

SQL

Faire **vivre** la base de données : **créer**,
lire, **remplacer**, **supprimer** des
informations



Compétence demandée :
Savoir écrire des requêtes SQL

Pour vivre de l'information (base de données, REST API, SOAP), il faut des opérations basiques : les opérations
CRUD

FRANCAIS	ANGLAIS
Créer	
Lire	
Remplacer	
Supprimer	

FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	UPDATE
Supprimer	DELETE

FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	UPDATE
Supprimer	DELETE

FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	UPDATE
Supprimer	DELETE

Opérations **CRUD**

Opérations **CRUD** pour les lignes

FRANCAIS	ANGLAIS	SQL
Créer	C REATE	I NSERT
Lire	R EAD	S ELECT
Remplacer	U PPDATE	U PPDATE
Supprimer	D ELETE	D ELETE

SQL : Structured Query Language

Deviner, c'est gagner !

proprietaires

id	nom	prenom	age	poids
1				
2				
3				
4				
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1				
2				
3				
4				
5				
6				
7				
8				

proprietaires

id	nom	prenom	age	poids
1				
2				
3				
4				
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1				
2				
3				
4				
5				
6				
7				
8				

A attributes / colonnes

```
SELECT nom FROM proprietaires;
```

proprietaires

id	nom	prenom	age	poids
1				
2				
3				
4				
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1				
2				
3				
4				
5				
6				
7				
8				

```
SELECT nom FROM proprietaires WHERE id = 3;
```

proprietaires

id	nom	prenom	age	poids
1				
2				
3				
4				
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1				
2				
3				
4				
5				1
6				
7				
8				4

```
SELECT nom FROM proprietaires WHERE proprietaire_id = 3;
```

INSERT INTO [table]([colonnes]) VALUES ([valeurs])
- UPDATE [table] SET [colonne] = [valeur] WHERE [lignes]
- DELETE FROM [table] WHERE [lignes]
SELECT [colonnes] FROM [table] WHERE [lignes]

DELETE / UPDATE
WHERE



```
SELECT [colonnes ou fonctions] FROM  
[combinaison_de_tables] WHERE [lignes]  
GROUP BY [nom_colonne] HAVING [groupes]  
ORDER BY [colonne] LIMIT [nombre]
```

FROM [table]

WHERE [lignes]

[table] LEFT OUTER JOIN
[table] ON [égalité]

[table] INNER JOIN [table]
ON [égalité]

GROUP BY [colonne]

HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

LIMIT 0,3
LIMIT 3 OFFSET 0

1. SELECT basique (`FROM`, `WHERE`)
2. SELECT groupé (`GROUP BY`)
3. SELECT ordonné limité (`ORDER BY`, `LIMIT`)
4. SELECT groupé conditionné (`HAVING`)
5. SELECT avec jointures (`LEFT OUTER JOIN`, `INNER JOIN`)

Basique

FROM, WHERE

FROM [table]

WHERE [lignes]

1^{er} volet de la saga :

SELECT [colonnes] FROM [table] WHERE [pédicats(s)]

Les groupes

GROUP BY

2^{ème} volet de la saga :

```
SELECT [colonnes ou fonctions] FROM [table]  
WHERE [prédicat(s)] GROUP BY [colonne]
```

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

```
SELECT date, _____ FROM factures GROUP BY date;
```

```
SELECT date, AVG(montant) FROM factures GROUP BY date;
```

AVG()
COUNT()
MAX()
MIN()
SUM()

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

SELECT date, _____ AS measure FROM factures GROUP BY date;

SELECT date, SUM(montant) AS measure FROM factures GROUP BY date;

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

GROUPE #1
date : 2020-05-21

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

GROUPE #2
date : 2020-05-22

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

GROUPE #3
date : 2020-05-23

id	date	montant
7	2020-05-23	20.0

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

GROUPE #1
date : 2020-05-21

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

GROUPE #2
date : 2020-05-22

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

GROUPE #3
date : 2020-05-23

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21
date, SUM(montant)
2020-05-21, 180.0

GROUPE #2
date : 2020-05-22

GROUPE #3
date : 2020-05-23

SELECT date, SUM(montant) FROM factures GROUP BY date;

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21
date, SUM(montant)
2020-05-21, 180.0

GROUPE #2
date : 2020-05-22
date, SUM(montant)
2020-05-22, 50.0

GROUPE #3
date : 2020-05-23

SELECT date, SUM(montant) FROM factures GROUP BY date;

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

id	date	montant
7	2020-05-23	20.0

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

GROUPE #1
date : 2020-05-21
date, SUM(montant)
2020-05-21, 180.0

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

GROUPE #2
date : 2020-05-22
date, SUM(montant)
2020-05-22, 50.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

GROUPE #3
date : 2020-05-23
date, SUM(montant)
2020-05-23, 20.0

id	date	montant
7	2020-05-23	20.0

```
SELECT date, SUM(montant) FROM factures GROUP BY date;
```

date	SUM(montant)
2020-05-21	180.0
2020-05-22	50.0
2020-05-23	20.0

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

GROUPE #2
date : 2020-05-22

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

GROUPE #3
date : 2020-05-23

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

GROUPE #2
date : 2020-05-22

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

GROUPE #3
date : 2020-05-23

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21
id, SUM(montant)
???, 180.0

GROUPE #2
date : 2020-05-22

GROUPE #3
date : 2020-05-23

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21
id, SUM(montant)
???, 180.0

GROUPE #2
date : 2020-05-22

GROUPE #3
date : 2020-05-23

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

id	date	montant
7	2020-05-23	20.0

GROUPE #1
date : 2020-05-21
id, SUM(montant)
???, 180.0

GROUPE #2
date : 2020-05-22

GROUPE #3
date : 2020-05-23

```
SELECT id, SUM(montant) FROM factures GROUP BY date;
```

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0

id	date	montant
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0

id	date	montant
7	2020-05-23	20.0

Quel est **le montant total** des factures **par** date ?

1. On groupe **par** quoi ? date
2. Quelle est la fonction d'agrégat pour sélectionner « **le montant total** » pour 1 **date** donnée

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

Ordonné et limité

ORDER BY, LIMIT

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

ORDER BY [colonne]

LIMIT [nombre]

3^{ème} volet de la saga :

```
SELECT [colonnes ou fonctions] FROM [table] WHERE [prédicat(s)]  
      GROUP BY [colonne]  
      ORDER BY [colonne] LIMIT [nombre]
```

ORDER BY	ASC ou DESC
LIMIT	

SELECT date FROM factures GROUP BY date ORDER BY date DESC LIMIT 10

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

ORDER BY [colonne]

LIMIT [nombre]

Conditions sur groupes

HAVING

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

4^{ème} volet de la saga :

```
SELECT [colonnes ou fonctions] FROM [table] WHERE [lignes]  
GROUP BY [nom_colonne] HAVING [groupes]  
ORDER BY [colonne] LIMIT [nombre]
```

AVG()
COUNT()
GROUP_CONCAT()
MAX()
MIN()
SUM()

id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

SELECT date FROM factures GROUP BY date HAVING ____ = 3;

SELECT date FROM factures GROUP BY date HAVING AVG(montant) > 36.6;

500
500

FROM [table]

WHERE [lignes]

GROUP BY [colonne]

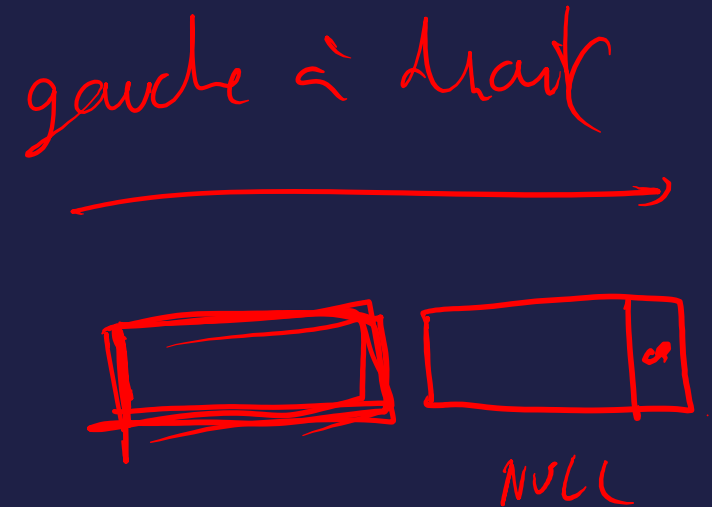
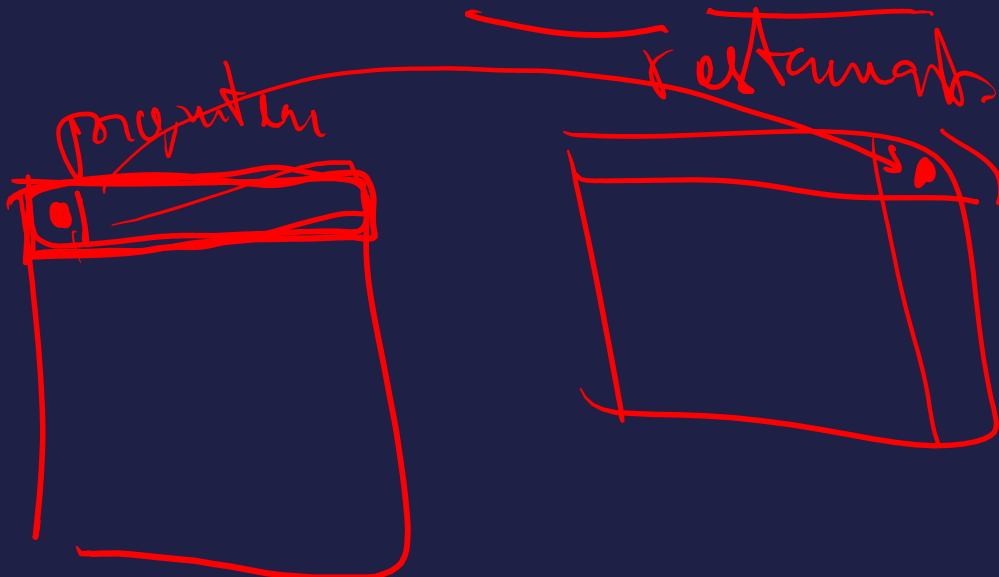
HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

Jointure

LEFT OUTER JOIN



FROM [table]

WHERE [lignes]

[table] LEFT OUTER JOIN
[table] ON [égalité]

[table] INNER JOIN [table]
ON [égalité]

GROUP BY [colonne]

HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

5^{ème} volet de la saga :

```
SELECT [colonnes ou fonctions] FROM [tables] WHERE [lignes]  
      GROUP BY [nom_colonne] HAVING [groupes]  
      ORDER BY [colonne] LIMIT [nombre]
```

proprietaires

restaurants

id	nom	prenom	age	poids
1		Michael		
2				
3				
4		Sophie		
5				
6				
7				
8				

id	nom	adresse	note	proprietaire_id
1	Raman			1
2				
3	Le ciel			4
4	Raimond			4
5	Bistro Toc			4
6				
7	Affine			1
8	Colin			1

proprietaires

id	nom	prenom	age	poids
1		Michael		
2				
3				
4		Sophie		
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1	Raman			1
2				
3	Le ciel			4
4	Raimond			4
5	Bistro Toc			4
6				
7	Affine			1
8	Colin			1

SELECT restaurants.nom FROM restaurants LEFT OUTER JOIN proprietaires ON proprietaires.id =
restaurants.proprietaire_id WHERE proprietaires.prenom = 'Michael';

SELECT restaurants.nom FROM proprietaires LEFT OUTER JOIN restaurants ON proprietaires.id = restaurants.proprietaire_id WHERE proprietaires.prenom = 'Michael';

r. proprietaire_id

p.id

proprietaires

id	nom	prenom	age	poids
1		Michael		
2				
3				
4		Sophie		
5				
6				
7				
8				

restaurants

id	nom	adresse	note	proprietaire_id
1	Raman			1
2				
3	Le ciel			4
4	Raimond			4
5	Bistro Toc			4
6				
7	Affine			1
8	Colin			1

```
SELECT r.nom FROM proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id WHERE  
p.prenom = 'Michael';
```

```
proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id
```

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id



p.id	p.nom	p.prenom	p.age	p.poids	r.id	r.nom	r.adresse	r.note	r.proprietaire_id
1		Michael			1	Raman			1
1		Michael			7	Affine			1
1		Michael			8	Colin			1
4		Sophie			3	Le ciel			4
4		Sophie			4	Raimond			4
4		Sophie			5	Bistro Toc			4

proprietaires
↓

restaurant

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire id

proprietaires
LEFT

restaurants
JOIN

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id

proprietaires

LEFT

restaurants

JOIN

id	nom	prenom	age	poids	id	nom	adresse	note	proprietaire_id
1		Michael			1	Raman			1
					7	Affine			1
					8	Colin			1

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id

proprietaires

LEFT

id	nom	prenom	age	poids
1		Michael		

restaurants

JOIN

id	nom	adresse	note	proprietaire_id
1	Raman			1
7	Affine			1
8	Colin			1

id	nom	prenom	age	poids
4		Sophie		

id	nom	adresse	note	proprietaire_id
3	Le ciel			4
4	Raimond			4
5	Bistro Toc			4

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id

proprietaires

LEFT

restaurants

JOIN

id	nom	prenom	age	poids	id	nom	adresse	note	proprietaire_id
1		Michael			1	Raman			1
1		Michael			7	Affine			1
1		Michael			8	Colin			1

id	nom	prenom	age	poids	id	nom	adresse	note	proprietaire_id
4		Sophie			3	Le ciel			4
4		Sophie			4	Raimond			4
4		Sophie			5	Bistro Toc			4

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id

proprietaires

LEFT

restaurants

JOIN

id	nom	prenom	age	poids	id	nom	adresse	note	proprietaire_id
1		Michael			1	Raman			1
1		Michael			7	Affine			1
1		Michael			8	Colin			1
4		Sophie			3	Le ciel			4
4		Sophie			4	Raimond			4
4		Sophie			5	Bistro Toc			4

proprietaires p LEFT OUTER JOIN restaurants r ON p.id = r.proprietaire_id

proprietaires

LEFT

restaurants

JOIN

p.id	p.nom	p.prenom	p.age	p.poids	r.id	r.nom	r.adresse	r.note	r.proprietaire_id
1		Michael			1	Raman			1
1		Michael			7	Affine			1
1		Michael			8	Colin			1
4		Sophie			3	Le ciel			4
4		Sophie			4	Raimond			4
4		Sophie			5	Bistro Toc			4

METHODOLOGIE

1. A-t-on besoin d'une ou de plusieurs tables ?
2. Quelles sont le **noms** de ces tables ?
3. Quelle est la relation entre les 2 deux ? Il me faut **ABSOLUMENT 2 COLONNES** dans des tables différentes

FROM [table]

WHERE [lignes]

[table] LEFT OUTER JOIN
[table] ON [égalité]

[table] INNER JOIN [table]
ON [égalité]

GROUP BY [colonne]

HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

FROM [table]

WHERE [lignes]

[table] LEFT OUTER JOIN
[table] ON [égalité]

[table] INNER JOIN [table]
ON [égalité]

GROUP BY [colonne]

HAVING [groupes]

ORDER BY [colonne]

LIMIT [nombre]

INNER JOIN vs OUTER JOIN ? STACKOVERFLOW

OUTER = Jointure **externe** (avec **NULL**)
INNER = Jointure **interne** (sans **NULL**)

Attention : dans les conditions WHERE, ne pas oubliez
« **IS NULL** » et non pas « = NULL »

