TP 1 Lab: CSV Sales Analysis (Standard Library) + Clean Project Setup

---

Goal

Build a small, reproducible Python project that:

- loads a sales CSV file (including a “dirty” version),
- computes basic KPIs,
- exports one result file.

Constraints

- Use a **virtual environment (venv)**.
- Use **standard library only** for the analysis: `csv` (no pandas in this lab).
- Convert types at load time:
  - `quantity` → `int`
  - `unit_price` → `float` (bonus: accept comma decimals like `2,50`)
- Invalid rows: **skip with a warning** (minimum expected behavior).
- Organize code into modules under `src/`.

Required Project Structure

```
session1/
  main.py
  requirements.txt
  README.md
  src/
    io_utils.py
    stats_utils.py
  data/
    sales_clean.csv
    sales_dirty.csv
```

out/

Data Files (copy/paste)

Create these two files:

`data/sales_clean.csv`

```
date,product,quantity,unit_price
2026-01-01,coffee,2,1.80
2026-01-01,tea,1,2.10
2026-01-02,coffee,1,1.80
2026-01-02,chocolate,3,2.50
2026-01-03,tea,2,2.10
2026-01-03,coffee,4,1.80
```

`data/sales_dirty.csv`

```
date,product,quantity,unit_price
2026-01-01,coffee,2,1.80
2026-01-01,tea,,2.10
2026-01-02,coffee,1,1.80
2026-01-02,chocolate,3,2,50
2026-01-03, tea ,2,2.10
2026-01-03,coffee,four,1.80
```

Tasks

Task A — Robust Loader (I/O) (20 min)

In `src/io_utils.py`, implement:

Function

- `load_sales(path: str) -> list[dict]`

Requirements

- Load the CSV using `csv.DictReader`.
- Clean `product` using `.strip()`.

- Convert:
  - `quantity` to `int`
  - `unit_price` to `float`
- Missing/invalid values:
  - minimum: **skip the row** and print:
    - `WARNING line X skipped: <reason>`
- Bonus:
  - accept comma decimals for `unit_price` (convert `"2,50"` → `2.50`)

Task B — KPI Computations (35 min)

In `src/stats_utils.py`, implement:

- `total_revenue(rows: list[dict]) -> float`
- `revenue_by_product(rows: list[dict]) -> dict[str, float]`
- `top_product(revenue_dict: dict[str, float]) -> tuple[str, float]`
- `export_revenue_by_product(revenue_dict: dict[str, float], path: str) -> None`

Definitions

- Row revenue = `quantity * unit_price`
- Total revenue = sum of row revenues
- Revenue by product = sum of row revenues per product
- Top product = product with the highest revenue

Export format

- CSV columns: `product,revenue`
- Output path: `out/revenue_by_product.csv`

Task C — Orchestrate (Main) (20 min)

In `main.py`:

- Parse `--input` (required) using `argparse`
- Call your loader + KPI functions
- Print:
  - total revenue
  - revenue by product

    - top product
- Export `out/revenue_by_product.csv`

Example run:

python main.py --input data/sales_clean.csv
python main.py --input data/sales_dirty.csv

Deliverables

- A working `session1/` folder matching the required structure
- `requirements.txt` (created using `pip freeze > requirements.txt`)
- `README.md` (use the template below)
- Generated file: `out/revenue_by_product.csv`

README Template (copy/paste into `README.md`)

Session 1 — CSV Sales Analysis (Python)

Setup
```bash
python -m venv venv
# activate venv (OS-specific)
pip install -r requirements.txt

# Run

python main.py --input data/sales_clean.csv
python main.py --input data/sales_dirty.csv

# Output

- Prints total revenue, revenue by product, top product
- Exports `out/revenue_by_product.csv`
```