

LAB No. 1

Introduction to VS Code and Google Colab for Data Analysis Using Python

LAB Description

In this lab, students will learn how to work with Python programming environments using Visual Studio Code (VS Code) and Google Colab. VS Code is a lightweight and powerful source-code editor that runs on a local system and supports Python development through extensions. It allows users to write, run, and debug Python programs efficiently on their own computer.

Google Colab is a cloud-based Python notebook environment provided by Google that runs in a web browser. It does not require any local installation and provides free access to computing resources. Colab is especially useful for data analysis and visualization, as it supports interactive notebooks, built-in libraries, and easy file uploads.

Using either VS Code or Google Colab, students will create a dataset in Python, upload or load the data, and perform basic data analysis operations. The lab focuses on understanding how datasets are handled, how simple statistics are calculated, and how graphical representations help in interpreting data.

Lab Objective

The objectives of this lab are:

- To understand the basic functioning of VS Code and Google Colab
- To learn how to create and upload a dataset using Python
- To perform basic statistical analysis (such as mean, median, and count)
- To visualize data using simple graphs (line charts, bar charts, or histograms)
- To develop foundational skills in data analysis and visualization using Python

How to use python in VS code?

1. Create environment named 'venv'

python -m venv

2. Activate environment

venv\Scripts\activate

3. Select Environment in VS Code After creating the environment:

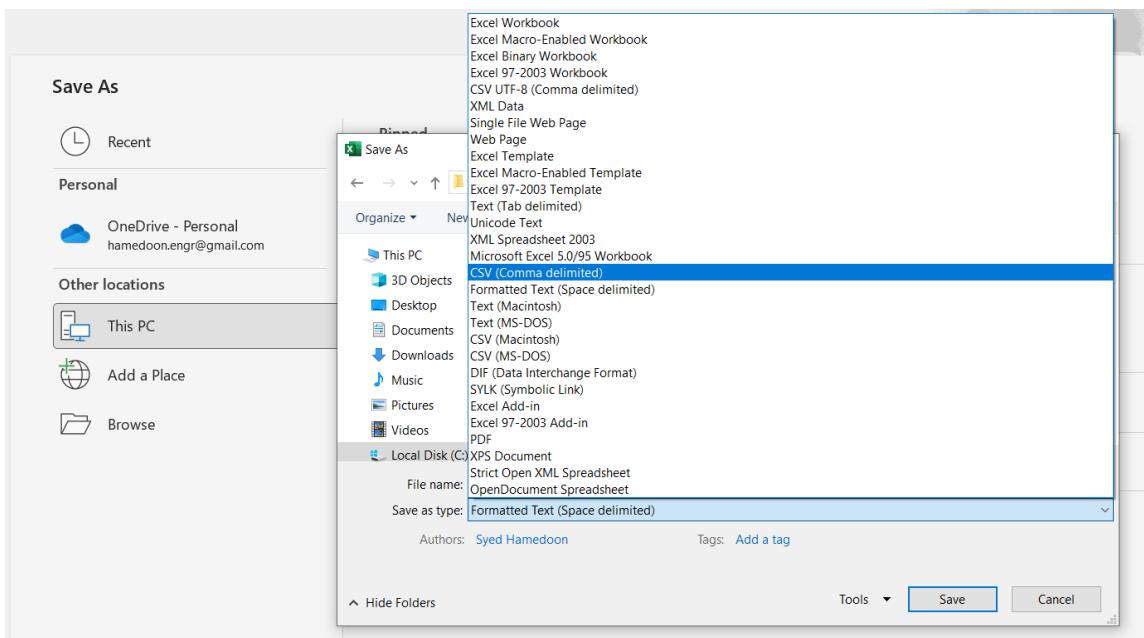
- 1) Open your project folder in VS Code.
- 2) Press Ctrl + Shift + P → search for Python:
- 3) Select Interpreter.
- 4) Choose your newly created environment (venv or myenv).

4. Install Machine Learning and Deep Learning Libraries

- pip install --upgrade pip
- pip install pandas
- pip install matplotlib
- pip install seaborn
- pip install scikit-learn
- pip install scipy
- pip install numpy
- pip install xgboost
- pip install lightgbm
- pip install catboost
- pip install tensorflow
- pip install keras
- pip install torch

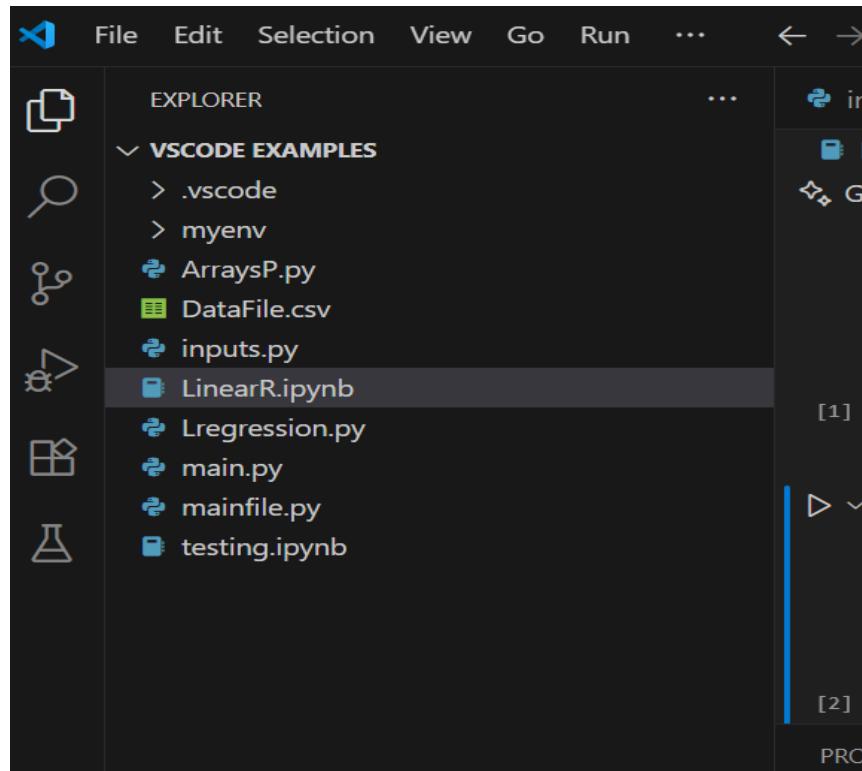
Task 1:

Save Excel file as .csv (Comma delimited)



Task 2:

Save .csv file in directory and folder where we created a python environment



Here we can see our file successfully

The screenshot shows a Google Colab notebook interface. At the top, there are tabs for 'inputs.py', 'Lregression.py', 'LinearR.ipynb', 'Untitled-1.ipynb' (which is the active tab), and 'test.ipynb'. Below the tabs are navigation buttons: 'Generate', '+ Code', '+ Markdown', 'Run All', 'Restart', 'Clear All Outputs', and 'Output' settings. The main area displays a Jupyter-style code cell history:

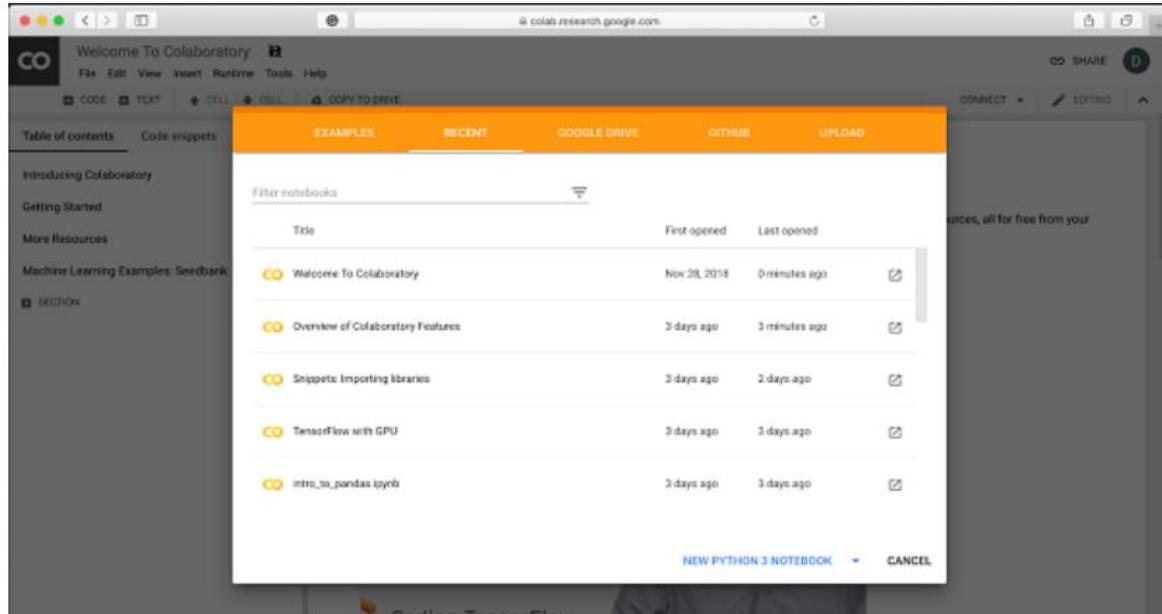
- [2] `import pandas as pd` ✓ 0.7s
- [3] `data=pd.read_csv("./DataFile.csv")` ✓ 0.1s
- [4] `data.head()` ▶ ✓ 0.1s

...
Sr No SAP_ID Name Subject University
0 1 70144780 MUHAMMAD HASSAN MADNI AI UOL
1 2 70144904 QAZI ZARYAB SAJJAD AI UOL
2 3 70144910 MOHTISHIM FAREED AI UOL
3 4 70145312 SHIZA ISHAQ AI UOL
4 5 70145452 MALAIKAH KHALID AI UOL

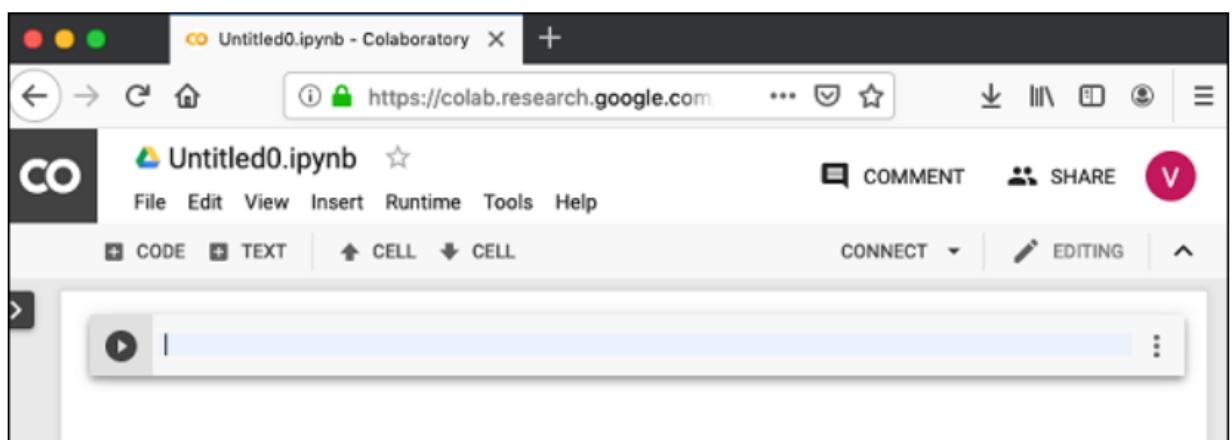
Introduction of Google Colab

- Colab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs.
- Colab supports many popular machine learning libraries which can be easily loaded in your notebook.
- Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science.
- Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook

- **Step 1** – Open the following URL in your browser
 – <https://colab.research.google.com> Your browser would display the following screen (assuming that you are logged into your Google Drive) –

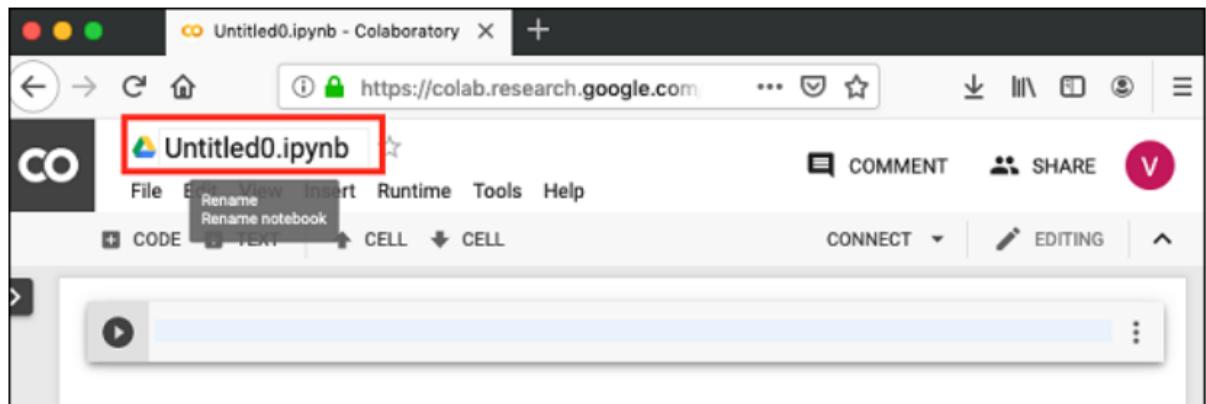


- **Step 2** – Click on the **NEW PYTHON 3 NOTEBOOK** link at the bottom of the screen. A new notebook would open up as shown in the screen below.

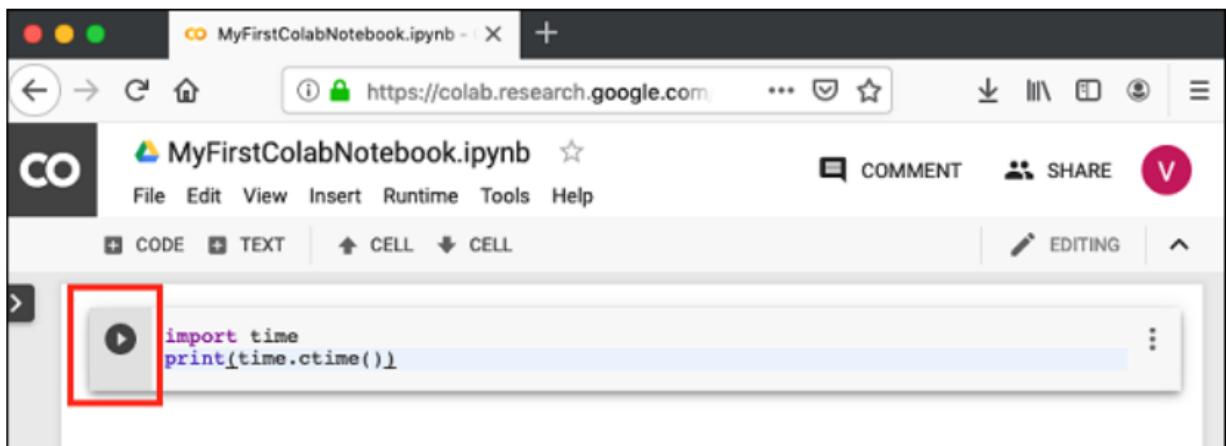


- **Setting Notebook Name**

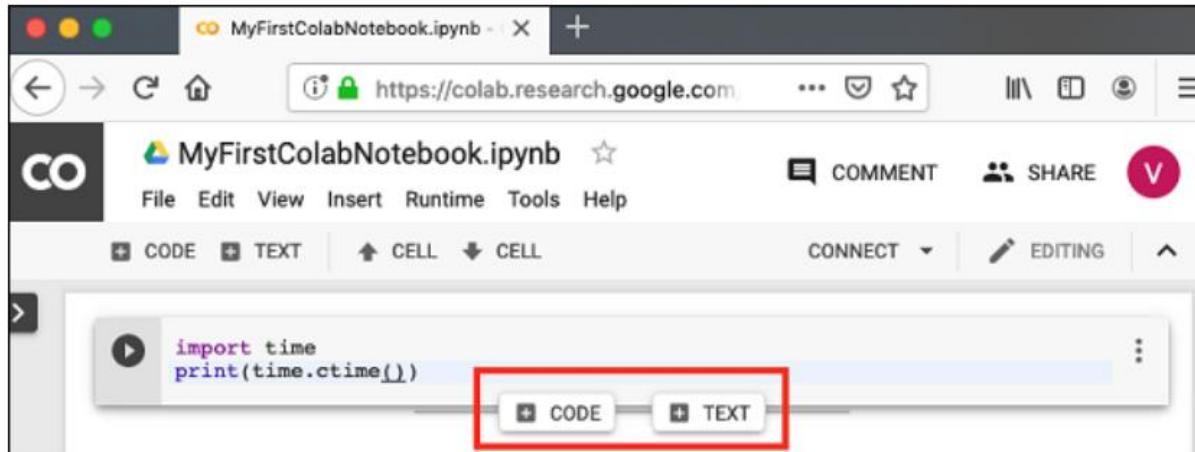
By default, the notebook uses the naming convention UntitledXX.ipynb. To rename the notebook, click on this name and type in the desired name in the edit box as shown here –



- Entering Code



- Adding Code Cells
- To add more code to your notebook, select the following **menu** options –

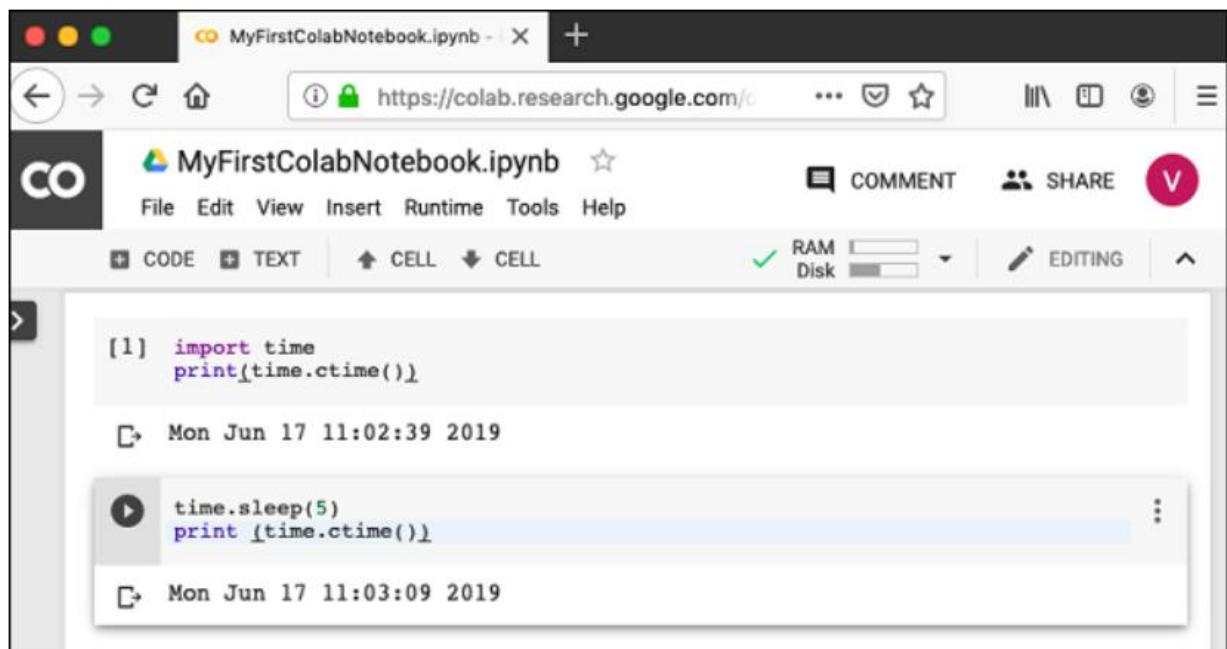


A screenshot of the Google Colab interface. At the top, there's a dark header bar with the title "MyFirstColabNotebook.ipynb" and a "+" button. Below it is a browser-style toolbar with back, forward, home, and search buttons. The main area shows a notebook titled "MyFirstColabNotebook.ipynb". The toolbar below the title has "CODE" and "TEXT" buttons, both of which are highlighted with a red box. The notebook content shows a single code cell containing the Python code:

```
import time  
print(time.ctime())
```

- **Run All**

To run the entire code in your notebook without an interruption, execute the following menu options –



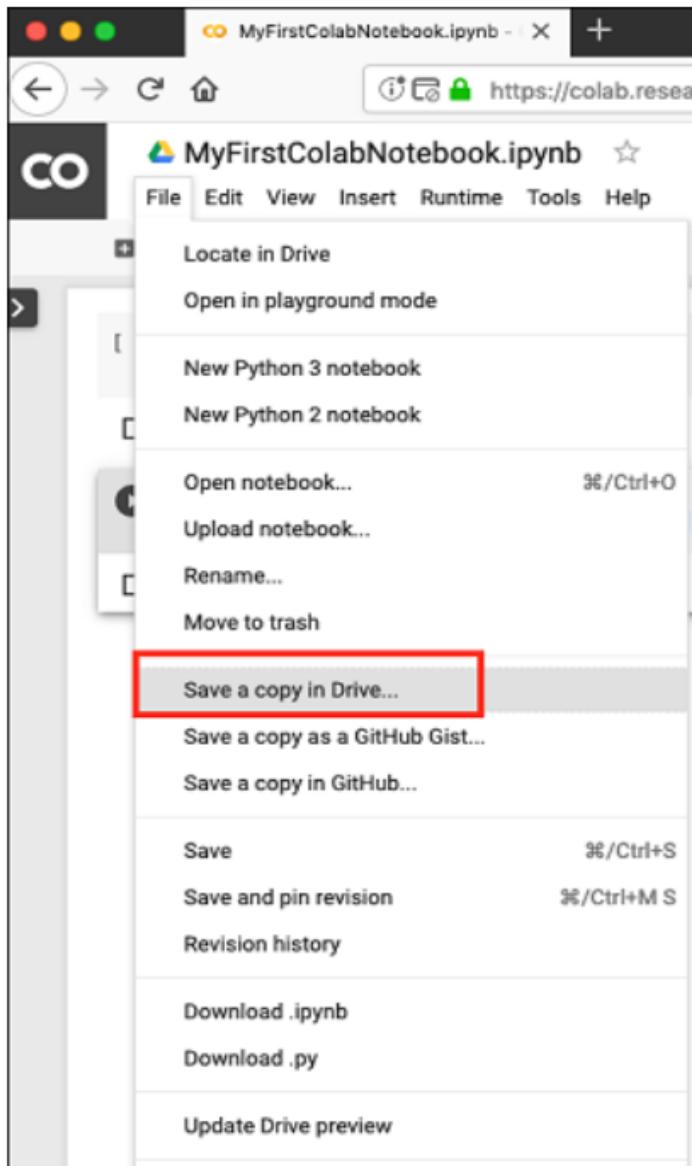
A screenshot of the Google Colab interface after running all cells. The toolbar now shows "RAM" and "Disk" status indicators. The notebook content displays two cells. The first cell contains the code and its output:

```
[1] import time  
print(time.ctime())  
Mon Jun 17 11:02:39 2019
```

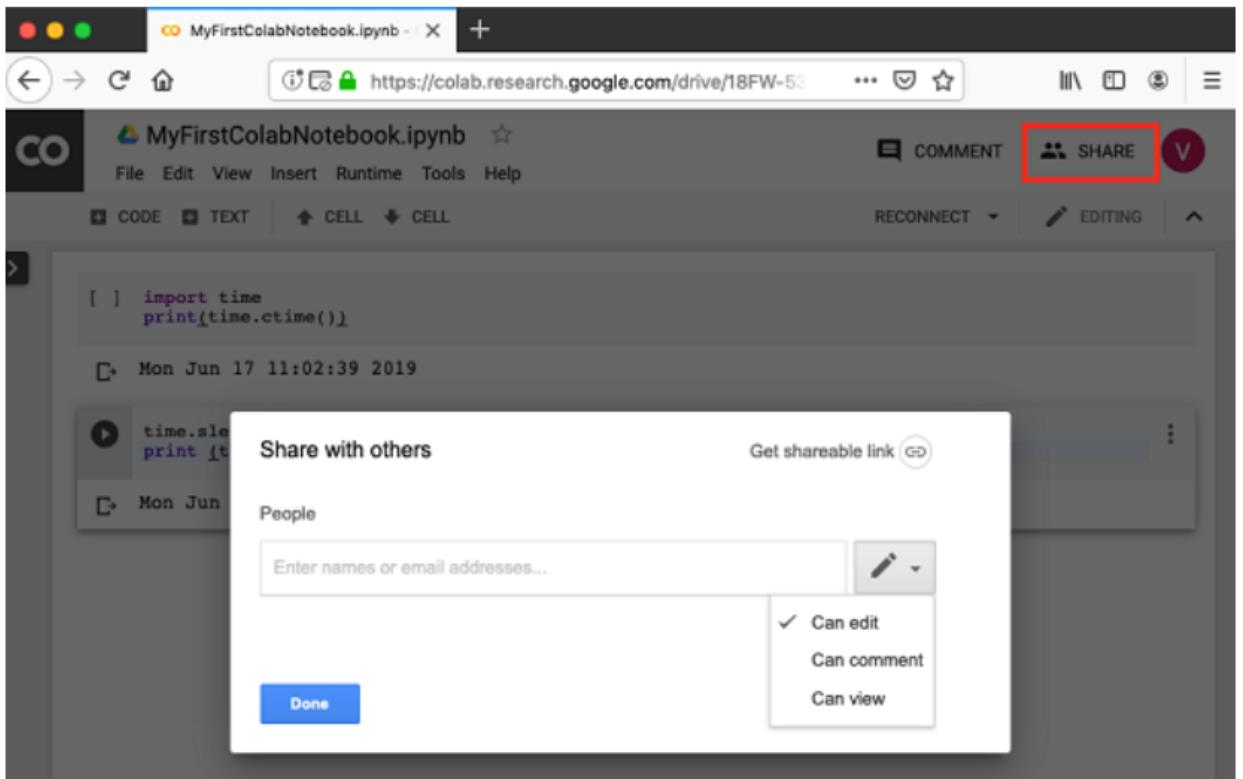
 The second cell contains the code and its output:

```
time.sleep(5)  
print(time.ctime())  
Mon Jun 17 11:03:09 2019
```

Saving to Google Drive



Google Colab - Sharing Notebook



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:
 - 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 a dark theme. The top bar includes tabs for 'Code' and 'Markdown', and buttons for 'Run All', 'Restart', 'Clear All Outputs', 'Jupyter Variables', 'Outline', and 'Python 3.13.7'. The code cell [6] contains the command `import pandas as pd`, which has a green checkmark and a duration of 0.0s. The code cell [7] contains `data = pd.read_csv("students.csv")`, also with a green checkmark and 0.0s duration. The code cell [8] contains `print(data)` and `print(data.describe())`, both with a green checkmark and 0.0s duration. Below these cells, the resulting DataFrame 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

```
lab1_Q1.ipynb > Data[Marks_Science].max()
+ Code + Markdown | Run All ⚡ Restart ⌂ Clear All Outputs ⌂ Jupyter Variables ⌂ Outline ... Python 3.13.7

D v
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    10.000000  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

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

[29] ✓ 0.0s Python

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

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

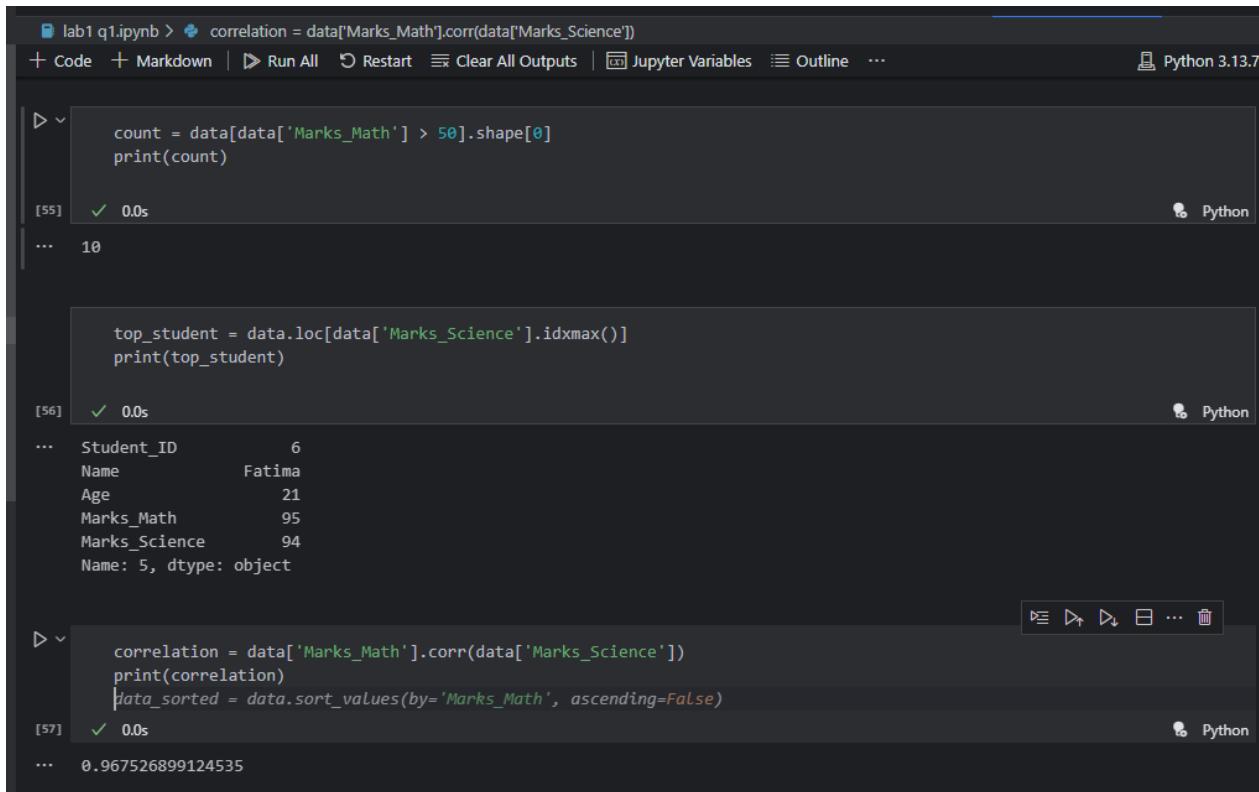
```
D v
data['Marks_Science'].max()

[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
... 10

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

[56] ✓ 0.0s
... Student_ID      6
     Name        Fatima
     Age         21
     Marks_Math    95
     Marks_Science 94
     Name: 5, dtype: object

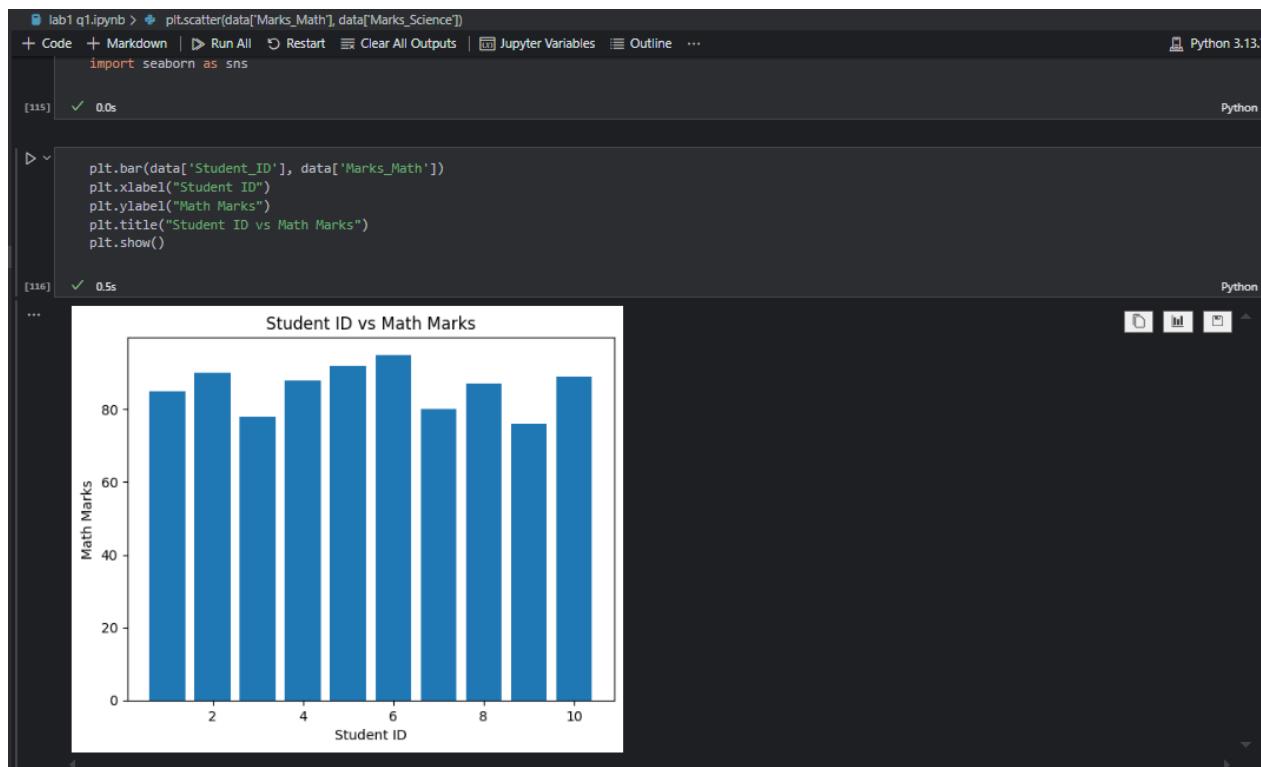
  correlation = data['Marks_Math'].corr(data['Marks_Science'])
  print(correlation)
  data_sorted = data.sort_values(by='Marks_Math', ascending=False)

[57] ✓ 0.0s
... 0.967526899124535
```

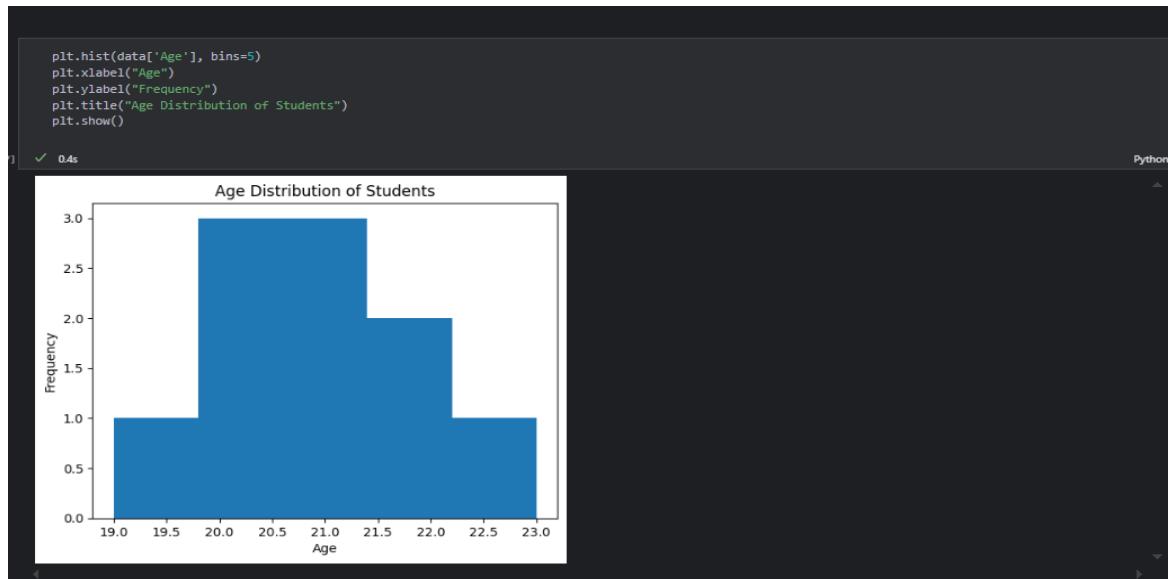
Q5: Data Visualization

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.

