**Name:** V Venkata Sri Prasad

**Batch:** Data Engineering

**Date:** 1/02/2024 – (Day 11)

Topics:

1. Creating dataframes using dynamic column list on csv data and inner join
2. Enriching data using numpy and pandas
3. Get count by status using Pandas Dataframe
4. Read JSON schemas from file to python dicts

**Introduction to NumPy:**

Importing NumPy and aliasing it as np.

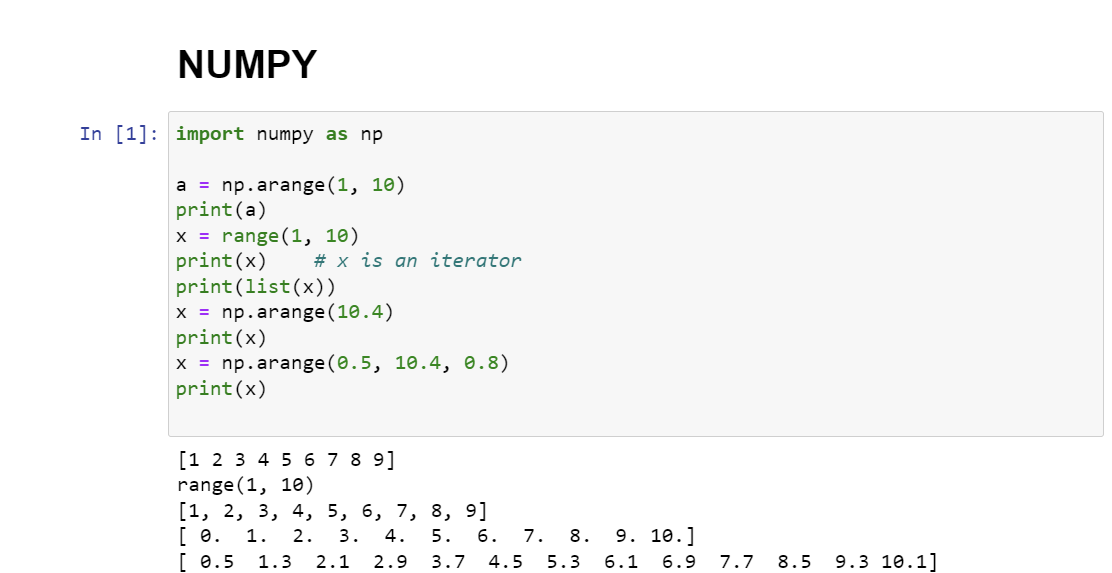
Creating a one-dimensional NumPy array from a list of Celsius temperatures.

**Creating Arrays with Evenly Spaced Values using arange:**

Using np.arange to create an array of evenly spaced values within a given interval.

Comparing it with the built-in Python range function.

Providing examples of using arange with different start, stop, and step parameters.



**Creating Arrays with Evenly Spaced Values using linspace:**

Introducing np.linspace to create an array with a specified number of evenly spaced values.

Demonstrating how to exclude the endpoint.

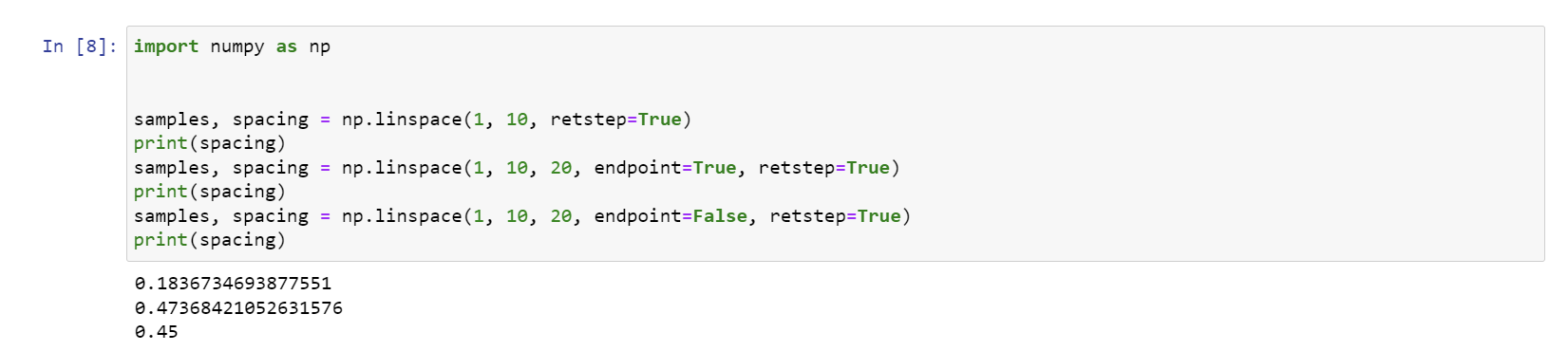
Explaining the parameters such as start, stop, num, and endpoint.



**Optional Parameter retstep in linspace:**

Explaining the optional parameter retstep in np.linspace.

Showing how to retrieve the step value between adjacent elements when using retstep.



**Reading CSV Files with Pandas:**

Method #1: Using read\_csv() method

Importing the Pandas library.

Using pd.read\_csv() to create a DataFrame from a CSV file.

Example:



Method #2: Using read\_table() method

Utilizing pd.read\_table() to create a DataFrame from a CSV file with a specified delimiter.

Example:



Method #3: Using the csv module

Reading a CSV file using the csv module and then creating a DataFrame with Pandas.

Example:



Join Operations in Pandas:

Types of Joins:

Inner Join: Returns rows with common characteristics in both DataFrames.



Left Outer Join: Displays all records from the first DataFrame, matching records from the second DataFrame.



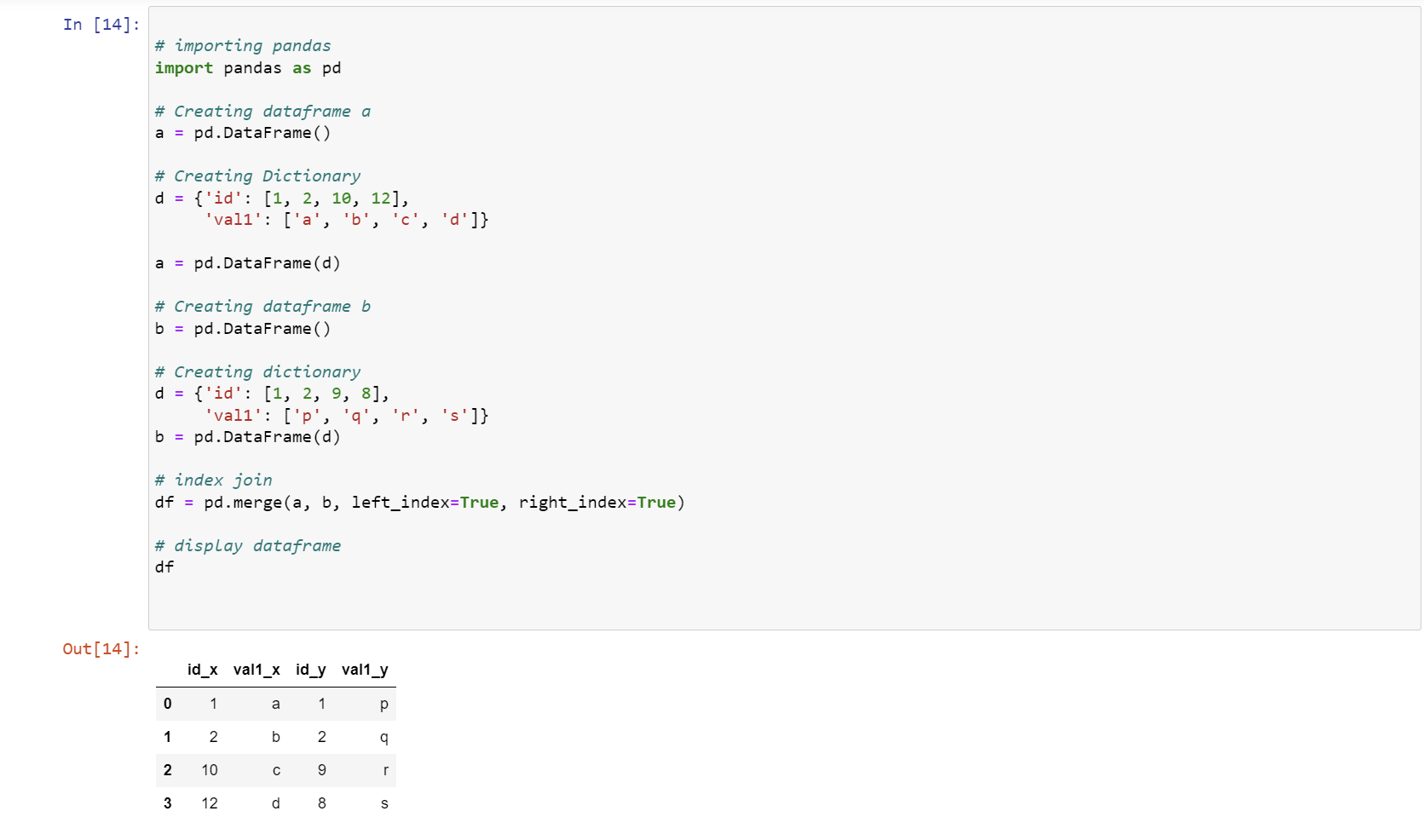
Right Outer Join: Displays all records from the second DataFrame, matching records from the first DataFrame.



Full Outer Join: Returns all rows from both DataFrames, matching up rows where possible, with NaNs elsewhere.



Index Join: Merges DataFrames based on their indices using the default Inner Join.

****

**Get Count by Status using Pandas Dataframe APIs**

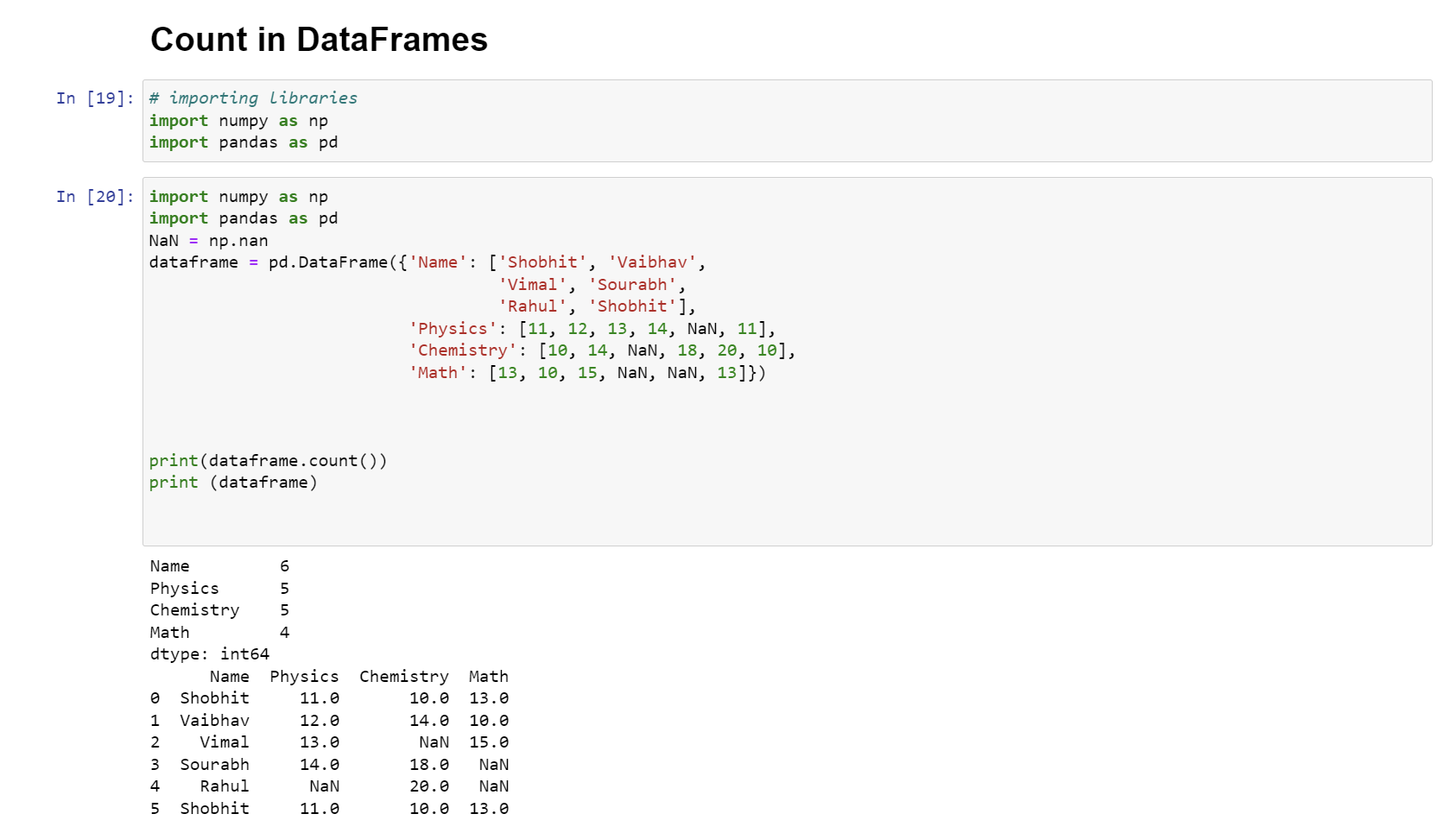
**Step 1: Importing Libraries**

Importing necessary libraries, namely numpy and pandas.

**Step 2: Creating DataFrame**

Creating a DataFrame with some missing values using the pd.DataFrame constructor.

Demonstrating the presence of NaN (Not a Number) values in the DataFrame.

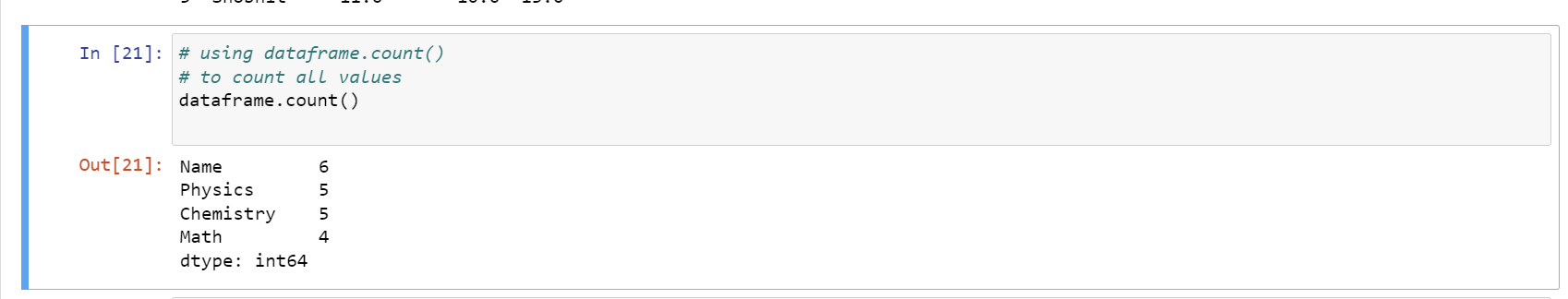


**Step 3: Using dataframe.count()**

Utilizing the .count() function to count non-null values for each column.

Printing the DataFrame to display its structure.

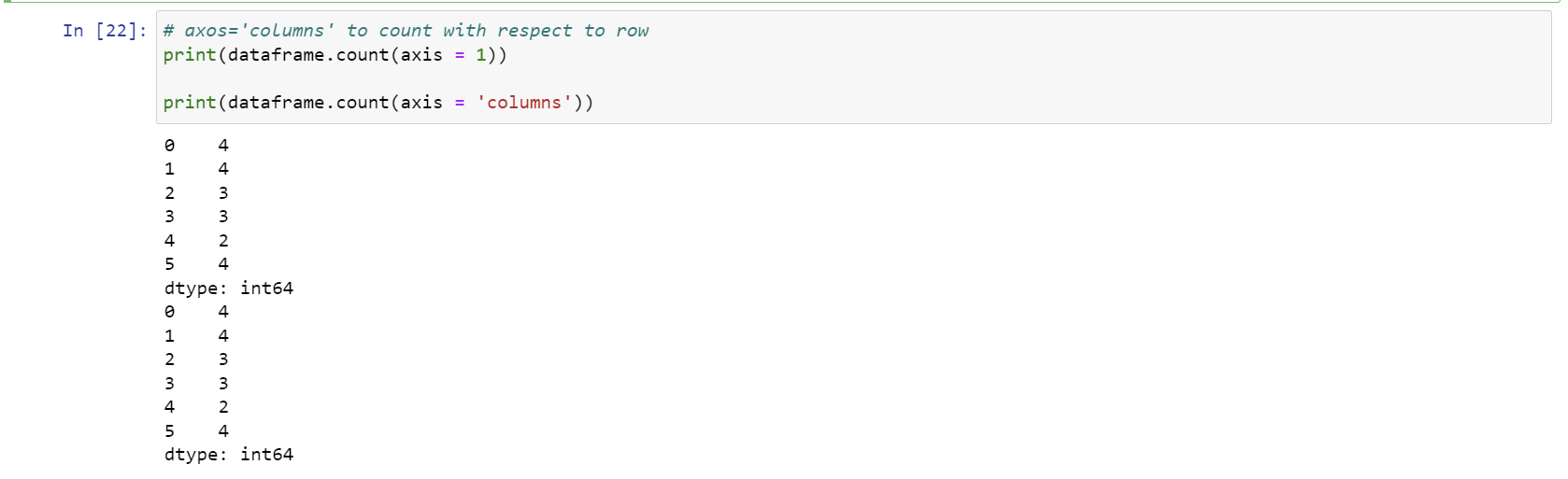
Outputting the count of non-null values in each column.



**Step 4: Counting with Respect to Rows**

Using axis=1 or axis='columns' to count values with respect to rows.

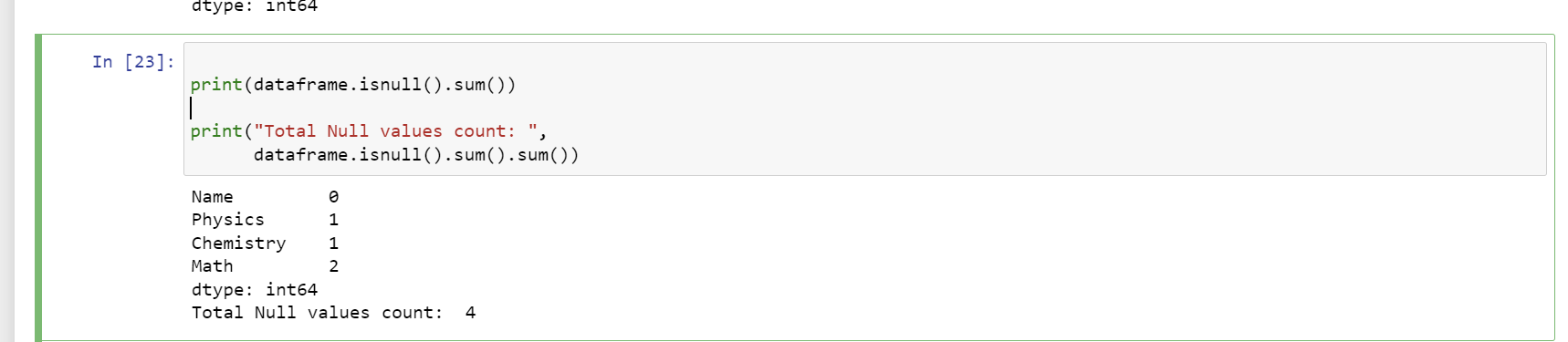
Displaying the count for each row.



**Step 5: Counting Null Values**

Using .isnull().sum() to count null values for each column.

Displaying the count of total null values in the DataFrame.

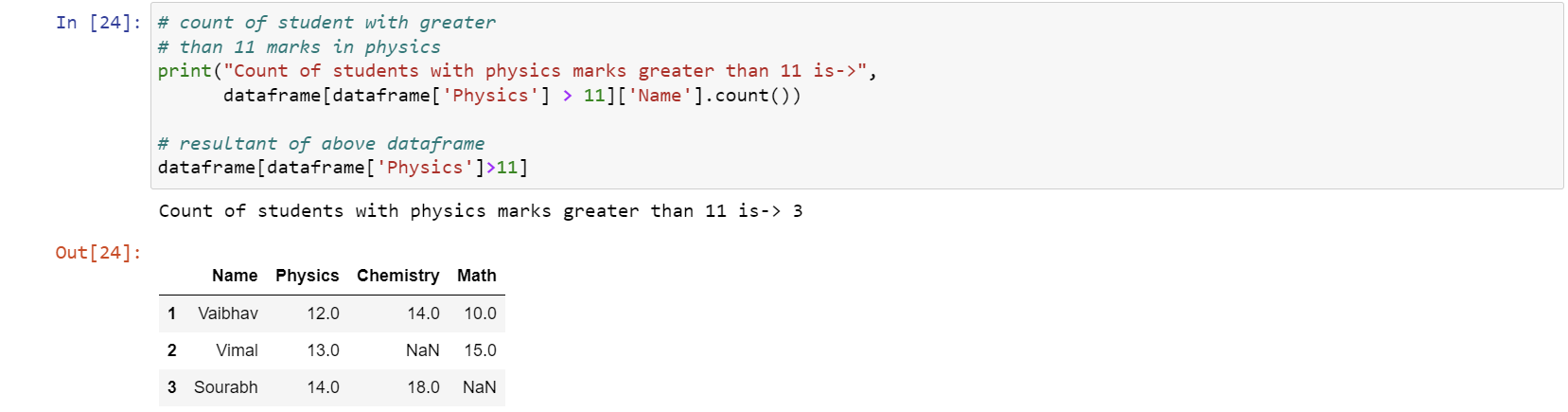


**Step 6: Examples Using .count()**

Counting students with physics marks greater than 11.

Counting students with physics marks greater than 10, chemistry marks greater than 11, and math marks greater than 9.

Displaying the resulting DataFrames for the above conditions.



**Full Implementation**

Combining all the steps into a complete Python script.

Creating the DataFrame, performing counts, and showcasing examples with conditions.

These topics cover various aspects of counting values in a Pandas DataFrame, including non-null values, null values, and specific conditions.



