# Documentation technique — \*\*test\_pipline\*\*

Version : 1.0

Dernière mise à jour : 2025-08-24

---

## 1) Objet du projet

\*\*test\_pipline\*\* est une pipeline ETL en Python qui :

* lit les données \*\*médicaments\*\* (`drugs.csv`), \*\*publications\*\* (`pubmed.csv` + `pubmed.json`) et \*\*essais cliniques\*\* (`clinical\_trials.csv`) ;
* détecte les \*\*mentions\*\* de médicaments dans les \*\*titres\*\* ;
* produit un \*\*JSON final groupé par code ATC\*\* ;
* fournit un \*\*outil ad-hoc\*\* pour identifier le \*\*journal\*\* qui mentionne le plus de \*\*médicaments distincts\*\*.

Le code est modulaire (I/O, nettoyage, matching, agrégation), prêt à être \*\*orchestré\*\* (ex. Airflow).

---

## 2) Architecture

test\_pipline/  
├─ src/test\_pipline/  
│ ├─ \_\_init\_\_.py  
│ ├─ config.py # gestion des chemins et paramètres  
│ ├─ io.py # lecture des fichiers (CSV/JSON tolérant)  
│ ├─ clean.py # nettoyage textes + dates ISO + helpers ID  
│ ├─ match.py # construction regex + détection de mentions  
│ ├─ aggregate.py # construction du JSON final par ATC  
│ └─ pipeline.py # orchestration (end-to-end)  
├─ tools/top\_journal.py # ad-hoc (journal le plus couvrant)  
├─ Data/ # entrées  
├─ outputs/ # sorties  
└─ run.py # entrée CLI

Flux :

1) `io.read\_\*` → 2) `clean.add\_iso\_date` → 3) `match.find\_mentions` → 4) `aggregate.build\_by\_atc` → 5) JSON

---

## 3) Entrées

### `drugs.csv`

| atccode | drug |

|--------:|------|

| A04AD | DIPHENHYDRAMINE |

### `pubmed.csv` / `pubmed.json`

Schéma : `id, title, journal, date` (JSON lu \*\*tolérant\*\*).

### `clinical\_trials.csv`

Schéma : `id, scientific\_title→title, journal, date`.

---

## 4) Règles de détection

* Une \*\*mention\*\* = le nom du médicament trouvé comme \*\*mot entier\*\* dans `title` (insensible à la casse).
* Une ligne de mention contient : `drug\_atccode, drug\_name, source, pub\_id, title, journal, date`.

---

## 5) Sortie JSON (par ATC)

Fichier : `outputs/drug\_publications\_by\_atc.json`

{  
 "A04AD": {  
 "drug": "DIPHENHYDRAMINE",  
 "atccode": "A04AD",  
 "pubmed": [  
 {"id": 1, "title": "...", "date": "YYYY-MM-DD", "journal": "..."}  
 ],  
 "clinical\_trials": [  
 {"id": "NCT001", "title": "...", "date": "YYYY-MM-DD", "journal": "..."}  
 ],  
 "journals": [  
 {"journal": "Nom", "first\_date": "YYYY-MM-DD", "last\_date": "YYYY-MM-DD", "n\_pubs": 3}  
 ]  
 }  
}

Notes : dates en \*\*ISO `YYYY-MM-DD`\*\* ; pas de `NaN` en JSON (valeurs manquantes → `null`).

---

## 6) Exécution

Prérequis : Python 3.9+, `pip install -r requirements.txt`

Commande :

python run.py --data-dir Data --out-dir outputs --dayfirst

---

## 7) Ad-hoc : journal le plus couvrant

python tools/top\_journal.py \  
 --input outputs/drug\_publications\_by\_atc.json \  
 --export-csv outputs/journal\_drug\_coverage.csv

---

## 8) Airflow (extrait)

from pathlib import Path; import sys, pendulum  
from airflow import DAG  
from airflow.operators.python import PythonOperator  
  
def airflow\_run\_pipeline(\*\*\_):  
 dag\_dir = Path(\_\_file\_\_).resolve().parent  
 project\_dir = dag\_dir / "test\_pipline"  
 sys.path.append(str(project\_dir / "src"))  
 from test\_pipline.config import Config  
 from test\_pipline.pipeline import run\_pipeline  
 cfg = Config(data\_dir=project\_dir/"Data", out\_dir=project\_dir/"outputs")  
 run\_pipeline(cfg)  
  
with DAG("test\_pipline\_etl", start\_date=pendulum.datetime(2025,1,1,tz="Europe/Paris"),  
 schedule="@daily", catchup=False) as dag:  
 PythonOperator(task\_id="run\_test\_pipline", python\_callable=airflow\_run\_pipeline)

---

## 9) Qualité

* Encodage UTF-8 ; normalisation texte (anti \*mojibake\*).
* Dates robustes ; JSON strict (sans NaN).
* Code modulaire, simple à tester.