SQL et Logistique de Noël

pour le 6 Janvier 2026

Introduction

Le Père Noël a rencontré la classe de PC et a décidé de moderniser sa logistique et il utilise désormais une base de données pour gérer et planifier la distribution de tous les cadeaux. L'objectif de ce devoir est d'utiliser le langage **SQL** pour interroger cette base de données et l'aider à répondre à divers besoins d'information.

Structure de la Base de Données

La base de données est composée de trois tables principales reliées entre elles.

Table Enfants

Elle contient les informations sur les destinataires des cadeaux.

Attribut	Description	Type de Donnée	Contrainte
`IdEnfant`	Identifiant unique	`INTEGER`	**Clé Primaire**
`Prénom`	Prénom de l'enfant	`VARCHAR(50)`	
`Adresse`	Adresse de livraison	`VARCHAR(255)`	
`Âge`	Âge de l'enfant	`INTEGER`	

Table Cadeaux

Elle répertorie tous les cadeaux disponibles dans l'entrepôt du Père Noël.

Attribut	Description	Type de Donnée	Contrainte
`IdCadeau`	Identifiant unique	`INTEGER`	**Clé Primaire**
`Nom`	Nom du cadeau	`VARCHAR(100)`	
`Prix`	Prix	`DECIMAL(5, 2)`	
`Stock`	Quantité en stock	`INTEGER`	
`ÂgeMini`	Âge minimum recommandé	`INTEGER`	
`Catégorie`	Type de cadeau (ex : Jouet, Livre, Nourriture, etc.)	`VARCHAR(50)`	

Table Distribution

Cette table d'association gère la relation entre les enfants et les cadeaux qui leur sont attribués, ainsi que l'heure de livraison prévue.

Attribut	Description	Type de Donnée	Contrainte
`IdEnfant`	Identifiant de l'enfant	`INTEGER`	**Clé Étrangère** vers `Enfants`
`IdCadeau`	Identifiant du cadeau	`INTEGER`	**Clé Étrangère** vers `Cadeaux`
`Heure`	Heure de livraison prévue	`TIME`	

La clé primaire de cette table est la combinaison des attributs (`IdEnfant`, `IdCadeau`).

Les Requêtes SQL à Rédiger

Pour chacune des questions ci-dessous, veuillez rédiger la **requête SQL complète** correspondante.

1. Afficher le `Prénom` et l'`Adresse` de tous les enfants âgés de **moins de 8 ans**.
2. Lister tous les `Noms` des cadeaux de la `Catégorie 'Jouet'`, triés par `Prix` décroissant.
3. Trouver, pour chaque `Catégorie`, le prix du cadeau le plus cher.
4. Trouver le nombre total de cadeaux en stock.
5. Compter le **nombre total de cadeaux** différents que doit livrer le Père Noël.
6. Donner le nom du livre qui a le plus d'exemplaires en stock.
7. Afficher le `Prénom` de l'enfant qui est censé recevoir son cadeau exactement à **00 :30 :00**.
8. Calculer le **prix total** de tous les cadeaux qui sont actuellement planifiés pour la distribution.
9. Identifier les **catégories de cadeaux** pour lesquelles le Père Noël a prévu **au moins deux** cadeaux différents. (Afficher la catégorie et le nombre de cadeaux dans cette catégorie.)
10. Afficher le `Prénom` de l'enfant et le `Nom` du cadeau qu'il va recevoir.
11. Le Père Noël souhaite vérifier si le stock de chaque cadeau est suffisant pour assurer toutes les distributions prévues. Lister les cadeaux dont le stock ne permet pas d'assurer toutes les distributions prévues.
12. Afficher le `Prénom` des enfants qui reçoivent un cadeau dont le `Prix` est supérieur au **prix moyen** de l'ensemble des cadeaux.
13. Afficher le `Nom` des cadeaux dont le `Stock` est inférieur au **stock moyen** des cadeaux de la même `Catégorie`.
14. Afficher les `Prénoms` des enfants qui reçoivent au moins un cadeau de la `Catégorie 'Livre'`.
15. Lister les `Noms` des cadeaux qui ne sont attribués à **aucun enfant** dans la table `Distribution`.
16. Afficher les `Prénoms` des enfants pour lesquels il existe au moins un cadeau dont l'`ÂgeMini` recommandé est **strictement supérieur** à leur `Âge`.

1. SELECT Prénom, Adresse FROM Enfants WHERE Age <= 8 ;
2. SELECT Nom, Prix FROM Cadeaux WHERE Catégorie = "Jouet" ORDER BY Prix DESC ;
3. SELECT Catégorie, MAX(Prix) FROM Cadeaux GROUP BY Catégorie ;
4. SELECT SUM(Stock) FROM Cadeaux ;
5. SELECT COUNT(DISTINCT IdCadeau) FROM Distribution ;
6. SELECT Nom FROM Cadeaux WHERE Catégorie = "Livre" AND Stock = (SELECT MAX(Stock) FROM Cadeaux WHERE Catégorie = "Livre") ;
7. SELECT Prénom, Heure FROM Enfants JOIN Distribution ON Distribution.IdEnfant = Enfants.IdEnfant WHERE Distribution.Heure = "00 :30 :00" ;
8. SELECT SUM(Prix) FROM Distribution JOIN Cadeaux ON Cadeaux.IdCadeau = Distribution.IdCadeau ;
9. SELECT Catégorie, COUNT(DISTINCT IdCadeau) AS NombreCadeaux FROM Cadeaux GROUP BY Catégorie HAVING NombreCadeaux>= 2 ;

   Explications de la requête :
   - GROUP BY Catégorie : On regroupe tous les cadeaux qui appartiennent à la même catégorie (ex : tous les 'Jouets' ensemble).
   - COUNT(DISTINCT IdCadeau) : On compte le nombre d'identifiants uniques de cadeaux dans ce groupe. Cela permet d'ignorer les doublons si un même cadeau était listé plusieurs fois par erreur.
   - HAVING ... >= 2 : Contrairement à WHERE (qui filtre les lignes individuelles), HAVING permet de filtrer les groupes après le calcul de l'agrégation. On ne garde que les catégories qui ont au moins 2 cadeaux.
10. SELECT Prénom, Nom FROM Distribution JOIN Enfants ON Distribution.IdEnfant = Enfants.IdEnfant JOIN Cadeau ON Distribution.IdCadeau = Cadeaux.IdCadeau ;
11. Pour répondre à cette question, il faut comparer deux valeurs agrégées : le stock actuel disponible dans la table Cadeaux et le nombre de fois que chaque cadeau apparaît dans la table Distribution.

    C'est une requête qui utilise une jointure, un groupement (GROUP BY) et une condition sur le résultat du calcul (HAVING).

    SELECT Nom, Stock, COUNT(Distribution.IdCadeau) AS NbDistributionsPrévues FROM Cadeaux JOIN Distribution ON Cadeaux.IdCadeau = Distribution.IdCadeau GROUP BY IdCadeau, Nom, Stock HAVING Stock < NbDistributionsPrévues ;

    Explications de la requête :
    - JOIN : On lie les cadeaux aux lignes de la table distribution. S'il y a 5 enfants qui attendent un "Robot", il y aura 5 lignes pour ce cadeau après la jointure.
    - COUNT(Distribution.IdCadeau) : On compte combien de fois le cadeau est présent dans la planification.
    - GROUP BY : On regroupe par cadeau pour pouvoir faire le calcul par article.
    - HAVING Stock < COUNT(...) : C'est ici que réside la logique de rupture de stock. On ne garde que les cadeaux où le chiffre du stock est strictement inférieur au nombre de réservations dans la table distribution.
12. SELECT Prénom FROM Enfants JOIN Distribution ON Enfants.IdEnfant = Distribution.IdEnfant JOIN Cadeaux ON Distribution.IdCadeau = Cadeaux.IdCadeau WHERE Prix > (SELECT AVG(Prix) FROM Cadeaux) ;
13. Cette requête nécessite l'utilisation d'une sous-requête corrélée. Contrairement à une sous-requête simple qui est calculée une seule fois, une sous-requête corrélée s'exécute pour chaque ligne de la requête principale afin de comparer le stock d'un cadeau à la moyenne spécifique de sa propre catégorie. SELECT Nom FROM Cadeaux AS C1 WHERE Stock < (SELECT AVG(Stock) FROM Cadeaux AS C2 WHERE C1.Catégorie = C2.Catégorie) ; **Explications de la requête :**
    - Requête externe (C1) : Elle parcourt la table Cadeaux un par un. Pour chaque cadeau (ex : un Robot de la catégorie 'Jouet'), elle transmet la catégorie à la sous-requête.
    - Sous-requête interne (C2) : Elle calcule la moyenne (AVG) du stock. La condition WHERE C1.Catégorie = C2.Catégorie force SQL à ne calculer la moyenne que pour les cadeaux appartenant à la même catégorie que celui examiné par la requête externe.

- La sélection (WHERE) : Le nom du cadeau n'est affiché que si son stock personnel est strictement inférieur au résultat renvoyé par la sous-requête.

14. SELECT Prénom FROM Enfants WHERE IdEnfant IN (SELECT IdEnfant FROM Distribution JOIN Cadeaux ON Distribution.IdCadeau =Cadeaux.IdCadeau WHERE Catégorie = 'Livre') ;
15. SELECT Nom FROM Cadeaux WHERE IdCadeau NOT IN (SELECT IdCadeau FROM Distribution) ;
    Explications de la requête :
    - La sous-requête entre parenthèses récupère tous les IdCadeau présents dans la table Distribution.
    - La requête principale sélectionne les noms des cadeaux dont l'IdCadeau n'est pas dans cette liste.
16. SELECT Prénom FROM Enfants JOIN Distribution ON Enfants.IdEnfant = Distribution.IdEnfant JOIN Cadeaux ON Distribution.IdCadeau = Cadeaux.IdCadeau WHERE ÂgeMini > Âge ;