

## SQL

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





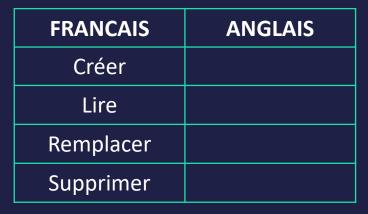
Compétence demandée : Savoir écrire des requêtes SQL



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









FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	UPDATE
Supprimer	DELETE



FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	<b>U</b> PDATE
Supprimer	DELETE



FRANCAIS	ANGLAIS
Créer	CREATE
Lire	READ
Remplacer	<b>U</b> PDATE
Supprimer	DELETE

### Opérations CRUD



# Opérations CRUD pour les lignes

FRANCAIS	ANGLAIS	SQL
Créer	CREATE	INSERT
Lire	READ	SELECT
Remplacer	UPDATE	UPDATE
Supprimer	DELETE	DELETE



# Deviner, c'est gagner!



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

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

restaurants



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

id	nom	adresse	note	proprietaire_id
		44.050		proprietane_ia
1				
2				
3				
4				
5				_
6				
7				
2				

restaurants

**SELECT** nom **FROM** proprietaires;



nom	prenom	age	poids
			+
			◆
	nom	nom prenom	nom prenom age

id	nom	adresse	note	proprietaire_id
		44.050		proprietane_ia
1				
2				
3				
4				
5				_
6				
7				
2				

restaurants

SELECT nom FROM proprietaires WHRE id = 3;

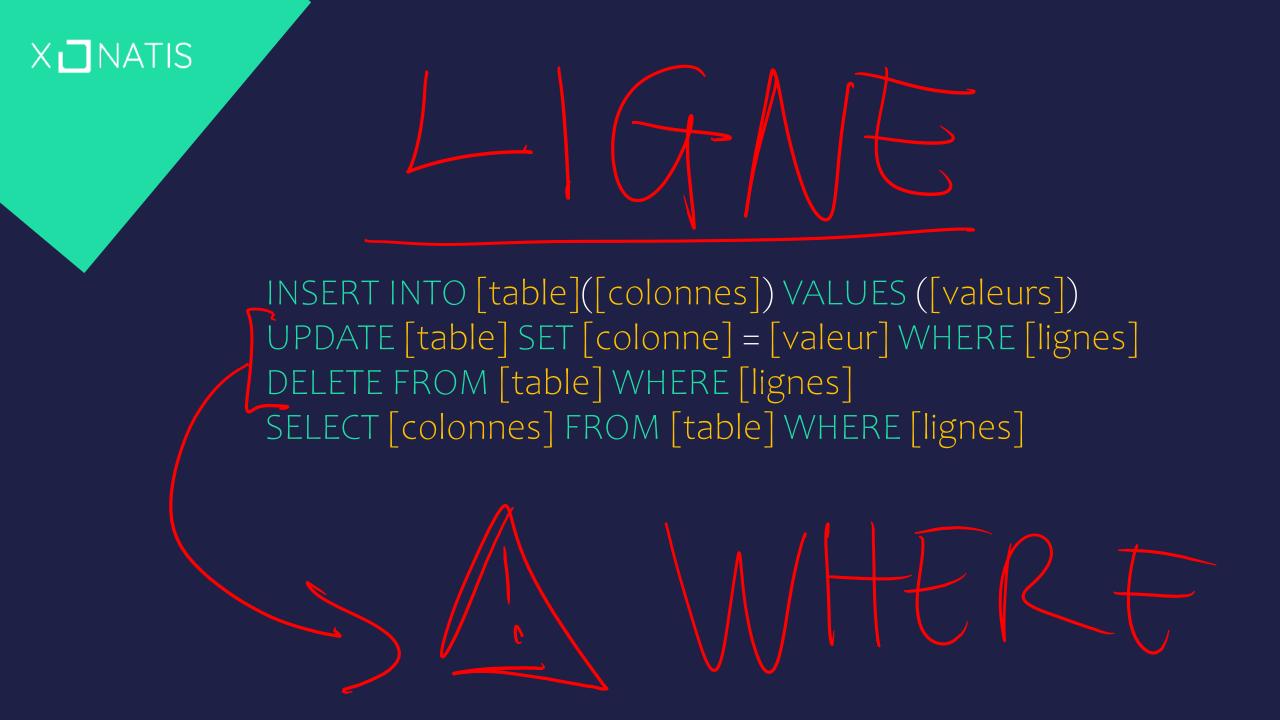


id	nom	prenom	age	poids
1				4
2				
3				
4				<b>←</b>
5				
6				
7				
8				

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

restaurants

ERREUR!





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] INNER JOIN [table] [table] ON [égalité] ON [égalité] **GROUP BY [colonne]** HAVING [groupes] ORDER BY [colonne] LIMIT [nombre]





- 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<sup>er</sup> volet de la saga:

SELECT [colonnes] FROM [table] WHERE [lignes]





### Les groupes GROUP BY



# 2<sup>ème</sup> volet de la saga:

SELECT [colonnes ou fonctions] FROM [table] WHERE [lignes] GROUP BY [nom\_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

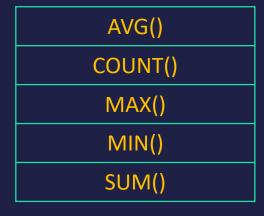


id	date	montant
1	2020-05-21	30.0
2	2020-05-21	60.0
3	2020-05-21	90.0
4	2020-05-22	10.0
5	2020-05-22	10.0
6	2020-05-22	30.0
7	2020-05-23	20.0

SELECT date, \_\_\_\_ FROM factures GROUP BY date;

SELECT date, AVG montant) AS 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, \_\_\_\_ AS mesure FROM factures GROUP BY date;

SELECT date, SUM(montant) AS mesure 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



id

GROUPE #1 date : 2020-05-21

 1
 2020-05-21
 30.0

 2
 2020-05-21
 60.0

 3
 2020-05-21
 90.0

date

montant

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

GROUPE #2

date: 2020-05-22

GROUPE #3

date: 2020-05-23

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

GROUPE #3 date : 2020-05-23

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

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
date, SUM(montant)
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

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



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



GROUPE #1 date : 2020-05-21

GROUPE #2 date : 2020-05-22

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

GROUPE #2

date: 2020-05-22

GROUPE #3

date: 2020-05-23

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

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

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)
777, 180,0

GROUPE #2 date : 2020-05-22

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



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<sup>ème</sup> volet de la saga:

SELECT [colonnes ou fonctions] FROM [table] WHERE [lignes]

GROUP BY [nom\_colonne]

ORDER BY [colonne] LIMIT [nombre]



ORDER BY

LIMIT

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



FROM [table]

WHERE [lignes]

**GROUP BY** [colonne]

ORDER BY [colonne]

LIMIT [nombre]





# Conditions sur groupes



FROM [table]

WHERE [lignes]

**GROUP BY [colonne]** 

HAVING [groupes]

ORDER BY [colonne]

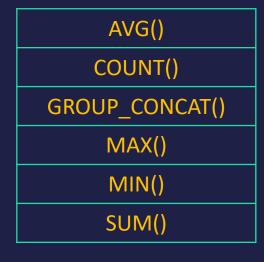
LIMIT [nombre]



### 4<sup>ème</sup> volet de la saga:

SELECT [colonnes ou fonctions] FROM [table] WHERE [lignes]
GROUP BY [nom\_colonne] HAVING [groupes]
ORDER BY [colonne] LIMIT [nombre]





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;



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] INNER JOIN [table] [table] ON [égalité] ON [égalité] **GROUP BY [colonne]** HAVING [groupes] ORDER BY [colonne] LIMIT [nombre]



# 5<sup>ème</sup> volet de la saga:

SELECT [colonnes ou fonctions] FROM [tables] WHERE [lignes]
GROUP BY [nom\_colonne] HAVING [groupes]
ORDER BY [colonne] LIMIT [nombre]



#### proprietaires

id	nom	prenom	age	poids
1		Michael		4
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

restaurants



proprietaires

id	nom	prenom	age	poids
1		Michael		<b>←</b>
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

restaurants





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

**LEFT** 

restaurants



#### proprietaires

**LEFT** 

#### restaurants

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



#### proprietaires

**LEFT** 

#### restaurants

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

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



#### proprietaires

**LEFT** 

#### restaurants

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

**LEFT** 

#### restaurants

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

**LEFT** 

#### restaurants

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] INNER JOIN [table] [table] ON [égalité] ON [égalité] **GROUP BY [colonne]** HAVING [groupes] ORDER BY [colonne] LIMIT [nombre]







FROM [table] WHERE [lignes] [table] LEFT OUTER JOIN [table] INNER JOIN [table] [table] ON [égalité] 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)

<u>Attention</u>: dans les conditions WHERE, ne pas oubliez « IS NULL » et non pas « = NULL »

