

## Libraries Used

### 1. pandas

The main library for reading and writing **Excel**, **CSV**, and **JSON**.

Functions used:

```
pd.read_excel()
```

```
pd.read_csv()
```

```
pd.read_json()
```

```
df.to_excel()
```

```
df.to_csv()
```

```
df.to_json()
```

### 2. openpyxl

Used internally by pandas to **read and write Excel (.xlsx)** files.

```
from openpyxl import load_workbook
```

## Excel (XLS / XLSX – Microsoft Excel Files)

1. **Feature-rich spreadsheet format:** Excel supports formulas, charts, cell formatting, pivot tables, and many built-in tools that make it powerful for both basic and advanced data analysis.
2. **Ideal for structured data management:** Excel organizes information in rows and columns, making it easy to clean, filter, sort, and analyze data without programming.
3. **Widely supported in Python:** Libraries like pandas, openpyxl, and xlrd allow Python to read, write, modify, and automate Excel files efficiently — making it perfect for data processing workflows.

## CSV (Comma-Separated Values)

1. **Simple and lightweight format:** CSV stores data in a plain-text table structure where each value is separated by a comma, making it easy to read and edit.

2. **Ideal for large datasets:** Since the format is minimal, CSV files load faster and take up less space compared to other structured formats.

### **JSON (JavaScript Object Notation)**

1. **Structured and human-readable:** JSON represents data using key–value pairs, making it easy to understand and work with in modern applications.
2. **Widely used in web APIs:** Its lightweight structure and flexibility make JSON the standard format for sending and receiving data over the web.
3. **Supports complex data:** JSON can store nested objects, lists, and various data types, allowing developers to represent detailed and hierarchical information.