



|  |  |
| --- | --- |
| **Student Name** | **Aarya Trifale** |
| **SRN No** | 31241458 |
| **Roll No** | 72 |
| **Program** | Computer Engineering |
| **Year** | Second Year |
| **Division** | E |
| **Subject** | Project Based Learning-Python (BTECCE23306) |
| **Assignment No** | 13 |



Assignment Number - 13

**Title /Problem Statement:**

Write a program to read data from a CSV file using Pandas. Display the first few rows (head()), information (info()), and basic statistics (describe()) of the DataFrame. Save the modified DataFrame to a new CSV file.

# Description:

Write a Python program to read data from a CSV file using the Pandas library. First, load the CSV file into a DataFrame and then display the first few rows with `head()`, the structure and summary of the DataFrame with `info()`, and basic statistics of the data using `describe()`. Finally, save the modified DataFrame to a new CSV file to preserve any changes made during data analysis or manipulation.

**Theory:**

Pandas is a powerful data manipulation library in Python, widely used for data analysis and handling structured data. Reading data from a CSV (Comma-Separated Values) file into a Pandas DataFrame is straightforward using the read\_csv() function. This function reads the CSV file and loads its content into a DataFrame, a tabular data structure with labeled axes (rows and columns).

To begin, you import the pandas library and use pd.read\_csv('filename.csv') to read the CSV file. The head() method displays the first few rows of the DataFrame, allowing a quick glance at the data. The info() method provides a concise summary of the DataFrame, including the number of non-null entries, data types of columns, and memory usage, which helps in understanding the dataset's structure.

The describe() method generates descriptive statistics of the numerical columns in the DataFrame, such as mean, standard deviation, minimum, and maximum values, along with the 25th, 50th (median), and 75th percentiles. This summary gives a quick statistical overview of the data.

After exploring and possibly modifying the DataFrame (e.g., adding, removing, or transforming columns), you can save the modified DataFrame back to a CSV file using the to\_csv('new\_filename.csv', index=False) method. The index=False argument ensures that row indices are not written to the new CSV file, maintaining a clean format.

**Experimental Setup:**

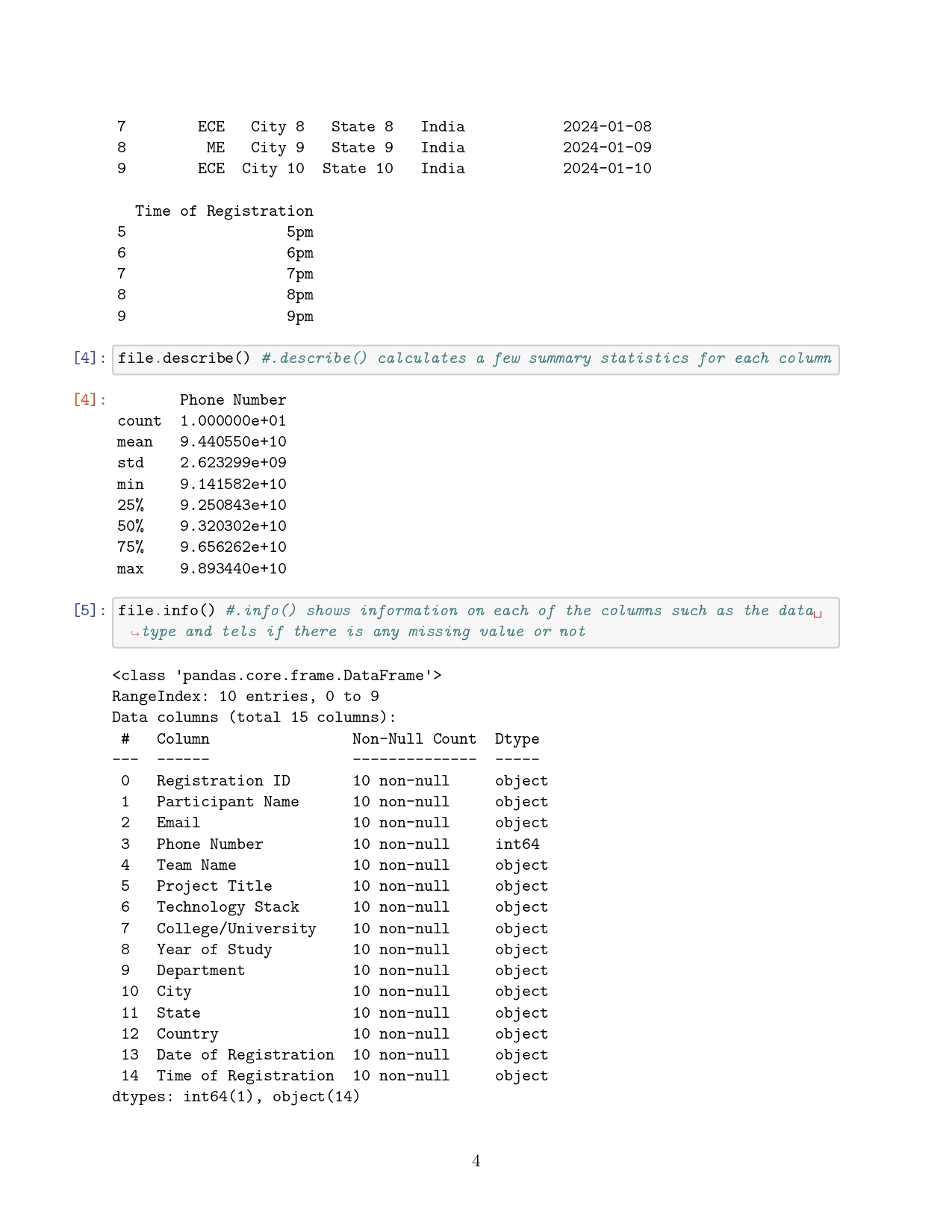
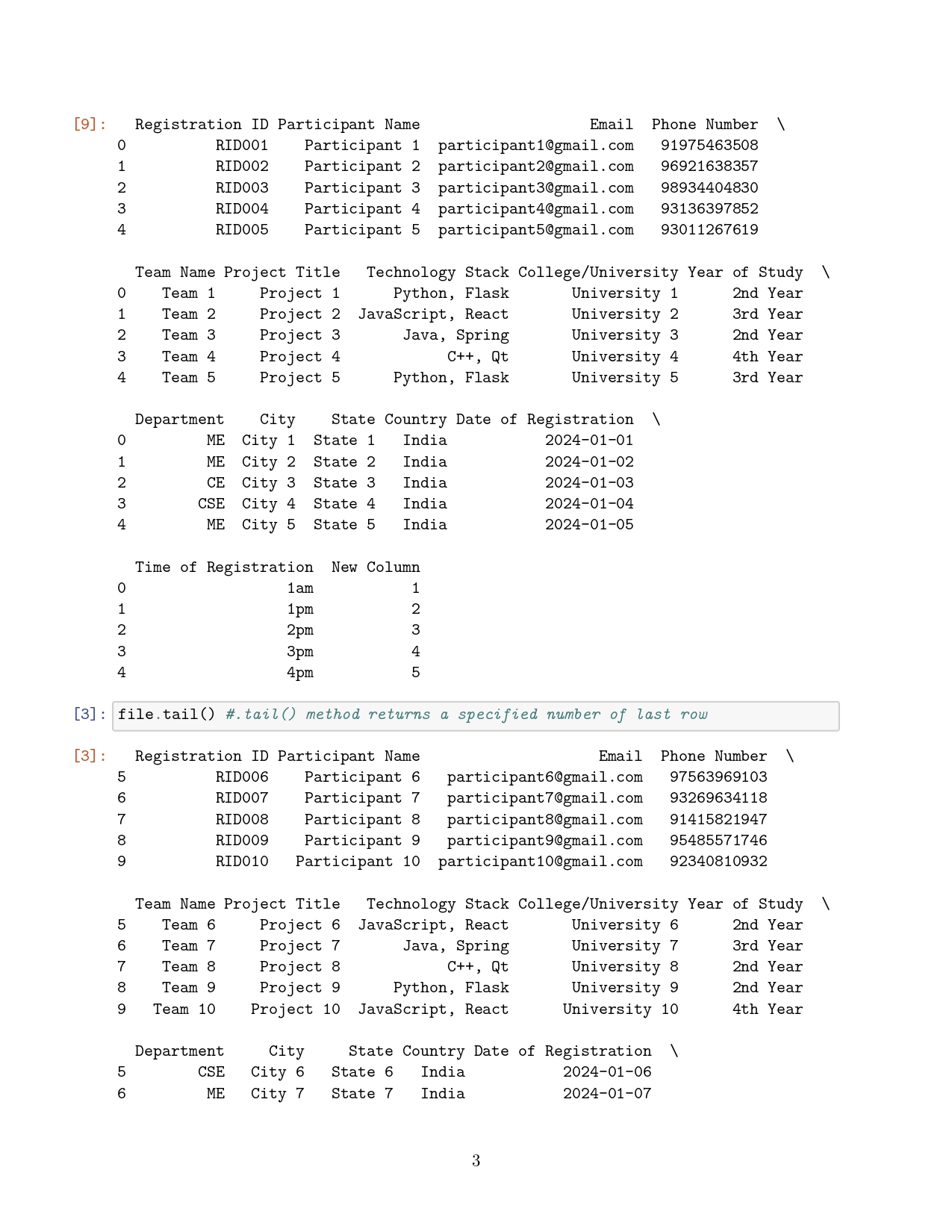
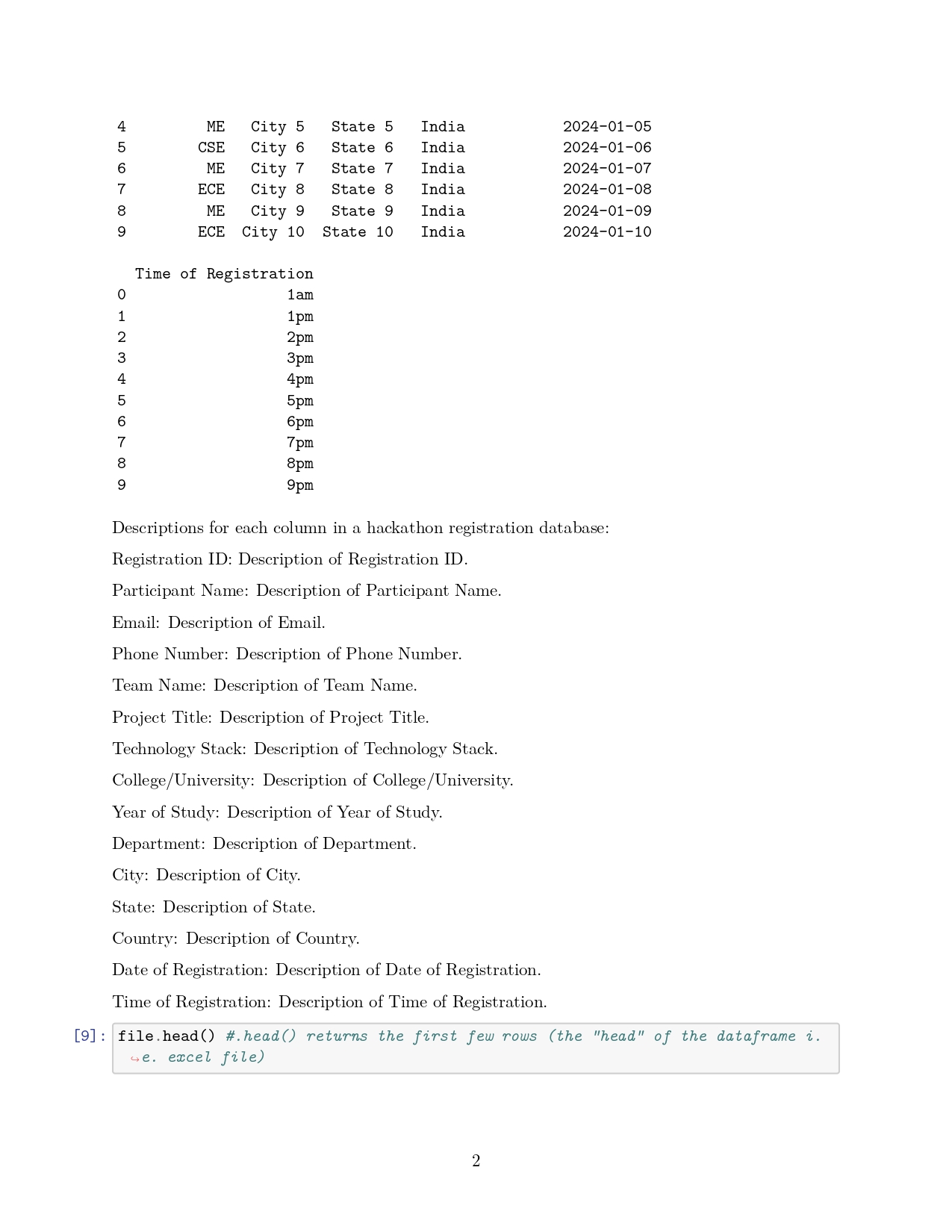
Complier/IDE used- Jupyter Notebook

Microsoft Windows Edition- Windows 11 pro

Device used- Laptop (Lenovo ideapad i5)

**Experimental Outcome:**

Input and output:

****

**Explanation:**

1. Read Data from CSV: pd.read\_csv(file\_path) reads the data from the specified CSV file and creates a DataFrame.

2. Display First Few Rows: df.head() shows the first few rows of the DataFrame, providing a quick preview of the data.

3. Display Information: df.info() provides a summary of the DataFrame, including data types, number of non-null values, and memory usage.

4. Display Basic Statistics: df.describe() generates descriptive statistics like mean, standard deviation, and min/max values for numeric columns.

5. Modify and Save Data: The DataFrame is optionally modified (e.g., adding a new column) and then saved to a new CSV file using df.to\_csv(output\_file\_path, index=False).

The index=False argument prevents writing row indices to the CSV file. Replace file\_path and output\_file\_path with the actual paths to your CSV files and adjust column names as needed.

**Conclusion:**

In conclusion, using Pandas to read, explore, and save data from a CSV file streamlines data analysis by providing easy access to initial data inspection and manipulation, ensuring efficient handling and storage of modified datasets