



Les requetes

Les jointures

Les Jointures

Les jointures en SQL sont utilisées pour combiner des données provenant de plusieurs tables en une seule vue résultante. Elles permettent de lier les enregistrements des tables en fonction de certaines conditions.

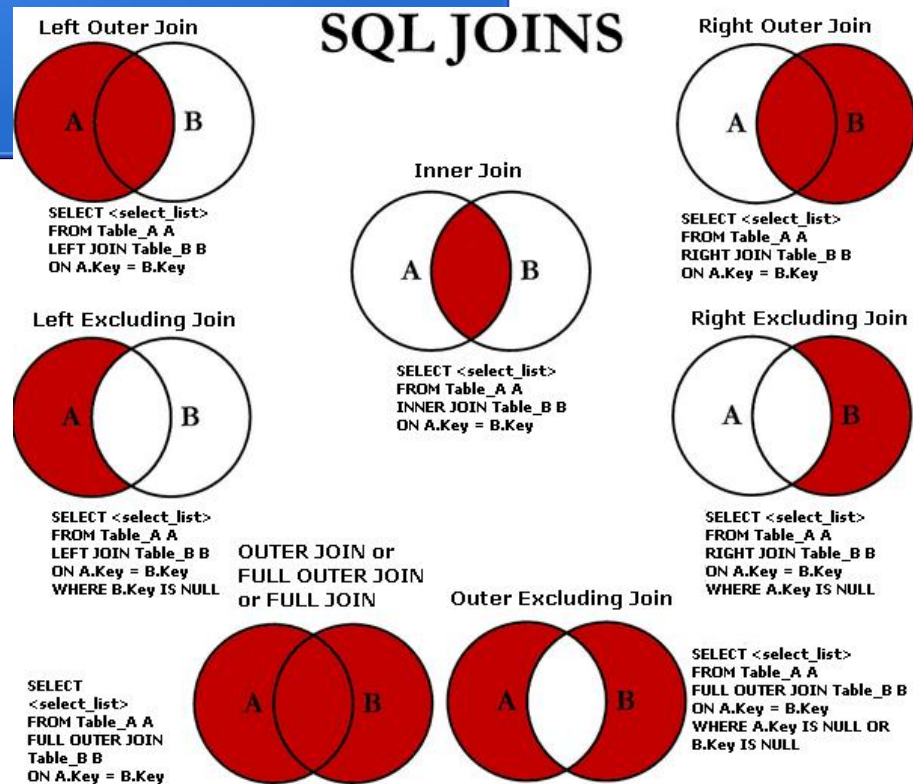
Clauses de Jointure

Les jointures sont spécifiées dans la clause FROM d'une requête SQL, et les conditions de jointure sont généralement spécifiées dans la clause ON.

SELECT ...

FROM Table1

INNER JOIN Table2 ON Table1.Colonne = Table2.Colonne;



'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

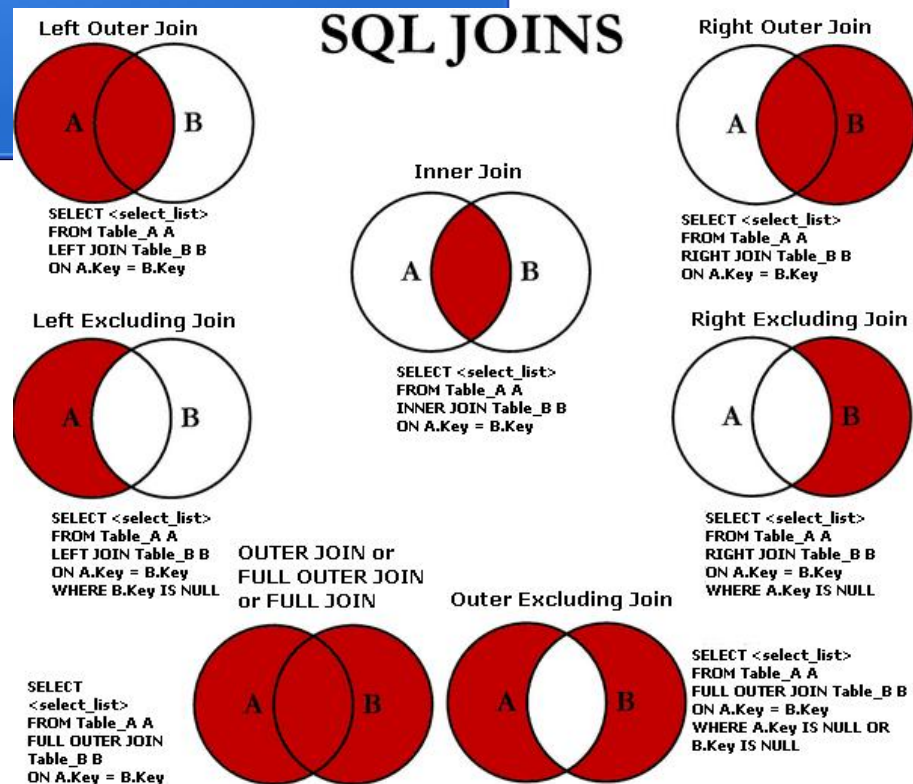
Les Jointures

1. INNER JOIN

Une INNER JOIN renvoie les enregistrements qui ont des valeurs correspondantes dans les deux tables. Elle exclut les enregistrements qui n'ont pas de correspondance.

Exemple : Sélectionnez tous les clients et leurs commandes correspondantes.

```
SELECT Clients.Nom, Commandes.Numéro  
FROM Clients  
INNER JOIN Commandes ON Clients.ID =  
Commandes.ClientID;
```



'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

Les Jointures

2. LEFT JOIN (ou LEFT OUTER JOIN)

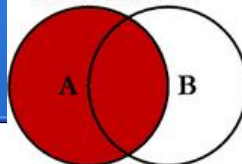
Une LEFT JOIN renvoie tous les enregistrements de la table de gauche (table de gauche dans la clause JOIN) et les enregistrements correspondants de la table de droite. Si aucune correspondance n'est trouvée dans la table de droite, les colonnes de la table de droite auront des valeurs NULL.

Exemple : Sélectionnez tous les employés et leurs affectations correspondantes (le cas échéant).

```
SELECT Employés.Nom, Affectations.Tâche
FROM Employés
LEFT JOIN Affectations ON Employés.ID =
Affectations.EmployéID;
```

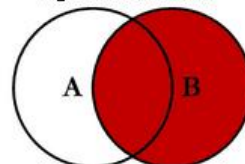
SQL JOINS

Left Outer Join



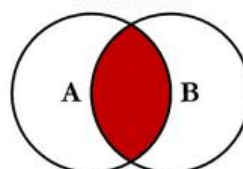
```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
```

Right Outer Join



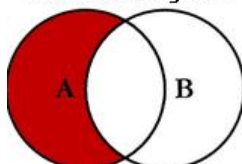
```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
```

Inner Join



```
SELECT <select_list>
FROM Table_A A
INNER JOIN Table_B B
ON A.Key = B.Key
```

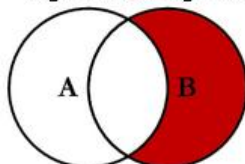
Left Excluding Join



```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

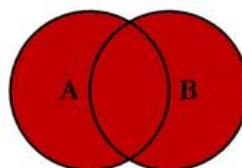
OUTER JOIN or
FULL OUTER JOIN
or FULL JOIN

Right Excluding Join

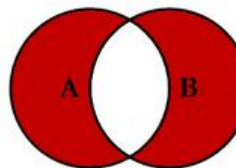


```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

Outer Excluding Join



```
SELECT
<select_list>
FROM Table_A A
FULL OUTER JOIN
Table_B B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM Table_A A
FULL OUTER JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL OR
B.Key IS NULL
```

'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

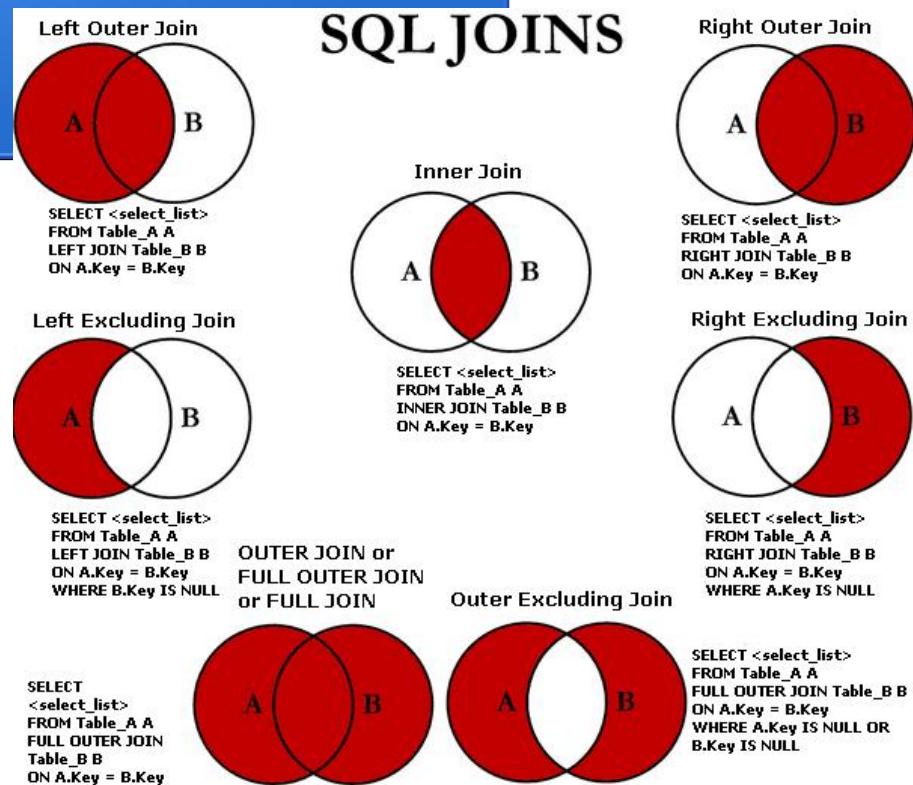
Les Jointures

3. RIGHT JOIN (ou RIGHT OUTER JOIN)

Une RIGHT JOIN est similaire à une LEFT JOIN, mais renvoie tous les enregistrements de la table de droite et les enregistrements correspondants de la table de gauche. Si aucune correspondance n'est trouvée dans la table de gauche, les colonnes de la table de gauche auront des valeurs NULL.

Exemple : Sélectionnez toutes les commandes et les clients correspondants (le cas échéant).

```
SELECT Clients.Nom, Commandes.Numéro  
FROM Clients  
RIGHT JOIN Commandes ON Clients.ID =  
Commandes.ClientID;
```



'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

Les Jointures

4. FULL JOIN (ou FULL OUTER JOIN)

Une FULL JOIN renvoie tous les enregistrements des deux tables avec des correspondances là où elles existent et des valeurs NULL là où il n'y a pas de correspondance.

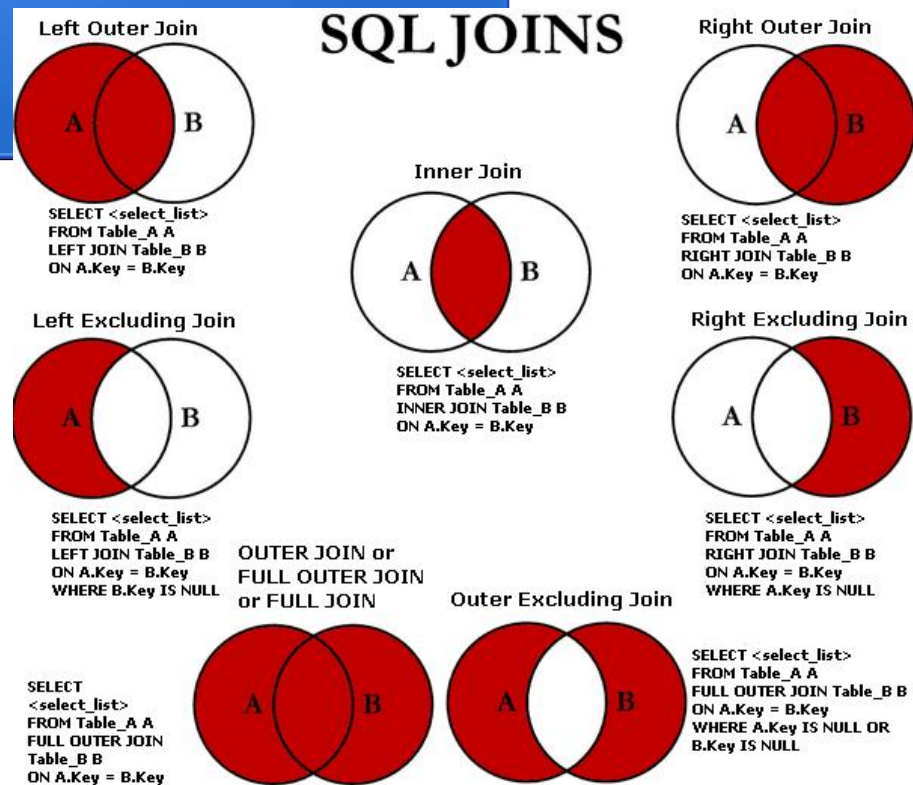
Exemple : Sélectionnez tous les produits et leurs commentaires correspondants (le cas échéant).

```
SELECT Produits.Nom, Commentaires.Texte
```

```
FROM Produits
```

```
FULL JOIN Commentaires ON Produits.ID =
```

```
Commentaires.ProduitID;
```



'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

Les Jointures

Les **alias** de table permettent d'attribuer des noms courts et significatifs aux tables, ce qui simplifie la rédaction des requêtes. Par exemple :

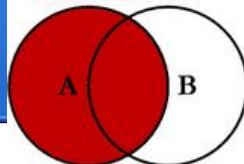
```
SELECT E.Nom, C.Numéro
```

```
FROM Employés AS E
```

```
INNER JOIN Commandes AS C ON E.ID = C.employéID;
```

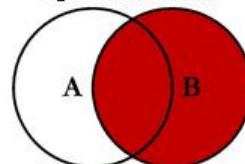
SQL JOINS

Left Outer Join



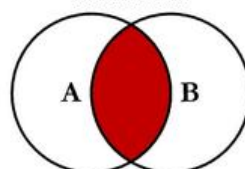
```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
```

Right Outer Join



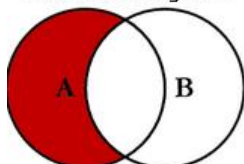
```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
```

Inner Join



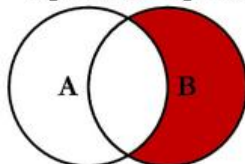
```
SELECT <select_list>
FROM Table_A A
INNER JOIN Table_B B
ON A.Key = B.Key
```

Left Excluding Join



```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

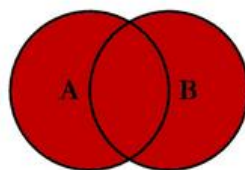
Right Excluding Join



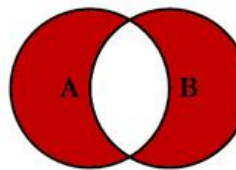
```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

Outer Join or
FULL OUTER JOIN
or FULL JOIN

Outer Excluding Join



```
SELECT
<select_list>
FROM Table_A A
FULL OUTER JOIN
Table_B B
ON A.Key = B.Key
```



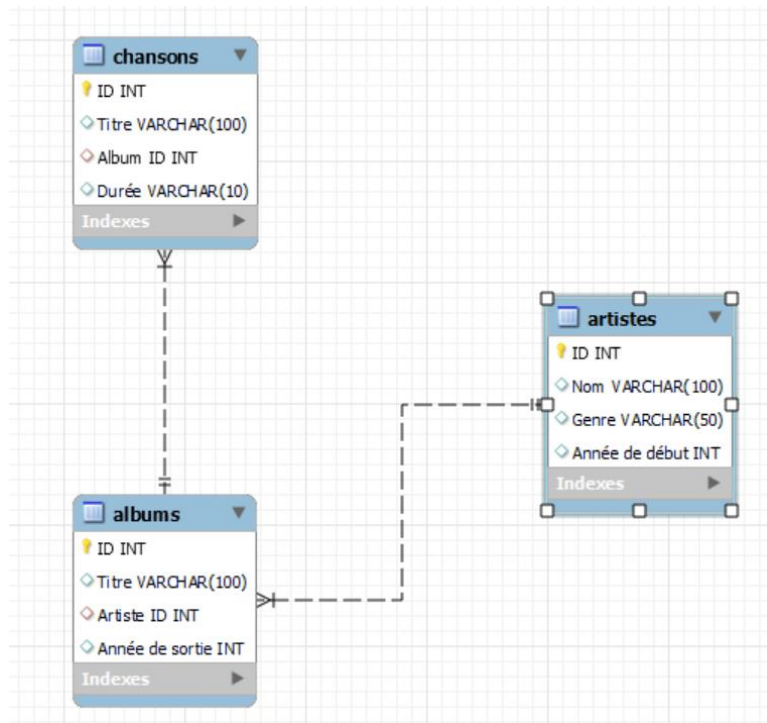
```
SELECT <select_list>
FROM Table_A A
FULL OUTER JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL OR
B.Key IS NULL
```

'A' & 'B' are two sets.

1. $A \cap B$ = Inner Join ('n' - intersection)
2. $A \cup (A \cap B)$ = Left Join ('u' - Union)
3. $(A \cap B) \cup B$ = Right Join
4. $A \cup B \cup (A \cap B)$ = Outer Join
5. $A - B$ = Left Join Excluding Inner Join or Relative Component
6. $B - A$ = Right Join Excluding Inner Join
7. $(A - B) \cup (B - A)$ = Outer Join Excluding Inner Join

Application

- Obtenez tous les artistes du genre "Pop" et leurs albums :
- Obtenez une liste des artistes et de leurs albums correspondants (INNER JOIN)
- Listez tous les artistes, même ceux sans albums (LEFT JOIN)
- Affichez tous les albums et leurs artistes correspondants (RIGHT JOIN)
- Sélectionnez les chansons et les albums correspondants
- Affichez les artistes qui n'ont pas d'albums



Les Agrégations

Agrégation

Les opérations d'agrégation en SQL permettent de calculer des valeurs résumées à partir d'un ensemble de données. Ces valeurs résumées peuvent inclure des totaux, des moyennes, des minimums, des maximums, etc. Les opérations d'agrégation sont couramment utilisées avec l'instruction GROUP BY pour regrouper les données avant d'appliquer une opération d'agrégation.

Opérations d'agrégation

1. COUNT


L'opération COUNT permet de compter le nombre d'enregistrements dans un ensemble de données.

Syntaxe :

SELECT COUNT(colonne) FROM table;

Exemple : Comptez le nombre d'albums pour chaque artiste.

```
SELECT `Artiste ID`, COUNT(ID) AS Nombre_d_albums  
FROM Albums  
GROUP BY `Artiste ID`;
```

Result Grid  Filter Rows: <input type="text"/>		
	Artiste ID	Nombre_d_albums
▶	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	7	1
	8	1
	10	1

Opérations d'agrégation

2. SUM

L'opération SUM permet de calculer la somme des valeurs numériques dans une colonne.

Syntaxe :

```
SELECT SUM(colonne) FROM table;
```

Exemple : Calculez la somme des durées de toutes les chansons.

```
SELECT SUM(Durée) AS Durée_totale  
FROM Chansons;
```

	Durée_totale
▶	94

Opérations d'agrégation

3. AVG

L'opération AVG permet de calculer la moyenne des valeurs numériques dans une colonne.

Syntaxe :

SELECT AVG(colonne) FROM table;

Exemple : Calculez la durée moyenne des chansons.

```
SELECT AVG(Durée) AS Durée_moyenne  
FROM Chansons;
```

	Durée_moyenne
▶	4.2727272727272725

Opérations d'agrégation

4. MIN et MAX

Les opérations MIN et MAX permettent de trouver la valeur minimale et maximale dans une colonne, respectivement.

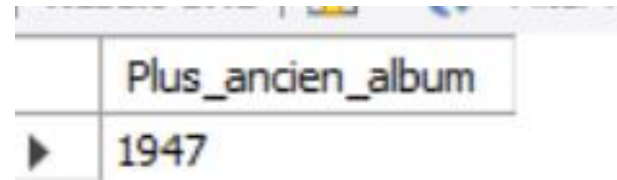
Syntaxe :

```
SELECT MIN(colonne) FROM table;
```

```
SELECT MAX(colonne) FROM table;
```

Exemple : Trouvez la date de sortie de l'album le plus ancien.

```
SELECT MIN(`Année de sortie`) AS Plus_ancien_album  
FROM Albums;
```



	Plus_ancien_album
▶	1947

GroupBy

L'instruction GROUP BY permet de regrouper les données en fonction d'une ou plusieurs colonnes, puis d'appliquer des opérations d'agrégation à chaque groupe.

Syntaxe :

SELECT colonnes, opération(colonne)

FROM table

GROUP BY colonnes;

Exemple : Comptez le nombre d'albums par année de sortie

```
SELECT `Année de sortie`, COUNT(ID) AS Nombre_d_albums
FROM Albums
GROUP BY `Année de sortie`
ORDER BY `Année de sortie`;
```

Année de sortie	Nombre_d_albums
1947	1
1954	1
1969	1
1982	1
1984	3
2013	2

Having

L'instruction HAVING est utilisée en conjonction avec GROUP BY pour filtrer les groupes résultants en fonction d'une condition.

Syntaxe :

SELECT colonnes, opération(colonne)

FROM table

GROUP BY colonnes

HAVING condition;

Exemple : Sélectionnez les années de sortie avec plus de 5 albums.

```
SELECT `Année de sortie`, COUNT(ID) AS Nombre_d_albums
```

```
FROM Albums
```

```
GROUP BY `Année de sortie`
```

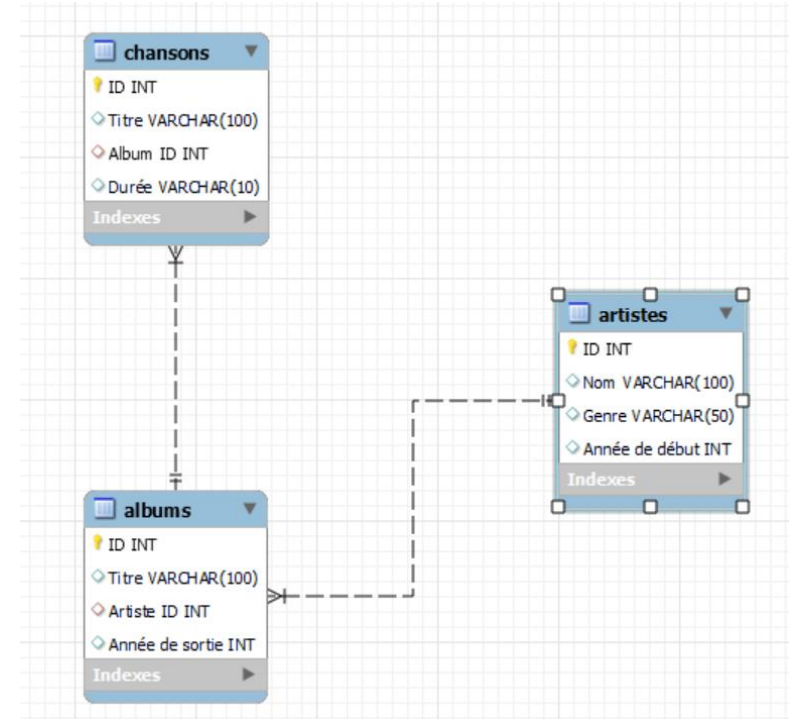
```
HAVING COUNT(ID) > 2
```

```
ORDER BY `Année de sortie`;
```

Année de sortie	Nombre_d_albums
1984	3
2013	2

Application

- Trouvez le nombre total d'albums dans la base de données :
- Calculez la durée totale de toutes les chansons dans la base de données :
- Déterminez la moyenne d'années de début des artistes :
- Trouvez le nombre d'albums pour chaque artiste (groupé par artiste) :
- Sélectionnez les années de sortie avec plus de 3 albums (groupées par année) :
- Trouvez la durée moyenne des chansons pour chaque album (groupée par album) :
- Calculez le nombre total de chansons dans chaque genre musical (groupé par genre) :
- Déterminez l'année de sortie de l'album le plus récent :
- Trouvez le genre musical le plus populaire (genre avec le plus grand nombre d'artistes) :
- Calculez la durée totale de chaque album en minutes (groupée par album) :



Hands On

Valider le chapitre 3 de la ressource suivante:

<https://my-learning.w3schools.com/tutorial/sql>

3. Statements Part 2

- Lesson 3.1 - Joins
- Lesson 3.2 - INNER JOIN
- Lesson 3.3 - LEFT JOIN
- Lesson 3.4 - RIGHT JOIN
- Lesson 3.5 - FULL JOIN
- Lesson 3.6 - Self Join
- Lesson 3.7 - UNION
- Lesson 3.8 - GROUP BY
- Lesson 3.9 - HAVING

Les Opérations ensemblistes

Les Opérations Ensemblistes

Les requêtes de types opération ensemblistes en SQL sont utilisées pour effectuer des opérations sur des ensembles de données dans les tables. Ces opérations incluent l'intersection, l'union, la différence, etc.

Les Opérations Ensemblistes

1. UNION

L'opération UNION permet de combiner les résultats de deux requêtes SQL distinctes en un seul ensemble de résultats.

Les doublons sont automatiquement éliminés.

Syntaxe :

```
SELECT colonnes FROM table1
```

```
UNION
```

```
SELECT colonnes FROM table2;
```

Exemple : Combinez les résultats des artistes pop et rock.

```
SELECT Nom, Genre FROM Artistes WHERE Genre = 'Pop'
```

```
UNION
```

```
SELECT Nom, Genre FROM Artistes WHERE Genre = 'Rock';
```

Nom	Genre
Michael Jackson	Pop
Madonna	Pop
Stromae	Pop
Justin Timberlake	Pop
Adele	Pop
The Beatles	Rock

Les Opérations Ensemblistes

2. INTERSECT

L'opération INTERSECT renvoie les enregistrements qui se trouvent à la fois dans le résultat de la première requête et dans le résultat de la deuxième requête.

Syntaxe :

MySQL ne prend pas en charge directement l'opération INTERSECT. Vous pouvez simuler l'INTERSECT en utilisant INNER JOIN ou EXISTS.

Exemple : Trouvez les artistes qui sont à la fois pop et rock (simulant INTERSECT avec INNER JOIN).

```
SELECT A.Nom  
FROM Artistes A  
INNER JOIN (SELECT ID FROM Artistes WHERE Genre = 'Pop') P  
ON A.ID = P.ID  
WHERE EXISTS (SELECT ID FROM Artistes WHERE Genre = 'Rock' AND ID = A.ID);
```

	Nom
--	-----

Les Opérations Ensemblistes

3. EXCEPT (ou MINUS)

L'opération EXCEPT renvoie les enregistrements qui se trouvent dans le résultat de la première requête mais pas dans le résultat de la deuxième requête. C'est similaire à la différence ensembliste.

Syntaxe :

MySQL ne prend pas en charge directement l'opération EXCEPT. Vous pouvez simuler l'EXCEPT en utilisant LEFT JOIN.

Exemple : Trouvez les artistes qui sont pop mais pas rock (simulant EXCEPT avec LEFT JOIN).

```
SELECT A.Nom  
FROM Artistes A  
LEFT JOIN (SELECT ID FROM Artistes WHERE Genre = 'Rock') R  
ON A.ID = R.ID  
WHERE R.ID IS NULL AND A.Genre = 'Pop';
```

Nom	genre
Michael Jackson	Pop
Madonna	Pop
Stromae	Pop
Justin Timberlake	Pop
Adele	Pop

Les Opérations Ensemblistes

4. UNION ALL

L'opération UNION ALL est similaire à UNION, mais elle inclut tous les enregistrements, y compris les doublons.

Syntaxe :

```
SELECT colonnes FROM table1
```

```
UNION ALL
```

```
SELECT colonnes FROM table2;
```

Exemple : Combinez tous les artistes, y compris les doublons de genre.

```
SELECT Nom, Genre FROM Artistes WHERE Genre = 'Pop'
```

```
UNION ALL
```

```
SELECT Nom, Genre FROM Artistes WHERE Genre = 'Rock';
```

Nom	Genre
Michael Jackson	Pop
Madonna	Pop
Stromae	Pop
Justin Timberlake	Pop
Adele	Pop
The Beatles	Rock

Les Opérations Ensemblistes

5. NOT IN

L'opération NOT IN est utilisée pour sélectionner des enregistrements qui ne correspondent pas à une liste de valeurs spécifiées.

Syntaxe :

SELECT colonnes FROM table1

WHERE colonne NOT IN (valeur1, valeur2, ...);

Exemple : Sélectionnez les artistes qui ne sont pas de genre "Pop" ou "Rock".

Nom	Genre
Bob Marley	Reggae
Prince	R&B
Édith Piaf	Chanson française
Charles Aznavour	Chanson française
Justin Timberlake	R&B

SELECT Nom, Genre FROM Artistes

WHERE Genre NOT IN ('Pop', 'Rock');

Application

- Listez tous les artistes qui sont soit de genre "Pop" soit de genre "Rock" (UNION)
- Sélectionnez les artistes qui sont à la fois de genre "Pop" et "R&B" (INTERSECT simulé avec INNER JOIN)
- Trouvez les artistes qui sont de genre "Pop" mais pas "Rock" (EXCEPT simulé avec LEFT JOIN)
- Listez tous les artistes de genre "Pop" et "Rock" avec doublons (UNION ALL) :
- Sélectionnez les artistes qui ne sont ni "Pop" ni "Rock"

