**Data Wrangling**, also known as **data munging**, is the process of cleaning, transforming, and organizing raw data into a structured and usable format for analysis. It involves handling missing data, correcting errors, converting data types, normalizing values, and encoding categorical variables to prepare the dataset for further statistical analysis or machine learning tasks.

**Key Steps in Data Wrangling:**

1. **Importing Data:** Loading data from sources such as CSV files, databases, or APIs.
2. **Handling Missing Values:** Detecting and dealing with null or incomplete data entries.
3. **Data Cleaning:** Removing duplicates, correcting errors, and standardizing formats.
4. **Transforming Data:** Changing the structure or format, like converting data types or reshaping tables.
5. **Normalization and Scaling:** Adjusting values to a common scale without distorting differences.
6. **Encoding Categorical Variables:** Converting text labels into numerical form (e.g., one-hot encoding).
7. **Exporting the Cleaned Data:** Saving the processed data for further use.

**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 (data‐frames).
  + import matplotlib.pyplot as plt brings in matplotlib for plotting.
* **Why**: We need pandas for data wrangling and matplotlib for later visualization.

**2. Reading the CSV**

python

CopyEdit

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

* **Problem Step**: “Load the Dataset into pandas DataFrame.”
* **Definition**: pd.read\_csv() reads a CSV file from disk into a pandas DataFrame named df.
* **Why**: First you must load the raw data into memory.

**3. Previewing the Data**

python

CopyEdit

df.head()

* **Problem Step**: “Provide variable descriptions… preview the data.”
* **Definition**: Shows the first 5 rows of df.
* **Why**: Quick sanity check: do the columns look right? Are string columns actually strings?

**4. Finding Missing Values**

python

CopyEdit

df.isnull().sum()

* **Problem Step**: “Find missing values column-wise.”
* **Definition**:
  + df.isnull() returns a same-sized Boolean frame (True where missing).
  + .sum() tallies each column’s missing count.
* **Why**: To know which columns need imputation or removal.

**5. Statistical Summary**

python

CopyEdit

df.describe()

* **Problem Step**: “Display statistical information.”
* **Definition**:
  + For numeric columns, returns count, mean, std, min, 25%, 50%, 75%, max.
* **Why**: Helps you understand distributions and spot outliers.

**6. DataFrame Info**

python

CopyEdit

df.info()

* **Problem Step**: “Provide variable descriptions… Types of variables etc.”
* **Definition**:
  + Lists each column, its data type, and non-null count.
* **Why**: Tells you which columns are int64, float64, object (string), etc., and if any have nulls.

**7. Data Types**

python

CopyEdit

df.dtypes

* **Problem Step**: “Summarize the types of variables… check data types.”
* **Definition**: A Series showing each column’s dtype.
* **Why**: You’ll need this to decide which dtype conversions to apply.

**8. Number of Dimensions**

python

CopyEdit

df.ndim

* **Problem Step**: “Check the dimensions of the DataFrame.”
* **Definition**: Returns the count of axes (always 2 for a DataFrame).
* **Why**: Confirms you have a 2-D table, not accidentally a Series.

**9. Shape (Rows × Columns)**

python

CopyEdit

df.shape

* **Problem Step**: “Check the dimensions of the DataFrame.”
* **Definition**: Returns a tuple (number\_of\_rows, number\_of\_columns).
* **Why**: You know the dataset size: e.g., 100 rows × 10 columns.

**10. Converting JoiningYear to Date**

python

CopyEdit

pd.to\_datetime(df['JoiningYear'], format='%Y')

* **Problem Step**: “If variables are not in the correct data type, apply proper type conversions.”
* **Definition**: Parses the integer year (e.g. 2015) into a pandas Datetime object representing January 1 of that year.
* **Why**: So you can treat joining year as a true date (e.g., to extract month/year functions later).

**11. Converting Data Types (Example)**

python

CopyEdit

df['JoiningYear'] = pd.to\_datetime(df['JoiningYear'], format='%Y')

* **Problem Step**: “Apply proper type conversions.”
* **Definition**: Assigns the converted datetime back into the DataFrame.
* **Why**: Stores it in the right format for downstream analysis.

**12. Checking Unique Values (Categoricals)**

python

CopyEdit

df['Education'].unique(), df['City'].unique(), df['Gender'].unique(), df['EverBenched'].unique()

* **Problem Step**: “Turn categorical variables into quantitative variables.” (First inspect them.)
* **Definition**: .unique() lists all distinct entries in each column.
* **Why**: To see which categories exist before encoding.

**13. Mapping Gender to Numeric**

python

CopyEdit

mapping = {'Male':0, 'Female':1}

df['Gender'] = df['Gender'].map(mapping)

* **Problem Step**: “Turn categorical variables into quantitative variables.”
* **Definition**:
  + Creates a Python **dictionary** (mapping) where each key is the original category and the value is the new numeric code.
  + .map(mapping) replaces each value in the column per that dictionary.
* **Why**: Converts “Male”/“Female” into 0/1 so ML models can work with them.

**14. Final Preview**

python

CopyEdit

df.head()

* **Problem Step**: “Verify all transformations.”
* **Definition**: Again shows the first 5 rows—now including the converted datetime and numeric gender.
* **Why**: Always double-check that your conversions did what you expected.

**Putting It All Together**

| **Step** | **Function** | **Why We Use It** |
| --- | --- | --- |
| Load data | pd.read\_csv() | Read CSV into memory |
| Preview | df.head() | Quick look at columns & values |
| Missing values | df.isnull().sum() | Identify columns needing cleaning |
| Summary statistics | df.describe() | Understand numeric distributions |
| DataFrame structure | df.info(), df.dtypes | See data types & non-null counts |
| Dimensions | df.ndim, df.shape | Confirm rows × columns |
| Type conversion | pd.to\_datetime(), .astype | Put columns into correct formats (dates, floats, etc.) |
| Inspect categories | .unique() | Know which categories to encode |
| Encode categoricals | .map() or pd.get\_dummies() | Convert text labels into numbers for ML |