

TP3 – Data Lake & Data Warehouse (Kafka, KSQLDB, Python, SQLite/MySQL)

1. Design du Data Lake

Élément	Description
Emplacement	Le Data Lake est un dossier local nommé data_lake/ sur la machine.
Structure	Les données provenant de Kafka/KSQLDB sont stockées dans des sous-dossiers par topic et date : data_lake/<topic>/<YYYY-MM-DD>/<topic>_<YYYY-MM-DD>.jsonl
Partitionnement	Partition par date (ou version) pour faciliter la traçabilité et les chargements incrémentaux.
Ajout de nouveaux feeds	Lorsqu'un nouveau flux Kafka est ajouté : 1) Déclaration dans config/settings.py 2) Le consumer Kafka crée automatiquement le dossier ; 3) Le pipeline est réutilisé sans modification du code.
Stockage Streams & Tables	Chaque stream et table est stocké sous forme de fichiers JSONL distincts.
Mode de stockage	Streams → append ; Tables agrégées → overwrite ; Données statiques → ignore.

Structure

```text

```
data_lake/
 ├── config/
 | └── settings.py
 ├── consumers/
 | ├── kafka_to_datalake.py
 | └── fake_kafka_producer.py
 ├── jobs/
 | └── sqlite_loader.py
 ├── pipeline/
 | ├── TRANSACTIONS_SECURE/
 | ├── TRANSACTIONS_USD/
 | ├── TRANSACTIONS_BLACKLISTED/
 | ├── USER_SPENDING_BY_TYPE/
 | └── SPEND_LAST_FIVE_MIN_BY_TYPE/
 └── data_warehouse.db
````
```

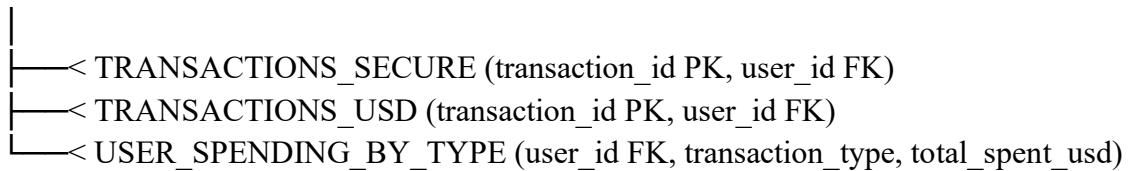
2. Design du Data Warehouse (MySQL / SQLite)

Toutes les tables de KSQLDB sont répliquées dans le Data Warehouse via un job Python (sqlite_loader.py).

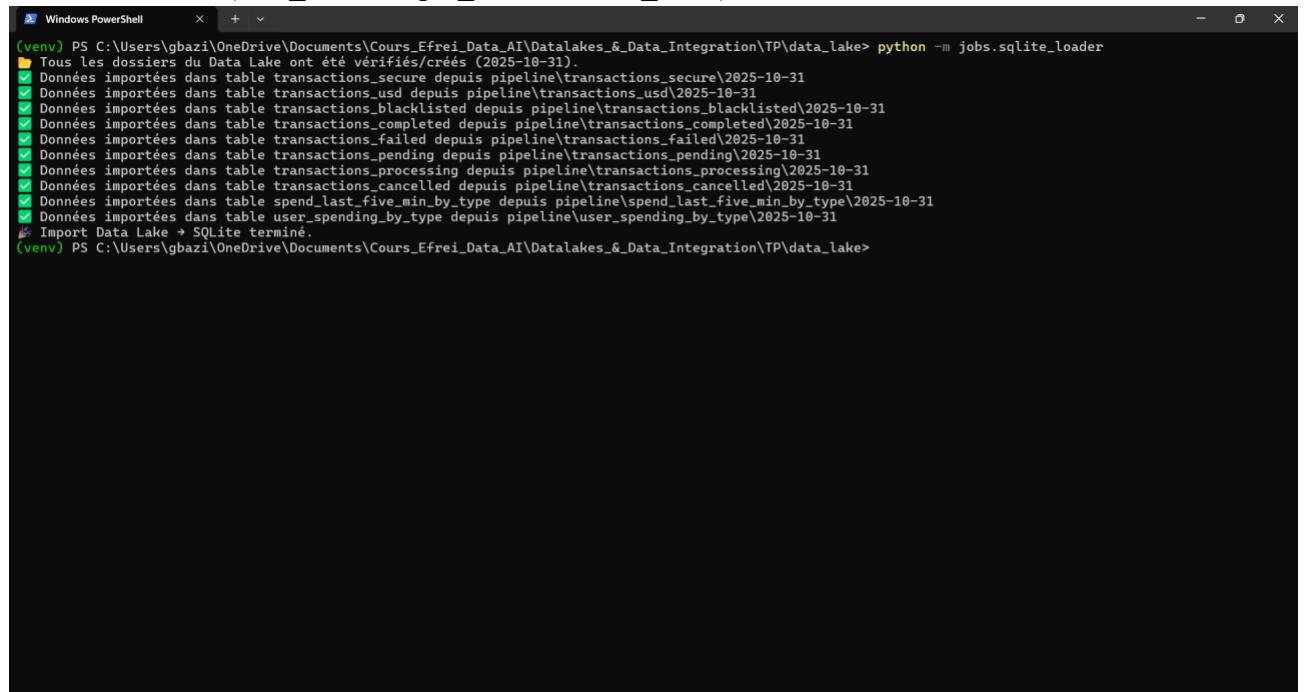
| Table | Clé primaire | Clés étrangères | Description |
|------------------------------------|-----------------------------|-----------------|-------------------------------------|
| TRANSACTIONS_SECURE | transaction_id | user_id | Transactions brutes anonymisées |
| TRANSACTIONS_USD | transaction_id | user_id | Transactions converties en USD |
| USER_SPENDING_BY_TYPE | (user_id, transaction_type) | user_id | Agrégations par type et utilisateur |
| SPEND_LAST_FIVE_MIN_BY_TYPE | transaction_type | - | Dépense glissante 5 min |
| USERS | user_id | - | Utilisateurs anonymisés |
| PERMISSIONS | (user_id, topic_name) | user_id | Gestion des accès par utilisateur |

Diagramme Entité-Association (textuel) :

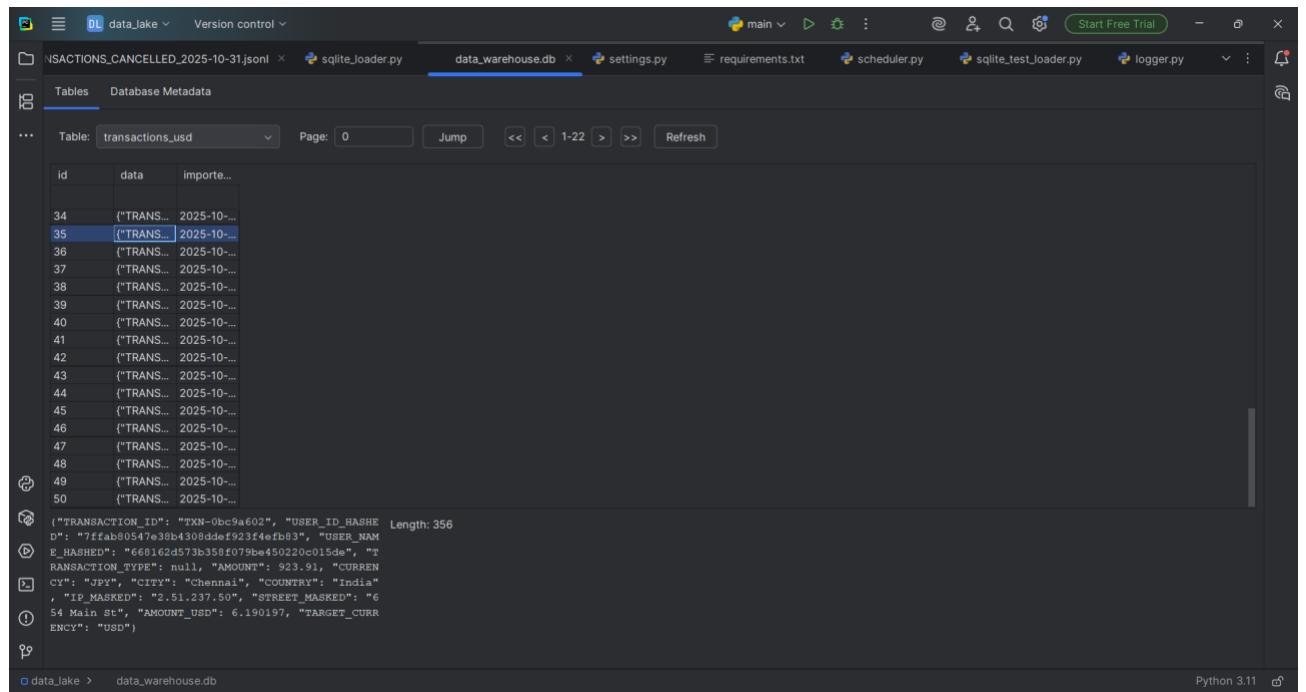
USERS (user_id PK)



PERMISSIONS (user_id FK, topic_name, access_level)



```
(venv) PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datolakes_&_Data_Integration\TP\data_lake> python -m jobs.sqlite_loader
[Tous les dossiers du Data Lake ont été vérifiés/créés (2025-10-31).
✓ Données importées dans table transactions_secure depuis pipeline\transactions_secure\2025-10-31
✓ Données importées dans table transactions_usd depuis pipeline\transactions_usd\2025-10-31
✓ Données importées dans table transactions_blacklisted depuis pipeline\transactions_blacklisted\2025-10-31
✓ Données importées dans table transactions_completed depuis pipeline\transactions_completed\2025-10-31
✓ Données importées dans table transactions_failed depuis pipeline\transactions_failed\2025-10-31
✓ Données importées dans table transactions_pending depuis pipeline\transactions_pending\2025-10-31
✓ Données importées dans table transactions_processing depuis pipeline\transactions_processing\2025-10-31
✓ Données importées dans table transactions_cancelled depuis pipeline\transactions_cancelled\2025-10-31
✓ Données importées dans table spend_last_five_min_by_type depuis pipeline\spend_last_five_min_by_type\2025-10-31
✓ Données importées dans table user_spending_by_type depuis pipeline\user_spending_by_type\2025-10-31
Import Data Lake → SQLite terminé.
(venv) PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datolakes_&_Data_Integration\TP\data_lake>
```



The screenshot shows a DBeaver database interface connected to the 'data_warehouse.db' database. The current table is 'transactions_usd'. The table has three columns: 'id', 'data', and 'imported...'. There are 50 rows of data. The 35th row is currently selected. The interface also shows other files in the background, such as 'main.py', 'sqlite_loader.py', 'settings.py', 'requirements.txt', 'scheduler.py', 'sqlite_test_loader.py', and 'logger.py'.

| id | data | imported... |
|----|------------|-------------|
| 34 | {"TRANS... | 2025-10... |
| 35 | {"TRANS... | 2025-10... |
| 36 | {"TRANS... | 2025-10... |
| 37 | {"TRANS... | 2025-10... |
| 38 | {"TRANS... | 2025-10... |
| 39 | {"TRANS... | 2025-10... |
| 40 | {"TRANS... | 2025-10... |
| 41 | {"TRANS... | 2025-10... |
| 42 | {"TRANS... | 2025-10... |
| 43 | {"TRANS... | 2025-10... |
| 44 | {"TRANS... | 2025-10... |
| 45 | {"TRANS... | 2025-10... |
| 46 | {"TRANS... | 2025-10... |
| 47 | {"TRANS... | 2025-10... |
| 48 | {"TRANS... | 2025-10... |
| 49 | {"TRANS... | 2025-10... |
| 50 | {"TRANS... | 2025-10... |

Below the table, there is a detailed view of the selected row (row 35). The row contains the following data:

- TRANSACTION_ID: "TXN-0bc9a602", "USER_ID_HASHED": "Length: 356", "USER_NAME": "fftab0547e3b4300dddf923f4efeb83"
- E_HASHED: "66b162d573b358f079be450220c015de", "TRANSACTION_TYPE": null, "AMOUNT": 923.91, "CURRENCY": "JPY", "CITY": "Chennai", "COUNTRY": "India", "IP_MASKED": "2.51.237.50", "STREET_MASKED": "654 Main St", "AMOUNT_USD": 6.190197, "TARGET_CURRENCY": "USD"

3. Kafka Consumers

Les Kafka Consumers permettent de lire les messages, les écrire dans le Data Lake et charger les données dans le Data Warehouse.

Exemple simplifié :

```
from kafka import KafkaConsumer
import json, os, datetime

consumer = KafkaConsumer('TRANSACTIONS_SECURE',
bootstrap_servers='localhost:9092')
for msg in consumer:
    data = json.loads(msg.value)
    date_str = datetime.date.today().isoformat()
    path = f"data_lake/TRANSACTIONS_SECURE/{date_str}"
    os.makedirs(path, exist_ok=True)
    with open(f"{path}/data.jsonl", "a") as f:
        f.write(json.dumps(data) + "\n")
```

4. Gouvernance et Sécurité

| Domaine | Mécanisme | Objectif |
|-------------------------------|---|----------------------------------|
| Suppression historique | Script Python supprimant les données après 30 jours | Optimisation & conformité RGPD |
| Gestion des droits | Table permissions (user_id, topic_name, access_level) | Contrôle des accès |
| Feed Kafka | Déclaration dans settings.py
→ dossier auto → pipeline réutilisé | Gain de temps |
| Anonymisation | Masquage KSQLDB (MASK, HASH, CONCAT) | Protection des données sensibles |
| Traçabilité | Transaction_id + timestamp | Audit & transparence |
| | | |

```
PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datalakes_&_Data_Integration\TP\data_lake> python -m jobs.cleanup
(venv) PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datalakes_&_Data_Integration\TP\data_lake> python -m jobs.permissions_manager
✓ Table 'permissions' initialisée.
🔒 Permission ajoutée pour admin sur TRANSACTIONS_SECURE.
🔓 Permission ajoutée pour analyst sur TRANSACTIONS_USD.
[(1, 'admin', 'TRANSACTIONS_SECURE', 1, 1, '2025-11-01 19:24:28'), (2, 'analyst', 'TRANSACTIONS_USD', 1, 0, '2025-11-01 19:24:28')]
(venv) PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datalakes_&_Data_Integration\TP\data_lake>
```

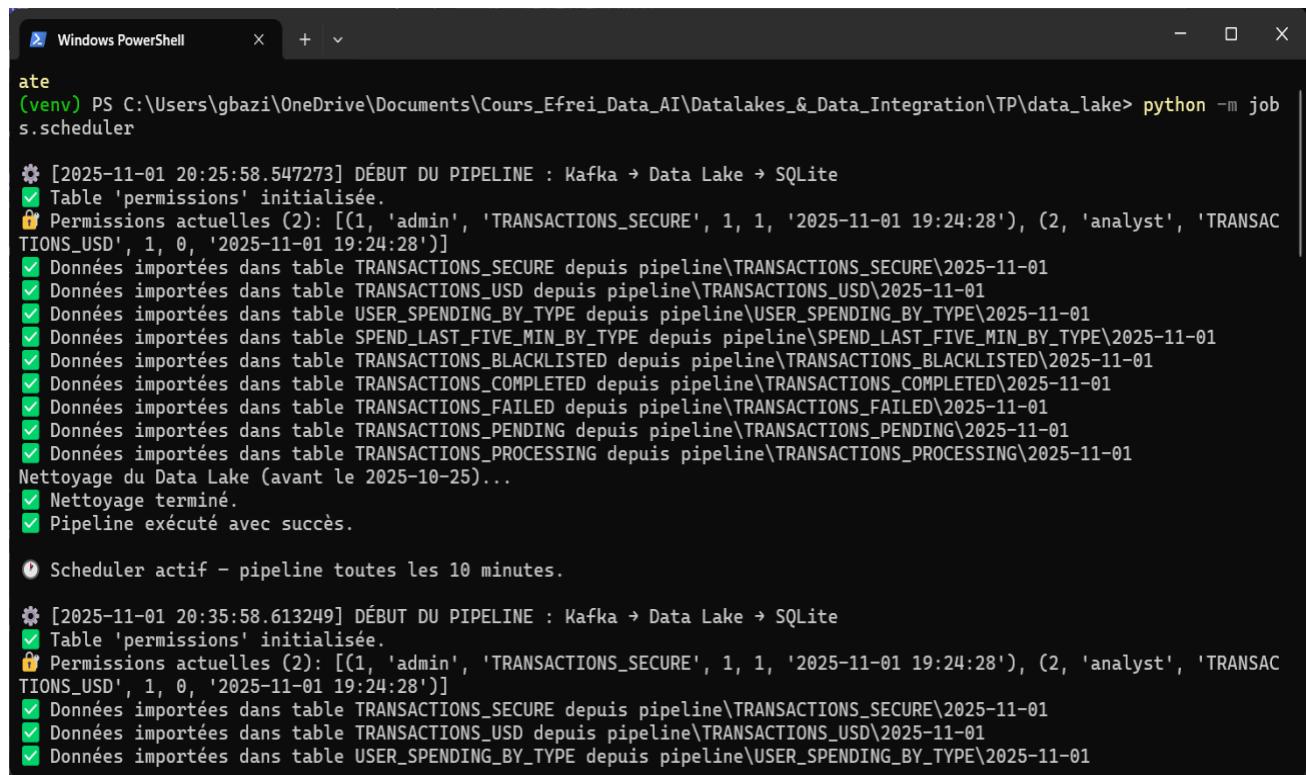
5. Orchestration et Optimisation

Orchestration automatisée via schedule ou Apache Beam (toutes les 10 minutes).

```
import schedule, os, time
```

```
def run_jobs():
    os.system("python consumers/kafka_to_datalake.py")
    os.system("python jobs/sqlite_loader.py")

schedule.every(10).minutes.do(run_jobs)
while True:
    schedule.run_pending()
    time.sleep(60)
```



```
PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datilakes_&_Data_Integration\TP\data_lake> python -m job_scheduler

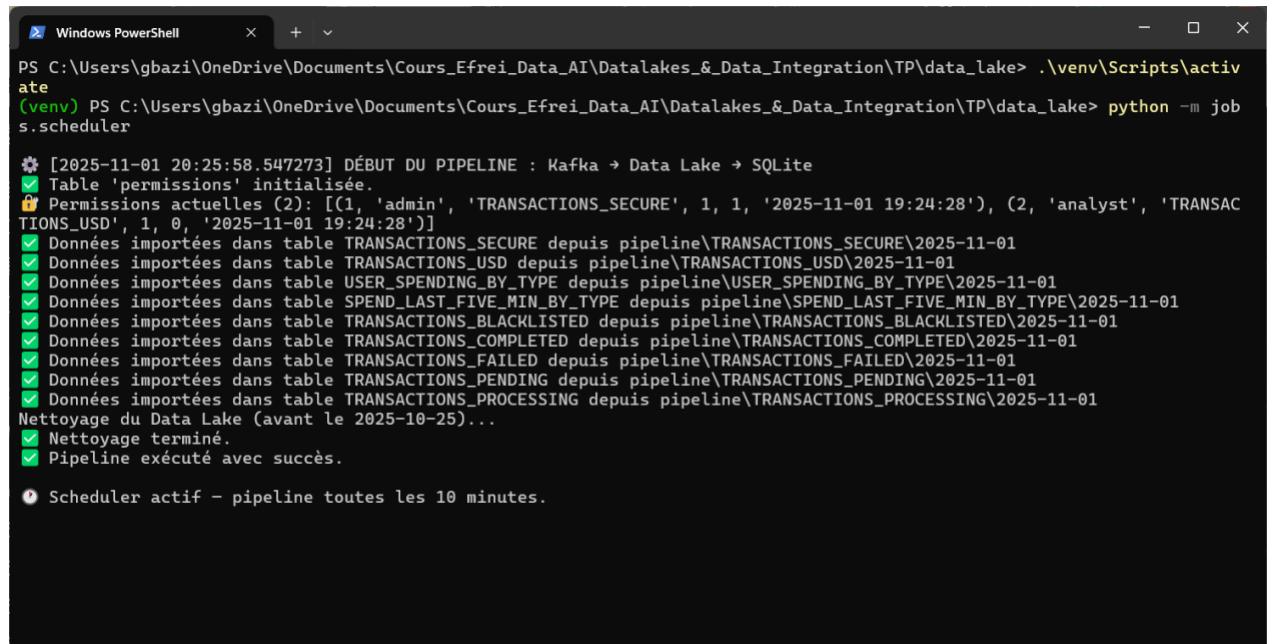
[2025-11-01 20:25:58.547273] DÉBUT DU PIPELINE : Kafka → Data Lake → SQLite
✓ Table 'permissions' initialisée.
⚠ Permissions actuelles (2): [(1, 'admin', 'TRANSACTIONS_SECURE', 1, 1, '2025-11-01 19:24:28'), (2, 'analyst', 'TRANSACTIONS_USD', 1, 0, '2025-11-01 19:24:28')]
✓ Données importées dans table TRANSACTIONS_SECURE depuis pipeline\TRANSACTIONS_SECURE\2025-11-01
✓ Données importées dans table TRANSACTIONS_USD depuis pipeline\TRANSACTIONS_USD\2025-11-01
✓ Données importées dans table USER_SPENDING_BY_TYPE depuis pipeline\USER_SPENDING_BY_TYPE\2025-11-01
✓ Données importées dans table SPEND_LAST_FIVE_MIN_BY_TYPE depuis pipeline\SPEND_LAST_FIVE_MIN_BY_TYPE\2025-11-01
✓ Données importées dans table TRANSACTIONS_BLACKLISTED depuis pipeline\TRANSACTIONS_BLACKLISTED\2025-11-01
✓ Données importées dans table TRANSACTIONS_COMPLETED depuis pipeline\TRANSACTIONS_COMPLETED\2025-11-01
✓ Données importées dans table TRANSACTIONS_FAILED depuis pipeline\TRANSACTIONS_FAILED\2025-11-01
✓ Données importées dans table TRANSACTIONS_PENDING depuis pipeline\TRANSACTIONS_PENDING\2025-11-01
✓ Données importées dans table TRANSACTIONS_PROCESSING depuis pipeline\TRANSACTIONS_PROCESSING\2025-11-01
Nettoyage du Data Lake (avant le 2025-10-25)...
✓ Nettoyage terminé.
✓ Pipeline exécuté avec succès.

⌚ Scheduler actif - pipeline toutes les 10 minutes.

[2025-11-01 20:35:58.613249] DÉBUT DU PIPELINE : Kafka → Data Lake → SQLite
✓ Table 'permissions' initialisée.
⚠ Permissions actuelles (2): [(1, 'admin', 'TRANSACTIONS_SECURE', 1, 1, '2025-11-01 19:24:28'), (2, 'analyst', 'TRANSACTIONS_USD', 1, 0, '2025-11-01 19:24:28')]
✓ Données importées dans table TRANSACTIONS_SECURE depuis pipeline\TRANSACTIONS_SECURE\2025-11-01
✓ Données importées dans table TRANSACTIONS_USD depuis pipeline\TRANSACTIONS_USD\2025-11-01
✓ Données importées dans table USER_SPENDING_BY_TYPE depuis pipeline\USER_SPENDING_BY_TYPE\2025-11-01
```

Synthèse

| Domaine | Résumé |
|-----------------------------------|---|
| Data Lake | Stockage brut JSON partitionné par date, extensible, append/overwrite selon le type de données. |
| Data Warehouse | Schéma relationnel clair (SQLite/MySQL), clés primaires et étrangères définies. |
| Kafka Consumers | Automatisation de la collecte et intégration continue. |
| Sécurité & Gouvernance | Rétention, anonymisation, permissions et logs d'audit. |
| Orchestration | Automatisation des jobs toutes les 10 minutes via schedule / Apache Beam. |



```

Windows PowerShell
PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datalakes_&_Data_Integration\TP\data_lake> .\venv\Scripts\activate
(venv) PS C:\Users\gbazi\OneDrive\Documents\Cours_Efrei_Data_AI\Datalakes_&_Data_Integration\TP\data_lake> python -m jobs.scheduler

⚙️ [2025-11-01 20:25:58.547273] DÉBUT DU PIPELINE : Kafka → Data Lake → SQLite
✅ Table 'permissions' initialisée.
⚠️ Permissions actuelles (2): [(1, 'admin', 'TRANSACTIONS_SECURE', 1, 1, '2025-11-01 19:24:28'), (2, 'analyst', 'TRANSACTIONS_USD', 1, 0, '2025-11-01 19:24:28')]
✅ Données importées dans table TRANSACTIONS_SECURE depuis pipeline\TRANSACTIONS_SECURE\2025-11-01
✅ Données importées dans table TRANSACTIONS_USD depuis pipeline\TRANSACTIONS_USD\2025-11-01
✅ Données importées dans table USER_SPENDING_BY_TYPE depuis pipeline\USER_SPENDING_BY_TYPE\2025-11-01
✅ Données importées dans table SPEND_LAST_FIVE_MIN_BY_TYPE depuis pipeline\SPEND_LAST_FIVE_MIN_BY_TYPE\2025-11-01
✅ Données importées dans table TRANSACTIONS_BLACKLISTED depuis pipeline\TRANSACTIONS_BLACKLISTED\2025-11-01
✅ Données importées dans table TRANSACTIONS_COMPLETED depuis pipeline\TRANSACTIONS_COMPLETED\2025-11-01
✅ Données importées dans table TRANSACTIONS_FAILED depuis pipeline\TRANSACTIONS_FAILED\2025-11-01
✅ Données importées dans table TRANSACTIONS_PENDING depuis pipeline\TRANSACTIONS_PENDING\2025-11-01
✅ Données importées dans table TRANSACTIONS_PROCESSING depuis pipeline\TRANSACTIONS_PROCESSING\2025-11-01
Nettoyage du Data Lake (avant le 2025-10-25)...
✅ Nettoyage terminé.
✅ Pipeline exécuté avec succès.

⌚ Scheduler actif - pipeline toutes les 10 minutes.

```

