SELECT

SELECT… FROM…

Sert à extraire des données stockées dans des colonnes issues de tables de la base de données.

Syntaxe :

```
SELECT nom, prenom, concat(nom, ' ', prenom), salaire * 2
FROM personnes;
```

Personnaliser les requêtes

Alias de colonne

Permet de renommer une colonne pour le traitement et l'affichage de la colonne.

Syntaxe :

```
SELECT nom n, prenom AS "Prenom pers"
FROM personnes AS p;
```

Eliminer les doublons

Le mot-clé distinct permet d'éliminer les doublons.

Syntaxe :

```
SELECT DISTINCT nom
FROM personnes;
```

La clause WHERE

La clause WHERE permet de restreindre la requête aux enregistrements qui respectent sa ou ses conditions.

Syntaxe :

```
SELECT nom, prenom
FROM personnes
WHERE service = 'machin';
```

Les chaines sont sensibles à la casse, les dates au format et doivent être placé entre simples quottes.

La clause ORDER BY

La clause ORDER BY permet de classer l'affichage des enregistrements par ordre croissant ou décroissant. Par défaut l'ordre est décroissant.

Syntaxe :

```
SELECT colonne1
FROM table1
ORDER BY [colonne | expression] [ASC | DESC]
```

Exemple :

```
SELECT nom, premom
FROM personnes
ORDER BY nom
```

Les enregistrements peuvent être triés sur une colonne qui n'a pas été sélectionné dans la clause SELECT. Les enregistrements peuvent être triés sur plusieurs colonnes.

La clause GROUP BY

La clause GROUP BY permet de diviser les enregistrements d'une table en groupes. Les fonctions de groupe peuvent alors être utilisées pour retourner les informations relatives à chaque groupe.

Note :

- La clause WHERE peut être utilisée pour pré-exclure des enregistrements avant la division en groupes.
- Les colonnes de la clause FROM qui ne sont pas inclues dans une fonction de groupe doivent être présentes dans la clause GROUP BY.
- Les alias de colonne ne peuvent pas être utilisés dans la clause GROUP BY.
- Par défaut, la clause GROUP BY classe les enregistrements par ordre croissant. L'ordre peut-être changé en utilisant la clause ORDER BY.

Syntaxe :

```
SELECT departement, count(nom)
FROM personne
GROUP BY departement
```

La clause HAVING

La clause WHERE restreint les enregistrements alors que HAVING restreint les groupes d'enregistrements.
Comme dans les clauses WHERE et GROUP BY, les alias de colonne ne peuvent pas être utilisés dans la clause HAVING.

Syntaxe :

```
SELECT departement, max(salaire)
FROM personne
GROUP BY departement
HAVING max(salaire) > 3000
```

Les jointures

Les jointures permettent de récupérer les données issues de plusieurs tables.
Il existe quatre types de jointures :

- Equi-jointure (equijoin)
- Non equi-jointure (non-equijoin)
- Jointure externe (outer join)
- Auto jointure (self join)

Le produit cartésien

Un produit cartésien se produit lorsqu'une condition de jointure est omise ou invalide. Tous les enregistrements de la 1ere table sont liés à tous les enregistrements de la seconde table.

L'équi-jointure

Une équi-jointure est utilisé pour afficher des données provenant de plusieurs tables lorsqu'une valeur dans une colonne d'une table correspond directement à une valeur d'une autre colonne dans une autre table.

Ancienne syntaxe :

```
SELECT table1.colonne1, table2.colonne3
FROM table1, table2
WHERE table1.colonne_id = table2.colonne_id
```

Nouvelle syntaxe :

```
SELECT table1.colonne1, table2.colonne3
FROM table1
JOIN table2 ON table1.colonne_id = table2.colonne_id
```

La non équi-jointure

Une condition de non équi-jointure est utilisée lorsque deux tables n'ont pas de colonnes qui correspondent directement. Il s'agit de la même syntaxe que pour la condition équi-jointure.

Ancienne syntaxe :

```
SELECT table1.nom, table2.grade
FROM table1, table2
WHERE table1.salaire BETWEEN table2.salMin AND table2.salMax
```

Nouvelle syntaxe :

```
SELECT table1.nom, table2.grade
FROM table1
JOIN table2 ON table1.salaire BETWEEN table2.salMin
      AND table2.salMax
```

La jointure externe

Une condition de jointure externe est utilisée pour afficher tous les enregistrements incluant ceux qui ne respectent pas la condition de jointure.

Une condition de jointure externe ne peut pas utiliser l'opérateur IN et ne peut pas être liée à une autre condition par l'opérateur OR.

Jointure externe gauche

Permets de récupérer tous les enregistrements de la table de gauche qui ne respecte la condition de jointure. Les champs des colonnes de la table de droite sont misent à NULL.

Ancienne syntaxe :

```
SELECT table1.colonne, table2.colonne
FROM table1, table2
WHERE table1.colonne(+) = table2.colonne;
```

Nouvelle syntaxe :

```
SELECT table1.nom, table2.grade
FROM table1
LEFT JOIN table2 on table1.colonne_id = table2.colonne_id
```

Jointure externe droite

Permets de récupérer tous les enregistrements de la table de droite qui ne respecte la condition de jointure. Les champs des colonnes de la table de gauche sont misent à NULL.

Ancienne syntaxe :

```
SELECT table1.colonne, table2.colonne
FROM table1, table2
WHERE table1.colonne = table2.colonne;(+)
```

Nouvelle syntaxe :

```
SELECT table1.nom, table2.grade
FROM table1
RIGHT JOIN table2 on table1.colonne_id = table2.colonne_id
```

L'auto-jointure

Une condition d'auto-jointure permet de faire une jointure sur deux colonnes appartenant à la même table.

Ancienne syntaxe :

```
SELECT alias1.colonne, alias2.colonne
FROM table1 alias1, table1 alias2
WHERE alias1.colonne_id = alias2.colonne_id;
```

Nouvelle syntaxe :

```
SELECT alias1.colonne, alias2.colonne
FROM table1 alias1
JOIN table1 alias2 on alias1.colonne_id = alias2.colonne_id;
```

Les sous-requêtes

Une sous-requête est une clause SELECT imbriquée dans une clause d'un autre ordre SQL.

Note :

- Une sous-requête doit être mise entre parenthèses.
- Une sous-requête doit être placée du côté droit de l'opérateur de comparaison.
- Une sous-requête ne possède pas de clause ORDER BY.
- Une sous-requête peut-être seulement placée dans les clauses WHERE, HAVING et FROM.

Single-row

Retourne une valeur contenue dans une colonne.

Syntaxe :

```
SELECT nom, job
FROM personne
WHERE job = (
    SELECT job
    FROM personne
    WHERE id_pers = 100);
```

Multiple-Row

Retourne plusieurs valeurs continues dans une colonne. Ne peut-être utilisé qu'avec les opérateurs de comparaison IN, NOT IN, ANY, ALL, BETWEEN.

Syntaxe :

```
SELECT nom, salaire
FROM personne
WHERE salaire IN (
    SELECT MIN(salaire)
    FROM personne
    GROUP BY service);
```

Multiple-column

Retourne les valeurs de plusieurs colonnes.

Syntaxe :

```
SELECT nom
FROM personne
WHERE (date_emb, job) IN (
    SELECT MIN(date_emb), job
    FROM personne
    GROUP BY job);
```

Utilisation dans la clause FROM

Syntaxe :

```
SELECT nom, salaire, service
FROM personne a, (
    SELECT service, AVG(salaire) salavg
```

```
    FROM personne
    GROUP BY deptno ) b
WHERE a.service = b.service;
```

Opérateurs arithmétiques

Opérateur	Description
+	Additionner
-	Soustraire
*	Multiplier
/	diviser

Opérateurs de comparaisons

Opérateur	Significations
=	Egal à
<	Inférieur à
<=	Inférieur ou egal à
>	Supérieur à
>=	Supérieur ou egal à
<>	Différent de

Ces opérateurs ne tiennent pas compte de la valeur NULL.

Opérateur	Signification	Exemple
BETWEEN	Vérifie les enregistrements sur une tranche de valeur	`colonne1 BETWEEN valeur1 AND valeur2`
IN	Vérifie les enregistrements dans une liste de valeur	`colonne1 IN (valeur1, valeur2, valeur3)`

ANY	Compare une valeur à chaque valeur retournée par la sous-requête.
	L'opérateur NOT ne peut être utilisé.
		`sal < ANY (SELECT ...) -- plus petit que la valeur la plus grande`
		`sal > ANY (SELECT ...) -- plus grand que la valeur la plus petite`
		`sal = ANY (SELECT ...) -- équivalent à IN`

ALL	Compare une valeur à toutes les valeurs retournées par la sous-requête. L'opérateur NOT ne peut-être utilisé.	`sal > ALL (SELECT ...) -- plus grand que la valeur la plus grande` `sal < ANY (SELECT ...) -- plus petit que la valeur la plus petite`

LIKE	Fait une recherche de caractères spécifique dans un enregistrement
	Le symbole '_' représente un seul caractère quelconque.
	Le symbole '%' représente une série de caractères quelconque.
	Le mot clé **ESCAPE** sert à rechercher un caractère spécial, comme '_' ou '%'.
		`colonne1 LIKE 'valeur'`
		`nom LIKE 'L%'`
		`nom LIKE 'dupon_'`
		`tva like '%/%' ESCAPE '/'`

IS NULL	Vérifie si les enregistrements sont NULL	`colonne1 IS NULL`
AND	Permet de vérifier plusieurs conditions.	`WHERE condition1` `AND condition2`
OR	Permet de vérifier au moins une des conditions.	`WHERE condition1` `OR condition2`

NOT	Permet d'afficher les enregistrements qui ne vérifient pas la condition.
	L'opérateur **NOT** ne peut être utilisé qu'avec les opérateurs de comparaisons **IN**, **LIKE**, **BETWEEN** et **IS NULL**.
		`WHERE col1 NOT IN (liste_valeur)`
		`WHERE col1 NOT LIKE '%p'`
		`WHERE col1 NOT BETWEEN 2 AND 10`
		`WHERE col1 IS NOT NULL`

Ordres d'évaluation des opérateurs :
En premier : opérateur de comparaison
En seconde : opérateur logique (dans cet ordre : **NOT**, **AND** , **OR**)
Les conditions entre parenthèses sont évaluées en premier.