



# HTML & CSS



MySQL = SGBDR

(**S**ystème de **G**estion de **B**ase de **D**onnées **R**elationnelles)

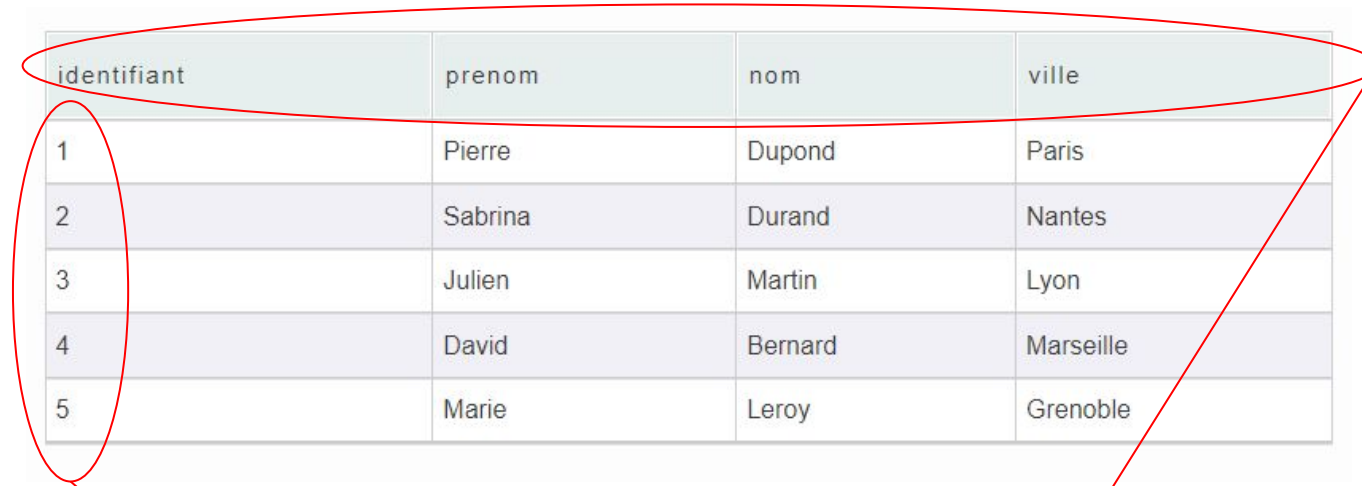
Gestion de base de données on nous allons pouvoir stocker des informations, les modifier, les récupérer et les supprimer.

# HTML & CSS

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

En plus on dit que MySQL est **R**elationnel car les données sont stockées sous forme de tableau (table).

# HTML & CSS



A diagram illustrating the structure of an HTML table. A table with four columns and five rows is shown. The first row is the header, with columns labeled 'identifiant', 'prenom', 'nom', and 'ville'. The subsequent four rows contain data. A red circle highlights the header row, and another red circle highlights the first column. Two red arrows originate from the text below: one points to the first column and the other points to the header row.

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

Une table se définit de 2 manières :

- Les colonnes (champs)
- Les lignes (enregistrements)

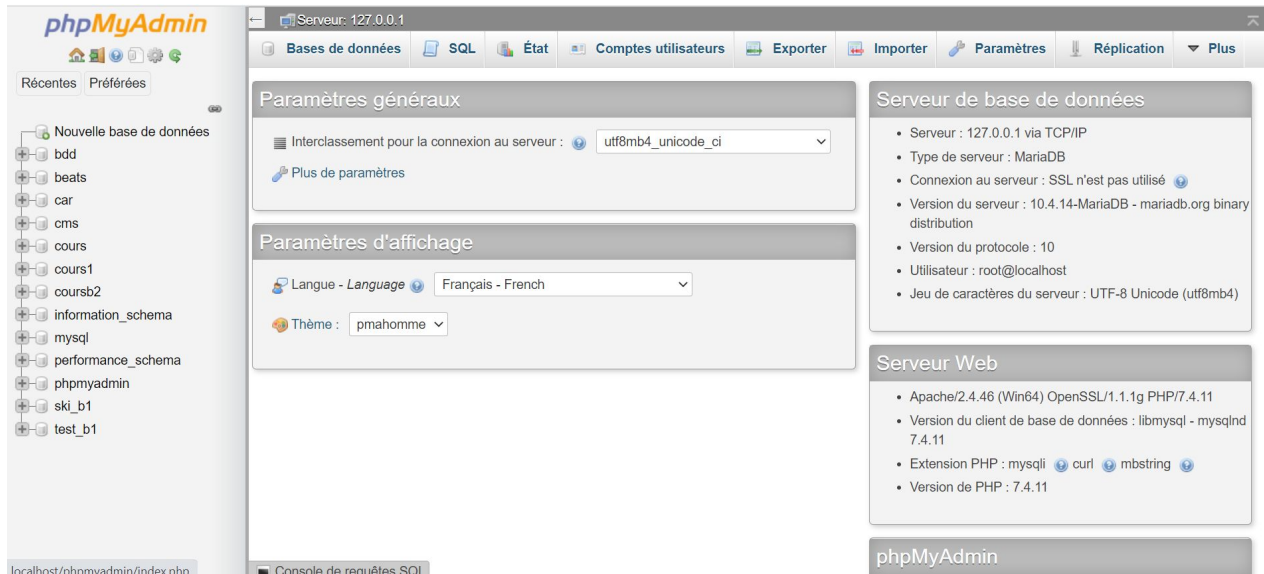
# HTML & CSS



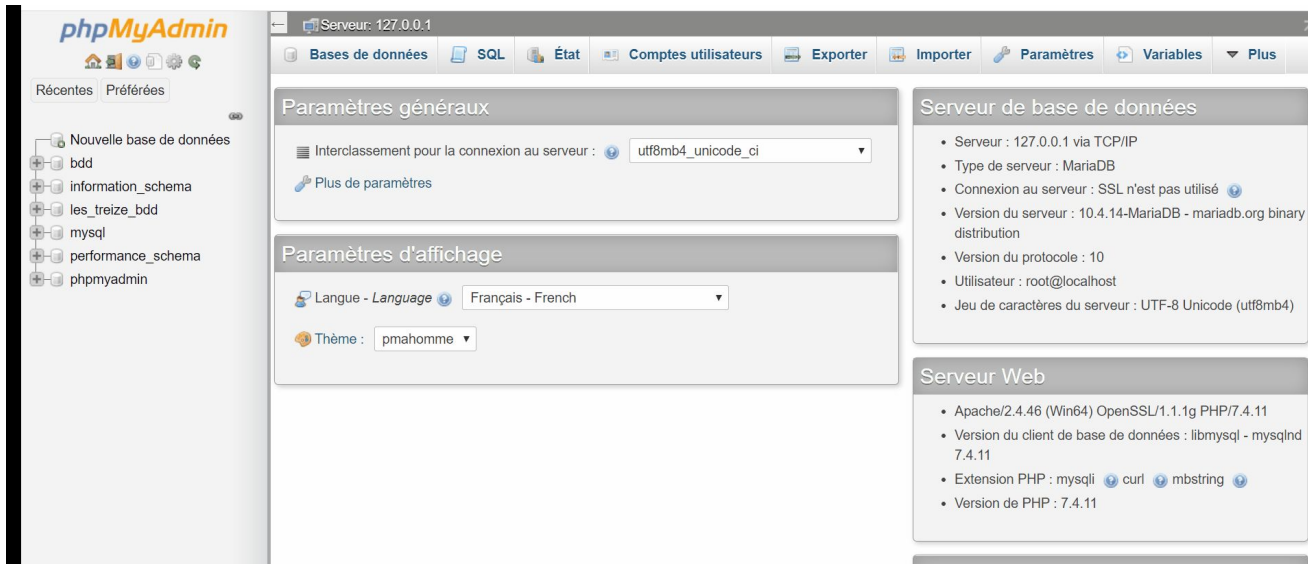
Il existe de nombreux SGBDR sur le marché, alors pourquoi apprendre MySQL plutôt qu'un autre ?

- Il est installé par défaut sur la plupart des serveurs car très répandu.
- MySQL a une grande communauté

# HTML & CSS



phpMyAdmin est une interface graphique développé en php (c'est dans le nom) et qui va nous aider à gérer nos base de données. Par défaut, MySQL ne se gère qu'en lignes de commande (un peu difficile quand on débute)



Il y a 2 façons de créer une base de données. Solution 1, grâce à l'interface graphique.

Dans le menu de gauche : “nouvelle base de données”

puis définir le nom de la base

Et enfin, choisir l'interclassement.

L'interclassement c'est le type de caractère supporté par la base. Par exemple, les signes arabes, russes, chinois, grecs, hébreux,... , n'ont pas la même table de caractères.

Nom : [REDACTED]

Prénom : Mickaël

Email : [REDACTED]

Âge : [REDACTED]

Dernière connexion : 2014-12-05 09:07:56



L'histoire de cet édifice commence en 1919, quand Albert Klein, marchand de meubles, achète son frère Bernard un terrain d'une superficie de 583 m<sup>2</sup> situé à Nogent-sur-Marne au 153 de la Grande Rue.

Il faudra peu de temps à ce passionné du 7<sup>ème</sup> art pour monter son projet : construire et exploiter une salle de spectacles cinématographiques.

Grâce aux fonds apportés par la famille Marcelot avec laquelle il s'associe au lendemain de l'acquisition du terrain, la construction du "Nogentais-Palace" commence, dirigée par l'ingénieur-architecte Milon.

Témoin de son époque et de l'originalité de son maître d'oeuvre, la façade de l'ouvrage, achevée en 1921 force l'admiration : larges baies vitrées, motifs d'ornementation, balcon travaillé en fonte. Elle sera inscrite, 70 ans plus tard, sur l'inventaire supplémentaire des Monuments Historiques.

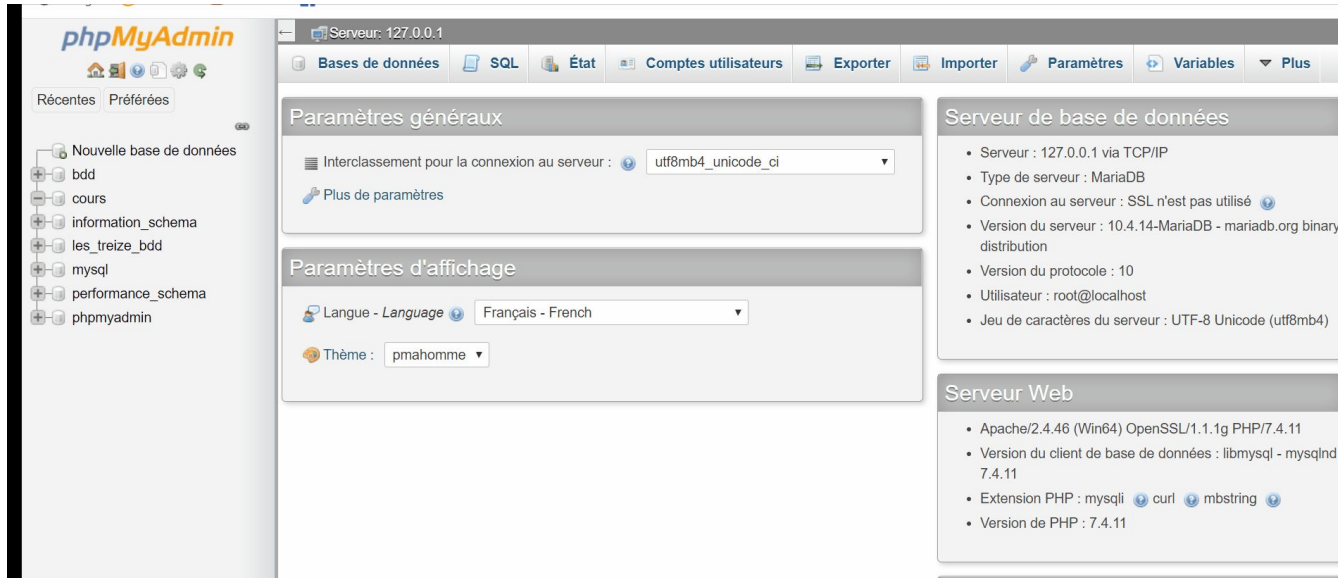
Choisir le bon interclassement permet d'éviter ce genre d'affichage.

UTF8\_general\_ci accepte quasiment tous les caractères.

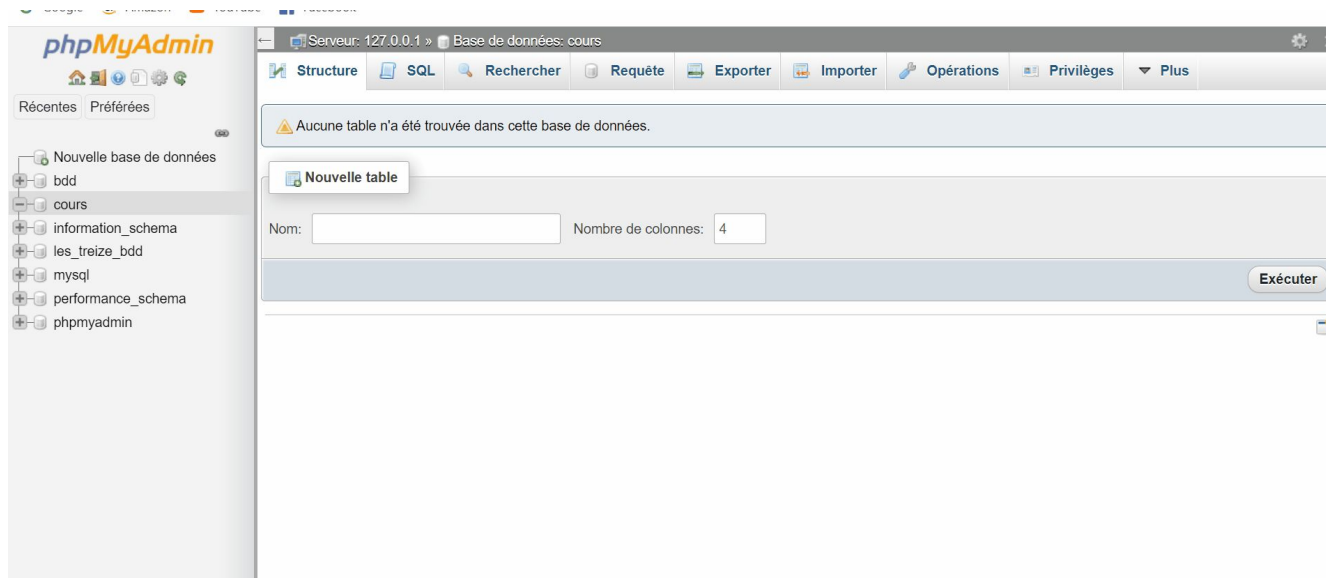
Après si vous êtes sûr de faire un site uniquement en turque par exemple, vous pouvez mettre utf8\_turkish\_ci.

utf8\_LangueVoulu\_ci

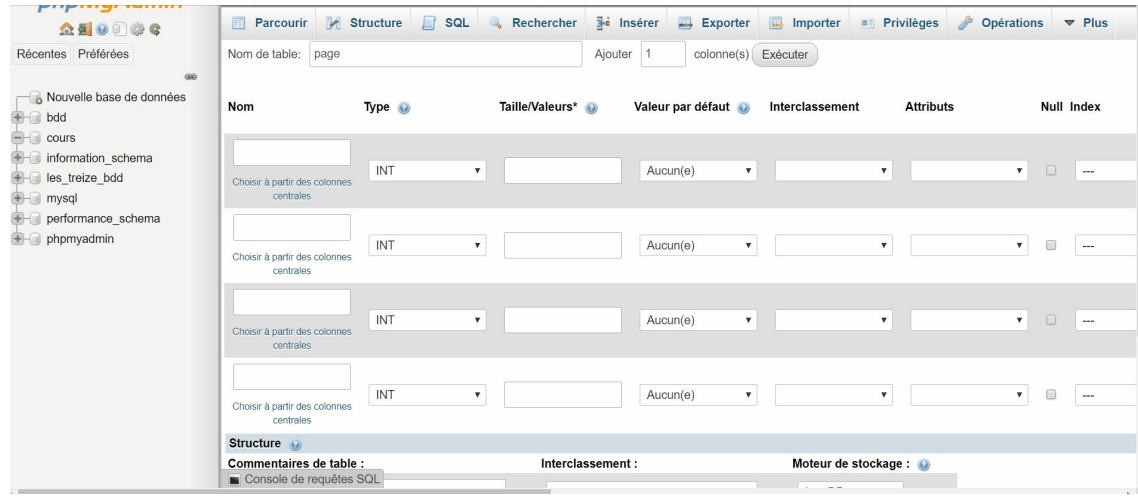




Ou alors, solution 2, directement en ligne de code. On voit que je n'ai pas définis l'interclassement et je ne vais pas rentrer dans le détail ici. Le but étant juste de vous faire voir que cette façon existait.



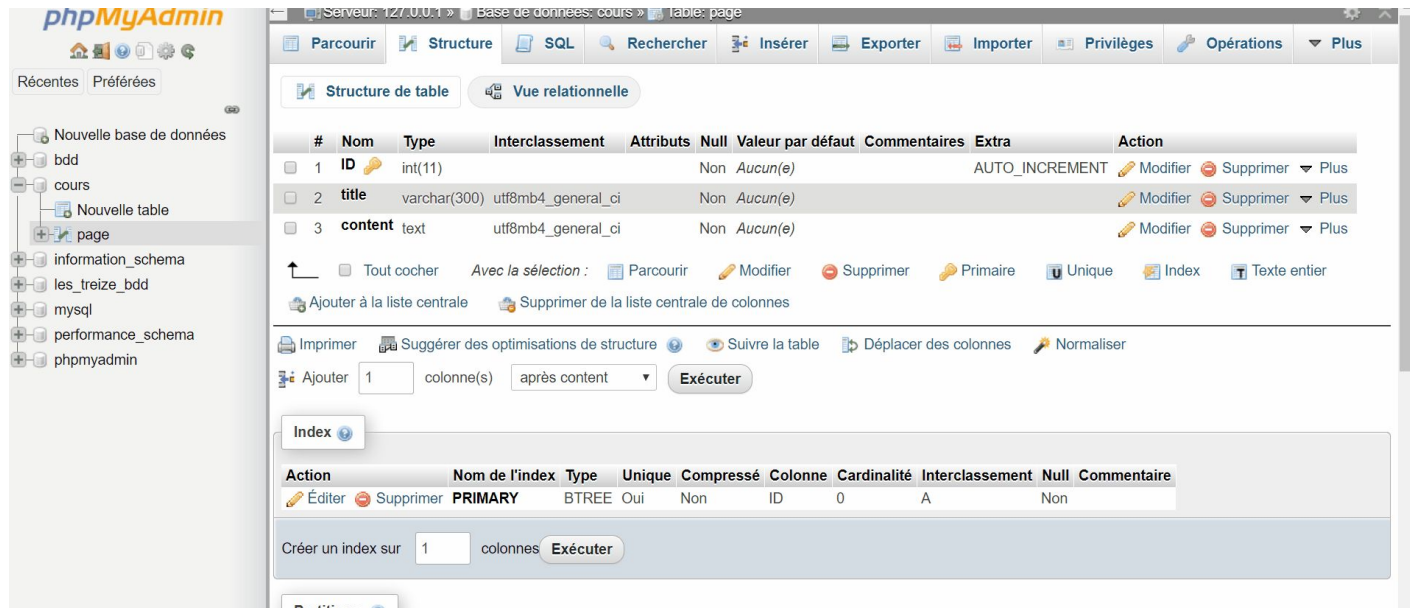
Nous allons créer notre première table. Il suffit de cliquer sur la BDD où nous voulons créer la table. Puis nommer cette table puis exécuter.



Nous allons commencer par créer un ID, avec le type INT. Ce champs a une particularité car il faut le mettre en “auto increment” (A\_I) et en clé primaire “primary”. Cela signifie que chaque ID sera unique. Même si vous supprimez une ligne avec un ID (ex: 4), il ne reviendra jamais.

Ensuite, deuxième champs sera “title” de type “VARCHAR”, c’est à dire “texte pas long”. Attention, quand on sélectionne ce type, il faut IMPÉRATIVEMENT le champs “taille/valeur”. Dans notre cas, je mets 300 (pas plus de 300 caractères).

Et pour finir, un champs “content” de type “TEXT”, c’est comme “VARCHAR” mais sans limite.

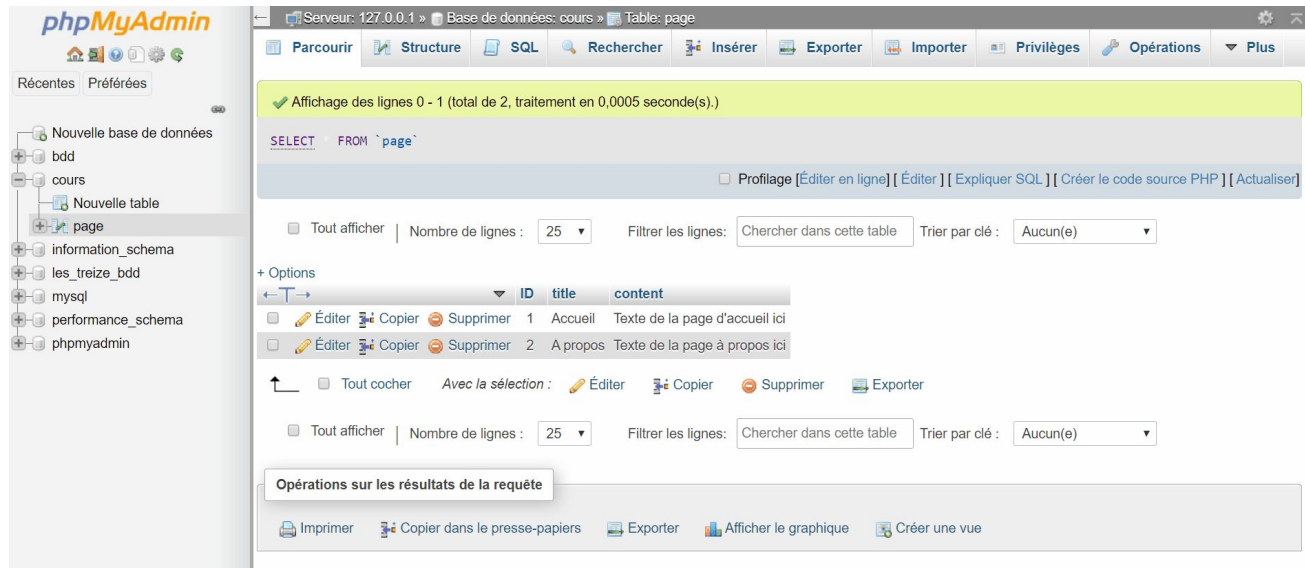


Vous pouvez à tout moment ajouter / modifier/ supprimer un champs dans l'onglet "structure" (en haut)

```
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)
```

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, ...  
VALUES ('valeur 1', 'valeur 2', ...)
```

```
INSERT INTO client (prenom, nom, ville, age)  
VALUES  
('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24),  
('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),  
('Marielle', 'Ribeiro', 'Maillères', 27),  
('Hilaire', 'Savary', 'Conie-Molitar', 58);
```



Il est également d'utiliser cette syntaxe (plus claire) qui est héritée de la commande de mise à jour (update). Seule soucis, c'est que cette syntaxe n'est pas normée et ne fonctionnera pas sur les autres SGBDR.

```
INSERT INTO `page` set title='mon titre', content='mon contenu';
```

Table "utilisateur" :

id	nom	prenom	date_inscription
1	Bazin	Daniel	2012-02-13
2	Favre	Constantin	2012-04-03
3	Clerc	Guillaume	2012-04-12
4	Ricard	Rosemonde	2012-06-24
5	Martin	Natalie	2012-07-02

```
DELETE FROM `utilisateur`  
WHERE `id` = 1
```

id	nom	prenom	date_inscription
2	Favre	Constantin	2012-04-03
3	Clerc	Guillaume	2012-04-12
4	Ricard	Rosemonde	2012-06-24
5	Martin	Natalie	2012-07-02

```
DELETE FROM `table`  
WHERE condition
```

Table "client" :

id	nom	rue	ville	code_postal	pays
1	Chantal	12 Avenue du Petit Trianon	Puteaux	92800	France
2	Pierre	18 Rue de l'Allier	Ponthion	51300	France
3	Romain	3 Chemin du Chiron	Trévérien	35190	France

id	nom	rue	ville	code_postal	pays
1	Chantal	12 Avenue du Petit Trianon	Puteaux	92800	France
2	Pierre	49 Rue Ameline	Saint-Eustache-la-Forêt	76210	France
3	Romain	3 Chemin du Chiron	Trévérien	35190	France

```
UPDATE client
SET rue = '49 Rue Ameline',
    ville = 'Saint-Eustache-la-Forêt',
    code_postal = '76210'
WHERE id = 2
```

```
UPDATE table
SET nom_colonne_1 = 'nouvelle valeur'
WHERE condition
```



Table "client" :

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

```
SELECT ville FROM client
```

ville
Paris
Nantes
Lyon
Marseille
Grenoble

```
SELECT nom_du_champ FROM nom_du_tableau
```

Table "client" :

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

```
SELECT prenom, nom FROM client
```

prenom	nom
Pierre	Dupond
Sabrina	Durand
Julien	Martin
David	Bernard
Marie	Leroy

Table "client" :

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

`SELECT * FROM client`

identifiant	prenom	nom	ville
1	Pierre	Dupond	Paris
2	Sabrina	Durand	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

id	nom	prenom	date_inscription	tarif_total
1	Durand	Maurice	2012-02-05	145
2	Dupond	Fabrice	2012-02-07	65
3	Durand	Fabienne	2012-02-13	90
4	Dubois	Chloé	2012-02-16	98
5	Dubois	Simon	2012-02-23	27

```
SELECT *
FROM utilisateur
ORDER BY nom
```

id	nom	prenom	date_inscription	tarif_total
4	Dubois	Chloé	2012-02-16	98
5	Dubois	Simon	2012-02-23	27
2	Dupond	Fabrice	2012-02-07	65
1	Durand	Maurice	2012-02-05	145
3	Durand	Fabienne	2012-02-	

```
SELECT colonne1, colonne2
FROM table
ORDER BY colonne1
```

id	nom	prenom	date_inscription	tarif_total
1	Durand	Maurice	2012-02-05	145
2	Dupond	Fabrice	2012-02-07	65
3	Durand	Fabienne	2012-02-13	90
4	Dubois	Chloé	2012-02-16	98
5	Dubois	Simon	2012-02-23	27

```
SELECT *
FROM utilisateur
ORDER BY nom, date_inscription DESC
```

id	nom	prenom	date_inscription	tarif_total
5	Dubois	Simon	2012-02-23	27
4	Dubois	Chloé	2012-02-16	98
2	Dupond	Fabrice	2012-02-07	65
3	Durand	Fabienne	2012-02-13	90
1	Durand	Maurice	2012-02-05	145

Requête SQL pour chercher les enregistrements pour lesquels la colonne “nom\_de\_colonne” débute par la lettre “a” :

```
SELECT *  
FROM table  
WHERE nom_de_colonne LIKE 'a%'
```

Requête SQL cherchant les enregistrements terminant par la lettre “a” :

```
[...] WHERE colonne LIKE '%a'
```

Requête SQL cherchant les enregistrements qui possède la lettre “a” :

```
[...] WHERE colonne LIKE '%a%'
```

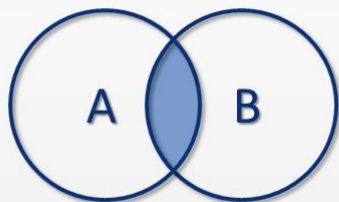
Requête SQL cherchant les enregistrements qui possède la lettre “a” en 2ème caractère :

```
[...] WHERE colonne LIKE '_a%'
```

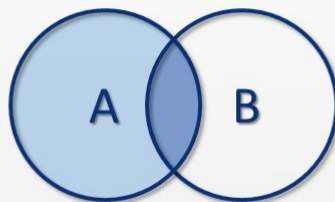
La syntaxe commune aux principales système de gestion de bases de données est la suivante :

```
SELECT *  
FROM table  
LIMIT 10
```

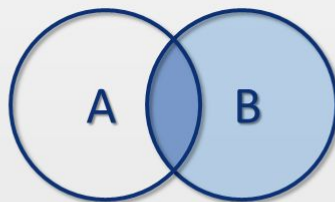
Cette requête permet de récupérer seulement les 10 premiers résultats d'une table. Bien entendu, si la table contient moins de 10 résultats, alors la requête retournera toutes les lignes.



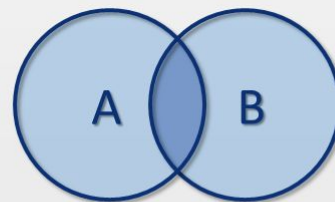
**INNER JOIN**



**LEFT OUTER JOIN**



**RIGHT OUTER JOIN**



**FULL OUTER JOIN**



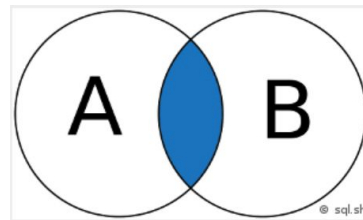
Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat, num_facture, prix_total
FROM utilisateur
INNER JOIN commande ON utilisateur.id = commande.utilisateur_id
```



```
SELECT *
FROM table1
INNER JOIN table2
WHERE table1.id = table2.fk_id
```

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

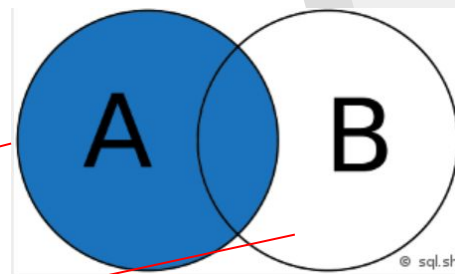


Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT *
FROM utilisateur
LEFT JOIN commande ON utilisateur.id = commande.utilisateur_id
```

```
SELECT *
FROM table1
LEFT JOIN table2 ON table1.id = table2.fk_id
```

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

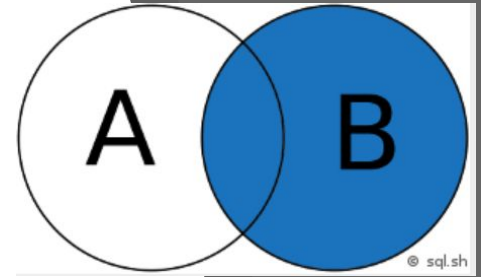
Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture
FROM utilisateur
RIGHT JOIN commande ON utilisateur.id = commande.utilisateur_id
```

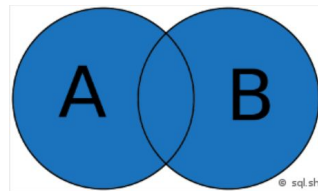
id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
NULL	NULL	NULL	5	2013-03-02	A00107

```
SELECT *
FROM table1
RIGHT JOIN table2 ON table1.id = table2.fk_id
```



**Table utilisateur :**

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1



**Table commande :**

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture
FROM utilisateur
FULL JOIN commande ON utilisateur.id = commande.utilisateur_id
```

```
SELECT *
FROM table1
FULL JOIN table2 ON table1.id = table2.fk_id
```

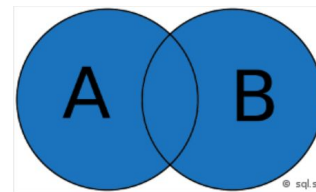
id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
4	Luc	Rolland	NULL	NULL	NULL
NULL	NULL	NULL	5	2013-03-02	A00107

Table "utilisateur" :

user_id	user_prenom	user_ville	pays_id
1	Jérémie	Paris	1
2	Damien	Montréal	2
3	Sophie	Marseille	NULL
4	Yann	Lille	9999
5	Léa	Paris	1

Table "pays" :

pays_id	pays_nom
1	France
2	Canada
3	Belgique
4	Suisse



```
SELECT *
FROM utilisateur
NATURAL JOIN pays
```

```
SELECT *
FROM table1
NATURAL JOIN table2
```

pays_id	user_id	user_prenom	user_ville	pays_nom
1	1	Jérémie	Paris	France
2	2	Damien	Montréal	Canada
NULL	3	Sophie	Marseille	NULL
9999	4	Yann	Lille	NULL
1	5	Léa	Paris	France

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39

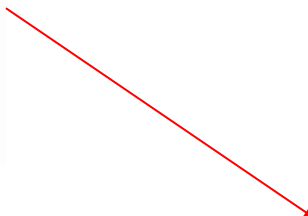
prenom	nom	ville	date_naissance	total_achat
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

```
SELECT * FROM magasin1_client
UNION
SELECT * FROM magasin2_client
```

```
SELECT * FROM table1
UNION
SELECT * FROM table2
```

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39
Marie	Leroy	Lyon	1982-10-27	285
Paul	Moreay	Lyon	1976-04-19	133

id	nom	ville	date_inscription	nombre_achat	id_dernier_achat
1	Marie	Paris	2010-04-22	5	24
2	Louis	Marseille	2011-08-18	3	36
3	Paul	Lyon	2011-11-02	0	NULL
4	Léon	Paris	2012-09-01	1	7
5	Paul	Nantes	2013-01-10	0	NULL



```
SELECT COUNT(*) FROM utilisateur
```

```
COUNT(*)
```

```
5
```

```
SELECT COUNT(nom_colonne) FROM table
```

id	nom	ville	date_inscription	nombre_achat	id_dernier_achat
1	Marie	Paris	2010-04-22	5	24
2	Louis	Marseille	2011-08-18	3	36
3	Paul	Lyon	2011-11-02	0	NULL
4	Léon	Paris	2012-09-01	1	7
5	Paul	Nantes	2013-01-10	0	NULL

```
SELECT COUNT(*) FROM utilisateur WHERE nombre_achat > 0
```

COUNT(\*)

3

```
SELECT COUNT(nom_colonne) FROM table
```



# EXERCICES

Avec les tables `villes_france_free` & `departements`, trouver la requête SQL permettant d'effectuer la demande suivante :

Obtenir la liste des 10 villes les plus peuplées en 2012

```
SELECT *  
FROM `villes_france_free`  
ORDER BY `ville_population_2012` DESC  
LIMIT 10
```

Avec les tables `villes_france_free` & `departements`, trouver la requête SQL permettant d'effectuer la demande suivante :

Obtenir la liste des 50 villes ayant la plus faible superficie

```
SELECT *  
FROM `villes_france_free`  
ORDER BY `ville_surface` ASC  
LIMIT 50
```

Avec les tables `villes_france_free` & `departements`,  
trouver la requête SQL permettant d'effectuer la  
demande suivante :

Obtenir la liste des  
départements d'outres-mer,  
c'est-à-dire ceux dont le  
numéro de département  
commencent par “97”

```
SELECT *  
FROM `departement`  
WHERE `departement_code` LIKE '97%'
```

Avec les tables `villes_france_free` & `departements`,  
trouver la requête SQL permettant d'effectuer la  
demande suivante :

Obtenir le nom des 10 villes les  
plus peuplées en 2012, ainsi  
que le nom du département  
associé



```
SELECT *  
FROM `villes_france_free`  
INNER JOIN departement ON  
departement_code = ville_departement  
ORDER BY `ville_population_2012` DESC  
LIMIT 10
```

Avec les tables `villes_france_free` & `departements`,  
trouver la requête SQL permettant d'effectuer la  
demande suivante :

Compter le nombre de villes  
dont le nom commence par  
“Saint”

```
SELECT COUNT(*)  
FROM `villes_france_free`  
WHERE `ville_nom` LIKE 'saint%'
```

Avec les tables client, commande et commande\_ligne,  
trouver la requête SQL permettant d'effectuer la  
demande suivante :

Récupérer les noms et  
prénoms des clients avec leur  
références de commandes  
ainsi que le détails des  
produits

```
SELECT * FROM client INNER JOIN commande  
as c ON client.id = c.client_id INNER JOIN  
commande_ligne as cl ON c.id =  
cl.commande_id
```

Avec la commande\_ligne, trouver la requête SQL permettant d'effectuer la demande suivante :

Mettre à jour la colonne  
“prix\_total” en fonction de la  
quantité et du prix unitaire

```
UPDATE `commande_ligne`  
SET `prix_total` = (`quantite` *  
    `prix_unitaire`)
```