TP - Cycle de vie de la donnée : de la source au Dashboard

Objectif : Simuler un projet data complet, de la découverte de la donnée brute à la création d'un Dashboard décisionnel, en intégrant les bonnes pratiques de modélisation (Médaillon) et de sécurité.

Outils : OpenMetadata(image Docker complète seulement), PostgreSQL, Metabase

Scénario : Vous êtes Data Engineer/Analyst chez "VéloCity", une entreprise de location de vélos en libre-service. La direction Marketing souhaite un Dashboard pour suivre l'activité quotidienne : nombre de locations, durée moyenne des trajets, les vélos les plus utilisés, habitude par ville, âge des consommateurs, type d'abonnement pris, ... etc .

Les données brutes sont ingérées dans une base PostgreSQL, mais elles sont "telles quelles", complexes et pas toujours propres. Votre catalogue OpenMetadata est (partiellement) à jour et documente ces sources.

Lien : github

Partie 1 : Découverte et Compréhension (OpenMetadata ou fichier yaml)

Objectif : Identifier les données sources pertinentes pour répondre aux besoins métiers.

1. **Connexion :** Connectez-vous à l'instance OpenMetadata de "VéloCity".
2. **Exploration :**
    - Naviguez dans le catalogue et identifiez les tables qui semblent pertinentes pour ce TP
    - Utilisez la recherche et les "Tags" (ex: "Source", "PII") pour trouver les bonnes tables.
3. **Analyse :**
    - Pour chaque table pertinente, examinez (si disponible) :
        - Le **schéma**
        - La **documentation**
        - Les **profils de données**
        - Le **propriétaire** ("Owner") pour savoir qui contacter
4. **Synthèse :** Listez les tables que vous allez utiliser, et justifiez pourquoi. Sont-elles des tables de fait ou de dimension ?

Partie 2 : Modélisation et Transformation (PostgreSQL)

Objectif : Appliquer l'architecture Médaillon (Raw / Silver /Gold) pour préparer des données propres et agrégées.

1. **Préparation :**
    - Connectez-vous à la base PostgreSQL avec pgadmin.
    - Requêter les tables identifiées sur OpenMetadata :
        - Métadonnée : trouvez-vous les mêmes informations ?
        - Qualité : relevez-vous des problèmes qui seront à corriger ?
    - Créez un nouveau schéma pour vos transformations : *CREATE SCHEMA analytics_nom1_nom2* (si vous travaillez en binôme).
2. **Couche Silver (Raffinage) :**

L'objectif est de nettoyer, typer et standardiser les données brutes afin qu'elles soient exploitables.

Pour chaque table identifiée dans le schema *raw*, créer une table dans le schema *silver* en apportant des corrections sur les anomalies détectées.

Quelques exemples :

- Transtyper les dates au format texte, à des dates / timestamps
- Calculer les durées en minutes
- Filtrer les trajets tests / avortés (ex : durée < 2 minutes)
- Standardiser les valeurs / formats
- Exclure les valeurs aberrantes
- ...

3. **Couche Gold (Agrégation Métier) :**

L'objectif est de créer une table *prête à l'emploi* pour le Dashboard, répondant au besoin métier.

Tâche : Créez une table analytics_nom1_nom2.gold_daily_activity, afin de retrouver par jour, par ville, station, type de vélos et par abonnement, certaines métriques agrégées :

- *total_rentals* : nombre total de location
- *average_duration_minutes* : temps moyen de la location en minute
- *unique_users* : nombre d'utilisateurs uniques

Partie 3 : Visualisation (Metabase)

Objectif : Créer un Dashboard simple pour le métier Marketing.

1. **Connexion :** Connectez-vous à Metabase.
2. **Source de données :**
    - Ajoutez la base PostgreSQL comme source
    - Ajoutez la table analytics_nom1_nom2.gold_daily_activity comme nouveau "Dataset" dans Superset.
3. **Création des "Charts" :**
    - **Chart 1 (Courbe) :** Évolution du total_rentals dans le temps (axe X = jour).
    - **Chart 2 (Bar) :** Top 3 des villes par total_rentals.
    - **Chart 3 (Indicateur/KPI) :** average_duration_minutes par ville.
4. **Dashboard :**
    - Créez un nouveau Dashboard "Suivi Activité VéloCity".
    - Ajoutez-y les 3 graphiques suivants :
        1. **Chart 1 (Courbe) :** Évolution du total_rentals dans le temps (axe X = jour).
        2. **Chart 2 (Bar) :** Top 3 des villes par total_rentals.
        3. **Chart 3 (Indicateur/KPI) :** average_duration_minutes par ville.

Partie 4 : Sécurité et Gouvernance (PostgreSQL +)

Objectif : Comprendre et appliquer le principe du moindre privilège.

Le service Marketing ne doit *jamais* voir les tables raw ou silver, ni les données personnelles des utilisateurs. Ils ne doivent avoir accès qu'à la table Gold agrégée.

1. **Scénario :** Votre administrateur a créé un rôle (utilisateur) pour le service marketing : marketing_user.
2. **Audit (Simulation) :**
    - Si marketing_user essaie de faire SELECT * FROM raw.user_accounts;, que doit-il se passer ?
    - Si il fait SELECT * FROM *analytics_nom1_nom2.gold_daily_activity*; ?
3. **Tâche (Script SQL) :**
    - Écrivez les commandes SQL pour implémenter cette règle de sécurité.
    - *Indice :* Vous aurez besoin de GRANT USAGE ON SCHEMA ..., GRANT SELECT ON TABLE ... et potentiellement REVOKE ou DENY sur les autres schémas.
    - GRANT USAGE ON SCHEMA analytics_nom1_nom2 TO marketing_user;
    - GRANT SELECT ON analytics_nom1_nom2.gold_daily_activity TO marketing_user;
    - REVOKE ALL ON SCHEMA raw FROM marketing_user; (ou s'assurer qu'ils n'ont pas d'accès par défaut).
4. **Row-level Security**
    - Créer un rôle *manager_lyon*
    - Ne lui donner accès qu'à la table GOLD, pour la ville 'Lyon'
    - Tester le rôle pour vous assurer que celui-ci fonctionne bien comme convenu

---

Livrables

Partie 1 : Un document (Markdown ou PDF) listant les tables sources identifiées et répondant aux questions posées.

Partie 2 : Un script SQL unique (.sql) contenant la création des schémas, tables, transformations

Partie 3 : Une capture d'écran du Dashboard final

Partie 4 : Les commandes GRANT / REVOKE pour les accès (à ajouter au script de la partie 2)