FORMATION DEV WEB WEB MOBILE
CCP 2 - DÉVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB

LES BASES DE DONNEES RELATIONNELLES

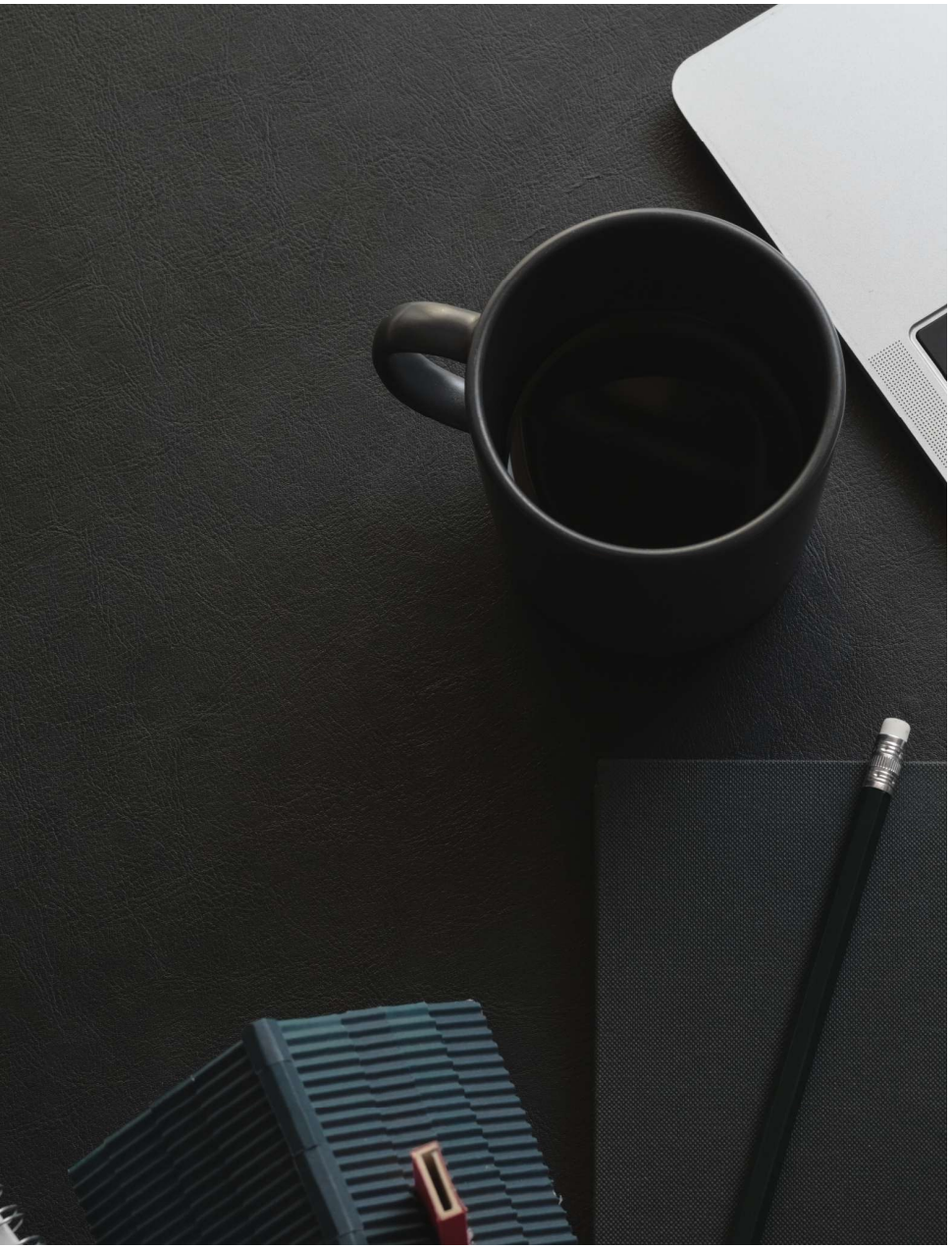
JOUR 5 : JOINTURES ET SOUS-REQUÊTES

PRESENTE PAR NATACHA DESSE

AFEC

DEROULE SEMAINE

- Découverte des BDD et Installation
- Les relations et modélisation (DEA)
- Contraintes et schéma physique
- Jeu d’essai, insertion et agrégations, requêtes avancées
- **Jointures et sous requêtes**



JOINTURES ET SOUS REQUÊTES

LES OBJECTIFS

Maîtriser les différents types de jointures SQL

Combiner les données de plusieurs tables avec efficacité

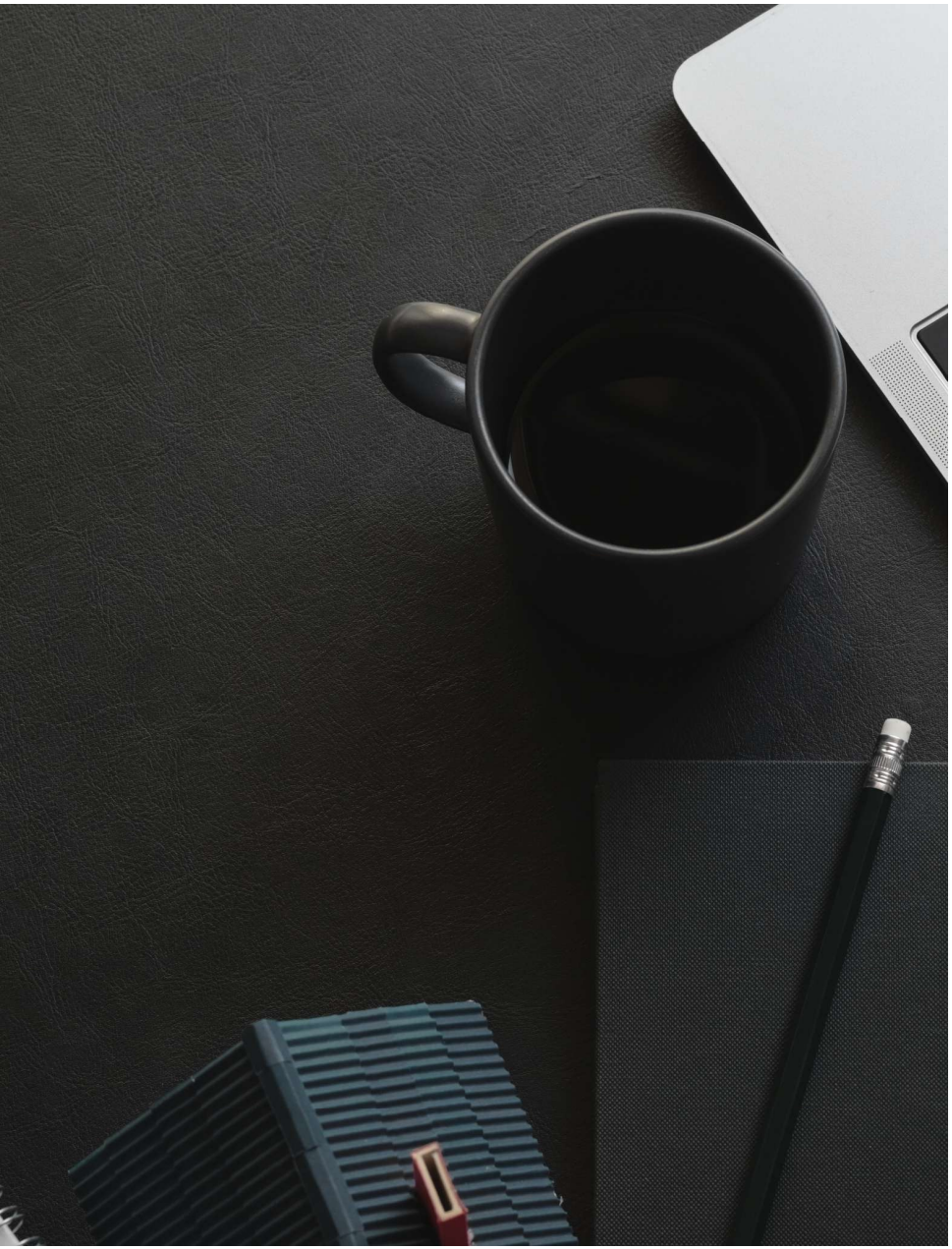
Appliquer des jointures dans des requêtes complexes

Utiliser des sous-requêtes dans les clauses SELECT, FROM et WHERE

NUAGE DE MOTS

Comment insérer de la données en SQL ?

https://app.wooclap.com/QOCXGJ?from=event-page



LES JOINTURES EN SQL

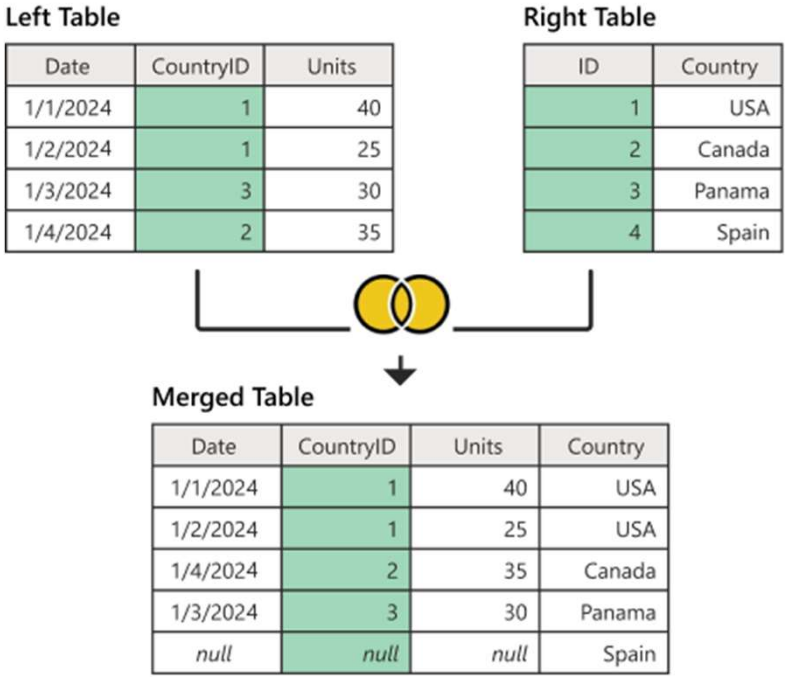
POURQUOI FAIRE ?

Définition :

Une jointure permet de **relier** des **données** de plusieurs **tables** en se basant sur une ou plusieurs **colonnes communes** (souvent une **clé étrangère**).

Utilisation typique :

On veut des informations sur une table extérieure en aggrégeant les données



Left Table

Date	CountryID	Units
1/1/2024	1	40
1/2/2024	1	25
1/3/2024	3	30
1/4/2024	2	35

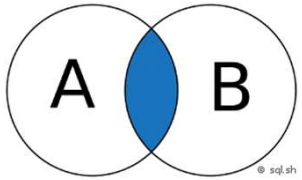
Right Table

ID	Country
1	USA
2	Canada
3	Panama
4	Spain

Merged Table

Date	CountryID	Units	Country
1/1/2024	1	40	USA
1/2/2024	1	25	USA
1/4/2024	2	35	Canada
1/3/2024	3	30	Panama
null	*null*	*null*	Spain

INNER JOIN



Une INNER JOIN retourne uniquement les lignes qui ont une **correspondance** dans les **deux** tables.

```sql
SELECT *
FROM table1
INNER JOIN table2
ON table1.colonne_commune =
table2.colonne_commune;
```

Remarque : on peut aussi écrire JOIN le INNER JOIN est le JOIN par défaut.

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

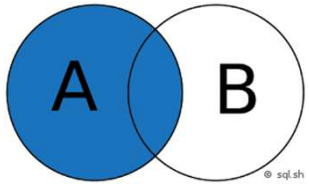
Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

```sql
SELECT *
FROM Employes
INNER JOIN Department
ON Employes .Dep = Department.id;
```

Id_dep	Nom_dep	Id	Nom	Age	Salaire	Profession	Dep
2	RH	1	Ismail	25	6000.00	Assistant	2
1	Informatique	2	Mohamed	30	8000.40	Directeur	1
3	Vente	3	Fatima	29	6000.00	Directeur	3
4	Strategies	4	Dounia	30	7000.00	Assistant	4
1	Informatique	5	Omar	29	9000.00	Ingenieur	1

LEFT JOIN



Retourne toutes les lignes de la table de **gauche**, et les lignes correspondantes de la table de gauche. Les lignes **sans correspondance** affichent **NULL**.

```sql
SELECT *
FROM table1
LEFT JOIN table2
ON table1.colonne =
table2.colonne;
```

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

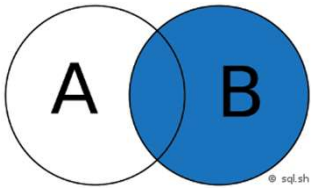
Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

```sql
SELECT *
FROM Employes AS E
LEFT JOIN Departement AS D
ON D.id= E.Dep;
```

Id	Nom	Age	Salaire	Profession	Dep	Id_dep	Nom_dep
2	Mohamed	30	8000.40	Directeur	1	1	Informatique
5	Omar	29	9000.00	Ingenieur	1	1	Informatique
1	Ismail	25	6000.00	Assistant	2	2	RH
3	Fatima	29	6000.00	Directeur	3	3	Vente
4	Dounia	30	7000.00	Assistant	4	4	Strategies
7	Mostafa	29	9000.00	Ingenieur	NULL	NULL	NULL

RIGHT JOIN



Retourne toutes les lignes de la table de droite, et les lignes correspondantes de la table de droite. Les lignes **sans correspondance** affichent **NULL**.

```sql
SELECT *
FROM table1
LEFT JOIN table2
ON table1.colonne =
table2.colonne;
```

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

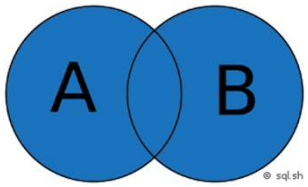
Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

```sql
SELECT *
FROM Employes AS E
RIGHT JOIN Departement AS D
ON D.id= E.Dep;
```

Id	Nom	Age	Salaire	Profession	Dep	Id_dep	Nom_dep
1	Ismail	25	6000.00	Assistant	2	2	RH
2	Mohamed	30	8000.40	Directeur	1	1	Informatique
3	Fatima	29	6000.00	Directeur	3	3	Vente
4	Dounia	30	7000.00	Assistant	4	4	Strategies
5	Omar	29	9000.00	Ingenieur	1	1	Informatique

FULL JOIN



Le FULL OUTER JOIN permet de récupérer toutes les lignes des deux tables, qu'il y ait correspondance ou non.

- Les lignes avec correspondance sont fusionnées.
- Les lignes sans correspondance apparaissent avec des NULL du côté manquant.

```sql
SELECT *
FROM table1
FULL OUTER JOIN table2
ON table1.colonne =
table2.colonne;
```

- Tous les employés sont affichés
- Tous les départements aussi

Mostafa n'a pas de département → Nom_dep = NULL
Pas de ligne orpheline côté département, donc pas de ligne avec Nom = NULL

Table - Employes

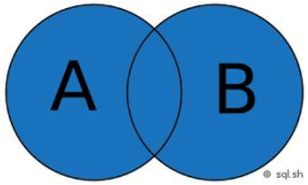
Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

Id	Nom	Age	Salaire	Profession	Dep	Id_dep	Nom_dep
2	Mohamed	30	8000.40	Directeur	1	1	Informatique
5	Omar	29	9000.00	Ingenieur	1	1	Informatique
1	Ismail	25	6000.00	Assistant	2	2	RH
3	Fatima	29	6000.00	Directeur	3	3	Vente
4	Dounia	30	7000.00	Assistant	4	4	Strategies
7	Mostafa	29	9000.00	Ingenieur	NULL	NULL	NULL

FULL JOIN



Le FULL OUTER JOIN = LEFT JOIN + RIGHT JOIN

⚠️ mySQL ne supporte pas FULL OUTER JOIN, il faut faire une union d'un LEFT et d'un RIGHT JOIN

```sql
SELECT e.Nom, e.Profession, d.Nom_dep
FROM Employes e
LEFT JOIN Departement d ON e.Dep = d.Id_dep

UNION

SELECT e.Nom, e.Profession, d.Nom_dep
FROM Employes e
RIGHT JOIN Departement d ON e.Dep = d.Id_dep;
```

Le mot-clé UNION retire les doublons par défaut.

- UNION = fusionne deux résultats et supprime les doublons
- UNION ALL = fusionne les deux résultats en gardant tous les doublons

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

Table - Departement

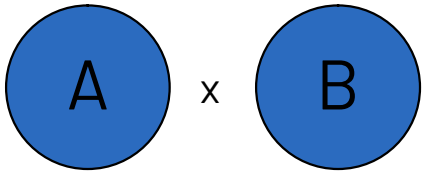
Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies


```sql
SELECT e.Nom, e.Profession, d.Nom_dep
FROM Employes e
LEFT JOIN Departement d ON e.Dep = d.Id_dep

UNION

SELECT e.Nom, e.Profession, d.Nom_dep
FROM Employes e
RIGHT JOIN Departement d ON e.Dep = d.Id_dep;
```

CROSS JOIN



Une CROSS JOIN (ou produit cartésien) retourne toutes les combinaisons possibles entre les lignes des deux tables.
Si table1 contient 3 lignes et table2 en contient 4, la CROSS JOIN renverra 3 × 4 = 12 lignes.

```sql
SELECT *
FROM table1
CROSS JOIN table2;
```

```sql
SELECT *
FROM table1, table2;
```

⚠️ Utilisée rarement seule, mais utile pour générer toutes les combinaisons possibles ou dans des calculs statistiques.

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

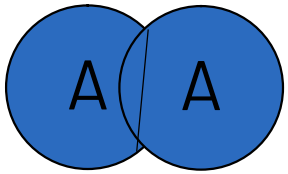
Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

```sql
SELECT *
FROM Departement
CROSS JOIN Employes
```

Id_dep	Nom_dep	Id	Nom	Age	Salaire	Profession	Dep
1	Informatique	1	Ismail	25	6000.00	Assistant	2
2	RH	1	Ismail	25	6000.00	Assistant	2
3	Vente	1	Ismail	25	6000.00	Assistant	2
4	Strategies	1	Ismail	25	6000.00	Assistant	2
1	Informatique	2	Mohamed	30	8000.40	Directeur	1
2	RH	2	Mohamed	30	8000.40	Directeur	1
3	Vente	2	Mohamed	30	8000.40	Directeur	1
4	Strategies	2	Mohamed	30	8000.40	Directeur	1
1	Informatique	3	Fatima	29	6000.00	Directeur	3
2	RH	3	Fatima	29	6000.00	Directeur	3
3	Vente	3	Fatima	29	6000.00	Directeur	3
4	Strategies	3	Fatima	29	6000.00	Directeur	3
1	Informatique	4	Dounia	30	7000.00	Assistant	4
2	RH	4	Dounia	30	7000.00	Assistant	4
3	Vente	4	Dounia	30	7000.00	Assistant	4
4	Strategies	4	Dounia	30	7000.00	Assistant	4
1	Informatique	5	Omar	29	9000.00	Ingenieur	1
2	RH	5	Omar	29	9000.00	Ingenieur	1
3	Vente	5	Omar	29	9000.00	Ingenieur	1
4	Strategies	5	Omar	29	9000.00	Ingenieur	1
1	Informatique	7	Mostafa	29	9000.00	Ingenieur	NULL
2	RH	7	Mostafa	29	9000.00	Ingenieur	NULL
3	Vente	7	Mostafa	29	9000.00	Ingenieur	NULL
4	Strategies	7	Mostafa	29	9000.00	Ingenieur	NULL

SELF JOIN



SELF JOIN est utilisée pour joindre une table à elle-même comme si la table était deux tables

```sql
SELECT e.nom AS Employé, m.nom
AS Manager
FROM employes e
LEFT JOIN employes m ON
e.manager_id = m.id;
```

Table - Employes

Id	Nom	Age	Salaire	Profession	Dep
1	Ismail	25	6000.00	Assistant	2
2	Mohamed	30	8000.40	Directeur	1
3	Fatima	29	6000.00	Directeur	3
4	Dounia	30	7000.00	Assistant	4
5	Omar	29	9000.00	Ingenieur	1
7	Mostafa	29	7500.00	Ingenieur	NULL

Table - Departement

Id_dep	Nom_dep
1	Informatique
2	RH
3	Vente
4	Strategies

```sql
SELECT *
FROM Employes AS T1, Employes AS T2
WHERE T1.Salaire >T2.Salaire;
```

Id	Nom	Age	Salaire	Profession	Dep	Id	Nom	Age	Salaire
2	Mohamed	30	8000.40	Directeur	1	1	Ismail	25	6000.00
4	Dounia	30	7000.00	Assistant	4	1	Ismail	25	6000.00
5	Omar	29	9000.00	Ingenieur	1	1	Ismail	25	6000.00
7	Mostafa	29	9000.00	Ingenieur	NULL	1	Ismail	25	6000.00
5	Omar	29	9000.00	Ingenieur	1	2	Mohamed	30	8000.40
7	Mostafa	29	9000.00	Ingenieur	NULL	2	Mohamed	30	8000.40
2	Mohamed	30	8000.40	Directeur	1	3	Fatima	29	6000.00
4	Dounia	30	7000.00	Assistant	4	3	Fatima	29	6000.00
5	Omar	29	9000.00	Ingenieur	1	3	Fatima	29	6000.00
7	Mostafa	29	9000.00	Ingenieur	NULL	3	Fatima	29	6000.00
2	Mohamed	30	8000.40	Directeur	1	4	Dounia	30	7000.00
5	Omar	29	9000.00	Ingenieur	1	4	Dounia	30	7000.00
7	Mostafa	29	9000.00	Ingenieur	NULL	4	Dounia	30	7000.00

ATELIER

Pratiquer les requêtes AcademyAFEC

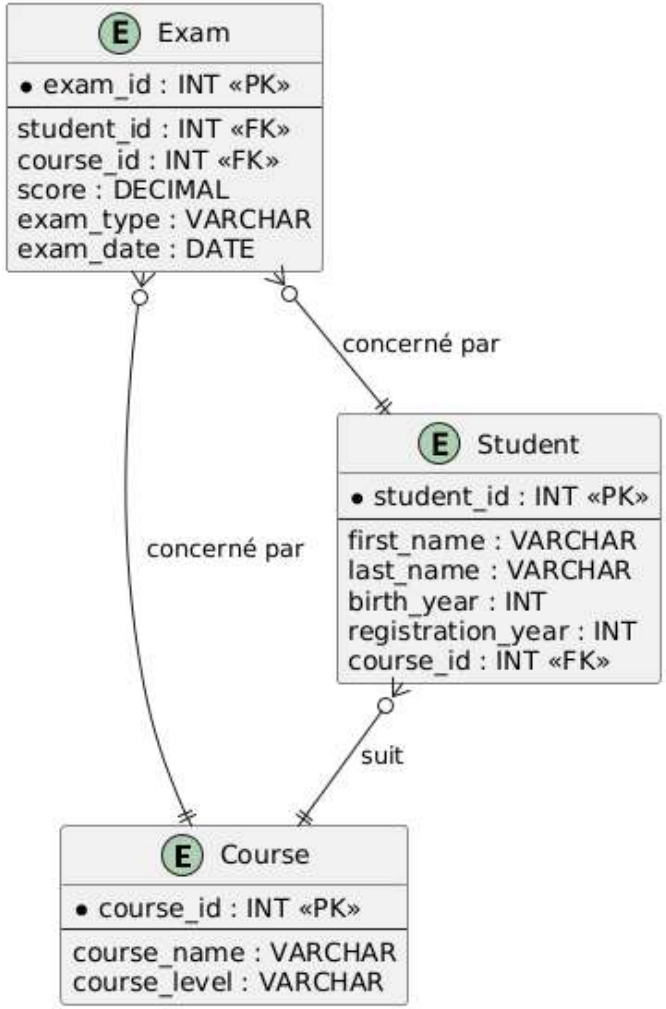
CONSIGNES

git@github.com:dessna12/05-Joins-SQL.git

Astuce pour éxécuter le fichier (attention pas de antislach ni de ‘single’ or “double quote”)

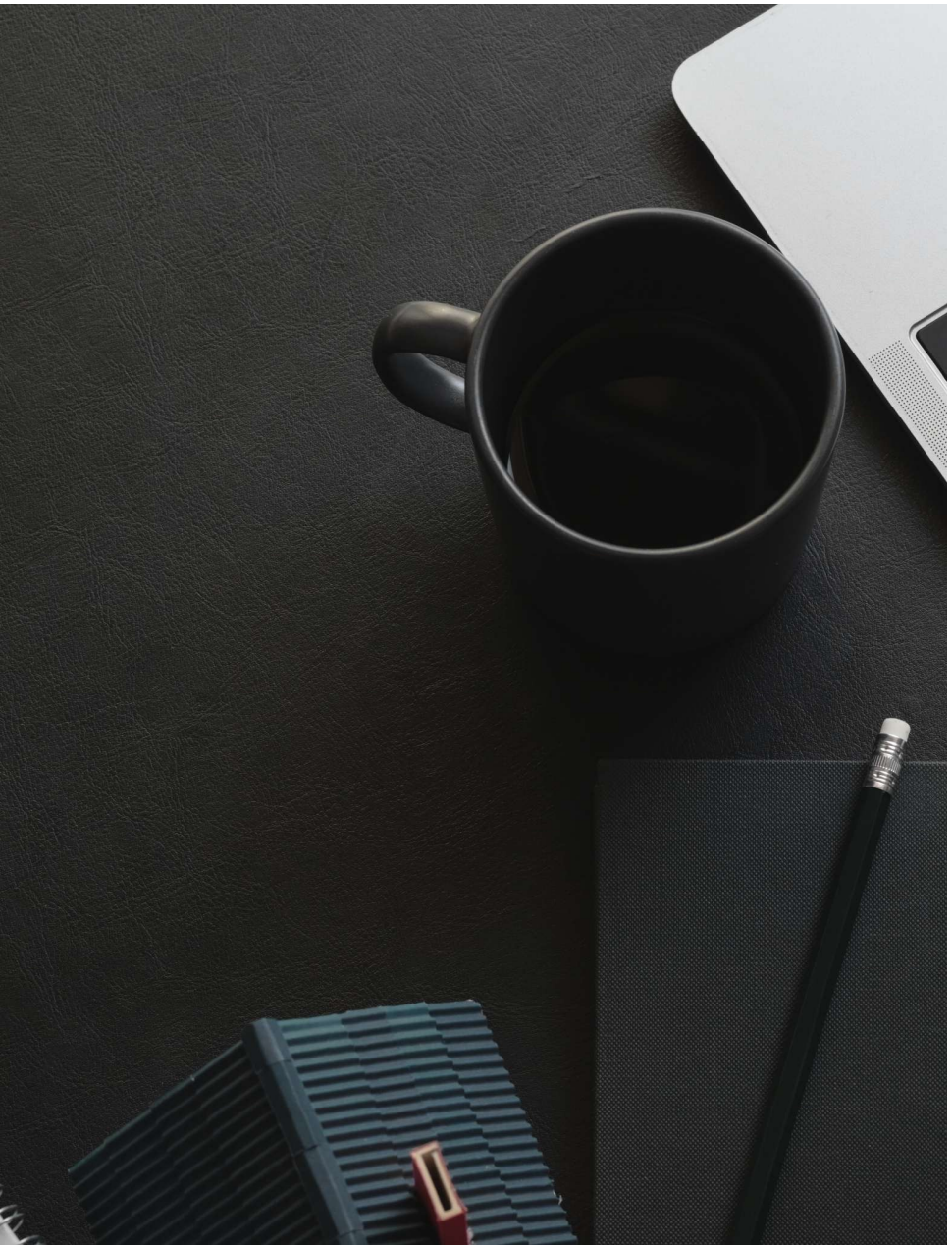
```
mysql > source chemin/vers/fichier.sql;
```

MLD


Exam
• exam_id : INT «PK»
student_id : INT «FK»
course_id : INT «FK»
score : DECIMAL
exam_type : VARCHAR
exam_date : DATE
concerné par
concerné par
Student
• student_id : INT «PK»
first_name : VARCHAR
last_name : VARCHAR
birth_year : INT
registration_year : INT
course_id : INT «FK»
suit
Course
• course_id : INT «PK»
course_name : VARCHAR
course_level : VARCHAR

MEMO

Jointure	Description	Exemple SQL MySQL
INNER JOIN	Lignes avec correspondance dans A & B	SELECT * FROM A INNER JOIN B ON A.id = B.a_id;
LEFT JOIN	Toutes lignes A + correspondances B	SELECT * FROM A LEFT JOIN B ON A.id = B.a_id;
RIGHT JOIN	Toutes lignes B + correspondances A	SELECT * FROM A RIGHT JOIN B ON A.id = B.a_id;
FULL OUTER JOIN	Toutes lignes A et B (simulé)	SELECT * FROM A LEFT JOIN B ON A.id = B.a_id UNION SELECT * FROM A RIGHT JOIN B ON A.id = B.a_id;
SELF JOIN	Jointure d'une table avec elle-même	SELECT a.name, b.name FROM Employee a INNER JOIN Employee b ON a.manager_id = b.id;
CROSS JOIN	Produit cartésien	SELECT * FROM A CROSS JOIN B;



SOUS REQUÊTES

LES SOUS REQUÊTES

Une **sous-requête** (ou subquery) est une requête SQL **imbriquée** à l'**intérieur** d'une autre.

Elle permet de filtrer, transformer ou produire un jeu de résultats **intermédiaires**.

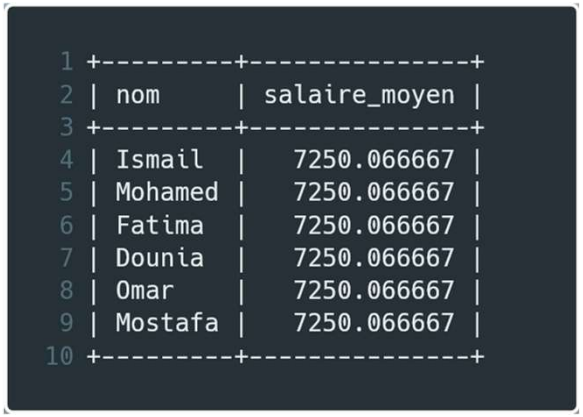
Les sous-requêtes constituent un moyen simple et efficace de gérer les requêtes qui dépendent des résultats d'une autre requête. Elles sont presque identiques aux instructions SELECT normales, mais il existe quesques restrictions :

- Une sous-requête doit toujours apparaître entre parenthèses.
- Une sous-requête doit renvoyer une seule colonne. Cela signifie que vous ne pouvez pas utiliser SELECT * dans une sous-requête à moins que la table à laquelle vous faites référence ne comporte qu'une seule colonne. Vous pouvez utiliser une sous-requête qui renvoie plusieurs colonnes si le but est la comparaison de lignes.

SOUS REQUÊTES EN SELECT

Objectif : Permet de calculer une valeur
pour chaque ligne.

```sql
SELECT nom,
    (SELECT AVG(salaire) FROM Employes) AS salaire_moyen
FROM Employes;
```



```
+---------+---------------+
| nom     | salaire_moyen |
+---------+---------------+
| Ismail  |   7250.066667 |
| Mohamed |   7250.066667 |
| Fatima  |   7250.066667 |
| Dounia  |   7250.066667 |
| Omar    |   7250.066667 |
| Mostafa |   7250.066667 |
+---------+---------------+
```

➡️ Affiche chaque employé avec le salaire moyen global.

Utilisée pour ajouter des colonnes calculées.

SOUS REQUÊTES EN FROM

Utilisée comme une table temporaire.

```sql
SELECT dept, moyenne
FROM (
  SELECT dep AS dept, AVG(salaire) AS moyenne
  FROM Employes
  GROUP BY dep
) AS dep_stats;
```

```
+------+-------------+
| dept | moyenne     |
+------+-------------+
| NULL | 7500.000000 |
|    1 | 8500.200000 |
|    2 | 6000.000000 |
|    3 | 6000.000000 |
|    4 | 7000.000000 |
+------+-------------+
```

➡️ La sous-requête devient une table virtuelle nommée dep_stats

SOUS REQUÊTES WHERE

Filtre les résultats selon un critère produit par une sous-requête.

```sql
SELECT nom
FROM Employes
WHERE salaire > (
  SELECT AVG(salaire)
  FROM Employes
);
```

```
+---------+
| nom     |
+---------+
| Mohamed |
| Omar    |
| Mostafa |
+---------+
```

➡️ Récupère les employés dont le salaire est au-dessus de la moyenne.

ATELIER

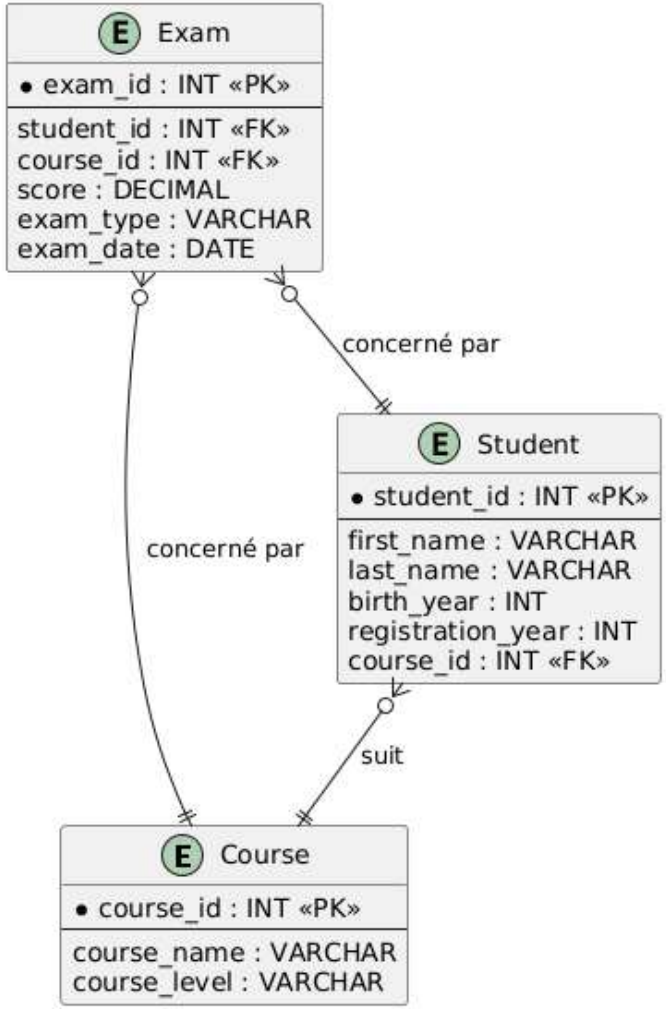
Pratiquer les sous-requêtes AcademyAFEC

ENONCE

git@github.com:dessna12/05-Joins-SQL.git

Se placer sur la branche 02-subRequest

MLD


Exam
• exam_id : INT «PK»
student_id : INT «FK»
course_id : INT «FK»
score : DECIMAL
exam_type : VARCHAR
exam_date : DATE
concerné par
concerné par
Student
• student_id : INT «PK»
first_name : VARCHAR
last_name : VARCHAR
birth_year : INT
registration_year : INT
course_id : INT «FK»
suit
Course
• course_id : INT «PK»
course_name : VARCHAR
course_level : VARCHAR

QUIZZ