Construisez vos pipelines de données avec Databricks

Trouvez et chargez vos données dans le Lakehouse

Correction

Retrouvez aussi les requêtes de correction sur GitHub.

Voici la démarche d'un praticien, commande par commande. Le chargement se fait par l'assistant d'import, dans `frequence.socle`. Dans l'aperçu, regardez `date_ajout`, écrite à la française (`01/12/2025`) : l'assistant doit l'avoir reconnue comme *date* (icône de calendrier) et afficher `2025-12-01`, pas `2025-01-12`. C'est le cas en novembre 2026. Si votre aperçu la montrait en texte, chargez-la telle quelle, puis convertissez-la en SQL avec `to_date(date_ajout, 'dd/MM/yyyy')`. Puis la documentation :

```
COMMENT ON TABLE frequence.socle.playlists IS
  'Ajouts de titres en playlist, une ligne par ajout. Source : export plateforme d'écoute, chargé à la main.';

ALTER TABLE frequence.socle.playlists
  ALTER COLUMN date_ajout COMMENT 'Date d'ajout du titre à la playlist';
```

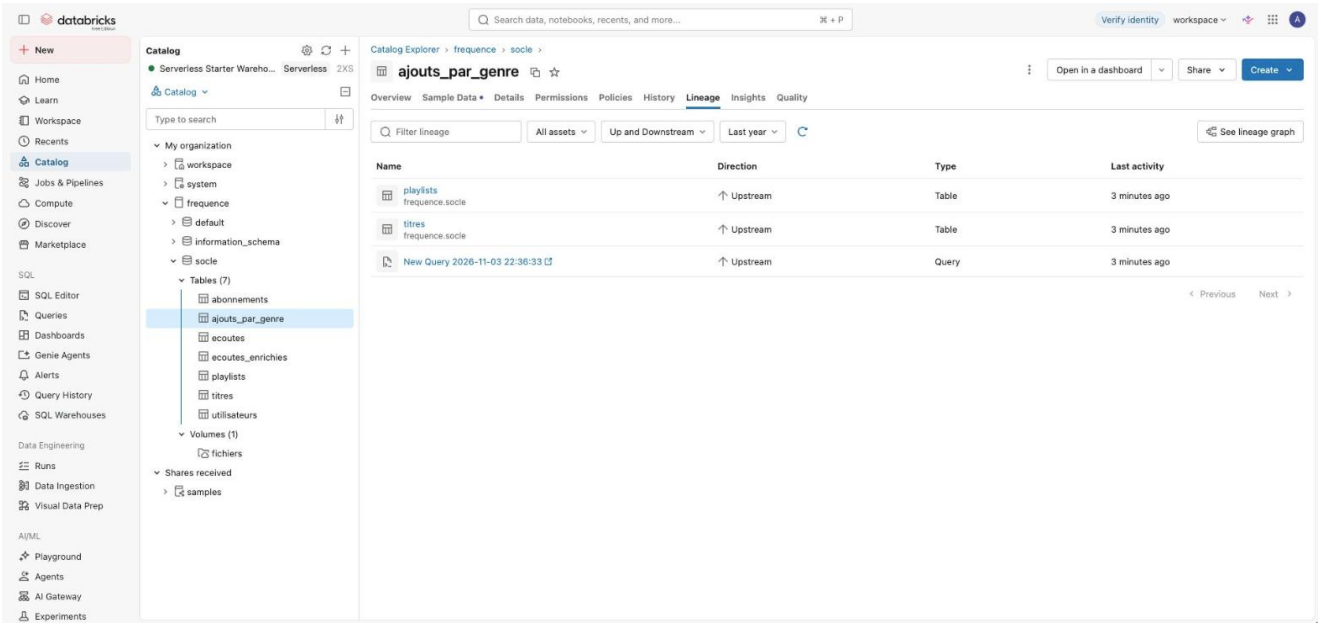
Les contrôles : `COUNT(*)` à comparer au fichier ; `WHERE id_playlist IS NULL OR id_titre IS NULL` doit renvoyer zéro ; `WHERE date_ajout IS NULL` doit renvoyer zéro ; un `GROUP BY id_playlist, id_titre HAVING COUNT(*) > 1` doit renvoyer zéro ligne. Si ce dernier renvoie quelques lignes, ce n'est pas forcément une erreur de chargement (un même titre ajouté deux fois à une playlist) : notez-le dans la description, c'est une information sur la source.

Résultats attendus : 33 034 lignes, zéro identifiant nul, zéro date nulle, des dates du 6 juin 2025 au 30 septembre 2026. Et le contrôle des doublons renvoie **37 couples** (`id_playlist`, `id_titre`) : le même titre ajouté deux fois à la même playlist, à des dates différentes. C'est la surprise du fichier, à consigner dans la description de la table.

La table dérivée :

```
CREATE TABLE frequence.socle.ajouts_par_genre AS
SELECT t.genre, COUNT(*) AS nb_ajouts
FROM frequence.socle.playlists p
JOIN frequence.socle.titres t ON p.id_titre = t.id_titre
GROUP BY t.genre;
```

Le résultat compte douze genres, menés par le Rap FR (6 396 ajouts), la Pop (5 948) et le Rock (4 236). Le lineage attendu montre `playlists` et `titres` en amont, rien en aval.



Les deux erreurs les plus probables. Première : `date_ajout` restée en texte. Symptôme : la table se crée, mais `MIN(date_ajout)` renvoie une chaîne, et tout filtre par mois est faux ou vide. Seconde : la table créée hors du schéma, dans le catalogue par défaut (`workspace.default`). Symptôme : `SELECT * FROM frequence.socle.playlists` renvoie « table introuvable » alors que l'import a réussi. Dans les deux cas, le nom complet et la requête de contrôle révèlent le problème ; la grille en 4 points existe pour ça.

Sur l'automatisation, enfin. Un fichier de playlists qui arrive chaque jour, c'est une fréquence régulière et, avec le temps, une fraîcheur attendue par le métier : les deux critères plaident pour l'automatisation. Le risque du manuel n'est pas le temps passé, c'est le jour oublié, qui fausse silencieusement la table dérivée et tout ce qui en dépendra. Un bon rendu nomme ce risque. Une réponse « à la main, ça prend cinq minutes » ignore qu'il faudra le faire 250 fois par an, et que le 25e jour, personne n'y pensera.