BASE RQT SQL

SELECT : Cette requête est utile lorsque vous souhaitez obtenir rapidement toutes les colonnes d'un tableau sans écrire chaque colonne dans l' instruction SELECT.

Code : SELECT *

FROM employees;

Explication

Lorsque vous souhaitez sélectionner un nombre quelconque de colonnes dans une table, vous devez utiliser l'instruction SELECT. Vous l'écrivez, de manière assez évidente, en utilisant le mot-clé SELECT.

Ce mot-clé est suivi d'un astérisque (*), qui signifie "toutes les colonnes de la table".

Pour spécifier la table, utilisez la clause FROM et écrivez le nom de la table à la suite.

2. Sélection d'une colonne d'une table

Vous pouvez utiliser cette requête lorsque vous n'avez besoin que d'une seule colonne du tableau.

Code : SELECT first_name

FROM employees;

Explication

L'approche est similaire à la requête précédente. Cependant, cette fois-ci, au lieu d'un astérisque, nous écrivons le nom de la colonne spécifique dans SELECT. Dans ce cas, il s'agit de la colonne first_name.

La deuxième ligne de la requête est la même : elle fait référence à la table dans la clause FROM.

Résultat

La requête renvoie la liste des prénoms des employés.

3. Sélection de deux colonnes dans un tableau

Cette requête est utile pour sélectionner deux colonnes (ou plus) dans un tableau.

Code :

```
SELECT first_name,
    last_name
FROM employees;
```

Explication

L'approche est similaire à celle des exemples précédents. Pour sélectionner deux colonnes, vous devez écrire leur nom à l'adresse SELECT. L'important est que les colonnes soient séparées par une virgule. Vous pouvez voir dans l'exemple qu'il y a une virgule entre les colonnes first_name et last_name.

Ensuite, comme d'habitude, faites référence au tableau employees dans FROM.

Résultat

La requête affiche maintenant les noms complets des employés.

4. Sélection de deux colonnes (ou plus) d'une table et filtrage à l'aide d'une comparaison numérique dans WHERE

Code :

```
SELECT
 first_name,
 last_name,
 salary
FROM employees
WHERE salary > 3800;
```

Explication

La requête sélectionne en fait trois colonnes, et non deux. C'est la même chose qu'avec deux colonnes : il suffit de les écrire dans SELECT et de les séparer par des virgules.

Nous faisons ensuite référence à la table dans FROM.

Nous devons maintenant afficher uniquement les employés dont le salaire est supérieur à 3 800. Pour ce faire, vous devez utiliser WHERE. Il s'agit d'une clause qui accepte des conditions et qui est utilisée pour filtrer les résultats. Elle parcourt le tableau et renvoie uniquement les données qui satisfont à la condition.

Dans notre cas, nous recherchons des salaires "supérieurs" à un certain nombre. En d'autres termes, une condition utilisant l'opérateur de comparaison >.

Pour définir la condition, nous écrivons le nom de la colonne à l'adresse WHERE. Vient ensuite l'opérateur de comparaison, puis la valeur à laquelle les données doivent être supérieures. Cette condition renverra tous les salaires supérieurs à 3 800.

Résultat

La requête renvoie quatre employés et leurs salaires. Comme vous pouvez le constater, ils ont tous un salaire supérieur à 3 800.

5. Sélection de deux colonnes et filtrage à l'aide d'une condition d'égalité dans WHERE

Une fois de plus, cet exemple de requête SQL de base est utile lorsque vous souhaitez sélectionner plusieurs colonnes, mais pas toutes les lignes du tableau. Vous voulez maintenant trouver les valeurs qui sont identiques à la valeur de la condition. Pour cela, vous avez besoin de la condition d'égalité (=).

Code:

```
SELECT
  first_name,
  last_name
FROM employees
WHERE first_name = 'Luca';
```

Explication

La requête sélectionne les noms et prénoms des employés.

Cependant, nous voulons afficher uniquement les employés dont le nom est Luca. Pour cela, nous utilisons à nouveau WHERE. L'approche est similaire à celle de l'exemple précédent : nous utilisons WHERE, écrivons le nom de la colonne et utilisons l'opérateur de comparaison. Cette fois, notre condition utilise le signe égal (=).

En d'autres termes, les valeurs de la colonne first_name doivent être égales à Luca. De plus, lorsque la condition n'est pas un nombre mais un texte ou une date/heure, elle doit être écrite entre guillemets simples (''). C'est pourquoi notre condition s'écrit 'Luca', et non simplement Luca.

Sortie

La sortie montre qu'il n'y a qu'un seul employé nommé Luca, et que son nom complet est Luca Pavarotti.

6. Sélection de deux colonnes et classement par une colonne

Voici un autre exemple de requête SQL de base qui vous sera utile. Il peut être utilisé chaque fois que vous devez ordonner le résultat d'une certaine manière pour le rendre plus lisible.

L'ordre ou le tri des résultats est effectué à l'aide de la clause ORDER BY. Par défaut, elle ordonne les résultats par ordre croissant, par ordre alphabétique (pour les données textuelles), du plus petit au plus grand nombre (pour les données numériques), ou de la date ou de l'heure la plus ancienne à la plus récente (pour les dates et les heures).

Code :

```
SELECT
 first_name,
 last_name
FROM employees
ORDER BY last_name;
```

Explication

Nous sélectionnons à nouveau le nom et le prénom des employés. Mais nous voulons maintenant trier les résultats d'une manière spécifique. Dans cet exemple, il s'agit du nom de famille des employés. Pour ce faire, nous utilisons ORDER BY. Nous y inscrivons simplement le nom de la colonne.

Nous pouvons ajouter le mot-clé ASC pour trier le résultat de manière ascendante. Cependant, ce n'est pas obligatoire, car le tri ascendant est une valeur par défaut de SQL.

Résultat

La requête renvoie une liste d'employés classés par ordre alphabétique de leur nom de famille.

DESC pour inverse

9. Sélection de deux colonnes avec une condition logique complexe dans WHERE

Cet exemple montre à nouveau comment filtrer les résultats à l'aide de WHERE. Il sera un peu plus avancé cette fois-ci, car nous utiliserons un opérateur logique. En SQL, les opérateurs logiques vous permettent de tester si la condition de filtrage est vraie ou non. Ils permettent également de définir des conditions multiples.

Les trois opérateurs logiques de base en SQL sont AND, OR et NOT. Dans la requête ci-dessous, nous utiliserons OR pour obtenir les salaires inférieurs à 3 000 ou supérieurs à 5 000.

Code :

```
SELECT
 first_name,
```

```
 last_name,
 salary
FROM employees
WHERE salary > 5000 OR salary < 3000;
```

10. Calculs simples sur les colonnes

Dans cet exemple, nous allons montrer comment vous pouvez effectuer des opérations mathématiques simples sur les colonnes de la table.

Nous utiliserons l'un des opérateurs arithmétiques de SQL.

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo, i.e. returns the remainder of the integer division.

Code :

```
SELECT
 employee_id,
 q1_2022 + q2_2022 AS h1_2022
FROM quarterly_sales;
```

Explication

Dans la requête ci-dessus, nous voulons trouver les ventes du premier semestre 2022 pour chaque employé.

Pour ce faire, nous sélectionnons d'abord la colonne employee_id dans la table quarterly_sales.

Ensuite, nous sélectionnons la colonne q1_2022 et utilisons l'opérateur arithmétique d'addition pour ajouter la colonne q2_2022. Nous donnons également à cette nouvelle colonne calculée un alias de h1_2022 en utilisant le mot-clé AS.

11. Utilisation de SUM() et GROUP BY

Cette requête utilise la fonction d'agrégation SUM() avec GROUP BY. En SQL, les fonctions d'agrégation fonctionnent sur des groupes de données ; par exemple, SUM(sales) affiche le total de

toutes les valeurs de la colonne sales. Il est utile de connaître cette fonction lorsque vous voulez placer des données dans des groupes et afficher le total pour chaque groupe.

Code :

```
SELECT

department,

SUM(salary) AS total_salaries

FROM employees

GROUP BY department;
```

Explication

L'objectif de la requête ci-dessus est de trouver le montant total des salaires pour chaque département. Cet objectif est atteint de la manière suivante.

Tout d'abord, sélectionnez la colonne département dans le tableau employees. Ensuite, utilisez la fonction SUM(). Comme nous voulons additionner les valeurs salariales, nous spécifions la colonne salaire dans la fonction. De plus, nous donnons à cette colonne calculée l'alias total_salaries.

Enfin, la sortie est groupée par la colonne département.

Remarque : toute colonne non agrégée apparaissant dans SELECT doit également apparaître dans GROUP BY. Mais c'est logique - l'objectif est de regrouper les données par département, donc nous la placerons bien sûr dans GROUP BY.

12. Utilisation de COUNT() et GROUP BY

Voici une autre requête SQL de base qui utilise une fonction d'agrégation. Cette fois, il s'agit de COUNT(). Vous pouvez l'utiliser si vous souhaitez regrouper des données et afficher le nombre d'occurrences dans chaque groupe.

Code :

```
SELECT

department,

COUNT(*) AS employees_by_department

FROM employees

GROUP BY department;
```

Explication

Nous voulons afficher le nombre d'employés par département.

Sélectionnez le département dans le tableau employees. Ensuite, utilisez la fonction d'agrégation COUNT(). Dans ce cas, nous utilisons la version COUNT(*), qui compte toutes les lignes. Nous donnons à la colonne l'alias employees_by_department.

Enfin, nous regroupons les résultats par département.

Note : COUNT(*) compte toutes les lignes, y compris celles avec les valeurs NULL. Si vous ne souhaitez pas inclure les éventuelles valeurs NULL dans votre résultat, utilisez la version COUNT(column_name) de la fonction. Nous pouvons utiliser COUNT(*) ici car nous savons qu'il n'y a pas de valeurs NULL dans le tableau.