

## LAB Assignment No. 1

### Lab Assignment – Dataset Creation & Analysis Objective

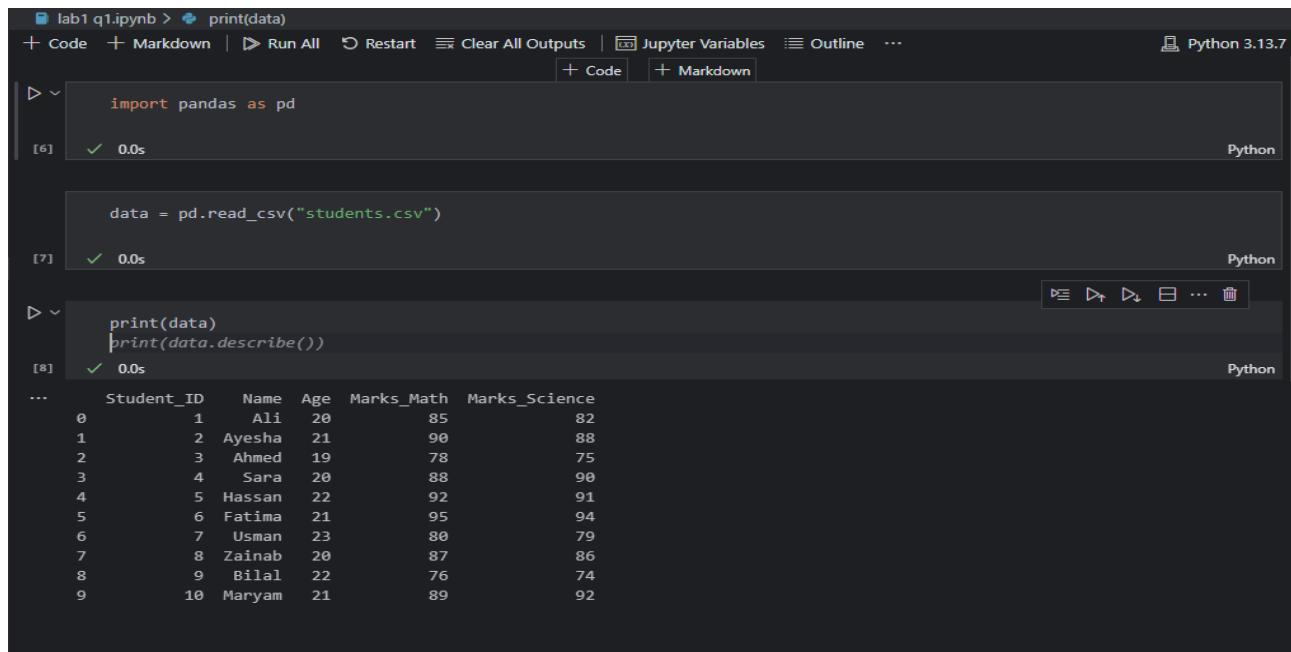
To learn how to create and upload a dataset in Python, perform basic statistical analysis, and visualize data using graphs.

#### 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 and their corresponding outputs:

- Cell 6:** Contains the code `import pandas as pd`. The output shows a green checkmark and "0.0s".
- Cell 7:** Contains the code `data = pd.read_csv("students.csv")`. The output shows a green checkmark and "0.0s".
- Cell 8:** Contains the code `print(data)` and `print(data.describe())`. The output shows a green checkmark and "0.0s". Below the code, a table is displayed:

	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

The screenshot shows a Jupyter Notebook interface with a code cell containing the command `data.info()`. The output is a detailed summary of the dataset's structure:

```
data.info()
[27]    ✓  0.0s
...
<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.)

The screenshot shows a Jupyter Notebook interface with a code cell containing the command `data.describe()`. The output is a summary statistics table:

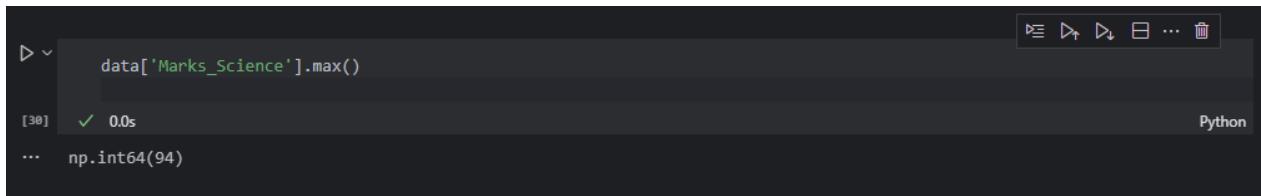
	Student_ID	Age	Marks_Math	Marks_Science
count	10.000000	10.000000	10.000000	10.000000
mean	5.500000	20.900000	86.000000	85.100000
std	3.027650	1.197219	6.218253	7.202623
min	1.000000	19.000000	76.000000	74.000000
25%	3.250000	20.000000	81.250000	79.750000
50%	5.500000	21.000000	87.500000	87.000000
75%	7.750000	21.750000	89.750000	90.750000
max	10.000000	23.000000	95.000000	94.000000

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

The screenshot shows a Jupyter Notebook interface with a code cell containing the command `data['Marks_Math'].mean()`. The output is the mean value of the Math marks:

```
data['Marks_Math'].mean()
[29]    ✓  0.0s
...
...     np.float64(86.0)
```

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

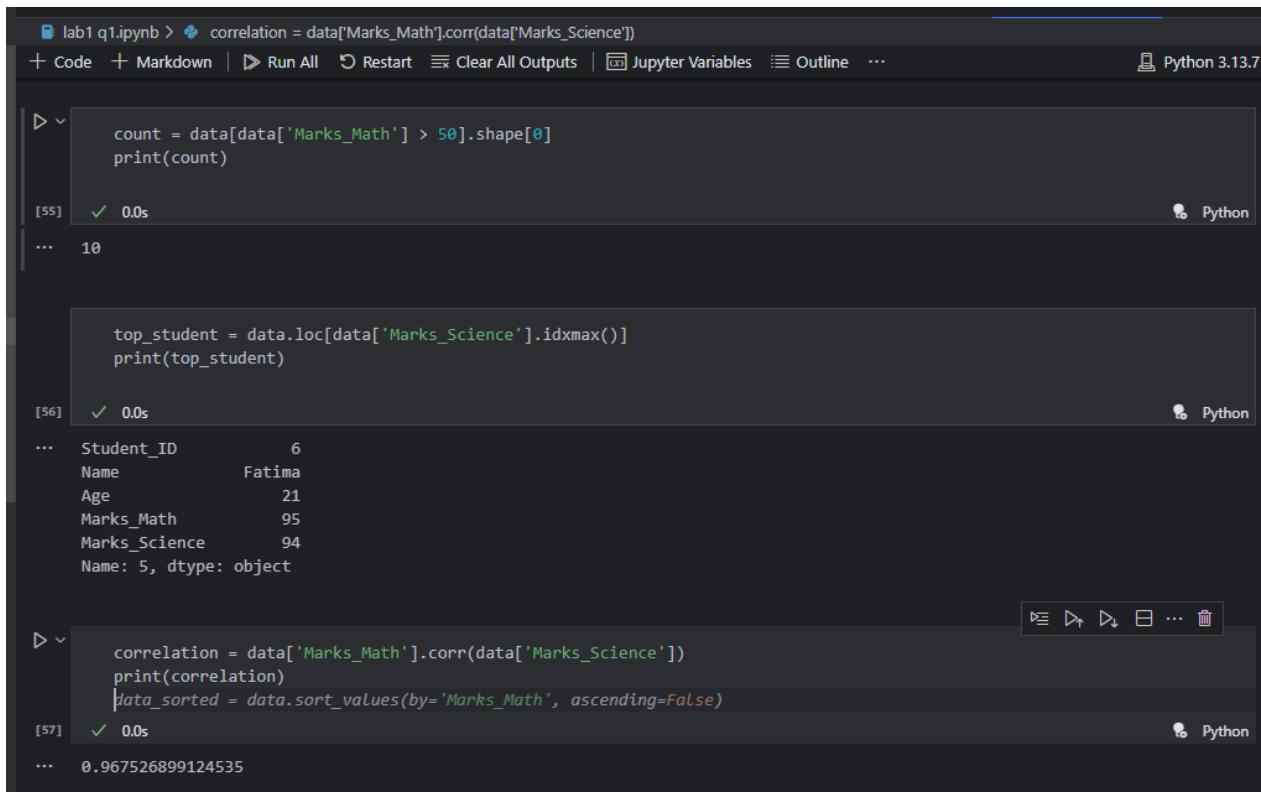


```
[30]: data['Marks_Science'].max()
...     0.0s
...     np.int64(94)
```

The screenshot shows a Jupyter Notebook cell with the code `data['Marks_Science'].max()`. The output is `0.0s` followed by `np.int64(94)`. The cell is labeled [30]. The Python logo is visible in the top right corner.

## 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)
... 0.0s
... 10

top_student = data.loc[data['Marks_Science'].idxmax()]
print(top_student)
... Student_ID      6
... Name        Fatima
... Age         21
... Marks_Math    95
... Marks_Science 94
... Name: 5, dtype: object

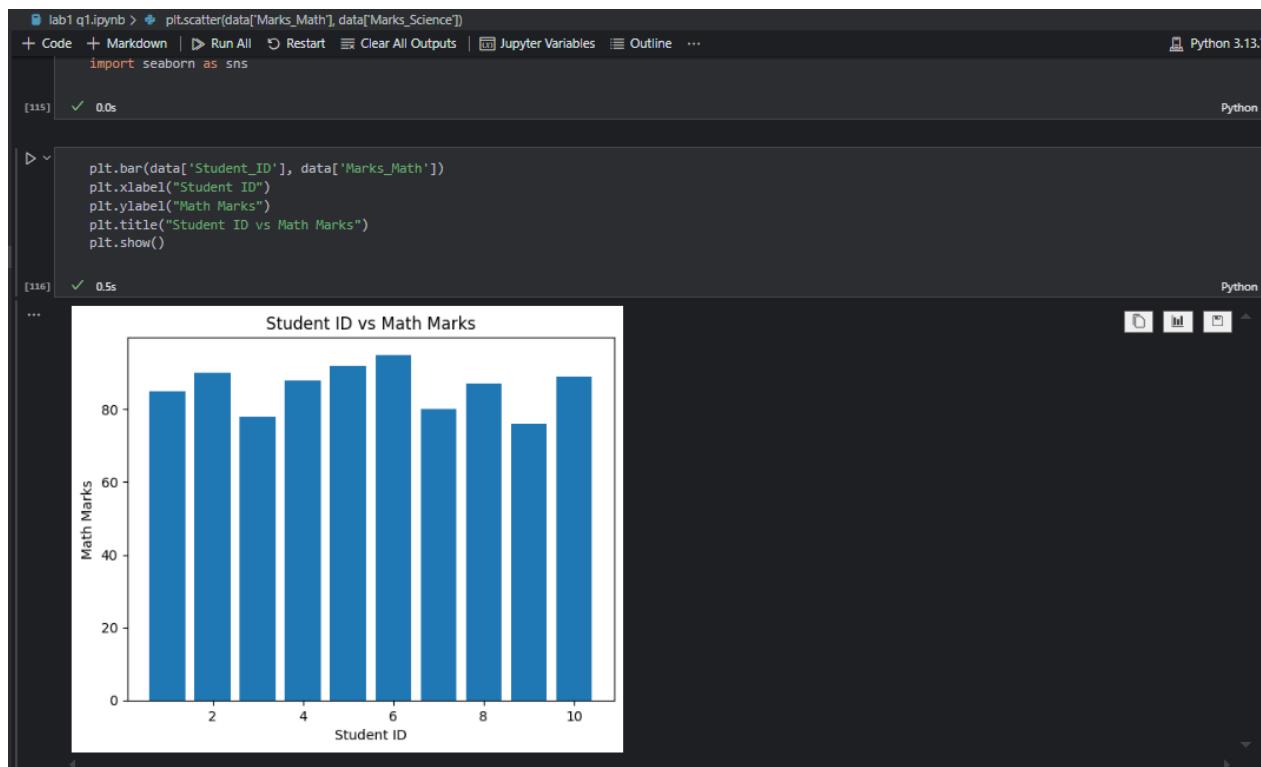
▶ correlation = data['Marks_Math'].corr(data['Marks_Science'])
print(correlation)
... data_sorted = data.sort_values(by='Marks_Math', ascending=False)
... 0.0s
... 0.967526899124535
```

The screenshot shows a Jupyter Notebook with three code cells. Cell 55: `count = data[data['Marks_Math'] > 50].shape[0]`, `print(count)`, output: `10`. Cell 56: `top_student = data.loc[data['Marks_Science'].idxmax()]`, `print(top_student)`, output: a Series with index 'Student\_ID' (value 6), 'Name' (value 'Fatima'), 'Age' (value 21), 'Marks\_Math' (value 95), 'Marks\_Science' (value 94), and 'Name' (Series). Cell 57: `correlation = data['Marks_Math'].corr(data['Marks_Science'])`, `print(correlation)`, output: `0.967526899124535`. The Python logo is visible in the top right corner.

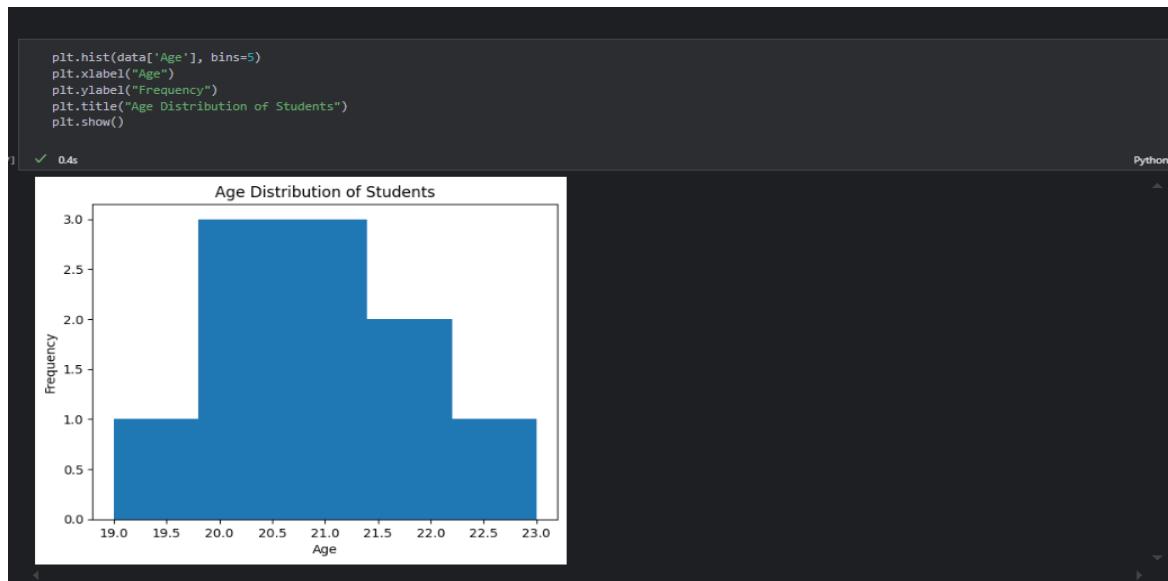
## Q5: Data Visualization

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.

