**✅ 1. Loading Libraries**

python

CopyEdit

import pandas as pd

import matplotlib.pyplot as plt

**Problem Step**: “Import all required Python Libraries.”

**Definition**:

* import pandas as pd brings in **pandas**, a library used to work with tabular data using **DataFrames**.
* import matplotlib.pyplot as plt brings in **matplotlib**, a plotting library for creating visualizations like **box plots, bar charts**, etc.

**Why**:  
We need **pandas** for data wrangling (loading, cleaning, and manipulating data), and **matplotlib** for plotting data during analysis (e.g., for normalization step with box plots).

**✅ 2. Loading the Dataset**

python

CopyEdit

df = pd.read\_csv('datasets/AcademicPerformance\_New.csv')

**Problem Step**: “Load the dataset into pandas DataFrame.”

**Definition**:

* pd.read\_csv() reads a CSV (Comma-Separated Values) file and converts it into a **DataFrame**.

**Why**:  
We need to **load the data** into memory so we can begin analyzing and processing it.

**✅ 3. Viewing Top Rows**

python

CopyEdit

df.head()

**Problem Step**: “Preview the dataset.”

**Definition**:

* df.head() shows the **first 5 rows** of the dataset.

**Why**:  
This gives a **quick overview** of the structure of the data, column names, and example entries.

**✅ 4. Displaying Column Info**

python

CopyEdit

df.info()

**Problem Step**: “Provide variable descriptions and check data types.”

**Definition**:

* df.info() displays **column names, non-null counts**, and **data types**.

**Why**:  
We use this to:

* **Check if there are missing values**,
* Understand which variables are **numeric**, **categorical**, etc.

**✅ 5. Checking Missing Values**

python

CopyEdit

df.isnull().sum()

**Problem Step**: “Find the missing values in each column.”

**Definition**:

* df.isnull() returns True for missing values.
* sum() counts how many True values (i.e., missing) each column has.

**Why**:  
To identify which columns have **incomplete data** and need to be cleaned or filled.

**✅ 6. Dropping Rows with Missing Roll No**

python

CopyEdit

df.dropna(subset=['Roll No'], inplace=True)

**Problem Step**: “Data Preprocessing – Handle missing values.”

**Definition**:

* dropna() removes rows that contain missing values.
* subset=['Roll No'] ensures only rows where **‘Roll No’ is missing** are dropped.
* inplace=True makes the change permanent to df.

**Why**:  
A missing **Roll No** is likely a critical ID, so such records are removed to ensure **data integrity**.

**✅ 7. Filling Missing 'WT' Values**

python

CopyEdit

df['WT'].fillna(df['WT'].median(), inplace=True)

**Problem Step**: “Data Cleaning – Handle missing numerical values.”

**Definition**:

* fillna() fills missing values.
* df['WT'].median() calculates the **median** of the 'WT' column.
* inplace=True saves changes in the original DataFrame.

**Why**:  
Filling missing values with the **median** prevents distortion due to outliers (compared to using the mean).

**✅ 8. Checking Updated Info**

python

CopyEdit

df.info()

**Problem Step**: “Confirm data cleaning and type info.”

**Definition**:

* df.info() again displays updated data types and non-null values after cleaning.

**Why**:  
We re-check this to **verify** that:

* Missing values were successfully handled.
* No critical data (like Roll No or WT) is missing anymore.

**✅ 9. Checking DataFrame Dimensions**

python

CopyEdit

df.shape

**Problem Step**: “Check the dimensions of the data frame.”

**Definition**:

* df.shape returns a **tuple (rows, columns)**.

**Why**:  
This tells us how many **records** and **features** we have after cleaning the dataset.

**✅ 10. Checking Column Data Types**

python

CopyEdit

df.dtypes

**Problem Step**: “Summarize types of variables.”

**Definition**:

* df.dtypes shows the **data type of each column** (e.g., int64, float64, object).

**Why**:  
This helps in understanding whether variables are:

* **Categorical** (object)
* **Numerical** (int, float)
* **Need type conversion** (like converting object to numeric)

**✅ 11. Converting 'Roll No' to String**

python

CopyEdit

df['Roll No'] = df['Roll No'].astype(str)

**Problem Step**: “Format data types correctly.”

**Definition**:

* astype(str) converts a column’s data type to **string (text)**.

**Why**:  
Since **‘Roll No’ is an identifier**, not a numeric value used in math, we store it as a **string**.

**✅ 12. Confirming Data Types Again**

python

CopyEdit

df.dtypes

**Problem Step**: “Verify type conversion.”

**Definition**:

* Re-check the column types using dtypes.

**Why**:  
Ensure the **conversion was successful** — for example, ‘Roll No’ should now show as object (pandas type for string).

**✅ 13. Statistical Summary**

python

CopyEdit

df.describe()

**Problem Step**: “Display statistical information.”

**Definition**:

* describe() provides **mean, std, min, max, and quartiles** for numeric columns.

**Why**:  
It helps understand the **distribution** of numerical features and check for possible outliers or skewed data.

**✅ 14. Z-Score Normalization**

python

CopyEdit

df['TOC\_Z'] = (df['TOC'] - df['TOC'].mean()) / df['TOC'].std()

**Problem Step**: “Perform Z-score normalization.”

**Definition**:

* This formula standardizes the data to **Z-scores**, which have a **mean = 0** and **standard deviation = 1**.

**Why**:  
Normalization is needed to **bring values to a similar scale**, especially for comparisons or input to machine learning models.

**✅ 15. Display Z-Score Values**

python

CopyEdit

df[['TOC', 'TOC\_Z']]

**Problem Step**: “Compare original and normalized values.”

**Definition**:

* Displays both the **original 'TOC' scores** and **Z-score normalized 'TOC\_Z'** side by side.

**Why**:  
Lets us visually compare the **effect of normalization**.

**✅ 16. Boxplot of TOC**

python

CopyEdit

plt.boxplot(df['TOC'])

plt.title("Boxplot of TOC")

plt.show()

**Problem Step**: “Visualize data distribution.”

**Definition**:

* plt.boxplot() creates a **boxplot** to show the spread and outliers.
* plt.title() sets the title.
* plt.show() displays the plot.

**Why**:  
Boxplots are used to detect **outliers** and understand the **distribution** of numeric variables.

**✅ 17. Convert Categorical to Numerical (One-Hot Encoding)**

python

CopyEdit

df\_encoded = pd.get\_dummies(df, columns=['Gender', 'Division'], drop\_first=True)

**Problem Step**: “Turn categorical variables into quantitative variables.”

**Definition**:

* pd.get\_dummies() converts **categorical variables** into **dummy/indicator (0/1) columns**.
* columns=['Gender', 'Division'] specifies which columns to convert.
* drop\_first=True avoids dummy variable trap by dropping the first category.

**Why**:  
Machine learning algorithms require **numerical input** — so we encode strings like ‘Male’/’Female’ into numbers (e.g., Female = 0, Male = 1).

"A boxplot of the Average column helps us visualize the distribution of student average scores. The box shows the middle 50% of scores, the line inside shows the median, and any dots outside show outliers. This helps in detecting anomalies or skewed data."