Construisez vos pipelines de données avec Databricks

Interrogez vos données avec le SQL Warehouse

Correction

Retrouvez aussi les requêtes de correction sur GitHub.

Avant toute réponse, la requête de contrôle est celle de l'exemple travaillé : `ecoutes_total` et `ecoutes_rattachees` doivent être égaux, et égaux au nombre de lignes d'`ecoutes`.

Valeurs attendues : 498 586 lignes dans `ecoutes`, et les mêmes 498 586 pour `ecoutes_total` et `ecoutes_rattachees`. Avec la jointure sur `id_utilisateur` seul, le total monte à 679 356 : c'est la duplication que le contrôle doit révéler.

Question 1, la saisonnalité mensuelle. Reformulation : nombre d'écoutes par mois, sur les douze mois. Aucune jointure n'est nécessaire, et c'est la première chose à remarquer : on ne joint pas par réflexe.

```
SELECT DATE_TRUNC('month', horodatage) AS mois, COUNT(*) AS nb_ecoutes
FROM frequence.socle.ecoutes
GROUP BY mois
ORDER BY mois;
```

Résultat attendu : douze lignes, de 38 038 écoutes (février 2026) à 46 403 (décembre 2025), avec un creux en juillet-août (39 543 et 39 056). Une variante correcte compte la durée plutôt que le nombre d'écoutes : acceptable, si la reformulation le dit.

Question 2, le terminal le plus utilisé par offre. Reformulation : pour chaque offre, le terminal qui totalise le plus d'écoutes, l'offre étant celle active à la date de l'écoute. Ici la jointure datée est obligatoire.

```
WITH par_terminal AS (
  SELECT a.offre, e.terminal, COUNT(*) AS nb_ecoutes
  FROM frequence.socle.ecoutes e
  JOIN frequence.socle.abonnements a
    ON e.id_utilisateur = a.id_utilisateur
   AND e.horodatage >= a.date_debut
   AND e.horodatage < COALESCE(a.date_fin, CURRENT_TIMESTAMP())
  GROUP BY a.offre, e.terminal
)
SELECT offre, terminal, nb_ecoutes
FROM par_terminal
QUALIFY ROW_NUMBER() OVER (PARTITION BY offre ORDER BY nb_ecoutes DESC) = 1;
```

Résultat attendu : le mobile arrive en tête pour les deux offres, avec 153 213 écoutes pour l'offre gratuite et 157 766 pour l'offre premium. Le détail est plus parlant que le podium : les abonnés gratuits n'écoutent que sur mobile, web et ordinateur, quand les premium utilisent aussi l'enceinte connectée (39 868 écoutes) et la télévision (17 152).

`QUALIFY` filtre le résultat d'une fonction de fenêtre, comme `HAVING` filtre le résultat d'un agrégat. Databricks SQL le prend en charge ; sans lui, il faudrait une sous-requête supplémentaire.

La variante fausse mais plausible : la même requête avec `ON e.id_utilisateur = a.id_utilisateur` seul. Elle s'exécute, renvoie un terminal par offre, et compte deux fois les écoutes des abonnés qui ont changé d'offre. Le podium peut rester le même, ou pas. On ne le sait pas, et c'est précisément le problème.

Question 3, la part des abonnés premium sans écoute depuis 30 jours. Reformulation : parmi les abonnés dont l'offre premium est active à la date de référence, la part de ceux qui n'ont aucune écoute dans les 30 jours précédant cette date. La date de référence est la dernière date du jeu de données, pas la date du jour : les CSV s'arrêtent à une date fixe.

```
WITH ref AS (
  SELECT MAX(horodatage) AS date_ref FROM frequence.socle.ecoutes
),
premium_actifs AS (
  SELECT a.id_utilisateur
  FROM frequence.socle.abonnements a, ref
  WHERE a.offre = 'premium'
    AND a.date_debut <= ref.date_ref
    AND (a.date_fin IS NULL OR a.date_fin > ref.date_ref)
),
actifs_30j AS (
  SELECT DISTINCT e.id_utilisateur
  FROM frequence.socle.ecoutes e, ref
  WHERE e.horodatage > ref.date_ref - INTERVAL 30 DAYS
)
SELECT 100.0 * COUNT(*) FILTER (WHERE x.id_utilisateur IS NULL) / COUNT(*) AS part_sans_ecoute
FROM premium_actifs p
LEFT JOIN actifs_30j x ON p.id_utilisateur = x.id_utilisateur;
```

Cette définition est celle que reprendra l'application de P2C3 : gardez-la. La variante fausse ici : filtrer sur `statut = 'actif'` sans regarder les dates. Un abonné résilié hier a encore un statut qui dépend de la fraîcheur de l'export, et un abonné passé de premium à gratuit garde sa ligne premium avec un statut terminé. Les dates sont la seule vérité.

Résultat attendu : 1 176 abonnés premium actifs au 30 septembre 2026, dont 111 sans aucune écoute dans les 30 derniers jours, soit **9,44 %**. Avec la variante fausse par le statut, vous partiriez de 1 193 premium « actifs » : 24 lignes ont un statut encore à `actif` alors que leur date de fin est passée, parce que l'export n'est pas à jour.

La taille de warehouse. Le raisonnement vaut plus que la taille retenue. Volume lu : 500 000 lignes, quelques dizaines de mégaoctets, bien en deçà de ce qui justifie une montée de taille. Fréquence : trois requêtes, une fois, avant une réunion. Concurrence : un seul analyste. Conclusion : la plus petite taille, arrêt automatique à dix minutes, et on éteint après. Une réponse « une taille moyenne, pour être tranquille » n'est pas fausse, elle est non justifiée : elle paie une marge dont rien ne prouve le besoin. Une réponse « petite, mais j'augmenterais la concurrence si le marketing consultait ces chiffres chaque matin » montre que vous avez compris la différence entre taille et concurrence, et c'est exactement ce qu'un recruteur attend.