

Nous allons à présent nous attacher à la syntaxe de l'opération d'interrogation des données

- La sélection (**SELECT**)

Cette instruction peut être la plus complexe de toutes mais si on l'étudie méthodiquement en suivant des mots réservés, la lecture est facilitée.

Les opérateurs UNION, EXCEPT, INTERSECT peuvent être utilisés pour combiner les requêtes.

SELECT

2

JOINTURE
WHERE
GROUP BY
HAVING
ORDER BY

Sous-requêtes

9

A la place d'une expression
Dans une clause WHERE
Dans une Clause WHERE avec une fonction d'agrégation

Documentation officielle

12

SELECT

L'instruction **SELECT** permet de visualiser les données stockées dans les bases, d'effectuer des calculs ou des transformations sur ces données, ou de valoriser des tables à partir d'autres tables.

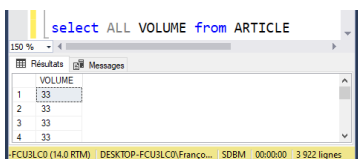
Syntaxe

```
SELECT [ALL|DISTINCT] * | liste_expressions  
FROM tableOuVue  
[JOINTURE]  
[WHERE conditions]  
[GROUP BY liste_expressions]  
[HAVING conditions]  
[ORDER BY liste_expression]
```

ALL / DISTINCT

ALL

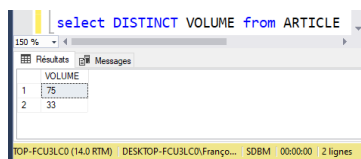
Permet l'extraction de toutes les lignes (option par défaut).



	VOLUME
1	33
2	33
3	33
4	33

DISTINCT

N'affiche pas les doublons, c'est-à-dire les lignes résultantes identiques.



	VOLUME
1	75
2	33

*

Extrait toutes les colonnes.

liste_expressions

Liste composée de noms de colonnes, constantes, fonctions, expressions calculées, de toute combinaison d'expressions séparées par des virgules ou de sous-requête. Chaque expression pourra être complétée par un titre de colonne sous la forme :

- **TITRE** = expression ou
- expression **ALIAS**,

afin de modifier le titre par défaut du résultat qui est, soit nul, soit le nom de la colonne dans la table.

```
select Couleur = NOM_COULEUR from COULEUR
```

	Couleur
1	Blonde
2	Brune
3	Blanche
4	Ambree

```
select NOM_COULEUR Couleur from COULEUR
```

	Couleur
1	Blonde
2	Brune
3	Blanche
4	Ambree

tablesOuVue

Nom de la table ou de la vue sur laquelle porte la requête. Le nom de la table ou de la vue, peut être suivi d'un alias.

```
select C.NOM_CONTINENT from CONTINENT C
```

	NOM_CONTINENT
1	Europe
2	Amerique
3	Océanie
4	Asie

```
select C.NOM_CONTINENT from CONTINENT as C
```

	NOM_CONTINENT
1	Europe
2	Amerique
3	Océanie
4	Asie

JOINTURE

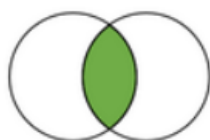
Une jointure permet d'extraire des données de deux ou de plusieurs tables en fonction des relations logiques existant entre ces tables. Les jointures indiquent comment SQL Server doit utiliser les données d'une table pour sélectionner les lignes d'une autre table.

Sa syntaxe sera :

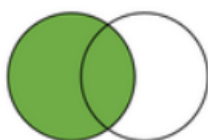
SELECT liste_expressions **FROM** tableA

JOINTURE tableB **ON** colonneTableA = colonneTableB

Ou JOINTURE fait référence à l'un des types de jointures suivant :



Interne



Externe Gauche



Externe droite



Externe Entière

Jointure Interne

Une jointure interne est définie grâce à l'instruction "**INNER JOIN**" ou encore "**JOIN**"

```
select A.ID_ARTICLE, a.NOM_ARTICLE, C.NOM_COULEUR from ARTICLE as A
inner join Couleur as C on A.ID_COULEUR = C.ID_COULEUR
```

	ID_ARTICLE	NOM_ARTICLE	NOM_COULEUR
1	1	das Echte Märzen	Blonde
2	3	das Schwarze	Brune
3	4	Dinkel Acker Märzen	Blonde
4	5	Dinkel Acker Privat	Blonde

Avec cette jointure, les articles qui ont la valeur **NULL** comme **ID_COULEUR**, n'apparaissent pas dans la liste.

Les couleurs qui ne sont utilisées par aucun article, non plus.

Jointure externe gauche

Une jointure externe gauche est définie grâce à l'instruction "**LEFT OUTER JOIN**" ou encore "**LEFT JOIN**"

```
select A.ID_ARTICLE, a.NOM_ARTICLE, C.NOM_COULEUR from ARTICLE as A
left outer join Couleur as C on A.ID_COULEUR = C.ID_COULEUR
```

	ID_ARTICLE	NOM_ARTICLE	NOM_COULEUR
1	1	das Echte Märzen	Blonde
2	2	das Helle	NULL
3	3	das Schwarze	Brune
4	4	Dinkel Acker Märzen	Blonde

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 3 922 lignes

Cette fois ci, toutes les lignes de la table à gauche de la jointure (la table **ARTICLE**), sont bien présentes, même si l'**ID_COULEUR** a la valeur **NULL**.

Les couleurs qui ne sont utilisées par aucun article, ne sont toujours pas présentes.

Jointure externe droite

Une jointure externe droite est définie grâce à l'instruction "**RIGHT OUTER JOIN**" ou encore "**RIGHT JOIN**"

```
select A.ID_ARTICLE, a.NOM_ARTICLE, C.NOM_COULEUR from ARTICLE as A
right outer join Couleur as C on A.ID_COULEUR = C.ID_COULEUR
```

	ID_ARTICLE	NOM_ARTICLE	NOM_COULEUR
1	NULL	NULL	Bleue
2	1	das Echte Märzen	Blonde
3	3	das Schwarze	Brune
4	4	Dinkel Acker Märzen	Blonde

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 3 217 lignes

Cette fois ci, toutes les lignes de la table à droite de la jointure (la table **COULEUR**), sont bien présentes, même si aucun article n'est de cette couleur.

Les articles qui ont la valeur **NULL** comme **ID_COULEUR**, n'apparaissent pas dans la liste.

Jointure externe complète

Une jointure externe complète, est définie grâce à l'instruction "**FULL OUTER JOIN**" ou encore "**FULL JOIN**"

```
select A.ID_ARTICLE, a.NOM_ARTICLE, C.NOM_COULEUR from ARTICLE as A
full outer join Couleur as C on A.ID_COULEUR = C.ID_COULEUR
```

	ID_ARTICLE	NOM_ARTICLE	NOM_COULEUR
1	NULL	NULL	Bleue
2	1	das Echte Märzen	Blonde
3	2	das Helle	NULL
4	3	das Schwarze	Brune

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 3 923 lignes

Cette fois ci, toutes les lignes de la table à gauche de la jointure (la table **ARTICLE**), sont bien présentes, même si l'**ID_COULEUR** a la valeur **NULL**.

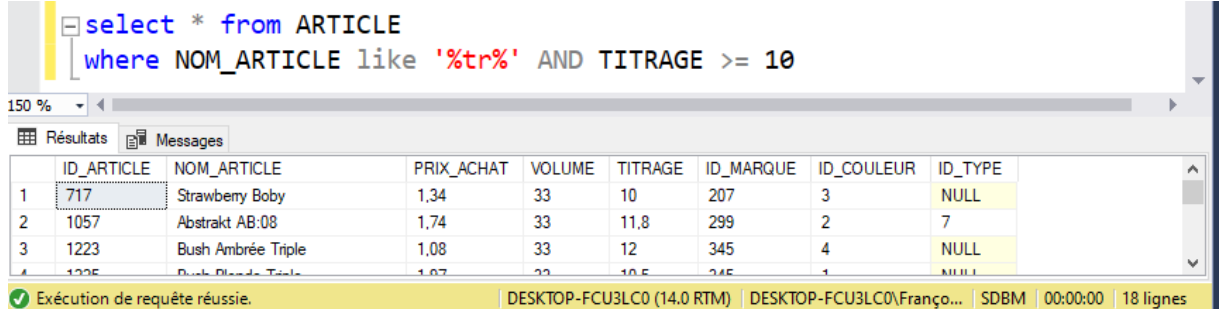
Et toutes les lignes de la table à droite de la jointure (la table **COULEUR**), sont bien présentes, même si aucun article n'est de cette couleur.

WHERE

La clause **WHERE** permet la mise en œuvre des restrictions. Les conditions sont des expressions booléennes composées de nom de colonnes, de constantes, de fonctions, d'opérateurs de comparaison et des opérateurs logiques.

Exemple

Si nous voulons les articles dont le nom comporte la chaîne de caractères 'tr' titrant 10° ou plus :



```
select * from ARTICLE
where NOM_ARTICLE like '%tr%' AND TITRAGE >= 10
```

150 %

Résultats Messages

	ID_ARTICLE	NOM_ARTICLE	PRIX_ACHAT	VOLUME	TITRAGE	ID_MARQUE	ID_COULEUR	ID_TYPE
1	717	Strawberry Boby	1,34	33	10	207	3	NULL
2	1057	Abstrakt AB:08	1,74	33	11,8	299	2	7
3	1223	Bush Ambrée Triple	1,08	33	12	345	4	NULL
4	1235	Bush Blende Triple	1,07	33	10,5	345	1	NULL

✓ Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 18 lignes

Les opérateurs de comparaisons

Opérateur	Description
=	Égale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (utile pour les nombres ou dates)
LIKE	Recherche en spécifiant le début, milieu ou fin d'un mot.
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

Les opérateurs Logiques

AND : permet de joindre plusieurs conditions dans une requête

OR : permet une alternative dans plusieurs conditions

NOT : permet d'exprimer l'inverse d'une condition

Ils peuvent bien sur être combiné pour effectuer des recherches puissantes

GROUP BY

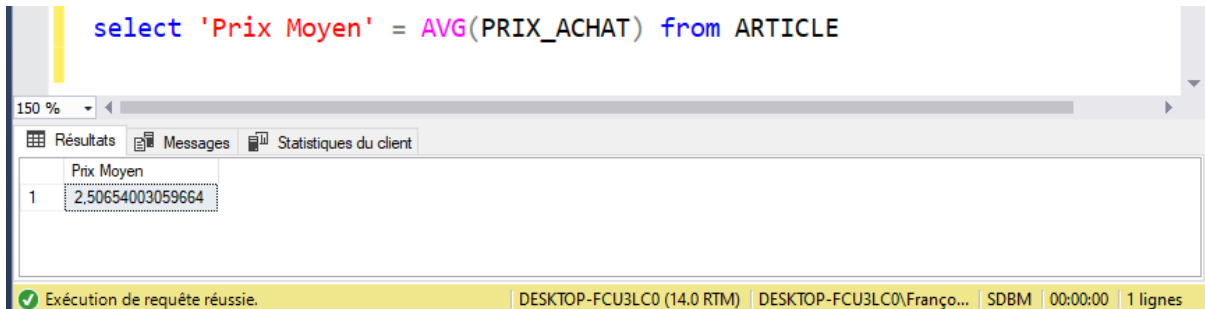
Lorsque l'on veut effectuer des calculs d'aggrégats (**COUNT**, **SUM**, **AVG**, **MIN**, **MAX**, ...), il est fréquent que le calcul doive être réalisé pour des sous-ensembles de valeurs bien particuliers.

Dans ce cas, les sous-ensembles sont définis par l'intermédiaire de la clause **GROUP BY**. Cette clause permet de construire les sous-ensembles avant de réaliser le calcul.

Exemple

Si nous voulons calculer le prix moyen des articles :

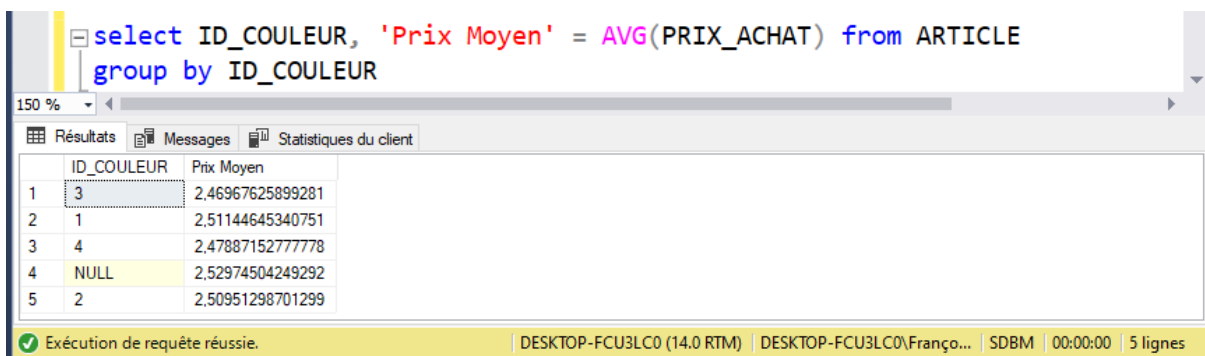
```
select 'Prix Moyen' = AVG(PRIX_ACHAT) from ARTICLE
```



	Prix Moyen
1	2.50654003059664

Si maintenant, nous voulons le prix moyen suivant la couleur (**ID_COULEUR**), nous allons utiliser la clause **GROUP BY** :

```
select ID_COULEUR, 'Prix Moyen' = AVG(PRIX_ACHAT) from ARTICLE  
group by ID_COULEUR
```



	ID_COULEUR	Prix Moyen
1	3	2.46967625899281
2	1	2.51144645340751
3	4	2.47887152777778
4	NULL	2.52974504249292
5	2	2.50951298701299

Conseil

Attention s'il y a des colonnes dans la partie *select* qui ne sont pas repris dans le *group by*, alors ces colonnes recevront une valeur au hasard

Soyez critique quant à la cohérence des données retournées par la requête

HAVING

Nous avons déjà vu la clause **WHERE**, qui permet de poser des restrictions. Cette clause ne s'applique pas aux fonctions d'agrégation.

Exemple

Si je veux éliminer les articles dont l'**ID_COULEUR** est null :

```
select ID_COULEUR, 'Prix Moyen' = AVG(PRIX_ACHAT) from ARTICLE
where ID_COULEUR is not null
group by ID_COULEUR
```

	ID_COULEUR	Prix Moyen
1	3	2.46967625899281
2	1	2.51144645340751
3	4	2.47887152777778
4	2	2.50951298701299

Exécution de requête réussie. | DESKTOP-FCU3LC0 (14.0 RTM) | DESKTOP-FCU3LC0\Franço... | SDBM | 00:00:00 | 4 lignes

Si je veux au contraire éliminer les prix moyens supérieurs ou égaux à 2.5, l'utilisation de la clause **WHERE**, déclenchera une erreur :

```
select ID_COULEUR, 'Prix Moyen' = AVG(PRIX_ACHAT) from ARTICLE
where AVG(PRIX_ACHAT) < 2.5
group by ID_COULEUR
```

Msg 147, Niveau 15, État 1, Ligne 28
Un agrégat ne peut pas apparaître dans une clause WHERE à moins que ce ne soit un

Heure de fin : 2020-11-10T08:41:55.9425294+01:00

Requête terminée avec des erreurs. | DESKTOP-FCU3LC0 (14.0 RTM) | DESKTOP-FCU3LC0\Franço... | SDBM | 00:00:00 | 0 lignes

Pour effectuer une restriction sur cette fonction d'agrégation, nous utiliserons la clause **HAVING** :

```
select ID_COULEUR, 'Prix Moyen' = AVG(PRIX_ACHAT) from ARTICLE
group by ID_COULEUR
having AVG(PRIX_ACHAT) < 2.5
```

	ID_COULEUR	Prix Moyen
1	3	2.46967625899281
2	4	2.47887152777778

Exécution de requête réussie. | DESKTOP-FCU3LC0 (14.0 RTM) | DESKTOP-FCU3LC0\Franço... | SDBM | 00:00:00 | 2 lignes

ORDER BY

La clause **ORDER BY**, va nous permettre de trier les lignes suivant les critères exprimés par **liste_expressions**.

ORDER BY liste_expressions DESC | ASC

ASC : Ascendant (valeur par défaut)

DESC : Descendant

Exemple

Si l'on veut la liste des articles classés par titrage décroissant, puis par nom croissant :

```
select * from ARTICLE
order by TITRAGE desc, NOM_ARTICLE asc
```

	ID_ARTICLE	NOM_ARTICLE	PRIX_ACHAT	VOLUME	TITRAGE	ID_MARQUE	ID_COULEUR	ID_TYPE
1	615	Black Damnation V Double Black (version du ACBF)	1,54	33	26	185	2	7
2	2576	Black Damnation V Double Black (version du ACBF)	2,34	75	26	185	2	7
3	3545	Big Worst	2,25	75	18,5	460	4	10
4	1584	Big Worst	1,34	33	18,5	460	4	10
5	1081	Tokyo*	1,66	33	18,2	299	2	7

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 3 922 lignes

Sous-requêtes

Les sous-requêtes peuvent être imbriquées c'est-à-dire s'exécutant une fois lorsque la requête externe s'exécute, ou en corrélation c'est-à-dire s'exécutant une fois pour chaque ligne renvoyée lors de l'exécution de la requête externe.

Une sous-requête est également appelée « requête interne » ou « sélection interne » et l'instruction qui la contient est aussi appelée « requête externe » ou « sélection externe ».

A la place d'une expression

Exemple

Si je veux une liste des articles avec leur référence, leur désignation et leur couleur, je peux procéder de manière classique avec une jointure :

```
select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = NOM_COULEUR
from ARTICLE A
left join COULEUR C on A.ID_COULEUR = C.ID_COULEUR
```

	id	désignation	Couleur
1	1	das Echte Märzen	Blonde
2	2	das Helle	NULL
3	3	das Schwarze	Brune
4	4	Dinkel Acker Märzen	Blonde

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 3 922 lignes

Mais je peux également éviter la jointure, en utilisant une requête imbriquée pour aller chercher le nom de la couleur :

```

select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = (select NOM_COULEUR from COULEUR where ID_COULEUR = A.ID_COULEUR)
from ARTICLE A
  
```

	id	désignation	Couleur
1	1	das Echte Märzen	Blonde
2	2	das Helle	NULL
3	3	das Schwarze	Brune
4	4	Dinkel Acker Märzen	Blonde

Exécution de requête réussie. | DESKTOP-FCU3LC0 (14.0 RTM) | DESKTOP-FCU3LC0\Franço... | SDBM | 00:00:00 | 3 922 lignes



Remarque

Ces requêtes sont équivalente uniquement si on utilise un LEFT JOIN.

Avez vous une explication?



Les 2 requêtes ramènent le même nombre de lignes, mais sont-elles équivalentes ?

Je vous propose de lancer tour à tour les 2 requêtes et de regarder les statistiques client

Statistiques du profil de requête			
Nombre d'instructions INSERT, DELETE et UPDATE	0	→	0
Lignes affectées par les instructions INSERT, DELETE ou ...	0	→	0
Nombre d'instructions SELECT	2	→	2
Lignes retournées par les instructions SELECT	3923	→	3923
Nombre de transactions	0	→	0
Statistiques réseau			
Nombre de boucles de serveur	2	→	2
Paquets TDS envoyés depuis le client	2	→	2
Paquets TDS reçus du serveur	30	→	30
Octets envoyés depuis le client	392	↑	374
Octets reçus du serveur	117433	→	117433
Statistiques de temps			
Heure de traitement du client	15	↑	9
Durée totale d'exécution	21	↑	13
Délai d'attente des réponses du serveur	6	↑	4

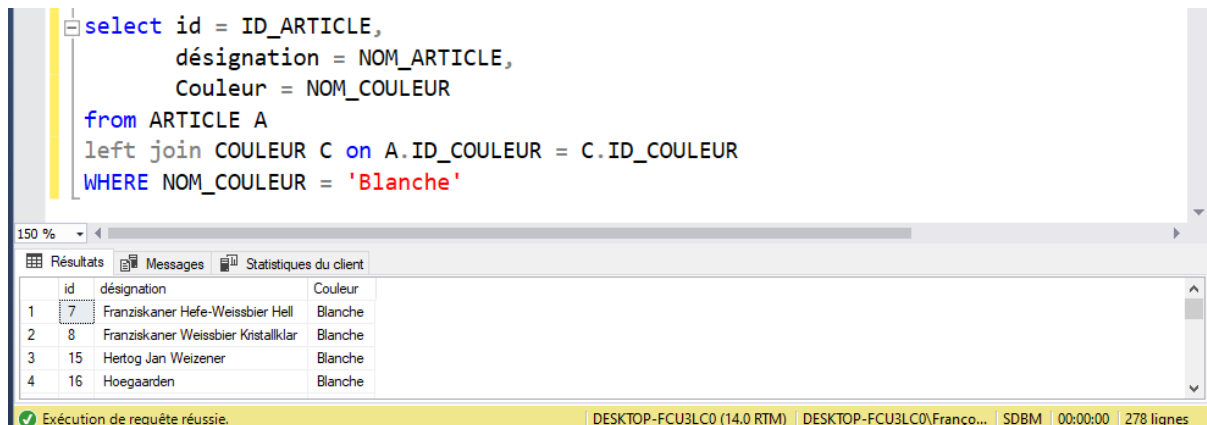
Est-ce significatif?

Dans une clause WHERE

Exemple

Si je veux une liste des articles de couleur **Blanche**, avec leur référence, leur désignation et leur couleur, je peux procéder de manière classique avec une jointure :

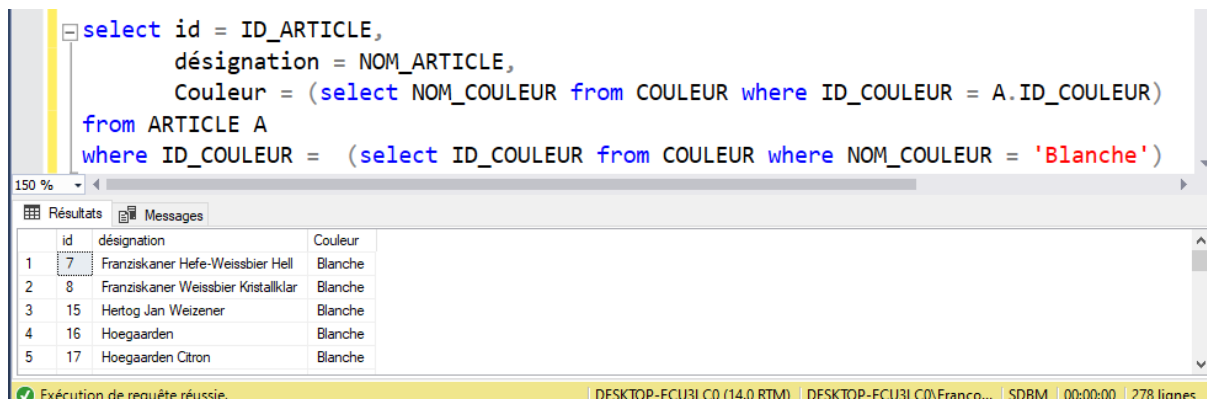
```
select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = NOM_COULEUR
from ARTICLE A
left join COULEUR C on A.ID_COULEUR = C.ID_COULEUR
WHERE NOM_COULEUR = 'Blanche'
```



	id	désignation	Couleur
1	7	Franziskaner Hefe-Weissbier Hell	Blanche
2	8	Franziskaner Weissbier Kristallklar	Blanche
3	15	Hertog Jan Weizener	Blanche
4	16	Hoegaarden	Blanche

Mais je peux également éviter la jointure, en utilisant une requête imbriquée pour aller chercher le nom de la couleur, puis une autre pour alimenter ma clause **WHERE** :

```
select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = (select NOM_COULEUR from COULEUR where ID_COULEUR = A.ID_COULEUR)
from ARTICLE A
where ID_COULEUR = (select ID_COULEUR from COULEUR where NOM_COULEUR = 'Blanche')
```



	id	désignation	Couleur
1	7	Franziskaner Hefe-Weissbier Hell	Blanche
2	8	Franziskaner Weissbier Kristallklar	Blanche
3	15	Hertog Jan Weizener	Blanche
4	16	Hoegaarden	Blanche
5	17	Hoegaarden Citron	Blanche

Remarque

La requête utilisée dans la clause where :

```
(select ID_COULEUR from COULEUR where NOM_COULEUR = 'Blanche')
```

ne renvoie qu'une valeur unique, mais si l'on voulait les articles des couleurs dont le nom commence par **B**, on utiliserait :

```
(select ID_COULEUR from COULEUR where NOM_COULEUR like 'B%')
```

Requête qui va nous renvoyer une liste d'**ID_COULEUR**

Il faudra donc utiliser l'opérateur de comparaison **IN**, à la place de =.

```

select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = (select NOM_COULEUR from COULEUR where ID_COULEUR = A.ID_COULEUR)
from ARTICLE A
where ID_COULEUR in (select ID_COULEUR from COULEUR where NOM_COULEUR like 'B%')

```

	id	désignation	Couleur
1	1	das Echte Märzen	Blonde
2	3	das Schwarze	Brune
3	4	Dinkel Acker Märzen	Blonde
4	5	Dinkel Acker Privat	Blonde
5	6	Franziskaner Hefe-Weissbier Dunkel	Brune

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 2 640 lignes

Dans une Clause WHERE avec une fonction d'agrégation

Exemple

Supposons que l'on veuille une liste des articles de titrage plus élevé que le plus fort des articles de couleur Blanche'.

La requête permettant d'obtenir le titrage le plus élevé des articles de couleurs **Blanche** est :

```

select max(TITRAGE)
from ARTICLE
where ID_COULEUR = (select ID_COULEUR from COULEUR where NOM_COULEUR = 'blanche')

```

	1
	10

(Aucun nom de colonne)

Il suffit ensuite d'utiliser cette requête dans la clause WHERE de la requête principale :

```

select id = ID_ARTICLE,
       désignation = NOM_ARTICLE,
       Couleur = (select NOM_COULEUR from COULEUR where ID_COULEUR = A.ID_COULEUR),
       titrage = TITRAGE
from ARTICLE A
where TITRAGE > (select max(TITRAGE)
                  from ARTICLE
                  where ID_COULEUR = (select ID_COULEUR from COULEUR where NOM_COULEUR = 'blanche')
                 )

```

	id	désignation	Couleur	titrage
1	135	Abbaye du Val Dieu Grand Cru (f)	Brune	10,5
2	142	Rochefort 10	Brune	11,3
3	145	Westvleteren 12	Brune	10,2
4	156	Old Numbokull	Ambrée	11

Exécution de requête réussie. DESKTOP-FCU3LC0 (14.0 RTM) DESKTOP-FCU3LC0\Franço... SDBM 00:00:00 172 lignes

Documentation officielle

SELECT



ver15

<https://docs.microsoft.com/fr-fr/sql/t-sql/queries/select-transact-sql?view=sql-server->



Sous-requêtes



[https://docs.microsoft.com/fr-fr/sql/relational-databases/performance/subqueries?
view=sql-server-ver15](https://docs.microsoft.com/fr-fr/sql/relational-databases/performance/subqueries?view=sql-server-ver15)



Œuvre collective de l'AFPA

Sous le pilotage de la Direction de l'Ingénierie.

© AFPA

Reproduction interdite

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la transformation, l'arrangement ou la reproduction par un art ou un procédé quelconque. »

Date de mise à jour 08/09/2022