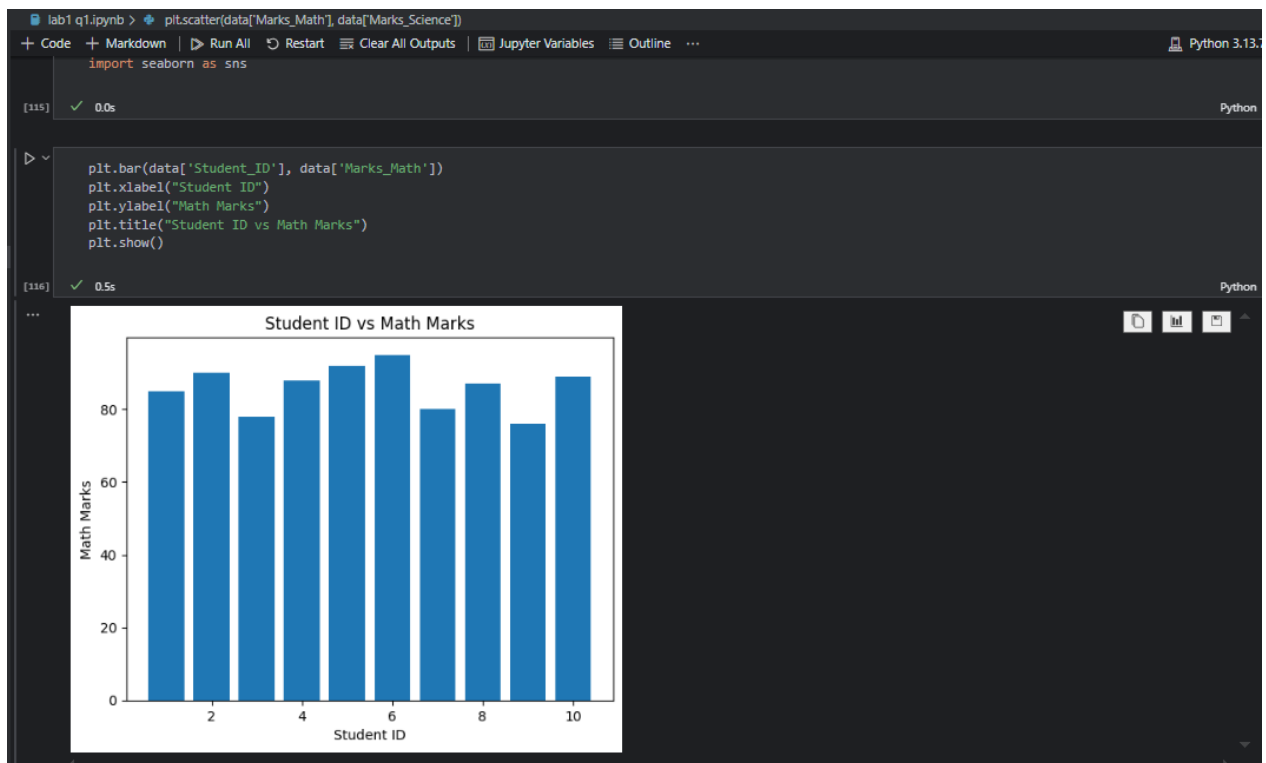
[57] ✓ 0.0s Python

... 0.967526899124535

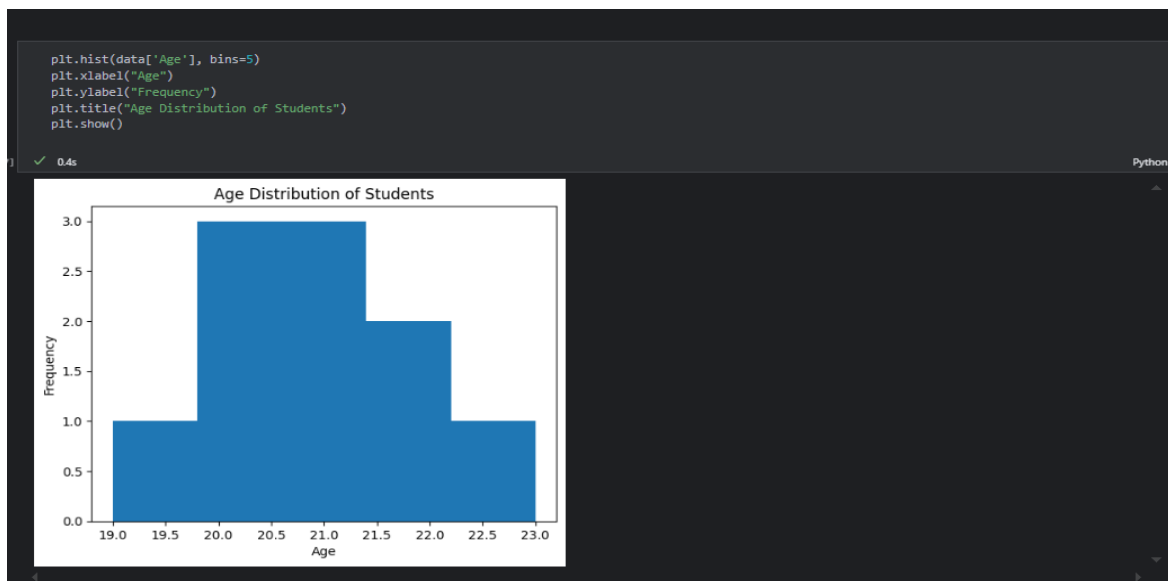
Q5: Data Visualization

Use **Matplotlib/Seaborn** to create graphs:

1. A bar chart of Student_ID vs Marks_Math.



A histogram of Age.



2. A scatter plot of Marks_Math vs Marks_Science.

```
plt.scatter(data['Marks_Math'], data['Marks_Science'])  
plt.xlabel("Math Marks")  
plt.ylabel("Science Marks")  
plt.title("Math vs Science Marks")  
plt.show()
```

