SELECT

L'instruction SELECT est souvent décrite comme la commande SQL la plus importante. La plupart des requêtes que vous écrirez en tant qu'étudiant ou spécialiste SQL commenceront par cette commande.

`SELECT` L'instruction SELECT est utilisée pour extraire des données d'une base de données. La syntaxe de l'instruction `SELECT` est la suivante :

Code

```
SELECT column1, column2, ...
FROM table_name;
```

Ici, `column1`, `column2`, ... sont les noms des colonnes dont vous voulez extraire les données. `table_name` est le nom de la table dont vous voulez extraire les données.

Vous pouvez également utiliser le symbole * pour récupérer les données de toutes les colonnes d'une table. Par exemple, si vous souhaitez récupérer toutes les données d'une table nommée `customers` vous devez utiliser l'instruction SQL suivante :

Code

```
SELECT *
FROM customers;
```

Pour voir d'autres exemples de `SELECT`, consultez notre article Comment écrire une instruction SELECT en SQL ?

La meilleure façon d'apprendre SQL est de pratiquer. Essayez notre cours interactif SQL pour les débutants.

OÙ

Cette commande SQL permet de filtrer les données sélectionnées. La clause `WHERE` suit la clause FROM dans une instruction `SELECT`, `UPDATE` ou `DELETE`. Elle spécifie une ou plusieurs conditions qui doivent être remplies pour que l'instruction s'exécute. Elle est souvent accompagnée d'un ou plusieurs opérateurs logiques ou opérateurs de comparaison.

La syntaxe de base d'une clause `WHERE` est la suivante :

Code

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Dans cette syntaxe, `column1`, `column2`, `...` sont les noms des colonnes qui stockent les données souhaitées ; `table_name` est le nom de la table qui contient les données. Le paramètre `condition` spécifie la condition qui doit être remplie pour que l'instruction soit exécutée.

Le paramètre condition peut inclure un ou plusieurs opérateurs logiques, tels que `AND`, `OR`, et `NOT` ; il peut également comporter un ou plusieurs opérateurs de comparaison, tels que is equal to (=), does not equal (<>), less than (<), greater than (>), less than or equal to (<=), et great than or equal to (>=). Vous pouvez également utiliser des fonctions et des sous-requêtes pour créer des conditions plus complexes.

Prenons un exemple simple. Supposons que vous disposiez d'un tableau nommé `employees` avec les colonnes `id`, `name`, `age` et `department`. Vous souhaitez récupérer les informations relatives à tous les employés âgés de moins de 30 ans. Pour ce faire, vous pouvez utiliser l'instruction SQL suivante :

Code

```
SELECT *
FROM employees
WHERE age < 30;
```

Dans cet exemple, la clause `WHERE` est utilisée pour spécifier la condition selon laquelle seuls les employés dont l'âge est inférieur à 30 ans seront inclus dans l'ensemble de résultats. Cela signifie que le résultat de cette requête sera constitué

de toutes les lignes de la table `employees` où la valeur de la colonne âge est inférieure à 30.

La maîtrise de la commande WHERE est essentielle pour tous ceux qui souhaitent utiliser SQL. Notre guide complet de la commande WHERE contient plus d'informations et d'exemples.

INSERT

L'instruction INSERT est l'une des commandes SQL qui permettent de modifier les données des tables de la base de données ; elle ajoute de nouvelles données à une table. La syntaxe de l'instruction `INSERT` est la suivante :

Code

```
INSERT INTO table_name (column1, column2, ... )
VALUES (value1, value2, ... );
```

Ici, `table_name` est le nom de la table, `column1`, `column2`, ... sont les noms des colonnes auxquelles vous souhaitez ajouter des données, et `value1`, `value2`, ... sont les valeurs que vous souhaitez ajouter à ces colonnes.

Par exemple, si vous souhaitez ajouter un nouveau client à la table `customers` vous utiliserez l'instruction SQL suivante :

Code

```
INSERT INTO customers (customer_name, customer_email, customer_phone)
VALUES ('John Doe', 'john.doe@example.com', '123-456-7890');
```

Pour en savoir plus sur cette commande SQL, consultez l'article Qu'est-ce que l'instruction INSERT en SQL ?

UPDATE

L'instruction `UPDATE` est utilisée pour modifier des données existantes dans une base de données. La syntaxe de l'instruction `UPDATE` est la suivante :

Code

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Ici, `table_name` est le nom de la table. Ensuite, `column1`, `column2`, ... sont les noms des colonnes que vous souhaitez modifier, et `value1`, `value2`, ... sont les nouvelles valeurs que vous souhaitez définir pour ces colonnes. Enfin, `condition` est le critère de sélection des lignes à modifier.

Par exemple, si vous souhaitez mettre à jour l'adresse électronique d'un client portant le nom "John Doe" dans la table `customers` vous utiliserez l'instruction SQL suivante :

Code

```
UPDATE customers
SET customer_email = 'johndoe@example.com'
WHERE customer_name = 'John Doe';
```

DELETE

L'instruction `DELETE` est utilisée pour supprimer des données d'une base de données. La syntaxe de l'instruction `DELETE` est la suivante :

Code

```
DELETE FROM table_name
WHERE condition;
```

Ici, `table_name` est le nom de la table dont vous souhaitez supprimer les données et `condition` est le critère de sélection des lignes à supprimer.

Par exemple, si vous souhaitez supprimer le client portant le nom "John Doe" de la table `customers` vous utiliserez l'instruction SQL suivante :

Code

```
DELETE FROM customers
WHERE customer_name = 'John Doe';
```

Il existe deux autres commandes SQL qui effectuent ce qui, à première vue, semble être un travail similaire : `TRUNCATE` et `DROP`. On vous demandera souvent quelles sont les différences entre ces commandes lors d'entretiens d'embauche en SQL. Heureusement, nous avons un article pour cela : TRUNCATE TABLE vs. DELETE vs. DROP TABLE : Supprimer des tables et des données en SQL.

ORDER BY

La clause `ORDER BY` est utilisée pour trier l'ensemble des résultats d'une instruction `SELECT` dans l'ordre croissant (A-Z, 1-10) ou décroissant (Z-A, 10-1). La syntaxe de la clause `ORDER BY` est la suivante :

Code

```
SELECT column1, column2, ...
FROM table_name O
ORDER BY column_name ASC|DESC;
```

Ici, `column_name` est le nom de la colonne sur laquelle vous souhaitez trier l'ensemble de résultats et `ASC` ou `DESC` indique si l'ordre de tri doit être croissant

ou décroissant. Si l'ordre n'est pas spécifié (ni `ASC` ni `DESC` ne sont écrits), l'ordre est défini par défaut comme étant croissant.

Par exemple, si vous souhaitez récupérer tous les clients d'une table nommée `customers` et les trier par leur nom dans l'ordre croissant, vous utiliserez l'instruction SQL suivante :

Code

```
SELECT * FROM customers
ORDER BY customer_name ASC;
```

Cette commande SQL permet d'autres astuces, comme le tri sur plusieurs colonnes. Si vous souhaitez en savoir plus, notre Guide détaillé de ORDER BY a été écrit spécialement pour vous.

GROUP BY

La clause GROUP BY est utilisée pour regrouper les lignes qui ont les mêmes valeurs dans une colonne donnée. Vous utiliserez souvent `GROUP BY` lorsque la tâche est du type "trouver le prix moyen par catégorie de produit". La syntaxe de la clause `GROUP BY` est la suivante :

Code

```
SELECT column1, column2, ...
FROM table_name
GROUP BY column_name;
```

Ici, `column_name` est le nom de la colonne par laquelle vous souhaitez effectuer un regroupement.

Pour obtenir le prix moyen par catégorie de produit à partir de la table `products` vous utiliserez l'instruction SQL suivante :

Code

```
SELECT category, AVG(price)
FROM products
```

```
GROUP BY category;
```

Pour une explication détaillée et d'autres exemples, lisez notre article Qu'est-ce que le GROUP BY en SQL ?

JOIN

La clause JOIN est utilisée pour combiner les lignes de deux tables ou plus en fonction des valeurs correspondantes dans une colonne donnée. Il existe différents types de clauses `JOIN` en SQL, notamment `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` et `FULL OUTER JOIN`. La syntaxe de la clause `INNER JOIN` est la suivante :

Code

```
SELECT column1, column2, ...
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

Ici, `table1` et `table2` sont les noms des tables que vous souhaitez joindre, et `column_name` est le nom de la colonne utilisée pour relier les deux tables.

Par exemple, si vous souhaitez récupérer toutes les commandes avec leurs noms de clients correspondants dans deux tables nommées `orders` et `customers` vous utiliserez l'instruction SQL suivante :

Code

```
SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers
ON orders.customer_id = customers.customer_id;
```

Cette requête consulte la colonne `customer_id` dans la table `orders` et la colonne `customer_id` dans clients. Si ces deux colonnes ont des valeurs correspondantes, la requête renverra l'ID de la commande de la table `orders` ainsi que l'adresse `customer_name` de la table `customers` de la table. La

concordance des identifiants des clients signifie que c'est ce client qui a passé cette commande.

JOIN est l'une des commandes SQL les plus compliquées et les plus diverses. Pour en savoir plus, consultez notre article Quels sont les différents types de JOIN SQL ?

Découvrez les différents types de jointures. Consultez notre cours interactif sur les Les jointures en SQL.

CREATE

L'instruction `CREATE` est utilisée pour créer un nouvel objet de base de données, tel qu'une table, une vue ou un index. La syntaxe de l'instruction `CREATE` varie en fonction du type d'objet que vous souhaitez créer. Voici un exemple de création d'une nouvelle table :

Code

```
CREATE TABLE table_name (
  column1 datatype,
  column2 datatype,
  column3 datatype,
   ...
);
```

Ici, `table_name` est le nom de la table que vous souhaitez créer, et `column1`, `column2`, `column3`, ... sont les noms de ces colonnes. Le type de données que chaque colonne stocke (texte, nombres entiers, nombres décimaux, etc.) est indiqué par `datatype`.

Par exemple, si vous souhaitez créer une nouvelle table `products` avec des colonnes pour l'ID du produit, le nom du produit et le prix, vous utiliserez l'instruction SQL suivante :

Code

```
CREATE TABLE products (
  product_id INT PRIMARY KEY,
  product_name VARCHAR(50),
  price DECIMAL(10, 2)
```

```
);
```

Pour en savoir plus sur `PRIMARY KEY` et sur d'autres aspects de la création d'une table, consultez notre article Comment créer votre première table en SQL.

ALTER

L'instruction ALTER est utilisée pour modifier la structure d'un objet de base de données existant, tel qu'une table ou une vue. La syntaxe de l'instruction `ALTER` varie en fonction du type d'objet à modifier. Voici un exemple d'ajout d'une nouvelle colonne à une table existante:

Code

```
ALTER TABLE table_name
ADD column_name datatype;
```

Ici, `table_name` est le nom de la table que vous souhaitez modifier, `column_name` est le nom de la nouvelle colonne et `datatype` est le type de données de la nouvelle colonne.

Par exemple, si vous souhaitez ajouter une nouvelle colonne nommée `description` à une table existante nommée `products` vous utiliserez l'instruction SQL suivante :

Code

```
ALTER TABLE products
ADD description VARCHAR(100);
```

Un autre exemple d'utilisation de l'instruction `ALTER` consiste à modifier le type de données ou la taille d'une colonne existante. Par exemple, si vous souhaitez modifier le type de données de la colonne `price` dans la table `products` de `DECIMAL(10,2)` à `DECIMAL(12,2)`, vous devez utiliser l'instruction SQL suivante :

Code