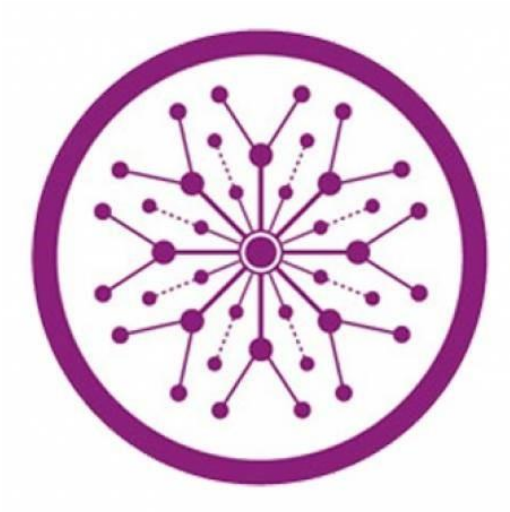
Artificial Intelligent (Lab)

Task # 09



Submitted To: Sir Rasikh Ali

Submitted By: Shumaila Maryam

Roll no: SU92-BSDSM-F24-062

Section: BSDS-3A

Department of Software Engineering,
Superior University, Lahore

Introduction:

This project documents the essential first steps in any data science workflow: **Data Loading, Reading, and Exploration**. Using the Python programming language and the powerful **Pandas** library, the goal is to transform raw data from a CSV file into a structured Data Frame. This allows for a comprehensive initial assessment of the dataset's quality, structure, and content, which is crucial before any advanced cleaning, feature engineering, or model training can begin.

Why I Made This Program (Purpose of Initial Exploration):

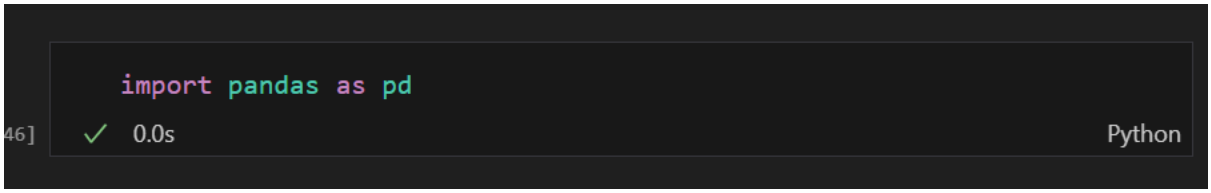
The main purpose behind performing these initial data exploration steps is to establish a strong foundational understanding of the data. This crucial phase helps to:

- **Verify Data Integrity:** Confirm that the data has been read correctly and identify any potential errors introduced during file transfer.
- **Understand Data Structure:** Determine the number of rows (observations) and columns (features) in the dataset.
- **Identify Data Types:** Check the data type (e.g., integer, float, object/string) of each column to ensure they are appropriate for subsequent analysis.
- **Detect Missing Values (Data Quality):** Quantify how many null or missing entries exist in each column, which informs the strategy for data cleaning (imputation or deletion).
- **Assess Feature Composition:** Analyse the unique values and ranges of key features to gauge their distribution and variability.

This process serves as the vital link between raw data acquisition and the development of a high-quality machine learning model.

How It Works:

1. **Import Libraries**

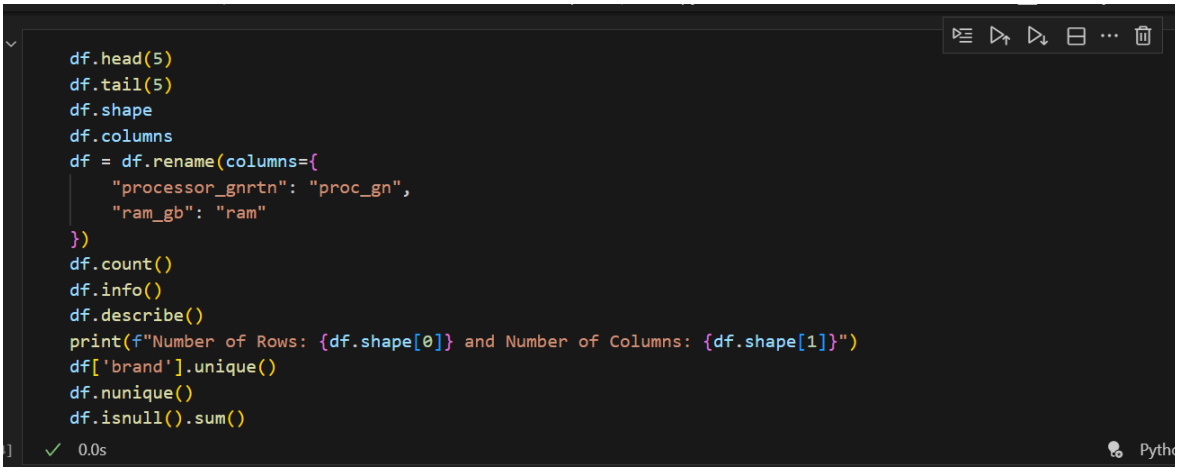


Imports the Pandas library, the primary tool for data manipulation and analysis in Python. It is aliased as pd for convenient and standard usage throughout the code.

2. Data Loading/Reading

This is the command that reads the external data file ("data.csv") into the environment. The pd.read_csv() function parses the CSV file, converts it into a tabular structure called a DataFrame, and assigns it to the variable df (DataFrame).

3. Data Exploration



1. **df.head(5)**: Displays the **first 5 rows** of the DataFrame. This is the first visual check to confirm the data loaded correctly, assess column headings, and inspect the initial format of the data entries.
2. **df.tail(5)**: Displays the **last 5 rows** of the DataFrame. This is useful for checking if the data terminated cleanly or if there are any summary rows or unusual values at the end of the file.
3. **df.shape**: Returns a **tuple** representing the dimensions of the DataFrame: (Number of Rows, Number of Columns). This provides the total size of the dataset.

4. **df.columns**: Returns an **Index list** containing the names of all the columns in the DataFrame. This is useful for accurately referencing columns later.
5. **df = df.rename(columns={...})**: **Column Renaming**.
   The rename() function changes the names of specified columns. This is often done to simplify long column names (processor_gnrtn to proc_gn) for easier coding and readability. The DataFrame df is then updated with these new names.
6. **df.count()**: Calculates the **number of non-null (non-missing) values** for every single column. This gives a column-by-column count of valid entries and helps identify columns that immediately have missing data if the count is less than the total number of rows.
7. **df.info()**: Provides a **concise, complete summary** of the DataFrame, including the number of entries, the number of columns, a list of all column names, their non-null counts, and their **Data Types (Dtype)**. This is generally the most important single command for a quick data quality check.
8. **df.describe()**: Generates a table of **descriptive statistics** for all numerical columns. The output includes count, mean, standard deviation (std), minimum (min), maximum (max), and the quartile values (25%, 50%, 75%). This is vital for understanding the distribution and central tendency of the numerical features.
9. **print(f"Number of Rows: {df.shape[0]} and Number of Columns: {df.shape[1]}")**: A simple **print statement** to clearly output the dimensions of the dataset in a user-friendly sentence format.
10. **df['brand'].unique()**: **Categorical Data Inspection**. This targets a specific column (brand) and returns a NumPy array containing every **unique value** found in that column. This is critical for identifying potential spelling errors, inconsistencies, or the full range of categories present.
11. **df.nunique()**: Calculates the **total number of unique values** for every column in the DataFrame. This helps quickly identify categorical columns (low unique count) versus continuous/ID columns (high unique count).

12. **df.isnull().sum()**: **Quantifying Missing Data**. This command first checks every cell for a null value (isnull()), then converts the resulting boolean DataFrame into numerical form (True=1, False=0), and finally sums the True values column-wise (sum()). The output is a clear count of missing values for each feature.

Summary:

The code comprehensively executes the initial phase of data analysis by loading a CSV into a Pandas DataFrame. It proceeds to inspect the structure (.shape, .head()) and performs basic cleaning (column renaming). The core function is data quality assessment, utilizing .info() and .isnull().sum() to identify data types and quantify missing values.
Finally, .describe() and .nunique() provide a statistical summary and feature analysis, establishing a full blueprint for subsequent data preprocessing**.**