

Data Sanitizer (POO) — Notice d'utilisation

Dernière mise à jour : 2025-08-26

1. Objet du projet

Data Sanitizer fournit une chaîne fiable pour importer, normaliser et écrire des historiques de prix d'actions dans PostgreSQL. La refonte POO vise : - **stabilité/fiabilité** des données (flux idempotent, upserts, vues de compatibilité); - **intégration simple** par d'autres projets (ex. *MarketAdvisor*) via une API CLI et des conventions de schéma stables.

2. Prérequis

- **Python** 3.12+
- **PostgreSQL** 14+ (local ou distant)
- Accès réseau vers *Yahoo Finance* (utilisé par `yfinance`).

3. Installation

```
python -m venv .venv
source .venv/bin/activate
pip install -e .
```

4. Configuration

Créer un fichier `.env` à la racine :

```
DATABASE_URL=postgresql://pea_user:pea_password@127.0.0.1:5432/pea_db
DS_PRICE_READ_VIEW=v_prices_compat
DS_PRICE_WRITE_TABLE=equity_prices
DS_PRICE_DATE_COL=price_date
LOG_LEVEL=INFO
REQUEST_PAUSE_S=0.6
YF_TIMEOUT_S=10
```

Le chargement de `.env` est **automatique** (aucun `export` requis).

4.1 Objets de base attendus côté DB

- Table **equities** : univers des titres (incluant `isin`, `symbol`, `is_active`, `is_valid`, etc.).
- Table **equity_prices** : historique des prix (clé : `(isin, symbol, price_date)`).
- Vue **v_prices_compat** : vue de lecture (compatibilité).

Exemple SQL idempotent pour la vue :

```

DROP VIEW IF EXISTS public.v_prices_compat;
CREATE VIEW public.v_prices_compat (
    isin, symbol, price_date, open_price, close_price, high_price, low_price,
    volume, adj_close
) AS
SELECT isin, symbol, price_date, open_price, close_price, high_price,
low_price, volume, adj_close
FROM public.equity_prices;

```

5. Utilisation — CLI

Le binaire `data-sanitizer` charge `.env` automatiquement et expose une **commande unique**.

5.1 Vérifier la connexion DB

```
data-sanitizer --check-db
```

Affiche un diagnostic   (hôte, base, user).

5.2 Mettre à jour les prix

```

# Lecture seule (aucun write)
data-sanitizer --dry-run --since 2024-01-01 -o SAN -o AIR.PA

# Écriture effective (par défaut), alias explicite possible
data-sanitizer --write --limit 10 --sleep 0.3


```

Options : - `--since YYYY-MM-DD` : date de départ (sinon reprend à la dernière date connue). - `--limit N` : nombre max de titres à traiter. - `--only / -o` : filtrer par `ISIN` / `SYMBOL` (répétable). - `--sleep S` : pause entre tickers (throttle Yahoo si besoin). - `--dry-run` : ne rien écrire en base. - `--write` : alias explicite pour écrire (optionnel, l'écriture est le mode par défaut si `--dry-run` est absent).

5.3 Journalisation & progression

Le service affiche des lignes de progression :

```

[1/10] AIR.PA (ISIN ...) ... résolution du ticker
[1/10] AIR.PA (ISIN ...) ... téléchargement (AIR.PA, since=2024-01-01)
[1/10] AIR.PA (ISIN ...) →  écriture : 254 barres (total=6200, 1y=220)

```

Ajuster la verbosité via `LOG_LEVEL` dans `.env`.

6. Intégration projet tiers (ex. MarketAdvisor)

Deux approches :

6.1 Intégration SQL directe (recommandée)

Lire depuis `v_prices_compat` (contrat stable) :

```
SELECT * FROM v_prices_compat WHERE symbol = 'AIR.PA' AND price_date >=
'2024-01-01';
```

Avantages : couplage minimal, pas de dépendance Python, compatibilité ascendante.

6.2 Intégration Python (en dépendance)

Dans le projet client :

```
# pyproject.toml
[project]
dependencies = [
    "data-sanitizer @ git+ssh://git@github.com/mnorodine/
data_sanitizer_poo.git#egg=data-sanitizer",
]
```

Et appel en sous-process ou module :

```
# sous-process
python -m venv .venv && source .venv/bin/activate
pip install "data-sanitizer @ git+ssh://git@github.com/mnorodine/
data_sanitizer_poo.git#egg=data-sanitizer"
DATA_SANITIZER_OPTS="--write --since 2025-01-01" data-sanitizer
```

Pour un couplage plus fort, exposer ultérieurement un mini-API Python (ports) est possible.

7. Modèle de données (résumé)

equity_prices - `isin` (text), `symbol` (text), `price_date` (date) - `open_price`, `high_price`, `low_price`, `close_price`, `adj_close` (numeric) - `volume` (bigint) - Index : (`isin`, `price_date`), (`symbol`, `price_date`); clé unique (`isin`, `symbol`, `price_date`).

equities - `isin` (PK), `symbol`, `is_active`, `is_valid`, etc. (attributs de suivi et qualité)

v_prices_compat - Projection lecture-seule identique à `equity_prices` pour compatibilité.

8. Bonnes pratiques d'exploitation

- Lancer d'abord en `--dry-run` sur un échantillon (`--limit 5`).
- Programmer un job périodique (ex. cron / Actions) avec `--sleep 0.3` pour limiter le throttling Yahoo.
- Surveiller la volumétrie : index présents, vacuum/analyse réguliers.

9. Dépannage

- **Pas de logs / très lent** : ajouter `--limit 5 --sleep 0.3`. Yahoo peut limiter.
- **Connexion DB KO** : vérifier `DATABASE_URL` dans `.env` ; tester `data-sanitizer --check-db`.
- **Aucun prix** : certains tickers Yahoo peuvent être déslistés ou renommés. Vérifier la résolution de ticker et le mapping `equities`.

10. CI & Qualité

- **GitHub Actions** : lint (`ruff`) + tests (`pytest`).
- **Ruff** : respect PEP8, imports, etc.
- **Tests** : smoke tests fournis, à compléter (mocks provider/DB) selon besoins.

11. Roadmap courte

- Option `--progress bar` (barre compacte)
- Endpoint Python (service) minimal pour intégration directe MarketAdvisor
- Stubs/implémentations pour `recompute_counts()` et `update_bounds()` selon usage réel