TD N°1 – Mise en pratique d'un ETL avec PostgreSQL

Contexte

Vous travaillez comme **data engineer junior** dans une PME.
L'entreprise stocke ses données clients et ventes dans des fichiers CSV exportés de différents systèmes.
Votre mission : mettre en place un **pipeline ETL simple** pour intégrer ces données dans une base **PostgreSQL** en respectant les bonnes pratiques.

Jeu de données fourni

Vous disposez de deux fichiers CSV (petit volume, quelques centaines de lignes) :

- `clients.csv`

```
id,nom,adresse,numero_telephone
1,Dupont,,0612345678
2,Martin,12 rue des Lilas,0033612345678
3,Durand,45 avenue de Paris,+33612345679
4,Doe,NULL,061234567
5,Test,"10 rue de la Paix",not_a_number
...
```

- `ventes.csv`

```
id,id_client,id_produit,date_vente,montant
1,1,101,2024-01-15,1200
2,1,102,2024-05-20,800
3,2,101,2023-10-10,2000
4,3,103,2099-01-01,500
5,4,104,2024-06-01,-50
6,2,101,2023-10-10,2000   -- doublon
...
```

Étapes du TD

Exercice 1 – Extraction des données

1. Importez les deux fichiers CSV dans PostgreSQL (`COPY` ou outil d'import).
2. Vérifiez le contenu avec quelques requêtes simples (`COUNT` , `DISTINCT` , `NULL`).
3. Écrivez les requêtes suivantes :
    - Clients ayant acheté dans les 6 derniers mois.
    - Clients avec au moins deux achats > 1000 €.
    - Clients n'ayant jamais acheté.
    - Montant total des ventes par client.

Exercice 2 – Transformation des données

Les données importées sont sales. Appliquez les transformations suivantes :

1. Supprimez les clients avec une adresse vide ou `NULL` .
2. Nettoyez les numéros de téléphone pour qu'ils soient tous au format **international français** (`+33XXXXXXXXX`).
    - Exemple : `0612345678` → `+33612345678`
    - `0033612345678` → `+33612345678`
    - Tout numéro incorrect → à mettre à `NULL` .
3. Supprimez les ventes avec des dates incohérentes (ex. ventes en 2099).
4. Remplacez tout montant négatif par `0` .
5. Supprimez les doublons dans `ventes` (même client, produit, date, montant).

► Produisez une table propre `clients_clean` et `ventes_clean` .

Exercice 3 – Chargement avec partitionnement

1. Créez une table `ventes_partitionnees` partitionnée par **année de vente** (`date_vente`).
    - Une partition pour `2023` , une pour `2024` , une pour `future` .

2. Chargez toutes les ventes nettoyées dans cette table.
3. Vérifiez que les lignes sont bien réparties dans les bonnes partitions.

Exercice 4 – Gestion des erreurs

Pendant le chargement, vous observez :

- Un problème de **doublon de clé primaire** sur `clients_clean`.
- Un problème de type (ex : `not_a_number` dans le téléphone).

Questions :

- Quelle est la cause de chaque erreur ?
- Quelles solutions proposez-vous (SQL `ON CONFLICT`, nettoyage préalable, règles de validation) ?

Exercice 5 – Optimisation du pipeline

Proposez des optimisations pour rendre le pipeline ETL plus performant :

- Index, transactions, parallélisation…
- Montrez un exemple concret avec `EXPLAIN ANALYZE` pour comparer une requête avec et sans index (par exemple sur `ventes_clean(id_client)`).

Bonus (si temps disponible)

- Écrire un petit script Python (pandas ou psycopg2) pour :
  - Lire `clients.csv`, nettoyer les téléphones, et recharger la table `clients_clean`.
- Comparer la flexibilité Python vs SQL pour les transformations.