Projet ELT

Services Bancaires

Présenté par *Gherson Gloria & Mahassadi Jean-marie*

Sommaire

Pourquoi ce projet ?

- Comprendre l’utilisation des services bancaires en Afrique à partir d’un jeu de données libre.
- Construire un pipeline ELT automatisé pour fiabiliser les chiffres.
- Fournir aux analystes une base PostgreSQL “clean” au lieu d’un CSV brut.

Vue d’ensemble du *pipeline*

- **Extract**: copie le CSV quotidien dans data/raw/.
- **Load**: exécute un COPY vers stg_services_bancaires (données brutes).
- **Transform**: nettoie et range dans core_services_bancaires.
- **Orchestration**: un script run_pipeline.py lance les trois étapes d’un seul coup.

Schéma des *tables*

- stg_services_bancaires
  ↳ 13 colonnes identiques au CSV, aucune contrainte.

- core_services_bancaires
  ↳ clés : année + pays + tranche_age + type_operation (dédoublonnage).
  ↳ valeurs numériques typées (BIGINT, NUMERIC(18,2)), textes normalisés (trim, initcap).

- Index pays, année pour accélérer les agrégations.

Organisation de *l'équipe*

1

Gherson Gloria

a conçu le schéma et créé les deux tables — stg_services_bancaires et core_services_bancaires

2

Mahassadi Jean-marie

a développé l'ensemble des scripts — extract.py, load.py, transform.py, run_pipeline.py

Difficultés rencontrées

Connexion Python ↔ Postgres

premiers échecs avec la chaîne de connexion et le mot de passe ; résolu avec un fichier .env

Import du CSV

erreur “nombre de colonnes incorrect” ; la cause : séparateur et encodage ; corrigé en précisant FORMAT csv, HEADER true, ENCODING UTF-8

Clé primaire & doublons

apparition de lignes dupliquées lors d’un second chargement ; solution : clé composite + ON CONFLICT DO NOTHING

Perspectives *d’amélioration*

- **Planifier le pipeline**

« Aujourd’hui on lance le script à la main ; l’étape suivante, c’est de le déclencher tous les jours automatiquement avec un simple cron ou l’extension pg_cron. »

- **Héberger la base dans le cloud**

« On peut déplacer PostgreSQL sur Supabase ou AWS RDS ; la base sera accessible partout et sauvegardée automatiquement. »

- **Ajouter des tests automatisés**

« Avec pytest et Docker, on pourra rejouer le pipeline en local ou en CI pour vérifier que chaque changement de code garde les résultats corrects. »

Conclusion

- Pipeline simple : Prendre → Déposer → Ranger ; désormais 100 % automatisé.
- Code et documentation publics sur GitHub.
- Prochaine étape : industrialisation Cloud + visualisation.

Merci pour votre écoute