Construisez vos pipelines de données avec Databricks

Interrogez vos données en langage naturel avec Genie

Correction

Retrouvez aussi la requête de référence sur GitHub.

Les réponses de Genie peuvent varier d'une exécution à l'autre. Voici celles observées le 3 novembre 2026 sur un agent configuré comme dans ce chapitre (trois instructions, un exemple de requête), suivies de la variante fausse la plus fréquente, à savoir reconnaître.

Question 1 : la part des écoutes premium sur mobile ce trimestre.

Sortie observée. Genie part du dernier trimestre présent dans les données (`DATE_TRUNC('quarter', MAX(horodatage))`), joint `ecoutes` et `abonnements` par les dates, filtre `offre = 'premium'`, puis divise les écoutes sur mobile par toutes les écoutes premium du trimestre : **55,14 %**. Point 1 : validé. Point 2 : validé, période et offre, grâce à l'instruction sur la date de référence. Point 3 : le dénominateur est bien le total des écoutes premium du trimestre. Verdict : exploitable. Votre propre requête doit donner le même chiffre, 55,1 %. Une réserve à garder en réflexe : vérifiez les valeurs distinctes de `terminal` (ici `mobile`, `web`, `desktop`, `enceinte` et `tv`), car si la colonne contenait `mobile_ios` et `mobile_android`, le `CASE` ne compterait rien.

Variante fausse. Même structure, mais la jointure avec `abonnements` se fait sur `id_utilisateur` seul, avec `WHERE a.offre = 'premium'`. Résultat : 57,5 %, calculé sur 86 663 écoutes au lieu de 71 014. Point 1 : la jointure n'est pas datée. Un abonné passé du gratuit au premium voit toutes ses écoutes attribuées au premium, et un abonné présent dans deux lignes d'abonnement est compté deux fois. Verdict : exploitable après correction de l'agent. Vérifiez que l'exemple de requête datée est bien présent, reformulez l'instruction, et reposez la question. Si la sortie reste identique, réécrivez la requête vous-même.

Question 2 : nos meilleurs abonnés.

Sortie observée. Genie classe les abonnés par **durée totale d'écoute**, sur toute l'histoire et toutes offres confondues, et renvoie les vingt premiers. Cette fois, il annonce son interprétation : « les meilleurs abonnés sont ceux ayant la plus grande durée totale d'écoute ». Point 1 : validé, avec une jointure à `abonnements` qui ne sert à rien (aucun filtre d'offre). Point 2 : aucun filtre de période ni d'offre, et la question n'en donnait pas : on ne peut ni valider ni invalider. Point 3 : deux problèmes. « Meilleur » est devenu « le plus de temps écouté », sans que personne l'ait décidé. Et la réponse annonce « 434 113 minutes écoutées » pour l'abonné en tête, alors que `duree_ecoutee` est en **secondes**, et que sa description le dit : il s'agit d'environ 120 heures. Une erreur d'unité d'un facteur 60, dans une phrase parfaitement rédigée.

Autres sorties possibles. Une autre exécution peut classer par nombre d'écoutes, ou par temps écouté sur les 30 derniers jours chez les seuls premium. Sur les dix premiers, ces deux lectures n'ont que cinq abonnés en commun.

Verdict dans les deux cas : à réécrire, ou plutôt à **redéfinir** avant toute requête. Trois lectures au moins se défendent : le plus d'écoutes, le plus de temps écouté, l'abonné premium le plus ancien encore actif. Et une quatrième, qui n'est pas dans les tables : celui qui rapporte le plus. Ce qu'il faut demander à Claire : « meilleur pour faire quoi ? ». Pour une campagne de fidélisation, c'est le temps écouté sur 30 jours chez les premium. Une fois la définition arrêtée, elle devient une instruction de l'agent, et la question redevient exploitable. Un bon rendu mentionne ce raisonnement, pas seulement « ambigu ».

Les deux pièges à repérer dans votre propre analyse : valider le point 3 parce que le résultat « a l'air bon », et corriger le SQL à la main sans toucher l'agent, ce qui laisse Claire face à la même erreur la semaine suivante.