Module 5 : Fonctions d'agrégation basiques

ACADYS

Objectifs pédagogiques

À la fin de ce module, vous serez capable de :

- Comprendre le concept de fonctions d'agrégation en SQL
- Utiliser les fonctions COUNT, SUM, AVG, MIN et MAX
- Gérer correctement les valeurs NULL dans les calculs d'agrégation
- Appliquer les fonctions d'agrégation à des cas pratiques d'analyse de données
- Combiner les fonctions d'agrégation avec d'autres éléments SQL

1. Introduction aux fonctions d'agrégation

1.1 Qu'est-ce qu'une fonction d'agrégation ?

Les fonctions d'agrégation sont des fonctions SQL qui effectuent un calcul sur un ensemble de valeurs et renvoient une valeur unique. Elles permettent de résumer des données, ce qui est essentiel pour l'analyse et la génération de rapports.

Les fonctions d'agrégation opèrent sur l'ensemble des lignes sélectionnées par la requête, après application éventuelle des filtres WHERE. Elles sont souvent utilisées avec la clause GROUP BY (que nous verrons en détail au niveau 2) pour regrouper les résultats selon certains critères.

1.2 Fonctions d'agrégation basiques

SQL propose cinq fonctions d'agrégation fondamentales :

| Fonction | Description | Exemple | |----------|-------------|---------| | COUNT() | Compte le nombre de lignes ou de valeurs non NULL | `COUNT(*)` , `COUNT(colonne)` | | SUM() | Calcule la somme des valeurs numériques | `SUM(prix)` | | AVG() | Calcule la moyenne des valeurs numériques | `AVG(prix)` | | MIN() | Trouve la valeur minimale | `MIN(prix)` | | MAX() | Trouve la valeur maximale | `MAX(prix)` |

1.3 Syntaxe générale

La syntaxe générale pour utiliser une fonction d'agrégation est :

```
SELECT fonction_agrégation(colonne)
FROM table
WHERE condition;
```

Exemples :

```
-- Nombre total de produits
SELECT COUNT(*) FROM Produits;

-- Prix moyen des produits
SELECT AVG(prix) FROM Produits;

-- Somme des stocks disponibles
SELECT SUM(stock) FROM Produits;
```

2. La fonction COUNT

2.1 COUNT(*) vs COUNT(colonne)

Il existe deux façons principales d'utiliser COUNT :

- **COUNT(*)** : Compte le nombre total de lignes, y compris les lignes contenant des valeurs NULL
- **COUNT(colonne)** : Compte le nombre de valeurs non NULL dans la colonne spécifiée

```
-- Nombre total de clients
SELECT COUNT(*) FROM Clients;

-- Nombre de clients ayant renseigné leur numéro de téléphone
SELECT COUNT(telephone) FROM Clients;
```

2.2 COUNT avec DISTINCT

La combinaison de COUNT avec DISTINCT permet de compter le nombre de valeurs uniques dans une colonne :

```
-- Nombre de villes différentes où habitent les clients
SELECT COUNT(DISTINCT ville) FROM Clients;

-- Nombre de catégories différentes auxquelles appartiennent les produits
SELECT COUNT(DISTINCT categorie_id) FROM Produits;
```

2.3 Applications pratiques de COUNT

La fonction COUNT est très utile pour obtenir des statistiques générales sur vos données :

```
-- Nombre de commandes par statut
SELECT statut, COUNT(*)
FROM Commandes
GROUP BY statut;

-- Nombre de produits dont le stock est épuisé
SELECT COUNT(*)
FROM Produits
WHERE stock = 0;

-- Vérifier si un produit spécifique existe
SELECT COUNT(*)
FROM Produits
WHERE produit_id = 123;
```

3. Les fonctions SUM et AVG

3.1 La fonction SUM

SUM additionne toutes les valeurs non NULL d'une colonne numérique :

```
-- Calcul du chiffre d'affaires total
SELECT SUM(montant_total)
FROM Commandes;

-- Calcul de la valeur totale du stock
SELECT SUM(prix * stock)
FROM Produits;
```

```
-- Somme des points de fidélité par ville
SELECT ville, SUM(fidelite_points)
FROM Clients
GROUP BY ville;
```

SUM ne peut être utilisée qu'avec des colonnes de type numérique.

3.2 La fonction AVG

AVG calcule la moyenne arithmétique des valeurs non NULL d'une colonne numérique :

```
-- Prix moyen des produits
SELECT AVG(prix)
FROM Produits;

-- Prix moyen par catégorie
SELECT categorie_id, AVG(prix)
FROM Produits
GROUP BY categorie_id;

-- Montant moyen des commandes
SELECT AVG(montant_total)
FROM Commandes;
```

3.3 Gestion des valeurs NULL avec SUM et AVG

Il est important de noter que SUM et AVG ignorent les valeurs NULL dans leurs calculs :

- SUM(colonne) additionne uniquement les valeurs non NULL
- AVG(colonne) divise la somme par le nombre de valeurs non NULL, pas par le nombre total de lignes

```
-- Dans cet exemple, si certains produits ont un prix NULL,
-- ils seront ignorés dans le calcul de la moyenne
SELECT AVG(prix) FROM Produits;
```

Si vous souhaitez inclure les valeurs NULL en les considérant comme zéro, vous pouvez utiliser la fonction COALESCE ou IFNULL (selon le SGBD) :

```
-- Considérer les prix NULL comme zéro
SELECT AVG(COALESCE(prix, 0)) FROM Produits;
```

4. Les fonctions MIN et MAX

4.1 La fonction MIN

MIN retourne la plus petite valeur non NULL d'une colonne :

```
-- Prix du produit le moins cher
SELECT MIN(prix) FROM Produits;

-- Date de la commande la plus ancienne
SELECT MIN(date_commande) FROM Commandes;
```

```
-- Prix minimum par catégorie
SELECT categorie_id, MIN(prix)
FROM Produits
GROUP BY categorie_id;
```

4.2 La fonction MAX

MAX retourne la plus grande valeur non NULL d'une colonne :

```
 -- Prix du produit le plus cher
SELECT MAX(prix) FROM Produits;

-- Date de la commande la plus récente
SELECT MAX(date_commande) FROM Commandes;

-- Montant de la commande la plus importante
SELECT MAX(montant_total) FROM Commandes;
```

4.3 Utilisation de MIN et MAX avec différents types de données

MIN et MAX fonctionnent avec différents types de données :

- **Numériques** : valeur la plus petite/grande
- **Chaînes de caractères** : ordre alphabétique (A-Z pour MIN, Z-A pour MAX)
- **Dates** : date la plus ancienne/récente
- **Booléens** : FALSE pour MIN, TRUE pour MAX

```
 -- Première lettre alphabétique des noms de clients
SELECT MIN(nom) FROM Clients;

-- Client le plus récemment inscrit
SELECT MAX(date_inscription) FROM Clients;
```

4.4 Combiner MIN et MAX

Les fonctions MIN et MAX sont souvent utilisées ensemble pour déterminer l'étendue des valeurs :

```
 -- Plage de prix des produits
SELECT MIN(prix) AS prix_minimum, MAX(prix) AS prix_maximum
FROM Produits;

-- Période couverte par les commandes
SELECT MIN(date_commande) AS premiere_commande,
    MAX(date_commande) AS derniere_commande
FROM Commandes;
```

5. Combinaison des fonctions d'agrégation

5.1 Utiliser plusieurs fonctions d'agrégation dans une même requête

Vous pouvez combiner plusieurs fonctions d'agrégation dans une même requête pour obtenir différentes statistiques en une seule opération :

```
 -- Statistiques générales sur les produits
SELECT COUNT(*) AS nombre_produits,
    AVG(prix) AS prix_moyen,
    MIN(prix) AS prix_minimum,
    MAX(prix) AS prix_maximum,
    SUM(stock) AS stock_total
FROM Produits;

-- Statistiques des commandes
SELECT COUNT(*) AS nombre_commandes,
    SUM(montant_total) AS chiffre_affaires_total,
    AVG(montant_total) AS panier_moyen,
    MIN(montant_total) AS commande_minimum,
    MAX(montant_total) AS commande_maximum
FROM Commandes;
```

5.2 Création d'un tableau de bord basique

En combinant différentes fonctions d'agrégation, vous pouvez créer un tableau de bord simple avec des indicateurs clés de performance :

```
SELECT
  -- Statistiques clients
  (SELECT COUNT(*) FROM Clients) AS nombre_clients,
  (SELECT COUNT(*) FROM Clients WHERE date_inscription >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)) AS nouveaux_clients,

  -- Statistiques produits
  (SELECT COUNT(*) FROM Produits) AS nombre_produits,
  (SELECT COUNT(*) FROM Produits WHERE stock = 0) AS produits_rupture,

  -- Statistiques commandes
  (SELECT COUNT(*) FROM Commandes WHERE date_commande >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)) AS commandes_recentes,
  (SELECT SUM(montant_total) FROM Commandes WHERE date_commande >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)) AS ca_mensuel,

  -- Statistiques générales
  (SELECT AVG(montant_total) FROM Commandes) AS panier_moyen;
```

6. Gestion avancée des valeurs NULL

6.1 Impact des valeurs NULL sur les fonctions d'agrégation

Rappelons que les fonctions d'agrégation (sauf COUNT(*)) ignorent les valeurs NULL :

- **COUNT(*)** : Compte toutes les lignes, qu'elles contiennent des NULL ou non
- **COUNT(colonne)** : Ignore les valeurs NULL
- **SUM**, **AVG**, **MIN**, **MAX** : Ignorent toutes les valeurs NULL

6.2 Solutions pour gérer les valeurs NULL

Selon votre besoin, vous pouvez :

1. **Remplacer NULL par une valeur par défaut** avec COALESCE ou IFNULL :

```
-- Remplacer les prix NULL par zéro
SELECT AVG(COALESCE(prix, 0)) FROM Produits;

-- Considérer les stocks NULL comme zéro pour le total
SELECT SUM(COALESCE(stock, 0)) FROM Produits;
```

2. **Filtrer les NULL** avant l'agrégation :

```
-- Ne calculer la moyenne que pour les produits avec un prix défini
SELECT AVG(prix) FROM Produits WHERE prix IS NOT NULL;
```

3. **Compter séparément les valeurs NULL** :

```
-- Compter combien de produits ont un prix NULL
SELECT COUNT(*) - COUNT(prix) AS produits_sans_prix FROM Produits;
```

7. Application pratique : Analyse des ventes

7.1 Indicateurs clés de performance (KPI)

Les fonctions d'agrégation sont essentielles pour calculer les KPI d'une entreprise :

```
-- Chiffre d'affaires total
SELECT SUM(montant_total) AS chiffre_affaires
FROM Commandes;

-- Chiffre d'affaires par mois
SELECT
    YEAR(date_commande) AS annee,
    MONTH(date_commande) AS mois,
    SUM(montant_total) AS chiffre_affaires_mensuel
FROM Commandes
GROUP BY YEAR(date_commande), MONTH(date_commande)
ORDER BY annee, mois;

-- Nombre moyen de commandes par client
SELECT AVG(nb_commandes) AS moyenne_commandes_par_client
FROM (
    SELECT client_id, COUNT(*) AS nb_commandes
    FROM Commandes
    GROUP BY client_id
) AS commandes_par_client;
```

7.2 Analyse comparative

Les fonctions d'agrégation permettent également de réaliser des analyses comparatives :

```
-- Comparaison du chiffre d'affaires actuel vs même période année précédente
SELECT
    SUM(CASE WHEN date_commande >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY) THEN montant_total ELSE 0 END) AS ca_mois_courant,
    SUM(CASE WHEN date_commande BETWEEN DATE_SUB(DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY), INTERVAL 1 YEAR)
            AND DATE_SUB(CURRENT_DATE, INTERVAL 1 YEAR) THEN montant_total ELSE 0 END) AS ca_meme_periode_an_dernier
```

```
FROM Commandes;

-- Comparaison des prix moyens par catégorie
SELECT
    c.nom_categorie,
    AVG(p.prix) AS prix_moyen,
    (SELECT AVG(prix) FROM Produits) AS prix_moyen_global,
    AVG(p.prix) - (SELECT AVG(prix) FROM Produits) AS difference
FROM Produits p
JOIN Categories c ON p.categorie_id = c.categorie_id
GROUP BY c.nom_categorie;
```

Prochaine étape

Félicitations ! Vous avez maintenant terminé les cinq modules du niveau 1 SQL. Vous maîtrisez les concepts fondamentaux pour interroger et analyser des données avec SQL. La prochaine étape sera d'aborder le niveau 2, qui couvrira des concepts plus avancés comme les jointures, les sous-requêtes, les clauses GROUP BY et HAVING, ainsi que la création et modification de tables.