TP2 : Refactor the CSV Sales Project into an OOP API (SalesDataset)

Goal

Refactor your Session 1 project (functions + modules) into a clean **object-oriented design**:

- a class `SalesDataset` that holds the dataset state (`path`, `rows`)
- methods for loading data and computing KPIs
- export results to a CSV file

Constraints

- Use a **virtual environment (venv)**.
- Use the standard **library only** for the analysis: `csv` (no pandas in this lab).
- Implement OOP basics: `class`, `__init__`, attributes, methods.
- Invalid rows: **skip with a warning** (same rule as Session 1).
- Keep the project structure clean.

Required Project Structure

```
session2/
  main.py
  requirements.txt
  README.md
  src/
    sales_dataset.py
  data/
    sales_clean.csv
    sales_dirty.csv
  out/
```

Data Files (copy/paste)

`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 — Create the Class + Loading Method (25 min)

In `src/sales_dataset.py`, implement:

Class

- `class SalesDataset`

Required

- `__init__(self, path: str)`
  - stores `self.path`
  - initializes `self.rows` as an empty list
- `load(self) -> None`
  - reads the CSV using `csv.DictReader`
  - cleans `product` with `.strip()`
  - converts:
    - `quantity` → `int`
    - `unit_price` → `float` (bonus: accept `2,50` → `2.50`)

    - invalid row → print warning and skip

Task B — Implement KPI Methods (35 min)

Add methods to `SalesDataset`:

- `total_revenue(self) -> float`
- `revenue_by_product(self) -> dict[str, float]`
- `top_product(self) -> tuple[str, float]`

Rules:

- If `self.rows` is empty, return safe values:
    - total revenue = `0.0`
    - revenue by product = `{}`
    - top product = `("", 0.0)`

Task C — Export Method + CLI Orchestration (25 min)

Add in the class:

- `export_revenue_by_product(self, path: str) -> None`
    - exports a CSV with columns: `product,revenue`

In `main.py`:

- parse `--input` (required) using `argparse`
- create and use the dataset object:
    - `ds = SalesDataset(args.input)`
    - `ds.load()`
    - print total revenue, revenue by product, top product
    - export to `out/revenue_by_product.csv`

Deliverables

- A working `session2/` folder matching the required structure
- `requirements.txt` created with `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 2 — OOP SalesDataset (CSV analysis)

## 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`
```