Fonctionnement et cas d'usage de la fonction d'agrégation COUNT.

La fonction d'agrégation *COUNT* est très utile si on souhaite connaître le nombre d'entrées dans une table, le nombre d'éléments non-nuls dans une colonne ou filtrer les groupes en fonction d'un nombre d'éléments pour un attribut donné. Pour l'exprimer différemment, la clause **COUNT** peut être utilisée avec différentes clauses comme **SELECT**, **HAVING**, ou **GROUP BY** pour affiner les résultats et obtenir des comptages précis selon divers critères.

La fonction COUNT permet de répondre à des questions courantes sur les données, telles que :

- Combien de produits ont été vendus ?
- Combien de produits ont un prix non nul ?
- Quel est le nombre total de produits par catégorie de prix ?

Pour répondre à ces questions, la fonction *COUNT* fournit diverses possibilités :

1. COUNT(*) : compte le nombre total de lignes dans une table, y compris celles qui contiennent des valeurs NULL.
2. COUNT(colonne) : compte le nombre de valeurs non-nulles dans une colonne spécifique.
3. COUNT(DISTINCT colonne) : compte le nombre de valeurs uniques dans une colonne, en excluant les doublons.

Afin d'illustrer les exemples d'usage de cette fonction nous utilisons la base de données '*sales*'. Elle contient 7 entrées et 4 attributs : l'identifiant unique du produit, son nom, son prix et le nbr d'unités vendues.

```
SELECT * FROM sales ;
```

```
 product_id | product_name | price | nbr_order
------------+--------------+-------+-----------
          1 | Jeans        |    15 |         8
          2 | Flipflop     |   8.5 |        45
          3 | Casquette    |    32 |         2
          4 | Chemise      | 29.99 |
          5 | Foulard      | 15.75 |        45
          6 | Veste        | 32.95 |        34
          7 | Jeans        |  49.9 |
(7 lignes)
```

Fonctionnement et usage de COUNT(*).

Tout d'abord nous allons introduire la fonction **COUNT(*)**. Cette première utilisation de la fonction **COUNT** renvoie le **nombre d'entrée dans votre table** et ce, quelle que soit la valeur prise par les attributs de votre table. Ainsi, **même si vous avez des valeurs nulles** dans votre table celles-ci seront prise en compte. Pour cause, **la fonction COUNT(*) va attribuer un astérisque à chaque entrée du tableau puis compter le nombre d'astérisques**.

Donc contrairement à ce que l'on croît, la fonction COUNT(*) ne parcours pas l'intégralité de la table pour connaitre le nombre d'entrées et elle ne diffère en rien de **COUNT(1) ou même COUNT(-38)**. En effet, elle attribut simplement la valeur spécifié entre parenthèse à chaque entrée du tableau puis compte le nombre d'occurence.

Voyez plutôt.

```
SELECT
    COUNT(*) AS nbr_lignes
    , COUNT(1) AS nbr_lignes_1
    , COUNT(-38) AS nbr_lignes_38
FROM sales ;
```

```
 nbr_lignes | nbr_lignes_1 | nbr_lignes_38
------------+--------------+---------------
          7 |            7 |             7
(1 ligne)
```

Nous avons bien sept lignes dans notre table. Le compte est bon, quel que soit la valeur spécifiée entre parenthèses.

Fonctionnement et usage de COUNT(nom_colonne).

Cette fois_ci nous allons nous intéresser au fonctionnement et à l'usage de la fonction **COUNT** sur un attribut spécifique d'une table. La fonction d'agrégation va parcourir l'attribut de la table et comptabiliser le nombre d'entrées non nulles. Voyez plutôt en pratique.

```
SELECT
    COUNT(*) AS nbr_produits
    , COUNT(nbr_order) AS nbr_produits_commandés
FROM sales ;
```

```
 nbr_produits | nbr_produits_commandés
--------------+------------------------
           7  |                      5
(1 ligne)
```

Le résultat concorde avec notre table puisque la variable "nbr_order" contient 2 valeurs nulles - i.e. deux produits ne font partie d'aucune commande.

Fonctionnement et usage de COUNT(DISTINCT nom_colonne).

Il est courant de vouloir comptabiliser seulement le nombre d'entrée distincte pour obtenir, par exemple, le **nombre de produits uniques commandés**. Pour cela on peut préciser l'argument **DISTINCT** (au sein de la fonction **COUNT**) avant le nom de la colonne d'intérêt.

```
SELECT
    COUNT(*) AS nbr_produits
    , COUNT(nbr_order) AS nbr_produits_commandés
    , COUNT(DISTINCT product_name) AS nbr_unique_produits_commandés
FROM sales ;
```

```
 nbr_produits | nbr_produits_commandés | nbr_unique_produits_commandés
--------------+------------------------+-------------------------------
            7 |                      5 |                             6
(1 ligne)
```

Nous avons bien 6 produits distintcs commandés : notre magasin dispose de deux types de jeans.

Jusqu'à présent nous nous sommes contenté d'évoquer les cas d'usage de la fonction *COUNT* au sein de la clause **SELECT**. Toutefois, comme toutes les fonctions d'agrégation vous pouvez l'utiliser avec les clauses **HAVING** et **ORDER BY**. Egalement, cette fonction d'agrégation s'avère très utile en complément de la clause **GROUP BY**, voyons cela !

Cas d'usage GROUP BY - COUNT.

Parfois on veut dénombrer les occurences d'un sous-échantillon. Pour cela on peut utiliser la clause **GROUP BY** et **COUNT(*)** dans une même requête. Nous allons maintenant utiliser la table **employees**. Supposons qu'on souhaite connaitre le nombre d'employée par département.

Tout d'baord, voici la composition de notre table **employees** :

```
SELECT *
FROM employees ;
```

```
 employee_id |  employee_name  | department_id | manager_id | salary
-------------+-----------------+---------------+------------+--------
           1 | John Dupont     |             1 |          3 |   5000
           2 | Anthony Smith   |             2 |          3 |   6000
           3 | Michelle Paris  |             1 |          4 |   4500
           4 | Emily Blanc     |             3 |            |   5500
           5 | Camille Johnson |             2 |          4 |   5200
(5 lignes)
```

Lançons la requête suivante pour connaître le nombre d'employés par département.

```sql
SELECT
    department_id
    , COUNT(*) AS nbr_employees_per_department
FROM employees
GROUP BY department_id
ORDER BY department_id ;
```

```
 department_id | nbr_employees_per_department
---------------+------------------------------
             1 |                            2
             2 |                            2
             3 |                            1
(3 lignes)
```

Les departements 1, 2 et 3 ont donc, respectivement, 2, 2 et 1 employés.

Tout ça c'est bien, révisons un peu !

> Supposons maintenant qu'on souhaite connaître le nombre de relations de subordination et le nombre de manager par département. Comment allez vous utiliser la fonction **COUNT** ?

On va chercher le nombre de relations de subordination unique dans chaque département.

```sql
SELECT
    department_id
    , COUNT(DISTINCT manager_id) AS nbr_relation_subordination
FROM employees
GROUP BY department_id ;
```

```
 department_id | nbr_relation_subordination
---------------+----------------------------
             1 |                          2
             2 |                          2
```

```
             3 |                           0
(3 lignes)
```

Le département 3 ne connait aucune relation de subordination. Vous observez d'ailleurs que la fonction **COUNT** ne prend pas en compte les valeurs nulles et renvoie donc zéro.

Maintenant, on va chercher le nombre de manager dans chaque département. Tout d'abord il faut utiliser une jointure sur la table elle-même (self-join) afin d'obtenir pour chaque employé le nom de son manager et son département d'exercice. Finalement, on dénombre le nombre unique de manager par département. Pour imbriquer l'ensemble de ces opérations, on utilise deux CTEs.

```
WITH cte1 AS (
    SELECT
         emp.employee_name AS employee
         , mng.employee_name AS manager
         , mng.department_id AS department
    FROM employees AS emp
    JOIN employees AS mng
         ON emp.manager_id = mng.employee_id
),

cte2 AS(
    SELECT
        department,
        COUNT(DISTINCT manager) AS nbr_manager
    FROM cte1
    GROUP BY department
    ORDER BY department ASC
)

SELECT
    department
    , nbr_manager
FROM cte2 ;
```

```
 department  | nbr_manager
-------------+-------------
           1 |           1
           3 |           1
(3 lignes)
```

Notez que la jointure permet de conserver seulement les entrées qui ont au moins une relation de subordination ; le nom de l'employé, de son manager et le département d'exercice du manager sont conservés. Puis la seconde CTE renvoie le nombre unique de manager par département. Enfin, le résultat est extrait de cette deuxième CTE.

Cas d'usage HAVING - COUNT.

Lorsqu'on regroupe des données comme on vient de le faire, la clause **HAVING** peu nous permettre de filtrer les groupes de données. Par exemple, on veut l'identifiant des départements qui ont au moins une relation de subordination.

```
SELECT
    department_id AS dpt_id_with_atleast_one_manager
FROM employees
GROUP BY department_id
HAVING COUNT(manager_id) > 0
ORDER BY department_id ;
```

```
 dpt_id_with_atleast_one_manager
---------------------------------
                               1
                               2
(2 lignes)
```

Effectivement, le département 3 ne contient aucun manager puisqu'il s'agit du département d'activité de la présidente de l'entreprise.

Cas d'usage ORDER BY - COUNT.

Enfin, notez qu'une fonction d'agrégation peut être utilisée avec la clause **ORDER BY**. Dans l'exemple qui suit, on va ordonner le résultats en fonction du nombre d'employés par département.

```
SELECT
    department_id
FROM employees
GROUP BY department_id
ORDER BY COUNT(employee_name) ;
```

```
 department_id
---------------
             3
             2
             1
(3 lignes)
```

Puisque les départements 1 et 2 ont chacun deux employés, ils apparaissent en dernier. Pourquoi apparaissent-ils en dernier puisqu'ils ont plus d'employés que le département 3 ? Parce que, par défaut, l'ordonnement ce fait de façon croissante (*ASC*). Si vous souhaitez obtenir ces départements en premier, vous devez spécifier l'option "DESC".

```
SELECT
    department_id
FROM employees
GROUP BY department_id
ORDER BY COUNT(employee_name) DESC ;
```

```
 department_id
---------------
             2
             1
             3
(3 lignes)
```

Cas d'usage avancé.

Dans cette dernière partie j'aimerai introduire un cas d'usage avancé de la fonction **COUNT** : l'emploi d'un filtre (d'une condition) au sein de la fonction. Notamment, il est possible d'intégrer la clause **CASE WHEN** dans la fonction afin de filtrer les résultats et compter le nombre d'occurences qui satisfont à la condition.

Dans ce qui suit, on souhaite trouver le nombre de salariés par département dont le salaire mensuel excède 5 000€.

```
SELECT
    department_id
    , COUNT(
        CASE
         WHEN salary > 5000 THEN 1
        END
        ) AS nbr_expensive_salary
FROM employees
GROUP BY department_id ;
```

```
 department_id | nbr_expensive_salary
---------------+----------------------
             3 |                    1
             2 |                    2
             1 |                    0
(3 lignes)
```

Avant de finir, un petit exercice !

> Trouvez le nombre de département dont la masse salariale excède 9'000€ par mois. Cette requête utilise une CTE ainsi que deux fonctions d'agrégation **SUM** et **COUNT**. Si vous souhaitez en savoir plus sur les CTEs, c'est par ici.

```
WITH ma_ctes AS (
SELECT
    department_id
    , SUM(salary) AS total_salary_per_dprt
FROM employees
GROUP BY department_id
)

SELECT
    COUNT(
         CASE
          WHEN total_salary_per_dprt > 9000 THEN 1
         END
         ) AS nbr_expensive_dprt
FROM ma_ctes ;
```

```
 nbr_expensive_dprt
--------------------
                  2
(1 ligne)
```

On a donc 2 départements dont la masse salariale semble excessive.

J'espère que cette note sur l'usage de la fonction d'agrégation **COUNT** vous à été utile. Pour une question ou un commentaire, écrivez-moi à statisserie@gmail.com !