Outils

Environnement, installation, IDE, versionning, qualité, DQ, exécution

Programme Data Engineer

Public intermédiaire — Travaux pratiques

23 octobre 2025

Pile logicielle du TP

- **Langage** : Python 3.10+ (numpy, pandas, matplotlib).
- **IDE/Notebooks** : VS Code (extensions Python, Jupyter) ou JupyterLab.
- **Stockage local** : CSV/Parquet ; option **DuckDB** embarqué pour SQL.
- **Versionning** : Git (Github/GitLab), branches feature.
- **Qualité** : black, isort, ruff, pre-commit ; tests `pytest`.
- **DQ (option)** : Great Expectations ou Soda.

Installation rapide (venv/pip) et alternative Conda

Option A — venv + pip

```
python3 -m venv .venv
source .venv/bin/activate    # Windows: .venv\\Scripts\\activate
python -m pip install --upgrade pip
pip install numpy pandas matplotlib duckdb pytest black isort ruff
    pre-commit
```

Option B — Conda/Miniconda

```
conda create -n dw_tp python=3.11 -y
conda activate dw_tp
pip install numpy pandas matplotlib duckdb pytest black isort ruff
    pre-commit
```

Structure de projet recommandée

```
mini_dw_tp/
  data/
    raw/          # fichiers sources (CSV)
    staging/      # nettoyages interm diaires
    dim/          # dimensions persist es
    fact/         # faits persist s
  notebooks/
  src/
    etl/
      __init__.py
      extract.py     # lecture CSV/Parquet
      transform.py   # nettoyage, assemblage
      load.py        # persist CSV/Parquet/DuckDB
    utils/
      io.py, dq.py, logging_conf.py
  tests/
  Makefile
  requirements.txt (ou pyproject.toml)
```

Makefile : commandes pratiques

```
.PHONY: setup format lint test run
setup:
  python -m pip install -r requirements.txt
  pre-commit install

format:
  black src notebooks
  isort src notebooks

lint:
  ruff check src notebooks
  ruff format --check
  ruff fix --select I --unsafe-fixes
  ruff check --select F401  # imports inutiles
  ruff check --select E --select W
  ruff check --select I
  ruff check --statistics
  ruff check --show-source
```

Git : bonnes pratiques & .gitignore

Flux : main (protégée) → *branches* `feature/..` → *PR & review*.

DuckDB (option) : moteur SQL embarqué

```
import duckdb
con = duckdb.connect("mini_dw.duckdb")
con.execute("CREATE SCHEMA IF NOT EXISTS dw")
con.execute("CREATE TABLE IF NOT EXISTS dw.dim_product AS SELECT *
    FROM read_csv_auto('data/dim/dim_product.csv')")
# Requeter le CA par mois
con.sql(
    """
    SELECT strftime(order_date, '%Y-%m') AS ym, SUM(amount) AS
        revenue
    FROM read_csv_auto('data/fact/fact_sales.csv')
    GROUP BY ym ORDER BY ym
    """
).df()
```

Tests (pytest) & Data Quality

pytest — exemple minimal

```python
# tests/test_amount.py
import pandas as pd

def test_amount_non_negative():
    df = pd.read_csv('data/fact/fact_sales.csv')
    assert (df['amount'] >= 0).all()
```

Great Expectations — CLI (option)

```
# initialisation dans le repo
great_expectations init
# cr er une suite et valider un fichier
great_expectations suite new
great_expectations checkpoint run my_checkpoint
```

Formatage, lint & hooks pre-commit

```yaml
# .pre-commit-config.yaml (extrait)
repos:
  - repo: https://github.com/psf/black
    rev: 24.8.0
    hooks: [{id: black}]
  - repo: https://github.com/charliermarsh/ruff-pre-commit
    rev: v0.5.5
    hooks: [{id: ruff}]
  - repo: https://github.com/PyCQA/isort
    rev: 5.13.2
    hooks: [{id: isort}]
```

Logs & configuration (.env)

```python
# src/utils/logging_conf.py
import logging, os
LOG_LEVEL = os.getenv("LOG_LEVEL", "INFO")
logging.basicConfig(
    level=LOG_LEVEL,
    format="%(asctime)s %(levelname)s [%(name)s] %(message)s",
)
logger = logging.getLogger("dw_tp")
```

```
# .env (exemple)
LOG_LEVEL=INFO
DATA_DIR=./data
```

Exécution : scripts, notebooks, VS Code

- **Scripts** : `python -m src.etl.load` (réutilisable, testable).
- **Notebooks** : exploration/validation (sauver en `.py` pour versionning).
- **VS Code** : tasks, debug config, intégration Git, extensions Python/Jupyter.
- **Make** : `make setup | make run | make test | make lint`.

Schéma d'outillage du TP